Sri Lankan Travel Mate

ELITE

134073J	BR Jayawickrama
134093U	HMTK Madhusanka
134061V	HKEP Hettiarachchi
134185E	NR Weeraddana
134217G	IU Kothalawala

Faculty of Information Technology

University of Moratuwa

June 2016

Sri Lankan Travel Mate

ELITE

134073J	BR Jayawickrama
134093U	HMTK Madhusanka
134061V	HKEP Hettiarachchi
134185E	NR Weeraddana
134217G	IU Kothalawala

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Honours Degree of Bachelor of Science in Information Technology.

June 2016

Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of the Student	Signature of the students.
B. R. Jayawickrama	
H. M. T. K Madhusanka	
H. K. E. P. Hettiarachchi	
N. R. Weeraddana	
I. U. Kothalawala	

_____

Date

Supervised by

Mr Chaman Wijesiriwardana _____

Signature

_____

Date

Acknowledgement

We would like to express our deepest gratitude to everyone who supported us to complete this report. We sincerely thank Mr Chaman Wijesiriwardana for his supervision in regard of our project. We would also like to thank the members of the Keen Mind Mobile Solutions for their guidance and information to carry out this project. Our gratitude also goes to our friends and batch mates who willingly helped us throughout the project.

Abstract

Even though tourism grows as an industry in Sri Lanka, technology related to tourism needs more development. The Sri Lankan government's attempt to gather more registered travel guides helps to make a foreign traveler's journey in Sri Lanka more reliable and safe. Due to the lack of having methods to instantly contact registered travel guides, the process of finding a reliable travel guide may take more time and effort. The Sri Lankan TravelMate or the SLTravelMate is a solution to address the problem of directly contacting registered travel guides in Sri Lanka. A web application, an admin portal and an Android mobile application allows different types of users to access the SLTravelMate's hosted database. All these components are then connected to a web service. The web service gives access to the database and transfers data collected from user inputs. The model-view-controller software architectural pattern was used to analyze and design the system into the above described components. The laravel web development framework, which uses the same architectural pattern was used to implement the web service, admin portal and the web application. Android development tools were used to implement the mobile application. A relational database made of structured query language is used here to store the relevant data as records. The components of SLTravelmate were evaluated with both user centric and system centric methods. The user centric method was beneficial for a system lika the SLTravelMate where many users are accessing and interacting.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 - Introduction

Since Sri Lanka being a popular tourist destination entertains a large number of tourists each year, Tourism is a popular industry in Sri Lanka. Many investors and entrepreneurs contribute for the balance of payment of Sri Lanka through Tourism industry. Some of the tourists who come to Sri Lanka have to face so many difficulties in finding the most appropriate shopping center that goes with their budget, nearest restaurant, leisure activities, accommodations, etc. however the most important necessity is finding a guide from a reliable source. Tourists prefer to have a travel guide person who can interactively help them while traveling. The Government of Sri Lanka together with the Sri Lanka Tourism Development Authority (SLTDA) [1] have initiated courses so that any interested native can sign up and become a registered travel guide in Sri Lanka [2].Even though travel guides can get registered by the government, there is no service that provides a way to instantly contact such a registered guide. If a tourist needs to find a travel guide, he has to contact an agency [3] to get such a qualified person prior to their journey or before their departure from home country. If the tourist did not arrange a meeting with a guide, contacting one in the last minute would be both effort and time consuming. With all the above discussed requirements, it can be concluded, there exists the necessity of a better mechanism to attract tourists to our country and guiding them throughout their journey providing them a catalogue to use as tool to help them navigation easily.

1.2 – Addressing technological barriers in the tourism industry

Potential barriers that limit the creation and enhancement of tourist experiences includes the restrictions in telecommunication bandwidth, Internet accessibility, hardware and software functionality, equipment usage and connection costs as well as privacy, security and legal concerns. Since the modern day technology had addressed all of the above mentioned barriers, a new system could built on the existing infrastructure to address the problems faced by the tourists coming to Sri Lanka.

The new system which appears as an integrated solution should skip the long, tedious process of searching for tourism advice/information. On the other hand allowing the users to review and rate the places they visited, so that the tourists can explore the views of the previously tourists who visited those places.

1.3 - Aim & objectives of the project.

Aim

Implementing and developing a travel guide system for foreign travelers in with an online chat facility to contact travel guides.

Objectives

- Designing and implementing a web application which is specifically designed for travelers so that travelers can comment, rate, and review for the places added in the web application.
- Designing and implementing a mobile app to enable the users to access the system
- Creating an online chat system for travelers to contact travel guides
- Creating an administration website (admin portal) to approve all actions from travelers, travel guides and sponsors
- Creating a database of to store all the data in the system
- Integrating a mapping service to locate the user and show the nearby service providers useful to the user.
- Creating a web service to integrate all modules of the system i.e., Web application, mobile application, admin portal, database, Google Cloud Service (GCM) and Payment gateway.

1.4 - Proposed solution

Sri Lankan Travel Mate, or the SLTravelMate is the solution suggested for the problems discussed in above sections of this chapter. It mainly aims to reduce the inconveniences due to lack of data and guidance. The solution, classifies its users to several categories and cater the relevant content for each category. This solution's design includes several modules that stores and processes data according to the user's needs.

There are two types of users

Internal users

- Super admin – Super admin can create more admin users to the system and also can create new categories for the apps.
- Admin – Admin has the responsibility to approve all business user requests. Comments are reviewed by the admin before they are added to the system.(ex: when a photo is uploaded to the system indicating a specific place, admin has to check whether that information is true or false). Another responsibility is sending push notifications for the registered users according to their interests which are identified by mostly viewed pages and also user requests.
- Business User: They can request to add new details to the defined categories. They can also be sponsors which will enable them to maintain their own webpage and publish advertisements. They are given labels as gold, silver and platinum according to their level of sponsorship. Guides are another category of the service providers. Only government certified guides can register in the system. They have the ability to indicate their availability at a particular time, so that a traveler who wishes to seek his service can directly contact him by sending messages.

External users

There are three types of external users, they are

- Guest users: They can view the details
- Travelers: they are the users who have registered in the system to get the special services offered by the system other than services which are generally offered by the system to all viewers. Ex: Other than the app updates, special push notifications related to specific apps are sent only to the registered travelers.

System modules

Following are the major modules in the SLTravelMate system.

- Web services: Feed information from the backend to apps on request. A backend stores all the information.
- Web application: This is the website of our system where guest users and registered travelers i.e. external users are interacting with the system.

- Mobile application: This is the mobile application of the system. All the features included in the web application are also included here with additional few features that are discussed to be in the upcoming sections of the report.
- Admin portal: this is where admins and the business users i.e. internal users are authenticating into the system.

Both web app and mobile app provides the facility to the users to share the information that they like by using social media like Facebook, Twitter, Instagram and Google+.

There are several technologies used in this project. For android application, Java, JavaScript, and Android developer kit were used. PHP and HTML 5 are the languages that were used for the web application. All the information is stored in a MySQL database. For data retrieving, inserting, deleting and updating, Structured Query Language (SQL) is used. The Global Positioning System (GPS) is used to identify the locations of the users. For the Admin Portal PHP, HTML 5 languages, and boostrap, laravel frameworks were used for implementation. All the communications within all modules are done through web service by exchanging JSON files.

This dissertation is organized to give a concluding analysis of the process of designing implementing and developing the SLTravelMate. The upcoming chapter reviews existing travel guide systems to find the areas that are left unaddressed. Adapted technologies within the SLTravelMate are described in the "Technology Adapted" chapter. The "Analysis and design" chapter will give a proper way to understand how each component or module was designed and what these components do. How the adapted technologies were used to implement each component, is given in the "Implementation" chapter. The final results are evaluated with proper methods in the "Evaluation" chapter. The "Conclusion" chapter describes the major findings and work that needs further attention to develop the SLTravelMate. The appendixes A,B and C describe the individuals' contribution to achieve the objectives and additional matters regarding the implementation and evaluation of the SLTravelMate.

1.5 - Summary

As a developing sector of industry in Sri Lanka, tourism needs special attention for its further growth. When traveling abroad, having a method to contact a local person for help is essential for a foreign traveler. Software solutions that exist to facilitate the tourism industry do not provide contacts to such travel guides. We addressed the problem of lacked accessibility with travel guides by implementing "SLTravelMate", a travel guide system with contact methods to travel guides. Several modules are interconnected to cater the needs of the travelers. The users are also categorized to manage the usage of our system.

The next chapter analyses existing travel guide systems to find out what they have achieved as a travel guide system.

Chapter 2

Reviewing others' work

2.1 - Introduction

The previous chapter described the problem that is addressed and the solution we proposed through our project. Now this chapter compares existing travel guide systems to emphasize the technological gaps that are left by them. Several studies have portrayed ICTs as central tools to connect and enable tourist experiences, to promote increased social engagement and involve consumers to co-create experiences. Mobile technologies have been explored as key instruments to promote tourism. Recent work has underlined the value of Smartphone applications to gather information, enrich and construct experiences the use of social networks to support and share on-trip experiences.

2.2 - Other's Approach and Comparison

Table 2.1: Existing travel guide systems and their limitations

Application	Special Features
Trip Advisor TripAdvisor is an American travel website company providing reviews of travel-related content. It also includes interactive travel forums. TripAdvisor was an early adopter of user-generated content. The website services are free to users, who provide most of the content, and the website is supported by an advertising business model	* Popular places in all over the world * Travel information in many countries * Maps integrated with 'Here maps' * Updated Flight Details * Registered travelers have facilities to comment, rate and review for places specified in the website * Mobile Application also available
Colombo City Guide ColomboCityGuide is a Sri Lankan website which include details only about Colombo	* Important places in Colombo city * Travel information related to Colombo city * Maps of Colombo city
Here Map Here Maps is a maps and navigation application by Here for Android, iOS, Windows Phone and the web.	* Live traffic information for more than 40 countries, so you're always up to date with road conditions and disruptions ahead. * Get the best walking directions with alternatives to choose from, helping you to find the shortest and most accurate way to get to your destination.

	* Offline maps & offline navigation

The table 2.1 shows three systems that exist to cater travelers. Even though there are lots of applications to find public and most important places, the analysis done in table 2.1 shows that there is no online communication method for tourists or travelers with a travel guide person. All those Application are based on the Data what are the added by the Application Developers, and no any application permit user to feed data to the System. Even most of application gives the near-by places, all those are lack of details, no visitors' reviews.

2.3 – Summary

The content in this chapter included the similar literature done by others in order to address the same problem discussed in the chapter 1. Trip Advisor is a well-known web application that contains information needed to tourists in most of the countries. Since it has details about many countries, it is more like a general application which is not native for Sri Lanka. City guide and Here maps are two other similar literature to promote tourism. Although there is a considerable amount of applications which are known to be travel guide applications such as trip advisor, city guide, here maps, those applications do not provide direct facilities to chat with travel guides. The main feature of this project is the chatting facility for travelers. The next chapter describes the different technologies that were used to implement the SLTravelMate. It also justifies our choice of technology using proper reasons.

Chapter 3

Adapted technology

3.1 - Introduction

The previous chapter described how other travel guide systems cater the user and what lacks in those existing systems. A proper set of technologies must be present to implement a solution including those lacking features. This chapter gives a description of the technologies that were used to implement the SLTravelMate. Reasons to use such technologies are also included in this chapter to justify our choices.

3.2 - Technologies adapted for the SL TravelMate

The major technologies adapted to implement the SLTravelMate are mentioned here. These choices of technology are justified here by briefly describing the features of each technology and their advantage in implementing SLTravelMate.

3.2.1 - Laravel PHP framework

As described in our objectives, we had to implement a web service, an admin web site and a web application, which are three web-based components in SLTravelMate. Choosing a proper web application framework was vital to reach our objectives.

We used laravel web application framework [4] to implement the above mentioned components. Several features justify our choice.

The framework uses model-view-controller (MVC) architecture [5]. This software architecture pattern reduces the complexity of the system during implementation.

Unlike other popular web development frameworks like for ASP.NET [6], laravel doesn't have a dedicated user interface or a developer environment. Commonly used user interfaces such as command prompt or notepad can be used to develop in laravel. This reduces the usage of the developer's hard disk space or memory for a separate developer environment. The language is mature enough to write completely indipendent libraries.The library size of laravel is about 17MB, comparatively smaller than ASP.NET, which is 5GB [7]. This makes laravel use less memory as mentioned before.

