



# **LINKUP – COMMUNITY SERVICE MOBILE APPLICATION**



## **A MINI PROJECT REPORT**

*Submitted by*

<b>KAMIL S</b>	<b>(622121104043)</b>
<b>KIRUBASANKAR K</b>	<b>(622121104048)</b>
<b>MITHIN R C</b>	<b>(622121104059)</b>

*in partial fulfillment for the award of the degree  
of*

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**PAAVAI ENGINEERING COLLEGE,  
(AUTONOMOUS)  
PACHAL, NAMAKKAL.**

**ANNA UNIVERSITY: CHENNAI 600 025  
DECEMBER 2024**

**PAAVAI ENGINEERING COLLEGE,  
(AUTONOMOUS)**

**BONAFIDE CERTIFICATE**

Certified that this project report “**LINKUP - COMMUNITY SERVICE MOBLIE  
APPLICATON**” is the bonafide work of “**KAMIL S (622121104043),  
KIRUBASANKAR K (622121104048) and MITHIN R C (622121104059)**” who  
have carried out the project work under my supervision.

**SIGNATURE**

**Dr. D. BANUMATHY, M.E., Ph.D.**

**HEAD OF THE DEPARTMENT**

Department of Computer Science and  
Engineering,  
Paavai Engineering College,  
Namakkal-637 018

**SIGNATURE**

**Mr. D.V. RAJKUMAR, M.E.,**

**SUPERVISIOR**

Department of Computer Science and  
Engineering,  
Paavai Engineering College,  
Namakkal-637 018.

Submitted for the university project Viva – Voce held on -----

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **DECLARATION**

We KAMIL S, KIRUBASANKAR K and MITHIN R C hereby declare that the Project Report titled “**LINKUP - COMMUNITY SERVICE MOBILE APPLICATION**” done by us under the guidance of **Mr. D.V.RAJKUMAR, M.E.**, at paavai engineering college, namakkal is submitted in partial fulfilment of the requirements for the award of bachelor of engineering degree in computer science and engineering. Certified further that, to the best of my knowledge, the work reported here in does not form part of any other project report or dissertation on the basic of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**1.**

**2.**

**DATE :**

**3.**

**PLACE: NAMAKKAL**

**SIGNATURE OF THE CANDIDATES**

## ACKNOWLEDGEMENT

We express our profound gratitude to our honourable Chairman, Shri. **C.A. N.V. NATARAJAN, B.com., F.C.A.**, and also to our Correspondent Smt. **N. MANGAI NATARAJAN, M.Sc.**, for providing all necessary facilities for the successful completion of this project.

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

We would like to thank our respected Principal **Dr. M. PREM KUMAR, M.E., Ph.D.**, for allowing us to do this project and providing require time to completion the same

We wish to express our sincere gratitude to **Dr. D. BANUMATHY, M.E., Ph.D.**, Head of Computer Science and Engineering Department for her extended encourage to fulfill this project.

We express our sincere thanks to **Mr. D.V. RAJKUMAR, M.E.**, Supervisor for the useful suggestions, which helped us for completing the project work in time.

We would like to extend our sincere thanks to **Dr. A. JEYAMURUGAN, M.E., Ph.D.**, Project Coordinator for giving this opportunity to do this project and also for his inspiring guidance, generous help and support.

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

## **ABSTRACT**

LinkUp is a mobile application designed to bridge the gap between individuals seeking household and labour services and skilled service providers within their community. The app enables users to book a variety of services, including but not limited to cleaning, plumbing, electrical work, and general labour. Through LinkUp, users can easily connect with experienced service professionals, ensuring both convenience and trust. The platform offers a seamless user experience, allowing individuals to browse available services, check service provider profiles, and book appointments directly from their smartphones. Service providers, on the other hand, can create detailed profiles showcasing their skills, previous work, and customer reviews. This two-way interaction ensures that both users and service providers have the opportunity to assess suitability before a service is booked. LinkUp also includes features like real-time messaging, ensuring effective and transparent communication between the user and service provider. By fostering a reliable and efficient marketplace for services, LinkUp empowers users to quickly address their needs while supporting local businesses. Whether it's a household task or larger labour requirements, LinkUp is the go-to solution for all community service bookings, simplifying the process, enhancing trust, and promoting community growth through accessible, reliable service connections.

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NUMBER
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF FIGURES</b>	<b>viii</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>ix</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Introduction	1
	1.2 Overview	1
	1.3 Objective	2
	1.4 Problem Statement	2
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>3</b>
	2.1 Literature Review	3
	2.2 Summary of Issues	7
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>8</b>
	3.1 Existing System	8
	3.1.1 Disadvantages	8
	3.2 Proposed System	9
	3.2.1 Advantages	9
<b>4</b>	<b>SYSTEM SPECIFICATION</b>	<b>10</b>
	4.1 Hardware Requirements	10
	4.2 Software Requirements	10
	4.3 Software Description	10
<b>5</b>	<b>SYSTEM DESIGN</b>	<b>17</b>
	5.1 System Architecture	17
	5.2 Use Case Diagram	19

<b>6</b>	<b>SYSTEM IMPLEMENTATION</b>	<b>20</b>
	6.1 Tools Implementation	20
	6.2 Federal Algorithm	23
<b>7</b>	<b>RESULT AND DISCUSSION</b>	<b>24</b>
	7.1 Outcome and Effects	24
	7.2 Screenshots	27
<b>8</b>	<b>CONCLUSION AND FUTURE ENCHANCEMENTS</b>	<b>36</b>
	8.1 Conclusion	36
	8.2 Future Enhancements	37
	<b>APPENDIX</b>	<b>40</b>
	<b>REFERENCES</b>	<b>50</b>

## LIST OF FIGURES

FIGURE NUMBER	TITLE	PAGE NUMBER
5.1	Flow Chart Diagram	18
5.2	Use Case Diagram	19
7.1	Login	27
7.2	User Signup	28
7.3	Service Man Signup	29
7.4	Service Man Page	30
7.5	Service Page	31
7.6	Service Man List	32
7.7	Admin Page	33
7.8	Total User Data	34
7.9	Specific User Data	35



## **LIST OF ABBREVIATIONS**

<b>ABBREVIATION</b>	<b>EXPANSION</b>
SDK	Software Development Kit
UI	User Interface
IDE	Integrated Development Environment
SQL	Structured Query Language
MTV	Model Template View
MVC	Model View Control
REST	Representational State Transfer

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 INTRODUCTION**

LinkUp is a dynamic mobile app designed to connect users with trusted service providers for household and labor tasks. From cleaning to plumbing, LinkUp allows users to seamlessly book services, while service providers can showcase their skills and past work. The app fosters direct communication through real-time messaging, ensuring a smooth, efficient process. By creating a reliable, local marketplace, LinkUp aims to simplify service bookings and support community growth.

### **1.2 OVERVIEW**

LinkUp is an intuitive community service app that connects users with skilled professionals for household and labor services. Users can easily browse and book services like cleaning, plumbing, and electrical work, while service providers create detailed profiles to showcase their expertise. The app facilitates direct communication through real-time messaging, ensuring transparency and convenience. LinkUp enhances trust and reliability in service bookings, promoting a seamless and efficient experience for both users and providers.

### **1.3 OBJECTIVE**

The primary objective of LinkUp is to create a seamless platform that connects individuals in need of household and labour services with skilled professionals within their local community. The app aims to simplify the process of booking services by offering an easy-to-use interface where users can browse a wide range

of service categories, such as cleaning, plumbing, electrical work, and labour. LinkUp also strives to empower service providers by allowing them to build detailed profiles, showcase their skills, and gain visibility in their local area. Through transparent communication via real-time messaging, the app ensures that both users and service providers can discuss expectations and requirements before confirming bookings.

Another key objective is to foster trust and reliability within the community by offering user reviews and ratings for service providers. By creating a reliable and efficient service marketplace, LinkUp aims to promote convenience, support local businesses, and strengthen community connections.

## **1.4 PROBLEM STATEMENT**

In many communities, finding reliable, skilled service providers for household or labour tasks can be a challenging and time-consuming process. Users often rely on word-of-mouth referrals, which may not always lead to trustworthy or qualified professionals. Additionally, there is no centralized platform where users can easily browse, compare, and book services with confidence.

Users may struggle to find the right professionals for their needs, while service providers may face challenges in managing bookings, promoting their services, and building a reliable client base.

