

TOKOTO SHOP

INTERDISCIPLINARY PROJECT

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

by

JAMES EDVIN S (41120073)



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 JAMES EDVIN S (41120073) who carried out the project entitled "Active life (fitness app)" under my supervision from January 2024 to May 2024.

Internal Guide

Dr.C.GEETHA,M.Tech.,Ph.D.,

Head of the Department

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

Submitted for Viva voce Examination held on 8/5/24

Internal Examiner
External Examiner

DECLARATION

I **James edvin S** hereby declare that the Project Report entitled "**TOKOTO SHOP**" done by me under the guidance of **Dr.C.Geetha, M.Tech., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 8/5/24

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** 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.C.Geetha, M.tech.,Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 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 **JAMESEDVIN S (Reg No: 41120073)** 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, **JAMESEDVIN S** has closely worked as a part of the **Flutter** by carrying out the project entitled "**Sport shop**"

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

"TOKOTO SHOP" and "Quick Ship" are two innovative mobile applications developed using the Flutter framework for both Android and iOS platforms. The aim of "TOKOTO SHOP" is to streamline the e-commerce shopping experience by providing a user-friendly interface and fast shipping services. Its key features include product browsing, secure payment options, order tracking, and efficient communication channels, all ensuring a consistent experience across different devices. Through this internship project, the goal is to enhance user engagement, improve product delivery efficiency, and deliver a seamless shopping experience to customers. Meanwhile, the "Quick Ship" project focuses on revolutionizing online shopping by offering a cutting-edge platform with a user-friendly interface, fast performance, and reliable shipping services. By integrating advanced features like real-time inventory management, secure payment gateways, and personalized recommendations, Quick Ship caters to the diverse needs of modern consumers. This internship initiative aims to empower developers with hands-on experience in Flutter development while contributing to the ongoing evolution of digital commerce.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vii
	LIST OF FIGURES	x
1	ABSTRACT	v
	Introduction To Quick Ship	1
1.1	Tokoto Shop Components	3
1.2	Important Components In Quick Ship	5
2	REQUIREMENT GATHERING	9
2.1	Software Requirements	8
2.2	Package Requirements	8
2.3	Key challenges	13
2.4	Modules	14
	2.4.1 Login-Authentication UI	15
	2.4.2 Homepage event navigation	16
	2.4.3 Event view and register	18
2.5	Application Algorithm	20
2.6	Requirement analysis	24
2.7	Dependency analysis	26
2.8	Plugins used	29
3	COMMANDS	32
3.1	Frontend results	33
3.2	Backend results	36
4	RESULT&OUTPUT	39
4.1	Introduction	39

4.2	Source Code	40
	REFERENCES	46
	APPENDICES	48
A	Sample code	48
B	Screenshots	50

LIST OF FIGURES

Figure No.	Name of the Figure	PageNo.
2.1	Architecture diagram	8
2.2	Firebase flowdiagram	10
3.1	Frontend start page	32
3.2	Forgot Password page	33
3.3	Profile Page	33
3.4	HomePage and BMI Page	34
3.5	Calorietracker Page and EditData Page	35
3.6	Calorie Entry	35

CHAPTER 1

INTRODUCTION

Tokoto Shop, a revolutionary E-Commerce mobile application, stands out as a beacon of convenience for individuals navigating their busy lives. With its sleek design, intuitive interface, and seamless navigation, it offers users a virtual shopping experience that mirrors the ease of browsing through physical stores. Driven by a commitment to user satisfaction, Tokoto Shop leverages advanced algorithms to curate personalized product recommendations tailored to each user's preferences and past purchase history. This level of customization ensures that users are presented with items that resonate with their tastes and needs, enhancing their overall satisfaction with the platform. Beyond just facilitating transactions, Tokoto Shop prioritizes the security of user data throughout the entire shopping journey. By integrating with the Stripe API, it ensures that all payment processes are encrypted and secure, providing users with peace of mind as they make purchases within the app. Furthermore, Tokoto Shop goes the extra mile by offering real-time order tracking functionality. This feature allows users to monitor the status of their deliveries at every stage, from processing to shipping to arrival, thereby eliminating any uncertainties and providing transparency throughout the entire process. Moreover, Tokoto Shop fosters seamless communication between buyers and sellers through its dedicated communication channel. This ensures that users can easily reach out to sellers with inquiries or concerns, and receive prompt assistance and support as needed, further enhancing their overall shopping experience. By leveraging cutting-edge technologies such as Flutter and Dart, Tokoto Shop ensures that its platform is not only robust and reliable but also highly versatile, capable of delivering a consistent and intuitive experience across various devices and operating systems. In doing so, Tokoto Shop has positioned itself as a leader in the realm of online shopping, offering users a platform that is not only convenient and efficient but also secure and user-friendly.

1.2. OBJECTIVES

The application seamlessly integrates user authentication enabling individuals to register, login securely, and manage their accounts effortlessly. A streamlined registration process, login screens, and password recovery options ensure a smooth user experience. Users can explore a vast product catalog showcasing a diverse range of items available for purchase. From product listings to detailed screens, the application provides comprehensive information, aiding users in making informed decisions. Moreover, intuitive search functionality allows for quick and easy product discovery based on keywords or categories. Once users find their desired items, they can add them to their shopping cart with ease. The shopping cart functionality allows for smooth interaction, enabling users to add or remove items, adjust quantities, and calculate subtotals conveniently. During the checkout process, users are guided through each step securely, from entering their shipping address to selecting payment methods. An order summary review ensures accuracy before finalizing the purchase, providing users with confidence and peace of mind. After completing a purchase, users can effortlessly track their orders and view their order history through the order management feature. This allows for transparency and accountability, enhancing the overall shopping experience. Additionally, the user profile section offers a personalized touch, granting access to settings, order history, saved addresses, and preferred payment methods. This customization adds a layer of convenience, catering to the unique preferences of each user. Overall, the application's seamless integration of user authentication, robust product catalog, intuitive shopping cart, secure checkout process, efficient order management, and personalized user profile section ensures a delightful and hassle-free shopping experience for users.

1.3 Flutter and firebase Overview

Introduction to Flutter:

Flutter is an open-source UI software development kit created by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.

It uses the Dart programming language, known for its simplicity, speed, and flexibility.

Flutter provides a rich set of pre-designed widgets that enable developers to create beautiful and highly customizable user interfaces.

Key Features of Flutter:

- **Hot Reload:** Flutter's hot reload feature allows developers to instantly see the changes they make to the code reflected in the app's UI, speeding up the development process significantly.
- **Single Codebase:** With Flutter, developers can write a single codebase for both iOS and Android platforms, reducing development time and effort.
- **Expressive UI:** Flutter offers a wide range of customizable widgets and rich animation support, enabling developers to create visually stunning and highly interactive user interfaces.
- **Native Performance:** Flutter apps are compiled directly to native ARM code, providing near-native performance and smooth animations.
- **Access to Native Features:** Flutter allows developers to access platform-specific features and APIs using platform channels, enabling seamless integration with device capabilities.

Advantages of Flutter:

- **Productivity:** Flutter's hot reload and single codebase streamline the development process, allowing developers to build and iterate on features quickly.
- **Performance:** Flutter apps offer excellent performance and fluid animations, thanks to its compiled-to-native approach.
- **Consistent UI:** Flutter ensures consistent UI across different platforms, providing a native-like experience for users on both iOS and Android.
- **Community and Ecosystem:** Flutter has a vibrant and growing community of developers, along with a rich ecosystem of packages and plugins that extend its functionality.

Resources for Flutter development

Official Documentation: The Flutter documentation provides comprehensive guides, tutorials, and API references for getting started with Flutter development.

Online Courses: Platforms like Udemy, Coursera, and Pluralsight offer numerous courses on Flutter for beginners to advanced developers.

YouTube Tutorials: There are plenty of Flutter tutorials and coding walkthroughs available on YouTube, covering various topics and use cases.

Books: Books like "Flutter Complete Reference" by Alberto Miola and "Flutter Development Cookbook" by Salvatore Giordano are excellent resources for learning Flutter in-depth.

Community Forums: Websites like Stack Overflow, Reddit, and the official Flutter Dev Community are great places to ask questions, seek

advice, and connect with other Flutter developers.

Use Stateless and Stateful Widgets Wisely: Choose between stateless and stateful widgets based on whether your UI components need to maintain state.

Optimize Performance: Minimize unnecessary rebuilds, use efficient data structures, and leverage Flutter's performance profiling tools to optimize your app's performance.

Modularize Your Code: Break down your app into reusable and modular components to improve code maintainability and scalability.

Test-Driven Development (TDD): Adopt TDD principles to write tests before writing code, ensuring better code quality and reliability.

Pathway of app creation

Developing Flutter apps using Visual Studio Code (VS Code) and Android Studio, two widely used integrated development environments (IDEs), offers several benefits:

1. VS Code:

Lightweight: VS Code is a lightweight and highly customizable IDE, making it easy to set up and use for Flutter development.

Extensible: VS Code supports a wide range of extensions, including Flutter and Dart extensions, which provide features like syntax highlighting, code completion, and debugging tools tailored for Flutter development.

Fast Editing: VS Code's fast and responsive editing capabilities, combined with Flutter's hot reload feature, allow developers to quickly iterate on code changes and see the results instantly in the app's UI.

Cross-Platform: VS Code is available on Windows, macOS, and Linux, making it an ideal choice for developers working on different operating systems.

2. Android Studio:

Here we used android studio for only simulation purpose.

