

NEIGHBOURLY INTERDISCIPLINARY PROJECT

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

NITHISH A V (41120132)



**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)

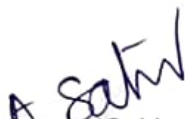
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

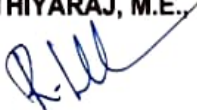
DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **NITHISH A V (41120132)** who carried out the project entitled **"NEIGHBOURLY"** under my supervision from **January 2024 to May 2024**.


Internal Guide

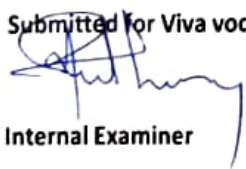
Dr.A. SATHIYARAJ, M.E., Ph.D.,



Head of the Department

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

Submitted for Viva voce Examination held on 08/05/2024


Internal Examiner


External Examiner

DECLARATION

I **NITHISH A V** hereby declare that the Project Report entitled "**NEIGHBOURLY**" done by me under the guidance of **Dr.A. SATHIYARAJ, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in **Information Technology**.

DATE: 08/05/2024

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., Ph.D,** 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 **Nithish A V (Reg No: 41120132)** 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, **Nithish A V** has closely worked as a part of the **Flutter** by carrying out the project entitled "**NEIGHBOURLY**".

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

Best Regards,

Ravikumar S
CEO & Managing Director
Hackwit Technologies Pvt Ltd



CERTIFICATE OF COMPLETION



ABSTRACT

In an era where e-commerce platforms dominate retail, Neighbourly emerges as a pioneering mobile application designed to revitalize and empower local economies by bridging the gap between local sellers and buyers through technology. Developed using Flutter on Android Studio with Dart Language, Neighbourly leverages cutting-edge, location-based features to offer a unique marketplace that promotes the convenience of online shopping while supporting the local business ecosystem. Neighbourly is a transformative e-commerce platform designed to strengthen local economies through technology. Created with Flutter, Dart Language and Firebase backend on Android Studio, this innovative app redefines local shopping by enabling direct connections between local sellers and buyers. By utilizing location-based functionalities, Neighbourly makes it possible for users to discover and purchase products from sellers in their immediate vicinity, promoting a sustainable and community-oriented shopping experience. The app's unique dual registration system caters to both sellers and buyers, simplifying the digital marketplace for local businesses and providing consumers with fast, cost-effective, and environmentally friendly access to goods. Neighbourly not only champions the convenience of online shopping but also actively contributes to the growth of local markets, supports small businesses, and reduces the environmental impact of traditional e-commerce logistics. This platform represents a leap forward in how we think about buying and selling locally, offering a seamless, efficient, and responsible alternative to conventional online shopping.

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	4
	1.4 Problem Statement	6
	1.5 Objectives	7
2	METHODOLOGY	10
	2.1 Initial Project Requirements	10
	2.2 Proposed Application Architecture	14
	2.3 Data Flow Diagram	18
	2.4 Modules	21
	2.4.1 User Authentication	21
	2.4.2 Dual Registration System	22
	2.4.3 Location Enhanced shopping	24
	2.4.4 Dynamic Homepage	26

	2.5	Development and Deployment Setup	27
3		RESULTS AND DISCUSSION	31
4		CONCLUSION AND FUTURE WORK	36
	4.1	Conclusion	36
	4.2	Future Work	37
		REFERENCES	39
		APPENDICES	41
	A.	Sample Code	41
	B.	Screen Shots	43

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	Proposed Application Architecture	14
2.2	Firebase Architecture	16
2.3	Data Flow Diagram	18
2.4	Create New Project	27
2.5	SDK Path Integration	28
2.6	Project Creation	28
2.7	Firebase Connection	29
2.8	Firebase Authentication	30
3.1	Sign In Screen	31
3.2	Sign-up Screen	32
3.3	Shopping Screen	33
3.4	Firebase Configurations	34
3.5	Initialize Firebase	35

CHAPTER 1

INTRODUCTION

In an age dominated by e-commerce platforms, the emergence of Neighbourly marks a significant paradigm shift in the realm of online shopping. With a commitment to revitalizing local economies and fostering community connections, Neighbourly stands as a beacon of innovation in the digital marketplace. Developed using cutting-edge technologies such as Flutter, Dart Language, and Firebase backend on Android Studio, Neighbourly epitomizes the convergence of technology and social responsibility.

As the world becomes increasingly interconnected, the need for localized solutions that empower communities while offering the convenience of online shopping has never been more pressing. Neighbourly rises to this challenge by seamlessly bridging the gap between local sellers and buyers through its intuitive platform. By leveraging location-based functionalities, Neighbourly enables users to discover and engage with nearby businesses, thereby revitalizing the local economy and nurturing a sense of belonging within communities.

Moreover, Neighbourly's innovative dual registration system caters to the diverse needs of both sellers and buyers, streamlining the online shopping experience and empowering small businesses to thrive in the digital age. Beyond its practical utility, Neighbourly embodies a broader ethos of sustainability and environmental stewardship, offering a responsible alternative to traditional e-commerce practices.

1.1 FLUTTER

At the nucleus of Neighbourly's technical architecture resides Flutter, a cross-platform development framework engineered by Google. Designed to facilitate the creation of high-performance, visually appealing applications across multiple platforms, Flutter has emerged as a leading choice for developers seeking to streamline the development process and deliver a consistent user experience. Rooted in the Dart programming language, Flutter offers a comprehensive suite of tools and libraries that empower developers to craft rich, interactive interfaces with remarkable ease. Through its innovative "hot reload" feature, Flutter enables developers to iteratively refine the application codebase in real-time, fostering rapid prototyping and facilitating seamless collaboration among development teams.

Moreover, Flutter's declarative UI paradigm and extensive widget catalog afford developers unparalleled flexibility in designing and customizing the user interface of Neighbourly. Leveraging Flutter's widget-based architecture, Neighbourly can seamlessly adapt its interface to accommodate various screen sizes and form factors, ensuring a cohesive user experience across a diverse array of devices.

Beyond its technical capabilities, Flutter embodies a commitment to performance optimization and resource efficiency, enabling Neighbourly to deliver fluid animations, smooth transitions, and snappy responsiveness to user interactions. By leveraging Flutter's native compilation capabilities, Neighbourly achieves near-native performance across Android and iOS platforms, mitigating the need for platform-specific optimizations and reducing development overhead.

Dart, developed by Google, stands as the driving force behind Neighbourly's innovative approach to e-commerce. This client-optimized programming language is meticulously crafted to ensure fast and efficient applications across various platforms, aligning seamlessly with Neighbourly's commitment to community integration and localized commerce.

Key Features of Dart in Neighbourly:

Optimized for UI Development: Dart's reactive programming model streamlines the development of visually appealing and highly interactive user interfaces within Neighbourly. Through its intuitive features, Dart simplifies the process of crafting engaging experiences, enhancing user satisfaction and retention.

Cross-Platform Performance: Neighbourly benefits from Dart's ability to compile code into machine language for mobile, desktop, and web platforms. This cross-platform compatibility ensures that Neighbourly maintains consistent performance and functionality across diverse devices, maximizing accessibility and user engagement.

Enhanced Development Productivity: Dart's robust set of core libraries and development tools accelerate Neighbourly's development lifecycle. With features like hot reload in Flutter, developers can iterate on Neighbourly's features rapidly, resulting in quicker turnaround times and a more responsive user experience.

Dart in Neighbourly's Ecosystem:

Mobile and Web Development: Dart, coupled with Flutter, drives Neighbourly's mobile and web development efforts. By leveraging Flutter's capabilities for natively compiled applications, Neighbourly provides a seamless shopping experience for users on iOS, Android, and web platforms alike.

1.3 FIREBASE

Firebase, a comprehensive mobile and web application development platform developed by Google, serves as the robust backend solution driving Neighbourly's innovative e-commerce ecosystem. Specifically, Firebase authentication services play a pivotal role in ensuring secure login and signup processes for Neighbourly users, thereby safeguarding in the e-commerce landscape their data and enhancing their overall experience on the platform.

Firebase Authentication provides a secure and straightforward user authentication process. This feature to ensure that only authorized users can access the platform. Multi-factor authentication and identity verification contribute to a robust security layer.

Firebase Cloud Firestore supports real-time data synchronization, allowing seamless updates across multiple devices. Firebase offers serverless cloud functions that enable the execution of backend logic without managing servers.

Key Features of Firebase Authentication in Neighbourly:

Secure Signup and Login Processes: Neighbourly leverages Firebase authentication services to enable users to sign up securely and log in to their accounts with confidence. Through Firebase's secure mechanisms, user credentials are encrypted and transmitted safely, protecting sensitive information from unauthorized access or interception.

Data Storage and Retrieval: Upon successful signup, user data is securely stored in Firebase's cloud database, ensuring that Neighbourly maintains accurate records of user profiles and preferences. When users log in, Firebase authentication verifies their credentials against the stored data, granting access only to authorized users with valid credentials.

Integration with Neighbourly's Ecosystem:

User Registration: When users sign up for Neighbourly, Firebase authentication securely stores their registration data, including usernames, email addresses, and encrypted passwords. This ensures that only authenticated users with valid credentials can access Neighbourly's features and functionalities.

Login Verification: Firebase authentication validates user credentials during the login process, verifying that the provided username and password match the records stored in the Firebase database. By enforcing strict authentication protocols, Neighbourly ensures that only authorized users gain access to their accounts and the platform's resources.

1.4 PROBLEM STATEMENT

In the rapidly evolving digital age, the e-commerce landscape has become dominated by large, centralized platforms that often prioritize efficiency and scale over local economic growth and environmental sustainability. This dominance presents several challenges:

Diminished Local Economies: Small, local businesses find it increasingly difficult to compete with large e-commerce giants. The complexity and costs associated with listing products on major platforms can be prohibitive, leading to reduced visibility and sales for these local sellers.

Environmental Impact: The current e-commerce model relies heavily on longdistance shipping, contributing significantly to carbon emissions and environmental degradation. Products are often shipped across vast distances, even when similar items are available locally, resulting in unnecessary environmental strain.

Loss of Community Connection: Traditional e-commerce platforms do not support the local community's economy in a meaningful way. There is a growing disconnect between consumers and local sellers, weakening the fabric of local communities and economies.

Inefficiencies in Logistics: Long-distance shipping not only impacts the environment but also leads to longer delivery times and higher costs for consumers. The reliance on centralized warehouses further exacerbates these issues, making it challenging to offer fast and cost-effective shipping for all products.

