Usability and Security in Medication Administration Applications

Master of Science in Technology
University of Turku
Department of Information Technology
Networked Systems Security
2016 June
Ranjita Pradhan

Supervisors:
Seppo Virtanen
Annamari Soini (ÅA)
Johan Lilius (ÅA)
Advisor:
Natalia Díaz Rodríguez (ÅA)

UNIVERSITY OF TURKU
Department of Information Technology

RANJITA PRADHAN: Usability and Security in Medication Administration Applications

Master of Science in Technology, 108 p., 9 app. p.

Networked Systems Security
June 2016

The traditional process of filling the medicine trays and dispensing the medicines to the patients in the hospitals is manually done by reading the printed paper medicine chart. This process can be very strenuous and error-prone, given the number of sub-tasks involved in the entire workflow and the dynamic nature of the work environment. Therefore, efforts are being made to digitalise the medication dispensation process by introducing a mobile application called Smart Dosing application. The introduction of the Smart Dosing application into hospital workflow raises security concerns and calls for security requirement analysis.

This thesis is written as a part of the smart medication management project at Embedded Systems Laboratory, Åbo Akademi University. The project aims at digitising the medicine dispensation process by integrating information from various health systems, and making them available through the Smart Dosing application. This application is intended to be used on a tablet computer which will be incorporated on the medicine tray. The smart medication management system include the medicine tray, the tablet device, and the medicine cups with the cup holders. Introducing the Smart Dosing application should not interfere with the existing process carried out by the nurses, and it should result in minimum modifications to the tray design and the workflow. The re-designing of the tray would include integrating the device running the application into the tray in a manner that the users find it convenient and make less errors while using it.

The main objective of this thesis is to enhance the security of the hospital medicine dispensation process by ensuring the security of the Smart Dosing application at various levels. The methods used for writing this thesis was to analyse how the tray design, and the application user interface design can help prevent errors and what secure technology choices have to be made before starting the development of the next prototype of the Smart Dosing application. The thesis first understands the context of the use of the application, the end-users and their needs, and the errors made in everyday medication dispensation workflow by continuous discussions with the nursing researchers. The thesis then gains insight to the vulnerabilities, threats and risks of using mobile application in hospital medication dispensation process. The resulting list of security requirements was made by analysing the previously built prototype of the Smart Dosing application, continuous interactive discussions with the nursing researchers, and an exhaustive state-of-the-art study on security risks of using mobile applications in hospital context. The thesis also uses Octave Allegro method to make the readers understand the likelihood and impact of threats, and what steps should be taken to prevent or fix them. The security requirements obtained, as a result, are a starting point for the developers of the next iteration of the prototype for the Smart Dosing application.

Keywords: medicine dispensation, usability, security, smart dosing, e-health

Contents

List of Figures

List of Tables

1 Introduction

E-health is a new term needed to describe the combined use of electronic communication and information technology in the health sector; the usage of digital data in the health sector – transmitted, stored and retrieved electronically – for clinical, educational and administrative purposes, both at the local site and at distance [8].

With the e-health technology, a variety of objectives are achieved, including increased access to medication-related and health-related information, improved ability to diagnose and track diseases, as well as timelier and more actionable public health information. The reason for making health applications available on portable or mobile devices is to make data or service available anytime and anywhere, at the discretion of the individual. This entails ensuring the right person, the right information, the right place, and the right time. Usually e-health services are accessed by the health worker through devices such as smartphones, PDAs, laptops, and tablet PCs.

Over time, the computational power of smartphones has improved. Moreover, portability, larger screens, higher resolution, and larger on-screen keyboards have attracted health care practitioners' attention. These factors have also contributed to the immediacy and convenience of services, thus making the services less stressful and less prone to errors.

Many e-health applications focus on health surveys, surveillance, patient records and monitoring [9]. However, this thesis concentrates on a drug dispensation mechanism used by nurses in hospitals. The project I am working on is based on research and a study which

suggest that electronic health-care applications can improve the medication tray-filling process at hospital wards [10] [11]. The new design of the smart medication management system, a supportive part of my thesis work, has been documented in a forthcoming article [12].

Healthcare applications demand accuracy, reliability, usability, and most importantly, security of medical data. Accuracy and reliability help physicians make critical decisions based on the information on the application. In case of this application, misinformation can lead to overdosing or under dosing of certain medicines, or even giving entirely wrong medicine to a wrong patient.

The life of a patient may depend on the integrity and availability of the data. This necessitates a thorough security requirement analysis consisting of identifying the vulnerabilities, evaluating the threats and determining the risks. In order to ensure the availability of the services to be used, the confidentiality and the integrity of information, the methods, techniques and tools need to be identified. The security requirements, when integrated at the design phase, allow vulnerabilities to be identified during the initial phase of the development process. The development of a security policy must be done at the same time as the functional design stage, and that must integrate, at the same time, the functional and security specifications.

The research project, as well as this thesis, is about developing a mobile application that helps in saving time and reducing the proneness of making errors in the complex nursing procedures, such as filling the medicine tray and dispensing the medicines to the patients. The project consists of gathering requirements from nurses, designing a medicine tray that could integrate a tablet computer in the process of filling and dispensing, and developing the e-health application to run on the tablet computer. The scope of this thesis consists of the application's usability, security, and privacy. Can they all be achieved simultaneously, or is there a trade-off and a need to compromise? Does increasing information security lead to decreasing usability?

1.1 Goal of this thesis

The goal of this thesis is to offer a detailed security requirement analysis for the Smart Dosing application from both usability and technological perspective. It involves identifying errors at various stages of medication dispensation workflow, analysing the security loopholes in the previously built prototype of the Smart Dosing application, and re-designing the system and the application accordingly. It also involves analysis of vulnerabilities and threats of the technology choices made, and risks of using mobile application in a hospital environment. Based on the above analysis, a list of security requirements was obtained that should be taken into account while designing the system and developing the software application. The implementation of the results obtained will be carried out as a part of the future work. The goal of this thesis is motivated by the criticality of this application, sensitivity of the information carried, and the fact that mobile applications are inherently insecure.

This work will benefit the developers of the Smart Dosing application in understanding the requirements and the design, and help them securely develop and deploy the application. This thesis guides the readers in estimating the overall probability and severity of harm that could result from using the Smart Dosing application.

1.2 Thesis structure

The thesis is structured in a top-down manner where the reader is given an overview of where this work fits, and explanation is provided in the context of this work. The chapters in this thesis are structured as follows:

Chapter 2 provides the reader with background about usability – which aspects are more applicable to the work at hand and how. It discusses how usability varies with respect to the tasks being performed and the environment of working, by detailing usability of different devices of varying sizes and specifications. This chapter also emphasises

how the study of usability assists the developers in developing secure-usable products for healthcare applications.

Chapter 3 gives the reader sufficient theory and background on security by discussing critical properties of security, available security defenses, and various strategies to address security threats. This chapter concentrates on threats, vulnerabilities and risks involved on various levels of mobile computing environment, followed by security requirements in healthcare applications running on mobile devices.

Chapter 4 introduces readers to the medication administration process currently existing in the hospitals and the problems related to it. This chapter mostly emphasises medication dispensation errors in hospitals, the risk factors involved in it, and how information technology can help reduce them. This chapter explains the need and the emergence of Smart Dosing application.

Chapter 5 gives the reader a list of security requirements from both the usability and the technological perspectives, concentrating on various stages of the application development. This chapter identifies and lists the possible human errors at every step of the Smart Dosing application's usage, followed by what modifications need to be made in the hospital workflow on incorporating the application.

Finally, Chapter 6 provides the reader with a conclusion and a prioritized list of future work, along with technological suggestions that would make Smart Dosing even smarter.

2 Usability considerations while using hand-held hospital applications

2.1 Introduction

This chapter begins with answering the question “What is usability?”, followed by distinguishing between broad and narrow approaches to usability.

The International Organization for Standardization (ISO) defines usability as ”the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use” [13]. This definition is directly derived from the broad, original sense of the term.

The broad approach to usability focuses on “quality of use”, whereas the narrow approach is complementary and mostly concerned with the design features of the product. These features are prerequisite for quality of use. Quality of use is considered to be the major design objective for an interactive product answering to questions like whether the product allows the intended users to achieve the intended tasks. This relates usability to business objectives and elevates usability from an optional extra to the prime design goal.

In an academic sense, usability has its roots in psychology, human factors and ergonomics, showing that it can be considered a design-related term. What makes usability different from the other fields of design is that it focuses on human issues. This is the narrow product-oriented view of usability which suggests that usability can be designed

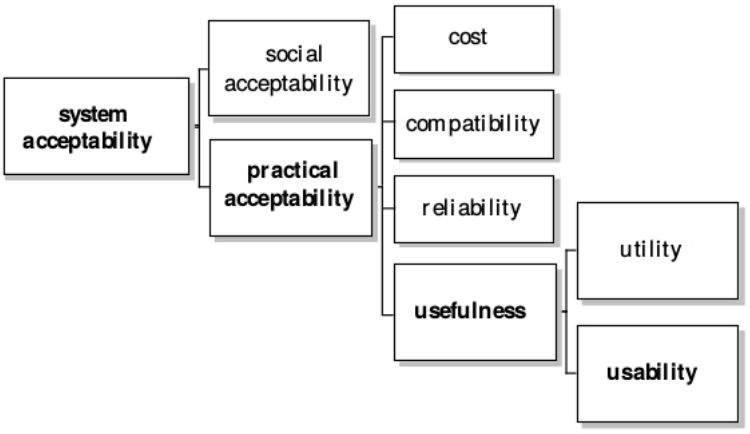

Figure 2.1: Usability as ease of use [1].

into a product. In this sense, usability is closely related to ease of use [1].

Among software engineers, the usual goal of quality requirements is to build a software product which meets the specifications. However, this alone is not very sufficient to ensure the quality of use – that the product can be used for its intended purpose in the real world.

The usability of a product depends on the particular group of users, the kind of tasks they perform, and their working environment. A framework shown in Figure 2.1 of system acceptability has been designed by usability consultant Jakob Nielson. In Figure 2.1, usability is considered as a part of usefulness, nesting it within practical acceptability, which is seen as a part of system acceptability [1].

The definitions of usability vary from being product- and user-oriented to being ergonomic and ease-of-use-oriented. Shortly summarised, usability is an important attribute that assists the developers in the design process to develop a software with a compelling user interface that is easy to use, thus allowing the users to achieve specific task oriented goals with effectiveness, efficiency, and satisfaction.

Usability depends on the user interaction with the product or system, and being a non-functional requirement it can be accurately measured by quantifying user performance, satisfaction, and acceptability. These behavioural data tell whether there is a problem in

usage. Hence, how to fix the problem depends on the way the data are interpreted. As usability is often associated with functionalities of the product or system, user, task and environment, a change in any of these attributes can produce a change in usability. It is the attributes of the product that determine if a product is usable for a particular user group, task, and environment.

In *Usability Handbook*, there are mentioned five reasons as to why so many hi-tech products end up being so hard to use [14]. Firstly, the development focuses mainly on machines or systems, ruling out the people who are the ultimate end-users of the product. Since the development of a system or product mostly aims at improving human performance in some area, designers should consider the human, the context, and the activity components during the design process. Secondly, the target audiences are expanding and no longer share a similar skill set, aptitude or expectations with the designers of the product. The designers thus have to look from the perspective of wide or all varieties of users, irrespective of their abilities. Since nearly one in five people have a disability [15], it is beneficial to target disabled people in order to aim at a foolproof, simple in design product/system, thus ensuring beneficial usability for all. Thirdly, designing usable products is difficult. It can easily happen that some aspect of usability is ignored. Good usability is truly invisible: if everything is going as expected, improving usability might be skipped. The fourth reason would be that team specialists do not succeed completely in integrating and finalizing the product, thus clashing with the user's expectations. Finally, with the evolution from a workable product to a more elegantly communicating product, the mismatch between the design and implementations is on the rise [14].

The goal of paying attention to usability is to achieve good quality of use. User questionnaires and other user-based evaluation can be used to validate the achieving of usability in terms of effectiveness, efficiency, and end-users' satisfaction [1]. Among software developers, this validation should be done for different iterations and versions of the product. Usability testing is done to measure the ease of use of a product (software

or physical), whereas usability engineering deals with research and design processes that ensure a final product with good usability [16]. If a software company is operating in a new market, the only way to get information on the product's usability is by making user questionnaires and running field studies where users perform in their actual working habitat. If the software developers are in an existing market, they can also test the usability of the designs of their competitors.

It is advisable to ensure usability evaluation for identifying usability issues at different levels, and to consider not only technical points of view, but also human factors to avoid potentially dangerous and unpleasant working situations. The overall satisfaction of a user is a combination of factors including intuitive design, ease of learning, efficiency of user, memorability, error frequency and severity, and even subjective satisfaction.

After introducing the general concept of usability, we shall now discuss usability on the Web, followed by usability in hand-held devices, mobile applications, and healthcare applications running on tablet PC's.

2.2 Usability in the Web

The World Wide Web (WWW) is a complete system of extensively interlinked documents that are written in the HTML language. These interlinked documents are transported using HTTP protocol. The WWW resides on the Internet and is accessible to users via a Web browser. It is primarily used for accessing required information and services through direct search, for browsing new or related information, availing and performing the services specific to certain Web applications, such as placing orders and downloading products. Hence, Web usability can be regarded as the ability of Web applications to perform such tasks effectively, efficiently, and with satisfactory results.

In WWW, the interlinked documents do not use HTTP protocol. HTTP protocol is used to transport the interlinked documents. The documents use (or are written in) the

HTML language.

Web application usability is directly related to home page usability [17], easy reachability of content, the ability to orient and navigate on the Web, finding a way out of an error zone, and feeling in control with respect to hypertext [18]. Web usability is important. According to Online Marketing Institute, the financial fallout from poor website design is enormous:

- 85 % of Internet users abandon a site if its design is poor
- 83 % leave if it takes too many mouse clicks to reach what they want
- 62 % give up when they are looking for an item while shopping online
- 40 % never return to a site if the content is hard to use
- 50 % of sales are lost because visitors can't find the content they desire [19]

Every website has a purpose, and good usability helps the website serve its purpose. In other words, a website is identified with its usability. The more easily, clearly, and efficiently it serves its purpose, the more satisfied the visitors are, and it is more likely that they will revisit the page or recommend it to others.

A comprehensive 25-point check list for regular website usability lined out in [20] might come very handy and useful in a time-constrained project. The list has three sections, namely accessibility, identity navigation, and content.

In order to have good accessibility, a website ought to have a reasonable load-time, adequate text-to-background contrast, easy-to-read font size or spacing, sparing use of Flash and add-ons, appropriate ALT tags for images, and a customized "404 Not found" page. Likewise, for a website to be authentic and genuine, it should have its identity notable on the website. For example, a company logo should be prominently placed, a tagline should be making the company's objective or purpose clear, the home page should be crisp and clear, and there should be an easy and clear path to company information and contact information.

Once visitors get the idea of the authenticity of the website, they look for clear paths to the contents that interest them. Here enters the role of navigation in website usability. Information architecture is a vast topic, but the following would be a few points to be kept in mind. The main navigation should be easily identifiable, the navigation labels should be clear and concise, and the links present should be easily identifiable. There should be a reasonable number of buttons/links, the company logo should be linked to home page, and there should an easily accessible site search functionality.

Finally, content needs to be consistent, organized, and easy to skim through. To ensure that, the following points would be helpful. The major heading should be clear and descriptive, and the critical content should be above the fold (an imaginary line where bottom of computer screen cuts a Web page). There should be consistency in styles and colours, so that visitors know that they are still on the same site. Emphasis or bold should only be used when necessary, or else the attention of the user might waver. The URL's used should be meaningful and descriptive. The HTML page titles should be explanatory so that they do not look irrelevant. The ads and pop-ups, if used, should not affect the content.

2.3 Usability in portable devices

The need and demand for interacting with portable devices while on the move is on the rise. The scope of the usage of mobile applications is extending rapidly. With a growing rate of increase in the processing power of portable devices, the range of services provided by the developers is also increasing. The display resolution and device size, as well as limited processing power, capability, and limited input modalities, have effects on the usability of portable devices.

Websites are not restricted to desktops. They can be accessible by various other devices, with sizes ranging from smartphones, netbooks, and tablets to laptops.

2.3.1 Mobile usability

Mobile website

A mobile website is designed specifically to fit smaller screens and the touch-screen capabilities of smartphones and tablets by formatting or optimizing the contents. It is accessible by any mobile device's Web browser, such as Safari on iOS or Chrome on Android, by typing in the URL or clicking on the link. The website automatically detects the device it is being used on, and redirects the user to the mobile version of the website accordingly.

Mobile application

Mobile applications are computer programs designed to run on smartphones, tablet computers, and other mobile devices. They must be downloaded and installed from application distribution platforms like Apple's App store or Google Play. They reside on the device, hence giving more control to the application. In terms of ease of use, it is easier to use a mobile application where all it takes is one tap, whereas with a mobile website one has to open a Web browser and type in a URL.

Mobile Web vs mobile application

Mobile websites have many advantages over mobile applications. Websites can be rendered across all platforms, they have uninhibited publishing authority, independence from application stores and their policies, and the coverage of a broader audience. Furthermore, website optimization is cheaper than application development. The technology required is easy and mainstream, and only the website content has to be updated when there is a need for update of the content across devices. Also, mobile websites are easily shareable via a simple link. They are also easier and less expensive, as compared to the cost for development of a mobile application, which includes upgrades, testing, compatibility issues, and ongoing development. Applications, on the other hand, need proper support and maintenance after their initial launch.

Some of the disadvantages of the mobile Web are the constant need for Internet connectivity, latency (the delay experience between a request and a response), and navigation. Furthermore, mobile applications at the moment offer quite often better usability for the end-user than mobile websites, as stated by the well-known usability researcher Jakob Nielsen [21] [22]. When the goal is interactive engagement with users, optimizing and designing responsive websites is a good idea.

The computing power of mobile devices has risen high, and so has the extent of use, from mere making calls and sending text messages to sending data to and from nearly anywhere with the help of 3G and 4G networks for data transfer. Despite these advances, Web browsing on mobile devices is considered inconvenient by many.

However, mobile applications are competing with mobile Web, which is now on the verge of decline. When it comes to the Internet connectivity of mobile devices, in 2013, 80 per cent of the time spent was on mobile applications and 20 per cent on mobile Web. In 2014, the time spent on mobile applications was already 86 per cent [23] [24].

In a mobile application development process, user interface design is essential. Mobile User Interface (UI) considers constraints and context, screen sizes and screen densities, and also mobility as outlines for design. Mobile devices have limited computing and communication power, smaller screen sizes, changing contexts, and lesser user attention [25]. Mobile phones have a relatively poor response time, they are less tolerant to complexity, and cause imbalance while making interactions with them [26]. Hence, mobile device interface design is more restrictive as compared to desktop interface design.

Mobile interface design requirements

There is a need for maintaining consistency in the “look and feel” across multiple platforms and devices for the same application. There are situations where the user wants to switch from a desktop computer to a mobile device and look for the same application, or vice versa. Mobile interface elements, such as colour, schemes, and dialogue appear-

ances, must be the same across all other counterparts. One way to achieve consistency could be by creating device independent input/output methodologies and avoiding mobile platform specific methods wherever possible. It is absolutely necessary that the native applications for a device share a common "look and feel". Hence, mobile platforms use their own UI libraries and guidelines. An application developer thus needs to adhere to platform standards, especially for touch-screen devices, as they involve a set of gestures, which is different for each platform.

Apple iOS Human Interface Guidelines propose that factors such as interaction with Multi-Touch screen, displays of different resolutions and dimensions, device orientation changes, and gestures such as tap, flick, and pinch, should be considered while designing iOS applications [27]. At the same time, Google has developed Android user interface guidelines [28], which guide developers to consider the following characteristics during the development and testing of Android applications: touch gestures, the size and location of icons and buttons, contextual menus and their responsiveness, the simplicity, the size and format of text, and certain aspects of messages [29].

In addition to the above, a literature review conducted introduces cognitive overload as an important aspect for mobile usability [30].

Mobile Web application user interfaces could be adopted from traditional Web applications to keep the consistency in the look and feel. But they must be redesigned to emphasize frequently used functions, and to make the most effective use of the screen available and the mobile user interface paradigm. This is an important factor, because the user must be able to operate an application, expecting it to function in a standard way. For example, special user settings (such as default language) or transition from portrait to landscape view during the application execution should be taken into account.

Owing to the limited power of hand-held mobile devices and their smaller memory capacity to save states of past events, restoration of previous states is difficult. In order to reduce short-term memory load, interfaces should be designed such that little or no mem-

orization of how to perform a task is required. Developers are likely to make errors with mobile devices because of the continuous change in context, limited user attention, and rapid pace of events. The smaller size of mobile devices makes it easier to confuse between the buttons. A single button should not have potential to cause harm. It is advisable for mobile applications to rely less on network connectivity and to provide flexible error reversal mechanisms. In time constrained situations, applications, if necessary, should be allowed to be stopped, started, and resumed securely and quickly with no loss of data.

The interface design should aim at multiple and dynamic contexts. Users should be allowed to configure the output based on their needs and preferences, e.g. text size, brightness, noise levels, or weather. Enabling the users to operate the device with a single hand or no hands at all, and having the applications self-adapt to the user's current environment can significantly improve the usability of applications.

The limited screen space can lead to trade-offs between content and user interactions. Providing word selection instead of complete text input can be useful. During times of multi-tasking, when little attention needs to be on the mobile interface, designing for hands-free or eye-free interaction is encouraged. Alternatively, sound or tactile output might be a solution to replace visual displays for relaying information wherever possible. Because of a limited screen size, there is a risk of potential information overload on a single view. So, a better approach would be to present information at multiple levels [31]. This will give users the power of deciding whether or not to interrupt their primary task at hand.

