INTEL UNNATI INDUSTRIAL INTERNSHIP PROGRAM REPORT

Title or Problem Statement

"Integrated common services to common people"

Submitted to

"Intel Unnati Industrial Training Program"

Affiliated to

Parul Polytechnic Institute, Parul University, Vadodara

Submitted By

Heer Patel

Bansari Patel

Jainil Patel

Aayushi Singhala

Company Guide

Mr Satish Sir

Faculty Guide

Prof. Ankit Jasoliya

ACKNOWLEDGEMENT

At the beginning I extend my heartfelt gratitude and it's a matter of pride and privilege for us to have done a summer internship project with "INTEL" and we are sincerely thankful to them, who took confidence in us and provided this opportunity to us. Where we gained invaluable experience in Android development.

This internship has been a significant milestone in our academic journey, allowing us to apply classroom knowledge to real-world projects.

We would like to express our sincere appreciation to Intel for welcoming us into their esteemed organization and for providing an enriching environment conducive to learning, special thanks and sincere gratitude towards **Mr. Satish** Sir (Mentor and Senior Expert at Intel), **Mrs. Vanshika** Mam (Working with Intel)**, Mr. Harikrishna** Sir (Expert at Intel) and **Mr. Debdyut Hazra** Sir (Mentor), for their guidance, mentorship, and unwavering support and making it much easier for us to adjust to the work. Their careful and precious guidance was extremely valuable to complete the tasks and their expertise and insights have been instrumental in shaping our understanding of Android development and professional growth.

We are highly indebted to Principal **Dr. Jatin Vaidya** Sir, for giving us the opportunity to accomplish this internship.

We would like to thank our Head of the Department **Mrs. Poonam Faldu** Mam or her constructive criticism throughout my internship.

We would like to thank **Mr. Ankit Jasoliya Sir,** College internship coordinator of Computer Engineering and our Mentor for this program and we are also thankful to him for their support and advices to get and complete internship in above said organization.

It is indeed with a great sense of pleasure and immense sense of gratitude that we acknowledged the help of these individuals.

At last, we perceive this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way.

Thank you all for making this internship experience at Intel a rewarding and memorable chapter in my career.

ABSTRACT

This project report documents the design, development, and implementation of a "Integrated Common Services to Common People App" called "HandyHelper" aimed at consolidating essential public and private sector services into a single, user-friendly platform. The HandyHelper seeks to simplify access to services such as Healthcare, Education, Government, Finance, Transportation and Housing service thereby enhancing convenience and efficiency for users everything together on a single platform.

The development of the HandyHelper spanned a period of Two months, utilizing Agile methodologies to accommodate evolving user needs and technological advancements. Key features of the app include an authorized and secure User login and Registration, unified Dashboard for accessing multiple services, different services for healthcare, housing and education with dummy service providers.

Challenges encountered during the project included data integration complexities, ensuring compatibility across diverse mobile and web platforms, and maintaining data security standards to protect user information. These challenges were addressed through rigorous testing phases, continuous feedback loops with stakeholders, and adherence to industry best practices.

Results indicated high user satisfaction with the app's intuitive interface, and responsive customer support. The app's performance metrics, including user engagement analytics, demonstrated its effectiveness in simplifying service access and improving user experience.

This comprehensive abstract provides an overview of the project's objectives, development approach, key features, challenges, evaluation outcomes, and future recommendations for the HandyHelper App.

TABLE OF CONTENT

LIST OF FIGURES

INTRODUCTION

1.1 Background of the Project

In the contemporary digital era, the accumulation of mobile applications has fundamentally altered how individuals access and interact with essential services. The rapid advancement of digital technologies has transformed how people access essential services. Despite the widespread availability of digital tools, yet there remains a significant challenge in integrating these services into a single platform and many individuals still face difficulties in navigating multiple platforms to obtain services related to Education, Healthcare, Housing, Transportation, Finance, and Government. Fragmentation across various applications makes it difficult for users, especially those with limited digital literacy, to access essential services efficiently. Users often need to navigate through different applications to access healthcare, education, housing, transportation, finance, and government services, which can be cumbersome and time-consuming, particularly for those with limited digital literacy.

In today's digital age, the demand for centralized platforms offering multiple services has grown significantly. Recognizing this trend, This project, the development of the HandyHelper App, seeks to address this fragmentation by integrating these diverse services into a single, cohesive platform. The project aimed to streamline the user experience, making it easier for individuals to access the essential public and private sector services they need without having to switch between multiple applications through a unified digital platform. This integration not only enhances user convenience but also promotes more efficient service delivery.

The impetus for this project stems from the recognition of several key issues like:

Fragmented Service Delivery: Currently, users must navigate through multiple apps, each catering to a specific service. This fragmentation can lead to inefficiencies and user frustration.

Accessibility Barriers: Many existing apps are not user-friendly for individuals with limited digital literacy, exacerbating the digital divide.

Security Concerns: With users having to manage multiple accounts across Different platforms, the risk of data breaches and security vulnerabilities increases.

User Experience: A seamless and cohesive user experience is often lacking in the current landscape of service-oriented apps.

To solve this problem and to build this app we focused on Android development, this project provided an opportunity to learn and apply practical skills in mobile app development. The HandyHelper was developed using Android Studio and Java, two fundamental tools in the Android development ecosystem. This project was not only a technical exercise but also a valuable learning experience, encompassing aspects of user interface design, database development and backend development.

1.2 Objective of the Application

The primary objective of developing the HandyHelper app is to create a cohesive platform that:

A. Enhance Accessibility to Public Services

- **Goal**: Ensure that citizens can easily access a variety of public services through a single, unified platform.
- **Outcome**: Increased utilization of public services, particularly by underserved and marginalized communities.

B. Improve Efficiency of Service Delivery

- **Goal**: Reduce the time and effort required to access and use public services and Streamline the process of service delivery, reducing the time and effort required by users.
- **Outcome**: Faster processing times and reduced administrative burdens for both service providers and users.

C. Foster Digital Inclusion

- **Goal**: Bridge the digital divide by providing a user-friendly interface that is accessible to individuals and ensuring that individuals of all digital literacy levels can navigate it with ease.
- **Outcome**: Higher adoption rates of digital services across different demographics.

D. Enhance Transparency and Accountability

- **Goal**: Provide a transparent platform where users can track the status of their service requests and provide feedback.

- **Outcome**: Increased trust in public institutions and improved service quality through user feedback mechanisms.

E. Promote Integration

- **Goal**: Integrate various public service systems to create a seamless user experience.
- **Outcome**: Reduced duplication of efforts and improved coordination among different government departments and agencies.

F. Support Sustainable Development Goals (SDGs)

- **Goal**: Align with and support the United Nations Sustainable Development Goals, particularly those related to reducing inequality, improving infrastructure, and promoting inclusive and sustainable economic growth.
- **Outcome**: Contribution to national and international development objectives through enhanced public service delivery.

G. Expected Outcomes

- **Consolidate Services:** Integrate various essential services into one cohesive platform to simplify access.
- **Security and Privacy:** To implement robust security measures to protect user data and ensure safe transactions within the app.
- **User Satisfaction**: High levels of user satisfaction due to the convenience and efficiency of accessing multiple services through a single app.
- **Operational Efficiency**: Significant reduction in the operational costs and time associated with delivering public services.
- **Digital Literacy**: Increased digital literacy and inclusion among the population.
- **Service Quality**: Improved quality and responsiveness of public services based on user feedback and data analytics.
- **Enhance Accessibility:** Ensure that common people, especially those with limited digital literacy, can easily use the app.
- Facilitates data-driven decision-making and service improvements based on user feedback and analytics.

1.3 Importance of the App

The significance of the app lies in its potential to revolutionize how common people interact with essential services. By providing a one-stop solution, the app addresses the inefficiencies and accessibility barriers currently faced by users. The importance of such an integrated platform

is further underscored by the following factors:

A. Centralized Access to Services

- **Explanation**: The app serves as a single gateway to various public services, reducing the need for citizens to navigate multiple platforms and with all services available on a single platform, users can save time and avoid the hassle of managing multiple applications.
- **Impact**: Simplifies the process of accessing government services education services healthcare services, housing services, finance services, transport services etc. saving time and effort for users.

B. Improved Service Delivery

- **Explanation**: By digitizing public services, the app enhances efficiency and reduces bureaucratic delays and a unified platform can provide a more seamless and cohesive user experience, increasing user satisfaction and engagement.
- **Impact**: Faster processing times and better user experiences lead to higher citizen satisfaction.

C. Increased Transparency and Accountability

- **Explanation**: The app has the potential to reach a broader audience, including those in remote or underserved areas who may not have easy access to these services.
- **Impact**: Enhances trust in public institutions and ensures accountability through transparent operations.

D. Digital Inclusion

- **Explanation**: Designed to be user-friendly and easy for people with varying levels of digital literacy.
- **Impact**: Promotes inclusivity, allowing more citizens to benefit from digital services.

E. Cost Efficiency

- **Explanation**: Reduces operational costs by streamlining service delivery and cutting down on physical paperwork and also by reducing the need for multiple subscriptions and accounts, the CSA can help users save money
- **Impact**: Frees up resources that can be redirected to other critical areas.

1.4 Future Scope of the App

The future scope of the app is vast, with numerous possibilities for expansion and enhancement:

a. Expansion of Services

- **Future Direction**: Continuously adding new services and features based on user demand and government initiatives such as legal aid, employment services, and social welfare programs, to further meet the needs of users
- **Potential Impact**: Broader service range, catering to more aspects of citizens' lives and needs.

b. Advanced Analytics and AI Integration

- **Future Direction**: Utilizing big data analytics to gain insights into user behaviour, to personalize services and predict user needs and preferences, allowing for the refinement and enhancement of services. Incorporating AI can provide personalized recommendations and improve user experience. For instance, AI could suggest relevant healthcare services based on user health records or recommend educational courses aligned with user interests.
- **Potential Impact**: Enhanced user experiences and proactive service delivery.

c. Enhanced User Engagement

- **Future Direction**: Introducing gamification, loyalty programs, and other engagement tools to increase user interaction.
- **Potential Impact**: Higher user retention and satisfaction.

d. Cross-Border Services and Multilingual Support:

- **Future Direction**: Facilitating services for expatriates and international users, including cross-border transactions and verifications. Adding support for multiple languages to cater to a diverse user base, making the app more inclusive.
- **Potential Impact**: Extending the app's utility to a global user base

e. Cross-Platform Availability:

- **Future Direction**: Expanding the app's availability to other platforms like iOS and web, increasing its accessibility.
- **Potential Impact**: Expanding the app to other users of IOS and web will increase the demand of app and fulfil all the user needs.

LITERATURE REVIEW

2.1 Overview of Existing Common Services Apps

The digital marketplace is replete with various mobile applications designed to offer individual services such as healthcare, education, housing, transportation, finance, and government services. However, there are few apps that integrate multiple services into a single platform. This section reviews existing common services apps, examining their functionalities and user experiences to identify gaps and opportunities for the Common Services App.

Real-Time Examples and Market Perspectives

The success of integrated service apps like WeChat in China, which combines messaging, social media, and payment services, demonstrates the potential of app in other regions. WeChat has revolutionized how users interact with digital services, making it a staple in daily life. Similarly, the HandyHelper app aims to become an indispensable tool for users by integrating essential services into one platform.

Healthcare Apps

Healthcare apps like MyChart, Medisafe, and Practo are widely used for managing health records, medication reminders, and booking medical appointments. MyChart, for example, allows users to view their medical history, schedule appointments, and communicate with healthcare providers. Medisafe offers medication management, providing reminders and health tips. Practo extends its services to include doctor consultations, health check-ups, and an online pharmacy.

For instance, during the COVID-19 pandemic, the Aarogya Setu app in India played a crucial role in providing health-related information and contact tracing services. The HandyHelper app can build on such examples by offering not only healthcare services but also integrating education, housing, transportation, finance, and government services, making it a comprehensive solution.

Education Apps

In the education sector, apps like Coursera, Khan Academy, and Duolingo are prominent. Coursera provides access to online courses from universities and companies, enabling users to learn new skills and earn certificates. Khan Academy offers free educational resources covering a wide range of subjects,

from elementary to advanced levels. Duolingo is a language learning app that makes learning new languages engaging through gamification.

Housing Apps

Zillow, Realtor.com, and Redfin are popular apps in the housing domain. These apps provide property listings, real estate market trends, and tools for buying, selling, or renting homes. Zillow, for instance, offers extensive listings and detailed information about properties, including pricing, photos, and virtual tours. Redfin combines real estate brokerage services with a comprehensive home search tool.

Transportation Apps

Apps like Uber, Lyft, and Google Maps dominate the transportation sector. Uber and Lyft provide ride-hailing services, allowing users to book rides quickly and track their drivers in real-time. Google Maps offers navigation and traffic information, helping users find the best routes for driving, walking, biking, and public transit.

Finance Apps

Financial management apps such as PayPal, Venmo, and Mint are widely used for transactions and budgeting. PayPal facilitates online payments and money transfers, while Venmo focuses on peer-to-peer payments with a social component. Mint helps users manage their finances by tracking income, expenses, and investments, providing budget insights and financial tips.

In the financial sector, apps like PayTM and Google Pay have simplified digital transactions. By integrating similar financial services, the HandyHelper app can provide users with a secure and efficient way to manage their finances. Additionally, integrating government services as seen in the UMANG app can help users access a wide range of public services from a single platform.

Government Services Apps

In the realm of government services, apps like MyGov, UMANG (Unified Mobile Application for New-age Governance), and USA.gov offer access to various public services. UMANG, for example, integrates numerous government services into a single platform, including utility bill payments, health services, and education resources. MyGov provides a platform for citizen engagement with government initiatives and services.

2.2 Analysis of Similar Platforms and Their Features

➢ Strengths

- **Multi-Service Integration**: These apps integrate various government services, such as healthcare, education, transportation, and social welfare, into a single platform. Examples are India's UMANG app, Estonia's e-Estonia platform.

- **User-Centric Design:** Many existing apps have user-friendly interfaces and intuitive designs that cater to a broad audience. This ease of use is crucial for ensuring accessibility across different user demographics.

- **Specific Functionality:** Apps like Uber and MyChart excel in providing specialized services. This specialization allows them to focus on enhancing specific user experiences and optimizing service delivery.

- **Robust Security:** Robust user authentication mechanisms to ensure secure access to personal data and services via secured login and registration. Financial and healthcare apps often have strong security measures in place to protect sensitive user data. Examples are Biometric authentication, two-factor authentication (2FA), and secure login credentials. PayPal uses encryption and multi-factor authentication to secure transactions.

- **Wide Adoption and Trust:** Apps like Google Maps and Coursera have gained widespread user trust and adoption due to their reliability and comprehensive service offerings.

- **Integration with External Services:** Many apps offer integration with other services and platforms. For example, Mint integrates with various banks and financial institutions to provide a holistic view of user finances.

➢ Weaknesses

- **Fragmentation:** One of the primary weaknesses is the fragmentation of services across multiple apps. Users must manage numerous accounts and interfaces, which can be cumbersome and inefficient.

- **Limited Accessibility:** Some apps are not fully accessible to users with disabilities or those with low digital literacy. This limitation can exclude significant portions of the population.

- **Data Privacy Concerns:** Despite robust security measures, concerns about data privacy and misuse persist, particularly in apps that collect and store sensitive information.

- **Overlapping Services:** Many apps offer overlapping services, leading to redundancy and confusion among users. For example, several payment apps provide similar functionalities but require separate accounts and interfaces.

Opportunities for the App

- **Integration of Services:** The CSA can differentiate itself by integrating a wide range of services into a single platform, reducing the need for multiple apps and streamlining the user experience. This integration can include features from healthcare, education, housing, transportation, finance, and government services.

1. **Enhanced Accessibility:** By focusing on user-centric design principles and ensuring the app is accessible to users with varying levels of digital literacy and disabilities, the CSA can expand its user base and promote inclusivity.
2. **Robust Security and Privacy:** Implementing advanced security measures and clear privacy policies can build user trust and protect sensitive data. The CSA can adopt best practices from leading apps to ensure user data is secure.
3. **Personalization and AI:** Incorporating artificial intelligence to personalize the user experience can enhance engagement and satisfaction. For example, AI can recommend relevant services based on user behavior and preferences.
4. **Cross-Platform Availability:** Expanding the app's availability to multiple platforms (Android, iOS, and web) can increase its accessibility and user base. Cross-platform compatibility ensures that users can access services regardless of their preferred device.

METHODOLOGY

3.1 Development Approach

Developing a robust and user-friendly mobile application like the "HandyHelper" app requires a well-defined and systematic approach. The development approach chosen for the App is a critical factor that determined the efficiency, quality, and success of the project. Given the complexity and scope of integrating multiple services like healthcare, education, housing, transportation, finance, and government services into a single platform, a flexible and adaptive development methodology was essential. The choice of development methodology significantly impacts the efficiency and success of the project. In this project, the Agile methodology was primarily adopted, although aspects of the Waterfall methodology were also considered for specific phases.

3.1.1 Agile Methodology

The Agile methodology is a flexible and iterative approach to software development that emphasizes collaboration, customer feedback, and small, rapid releases. Agile methodologies are well-suited for projects where requirements are expected to evolve, and continuous feedback is necessary to refine and improve the product.

- **Key Principles of Agile Methodology:**

1. **Iterative Development:** Agile divides the project into small, manageable units called iterations or sprints, typically lasting two to four weeks. Each sprint involves a complete development cycle, including planning, development, testing, and review.

2. **Customer Satisfaction through Continuous Delivery:** Agile prioritizes delivering functional software frequently, with iterations typically ranging from two to four weeks. This ensures that stakeholders see continuous progress and can provide timely feedback.

3. **Embrace Change:** Agile methodologies accept that requirements will change over time, even late in the development process. The focus is on responding to change rather than strictly adhering to a predefined plan.

4. **Frequent Delivery of Working Software:** Each iteration aims to produce a potentially shippable product increment. This approach allows for regular demonstrations to stakeholders and early detection of issues.

5. **Collaboration between Technical Teams:** Agile promotes close, daily collaboration between developers to ensure that the project remains aligned with business needs and objectives.

6. **Motivated Individuals and Teams:** Agile values motivated individuals, providing them with the environment and support they need to succeed, and trusting them to get the job done.

7. **Face-to-Face Communication:** While written documentation is important, Agile emphasizes face-to-face communication as the most effective method of conveying information within a development team.

8. **Sustainable Development:** Agile processes promote sustainable development, where the sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. **Technical Excellence and Good Design:** Continuous attention to technical excellence and good design enhances agility.

10. **Simplicity:** Agile focuses on the art of maximizing the amount of work not done, promoting simplicity.

11. **Self-Organizing Teams:** The best architectures, requirements, and designs emerge from self-organizing teams.

12. **Regular Reflection and Adjustment:** At regular intervals, the team reflects on how to become more effective and adjusts its behaviour accordingly.

13. **Focus on Value:** Agile prioritizes delivering functional features that provide the most value to users as early as possible.

Agile Implementation in HandyHelper App Development

- **Sprint Planning and Execution:** The project was divided into multiple sprints, each focusing on a specific set of features. For example, the initial sprints focused on developing the core functionalities, such as the app logo, user Registration, user Login and its validation, authentication, while later sprints addressed service integration and UI enhancements.

- **Daily Stand-Ups:** Short, daily meetings and discussion were held among our team to discuss everyone's on going tasks as per the schedule and to check the progress, challenges, errors and plans for the day. These stand-

ups facilitated communication and helped identify and resolve issues quickly.

- **Sprint Reviews:** At the end of each sprint, a review meeting was conducted within the team to demonstrate the completed features as per the user requirement.

- **Sprint Retrospectives:** Following the sprint review, the team held a meeting to reflect on what went well, what didn't, and how processes could be improved. This continual improvement process was key to the success of the project.

- **User Stories and Backlog Management:** Development tasks were organized into user stories, each describing a feature from the end-user's perspective. This approach ensured that development was aligned with user needs.

- **Continuous Integration and Testing:** Agile emphasizes the importance of regular testing and integration to ensure that new code integrates seamlessly with the existing codebase.

- Regular testing and integration ensured that the app was stable and functional throughout the development process. Automated tests were used to verify that new changes did not introduce bugs or regressions.

- **Sprint Reviews and Retrospectives:** At the end of each sprint, reviews were conducted to demonstrate the completed features to stakeholders. Retrospectives were held to reflect on the sprint and identify areas for improvement.

3.1.2 Waterfall Methodology (Complementary Aspects)

Waterfall Implementation in HandyHelper App Development

While Agile was the primary methodology used, certain aspects of the Waterfall methodology were incorporated, particularly during the initial phases of the project. The Waterfall methodology is a linear and sequential approach to software development, characterized by distinct phases and each phase must be completed before moving on to the next, and there is little room for revisiting previous phases. The distinct phases of waterfall model are:

- Requirements analysis,

- Design,
- Implementation,
- Testing,
- Deployment,
- Maintenance.

Key Features of Waterfall Methodology

1. **Sequential Phases:** Each phase must be completed before moving on to the next. This provides a clear, structured and predictable process.

2. **Clear Documentation:** Detailed documentation is produced at each stage, ensuring that all aspects of the project are well-documented.

3. **Well-Defined Requirements:** Requirements are gathered and defined at the beginning of the project, reducing the risk of scope changes later on.

Waterfall Implementation in our App Development

While the Agile methodology was the primary approach, certain aspects of the Waterfall methodology were incorporated, particularly during the initial phases of the project.

- **Initial Requirements Analysis:** At the start of the project, a comprehensive analysis was conducted to gather and document user requirements and organizations expectations. This phase involved market research, user surveys, to ensure a thorough understanding of the needs the app aimed to address. This phase was critical for understanding the scope of the project and ensuring that all necessary features and functionalities were identified.

- **Design Phase:** The design phase involved creating detailed design documents, including the system architecture, database schema, and user interface designs. This phase provided a clear blueprint for the development team to follow during implementation.

    - **System Architecture:** The system architecture was designed to ensure scalability, reliability, and performance. This included defining the overall structure of the app, including the interactions between different components and services.

    - **UI/UX Design:** User interface and user experience design were critical for ensuring that the app was user-friendly and accessible.

- Prototypes were created to visualize the app's design and flow.