One-Size-Fits-All Approach: Existing platforms often fail to address the unique needs and preferences of local markets. They lack the flexibility to provide a personalized shopping experience that caters to the tastes and requirements of local communities.

Neighbourly seeks to address these challenges by creating a decentralized, community-focused e-commerce platform that empowers local businesses, reduces the environmental impact of shipping, and fosters a stronger connection between consumers and their local economies. By leveraging location-based technology to highlight local products and enabling both sellers and buyers to register and transact within the same app, Neighbourly aims to promote a sustainable, efficient, and community-oriented shopping experience that benefits everyone.

1.5 OBJECTIVES

Neighbourly's objectives are strategically designed to address the identified challenges in the e-commerce landscape and fulfill its mission of fostering community integration and supporting local economies. The objectives encompass various facets of the platform's functionality and user experience, aiming to deliver tangible benefits to both consumers and local businesses. The objectives include:

Enhance Accessibility to Local Goods and Services: Neighbourly aims to provide users with easy access to a diverse range of products and services available within their neighborhoods. By leveraging location-based features and intuitive search functionalities, Neighbourly ensures that users can discover and engage with local businesses effortlessly, thereby fostering community connection and promoting economic vitality.

Personalize the User Experience: Neighbourly endeavors to create a personalized shopping experience that resonates with the unique preferences and interests of each user. Through tailored recommendations, localized promotions, and interactive features, Neighbourly seeks to enhance user satisfaction and engagement, ensuring that every interaction with the platform is relevant, meaningful, and memorable.

Facilitate Seamless Transactions: Neighbourly is committed to streamlining the buying and selling process for both consumers and local businesses. By offering secure payment options, transparent transactional processes, and reliable order fulfillment services, Neighbourly aims to instill confidence and trust in its platform, fostering a seamless and frictionless shopping experience for all parties involved.

Empower Local Sellers: Neighbourly recognizes the importance of empowering local businesses and independent sellers to thrive in the digital marketplace. Through user-friendly registration processes, robust seller tools, and promotional opportunities, Neighbourly provides local sellers with the resources and support they need to succeed, enabling them to reach a broader audience and grow their businesses sustainably.

Cultivate Community Engagement: Neighbourly seeks to cultivate a sense of community and belonging among its users, fostering meaningful connections and interactions within local neighborhoods. Through social features, community forums, and collaborative initiatives, Neighbourly encourages users to engage with one another, share experiences, and support local businesses, thereby enriching the overall community fabric.

By aligning its objectives with its overarching mission, Neighbourly aims to redefine the e-commerce landscape, championing the principles of inclusivity, sustainability, and community empowerment. Through strategic execution and continuous innovation, Neighbourly endeavors to create lasting value for its users, local businesses, and the communities they serve.

CHAPTER 2

METHODOLOGY

2.1. INITIAL PROJECT REQUIREMENTS

Project Scope:

The project scope delineates the boundaries and objectives of Neighbourly's development effort, encompassing the features, functionalities, and deliverables that constitute the platform's core offering. It outlines the specific areas of focus and defines the project's overarching goals and objectives.

Functionalities:

User Authentication and Profile Management:

Implement secure authentication mechanisms for user registration, login, and profile management.

Enable users to customize their profiles, manage preferences, and view order history.

Location-Based Access and Search Functionality:

Integrate location services to provide users with access to local businesses and products based on their proximity.

Develop search functionalities with filters for category, distance, and ratings to facilitate efficient product discovery.

Seller Registration and Storefront Setup:

Develop a seamless registration process for sellers to onboard onto the platform and set up their storefronts.

Offer customization options for sellers to showcase their brand, logo, and product offerings effectively.

Technologies:

Frontend Development:

Utilize Flutter framework with Dart language for cross-platform application development. The Neighbourly app is built using Flutter, a powerful framework developed by Google, which allows for the creation of native interfaces across iOS, Android, and web platforms. Dart, the language used in Flutter development, ensures efficient and expressive code that seamlessly integrates with the Flutter framework.

Implement responsive design principles for optimal user experience across devices. With Flutter's built-in support for responsive design, Neighbourly ensures that its interface adapts fluidly to various screen sizes and orientations. This ensures a consistent and user-friendly experience for all users, regardless of the device they are using.

Backend Development:

Leverage Firebase backend services for user authentication, real-time database, and cloud functions. While the majority of the backend development is handled within the Flutter framework, Firebase is specifically utilized for user authentication purposes. Firebase Authentication provides a secure and reliable solution for managing user accounts, allowing registered users to log in securely using their email and password credentials.

Integrate Firebase Firestore for data storage and retrieval, ensuring scalability and performance. Although Firebase Firestore is primarily used for user authentication in Neighbourly, its scalability and real-time data synchronization capabilities make it an ideal choice for managing user credentials securely. Additionally, Firestore's seamless integration with Flutter ensures efficient data retrieval and storage, enhancing the overall performance and reliability of the application.

Packages Used:

i) get:

Version: 4.6.6

Purpose: State management and navigation framework.

ii) firebase_core:

Version: 2.25.3

Purpose: Firebase initialization and core functionality.

iii) firebase_auth:

Version: 4.17.3

Purpose: User authentication with Firebase.

iv) cloud_firestore:

Version: 4.15.3

Purpose: Interaction with Cloud Firestore for real-time data.

v) cloud_firestore_web:

Version: 3.10.3

Purpose: Web implementation of Cloud Firestore.

vi) firebase_auth_web:

Version: 5.9.3

Purpose: Web implementation of Firebase Authentication.

vii) firebase_core_web:

Version: 2.11.3

Purpose: Web implementation of Firebase Core.

viii) google_sign_in:

Version: 5.2.1

Purpose: Google Sign-In authentication.

ix) toast:

Version: 0.3.0

Purpose: Displaying simple toast notifications.

x) fluttertoast:

Version: 8.0.7

Purpose: Displaying customizable toast messages.

xi) image_picker:

Version: 1.0.7

Purpose: Picking images from the gallery or taking photos.

2.2.PROPOSED APPLICATION ARCHITECTURE

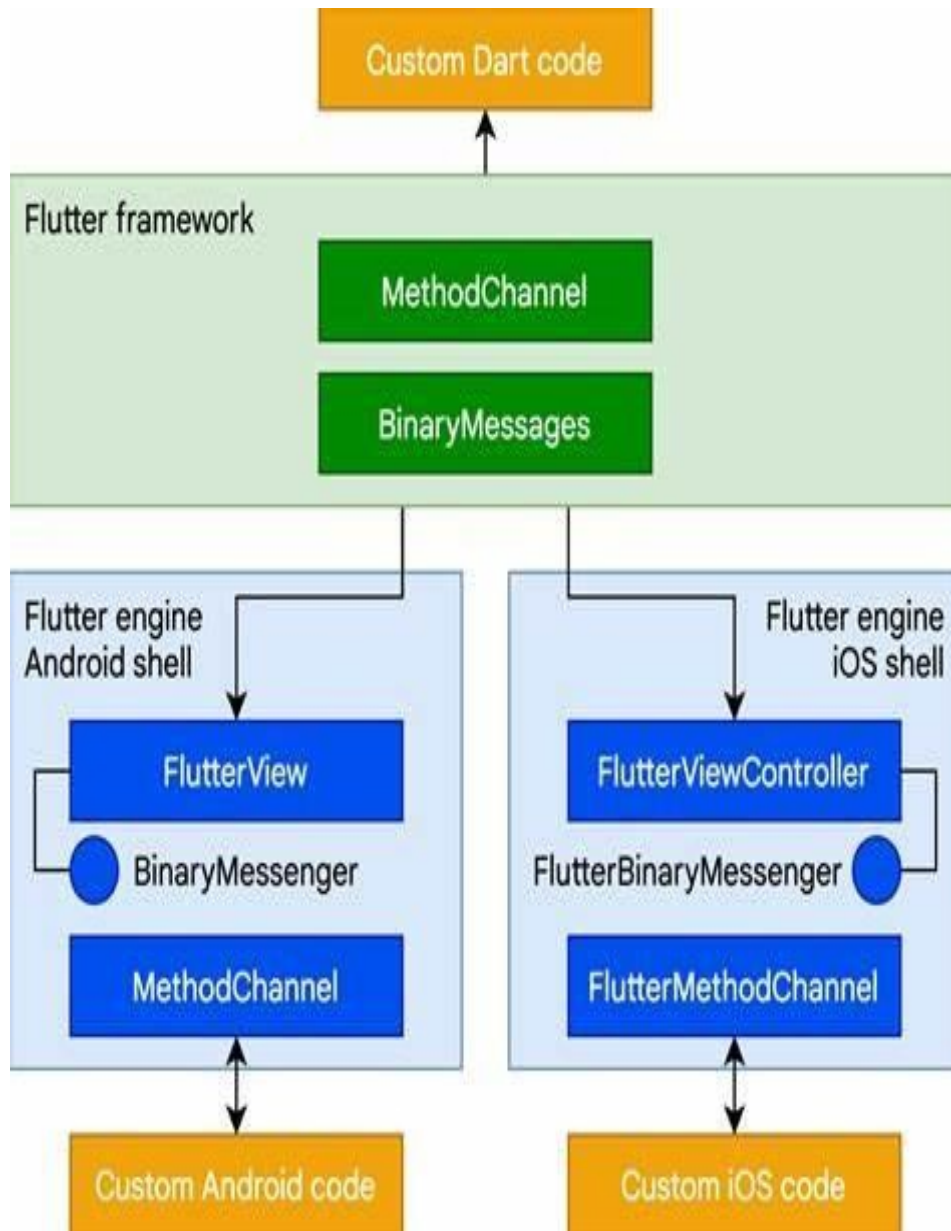


Fig:2.1: Proposed Application Architecture

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

1. Custom Dart Code:

This represents any custom code you write in Dart specifically for your Flutter app. It could be business logic, UI components, or any other functionality unique to your application. The custom Dart code interacts with the Flutter framework to achieve various tasks.

2. Flutter Framework:

The Flutter framework is the heart of your app. It provides a rich set of widgets, tools, and APIs for building cross-platform mobile applications.

3. MethodChannel:

The MethodChannel allows communication between Dart code and native platform code (Android or iOS). When we invoke a method on the MethodChannel from Dart, it triggers a corresponding method call on the native side.