Mobile devices are personal by nature, as they are primarily not intended for being shared among users unless otherwise required. It is important to provide users the choice of changing settings according to their needs or preferences. The usage patterns and the aesthetic sense varies among users. Colours are used in software applications for reasons varying from enhancing display aesthetics to utility ones. In the context of user interface design, colour might be useful in highlighting an item, indicating the user about the status

of the action (as in green for acceptable, yellow for caution, red for error or stop), and establishing logical relations among the elements of the user interface.

Colour vision defect is known to affect people in either of two ways: confusion between red-green or blue-yellow. This defect is a sex-lined trait where more men are known to be affected than women [32]. During the design of user interface elements, colour blindness factor needs to be taken into consideration as well. While custom-designing with colour, significance associated with different colours in various cultures needs to be kept in mind.

A survey on mobile developers [33] concerning development practices of mobile applications revealed that most of the applications being developed are relatively small, engaging one or two developers on an average, for conceiving, designing, and implementing the applications. While the developers adhered to recommended sets of best practices, they failed to use any formal development processes. Also, very little was done when it came to organized tracking of development efforts and metric gathering.

However, with increasing complexity in mobile applications, for example those for business critical or health related use, applying software engineering processes will assure the development of secure, high-quality mobile applications.

Mobile application development issues

Mobile applications development shares common issues with other embedded applications, e.g. integration with device hardware, security, performance, reliability, and storage limitations. Apart from these, what makes mobile devices and the applications different lies in the unique aspects of mobile application development. Mobile applications need to interact with other applications from various sources, including pre-installed software. There is a need for sensor handling in situations like movement of device or numerous gestures on a touch screen. Mobile applications, unlike other embedded systems, are capable of using both telephone network services and the Internet services, i.e. they require

both native and hybrid (mobile Web) applications. They may need support for applications written for various devices running on the same operating systems, and at times also different versions of operating systems. The mobile platforms are open, i.e. they allow change in existing functionalities or adding of features, including installation of "malware" applications. The mobile application developers have no control on the user experience aspects, as they must adhere to user interface guidelines listed by the platforms. Testing mobile Web applications is complex because of transmission through gateways and the involvement of a telephone network, not to forget that the optimization of power consumption is challenging.

Besides functional requirements, mobile applications rely on non-functional qualities, such as performance, reliability, quality, and security. The performance of a mobile application can be summed up with parameters like efficient use of device resources, its responsiveness, and its scalability. The reliability of mobile applications, on the other hand, can be measured in terms of their robustness, their connectivity, and their stability. The quality of a mobile application would depend broadly on its usability and installability.

2.3.2 Tablet usability

A tablet is a device that is classified as a computer which is integrated to a large interactive screen, and fulfils the functionality of both personal computers and mobile phones [34]. These devices also share the benefits of both a smartphone and a computer. However, the design of each device, be it a smartphone or a tablet, has its primary function at its very core.

Developing applications for tablet devices should be considered a separate skill and not a scaled version of phone designs. A tablet PC is not an overgrown smartphone, nor is it a laptop or PC. With an increase in mobile Internet usage and tablet PCs, it is important that the websites are mobile-friendly.

Responsive Web design is the solution to manage one website for various-sized devices, and befitting the user experience of all the devices. A responsive design means a website constructed so that it allows the contents, the images, and the structure of the website to flex and adapt to the size and screen resolutions of the device it is being viewed on. Responsive Web design methodology utilizes fluid grids that ebb and flow with the screen size of the actual device, flexible images and media to keep the content flawless on any resolution, and CSS (Cascading Style Sheet), Media Queries, and JavaScript that allow designs to adapt the content presentation to the devices [35].

Media Queries are a sub-specification of CSS3 that enable different CSS-codes for different media types, or different layout for different devices. They save the users from loading huge websites and zooming in or out to find the required content. They also make websites easier to use by integrating touch screens to help navigation. All in all, they give users better, faster, and smarter user experience, and happy customers to business.

According to Jakob Nielsen [36], tablet applications have various inherent problems that need attention. Those include accidental activation, swipe ambiguity, invisibility, and low learnability. Jakob Nielsen [36] also states that flat and rescaled design are two main threats associated with tablet usability. Flat design means a minimalistic style of interface design, which removes all stylistic choices that give any illusion of three dimensions, and consists of the usage of simple elements, typography and flat colours. Rescaled design, on the other hand, means that either an application which was originally designed for big screens is viewed on much smaller screens, or vice versa. Since flat design is a matter of taste, it is technically easy to get rid of it. The second issue, rescaled design, might be here to stay; it is caused by the somewhat naïve idea that a single design is good, as long as it adapts to the available screen space.

2.4 Usability in devices running medical applications

Tablet computers have found their place in various sectors, including the medical field. The versatile and useful features of tablet computers, such as high portability, increased processing power, touch-screens with increased display resolution, quality cameras, and wireless connectivity, have given tablets a place in healthcare systems [37]. The studies in [38], [39] and [40] suggest how useful and beneficial portable devices and applications are, and how tablet computers have found their place in medical school curriculum, during various surgeries, printing patient information handouts for patients, assisting surgeons through different softwares during post-operative, pre-operative and intra-operative phases, enabling surgical consultation through video conferencing, ward-setting, and so on.

Portable devices, like tablet computers and smartphones, are helpful in improving the coordination of patient care by thorough transfer of information among the physicians, nurses, and allied health professionals [41]. The sources such as [42], [43] and [44] give evidence of the effectiveness of hand-held computers in the field of medicine. The end-users of such applications can have completely different levels of experience, knowledge, and expertise. Hence, it is vital to design a system that can accommodate all these different levels of aptitude and training appropriately. Therefore, while considering development of a secure software system, usability plays a major part. This gives rise to the concept of usable security.

Medical-related software applications call for a careful design in order to input and retrieve accurate and reliable patient data. The continuing use of the system by the users mostly depends on the correct and reliable storage of the medical data extracted by various hospital processes. Efficient and usable user interfaces, in turn, provide more reliable data to improve the quality of electronic health records. Designing and developing a usable software system is indeed a complex and time-consuming task. But no matter how difficult this may seem to accomplish, it has the capability of reducing errors, increasing

efficiency and data security, and saving time. Therefore, for a qualitative service, the importance of usable medical information systems should be realized and practised. Many end-users are facing usability problems, such as learnability, flexibility, and robustness, with different kinds of software interfaces for computer applications, websites, or mobile devices [45]. Based on the above factors the systems are accepted or rejected in medical information systems.

An application may be completely secure, but a faulty usability design is very much capable of rendering it vulnerable. There are studies demonstrating that the lack of usability considerations in the design of medical data management systems can potentially lead to human-computer interaction issues. It may introduce increased complexity in the workflow, and decreased quality of patient case scenarios, resulting in a loss of productivity in both medical practice and research [46] [47]. Applications targeted for mobile devices or tablet computers should use special interfaces. Entering data using tablet PCs can incur more errors, and the target should be on developing usable software that would minimize the error rate while inputting data. Once a medical application is developed, it needs integration with existing medical information systems. It is useful to develop strategies for simplifying the integration phase.

Ergonomic issues need to be equally considered. When the applications are such that they need to be used on a regular basis, the long-terms effects should not be neglected. A study on the trade-off between ergonomic factors and the form-factor of various devices would help to further improve user satisfaction. It should not be so that continuous usage of these devices leaves the user's health compromised.

The lack of visibility of certain information can easily lead to errors in a patient's medication, jeopardizing the health of the patient. Working with mobile devices can elevate these problems, as there is constant change of activities in the real world. In order to insure a high usability while actually being mobile, the user interface must remain relatively simple and minimize the required interactions [48]. It is essential that the human inter-

faces be designed keeping ease of use in mind, so that users routinely and automatically get used to applying protection mechanisms correctly.

In order to build a usable and foolproof mobile interface, five principles should be kept in mind [48]. These five principles are homogeneity, hierarchical organisation, dynamic organisation, context awareness, and indexicality. Homogeneity suggests that the interface should be kept familiar and homogeneous, as applications with a familiar design are good for user acceptance and security [49]. The hierarchical organisation of information allows the users to have a better overview of activities. Hierarchical organisation also helps users to deal with the problem of small displays. However, the deeper the hierarchy goes, the number of interactions to be made by the users increases. So, in order to keep the user's interaction simple and fast, the complexity of hierarchy should be kept optimal. The dynamic organisation of data complements the hierarchical organisation of information. Hierarchical organisation may not ensure all the data to be displayed at once, and some important information may go missing unless a scrolling action is used. In order to minimise the risk of missing information, the dynamic organisation principle suggests capitalizing on the real-time usage of the devices. That is, dynamic organisation of data should be able to optimise the information shown at any time, according to the actions to be performed. The context awareness principle assures that it is possible to present only information that is relevant to a specific situation, by making mobile computer systems aware of the user's contextual setting [50]. By providing context awareness based on the user's situation and context, the information already provided by the context becomes implicit and does not need displaying. Hence, the user's environment becomes part of the interface. Finally, indexicality, the last principle, relies on semiotics theory to advise the use of contextual information to improve the user experience.

3 Security factors to consider while designing life-critical applications

One of the most important aspects in achieving secure software systems in the software development process is known as security requirements engineering. The long-standing credo of requirements engineering reads: ”If you don’t know what you want, it’s hard to do it right” [51]. This statement has particular significance for security requirements, because unless it is known what to secure, against whom, and to what extent, it is difficult to develop a secure system, or to make a substantial statement about its security. In order for a resulting system to be evaluable for its success or failure prior to its implementation, its security requirements need to be effectively defined. And in order to specify security requirements, it is vital to first understand the concepts underlying security engineering, including the concept of security.

Security is the degree to which malicious harm to vulnerable and valuable system assets is prevented, reduced, and properly responded to. It is a complex and important non-functional requirement of software systems. According to Ullman’s observation [52], one way of moving towards a more comprehensive definition of security is to ask what one would be willing to give up in order to obtain more security. Likewise, his statement that ”we may not realize what security is . . . until we are threatened with losing it” is difficult to comprehend. If one has no concept of security, one cannot know whether one is threatened by losing security or not. Inquiry into the opportunity costs (benefit gained

by an alternative use of the same resource) of security is an excellent way to determine the value of security. Security can be applied in various domains, for example, physical security, personnel security, operations security, application security, communications security, network security, and information security. In this thesis, we shall mainly focus on application security and information security.

When it comes to applying security in the world of information systems, security is rarely at the forefront of the stakeholders' concern, except perhaps to comply with basic standards or legal requirements. Hence, work on the requirements has primarily focused on functional requirements, paying little attention to security concerns. The biggest problem, however, is that in the majority of software projects security is dealt with when the system has already been designed and put to operation. So, security solutions have been mainly focusing on providing security defences, instead of solving the causes for the security problems. In addition to that, even if there have been attempts to define security requirements, developers tend to describe design solutions in terms of protection mechanisms, instead of making declarative propositions regarding the level of protection required [53].

With the growing dependence of computer-based systems and network-based applications on various software and software-controlled systems, software security has become an essential issue [54]. The increasing complexities of applications and services in modern day information systems are not only vulnerable to a host of threats but also face a greater risk of suffering from security breaches [55]. Most software-controlled systems are prone to attacks by both internal and external users of highly connected computing systems. This threat of technology-enabled crime has given rise to a growing demand for the creation of new response strategies, emphasising that information security must go beyond technical controls [55]. When it comes to the software applications where the consequence of a security breach may range from extensive financial losses to dangers to human life, the information systems should be secured right from the beginning [56].

In order to prevent software from failing while under attack, the software should be designed with the objective not only of implementing the quality functionalities required for its users, but also of combating potential and unexpected threats. This emphasises the unification of software engineering with security engineering [54].

Application security involves measures taken throughout the application development life-cycle to prevent gaps (vulnerabilities) in the security policy of an application or the underlying system through flaws in the design, development, deployment, upgrade, or maintenance of the application.

Information security is defined as the protection of information assets, and aims at protecting the confidentiality, integrity, and availability of information, whether in storage, processing, or transmission [57]. Information security is achieved by a proper application of policy, education, training and awareness, and technology. Information security includes the broad areas of information security management, computer and data security, and network security [57].

3.1 Basics of security

3.1.1 What does security consist of

In the hospital where the study about Smart Dosing application is being done, security is categorized into information, systems and services security, further broken down to eight sub-processes. The sub-processes are organisational information security, staff information security, software security, data security, usage security, data transfer security, hardware security, and physical security [58]. Although this is a security categorisation of the hospital in question, it coincides with the generic security categorisation, and it is adequate for the purpose of this thesis. Organisational information security provides descriptions of principles and protocols of practical information security arrangements, maintenance, development review, and organising. Staff information security plans out

the information hierarchy among staff, their duties and concerns. Software security is ensured by implementing usability, accessibility, and functionality in a secured way. For data to be secured, all data should be stored securely for a sufficient time under legal regulations. At the same time, the data must be easily accessible, and permanent deletion must be possible if necessary. Usage security targets the novice users of the systems or services. The systems should be developed so that they are secured from stupidity, incapability, mistakes, and human errors. Only authorized usage should be possible. Data transfer security targets portable systems where data are in transit through the networks. The data transfer systems need to be well maintained to ensure the security and integrity of data in transit. Hardware assets are physical entities in a system which need protection. An attacker with physical access to the system is much more likely to cause harm than one who does not. Hardware security can be achieved through good usability, for example, by applying anti-tamper protection or by using physical entry control. Physical security is of most importance, as it includes both data and equipment. It demands a well-managed usage environment in which data and technologies can be secured, for example by deploying safety cameras, secure locks, building design, and ergonomics.

According to Bishop [59], security has three components: requirements, policy, and mechanisms. The requirements specify the security goals of the system or the application, that is, what is expected of enforcing security. The policy defines the meaning of security, that is, what steps should be taken to meet the goals that are set out above. The mechanisms enforce policy by providing tools, procedures, and other ways to ensure that the above steps are followed. These three components exist in all manifestations of security.

Security consists of both developmental and operational elements. The developmental element of security consists of a secure design and a flawless implementation, whereas the operational element of security comprises of a secure deployment of systems and networks.

3.1.2 Key security concepts

The following are the security-oriented concepts that are used throughout the different fields of security. An *asset* is anything of value that needs protection from harm. It can be logical, such as a website, information, or data, or physical, such as a person, a computer system, or another tangible object. An *attack* is an attempt that is intended to cause harm to an asset. Attacks can be either successful or unsuccessful. An *attacker* is an agent (*e.g.*, a human, a program, a process, a device, or another system) that causes an attack with a motivation to cause harm to an asset. A *harm* is a negative impact associated with an asset due to an attack. A *threat* is a category of objects, persons, or other entities posing danger to an asset that may result in one or more related attacks. Threats are always present and can be purposeful or inadvertent. A *security risk* is the probability of harm happening to the assets. A *vulnerability* is a weakness or fault in a system or protection mechanism that can potentially be exploited by threats. An *exploit* is a technique used to compromise the system. An *exposure* is a condition where a vulnerability known to an attacker prevails. *Control, safeguard,* and *countermeasure* are the security mechanisms, policies, or procedures that can successfully counter the attacks, reduce the risks, and resolve the vulnerabilities within an organisation, resulting in improved security. *Defence in depth* is the practice of using multiple security techniques to mitigate the risks. [57] [53]

3.1.3 Critical properties of security

According to Firesmith [2], security is a quality factor (attribute, characteristic, or aspect) that decomposes into a hierarchical taxonomy of underlying quality sub-factors as illustrated in Figure 3.1. The quality factors and the quality sub-factors can be described as follows: *Access control* is the degree to which a system limits access to its resources only to its authorised externals, such as human users, programs, devices, or other systems. *Identification*, *authentication,* and *authorisation* are the quality sub-factors of access control.

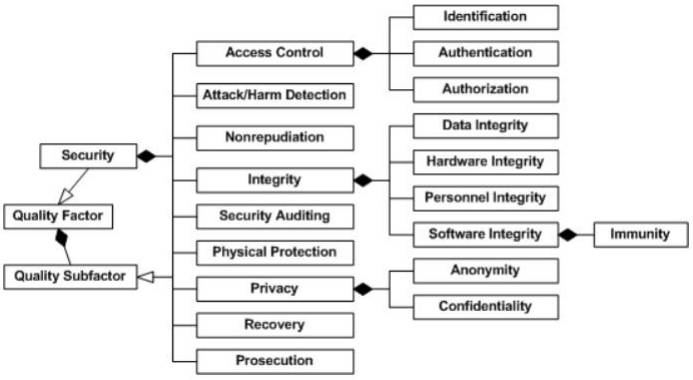

Figure 3.1: Taxonomy of security quality factors and sub-factors [2].

Identification is the degree to which the system recognises its externals before interacting with them. Authentication is the mechanism of a system verifying the identities of its externals before interacting with them. Authorization is the process that governs the system that the authenticated external is permitted to access, by properly granting access and usage privileges. *Attack/harm detection* is the method which detects, records, or notifies attempted or successful attacks (or their resulting harm). *Integrity* refers to the property that ensures the consistency of assets, and does not allow any alteration of assets by intentional or unauthorised corruption. Depending on the components, integrity is applicable to data, hardware, personnel, and software. *Non-repudiation* is the assurance that a party to an interaction (e.g., transaction, transmission of data) is prevented from denying any aspect of the interaction. *Privacy* is the extent to which unauthorized parties are prevented from obtaining sensitive information. The sub-factors of privacy are *anonymity* and *confidentiality*. Anonymity is the degree to which the identity of the users is protected from unauthorized storage or disclosure, whereas confidentiality is the degree to which sensitive information is not disclosed to any unauthorized parties like individuals, programs, devices, or other systems. *Security auditing* is the process of auditing the status and the use of security mechanisms by security-related events. *Physical protection* is the degree to which the system protects itself and its components from physical attacks. *Availability* enables the authorised users, systems, or services to access information without interfer-

ence, within a specified period of time, depending on the request, and to receive it in the required format.

3.1.4 Security attack types and risk management

According to Pfleeger [60], security attack types can be of four types: *interception, interruption, modification,* and *fabrication*. Interception refers to a situation where an unauthorised user has gained access to a service or data, mostly by eavesdropping or illegal copying of data. Interruption refers to a situation where services or data become unavailable, unusable, or destroyed. Denial-of-service attacks can be classified under interruption as a security threat. Modification refers to unauthorised changing of data or tampering with a service, making it different from its original specifications. Examples of modification include intercepting and subsequently changing transmitted data, tampering with database entries, or changing a program so that it secretly logs the activities of its user. Fabrication refers to a situation where additional data, or activities that would normally not exist, are generated.

The attacks can come in the form of malicious code, hoaxes, back doors, a password crack, brute force, a dictionary, denial-of-service and distributed denial of service, spoofing, man-in-the-middle, spam, mail bombing, sniffers, social engineering, pharming, or timing attacks.

Risk management involves three major undertakings that are *risk identification*, *risk assessment*, and *risk control*. Risk identification involves the scrutiny and documentation of security capability of an organisation's information technology, and the risks that organisation is likely to face. Risk assessment determines the degree to which an organisation's assets are exposed or at risk. Risk control refers to the application of controls that would reduce the risks or prevent them. Some of the risk control strategies are *defend, transfer, mitigate, accept,* and *terminate*. The defend control strategy aims at the prevention of an exploitation of a vulnerability by countering the threats, removing the

vulnerabilities from the assets, limiting access to the assets, and providing the protecting safeguards. The common methods of the defend control strategy are application of policy, education and training, and the application of technology. The transfer control strategy aims at shifting risks to other assets that are less valuable, other processes, or other organisations. The mitigate control strategy is an effort to reduce the impact caused by an exploitation of the vulnerabilities through planning and preparation. [57]

3.2 Secure application development cycle

The time when organisations were directly selling the software applications after development to the clients without considering other complexities are long gone. Now with a growing demand for robust applications, secure Software Development Life Cycle (sSDLC) is gaining momentum. sSDLC has been proved beneficial to organisations in identifying and addressing the security issues earlier in the development life cycle and reducing expenses significantly. Based on the individual need, every organisation adds security to their SDLC.

A graphical representation of a sample sSDLC is given in Figure 3.2. The figure consists of major steps that are common throughout the SDLC process, regardless of the type of organisation. The major steps consist of requirements gathering, design, coding, testing, and deployment. During the requirements gathering phase, the security requirements of the software to be developed are analysed and recorded, followed by the process of risk assessment.

The secure application design phase consists of preparing a threat modelling document and analysing the application of adequate security controls. The threat modelling process is comprised of actions such as defining the common usage scenarios of the application and the user types, identifying the application's external dependencies, enumerating the security assumptions about the operating environment of the application, identifying all

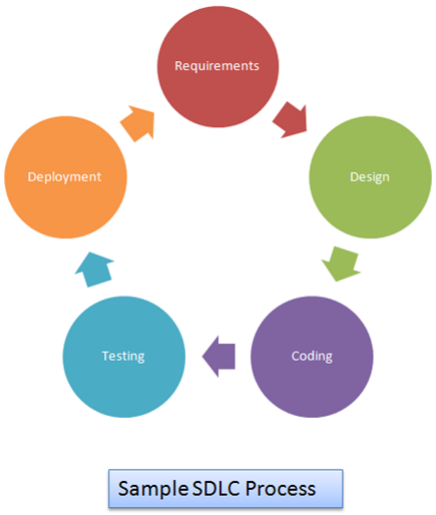

Figure 3.2: Secure software development life cycle [3].

the objects and the interactions, identifying the entry points of the application, determining the threat risks, and identifying potential mitigation for the threats. When it comes to the application of adequate security controls, these may include multi-factor authentication schemes, digital signatures, and transport-layer and data encryption techniques, depending on the type and the value of the assets in question.