LinkUp addresses these issues by creating a streamlined platform that simplifies the service booking process, fosters transparent communication, and builds trust within local communities, benefiting both users and service providers.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 LITERATURE REVIEW**

Choudhury.L et.al,... [1] (2023) Enhancing Urban Service Delivery through Mobile Technology. discussed how mobile technology can play a transformative role in improving service delivery, particularly in densely populated urban areas. The study presented several case studies highlighting successful implementations of mobile platforms that connect users with essential services such as repair and maintenance. The authors emphasized that urban areas often face challenges related to service accessibility, reliability, and efficiency, which can be addressed through technology-driven solutions. They explored how mobile applications can streamline service delivery by offering real-time updates, tracking features, and user reviews, thereby ensuring transparency and accountability. The research also highlighted the potential for these platforms to create employment opportunities.

Gupta.A, et.al,... [2] (2020) The Role of Mobile Apps in Enhancing Daily Life Services. Investigated the impact of mobile applications on improving access to everyday services in urban settings. The study highlighted the growing dependence on mobile technology for managing day-to-day tasks and the role of apps in connecting users with essential services such as maintenance, repairs, and deliveries. The authors emphasized the importance of real-time service matching and efficient search functionalities in enhancing user satisfaction. The research underscored that well-designed mobile apps with user-friendly interfaces could significantly reduce the complexity associated with finding and booking local services.

Hendrik Santoso Sugiarto, et.al,... [3] (2019) On Analysing Supply and Demand in Labor Markets: Framework, Model and System. IEEE International Conference on Data Science and Advanced Analytics (DSAA). The above system is a framework for analyzing labor markets using supply and demand theory, leveraging data from job postings and applications. It calculates equilibrium salaries and job quantities to model labor market dynamics. Applied to Singapore's job market, it examines trends, biases (e.g., gender, age), and salary insights. The study introduces a Wage Dashboard to assist employers, job seekers, and policymakers with actionable insights into market conditions, competitiveness, and salary adjustments.

Jianan, Zhang, et.al,... [4] (2021) Research on the Transfer of Rural Labor Force under the Construction of Intelligent Society. International Conference on Public Management and Intelligent Society (PMIS). The system explores the challenges and strategies for transferring the rural labor force in China amidst the development of an intelligent society. It identifies key issues such as low education and skill levels, lack of organization, urban-rural labor market divides, and social problems like left-behind children and elderly. To address these, it recommends enhancing rural education, guiding orderly labor migration, unifying labor markets, and leveraging modern technology to improve family communication and rural labor competitiveness. This aims to support urbanization and the broader goals of an intelligent society.

Kumar.S, et.al,... [5] (2019) Mobile Service Platforms: Challenges and Opportunities. explored the transformative potential of mobile applications in developing countries, particularly focusing on their role in improving service

delivery and employment opportunities. They identified several key challenges that hinder the widespread adoption of mobile service platforms, such as user trust issues, lack of localized content, and interface complexity. These challenges often prevent users from fully utilizing the benefits of these platforms, especially in regions with diverse languages and varying literacy levels. The authors emphasized that creating localized, user-friendly applications tailored to the cultural and social context of users is essential for bridging service gaps. They argued that by addressing these challenges, mobile platforms could play a significant role in promoting small-scale employment and enhancing service accessibility.

Mehta.P, et.al,... [6] (2022) User-Centric Design in Mobile Service Applications. focused on the critical role of user-centric design principles in the success of mobile applications. The study revealed that many service-based platforms fail to retain users due to complicated interfaces and lack of intuitive navigation. The authors argued that simplicity, clarity, and responsiveness are essential factors that contribute to a positive user experience. They emphasized that designing applications with the end-user in mind ensures higher adoption rates and encourages long-term engagement. The research also explored the importance of incorporating feedback mechanisms and personalization features to cater to diverse user needs.

Patel.N, et.al,... [7] (2021) Digital Platforms and Employment in Emerging Economies. , examined the role of digital platforms in addressing unemployment and fostering economic participation in emerging markets. The research focused on how these platforms can bridge the gap between skilled professionals and

potential clients, creating new employment opportunities. The authors highlighted the challenges faced by freelancers and technical experts in reaching a wider audience and securing consistent work. They argued that digital platforms could mitigate these challenges by offering transparent and accessible channels for connecting service providers with consumers. The study also emphasized the importance of maintaining reliability and trust within these platforms to ensure user retention and satisfaction.

Vivek Kumar Sehgal, et.al,... [8] (2013) Job Portal - A Web Application for Geographically Distributed Multiple Clients. Acquiring knowledge and specific job skills have become the main objectives for students in the universities. Knowledge is necessary to make informed decisions, especially, in a critical situation. Knowledge and knowledge management (KM) in any organization are crucial to give it a competitive edge in today's challenging and globalised environment. In this paper authors have proposed a design of on-line recruitment system, that allows employers to post their job advertisements, which job seeker can refer to, when looking for jobs. This job portal is able to capture job requirements based on industry needs.

Yashi Jain, et.al,... [9] (2022) job portal: finding best job and best candidate. International Research Journal of Engineering and Technology (IRJET). The document presents a job portal application aimed at bridging the gap between job seekers and recruiters by offering a centralized platform for efficient recruitment. Designed as an Android application, it features two primary modules: one for job seekers and another for recruiters. Job seekers can register, create profiles with their resumes, search and apply for jobs, and communicate with recruiters through

a built-in chat system. Recruiters, on the other hand, can manage job postings, search for candidates based on specific criteria, and interact with applicants. The system overcomes traditional recruitment challenges, such as lack of direct communication and application status updates, by providing notifications and a user-friendly interface accessible via mobile devices.

Zaijin Lin, et.al,... [10] (2012) Labor Structure Wage and Efficiency of Economic Growth. 4th International Conference on Intelligent Human-Machine Systems and Cybernetics. The above system details the China's labor market shifts, highlighting a decline in low-end labor due to aging and an increase in educated, high-end labor. Low-end workers now command higher wages due to scarcity, while high-end workers face weaker bargaining power despite greater economic contributions. Younger workers drive productivity, but mismatched industrial structures hinder the employment of educated labor. The study stresses adapting industrial strategies to align with labor changes for sustained economic growth.

## **2.2 SUMMARY OF ISSUES**

- Service details are obtained via phone calls
- Lack of booking functionality
- Absence of notifications
- Time-consuming process



## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM:**

In the existing system, there is no dedicated application for home services, which significantly hampers the efficiency of operations. The worker details are shared only through a manual process, relying on outdated methods such as physical records, phone calls, or spreadsheets. This system is prone to errors, delays, and inconsistencies, leading to customer dissatisfaction and missed opportunities.

The absence of an integrated booking system means that customers cannot check the availability of workers or select specific services based on their requirements. Additionally, there is no easy way for customers to view worker profiles, such as their skills, ratings, or past job performance, which limits transparency and trust. With no option to book workers online, the system is constrained by outdated processes that ultimately affect both customer experience and business growth. The system also lacks the ability to track previous services or gather customer feedback, making it challenging for businesses to improve service quality, retain customers, or streamline their operations.

##### **3.1.1 DISADVANTAGES:**

- No booking facility.
- Notifications are not performed.
- Details are enquired through phone.
- It consumes more time

### **3.2 PROPOSED SYSTEM:**

In today's rapidly advancing technological landscape, smartphones play a crucial role in everyday life. However, individuals in India face challenges in locating and hiring skilled professionals for household and office repair services. New residents in urban areas also struggle to find reliable local technicians and employment opportunities.

To address this gap, this report proposes the development of a robust Android application using Flutter and Python Django, designed to connect end-users with technical experts. The app provides a user-friendly interface to bridge the communication gap, enabling seamless interactions between customers and professionals across four primary service categories: painters, electricians, mechanics, and plumbers. This expandable platform leverages advanced technology to ensure efficient, reliable, and transparent service access, ultimately improving daily convenience and fostering local employment opportunities

#### **3.2.1 ADVANTAGES:**

- Simple registration for customers and service providers.
- Users can search for professionals by category and location.
- Customers can book services and schedule appointments.
- Users can provide feedback to ensure quality control.
- The system can integrate new service categories beyond the initial four.

## **CHAPTER 4**

### **SYSTEM SPECIFICATON**

#### **4.1 SYSTEM HARDWARE:**

- Smartphone (Android 8.0 or higher)
- 2GB RAM or higher

#### **4.2 SOFTWARE REQUIRMENTS:**

- Flutter SDK
- Django framework
- Python 3.7 or higher
- SQL database
- Android Studio (for development).

#### **4.3 SOFTWARE DESCRIPTION:**

##### **FLUTTER SDK:**

Flutter widgets are built using a modern framework that takes inspiration from React. The central idea is that you build your UI out of widgets. Widgets describe what their view should look like given their current configuration and state. When a widget's state changes, the widget rebuilds its description, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.

The `runApp()` function takes the given Widget and makes it the root of the widget tree. In this example, the widget tree consists of two widgets, the Center widget and its child, the Text widget. The framework forces the root widget to

cover the screen, which means the text "Hello, world" ends up centered on screen. The text direction needs to be specified in this instance; when the `MaterialApp` widget is used, this is taken care of for you, as demonstrated later.

