# CHAPTER 1

# INTRODUCTION

## 1.1. Overview

"Tour Buddy" is a Java-based tour guide application by which the value of the travel experience will be increased through one-stop shops for picking the ideal travel itinerary replete with a comprehensive scheduling and messaging platform. The user will be able to log in and view their profile, through which they can manage their tour schedules. This application will provide basic tour application functionalities like map viewing, messaging, and profile management, and most importantly, provision to set preferences of the user details.

## 1.2. Problem Definition

The problem the "Tour Buddy" is solving is that there is no single application for the tour by which a tourist will manage his schedule and communicate with his appointed guide and the addition of maps for easier navigation. The traditional way includes too many applications and manual scheduling, which is redundant and error-prone.

## 1.3. Objectives

Develop an easy to use application for managing schedules of tours.

Develop an android application as communication between a tourist and his tour guide.

Feature the basic functionalities like maps and profile customization.

Provide tourists an integrated platform that can handle and manage their travel experience effectively.

## 1.4. Approach to be Adopted

Requirement Analysis: Collecting requirements and scoping the project at a high level

Design: Designing the application architecture and user interface.

Implementation: Coding of core functionalities and integration of all the modules

Testing: Testing at different levels, i.e., unit, integration and system

Deployment: Will deploy the application so that it is accessed by the end-users

Maintenance: Will update and improve at regular intervals with the user's feedback

## 1.5. Anticipated Results

A user-friendly interface for a tour guide program

Improved user experience in interacting with and managing tour schedules

Better communication between tourists and tour guides

Access to maps and user-profile customization

## 1.6. Hardware and Software Requirements

Hardware:

4GB RAM and 500GB hard drive

Computer

Android

Software:

Java Development Kit (JDK)

Android Studio

MySQL

Google Maps API

Firebase (authentication and messaging)

## CHAPTER 2

# FUNDAMENTALS OF JAVA

## 2.1. Java Introduction

Java is a high-level, class-based, object-oriented programming language designed for implementation dependencies to be as minimal as possible. It is a very popular language in writing secure, safe applications.

## 2.2. Java Benefits

Platform Independent: Java is compiled to be byte code, which can either be interpreted or be run in any, that is, by any computer with Java Virtual Machine (JVM).

Object-Oriented: Helps in being able to write modular programs and is also reusable.

Robust: Strong memory management and exception handling.

Security: It offers a secure platform for application development

Multithreading: It allows a concurrent execution of several threads

## 2.3. Data Types

Java provides eight primitive data types in programming that includes byte, short, int, long, float, double, char, and boolean. These primitive data types lays the foundation for data manipulation in Java

.

## 2.4. Control Flow

In Java, control flow is a part of the loops including for, while, do-while, the conditional statements such as if, if-else, switch, and also branching with the statements, break, continue, and return

.

## 2.5. Methods

Methods in Java consist of the block of codes for giving functionalities to execute a specific task and can be invoked to execute that task. Thus, methods in Java facilitate modularity and support the code reuse facility.

## 2.6. Object-Oriented Concepts

Classes and Objects: The basic building blocks of Java

Inheritance: The ability of one class to inherit the fields and methods of another

Polymorphism: It allows one single interface to be used to represent different underlying forms (method overloading and overriding).

Encapsulation: Wraps the data and the functions working on the data into a single unit.

Abstraction: OOP feature insulates the complex implementation details and shows only those features to the outside world which are necessary.

## 2.7. Exception Handling

Developed in Java, exception handling is easily

with the help of blocks comprising try, catch, and finally keywords and user-defined exceptions.

## 2.8. File Handling

It is one of the important parts of any programming language. Java also provides a set of classes to read/write file and manipulate the file and its content.

using input/output stream and file handling,

## 2.9. Packages and Import

Packages is used to group related classes and interfaces and the import statement is used to import other classes and interfaces in a class.

## 2.10. Interfaces

In Java, it represents the contract that a class should follow to implement it. It defines methods, which must be implemented.

