Final Report

Level 4

Sappu Savari - Location Based Advertising & Marketing

I S Dewasurendra
139160M

Supervised by: Mr Saminda Premaratne
Faculty of Information Technology
University of Moratuwa
2015

Final Report

Level 4

Sappu Savari - Location Based Buying & Selling System

I S Dewasurendra
139160M

Supervised by: Mr Saminda Premaratne
Faculty of Information Technology
University of Moratuwa
2015

Declaration

I declare that this dissertation on my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education to the best of my knowledge and belief. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references in given I also hereby give consent for my dissertation. If accepted, to be made available for photocopying ad for interlibrary loans, and for the title and summary to be made available to outside organizations.

I S Dewasurendra

……..
Signature

Date: …….

Supervised by,

Mr Saminda Premaratne

……..
Signature

Date: …….

Acknowledgment

First, I express my heartfelt appreciation and gratitude to my supervisor, Mr Saminda Premaratne for his most valued guidance and commitment to make this project successful.

Also sincere appreciation is extended to Dean of the faculty of Information Technology of University of Moratuwa, Dr. Lochandraka Ranathunga.

My sincere thanks also goes to my parent, my wife Chamilka and her parents for their continuous support, encouragement and cooperation extended to me to make my effort's success.

Last but not least my office colleagues Hasara, Nadeeshani, Thilina, Thejavi and Danushka who helped me in difficult time by providing continuous support and encouragement to successfully complete this project.

Finally, I thank all who helped me.

Abstract

There are many different advertising media, all of which serves different purposes. Advertising helps buyers to find sellers and sellers to sell their goods and services for buyers. In early stages, people used to use print media such as newspapers, flyers and notices to buy and sell industry. But later with the tremendous success of the internet, evolution of advertising begins. The World Wide Web, mobile communication and different kinds of software has introduced to the market. At present these electronic advertising methods are more popular and built a huge audience through the internet. Later World Wide Web integrated with mobile communication and became the main advertising media to buy and sell.

When the systems get expanded the number of posts of advertisements will increase at higher rates. Users have to spend several hours searching for exact good and services through the internet. Therefore systems integrated with some advanced search functionalities to help find the best match, but still users have to keep searching every time since new advertisements are popping up in every second.

'Sappu Savari' is the new proposed system which is more helpful and interactive to the users. People can easily use this new system not only in desktops and laptops but also in their devices like mobile phones, tablets. The main purpose of the mobile related technology is to satisfy user needs to access information and services, including Location Based Services (LBS) anywhere anytime. The system will keep searching automatically, match with the given details and notify the best match to for the users. The user doesn't need to browse go through the available lengthy list of advertisements and find the best matching goods and services. While the user (buyer/seller) uses the system, it will always track the location of the user and notify nearby goods and services with relevant buying and selling details. This will make easy for users to contact the other party and could be able to get the exact services or good's location using the generated map route.

This concept will help users to avoid unnecessary time waste. The system will be able to track down all the advertisements according to the given required details and provide a reliable and interactive way of tracking the locations according to the advertisements. The system will be read the locations by the available sensors of the devices have been used by the user. Nearby locations will be notified to the users and give alerts with summarized details. This will allow users to find and meet the buyers or sellers on their way.

The system will be automated to search advertisements and tracking the locations according to the available details.

The technologies adopted for this design are based on open standards and designed aiming to successfully meet the requirements of the user needs.

Table of Contents

Chapter 1 1

Sappu Savari - The Next Level of Mobile Marketing 1

1.1 Introduction 1

1.2 Background 2

1.3 Aim and Objective 4

1.4 Proposed System 5

1.5 Summary 7

Chapter 2 9

Existing Issues in Internet Based Advertising Systems 9

2.1 Introduction 9

2.2 Existing Location Based Systems 9

2.3 Summary 16

Chapter 3 21

Beyond Internet Advertising and Marketing 21

3.1 Introduction 21

3.2 Mobile Communication – Beyond Internet Marketing 21

3.3 Summary 24

Chapter 4 25

Sappu Savari – The Revolutionary Concept of Advertising 25

4.1 Introduction 25

4.2 Approach 25

4.3 Summary 28

Chapter 5 29

Sappu Savari – The New Mobile Oriented Architecture 29
5.1 Introduction 29
5.2 Analysis and Design 29
5.2.1 Scope 29
5.2.2 Design Considerations 30
5.2.3 Detailed Software Architecture and Technology Stack 32
5.2.4 Detailed Component Design 34
5.2.5 Detailed Program Sequence 34
5.2.6 Detailed Program Specification 35
5.2.7 Deployment View 36
5.2.8 Implementation Model 36
5.3 Summary 37
Chapter 6 38
Sappu Savari – Implementation 38
6.1 Introduction 38
6.2 Implementation of the Sappu Savari 38
6.2.1 User Authentication 38
6.2.2 Detect User Location 39
6.2.3 Add Product Items 40
6.2.4 Search Products 41
6.2.5 Notifications 42
6.2.6 Messaging Platform 43
6.2.7 Product Rating 44
6.2.8 Comments 45
6.2.9 Other 45

6.3 Summary 46
Chapter 7 47
Evaluation of the System 47
7.1 Introduction 47
7.2 Evaluation and Testing of the Project 47
7.3 Evaluation of Sappu Savari 48
Chapter 8 54
Discussion 54
7.1 Introduction 54
7.2 Whether the project goal achieved? 54
7.3 Problem encountered and limitation 55
7.4 Future work 55
Currently the messaging platform functions as it's intended to use, but in the future this messaging platform can be used as a chat platform. Then the users can chat with each other to communicate efficiently and effectively. 56
Integrate Auto Suggessions 56
7.3 Summary 57
Chapter 8 58
References 58
Chapter 9 60
Appendix 60
9.1 Appendix A – Sequence Diagrams 60
9.1.1 Login User 60
9.1.2 Post Advertisement 60
9.1.3 Search Advertisement 61
9.1.4 Rank Advertisement 61

9.1.5 Comment on Advertisement 62
9.1.6 Mark Favorite Advertisement 62
9.1.7 Send Messages 63
9.2 Appendix B – Class Diagram 64
9.3 Appendix C – ER Diagram 65
9.4 Appendix D – Screen Shots 66
9.4.1 Computer Screen Shots 66
9.4.2 Mobile Screen Shots 71
9.5 Appendix E – Source Code 82
9.5.1 Back End Java Code 82
9.5.1.1 Product Domain 82
9.5.1.2 Product Category Domain 88
9.5.1.3 Product Sub Category Domain 89
9.5.1.4 Product Multimedia Domain 91
9.5.1.5 Product Search Request Domain 92
9.5.1.6 Notification Domain 98
9.5.1.7 Rating Domain 100
9.5.1.8 Message Domain 102
9.5.1.9 Comment Domain 104
9.5.1.10 User Domain 105
9.5.1.11 User Role Domain 111
9.5.1.12 Product Service 112
9.5.1.13 Product Service Implementation 113
9.5.1.14 Product Category Service 117
9.5.1.15 Product Category Service Implementation 117

9.5.1.16 Product Multimedia Service 118
9.5.1.17 Product Multimedia Service Implementation 118
9.5.1.18 Location Service 118
9.5.1.19 Location Service Implementation 119
9.5.1.20 Search Request Service 119
9.5.1.21 Search Request Service Implementation 120
9.5.1.22 Rating Service 121
9.5.1.23 Rating Service Implementation 121
9.5.1.24 Comment Service 123
9.5.1.25 Comment Service Implementation 123
9.5.1.26 Message Service 124
9.5.1.27 Message Service Implementation 125
9.5.1.28 User Service 126
9.5.1.29 User Service Implementation 126
9.5.1.30 Scheduler Service Implementation 127
9.5.1.31 Product Data Access Service 131
9.5.1.32 Product Category Data Access Service 133
9.5.1.33 Product Sub Category Data Access Service 133
9.5.1.34 Product Multimedia Data Access Service 133
9.5.1.35 Search Request Data Access Service 134
9.5.1.36 Rating Data Access Service 134
9.5.1.37 Comment Data Access Service 135
9.5.1.38 Message Data Access Service 135
9.5.1.39 User Data Access Service 135
9.5.1.40 Login Success Service 136

9.5.1.41 Login Failure Service 137
9.5.2 Front End HTML/JAVASCRIPT Code 138
9.5.2.1 Home Page..... 138
9.5.2.2 Follow Seller..... 140
9.5.2.3 User Location Identification..... 142
9.6 Appendix F – Evaluation and Testing..... 146

List of Figures/Tables

Figure 1: Table of Feature Comparison 17
Figure 2: Table of User Roles and Duties 29
Figure 3: Software Architecture 32
Figure 4: Table of Available Components 33
Figure 5: Technology Stack 33
Figure 6: Component Design 34
Figure 7: Deployment View 36
Figure 8: Table of Package Structure 36
Figure 9: Table of Folder Structure 37
Figure 10: Observation summary – Information provided in the application 49
Figure 11: Observation summary – User satisfaction on interfaces 50
Figure 12: Observation summary – Usability 51
Figure 13: Observation summary – Ease of Learning 52
Figure 14: Observation summary – Overall Impression 53
Figure 11: Login User - Sequence Diagram 60
Figure 12: Post Advertisement - Sequence Diagram 60
Figure 13: Search Advertisement - Sequence Diagram 61
Figure 14: Rank Advertisement - Sequence Diagram 61
Figure 15: Comment on Advertisement - Sequence Diagram 62
Figure 16: Mark Favorite Advertisement - Sequence Diagram 62
Figure 17: Send Messages - Sequence Diagram 63
Figure 18: Class Diagram 64
Figure 19: ER Diagram - Database Level 65
Figure 20: Home 66
Figure 21: Follow User 67
Figure 22: Add Product 67
Figure 23: My Store 68
Figure 24: Search Product 68
Figure 25: View Product 69

Figure 26: Search Results 70
Figure 27: Send Messages 70
Figure 28: User Profile 71
Figure 29: Login 72
Figure 30: Home 73
Figure 31: Follow Seller 74
Figure 32: Home Menu 75
Figure 33: Add Product 76
Figure 34: View Product 77
Figure 35: View Product Images 78
Figure 36: Message Dashboard 79
Figure 37: Create Messages 80
Figure 38: User Profile 81
Figure 39: Observation Summary – Information provided in the application 150
Figure 40: Observation Summary – User satisfaction on interfaces 151
Figure 41: Observation Summary – Usability 152
Figure 42: Observation Summary – Ease of learning 153
Figure 43: Observation Summary – Overall Impression 154

Abbreviation

LBS – Location Based Services

GPS – Geo Positioning System

ATM – Automated Teller Machine

SMS – Short Message Service

IT – Information Technology

API – Application Programming Interface

PDA – Personnel Digital Assistance

NFC – Near Field Communication

WAP – Wireless Application Protocol

GPRS – General Packet Radio Service

HSDPA – High Speed Download Packet Access

IP – Internet Protocol

MVC – Model View Controller

AJAX – Asynchronous JavaScript and XML

CSS – Cascade Style Sheet

DAO – Data Access Object

JVM – Java Virtual Machine

UML – Unified Modeling Language

Chapter 1

Sappu Savari - The Next Level of Mobile Marketing

1.1 Introduction

These days new smartphone applications all seems to want the same thing from us—our latitude and longitude. Three-quarters of America's smartphone owners use their devices to retrieve information related to their location, driving directions, dining suggestions, weather updates, the nearest ATM. Such location data are a boon to advertisers, who use the information on our movements to discern our habits and interests, and then target ads to us [1]. With the addition of GPS and cellular technology, advertising can make more interactive to the users.

Both advertising and marketing software systems don't use this Geo location information to trace users. There is few systems uses Geo location specific for a shop or a place not for a user. This means every seller should have a shop or a place for marketing or advertising. But in the real world, people sell their old electronic items, vehicles, books through these systems and they don't have an exact Geo location. In this scenario almost every advertising system failed to provide a proper Geo location based marketing. But newly proposed "SappuSavari" system is capable of tracing individual users who are acting as sellers or buyers in the system perspective and provide notifications when there are advertisements that match and nearby.

The new proposed system is more helpful and interactive to the users for advertising. People can easily use this new system in their mobile phones, tablets or laptop computers. The main promise of the mobile related technology is to satisfy user needs for anywhere, anytime access to information and services, including Location Based Services (LBS) [2]. The system will keep searching automatically, match with the given details and notify the best match to the users. The user doesn't need to browse through the long list of available advertisements and find the best matching goods or service. While user

(buyer/seller) on the move system always tracing the location of the user and notify nearby best matching goods and services with relevant details. The user will be able to easily contact the other party and could be able to meet using generated routes on a map.

This concept will leave the user free and save a lot of time. The system will be able to track down all the advertisements according to the given required details and provide a reliable and interactive way of tracking the locations according to the advertisements. The system will be reading the locations of the available sensors of the devices which are used by the users. Nearby locations will be notified to the users and give alerts with the summarized details. This will allow users to find and meet the buyers or sellers on their way.

1.2 Background

Location aware advertising opens up new opportunities for brands to place their advertisements in front of consumers at or near the place where they can obtain them; thanks to the ubiquitous nature of mobile. Basically, if an advertisement server is made aware of a user's location, then a decision can be made to deliver an advertisement based on either the user's current location, or a history of where they have been, coupled with a user profile of their interests.

While the often used "Starbucks example" – you're walking past a Starbucks and you get an SMS for a free coffee, it is unlikely that this would ever work in practice. Importantly in this example, Starbuck don't discount their coffee, or have a loyalty program and in cities like London they are simply everywhere [3].

A more likely another use case which location services could be used for a movie chain. They most likely will have a movie club and a list of subscribers who have opted in to receive movie news and discounts. If the movie chain wished to fill seats on a slow Thursday afternoon, they would want to know how many of these opted-in members are near one of their cinemas. Importantly, they do not want to know if they are actually at

the cinema (then there is no point in sending them an offer as they are already there seeing a movie), but if they are close enough, and the offer is compelling enough (e.g. 50% off any movie in the next 30 minutes) then they are more likely to respond to the offer – creating a very targeted, and instantly successful marketing campaign.

As we know most of companies put there step forward to begin marketing and advertising via mobile phones. So this area is the best to use mobile based technologies to use in our day to day life. Smart phones became very common and mobile internet became a mandatory for the human life. People tend to use mobile phones for everything they want. In this scenario location based services are becoming mandatory for advertising systems. But in real life there are so many obstacles to use location based services, but there are ways which we can use them reliably and efficiently.

There are few systems available with LBS (Location Based Service) used for advertising purposes in some countries.
They are,

- 'Closs 5' is a consumer to consumer online marketing application developed by EBay. This application allow user to find consumers in their region and online chat with the other party regarding the trade. Even though this closs 5 application uses LBS on their system, they used it to categories the goods and services not to trace the consumers [4].

- 'Yardsale' is an iOS location-based app made buying and selling easily. This application allows users to post advertisements for goods and services to buy or sell. It shows what type of goods and services around buyer's location in a list view. This application detects only the fixed locations and doesn't provide any information to find the other trade party by route details using maps [5].

Also, there are some advertising systems within Sri Lanka, which are very famous in Sri Lankan society.

They are,

- Ikman.lk – users can post advertisements based on different product categories. Advertisements are categorized using locations. Users can search advertisements on specific locations such as Galle, Colombo, Dehiwala, etc. Still location is a constant and only uses for categorize the advertisements.
- HitLanka – users can post advertisements just like the previous system. Advertisements are categorized by the locations and don't provide any location based services.

As you can see location based services are not being used for advertising and marketing systems in Sri Lanka.

By reviewing the existing systems and their missing functionalities are well identified and have a very good approach by using the new technological advantage to the new proposed system.

1.3 Aim and Objective

The aim of this newly proposed system design is to minimize the user involvement and automate the advertisement searching and trace the buyer/seller using the Location services available on mobile devices. This will reduce the time consuming tasks and allow using the services more efficient and reliable.

Sappu Savari will be provided efficient and reliable service to every type of buyers and sellers by reaching below objectives,

- Providing user friendly web/mobile client to access the system functionalities such as posting or reading buyer/seller information, ranking the seller, search options to find advertisements... Etc.

- Tracing the user location continuously using Location Based Services and provide relevant route details to find the buyer or seller.

- Minimize the time and effort of browsing through lists of available advertisements by providing an automated process of finding the best matching parties for the trade using available information in the system.

- Back end services with capabilities of handling huge amount of information related to marketing and its categories.

- Allow users to access the system with almost any devices with Internet capabilities.

- Automate the searching and provide continuous feedback according to the specific user's search request.

1.4 Proposed System

The proposed system is conceptually simple and amazingly efficient as a marketing service. Simply the most of the process is automated and users are seeing and visible advertisements accordingly. The services are very effective and most users are benefited by automated descriptive services.

The proposed system is supporting the services that described above and have the following characteristics:

- The system mainly uses web protocols which means system functions as a web oriented system via internet.

- The server side of the system is developed by Java and platform independent (can work on windows/Linux).

- The client side of the system is platform independent (can work on windows /Linux /android /apple/... etc.) and device independent (smart phone /tablet /laptops /workstations /... etc.)

- Various types of devices can be connected to the system according their internet capabilities. Availability of the sensors and performance system may vary the accuracy and the speed of the responses. Since the system services depend on the Geo location of the users; devices with Location services are recommended for higher accuracy and efficiency.

- Every user needs to create an authentication to save and retrieve relevant information.

- The system is developed over three tier architecture for support distributed database systems for better performance.

- Spring framework used for system development which is providing security, scalability and performance.

- Databases and Services developed and designed over Open Source Technologies such as Java, Jasper, Tomcat, etc.

- Uses geo location services on browsers to locate the users, so system can get the location even on IP based information(less accurate on IP based information and high accuracy when GPS is available [6])

- Users are free to use any type of browsers since it supports geo location API. Most of the existing browsers are compatible with the geo location API [7].

- Uses Google map API for generate maps and route [8].

The above system has been designed and implemented by following well defined standards. So the same design structure and the technologies can be used to against real world conditions.

The remainder of this paper is organized as follows. In the next chapter we will provide the basic information of the mobile marketing and existing tools. Next chapter includes the technological information and the way of approach to the existing mobile marketing systems with better suggestions. In chapter 4 includes my new approach to the internet based advertising. In chapter 5, we will provide the problem analysis and the design of the solution in detail which describe the usage of the available technological components work together to provide a better service to the users. Next chapter includes all the relevant information regarding the implementation of the system. In the last section includes my conclusion on my newly proposed concept and discuss what is going to develop in the future.

1.5 Summary

Location based services are used all over the world for many different software systems. Most of the advertising and marketing software systems also use location based services but only of market a land marked shop. The newly proposed systems main approach is to use location based services in a way of supporting users to meet each other to complete their trade.

Also automate the search advertisement process over user interests and pre searched information will save time and effort. Except above, there are few more newly proposed features in the system such as rank, comment, mark as favorite, etc.

In the next chapter, we will discuss about the few of existing advertising and marketing software systems in Sri Lanka and some other countries. Mainly the drawback of these systems and how our new approach overcomes these drawbacks.

Chapter 2

Existing Issues in Internet Based Advertising Systems

2.1 Introduction

This chapter will give you an idea about the existing internet based advertising software systems, How they advertise their advertisement how they manage to market those advertisements in the community within their system, How they used new technologies like mobile location services and other functionalities which supports the users to do their tasks easily. Again, this chapter will highlight the existing issues in those systems and how the new proposed concept solves over.

There are lots of existing advertising software systems around the world. Some of them are very famous and have thousands of users around the world. In Sri Lanka there are a lot of people who use these types of systems. Some IT related companies tend to build these types of systems and currently they are executing them successfully. In this chapter selected few systems in Sri Lanka as well as around the world to have a conclusion which area should miss in this application and how new systematic approach to those missing functionalities.

2.2 Existing Location Based Systems

2.2.1 Ikman.lk

2.2.1.1 What is Ikman.lk?

ikman.lk is a website where you can buy and sell almost everything. The best deals are often done with people who live in your own city or on your own street, so on ikman.lk it's easy to buy and sell locally. All you have to do is select your region.

It's completely free to publish a classified advertisement on ikman.lk, and it takes you less than 2 minutes. You can sign up for a free account and post advertisements easily every time.

ikman.lk has the widest selection of popular second hand items all over Sri Lanka, which makes it easy to find exactly what you are looking for. So if you're looking for a car, mobile phone, house, computer or maybe a pet, you will find the best deal on ikman.lk.

ikman.lk does not specialize in any specific category - here you can buy and sell items in more than 50 different categories. Ikman.lk also carefully reviews all ads that are being published, to make sure the quality is up to our standards [9].

2.2.1.2 Existing Issues in Ikman.lk

Even though there are a vast community already using the ikman. Lk internet advertising system for their advertising purposes, there are few issues exist which is not solved so far.
The few of them are as follows:

- The buyer cannot rate the products or give a comment about it.

- Users cannot share the advertisement over the social media; System doesn't provide the proper facilities for this.

- The buyer has to find the specific products using the search options over many categories. The system itself, not response for newly added advertisements after the search results retrieved. As an example a buyer finds a red color Yamaha 200cc motor bike in the morning, but he/she couldn't find any advertisement on the above type of motorbike. But in the evening a seller posts an advertisement on exactly the wanted type of motorbike. But the buyer is not notified and doesn't

know that an advertisement is already posted which is exactly matched. So the buyers have to search the product or service every time to find the best and newest advertisements. Due to this reason users waste much time to search their products and services.

- Advertisements are categorized over product type and address. So buyers search products or services on specific district or area. But using this constant buyers and sellers misses most of the matching advertisement all the time. As an example a seller in Jaffna post an advertisement to sell his car in the system under district Jaffna. A buyer from Colombo searches a same type of car in Colombo area; and Jaffna seller's advertisement never been picked for the search criteria. But if Jaffna seller moved to Colombo for two days, buyers in Colombo can easily can check the car and meet the seller, but unfortunately still Jaffna seller's advertisement doesn't pick for Colombo buyer's search criteria. This type of issues happens always and never been given a solution from existing advertising systems.

- Buyers and sellers are capable of post text and images for the advertisement. The seller can post his/her mobile number to contact him/her. But buyers have to allocate time and a place to meet the seller and check the product before buy. Due to this reason users have to spend a lot of time to meet the other party and check the product or service.

2.2.2 Oodle Marketplace – United States

2.2.2.1 What is Oodle Marketplace?

Using the power of social media, Oodle is reinventing online classifieds. Oodle provides consumers with a friendly local marketplace to buy, sell and trade. Oodle operates a

network of online marketplaces with more than 15M monthly unique users including the Oodle Marketplace.

Has something you are not using? Sell it or give it away! It's as easy as snapping a picture. Your items will instantly be posted on Oodle. See what your friends on Facebook are selling, giving away or looking for. Someone may have extra tickets for tonight's big show! Looking for something specific or a great deal? Quickly search through thousands of nearby listings from Oodle, Marketplace on Facebook and Craigslist.

Oodle Marketplace is currently available for the iPhone & Android [10].

2.2.2.2 Existing Issues Oodle Marketplace

Even though there are a vast community already using the Oodle marketplace in the UK as their internet advertising system for their advertising purposes, there are few issues exist.
The few of them are as follows:

- The buyer cannot rate the products or give a comment about it.

- Same as the ikman.lk software system advertisements are categorized over product type and address. So buyers search products or services to specific district/area. But using this district and area as a constant buyers and sellers misses most of the matching advertisement all the time.

- Buyers and sellers are capable of post text and images for the advertisement. The seller can post his/her mobile no to contact him/her. But buyers have to allocate time and a place to meet the seller and check the product before buy. Due to this reason users have to spend a lot of time to meet the other party and check the product or service.

2.2.3 HitLanka

2.2.3.1 What is HitLanka?

Hitlanka.com is no.1 free online classified ads website in Sri Lanka, a place where people can associate with one another to buy or sell products and services free in Sri Lanka and globally as well. Our website was launched in 2008 with the vision for buyers and sellers to “meet online, finalization offline”, today HitLanka have over thousands of listings.

Headquartered in Colombo, Sri Lanka, hitlanka.com is accessed by more than 75% unique users and thousands of new customers every week. HitLanka, now have over 27 categories and of them are, HOUSE FOR RENT | HOUSE FOR SALE | REAL ESTATE | RENT A CAR | AUTOMOBILE | TOURS/TRAVEL | FASHION/BEAUTY | HEALTH/FITNESS |FURNITURE | COMPUTERS AND ACCESSORIES | PHONES | OTHERS | MOTORBIKES | ANTIQUES AND COLLECTABLES | SPECIAL PROMOTION | SPORT GOODS | KIDS’ TOYS |JOBS | FOOD & AGRICULTURE | PETS | EDUCATION | SERVICES | HOME & PERSONAL ITEMS | CLOTHES, FOOTWEAR, JEWELLERY & ACCESSORIES | ELECTRONICS | MOVIES, BOOKS MAGAZINES|TICKETS| HOBBY & LEISURE.

At hitlanka.com, HitLanka have formed an online community which is simple and safe. HitLanka is continuously modernizing so that users can buy and sell in the easiest and most suitable way possible. It's totally free to post classified ads on hitlanka.com, and it takes you less than 2 minutes. To make the visibility of your advertisement high for our users, paid ads are also available on hitlanka.com [11].

2.2.3.2 Existing Issues in HitLanka

Even though there are a vast community already using the hitlanka.com as an internet advertising system for their advertising purposes, there are few issues exist which is not solved so far.

The few of them are as follows:

- The buyer cannot rate the products or give a comment about it.

- Buyer or sellers cannot share the product or services over the social media; the system is not providing proper facilities for this.

- The buyer has to find the specific products using the search options over many categories. The system itself, not response for newly added advertisements after the search results retrieved. As an example a buyer finds a red color Yamaha 200cc motor bike in the morning, but he/she couldn't find any advertisement on the above type of motorbike. But in the evening a seller posts an advertisement on exactly the wanted type of motorbike. But the buyer is not notified and doesn't know that an advertisement is already posted which is exactly matched. So the buyers have to search the product or service every time to find the best and newest advertisements. Due to this reason users waste much time to search their products and services.

- Same as the ikman.lk software system advertisements are categorized over product type and address. So buyers search products or services on specific district or area. But using this district and area as a constant buyers and sellers misses most of the matching advertisement all the time.

- Buyers and sellers are capable of post text and images for the advertisement. The seller can post his/her mobile number to contact him/her. But buyers have to allocate time and a place to meet the seller and check the product before buy. Due to this reason users have to spend a lot of time to meet the other party and check the product or service.

2.2.4 Marketplace

2.2.4.1 What is Marketplace?

marketplace.lk is a website where you can buy, sell and rent almost everything. All you have to do is select your region. It's completely free to publish a classified ad on marketplace.lk, and it takes you less than 2 minutes. You can sign up for a free account and post ads easily every time.

Marketplace. Lk is a fast growing website in Sri Lanka, that's why it easy to find exactly what you are looking for. So if you're looking for a car, mobile phone, house, computer or maybe a pet, you will find the best offers on marketplace.lk.

With marketplace.lk you can buy, sell and rent items in more than 40 different categories. Marketplace also carefully reviews all ads that are being published, to make sure the quality is up to our standards [12].

2.2.4.2 Existing Issues in Market Place

Even though there are a vast community already using the marketplace. Lk internet advertising system for their advertising purposes, there are few issues exist which is not solved so far.
The few of them are as follows:

- The buyer cannot rate the products or give a comment about it.

- Buyer or sellers cannot share the product or services over the social media; The system is not providing proper facilities for this.

- The buyer has to find the specific products using the search options over many categories. The system itself, not response for newly added advertisements after the search results retrieved. Due to this reason users waste much time to search their products and services.

- Same as the ikman.lk software system advertisements are categorized over product type and address. So buyers search products or services on specific district or area. But using this district and area as a constant buyers and sellers misses most of the matching advertisement all the time.

- Buyers and sellers are capable of post text and images for the advertisement. The seller can post his/her mobile number to contact him/her. But buyers have to allocate time and a place to meet the seller and check the product before buy. Due to this reason users have to spend a lot of time to meet the other party and check the product or service.

2.3 Summary

Most of the existing internet based advertising systems have the same kind of issues. These issues have not been resolved so far due to many reasons. So the newly proposed "Sappu Savari" addresses these issues with reliable and robust solutions. The concept behind the "Sappu Savari" is simple but efficient for every uses both seller and buyer.

Please find below common and well known issues and the way to address the same issue

with the newly proposed “Sappu Safari” mobile oriented advertising classification system:

Figure 1: Table of Feature Comparison

Common Issues	Approach of the newly proposed concept
The user doesn’t get notified when there are new matching advertisements available according to his/her search criteria.	This is very important; users always spend a lot of time to search the products or services he/she wants. Sometimes users browse or search the lengthy list over and over again to search the best matching products. What if a user gets relief from this and notifies users when there are any matching products or services. Automating this process will help the user to save a lot of time. “Sappu Savari” keeps the information of searches and save them as search requests in the system. Then an automated process is searching the available advertisements periodically to check whether there any matching advertisement to the user's search requests. If there any system provides notifications to the users. Rather than searching the advertisements, every time, users can search the product when there any notifications. This will help users to save their time and money to spend particular sellers.
Buyers don’t get notification when there are any sellers are available nearby. So chances of meeting the	This is also an important feature that must have on an advertising system. Most of the buyers need to meet the seller personally and check the device

seller personally miss.	before preceding the payments. According to the search request functionality in "Sappu Savari" system always keeps the information regarding the searches of specific users. System processes always keep following the existing advertisements to match the best matching advertisements. Also system keeps tracing the seller Geo location as well. When the sellers location is nearby for specific selected advertisements system gives notifications, and then seller can find the buyer using generated route maps to the buyer location. Using this proposed concept buyers can find sellers even when they travel, No need to call or email to the seller to allocate time to personally meet and check the product or services. This automated process saves a lot of time for buyers.
Keeping the product or service location as a constant will miss lots of opportunities to sell or buy products.	Existing systems store all the advertisements under several category types. One of the main categories is the location parameter. Advertisements are categorized and grouped according to the location. System search functionality always uses these groups to retrieve advertisements when searching for specific locations. So buyers such products or services on specific district or area. But using location as a constant buyers and sellers miss most of the matching

	advertisement all the time. As an example A car seller in Nuwara Eliya post advertisement in the system, buyer in Colombo searches cars in the Colombo area. But if the seller came to Colombo for any reason, still Colombo user cannot pick the Nuwara Eliya seller. The problem in this situation is systems keep the location as a constant; these systems couldn't dynamically change the advertised location due to unavailability of a way to identify the location of the seller/buyer. In "Sappu Savari" system always get the user's Geo location from the devices they logged in to the system. In the search algorithms this dynamically updated location is always picked. As an example, even seller post his/her advertisement in the Nuwara Eliya, if he/she is travelling to Colombo; systematically categorize the seller in to Colombo district and nearby locations. So buyers never miss any sellers who they nearby.
Cannot rank the product or services according to the user experience.	Most of existing advertising systems don't allow users to rank the seller or comment. So buyers cannot have a genuine idea about the sellers, if buyers can always take correct decisions before proceeding to buy the products and make the payments. So "Sappu Savari" allows users to rank the advertisements, and the higher ranks always have the priorities over other advertisements. Also, users

	can comment on specific products or services; this allows communicating between buyers before buying any products or services. Also, this concept will reduce the frauds. Sellers always try to do their best to maintain the customer satisfaction at the highest level; otherwise their business will be lost. Also, any buyer can have a general idea about the seller according gained rank and other user's comments.
Most of the systems are optimized and tuned for laptop computers and desktop computers, simply for bigger screens. But a bigger portion of the community uses mobile phones and tablets.	"Sappu Savari" is optimized for any mobile platform such as Android, Apple, Windows or Blackberry. Also the system is capable of viewing any screen size available in the devices. Also the system is capable of reading the Geo location information from any device through browsers, do depend on the device capability system can get more accurate Geo location information.