When writing an app, you'll commonly author new widgets that are subclasses of either `StatelessWidget` or `StatefulWidget`, depending on whether your widget manages any state. A widget's main job is to implement a `build()` function, which describes the widget in terms of other, lower-level widgets. The framework builds those widgets in turn until the process bottoms out in widgets that represent the underlying `RenderObject`, which computes and describes the geometry of the widget.

## **PYTHON DJANGO:**

Django is a high-level, open-source web framework written in Python that simplifies the development of complex, database-driven websites. Designed for rapid development and clean, pragmatic design, Django follows the Model-Template-View (MTV) architecture, which is a variation of the more commonly known Model-View-Controller (MVC) pattern. It provides a powerful Object-Relational Mapping (ORM) system, which allows developers to interact with the database using Python objects instead of SQL queries, streamlining database operations and ensuring security.

Django comes with a built-in admin interface that automatically generates a web-based dashboard for managing database records, making content management intuitive and efficient. Its robust security features include protection against SQL injection, Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS), and clickjacking, ensuring the framework is secure by default.

Additionally, Django is highly scalable, modular, and flexible, allowing developers to build a wide range of applications—from simple websites to large-scale enterprise systems—while maintaining high performance and security. Its emphasis on convention over configuration, along with its comprehensive documentation, makes it an excellent choice for both beginner and advanced developers alike.

### **PYTHON 3.7:**

Python 3.7, released in June 2018, introduced several important features such as data classes, built-in breakpoint(), and improved performance with the introduction of a faster method for handling dictionary operations. It also enhanced type hinting, made asynchronous programming more efficient, and improved the handling of postponed evaluation of type annotations. Python 3.7 marked a step forward in optimizing both readability and performance, making it a stable version for production use while maintaining the flexibility Python is known for.

One of the most significant features was the introduction of data classes via the data classes module. Data classes simplify the creation of classes used for storing data by automatically generating methods such as `__init__`, `__repr__`, and `__eq__`, drastically reducing boilerplate code and making class definitions more readable and concise. Another major improvement was the addition of the built-in `breakpoint()` function, which provides an easy way to enter the Python debugger (PDB) at any point in the code. This feature streamlined the debugging process by eliminating the need for manual `import pdb` statements.

Python 3.7 also brought postponed evaluation of type annotations (PEP 563), which deferred the evaluation of type hints until runtime. This change helps to avoid issues with forward references and circular imports in type annotations, making the type system more flexible and easier to work with. Additionally, there were significant performance improvements, particularly in the implementation of dictionaries, which became faster due to changes in memory layout and access patterns. This optimization led to more efficient handling of key-value pairs, especially in cases where dictionaries are frequently accessed or modified. Other improvements included enhanced support for asynchronous programming with better handling of asyncio, faster startup times for Python programs, and improvements to the handling of memory management.

Furthermore, Python 3.7 also introduced better error messages and enhanced debugging capabilities, making it easier for developers to diagnose and fix issues in their code. The release focused on refining the language's core features, offering a more stable and efficient environment for both beginner and advanced developers. Overall, Python 3.7 was a significant step forward in terms of both performance and developer productivity, and it laid the groundwork for even further improvements in subsequent releases.

## **SQL DATABASE:**

An SQL database (Structured Query Language database) is a type of relational database that uses SQL, a standardized programming language, for managing and manipulating structured data. SQL databases are based on the relational model, which organizes data into tables (also known as relations), consisting of rows (records) and columns (attributes or fields). Each table typically represents an

entity, such as customers, orders, or products, and the columns represent the properties or attributes of that entity. Relationships between tables are established through keys, with primary keys uniquely identifying records within a table, and foreign keys linking records across different tables.

SQL databases provide powerful query capabilities, allowing users to retrieve, update, delete, and insert data using SQL commands such as `SELECT`, `INSERT`, `UPDATE`, and `DELETE`. Data integrity and consistency are ensured through constraints (e.g., `NOT NULL`, `UNIQUE`, and `CHECK`) and ACID properties (Atomicity, Consistency, Isolation, Durability), which ensure that database transactions are processed reliably. These databases support complex operations like joins, aggregations, and subqueries, making them highly effective for handling large volumes of structured data and supporting complex business applications.

Some popular SQL database management systems (DBMS) include MySQL, PostgreSQL, Microsoft SQL Server, and Oracle Database. These systems are widely used in various applications, from small websites to large enterprise systems, due to their reliability, scalability, and support for complex queries. SQL databases are typically used in scenarios where data is highly structured and needs to be consistently maintained and queried, such as in banking systems, e-commerce platforms, and customer relationship management (CRM) software. They provide a solid foundation for data storage, management, and retrieval in numerous industries.

## **ANDROID STUDIO:**

Android Studio is the official integrated development environment (IDE) for building Android applications, developed by Google. It is based on JetBrains' IntelliJ IDEA and provides a comprehensive set of tools and features specifically designed for Android development. Android Studio supports both Java and Kotlin programming languages, with Kotlin being the preferred language recommended by Google for modern Android app development due to its conciseness and enhanced features. The IDE offers an array of powerful tools, including a visual layout editor, code editor, debugging tools, and performance analyzers, making it an essential platform for Android developers.

Android Studio's key features include the Android Emulator, which allows developers to test and debug their applications on a variety of virtual devices without needing physical hardware, and the Layout Editor, which helps in designing user interfaces (UI) through a drag-and-drop interface, allowing developers to create responsive layouts for different screen sizes and resolutions. Additionally, Android Studio includes built-in support for Gradle, a powerful build automation system, which handles app building, packaging, and dependency management efficiently. It also integrates with Android SDK (Software Development Kit), providing essential libraries, tools, and APIs for Android app development.

Other notable features of Android Studio include Lint tools for code quality analysis, Firebase integration for backend services like authentication, real-time databases, and cloud storage, and instant run for faster app deployment. The IDE also supports version control systems like Git, offering seamless collaboration



features. Android Studio is available for Windows, macOS, and Linux, and its robust ecosystem, along with constant updates from Google, makes it a powerful and essential tool for Android developers. It enables developers to build high-performance, feature-rich mobile applications for smartphones, tablets, wearables, and other Android-powered devices.

### **DART:**

Dart is a general-purpose, object-oriented programming language developed by Google, primarily used for building cross-platform mobile, web, and desktop applications. It is most commonly associated with the Flutter framework, a UI toolkit also developed by Google, which allows developers to create natively compiled applications from a single codebase. Dart is designed to be efficient, easy to learn, and highly productive, with features that support both functional and object-oriented programming paradigms. It is a statically-typed language with a strong emphasis on performance, enabling developers to build fast, responsive applications.

One of the key features of Dart is its just-in-time (JIT) compilation during development, which allows for fast iteration and debugging, and ahead-of-time (AOT) compilation for production builds, ensuring high performance on mobile and web platforms. Dart's rich standard library provides a wide range of built-in libraries for networking, file I/O, and asynchronous programming, making it well-suited for modern app development. Dart also features null safety, which reduces runtime errors by ensuring that null values are handled explicitly, improving code reliability and developer productivity.

