

HOMECONNECT+

INTERDISCIPLINARY PROJECT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in Information Technology
by

POOJITHA K(411200138)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC

Accredited with Grade “A++” by NAAC

12B Status by UGC |Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

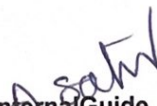
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

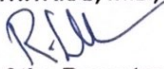
DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **POOJITHA K (41120138)** who carried out the project entitled "HomeConnect+" under my supervision from **January 2024** to **May 2024**.

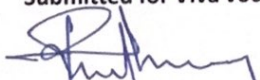

Internal Guide

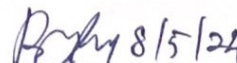
Dr A. SATHIYARAJ, M.E., PhD.,


Head of the Department

Dr R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 8/5/2024


Internal Examiner

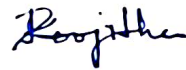

External Examiner

DECLARATION

I POOJITHA K hereby declare that the Project Report entitled “HomeConnect+” done by me under the guidance of Dr A. SATHIYARAJ, M.E., PhD., is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 8.5.2024

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing **Dr.R.Subhashini M.E.,Ph.D.**,Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr A. Sathiyaraj, M.E., PhD.**, for his valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE OF COMPLETION



HACKWIT TECHNOLOGIES PVT LTD
Redefine by Technologies

149, 1C, 2nd Floor, Mount Poonamallee Rd,
Ramapuram, Chennai - 600089.
admin@hackwittechnologies.com
www.hackwittechnologies.com

10.02.2024

CERTIFICATE FOR INTERNSHIP

TO WHOMSOEVER IT MAY CONCERN

This is to Certify that **Poojitha. K (Reg No: 41120138)** student of Sathyabama Institute of Science and Technology, B. Tech Information Technology, has successfully completed the Internship at Hackwit Technologies from 22nd January to 30th January 2024.

During the internship, **Poojitha. K** has closely worked as a part of the **Flutter** by carrying out the project entitled **"HomeConnect+"**.

She demonstrated good developing and learning skill with self-motivated attitude to learn new things. Her contribution to implement the Flutter was good. We wish her all the best for her future endeavors.

Best Regards,

Ravikumar S
CEO & Managing Director
Hackwit Technologies Pvt Ltd



CERTIFICATE OF COMPLETION



ABSTRACT

HomeConnect+ is a Flutter-based mobile application prototype that provides a user-friendly platform for scheduling and managing home services. It is poised to offer an efficient, reliable, and scalable solution for home service management and booking, epitomizing the convenience and innovation of modern mobile applications. It brings together a diverse range of services, such as cleaning, plumbing, and electrical work, into a single, cohesive experience. With its clean design and intuitive navigation, the app guides users through selecting services, customizing tasks, scheduling appointments, and viewing detailed profiles and ratings of service providers. At the core of HomeConnect+'s functionality is the integration with Firebase, offering robust back-end capabilities, including secure user authentication. This ensures that users can create personal accounts, log in with ease, and manage their bookings with confidence, knowing that their data is protected. The app's design emphasizes a streamlined user flow, convenience in service customization, and the ability to manage appointments according to the user's schedule—whether they need a one-time service or recurring appointments. As it stands, HomeConnect+ is a promising prototype that showcases the capabilities of Flutter for creating high-quality, cross-platform applications. With its current foundation, it is poised for further development to incorporate dynamic data handling, payment processing, and real-time updates, transforming it into a fully-fledged service booking solution.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vii
	LIST OF ABBREVIATIONS	x
	LIST OF FIGURES	xi
1	INTRODUCTION	1
1.1	Flutter	2
1.2	Dart	3
1.3	Firebase	5
1.4	Problem Statement	7
1.5	Objectives	9
2	METHODOLOGY	11
2.1	Initial Project Requirements	11
2.2	Proposed Application Architecture	15
2.3	Dataflow Diagram	21
2.4	Modules	22
	2.4.1 Authentication	22
	2.4.2 Service Selection	23
	2.4.3 Scheduling and booking	24
	2.4.4 Professional Profile and Ratings	25
2.5	Development and Deployment Setup	26

3	RESULT AND DISCUSSION	29
4	CONCLUSION AND FUTURE WORK	34
4.1	Conclusion	34
4.2	Future Work	35
	REFERENCES	36
	APPENDICES	39
A.	Sample Code	39
B.	Screen Shots	41

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
UI	- User Interface
UX	- User Experience
iOS	- iPhone Operating System
UAT	- User Acceptance Testing
IDE	- Integrated Development Environment

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
2.1	Flutter Architecture	15
2.2	Firebase Architecture	18
2.3	Proposed Application Architecture	19
2.4	Dataflow Diagram	21
2.5	Project Creation	26
2.6	Firebase Connection	27
2.7	Firebase Authentication	28
3.1	Signup Page Interface	29
3.2	List of service provided Interface	30
3.3	Date and time selection Interface	31
3.4	Firebase Configurations	32
3.5	Initialize Firebase	33

CHAPTER 1

INTRODUCTION

HomeConnect+ is a mobile application prototype crafted with Google's Flutter framework, serving as a comprehensive platform that facilitates the connection between homeowners and a broad spectrum of home services. It is meticulously designed for users who prioritize efficiency, reliability, and superior quality in home maintenance and improvement endeavors. The application epitomizes convenience, offering a seamless experience for users to discover, schedule, and manage services ranging from routine cleaning to complex electrical installations.

The interface of HomeConnect+ is remarkably intuitive, providing a straightforward and customizable navigation pathway for users to detail their service requirements. This level of customization extends from selecting the exact service needed to specifying the frequency and timing of appointments, which can be set for single or recurring sessions, accommodating the fluctuating demands of modern lifestyles. Additionally, each service provider is presented with an in-depth profile that includes a transparent rating system and reviews from the community, laying the groundwork for a trusted and informed selection process by the users.

Beyond its sleek front-end presentation, HomeConnect+ is engineered with a robust backend architecture utilizing Google's Firebase platform. This integration is pivotal for real-time updates, data persistence, user authentication, and seamless payment processing. Firebase's scalable infrastructure ensures that HomeConnect+ can efficiently handle a growing user base and an

increasing volume of transactions without compromising on performance or security. The use of Firebase also allows for push notifications to keep users informed about their bookings and provider status updates, fostering a dynamic and interactive user experience.

Moreover, the application's backend is set up to collect insightful analytics, which is instrumental for understanding user behavior and preferences, driving the app's continuous improvement and personalization. Future developments also include the implementation of machine learning algorithms to offer service recommendations and smart scheduling options based on user history and preferences. HomeConnect+ stands as a forward-thinking solution in the realm of digital home service platforms, perfectly aligning with the high standards of contemporary mobile app users and showcasing the full potential of combining Flutter's UI capabilities with Firebase's powerful backend services.

1.1. FLUTTER

Flutter, Google's cutting-edge UI toolkit, has been chosen as the cornerstone for developing HomeConnect+, a mobile application prototype that serves as an innovative marketplace for home services. Flutter is renowned for its ability to deliver apps that are indistinguishable from native applications in terms of performance and fidelity, while offering the convenience of a single codebase for cross-platform development.

In the context of HomeConnect+, Flutter's rich set of fully-customizable widgets and its reactive framework have been utilized to create a user interface that is not only intuitive and visually

appealing but also seamlessly adaptive across mobile platforms. The 'hot reload' feature significantly accelerates the app development process, enabling real-time updates and rapid iteration without the need for full rebuilds. This translates to a highly efficient development cycle for HomeConnect+, facilitating the swift incorporation of user feedback into the design.

Flutter's layered architecture allows for the creation of highly responsive and expressively animated user experiences, which is integral to HomeConnect+'s engaging service selection and scheduling process. Furthermore, its integration with Firebase provides a cohesive backend solution, leveraging Firebase's suite of tools such as Firestore for real-time database interactions, Firebase Auth for secure authentication, and Firebase Cloud Messaging for timely notifications.

