



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MUHAMMAD OMAIR FATIMI
COMPARISON OF MOBILE AND NATIVE TECHNOLOGIES FOR
MOBILE MES APPLICATIONS

Master of Science thesis

Examiner: prof. Kari Systä & AP.
Minna Lanz
Examiner and topic approved by the
Faculty Council of the Faculty of
Pervasive Computing
on 13th January 2016

ABSTRACT

MUHAMMAD OMAIR FATIMI: Comparison of Mobile and Native Technologies for Mobile MES Applications

Tampere University of technology

Master of Science Thesis, 71 pages, 79-96 Appendix pages

May 2016

Master's Degree Programme in Information Technology

Major: Pervasive Computing

Examiner: prof. Kari Systä & AP. Minna Lanz

Keywords: MES, mobile MES, portable MES, Web vs native, HTML5 vs android

The MES (manufacturing execution system) is mostly used from desktop based terminals in a factory. These terminals are distant from the machines and materials used on the factory floor. To access the information available through MES from anywhere in the factory floor, use of mobile terminal instead of desktop computers has been proposed. To evaluate two alternative implementation technologies, Web and native, we have developed and compared two prototypes of the MES application. In addition, we have studied the advantages of native and Web approaches through the literature and survey.

Mobile devices are categorized by its different platforms and screen sizes. Android, iOS, and Windows phone are most common among them. Mobile applications are platform dependent and an application made for one platform does not work on others. Web applications are platform independent that work on all devices. HTML5 has introduced some APIs through which a Web app can behave like a native app and can compete with the native app. So, in this thesis we have tried to compare Web and native app and tried to find out which is better for MES applications. A general answer to this question is native because of its better performance. In this thesis, we have analyzed some of the factors that are responsible for the performance difference between a Web app and native app. In addition to this, we have had an online survey to find out what developers think about the development, testing, maintenance and deployment of Web and native technologies.

Based on all the data, i.e. literature review, some experiments, feedback from participants and online survey, we made a conclusion that native app is the best solution for mobile MES because native app is more responsive and more secure. However, native apps require more time, effort, cost and skills to be developed and maintained.

PREFACE

This thesis was done for TUT but the company case came from Finn Power to solve the problem with their existing terminal-based MES system. During this thesis, I have been able to broaden my knowledge on Web and native technologies. I have got some hands on experience working on Java and AngularJS which seems to be quite promising for my future. After this thesis work, I am able to analyze the factors on which the native technologies can differ from Web.

I am so thankful to both of my supervisors, prof. Kari Systä & AP. Minna Lanz for giving me the opportunity to work with them on this topic. They were always there when any kind of help was required. I am also thankful to all the friends and colleagues who took part in online survey and heuristic evaluation test respectively. Last but not the least; I am so grateful to my parents who supported me throughout my entire life and kept me motivated during this whole period of thesis work.

Tampere, 25.5.2016

Muhammad Omair Fatimi

CONTENTS

1.	INTRODUCTION	1
1.1	Thesis Description.....	2
1.2	Research Question.....	2
1.3	Structure of the Thesis.....	3
2.	MOBILE BASED MES	4
2.1	MES.....	4
2.1.1	LeanMES	4
2.1.2	Need of a Portable MES.....	5
2.2	The FinnPower Case	6
2.3	Problems with the Existing System.....	7
2.4	Native or Web MES	7
3.	APPLICATION PLATFORMS	10
3.1	Mobile Platforms and Their Market Shares	10
3.1.1	Architecture of the Android Platform	11
3.1.2	iOS	13
3.1.3	Windows Phone	14
3.2	Web Applications.....	16
3.2.1	Web Development Frameworks and Concepts.....	19
3.2.2	HTML4 vs HTML5	21
4.	IMPLEMENTATION OF MOBILE BASED MES	23
4.1	Manufacturing Process	23
4.2	High Level Architecture.....	24
4.2.1	Concept Diagram of Mobile MES	24
4.2.2	Sequence Diagram of the Processing of Manufacturing Order	27
4.3	Application Backend	29
4.4	Frontend	30
4.4.1	Web	31
4.4.2	Native	31
4.5	Features	32
4.5.1	User Profile	32
4.5.2	Location Using QRCode.....	33
4.5.3	Item Detection Using BarCode	33
4.5.4	UI Adaption (Non-Functional Property).....	34
4.5.5	Filters	35
4.5.6	3D Visualization	36
4.5.7	Other Features	36
4.6	Application Usability Evaluation by Participants	37
4.6.1	Heuristic Evaluation.....	37
4.6.2	Extra Features	38

5.	COMPARISON	39
5.1	Performance	39
5.2	Frames per Second	42
5.3	Tap Delay	48
5.4	WebGL is Handy but Slow and a Potential Risk	51
5.4.1	WebGL.....	51
5.4.2	Performance	52
5.4.3	Security	52
5.4.4	OpenGL ES vs WebGL.....	54
5.5	Online Survey (the developer’s perspective)	55
5.5.1	Cross Platform Compatibility	55
5.5.2	Easy of Development, Maintenance and Testing.....	55
5.5.3	Access to Device Features	56
5.5.4	App Stores.....	56
5.5.5	Rating by Developers.....	56
5.6	Interface and Performance Evaluation by users	57
6.	ANALYSIS	59
6.1	The MES Mobile Interface.....	59
6.2	Evaluation by Users	61
6.3	Performance (JavaScript vs Java)	62
6.4	Rendering Performance.....	63
6.5	WebGL, 3D Animations	65
6.6	Tap Delay	65
6.7	Device Features	66
6.8	Cross Platform Compatibility.....	66
6.9	App Stores	67
6.10	Conclusion.....	67
7.	SUMMARY	69

APPENDIX A: Online Survey (The developer’s perspective on web vs native technologies)

APPENDIX B: Evaluation Test by Participants

APPENDIX C: Screen Shots of Mobile MES

APPENDIX D: JavaScript Code for Performance Test

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Interface
APS	Advance Planning and Scheduling
AJAX	Asynchronous JavaScript and XML
AsyncTask	Asynchronous Task
CSS	Cascading Style Sheets
CPU	Central processing Unit
CNC	Computer Numeric Control
CLR	Common Language Runtime
CORS	Cross Origin Resource Sharing
DTD	Document Type Definition
DAO	Data Access Object
DoS	Denial of Service
DOM	Document Object Model
ERP	Enterprise Resource Planning
EJB	Enterprise JavaBeans
FPS	Frames per Second
GPS	Global Positioning System
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
iOS	iPhone Operating System
IDC	International Data Corporation
IDE	Integrated Development Environment
IE	Internet explorer
JS	JavaScript
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MES	Manufacturing Execution System
MVC	Model-View-Controller
NT	New Technology
NC	Numeric Control
OS	Operating System
ORM	Object Relational Model
OpenGL	Open Graphics Library
OpenGL ES	OpenGL for Embedded Systems
PC	Personal Computer
POJO	Plain Old Java Object
QRCode	Quick Response Code
REST	Representation State Transfer
SMEs	Small and Medium Size Enterprises
SCADA	Supervisory Control and Data Acquisition
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
UML	Unified Modelling Language
UI	User Interface
URL	Uniform Resource Locator

USB	Universal Serial Bus
WHATWAG	Web Hypertext Application Technology
WebGL	Web Graphics Library
W3C	World Wide Web Consortium
XML	Extended Markup Language
XHTML	Extended HTML

1. INTRODUCTION

Research work in the area of mobile technologies, networking technologies and lowering of mobile data package cost have increased the use of smartphones and it is growing day by day. People are using phones for online shopping, to find the route on GPS (Global Positioning System)-enabled phones and for validating some coupons or tickets [1]. Moreover, they are using their mobile devices for activity tracking, fitness purposes, diet planning etc. Mobile devices have now become an integral part of people's life. The number of mobile phone users is growing by 42% every year but there is still much room available [1].

The emergence and continuous advancement of HTML5 (Hypertext Markup Language 5) and other related technologies like CSS3 (Cascading Style Sheet 3) and various JavaScript development frameworks like AngularJS, ReactJS, BackboneJS, PolymerJS etc. have brought revolution in the area of Web development. These inventions have made Web technology more powerful that can compete with native applications in functionalities, look and feel. However, HTML5 still cannot be compared to the native applications when it comes to the device feature accessibility and performance. According to the CEO of Facebook, Mark Zuckerberg, **“Betting too much on HTML5 was our biggest mistake”** [2].

The developer who is building a mobile app probably asks himself “how many devices would be running my app?” [3]. Mobile devices are highly divided into fragments by their operating systems, versions, manufacturer, and screen size. Every platform has its own programming language, development tools, and app store. An application developed for one platform doesn't work for others. Developing applications for each platform takes time if it is supposed to work on all devices. One possible solution might be to develop a Web app. The Web app is usually developed with HTML5 and JavaScript APIs (Application Interfaces) that provides a cross-platform feature. Another solution is the hybrid application, but hybrid applications were not taken into consideration in this thesis.

So, generally, there are three types of mobile applications: native, Web or hybrid. Each of them has its own benefits and drawbacks. Native apps are downloadable and developed for a specific operating system installed on the device and run directly on a mobile device. The Web application doesn't need to be downloaded, it runs on a browser and a single Web app can work on all devices, it is not platform-dependent. Hybrid applications have properties of both native and Web. Hybrid apps are installed on a device just

like native app but they run in a platform's Web view. Hybrid apps are platform independent and run on all devices. They can access core hardware of the device; however they have limited access to the device hardware as compared to the native applications.

But, can these cross-platforms tools provide all the features as native apps do, or Web app can be an alternative solution of a native app? The thesis provides a detailed analysis by comparing some core features of both technologies. The results achieved by this comparison analysis will be used by Finn Power that will help them to develop their MES (Manufacturing Execution System) system based on mobile.

1.1 Thesis Description

This thesis is the study of the mobile application development for the industry. However, the main focus is to study the features of native and Web applications and compare them using a set of criteria. A prototype of a client application was developed for a MES system, this prototype is the part of the novel idea "LeanMES". The prototype was developed for a FinnPower MES system which already has a similar client application for a desktop PC. FinnPower is one of the partners of the LeanMES project. They are a leading manufacturer of laser systems for industrial application.

The thesis has basically two main parts: development of the mobile user interface for the LeanMES concept and comparison of HTML5 and native technologies on the basis of the test application developed for the Finn Power. So, two versions of the application were developed: native and Web. The native version was developed for Android platform while Web application is generic which means it can run on all platforms.

1.2 Research Question

The main area of research for this case study is to compare Web and native technologies (as mentioned above) on the basis of some core features of MES system. In manufacturing industries, the applications might have an intensive use of hardware like camera, GPU (Graphics Processing Unit), Magnetometer etc. Also, offline operations are needed since the network connection is not available everywhere in a factory floor. In some cases, indoor navigation might be needed for someone who doesn't know the map of the factory. The performance of the application is also important since lots of manufacturing processes run at the same time, so knowing the status of every process in real-time is of primary importance. Thus, it is important to have smooth and deterministic responsiveness through UI (User Interface). In addition, push notifications are also an important part to get notifications about the status of the processes. In this research, we will come up with the best solution that is best to implement all these features. We are using design science research methodology in this research. We have created two apps:

Web and native, Web was created using HTML5 and native was created using Java in Android studio. We have tested some features using this test application. Moreover, this comparison is not limited to the implemented features in test application but also we have taken into account other aspects that can play a vital role to differentiate both these technologies.

1.3 Structure of the Thesis

This thesis is structured as follows: Chapter 2 is about the need of mobile based MES and background study from literature. Chapter 3 gives an overview of mobile application platforms. Chapter 4 consists of the implementation of MES in mobile: Web and native. Chapter 5 discusses the differences between Web and native applications based on the previous literature review, experiments, online surveys and usability evaluation. Chapter 6 is the analysis of those aspects which we considered in chapter 5 that are responsible for differentiating the two technologies. Chapter 7 is the summary of the thesis.

2. MOBILE BASED MES

2.1 MES

According to Wikipedia [42], MES is computerized system that is used in manufacturing to track and report all the information needed to transform raw materials into a finished product. MES is used to control and manage the workflow on a factory floor. It keeps track of all the processes and up-to-date information coming from other available sources like ERP, machine monitors, APS (Advance Planning and Scheduling) and others. The goal of MES is to improve the efficiency and productivity of work-in-progress in a factory floor.

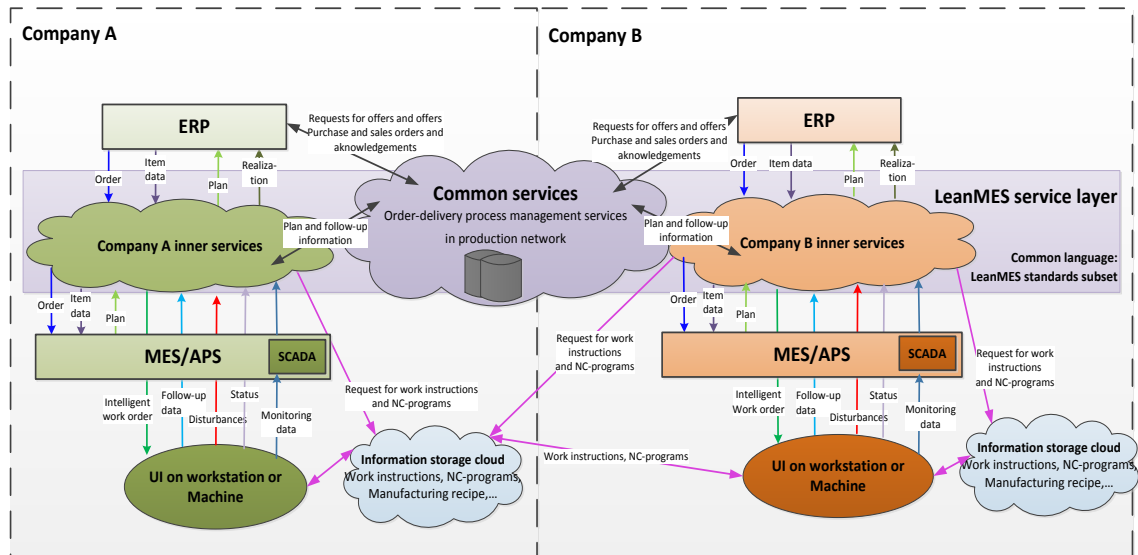
2.1.1 LeanMES

LeanMES is one of the six sub-projects in the FIMECC MANU [78] program. The main idea of the project is to develop a solution that provides a lean, scalable and extendable solution for a new type of MES (Manufacturing Execution System) that supports the human operator in a dynamically changing environment [4]. This allows SMEs (Small and Medium Size Enterprises) to work together in a better way, in a dynamically changing production environment by making the full use of the skills and knowledge of the operator through some flexible interfaces in his daily work on factory floor [4]. The core features of LeanMES involve real-time tracking of manufacturing processes, transparency of production orders and balance data of materials, human-centered production, intelligent work orders and utilizing the capabilities and skills of operators working in a factory.

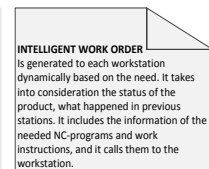
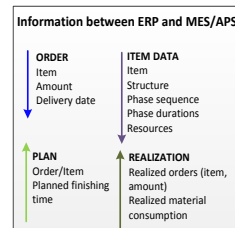
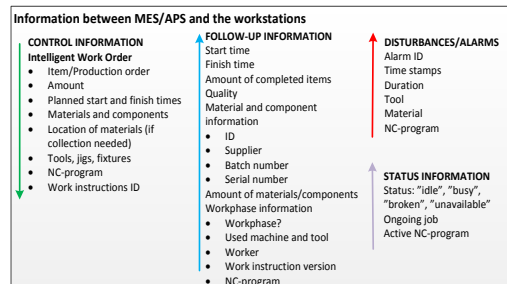
The LeanMES is an extended concept of core Lean concept which is utilizing the Lean principles in MES. The consideration of human in order management and processing is the idea but the concept is more a process-oriented than human-oriented. Process-oriented means LeanMES focuses on manufacturing processes and human-centered means; it tries to maximize the performance of manufacturing processes by utilizing humans efficiently.

Visibility of the information enables the system to process right order according to the available amount of required resources in the factory. Consequently, it helps operators to have right information of order at the right time and hence the transparency mechanism proves to be most vital part in MES. In addition to transparency, LeanMES uses

some high-level interfaces to track and monitor the manufacturing processes. For example, SCADA (Supervisory Control and Data Acquisition) is used for monitoring data from the workstation and watch/mobile interface is used for visualization of the processes. The LeanMES has an inner service layer which can be efficiently used for production management on the factory floor. The common service layer is used by companies for supply chain management as shown in the figure given below.



LEGEND:



LeanMES Service Layer is a collection of modular services, which acquire information from different systems from different actors, perform reasoning based on that information and deliver information to those who need it.

Information storage clouds act as information storages, from where the service layer gets and distributes the relevant information to the workstations when needed. The actors on the network may have their own clouds or other information storages, to where they may, or may not, offer access to their partners.

Figure 1. LeanMES Concept Diagram [4]

2.1.2 Need of a Portable MES

The term ubiquitous means everywhere and when we talk in terms of technology then it means technology is available everywhere. The technology can be in the form of tablets, laptops, and mobile phones etc. which are portable and can be accessed from anywhere. Field engineers or operators, working on a factory floor, have limited information available to perform task [5]. Instant information is needed to find out the right equipment needed to diagnose a certain problem [5].

Mobile computing is a promising choice to access information to operators who would be able to work smartly and remotely. The emergence of camera, digital maps, and sound recorder in the mobile device has made them even more promising that can help them perform daily work orders, service reports and documentation more efficiently [5]. However, the adaptability of a new technology won't be too easy especially for those who have less experience in using smartphones.

Indoor navigation is helpful for the new workers in a factory. The new workers usually do not have idea about the map of the factory, so, this could help Operators to navigate to any place in a large factory floor using indoor navigation in mobile MES. This helps operators to find meeting rooms, stores, and other places in a factory.

There is always a high risk of working in plants like nuclear power plant where safety precaution is most important. Some of the companies are very large in size and comprised of many buildings located at different places. The GPS technology in a mobile can facilitate managers and safety team to track the position of their employees in a plant which is necessary to know how long they have been there [6]. GPS technology in a factory can also be used to know if the operator is working correctly at a correct place [6].

With the enormous amount of benefits of using smartphones for manufacturing, companies are moving towards mobile technologies. In the near future, industries will be relying more on mobile based system like mobile based ERP (Enterprise Resource Planning) and MES. Manufacturers are gradually heading towards mobile applications and in next three to five years, mobile applications will be adopted by most of the manufacturers to offer value-added services to their clients [7].

2.2 The FinnPower Case

The existing MES system in FinnPower is named as Power Processing. The Power Processing is used for order management, machine programming, and time scheduling. It connects the centralized execution system to the user interface. The operator who is responsible for order management controls all the order using this interface.

Power Processing is a desktop based MES that always requires a desktop PC to run. The problems with the existing MES system, as described in section 2.2.1, made them think to develop it with some other technology. Since most of the problems are associated with mobility and ubiquity of the system, it was decided to implement MES using mobile technologies but in mobile, we have two options: mobile Web app and native app. The main goal of this case study is to compare both these technologies and to come up with the solution that can overcome the problems in the existing system as discussed in section 2.2.1. Besides implementing some core features of this current system, they also

want some other features to make it more useful in a production environment. The test application also includes those extra features as suggested by FinnPower

- **QRCode:** it was implemented to find places within factory. QRCodes can be scanned using mobile device.
- **3d Visualization:** it is for visualizing the image of item in an interactive 3d view.
- **Indoor Map:** the indoor map of the factory.

2.3 Problems with the Existing System

Desktop Based: Desktop based software which always requires a desktop PC to run.

Limited Terminals: Terminals are limited, so the operators do not have access to the terminals all the time.

Terminal at Distance Locations: Terminals are located at distant locations, so an operator has to move a lot in order to access terminal.

Information Overflow: The instructions given by the MES, in order to perform a manufacturing process cannot be memorized by the operator.

QRCode/Barcode: The system doesn't have any implementation of QRCode/BarCode, which is needed to detect an item in manufacturing industries.

3D Visualization: Interactive 3d view of product/part was not present in the current system.

2.4 Native or Web MES

The decision of having the mobile based MES still leaves one question unsolved, i.e., whether to have a Web-based MES or a native MES app? Since mobile apps can be in many forms: native, Web and hybrid. So, we continued exploring all these technologies by studying some related work focusing the differences between native and Web application.

HTML5 has its own advantages when it comes to cross-platform functionality and low development effort. The cross-platform compatibility is actually the reason of low development effort that saves developers' time to build different versions of the application for different platforms. The low development cost and platform independency can attract several organizations who want to serve a huge number of customers without writing several implementations for every platform [15].

Development of native app is not easy due to the huge fragmentation of mobile devices according to mobile operating system, size and manufacturers. Each fragment requires the application developer to develop a separate application, as every fragment has its own development tools and language. Web application or hybrid applications do not have this problem, since they require developers to know only HTML5 and JavaScript. Learning Web app development using HTML5 has made programming very simple and easy. Therefore, in the past years, HTML5 has gained some popularity among non-programmers or less-experienced programmers. Integration of existing frameworks and APIs like JQuery mobile and google map APIs has made programming much simple for the less-experienced people [14].

Researchers found out benefits and drawbacks of each technology in the context of their research topic, but still many of them believe that in the upcoming days, the Web technology will take the place of native apps. According to researchers, due to the continuous development of Web technologies, it is more likely that in the next few years the Web technology will be the main development platform for developers [14].

Applications relying heavily on device hardware are mostly developed on native platforms. This problem has been reduced to some extent after the emergence of the hybrid app, but still, hybrid apps cannot be compared to the native app. A native application can make full use of device hardware. Native applications are more effective in accessing device hardware like GPS, cameras, gyroscopes, microphones, and other device built-in applications like contact book and calendar [16]. Device file access, local notification, and alarm manager are not possible in Web technologies using JavaScript and HTML5 [16].

However, the decision to make the Web or native app does not only depend on the functional requirements but also the available budget of the company and skills of their employees and of course the time to complete project [16].

A different performance tools are available for measuring the performance of the Web, for example, Google V8 Benchmark Suite and SunSpider JavaScript Benchmark are very famous tools to minimize the execution speed of the Web [12]. Graphics rendering is measured by measuring the frames per second (FPS) of different graphic objects on HTML canvas [12]. However, the performance of graphics rendering can be different in mobile and laptop browsers. Spaceports.io published a study that compares graphics rendering in mobile and laptop browsers [13]. According to their results, the graphics rendering on different mobile platforms browsers is, on an average, 889x times slower than in laptop browsers.

One of the most attractive features of native applications is that they have their own app stores. Using app store user can search and download an app with a single touch. This is

also helpful for those mobile users who have little or fewer skills in Web searching. In Web searching, users go through different search engines and websites to find the required app. App stores that work like a one-stop shopping center, through which a user can buy the app with just a single click, has increased the market of mobile apps [9]. Besides searching feature, the app store has also been a good source for developers to monetize their applications. There are various methods available, to get payment from users. A one-time payment method that asks the user to pay and unlock the application for good, and in-app and subscription methods offered by some platforms [10]. Besides these app payments, advertisements and sponsorships (tradition earning method through websites) are the other ways to monetize the application.

The efficiency of GPS location was tested using a mobile Web app in Kenya. The result shows that the GPS position shown by the mobile Web app was incorrect. According to the result, the GPS location functionally was ahead a few hundred meters [8]. This was due to the fact that GPS location jumped few meters away in different directions before they actually started to run. So the distance app recorded the distance even before they started the race and they eventually switched to the native application [8]. The test was also held in Europe and the result confirmed the outcome of the test held in Kenya [8]. However, we believe that the result is probably due to some inefficient GPS algorithm which would not accurately track your position. HTML5 provides an option which can be enabled to have an accurate GPS location. This can be achieved by setting **enableHighAccuracy** to true.

Applications that carry sensitive data like banking app, healthcare app, ERP application etc. need a special care of their security. The most dangerous vulnerability in a Web app is cross-site scripting; this threat is caused by the inadequate validation of data from untrusted sources [11]. Firewall has been one of the solutions in this case but it is not a complete solution since it can only block ports.

The most important thing in any kind of system is its ability to deliver the required service or information in an efficient and responsive way. Most of the users don't like to wait for a screen or page to show up, so the speed of the applications is one of the most prominent features that distinguishes Web and native applications. In general, it is the common belief that native applications are faster than Web applications because Web applications run on the browser and native app can talk to the device operating system directly.

3. APPLICATION PLATFORMS

At present world, smartphones come with different operating systems including Android, iOS, Windows, Symbian, Blackberry. Every platform has its own architecture, framework, language and tool. In this chapter, we will discuss the architecture of Android, iOS, and Windows mobile platforms.

