

VOYAGER TRAVEL GUIDE APP

A project report submitted by

NIRAJ C S(GTATSCS030)

Under the guidance of

DR.JEEVAMOL JOY

Assistant Professor

Department of Computer Science

Sri. C Achutha Menon Government College Kuttanellur

For the award of the Degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

(Calicut University)

2019-2022

Submitted to



Department of Computer Science

Sri. C. Achutha Menon Government College, Thrissur

Kuttanellur, Kerala



CERTIFICATE

This is to certify that the project titled “VOYAGER”, submitted by NIRAJ C S(GTATSCS030) is a bonafide record of project work done at the Department of Computer Science, Sri C Achutha Menon Government College, Kuttanellur in partial fulfilment of the requirement of Bachelor of Computer Science of the University of Calicut.

Submitted for the Viva-Voce Examination held on

Internal Examiner

External Examiner

Head of the Department

(Dept Seal)

DECLARATION

We wish to state that the work embodied in this project titled VOYAGER GUIDE TRAVEL APP forms our own contribution to the project work carried out under the guidance of DR.JEEVAMOL JOY, Assistant Professor, Department of Computer Science, Sri. C Achutha Menon Government College Kuttanellur . We hereby declare that this work is submitted to University of Calicut in partial fulfilment of the requirement for the award of the degree of Bachelor of Computer Science and this submission contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

E Nithin Neelakandan (GTATSCS027)

Hazeeb(GTATSCS011)

Niraj C S(GTATSCS030)

Sreelakshmi M P(GTATSCS004)

Place: Kuttanellur

Date:

Contents

Title		Page No
Voyager Travel Guide App.		
1	Introduction	10
1.1	Features of Existing Systems	10
1.2	Limitations of Existing Systems	10
1.3	Area and Category of the Project Work	10
2	Problem Definition and Methodology	11
2.1	Problem Definition	11
2.2	Motivation	11
2.3	Objectives	11
2.4	Methodology	12
2.5	Scope	13
3	Analysis	13
3.1	Requirement Analysis	13
3.2	Existing System	14
3.3	Proposed System	14
3.4	Requirement Specification	14
3.4.1	Functional Requirements	15
3.4.2	Non-functional Requirements	15
3.4.3	Hardware Requirements	17
3.4.4	Software Requirements	17
3.5	Feasibility Study	21
3.5.1	Technical feasibility	21
3.5.2	Economic feasibility	21
3.5.3	Operational feasibility	22
4	Design	23
4.1	Introduction	23
4.2	Modularity Criteria	23
4.3	Architecture Diagrams/DFD	23
4.3.1	DFD LEVEL 0	25

4.3.2		DFD LEVEL 1	26
4.3.3		DFD LEVEL 2	27

4.4		User Interface Layout	27
4.4.1		Registration page	28
4.4.2		Login page	29
4.4.3		Home page	29
4.4.4		Booking page	30
4.4.5		Assistance page	30
4.4.6		Guidance page	31
4.5		Structure of reports being created	31
4.6		Database design	32
4.6.1		List of entities and Attributes	35
4.6.2		E R Diagram	36
4.6.2		Structure of Tables	38
4.6.3.1		Registration table	38
5		Implementation	39
5.1		Introduction	39
5.2		Tools/Scripts/Platforms for Implementation	39
5.3		Process Logic of each module	39
5.4		Coding	39
5.5		Screen Shots	86
6		Testing	88
6.1		Introduction	88
6.2		Testing Methodologies Adopted	89
6.2.1		Unit Testing	89
6.2.2		Integration Testing	89
6.2.3		System Testing	90
6.3		Test Plan and Test cases	90
6.3.1		Login Screen	91

6.3.2	Registration Screen	92
6.3.3	Home Screen	92
7	Conclusion	95
	References	96

List of Figures

Section No	Caption	Page No
4.3.1	Level 0 DFD	26
4.3.2	Level 1 DFD	26
4.3.3	Level 2 DFD	27
4.4.1	Registration Page Layout	28
4.4.2	Login Page Layout	29
4.4.3	Home Page Layout	29
4.4.4	Booking Page Layout	30
4.4.5	Assistance Page Layout	31
4.4.6	Guidance page Layout	31
4.6.2	Entity-Relationship Diagram	36

List of Tables

[illegible]

VOYAGER

Chapter 1

Introduction

Travelling is one of the best things that each of us does to refresh ourselves. Most people love to enjoy new sceneries and meet new people. They like to learn new cultures and explore new things. Hence travelling is a great journey for a person to enhance his/her life experience, and that's why we are here to guide everyone in choosing their tourist spots according to the user's interest.

1.1 Features of Existing Systems

1. There are applications to provide information about places.
2. There are also applications to provide bookings for user's.
3. There are applications that serve as guides to user's.
4. There are applications that provide medical and vehicle assistance to users.

1.2 Limitations of Existing Systems

1. There exists no applications which provides all these features together in a single app

1.3 Area and Category of the Project Work

The project is intended for travellers. Anyone can login and use this application.

The project is categorised as an android app and can be used by anyone with access to the internet and a smartphone.

Chapter 2

Problem Definition & Methodology

2.1 Problem Definition

This project provides the tourist with needful information depending on their current location given by the android user. This information helps the user to explore the nearby places . It also includes details about each place , the way to reach these places and emergency details like hospital and vehicle assistance. This project is also useful for users who have no idea about the places they want to visit.

Our main idea is to group together all the things required for a tour inside a single app. This app is initially built for a small scale. But with final updations people could completely rely on our app for all their tours.

2.2 Motivation

At earlier times travellers explored new places by seeking out information from the local people of that area. Booking facilities were not easy as of nowadays and people faced a lot of difficulties in order to find an apartment to take rest. Even in order to take a vehicle or medical assistance people had to completely rely on local people. With the advancements in technology today all such difficulties are removed and there are a wide range of applications for navigation, bookings, assistance and online shops. But there isn't an application that provides all these features in a single app.

2.3 Objectives

Our objective is to bring together all the features that a smartphone can provide to assist a tourist in his/her journey. At the initial stage we aim to provide information about nearby spots, a booking section , assistance section for medical and vehicle assistance, and a guidance section to seek guidance from efficient people.

2.4 Methodology

The system can start learning from scratch using frequency of visits to various destinations from a user's source. The most frequent destinations can be sorted by it and displayed to a new traveller and suggest to him/her to travel to the most frequently visited destination. Slowly this system can match the preferences of a user visiting a certain place who also visits another place. Thus, it will need a machine learning algorithm for training the system into two classes: similar and not similar destinations. It can count the number of persons visiting the same destination for any two such places and normalised by total visitors from a source to these two places. Then this count will have a value between 0 and 1. 1 if both places are visited by the same visitors. 0 if no two places are visited by the same people. Based on this similarity score system will predict the user preferred routes from his/her travel history.

II The system can also predict the difficulties in any route from user's feedback and predict a route as difficult or easy from these feedback scores (1-5) given by visitors for these places. The score is weighted averaged over all the visitors and level of difficulty is known from these values. Users can be warned of difficult routes using these scores. The system can keep the previous score and number of visitors and update both values when the next visitor travels there to get a new score.

The system can also be trained based on textual feedback comments on a route from travellers. If a difficult route is known, then its comments can be stored in a document which relates to the difficult journey. In a smooth and nice way comments can be stored as a good journey. Then logistic regression can be applied to train the system with input nodes equal to the number of words possible in a sentence, say 100. For each word tf-idf can be found and given to the Neural Network as input and output defined for which document class is difficult and which is good. When we give a new place travellers feedback then it will automatically put it into a difficult or easy route. Like this user can be displayed difficulty level in any chosen route.

The difficulty of any route can be trained using input parameters like:

1. Road conditions
2. Weather conditions
3. Traffic
4. Number of accidents
5. Personality of an individual (e.g. age, gender of person)

2.5 Scope

This app is built with simple designs and features which makes it more user friendly and easy to use. With the features provided, a user can completely rely on our app for almost all types of assistance in his journey and he/she doesn't have to install other apps or websites for it.

Our app provides best booking sites, packages, guides, medical assistance, vehicle assistance and notifications / alerts to the users. As further updates we are planning to add a navigation map and also an e-commerce website to purchase travelling equipment. As the size of the system is very less compared to other existing systems it will be effectively working on low end devices smoothly.

Chapter 3

Analysis

3.1 Requirement Analysis

Requirements analysis focuses on the tasks that determine the needs or conditions to meet the new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analysing, documenting, validating and managing software or system requirements. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

3.2 Existing System

The existing system provides all the features but in different apps. The user has to install different apps depending on his/her requirements.

3.3 Proposed System

Our project combines all the features to assist a tourist for his/her journey in a single android application. We provide:-

1. A booking section to book hotels, flights or to choose any travelling packages.
2. An assistance section to provide medical and vehicle assistance in case of emergency.
3. A preference section to get the interest of the user.
4. A guide section to provide guidance from efficient people to users and it also provides a facility for other people to join as guides in our app..

3.4 Requirement Specification

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure. The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have a clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

3.4.1 Functional Requirements

Implementation is the stages of a project when the theoretical design is turned into a working system. If the implementation stage is not properly planned and controlled, it can cause chaos. Thus, it can be considered to be the most crucial stage in achieving a successful new system and in giving the users confidence that the new system will work and be effective.

Normally this stage involves setting up a co-ordination committee, which will act as a sounding board of ideas, complaints and problems. The first task is implementation planning ie; decision of the methods and time scale to be adopted. Apart from planning, the two major tasks of preparing for implementation are education and training of administrators and testing of the system.

After the implementation phase is completed and the user staff adjusted to the changes created by the candidate system, evaluation and maintenance is continued to bring the new system standards.

The activities of the implementation phase can be summarised as;

- Implementation planning
- Education and training
- System training
- System implementation is the final stage that put the utility into action. Implementation is the state in the project where the theoretical design turned into a working system.

3.4.2 Non-functional Requirements

In software engineering, a functional requirement defines a system or its components. It describes the functions that software must perform. Function is nothing but inputs, its behaviour, and output. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirement helps you to capture the intended behaviour of the system. This behaviour may be expressed as functions, services, or tasks which system is required to perform.

SL.NO	TYPES OF REQUIREMENTS	DETAILS
1	Performance requirements	The response time should not vary with the increasing size of the data storage.
2	Security requirements	This application should not modify any pictures. The username and password should be a unique token id.
3	Training requirements	Training has to be provided by the supervisor.

Performance

System performance is the most important quality in non-functional requirements and affects almost all the other preceding ones. Furthermore, reliability, availability, and maintainability (RAM) features fall exclusively under these requirements. System performance defines how fast a system can respond to a particular user's action under a certain workload.

Reliability

Reliability is the probability and percentage of the software performing without failure for a specific number of uses or amount of time.

Availability

This feature defines the amount of time the system is running, the time it takes to repair a fault, and the time between lapses.

Security

Security measures ensure your software's safety against espionage or sabotage. These features are necessary even for stand-alone systems; you don't want anyone to have access to your sensitive data.

Maintainability

This feature indicates the average time and ease and rapidity with which a system can be restored after a failure.

Portability

Portability in high-level computer programming is the usability of the same software in different environments. The Pre requirement for portability is the generalised abstraction between the application logic and system interfaces. When software with the same functionality is produced for several computing platforms, portability is the key issue for development cost reduction.

3.4.3 Hardware Requirements

Operating system:Android 5.1.1+

Available Storage:2GB

RAM:2GB+

CPU:Snapdragon 400 series or above

GPU:Adreno 500 series or above

3.4.4 Software Requirements

Operating System: Windows 7,8,10

Platform:Android Studio

Front-end:Android,Java

Back-end:PHP,MySQL

Server:C panel

OPERATING SYSTEM

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. The operating system used in this project is windows OS

ANDROID (FRONT END)

Android is a mobile operating system developed by Google. It is based on a modified version of the Linux kernel and other open source software, and is designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialised user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics. One of the most widely used mobile OS these days is ANDROID. Android is a software bunch comprising not only operating system but also middleware and key applications. Google's Android division certainly has a sense of humour: It named all of its version code names after desserts

FEATURES OF ANDROID

- **Near Field Communication (NFC):** Most Android devices support NFC, which allows electronic devices to easily interact across short distances. The main aim here is to create a payment option that is simpler than carrying credit cards or cash, and while the market

hasn't exploded as many experts had predicted, there may be an alternative in the works, in the form of Bluetooth Low Energy (BLE).

- **Infrared Transmission:** The Android operating system supports a built-in infrared transmitter, allowing you to use your phone or tablet as a remote control.
- **Automation:** The Tasker app lets you not only control app permissions but also automate them.
- **Alternate Keyboards:** Android supports multiple keyboards and makes them easy to install; the SwiftKey, Skype, and 8pen apps all offer ways to quickly change up your keyboard style. Other mobile operating systems either don't permit extra keyboards at all, or the process to install and use them are tedious and time-consuming.
- **Storage and Battery Swap:** Android phones also have unique hardware capabilities. Google's OS makes it possible to remove and upgrade your battery or to replace one that no longer holds a charge. In addition, Android phones come with SD card slots for expandable storage.
- **Custom Home Screens:** While it's possible to hack certain phones to customise the home screen, Android comes with this capability from the get-go. Download a third-party launcher like Nova, Apex or Slide and you can add gestures, new shortcuts, or even performance enhancements for older-model devices.

