ASHESI

ASHESI UNIVERSITY COLLEGE

AN IMPLEMENTATION OF A WEB PLATFORM TO SUPPORT FREE RECYCLING IN GHANA

UNDERGRADUATE APPLIED PROJECT

B.Sc. Computer Science

Kwabena Bamfo

ASHESI UNIVERSITY COLLEGE

An implementation of a web platform to support free recycling in Ghana

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

Kwabena Bamfo

April 2017

DECLARATION

I hereby declare that this [capstone type] is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……………………………………………………………………………………………

Candidate's Name:

……………………………………………………………………………………………

Date:

……………………………………………………………………………………………

I hereby declare that preparation and presentation of this [capstone type] were supervised in accordance with the guidelines on supervision of [capstone type] laid down by Ashesi University College.

Supervisor's Signature:

……………………………………………………………………………………………

Supervisor's Name:

……………………………………………………………………………………………

Date:

……………………………………………………………………………………………

Acknowledgement

My sincere and utmost gratitude goes out primarily to the Almighty God for granting me the strength and wisdom to complete this dissertation. I am also thankful for the support and patience of my supervisor, family, and friends

Abstract

The advent of free recycling is largely alien and unheard of in Ghana. This is the practice of giving out unwanted but reusable items to other people who may have need of it. Freecycling is a linguistic blend of the words free and recycling which connotes the practice of giving unwanted items for free rather than discarding them in landfills. This project implements a web platform that enables local community members to advertise free items and request for free items that have been advertised.

Table of Contents

1. Chapter 1: Introduction

1.1 Background

Freecycling is a portmanteau of free recycling that describes the act of giving out unwanted but reusable items to others for free rather than discarding them in landfills (Freecycle.net, 2017). A linguistic blend of the words free and recycling, the term is most often linked with online groups who run mailing lists which offer items to registered members for free (Wikipedia, 2017). These groups underscore their activities with a mission of building a worldwide sharing movement that reduces waste, saves resources, and eases the burden on landfills (The Freecycle Network, 2017). With member groupings usually based on local residence, freecycling advocates are attempting to turn community trash into treasure with a long-term view of changing the world, one gift at a time (The Freecycle Network, 2017). Forget eBay – here’s Freebay (The Scotsman, 2006).

The concept of freecycling emanates from the idea that people are in possession of used items that have remaining useful life to individuals other than themselves (Ehrman, 2015). Rather than throwing these items away, proponents of this phenomenon encourage owners to give their unwanted items for free to others in their proximate locality who may have need of it. (Ehrman, 2015). In some instances, members of a freecycling network also have the option of broadcasting their need of a particular item within the network before consideration is given to purchasing the item from a market or some other place as appropriate.

In Ghana, the paradigm of freecycling is rather alien. Most people would rather put up their unwanted items on burgeoning electronic commerce applications such as Tonaton and

Jumia than give it for free. That notwithstanding, most of the established freecycling organizations do have outlets in Ghana. Their patronage, however, is a different issue.

1.2 Related Works

As mentioned earlier, the concept of freecycling is rather alien to the Ghanaian populace despite the existence of platforms that support the giving away of unwanted items for free. These platforms are also not without obvious pitfalls, chief among them their reliance on Yahoo Groups. Yahoo! Groups is one of the world’s largest collections of online discussion boards (Wikipedia, 2017). It refers to Internet communication which is a hybrid between an electronic mailing list and a threaded internet forum, and appears to be the backbone of almost all existing freecycling networks (Wikipedia, 2017).

The Freecycle Network is a grassroots and entirely nonprofit organization made up of 5,302 groups with 9,079,882 members around the world who are giving and receiving unwanted items for free in their various localities (The Freecycle Network, 2017). The movement was started by Deron Bel in the USA in May 2003 and the first UK group commenced its activities in October of the same year (The Freecycle Network, 2016). The network is run by volunteers and day-to-day decision making is made by grassroots local groups, each of which has a local moderator responsible for checking messages and helping members (The Freecycle Network, 2016). These local moderators are none other than volunteers who give up time to help run local groups without pay (The Freecycle Network, 2016). The organization uses a web application to facilitate its activities. Users of the application can either join a local group on site or an existing Yahoo Group through the networks web interface. After joining, users have the option to offer an item, request for an item, search for offered items and mark previously offered items of theirs as ‘taken’ in the event that the item does find a new home, after which the onus is on the recipient to mark the item as ‘received’.

Trash Nothing is another web application with a mobile app, that serves as an improved interface to already existing freecycling Yahoo Groups making these mailing groups quicker and easier to use (Trash Nothing, 2017). The website and mobile app provide two separate services – free email accounts and free hosting for freecycling groups (Trash Nothing, 2017). Akin to The Freecycle Network, after joining, users have the option to offer an item, request for an item, search for offered items and mark previously offered items of theirs as ‘taken’ in the event that the item does find a new home, after which the onus is on the recipient to mark the item as ‘received’.

Concerning free email accounts, Trash Nothing gives everyone who signs up a free trash nothing email address that can be used to subscribe to freecycling groups on Yahoo Groups (Trash Nothing, 2017). Trash Nothing’s email accounts have extended features – such as a custom inbox – which makes them quicker and easier for people to use with existing freecycling groups on Yahoo Groups (Trash Nothing, 2017). The custom inbox, itself just one of many examples of the extended services provided by Trash Nothing’s email accounts, ensures that users do not have to put up with all their group’s email in their personal email account (Trash Nothing, 2017). This strategy stems from the fact that Yahoo Groups and other email accounts are not designed particularly for freecycling (Trash Nothing, 2017).

As mentioned above, Trash Nothing also provides free hosting for freecycling groups as an alternative to Yahoo Groups. This hosting service comes with automation of many of the repetitive and tedious moderation tasks on Yahoo Groups (Trash Nothing, 2017). The platform has built-in support for many of the common freecycling rules and allows local groups to customize the rules as appropriate (Trash Nothing, 2017). In addition, posts on the platform are colour-coded by type to allow members easily search and find posts they are interested in (Trash Nothing, 2017). Trash Nothing asserts to have one driving goal – the growing and spreading of freecycling by making it quicker, easier and more accessible to everyone (Trash Nothing, 2017).

1.3 Problem and Significance of Problem

Although a budding practice in the world, especially the United States of America, the advent of freecycling is largely unknown and unheralded in Ghana, and Africa at large. To contextualize the issue – the Accra Freecycle group on Yahoo! Groups has 4 members. In no way does this suggest a different behavioural trait in Ghana as compared to nations where freecycling is blossoming. Rather, the freecycling movement depends on a service that has a low adoption rate in Ghana – Yahoo! Groups. Ghanaians are simply not engaging in free recycling because there are currently no well-known avenues for them to do so.

Ghana is a country with a rising middle class population, and with this comes an increase in expandable income and purchasing power. The more money people have, the more likely they are to spend on newer versions of consumer items they already possess. With this increased expenditure comes the need for the country's inhabitants to get rid of their unwanted items that have the propensity to clutter their homes. As a nation already battling with waste management, it is quintessential that these items do not end up in landfills, and free recycling is one means to that end.

1.4 Objective

This project would develop a web platform that allows users to advertise items they no longer have need of and are willing to give to others for free. The application would also serve as a first-stop online marketplace for people in search of a wide range of items, prior to them making any new purchases.

1.5 Overview of Remaining Chapters

The paper will next discuss the requirements of the application, after which there will be a chapter on the architecture and design considerations for the web application. The paper will then detail on the tests that were conducted on the developed application and present its accompanying results. The paper will conclude with a chapter on limitations of the project and provide suggestions for future development.

2. Chapter 2: Requirements

2.1 Overview

This section discusses the scope of this project and provides an overview of the requirement chapter.

2.1.1 Scope

This project designs and implements a web application to support the paradigm of free recycling in Ghana. Three implementation approaches were considered: a web platform, an android mobile application, and a cross-platform mobile application built with HTML5 and Phonegap. The main aim of all three options is to provide a medium for individuals to give and receive unwanted items that are reusable.

With the web platform, users will be able to add items to a persistent database and retrieve items with an underlying client-server architecture. The web application – particularly the client side – will be responsible for making requests for items in the database. The web server will be responsible for responding to these requests and serving the necessary web pages to the client.

The option of the android mobile application will allow users to swipe through items added to the web application's database. Users will then have the option to request these items either by calling the lister, sending the lister of the item a text message or an email. The mobile application will significantly simplify the process of adding an item to the application's database since users will be able to capture and upload images of their unwanted but reusable items much quicker as compared to a web application. However, building with android will render the application unavailable to a significant portion of the population, a major drawback.

The third and final option is to build a cross-platform mobile application using a service such as Phonegap. Phonegap is a mobile application development framework using standards-based web technologies to bridge web applications and mobile devices (Phonegap, 2016). This framework allows developers to build mobile applications using CSS3, HTML5, and JavaScript instead of platform-specific APIs such as Android, iOS, and Windows Phone (Fermonso, 2009). This option will have characteristics of design and implementation similar to that of the android mobile application, however, critically, it will be available across the myriad of mobile phone operating systems.

As stated earlier, this project gears towards the development of a web application to fast-track the concept of free recycling in Ghana. Web applications form the basis of many mobile applications and this will effectively serve as the foundation for future works on a mobile application integrated with the web application.

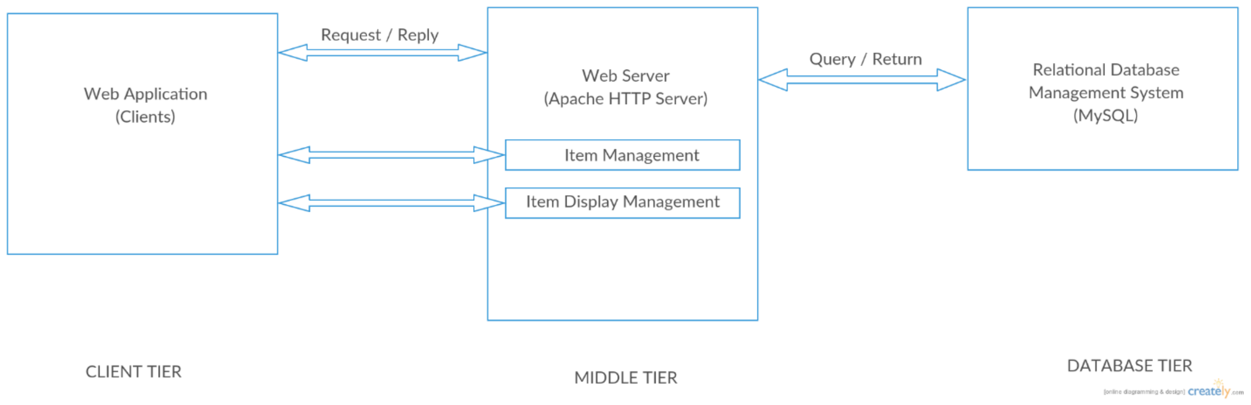


Figure 2.1: Block Diagram showing the components of the web platform

The different components of the web application are described below:

- **MySQL Database:** This is the most popular open-source database system used with PHP (W3Schools, 2017). This project makes use of the MySQL database because it is a relational database management system (RDBMS) that is ideal for both small and larger applications with the ability to store huge volumes of both data and end-users (W3Schools, 2017). For this application, the MySQL database will be used to store application data and user data as well as provide an interface to the web application for retrieving and adding to the data held by the RDBMS.
- **Apache HTTP Server:** The Apache HTTP Server is the world's most used web server software (Wikipedia, 2016). As of July 2016, Apache was estimated to serve 46.41% of all active websites and 43.18% of the top million websites (Netcraft, 2016). The Apache Server will be responsible for storing, processing and delivering web pages to clients on request using the Hypertext Transfer Protocol (HTTP) (Wikipedia, 2017).
- **Web Application Client:** The web application will serve as the client-side of the application. Users will interact with the entire application through the web platform developed using HTTP, CSS, JavaScript and the Materialize framework. The web application will have a minimal, user-centered graphical interface in an attempt not to digress from the core functions of the application. The web application will primarily give users a view of items added to the online market as well as allow users to add items to the platform.

2.1.2 Overview of Software Requirement Specification

The requirement specification in this chapter is a detailed description of the web application to be developed. The chapter will provide a holistic overall description of the requirement specification – enumerating the product functions, user classes, and implementation and design constraints. The chapter will also describe in detail the main functions of the product – breaking them down into functional requirements as well as providing a scenario and use case for each function of the product. The chapter will also specify the interfaces of the web application and end with a description of the non-functional requirements of the software.

2.2 Overall Description

This section provides an overview of the web application's requirements which are defined in detail later in the chapter. The context diagram in Figure 2.1 serves as a high-level description of the operations of the web application. The two main actors, a user with an account and a user without an account are also shown in Figure 2.1 alongside the manner of their interactions with the application. The application uses a client-server architecture where communication is based on a request-reply protocol.

(insert context diagram)

Figure 2.2: Context Diagram of Application to be Developed

2.2.1 Product Functions

The principal functions of the web application are as listed below:

- **Item Management:** This function will enable users of the application to add and consequently control their unwanted but reusable items on the online marketplace.
- **Item Display Management:** This function will enable users of the application to control their view of items on the online marketplace.

- **Auxiliary Services:** This function encompasses a set of functions that will enable the user customize their experience of the web application as well as access supporting features to the main features of the application.

2.2.2 User Classes and Characteristics

As shown in the context diagram of the web application (Figure 2.1), there are two main user classes of the application – one representing a user without an account and the other, a user with an account.

- **Application User without an Account:** This could be any individual with a practical understanding of technology who decides to use the application without logging in through the applications login portal or social media alternatives. The user should minimally be able to operate web applications with consummate ease. Gender, educational level, marital status, and the likes, are not important characteristics for this user class. The user should however be able to read in English to operate the web application as appropriate. This user would have access to less features due to the lack of authorized personal information.
- **Application User with an Account:** The application user with an account has characteristics and traits akin to that of a user without an account. The difference lies in the fact that this user decides to log in through the applications login portal or social media alternatives. This user would be privy to extra features due to the availability of authorized personal information.

2.2.3 Design and Implementation Constraints

- The web application makes use of an SMS Gateway. This Gateway is use to send text messages containing details of unwanted but reusable items added to the marketplace to prospective recipients. Due to budget and time constraints, for this project, SMS Gateway Me was used. SMS Gateway Me is an Android application that acts as a gateway using the messaging capabilities of any phone on which it is installed.
- The web application is locally hosted. (insert more details)

2.3 Specific Requirements

The functional requirements of the web application have been organized according to the major product functions: Item Management, Item Display Management, and Auxiliary Services.

2.3.1 Item Management

This section describes the Item Management requirement and provides use cases and a scenario. Detailed functional requirements are then listed and discussed.

2.3.1.1 Description and Priority

This feature allows users to add and control unwanted but reusable items in their possession. It is a high priority feature since it forms the basis of the entire web application. The use case diagram in Figure 2.2 describes a user's interactions with the web application concerning this requirement.

2.3.1.2 Use Case: Item Management

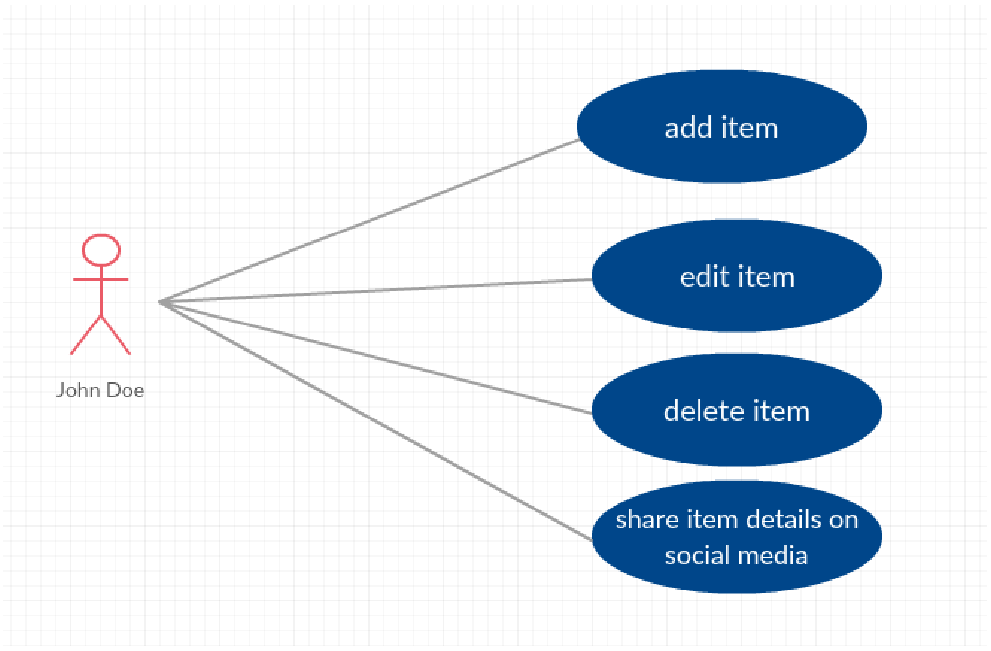


Figure 2.3: Use Case Diagram for Online Marketplace Item Control

2.3.1.3 Requirement Scenario: Item Management

Mr. A, a frequent user of the Internet, navigates to the web application's homepage through its Uniform Resource Locator (URL). Mr. A's home is currently cluttered with many items he no longer has need of, a sight which is very much displeasing to him. Since the items are in a reusable condition, Mr. A decides against putting them in the trash bin, opting instead to advertise his unwanted items on the web application. He takes a picture of an unwanted item with his mobile device, uploads the image unto his personal computer, and adds the item with the necessary details to the web application. Mr. A repeats the actions stated prior for all the items in his home he wishes to do without.

2.3.1.4 Functional Requirements

REQ-IM-1: A user should be able to add an item to the market

Table 2.1: Table of Requirements for REQ-IM-1

Description:	A user would have an interface that allows him/her to add an unwanted item to the marketplace so others can acquire it for free.
Inputs:	Item Name Item Value Item Category More Details (optional) Lister Name Lister’s Name Lister’s Phone Number Lister’s Email Address Lister’s Location
Source:	The user adding the item will decide on the appropriate input for each section. For users with an account, personal information will be auto-filled but editable.
Outputs:	The user will receive feedback on the success or otherwise of the addition of a new item
Destination:	The feedback is sent to the user’s interface The added item is sent to the web applications database
Action:	At the click of a button, a user should be able to add an item to the market and specify its name, value, category, and details as well as his/her personal name, phone number, email address and location through an intuitive interface.
Pre-condition:	The user’s computer is connected to the Internet
Post-condition:	The new item can be viewed by other users of the application and edited solely by its owner. A text message containing details of the added item is sent from the web application to the best suggested recipient.
Side effects:	None

REQ-IM-2: A user should be able to update an item they have added to the market

Table 2.2: Table of Requirements for REQ-IM-2

Description:	Once an item has been added, a user should be able to conveniently edit the item’s details displayed on the online marketplace
Inputs:	The detail(s) to be updated
Source:	The user will select the detail(s) to be modified
Outputs:	The user will receive feedback on the success or failure of the item update.
Destination:	The feedback will be viewed on the user’s interface The updated detail(s) is/are sent to the web application’s database for the changes to take place on the web application

Action:	At the press of a button, a user should be able to view unwanted items they have added and modify any of the details as appropriate
Pre-condition:	The user needs to be able to view items they have added to the marketplace
Post-condition:	The online marketplace now displays the new details of the item
Side effects:	None

REQ-IM-3: A user should be able to delete an item they have added to the market

Table 2.3: Table of Requirements for REQ-IM-3

Description:	After an advertised item has exchanged hands, the previous owner should be able to remove this item from the online market, so as not to continue to receive requests for the item. Users without an account will have to verify their ownership of the item before being authorized to remove it from the market.
Inputs:	For users without an account, special code sent to item owners email address and phone number.
Source:	The user decides the item to be deleted
Outputs:	Notification feedback on the success or failure of the deletion
Destination:	The feedback is viewed on the applications graphical user interface. Identification details of the deleted item is sent to the application's database to take effect on the item's display page
Action:	For users with an account, at the click of a button, and subsequent confirmation of the action, a user should be able to delete an item they have added to the online marketplace. For users without an account, at the press of a button, and input of special code, a user should be able to conveniently delete an item they added previously
Pre-condition:	A user needs to be able to view an item they have added to the market
Post-condition:	The item is no longer displayed on the online marketplace
Side effects:	None

REQ-IM-4: A user should be able to share the details of an item on social media

Table 2.4: Table of Requirements for REQ-IM-4

Description:	A user should be able to share information of an item on the online marketplace with their connections on social media.
Inputs:	None

Source:	The user decides which item's details they want to share on social media. The message however is composed by the application's source code
Outputs:	A pre-composed message of the item's details is displayed on the current user's selected social media account
Destination:	Details of the item and a link to the item on the application will be posted on the user's selected social media account
Action:	At the click of a button, a user should be able to share details of an item on the marketplace through their social media accounts.
Pre-condition:	There must be items added to the online marketplace The user must have a Facebook, Twitter, or Google+ account
Post-condition:	The user has a message on their social media account advertising the item
Side effects:	None

2.3.2 Item Display Management

This section describes the item display management requirement and provides use cases and a scenario. Detailed functional requirements are then listed and discussed.

2.3.2.1 Description and Priority

This feature allows the user to manage his/her view of the online market. It is of high priority as it gives users control over what they view on the online market. A user should be able to view items added to the market by themselves and other users which is a core function of this web application. User should have options to filter their view of the market based on certain parameters as well as search for items based on certain parameters. The use case diagram in Figure 2.3 describes a user's interactions with the web application concerning this requirement.

2.3.2.2 Use Case: Item Display Management

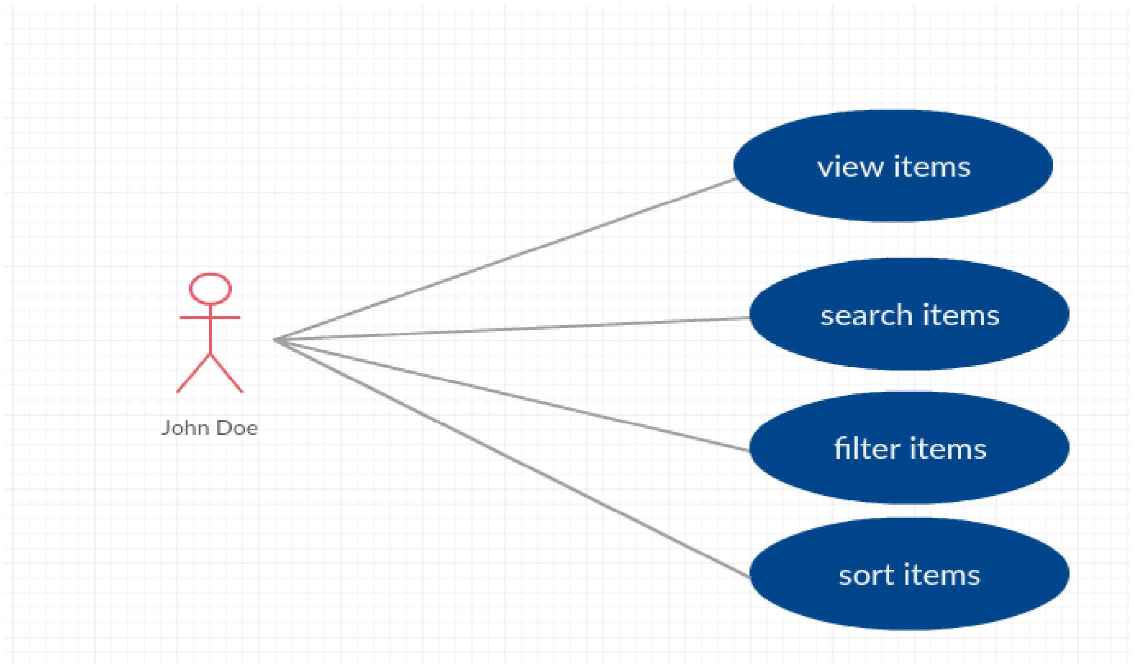


Figure 2.4: Use Case Diagram for Online Marketplace Item Display

2.3.2.3 Requirement Scenario: Item Display Management

Mr. A, a teenager preparing for a national external examination in a year, is in dire need of textbooks for a number of courses. After coming to the realization that people who have taken the examination before him will have little to no use for these textbooks, he decides to visit the web application and search for the needed textbooks.

2.3.2.4 Functional Requirements

REQ-IDM-1: A user should be able to view all unwanted but reusable items on the market

Table 2.5: Table of Requirements for REQ-IDM-1

Description:	A user should be able to view all the items that have been added to the online marketplace
Inputs:	None
Source:	N/A

Outputs:	The user views all the items advertised on the online marketplace
Destination:	The items will be displayed on the user's graphical user interface
Action:	At the press of a button, a user should be directed to a page with all the items currently available on the online market
Pre-condition:	The online marketplace must have had items added to it
Post-condition:	None
Side effects:	None

REQ-IDM-2: A user should be able to search for an item on the market

Table 2.6: Table of Requirements for REQ-IDM-2

Description:	A user should be able to search for a specific item on the marketplace
Inputs:	The name of the item the user is searching for
Source:	The user searching for an unwanted but reusable item
Outputs:	If successful, a view of items in the online marketplace that match the search keyword If unsuccessful, a graphical notification stating the unavailability of the item on the market
Destination:	The feedback, successful or unsuccessful, is displayed on the graphical user interface of the web application
Action:	At the click of a button, a user should be able to search for an item on the online marketplace by keyword
Pre-condition:	Items should have been added to the online marketplace
Post-condition:	The web applications graphical user interface should display items that match the stated keyword
Side effects:	None

REQ-IDM-3: A user should be able to filter items displayed on the market

Table 2.7: Table of Requirements for REQ-IDM-3

Description:	A user should be able to filter their view of the online marketplace
Inputs:	Lister Location Item Category
Source:	The user selects the filter parameters (location and category) from a dropdown list
Outputs:	View of items that match the parameters given by the user
Destination:	The items that fit the filter parameters are displayed on the application's web

	application
Action:	At the selection or click of a filter parameter, a user should be able to view items that fit a given filter characteristics only
Pre-condition:	Items should have been added to the online marketplace
Post-condition:	None
Side effects:	None

REQ-IDM-4: A user should be able to sort items displayed on the market

Table 2.8: Table of Requirements for REQ-IDM-4

Description:	Once items in the online market are displayed, a user should be able to sort his/her view based on certain parameters
Inputs:	Parameter for sorting (most recent, least recent, highest value, least value)
Source:	The user selects the sorting parameter from a dropdown list
Outputs:	View of items according to the sorting parameter
Destination:	The items, rearranged according to the sorting parameter, are displayed on the web application’s user interface
Action:	At the selection of an item from a dropdown list, a user should be able to sort items on the online market according to certain parameters
Pre-condition:	The online marketplace must have had items added to it
Post-condition:	None
Side effects:	None

2.3.3 Auxiliary Services

This section describes the requirement for Auxiliary Services and provides use cases and a scenario. Detailed functional requirements are then listed and discussed.