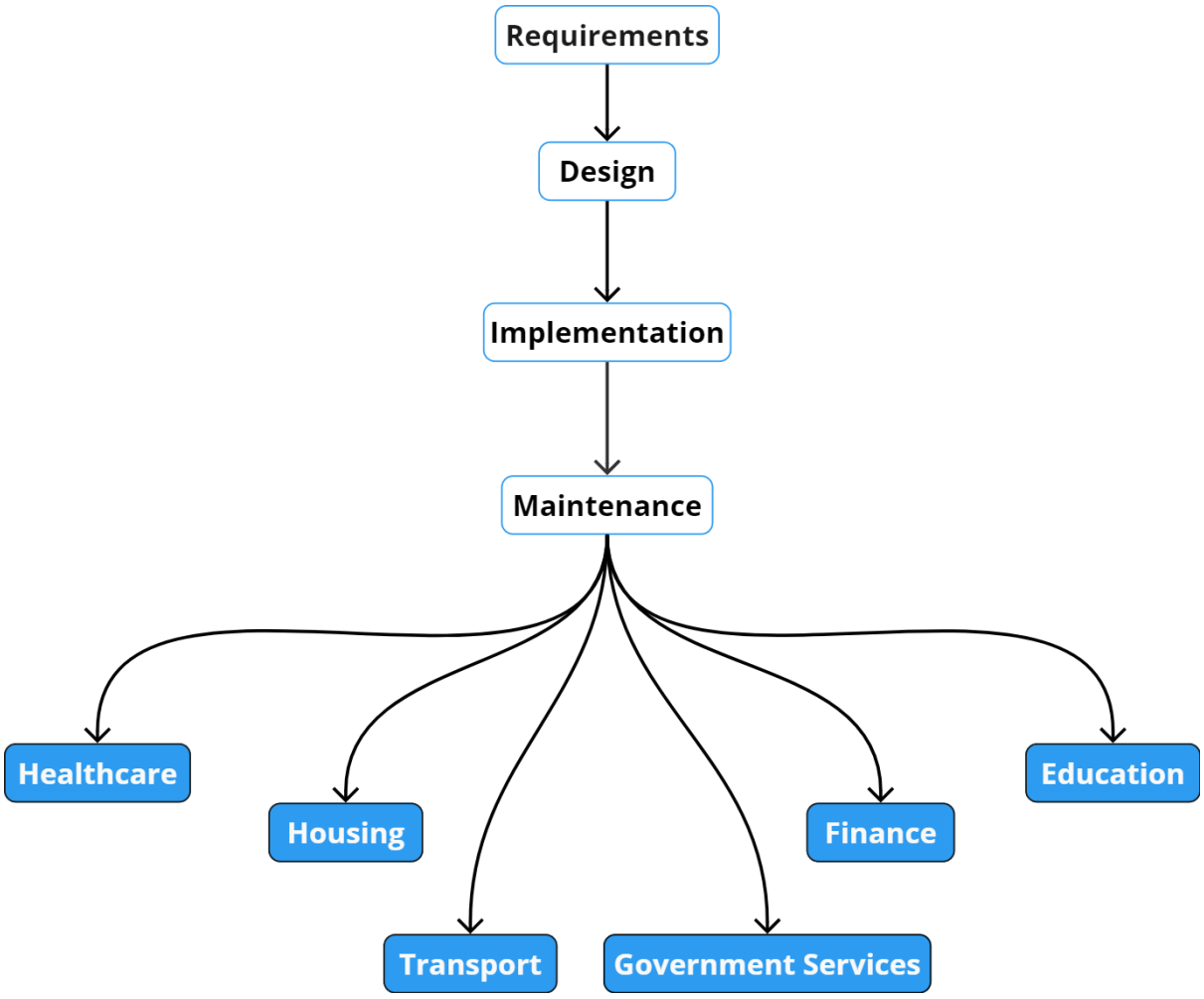


Fig. 1.1. Waterfall Model

3.2 Project Planning and Timeline

Effective project planning and a well-defined timeline were crucial for the successful completion of the App project. The project was divided into multiple phases, each with specific milestones and deliverables. This section outlines the project planning process and provides a detailed timeline followed.

Project Planning

- **Initial Planning Phase**

- **Requirement Gathering:** The project began with a comprehensive requirement gathering phase, involving discussions with users, stakeholders, among the team and market research, and analysis of existing solutions. This phase aimed to understand the needs and expectations of users and stakeholders.

- **Feasibility Study:** A feasibility study was conducted to assess the technical, economic, and operational feasibility of the project. This included evaluating the technical requirements, estimating costs, and identifying potential risks.

Development Planning

- **Sprint Planning:** The development process was organized into sprints, with each sprint lasting two weeks. Sprint planning meetings were held to define the goals and tasks for each sprint, ensuring that the team had a clear roadmap.
- **Task Breakdown:** The project was broken down into smaller tasks, each assigned to specific team members. This breakdown ensured that the work was manageable and allowed for parallel development efforts.
- **Milestones:** Key milestones were defined to track progress and ensure that the project stayed on schedule. Milestones included completing major features, integrating services, and achieving specific performance benchmarks

Phase 1: Requirements Analysis and Planning (Weeks 1-2)

Week 1:

- Initial meetings with stakeholders to gather requirements and understand project goals.
- Market research to analyze existing solutions and identify gaps.
- Feasibility study to assess technical, economic, and operational feasibility.

Week 2:

- Defining the scope and objectives of the project.
- Creating a high-level project plan, including a timeline and milestones.
- Setting up the development environment and tools

Phase 2: Design (Weeks 3-4)

Week 3:

- Designing the system architecture, including defining components, interactions, database and data flow.

- Creating wireframes and high-fidelity UI/UX designs and interactive prototypes to visualize the user interface and flow.

- Conducted design reviews with mentor, team and stakeholder and users to gather feedback and after that reviewing and finalizing the design documents with stakeholders and make necessary adjustments.

Week 4:

- Developing detailed design documents, including database schema.
- Setting up the project repository and configuring version control.
- Preparing for the development phase by creating initial code structure and setting up dependencies.

Phase 3: Development (Weeks 5-10)

Week 5-6:

- Sprint 1: Divided the development phase into multiple sprints, each focusing on specific features and functionalities and are developing, login and registration core functionalities, including user authentication and initial database setup.
- Developed core features, including user registration, authentication, and service browsing.

- Daily stand-up meetings to discuss progress and address issues.
- Continuous integration and testing of completed features.

Week 7-8:

- Sprint 2: Implementing key services, such as healthcare and education modules and integrated various services such as healthcare, education, housing, transportation, finance, and government services.
- Conducting sprint review meetings to gather feedback and make necessary adjustments.

Week 9-10:

- Sprint 3: Adding additional services, such as housing and transportation.
- Enhancing UI/UX based on feedback from sprint reviews.

- Performing integration tests to ensure seamless interaction between different modules.

- Developed the backend infrastructure, including APIs, database integration, and server-side logic.
- Implemented the user interface and ensured seamless interaction with the backend.

Phase 4: Testing and Quality Assurance (Weeks 11-12)

Week 11:

- Conducting comprehensive testing, including unit tests, integration tests, and UI tests. Conducted unit tests to verify the functionality of individual components.
- Performed integration tests to ensure that different modules worked together seamlessly.
- Conducted unit tests to verify the functionality of individual components.

- Identifying and fixing bugs and issues.
- Preparing for user acceptance testing (UAT) by creating test plans and scenarios.

Week 12:

- Engaged with a group of users to test the app and provide feedback on usability and functionality using User Acceptance Testing (UAT).
- Gathering feedback and making final adjustments.
- Conducting performance testing to ensure the app meets performance benchmarks.

Phase 5: Review (Week 13)

- **Bug Fixes and Enhancements:** Addressed bugs and made necessary enhancements based on user feedback.
- **Final Review:** Conducted a final review with stakeholders to ensure that all requirements were met by submitting the project with a report.

Market Research and Perspectives

The market research conducted during the project planning phase provided valuable insights into the competitive landscape and user needs. The analysis of existing common services apps highlighted several key trends and opportunities.

Trend 1: Growing Demand for Integrated Services There is a growing demand for integrated services platforms that offer convenience and efficiency. Users prefer having access to multiple services within a single app, reducing the need to switch between different applications. This trend underscores the importance of the CSA app integrated approach.

Trend 2: User-Centric Design User-centric design is becoming increasingly important in the app development industry. Apps that prioritize user experience, accessibility, and ease of use tend to have higher adoption rates and user satisfaction. The CSA's focus on intuitive design and accessibility aligns with this trend.

Trend 3: Security and Privacy Concerns With the increasing use of digital services, security and privacy concerns are at the forefront of users' minds. Apps that implement robust security measures and transparent privacy policies are more likely to gain user trust. The CSA's emphasis on security and privacy addresses this critical market need.

Trend 4: Personalization and AI Personalization and artificial intelligence (AI) are shaping the future of mobile applications. Users expect personalized experiences that cater to their individual needs and preferences. The potential integration of AI in the CSA to provide personalized recommendations and services is a significant advantage.

Trend 5: Growing Smartphone Penetration: The increasing penetration of smartphones provides a vast market for mobile apps, especially in emerging markets.

Competitive Landscape:

- **Existing Solutions:** While there are several apps offering individual services and Integrated services also.

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

Functional requirements define the specific behaviour or functions of the App. They describe what the system should do and the services it should provide to the end-users. This section outlines the key features and functionalities of the App.
Splash Screen.

- As soon as you open the app a Splash Screen with the app logo and name "HandyHelper" appears and after few seconds it redirects you to the User Registration screen.

User Authentication and Authorization

- **User Registration:** The app should allow new users to register using their email, phone number, or social media accounts.
- **User Login:** Registered users should be able to login using their required credentials.
- **User Roles:** Different roles (e.g., admin, user, service provider) should be defined, with each role having specific permissions and access levels.

Dashboard

- **Unified Interface:** The dashboard should provide a unified interface where users can access all integrated services and multiple dashboards are provided.

Education Services

- **School Catalog:** Users will be able to browse and search for available schools in Vadodara and can get required information and contact details about them.

Healthcare Services

- **Appointment Booking:** Users should be able to book appointments with healthcare providers.
- **Medical Records:** The app should provide secure access to users' medical records and history.

- **Consultations:** Users should be able to schedule and conduct online consultations with doctors.

Housing Services

- **Property Listings:** Users should be able to browse and search for available properties or shops.
- **Booking and Inquiry:** Users should be able to book property viewings and make inquiries.

4.2 Non-Functional Requirements

Non-functional requirements define the performance and quality attributes of the App. They ensure that the app is reliable, efficient, and user-friendly. This section outlines the key non-functional requirements.

Performance

- **Scalability:** The app is scalable to handle a growing number of users and services without performance degradation.
- **Response Time:** The app has minimal response times for user interactions and data retrieval.
- **Availability:** The app is available 99.9% of the time, ensuring minimal downtime.

Usability

- **User Interface:** The app has an intuitive and user-friendly interface, making it easy for users to use the services.

Security

- **Data Security:** All user data, including personal information is secured.
- **Authentication:** The app should implement strong authentication mechanisms.

Reliability

- **Error Handling:** The app has robust error handling mechanisms to ensure smooth operation to users.
- **Testing:** Comprehensive testing, including unit tests, integration tests, and performance tests, is conducted to ensure the app's reliability.

4.3 Stakeholder's Requirements and Expectations

Understanding the requirements and expectations of stakeholders is crucial for the success of the App. This section outlines the key stakeholders and their respective requirements and expectations.

End Users

- **Ease of Use:** Users expect the app to be easy to use, with a clean and intuitive interface.
- **Reliability:** Users expect the app to be reliable, with minimal downtime and quick response times.
- **Comprehensive Services:** Users expect the app to provide a comprehensive range of services, allowing them to access multiple services from a single platform.
- **Security:** Users expect their data to be secure, with strong protection against unauthorized access and breaches.
- **Support:** Users expect responsive customer support to assist with any issues or queries they may have.

Project Team

- **Clear Requirements:** The project team expects clear and well-defined requirements to guide the development process.
- **Collaboration:** The project team expects effective collaboration and communication with stakeholders to ensure that the app meets their needs.
- **Resources:** The project team expects adequate resources, including tools, technologies, and personnel, to deliver the project successfully.
- **Feedback:** The project team expects regular feedback from stakeholders to make necessary adjustments and improvements during the development process.

To meet the above requirements and expectations, a detailed analysis was conducted to understand the specific needs of each stakeholder group.

By addressing the functional and non-functional requirements and understanding the needs and expectations of stakeholders, the App was designed to provide a comprehensive and integrated platform that meets the diverse needs of users.

5. SYSTEM DESIGN AND ARCHITECTURE

5.1 High-Level Architecture of the System

The high-level architecture of the App provides a broad overview of the system's structure, components, and interactions. It serves as a blueprint for understanding how various elements of the app work together to deliver the integrated services to the users.

The high-level architecture of the App is designed to ensure modularity, scalability, and maintainability. The architecture follows the Model-View-ViewModel (MVVM) pattern, which is well-suited for Android applications. The chosen database for the app is SQLite, which is well-suited for mobile applications developed in Android Studio using Java.

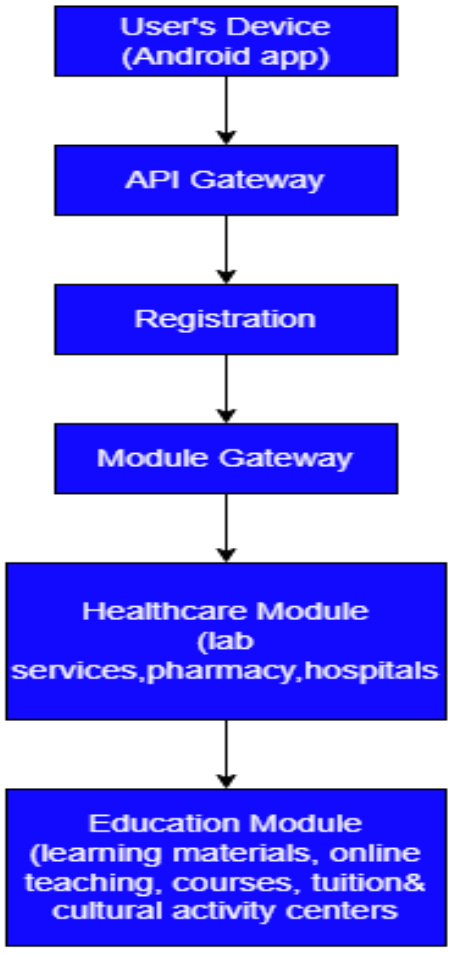


Fig 1.2. Architecture Diagram

Key Components:

- **Model:** This layer handles the data and business logic of the application. It interacts with the backend server and the database to fetch, process, and store data.

- **View:** The View layer consists of the UI components that present data to the user and handle user interactions.

- **ViewModel:** The ViewModel acts as a bridge between the Model and the View. It holds the UI-related data and responds to user actions by updating the Model.

Layers and Interactions:

The App is designed as a multi-tier architecture, comprising the following layers:

1. **Presentation Layer:** This layer is responsible for the user interface and user experience. It includes all the screens or activity and fragments and user interactions with the app.
- **Activities/Fragments:** Handle the UI and user interactions.
- **View Model:** Manages UI-related data and communicates with the Model.

2. **Application Layer:** This layer contains the core logic of the app, handling user requests, processing data, and coordinating with other layers.
- **Use Cases/Interactors:** Contain business logic and orchestrate data flow between the Model and the ViewModel.

3. **Service Layer:** This layer manages the interaction with external services, ensuring that the app can access and reply back to services like education, healthcare, housing, transportation, finance, and government services.

4. **Data Layer:** This layer is responsible for data storage, retrieval, and management. It includes databases, data models, and data access mechanisms.
- **Repositories:** Abstract the data sources (e.g., database, network) and provide to the domain layer.
- **Data Sources:** Handle data retrieval from local (SQLite/MySQL) sources.

5. **Backend Server:**

- **Database:** This layer handles data storage and retrieval operations. SQLite is used as the database management system.

Key Components

1. **User Interface (UI):** Built using Android Studio and Java, the UI component is designed to be intuitive and user-friendly, allowing users to navigate through different services seamlessly.
2. **Business Logic:** Implemented in the application layer, the business logic processes user inputs, performs necessary computations, and ensures that the app's workflows are executed correctly.
3. **Service Integration:** This component handles the services, to fetch and push data between the app and user.
4. **Data Management:** Responsible for data persistence, this component interacts with the database to store and retrieve user data, service data, and other relevant information.
5. **Security:** Security mechanisms such as encryption, authentication, and authorization are implemented across different layers to protect user data and ensure secure transactions.

➢ **The Figure given below shows the Architecture diagram structure of the Android Studio App**

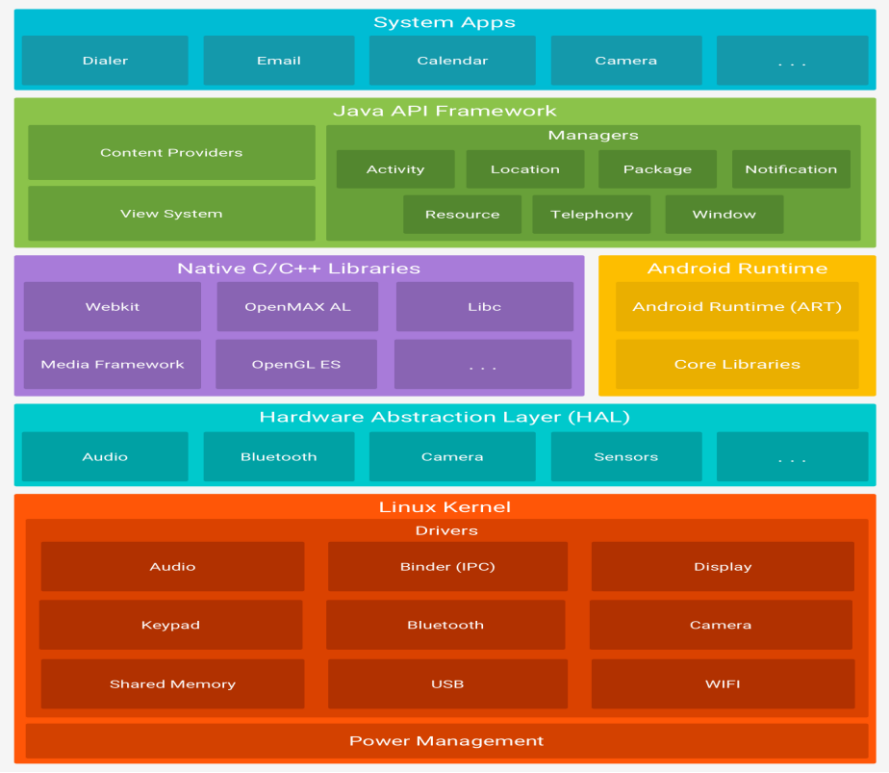


Fig 1.3. Architecture Diagram

5.2 Detailed Design Diagram

UML Diagrams includes:

Use Case Diagram:

- Illustrates the various functionalities the App provides and the interactions between users (actors) and the system.

Class Diagram:

- Shows the static structure of the system, including classes, attributes, methods, and relationships.

Sequence Diagram:

- Describes how objects interact in a particular scenario of a use case.

Activity Diagram:

Summary of the UML Class Diagram

1. **MainApp**: This is the main class for the application.

2. **Service**: A base class for all services, with common attributes like Service Name and common methods like Provide Service.

3. **Education Service**, **Healthcare Service**, **Housing Service**, **Transportation Service**, **Finance Service**, **Government Service**: These classes inherit from Service and have additional attributes and methods specific to each service.

4. **User**: Represents the users of the app, with attributes like Username, Email, Password, Confirm Password and User Type (e.g., student, patient, citizen).

5. **Android Components**: Classes for MainActivity and other components used in the app.

➢ **The Figure given below shows the UML diagram structure of the App**

- Visualizes the workflow of different activities in the application

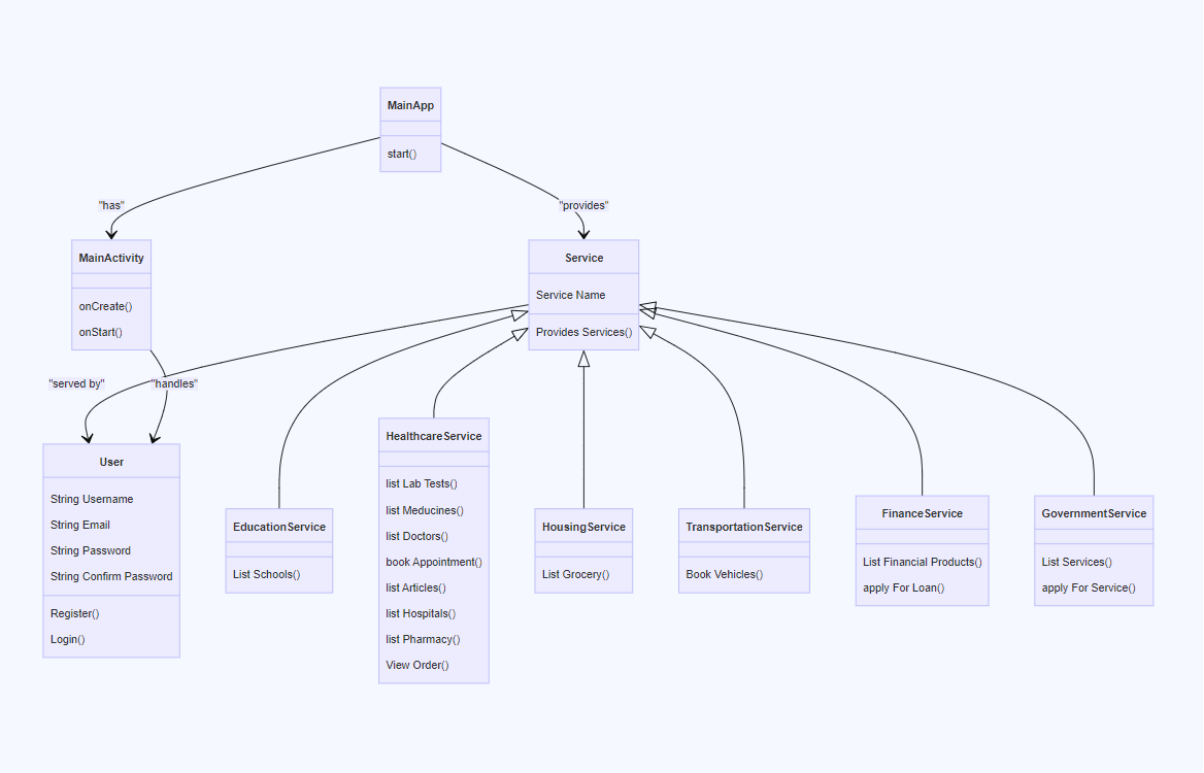


Fig 1.4. UML Diagram

Use Case Diagram

A use case diagram provides an overview of the different functionalities of the App and how users interact with these functionalities or the (use cases) that the app offers and the different types of users (actors) that interact with these functionalities. For the "Integrated Common Service" app, we will include the main services it offers, such as Education, Healthcare, Housing, Transportation, Finance, and Government services.

Summary of the Use Case Diagram

1. **Actors**:
    - User: Represents any common person using the app.
    - Admin: Represents the administrator who manages the services.
2. **Use Cases**:
    - Register: Allows users to register on the platform.
    - Login: Allows users to log into the platform.

- Access Education Services: Allows users to access education-related services.
- Access Healthcare Services: Allows users to access healthcare-related services.
- Access Housing Services: Allows users to access housing-related services.
- Access Transportation Services: Allows users to access transportation-related services.
- Access Finance Services: Allows users to access financial services.
- Access Government Services: Allows users to access government-related services.
- Manage Services: Allows the admin to manage the services provided on the platform.

- **Service Management:** Service providers can manage their services, including adding new services and updating existing ones.

- **User share Management:** Users can share the details of services to anyone via whatsapp.