3.1 Mobile Platforms and Their Market Shares

Today, the most well-known platforms are Android, iOS (iPhone Operating System), Windows and Blackberry. Mobile operating systems were built on top of personal computer operating systems by combining the features of PC operating system and some features of mobile like touchscreen, GPS, magnetometer etc. Jean-Louis Gassée, the former executive of Apple, argued that the operating system doesn't matter anymore, what matters is the user experience and the development ecosystem of particular mobile platform [22]. The continuous growth of Android platform took away the market shares of apple, Symbian and RIM (Blackberry OS) [22]. According to the data collection from IDC (International Data Corporation), the Android leads the market with 82.8% share in the second quarter of 2015 [23]. The data collected by IDC research is shown in Table 1. According to them, the iOS market share has dropped by almost 3% in last three years. Similarly, market shares of other platforms have dropped in the past three years. However, the Windows platform has shown some promise in last 1 year and its market share remained constant.

Because our goal is to compare native and Web technologies, it is better to understand the architecture of native platforms and how the native applications are built using available frameworks. The next section is about the architecture.

Table 1. Smartphone Market shares [23]

Time	Android	IOS	Windows Phone	BlackBerry	Others
2015 Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014 Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013 Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012 Q2	69.3%	16.6%	3.1%	4.9%	6.1%

3.1.1 Architecture of the Android Platform

Android OS runs on top of Linux Kernel. Android is a stack of software components divided into 4 layers as shown in Figure 2.

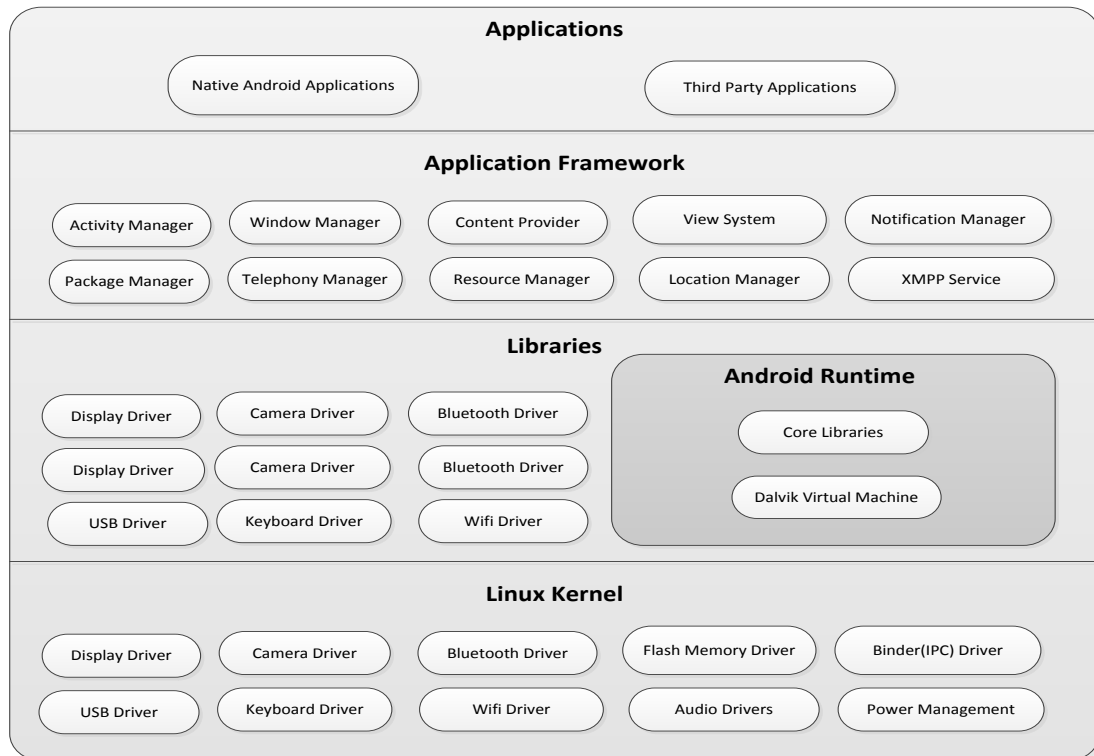


Figure 2. Architecture Diagram of Android OS [41]

Application layer lies on top of the stack. Android has a built-in set of core applications including SMS program, email client, calendar, maps, browser, contacts etc. [17]. All these applications and other applications developed for Android are installed on this top layer of the Android framework.

Application Framework layer provides high-level services to the applications. The framework was designed to simplify components reusability, i.e., any application can publish its capabilities which can then be used by other applications [17]. The developers have full access to these services and can use in their applications. Some of the major services include activity manager, content provider, view system, libraries, android runtime and Linux kernel.

Activity Manager is responsible for controlling the lifecycle of the applications and providing navigation back stack [18]. The back stack is a stack which contains all the activities in background and user can navigate to the activity contained at the top of the back stack. For example, a user navigates to the activity C starting from activity A, i.e. A -> B -> C. The current active activity and top of the stack is C. Now, user presses the

back button, the current activity, which is C, is destroyed and top of the stack becomes B. The activity Manager navigates user to activity B on pressing of back button.

Content Provider allows applications to publish their own data or share data with other applications.

Resource Manager is used to access non-code resources like strings, color settings, and layouts.

Notification Manager allows developers to develop their own custom display alerts or notifications and show them to the user.

View System contains a set of views which can be used to develop graphical user interfaces.

Android platform has some C++ core libraries on top of Linux Kernel, which can be accessed by developers through application framework. Some of the key libraries include browser Web engine Webkit, Libc, SQLite database used to store application data, media libraries for audio and video and security, 3D libraries such as OpenGL for rendering 3d and vector objects.

Android runtime lies on the same layer as libraries. Android has Dalvik Virtual Machine which is very similar to JVM. JVM is platform independent execution environment. JVM is an abstract machine that converts java bytecode into machine code and executes it. DVM was particularly designed for Android. DVM uses the core functionalities of Linux like memory management and multi-threading, which enables every Android application to run in its own process, i.e. every application has its own instance of DVM.

Android also has a set of core libraries which offer the functionalities of Java core libraries [17]. This set of libraries enables developers to write Android applications using Java programming language.

Linux Kernel layer lies at the bottom, which provides the level of abstraction between device hardware and remaining software stack [17]. All hardware drivers are contained in Linux Kernel, such as camera, keypad, sound, graphics etc. Linux Kernel is also responsible for memory management, network management, process management and power management.

There are four basic components on which an Android application is based on. These components include activities, services, broadcast receivers and content providers. Each of these components is defined in the manifest file.

The service is an Android component that runs on a background. Services don't have any GUI, so they are often used with other components like activities [19]. Services are used for a long running process to avoid blocking of user interaction with GUI, i.e. it is a non-blocking process. One good example of a service might be the music running in a background while the user is performing other tasks.

Content providers are used to manage data within application or share data with other applications. Data can be stored in many ways, i.e. in a file system, database or some other place. Applications use content provider through the methods of content resolver.

Broadcast Receivers respond to the broadcast message from other applications or from the system itself. A typical example is updating the display on battery low broadcast message from system [19].

Activity is one of the most important building blocks of Android application. Activity object is responsible for creating windows where GUI (Graphical User Interface) is shown up. An application may contain a single or many activities. If an application consists of many activities, we need to define the activity that should show up when the application is launched.

Other components include fragments, views, layouts, intent, resources and manifest. Fragments represent the portion of an activity. An activity may have multiple fragments or the same fragment can be used in multiple activities. Views are the UI elements that show up on the layout. View can be added from code or it can be defined in XML (Extended Markup Language) files. Layouts are the hierarchies that manage the structure of UI. Similar to views, it can also be defined in code at run time or it can be added to XML files. Intent exactly meant what it describes, i.e. intention to do something [20]. It is a messaging object that is used to request an action from another application component [21]. Resources are the external elements that are kept separate from the code, e.g. images, colors, string etc. and Manifest is used to configure the application.

3.1.2 iOS

iOS operating system of Apple was originally released by Apple in the year 2007. It was built on top of MAC OS. iOS is not only limited to iPhone but also a platform for iPad and iPod. Compare to the Android platform, iPhone is not so open to development perspective. It restricts the developers to publish applications that use private APIs provided by iOS [24].

iOS is layer-based architecture with lower levels providing the core and fundamental functionalities and services. iOS is a 4-layered architecture.

Cocoa Touch is the top layer which contains the frameworks required to build iOS application. It provides the infrastructure and other high-level services like push notifications, multi-tasking, inputs based on touch [25]. Media layer provides all the services of graphics, audio and video. The core service layer provides fundamental services which have no effect on the interface of the application. This layer contains services like iCloud, networking, location etc. The lowest level layer is the core OS on which the other technologies are built on. The core OS of iOS is Mach BSD UNIX Kernel [83]. iOS is basically based on OS X which is a variant of BSD unix kernel running on top of Mach, a micro kernel.

Core OS layer is responsible for managing security, sockets, filesystems etc. These services are essential parts in every application. Even if users don't use them directly in the application, they are used indirectly. The layer also includes core features like Bluetooth access, image processing, algebra etc.

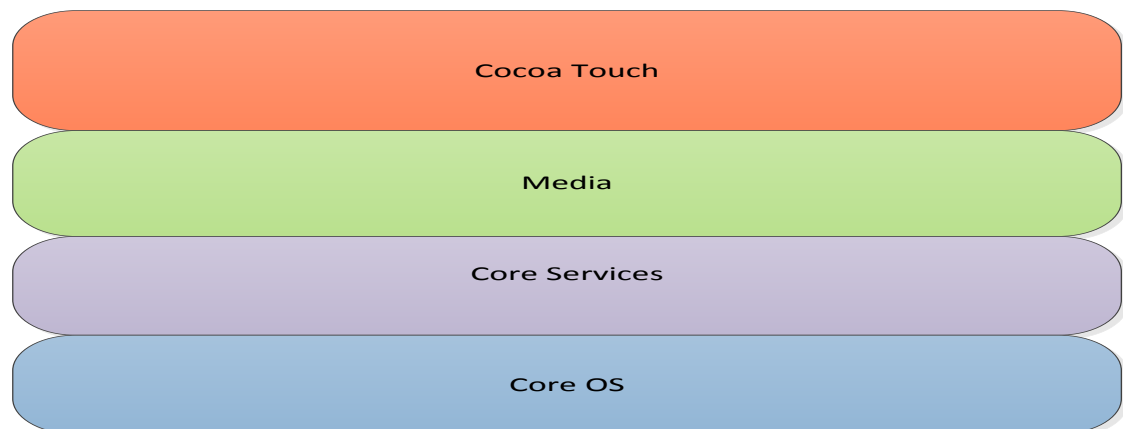


Figure 3. Architecture Diagram of iOS [82]

3.1.3 Windows Phone

The development of Windows phone started in the year 2008 when Microsoft reorganized their Windows phone division. The first version of Windows phone, known as Windows phone 7, was released in the year 2010 by Microsoft [69]. The Windows phone division started working on Windows phone with the aim of better user experience and usability like the touch screen and social media [38]. The Windows phone comes with the “METRO” theme which gives unique user experience to the user.

Windows phone 8 belongs to the second generation of Microsoft Windows phone which uses Windows NT (New Technology) Kernel unlike Windows 7 which uses Windows CE-based architecture. The architecture of Windows phone 8 is shown in the figure.

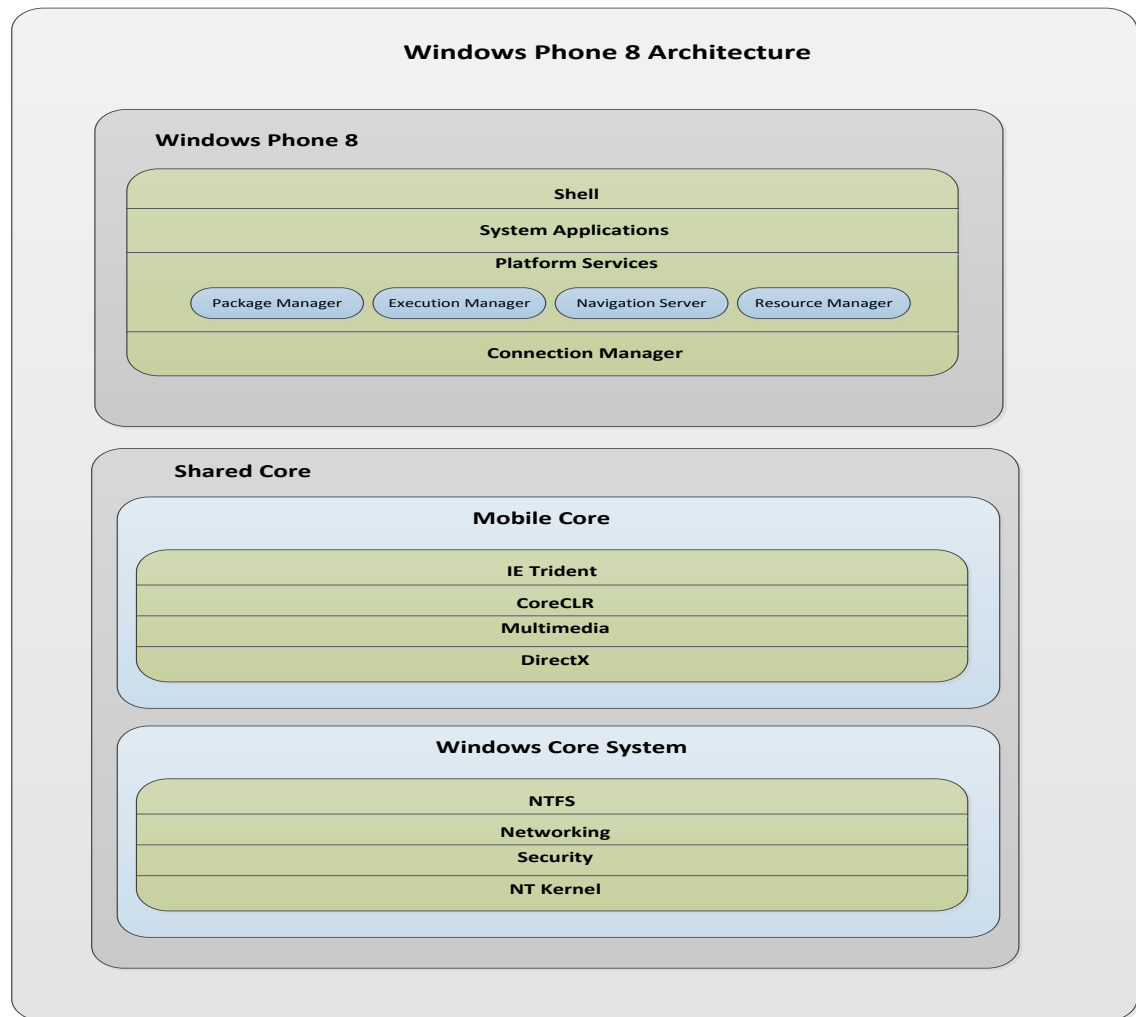


Figure 4. Architecture Diagram of Windows Phone 8 [39]

The shared core layer lies at the bottom which is a core part of Windows NT kernel. This layer is further divided into two components, i.e. Mobile Core and Windows Core System. Windows core system shares the basic functionalities like NTFS file system, networking security and NT kernel of Windows. Mobile core component shares the functionalities of the Windows which are not present in Windows core but still relevant to smartphones [40]. These functionalities include multimedia, DirectX, core CLR (which is similar to CLR (Common Language Runtime) of .NET environment) and IE Trident (a layout/rendering engine for Microsoft internet explorer).

The Windows phone layer lies above Shared Core that is the top layer of the stack. It contains all the external and built-in systems apps like music player, contact book, alarm manager etc. It also contains connection manager, Windows shell and platform services. Platform services are further divided into four parts, i.e. Package Manager, Execution Manager, Navigation server and Resource Manager.

Package manager takes care of installation and uninstallation of the application and maintains their metadata when they reside in the phone.

Execution manager, as the name suggests, is responsible for controlling all the logic associated with application execution lifecycle. Application states messages such as startup, shutdown or deactivation, and host processes are established by execution manager [39].

The movements between all foreground apps are controlled by Navigation server [40], i.e., it handles states of the applications by telling execution manager which application to launch or reactivate. It is responsible for keeping the track of navigation stack [39]. For example, when a user taps app tile from start screen, he navigates from the starting app to the app he just tapped for, and the navigation server asks the execution manager to activate the chosen app. Similarly, when user presses back button, the navigation server asks the execution manager to activate the app you started previously.

Resource manager manages all the resources of the active process to make sure that the phone doesn't get slow. It focuses the CPU (Central processing Unit) and memory usage of all the processes and if a certain application misbehaves and exceeds the allocated memory or space, it just terminates the application.

3.2 Web Applications

The Web applications are programs that reside on remote server and are delivered to the internet on user's request through browser interface. The programming language that is used for the development of client is HTML and JavaScript, and for styling we use HTML attributes or CSS. In order to fetch the page, the browser sends HTTP request to the server which is mapped to the available resources in the server and then server sends the HTTP response. The server typically has a three-tier architecture: data layer, business layer, and presentation layer. The HTML tags represent the HTML elements while CSS is used to format them.

DOM is cross-platform and language independent application interface which is used to dynamically access and update the style, content, and structure of XML or HTML [27]. DOM defines the logical structure of the document which is quite similar to a tree which allows the programmer to navigate, access, modify and delete elements in the document. There are several DOMs exist, e.g. legacy, W3C (World Wide Consortium) DOM and IE 4 DOM, however, W3C DOM is the standard set by W3C, that is able to access and update all content in a document and is supported by all browsers.

The first version of HTML developed by W3C was HTML 3.2, which was followed by HTML 4.0 and then HTML 4.01 [47]. After HTML 4.01, they developed XHTML 1.0 that was a new generation and more flexible version of markup language [47]. XHTML stands for extended markup language that was developed by W3C. XHTML is an XML form of HTML. It is a strict version of HTML which focuses on the structure of HTML, e.g. proper closing of each HTML tag is required. The consistent structure of XHTML allows browsers to easily parse web pages.

CSS3 is the extension of old CSS specifications and it is the latest version of CSS, which provides additional features which were not included in previous versions of CSS. Some of these new features include sliding, round corners, animations, text effects, shadows, transformations, rotations and many more. All modern browsers have support for CSS3.

A JavaScript is a scripting language that was originally developed by Netscape. Scripting languages are interpreted whereas structured languages like C++ are compiled. Moreover, scripting language is a bit slower than compiled C++ language, but very handful for writing short programs [28].

JavaScript is the client-side language that runs in a browser and is used to enhance the interactivity of Web pages. JavaScript is usually written in HTML page inside the head tag of HTML or it may also be written in the separate JS (JavaScript) file. However, all the external JS files must be included in HTML file under the head tag. JavaScript can be used to manipulate HTML elements dynamically and access them through DOM.

After loading of DOM, the Web page is ready and shows up. JavaScript can be used to implement event handlers that are fired up on user's interaction with the page. For example, event handlers for Web components like button, links and text input are defined by JavaScript which effectively defines the functionality of the Web page after the user interacts with Web page [30].

JavaScript can also be used to send HTTP requests to the server and update the Web page accordingly with respect to response. Thus, allowing a Web page to update its DOM content dynamically without reloading of the page. This approach is known as AJAX (Asynchronous JavaScript and XML) call. AJAX sends HTTP request to web server and receives JSON (JavaScript Object Notation) or XML response which is then used to update DOM. Ajax call is typically used to access remote services. Earlier, JavaScript was used only on the frontend of the client, i.e. it executed on browsers only to create interactive elements like slideshows and other interactive elements. But then Ajax came into play, which allowed developers to do smart stuff with JavaScript like loading the new content on a Web page without refreshing it [29]. AJAX request runs asynchronously in the background, thus, it is a non-blocking call. Old fashion client and server

interaction consisted of accessing the static Web page from the server through specific URL (Uniform Resource Locator) which always required the page refreshing.

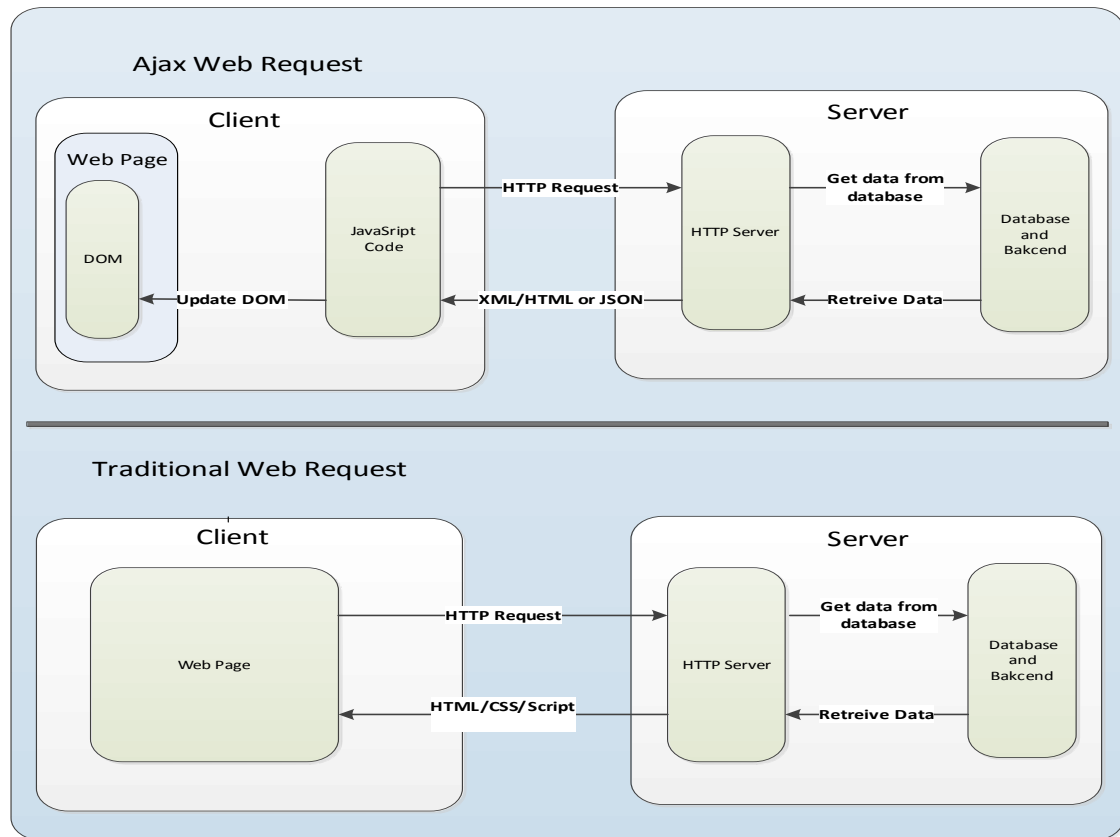


Figure 5. A graphical representation of normal web request and Ajax web request

The client and server communication is mostly done through RESTful calls. REST stands for Representational State Transfer. REST is not a protocol but an architecture style. REST does not impose any restriction of the protocol on client-server communication, but HTTP (Hypertext Transfer Protocol) is the most commonly used protocol because it has been a primary transfer protocol for Web [31]. The client-server communication in REST is stateless which means no client context is stored on the server between requests. One important principle of REST is CRUD operations which are carried out by using HTTP commands to create data, read data, update data and delete data. CRUD is based on basic operations which are done in a data repository. REST uses URIs to access resources from the server, which is quite similar to access website from the browser using URLs.

Most of the browsers have plugins to debug JavaScript that make it easier for developers to find and fix errors. The plugins also allow developers to analyze the code: which portion of the code takes a lot of time and makes a bad user experience. For example, the profiler tools allow developers to see the time taken by your JavaScript code. Simi-

larly, with the console tab in developer tools, users can debug the JavaScript code. In addition, there also exist some tools that can automate the certain task. Grunt is one of them [32], which is quite popular nowadays among developers. Grunt is a task runner that can improve the flow of front-end development work dramatically [32]. Task such as validating JavaScript code or HTML code, Sass and Coffee script compilation, and validation of JS code using JSHint can easily be automated [32].

3.2.1 Web Development Frameworks and Concepts

The increasing popularity of Web applications and a vast range of mobile devices through which we access Web, has made the life of developers a bit complicated. In past, development was simple because Web pages were mostly accessed from a desktop computer on large screen. But now, the device size is so dynamic which requires the website to be responsive to open in the browser. Similarly, the idea of single page applications gives better user experience where Web app resides on a single page. These Web apps are also called dynamic Web applications.

Single page applications are the Web apps that load a single HTML page and update its content dynamically on interaction with the user. It gives a more fluid and responsive user experience similar to a desktop application. Single page apps are usually implemented by AJAX and HTML5.