2.3.3.1 Description and Priority

This requirement consists of all the services aimed at supporting the primary functions of the web application. This feature also further allows users to customize their experience of the web application per their preferences. The use case diagram in Figure 2.2 describes a user’s interactions with the web application concerning this requirement.

2.3.3.2 Use Case: Auxiliary Services

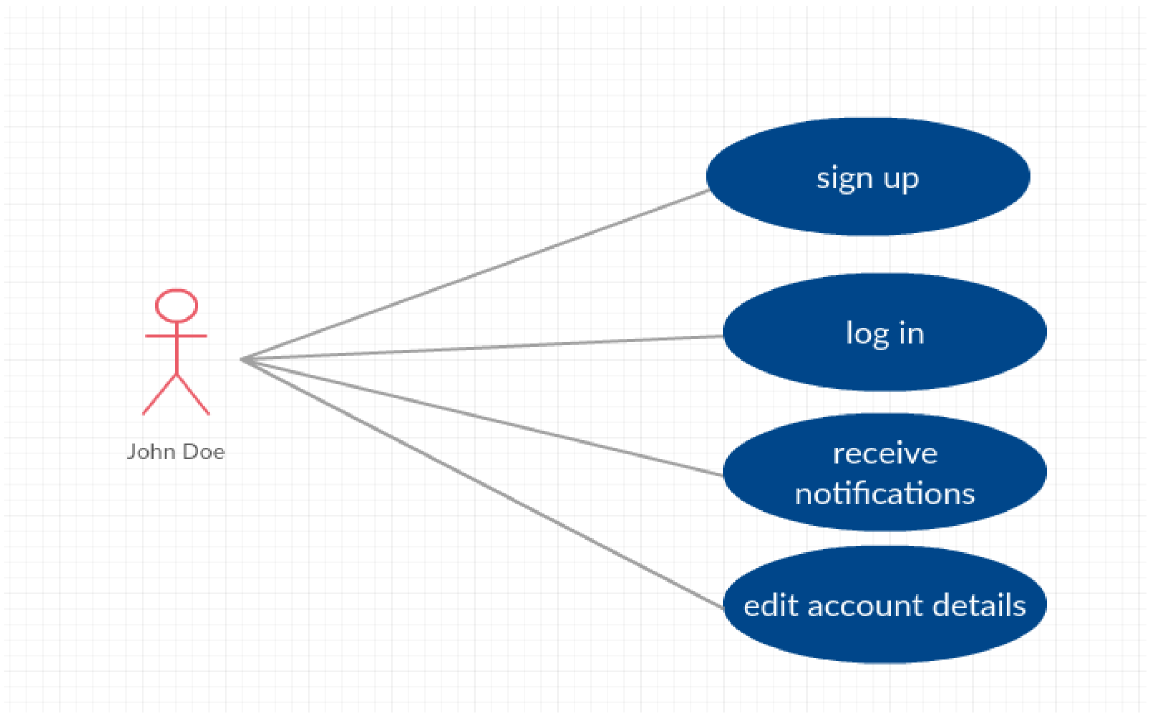


Figure 2.5: Use Case Diagram for Auxiliary Services

2.3.3.3 Functional Requirements

REQ-AS-1: The user should be able to sign up to use the web application

Table 2.9: Table of Requirements for REQ-AS-1

Description:	A user should be able to create an account on the web application
Inputs:	Name Email Address Phone Number Location Password
Source:	The user wanting to create an account will provide the required information for creating an account
Outputs:	Notification on the success or otherwise of the account creation
Destination:	The feedback notification is displayed to the user on the application's graphical user interface. The user's information is added to the web applications underlying database

Action:	After filling a form and at the click of a button, a user should be able to access features of the web application exclusive to users with an account
Pre-condition:	None
Post-condition:	The user can now log into the web application anywhere and at any time with their email address and password
Side effects:	None

REQ-AS-2: The user should be able to log in to use the web application

Table 2.10: Table of Requirements for REQ-AS-2

Description:	A user of the web application should be able to use the application with their credentials, credibility and authenticity established
Inputs:	Email Address Password
Source:	The user wanting to log into the web application will provide the necessary credentials
Outputs:	None
Destination:	The web application interface redirects to a personalized dashboard for the user
Action:	At the click of a button, a user should be able to authenticate his identity and access features exclusive to users with an account
Pre-condition:	The user should have created an account on the web application
Post-condition:	A session is started for the user containing all the details of the user
Side effects:	None

REQ-AS-3: The best suggested recipient of an item should be notified whenever items of their interest are added to the market

Table 2.11: Table of Requirements for REQ-AS-3

Description:	After creating an account, a user can select categories of items for which they want to be notified whenever there are new additions. Whenever there is a new addition belonging to this category, the web application should send a text message containing details of the item to the best suggested recipient as determined by the web application's algorithm
Inputs:	Addition of an item that suits a user's interest
Source:	Application generated based on the occurrence

Outputs:	An SMS message and email containing details of the newly added item is sent to a best suggested recipient immediately the item is added
Destination:	SMS and Email
Action:	The application can connect to external services that allow the sending of SMS's and Emails from within the web application whenever a new item is added that suits another user's interests. The interested user is notified accordingly
Pre-condition:	An unwanted but reusable item that suits a user's stated interest is added to the online market
Post-condition:	None
Side effects:	None

REQ-AS-4: The user should be able to modify account details

Table 2.12: Table of Requirements for REQ-AS-4

Description:	After creating an account, a user should be able to modify account details easily as and when the need arises
Inputs:	The account information to be modified (name, location, phone number, email address)
Source:	The user would provide the details needed to reconfigure his/her account
Outputs:	Feedback on the success or otherwise of the account details modification
Destination:	The feedback is displayed on the user's interface The users changed details is sent to the web application's database
Action:	After inputting modification details and at the click of a button, a user should be to modify details pertaining to his or herself
Pre-condition:	The user should already have created an account on the web application
Post-condition:	The web application now employs the new details of the user where appropriate
Side effects:	None

2.4 External Interface Requirements

This section discusses the web applications interfaces. These comprise graphical user interfaces, software interfaces and communication interfaces

2.4.1 Graphical User Interfaces

The web application would need to present a graphical user interface (GUI) that allows a user to interact with the services provided by the application's internal source code. When the user navigates to the application's homepage, he/she has the option to add an item to the market, view items already added to the market, create an account on the application or login to the application if the user already has an account. If the user does enter their login details, they will be privy to an additional three interfaces where they can locate items on Google Maps, view items they have added to their favourites, as well as an interface that allows the user to manage all items they have listed on the online market easily. Most of the interfaces will have footers which will contain quick links and contact information. The web application also makes intensive use of z-buffering through the Materialize framework's z-depth property. The Materialize framework's z-depth property is used to convey a feeling of 3D for certain forms and buttons on the web application.

- **Add to Market GUI:** With this interface, users will be able to input details about an unwanted item as well as their personal contact details. The inputs will make use of colour coding and restrictive input to enable users recognize faulty inputs before they click the 'add' button. With colour coding, an input will be underlined green if valid and red if invalid. The input would also restrict certain input as appropriate. As an example, the input space for the estimated value of an item will accept numerical input only.
- **View Market GUI:** This interface is the core of the application as it would allow users to view all unwanted but reusable items that have been added to the online market. The interface will allow users to filter what they see by location and item type. Users will also be able to sort the online market based on certain parameters. The interface will also have

a search bar at the top of the page to allow users search for an item by keyword. The page will also have to be paginated to cater for scalability.

- **Item Details GUI:** This interface gives users a larger view of the item's image, details of the listed item, and contact details of the lister. The interface also allows owners of an item to edit or delete the item on the online market if they so wish. The interface also has options for users to share details of the item on Facebook, Twitter and Google+.
- **Locate Items on Map GUI:** This interface will allow users with an account on the web application to view listed items near their locality on Google Maps. Users will also be able to search for possible safe places in a specified locality where unwanted items can exchange hands. The interface will also allow users to filter the items they see on the map per the items category or type.
- **View Favourites GUI:** Here, users with an account on the web application can see all the items they have favourited in the past.
- **Manage Listings GUI:** This interface will serve as a repository for all the items that a user has added to the online market. The interface also gives the user options to edit or delete their item(s) from the online marketplace.
- **Login GUI:** This interface serves as a portal into the web application. Users will have the option to log in through the applications portal, or use an existing social media account on Facebook or Google.
- **Signup GUI:** Here again, users will be able to sign up to use the application by providing certain details about themselves. The interface will make use of colour coding for the inputs – an input will be underlined green if valid and red if invalid.

- **Account Settings GUI:** This interface will allow users to set and customize their preferences.
- **Edit User Details GUI:** With this interface, users with an account on the web application will have the opportunity to alter their personal details as and when the need arises.

2.4.2 Software Interfaces

The web application will primarily interface with XAMPP – a free and open source cross-platform web server solution stack package developed by Apache Friends (Wikipedia, 2017). The web application is built on the components provided by XAMPP: The Apache HTTP server provided by XAMPP is used to serve web pages on the local host, MySQL serves as the database management system for the application, and PHP is the scripting language used for the application's server-side web development (Wikipedia, 2017).

The application will also interface with an SMS Gateway, an android application that allows web developers to send and receive messages programmatically through their mobile phone using an Application Programming Interface (API) (SMS Gateway, n.d). The application will also interface with Facebook and Google Authentication API's to allow users authenticate their identity without having to create an account on the web application.

2.4.3 Communication Interface

The protocol for communication that will be employed by the web platform is the Hypertext Transfer Protocol (HTTP). HTTP is the underlying protocol employed by the World Wide Web (Beal, 2017). The protocol defines how messages are formatted and transmitted across the web and also describes the actions of web servers and browsers in response to directives (Beal, 2017).

2.5 Nonfunctional Requirements

There are 7 non-functional requirements for this web platform. These requirements are listed and described below:

2.5.1 Scalability

The web platform must not crush or fail should it be used by a large number of people within a short time frame.

2.5.2 Standardization

Development of the web platform must follow custom development practices to ensure that others can build on it and expand it.

2.5.3 Design Consistency

The web platform should exhibit a consistent user experience across multiple web pages and within multiple web browsers. Colours and font schemes should also be consistent

2.5.4 Security

The web platform should not allow exfiltration of user's personal information, unless authorized by the user.

2.5.5 Documentation

The web platform must have extensive documentation in order to aid user navigation and understanding of how the platform works.

2.5.6 Platform Compatibility

The web platform should be compatible with a host of web browsers, particularly the most used such as Google Chrome, Mozilla Firefox, and Internet Explorer.

2.5.7 Privacy

The platform must not gather, use, or disclose client’s data without the necessary authorization (Wikipedia, 2017). The platform must have information on how and why data is gathered and the measures set in place to keep the data private (Engel, 2016).

3. Chapter 3: Architecture and Design

3.1 Introduction to Architecture and Design

This section on architecture and design introduces the chapter with the purpose and a brief summary.

3.1.1 Purpose

This chapter describes how the web platform described in the requirements section is to be designed and implemented. The chapter presents an all-encompassing view of the web platform architecture to enable a developer build the platform according to the requirements specified. The chapter will include an in-depth view of the Data Design used by the web platform as well as a picture and description of the platform's graphical user interfaces.

3.1.2 Summary and Overview of Chapter

This chapter will put into perspective the details of the web platform architecture with the aid of a deployment diagram, sequence diagrams, and user interfaces. The document will also discuss design considerations and alternative architectures that were considered but not deployed.

3.2 Design Considerations

This section details on the architecture used by the web platform as well as alternative designs that were considered.

3.2.1 Web Platform Architecture

Figure 3.1 is a block diagram showing the interconnections between the entities that constitute the web platform. The diagram shows the various software components and their ensuing hardware components. The block diagram also shows the interfaces between the entities that make the platform and specifies the protocol or API used in communication.

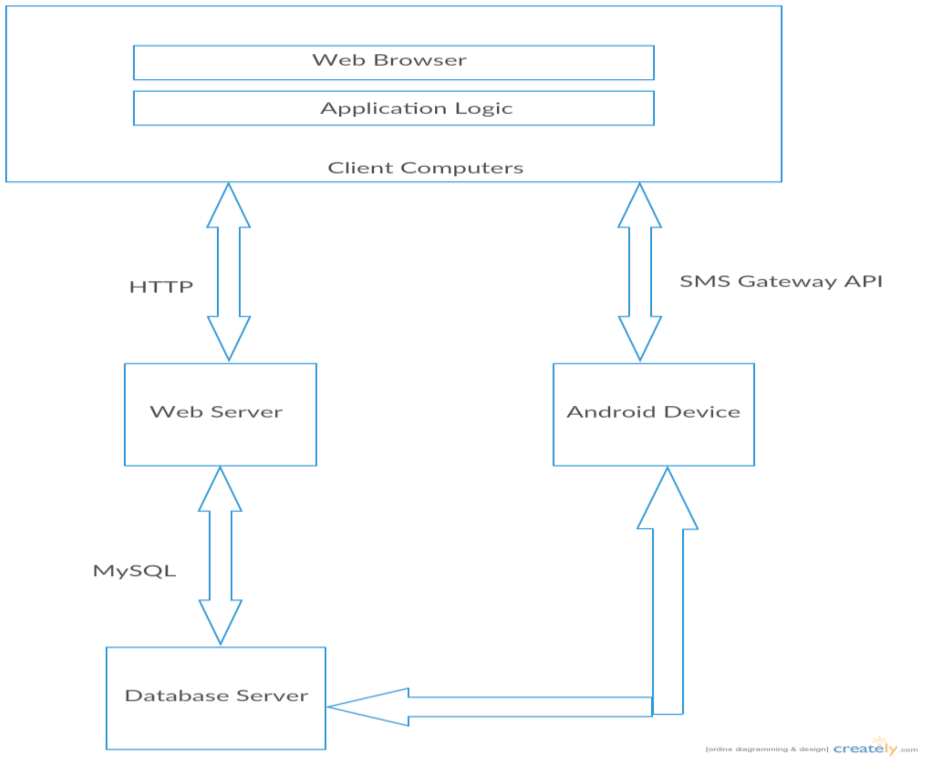


Figure 3.1: A Deployment Block Diagram Showing the Components of the Web Platform

3.2.2 Component Decomposition: Sequence Diagram

Figure 3.2 shows a sequence diagram detailing the interactions between an end user and the software components of the web platform. The diagram shows the chain of activities that enable a user to add, edit and delete an item on the online marketplace. The sequence diagram traces requests from initiation on the client side to the backend of the platform, also detailing how corresponding database query results are returned to the client. The sequence diagram shows how the platform makes significant use of Asynchronous JavaScript and XML (AJAX) to send data to a server in the background, receive data from the server, request data from the server and most crucial to performance, update a web page without reloading the entire page (W3Schools, 2017). The diagram also depicts the Object-Oriented Programming (OOP) approach adopted in developing this platform. To perform an activity, the web platform initializes an appropriate object which is then used to perform the requisite database queries and store query results. The object's functions are used for queries whereas query results are stored in member variables of the object's class. For all interactions between the client computer and the web platform, the user receives a notification from the application on the user interface as to the success or failure of the operation.

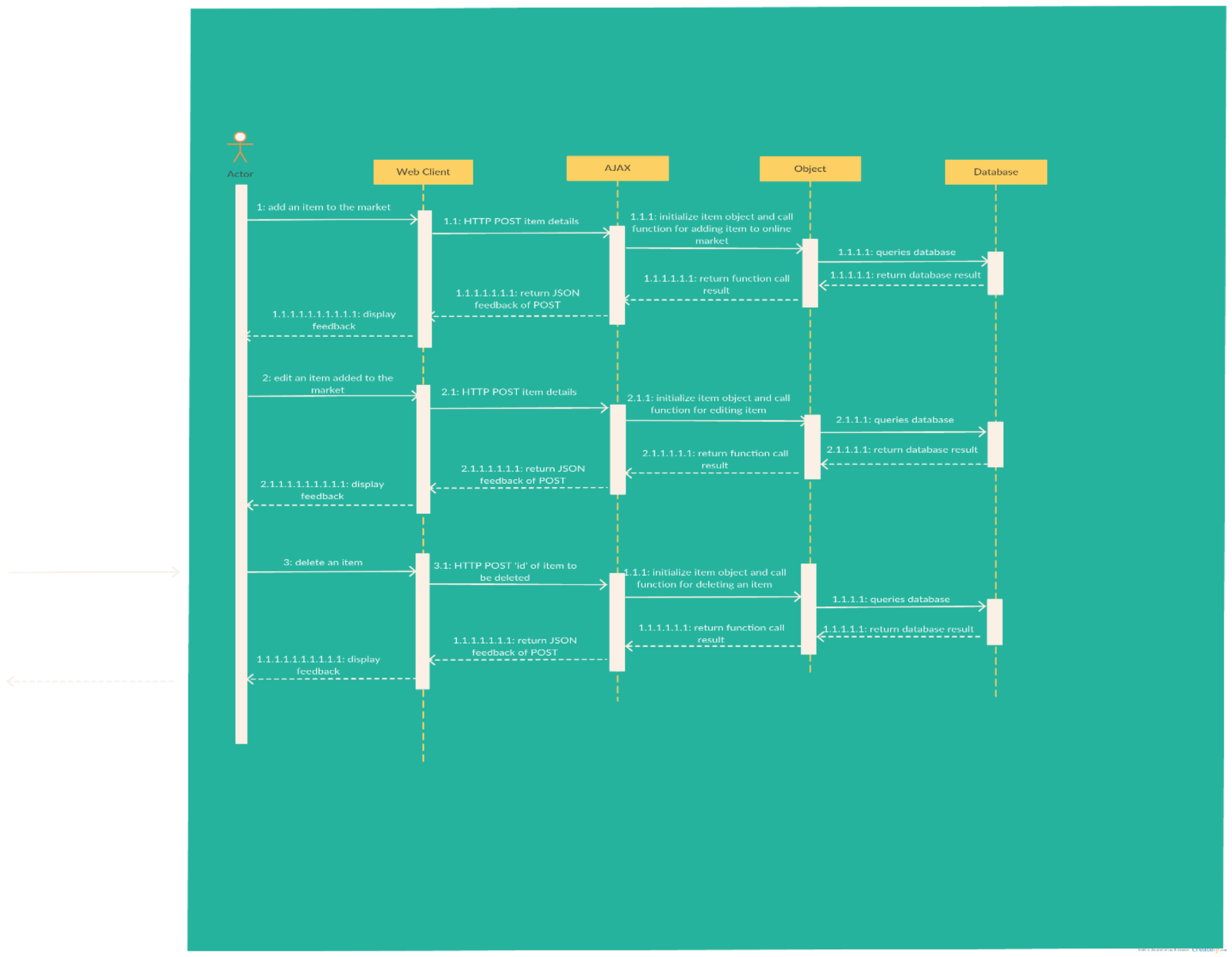

Actor
Web Client
AJAX
Object
Database
1: add an item to the market
1.1: HTTP POST item details
1.1.1: initialize item object and call function for adding item to online market
1.1.1.1: queries database
1.1.1.1.1: return database result
1.1.1.1.1.1: return function call result
1.1.1.1.1.1.1: return JSON feedback of POST
1.1.1.1.1.1.1.1.1.1: display feedback
2: edit an item added to the market
2.1: HTTP POST item details
2.1.1: initialize item object and call function for editing item
2.1.1.1: queries database
2.1.1.1.1: return database result
2.1.1.1.1.1: return function call result
2.1.1.1.1.1.1: return JSON feedback of POST
2.1.1.1.1.1.1.1.1.1: display feedback
3: delete an item
3.1: HTTP POST 'id' of item to be deleted
1.1.1: initialize item object and call function for deleting an item
1.1.1.1: queries database
1.1.1.1.1: return database result
1.1.1.1.1.1: return function call result
1.1.1.1.1.1.1: return JSON feedback of POST
1.1.1.1.1.1.1.1.1.1: display feedback

Figure 3.2: A Sequence Diagram for Item Management

Figure 3.3 shows a sequence diagram describing the interactions between a user and the software components of the platform to manage the display of items on the web application's user interface.

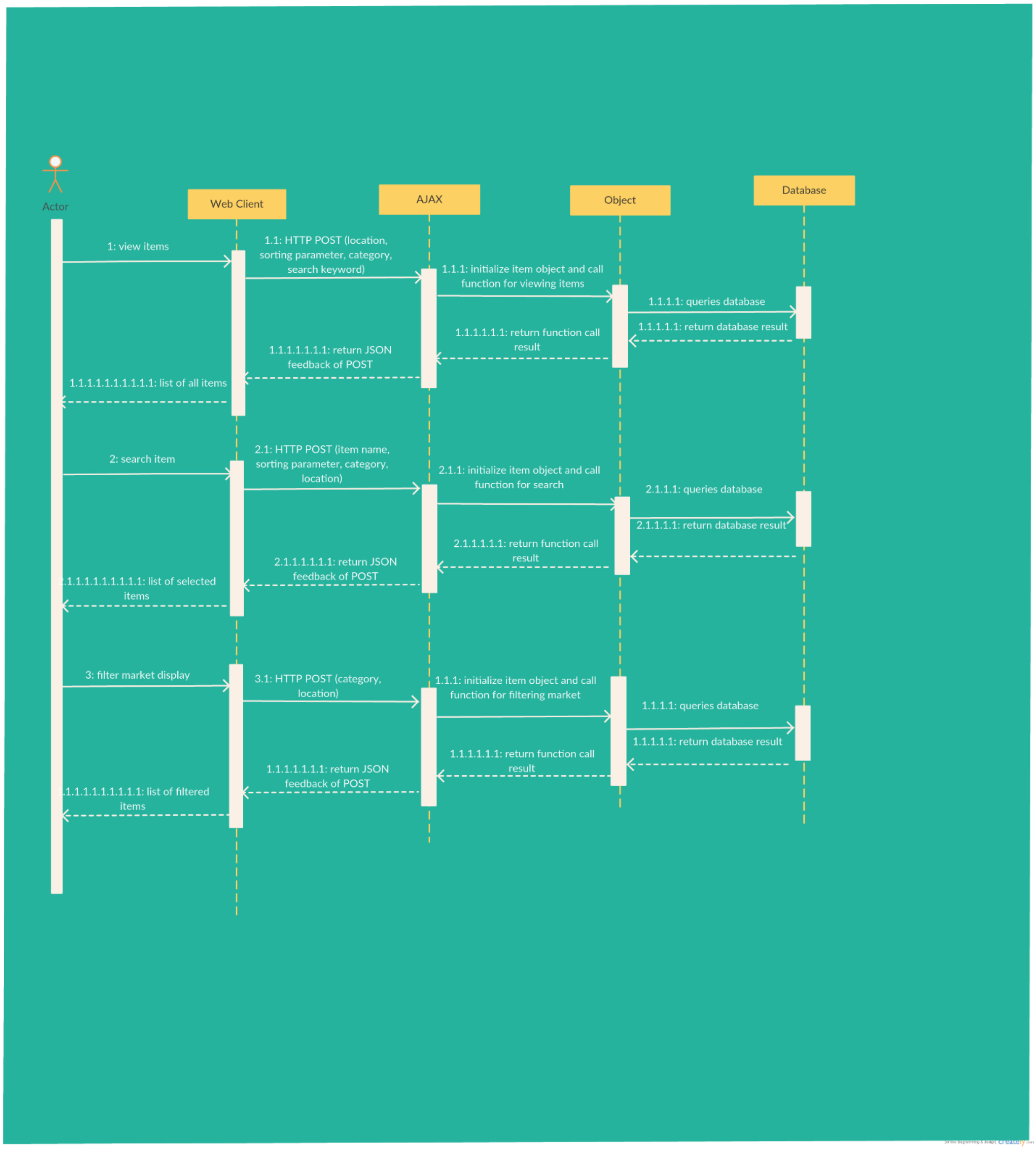

Actor
Web Client
AJAX
Object
Database
1: view items
1.1: HTTP POST (location, sorting parameter, category, search keyword)
1.1.1: initialize item object and call function for viewing items
1.1.1.1: queries database
1.1.1.1.1: return database result
1.1.1.1.1.1: return function call result
1.1.1.1.1.1.1: return JSON feedback of POST
1.1.1.1.1.1.1.1.1.1: list of all items
2: search item
2.1: HTTP POST (item name, sorting parameter, category, location)
2.1.1: initialize item object and call function for search
2.1.1.1: queries database
2.1.1.1.1: return database result
2.1.1.1.1.1: return function call result
2.1.1.1.1.1.1: return JSON feedback of POST
2.1.1.1.1.1.1.1.1.1: list of selected items
3: filter market display
3.1: HTTP POST (category, location)
1.1.1: initialize item object and call function for filtering market
1.1.1.1: queries database
1.1.1.1.1: return database result
1.1.1.1.1.1: return function call result
1.1.1.1.1.1.1: return JSON feedback of POST
1.1.1.1.1.1.1.1.1.1: list of filtered items

Figure 3.3: A Sequence Diagram for Item Display Management

Figure 3.4 is a sequence diagram depicting a user's interactions with the web platform while accessing the platform's auxiliary services.

