MOBILE APPLICATION FOR REAL ESTATE

A PROJECT REPORT

Submitted by

GUNASEELAN.S	**(20202020)**
SELVAMANI.S	**(20202048)**
VASEEKARAN.M	**(20202057)**

In partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

PAAVAI ENGINEERING COLLEGE, NAMAKKAL

(AUTONOMOUS)

ANNA UNIVERSITY: CHENNAI –600025

MAY 2024

ANNA UNIVERSITY: CHENNAI –600025

BONAFIDE CERTIFICATE

Certified that this project report titled **"MOBILE APPLICATION FOR REAL ESTATE"** is the bonafide work of **"GUNASEELAN.S (20202020), SELVAMANI.S (20202048)** and **VASEEKARAN.M (20202057)"** who have carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. B. VENKATESAN, M.E., Ph.D.,	**Mrs.M.PUSHPALATHA, M.E.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Associate Professor,	Associate Professor,
Department of Information Technology,	Department of Information Technology,
Paavai Engineering College,	Paavai Engineering College,
Namakkal - 637 018.	Namakkal - 637 018.

Submitted for End Semester Examinations held on ________________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We **"GUNASEELAN.S (20202020), SELVAMANI.S (20202048)** and **VASEEKARAN.M (20202057)"** hereby declare the project report titled "**MOBILE APPLICATION FOR REAL ESTATE**" done by us under the guidance of **Mrs.M.PUSHPALATHA, M.E.,** Assistant Professor at Paavai Engineering College, Pachal, Namakkal is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Information Technology.Certified further that, to the best of our knowledge, the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other occasion**.**

1.

2.

3.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

A great deal of arduous work and effort has been spent in implementing this project work. Several special people have guided us and have contributed significantly to this work so this becomes obligatory to record our thanks to them.

We express our profound gratitude to our honorable **Chairman, Shri.CA.N.V. NATARAJAN,B.com., FCA.,** and also to our **Correspondent Smt.N. MANGAINATARAJAN, M.Sc.,** for providing all necessary facilities for the successful completion of this project.

We wish to express our sincere thanks to our respected **Director of Administration, Dr.K.K.RAMASAMY, M.E., Ph.D.,** for all the blessing and help provided during the periodof project work.

We would like to thank our respected **Principal Dr. M. PREMKUMAR, M.E., Ph.D.,** for allowing us to do this project and providing the required time to complete the same.

We wish to express our sincere gratitude to **Dr. B. VENKATESAN, M.E., Ph.D., Headof the Department** for his extended encouragement to fulfill this project.

We express our sincere thanks to **Dr. G. MADASAMYRAJA, M.E, M.B.A, Ph.D.,** Project Coordinator for the useful suggestions, which helped us to complete the project work in time.

We would like to extend our sincere thanks to **Mrs.M. PUSHPALATHA, M.E.,** Supervisor, for giving us this opportunity to do this project and also for her inspiring guidance, generous help, and support.

We would like to extend our sincere thanks to **all our department staff members and to our parents** for their advice and encouragement to do the project work with full interest and enthusiasm.

ABSTRACT

The concept of mobile application development tailored for real estate tracking. The real estate market thrives on constant information access and effective communication. A mobile application can address these needs by providing a user-friendly platform for real estate professionals and clients a like. In today's dynamic real estate market, the integration of mobile technology has become paramount for agents, brokers, and clients alike. This paper explores the development of a mobile application tailored specifically for the real estate industry, aiming to streamline processes, enhance communication, and elevate user experience.

The mobile application will leverage cutting-edge technologies such as augmented reality (AR), geolocation services, and artificial intelligence (AI) to provide users with comprehensive tools for property search, viewing, and evaluation. Through intuitive interfaces and personalized features, the application will empower both agents and clients to efficiently navigate the complexities of buying, selling, or renting properties.

Keywords: Interactive Property Viewing, Data security, Seamless Property Search

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

In today's digital age, mobile application development plays a pivotal role in various domains, providing tailored solutions to meet the diverse needs of users. One such critical domain is the real estate industry, where mobile applications serve as indispensable tools for property search, evaluation, and transaction management. This introduction delves into the significance of mobile applications in real estate, highlights the challenges faced, and underscores the importance of developing a robust platform for property management.

1.2 SIGNIFICANCE OF REAL ESTATE APPLICATION

The real estate sector encompasses a broad spectrum of activities, from property buying, selling, and renting to investment and management. Mobile applications offer a convenient and efficient way for users to access property listings, conduct virtual tours, and communicate with agents and brokers. These applications provide comprehensive features such as property search filters, neighborhood information, mortgage calculators, and appointment scheduling, enhancing the overall user experience. By leveraging mobile technology, individuals can make informed decisions regarding property transactions, leading to greater efficiency and satisfaction in the real estate process.

1.3 CHALLENGES IN REAL ESTATE APPLICATION

Despite the advantages offered by real estate applications, several challenges exist in their development and implementation. One significant challenge is aggregating and updating property listings from various sources, including real estate agencies, developers, and individual sellers. Ensuring the accuracy and timeliness of these listings poses a considerable logistical hurdle. Additionally, integrating advanced features such as augmented reality for virtual property tours and predictive analytics for market trends requires sophisticated technological capabilities and seamless integration with existing platforms.

Furthermore, addressing data privacy and security concerns is paramount in real estate application development. Given the sensitive nature of personal and financial information involved in property transactions, robust security measures must be implemented to safeguard user data from unauthorized access and breaches.

1.4 IMPORTANCE OF REAL ESTATE APPLICATION

Developing a robust real estate platform entails more than just providing a digital marketplace for property listings. It requires a comprehensive approach that encompasses user-centric design, advanced technological solutions, and stringent security protocols. A well-developed platform should offer features such as: Seamless property search and filtering options, Interactive property viewing experiences, including virtual tours and 3D visualization. Real-time communication channels between agents, brokers, and clients. Integration with financial institutions for mortgage pre-approval and payment processing. Comprehensive analytics and reporting tools for market insights and performance tracking. Stringent data security measures, including encryption, authentication, and regular audits.

Mobile application development for the real estate industry holds immense potential in revolutionizing property transactions and enhancing user experiences. By addressing the challenges and leveraging innovative technologies, developers can create platforms that empower individuals to make informed decisions and navigate the complexities of the real estate market with confidence.

CHAPTER 2

LITERATURE REVIEW

2.1 TITLE: DEVELOPING A MOBILE APPLICATION FOR SMART REAL ESTATE INFORMATION

AUTHOR: R. BOVKIR

YEAR: 2023

This literature review Web-based and mobile GIS technologies provide the capability of operating and sharing local data and provide geographic analysis tools to users via the web. In this way, various mobile GIS applications can be developed in many different application areas. In this study, a mobile application titled as Smart Real Estate was developed for presenting urban real estate characteristics in different thematic groups by analyzing data in different formats coming from different sources.

2.2 TITLE: AN IMPROVED ANDROID-BASED REAL ESTATE APP

AUTHOR: ABDULLAHI ISA

YEAR: 2023

Android based applications are getting wider popularity and applicability across range of problem domains. However, literature investigation shows that most existing Real Estate Management Solutions are either web based or cloud based, and very few designed for Android platform. The existing Android based Real Estate Management Systems lack capacity to display property on map, and lack navigational support which could guide client to physical location of property. Consequently, this paper proposed and developed an improved Android based Real Estate Application, named AREA, for advertising/finding properties for sale or lease.

2.3 TITLE: REAL ESTATE VALUATIONS WITH SMALL DATASET: A NOVEL METHOD BASED ON THE MAXIMUM ENTROPY PRINCIPLE AND LAGRANGE MULTIPLIERS

AUTHOR: PIER FRANCESCO DE PAOLA

YEAR: 2022

Accuracy in property valuations is a fundamental element in the real estate market for making informed decisions and developing effective investment strategies. The complex dynamics of real estate markets, coupled with the high differentiation of properties, scarcity, and opaqueness of real estate data, underscore the importance of adopting advanced approaches to obtain accurate valuations, especially with small property samples. The objective of this study is to explore the applicability of the Maximum Entropy Principle to real estate valuations with the support of Lagrange multipliers, emphasizing how this methodology can significantly enhance valuation precision, particularly with a small real estate sample. The excellent results obtained suggest that the Maximum Entropy Principle with Lagrange multipliers can be successfully employed for real estate valuations. In the case study, the average prediction error for sales prices ranged from 5.12% to 6.91%, indicating a very high potential for its application in real estate valuations. Compared to other established methodologies, the Maximum Entropy Principle with Lagrange multipliers aims to be a valid alternative with superior advantages.

2.4 TITLE: APPLICATION DEVELOPMENT USING FLUTTER

AUTHOR: AAKANKSHA TAHSILDAR

YEAR: 2022

Cross-platform mobile application development is the pressing priority in today’s world and generation. Developers are enforced to either construct the same application numerous times for various OS(operating systems) or accept a lowquality similar solution that trades native speed and accuracy for portability. Flutter is an open-source SDK for developing high-performance and more reliable mobile applications for operating systems like iOS and Android. Significant features of the Flutter are Just-in-time.

2.5 TITLE: HYBRID DEVELOPMENT IN FLUTTER AND ITS WIDGETS.

AUTHOR: SWATI SHARMA

YEAR: 2021

The development of Cross-platform mobile applications is a goal of every client in today's world. Engineers are forced to build the same system multiple times for different OS (operating systems). Google provides a solution by introducing Flutter. It is an open-source SDK for improving high performance and the most reliable mobile apps for apps like iOS, Android, Linux, web and windows. It provides feature of timely compiling using a computer code that includes integration during the execution of the program in working time instead of previous practice. Flutter provides different frameworks, widgets that makes it easy to use and implement code. In this Research paper we are gone discuss about flutter and about its widgets.

CHAPTER 3
SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

In the existing system of mobile application development for real estate in Flutter, developers typically rely on a combination of various tools, frameworks, and methodologies to create functional and user-friendly applications. These applications serve as platforms for property buyers, sellers, and agents to interact, browse listings, and conduct transactions seamlessly. Currently, developers leverage Flutter's cross-platform capabilities to build apps that work seamlessly on both Android and iOS devices, reducing development time and costs.

The existing system often involves utilizing Flutter's extensive widget library to create visually appealing and responsive user interfaces tailored to the real estate industry's needs. Developers integrate features such as property search, filtering, and sorting to help users narrow down their options based on specific criteria like location, price range, property type, and amenities. Additionally, functionalities like map integration provide users with a visual representation of property locations, nearby amenities, and directions, enhancing the browsing experience.

Backend integration plays a crucial role in the existing system, with developers employing cloud-based solutions like Firebase Firestore for storing property data, user profiles, and app configurations. This allows for real-time updates, seamless synchronization across devices, and efficient data management. Furthermore, user authentication and authorization mechanisms are implemented to secure user data and ensure a safe and reliable experience for all users.