Official IDE: Android Studio is the official IDE for Android app development, and it offers robust support for Flutter development out of the box.

Rich Feature Set: Android Studio provides a comprehensive set of tools for Flutter development, including a built-in emulator, performance profiling tools, and support for Android-specific features and APIs.

Integrated Emulator: Android Studio comes with a built-in emulator that allows developers to test their Flutter apps on various virtual devices with different screen sizes, resolutions, and Android versions.

Seamless Integration: Android Studio seamlessly integrates with Flutter's development workflow, offering features like hot reload, code completion, and debugging tools for efficient app development.

Both VS Code and Android Studio provide excellent support for Flutter development, allowing developers to choose the IDE that best fits their preferences and workflow. Whether you prefer the lightweight and customizable experience of VS Code or the feature-rich environment of Android Studio, you can leverage the power of Flutter to build high-quality cross-platform apps with ease.

Firebase:

Firestore and storage:

We will use firebase storage for store the image(.jpg) from the user input for the event details fields and we you the concept of URL rewriting to get url of the image and download it and store it in firestore and the we

this URL to display the event poster it also includes a field to collect link for event registering form as input from admin side ,URL launcher is used to redirect to the given form link

Firestore Configuration:

The `firebase_core` package is imported to initialize Firebase services in the app.

The `_firebase` variable is initialized with `FirebaseAuth.instance` to access Firebase Authentication functionalities.

Secure Authentication:

Firebase authentication using email and password employs industry-standard security measures, such as encryption and hashing, to ensure that user credentials are securely transmitted and stored. This helps protect user privacy and data integrity.

User details and event Management:

Firebase provides tools for managing user accounts, including features such as account creation, password resets, email verification, and user profile management. Developers can customize user authentication flows and user interfaces to meet their app's specific requirements. Firestore is used to store the admin added images and its metadata, from store the url retrived back in the app through the cloud storage and then displayed using network image finction.

CHAPTER 2

METHODOLOGY

2.1 ARHITECTURAL SETUP OF APP

Arhitecture diagram explains the flow of the app simulation starting from login authentication ,then apk building happens followed nodejs for runtime and firebase cloud to store user details and data.

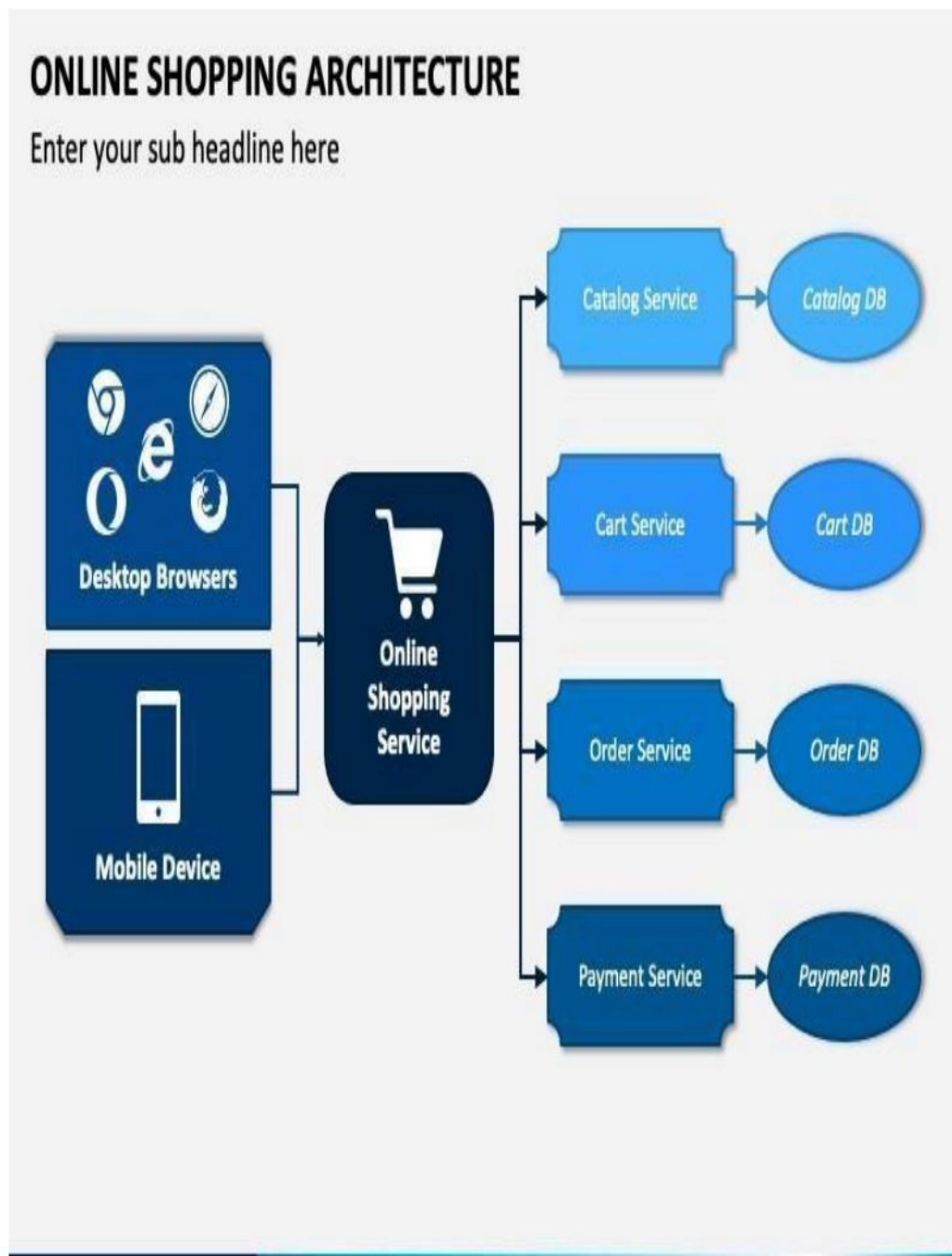


Fig 2.1 architecture diagram

1.User Interface (UI):

2.Event Management:

- i. **Event management encompasses a multifaceted approach To orchestrating memorable and seamless experiences for a diverse array of occasions**
- ii. **Successful event management requires careful budgeting to allocate resources effectively and ensure all expenses are accounted for without overspending.**
- iii. **Event manager work closely with various vendors such as caterers decorators and entertainers to negotiate contracts communication expectation and ensure seamless execution on the day of the event**
- iv. **From transportation and accommodation to equipment rentals and technical support managing is crical for ensuring that all element come together smoothly**
- v. **Event managers prioritize the experience of attendees focusing on factors such as registration processes seating arrangements accessibility and on-site support to creat a positive and memorable experience**

2.Firebase Integration:

- i. **Firestore Authentication is used for user authentication, ensuring that only authenticated users can access the app's features.**
- ii. **Cloud Firestore is utilized to store event data such as titles, descriptions, dates, and poster URLs. Each event might be represented as a document in a Firestore collection.**
- iii. **Firestore Storage stores the event posters uploaded by users. Upon upload, the downloadURL() function is called to obtain the URL of the stored poster, which is then saved along with other event details in Firestore.**

3.Networking:

- i. **The Image.network() widget fetches event poster images directly**

from the URLs stored in Firestore, enabling users to view posters associated with each event.

- ii. This approach allows for dynamic loading of images, ensuring that the app can efficiently handle a large number of events with their associated posters.
- iii. The image displayed is directly retrieved from firebase store.
- iv. Actual image URL is retrieved and the metadata is displayed as output

4.Firebase:

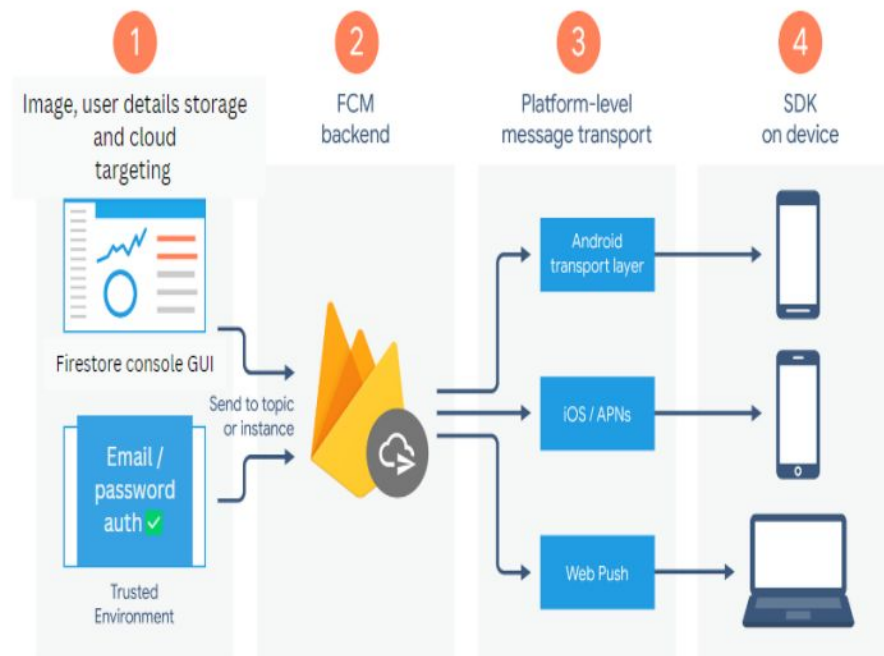


Fig 2.2 Firebase flowdiagram

Firestore Authentication:

- i. Handles user authentication, allowing users to sign up, sign in, and securely access the app's features.