JAVA (FRONT END)

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs. Java is one of the world's most important and widely used computer languages, and it has held this distinction for many years. Object Oriented meaning the capability to reuse code. It is possible to develop a single application which can run on multiple platforms like Windows, UNIX, and Macintosh systems. Java does not support the use of pointers. It automatically manages memory garbage-collection routines that are activated when the system runs short of memory. Java provides the most secure programming environment. Java doesn't just fix security loopholes-it eliminates them, which makes Java the perfect language for programming on the Web.

FEATURES OF JAVA

- **Simple** :Java is an extension of C and C++ with the added feature of garbage collection and improved memory management.
- **Object oriented** :Object-oriented programming deals with objects and their behaviours and hence an analogy of the real world can be found in programs.
- **Network** :Java has an extensive library of routines for coping easily with TCP/IP protocols like HTTP and FTP. This makes creating network connections much easier.
- **Robust** :Java is intended for writing programs that must be reliable in a variety of ways. Java puts a lot of emphasis on early checking for possible problems, later dynamic (runtime) checking, and eliminating situations that are error-prone.
- **Secure** :Java is intended for use in networked/distributed environments. Toward that end, a lot of emphasis has been placed on security. The changes to the semantics of pointers make it impossible for applications to forge access to the user's hard disk.
- **Portable** :There are no "implementation dependent" (machine/ processor dependent) aspects of the specification. The sizes of the primitive data types (integer, float) are specified

PHP (BACK END)

PHP was at first created as a simple scripting platform called "Personal Home Page". Nowadays PHP is an alternative of Microsoft's Active Server Pages (ASP) technology. PHP is an open source server-side language which is used for creating dynamic web pages. It can be embedded into HTML. PHP is usually used in conjunction with a MySQL database on Linux/UNIX web servers. It is probably the most popular scripting language.

PHP is a widely-used general-purpose scripting language and interpreter that is freely available. A full explanation of all the PHP functions, complete user manual and lots of tutorials can be found on the PHP's official page. PHP code may be executed with a command line interface (CLI), embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks.

FEATURES OF PHP

- **Interpreted** :It is an interpreted language, i.e. there is no need for compilation.
- **Faster** :It is faster than other scripting languages e.g. asp and jsp.
- **Open Source** :Open source means you no need to pay to use php, you can freely download and use.
- **Platform Independent** :PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.
- **Case Sensitive** :PHP is case sensitive scripting language at time of variable declaration. In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.
- **Error Reporting** :PHP has some predefined error reporting constants to generate a warning or error notice.
- **Real-Time Access Monitoring** :PHP provides access logging by creating the summary of recent accesses for the user.

MYSQL (DATABASE)

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of “My”, the name of co-founders Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organises data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database.

3.5 Feasibility Study

Feasibility study is a process that identifies, describes and evaluates proposed systems and selects the best system. During the study, the problem definition is solved and all aspects of the problem to be included in the system are determined. Size of project, cost of benefits is also estimated with greater accuracy. A good feasibility study will show the strength and defects before the project is planned or budgeted.

It evaluates the project's potential success, because it rationally uncovers strengths and weaknesses of a proposed project.

3.5.1 Technical feasibility

It determines whether the technology needed for the proposed system is available and how it can be integrated with the government. Technical evaluation must assess whether the user has technical expertise to understand and use the new system. This assessment is based on an outline design of system requirements, to determine whether the proposal is technically and legally feasible. It is the evaluation of the hardware and software and how it meets the needs of the proposed system.

3.5.2 Economical feasibility

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organisation that the proposed system will provide. It includes quantification and identification. It identifies the financial benefits and costs associated with the development of the system. Economic feasibility is often known as the cost benefit analysis. To carry out an economic feasibility study it is necessary to estimate actual money value against activities needed for implementing the system.

While implementing our system we can ensure that the cost of prospective new ventures will ultimately be profitable to the people. So, we can say it is financially feasible.

The technology used can be developed with the current equipment and has the technical capacity to hold data required by the old system.

- This technology supports the modern trends of technology.
- Easily accessible, more secure technologies.

3.5.3 Operational feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the

requirements identified in the requirement analysis phase of system development. It focuses on the degree to which the proposed development project fits in with the existing environment and objectives it regards to development schedule. Operational feasibility focuses on human, organisational and political.

The proposed system can easily be implemented, as this is based on java coding and android studio. The database was created with the Xampp server which is more secure and easy to handle. The resources that are required to implement/install these are available. So the project is operationally feasible.

Chapter 4

Design

4.1 Introduction

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. System development is the process of creating or altering systems, along with the processes, practices, models, and methodologies used to develop them. The different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organisation through the engineering of a coherent and well-running system.

Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

4.2 Modularity Criteria

The concept of modularity is used primarily to reduce complexity by breaking a system into varying degrees of independence and independence across and hiding the complexity of each part behind an abstraction and interface. Effective modular design can be achieved if the partitioned modules are separately solvable, modifiable as well as compilable.

4.3 Architecture Diagrams/DFD

A data-flow diagram (DFD) is a way of representing the flow of data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented. There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DE Marco as part

of Structured Analysis. For each data flow, at least one of the endpoints must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. The data-flow diagram is part of the structured-analysis modelling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan. Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.

Terminologies used in DFD diagrams are:

Process

The process is part of a system that transforms inputs to outputs. The symbol of a process is a circle, an oval, a rectangle or a rectangle with rounded corners. The process is named in one word, a short sentence, or a phrase that is clearly to express its essence.

Data Store

Most information systems capture data for later use. The data is kept in a data store. It is represented by the open-ended box. At the left end is a small box used to number the data store and inside the main part of the rectangle is a meaningful label for the data store.

Data Flow

Data flow shows the transfer of information from one part of the system to another. The symbol of the flow is the arrow. The flow should have a name that determines what information (or what material) is being moved. Exceptions are flows where it is clear what information is transferred through the entities that are linked to these flows. Material shifts are modelled in systems that are not merely informative. Flow should only transmit one type of information (material). The arrow shows the flow direction. Flows link processes, warehouses and terminators.

Warehouse

The warehouse (data store, data store, file, and database) is used to store data for later use. The symbol of the store is two horizontal lines; the other way of view is shown in the DFD Notation. The name of the warehouse is a plural noun (e.g., orders) - it derives from the input and output streams of the warehouse. The warehouse does not have to be just a data file, for example, a folder with documents, a filing cabinet, and optical discs. Therefore, viewing the warehouse in DFD is independent of implementation. The flow from the warehouse usually represents the reading of the data stored in the warehouse, and the flow to the warehouse usually expresses data

entry or updating (sometimes also deleting data). Warehouse is represented by two parallel lines between which the memory name is located.

Source/Sink

A source/sink is the origin and/or destination of the data. Source/sinks are sometimes referred to as external entities because they are outside the system and they define boundaries of the system. Data must originate outside a system from one or more sources and the system must produce information to one or more sinks.

Terminator

The Terminator is an external entity that communicates with the system and stands outside of the system. It can be, for example, various organisations, groups of people (e.g., customers), authorities or a department (e.g., a human-resources department) of the same organisation, which does not belong to the model system. The terminator may be another system with which the modelled system communicates

Symbols used in DFD



Process



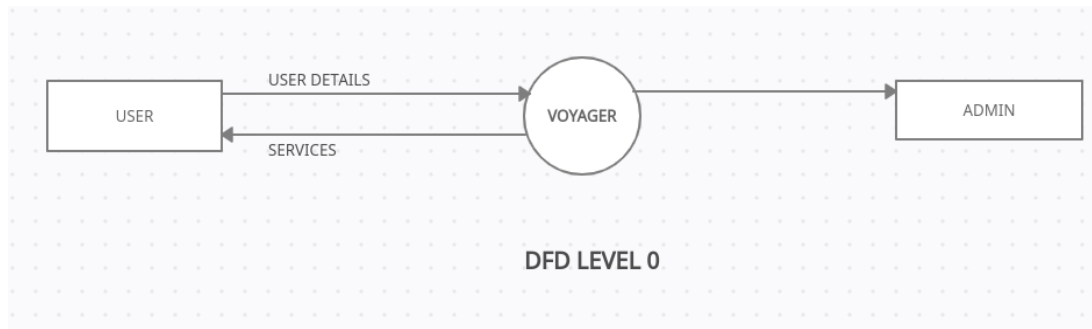
Data store



Data flow

4.3.1 DFD Level 0

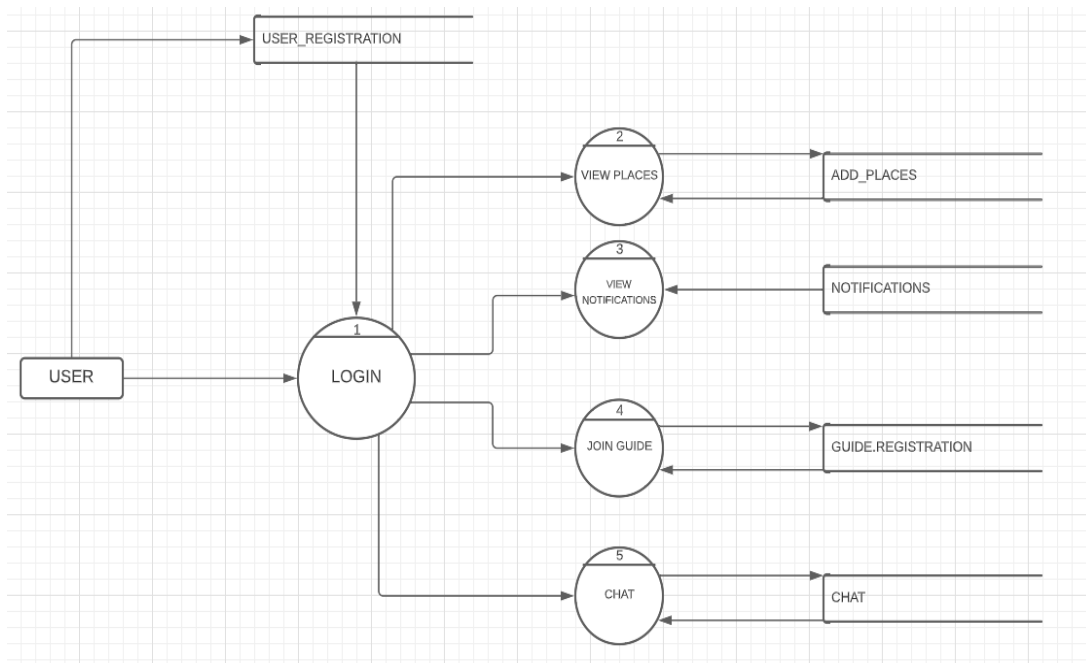
The Level 0 DFD represents the information flow through the system at the top most level.

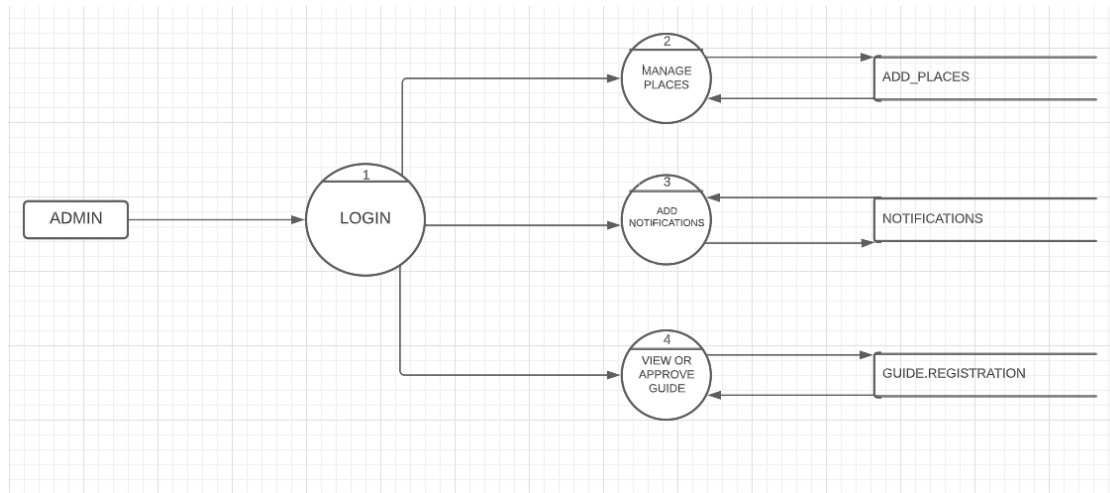


4.3.2 DFD Level 1

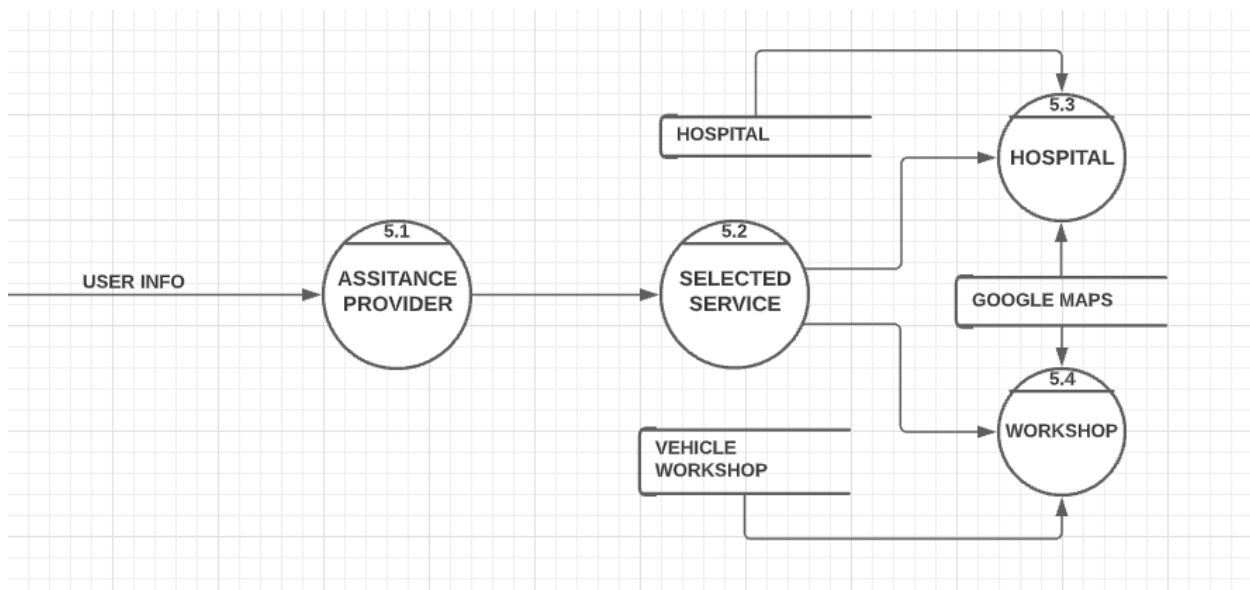
The Level 1 DFD defines four processes, two databases and two data entities(sources) and the data flow through the system in a more detailed structure.