## CHAPTER 5

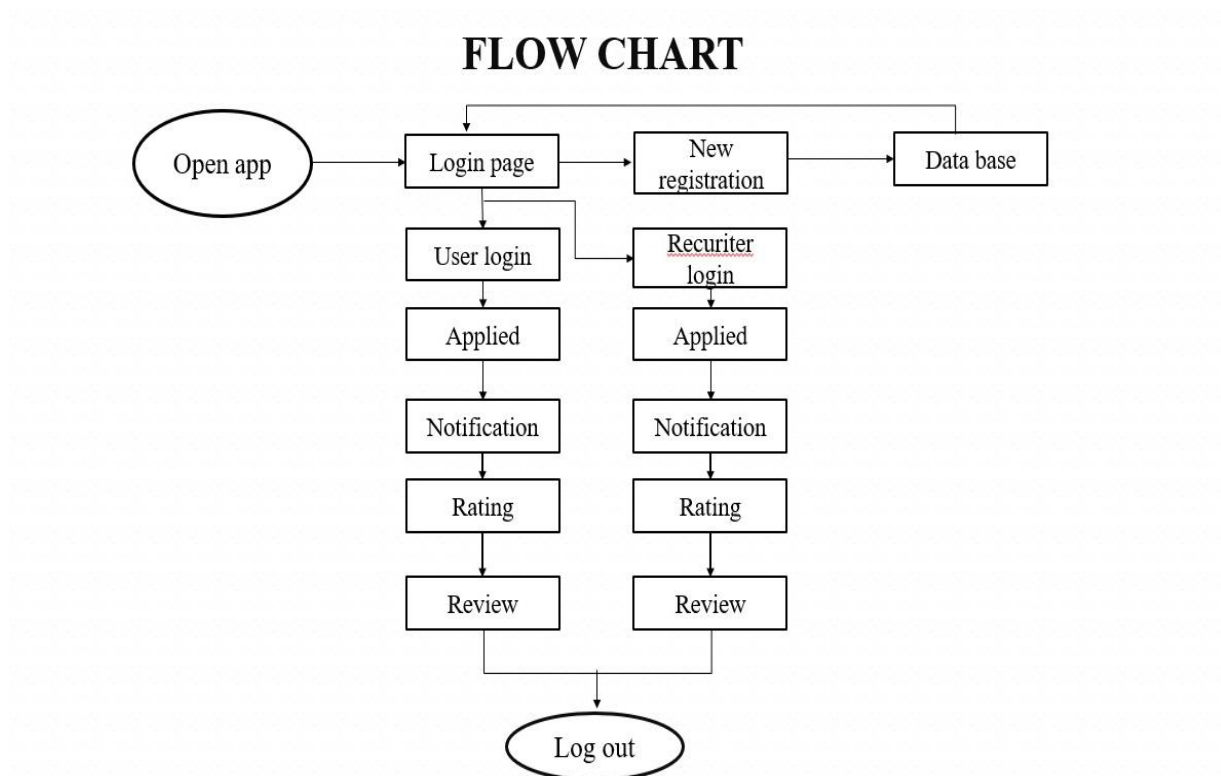
### SYSTEM DESIGN

#### 5.1 SYSTEM ARCHITECTURE

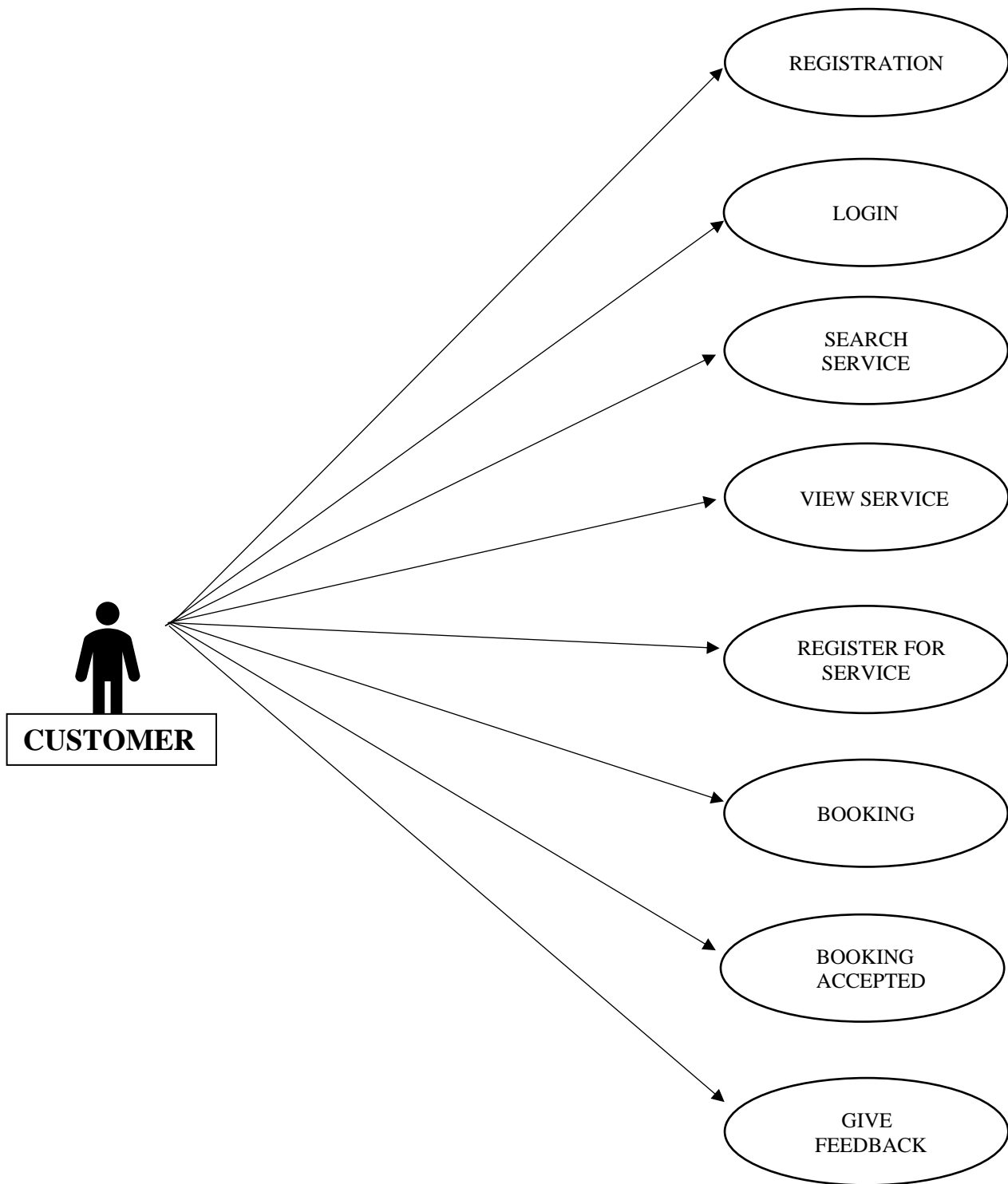
The Linkup app architecture is designed as a modular, scalable, and secure system, supporting community service discovery and interaction.

- **Frontend (Client Layer):** Built using frameworks like React Native or Flutter for cross-platform compatibility (iOS, Android). Provides features for user registration, service search, service reviews, and seamless navigation through expandable categories. Uses a responsive design to support diverse devices and screen sizes.
- **Backend (Application Layer):** Built on frameworks like Django or Node.js, ensuring high performance. Manages APIs for user and service data, authentication, search functionality, and real-time service updates. Incorporates a scalable database schema for handling user profiles, services, reviews, and categories.
- **Database Layer:** A relational database like PostgreSQL or MySQL for structured data (e.g., user/service details). Elasticsearch or similar for optimizing search queries across services.
- **Cloud and Storage:** Hosted on AWS, Azure, or Google Cloud for scalability. Secure file storage (e.g., S3 buckets) for service images or documents.
- **Microservices and Expandability:** Each service category (e.g., plumbing, tutoring) is managed as an independent module. APIs allow seamless addition of new categories without system-wide disruption.

- Security: Implements OAuth 2.0 for user authentication. SSL encryption for secure communication.
- Analytics and Reviews: A review engine collects user feedback and provides data for service improvement. Dashboard analytics for admins to monitor app performance and growth. This architecture ensures a robust foundation for Linkup, supporting dynamic growth and user engagement.



**FIGURE 5.1 FLOW CHART**



**FIGURE 5.2 USECASE DIAGRAM**

## CHAPTER 6

### SYSTEM IMPLEMENTATION

The integration of Flutter, Django, SQL, Android Studio, and Dart creates a cohesive and efficient system architecture. Flutter's dynamic front-end capabilities, combined with Django's powerful backend infrastructure, ensure smooth communication and data processing. The SQL database provides a reliable foundation for data storage and retrieval, while Android Studio facilitates streamlined development and testing. This robust combination ensures that the application is scalable, secure, and user-friendly, addressing the needs of both service providers and end-users.

#### 6.1 TOOLS IMPLEMENTATION:

- **Flutter** is used for the front-end development of the application. It ensures a seamless and user-friendly interface that allows users to search for professionals, book services, and leave reviews. The flexible architecture of Flutter ensures that the app is responsive and can handle complex UI components, enhancing the overall user experience.
- **Django** serves as the backend framework, managing data processing, business logic, and API development. It handles user registration, service requests, and interactions between customers and professionals. The Django framework ensures that data is processed efficiently, securely stored, and easily retrievable by the front end, providing a solid foundation for the application's backend infrastructure.

- **SQL** database is used to store and manage all critical data, including user information, service provider details, service categories, bookings, and transaction histories. The structured approach ensures that data can be accessed and managed efficiently, supporting the application's functionality and performance under high user loads.
- **Android Studio** is used as the primary development environment for building and testing the Android application. It facilitates the integration of Flutter, provides tools for debugging and optimization, and ensures that the final product meets performance and reliability standards.
- **Dart** is a modern, high-performance programming language that offers several features to enhance developer productivity and application performance. One of its key attributes is strong typing, which ensures that errors are caught early during development, resulting in more robust and maintainable code. Dart also excels in asynchronous programming, providing built-in support for handling tasks such as data fetching, user input, and other time-consuming operations without blocking the main thread. It allows applications to remain responsive, even when performing complex or long-running tasks. Dart supports both Just-In-Time (JIT) compilation and Ahead-Of-Time (AOT) compilation. JIT compilation speeds up development by enabling features like hot reload, which allows developers to see code changes immediately during the development process, while AOT compilation enhances the performance of production apps by compiling the code directly to machine code.
- Dart's combination of modern language features, performance

optimizations, and a supportive ecosystem, particularly with Flutter, makes it an excellent choice for developing cross-platform applications for mobile, web, and desktop environments.