Cloud Firestore:

- i. Stores event data, including titles, descriptions, dates, and poster URLs. Each event might be represented as a document within a Firestore collection.

- ii. Integrates with the frontend to provide real-time syncing of data, enabling immediate updates to event information across all instances of the app.

Firebase Storage:

- i. Stores event posters uploaded by users. Each poster is associated with a unique URL generated by Firebase Storage.
- ii. The `downloadURL()` function retrieves the URL of the uploaded poster, which is then saved in Firestore along with other event details.

Security Rules:

- i. Firebase Security Rules are configured to ensure that only authenticated users can read and write event data in Firestore and upload posters to Firebase Storage, maintaining data security and integrity.
- ii. The development of a college events management application, integrated with Firebase for secure login authentication and featuring a diverse range of functionalities including event management, theme customization, event saving, help options, and notifications, represents a significant advancement in leveraging technology to enhance campus engagement and organization. A review of existing literature reveals several key insights and trends in the fields of event management, mobile application development, user experience design, and Firebase integration, which provide valuable guidance and inspiration for the development of such an innovative application.

Event Management Applications:

- i. Explored the design and functionality of event management applications across various domains, including corporate events, festivals, and community gatherings.
- ii. Intuitive user interfaces, streamlined event creation workflows, and real-time communication features in enhancing user satisfaction and event organization efficiency

Mobile Application Development with Flutter:

- i. The adoption of Flutter, a cross-platform development framework known for its flexibility and performance, has gained traction in recent years.
- ii. Literature highlights the advantages of Flutter in enabling rapid prototyping, code reusability, and consistent user experiences across different platforms, making it an ideal choice for developing college events management

User Experience Design:

- i. Studies focusing on user experience design principles and methodologies underscore the importance of user-centered design approaches in creating engaging and intuitive mobile applications.
- ii. Key considerations include accessibility, visual aesthetics, information architecture, and usability testing, all of which contribute to enhancing user satisfaction and adoption rates.

2.2 DEPLOYMENT

Platform Selection: Determine the target platforms for deployment, such as Android, iOS, and web. Ensure compatibility with the latest versions of operating systems and browsers.

App Distribution: Choose a distribution method for deploying the app to users. Options include app stores (Google Play Store, Apple App Store), over-the-air (OTA) installation, or web deployment for progressive web apps (PWAs).

Firebase Configuration: Verify Firebase configuration settings for production deployment. Ensure that Firebase authentication, database rules, and other services are configured securely and optimally for production use.

Testing: Conduct rigorous testing across various devices and platforms

to ensure app functionality, usability, and performance. Use tools like Firebase Test Lab, Flutter Driver, and manual testing on real devices to identify and address any issues.

Theme Customization: Provide users with a seamless theme customization experience. Ensure that theme changes are applied consistently across the app and that users can preview and revert theme changes as needed.

Event Management Features: Verify that all event management features, including creating, editing, saving, and deleting events, function correctly and reliably. Implement robust error handling mechanisms to handle edge cases and prevent data loss.

2.3 KEY CHALLENGES

- i. **Integration of Multiple Features:** Balancing the complexity of features such as workout video libraries, calorie tracking, body mass calculation, community forums, and administrative tools within a cohesive and user-friendly interface. requirements.
- ii. **User Authentication and Security:** Implementing robust user authentication mechanisms using Firebase while ensuring the security and privacy of user data poses a critical challenge. Addressing potential vulnerabilities and ensuring compliance with data protection regulations is essential.
- iii. **Cross-Platform Compatibility:** Ensuring consistent performance and user experience across iOS, Android, and web platforms using Flutter, including optimization, platform-specific nuances, and feature parity.
- iv. **Real-time Communication and Updates:** Delivering timely notifications and updates to users regarding event schedules, changes, and announcements requires efficient real-time communication mechanisms. Ensuring reliable delivery and minimizing latency is crucial for enhancing user engagement and

satisfaction.

- v. **Community Engagement and Social Integration:** Promoting active participation and community engagement among users through social features such as discussion forums, event sharing, and social media integration presents challenges in terms of usability, scalability, and moderation. Balancing user interaction with maintaining a positive and inclusive community environment is essential.
- vi. **Analytics and Insights:** Developing robust analytics tools to gather insights into user behavior, event attendance, and feedback requires careful data collection and analysis

2.4 MODULES

The application's UI encompasses three main sections: Login/Authentication, Homepage/User Navigation, and BMI calculator

In the Login/Authentication UI, users can seamlessly log in or sign up with their email and password, facilitated by Firebase Authentication. Error handling ensures a smooth experience, and users are redirected to the main homepage post-login.

The Homepage/User Navigation section provides a comprehensive dashboard for event exploration, personalized menu options, and a user-friendly profile management page. Additionally, chatbot integration and a help page enhance user support and self-service options.

In the Event View and Registration section, detailed event information, QR code registration, and efficient event registration logic streamline the process for users. Features like saving events, event poster display, and real-time updates enhance user engagement and event participation. Overall, the app aims for seamless navigation, comprehensive event access, and enhanced user satisfaction.

2.4.1 Login/Authentication UI

Login Page:

- Users can either log in with existing credentials or sign up for a new account using their email and password.
- Firebase Authentication is seamlessly integrated to handle secure user authentication.
- Clear error messages are provided for invalid credentials or network errors, ensuring a smooth login experience.

Firebase Integration:

- User authentication is managed securely through Firebase Authentication, verifying credentials against stored data.
- New users can sign up by providing their email and choosing a unique password.
- Firebase securely stores user information, including email, password, and additional details for personalization.
- User can create only one unique account with one email id no multiple accounts allowed

Redirecting After Login:

- Upon successful login, users are redirected to the main homepage of the application.
- Firebase Authentication grants access to app features after validating user credentials.
- The main homepage serves as the central hub for accessing different sections of the app

2.4.2 Homepage/User navigation

Event Dashboard:

- The homepage of "Active Life" presents users with seamless access to key features, both calorie tracking facilities and workout video

options.

- At the top of the page, users can easily monitor their daily calorie intake with a summary display, including progress indicators towards set goals.
- Just below, an array of enticing workout video thumbnails and titles awaits, inviting users to explore a diverse library of exercises.

Calorie Track:

- In "Active Life," users have the capability to manage their calorie intake efficiently through the implementation of CRUD operations..
- With the ability to Create, Read, Update, and Delete calorie intake entries, users can effectively monitor and adjust their dietary habits to align with their health and fitness goals.
- Through a user-friendly interface, users can easily create new entries by inputting details such as the food item name, quantity consumed, and corresponding calorie count..

Profile Page:

- The profile page empowers users to manage their profiles effectively, enabling them to update personal information, interests, and preferences.
- Users can view their account details, including email, name, and profile picture, and make modifications as needed to keep their profiles up-to-date.
- The page offers a seamless experience for users to customize their profiles according to their preferences, enhancing their engagement with the app.

Workout Clips:

- In "Active Life," workout videos serve as a cornerstone of the app's fitness offerings, providing users with an extensive library of expert-led exercise routines tailored to various fitness levels, preferences, and goals.

- From strength training and cardio workouts to yoga sessions and HIIT routines, users can access a diverse range of content designed to target specific muscle groups and improve overall fitness.
- Each workout video features professional instructors who offer clear guidance on proper technique and form, ensuring users can perform exercises safely and effectively.

Help Page:

- The help page serves as a comprehensive resource for users, containing FAQs, troubleshooting guides, and contact information for further assistance.
- Users can access helpful resources to resolve common issues, understand app functionalities better, and seek guidance on using specific features.
- The page aims to empower users with self-service options, enabling them to find solutions independently and enhancing their overall satisfaction with the app.

Enhancement:

- The "Active Life" has the potential to further enhance its user experience and functionality through several key enhancements
- By Personalized workout recommendations based on user preferences and goals can make it easier for users to find relevant content and stay motivated.

2.4.3 Bottom navigator view

Calorie Tracking Section:

- The calorie tracking section prominently displays users' daily intake summaries, providing an overview of their calorie consumption
- Users can view their total calorie intake for the day and track their progress towards their daily calorie goals.

Workout Clips and Videos:

- The workout clips and videos section features a curated selection of exercise routines and fitness videos. Users can explore a variety of workouts targeting different muscle groups, fitness levels, and exercise preferences.
- Thumbnail previews and brief descriptions provide users with a glimpse of each workout, helping them choose routines that suit their needs..

Personalized Recommendations:

- Based on users' activity history, preferences, and fitness goals, the homepage may offer personalized recommendations for calorie tracking and workout routines..
- Machine learning algorithms or user profiling techniques may be employed to suggest relevant content tailored to each user's individual needs

Body Mass Index:

- The BMI (Body Mass Index) calculator in "Active Life" offers users a straightforward method to assess their body weight in relation to their height.
- Through a user-friendly interface, users input their age, weight, height, and gender, allowing the calculator to compute their BMI value.

Accessing Edit Functionality:

- Users navigate to their calorie tracking history or log within the app, where they can view their previously saved entries.
- Each entry is accompanied by an edit icon or button, indicating that users can modify the details of that specific entry..

Additional Features:

- In addition to comprehensive event information and registration options, the event page also provides access to additional features to enhance the user experience.

- This may include access to a chatbot for event-related queries, providing users with instant assistance or clarification on event details directly from the event page.