The secure application development phase deals mostly with secure coding guidelines and implementing platform specific security features. Some examples of the secure coding guidelines may include secure logging and error handling, following the principle of least privilege, validating input data, implementing secure data storage, avoiding insecure OS features, and so forth.

The secure testing phase includes actions like vulnerability assessment and secure code review. The last, the deployment phase, consists of server configuration review and network configuration review. The deployment can happen via the Web, mobile, or in a cloud. No matter how securely an application has been developed, the deployment strategy of an application decides its success or failure [61]. A software has no real value until it is deployed and put to work. The deployment of an application consists of an installation of the application, the configuration of resources like databases, the

configuration of middleware components, the starting or stopping the components, and the configuration of the installed application for the target environment. All these steps are interdependent, can be recursive, and need to be carried out carefully in the right order [61].

Since security is becoming an increasingly important concern starting from the design phase of the software cycle, the process of security analysis runs parallel with the development. Aberdeen group [62] points out that companies leverage three distinct strategies to address the security threats and vulnerabilities that are latent in their currently deployed portfolios of application software. The strategies include find and fix, defend and defer, and secure at the source.

The find-and-fix strategy makes the application vulnerability scanning and penetration testing solutions to identify the security vulnerabilities in the applications currently in production, to be addressed subsequently by the application developers.

The defend-and-defer strategy concerns enhancing the security of applications currently in production through the use of Web application firewalls or application-level proxies, to reduce or defer the need for security vulnerabilities to be addressed by the developers.

The approach of secure-at-the-source refers to the integration of secure application development tools and practices into the SDLC. Such practice will increase the elimination of security vulnerabilities before the applications are deployed.

The need for the find-and-fix method mostly comes into picture when the software has been developed independently of security requirement engineering activities. This method aims at finding known security problems and attempting to fix them after they have been exploited. The fixing is done by embracing standard approaches such as penetrate-and-patch and input filtering (trying to block malicious input), and by providing value in a reactive way. This approach, however, can be unending for various reasons like software dependencies, or introduction of new flaws. Research experience shows that se-

curity needs to be considered from the beginning of the SDLC in order to avoid expensive rework and reduce potential security vulnerabilities. Fixing an error found in an application after it is deployed costs approximately 30 times more than addressing it during the design phase [63].

The secure-at-the-source method, on the other hand, demands security experts in the SDLC, and can be very time-consuming. Specific security requirements for the targeted system need to be considered at all stages of the sSDLC. Especially, when it comes to the security of critical applications, for example, the applications to be used in a hospital environment, they should be secured at the source, by eliciting security requirements, and integrating the security requirements with the functional requirements. For enhancing the security, the defend-and-defer mechanism should be used.

The causes of insecure software depend on different factors such as making errors while coding, not following secure coding standards and security requirements policy, and integration with vulnerable software libraries. Insecure coding practices are the reason for a large number of vulnerabilities in the applications. Other factors that could result in an insecure software are not adhering to secure code review, skipping the security testing process, and not providing formal secure code training or awareness for software developers [64]. Many common security coding bugs and design errors can lead to reliability issues in the form of buffer overruns, integer arithmetic bugs, memory exhaustion, referencing invalid memory, or array bounds errors. All of these issues have forced software developers to create security updates [64].

3.3 Why securing mobile devices is important

Mobile devices are becoming our computers of choice. They not only serve as constant beacons of our activities and our locations, but also provide treasure troves of data to advertisers and criminals. Mobile security is the practice of protecting smartphones, tablets,

laptops, and other portable computing devices, and the networks they connect to, from threats and vulnerabilities associated with wireless computing. Mobile security could also be called wireless security. The smartphones are among the easiest devices to hack, considering that their software is notoriously easy to subvert, the risks are poorly understood, and the systems for device protection are immature and wholly underdeveloped [65]. It has been observed that the more widely a technology is used, the more likely it is to become a target for threats and vulnerabilities. The number of smartphone subscriptions currently present is 2.6 billion globally, and it is estimated to rise up to 6.1 billion by 2020 [66]. As of July 2015, the number of available applications on the Android market were 1.6 million, and 1.5 million on Apple store [67]. Hence, mobile technology is currently the target for varied threats and vulnerabilities. The mobile device operating systems are newer and more insecure than their long-standing desktop counterparts.

Mobile has become the platform of choice, as it is intimate and always present with us. Mobile phones being small and portable, they are are easily stolen and they can reveal sensitive information. Similar to wired security, mobile security boils down to information protection and unauthorised system access prevention. However, it is challenging to implement security in small devices, as they have limited processing power, possess smaller memory capacity, and use unreliable and low-bandwidth wireless networks. Another particular problem is that wireless devices, including smartphones and PDAs with Internet connectivity, were not originally designed with security as a top priority. Therefore, security researchers are now busy developing new technologies and fixing the holes in the existing ones.

A mobile environment faces prominent threats like loss or theft of devices, data interception and tampering, data leakage, malware attacks, vulnerable applications, compromised devices, Web browser exploitation and OS vulnerability, phishing and social engineering, direct attacks by hackers, malicious insider actions, and user policy violations. Malware infection, a threat to mobile application, could come from application

developers who do not take appropriate measures to ensure data security, leaving the data vulnerable to violations of confidentiality and integrity. The weakest link in any IT security chain is the user. The human factor is the most challenging aspect of mobile device security [68].

A primary reason for attacks on mobile devices could be accessing secret information, which in turn could lead the attacker gain monetarily. Besides confidential data, gaining access to computational power of the device opens up opportunities for committing varieties of cybercrime.

The vulnerabilities are errors in design or implementation that expose the mobile device data to interception and retrieval by the attackers. Vulnerabilities are also capable of exposing mobile devices, or cloud applications used from the device, to unauthorized access. Some examples of vulnerabilities include sensitive data leakage (inadvertent or side channel), unsafe sensitive data storage, unsafe sensitive data transmission, and hard-coded password or keys. When there exists a vulnerability (flaw or weakness), security controls should be implemented to reduce the probability of the vulnerability being exploited. On the other hand, if a vulnerability is such that it can be exploited, layered protections, architectural designs, and administrative controls should be applied to minimise the risk or prevent its occurrence.

The threats to mobile phones can be categorised into two classes: soft and hard. Soft security threats are those that affect the data, such as attacks from Bluetooth, Wi-Fi, or infected installed applications, while hard threats are those that affect the physical device itself, such as theft, illegal access, and getting the possession of its Micro-SD memory card. Hard threats are a result of physical attacks, whereas soft threats are caused by logical attacks through a network or other communication interfaces. The logical attacks could come in the form of rootkits, malicious software, and configurations. A rootkit is an application or a set of applications which is stealthy in nature. It not only hides itself but also hides the presence of other applications like virus or spyware, using some lower

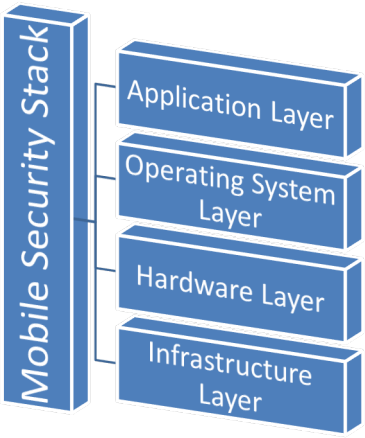

Figure 3.3: Mobile security stack [4].

levels of the operating system. Once the rootkit is installed, it allows the attacker to mask intrusion and gain root or privileged access to the system and, possibly, other machines on the network. Rootkits go almost undetectable by common anti-malware software. They can make their way to the system through a Trojan horse or a suspicious email attachment, or a download of an illegitimate plug-in while surfing the Web.

3.3.1 Vulnerabilities in a mobile device

In order to discover vulnerabilities in a mobile device, and put mobile application risks in perspective, it is important to examine the mobile security stack. The mobile security stack consists of four distinct layers as shown in Figure 3.3. Each layer defines a separate section of the security model of a smartphone or a mobile device. Each layer of the model is responsible for the security of its defining components only. Breaking down security models into well-defined groupings will be helpful in mobile application development and future features development. It will also allow a comprehensive analysis for the designers or developers on how to incorporate or refine security.

Infrastructure layer

The infrastructure layer is the bottom-most, and thus the foundation of the mobile security stack that supports all of the other tiers of the model. The majority of the functional components at this layer are owned and operated by a mobile carrier or infrastructure provider; however, integration into the handset occurs as data is transmitted from this tier upwards. This layer facilitates interception of data over the air. The security of components at this level typically revolves around the protocols in use by the carriers and infrastructure providers themselves. Examples of such protocols include code division multiple access protocol (CDMA), global system for mobile communications (GSM), global positions systems (GPS), short messaging systems (SMS), and multimedia messaging systems (MMS). The flaws and vulnerabilities found at this layer are generally effective across different platforms, different carriers, and multiple device providers [4]. The majority of mobile device manufacturers and phone carriers implement the security software poorly, exposing the device to a plethora of risks.

Hardware layer

Strong mobile device security begins in the hardware. The hardware layer is the domain of physical units. It is generally called firmware and gets upgraded by the physical manufacturer of the handset. The security flaws or vulnerabilities that get discovered at this layer typically affect all the end-users who use a particular piece of hardware or an individual hardware component.

Operating system layer

The operating system layer works in accordance with the software running on a device that allows communication between the hardware and the application tiers. The operating system is periodically updated with feature enhancements, patches, and security fixes. The updates made to the operating system may or may not coincide with patches made

to the firmware by the physical handset manufacturer. Depending on the hardware in use, the operating system may be secured by the same organization as the hardware, or potentially by a third party.

The operating system provides access to its resources through application programming interfaces (API). These resources are available for use by the application layer, and simultaneously, the operating system communicates with the hardware or firmware to run processes, and pass data to and from the device. It is at this layer, and above, that software is the overriding enforcement mechanism for security. Specifically due to the fact that software is relied upon the operating system and the application layer above, it is the most common location where security flaws are discovered. Each mobile platform is known to provide its own security features for device and application security. Most mobile operating systems provide APIs for encryption that can be leveraged in the applications. However, these platforms rely on the application developers to implement some of these application security features, and that is where security issues can creep into applications. [4]

According to a study by Cisco [69], 99% of all mobile malware is targeted against Google's Android mobile operating system. On the one hand, the open-source nature of the Android operating system makes it sell the most, but on the other hand, its openness, and the ability to customise free software, make it vulnerable to security attacks. According to CYREN's security report for 2013 [70], the Android operating system hit an average of 5,768 malware attacks daily over a six month period. Some devices got attacked more than the others. According to a Q1 2014 Mobile Threat Report by the Finland-based security company F-Secure [71], more that 99 % of malware is being designed to target the Android system. This, however, does not leave iOS for Apple iPhones or iPads immune to security attacks. Despite the smaller market share of iOS products, the number of documented vulnerabilities in iPhone or iPad is 82 %, as compared to 13 % in Android, in the year 2013, according to a Symantec report [72]. To get around the

limitations of iPhones like customising keyboards, changing default browsers, or managing files locally, many users ”jailbreak” their iOS devices. While successful jailbreaking gives the users root administrative access to their devices, and control over features that are not officially approved by Apple, it also exposes iOS devices to the same security threats as present in the Android ecosystem.

Application layer

The application layer is the layer that the end-user directly interfaces with while accessing the software. This layer is identified with the running processes that utilise APIs that are provided by the operating system layer [4]. Generally, the security flaws in the application layer result from the coding flaws in applications that are shipped with the mobile device, or have been installed onto the mobile device from application stores. Some examples of the software vulnerabilities that get induced into the application while developing it are buffer overflows, insecure storage of sensitive data, unsuitable cryptographic algorithms, hard-coded passwords, and back-doored applications [4]. If the software is poorly written or contains undetected security vulnerabilities, it can put the data at risk by rendering encryption or other security features ineffective. Chris Wysopal, Chief Technology Officer of Vericode, stated that ”Developers sometimes hard-code cryptographic keys to make it easier to develop the application as they don’t want to reinvent their authentication system when they move to mobile” [73]. A study by Veracode, Inc. found that 40 % of Android mobile applications contain a hard-coded cryptographic key [73]. In addition, coding errors are plentiful in mobile applications, as the tools and frameworks for building them are less mature [73]. The exploitation of application layer security flaws can result in attacks ranging from elevated operating system privilege to ex-filtration of sensitive data [4].

Ex-filtration is defined as the deliberate dissemination of sensitive information from a mobile hand-held device to a third party via common data transmission methods. Ex-

amples of malicious ex-filtration include deliberate transmission of address book data, email, phone log, or SMS; surreptitious transmission of microphone, GPS, or camera data; or ex-filtration via sockets, email, HTTP, etc.

In addition to the vulnerabilities found in the individual layers of the stack, an enumeration of the transitions between layers can disclose additional risk. What is of special interest is the communication of secure data upward on the stack. Depending on the risk tolerance of the organization, one may find that a particular brand or model of mobile device does not have an adequate security model for the sensitivity of the data that one plans to access [4].

3.3.2 Vulnerabilities in a mobile software application

The security vulnerabilities in medical mobile applications can be triggered by errors in code, incorrect logic, or poor design, among many others [5]. Most of mobile security vulnerabilities are linked with *intents* which can be used for both intra- and inter-application communication. An intent is a message containing information on recipient and data [74]. The following are some prevalent vulnerabilities on mobile health applications [5]: Cross-site scripting (XSS) is a vulnerability commonly seen in Web applications. Such vulnerability allows injection of client-side scripts into Web pages, allowing attackers to gain full access to sensitive data and information such as cookies, contacts, and location. This information can then be used by attackers to bypass access controls. Such vulnerability occurs largely because of accepting untrusted user input without properly examining them, followed by using them without sanitizing or escaping them. Mobile websites are prone to XSS attacks as they might have less control in validating or sanitizing user input.

SQL Injection allows an intruder to insert an SQL query from a client into an application, altering the SQL instruction logic. On a successful exploitation of such a vulnerability, an intruder can gain access to SQL servers, allowing them to query, update, or delete data on databases. Such attacks become possible due to the usage of untrusted

and unvalidated user input in crafting SQL queries. In order to prevent SQL injection attacks while accepting inputs through mobile websites, parameterized queries or stored procedures should be used.

Hijacking vulnerability occurs when a malicious application or service intercepts intent meant for a legitimate service. Unless the intent is protected by a permission, this interception can successfully lead to control-flow attacks like denial-of-service or phishing. Hijacking can be done on activities and services [74]. Some activities or services are expected to return response. If a hijacking attack is successful, false response can be sent to the victim by spoofing intents.

Freak SSL/TLS vulnerability is a weakness in some implementations of SSL/TLS. Both SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols that provide communications security. This vulnerability exists across various mobile platforms. It allows intruders to intercept HTTPS connections established between vulnerable clients and servers, and forces them to use weak encryption. On successfully doing so, the attacker steals or manipulates sensitive data.

SSL-stripping is a vulnerability that allows an intruder to perform a man-in-the-middle attack. The SSL-stripping is a technique of downgrading a website from HTTPS requests to just HTTP.

Insecure storage is generally a result of unencrypted sensitive data, caching of information that is not intended for long term storage, global file permissions, or not enforcing platform best practices. The above stated actions can lead to the exposure of sensitive information, privacy violations, and non-compliance.

The Open Web Application Security Project (OWASP) has listed some possible risk factors that affect mobile application security. The risk classification by OWASP, and how the above described mobile application vulnerabilities fit into those categories, has been shown in Figure 3.4.

The mobile software application can be of various types, depending on the develop-

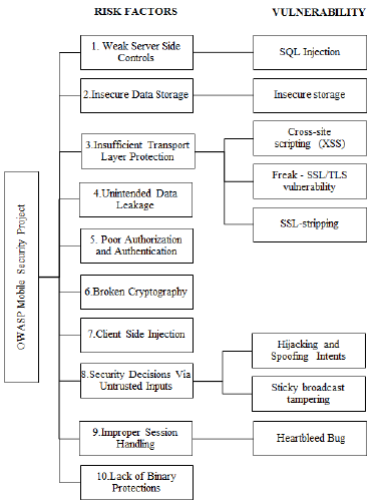

Figure 3.4: Mobile application vulnerabilities and risk factors by OWASP [5].

ment platform used, programming language used to write the software, the APIs used, deployment strategy, and the platforms to be run on. Vulnerabilities in these applications can be examined by understanding and analysing their components. This analysis will give the developers an idea of what kind of security countermeasures or defensive mechanisms to implement when developing HTML5, native, or hybrid applications. These three application types are discussed in greater detail in the following.

HTML5 applications

HTML5, the latest version of HTML, is a combination of various components like XML-HttpRequest (XHR), Document Object Model (DOM), Cross Origin Resource Sharing (CORS), and enhanced HTML/Browser rendering. The evolution of HTML5 has made the Web applications more popular than ever before, as it provides additional capabilities for the developers, and diminishes the performance gap between the mobile Web applications and native applications. HTML5 applications are supported on the three biggest platforms (iOS, Android, RIM).

HTML5 introduces markup and APIs for a complex Web application, making it an attractive development platform in mobile application development technology. These

Web-based mobile applications are challenging the application store model, by allowing the developers to directly interface with the end-users [75]. Since the users can access the Web-based application through a standard mobile browser, this Web-based mobile application addresses the limitations of OS fragmentation. This abstains the developers from developing a version of an application for each platform, saving time and effort. Owing to the similarities with traditional Web applications, mobile Web applications suffer from similar security issues as the Web applications for a desktop or laptop environment, such as cross-site scripting (XSS), SQL injections, nonSSL login, cross-site request forgery, session fixation, and HTTP redirects.

Native applications

Native applications are developed using an Integrated Development Environment (IDE) that provides the necessary development tools for building and debugging the applications. They can fully use the device capabilities, require no Internet connectivity, and can be distributed through the application store. Native applications are more difficult to develop, and require a higher level of experience and technological know-how, than other types of applications. They are written in Java for Android and BlackBerry, Objective-C for iOS, and .NET for Windows Phone. From the end-user perspective, native applications have faster performance, and they provide the richest user experience, along with a consistent look and feel.

Native applications come with their own security risks, in addition to inheriting a few of the vulnerabilities of Web applications. Although the local storage of data in the device speeds the performance of the application, it also introduces a significant risk, if mobile devices are lost or stolen. Other limitations in native applications are that the mobile space is fragmented, and the native applications are tied to a specific platform (such as Apple's iOS, or Google's Android). This diversity imposes severe constraints, such as the use of different development environments, technologies, and APIs for each mobile platform.

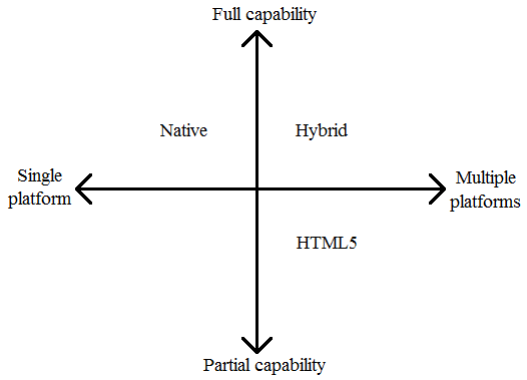

Figure 3.5: Native, HTML5 or Hybrid: Mobile application development options [6].

This OS fragmentation issue exists even within a single OS, as a new version of an OS may not support the old applications.

Native applications make API calls to retrieve data. These API calls need to be authenticated and authorised to make sure that only valid devices, users, and applications are able to access the requested data. The existing security mechanisms used to protect activities on the Web, like Kerberos, Secure Sockets Layer (SSL), and Security Assertion Markup Language (SAML), are not optimized for protecting API calls. When it comes to the security of native applications, vulnerabilities are almost the same for any platform, though the exploitation techniques and tools are different. The attacks that native applications have in common include insecure local data storage, weak SSL implementation, unintended data leakage, reverse engineering, and code injection.

Hybrid applications

The ultimate goal of cross-platform mobile application development is to achieve native application performance and run on as many platforms as possible. A hybrid mobile application is a Web application which is wrapped inside a native container that provides access to native platform features. As shown in Figure 3.5, a hybrid application has access to all the platform features and full device capabilities. It uses a combination of technologies like HTML, CSS, and JavaScript for developing the application. For

hosting such an application, developers target a WebView hosted inside a native container. This provides developers with access to device hardware capabilities. Various tools like Phonegap, Sencha, RhoMobile, Xamarin, and Appcelerator have made the development of hybrid applications easier. These tools provide a consistent set of JavaScript APIs to access device capabilities through plug-ins, which are built with native code. According to the estimations by Gartner [76], the usage of hybrid applications is going to increase, and they would be used in more than 50 % of mobile applications by the end of 2016. While hybrid applications provide the best elements of both worlds, they also come with combined security risks deriving from both native and HTML5 applications [77]. Hybrid applications have their own design issues. For example, it is possible to maliciously modify a hybrid application's functionality from a remote location to the application on-the-fly. This provides a platform for attacks, such as reverse engineering and Man-In-the-Middle (MIM) attacks. In addition to that, hybrid applications do not allow full access to the platform's security features, and therefore may pose serious security risks to the users. Hybrid applications make extensive use of Web views, leaving the applications exposed to vulnerabilities, if they contain poorly written code. JavaScript injections, a weak SSL implementation, and caching of information are some of the common attacks against hybrid applications.

From the user perspective, a well-designed hybrid application looks very similar to native applications. It is downloaded from an application store, installed on the mobile device, and launched just like a native application. But from a developer's point of view, there is a huge difference in terms of cost, time, and requirement of knowledge of different development technologies. The developers of hybrid applications need to write some code in HTML, CSS, and JavaScript, and reuse it across different devices, rather than rewriting the entire application for each mobile platform. One way to minimize the attack surface is to use a well-accepted framework to build the applications, as the frameworks contain some inbuilt security controls to begin with.