4. BinaryMessages:

BinaryMessages is another channel for communication between Dart and native code. It enables the exchange of binary data (such as images, files, or serialized objects) between the two sides.

5. Flutter Engine Android Shell:

On the Android side, the Flutter engine runs within an Android shell.

6. Flutter Engine iOS Shell:

On iOS, the Flutter engine runs within an iOS shell.

- i)FlutterViewController: The iOS view controller responsible for displaying your Flutter UI.
- ii)FlutterBinaryMessenger: Handles communication between the Flutter engine and the iOS platform.

7. Custom Android Code and Custom iOS Code:

These represent platform-specific code written in Java/Kotlin (for Android) and Objective-C/Swift (for iOS).You'll write custom code here to interact with native features or APIs that aren't directly accessible from Flutter.

8. Firebase:

Firebase

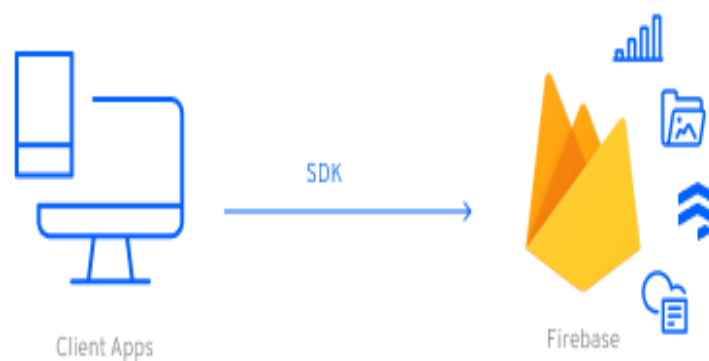


Fig:2.2: Firebase Architecture

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

i) Firebase Authentication:

Firebase Authentication offers ready-to-use authentication services. It integrates seamlessly with Flutter, allowing users to sign up, sign in, and manage their accounts. MFA (Multi-factor Authentication) can be enabled to enhance security by requiring additional verification steps (e.g., SMS codes, email, or biometrics).

ii) Firestore (Realtime Database):

Firestore is a NoSQL database provided by Firebase. It stores structured data in collections and documents. You can use Firestore to store user profiles, chat messages, app settings, and more.

iii) Firebase Storage:

Firebase Storage provides secure cloud storage for files (e.g., images, videos, documents). You can upload and download files directly from your Flutter app. Data-at-rest encryption ensures sensitive files are protected.

iv) Firebase Analytics:

Firebase Analytics tracks user behavior, app usage, and events. You can gain insights into user engagement, retention, and conversion rates. Use analytics data to make informed decisions about app improvements.

9. 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).

2.3.DATA FLOW DIAGRAM

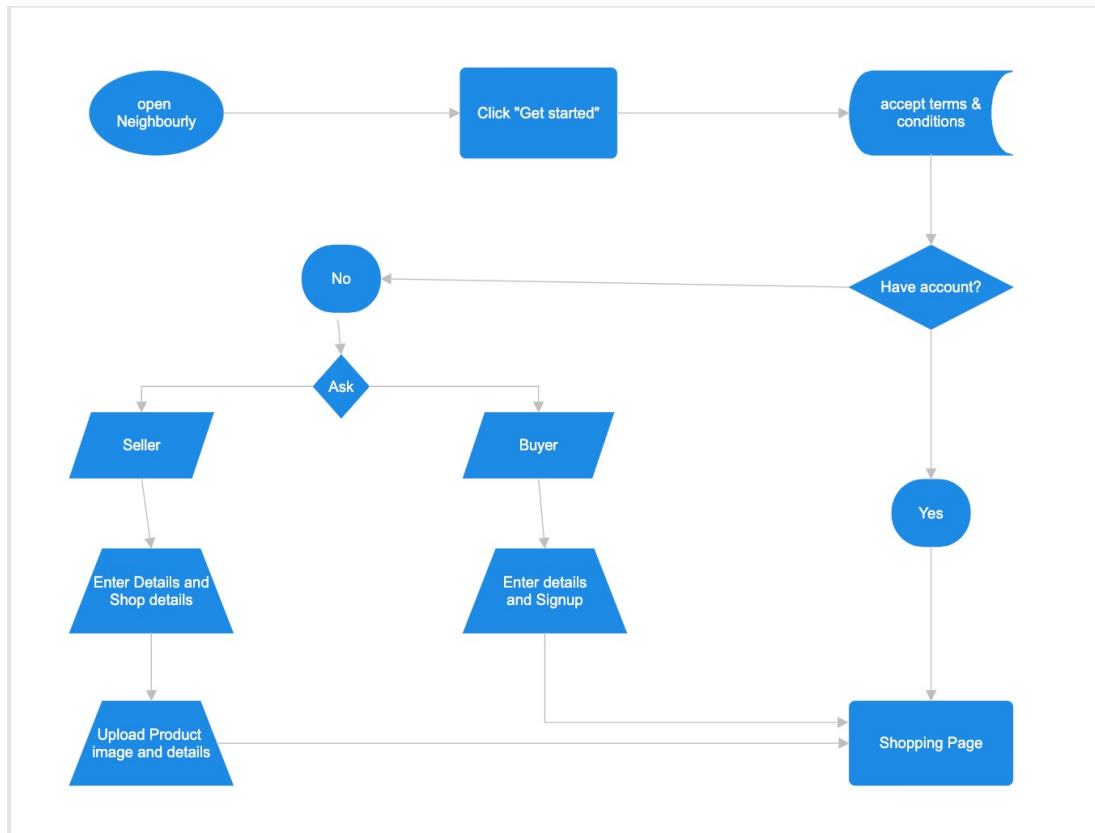


Fig:2.3: Data Flow Diagram

The following steps are used to describe Fig:2.3 Data flow diagram:

User Opens App: The user initiates the process by launching the Neighbourly app on their device.

Getting Started: Upon opening the app, the user encounters a "Get Started" button. Clicking this button signifies their intent to proceed with using the app.

Terms and Conditions: The app presents the user with the terms and conditions governing its use. The user must accept these terms to continue.

Account Check: The app checks if the user already has an existing account within the Neighbourly system. This verification step might involve checking for stored login credentials on the device.

Account Exists (Yes):

Shopping Page: If the user has a previously established account, they are automatically directed to the "Shopping Page." This page allows them to browse product listings, search for specific items, and initiate purchases.

Account Exists (No):

Role Selection: If the user doesn't have an account, the app prompts them to choose their desired role within the platform: Seller or Buyer.

Seller Registration:

Seller Details: If the user selects "Seller," they are presented with a registration form where they enter their personal details and shop information. This might include name, contact information, and shop description.

Product Listing: Upon successful registration, the seller is granted access to functionalities that allow them to upload product images and descriptions. This enables them to list their products for sale within the Neighbourly app.

Buyer Registration:

Buyer Details: If the user chooses "Buyer," they are directed to a registration form where they enter their personal details for account creation. This might include name, email address, and potentially a password.

Shopping Page: After successful buyer registration, the user is directed to the "Shopping Page." This page allows them to browse product listings, search for specific items, and initiate purchases.

End

The flow diagram concludes here, having described the user's journey from opening the app to reaching the main functionality based on their chosen role (Seller or Buyer). This explanation provides a step-by-step breakdown of the initial user experience within the Neighbourly app.

2.4.MODULES

2.4.1. User Authentication:

Description: This module manages user authentication, including signup and signin functionalities, ensuring secure access to the Neighbourly platform.

Algorithm:

User Signin:

User navigates to the login page and enters their registered email address and password.

The system verifies the entered credentials against the data stored in the Firebase Authentication service.

Upon successful verification, the user gains access to their Neighbourly account.

In case of incorrect credentials, appropriate error messages are displayed, prompting the user to retry or reset their password.

User Signup:

New users select the "New User" option on the login page.

Users are presented with the choice to register as either a buyer or a seller. If registering as a buyer, the user provides essential details such as their name, email address, and password. If registering as a seller, additional information such as shop details and product categories are requested.

The system validates the provided information, ensuring adherence to required formats and completeness.

Upon successful registration, user credentials are securely stored in the Firebase Authentication service, and relevant profile data is recorded in the Firebase Firestore database.

2.4.2. Dual Registration System:

Description: This module facilitates the dual registration system, allowing new users to register as both buyers and sellers during the signup process.

Algorithm:

Buyer Signup Process:

Initiating Signup:

The user selects the "New User" option during the signup process on the Neighbourly platform.

Providing Personal Information:

The user enters personal details such as their name, email address, and a secure password.

Verification and Validation:

The system verifies the entered email format and checks for password.

It ensures that all required fields are filled out correctly before proceeding.

Confirmation and Account Creation:

Upon successful verification and validation, the user's account is created with the role of a buyer.

User data, including personal information, is securely stored in the Firebase Authentication service and Firestore database.

Completion and Access:

They can now log in to the Neighbourly platform and start exploring products and services available for purchase.

Seller Signup Process:

Initiating Signup:

Similar to the buyer signup process, the user selects the "New User" option during signup on the Neighbourly platform.

Choosing Seller Role:

Users interested in selling their products or services choose the option to register as a seller.

Providing Additional Business Information:

Sellers are prompted to provide additional details such as the business name, address, and preferred product categories.

This information is essential for establishing their seller profile and showcasing their offerings on the Neighbourly platform.

Verification and Validation:

Similar to the buyer signup process, the system verifies and validates the provided information to ensure accuracy and completeness.

Confirmation and Account Creation:

Upon successful verification and validation, the user's account is created with the role of a seller.

User data, including personal and business information, is securely stored in the Firebase Authentication service and Firestore database.

Completion and Access:

They can now log in to the Neighbourly platform with their seller credentials and begin managing their shop, uploading products, and interacting with customers.

2.4.3. Location-Enhanced Shopping:

Description:

The Location-Enhanced Shopping module enhances the shopping experience on the Neighbourly platform by enabling users to filter product listings based on their desired location. By clicking on the location icon on the shopping page, users can select a specific location, and the system will display products available within or from that location.

Algorithm:

Initiating Location Selection:

Upon accessing the shopping page, users notice the location icon displayed prominently at the top of the page.

User Interaction:

Users click on the location icon to initiate the location filtering process.

Selecting Desired Location:

A searchbar appears, presenting users to select their desired location.

Users can input a custom location in the search bar.

Processing User Input:

Upon selecting a location, the system processes the user's input and retrieves relevant product listings associated with the chosen location.