The laravel documentation [8] is highly user friendly and easy to use as guidance while developing. Frameworks like Laravel can make good use of powerful and tested third party libraries without having to reinvent the wheel and rewriting something which already exists. It also provides powerful engines to develop visual content on the web pages. The use of such libraries and engines are described in the Implementation chapter.

PHP programing language [9] was used for the web development. The simplest reason was the fact of laravel being a PHP framework [4]. Many other libraries such as cURL [10] supports in PHP. cURL is an important factor for our project when implementing methods to handle HTTP [11] requests. It is advantageous to use a

language like PHP as it supports such useful libraries. The use of these libraries are also described in the Implementation chapter of this dissertation.

3.2.2 - JSON (JavaScript Object Notation)

Json is a lightweight data interchange format [12]. This format was used in our system to parse data through the web service. This format is language independent [12]. Making it flexible for usage in different applications.

The conventions used in JSON are familiar to the conventions used in other major programming languages such ac C, C++, Java, Python [12]. Coding for the JSON format becomes easier due to this feature.

3.2.3 - Android Studio

Achieving the objective of developing a mobile application is essential for the purpose of easy accessibility and to quickly contact travel guide persons. We had to use a platform which can effectively enable many users to access the SLTravelMate. According to the International data Corporation [13], during the second quarter of 2015, 82% of the worldwide market share of smartphone operating systems was under Android mobile operating system [14]. The graph uses these percentages to show that majority of mobile users are under the Android platform.

Embed

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

Source: IDC, Aug 2015

Table 3.1 : Worldwide Market share of Mobile Operating Systems in the second quarter of 2012-2015

We used Android Studio [15] to develop a single mobile application for the use of travelers. Many other technical features of Android studio made ease of developing the mobile application [16]. Google's Push notification service [17] was another prominent area that could be used to give quick notifications to the mobile user.

3.2.4 – Structured Query Language (SQL)

Much like any other travel guide system, the SLTravelMate is heavily based on data. It must store lots of data related with location and user information and view the data relevant to the user's need. A database is a necessity for such tasks. A proper Database Management System (DBMS) must be chosen to fit our needs.

A DBMS can be either relational, relation-less or in-memory [18]. In-memory DBMS uses the main memory, which is volatile has a limited capacity [18]. This is not suitable to store data of a web-based system having lots of data.

A relation-less or NoSQL DBMS implements a flexible schema for the data [18]. This type is suitable for an environment where the data changes rapidly [18]. Since we take locational data and user information, such rapid change of data is rare to happen. A strongly defined schema would benefit our data which does not change rapidly.

The relational DBMS (or RDBMS) uses the relational model to shape up the information to be stored. These relations can be considered as mathematical sets containing a series of attributes [19]. This mathematical basis makes the relational schema stronger than the other definition of schemas. The RDBMS also supports well with the transactional atomicity, consistency, isolation and durability (ACID properties [20]), which is a compelling feature of RDBMS [18]. The MVC architectural pattern [] of the web-based components of SLTravelMate uses Model classes to handle data. These model classes and relations in RDBMS similar to each other.

Such features makes an RDBMS more suitable for the database of SLTravelMate.

Many popular relational database management systems such as SQLite [21], MySQL [22], and PostgreSQL [23] use the relational model.

MySQL is a popular DBMS among the large scale database servers [19]. Many third party applications and libraries support MySQL, making the implementation and data handling easier [19]. It is also scalable and secure [19], making the data reliable and safe from threats. Software packages such as XAMPP [24] provide many functions for MySQL development, including a designer mode to visually model the database. These features made us choose MySQL database management system to implement our database.

3.2.5 – Apache HTTP server

The web-based components of SLTravelMate needs a web server [25] to host the data relevant to it. Practically web servers need an active internet connection. Web server softwares are available to make servers locally inside the hard disk. Since locally hosted servers do not need an active internet connection, implementation and development of web content will be easy and efficient due to no connectivity interruptions.

Apache HTTP server is the world's most used web server software [26]. It is developed and maintained by an open community of developers under the Apache Software Foundation [27]. Having such an open community-based maintenance means that there is a plenty of community support for errors during the implementation level.

Web server softwares such as XAMPP includes services for Apache HTTP Server, MariaDB database [28], and interpreters for scripts written in the PHP and Perl [29] programming languages. Maria DB is a community developed fork of MySQL DBMS [30]. So for the purpose of web-based development, XAMPP was a useful tool for the project as a local web server and to host and edit the MySQL database of the SLTravelMate without an internet connection.

3.2.6 - Global Positioning System (GPS)

Since SLTravelMate is a travel guide application, it should include a method to show locations of different places within a geographical area. To save such locations and show them to the user, we used the GPS [31] coordinates relevant to that particular place.

3.2.7 – Google Cloud Messaging (GCM)

GCM [32] was used when a specific message need to be sent form the server to a specific traveler's mobile device. This was the best technology that could be used to for the purpose as it was free, easy to implement and ability to handle all aspects of message queuing, support to resend message to an offline device that is registered, is secure and ability to use the free service for moderate testing of the app. It is able to identify each of client's mobile devices uniquely using a specific token and send the required message to the specific device the relevant user has logged in. Even if the user is offline it has the facility to queue the messages and the next time the device is online those messages are sent to the device. It also has the ability to retrieve all the messages without letting overwrite one another.

So because of the above mentioned factors GCM was recognized as the best technology to send messages from server to the traveler mobile devices.

3.2.8 - Asynchronous JavaScript and XML AJAX

In the web application messaging functionality has been implemented by using a set of web development techniques called asynchronous JavaScript and XML [33] which uses many web technologies on client side to create asynchronous web applications. By using the Ajax, it is possible to update the web page without reloading the page, to request new messages from a server after page has loaded, receive the new messages from a server after the page has loaded and send the messages to the server in the background. And also Ajax has the capability of sending and receiving information in a variety of formats including JSON, XML, HTML and even text files. In this web application Ajax will send and receive the information as JSON objects. Usually in SL travel mate chatting will happen between the registered users of the web application and guide but guides will login to the system by using the admin portal so that the communication between the web

application and admin portal will happen by the JSON objects and all the message will be stored in the data base by specifying the sender and the receiver.

3.3 Summary

This chapter justified our choices for specific technologies to implement the SLTravelMate. These technologies were adapted to the project in order to implement the solution which was proposed in the Introduction chapter. Laravel PHP framework was chosen to implement the web application, admin website and the web service. JSON format was used for the purpose of data transfer within the system. MySQL database management system was used here to implement the database. The mobile application was decided to be implemented using Android Studio. GPS technology was chosen to implement the location finding features. AJAX and Google Cloud Messaging services were used to achieve the messaging facility, which is the unique feature in SLTravelMate. The next chapter is allocated to describe how the SLTravelMate project was executed using proper software development methodologies.

Chapter 4

Our Approach

4.1 - Introduction

Since the first chapter of this dissertation, it was known that SLTravelMate is aimed to provide a travel guide system with a messaging facility between travel guides and travelers. After analyzing existing work on a travel guide system, this chapter provides the information on the methods used to reach our aim and objectives in this project. These methodological information is useful to replicate the SLTravelMate System or to further develop a similar system. Application of Software Development Life Cycle and Software process models and the methods used to pass different lifecycle stages are described here.

4.2 Application of the Software Development Life Cycle

Software development life cycle or SDLC is a well-defined, structured sequence of stages in software engineering to develop a software product [34]. There are different SDLC models which can be used during the development process of a software. These models are also known as Software Development Process Models. Waterfall model, incremental model, prototyping model and agile model are examples for such Software Development Process Models [35]. The following stages are visible in any Software Development Process Model.

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

Figure 4.1 clarifies how these six stages are connected in the SDLC.

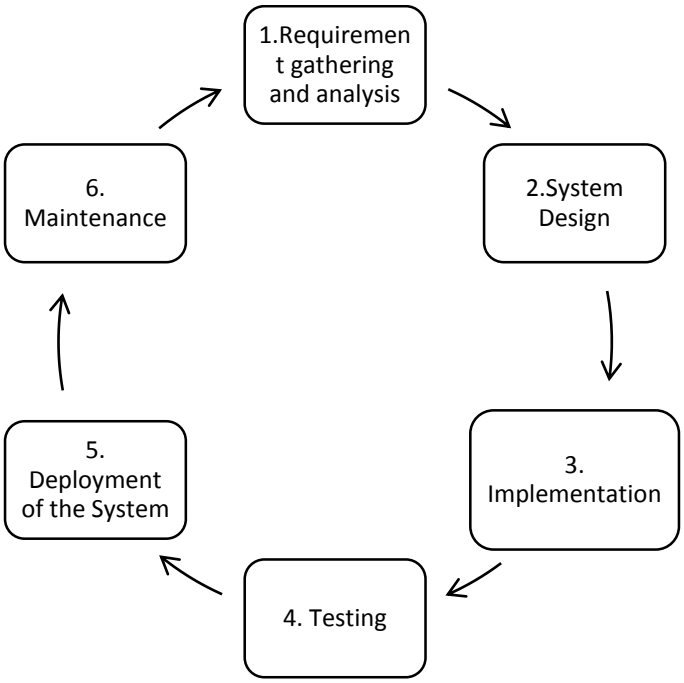


Figure 4.1: Software Development Life Cycle

To use the concepts of a Software Development Life Cycle, a proper Software Development Process Model must be selected in the development of SLTravelMate.

4.2.1- Applying the Waterfall Model for development process.

The Waterfall Model [36] is a SDLC Model with a linear, sequential style. The six stages of SDLC are arranged sequentially as shown in figure 4.2.

The Waterfall Model was chosen to develop SLTravelMate due to the following reasons.

- Requirements of the project were initially defined and fixed by the Industry members.
- The Industry members had given a stable definition of the final product.
- High availability of resources required for the development (e.g.: Software packages, tutorials, etc...).

Other process models such as prototyping model [37] were not used here due to the fixed definition of the product and the requirements.

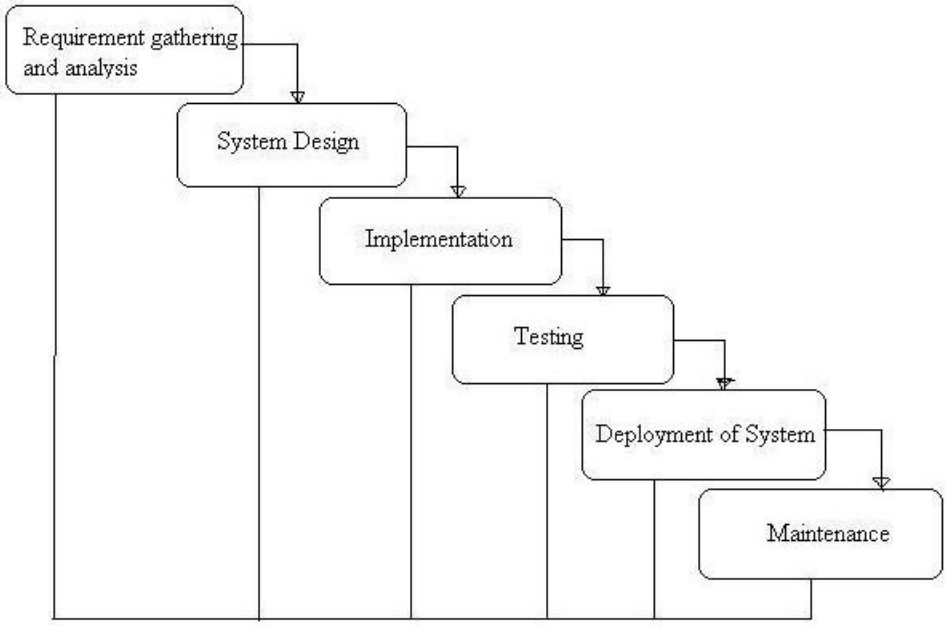


Figure 4.2: Waterfall Model of Software Development

The stages of the development process of SLTravelMate ae described below.

1. Requirement gathering and analysis

Initial meetings were held with the members of Keen Mind Mobile Solutions to introduce their project idea. Requirements were collected with their guidance. A set of basic functional and non-functional requirements are listed below.