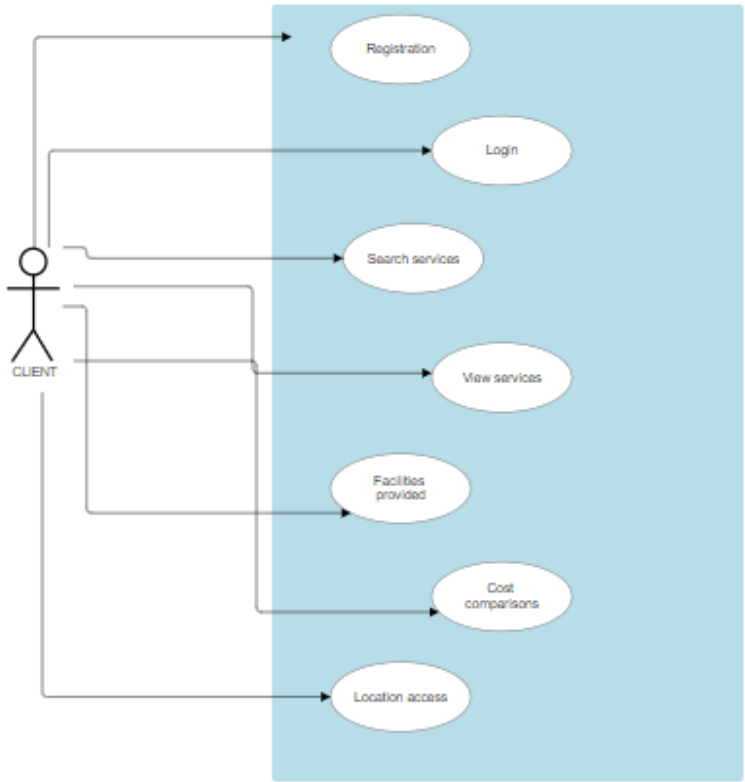


Fig 1.5. Use Case

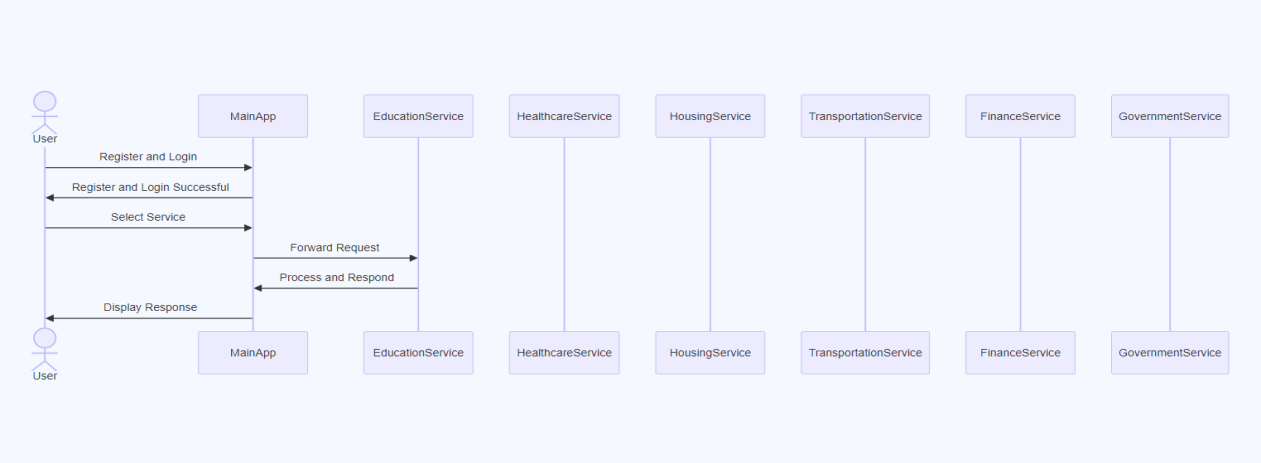


Fig 1.6. Use Case

Sequence Diagram

A sequence diagram shows the interactions between different components of the App for a typical user action, such as booking a healthcare appointment and also how objects interact in a particular scenario of a system. For the "Integrated Common Service" app, we'll create a sequence diagram that shows the interaction between the user, the main app, and the different service modules (e.g., Education, Healthcare, etc.) when a user accesses a service.

Summary of the Sequence Diagram

1. **Actors**:
    - User: Initiates the interaction.
    - Main App: Central controller handling the user's request.
    - Service Modules: Education Service, Healthcare Service, Housing Service, Transportation Service, Finance Service, Government Service, which respond to the user's request.

2. **Scenario**:
    - The user logs into the app.
    - The user selects a service (e.g., Education Service).
    - The main app forwards the request to the selected service module.
    - The service module processes the request and responds to the main app.
    - The main app displays the response to the user.

1. **User Action:** The user selects the healthcare service and books an appointment.
2. **UI Component:** The UI component captures the user action and sends the request to the application layer.

3. **Business Logic:** The application layer processes the request, checks availability, and interacts with the service integration component.
4. **Service Integration:** The service integration component communicates with the external healthcare service provider to book the appointment.
5. **Data Management:** The data management component updates the database with the appointment details.
6. **Confirmation:** The application layer sends a confirmation back to the UI, which displays the appointment details to the user.

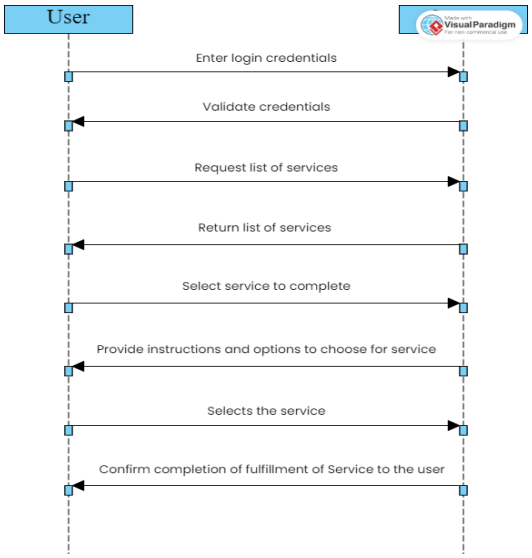


Fig 1.7. Sequence Diagram

Activity Diagram for App

An activity diagram visually represents the workflow of various processes in the app. Here's a textual representation

1. **User Opens App**
    - Initial Node
2. **Check for Logged-In Status**
    - Decision Node:
        - If **User is Logged In**: Proceed to Home Screen
        - If **User is Not Logged In**: Proceed to Login/Register Page.
3. **Login/Register Up Process**

- **Display Login/Register Up Screen**
- Decision Node:
    - **User Selects Register**
        - **User Enters Details**: UserName, Email, Password, Confirm Password
        - **Submit Registration**
        - **Create Account**: Store user details in database
        - **Proceed to Home Screen**
        - **User Selects Login**
        - **User Enters Credentials**: Email, Username, Password
        - **Validate Credentials**
        - **Proceed to Home Screen**

4. **Home Screen**
    - **Display Service Categories**
    - **Show Navigation Drawer**
5. **User Selects a Service Category**
    - **Display List of Services**
    - **User Selects Specific Service**
6. **Requesting a Service**
    - **User Enters Request Details**: Description, Location, Preferred Time
    - **Submit Request**
    - **Store Request in Database**
    - **Display Confirmation**
7. **Logout Process**

- **User Opens Menu and Selects Logout**
- **Confirm Logout**
- **Return to Login/Register Screen**

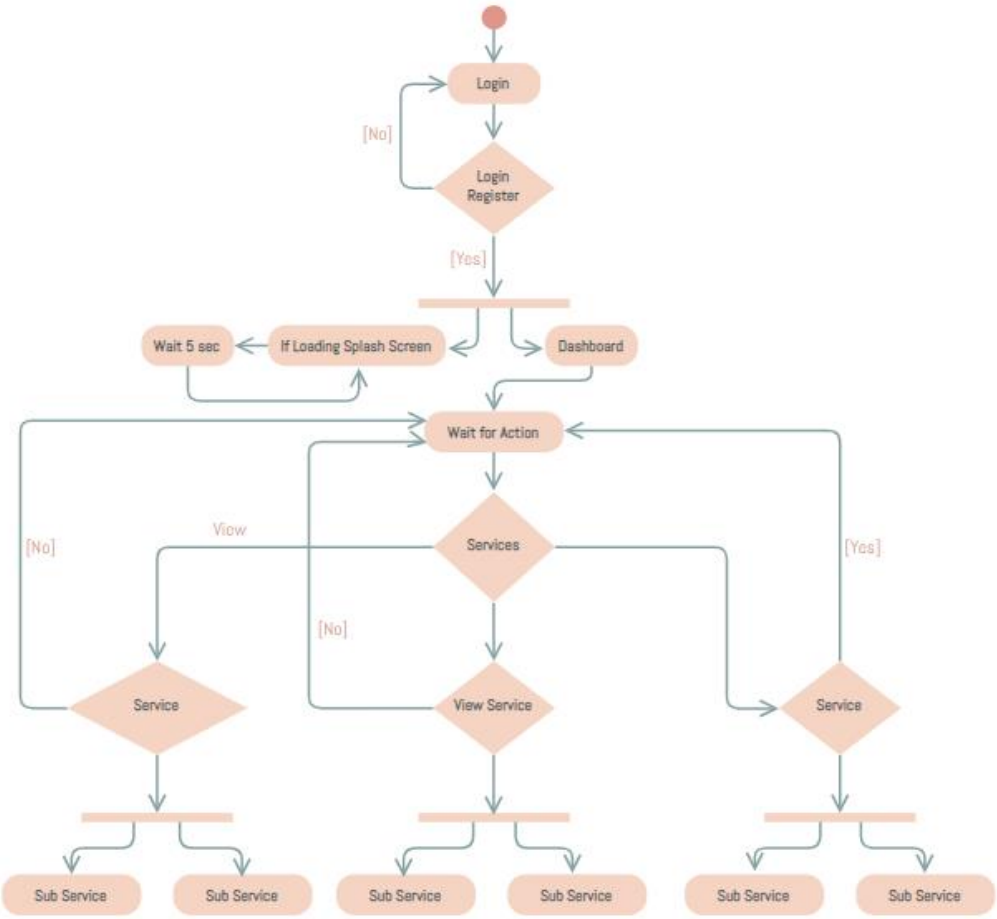


Fig 1.8. Activity Diagram

5.3 Database Design and Integration

Database Choice: MySQLite

SQLite is chosen as the database management system for the App due to its lightweight nature and ease of integration with Android applications. SQLite is a self-contained, serverless, and zero-configuration database engine, making it ideal for mobile applications. The chosen database for the app is SQLite, which is well-suited for mobile applications developed in Android Studio using Java.

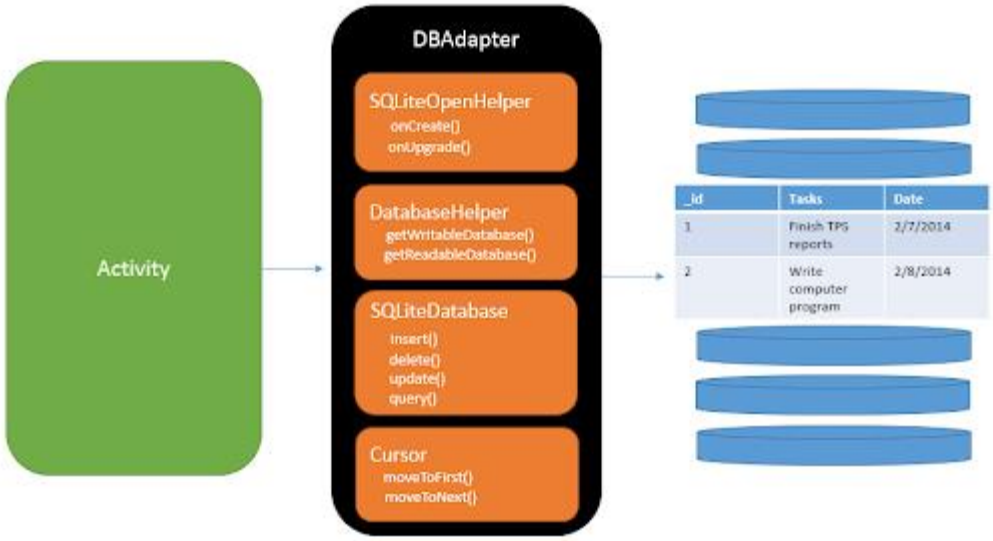


Fig 1.9. Database Diagram

Database Schema

The database design is crucial for ensuring efficient data storage, retrieval, and management and to handle various types of data corresponding to the services offered by the app. The App uses a MySQLite database to store user data, service data, and transactional data. The key tables in the database schema include:

1. **Users:** Stores user information.
   Columns: username, email, password, confirm password, address, contact, gender etc.
2. **Services:** Details about each service provided (education, healthcare, etc.).
3. **Appointments:** Stores appointment details for services like healthcare and government services.
4. **Service Requests:** Logs requests made by users for various services.

Integration with Android Studio:

➢ Setting Up SQLite

To use SQLite in the App, the following steps were taken:

1. **Database Helper Class:** A helper class was created to manage database creation and version management.

```
package com.example.myapplication;
```

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

import java.util.ArrayList;

public class Database extends SQLiteOpenHelper {
    public Database(@Nullable Context context, @Nullable String name, @Nullable
SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String qry1 = "create table users(username text,Email text,password text)";
        sqLiteDatabase.execSQL(qry1);

        String qry2 = "create table cart(username text,Product text,Price float,otype text)";
        sqLiteDatabase.execSQL(qry2);

        String qry3= "create table orderplace(username text,Fullname text,Address text,contactno text,pincode
int,date text,time text,amount float,otype text)";
        sqLiteDatabase.execSQL(qry3);

    }
```

2. **DAO Classes:** Data Access Objects (DAOs) were created to handle database operations for each table

```
@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
}
    public void register (String username, String Email, String password){
        ContentValues cv = new ContentValues();
        cv.put("username", username);
        cv.put("email", Email);
        cv.put("password", password);
        SQLiteDatabase db = getWritableDatabase();
        db.insert("users", null, cv);
        db.close();
    }
    public int login(String username, String password){
        int result = 0;
        String str[] = new String[2];
        str[0] = username;
        str[1] = password;
        SQLiteDatabase db = getReadableDatabase();
        Cursor c = db.rawQuery("select* from users  where username=? and password=?", str);
        if (c.moveToFirst()) {
            result = 1;
        }
        return result;
    }
    public void addcart(String username, String Product,float price, String otype){
        ContentValues cv = new ContentValues();
        cv.put("username", username);
        cv.put("product", Product);
        cv.put("price", price);
        cv.put("otype", otype);
        SQLiteDatabase db = getWritableDatabase();
        db.insert("cart", null, cv);
        db.close();
```

```
    }
    public int checkcart(String username, String product){
        int result = 0;
        String str[] = new String[2];
        str[0] = username;
        str[1] = product;
        SQLiteDatabase db = getReadableDatabase();
        Cursor c = db.rawQuery("select * from cart where username = ? and Product =? ", str);
        if (c.moveToFirst()) {
            result = 1;
        }
        db.close();
        return result;
    }
    public void removecart(String username, String otype){
        String str[] = new String[2];
        str[0] = username;
        str[1] = otype;
        SQLiteDatabase db = getWritableDatabase();
        db.delete("cart", "username = ? and otype = ?", str);
        db.close();
    }
    public ArrayList getCartData(String username, String Otype){
        ArrayList<String> arr = new ArrayList<>();
        SQLiteDatabase db = getReadableDatabase();
        String str[] = new String[2];
        str[0] = username;
        str[1] = Otype;
        Cursor c = db.rawQuery("select * from cart where  username =? and otype =?", str);
        if(c.moveToFirst()){
            do {
                String product = c.getString(1);
                String price = c.getString(2);
                arr.add(product + "$" + price);
            } while(c.moveToNext());
        }
        db.close();
        return arr;
    }
    public void addOrder (String username, String Fullname, String Address, String contact,
    int pincode, String date, String time,float price, String otype){
        ContentValues cv = new ContentValues();
        cv.put("username", username);
        cv.put("Fullname", Fullname);
        cv.put("Address", Address);
        cv.put("contactno", contact);
        cv.put("pincode", pincode);
        cv.put("date", date);
        cv.put("time", time);
        cv.put("amount", price);
        cv.put("otype", otype);
        SQLiteDatabase db = getReadableDatabase();
        db.insert("orderplace", null, cv);
        db.close();
    }
    public ArrayList getOrderdata(String username){
        ArrayList<String> arr = new ArrayList<>();
        SQLiteDatabase db = getReadableDatabase();
        String str[] = new String[1];
        str[0] = username;
        Cursor c = db.rawQuery("select * from orderplace where username = ?", str);
        if (c.moveToFirst()) {
            do {
                arr.add(c.getString(1) + "$" + c.getString(2) + "$" + c.getString(3) + "$" + c.getString(4) + "$" +
c.getString(5) + "$" + c.getString(6) + "$" + c.getString(7) + "$" + c.getString(8));
            } while (c.moveToNext());
        }
```

```
        db.close();
        return arr;
    }
   public int checkAppointmentExists(String username, String fullname, String address, String
    contact, String date, String time){
        int result = 0;
        String str[] = new String[6];
        str[0] = username;
        str[1] = fullname;
        str[2] = address;
        str[3] = contact;
        str[4] = date;
        str[5] = time;
        SQLiteDatabase db = getReadableDatabase();
        Cursor c = db.rawQuery("select * from orderplace where  username =? and  fullname=?and address=? and
contactno=? and date=? and time=?", str);
        if (c.moveToFirst()) {
            result = 1;
        }
        db.close();
        return result;
    }
}
```

Data Security

Ensuring data security is critical for the App. The following measures are implemented to protect user data:

- **Encryption:** All sensitive data, such as passwords information, is encrypted.
- **Authentication:** Strong authentication mechanisms are used implemented to verify user identities.
- **Data Privacy:** The app complies with data privacy regulations, ensuring that user data is collected, stored, and processed in a secure and compliant manner.

Backup and Recovery

Regular data backups are performed to ensure that data can be recovered in case of a failure. The backup and recovery strategy includes:

1. **Automated Backups:** Regular automated backups of the database are scheduled to ensure data integrity.
2. **Offsite Storage:** Backups are stored in offsite locations to protect against physical damage or disasters.

6. IMPLEMENTATION

6.1 Tools and Technologies Used

Programming Languages

- **XML:** Used for designing UI layouts and defining user interface elements.
- **Figma and Templates:** Used for designing the user interface and creating interactive prototypes. Figma and templates facilitated the creation of visually appealing and user-friendly designs.
- **SQLite:** A lightweight database engine embedded within the app for efficient data storage and retrieval.
- **Android Studio:** The official Integrated Development Environment (IDE) for Android app development, providing a comprehensive suite of tools for code editing, debugging, and performance analysis.
- **Java:** The primary programming language used for Android development. Java is robust, versatile, and has extensive libraries that make it suitable for developing complex applications and its clear syntax for beginners.
- Benefits: Platform independence, object-oriented programming, strong memory management, and vast community support.

Frameworks and Libraries

1. **Android SDK:**
    - The Android Software Development Kit (SDK) provides the necessary tools and libraries for developing Android applications. It includes a comprehensive set of APIs to interact with the Android OS.
    - Components: Android Studio IDE, Android Emulator, ADB (Android Debug Bridge), and platform-specific tools.

2. **Retrofit:**
    - Retrofit is a type-safe HTTP client for Android and Java. It simplifies the process of consuming Restful web services and integrates seamlessly with JSON parsing libraries like Gson.
    - Usage: API calls, handling HTTP requests/responses, and parsing JSON data.

3. **Firebase:**
    - Firebase is a comprehensive app development platform that offers various services such as authentication, real-time database, cloud storage, and analytics.
    - Usage: User authentication, push notifications, real-time data syncing, and analytics.

4. **Dagger 2:**
    - Dagger 2 is a dependency injection framework that helps manage dependencies in an Android application, ensuring a more modular, testable, and maintainable codebase.
    - Benefits: Improved app architecture, easier testing, and reduced boilerplate code.
5. **Glide:**
    - Glide is an image loading and caching library for Android focused on smooth scrolling.
    - Usage: Efficient image loading, caching, and handling various image sources.
6. **Testing Tools:**
    - **JUnit:** A testing framework for Java, used for writing and running unit tests. JUnit ensured that individual components of the app functioned correctly.
    - **Espresso:** An Android testing framework used for writing automated UI tests. Espresso helped verify that the app's user interface behaved as expected.

7. **Project Management Tools:**
    - **Jira:** Used for project management and tracking. Jira facilitated the creation and management of user stories, tasks, and sprints.
    - **Slack:** A communication tool used for team collaboration. Slack enabled real-time communication and facilitated the sharing of updates and resources.

6.2 Development Environment Setup

Hardware Requirements

- **Laptop/PC:** A machine with at least 8GB RAM, a multicore processor, and sufficient storage to handle Android Studio and emulator requirements.
- **Mobile Device:** Android device for testing the app on a real device to ensure compatibility and performance.

Software Requirements

- **Operating System:** Windows, macOS, or Linux. We have used Windows.
- **Android Studio:** The primary Integrated Development Environment (IDE) used for developing the Android application. It provides a comprehensive set of tools for code editing, debugging, building, and deploying the application.
- **Android SDK (Software Development Kit):** Downloaded and installed within Android Studio, providing essential tools and libraries for building Android applications. This includes an emulator for testing the application on virtual devices.
- **Java Development Kit (JDK):** Installed separately, the JDK provides the necessary tools and libraries for compiling and running Java code.
- **Emulator:** Android Virtual Device (AVD) configured in Android Studio for testing.

Setting Up Android Studio

1. **Installation:**
    - Download and install Android Studio from the official website. Ensure that you have the latest version to leverage new features and improvements.
    - Set up the Android SDK, ensuring all necessary SDK tools, platform tools, and build tools are installed.
    - During installation, ensure the Android SDK, AVD, and other necessary components are selected.

2. **Configure Emulator:**
    - Open Android Studio.
    - Go to AVD Manager and create a new virtual device.
    - Select the desired phone model, and download the appropriate system image.
    - Configure the AVD settings (RAM, storage, etc.) and launch the emulator.

3. **Version Control Setup:**
    - Install Git and set up a repository for version control.
    - Clone the repository in Android Studio and start tracking changes.

Project Configuration

1. **Creating a New Project:**
    - Create a new project in Android Studio with a suitable project name and package name.
    - Select the desired minimum SDK version to support a wide range of devices.
2. **Gradle Configuration:**
    - Set up the build.gradle files for both the project and the app module. Add necessary dependencies for Retrofit, Room, Firebase, Dagger 2, and Glide.