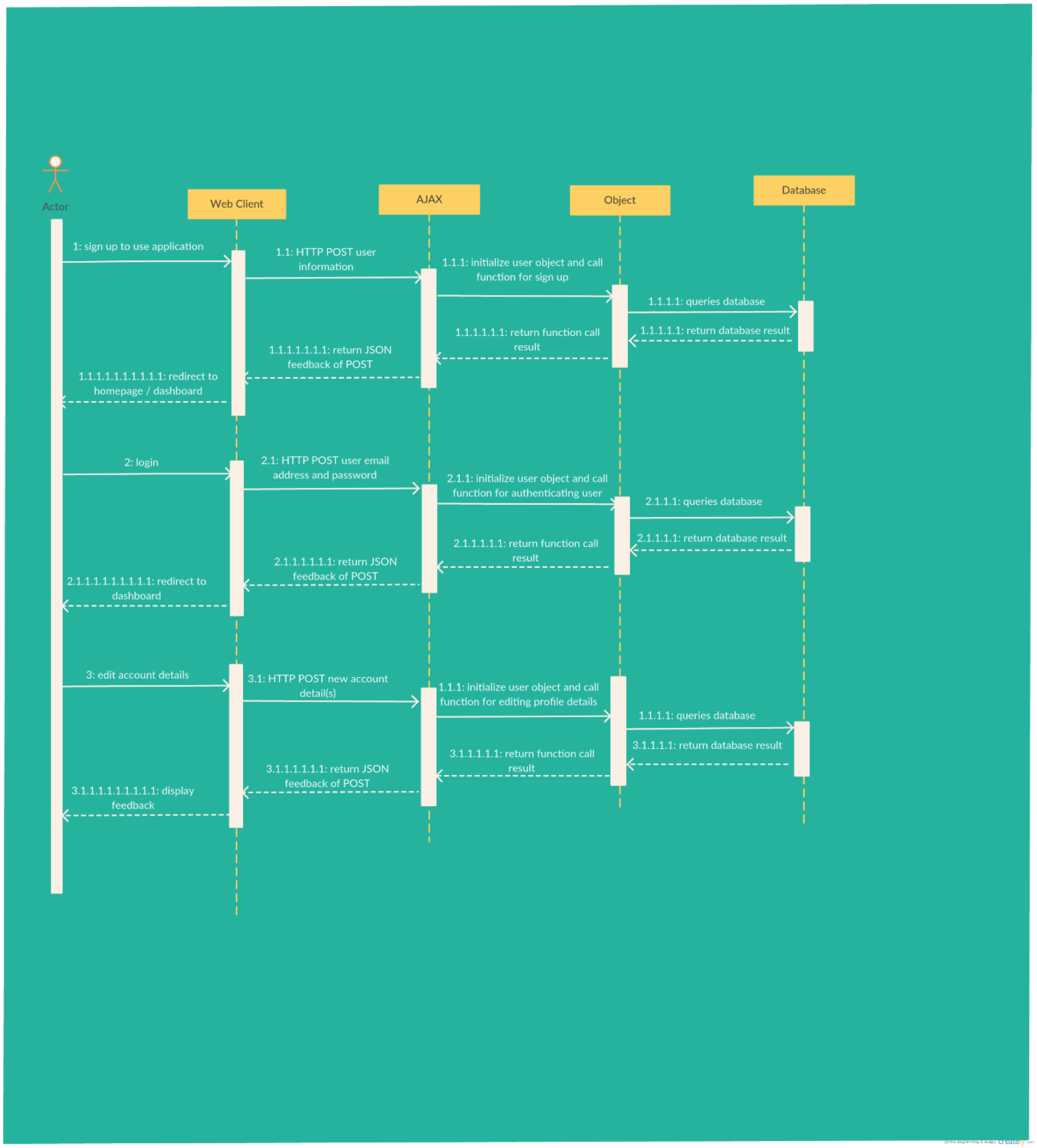

Actor
Web Client
AJAX
Object
Database
1: sign up to use application
1.1: HTTP POST user information
1.1.1: initialize user object and call function for sign up
1.1.1.1: queries database
1.1.1.1.1: return database result
1.1.1.1.1.1: return function call result
1.1.1.1.1.1.1: return JSON feedback of POST
1.1.1.1.1.1.1.1.1.1: redirect to homepage / dashboard
2: login
2.1: HTTP POST user email address and password
2.1.1: initialize user object and call function for authenticating user
2.1.1.1: queries database
2.1.1.1.1: return database result
2.1.1.1.1.1: return function call result
2.1.1.1.1.1.1: return JSON feedback of POST
2.1.1.1.1.1.1.1.1.1: redirect to dashboard
3: edit account details
3.1: HTTP POST new account detail(s)
1.1.1: initialize user object and call function for editing profile details
1.1.1.1: queries database
3.1.1.1.1: return database result
3.1.1.1.1.1: return function call result
3.1.1.1.1.1.1: return JSON feedback of POST
3.1.1.1.1.1.1.1.1.1: display feedback

Figure 3.4: A Sequence Diagram for Auxiliary Services

3.2.3 Architectural Alternatives

Figure 3.5 shows a sequence diagram detailing the workflow of an alternative architecture which satisfies the product functions using an android mobile application. The architecture in Figure 3.6 simplifies the process of adding an item to the market as compared to the web platform architecture. With this architecture, users can take an image of unwanted but reusable items they possess with their phone's in-built camera, fill in the item details as well as their location details, and then publish the item on the market with no pre-processing. To put the difference between the two architectures in perspective, the web platform requires that users have an image of the item uploaded onto their computer prior to engaging the web platform's item addition functionality whereas the mobile application allows users to walk through the process of adding an item – from capturing of the item image to addition of item details and submission of item to the online market – at once.

The display of items on the online market will also differ with this alternative architecture. With the android application, users can use their phone's in-built touch functionality to swipe through items added to the online market and request for an item at the touch of a button. The request functionality will be three-pronged – the application will give users the options of calling the item owner, sending him/her a text message from the native android application or sending an email using android's API. Here again, this architecture simplifies the process of requesting an item on the market since users can view and request items from one interface. With the web platform architecture, users ascertain the details of the item on the applications interface but cannot request for the item through that interface.

(insert sequence diagram)

Figure 3.5: A Sequence Diagram for Functions of Alternative Architecture

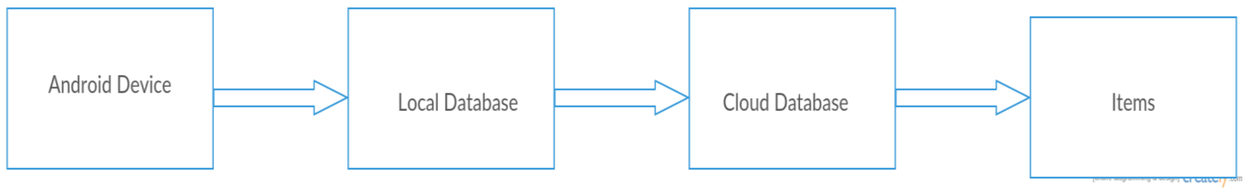


Figure 3.6: An Alternative Architectural Design for the Product

3.2.4 Design Rationale

On the surface, it would appear more expedient to use a mobile application. However, these architectures are not without their difficulties and pitfalls. Although simplifying key functions of the product, native application development would have significantly hindered the reachability of the service. Admittedly, with the inception of software services such as Phonegap, it is possible to write code once and deploy on multiple mobile operating systems. However, these technologies are also not without notable challenges. Phonegap requires users to develop using HTML5 and is unable to interpret PHP code, which is the most used coding language for server side scripting. It is also a widely common and accepted practice among developers to build mobile applications on top of existing web platforms, and not the other way around. After prolonged consideration, the architecture shown in Figure 3.1 was chosen to be implemented and deployed, with the alternative architecture considered for future works.

3.3 Data Design

This section describes how the web platform stores and organizes data.

3.3.1 Data Description

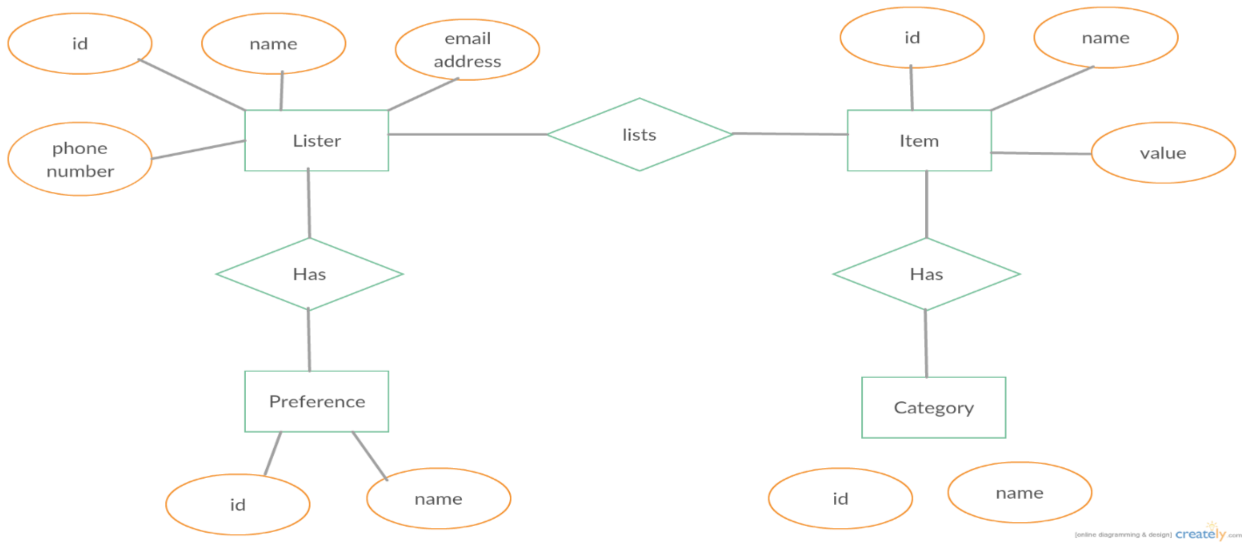


Figure 3.7: Entity Relational (ER) Diagram for the Web Platform

The web platform uses a relational database. The table columns have IDs which serve as primary keys and are necessary to maintain the uniqueness of database entries. The database structure also makes use of foreign keys to create links between tables and maintain referential integrity.

3.3.2 Data Dictionary

Table 3.1: Table of Database Fields and Descriptions

Document Name	**Field Name**	**Data Type**	**Allow Nulls**	**Field Description**
Lister	lister_id	Integer	No	This field uniquely identifies a registered user
	lister_name	Text	No	This field identifies a registered user by a recognizable name
	lister_email	Text	No	This field identifies a registered user by a recognizable email address
	lister_telephone	Text	No	This field identifies a registered user by a recognizable telephone

				number
	lister_location	Integer	No	This field describes the residence of the registered user
	lister_password	Text	No	This field contains the unique authentication and authorization key for a registered user
	lister_preference	Integer	Yes	This field contains a reference to an item category that interests the user
Item	item_id	Integer	No	This field uniquely identifies an item added to the online market
	item_image	Text	No	This field contains a reference to an image of the item uploaded to the server
	item_name	Text	No	This field identifies an item added to the online market by a recognizable name
	item_value	Double	Yes	This field contains an estimated current value of the item added to the market
	item_category	Integer	No	This field contains a reference to the category of an item added to the online marketplace
	item_details	Text	Yes	This field contains extra details concerning the unwanted but reusable item added to the online marketplace
	item_listed_time	Timestamp	No	This field describes the last time changes were made to the details of an item on the online market
	item_lister_id	Integer	Yes	This field contains a reference to the owner of the unwanted but reusable item added to the online marketplace
	item_lister_name	Text	Yes	This field contains the name of the owner of an item added to the online marketplace
	item_lister_phone	Text	Yes	This field contains the telephone number of the owner of an item added to the online marketplace
	item_lister_email	Text	Yes	This field contains the email

				address of the owner of an item added to the online marketplace
	item_lister_location	Integer	Yes	This field contains the residence of the owner of an item added to the online marketplace
Category	category_id	Integer	No	This field uniquely identifies an item category
	category_name	Text	No	This field identifies an item category by a recognizable name

3.4 Graphical User Interface Design

This section discusses the interfaces of the web application and the design considerations that were made.

3.4.1 Overview of User Interface Design

The user interface of the web platform will serve as the primary point of contact between users and the backend services provided by the web platform. The graphical user interface (GUI) of the application will allow registered and unregistered users alike to add items to the online market as well as view items already added to the market. Users without an account will also be able to sign up to access auxiliary services provided by the platform. These auxiliary services include locating items on Google Maps, managing listed items and adding items to a favourites list for future reference.

3.4.2 User Interface Design

This section documents the design of each module that make up the web platform

3.4.2.1 User Interface Design: Homepage

For the homepage, three interfaces were designed and assessed. Option A (Figure 3.8) has a horizontal navigation menu at the center of the page. The view is populated with five images, a large one placed on top of the centered navigation menu and four smaller images of equal size placed below the navigation menu. The idea behind this homepage design is to allow users visualize a select group of items added to the market on first visit in order to encourage them to add their unwanted but reusable items to the market as well. From the homepage, users can access all services provided by the platform except those exclusive to users with an account.

Option B (Figure 3.9) shows the second user interface design created and evaluated. This design takes a minimal approach as compared to Option A with a single background image, the website name, and brief tagline appearing on the homepage of the web application. This design also enables users to navigate to all available web pages except those exclusive to registered users.

Option C (Figure 3.10) shows the third design prototyped and assessed. The design takes motivation from popular search engines on the Internet. This design portrays the web platform as a search engine for unwanted items that can still be used by others. Clicking on the search icon takes the user to a view of the online market. Unlike the two previous designs considered, there is no specific menu item for viewing items on the market.

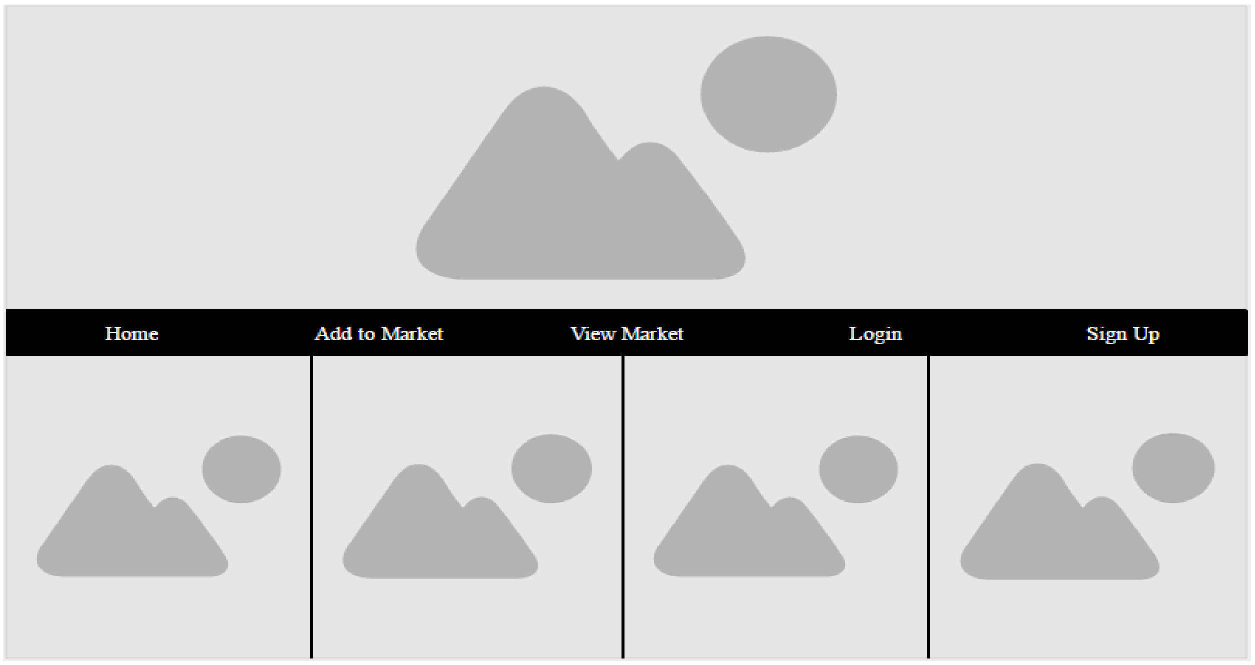


Figure 3.8: User Interface Design (Option A)

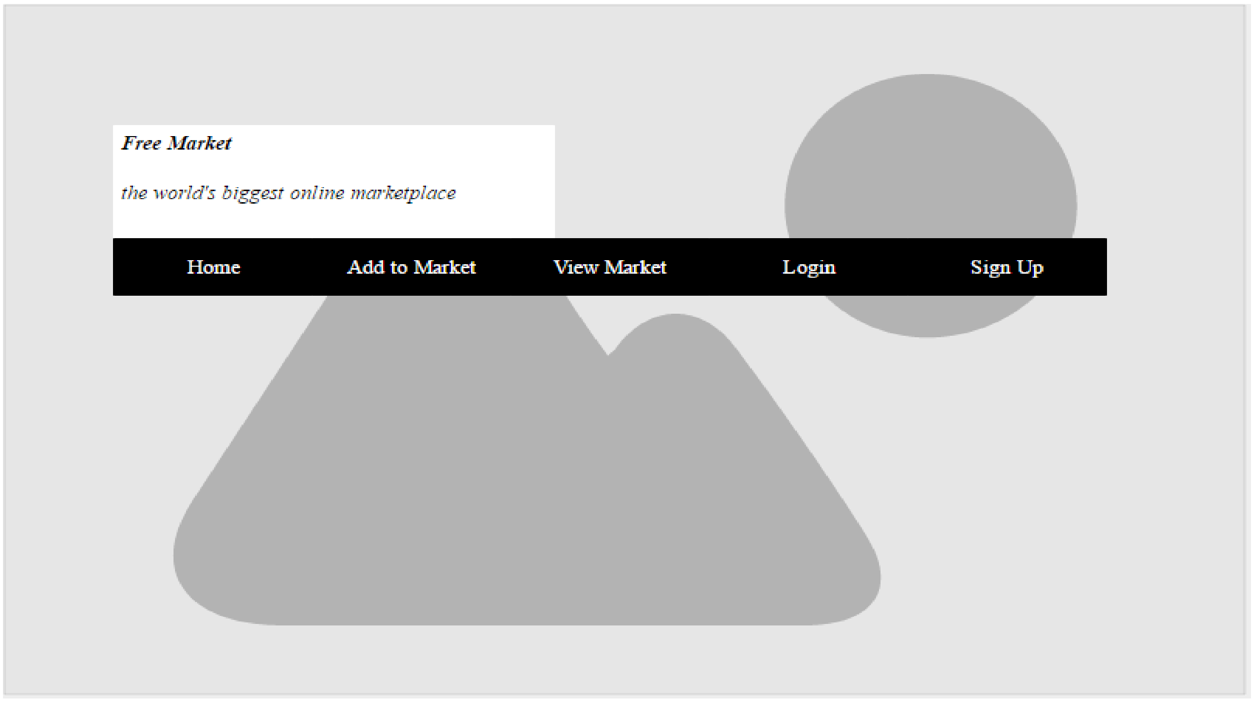


Figure 3.9: User Interface Design (Option B)

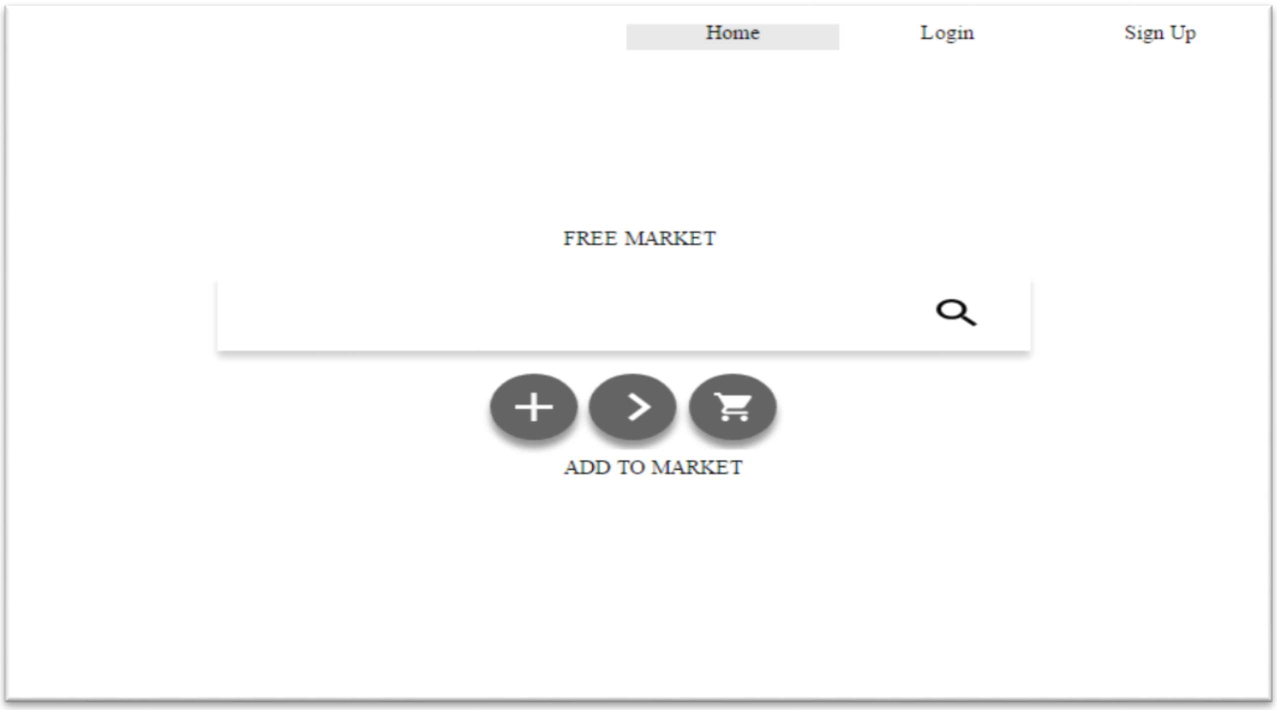


Figure 3.10: User Interface Design (Option C)

3.4.2.2 User Interface Design: Online Marketplace

Figure 3.11 (D1) shows a mockup design for the online marketplace. The interface has a search bar at the apex of the page to make provision for specificity in view of future scale. The design also makes provision for filters with the use of dropdowns positioned at the top of the web page. Details of the item (image and name) are displayed on cards with action buttons at the base of the card. The page has a footer where useful links can be placed.

On click of the description icon at the bottom of a card, the web page redirects to the interface shown in Figure 3.12 (D2). The user is made privy to details about one item – that which was clicked on. The interface has buttons that allow the user to edit or delete an item, in the event the item belongs to the current user. The user is also able to share details of the item with others on social media (Google+, Facebook and Twitter).