In the next chapter we will provide information about the type of technologies are available for the newly proposed concept. Mainly the way of communication via mobile and how they developed over the past years and what is the best to the design of the "SappuSavari".

Chapter 3

Beyond Internet Advertising and Marketing

3.1 Introduction

In the previous chapter, we discussed about the existing advertising systems and their drawbacks; and how the new system approach overcome those. In this chapter, we are going to discuss about how technological background behave related to the proposed system.

We are going to discuss about what type of technologies which new system used, how these technologies should be used and why those technologies are selected over other existing technologies.

3.2 Mobile Communication – Beyond Internet Marketing

Advances in wireless communication and information technology have made the mobile web a reality. Mobile web should respond to the need for anytime, anywhere access to information and services. Many software applications were deployed and available to customers via mobile phones and wirelessly connected tablets/PDAs. Except mobile web, services such as Location Based Services are promising technologies which are always providing important and interactive information to the users.

In 2000 Gravitate Inc. has published a white paper which identifies three evolution steps for Location Based Services [13]. The first generation of Location based system identified the location by getting the user input for location details. In the second generation user can be located with very little accuracy. The third generation refers to services where the position of the subscriber is automatically discovered with accuracy

and which have the intelligence to inform or warn the subscriber about events depending on his position.

Location based services can be applied to different categories of systems. The GSM Alliance service working group [14] has defined the following types of Location Based Services:

- Marketing
- Emergency
- Information Services
- Navigation
- Location Based Social Media
- Mobile Location-Based Gaming
- Sports
- Billing
- Geo Tagging
- Tracking
- Augmented Reality

3.2.1 Driving Forces for Location Based Services

Even in Sri Lanka, people tend to use smart mobile phones and tablets for their day to day work. They use different kind of available services such as mobile internet, location based services, NFC services, Etc. The newly proposed system is mainly using mobile internet and location based services available in the smart devices. The proposed system design will be very successful and much suitable to the current mobile market and time due to below reasons.

3.2.1.1 Market Force

Market researches show that most of the mobile consumers are ready and willing to pay for Location Based Services (e.g.: [15]). Some mobile subscribers are considering

changing their mobile phone operators in order to gain access to the Location Based Services. Most of the people pay for Location Based Services in some countries. But the new generations of smart devices have built in capabilities to access the Location Based Services and companies like Google provide free services for their consumers.

3.2.1.2 Competition Force

Mobile phone manufacturers such as Samsung, HTC, LG, Apple, Sony, Etc. always develop their mobile phones with LBS capabilities. In a current mobile phone market almost all the smartphones have integrated GPS sensors and LBS capabilities. The competitive mobile phone market always tries to manufacture their mobile phones and tablets to meet the high end processing power and memory levels and accomplish the proper user requirements such as mobile web capabilities, LBS access capabilities, etc.

3.2.1.3 Technology Force

The first location based services are expected or already offered to mobile phones via WAP or SMS. WAP and SMS were cheap and every GSM phones have the capabilities to use both. Next GPRS were very famous among GSM mobile subscribers and data communications are done using 9.6Kbps to 115 Kbps bandwidth speed. Later HSDPA, 3G, 4G technologies came to smart mobile devices and tablets and they provide higher bandwidth for data communication. Location Based Services are functioning in a more reliable way on top of HSDPA, 3G and 4G network communications. Also the services are enhanced with much detail and graphical contents parallel to the network mobile communication developing.

3.3 Summary

As you can see above facts, there are greater possibilities to use location based services for advertising and marketing. Technologies are well built and adapt with mobiles and it is never been easy to use software systems to manipulate and get the results.

But in real life every user cannot afford for the best technological advanced equipment. In our newly proposed system using a strategy to use different type of available technologies in several ways, e.g.: system is fully supported for GPRS, 3G or 4G and capable to detect Geo location using GPS, cell id, IP, Etc. Also system is platform independent since its architecture is browser based which the user can use the system on android, apple, Windows, Linux, blackberry almost on any platform.

In the next chapter, we will discuss the approach of the newly proposed system. About various kinds of information and data will be needed to serve a better service to the users. Also about new features need to address to overcome existing issues and provide and more user friendly advertising and marketing system.

Chapter 4

Sappu Savari – The Revolutionary Concept of Advertising

4.1 Introduction

In Previous chapter, we discussed about the technological advantage which we can use for the proposed concept and what type of technologies planned to use with the "Sappu Savari" advertising and marketing system. In this chapter includes the approach of the system. How and what type of functionalities and features used for the system.

4.2 Approach

There are several applications available for advertising, but the problem is; Are they really market the advertisement or just allow posting texts or several images in their applications. These types of services are not taking the advantage of the real power of user devices and internet. Some of the advertising and marketing systems don't support for mobile platforms, but globally the majority is using mobile phones according to latest mobile phone usage statistics [16].The advertisements are categorized according to the product type and the posted address. This categorization is not providing a proper benefit for the users. Users waste much time to find and search the relevant advertisement and no proper way to communicate to others. Also, there have been lots of fraud advertisements and most users are being fraud, there are no any way to communicate these frauds though users.

By analyzing above drawbacks my aim is to provide an IT solution to provide a better solution to the users with the help of existing technologies; A new concept of advertising and marketing in internet.

This new approach is a web oriented system. Users can access the system using their internet capable devices such as smart mobiles, tablets, laptop computers, Etc. Users can register in the system to post advertisements. Initially the advertisement will be categorized over posting address location and the type of advertisement just like the existing systems. This will help to break the advertisements into groups. Users act as sellers are capable of uploading images and post texts to create advertisement in the system. The posted advertisement should be posted under relevant product category, e.g.: electronics, automobile, Sport, etc. One user can post many advertisements under several product categories.

Most of the features are open for users who are acting as buyers. The system provides search facilities; users can simply give relevant details and search available advertisements. As an example a buyer who is interested in Toyota cars, can provide search parameters like white color, sedan type, manufacture year, brand new. Then the system will search list of matching advertisements, according to the search parameters. User can select one of the advertisements and go through given images and text description. If the user wants more details, he/she can check the comments on other interested buyers. Most probably someone well known of the same car might comment on the car saying “this car is in great condition, but there is an issue in the right head light”. So buyers always have a genuine status of the car. Also, if the buyer wants far more details he/she can simply send a message to the seller though the system. Once he/she gets a reply he/she can decide whether this car is worth buying or not.

In this level buyer have enough information to continue his search on this car. So buyer can mark this car as his favorite advertisement. Then the system starts to follow this advertisement flawlessly. If someone comments about it you will be notified, also the system will be tracing the seller’s location to check whether the seller or buyer is nearby.

In addition to the buyer can create requests as per his need. In our scenario buyer can create a search request, e.g.: white color, Toyota, sedan, brand new condition. Users can create multiple requests over different products or services. Now the system is open to

search more cars over available advertisement. But this search is done automatically by the system; buyer is not involved in searching now. System flawlessly searches the proper best matching advertisements trough out the existing advertisements; if there any buyer will be notified with descriptive information.

Above search requests are the key points of details for most of the automated processes exist in the proposed system. In addition to above details system also considers the marked favorite advertisements and higher ranked advertisements to its processes. These searched are notified to the users on priority based queue.

Except above features the most important concept is automated tracing the user locations. This allows the system to identify the user's location even when users are on travel. Locations are detected by the smart phones, tablets using their location services. The system is capable of tracing the location even if users use non GPS compatible devices since the browsers have the Geo location capabilities. The accuracy may vary according to the accessing device and availability of the internet connections. When the system successfully identifies the user's location it automatically maps the users who post the search requests and users who post advertisements. If the request and the advertisement are matching and the both users are in a shorter distance to each other, system notify immediately to both users saying matching buyer or seller exist nearby, also system is capable of generating maps with route to meet each other using Google maps API. Accordingly, both buyer and seller could meet up and discuss the trade; But both users never planned to meet each other at the first point, they were their daily routine of their life and suddenly the system provides notifications and they were able to meet each other to discuss the trade.

After discussing the trade buyer and seller end the relationship in the system. The seller could remove the advertisement from the system and system removes all the back links with the specific advertisements.

Except above, there are several features exist in the new concept of advertisement marketing application. Users act as buyers can rate the product, write comments about it. Then other users can easily make a decision over buying the specific product or service. Also, users can select specific product as favorite and a system always consider product ranks and favorite states when prioritizing the matching requests and advertisements.

4.3 Summary

Simply our main approach is a web based software system which is highly user friendly and device independent; enabling the users to access the system using any smart device with internet capabilities. Software system fetches users Geo location of the device, again accuracy depend on the users device and the map in to advertisements. Every time users are interacting with the system according to their age locations.

Again, the system has an automated process to search best suitable and match the advertisements with user's interest and pre-search details. The system can continuously provide notifications according to the users requested advertisements. This could save user's time and effort by browsing the system for flawless advertisements queues.

In the next chapter, we will discuss analysis and design part of the "SappuSavari". This will provide detailed information on the analysis and design of the system by using diagrams and charts. Also, different module exists in the system and the different technologies used for the design.

Chapter 5

Sappu Savari – The New Mobile Oriented Architecture

5.1 Introduction

In the previous chapter, we discussed about the approach to the system. Also, what types of features available for the system and how they serve to the users. This chapter will give a detailed idea about the architectural design of the system. The type of technologies used for development and deployment. The way of using the different services available in the mobiles will be described in detail using figures. Modules of the system will be described separately in this chapter over their technical perspective.

5.2 Analysis and Design

5.2.1 Scope

The functional scope of this design has two main roles. Multiple categories of users are kept under specific user roles.

Figure 2: Table of User Roles and Duties

User Role	Users	Duties
Admin	Admin User	• Review Advertisements • Review Comments • Review user logins • Create master data
User	• Seller • Buyer	• Login • Post Advertisements (text and images) • Search Advertisements • Rank advertisements • Comment advertisements

		• Mark Advertisement as favorite • Send Messages

5.2.2 Design Considerations

5.2.2.1 UI Considerations

- Model View Controller (MVC) Pattern – Spring framework front end controller has been selected.
- Reusable Sorting for tables – Tables and list sorting will be controlled via JavaScript, negating any need for server communication or page refresh for sorting.
- Internet Browser Support – The application will support Internet Explorer 9 and above/ Firefox 20 and above/ Google Chrome 28 and above. All UI features may not be supported in IE 6.0.
- A JQuery based AJAX framework will be used for populating data independent dropdown rather than refreshing the whole page.
- A bootstrap framework for support different size of screen sizes.
- Segregate Look and Theme – The application will have CSS (cascaded style sheet) for storing font, color, table styles, etc. This would certainly make the core application logic decoupled from application look and style.

5.2.2.2 Data Model Design

- The physical data model considers MySQL database.
- The model assumes soft deletion of entities compared to hard delete.

5.2.2.3 Uses of Design Pattern

A number of design patterns have been used. The Business component uses Data access Object (DAO) pattern, Factory Pattern, Singleton Pattern. The presentation layer uses MVC pattern.

5.2.2.4 Connection Pooling

The connection object is obtained by the spring data framework. The connection pool is used for obtaining database connections & those connections are released back to the pool after usage.

5.2.2.5 Transaction and Rollback

Connections will be fetched from connection pool in Business layer and passed as parameter to DAO layer. In case database exception occurs in DAO layer that will propagate up to Business layer and the transaction will be rolled back. So business classes will have entry and exit points of transactions. In case of transactions involving multiple DAO layer, the business layer ensures that same connection object is passed to all. Any create/update operation will be done in a single transaction – so that if anything goes wrong within a transaction, then the whole operation will be rolled back.

5.2.2.6 Concurrent Access

In "Sappu Savari" application, for use cases implementing mutex is not required. Version column will be used on all tables to handle concurrent updates and to avoid data being overwritten from other sessions.

5.2.2.7 Clustering Support

Business classes are stateless and singleton in nature. This application supports single instance or cluster environment deployment. Since business classes will generate a single stateless instance, in each JVM of clustering, it does not affect session affinity capabilities of the load-balancer, if used.

Also, all value object classes will implement Serializable interface to ensure no hindrance occurs if and when session replication is enabled.

5.2.2.8 Logging Facilities

Log4j logging framework will be integrated to capture a log into a file system for any kind of operation in the system. But in production, logging level would be set as ERROR to avoid performance deterioration because of capturing huge amounts of log.

5.2.2.9 Audit Facilities

Every transaction table will have two audit columns, namely “Created By”, “Created Date”, “Updated Date” – to track who has created/updated a particular entry in the table and when that is happening. Also for every transaction table, there will a corresponding audit table to capture all the audit trail data for any modification of data in the main table.

5.2.2.10 Security

The application is accessible only through a valid login with necessary privileges.

Connections to the applications shall be available only via https connectivity.

5.2.3 Detailed Software Architecture and Technology Stack

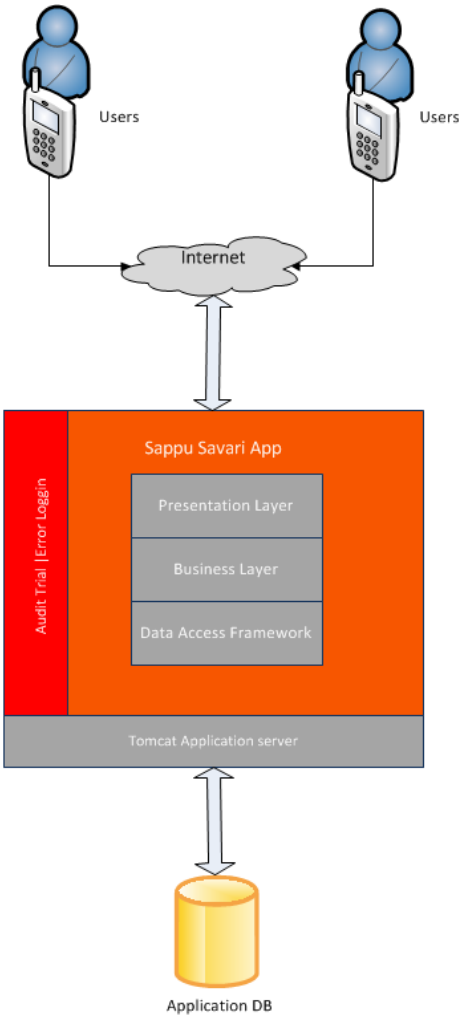

Figure 3: Software Architecture

Figure 4: Table of Available Components

Component	Description
Application	Accessed by authenticated Users.
Tomcat Web Server	Application server which hosts the application.
MySQL DB	Relational database used to store data captured in the business process.
File Repository	Used to store uploaded images like advertisement images/ product images/ User profile images.

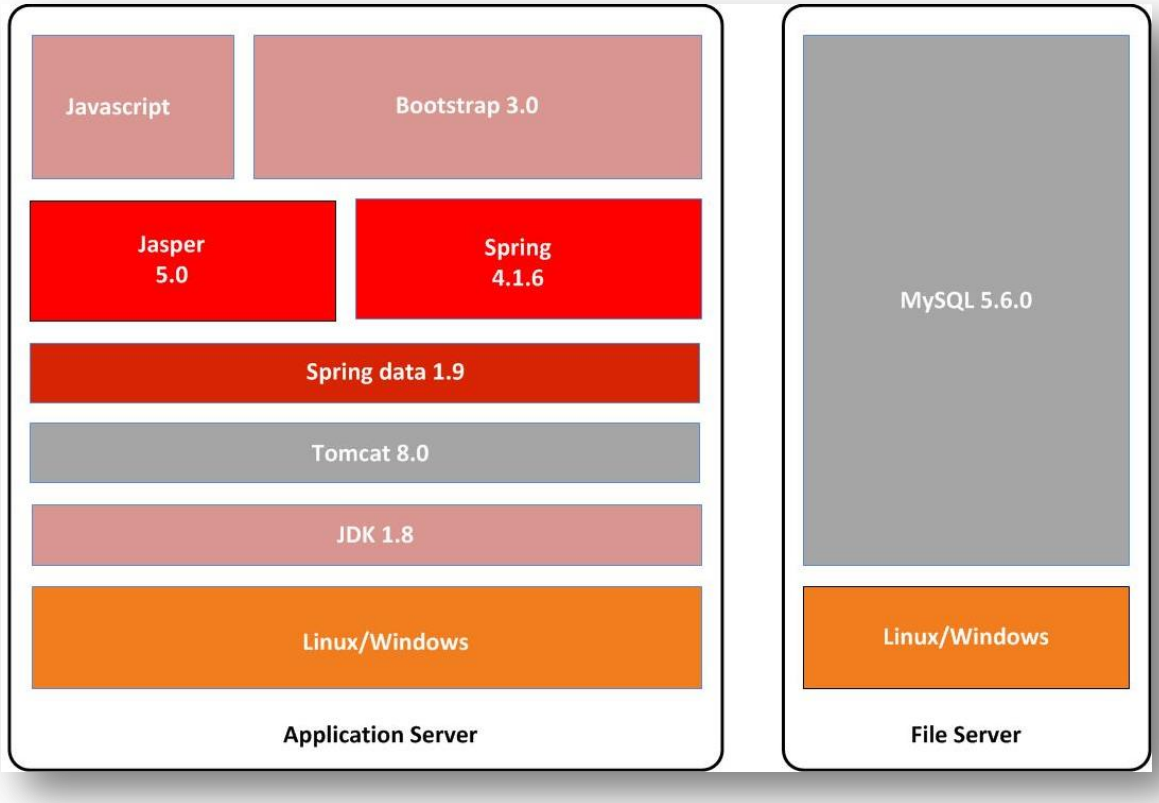

Figure 5: Technology Stack

5.2.4 Detailed Component Design

Component Design deals with designing different components of a software system in a layered architecture. It shows structural relationships between various components in a system. In “Sappu Savari” System, Component Design Diagram shows the different components of a module and their relationship in terms of interaction.

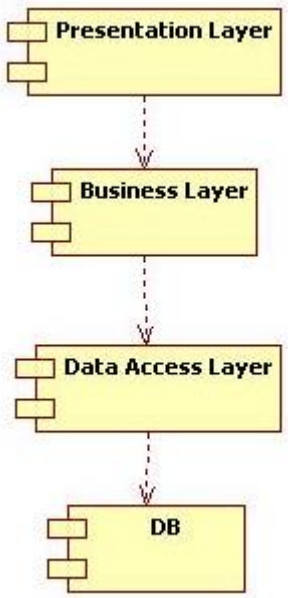

Figure 6: Component Design

5.2.5 Detailed Program Sequence

A sequence diagrams are a UML diagram that depicts the interactions among various application components or participants over time, including but not limited to, system objects, actors, and other systems or services, in order to accomplish a task. It shows object interaction in time sequence.

1. Login - see Appendix A Login User
2. Post advertisements - see Appendix A Post advertisement

3. Search advertisements - see Appendix A Search advertisement
4. Rank advertisements - see Appendix A Rank advertisement
5. Comment on advertisements - see Appendix A Comment advertisement
6. Mark Favorite advertisements - see Appendix A Mark Favorite advertisement
7. Send messages to the users - see Appendix A Send messages

5.2.6 Detailed Program Specification

The class diagram is a structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

5.2.6.1 Class diagram For Application

This class diagram shows all the important classes with their attributes and methods pertaining to the application.

See Appendix B – Figure 14: Class Diagram

5.2.6.2 ER diagram For Application

This ER diagram shows all the tables with their attributes. This ER diagram is reverse engineered from the actual database of the application. This consists of all the data types and the relationships between each table.

See Appendix C – Figure 15: ER Diagram – Database Level

5.2.7 Deployment View

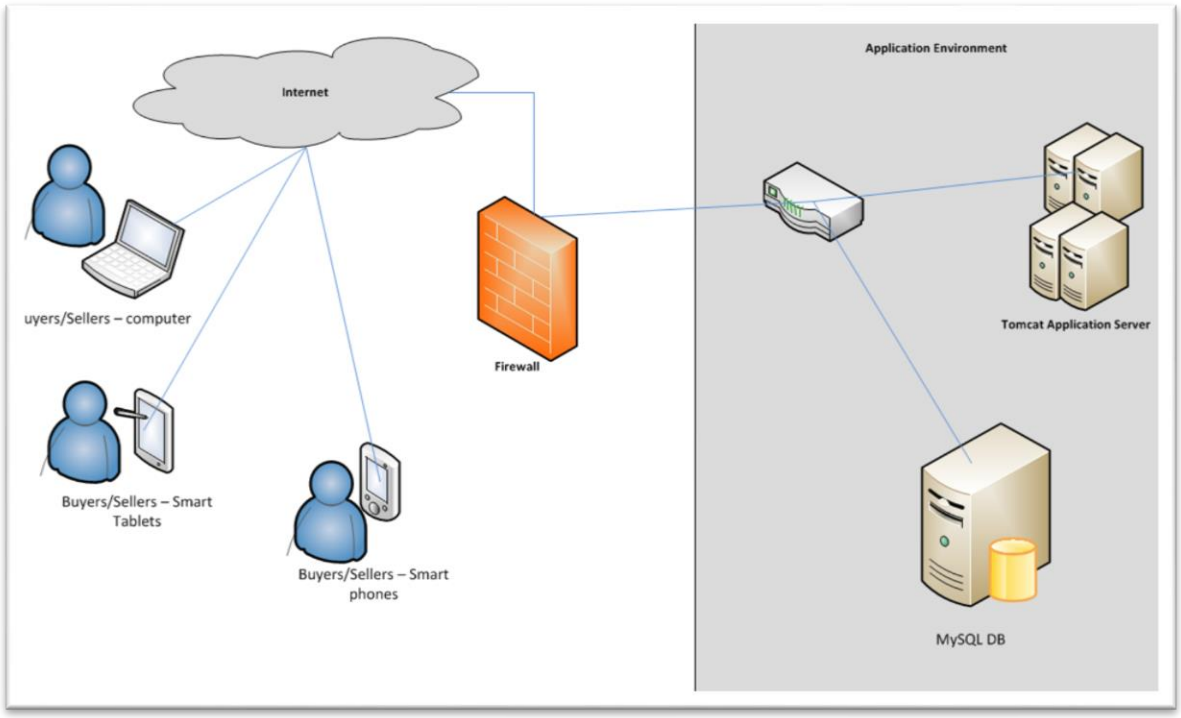

Figure 7: Deployment View

5.2.8 Implementation Model

5.2.8.1 Package Structure

Following package Structure will be implemented for application development.

Figure 8: Table of Package Structure

Application Layer	Package
Web Layer	com.isd.sappu.savari.controller
Business Layer	com.isd.sappu.savari.service
DAO Layer	com.isd.sappu.savari.dao
Model Object Layer	com.isd.sappu.savari.model

5.2.8.2 Folder Structure

Following Folder Structure will be implemented for application development.

Figure 9: Table of Folder Structure

Content	Folder Structure
JSP Files	WEB-INF/views/content
Images	/images
Java Scripts Files	/js
Style Sheet Files	/css

5.3 Summary

The proposed system will be developed and deploy on open source technologies.

Mainly back end of the system will be developed with Java and Spring framework. Spring framework consists of different layers which provide different functionalities [17]. Proposed system security handled by the Spring security which is more reliable and robust for web security [18]. For web oriented functionalities Spring front end controller will be used. Also for the back end database layer Spring Data JPA [19] will be used. The goal of Spring Data JPA and repository abstraction is to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores [19]. This will increase the efficiency of implementation and maintain overhead.

In the next chapter, we will discuss detailed information about the implementation according to the analysis and design we have gone through in this chapter. The user interfaces, source code, algorithms will be described in detail in following chapter.

Chapter 6

Sappu Savari – Implementation

6.1 Introduction

In the previous chapter, we discussed about the design of the proposed system. The technologies planning to use and the finalized architectural design were discussed in detail. In this chapter includes detailed information about the implementation of the software system. The user interfaces, algorithms and logics we need to use for the implementation are included in this chapter. Component wise, how coding will be used and what design patterns will be used in which area is described in detail within this chapter.

6.2 Implementation of the Sappu Savari

6.2.1 User Authentication

The Users need to be authenticated before accessing the system, it allows the system to identify the location and other relevant information user wise. Users can create their own username and passwords using Signup. Then login page (see appendix D – Figure 19: Login) can be used to enter relevant username and password. The system encrypts the entered user name and password and decrypts them when reached to the server. Decrypted username and password will be matched from the data fetched from the database. If the system could match the exact username and password, it will authenticate user as a valid user. Then within the same process the user role will be identified and basic user information will be injected to the session for future use. Also system creates a backlog, including login time and date (See Appendix C – 9.1.5.40 – Login Success Service).

If the system cannot find a proper match for the entered username and password, the system will be create a back log including access date time and redirected to the login screen with an error message (See Appendix C – 9.1.5.41 – Login Failure Service).

The system is using http secured protocol for username and password encryption and decryption which will provide more security for network communications. In server end passwords are hashed using MD5 [20] cryptographic hashing algorithm.

6.2.2 Detect User Location

Authenticated user's location will be identified; longitude and latitude are saved in a database under user references. The system will be fetched the user location in every 5 second interval and this allow to identify the user location even if the user is on the move.

The longitude and latitude are identified by GEO API which allows identifying the location even if the user doesn't have Geo location sensors in their devices. Using below code snippet the system could fetch the user locations.

```
function success(pos) {
  var crd = pos.coords;
  console.log("latitude="+crd.latitude);
  console.log("longitude="+crd.longitude);
  console.log("accuracy="+crd.accuracy);
};

function error(err) {
  console.warn('ERROR(' + err.code + '): ' + err.message);
};

var options = {
  enableHighAccuracy: true,
  timeout: 5000,
  maximumAge: 0
};

navigator.geolocation.getCurrentPosition(success, error, options);
```

Jquery AJAX is used to save the user location details in the database without refreshing the web page.

6.2.3 Add Product Items

Users act as sellers can add multiple products items to advertise through the system. The products are categorized by product categories such as Electronics, Automobiles, ..etc. According to the categories there are sub categories available, eg: mobile, home electrical under Electronics.

According to the categories and sub categories product parameters are changed. So the system populates different types of forms to capture the information. The categories are implemented using Java enumerations. Sub categories also implemented as Java enumerations.

In the database layer categories, subcategories and products are saved under three separate data tables and they are linked together under correct relationships.

Also, users can upload images of the product. These image files are saved in a different file location external to the system project files. Also the Meta data information of the files are captured and saved in a different database table. Below code snippet used to capture the image files Meta data information.

```
File file = new File(filePath);
AttachmentFile attachmentFile = new AttachmentFile();
attachmentFile.setName(name);
attachmentFile.setDocumentName(documentName);
attachmentFile.setFilePath(filePath);
attachmentFile.setFileName(FilenameUtils.getBaseName(filePath));
attachmentFile.setActualFileName(actualFileName);
attachmentFile.setAttachmentType(attachType);
attachmentFile.setFileSize(file.length());
attachmentFile.setMediaType(Files.probeContentType(file.toPath()));
attachmentFile.setExtension(FilenameUtils.getExtension(filePath));
attachmentFile.setCreatedDateTime(new Date());
attachmentFile.setUploadedUser(Session.getLoggedUserId());
```

6.2.4 Search Products

Authenticated users can search the available products in the system using the advanced searching functionality. Since the product items are under different categories and subcategories before moving into the search; user has to select what type of categories and sub categories. According to the selected categories and sub categories the search parameters are derived.

After capturing all the required parameters; system transfers those parameters to the data access layer. The search functionality needs to execute complex queries to the read the correct product items from the database. Since the easiest access point of the database is a data access layer; all the queries are executed in the data access layer to increase the performance of the search functionality. According to the categories and sub categories query executions may differ. In search product data access service mapped list of product objects from fetched product details from the database. Those sets of objects are transferred to the service layer to do the rest for displaying the search results.

Search queries are structured to increase the performance. Since the products are categorized under different product categories and subcategories; queries are designed to search only specific product categories and sub categories, the queries are executing to search deeper on specific parameters such the product status, price range, color, … etc. Also, since queries are optimized for different product categories; unwanted parameters are skipped without executing. All the queries are written under MySQL syntax standards.

While user search different product items, the system keeps track what are the parameters of the search for specific users. These data gathered in a separate table in a database. This data set can be used for automated notifications later. In search request list user can re execute their search without re-entering all the parameters.

6.2.5 Notifications

The system always provides real time notifications. When the new product item searched, when your seller is reached by, when you are selling a product is reviewed, the system will be provided notification messages on top of the application. These notifications are not a plain text of providing information; it also interactive to the user, when the user clicks on the notification the application will redirect you to the relevant product item or a web page.

The newly proposed system has two main scheduler processes,

1. User location scheduler process.
2. Notification scheduler process.

We already discussed about the first scheduler process which is mainly focusing on fetching the user's location based on Geo API.

The second main process is a Notification scheduler process. This process is responsible for many tasks,

1. The main job of this process is to constantly check the product items, whether there any matching product item for any search requests which users have provided.
2. According to the location of the users, provide high priority notifications notifying there are sellers you are interested nearby. To get the notification, the relevant seller should be in the 1km radius. The below Java code snippet will calculate the distance of the two users.

```
public double getDistanceFromLatLonInKm(double fromLat, double fromLon,
double toLat, double toLon) {
      try {
            int R = 6371; // Radius of the earth in km
            double dLat = deg2rad(toLat - fromLat); // deg2rad below
            double dLon = deg2rad(toLon - fromLon);
            double a = Math.sin(dLat / 2) * Math.sin(dLat / 2)
                        + Math.cos(deg2rad(fromLat)) *
            Math.cos(deg2rad(toLat)) * Math.sin(dLon / 2) *
            Math.sin(dLon / 2);
```

```
                double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
                double d = R * c; // Distance in km
                return d;
        } catch (Exception e) {
                System.out.println("****error occured when calculating the
                distance");
                e.printStackTrace();
                return -1;
        }

}

public double deg2rad(double deg) {
        return deg * (Math.PI / 180);
}
```

These scheduler processes are implemented in service layer which is more suitable for implementation of business logics.

