

SATHYABAMA INTERCOLLEGE EVENTS APP

INTERDISCIPLINARY PROJECT

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

by

KISHORE S (Reg.no-41120095)



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
KISHORE S (41120095) who carried out the project entitled
"Sathyabama intercollege events app" under my supervision
from January 2024 to May 2024.

Internal Guide

Dr.S. URMELA, M.Tech.,Ph.D.,

Head of the Department

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

Submitted for Viva voce Examination held on 08-05-24

Internal Examiner

8/5/24

External Examiner

DECLARATION

I Kishore S(41120095) hereby declare that the Project Report entitled "Sathyabama intercollege events app" done by me under the guidance of Dr.S.URMELA, M.Tech., 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

S.Kishore

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.S. URMELA,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 **Kishore S (Reg No: 41120095)** 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, **Kishore S** has closely worked as a part of the **Flutter** by carrying out the project entitled "**Sathyabama Intercollege Events Site**".

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

College events management application meticulously crafted using the versatile Flutter framework, harmoniously fused with Firebase for fortified user authentication and dynamic data management. The app represents a pinnacle in event organization and facilitation within academic environments, offering an all-encompassing platform to orchestrate a myriad of campus activities. From informative workshops to sought-after internships, vibrant cultural celebrations, and adrenaline-fueled hackathons, the app caters to diverse interests and pursuits prevalent in collegiate life. Harnessing the agility and robustness of Flutter, a cross-platform development framework renowned for its fluidity and performance, the application guarantees a seamless user experience across a spectrum of devices and operating systems. Its intuitive interface and responsive design ensure accessibility and ease of navigation, enhancing user engagement and satisfaction. Key features of the application include comprehensive event creation and management capabilities, empowering organizers to curate compelling experiences with detailed event descriptions, schedules, and participant criteria. Seamless integration with Firebase facilitates secure user authentication, enabling smooth registration and login processes while safeguarding user data and privacy.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vii
	LIST OF FIGURES	x
1	INTRODUCTION	1
	1.1 Problem Statement	2
	1.2 Objectives	2
	1.3 Flutter and firebase overview	3
2	METHODOLOGY	6
	2.1 Architecture setup of app	6
	2.2 Deployment	10
	2.3 Key challenges	11
	2.4 Modules	12
	2.4.1 Login-Authentication UI	12
	2.4.2 Homepage event navigation	13
	2.4.3 Event view and register	15
	2.5 Algorithm	16
	2.6 Requirement analysis	18
	2.7 Dependency analysis and plugin used	19
3	RESULTS AND DISCUSSION	21
	3.1 Frontend results	22
	3.2 Backend results	25
4	CONCLUSION AND FUTURE WORKS	28
	4.1 Conclusion	28

4.2	Future works and Implementation issues	29
	REFERENCES	35
	APPENDICES	36
A	Sample code	36
B	Screenshots	38

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
2.1	Architecture diagram	6
2.2	Firebase flow diagram	8
3.1	Welcome page and signup	22
3.2	Chatbot and event page	23
3.3	Home page	23
3.4	Event add and Settings	24
3.5	Help page	25
3.6	Firestore	25
3.7	Firebase storage	26
3.8	Authentication page	27

CHAPTER 1

INTRODUCTION

In this sathyabama intercollege events app we will prior focus on student to discover, and participate in campus activities is made effortlessly seamless and engaging. This introduction unveils a revolutionary application meticulously engineered using the Flutter framework and powered by Firebase, designed exclusively to streamline the orchestration of diverse college events. From educational workshops to cultural extravaganzas, internships, and hackathons, this app redefines the landscape of campus engagement, offering a comprehensive platform that caters to the multifaceted interests and passions of today's students. Embracing the cross-platform capabilities of Flutter, the application ensures a consistent and immersive user experience across a myriad of devices and operating systems. By seamlessly integrating with Firebase, it leverages robust authentication protocols to safeguard user data and privacy while facilitating frictionless login and registration processes. With its intuitive interface and dynamic features, the app empowers both event organizers and participants, fostering a vibrant and inclusive campus culture where collaboration, innovation, and exploration thrive. The innovative features and transformative potential of this cutting-edge college events management application, poised to revolutionize the way campus events are organized, . Despite the plethora of activities available, students and organizers alike face hurdles in efficiently discovering, coordinating, and participating in these events. In this context, the need arises for a comprehensive and user-friendly college events management application

1.1 PROBLEM STATEMENT

The aim of this project is to build an app using flutter for the organization and management of diverse events, spanning workshops, internships, cultural festivals, non-technical gatherings, and hackathons, often pose significant logistical challenges. Despite the plethora of activities available, students and organizers alike face hurdles in efficiently discovering, coordinating, and participating in these events. In this context, the need arises for a comprehensive and user-friendly college events management application, seamlessly integrated with Firebase for secure user authentication, to streamline the process of event organization and participation.