The different security risks posed by the application, whether HTML5, native, or hybrid, can often be mitigated using common security techniques, such as adequate input validation, using only encrypted communication channels, adequately verifying the authenticity of remote systems (using so-called "certificate pinning"), and implementing the principle of least privilege for the application. The principle of least privilege is a practice of limiting access to a system to the minimum level, while maintaining the normal functioning of the application.

Depending on the platform of the chosen device for running the targeted application, the application may inherit the vulnerabilities specific to the platform. In addition to platform-specific risks, there are language inherent risks which are based on common security flaws present in the programming language in which the application is written. Some of these risks may be mitigated by the run-time environment in which the applications are deployed. However, there may still be threats to confidentiality, integrity, or the availability of the application and the data it accesses, due to these kinds of flaws. Some examples of language-inherent flaws are cryptographic flaws, credentials management, code quality, buffer overflows, numeric flaws, and information leakage.

When it comes to the level of security that can be achieved with the above approach, native applications surpass the others, as they can use the device itself to aid security. If an application is native, achieving security is easier [6]. The hassle for a choice between native, HTML5, or hybrid is endless, as each of them has its own advantages and disadvantages. But choosing the wrong one could prove expensive. Hence, it is recommended to analyse one's own requirements extensively, and only then make the decision for the development of the application.

3.3.3 Threats in mobile computing environment

The software on the mobile device consists of an operating system, and an application that is intended to be installed and run. This section explains the mobile threats across all

platforms of mobile operating systems. The phenomenon of threats and attacks on mobile devices is putting both individual's and company's safety in danger. Hence, appropriate and strict measures should be taken, in order to counter them and ensure a safe and secure environment.

Mobile threats are broadly categorised into application-based threats, Web-based threats, network-based threats, and physical threats [78].

Application-based threats

Application-based threats can be software specifically designed to be malicious, as well as software that can be exploited for malicious purposes. Typical malicious behaviours include disrupting regular operations of the system, and collecting sensitive and unauthorised information from a system or a user. Application-based threats can be in the form of malware, spyware, a privacy threat, or vulnerable applications [78]. While malware is a software designed to engage in malicious behaviour on a device, spyware is designed to collect or use data without the user's knowledge or approval. Malware can be in the form of viruses, worms, Trojans, spyware, key loggers, adware, rootkits, and other malicious code [79].

The malicious code is known to install itself on computing devices, through some vulnerability in an application, or by using social engineering techniques to trick the user. It could also be available for free on an application marketplace for users to download. It is designed to seemingly perform as expected, but it secretly uploads sensitive data to a remote server. Such malware, once installed on the device, could obtain sensitive information stored on the device or from other applications, and can send it to untrusted applications or remote malicious entities.

Spyware threats are common to data such as phone call log, text messages, location, browser history, contact lists, email, and camera photos [78]. The threats that result in privacy violation may not be caused with a malicious intention by the applications. How-

ever, they gather and use some sensitive information more than what is necessary for the applications to function, or than a user is comfortable with.

Vulnerable applications are a result of insecure development practices, and contain software vulnerabilities that can be exploited for malicious purposes. Once these vulnerabilities are exposed and exploited, they can allow an attacker to access sensitive information, perform undesirable actions, stop a service from functioning correctly, or automatically download additional application [78].

Web-based threats

Mobile devices are often connected to the Internet and are used to access Web-based services through mobile browsers. The mobile browsers are susceptible to attacks similar to those of traditional browsers, such as cross-site scripting, SQL injection, and cross-site request forgery. The Web-based threats include phishing scams, a party posing as a legitimate service, drive-by downloads, or browser exploits [78].

Phishing attacks use Web pages or other user interfaces to trick a user into providing sensitive information, such as social security number, passwords, and account login information. Since mobile devices have smaller UI screens, the maliciously disguised URLs are difficult to distinguish from the legitimate ones. Even Quick Response (QR) codes can be used for such attacks.

A drive-by download attack is caused by automatically downloading an application to the user's device when the user visits an infected website.

Network-based threats

Mobile devices have multiple ways of establishing connections, such as via cellular network, Bluetooth, Near Field Communication (NFC), or local wireless networks. These connectivity options open the doors for attacks like network exploits, Wi-Fi sniffing, man-in-the-middle, and eavesdropping.

Network exploits take advantage of existing software vulnerabilities in the mobile operating system or other software that operates on local networks, for example Bluetooth or Wi-Fi, or cellular networks such as SMS and MMS [78].

Wi-Fi sniffing compromises the data that get transmitted to or from a device, by exploiting the fact that many applications and Web pages do not follow proper security measures. So, if unencrypted data is transmitted across an insecure local wireless network, those data can easily be intercepted.

In order to mitigate such risks, it is important to ensure that the communications are authenticated and encrypted before transmitting sensitive data.

3.4 Threats and challenges to the security of hospital information systems

In the healthcare industry, worldwide, there is a demand and acceptance for transforming healthcare services into digital-based environment. On transforming to such an environment, the security requirements need to evolve accordingly. ICT-integrated healthcare services produce digitized health data. While digitised patient records deliver greater mobility and accessibility to information, and improve healthcare provider responsiveness, they also increase security risks, if appropriate precautions are not taken. The integrity, availability, and confidentiality of digitized health data become the highest priority for developers as well as end-users. The health IT environment faces risks like unauthorized access, use, disclosure, disruption, modification, or destruction of electronic health information. Major problems being faced while implementing healthcare IT include administrative complications, formatting and usability issues, and errors in interoperability [80].

The use of technology for enhancing healthcare delivery necessitates security and privacy, in order to maintain fundamental medical ethics and social expectations. This

includes data access rights; where, when and how data is stored; security during transmission; data analysis rights, and governing policies. While the introduction of technology brought us new ways to store and transmit patient data, it also brought hackers and wrong-doers new ways of misusing that data, creating risks for patient privacy, security, and safety. This created the need for introducing ways of protecting patient data — and the cycle continues.

The hand-held devices, such as tablet computers or smartphones have been widely accepted in healthcare sector, because of being small, portable, and lightweight computers with wireless connectivity. Besides their benefits of portability, safety, and efficiency in the healthcare environments, they come with challenges for privacy and security. A study on privacy and security of healthcare data released in May 2015 [81] provides the findings that 45% of healthcare organizations suffered from a data breach due to criminal attacks, and 12% of the data breach was caused by a malicious insider. This is the first time that criminal attack is the primary cause for data breach, outnumbering the insecure devices and malware infection. A data breach investigation report in 2012 [82] recorded that 51% of organisations have suffered data loss due to insecure devices, and 59% of organisations have experienced an increase in malware infection due to insecure mobile devices. The total amount of stolen records was estimated to be 174 million records in 855 data breaches, and the average cost of a breach was calculated to be 5.5 million USD.

The motivation for securing digitized health data accessed over hand-held devices comes from the fact that they can run a variety of applications across devices which can communicate with each other, and also with the external entities needed by the health applications. Such communications are capable of unauthorised disclosure of patient health data, leading to serious consequences on patients, such as identity theft, blackmail, or discrimination. On the same lines, if the desired health application updates new data into medical records maintained in a remote repository, unauthorised and nefarious updates could potentially corrupt the medical history of a patient, affecting future diagnosis and

No.	Categories of Threat	Description
1	Power failure/loss	• Server down due to power failure • Air-conditioning failure of the server • Interruption by service provider (e.g. Electrical Department & Internet Service Provider)
2	Acts of Human Error or Failure	• Entry of erroneous data by staff • Accidental deletion or modification of data by staff • Accidental misrouting by staff • Confidential information being sent to the wrong recipient • Storage of data/ classified information in unprotected areas by staff
3	Technological Obsolescence	• Outdated Hardware • Outdated application software • Outdated system software • Obsolete network equipment
4	Hardware failures or errors	• Insufficient storage space • Hardware maintenance error
5	Software failures or errors	• Application software failure • Software maintenance error

Figure 3.6: Most critical threats to health information systems [7].

treatment. The most critical threats in a healthcare information system are illustrated in Table 3.6.

The threats against security can be deliberate, accidental, or of environmental origin. Deliberate threats consist of deliberate software attacks like malicious code or malwares, deliberate acts of espionage or trespass, deliberate acts of sabotage or vandalism, acts of theft, compromises to intellectual property, deviations in quality of service, and so on. Accidental threats consist of user errors, software malfunctions, hardware failures, and similar. Threats of environmental origin could arise from a fire or a power failure. Deliberate threats against security can be broadly categorised into threats to confidentiality, threats to integrity, threats to authenticity, and threats to system performance [83].

3.4.1 Threats to confidentiality, integrity and authenticity

In the domain of healthcare and for the purpose of this thesis, confidentiality, integrity, and availability are defined as following: confidentiality means that electronic health information is not made available or disclosed to unauthorized persons or processes. Integrity entails that electronic health information has not been altered or destroyed in an unautho-

rized manner. Availability suggests that electronic health information is accessible and usable upon demand by an authorized person.

An attacker would be motivated to exploit threats to confidentiality for the purpose of gaining access to private information. In case of an e-health service, the main aim of maliciously accessing information is to get access to patients' health data, either when it is generated and transmitted, or stored in the servers. The attacks to confidentiality consist of eavesdropping the communication links, without interfering with the transmissions, or inspecting data stored in the system.

An attacker, on exploiting threats to integrity, will gain the capability of altering the information exchanges within a mobile health service. The types of attacks consist of interfering with transmissions, and modifying the patient data stored in the system. The objective of such attacks is to deliver data to the recipient that are different from the original. The type of information that is subject to these threats is similar to that in the threats to confidentiality. The safeguards to protect against threats to integrity are based on redundancy codes (Message Integrity Codes or MIC) added to the data.

An attacker, by exploiting threats to authenticity, can counterfeit data and deceive the recipient into believing that the data come from an authentic originator. Such attacks consist of forging the part of the data that is in the headers where the originator is identified, affecting both entity identity and data origin. The components of mobile e-health service architecture that may be subjected to authenticity threats are sensors or actuators, a front-end, a mobile terminal, an e-health server, or an end-user application. The safeguards to protect against threats to authenticity are based on shared secrets (Message Authenticity Codes or MAC) or, if authentication before a third party is needed, on asymmetric cryptography. Repudiation is a variant of this type of attack that leads to denying authorship, or the contents of data previously sent. The safeguards applied to authenticity also protect against repudiation. Threats to system performance can come in terms of availability, reliability, and accountability. The denial-of-service attacks fall under this category.

Wireless communication is more vulnerable than its wired counterpart. With the growth of wireless applications in healthcare environments, the security and privacy issues are on the rise. Although most wireless data networks today provide reasonable levels of encryption and security, the technology does not ensure transmission security in the network infrastructure. It is possible to lose data due to mobile terminal malfunctions. Additionally, these terminals could be stolen, and ongoing transactions can be altered. In short, the mobility enjoyed by mobile applications in healthcare also raises many more challenging security and privacy issues. Serious consideration must be given to the issue of security and confidentiality in developing mobile applications for healthcare. [84]

Although real-time monitoring and data transmission facilitate quick access to information, they have the capability of exposing a patient's medical data to malicious intruders or eavesdroppers. If a mobile healthcare system fails to provide the necessary protection when communicating data, the patients' private data gets easily accessible by unauthorised parties or persons; medical records may get freely tampered with by malicious attackers, and false information may get injected into the data stream by a prohibited node. Therefore, when planning mobile healthcare, security is indispensable, owing to the shared nature of wireless devices, the inherent security risks of mobile devices, the mobility of the patients, and the vulnerabilities of pervasive and ubiquitous environments [52].

3.4.2 Security requirements for mobile healthcare applications

Controlling remote communication

Healthcare applications require to communicate with external entities over several channels, such as the Internet or Bluetooth. These channels allow transfer of healthcare data to and from the application, and thus need to be secured. However, all the information being transferred may not be sensitive. For example, information being accessed from the healthcare portal need not need protection. A secure establishment of communication

with the healthcare information systems should be sufficient to meet the security needs. The application, however, needs to ensure a fine-grain control over the sharing of the application state by tracking the access made to the sensitive data by the application.

There should be a policy enforcement engine that can detect a data flow on both incoming and outgoing channels. Knowledge of the source of incoming data would help in deciding whether the data is sensitive (e.g., information is received from an electronic medical record repository).

For communication on an Internet channel, there should be a security framework which should be aware of all the connections made by the application, and monitor those. The framework should allow communication only with the list of trusted external entities for the application. [85]

Preventing data sharing with other applications

A healthcare application must avoid interacting with other unsafe applications, and retain its functionality to itself [85]. Information sharing between the healthcare application and untrusted applications on the same mobile device could lead to the disclosure of sensitive information. There should be a way to monitor inter-process communication channels and place safeguards to prevent such disclosures.

Controlling insecure data storage

If a health application needs to store data on a smartphone's primary memory or an external memory card, there should be a mechanism for detecting operations performed on files containing sensitive information, and preventing untrusted applications from accessing those files. There could also be a way to prevent the storage of sensitive information on the file system, and to require the application to connect to a remote repository for storing sensitive information. This mechanism would have a higher degree of security, but might be restricted to the application, if the application absolutely needs to store data

on the device. Usually, mobile operating systems provide data protection and a separation between the applications on the device's main memory. However, to ensure security, these existing data protection systems can be augmented to achieve the above needs. External memory on the smartphone device, such as an SD card, can be inserted on other devices reading data stored on the SD card. Also, smartphone devices can be connected to other devices via USB, making the sensitive data accessible. This can lead to health data disclosure or loss of sensitive information. In such cases, either the application should be prevented from storing sensitive information in such memory locations, or explicit user consent should be obtained to sanitize this information before it can be stored at such a location, by asking or warning the user whether or not they should be storing the data, and how sensitive the data are.

4 Errors involved in medication administration process

A medication error is either a deviation from the physician's medication order on the patient's record, or any error occurring in the medication use process. These errors include the prescription of an incorrect dose, documenting incorrect information, dispensation of wrong medicine, administration of a wrong dose of a prescribed medication, and failure of the healthcare provider to administer, or failure of the patient to ingest, a medication. Medication errors are never the result of a single, isolated human error but comprise a chain of events leading to that error.

Medication errors are the most frequent cause of morbidity and preventable deaths in hospitals [86]. A landmark study by the Institute of Medicine (IOM) entitled *To Err is Human: Building a Safer Healthcare System* [68], estimated that about 98,000 patients die each year worldwide due to a variety of medical errors. Gurwitz et al. [87] reported that 38% of medication errors are serious or fatal, and 42% of those could have been potentially prevented by reducing the number of medication errors. According to Yle [88], hospital medical errors claim 700–1700 lives in Finland every year.

Mistakes with medication, like over-medication, under-medication, wrong medication, or wrong dosage of medication, are the most common errors in the hospitals. These errors occur as often as one out of every four times the medications are given [89]. Over the past five years, the electronic reporting systems that are used in Finland's hospital

districts have registered over 340,000 medical errors and patient safety incidents. Drug dispensation for the patients in the hospitals is a potential problem because of the large range of medicines available and chance for drug mix-ups.

4.1 Risks and factors accounting for medication errors

Medication errors can be of many forms, and can occur at various phases of the medication administration process, starting from the prescription to the dispensation of medicines to the patients. Important risk factors during drug administration include insufficient pharmacology knowledge of the nurses, errors in the patient chart or documentation by the nurses, and inadequate pharmacy services. Poor medication labelling, miscommunication among the nurses, a lack of verification of the medicines whenever and wherever needed, an inefficient organisation of the medicine tray, and incomplete medicine prescription are other risk factors accounting for medication errors. Medication administration being the riskiest task performed by the nurses, there have been policies and guidelines for them to help them prevent medication errors. One of the standard recommendations for safe medical practices has been adhering to the "five rights" principle (right patient, medication, dose, route, and time) [90] of the medication use process. Over time, the five rights have been updated by seven rights, which appended documentation and reason. Recently, the seven rights were upgraded to nine rights of medication, as identified by Elliot in [90]. According to the author, following the rights does not guarantee a complete eradication of medication errors, but helps improve and ensure safety and patient care.

However, error-prone situations arise when the nurses have inadequate training in the procedures for medication administration. Hence, to reflect on the responsibility and accountability associated with medication administration, there is a need for in-depth knowledge of medication, such as the pharmacodynamics, therapeutic use, side effects, adverse events, and appropriateness of administering the medication, while taking into considera-

tion the patient's current response to treatment.

Human factor is a common cause accounting for medication errors, but human error can be the result of inadequacies in system design, such as task complexity, error-prone situations, and individual differences. A single medication error is most often due to the convergence of multiple contributing factors. In terms of the individual, several error-producing conditions have been identified to affect the nurses' performance during the medication administration process. These conditions include motivation, fatigue, and stress; depending on the level of these conditions, the performance of the individual nurse can vary from good to poor. Individual differences in the propensity for human error are related to the nurses' abilities, attitudes, susceptibility to stress, and lack of experience during medication administration. Other contributing factors are workplace stress, distractions, interruptions, inadequate training, and fragmented information. In order to avert the medication errors due to human factors, nurses must play a major role in the design and implementation of any new proposed drug administration system. This will not only ensure a smooth transition to a new system, but will also result in an accurate and efficient administration of medication.

Information technology can help in reducing some medication errors through an eradication of transcription and dosing errors. Harnessing the potential of information technology can reduce the rate of errors, by preventing errors and adverse events, by facilitating a rapid response on the occurrence of an error, and by tracking and providing feedback about possible contextual errors. Research suggests that implementing information technology has been proved successful in reducing the frequency of errors of different types in different hospital settings, and also the reducing frequency of the associated adverse drug events [91] [92]. A research conducted by Díaz Rodríguez et al. [10] in two university hospitals located in Finland suggests that the medication errors and incidents are preventable, and that patient safety can be increased with the help of a task-driven digitalized mobile application.

4.2 Medication errors in the hospital workflow

Based on the research done by Díaz Rodríguez et al. [10], the tasks carried out by nurses every day in hospital wards, such as filling the medicine tray and dispensing the medicines to the patients, are time-consuming, stressful, and painstaking. As a solution to the stated problems, a step-by-step mobile application was suggested that would effectively reduce the time, make the tasks more convenient, and increase the safety during the process of filling and dispensing medicines. Such an application, called *Smart Dosing*, was later developed by Khan et al. [11]. The Smart Dosing application aims at integrating different information systems into a single application. The application aims at accessing patient records, accessing the nurse shifts from hospital information system, and connecting to the *Pharmaca Fennica* [93] database (a medication and equivalences database from Lääketietokeskus) for retrieving equivalence for a given medication.

The work of Khan et al. [11] presents the architecture and functionality of the new Smart Dosing tablet application, as well as a survey conducted among Finnish hospital nurses about the application's usability. However, the only existing prototype version of the application was never tested in real-life treatment cases. Khan et al. [11] have pointed out in their work that "in a future pilot test [...] we expect to reduce the nurses' time spent on dealing with medications." This goal is indeed relevant and worth aiming at, but the application itself turns out to have quite a few issues that need to be solved before one can focus on saving time and increasing safety.

The prototype developed by Khan et al. [11] was developed using the Vaadin framework. The version of the Vaadin framework used was Vaadin 6.8. Since the application was intended for running on a tablet device, Vaadin touch-kit was used. However, since Vaadin has been upgrading its framework continually, the deprecated files and functions from Vaadin 6 that were not outdated, made it challenging to keep the application running and responsive.

During the intervening time, the design specifications kept evolving with respect to the

requirements and inconvenience learned from the first prototype. The tablet device was thought to better suit the need of the nurses if placed on the tray. With the new orientation of the tray and device, the user interface design of the application was modified to meet the changes. The updated design specification and description of the application is attached in appendix B for reference. Before proceeding with the development of the new design of the application, a security analysis of the software application, the design and layout of the tray, and the workflow was prioritised.

Nurses are involved in the entire medication administration process, including prescribing, documenting (transcribing), dispensing, administering, and monitoring, except for prescribing medicines to the patients [94]. Since nurses play an essential part while dealing with patient medication, and are the last persons who can verify whether the correct medication is prescribed and being dispensed, they can be considered as administrators of safety. The role of nurses, while prescribing and documenting medication, along with the entire workflow of the medication administration process, is described in a study conducted by Pirinen et al. [94]. While medication errors in the hospital setting occur at all stages, the probability of errors is more frequent during administration, prescription, and dispensation [94]. A study conducted in the U.S indicates that 11% of medication mistakes are prescribing errors, while administration errors account for 40% of such mistakes [95].

Since medication errors are strong risk factors for preventable adverse drug events or reactions, strategies must be used for their reduction. The strategies include the pharmacological knowledge of nurses, pharmacists, and physicians, engaging a sufficient number of pharmacists on the wards, and the computerization of the entire medication process. For a thorough security analysis, the potential errors should include the entire range of severity, from trivial errors to life-threatening errors, such as overdosing, underdosing and clarification of medication prescriptions.