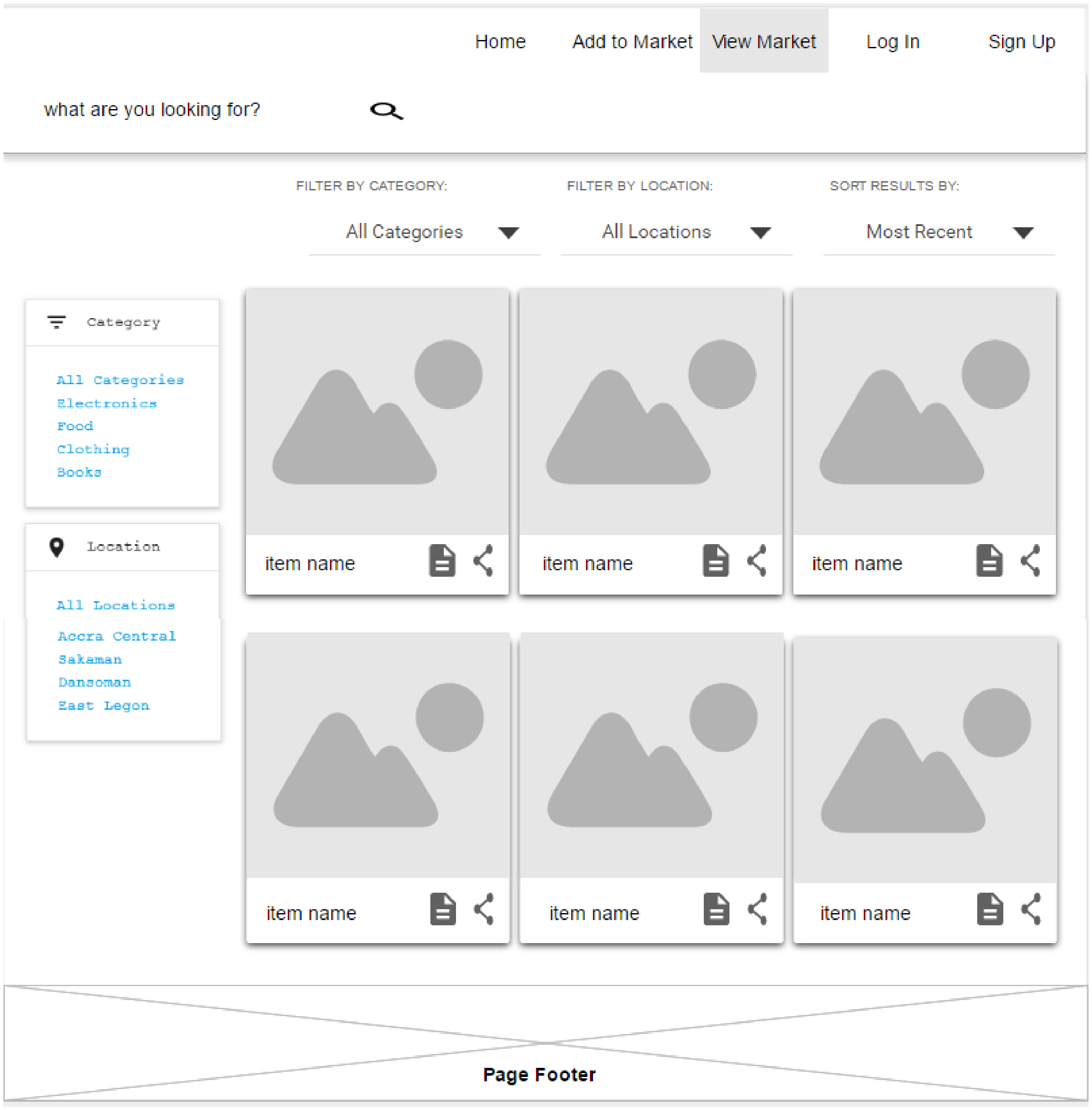


Figure 3.11: Mockup Design – Online Marketplace (D1)

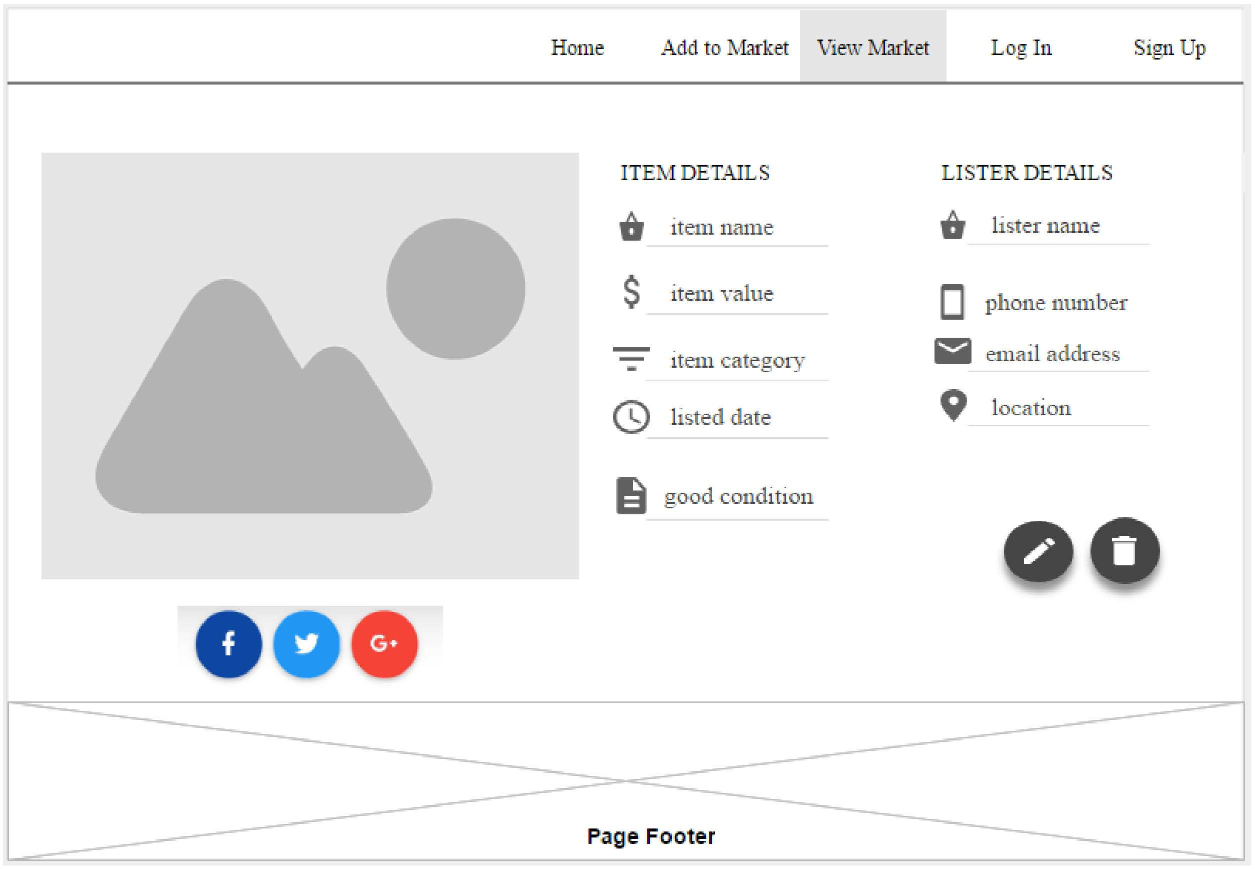


Figure 3.12: Mockup Design – Item Details (D2)

3.4.2.3 User Interface Design: Adding Item to Market

Figure 3.13 is a mockup design of the interface employed by users to add an item to the online marketplace. The interface places two forms side-by-side to avoid scrolling. This allows user to fill the details form with consummate ease. Material icons and text placeholders are used as visual and textual aids for the various input spaces. The interface also makes use of a footer where useful links and information can be placed.

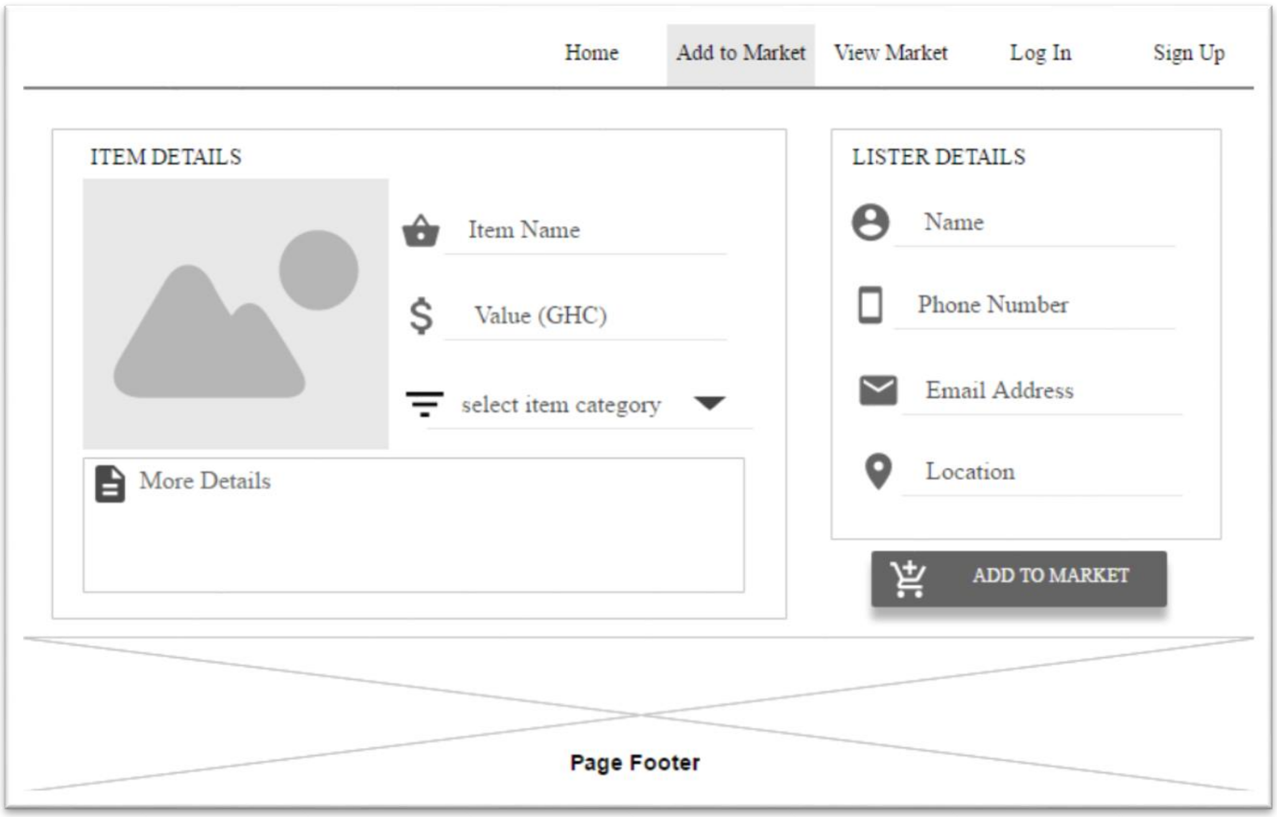


Figure 3.13: Mockup Design – Adding Item to Market

3.4.2.4 User Interface Design: Login

This interface serves as the portal into the application. Here, users will be authenticated and authorized to use auxiliary services provided by the web platform. Figure 3.14 (E1) shows the primary interface where users can decide to login using social media accounts or email address. Users without an account are also prompted at the bottom of the interface to create an account. Clicking the buttons to log in with social media accounts will lead to interfaces provided by the social media company whereas clicking on the 'Login with Email' button will lead to the interface shown in Figure 3.15 (E2). The follow up interface makes use of Material icons and text placeholders as visual aids for input spaces. Here again, users without an account ae prompted to create an account at the bottom of the page.

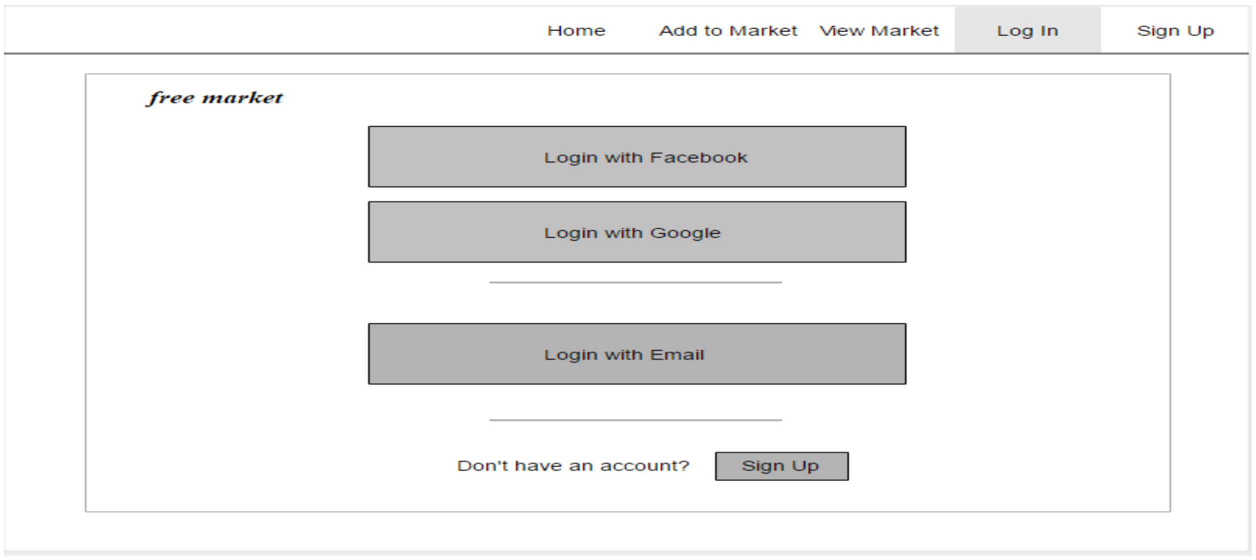


Figure 3.14: Mockup Design – Login (E1)

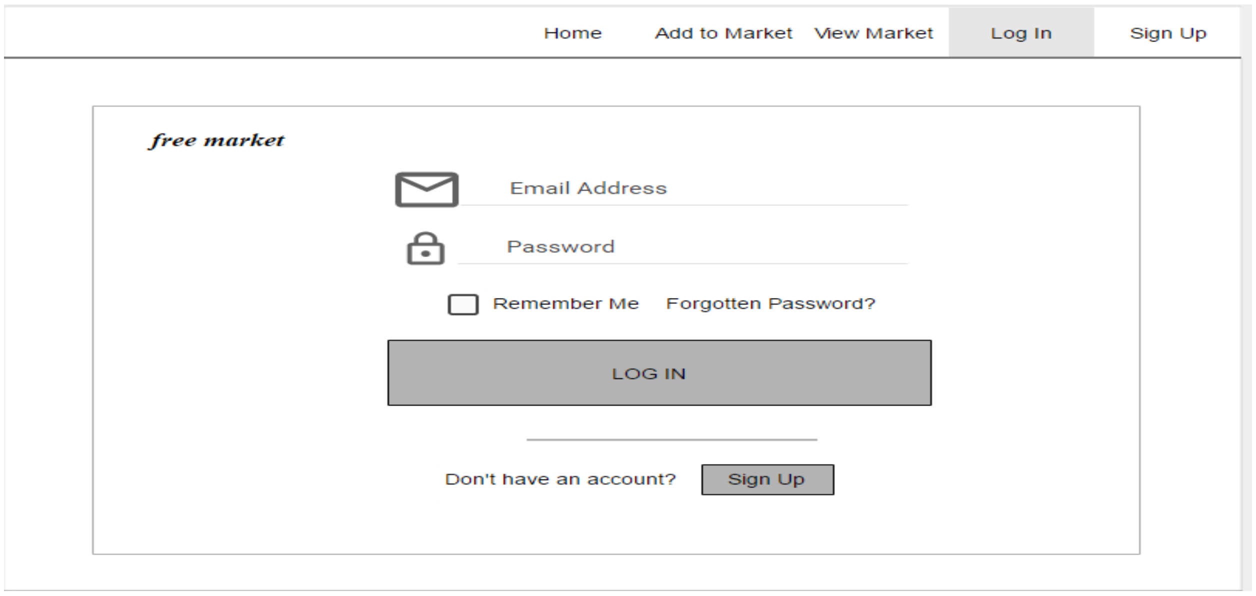


Figure 3.15: Mockup Design – Login (E2)

3.4.2.5 User Interface Design: Signup

Figure 3.16 (F1) depicts the interface used to create an account on the web application. This view allows users to create an account to authenticate their identity. The interface is consistent with that of the login web page. Users have the option to sign up with social media accounts or an email address. Users who already have an account are advised at the bottom of the page to log in. Clicking the buttons to sign up with social media accounts will lead to interfaces provided by the respective social media company whereas clicking on the 'Sign up with Email' button will lead to the interface shown in Figure 3.17 (F2).

The interface shown in Figure 3.17 (F2) contains a form where users can input details necessary to create an account to access services provided by the web platform through the web application. The interface makes use of Material icons and placeholders to assist users in filling the input spaces. Uses who have already created an account on the web application are directed to login at the bottom of the page.

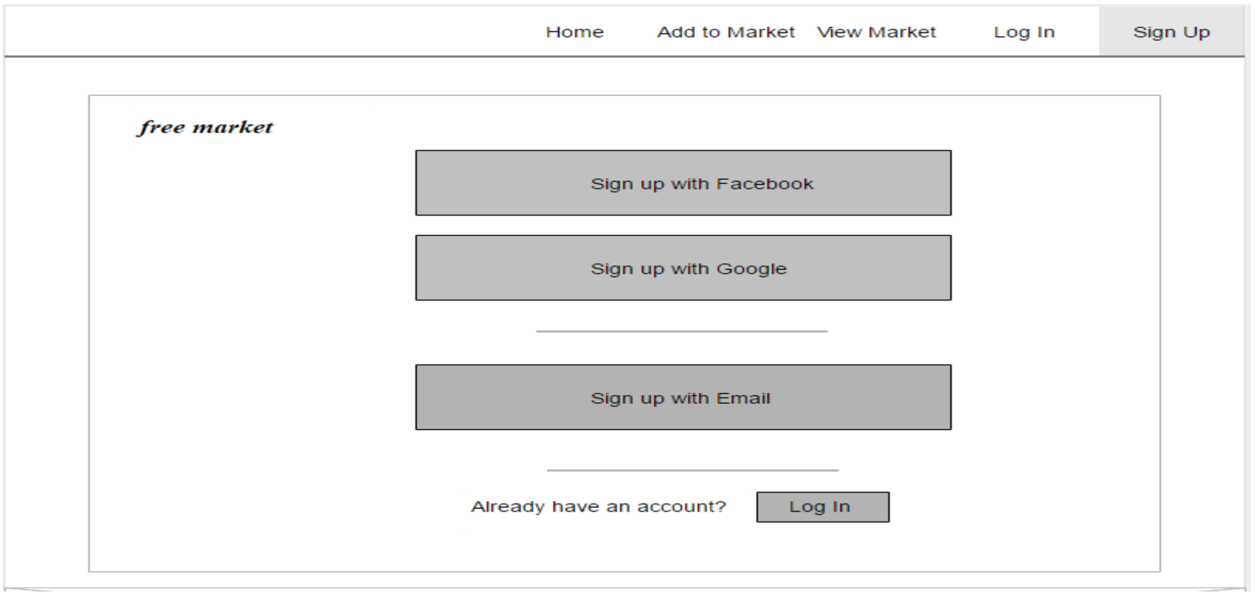


Figure 3.16: Mockup Design – Signup (F1)

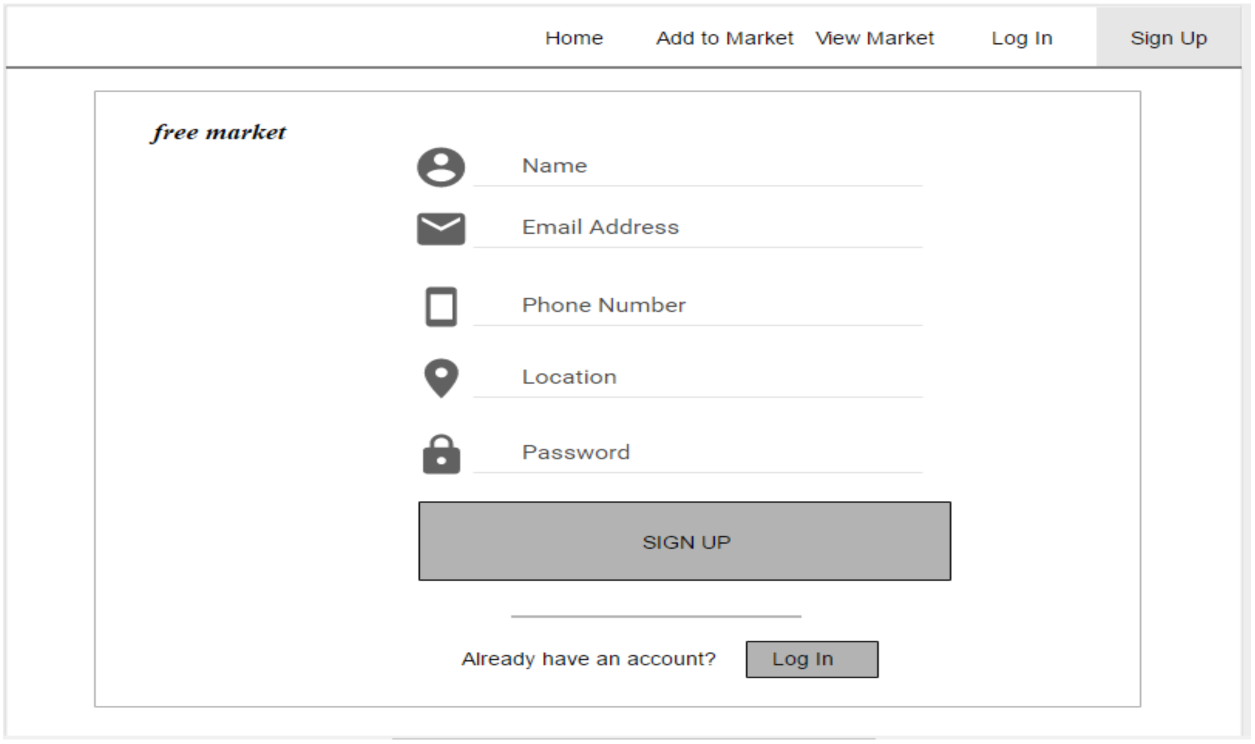


Figure 3.17: Mockup Design – Signup (F2)

3.4.2.6 User Interface Design: Dashboard

Figure 3.18 shows the interface design for a user's dashboard. After logging in, users are redirected to this page from which they can navigate to all the other available pages. A user image is placed at the top of the page to enable users personalize their dashboard view. Users also have their names displayed on the dashboard interface to further reinforce the personal feel of the view. Material icons are used together with descriptive words to label the buttons on the interface. The page has a footer which can contain useful links.

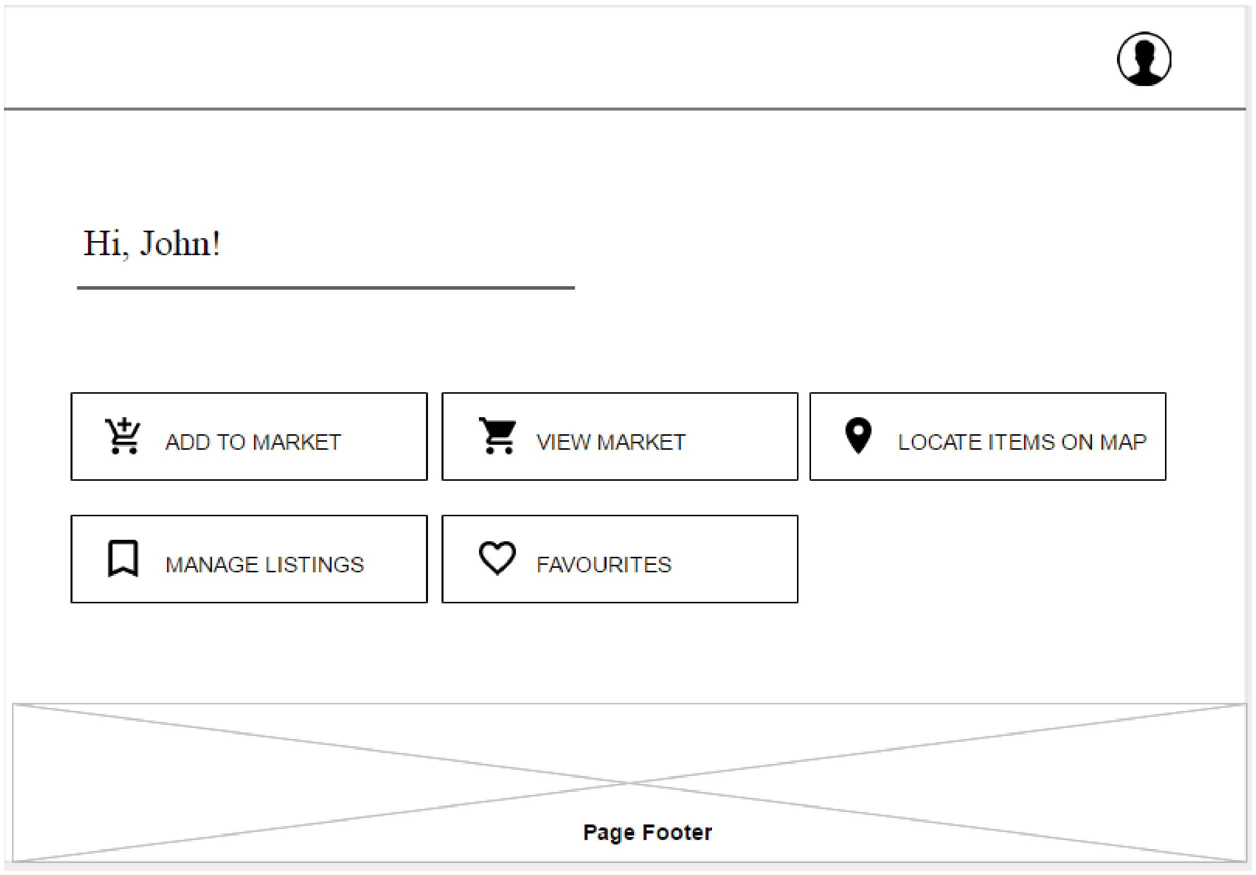


Figure 3.18: Mockup Design – Dashboard

3.4.2.7 User Interface Design: Locate Items on Map

This interface enables users to visualize unwanted but reusable items in their locality on a map embedded into the web application. Figure 3.19 shows a mockup design of the interface. A map is embedded in the application to allow users visualize items close to their current vicinity or viewport. The interface also has an accordion at the left side of the screen where users can filter the items displayed on the map by category. The page also has a footer where useful links can be placed for quick reference.