Functional requirements of the SLTravelMate system:

- Ability to authenticate the users
- Ability to give different privileges to different users
- Send messages among a selected set of users
- Input text, images and videos.
- Output data relevant to users' request.
- Ability to show a certain place (E.g.: a Hotel) on a map of Sri Lanka.

Non-functional requirements of the SLTravelMate system:

- The system must be scalable.
- Security of data
- User-friendliness.
- Availability to the users
- The system must be recoverable

The initial system analysis defined who will use this system. These types of users are mentioned below. The roles of these users were further explained in the "Introduction" chapter of this dissertation.

Types of users in SLTravelMate

- Travelers : All external users of the SLTravelMate
- Business users : Hotel owners, travel guides, sponsors and other service providers
- System Administrators
- Super Administrator

After defining the requirements and analyzing the system, these information were then used in the system's designing stage.

2. System Design

The defined requirements of SLTravelMate are then used in the next step to design the system properly. Methods such as the Unified Modelling Language (UML) [38]diagrams were used to clarify the design of SLTravelMate. UML consists of structural and behavioral diagrams which visualize the design of the system [39]. Diagrams such as the Use Case diagram were helpful to analyze how users interact with SLTravelMate.

"Analysis and Design" chapter of this dissertation gives more information of the design of SLTravelMate with proper UML diagrams.

Visual methods such as Entity-Relationship diagrams [40]were used when modelling the database. These visualizations of the database also was useful to model the database to fit with the requirements.

Five main components were designed in this stage. Each component with their functions is mentioned in the table 4.1.

Table 4.1: Designed components and their functions

Designed component	Functions
Administration website	• Authenticate system admins • Authenticate business users • Allow travel guides to message travelers • Review and manage the uploaded content
Web application	• Authenticate travelers • Get travel related information • Upload text or images • Message to travel guides
Mobile application	• Authenticate travelers

	• Get travel related information • Upload text or images • Message to travel guides
Database	• Store all uploaded data • Store all user related data
Web service	• Communicate with the database and send data to users.

The messaging facility is a special feature of the SLTravelMate. The table 4.1 shows that this facility should be involved between admin website, web application and the mobile application.

The design of SLTravelMate was then finalized and ready to be implemented.

3. Implementation

The final design of SLTravelMate was then used in the implementation stage. The technologies that were adapted in the "Technology Adapted" chapter were adopted here to implement the main components of SLTravelMate. A major part of the development process was given to the implementation stage.

The Laravel PHP framework was used here to implement the web service, web application and the administration website. Android Software Development Kit was used to implement the mobile application. MySQL database management system was used to implement the database and the Apache HTTP server was used for the hosting purposes in this stage.

The design stage defined that the messaging function should happen between the admin portal, web application and the mobile application. Hence, this function was implemented in these three components.

The "Implementation" chapter gives further information on how each component was implemented with the adapted technologies.

4. **Testing**

   The implemented components of the SLTravelMate were then tested against the requirements that were defined in the requirement gathering stage. Several test cases were applied to the components of SLTravelMate and the outcomes were evaluated afterwards. This stage made sure whether the system actually solves the problem that was meant to be addressed.

   The test cases and their evaluation are further explained in the "Evaluation" chapter of this dissertation.

   The system was then ready to be deployed to a shared server and for the public use. This was handled by the members of the industry.

5. **Deployment and Maintenance**

   SLTravelMate will be maintained with the supervision of the industry members. The system will be further developed with the user inputs and the changes of technology.

4.3– Summary

This chapter described the methodologies that were used to analyze, design, implement and evaluate the SLTravelMate before deploying it as a final product for maintenance. The Waterfall model of the Software Development Life Cycle was applied to develop the project through a sequence of well-defined stages. Our approach to SLTravelMate started with gathering and defining the requirements. Then the system was designed with the guidance of the defined requirements. Adapted technologies were then adopted to implement the designed system. Testing the implemented system made sure whether the initially identified problem of the project is being addressed by the implemented solution. The next chapter of this dissertation will give information about the design of the system.

Chapter 5

Analysis and Design

5.1 – Introduction

The previous chapter described how the proposed solution was achieved using proper methodologies. Now, this chapter gives a detailed analysis of the SLTravelMate system. It also describes the design of the solution and the functions of each separate component designed for this solution.

5.2 – Analysis of the Sri Lankan Travel Mate System

The following diagram shows a very high level interpretation of the design of our proposed solution.

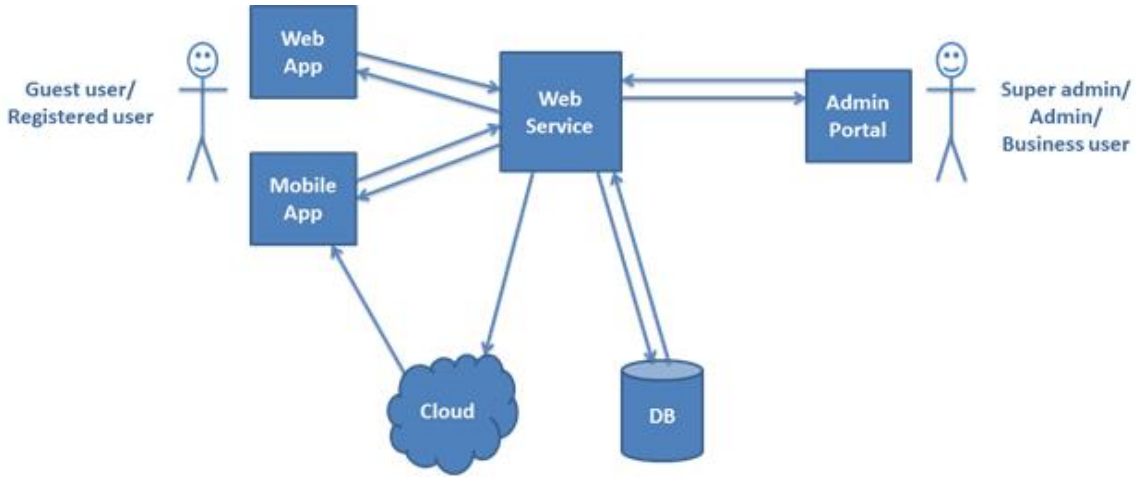


Figure 5.1: High level system diagram of the proposed solution

This consists of three modules and the database connected by a web service. The three modules include,

1. Web Application
2. Mobile Application
3. Admin Portal

5.2.1 - Web Application:

This is the website where general users such as travelers are interacting with the system. They can view the items belongs to our defined categories. They can register in the system through the Web Application and enjoy the additional privileges offered by the system. Those additional privileges include comment, rate, send photos and messages to admin panel and guides, and see the online status of the guides.

5.2.2 - Mobile Application:

This is the android application where all the features of the website are included along with some special features. The registered users who have installed the android application will have the facility to select their preferences to get push notifications with latest updates in the Tourism Industry and our system.

5.2.3 - Admin Portal:

This is the website where business users and admins are logging into the system. The business users cannot directly register in the system, they have to send a request to join in the system. Once the request is accepted by the admin panel only the business user get privileges to enter into the system. The business users can always send requests to admin panel to add their items to the available categories in the system. They can also request for a new category type to be implemented in the system. That kind of category requests is handled by the super user who is above all. The business users who had paid and obtained the sponsorship of the system can maintain their own pages in the Web Application and have the facility to publish advertisements in their pages. For this we need to use a payment gateway. Guides also can register in the system as business users. They have the special privilege of setting their online status, so that the registered travelers can see the guides who are available to send a message at the moment. Admins can collect the photos sent by the travelers to the admin panel and customize the website accordingly. They have the ability to filter user comments, approve and reject business user requests, approve or reject item requests.

5.2.4 - Web Service:

All the above modules we discussed were connected to the database through the web service. the web service responsible for the task of handling the requests coming from the Web Application, mobile Application, and the Admin Portal and serving those requests with appropriate responses.

5.2.5 - Database:

Database is where all the information is stored. It contains the tables such as users, categories, business users, admins, messages, comments, etc. The Entity-Relationship Diagram of our system is given in figure 5.2.

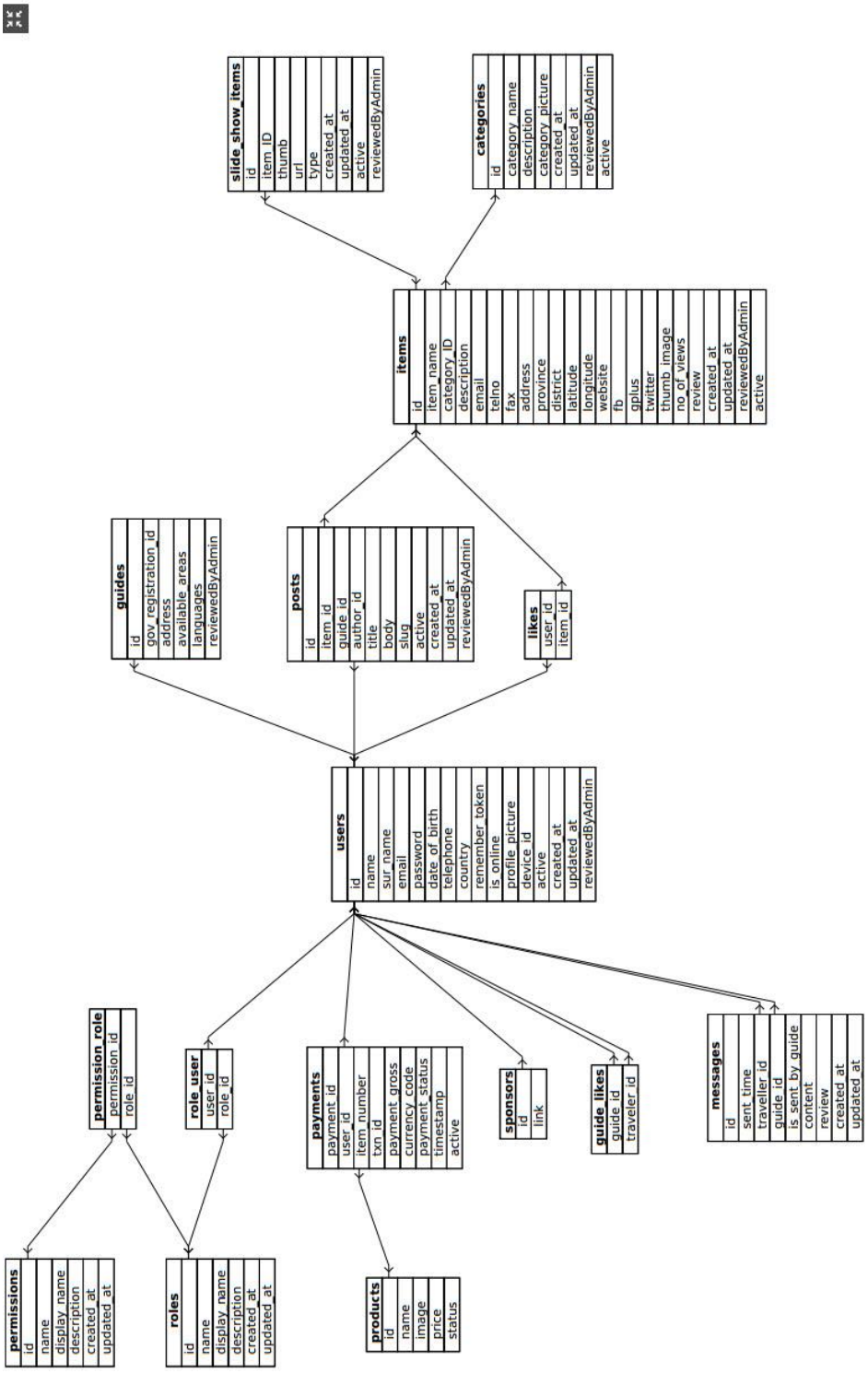


Figure 5.2: Entity Relationship Diagram of SLTravelMate

figure 5.3 shows how the users interact with the system. Unified Modelling Language (UML) [38] is used here to design this use case diagram.