Filtering Product Listings:

The system filters product listings based on the selected location, displaying only those products that are available within or from the specified area.

Displaying Filtered Results:

Filtered product results are displayed on the shopping page, replacing the existing product listings with those matching the selected location.

Real-Time Updates:

If users change the selected location or input a new location, the system dynamically updates the displayed product listings to reflect the changes.

User Satisfaction and Engagement:

By providing users with the ability to filter products based on their desired location, the Location-Enhanced Shopping module enhances user satisfaction and engagement with the Neighbourly platform.

Users can easily find products available in their vicinity or from specific locations of interest, facilitating a more personalized and efficient shopping experience.

2.4.4. DYNAMIC HOMEPAGE:

Description:

The Homepage with Dynamic Features module serves as the central hub of the Neighbourly platform, providing users with access to essential functionalities through dynamic navigation tabs. Users can explore personalized content, manage their shopping cart, navigate product categories, and access their profile settings directly from the homepage.

Algorithm:

Navigation Tabs Initialization:

Upon accessing the Neighbourly platform, the homepage initializes dynamic navigation tabs for essential sections.

Home Tab Content:

The Home tab displays personalized content, including featured products and recommendations to the user's preferences.

Cart Tab Functionality:

The Cart tab allows users to view and manage items added to their shopping cart. They can add, remove, or modify items as needed.

Categories Tab Navigation:

The Categories tab provides users with a categorized view of available products, enabling them to browse and explore various product categories.

Profile Tab:

The Profile tab allows users to access and manage their account settings, view order history, and update personal information as necessary.

2.5.DEVELOPMENT AND DEPLOYMENT SETUP

Development Environment:

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

IDEs like Visual Studio Code or Android Studio for coding.

Create Project:

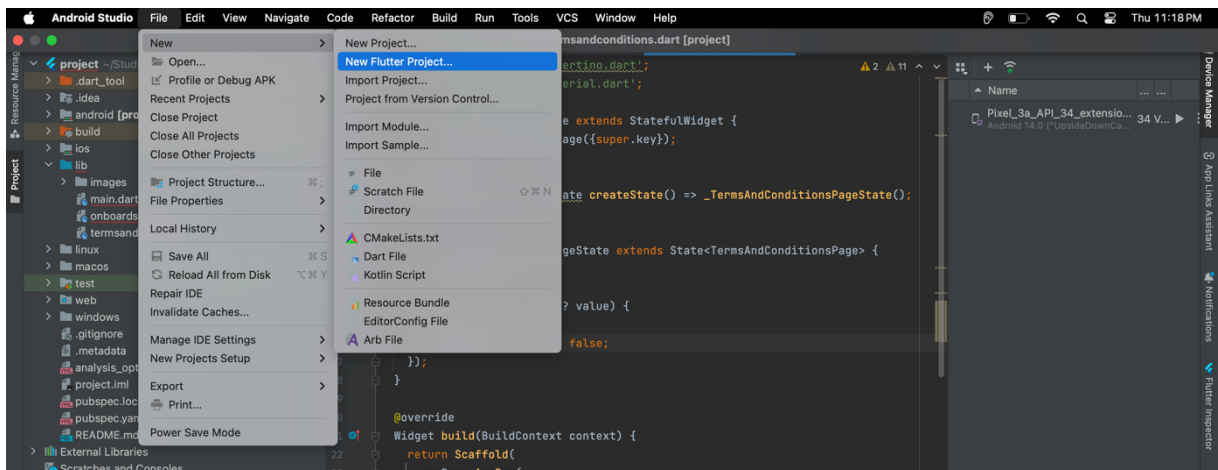


Fig:2.4: Create New Flutter Project

The following steps are used to describe Fig:2.4 to Create New Flutter Project

Open android studio >> File >> New >> New Flutter Project

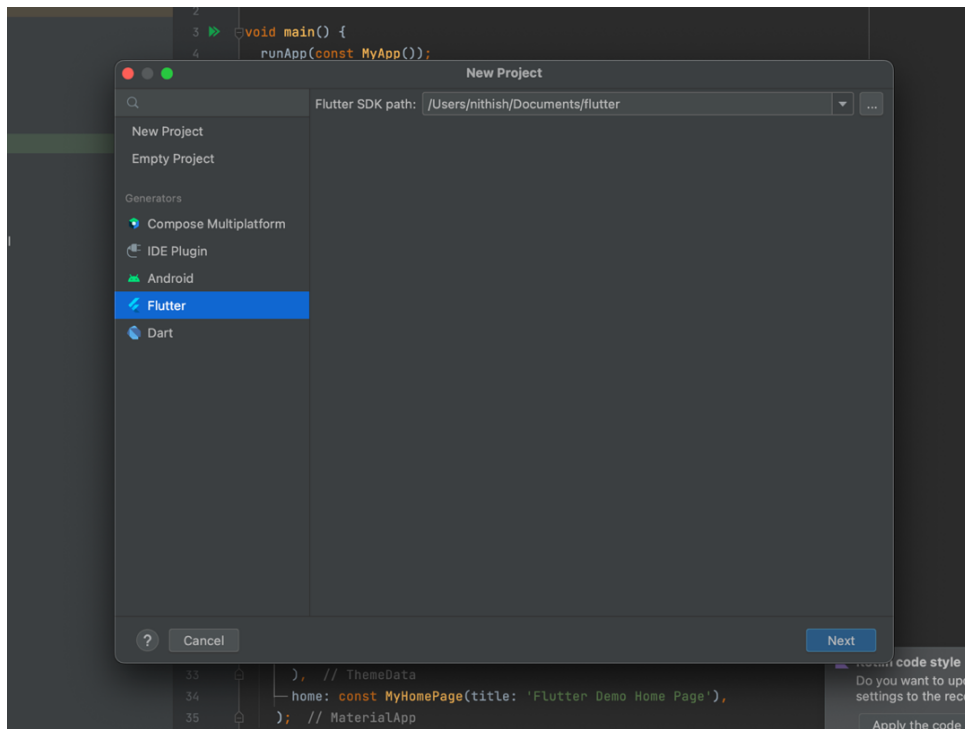


Fig:2.5: SDK Path Integration

Fig 2.5 Shows the SDK path integration to our flutter project.

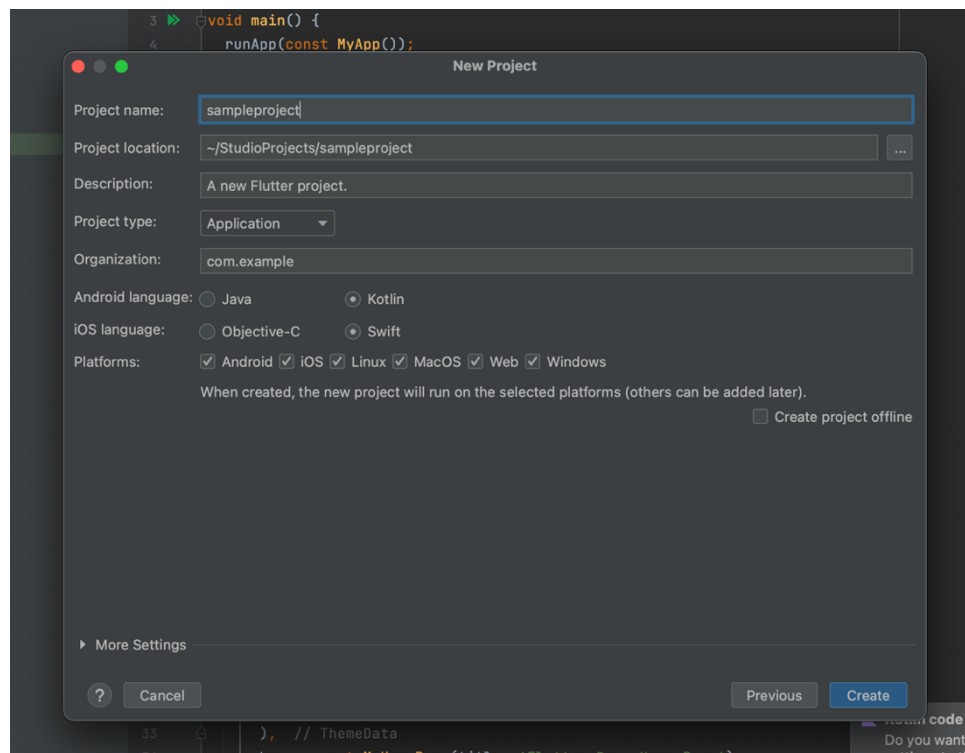


Fig:2.6: Project Creation

Fig 2.6 Shows the project creation of a flutter project with the project name, description and languages used.

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.

Firebase Configuration:

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

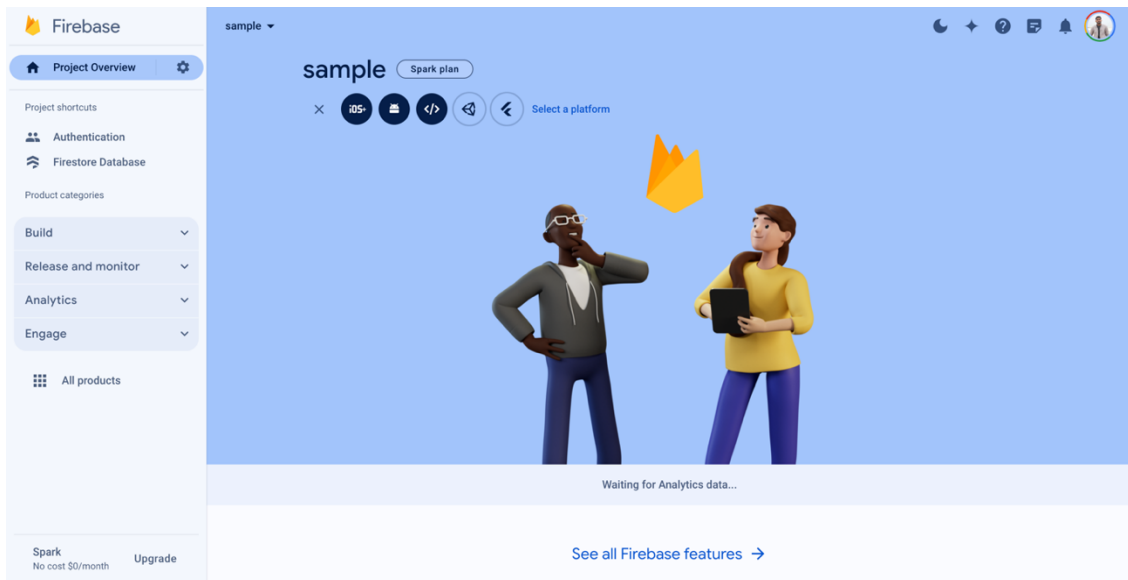


Fig:2.7: Firebase Connection

Fig:2.7 Depicts the firebase connection to the desired flutter project, where various platforms like android, iOS are integrated.