DFD level 1 (user):



DFD level 1 (admin) :**4.3.3 DFD Level 2**

The DFD level 2 further describes the detailed flow of data through each module of the system.

Dfd level 2 (Assistance):**4.4 User Interface Layout**

User interface or UI describes the way in which a human interacts with a machine. It is the virtual part of a computer application or operating system through which a client interacts with a

computer or software. It determines how commands are given to the system or the program and how data is displayed on the screen.

There are mainly two types of user interfaces. They are:


- Text based User Interface or Command Line Interface
- Graphical User Interface (GUI).

An effective User Interface design must have following principles:

- Structure
- Simplicity
- Visibility
- Feedback
- Flexible
- Tolerance

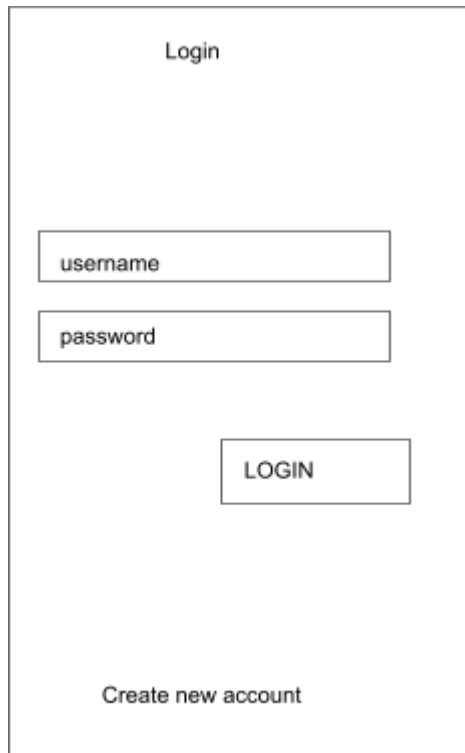
A layout defines the structure for a user interface in your application, such as in an activity. All elements in the layout are built using a hierarchy of view and view Group objects. A view usually draws something the user can see or interact with. Whereas a view group is an invisible container that defines the layout structure for view and other view group objects.

4.4.1 Registration page



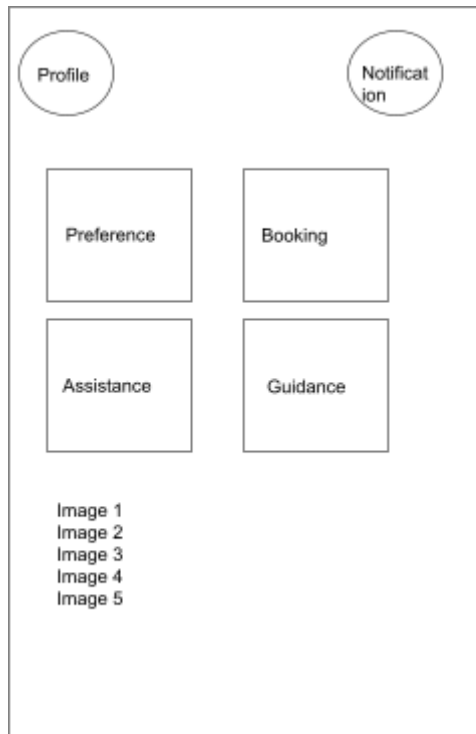
The diagram illustrates a registration form layout within a rectangular container. At the top center, the title "Registration" is displayed. Below the title, five text input fields are arranged vertically, each containing a placeholder label: "username", "password", "email", "phone", and "address". At the bottom center of the form, there is a rectangular button labeled "REGISTER".

4.4.2 Login page



A wireframe of a login page. At the top center is the text "Login". Below it are two rectangular input fields, the first labeled "username" and the second labeled "password". To the right of the password field is a rectangular button labeled "LOGIN". At the bottom center is the text "Create new account".

4.4.3 Home page



A wireframe of a home page. At the top left is a circular button labeled "Profile". At the top right is a circular button labeled "Notification". Below these are four rectangular buttons arranged in a 2x2 grid: "Preference" (top left), "Booking" (top right), "Assistance" (bottom left), and "Guidance" (bottom right). At the bottom left is a vertical list of five items: "Image 1", "Image 2", "Image 3", "Image 4", and "Image 5".

4.4.4 Booking

BOOKING

BUTTON 1

BUTTON 2

BUTTON 3

4.4.5 Assistance

ASSISTANCE

MEDICAL

VEHICLE

4.4.6 Guidance

GUIDANCE

GUIDE

JOIN US

4.5 Structure of Reports Being Created

List all kinds of reports that you intend to produce. Identify each report with a unique title and list the fields that appear in each of the reports.

4.6 Database Design

A database is a data structure that stores organised information. Most databases contain multiple tables, which may each include several different fields. For example, a company database may include tables for products, employees, and financial records. Each of these tables would have different fields that are relevant to the information stored in the table. Nearly all e-commerce sites use databases to store product inventory and customer information. These sites use a database management system (or DBMS), such as Microsoft Access, FileMaker Pro, or MySQL as the "back end" to the website. By storing website data in a database, the data can be easily searched, sorted, and updated. This flexibility is important for e-commerce sites and other types

of dynamic websites. Early databases were relatively "flat," which means they were limited to simple rows and columns, like a spreadsheet. (See also "flat file database"). However, today's relational databases allow users to access, update, and search information based on the relationship of data stored in different tables. Relational databases can also run queries that involve multiple databases. While early databases could only store text or numeric data, modern databases also let users store other data types such as sound clips, pictures, and videos.

The database design has several specific objectives:

- Control redundancy
- Ease of Learning and use
- Data independence
- More information at low cost
- Accuracy and integrity
- Recovery from failure
- Privacy and security
- Performance

Normalization

Normalization is the process of organising data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency. Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database. There are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications. As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

The following descriptions include examples:

First Normal Form

- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary

Do not use multiple fields in a single table to store similar data. For example, to track an inventory item that may come from two possible sources, an inventory record may contain fields for Vendor Code 1 and Vendor Code 2. What happens when you add a third vendor? Adding a field is not the answer; it requires program and table modifications and does not smoothly accommodate a dynamic number of vendors. Instead, place all vendor information in a separate table called Vendors, then link inventory to vendors with an item number key, or vendors to inventory with a vendor code key.

Second Normal Form

- Create separate tables for sets of values that apply to multiple records.
- Relate these tables with a foreign key.

Records should not depend on anything other than a table's primary key (a compound key, if necessary). For example, consider a customer's address in an accounting system. The address is needed by the Customers table, but also by the Orders, Shipping, Invoices, Accounts Receivable, and Collections tables. Instead of storing the customer's address as a separate entry in each of these tables, store it in one place, either in the Customers table or in a separate Addresses table.

Third Normal Form

- Eliminate fields that do not

Values in a record that are not part of that record's key do not belong in the table. In general, any time the contents of a group of fields may apply to more than a single record in the table, consider placing those fields in a separate table. For example, in an Employee Recruitment table, a candidate's university name and address may be included. But you need a complete list of universities for group mailings. If university information is stored in the Candidates table, there is no way to list universities with no current candidates. Create a separate Universities table and link it to the Candidates table with a university code key.

EXCEPTION:

Adhering to the third normal form, while theoretically desirable, is not always practical. If you have a Customers table and you want to eliminate all possible interfiled dependencies, you must

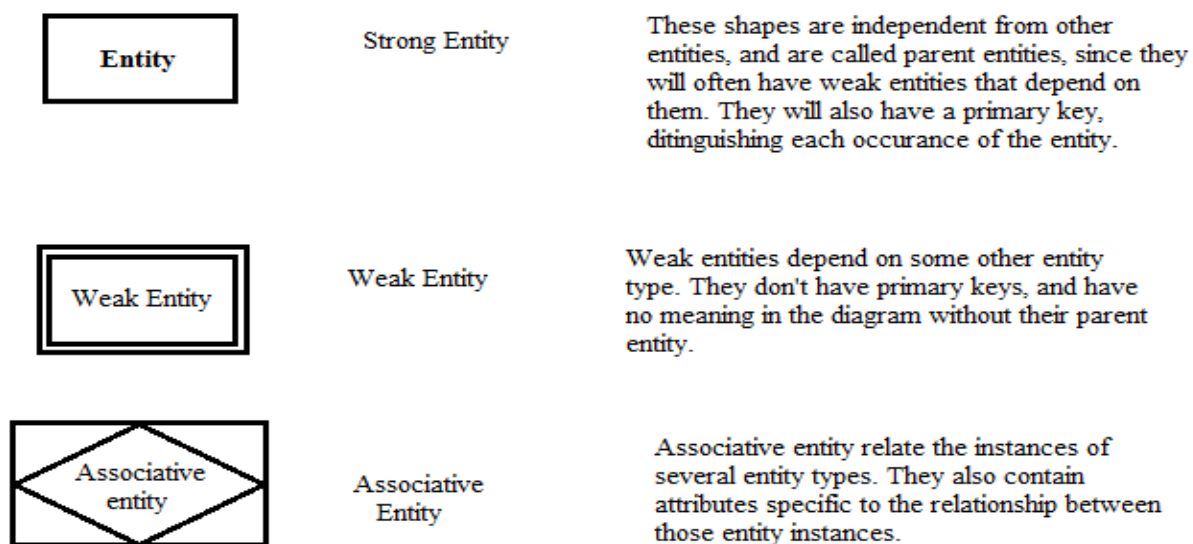
create separate tables for cities, ZIP codes, sales representatives, customer classes, and any other factor that may be duplicated in multiple records. In theory, normalization is worth pursuing. However, many small tables may degrade performance or exceed open file and memory capacities. It may be more feasible to apply third normal form only to data that changes frequently. If some dependent fields remain, design your application to require the user to verify all related fields when any one is changed.

Other Normalization Forms

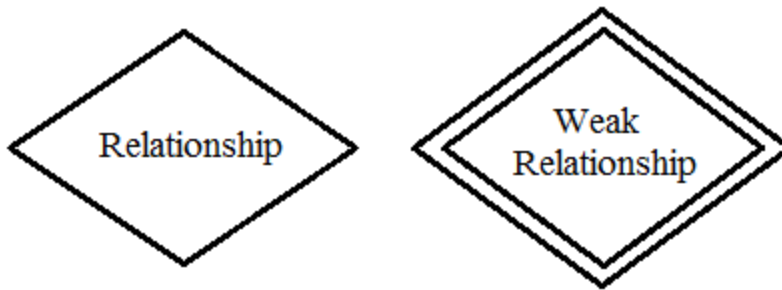
Fourth normal form, also called Boyce Codd Normal Form (BCNF), and fifth normal form do exist, but are rarely considered in practical design. Disregarding these rules may result in less than perfect database design, but should not affect functionality.

4.6.1 List of Entities and Attributes

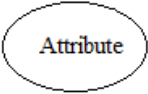
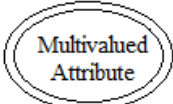
Entities are objects or concepts that represent important data. Entities are typically nouns such as product, customer, location or promotion. There are three types of entities commonly used in entity relationship diagrams.



Relationships are meaningful associations between or among entities. They are usually verbs, example-assign, associate or track. A relationship provides usual information that couldn't be discerned with the just entity types. Weak relationships or identifying relationships are connections that exist between a weak entity type and its owner.



ERD attributes are characteristics of the entity that help users to better understand the database. Attributes are included to include details of the various entities that are highlighted in a conceptual ER diagram

Attribute Symbol	Name	Description
	Attribute	Attributes are characteristics of an entity, a many-to-many relationship, or a one to one relationship
	Multivalued attribute	Multivalued attributes are those that can take on more than one value.