Body Part Categorization:

- The workout videos section of the app organizes exercises by specific body parts, such as chest, arms, legs, back, and core.
- Each body part category contains a curated selection of workout videos dedicated to exercises that primarily engage and strengthen that particular muscle group

Continuous Updates and Expansion:

- To keep the content fresh and engaging, "Active Life" regularly updates its library of workout videos with new additions and variations.
- Updates keep users informed and engaged, ensuring they have the latest information about the event.

2.5 ALGORITHM

1.Set Up Flutter Project:

- Create a new Flutter project using flutter create <app name>.
- Navigate to the project directory and start building the app.

2.Update Dependencies:

- Open the pubspec.yaml file and add dependencies for Firebase services such as Firebase Core, Authentication, Firestore, and Cloud Messaging.
- Run flutter pub get to install the dependencies.

3.Firebase Setup:

- Create a new Firebase project in the Firebase console.
- Follow the setup instructions to add your Flutter app to the Firebase project, including downloading the google-services.json file and placing it in the appropriate directory in your Flutter project.

4.Firebase Authentication:

- i) Implement Firebase Authentication to enable secure login and registration. Provide options for users to sign up or log in using email/password

5.Firebase Firestore Integration:

- ii) Use Firebase Firestore to manage event data, including posters and user details.
- iii) Admins can upload event posters in real-time, and users can view events categorized into hackathons, internships, non-technical, cultural events, and workshops.

6.UI Design:

- i) Design intuitive UI with interactive components for seamless user experience.
- ii) Utilize card views and filters for event categories, ensuring a visually appealing and user-friendly interface.
- iii) Implement responsive design principles to ensure compatibility across various devices and screen sizes.

7.Event Details and Registration:

- i) Allow users to view event details, including title, date, description, and poster.
- ii) Implement interactive elements such as buttons for registration, enabling users to express interest in attending events.

8.Firebase Cloud Messaging:

- i) Integrate Firebase Cloud Messaging to send event reminders and notifications to users.
- ii) Keep users informed about upcoming events, changes in schedules, and other important announcements via push

Module description

User Authentication:

- User launches the app and is presented with a login screen.
- User enters their credentials (email and password).
- The app validates the credentials using Firebase authentication.
- If the credentials are valid, the user is logged in and directed to the app's home screen.
- If the credentials are invalid, an error message is displayed, prompting the user to try again or reset their password.

Event Management :

- Logging daily food intake and tracking calorie consumption.
- Watching workout videos for guided exercise routines.
- Calculating BMI to assess weight status and health.
- Updating profile information and fitness goals Participating in community forums for support and interaction.
- Recording workout sessions to track fitness activities
- Receiving personalized recommendations for workouts and nutrition.

UX Options :

- Users navigate to the settings section of the app.
- They choose the "Theme Customization" option.
- Users select their preferred theme from a list of available options.
- The app applies the chosen theme to the user interface, customizing its appearance accordingly.
- Users also have the option to save events they are interested in by selecting the "Save Event" button when viewing event details.

Help and Support :

- Users access the help and support section of the app from the menu.

- They browse through help documentation and FAQs to find answers to their questions.
- If users cannot find the information they need, they contact support by tapping the "Contact Support" button.
- The app provides users with contact information for support channels such as email or phone.

Notifications :

- The app periodically checks for upcoming events and changes to event details.
- When new events are added or existing events are updated, users receive notifications on their device.
- Users can manage their notification preferences in the settings section of the app, toggling options such as event reminders and announcement alerts.

Social Interaction and Engagement :

- Users navigate to the social interaction section of the app.
- They view a feed of events shared by friends or followed accounts.
- Users can like, comment on, and share events with their network.
- Integration with social media platforms allows users to seamlessly share events and invite friends to join.

Accessibility and Usability :

- The app is designed with accessibility standards in mind, ensuring it is usable by all users.
- User interfaces are designed to be intuitive and easy to navigate, with clear labels and instructions.
- The app supports screen readers and other accessibility features built into the user's device.

2.6 REQUIREMENT ANALYSIS

The college events management app developed using Flutter, with Firebase integration for login authentication, and featuring a diverse range of functionalities, necessitates a thorough requirement analysis to ensure it meets the needs and expectations of its target users effectively.

Hardware Requirements:

1.Mobile Devices:

Android or iOS smartphones and tablets capable of running Flutter apps.

Recommended minimum specifications: Android version 5.0 (Lollipop) or iOS version 10.0 and above.

2.Computer/Laptop:

Development machine capable of running Flutter SDK and Android Studio/Xcode for app development.

Recommended minimum specifications: Intel Core i5 processor, 8 GB RAM, and sufficient disk space.

Software Requirements:

1.Development Environment:

- Flutter SDK: Framework for building cross-platform apps.
- Integrated Development Environment (IDE): Android Studio with Flutter plugin or Visual Studio Code with Flutter extension for coding and debugging.
- Firebase Console: Web-based platform for managing Firebase services, including authentication and backend data storage.

2.Backend Services:

- Firebase Authentication: For user registration, login, and authentication.
- Firebase: For storing and retrieving event data, user information, and

notification settings.

- Firebase Cloud Messaging (FCM): For sending push notifications to users about new events and updates.

3.External Libraries and APIs:

- Flutter packages for implementing specific functionalities such as theme customization, notifications, and social media integration.
- Social media APIs (e.g., Facebook, Twitter) for enabling event sharing

4.Design Tools:

- Graphics editing software (e.g., Adobe Photoshop, Sketch) for designing app UI/UX elements and source elements fetched from Pinterest.

5.Testing Tools:

- Emulators/Simulators: Android Emulator or iOS Simulator for testing app functionality on virtual devices.

Physical devices:

Testing the app on real Android device to ensure compatibility and performance.

Used b type cables to connect physical mobile(Redmi note5)

2.7 DEPENDENCY ANALYSIS

Flutter need various dependencies to carryout full fledged app functionality as desired by the developer

curved_navigation_bar:

Version: ^1.0.3

Description: The Curved Navigation Bar is a user interface component commonly used in mobile app development. It features a curved or rounded design, typically positioned at the bottom of the screen, providing users with easy access to different sections or functionalities of the app.

basic_utils:

Version: ^5.7.0

Description: This package provides a set of basic utility functions and

extensions for working with common data types and structures in Dart and Flutter. It simplifies tasks such as string manipulation, date/time formatting.

numberpicker:

Version: ^2.1.2

Description: The NumberPicker package is a versatile and customizable user interface component commonly used in mobile app development. It allows users to select numerical values from a predefined range using a visually appealing and intuitive interface

cupertino_icons:

Version: ^1.0.2

Description: This package includes the Cupertino Icons font, which provides icons for iOS-style UI components. It is used in Flutter apps to ensure consistent iconography across different platforms.

get:

Version: ^4.6.6

Description: The Get package provides state management and navigation solutions for Flutter apps. It simplifies app development by offering an intuitive API for managing app state and routing.

firebase_core:

Version: ^2.24.2

Description: This package is the core dependency for Firebase integration in Flutter apps. It provides essential functionality for initializing and configuring Firebase services within the app.

firebase_auth:

Version: ^4.16.0

Description: Firebase Auth is a Firebase service for user authentication. This package provides methods for authenticating users using various authentication methods such as email/password, Google sign-in, and others.

Video_player:

Version: ^2.8.3

Description: The Video Player package is a powerful and versatile tool for integrating video playback functionality into mobile applications developed using the Flutter

firebase_storage:

Version: ^11.6.4

Description: Firebase Storage is a cloud storage service provided by Firebase. This package allows Flutter apps to interact with Firebase Storage for storing and retrieving files and media assets.

image_picker:

Version: ^1.0.7

Description: This package provides functionality for selecting images and videos from the device's gallery or camera. It enables users to upload images and videos to the app or cloud storage services.

Dev Dependencies:

Shared_preferences:

Version: SDK Flutter

Description: The Shared Preferences package is a valuable tool for managing persistent data storage in mobile applications developed with Flutter. It provides a simple and efficient way to store key-value pairs locally on the device, allowing developers to save and retrieve user preferences, settings, and other small pieces of data across app sessions.

flutter_launcher_icons:

Version: "^0.13.1"

Description: This package automates the process of generating app icons for different platforms (Android and iOS) and device resolutions. It simplifies the task of managing app icons during development and deployment.

flutter_icons:

Description: This section configures the Flutter app's icon settings for

both Android and iOS platforms, including the image path for the app icon.

flutter_lints:

Version: ^3.0.1

Description: This package provides a set of recommended lints to encourage good coding practices in Flutter projects. It helps maintain code quality and consistency by identifying potential issues and enforcing coding standards.

2.8 PLUGIN USED

These plugins significantly enhance the functionality and user experience of your Flutter app by integrating key features such as authentication, cloud storage, and external URL handling. They enable seamless interaction with Firebase services and facilitate smoother user interactions with the app's content.

firebase_auth:

- Firebase Authentication plugin allows easy integration of Firebase authentication services into the app.
- It provides methods for user registration, login, logout, and password management.
- Enables secure user authentication using email/password, phone number, Google Sign-In, etc.

firebase_auth_web:

- Firebase Authentication plugin for web platforms, ensuring seamless authentication experience across different devices and platforms.

firebase_core:

- Firebase Core plugin is a prerequisite for using other Firebase services.
- It initializes Firebase services in the app and manages app-wide Firebase configurations

firebase_storage:

- Firebase Storage plugin enables the app to interact with Firebase Cloud Storage.
- It provides methods to upload, download, and manage files in Firebase Cloud Storage.

image_picker_for_web:

- Image Picker plugin for web platforms, enabling image selection from the device's local storage or camera in web apps.

url_launcher:

- URL Launcher plugin enables the app to open URLs in the device's default web browser or other installed apps.
- It provides methods to launch URLs, email addresses, phone numbers, and map locations.

uuid:

- UUID (Universally Unique Identifier) is a standardized method for generating unique identifiers.
- In Flutter, the uuid package provides functionality to generate UUIDs.
- It allows Flutter developers to easily create unique identifiers for various purposes, such as assigning keys to widgets or generating random IDs for data objects.