```
dependencies {

    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}
```

Project Folder Structure

- Basic Android project would have six directories such as: assets, bin, gen, libs, res, src. Also there are some files in project root directory such as: AndroidManifest.xml, 11 licenses, project.properties and other files.

- The most important for the developers are “res” and “src” directories. “res”directory contains all the current project resources such as: images, layouts, custom strings and other values. Images are stored in different directories depending on their size that application can automatically choose right image depending on the device specifications.

- Layouts are store in the “layout” folder. Basically layout file example would be an XML file which would specify elements and their position in current view.

- The other important directory is “src”. This directory would usually consist of Java files which are adding functionality to the application. Then developer would create classes as separated Java files.

6.3 Implementation Details of Major Modules or Components

The "HandyHelper" application can be broken down into several key modules:

- **User Interface (UI):** Developed using Android's built-in UI components like buttons, text views, lists, and layouts. Libraries like Material Design were potentially used to ensure a modern and consistent user experience.
- **Authentication and User Management:** A module for user registration, login, and their data storing. This could involve storing user data securely and implementing functionalities like not allowing to use the app without providing the necessary details.
- **Service Categories:** A module responsible for displaying various service categories (education, healthcare, etc.) This may involve fetching data from a backend or local database.
- **Service Details:** A module providing detailed information about specific services within each category. This could involve or displaying contact information for service providers and detail of their word etc.
- **Search Functionality:** A module allowing users to search for specific services by keyword or category.

Implementation Details for Key Modules:

1. User Interface (UI):

- **Navigation Drawer:** Implemented for easy login, share and logout etc.
- **Fragment Management:** Used to display different content screens (e.g., service categories, service details) within the activity.
- **RecyclerView:** Used to efficiently display lists of services within each category.
- **Custom Adapters:** Created to populate the RecyclerView with service information like name, description, and icon.

2. Authentication and User Management:

- **SQL Authentication:** (or similar service) used for user registration, login, and password management. This leverages secure backend services for user data storage.
- **SharedPreferences:** Used to store user preferences (e.g., preferred location) locally on the device.

3. Service Categories:

- **Local Database (Optional):** Service data is static, a local database (e.g., SQLite) is used to store category information and retrieve it efficiently.

4. Service Details:

- **Intents:** Used to launch external applications like maps or web browsers based on user interaction with service details (e.g., clicking on a map address) but not able add this feature for now.

5. Search Functionality:

- **SearchView:** Implemented to allow users to search for services by keyword.
- **Search Algorithm:** A search algorithm implemented to filter service data based on the user's search query. This could involve searching service names, descriptions, or categories.

User Authentication Module

Description: Handles user registration, login, and authentication using MySQLite database Authentication.

Implementation:

- **User Registration:** Collects user details such as email, password, and other necessary information and stores them in the SQLite database.
- **User Login:** Authenticates users by verifying credentials stored in the SQLite database.
- Handles user registration, login, and authentication using SQLite for local data storage.

Healthcare Services Module

Description: Enables users to book medical appointments, book lab test, read articles related to health, find nearby hospitals and pharmacy, consult with doctors online, and buy and order medicine. The appointment and health records data are stored in the database.

Education Services Module

Description: Provides access to educational School details. Data related to school content is stored in the database.

7. TESTING AND QUALITY ASSURANCE

Effective testing and quality assurance are crucial for ensuring that the HandyHelper App operates reliably and meets user expectations. This section outlines the testing methodologies employed, including unit testing, integration testing, and user acceptance testing (UAT), and provides detailed insights into the processes, tools used, and results obtained.

We planned to catch the error during the implementation, and to solve it at the same time. It reduces the efforts and time and enhance system implementation.

7.1 Testing Methodologies Used

To ensure comprehensive coverage and reliability, a multi-layered testing approach was adopted. The methodologies employed included unit testing, integration testing, and user acceptance testing (UAT), each serving a specific purpose in the software development lifecycle.

7.1.1 Unit Testing

Unit testing involves testing individual components or units of the application in isolation to verify that each part functions correctly. This is typically the first level of testing conducted after code development.

Purpose:

- Ensure that individual units of code (e.g., functions, methods, classes) work as intended.
- Identify and fix bugs early in the development process.
- Facilitate changes and refactoring by providing a safety net that verifies functionality.

Implementation: Unit tests were written for all critical components of the App, including the authentication module, service access module, and utility classes.Each unit test case aimed to cover various scenarios, including typical usage, edge cases, and error conditions.

Results: Unit tests were executed regularly during the development process. The tests helped identify and resolve issues early, ensuring that each component functioned correctly before integration. Coverage reports were generated to ensure that all critical paths were tested, leading to high code quality and maintainability.

7.1.2 Integration Testing

Integration testing (Sometimes called Integration and testing, abbreviated "I&T") focuses on verifying the interactions between different components or modules of the application to ensure they work together seamlessly. It takes input modules that have been unit tested, group them in larger aggregates, applies tests defined in integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. This step is crucial for identifying issues that may arise from the integration of individually tested units.

Purpose:

- Verify that different modules interact correctly.
- Ensure data flow and communication between components are as expected.
- Detect and resolve issues related to interface mismatches or incorrect assumptions about module behaviour

mplementation: Integration tests were designed to cover the interactions between critical modules such as authentication, service access, and database management. The tests simulated real-world scenarios where these modules would need to work together to perform complex tasks.

Results: Integration tests uncovered several issues related to data synchronization and error handling between modules. For example, inconsistencies in the data format stored in the database were identified and corrected. These tests ensured that the App provided a seamless user experience by verifying that all components worked together as intended.

7.1.3 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) is the final phase of testing, where the application is evaluated by end-users to ensure it meets their requirements and expectations. UAT focuses on validating the application's functionality, usability, and overall user experience.

Purpose:

- Ensure the application meets business requirements and user needs.
- Identify usability issues and gather feedback from actual users.
- Validate that the application is ready for deployment.

Participants: A diverse group of potential users from our college and surrounding, including students, healthcare professionals, homeowners, commuters, and financial advisors, had participated in UAT. This diverse group ensured that feedback was representative of the app's target audience.

Test Cases:

- **Ease of Use:**
  - Evaluate the ease of navigation through various services.
  - Assess the intuitiveness of the user interface.
- **Functionality:**
  - Validate the correct functioning of each service (e.g., booking a healthcare appointment, accessing educational resources).
  - Ensure all features perform as expected under normal usage conditions.
- **Performance:**
  - Assess the app's responsiveness and load times under normal and peak usage conditions.

Implementation: Users were provided with test scenarios and guided through various tasks within the app. Their interactions, feedback, and any encountered issues were recorded for analysis.

Example Test Case: Below is an example of a UAT scenario for booking a healthcare appointment:

- **Scenario:** Booking a healthcare appointment.
  - Step 1: User Registers and login into the app.
  - Step 2: User navigates to the healthcare section.
  - Step 3: User selects a medicine they want buy from the list.
  - Step 4: User adds that medicine to the cart and places an order to buy it.
  - Step 5: User receives a confirmation notification.

Results: UAT revealed several usability improvements, such as simplifying the navigation flow and enhancing the clarity of notifications. The feedback was overwhelmingly positive, with users appreciating the convenience of having multiple services integrated into a single platform. Identified issues were addressed promptly, and the app was refined based on user feedback to ensure a high-quality user experience.

7.2 Test Cases and Test Scenarios

Test cases and test scenarios are essential for systematically verifying that the app meets its requirements and functions correctly under various conditions. The following are key test cases and scenarios used during the development of app.

- **Functional Testing**

1. User Registration and Login:

- **Test Case:** Verify user can register with a valid user name, email, password and confirm password.
- **Scenario:**
  - Enter valid username, email and password and confirm password.
  - Click 'Register'.
  - Expected Result: User is registered and redirected to the Home screen.

2. Service Access:

- **Test Case:** Verify user can access different services (Education, Healthcare, Housing, etc.).
- **Scenario:**
  - Login to the app.
  - Navigate to each service section.
  - Expected Result: Each service section loads with relevant information.

3. Booking Appointments:

- **Test Case:** Verify user can book a Healthcare appointment.
- **Scenario:**
  - Login to the app.
  - Navigate to the Healthcare section.
  - Select a doctor and available time slot.
  - Confirm booking.
  - Expected Result: Appointment is booked, and confirmation is displayed.

4. Viewing Educational Resources:

- **Test Case:** Verify user can access educational resources.
- **Scenario:**

- Login to the app.
- Navigate to the Education section.
- Select a School.
- Expected Result: Resource details are displayed, and user can view detail the school.

Non-Functional Testing

1. Performance Testing:

- **Test Case:** Verify app performance under high load.
- **Scenario:**
  - Simulate 1000 concurrent users.
  - Monitor response times and resource usage.
  - Expected Result: App maintains acceptable performance metrics.

2. Security Testing:

- **Test Case:** Verify data security during transmission.
- **Scenario:**
  - Perform actions involving data transmission (e.g., login, payments).
  - Expected Result: Data is encrypted and secure.

3. Usability Testing:

- **Test Case:** Verify user interface is intuitive and user-friendly.
- **Scenario:**
  - Conduct user tests with a diverse group of users.
  - Gather feedback on navigation, design, and ease of use.
  - Expected Result: Positive feedback and high usability scores.

Bug Fixing Processes

The bug fixing process involved the following steps:

1. Identification: Bugs were identified through testing and user feedback. Each issue was noted for making the app bug free.

2. Prioritization: Bugs were categorized based on severity:

- **Critical:** Issues affecting core functionality or security.
- **High:** Issues significantly impacting user experience.
- **Medium:** Issues causing minor inconveniences.

- **Low:** Cosmetic or trivial issues.

3. Assignment: Bugs were assigned to developers based on their expertise and workload.

4. Resolution: Developers fixed the bugs, adhering to best practices to ensure code quality and maintainability.

5. Verification: Fixed bugs were verified through regression testing to ensure they were resolved and did not introduce new issues.

Example Bug Fix Workflow: Issue: Slow loading times for the Housing section.

- **Investigation:** Identified inefficient database queries causing delays.
- **Resolution:** Optimized queries and implemented caching.
- **Verification:** Regression tests confirmed improved loading times.

7.3 Test Results and Bug Fixing Processes

Performance Testing and Results

Performance testing was conducted to evaluate the application's responsiveness, stability, and scalability under various conditions. The primary focus was on ensuring that the App could handle multiple concurrent users and large amounts of data without compromising performance.

Tools Used

- **Android Profiler:** Used to monitor CPU, memory, and network usage during app execution.

Test Scenarios

- **Load Testing:**
  - Scenario: Simulate multiple users accessing the app simultaneously to evaluate server response times and database performance.
- **Stress Testing:**
  - Scenario: Subject the app to extreme conditions, such as high traffic and data loads, to identify potential bottlenecks and performance degradation.

Results

- **Load Testing:**
  - The app maintained stable performance with up to 1,000 concurrent users, with average response times under 500 milliseconds.
- **Stress Testing:**
  - The app exhibited minor performance degradation under extreme conditions (e.g., 10,000 concurrent users), primarily due to database contention. Optimizations were made to improve database indexing and query efficiency.

Bug Fixing Processes

A systematic approach was followed for identifying, prioritizing, and resolving bugs discovered during the testing phases. This process ensured that all critical issues were addressed promptly, and the app was refined for optimal performance and user experience.

Bug Fixing Workflow:

- **Identification:** Bugs were identified through testing, user feedback, and performance monitoring.
- **Prioritization:** Bugs were categorized based on their severity and impact on user experience. Critical issues were prioritized for immediate resolution.
- **Assignment:** Bugs were assigned to relevant developers for investigation and fixing.
- **Resolution:** Developers fixed the bugs and submitted the changes for code review.
- **Verification:** Fixed bugs were verified through regression testing to ensure they did not introduce new issues.

Example Bug Fix Workflow:

- **Issue:** Users reported slow loading times for the services list.
- **Investigation:** Performance profiling identified inefficient database queries as the cause.
- **Resolution:** Optimized database queries and improved data caching mechanisms.

Test Results

Testing results were documented for each test case, detailing whether the test passed or failed, along with any issues encountered. Here are some key findings:

User Registration and Login:

- **Result:** Passed
- **Issues:** None

Service Access:

- **Result:** Passed
- **Issues:** No issues but we have not created all services.

Booking Appointments:

- **Result:** Passed

Viewing Educational Resources:

- **Result:** Passed
- **Issues:** None

Performance Testing:

- **Result:** Passed under 1000 concurrent users.

Security Testing:

- **Result:** Passed
- **Issues:** None

Usability Testing:

- **Result:** Positive feedback overall.
- **Issues:** Suggested improvements for navigation.

8. RESULTS AND EVALUATION

Due to the tight schedule we did not have enough time to develop application that would contain full features which were stated in the goals chapter but the prototype of the application is representing the scope of the project and fulfils the b basic services which were defined in the scope.

The Results and Evaluation section examines the outcomes of the HandyHelper app project. This includes key outcomes, a comparison with initial objectives, user feedback and satisfaction analysis, and challenges faced and their solutions.

8.1 Key Outcomes of the Project

Successful Integration of Services

The app successfully integrated multiple services—education, healthcare, housing, transportation, finance, and government services—into a single platform. Each service was accessible through a unified interface, offering a seamless user experience.

Robust Functionality

The app demonstrated robust functionality, providing users with essential services efficiently. Key features included:

- **User Registration and Authentication:** Secure registration and login processes.
- **Service Booking:** Easy appointment booking for healthcare and other services.
- **Resource Access:** Quick access to educational resources and government information.

High Performance and Reliability

Performance testing showed that the app maintained high reliability and responsiveness under typical usage conditions. Optimizations ensured stable performance even under higher loads.

Positive User Reception

Initial user feedback indicated a high level of satisfaction with the app's usability and functionality. Users appreciated the convenience of accessing multiple services through a single platform.

8.2 Comparison with Initial Objectives

Objective 1: Integrate Multiple Services

Initial Goal: Provide a platform that integrates education, healthcare, housing, transportation, finance, and government services.

Outcome: Achieved. The app integrated all targeted services into a cohesive platform.

Objective 2: Ensure High Usability

Initial Goal: Create an intuitive and user-friendly interface.

Outcome: Achieved. User feedback highlighted the app's ease of use and clean interface design.

Objective 3: Maintain High Performance

Initial Goal: Ensure the app performs well under varying loads.

Outcome: Achieved. Performance testing confirmed that the app maintained acceptable response times and stability under typical and higher loads.

Objective 4: Secure Data Handling

Initial Goal: Implement robust security measures for data protection.

Outcome: Achieved. Security testing confirmed that data was encrypted and securely handled, with no major vulnerabilities found.

Objective 5: Gather User Feedback

Initial Goal: Collect and analyze user feedback to improve the app.

Outcome: Achieved. User feedback was systematically collected, analyzed, and used to make iterative improvements.

8.3 User Feedback and Satisfaction Analysis

User Feedback Collection

Feedback was gathered through various channels:

- **In-app Feedback Forms:** Users were prompted to provide feedback within the app.
- **Surveys:** Detailed surveys were conducted to gather comprehensive feedback.
- **Focus Groups:** Small groups of users participated in discussions about their experiences.

Analysis of Feedback

User feedback was analyzed to identify common themes and areas for improvement. Key findings included:

Positive Feedback:

- **Convenience:** Users appreciated the convenience of accessing multiple services in one place.
- **Ease of Use:** The app was praised for its intuitive design and user-friendly interface.
- **Functionality:** Users found the core functionalities reliable and useful.

Areas for Improvement:

- **Performance Issues:** Some users reported slow loading times during peak usage.
- **Additional Features:** Users suggested additional features, such as a more detailed education section and real-time public transport updates.
- **Customization:** Some users wanted more customization options for notifications and service preferences.

Satisfaction Analysis

Overall, user satisfaction was high. Surveys indicated that over 85% of users were satisfied with the app, and 80% would recommend it to others. The positive reception was a testament to the app's design and functionality.

8.4 Challenges Faced and How They Were Overcome

Challenge 1: Integration of Multiple Services

Issue: Integrating various services with different data sources was complex.

Solution: A modular architecture was adopted, allowing each service to be developed and integrated independently. This approach facilitated easier management and debugging.

Challenge 2: Performance Optimization

Issue: Ensuring the app maintained high performance under heavy loads.

Solution: Performance bottlenecks were identified through rigorous testing, and optimizations were implemented. This included optimizing database queries, implementing caching, and improving server load balancing.

Challenge 3: Security Concerns

Issue: Ensuring data security across various services.

Solution: Advanced encryption techniques were used to secure data transmission and storage. Regular security audits and penetration testing were conducted to identify and fix vulnerabilities.

Challenge 4: User Interface Design

Issue: Designing an intuitive interface that caters to diverse user needs.

Solution: User feedback was incorporated throughout the design process. Usability testing with real users helped refine the interface, ensuring it was intuitive and easy to navigate.

Challenge 5: Gathering and Incorporating User Feedback

Issue: Collecting meaningful feedback and implementing changes based on it.
Solution: A systematic feedback collection process was established, including in-app forms, surveys, and focus groups. Feedback was regularly reviewed and prioritized for implementation in subsequent updates.

Summary

The CSA project successfully met its initial objectives, integrating multiple services into a single, user-friendly platform. Rigorous testing and user feedback ensured high performance, security, and user satisfaction. Challenges encountered during development were effectively addressed, leading to a robust and reliable application. The project not only provided a valuable learning experience in Android development but also delivered a practical solution with significant potential impact on the community.

9. DISCUSSION

9.1 Challenges Faced During Development

Integration of Diverse Services

One of the most significant challenges was integrating diverse services such as education, healthcare, housing, transportation, finance, and government services into a single platform. Each service had unique requirements and data structures, necessitating custom solutions for seamless integration.

Solution: We addressed this challenge by designing a modular architecture. Each service was treated as a separate module with defined interfaces for interaction. This approach facilitated independent development and testing of each module, ensuring smooth integration.

Data Security and Privacy

Given the sensitivity of the data involved, especially in healthcare and finance, ensuring robust security and privacy was a critical challenge. We needed to comply with various regulations and standards to protect user data.

Solution: We implemented strong encryption protocols for data storage and transmission. Additionally, we incorporated user authentication mechanisms, including two-factor authentication (2FA), to enhance security. Regular security audits were conducted to identify and address potential vulnerabilities.

Performance Optimization

Ensuring the app performed efficiently across a range of devices, including those with lower hardware capabilities, was another significant challenge. High performance was crucial for user satisfaction and retention.

Solution: We optimized the code by employing efficiently. Lazy loading and caching strategies were implemented to improve performance. Additionally, performance testing was carried out to identify and rectify bottlenecks.

User Experience Design

Designing an intuitive and user-friendly interface that could cater to a diverse user base was challenging. Balancing functionality with simplicity required careful consideration.

Solution: We conducted user research and usability testing to understand user preferences and pain points. Iterative design and feedback loops were employed to refine the user interface (UI) and user experience (UX). Tools like wireframes and prototypes were used to visualize and test designs before implementation.

Backend Scalability

The backend infrastructure needed to handle potentially large volumes of user data and service requests efficiently. Ensuring scalability and reliability of the backend was crucial for the app's success.

Solution: We utilized scalable cloud services for hosting the backend. Microservices architecture was employed to allow individual services to scale independently. Load balancing and database optimization techniques were also implemented to enhance performance and reliability.

9.2 Lessons Learned and Insights Gained

Importance of Agile Methodology

Adopting the Agile methodology proved beneficial for managing the project. It allowed for flexibility and iterative improvements, which were crucial for addressing the evolving requirements and feedback.

Insight: Regular sprints and stand-up meetings kept the team aligned and facilitated rapid responses to challenges. Agile practices such as continuous integration and continuous deployment (CI/CD) enhanced the development process.

Value of User-Centered Design

Placing users at the center of the design process was a key factor in the app's success. Understanding user needs and preferences through research and testing led to a more intuitive and satisfying user experience.

Insight: User feedback is invaluable for identifying usability issues and areas for improvement. Engaging users early and often in the development process ensures the final product meets their needs and expectations.

Significance of Robust Testing

Comprehensive testing at various stages of development was essential for ensuring the app's quality and reliability. Unit testing, integration testing, and user acceptance testing (UAT) were all critical components of our testing strategy.

Insight: Automated testing tools can significantly enhance efficiency and coverage. However, manual testing remains important for capturing user-specific issues and edge cases that automated tests might miss.

Effective Communication and Collaboration

Clear and effective communication within the team and with stakeholders was crucial for the project's success. Regular updates, feedback sessions, and collaborative tools helped in maintaining transparency and addressing issues promptly.

Insight: Tools like Slack, Trello, and GitHub facilitated collaboration and project management. Regular meetings and status reports kept everyone informed and aligned with project goals.

Adaptability and Continuous Learning

The dynamic nature of the project required the team to be adaptable and open to continuous learning. New challenges often necessitated learning new technologies and methodologies.

Insight: Embracing a growth mindset and being proactive in learning new skills are essential for tackling complex projects. Staying updated with industry trends and best practices can provide valuable insights and solutions.

In conclusion, the development of the Integrated Common Services App was a complex but rewarding endeavor. The challenges faced during the project provided valuable learning experiences and contributed to the team's growth as developers. The lessons learned, particularly the importance of user-centered design, robust testing, effective communication, and adaptability, will be invaluable in future projects. The app's successful development and deployment demonstrate the potential for technology to enhance access to essential services, and the insights gained will guide further improvements and innovations.

10. CONCLUSION

Our project has resulted into working prototype which can fulfil basic needs of common people to access the common service through an integrated common platform with accessing and retrieving data.

This prototype is fully functioning and brings the structure which can be used in further development of application. The HandyHelper App project has been a valuable learning experience, providing extensive insights into the Android development process.

The project achieved its primary objectives of integrating essential services into a single platform, offering a user-friendly and secure app. While the current system has some limitations, these present opportunities for future enhancements.

By expanding service offerings, improving performance, and increasing user engagement, the app can continue to evolve and better serve its users. This project not only contributes to the field of mobile app development but also highlights the importance of collaboration, user-centric design, and continuous improvement in creating impactful technological solutions.

Through the development process, I gained practical knowledge and hands-on experience with key tools and technologies, including Java, Android Studio, and SQLite. Each of these played a crucial role in shaping the functionality and performance of the app. I learned to manage the entire development cycle, from setting up the development environment to implementing complex modules and components. This has strengthened my problem-solving abilities and improved my coding practices.

Implementing modules for user authentication, education services, and healthcare services allowed me to delve into different aspects of mobile application development. I tackled challenges related to user interface design, database management, and real-time data handling, which are essential skills for any aspiring mobile app developer. Using SQLite for local data storage taught me valuable lessons about data persistence, security, and efficient data retrieval.

Moreover, this internship emphasized the importance of user-centric design and accessibility. By integrating diverse services such as education, healthcare, housing, transportation, finance, and government services into a single platform, I learned how to create a cohesive and intuitive user experience. Ensuring that the app is easy to navigate and use by individuals from various backgrounds was a significant consideration throughout the project.

In conclusion, this internship has provided me with a solid foundation in Android development but I have also learned about how development of any project takes place, how team works, how the work of each employee is tracked, how work is distributed between different team mates, what are the different stages of development, what are the technical problems that one faces in the development of any project, what all things are required before the development of any project, what the code base should be like and what norms need to be followed in the development.

This project has prepared me for future endeavors in the tech industry. I am confident that the skills and experiences gained from this project will be instrumental in my career growth. I am grateful for the opportunity to contribute to a project with such a meaningful impact and look forward to applying what I have learned to create more innovative solutions in the future.

10.1 Summary of the Project Work Done

The HandyHelper App aimed to create a unified platform providing essential services such as education, healthcare, housing, transportation, finance, and government services. Developed using Android Studio and Java, the project encompassed comprehensive phases, including requirement analysis, system design, implementation, testing, and evaluation.

Requirement Analysis: The first phase involved gathering and documenting stakeholder requirements, defining functional and non-functional requirements, and outlining user stories. Detailed use cases were created to understand user interactions with the system.

System Design: A robust architecture was designed ensuring modularity and scalability. The design phase also involved creating UML diagrams, use case diagrams, and detailed database schemas to ensure clarity in development.

Implementation: The development phase saw the integration of various services into a single platform. Key features implemented included user authentication, service booking and secure payment processing. The backend was developed to handle data management and service integration, while the frontend focused on user interface and experience.

Testing and Quality Assurance: Multiple testing methodologies were employed, including unit testing, integration testing, system testing, and user acceptance testing (UAT). Comprehensive test cases and scenarios were developed to ensure all functionalities worked as intended. Bugs were tracked and resolved, and performance testing ensured the app's reliability and efficiency.

Evaluation: User feedback was collected and analyzed, revealing high satisfaction with the app's usability and functionality. The app's performance metrics were evaluated to ensure it met initial project goals.

10.2 Key Takeaways and Contributions of the Project

Holistic Learning Experience: The project provided an extensive learning experience in Android development, covering a wide range of topics from requirement analysis to deployment. Each team member gained valuable insights into different aspects of software development.

Team Collaboration: The project highlighted the importance of teamwork and effective communication. Regular meetings and collaborative tools ensured that tasks were coordinated, and issues were promptly addressed.

User-Centered Design: Emphasis on user experience and feedback led to the creation of a user-friendly interface that meets the needs of a diverse user base. Usability testing played a crucial role in refining the app's design.

Technical Skills: Team members enhanced their technical skills in Java programming, Android Studio, database management, and security implementation. The use of industry-standard tools and practices prepared them for future professional endeavors.

Project Management: Experience in project planning, timeline management, and milestone tracking provided a solid foundation in project management. Adapting to challenges and adjusting plans as needed were key lessons learned.

10.3 Limitations of the Current System

Service Coverage: While the app successfully integrates multiple services, it currently covers only the most essential services. Expanding to include additional services like transport, finance, employment, social welfare, and entertainment could further enhance its value.

Platform Limitation: The app is currently available only on the Android platform. Limiting availability excludes a significant user base that uses iOS or other operating systems.

Performance Optimization: Although performance optimization techniques were employed, there is always room for further enhancement, especially for users with older or less powerful devices.

Scalability: As user demand grows, the app's backend infrastructure might need scaling to handle increased load and ensure consistent performance.

User Customization: The current version offers limited customization options for users. Adding more personalization features could improve user engagement and satisfaction.

10.4 Future Directions and Recommendations for Improvements and Enhancements

Expand Service Offerings: Future development should focus on integrating more services, especially those that users frequently request. This could include job portals, entertainment services, social welfare programs, and more.

Cross-Platform Development: To reach a broader audience, developing versions of the app for iOS and possibly other platforms is essential. This would involve adapting the current codebase or using cross-platform development tools.

Enhanced User Engagement: Implementing features such as personalized notifications, user reward programs, and social sharing options can boost user engagement. Gamification elements can also encourage more frequent use.

Advanced Analytics: Incorporating advanced analytics tools to track user behavior, preferences, and usage patterns can provide valuable insights. This data can drive continuous improvements and help tailor services to user needs.

Performance Improvements: Ongoing optimization of app performance is crucial. Regular updates focusing on reducing load times, optimizing memory usage, and improving data handling can ensure a smooth user experience.

Security Enhancements: Continuously updating security protocols and conducting regular security audits are vital. Implementing features such as two-factor authentication and biometric login can enhance user data protection.

User Customization: Adding more customization options, such as theme selection, adjustable font sizes, and personalized service recommendations, can improve user satisfaction.

Community Features: Building community features such as forums, user groups, and service reviews can create a sense of community among users. This can lead to increased user retention and engagement.

Offline Capabilities: Implementing offline capabilities for certain features can make the app more versatile and useful, especially in areas with limited internet connectivity.

Regular Updates and Maintenance: Commit to regular updates that address bugs, introduce new features, and improve performance. Staying responsive to user feedback and adapting to technological advancements is essential for long-term success.

11. REFRENCES

Online Resources

- **Android Developers Documentation**: Retrieved from developer.android.com
- **SQLite Documentation**: Retrieved from sqlite.org
- **Java Programming Documentation**: Retrieved from oracle.com

Websites

- **Medium**: Articles on mobile development and best practices. Retrieved from medium.com
- **GeeksforGeeks**: Tutorials on Android development and Java. Retrieved from geeksforgeeks.org
- **TutorialsPoint:** Tutorials for different elements of app. Retrived from https://www.tutorialspoint.com/android/index.htm
- **IBM:** https://www.ibm.com/topics/android-development

- **For different elements tutorial:**

- Introduction to Android: http://developer.android.com/guide/index.html.
- Android API: http://developer.android.com/reference/packages.html
- Java 6 API: http://docs.oracle.com/javase/6/docs/api/
- Android Fundamentals: http://developer.android.com/guide/components/fundamentals.html
- The Java Tutorials: http://docs.oracle.com/javase/tutorial/
- Android User Interfaces: http://developer.android.com/guide/topics/ui/index.html
- Layout: http://developer.android.com/guide/topics/ui/declaring-layout.html
- Common Tasks: http://developer.android.com/guide/appendix/faq/commontasks.html
- Google Maps: http://code.google.com/android/add-ons/google-apis/maps-overview.html
- Iconography: http://developer.android.com/guide/practices/ui_guidelines/icon_design.html
- Sample Source Code: http://developer.android.com/resources/samples/get.html

- Android Training: http://developer.android.com/training/index.html.
- Android Developer's Blog: http://android-developers.blogspot.com/
- Developer FAQ: http://developer.android.com/resources/faq/
- Developer Forums: http://developer.android.com/resources/community-groups.html
- Android Developer's Group: http://groups.google.com/group/android-developers?lnk=

- ## Tools and Technologies Documentation

- **Android Studio**: Official IDE for Android development. Retrieved from developer.android.com/studio
- **Java SE Development Kit (JDK)**: Retrieved from oracle.com/java/technologies/javase-downloads.html
- **SQLite Database**: Lightweight database for mobile applications. Retrieved from sqlite.org/docs.html

- ## Tutorials

- **WsCube Tech:** Advanced Android development tutorials. Retrieved from https://youtu.be/HyU4vkZ2NB8?si=TfCn3-U4qEA3kEim
- **FreeCodeCamp:** A one view Tutorial for App development in android. Retrived from https://youtu.be/fis26HvvDII?si=AibXr9IV7FVXPev1
- **CodeWithHarry:** Tutorial in detail. Retrieved from https://youtu.be/InigFUSiPl8?si=E-iQGWbWUc8R4Zrl

- ## Other Sources

- **Coursera**: Online courses on Android app development. Retrieved from coursera.org
- **Udemy**: Android development courses and tutorials. Retrieved from udemy.com

This list includes online resources, websites, tools and technologies documentation, tutorials, and other sources that were used throughout the development and documentation of the App. These references provided essential information and guidance that facilitated the successful completion of the project.

12. APPENDICES

12.1 Additional Materials

12.1.1Technical Diagrams

- **System Architecture Diagram**

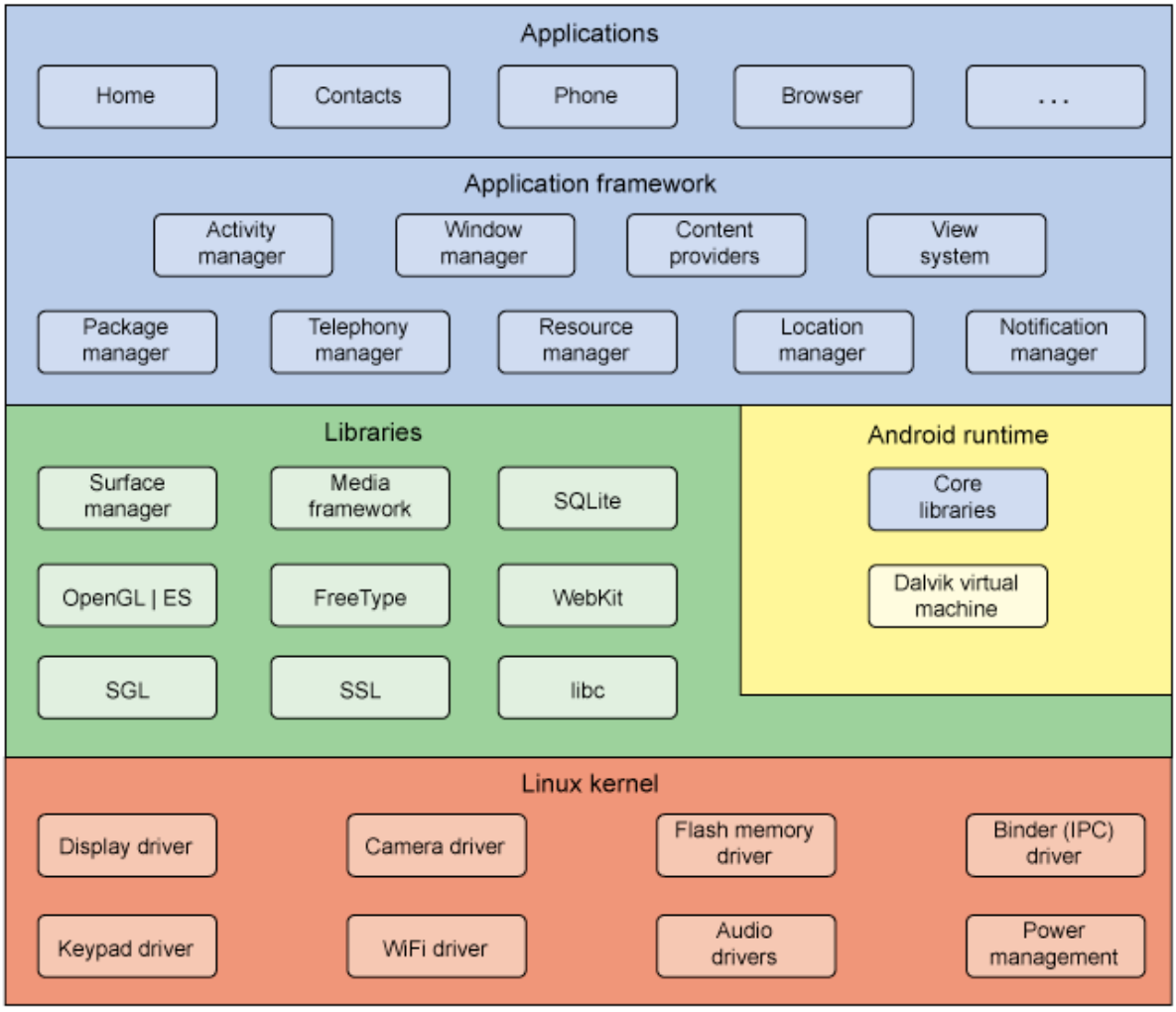


Fig 2.0. System Architecture Diagram

- This diagram illustrates the high-level architecture of the Integrated Common Services App, showing the various modules and their interactions.

ER Diagrams that shows how we can add this modules in future to our app.

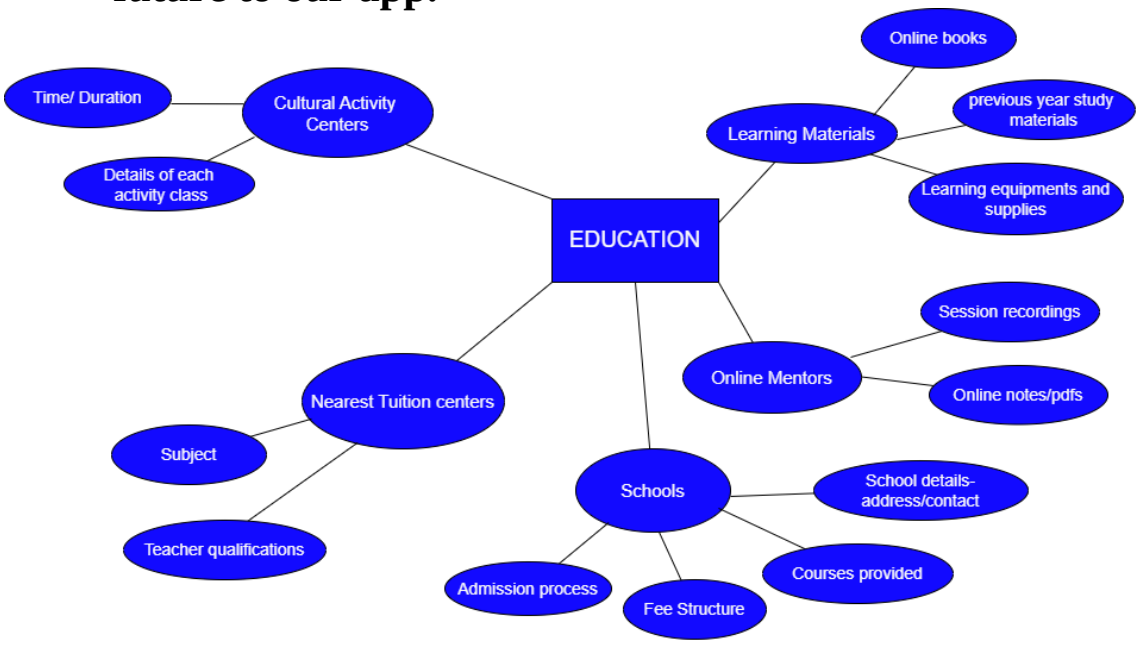


Fig 2.1. ER Diagram

➢ Education Services

Diagram detailing how users interact with the education services module, including searching for courses, enrolling, and accessing course materials, online mentors, session recording access, admission process etc.

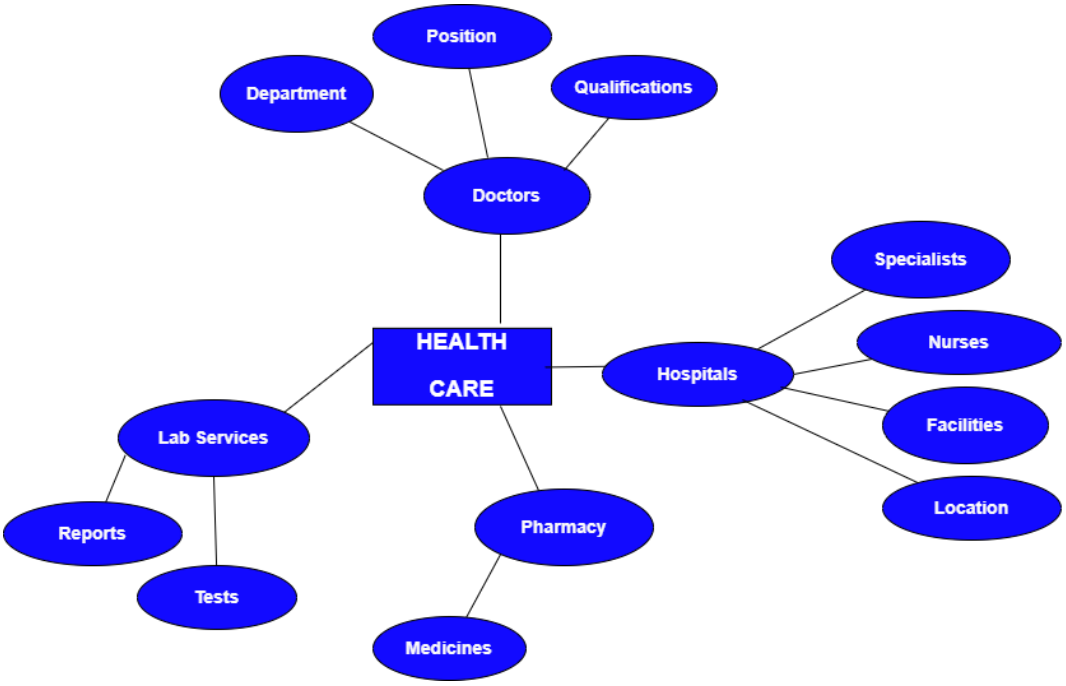


Fig 2.2. ER Diagram

➢ Healthcare Services

Diagram depicting the interaction flow for healthcare services, including booking appointments, accessing medical records, and consulting with healthcare providers, tests, location etc.

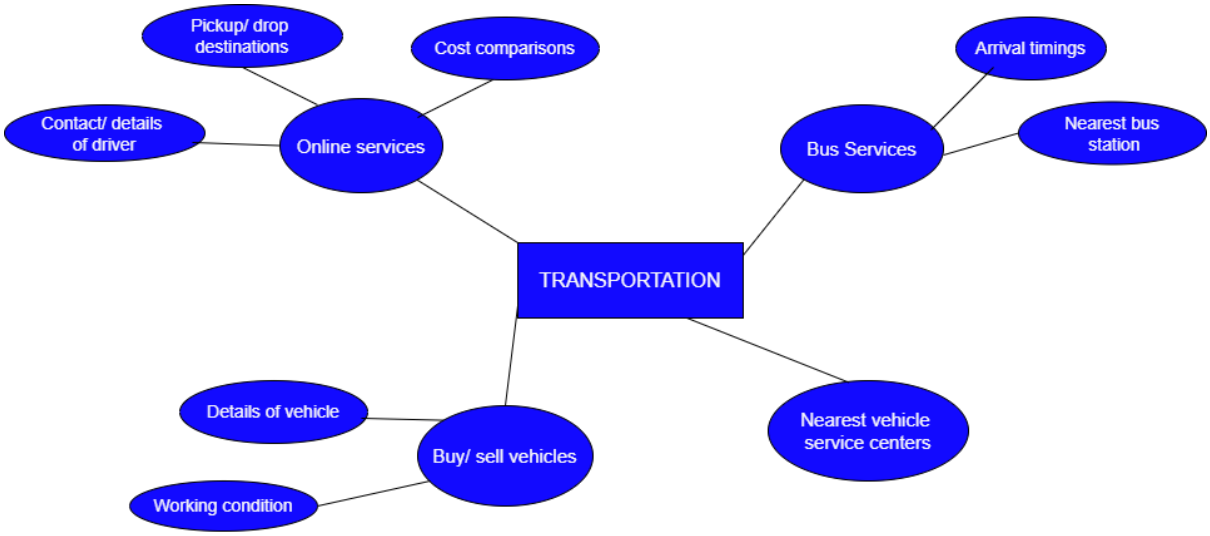


Fig 2.3. ER Diagram

➢ Transportation Services

Diagram depicting the interaction flow for transportation services, including Details of vehicle, service centre, bus service, nearest bus station location etc.

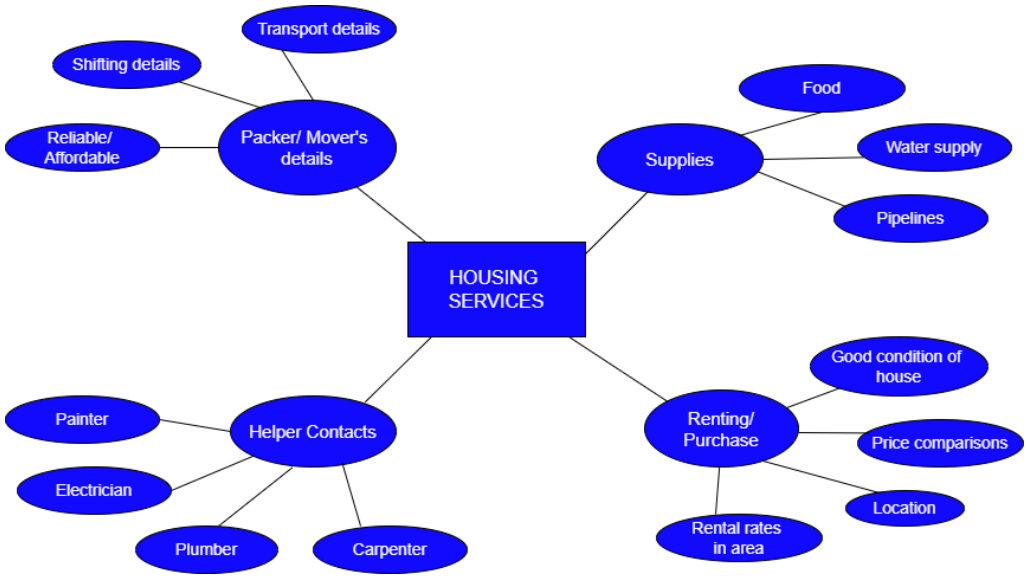


Fig 2.4. ER Diagram

- **Housing Services**

Diagram detailing how users interact with the housing services module, including searching for new home, rent home, plumber, carpenter, electrician, painter, packer, supplier etc.

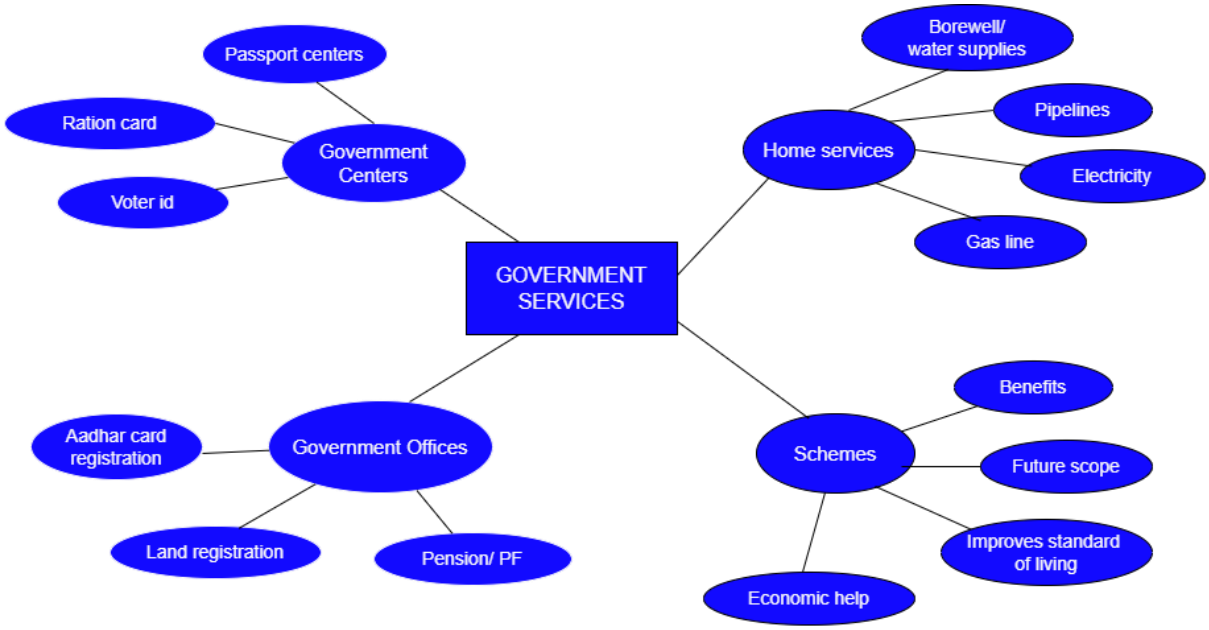


Fig 2.6. ER Diagram

- **Goverment Services**

Diagram detailing how users interact with the goverment services module, including services for land registration, pension, adhar card registration, economic help, electricity and gas line bill payment, pipeline etc.

12.1.2 Code Snippets

User Authentication Module or Login and Registration Module

User Login [XML Code]

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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:id="@+id/main"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:background="@color/white"
    android:paddingStart="20dp"
    android:paddingTop="10dp"
    android:paddingEnd="10dp"
    android:paddingBottom="10dp"
    tools:context=".loginActivity2"
    tools:layout_editor_absoluteX="-5dp"
    tools:layout_editor_absoluteY="2dp">


    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="84dp"
        android:text="@string/res2"
        android:textColor="@color/black"
        android:textColorHint="@color/black"
        android:textSize="21sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.471"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/textViewOD"
        android:layout_width="209dp"
        android:layout_height="57dp"
        android:text="LOGIN"
        android:textAlignment="center"
        android:textColor="#000099"
        android:textSize="40dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/editTextLTBAppAddress"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.47"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView5"
        app:layout_constraintVertical_bias="0.16" />

    <EditText
        android:id="@+id/editTextLTBAppAddress"
        android:layout_width="347dp"
        android:layout_height="62dp"
        android:layout_marginBottom="8dp"
        android:background="@drawable/border"
        android:drawableStart="@drawable/baseline_person_24"
        android:drawablePadding="10dp"
        android:ems="10"
        android:hint="@string/u1"
        android:importantForAutofill="no"
        android:inputType="text"
        android:paddingStart="20dp"
        android:paddingTop="10dp"
        android:paddingEnd="10dp"
        android:paddingBottom="10dp"
        android:textColor="@color/black"
        android:textColorHint="@color/black"
        android:textSize="22dp"
        app:layout_constraintBottom_toTopOf="@+id/editTextLTBAppContact"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.352"
        app:layout_constraintStart_toStartOf="parent" />

    <EditText
        android:id="@+id/editTextLTBAppContact"
```