Firestore Authentication:

Firestore Authentication set up for secure user login and identity verification.

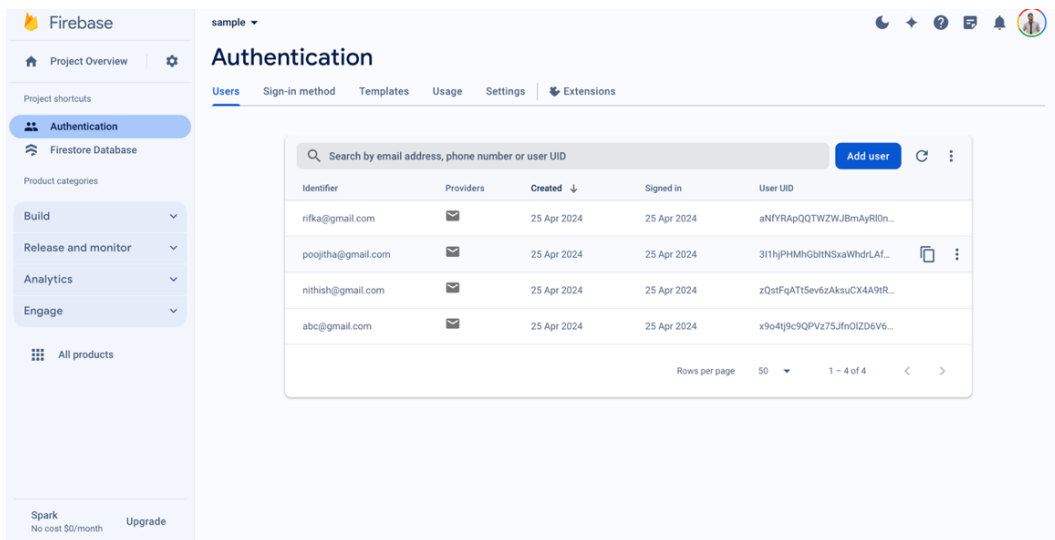


Fig:2.8: Firestore Authentication

Fig:2.8 details the setup of Firestore Authentication for secure user login and identity verification.

Deployment Process:

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

Supported Platforms: Cross-platform support for Android, iOS, and web Integrated

Development Environment (IDE):Developers can choose between Visual Studio Code or Android Studio for coding.

User Authentication: Firestore Authentication ensures secure user login

CHAPTER 3

RESULTS AND DISCUSSION

DART CODES:

main.dart: Entry point for the Neighbourly app, initializing the application and defining its structure.

HomePage.dart: Welcome page of the app, displaying dynamic contents.

TermsandConditionPage.dart: Page displaying terms and conditions for app usage.

SigninPage.dart: Page for user authentication and login.

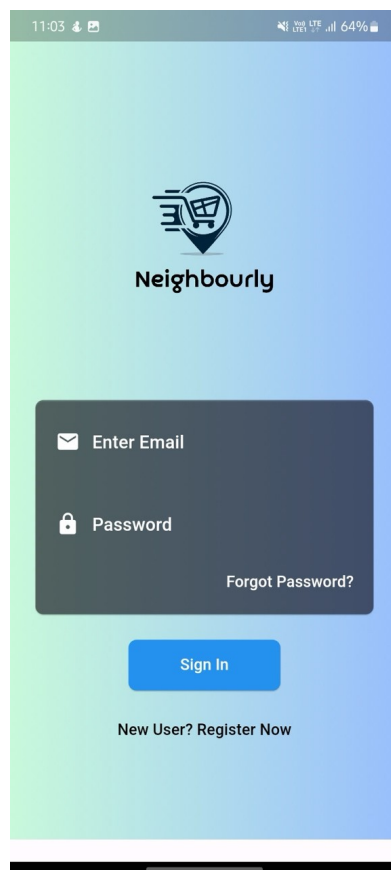


Fig:3.1: SignIn Screen

Fig 3.1 illustrates the Page for user authentication and login.

signuppage.dart: Page for user registration and account creation and ask either Buyer or Seller.

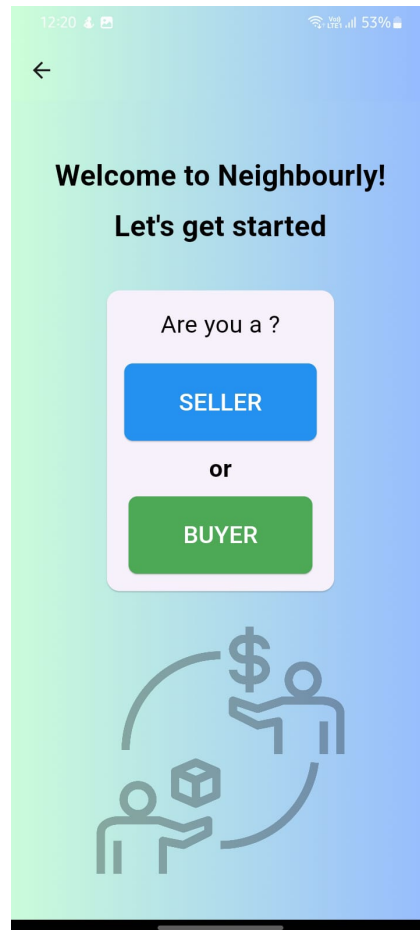


Fig:3.2: Sign-up Screen

Fig 3.2 depicts the Sign-up page, offering users the option to create a new account as either a buyer or a seller.

BuyerSignup Page.dart: Page specifically for buyer registration and account creation.

SellerSignup Page.dart: Page specifically for seller registration and account creation.

UploadProductPage.dart: Page for sellers to upload product details and images.

HomeShoppingPage.dart: Page displaying products for home shopping, with dynamic features.

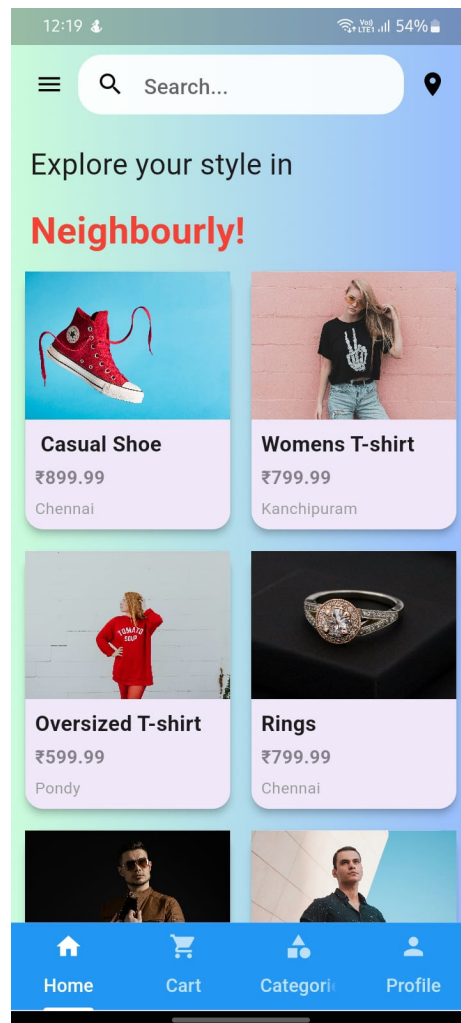


Fig:3.3: Shopping Screen

Fig:3.3 showcases the main shopping page featuring home shopping products, enriched with dynamic functionalities

ShoppingPage.dart: Detailed shopping page displaying product details other informations.

CartPage.dart: Page displaying items added to the shopping cart for purchase.

Categoriespage.dart: Page displaying various product categories for browsing.

profilePage.dart: Page displaying user profile information and settings.

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:

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.

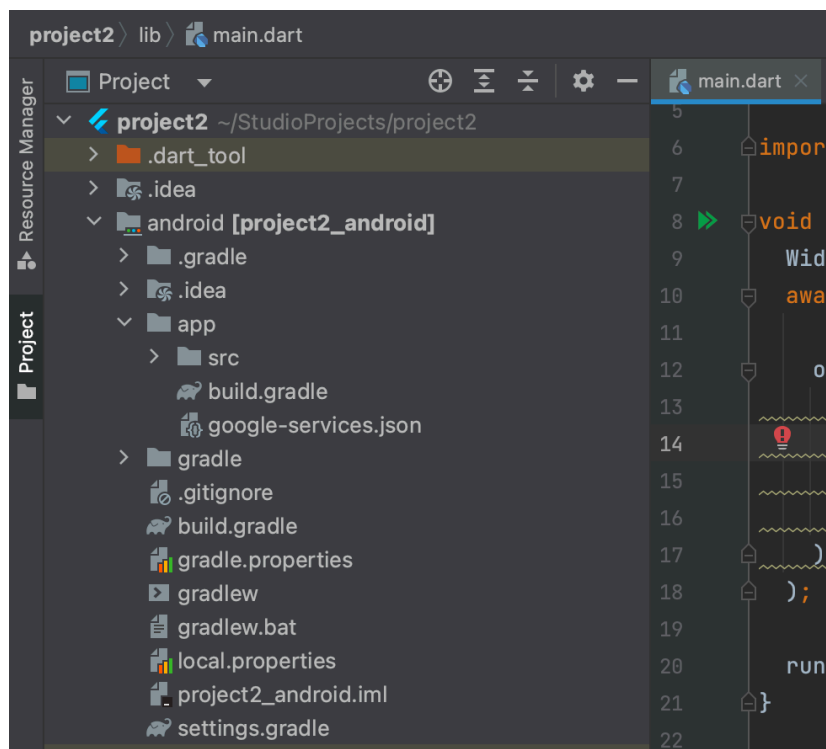


Fig:3.4: Firebase Configurations

Fig 3.4 shows Firebase Configurations involve adding Firebase configuration files to your Flutter project and following provided instructions for Android and iOS.