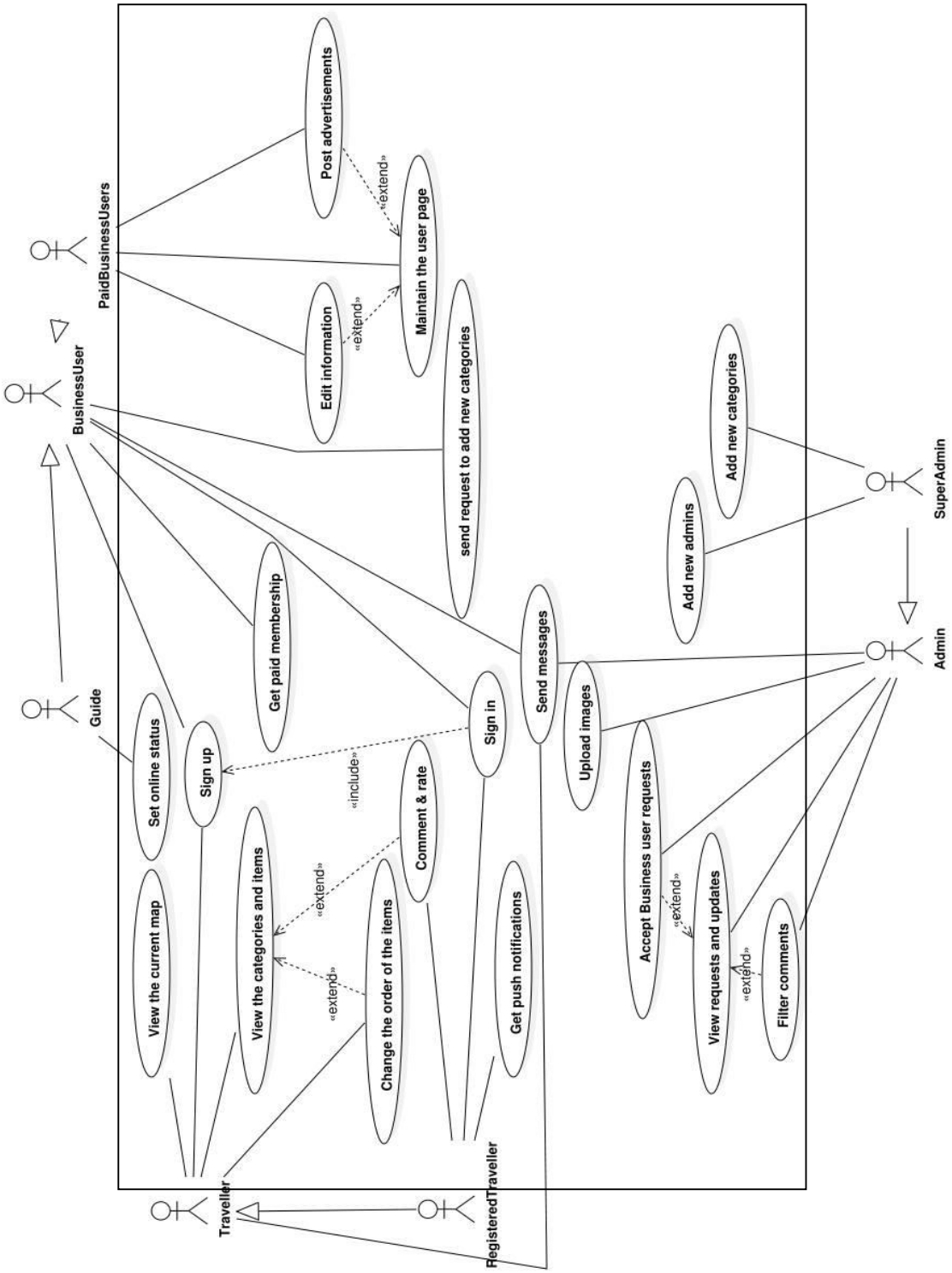


Figure 5.3: Use Case diagram of SLTravelMate

5.3 - Summary

This chapter gave a further analysis of how SLTravelMate system was designed after the initial analysis of the system. Visual methods such as the Unified Modelling Language (UML) were used to design each component of the system. The database too was visually designed using an Entity-Relationship diagram. The design procedures resulted with an outcome of five separate components for SLTravelMate: a web service, an administration website (admin portal), a web application, a mobile application, and a database. Each component described in this chapter comprises of functions and attributes that are designed to fit with the set of requirements which were defined in the initial stages of the development process. The different ways each user interacts with the system's components were also described in this chapter's use case diagram. The designed components were then implemented using specified technologies which were described in the "Technology Adapted" chapter. Information regarding the implementation stage will be described in the next chapter "Implementation".

Chapter 6

Implementation

6.1 – Introduction

The previous chapter described the design of the SLTravelMate and analyzed the function of each component of the designed system. This chapter is about how the designed components were implemented with the specific technologies which were chosen before in the third chapter.

6.2 – Implementation of the SL TravelMate

As described in the previous chapters, SLTravelMate consists of five main components. The technologies chosen by us were used to implement those components.

6.2.1 - Web service:

Laravel framework was used to implement the entire web service. As described in previous chapters, the Web Service used the MVC architectural pattern. All the controller classes execute the functions of the Web Service. The function of handling requests and responses was implemented using cURL libraries, which are supported in PHP [10]. The code segment 6.1 shows how cURL sends a HTTP GET request and returnes the response into the program logic. Code segment 6.2 shows how a HTTP POST request is made using cURL. The JSON format is used to encode the data while transferring the data. The JSON format also decodes them before they are stored in the database or before the data is output as visual content in web or android components of SLTravelMate. Code segment 6.1 also shows how the json_decode function is added into the code in such a way that the HTTP response with JSON data gets decoded and returns the original data.

```
$url=”http://localhost/webservice/items/getallitems”;

//  Initiate curl
$ch = curl_init();
// Disable SSL verification
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
// Will return the response, if false it print the response
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
// Set the url
curl_setopt($ch, CURLOPT_URL,$url);
// Execute
$result=curl_exec($ch);
// Closing
curl_close($ch);

// Will dump a beauty json :3
var_dump(json_decode($result, true));
```

Code Segment 1: Sending a HTTP GET request into the web service using cURL

```
$url =
'http://localhost:81/WebService2/public/portal/item/updateitem';

//  Initiate curl
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, $url);

curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

curl_setopt($ch, CURLOPT_TIMEOUT, 3);

$response = curl_exec($ch);

curl_setopt($ch, CURLOPT_POST, 1);

//post parameters
$string='item_ID=41&item_name=RiivinkaIslands&category_ID=1&descript
ion=goodHotel&email=nimmidotcom&telno=0774933705';
curl_setopt($ch, CURLOPT_POSTFIELDS, $string);

$respose = curl_exec($ch);

curl_close($ch);

print_r($respose);
```

Code Segment 2: Sending a HTTP POST request into the web service using cURL

As described in the Technology Adapted chapter, different third party libraries were used to implement the web service.

6.2.2 - Database:

We used a MySQL database management system to create our database. To match with the architectural pattern of the web-based components, we used the structure of the model classes used by the model-view-controller pattern. The table structure of our database is given in the ER diagram in figure 5.2. The data movement between web service and the database is managed by the controller classes and model classes that were implemented in the web service. For the purpose of giving special privileges to users, database-level roles [41] were added to the database as additional tables. This method helps to limit the accessibility of different users to specific contents of the system. Only the super admin is given with most privileges so that the super admin can access any content in the database. Even though the initial table generation was done using queries, the newly added tables such as the database-level roles were modeled using the designer view provided by the XAMPP software. The database was locally hosted using Apache HTTP server during the modelling phase and the data feeding phase.

6.2.3 - Web Application:

The web application provides the interfacing facility between guest users and registered traveler users. The MVC architecture was used to implement the web application. We used bootstrap [42] which is an open source Cascading Style Sheet (CSS) [43] language to edit the styles of the web pages. The message passing between Web Application and the Web Service is implemented by Views of the MVC architecture. As described in "Adapted Technology" chapter, the development of the web application was supported by many powerful engines and libraries. The Blade [44] templating engine is such an engine provided by laravel. To show the privileges of logged users and guest users we use blade templating engine. All the Views of the web-content are saved as "view name.blade.php" to use the Blade engine. As shown in code segment 6.1, HTML [45] is used to write the visual content of the View php files.

```
@foreach($hotel as $hotelname):
<li class="im-g"><a href="#"><img
src="{{$hotelname['thumb_image_in_list']}}"  class="img-responsive"
alt=""></a></li>
<li class="data">
<h4>{{$hotelname['item_name']}}</h4>
<p>{{$hotelname['description']}}</p>
<li class="bt-nn">
<a class="hvr-shutter-in-horizontal button"
href="{{URL::route('categorylist',[$hotelname['item_ID']])}}">View
Page</a>
</li>
<div class="clearfix"></div>
</div>
@endforeach;
```

Code Segment 3: A sample Code Segment of a .blade.php file

6.2.4 - Admin portal:

The Admin Portal and the Web Application both requires the same tools when developing the system because both are having common features such as logging, registration, etc. Additional to common features admin portal needs special attention when implementing security. The links should be more secured than the web application because this is more like an internal network and it consists of a payment gateway

6.2.5 - Mobile Application:

The mobile Application is to be built using the Android software development kit. The IDE we are using is the Android Studio. The manifest file which contains the basic details about the activities and screenshots of the mobile interfaces are included in Appendix B.

Main Activity is the first interface displayed in the mobile application. After that it can navigate to other activities based on the user actions. Every activity contains a java file which contains the functions related to that activity (ex. Button clicks) and a XML file which defines the design of the activity (ex.Textviews).

Volley

HTTP communications between the android app and the web service were achieved with the Volley [46] library for Android. The responses from the web service are received in the json format. To send both POST requests and GET requests methods in volley library are used. Using Volley library the number of lines of the code could be further reduced. One of the most important advantages of using volley library is that the images can be downloaded and cached efficiently. Otherwise it takes a considerable amount of time to do that.

Lists

There are no of places where lists are used (ex. Category list in main activity, category items lists, list of comments, list of conversations, list of messages in a conversation etc…). To create these lists have to create a separate layout file to define the design for one list item. Adapters which are a type of classes in android are used to create the listview using the above mentioned layout file for one list item. OnClickListeners [47]are also written to indicate the actions to be performed when a specific item from the list is selected (ex. start the next activity that contains more details of the selected item.).

SQLite Database

In order to show the details when offline, the data should be stored in a database. For that the data are stored in the SQLite [48] database after extracting from json files. So when the device is connected to internet, the details displayed are extracted from the json files and also the SQLite database is updated. When offline the data stored in the SQLite database are used.

Google Maps

Google maps were used at several places in the mobile application. To integrate Google Maps with the application, a project was created for the mobile application and Google Maps API was enabled. Then a key was obtained from the Google play services.

Markers of the locations are shown in the map when their latitude and longitudes are given. A custom info window was created with a layout file to be displayed when the

marker is selected. An OnClickListener was written to open the relevant activity when the custom info window was clicked.

Fragments

Fragments [49] can be used within an activity. There are several places where fragments are used in the application (ex. In the second activity for two tabs-list tab and map view) . Using fragments different parts of the same activity can be handled independently. For an example, if it is needed to disappear a certain part of the activity on a button click, it can be achieved using fragments. All fragments within the same activity can access the attributes in that activity. And also, when a set of interfaces have to use the same set of attributes, and if activities are used for the interfaces instead of fragments, it is necessary to pass all those attributes from one activity to another. But if one activity and a set of fragments are used, it is not necessary to pass those attributes. All fragments can access the attributes of the activity they belong. Therefore using fragments can eliminate the burden of passing attributes between intents.