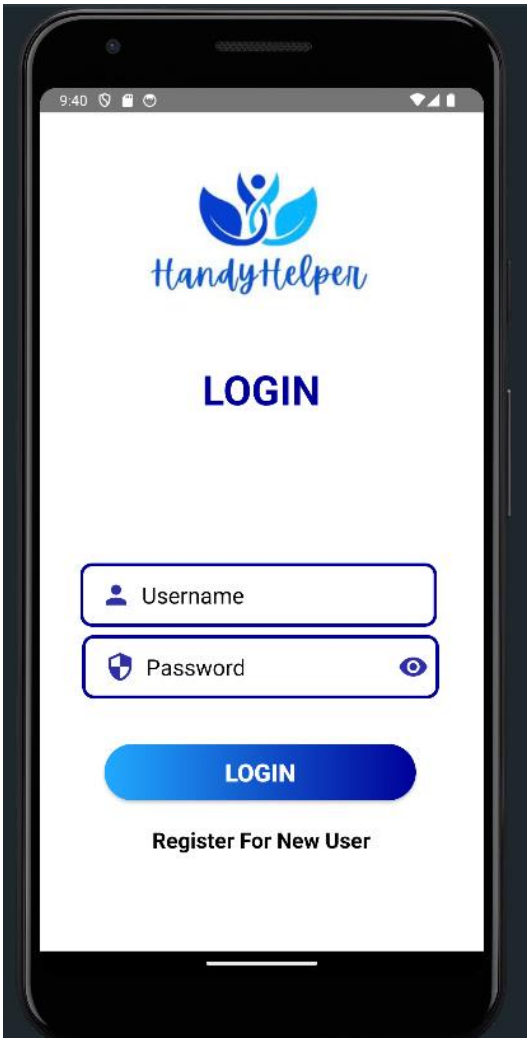
```
        android:layout_width="347dp"
        android:layout_height="62dp"
        android:layout_marginBottom="44dp"
        android:background="@drawable/border"
        android:drawableStart="@drawable/baseline_security_24"
        android:drawableEnd="@drawable/baseline_visibility_24"
        android:drawablePadding="10dp"
        android:ems="10"
        android:hint="@string/r2"
        android:importantForAutofill="no"
        android:inputType="textPassword"
        android:longClickable="false"
        android:paddingStart="22dp"
        android:paddingTop="10dp"
        android:paddingEnd="10dp"
        android:paddingBottom="10dp"
        android:textColorHint="@color/black"
        android:textSize="22dp"
        app:layout_constraintBottom_toTopOf="@+id/buttonBookApp2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.388"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/buttonBookApp2"
        android:layout_width="303dp"
        android:layout_height="55dp"
        android:layout_marginBottom="24dp"
        android:background="@drawable/button"
        android:text="@string/r1"
        android:textColor="@color/white"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.435"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="DuplicateSpeakableTextCheck" />

    <ImageView
        android:id="@+id/imageView5"
        android:layout_width="315dp"
        android:layout_height="182dp"
        android:layout_marginTop="24dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.424"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/back" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

User Login [Java Code]

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.view.MotionEvent;
import android.view.View;
```