i) 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
```

ii) Initialize Firebase:

In your main.dart or wherever you initialize your Flutter app, make sure to initialize Firebase. If you want to implement user authentication, Firebase provides the firebase_auth package. You can use it to sign in users using various methods like email/password, Google Sign-In, etc.

Dart code:

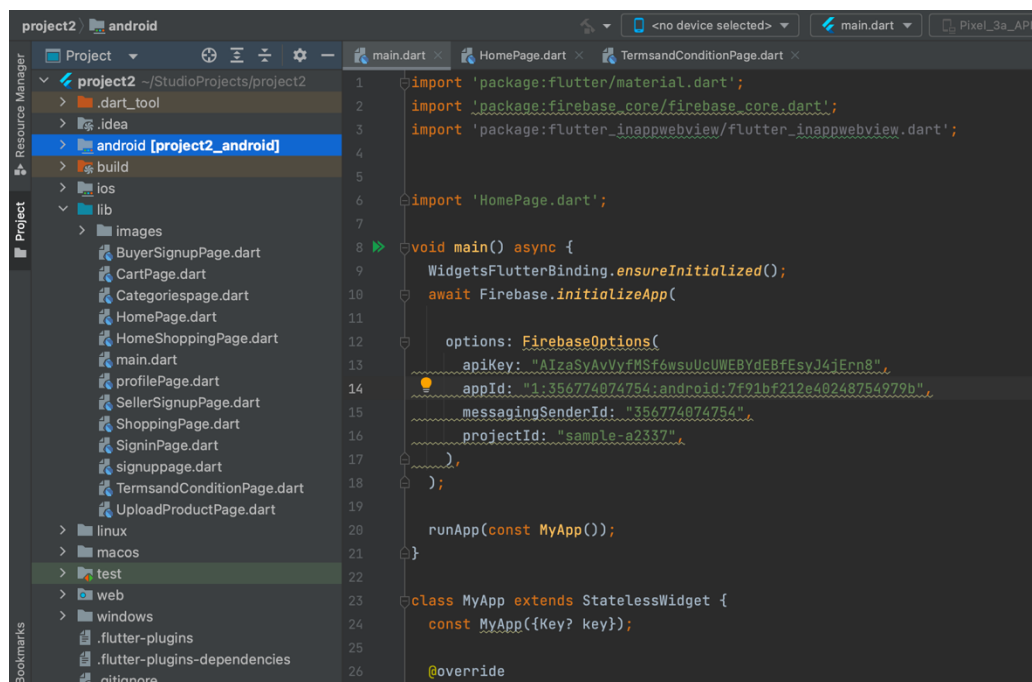


Fig:3.5: Initializing Firebase

Fig:3.5 highlights the importance of initializing Firebase in your main.dart file for enabling user authentication functionalities using firebase_auth package

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1. CONCLUSION

In conclusion, Neighbourly stands as a testament to the fusion of technology and community spirit, epitomizing the potential for digital platforms to empower local economies and foster meaningful connections. By harnessing the capabilities of Flutter and Firebase, Neighbourly has reimagined the traditional e-commerce landscape, offering a sophisticated yet accessible platform for buyers and sellers alike.

Through its intuitive interface and innovative features such as location-based filtering and personalized recommendations, Neighbourly has sought to enhance the shopping experience, providing users with a seamless journey from discovery to purchase. Furthermore, its emphasis on supporting local businesses and promoting sustainability reflects a commitment to responsible commerce and social responsibility.

In the realm of academia, Neighbourly serves as both a case study and a testament to the transformative power of technology-driven solutions. As students, we have not only conceptualized and developed a functional e-commerce platform but also contributed to the discourse surrounding digital innovation and societal impact.

In closing, Neighbourly stands as a radiant beacon amidst the ever-expanding horizon of innovation, casting its illuminating glow upon the profound potential lying at the nexus of technology, commerce, and community.

4.2. FUTURE WORK

Enhanced Personalization: Further refine the recommendation engine to deliver more personalized suggestions based on user behavior, preferences, and demographics.

Advanced Analytics: Implement robust analytics tools to gather insights into user interactions, purchasing patterns, and market trends, enabling data-driven decision-making and targeted marketing strategies.

Performance Optimization: Conduct thorough performance analysis and optimization efforts to enhance app speed, reduce loading times, and minimize resource consumption, ensuring a seamless user experience across various devices and network conditions.