4.6.2 E R Diagram

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an

information system's entities and the relationships between those entities.

An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

- Attributes
- Relationships
- Entities

Steps involved in creating an ERD include:

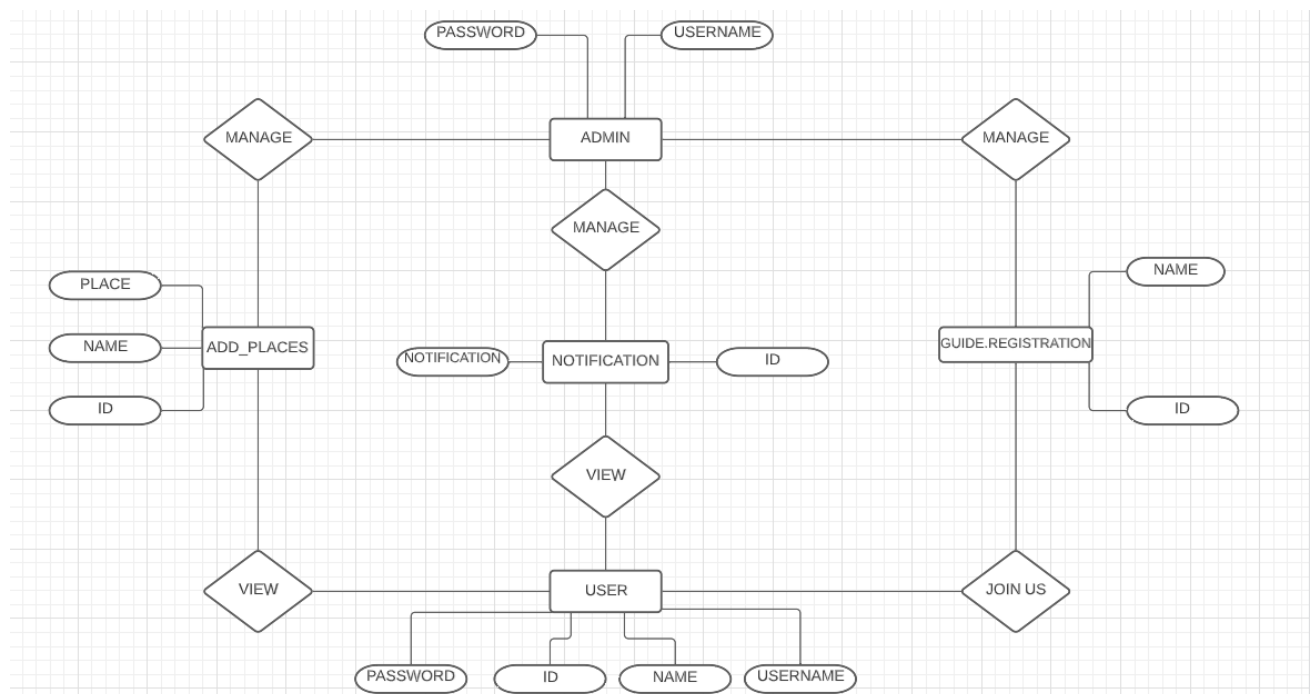
- Identifying and defining the entities
- Determining all interactions between the entities
- Analysing the nature of interactions/determining the cardinality of the relationships.
- Creating the ERD

An entity-relationship diagram (ERD) is crucial to creating a good database design. It is used as a high-level logical data model, which is useful in developing a conceptual design for databases. An entity is a real-world item or concept that exists on its own. Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity.

An attribute of an entity is a particular property that describes the entity. A relationship is the association that describes the interaction between entities. Cardinality, in the context of ERD, is the number of instances of one entity that can, or must, be associated with each instance of another entity. In general, there may be one-to-one, one-to-many, or many-to-many relationships. For example, let us consider two real-world entities, an employee and his department. An employee has attributes such as an employee number, name, department number, etc. Similarly, department number and name can be defined as attributes of a department. A department can interact with many employees, but an employee can belong to only one department, hence there can be a one-to-many relationship, defined between department and employee.

In the actual database, the employee table will have department number as a foreign key, referencing from department table, to enforce the relationship.

The Entity Relationship diagram of Unification of Images is as follows:



4.6.3 Structure of Tables

Database design is the process of producing a detailed data model of a database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database system (DBMS).

- Determine the relationships between the different data elements.
- Determine the data to be stored in the database.
- Superimpose a logical structure upon the data on the basis of these relationships.

4.6.3.1 Registration table

Column Name	Data Type	Constraints
Name	Varchar(30)	
Phone Number	Varchar(30)	
Address	Varchar(30)	
E-mail	Varchar(30)	primary key
Password	Varchar(30)	

Chapter 5

Implementation

5.1 Introduction

Systems implementation is a set of procedures performed to complete the design contained in the approved systems design document and to test, install, and begin to use the new or revised Information System. It is the fifth major step in the development of an Information System. In software development, an implementation is a realisation of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment.

5.2 Tools/Scripts for Implementation

Project planning tools include charts and graphs designed to track progress, repetition-based approaches to testing and adjusting everyday processes, and other actions that allow organisations to manage and improve important projects.

Tools for project implementation are defined as the series of systems and methodologies designed to ensure teams are able to accomplish both short- and long-term projects.

In our project, we used flow charts, Data Flow Diagrams, Entity-Relationship diagrams as tools for project implementation

5.3 Process Logic

Project logic provides the basis for planning and implementing monitoring and evaluation at project level. Project logic is defined as a conceptual framework of how a program or project is understood, or intended, to contribute to its specified outcomes. It focuses on outcomes rather than process.

Process logic is a cause-and-effect explanation of a process. It expresses all the principal definitions and arguments that appear to be true for the process and its events, causes and circumstances.

5.4 Coding

XML CODES:

Activity_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="HomeActivity"
    android:background="@drawable/background">
    <ImageView
        android:id="@+id/btnProfile"
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:layout_marginTop="10dp"
        app:srcCompat="@drawable/ic_baseline_account_circle_24" />
    <ImageView
        android:id="@+id/btnNotifications"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_alignParentRight="true"
        android:layout_marginTop="15dp"
        app:srcCompat="@drawable/ic_baseline_notifications_24" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/btnProfile"
        android:orientation="vertical"
        android:layout_centerVertical="true"
        android:layout_marginHorizontal="30dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="110dp"
        android:orientation="horizontal"
        android:gravity="center_vertical"
        android:weightSum="2">
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_margin="10dp"
        android:layout_height="100dp"
        android:layout_weight="1"
        app:cardCornerRadius="10dp"
        android:id="@+id/cardPrefrence"
        app:cardElevation="6dp">
```

```
</androidx.cardview.widget.CardView>
<androidx.cardview.widget.CardView
    android:layout_width="0dp"
    android:layout_margin="10dp"
    android:layout_height="100dp"
    android:layout_weight="1"
    app:cardCornerRadius="10dp"
    android:id="@+id/cardAssistence"
    app:cardElevation="6dp">

</androidx.cardview.widget.CardView>

</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="110dp"
    android:orientation="horizontal"
    android:gravity="center_vertical"
    android:weightSum="2">
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:id="@+id/cardBook"
        android:layout_margin="10dp"
        android:layout_height="100dp"
        android:layout_weight="1"
        app:cardCornerRadius="10dp"
        app:cardElevation="6dp">

        </androidx.cardview.widget.CardView>
        <androidx.cardview.widget.CardView
            android:layout_width="0dp"
            android:layout_margin="10dp"
            android:layout_height="100dp"
            android:layout_weight="1"
            android:id="@+id/cardGuid"
            app:cardCornerRadius="10dp"
            app:cardElevation="6dp">

            </androidx.cardview.widget.CardView>

        </LinearLayout>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler"
```



```

    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="10dp"
    android:layout_marginTop="12dp"
    android:layout_below="@id/tvHead"/>

```

```

<TextView
    android:id="@+id/tvNotFound"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="No nearby places found"
    android:layout_centerInParent="true"
    android:textColor="@color/white"
    android:textSize="17sp"
    android:textStyle="bold"
    android:visibility="gone"/>

```

```

</LinearLayout>

```

```

</RelativeLayout>

```

Activity_register.xml

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="RegisterActivity"
    android:background="@drawable/background">

```

```

<TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Registration"
    android:textSize="30sp"
    android:textColor="@color/yellow_500"
    android:textStyle="bold"
    android:fontFamily="serif"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="10dp"/>

```

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"

```

```
android:layout_centerVertical="true"
android:layout_marginHorizontal="30dp">

<EditText
    android:id="@+id/etusername"
    android:layout_width="match_parent"
    android:layout_height="45dp"
    android:hint="Username"
    android:background="@drawable/edittext_background"
    android:paddingHorizontal="10dp"/>

<EditText
    android:id="@+id/etpassword"
    android:layout_width="match_parent"
    android:layout_height="45dp"
    android:hint="Password"
    android:inputType="textPassword"
    android:layout_marginTop="8dp"
    android:background="@drawable/edittext_background"
    android:paddingHorizontal="10dp"
/>

<EditText
    android:id="@+id/etEmail"
    android:layout_width="match_parent"
    android:layout_height="45dp"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:layout_marginTop="8dp"
    android:background="@drawable/edittext_background"
    android:paddingHorizontal="10dp"/>

<EditText
    android:id="@+id/etphone"
    android:layout_width="match_parent"
    android:layout_height="45dp"
    android:hint="Phone"
    android:inputType="phone"
    android:layout_marginTop="8dp"
    android:background="@drawable/edittext_background"
    android:paddingHorizontal="10dp"/>

<EditText
    android:id="@+id/etaddress"
    android:layout_width="match_parent"
    android:layout_height="45dp"
    android:hint="Address"
```

```

    android:inputType="text"
    android:layout_marginTop="8dp"
    android:background="@drawable/edittext_background"
    android:paddingHorizontal="10dp"/>

```

```

<Button
    android:id="@+id/BtRegister"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Register"
    android:layout_marginTop="30dp" />

```

```

</LinearLayout>

```

```

</RelativeLayout>

```

Activity_assistance.xml

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:background="@drawable/background"
    tools:context=".AssistanceActivity">

```

```

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:orientation="vertical">

```

```

    <androidx.cardview.widget.CardView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnTransport"
        app:cardElevation="10dp"
        app:cardCornerRadius="10dp"
        android:layout_margin="10dp">

```

```

    <LinearLayout
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:orientation="vertical">
        <ImageView
            android:layout_width="150dp"

```

```

        android:layout_height="130dp"
        android:src="@drawable/healthcare"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/black"
            android:textSize="17sp"
            android:fontFamily="serif"
            android:layout_gravity="center_horizontal"
            android:text=" MEDICAL" />
    </LinearLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    app:cardCornerRadius="10dp"
    android:id="@+id/btnMedicine"
    app:cardElevation="10dp"
    android:layout_margin="10dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:orientation="vertical"
        >
        <ImageView
            android:layout_width="150dp"
            android:layout_height="130dp"
            android:src="@drawable/wrench"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/black"
            android:textSize="17sp"
            android:fontFamily="serif"
            android:layout_gravity="center_horizontal"
            android:text="vehicle" />
    </LinearLayout>
</androidx.cardview.widget.CardView>

</LinearLayout>

</RelativeLayout>
Activity_book.xml

```

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"

    tools:context=".BookActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_margin="10dp"
        android:orientation="vertical">
    <Button
        android:id="@+id/btnTicket1"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginTop="20dp"
        android:text="Book ticket" />
    <Button
        android:id="@+id/btnTicket2"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginTop="20dp"
        android:text="Book ticket" />
    <Button
        android:id="@+id/btnTicket3"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginTop="20dp"
        android:text="Book ticket" />
    </LinearLayout>
</RelativeLayout>

```

Activity_preference.xml

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".PreferencesActivity"
    android:background="@drawable/background">
<TextView
    android:id="@+id/tvHead"

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Preferences"
        android:textColor="@color/yellow_500"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp" />
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="21dp"
    android:layout_below="@id/tvHead"
    android:layout_marginBottom="21dp">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_marginHorizontal="12dp">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Select your interested categories:"
    android:textColor="@color/white" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="12dp">

<androidx.cardview.widget.CardView
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    app:cardCornerRadius="6dp"
    app:cardElevation="6dp"
    android:layout_marginEnd="5dp">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:src="@drawable/oceans"

```

```

        android:scaleType="fitXY"/>
        <CheckBox
            android:id="@+id/chkOcean"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Ocean"/>
    </LinearLayout>
    </androidx.cardview.widget.CardView>
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:cardCornerRadius="6dp"
        app:cardElevation="6dp"
        android:layout_marginStart="5dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:gravity="center">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="150dp"
                android:src="@drawable/mountain"
                android:scaleType="fitXY"/>
            <CheckBox
                android:id="@+id/chkMountain"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Mountain"/>
            </LinearLayout>
        </androidx.cardview.widget.CardView>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="10dp">
        <androidx.cardview.widget.CardView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            app:cardCornerRadius="6dp"
            app:cardElevation="6dp"
            android:layout_marginEnd="5dp">
            <LinearLayout