The most frequent and possible errors, while filling the tray and dispensing the medicines,

Errors while filling the medicine tray	
Reasons/context	**Solutions**
The medicine names in the printed medicine charts are written according to trade names and not pharmacological names	Integration with Pharmaca Fennica [93] system for having both options
The limited size of the medicine chart makes the medicine information illegible and difficult to track their names while filling them one row after another	Electronic charts
The search for equivalent medicines is time consuming and sometimes confusing, owing to the large number of available substitutes	Integration with Pharmaca Fennica system
Disconnection between the medicine software and the printed medicine chart	Ubiquitous and real-time information update via a tablet application
Difficulty in identifying and verifying the medicines while filling the tray	Integration with the Pharmaca Fennica system enabling pictorial information
Dealing with unfamiliar medicines (new drugs, seldom used drugs, new packages etc.)	Integration with the Pharmaca Fennica system enabling pictorial information
Reminders for injections, liquid medicines	Notifications or reminders

Table 4.1: Errors while filling the medicine tray

Errors while dispensing from the tray	
Reasons/context	**Solutions**
A paper medicine chart limits the tracking of the medicine dispensation process by making the tracking inefficient	An IT application and the new tray design together make the tracking more visual and user-friendly
A large number of medicines scheduled at peak times increase the risk for human errors	An IT application with a task-driven digitalized patient medicine list guides the user step-by-step
Difficulty in identifying and verifying the medicines while dispensing the medicines from the tray	Integration with the Pharmaca Fennica system enabling pictorial information
Difficulty in keeping up with the patients' changing treatment and the patients' location	Ubiquitous and real-time information update via a tablet application, enabling better communication
Improper or no reminders for the limited-access medicines	Notification for controlled drugs

Table 4.2: Errors while dispensing medicines from the tray

as well as the solutions that an IT application can provide, are provided in Table 4.1 and Table 4.2. The complexity of the medication administration process, coupled with the need for nurses to simultaneously deal with multiple tasks in a hectic atmosphere, has the potential to make the nurses more error-prone. Studying and analysing the current medication administration systems in hospitals led to a list of the wishes of the nurses to facilitate the process [94]. The nurses expressed their need for support at all stages of the medication administration process. One of the most important wishes was a real-time update of prescriptions so that the nurses are informed about new or changed prescriptions. The nurses should be able to document prescriptions ubiquitously, regardless of time and place. The nurses also demand a way to verify the medicines during the filling of the tray, and before administering the medicines to the patients. The medicine software is also desired to have precautions for the nurses, and collective directions, like adhering to the unit policies, while entering the medicine dose for the patients. The nurses demand a software where all important information, such as patient records, medication history, prescription dates and expiration dates, and unscheduled or need-based medicines, is integrated. Assistance with finding, identifying, and verifying medicines or equivalent components would reduce the time, effort, and confusion of the nurses, resulting in reduced errors. Nurses call for a clarity in the medicine chart list, the lack of which is a big barrier for safe medication.

Based on the wishes of the nurses expressed above, the probable errors identified in the workflow and the issues in the previous prototype, a new design was developed, keeping better security and usability in the forefront. The description of the new design of the Smart Dosing management system is attached in appendix B for reference. The smart medication management system includes a medicine tray with a tablet device placed in the centre of the tray, with cup holders on both sides of the tablet, as shown in Figure B.2. The cup holder is a new feature in this design, overcoming the dysfunctions of the cups, and allowing a controlled and sequential workflow. One cup holder carries medicines for

a single patient. The colour of the cup represents the medication timing. For example, a red cup is for morning medicines, a yellow cup for afternoon medicines etc. The entire description of the new design can be referred from appendix B.

5 Security requirements analysis of Smart Dosing application

Provision of healthcare takes place in fluid and complex environments, where the ability to deliver patient care is dependent on the quality and availability of patient data, as well as on the facilities and skills of the healthcare provider. The introduction of mobile healthcare applications in the hospitals has reshaped the patient care. However, besides the opportunities provided by the healthcare applications, these also come with various challenges, such as their interoperability, usability, standardisation, adoption, and maintenance of privacy and security.

The difficulty in adopting a healthcare application lies in the differences in the languages, cultures, and motives of the users, and operational constraints of all stakeholders (hospital management, medical staff, designers and evaluators of healthcare applications and services). The adoption of a hand-held device running a medical application faces not only usability and security issues, but also resistance from clinicians in adopting and accepting IT solutions. The inclusion of IT-solutions in the hospital workflow requires redesigning the workflow.

The motivation for securing health data accessed over hand-held devices comes from the fact that the devices are inherently insecure, and that they can run a variety of applications which can communicate with each other. The applications also need to communicate with the external entities needed by the health applications. Such data transmission and

communication are susceptible to unauthorised disclosure of patient health or medication information, which could lead to serious consequences for the patients, the nurses, and the hospitals. Therefore, it is essential that the security requirements for the application to be introduced in the hospital are analysed and listed, so that they can be incorporated from the very earliest design phase of the application.

There is no universally accepted definition of security requirements in the literature. Security requirements can include anything from high level goals to detailed specifications of what security measures ought to be used. Security requirements focus on what should be achieved, not how. However, they need to be specific regarding who they concern and when they apply, according to a survey on security requirements [96]. The same survey [96] also points out that even if the application's primary function is not security, its information security should be given importance, and its security features and mechanisms should be made prominent in the user interface. Unlike typical functional requirements, i.e., what activities are expected of an application, security requirements can be highly reusable even across different application domains. The same applies for healthcare-related technical security requirements.

Healthcare applications need to be interactive, interoperable, easy to use, engaging, adaptable, and accessible for diverse audiences [8]. For the error free operation of healthcare technologies, their usability, security, and privacy need to be taken into consideration during the development phase. In case of the Smart Dosing system, the security requirements consist of designing the secure application, implementing security techniques, and defining a secure workflow. In this chapter, the security requirements for the Smart Dosing application are analysed from various perspectives, such as usability, the technology to be used, and the hospital workflow. For the purpose of having an exhaustive analysis of the security requirements for the Smart Dosing system, both the new design and the previous prototype have been taken account of. The issues found while analysing the previous prototype have been listed in appendix A for reference.

A qualitative analysis of the likelihood and impact of the threats and risks is done using Octave Allegro methodology [97]. Depending on the severity of the threats, the impact is analysed, and depending on the impact, possible countermeasures are suggested. The likelihood and impacts range from 1 to 5, where 1 is the least and 5 is the highest in magnitude. The impact factors 4 and 5 are assigned when the threats are irreversible or fatal.

5.1 Security requirements analysis from a usability point of view: designing a secure application

If the security controls and configurations of the applications are too complicated, non-technical medical personnel will not be able to use the application easily, and will try to ignore or circumvent the security features. The users should be trained to lock the device when it is not in use, they should refrain from sharing the device, use a screen shield, log in to a secure Wi-Fi network, and register the device with the settings defined for healthcare applications by that organisation.

The usability of the smart medication management system depends on the user interaction with the Smart Dosing application and the integrated design with the tray as a whole. The frequency or likelihood of making errors while using the Smart Dosing application depends on the user-friendliness of the application.

Before designing the application, it is important to choose the type of device it is going to run on, its hardware specifications, software, and the operating system of the device.

5.1.1 Choosing an appropriate device for the application to run on

The choice of an appropriate device for running healthcare applications is very crucial, as it dictates the available computational power, and constrains or facilitates the visibility and usability.

For example, a device with a large screen size (a 10-inch screen) offers better visibility and allows more and different information to get displayed at the same time. However, because of its weight and size, it is inconvenient to operate the device with one hand, and impossible to carry it in a pocket. A smaller sized device (a 4-inch screen) is convenient to operate with one hand, and it can be carried in a pocket when not in use. Nonetheless, it provides a limited area to display information on.

With each device having its own merits and demerits, it is difficult to choose the best device to be used in a hospital environment. The choice of device depends on certain criteria, such as the usage, the users, and the purpose. However, to make choosing easier, some generic specifications can be taken into account. For example, hand ergonomics, the size of the display screen, the display resolution and aspect ratio. Choosing a device with the best hand ergonomics will allow the nurses to carry and operate the device conveniently. The size of the device screen is an important factor. A small screen size will activate a scrolling feature if and when the amount of information to be displayed exceeds its allocated screen size. This could lead to a risk of missing out on information. In case of the Smart Dosing application, missing out on information implies loosing out on the patients' medication. Display resolution is a parameter that influences the amount of information that can be displayed on the screen. The aspect ratio, as well, plays an important role in choosing an appropriate device. The tablets are typically used in the landscape mode, while smartphones are used in the portrait mode, though they can be used the other way around. Carrying a heavy device all day long can be cumbersome. So, the weight of the device should be taken into consideration. Last but not least, the device's battery must last at least long enough to perform all the daily tasks of the nurse. While designing the Smart Dosing application, it should be borne in mind whether the design best fits the landscape mode, the portrait mode, or whether it can be used interchangeably. If the design should strictly comply to the chosen mode to meet the desired requirements, the undesirable mode must be disabled.

When choosing a device to run the Smart Dosing application, along with the above features, there is a need for finding a fit between the task characteristics, the environment where the device is going to be used, and the users.

5.1.2 Choosing the hardware, the software, and the operating system of the device

The constraints on the usability, such as slow download speed, troublesome input mechanisms, and low processing powers, can be managed by making a proper hardware choice. Choosing a device from the wide range of different platforms and device types available is a major challenge while developing applications for smartphones, tablets, or both. There are currently five major mobile platforms available, Android, iOS, Windows, Symbian, and Blackberry. There exist different carriers with different versions of the same operating system. In addition to that, there exist incompatibility issues among operating systems and devices. Developing applications that can run on all of the platforms mentioned above is very demanding. All the platforms have different software development kits (SDKs), along with different libraries, and different methods to design user interfaces. The programming language also differs for different platforms. For example, Android and Blackberry run on Java, iOS runs on Objective-C, Windows runs on C#, and Symbian runs on C++. For tablet devices, the number of platforms comes down to three, Android, iOS, and Windows. Out of these three, Android and iOS are more popularly used. Android is available on a wide array of devices, including small smartphones, large smartphones, and tablets, with screen sizes ranging from 6 inches to 13 inches. iOS, on the other hand, runs on the iPhone and the iPad. In case of iPad, there are just two screen sizes to choose from. Full-size iPads have a 9.7-inch display, while the iPad mini models have a 7.9-inch one.

Depending on the usage period of the application and the environment, one major drawback the application could face is drainage of its battery. The device must be put in

sleep mode, and the application must use lighter frameworks or libraries for conserving power utilisation. However, the application must be able to restore its interrupted state when the tablet is taken out of its sleep mode or gets switched on. One solution to avoid the interruption of the application due to low power would be continuously charging the device. However, that might eventually damage the battery of the device. From battery point of view, iPAD should come handy because of its battery cycle count, which is calculated based on the use entire battery's power, but not necessarily on a single charge. So, the device can be continuously kept on charge to bring it to 100 percent and prolong its battery cycle. The application should also be able to decide how it would respond if and when suspended, that is, how it would handle multi-tasking.

My recommendation would be that the chosen device for the Smart Dosing application has computation capabilities for IPSEC (VLAN), a well-designed user interface, secure access to the touch-screen, and lower power consumption. The chosen device must have the possibility of installing specialised applications, like the remote wipe software, automatic lock-out, or wipe after failed login attempts. All in all, the design should have easy reachability of content, the ability to orient and navigate, the ability to find a way out of an error zone and to restore successfully from an interrupt state.

5.1.3 Secure functional requirements while filling the tray and dispensing the medicines

A secure functional requirement is a security-related description that is integrated into each functional requirement, stating what should not happen. The security and privacy issues originate from human concerns and intents. For every existing system that is in use, there exists the possibility of it being misused. The primary users of the Smart Dosing application and the device it is running on would be the nurses working in the hospitals. The most common user-based security errors, while operating the smart medication management system, would occur due to a physical misinterpretation of the tray design, and the

complexity in the tray and tablet integration. Therefore, the design should be kept simple and practical in order to limit design and implementation errors. The design should also encourage the mental involvement of the user by not making the application completely automated. A user of a life-critical application must always know what is going on, what direction should be taken next, and she or he should be able to take responsibility if and when the application goes awry. The mental involvement of the nurses should not be requiring them to be pressed or tense, but to keep them focused and directed. For example, the nurses can be compelled to mark the check-boxes alongside the list of patients or medicines after filling or dispensing the medicines. Marking the check-boxes, after filling the medicines in the tray or dispensing them to the patients, would also help the nurses keep track of their progress in the workflow. The design should be open and fairly understandable by the users. While designing the Smart Dosing application, it should also be kept in mind that the security mechanisms should not depend on the ignorance of the attackers, but on the implementation of protecting the assets.

Following is a list of secure functional requirements for the Smart Dosing application:

The orientation of the tablet device in the medicine tray should be properly defined. It should be checked whether there exists only one way of integrating the tablet device to the tray, or whether the orientation can be chosen freely by the nurses. If the orientation affects the patient information or the access to the patient's medicine cup holders, the device must be allowed to fit only in the desired position, in order to curb the mistakes that would be caused due to misorientation. I would recommend that the tray should contain a rubber flap to surround and hold the device firmly, and a small outlet for the charging plug. There could be an arrow mark for the charging outlet so it were clear for the nurses how and where to fix the device onto the tray.

The placement of the medicine cup holders with respect to the tablet device must clearly define the direction of filling and dispensing, in order to ensure that the right cups are being dealt with. Some configurations can allow filling the tray from centre to left, left

to right, or centre to right. In such situations, there should not be any confusion. I would recommend that the filling and dispensing actions take place from left to right, irrespective of which side of the device the cup holders are placed. The left to right direction seems reasonable as most human beings are accustomed to read from left to right. However, this will vary among cultures; for example, people who are habituated to read from right to left might find it confusing. In that case, the colours of the cup can play an important role as they can be placed from left to right in a sequence from the morning cup colour to the evening cup colour.

The nurses may find it convenient to have their user names and passwords remembered by the application, so that their authentication process is fast and easy. However, the application should not allow or prompt the nurses to save their credentials. It might seem labour-saving from a usability point of view, but from the security perspective it is indefensible.

The feature for retrieving a forgotten user password should not be allowed by the application. The passwords must be generated only by the IT department of the organisation. As time-consuming a process as it may sound, security must be kept as the top priority. The trade-off in this case would be that the user would be unable to provide the service as expected, because of which some patients would be affected.

As discussed in the usability chapter section 2.4, while shifting from the current system of medication administration to a new one, the interface of the application should be designed so that it is not very far from the current system. The familiarity and homogeneity will help in user acceptance and satisfaction. If a user can map the current workflow to the new digitalised Smart Dosing application, the probability of errors will be reduced. The design that I have proposed for the Smart Dosing management system has given great care to replicating the current workflow which can be seen in the Figure B.3

Followed by the physical ordering of the tray with the tablet device and the arrangement of the cup holders, the design should allow a controlled and orderly flow of actions

within one to two clicks. During the main actions, like filling the medicines and dispensing the medicines, the design should allow complete concentration on a particular patient's medication and treatment details. This will help in reducing confusion and errors while operating the application. For example, all the medicines scheduled for one patient should get filled up before going to the next patient. This will ensure a concentration on the medication, the dosage, and the correct timing for each patient, followed by the next patient. The hierarchical organisation of information, as pointed out in the section 2.4, should be taken into consideration while designing and developing the Smart Dosing application. The design put forward by me has one patient's details on a single screen, for example, that patient's name, social security number, and list of medicines on one side, and their pictorial data next to the medicine list. The same screen also helps the nurse to navigate among the treatment details, medical history, and allergies of the patient.

The selection of a list of patients or a list of medicines, which also represent actions to be taken or actions already taken, should not be allowed by a mere touch of the screen. Non-traditional data communication interfaces, such as touch-screen icons, are capable of increasing the integrity vulnerability of systems in the healthcare domain, as pointed out in [98]. An alternative process, checking off check-boxes, will create less confusion and errors, as compared to the touch-screen. Over and above, the patient and medication information should be made clearly visible by giving special attention to the font and font size.

The lists of patients, or the medicines of the selected patient, should not exceed the space allocated for them. If they exceed, a scrolling bar should be made available very clearly, so that a nurse does not miss out on relevant information. The application should be designed in such a way that the nurses cannot proceed to the next process without going through the entire list.

The navigation among the filling option, dispense option, medicine scheduling, pa-

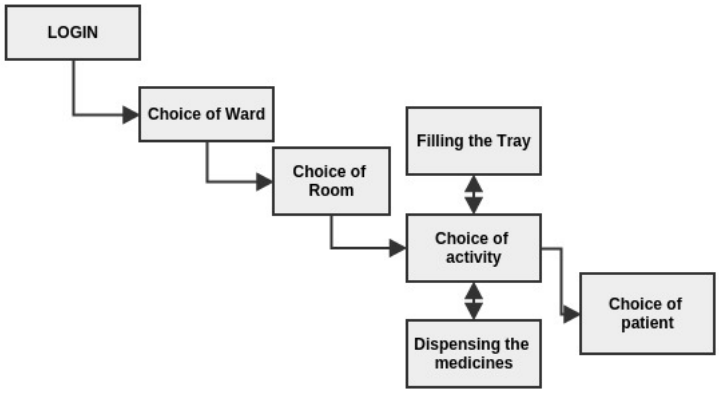

Figure 5.1: Context awareness.

tient treatment details, and information on patient allergies, needs to be fairly clear and easy. The complexity of the application should be kept as low as possible by keeping the navigation hierarchy minimal. Simple and concise navigation will leave less room for cognitive overloading and will ensure less possibility of making human errors. Section 2.4 describes both hierarchical and dynamic organisation of information. While designing the Smart Dosing application, an optimal balance between both seems to bring a better result.

There should be a continuous communication between the tray, the device, and the cup holders, for pointing out and guiding which cup holder belongs to which patient, which medicine-filled cup needs action at what time, and so on. The use of context awareness and contextual information, as described in the section 2.4, would be helpful in enhancing the user experience. In the case of the Smart Dosing application, once a nurse successfully logs in and chooses the assigned ward, the application can be made to automatically retrieve a list of patients in the ward. From the design perspective, I have achieved the context awareness, as every nurse is made to follow a succession of steps to define the working context, as shown in the Figure 5.1.

To be able to cross-verify the medicines whenever required, the medicine, on being

Figure 5.2: Indexicality.

selected, should retrieve its pictorial data. Besides the correctness of the image of the medicines being fetched and displayed, the clarity and placement of those images on the tablet screen should be given attention to. The notification of bulky medicines, injections, refrigerated medicines, or critical drugs which are not at a nurse's easy reach, should be such that they draw the nurse's attention. This requirement agrees with the principle of indexicality that is stated in section 2.4. While designing the Smart Dosing application, I have encouraged the use of semiotics to improve visibility, and to simplify information retrieval, as shown in the Figure 5.2.

A list of equivalent medicines retrieved for the searched medicine should be distinguishable and visible, helping the nurses to make a faster and safer choice. While placing an out-of-stock order, if a nurse is allowed to input text through the application, it may result in erroneous data getting sent. The out-of-stock order for the medicine should automatically choose the details of the medicine selected, that is, the currently selected medicine. If the strength of the medicine needs to be changed, I would recommend that the selection is done from a dial, rather than manually inputting any numbers. This will help prevent input errors or data entry errors.

The security concerns while operating the Smart Dosing application by the nurses, and the likelihood of the threats occuring and their impact, are presented in Table 5.1.

Threat	Likelihood	Impact	Consequences	Recommendations
Breaking of the device screens	1	3	Device partially or completely unusable	Use a device screen guard or replace device/screen
Touchscreen unresponsive	2	3	Navigating inside the application becomes difficult	Update, repair or replace the device
Small fonts	1	3	Unreadable text creating confusion	Use an appropriate font size
Small buttons	1	2	Accidental errors in navigations	Use appropriate sized buttons
Inactive scrolling feature	2	3	An incomplete list of patient or medicine information	Use simple and clear user interfaces
Complex navigation	3	3	Creates confusion, delays activities and increases errors	Use a simple and clear user interface
Misorienta-tion of the device and the tray	2	3	Medicines filled and dispensed incorrectly	Guide the user in fixing the device to the tray
Cup holders and application mapping	4	5	Medicines filled and dispensed incorrectly	Communication between cup holders and application
Battery drainage	3	3	Device not usable	Schedule the charging of the battery periodically

Table 5.1: Usability threats analysis

5.2 Security requirements analysis from a technological perspective

Firesmith [53] and Sindre [99], among many others, assert that there will be several mechanisms to achieve similar objectives, depending on the technology choices. While writing reusable requirements, it is reasonable to concentrate on what kind of security to achieve, and leave out details on how to achieve those. Since there will be several mechanisms to achieve the same or similar objectives, it is impossible to make a general recommendation for what mechanism to choose, as this will depend on the technology choices.

5.2.1 Smart Dosing embracing critical properties of security

The Smart Dosing management system must embrace the following critical properties of security:

Since the Smart Dosing application concerns patient medication data, which are restrictive in nature, the access to them must be controlled. The identification, authentication, and authorisation of the nurses, the head nurses, and the physicians trying to get access into the Smart Dosing application, is an absolute security requirement. It must not be possible for unauthorized persons to access the application. The access control and authorisation levels will be inherited from the application called Miranda medicine system currently in use. The application, therefore, will inherit the same access control flaws that exist, if any. The Smart Dosing application is intended to integrate seven information systems. It is, therefore, required that the application verify the identity of all the services it uses.

The Smart Dosing application must be able to detect any attack or harm being done to it. Additionally, the application should be able to detect suspicious behaviour, and it must have a contingency plan on occurrence of such behaviour. For example, a limited number of attempts should be allowed for the nurses to log into the system, and on failing

to successfully log in within the limited number of attempts, the system should record the username in the system for reference, after temporarily locking down that user. Here exists a critical trade-off between usability and security. If this was a legitimate user, locking down the user will temporarily restrict her or him from providing necessary service. As a result, some patients expecting service from that particular nurse will get affected.

The patients' medical data is the most critical asset in this application. Hence, there is a need for data integrity protection while the data is at rest or in transit. It must not be possible for a data sender to repudiate the transmission of this data. The device being used, the users of the application, and the software application itself are equally important assets, as they are directly concerned with the patient data. The integrity of the tablet devices, the nurses, and the software application must be protected.

The data transmitted externally to or from the application must not be read or modified by any unauthorized persons, and it should be possible to verify whether the sender and receiver of the data are authorised users. It is also required to log security incidents (successful or unsuccessful) and every possible request made to access information from the application. The log of data generated must be securely stored locally or transferred securely to the server. It must be possible to determine at what time data were originated and from what source, and whether they have been tampered with or not.