Security Enhancements: Implement advanced security measures such as encryption, data masking, and multi-factor authentication to fortify user data protection and safeguard against potential cyber threats and data breaches.

Machine Learning Integration: Integrate machine learning algorithms for tasks such as product recommendation, sentiment analysis of user reviews, and predictive analytics to enhance user engagement, personalization, and decision-making capabilities within the app.

Progressive Web App (PWA) Development: Explore the conversion of Neighbourly into a Progressive Web App (PWA) to extend its reach to users across a wider range of devices and platforms, leveraging features such as offline access, push notifications, and home screen installation.

Expanded Payment Options: Introduce additional payment options such as digital wallets, cryptocurrency, and installment plans to cater to diverse user preferences and increase conversion rates.

Localized Features: Develop region-specific features and content to cater to the unique needs and preferences of different communities, fostering inclusivity and relevance across diverse geographical regions.

Community Engagement Initiatives: Launch community-driven initiatives such as local events, promotions, and loyalty programs to foster a sense of belonging and encourage active participation within the Neighbourly community.

Sustainable Practices: Implement environmentally conscious practices such as eco-friendly packaging options, carbon offset programs, and partnerships with sustainable brands to promote environmental responsibility and contribute to a greener future.

Accessibility Enhancements: Ensure that the Neighbourly platform is accessible to users with disabilities by incorporating accessibility features such as screen reader compatibility, alternative text descriptions, and keyboard navigation options.

International Expansion: Explore opportunities for international expansion to reach new markets and connect with users worldwide, adapting the platform to accommodate diverse languages, currencies, and cultural preferences.

REFERENCES

1Journals:

[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/flutter-article> feeds
(accessed Apr. 25, 2024).

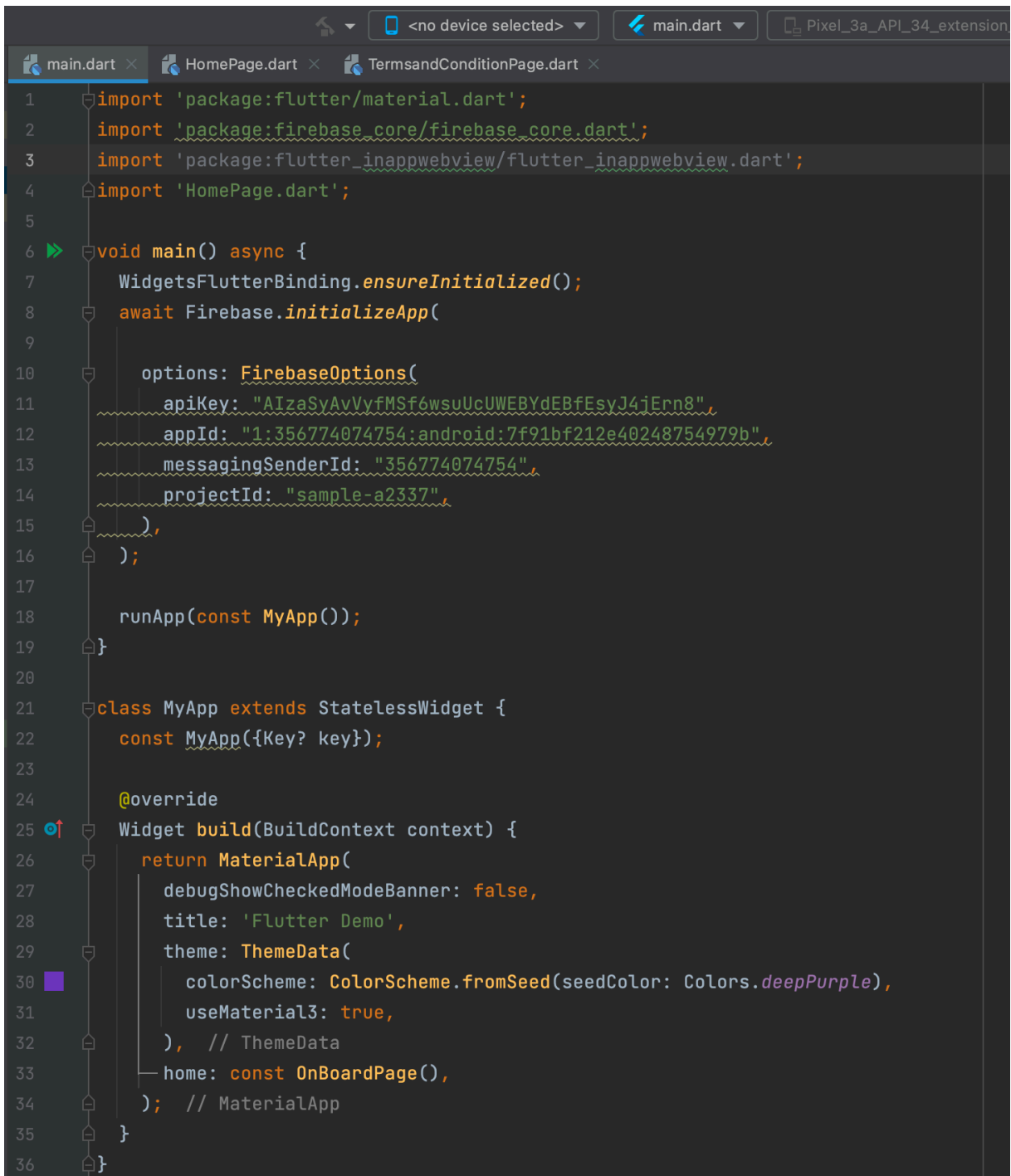
[2] Google, "Firebase documentation," Flutter Documentations,
<https://firebase.google.com/docs>

[3] Wilson, W. (2014). The ultimate list of flutter resources (over 200!). Flutter Resources, <https://wilsonwilson.dev/articles/flutter-resources> .

APPENDICES

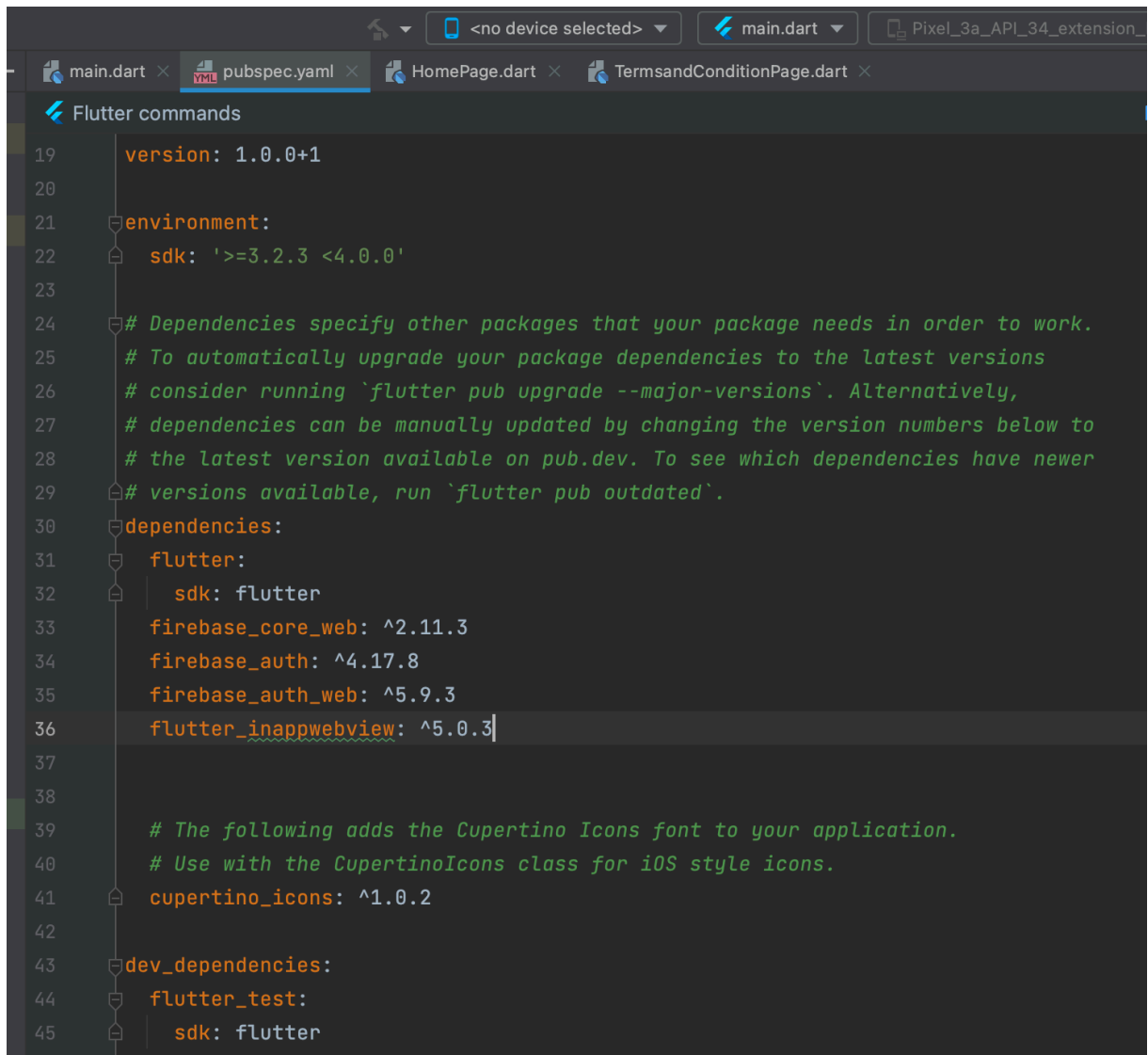
A.SAMPLE CODE

i)Main Code:



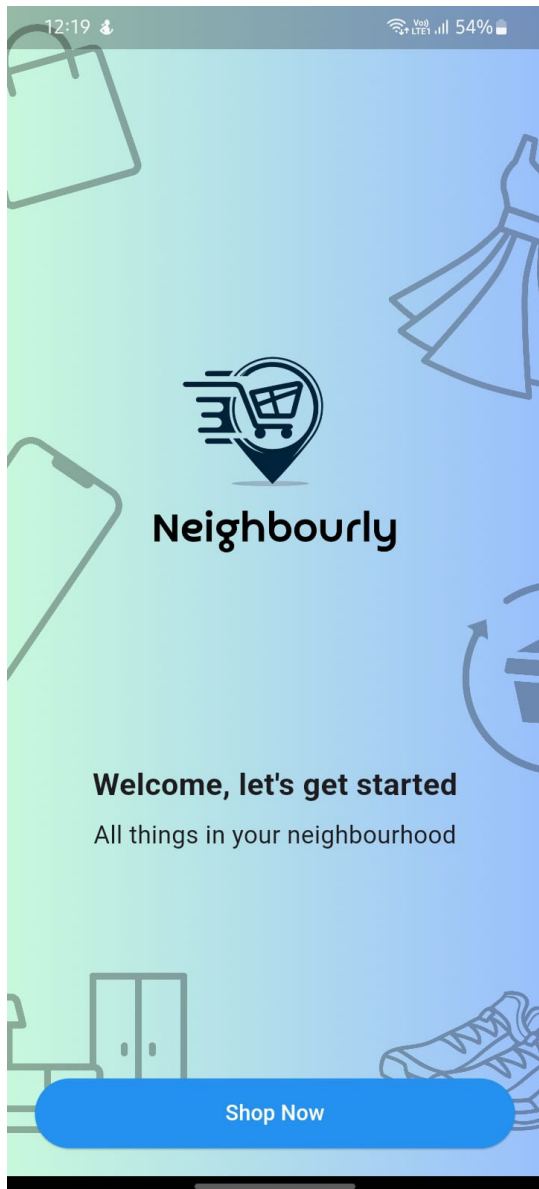
```
1 import 'package:flutter/material.dart';
2 import 'package:firebase_core/firebase_core.dart';