```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center">
        <ImageView
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:src="@drawable/forest"
            android:scaleType="fitXY"/>
        <CheckBox
            android:id="@+id/chkForest"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Forest"/>
    </LinearLayout>
</androidx.cardview.widget.CardView>
<androidx.cardview.widget.CardView
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    app:cardCornerRadius="6dp"
    app:cardElevation="6dp"
    android:layout_marginStart="5dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center">
        <ImageView
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:src="@drawable/zoo"
            android:scaleType="fitXY"/>
        <CheckBox
            android:id="@+id/chkZoo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Zoo"/>
    </LinearLayout>
</androidx.cardview.widget.CardView>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="10dp">

```



```

<androidx.cardview.widget.CardView
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    app:cardCornerRadius="6dp"
    app:cardElevation="6dp"
    android:layout_marginEnd="5dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center">
        <ImageView
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:src="@drawable/lake"
            android:scaleType="fitXY"/>
        <CheckBox
            android:id="@+id/chkRiver"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="River"/>
        </LinearLayout>
    </androidx.cardview.widget.CardView>
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:cardCornerRadius="6dp"
        app:cardElevation="6dp"
        android:layout_marginStart="5dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:gravity="center">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="150dp"
                android:src="@drawable/others"
                android:scaleType="fitXY"/>
            <CheckBox
                android:id="@+id/chkOther"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

```

```

        android:text="Others"/>
    </LinearLayout>
</androidx.cardview.widget.CardView>
</LinearLayout>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Select your preferred radius (in KM) for getting suggestions:"
    android:textColor="@color/white"
    android:layout_marginTop="21dp"/>
<EditText
    android:id="@+id/etRadius"
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:hint="Radius in KM"
    android:text="10"
    android:background="@drawable/edittext_background"
    android:inputType="number"
    android:layout_marginTop="5dp"
    android:paddingHorizontal="10dp"
    android:layout_marginHorizontal="20dp"/>
<Button
    android:id="@+id/btnUpdate"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Update Preferences"
    android:layout_marginTop="30dp"
    android:layout_marginHorizontal="20dp"
    android:layout_marginBottom="5dp"/>
</LinearLayout>
</ScrollView>
</RelativeLayout>
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="ProfileActivity"
    android:background="@drawable/background">

    <TextView
        android:id="@+id/tvHead"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Profile"

```

```

        android:textColor="@color/yellow_500"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp" />
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/tvHead"
    android:layout_marginTop="10dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginHorizontal="30dp"
        android:layout_marginTop="10dp">
        <com.google.android.material.textfield.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColorHint="@color/yellow_500"
            android:layout_marginTop="15dp"
            style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.Dense">
            <EditText
                android:id="@+id/etUsername"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Username"
                android:textColor="@color/white"/>
            </com.google.android.material.textfield.TextInputLayout>
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textColorHint="@color/yellow_500"
                android:layout_marginTop="6dp"
                style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.Dense">
                <EditText
                    android:id="@+id/etEmail"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Email"
                    android:inputType="textEmailAddress"
                    android:textColor="@color/white"/>
                </com.google.android.material.textfield.TextInputLayout>
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"

```

```

        android:textColorHint="@color/yellow_500"
        android:layout_marginTop="6dp"
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.Dense">
    <EditText
        android:id="@+id/etPhone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Phone"
        android:inputType="phone"
        android:textColor="@color/white"/>
    </com.google.android.material.textfield.TextInputLayout>
    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColorHint="@color/yellow_500"
        android:layout_marginTop="6dp"
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.Dense">
    <EditText
        android:id="@+id/etAddress"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:hint="Address"
        android:gravity="top"
        android:textColor="@color/white"/>
    </com.google.android.material.textfield.TextInputLayout>

    <TextView
        android:id="@+id/tvType"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Account type: User"
        android:layout_gravity="center_horizontal"
        android:layout_marginVertical="10dp"
        android:textColor="@color/yellow_500"/>
    <Button
        android:id="@+id/btnSave"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Save Changes"
        android:layout_marginTop="21dp"
        android:layout_marginBottom="30dp"/>
    </LinearLayout>
</ScrollView>
</RelativeLayout>
JAVA
RegisterActivity

```

```
package com.example.tourist;

import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.RetryPolicy;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.tourist.utils.Config;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegisterActivity extends AppCompatActivity {

    EditText etusername, etpassword, etEmail, etphone, etaddress;
    Button btnRegister;
    String status,message,url = Config.baseURL + "reg.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        getSupportActionBar().hide();

        etusername = findViewById(R.id.etusername);
        etpassword = findViewById(R.id.etpassword);
        etEmail = findViewById(R.id.etEmail);
        etphone = findViewById(R.id.etphone);
        etaddress = findViewById(R.id.etaddress);
        btnRegister = findViewById(R.id.BtnRegister);
        btnRegister.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
        public void onClick(View view) {
            registeruser();
        }
    });
}

private void registeruser() {
    String username, password, email, phone, address;
    username = etusername.getText().toString();
    password = etpassword.getText().toString();
    email = etEmail.getText().toString();
    phone = etphone.getText().toString();
    address = etaddress.getText().toString();

    //validation
    if (TextUtils.isEmpty(username.trim())) {
        etusername.setError("username is required");
        etusername.requestFocus();
        return;
    } else if (TextUtils.isEmpty(password.trim())) {
        etpassword.setError("password is required");
        etpassword.requestFocus();
        return;
    }

    if (TextUtils.isEmpty(email.trim())) {
        etEmail.setError("Email id is required");
        etEmail.requestFocus();
        return;
    }

    else if (!isEmailValid(email)) {
        etEmail.setError("Invalid Email id");
        etEmail.requestFocus();
        return;
    }

    else if (TextUtils.isEmpty(phone.trim())) {
        etphone.setError("phone number is required");
        etphone.requestFocus();
        return;
    }

    else if (!isPhoneValid(phone)) {
        etphone.setError("Invalid phone number");
        etphone.requestFocus();
    }
}
```

```

        return;
    }
    else if (TextUtils.isEmpty(address.trim())) {
        etpassword.setError("address is required");
        etpassword.requestFocus();
        return;
    }

    Config.showSimpleProgressDialog(this);
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Config.removeSimpleProgressDialog();
                try {
                    JSONObject c = new JSONObject(response);
                    status = c.getString("status");
                    message = c.getString("message");

                    if (status.equals("0")) {
                        Toast.makeText(RegisterActivity.this, message, Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(RegisterActivity.this, message, Toast.LENGTH_SHORT).show();
                        finish();
                    }

                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Config.removeSimpleProgressDialog();
                Toast.makeText(RegisterActivity.this, String.valueOf(error),
                    Toast.LENGTH_SHORT).show();
            }
        }
    );

    @Override
    protected Map<String, String> getParams() {
        Map<String, String> params = new HashMap<>();
        params.put("username", username);
        params.put("password", password);
        params.put("email", email);
        params.put("phone", phone);
    }

```

```

        params.put("address", address);
        return params;
    }
};

stringRequest.setRetryPolicy(new RetryPolicy() {
    @Override
    public int getCurrentTimeout() {
        return 20000;
    }
    @Override
    public int getCurrentRetryCount() {
        return 20000;
    }
    @Override
    public void retry(VolleyError error) {
        Toast.makeText(RegisterActivity.this, error.toString(), Toast.LENGTH_SHORT).show();
    }
});
RequestQueue requestQueue = Volley.newRequestQueue(this);
requestQueue.add(stringRequest);
}

//Regex
public static boolean isPhoneValid(String s) {
    Pattern p = Pattern.compile("(0/91)?[6-9][0-9]{9}");
    Matcher m = p.matcher(s);
    return (m.find() && m.group().equals(s));
}

public static boolean isEmailValid(String email) {
    String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\." + "[a-zA-Z0-9_+&*-]+)*@" +
    "(?:[a-zA-Z0-9-]+\\.)+[a-z" + "A-Z]{2,7}$";
    Pattern pat = Pattern.compile(emailRegex);
    return pat.matcher(email).matches();
}
}

ProfileActivity
package com.example.tourist;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

```



```
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.tourist.session.SessionManager;
import com.example.tourist.utils.Config;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;

public class ProfileActivity extends AppCompatActivity {

    EditText etUsername, etEmail, etPhone, etAddress;
    TextView tvType;
    Button btnSave;
    String id, username, email, phone, address, type;
    String status = "", message = "", url = Config.baseURL + "update_profile.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile_app);
        getSupportActionBar().hide();

        etUsername = findViewById(R.id.etUsername);
        etEmail = findViewById(R.id.etEmail);
        etPhone = findViewById(R.id.etPhone);
        etAddress = findViewById(R.id.etAddress);
        tvType = findViewById(R.id.tvType);
        btnSave = findViewById(R.id.btnSave);

        HashMap<String, String> data = new SessionManager(this).getUserDetails();
        id = data.get("id");
        username = data.get("username");
        email = data.get("email");
        phone = data.get("phone");
```

```

address = data.get("address");
type = data.get("type");

etUsername.setText(username);
etEmail.setText(email);
etPhone.setText(phone);
etAddress.setText(address);
tvType.setText("Account type: " + type);

btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        saveChanges();
    }
});
}

private void saveChanges() {
    status = "";
    message = "";
    String username, email, phone, address;

    username = etUsername.getText().toString();
    email = etEmail.getText().toString();
    phone = etPhone.getText().toString();
    address = etAddress.getText().toString();

    Config.showSimpleProgressDialog(this);
    StringRequest request = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Config.removeSimpleProgressDialog();
                try {
                    //JSON Parsing
                    JSONObject data = new JSONObject(response);
                    status = data.getString("status");
                    message = data.getString("message");

                    if (status.equals("1")) {
                        Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT).show();
                        new SessionManager(ProfileActivity.this).updateProfile(
                            username, email, phone, address
                        );
                    }
                }
            }
        });
}

```

```

        finish();
    }
    else {
        Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT).show();
    }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Config.removeSimpleProgressDialog();
        Toast.makeText(getApplicationContext(), error.toString(),
Toast.LENGTH_SHORT).show();
    }
}) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("id", id);
        params.put("username", username);
        params.put("email", email);
        params.put("phone", phone);
        params.put("address", address);
        return params;
    }
};
Volley.newRequestQueue(this).add(request);
}
}

```

PreferenceActivity

package com.example.tourist;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.EditText;

```
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.tourist.session.SessionManager;
import com.example.tourist.utils.Config;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class PreferencesActivity extends AppCompatActivity {

    CheckBox chkOcean, chkMountain, chkForest, chkZoo, chkRiver, chkOthers;
    EditText etRadius;
    Button btnUpdate;
    ArrayList<String> pref;
    String id, p, preferences, status, message, url = Config.baseURL + "set_preferences.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_preferences);
        getSupportActionBar().hide();

        chkOcean = findViewById(R.id.chkOcean);
        chkMountain = findViewById(R.id.chkMountain);
        chkForest = findViewById(R.id.chkForest);
        chkZoo = findViewById(R.id.chkZoo);
        chkRiver = findViewById(R.id.chkRiver);
        chkOthers = findViewById(R.id.chkOther);
        etRadius = findViewById(R.id.etRadius);
        btnUpdate = findViewById(R.id.btnUpdate);

        id = new SessionManager(this).getUserDetails().get("id");
        p = new SessionManager(this).getUserDetails().get("preferences");
```

```

try {
    JSONArray array = new JSONArray(p);
    for (int i = 0; i < array.length(); i++) {
        String p = array.getString(i);
        if (p.equals("Ocean")) {
            chkOcean.setChecked(true);
        }
        if (p.equals("Mountain")) {
            chkMountain.setChecked(true);
        }
        if (p.equals("Forest")) {
            chkForest.setChecked(true);
        }
        if (p.equals("Zoo")) {
            chkZoo.setChecked(true);
        }
        if (p.equals("River")) {
            chkRiver.setChecked(true);
        }
        if (p.equals("Others")) {
            chkOthers.setChecked(true);
        }
    }
} catch (JSONException e) {
    e.printStackTrace();
}

```

```

btnUpdate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        updatePreferences();
    }
});
}

```

```

private void updatePreferences() {
    pref = new ArrayList<>();
    preferences = "";

    boolean ocean = chkOcean.isChecked();
    boolean mountain = chkMountain.isChecked();
    boolean forest = chkForest.isChecked();
    boolean zoo = chkZoo.isChecked();
    boolean river = chkRiver.isChecked();
    boolean others = chkOthers.isChecked();
}

```

```

String radius = etRadius.getText().toString();

if (!ocean && !mountain && !forest && !zoo && !river && !others) {
    Toast.makeText(this, "Please select a preference for updatation", Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(radius)) {
    etRadius.setError("Enter radius");
    etRadius.requestFocus();
    return;
}

if (ocean) {
    pref.add("Ocean");
}
if (mountain) {
    pref.add("Mountain");
}
if (forest) {
    pref.add("Forest");
}
if (zoo) {
    pref.add("Zoo");
}
if (river) {
    pref.add("River");
}
if (others) {
    pref.add("Others");
}

for (int i = 0; i < pref.size(); i++) {
    preferences = preferences + pref.get(i) + ",";
}

if (preferences != null && preferences.length() > 0 && preferences.charAt(preferences.length() - 1)
== ',') {
    preferences = preferences.substring(0, preferences.length() - 1);
    preferences = "[" + preferences + "]";
}

/*try {
    JSONArray array = new JSONArray(preferences);
    for (int i = 0; i < array.length(); i++) {
        String p = array.getString(i);
        Toast.makeText(this, p, Toast.LENGTH_SHORT).show();
    }
}

```