Overall, the existing system for mobile application development for real estate in Flutter emphasizes the importance of creating intuitive interfaces, robust backend infrastructure, and seamless user experiences. Developers continually explore new features, enhancements, and optimizations to meet the evolving needs of the real estate market and provide users with valuable tools for property discovery and transactions.

3.1.1 DISADVANTAGES

One potential disadvantage of mobile application development for real estate in Flutter is the limitation in accessing device-specific features. Since Flutter is a cross-platform framework, it abstracts away some of the platform-specific functionalities, which may restrict developers from fully leveraging the native capabilities of each operating system (iOS and Android).

Limited Access to Platform-Specific APIs: Flutter provides its own set of APIs for accessing device features like sensors, cameras, GPS, and Bluetooth. However, these APIs may not cover all the functionalities available on each platform. Developers may encounter challenges when trying to integrate advanced features or utilize platform-specific capabilities that are not directly supported by Flutter.

Performance Overhead: While Flutter boasts high-performance rendering with its Skia graphics engine, there may be some performance overhead compared to fully native applications, especially for complex UI animations or heavy computations. This overhead can impact the responsiveness and smoothness of the application, particularly on lower-end devices or in scenarios requiring intensive processing.

Plugin Dependency and Maintenance: Flutter relies on plugins to access platform-specific functionalities and native APIs. While there is a wide range of plugins available, developers may face issues with plugin compatibility, maintenance, and updates over time. Dependency on third-party plugins can introduce additional complexity and potential stability concerns to the development process.

Learning Curve: Although Flutter offers a simplified and streamlined development experience with its reactive framework and hot reload feature, there is still a learning curve, especially for developers transitioning from other frameworks or native development. Mastering Flutter's widget-based UI architecture and reactive programming paradigm may require additional time and effort.

Platform-Specific Design Guidelines: Both iOS and Android have their own design guidelines and UI/UX conventions. While Flutter allows for a consistent UI across platforms, developers need to carefully consider platform-specific design patterns and user expectations to ensure the application feels native on each platform. Straying too far from platform conventions may lead to usability issues and a lack of coherence with other apps on the device.

Dependency on Flutter's Ecosystem: Flutter's ecosystem is growing rapidly, but it may not be as mature or comprehensive as those of more established frameworks. Developers may encounter limitations or gaps in functionality when searching for specific tools, libraries, or resources to support their development needs.

Integration with Native Code: While Flutter supports platform channels for integrating with native code when necessary, this process adds complexity to the development workflow. Developers may need to write platform-specific code snippets or plugins in languages like Java/Kotlin for Android or Objective-C/Swift for iOS, which can introduce overhead and potential compatibility issues.

3.2 PROPOSED SYSTEM

In the proposed system for mobile application development for real estate in Flutter, we aim to create a comprehensive solution that revolutionizes the way users engage with property listings and real estate agents. Our system will leverage the power of Flutter's cross-platform development capabilities to deliver a seamless and intuitive user experience across both iOS and Android devices.

The key features of our proposed system include a robust property listing management module, enabling users to easily create, edit, and delete property listings with rich multimedia content such as images, videos, and detailed descriptions. We will prioritize user interface design to ensure that the app is visually appealing and easy to navigate, utilizing Flutter's extensive library of UI widgets and customizable design elements.

Backend integration will be a critical component, facilitating seamless data exchange between the mobile app and cloud-based databases for efficient storage and retrieval of property data, user profiles, and application settings. Search and filtering functionality will enable users to quickly find properties based on specific criteria such as location, price range, property type, and amenities.

Map integration will provide users with a visual representation of property locations, allowing them to explore nearby amenities and get directions to properties of interest. User authentication and authorization mechanisms will ensure secure access to the app's features, including user registration, login, and account management.

Additionally, features such as favorites and saved searches will enhance user engagement by allowing users to save their favorite properties and search criteria for future reference. Communication channels will facilitate direct communication between users and real estate agents, enabling inquiries, scheduling viewings, and receiving updates on property listings.

To ensure the success of the proposed system, we will prioritize continuous updates and maintenance, addressing user feedback, fixing bugs, and adding new features to enhance the app's functionality and user experience. With our proposed system, we aim to deliver a cutting-edge mobile application that sets new standards for the real estate industry, empowering users to make informed decisions and streamline their property search process.

3.2.1 ADVANTAGES

Cross-Platform Development: Flutter allows developers to write code once and deploy it on both iOS and Android platforms. This saves time and effort as developers don't need to maintain separate codebases for each platform, resulting in faster development cycles and reduced costs.

Fast Development: Flutter's hot reload feature enables developers to see changes in real-time as they code, speeding up the development process. This instant feedback loop allows for quick iteration and refinement of features, leading to faster time-to-market for real estate applications.

Beautiful UI Design: Flutter offers a rich set of customizable widgets and a flexible UI framework that allows developers to create stunning and highly responsive user interfaces. This is crucial for real estate applications, where visually appealing listings and property details can attract potential buyers or renters.

Native Performance: Flutter apps are compiled to native ARM code, resulting in high performance and smooth animations comparable to native applications. This ensures that real estate apps built with Flutter provide a seamless user experience.

Access to Native Features: Flutter provides plugins that give developers access to native device features such as camera, location, sensors, and more. This allows real estate apps to leverage features augmented reality (AR) viewings.

Open-Source and Community Support: Flutter is an open-source framework backed by Google, with a large and active community of developers. This means there are plenty of resources, tutorials, and third-party packages available to help developers overcome challenges and add new features to their real estate apps.

Single Codebase Maintenance: With Flutter, developers only need to maintain a single codebase for both iOS and Android platforms. This simplifies maintenance and updates, as changes can be applied uniformly across all platforms, ensuring consistency and reducing the risk of platform-specific bugs.

Cost-Effectiveness: By leveraging Flutter's cross-platform capabilities and fast development cycles, real estate businesses can save on development costs compared to building separate native apps for iOS and Android. This makes Flutter an attractive choice for startups and businesses with limited budgets.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 SYSTEM REQUIREMENTS

4.1.1 SOFTWARE SYSTEM CONFIGURATION

• Operating System	: Windows 11 or 10
• Programming	: Flutter SDK
• Database	: Firestore
• Back End	: Firebase
• Testing Tools	: Mockito
• IDE	: Android Studio

4.1.2 HARDWARE SYSTEM CONFIGURATION

• RAM	: 4 GB (MIN)
• Storage	: 20 GB
• Operating System	: Andriod

4.2 SOFTWARE SPECIFICATION

4.2.1 Flutter Language

Flutter is a popular choice for mobile application development in various domains, including real estate. Here's how Flutter can be leveraged for developing mobile applications for the real estate industry:

Cross-Platform Development: Flutter allows developers to write code once and deploy it across multiple platforms, including iOS and Android. This significantly reduces development time and effort, making it an efficient choice for real estate app development.

Rich User Interfaces: Flutter provides a rich set of customizable widgets that enable developers to create beautiful and intuitive user interfaces. This is essential for real estate apps, where users expect visually appealing listings and property details.

Fast Development Cycle: Flutter's hot reload feature allows developers to see changes in real-time as they modify the code.

Native Performance: Flutter apps are compiled to native ARM code, resulting in excellent performance and smooth animations. This ensures a high-quality user experience, which is crucial for real estate apps to showcase properties effectively.

Access to Device Features: Flutter provides plugins that allow developers to access device features such as GPS, camera, and sensors. This enables real estate apps to incorporate location-based services, property image capture, and augmented reality features.

Integration with Backend Services: Flutter apps can easily integrate with backend services using HTTP requests, REST APIs, or Firebase services. This allows real estate apps to fetch property data, user information, and other relevant data from servers.

Maps Integration: Flutter offers plugins for integrating interactive maps into mobile applications. Real estate apps can utilize maps to display property locations, nearby amenities, and neighborhood information to users.

State Management: Flutter provides various options for state management, including built-in solutions like setState, provider package, and state management architectures like BLoC (Business Logic Component) pattern. Effective state management ensures data consistency and enhances the overall app performance.

Customization and Branding: Flutter allows developers to customize every aspect of the app's design to match the branding and identity of the real estate company. This includes custom themes, fonts, colors, and animations.

Community Support and Resources: Flutter has a large and active community of developers, along with extensive documentation, tutorials, and libraries. Developers building real estate apps can leverage these resources to accelerate development and troubleshoot issues.

4.2.2 Firebase :

Firebase offers a comprehensive suite of tools and services that can greatly benefit mobile application development for real estate in Flutter. Here's how Firebase can be utilized:

Authentication: Firebase Authentication provides secure user authentication and authorization mechanisms, allowing users to sign in to the real estate application using email/password, phone number, or third-party providers like Google or Facebook. This ensures that user data remains protected and only accessible by authorized individuals.

Realtime Database or Firestore: Firebase Realtime Database or Firestore can serve as the backend database for storing property listings, user data, and application settings. These NoSQL databases offer real-time synchronization and offline support, enabling seamless data access and updates across devices.

Cloud Storage: Firebase Cloud Storage allows for the storage and retrieval of property images and other media files associated with property listings. This ensures that high-quality images are accessible to users and improves the visual appeal of the application.

Cloud Functions: Firebase Cloud Functions can be used to implement server-side logic, such as sending notifications to users for property updates, triggering data validation and processing, or integrating with third-party APIs for additional functionality.

Analytics: Firebase Analytics provides insights into user engagement, retention, and app usage patterns. By tracking user interactions within the real estate application, developers can gain valuable insights to optimize user experience, identify popular properties, and tailor marketing strategies.

Remote Config: Firebase Remote Config allows developers to dynamically adjust app configurations, such as UI layouts, property search filters, or promotional messages, without requiring a new app release. This enables A/B testing, personalized experiences, and rapid iteration based on user feedback.

Cloud Messaging: Firebase Cloud Messaging enables targeted push notifications to notify users about new property listings, updates on favorite properties, or important announcements. This helps keep users engaged and informed, leading to higher user retention rates.

App Indexing: Firebase App Indexing helps improve the discoverability of the real estate application by enabling deep linking from search engine results. This allows users to directly access relevant property listings from search results, increasing organic traffic and user acquisition.

Performance Monitoring: Firebase Performance Monitoring allows developers to monitor app performance metrics, such as app startup time, screen rendering, and network latency. By identifying performance bottlenecks, developers can optimize app performance and deliver a smoother user experience.

AdMob: Firebase AdMob integration enables developers to monetize the real estate application by displaying targeted ads to users. This can generate additional revenue streams and offset development costs while ensuring a non-intrusive.

4.2.3 Android Studio:

Android Studio is a powerful integrated development environment (IDE) commonly used for mobile application development, including projects built with Flutter for real estate applications. Here's how you can leverage Android Studio for developing real estate mobile applications in Flutter:

Project Setup: Begin by setting up a new Flutter project in Android Studio. Use the Flutter plugin to create a new Flutter project or open an existing one.

UI Design: Android Studio provides a rich set of tools for designing user interfaces. Utilize the built-in layout editor to design screens for property listings, property details, user profiles, and other app features. You can drag and drop UI components, adjust layouts, and preview your designs in real-time.