## 2.11. Concurrency

Java supports concurrent programming under: Threads, Synchronized Method, Concurrent Collections.

# CHAPTER 3

# JAVA COLLECTIONS GUIDE

## 3.1. Java Collections Overview

It is a part of a Java Collections Framework, which includes interfaces and classes of collection that assist in the manipulation of groups of objects. Collections include lists, sets, and maps.

## 3.2. Role in Java Development

The java.util package gives several algorithms implementations so that a programmer can store or call back data in an efficient and effective way.

## 3.3. Advantages and Application

Efficiency: Performance is high, as well as consuming less memory.

Flexibility: It includes a variety of different data structures suitable for applying for different purposes.

Convenience: Lots of predefined methods used to provide the most common data manipulation operations.

## 3.4. Main Collection Interfaces

List: It is an interface for a sequence of objects and implies that the implementation is ordered.

Set: A group that does not contain duplicate elements ( e.g. HashSet, TreeSet).

Map: A group of key-value pairs ( e.g. HashMap, TreeMap).

## 3.5. Some of the Common Collection Implementations

ArrayList, LinkedList, HashSet, TreeSet, HashMap, TreeMap are some common collection implementation

## 3.6. Iterators and Collections API

The Iterators assist in traversing the collections. Collections API comprises utility methods for sorting, searching, and changing and manipulating collections.

## 3.7. Custom Collections and Generics

Collection interfaces are used to create a custom collection. The generics provide type-safety, and no need for typecasting will be required.

# CHAPTER 4

# DESIGN AND ARCHITECTURE

## 4.1. Design Goals

Usability: User interface that is easy to use.

Scalability: There can be an increment in the number of users and data.

Performance: Quick response and along with the economy of resources.

Security: Application data will be saved and secured with complete integrity.

## 4.2. Database Structure

The database holds all the user information, tour schedules, messages, and settings. The existing tables are Users, Tours, Messages, and Settings.

## 4.3. High-Level Architecture

It is a client-server architecture. The client provides the user interface, whereas the server offers the database and business logic.

## 4.4. Class Diagram

The class diagram illustrates the entity relationship between User, Tour, Message, and Settings classes.

# CHAPTER 5

# FUNDAMENTALS OF DBMS

## 5.1. Introduction

A Database Management System (DBMS) is software that uses a standard method to store and organize data, providing mechanisms for data retrieval, manipulation, and administration.

## 5.2. Characteristics of a DBMS

- **Data Abstraction:** Hides the complexity of the database system from users.
- **Data Independence:** Changes to the data structure do not affect the application program.
- **Concurrency Control:** Manages concurrent data access.
- **Recovery:** Restores the database to a correct state in case of failures.
- **Security:** Protects data from unauthorized access.

## 5.3. Data Model

Data models define the structure, manipulation, and integrity rules of the data stored in a database.

## 5.4. Three-Schema Architecture

Consists of the internal schema, conceptual schema, and external schema, providing a clear separation between the database's physical storage and user interactions.

## 5.5. DBMS Component Modules

- **Storage Manager:** Manages the storage of data.
- **Query Processor:** Interprets and executes database queries.
- **Transaction Manager:** Ensures database transactions are processed reliably.
- **DBMS Engine:** Core service that manages database operations.

## 5.6. SQL Commands

SQL (Structured Query Language) is used to interact with databases. Commands include SELECT, INSERT, UPDATE, DELETE, and more.

## 5.7. Data Definition Language

DDL commands are used to define and modify database structures, including CREATE, ALTER, and DROP statements.

# CHAPTER 6

# IMPLEMENTATION

## 6.1. Database Design

Developing the database schema, tables, relationships, and constraints to be used in the application.

## 6.2. Database Interface

JDBC implementation will be used for interfacing the database with the Java application.

## 6.3. Main Interface

The main interface from which the user will log in to and access the major part of the system functionalities.