By harnessing the power of Flutter and Firebase together, HomeConnect+ is set to offer a scalable, reliable, and flexible infrastructure that supports the dynamic nature of service bookings and management. The use of Flutter ensures that HomeConnect+ is not just a prototype but a scalable model ready to evolve with backend integration, embodying the future of multi-platform mobile applications with its sophisticated UI and robust feature set.

1.2. DART

Dart, the programming language behind Flutter, is a pivotal element in the development of HomeConnect+, providing a reactive, concise, and robust framework to build a high-performance mobile marketplace for home services. Dart's object-oriented principles and syntax simplicity play a significant role in accelerating the

development lifecycle and enhancing the maintainability of code. As a language optimized for the modern web and mobile era, Dart enables HomeConnect+ to deliver a smooth and responsive user experience. Its just-in-time (JIT) compilation feature aids in a faster development cycle with the hot reload capability of Flutter, allowing for immediate viewing of changes in the code without losing the app state. This leads to an incredibly efficient development process, vital for iterating on HomeConnect+'s user interface and features based on real-time feedback.

Furthermore, when the app is ready for release, Dart's ahead-of-time (AOT) compilation ensures that HomeConnect+ can be compiled into native code for both Android and iOS platforms, providing performance that is on par with apps written in native languages like Java or Swift. Dart's rich standard library and ecosystem of packages, including seamless Firebase integration, empower HomeConnect+ with a vast array of functionalities, from database operations and authentication flows to complex business logic and service scheduling algorithms.

In the broader scope of HomeConnect+, Dart's ability to transpile into JavaScript for web support illustrates the language's versatility, suggesting potential future expansions of the application beyond mobile to a web platform with a single, unified codebase. Overall, Dart is not just a language choice but a strategic enabler for HomeConnect+, allowing it to leverage the full spectrum of features and benefits that Flutter offers for a cross-platform, high-quality app development experience.

1.3. FIREBASE

Firebase serves as the backbone of the application's backend infrastructure, offering a comprehensive suite of services that are crucial for the app's real-time functionality, scalability, and user engagement. Firebase, a platform developed by Google, is known for its robust set of tools that power the backend of web and mobile applications, making it an exemplary choice for HomeConnect+'s needs.

Firebase provides HomeConnect+ with a real-time database through Firestore, allowing instantaneous data synchronization across user devices. This is instrumental for updating the app with new service listings, booking confirmations, and user reviews without any perceptible delay, ensuring a dynamic and interactive experience. The integration with Firestore is a key component, especially when dealing with the scheduling and management aspects of home services, as it provides users and service providers with up-to-the-minute updates.

Authentication is another cornerstone of HomeConnect+, and Firebase Auth offers a secure, hassle-free authentication system supporting various sign-in methods, including email/password, phone, and social media accounts. This simplifies the process for users to create and access their accounts while maintaining high security standards.

Furthermore, Firebase Cloud Functions allow HomeConnect+ to run backend code in response to events triggered by Firebase features and HTTPS requests. This means the app can efficiently handle complex operations, like payment processing or scheduling

algorithms, without the need for managing server infrastructure.

Firebase Cloud Messaging (FCM) enhances user engagement through timely push notifications about service status updates, reminders, or promotions, keeping the users informed and engaged. Additionally, Firebase's machine learning capabilities can be harnessed to further personalize the user experience by recommending services based on user behavior and preferences.

Analytics provided by Firebase gives HomeConnect+ valuable insights into user interactions and app performance, guiding data-driven decisions to improve services and user experience. This can lead to targeted marketing campaigns, optimized app performance, and enhanced service offerings.

Firebase's multifaceted platform not only streamlines the development process with its powerful and easy-to-use services but also provides a scalable solution that grows with the app. For HomeConnect+, Firebase is not just a tool but a strategic partner that supports the app's vision to deliver a seamless, intuitive, and efficient marketplace for home services.

Incorporating Firebase into HomeConnect+ means not just providing a robust backend solution but also setting the stage for a scalable, feature-rich platform that can grow alongside the user base and the expanding needs of the home services market. Firebase's integration across HomeConnect+'s systems epitomizes the synergy between a strong frontend framework and a powerful backend, a combination that is essential for today's top-tier app offerings.

1.4. PROBLEM STATEMENT

In today's fast-paced world, homeowners and tenants face the challenge of finding reliable and quality home maintenance services. The process of searching for trusted service providers, getting quotes, booking appointments, and making payments is often time-consuming and fragmented, involving multiple communication channels and platforms. Furthermore, service providers, especially independent contractors and small businesses, struggle to reach potential customers due to the lack of a centralized platform that can showcase their services, manage bookings, and handle payments efficiently. In the realm of home management, contemporary homeowners and tenants encounter significant inefficiencies and inconvenience when attempting to secure reliable home maintenance services. This modern-day challenge is twofold, impacting both service consumers and providers within the home service industry.

For users, the primary struggle lies in the discovery and selection of reputable home service providers. The conventional approach often involves sifting through multiple sources, which is time-consuming and offers no guarantee of service quality. Once a potential service is found, users are faced with the cumbersome task of scheduling appointments at times that fit into their busy lives. Additionally, payment methods are frequently limited to cash transactions, which are becoming increasingly inconvenient in an era favoring digital solutions. There is also the issue of effectively managing and tracking past and upcoming service bookings, which can become chaotic without a centralized system.

Service providers, on the other hand, find themselves in a competitive and disjointed marketplace. Their visibility to potential clients is often obscured by the lack of a unified platform that could highlight their offerings and facilitate direct communication with customers. The management of appointments is fraught with the potential for scheduling conflicts, and there's the ever-present challenge of securing timely payments to maintain cash flow. Moreover, building a trusted reputation is a constant uphill battle in an environment where customer feedback is scattered and difficult for new customers to verify.

The application, built on the Flutter framework, will offer an all-in-one solution that simplifies the process of finding, scheduling, and paying for home maintenance services. Users will benefit from a streamlined platform featuring a comprehensive list of vetted service providers, an intuitive booking system with flexible scheduling options, secure in-app payment functionalities, and a personal dashboard to oversee all service interactions.

Simultaneously, the platform aims to empower service providers with tools to increase their visibility, manage bookings more effectively, receive prompt payments, and cultivate a reputation based on transparent customer feedback. Ultimately, the deployment of this application is anticipated to enhance user satisfaction and provider accessibility, resulting in a robust marketplace that can scale and adapt to an evolving service industry. By bridging the gap between consumers and providers, the app aims not only to enhance the efficiency and satisfaction on both ends but also to cultivate a reliable and trusted home services community.

1.5. OBJECTIVES

The primary objective of HomeConnect+ is to create a robust and user-friendly mobile application that simplifies the process of connecting homeowners with professional home service providers. Leveraging the capabilities of Flutter and Firebase, the app aims to provide a seamless and integrated experience for booking, managing, and customizing home-related services. The app's goal is to offer a diverse range of home services, from cleaning to plumbing and electrical work, through an accessible mobile platform and enable users to schedule and customize services with ease, providing flexible options that cater to various needs and preferences. HomeConnect+ presents a trustworthy environment where service providers are vetted and rated, ensuring quality and reliability for the users.