Code Editing: Android Studio offers powerful code editing features for writing Flutter code. Take advantage of features such as code completion, syntax highlighting, and code refactoring to write clean and efficient Dart code for your real estate application.

Emulator and Device Testing: Android Studio includes an emulator that allows you to test your Flutter app on various virtual devices with different screen sizes and Android versions. Additionally, you can connect physical Android devices for testing and debugging directly from Android Studio.

Flutter Inspector: The Flutter Inspector tool in Android Studio enables you to inspect and debug the layout and state of Flutter UI widgets. Use it to identify layout issues, view widget properties, and debug UI-related problems in your real estate app.

Integration with Firebase: Android Studio seamlessly integrates with Firebase, Google's mobile platform for app development. You can use Firebase services such as Firestore for storing property data, Firebase Authentication for user authentication, and Firebase Cloud Messaging for push notifications.

Version Control: Android Studio includes built-in support for version control systems such as Git. You can easily manage your project's codebase, track changes, and collaborate with team members using version control features integrated into the IDE.

Performance Profiling: Android Studio offers performance profiling tools for analyzing your app's performance and identifying areas for optimization. Use these tools to monitor CPU, memory, and network usage, and optimize.

Continuous Integration and Deployment: Android Studio integrates with popular continuous integration and deployment (CI/CD) platforms such as Firebase App Distribution and Google Play Console. You can automate the build, testing, and deployment process of your real estate app directly from Android Studio.

Community Support: Android Studio has a large and active community of developers and Flutter enthusiasts. Take advantage of online forums, tutorials, and resources to learn new techniques, troubleshoot issues, and stay updated on the latest trends and best practices in Flutter app development for real estate.

CHAPTER 5
SYSTEM ARCHITECTURE

5.1 INTRODUCTION

The software system comprises various interconnected components, each serving a specific purpose in ensuring the functionality, performance, and usability of the application. Let's expand on each element:

Presentation: This component encompasses the user interface elements responsible for rendering visual components and facilitating user interaction. It includes widgets, screens, buttons, forms, navigation menus, and other UI elements designed to enhance user experience and usability.

Logic Holders: Logic holders encapsulate the business logic and rules governing the behavior and operations of the application. This may include validation logic, calculation algorithms, decision-making processes, and other core functionalities essential for the proper functioning of the system.

Domain: The domain refers to the specific problem domain or application area that the software system addresses. It encompasses the concepts, entities, and rules relevant to the application's purpose, such as real estate management, financial transactions, healthcare, or e-commerce.

Use Cases and Call Flow: Use cases represent the various functionalities and interactions that users can perform within the system. They outline specific scenarios and actions users can take to achieve their goals. Call flow refers to the sequence of steps or operations executed during the execution of a use case, depicting the flow of control and data within the system.

Entities: Entities represent the core data objects or business entities within the application's domain. These entities encapsulate the data attributes and behaviors relevant to the application, such as properties, users, transactions, or products.

Repositories: Repositories serve as intermediaries between the application's data sources and the rest of the system. They abstract the data access logic and provide a unified interface for fetching, storing, and manipulating data from various sources, such as databases, APIs, or external services.

Data Models: Data models define the structure and relationships of the data stored within the application. They represent the schema or blueprint for organizing and managing data entities, including attributes, relationships, constraints, and data integrity rules.

Remote and Local Data Sources: Remote data sources refer to external data repositories accessed over a network, such as web servers, cloud databases, or third-party APIs. Local data sources, on the other hand, involve data stored locally on the user's device, such as device storage, SQLite databases, or shared preferences.

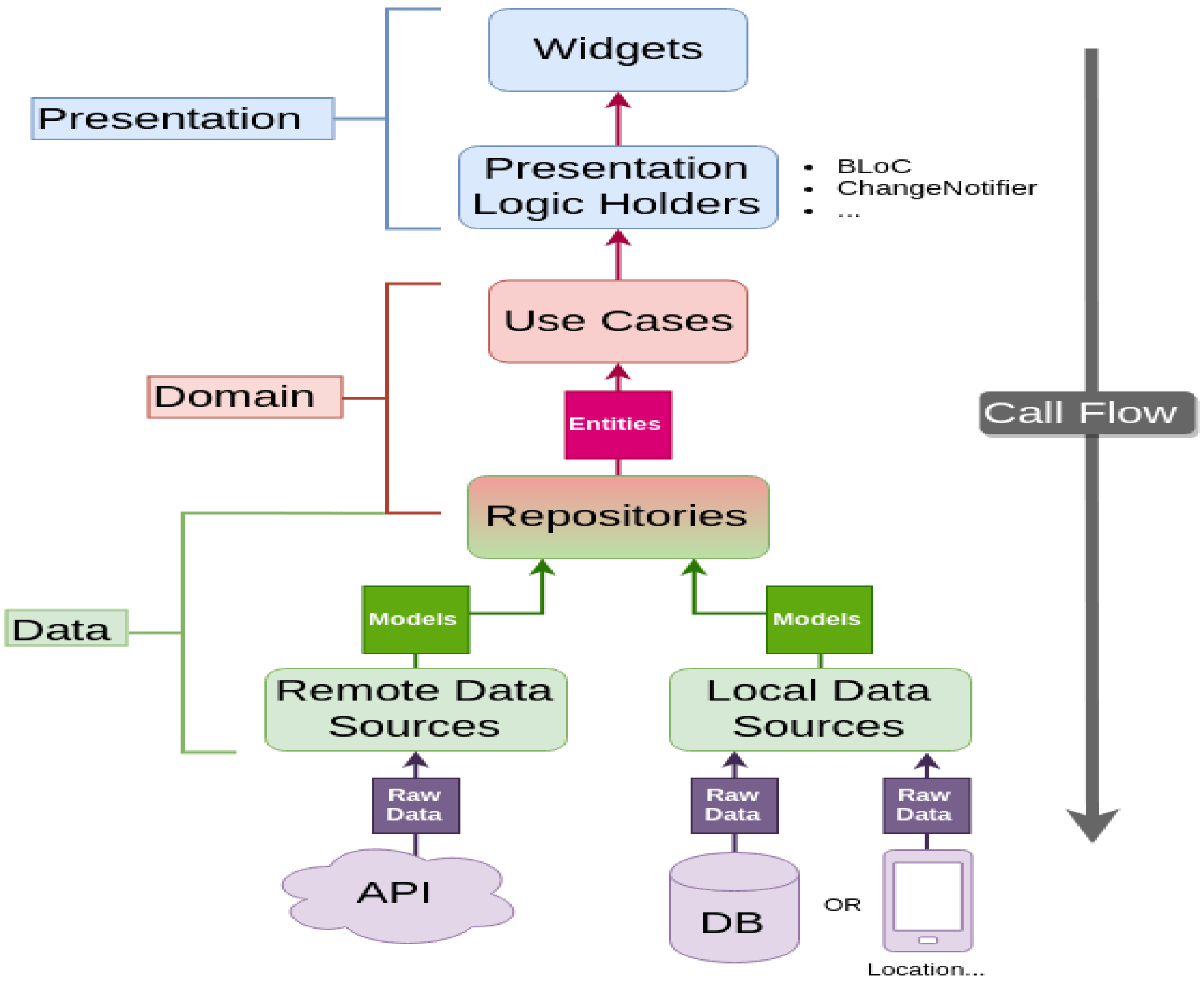


Fig 1 System Architecture

5.2 DESCRIPTION

Represents a comprehensive diagram of a software system architecture, specifically a data-driven application. The diagram illustrates various components and their interconnections, which can be categorized into several main sections: Presentation, Logic Holders, Domain, Use Cases, Call Flow, Entities, Repositories, Data Models, and Data Sources.

The Presentation section encompasses the user interface and interaction elements of the software system. This category includes Widgets, which are visual components that users interact with, such as buttons, text fields, or sliders. The Presentation Logic Holders may involve components responsible for managing the presentation logic, ensuring that the user interface behaves as expected and responds appropriately to user input.

The Domain section likely refers to the specific problem domain or application area that the software system is designed to address. This area may include various concepts, entities, and relationships unique to the problem domain.

The Use Cases and Call Flow sections represent the functionality and interactions within the software system. Use Cases outline specific scenarios or tasks that the system can perform, while Call Flow illustrates the sequence of interactions and events that occur during these scenarios.

Entities, Repositories, and Data Models are responsible for managing data storage and manipulation within the software system. Entities represent real-world objects or concepts relevant to the problem domain. Repositories serve as interfaces for accessing and manipulating data, while Data Models define the structure and organization of data within the system.

Remote and Local Data Sources, as well as Raw Data, are related to data input and retrieval from various sources. Remote Data Sources may involve external systems, services, or databases that the software system interacts with, while Local Data Sources might include internal databases or files. Raw Data represents the initial, unprocessed form of data before it is structured and organized according to the Data.

The API DB (Atomicity, Repeatability, Isolation, Durability Database) is a specific database system or a concept in database management systems that ensures data consistency, reliability, and performance. The API acronym stands for Atomicity, Repeatability, Isolation, and Durability, which are essential properties of database transactions.

In summary, this diagram offers a detailed view of a software system's architecture, highlighting the components and relationships necessary for data-driven applications. By understanding these elements and their interactions, developers can create efficient, maintainable, and scalable software systems tailored to specific problem domains and user needs.

CHAPTER 6

MODULE IMPLEMENTATION

6.1 Authentication Module

User Registration

Implementation

- Collect user information such as name, email, and password.
- Validate input data to ensure correctness.
- Utilize Firebase Authentication or custom backend service for user registration.

User Login

Implementation

- Provide a login screen with fields for email and password.
- Authenticate user credentials against the stored data in the database.
- Utilize Firebase Authentication or custom backend service for user login.

Social Authentication

Implementation

- Integrate social media SDKs such as Google Sign-In, Facebook Login, or Twitter Sign-In.
- Handle the authentication flow provided by the respective social media platforms.
- Link social accounts to user profiles for seamless login experiences.

Forgot Password

Implementation

- Offer a "Forgot Password" screen where users can input their email address.
- Send a password reset email containing a unique link to the user's email address.
- Handle the password reset flow using Firebase Authentication or custom backend service.

User Profile Management

Implementation

- Display user profile details such as name, email, and profile picture.
- Provide options to edit profile information, including updating email or password.
- Utilize Firebase Authentication or custom backend service to manage user profile data.

Authentication State Management

Implementation

- Use state management techniques such as Provider, Bloc, or Redux to handle authentication state.
- Maintain the user's authentication status (logged in or logged out) across app sessions.
- Redirect users to appropriate screens based on their authentication status.

Access Control

Implementation

- Implement middleware or route guards to restrict access to certain screens or actions.
- Allow only authenticated users to perform sensitive operations such as posting listings or accessing personal data.
- Provide error handling and feedback for unauthorized access attempts.