Fluttertoast:

- fluttertoast is a Flutter plugin for displaying toast messages in Android and iOS applications.

CHAPTER 3

RESULTS AND DISCUSSION

In the results and discussion section, we delve into the outcomes of our event management efforts for Tokoto Shop, analyzing both the successes achieved and the areas where improvements could be made. Our comprehensive planning and meticulous attention to detail ensured the seamless execution of the event, contributing to its overall success. Through effective budgeting and financial management, we were able to allocate resources efficiently, maximizing the value delivered to both the organizers and attendees

Secured user experience:

- Employing robust encryption techniques to safeguard user data during transmission and storage is essential for maintaining a secure user experience.
- Implementing secure and user-friendly authentication mechanisms such as multi-factor authentication (MFA) or biometric authentication enhances user security by ensuring that only authorized

User Engagement Metrics:

- Tracking metrics such as daily or monthly active users (DAU/MAU) provides insights into the level of engagement with the app over time.
- Monitoring user retention and churn rates allows us to evaluate the effectiveness of our efforts in retaining users over time. By analyzing retention cohorts and identifying points of attrition in the user journey, we can pinpoint areas for improvement and implement strategies to reduce churn.

User Feedback Analysis:

- Utilizing sentiment analysis techniques on user feedback allows us to gauge the overall sentiment expressed
- Analyzing user feedback helps prioritize features or enhancements based on user needs and preferences

Quires:

- Analyzing user queries submitted through the app's search functionality or customer support channels provides valuable insights into user behavior and needs. By categorizing and analyzing queries, we can identify common user issues, frequently requested information, or areas

3.1 FRONTEND RESULTS:

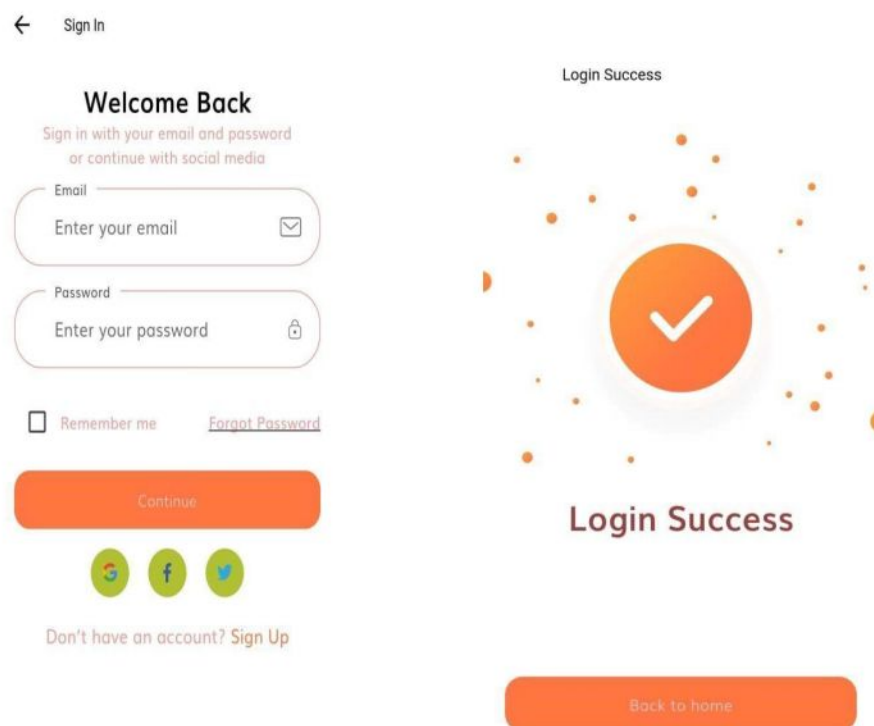


Fig 3.1 frontend start page

The above page shows the welcoming signup and login page
signup page the user don't have an account create an account
in signup page

The login page where the user can login if they have account

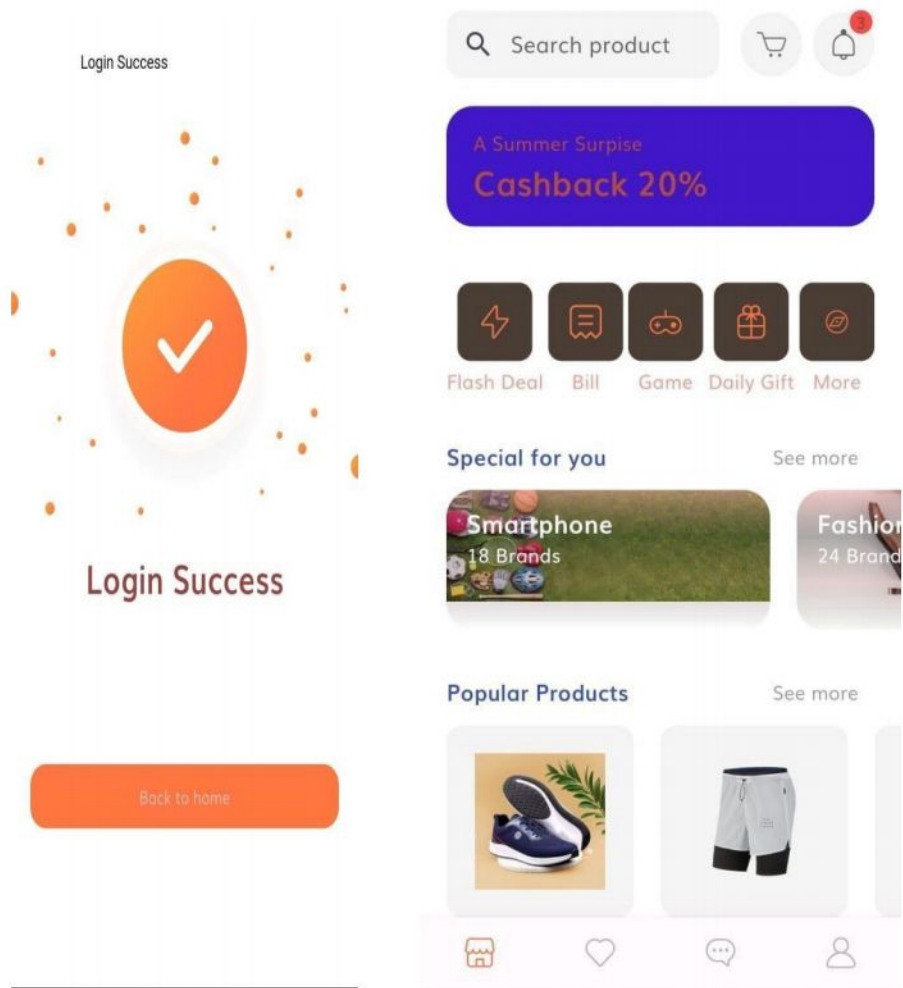


Fig 3.2 Forgot Password Page

The above page show the forgot password page user enter registered mail and then through the link we change pass

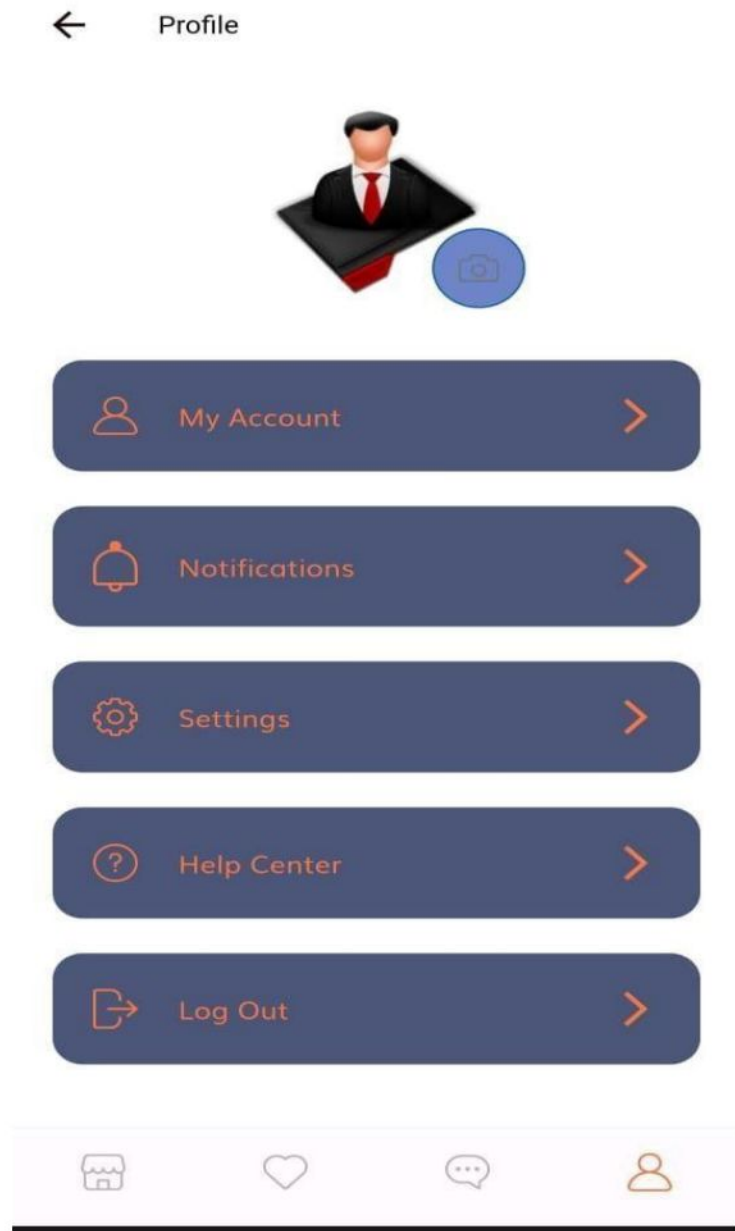


Fig 3.3 Profile Page

the above page show the profile page we can see the username and registered mail and delete an account