1.2. OBJECTIVES

The primary objective of the college events management application developed using Flutter and integrated with Firebase for login authentication is to provide a user-friendly and comprehensive platform for efficiently managing and participating in a diverse range of campus events, including workshops, internships, cultural events, non-technical gatherings, and hackathons. This app aims to streamline the event organization process for administrators and organizers while enhancing the event discovery and participation experience for students. Key goals include enabling secure user authentication, facilitating seamless event creation and management, promoting community engagement through social integration, and leveraging analytics to optimize event planning and user satisfaction. Ultimately, the objective is to revolutionize campus event coordination, fostering a vibrant and inclusive campus culture.

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 improves the performance of the application apk

Single Codebase: With Flutter, developers can write a single codebase for both iOS and Android platforms, reducing development time and efforts are made in easetime

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 for buttons and drawer menu within appui

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 into the application.

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 and also supports web application.

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.

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:

Lightweight: VS Code features offers 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.

Android Studio:

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.

Interactive environment: 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: We will use firebase storage to store the image(.jpg) from the user input for the event details fields and we use the concept of URL rewriting to get url of the image and download it and store it in firestore and then we use 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 to another page where user can fill up the registration for the events . the firebase_core package is imported to initialize Firebase services in the app.

CHAPTER 2

METHODOLOGY

2.1 ARCHITECTURAL SETUP OF APP

Fig2.1 refers to architecture diagram that explains flow of the app simulation starting from login authentication ,then apk building happens followed by nodejs for runtime and firebase cloud to store user details and image data.

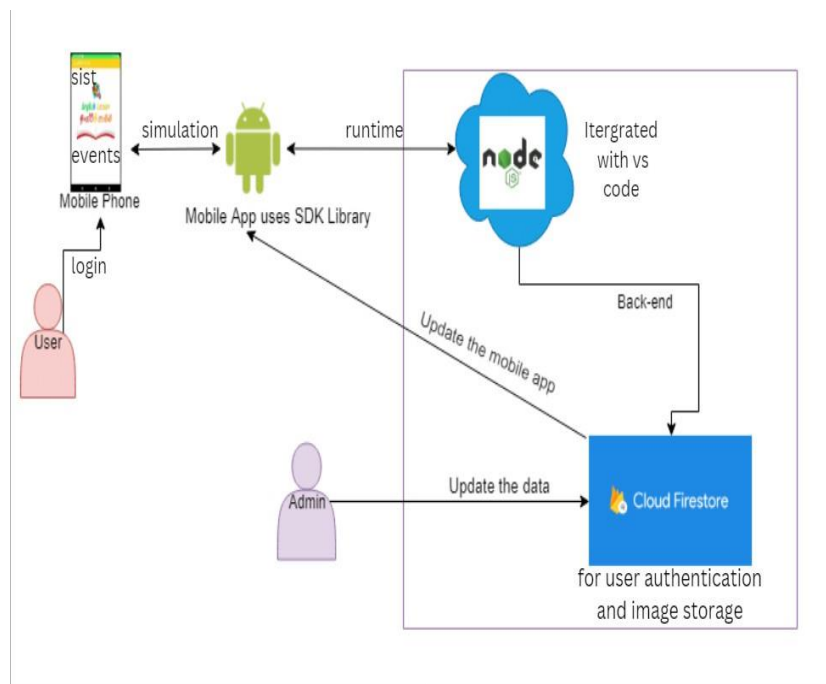


Fig 2.1 Architecture diagram

User interface: Designed the events app user interface using Flutter's widget system, creating screens for displaying events, adding new events, and managing existing ones. To display event posters, you're likely using the `Image.network()` widget, which fetches images from a given URL and displays them in the app and the UI is designed in such a way that user can easily register to events from different section and can get help from.

Event Management:Users can add new events through a form where they input details like event title, description, date, and upload an event poster.The app utilizes Firebase's `downloadURL()` function to retrieve the URL of the uploaded poster from Firebase Storage after it's successfully uploaded.

Firebase Integration:Firebase Authentication is used for user authentication, ensuring that only authenticated users can access the app's features.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.Firebase 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.

Networking:The `Image.network()` widget fetches event poster images directly from the URLs stored in Firestore, enabling users to view posters associated with each event.This approach allows for dynamic loading of images, ensuring that the app can efficiently handle a large number of events with their associated posters.The image displayed is directly retrived from firebase store.Actual image URL is retrieved and the metadata is displayed as output

Firebase flow: Fig2.2 refers to explains firebase integration flow for storing the images of the events and user details and also give googleanalytics for debugging the errors.