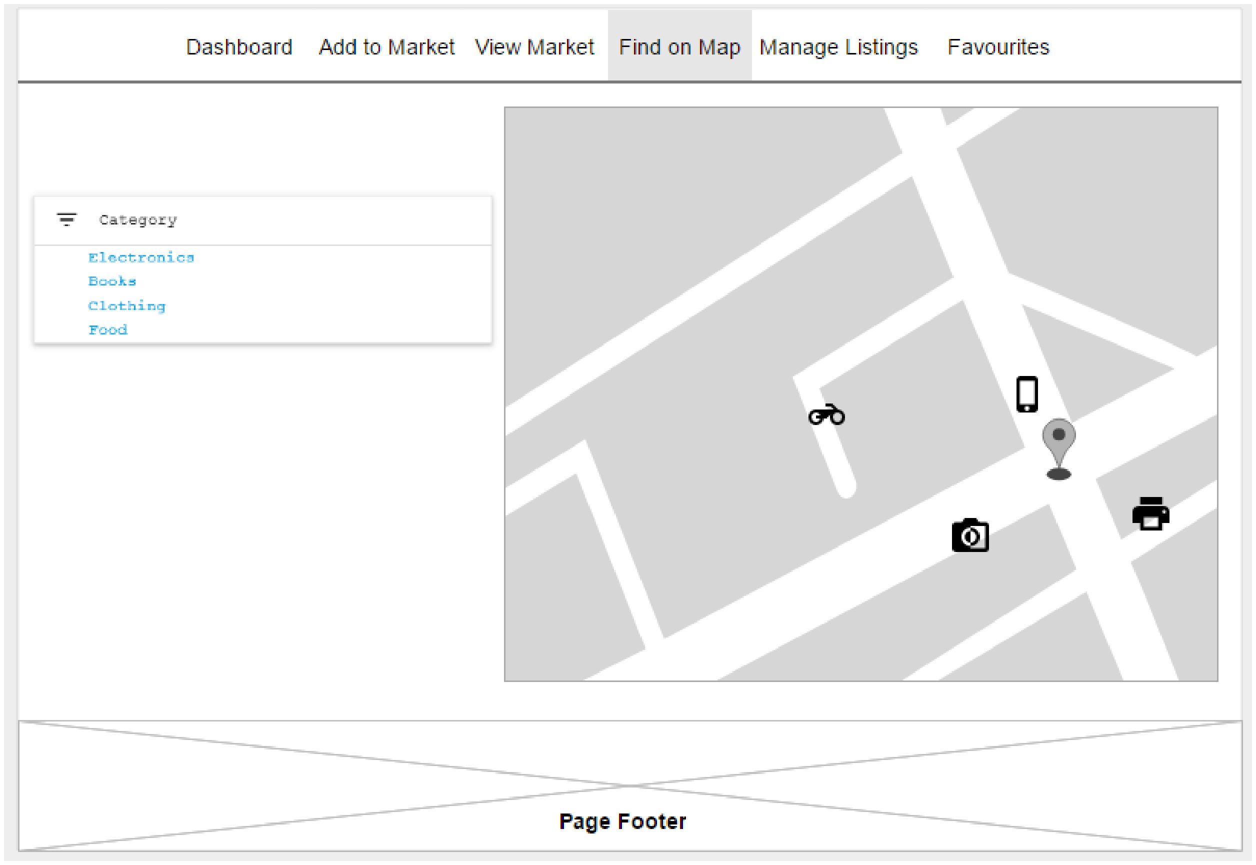


Figure 3.19: Mockup Design – Locate Items on Map

3.5 Traceability Requirement Matrix

Table 3.2 below shows a clear mapping of design to requirement analysis. A dot in the table indicates that the requirement was partially or completely met using the corresponding component.

Web Platform Components:

1. SMS Gateway API (SMS)
2. Database Server (DB)
3. Apache HTTP Web Server (WS)
4. Web Application (WA)

5. Google API (GAPI)
6. Facebook API (FAPI)
7. Twitter API (TAPI)
8. Email Client API (EAPI)

Table 3.2: Matrix Mapping Requirements to Platform Components

Requirement Code	**SMS**	**DB**	**WS**	**WA**	**GAPI**	**FAPI**	**TAPI**	**EAPI**
REQ-IM-1	●	●	●	●				●
REQ-IM-2	●	●	●	●				
REQ-IM-3	●	●	●	●				
REQ-IM-4		●	●	●	●	●	●	
REQ-IDM-1		●	●	●				
REQ-IDM-2		●	●	●				
REQ-IDM-3		●	●	●				
REQ-IDM-4		●	●	●				
REQ-AS-1	●	●	●	●	●	●	●	●
REQ-AS-2		●	●	●	●	●	●	
REQ-AS-3	●	●						●
REQ-AS-4		●	●	●				

4. Chapter 4: Implementation

4.1 Overview

This chapter provides a high-level description of implementation including tools, libraries, frameworks, API's, and components. The document describes components utilized and implementation techniques that were followed in the development of the software product. The chapter also provides evidence of a high-quality implementation and a working prototype.

4.2 Tools, Libraries, Languages and Frameworks Employed

- **PHP:** PHP is the server side scripting language used in the development of the web platform
- **JavaScript:** JavaScript is a web language that was used in this project for dynamic event handling and conditional data display.
- **SQL:** SQL was used for database queries.
- **JSON:** JavaScript Object Notation was used for communication between the server in the backend and the client at the frontend,
- **HTML5:** This markup language was used for placing elements on the web page
- **CSS:** Cascading Style Sheets (CSS) was used in this project to style HTML content.
- **NetBeans IDE:** NetBeans IDE is an open source integrated development environment that supports development with Java, JavaScript, HTML5, PHP, C/C++ among others (NetBeans, 2017). NetBeans has colour-coding and text predictive features essential for code-intensive projects which made it ideal for this project.
- **SMS Gateway API:** This is an integrated service that enables web developers to send and receive messages programmatically from any phone that supports the android

operating system (SMS Gateway, 2017). A request by a developer to send a message from a web application is processed by the API and forwarded to the API's corresponding mobile application on the developer's phone of choice (SMS Gateway, 2017). The mobile phone must support the android operating system. The android application then processes the message and forwards it to the right audience as specified by the developer (SMS Gateway, 2017). The SMS Gateway API serves as a cheap alternative to established SMS service providers such as SMS GH and was thus employed for this project

- **jQuery:** jQuery is a quick, lightweight, and feature-rich JavaScript library that supports HTML document traversal and manipulation, even handling, animation, and AJAX (jQuery, 2017). The library has a simple API and functions across a myriad of web browsers (jQuery, 2017). This library enables developers to build highly responsive and bandwidth efficient web applications with simplified Asynchronous JavaScript and XML (AJAX) code.
- **Materialize:** Materialize is a modern and responsive CSS framework that hinges on Google's Material Design (Materialize, 2014). Materialize is a front-end framework that speeds up development, focuses on the user experience and can be easily integrated for use on web and mobile applications (Materialize, 2014). The framework provides designed components and HTML elements that can easily be adapted for any project.
- **XAMPP:** XAMPP is a lightweight, open source PHP development environment developed and distributed by Apache Friends (Apache Friends, 2017). The environment used for this project consisted of an Apache HTTP Server, MySQL, PHP, and Perl. Recent developments have seen Apache replace MySQL with MariaDB – which is one of

the most popular database servers made by the original developers of MySQL (MariaDB, 2017). XAMPP allows users to develop locally and deploy on a public server later. This radically increases development time with no significant increase to integration time when the project is to be deployed on a public server. This is because XAMPP closely resembles a public server therefore changes prior to deployment are minimal.

- **phpMyAdmin:** This is an open source PHP tool that simplifies the administration of MySQL over the Web (phpMyAdmin, 2003). The tool provides an interface that allows developers to perform useful SQL operations at the click of a button or by conventionally typing out and executing SQL statements (phpMyAdmin, 2003). This tool also simplifies connection to a MySQL database over the Internet since it was developed for over-the-Web implementations.
- **Git:** Git is a free and open source distributed version control system (Git, 2017). The software helps developers to build progressively and return to previous functioning versions in the event of considerable bugs to the application. Git employs a simple syntax and has a desktop application linked to its web platform which further simplifies version control. The desktop application enables users to make changes to specific versions of their development at the click of a button.
- **Android Studio:** Android Studio is the official Integrated Development Environment (IDE) for Android (Android Studio, 2017). This tool provides users with a feature-rich interface that supports debugging, performance tooling, and instant building and deployment (Android Studio, 2017). The environment also possesses an intelligent code editor that supports advanced code prediction and completion (Android Studio, 2017).

Android Studio also allows developers to write once and deploy on all Android devices (Android Studio, 2017).

- **Google Maps API:** Google provides an interface that enables developers to give their users location functionalities and experiences (Google Maps APIs, 2017). APIs are available for Android, iOS, web browsers and HTTP web services (Google Maps APIs, 2017). The API makes a request to users before identifying and displaying their location. Developers can also modify the view of the map as well as add to the map using JavaScript.
- **Google+ Share, Facebook Share, and Twitter Tweet API:** These APIs provide a button and code that links users from a web application to their Google+ dashboard, Facebook wall, and Twitter timeline respectively, if they have the required account. These interfaces give developers an opportunity to increase their social media impact as users connect from an unrelated web application to a social media service provider – in this case Google, Facebook, and Twitter. The APIs require no user authentication.

4.3 Implementation Techniques

This section describes implementation techniques that were employed in the development of the web platform.

4.3.1 Model - View - Controller Pattern (MVC)

This is an implementation strategy in which a software product is categorized into three key sectors – the model, the view, and the controller. The model handles data within the application. The view is responsible for the rendering of the elements that make up the software.

The controller acts as the glue between the model and view and is responsible for the actual functionality of the software product.

- **Reasons for using MVC**
    - MVC reduces code duplication because it separates data logic, application business logic and display
    - MVC is compatible with asynchronous techniques and this aids in the building of highly responsive applications
    - MVC results in modularized applications. This means modification to a part of the software product will not result in changes to the whole application

4.3.2 Asynchronous JavaScript and XML (AJAX)

AJAX is a collection of modern client-side implementation techniques used by developers to build responsive and bandwidth efficient web applications. These techniques allow users to send data to a server in the background and receive data from a server even after the web page has loaded, a paradigm referred to as asynchronous communication (W3Schools, 2017). Users can request data from a server after the page has loaded as well as update a web page without reloading the entire page (W3Schools, 2017). This implementation technique works on the premise of data exchange between server and client through XML and JSON among other communication protocols.

- **Reasons for using AJAX**
    - AJAX is bandwidth efficient since data can be retrieved from a server in the background without a page reload.

    - AJAX increases response time of a web application which results in improvement of performance and speed (Baldota, 2011).
    - The existence of good JavaScript libraries available for use that extensively simplify the technique (Baldota, 2011). Examples include jQuery, Prototype and Scriptaculous (Baldota, 2011).

4.3.3 Feature Driven Development (FDD)

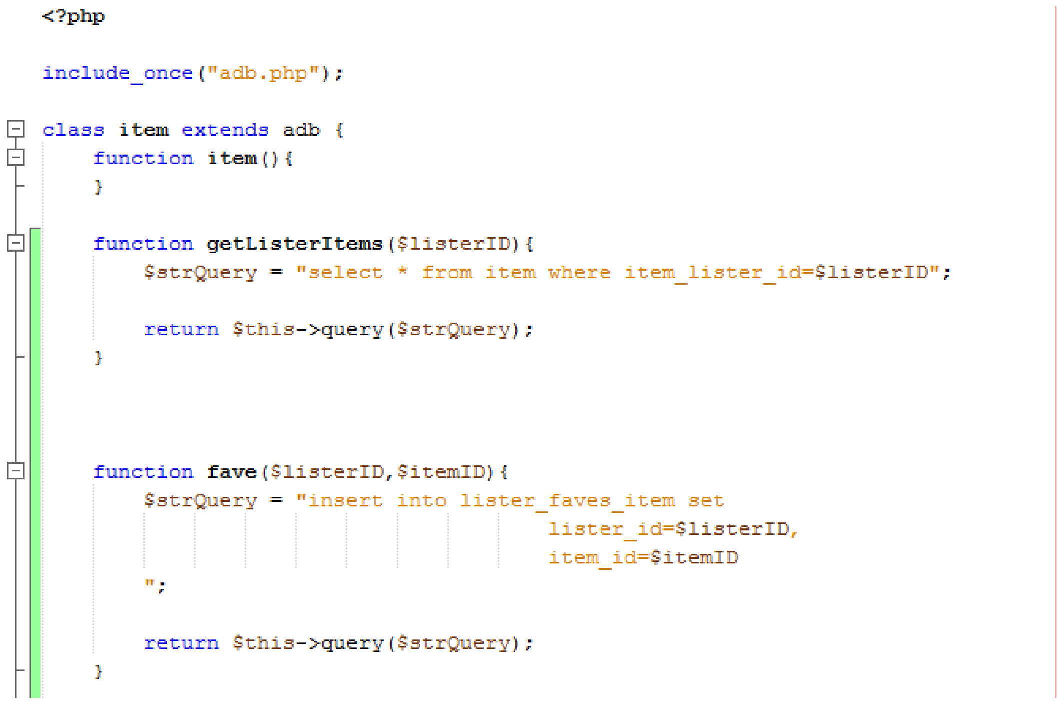
Feature-Driven Development is a pragmatic implementation technique that primes itself on breaking down a complex project into simplified unit features that can be developed in relatively less time as compared to that of the overall project. There are five iterative steps in the development process. First, the developer must come up with an overall model of the system to be developed. The system is then decomposed into a feature list. The software product development is then planned according to the feature list and designed one feature at a time. The application is then built feature by feature. FDD is a type of agile development technique

- **Reasons for using FDD**
    - FDD is a high-level methodology that allows for the incorporation of other low-level techniques.
    - The methodology employed by FDD is simple and concise.
    - FDD results in greater project organization.

4.3.4 Inheritance

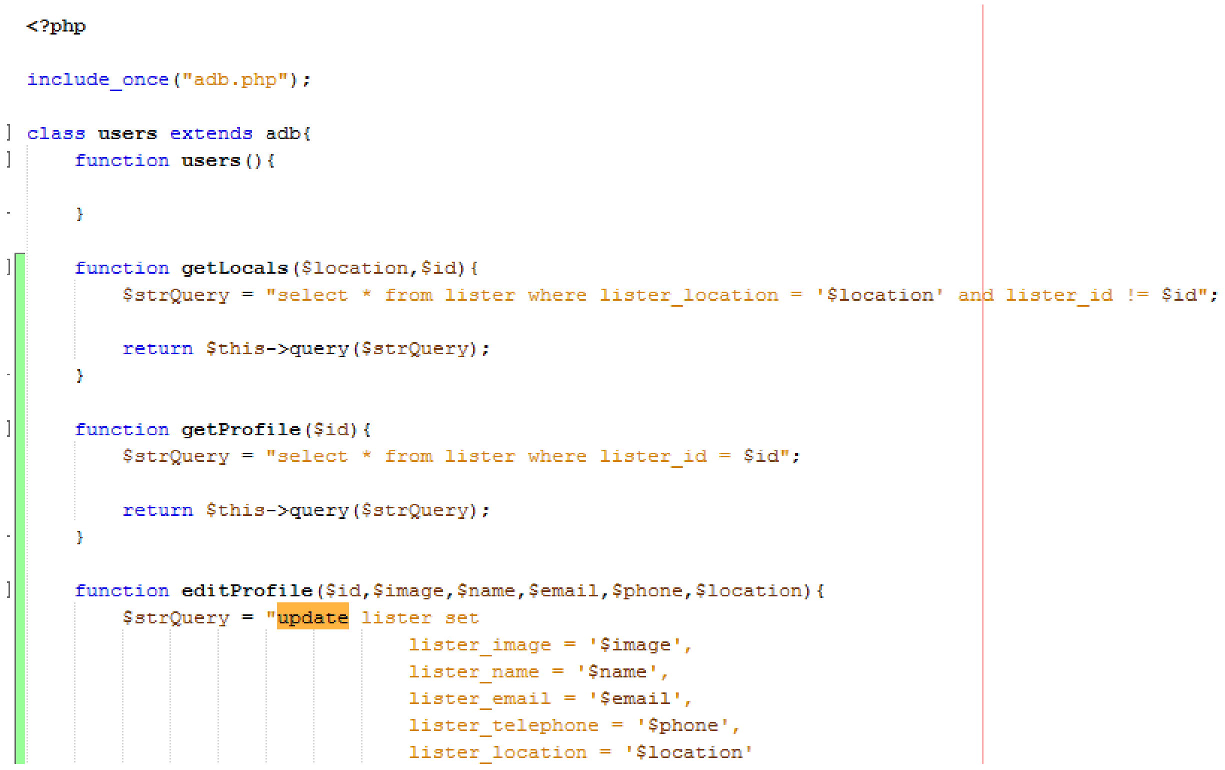
The Object-Oriented Programming (OOP) concept of inheritance was also adopted during implementation. Classes which share certain distinct traits were built on top of each other using the ‘extends’ keyword. For example, both the Item class and User class made use of one database connection class with the aid of inheritance. Listing 4.1 and 4.2 below show evidence of the programming paradigm in use.

Listing 4.1: Evidence of inheritance in Item Class



```
<?php

include_once("adb.php");

class item extends adb {
    function item(){
    }

    function getListerItems($listerID){
        $strQuery = "select * from item where item_lister_id=$listerID";

        return $this->query($strQuery);
    }



    function fave($listerID,$itemID){
        $strQuery = "insert into lister_faves_item set
                                lister_id=$listerID,
                                item_id=$itemID
        ";

        return $this->query($strQuery);
    }
```

Listing 4.2: Evidence of inheritance in Users class


```
<?php

include_once("adb.php");

class users extends adb{
    function users(){

    }

    function getLocals($location,$id){
        $strQuery = "select * from lister where lister_location = '$location' and lister_id != $id";

        return $this->query($strQuery);
    }

    function getProfile($id){
        $strQuery = "select * from lister where lister_id = $id";

        return $this->query($strQuery);
    }

    function editProfile($id,$image,$name,$email,$phone,$location){
        $strQuery = "update lister set
                        lister_image = '$image',
                        lister_name = '$name',
                        lister_email = '$email',
                        lister_telephone = '$phone',
                        lister_location = '$location'
```

4.4 How the Application Works

The web platform works in three main ways. The item management aspect allows users to add and manipulate items they add to the online market. The item display management section enables users control what they see when they visit the online market. The functionality categorized as auxiliary services are added-on services privy to users with an account only. This is because these functionalities require authentication of a user's identity due to increased privacy and security requirements. An example is the auxiliary service that enables users to visualize the location of advertised items on an embedded map.

4.5 Evidence of Implementation

This section provides evidence of the implemented web platform and elementary implementation of an alternative architecture considered prior to the development of the web platform.

4.5.1 Alternative Architecture: Mobile Application

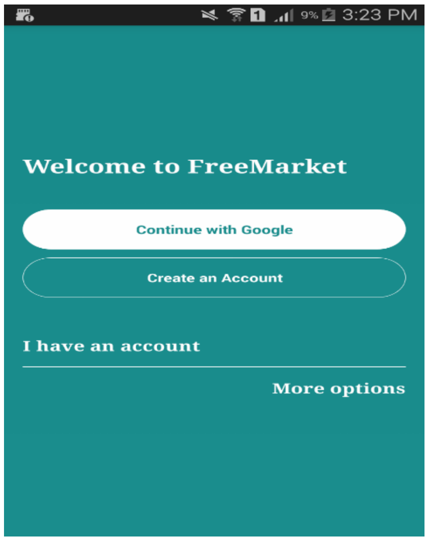
Figure 4.1 to Figure 4.3 show implementation of an alternative architecture initially considered. At the beginning of the project, a mobile application was considered to solve the problem. However, upon further consideration, a web platform was chosen as the preferred architecture, with the mobile application considered for future works.

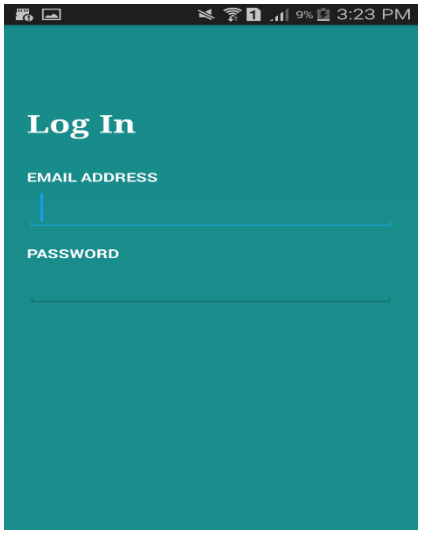
Figure 4.1 (A1) shows the mobile application's first interface where users can create an account or login to an existing account. Figure 4.1 (A3) shows a registered user's dashboard view after logging in.

Figure 4.2 (Group B) shows the online market view. Users can view items one at a time and can traverse through the online marketplace by swiping their screen. There is a request option at the top of the screen where users can get in touch with the owner of an item either by text message, email, or voice call.

Figure 4.3 (Group C) shows the interface that allows users to request for items from owners. Figure 4.3 (C1) shows an interface with three tabs with which users can request for an item of their interest. Figure 4.3 (C2) and Figure 4.3 (C3) show interfaces that correspond to a text message and email request respectively

A1 A2 A3





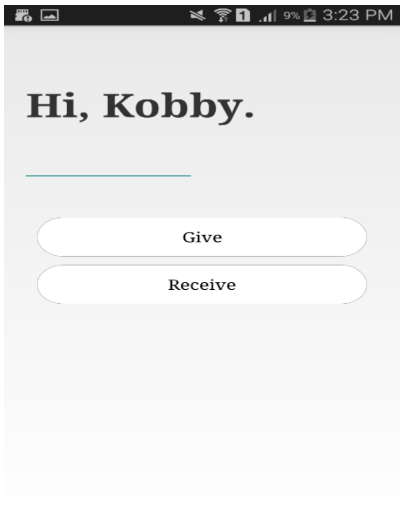
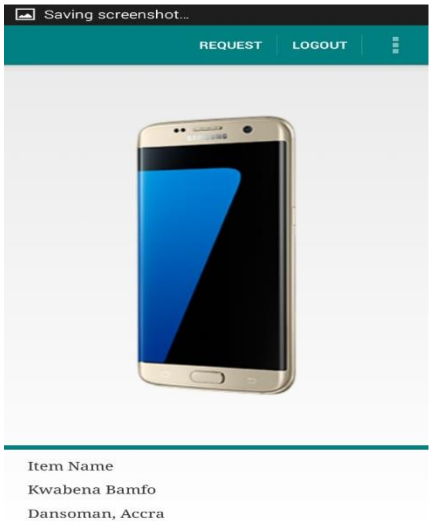
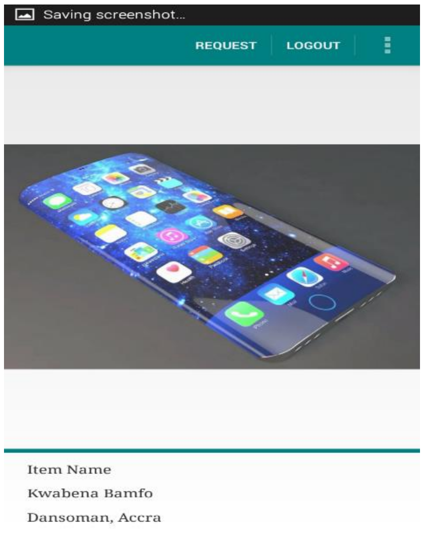


Figure 4.1: Mobile Application Interface (Group A)

B1 B2 B3





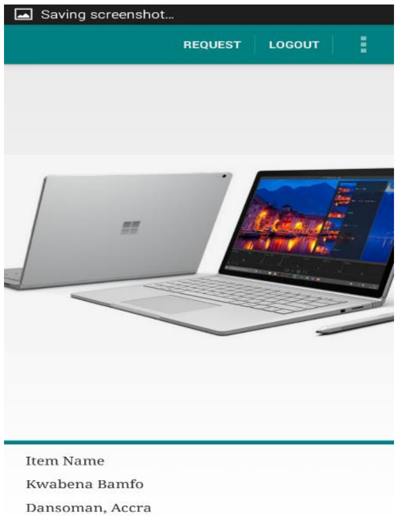
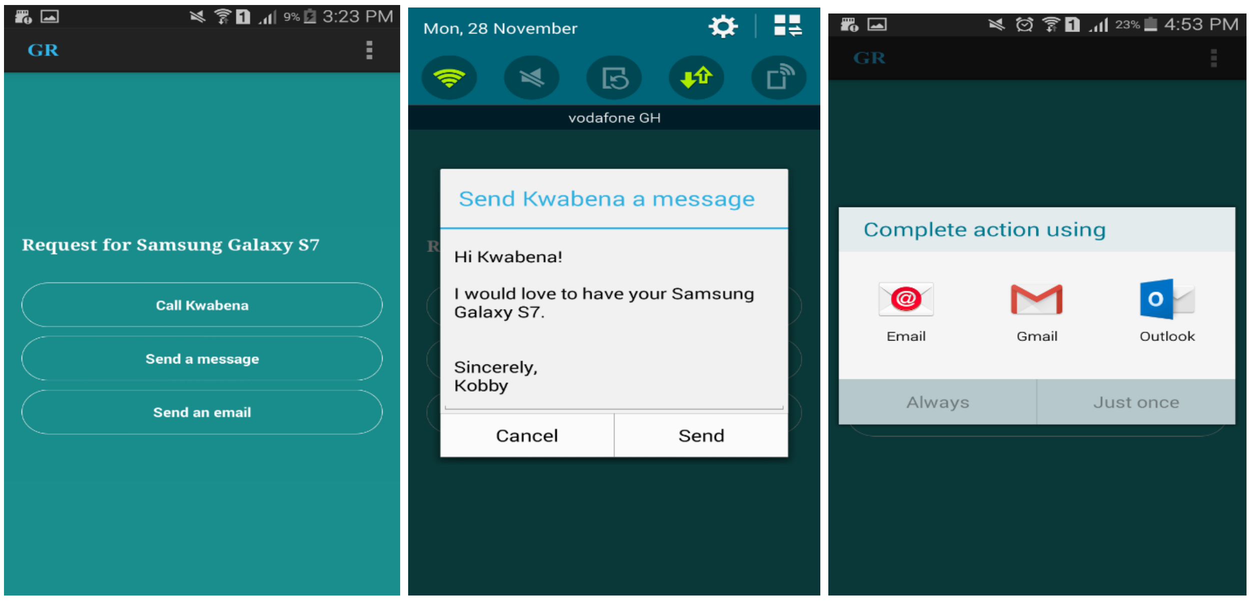


Figure 4.2: Mobile Application Interface (Group B)

C1 C2 C3



Hi Kwabena!