## 6.4. Frames to be shown on Top of the Main Interface

The frames depict most of the parts of the application—for example, the user profile, tour schedule, and settings.

## 6.5. Query Execution

Executing SQL queries to fetch, insert, update, and delete data in the database.

## 6.6. Core Functionality

Implementation of core features of the application—that is, user authentication, tour scheduling, and messaging.

## 6.7. Handle Inputs from Users

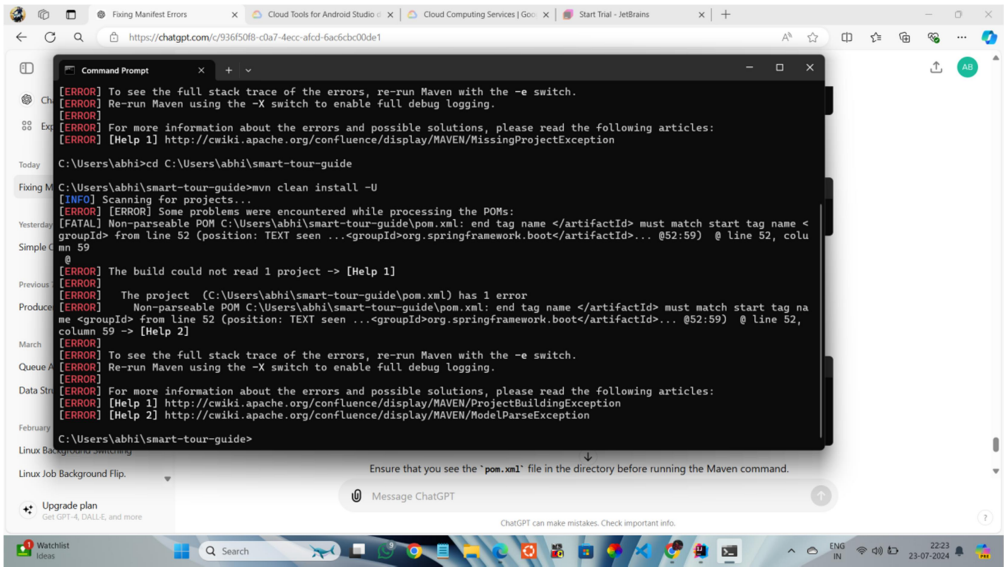
Process the inputs coming from the user by using forms and the validation of data before it gets to be stored in the database.

## 6.8. Error Handling and Validation

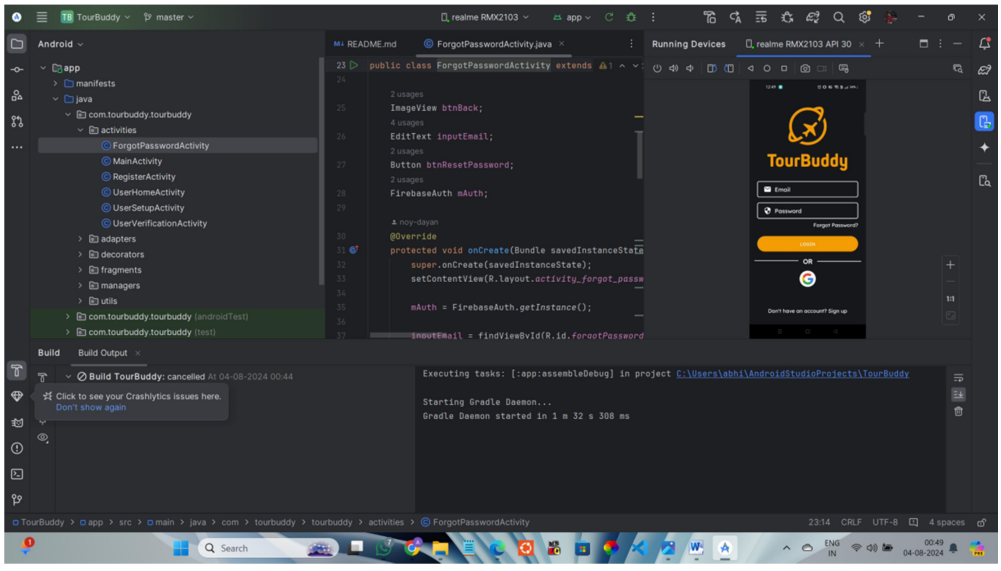
Implement a soundly constructed error handling and data validation system, and the application can operate efficiently and safely.