Images and videos upload

To use camera to capture images or record videos, existing android camera application is used. In mobile applications it is possible to use the other applications in the android phone using intents. For an example following line is used to start camera to capture image.

```
Intent intent = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

Code Segment 4: Using intents to enable the camera application

After capturing the image, a location to temporary store that image can be specified. So that when sending the image for the path it can be used. Videos are recoded and stored in the same way.

When selecting an image or a video from the gallery, a file chooser is created to specifying the type of file. Then it opens an activity containing all the files of that type in the phone. After selecting the image or the video, that path is sent to convert to a bit stream. This is done using Multipart Entity [50]. It converts a file to a bit stream and sent to web service specifying the type of the file. Web service on the other side decodes it and store in the relevant folder. The code fragment 6.5 below is a part of the code used to convert to multipart entity and send it to web service.

Code Segment 5: Use of Multipart Entity

```
HttpClient client = new DefaultHttpClient();
HttpPost postMethod = new
HttpPost("http://localhost/Upload/index.php");
File file = new File(filePath);
```

```
MultipartEntity entity = new MultipartEntity();
FileBody contentFile = new FileBody(file);
entity.addPart("userfile",contentFile);
StringBody contentString = new StringBody("This is contentString");
entity.addPart("contentString",contentString);

postMethod.setEntity(entity);
client.execute(postMethod);
```

Push notifications

To enable the web service to identify a specific device a specific user has logged in and send messages relevant to that user, Google Cloud Messaging [32] was used. For that the app had to be registered with the GCMServer to get Server API key and sender ID.

First a project had to be created in the name of the application name. The package also had to be specified. Then the Server API key and the sender ID were obtained. Then the google-services.json file which is created uniquely to the application is downloaded and integrated with the application.

Next build.gradle was configured for both app and the project. Following are the lines used.

App:

```
apply plugin: 'com.google.gms.google-services'
```

and for dependencies

```
compile 'com.google.android.gms:play-services:8.3.0'
```

Project:

For dependencies