Key Challenges:

i) User Experience Design:

Crafting an intuitive and engaging user interface that maintains consistency across different platforms while meeting the high standards of modern users.

ii) Real-Time Data Management:

Implementing a real-time database solution that can handle concurrent requests and updates efficiently without latency issues.

iii) Authentication and Security:

Ensuring the app provides secure authentication mechanisms to protect user data, especially when handling personal details.

iv) Scalability:

Designing a backend infrastructure that can scale seamlessly as the app grows in popularity and service scope, managing the increased load without performance degradation.

v) User Engagement:

Maintaining high levels of user engagement through timely notifications, personalized recommendations, and a responsive feedback system.

vi) Service Provider Integration:

Creating a system that can manage service provider profiles, services offered, availability schedules, and customer ratings effectively.

vii) Flutter Codebase Maintenance:

Managing the complexity of a Flutter-based codebase to ensure code readability, maintainability, and scalability. Keeping the codebase up-to-date with the latest Flutter releases and best practices to leverage new features and improvements

CHAPTER 2

METHODOLOGY

2.1. INITIAL PROJECT REQUIREMENTS

For the HomeConnect+ project, the initial requirements across various dimensions such as project scope, technologies, functionalities, and packages are suggested in breakdown:

i) Project Scope:

- 1) Develop a mobile application named "HomeConnect+."
- 2) Include a wide range of home services such as cleaning, plumbing, electrical repairs, lawn care, and pest control.
- 3) Define multiple user roles including homeowners (service users), service providers, and administrative staff.
- 4) Address legal and regulatory compliance issues, particularly in relation to data protection and security.

ii) Technologies:

- 1) Flutter: Use Flutter for cross-platform mobile app development, ensuring a consistent user experience on both iOS and Android.
- 2) Dart: Employ the Dart programming language as the primary language for Flutter app development.
- 3) Firebase: Leverage Firebase for backend services including Firestore for real-time database, Firebase Auth for authentication, and Firebase Cloud Functions for server-side logic.

iii) Functionalities:

- 1) User Registration and Login: Secure authentication system with support for multiple methods (email/password, social media accounts, and mobile numbers).
- 2) Service Booking System: A dynamic booking interface where users can schedule, reschedule, or cancel services.
- 3) Customizable Service Options: Allow users to specify details like the size of the area to be serviced (e.g., square footage), special requests, and frequency of service.
- 4) Real-Time Notifications: Push notifications for booking confirmations, service reminders, and promotional offers.
- 5) Ratings and Reviews: Users can rate service providers and write reviews, influencing the provider's visibility and credibility on the platform.
- 6) Provider Profiles: Detailed profiles with qualifications, services offered, availability, and user ratings.

iv) Packages Used:

i) provider:

Version: 6.0.2

Purpose: State management library that allows for efficient data sharing across different screens of the app.

ii) http:

Version: 0.14.0

Purpose: Facilitates network requests to REST APIs, essential for server communications outside of Firebase.

iii) flutter_form_builder:

Version: 7.3.1

Purpose: Simplifies form creation and validation, helping to manage user inputs more effectively.

iv) google_maps_flutter:

Version: 2.1.8)

Purpose: Simplifies form creation and validation, helping to manage user inputs more effectively.

v) firebase_auth:

Version: 3.4.0

Purpose: Handles user authentication processes, supporting various methods like email, password, and social media logins.

vi) cloud_firestore:

Version: 3.1.7

Purpose: Accesses Firestore, Firebase's real-time database, to store and sync data in realtime.

vii) cloud_firestore_web:

Version: 3.10.3

Purpose: Web implementation of Cloud Firestore.

viii) firebase_auth_web:

Version: 5.9.3

Purpose: Web implementation of Firebase Authentication.

ix) firebase_core_web:

Version: 2.11.3

Purpose: Web implementation of Firebase Core.

x) google_sign_in:

Version: 5.2.1

Purpose: Google Sign-In authentication.

xi) toast:

Version: 0.3.0

Purpose: Displaying simple toast notifications.

xii) fluttertoast:

Version: 8.0.7

Purpose: Displaying customizable toast messages.

xiii) image_picker:

Version: 1.0.7

Purpose: Picking images from the gallery or taking photos.

xiii) firebase_crashlytics:

Version: 2.4.2

Purpose: Monitors app stability, reports crashes, and helps in diagnosing issues across user devices.

2.2. PROPOSED APPLICATION ARCHITECTURE

1. Flutter

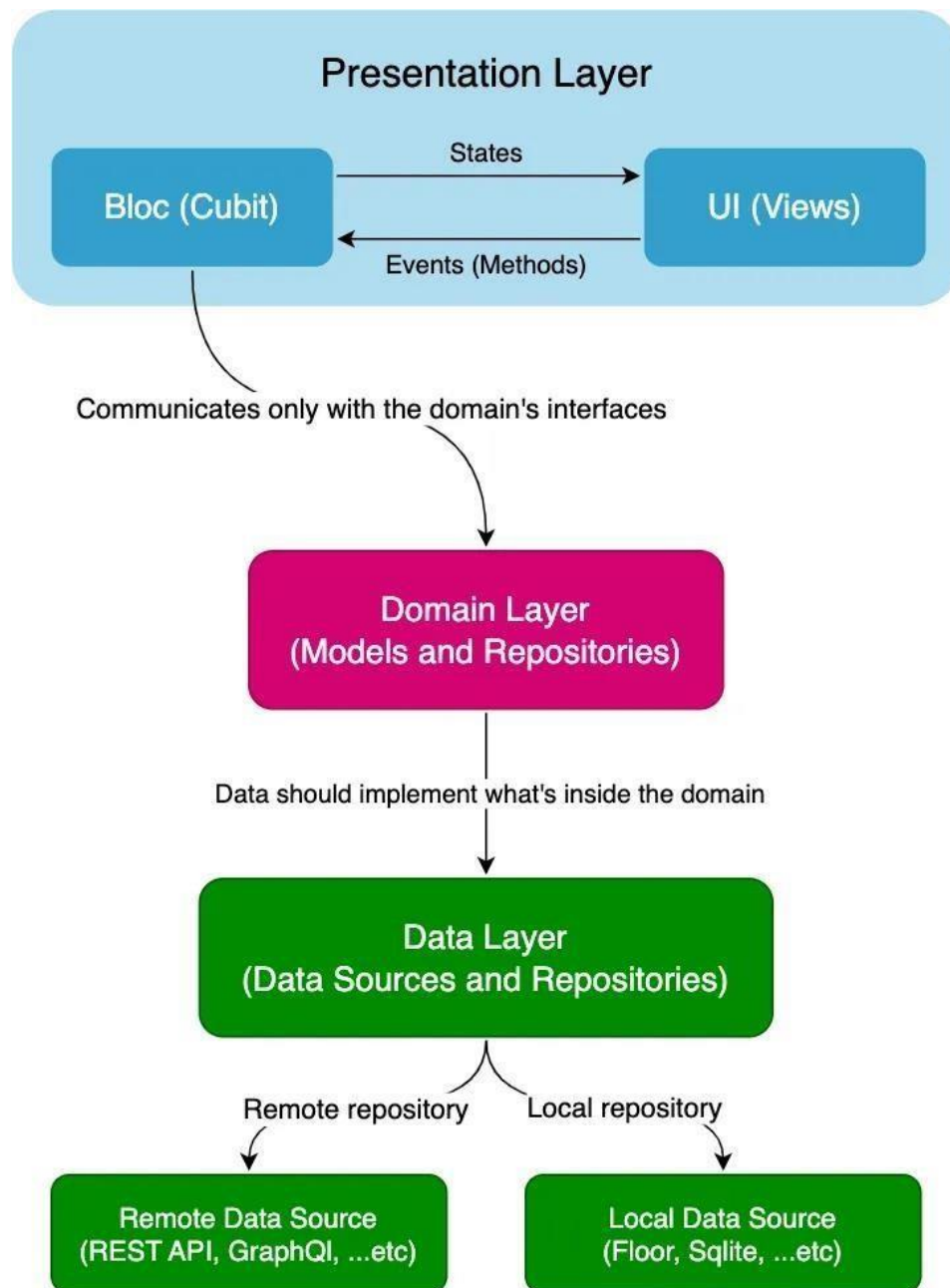


Fig:2.1: Flutter Architecture

The following steps are used to describe Fig:2.1 Flutter Architecture:

1. Presentation Layer:

- i) Flutter UI Framework: Since HomeConnect+ is a Flutter-based mobile application, the presentation layer primarily consists of Flutter widgets, screens, and components.
- ii) Widgets and Screens: Different screens for services selection, task customization, scheduling, provider profiles, and user account management.
- iii) Navigation: Utilize Flutter's navigation system (e.g., Navigator) for seamless screen transitions and user flow.
- iv) UI/UX Design: Implement clean, intuitive design principles for user-friendly interactions and accessibility.
- v) Integration with Firebase: Use Firebase services for user authentication, real-time updates, and data synchronization between the app and backend.