# CHAPTER 7

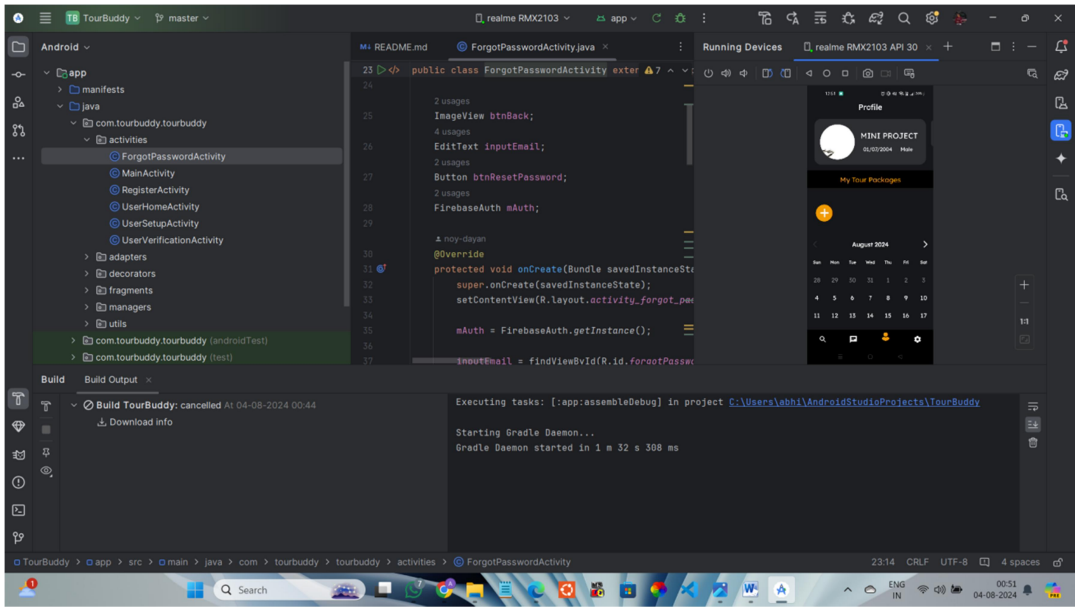
# TESTING



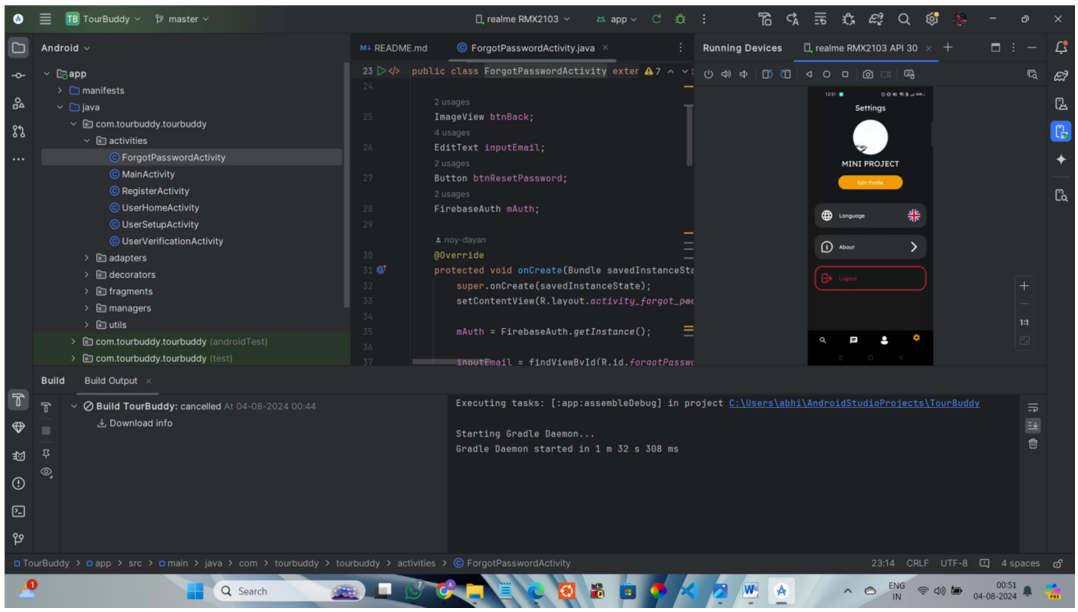

**(1.0. Configuring JDK In Android Studio)**



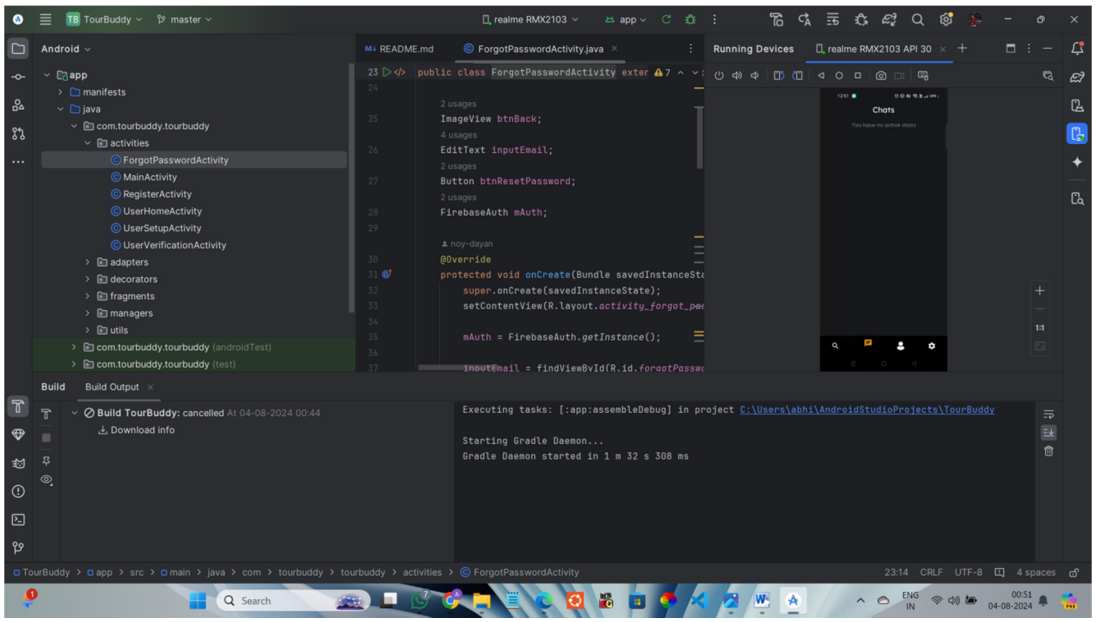

**(2.0. Testing The Application Front Page)**



**(3.0. Testing Front Page)**



**(4.0. Testing Settings Page)**




**(5.0. Testing Chats Options Page)**

- In he testing time we get lot of errors and we removed by debugging the program and tested.
- JDK latest version is required to run this program and we configured JDK with ANDROID STUDIO.
- In configuration time we get lot of complicated prompt to use. We cleared that using sources.