- Django REST Framework (DRF) is a powerful and flexible toolkit for building web APIs in Django, a popular Python web framework. It simplifies the process of creating RESTful APIs by providing features like serialization, viewsets, and authentication. DRF allows complex data types, such as Django model instances, to be converted into Python-native data types and then rendered as JSON or XML, making it easy to handle data in web applications. With viewsets, DRF automatically provides basic CRUD operations for models, and routers generate URLs for these viewsets, reducing the need for repetitive code. The framework supports various authentication methods (such as token-based and session authentication) and lets developers easily implement fine-grained permissions to control access to API endpoints. DRF also provides built-in support for pagination, filtering, and throttling, ensuring that APIs can efficiently handle large datasets, manage high traffic, and maintain security. A standout feature is theBrowsable API, which allows developers to interact with and test APIs directly through a web interface. Additionally, DRF integrates smoothly with Django's security features and offers tools for easily managing errors and validations. Overall, Django REST Framework streamlines API development by offering a rich set of tools for building scalable, secure, and maintainable web services.

## **6.2 FEDERAL ALGORITHM:**

A federal algorithm generally refers to algorithms designed for distributed systems, particularly in the context of federated learning. Federated learning is a machine learning approach where multiple decentralized devices or systems collaboratively train a model while keeping the data localized. Instead of transferring the data to a central server for processing, the model training happens on local devices, and only model updates (like weights or gradients) are sent back to a central server. This approach ensures data privacy, reduces the need for massive data transfer, and enables more efficient and scalable learning across a large number of devices.

In the context of federated learning, the federal algorithm refers to the methods and strategies used to aggregate the locally trained models or gradients to update the global model. A common example of this aggregation is the FedAvg (Federated Averaging) algorithm, where local models trained on various devices are averaged to form a global model, which is then sent back to the devices for further training. This decentralized model training is particularly useful in scenarios where data privacy and security are paramount, such as in healthcare, mobile devices, or financial services.



## **CHAPTER 7**

### **RESULT AND DISCUSSION**

#### **7.1 OUTCOME AND EFFECT:**

The Link Up - Community Service App was successfully developed to connect customers with service providers in an easy-to-use and efficient platform. The core features of the app include User Registration, Service Search, Service Booking, Reviews and Ratings, and Expandable Categories.

- **User Registration:** The registration process is straightforward, allowing both customers and service providers to create accounts and specify their roles. This ensures that users have customized access to the platform based on whether they are seeking services or offering them.
- **Service Search:** The app allows customers to search for professionals by service category (e.g., plumbing, tutoring, cleaning) and location. This feature makes it easier for users to find relevant services nearby, enhancing the app's usability and convenience.
- **Service Booking:** After finding a suitable service provider, customers can book services directly through the app and schedule appointments based on availability. This feature streamlines the process of scheduling and ensures that both customers and service providers are synchronized in terms of timing.
- **Reviews and Ratings:** To ensure high service quality, the app includes a review and rating system where customers can leave feedback on the services they receive. This not only helps future users make informed decisions but also holds service providers accountable for the quality of their work.

- **Expandable Categories:** The app was designed with scalability in mind. While it initially supports four core service categories, it is easily expandable to incorporate more categories in the future based on user demand and market trends. This ensures that the app remains flexible and adaptable to a growing range of community service needs.

The app was successfully tested, and user feedback showed positive results. Users reported a smooth experience in registering, searching for services, booking appointments, and providing reviews. The system's ability to scale with new service categories was also confirmed during the testing phase, with the app showing flexibility to add new service types.

## **DISCUSSION:**

The development of the Link Up - Community Service App addresses a clear gap in the market for an easy-to-use, flexible platform connecting customers with service providers in their local communities. The core features of the app were designed with simplicity and user experience in mind, ensuring that both customers and service providers can easily navigate the system.

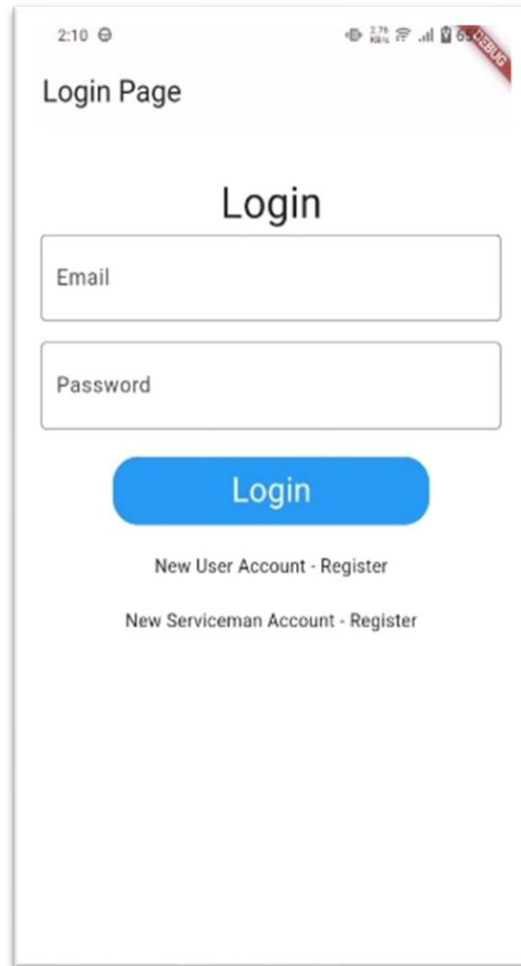
One of the key strengths of the app is its User Registration and dual-access system for customers and service providers. This clear differentiation between the two user roles ensures that users can access the features most relevant to them without confusion. The Service Search functionality adds another layer of convenience, allowing customers to filter results based on location and category, which is crucial for ensuring they find the right services in their area.

The Service Booking feature is highly beneficial as it streamlines the process of scheduling appointments, reducing back-and-forth communication between customers and providers. This makes the platform much more efficient compared to traditional methods of finding and booking services.

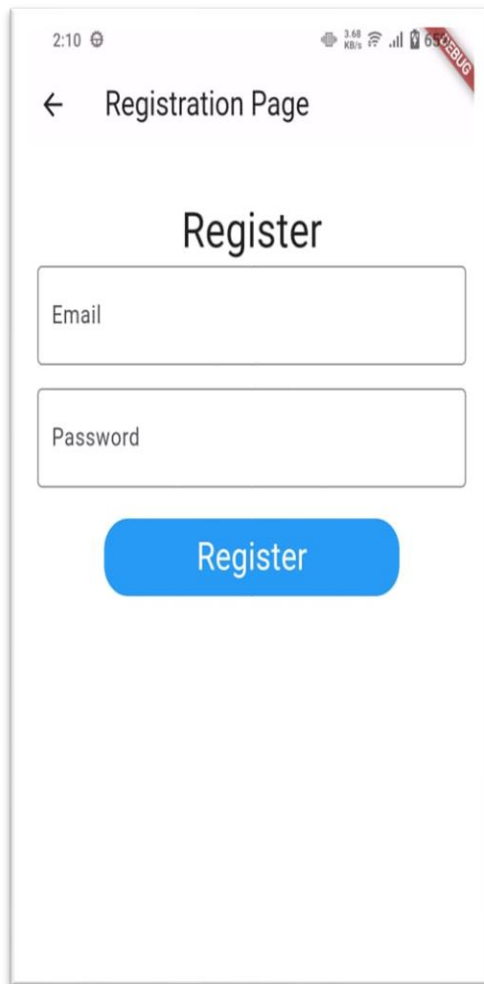
### **CHALLENGES:**

The drawbacks, which are faced during existing system, can be eradicated system is by using the proposed application. The main objective of the proposed to provide a user-friendly application for society user. The intention of this project aimed at develop is to is to minimize the manual interaction. This project is minimize the manual process. Project proposes a new android application which can be used when a user needs any home related services. The registered user can login to this application and make request for service employee based on location. The application will list out the registered home related service employee in that area. The user can pick a service employee and make request for his service. Based on request service employee they can select or reject user request effectively. Proposed to make the new system extremely user friendly. This should develop to minimizing the manual process.