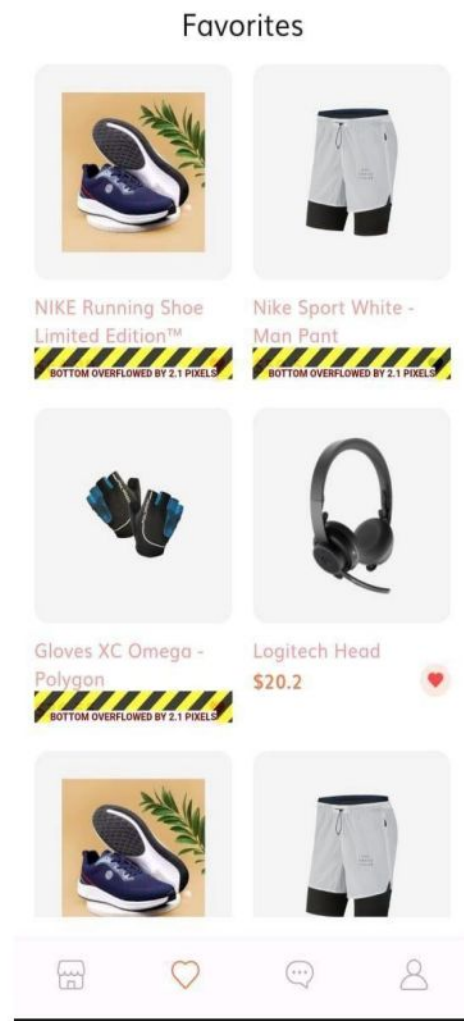
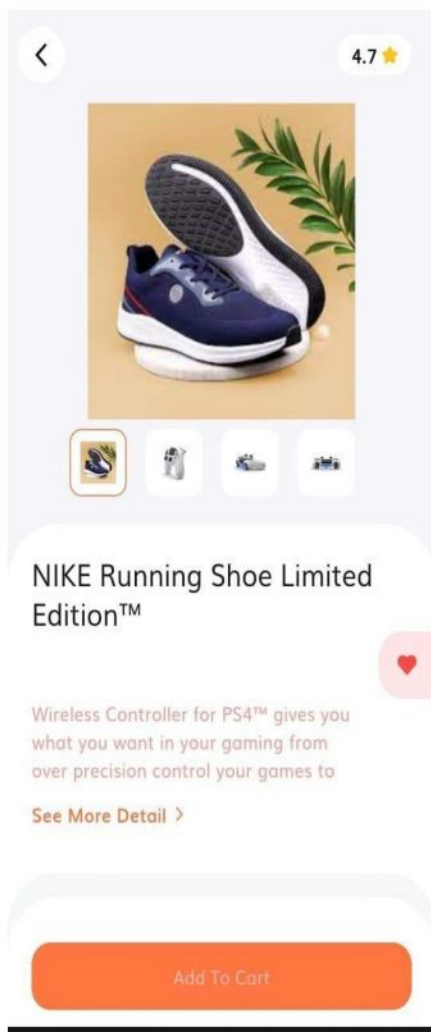


Fig 3.4 HomePage and mainPage

4.2 BACKEND RESU

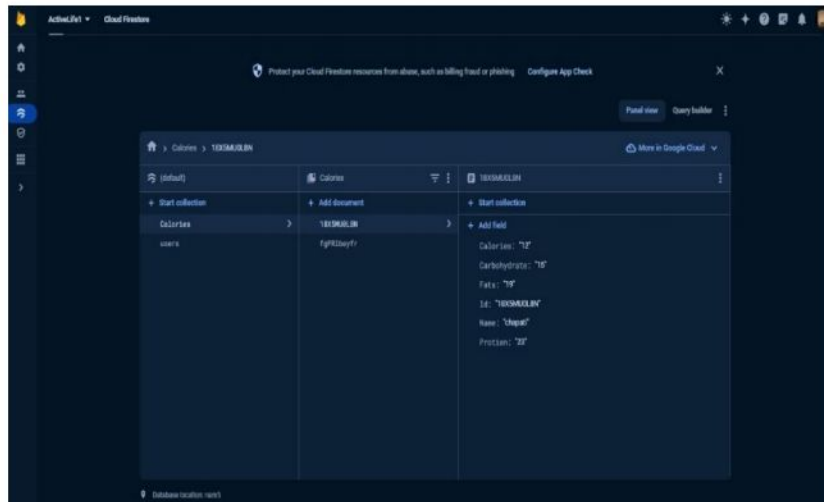


Fig 3.8 Firebase Calories Storage

The above image shows the firebase storage and it stores the calories entry data with unique id

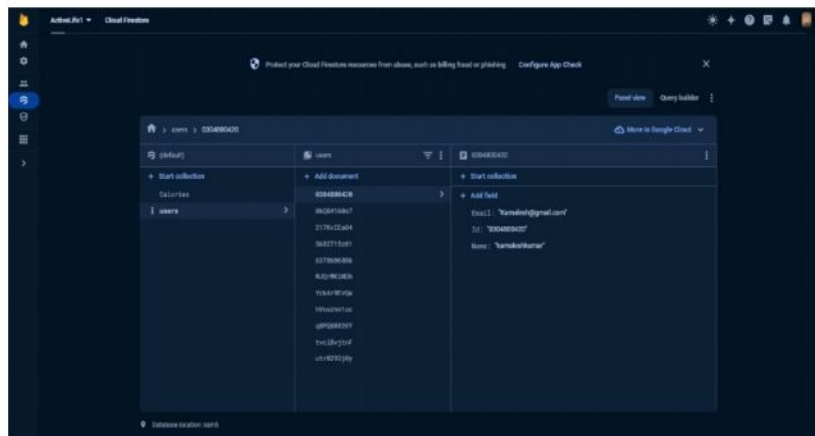


Fig 3.9 Firebase Users data Storage

The above image shows the firebase storage which is users data store by shared preference method its stored the username registered mail id

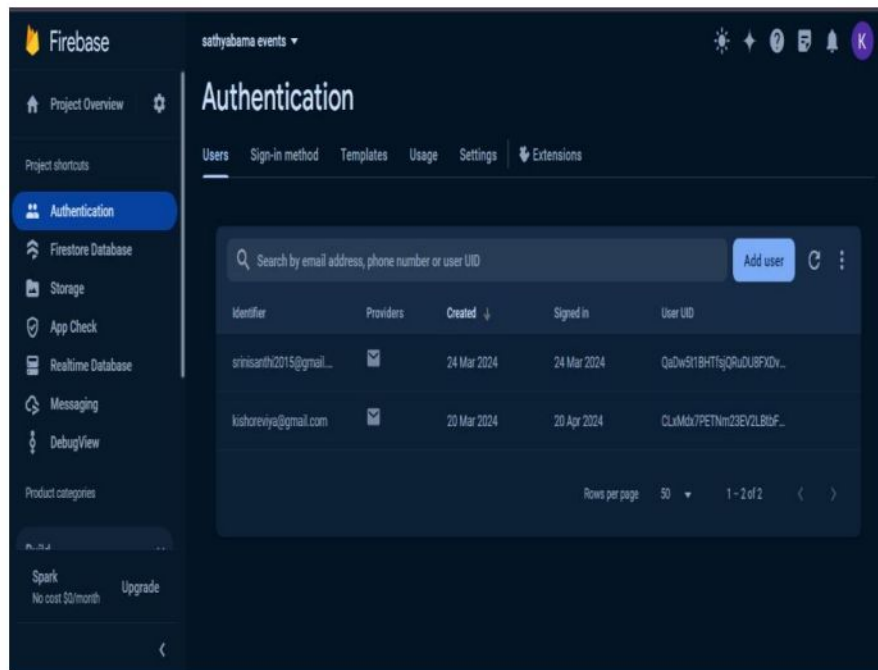


Fig 3.8 Auth page

The above image displays firebase authentication, where the above pic displays the valid accounts who signed up to app and only those 2 email ids in identifier column and specific passwords set are allowed to login to application

CHAPTER 4

CONCLUSION AND FUTURE WORKS

4.1 conclusion

In conclusion, the college events management app developed using Flutter and integrated with Firebase backend authentication serves as a dynamic and indispensable tool for organizing and participating in a diverse range of campus events. By offering robust features such as secure login authentication, comprehensive event management functionalities tailored to various event types, intuitive event categorization, and a plethora of user experience options including theme customization and event saving, the app enhances the overall event management experience for both organizers and attendees.