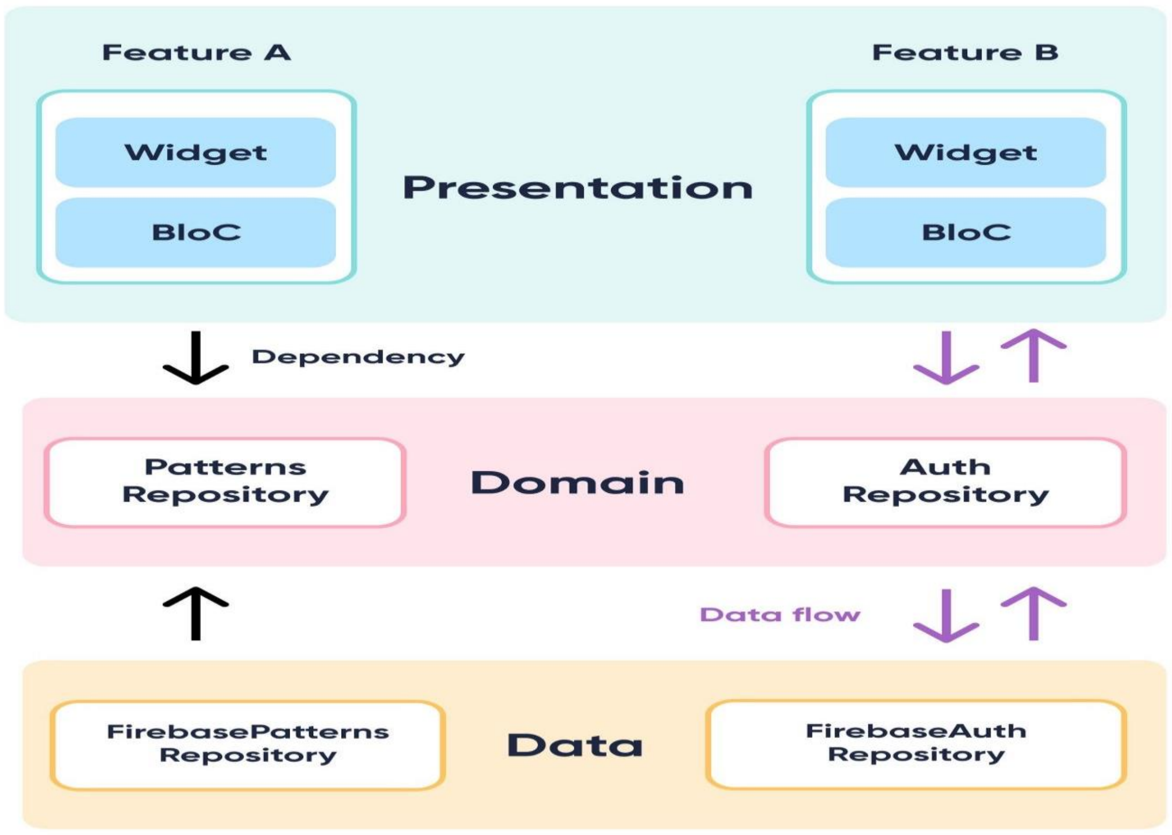


Fig 2 Authentication Module

6.2 Property list Module

Module Overview

The property listing module is a crucial component of a real estate mobile application, allowing users to browse and explore available properties.It provides a user-friendly interface for viewing property details, including images, descriptions, prices, and additional features.

Property List Screen

The module starts with a Property List screen, displaying a list of available properties in a visually appealing format.Each property item in the list shows a thumbnail image, property details (e.g., price, location, number of bedrooms), and a brief description.

Property Details Screen

Tapping on a property item navigates the user to the Property Details screen, which provides comprehensive information about the selected property.This screen displays highresolution images of the property, a detailed description, price, location map, property type, amenities, and contact information.

Search and Filter Functionality

The module includes search and filter functionality to help users find properties that meet their specific requirements.Users can search for properties by location, price range, property type, number of bedrooms, and other relevant criteria.

Integration with Backend Services

The module integrates with backend services, typically a cloud-based database like Firestore or a custom API, to fetch property data.Real-time synchronization ensures that users see up-to-date property listings and details.

User Interactions

The module supports various user interactions such as favoriting properties, contacting property agents, and sharing property listings with others.Users can add properties to their favorites list for future reference and easily contact property agents for inquiries or to schedule viewings.

Responsive Design

The module employs responsive design principles to ensure a consistent and optimal user experience across different screen sizes and orientations. Layouts are designed to adapt gracefully to various device sizes, from smartphones to tablets, providing an intuitive user interface on any device.

6.3 User Profile Module

Profile Page UI

Design a visually appealing profile page UI where users can view and manage their personal information. Include profile picture, username, contact details, and any other relevant information. Provide options to edit profile details, change password, and manage notification preferences.

Authentication Integration

Implement user authentication using Firebase Authentication or any other authentication service. Allow users to sign up, log in, and log out securely. Ensure proper error handling for authentication failures and provide appropriate feedback to users.

Profile Data Storage

Store user profile data in a Fire store database or any other suitable database solution. Define data models to represent user profiles and manage CRUD operations for profile data.

Profile Editing Functionality

Implement forms or dialogs for users to edit their profile information. Handle input validation and ensure data consistency before updating user profiles. Provide feedback to users upon successful profile updates and handle any errors gracefully.

Profile Picture Management

Allow users to upload, change, or remove their profile pictures. Implement image uploading functionality with appropriate validation and error handling. Store profile pictures securely in cloud storage (e.g., Firebase Storage) and manage access permissions.

Privacy and Security

Implement privacy settings to allow users to control the visibility of their profile information. Ensure that sensitive information is handled securely and only accessible to authorized users. Implement security measures such as two-factor authentication (2FA) for enhanced account protection.

Integration with Other Modules

Integrate the user profile module with other modules of the real estate application, such as property listings, saved searches, and favorites. Enable seamless navigation between the user profile and other sections of the app for a smooth user experience.

Localization and Accessibility

Support localization to provide the user interface in multiple languages and cater to a diverse user base. Ensure accessibility by following Flutter's accessibility guidelines and making the profile module usable for users with disabilities.

Testing and Quality Assurance

Conduct thorough testing of the user profile module to identify and fix any bugs or issues. Perform usability testing to ensure that the profile UI is intuitive and easy to navigate. Implement automated tests for critical functionalities to maintain code quality and reliability.

Documentation and Support

Document the implementation details, API usage, and configuration steps for the user profile module. Provide user support and assistance through in-app help sections, FAQs, or customer support channels to address user queries and issues effectively.

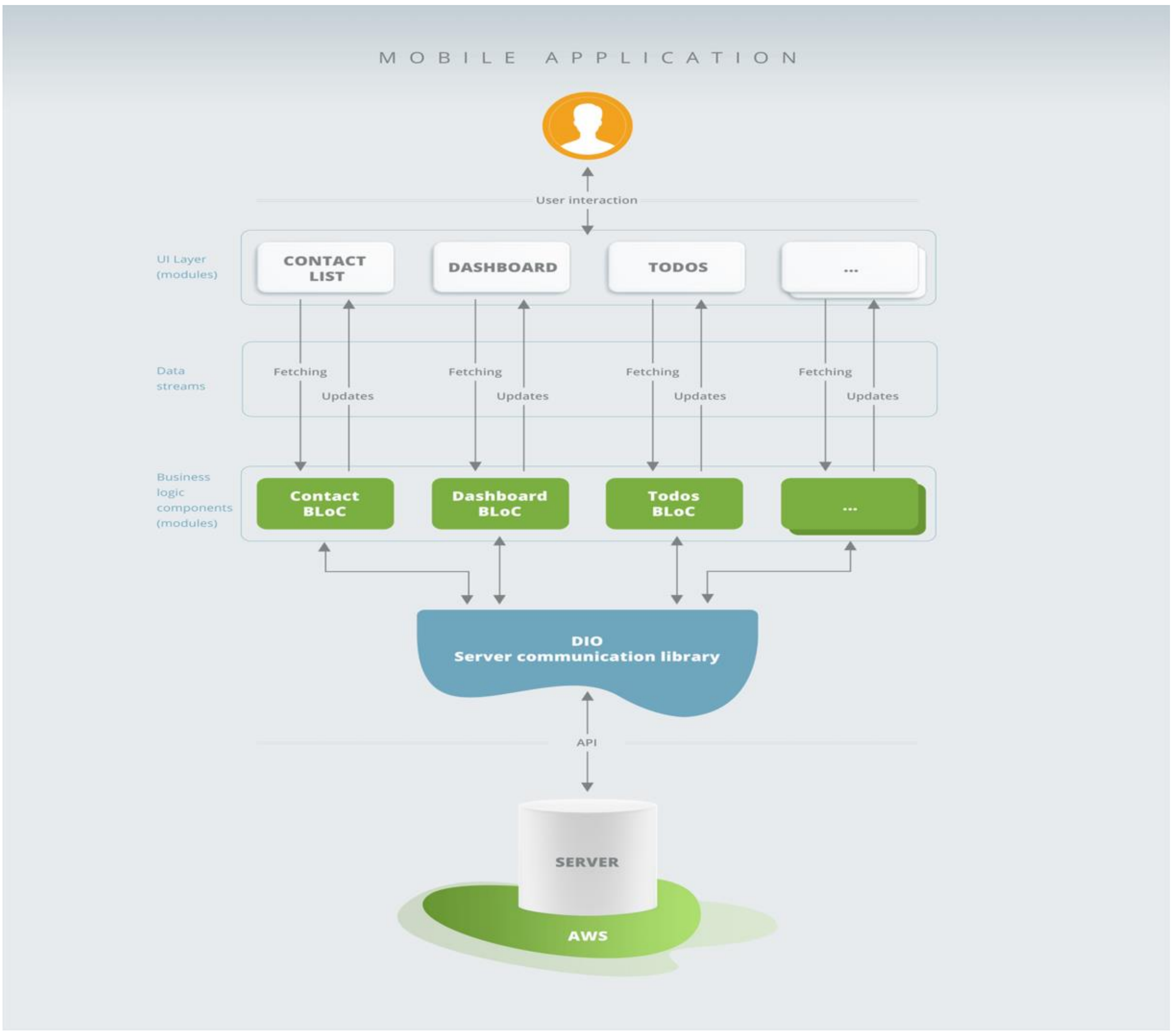


Fig 3 User Module

6.4 PROPERTY DETAILS MODULE

Design UI Layout: Begin by designing the user interface (UI) layout for the property listing page. This layout typically includes components such as a list view or grid view to display property listings, along with any filters, search bars, or sorting options for users to refine their search.

Create Property Model: Define a data model to represent a property listing. This model should include attributes such as property type, location, price, number of bedrooms/bathrooms, amenities, description, and images.

Fetch Property Data: Implement a mechanism to fetch property data from a data source, such as a local database or an API. This could involve making HTTP requests to a backend server or querying a local database to retrieve property listings.