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


public class loginActivity2 extends AppCompatActivity {
    EditText edUsername, edPassword;
    boolean passwordvisible;
    Button btn;
    TextView tv,tv1;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login2);

        edUsername = findViewById(R.id.editTextLTBAppAddress);
        edPassword = findViewById(R.id.editTextLTBAppContact);
        btn = findViewById(R.id.buttonBookApp2);
        tv = findViewById(R.id.textView3);

        edPassword.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent Event) {
                final int Right=2;
                if(Event.getAction()==MotionEvent.ACTION_UP){
                    if(Event.getRawX()>=edPassword.getRight()-
edPassword.getCompoundDrawables()[Right].getBounds().width()){
                        int selection=edPassword.getSelectionEnd();
                        if (passwordvisible) {


edPassword.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.baseline_
visibility_off_24,0);

edPassword.setTransformationMethod(PasswordTransformationMethod.getInstance());
                            passwordvisible=false;
                        }
                        else{

edPassword.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.baseline_
visibility_24,0);

edPassword.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
                            passwordvisible=true ;
                        }
                        edPassword.setSelection(selection);
                        return true;
                    }
                }
return false;
            }
        });
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String username = edUsername.getText().toString();
                String password = edPassword.getText().toString();
                Database db = new Database(getApplicationContext(), "healthcare",
null, 1);
                if (username.length() == 0 || password.length() == 0) {
                    Toast.makeText(getApplicationContext(), "Please fill all
details", Toast.LENGTH_SHORT).show();
                } else {
                    if (db.login(username, password) == 1) {
                        Toast.makeText(getApplicationContext(), "Welcome" +
```

```
username + "!", Toast.LENGTH_SHORT).show();
                        SharedPreferences SharedPreferences =
getSharedPreferences("shared_prefs", Context.MODE_PRIVATE);
                        SharedPreferences.Editor editor = SharedPreferences.edit();
                        editor.putString("username", username);
                        editor.apply();
                        startActivity(new Intent(loginActivity2.this
                                , homeActicity1.class));
                    } else {
                        Toast.makeText(getApplicationContext(), "Invalid username
and password", Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });
        tv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(loginActivity2.this
                        ,Registeractivity.class));
            }
        });


    }
}
```



User Registreation [XML Code]

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    tools:context=".Registeractivity">


    <TextView
        android:id="@+id/textViewOD"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="480dp"
        android:text="REGISTRATION"
        android:textColor="#2F2FA7"
        android:textSize="40sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <EditText
        android:id="@+id/editTextLTBPinCode"
        android:layout_width="347dp"
        android:layout_height="50dp"
        android:layout_marginTop="100dp"
        android:background="@drawable/border"
        android:drawableStart="@drawable/baseline_person_24"
        android:drawablePadding="10dp"
        android:ems="10"
        android:hint="Username"
        android:importantForAutofill="no"
        android:inputType="text"
        android:paddingStart="20dp"
        android:paddingTop="10dp"
        android:paddingEnd="10dp"
        android:paddingBottom="10dp"
        android:textColorHint="#141212"
        android:textSize="22sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textViewOD" />

    <EditText
        android:id="@+id/editTextLTBAppAddress"
        android:layout_width="347dp"
        android:layout_height="50dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/border"
        android:drawableStart="@drawable/baseline_email_24"
        android:drawablePadding="10dp"
        android:ems="10"
        android:hint="Email"
        android:importantForAutofill="no"
```

```
        android:inputType="textEmailAddress"
        android:paddingStart="20dp"
        android:paddingTop="10dp"
        android:paddingEnd="10dp"
        android:paddingBottom="10dp"
        android:textColorHint="#141212"
        android:textSize="22sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextLTBPinCode" />

    <EditText
        android:id="@+id/editTextLTBAppContact"
        android:layout_width="347dp"
        android:layout_height="50dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/border"
        android:drawableStart="@drawable/baseline_security_24"
        android:drawableEnd="@drawable/baseline_visibility_24"
        android:drawablePadding="10dp"
        android:ems="10"
        android:hint="@string/r2"
        android:importantForAutofill="no"
        android:inputType="textPassword"
        android:paddingStart="20dp"
        android:paddingTop="10dp"
        android:paddingEnd="10dp"
        android:paddingBottom="10dp"
        android:textColorHint="#141212"
        android:textSize="22sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextLTBAppAddress" />

    <EditText
        android:id="@+id/editTextAppPassword2"
        android:layout_width="347dp"
        android:layout_height="50dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/border"
        android:drawableStart="@drawable/baseline_security_24"
        android:drawableEnd="@drawable/baseline_visibility_24"

        android:drawablePadding="10dp"
        android:ems="10"
        android:hint="Confirm Password"
        android:importantForAutofill="no"
        android:inputType="textPassword"
        android:paddingStart="20dp"
        android:paddingTop="10dp"
        android:paddingEnd="10dp"
        android:paddingBottom="10dp"
        android:textColorHint="@color/black"
        android:textSize="22sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextLTBAppContact" />

    <Button
        android:id="@+id/buttonBookApp2"
        android:layout_width="303dp"
        android:layout_height="55dp"
        android:layout_marginTop="36dp"
        android:background="@drawable/button"
        android:text="Register"
        android:textColor="@color/white"
        android:textSize="22sp"
        android:textStyle="bold"
```

```
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextAppPassword2"
        app:layout_constraintVertical_bias="0.0"
        tools:ignore="DuplicateSpeakableTextCheck" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:text="Already have an account?"
        android:textColor="@color/black"
        android:textSize="21sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent" />

    <ImageView
        android:id="@+id/imageView5"
        android:layout_width="315dp"
        android:layout_height="182dp"
        android:layout_marginTop="4dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/back" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

User Registreation [Java Code]

```
package com.example.myapplication;

import android.content.Intent;
import android.os.Bundle;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.view.MotionEvent;
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;



public class Registeractivity extends AppCompatActivity {

    EditText edUsername, edEmail, edPassword, edConfirm;
    Button btn;
    TextView tv;
    boolean passwordvisible;


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

        edUsername = findViewById(R.id.editTextLTBPinCode);
        edEmail = findViewById(R.id.editTextLTBAppAddress);
```

```
        edPassword = findViewById(R.id.editTextLTBAppContact);
        edConfirm = findViewById(R.id.editTextAppPassword2);
        btn = findViewById(R.id.buttonBookApp2);
        tv = findViewById(R.id.textView3);

        tv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(Registeractivity.this
                        , loginActivity2.class));
            }
        });


        edPassword.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent Event) {
                final int Right=2;
                if(Event.getAction()==MotionEvent.ACTION_UP){
                    if(Event.getRawX()>=edPassword.getRight()-
edPassword.getCompoundDrawables()[Right].getBounds().width()){
                        int selection=edPassword.getSelectionEnd();
                        if (passwordvisible) {


edPassword.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.baseline_
visibility_off_24,0);

edPassword.setTransformationMethod(PasswordTransformationMethod.getInstance());
                            passwordvisible=false;
                        }
                        else{

edPassword.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.baseline_
visibility_24,0);

edPassword.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
                            passwordvisible=true ;
                        }
                        edPassword.setSelection(selection);
                        return true;
                    }
                }
                return false;
            }
        });


        edConfirm.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent Event) {
                final int Right=2;
                if(Event.getAction()==MotionEvent.ACTION_UP){
                    if(Event.getRawX()>=edConfirm.getRight()-
edConfirm.getCompoundDrawables()[Right].getBounds().width()){
                        int selection=edConfirm.getSelectionEnd();
                        if (passwordvisible) {


edConfirm.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.baseline_v
isibility_off_24,0);

edConfirm.setTransformationMethod(PasswordTransformationMethod.getInstance());
                            passwordvisible=false;
                        }
                        else{
```

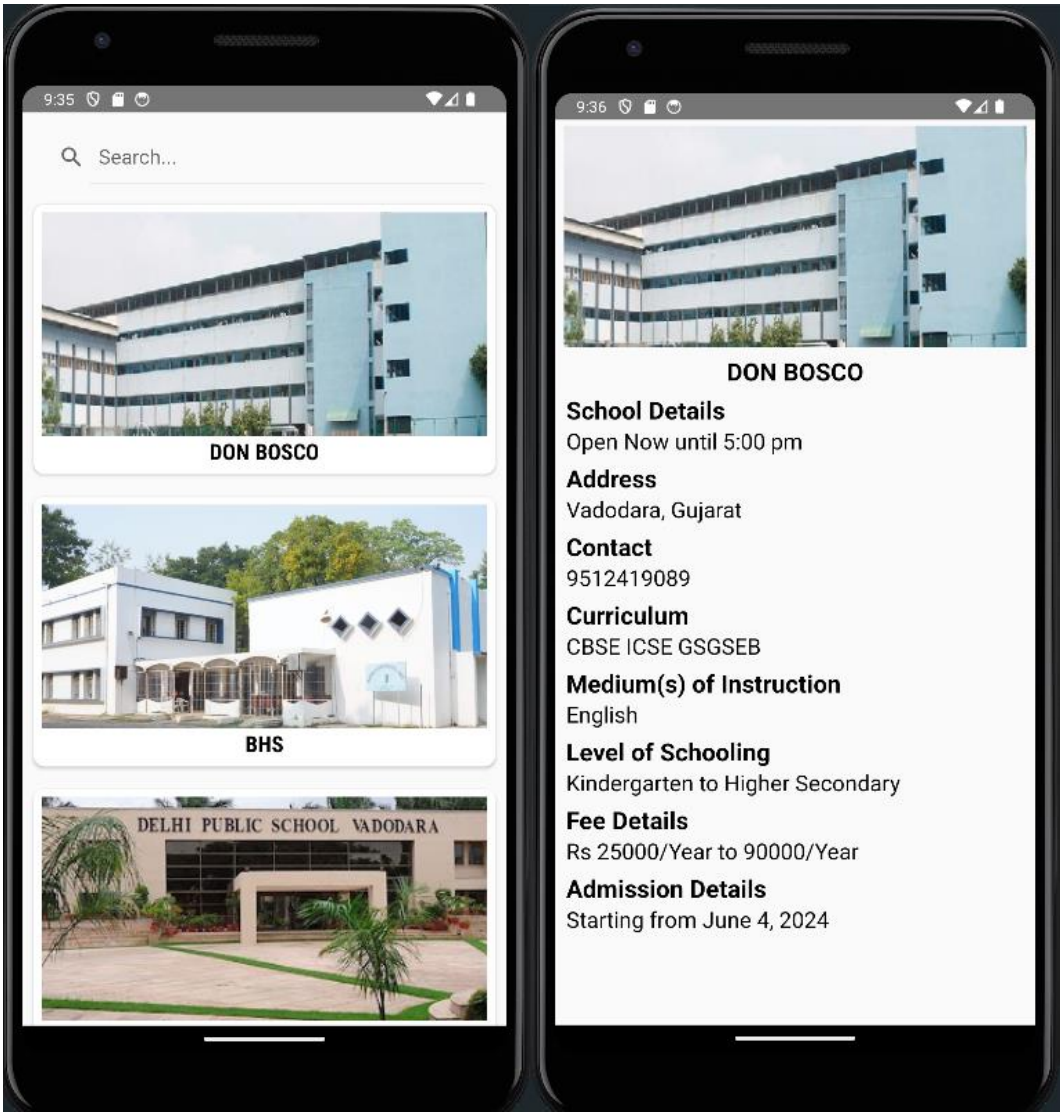
```
edConfirm.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.baseline_v
isibility_24,0);

edConfirm.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
                            passwordvisible=true ;
                        }
                        edConfirm.setSelection(selection);
                        return true;
                    }
                }
                return false;
            }
        });

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = edUsername.getText().toString();
                String Email = edEmail.getText().toString();
                String password = edPassword.getText().toString();
                String confirm = edConfirm.getText().toString();
                Database db = new Database(getApplicationContext(), "healthcare",
null, 1);


                if (username.length() == 0 || password.length() == 0) {
                    Toast.makeText(getApplicationContext(), "Please fill all
details", Toast.LENGTH_SHORT).show();
                } else {
                    if (password.compareTo(confirm) == 0) {
                        if (isValid(password)) {
                            db.register(username, Email, password);
                            Toast.makeText(getApplicationContext(), "Record
Inserted", Toast.LENGTH_SHORT).show();
                            startActivity(new Intent(Registeractivity.this
                                    , loginActivity2.class));
                        } else {
                            Toast.makeText(getApplicationContext(), "Password must
contain at least  8 character,having letter,digit and special symbols",
Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        Toast.makeText(getApplicationContext(), "Password and
confirm Password did't match", Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });
    }

    public static boolean isValid(String password) {


            int f1 = 0, f2 = 0, f3 = 0;
            if (password.length() < 8) {
                return false;
            } else {
                for (int p = 0; p < password.length(); p++) {
                    if (Character.isLetter(password.charAt((p)))) {
                        f1 = 1;
                    }
                }
                for (int r = 0; r < password.length(); r++) {
                    if (Character.isDigit(password.charAt((r)))) {
                        f2 = 1;
                    }
                }
                for (int s = 0; s < password.length(); s++) {
```

```
                    char c = password.charAt(s);
                    if (c >= 33 && c <= 46 || c == 64) {
                        f3 = 1;
                    }
                }
                if (f1 == 1 && f2 == 1 && f3 == 1)
                    return true;
                return false;

            }


    }
}
```

Education Services Module [Java Code]

```
package com.example.myapplication;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.myapplication.Adapter.ItemAdapter;
import com.example.myapplication.Model.ItemList;

public class Education extends AppCompatActivity {

    RecyclerView recyclerView;

    ItemList[] mydata = new ItemList[]{
            new ItemList("DON BOSCO",R.drawable.don_bosco),
            new ItemList("BHS",R.drawable.bhs),
            new ItemList("DPS",R.drawable.dps),
            new ItemList("TREE HOUSE",R.drawable.tree_house),
            new ItemList("NALANDA",R.drawable.nalanda),
            new ItemList("ZENITH",R.drawable.zenith),
            new ItemList("BHAVANS",R.drawable.bhavans),
            new ItemList("AMICUS",R.drawable.amicus),

    };

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

        recyclerView = findViewById(R.id.recyclerView);
        ItemAdapter adapter = new ItemAdapter(mydata);
        recyclerView.setHasFixedSize(true);



   recyclerView.setLayoutManager(new LinearLayoutManager(this));

        // For vertical and horizontal views
        LinearLayoutManager linear = new
LinearLayoutManager(getApplicationContext());
        linear.setOrientation(LinearLayoutManager.VERTICAL);
        recyclerView.setLayoutManager(linear);

        //Grid layout view
//        GridLayoutManager gridLayoutManager = new
GridLayoutManager(getApplicationContext(),2);
//        gridLayoutManager.setOrientation(LinearLayoutManager.VERTICAL);
//        recyclerView.setLayoutManager(gridLayoutManager);

        //Staggered Grid Layout
//        StaggeredGridLayoutManager staggeredGridLayoutManager = new
StaggeredGridLayoutManager(2,LinearLayoutManager.VERTICAL);
```

```
//         recyclerView.setLayoutManager(staggeredGridLayoutManager);

            recyclerView.setAdapter(adapter);

        }
    }
```

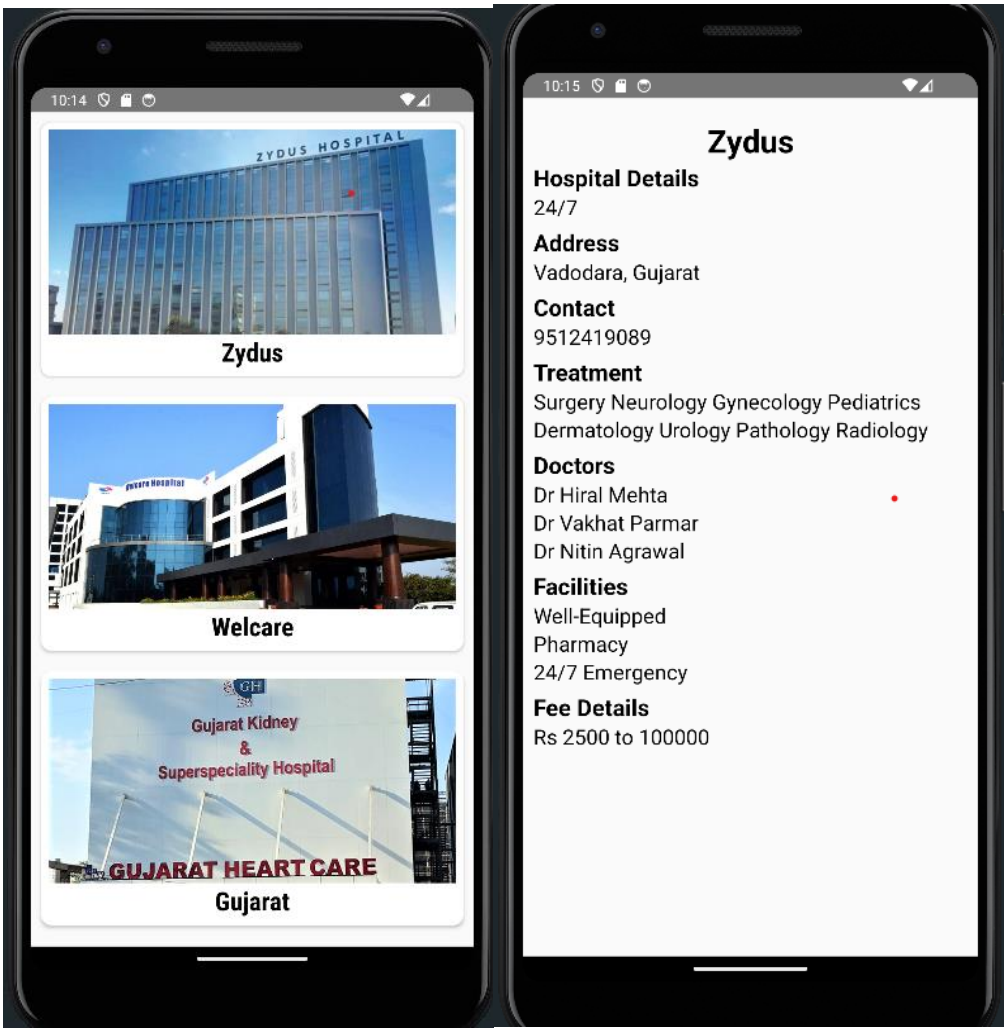


Healthcare Services Module [Java Code]

```
package com.example.myapplication;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

public class homeActivity2 extends AppCompatActivity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home2);

        SharedPreferences SharedPreferences=getSharedPreferences("shared_prefs",
Context.MODE_PRIVATE);
        String username=SharedPreferences.getString("username","").toString();


        CardView exit=findViewById(R.id.cardExitCard);
        exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                SharedPreferences.Editor editor=SharedPreferences.edit();
                editor.clear();
                editor.apply();
                Toast.makeText(getApplicationContext(), "BACK",
Toast.LENGTH_SHORT).show();
                startActivity(new Intent(homeActivity2.this
                        ,homeActicity1.class));
            }
        });
CardView findDoctor=findViewById(R.id.cardFindDoctor);
findDoctor.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(homeActivity2.this
                ,FindDoctorActivity2.class));
    }
});

CardView labTest= findViewById(R.id.cardLabTest);
labTest.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(homeActivity2.this,LabTestActivity.class));
    }
});

CardView orderDetails=findViewById(R.id.cardOrderDetails);
orderDetails.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new
Intent(homeActivity2.this,OrderDetailsMainActivity2.class));
    }
});


        CardView BuyMedicine=findViewById(R.id.cardBuyMedicine);
        BuyMedicine.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(homeActivity2.this,BuyMedicineActivity2.class));
            }
        });

CardView health=findViewById(R.id.cardHealthDoctor);
health.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(homeActivity2.this,HealthArticlesActicity.class));
    }
});
```

```
        CardView hospital=findViewById(R.id.Hospital);
        hospital.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(homeActivity2.this,Hospital_Acticity.class));
            }
        });


        CardView insurance=findViewById(R.id.Insurance);
        insurance.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(homeActivity2.this,
InsuranceActivity2.class));
            }
        });