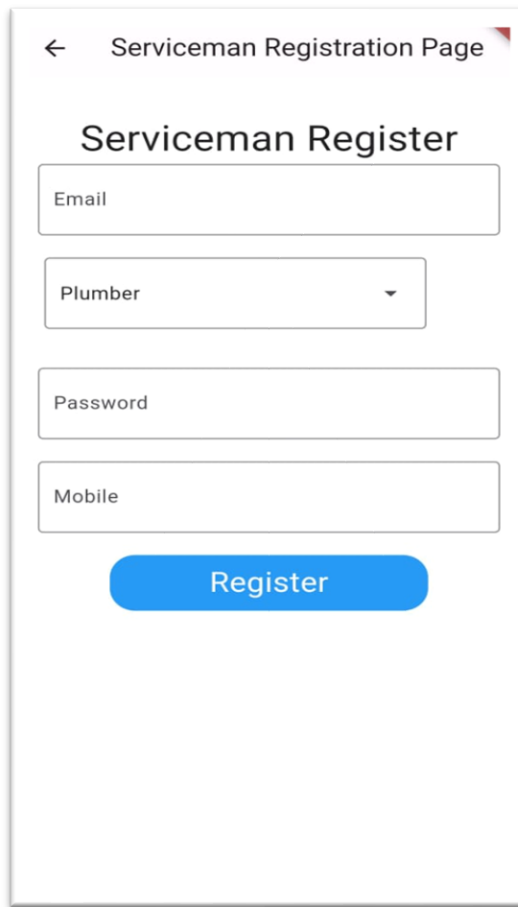
## 7.2 SCREENSHOTS:



**Figure 7.1** Login a login page with fields for "Email" and "Password" and a prominent blue "Login" button. It also includes options for new users to register as either "New User Account" or "New Serviceman Account." The layout is minimalistic and user-friendly.



**Figure 7.2** User signup a registration page with fields for "Email" and "Password" and a blue "Register" button. The design is simple and focused on collecting basic user details. A back arrow at the top allows users to navigate to the previous screen

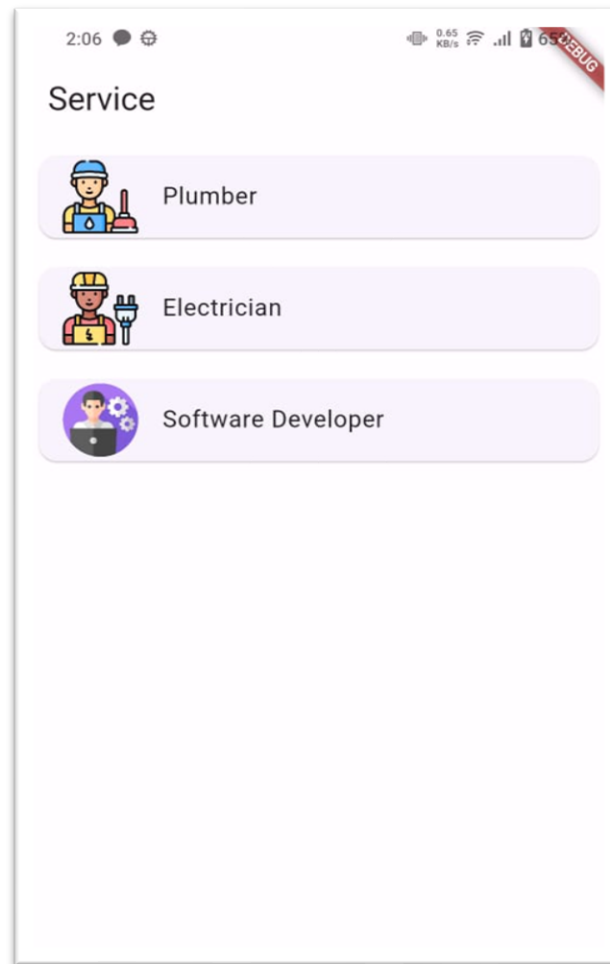


The image shows a mobile application interface for a 'Serviceman Registration Page'. At the top, there is a back arrow and the page title 'Serviceman Registration Page'. Below this is the main heading 'Serviceman Register'. The form consists of four input fields: 'Email', a dropdown menu currently showing 'Plumber', 'Password', and 'Mobile'. At the bottom of the form is a prominent blue button labeled 'Register'.

**Figure 7.3** Service Man Signup a Serviceman Registration Page with a clean and minimalistic design. It includes input fields for the user to enter their email, password, and mobile number. Additionally, there is a dropdown menu to select a profession, with "Plumber" as the default option. At the bottom, a prominent blue "Register" button is displayed, making the interface simple and user-friendly.

The image shows a web form titled "Serviceman Home" with a sub-header "Serviceman Register". The form contains several input fields: "Email" with the value "456service@mail.com", a "Plumber" dropdown menu for profession, "Mobile" with the value "09876", and empty fields for "Name", "Location", "Experience", and "Salary". A prominent blue "Update" button is located at the bottom of the form.

**Figure 7.4** Service Page a Serviceman Register form under the "Serviceman Home" page. It includes fields for entering details such as email, profession (with a dropdown, defaulting to "Plumber"), mobile number, name, location, experience, and salary. At the bottom, a blue "Update" button is prominently placed for submitting the form, maintaining a clean and organized layout.

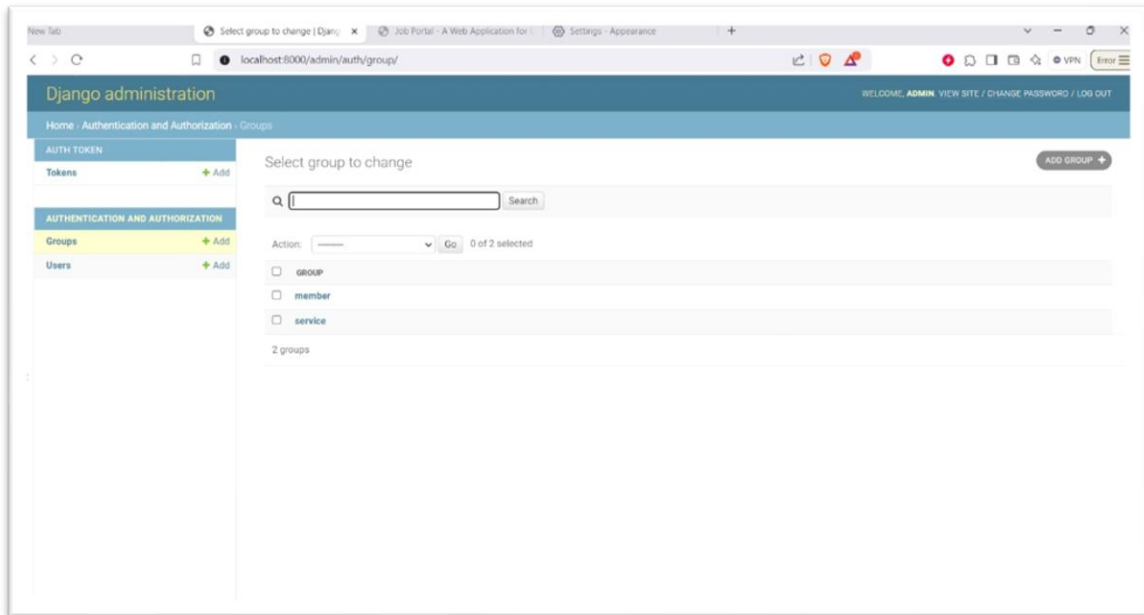


**Figure 7.5** a clean and minimalistic user interface under the heading "Service," featuring three selectable options: Plumber, Electrician, and Software Developer. Each option is represented with an appropriate icon—Plumber with plumbing tools, Electrician with electrical tools, and Software Developer with a laptop and gear symbols—ensuring clear visual cues. The design is simple yet effective, with neatly aligned buttons that make navigation intuitive and user-friendly.