```

    }
} catch (JSONException e) {
    e.printStackTrace();
}*/

Config.showSimpleProgressDialog(this);
StringRequest request = new StringRequest(Request.Method.POST, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Config.removeSimpleProgressDialog();
            try {
                //JSON Parsing
                JSONObject data = new JSONObject(response);
                status = data.getString("status");
                message = data.getString("message");

                if (status.equals("1")) {
                    Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT).show();
                    new SessionManager(PreferencesActivity.this).updatePreferences(preferences,
radius);

                    finish();
                }
                else {
                    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_LONG).show();
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(getApplicationContext(), error.toString(),
Toast.LENGTH_SHORT).show();
        }
    }) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("id", id);
    }
}

```

```

        params.put("preferences", preferences);
        params.put("radius", radius);
        return params;
    }
};
Volley.newRequestQueue(this).add(request);
}
}

```

LoginActivity

```
package com.example.tourist;
```

```

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

```

```
import androidx.appcompat.app.AppCompatActivity;
```

```

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.tourist.session.SessionManager;
import com.example.tourist.utils.Config;

```

```

import org.json.JSONException;
import org.json.JSONObject;

```

```

import java.util.HashMap;
import java.util.Map;

```

```
public class LoginActivity extends AppCompatActivity {
```

```

    EditText etuser, etpass;
    Button btlogin;
    TextView tvregister;
    String status = "", message = "", url = Config.baseURL + "login.php";
    String id, email, phone, address, latitude, longitude, type, preferences, radius;

```



```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    getSupportActionBar().hide();

    etuser = findViewById(R.id.etusername);
    etpass = findViewById(R.id.etpassword);
    btlogin = findViewById(R.id.btnlogin);
    tvregister = findViewById(R.id.tvregister);

    tvregister.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent(LoginActivity.this, RegisterActivity.class);
            startActivity(i);
        }
    });

    btlogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Loginuser();

        }
    });

}

private void Loginuser(){
    String username = etuser.getText().toString();
    String password = etpass.getText().toString();
    if (TextUtils.isEmpty(username.trim())){
        etuser.setError("username is required");
        etuser.requestFocus();
        return;

    }
    else if (TextUtils.isEmpty(password.trim())){
        etpass.setError("password is required");
        etpass.requestFocus();
        return;
    }

    Config.showSimpleProgressDialog(this);
}

```

```

StringRequest request = new StringRequest(Request.Method.POST, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Config.removeSimpleProgressDialog();
            try {
                //JSON Parsing
                JSONObject data = new JSONObject(response);
                status = data.getString("status");
                message = data.getString("message");

                id = data.getString("id");
                email = data.getString("email");
                phone = data.getString("phone");
                address = data.getString("address");
                latitude = data.getString("latitude");
                longitude = data.getString("longitude");
                preferences = data.getString("preferences");
                radius = data.getString("radius");
                type = data.getString("type");

                if (status.equals("1")) {
                    Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT).show();
                    new SessionManager(LoginActivity.this).createLoginSession(
                        id, username, password, email, phone, address,
                        latitude, longitude, type, preferences, radius);
                    Intent i = new Intent(getApplicationContext(), HomeActivity.class);
                    startActivity(i);
                    finish();
                }
                else {
                    Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT).show();
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Config.removeSimpleProgressDialog();

```

```

        Toast.makeText(getApplicationContext(), error.toString(),
        Toast.LENGTH_SHORT).show();
    }
}) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("username", username);
        params.put("password", password);
        return params;
    }
};
Volley.newRequestQueue(this).add(request);
}
}

```

ListPlacesActivity

package com.example.tourist;

```

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

```

```

import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

```

```

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.tourist.utils.Config;

```

```

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

```

```

import java.util.ArrayList;

```

```
import java.util.HashMap;
import java.util.Map;

import mumayank.com.airlocationlibrary.AirLocation;

public class ListPlacesActivity extends AppCompatActivity {

    String url = Config.baseURL + "list_places.php";
    ArrayList<PlaceDataModel> list;
    RecyclerView recycler;
    TextView tvNotFound;
    AirLocation airLocation;
    String latitude, longitude;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_places);
        getSupportActionBar().hide();

        recycler = findViewById(R.id.recycler);
        tvNotFound = findViewById(R.id.tvNotFound);

        fetchLocation();
    }

    private void fetchData() {
        list = new ArrayList<>();

        Config.showSimpleProgressDialog(this);

        StringRequest request = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Config.removeSimpleProgressDialog();
                    if (response.equals("null")) {
                        tvNotFound.setVisibility(View.VISIBLE);
                    }
                    else {
                        try {
                            JSONArray data = new JSONArray(response);

                            for (int i = 0; i < data.length(); i++) {
                                JSONObject user = data.getJSONObject(i);
```

```

        list.add(new PlaceDataModel(
            user.getString("id"),
            user.getString("name"),
            user.getString("place"),
            user.getString("category"),
            user.getString("image"),
            user.getString("history"),
            user.getString("rate"),
            user.getString("latitude"),
            user.getString("longitude")
        ));
    }

    } catch (JSONException e) {
        e.printStackTrace();
    }

    PlaceAdapter adapter = new PlaceAdapter(ListPlacesActivity.this, list);
    recycler.setHasFixedSize(true);
    recycler.setAdapter(adapter);
    recycler.setLayoutManager(new LinearLayoutManager(ListPlacesActivity.this,
LinearLayoutManager.VERTICAL, false));
    }
    }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(ListPlacesActivity.this, error.toString(), Toast.LENGTH_SHORT).show();
        }
    }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("latitude", latitude);
            params.put("longitude", longitude);
            return params;
        }
    };

    RequestQueue queue = Volley.newRequestQueue(this);
    queue.add(request);
}

```

```

private void fetchLocation() {

    Config.showSimpleProgressDialog(this);

    //AirLocation fetching process...
    airLocation = new AirLocation(ListPlacesActivity.this, true, true, new AirLocation.Callbacks() {
        @Override
        public void onSuccess(@NonNull Location location) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(ListPlacesActivity.this, "Location fetched successfully",
Toast.LENGTH_LONG).show();
            double lat, lng;
            lat = location.getLatitude();
            lng = location.getLongitude();
            latitude=Double.toString(lat);
            longitude=Double.toString(lng);

            fetchData();
        }

        @Override
        public void onFailed(@NonNull AirLocation.LocationFailedEnum locationFailedEnum) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(getApplicationContext(), "Location fetching failed. Please check your internet
connection", Toast.LENGTH_LONG).show();
        }
    });
}

// override and call airLocation object's method by the same name
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    airLocation.onActivityResult(requestCode, resultCode, data);
}

// override and call airLocation object's method by the same name
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    airLocation.onRequestPermissionsResult(requestCode, permissions, grantResults);
}

```

```
    }  
}  
JoinActivity  
package com.example.tourist;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.database.Cursor;  
import android.location.Location;  
import android.net.Uri;  
import android.os.Bundle;  
import android.provider.OpenableColumns;  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import com.android.volley.AuthFailureError;  
import com.android.volley.DefaultRetryPolicy;  
import com.android.volley.NetworkResponse;  
import com.android.volley.Request;  
import com.android.volley.RequestQueue;  
import com.android.volley.Response;  
import com.android.volley.VolleyError;  
import com.android.volley.toolbox.Volley;  
import com.example.tourist.session.SessionManager;  
import com.example.tourist.utils.Config;  
import com.example.tourist.utils.VolleyMultipartRequest;  
  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.ByteArrayOutputStream;  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.HashMap;  
import java.util.Map;  
  
import mumayank.com.airlocationlibrary.AirLocation;
```

```
public class JoinActivity extends AppCompatActivity {

    EditText etPrice;
    Button btnResume, btnCertificate;
    String price, id, status, message, latitude, longitude;

    private AirLocation airLocation;
    private RequestQueue rQueue;
    String resumeURL = Config.baseURL + "upload_resume.php";
    String certificateURL = Config.baseURL + "upload_certificate.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_join);
        getSupportActionBar().hide();

        id = new SessionManager(this).getUserDetails().get("id");

        etPrice = findViewById(R.id.etPrice);
        btnResume = findViewById(R.id.btnResume);
        btnCertificate = findViewById(R.id.btnCertificate);

        btnResume.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                fetchLocation();
            }
        });

        btnCertificate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                certificate();
            }
        });

        fetchLocation();
    }

    private void resume() {
        price = etPrice.getText().toString();

        if (TextUtils.isEmpty(price)) {
            etPrice.setError("Enter price");
        }
    }
}
```



```
        etPrice.requestFocus();
        return;
    }

    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.setType("application/pdf");
    startActivityForResult(intent,1);
}

private void certificate() {
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.setType("application/pdf");
    startActivityForResult(intent,2);
}

private void fetchLocation() {
    Config.showSimpleProgressDialog(this);

    //AirLocation fetching process...
    airLocation = new AirLocation(JoinActivity.this, true, true, new AirLocation.Callbacks() {
        @Override
        public void onSuccess(@NonNull Location location) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(JoinActivity.this, "Location fetched successfully",
Toast.LENGTH_LONG).show();
            latitude = Double.toString(location.getLatitude());
            longitude = Double.toString(location.getLongitude());

            resume();
        }

        @Override
        public void onFailed(@NonNull AirLocation.LocationFailedEnum locationFailedEnum) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(JoinActivity.this, "Location fetching failed. Please check your internet
connection", Toast.LENGTH_LONG).show();
        }
    });
}
```

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK) {
        if (requestCode == 1) {
            // Get the Uri of the selected file
            Uri uri = data.getData();
            String uriString = uri.toString();
            File myFile = new File(uriString);
            String path = myFile.getAbsolutePath();
            String displayName = null;

            if (uriString.startsWith("content://")) {
                Cursor cursor = null;
                try {
                    cursor = this.getContentResolver().query(uri, null, null, null, null);
                    if (cursor != null && cursor.moveToFirst()) {
                        displayName =
cursor.getString(cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME));
                        Log.d("nameeeee>>>> ",displayName);

                        uploadResume(displayName,uri);
                    }
                } finally {
                    cursor.close();
                }
            } else if (uriString.startsWith("file://")) {
                displayName = myFile.getName();
                Log.d("nameeeee>>>> ",displayName);
            }
        }
        else if (requestCode == 2) {
            // Get the Uri of the selected file
            Uri uri = data.getData();
            String uriString = uri.toString();
            File myFile = new File(uriString);
            String path = myFile.getAbsolutePath();
            String displayName = null;

            if (uriString.startsWith("content://")) {
                Cursor cursor = null;
                try {
                    cursor = this.getContentResolver().query(uri, null, null, null, null);
                    if (cursor != null && cursor.moveToFirst()) {

```

```

        displayName =
cursor.getString(cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME));
        Log.d("nameeeee>>>> ",displayName);

        uploadCertificate(displayName,uri);
    }
    } finally {
        cursor.close();
    }
    } else if (uriString.startsWith("file:///")) {
        displayName = myFile.getName();
        Log.d("nameeeee>>>> ",displayName);
    }
}
}

super.onActivityResult(requestCode, resultCode, data);

}

private void uploadResume(final String pdfname, Uri pdffile) {
    InputStream iStream = null;
    try {

        iStream = getContentResolver().openInputStream(pdffile);
        final byte[] inputData = getBytes(iStream);

        Config.showSimpleProgressDialog(this);

        VolleyMultipartRequest volleyMultipartRequest = new
VolleyMultipartRequest(Request.Method.POST, resumeURL,
        new Response.Listener<NetworkResponse>() {
            @Override
            public void onResponse(NetworkResponse response) {
                Config.removeSimpleProgressDialog();
                Log.d("resssssoo",new String(response.data));
                rQueue.getCache().clear();
                try {
                    JSONObject jsonObject = new JSONObject(new String(response.data));

                    jsonObject.toString().replace("\\\\", "");

                    status = jsonObject.getString("status");
                    message = jsonObject.getString("message");

```

```

        if (status.equals("1")) {
            Toast.makeText(JoinActivity.this, message, Toast.LENGTH_SHORT).show();
            etPrice.setVisibility(View.GONE);
            btnResume.setVisibility(View.GONE);
        }
        else {
            Toast.makeText(JoinActivity.this, message, Toast.LENGTH_SHORT).show();
        }

    } catch (JSONException e) {
        e.printStackTrace();
    }
}

},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Config.removeSimpleProgressDialog();
        Toast.makeText(getApplicationContext(), error.toString(),
Toast.LENGTH_LONG).show();
    }
}) {

@Override
protected Map<String, String> getParams() throws AuthFailureError {
    Map<String, String> params = new HashMap<>();
    params.put("id", id);
    params.put("price", price);
    params.put("latitude", latitude);
    params.put("longitude", longitude);
    return params;
}

@Override
protected Map<String, DataPart> getByteData() {
    Map<String, DataPart> params = new HashMap<>();
    params.put("filename", new DataPart(pdfname ,inputData));
    return params;
}
};

volleyMultipartRequest.setRetryPolicy(new DefaultRetryPolicy(
    0,
    DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
    DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