2. Domain Layer:

- i) Business Logic: Contains the core logic of HomeConnect+, including service management, appointment scheduling, user preferences, and provider interactions.
- ii) Service Classes: Define classes for each service category (e.g., cleaning, plumbing, electrical work) to handle service-specific logic.
- iii) Appointment Management: Manage user appointments, scheduling conflicts, recurring appointments, and notifications.
- iv) User Preferences: Store and manage user preferences, such as preferred service providers, custom tasks, and scheduling preferences.

3.Data Layer:

- i) **Firestore Integration:** Firestore serves as the backend for HomeConnect+, providing data storage, authentication, and real-time updates.
- ii) **Firestore:** Store user profiles, service data, provider information, appointment details, and user preferences in Firestore collections and documents.
- iii) **Firestore Authentication:** Implement Firestore Auth for secure user authentication, including email/password login, social login (e.g., Google Sign-In), and user account management.
- iv) **Cloud Functions:** Use Firestore Cloud Functions for server-side logic, such as sending notifications, processing payments (if applicable), and handling background tasks.
- v) **Real-time Updates:** Leverage Firestore's real-time capabilities to synchronize data across devices and provide instant updates for appointments, service availability, and user interactions.

By structuring the HomeConnect+ application architecture in this manner, you can achieve a scalable, maintainable, and efficient solution that integrates seamlessly with Firestore for robust backend functionalities while offering a smooth and intuitive user experience through Flutter's UI framework.

2. Firebase:

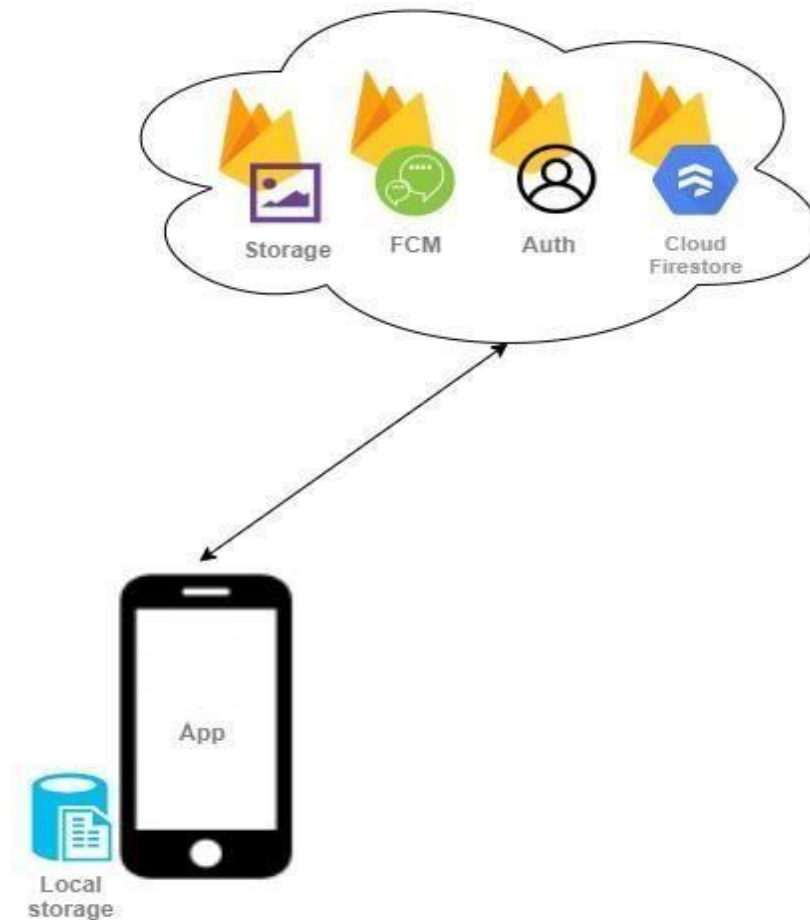


Fig:2.2: Firebase Architecture

The following steps are used to describe Fig:2.2 Firebase Architecture:

Firestore Storage:

Purpose: Firestore Storage is ideal for storing user-generated content, such as profile pictures, task-related images (e.g., before/after cleaning photos), and documents.

Firestore Cloud Messaging (FCM):

Purpose: FCM enables real-time communication and notification

delivery between the server and devices, making it perfect for sending push notifications to users.

Firestore Authentication (Auth):

Purpose: Firebase Auth provides secure user authentication and authorization services, allowing users to sign up, log in, and manage their accounts securely.

Cloud Firestore

Purpose: Cloud Firestore is a flexible, scalable NoSQL database in Firebase, suitable for storing structured data and real-time synchronization across devices.

3. To integrate Firebase with your Flutter app:

Add the necessary Firebase plugins (e.g., `firebase_core`, `firebase_auth`, `cloud_firestore`, etc.) to your `pubspec.yaml`. Initialize Firebase in your app (usually in the `main` function). Use Flutter widgets to build UI that interact with Firebase services.

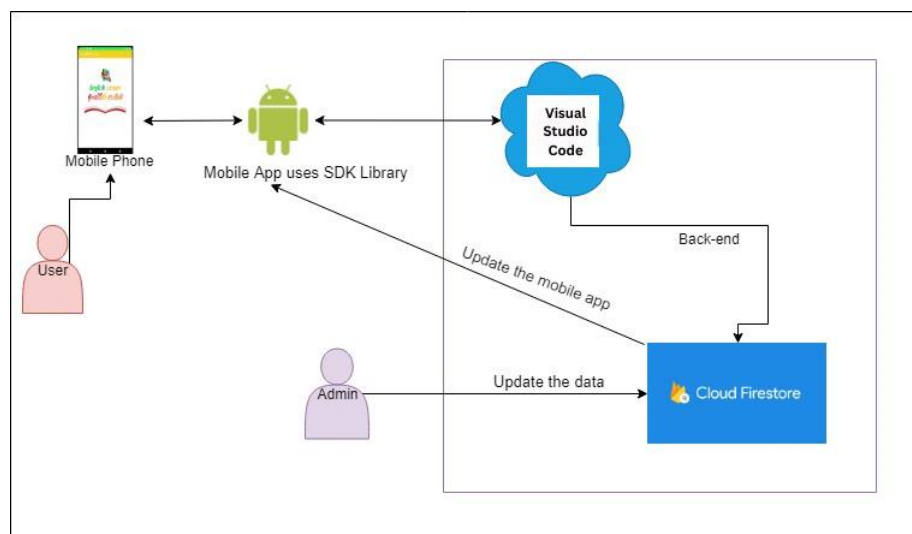


Fig:2.3: Proposed Application Architecture

The following steps are used to describe Fig:2.3 Application Architecture:

User: The process begins with a user who wants to use the HomeConnect+ app.

Login/Sign-up: The user either logs in using their existing credentials or signs up for a new account. This is handled by Firebase Authentication, which is part of the back-end.

User: After successful login or sign-up, the user is back to the main stage.

Select Service: The user selects the service they require from the options available in the HomeConnect+ app.

Enter Details: The user enters the necessary details for the selected service.

Back-end: The back-end, which includes Firebase Authentication and other Firebase services, processes the user's request and service details.

Admin: An admin (or the system itself) manages the service providers' profiles in the back-end.

Service Providers' Profile Dashboard: Based on the user's required service and details, the appropriate service providers' profiles are displayed in the app.

Select a Service Provider: The user can select one service provider from the profiles displayed.

Visual Studio Code: This is the development environment where the code for the HomeConnect+ app is written and managed.

Mobile Phone: This is the device on which the user is interacting with the HomeConnect+ app.