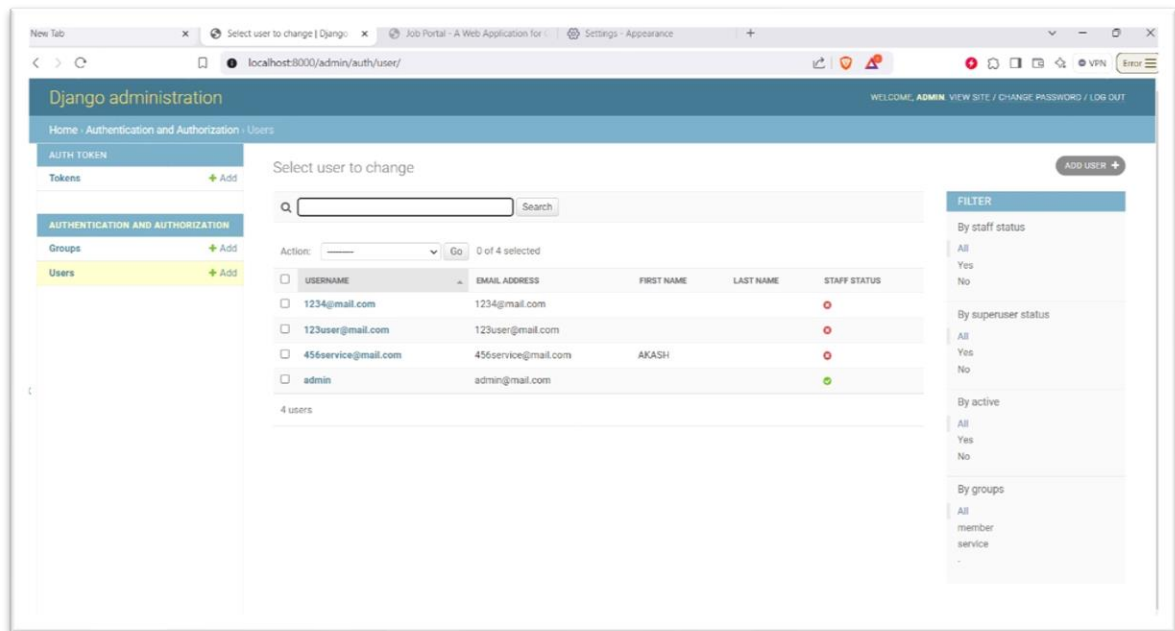




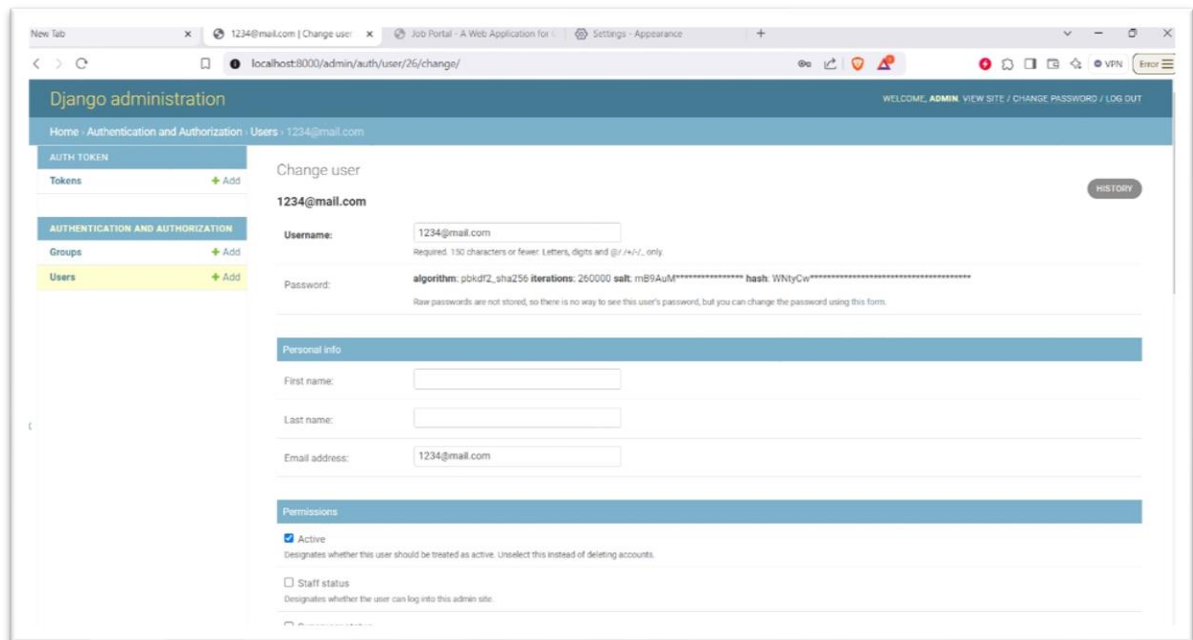
**Figure 7.6** a Serviceman List page with details of a serviceman named Akash. It includes information such as his mobile number, email, 1 year of experience, location (Salem), and a daily salary of 1000. The design features a clean, centered card layout to present the details in an organized manner.



**Figure 7.7** the Django administration panel, specifically the "Groups" section under "Authentication and Authorization." It allows the admin to manage user groups, with options to add new groups or edit existing ones. The interface lists groups such as "member" and "service," providing a search bar and bulk action functionality for efficient group management.



**Figure 7.8** the Django administration panel in the "Users" section under "Authentication and Authorization." It shows a list of users with details like username, email address, first name, last name, and staff status. Admins can manage users by adding new ones, editing existing ones, or applying bulk actions. Filters on the right side allow sorting by staff status, superuser status, activity, or group membership for easier user management.



**Figure 7.9** the Django admin interface for editing a user's details. It includes fields like the username, password (hashed and secured), personal information (first name, last name, and email), and permissions such as whether the user is active or has staff privileges. The interface allows administrators to manage user accounts securely.

## **CHAPTER 8**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

#### **8.1 CONCLUSION:**

The LinkUp Community Service App has established a solid foundation for connecting customers with service providers in a user-friendly, efficient, and scalable manner. Its intuitive features—such as user registration, service search, booking, reviews, and expandable categories—address key needs for both customers and service providers, enhancing the overall experience.

The use of Flutter for the front-end ensures a smooth and responsive user interface across multiple platforms, while Python Django and a SQL database for the back-end provide a reliable and robust infrastructure to handle user data, service requests, and professional information.

The simplicity of the app's design allows for easy adoption by a broad user base, and the flexibility to add new service categories means the app can continue to expand and adapt to market demands. The reviews and ratings system helps maintain high-quality services, ensuring trust between customers and professionals. Overall, LinkUp serves as a bridge to foster community engagement by streamlining service transactions, making it a valuable platform for both users and providers.

## **8.2 FUTURE ENHANCEMENTS:**

- **Dynamic Service Pricing:**Implement dynamic pricing based on factors such as demand, time of day, location, or the service provider's experience. This would allow for flexible pricing, benefiting both customers and providers. For instance, prices could be adjusted during peak times or for high-demand services, enhancing profitability for providers while still offering competitive rates to customers.
- **Geo-Location & Augmented Reality (AR) Integration:**Use geo-location to offer real-time tracking of service providers, allowing customers to see the estimated time of arrival and track the service professional's progress. Augmented Reality (AR) could also be used in certain service categories (e.g., home improvement, cleaning) to provide virtual previews or simulations of how a service will look once completed.
- **Service Bundles and Packages:**Offer service bundles where customers can book a set of related services at a discounted price. For example, a "Home Renovation" package could include plumbing, electrical, and painting services, providing convenience for the customer and increasing revenue for service providers.
- **Subscription Model for Frequent Users:**Introduce subscription-based plans for customers who need recurring services, such as monthly cleaning, maintenance, or repairs. This could offer discounts and benefits, encouraging customer loyalty and providing service providers with a predictable, ongoing stream of work.

- **Service Provider Verification & Training:**Implement a more thorough verification process for service providers, including background checks, certifications, and proof of skills. Additionally, LinkUp could offer training resources or certifications to professionals, ensuring that only qualified individuals are offering services and maintaining the app's high standards.
- **Local Community Features:**Add a social component that allows users to create profiles, share experiences, and recommend service providers within their local community. This feature could also include a community forum for users to discuss service trends, ask questions, and share advice.
- **Eco-Friendly Service Options:**For customers interested in environmentally conscious choices, add a category for eco-friendly or sustainable services. For example, green cleaning services, eco-friendly construction materials, or energy-efficient home improvements could appeal to a growing customer base focused on sustainability.
- **Advanced User Analytics for Personalization:**Use machine learning to analyze user behavior and preferences, offering personalized service recommendations, tailored discounts, or custom promotions. By understanding patterns and customer preferences, LinkUp can provide an even more relevant and personalized experience.
- **Sustainability and CSR Initiatives:**Build a feature that allows service providers to participate in sustainability or corporate social responsibility (CSR) initiatives. For example, providers could partner with charitable

organizations, plant trees for every service booked, or offer free services to underserved communities, creating goodwill and positive brand image.

- **Virtual Assistant for Customer Support:** Integrate a virtual assistant or chatbot to handle common customer inquiries, such as service inquiries, pricing, or troubleshooting. This would help reduce the workload on support teams while providing immediate assistance to users.
- **In-App Advertising & Sponsored Listings:** To generate additional revenue, introduce an advertising model where service providers can pay for premium listings or targeted ads. These ads could appear in search results, on the home screen, or in category-specific pages.