I would love to have your Samsung Galaxy S7.

Sincerely,
Kobby

Figure 4.3: Mobile Application Interface (Group C)

4.5.2 Chosen Architecture: Web Platform

This section shows evidence of implementation for the chosen architecture. The document describes the interfaces shown to users with an account and users without an account.

4.5.2.1 Graphical User Interface: Users without an Account

This section shows the web application interfaces rendered to users who have not created an account on the web application.

4.5.2.1.1 Homepage

Figure 4.4 below shows the homepage of the web application. From this page, users can navigate to all other available pages. Users have the option to add an item to the online market, view the market, create an account, and log in to an existing account.

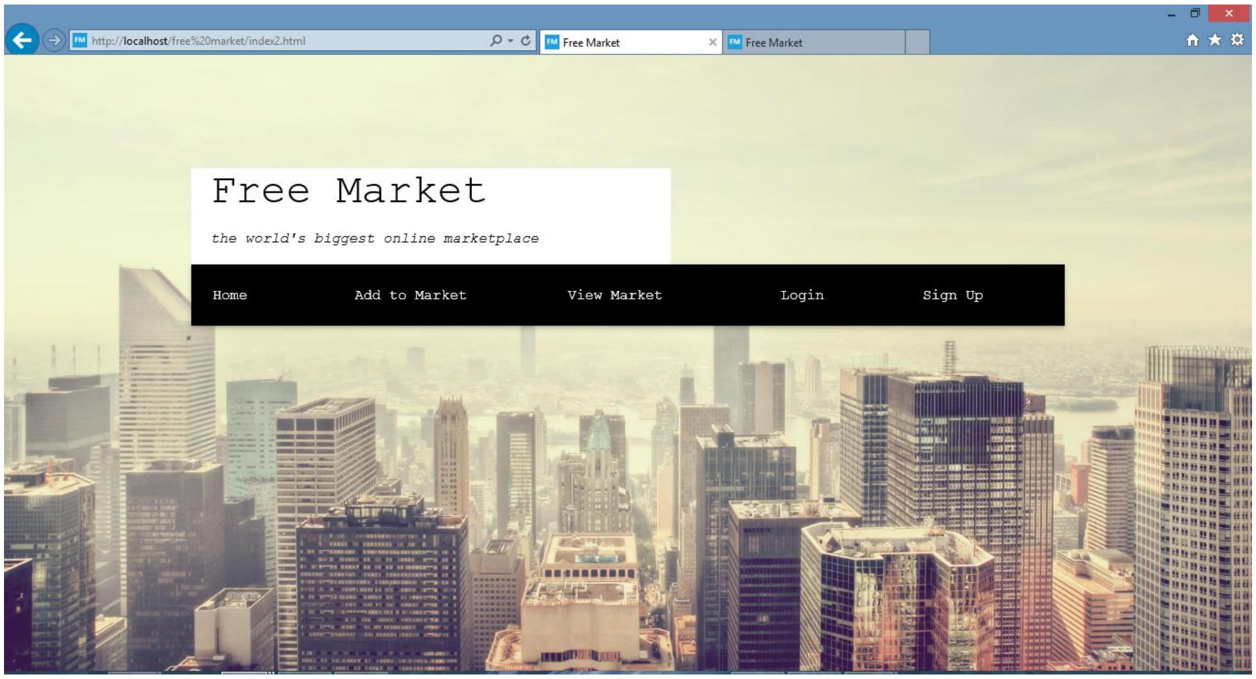

http://localhost/free%20market/index2.html
Free Market
Free Market
Free Market
the world's biggest online marketplace
Home
Add to Market
View Market
Login
Sign Up

Figure 4.4: Web Application Interface – Homepage

4.5.2.1.2 Add to Market

Figure 4.5 shows the interface used to add an item to the market. Users can add an image and preview the image on the page prior to adding the item to the market. The page has two forms whose inputs are validated before adding the item to the market. The page has a footer, a theme which runs across many of the other pages on the platform, where users can connect to the application's social media accounts.

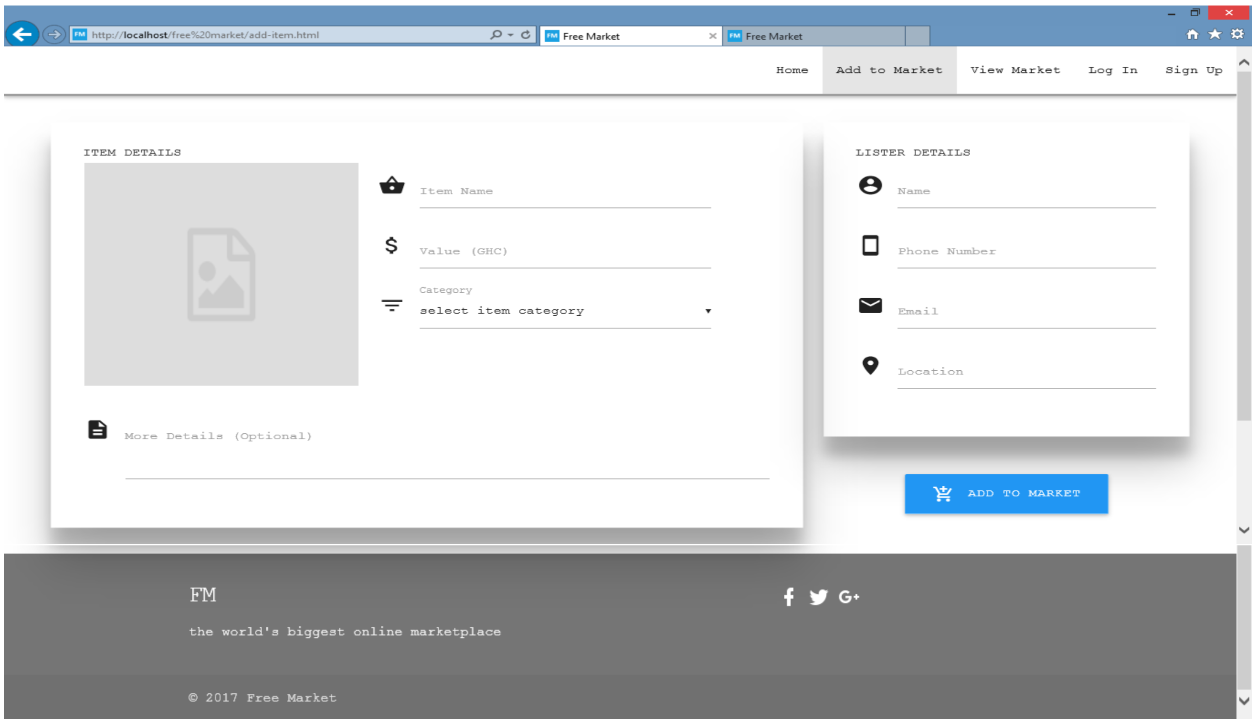


Figure 4.5: Web Application Interface – Add to Market

4.5.2.1.3 View Online Market

Figure 4.6 and Figure 4.7 below show the view of the online marketplace. The interface has a search bar where users can specify the item(s) displayed on the market. The interface also has filters to the left of the page and on top of the page where users can specify a category or location or both. There is also a sorting filter at the top of the page which enables users sort the market view based on the time added and the estimated value of the item. Figure 4.7 shows the result of a click on an item image. Comparing Figure 4.6 and Figure 4.7, it can be observed that the item image in Figure 4.6 is replaced with the item details in Figure 4.7 in response to a click on the image. The items are displayed in cards, with two buttons at the bottom right which allows users to share the item on social media and expand the view of the item respectively.

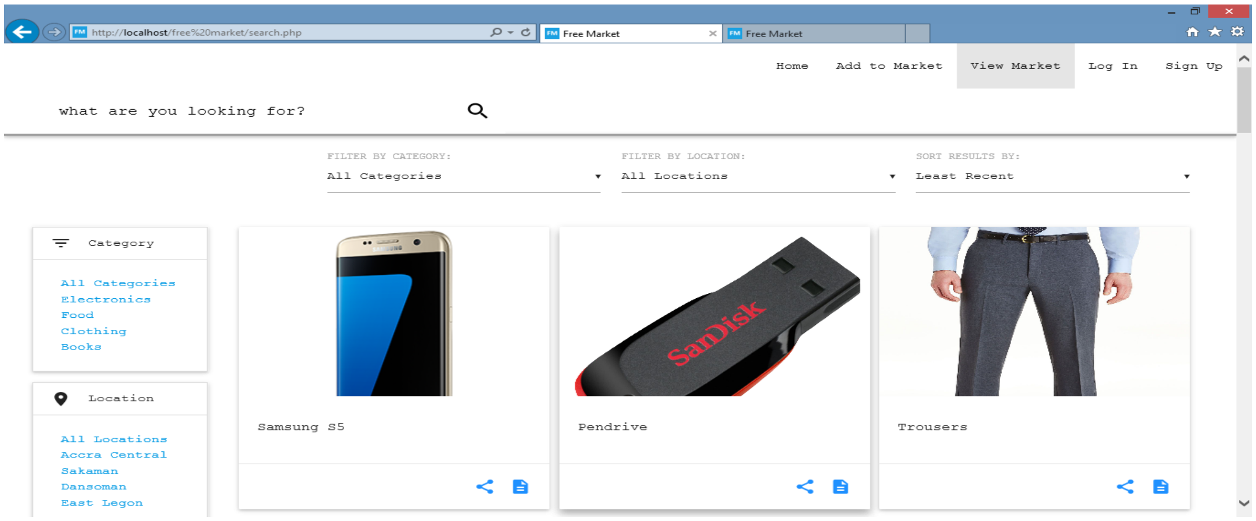
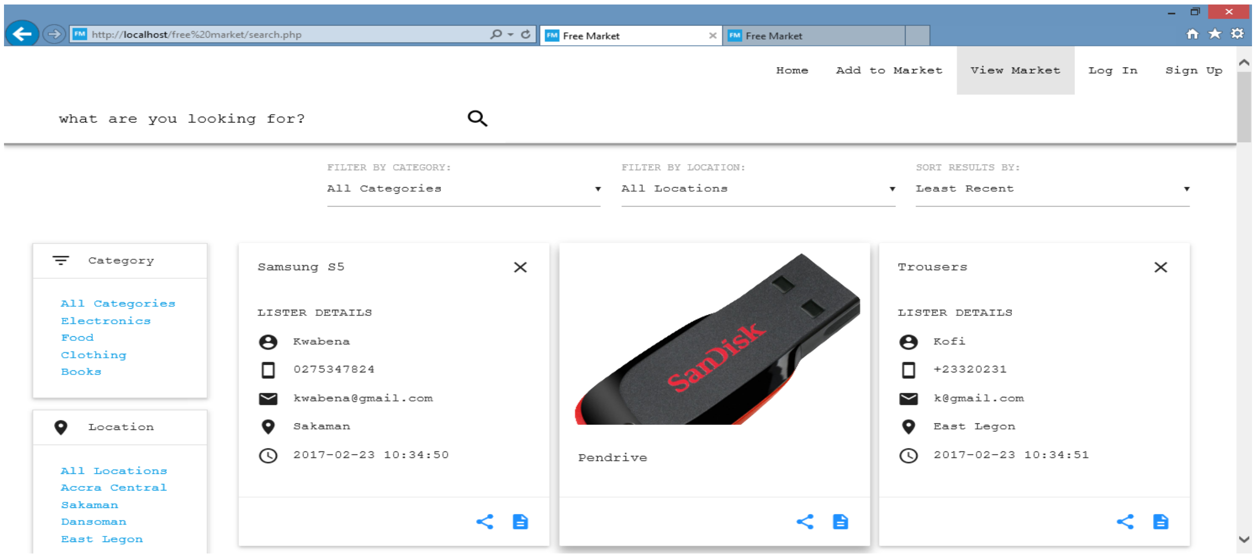


Figure 4.6: Web Application Interface – View Market (D1)



Samsung S5

LISTER DETAILS

Kwabena
0275347824
kwabena@gmail.com
Sakaman
2017-02-23 10:34:50

Trousers

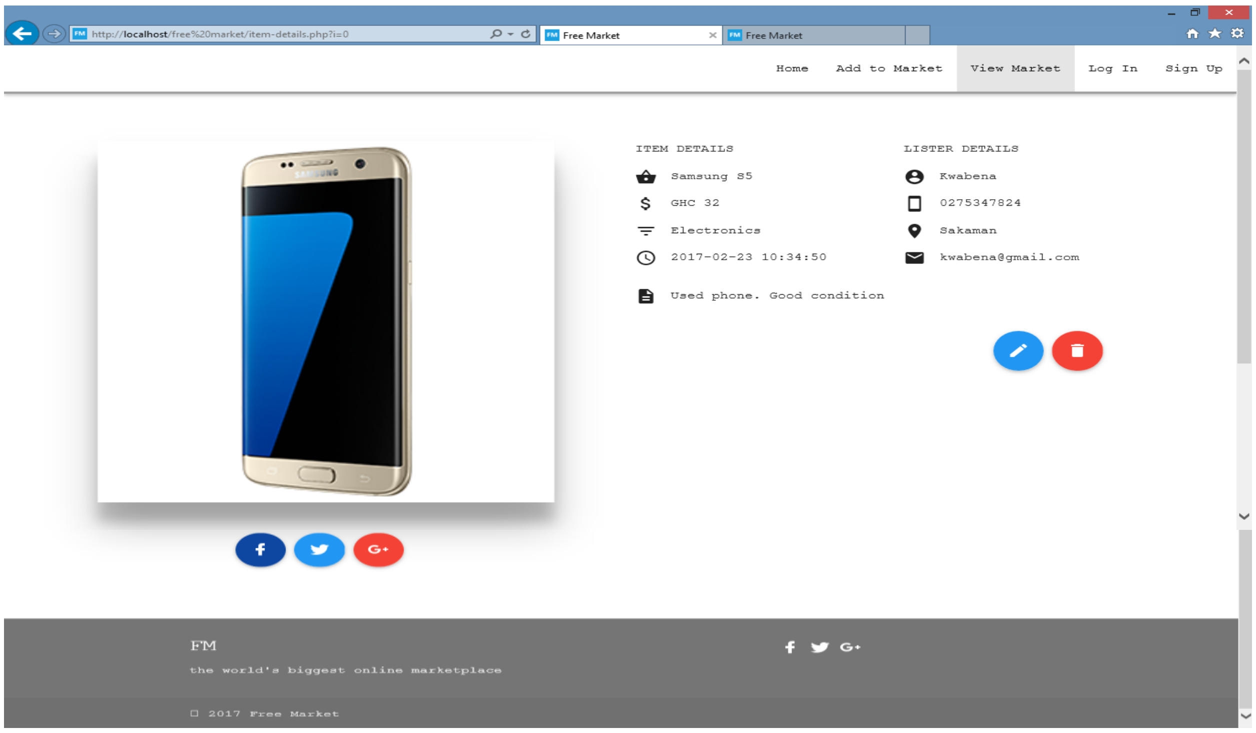
LISTER DETAILS

Kofi
+23320231
k@gmail.com
East Legon
2017-02-23 10:34:51

Figure 4.7: Web Application Interface – View Market (D2)

4.5.2.1.4 View Item Details

Figure 4.8 shows the interface where a comprehensive list of an item's details can be viewed. The interface also enables users to edit the items details, delete the item, and share the item on specific social media platforms (Google+, Twitter, Facebook).



ITEM DETAILS

Samsung S5

GHC 32

Electronics

2017-02-23 10:34:50

Used phone. Good condition

LISTER DETAILS

Kwabena

0275347824

Sakaman

kwabena@gmail.com

FM

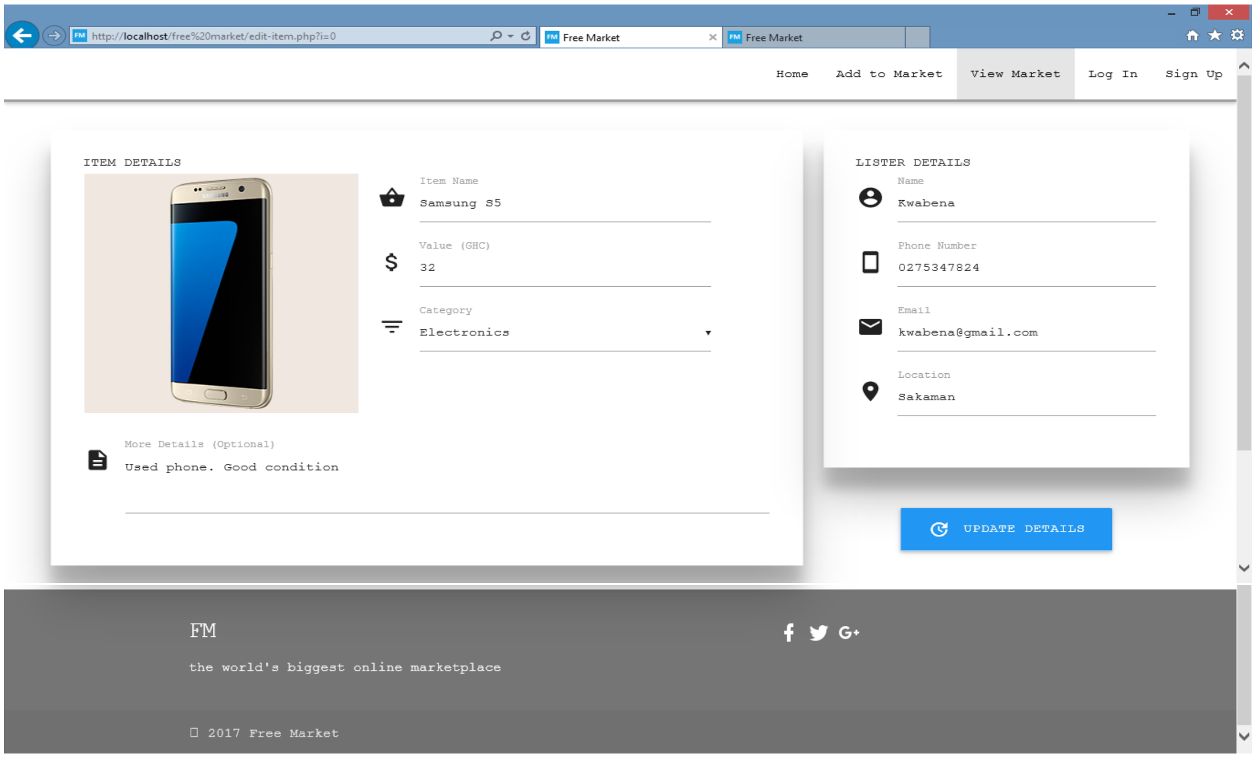
the world's biggest online marketplace

2017 Free Market

Figure 4.8: Web Application Interface – View Item Details

4.5.2.1.5 Update Item Details

Figure 4.9 shows the interface where users can edit the details of an item. The forms are auto filled with the items current details which can then be edited.


http://localhost/free%20market/edit-item.php?i=0

ITEM DETAILS

Item Name: Samsung S5

Value (GHC): 32

Category: Electronics

More Details (Optional): Used phone. Good condition

LISTER DETAILS

Name: Kwabena

Phone Number: 0275347824

Email: kwabena@gmail.com

Location: Sakaman

Figure 4.9: Web Application Interface – Update Item Details

4.5.2.1.6 Log In

Figure 4.10 shows the interface where users can log into the web application. At the top of the form, there are two buttons that allow users to log into the application via Facebook and Google+ respectively. Users can also ask the application services to remember their password so they can skip authentication the next time they visit the web page. There is also a 'forgotten password' feature that allows users to reset their password through their email.

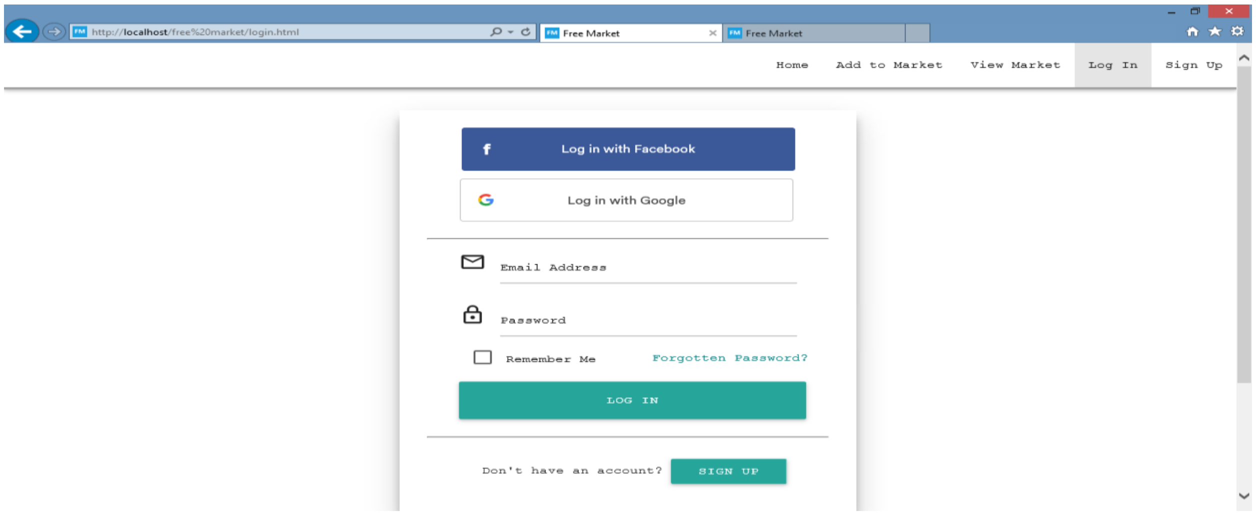
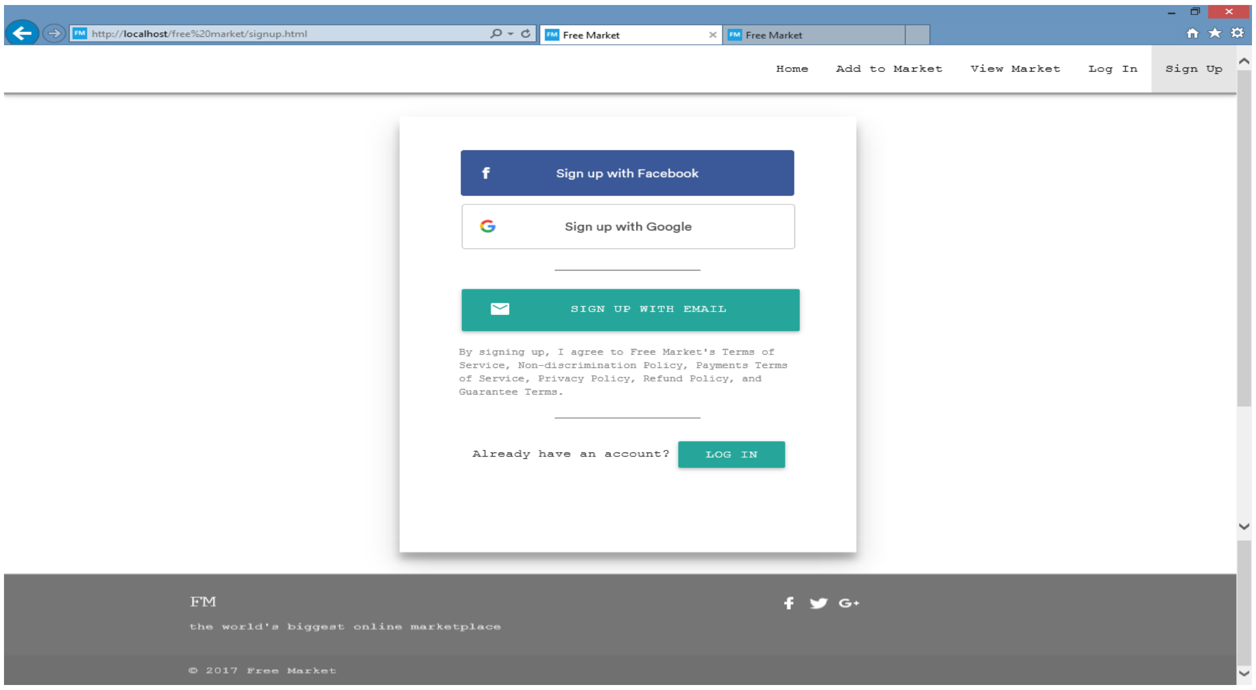


Figure 4.10: Web Application Interface – Login

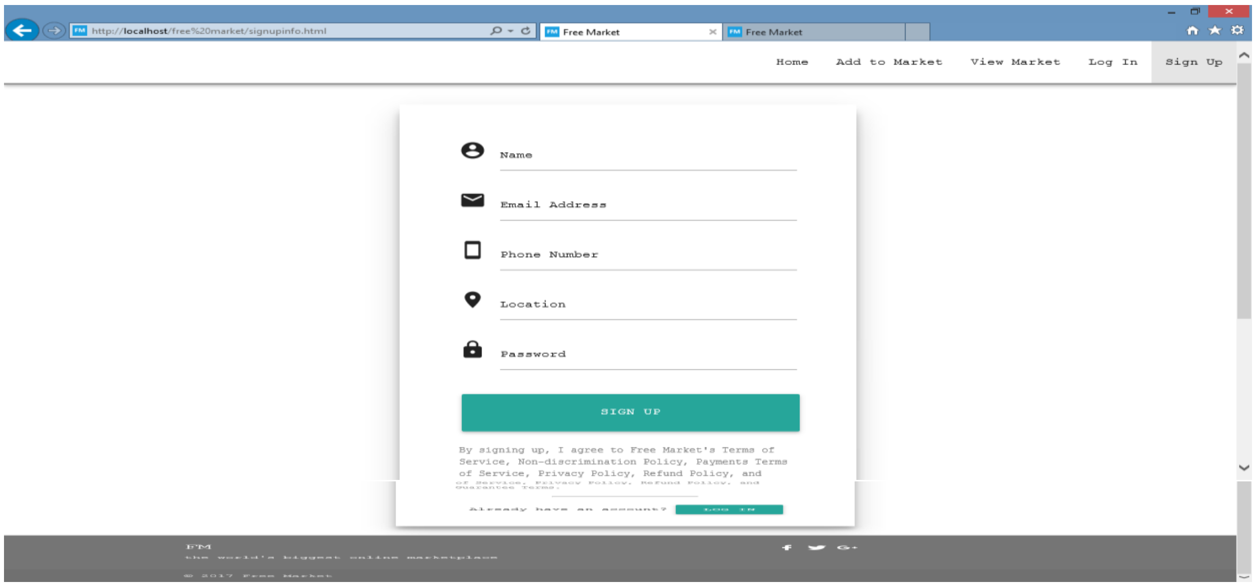
4.5.2.1.7 Sign Up

Figure 4.11 and Figure 4.12 below highlight the interfaces that allow users to create an account on the platform. Users can alternatively sign up with Facebook or Google. Figure 4.12 shows the form users have to fill in order to create an account on the platform.