```
classpath 'com.android.tools.build:gradle:1.2.3-alpha6'
classpath 'com.google.gms:google-services:1.5.0-beta2'
```

When specifying the versions, it is important to consider about the versions of the google play services used in the application.

After that following classes were written.

- GCMRegistrationIntentService.java- to send request to GCM server to register the device.
- GCMTokenRefreshListenerService.java –to handle token change and then send registration request to GCM Server for new token.

- GCMPushReceiverService.java to receive message from the GCM server and then put it as a push notification to user interface
- Inside ShowMessages activity BroadcastReceiver is created to handle message from service.

Then the manifest file was configured. As mentioned before, the final manifest file is included in the Appendix B.

If briefly explained, following were the steps of how the GCM push notifications are used.

- When a specific traveller logged in with a specific android phone, GCM registration id was taken by sending the sender ID to GCM server which was obtained when registering the application in the GCM server.
- That registration id is sent to web service along with the user id. Then the web service stored it with the user id in the database. That id was always updated with the traveler login in a different device or when the token change happened.
- Once the web service received a message from a specific guide to a specific traveler, it looked for the registration id (device id) for the specified traveler and sent a request to GCM server with the server API key, message to be sent and the registration id or the device id.
- Then the GCM server identified the device the message had to be sent and checks whether the device is online. If the device is offline, it uses a message queuing and stored the messages. Once the device is online the stored messages are sent to the device.
- Here at first the received messages were overwritten by the new messages received. To receive all the messages, a method was written in the mobile application to avoid from overwriting one another.
- Once the message is received as a push notification, an OnClickListener was written to display the whole conversation. Before opening the conversation the user ids are again checked.
- The messages were received even the application was not opened at a particular time.
- When the user viewed the new messages a method in the web service was called. So that the view status of messages could be used to show the traveler the no of new messages for a particular conversation when viewing the list of conversations.

6.3 - Summary

This chapter shows how the web service, database, web application, mobile application and the admin portal were implemented with the specific technologies. In SL travel mate web service has been implemented in laravel framework by following the MVC architecture. The function handling requests and responses was implemented using CURL libraries and the JASON format is used to encode the data while transferring the data. MySQL database management system has been used to implement the data base and the data movement between the web service and the data base is managed by the controller classes and the model classes in the web service. Web application provide the interface for user interactions, These interfaces have been designs using an open source style sheet language called bootstrap and to show the privileges of the logged users and guest user blade template engine provided by the Laravel has been used. For implementing the chat functionality between the web application users and the guides a set of web development technique called Ajax has been used to update the web page or to receive the messages sent by the guide and to send the messages to the server in order to deliver to the guide without reloading the page. Mobile application of SL travel mate has been built using the android software development kit by using the Android studio IDE and the HTTP communication between the android application and the web service were achieved with the volley library. Some features of the Admin portal of SL travel mate like logging, registration and chat functionality could be implemented by using the same tools that used for the web application, but the links that will be used to call the routers of the admin portals should be more secured than the web application.

Chapter 7

Evaluation

7.1 – Introduction

The previous chapter described the implementation of each component in the SLTravelMate. Here in this chapter, we evaluate the SLTravelMate with regards of it's outcomes.

7.2 – Evaluation of the results.

User centric evaluation and system centric evaluation methods are used here to evaluate the results of each separate component of the the SLTravelMate.

In a user centric level, the utility and the value of the system to the user is evaluated. Different test procedures are executed to mock a user interaction and the outputs are then analyzed. This method is used below in table 7.1 to evaluate the web service.

Table 7.1 : User centric evaluation of web service.

Description	Test Procedure	Expected Output	Achieved/Not
User login	Not entered password	Display Error Message	Achieved
	Enter incorrect password	Display Error Message	Achieved
	Enter correct password	Redirect to the relevant page	Achieved
User registration	Enter already registered email	Display Error Message	Achieved
	Enter invalid data	Display Error Message	Achieved
	Enter incorrect Confirm Password	Display Error Message	Achieved
	Enter valid data with correct confirm password	Registering the user	Achieved
Authenticating to Web application or Mobile Application	Enter email and password of a business user	Permission not allowed Message	Achieved
	Enter email and password of a Admin or super admin	Permission not allowed Message	Achieved
	Enter email and password of a registered traveler	Login if email and password matched	Achieved

Authenticating to Admin Portal	Enter email and password of a business user	Login if email and password matched	Achieved
	Enter email and password of a Admin or super admin	Login if email and password matched	Achieved
	Enter email and password of a registered traveler	Forbidden Message	Achieved
Hide admin portal routes	Registered traveler tried to access	Forbidden Message	Achieved
	Registered business user tried to access	Forbidden Message	Achieved
	An admin tried to access	Allow access	
Messaging	Unregistered user trying to send a message to a guide	Message asking to be login	Achieved
	Registered user trying to send a message to a guide	Allow to send message	Achieved

System centric evaluation of the SLTravelMate's web service can be described under the following categories.

- Performance – for example Response Time, Throughput, Utilization, Static Volumetric
- Scalability
    - Since Client side applications and Server side backend was separated in this project, further scaling could be done in future. At the moment client applications were built for both mobile and web. Admin portal was only made for web, but it also could be extended to mobile versions in future.
- Availability
    - Travelers can either log into the system from Mobile android Application or Web Application. Availability is most of the users.
- Reliability
    - CSRF protection was implemented to stop a unknown third party intermediation when transferring data though web. Dingo api was used for Exception handling and error handling to increase the reliability and self recoverability.

- Maintainability and Manageability
    - Since Client side applications and Server side backend were separated, managing parts of the system is easy. Client web application and server web application need not necessarily be in the same web server. However changes could be done maintain the common protocol to keep the interoperability alive.
    - For the web development, laravel was used as the framework. MVC architecture was the driver in laravel. Therefore development was easy and it will be easy to edit the code in the future when it is necessary.
- Serviceability
    - A variety of open source plug-ins were available for server the web development. Following were some of the plug-in used to build the system.
    - Ex:
    - Tymon\JWTAuth for token based authentication
    - Dingo\api for Exception handling and error handling
    - Zizaco\Entrust for multiple role-permission authentication
    - Tinymce for image and text uploading in we app
- Security

CSRF (Cross Side Request Forgery) protection was used to block regular online attacks that hackers used to do with a web site's form submitting page [51].

The same user centric and system centric methods were used to evaluate the Android application. The table below shows the user centric evaluation.

Table 7.2 : User centric evaluation of Mobile Application

Description	**Test Procedure**	**Expected output**	**Achieved or not**
User login	Trying to login without internet connection	Display error message	Achieved
	Not entering email or password	Display error message	Achieved
	Entering incorrect email or password	Display error message	Achieved
	Entering correct email and password	Redirect to the relevant activity	Achieved
	Login with gplus account	Register user and redirect to the relevant activity	Achieved

Registration	Enter already registered email	Display Error Message	Achieved
	Enter invalid data	Display Error Message	Achieved
	Enter incorrect Confirm Password	Display error message	Achieved
	Enter valid data with correct confirm password	Registering the user	Achieved
Messaging	Trying send a message without login	Directs to login activity	Achieved
	Trying send a message after login	Allow messaging	Achieved
Photo/Video uploading	Trying to upload photos without login	Directs to login activity	Achieved
	Trying to upload photos after login	Allow photo uploading	Achieved
Commenting	Trying comment without login	Directs to login activity	Achieved
	Trying comment after login	Allow commenting	Achieved
View conversations	Trying to view conversations without login	Directs to login activity	Achieved
	Trying to view conversations after login	View conversations	Achieved
Change password	Trying to change password without login	Directs to login activity	Achieved
	Logged in user enter wrong current password	Display error message	Achieved
	Logged in user enter correct current password but confirmed new password do not match with new password	Display error message	Achieved
	Logged in user enter correct current password and confirmed new password and new password match	Allow change password	Achieved

The system centric evaluation was also done to the android application. The table 7.3 shows the system centric evaluation of Android application

Description	Achieved or not	Reasons for failures
Send data to server (messages, comments, data for authentication and registration)	Achieved	
Upload photos	Achieved	
Upload videos	Not achieved	Some videos could not be uploaded because they exceed the maximum file size that can be uploaded to server.
Display details when offline	Achieved	
Display details when online	Achieved	
Download and display images/play videos efficiently	Achieved	
Navigation between activities	Achieved	
Keeping the SQLite database up-to-date with server side data	Achieved	

Table 7.3: System centric Evaluation of Android Application

Again the web application too was treated with the same two evaluations. The tables 7.4 and 7.5 shows the evaluation of the web application.

Table 7.4: User centric evaluation of web application

Description	Test Procedure	Expected output	Achieved or not
Login	Not entering email or password	Display error message	Achieved
	Entering incorrect email or password	Display error message	Achieved
	Entering correct email and password	Redirect to the relevant activity	Achieved

	Login with GooglePlus or facebook account	Register user and redirect to the relevant activity	Achieved
Registration	Enter already registered email	Display Error Message	Achieved
	Enter incorrect Confirm Password	Display error message	Achieved
	Enter valid data with correct confirm password	Registering the user	Achieved
Messaging	Trying send a message without login	Directs to login activity	Achieved
	Trying send a message after login	Allow messaging	Achieved
Photo/Video uploading	Trying to upload photos without login	Directs to login activity	Achieved
	Trying to upload photos after login	Allow photo uploading	Achieved
Commenting	Trying comment without login	Directs to login activity	Achieved
	Trying comment after login	Allow commenting	Achieved
View conversations	Trying to view conversations without login	Directs to login activity	Achieved
	Trying to view conversations after login	View conversations	Achieved
Change password	Trying to change password without login	Directs to login activity	Achieved
	Logged in user enter wrong current password	Display error message	Achieved
	Logged in user enter correct current password but confirmed new password do not match with new password	Display error message	Achieved
	Logged in user enter correct current password and confirmed new password and new password match	Allow change password	Achieved

Table 7.5: System centric evaluation of the web application.

Description	Achieved or not	Reasons for failures
Send data to server (messages, comments, data for authentication and registration)	Achieved	
Get relevant data from sever through web service and displaying	Achieved	
Upload photos	Achieved	
Upload videos	Achieved	
Messaging between sender and receiver	Achieved	

7.3 - Summary

This chapter shows how can be evaluated the each components of SL travel mate with regards to its outcomes by using user centric evaluation and system centric evaluation methods. In user centric evaluation by executing different test procedures the value of the system to the user is evaluated. Under this evaluation method the main features of SLTravelMate like photo or video uploading and commenting will be evaluated as well the basic feature like user login, user registration, Authentication hiding admin portal router, view conversation and change password also will be evaluated under the user centric evaluation method. When a system be evaluated in a user centric evaluation method different combinations of user input will be entered to the system and check whether the system is only working for correct combination. Other than this evaluation method SLTravelMate will evaluated base on the system centric evaluation method in that method the features like performance, Scalability, Availability, Reliability, Maintainability and Serviceability will be evaluated. The next chapter will conclude the dissertation with suggestions for further development of SLTravelMate.

Chapter 8

Conclusion and Further work

8.1 – Introduction

After evaluating the results of the implemented system, this chapter analyses the overall achievements and findings of implementing the SLTravelMate. This chapter also describes the possible developments that would benefit the future of SLTravelMate system.

8.2 – SLTravelMate as a travel guide

The efforts of this project were aimed to design, implement and develop a travel guide application. Studying about Sri Lankan Tourism industry revealed that there are travel guides who are registered under government authority. After analyzing existing travel guide systems, the lack of a method to contact local travel guides was found out to be a common issue. To address this problem, we designed SLTravelMate, a native travel guide system with methods to contact registered travel guides in Sri Lanka.

The implemented solution is of five main components. A web application, a database, an admin portal, an Android mobile application and a web service to properly connect those four components. Since the Model-View-Controller Architecture's less complexity, it was used to design the web-based components of SLTravelMate. Laravel PHP framework's support to MVC architecture became useful when implementing these components. Laravel's support to powerful third party libraries was also beneficial during implementation. Engines such as Blade templates were efficient in developing the front-end visuals of the web application. cURL libraries were useful for HTTP requests.

The implemented web service is able to process requests to get data from the database and to post data into the database. Data is encoded to JSON when they are trensfered through the web service.

The database was implemented with the use of XAMPP's local host services. The MySQL database comprises of tables which are compatible with the Model classes of the MVC patterned web-based components. A set of tables were added to give privileges to different user types. These tables give database level roles to each user type.

Web application acts as a front-end to the users. It facilitates users to register into SLTravelMate and make posts relevant to travelling. Laravel's support to external libraries was used to enhance the function of posting.

Admin portal, which was also based on laravel, retrieves content and reviews them before being posted to the public. The facility of paid sponsorship needs more clarifications to complete the payment gateway.

Mobile application which was implemented with Android Studio, gets required content from the database for viewing. It could use more methods to sign up by using

socil media APIs. GPS coordinates of a retrieved location can be returned into a map through the mobile application.

A user centric method of evaluation was mainly used to evaluate the major components of this system. The high number of users interacting with the system and the fact of users being categorized with different privileges made the user centric method more beneficial in evaluating this system. It proved that even though the main objective of implementing the main components was completed, their connectivity shows different malfunctions. Further development focused on these malfunctions will enhance the user experience attainable by using SLTravelMate.

8.3 - Further developments

The payment gateway connected with the admin portal needs further attention. It garners the business model of SLTravelMate using the business users of the system. More integration from social media would make this system more popular among the travelers. Our solution proposed to gather revenue by sponsorships of the business users. Improving the business model of the SLTravelMate would gather more revenue to the developers and admins. This is beneficial to maintain and develop the system for more users.

8.4 - Summary

This chapter concluded the dissertation of SLTravelMate; a system that uses web and mobile platforms to provide useful travel related details to foreign travelers in Sri Lanka. A web application and an Android mobile application serves the foreign travelers to collect travel-based information and upload new content. A panel of admins control the SLTravelMate system through an Admin website. A web service passes data between the users and the database. A specially integrated messaging service gives the travelers to directly connect with registered travel guides in Sri Lanka. Further developments could be made to this system to attract more users and collect revenue. The upcoming sections of the dissertation provides additional information regarding the implementation and evaluation of SLTravelMate.

References

[1] S. L. T. D. Authority, "About Us," [Online]. Available: http://www.sltda.lk/about_us.

[2] S. L. T. D. Authority, "Registration of Tour Guides," [Online]. Available: http://www.sltda.lk/registration_of_tour_guides.

[3] S. L. T. D. Authority, "Travel Agencies," [Online]. Available: http://www.sltda.lk/travel_agencies.

[4] T. Otwell. [Online]. Available: https://laravel.com/.

[5] tutorialspoint, "Basic MVC Architecture," tutorialspoint, [Online]. Available: http://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm.

[6] Microsoft, "ASP.NET," Microsoft, [Online]. Available: http://www.asp.net/.

[7] vsChart.com, "Laravel vs. ASP.NET MVC," [Online]. Available: http://vschart.com/compare/laravel/vs/asp-net-mvc-framework.

[8] T. Otwell, "Laravel," [Online]. Available: https://laravel.com/docs/5.2.

[9] T. P. Group, "php," [Online]. Available: http://php.net/.

[10] PHP, "Client URL Library ¶," [Online]. Available: http://php.net/manual/en/book.curl.php.

[11] wikipedia.org, "Hypertext Transfer Protocol," [Online]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

[12] "Introducing JSON," [Online]. Available: http://www.json.org/.

[13] I. D. Corporation, "About IDC," [Online]. Available: http://www.idc.com/about/about.jsp.

[14] I. D. Corporation, "Smartphone OS Market Share, 2015 Q2," [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[15] Google, "Android Studio," [Online]. Available: https://developer.android.com/studio/index.html.

[16] Google, "Android Studio," [Online]. Available: https://developer.android.com/studio/index.html#features.

[17] Google, "Push Notifications," [Online]. Available: https://developers.google.com/drive/v3/web/push.

[18] C. S. Mullins, "Evaluating the different types of DBMS products," searchdatamanagement, [Online]. Available: http://searchdatamanagement.techtarget.com/feature/Evaluating-the-different-types-of-DBMS-products.

[19] O. Tezer, "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems," digitalocean.com, [Online]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems.

[20] wikipedia.org, "ACID," [Online]. Available: https://en.wikipedia.org/wiki/ACID.

[21] SQLite, "SQLite," [Online]. Available: https://www.sqlite.org/.

[22] "MySQL Documentation," Oracle Corporation, [Online]. Available: https://dev.mysql.com/doc/.

[23] PostgreSQL, "PostgreSQL," PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/.

[24] wikipedia.org, "XAMPP," [Online]. Available: https://en.wikipedia.org/wiki/XAMPP.

[25] V. Beal, "Web server," [Online]. Available: http://www.webopedia.com/TERM/W/Web_server.html.

[26] wikipedia.org, "wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/Apache_HTTP_Server.

[27] wikipedia.org, "Apache Software Foundation," [Online]. Available: https://en.wikipedia.org/wiki/Apache_Software_Foundation.

[28] wikipedia.org, "MariaDB," [Online]. Available: https://en.wikipedia.org/wiki/MariaDB.

[29] wikipedia.org, "Perl," [Online]. Available: https://en.wikipedia.org/wiki/Perl.

[30] wikipedia.org, "MariaDB," [Online]. Available: https://en.wikipedia.org/wiki/MariaDB.

[31] Wikipedia, "Global Positioning System," [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System.

[32] developers.google.com, "Google Cloud Messaging: Overview," [Online]. Available: https://developers.google.com/cloud-messaging/gcm.

[33] careerride.com, "What Is AJAX? Advantages of AJAX, History of AJAX," [Online]. Available: http://www.careerride.com/Ajax-Defined-History-Advantages.aspx.

[34] Tutorialspoint.com, "Software Development Life Cycle," [Online]. Available: http://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm.

[35] ISTQBExamCertification.com, "What are the Software Development Life Cycle (SDLC) phases?," [Online]. Available: http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/.

[36] ISTQBExamCertification.com, "What is Waterfall Model," [Online]. Available: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/.

[37] ISTQBExamCertification.com, "Prototyping Model," [Online]. Available: http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/.

[38] Wikipedia.org, "Unified Modeling Language," [Online]. Available: https://en.wikipedia.org/wiki/Unified_Modeling_Language.

[39] creately.com, "UML Diagram Types," [Online]. Available: http://creately.com/blog/diagrams/uml-diagram-types-examples/.

[40] wikipedia.org, "Entity Relationship Model," [Online]. Available: https://en.wikipedia.org/wiki/Entity–relationship_model.

[41] Microsoft, "Database-Level Roles," [Online]. Available: https://msdn.microsoft.com/en-us/library/ms189121.aspx.

[42] @. a. @fat, "Bootstrap," GitHub , [Online]. Available: http://getbootstrap.com/.

[43] www.w3.org, "Cascading Style Sheets," [Online]. Available: https://www.w3.org/Style/CSS/Overview.en.html.

[44] T. Otwell, "Blade Templates," Laravel , [Online]. Available: https://laravel.com/docs/5.1/blade.

[45] w3schools.com, "Hyper Text Markup Language," w3schools.com, [Online]. Available: http://www.w3schools.com/html/html_intro.asp.

[46] R. Tamada, "Android working with Volley Library," [Online]. Available: http://www.androidhive.info/2014/05/android-working-with-volley-library-1/.

[47] developer.android.com, "View.OnClickListener," [Online]. Available: https://developer.android.com/reference/android/view/View.OnClickListener.html.

[48] developer.android.com, "SQLiteDatabase," [Online]. Available: https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html.

[49] developer.android.com, "fragments," [Online]. Available: https://developer.android.com/guide/components/fragments.html.

[50] developer.oesf.biz, "MultipartEntity.," [Online]. Available: http://developer.oesf.biz/em/developer/reference/eggplant/com/android/internal/http/multipart/MultipartEntity.html.

[51] wikipedia.org, "Cross-site request forgery," [Online]. Available: https://en.wikipedia.org/wiki/Cross-site_request_forgery.

[52] Google, "Google Maps," Google, [Online]. Available: https://www.google.lk/maps.

Appendix A

Individuals' contribution to the project

Name of the Student: B.R. Jayawickrama

Student Registration Number: 134073J

To implement the database for the SLTravelMate, the MySQL database management system was used. Prior to the implementation, the exact structure and relations between the entities were clarified with the guidance of the industry members. Then an entity relationship diagram was drafted. Since the use of MVC architecture to implement the web service, admin portal and the web application, making tables compatible with model classes of each web component was critical. The designer mode in XAMPP was efficient in modeling the database visually.

The basic requirement was to have a "users" table to record user data, and a separate table named "Items" to store all the travel based data records. This table includes contact and location information of different things that are required to a foreigner in Sri Lanka. Another table named "categories" was required to categorize the above said data records. Using the categories table, users can choose a specific category and view all data records related to it. Figure 5.2 of the Analysis and design chapter gives the Entity-Relationship diagram for further clarifications.

After implementing the required tables with relationships, the database was tested with the use of simple get and post methods to check whether data is read and written properly in the database. After several modifications, the important tables of the database were finalized.

To implement a role-based privilege system, more tables were added to group SLTravelMate's users and map each user group to a specific privilege. The records inside these specific tables are processed by the web service during authentication. This separates the privileges between system admins and normal users.

After implementing the required table structures, feeding actual data was the next step of implementation. Existing content were searched and categorized according to the database's requirement. Then they were entered as separate records into the specified database table. Relationship between "Items" table and "categories" table helped to achieve the task of viewing Items under a certain category. Sample users were added to test the posting functions of the web application.

Name of the Student : N.R. Weeraddana

Student Registration Number: 134185E

Web-service is my contribution to the sltravelmate project. As discussed in the previous sections it was decided to use Laravel PHP framework to create the web-service. This is the back end of the system which has direct access to the database. Mobile Application, Web Application and Admin Portal get authenticate in the web-service in order to access the database. All the users of the system are using a single authentication system although they are made to enter authentication details through different login views of Mobile Application, Web Application, and Admin Portal.

According to the logged in user, the authorization levels are changing. To implement that Zizaco/entrust: Role-based Permissions API was used. According to the implementation a Role can have one or more Permissions. Each system user can be assigned to one or more roles. When a user send an authentication request to the web-service, if the credentials matched, the web-service will send a json response including list of roles and list of permissions to the authenticated user.

Dingo API was used for handling of Exceptions and Errors.

There are two ways of implementing the authentication in a system, i.e., session based authentication and token based authentication. For the systems where client side application and server side applications are not separated, a session based authentication could be implemented. The sessions are stored in the web server, or in the database based on the recent activity of the authenticated users. For the systems where client side applications and server side application are separated, sessions cannot be maintained to detect whether a user is authenticated or not. For systems where client side application and server side applications are separated, a token based authentication is used. When the user sends a login request, server sends a valid token as the response if authentication was successful. Client application should send that token to the web-service each time it sends a Http request. JWTAuth API is used to handle the token based.

The routes of the web-service were filtered by filters that declared in Laravel project. There security could be enhanced thoroughly considering the above mentioned filters.

Name of Student: I.U. Kothalawala

Student Registration Number: 134217G

After the group leader distributed the work, I was given the responsibility of the android application. Since it was a new technology for me, first I watched a series of tutorials to get familiar with the terms and the functions. Then I made a list of the functions expected from mobile app and prioritized them considering the importance and the difficulties of implementing them. At first I thought it is better to learn android from scratch and then implement everything by myself. But after sometime I realized that with the limited time that is not possible and the best way to learn android is through the experience while developing the application.

Therefore I started developing the app. Every time I come across a question I googled it and there were blogs and sample applications where I could find the answers. Most of the codes could not be directly used and some of them also contained false information. So I had to first understand them, and gain what was required. While studying them I learnt about the different classes available in android and how they are used. Android Developers site was a great resource for me, because it contained all the information required for an android developer. For an example the unique classes in android, different methods and variables in them, guidelines to integrate services like Google Maps etc… So whenever I had a problem about a class or a method, I could find a solution on the Android Developers website. Videos on Youtube were also very helpful in developing the mobile application.

There were several problems I had to face while developing the android application. The major problem was, it took considerable amount of time to build the app and show the output. When I searched for a solution many websites had suggested using a real device instead of the emulator. That method was efficient but sometimes it also took about 10mins to show the output depending on the application. The reason was the RAM size of my laptop was not the recommended RAM size for android development which is 8GB. Another issue I faced was, since the modules depended on each other, some of the functions could not be implemented till the problems of those modules were solved. However the project was successfully completed with great effort of all the group members.

Although at first, learning a new technology like android development seemed to be tiresome and difficult, at the end it gave a great self satisfaction. Since this android application covered no of functionalities, I could learn many things within a small time period. I also got the opportunity to apply the knowledge I gained from different course modules. (ex: designing and maintaining the SQLite database) and the practice of learning new technologies through self studying. Finally as a result of great team work we could complete the project successfully. But there are certain areas that can be improved. With further development hope this would be a fruitful product in future.

Name of the Student: H.K.E.P. Hettiarachchi

Student Registration Number: 134061V

My contributions of this project are User Interface designing web application and the back end development of the web application. Web application of this system have been implemented by using the Laravel php frame work because in Laravel it was very easy to realize the MVC architecture. As a result of separating models, views and controllers of the system into three separate parts by using MVC architecture it was easy to implement these three part separately and edit one part without effecting to the whole system. As Laravel is having a well formatted and detailed documentations, tutorials and videos it was very easy to get familiar with Laravel frame work.

In this project, the backend of the web application has been developed by using the PHP 5.2. Back end mean connecting the web application and the web service in order to get the data from the server. Web application haven't the capability to access server directly, it need to be done through the web service when a request is sent to the web service by asking some data it send a json object of requested data as the respond that mean between the web application and web service the communication will happen using a light weight data interchange format called json but in the web application it is not possible to use a json object as it is then it need to be decode and the decoding will output an array of data this array is passed to the view. When sending the data from the web application to the web service in order to store in the server, web application have to create a json object and pass it to the web service. That mean there are two process that should happen in the backend of web application they are getting the data from the server and passing the data to the server through the web service in mvc architecture these two processes are happened by using the user defined functions in the controller. Therefore I had to identify the functions that need to be defined in the controllers.

For the development of front end web application basically html has been used and by using css, javascript and bootstrap it was possible to enhance the attractiveness and user-friendliness. As mentioned above after decoding a json object it is passing to the view as an array then inside the views by using a foreach loop it will iterate this array and the content in the array will be displayed. But there may be some occasions that need to display only some selected elements in the array, for that kind of occasions conditional statements like if , else has been used to control the flow of the iteration. As well as always the content of the web application is being displayed to the user by checking whether the user has been login to the system. Some features like uploading photos, commenting to an item, editing a comment and sending messages are available only for online users. When displaying that kind of content in the view authentication has to be done to check whether the user is a guest or not. To enable the features like uploading videos and images, enable login with face books and google plus, it was necessary to use external plugins tinyMCE. So that under the front end development it was my responsibility of creating the views of the web application by considering all the above facts.

Name of the Student: H.M.T.K.Madhusanka

Student Registration Number: 134093U

For this project, my contribution is to develop the Dashboard/Admin Portal which is the Administration area. The most critical parts of this is to giving Privileges according the User Role and Messaging. Here, Dashboard is used by 6 users. Divided this in to two Parts, UserInterface and Backend. To develop UI, I used HTML 5, CSS, Bootstrap and JScript, Jquery and for the Backend I used Laravel Php Framework. And there is no direct Db access, it retrieves Messaging Function handled between Users and the Administration. That is an Unique feature in this. And all Customizations, except Design Change, are done by here.

Appendix B

Code Segments from the project

Code Segment_B. 1: Source code from Android Manifest file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.dell.sltravelmate" >

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS"
/>
    <uses-permission
android:name="android.permission.USE_CREDENTIALS" />
    <uses-permission android:name="android.permission.RECORD_AUDIO"
/>

    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <permission

android:name="com.example.dell.sltravelmate.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />

    <uses-permission
android:name="com.example.dell.sltravelmate.permission.C2D_MESSAGE"
/>

    <application
        android:name=".AppController"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