By implementing these additional features, LinkUp can further enhance its value proposition and cater to the evolving needs of both customers and service providers. The future enhancements would not only improve user satisfaction but also position LinkUp as a competitive, future-ready app in the rapidly growing community service market.



## APPENDIX

### SOURCE CODE:

#### LOGIN PAGE

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:http/http.dart' as http;
import 'package:service_app/home_service.dart';
import 'package:service_app/register.dart';
import 'package:service_app/service_register.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'const.dart';
import 'home.dart';

class LoginDemo extends StatefulWidget { @override
  _LoginDemoState createState() => _LoginDemoState();}
class _LoginDemoState extends State<LoginDemo> {
  String email = "user@mail.com";
  String password = "123456";
  showToast(String msg) {
    Fluttertoast.showToast(
      msg: msg,
      toastLength: Toast.LENGTH_SHORT,
      gravity: ToastGravity.CENTER,
      timeInSecForIosWeb: 1,
      backgroundColor: Colors.red,
      textColor: Colors.white,
```

```

fontSize: 16.0);}
save() async {
try {
Map<String,String> body={
"username":email,
"password":password};
http.Response response=await http.post(Uri.parse(login_url),body: body);
print(response.body);
if(response.statusCode==200){
Map<String,dynamic> data=jsonDecode(response.body);
showToast("Login success");
final SharedPreferences prefs = await SharedPreferences.getInstance();
print(data['token']);
await prefs.setString('token', data['token']!);
await prefs.setString('id', data['id']!.toString());
if(data['group']=="member"){
print("member");
Navigator.pushReplacement(context, MaterialPageRoute(builder:
(context)=>HomePage()));}
if(data['group']=="service"){
print("member");
Navigator.pushReplacement(context, MaterialPageRoute(builder:
(context)=>ServiceHome())); } }
else{
showToast("Login fail");}}
catch (e) {

```

```

print(e);
showToast("Login fail");}}
"Food quality"))));
Widget build(BuildContext context) {
return Scaffold(
  backgroundColor: Colors.white,
  appBar: AppBar(
    title: Text("Login Page"),),
  body: SingleChildScrollView(
    child: Column(
      children: <Widget>[
        SizedBox(
          height: 30,),
        Container(
          child: Text(
            "Login",
            style: TextStyle(fontSize: 32),)),{
          email = e;},),),
        Padding(
          padding: const EdgeInsets.only(
            left: 15.0, right: 15.0, top: 15, bottom: 0),
          padding: EdgeInsets.symmetric(horizontal: 15),
          child: TextField(
            obscureText: true,
            decoration: InputDecoration(
              border: OutlineInputBorder(),

```

```

labelText: 'Password',
hintText: 'Enter secure password'),
onChanged: (e) {
password = e;},),),
InkWell(
onTap: () {
save();},
child: Container(
alignment: Alignment.center,
margin: EdgeInsets.all(20),
height: 50,
width: 250,
decoration: BoxDecoration(
color: Colors.blue,
borderRadius: BorderRadius.circular(20)),
child: Text(
'Login',
style: TextStyle(color: Colors.white, fontSize: 25),),),),
Container(
child: InkWell(
onTap: () {
Navigator.push(context,
MaterialPageRoute(builder: (context) => Register()));},
child: Text("New User Account - Register")),),
SizedBox(height: 20,)
Container(

```

```

child: InkWell(
  onTap: () {
    Navigator.push(context,
      MaterialPageRoute(builder: (context) => SRegister()));},
  child: Text("New Serviceman Account - Register")),),),),);
}
}

```

## HOME PAGE

```

import 'package:flutter/material.dart';
import 'package:service_app/login.dart';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Service app',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: LoginDemo(),
    );
  }
}

```

```
}
```

## **SERVICE MAN PAGE**

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:service_app/const.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
class ServiceManList extends StatefulWidget {
  const ServiceManList({super.key, required this.sname});
  final String sname;
  @override
  State<ServiceManList> createState() => _ServiceManListState();
}
class _ServiceManListState extends State<ServiceManList> {
  List services = [];
  showToast(String msg) {
    Fluttertoast.showToast(
      msg: msg,
      toastLength: Toast.LENGTH_SHORT,
      gravity: ToastGravity.CENTER,
      timeInSecForIosWeb: 1,
      backgroundColor: Colors.red,
      textColor: Colors.white,
      fontSize: 16.0);
  }
}
```

```

}

getData(String sname) async {
  try {
    final SharedPreferences prefs = await SharedPreferences.getInstance();
    String token = prefs.getString('token')!;
    http.Response response = await
    http.get(Uri.parse("$serviceman_url?q=$sname"),headers:
    {'Authorization':"token $token"});
    if (response.statusCode == 200) {
      showToast("get data success");
      setState(() {
        services=jsonDecode(response.body);
      });
      print(services);
    } else {
      showToast("Fetch fail");
    }
  } catch (e) {
    print(e);
    showToast("Fetch fail");
  }
}

@override
void initState() {
  super.initState();

```

```

getData(widget.sname); }
@override
Widget build(BuildContext context) {
return Scaffold(
  appBar: AppBar(
    title: Text("Serviceman List"),
  ),
  body: Container(
    child: Column(
      children:
    [
      Container(
        height: 300,
        child: ListView.builder(
          itemCount: services.length,
          itemBuilder: (context, index) {
            return Card(
              margin: const EdgeInsets.all(10),
              child: Container(
                margin: EdgeInsets.all(20),
                child: Column(
                  children: [
                    Text( "Name: "+services[index]['name']!),
                    Text( "Mobile: "+services[index]['mobile']!),
                    Text( "Email: "+services[index]['email']!),
                    Text( "Experience: "+services[index]['experience']!),

```



```
Text( "Location: "+services[index]['address']!),  
Text( "Salary: "+services[index]['salary_scale']!),
```

## **SERVICE MAN DETAILS**

```
from django.db import models  
from django.contrib.auth.models import User  
# Create your models here.  
class AppUser(models.Model):  
    user=models.ForeignKey(to=User,on_delete=models.CASCADE)  
    mobile=models.CharField(max_length=20)  
    address=models.CharField(max_length=200)  
    def todict(self):  
        dct={ }  
        dct['email']=self.user.email  
        dct['mobile']=self.mobile  
        dct['group']='member'  
        return dct  
class Serviceman(models.Model):  
    user=models.ForeignKey(to=User,on_delete=models.CASCADE)  
    service=models.CharField(max_length=50)  
    mobile=models.CharField(max_length=20)  
    address=models.CharField(max_length=200,null=True,blank=True)  
    experience=models.CharField(max_length=200,null=True,blank=True)  
    salary_scale=models.CharField(max_length=100,null=True,blank=True)  
    def todict(self):
```

```
dct={ }  
dct['email']=self.user.email  
dct['name']=self.user.first_name if self.user.first_name else ""  
dct['mobile']=self.mobile  
dct['service']=self.service  
dct['address']=self.address if self.address else ""  
dct['experience']=self.experience if self.experience else ""  
dct['salary_scale']=self.salary_scale if self.salary_scale else ""  
dct['group']='service'  
return dct
```

## REFERENCES

- [1] Choudhury. L, A. Singh (2023) Enhancing Urban Service Delivery through Mobile Technology.
- [2] Gupta. A, M. Joshi (2020) The Role of Mobile Apps in Enhancing Daily Life Service.
- [3] Hendrik Santoso Sugiarto, Ee-Peng Lim, Ngak-Leng Sim (2019) On Analysing Supply and Demand in Labor Markets: Framework, Model and System. IEEE International Conference on Data Science and Advanced Analytics (DSAA).
- [4] Jianan, Zhang, Shuoyi, Zhu (2021) Research on the Transfer of Rural Labour Force under the Construction of Intelligent Society. International Conference on Public Management and Intelligent Society (PMIS).
- [5] Kumar. S, R. Sharma (2019) Mobile Service Platforms: Challenges and Opportunities.
- [6] Mehta. P, R. Verma (2022) User-Centric Design in Mobile Service Applications.
- [7] Patel. N, K. Desai (2021) Digital Platforms and Employment in Emerging Economies.

- [8] Vivek Kumar Sehgal, Akshay Jagtiani, Meha Shah, Anupriya Sharma, Arpit Jaiswal, Dhananjay Mehta (2013) Job Portal - A Web Application for Geographically Distributed Multiple Clients.
- [9] Yashi Jain, Aneesha Jaykumar, Ateeba Jawaaid, Prof. Silviya D'monte (2022) job portal: finding best job and best candidate. International Research Journal of Engineering and Technology (IRJET).
- [10] Zaijin Lin, He Chen (2012) Labor Structure, Wage and Efficiency of Economic Growth. 4th International Conference on Intelligent Human-Machine Systems and Cybernetics.