```

```

rQueue = Volley.newRequestQueue(JoinActivity.this);
rQueue.add(volleyMultipartRequest);

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
}

private void uploadCertificate(final String pdfname, Uri pdfFile) {
    InputStream iStream = null;
    try {

        iStream = getContentResolver().openInputStream(pdfFile);
        final byte[] inputData = getBytes(iStream);

        Config.showSimpleProgressDialog(this);

        VolleyMultipartRequest volleyMultipartRequest = new
        VolleyMultipartRequest(Request.Method.POST, certificateURL,
            new Response.Listener<NetworkResponse>() {
                @Override
                public void onResponse(NetworkResponse response) {
                    Config.removeSimpleProgressDialog();
                    Log.d("resssssoo", new String(response.data));
                    rQueue.getCache().clear();
                    try {
                        JSONObject jsonObject = new JSONObject(new String(response.data));
                        jsonObject.toString().replace("\\\\", "");

                        status = jsonObject.getString("status");
                        message = jsonObject.getString("message");

                        if (status.equals("1")) {
                            Toast.makeText(JoinActivity.this, message, Toast.LENGTH_SHORT).show();
                            btnCertificate.setVisibility(View.GONE);
                            finish();
                        }
                        else {
                            Toast.makeText(JoinActivity.this, message, Toast.LENGTH_SHORT).show();
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            },
    },

```

```

        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Config.removeSimpleProgressDialog();
                Toast.makeText(getApplicationContext(), error.toString(),
Toast.LENGTH_LONG).show();
            }
        }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> params = new HashMap<>();
                params.put("id", id);
                return params;
            }
            @Override
            protected Map<String, DataPart> getByteData() {
                Map<String, DataPart> params = new HashMap<>();
                params.put("filename", new DataPart(pdfname ,inputData));
                return params;
            }
        };
        volleyMultipartRequest.setRetryPolicy(new DefaultRetryPolicy(
            0,
            DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
            DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
        rQueue = Volley.newRequestQueue(JoinActivity.this);
        rQueue.add(volleyMultipartRequest);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public byte[] getBytes(InputStream inputStream) throws IOException {
    ByteArrayOutputStream byteBuffer = new ByteArrayOutputStream();
    int bufferSize = 1024;
    byte[] buffer = new byte[bufferSize];
    int len = 0;
    while ((len = inputStream.read(buffer)) != -1) {
        byteBuffer.write(buffer, 0, len);
    }
    return byteBuffer.toByteArray();
}
}

HomeActivity
package com.example.tourist;

```

```
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.tourist.session.SessionManager;
import com.example.tourist.utils.Config;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import mumayank.com.airlocationlibrary.AirLocation;
public class HomeActivity extends AppCompatActivity {
    AirLocation airLocation;
    String latitude, longitude;
    CardView cardBook,cardAssistence,cardGuid,cardPrefrence;
    ImageView btnProfile,btnNotifications;
    String url = Config.baseURL + "list_places.php";
    ArrayList<PlaceDataModel> list;
```

```
RecyclerView recycler;
TextView tvNotFound;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);
    getSupportActionBar().hide();
    recycler = findViewById(R.id.recycler);
    tvNotFound = findViewById(R.id.tvNotFound);
    cardAssistence = findViewById(R.id.cardAssistence);
    cardGuid = findViewById(R.id.cardGuid);
    btnNotifications = findViewById(R.id.btnNotifications);
    btnProfile = findViewById(R.id.btnProfile);
    cardPrefrence = findViewById(R.id.cardPrefrence);
    cardBook = findViewById(R.id.cardBook);
    fetchLocation();
    cardBook.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(getApplicationContext(),BookActivity.class));
        }
    });
    cardPrefrence.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(getApplicationContext(),PreferencesActivity.class));
        }
    });
    cardAssistence.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(getApplicationContext(),AssistanceActivity.class));
        }
    });
    cardGuid.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(getApplicationContext(),GuideActivity.class));
        }
    });
    btnProfile.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(getApplicationContext(), ProfileActivity.class));
        }
    });
}
```



```

});
btnNotifications.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), ListNotificationsActivity.class));
    }
});
startService(new Intent(getApplicationContext(), NotifyService.class));
}
private void fetchData() {
    list = new ArrayList<>();
    Config.showSimpleProgressDialog(this);
    StringRequest request = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Config.removeSimpleProgressDialog();
                if (response.equals("null")) {
                    tvNotFound.setVisibility(View.VISIBLE);
                }
                else {
                    try {
                        JSONArray data = new JSONArray(response);

                        for (int i = 0; i < data.length(); i++) {
                            JSONObject user = data.getJSONObject(i);

                            list.add(new PlaceDataModel(
                                user.getString("id"),
                                user.getString("name"),
                                user.getString("place"),
                                user.getString("category"),
                                user.getString("image"),
                                user.getString("history"),
                                user.getString("rate"),
                                user.getString("latitude"),
                                user.getString("longitude")
                            ));
                        }

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                    PlaceAdapter adapter = new PlaceAdapter(HomeActivity.this, list);
                    recycler.setHasFixedSize(true);
                }
            }
        });
}

```

```

        recycler.setAdapter(adapter);
        recycler.setLayoutManager(new LinearLayoutManager(HomeActivity.this,
LinearLayoutManager.VERTICAL, false));
    }
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Config.removeSimpleProgressDialog();
        Toast.makeText(HomeActivity.this, error.toString(), Toast.LENGTH_SHORT).show();
    }
}) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("latitude", latitude);
        params.put("longitude", longitude);
        return params;
    }
};

RequestQueue queue = Volley.newRequestQueue(this);
queue.add(request);
}

@Override
public void onBackPressed() {
    showExitAlert();
}

private void showExitAlert() {
    new AlertDialog.Builder(this)
        .setTitle("Logout")
        .setMessage("Are you sure you want to logout from your account?")
        .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                new SessionManager(HomeActivity.this).logoutUser();
                startActivity(new Intent(HomeActivity.this, LoginActivity.class));
                finish();
            }
        })
        .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

```

```

        }
    })
    .show();
}

private void fetchLocation() {

    Config.showSimpleProgressDialog(this);

    //AirLocation fetching process...
    airLocation = new AirLocation(HomeActivity.this, true, true, new AirLocation.Callbacks() {
        @Override
        public void onSuccess(@NonNull Location location) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(HomeActivity.this, "Location fetched successfully",
Toast.LENGTH_LONG).show();
            double lat, lng;
            lat = location.getLatitude();
            lng = location.getLongitude();
            latitude=Double.toString(lat);
            longitude=Double.toString(lng);
            fetchData();

        }

        @Override
        public void onFailed(@NonNull AirLocation.LocationFailedEnum locationFailedEnum) {
            Config.removeSimpleProgressDialog();
            Toast.makeText(getApplicationContext(), "Location fetching failed. Please check your internet
connection", Toast.LENGTH_LONG).show();
        }
    });
}
// override and call airLocation object's method by the same name
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    airLocation.onActivityResult(requestCode, resultCode, data);
}
// override and call airLocation object's method by the same name
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    airLocation.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
}

```

```

}
GuideActivity
package com.example.tourist;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class GuideActivity extends AppCompatActivity {
    Button btnGuides,btnJoin;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_guide);

        btnGuides = findViewById(R.id.btnGuides);
        btnJoin = findViewById(R.id.btnJoin);

        btnGuides.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getApplicationContext(), ListGuidesActivity.class));
            }
        });

        btnJoin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getApplicationContext(), JoinActivity.class));
            }
        });
    }
}

```

```

BookTicketActivity
package com.example.tourist;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class BookTicketActivity extends AppCompatActivity {
    WebView web;

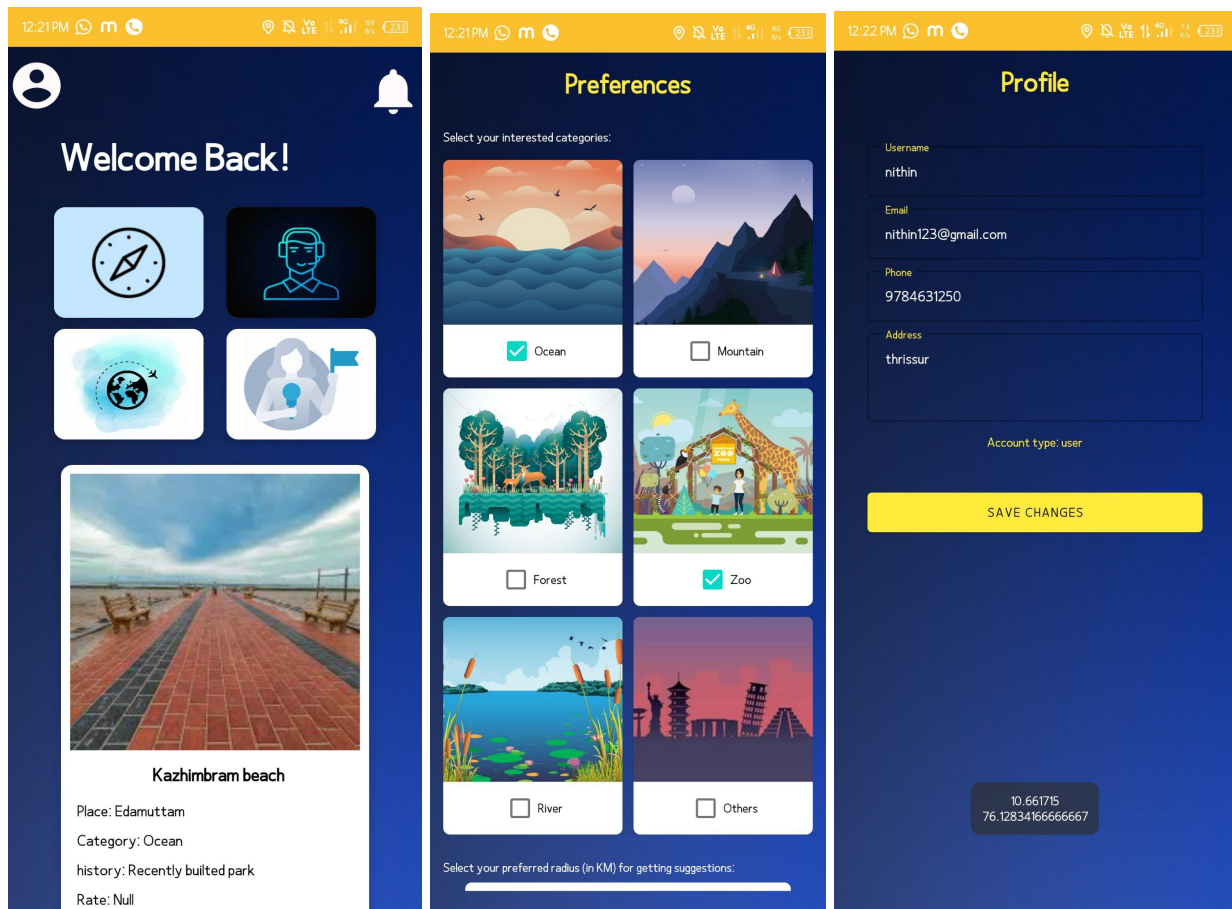
```