Populate UI with Data: Once the property data is fetched, populate the UI components (list view or grid view) with the retrieved property listings. Map each property listing to its corresponding UI representation, displaying relevant information such as property images, title, location, price, and other details.

Handle User Interactions: Implement functionality to handle user interactions on the property listing page. This may include allowing users to tap on a property listing to view more details, swipe to refresh the property list, or interact with filters/search bars to refine their search criteria.

Navigation: Set up navigation to allow users to navigate to a detailed view of each property listing. When a user taps on a property listing, navigate to a new screen or page displaying detailed information about the selected property.

Add Filtering and Sorting: Enhance the user experience by adding filtering and sorting options to the property listing page. Allow users to filter property listings based on criteria such as property type, location, price range, number of bedrooms/bathrooms, and amenities.

CHAPTER 7

SYSTEM DESIGN

7.1 UML DIAGRAM

The system design for our mental health care chatbot platform is visualized through various Unified Modeling Language (UML) diagrams, including the Use Case Diagram, Class Diagram, Sequence Diagram, and Deployment Diagram. These diagrams collectively provide a comprehensive overview of the platform's structure, functionality, and deployment architecture.

7.1.1 USECASE DIAGRAM

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

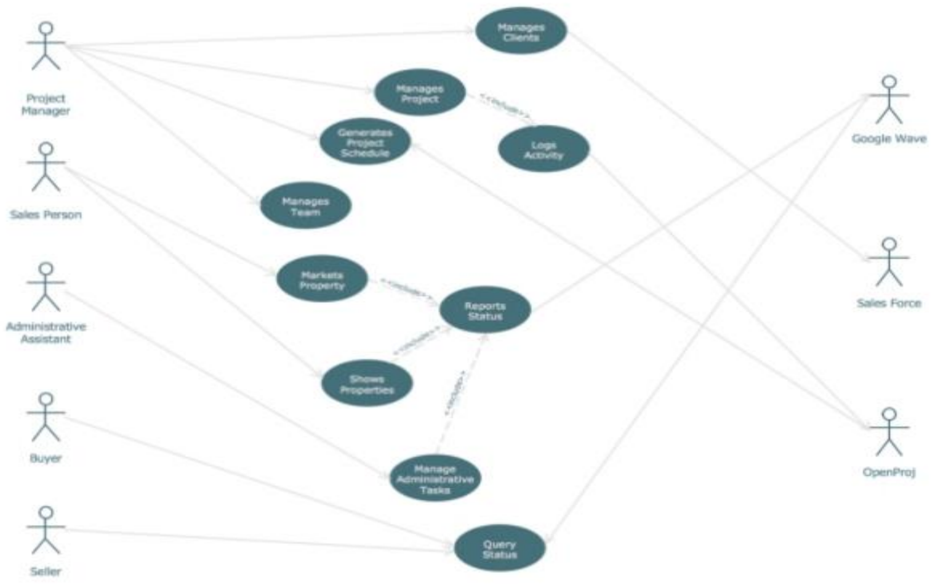


Fig 4 Use Case Diagram

7.1.2 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code.

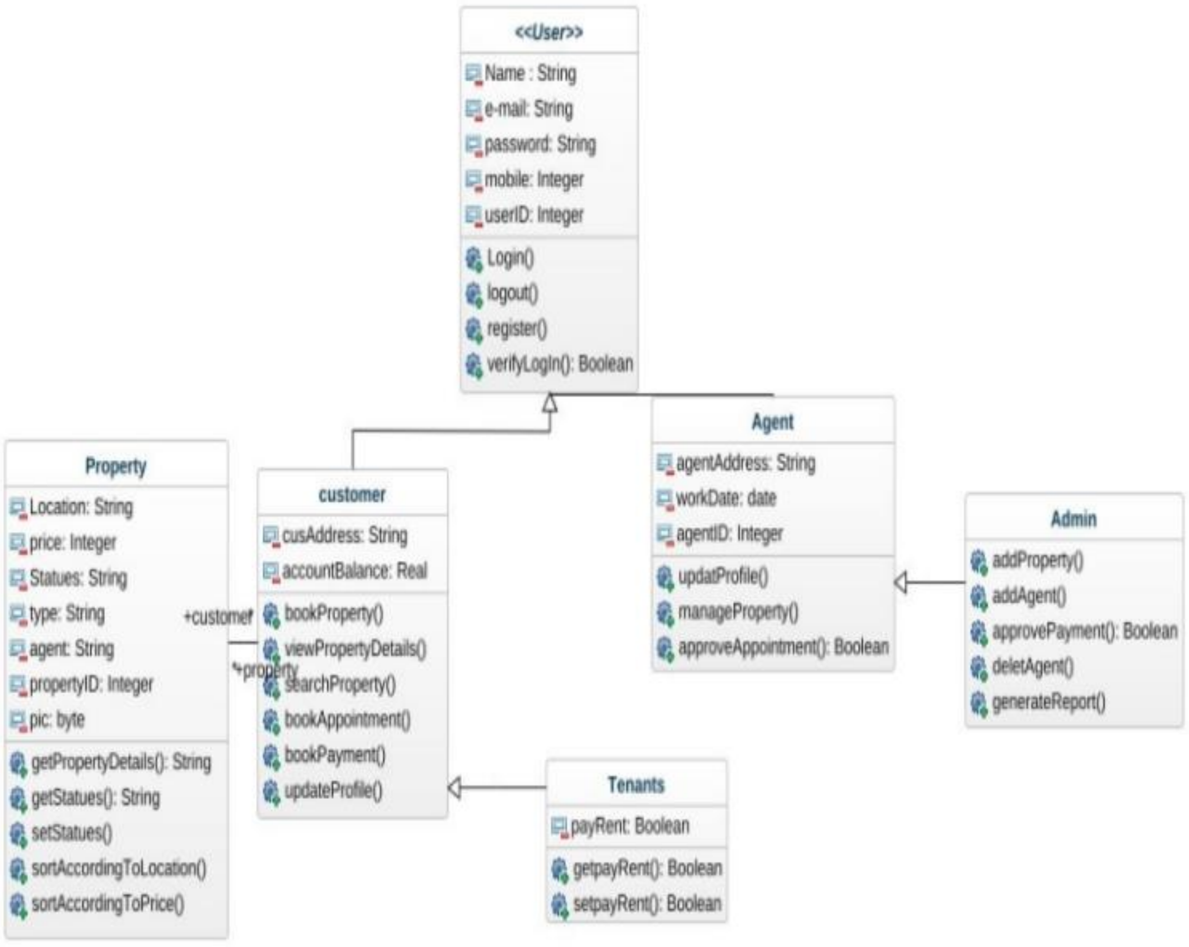


Fig 5 Class Diagram

7.1.3 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

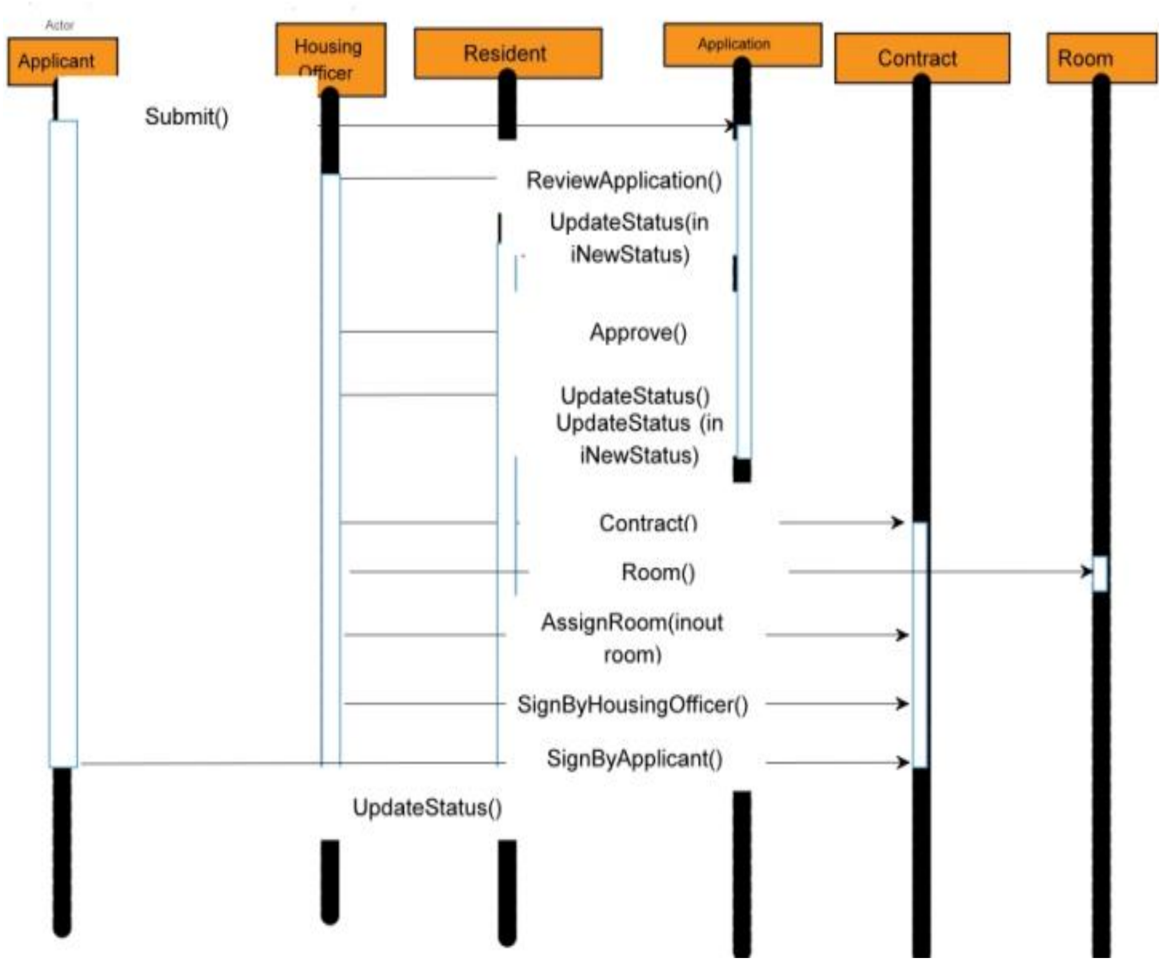


Fig 6 Sequence Diagram

7.1.4 DEPLOYMENT DIAGRAM

A deployment diagram shows components and artifacts in relation to where they are used in the deployed system. A component diagram defines the composition of components and artifacts in the system. A deployment diagram is just a special kind of class diagram, which focuses on a system's nodes. Graphically, a deployment diagram is a collection of vertices and arcs.