See Appendix E - 9.5.1.30 – Scheduler Service

6.2.6 Messaging Platform

The newly proposed system provides a reliable messaging platform. Any user can send messages to the relevant buyer or seller. The message includes a subject field and content with 10,000 character possibility.

The architecture of the messaging platform is simple but reliable. Every message is saved in the database with relevant information such as recipient, subject, date time, etc. Scheduled service is performed to inform the user that a new message is arrived. This provides more secure communication media than sharing mobile phone numbers or email addresses. The message is encrypted while sending through the network using https web protocols.

The system provides a message dashboard which includes all the past sent messages and inbox messages. Also system back log is maintained for auditing purposes, including all the relevant information such as received date/time, sent date/time, etc.

6.2.7 Product Rating

Users can rate the product item by giving marks (1 -worse and 5 - best). Summarized rating will be visible on top of the product view screen. The overall rating will give a better idea about the product to the users.

The system captures all the rating fewer than two relationships; product and the user. These rating information is saved in the MySQL database and every time it calculates the summarized rating. The summarized rating will be calculated using below code snippet.

```
public int calculateOverallRating(long productId) {
	List<Rating> ratingList = this.getRatingsByProductId(productId);
	int ratingSum = 0;
	int ratingCount = 0;

	for (Rating rating : ratingList) {
		ratingCount++;
		ratingSum = ratingSum + rating.getRating();
	}

	if(ratingCount>0 && ratingSum>0){
		return ratingSum/ratingCount;
	}else{
		return 0;
	}
}
```

According the code snippet the logic will be getting the sum of the total rating score of a particular product item and divide it from the no of rating.

The formula as below,

Product Rating = Sum of Rating / No of Rating

6.2.8 Comments

Users can write comments to specific product items. This option can be used to provide true information about the product and give some reviews. Then the other users can decide or take any decisions with more information rather than understand which seller already described.

The system uses JavaScript, AJAX for saving comments in the database without refreshing the web page. Also, these comments are saved under different product items in the database.

Also system back log is maintained for audit purposes.

6.2.9 Other

The system is designed to be developed by open source technologies. The tools are used for the implementation process also open source.

For the Database layer MySQL database and Java will be used. For further functionalities such as connection pooling, maintain scalability Spring data JPA framework layer will be used.

For the Business layer will be developed using Java and Spring framework.

The presentation layer will be developed using JSP, JSTL, JQuery and BootStrap.

Additionally Jasper reporting will be used for report generations.

Eclipse, MySQL workbench, iReport, the Chrome web browser will be used as development tools for the newly proposed system.

6.3 Summary

The proposed system is based on open source technologies. Since the implementation to deployment open source technologies and open source tools are used.

In the next chapter includes information on related to test, further development and conclusion of the proposed system.

Chapter 7

Evaluation of the System

7.1 Introduction

In the previous chapter, we discussed the implementation of the system. The wireframes, logics and algorithms we used for different business logics were discussed in detail in the previous chapter.

This chapter includes the testing of the proposed solution, evaluation of each functionality. Also, this chapter will discuss evaluation aspects to see whether the objectives of the project have been met and also identify any changes required to achieve the objectives. This has been done based on specified ranks.

7.2 Evaluation and Testing of the Project

Sappu Savari is consisting with much functionality which supports users in their marketing and advertising. Each functionality has similar priority and will have an equal contribution to achieve the aim and objective of the project. Testing process will carry out to see whether the implemented solution is worked as intended. This will include component testing and system testing. Testing of individual components comes under component testing while system testing will cover the testing of groups of components integrated to create a system or subsystem.

In the evaluation process, each functionality will be evaluated separately using necessary criteria. In this context questionnaire is provided to be filled up by the relevant audience. 2 quality assurance engineers, 5 IT executives, 10 general users participated in this evaluation process. A questionnaire has been designed to get the feedback of the important functions of all modules.

When preparing the evaluation criteria for five modules following aspects are given more consideration among others.

Information on the program: These information criteria will cover the relevance to the audience, appropriate language, the organization of the information and quality of the information. Clearly explain how the facts etc.

User satisfaction and interfaces: These criteria evaluate the appearance of interface, ease of use them, the color choices of the interfaces, how clearly fonts could be read, organization and placing of interfaces in the screen etc. Briefly, how far users are motivated when using interface and evaluated.

Usability: Whether users are happy to use the modules frequently or are they reluctant to use it, whether users are confident when using the modules/system, will clear guidelines provide by the system when user get stuck at any point are considered when evaluating this part. If it is necessary to enter data referring a source document, whether entries going to be entered in the screen are matched with the data in the source document.

Ease of learning: Easy remembrance of terms used, how easy the novice can use the system, the simplicity of using help menu are taken into consideration when evaluation easy of learning criteria.

User Interaction: These criteria cover the user's interaction with the system or modules, the objectives and features of the program, the purpose of the program is well defined and clearly explained to the user etc.

7.3 Evaluation of Sappu Savari

In this process, the questionnaire contained evaluation criteria for an e - commerce site has been provided only for 17 personnel. This group includes 2 QA engineers, 5 IT executives, 10 general users. The results of the evaluation team are in a worksheet with the average figures. These average figures are demonstrated in a graph to get the overall picture of the evaluation.

The information provided in the application

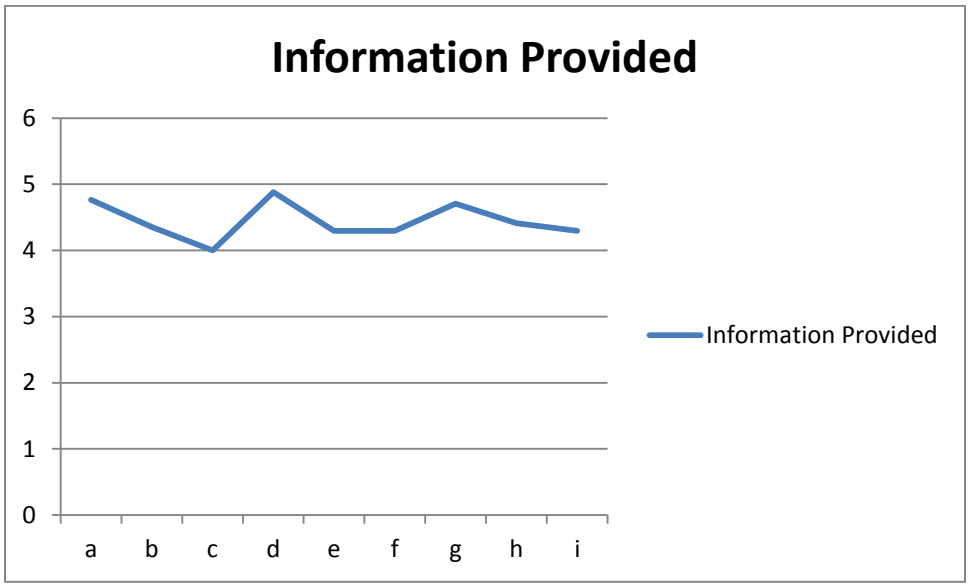

Figure 10: Observation summary – Information provided in the application

a. Content is free from spelling and grammatical errors.
b. The information provided in the site is clear, concise and informative to the intended user.
c. Information given in the site is helpful for a learner.
d. The language is non-discriminatory. Content is free from race, ethnic, gender, age and other stereotype.
e. By using the information given in the site consumers are able to solve most of their problems.
f. All the needed information is available in the system, as an example the mobile phone brand list.
g. Information given in non-technical understandable language without technical jargons.
h. Information is well structured in the application, suitable for marketing purposes.
i. Information given in the system not confusing the operator

According to the above chart; average of rating is above 3 with all the questions. The least rating is received for c. Information given in the site is helpful for a learner. According to the least rating, the information needs to be more descriptive to read and understand for a beginner user. There should be small description on all the headings of the pages illustrating which type of functionality and what type of information required completing the operation.

User satisfaction on interfaces

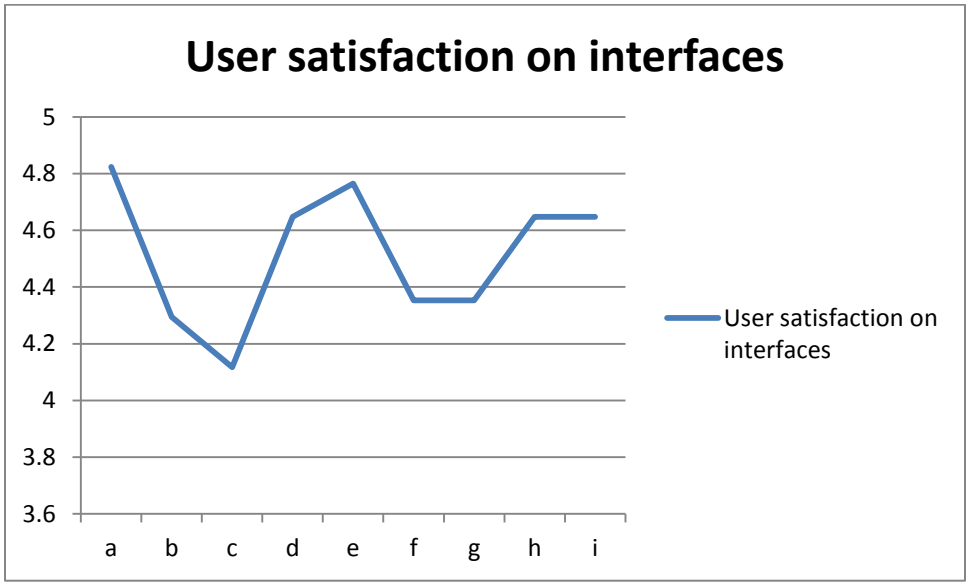

Figure 111: Observation summary – User satisfaction on interfaces

a. Reading of letters on application.
b. Organization of information.
c. Use of terminology throughout the application.
d. Proper navigation.
e. Design of a page provides relevant information only.
f. Audit Information is visible in the application.
g. Position of messages on screen is easy to view.
h. Color choices visually accessible and pleasant to see.
i. The site achieves its purpose.

According to the above chart; average of rating is above 3 on all the questions. The least rating is received for c.Use of terminology though out the application. According to the least rating, the information the terminology should maintain throughout the application. As per the evaluator's comments and ratings overall interface design is matched to the intended purpose and selected color schema also unified throughout the application. Interfaces are well designed to match the resolution of any type of devices.

Usability

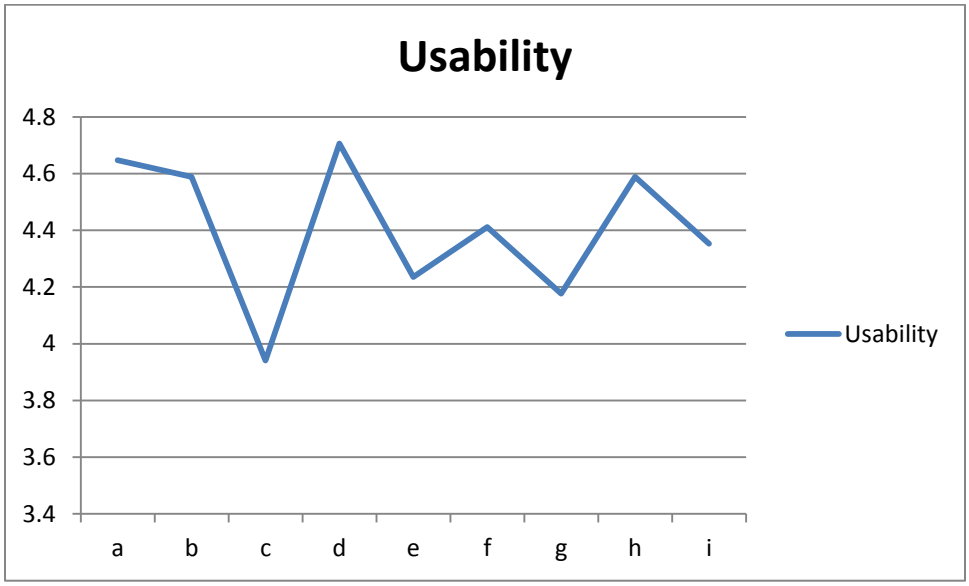

Figure 122: Observation summary – Usability

a. The layout and the design of the application encourage frequent use of the application.
b. Easy navigation.
c. The organization is clear, logical and effective, making it easy and simple for the intended audience to understand.
d. Individuals can easily start and exit the program.
e. The individual has the choice of going directly to desire information or using a structure search to identify relevant topics.
f. Feel very confident when using the application.
g. Not necessary to get to know about the site before it could effectively use it.
h. The language use in the application is clear to the intended audience.
i. Application can be used without written instructions.

According to the above chart; average of rating is above 3 on all the questions. Base on the rating and feedback usability is maintained throughout the application well. Users were able to do what they have intended.

Ease of learning

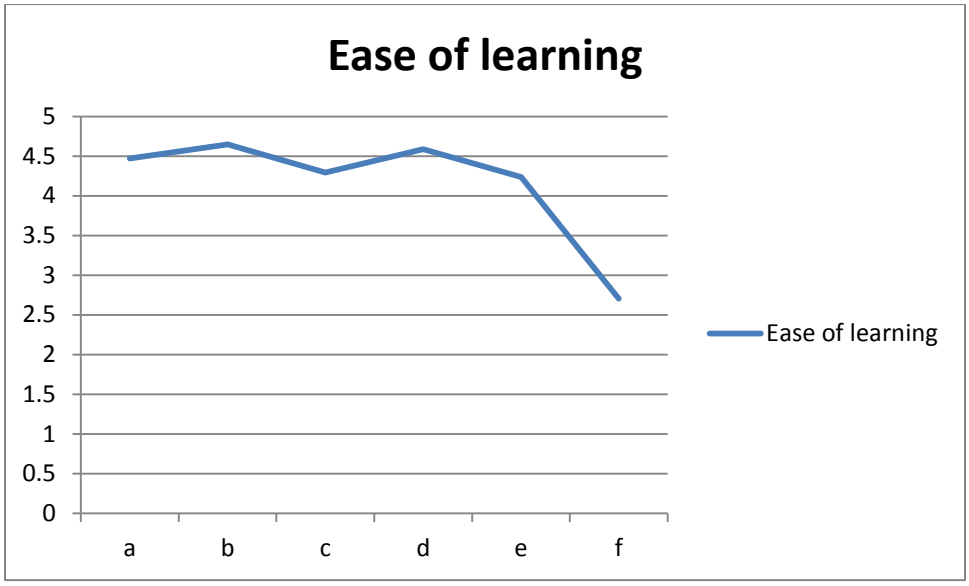

Figure 133: Observation summary – Ease of Learning

a. Can learn to navigate the application quickly.
b. Can easily remember how to use the application next time.
c. It is easy to remember main and top many item names and use of commands.
d. Tasks can be performed in a straight-forward manner.
e. Help messages on the screen.
f. Links to supplementary reference materials.

According to the above chart; average of rating is not above 3 on all the questions. Least rating is given for f.links to supplementary reference materials. The application is lack of help documentation to the users. So in the future help documentation should be provided; whenever user finds some difficulties on their operations, help documentation should provide proper answer.

Overall Impression

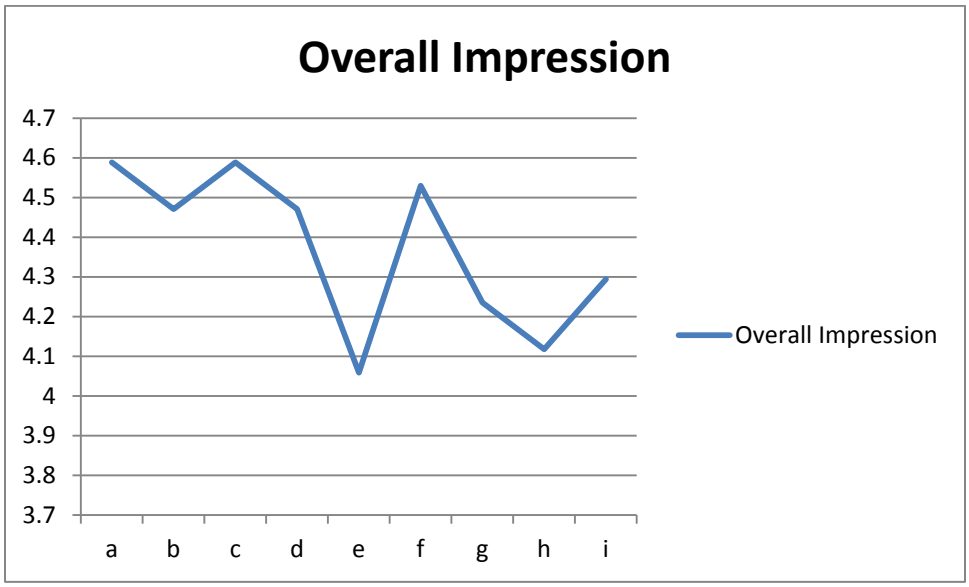

Figure 144: Observation summary – Overall Impression

a. User friendliness.
b. Make life easy with its facilities.
c. Usefulness.
d. Ease of use.
e. Efficiency.
f. System is easy to learn and the information provided is meaningful and helpful.
g. Application is one-stop advertising system.
h. Time taken for search products is acceptable.
i. Time taken to find sellers are acceptable.

According to the above chart; average of rating is above 3 on all the questions. Overall impression of the evaluators was good enough to accept for the production level. Also most of the general users were able to operate the application without any help, except few. Sometimes auto searches take more time than expected, those performance issues are planned to fix in the future.

Chapter 8

Discussion

7.1 Introduction

In the previous chapter, we discussed the implementation of the system. The wireframes, logics and algorithms we used for different business logics were discussed in detail in the previous chapter.

This chapter includes the testing of the proposed solution. Also includes the specialty and the difference of the system over existing advertising and marketing system. When we are implementing this concept to the real world what should be done in the future and further development of the project will be discussed in this chapter.

7.2 Whether the project goal achieved?

The aim of this project is to provide the maximum benefit of using IT for marketing and advertising. As the objective is well described in chapter 1, the newly proposed system 'Sappu Savari' will be well benefited to the people who spend a busy schedule in their life. The time spent on browsing through the posted advertisements and allocate time and venue to meet and investigate are not needed any more for users; they can buy or sell on their day to day life. Users are notified automatically when the best matching seller or buyer comes nearby, so the user will be free of tension and stress of searching the proper buyer or seller to contact. System notification will be included the contact details of the buyer or seller and shortest path details to find the other party. Until you find the best matching seller or buyer the system will continue searching.

If the user needs to find the matching goods or services manually, the system also allows the user to browse through the advertisements. Buyers can rank the sellers for future references which help other buyers to find better sellers.

The users need average computer literacy to use the system. To experience the best of the system user will need a smart phone with internet connection and location facilities. Also internet connection and satellite signal for location facility need should be available and reliable to operate the system properly.

So, according to the comments and the feedbacks, the main objectives are successfully accomplished by the system.

7.3 Problem encountered and limitation

The most challenging part of the project is to identify the location of the users, for the user's location can be detected by different methods, but the method should be reliable and work on almost all the mobile devices and portable devices. Also user location needs to be detected by the laptops and computers, So the best option would be the Geo Location API. But the main issue of this API is, the less accuracy on the internet based location identification using IP addresses. Also the user has to use GPS enabled mobile device for high accuracy.

Also, due to the scheduled operations and automated searches, database traffic is very high compared to other marketing and advertising systems. So the software level, we have to use the maximum level of optimization of the database query to increase the performance of the scheduled operations. But there is a level we could optimize the performance level using software side, the major role of the performance is act of hardware part. The system needs a lot of high performed web, database servers and reliable network with higher bandwidth.

7.4 Future Work

Certain further works have been identified while developing the project. These facts would be briefly described in the section enabling upgrade of the system in future.

Increase Performance

Before moving the production level of this system, it needs to reach its top level performance level, for that there will be some minor changes in the software level. But the whole system will need a high performance web servers and database servers to serve no of users at once. Also web servers will need a higher bandwidth network since huge amount of data needed to transfer between web server and clients.

Increase User friendliness

Also, there are some noticed features to be done before moving to production is, increasing the user friendliness of the system.

Upgrading route mapping

The route mapping is used to find the seller. In the future this same functionality can be upgraded to pinpoint a location to meet the seller and buyer. This location will be placed where both parties can meet located in between. This location can be a restaurant, hotel or a cafeteria.

Upgrading Message Platform

Currently the messaging platform functions as it's intended use, but in the future this messaging platform can be used as a chat platform. Then the users can chat with each other to communicate efficiently and effectively.

Integrate Auto Suggessions

Currently user's interested products are selected by the users themselves. But this feature also can be automated using a knowledge management. The information needs to feed the knowledge management also can be captured using the system business information. Then the products can be suggested according to the knowledge management results.

7.3 Summary

The objective of this newly proposed system is to provide a better and efficient service to the users than existing available internet based advertising and marketing systems. The system mainly focuses for user's Geo location information and provides better informative information from the system. Also, automated processes will relief the users from overhead of searching proper advertisements over and over again; the system provides notifications when there are proper match is available.

Chapter 8

References

[1] K. ZICKUHR, "Location Based Services," PewResearchCenter, 12 9 2013. [Online]. Available: http://www.pewinternet.org/2013/09/12/location-based-services/. [Accessed 21 11 2015].

[2] H. Bray, 30 5 2014. [Online]. Available: http://blogs.discovermagazine.com/crux/2014/04/30/how-location-based-apps-will-shape-the-future-of-shopping/#.Vk9KZ3YrLDd. [Accessed 24 8 2015].

[3] A. Grill, "londoncalling.co," [Online]. Available: http://londoncalling.co/2008/05/location-based-advertising-introduction/. [Accessed 10 11 2015].

[4] E. Corporation, "Closs 5," Ebay Pvt(Ltd), [Online]. Available: https://www.close5.com/. [Accessed 3 9 2015].

[5] D. Olandff, 12 7 2012. [Online]. Available: http://thenextweb.com/apps/2012/07/05/yardsale-for-ios-might-be-the-quickest-way-to-sell-all-of-the-stuff-sitting-around-your-house/. [Accessed 11 8 2015].

[6] A. Popescu, "W3C," Google, Inc, 11 7 2014. [Online]. Available: http://dev.w3.org/geo/api/spec-source.html. [Accessed 12 8 2015].

[7] "W3Schools.com," W3Schools.com, [Online]. Available: http://www.w3schools.com/html/html5_geolocation.asp. [Accessed 12 9 2015].

[8] G. Inc, "Google Maps Directions API," Google Inc, [Online]. Available: https://developers.google.com/maps/documentation/directions/intro. [Accessed 21 11 2015].

[9] ikman.lk, "ikman.lk," ikman.lk, [Online]. Available: http://ikman.lk/en/help/about. [Accessed 20 10 2015].

[10] O. Marketplace, "Oodle Marketplace," Oodle Marketplace, [Online]. Available: http://www.oodle.com/info/about/. [Accessed 8 10 2015].

[11] H. L. Solutions, "http://hitlanka.com/," Hit Lanka Solutions, [Online]. Available: http://hitlanka.com/web-advertising-online-buy-and-sell-best-free-classifids-website-in-sri-lanka.html. [Accessed 8 11 2015].

[12] marketplace.lk, "marketplace.lk," marketplace.lk, [Online]. Available: http://marketplace.lk/en/static/index/id/3. [Accessed 8 11 2015].

[13] B. Jiang, "Location-based services and GIS in perspective," 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.3566&rep=rep1&type=pdf. [Accessed 8 11 2015].

[14] A. Buczkowski, "http://geoawesomeness.com/," 2012. [Online]. Available: http://geoawesomeness.com/knowledge-base/location-based-services/location-based-services-applications/. [Accessed 8 11 2015].

[15] B. Media, "http://www.geocomm.com/," [Online]. Available: http://www.geocomm.com/channel/mobile/airflashsurvey/. [Accessed 8 11 2015].

[16] B. Bosomworth, "Mobile Marketing Statistics 2015," 22 8 2015. [Online]. Available: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. [Accessed 20 11 2015].

[17] H. Gupta, "dzone.com," 10 7 2010. [Online]. Available: https://dzone.com/articles/spring-framework-architecture. [Accessed 20 11 2015].

[18] L. T. Ben Alex, "docs.spring.io," [Online]. Available: https://docs.spring.io/spring-security/site/docs/3.0.x/reference/springsecurity.html. [Accessed 20 11 2015].

[19] T. D. C. S. Oliver Gierke, "2015," 15 11. [Online]. Available: http://docs.spring.io/spring-data/jpa/docs/current/reference/html/. [Accessed 21 11 2015].

[20] MD5.net, "MD5.net," MD5.net, [Online]. Available: http://www.md5.net/. [Accessed 07 03 2016].