Data bining is the process that synchronizes the data between model and view. If a value changes in the model, the view is updated automatically and if the view changes, the model is updated automatically. This is called two-way binding which is used by AngularJS. In the past, when there were no frameworks available to bind views with model, to populate container controls with data and to develop a modular website, the development of dynamic Web applications required too much effort and time but fortunately, now there are lots of Web frameworks available that were designed to support the development of dynamic websites, Web applications, and Web services. Most of the Web frameworks are open source and free to use. These frameworks help developers to write faster, cleaner, structured and reusable codes. Some of the well-known front-end development frameworks and tools are AngularJS, PolymerJS, ReactJS, KnockoutJS, SkeletonJS, EmberJS and many more but we will elaborate only AngularJS and ReactJS in next two paragraphs since they are the most commonly used concepts/frameworks at the present time. Results given below were extracted from Google trends, that shows that the most popular frameworks over the last 5 years have been AngularJS, BackboneJS, and ReactJS. However, the graph of backbone.js is decreasing since the emergence of ReactJS. ReactJS was developed in the year 2013 and is quite new, that is why it is not as popular as angularJS but in future it may become a more prominent framework among developers.

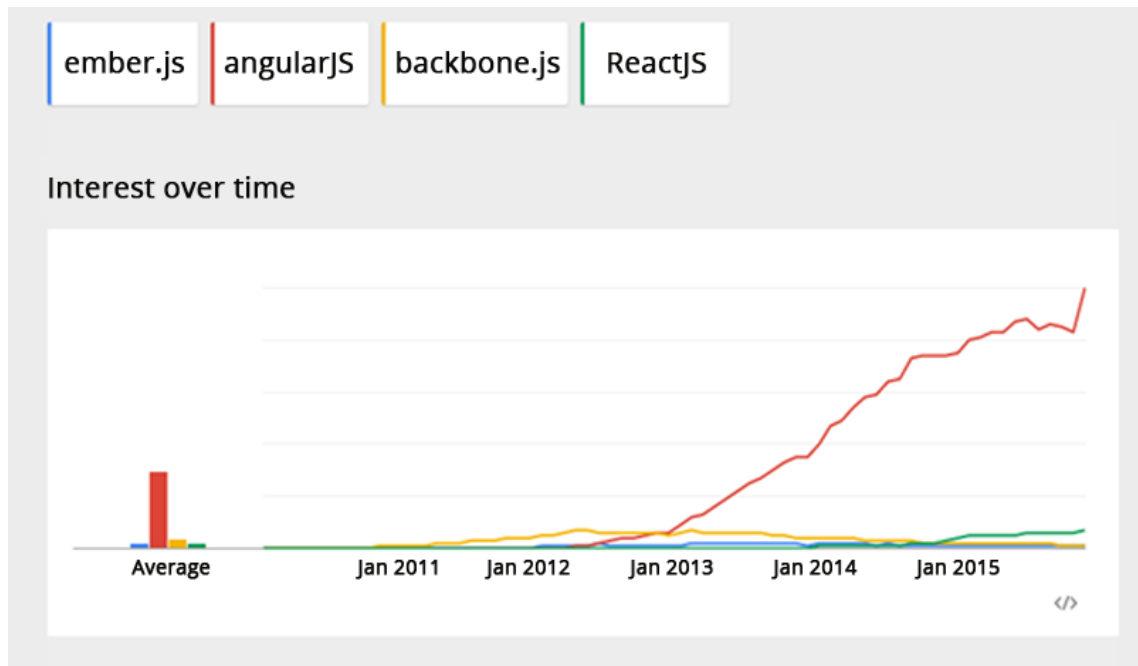


Figure 6. Search Result of front-end Frameworks extracted from Google Trend

AngularJS was developed by google in the year 2009. AngularJS is an open-source framework for Web applications that provides MVC (Model-View-Controller) architecture to the website [33]. This is in contrast to the traditional implementation of MVC architecture like Spring MVC, where MVC design patterns are implemented on the server side, whereas, in AngularJS, View is created in the browser from Model that holds the required data [33]. This helps developers to build rich internet applications. The core features include two-way bindings between Model and View, MVC architecture, filters, directives, templates and dependency injections. AngularJS works best with HTML5 which makes it more dynamic and intensive in capabilities. The data driven approach of AngularJS lets developers concentrate on application data only, i.e. programmers take care of data and data takes care of HTML [34].

ReactJS was developed by Facebook in the year 2013 and is quite new compared to AngularJS. ReactJS is not a framework but it can be thought as a “VIEW” of the application. It is a concept which focuses more on component driven development. It allows developers to break down application in smaller components which may be used somewhere else in the applications. A component can also be built on another component. React keeps the virtual DOM in memory which is used to detect the change in state of the components using observer model rather than dirty data checking (continues checking for data change) [35]. This virtual DOM concept enables ReactJS to render DOM very fast. ReactJS re-renders the whole sub-tree when its parent node is updated by marking it as dirty [35].

3.2.2 HTML4 vs HTML5

For many years, HTML4 has been the standard language set by W3C for the Web development. However, it doesn't fulfill the latest computing requirement of industries. In past, HTML was mainly used for building static pages. Additional features like animations required additional plugins which was an extra effort for the developers and the Web users. Secondly, installations of additional plugins also require memory and CPU that was an overhead. Due to this problem, Apple removed flash support from their devices [37].

To overcome these problems of limited functionality, WHATWAG **Web Hypertext Application Technology** Working Group and W3C sat together and started working on the new generation of HTML, i.e., HTML5.

WHATWG and W3C are the two organizations working on the development of HTML5, before the year 2006 they were working separately but now they are working together and they came up with the latest version of HTML known as HTML5. HTML5 has overcome the problems by having some additional and attracting features.

Here are some of the additional features that are offered by HTML5.

A SIMPLE DOCTYPE: In any HTML version, the first tag must be the `<!doctype>`. HTML4 was based on SGML and that is why it contains the reference to DTD in `<!doctype>` tag. Also, the doctype is not fixed in HTML4 and it can be strict, transactional and frameset.

A doctype is simple and not lengthy in case of HTML5. Since it doesn't depend on SGML (Standard Generalized Markup Language), it doesn't contain the reference to DTD (Document Type Definition). Moreover, unlike HTML4, the doctype is common for HTML5.

Audio and video tag: HTML4 does not have video or audio tags; therefore, embedding multimedia in the browser was not easy in HTML4. Browser required additional software to run and recognize multimedia. But HTML5 has included tags, like `<audio>` and `<video>`, which provide native support to the browser, i.e., browsers don't need additional softwares or plugins.

Vector graphics: with HTML4 browsers require additional plugins like Flash and Microsoft Silverlight for 3d graphics or animations. Vectors graphics has been integrated into HTML5 [36]. For example, to draw 2D elements, we can use canvas or SVG (Scalable Vector Graphics) element in HTML5.

Geo Location: finding the location of the user doesn't require any external library now. HTML5 provides a feature to find your GPS location

Web sockets: We can have a full duplex communication with the server which can easily be done through JavaScript. These socket connections are used for pusher services and notifications. Pusher.js, a JavaScript library for real-time communication with the server, uses this technology. We have also used this library in our prototype.

Web worker API: Before the invention of HTML5, the pages required some time to show up, because of the intensive JavaScript code running. This means that in HTML4, the browser and JavaScript code run in the same thread. But HTML5 runs JavaScript code in a separate thread which does not block user interaction with UI of the web page. So, we can say that HTML5 is a non-blocking technology.

Application Cache: HTML5 provides an application cache which was mainly developed to access the application in offline mode. Traditional browser cache can only cache visited pages and it doesn't guarantee the persistence of the pages, for example, the cache can be cleared by the user, or it can automatically be removed by the browser to make room for new pages. However, in application cache, we can define resources that need to be cached. It works on the pre-fetch phenomenon meaning that it can cache those page or resources which users have not yet visited. HTML5 also has WEBSQL which is used to store relational data.

4. IMPLEMENTATION OF MOBILE BASED MES

This section describes the implementation of the mobile MES system. The implementation was done with native technology on Android platform and with Web technology. In addition to implementation, this chapter also describes the manufacturing process, MES, and mobile technology for MES system.

4.1 Manufacturing Process

Manufacturing is an economic term which involves the process of making goods and services to satisfy customer needs [43]. This process is carried out by making use of various computer aided software and machine tools. These machines are usually called CNC (Computer Numeric Control) machines which are fed with NC (Numeric Control) programs. NC programs are used to control machine tools. Manufacturing involves a series of steps [43] which are listed below.

- Product design and development
- Material Selection
- Process Planning
- Inventory Control
- Quality assurance
- Marketing / Delivery to customer

In product design and development, the needs of the customers and all the requirements that are needed to achieve the end-product are analyzed. This involves the designing and simulation of the parts, product, and assemblies.

Material selection is the step which focuses on selecting the best material for the part. In this step, manufacturers examine cost, performance and availability of different materials required to develop a final product.

Process planning involves identifying all the processes that are needed to be done in factory floor in order to produce a final assembly. These processes include nesting, blanking, welding, forming etc. The sequence and need of all these steps are analyzed in process planning.

Inventory control manages the stock of each item, raw material or product in a factory. It also involves tracking the location of the item in factory storage. The availability of raw materials needed for end-product is examined in this step.

Quality assurance ensures that the final product meets all the requirements and needs of the customer by eliminating and reducing all the possible defects.

In the final step which is marketing / delivery, the end-product is ready to be shipped to the customer and if the product was manufactured without the order from a customer, then it can be sold in the market.

4.2 High Level Architecture

The high level architecture of the system is described using UML diagrams. UML stands for Unified Modeling Language which is used to model the whole system. UML diagrams are used by the software architecture, engineer or designer to analyze and visualize the system that is required to be developed. There are various types of UML diagrams. Some of them are class diagram, sequence diagram, use case diagram, activity diagram etc.

4.2.1 Concept Diagram of Mobile MES

The concept diagram represents the attributes of objects or classes, relationship and code dependencies between them. The concept diagram of MES application is shown in Figure 7.

The Item can be of three types, i.e. subassembly, product, or part, therefore, item has “Is a” relationship with “SubAssembly”, “FinalAssembly”, and “Part” objects. The TypeID attribute determines the type of the item to determine whether it is SubAssembly, FinalAssembly (product) or a Part. Other attributes of Item are ItemID and Name. SubAssembly is composed of one or more parts, so it has a 1-to-many relationship with Part. Similarly, a Final Assembly is composed of one or many parts or sub-assemblies, so it has a One-to-Many relationship with “Part” and “SubAssembly”.

Every item is placed in a store, so it has a many-to-many relationship with the “Storage” object. Many-to-Many relationship means an item can be placed in several stores. For example, an item “FP_006” has 20 quantities, out of which 10 quantities are placed in store A1 and the remaining 10 is placed in store A2. Moreover, an item can be added to store or it can be moved from one store to another. The “Store” object has two attributes “ID” and “Place”, where Place is the name of the store.

Object “Order” has “has a” relationship with Item because every order must consist of an item. However, the order can process several quantities of an item at one time but it can process only one type of item at a time. The “Has a” relation is also known as composition which is represented by a filled diamond shape at one end of the arrow. Every “Order” has a work-step, so there is a one-to-one relationship of “Order” object with “WorkStep”. WorkStep is defined as the type of work that is carried out in a manufacturing process. OrderID, OrderAmount, DueDate, and ERP_Reference are the attributes of the Order where OrderAmount is the quantity of the item which is required to be produced.

The “Order” has also a relationship with “Operator” object because every order is executed by an operator but placing an order is only done by an operator of type “Master Operator”. Operators are categorized as “normal operator” and “master operator” that is why Operator has an association relationship with “normal operator” and “master operator”. All the operators have UserID and password which they can use to log in and after login they can edit/update their profile.

Similarly, “Process” object has “Has a” relationship with “Order” because every process has an order associated with it. The Process has three operations in its object, i.e. StartProcess (a process can be started or resume if it is stopped), StopProcess (a process can be stopped if it is running) and FinishProcess (a process is completed and need to be marked as finished, either failed or succeeded). The attributes of Process are OrderId, StartedOn, OrderId, and StartQuantity. StartedQuantity is the amount of the item we started with, for example, an order has 10 quantity but we started with 8 leaving out 2, so StartedQuantity would be 2. StartedOn is the time on which we started the process. OrderID is the id of the order which is currently being processed.

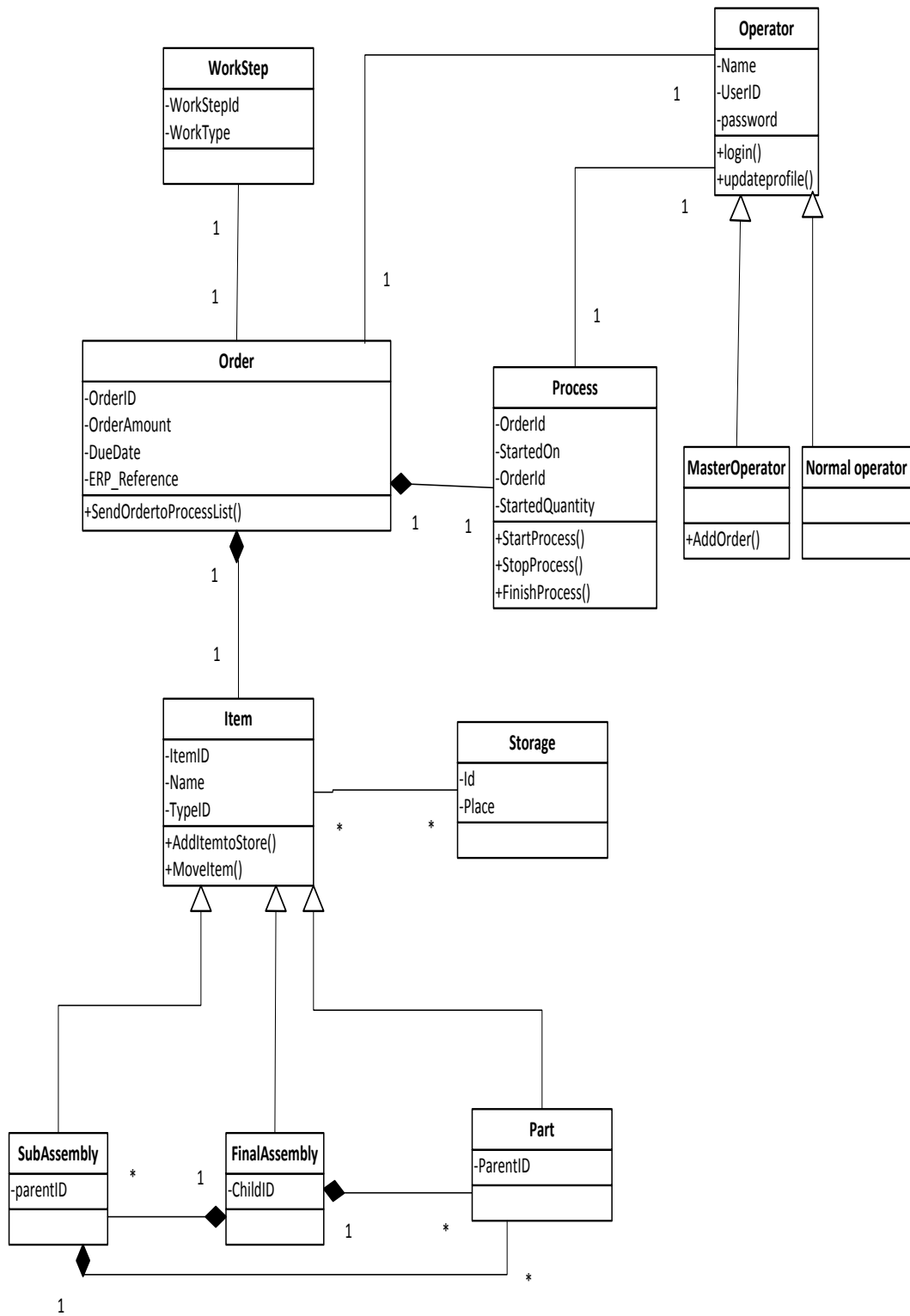


Figure 7. Concept Diagram of Mobile MES

4.2.2 Sequence Diagram of the Processing of Manufacturing Order

A sequence diagram is the interaction diagram that describes the order in which the processes of the system are executed and how the processes operate with one another [44].

The sequence diagram shown in Figure 8 shows the steps to process an order using a MES (all UML diagrams are based on assumption). In addition to the basic manufacturing steps, the diagram also contains some additional features which are implemented in mobile based MES, for example searching the order, moving the items, add or remove items and others.

The manufacturing process starts with an order from a customer. The order from the customer then goes to the ERP (Enterprise Resource Planning) system. The MES system can see all the orders from ERP system. The operator who has the MES either a mobile based or a terminal based can see the orders. The first step of manufacturing is the nesting. The nesting is a process to arrange parts in a pattern that produce minimum raw material waste, where parts are produced from the raw materials such as sheet metal [45]. The nesting is done by CNC machine, so the parts are sent to the machine. After the machine produces parts from nesting process, the operator picks it from there and stores these parts in a store. The operator might want to move a part from one store to the other depending on the needs. Similarly, more stores can be added to the system by the admin if needed. Once the order is in the MES, the operator can search the orders. The operator then selects and processes an order. An order may be an order of sub-assembly or a final assembly. This step is recursive because an assembly might consist of several sub-assemblies and operator keeps processing sub-assembly orders until the final product is achieved. The sub-assembly is a part or combination of different parts but not a final product. The sub-assembly may consist of different work steps like blanking, forming, welding etc. The final assembly may also go through several steps like forming, blanking and others, before a final end-product is achieved. Once the final assembly has gone through all the required processes, it is recorded in a MES and ERP, and end-product is delivered to the customer.

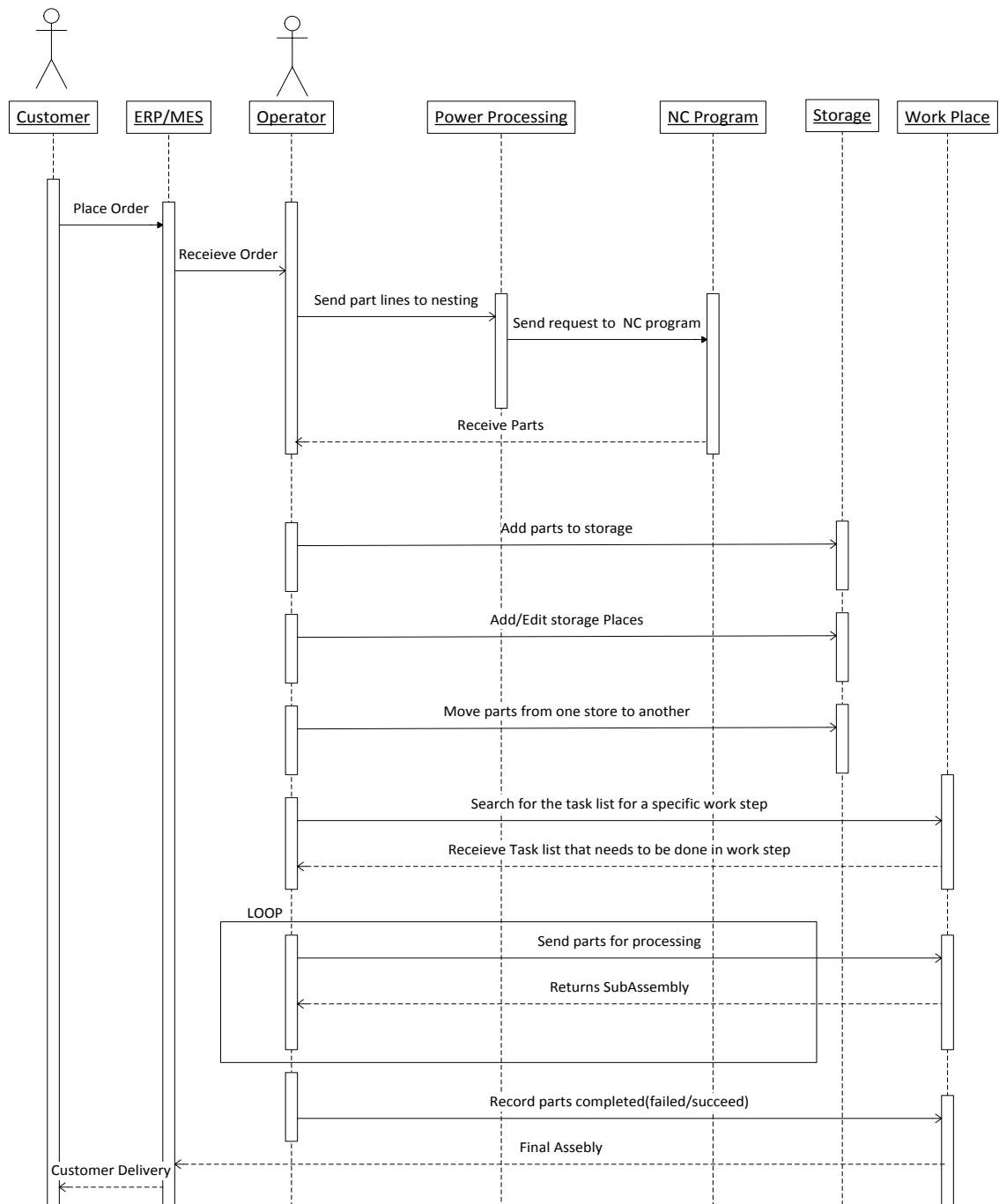


Figure 8. Sequence Diagram of Mobile MES

4.3 Application Backend

The backend was implemented with Java using Spring [79] as a framework and Hibernate [80] as ORM (Object Relational Model). The PostgreSQL 9.4 [81] was used as a database. The database was filled with the dummy data since we didn't get any real data from the company. Apache Tomcat was used as an application server to deploy backend. RESTful Web service was used for communication between client and server.

Spring is a Java application development framework which provides easy and fast development and provides an opportunity for code reusability. Spring works with POJO (Plain old Java Object) rather than EJB (Enterprise JavaBeans) which has some overhead. Spring also provides dependency injection that allows building decoupled applications.

The server application is divided into 4 layers, i.e. Model, Controller, Dao (Data Access Object) and Service. Controller is used to handle HTTP request from REST client. The Controller maps the HTTP request to the corresponding request handler.

The model layer contains all the POJO classes with Hibernate annotations because we are using Hibernate as ORM (object relational model). ORM is the conversion of data between incompatible type systems in object-oriented programming languages [46]. In this server application, it is used to map the Java class objects into database (PSQL) objects and vice versa. Using Hibernate, developers don't need to write stored procedures in the database. Hibernate annotations can be used to map, filter and join data from database.

Then there is a Dao layer which contains Dao and Dao implementation classes. Dao classes define the interface of the operations which are performed on the model objects while DaoImpl classes actually contain the concrete functions of the interface defined in Dao classes.

The Service layer invokes Dao methods. The Service layer lies between Dao layer and Controller, so Controller has access only to those methods which are available in the Service layer. Similar to Dao layer, Server layer has two types of classes, i.e., Service interface and Service implementation classes which just call the method of Dao in its implementation.

The architecture diagram is shown in Figure 9. The client sends the request to the Controller that is handled by request handler. After mapping the request to the request handler, it calls the corresponding Service interface which calls data access layer Dao. Dao

calls the database which returns data in DataTable format which is then mapped to the class object by Hibernate. Dao, after converting DataTable into a data object, sends data to the service and then service sends it to the Controller. The controller then sends the JSON data to the client who made the request.