Furthermore, the app's emphasis on user support through comprehensive help resources, timely notifications, and seamless social interaction and engagement fosters a vibrant and inclusive community atmosphere. Additionally, its commitment to accessibility and usability ensures that all users, regardless of their abilities or preferences, can seamlessly navigate and engage with the app's functionalities.

Moreover, the incorporation of offline functionality and data synchronization capabilities guarantees uninterrupted access to critical event information and actions, even in low or no internet connectivity scenarios, thereby maximizing user convenience and productivity.

Overall, the college events management app stands as a testament

to the power of innovative technology in facilitating efficient event planning, fostering community engagement, and enriching the overall campus experience. With its user-centric design and robust feature set, the app is poised to become an indispensable tool for colleges and universities seeking to streamline their event management processes and enhance campus life.

4.2 FUTURE WORKS AND IMPLEMENTATION ISSUES

Enhanced User Profiles:

- Allow users to create detailed profiles showcasing their interests, skills, and past event participation.
- Implement a rating and review system for event attendees and organizers to provide feedback and improve future events.

Advanced Event Management:

- Introduce a calendar view to visualize upcoming events and deadlines.
- Implement a ticketing system for paid events, including payment processing integration.

Social Features:

- Incorporate social sharing functionality to allow users to share event details with their friends and contacts on social media platforms.
- Enable in-app messaging or discussion boards for event attendees to communicate and network before, during, and after events.

Integration with External Platforms:

- Integrate with popular event management platforms or university systems to synchronize event data and streamline event organization processes.

- Collaborate with local businesses or sponsors to promote events and offer exclusive discounts or perks to app users.

Gamification and Engagement:

- Implement gamification elements such as badges, achievements, or leaderboards to encourage active participation and engagement.
- Introduce interactive quizzes, polls, or surveys related to event topics to gather feedback and insights from attendees.

Offline app operation:

- In future update we will make the app to use local storage to store the specific app data to make it completely offline operatable and interactable app.

IMPLEMENTATION ISSUES

Implementing a college events management app using Flutter and integrating it with Firebase for backend login authentication can indeed bring various implementation challenges. Here's a brief overview of some potential issues you might encounter:

Firebase Integration: While Firebase provides robust authentication services, integrating it into your Flutter app might require careful attention to ensure seamless communication between your app and the Firebase backend. You may encounter challenges in setting up Firebase configurations, handling authentication flows, and managing user sessions.

Real-time Data Synchronization: Implementing real-time updates for event information, such as availability, schedule changes, or new event additions, can be complex. You'll need to utilize Firebase's real-time database to ensure that event data is synchronized across

all users' devices in real-time while maintaining data consistency and scalability.

User Experience Consistency: Providing a consistent user experience across different platforms (iOS, Android, web) can be challenging, especially when customizing themes and UI elements. Ensuring that your app's UX remains intuitive and user-friendly on all platforms might require additional effort and testing.

Error Handling and User Feedback: Handling authentication errors, network issues, and other edge cases gracefully is essential for a seamless user experience. You'll need to implement error handling mechanisms and provide clear feedback to users when issues occur, such as invalid login credentials, network timeouts, or server errors.

Linking and Navigation: Implementing clickable links to apply for events within the app requires careful navigation management. You'll need to ensure that users can seamlessly navigate between different screens, view event details, and initiate the application process without encountering navigation-related glitches or confusion.

Performance Optimization: As your app grows and handles a large volume of event data and user interactions, optimizing performance becomes crucial. You may need to optimize data fetching, rendering, and processing to minimize app loading times, reduce memory usage, and enhance overall responsiveness.

Testing and Quality Assurance: Thoroughly testing your app across various devices, screen sizes, and network conditions is essential to identify and address any implementation issues or bugs. Automated testing, user acceptance testing, and beta testing can

help ensure the reliability and stability of your app before release.

By addressing these implementation challenges proactively and leveraging Flutter's flexibility and Firebase's powerful features, you can develop a robust and user-friendly college events management app that meets the needs of both organizers and participants.

Issues using editors are given brief:

When developing Flutter apps using both Visual Studio Code (VS Code) and Android Studio, developers may encounter certain implementation issues unique to each IDE. Here are some common implementation issues and potential solutions for both environments:

Implementation Issues in Visual Studio Code (VS Code):

Dependency Management: Managing dependencies in VS Code can sometimes be challenging, especially when dealing with complex dependency graphs or conflicting versions. Developers may face issues related to package conflicts or outdated dependencies.

Solution: Regularly update dependencies in the pubspec.yaml file and use the flutter pub get command to ensure that all dependencies are up to date. Consider using the flutter pub outdated command to identify outdated packages and update them manually.

Performance Issues: In some cases, VS Code may experience performance issues, particularly when working with large Flutter projects or running multiple extensions simultaneously.

Implementation Issues in Android Studio:

Gradle Build Errors: Android Studio uses Gradle for building Flutter projects, and developers may encounter build errors or configuration issues related to Gradle scripts, dependencies, or plugin compatibility.

Solution: Resolve Gradle build errors by carefully reviewing the error messages and logs provided by Android Studio. Check the Gradle files (build.gradle and settings.gradle) for any syntax errors or misconfigurations and make necessary adjustments. Additionally, ensure that the Flutter and Gradle plugins are compatible with each other and up to date.

Emulator Performance: Running the Android emulator in Android Studio can sometimes be slow or resource-intensive, especially on systems with limited hardware resources.

Solution: Optimize emulator performance by configuring the emulator settings, such as adjusting the RAM size, CPU cores, and graphics acceleration. Consider using hardware acceleration (Intel HAXM or AMD Hypervisor) if supported by your system for improved emulator performance. Alternatively, test the app on physical devices for faster and more reliable results.

Integration with Flutter Tools: Android Studio provides built-in support for Flutter development, including features like hot reload, debugging, and widget inspection. However, developers may encounter issues with these tools due to compatibility issues or bugs.

Integrated Terminal: VS Code comes with an integrated terminal that allows developers to execute Flutter commands directly within the editor. This means you can run commands like flutter devices to list all connected devices without leaving the editor environment.

Flutter Device Selector: VS Code provides a device selector in the status bar, allowing you to quickly switch between connected devices for running and debugging Flutter apps. You can easily select your desired device from

the dropdown menu, making it convenient to test your app on different emulators, simulators, or physical devices.

Debugging Tools: VS Code offers powerful debugging tools for Flutter apps, including breakpoints, watch expressions, and variable inspection. You can set breakpoints in your code and debug your app step-by-step on connected devices, gaining insights into app behavior and identifying issues more efficiently.

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.

Internet Articles:

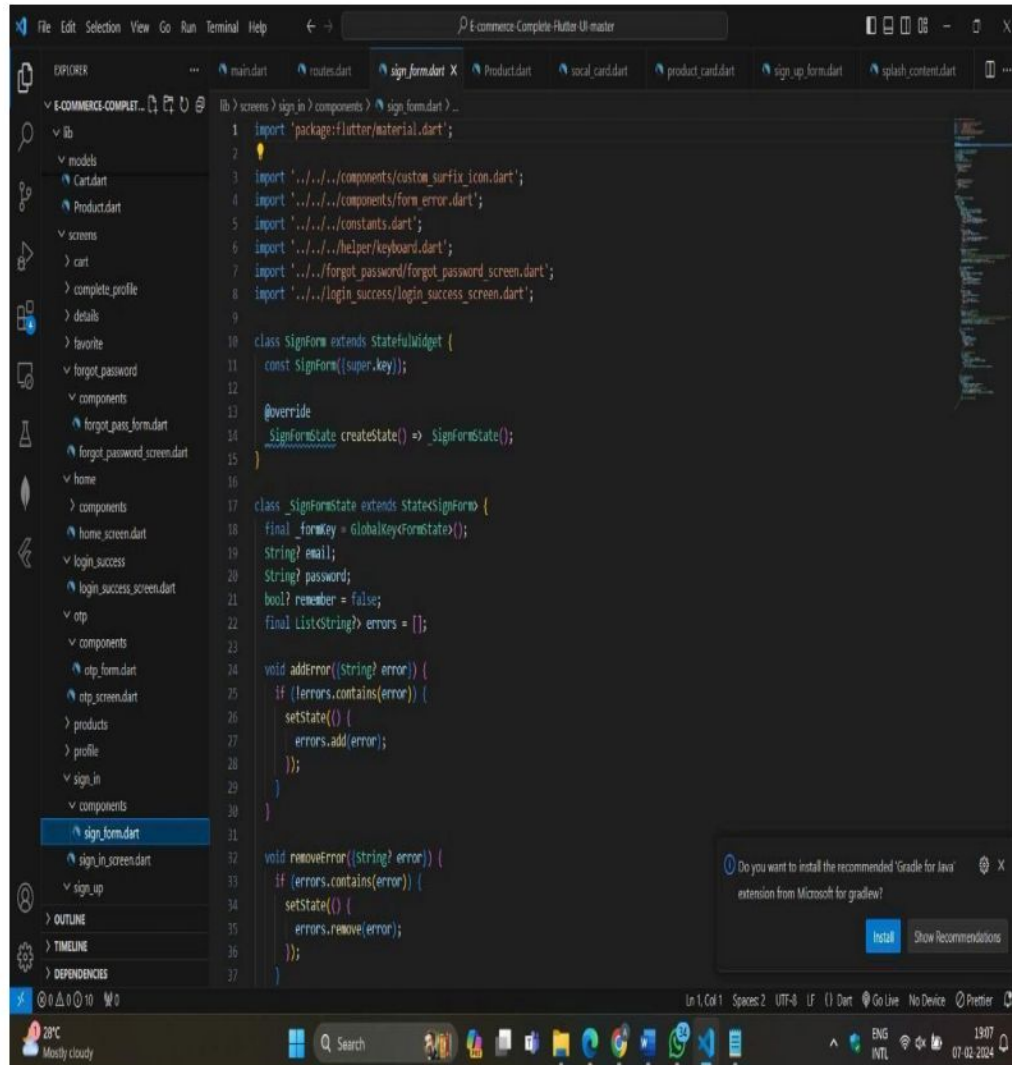
[1] D. Camp, "Best flutter blog articles from around the web," Flutter articles, <https://www.devglan.com/programming/flutter-article-feeds> (accessed Apr. 25, 2024).