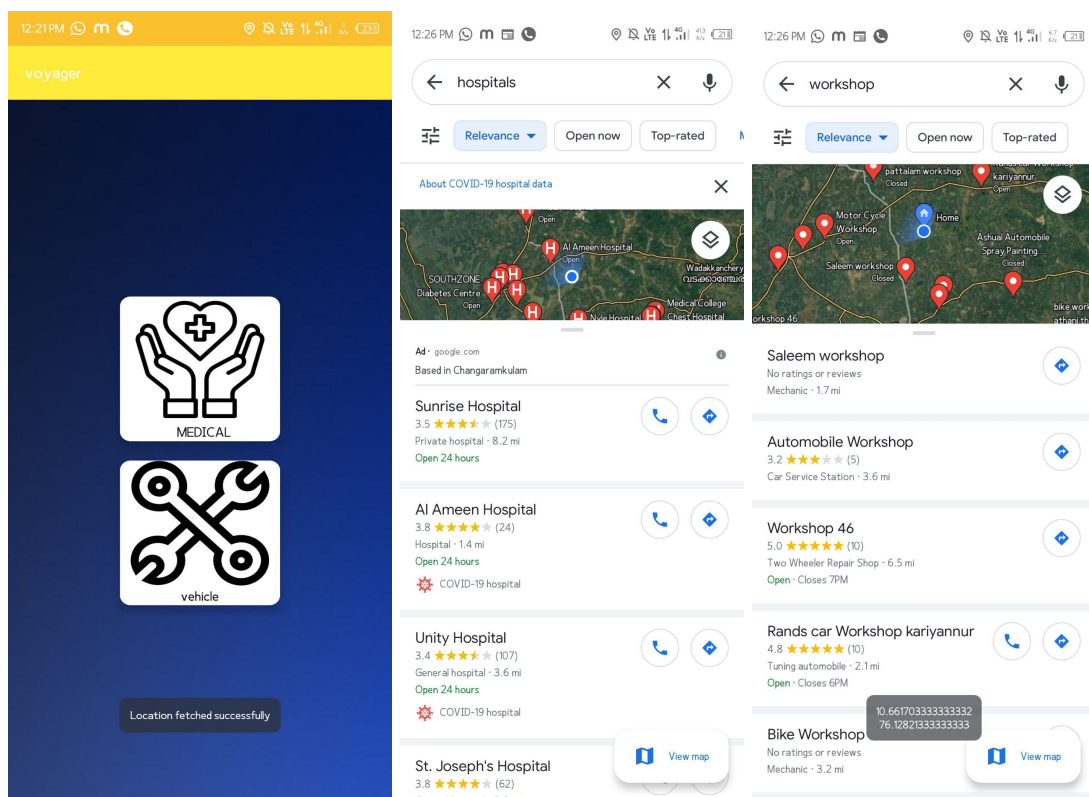
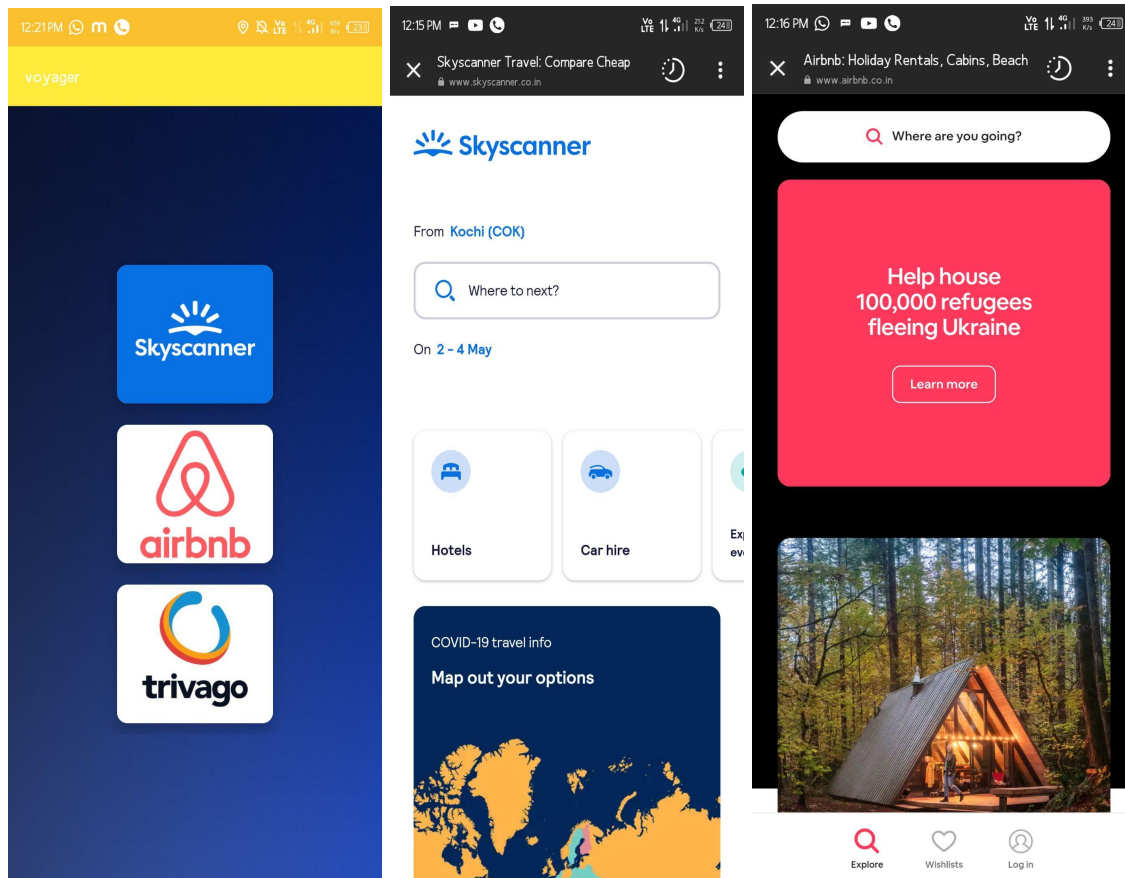
@Override

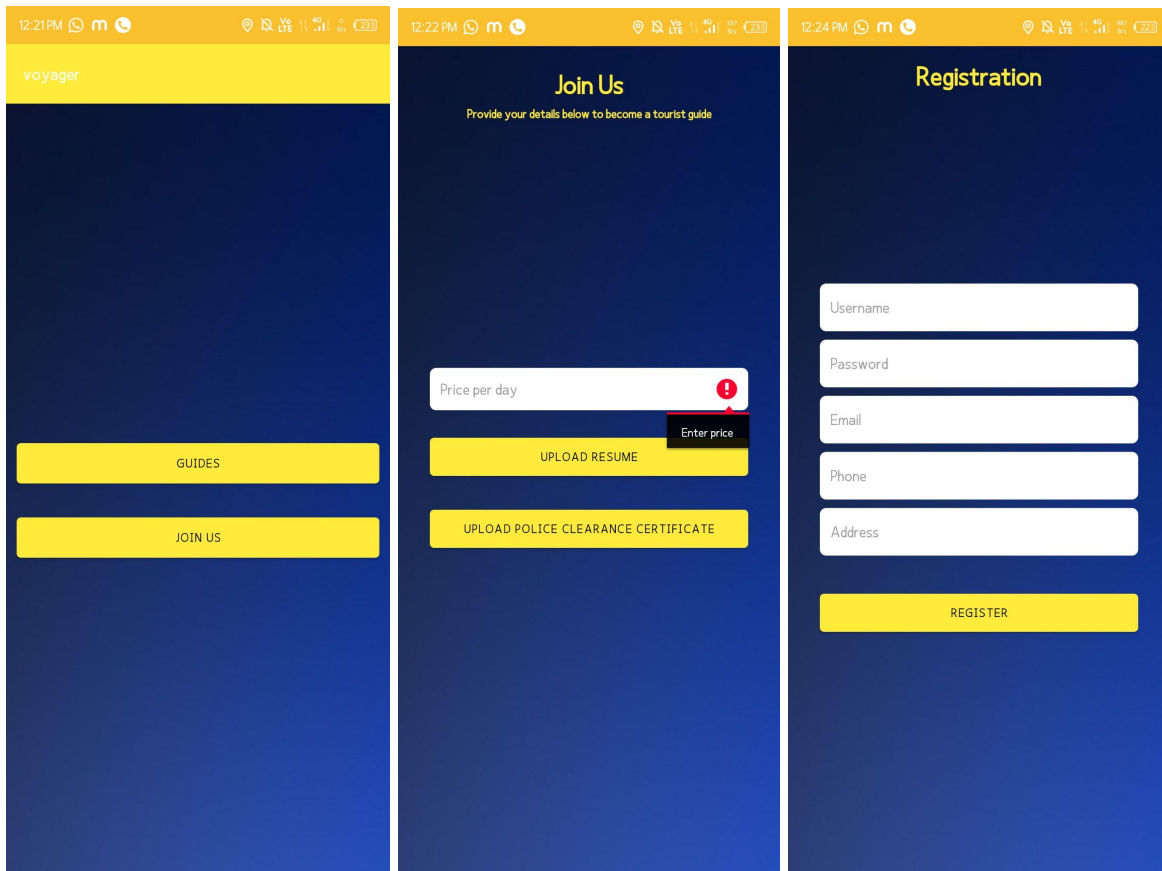
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_book_ticket);

    web = findViewById(R.id.web);
    WebSettings webSettings = web.getSettings();
    webSettings.setJavaScriptEnabled(true);
    web.loadUrl("https://www.trivago.in/");
    web.setWebViewClient(new WebViewClient());
}
}
```

5.5 Screen Shots







Chapter 6

Testing

6.1 Introduction

Testing is the process of checking whether the developed system is working according to the original objectives and requirements. Software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the project is not said to be complete. The system should be tested experimentally with test data so as to ensure that the system works according to the required specification. When the system is found working, test to with actual data and check performance. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to. the process of executing a program or application with the intent of finding software bugs (error or other defects).It can be stated as the process of validating and verifying that a computer program/application/product.

- Meets the requirements that guided its design and development,
- Works as expected.
- Can be implemented with the same characteristics.
- Satisfies the needs of stakeholders.

6.2 Testing Methodologies Adopted

6.2.1 Unit Testing

This is the first level of testing. A Unit Testing focuses verification effort on the smallest unit of software design. Here they test modules individually and integrate the overall system. Unit testing focused verification efforts even in the smallest unit of software design in each module. This is also known as “Module testing”.

This testing was carried out during the coding itself. In this testing step, each module is going to satisfactorily as expected output from the module. After coding each dialogue is tested and run individually. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence the naming is unit testing.

6.2.2 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. That is, integration testing is the complete testing of the set of modules, which makes up the products. The objectives are to take untested modules and build a program structure that has been dictated by design as integration testing is conducted, the tester should identify the critical modules.

One approach is to wait until all the units have passed the testing, and then combine them and then test. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination, and so on. The advantage of this approach is that interface dispenses can be easily found and corrected. Testing is completed when the last module is integrated and testing.

6.2.3 System testing

The process of performing a variety of tests on a system to explore functionality or to identify problems. System testing is usually required before and after a system is put in place. A series of system procedures are referred to while testing is being performed. These procedures tell the tester how the system should perform and where common mistakes may be found. Testers usually try to “break the system” by entering data that may cause the system to malfunction or return incorrect information. For example, a tester may put a city in a search engine designed to only accept states, to see how the system will respond to the incorrect input.

6.3 Test Plan & Test Cases

The primary objectives of test case design methods are to drive a set of tests that has the highest likelihood of uncovering the defects. To accomplish this objective, two categories of test case design techniques are used. Black box testing and White box testing.

Black box testing

Black box testing treats the software as a “black box”, examining functionality without any knowledge of internal implementation. The tester is only aware of what the software is supposed

to do, not how it does it. Black box test includes: partitioning analysis, all pairs testing state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

In black box type of testing, testing is performed by knowing the internal workings of a product, tests can be conducted to ensure that internal operations perform according to specifications and all internal components have been adequately exercised.

One advantage of the black box testing is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasise different areas of functionality. On the other hand, black box testing has been said to be “like a walk in a dark labyrinth without a flashlight” Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested. This method of test can be applied to all level software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing at higher levels, but can also dominate unit testing as well.

White Box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e., black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases.

The

tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

This is analogous to testing nodes in a circuit, e. g. in-circuit testing (ICT).

White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today.

It can test paths within a unit, paths between units during integration, and between subsystems during a system level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

6.3.1 Login Screen

SI No	Test Case	Expected result	Observed result	Pass/Fail
1	Without entering username and password, press the login button.	It should prompt message “Invalid entry”	Message is prompted	Pass
2	Enter the correct username and password. Click the login button.	The application should be loaded.	Application loaded without error.	Pass

6.3.2 Registration Screen

SI No	Test Case	Expected Result	Observed Result	Pass/Fail
1	Without entering the details, press the register button.	It should prompt message”Invalid entry”	Message is prompted.	Pass
2	Entering invalid password, press register.	It should prompt message”Please enter valid password”	Message is prompted.	Pass
3	Entering invalid phone number format, enter register.	It should prompt a message ”Please enter valid phone number”	Message is prompted.	Pass
4	Enter correct format of all	It should prompt the message	Message is prompted and is	Pass

	details and click register button	“Registered successfully” and it should be directed to the login screen.	directed to the login screen.	
--	-----------------------------------	--	-------------------------------	--

6.3.3 Home screen

Sl No	Test Case	Expected result	Observed result	Pass/Fail
1	Clicking on the profile button at the top-left corner	It should open the user’s profile screen and should show the user details and can also be updated if any changes are necessary	The profile screen is shown with details of the user and the changes made are updated successfully.	Pass
2	Clicking on the bell icon on the top-right corner	It should open the notification centre	The notification centre is opened and the alerts are shown	Pass
3	Clicking on the Preference image button	It should open the screen showing different types of places for the user to choose.	The preference screen is opened and user can select their preference within a range it is updated	Pass
4	Clicking on the Assistance image button	It should open a screen showing image buttons for medical and vehicle assistance and a user	The Assistance screen is opened and the user can click on the required buttons which redirects to the maps	Pass

		can select the required one.	showing the locations of nearby hospitals and workshop	
5	Clicking on the Booking image button	It should open a screen showing 3 image buttons for hotel bookings, flight bookings and package bookings and on clicking these buttons it should be redirected to the required sites.	The screen is opened with 3 image buttons for bookings and on clicking they get redirected to the required sites.	Pass
6	Clicking on Guide image button	It should open a screen showing 2 buttons, one for showing the nearby guides and the other to join as guides in our app.	The screen is opened and shows a “guides” button which on clicking provides the details of nearby guides and a “join us” button which opens a new screen for uploading resumes .	Pass

Chapter 7

Conclusion

The problem of depending on various apps and websites for a traveller in his journey is removed with our app. As the app is built with simple designs and features, it makes it more user friendly and easy to use. With the features provided, a user can completely rely on our app for almost all types of assistance in his journey and he/she doesn't have to install other apps or websites for it. Our app provides best booking sites, packages, guides, medical assistance, vehicle assistance and notifications / alerts to the users. As the size of the system is very less compared to other existing systems, it will be effectively working on low end devices smoothly which makes it accessible to all the users.

Future Work

As further updates we are planning to add a navigation map and also an e-commerce website for purchasing travelling equipments.

References

- Fundamentals of Software Engineering fourth edition by Rajib Mall
- Head First Android Development second edition by Dawn & David
- Professional Android fourth edition by Reto Meier & Ian Lake
- Android Studio Development fifth edition by Neil Smyth
- Software Testing by Gopalaswamy Ramesh & Srinivasan Desikan