By signing up, I agree to Free Market's Terms of Service, Non-discrimination Policy, Payments Terms of Service, Privacy Policy, Refund Policy, and Guarantee Terms.

Figure 4.11: Web Application Interface: Signup (E1)



By signing up, I agree to Free Market's Terms of Service, Non-discrimination Policy, Payments Terms of Service, Privacy Policy, Refund Policy, and Guarantee Terms.

Figure 4.12: Web Application Interface – Signup (E2)

4.5.2.2 Graphical User Interface: Users with an Account

This section shows the web application interfaces rendered to users who have an account on the web platform.

4.5.2.2.1 Dashboard

Figure 4.13 displays the interface where users can access all the functionalities the web platform affords. The user's image is placed at the top right hand corner of the web page.

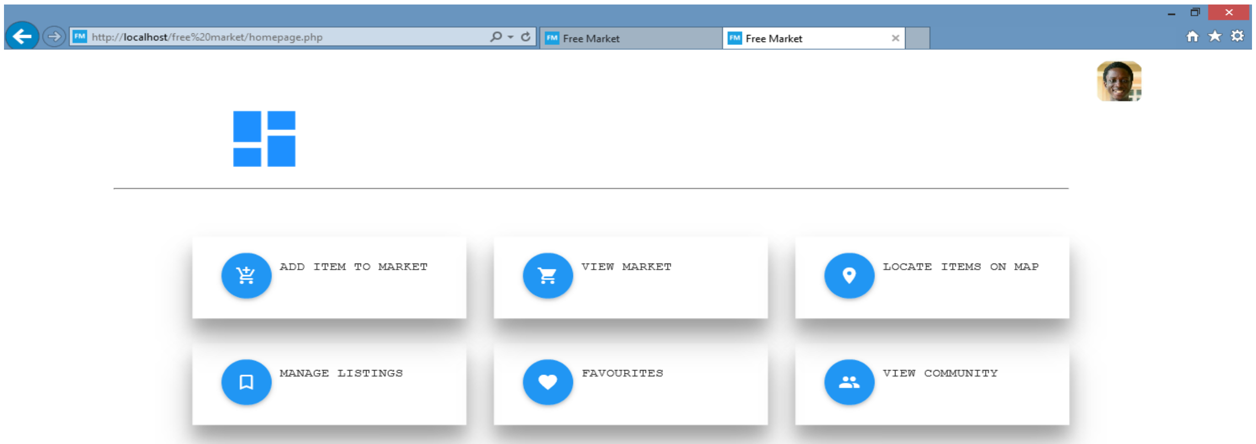
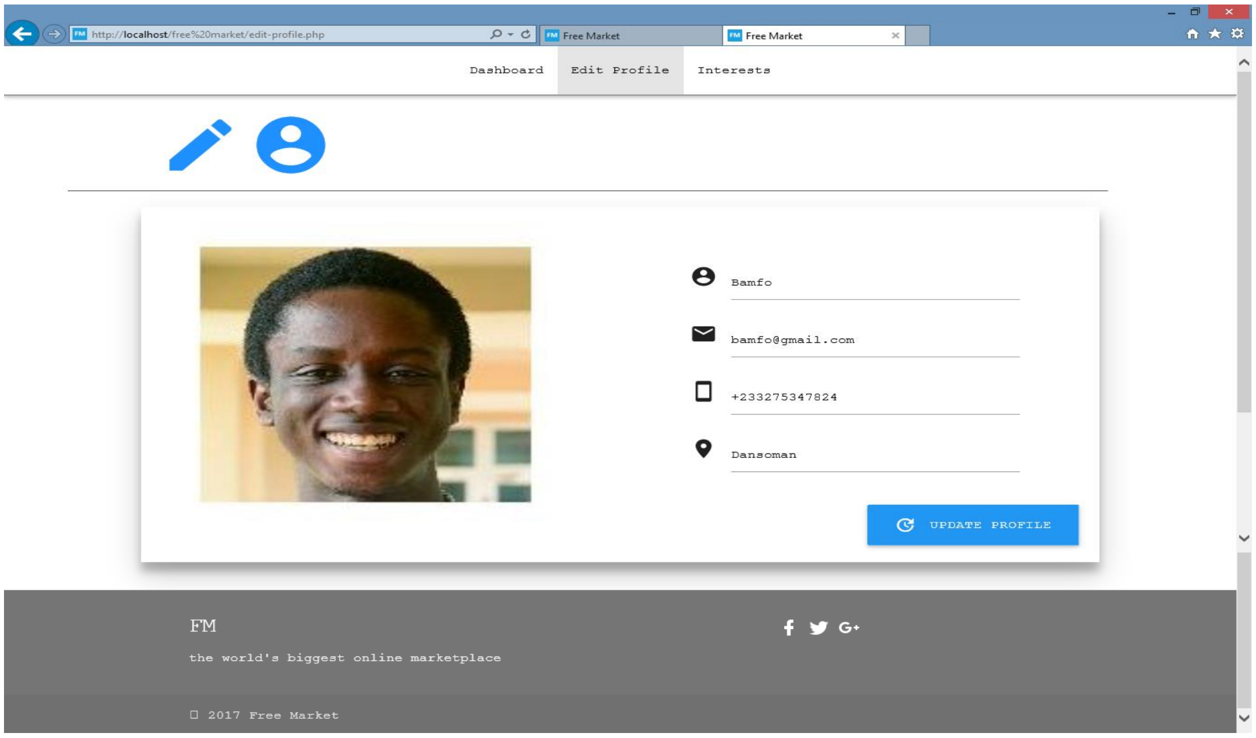


Figure 4.13: Web Application Interface – Dashboard

4.5.2.2.2 Edit Profile Details

Figure 4.14 displays the interface where users can edit their account details.



Bamfo

bamfo@gmail.com

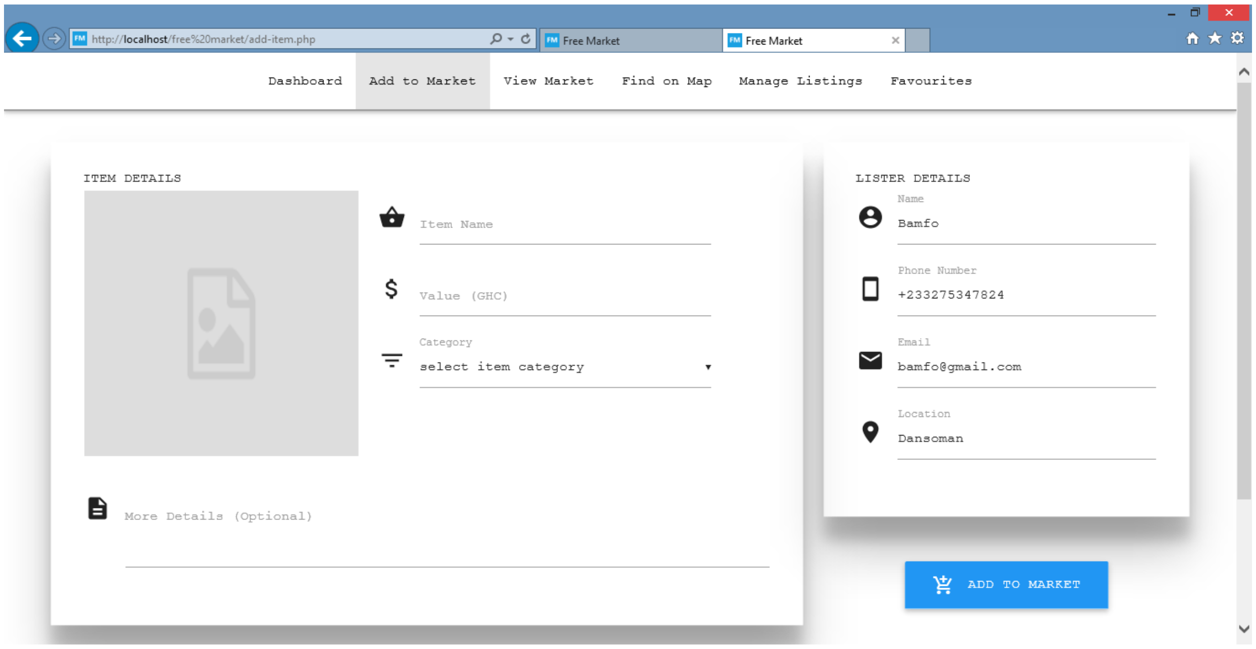
+233275347824

Dansoman

Figure 4.14: Web Application Interface – Edit Profile

4.5.2.2.3 Add to Market

Figure 4.15 show below displays the interface where users with an account can add an item to the market. As can be seen below, the lister details form is already pre-filled since the identity of the user has been authenticated. This interface also shows the navigation menu items centered at the top of the page, as compared to the 'add to market' interface for users without an account in Figure 4.5 where they are aligned to the right of the screen.

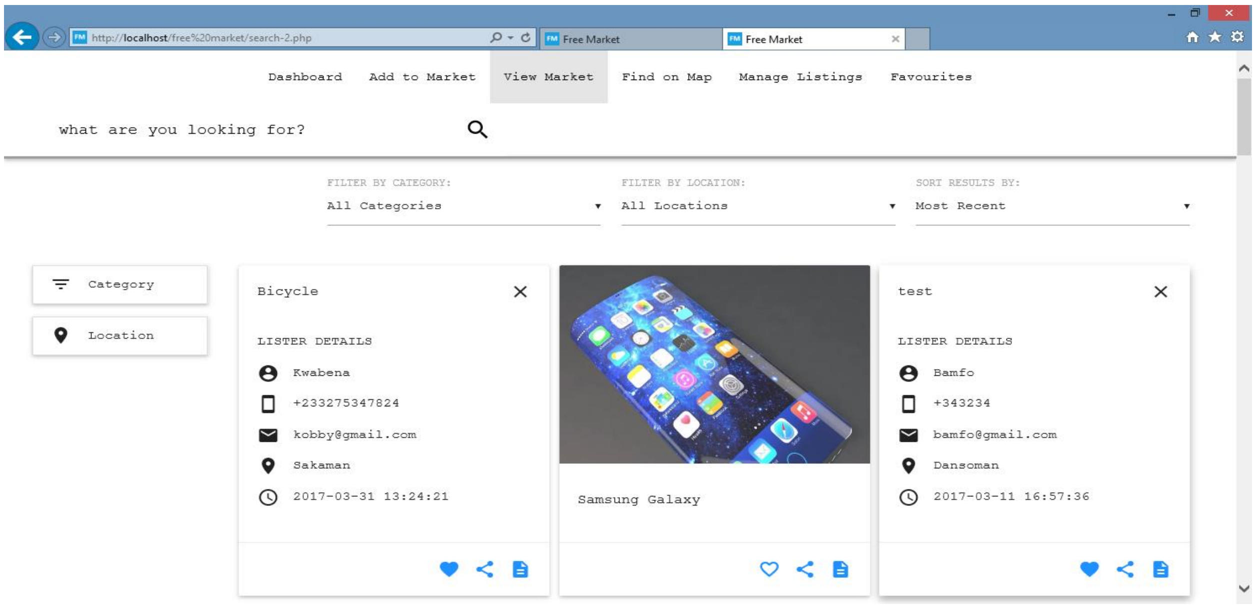


Name
Bamfo
Phone Number
+233275347824
Email
bamfo@gmail.com
Location
Dansoman

Figure 4.15: Web Application Interface – Add to Market

4.5.2.2.4 View Online Market

Figure 4.16 shows the online marketplace interface displayed to users who have created an account on the platform. The items are placed in cards with action buttons at the bottom right side of the card. Compared to Figure 4.6 which shows the market display for users without an account, there is an additional icon at the bottom of the card. This icon is placed in support of the favourite feature. Users can click on the 'love' symbol to favourite or unfavourite an item as desired.



Bicycle

LISTER DETAILS

Kwabena

+233275347824

kobby@gmail.com

Sakaman

2017-03-31 13:24:21

Samsung Galaxy

test

LISTER DETAILS

Bamfo

+343234

bamfo@gmail.com

Dansoman

2017-03-11 16:57:36

Figure 4.16: Web Application Interface – View Market

4.5.2.2.5 Locate Items on Google Maps

Figure 4.17 below shows an auxiliary feature where users can visualize the online marketplace on Google Maps. The interface enables users to view items that have been advertised for free in their locality. Users can also filter the items displayed on the Map according to category.

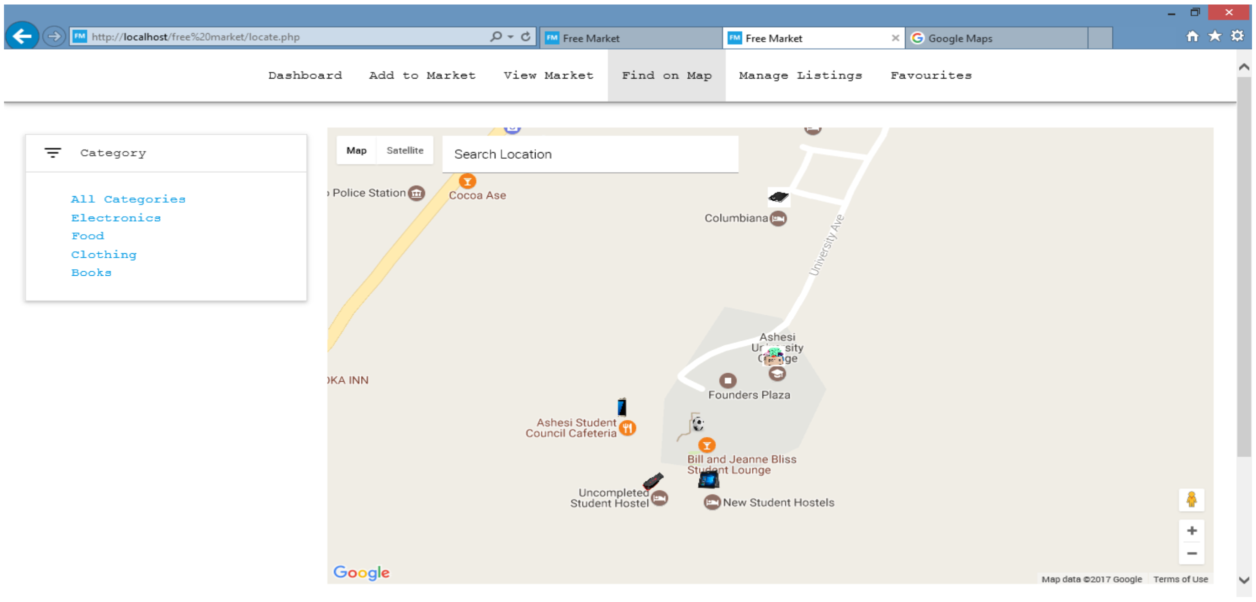


Figure 4.17: Web Application Interface – Locate Items on Map

4.5.2.2.6 Manage Listings

Figure 4.18 shows the interface where users can view items they have added to the online marketplace. From this page, users can edit and delete items they have added to the market.

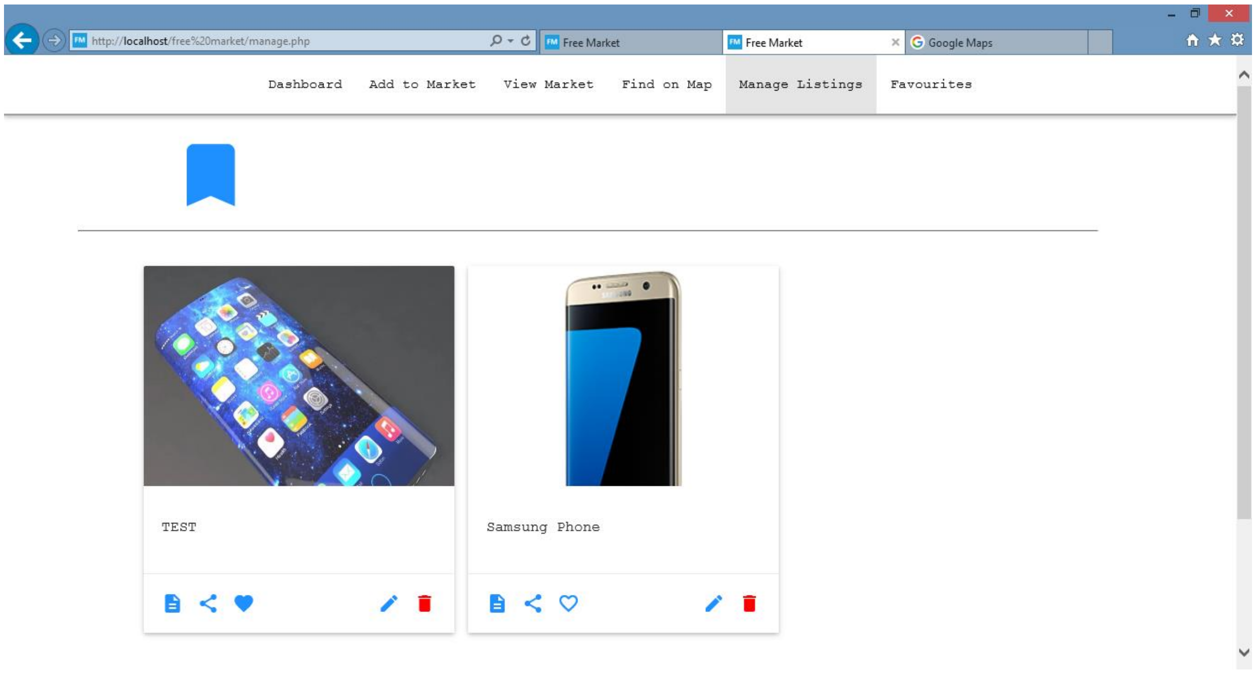
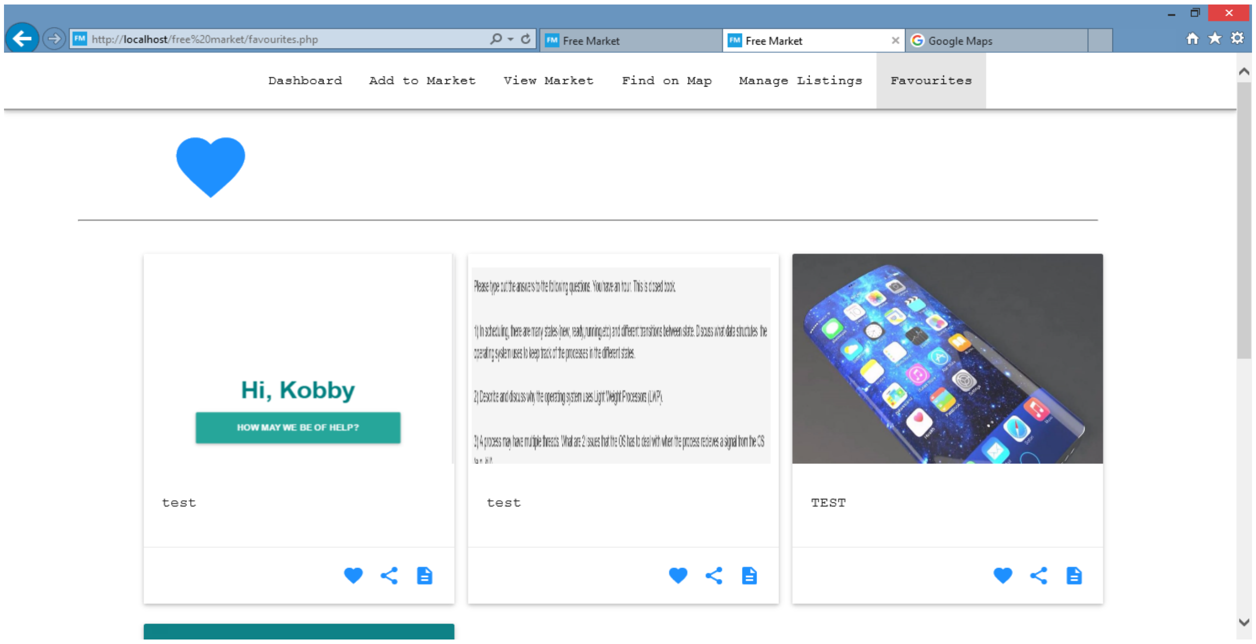


Figure 4.18: Web Application Interface – Manage Listings

4.5.2.2.7 View Favourites

Figure 4.19 shows the interface where users with an account can view items they have favourited in the past. This serves as a reference repository for users with an account.



Please type out the answers to the following questions. You have an hour. This is closed book.

1) In scheduling, there are many states (new, ready, running etc) and different transitions between state. Discuss what data structures the operating system uses to keep track of the processes in the different states.

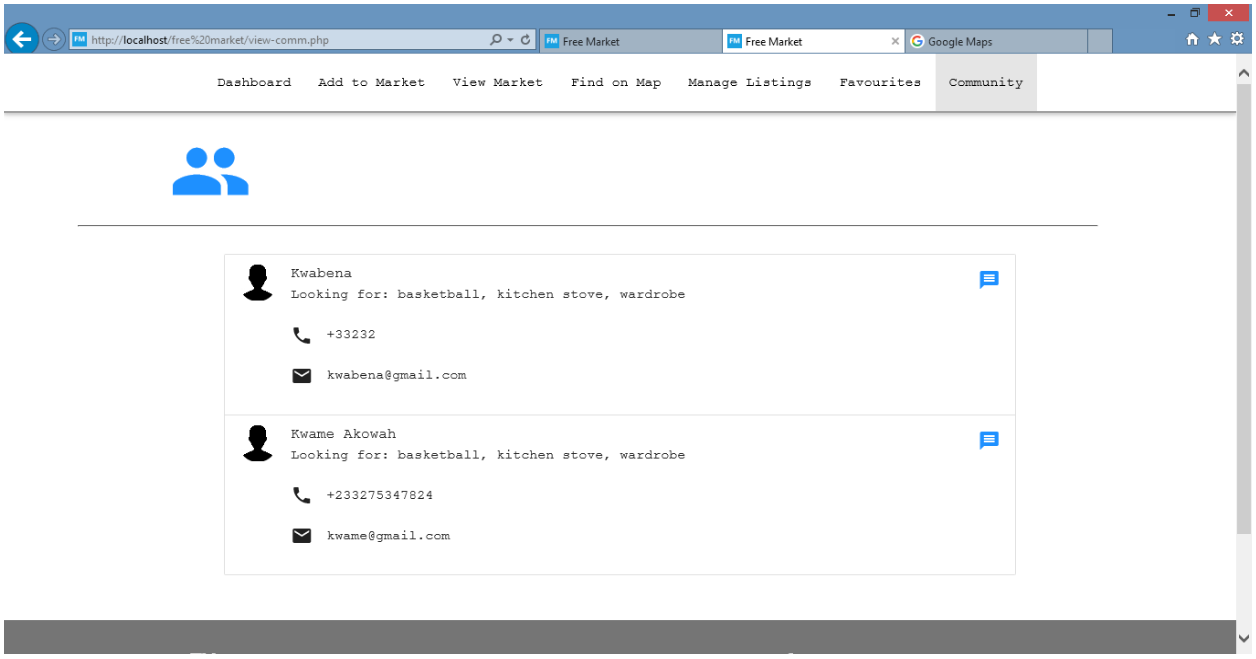
2) Describe and discuss why the operating system uses Light Weight Processes (LWP).

3) A process may have multiple threads. What are 2 issues that the OS has to deal with when the process recieves a signal from the OS

Figure 4.19: Web Application Interface – Favourites

4.5.2.2.8 View Community

Figure 4.20 below displays an interface where users can view contact details of other users in their specified locality. Users can also get to know the interests, needs and wants of people close to them and offer to help should they be in a position to do so. Users can also communicate among themselves through the web platform rather than email or voice call.