Patient medical data being very sensitive in nature, patient privacy and safety, which directly affect the organisation's credibility, must be protected. Maintaining patient privacy is a bigger challenge with portable devices. If the device falls in the hands or eyes of an unauthorised user, the privacy of the patients can be breached easily. The application should have an auto-logoff feature or the device should have an auto-blackout feature, preventing unintended peeking at patient information. However, these features result in a trade-off between usability and security. Making a user to log in to the application frequently can be very frustrating. This goes against the very purpose of introducing the application to the hospitals in the first place, as user frustration and stress would amount

to errors in the nurses' judgement.

Regular security auditing must be done to ensure that the security techniques intended and implied for the Smart Dosing system are in active usage. A failure in security auditing may give a false impression of security controls in use. The implementation and proper usage need to be ensured at regular time intervals.

Portable devices can easily fall prey to theft. With the device being stolen, the data contained in it are at risk of getting compromised. It is, therefore, highly recommended that no sensitive data should be stored on the device. The device should have properties supporting anti-theft features. For example, remote wipe software allows a user to remotely erase the data stored on a mobile device in case the device is lost or stolen. Auto-lockout or wipe software allows locking the device or wiping out the data after a certain number of failed logins. The application itself will lock the device or erase the data.

Judging from the nature and purpose of the Smart Dosing application, where patient medication details need to be continuously synced with the main server, it should be ensured that the data is available to the authorised users or concerned service, without interference. If a nurse accessing the Smart Dosing application does not receive the updated medicine information of the patients, it could prove hazardous.

5.2.2 Secure development requirements for the Smart Dosing application

The security requirements from a software point of view for the project concerned are directly dependent on the language being used to write the code, the framework being used for developing the application, the kind of application being developed (such as HTML5, native, or hybrid), default security features available in the chosen operating system, and secure coding practices being followed. To achieve robustness against malformed content, careful coding practices should be used, along with mature and security tested interfaces.

The primary obstacles in the development of secure mobile healthcare applications are speculated to be device limitations, wireless networking problems, infrastructure constraints, security concerns, and user distrust, according to [100].

A strategic approach is needed for designing or developing any life-critical application, such that the code produced has very few defects from the start. Errors made during the design process, poor coding practices during the development of the application, and following improper implementation methods, can expose device data to interception or tampering. Hence, it is advisable to minimise the likelihood of exploitation by fixing the vulnerabilities from an early stage. If security is considered as an afterthought, issues that come to light in the future may not necessarily be resolved by security measures. Therefore, a security viewpoint must be included from the very beginning in planning, building, and maintaining all operations. The secure-at-the-source approach, as discussed in section 3.2, will decrease the friction between usability and security. This approach will also improve the reliability of the operations and the ability to quickly react to disruptions.

When a developer inadvertently stores sensitive information or data in a location on a mobile device that is accessible by another application running on the same device, it may result in unintended data leakage. This could happen as a result of programmatic flaws or enabling insecure OS features in the application. Unintended data leakage has the capability of making sensitive data end up at places like web caches, global OS logs, screen shots (iOS backgrounding issue), or temp directories etc. In the case of the Smart Dosing application, such a leakage may lead to frauds or privacy violations.

If device identifiers such as IMEI (International Mobile Equipment Identity), IMSI (International Mobile Subscriber Identity), or UUID (Universally Unique Identifier) values are used for implementing authentication and authorization, they make up for the perfect recipe for a security failure. Such actions can lead to broken authentication and privilege access issues. Since all client-side authorization and authentication controls can be bypassed, the authorization and authentication controls must be re-enforced on the

server-side whenever possible.

Insecure development practices while implementing the encryption process, such as use of custom cryptographic algorithms instead of standard cryptographic algorithms, confusing encryption methods with encoding or code obfuscation methods, and hard-coding cryptographic keys into the application code itself, can lead to broken cryptography. Besides a failure in the cryptographic implementation, such developmental practices can result in a loss of confidentiality of the data, privilege escalations, privacy violations, and so on. This addresses the risk of cryptographic flaws, as stated in section 3.3.2. The associated issues with the cryptographic flaws are improper validation of certificates, clear text storage of information, and the use of broken or weak cryptographic algorithms.

Buffer overflow is a language-inherent risk, as listed in section 3.3.2. It occurs when more than intended data gets stored in a buffer, a temporary area for data storage, causing some data loss as a result of data overwriting or data corruption. In a buffer-overflow attack, the extra data at times hold specific instructions that could respond in the form of damaging files, changing data, or disclosing sensitive information. A malicious user could use a buffer-overflow vulnerability to exploit an application that is waiting on a user's input. I would recommend that the Smart Dosing application limits or filters user input, and prevents data entries, typos, abbreviations, and jargon. There should be data validation on entry and retrieval from an authenticated source, in order to prevent deliberate or inadvertent input errors by the users. The application also should avoid using insecure third-party library files.

The credential management risk, as pointed out in section 3.3.2, encourages the use of alphanumeric passwords of a minimum length while operating the Smart Dosing application. Also, the passwords must not be saved on the device but should be encrypted and saved in a database. Under no condition should the mandatory fields, such as username and password, accept empty parameters. The error messages should not give away more information than what is required for fixing the error. For example, while logging in, the

error message should not state which part of the given information was correct. Here, it is a trade-off between usability and security. It is strongly recommended to disable the "Make passwords visible" option or not have that feature at all. Doing so will prevent shoulder surfing attacks, as the characters will not be visible when the user is typing a password or a PIN. In the current prototype, both the username and the password fields have been made mandatory, sufficient error messages have been provided, and password visibility option has been disabled.

The electronic health records accessible via the application need secure guarding. The creation of electronic health records by the application and their maintenance should preserve content authenticity and data integrity. The electronic health records have brought up privacy issues in healthcare [101]. However, in the case of the Smart Dosing application, the risks carried by paper-based storage and server storage are equivalent, in my opinion. While paper-based medical charts are physically limited to the people concerned, connecting the medical data to the Internet and making the data accessible via a portable device expose the data to more hostile attacks. It is recommended by the author in [101] that the access and sharing of electronic health data should provide end-to-end source verification through signatures and certification process against authorised modification in the critical data content and user agreements.

Cross site scripting (XSS), as referred in section 3.3.2, is an attack usually against a server. As long as the Smart Dosing application is not serving Web pages to external connections, this vulnerability has no relevance.

As opposed to the attack strategy against web-based applications where attackers mostly use pure blackbox methodology, implying that the attackers need not peek into the internal structures, the code is already in the hands of the attackers in the case of the mobile applications [102]. In the same article, the author explains that both Android and iOS applications can be extracted from the devices in their binary format, and a mixture of techniques like reverse engineering or dynamic analysis can be used to decompose the

application. Therefore, the client side logic code should be obfuscated. Code obfuscation is the deliberate act of making source code difficult to read and understand by humans, making the code a bit more difficult to debug and reverse engineer only from executable files. However, a complete reliance on the code obfuscation method should not be encouraged, as reverse engineering obfuscated code may be difficult, but not impossible. My recommendation would be that once the Smart Dosing application successfully authenticates and verifies the user, the application logic should be performed on the server side, making it out of reach for reverse engineering.

In order to minimise information leakage due to reverse engineering, I would recommend that the Smart Dosing application perform code obfuscation, execute server side processing wherever possible, carry out iterative hashing, and use salt. It is also very important to choose the right location for saving sensitive information.

Irrespective of what kind of application the Smart Dosing would be, a Web-based or a native one, if the code is poorly written, the application will be susceptible to attacks. For ensuring better security, I strongly recommended that the developers follow secure coding guidelines during the development, conduct a penetration test after completing the development, and continue to do so at regular intervals even afterwards.

5.2.3 Analysis of vulnerabilities in the Smart Dosing system

A weakness can be present in the software, environment, system, network, etc. In order to build a secure and robust Smart Dosing system, it is important that every component involved in the system is secure. It is, therefore, necessary to identify the components and evaluate the currently existing vulnerabilities in them. Since a tablet device is being targeted to run the application on, any security flaws of the device need to be examined. Depending on the kind of application to be developed, HTML5, native, or hybrid, vulnerabilities in the same should be identified.

Vulnerabilities in the Smart Dosing system can be broken down to vulnerabilities in

the device, vulnerabilities in the software, and vulnerabilities in the tray design. Vulnerabilities in the tablet device can be analysed after making the choice of what hardware and operating system are to be used for running the software application.

Vulnerabilities in the device running the software application

As discussed in section 3.3.1, the operating system of the device is the common location of security flaws. Flawed applications, coupled with platform vulnerabilities, are the perfect recipe for an attack. Some security features that have been built into the platforms, if used properly, could mitigate the application vulnerabilities [73]. Those features include code signing, sandboxing, and permission notifications. The code signing process will allow the users to verify an application's source, the sandboxing process is known to separate an application from other processes, and the permission notifications will assist the users by warning them when an application is attempting to access their data.

To meet security requirements such as authentication, integrity, confidentiality, message authentication, and non-repudiation in a mobile environment, additional security software and features (e.g., certificates, private and public keys) should be installed on the tablet devices.

Vulnerabilities in the software application

The first prototype of the Smart Dosing application was developed using the Vaadin framework. The Vaadin framework features a robust server-side architecture, that is, the largest part of the application logic runs securely on the server. Vaadin has a large collection of UI components from which application user interfaces can be composed. These components use events, listeners, and data binding to communicate with each other and with the business logic. The component-based architecture, along with the Java language and data binding features, help in building applications that can be easily modularized and re-factored if needed. It also supports all common browsers for traditional PC's, as

well as mobile devices and tablets. However, loading the Vaadin client side takes rather a long time, affecting bandwidth and performance of the developed application. This raises the need for building an optimised widget set. The Vaadin client side also limits the use of immediate mode, text change events, and polling. The browser support in the touch-kit is concentrated on web-kit, and there is no dedicated back button support.

There is no definitive recommendation on whether to develop a native application, a Web application, a hybrid application, or whether to use a multi-platform framework. It depends on the requirements of the application, the application's intended features, the resources available, the target users of the application, and the skills of the development team. Limiting to a single platform potentially diminishes the market share of the application or demands a re-development, if a device or platform becomes obsolete. The inception of cross-platform development, in many respects, eases this, as one can develop software that runs on any device where the run-time environment is available [103]. In my opinion, for rapid prototyping, multi-platform frameworks could be used, but in case security is of paramount importance, native applications should be considered. My recommendation is that available platform features should be inspected and then a careful choice of the framework should be made.

HTML5 promises easy multi-platform development, where only a single application needs to be created and executed on multiple platforms. As easy and straightforward as it sounds, it comes with disadvantages, such as limited access to the device hardware or to platform features, or look and feel that users are not acquainted with from other applications for the same platform. An HTML5-based application is also associated with the Web-based application security vulnerabilities (SQL Injection, XSS, HTTP redirect, session fixation) that occur generally due to a lack of input validation.

The vulnerabilities present in mobile software applications, as enumerated in section 3.3.2, are analysed from the Smart Dosing application perspective and presented below:

The Smart Dosing application will be connected to an interface of the hospital infor-

mation system. If the application possess SQL injection vulnerability, listed in 3.3.2, it can alter the patient medication data. To prevent SQL injection vulnerability and its after-effects, my recommendation would be not to allow untrusted and unvalidated user inputs from the Smart Dosing application.

If the device running the Smart Dosing application is connected to the Web, and the users are allowed to download or install applications freely, there is a possibility that malicious applications or services can hijack the services provided by the Smart Dosing application. Once successfully hijacked, there can be a denial-of-service attack, as discussed in section 3.3.2. There is a need for preventing data sharing with other applications, as explained in section 3.4.2. The application must be made to avoid interaction with other unsafe applications and retain its functionality to itself. Information sharing between the Smart Dosing application and untrusted applications on the same mobile device is capable of disclosing sensitive information. This raises a need for monitoring inter-process channels and applying safeguards to prevent such disclosures. It is highly advisable that the device be not connected to the Web.

The Smart Dosing application would need to implement SSL/TLS for authenticating and verifying the users, and for secure transmission of patient data between the application and the hospital server. Freak SSL-TLS vulnerability, as explained in section 3.3.2, if exploited, would result in leaking sensitive data by means of man-in-the-middle attacks. I would recommend that the Smart Dosing application implement SSL validation correctly, and verify the authenticity of the systems it is connected to. As explained in section 3.3.2, SSL-stripping vulnerability is not relevant for the Smart Dosing application, as it is not a Web-based application.

For dealing with effects of insecure data storage, one of the mobile vulnerabilities discussed in section 3.3.2, I would strongly recommend to avoid hard-coded encryption or decryption keys when storing sensitive information. Although mobile operating systems provide data protection and separation between applications, there should be additional

layers of encryption on top of the default encryption mechanisms.

5.2.4 Threats to the Smart Dosing system

A commonly held misconception is that the primary risk to privacy and confidentiality of patients or physicians comes from intruders. On the contrary, both Simpson [104] and Rindfleisch [105] emphasize internal vulnerabilities, more specifically, accidental or malicious disclosure, inside curiosity, or uncontrolled secondary usage like forwarding information to unauthorised third parties. Security and confidentiality can be breached either accidentally or deliberately, as discussed in section 3.3.3. However, an effective administration of the medical database, while balancing both technical and non-technical challenges, and incorporating security into health information systems from the initial design stage can result in a better safeguarding of confidentiality.

Threats to the Smart Dosing system can be an insider like a nurse or an outsider like hackers. Malicious insider action could include theft of a tablet device, theft of user credentials, theft of patient data, using the application on another user's name, and similar. Other threats to the Smart Dosing application would be using cache data to get passwords of logged-in or authorised users, modifying a patient record for personal gain, and the like.

Application-based threats

The application should be allowed to communicate only with trusted external entities, and monitoring the connections made by the application, as discussed in section 3.4.2, should be possible. If the application gets afflicted by application-based threats like malwares, spywares, and vulnerable applications, the patient privacy can be compromised and the application can be made to operate undesirably. These threats can be controlled and prevented by limiting the access of the device to the hospital network only, and not allowing any download of applications to the device. An additional piece of advice would be cre-

ating a whitelist of applications, and educating the users about not installing applications outside of the approved list.

For improving the security of the application, the data among different applications that are running in the same network should be insulated, as listed in section 3.4.2. I would recommend installing a virtual private network (VPN) in the software, in order to provide a direct and encrypted tunnelling of data between the Smart Dosing application and the server. It will also minimise the traffic at the same time. There are several VPN's available for a mobile environment [106].

Another recommendation would be to keep the USB debugging feature turned off on the devices. USB debugging allows a user to connect the device to a computer which can, lead to the display of information on the device.

Web-based threats

Web-based threats like phishing scams, drive by downloads, a party posing as a legitimate user, and browser exploits, come into the picture when the devices are connected to the Internet for accessing Web-based services. If the Smart Dosing application is deployed on the Web and is accessed through a browser, the exploitation of the listed threats can steal or modify data by allowing illegal access to the application. Hence, the browser security and privacy settings should be fine-tuned (e.g., disable location access).

Network-based threats

The Smart Dosing application will be using the existing network available in the hospitals. Network-based threats, such as Wi-Fi sniffing and network exploit, can give the attackers access to sensitive data during their transit. Denial-of-service attacks can render the application useless. Threats such as Wi-Fi sniffing can be prevented by limiting the device to connect to other service set identifiers (SSIDs). The SSIDs are the primary names associated with the wireless local area network. The threat of network exploit and

Threat	**Likelihood**	**Impact**	**Consequences**	**Countermeasures**
Loss	2	3	Compromising patient medication information	Sensitive data should not be saved on the device
Theft	1	3	Loss of a hardware device and the data on it	Sensitive data should not be saved on the device
Illegal access	2	3	Unauthorised usage of application	Authentication and authorisation

Table 5.2: Physical threats analysis

denial-of-service can be prevented by using a VLAN. Mobile network service threats can be prevented by using a Tablet PC without the provision of SIM card slots. Security in wireless network environments cannot be solved statically, but must be engaged in on an ongoing basis. There should be active security administration, and the best network security practices must be observed.

I suggest that the services like Bluetooth, NFC, and location features are kept turned off unless they are being actively used. The network notification feature should also be disabled, so that a user is forced to choose a connection rather than connecting to any available network. The tablet device should be dedicated to the hospital related applications alone, and the required applications should be downloaded and installed by the hospital system administration or who so ever is in charge.

Physical threats

Physical threats like theft, loss, or an improper disposal of devices are risks the Smart Dosing system carries with itself. The application will be running on a portable device which will be carried around by the nurses for patient care.

A solution for preventing stealing sensitive information from the device can include password protection, followed by locking or logging out of the application on the occurrence of prolonged inactivity. The automatic locking of the devices requires frequent authentication. Therefore, the chosen authentication method must be free from cumbersome manipulations. Due to the limitation of the application being accessible only in the hospital network, theft of the device would not expose sensitive data stored on the device, if any. However, data should not be saved on the device but be transmitted continuously and saved securely on a server. On a side note, a data security plan for portable devices, such as remote wipe of data, should be present.

Additionally, my recommendation for dealing with the physical threats would be using an NFC tag on the tray and a personal NFC tag for the nurse. This will add another layer of authorisation for using the medicine tray with the device.

The operating system Android 3.0 and later have the capability to perform full-disk encryption, excluding the SD card. Enabling this feature will encrypt all the data on the device, saving the data from being recovered when the device is lost or stolen. My advice, however, would be to use both software encryption and hardware encryption for an extra layer of security.

Physical security concerns related to the assets of Smart Dosing system have been tabulated in Table 5.2.

The threats and challenges to the hospital information system, as discussed in section 3.4, have been analysed with respect to the Smart Dosing application as follows:

Accidental user errors while operating the Smart Dosing application can cost lives of patients. Firstly, the design of the application should be such that there is no or little room

for user errors. At the same time, a user should be able to undo the errors on realising them. The application should be developed so that it is resilient to the accidental errors made by users during their usage or during software malfunctions. Hardware failure in case of the Smart Dosing will render the software application unusable. Saltzer and Schroeder's [107] principle of fail-safe defaults suggests that access to the application or systems should be based on permission rather than exclusion i.e., the default situation should be lack of access. It also recommends adhering to the principle of least privilege, as explained in section 3.3.2. This principle limits the damage that would occur from an accident or an error, by allowing access to only the information or resources that are necessary for its legitimate purpose.

Threats of environmental origin like a fire or a power failure have no influence on the Smart Dosing application, as the application runs on a portable device which can be easily carried away from the fire. Since the tablet runs on a battery, a power failure will have no effect on the operation of the device.

5.3 Security requirements from the medication administration workflow perspective

Unauthorised individuals should not be allowed to create user accounts. New users should be added to the system only by the head nurse, after verifying themselves by showing their work contract and ID. Therefore, the Smart Dosing application should be designed to accept only exclusively verifiable personal information of the users. There should not be a sign-up functionality in the application.

During the process of filling the tray, data such as who has logged in, what medicines have been filled and for which patients, along with a date and time stamp, must be saved continuously. Doing so would keep an electronic log of all the activities for monitoring purpose and improve operational efficiency.

It is required that a nurse does not jump to the next patient without having filled all the medicines prescribed for the patient selected. In case there appears to be an emergency break from the activity, the action should get saved, and the nurse should be able to continue from where the activity was left off.

Once the filling of the tray is over for the selected patients in the ward, and the medicines are ready to be dispensed, the nurse should log out from the application voluntarily. The nurse should otherwise get automatically logged out after a certain duration of inactivity.

The list of patients for whom the medicines need to be dispensed should be exactly the same as the list of patients for filling the tray was, unless there has been an emergency admittance of a patient after the filling process for the day is done.

The application is meant to be run on Tablet PC's or iPad, which makes it available to nurses at any time and place. As the nurses would be going from one patient's bedside to another carrying the device, a patient can cross-check the medicine he or she is being handed over, if the patient wants to. However, care must be taken that there is just enough information displayed at that time on the screen, for privacy reasons.

It should be made sure that the nurse checks off the medicines after they have been dispensed to the patients. If the check-boxes beside the medicine dispensed are not marked, the application will save wrong logs, saving inaccurate data in the system. I suggest that the nurse is not allowed to move to the next patient until all the necessary medicines listed for a patient are checked off.

If a medicine cup is filled with a non-prescribed medicine by mistake, there should be a provision in the application to detect this, for example, a comparison of the actual pill and the pill filled into the medicine cup. The verification of the medicines should be possible while filling the tray and before dispensing them to the patients. That is why we, i.e the contributors of the Smart Dosing project, recommend that the application is connected to the Pharmaca Fennica system [93].

In case there is a need to skip the medicines, or to schedule a change in the prescription, an option should be provided for documenting the reasons instantly via the application, without having to return to the medication room. The details then need to get integrated with the main server. For input validation purposes, I suggest that there is a list of frequent or possible pre-defined reasons to choose from for skipping medicines.

A large number of medicines scheduled at peak times causes frequent medication errors, as pointed out in [10]. For that, I suggest a sorted list of patients due for their morning medications, afternoon medications, and evening or night medications. This feature would save the nurses the effort of going through the entire list of patients' medications, and timing or memorizing their administering times. It could show a list of patients with the medications due next. This would make the workflow smooth and hassle-free.

On several occasions there are same medicines prescribed for different patients. In order to save time, I suggest that there is a provision for filling the tray according to medicine, that is, on selecting a medicine, the application will give the list of patients for whom the medicine is prescribed.

6 Conclusions and future work

In the world of security, it is believed that there exist two types of organisations, secured and unsecured. Whether security features are implemented or not, critical infrastructures are continuously under attack.