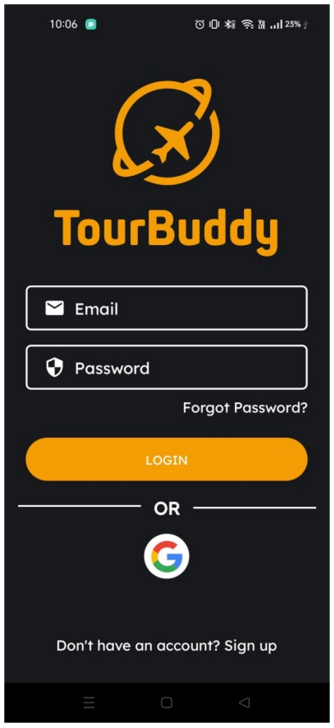
# CHAPTER 8

## RESULTS

After so much efforts we created an application for the topic of “SMART TOUR GUIDE APPLICATION”.

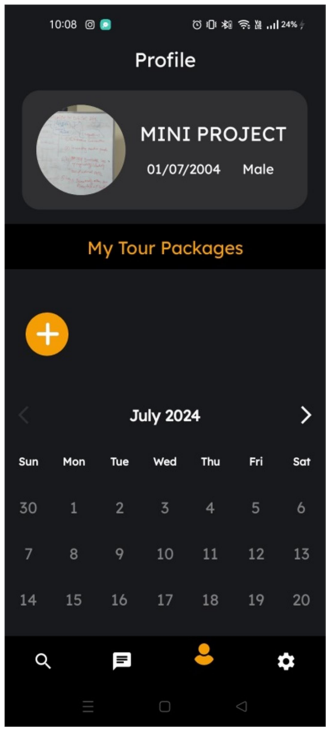
We given the name for this application as “TOUR BUDDY”

For user interface purpose we use “canva” for logo and “google” symbol for login page.



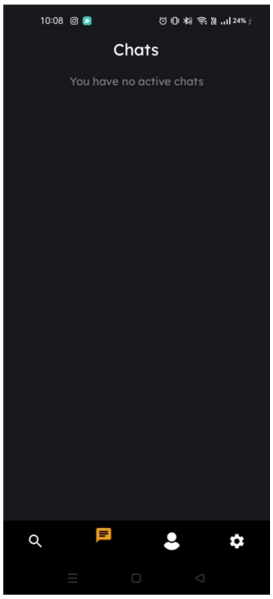

(fig.1 Front Page Of Application)


(fig.2 Second Page Of Application)




(fig.3 Third Page Of Application)

- When we open the real application we get this type of interface (fig.1).this is the login page. We can access the next page using email and password.
- When we login to the application if we are first time user, then we will get profile setting page (fig.2)to create our profile.
- After the creating of an profile then page automatically redirect to scheduling of an package of trip.(fig.3)and we can access our friends also for trip.




(fig.4 Fourth Page Of Application)

- In the fourth page of interface we can interact with our friends who came to the trip.
- We can also find random peoples who are going to that trip and we can also send a message also.
- To send a message you must have to send a request. If they accept, than only conversation will be start (Currently we are working for that).
- These all are about our “SMART TOUR GUIDE APPLICATION”.

## CHAPTER 9

# CONCLUSION

The "Tour Buddy" application effectively solves the challenges of visitors managing their itineraries, communicating with tour guides and accessing maps. By combining essential functions on one platform, the app enhance the user experience, and provide an easy and effective way to organize tours. Using Java for development ensures robustness, security and cross-platform compatibility. Application design prioritizes user friendliness and scalability, leading to future development and increased user adoption. Although currently only available in English, the application's configuration supports several possible languages. Overall, "Tour Buddy" stands as a comprehensive and reliable tool for today's travellers, promising continued improvement and feature expansion in future versions.

# REFERENCES


- USER INTERFACE: https://github.com/
- APPLICATION DEVELOPMENT : https://developer.android.com/studio
- MAP API : https://developers.google.com/maps
- JDK BY ORACLE: https://www.oracle.com/in/java/technologies/downloads/
- RUN THE APPLICATION : https://realme.com/devices
- DATABASE : https://cloud.google.com/sql/docs/mysql/connect-admin-ip