Kwabena
Looking for: basketball, kitchen stove, wardrobe
+33232
kwabena@gmail.com

Kwame Akowah
Looking for: basketball, kitchen stove, wardrobe
+233275347824
kwame@gmail.com

Figure 4.20: Web Application Interface – View Community

5. Chapter 5: Testing and Results

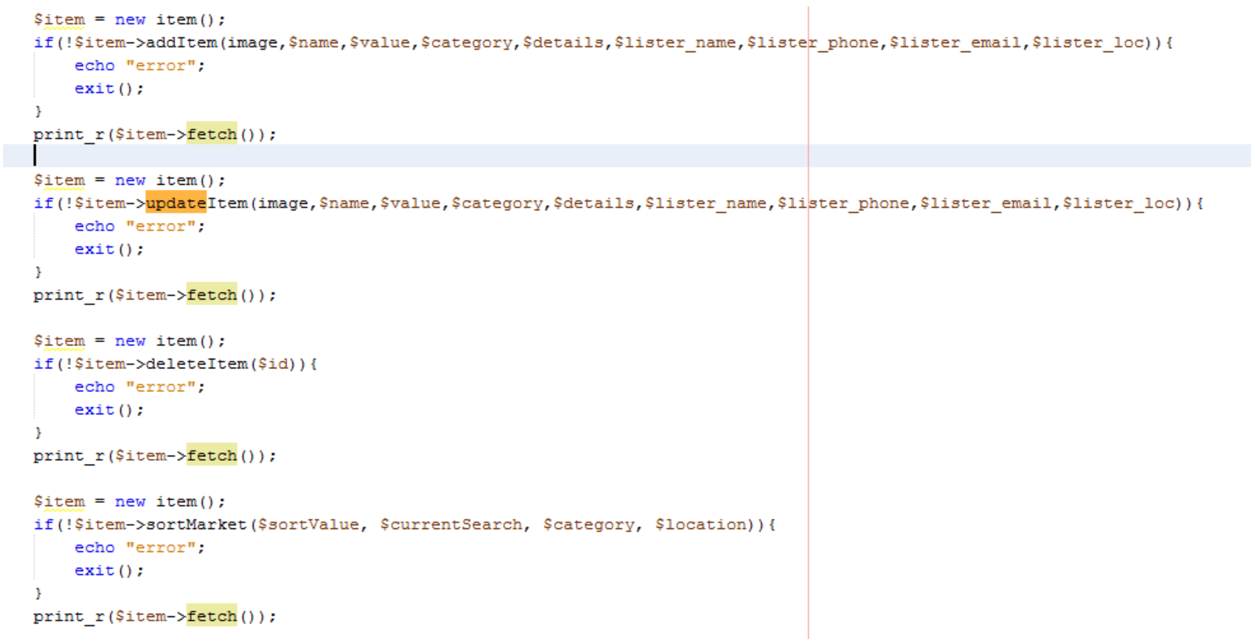
5.1 Overview

This chapter highlights the tests that were conducted to verify satisfaction of functional requirements. The chapter describes the testing process and provides test results

5.2 Unit Testing

The web platform was tested incrementally in units during the development process. The functions shown in Listing 5.1 and Listing 5.2 were executed with actual results compared against expected results. In instances where failures were recorded, the source code was reviewed and corrected as appropriate.

Listing 5.1: Unit test code for Item Class

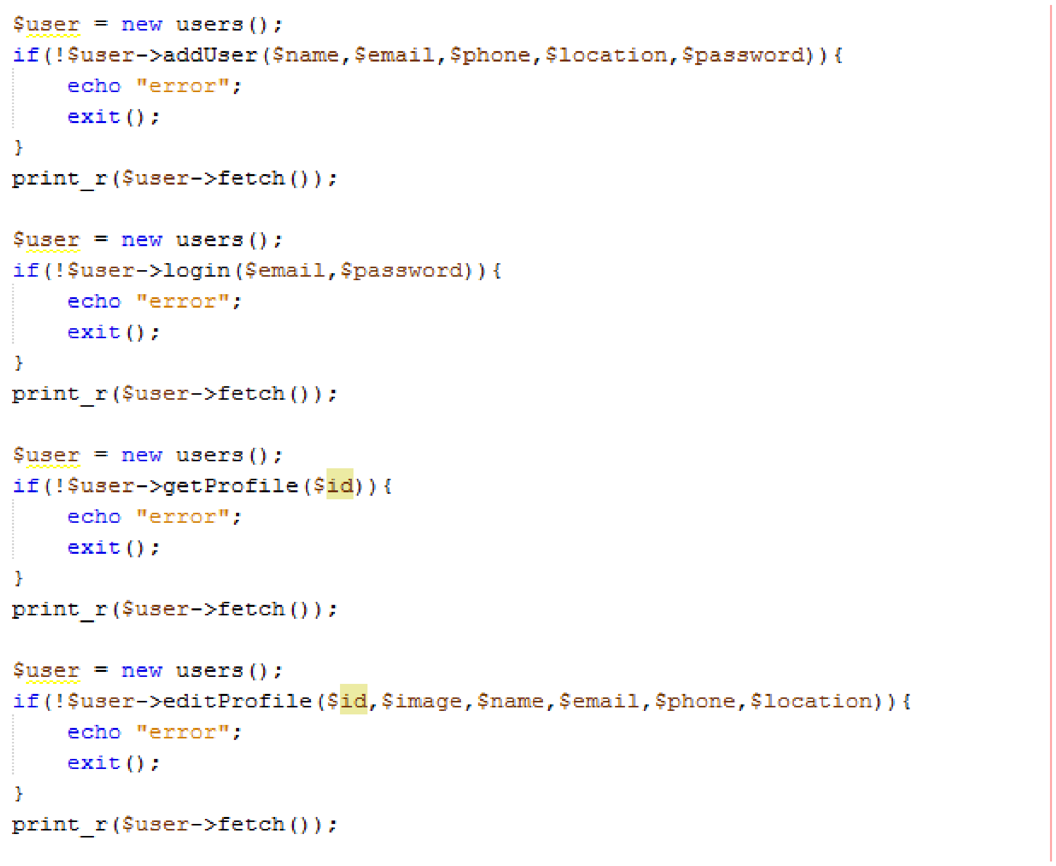


```
$item = new item();
if(!$item->addItem(image,$name,$value,$category,$details,$lister_name,$lister_phone,$lister_email,$lister_loc)){
    echo "error";
    exit();
}
print_r($item->fetch());

$item = new item();
if(!$item->updateItem(image,$name,$value,$category,$details,$lister_name,$lister_phone,$lister_email,$lister_loc)){
    echo "error";
    exit();
}
print_r($item->fetch());

$item = new item();
if(!$item->deleteItem($id)){
    echo "error";
    exit();
}
print_r($item->fetch());

$item = new item();
if(!$item->sortMarket($sortValue, $currentSearch, $category, $location)){
    echo "error";
    exit();
}
print_r($item->fetch());
```

Table 5.1: Summary of unit test results for Item Class

Functionality	**Expected Result**	**Actual Result**
Add Item	Item can be seen on online market	Item can be seen on online market
Update Item	Item details changes to new details	Item details changes to new details
Delete Item	Item cannot be viewed on the online market	Item cannot be viewed on the online market
Sort Market(sort value)	Online market is sorted according to sort value	Online market is sorted according to sort value

Listing 5.2: Unit test code for User Class



```
$user = new users();
if(!$user->addUser($name,$email,$phone,$location,$password)){
    echo "error";
    exit();
}
print_r($user->fetch());

$user = new users();
if(!$user->login($email,$password)){
    echo "error";
    exit();
}
print_r($user->fetch());

$user = new users();
if(!$user->getProfile($id)){
    echo "error";
    exit();
}
print_r($user->fetch());

$user = new users();
if(!$user->editProfile($id,$image,$name,$email,$phone,$location)){
    echo "error";
    exit();
}
print_r($user->fetch());
```

Table 5.2: Summary of unit test results for User Class

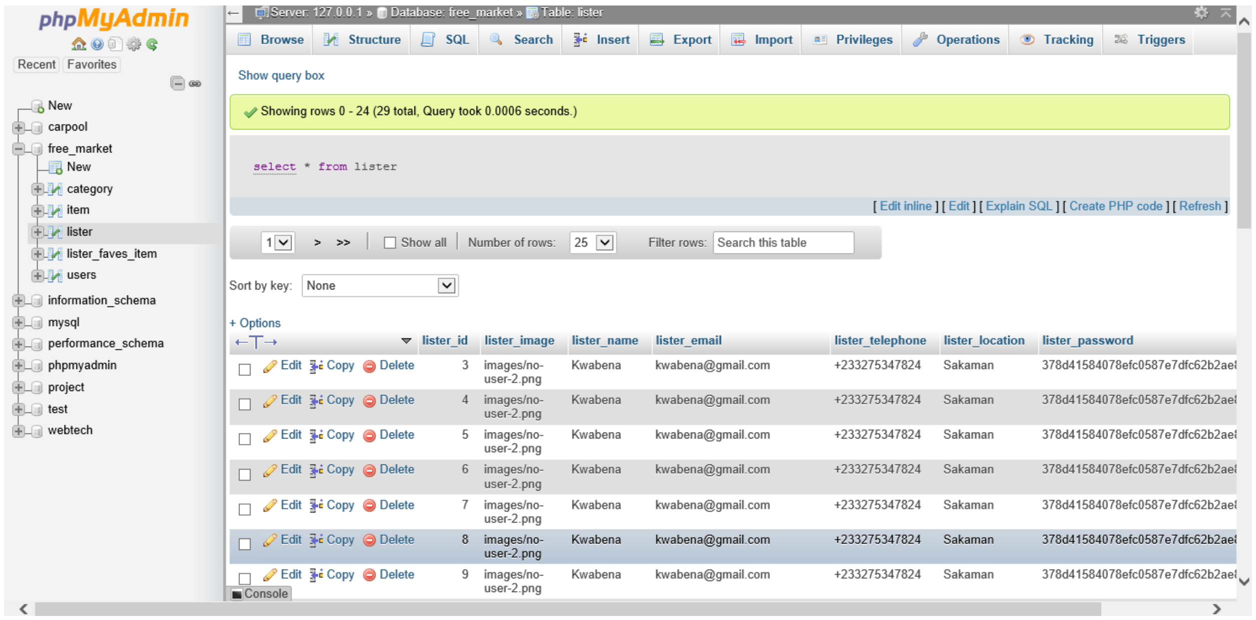
Functionality	**Expected Result**	**Actual Result**
Add User	User's credentials are added to	User's credentials are added to

	the database	the database
Login	User is redirected to dashboard	User is redirected to dashboard
Get Profile	User's account details are displayed	User's account details are displayed
Edit Profile	User's account details are changed	User's account details are changed
Get Locals	Users with a similar location are displayed	Users with a similar location are displayed

5.3 Component Testing

The components that make up the system were tested after establishing the absence of individual unit faults.

The database was tested using phpMyAdmin, a web interface for MySQL. SQL queries were executed and the returned results compared with expected results. Figure 5.1 shows the results of a database test.



Showing rows 0 - 24 (29 total, Query took 0.0006 seconds.)

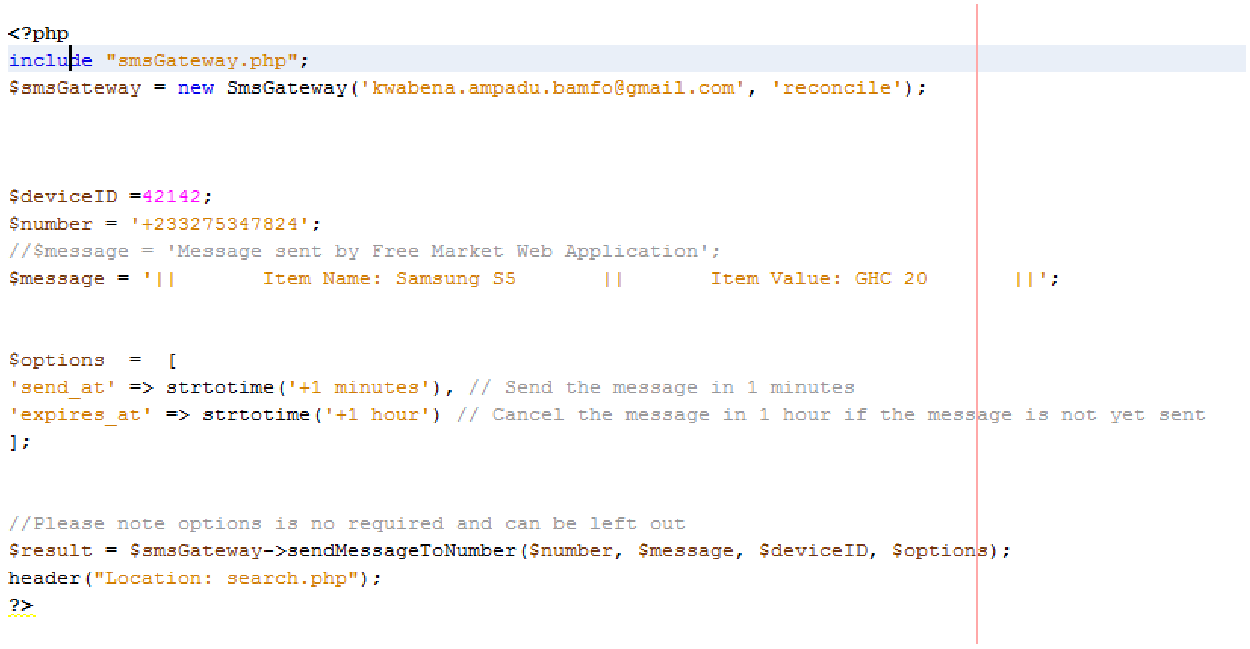
```
select * from lister
```

lister_id	lister_image	lister_name	lister_email	lister_telephone	lister_location	lister_password
3	images/no-user-2.png	Kwabena	kwabena@gmail.com	+233275347824	Sakaman	378d41584078efc0587e7dfc62b2ae
4	images/no-user-2.png	Kwabena	kwabena@gmail.com	+233275347824	Sakaman	378d41584078efc0587e7dfc62b2ae
5	images/no-user-2.png	Kwabena	kwabena@gmail.com	+233275347824	Sakaman	378d41584078efc0587e7dfc62b2ae
6	images/no-user-2.png	Kwabena	kwabena@gmail.com	+233275347824	Sakaman	378d41584078efc0587e7dfc62b2ae
7	images/no-user-2.png	Kwabena	kwabena@gmail.com	+233275347824	Sakaman	378d41584078efc0587e7dfc62b2ae
8	images/no-user-2.png	Kwabena	kwabena@gmail.com	+233275347824	Sakaman	378d41584078efc0587e7dfc62b2ae
9	images/no-user-2.png	Kwabena	kwabena@gmail.com	+233275347824	Sakaman	378d41584078efc0587e7dfc62b2ae

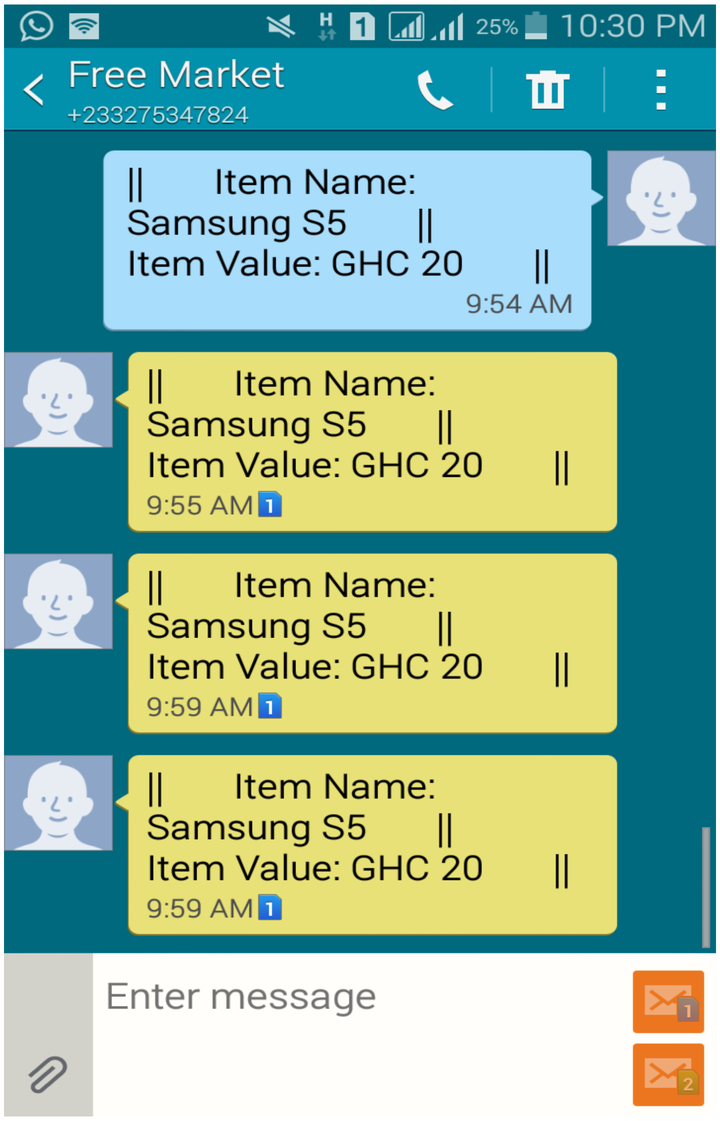
Figure 5.1: Component testing - Database

The applications interfacing with other components were also tested on a variety of browsers. The web platforms use of Google Maps was verified on Internet Explorer, Google Chrome and Firefox. All three browsers responded favourably with an accurate rendering of the map. The web platforms integration with SMS Gateway, an android mobile application that serves as an SMS gateway was also tested with a script developed by the API providers and modified to suit this web application. The script is shown in Listing 5.3 with corresponding results shown in Figure 5.2 below detail the test code employed and the test results returned.

Listing 5.3: Test code for SMS Gateway API



```php
<?php
include "smsGateway.php";
$smsGateway = new SmsGateway('kwabena.ampadu.bamfo@gmail.com', 'reconcile');



$deviceID =42142;
$number = '+233275347824';
//$message = 'Message sent by Free Market Web Application';
$message = '||       Item Name: Samsung S5       ||       Item Value: GHC 20       ||';


$options  =  [
'send_at' => strtotime('+1 minutes'), // Send the message in 1 minutes
'expires_at' => strtotime('+1 hour') // Cancel the message in 1 hour if the message is not yet sent
];


//Please note options is no required and can be left out
$result = $smsGateway->sendMessageToNumber($number, $message, $deviceID, $options);
header("Location: search.php");
?>
```



|| Item Name: Samsung S5 || Item Value: GHC 20 ||
9:54 AM

|| Item Name: Samsung S5 || Item Value: GHC 20 ||
9:55 AM

|| Item Name: Samsung S5 || Item Value: GHC 20 ||
9:59 AM

|| Item Name: Samsung S5 || Item Value: GHC 20 ||
9:59 AM

Figure 5.2: Test result for SMS Gateway API

5.4 System Testing

The system as a whole was tested based on observation due to the fact that the web platform is currently hosted on a local server. From this test, it was observed that the application was without any significant latency particularly due to the widescale adoption of AJAX in development.

The ability of the system to resist excessive load in a short period of time was also tested by adding a large number of items and users on the web application. This had no effect on the normal functioning of the application due to pagination principles employed. It was concluded that the web platform scales appropriately without any functional challenges.

5.5 User Testing

The application was also tested by a technologically competent user. The user described the web application's user interface design as intuitive. The user also managed to navigate to all available pages with no supervision. With respect to the functionality that allows users to locate items on Google Maps, the user had reservations about allowing the web platform to access his location, which is a key consideration for the functionality. The user was also particularly impressed with the web platform allowing users with and without an account access to key features of the web application.

6. Chapter 6: Conclusions and Recommendations

6.1 Summary

The web platform developed and documented in this write-up successfully satisfies the requirements listed at the beginning of the document. The application allows users primarily to add items to an online marketplace as well as view items added to the marketplace. The application also allows users to manage items and manage their display of the online marketplace. The project however does need reinforcements in its security and privacy should it be considered for real-time production.

6.2 Limitations

The project meets the functional requirements; however, it is not without notable limitations. This section describes the limitations of the web platform and details on flaws in its design.

- **AJAX:** The project was developed with consideration for Asynchronous JavaScript and XML (AJAX). AJAX has grave consequences on a web browsers back button. This means users are unable to recover the state of a previously visited page when they leave the page and return later.
- **Localhost:** The web platform is currently run on a locally hosted Apache HTTP Server. This makes it virtually impossible to obtain a true reflection of the extent to which non-functional requirements are met. The absence of a public web server in the project implementation can also be cause for problems in the future should the source code be migrated to a public server, since major changes would have to be implemented

- **Database Normalization:** The database on which the application is built is not in the third normal form. This can lead to database failures in the future. Normalization is an essential part of relational database management systems. The applications unnormalized database limits the scale to which the application can grow.
- **Security Concerns:** The web platform integrates with Google Maps and requests to access the location of users with their permission. The web application however is not hosted on a secure server (HTTPs) and this means users location can be exfiltrated by malicious users for dubious reasons.
- **SMS Gateway:** The project makes use of a mobile application on an Android device as an SMS Gateway. Although considerably cheaper, this choice limits the scalability of the application.

6.3 Future Work

Despite satisfying set out functional requirements, it is possible to improve the service provided by the web platform in the near and extended future. Ways to do this include but are not limited to:

- Develop a mobile application on the back of the web platform: An increasing number of users are mobile today and may prefer to interact with such a service via their mobile phones rather than personal computers. Building a cross platform mobile application will enable users to interact with the service no matter where they find themselves.
- Data mining of platform generated data: As more users adopt the concept of free recycling through the web platform, it is possible to mine the data generated through frequent use of the web application. This can be used to determine giving patterns among

a group of people, locality or neighbourhood. These conclusions derived from the data can significantly aid in the web platform’s design and structure among other things.

6.4 Conclusion

As a nation, it is quintessential that we develop an attitude of giving. Applications such as is described in this document go a long way in encouraging this behaviour by simplifying the process of giving and receiving. It is important to use technology in such a savvy manner in attempt to solve the challenges associated with charity. This form of giving however is not entirely based on inculcating a habit of generosity. In addition, it serves as a means of recycling usable goods and keeping them out of landfills. This project therefore also has an underlying environmental impact.

7. References

Agile Modeling. (2014). *Feature driven development (FDD) and agile modeling.* Retrieved from http://agilemodeling.com/essays/fdd.htm

Android Studio. (2017). *Android studio: The official IDE for android.* Retrieved from https://developer.android.com/studio/index.html#features

Apache Friends. (2017). *What is XAMPP.* Retrieved from https://www.apachefriends.org/index.html

Baldota, P. (2011). *What is AJAX.* Retrieved from http://www.pritambaldota.com/what-is-ajax/

Beal, V. (2017). *HTTP: Hypertext transfer protocol.* Retrieved from http://www.webopedia.com/TERM/H/HTTP.html

Brainvire. (2017). *Six benefits of using mvc model for effective web application development.* Retrieved from https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/

Engel, K. (2016). *Have a website: You need a privacy policy: Here's why.* Retrieved from http://www.webhostingsecretrevealed.net/blog/blogging-tips/have-a-website-you-need-a-privacy-policy-heres-why/

Fermoso, J. (2009). *Phonegap seeks to bridge the gap between mobile app platforms*. Retrieved from https://gigaom.com/2009/04/05/phonegap-seeks-to-bridge-the-gap-between-mobile-app-platforms/

Freecycle.net. (2017). *How to freecycle.* Retrieved from http://www.freecycle.net/

Freecycle.org. (2016). *Introduction.* Retrieved from https://wiki.freecycle.org/Introduction

Git. (2017). *Git: Distributed is the new centralized.* Retrieved from https://git-scm.com/

Google. (2017). *Google map apis.* Retrieved from https://developers.google.com/maps/

jQuery. (2017). *jQuery: write less, do more.* Retrieved from https://jquery.com/

Maria DB Foundation. (2017). *About MariaDB.* Retrieved from https://mariadb.org/

Materialize. (2014). *Materialize: A modern responsive front-end framework based on material design.* Retrieved from http://materializecss.com/

NetBeans. (2017). *NetBeans IDE: Fit the pieces together.* Retrieved from https://netbeans.org/

phpMyAdmin. (2003). *About.* Retrieved from https://www.phpmyadmin.net/

Phonegap. (2016). *About: A high level summary of what Phonegap is all about.* Retrieved from http://phonegap.com/about/

Review Journal. (2017). *31 best websites for free stuff.* Retrieved from http://www.reviewjournal.com/business/money/31-best-websites-free-stuff

SMS Gateway. (2017). *Your android phone: Your sms gateway.* Retrieved from https://smsgateway.me/

SMS Gateway API. (2017). *Behind the scenes.* Retrieved from https://smsgateway.me/what-is-sms-gateway-api

The Scotsman. (2006). *Forget about eBay: Here's freebay*. Retrieved from http://www.scotsman.com/lifestyle/forget-about-ebay-here-s-freebay-1-1410498

Trash Nothing. (2017). *Same free recycling groups: New and improved interface.* Retrieved from https://trashnothing.com/upgrade

Trash Nothing. (2017). *An easier way to run freecycling groups.* Retrieved from https://trashnothing.com/hosting?r=modfaq1

Trash Nothing. (2017). *Moderator FAQ.* Retrieved from https://trashnothing.com/moderator_faq#why-would-people-want-to-use-trash-nothing-instead-of-the-yahoo-groups-web

Wikipedia. (2017). *Yahoo! Groups.* Retrieved from https://en.wikipedia.org/wiki/Yahoo!_Groups

Wikipedia. (2017). *Web server.* Retrieved from https://en.wikipedia.org/wiki/Web_server

Wikipedia. (2017). *Apache HTTP server.* Retrieved from https://en.wikipedia.org/wiki/Apache_HTTP_Server#cite_note-10

Wikipedia. (2017). *PHP.* Retrieved from https://en.wikipedia.org/wiki/PHP

Wikipedia. (2017). *XAMPP.* Retrieved from https://en.wikipedia.org/wiki/XAMPP#cite_note-x_mariadb-3

Wikipedia. (2017). *Freecycling.* Retrieved from https://en.wikipedia.org/wiki/Freecycling

Wikipedia. (2017). *Privacy policy.* Retrieved from https://en.wikipedia.org/wiki/Privacy_policy

W3Schools. (2017). *PHP MySQL database.* Retrieved from https://www.w3schools.com/php/php_mysql_intro.asp

W3Schools. (2017). *AJAX introduction.* Retrieved from https://www.w3schools.com/xml/ajax_intro.asp