2.3. DATAFLOW DIAGRAM

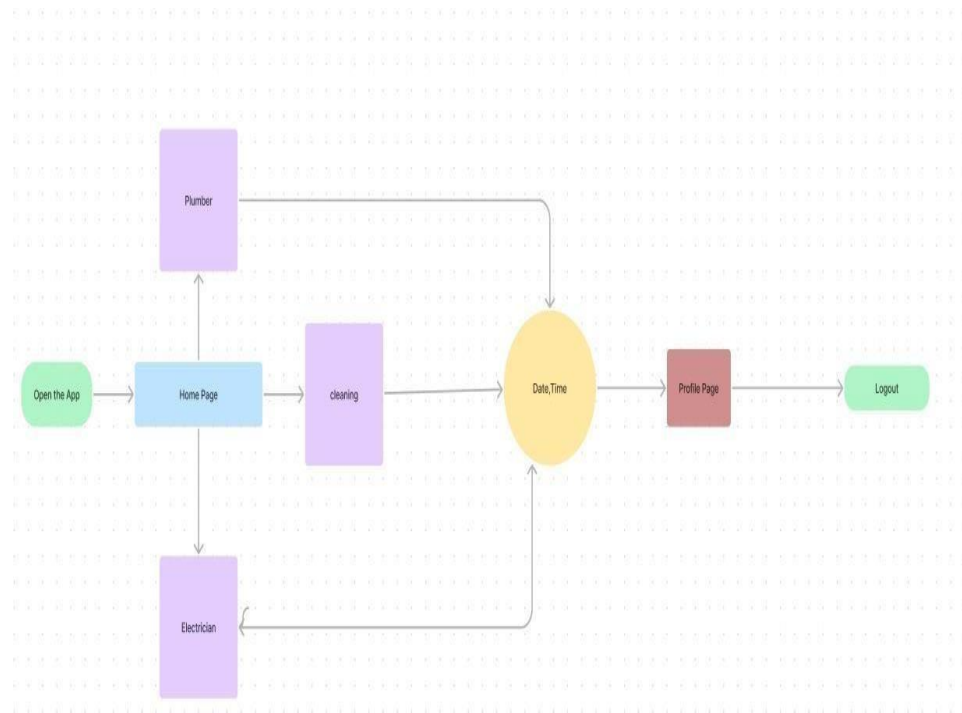


Fig:2.4: Dataflow Diagram

The flow diagram Fig:2.4 depicts the user journey within the HomeConnect+ app, from the initial point of entry to the final interaction. The steps are as followed:

1. Open the App: This is the starting point where the user launches the HomeConnect+ application.
2. Home Page: After opening the app, the user lands on the home page, which is the central hub for navigation within the app.
3. From the Home Page, the user can navigate to one of the service categories:
4. Cleaning: The user selects the cleaning service from the list of available services.

5. Plumber: Alternatively, the user can select the plumbing services option.
6. Electrician: Similarly, the user has the option to choose services provided by an electrician.
7. Date/Time: After selecting a service category, the user is prompted to choose a date and time for the service. This step is crucial as it schedules the appointment based on the user's availability and the service provider's schedule.
8. Profile Page: Following the date and time selection, the user is directed to the profile page of the service provider. This page likely includes details about the service provider, their ratings, reviews, and other relevant information that aids the user in making an informed decision.
9. Logout: After the activities on the profile page are concluded, the user has the option to log out of the application, which would end their session.

2.4. MODULES

The project is divided into four modules and each module has a detailed the algorithmic approach and functionalities to achieve an efficient and user-friendly system. The modules are listed as:

2.4.1. Authentication Module:

Allow new users to create an account by providing necessary information such as name, email, and password and allows existing users to log into the app using their credentials and enable users to recover forgotten passwords through a secure process.

Algorithm:

1. Initialize Interface: Display login and registration options upon app launch.
2. Session Validation: Check for existing sessions using local storage; if valid, bypass login.
3. Input Validation: Ensure all user input meets security standards before submission.
4. User Authentication: Use Firebase Auth to verify credentials and manage sessions.
5. Error Handling: Provide feedback on authentication failures and guide for recovery options.
6. Data Encryption: Securely encrypt sensitive data like passwords before storing or transmitting.

2.4.2. Service Selection Module:

Display a list of services (e.g., cleaning, plumbing) and allows users select specific details (e.g., rooms for cleaning). It provides dynamic Content that is the update service options based on user selections and provider availability.

Algorithm:

1. Data Retrieval: Fetch service categories and details from Firestore in real-time.
2. Dynamic Display: Render services dynamically based on user interactions or preferences.
3. User Input Collection: Store user choices in app state for session persistence.
4. Cost Calculation: Algorithmically calculate costs as users customize services (adding rooms, special requirements).

5. Filtering and Sorting: Implement sorting and filtering based on service type, price, and other user-defined parameters.
6. Update Interface: Refresh the UI dynamically as users adjust their service preferences.

2.4.3. Scheduling and Booking Module:

Allows date and Time Selection using Calendar interface which is used to select the date and time. Additional Services options to add more services to a single booking is available. Booking Confirmation to finalize the booking details and confirm is present in this module

Algorithm:

1. Date Selection: Display an interactive calendar for users to select service dates.
2. Time Slot Validation: Check availability of service providers for selected times, adjusting available options dynamically.
3. Multi-service Management: Manage a cart-like system for multiple services, including addition, deletion, or modification of selections.
4. Booking Finalization: Securely transmit booking details to Firestore, including user ID and comprehensive service information.
5. Conflict Resolution: Implement logic to handle booking conflicts and suggest alternate options.
6. Confirmation Notifications: Send automated confirmation notifications to users and service providers.

2.4.4. Professional Profiles and Ratings Module:

This module shows detailed profiles of service providers and enable users to view and submit ratings and reviews.

The selection algorithm is used to recommend service providers based on ratings and proximity.

Algorithm:

1. Profile Access: Retrieve and display detailed profiles from Firestore based on user or system criteria.
2. Rating System: Calculate and display average ratings from user reviews, updating dynamically with new input.
3. Review Submission: Enable submission of reviews, incorporating input validation and user authentication.
4. Provider Sorting: Use algorithms to sort providers by ratings, distance, and user preferences.
5. Geolocation Integration: Utilize geolocation data to suggest service providers based on proximity to the user's location.
6. Feedback Loop: Incorporate user feedback to refine profile recommendations and enhance service provider performance metrics.

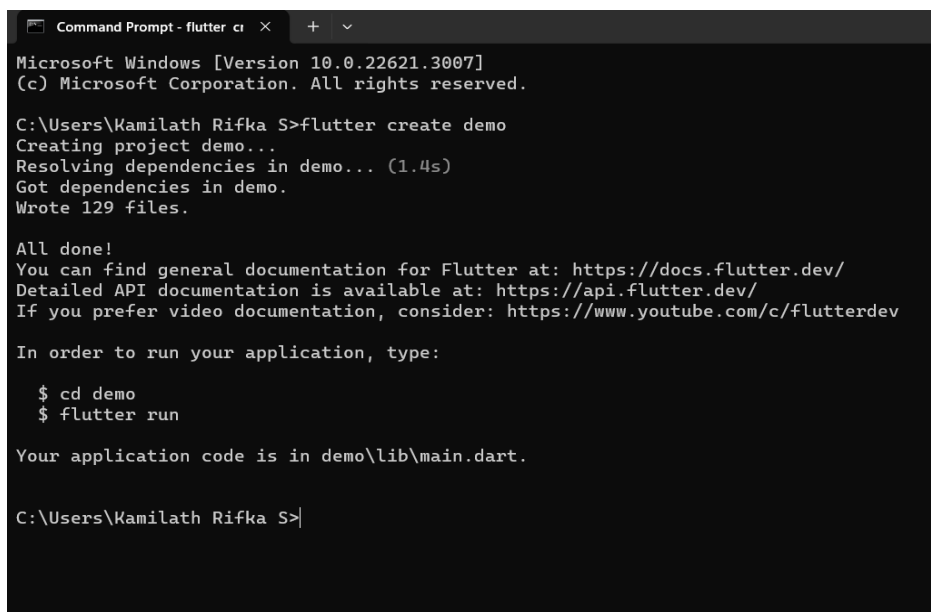
2.5. DEVELOPMENT AND DEPLOYMENT SETUP

1. Development Environment:

Flutter and Dart need to be installed on the developer's machine.

IDEs like Visual Studio Code or Android Studio for coding.

2. Create Project:



```
Command Prompt - flutter ci x + v
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kamilath Rifka S>flutter create demo
Creating project demo...
Resolving dependencies in demo... (1.4s)
Got dependencies in demo.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

$ cd demo
$ flutter run

Your application code is in demo\lib\main.dart.

C:\Users\Kamilath Rifka S>
```

Fig:2.5: Project Creation

The following steps are used to describe Fig:2.5 Project Creation:

1. >>flutter create project name
2. >>cd project name
3. >>flutter run

3. Dependency Management:

Project dependencies specified in the pubspec.yaml file.