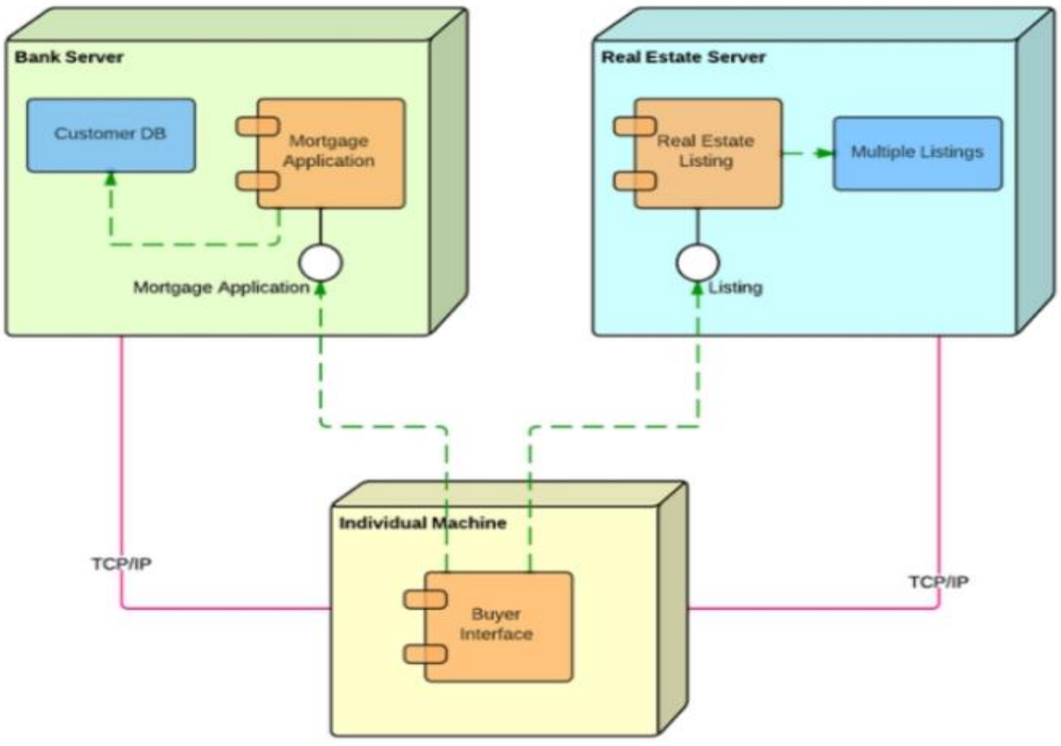


Fig 7 Deployment Diagram

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 CONCLUSION

In conclusion, the development of a mobile application for real estate in Flutter presents a transformative opportunity to revolutionize the way individuals interact with the real estate market. By leveraging the versatility and power of Flutter, coupled with robust backend services, the application aims to streamline the process of buying, selling, and renting properties. Throughout the development journey, a focus on user experience, accessibility, and functionality has been paramount, ensuring that the application meets the diverse needs and preferences of its users. With features such as property listings, search filters, interactive maps, and secure payment gateways, the application empowers users to navigate the real estate landscape with ease and confidence. Moreover, the incorporation of advanced technologies like augmented reality (AR) and virtual tours enhances the immersive experience, allowing users to visualize properties in stunning detail from the comfort of their devices. As the real estate industry continues to evolve and embrace digital innovation, this mobile application stands as a testament to the potential of Flutter in driving meaningful change and reshaping traditional paradigms in the real estate sector.

8.2 FUTURE ENHANCEMENTS

For future enhancement and scalability of a mobile application developed for real estate in Flutter, several avenues can be explored to improve user experience, expand functionality, and accommodate growing demands

Advanced Property Search: Enhance property search capabilities by integrating advanced filters, such as property size, number of bedrooms/bathrooms, age of property, and more. Implementing machine learning algorithms or natural language processing for smarter search queries could further refine search results.

Virtual Property Tours: Introduce virtual reality (VR) or augmented reality (AR) features to offer immersive property tours. This allows users to explore properties remotely, providing a more engaging and interactive experience.

Integration with Real-Time Data: Incorporate real-time data feeds for property availability, market trends, and pricing updates. This ensures that users have access to the most current information, enhancing transparency and trust in the platform.

Collaborative Tools for Agents and Clients: Develop collaboration tools that enable seamless communication and interaction between real estate agents and clients. Features like in-app messaging, document sharing, and appointment scheduling streamline the buying or renting process.

Integration with Financial Services: Partner with financial institutions to integrate mortgage calculators, loan pre-approval tools, and property valuation services within the app. This provides users with comprehensive financial insights and facilitates decision-making.

Localized Market Expansion: Expand the app's reach by localizing content and services for specific regions or markets. This includes supporting multiple languages, currencies, and property regulations to cater to diverse audiences.

Scalable Infrastructure: Invest in a scalable backend infrastructure capable of handling increased traffic, data storage, and user interactions as the user base grows. Utilize cloud-based solutions like AWS or Google Cloud Platform for flexibility and scalability.

Data Analytics and Insights: Implement robust analytics tools to track user engagement, behavior patterns, and app performance metrics. Analyzing this data helps in identifying areas for improvement, optimizing features, and making data-driven decisions.

Customization and Personalization: Offer personalized recommendations and custom search preferences based on user behavior, preferences, and past interactions with the app. Personalization enhances user satisfaction and increases engagement

CHAPTER 9

APPENDIX

9.1 SOURCE CODE

Main Dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:zeeboombha/firebase_options.dart' show DefaultFirebaseOptions;
import 'package:zeeboombha/pages/splashscreen.dart';
import 'package:firebase_core/firebase_core.dart';
void main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp(
options: DefaultFirebaseOptions.currentPlatform,);
runApp(const MyApp());}
class MyApp extends StatelessWidget {
const MyApp({super.key});
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Real Estate',
theme: ThemeData(
primarySwatch: Colors.blue,),
home: const SplashScreen(),
debugShowCheckedModeBanner: false);
```

Login.Dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:zeeboombha/pages/mainscreen.dart';
import 'package:zeeboombha/pages/signup.dart';
import 'package:zeeboombha/widget/support_widget.dart';
import 'forgetpassword.dart';
class Login extends StatefulWidget {
const Login({super.key});
@override
State<Login> createState() => _LoginState();}
class _LoginState extends State<Login> {
String email = "", password = "";
final _formkey= GlobalKey<FormState>();
TextEditingController useremailcontroller = TextEditingController();
TextEditingController userpasswordcontroller = TextEditingController();
userLogin() async {
try {
await FirebaseAuth.instance
.signInWithEmailAndPassword(email: email, password: password);
Navigator.push(context,
MaterialPageRoute(builder: (context)=> const HomeScreen()));}
on FirebaseAuthException catch (e) {
if (e.code == 'user-not-found') {
ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
content: Text(
"No User Found With these Details",
style: TextStyle(
fontSize: 18.0, color: Colors.black, fontFamily: 'Poppins'),)));}
```

```
else if(e.code=='wrong-password'){
ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
content: Text(
"Invalid Password",
style: TextStyle(
fontSize: 18.0, color: Colors.black, fontFamily: 'Poppins'),)));
@override
Widget build(BuildContext context) {
return Scaffold(
body: SingleChildScrollView(
child: Stack(
children: [
Container(
width: MediaQuery.of(context).size.width,
height: MediaQuery.of(context).size.height / 2.5,
decoration: const BoxDecoration(
gradient: LinearGradient(
begin: Alignment.topLeft,
end: Alignment.bottomRight,
colors: [
Color(0xff1541dc),
Color(0xff1541dc),
// Color(0xFF)])),),
Container(
margin:
EdgeInsets.only(top: MediaQuery.of(context).size.height / 3),
width: MediaQuery.of(context).size.width,
height: MediaQuery.of(context).size.height / 2,
decoration: const BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.only(
topLeft: Radius.circular(30),
topRight: Radius.circular(30))),),
Container(
margin: const EdgeInsets.only(top: 60.0, left: 20.0, right: 20.0),
child: Column(
children: [
Center(
child: Image.asset(
"images/logo.png",
width: MediaQuery.of(context).size.width / 2,
fit: BoxFit.cover,),),
const SizedBox(
height: 30.0,),
Material(
elevation: 10,
borderRadius: BorderRadius.circular(20),
child: Container(
padding: const EdgeInsets.only(left: 20.0, right: 20.0),
width: MediaQuery.of(context).size.width,
height: MediaQuery.of(context).size.height / 2.2,
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.circular(20)),
child: Form(
key: _formkey,
child: SingleChildScrollView(
child: Column(
children: [
const SizedBox(
height: 20,),
Text(
"LOGIN",
style: AppWidget.headerTextFieldStyle(),),
const SizedBox(
height: 30,),
TextFormField(
controller: useremailcontroller,
validator: (value){
if(value==null||value.isEmpty){
return 'Please Enter Email';}
return null;},
decoration: InputDecoration(
hintText: 'Email',
hintStyle: AppWidget.lightTextFieldStyle(),
prefixIcon: const Icon(Icons.email_outlined)),),
const SizedBox(
height: 30.0,),
TextFormField(
controller: userpasswordcontroller,
validator: (value){
if(value==null||value.isEmpty){
return 'Please Enter email';}
return null;},
obscureText: true,
decoration: InputDecoration(
hintText: 'Password',
hintStyle: AppWidget.lightTextFieldStyle(),
prefixIcon: const Icon(Icons.password_outlined)),),
const SizedBox(
height: 30.0,),
GestureDetector(
onTap: (){
Navigator.push(context,MaterialPageRoute(builder:(context)=>const
ForgotPassword()));},
child: Container(
alignment: Alignment.topRight,
child: const Text("Forget Password?",
style: TextStyle(
fontFamily: 'Poppins',
fontWeight: FontWeight.w300,
fontSize: 14,
color: Colors.black))),),
const SizedBox(
height: 30.0,),
GestureDetector(
onTap: (){
if(_formkey.currentState!.validate()){
setState(() {
email = useremailcontroller.text;
password = userpasswordcontroller.text;});}
userLogin();},
child: Material(
borderRadius: BorderRadius.circular(10),
elevation: 5,
child: Container(
padding: const EdgeInsets.symmetric(vertical: 15),
width: 200,
// height: 50,
decoration: BoxDecoration(
color: const Color(0xff1541dc),
borderRadius: BorderRadius.circular(10)),
child: const Center(
child: Text(
"LOGIN",
style: TextStyle(
color: Colors.white,
fontSize: 18.0,
fontFamily: 'Poppins',
fontWeight: FontWeight.bold),)),),
const SizedBox(height: 20.0,)],),
const SizedBox(
height: 50,),
GestureDetector(
onTap: () {
Navigator.push(context,
MaterialPageRoute(builder: (context) => const Signup()));},
child: const Text(
"Don't have an Account? Sign Up",
style: TextStyle(fontFamily: 'Poppins'),),)],
```