Chapter 9

Appendix

9.1 Appendix A – Sequence Diagrams

9.1.1 Login User

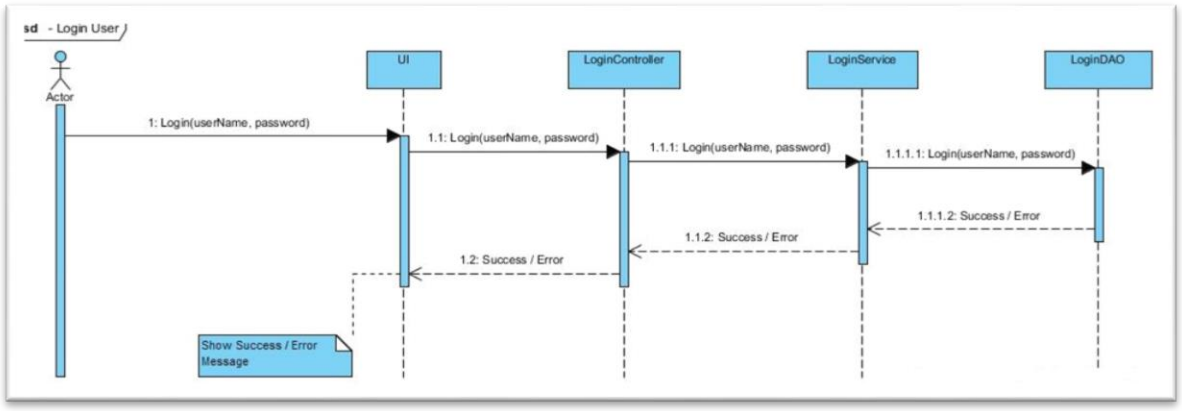

Figure 15: Login User - Sequence Diagram

9.1.2 Post Advertisement

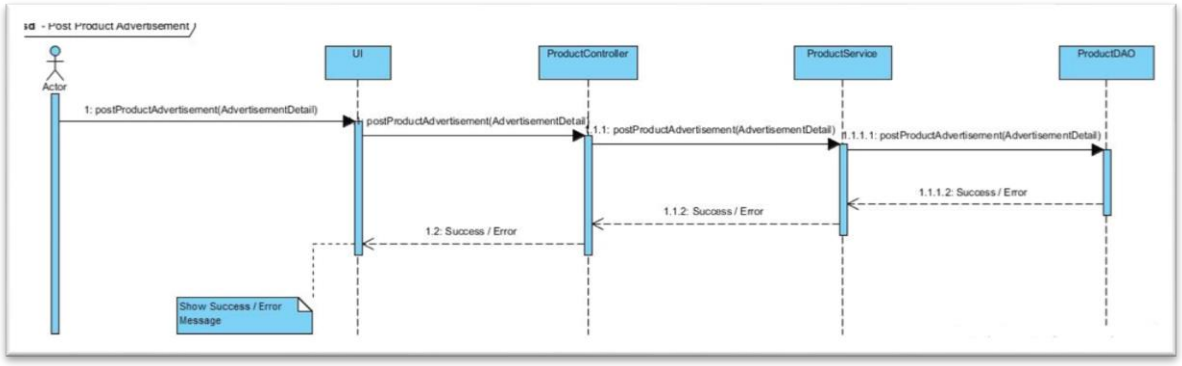

Figure 16: Post Advertisement - Sequence Diagram

9.1.3 Search Advertisement

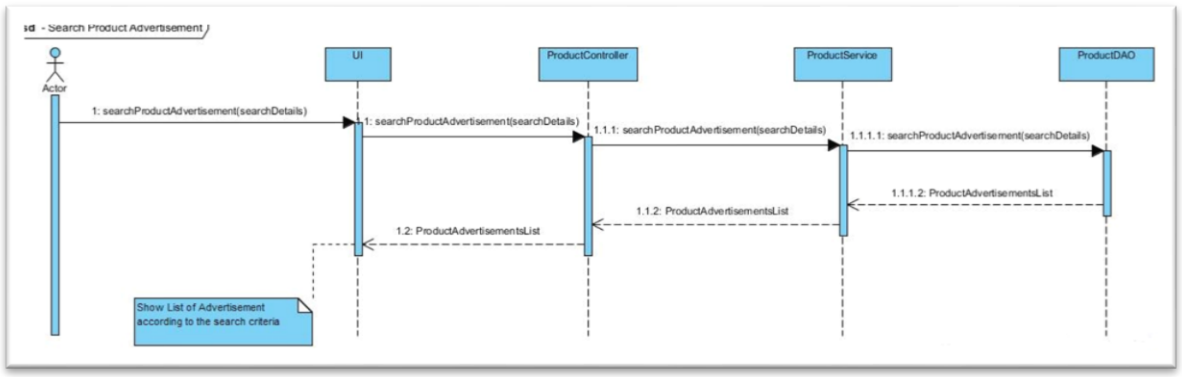

Figure 17: Search Advertisement - Sequence Diagram

9.1.4 Rank Advertisement

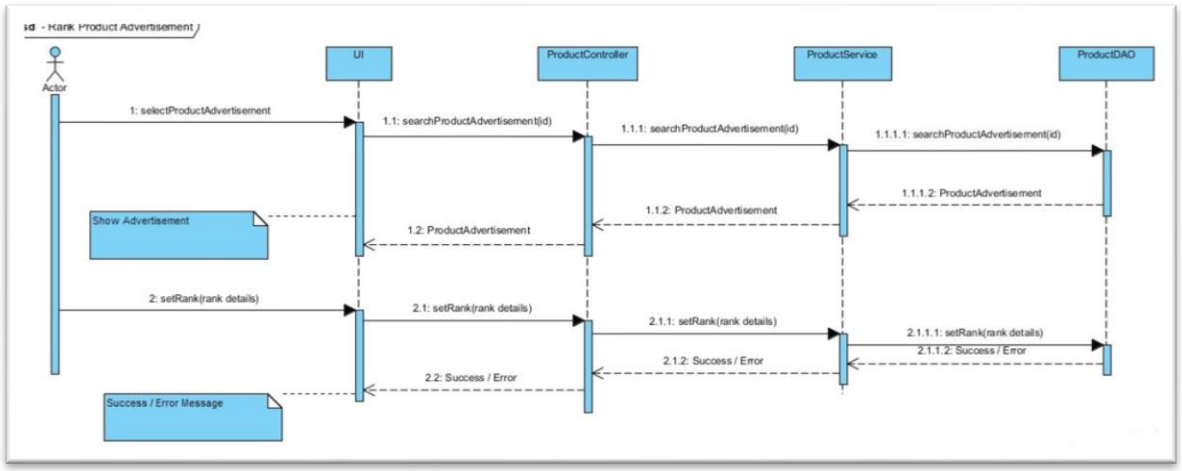

Figure 18: Rank Advertisement - Sequence Diagram

9.1.5 Comment on Advertisement

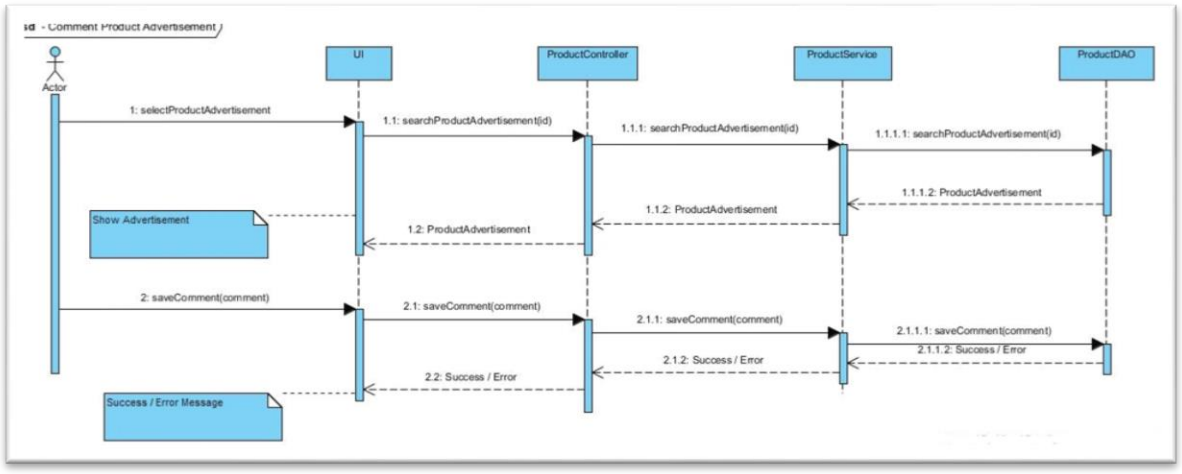

Figure 19: Comment on Advertisement - Sequence Diagram

9.1.6 Mark Favorite Advertisement

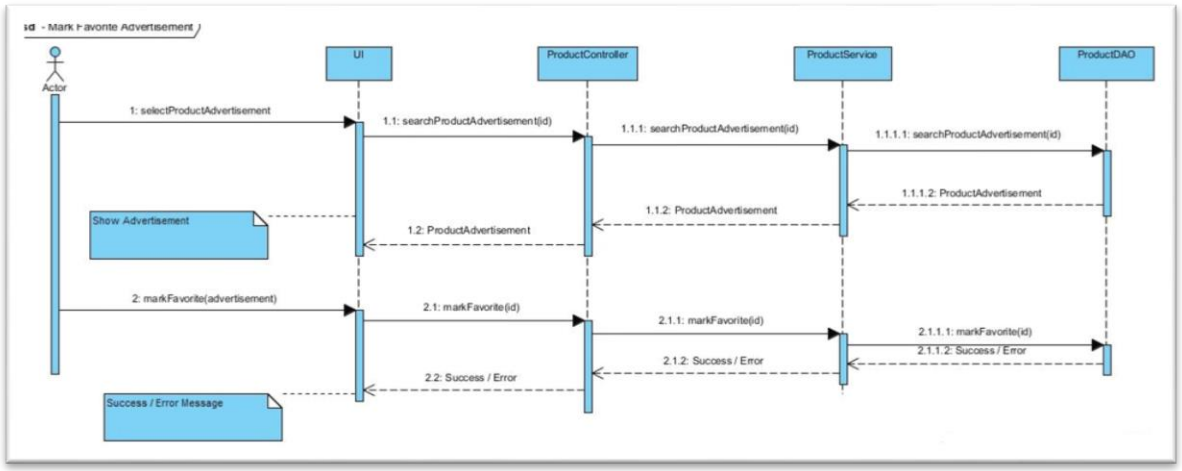

Figure 20: Mark Favorite Advertisement - Sequence Diagram

9.1.7 Send Messages

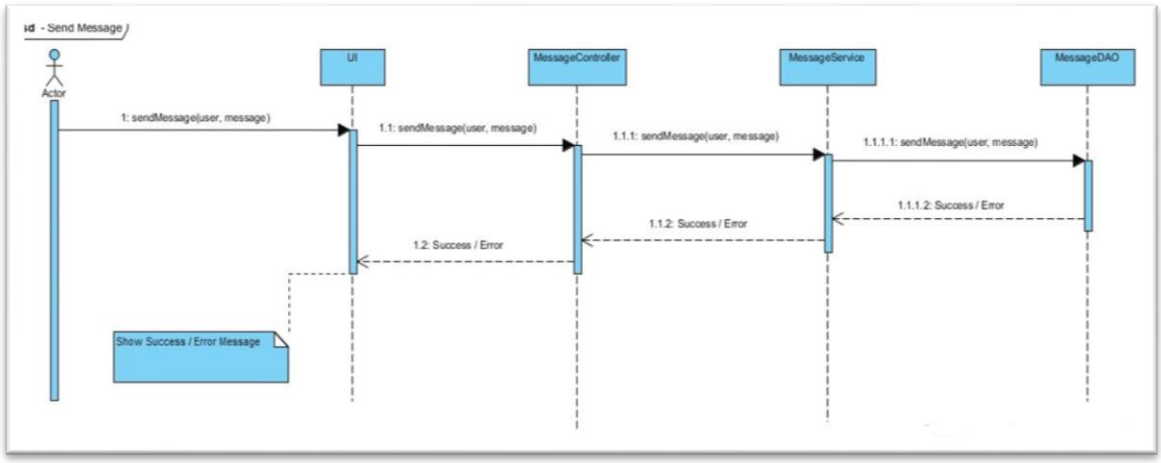

Figure 21: Send Messages - Sequence Diagram

9.2 Appendix B – Class Diagram

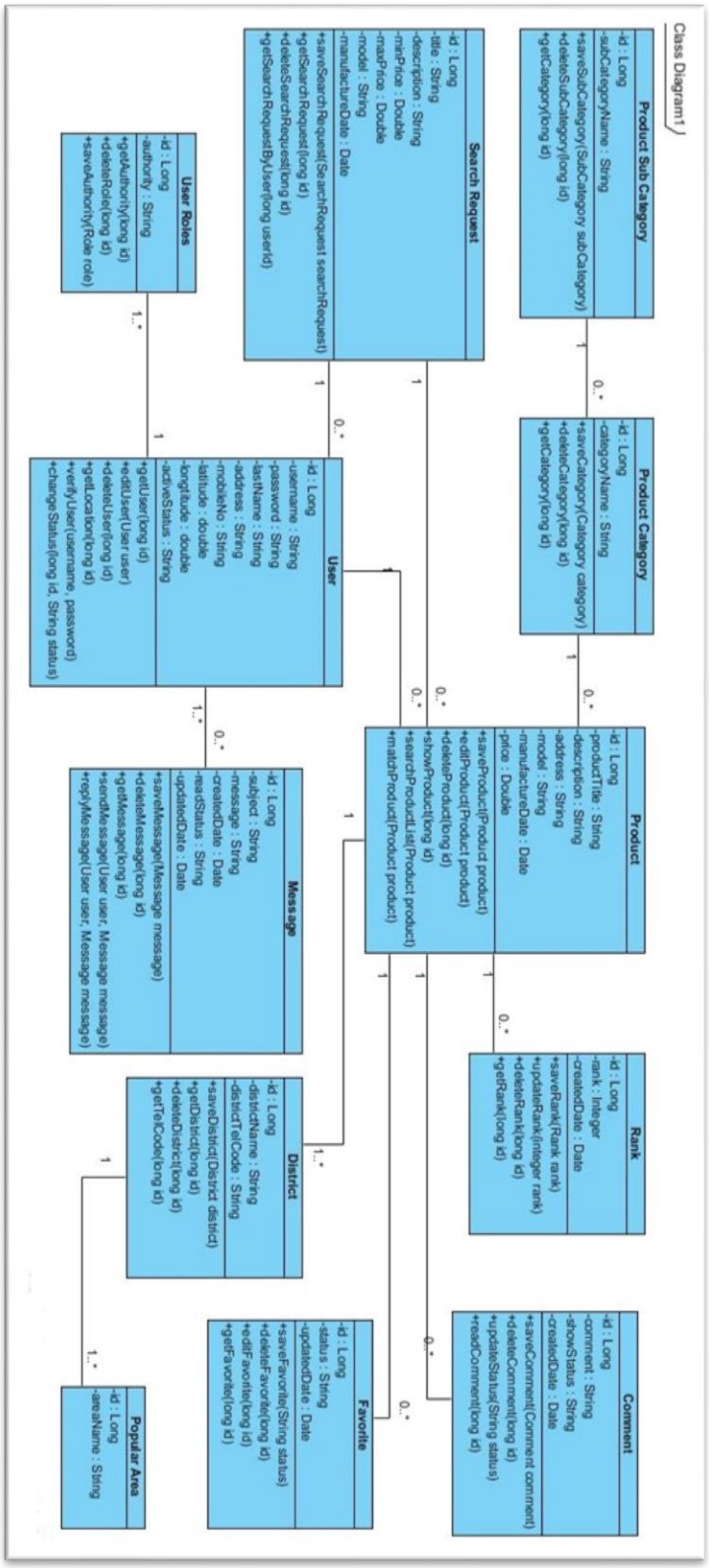

Figure 22: Class Diagram

9.3 Appendix C – ER Diagram

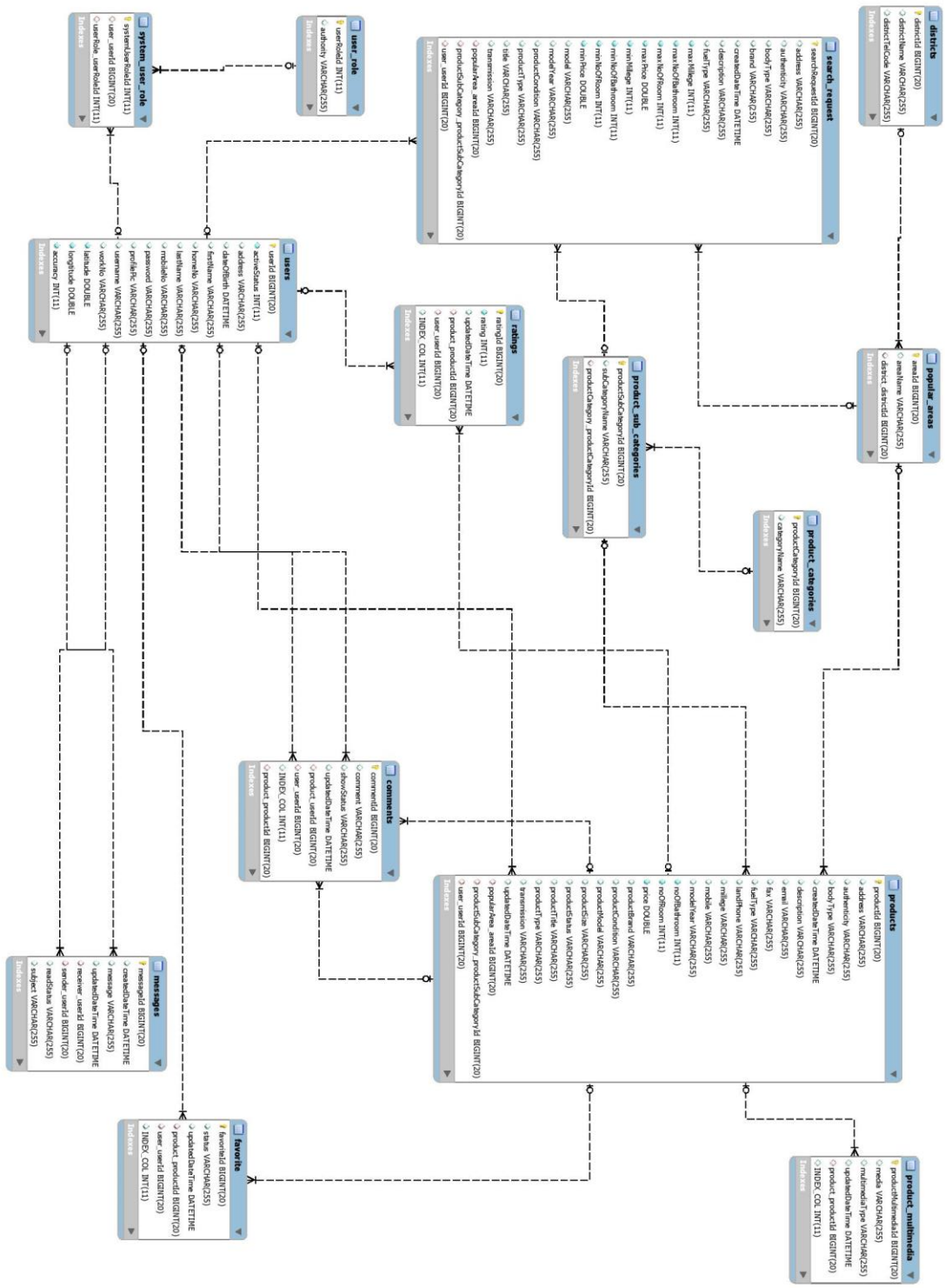

Figure 23: ER Diagram - Database Level

9.4 Appendix D – Screen Shots

9.4.1 Computer Screen Shots

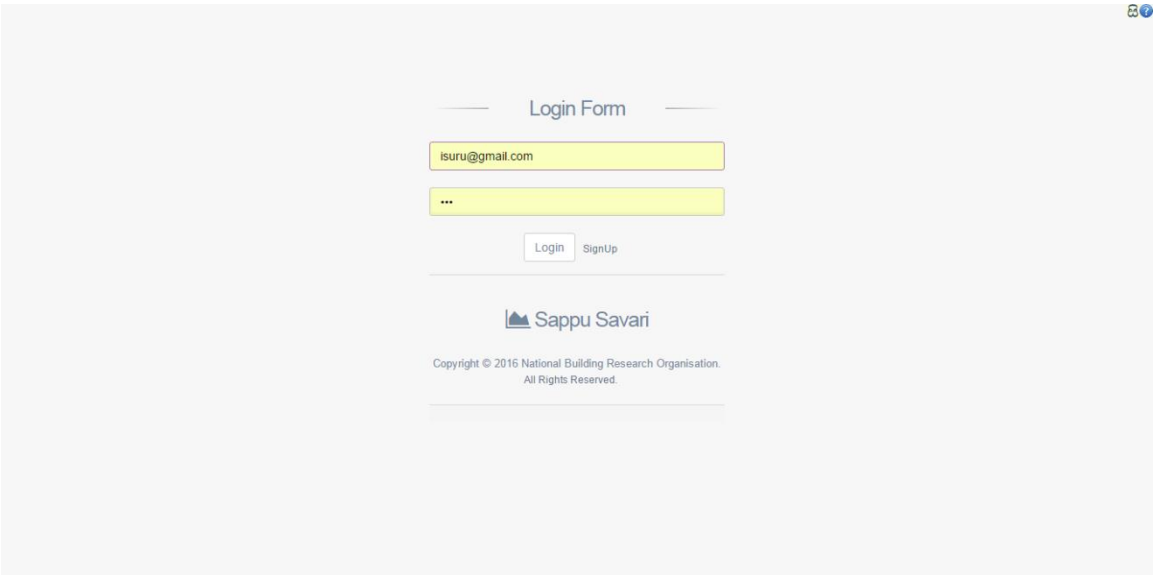

Figure 19: Login

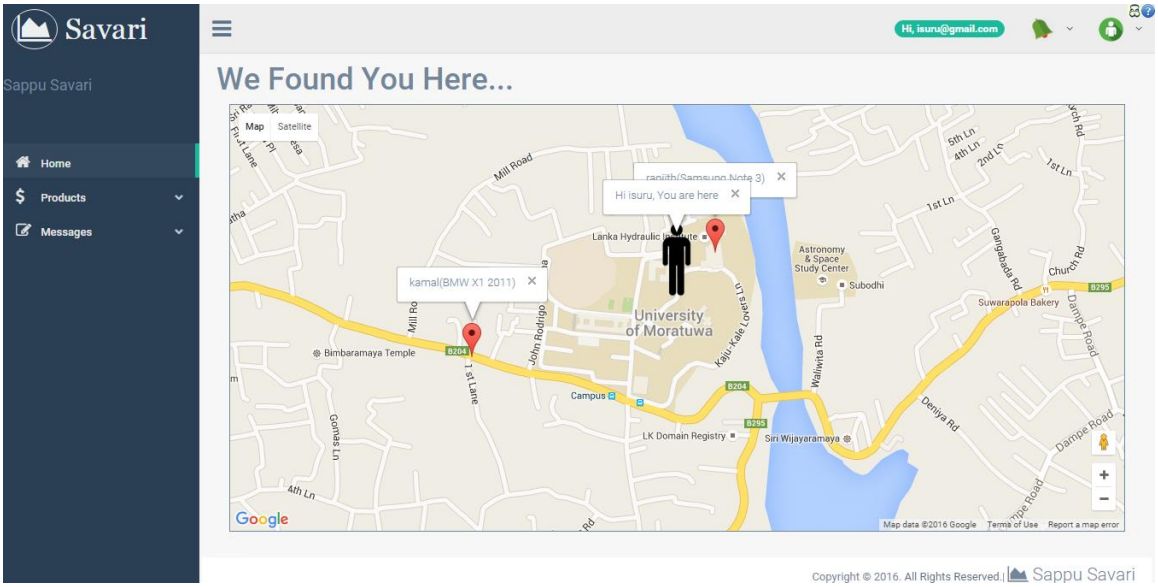

Figure 24: Home

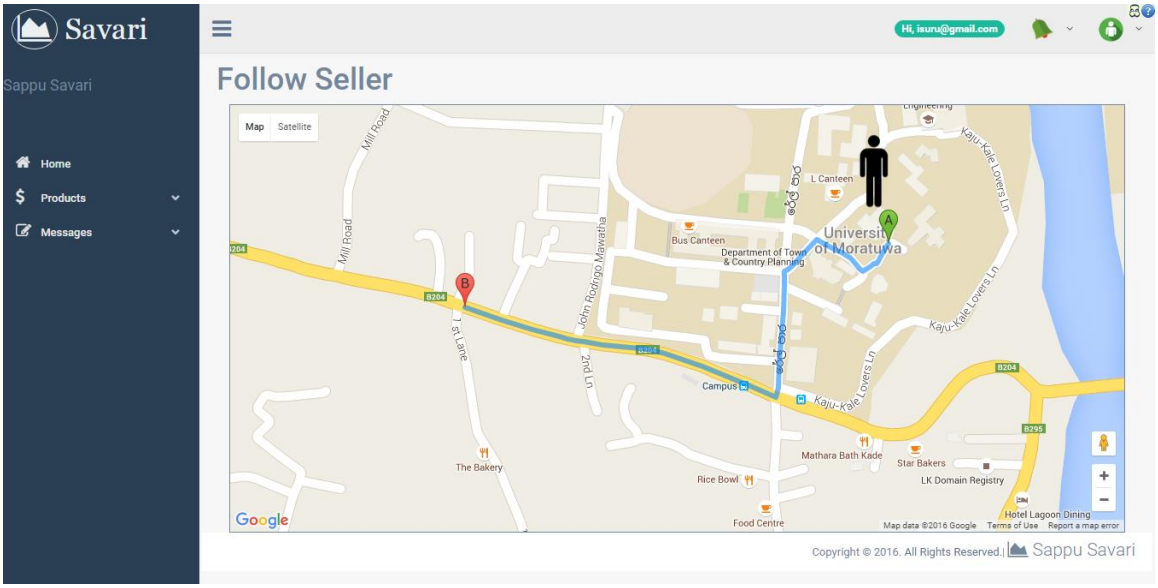

Figure 25: Follow User

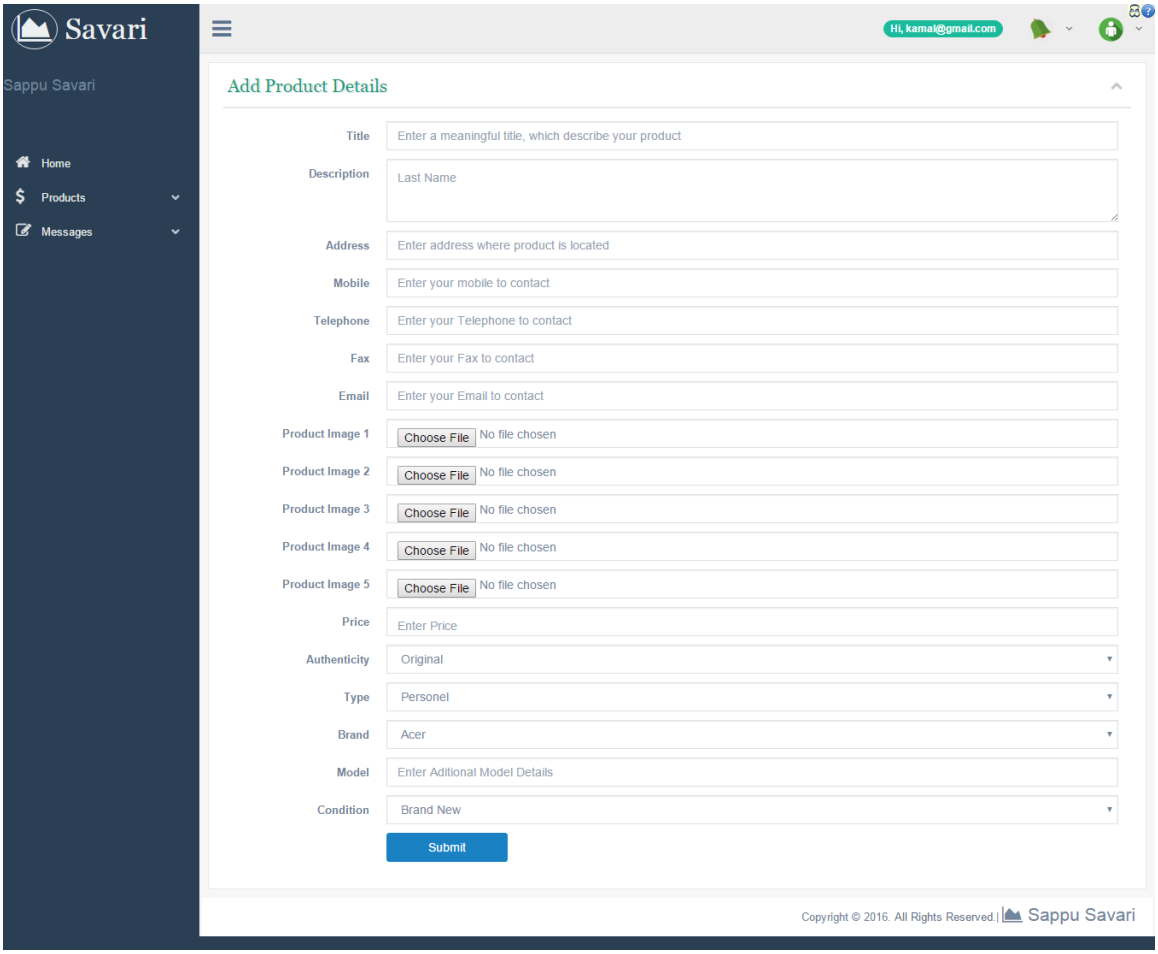

Figure 26: Add Product

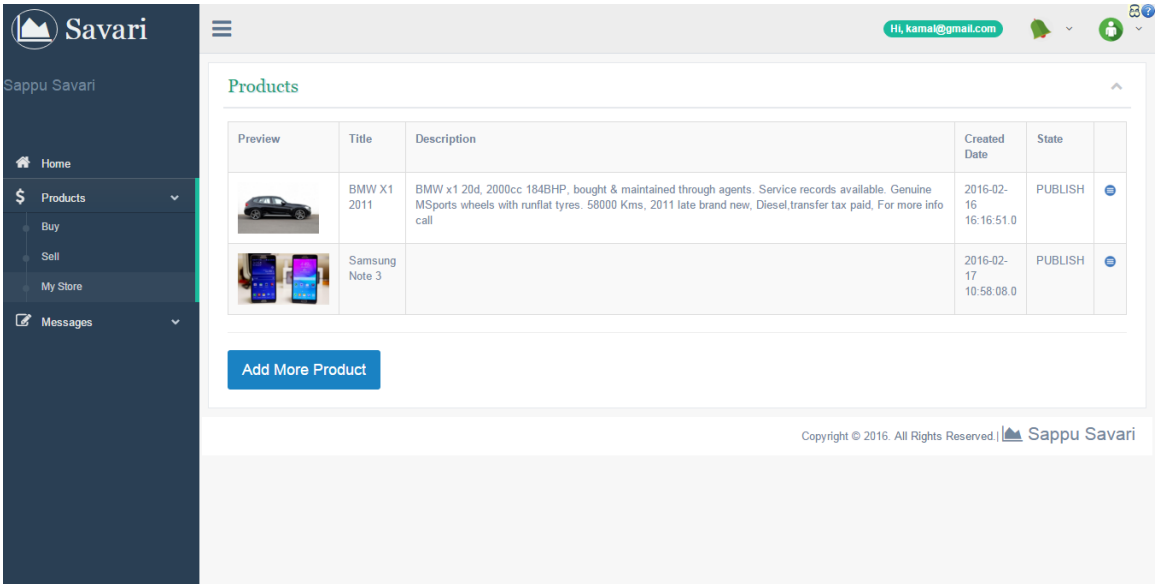

Figure 27: My Store

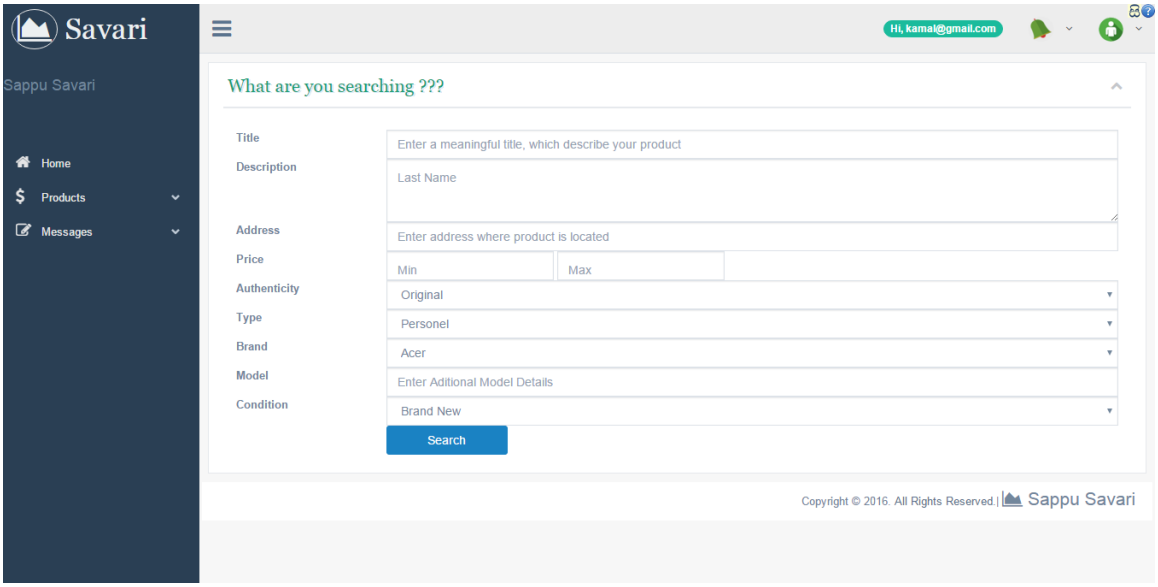

Figure 28: Search Product

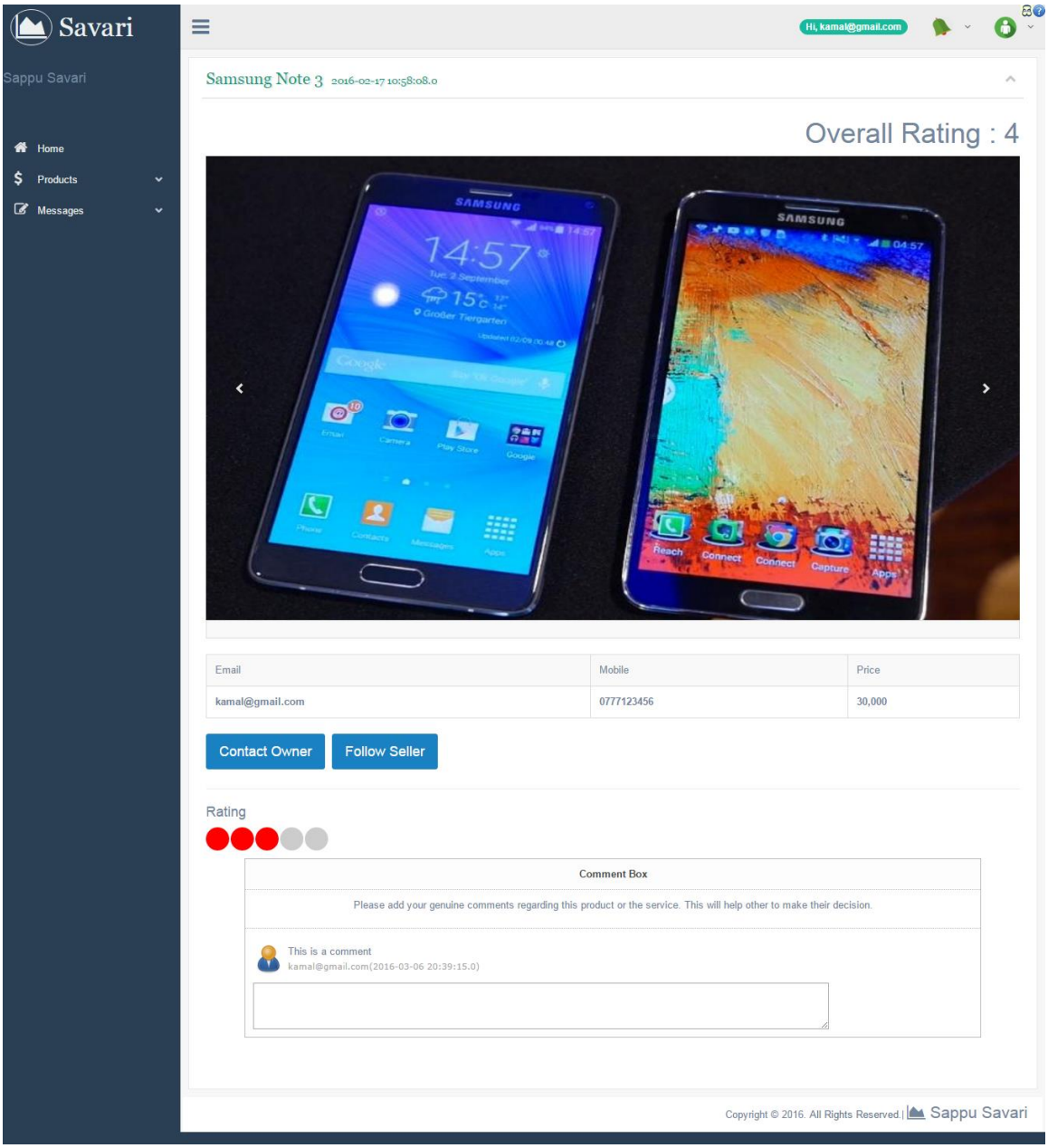

Figure 29: View Product

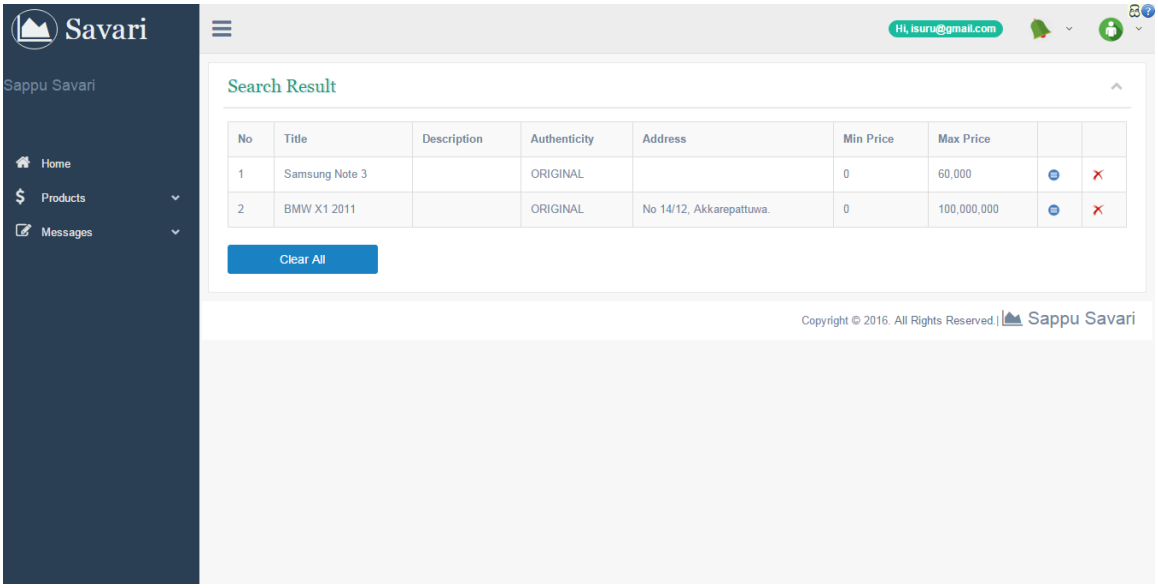

Figure 30: Search Results

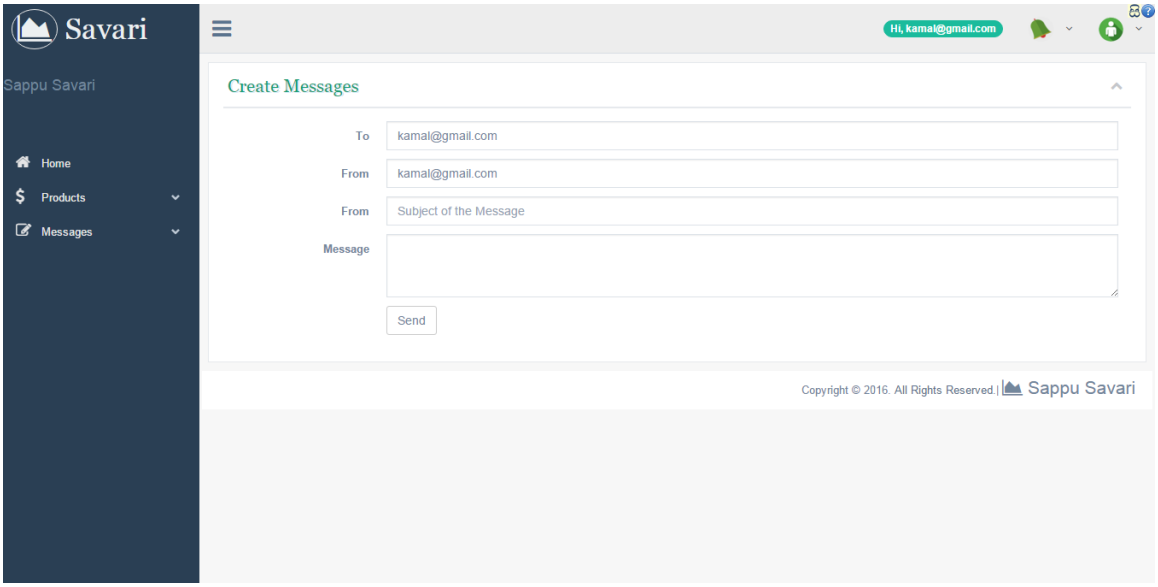

Figure 31: Send Messages

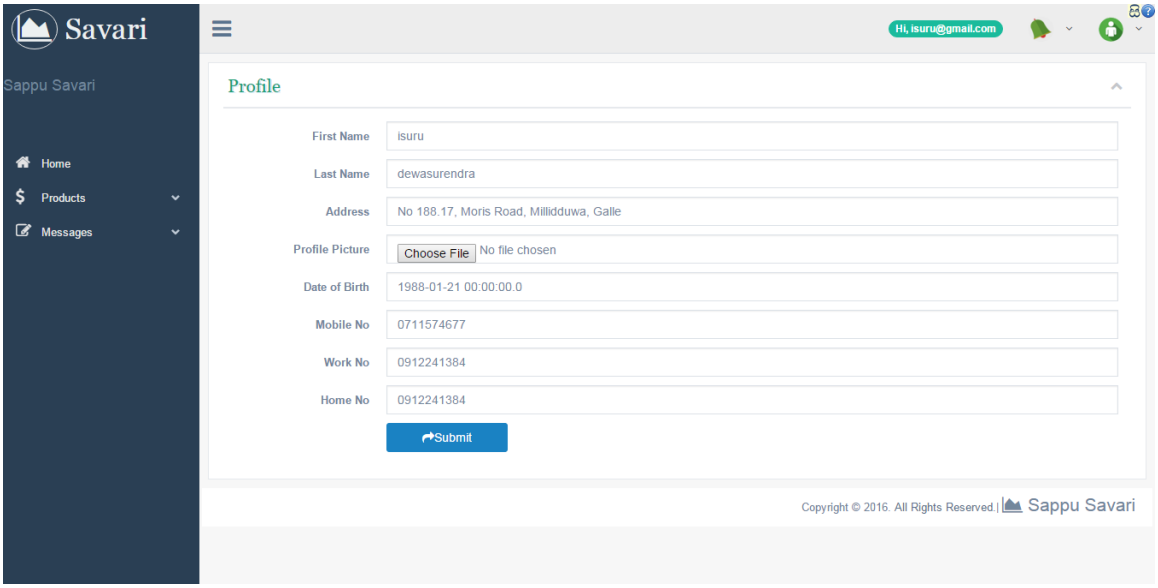

Figure 32: User Profile

9.4.2 Mobile Screen Shots

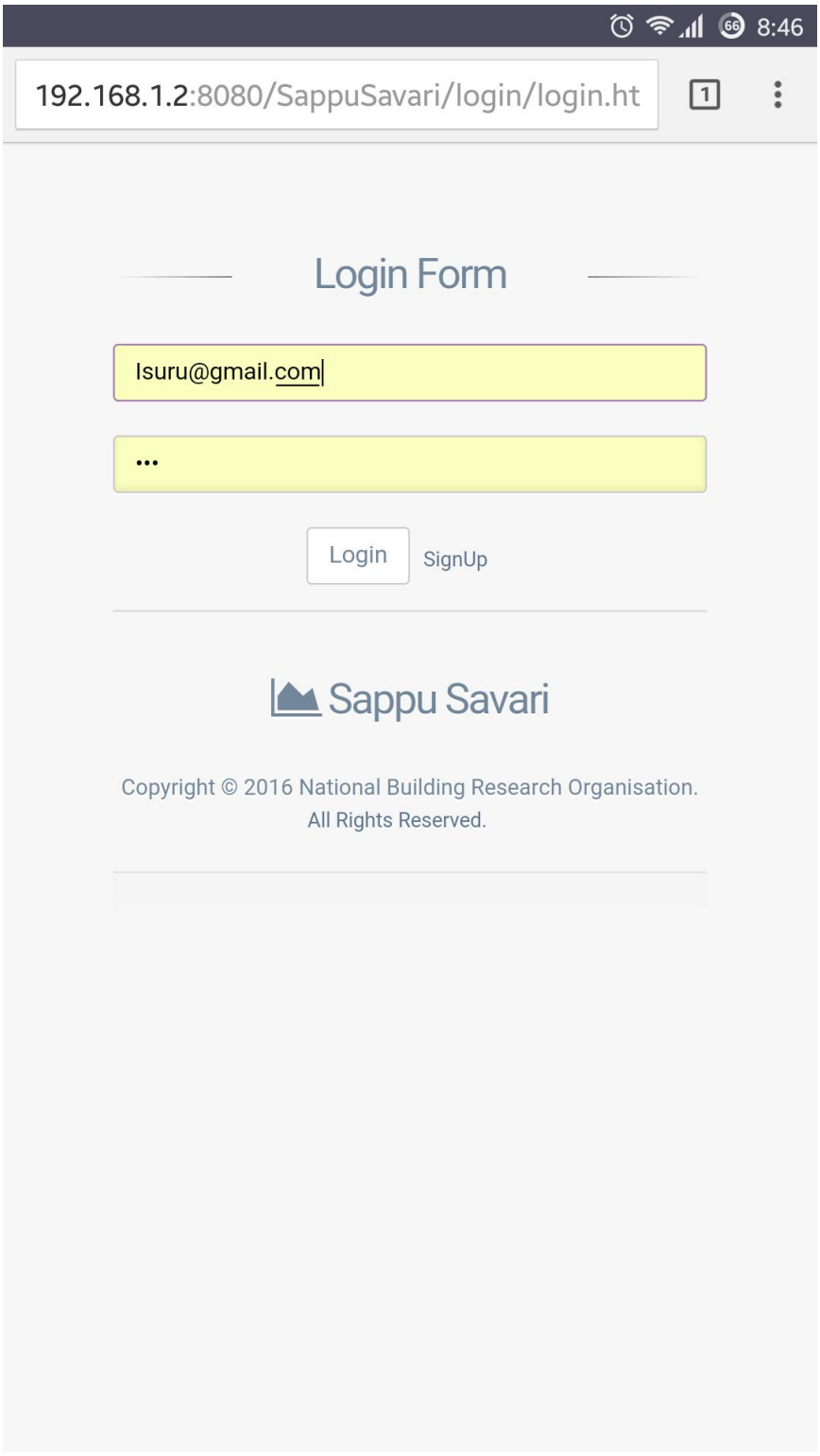

Figure 33: Login

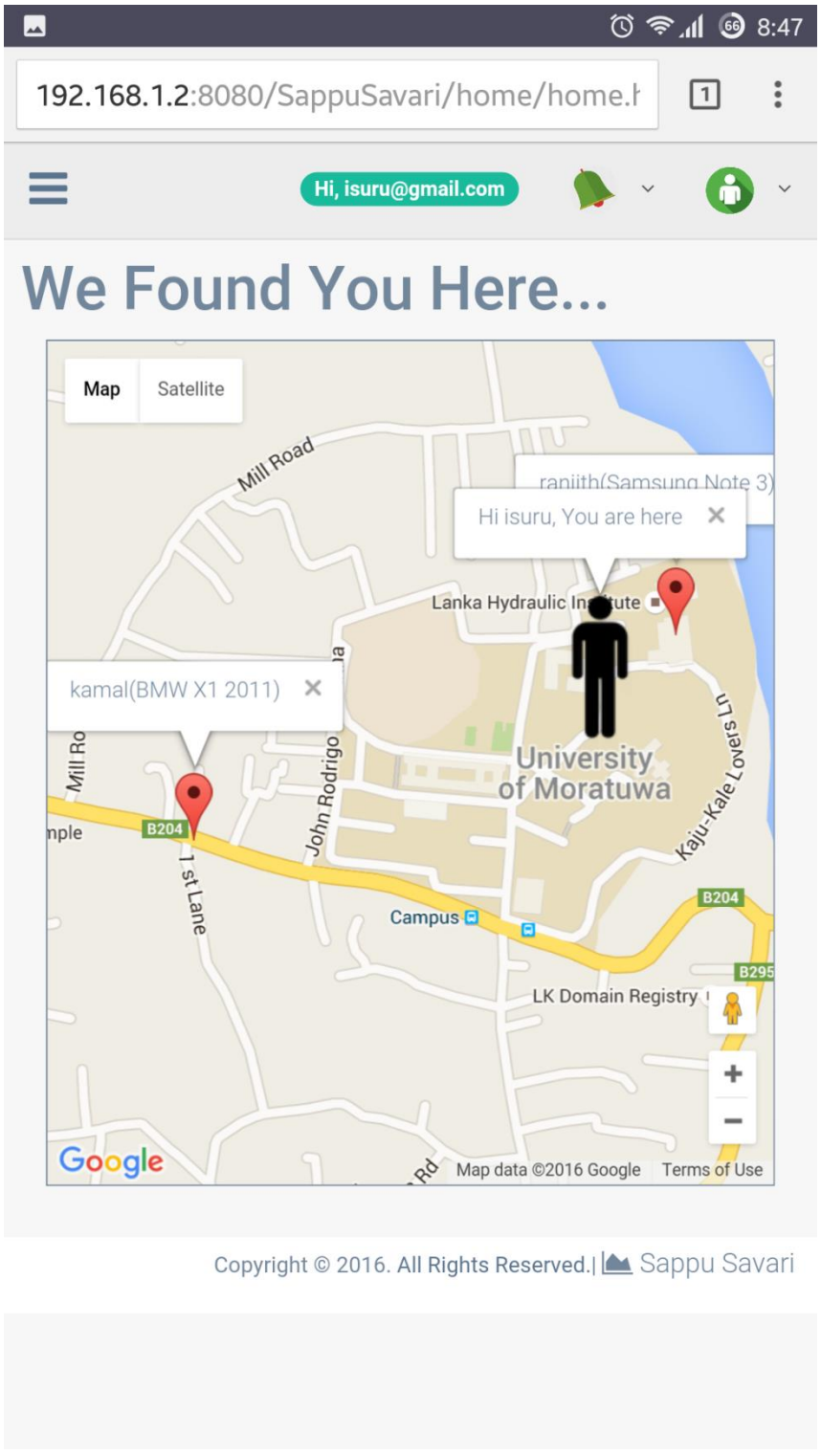

Figure 34: Home

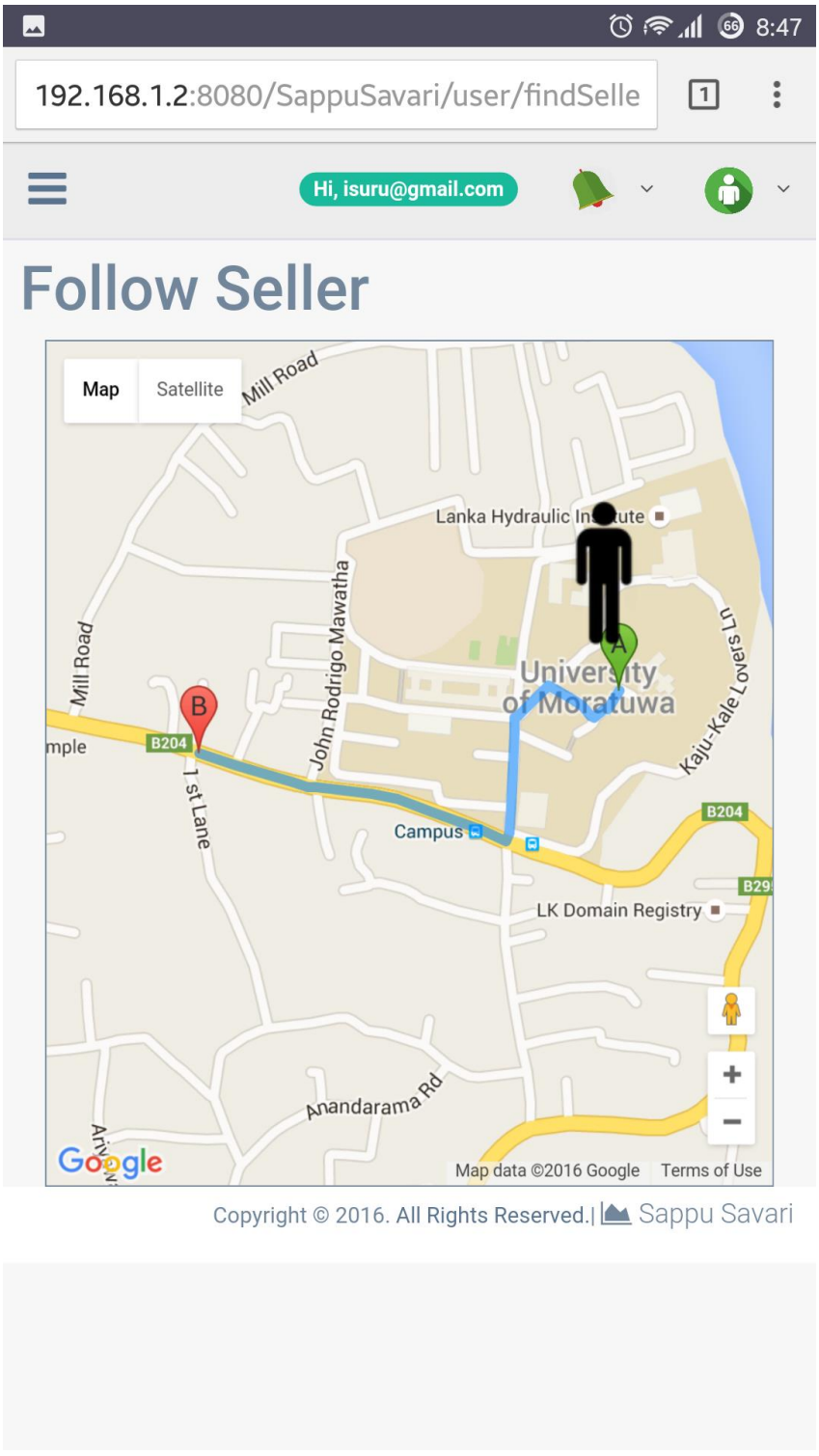

Figure 35: Follow Seller

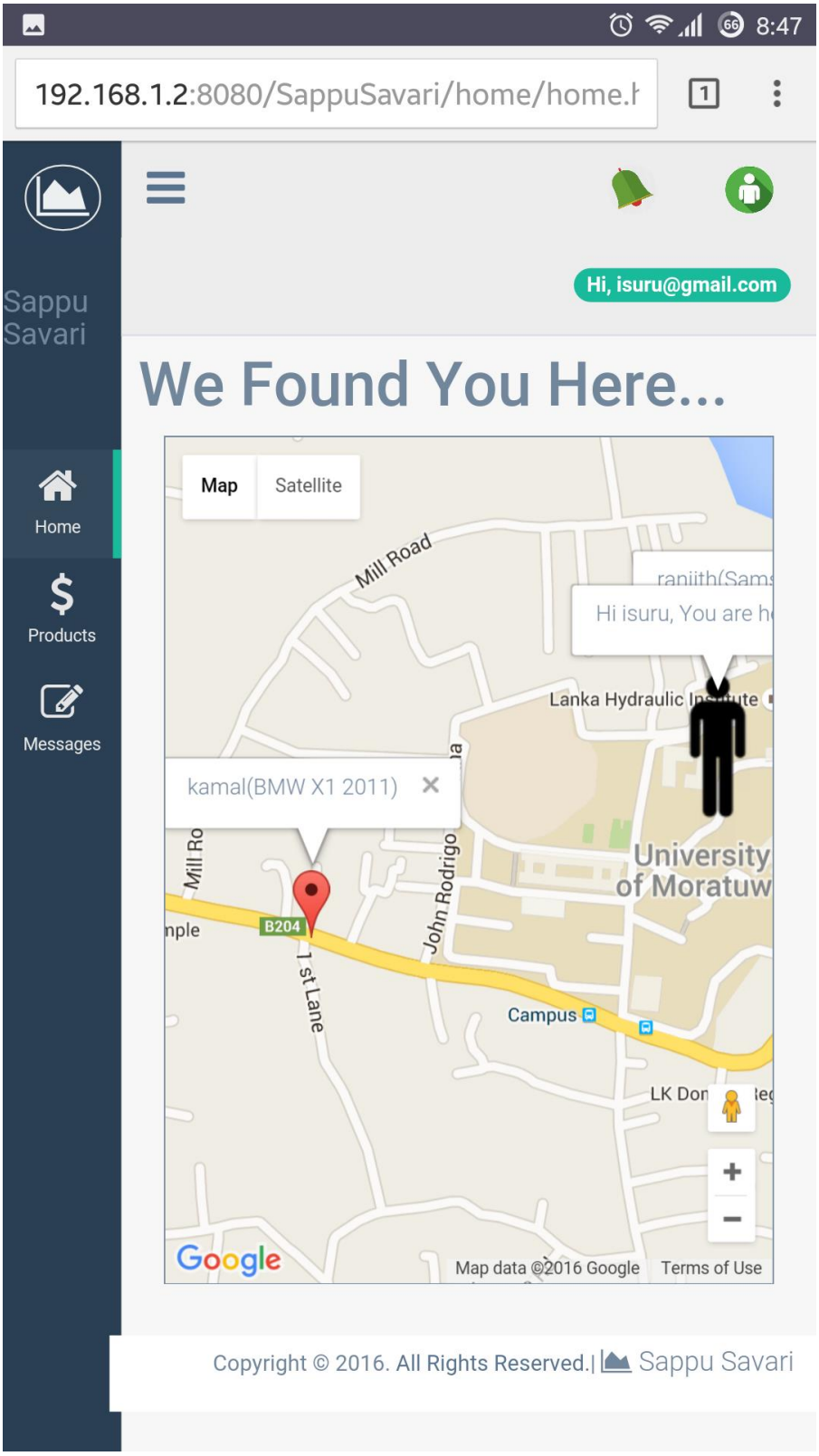

Figure 36: Home Menu

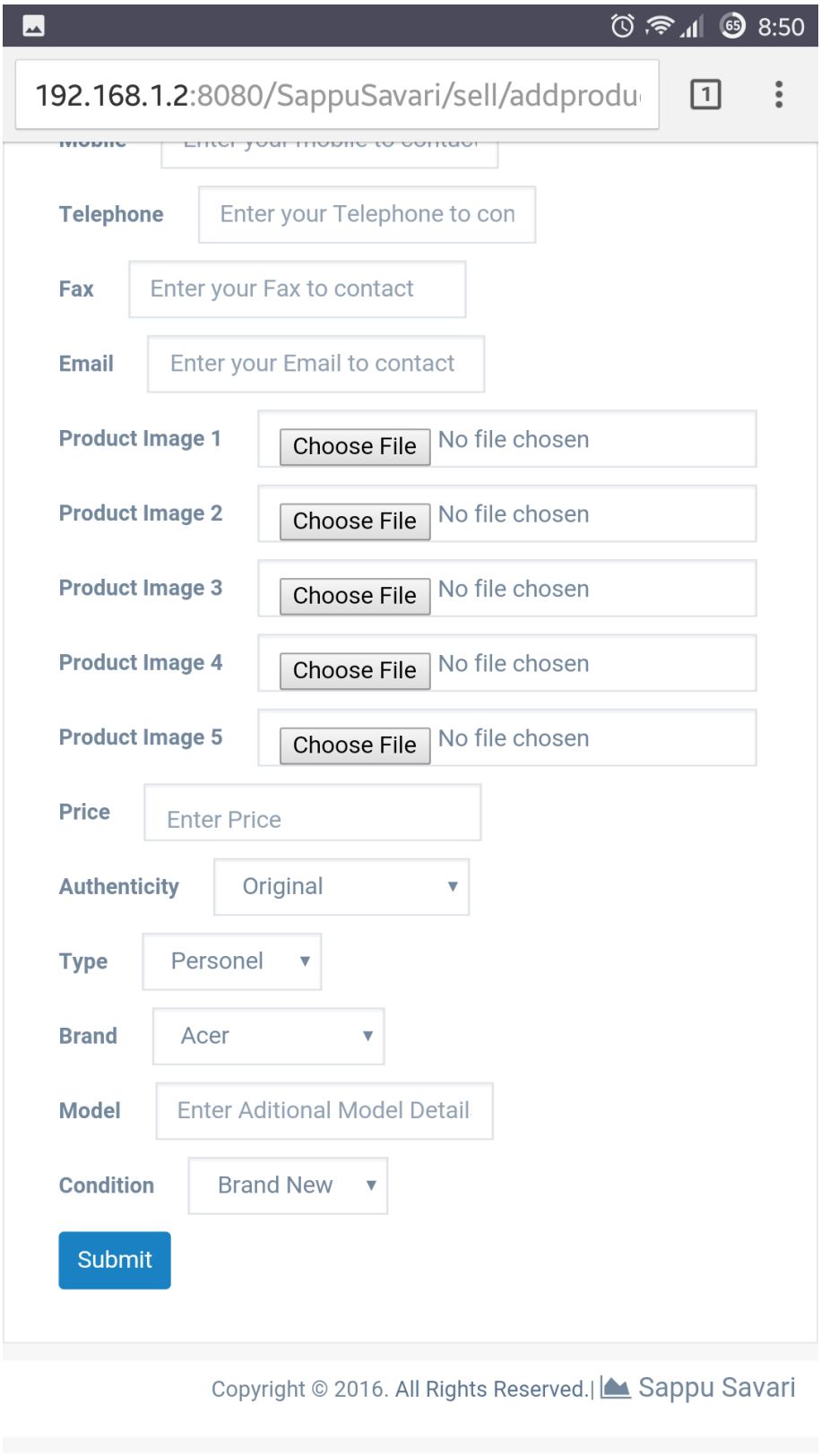

Figure 37: Add Product

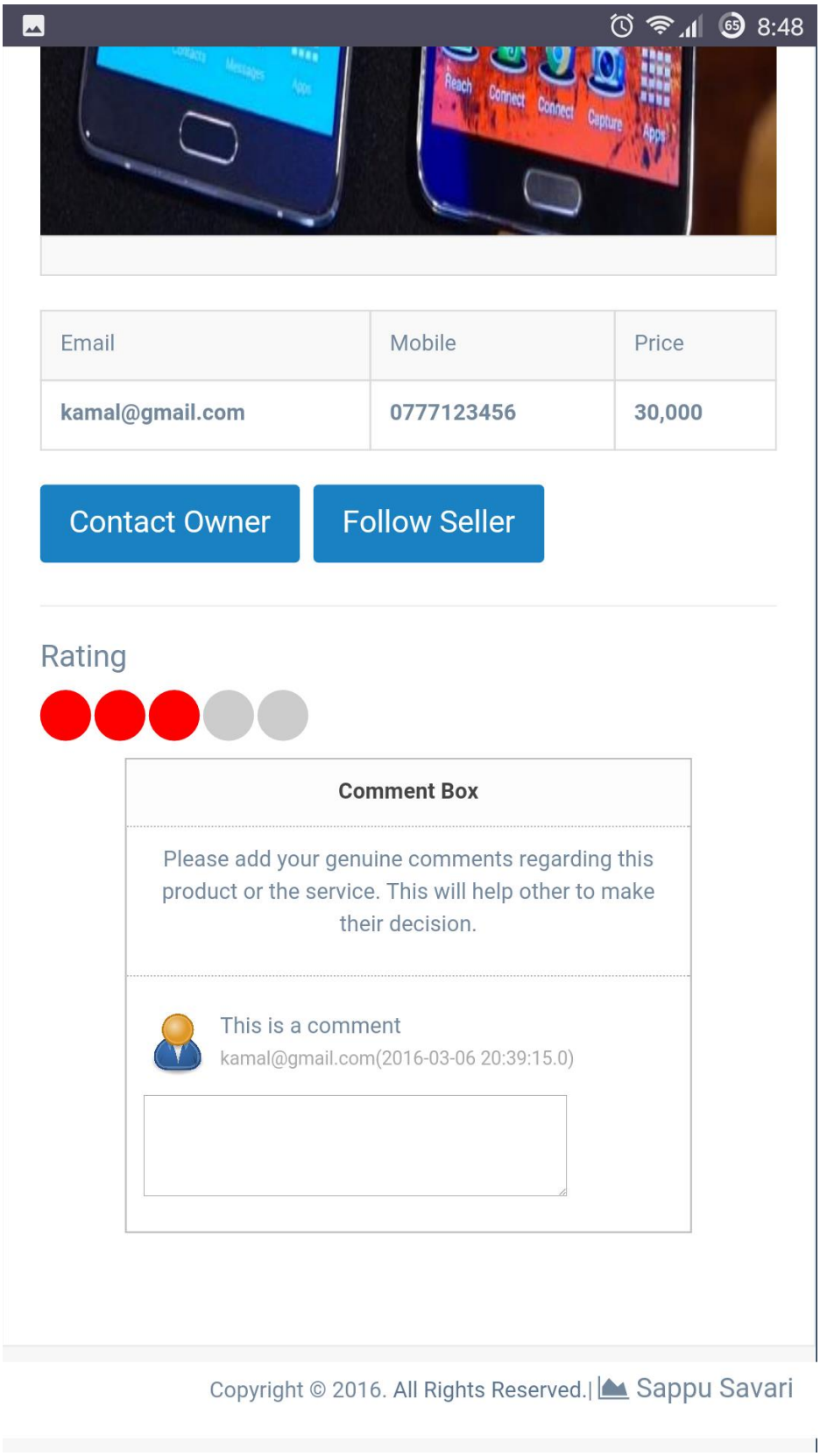

Figure 38: View Product

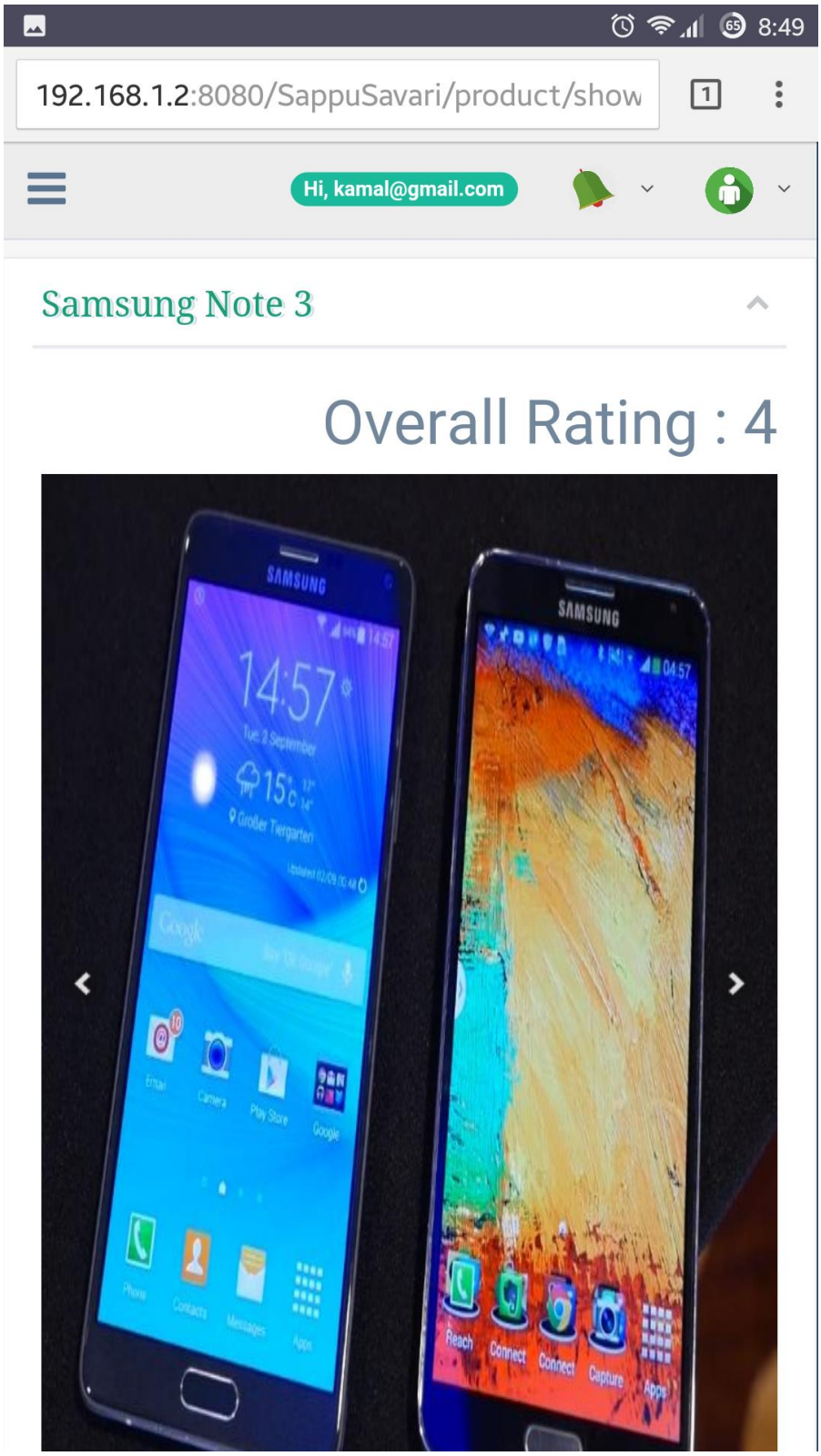

Figure 39: View Product Images

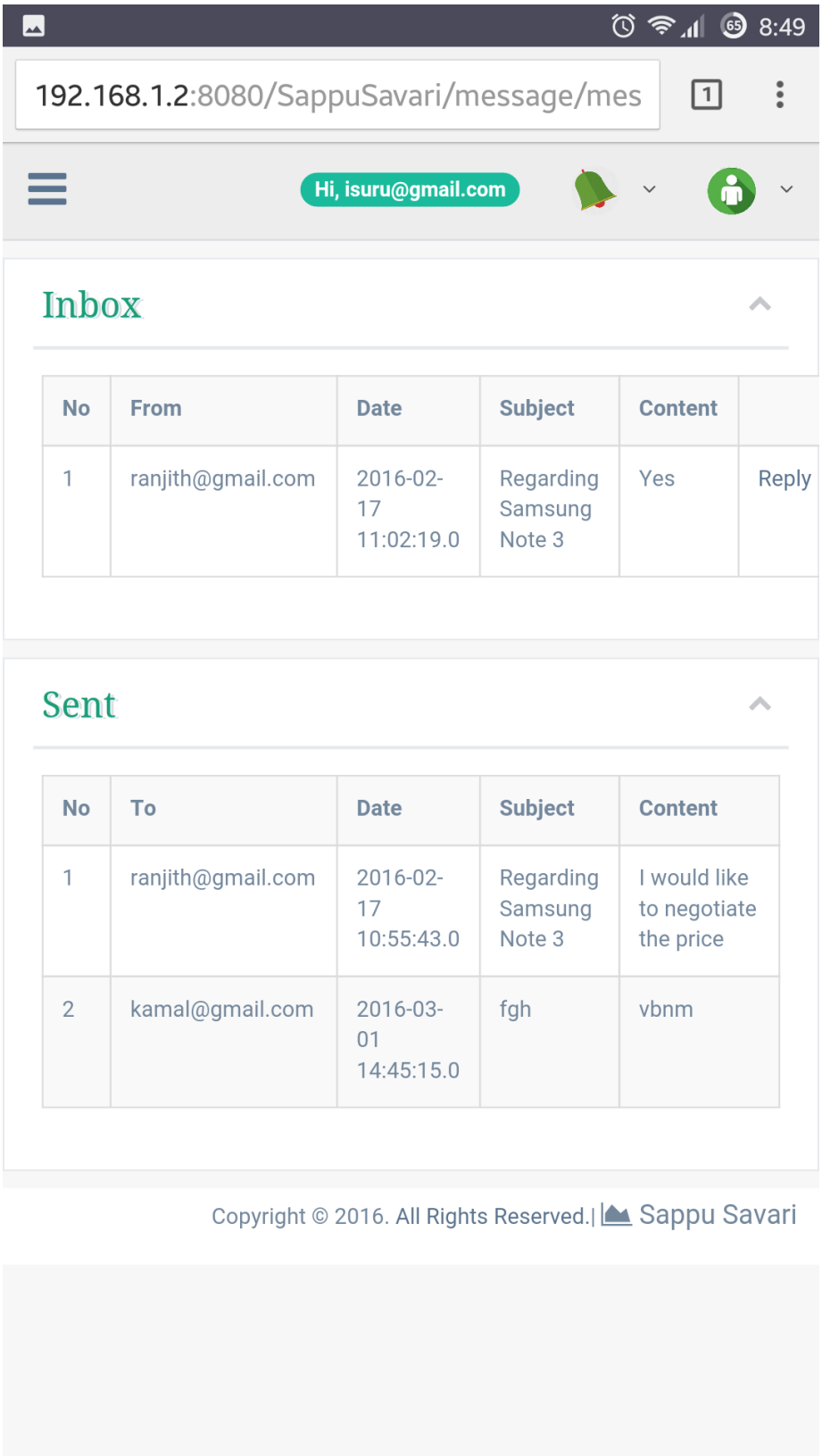

Figure 40: Message Dashboard

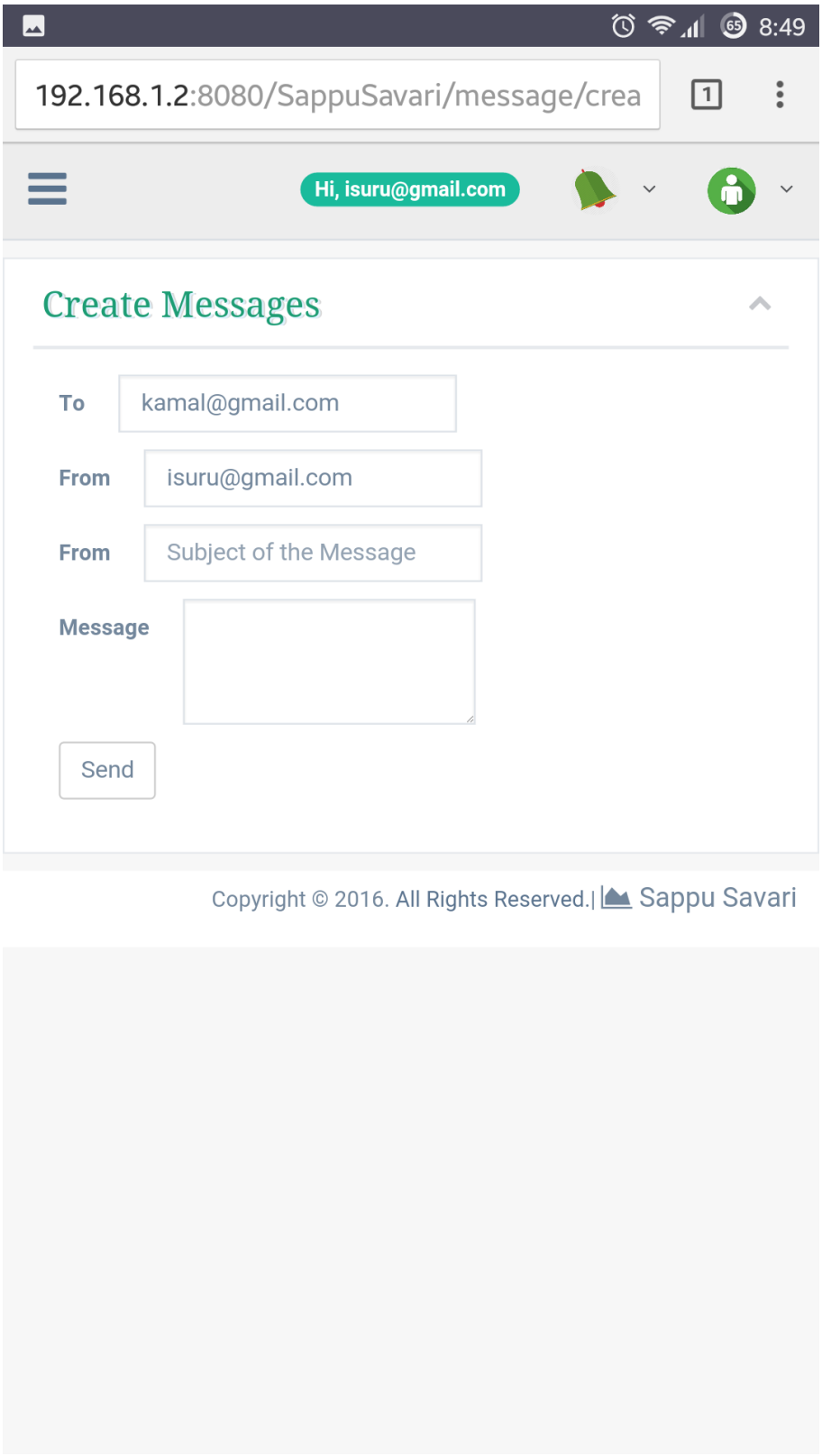

Figure 41: Create Messages

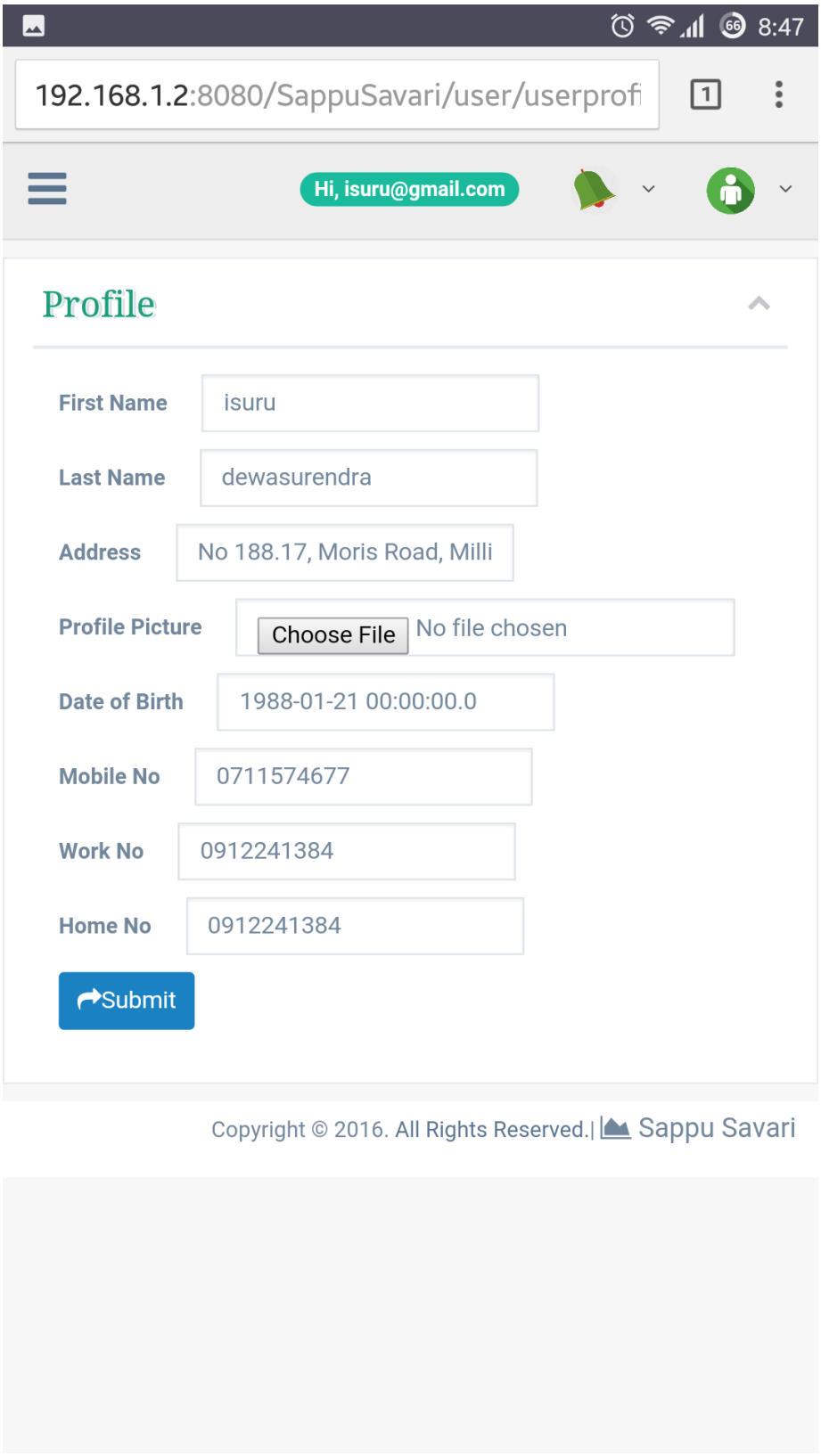

Figure 42: User Profile

9.5 Appendix E – Source Code

9.5.1 Back End Java Code

9.5.1.1 Product Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;


@Entity
@Table(name="products")
public class Product implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long productId;

	private String productTitle;

	private String description;

	private String address;

	private String mobile;

	private String landPhone;

	private String fax;

	private String email;

	private double price;

	private String productType;

	private String productBrand;
```

```
private String productModel;

private String authenticity;

private String modelYear;

private String productCondition;

private String millege;

private String transmission;

private String fuelType;

private String bodyType;

private String productSize;

private int noOfBathroom;

private int noOfRoom;

private String productStatus;

private Date createdDateTime;

private Date updatedDateTime;

@ManyToOne
private SystemUser user;

@ManyToOne
private PopularArea popularArea;

@OneToMany(mappedBy="product")
@Cascade(CascadeType.ALL)
private List<Favorite> favoriteList;

@OneToMany(mappedBy="product")
@Cascade(CascadeType.ALL)
private List<Comment> commentList;

@OneToMany(mappedBy="product")
@Cascade(CascadeType.ALL)
private List<Rating> ratingList;

@ManyToOne
private ProductSubCategory productSubCategory;

@OneToMany(mappedBy="product" ,fetch = FetchType.EAGER)
@Cascade(CascadeType.ALL)
private List<ProductMultimedia> multiMediaList;


public long getProductId() {
```

```
        return productId;
}

public void setProductId(long productId) {
        this.productId = productId;
}

public PopularArea getPopularArea() {
        return popularArea;
}

public void setPopularArea(PopularArea popularArea) {
        this.popularArea = popularArea;
}

public List<Favorite> getFavoriteList() {
        return favoriteList;
}

public void setFavoriteList(List<Favorite> favoriteList) {
        this.favoriteList = favoriteList;
}

public List<Comment> getCommentList() {
        return commentList;
}

public void setCommentList(List<Comment> commentList) {
        this.commentList = commentList;
}

public List<Rating> getRatingList() {
        return ratingList;
}

public void setRatingList(List<Rating> ratingList) {
        this.ratingList = ratingList;
}

public String getProductTitle() {
        return productTitle;
}

public void setProductTitle(String productTitle) {
        this.productTitle = productTitle;
}

public String getDescription() {
        return description;
}

public void setDescription(String description) {
        this.description = description;
}
```

```
public String getProductType() {
	return productType;
}

public void setProductType(String productType) {
	this.productType = productType;
}

public String getProductBrand() {
	return productBrand;
}

public void setProductBrand(String productBrand) {
	this.productBrand = productBrand;
}

public String getProductModel() {
	return productModel;
}

public void setProductModel(String productModel) {
	this.productModel = productModel;
}

public String getAuthenticity() {
	return authenticity;
}

public void setAuthenticity(String authenticity) {
	this.authenticity = authenticity;
}

public String getModelYear() {
	return modelYear;
}

public void setModelYear(String modelYear) {
	this.modelYear = modelYear;
}

public String getProductCondition() {
	return productCondition;
}

public void setProductCondition(String productCondition) {
	this.productCondition = productCondition;
}

public String getMillege() {
	return millege;
}

public void setMillege(String millege) {
	this.millege = millege;
}
```

```
	public String getTransmission() {
		return transmission;
	}

	public void setTransmission(String transmission) {
		this.transmission = transmission;
	}

	public String getBodyType() {
		return bodyType;
	}

	public void setBodyType(String bodyType) {
		this.bodyType = bodyType;
	}

	public String getProductSize() {
		return productSize;
	}

	public void setProductSize(String productSize) {
		this.productSize = productSize;
	}

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	public int getNoOfBathroom() {
		return noOfBathroom;
	}

	public void setNoOfBathroom(int noOfBathroom) {
		this.noOfBathroom = noOfBathroom;
	}

	public int getNoOfRoom() {
		return noOfRoom;
	}

	public void setNoOfRoom(int noOfRoom) {
		this.noOfRoom = noOfRoom;
	}

	public ProductSubCategory getProductSubCategory() {
		return productSubCategory;
	}

	public void setProductSubCategory(ProductSubCategory productSubCategory)
{
```

```
        this.productSubCategory = productSubCategory;
    }

    public List<ProductMultimedia> getMultiMediaList() {
        return multiMediaList;