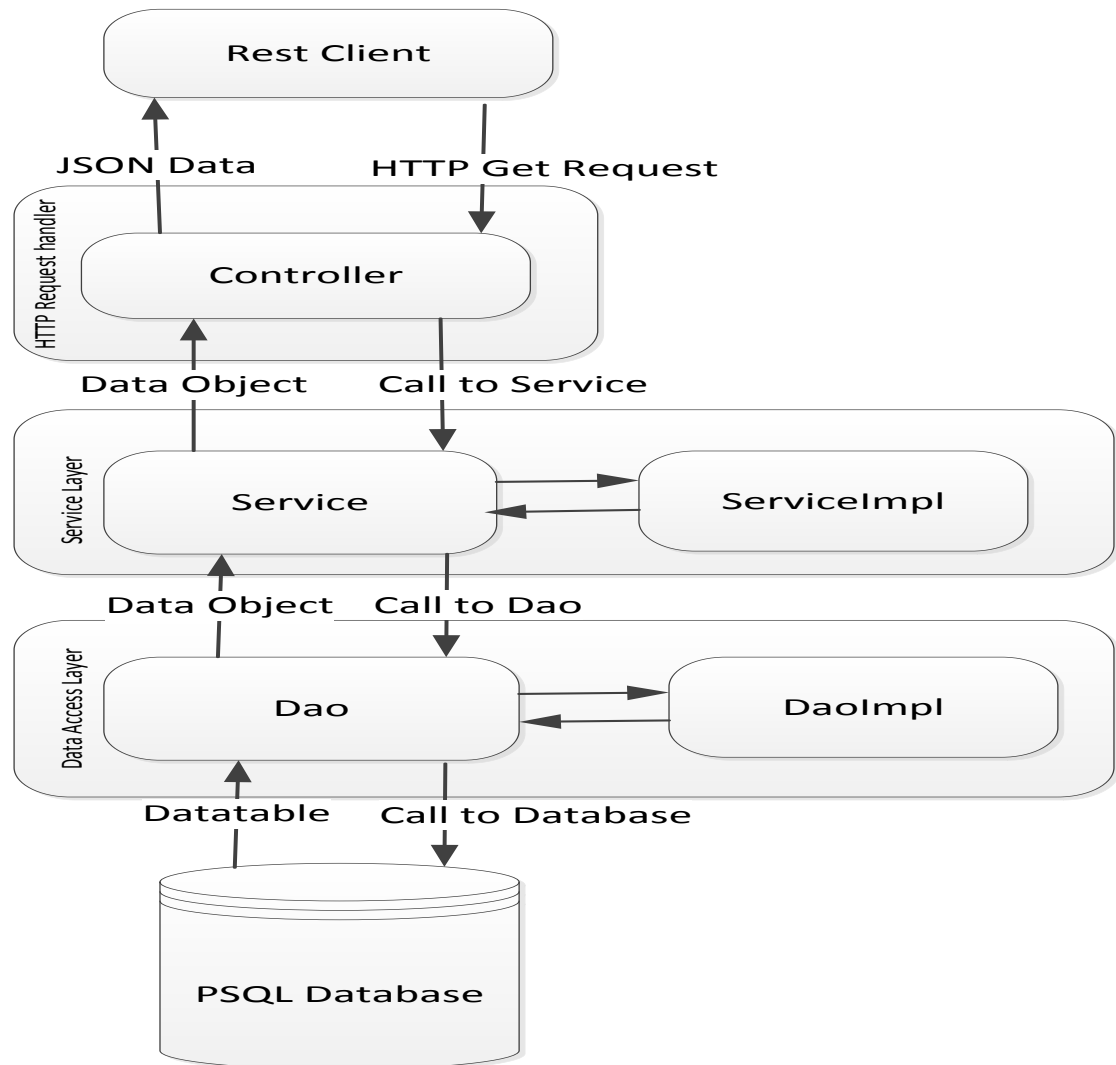


Figure 9. Architecture Diagram of Mobile MES Backend

4.4 Frontend

Two frontend applications were developed, i.e. native and Web. Native is based on Android platform that will work only on Android devices while Web is cross platform which should work on any platform and any browser. Both applications use the same backend and REST API.

4.4.1 Web

The Web application was implemented with HTML5 and AngularJS as development framework. AngularJS is based on MVC (Model-View-Controller) pattern. So, there are Models, Views and Controllers modules in this application. Bootstrap was used for styling and to make our Web app responsive. The service module contains all the REST calls. The controller calls the service methods and binds the returned data with the model. The model then binds up with the HTML controls. AngularJS provides two-way binding features, so we don't need to worry about binding. Besides these, there are external JavaScript libraries for implementation of other features. The table below gives an overview of the frameworks and libraries we used for our web implementations.

Table 2. Libraries/Frameworks used in Web App

Features	Description	Library/Framework
Development	The frontend development framework	AngularJS
Responsive	The layout that can adjust in any screen size	Bootstrap
BarCode	This is used to detect item	QuaggaJS
QRCode	This is used to find location	WebCodeCam
3D Visualization	This is used to view item in 3D	X3Dom
2D Map	2D map of the factory floor	HTML Canvas
Real-time Updates	Real-time data coming from server	Pusher.js

4.4.2 Native

Native application was built for Android platform using Java language and Android studio as IDE (Integrated Development Environment). The MES native application contains the same features as of Web app. The application has one activity and many fragments. We have used fragment navigation drawer, so the application loads the corresponding fragment when user clicks on the side menu. Fragment navigation drawer is the side menu control in Android. The calling of REST service methods is done in fragment as an AsyncTask (Asynchronous Task). The AsyncTask after fetching data from server binds it with the layout. The application also contains the model layer which contains the classes.

Table 3. Libraries/Frameworks used in Native App

Features	Description	Library/Framework
Platform	The operating system support	Android
Language	Language used for development	Java
BarCode/QRCode	Barcode for item detection, QRCode for location	Xing
3D Visualization	This is used to view item in 3D	Min3d
2D Map	2D map of the factory floor	Java Canvas Object
Real-time Updates	Real-time data coming from server	Pusher Java Client

4.5 Features

The list of features implemented in Web and native app are listed in this section.

4.5.1 User Profile

The first section that appears to the user is profile page. The user is asked to fill the profile. Profile has two section, interest and skills. Skills are those tasks; in which operator is good at, while interest is those tasks which operator wants to learn. On the basis of selected interest and skills in the profile, the operator can filter the task list in “Order” section.

The screenshot shows a mobile application interface for a 'Profile' page. At the top, there is a hamburger menu icon and the title 'Profile'. Below the title, there is a section titled 'Skills' which contains a list of tasks, each with a checkbox: Tool Maintenance, Tool change, Forming, Punch/Press, Blanking, Coating, Nesting, Welding, Bending, Assembly, and Electric work. The 'Electric work' checkbox is checked. Below the list are two buttons: 'Check all' and 'Uncheck all'. Below the 'Skills' section is an 'Interests' section, which is currently empty. At the bottom of the page, there is a blue bar with the text 'save'.

Figure 11. Profile Page

4.5.2 Location Using QRCode

The QRCode is used to find the location of places in the factory floor. The operator, who has a task to pick something from the store, can decode the QR code placed at certain location, to find the location of the store. The app, after scanning, shows the distance and direction from the current position to the destination. In Figure 10, left picture is the screenshot taken before scanning the QRCode while right picture is the screenshot taken after QRCode scanning.

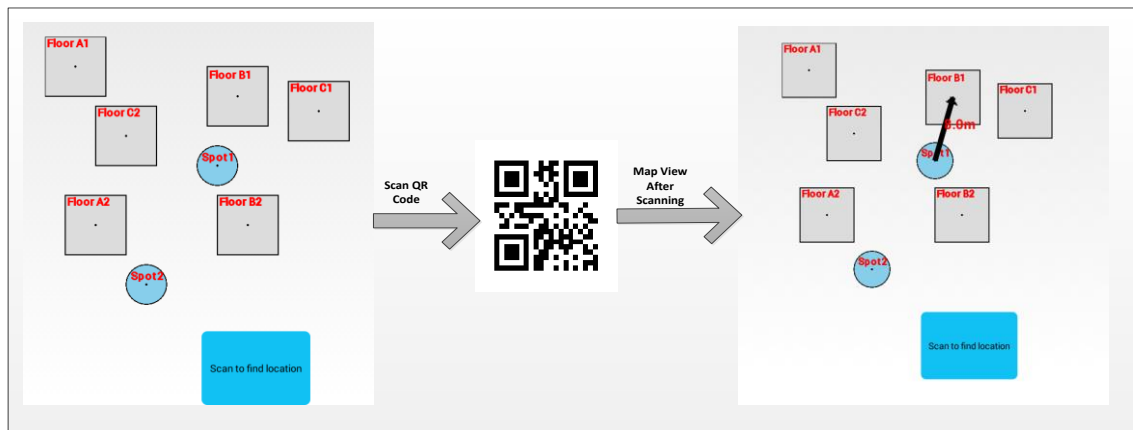


Figure 10. Location Finding using QRCode

4.5.3 Item Detection Using BarCode

The barcodes written on the item can be used to detect and identify the item. The operator can use it when he is not able to identify a part, subassembly or final assembly on a factory floor. One example could be the use of barcode detection while storing an item in the item store. In Figure 11, left picture is the screenshot taken before scanning the BarCode while right picture is the screenshot taken after BarCode scanning. In the left picture, “pp_Part2” is selected in the dropdown but after scanning the item using BarCode, the application automatically selects “Box” in the dropdown. The right picture has “Box” selected in the dropdown.

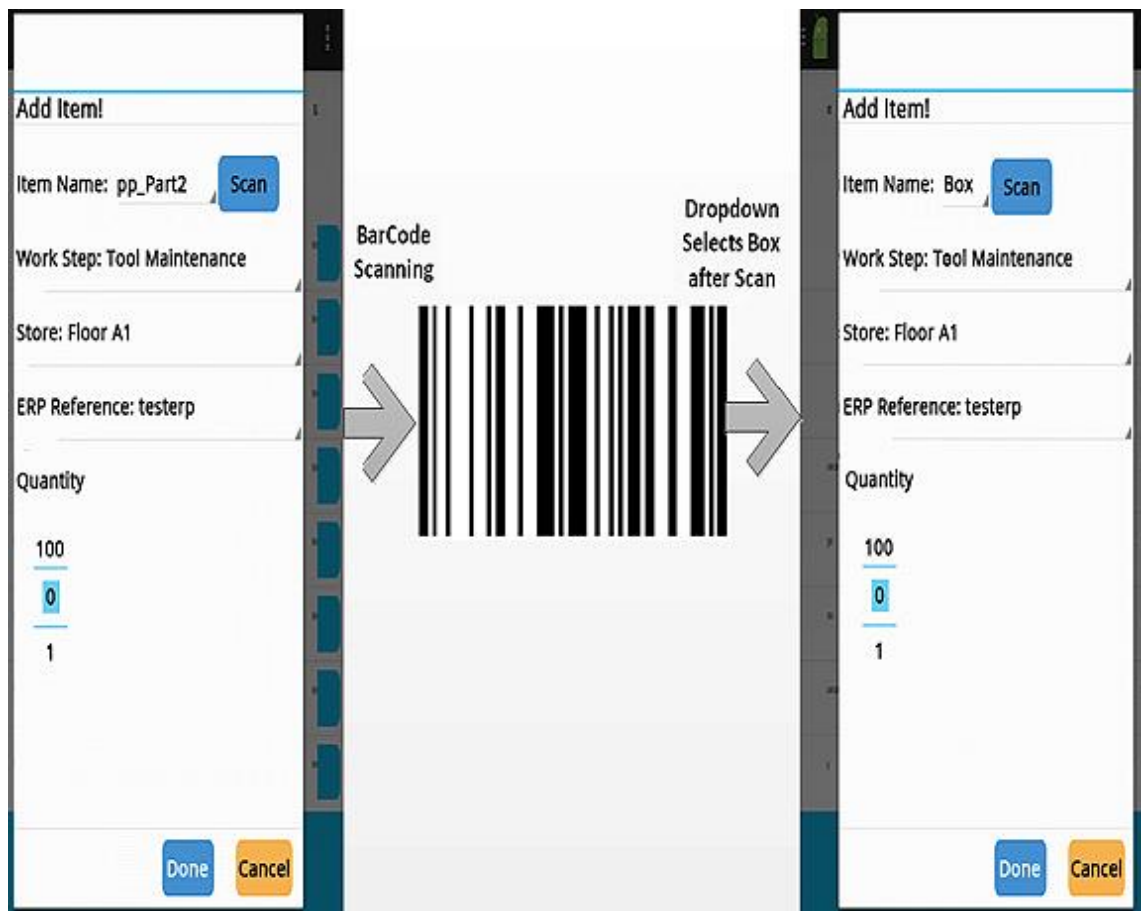


Figure 12. Item detection using barcode

4.5.4 UI Adaption (Non-Functional Property)

The responsiveness of the app is its ability to adjust in the screen of different sizes. Figure 13 shows the view of the page in three different screen sizes. The top left image was taken from the screen of size 320 x 480, the top right from the screen of size 800 x 1280 and the bottom was taken from the screen of size 1920 x 900. Note that, how the screen is changing its layout according to the screen size. The 320 x 480 screen actually transforms the horizontal layout into vertical. The screen shot only displays the running processes section in the top left image, remaining part can be shown up by scrolling.

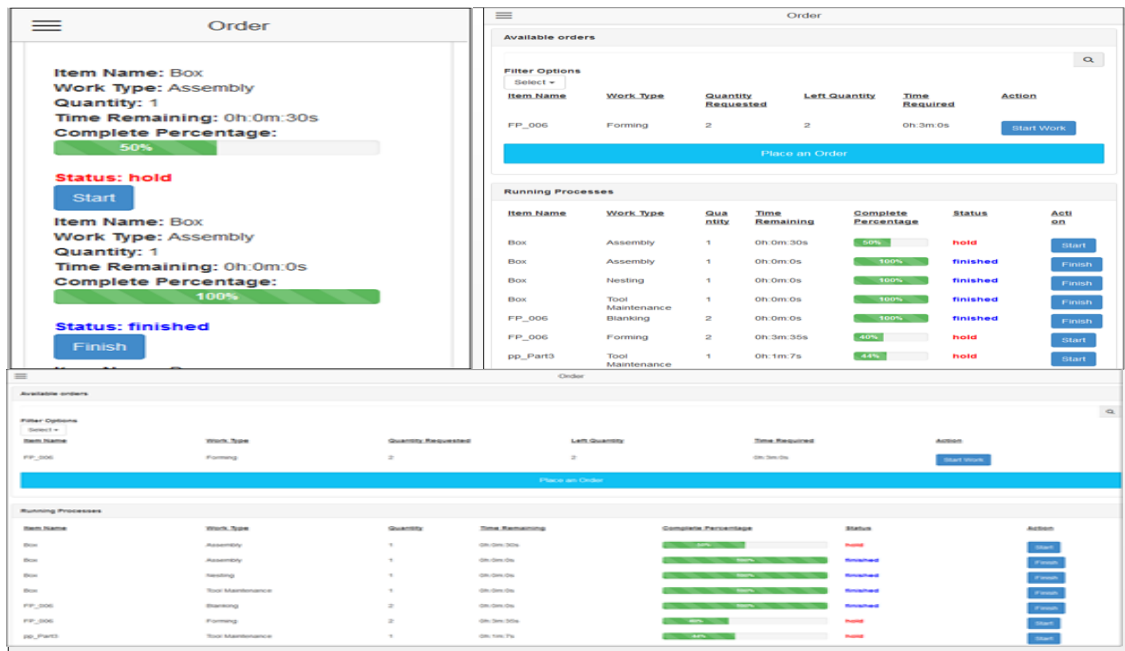


Figure 13. Application view in different screen sizes

4.5.5 Filters

User can filter the task list according to his skills, interests or both. Task list can also be filtered by using keywords in the search bar. In Figure 14, the corresponding screen display and screenshot documents that the unfiltered task list is on the left side, and then after the filtrations, it shows only those tasks which the user is good at. The right screenshot shows the list of tasks filtered by operator’s skills. ElectricWork is the only task out of the available orders that is in accordance with the operator skills.

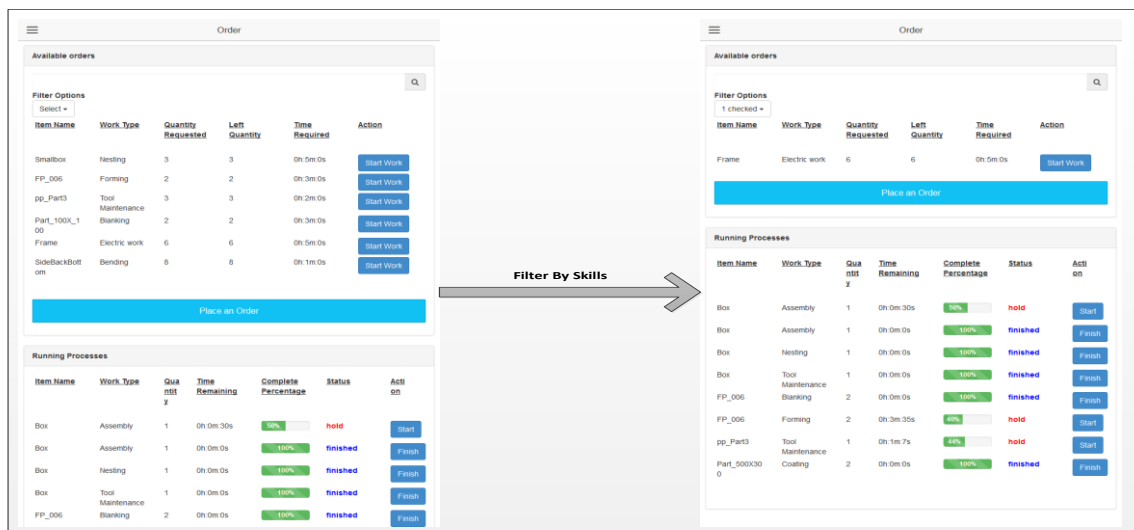


Figure 14. Filter By Skills

4.5.6 3D Visualization

This is used to visualize the items in 3D view. Touch gesture motion is used to rotate the 3D object. Multi-touch can be used to enlarge the image.

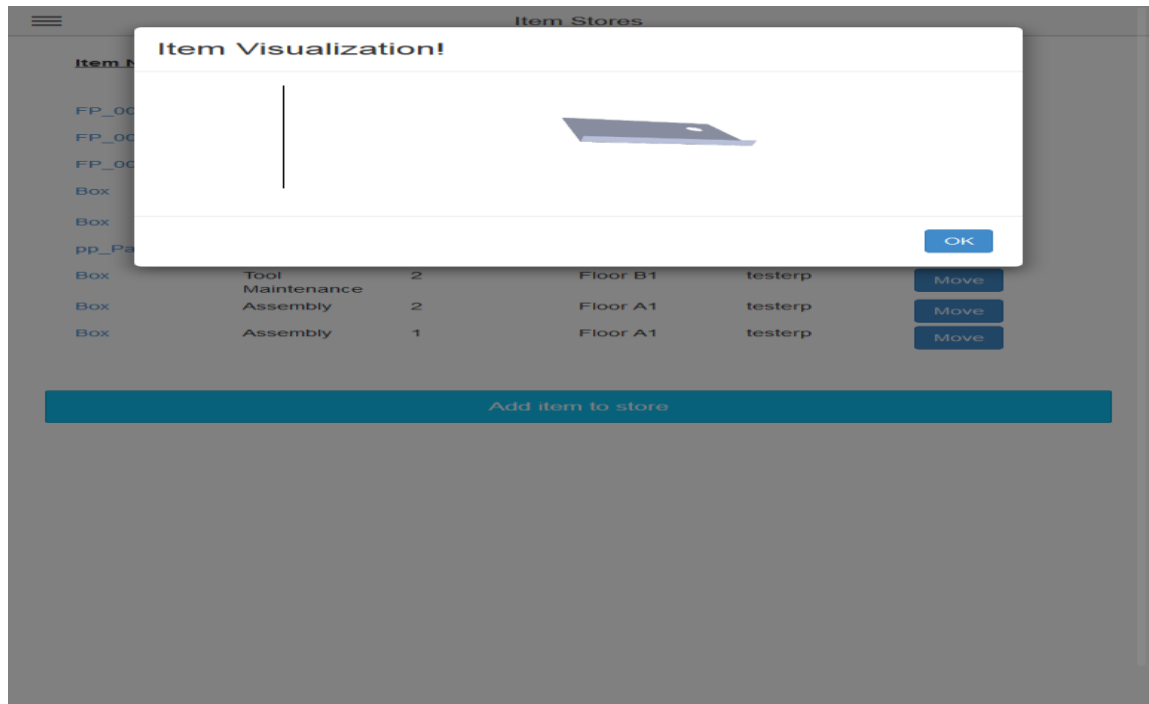


Figure 15. 3D Interactive Visualization

4.5.7 Other Features

In addition to the features described above, we have also implemented other features. Using mobile MES, users can add new manufacturing order in the system. To add a new order, go to the order screen by clicking “Orders” from side menu. Then using the button “Place an Order”, users can add new manufacturing orders in the system. On clicking the “Place an Order” button, a popup will show up where users can enter order details. Then user can start processing the order by selecting any order from the order list and sending it to processing list. User can start processing an order by clicking “Start Work” button in the order list. Once the order is started, it appears on the processing list. The orders in the processing list can be “stopped”, “finished” or “started” by the users. There are two cases of finished order i.e. successful and failure. Once the user clicks the “Finish” button to finish the order, a popup is appeared where he/she marks the process with failure or successful. In case successful process, users don’t have to enter any comments. But in case of a failed process, user marks the process with failure and gives the reason of failure in the comment box. A progress bar is attached with every process which shows the percentage of completion for every process.

Users can also add items to store. To add item in a store, go to the “Item Store” page by clicking “Item Store” from side menu. The item store screen will show the list of all items placed in store. For adding item in store, press the button “Add item to store” which will open a popup. In the popup, user selects the item, work step and store and enters the quantity of item. Item can also be read using barcode as explained in section 4.5.3. The item added to store, starts to appear in the list. Similarly, user can move the item from one store to other. User presses the “move” button which opens up the popup. User selects the target store and enters the quantity of item he wants to move.

4.6 Application Usability Evaluation by Participants

We conducted the usability test of our two applications, native and Web version. Five participants took part in this task. Four of them were mechanical engineers from the background and had good idea about how manufacturing is done in industries while one was a pure IT guy. They gave their feedback based on their understanding and they also gave suggestions on how could we make the better GUI of application. In addition to that, they gave us ideas about the extra features that could be added in MES interface.

4.6.1 Heuristic Evaluation

For one of the users who had completely no idea about this idea of MES on mobile, it was a bit difficult for him to understand the interface. He didn’t like the idea of side menu, i.e., he said everything should be visible on the screen. He didn’t want to open the side menu to navigate to the required screen.

For another user, the search bar was not clear and he had no idea about how to search, i.e. the searching is generic in our app but he thought it was according to the name of the item. He also said the Web app doesn’t ask for confirmation when the user taps the button to change the status of running process. In native, an alert box is appeared asking for the confirmation of the action.

One participant suggested that the items that are finished successfully should be visible somewhere in the system but we just hide the item from the list whenever it gets completed. Also, adding an item in item store contains a list of all items in a factory but according to one of the participants, the dropdown should only show completed items in the list. Some participants didn’t like the idea of manually storing the item in the store; they suggested that it should only be done using bar codes which can detect the item name and the number of items in a lot/box. And yes, we already have this barcode feature in our app which can detect the item name.

4.6.2 Extra Features

Participants also suggested the additional features that might be added in future. One of the participants suggested that the barcode must be used for lot tracking, i.e. it should be able to detect batch number, the processed work step, next work step, quantity and other product details. Operators should be able to see manufacturing instruction on his tablet/mobile, i.e. the work step hierarchy and the next step. It should also have a link to the product documents which can be used to see the complete specification of products. In addition to entering the reason for the failure of failed product, the operator should also be able to upload the picture of failed product.

Warning notification may appear in case of slow machine processing or when there is a risk of delay in the manufacturing process. Notifications should also appear if a process is failed for some reason. There must be some option for higher authorities to see the status of all operators i.e. who is doing what. However, this might increase the pressure on workers as they will always have in their mind that they are being watched by someone. The indoor navigation might also be helpful to locate places on the factory floor. This might be helpful for those factories which have large factory floors.

5. COMPARISON

In the last chapter, we developed two variants of MES client application: Web and native. In this chapter, we will evaluate both of these alternative implementation based on our research criteria. Our research is based on following methods.

- Experimental Testing
- Users' feedback
- Online survey
- Literature review

We tried to evaluate four important aspects in our research. These aspects include user experience, development perspective, limitations, and performance.

5.1 Performance

In order to measure the computing speed of JavaScript and Java (Android development language for this thesis), we ran a very simple benchmark program. The program simply generates a sequence of n random numbers, sorts them with bubble sort sorting algorithm and then displays the total time spent in executing the code. The user gives the number of random numbers that he wants to sort in a textbox. When he clicks the button, the application displays the alert box which shows the amount of time (in millisec-

```

private class MyRandom
2 {
    double seed=0;
4   public MyRandom(double seed)
    {
6       this.seed=seed;
    }
8   private Double next() {
        this.seed *= 1103515245;
10    this.seed += 12345;
        this.seed /= 65536;
12    this.seed %= 20;
        return (long)Math.floor(this.seed);
    }
}

```

Program 1. Program used for random number generation.

onds) spent to execute the code. To ensure that the input data was same for both technologies, we wrote our own random number generator function. The code for random number generator is given in program 1.

The Web application was written with JavaScript and native application with Java. Both the codes have same time complexity, i.e. $O(n^2)$ which is the average time complexity of bubble sort algorithm. The program written in Java can be found in Program 2. Variable “n” is the number of random numbers that user wants to generate and sort. The same code was written for Web with JS. The seed value that we used in our custom random number function is 27. The Javascript version for this performance test can be

```

1   n = Integer.parseInt(txt.getText().toString());
2   long startTime = System.currentTimeMillis();
3   MyRandom rand=new MyRandom(27);
4   for (int i = 0; i <= n - 1; i++) {
5       arr.add(rand.next());
6   }
7   for(int i=0;i<(n-1);i++){
8       for(int j=i+1;j<n;j++)
9           {
10          if(arr.get(j) < arr.get(i))
11              {
12                  temp  = arr.get(i);
13                  arr.set(i, arr.get(j));
14                  arr.set(j,temp);
15              }
16          }
17      }
18      long estimatedTime = System.currentTimeMillis() - startTime;

```

Program 2. *Program used for the calculation of computation speed.*

found in Appendix D in Program 4.