MAINSCREEN.DART

```
import 'package:flutter/material.dart';
import 'package:google_nav_bar/google_nav_bar.dart';
import 'package:line_icons/line_icons.dart';
import 'package:zeeboombha/components/screens/home/components/houses.dart';
import 'package:zeeboombha/pages/contact_messages.dart';
import 'package:zeeboombha/pages/profile_screen.dart';
import 'package:zeeboombha/pages/searchpage.dart';

import '../components/screens/details/components/custom_app_bar.dart';
import '../components/screens/home/components/categories.dart';
class HomeScreen extends StatefulWidget {
static const routeName = '/home';
const HomeScreen({super.key});
@override
_HomeScreenState createState() => _HomeScreenState();}
class _HomeScreenState extends State<HomeScreen> {
int _selectedIndex = 0;
// Define separate page widgets
static final List<Widget> _widgetOptions = <Widget>[
const HomePage(),
const SearchPage(),
const ContactForm(),
const ProfileScreen(),];
@override
Widget build(BuildContext context) {
return Scaffold(
backgroundColor: Colors.white,
appBar: AppBar(
elevation: 20,
title: const Center(
child: Text('ZeeBoomBha'),),
backgroundColor: Colors.white,),
body: _widgetOptions.elementAt(_selectedIndex),
bottomNavigationBar: Container(
decoration: BoxDecoration(
color: Colors.black,
boxShadow: [
BoxShadow(
blurRadius: 20,
color: Colors.black.withOpacity(.1),)],
child: SafeArea(
child: Padding(
padding:
const EdgeInsets.symmetric(horizontal: 15.0, vertical: 8),
child: GNav(
rippleColor: Colors.grey,
hoverColor: Colors.grey,
gap: 8,
activeColor: Colors.blue,
iconSize: 24,
padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 12),
duration: const Duration(milliseconds: 400),
tabBackgroundColor: Colors.white,
color: Colors.blue,
tabs: const [
GButton(
icon: LineIcons.home,
text: 'Home',),
GButton(
icon: LineIcons.quora,
text: 'Likes',),
GButton(
icon: LineIcons.search,
text: 'Search',),
GButton(
icon: LineIcons.user,
text: 'Profile',),],
selectedIndex: _selectedIndex,
onTabChange: (index) {
setState(() {
_selectedIndex = index;});},
// Home Page Widget
class HomePage extends StatelessWidget {
const HomePage({super.key});
@override
Widget build(BuildContext context) {
return const Scaffold(
body: Stack(
alignment: Alignment.bottomCenter,
children: [
Column(
children: [
CustomAppBar(),
Categories(),
Houses(),],}}
```