    }

    public void setMultiMediaList(List<ProductMultimedia> multiMediaList) {
        this.multiMediaList = multiMediaList;
    }

    public Date getUpdatedDateTime() {
        return updatedDateTime;
    }

    public void setUpdatedDateTime(Date updatedDateTime) {
        this.updatedDateTime = updatedDateTime;
    }

    public SystemUser getUser() {
        return user;
    }

    public void setUser(SystemUser user) {
        this.user = user;
    }

    public String getProductStatus() {
        return productStatus;
    }

    public void setProductStatus(String productStatus) {
        this.productStatus = productStatus;
    }

    public Date getCreatedDateTime() {
        return createdDateTime;
    }

    public void setCreatedDateTime(Date createdDateTime) {
        this.createdDateTime = createdDateTime;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getFuelType() {
        return fuelType;
    }
```

```
	public void setFuelType(String fuelType) {
		this.fuelType = fuelType;
	}

	public String getMobile() {
		return mobile;
	}

	public void setMobile(String mobile) {
		this.mobile = mobile;
	}

	public String getLandPhone() {
		return landPhone;
	}

	public void setLandPhone(String landPhone) {
		this.landPhone = landPhone;
	}

	public String getFax() {
		return fax;
	}

	public void setFax(String fax) {
		this.fax = fax;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}


}
```

9.5.1.2 Product Category Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
```

```
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;


@Entity
@Table(name="product_categories")
public class ProductCategory implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long productCategoryId;

	private String categoryName;

	@OneToMany(mappedBy="productCategory")
	@Cascade(CascadeType.ALL)
	private List<ProductSubCategory> productSubCategoryList;




	public long getProductCategoryId() {
		return productCategoryId;
	}

	public void setProductCategoryId(long productCategoryId) {
		this.productCategoryId = productCategoryId;
	}

	public String getCategoryName() {
		return categoryName;
	}

	public void setCategoryName(String categoryName) {
		this.categoryName = categoryName;
	}
}
```

9.5.1.3 Product Sub Category Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
```

```
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;

@Entity
@Table(name="product_sub_categories")
public class ProductSubCategory implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long productSubCategoryId;

	private String subCategoryName;

	@OneToMany(mappedBy="productSubCategory")
	@Cascade(CascadeType.ALL)
	private List<Product> productList;

	@OneToMany(mappedBy="productSubCategory")
	@Cascade(CascadeType.ALL)
	private List<SearchRequest> searchRequestList;

	@ManyToOne
	private ProductCategory productCategory;



	public long getProductSubCategoryId() {
		return productSubCategoryId;
	}

	public void setProductSubCategoryId(long productSubCategoryId) {
		this.productSubCategoryId = productSubCategoryId;
	}

	public String getSubCategoryName() {
		return subCategoryName;
	}

	public void setSubCategoryName(String subCategoryName) {
		this.subCategoryName = subCategoryName;
	}

	public List<Product> getProductList() {
		return productList;
	}

	public void setProductList(List<Product> productList) {
		this.productList = productList;
	}
```

```
	public ProductCategory getProductCategory() {
		return productCategory;
	}

	public void setProductCategory(ProductCategory productCategory) {
		this.productCategory = productCategory;
	}

	public List<SearchRequest> getSearchRequestList() {
		return searchRequestList;
	}

	public void setSearchRequestList(List<SearchRequest> searchRequestList)
	{
		this.searchRequestList = searchRequestList;
	}

}
```

9.5.1.4 Product Multimedia Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name="product_multimedia")
public class ProductMultimedia implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long productMultimediaId;

	private String multimediaType;

	private String media;

	private Date updatedDateTime;

	@ManyToOne
	private Product product;
```

```
	public long getProductMultimediaId() {
		return productMultimediaId;
	}

	public void setProductMultimediaId(long productMultimediaId) {
		this.productMultimediaId = productMultimediaId;
	}

	public String getMultimediaType() {
		return multimediaType;
	}

	public void setMultimediaType(String multimediaType) {
		this.multimediaType = multimediaType;
	}

	public String getMedia() {
		return media;
	}

	public void setMedia(String media) {
		this.media = media;
	}

	public Date getUpdatedDateTime() {
		return updatedDateTime;
	}

	public void setUpdatedDateTime(Date updatedDateTime) {
		this.updatedDateTime = updatedDateTime;
	}

	public Product getProduct() {
		return product;
	}

	public void setProduct(Product product) {
		this.product = product;
	}

}
```

9.5.1.5 Product Search Request Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;
```

```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name="search_request")
public class SearchRequest implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long searchRequestId;

	private String title;

	private String description;

	private String address;

	private double minPrice;

	private double maxPrice;

	private String authenticity;

	private String productType;

	private String brand;

	private String model;

	private String productCondition;

	private String modelYear;

	private int minMillege;

	private int maxMillege;

	private String transmission;

	private String fuelType;

	private String bodyType;

	private int minNoOfBathroom;

	private int maxNoOfBathroom;

	private int minNoOfRoom;
```

```
private int maxNoOfRoom;

@ManyToOne
private PopularArea popularArea;

@ManyToOne
private ProductSubCategory productSubCategory;

@ManyToOne
private SystemUser user;

private Date createdDateTime;



public long getSearchRequestId() {
	return searchRequestId;
}

public void setSearchRequestId(long searchRequestId) {
	this.searchRequestId = searchRequestId;
}

public String getTitle() {
	return title;
}

public void setTitle(String title) {
	this.title = title;
}

public String getDescription() {
	return description;
}

public void setDescription(String description) {
	this.description = description;
}

public String getAddress() {
	return address;
}

public void setAddress(String address) {
	this.address = address;
}

public double getMinPrice() {
	return minPrice;
}

public void setMinPrice(double minPrice) {
	this.minPrice = minPrice;
}
```

```
public double getMaxPrice() {
	return maxPrice;
}

public void setMaxPrice(double maxPrice) {
	this.maxPrice = maxPrice;
}

public String getAuthenticity() {
	return authenticity;
}

public void setAuthenticity(String authenticity) {
	this.authenticity = authenticity;
}

public String getProductType() {
	return productType;
}

public void setProductType(String productType) {
	this.productType = productType;
}

public String getBrand() {
	return brand;
}

public void setBrand(String brand) {
	this.brand = brand;
}

public String getProductCondition() {
	return productCondition;
}

public void setProductCondition(String productCondition) {