    }
}
```



There are many more code screenshoot and output photos but it will be all available in the video.

12.2 Supporting Documentation

12.2.1 User Manuals

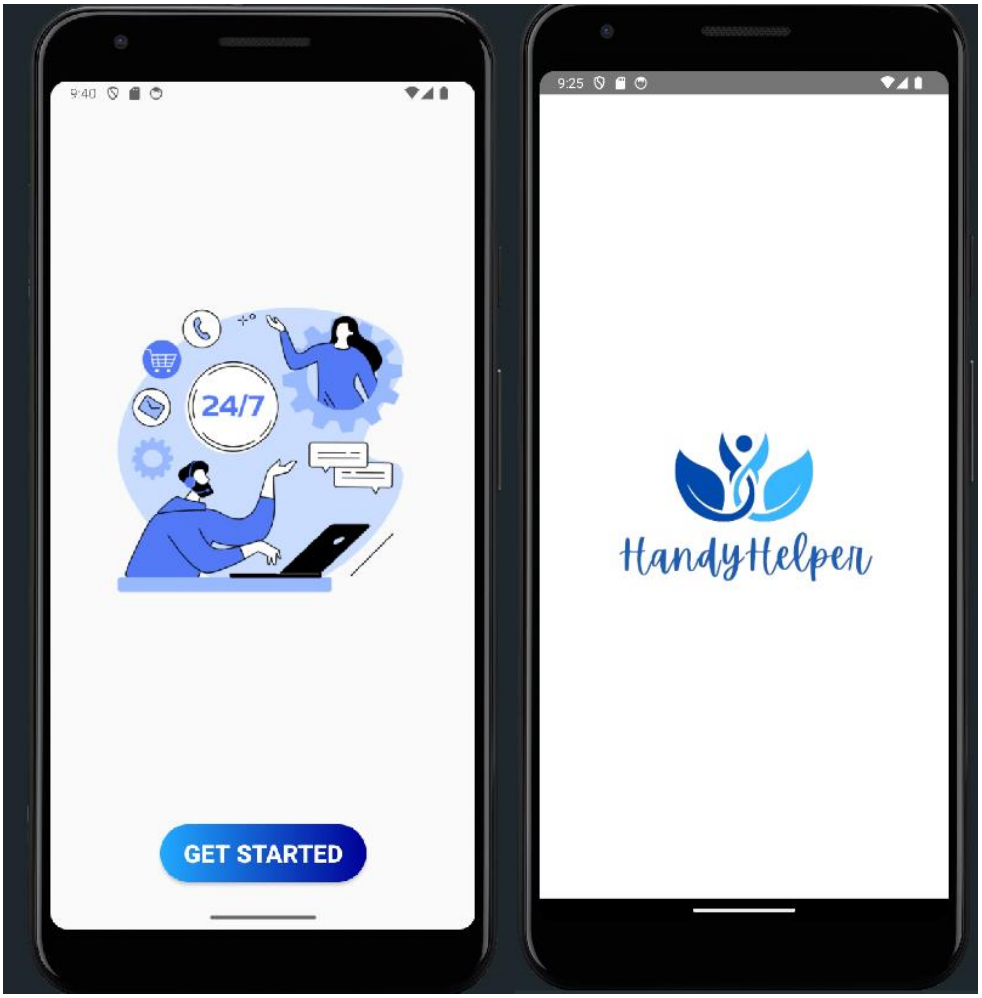
Getting Started

- **System Requirements**:

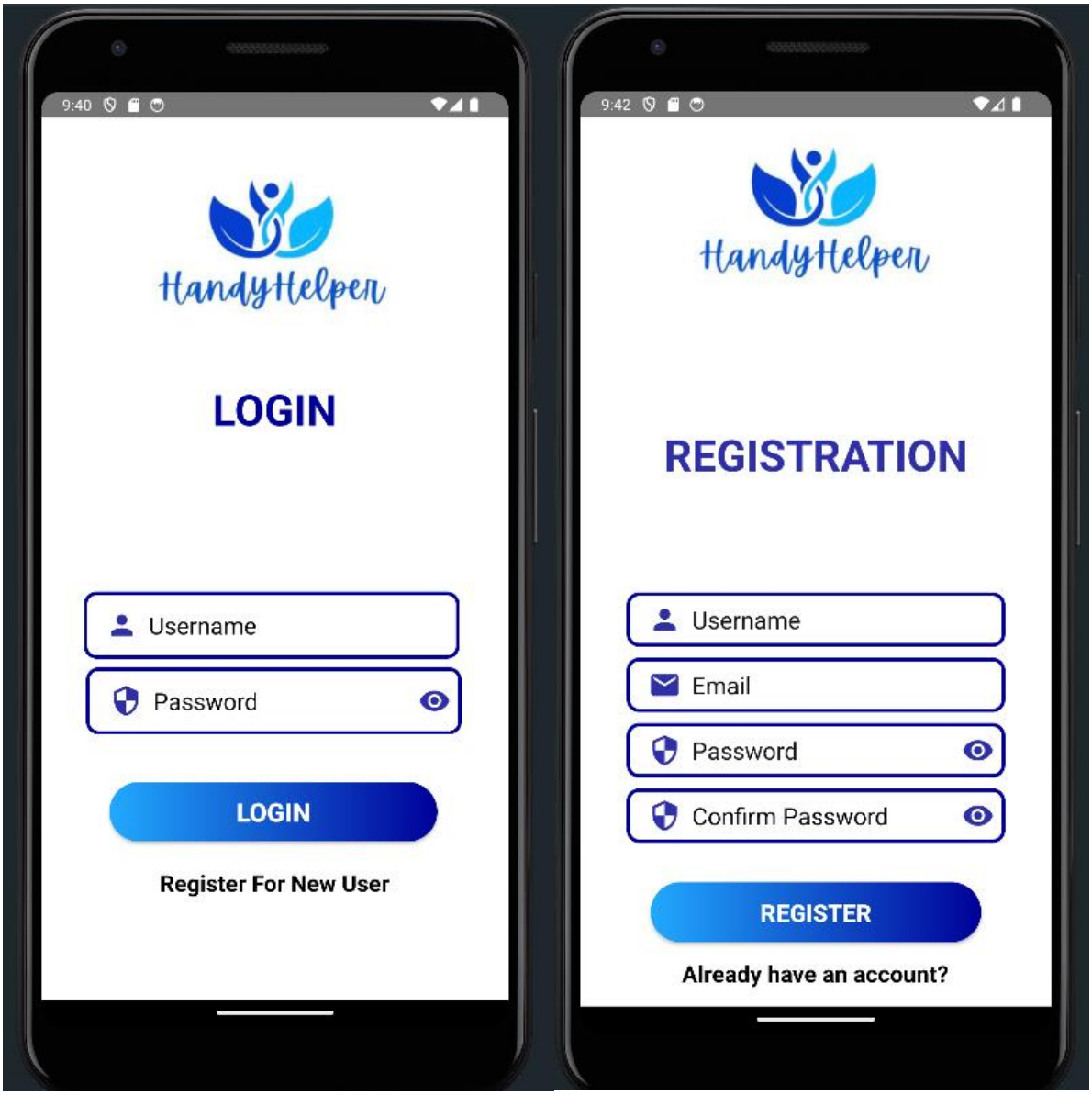
- Android version 5.0 (Lollipop) or higher to run the application in your AVD in android studio or you can run it in your android phone also via a USB or QR code.
- Internet connection is must.

- Creating an Account

1. **Open the App**:
    - Tap the app icon to open.
    - A splash screen with symbol will appear for some seconds and a page asking to get started will appear.
    - Click on Get started to start the app.



2. **Select "Register"**
    - After get started Login screen will appear and click on Register for new user.
    - Enter your details: username, email, password and confirm password.
    - Click on register and a Login page will appear.



3. **Select “Login”**

    - If you are already a user or you have completed with your registration process, you have login to use the app.
    - Enter your Username and Password you have created at the time of registration.
    - Tap on "Login."

3. Navigating the App

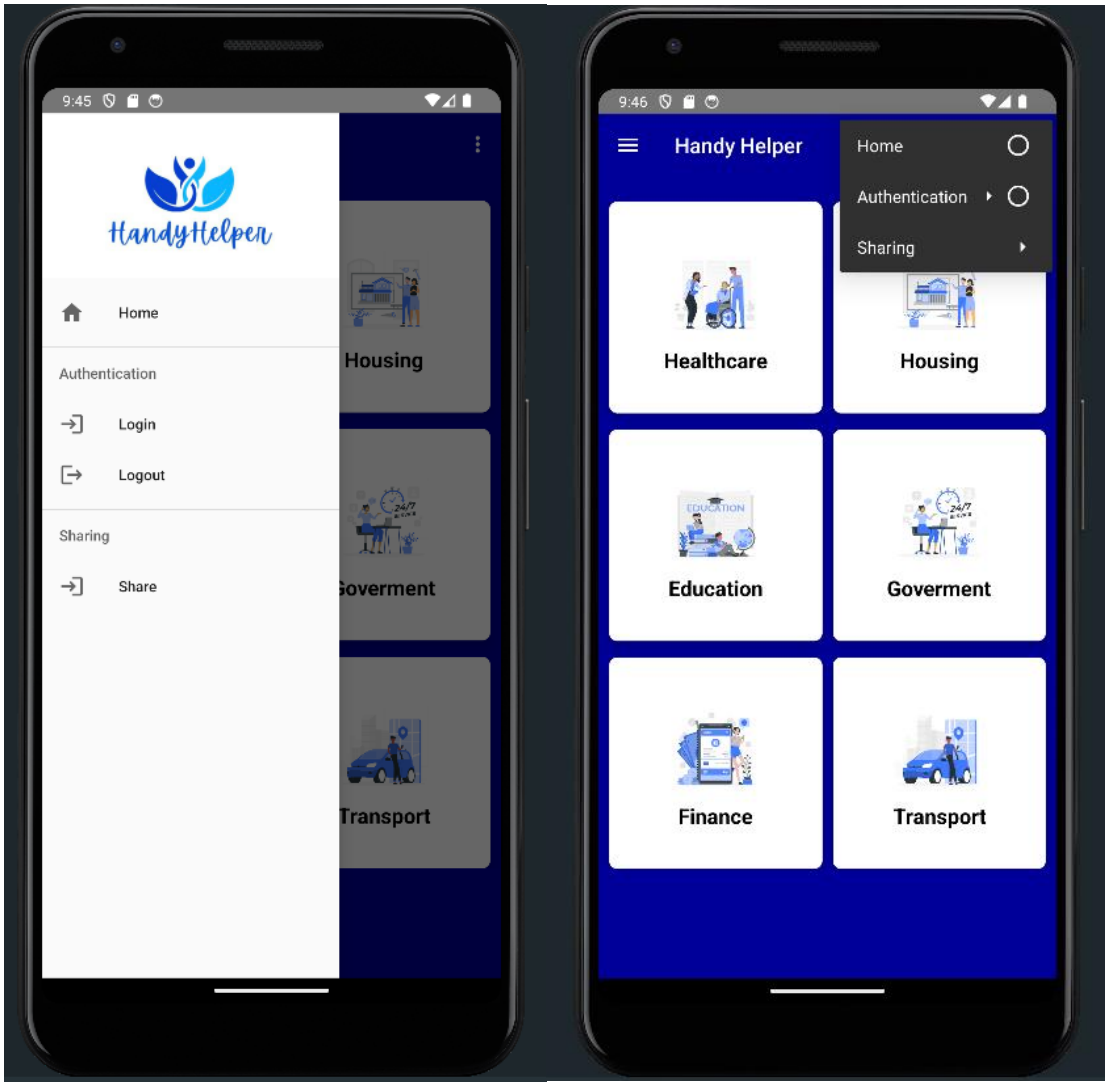
Main Menu Overview

- After you complete your login you will be redirected to Home screen or dashboard.
- **Dashboard**: Overview of all services.

Navigating the side services

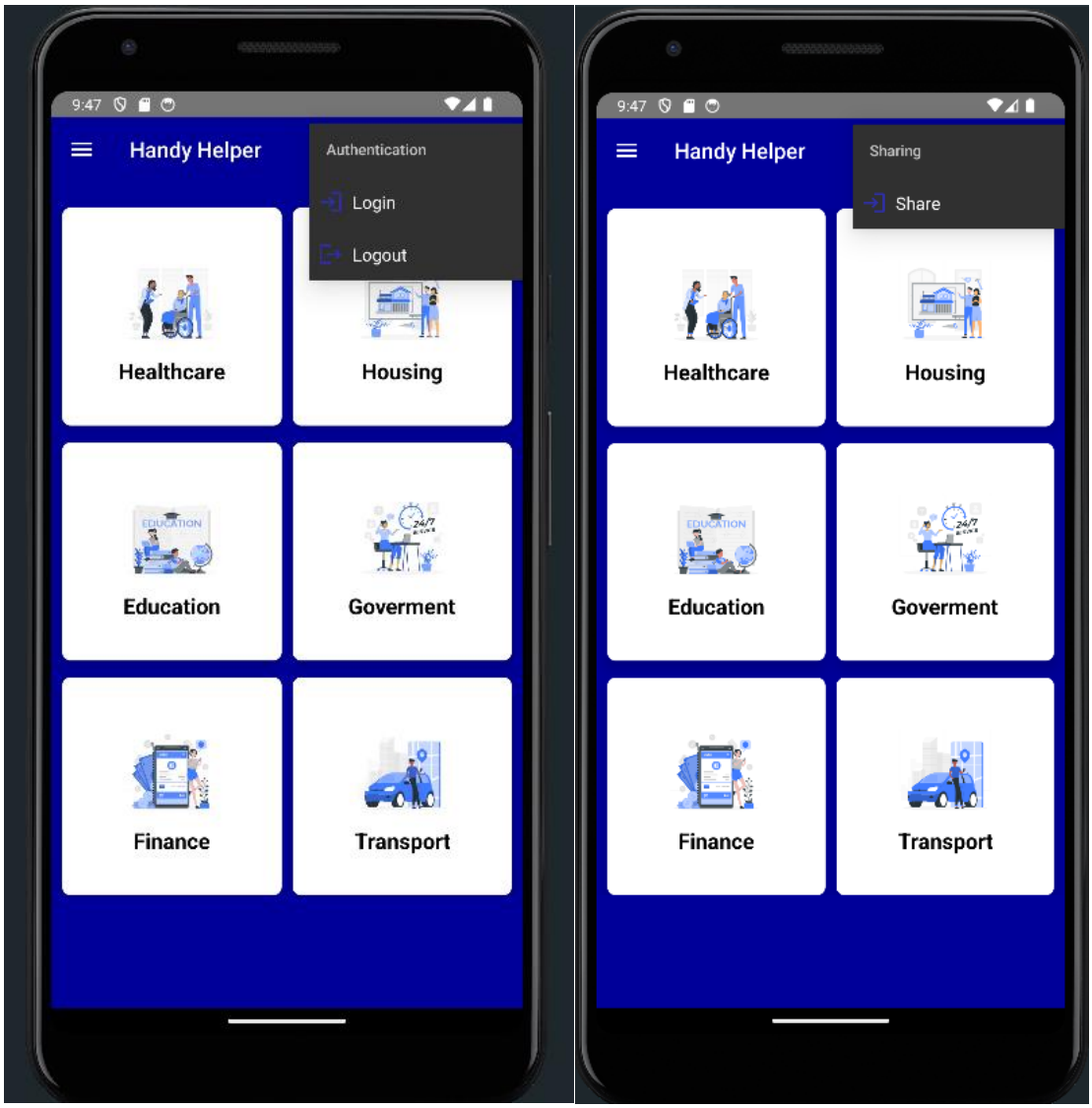
Left Navigation Drawer

- **Home**: This button has no command for now.
- **Login**: This button has no command for now
- **Logout**: This button has no command for now.
- **Share**: This button has no command for now.



Right Navigation Drawer

- **Home**: This button redirects to home.
- **Authentication**: This button has two command button for now.
- **Login**: This button command to login page.
- **Logout**: This button command user to logout.
- **Share**: This button commands to share the app via different platforms.

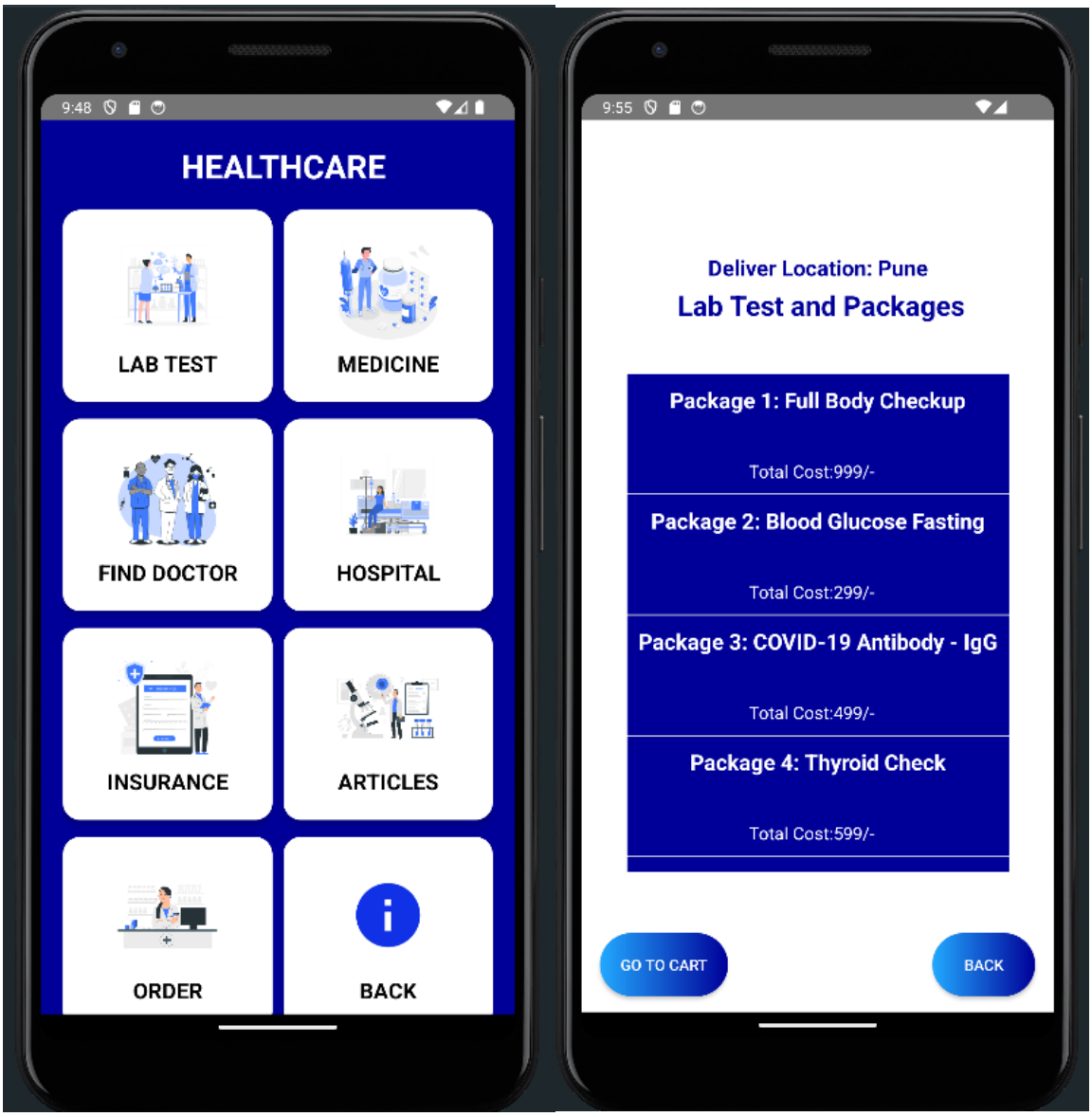


4. Accessing Services

Healthcare Services

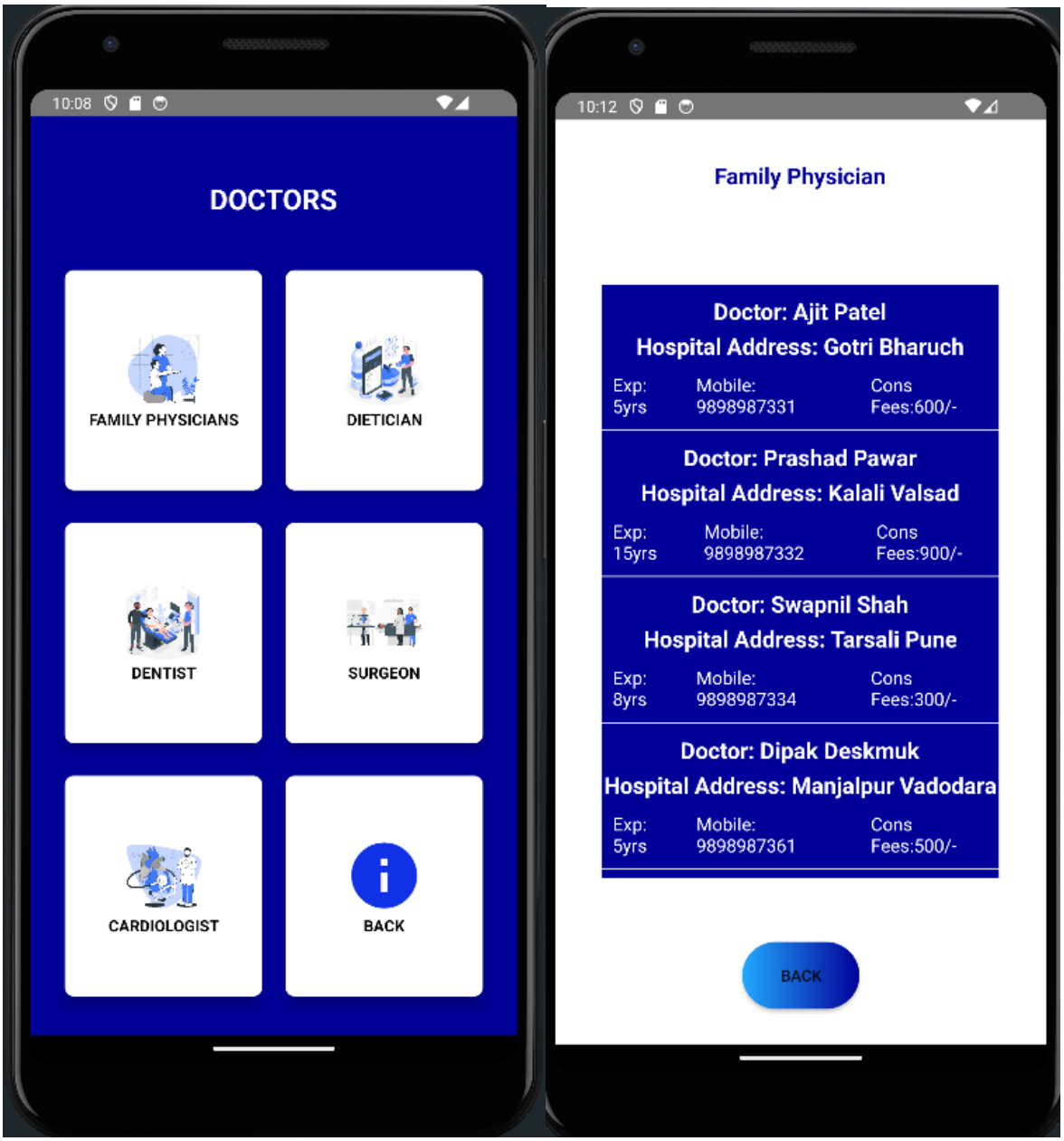
- After Clicking on Healthcare Service it will redirect you to the submodule services of healthcare and you can any service you want.

1. **Lab Test: Buy or Book lab test Package**
    - Go to the "Healthcare" section.
    - Select a healthcare provider or lab test.
    - Choose any package of your choice and add to cart and go to cart and select the date and time for delivery and payment and checkout, fill the required details and book.

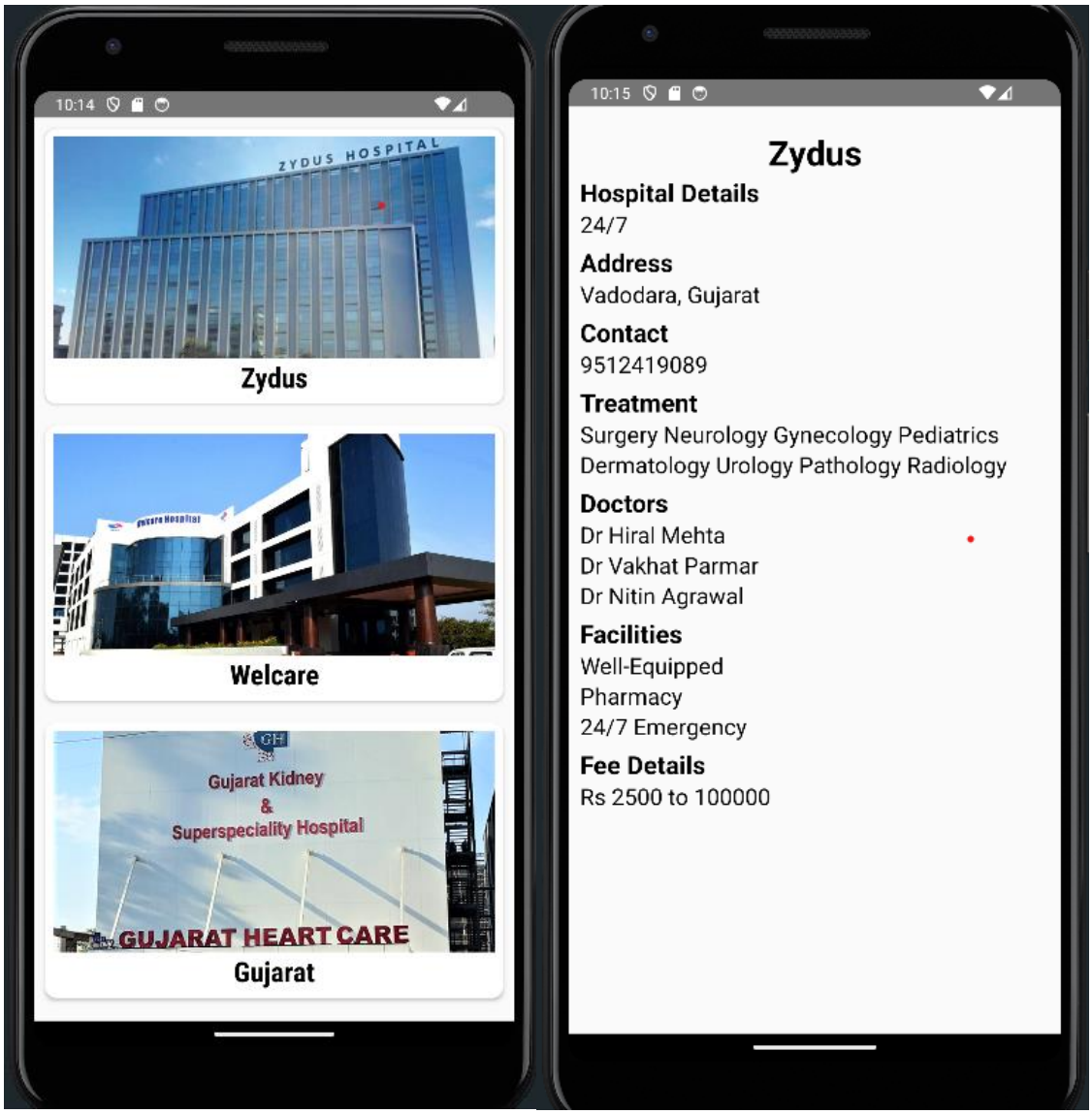


2. **Buy Medicine: Buy or Book Medicine**
    - Go to the "Healthcare" section.