The experiment was performed on 2 mobile devices with different specifications. The specifications of these devices are given in Table 4. Each experiment was performed 5 times for every mobile and an average result was taken. We generated 10,000 random numbers and performed $(10,000^2)$ comparisons on an average. Chrome47 and Mozilla 43 were used in all mobiles/tablets for computation speed comparison testing.

Table 4. *Specification of devices used in Experiment*

Device	Model Number	CPU	Android Version	Chipset	Internal Memory
Samsung Galaxy Express 2	SM-G3815	1.7 GHz Krait	4.4.2 (Jelly Bean)	Qualcomm Snapdragon S4	1.5 GB RAM
Samsung Galaxy Tab 4 10.1	SM-T530	Quad-core 1.2 GHz	5.0.2 (Lollipop)	Qualcomm Snapdragon 400	1.5 GB RAM

Table 5. *Comparison of Computation time*

Device	Avg. time (MS) for Mozilla Firefox	Avg. time (MS) for Google Chrome	Avg. time (MS) for Native Android App
Samsung Galaxy Express 2	9.8	7.5	6.5
Samsung Galaxy Tab 4 10.1	28	20	25

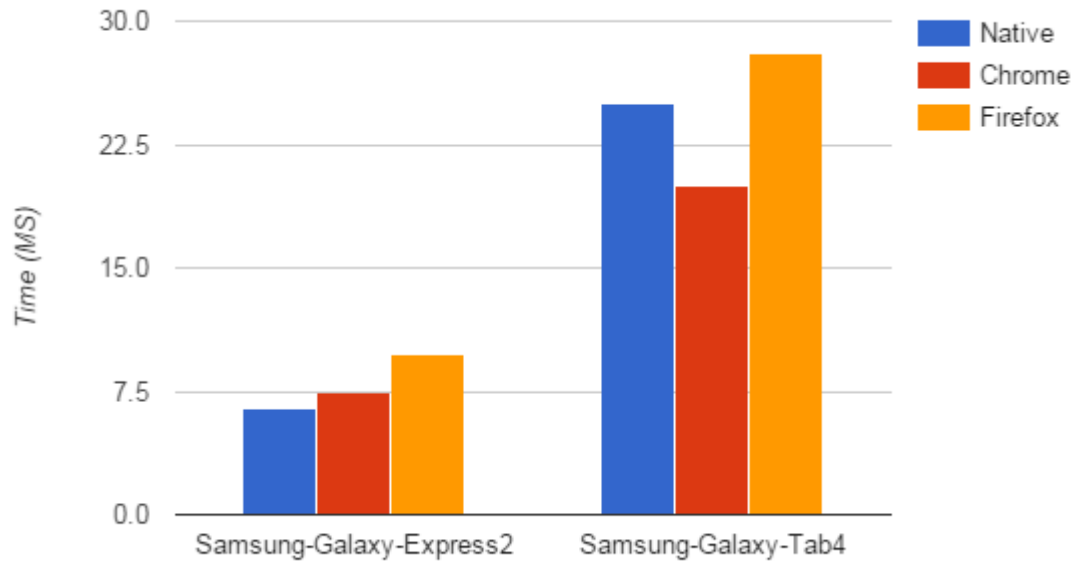


Figure 16. A bar graph representing the execution time of native and web application taken by each device

The bar graph shows the execution time for each platform in different devices. The result is quite expected for the device Samsung Express 2, as it shows that the native Java code runs fastest. But, for Samsung Galaxy Tab 4 Chrome is faster than native. This is due to the fact that android Dalvik and JavaScript virtual machines have different implementations which might show different result for different benchmark test. However, Java seems to be faster for most of the benchmark tests [73].

The execution times for Android native, Chrome and Firefox are quite close for our algorithm. The difference seems to be very small but this is a very simple program. We also checked another algorithm [65], Sieve of Eratosthenes, to find out prime numbers where the difference was minor just like our algorithm. This small difference can be bigger if more complex algorithm is used. For example in binary tree test [63], the difference is too big because it constantly allocates and deallocates the trees. The binary tree algorithm executed in 48.92 in JavaScript v8, but in Java, it executed much faster in 5.75 seconds. The comparison is mainly between JavaScript and native code. The small difference is also due to the fact that most of the modern Web browsers implement JIT for JavaScript which makes JavaScript code faster. Chrome is using v8 JavaScript engine while Mozilla is using SpiderMonkey JavaScript engine.

5.2 Frames per Second

The browser has to do a lot of things in order to render a page on the screen. These tasks include download HTML and CSS from the server, checking out the style of HTML

elements defined in CSS or inline styling, rechecking the styles if modified by JavaScript code, making the subset of layers of Web page and then finally sending it to GPU to composite these layers [50].

But definitely animations or scrolling doesn't need all these steps. However, animations and scrolling go through most of the steps defined above [71]. For example, JavaScript code, responsible for visual changes such as animation function or adding or removing the element to DOM, is executed. Then it finds out which CSS rules apply to which element and final style of each element is calculated. After that, the browser finds out how much space and what position would every element take on the screen. Then the actual painting process of filling out the pixels is carried out. Finally, the compositing process takes place in which the browser renders the layers on screen in correct order. Normally, the elements of the page are drawn on multiple layers and during rendering the browser put them on screen in the right order so that the page renders correctly. If we want to achieve 60 FPS, then all of the tasks defined above need to be done in 16ms which is not possible to achieve at the present time. The performance is worst if the page has lots of long running JavaScript code and at the same time image resizing and decoding takes more time. If a device takes more than 16 ms then users will miss some of the frames and will see a janky display.

Scrolling the listview at 60 FPS is difficult to achieve because the DOM structure of Web doesn't allow Webs to achieve 60 FPS [52]. This is due to the fact that DOM is a retained mode API which keeps the hierarchy of objects drawn on it [72]. The advantage of retained mode is that we can draw and reconstruct complex scenes [72]. However, performance often becomes an issue since extra memory is needed to maintain the scenes and updating the scene could be slow [72]. This is in opposition to the canvas which is an immediate mode API which means that it doesn't retain the information of object drawn on drawing surface [72]. So, we can make use of the hardware accelerated canvas to improve the performance of animation [72]. For example, hardware accelerated CSS is one of them [60]. But it should be used smartly and can be used for 2D animations but not for the whole Web content. Utilizing the GPU unnecessarily can cause execution issues because of large memory usage. It can also influence the battery life of mobile.

During scrolling, the browser paints some of the pixels in layers [51]. By grouping elements into layers, we need to update only that specific layer which undergoes change. So, if you are damaging most of the layers while scrolling, e.g. in parallax website when you have things moving as you scroll, the page needs lots of painting which causes the pauses in the display. So, in general, less painting is good for better scrolling. The performance becomes even worse due to unnecessary events during scrolling like CSS

change during hovering which requires extra painting. Also, the execution of long and inefficient JavaScript code on hovering brings performance issues.

In our MES application, we have a long scrolling list of production orders and running processes, we used this list to find out the FPS in our Web application and Android native app.

The FPS of the native application was more consistent and it has higher values than Web app. The FPS of the mobile native app was taken by turning on the “Profile GPU Rendering”. The Profile GPU rendering, shown in Figure 17, was displayed using bar graphs. The horizontal axis shows the elapsed time while the vertical axis represents the time (milliseconds) taken by each frame to render. Each vertical bar represents one rendering frame which means the higher the vertical bar the longer it took to render. Moreover, each vertical bar consists of four colors, i.e. blue, purple, red and orange [70]. The green horizontal line represents 16 ms. To achieve 60 FPS, each of the vertical lines must be below this green line. If a vertical bar crosses the green line then we will see a pause in animations or a janky screen. The blue segment of the bar represents the time used to create and update view’s display list. The purple segment represents time elapsed in transferring resources to the render thread. The red line represents the time taken by 2D renderer to send commands to OpenGL to draw and redraw display list. The orange line is the time for which the CPU waits for GPU to finish its work. The max FPS value is 60 because most of the display devices have a refresh rate of 60 FPS. The screen captures of our Android app were taken at 4 different instances as shown in Figure 16. It is clear the native application is achieving 60 FPS in the top left and top right image. However, in the bottom left and bottom right, there are few points where the vertical bar crosses the green horizontal lines which means that there were some pauses when we were scrolling the list. However, these lines are very few as compared to the lines which are below green line, so the overall effect is that we get a smooth scrolling in native application.



Figure 17. A bar graph representing GPU rendering of MES (native application) at four different scrolling instances

The FPS for Web app was calculated in Chrome 46 by enabling USB debugging in developer tools. Here USB debugging means remote debugging which let us debug the content of Web app, running on Chrome browser of mobile, on a development machine or a desktop PC. Debugging the Web app on mobile using developer tool is not possible on mobile. Therefore, we used USB debugging to calculate Web app FPS. Remote debugging is done by following the steps given below.

- Connect the mobile device with PC
- Go to the developer option in android setting and enable “USB debugging”
- Open the link “chrome://inspect/#devices” on chrome browser of PC.
- Check the option “Discover USB devices” to find out the connected device with the PC

Similar to the mobile app, screenshots of Web app with FPS meter were taken 4 times as shown in Figure 18. The white line in the graph shows the maximum FPS which is 60 FPS. The graph for Web app is quite low and has more inconsistencies as compared to native app. Some of the lines in the second graph show few frames achieving 60 FPS but the overall performance is low. The worst performance can be seen in the 3rd graph where most of the instances are touching 0 borderline. The FPS meter shows the FPS measured value is 14.5 which is quite low.

If we take a look at the graph of native app, then it is quite clear that the native has more consistent lines and it achieves 60 FPS most of the time. First three frames are quite consistent while the fourth graph has some janky frames where it crosses over the limit of 16 ms for some frames. However, overall the rendering frequency of native app is higher and consistent than Web.

Hence we conclude that the native has better rendering performance, since it is fast and consistent. We proved this by measuring the FPS of native and Web at four instances. The bar graph showed that native Android app had more consistent and higher FPS.

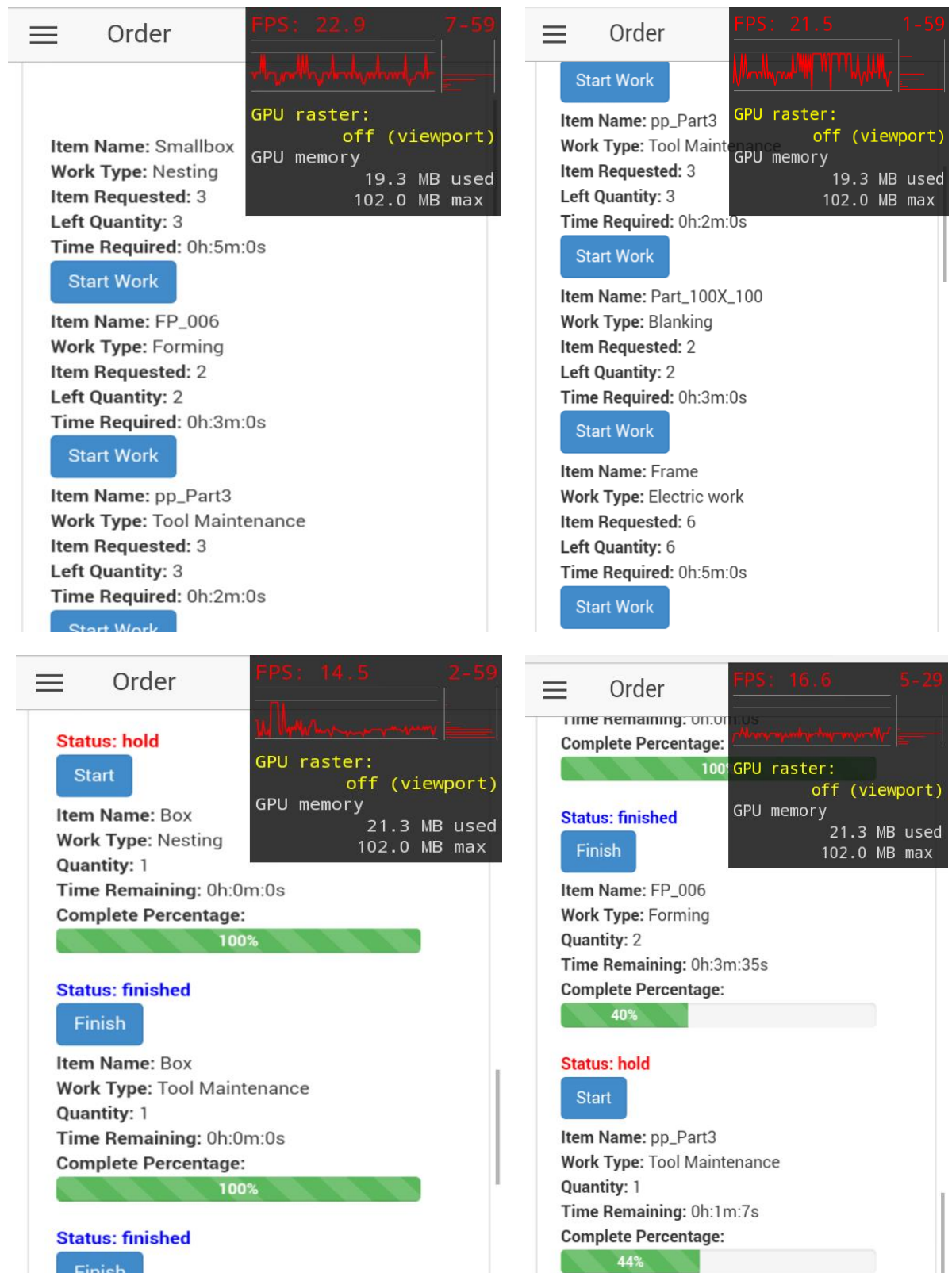


Figure 18. A graph representing GPU rendering of MES (web application) at four different instances

5.3 Tap Delay

Mobile devices have capacitive touch sensor screen which is able to respond to user's interaction with fingers. One of the main reasons for the popularity of mobile devices is the gesture based interaction. For mobile, gestures can be divided into touch mechanics and touch activities [53]. Touch mechanics are the specific gestures with your fingers on the screen while touch activities are the result of the specific gesture [53]. Some of the examples of gesture mechanics are touch, double-touch, swipe, multi-touch, pinch zoom-in (drag two fingers away), pinch zoom-out (drag two fingers towards each other), long press and many more.

One of the reasons of slow Web performance is the 300 ms delay between tapping on the screen and releasing tap. Mobile browsers have the double tap gestures which they use to zoom anything on the screen. Single tap does not fire before 300 ms because the browsers wait for the second tap till 300 ms. So the touch event works in this flow [54].

- Touch Start
- Touched
- Wait until 300 ms
- Triggering the click event

This was the huge problem in terms of user experience. The users don't like to wait to get the response and especially in the games where user experience matters the most. So, Chrome and Firefox for Android came up with the solution to remove this at the cost of losing zooming feature [54].

However, zooming feature is important to enlarge the content to take a closer look. For example, the buttons or links are placed together and user needs to click a specific link, without zooming it is most probably that he might click a wrong link because of a small area. Sometimes users also want to zoom in text or images. Google Chrome team came up with a better idea, i.e. setting the content width to the width of the device. Doing this, the viewport of the browser is set to the same size as device [54]. This is the way, Chrome implemented zooming features rather than relying on double tap which is the cause of 300 ms delay in browsers. Users can zoom the content of the sites using pinch-zooming gesture. Chrome announced that Chrome 32 and later versions will have double tap disabled while zooming would still be available [54].

```
<meta name="viewport" content="width=device-width">
```

The above solution works with Firefox and Chrome but IE handles this issue using touch-action CSS property. This is achieved by applying CSS to all elements that are clickable, e.g. buttons. The touch-action property is set to manipulation or none.

```
<style> button { touch-action: manipulation; }</style>
```

iOS Safari has not been able to remove this click delay completely. However, Patrick H. Lauke [55] found some interesting facts about iOS Safari. He found out that, the delay doesn't exist anymore for slow taps in iOS 8 Safari but for faster taps it still exists. The speed of tapping is determined by how long you keep your finger on mobile screens.

In order to verify above claim, we made a simple Web page which contains a button. When the user taps this button on his mobile he gets the delay time (in milliseconds) between the touch-end event and triggering of actual click event. The code is given in Program 3. The JavaScript code to find time delay can be found under the script tag.