	this.productCondition = productCondition;
}

public SystemUser getUser() {
	return user;
}

public void setUser(SystemUser user) {
	this.user = user;
}

public Date getCreatedDateTime() {
	return createdDateTime;
}

public void setCreatedDateTime(Date createdDateTime) {
	this.createdDateTime = createdDateTime;
}
```

```
public String getModelYear() {
	return modelYear;
}

public void setModelYear(String modelYear) {
	this.modelYear = modelYear;
}

public int getMinMillege() {
	return minMillege;
}

public void setMinMillege(int minMillege) {
	this.minMillege = minMillege;
}

public int getMaxMillege() {
	return maxMillege;
}

public void setMaxMillege(int maxMillege) {
	this.maxMillege = maxMillege;
}

public PopularArea getPopularArea() {
	return popularArea;
}

public void setPopularArea(PopularArea popularArea) {
	this.popularArea = popularArea;
}

public ProductSubCategory getProductSubCategory() {
	return productSubCategory;
}

public void setProductSubCategory(ProductSubCategory productSubCategory)
{
	this.productSubCategory = productSubCategory;
}

public String getTransmission() {
	return transmission;
}

public void setTransmission(String transmission) {
	this.transmission = transmission;
}

public String getFuelType() {
	return fuelType;
}

public void setFuelType(String fuelType) {
```

```
		this.fuelType = fuelType;
	}

	public String getBodyType() {
		return bodyType;
	}

	public void setBodyType(String bodyType) {
		this.bodyType = bodyType;
	}

	public int getMinNoOfBathroom() {
		return minNoOfBathroom;
	}

	public void setMinNoOfBathroom(int minNoOfBathroom) {
		this.minNoOfBathroom = minNoOfBathroom;
	}

	public int getMaxNoOfBathroom() {
		return maxNoOfBathroom;
	}

	public void setMaxNoOfBathroom(int maxNoOfBathroom) {
		this.maxNoOfBathroom = maxNoOfBathroom;
	}

	public int getMinNoOfRoom() {
		return minNoOfRoom;
	}

	public void setMinNoOfRoom(int minNoOfRoom) {
		this.minNoOfRoom = minNoOfRoom;
	}

	public int getMaxNoOfRoom() {
		return maxNoOfRoom;
	}

	public void setMaxNoOfRoom(int maxNoOfRoom) {
		this.maxNoOfRoom = maxNoOfRoom;
	}

	public String getModel() {
		return model;
	}

	public void setModel(String model) {
		this.model = model;
	}

}
```

9.5.1.6 Notification Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name="notification")
public class Notification implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long notificationId;

	private int seenStatus;

	private String notificationType;

	private String description;

	private int priority;

	private Date lastSeenTime;

	private Date updatedDateTime;

	@ManyToOne
	private SystemUser user;

	@ManyToOne
	private Product product;


	public Notification() {
		super();
	}

	public Notification(int seenStatus, String notificationType, String
description, int priority, Date lastSeenTime, Date updatedDateTime, SystemUser
user, Product product) {
		super();
		this.seenStatus = seenStatus;
		this.notificationType = notificationType;
		this.description = description;
```

```
        this.priority = priority;
        this.updatedDateTime = updatedDateTime;
        this.lastSeenTime = lastSeenTime;
        this.user = user;
        this.product = product;
    }

    public long getNotificationId() {
        return notificationId;
    }

    public void setNotificationId(long notificationId) {
        this.notificationId = notificationId;
    }

    public int getSeenStatus() {
        return seenStatus;
    }

    public void setSeenStatus(int seenStatus) {
        this.seenStatus = seenStatus;
    }

    public String getNotificationType() {
        return notificationType;
    }

    public void setNotificationType(String notificationType) {
        this.notificationType = notificationType;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public int getPriority() {
        return priority;
    }

    public void setPriority(int priority) {
        this.priority = priority;
    }

    public Date getUpdatedDateTime() {
        return updatedDateTime;
    }

    public void setUpdatedDateTime(Date updatedDateTime) {
        this.updatedDateTime = updatedDateTime;
    }
```

```
	public SystemUser getUser() {
		return user;
	}

	public void setUser(SystemUser user) {
		this.user = user;
	}

	public Product getProduct() {
		return product;
	}

	public void setProduct(Product product) {
		this.product = product;
	}

	public Date getLastSeenTime() {
		return lastSeenTime;
	}

	public void setLastSeenTime(Date lastSeenTime) {
		this.lastSeenTime = lastSeenTime;
	}

}
```

9.5.1.7 Rating Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;


@Entity
@Table(name="ratings")
public class Rating implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long ratingId;

	private int rating;

	private Date updatedDateTime;
```

```
	@ManyToOne
	private SystemUser user;

	@ManyToOne
	private Product product;



	public long getRatingId() {
		return ratingId;
	}

	public void setRatingId(long ratingId) {
		this.ratingId = ratingId;
	}

	public int getRating() {
		return rating;
	}

	public void setRating(int rating) {
		this.rating = rating;
	}

	public Date getUpdatedDateTime() {
		return updatedDateTime;
	}

	public void setUpdatedDateTime(Date updatedDateTime) {
		this.updatedDateTime = updatedDateTime;
	}

	public SystemUser getUser() {
		return user;
	}

	public void setUser(SystemUser user) {
		this.user = user;
	}

	public Product getProduct() {
		return product;
	}

	public void setProduct(Product product) {
		this.product = product;
	}

}
```

9.5.1.8 Message Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name="messages")
public class Message implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long messageId;

	private String subject;

	private String message;

	private Date createdDateTime;

	private Date updatedDateTime;

	private String readStatus;

	@ManyToOne
	private SystemUser sender;

	@ManyToOne
	private SystemUser receiver;

	private long productId;


	public long getMessageId() {
		return messageId;
	}

	public void setMessageId(long messageId) {
		this.messageId = messageId;
	}

	public String getMessage() {
		return message;
	}
```

```
public void setMessage(String message) {
	this.message = message;
}

public Date getCreatedDateTime() {
	return createdDateTime;
}

public void setCreatedDateTime(Date createdDateTime) {
	this.createdDateTime = createdDateTime;
}

public Date getUpdatedDateTime() {
	return updatedDateTime;
}

public void setUpdatedDateTime(Date updatedDateTime) {
	this.updatedDateTime = updatedDateTime;
}

public SystemUser getSender() {
	return sender;
}

public void setSender(SystemUser sender) {
	this.sender = sender;
}

public SystemUser getReceiver() {
	return receiver;
}

public void setReceiver(SystemUser receiver) {
	this.receiver = receiver;
}

public String getReadStatus() {
	return readStatus;
}

public void setReadStatus(String readStatus) {
	this.readStatus = readStatus;
}

public String getSubject() {
	return subject;
}

public void setSubject(String subject) {
	this.subject = subject;
}

public long getProductId() {
	return productId;
```

```
	}

	public void setProductId(long productId) {
		this.productId = productId;
	}


}
```