Key packages include:

- firebase_core for Firebase integration.
- firebase_auth for user authentication.
- cloud_firestore for real-time data synchronization.
- Other packages like get, toast, image_picker, etc., for additional functionalities.

4. Firebase Configuration:

Firebase project created and configured for the application.

Firebase Hosting used for deploying the Flutter web application.

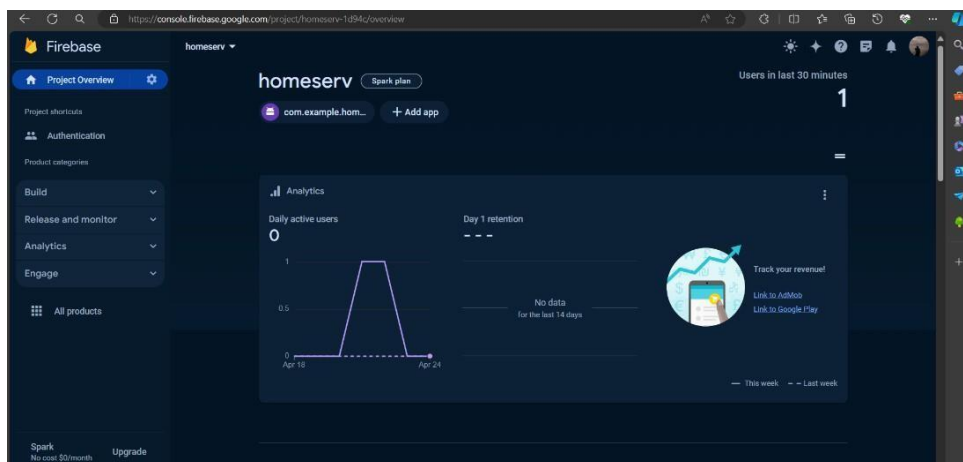


Fig:2.6: Firebase Connection

The following steps are used to describe Fig:2.6 Firebase Connection:

- Firebase project created and configured for the application.
- Firebase Hosting used for deploying the Flutter application.

5. Firebase Authentication:

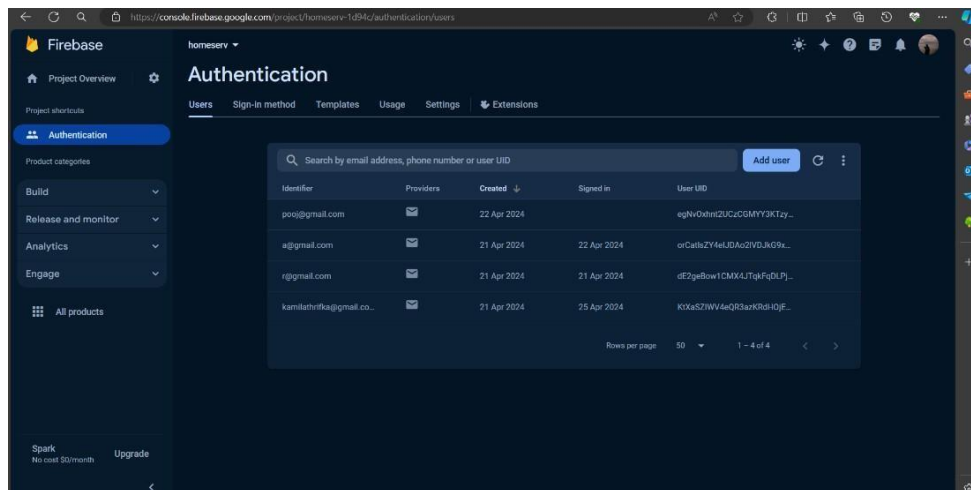


Fig:2.7: Firebase Authentication

Fig:2.7 details the setup of Firebase Authentication for secure user login and identity verification.

6. Development Tools:

Flutter commands used for testing and building the application. Real-time testing on Android and iOS devices.

7. Deployment Process:

Firebase used for hosting the web version of the application. Seamless deployment using Flutter and Firebase commands.

8. Supported Platforms:

Cross-platform support for Android, iOS, and web.

CHAPTER 3

RESULT AND DISCUSSION

DART CODES:

i) `main.dart`: This file is typically the entry point of a Flutter application. It initializes the app, sets up necessary configurations, and runs the main application widget.

ii) `loginPage.dart`: This file likely contains the UI and logic for user login. It includes form fields for entering credentials and a mechanism to authenticate users.

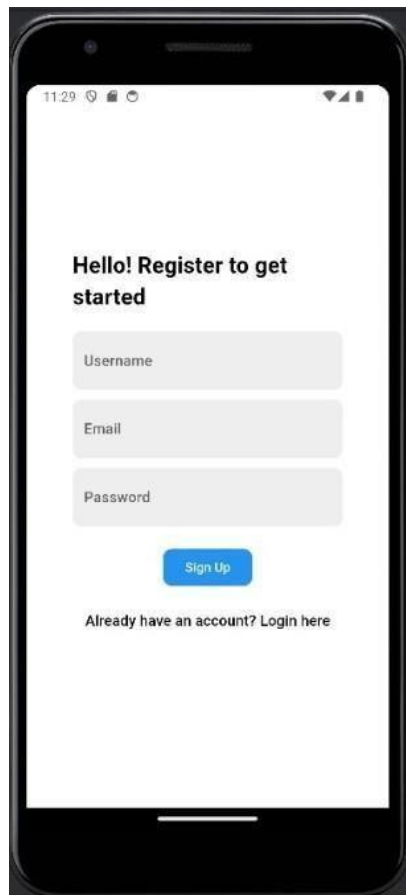


Fig:3.1: Sign up Page Interface

iii) signupPage.dart: Fig 3.1 depicts the sign-up page where new users can sign in. Similar to the login file, signupPage.dart is likely responsible for handling user registration. It includes form fields for users to create accounts.

iv) start.dart: This file represents the main content of your app after a successful login or signup. It contains the list of services available with various features.

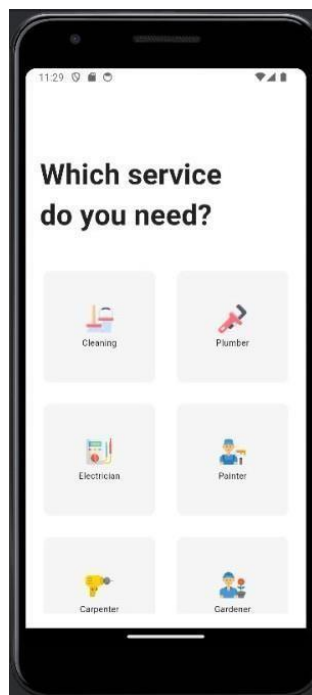


Fig:3.2: List of service provided Interface

v) select_service.dart: Fig:3.2 depicts the interface where the available services are displayed. This file is responsible for selecting the required service the user wants. Based on the service selection the other aspects of the service is proceeded.

vi) cleaning.dart: This file represent a page the service would be selected. Here when the service cleaning is selected the associated questions like how many rooms to be cleaned etc. will be displayed.