9.2 SCREENSHOTS

9.2.1 SPLASH SCREEN PAGE

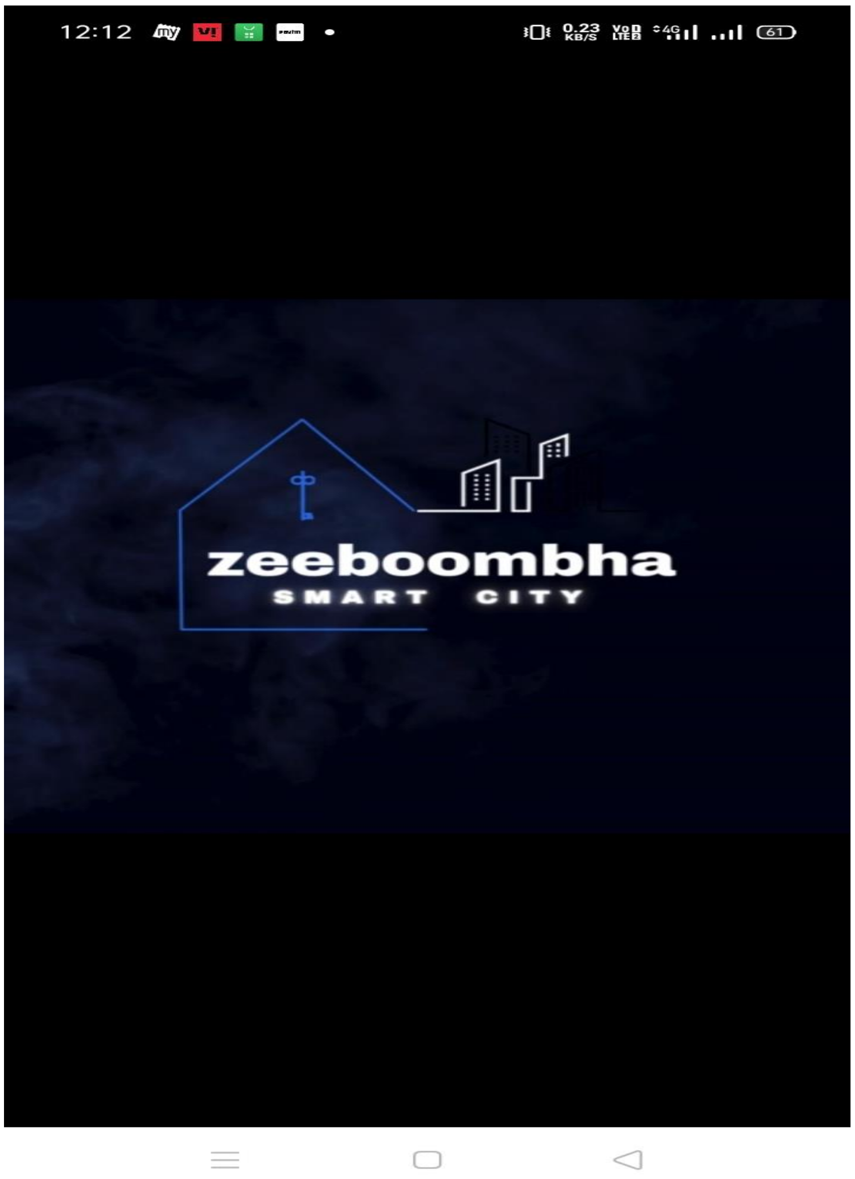


Fig 8 Splash Screen Page

9.2.2 LOGIN AND SIGNUP PAGE

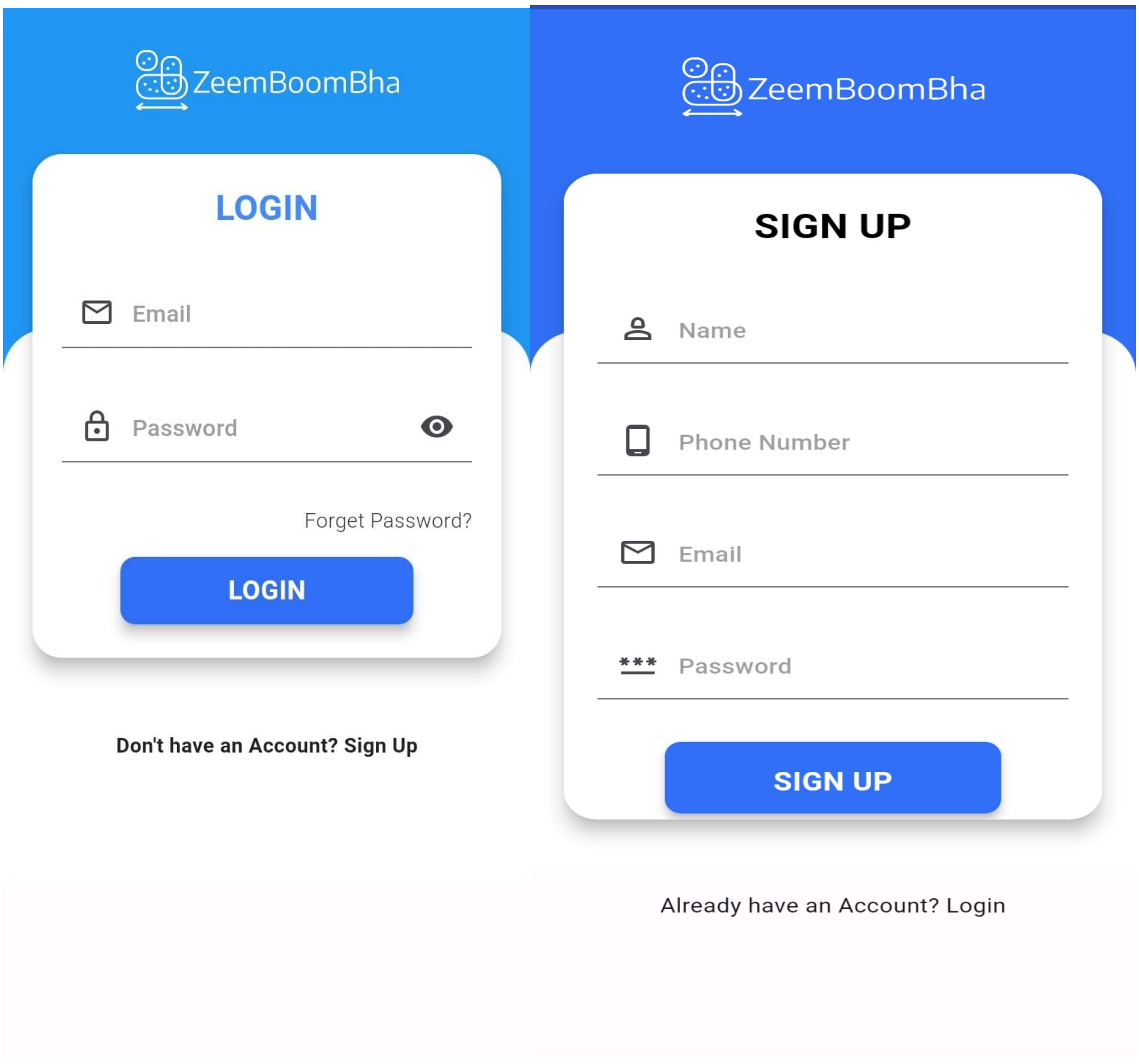


Fig 9 Login Page and SignUp Page

9.2.3 PROPERTY LISTING PAGE

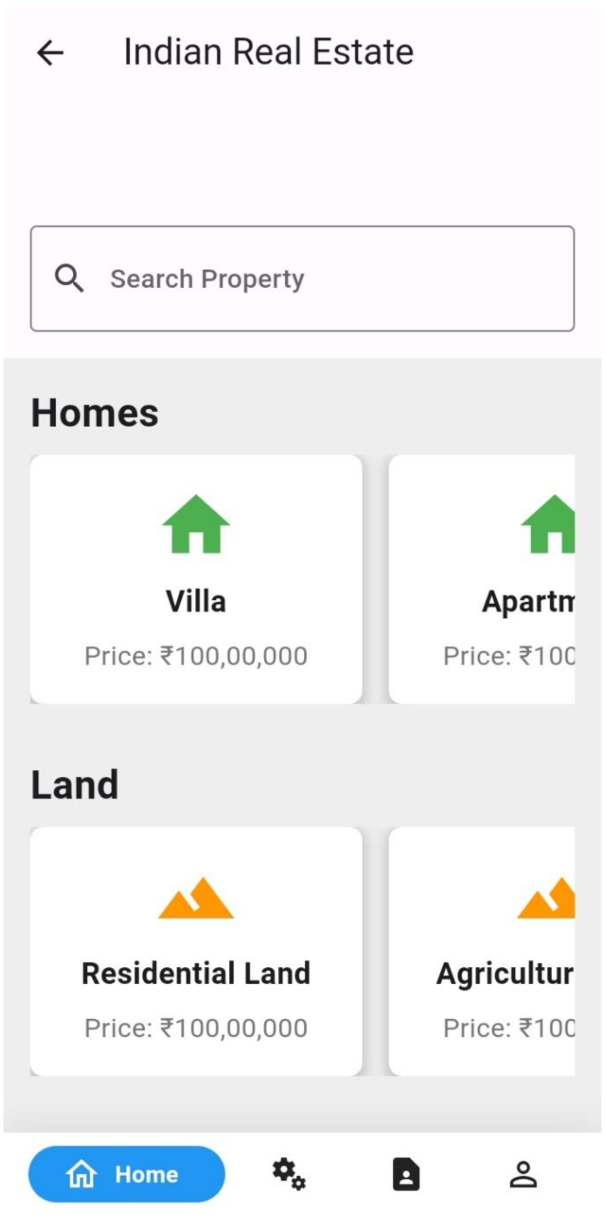
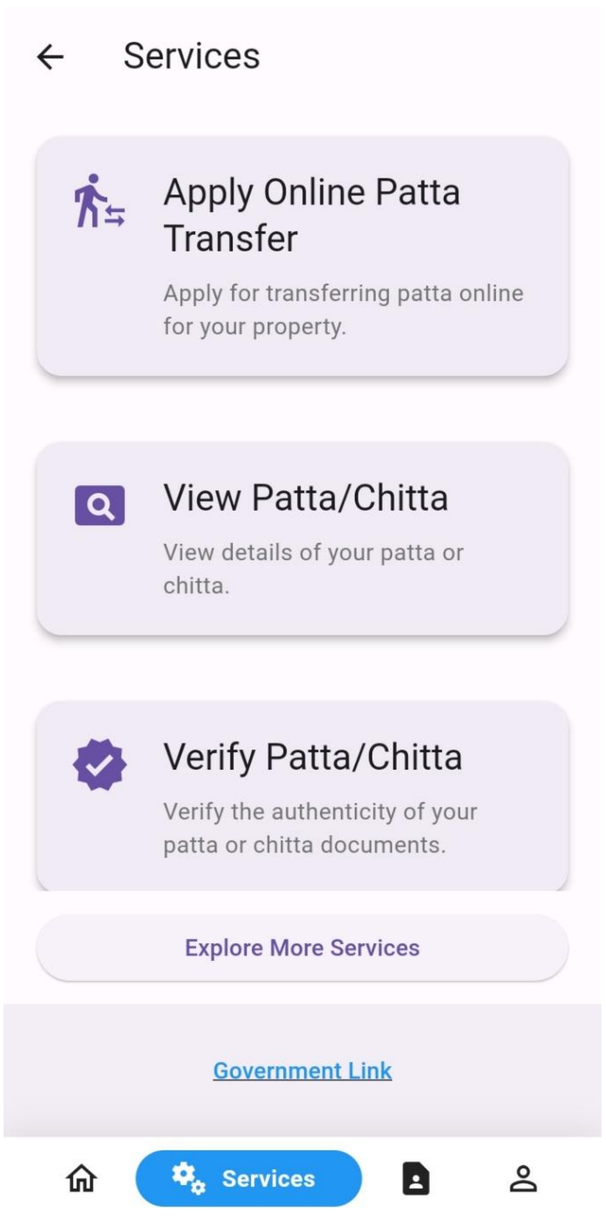


Fig 10 Property Listing Page

9.2.4 SERVICES PAGE


Services

Apply Online Patta Transfer

Apply for transferring patta online for your property.

View Patta/Chitta

View details of your patta or chitta.

Verify Patta/Chitta

Verify the authenticity of your patta or chitta documents.

Fig 11 Services Page

9.2.5 USER PROFILE AND SETTING PAGE

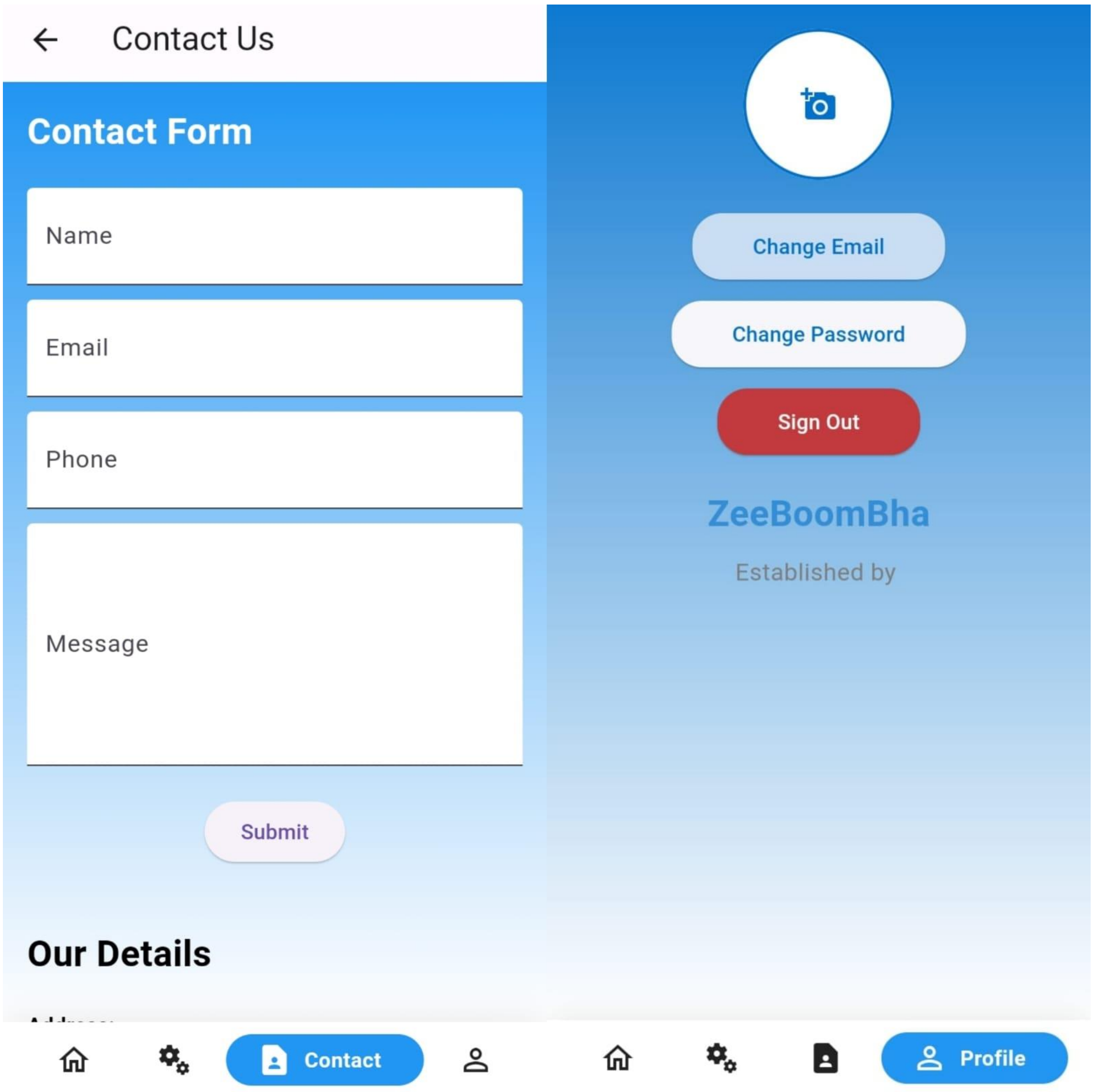


Fig 12 User Profile and Setting Page

CHAPTER 10

REFERENCES

1. Applications, Opportunities and Challenges of AI in the Real Estate Sector, Ricardo Ribeiro, João Antão, 2024

2. Artificial intelligence algorithms to predict Italian real estate market prices, Luca Rampini, Fulvio Re Cecconi, 2024

3. Blockchain in real estate: Recent developments and empirical applications, Anniina Saari, Jussi Vimpari, 2023

4. Booch, Grady. The unified modeling language user guide. Pearson Education India, 2023.

5. Consignado, Mark Lloyd Lester S., et al. "HAYBOL: An Android-Based Apartment Locator Application." International Journal of Computing Sciences Research 1.2 (2023): 1-9.

6. Developing a mobile application for smart real estate information, arif aydinoglu, rabia bovkir, 2023

7. Evaproperty.com, [Online]. Available: http://ww5.evaproperty.com/ [Accessed February 3, 2023]

8. Gartner, L. G. worldwide sales of smartphone grew 7 percent in the fourth quarter of Retrieved January 15, 2023, from http://www.gartner.com/newsroom/id/3609817

9. Ibisola,A.S.,A.S. Oni, and Nkolika Joy Peter. "The Relevance and Application of ICT in Estate Surveying and Valuation in Ogun State." 112-120, 2023

10. Mobile CRM development for real estate agents, Ruben Pereira, Ricardo Ribeiro, 2022

11. Mobile gis applications for environmental field surveys: a state of the art, maciej marcin nowak, katarzyna dzióberta, 2022

12. Montenegro, B.. The Agoda mobile app makes short work of hotel booking., [Online].Available:http://www.gmanetwork.com/news/scitech/content/382558/review-the-agoda-mobile-app-makes-short-work-of-hotel-booking/story. [Accessed January 16, 2022]

13. Nurul, N. S., Nurul, H. A., Noorsidi, A. M., Nurul, S. M., & Mohd, S. A.. E-Commerce In The Malaysian Real Estate Agency Industry. International Journal of Real Estate Studies, 11(5) 2022.

14. Realista,[Online].Available:http://realistatoken.com/public/assets/files/RealistaToken (RE) whitepaper.pdf. [Accessed March 1, 2022]

15. Royce, F. W., & Kinn, B. A. Eureka: Analysis, Design And Implementation Of A Smartphone Application For Real Estate Seekers. International Journal of Information System and Engineering, 5(1) 2021.

16. Singh, H. . Best mobile app for hunting house. Retrieved February 3, 2021, from http://technomedia.in/205/04/best-mobile-apps-forhunting-the-house-28858

17. Sommerville, Ian. Software engineering. New York: Addison-Wesley, 2021.

18. Using Mobile Dashboards to Track Real Estate Brokers Productivity, João Antão, Ruben Pereira, 2021

19. Williams, E. Best Apps for Real Estate Agents and Agencies. Retrieved June 26, 2021, [Online]. Available: https://pdf.wondershare.com/business/apps-for-real-estate-agents.html. (Accessed: 28th June 2021).

20. Xu, Rubin, Hassen Saïdi, and Ross J. Anderson. "Aurasium: Practical Policy Enforcement for Android Applications." USENIX Security Symposium. 2020.