The objective of this Master's thesis was to analyse the security requirements for the Smart Dosing application, by understanding the attacks relevant for the organisation concerned. The seriousness of the threats depends on the value of the assets, and the ramifications of loosing or compromising those assets. Although building a secure application is an iterative approach, this thesis aims to assist the developers of the Smart Dosing application in building a compelling user interface and a secure application from the very beginning. The thesis brings to the reader's attention the importance of thinking about security from the developmental stage, and following secure developmental practices. There will be bugs and vulnerabilities at least as long as the programs are written by the humans. The thesis has identified the vulnerabilities, evaluated the basic threats, and determined the risks of introducing the Smart Dosing application in a hospital environment. The most important results obtained in this thesis are error identification in the medication dispensation workflow, security analysis of previous version of the prototype and security requirement analysis from both usability and technological perspective for the upcoming prototype.

This thesis also guides the readers towards the secure technology choices that should be made, depending on the environment where the application will be used. The applica-

tion's security depends on the technical specifications, interoperability with other health IT systems, data integration with back-end systems, and compelling user interfaces. Human factor being the most common factor responsible for errors, both usability and security have been weighed in. In conclusion, since both usability and security are paramount in a successful running of a medical application, a right balance must be maintained between the two.

The Smart Dosing management system can protect patient data by implementing a mix of technology and best practices. The organisation using Smart Dosing application should manage the devices by providing tools for grouping devices, forcing device passwords, forcing storage encryption, and wiping the data on the device when they are lost or stolen. Also, the organisation should implement bring your own device (BYOD) policy for devices that are not affiliated by that healthcare organisation. A combination of technology and user training would make Smart Dosing operate securely.

Any future work would include developing the Smart Dosing application, keeping the security requirements in mind, incorporating and refining the security features as time passes, reassessing the health information security practices, and developing appropriate security policies to be used on the adoption and implementation of the application in the hospital workflow. The threats and risks need to be identified and prioritised, depending on the technological choices made for the application.

During the development of the application, there has to be iterative usability evaluation, by making the nurses use and share their practical knowledge and suggestions for improvement.

Communication must be established between the medicine tray, the cup holders, and the tablet device.

The application needs to be integrated with the Pharmaca Fennica [93] system and the interfaces of the hospital information system. There have been discussions with CGI, the IT company that currently manages Uranus software, the medication system which is

being used by the hospital. In the discussions, the CGI has agreed to create the interfaces for the application, and provide a test environment.

References

[1] N. Bevan. Usability is quality of use. Advances in human factors ergonomics, 20:349–349, 1995.

[2] D. G. Firesmith. Security use cases. Journal of object technology, 2(3):53–64, 2003.

[3] Infosec institute. Introduction to secure software development life cycle, 2013. http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle[Online; accessed 23-Sept-2015].

[4] Veracode. Identifying the mobile security stack. 2011. https://www.veracode.co.uk/blog/2011/03/identifying-the-mobile-security-stack[Online; accessed 15-July-2015].

[5] Y. Cifuentes, L. Beltrán, and L. Ramírez. Analysis of security vulnerabilities for mobile health applications. In 2015 Seventh International Conference on Mobile Computing and Networking (ICMCN 2015), 2015.

[6] K. Mario and O. Eugene. Native, html5, or hybrid: Understanding your mobile application development options, 2012. https://developer.salesforce.com/page/Native,_HTML5,

_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options[Online; accessed 23-Sept-2015].

[7] G. N. Samy, R. Ahmad, and Z. Ismail. Threats to health information security. In Information Assurance and Security, 2009. IAS'09. Fifth International Conference on, volume 2, pages 540–543. IEEE, 2009.

[8] G. Eysenbach. What is e-health? Journal of medical Internet research, 3(2):e19, 2001.

[9] L.H. Iwaya, M.A. Gomes, et al. Mobile health in emerging countries: A survey of research initiatives in brazil. International journal of medical informatics, 82(5):283–298, 2013.

[10] N. Rodriguez, J. Lilius, S. Bjorklund, J. Majors, K. Rautanen, R. Danielsson-Ojala, H. Pirinen, L. Kauhanen, S. Salantera, and T. nd others Salakoski. Can it health-care applications improve the medication tray-filling process at hospital wards? an exploratory study using eye-tracking and stress response. In e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on, pages 423–428. IEEE, 2014.

[11] N.A. Khan, N. Díaz Rodríguez, et al. Smart dosing: A mobile application for tracking the medication tray-filling and dispensation processes in hospital wards. Recent Advances in Ambient Assisted Living-Bridging Assistive Technologies, E-Health and Personalized Health Care, 20:134, 2015.

[12] R. Pradhan, N. Díaz Rodríguez, R. Danielsson-Ojala, H. Pirinen, L. Hamari, I. Tuominen, S. Salanterä, A. Soini, and J. Lilius. Medicine tray design and security for medication administration processes at hospital wards. In Audrey Girouard, David McGookin, Katie Siek, Orit Shaer, Marilyn Lennon, and Peter

Bennett, editors, CHI 2016 Tangibles for Health Workshop, page 1–5. ACM, 2016.

[13] A. Abran, A. Khelifi, W. Suryn, and A. Seffah. Usability meanings and interpretations in iso standards. Software Quality Journal, 11(4):325–338, 2003.

[14] J. Rubin and D. Chisnell. Handbook of usability testing: how to plan, design and conduct effective tests. John Wiley —& Sons, 2008.

[15] United States Census Bureau. Nearly 1 in 5 People Have a Disability in the U.S, 2013. https://www.census.gov/newsroom/releases/archives/miscellaneous/cb12-134.html[Online; accessed 12-Dec-2014].

[16] Foraker Labs. Usability methods. http://www.usabilityfirst.com/usability-methods [Online; accessed 1-Jan-2015].

[17] N. Jakob and T. Marie. Homepage usability: 50 websites deconstructed, 2001. http://www.nngroup.com/books/homepage-usability/[Online; accessed 2-Jan-2015].

[18] M. Matera, F. Rizzo, and G. T. Carughi. Web usability: Principles and evaluation methods. In Web engineering, pages 143–180. Springer, 2006.

[19] J. Thyfault. Why You Should Care About Website Usability, 2013. http://www.onlinemarketinginstitute.org/blog/2013/05/importance-website-usability/[Online; accessed 2-Jan-2015].

[20] W. Nicole. 25-point website usability checklist, 2015. https://web.wsu.edu/2015/07/15/website-usability-checklist/[Online; accessed 2-Jan-2016].

[21] B. Stefania, B. Jonathan and G. Giuliano. Usability, design and content issues of mobile apps for cultural heritage promotion:the malta culture guide experience, 2012. http://arxiv.org/pdf/1207.3422.pdf[Online; accessed 2-Jan-2015].

[22] T. Basanta. Mobile Sites vs. Apps Comparison and Best Practices, 2010. http://www.usabilitysciences.com/assets/pdf/Usability_Sciences_Mobile_Sites_vs_Apps_Sept2010.pdf[Online; accessed 3-Jan-2015].

[23] venturebeat. Study: Mobile app usage grows 35 %, TV & web not so much, 2012. http://venturebeat.com/2012/12/05/mobile-app-usage-tv-web-2012/[Online; accessed 20-Jan-2015].

[24] CDIXON. The decline of the mobile web, 2014. http://cdixon.org/2014/04/07/the-decline-of-the-mobile-web/[Online; accessed 20-Jan-2015].

[25] L. Gorlenko and R. Merrick. No wires attached: Usability challenges in the connected mobile world. IBM Systems Journal, 42(4):639–651, 2003.

[26] L. L. James. User interface design for the mobile web, 2011. http://www.ibm.com/developerworks/library/wa-interface/[Online; accessed 20-Jan-2015].

[27] Apple. Apple iOS Human Interface Guidelines. http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html/[Online; accessed 4-Jan-2015].

[28] Google Android. Google User Interface Guidelines. http://developer.android.com/guide/practices/ui_guidelines/index.html[Online; accessed 4-Jan-2015].

[29] F. Nayebi, J. Desharnais, and A. Abran. The state of the art of mobile application usability evaluation. In CCECE, pages 1–4, 2012.

[30] R. Harrison, D. Flood, and D. Duce. Usability of mobile applications: literature review and rationale for a new usability model. Journal of Interaction Science, 1(1):1–16, 2013.

[31] S. Brewster. Overcoming the lack of screen space on mobile computers. Personal and Ubiquitous Computing, 6(3):188–205, 2002.

[32] United States Census Bureau. Colour blind considerations for ui design, 2013. http://davemeeker.com/color-blind-considerations-for-ui-design/[Online; accessed 20-Jan-2015].

[33] S. Agrawal and A.I. Wasserman. Mobile application development: A developer survey. submitted for publication, 2010.

[34] T. Flew. What will the apple ipad deliver for newspapers? 2010.

[35] F. Brad. Mobile-first responsive web design, 2011. http://bradfrost.com/blog/web/mobile-first-responsive-web-design/[Online; accessed 20-Jan-2015].

[36] J. Nielsen. Tablet Usability, 2013. http://www.nngroup.com/articles/tablet-usability/[Online; accessed 12-Dec-2014].

[37] B. Smith. 5 Useful iPad Apps for Doctors, Patients and Med Students, 2012. http://mashable.com/2012/01/16/ipad-medical-apps/[Online; accessed 5-Jan-2014].

[38] MBBS. Dwayne and K. Yao. Tablet computers in surgery. Journal of Mobile Technology in Medicine, 2(2):15–19, 2013.

[39] I. Robertson, E. Miles, and J. Bloor. The ipad and medicine. BMJ Careers web site, 2010.

[40] C. L. Ventola. Mobile devices and apps for health care professionals: uses and benefits. Pharmacy and Therapeutics, 39(5):356, 2014.

[41] R. Wu, P. Rossos, S. Quan, S. Reeves, V. Lo, B. Wong, M Cheung, and D. Morra. An evaluation of the use of smartphones to communicate between clinicians: a mixed-methods study. Journal of medical Internet research, 13(3):e59, 2011.

[42] M. Prgomet, A. Georgiou, and J. I. Westbrook. The impact of mobile handheld technology on hospital physicians' work practices and patient care: a systematic review. Journal of the American Medical Informatics Association, 16(6):792–801, 2009.

[43] S. Mickan, J. K. Tilson, H. Atherton, N. W. Roberts, and C. Heneghan. Evidence of effectiveness of health care professionals using handheld computers: a scoping review of systematic reviews. Journal of medical Internet research, 15(10):e212, 2013.

[44] P. A. Abbott. The effectiveness and clinical usability of a handheld information appliance. Nursing research and practice, 2012, 2012.

[45] R. G. Saadé and C. A. Otrakji. First impressions last a lifetime: effect of interface type on disorientation and cognitive load. Computers in human behavior, 23(1):525–535, 2007.

[46] M.S. Kim, J.S. Shapiro, N. Genes, M.V. Aguilar, D. Mohrer, K. Baumlin, J.L. Belden, et al. A pilot study on usability analysis of emergency department information system by nurses. Appl Clin Inform, 3(1):135–153, 2012.

[47] M-C. Beuscart-Zéphir, P. Elkin, S. Pelayo, R. Beuscart, et al. The human factors engineering approach to biomedical informatics projects: state of the art, results, benefits and challenges. IMIA Yearbook, 2:109–127, 2007.

[48] F. Ehrler, R. Wipfli, D. Teodoro, E. Sarrey, M. Walesa, and C. Lovis. Challenges in the implementation of a mobile application in clinical practice: case study in the context of an application that manages the daily interventions of nurses. JMIR mHealth and uHealth, 1(1):e7–e7, 2013.

[49] D. Zhang and J. Lai. Can convenience and effectiveness converge in mobile web? a critique of the state-of-the-art adaptation techniques for web navigation on mobile handheld devices. International Journal of Human-Computer Interaction, 27(12):1133–1160, 2011.

[50] C. Biancalana, F. Gasparetti, A. Micarelli, and G. Sansonetti. An approach to social recommendation for context-aware mobile services. ACM Transactions on Intelligent Systems and Technology (TIST), 4(1):10, 2013.

[51] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt. A comparison of security requirements engineering methods. Requirements engineering, 15(1):7–40, 2010.

[52] D. He, C. Chen, S. Chan, J. Bu, and A. V. Vasilakos. A distributed trust evaluation model and its application scenarios for medical sensor networks. Information Technology in Biomedicine, IEEE Transactions on, 16(6):1164–1175, 2012.

[53] D.G Firesmith. Specifying reusable security requirements. 2004. Journal of Object Technology, pages 61–75.

[54] M. Zulkernine and S. I. Ahamed. Software security engineering: toward unifying software engineering and security engineering. Enterprise Information Systems Assurance and Systems Security: Managerial and Technical Issues, M. Warkentin & R. Vaughn, pages 215–232, 2006.

[55] R.G. Choo, K.K.R. Smith and R. McCusker. Future directions in technologyenabled crime: 2007e09. research and public policy series no 78. canberra: Australian institute of criminology, 2007.

[56] J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual, pages 55–64. IEEE, 1999.

[57] M. Whitman and H. Mattord. Principles of information security. Cengage Learning, 2011.

[58] Osa 1 Tietoturvakäytännöt ja periaatteet. Varsinais-suomen sairaanhoitopiirin tietoturvasuunnitelma, 2009.

[59] M. Bishop. What is computer security? Security & Privacy, IEEE, 1(1):67–69, 2003.

[60] C. P. Pfleeger. The fundamentals of information security. Software, IEEE, 14(1):15–16, 1997.

[61] XebiaLabs Coert Baart, CEO. Roi study on automated application deployments for the enterprise, 2011. `http://go.xebialabs.com/rs/xebialabs/images/WP_2012-08_ROI.pdf`[Online; accessed 23-Oct-2015].

[62] Aberdeen Group. Securing your applications: Three ways to play. 2010. `http://www.aberdeen.com/research/6580/ra-web-application-security/content.aspx` [Online; accessed 30-July-2015].

[63] G. Kenneth. Addressing challenges in application security, 2005. ftp://ftp.software.ibm.com/software/fr/tendances_logicielles/addressing_challenges_in_application_security.pdf[Online; accessed 23-Oct-2015].

[64] M. Howard and S. Lipner. The security development lifecycle. O'Reilly Media, Incorporated, 2009.

[65] M. Goodman. Future crimes: A journey to the dark side of technology–and how to survive it. 2015.

[66] L. Ingrid. 6.1B Smartphone Users Globally By 2020, Overtaking Basic Fixed Phone Subscriptions, 2015. http://techcrunch.com/2015/06/02/6-1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-#.jqlaka:86lq/[Online; accessed 9-July-2015].

[67] Statista. Number of apps available in leading app stores as of july 2015. 2015. http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/[Online; accessed 27-Aug-2015].

[68] L. T. Kohn, J. M. Corrigan, M. S. Donaldson, et al. To err is human:: building a Safer Health System, volume 6. National Academies Press, 2000.

[69] Cisco. Cisco 2014 Annual Security Report, 2014. http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf[Online; accessed 19-Jan-2015].

[70] V.A. McLean. Cyren publishes 2013 internet security yearbook. 2014. http://ir.cyren.com/releasedetail.cfm?releaseid=833013/[Online; accessed 12-July-2015].

[71] F-secure. Threat report h1. 2014. https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2014.pdf[Online; accessed 12-July-2015].

[72] Symantec Corporation. INTERNET SECURITY THREAT REPORT 2014, 2014. http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf[Online; accessed 19-Jan-2016].

[73] W. Robert. Android app security: Study finds mobile developers creating flawed android apps, 2011. http://searchsecurity.techtarget.com/news/2240112235/Android-app-security-Study-finds-mobile-developers-creating-fl accessed 19-Jan-2016].

[74] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In Proceedings of the 9th international conference on Mobile systems, applications, and services, pages 239–252. ACM, 2011.

[75] A. Juntunen, E. Jalonen, and S. Luukkainen. Html 5 in mobile devices–drivers and restraints. In System Sciences (HICSS), 2013 46th Hawaii International Conference on, pages 1053–1062. IEEE, 2013.

[76] Gartner. Gartner Recommends a Hybrid Approach for Business-to-Employee Mobile Apps, 2013. http://www.gartner.com/newsroom/id/2429815/[Online; accessed 23-July-2015].

[77] H. Brinio. The other, darker side of hybrid apps, 2014. http://hackingthroughcomplexity.nl/tag/hybrid-apps/[Online; accessed 24-July-2015].

[78] M. Sujithra and G. Padmavathi. Mobile device security: A survey on mobile device threats, vulnerabilities and their defensive mechanism. International Journal of Computer Applications, 56(14), 2012.

[79] D. Li, J. Gu and Y. Luo. Android malware forensics: Reconstruction of malicious events. In Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on, pages 552–558. IEEE, 2012.

[80] J. C. Goodman. Health information technology: Benefits and problems.

[81] Ponemon Institute. Fifth annual benchmark study on privacy —& security of healthcare data. 2015. http://media.scmagazine.com/documents/121/healthcare_privacy_security_be_30019.pdf[Online; accessed 02-Nov-2015].

[82] Dutch National High Tech Crime Unit Irish Reporting Verizon RISK Team, Australian Federal Police, Police Central e-Crime Unit Information Security Service, and United States Secret Service. 2012 data breach investigations report. 2012. http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigations-report-2012-press_en_xg.pdf[Online; accessed 12-July-2015].

[83] R. Martí, J. Delgado, and X. Perramon. Security specification and implementation for mobile e-health services. In e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004 IEEE International Conference on, pages 241–248. IEEE, 2004.

[84] M. Meingast, T. Roosta, and S. Sastry. Security and privacy issues with health care information technology. In Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE, pages 5453–5458. IEEE, 2006.

[85] M. Ahmed and M. Ahamad. Protecting health information on mobile devices. In Proceedings of the second ACM conference on Data and Application Security and Privacy, pages 229–240. ACM, 2012.

[86] J. M. Dickerson, B. L. Koch, J. M. Adams, M. A. Goodfriend, and L. F. Donnelly. Safety coaches in radiology: decreasing human error and minimizing patient harm. Pediatric radiology, 40(9):1545–1551, 2010.

[87] J. H. Gurwitz, T. S. Field, J. Judge, P. Rochon, L. R. Harrold, C. Cadoret, M. Lee, K. White, J. LaPrino, J. Erramuspe-Mainard, et al. The incidence of adverse drug events in two large academic long-term care facilities. The American journal of medicine, 118(3):251–258, 2005.

[88] Yle. Medical errors becoming epidemic, 2013. `http://yle.fi/uutiset/medical_errors_becoming_epidemic/6438319/` [Online; accessed 19-Oct-2014].

[89] Yle. Hospital errors need oversight, health care professionals say, 2015. `http://yle.fi/uutiset/hospital_errors_need_oversight_health_care_professionals_say/7892914`[Online; accessed 16-October-2015].

[90] M. Elliott and Y. Liu. The nine rights of medication administration: an overview. British Journal of Nursing, 19(5):300, 2010.

[91] A. J. Jara, M. A. Zamora, and A. F. Skarmeta. Drug identification and interaction checker based on iot to minimize adverse drug reactions and improve drug compliance. Personal and ubiquitous computing, 18(1):5–17, 2014.

[92] C. Chapuis, M. Roustit, G. Bal, C. Schwebel, P. Pansu, S. David-Tchouda, L. Foroni, J. Calop, J. Timsit, B. Allenet, et al. Automated drug dispensing system

reduces medication errors in an intensive care setting. Critical care medicine, 38(12):2275–2281, 2010.

[93] Pharmaca Fennica. Pharmaca Fennica 2014 - Lääketietokeskus. www.laaketietokeskus.fi. 2014. www.laaketietokeskus.fi [Online; accessed 14-Nov-2014].

[94] H. Pirinen, L. Kauhanen, R. Danielsson-Ojala, J. Lilius, I. Tuominen, N. Díaz Rodríguez, and S. Salanterä. Registered nurses' experiences with the medication administration process. Advances in Nursing, 2015, 2015.

[95] F. Tang, S. Sheu, S. Yu, I. Wei, and C. Chen. Nurses relate the contributing factors involved in medication errors. Journal of clinical nursing, 16(3):447–457, 2007.

[96] I. A. Tøndel, M. G. Jaatun, and P. Meland. Security requirements for the rest of us: A survey. Software, IEEE, 25(1):20–27, 2008.

[97] R. A. Caralli, J. F. Stevens, L. R. Young, and W.R. Wilson. Introducing octave allegro: Improving the information security risk assessment process, 2007. ftp://ftp.sei.cmu.edu/pub/documents/07.reports/07tr012.pdf [Online; accessed 23-Feb-2016].

[98] H. Nakashima, H. Aghajan, and J. C. Augusto. Handbook of ambient intelligence and smart environments. Springer Science & Business Media, 2009.

[99] G. Sindre, D. G. Firesmith, and A. L. Opdahl. A reuse-based approach to determining security requirements. In Proceedings of the 9th international workshop on requirements engineering: foundation for software quality (REFSQ'03), Klagenfurt, Austria. Citeseer, 2003.

[100] K. Siau and Z. Shen. Mobile healthcare informatics. Informatics for Health and Social Care, 31(2):89–99, 2006.

[101] R. Zhang and L. Liu. Security models and requirements for healthcare application clouds. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pages 268–275. IEEE, 2010.

[102] J. T. Lounsbury. Application security: From web to mobile. different vectors and new attack. Security in Knowledge, pages 2–30, 2013.

[103] J. Bishop and N. Horspool. Cross-platform development: Software that lasts. Computer, 39(10):26–35, 2006.

[104] R. L. Simpson. Security threats are usually an inside job. Nursing management, 27(12):43–43, 1996.

[105] T. C. Rindfleisch. Privacy, information technology, and health care. Communications of the ACM, 40(8):92–100, 1997.

[106] Netmotion wireless whitepaper. Comparing mobile vpn technologies, 2012. http://www.enterprisemobilitynetwork.com/wp-content/uploads/2012/01/netmotion-wireless-comparing-mobile-VPN-technologies.pdf [Online; accessed 23-Sept-2015].