The web page was opened in Firefox, Mozilla and Chrome and button was tapped ten

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta name="viewport" content="width=device-width;">
5 <title>Delay Checking</title>
6 </head>
7 <body>
8 <button id='btn'>Tap to findout delay</button>
9 <script>
10     var btn = document.getElementById( "btn" );
11     var delay=0;
12     btn.addEventListener( "touchend", function() {
13
14         delay = new Date().getTime();
15     });
16     btn.addEventListener( "click", function() {
17         alert( new Date().getTime() - delay + ' milliseconds' );
18     });
19 </script>
20 </body>
21 </html>
```

Program 3. *Program to find tap delay of Web applications.*

times on each browser. The result was quite expected, i.e. iOS Safari didn't perform well and took the longest time to execute click event of the button. The performance of Chrome, iOS, and Android native was almost same.

Table 6. A comparison table of tap delay

Brows-er	Time(MS) 1 st Try	Time(MS) 2 nd Try	Time(MS) 3 rd Try	Time(MS) 4 th Try	Time(MS) 5 th Try	Time(M s) Avg.
Chrome	11	8	7	5	3	6.8
Firefox	11	7	12	6	8	8.8
Safari	291	254	236	261	244	257.2
Native Android	6	2	2	2	3	3

The bar graph is drawn based on the average time taken by each browser and Android native application.

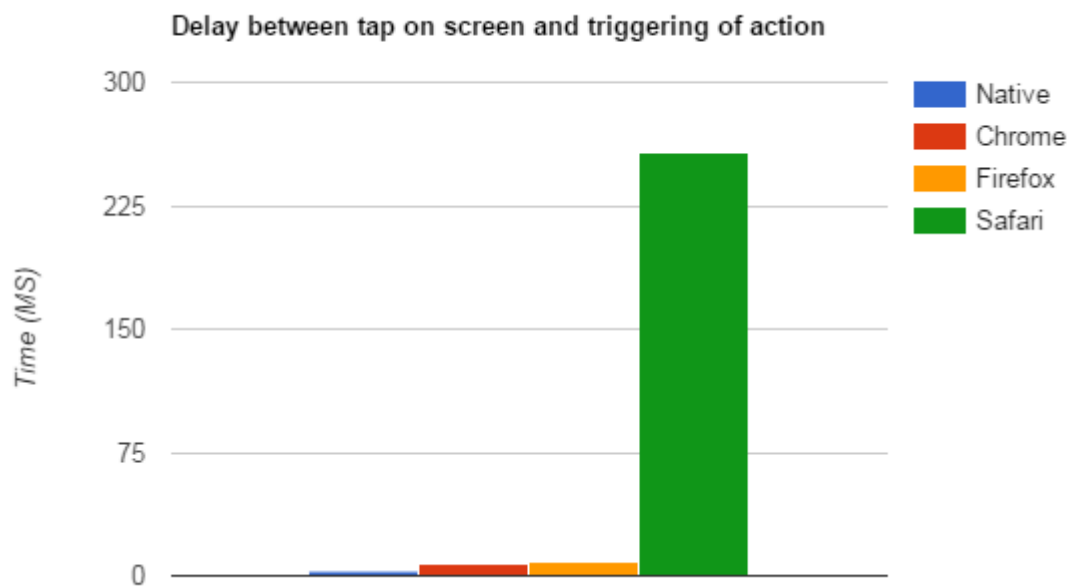


Figure 19. A graph representing Time delay of Web and native MES application

5.4 WebGL is Handy but Slow and a Potential Risk

5.4.1 WebGL

Before HTML5, advance animations and graphics were implemented with plugins to display 2d or 3d animations. These browsers plugins were necessary in past because there was no other way to render 3d animations or video on browsers. One of the main reasons for relying on plugin was slow development of Web browsers [56]. Microsoft launched ie6 in 2001 and at that time Microsoft was the leader and had won the browsers war so they stopped developing browsers anymore [56]. The next version of IE came in the year 2006, i.e. 5 years later after the launching of IE6 [56]. Microsoft released Silverlight to support video and animations in 2007. However, these additional plugins had some serious problems. Some of the problems are security issues, i.e. there was no sandboxing which allows it to access user's account and operating system, cross-platform issues because of different plugins by different vendors that will work on their specific platform [56]. Netflix was based on Silverlight so it relied on the external plugin. However, they soon switched to HTML5 and ditched Silverlight for Google chrome [57]. Soon after the transition to HTML5 for chrome, Netflix did it for Safari. However, for Firefox, it took some time but luckily in December 2015 Netflix announced that their HTML5 player is now working in Firefox.

After the rise of HTML5, the Web doesn't look same any longer. Html5 provides the HTML canvas element which can be used to draw 2d elements using canvas 2d context or 3d object using WebGL. WebGL is a JavaScript API to render 3D graphics without using additional plugins. It is based on the subset of OpenGL which is called OpenGL ES. WebGL is supported by almost all modern browsers including IE 11 and its later versions. The WebGL makes use of HTML5 canvas to draw graphics. At the present time, we have too many options available for graphics display. We can use CSS, canvas, SVG and other external plugins like Silverlight and Flash. But WebGL has some advantages over other APIs [57].

When there was no HTML5, developers were forced to use div with filled colors to draw shapes. Moreover, they had to implement their own functions in JavaScript to animate them, e.g. rotating, moving, scaling etc. But then HTML5 canvas made implementation of 2d graphics a lot easier. The canvas is a DOM element and is supported by almost all modern browsers. HTML canvas is highly interactive which can respond to user's interaction with it, for example, it can respond to click or touch events. Moreover, it can be animated and can be used for any kind of drawing such as shapes, images, and text. However, all the shapes that are drawn on canvas don't form DOM tree which compels developers to redraw the whole canvas whenever he wants to modify it.

The WebGL can perform some complex 3d scenes which are not easily doable with other APIs. WebGL can fully utilize hardware acceleration. Others can also but WebGL was designed to keep performance in mind. WebGL is the best choice for 3d animations but for simple 2d games Canvas 2D might be a better choice because it provides more 2D focused libraries [57].

5.4.2 Performance

Since WebGL is based OpenGL ES so it holds the semantics of OpenGL ES. Therefore, theoretically there should be no differences but there are some facts about WebGL performance. The OpenGL when used on the desktop computer, then the fast C++ language can use the fast GPU quite efficiently. However, OpenGL ES when used with mobile, then the fast programming language usually accesses a slow GPU (mobile GPU is slow) [58].

WebGL doesn't have the capacity to reach a wide range of users because it is not supported in older versions of browsers. For example, it is only supported in IE 11 and not in other previous IE versions. Even Opera and older versions of Safari (pre version 8.0) have disabled this feature by default. WebGL is controlled by JavaScript engine which is not as fast as native compiled code which makes it slower.

The bad performance of 3d in WebGL application has two reasons, i.e. one is when the process becomes CPU limited and GPU is just waiting for work to do and the other is GPU limited when GPU does all but CPU waits for the work to do [59]. It's not easy for WebGL to become GPU limited because JavaScript ordinarily isn't capable of drawing cells fast enough to overpower the GPU [59]. However, the complex scripting in your Web app can cause your application to become CPU limited which is one of the reasons for the slowness of WebGL apps [59]. One of the great features of WebGL is that it can be fit together with other Web elements to give better overall visual effects which mean WebGL has to synchronize itself with other Web elements which certainly results in low frame rates and it can easily be seen in large canvases [59].

The performance of WebGL also depends on the way we use the API. If we have a large number of draw calls then the rendering would be slow. Similarly, the unnecessary and redundant draw calls make it slow too. The performance also depends on how efficiently we use the two shaders, i.e. vertex and fragment shaders of WebGL.

5.4.3 Security

In addition to slow performance, WebGL APIs have some security vulnerabilities. It can directly access video card to run the code on GPU which is a major risk involved with WebGL. That means any untrusted site can access your video card without your

permission and they can run some malicious code with your GPU. Everybody was happy after the launching of WebGL with the support from Mozilla, Google, Apple and Opera [67]. However, soon after the launching of WebGL, Microsoft stated that WebGL is a big risk that anyone can easily pour virus and WebGL is even open to DoS (Denial of Service) attacks [67]. However WebGL was immature at that time like every new technology is immature in beginning. KHRONOS group, who developed WebGL, identified those security issues and came up with the possible solutions which are described below [68].

In native application if a read pixel call contains pixels that are not in frame buffer then it would return undefined. This is not acceptable in Web since it can return the content of another application. So WebGL returns (0,0,0,0) for the pixels residing outside buffer.

There might a possibility that the vertex shader buffer has some invalid index of buffer which can cause out of range exception. The APIs for trusted native code like OpenGL and OpenGL ES didn't implement any check to find out the invalid indexes of the vertex. Since it doesn't have any implementation of "out of range" index checking, so it performs faster. But, on the other hand, this was necessary for websites where an untrusted code makes 3d graphics call. The WebGL specifications check for invalid index at the performance cost. But the new API like OpenGL extension `GL_ARB_robustness` can reduce this impact on performance while guarantying the out of range memory access.

In order to achieve maximum performance, 3D APIs for native do not clear the content of newly allocated resources. But for untrusted code in Web, it is not wise to not clear the resource content, thus, allowing them to view the content of other windows on the screen. This overhead of zeroing the GPU resources is also the cause of slow performance of WebGL.

If a draw call is complex, i.e. the shader buffers are expensive to compute, it takes a long time to execute. Consequently, the system may become unresponsive if the computation call takes too long. Some OS like Windows Vista and later versions reset the graphic driver once it becomes unresponsive due to malicious draw call. The WebGL using `GL_ARB_robustness` notifies the user that the graphics card has been reset, and do you still want to continue with this Web content. However, GPU reset might have a side effect that one application can have on other.

In OpenGL, texture image data e.g. `glTexImage2D` or `glTexSubImage2D`, can come from any source [74]. It can be read from a file, an external server or it can be generated with code. But, WebGL doesn't allow image data from the external server. The server and image must be at same domain if we want to get image data for WebGL texture. In

this way, WebGL prevents other sites from using user's browser as a proxy to access images that are private or behind firewall. WebGL only reads external images if it is CORS (Cross Origin Resource Sharing) allowed.

5.4.4 OpenGL ES vs WebGL

Just like WebGL, Android also provides support for high-performance 2d and 3d animations using an API OpenGL ES. OpenGL ES is a subset of OpenGL specification made specifically for embedded devices. WebGL is based on OpenGL specs so technically functionality wise there should be no difference between them but there are different ways to implement functionalities. Some of the differences are described below [75].

Android provides GLSurfaceView to draw and manipulate objects. GLSurfaceView provides a surface that handles the creation of framebuffer and composites it on the view system of Android. GLSurfaceView.Renderer interface in Android provides the functionalities needed to draw objects on GLSurface. We override the GLSurfaceView.Renderer's functions like onDrawFrame, androidSurfaceCreated, onSurfaceChanged function, which are called on a separate thread.

To draw vertices in Android, we simply define an array of floats and then pass it to VertexAttributePointer function. Whereas, in WebGL we create vertex buffer object, bind it, copy data and then render it. The main thing to be noted here is that, unlike Android, JavaScript provides dynamically typed array to define vertices array, this gives flexibility from coding perspective at performance cost. So, better is to use Float32Array right from the starting which helps to avoid conversion overhead before loading it to WebGL because WebGL applications use Float32Array [76]. JavaScript types array is very fast as it exists as a fixed memory block whereas a regular array is slow, which uses a hashed array tree data structure [77].

Android uses native texture loaders to load and unload images which are supported by these loaders. However, we can also use external libraries to load images which are not supported by native texture loaders. In WebGL, loading a texture image is as easy as putting an image on a Web page using the image tag. The main difference with OpenGL ES is that instead of creating ID using glGenTextures, we use gl.createTexture() in WebGL which returns WebGLTexture Object. Then DOM image object is applied to this texture which is responsible for loading and unloading of all native browser image formats.

Another difference is to handle asynchronous texture loading in WebGL. For example, we are loading a texture that will be drawn as a button, so we would like to set the size of the widget equal to the texture size. But, since the image is uploaded asynchronously

in JavaScript, we don't know the size of the image beforehand in JavaScript. To handle this, we use callback function which is passed to the loadTexture function.

In order to set the camera in OpenGL ES, we specify viewport size and doing this should be same for all devices. But for WebGL, we specify the size of the canvas to scale the view.

To animate the view, Android apps create another thread to run 3d rendering loop. The separate thread for 3d rendering loop is important to avoid UI thread stalls. But in Web, we use RequestAnimationFrame to call update function from our view at the best available time. This is very important for smooth animations, also it allows browsers to call update view function only when it is needed, e.g. if a user is on another tab then there is no need to call update function.

5.5 Online Survey (the developer's perspective)

We conducted an online survey to find out the developers' opinions about mobile Web and native app development. 18 developers took part and filled the online survey. All of them were professionals and currently working in a software industry. Half of the developers had more than 3 years of working experience while others were less in experience as software developers. 70 % of them mostly work on Android application development, so answers from this survey will be based mostly on Android platform. However, most of them were more skillful in Web app development rather mobile application development.

5.5.1 Cross Platform Compatibility

They think that in native, developers have to take care of cross-platform compatibility, i.e. developing different version of native app for different platforms which would take time while in Web we only have to take care of different browsers and most of the modern Web browsers have the same behaviour so there is not much work to do in Web as compared to native.

5.5.2 Easy of Development, Maintenance and Testing

Most of the developers think that the Web app is easier to develop because we can make a responsive Web app using frontend frameworks and Bootstrap CSS very easily while native apps require more training and learning. Also, HTML5 has made the Web development very easy. However, some believe that development totally depends upon experience, if you have more experience of Web development; you would go for Web otherwise native.

They also think that testing a Web app is very easy but in native every time you want to test, you have to build the code and start emulator or connect a real device which is very time-consuming.

Web apps have common code base across multiple mobile platforms, so it is much easier to maintain. On the other hand, maintaining native app is difficult because users use a different version of the app on different platforms, so maintenance and offering a support to all users is pretty difficult. Some believe that although native has cross platform issues but it can be overcome by make hybrid apps.

5.5.3 Access to Device Features

Although native app takes more time to be developed but it can easily be integrated with device features. Developers will love to work with native app if requirements are to take full advantage of all device features. For Web, you have to rely on API's to access hardware. For native, though, the features are built in, hence it is much easier to use intent on native applications. And they think that Web still has some limitations when it comes to hardware accessibility.

5.5.4 App Stores

Native applications are easier to find and download using app stores. Some believe that app stores are a good source to show experience of the developer, i.e., app stores can tell you who developed this application and what are the other applications developed by this developer / company. This is also useful to check the download of the applications and reviews from users that are quite helpful to make improvements. Also, all the big names have their product on app stores that say a lot about app stores. You can provide your services to as many users as you want, i.e. more download more outreach.

However, many of the developers believe that there are no differences when it comes to deployment and both are essentially equal to deploy. One of the iOS developers said that, in iOS, deployment is not easy and takes some time. Some believe that native apps should get permission first to deploy on the app store. In this way, we can avoid fake apps.

5.5.5 Rating by Developers

In addition to answering the questions, developers also rated Web and native technology based on the given criteria. The set of criteria that we chose to evaluate technologies are given below. Each developer rated the criteria with a score from 1 to 5, with 1 is the lowest (worst) and 5 is the highest (best) score.

- Development Effort
- Hardware access capability
- App stores for monetization
- App stores for distribution
- Cross platform compatibility
- Maintenance
- Deployment
- Troubleshooting

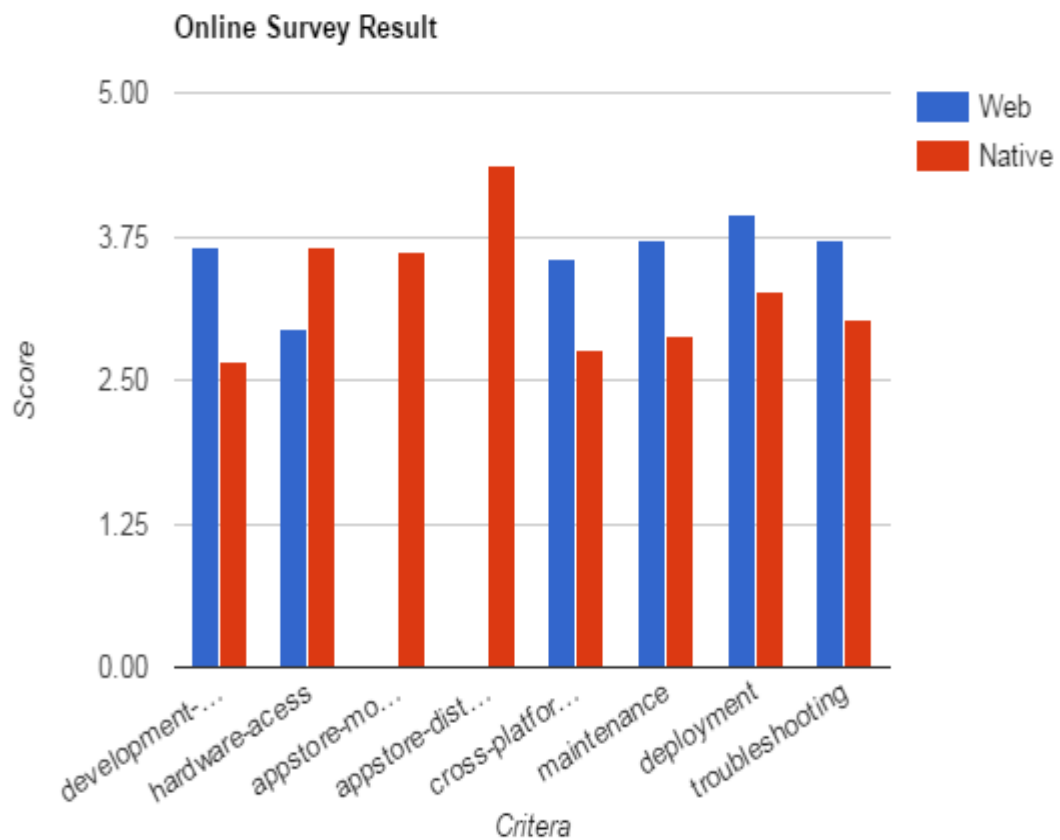


Figure 20. 3D Interactive Visualization

5.6 Interface and Performance Evaluation by users

The two user interfaces were evaluated by users as mentioned in section 4.6. In addition to the application evaluation, they also evaluated the two technologies and gave their opinions about the performance and interface of Web and native app.

Regarding the look and feel of GUI, some voted in favor of native while others voted in favor of Web. Those who favored native said that the controls and colors look more

natural while those who voted Web suggested that it has more advanced and attractive GUI, may be because of the use of advance CSS to make it look better.

They were asked to rotate the 3d interactive image of items. They all were agreed that the 3d animated image performed must faster and smoother in native, i.e. they were able to rotate image very smoothly with their tap gestures in native application but in Web it was slow. They also noticed that the native responded much quickly on swipe gesture.

Then they evaluated the scrolling performance of both apps and as expected, they were all convinced that the Web was slow in updating the screen while native was lightning fast. Some also noticed the delay in button click firing, when they inspected it carefully by tapping the button to stop / run / finish the running process. They said the native app responded immediately. Some also noticed the performance of barcode API and said that barcode scanning is faster in native, but it should depend on the API we used. However, the API (quaggsJS) we used in Web was better in a sense that it was able to localize the barcode but native API (Zxing) was not able to do so.

Finally, at the end, they said if they are asked to choose one, then they will choose native because of the overall performance and responsiveness.

6. ANALYSIS

In this chapter, we summarize the advantages and disadvantages of having a mobile MES app and then we will analyze all the comparison aspects that we took into account in the previous chapter. We will discuss why Web or native technology is better at one aspect but not at other. We will talk about ways to overcome problems if Web or native has some issues with particular comparison factor. In addition to these, we will also find out the importance of every comparison factor for MES.

6.1 The MES Mobile Interface

In the first part of our thesis, we developed MES interface which was developed for Web and Android mobile app. The demo is useful and can be shown to those industries that are still relying on paperwork or desktop based technologies and are reluctant to use modern technologies.

The operators have limited information on a factory floor, but using MES on mobile, they can access documents whenever they want. Moreover, hardware components like camera, GPS, gyroscopes make it more fascinating. Mobile MES can be used to avoid high risk, e.g. there is always a high risk involved while working in a plant like a nuclear plant. In the case of any emergency, the operators/workers position can be tracked using GPS feature in mobile and it can also be used to find out if he is working in a right place. The camera could be used as barcode/QRCode scanner which eliminates the use of separate scanning devices.

The mobile application can benefit the operators to work remotely, however internet connection is mandatory to get updates from the server. In order to use mobile based MES, the company must have a good internet connection which can be accessed through Wi-Fi all over the factory.

The mobile applications are not only useful for doing work remotely but also to monitor the statuses of ongoing processes. Moreover, information will be highly available at fingertips in contrast to traditional desktop based MES where users need to access desktop based terminals in order to get required information about a product or the orders. The indoor positioning is one of the most attractive features that can be associated with mobile based MES. The operators working in a large factory floor can locate the store rooms to place the item and they can also find the meeting rooms.

The MES applications having barcode scanning feature can reduce the cost by eliminating the need of separate barcodes scanning devices. The barcodes and QR Codes can benefit operators in many ways: The barcodes can be used to overcome the possibility of human error. For example, it can be used to extract item information which can be fed to the system directly after the mobile device scans the item. In addition to this, barcodes can be used for the inventory control which is very useful to find out if an item is out of stock or not. Similarly, QR Codes has many advantages; one benefit is to find places in factory floor (if not using location services). For example, an operator who has to go somewhere in a factory to do a task but he is not aware of the place, so he can use his mobile to detect QRCode placed at some point in a factory. On detection of QRCode the system detects the current location of the operator and based on this current location, it gives the direction and distance that he has to travel to reach the required place where he has to do his task.

Sometimes, in manufacturing industries, the operator needs to see the part or assembly which he is going to manufacture. Using desktop based MES, it is difficult to view the dimensions or shape of the part/assembly frequently because of the limited amount of terminals. In mobile MES, they can always view the 3d interactive image of the item which can show the dimensions and shape of the part.

Operators can get intelligent work orders according to their skills and interests. Every operator must fill his profile once he starts the application. The profile is saved on the server and then the server can generate work orders based on their skills and interests. The operators should be able to work more efficiently and with full of interest if the tasks they are doing involve their interests.

On the other hand, it might be too costly for some small companies to invest in the MES system. Moreover, it will be hard to use a smartphone for those users who do not prefer to use smartphones. And, it will be necessary to train every operator about how to use the application because it will be a new technology for most of the companies who still rely on desktop based MES or does not use MES at all. Another risk is the security of the information that can be caused if mobile phones are lost or stolen. All confidential data of the company will be at high risk. We suggest biometric authentication to avoid this kind of possible risks. Another solution is that the application is used only within the premises of factory floor. Using location services, we can track the location from where the application is being used.

6.2 Evaluation by Users

Those participants who took part in the evaluation test were quite satisfied with the overall idea and performance of the applications. Only one of them had some difficulties in understanding the interface, but he was able to use it without any difficulty after some explanations.

All of them were agreed that the native was good in overall performance and they said that the response time is very fast in native application. The observations from them were quite expected and were according to our experiments. Due to low FPS, it is difficult to get smooth scrolling in Web. The participants experienced some stuttering while they were scrolling the list very fast in Web. Evaluation test also confirmed our findings that the web button tapping is slow and other gestures like swiping or sliding are also slow and interactive 3d animations were smoother to rotate in native, i.e. OpenGL ES (for Android) performed better as compared to WebGL.