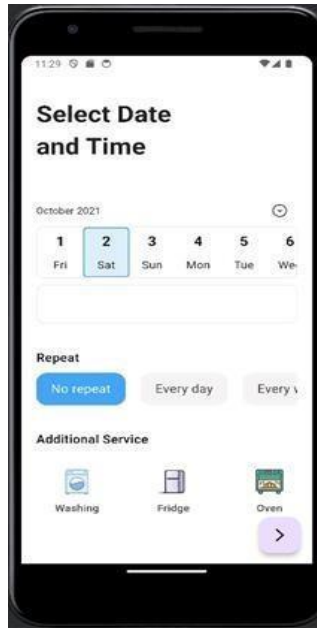


Fig:3.3: Date and Time Selecting Interface

vii) date_time.dart: Fig:3.3 depicts the page for selecting the date and time. This file defines the page where the user can select the slot required for the service. Slots for the date and time will be included in this page

viii) home.dart: This file represents profiles of the service personals who perform specify task along with their profile from which the user can select

FIREBASE:

i) Create a Firebase Project:

1. Go to the Firebase Console.
2. Click on "Add Project" and follow the setup instructions.

ii) Configure Flutter Project:

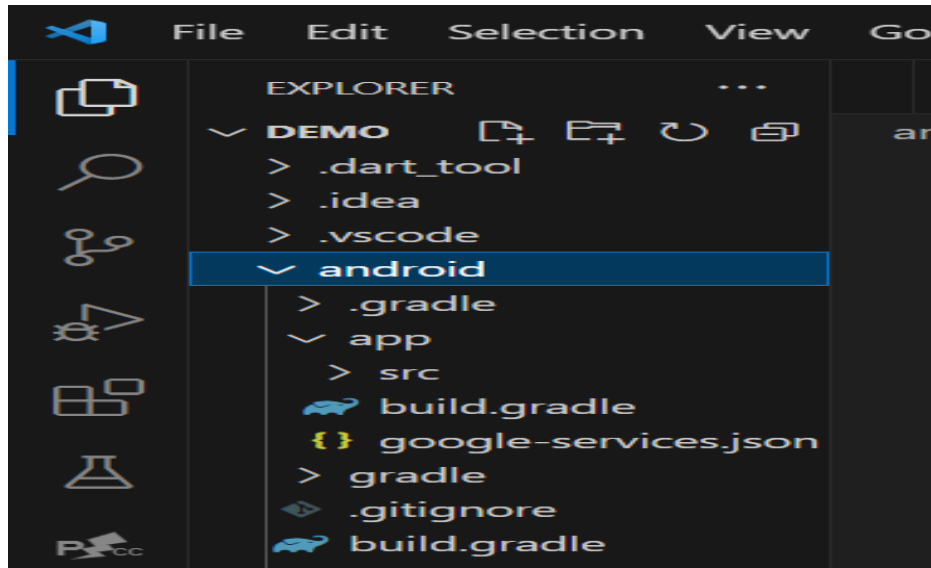


Fig:3.4: Firebase Configurations

The following steps are used to describe Fig:3.5 Firebase Configurations

1. Add the Firebase configuration files to your Flutter project.
2. Follow the instructions provided in the Firebase Console to add google-services.json for Android and GoogleService-Info.plist for iOS.
3. These files contain configuration details needed for your app to connect to Firebase services.

iii) Add Firebase Dependencies:

Open the pubspec.yaml file and add the following dependencies:

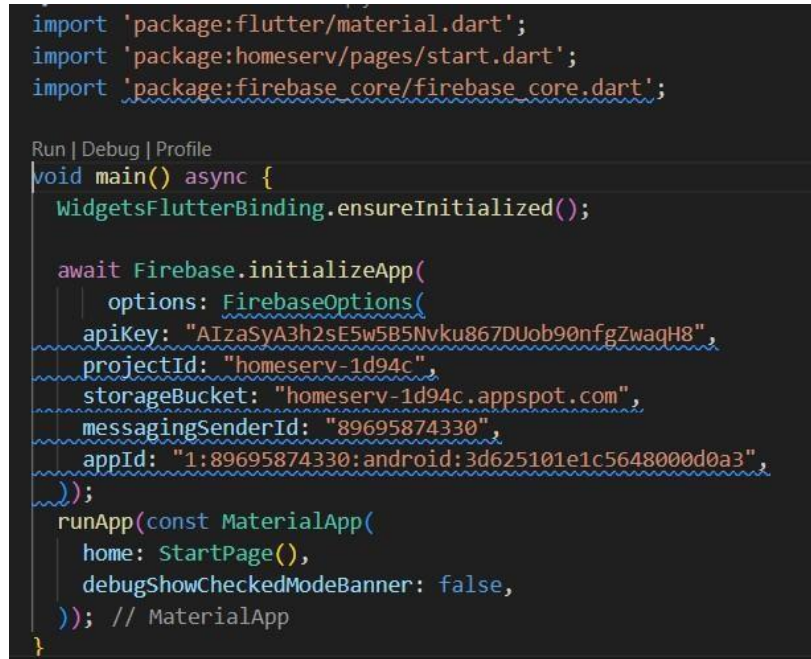
yaml code:

dependencies:

firebase_core: ^2.25.3

firebase_auth: ^4.17.5

iv) Initialize Firebase:



```
import 'package:flutter/material.dart';
import 'package:homeserv/pages/start.dart';
import 'package:firebase_core/firebase_core.dart';

Run | Debug | Profile
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(
    options: FirebaseOptions(
      apiKey: "AIzaSyA3h2sE5w5B5Nvku867DUob90nfgZwaqH8",
      projectId: "homeserv-1d94c",
      storageBucket: "homeserv-1d94c.appspot.com",
      messagingSenderId: "89695874330",
      appId: "1:89695874330:android:3d625101e1c5648000d0a3",
    ));
  runApp(const MaterialApp(
    home: StartPage(),
    debugShowCheckedModeBanner: false,
  )); // MaterialApp
}
```

Fig:3.5: Initializing Firebase

Fig:3.5 highlights the initialization of Firebase within your main.dart file or wherever you initialize your Flutter app. It emphasizes the importance of initializing Firebase to enable functionalities like user authentication. Firebase offers the `firebase_auth` package for implementing user authentication, allowing you to sign in users using methods such as email/password authentication, Google Sign-In.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1. CONCLUSION

In conclusion, HomeConnect+ showcases the potential of mobile technology to revolutionize the booking and management of home services. With its user-friendly design powered by Flutter and Firebase, it offers a seamless interface for selecting and scheduling a variety of home services. Although it currently lacks a payment feature, the app's solid foundation and modular architecture hint at a promising future with ample scope for incorporating additional functionalities like secure in-app transactions and personalized user recommendations.

Moving forward, the potential for expansion and enhancement is vast. The incorporation of a payment system, expansion of service categories, and additional personalization of user experiences through machine learning and predictive analytics represent future milestones. HomeConnect+'s modular design promises straightforward scalability, ensuring that as the user base grows, the app will continue to deliver exceptional performance and reliability.

As it stands, HomeConnect+ exemplifies a significant step forward in connecting homeowners with service providers efficiently and securely. Its current capabilities and envisioned enhancements position it to be an indispensable tool for managing home maintenance tasks with convenience and confidence.

4.2. FUTURE WORK

1.Enhanced User Profiles:

The future work for HomeConnect+ should focus on both enhancing current functionalities and adding new features to expand its scope and improve the user experience. Here are some recommendations for future developments:

In-App Payment Integration: Introduce secure in-app payment systems to allow users to pay for services directly through the app. Offer multiple payment options, including credit cards, digital wallets, and bank transfers.

Advanced Scheduling Algorithms: Develop smarter scheduling algorithms that can suggest the best times for services based on user preferences and provider availability.

Include machine learning to optimize scheduling over time based on user behavior.

Service Provider Expansion: Broaden the range of services offered, potentially including specialized home improvement tasks. Implement a service provider onboarding module to streamline the process of adding new professionals.

User Experience Personalization: Utilize user data and machine learning to provide personalized service recommendations.

Offer customizable profiles where users can set preferences for service types, communication, and scheduling.

Enhanced Rating and Review System: Implement a more comprehensive system for users to rate and review services with detailed feedback.

Use sentiment analysis on reviews to gain better insights into service quality and customer satisfaction.

Real-Time Chat Functionality: Create a communication channel within the app for users and service providers to discuss service details in real-time.

Marketing and Promotional Tools: Integrate marketing features such as promo codes, discounts for recurring bookings, and referral bonuses.

Geolocation Services: Add geolocation features to match users with nearby service providers for quicker and more efficient service delivery.

Implement map integration for users to track the arrival of service providers.

Analytics Dashboard: Develop a dashboard for service providers and administrators to track usage patterns, earnings, and user feedback.

Provide analytics to users to track their service history and spending.

Accessibility Enhancements: Ensure that the app meets accessibility standards to cater to users with disabilities.