9.5.1.9 Comment Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;


@Entity
@Table(name="comments")
public class Comment implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long commentId;

	@Column(length=10000)
	private String comment;

	private String showStatus;

	private Date updatedDateTime;

	@ManyToOne
	private SystemUser user;

	@ManyToOne
	private Product product;


	public long getCommentId() {
		return commentId;
	}

	public void setCommentId(long commentId) {
```

```
		this.commentId = commentId;
	}

	public String getComment() {
		return comment;
	}

	public void setComment(String comment) {
		this.comment = comment;
	}

	public String getShowStatus() {
		return showStatus;
	}

	public void setShowStatus(String showStatus) {
		this.showStatus = showStatus;
	}

	public Date getUpdatedDateTime() {
		return updatedDateTime;
	}

	public void setUpdatedDateTime(Date updatedDateTime) {
		this.updatedDateTime = updatedDateTime;
	}

	public SystemUser getUser() {
		return user;
	}

	public void setUser(SystemUser user) {
		this.user = user;
	}

	public Product getProduct() {
		return product;
	}

	public void setProduct(Product product) {
		this.product = product;
	}

}
```

9.5.1.10 User Domain

```
package com.isd.sappu.savari.domains;

import java.io.Serializable;
import java.util.Date;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;

@Entity
@Table(name="users")
public class SystemUser implements Serializable{

	private static final long serialVersionUID = 5489679679570043539L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long userId;

	private String username;

	private String password;

	private String firstName;

	private String lastName;

	private String address;

	private String profilePic;

	private Date dateOfBirth;

	private String mobileNo;

	private String workNo;

	private String homeNo;

	private double latitude;

	private double longtitude;

	private int accuracy;

	private int activeStatus;

	@OneToMany(mappedBy="user")
	@Cascade(CascadeType.ALL)
	private List<Favorite> favoriteList;
```

```
@OneToMany(mappedBy="user")
@Cascade(CascadeType.ALL)
private List<Comment> commentList;

@OneToMany(mappedBy="user")
@Cascade(CascadeType.ALL)
private List<Rating> ratingList;

@OneToMany(mappedBy="sender")
@Cascade(CascadeType.ALL)
private List<Message> messageSentList;

@OneToMany(mappedBy="receiver")
@Cascade(CascadeType.ALL)
private List<Message> messageRecievedList;

@OneToMany(mappedBy="user")
@Cascade(CascadeType.ALL)
private List<Product> productList;

@OneToMany(mappedBy="user")
@Cascade(CascadeType.ALL)
private List<SystemUserRole> systemUserRoleList;

@OneToMany(mappedBy="user")
@Cascade(CascadeType.ALL)
private List<SearchRequest> searchRequestList;



public long getUserId() {
	return userId;
}

public void setUserId(long userId) {
	this.userId = userId;
}

public String getUsername() {
	return username;
}

public void setUsername(String username) {
	this.username = username;
}

public String getPassword() {
	return password;
}

public void setPassword(String password) {
	this.password = password;
}

public String getFirstName() {
```

```
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getProfilePic() {
        return profilePic;
    }

    public void setProfilePic(String profilePic) {
        this.profilePic = profilePic;
    }

    public Date getDateOfBirth() {
        return dateOfBirth;
    }

    public void setDateOfBirth(Date dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    public String getMobileNo() {
        return mobileNo;
    }

    public void setMobileNo(String mobileNo) {
        this.mobileNo = mobileNo;
    }

    public String getWorkNo() {
        return workNo;
    }

    public void setWorkNo(String workNo) {
        this.workNo = workNo;
    }
```

```
public String getHomeNo() {
	return homeNo;
}

public void setHomeNo(String homeNo) {
	this.homeNo = homeNo;
}

public List<Favorite> getFavoriteList() {
	return favoriteList;
}

public void setFavoriteList(List<Favorite> favoriteList) {
	this.favoriteList = favoriteList;
}

public List<Comment> getCommentList() {
	return commentList;
}

public void setCommentList(List<Comment> commentList) {
	this.commentList = commentList;
}

public List<Rating> getRatingList() {
	return ratingList;
}

public void setRatingList(List<Rating> ratingList) {
	this.ratingList = ratingList;
}

public List<Message> getMessageSentList() {
	return messageSentList;
}

public void setMessageSentList(List<Message> messageSentList) {
	this.messageSentList = messageSentList;
}

public List<Message> getMessageRecievedList() {
	return messageRecievedList;
}

public void setMessageRecievedList(List<Message> messageRecievedList) {
	this.messageRecievedList = messageRecievedList;
}

public List<Product> getProductList() {
	return productList;
}

public void setProductList(List<Product> productList) {
	this.productList = productList;
}
```

```
	public List<SystemUserRole> getSystemUserRoleList() {
		return systemUserRoleList;
	}

	public void setSystemUserRoleList(List<SystemUserRole>
systemUserRoleList) {
		this.systemUserRoleList = systemUserRoleList;
	}

	public int getActiveStatus() {
		return activeStatus;
	}

	public void setActiveStatus(int activeStatus) {
		this.activeStatus = activeStatus;
	}

	public List<SearchRequest> getSearchRequestList() {
		return searchRequestList;
	}

	public void setSearchRequestList(List<SearchRequest> searchRequestList)
{
		this.searchRequestList = searchRequestList;
	}

	public double getLatitude() {
		return latitude;
	}

	public void setLatitude(double latitude) {
		this.latitude = latitude;
	}

	public double getLongtitude() {
		return longtitude;
	}

	public void setLongtitude(double longtitude) {
		this.longtitude = longtitude;
	}

	public int getAccuracy() {
		return accuracy;
	}

	public void setAccuracy(int accuracy) {
		this.accuracy = accuracy;
	}

	public Map<String, Object> toMap() {
		Map<String, Object> map = new HashMap<String, Object>();
		map.put("userId", this.userId);
		map.put("username", this.username);
```

```
		map.put("firstName", this.firstName);
		map.put("address", this.address);
		map.put("mobileNo", this.mobileNo);
		map.put("latitude", this.latitude);
		map.put("longtitude", this.longtitude);
		map.put("accuracy", this.accuracy);
		map.put("activeStatus", this.activeStatus);
		return map;
	}

}
```

9.5.1.11 User Role Domain

```
package com.isd.sappu.savari.domains;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;
import org.springframework.security.core.GrantedAuthority;

@Entity
@Table(name="user_role")
public class UserRole implements GrantedAuthority {

	private static final long serialVersionUID = 6482583898800868 82L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int userRoleId;

	@OneToMany(mappedBy="userRole")
	@Cascade(CascadeType.ALL)
	private List<SystemUserRole> systemUserRoles;

	private String authority;



	public int getUserRoleId() {
		return userRoleId;
	}

	public void setUserRoleId(int userRoleId) {
```

```
		this.userRoleId = userRoleId;
	}

	public List<SystemUserRole> getSystemUserRoles() {
		return systemUserRoles;
	}

	public void setSystemUserRoles(List<SystemUserRole> systemUserRoles) {
		this.systemUserRoles = systemUserRoles;
	}

	public String getAuthority() {
		return authority;
	}

	public void setAuthority(String authority) {
		this.authority = authority;
	}

}
```

9.5.1.12 Product Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.SearchRequest;

public interface ProductService {

	public Product saveUpdateProduct(Product product);

	public Product getProductById(long productId);

	public List<Product> getAllProducts();

	public List<Product> getAllProductByUserId(long userId);

	public List<Product> getSearchedProductList(SearchRequest
searchRequest);

	public List<Product> getFavoriteProductsByUser(long userId);

	public List<Product> getCommentedProductsByUser(long userId);

}
```

9.5.1.13 Product Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.isd.sappu.savari.dao.ProductDao;
import com.isd.sappu.savari.domains.Comment;
import com.isd.sappu.savari.domains.Favorite;
import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.ProductCategory;
import com.isd.sappu.savari.domains.ProductSubCategory;
import com.isd.sappu.savari.domains.SearchRequest;
import com.isd.sappu.savari.domains.SystemUser;
import com.isd.sappu.savari.util.EnumConstant;

@Service
public class ProductServicesImpl implements ProductService{

	@Autowired
	ProductDao productDao;

	@Autowired
	FavoriteService favoriteService;

	@Autowired
	CommentService commentService;

	@Override
	public List<Product> getAllProducts() {
		return productDao.findAll();
	}

	@Override
	public List<Product> getAllProductByUserId(long userId) {
		SystemUser user = new SystemUser();
		user.setUserId(userId);
		return productDao.findProductByUser(user);
	}

	@Override
	public Product saveUpdateProduct(Product product) {
		return productDao.save(product);
	}

	@Override
	public List<Product> getSearchedProductList(SearchRequest searchRequest)
	{
```

```
                ProductSubCategory productSubCategory =
                searchRequest.getProductSubCategory();
                ProductCategory productCategory =
                productSubCategory.getProductCategory();

                List<Product> productList = new ArrayList<Product>();


if(productCategory.getCategoryName().equals(EnumConstant.ProductCategory.OTHER
.toString())){

        productList = productDao.getOtherProductList(

        searchRequest.getProductSubCategory().getProductSubCategoryId(),

        (searchRequest.getMinPrice()==0)?Double.MIN_VALUE:searchRequest.getMinPr
ice(),

        (searchRequest.getMaxPrice()==0)?Double.MAX_VALUE:searchRequest.getMaxPr
ice(),

        searchRequest.getAuthenticity(),
(searchRequest.getAuthenticity()==null)?1:0,

        searchRequest.getProductType(),
(searchRequest.getProductType()==null)?1:0,

        searchRequest.getBrand(), (searchRequest.getBrand()==null)?1:0,

        searchRequest.getProductCondition(),
(searchRequest.getProductCondition()==null)?1:0
                                );
                }else
if(productCategory.getCategoryName().equals(EnumConstant.ProductCategory.ELECT
RONICS.toString())){

        productList = productDao.getOtherProductList(

        searchRequest.getProductSubCategory().getProductSubCategoryId(),

        (searchRequest.getMinPrice()==0)?Double.MIN_VALUE:searchRequest.getMinPr
ice(),

        (searchRequest.getMaxPrice()==0)?Double.MAX_VALUE:searchRequest.getMaxPr
ice(),

        searchRequest.getAuthenticity(),
(searchRequest.getAuthenticity()==null)?1:0,

        searchRequest.getProductType(),
(searchRequest.getProductType()==null)?1:0,

        searchRequest.getBrand(), (searchRequest.getBrand()==null)?1:0,
```

```
        searchRequest.getProductCondition(),
(searchRequest.getProductCondition()==null)?1:0
                                    );
                }else
if(productCategory.getCategoryName().equals(EnumConstant.ProductCategory.VEHIC
LE.toString())){

        productList = productDao.getVehicleProductList(

        searchRequest.getProductSubCategory().getProductSubCategoryId(),

        (searchRequest.getMinPrice()==0)?Double.MIN_VALUE:searchRequest.getMinPr
ice(),

        (searchRequest.getMaxPrice()==0)?Double.MAX_VALUE:searchRequest.getMaxPr
ice(),

        searchRequest.getProductType(),
(searchRequest.getProductType()==null)?1:0,

        searchRequest.getProductCondition(),
(searchRequest.getProductCondition()==null)?1:0,

        (searchRequest.getMinMillege()==0)?Integer.MIN_VALUE:searchRequest.getMi
nMillege(),

        (searchRequest.getMaxMillege()==0)?Integer.MAX_VALUE:searchRequest.getMa
xMillege(),

        searchRequest.getTransmission(),
(searchRequest.getTransmission()==null)?1:0,

        searchRequest.getFuelType(), (searchRequest.getFuelType()==null)?1:0,

        searchRequest.getBodyType(), (searchRequest.getBodyType()==null)?1:0
                                    );
                }else
if(productCategory.getCategoryName().equals(EnumConstant.ProductCategory.PROPE
RTY.toString())){

        productList = productDao.getPropertyProductList(

        searchRequest.getProductSubCategory().getProductSubCategoryId(),

        (searchRequest.getMinPrice()==0)?Double.MIN_VALUE:searchRequest.
        getMinPrice(),

        (searchRequest.getMaxPrice()==0)?Double.MAX_VALUE:searchRequest.
        getMaxPrice(),
        searchRequest.getProductType(),
        (searchRequest.getProductType()==null)?1:0,

        searchRequest.getProductCondition(),
        (searchRequest.getProductCondition()==null)?1:0,
```

```
(searchRequest.getMinNoOfBathroom()==0)?Integer.MIN_VALUE:
searchRequest.getMinNoOfBathroom(),

(searchRequest.getMaxNoOfBathroom()==0)?Integer.MAX_VALUE:
searchRequest.getMaxNoOfBathroom(),

(searchRequest.getMinNoOfRoom()==0)?Integer.MIN_VALUE:
searchRequest.getMinNoOfRoom(),

(searchRequest.getMaxNoOfRoom()==0)?Integer.MAX_VALUE:
searchRequest.getMaxNoOfRoom()
                            );
      }

      return productList;
}

@Override
public Product getProductById(long productId) {
      return productDao.findOne(productId);
}

@Override
public List<Product> getFavoriteProductsByUser(long userId) {
      try {
            List<Favorite> favoriteList =
            favoriteService.getFavoritesByUserId(userId);
            List<Product> productList = new ArrayList<Product>();
            for (Favorite favorite : favoriteList) {
            productList.add(productDao.findOne(
            favorite.getProduct().getProductId()));
            }
            return productList;
      } catch (Exception e) {
            System.out.println("error occured when get products for
            the user's favorite selection");
            e.printStackTrace();
            return null;
      }

}

@Override
public List<Product> getCommentedProductsByUser(long userId) {
      try {
            List<Comment> commentList =
            commentService.getCommentListByUserId(userId);
            List<Product> productList = new ArrayList<Product>();
            for (Comment comment : commentList) {

            productList.add(productDao.findOne(
            comment.getProduct().getProductId()));
            }
            return productList;
```

```
		} catch (Exception e) {
			System.out.println("error occured when get products for
			the user's comments");
			e.printStackTrace();
			return null;
		}
	}

}
```

9.5.1.14 Product Category Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.ProductCategory;

public interface ProductCategoryService {

	public ProductCategory saveOrUpdateProductCategory(ProductCategory
productCategory);

	public List<ProductCategory> getAllProductCategory();

}
```

9.5.1.15 Product Category Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.isd.sappu.savari.dao.ProductCategoryDao;
import com.isd.sappu.savari.domains.ProductCategory;

@Service
public class ProductCategoryServiceImpl implements ProductCategoryService{

	@Autowired
	ProductCategoryDao productCategoryDao;

	@Override
	public ProductCategory saveOrUpdateProductCategory(ProductCategory
productCategory) {
		return productCategoryDao.save(productCategory);
	}
	@Override
	public List<ProductCategory> getAllProductCategory() {
		return productCategoryDao.findAll();
	}
}
```

9.5.1.16 Product Multimedia Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.ProductMultimedia;

public interface ProductMultimediaService {

	public List<ProductMultimedia> getMultimediaList(long productId);

}
```

9.5.1.17 Product Multimedia Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.isd.sappu.savari.dao.ProductMultimediaDao;
import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.ProductMultimedia;

@Service
public class ProductMultimediaServiceImpl implements ProductMultimediaService{

	@Autowired
	private ProductMultimediaDao productMultimediaDao;

	@Override
	public List<ProductMultimedia> getMultimediaList(long productId) {
		Product product = new Product();
		product.setProductId(productId);
		return productMultimediaDao.findByProduct(product);
	}

}
```

9.5.1.18 Location Service

```
package com.isd.sappu.savari.services;

public interface LocationService {

	public double getDistance(double fromLat, double fromLon, double toLat,
double toLon, char unit);

}
```

9.5.1.19 Location Service Implementation

```
package com.isd.sappu.savari.services;

import org.springframework.stereotype.Service;

@Service
public class LocationServiceImpl implements LocationService {

	@Override
	public double getDistance(double fromLat, double fromLon, double toLat,
double toLon, char unit) {
		double theta = fromLon - toLon;
		  double dist = Math.sin(deg2rad(fromLat)) *
Math.sin(deg2rad(toLat)) + Math.cos(deg2rad(fromLat)) *
Math.cos(deg2rad(toLat)) * Math.cos(deg2rad(theta));
		  dist = Math.acos(dist);
		  dist = rad2deg(dist);
		  dist = dist * 60 * 1.1515;
		  if (unit == 'K') {
		    dist = dist * 1.609344;
		  } else if (unit == 'N') {
		      dist = dist * 0.8684;
		    }
		  return (dist);
	}

	private double deg2rad(double deg) {
	  return (deg * Math.PI / 180.0);
	}

	private double rad2deg(double rad) {
	  return (rad * 180 / Math.PI);
	}

}
```

9.5.1.20 Search Request Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.SearchRequest;

public interface SearchRequestService {

	public SearchRequest getSearchRequestById(long searchRequestId);

	public SearchRequest saveUpdateSearchRequest(SearchRequest
searchRequest);

	public void deleteSearchRequest(long searchRequestId);

	public String deleteUserSearchRequst(long userId);
```

```
	public List<SearchRequest> getAllSearchRequestByUserId(long userId);

}
```

9.5.1.21 Search Request Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.isd.sappu.savari.dao.SearchRequestDao;
import com.isd.sappu.savari.domains.SearchRequest;
import com.isd.sappu.savari.domains.SystemUser;
import com.isd.sappu.savari.util.EnumConstant;

@Service
public class SearchRequestServiceImpl implements SearchRequestService{

	@Autowired
	SearchRequestDao searchRequestDao;

	@Override
	public SearchRequest getSearchRequestById(long searchRequestId) {
		return searchRequestDao.findOne(searchRequestId);
	}

	@Override
	public SearchRequest saveUpdateSearchRequest(SearchRequest
searchRequest) {
		return searchRequestDao.save(searchRequest);
	}

	@Override
	public List<SearchRequest> getAllSearchRequestByUserId(long userId) {
		SystemUser user = new SystemUser();
		user.setUserId(userId);
		return searchRequestDao.findByUser(user);
	}

	@Override
	public void deleteSearchRequest(long searchRequestId) {
		searchRequestDao.delete(searchRequestId);
	}

	@Override
	public String deleteUserSearchRequst(long userId) {
		List<SearchRequest> searchRequestList =
this.getAllSearchRequestByUserId(userId);
		for (SearchRequest searchRequest : searchRequestList) {
```

```
		this.deleteSearchRequest(searchRequest.getSearchRequestId());
			}
			return EnumConstant.ReturnStatus.SUCCESS.toString();
		}

}
```

9.5.1.22 Rating Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.Rating;

public interface RatingService {

	public Rating saveUpdateRating(Rating rating);

	public Rating getRating(long ratingId);

	public Rating getRating(long userId, long productId);

	public List<Rating> getRatingsByUserId(long userId);

	public List<Rating> getRatingsByProductId(long productId);

	public void deleteRating(Rating rating);

	public int calculateOverallRating(long productId);

}
```

9.5.1.23 Rating Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.isd.sappu.savari.dao.RatingDao;
import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.Rating;
import com.isd.sappu.savari.domains.SystemUser;

@Service
public class RatingServiceImpl implements RatingService {

	@Autowired
	private RatingDao ratingDao;
```

```
	@Override
	public Rating saveUpdateRating(Rating rating) {
		Rating rate = this.getRating(rating.getUser().getUserId(),
rating.getProduct().getProductId());
		if(rate != null){
			rate.setRating(rating.getRating());
			return ratingDao.save(rate);
		}else{
			return ratingDao.save(rating);
		}
	}

	@Override
	public Rating getRating(long ratingId) {
		return ratingDao.findByRatingId(ratingId);
	}

	@Override
	public List<Rating> getRatingsByUserId(long userId) {
		SystemUser user = new SystemUser();
		user.setUserId(userId);
		return ratingDao.findByUser(user);
	}

	@Override
	public List<Rating> getRatingsByProductId(long productId) {
		Product product = new Product();
		product.setProductId(productId);
		return ratingDao.findByProduct(product);
	}

	@Override
	public void deleteRating(Rating rating) {
		ratingDao.delete(rating);
	}

	@Override
	public int calculateOverallRating(long productId) {
		List<Rating> ratingList = this.getRatingsByProductId(productId);
		int ratingSum = 0;
		int ratingCount = 0;

		for (Rating rating : ratingList) {
			ratingCount++;
			ratingSum = ratingSum + rating.getRating();
		}

		if(ratingCount>0 && ratingSum>0){
			return ratingSum/ratingCount;
		}else{
			return 0;
		}
	}

	@Override
```

```
	public Rating getRating(long userId, long productId) {
		SystemUser user = new SystemUser();
		user.setUserId(userId);
		Product product = new Product();
		product.setProductId(productId);
		return ratingDao.findByUserAndProduct(user, product);
	}

}
```

9.5.1.24 Comment Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.Comment;

public interface CommentService {

	public Comment saveUpdateComment(Comment comment);

	public Comment getComment(long commentId);

	public void deleteComment(Comment comment);

	public List<Comment> getCommentListByProductId(long productId);

	public List<Comment> getCommentListByUserId(long userId);

}
```

9.5.1.25 Comment Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.isd.sappu.savari.dao.CommentDao;
import com.isd.sappu.savari.domains.Comment;
import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.SystemUser;

@Service
public class CommentServiceImpl implements CommentService{

	@Autowired
	private CommentDao commentDao;

	@Override
	public Comment saveUpdateComment(Comment comment) {
```

```
		return commentDao.save(comment);
	}

	@Override
	public Comment getComment(long commentId) {
		return commentDao.findByCommentId(commentId);
	}

	@Override
	public void deleteComment(Comment comment) {
		commentDao.delete(comment);
	}

	@Override
	public List<Comment> getCommentListByProductId(long productId) {
		Product product = new Product();
		product.setProductId(productId);
		return commentDao.findCommentByProduct(product);
	}

	@Override
	public List<Comment> getCommentListByUserId(long userId) {
		SystemUser systemUser = new SystemUser();
		systemUser.setUserId(userId);
		return commentDao.findCommentByUser(systemUser);
	}

}
```

9.5.1.26 Message Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.Message;

public interface MessageService {

	public Message saveUpdateMessage(Message message);

	public Message getMessage(long messageId);

	public List<Message> getMessagesBySender(long userId);

	public List<Message> getMessagesByReceiver(long userId);

	public void deleteMessage(Message message);

}
```

9.5.1.27 Message Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import com.isd.sappu.savari.dao.MessageDao;
import com.isd.sappu.savari.domains.Message;
import com.isd.sappu.savari.domains.SystemUser;

@Repository
@Transactional
public class MessageServiceImpl implements MessageService {

	@Autowired
	private MessageDao messageDao;

	@Override
	public Message saveUpdateMessage(Message message) {
		return messageDao.save(message);
	}

	@Override
	public Message getMessage(long messageId) {
		return messageDao.findOne(messageId);
	}

	@Override
	public List<Message> getMessagesBySender(long userId) {
		SystemUser sender = new SystemUser();
		sender.setUserId(userId);
		return messageDao.findBySender(sender);
	}

	@Override
	public List<Message> getMessagesByReceiver(long userId) {
		SystemUser reciever = new SystemUser();
		reciever.setUserId(userId);
		return messageDao.findByReceiver(reciever);
	}

	@Override
	public void deleteMessage(Message message) {
		messageDao.delete(message);
	}

}
```

9.5.1.28 User Service

```
package com.isd.sappu.savari.services;

import java.util.List;

import com.isd.sappu.savari.domains.SystemUser;

public interface SystemUserService {

	public SystemUser saveUpdateSystemUser(SystemUser user);

	public SystemUser getSystemUserByUsername(String username);

	public SystemUser getSystemUser(long userId);

	public void deleteSystemUser(SystemUser user);

	public List<SystemUser> getAllSystemUsers();

}
```

9.5.1.29 User Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.isd.sappu.savari.dao.SystemUserDao;
import com.isd.sappu.savari.domains.SystemUser;

@Service
public class SystemUserServiceImpl implements SystemUserService{

	@Autowired
	private SystemUserDao systemUserDao;

	@Override
	public SystemUser saveUpdateSystemUser(SystemUser user) {
		return systemUserDao.save(user);
	}

	@Override
	public SystemUser getSystemUser(long userId) {
		return systemUserDao.findOne(userId);
	}

	@Override
	public void deleteSystemUser(SystemUser user) {
		systemUserDao.delete(user);
	}
```

```
	@Override
	public SystemUser getSystemUserByUsername(String username) {
		return systemUserDao.findByUsername(username);
	}

	@Override
	public List<SystemUser> getAllSystemUsers() {
		return systemUserDao.findAll();
	}

}
```

9.5.1.30 Scheduler Service Implementation

```
package com.isd.sappu.savari.services;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

import com.isd.sappu.savari.domains.Notification;
import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.SearchRequest;
import com.isd.sappu.savari.domains.SystemUser;
import com.isd.sappu.savari.util.EnumConstant;

@Component
public class ScheduleLocationSearchService {

	@Autowired
	SystemUserService systemUserService;

	@Autowired
	ProductService productService;

	@Autowired
	SearchRequestService searchRequestService;

	@Autowired
	NotificationService notificationService;

	@Scheduled(fixedDelay = 1000 * 20)
	public void demoServiceMethod() {
			System.out.println("****************Schedular started
			running at " + new Date() + "**********************");

		// get application users
				List<SystemUser> applicationUserList =
		systemUserService.getAllSystemUsers();
```

```
for (SystemUser applicationUser : applicationUserList) {

	List<Product> relevantProducts = new ArrayList<Product>();
                        List<Product> searchedProductList = new
	ArrayList<Product>();

	// get favorite products
                        List<Product> favoriteProducts =
	productService.getFavoriteProductsByUser(applicationUser.g
	etUserId());
	relevantProducts.addAll(favoriteProducts);

	// get commented products
                        List<Product> commentedProducts =
	productService.getCommentedProductsByUser(applicationUser.
	getUserId());
	relevantProducts.addAll(commentedProducts);

	// get all search requests
	List<SearchRequest> searchRequests = searchRequestService

		.getAllSearchRequestByUserId(
		applicationUser.getUserId());
	for (SearchRequest searchRequest : searchRequests) {
		List<Product> searchedProducts =
		productService.getSearchedProductList(searchRequest)
		;
		relevantProducts.addAll(searchedProducts);
		searchedProductList.addAll(searchedProducts);
	}

	for (Product product : relevantProducts) {
		System.out.println("--------relevant product id-----
		---" + product.getProductId());
	}

	System.out.println("*******************************
	provide favorite
	notification**********************************");
	for (Product product : favoriteProducts) {
		//get no of comment after favorite added date and
		time
		//check existing notification
		//if not exist add notification as not seen
	}


	System.out.println("*******************************
	provide comment
	notification**********************************");
	for (Product product : commentedProducts) {
		//get no of comment after last comment added date
		and time
		//check existing notification
		//if not exist add notification as not seen
```

```
	}

System.out.println("*******************************provide short
distance notification*********************************");
	for (Product product : relevantProducts) {
		// calculate the distance
		double fromLat = applicationUser.getLatitude();
		double fromLon = applicationUser.getLongtitude();
		double toLat = product.getUser().getLatitude();
		double toLon = product.getUser().getLongtitude();
				double distance =
	this.getDistanceFromLatLonInKm(fromLat, fromLon, toLat,
	toLon);
		if (distance < 1) {
			//this product details should post the users
			notification panel high priority
			Notification notification = new
			Notification(0,
			EnumConstant.NotificationType.DISTANCE.toStri
			ng(), "your seller is near by", 5, null, new
			Date(), applicationUser, product);
			List<Notification> existNotifications =
			notificationService.getNotification(applicati
			onUser.getUserId(), product.getProductId(),
			EnumConstant.NotificationType.DISTANCE.toStri
			ng());
			if(existNotifications == null ||
			(existNotifications != null &&
			existNotifications.size() == 0)){

	notificationService.
	saveUpdateNotification(notification);
			}
		}
	}

	System.out.println("*******************************provid
	e search notification*********************************");
	for (Product product : searchedProductList) {
		//this product details should post the users
		notification panel low priority


				Notification notification = new
	Notification(0,
	EnumConstant.NotificationType.SEARCH.toString(), "we found
	out a new product", 4, null, new Date(), applicationUser,
	product);
		List<Notification> existNotifications =
	notificationService.getNotification(applicationUser.getUse
	rId(), product.getProductId(),
	EnumConstant.NotificationType.SEARCH.toString());
		if(existNotifications == null || (existNotifications
	!= null && existNotifications.size() == 0)){
```

```
                        notificationService.
                        saveUpdateNotification(notification);
                              }
                        }

                  }

                  System.out.println("****************Schedular finished running at
                  " + new Date() + "*********************");
            }

                  // using Haversine_formula to get the distance
                  // http://www.movable-type.co.uk/scripts/latlong.html ||
                  // https://en.wikipedia.org/wiki/Haversine_formula
                  public double getDistanceFromLatLonInKm(double fromLat, double
                  fromLon, double toLat, double toLon) {
                  try {
                        int R = 6371; // Radius of the earth in km
                        double dLat = deg2rad(toLat - fromLat); // deg2rad below
                        double dLon = deg2rad(toLon - fromLon);
                        double a = Math.sin(dLat / 2) * Math.sin(dLat / 2)
                                    + Math.cos(deg2rad(fromLat)) *
                        Math.cos(deg2rad(toLat)) * Math.sin(dLon / 2) *
                        Math.sin(dLon / 2);
                        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
                        double d = R * c; // Distance in km
                        return d;
                  } catch (Exception e) {
                        System.out.println("****error occured when calculating the
                        distance");
                        e.printStackTrace();
                        return -1;
                  }

            }

            public double deg2rad(double deg) {
                  return deg * (Math.PI / 180);
            }

}
```

9.5.1.31 Product Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.SystemUser;

public interface ProductDao extends CrudRepository<Product, Long>{

	public Product findByProductId(long productId);

	public List<Product> findAll();

	public List<Product> findProductByUser(SystemUser user);