In addition to performance evaluation, they gave some suggestions. One suggestion was to add lot tracking with barcodes that is quite common in every manufacturing industry. Our demo application is detecting only the item name with the barcode. We will add the lot tracking feature in future.

Another suggestion was to add a link to the document, i.e. the production hierarchy of an item. That could be a good option for operators to see what to do next and to check the other details of production steps. For now, our assumed database has items hierarchies in it. For example, if someone places an order of Mailbox then it inserts all the orders that are needed to manufacture the final product Mailbox. These related orders to manufacture mailbox are filled using that hierarchy data of Mailbox. So, this could be used to show the production hierarchy to the users which will help them to not memorize production steps.

One major flaw noticed by participants was that when they add the processed item in the item store then the dropdown shows the list of all items in the store but actually it should display only those items that were processed but not all the item in the factory. This could be solved by having another dropdown which will contain the list of processed items which have not been added to the store.

There were some other good suggestions like having alarm notification in case of any delay or problem in production, the interface to track tasks of an operator, i.e. who is doing what. These extra features might also be added in our future work.

6.3 Performance (JavaScript vs Java)

The performance of JavaScript was quite close to the Java for the benchmark algorithm we used. The reason for the improved performance of browsers is the JavaScript engine that they use. Chrome uses the v8 engine, Mozilla uses SpiderMonkey and Safari uses Nitro engine. These JavaScript engines are based on JIT (Just in time) compiler. The JIT compilation is the runtime compilation of the code which compiles the code into machine code whenever the code needs to be compiled rather prior to execution. The Chrome engine has performed best as compare to other engines in our experiment. In our experiment, we took the average of execution times in our result; but during testing in some cases, the chrome execution time was exactly equal to the Java in native code.

In old days, the performance of JavaScript was not good enough when JavaScript code used to be interpreted. But things have been changed now and JIT has changed the world of Web by making it faster. Another problem is the performance of mobile phones which makes it slow. The algorithm in our experiment, when used on the desktop computer, executed in 2 seconds. So once the performance of mobile phones gets better, the performance of Web app will consequently be increased.

The performance of the application leaves a heavy impact on the user experience. The slow performance of the website makes the users less likely to visit the websites again. For example, the business website like e-commerce depends heavily on the performance of the website to increase their business and sales. The slow performance results in fewer visits which affect the business. The online shoppers expect website to be loaded within 2 seconds and if it is more than 3 seconds than 40% of the visitors will close the website [64]. According to Gomez's study, the page abandoned rate is increased by 38% if the response time increases from 2 to 10 seconds [64]. Single page apps like our MES Web app can have the same impact on user experience if the loading of the content takes too much time.

This performance matters only in business promotion and sales? The answer is NO. The time consuming and more CPU intensive scheduling and smart work orders are done in the server, so it's not a problem. But, does the mobile Web app have enough performance that operators would choose mobile Web app over a desktop terminal or mobile native app? What about just-in-time production scheduling which can add any item dynamically to the production order list. The operators might not see it in time and might miss it due to the slow response of web page and then the order which was on priority stays unprocessed.

In some type of manufacturing process, the machine automatically inspects the processed item and during that time the machine ceases and operator waits to see the response from the machines. So, in general, we believe that operators would want to see the result of the process as quickly as possible and would like to go for the next production step; but if the application has some delay in rendering the result then it can affect the mood of the operator which can put a negative impact on the overall performance of the operator. The communication between client and server is also important if we want the production process to be speed up. The slow performance can be caused by a slow request from Web client, for example, we have checked that the browsers have some delay especially Safari when triggering the click action of the button. This slow action from mobile Web app gestures will also result in slow communication which will consequently affect the production performance. However, the difference of delay is small but it can have a huge impact on the overall performance.

6.4 Rendering Performance

The major time spent by the users on Web apps is the time spent on interacting with content. Unlike old days, the Web doesn't depend on the frequent request from the servers. The user enters the URL on the address bar, the HTML pages and all required resources associated with it are downloaded from the server. If a loading takes time then it might be due to the slow network or slow response from the server but if the user leaves your site after loading then it is probably due to bad design or bad front-end code that slows down the performance.

The browsers do a series of complex steps in order to render the content on it. It starts it by making DOM from HTML, creating CSS tree from CSS, creating render trees and layers and then finally send it to GPU to paint it. These paintings are needed when there is a need to change in page visuals which can happen as a result of scrolling, rendering, content added or changed as a result of some action from users or content added without any action from the user, i.e. updating view based on data being updated from the server using pusher services. Not just this, but it also needs to execute JavaScript function and event handlers on all HTML elements. Due to all these complex operations achieving 60 FPS is difficult but we can improve the rendering and scrolling performance and make a Web app to perform like native if we take necessary actions.

The first thing that comes is the page rendering and if the page rendering contains too many dirty DOM nodes then you might end up recalculating the whole DOM tree which is very expensive. So we can avoid this by keeping the count of dirty nodes as less as possible. It can be achieved by avoiding unnecessary style changes, applying style changes to the required elements only, e.g. in some cases if most of the child elements need a style changes then we do style changes in the parent container which is not good.

For example, an element is contained in a div and we do styles changes in div rather than that specific element[66].

Layout thrashing is the phenomenon where the layout becomes invalid. The whole Web page can be crashed in appearance if that happens constantly in a loop. If the layout becomes invalid it recalculates the layout. So, once all the intensive calculation work is done by the browser, the bad thing is to invalidate it [66]. The situation becomes worst when you do that in the loop. This occurs as a result of the continuous read-write-read-write cycle [66]. We could avoid this by doing all our reads first and then writes [66].

Then we can use hardware accelerated CSS to make use of GPU and divide the workload between CPU and GPU. The GPU is exceptionally effective at performing essential drawing operations like moving layers around with respect to one another in 2d and 3d space, rotating and scaling layers, and drawing them with changing opacities [50]. There are many ways through which can force CSS to use GPU, e.g. transform, opacity [50]. However, these techniques are needed to be used very carefully, otherwise, it can slow down the overall performance, so, therefore, in this case, high Web development skills are required.

The browser repaints all those nodes which it marked dirty. The dirty DOM nodes will be repainted to the screen. In general, browsers make a smallest possible rectangle region which covers all the dirty nodes [2]. So, if top left corner and bottom right corners of the layout are dirty then the whole page will be repainted. We can overcome this problem by making sure that every element is rendered in different layers [50]. Image resizing is also the factor that could be avoided to make Web faster. So, the best solution is to give the source of the actual size of the image rather than resizing it.

Then we can omit some unnecessary JavaScript events to make the page rendering faster. For example, If all the elements have the hover event on it, then during scrolling, these hover events will be executed before the repainting of the elements. There are many other ways that could be used to improve the performance of Web page.

So, do you remember, what did the developers say in online survey? Yes, they said, Web development is easy. We agree with them but developing Web keeping performance in mind is not easy and it really requires some good skills and performance tools to analyze and improve the performance.

In MES applications, the scrolling and page rendering really matters, since the operator constantly wants to see the process updates, process status and new orders coming from ERP. Doing this, he might also navigate to different screens frequently and if he feels jerks or stuttering then he might be frustrated. Consequently, it will affect his overall performance.

6.5 WebGL, 3D Animations

We know that WebGL has some overhead which is a reason for its slow performance. However, if we use it carefully and efficiently then we can improve its performance. For example, the draw calls of all the models sharing the same shaders can be called once [62]. Avoiding unnecessary and redundant draw calls can also increase performance. On every rendering pass, the fragment shaders execute many times, so move your calculation that could be done in vertex shaders. Once you are done with calculation move it to fragment shader [61]. Don't use "#ifdef GL_ES" in shaders because it will always return true [61]. Textures with small size are faster, so use mipmap to boost performance. There are many more other techniques that can be used to enhance WebGL performance.

The security of WebGL is a big concern but Khronos is working on it and they have been able to figure out some of the problems and in upcoming versions of WebGL, it will be more secured.

The security vulnerability is not safe for industries where security of information is on top priority. We know that these industries have been facing security issues like DoS attacks and Web app attacks. The industrial cyber attack is very common nowadays where the hackers try to access industry data through the network. We have checked in chapter 6, how many ways the information on your graphic card can be accessed by an external entity. But, graphics cards don't necessarily contain the company's sensitive data. So, using WebGL for MES is safe to some extent.

6.6 Tap Delay

The native application was fractionally faster than Chrome and Firefox but this small difference can be neglected. We can conclude that native, Chrome and Firefox do not have any tap delay. But for Safari, we observed a long delay between tap and firing of the click event. The responsive tap is very important in terms of user experience. The operators using a mobile MES would not like to wait for some action to happen. Due to the delay in tap, the operators might end up tapping the button several times which will cause nothing to happen but only enlarging a button, at least in the case of Safari browser. The operator might get annoyed due to this behavior of browser and that will certainly affect the overall performance of operators working in a factory. However, switching between browsers is not an issue. So, even if the operator is using Safari, he can start using Chrome or Firefox for better and responsive tap gesture. So, we believe that this tap delay is not critical in the comparison analysis of the two technologies.

6.7 Device Features

The response from the developers is based on their own perspective. Most of them had worked on Web technologies more than native, so we were expecting that they will speak in the favor of Web. However, the result was completely different and was according to background theory but we got some interesting comments from them. They said that native would be best for those applications which require intensive device features and If we think about mobile MES then what hardware we make use, camera (very important) for scanning, GPU (very important) since every item is 3d, GPS is important for tracking employees, WI-FI very important to get internet connection as this interface is not stand-alone interface but constantly requires updates from server. Gyroscope and accelerometer important for indoor navigation, touch gesture important for interactive images, but sim card not very important. So we believe that the MES application can use enough device features, so it can be a hardware intensive application.

6.8 Cross Platform Compatibility

When we asked about cross-platform compatibility from developers in the online survey then they rated native approx. 2.5/5 and for Web 4/5. So, why native even got 2.5 if it is platform dependent and needs to be developed for every OS?

Because of the hybrid solution, that can be developed at once and can be built for multiple platforms. They can be hosted in a device using Web view and thus are capable of accessing device features. However, comparing hybrid apps with native or Web was not the part of our research work.

Developers also said that the native applications require more development effort and maintenance time. If we have to fix something in our application then we must do it for all platforms, but for Web, fixing it for one should work for all. Also, developing native applications for more than one platform takes much more time as compared to Web.

But, this platform dependency of native apps should not be a big problem for manufacturing industries as long as they wish to use MES application for their internal use. However, If the employer wants to sell their MES solutions to other companies where the requirement might be to use different platforms. Then Web, a platform independent technology, might be a good option for those firms who sell their MES to other companies and wish to serve a huge number of their customers with their services.

6.9 App Stores

The developers think that app stores are very important feature of mobile applications. It is good for distribution purposes, developers can get feedback of the application they developed and they have their own profile in app stores which might be good for showing their experience.

However, if we think in terms of MES application then most of the MES applications are usually customized according to the industry needs. It would be slightly different for a company that manufactures product A than the company that manufactures product B. Both companies might be interested in having their customized MES that would benefit more based on their business. And, the app store is generally for distribution, i.e. to sell the software to a large number of audience, which might not be so important in the case of MES applications. Firstly, the app store usually has general applications that might be useful in daily life and secondly you can't just know the requirement of a manufacturing company beforehand and put it in app stores. Thirdly, if you are developing the MES application for a company then no need to put it on app store due to the copyright problem.

6.10 Conclusion

The portable MES, like the mobile interface of MES, can benefit workers in several ways. Mobile MES would help and facilitate operators in manufacturing industries to perform manufacturing tasks in a more efficient and better way on a factory floor. The application takes operators' interest into account that will definitely motivate operators to perform their tasks on the factory floor. Also, the easiness of performing manufacturing task would increase production efficiency in manufacturing industries. The MES in mobile device helps operators to use their time efficiently that can result in high productivity. Having MES on own mobile device gives freedom to the operators that allows them to use the application whenever they want. This freedom of use is not possible in desktop based terminals which are available in limited amounts in a factory.

The criteria to choose a technology for the development of specific software like MES are based on certain factors, i.e. cost (time, funds), available resources, requirements and target audience. One technology is not better than the other but it has several benefits and drawbacks over one another. The Web technology is best suited to target a large number of audience and to target multiple platforms, i.e. you develop it once and it works on all platforms. Users don't need to download and install it from app stores, but they can access it directly by just typing URL on browsers. Deploying a Web app is easier than native apps, i.e. they are not uploaded to app stores and no approval is needed, unlike native apps.

On the other hand, the native technology, performance wise, is better than Web. JavaScript performs slower than Java but the difference is not that big, and in future, we believe that with the improvement in JavaScript engines like Chrome V8, Spider Monkey, and others, the JavaScript engine will soon be able to compete with native code engines like JVM (Java Virtual Machine) and others. The native applications perform very smoothly whenever we scroll the list and move the 3d animated objects with fingers. This is because the FPS rate in native application is greater than Web. But we have seen that there are some hardware accelerated CSS and other techniques through which we can achieve a better FPS rate in Web. WebGL API in Web has some security vulnerabilities due to untrusted code. Native applications respond faster than Web on button tapping. It is possible that if something working in one browser, it might not work in other. One common example is that the appearance of Web page very often looks different in different browsers due to different rendering engines that every browser has. So, even though Web is platform independent but the browsers behave differently in few cases. Thus, cross compatibility of browsers is a problem in Web technology. Native apps are installed on device, so it can access device hardware more efficiently whereas Web has some limitations on accessing device features. App stores of native apps are a good source for monetization and distribution, but for MES applications, app stores are not important.

Keeping the requirements of MES application in mind, we believe that native application would be a better choice for MES interface development because it's more responsive and more secure than Web. The fast response of native application gives a better user experience to operators which help them to perform better. No company would want any external entity to attack and access its data from outside world, so native is the best solution for more secured application. In addition to these factors, we have seen that the MES interface needs a heavy use of device features like camera, magnetometer, GPS, GPU, Wi-Fi etc. This factor also becomes a reason to choose native over Web for MES applications. However, the time and cost of developing a native application are more than Web, so industries would have to invest much more in order to have more responsive and more secure MES interface.

7. SUMMARY

In this thesis, we have tried to find out the best technology, i.e. Web app or native app, for mobile MES. The thesis has two parts, one is to develop prototypes for MES and the other is evaluating the two technologies. In the first part, to evaluate the two technologies, we developed two prototypes, i.e. Web and Android native applications. In the second part, we compared HTML5 and native technologies.

MES stands for Manufacturing Execution System which is used in manufacturing industries to control and manage workflow. It also keeps tracking of all information coming from other sources like machine monitors and ERP. Mobile MES is a part of the LeanMES concept. The concept was designed to improve manufacturing processes in industries.

The existing desktop based MES in FinnPower has some functional limitations and accessibility problems. The terminals are limited in quantity and placed at distant location from machines and raw materials used in a factory. The operators working in a factory have limited information about orders and product specifications. Therefore, we chose and developed a portable mobile-based MES technology. Mobile technology has some extra features which can be used in MES system to enhance the productivity and efficiency of work in the factory floor. The GPS feature can be used to track the position of workers. The WI-FI or gyroscope can be used for indoor navigation. The camera is used for barcode/QRCode detection that can be used for item lot tracking.

We performed simple benchmark test to compare the performance of JavaScript and Java. We made a very simple app which generates the random number and sorts them using bubble sort which has n^2 average time complexity. The application calculates the time (in milliseconds) taken to generate and sort random numbers. The result was a bit different than what we expected. The difference of computation time was not that big due to the JIT (Just-In-Time) compilers that modern browsers use nowadays. The Web app was tested on Chrome and Mozilla.

Then we found out that native app has better FPS (Frames per Second) than Web app. The FPS of native app was more consistent and higher than of Web app. But, there are some hardware-accelerated CSS which make use of the GPU that could be used to increase the rendering performance of Web app.

The 300 millisecond tap delay is also one of the factors that is responsible for the performance difference between Web and native app. This is due to the fact that browsers used to rely on double tap for zooming of images and other Web contents. So, they wait 300 milliseconds for the second tap. But fortunately, Google Chrome, Mozilla Firefox, and IE have resolved this problem by enabling the zooming feature using pinch gesture. However, Safari browser still has 300 milliseconds delay. We wrote a very simple app which calculates the delay between tap event and firing of the actual click event. The result confirmed our study from literature and showed that Safari has maximum tap delay. The delay was about 270 milliseconds for Safari.

HTML5 provides WebGL API for rendering 3D objects. Before WebGL, browsers had to rely on external plugins like Silverlight and Flash. For example, Netflix relied on Silverlight for a long time. So, WebGL is a good feature to avoid the use of an external plugin. However, WebGL has some security vulnerabilities, which make it unsafe as compared to OpenGL ES which is used in mobile devices platform. The untrusted Web page can use your GPU card without having your permission. The WebGL has two buffers, i.e. fragment shader and vertex shader. If the vertex shader has some invalid index then it can cause out of range exception. Similarly, if shader is expensive to compute and takes a long time to compute, then GPU card becomes unresponsive and stops working. In addition to these, Using of uninitialized memory allows them to access the content of other windows opened on your screen. Khronos group is working on the security of WebGL and they have been able to resolve some of the security problems of WebGL while some of them are still left.