REFERENCES

1. Journals:

[1] P. Tyagi, “Building layouts,” Pragmatic Flutter, pp. 87–126, Jun. 2021. doi:10.1201/9781003104636-7

[2] P. Tyagi, “Setting up environment,” Pragmatic Flutter, pp. 23–34, Jun. 2021. doi:10.1201/9781003104636-3

[3] R. Payne, “Using firebase with flutter,” Beginning App Development with Flutter, pp. 255–285, 2019. doi:10.1007/978-1-4842-5181-2_12

2. Books:

[1] A. K. S., Mastering Firebase for Android Development: Build Real-Time, Scalable, and Cloud-Enabled Android Apps with Firebase. Birmingham, UK: Packt Publishing, Ltd., 2018.

[2] A. Miola, Flutter Complete Reference 2.0: The Ultimate Reference for Dart and Flutter. Independently published, 2023.

[3] J. H. Iversen and M. A. Eierman, Learning Mobile App Development: A Hands-on Guide to Building Apps with IOS and Android. Upper Saddle River, New Jersey: Addison-Wesley, 2014.

3. Internet Articles:

[1] D. Camp, "Best flutter blog articles from around the web," Flutter articles, <https://www.devglan.com/programming/flutterarticle> feeds (accessed Apr. 25, 2024).

[2] Google, "Firebase documentation," Flutter Documentations, <https://firebase.google.com/docs> (accessed Apr. 25, 2024).

[3] W. Wilson, "The ultimate list of flutter resources (over 200!)," Flutter Resources, <https://wilsonwilson.dev/articles/flutter> resources (accessed Feb. 20, 2024).

APPENDICES

A.SAMPLE CODE

i) Main Code:

```
import 'package:flutter/material.dart';
import 'package:homeserv/pages/start.dart';
import 'package:firebase_core/firebase_core.dart';

Run | Debug | Profile
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

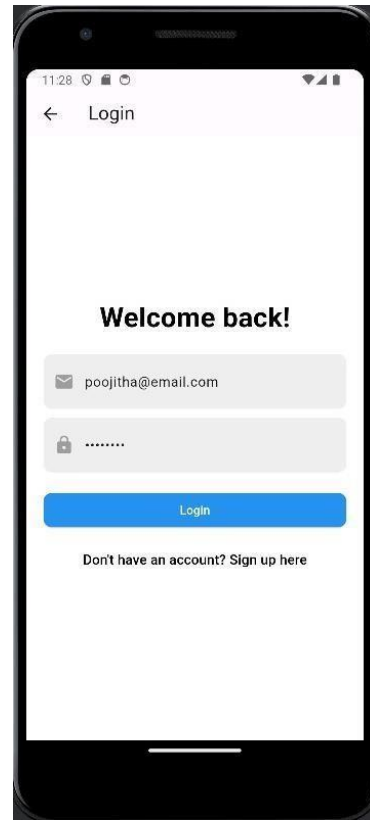
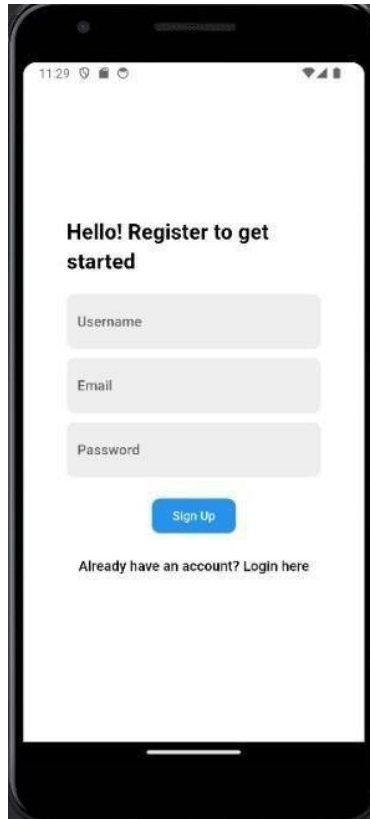
  await Firebase.initializeApp(
    options: FirebaseOptions(
      apiKey: "AIzaSyA3h2sE5w5B5Nvku867DUob90nfgZwaqH8",
      projectId: "homeserv-1d94c",
      storageBucket: "homeserv-1d94c.appspot.com",
      messagingSenderId: "89695874330",
      appId: "1:89695874330:android:3d625101e1c5648000d0a3",
    ));
  runApp(const MaterialApp(
    home: StartPage(),
    debugShowCheckedModeBanner: false,
  )); // MaterialApp
}
```

ii) pubspec.yaml

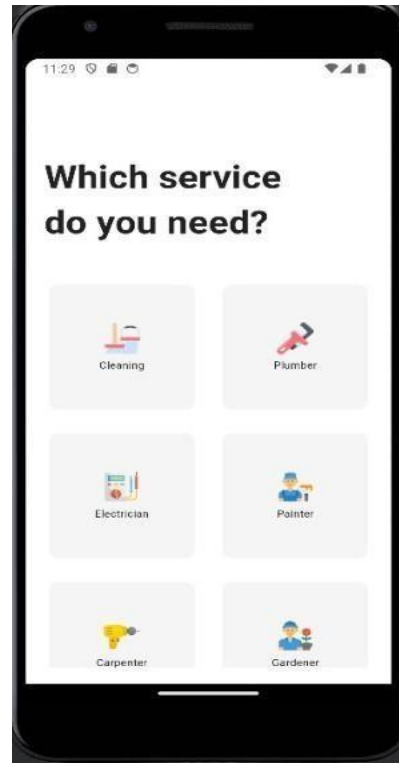
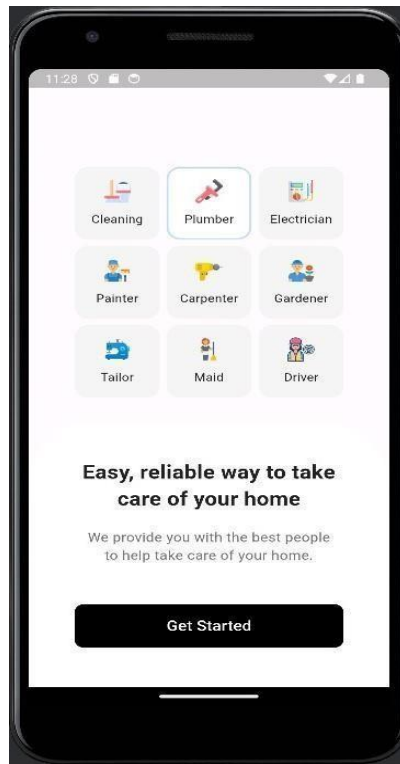
```
pubspec.yaml
  Click here to ask Blackbox to help you code faster
1  name: homeserv
2  description: "A new Flutter project."
3  # The following line prevents the package from being accidentally published to
4  # pub.dev using `flutter pub publish`. This is preferred for private packages.
5  publish_to: 'none' # Remove this line if you wish to publish to pub.dev
6
7
8  version: 1.0.0+1
9
10 environment:
11   sdk: '>=3.2.6 <4.0.0'
12
13
14 dependencies:
15   flutter:
16     sdk: flutter
17
18
19   # The following adds the Cupertino Icons font to your application.
20   # Use with the CupertinoIcons class for iOS style icons.
21
22   cupertino_icons: ^1.0.3
23   scrollable_positioned_list: ^0.3.8
24   simple_animations: 5.0.2
25   firebase_auth: ^4.17.8
26   cloud_firestore: ^4.15.3
27   cloud_firestore_web: ^3.10.3
28   firebase_auth_web: ^5.9.3
29   firebase_core_web: ^2.11.3
30   google_sign_in: ^6.2.1
31   toast: ^0.3.0
32   fluttertoast: ^8.0.7
33   image_picker: ^1.0.7
34
35   permission_handler: ^11.2.0
36   provider: ^6.1.1
37   flutter_rating_bar: ^4.0.0
38
39
```

B. SCREEN SHOTS

i) Login page And Signup Page



ii) Homepage & Select Service Page



iii) Select time and dashboard