```
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLay
out|uiMode|screenSize|smallestScreenSize"
            android:theme="@android:style/Theme.Translucent" />
        <activity
            android:name=".SecondActivity"
            android:label="@string/title_activity_secondactivity"
            android:launchMode="singleTop" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.MainActivity" />
        </activity>
        <activity
            android:name=".Settings"
            android:label="@string/title_activity_settings" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.MainActivity" />
        </activity>
        <activity
            android:name=".GuidesActivity"
            android:label="Guides" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.MainActivity" />
        </activity>
        <activity
            android:name=".GuidesDetails"
            android:label="Sanka Wijethunga" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.GuidesActivity" />
        </activity>
```

```
        <activity
            android:name=".SendMessage"
            android:label="@string/title_activity_send_message" />

        <activity
            android:name=".CategoryItemDetails"

android:label="@string/title_activity_category_item_details"
            android:launchMode="singleTop" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.SecondActivity" />
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyC7eWnf9_FwBmNGwlFF_NiSOjCT4ACwdEs"
/>

        <activity
            android:name=".Images"
            android:label="@string/title_activity_images" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.CategoryItemDetails" />
        </activity>
        <activity
            android:name=".Videos"
            android:label="@string/title_activity_videos" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.CategoryItemDetails" />
        </activity>
        <activity
            android:name=".SearchResultsActivity"
            android:label="@string/title_activity_search_results" />
        <activity
            android:name=".Login"
            android:label="Sign In"
            android:windowSoftInputMode="stateHidden" >
        </activity>
        <activity
            android:name=".Register"
```

```
            android:label="Create account"
            android:launchMode="singleTop"
            android:windowSoftInputMode="stateHidden" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.dell.sltravelmate.Login"
/>
        </activity>
        <activity
            android:name=".Messages"
            android:label="@string/title_activity_messages" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.MainActivity" />
        </activity>
        <activity
            android:name=".My_profile"
            android:label="@string/title_activity_my_profile" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"

android:value="com.example.dell.sltravelmate.MainActivity" />
        </activity>
        <activity
            android:name=".ShowComments"
            android:label="@string/title_activity_show_comments" >
        </activity>
        <activity
            android:name=".WriteComments"
            android:label="Write a comment" >
        </activity>
        <activity
            android:name=".UploadActivity"
            android:label="@string/title_activity_upload" >
        </activity>
        <activity
            android:name=".ChangeUserDetails"
            android:label="Change Password" >
        </activity>
        <activity
            android:name=".ChangePhonenumber"
            android:label="Change Phone Number" >
        </activity>
        <activity
            android:name=".SponsorsActivity"
            android:label="@string/title_activity_sponsors" >
        </activity>
        <activity
            android:name=".SponsorsDetails"
```

```
            android:label="Our Sponsors" >
        </activity>
        <activity
            android:name=".ShowMessages"
            android:label="@string/title_activity_show_messages" >
        </activity>

        <receiver
            android:name="com.google.android.gms.gcm.GcmReceiver"
            android:exported="true"

android:permission="com.google.android.c2dm.permission.SEND" >
            <intent-filter>
                <action
android:name="com.google.android.c2dm.intent.RECEIVE" />

                <category
android:name="com.example.dell.sltravelmate" />
            </intent-filter>
        </receiver>

        <service
            android:name=".GCMPushRecieverService"
            android:exported="false" >
            <intent-filter>
                <action
android:name="com.google.android.c2dm.intent.RECEIVE" />
            </intent-filter>
        </service>
        <service
            android:name=".GCMRegistrationIntentService"
            android:exported="false" >
            <intent-filter>
                <action
android:name="com.google.android.gms.iid.InstanceID" />
            </intent-filter>
        </service>

    </application>

</manifest>
```

Code Segment_B. 2: Sample code from the web application

```
View a list of category
Example: View a list of hotels
Router:
Route::get('category/{id}',['as' => 'category', 'uses' =>
'CategoryController@category']);
View:
@foreach($hotel as $hotelname):
<li class=”im-g”><a href=”#”><img
src=”{{$hotelname[‘thumb_image_in_list’]}}”  class=”img-responsive”
alt=””></a></li>
<li class=”data”>
<h4>{{$hotelname[‘item_name’]}}</h4>
<p>{{$hotelname[‘description’]}}</p>
<li class=”bt-nn”>
<a class=”hvr-shutter-in-horizontal button”
href=”{{URL::route(‘categorylist’,[$hotelname[‘item_ID’]])}}”>View
Page</a>
</li>
<div class=”clearfix”></div>
</div>
@endforeach;

Controller:
public function category($id)
    {

$url='http://localhost:8082/WebService/public/item/getitemswithslide
showitems/'.$id;
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_TIMEOUT, 3);
$response = curl_exec($ch);
curl_close($ch);
$hotel = json_decode($response,true);

        if($id == 1)
        return view('categorylist', ['title' => 'Accomadation'] )-
>with(compact('hotel'));
        if($id == 2)
        return view('categorylist', ['title' => 'Dining'] )-
>with(compact('hotel'));
        if($id == 3)

        return view('categorylist', ['title' => 'Health Centers'] )-
>with(compact('hotel'));
```

```
        if($id == 4)
        return view('categorylist', ['title' => 'Shopping Centers']
)->with(compact('hotel'));
         if($id == 5)
        return view('categorylist', ['title' => 'Police Stations']
)->with(compact('hotel'));
         if($id == 6)
        return view('categorylist', ['title' => 'Banks'] )-
>with(compact('hotel'));
         if($id == 7)
        return view('categorylist', ['title' => 'Historical Places']
)->with(compact('hotel'));
         if($id == 8)
        return view('categorylist', ['title' => 'Forest'] )-
>with(compact('hotel'));
         if($id == 9)
        return view('categorylist', ['title' => 'Beaches'] )-
>with(compact('hotel'));
         if($id == 10)
        return view('categorylist', ['title' => 'Guides'] )-
>with(compact('hotel'));
         if($id == 11)
        return view('categorylist', ['title' => 'Activities'] )-
>with(compact('hotel'));
         if($id == 12)
        return view('categorylist', ['title' => 'Cab Services'] )-
>with(compact('hotel'));
        }
}



View page of a specific item
Router
Route::get('categorylist/{id}',['as' => 'categorylist', 'uses' =>
'ItemController@home']);

View:
Google map integration
<script type="text/javascript">
  function initialize() {
    var position = new google.maps.LatLng($hotelname['longitude'],
$hotelname['latitude']);
    var myOptions = {
      zoom: 10,
      center: position,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
```

```
        myOptions);

    var marker = new google.maps.Marker({
        position: position,
        map: map,
        title:"This is the place."
    });

    var contentString = $hotelname[‘item_name’];
    var infowindow = new google.maps.InfoWindow({
        content: contentString
    });

    google.maps.event.addListener(marker, 'click', function() {
      infowindow.open(map,marker);
    });
 }
</script>

Displaying the details of a particular item
<div><h1><center>{{$hotelname['item_name']}}</center></h1></div>
<div>{{$hotelname['description']}}</div>
<div>Address :{{$hotelname['address']}}</div>
<div>Tel no  :{{$hotelname['telno']}}</div>
<div>fax     :{{$hotelname['fax']}}</div>
<div>email   :{{$hotelname['email']}}</div>