    - Select a healthcare provider or Medicine.
    - Choose any package of your choice and add to cart and go to cart and select the date and time for delivery and payment and checkout, fill the required details and book.

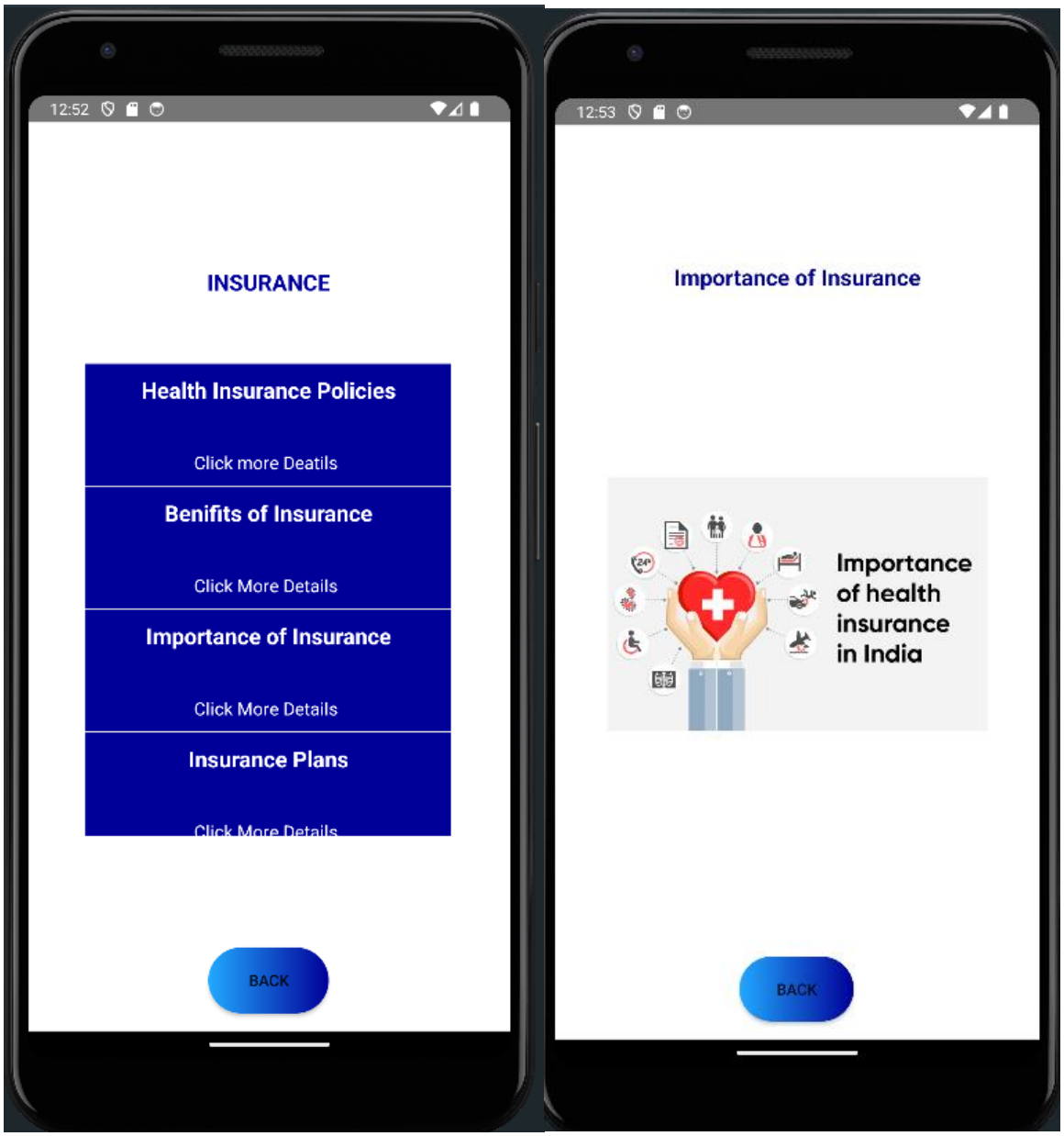
3. **Doctor: Book Doctor**
    - Go to the "Healthcare" section.
    - Select a healthcare provider or Find Doctor.
    - Choose any doctor of your choice and add to cart and go to cart and select the date and time for delivery and payment and checkout, fill the required details and book the appointment.



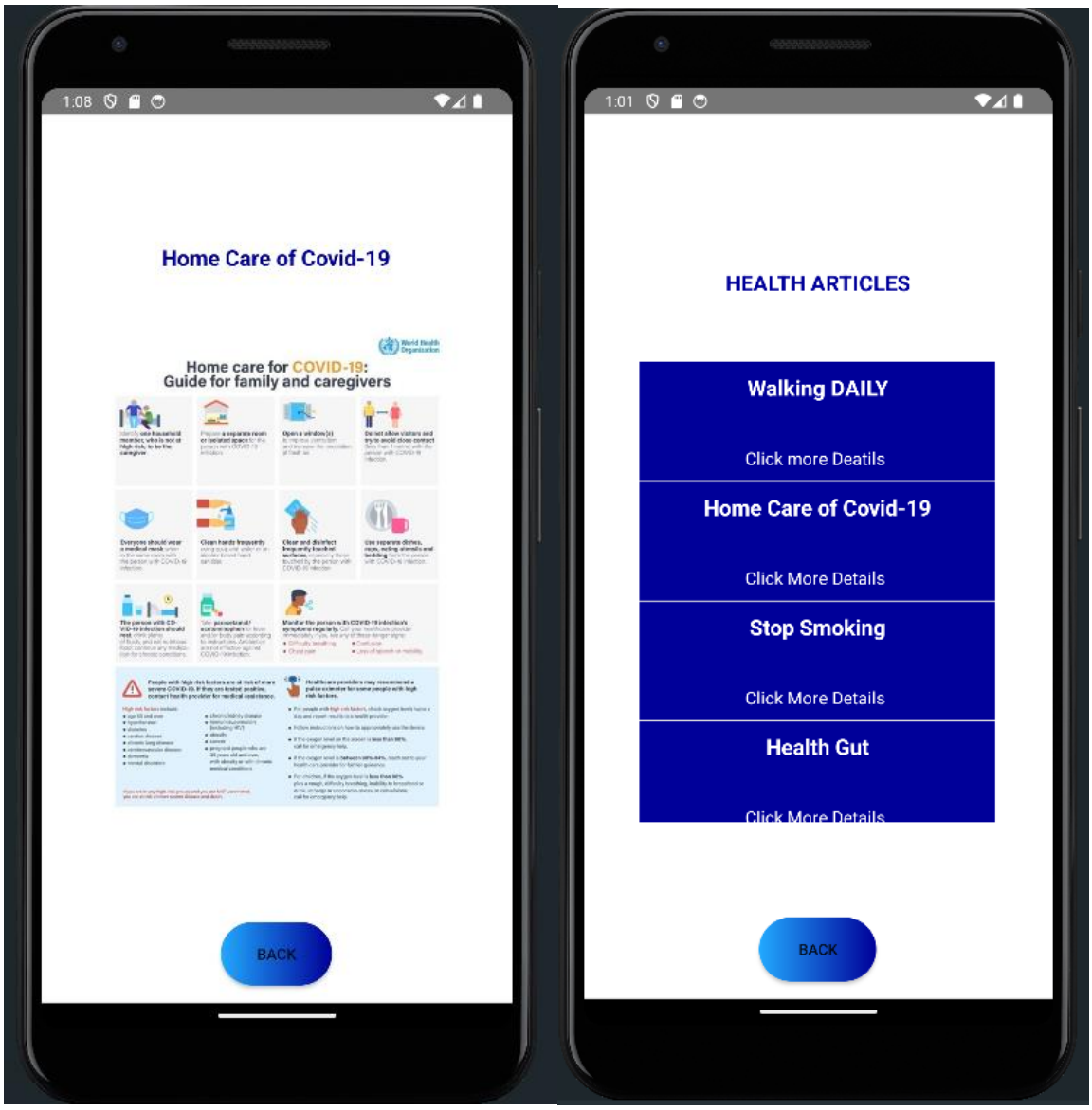
4. **Hospital: View Hospital details**
    - Go to the "Healthcare" section.
    - Select a healthcare provider or Hospital.
    - Choose any Hospital of your choice and open the hospital details and contact and other necessary details.



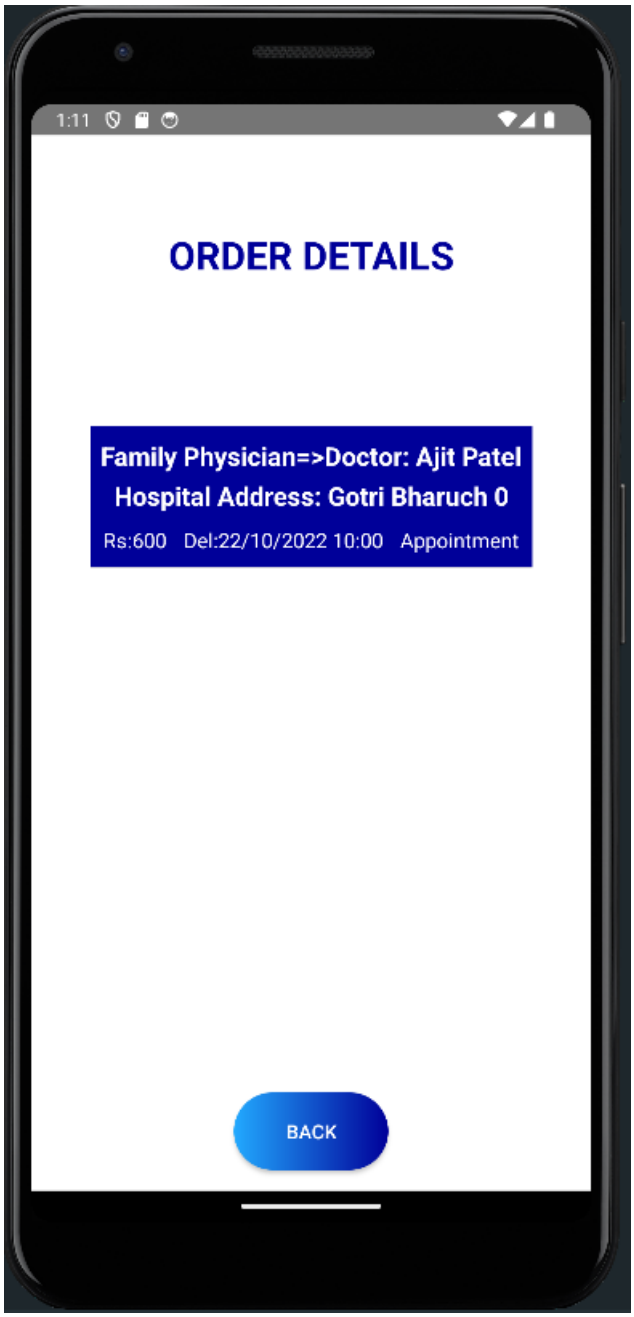
5. **Insurance: View insurance details**
    - Go to the "Healthcare" section.
    - Select a healthcare provider or Insurance.
    - Choose any Insurance package of your choice and open the Insurance details and other necessary details.



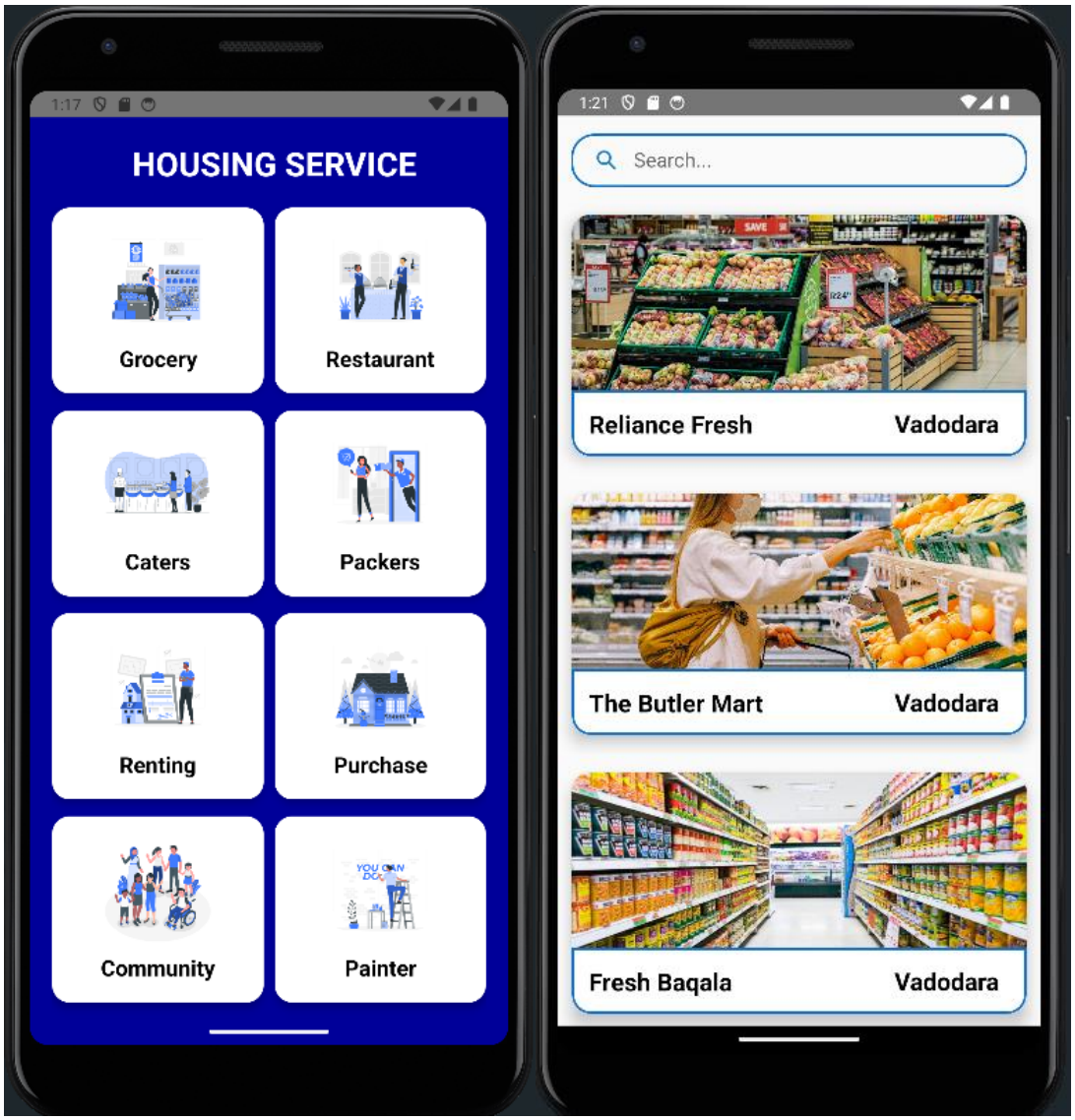
6. **Article: View Article details**
    - Go to the "Healthcare" section.
    - Select a healthcare provider or Article.
    - Choose any Article package of your choice and open the Article details and read other necessary details.

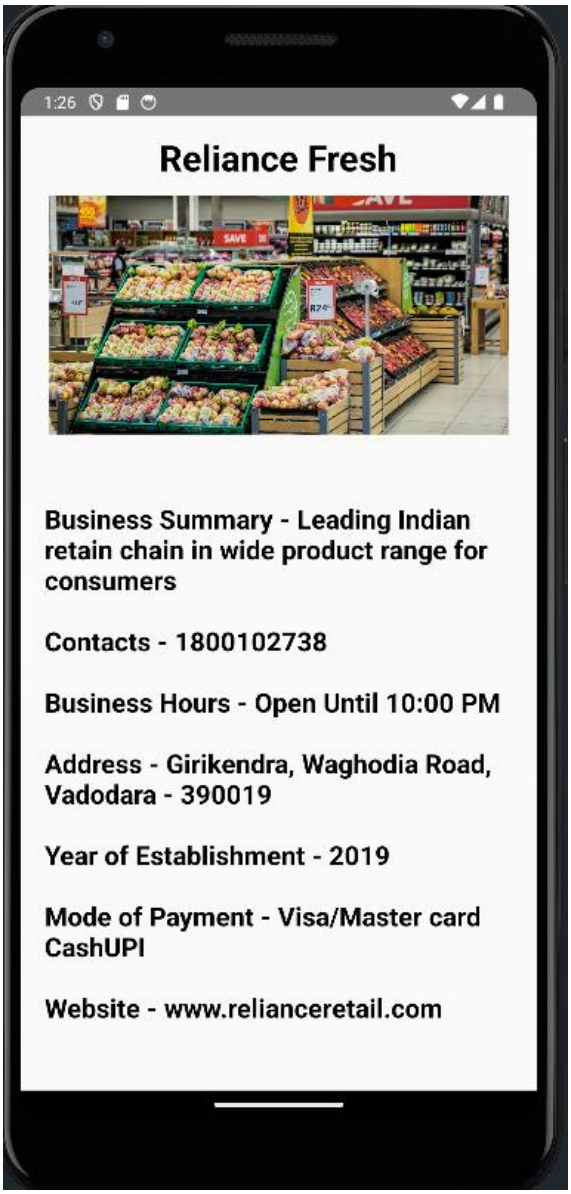


7. **Order: View Order details**
    - Go to the "Healthcare" section.
    - Select a healthcare provider or Order.
    - Any Order you have placed of your choice will be seen over here with its detail.

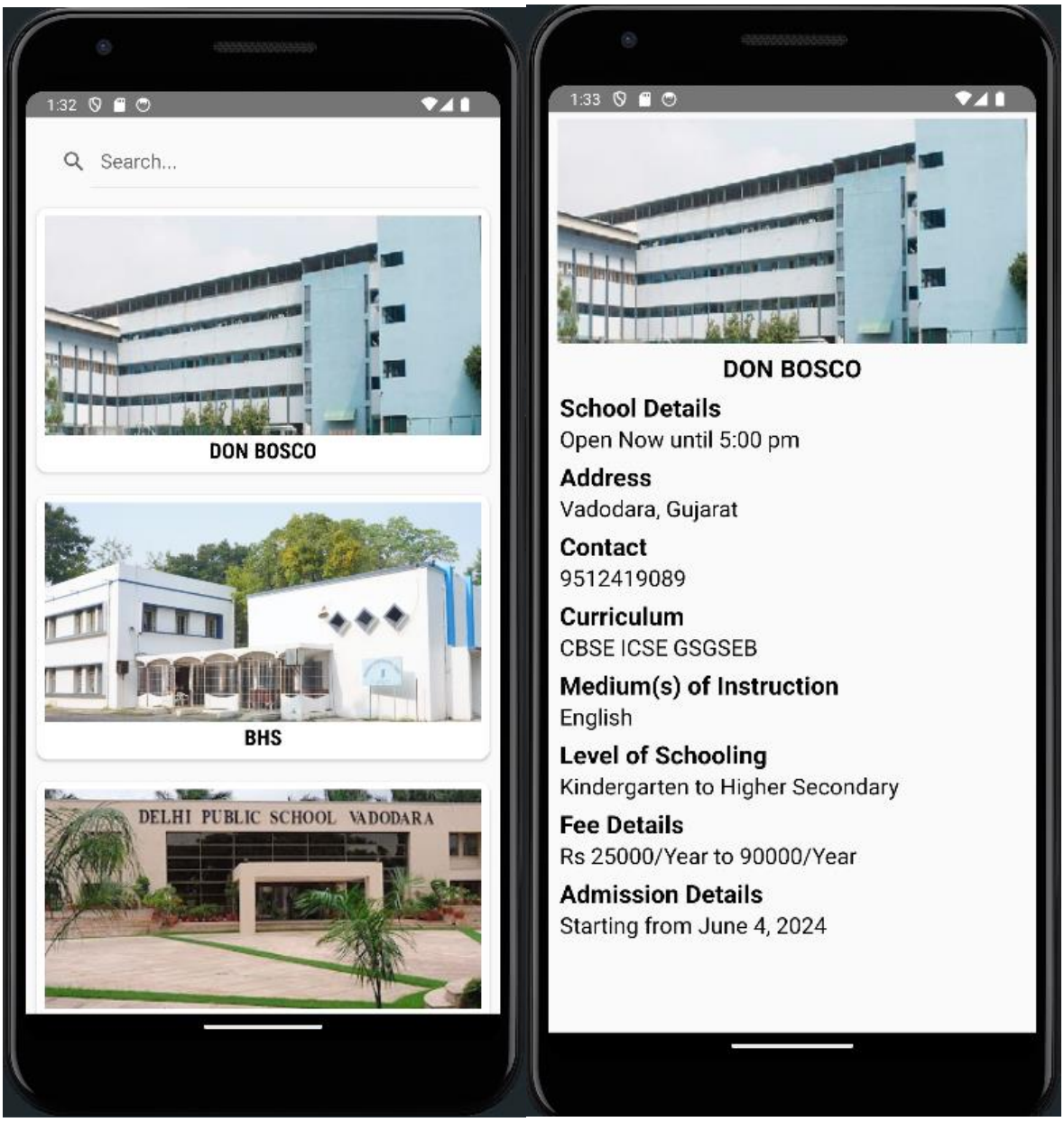


1. **Housing: View Housing details**
    - Go to the "Housing" section.
    - Select a Housing provider or Housing.
    - Choose any Housing service like grocery for now of your choice and open the different grocery details and contact and other necessary details.


1:17
HOUSING SERVICE
Grocery
Restaurant
Caters
Packers
Renting
Purchase
Community
Painter
1:21
Search...
Reliance Fresh
Vadodara
The Butler Mart
Vadodara
Fresh Baqala
Vadodara



1. **Education: View Education details**
    - Go to the "Education" section.
    - Select an Education provider or Education.
    - Choose any Education service your choice and open the different school details and contact and other necessary details.


1:32
Search...
DON BOSCO
BHS
DELHI PUBLIC SCHOOL VADODARA
1:33
DON BOSCO
School Details
Open Now until 5:00 pm
Address
Vadodara, Gujarat
Contact
9512419089
Curriculum
CBSE ICSE GSGSEB
Medium(s) of Instruction
English
Level of Schooling
Kindergarten to Higher Secondary
Fee Details
Rs 25000/Year to 90000/Year
Admission Details
Starting from June 4, 2024