[107] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. Proceedings of the IEEE, 63(9):1278–1308, 1975.

Appendix A Issues found in the previous prototype of the Smart Dosing application

The Smart Dosing application development project had started before I joined it, and the project had already generated some research papers and one Master's thesis [10] [11]. When I joined this project, I was handed over an already developed application by [11] along with a prioritised to-do list. My first goal was to fix the bugs of the code, make the required changes to enhance the features, and further the application development by connecting the application to the Pharmaca Fennica database [93] and hospital information system. However, the application could not be accessed from the Amazon cloud where it was deployed, as the contract with the Amazon had ended by the time I joined the project. I downloaded the code from the server and tried to run it locally on my PC, but the code refused to work because of the obsolete library files, outdated dependencies, and an old version of the Vaadin framework. The code was developed using Vaadin version 6 and touchkit add-on version 2 which had undergone major changes. Touchkit add-on was essential for the development of Smart Dosing application, as the application was meant to be run on mobile device. However, with the newer releases, there were continuous incompatibility issues with the upgraded Vaadin version. After resolving the incompatibility issues and upgrading the library files used, I

could finally run the application after many unsuccessful attempts.

Taking a look at the first prototype of the Smart Dosing application's functionality gives an idea of what kinds of security, privacy, and usability requirements there should be for such e-health applications. The following is a brief summary of notes from testing Khan's application [11]. These notes may hopefully help in building the next iteration of the application.

The issues detected with the first iteration of the application prototype begin at the homepage. The sign-up function for new users is presented on the application's starting page. Unauthorized individuals are able to create user accounts, and these unauthorized accounts can be untraceable, because any user information is accepted by the system – it is even seen that first and last name fields accept numerical values and special characters. To improve the system, new users should be added to the system only by head nurses, and they should show their work contract and ID beforehand. Input validation is absolutely critical to the security of the application. Therefore, the implementation should be designed to accept only exclusively verifiable personal information of the users.

User names and passwords are not validated with a confirmation email, or by any other way. Email addresses with the hospital domain would most likely be the best user names, but any type of user name is accepted by the existing version. It is hence required that the new users are verified.

There also seems to be no minimum length for passwords, nor a requirement for numbers or special characters in the passwords. When a user logs in, the application even accepts the combination of an empty user name field and empty password field, thus making the system completely insecure. Passwords should be kept encrypted in a database, which is not the case in the current iteration of the application. Secure data storage and transfer of sensitive information need to be kept in mind while developing the next iteration of the application. I have resolved the above stated issues on password

fields, but not the encryption of data.

After logging in, the information security issues change to usability and patient safety issues. The two main functions of Smart Dosing application are filling the medication tray and dispensing the medicines. The application has radio buttons for selecting either filling or dispensing, and one has to be selected before clicking the corresponding navigation bar. If the user accidentally tries to dispense medication while the tray is still empty, the tray image in the application looks empty but provides no option for filling it. The user is compelled to log out and then log in again in order to be able to achieve both functionalities side by side. For the sake of good usability, the user should be able to access all the pages of the application during one session.

When the tray is partially filled, the application shows buttons for returning to the previous tray slot and going forward to fill the next one. If, for example, a nurse fills the second medicine slot and then returns to the first slot, the second slot remains filled in the application. Now on moving forward to fill the second slot, instead of continuing the medicine list to be filled, it refills from the beginning, leading to overdosing. This is not a bug *per se*, but refilling one slot refills all of those slots present before it. It is thus required that filling a medication slot should be effective on only one slot at a time.

After filling the medicine tray, the next step is to dispense the medication. Going back from the dispense view to the fill view is needed in those cases where the nurse wants to cross-check the medications. However, returning to the filling view shows an empty tray where it should show the medicines already filled. Both the filling and dispense views should be identical at any given point of time. Hence, there is a need for the synchronization of the tray's dispense view and filling view.

In the dispense view, the option for skipping medications was found problematic. It made the entire medicine cup image disappear from the tray, instead of making the image of the pill from the medicine cup to vanish. This makes the application unreliable and prone to medication errors, as the information about the skipped medication is no

longer saved for dispensing later, if need be.

There is a need for consistency and a proper flow of information from the fill tray view and dispense medicines view. The history of activities is stored in the database for tracing the filling and dispensing activities. There are cases when it is observed that even when the activity log shows multiple instances of filling the tray and no dispensation activities, the database records filling activity only once. The same activity is not reflected to the user at the front-end. The back-end data and front-end data need to be coherent. The deletion actions, along with other actions, need to be recorded. The back button functionality and the deletion functionality should not overlap.

There are also some navigation issues. If a user chooses to go back one step from the filling view and returns to continue the filling process from where it was left, the medicines filled in the tray are not saved. It is expected that the medications are to be saved across the sessions, so that when another nurse logs in, the filled medicines on the tray are visible and ready to be dispensed. On the other hand, the dispensed tray view appears as a completely filled tray, despite the filling process that had already taken place. This leads to an erroneous treatment process.

The end date of the treatment of the patients is not taken into account while filling the tray, which can cause erroneous treatment for the patients. It is required that the patient whose medication period has been completed, should be out of the list of patients who are due for medication.

While logging in with an invalid username or password, it is seen that the error messages are briefly appearing notifications which fade away as soon as there is any movement by the user on the interface. This leaves the user confused, as there is no direction on the actions to be taken. The error messages should be useful, but should not give away more information than what is required for fixing the error. For example, while logging in, the application should not state which of the given information was correct. Here, it is a trade-off between usability and security. In my developmental work, I have used proper

viewing of error messages and the information they carry.

The application seems to allow dispensation of the medicines strictly in the order of tray slots. There should be an option for selecting a patient for dispensing the medicine when the patient's administering time approaches. Also, in situations where the need for dispensation arises in the middle of the process of filling the tray, the application should allow the user to do so. This inflexibility can lead to skipping a patient's medications at the prescribed times.

The "skip medicines" functionality is both necessary while dispensing and while filling the tray. Also, an option to change medication after checking the patient's condition while on ward round could be helpful.

The application seems to lack the option to verify the medicines before administering them to the patients. Also, due to look-alike images of medications in the tray shown in the application, it is unclear what medicines are filled, and this adds to the confusion when there is a need of filling more than one type of medicine into a single medicine cup. To avoid such situations, an integration of medicines with their corresponding electronic images is essential. The images of different pills in a single cup should be clearly specified. There is a need for integrating the application with the *Pharmaca Fennica* [93] database.

All in all, a series of dysfunctions and faulty designs were identified. It should be underlined that software development is a feedback-driven process where iterative loops follow one after another. Despite the seriousness of the previously explained problems in the Smart Dosing application, all of these problems can be fixed.

Appendix B Description of the new design of the smart medication management system

The technical trouble in running the previous prototype of the application, as well as non-finalised design of the tray and the software application, led me to create a new and better design for the tray and the application. I was greatly helped by my advisor and the nursing team. The new design demanded not only changes in the user interface of the application but also modification of the medicine tray. The development of the application had to be done from the scratch. But before the development work it was necessary to make sure that the tray design complements the application design. For this reason, I borrowed a medicine tray and some cup holders from the hospital. I 3D-printed the cup holders as expected of the new design, and came up with a tray specification as shown in Figure B.1.

The challenge I faced while developing the software application was that the application is expected to run across all the platforms. As mobile applications are on the rise, various technologies have come up with their own mobile frameworks. As intriguing as developing cross-platform mobile application is, these frameworks provide limited support and insufficient tutorials. I tried my hands on frameworks like Qt mobile, JqueryMobile and AngularJs. In the end, I decided to settle with Vaadin, as by that time I

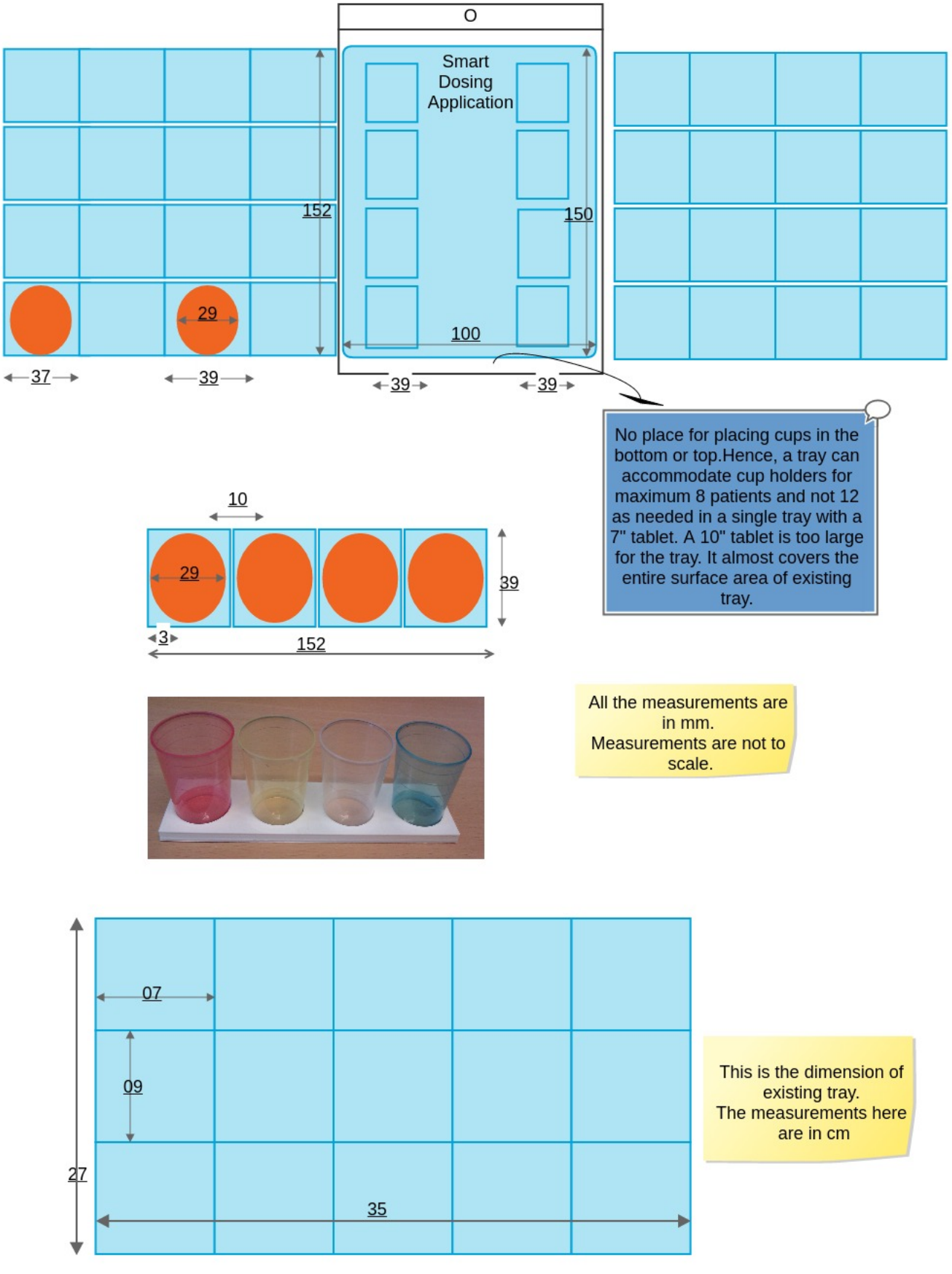

Figure B.1: Medicine tray design specifications

had been successful in running the previous prototype after fixing the errors. Further, I planned to port the old project code to the upgraded version of Vaadin and make necessary changes on the go.

Before I started writing my thesis, I had done the security analysis of the first prototype of the application, developed a hardware prototype of the new tray design, and the design of the application. The development work was not complete, except for the login and viewing medication and treatment details of the patients, . While working on the login page, it was realised that a security requirement document would prove beneficial. The source code of the application is available online [1].

The smart medication management system includes a) a medicine tray, b) a tablet device, running a software application that is placed in the centre of the tray, and c) cup holders surrounding the tablet, as shown in Figure B.2. The cup holder is a new feature in this design. It overcomes the dysfunctions of the cups, such as spilling of medicines, when the medicines outweigh the cup capacity, and it allows a controlled and sequential workflow. One cup holder carries medicines for a single patient. The colour and placing of the medicine cups represent the correct timing for the medicine intake. For example, a red cup could be for morning medicines, a yellow cup for afternoon medicines, and so on. However, currently there exists no standardisation as to which colour represents what time of medication administration. This still varies from hospital to hospital.

The new design of the application being introduced here focuses on the main functionalities, which are the *filling of medicine tray* and the *dispensation of medicines* from the tray to the patients. The design has attempted on concentrating on the nurse-centric activities as the nurses are the primary users of this application.

The state diagram of the proposed design is shown in Figure B.3. As shown, the application gives two choices after a successful login, *Filling the tray* or *Dispensing the medicines*. For filling the medicine tray, the nurses need to see the ward's list of patients

[1] Smart Dosing: `https://github.com/NataliaDiaz/SmartDosing`

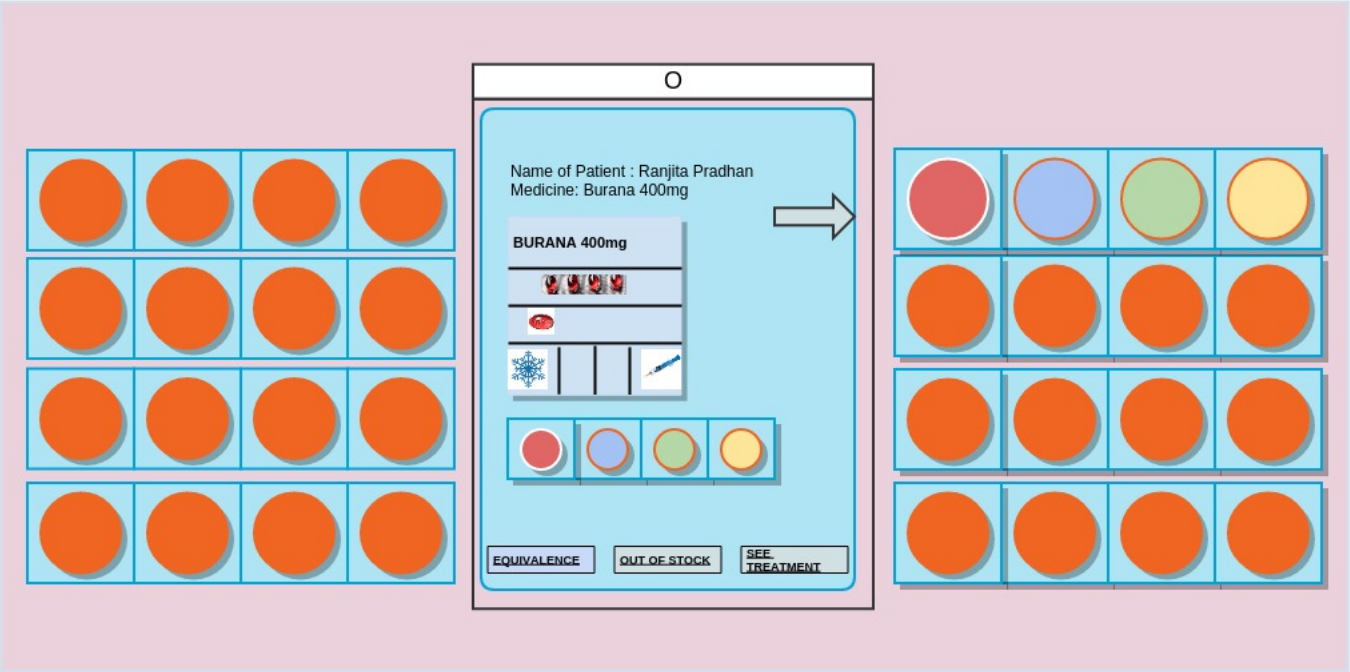

Figure B.2: Medicine tray with tablet device in the centre, surrounded by cup holders.

(LP), for whom the medicines need to be filled that day. The LP also allows adding new patients to the ward and adding more patients to the list. The LP is then rearranged on the tablet screen in the form of iconic elements, leading the user to the next state called patient menu (PM). The PM is essentially built over LP. The detailed view of the PM can be seen in Figure B.2. The iconic elements carry patient names and location details (room number, bed number) to help the nurses navigate through the ward patient-by-patient. Corresponding to each icon, cup holders are placed horizontally on both sides of the tablet. The PM is followed by the list of medications (LM) for one selected patient. This screen carries the list of medicines for the selected patient, the functionality for searching equivalent medicines, placing orders for out-of-stock medicines, treatment details (ongoing medication and medical history), as well as the scheduling and programming of medicines prescribed by the physician. The LM also shows visual information on the prescribed medicines to help the nurses verify the

correct medicines, their dosage, and their timing. It consists of electronic images of the prescribed medicine packages, pills, timings with colour-coded cups, along with the same picture of the horizontal slots. The visual verification can ease the flow of work and eliminate unnecessary confusions. The LM also reminds the nurse about controlled drugs, injections, liquid medicines, and refrigerated medicines which are stored away from the regular medicine cabinets. These are shown on the screen as glowing notifications.

The application's search functionality for equivalent medicines would save the nurses' time and effort, as they no longer have to leave the medicine room to browse through a pharmacy book or the Internet. In addition to that, results obtained via the application search function will be accurate and neatly available, as compared to searching from an entire textbook of medicine names or the Internet. Making online searches and browsing books result in a large number of generic substitutions. As pointed out by Díaz Rodríguez et al. [10], when there is confusion between two drugs with similar names, it increases the risk of drug errors. It is therefore required that the Smart Dosing application will be integrated with Pharmaca Fennica [93] in order to find medicines or generic substances easier and faster. Picking or selecting from fewer available options will also reduce the error possibility.

In addition to the search for equivalent medicines, an option to place an out-of-stock order with the in-house pharmacy is proposed to help nurses work uninterruptedly, thereby cutting down the probability of making errors.

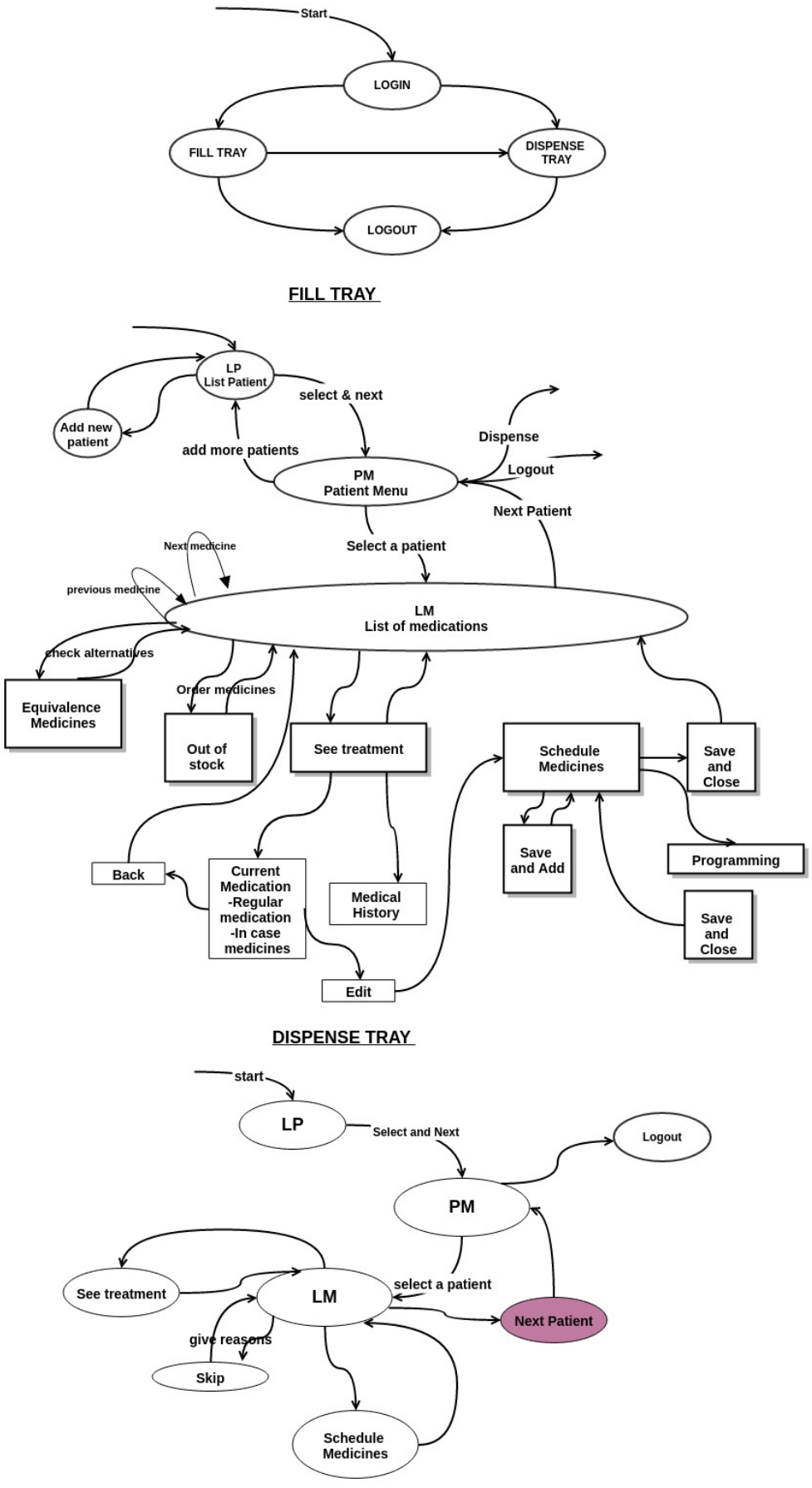

Figure B.3: State diagram of the Smart Dosing application.