Then we had an online survey to find out the differences w.r.t development and maintenance of technologies. 18 professional software developers, who are working in companies, took part and filled the online survey. HTML5 is cross platform compatible which works on all devices while the native app is platform dependent. So, developers said that HTML5 is easier to develop and maintain because we have to develop a separate version of the native app for different platforms which require more effort and time. They rated native application high in hardware accessibility because they think that native applications can use device features more efficiently than a Web app. In addition to this, they said app stores are good for making money from your app; however, sharing the revenue with the app store is a problem. App stores are a one-stop shop which makes easy for users to download and install the app. That is why; all big names have their app on app stores. They also said that app stores are good to show the experience that we have.

Due to the security and performance of the native app, we suggest native app would be the best choice for mobile MES. But, relatively high development skills would be need-

ed to develop native applications and it will also need more time and budget for development and maintenance.

REFERENCES

- [1] Olson, P. 2012. "5 Eye-Opening Stats That Show The World Is Going Mobile". Forbes. [<http://www.forbes.com/sites/parmyolson/2012/12/04/5-eye-opening-stats-that-show-the-world-is-going-mobile/>]. Retrieved: 15.12.2015
- [2] Colao, J.J. 2012 "Facebook's HTML5 Dilemma, Explained". Forbes. [<http://www.forbes.com/sites/jjcolao/2012/09/19/facebooks-html5-dilemma-explained/>]. Retrieved: 15.12.2015
- [3] BURD, B. "Mobile operating systems and fragmentation: the insider's point of view", androidauthority, [<http://www.androidauthority.com/fragmentation-the-insiders-point-of-view-618427/>]. Retrieved: 15.12.2015
- [4] Lanz, M. & Jaervenpaeae, E. 2015. "LeanMES". Tampere University of Technology. [<https://wiki.tut.fi/LeanMES/>]. Retrieved: 15.12.2015
- [5] Tesfy, W.B., & Aleksy, M. & Andersson, K. & Lehtola, M. "Mobile Computing Application for Industrial Field Service Engineering: A Case for ABB Service Engineers" in "The 7th IEEE LCN Workshop On User MOBility and VEHicular Networks", Sydney. NSW. 2013. pp. 188-193
- [6] Lyer, V. 2015. "5 Reasons to Build Mobile Apps for the Manufacturing Industry", appsFreedom, [<http://www.appsfreedom.com/5-reasons-build-mobile-apps-manufacturing-industry/>], Retrieved: 15.12.2015
- [7] Katz, J. 2012. "Mobile Apps Break Into Manufacturing", IndustryWeek, [<http://www.industryweek.com/companies-amp-executives/mobile-apps-break-manufacturing/>]. Retrieved: 15.12.205
- [8] Jobe, W. "Native Apps vs. Mobile Web Apps" In International Journal of Interactive Mobile Technologies (iJIM), vol 7. 2013 pp 27-32
- [9] Holzer, A., Ondrus. "Mobile Application Market: A Developer's Perspective" In Telematics & Informatics. Elsevier. 2011, pp. 22-31
- [10] Mahemoff, M. 2011 "HTML5 vs Native: The Mobile App Debate. [<http://www.html5rocks.com/en/mobile/nativedebate/>]. Retrieved: 15.12.2015
- [11] Avancini, A., Ceccato, M., 2011. "Security Testing of Web Applications: a Search Based Approach for Cross-Site Scripting Vulnerabilities" In 11th IEEE International Working Conference on Source Code Analysis and Manipulation, Williamsburg, VI, 2011, PP. 85-94.

- [12] Erkkila, J.P., “Web and Native Technologies in Mobile Application Development”, M.S. Thesis, Dept. CSE., Aalto Univ., Espoo, 2013
- [13] Burlingame, CA. “HTML5 Performance 8X Slower On Mobile Than Desktop, According to PerfMarks II Study By spaceport.io”, prweb, [http://www.prweb.com/releases/2012/5/prweb9531647.htm], Retrieved: 15.12.2015
- [14] Sin, D. & Lawson, E. & Kannoopatti, K., “Mobile web apps – the non-programmer’s alternative to native applications” In 5th International Conference on Human System Interactions, Perth, WA, 2012, pp. 8-15
- [15] Juntunen , A., Jalonen, E., & Luukkainen, S., “HTML 5 in Mobile Devices – Drivers and Restraints” In 46th Hawaii International Conference on System Sciences, Wailea, Maui, HI, 2013, pp. 1053-1062
- [16] Selvarajah, K. & Craven, M.P. & Massey, A. & Crowe, J. & Vedhara, K. & Raine-Fenning, N., “Native Apps versus Web Apps: Which is Best for Healthcare Applications?”, In 15th International Conference, HCI International, Application and Services , Las Vegas, NV, 2013, pp 189-196
- [17] Wang, Chao. & Duan, W. & Ma, J. & Wang, Chenhui., “The research of Android System architecture and application programming” In International Conference on Computer Science and Network Technology, vol 2, Harbin, 2011, pp. 785-790
- [18] Android developers official site. [http://developer.android.com/sdk/index.html] Retrieved: 15.12.2015
- [19] Smith, S. 2013, “Android SDK: Common Android Components”, envatotuts+, [http://code.tutsplus.com/tutorials/android-sdk-common-android-components--mobile-20873], Retrieved: 15.12.2015
- [20] Raval, C. & Shubham. 2015. “What is Intent in Android”, Stackoverflow, [http://stackoverflow.com/questions/6578051/what-is-intent-in-android], Retrieved: 15.12.2015
- [21] Android developers official site, “Intents and Intent Filters”, [http://developer.android.com/guide/components/intents-filters.html], Retrieved: 15.12.2015

- [22] Warren, C. 2010, “5 Platforms that Defined the Mobile Space in 2010 [Mashable Awards]”, Mashable, [<http://mashable.com/2010/10/15/defining-mobile-platforms/#Kxm46bnJ2iqc>], Retrieved: 15.12.2015
- [23] International Data Corporation, “Smartphone OS Market Share, 2015 Q2”, [<http://www.idc.com/prodserv/smartphone-os-market-share.jsp>], Retrieved: 15.12.2015
- [24] Mohsen, A. & Jansen, S., “Evaluating Architectural Openness in Mobile Software Platforms” In 4th European Conference on Software Architecture: Companion Volume, Copenhagen, 2010, pp. 85-92
- [25] iOS Developer Library, “iOS Technology Overview”, [<https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html>], Retrieved: 15.12.2015
- [26] Wikipedia, “Web Application”, [https://en.wikipedia.org/wiki/Web_application], Retrieved: 15.12.2015
- [27] W3C official site, “Document Object Model (DOM)”, [<http://www.w3.org/DOM/>], Retrieved: 15.12.2015
- [28] Rouse, M. “Javascript Defination”, SearchSOA, [<http://searchsoa.techtarget.com/definition/JavaScript>], Retrieved: 15.12.2015
- [29] Birnir, A. 2014, “The first programming language you should learn is JS”, [<http://searchsoa.techtarget.com/definition/JavaScript>], Retrieved: 15.12.2015
- [30] Bajaj, K. & Pattabiraman, K. & Mesbah, A., “An Empirical Study of Client-Side JavaScript Bugs”, 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, MD, 2013, pp. 55-64
- [31] Belqasmi, F. & Glitho, R. & Chunyan Fu, “RESTful web services for service provisioning in next-generation networks: a survey” In IEEE Communications Magazine, vol 49, 2011, pp. 66-73
- [32] West, M., 2013, “Getting Started with Grunt”, treehouse, [<http://blog.teamtreehouse.com/getting-started-with-grunt>], Retrieved: 15.12.2015
- [33] Balasubramanee, V. & Wimalasena, C. & Singh, R. & Pierce, M., “Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development” In 2013 IEEE International Conference on Cluster Computing (CLUSTER), Indianapolis, IN, 2013, pp. 1.

- [34] Airpair, “AngularJS Tutorial: A Comprehensive 10,000 Word Guide”,
[<https://www.airpair.com/angularjs>], Retrieved: 15.12.2015
- [35] RisingStack blog, “The React.js Way: Getting Started Tutorial”,
[<https://blog.risingstack.com/the-react-way-getting-started-tutorial/>], Retrieved:
15.12.2015
- [36] Techsutram, 2010, “HTML VS HTML5”,
[<http://www.techsutram.com/2010/09/html-vs-html5.html>], Retrieved: 15.12.2015
- [37] Lee, J. 2013, “What Is HTML5, And How Does It Change The Way I Browse?
[MakeUseOf Explains]”, [<http://www.makeuseof.com/tag/what-is-html5-and-how-does-it-change-the-way-i-browse-makeuseof-explains/>], Retrieved: 15.12.2015
- [38] Wikipedia. 2012. “Windows Phone”,
[http://en.wikipedia.org/wiki/Windows_Phone] Retrieved: 25.12.2016.
- [39] Nevalainen, J., 2012, “Windows Phone 8 Kernel Architecture”,
[<http://j4ni.com/blog/?p=107>], Retrieved: 15.12.2015
- [40] WhiteChapel, A. & McKenna, S., “Vision and Architecture” In Windows Phone 8
Development Internal, 1st ed., Microsoft, pp. 8-17
- [41] Tutorials Point, “Android - Architecture”,
[http://www.tutorialspoint.com/android/android_architecture.htm], Retrieved:
15.12.2015
- [42] Wikipedia, “Manufacturing Execution Systems”,
[https://en.wikipedia.org/wiki/Manufacturing_execution_system], Retrieved:
15.12.2015
- [43] Rajput, R.K., “Concept of Manufacturing” In Manufacturing Technology, pp. 1-
37
- [44] Wikipedia, “Sequence Diagram”,
[https://en.wikipedia.org/wiki/Sequence_diagram], Retrieved: 19.12.2015
- [45] Wikipedia, “Nesting(process)”, [[https://en.wikipedia.org/wiki/Nesting_\(process\)](https://en.wikipedia.org/wiki/Nesting_(process))],
Retrieved: 19.12.2015
- [46] Wikipedia, “Object-Relational mapping”, [https://en.wikipedia.org/wiki/Object-relational_mapping], Retrieved: 19.12.2015

- [47] WebStandards, “HTML Versus XHTML”,
[<http://www.webstandards.org/learn/articles/askw3c/oct2003/>], Retrieved:
17.12.2015
- [48] CodeAurora, “Measuring FPS on the web”,
[<https://www.codeaurora.org/blogs/mbapst/measuring-fps-web>], Retrieved:
25.12.2015
- [49] Lawson, N., 2012 , “The Quest for Smooth Scrolling”,
[<http://www.pocketjavascript.com/blog/2015/2/3/the-quest-for-smooth-scrolling>],
Retrieved: 25.12.2015
- [50] JT, 2014, ”Optimising for 60fps Everywhere”,
[<https://engineering.gosquared.com/optimising-60fps-everywhere-in-javascript>],
Retrieved: 27.1.2015
- [51] Lewis, P., 2012, “Scrolling Performance”,
[<http://www.html5rocks.com/en/tutorials/speed/scrolling/>], Retrieved: 28.12.015
- [52] Weil, A., 2015, “Native app or Mobile app, Where do Customers Spend More Money?”, [<http://www.luxurydaily.com/native-app-or-mobile-web-where-do-consumers-spend-more-money/>], Retrieved: 28.12.2015
- [53] Google, “Patterns - Gestures”,
[<https://www.google.com/design/spec/patterns/gestures.html>], Retrieved:
28.12.2015
- [54] Archibald, J., “300ms tap, gone away ”,
[<https://developers.google.com/web/updates/2013/12/300ms-tap-delay-gone-away?hl=en>], Retrieved: 3.1.2016
- [55] VanToll, TJ., 2015, ”The 300ms Click Delay and iOS 8”,
[<http://developer.telerik.com/featured/300-ms-click-delay-ios-8/>], Retrieved:
4.1.2016
- [56] Hoffman, C., 2014, “Why Browsers plug-Ins Are Going Away and what’s Replacing Them”, [<http://www.howtogeek.com/179213/why-browser-plug-ins-are-going-away-and-whats-replacing-them/>], Retrieved: 10.1.2016
- [57] Roettgers, J., 2014, “No blackout ahead: Netflix already ditched Silverlight for chrome”, [<https://gigaom.com/2014/11/26/netflix-silverlight-chrome/>], Retrieved:
10.1.2016

- [58] Tamats, 2014, “Things I hate about WebGL”,
[<http://www.tamats.com/blog/?p=604>], Retrieved: 10.1.2016
- [59] Jones, B., 2010, ”WebGL’s greatest challenge as gaming platform”,
[<http://blog.tojicode.com/2010/07/webgls-greatest-challenge-as-gaming.html>],
Retrieved:15.1.2016
- [60] Hernandez, G., 2012, “Increase your Site’s Performance with Hardware- Accelerated CSS”, [<http://blog.teamtreehouse.com/increase-your-sites-performance-with-hardware-accelerated-css>], Retrieved: 17.1.2016
- [61] Mozilla, 2015, ”WebGL best practices”, [https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/WebGL_best_practices], Retrieved:7.1.2016
- [62] Lorenzo, DC., “WebGL™ Optimizations for Mobile”,
[<http://malideveloper.arm.com/downloads/GDC14/Thursday/10.30amWebGL.pdf>], Retrieved: 20.1.2016
- [63] Gomez, “Why Web Performance Matters: Is Your Site Driving Customers Away?”,
[http://www.mcrinc.com/Documents/Newsletters/201110_why_web_performance_matters.pdf], Retrieved: 25.1.2016
- [64] Benchmarksgame,”The Computer Language Benchmarks game”,
[<http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=binarytrees>], Retrieved: 25.1.2016
- [65] Geekregator, 2015, “Benchmarking C, Java, JavaScript and FreePascal”,
[https://geekregator.com/2015-01-15-benchmarking_c_java_javascript_and_freepascal.html], Retrieved: 15.1.2016
- [66] Lewis, P., 2013, “The Runtime Performance Checklist”,
[<http://calendar.perfplanet.com/2013/the-runtime-performance-checklist/>], Retrieved: 14.1.2016
- [67] Peddie, J., 2011, “WebGL Security – Kill it before it grows”,
[<http://jonpeddie.com/blogs/comments/webgl-security-kill-it-before-it-grows/>], Retrieved:
- [68] KHRONOS, “WebGL Security”, [<https://www.khronos.org/webgl/security/>], Retrieved:14.1.2016
- [69] Wikipedia, “Windows Phone”, [https://en.wikipedia.org/wiki/Windows_Phone], Retrieved: 9.2.2016

- [70] Android developers official site,
[<http://developer.android.com/tools/performance/profile-gpu-rendering/index.html>] Retrieved: 14.2.2016
- [71] Android developers official site,
[<https://developers.google.com/web/fundamentals/performance/rendering/?hl=en>]
Retrieved: 14.2.2016
- [72] Johnston, M., 2015, “60 FPS on the Mobile Web”,
[<http://engineering.flipboard.com/2015/02/mobile-web/>], Retrieved: 15.2.2016
- [73] Benchmark Games,
[<http://benchmarkgame.alieth.debian.org/u64q/javascript.html>], Retrieved:
15.2.2016
- [74] Cozzi, P., & Riccio, P. “OpenGL Insights”, Taylor & Francis Group, LLC, pp. 27-46
- [75] Cozzi, P., & Riccio, P. “OpenGL Insights”, Taylor & Francis Group, LLC, pp. 47-60
- [76] Empaempa, “Using Float32Array slower than var”,
[<https://github.com/empaempa/GLOW/issues/3>], Retrieved: 16.2.2016
- [77] Macdonald, A., “JavaScript Typed Arrays”,
[<https://bocoup.com/weblog/javascript-typed-arrays>], Retrieved: 16.2.2016
- [78] Fimecc, [<https://www.fimecc.com/content/manu-future-digital-manufacturing-technologies-and-systems>], Retrieved: 5.5.2016
- [79] Spring, [<https://spring.io/>], Retrieved: 6.5.2016
- [80] Hibernate, [<http://hibernate.org/orm/>], Retrieved: 6.5.2016
- [81] Postgresql, [<http://www.postgresql.org/>], Retrieved: 6.5.2016
- [82] Apple,
[<https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>], Retrieved: 8.5.2016
- [83] Apple,
[https://developer.apple.com/library/mac/documentation/Darwin/Conceptual/KernelProgramming/BSD/BSD.html#//apple_ref/doc/uid/TP30000905-CH214-TPXREF101], Retrieved: 8.5.2016

APPENDIX A: ONLINE SURVEY (THE DEVELOPER'S PERSPECTIVE ON WEB VS NATIVE TECHNOLOGIES)

1/28/2016 The Developer's Perspective on Web vs Native App

The Developer's Perspective on Web vs Native App

*Required

1. **Age ***
Mark only one oval.

20 - 29
 30 - 39
 40 - 50

2. **Overall Working experience as software developer. ***
Mark only one oval.

1 - 3 years
 3 - 6 years
 More than 6 years

3. **Mobile application (web or native) development experience in particular. ***
Mark only one oval.

0 - 1 years
 1 - 3 years
 3 - 5 years
 More than 5 years

4. **If you are asked to choose one mobile platform to work on, what would you choose? ***
Mark only one oval.

Android
 iOS
 Windows Phone

5. **From your experience and skills, what would you call yourself? ***
Mark only one oval.

Web Application Developer
 Mobile Application (Native) Developer

https://docs.google.com/forms/d/1y6ibOuXqxdTdnr0j4W2wq-eLL3lUUsXUEwnB-xeyVwc/edit?usp=drive_web 1/5

Figure 21. Questionnaire for Online survey, page 1/5

1/28/2016 The Developer's Perspective on Web vs Native App

6. **How easy and fast it to develop and design application? Consider the skills and experience needed to work with each of these technologies. Also consider the range of development tools available for both technologies. ***
Mark only one oval per row.

	very difficult (1)	difficult (2)	normal (3)	easy (4)	very easy (5)
Web	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Native	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

7. **Add some comments If you wish to justify above answer.**

.....

.....

.....

.....

.....

8. **Rate the ability of the technology to access device's native features such as gyroscopes, camera, GPS, contact list etc. ***
Mark only one oval per row.

	Not good at all (1)	Not good (2)	Good (3)	Very good (4)	Excellent (5)
Web	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Native	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

9. **Add some comments If you wish to justify above answer.**

.....

.....

.....

.....

.....

10. **App stores for monetization ***
Mark only one oval.

Not useful at all

Useless

useful to some extent

useful

very useful

https://docs.google.com/forms/d/1y6ibOuXqxdTdnr0j4W2wq-eLL3lUUsXUEwnB-xeyVwo/edit?usp=drive_web
2/5

Figure 22. Questionnaire for Online survey, page 2/5

1/28/2016 The Developer's Perspective on Web vs Native App

11. **Add some comments if you wish to justify above answer.**

.....

.....

.....

.....

.....

12. **App stores for making your application available. ***
Mark only one oval.

Not useful at all

Useless

Useful to some extent

Useful

Very useful

13. **Add some comments if you wish to justify above answer.**

.....

.....

.....

.....

.....

14. **Rate the cross platform compatibility of technology. ***
Mark only one oval per row.

	Not good at all (1)	Not good (2)	Good (3)	Very good (4)	Excellent (5)
Web	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Native	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

15. **Add some comments if you wish to justify above answer.**

.....

.....

.....

.....

.....

https://docs.google.com/forms/d/1y6ibOuXqxdTdnr0j4W2wq-eLL3lUUsXUEwnB-xeyVwc/edit?usp=drive_web 3/5

Figure 23. Questionnaire for Online survey, page 3/5

1/28/2016 The Developer's Perspective on Web vs Native App

16. How would you rate the technology, if it is used for the development of mobile based ERP. *
Mark only one oval per row.

	Not good at all (1)	Not good (2)	Good (3)	Very good (4)	Excellent (5)
Web	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Native	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

17. Add some comments if you wish to justify above answer.

.....

.....

.....

.....

18. Rate the maintenance of the technologies. *
Mark only one oval per row.

	Very difficult (1)	Difficult(2)	Normal (3)	Easy (4)	Very Easy (5)
Web	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Native	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

19. Add some comments if you wish to justify above answer.

.....

.....

.....

.....

20. How easy is it to deploy? *
Mark only one oval per row.

	Very Difficult (1)	Difficult (2)	Normal (3)	Easy (4)	Very Easy (5)
Web	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Native	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

https://docs.google.com/forms/d/1y6ibOuXqxdtDnrOj4W2wq-eLL3UUuSXUEwnB-xeyVwc/edit?usp=drive_web 4/5

Figure 24. Questionnaire for Online survey, page 4/5

1/28/2016 The Developer's Perspective on Web vs Native App

21. **Add some comments if you wish to justify above answer.**


.....
.....
.....
.....
.....

22. **Troubleshooting or fixing a bug ***
Mark only one oval per row.

	Very Difficult (1)	Difficult (2)	Normal (3)	Easy (4)	Very Easy (5)
Web	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Native	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

23. **Add some comments if you wish to justify above answer.**

.....
.....
.....
.....
.....

Powered by


https://docs.google.com/forms/d/1y6ibOuXqxdTdnr0j4W2wq-eLL3lUUsXUEwnB-xeyVw/edit?usp=drive_web 5/5

Figure 25. Questionnaire for Online survey, page 5/5

APPENDIX B: EVALUATION TEST BY PARTICIPANTS

Table 7. List of steps performed during usability test by participants

Steps	Description
1	Login with username “janne” and password “operator”.
2	Fill up the profile with your interests and skills.
3	Go to the order screen and search the order by keyword
4	Filter Order list by interest and then by skills
5	Remove filters on list
6	Place an order of part “pp_part3” with work type “Bending” and the quantity should be 5
7	Start that order; select the amount of items you want to start with, let’s say 3
8	Go to the process list and check if the order you started is running?
9	If it is running then stop it. Wait for 5 seconds and then resume it again and wait till the process ends.
10	Finish the process that just completed. Mark it as successful if it successfully completed otherwise mark it as failure and give the reason of failure.
11	Go the Item store screen and add the finished item to the store. Select the type of work that the item went through. Add the item in store A1 with quantity 2 (although we processed 3 quantity of item)
12	Add the remaining one part of “PP_PART3”.
13	Move 2 quantities of item from store A1 to B1.

14	View the item, rotate it, zoom it.
15	Go to the map screen and scan the QRCode to check the map. What does the map say?
16	Logout

Table 8. *List of questions that were asked from participant after they performed all steps of usability test*

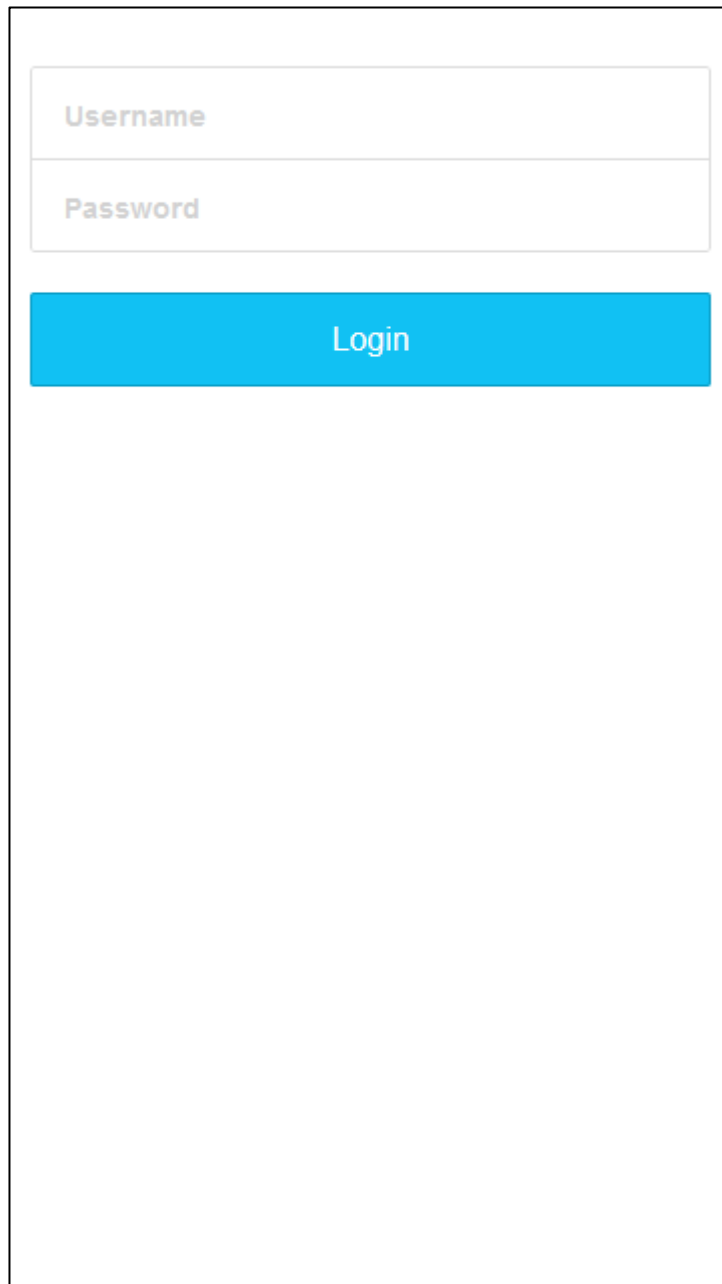
Have you been able to perform all steps smoothly?
Was the process updates from the server smooth? The progress bar was running smooth? Did the stop, finish and hold button respond quickly or there was any delay? Compare web and native tap gesture.
How well is the interaction with 3d image of the items? Are you able to enlarge the image in web and in native? How smooth is the rotation of the image both in web and native?
The barcode responded quickly? Did it localize the bar code by itself? How quickly did the scanning process take place?
Scroll the order screen as fast as you can. Was the update of the screen smooth?
Is the web page rendering fast enough as compare to native?

GUI: intuitive, responsive, Clear, consistent, Attractive. Compare web and native.

Overall which app was better in terms of performance?

What are the extra features that MES interface could have?

APPENDIX C: SCREEN SHOTS OF MOBILE MES



The image shows a mobile login screen. It features two input fields: the top one is labeled "Username" and the bottom one is labeled "Password". Below these fields is a prominent blue button with the text "Login" centered on it. The entire interface is contained within a thin black rectangular border.

Figure 26. Login Screen

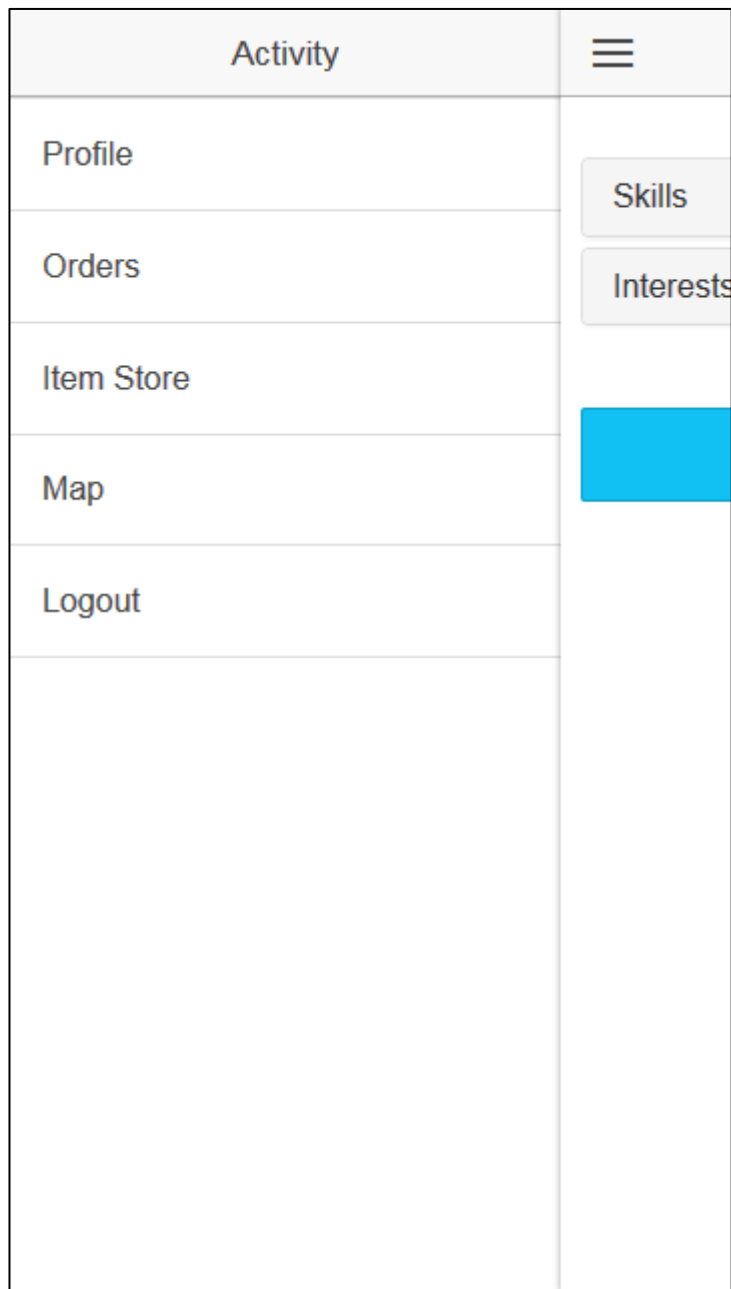


Figure 27. Side Menu for navigation

Add item to store!

FP_006

Welding

Floor A2

testerp

- Quantity 2 +

Scan Item

Done Cancel

Item Name: FP_006
Work Type: Nesting
Quantity: 1
Store: Floor A1
ERP Reference: testerp
Move

Item Name: Box
Work Type: Tool Maintenance
Quantity: 5
Store: Floor A1

Figure 28. Popup that appears when you add item to Store

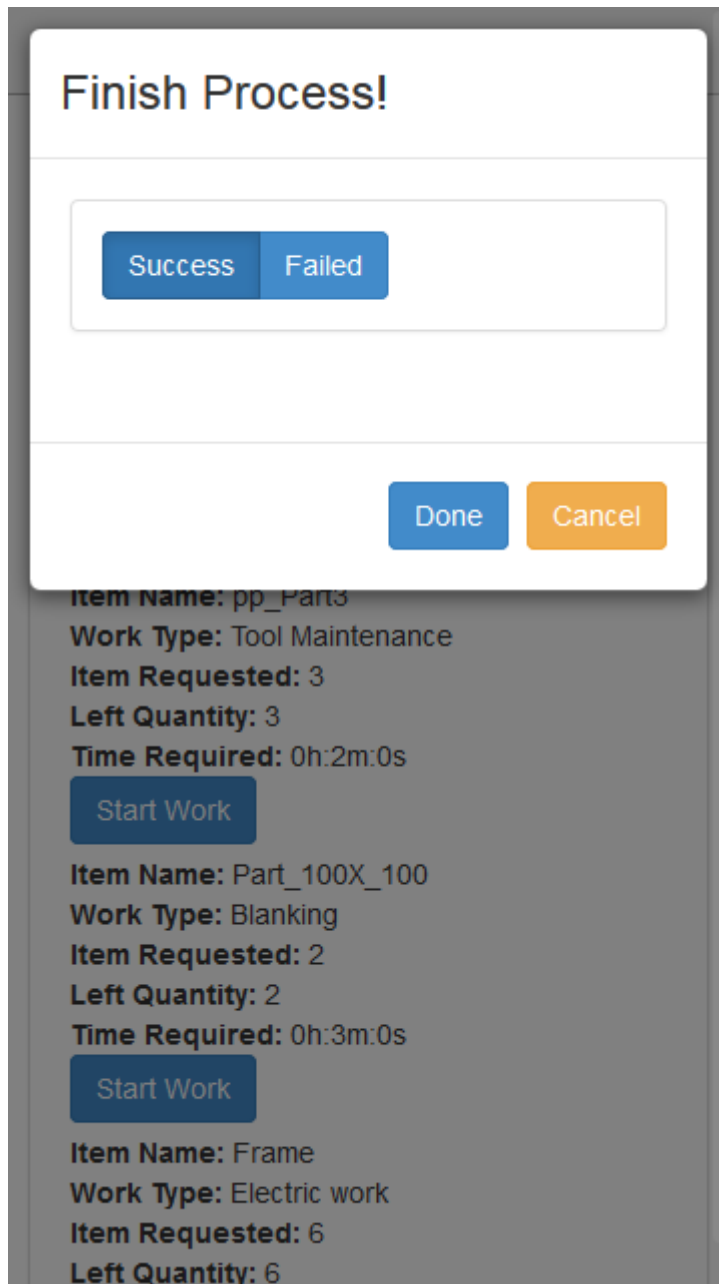


Figure 29. Popup that appears when you finish the processed item

☰ Item Stores

Item Name: [FP_006](#)
Work Type: Nesting
Quantity: 1
Store: Floor B1
ERP Reference: testerp
[Move](#)

Item Name: [FP_006](#)
Work Type: Nesting
Quantity: 1
Store: Floor C2
ERP Reference: testerp
[Move](#)

Item Name: [FP_006](#)
Work Type: Nesting
Quantity: 1
Store: Floor A1
ERP Reference: testerp
[Move](#)

Item Name: [Box](#)
Work Type: Tool Maintenance
Quantity: 5
Store: Floor A1
ERP Reference: testerp
[Move](#)

Item Name: [Box](#)

Figure 30. Item store screen which shows where the item is placed, what is the quantity and with what manufacturing step it went through

Move Item!

Item Information

place: Floor A1
erp_reference: testerp
name: Box
workstep: Tool Maintenance
production balance: 5

Target Place

- Quantity 2 +

Floor B1 ▼

Done Cancel

Item Name: Box
Work Type: Tool Maintenance
Quantity: 5
Store: Floor A1

Figure 31. *Popup that appears when you move an item from one store to another*

Place an order!

- Quantity 6 +

FP_006 ▼

Bending ▼

ERP Reference:
testERP

Done Cancel

Item Name: FP_006
Work Type: Forming
Item Requested: 2
Left Quantity: 2
Time Required: 0h:3m:0s
Start Work

Item Name: pp_Part3
Work Type: Tool Maintenance
Item Requested: 3
Left Quantity: 3
Time Required: 0h:2m:0s

Figure 32. Popup that appears when we place an order

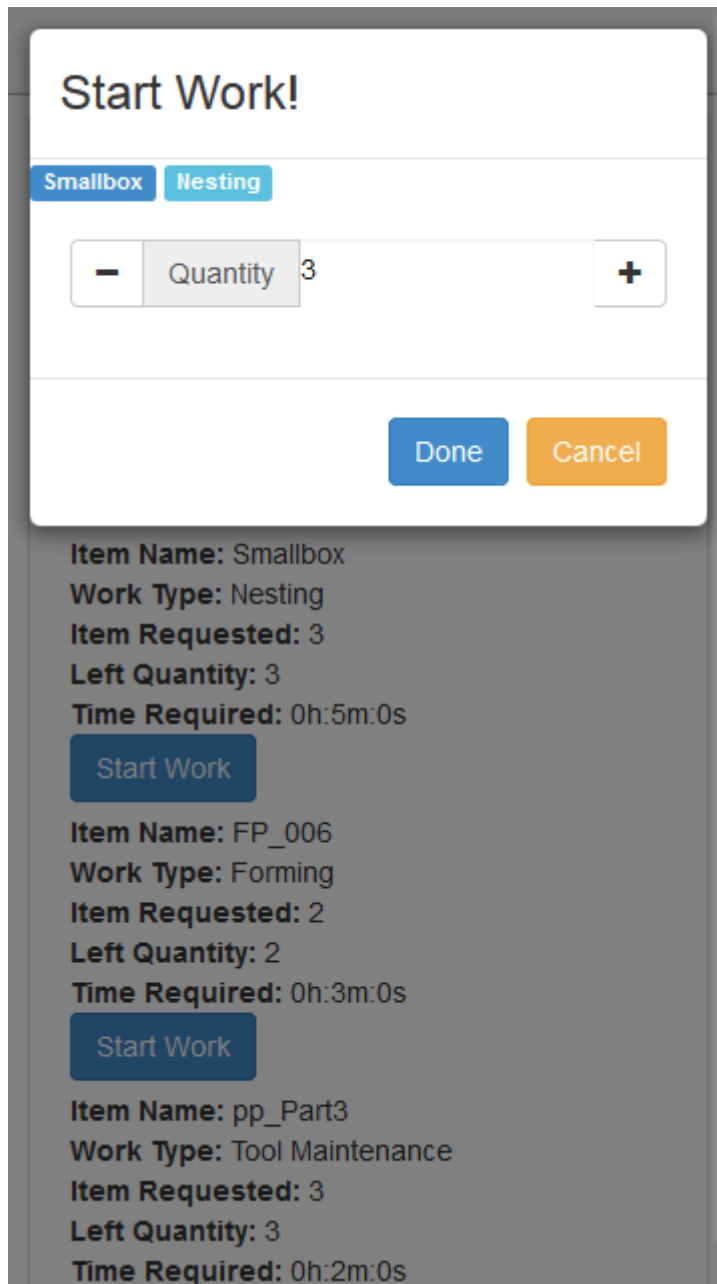


Figure 33. Popup that appears when we start to process an order

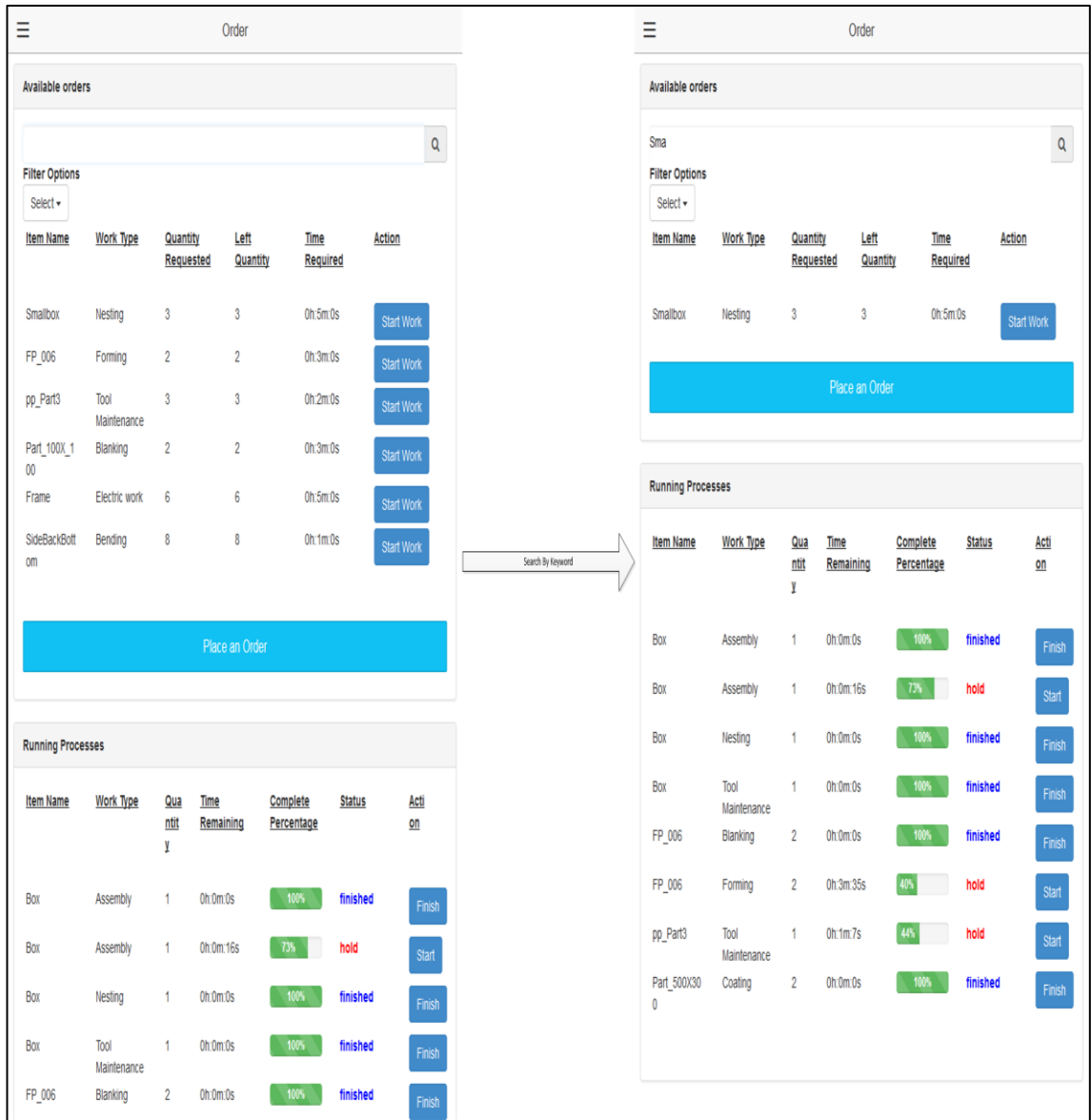


Figure 34. Filter order list by typing a keyword on search bar

Appendix D: JavaScript Code for Performance Test

```
<script>
2     var arr = [];
3     function MyRandom(myseed) {
4         this.seed = myseed;
5     }
6     MyRandom.prototype.next = function() {
7         this.seed *= 1103515245;
8         this.seed += 12345;
9         this.seed /= 65536;
10        this.seed %= 20;
11        return Math.floor(this.seed);
12    };
13    var mysort = function(n) {
14        for (var i = 0; i < (n - 1); i++) {
15            for (j = i + 1; j < n; j++) {
16                if (arr[j] < arr[i]) {
17                    temp = arr[i];
18                    arr[i] = arr[j];
19                    arr[j] = temp;
20                }
21            }
22        }
23    }
24    function a() {
25        var n = document.getElementById('txtbx').value;
26        var start = new Date().getTime();
27        var rand = new MyRandom(27);
28        for (var i = 0; i <= n - 1; i++) {
29            arr.push(rand.next());
30        }
31        mysort(n);
32        var end = new Date().getTime();
33        var time = end - start;
34        alert('Execution time: ' + time);
35    }
36 </script>
```

Program 4. Javascript code for performance test.