Controller:
public function index($x)
{
$posts = Posts::where('active','1')->where('item_ID',$x)-
>orderBy('created_at','desc')->paginate(5);
            $title = 'Latest Posts';
$url='http://localhost:8082/WebService/public/item/getallitems';

        $ch = curl_init();

        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

        curl_setopt($ch, CURLOPT_TIMEOUT, 3);

        $response = curl_exec($ch);

        curl_close($ch);

        $hotel = json_decode($response,true);
        foreach ($hotel as $hotel) {
            if($hotel['item_ID']==$x)
            {
```

```
                    return view('Itempage')->withPosts($posts)-
>withTitle($title)->withHotelname($hotel);
                }
            }
}

Upload an image or post
Router:
Route::get('new-
post/{item_id}',['as'=>'createpost','uses'=>'PostController@create']
);
View:
<script type="text/javascript" src="{{
asset('/js/tinymce/tinymce.min.js') }}"></script>
<script type="text/javascript">
    tinymce.init({
            selector : "textarea",
            plugins : ["advlist autolink lists link image charmap
print preview anchor", "searchreplace visualblocks code fullscreen",
"insertdatetime media table contextmenu paste jbimages"],
            toolbar : "insertfile undo redo | styleselect | bold
italic | alignleft aligncenter alignright alignjustify | bullist
numlist outdent indent | link image jbimages",
    });
</script>

<form action="{{URL::route('storepost',[$item_id])}}" method="post">
    <input type="hidden" name="_token" value="{{ csrf_token() }}">
    <div class="form-group">
            <input required="required" value="{{ old('title') }}"
placeholder="Enter title here" type="text" name =
"title"class="form-control" />
    </div>
    <div class="form-group">
            <textarea name='body'class="form-control">{{ old('body')
}}</textarea>
    </div>
    <input type="submit" name='publish' class="btn btn-success"
value = "Publish"/>
    <input type="submit" name='save' class="btn btn-default" value
= "Save Draft" />
</form>

Controller
public function create(Request $request,$item_id)
{
  if($request->user()->can_post())
{
return view('posts.create',['item_id'=>$item_id]);
}
```

```
else
{
return redirect('/')->withErrors('You have not sufficient
permissions for writing post');
}
}
```

```
Displaying Posts
Router:
Route::get('uploads',['as' => 'home', 'uses' =>
'ItemController@home']);
Controller:
public function home()
    {
     $url='http://localhost:8082/WebService/public/post/getpost';
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL, $url);
       curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

       curl_setopt($ch, CURLOPT_TIMEOUT, 3);

       $response = curl_exec($ch);

       curl_close($ch);

       $Posts = json_decode($response,true);



$url='http://localhost:8082/WebService/public/category/getcategories
';

       $ch = curl_init();

       curl_setopt($ch, CURLOPT_URL, $url);
       curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

       curl_setopt($ch, CURLOPT_TIMEOUT, 3);

       $response = curl_exec($ch);

       curl_close($ch);

       $hotel = json_decode($response,true);
foreach ($hotel as $hotel) {
           if($hotel['item_ID']==$x)
```

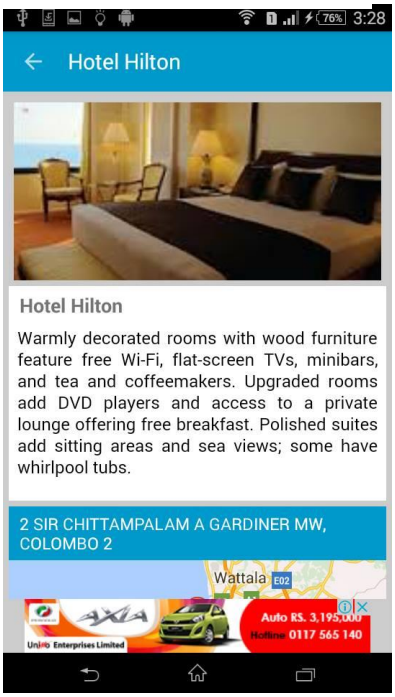
```
            {
                  return view('home')->withPosts($posts)-
>withTitle($title)->withHotelname($hotel);
                }
    }

@foreach( $posts as $post )
<div class="list-group">
<div class="list-group-item">
<h3><{{ $post->title }}</a>
@if(!Auth::guest() && ($post->author_id == Auth::user()->id ||
Auth::user()->is_admin()))
    @if($post->active == '1')
               <button class="btn" style="float: right"><a href="{{
url('edit/'.$post->slug)}}">Edit Post</a></button>
             @else
            <button class="btn" style="float: right"><a href="{{
url('edit/'.$post->slug)}}">Edit Draft</a></button>
    @endif
    @endif
</h3>
<p>{{ $post->created_at->format('M d,Y \a\t h:i a') }} By <a
href="{{ url('/user/'.$post->author_id)}}">{{ $post->author->name
}}</a></p>
</div>
<div class="list-group-item">
</div>
</div>
```

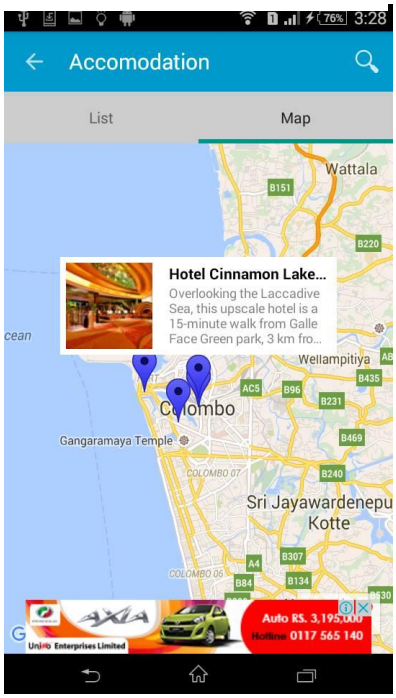
Screenshots from the Android Application

Figure_B. 1: Category list of SLTravelMate

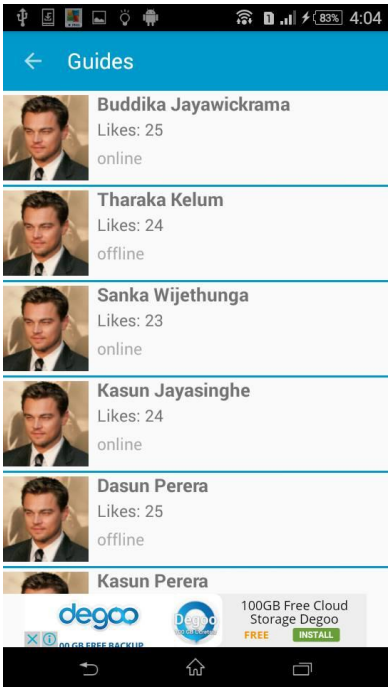
Figure_B. 2: Items under the "Accomodations category"



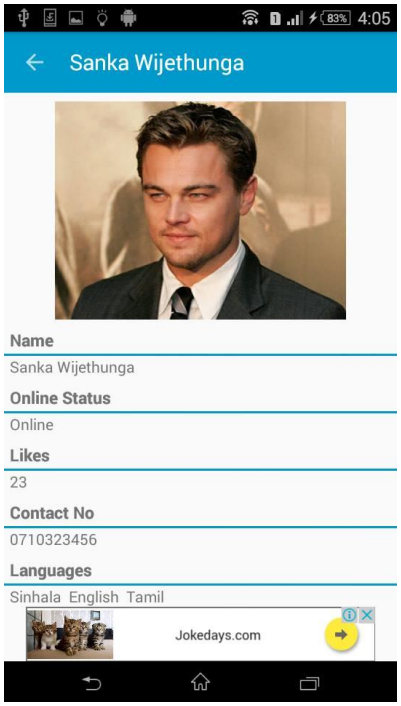
Figure_B. 3: Detailed view of a Hotel Item



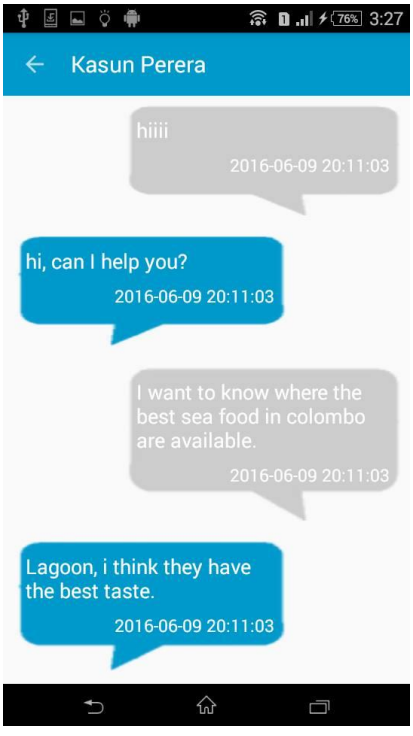
Figure_B. 4: Map view with locations



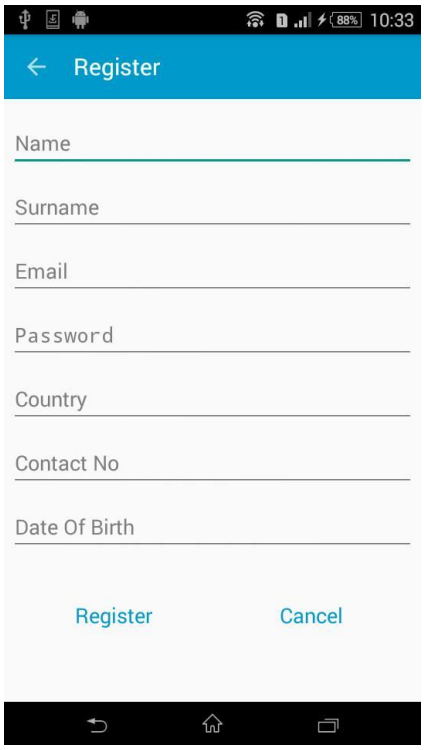
Figure_B. 5: List of Travel Guides



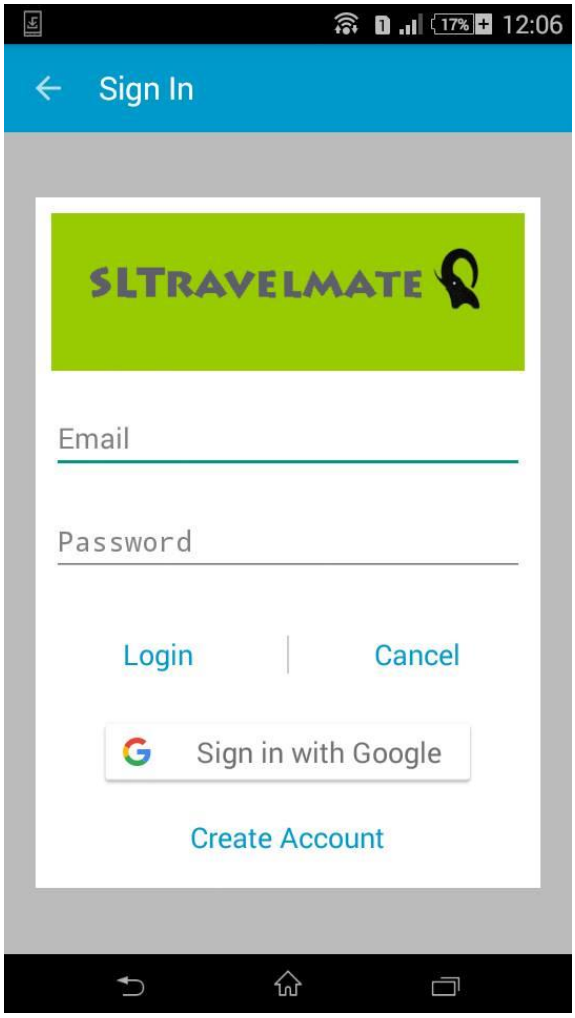
Figure_B. 6: Detailed view of a travel guide



Figure_B. 7: Messaging between a guide and a traveler



Figure_B. 8: Registration area for travelers



Figure_B. 9: Login Area for travelers