	@Query(value = "select * from products where "
			+
			"productSubCategory_productSubCategoryId=:productSubCatego
			ryId and "
			+ "price between :minPrice and :maxPrice and "
			+ "authenticity=:authenticity or 1=:authenticityBoolVal
			and "
			+ "productType=:productType or 1=:productTypeBoolVal and "
			+ "productBrand=:productBrand or 1=:productBrandBoolVal
		and "
			+ "productCondition=:productCondition or
			1=:productConditionBoolVal"
			, nativeQuery=true)
	public List<Product> getOtherProductList(
			@Param("productSubCategoryId") long productCategoryId,
			@Param("minPrice") double minPrice,
			@Param("maxPrice") double maxPrice,
			@Param("authenticity") String authenticity,
			@Param("authenticityBoolVal") int authenticityBoolVal,
			@Param("productType") String productType,
			@Param("productTypeBoolVal") int productTypeBoolVal,
			@Param("productBrand") String productBrand,
			@Param("productBrandBoolVal") int productBrandBoolVal,
			@Param("productCondition") String productCondition,
			@Param("productConditionBoolVal") int
			productConditionBoolVal);

			@Query(value = "select * from products where "
			+
			"productSubCategory_productSubCategoryId=:productSubCatego
			ryId and "
			+ "price between :minPrice and :maxPrice and "
			+ "productType=:productType or 1=:productTypeBoolVal and "
```

```
            + "productCondition=:productCondition or
            1=:productConditionBoolVal and "
            + "millege between :minMillege and :maxMillege and "
            + "transmission = :transmission or 1 =
            :transmissionBoolVal and "
            + "fuelType = :fuelType or 1 = :fuelTypeBoolVal and "
            + "bodyType = :bodyType or 1 = :bodyTypeBoolVal"
            , nativeQuery=true)
    public List<Product> getVehicleProductList(
            @Param("productSubCategoryId") long productCategoryId,
            @Param("minPrice") double minPrice,
            @Param("maxPrice") double maxPrice,
            @Param("productType") String productType,
            @Param("productTypeBoolVal") int productTypeBoolVal,
            @Param("productCondition") String productCondition,
            @Param("productConditionBoolVal") int
            productConditionBoolVal,
            @Param("minMillege") int minMillege,
            @Param("maxMillege") int maxMillege,
            @Param("transmission") String transmission,
            @Param("transmissionBoolVal") int transmissionBoolVal,
            @Param("fuelType") String fuelType,
            @Param("fuelTypeBoolVal") int fuelTypeBoolVal,
            @Param("bodyType") String bodyType,
            @Param("bodyTypeBoolVal") int bodyTypeBoolVal
            );


    @Query(value = "select * from products where "
            +
            "productSubCategory_productSubCategoryId=:productSubCatego
            ryId and "
            + "price between :minPrice and :maxPrice and "
            + "productType=:productType or 1=:productTypeBoolVal and "
            + "productCondition=:productCondition or
            1=:productConditionBoolVal and "
            + "noOfBathroom between :minNoOfBathroom and
            :maxNoOfBathroom and "
            + "noOfRoom between :minNoOfRoom and :maxNoOfRoom"
            , nativeQuery=true)
    public List<Product> getPropertyProductList(
            @Param("productSubCategoryId") long productCategoryId,
            @Param("minPrice") double minPrice,
            @Param("maxPrice") double maxPrice,
            @Param("productType") String productType,
            @Param("productTypeBoolVal") int productTypeBoolVal,
            @Param("productCondition") String productCondition,
            @Param("productConditionBoolVal") int
            productConditionBoolVal,
            @Param("minNoOfBathroom") int minNoOfBathroom,
            @Param("maxNoOfBathroom") int maxNoOfBathroom,
            @Param("minNoOfRoom") int minNoOfRoom,
            @Param("maxNoOfRoom") int maxNoOfRoom
            );
```

```
}
```

9.5.1.32 Product Category Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.isd.sappu.savari.domains.ProductCategory;

public interface ProductCategoryDao extends CrudRepository<ProductCategory,
Long>{

	public List<ProductCategory> findAll();

}
```

9.5.1.33 Product Sub Category Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.isd.sappu.savari.domains.ProductCategory;
import com.isd.sappu.savari.domains.ProductSubCategory;

public interface ProductSubCategoryDao extends
CrudRepository<ProductSubCategory, Long>{

	public ProductSubCategory findByProductSubCategoryId(long
	productSubCategoryId);

	public List<ProductSubCategory> findAll();

	public List<ProductSubCategory> findByProductCategory(ProductCategory
	productCategory);

}
```

9.5.1.34 Product Multimedia Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
```

```
import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.ProductMultimedia;

public interface ProductMultimediaDao extends
CrudRepository<ProductMultimedia, Long>{

	public List<ProductMultimedia> findByProduct(Product product);

}
```

9.5.1.35 Search Request Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.isd.sappu.savari.domains.SearchRequest;
import com.isd.sappu.savari.domains.SystemUser;

public interface SearchRequestDao extends CrudRepository<SearchRequest, Long>{

	public SearchRequest findBySearchRequestId(long searchRequestId);

	public List<SearchRequest> findByUser(SystemUser user);

}
```

9.5.1.36 Rating Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.Rating;
import com.isd.sappu.savari.domains.SystemUser;

public interface RatingDao extends CrudRepository<Rating, Long>{

	public Rating findByRatingId(long ratingId);

	public Rating findByUserAndProduct(SystemUser user, Product product);

	public List<Rating> findByUser(SystemUser user);

	public List<Rating> findByProduct(Product product);

}
```

9.5.1.37 Comment Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.isd.sappu.savari.domains.Comment;
import com.isd.sappu.savari.domains.Product;
import com.isd.sappu.savari.domains.SystemUser;

public interface CommentDao extends CrudRepository<Comment, Long>{

	public Comment findByCommentId(long commentId);

	public List<Comment> findCommentByProduct(Product product);

	public List<Comment> findCommentByUser(SystemUser systemUser);

}
```

9.5.1.38 Message Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.isd.sappu.savari.domains.Message;
import com.isd.sappu.savari.domains.SystemUser;

public interface MessageDao extends CrudRepository<Message, Long>{

	public Message findByMessageId(long messageId);

	public List<Message> findBySender(SystemUser sender);

	public List<Message> findByReceiver(SystemUser receiver);

}
```

9.5.1.39 User Data Access Service

```
package com.isd.sappu.savari.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.isd.sappu.savari.domains.SystemUser;
```

```
public interface SystemUserDao extends CrudRepository<SystemUser, Long>{

	public SystemUser findByUsername(String username);

	public SystemUser findByUserId(long userId);

	public List<SystemUser> findAll();

}
```

9.5.1.40 Login Success Service

```
package com.isd.sappu.savari.services;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.authentication.AuthenticationSuccessHandler;
import
org.springframework.security.web.authentication.SavedRequestAwareAuthenticatio
nSuccessHandler;

import com.isd.sappu.savari.domains.SystemUser;
import com.isd.sappu.savari.util.AppConstant;
import com.isd.sappu.savari.util.SessionUtil;

public class LoginSuccessHandler implements AuthenticationSuccessHandler {

	@Autowired
	SessionUtil sessionUtil;

	@Autowired
	SystemUserService systemUserService;

	private AuthenticationSuccessHandler target = new
	SavedRequestAwareAuthenticationSuccessHandler();

	@Override
	public void onAuthenticationSuccess(HttpServletRequest request,
			HttpServletResponse response, Authentication
			authentication) throws IOException,
			ServletException {
		System.out.println("CAME HERE onAuthenticationSuccess");

		if(authentication != null){
			Object principal = authentication.getPrincipal();
```

```
                UserDetails userDetails = (UserDetails) (principal
                instanceof UserDetails ? principal : null);

                SystemUser user =
                systemUserService.getSystemUserByUsername(userDetails.getU
                sername());
                sessionUtil.addLoggedUserToSession(AppConstant.LOGGED_USER
        , user, request);

                if(userDetails != null){
                      try {
                            target.onAuthenticationSuccess(request,
                            response, authentication);
                      } catch (Exception e) {
                            e.printStackTrace();
                      }
                }
        }

      }
```

9.5.1.41 Login Failure Service

```
package com.isd.sappu.savari.services;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.AuthenticationException;
import
org.springframework.security.web.authentication.AuthenticationFailureHandler;
import org.springframework.stereotype.Component;

@Component(value="loginFailureHandler")
public class LoginFailureHandler implements AuthenticationFailureHandler {

      @Override
      public void onAuthenticationFailure(HttpServletRequest request,
                  HttpServletResponse response, AuthenticationException
                  arg2)
                  throws IOException, ServletException {

            response.sendRedirect("login/login.htm?login_error=t");

      }

}
```

9.5.2 Front End HTML/JAVASCRIPT Code

9.5.2.1 Home Page

```
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAm5pznwDLoa65sMEc5uIonQ
O61ErAStKg"></script>

<script type="text/javascript">

$("document").ready(function(){

//     window.setInterval(function(){
//     }, 5000);
var firstName = "";

       $.get("../user/getSystemUser.do",{
              userId:${sessionScope.LOGGED_USER.userId}
           },function(data, status){
              var fromLatitude = data.latitude;
              var fromLongtitude = data.longtitude;
              console.log(fromLatitude);
              console.log(fromLongtitude);
              firstName = data.firstName;
              initializer(fromLatitude, fromLongtitude);
           });

       function initializer(latitude, longtitude){
              var myCenter=new google.maps.LatLng(latitude, longtitude);

              var map = new google.maps.Map(document.getElementById('map-
              canvas'), {
                  zoom: 16,
                  center: myCenter
              });

              var marker=new google.maps.Marker({
                       position:myCenter,
                       animation:google.maps.Animation.BOUNCE,
                       url:"/hhfhf.htm",
                       icon: '../images/man_icon.png'
                });
              marker.setMap(map);
              var infowindow = new google.maps.InfoWindow({
                     content:"Hi "+firstName+", You are here"

                });
              infowindow.open(map,marker);
```

```
                <c:forEach var="followSeller" items="${followSellers}"
                varStatus="count">
                        var myCenter_${count.index}=new
                        google.maps.LatLng('${followSeller.sellerLatitude}',
                        '${followSeller.sellerLongtitude}');
                        var marker_${count.index}=new google.maps.Marker({
                                position:myCenter_${count.index},
                                animation:google.maps.Animation.DROP
                          });
                        marker_${count.index}.addListener('click', function() {

                        window.location="../user/findSeller.htm?sUserId=${followSe
                        ller.sellerId}";
                          });
                        marker_${count.index}.setMap(map);
                        var infowindow_${count.index} = new
                        google.maps.InfoWindow({

                        content:"${followSeller.sellerName}
                        (${followSeller.productTitle})"
                          });
                        infowindow_${count.index}.open(map,marker_${count.index});
                </c:forEach>
        }

});

</script>

<style>

#map-canvas{
        border-width:1px;
        border-style:solid;
        width: 100%;
        height: 500px;
}

</style>

<div class="main-content">
  <div class="swipe-area"></div>
  <a href="#" data-toggle=".container" id="sidebar-toggle"> <span
class="bar"></span> <span class="bar"></span> <span class="bar"></span> </a>
  <div class="content">
    <h1>We Found You Here...</h1>
                <div class="container-fluid">
                        <div id="map-canvas"></div>
                </div>
  </div>
  <a href="../user/findSeller.htm?sUserId=2"><button>xx</button></a>
</div>
```

9.5.2.2 Follow Seller

```
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAm5pznwDLoa65sMEc5uIonQ
O61ErAStKg"></script>

<script type="text/javascript">

$("document").ready(function(){

	var fromlat = ${buyer.latitude};
	var fromlong = ${buyer.longtitude};

	window.setInterval(function(){
		$.get("../user/getSystemUser.do",{
			userId:${sessionScope.LOGGED_USER.userId}
		    },function(data, status){
			fromlat = data.latitude;
			fromlong = data.longtitude;
			console.log(fromlat);
			console.log(fromlong);
		    });
	}, 1000);

	var Center = new google.maps.LatLng(${buyer.latitude},
	${buyer.longtitude});
	var directionsDisplay;
	var directionsService = new google.maps.DirectionsService();
	var map;

	function initialize() {
	  directionsDisplay = new google.maps.DirectionsRenderer();
	  var properties = {
	    center: Center,
	    zoom: 20,
	    mapTypeId: google.maps.MapTypeId.MAP
	  };

	  map = new google.maps.Map(document.getElementById("map"), properties);
	  directionsDisplay.setMap(map);

	  var marker = new google.maps.Marker({
	    position: Center,
	    animation: google.maps.Animation.BOUNCE,
	    icon: '../images/man_icon.png'
	  });

	  window.setInterval(function(){
			changeMarkerPosition(marker);
		}, 1000);
```

```
        marker.setMap(map);
        Route();
      }

      function changeMarkerPosition(marker) {
          var latlng = new google.maps.LatLng(fromlat, fromlong);
          marker.setPosition(latlng);
      }

      function Route() {

        var start = new google.maps.LatLng(${buyer.latitude},
      ${buyer.longtitude});
        var end = new google.maps.LatLng(${seller.latitude},
      ${seller.longtitude});
        var request = {
          origin: start,
          destination: end,
          travelMode: google.maps.TravelMode.DRIVING
        };
        directionsService.route(request, function(result, status) {
          if (status == google.maps.DirectionsStatus.OK) {
            directionsDisplay.setDirections(result);
          } else {
            alert("couldn't get directions:" + status);
          }
        });
      }

      google.maps.event.addDomListener(window, 'load', initialize);

});

</script>

<style>

#map{
      border-width:1px;
      border-style:solid;
      width: 100%;
      height: 500px;
}

</style>

<div class="main-content">
  <div class="swipe-area"></div>
  <a href="#" data-toggle=".container" id="sidebar-toggle"> <span
class="bar"></span> <span class="bar"></span> <span class="bar"></span> </a>
  <div class="content">
    <h1>Follow Seller</h1>
            <div class="container-fluid">
                  <div id="map"></div>
```

```
            </div>
  </div>
</div>
```

9.5.2.3 User Location Identification

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles"%>
<%@taglib uri="http://www.springframework.org/tags" prefix="fmt"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="security"
uri="http://www.springframework.org/security/tags" %>

<div class="col-md-3 left_col">
      <div class="left_col scroll-view">
            <div class="navbar nav_title" style="border: 0;">
                  <a href="../home/home.htm" class="site_title"><i class="fa
fa-area-chart"></i> <span>Savari</span></a>
            </div>
            <div class="clearfix"></div>

            <br />

            <!-- sidebar menu -->
            <div id="sidebar-menu" class="main_menu_side hidden-print
main_menu">

                  <div class="menu_section">
                        <h2>Sappu Savari</h2><br> <br>

                        <ul class="nav side-menu">
                              <li><a href="../home/home.htm"><i class="fa
fa-home"></i> Home</a></li>
                              <li><a href="#"><i class="fa fa-usd"></i>
Products <span class="fa fa-chevron-down"></span></a>
                                    <ul class="nav child_menu"
style="display: none">
                                          <li><a
href="../buy/listProductCategory.htm">Buy</a></li>
                                          <li><a
href="../sell/listProductCategory.htm">Sell</a></li>
                                          <li><a
href="../sell/listProduct.htm">My Store</a></li>
                                    </ul>
                              </li>
                              <li><a href="#"><i class="fa fa-edit"></i>
Messages <span class="fa fa-chevron-down"></span></a>
                                    <ul class="nav child_menu"
style="display: none">
                                          <li><a
href="../message/messageDashBoard.htm">Dash board</a></li>
                                          <li><a
href="../message/messageInbox.htm">Inbox</a></li>
```

```
                                                <li><a
href="../message/messageSent.htm">Sent</a></li>
                                            </ul>
                                        </li>
                                    </ul>
                                </div>
                            </div>
                            <!-- /sidebar menu -->
                    </div>
            </div>

            <!-- top navigation -->
            <div class="top_nav">

                    <div class="nav_menu">
                            <nav class="" role="navigation">
                                    <div class="nav toggle">
                                            <a id="menu_toggle"><i class="fa fa-bars"></i></a>
                                    </div>

                                    <ul class="nav navbar-nav navbar-right">
                                            <li class="">
                                                    <a href="javascript:;"
                                                    class="user-profile dropdown-toggle" data-
toggle="dropdown"
                                                    aria-expanded="false"> <img src="<c:url
value="/images/img.jpg"/>" alt="">
                                                    <span class=" fa fa-angle-down"></span>
                                                    </a>
                                                    <ul    class="dropdown-menu dropdown-usermenu
animated fadeInDown pull-right">
                                                    <li><a
href="../user/userprofile.htm?username=${sessionScope.LOGGED_USER.username}">
Profile</a></li>
                                                    <li><a
href="../user/viewSearchHistory.htm?userId=${sessionScope.LOGGED_USER.userId}"
> Search History</a></li>
                                                    <li><a
href="../notification/notificatoinDashBoard.htm?userId=${sessionScope.LOGGED_U
SER.userId}">Notification</a></li>
                                                    <li><a href="../login/logout.htm"><i
                    class="fa fa-sign-out pull-right"></i> Log Out</a></li>
                                                    </ul>
                                            </li>
                                            <li class="">
                                                    <a href="javascript:;"
                                                    class="user-profile dropdown-toggle" data-
toggle="dropdown"
                                                    aria-expanded="false"> <img src="<c:url
value="/images/bell.png"/>" alt="">
                                                    <span class=" fa fa-angle-down"></span>
                                                    </a>
                                                    <ul    class="dropdown-menu dropdown-usermenu
animated fadeInDown pull-right" id="notificationUl">
```

```
                                    </ul>
                                </li>
                                <li class="">
                            <a
href="../user/userprofile.htm?username=${sessionScope.LOGGED_USER.username}"><
span class="badge bg-green"> Hi,
${sessionScope.LOGGED_USER.username}</span></a>
                         </li>
                            </ul>
                    </nav>
            </div>

</div>
<!-- /top navigation -->

<script type="text/javascript">

$("document").ready(function(){
        /*function success(pos) {
                var crd = pos.coords;

                    $.post("../user/updateUserLocation.do",
                    {
                            userId:"${sessionScope.LOGGED_USER.userId}",
                        latt: crd.latitude,
                        longt: crd.longitude,
                        acc:crd.accuracy
                    },
                    function(data, status){
                        console.log("location saved...");
                    });

                };

                function error(err) {
                  console.warn('ERROR(' + err.code + '): ' + err.message);
                };

                var options = {
                  enableHighAccuracy: true,
                  timeout: 5000,
                  maximumAge: 0
                };

                navigator.geolocation.getCurrentPosition(success, error,
                options);*/

        function notificationProcess(notificationId, redirectLocation){
                console.log(notificationId);
                console.log(redirectLocation);
        //            $.post("../notification/updateNotificationSeenStatus.do",
        //        {notificationId:notificationId},
        //        function(data, status){
        //              window.location = redirectLocation;
        //        });
```

```
        }

        $.post("../notification/getNotifications.do",
    {userId:"${sessionScope.LOGGED_USER.userId}"},
    function(data, status){
        console.log(data);
        $("#notificationUl").html("");
        var htmlString = "";
        $.each(data, function(i, item) {
             var redirectUrl =
             '../product/showProduct.htm?productId='+item.product.productId+'&
             notify='+item.notificationId;
            if(item.seenStatus == 0){
             htmlString = htmlString + "<li><a style='font-weight:bold;'
             href='"+redirectUrl+"'>"+item.description+"("+item.user.firstName
             +"/"+item.product.productTitle+")</a></li>";
            }
        });
        $("#notificationUl").html(htmlString);
    });



});

</script>
```

9.6 Appendix F – Evaluation and Testing

Sappu Savari – Location Based Advertising and Marketing User Evaluation Criteria for the System

General Instructions:

You may circle the appropriate number appear in cages to represent your choice.

1	2	3	4	5
Outstanding	Good	Satisfactory	Poor	Unsatisfactory

Skip question/s if not applicable.

User Information

a. Select your role

1. General user with IT knowledge	2.QA	3.IT Expert	4.Other

b. What is your age group?

1. 18-23 years	2. 24-29 years	3. 30-35 years	4. 35-55 years	5. More than 55

c. Gender

1. Male	2. Female

The information provided in the application

j. Content is free from spelling and grammatical errors.

1	2	3	4	5

k. The information provided on the site is clear, concise and informative to the intended user.

1	2	3	4	5

l. Information given in the site is helpful for a learner.

1	2	3	4	5

m. The language is non-discriminatory. Content is free from race, ethnic, gender, age and other stereotype.

1	2	3	4	5

n. By using the information given in the site, consumers are able to solve most of their problems.

1	2	3	4	5

o. All the needed information is available in the system, as an example the mobile phone brand list.

1	2	3	4	5

p. Information given in non-technical, understandable language without technical jargons.

1	2	3	4	5

q. The information is well structured in the application, suitable for marketing purposes.

1	2	3	4	5

r. Information given in the system not confusing the operator

1	2	3	4	5

User satisfaction on interfaces

j. Reading of letters of application

1	2	3	4	5

k. Organization of information.

1	2	3	4	5

l. Use of terminology throughout the application.

1	2	3	4	5

m. Proper navigation.

1	2	3	4	5

n. Design of a page provides relevant information only.

1	2	3	4	5

o. Audit Information is visible in the application.

1	2	3	4	5

p. Position of messages on the screen is easy to view.

1	2	3	4	5

q. Color choices visually accessible and pleasant to see.

1	2	3	4	5

r. The site achieves its purpose.

1	2	3	4	5

Usability

j. The layout and the design of the application encourage frequent use of the application.

1	2	3	4	5

k. Easy navigation.

1	2	3	4	5

l. The organization is clear, logical and effective, making it easy and simple for the intended audience to understand.

1	2	3	4	5

m. Individuals can easily start and exit the program.

1	2	3	4	5

n. The individual has the choice of going directly to desire information or using a structure search to identify relevant topics.

1	2	3	4	5

o. Feel very confident when using the application.

1	2	3	4	5

p. Not necessary to get to know about the site before it could effectively use it.

1	2	3	4	5

s. Use the language of the application is clear to the intended audience.

1	2	3	4	5

t. Application can be used without written instructions.

1	2	3	4	5

Ease of Learning

g. Can learn to navigate the application quickly.

1	2	3	4	5

h. Can easily remember how to use the application next time.

1	2	3	4	5

i. It is easy to remember main and top many item names and use of commands.

1	2	3	4	5

j. Tasks can be performed in a straightforward manner.

1	2	3	4	5

k. Help messages on the screen.

1	2	3	4	5

l. Links to supplementary reference materials.

1	2	3	4	5

Overall Impression

j. User friendliness.

1	2	3	4	5

k. Make life easy with its facilities.

1	2	3	4	5

l. Usefulness.

1	2	3	4	5

m. Ease of use.

1	2	3	4	5

n. Efficiency

1	2	3	4	5

o. The system is easy to learn and the information provided is meaningful and helpful.

1	2	3	4	5

p. The application is a one-stop advertising system.

1	2	3	4	5

q. Time taken for search products is acceptable.

1	2	3	4	5

r. Time taken to find sellers are acceptable.

1	2	3	4	5

Evaluators Observation

- **Information provided in the application**

Evaluator	Age	Gender	a	b	c	d	e	f	g	h	i
General user with IT knowledge	30-35 year	male	2	2	1	1	2	1	1	2	2
General user with IT knowledge	24-29 year	male	2	3	2	2	2	1	2	2	3
General user with IT knowledge	30-35 year	male	1	1	2	1	2	2	1	1	1
General user with IT knowledge	24-29 year	female	1	2	2	1	2	2	1	1	1
General user with IT knowledge	24-29 year	female	1	2	3	1	3	3	2	3	2
General user with IT knowledge	24-29 year	female	1	1	2	1	2	2	1	2	2
General user with IT knowledge	30-35 year	female	1	3	3	1	2	3	2	2	3
General user with IT knowledge	30-35 year	male	1	2	2	1	1	2	1	1	1
General user with IT knowledge	24-29 year	male	1	1	2	1	3	2	1	2	1
General user with IT knowledge	24-29 year	male	1	2	2	1	2	2	1	1	1
QA	24-29 year	female	1	1	1	1	1	1	1	2	1
QA	30-35 year	female	2	1	2	1	1	2	1	2	2
IT Expert	24-29 year	female	1	1	2	1	1	1	1	1	2
IT Expert	24-29 year	male	1	1	3	2	2	1	2	1	2
IT Expert	30-35 year	female	1	1	2	1	1	1	1	1	2
IT Expert	24-29 year	female	1	2	2	1	1	1	2	1	1
IT Expert	24-29 year	female	2	2	1	1	1	2	1	2	2
Total			21	28	34	19	29	29	22	27	29
Average			1.2	1.6	2	1.1	1.7	1.7	1.3	1.6	1.7

Figure 43: Observation Summary – Information provided in the application

- **User satisfaction on Interfaces**

Evaluator	Age	Gender	a	b	c	d	e	f	g	h	i
General user with IT knowledge	30-35 year	male	2	1	2	1	1	2	3	2	1
General user with IT knowledge	24-29 year	male	2	1	2	3	1	2	1	1	2
General user with IT knowledge	30-35 year	male	2	2	2	1	1	2	1	1	1
General user with IT knowledge	24-29 year	female	1	3	2	1	1	1	2	1	2
General user with IT knowledge	24-29 year	female	1	2	2	1	2	2	2	1	2
General user with IT knowledge	24-29 year	female	1	2	2	1	1	2	2	1	1
General user with IT knowledge	30-35 year	female	1	3	2	2	2	2	2	1	2
General user with IT knowledge	30-35 year	male	1	2	2	1	1	1	1	1	1
General user with IT knowledge	24-29 year	male	1	2	1	1	1	1	1	1	1
General user with IT knowledge	24-29 year	male	1	1	2	1	2	2	2	1	1
QA	24-29 year	female	1	2	2	1	1	1	1	2	2
QA	30-35 year	female	1	1	1	2	1	2	2	1	1
IT Expert	24-29 year	female	1	1	2	1	1	2	2	1	1
IT Expert	24-29 year	male	1	1	2	1	2	1	2	1	1
IT Expert	30-35 year	female	1	2	3	2	1	2	1	2	2
IT Expert	24-29 year	female	1	2	1	2	1	1	1	2	1
IT Expert	24-29 year	female	1	1	2	1	1	2	2	3	1
Total			20	29	32	23	21	28	28	23	23
Average			1.2	1.7	1.9	1.4	1.2	1.6	1.6	1.4	1.4

Figure 44: Observation Summary – User satisfaction on interfaces

- **Usability**

Evaluator	Age	Gender	a	b	c	d	e	f	g	h	i
General user with IT knowledge	30-35 year	male	2	3	2	1	2	2	2	2	1
General user with IT knowledge	24-29 year	male	2	2	2	1	2	2	3	2	2
General user with IT knowledge	30-35 year	male	2	1	2	1	2	1	2	1	1
General user with IT knowledge	24-29 year	female	1	1	2	1	3	2	2	2	2
General user with IT knowledge	24-29 year	female	2	1	3	2	2	2	3	2	2
General user with IT knowledge	24-29 year	female	1	1	2	1	2	2	2	1	2
General user with IT knowledge	30-35 year	female	2	2	3	3	2	3	2	2	3
General user with IT knowledge	30-35 year	male	1	1	2	1	2	1	2	1	1
General user with IT knowledge	24-29 year	male	1	1	2	1	1	1	2	1	1
General user with IT knowledge	24-29 year	male	1	1	2	1	1	1	2	1	1
QA	24-29 year	female	1	1	2	1	2	1	1	1	2
QA	30-35 year	female	1	1	2	1	2	1	1	1	2
IT Expert	24-29 year	female	1	1	2	1	1	1	1	1	1
IT Expert	24-29 year	male	1	2	2	1	1	2	1	2	2
IT Expert	30-35 year	female	1	1	2	1	1	2	2	2	1
IT Expert	24-29 year	female	1	3	1	2	3	2	2	1	2
IT Expert	24-29 year	female	2	1	2	2	1	1	1	1	2
Total			23	24	35	22	30	27	31	24	28
Average			1.4	1.4	2.1	1.3	1.8	1.6	1.8	1.4	1.6

Figure 45: Observation Summary – Usability

- **Ease of Learning**

Evaluator	Age	Gender	a	b	c	d	e	f
General user with IT knowledge	30-35 year	male	2	1	2	2	2	5
General user with IT knowledge	24-29 year	male	2	2	2	2	2	4
General user with IT knowledge	30-35 year	male	1	1	1	2	1	2
General user with IT knowledge	24-29 year	female	1	1	2	1	3	5
General user with IT knowledge	24-29 year	female	2	1	3	2	2	2
General user with IT knowledge	24-29 year	female	1	1	1	2	1	2
General user with IT knowledge	30-35 year	female	2	2	2	1	2	3
General user with IT knowledge	30-35 year	male	1	1	2	1	1	4
General user with IT knowledge	24-29 year	male	2	1	1	1	1	2
General user with IT knowledge	24-29 year	male	1	1	2	1	2	3
QA	24-29 year	female	2	1	2	1	3	5
QA	30-35 year	female	1	2	1	1	1	5
IT Expert	24-29 year	female	1	1	1	1	2	3
IT Expert	24-29 year	male	1	1	2	1	1	2
IT Expert	30-35 year	female	2	2	1	3	2	2
IT Expert	24-29 year	female	3	2	2	1	1	2
IT Expert	24-29 year	female	1	2	2	1	3	5
Total			26	23	29	24	30	56
Average			1.5	1.4	1.7	1.4	1.8	3.3

Figure 46: Observation Summary – Ease of learning

- **Overall Impression**

Evaluator	Age	Gender	a	b	c	d	e	f	g	h	i
General user with IT knowledge	30-35 year	male	1	2	2	2	2	2	1	2	1
General user with IT knowledge	24-29 year	male	2	2	2	2	2	2	2	2	3
General user with IT knowledge	30-35 year	male	1	1	2	2	2	1	1	2	1
General user with IT knowledge	24-29 year	female	2	2	2	2	2	1	2	3	2
General user with IT knowledge	24-29 year	female	2	2	3	2	2	2	2	3	3
General user with IT knowledge	24-29 year	female	1	1	2	1	2	2	3	2	2
General user with IT knowledge	30-35 year	female	2	2	1	2	2	1	2	2	2
General user with IT knowledge	30-35 year	male	1	2	1	1	1	1	2	1	1
General user with IT knowledge	24-29 year	male	1	1	1	2	2	1	1	2	2
General user with IT knowledge	24-29 year	male	1	1	1	2	1	1	2	1	1
QA	24-29 year	female	3	2	1	1	3	3	2	3	2
QA	30-35 year	female	2	1	1	1	3	1	2	2	1
IT Expert	24-29 year	female	1	1	1	1	1	2	1	1	1
IT Expert	24-29 year	male	1	1	1	1	2	1	1	2	1
IT Expert	30-35 year	female	1	1	1	1	3	1	2	2	2
IT Expert	24-29 year	female	1	2	1	2	1	2	2	1	1
IT Expert	24-29 year	female	1	2	1	1	2	1	2	1	3
Total			24	26	24	26	33	25	30	32	29
Average			1.4	1.5	1.4	1.5	1.9	1.5	1.8	1.9	1.7

Figure 47: Observation Summary – Overall Impression