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

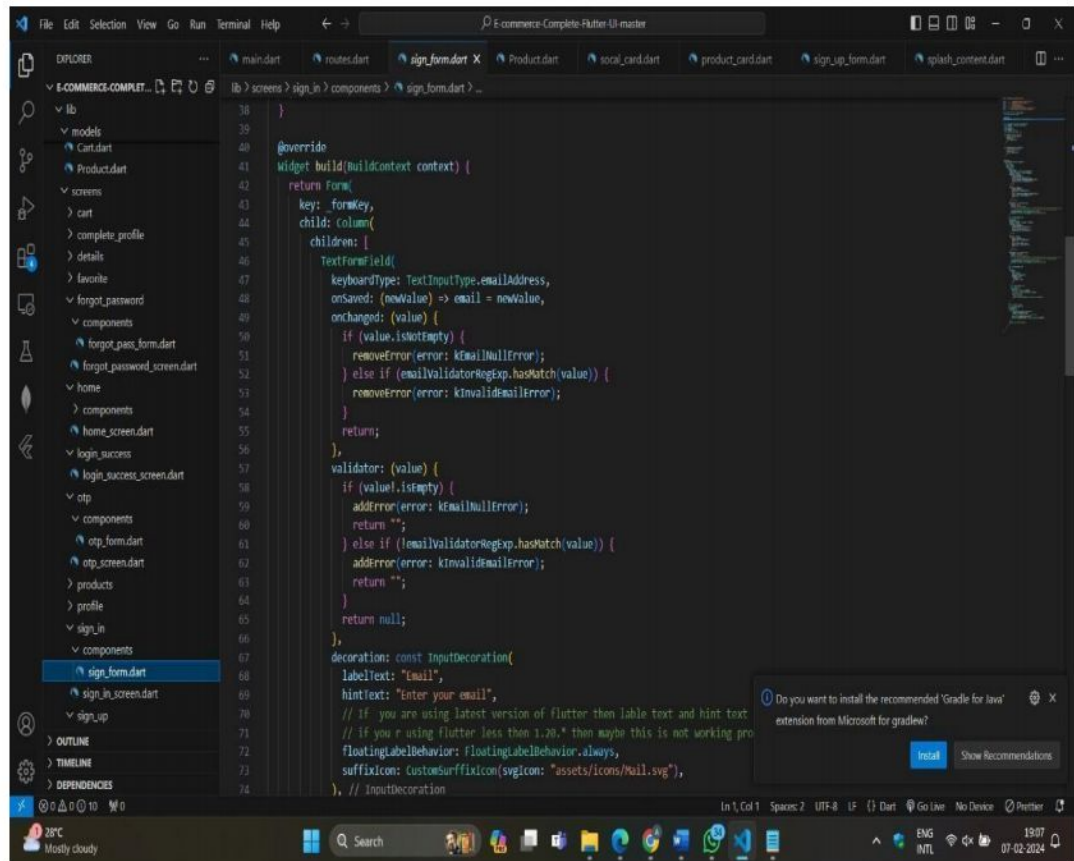
APPENDICES

A.SAMPLE CODE



```
1 import 'package:flutter/material.dart';
2
3 import '../components/custom_surfix_icon.dart';
4 import '../components/form_error.dart';
5 import '../constants.dart';
6 import '../helper/keybord.dart';
7 import '../forgot_password/forgot_password_screen.dart';
8 import '../login_success/login_success_screen.dart';
9
10 class SignForm extends StatefulWidget {
11   const SignForm({super.key});
12
13   @override
14   SignFormState createState() => _SignFormState();
15 }
16
17 class _SignFormState extends State<SignForm> {
18   final _formKey = GlobalKey<FormState>();
19   String? email;
20   String? password;
21   bool? remember = false;
22   final List<String> errors = [];
23
24   void addError(String? error) {
25     if (!errors.contains(error)) {
26       setState(() {
27         errors.add(error);
28       });
29     }
30   }
31
32   void removeError(String? error) {
33     if (errors.contains(error)) {
34       setState(() {
35         errors.remove(error);
36       });
37     }
38   }
39 }
```

1. Main.dart



2.pubspec.yaml

Homepage and Profilepage

← Sign In

Welcome Back

Sign in with your email and password
or continue with social media

Email

Enter your email 

Password

Enter your password 

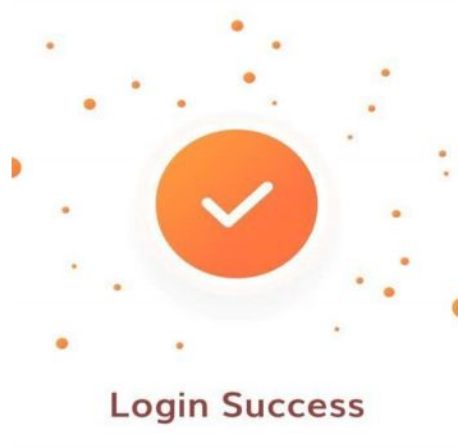
☐ Remember me [Forgot Password](#)

Continue

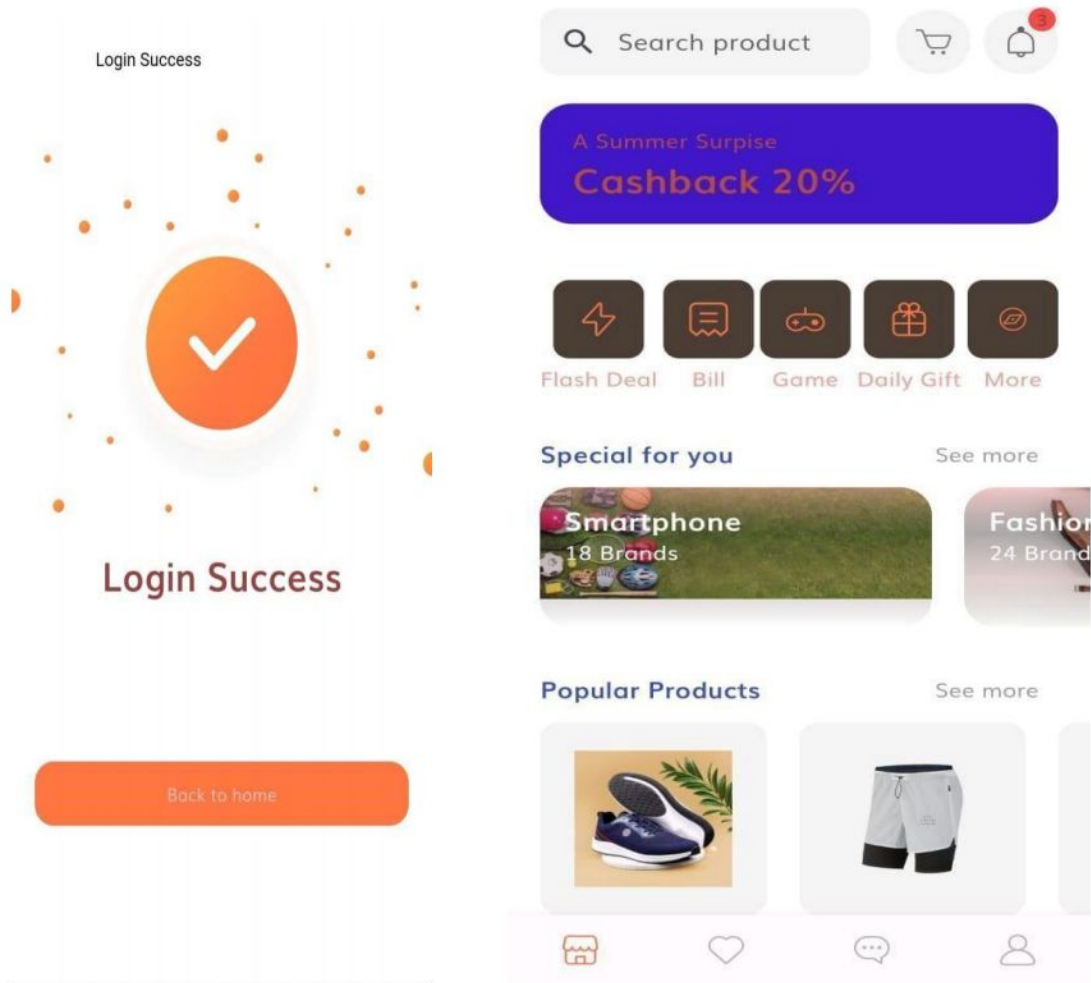


Don't have an account? [Sign Up](#)

Login Success



Back to home



Login page and shopping products