3 import 'package:flutter_inappwebview/flutter_inappwebview.dart';
4 import 'HomePage.dart';
5
6 void main() async {
7   WidgetsFlutterBinding.ensureInitialized();
8   await Firebase.initializeApp(
9     options: FirebaseOptions(
10       apiKey: "AIzaSyAvVyfMSf6wsuUcUWEBYdEBfEsvJ4jErn8",
11       appId: "1:356774074754:android:7f91bf212e40248754979b",
12       messagingSenderId: "356774074754",
13       projectId: "sample-a2337",
14     ),
15   );
16
17   runApp(const MyApp());
18 }
19
20 class MyApp extends StatelessWidget {
21   const MyApp({Key? key});
22
23   @override
24   Widget build(BuildContext context) {
25     return MaterialApp(
26       debugShowCheckedModeBanner: false,
27       title: 'Flutter Demo',
28       theme: ThemeData(
29         colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
30         useMaterial3: true,
31       ), // ThemeData
32       home: const OnBoardPage(),
33     ); // MaterialApp
34   }
35 }
```

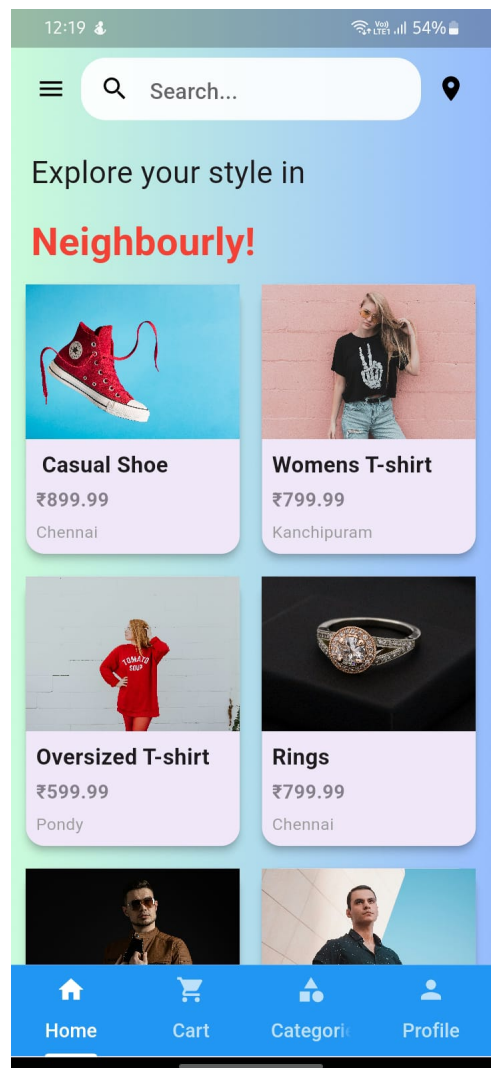
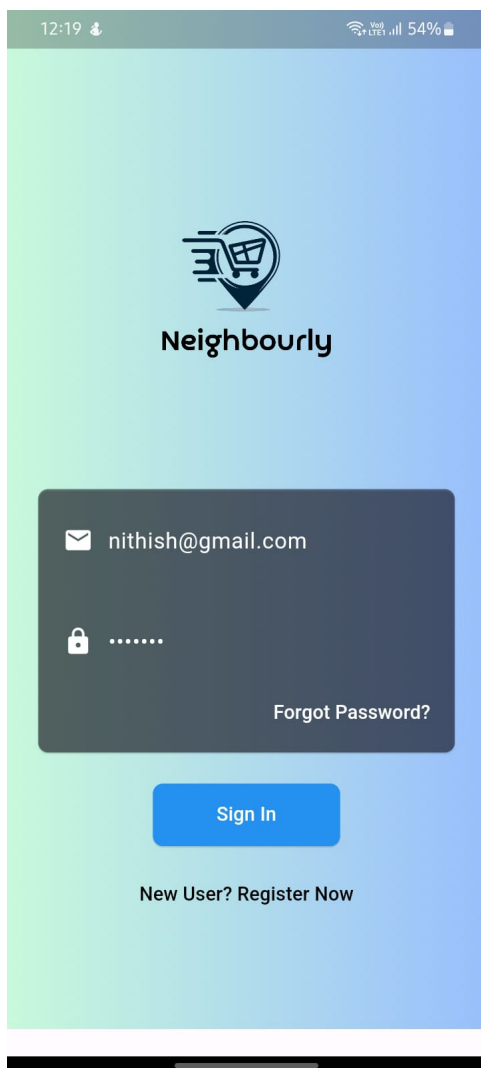
ii)pubspec.yaml

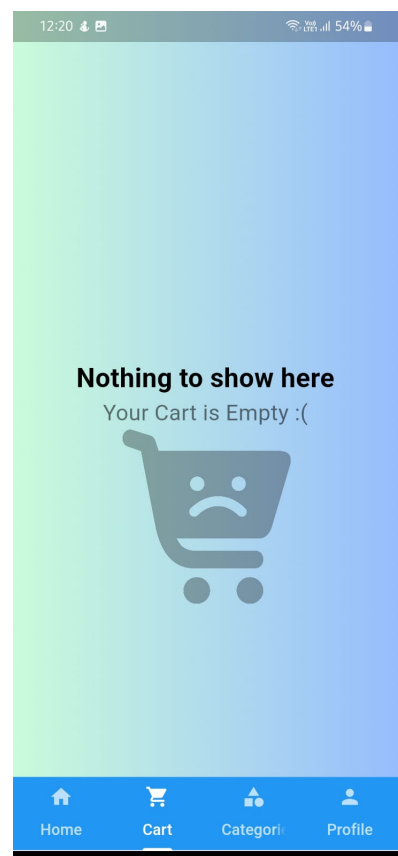
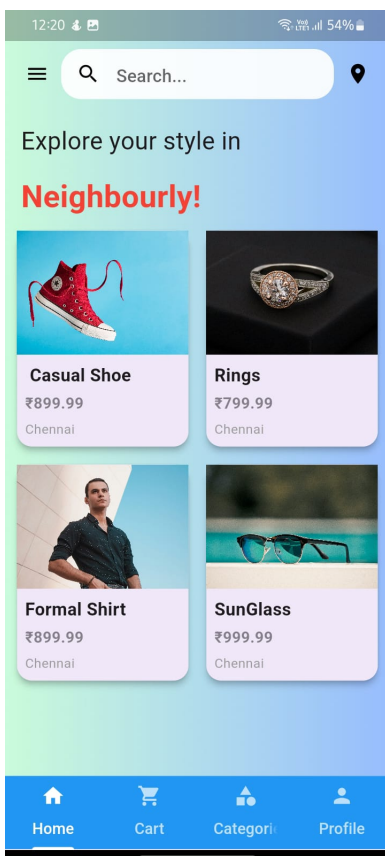
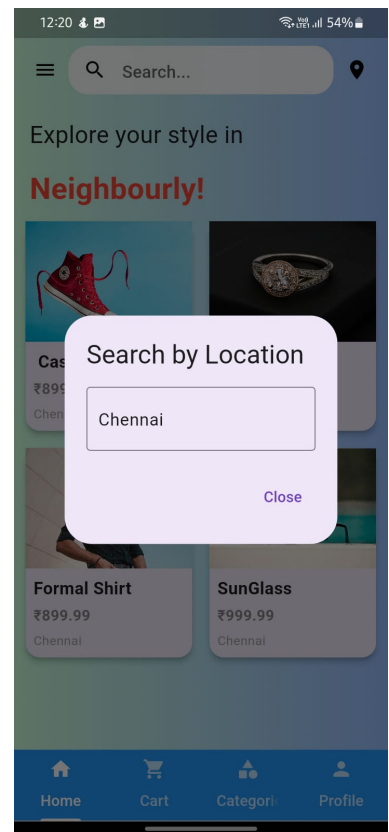
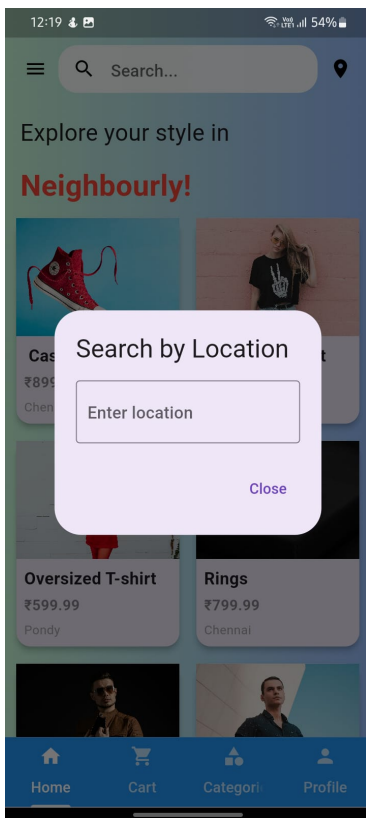


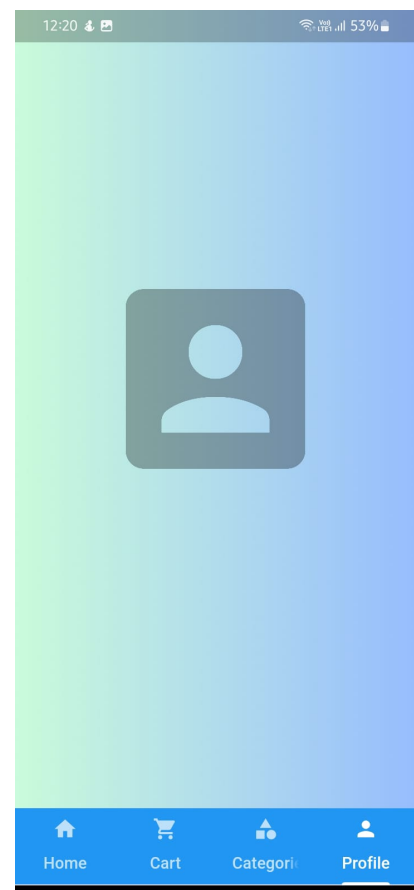
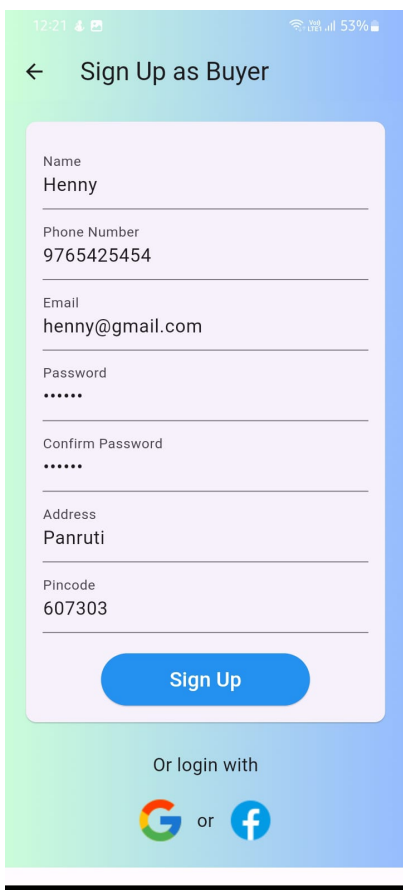
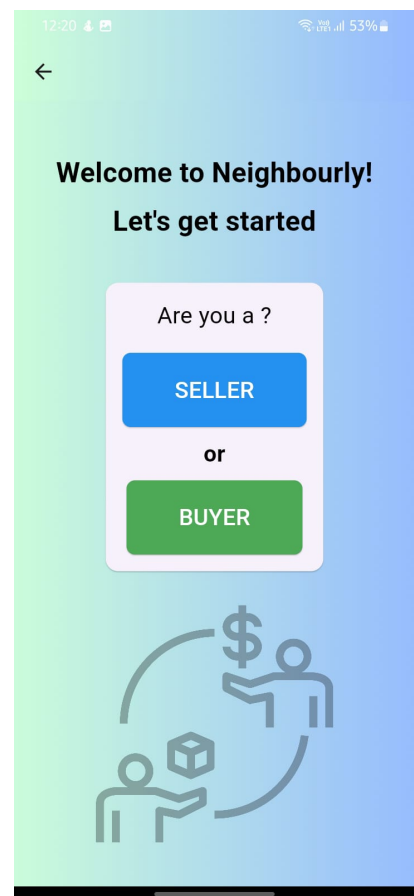
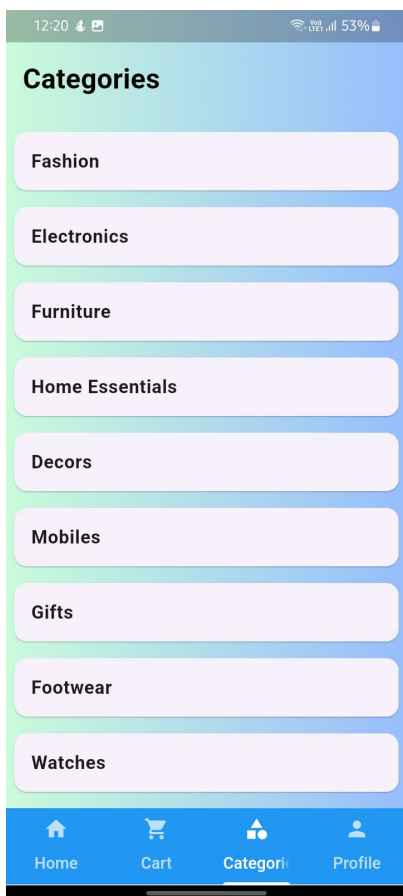
```
19   version: 1.0.0+1
20
21   environment:
22     sdk: '>=3.2.3 <4.0.0'
23
24   # Dependencies specify other packages that your package needs in order to work.
25   # To automatically upgrade your package dependencies to the latest versions
26   # consider running `flutter pub upgrade --major-versions`. Alternatively,
27   # dependencies can be manually updated by changing the version numbers below to
28   # the latest version available on pub.dev. To see which dependencies have newer
29   # versions available, run `flutter pub outdated`.
30   dependencies:
31     flutter:
32       sdk: flutter
33     firebase_core_web: ^2.11.3
34     firebase_auth: ^4.17.8
35     firebase_auth_web: ^5.9.3
36     flutter_inappwebview: ^5.0.3
37
38
39   # The following adds the Cupertino Icons font to your application.
40   # Use with the CupertinoIcons class for iOS style icons.
41   cupertino_icons: ^1.0.2
42
43   dev_dependencies:
44     flutter_test:
45       sdk: flutter
```

B.SCREEN SHOTS









12:22 53%

← Sign Up as Seller

Name
Ranvir

Phone Number
7959425653

Email
ranvir@gmail.com

Password
••••••

Confirm Password
••••••

Shop Name
RS Electrics

Shop Category
Electronic

Shop Address
Chennai

Shop Pincode
600019

Upload Products

12:23 53%

← Upload Product

Product Name
Macbook air m1

Product Category
Laptop

Product Price Range
85000



Upload