Nodejs environment:Nodejs supports the CLI commands in ths visual studio code editor so that we can get connected to firebase by using few lines of commands and rest CLI commands will lookover to add all the necessary api key and firebase dart file and json file

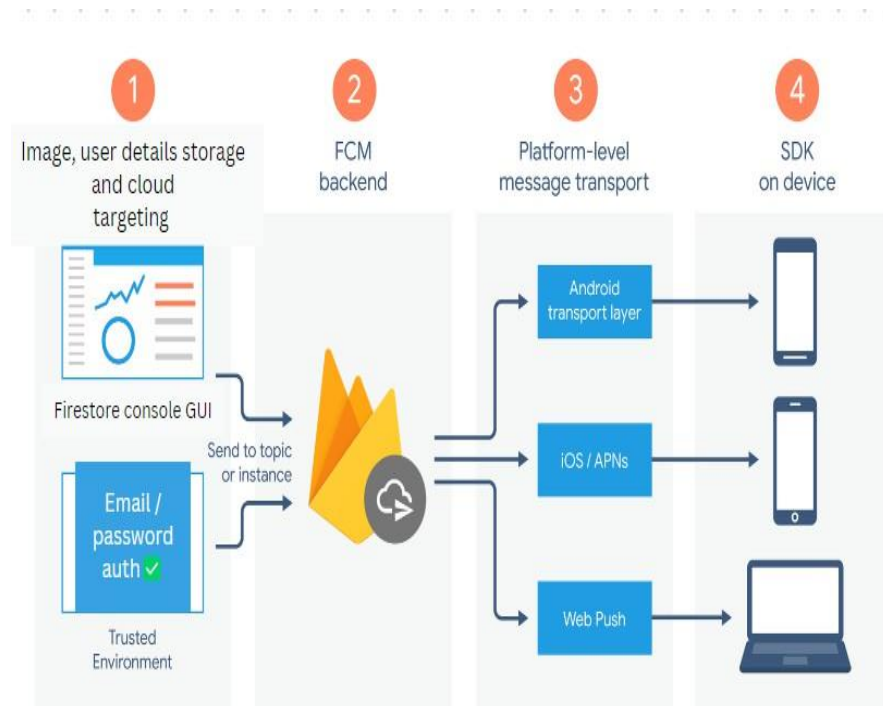


Fig 2.2 Firebase flow diagram

Firestore Authentication: Handles user authentication, allowing users to sign up, sign in, and securely access the app's features.

Cloud Firestore: Stores event data, including titles, descriptions, dates, and poster URLs. Each event might be represented as a document within a Firestore collection. Integrates with the frontend to provide real-time syncing of data, enabling immediate updates to event information across all instances of the app.

Firestore Storage: Stores event posters uploaded by users. Each poster is associated with a unique URL generated by Firestore Storage. we will use it store categories of event images with the specific names and store it in jpg format.

Security Rules: Firestore Security Rules are configured to ensure that only authenticated users can read and write event data in Firestore firebase and upload posters to Firestore Storage, maintaining data security and integrity.

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: Explored the design and functionality of event management applications across various domains, including corporate events, festivals, and community gatherings. 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: The adoption of Flutter, a cross-platform development framework known for its flexibility and performance, has gained traction in recent years. 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: Studies focusing on user experience design principles and methodologies underscore the importance of user-centered design approaches in creating engaging and intuitive mobile applications. Key considerations include accessibility, visual aesthetics.

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. 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 and 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.

Apk build: The apk for the application is build directly through the physical cable for the testing purposes ,but during production of the application it will deployed in goolge playstore

2.3 KEY CHALLENGES

Diverse Event Management: Effectively managing a wide array of events, including workshops, internships, cultural events, non-technical gatherings, and hackathons, requires a flexible and scalable architecture capable of accommodating varied event types and requirements.

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.

Event Organization and Coordination: Streamlining the event creation, scheduling, and coordination process for administrators and organizers presents challenges in terms of usability, efficiency, and scalability. Providing intuitive tools for event management while accommodating diverse organizational.

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.

Analytics and Insights: Developing robust analytics tools to gather insights into user behavior, event attendance, and feedback requires careful data collection and analysis. Balancing user interaction with maintaining a positive and inclusive community environment is essential.

2.4 MODULES

The application's UI encompasses three main sections: Login/Authentication, Homepage/User Navigation, and Event View and Registration. 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. 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.

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 dashboard showcases various events categorized into five sections through card views, providing users with easy access to information about hackathons, internships, cultural events, and more. Each card displays essential details such as event title, date, and a brief description, facilitating event discovery and exploration. Users can seamlessly navigate through the dashboard to find events of interest and stay updated on upcoming opportunities.

Menu Drawer: The menu drawer prominently displays the user's email and name at the top, offering a personalized touch and easy identification. It provides quick access to essential features such as settings, allowing users to manage their account preferences, logout, reset passwords, and update profile information. Additionally, the menu drawer features a chatbot option for instant assistance, ensuring users have access to support whenever needed without interrupting their browsing experience.

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.

Chatbot Integration:The app seamlessly integrates a chatbot feature to provide instant support for event-related queries and technical issues users can interact the application with chatbot to seek assistance, ask questions about events, get troubleshooting help, or receive guidance on using app features the chatbot enhances user experience by offering real-time responses and solutions, reducing the need for manual intervention and improving overall support accessibility.

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 app's enhancements focus on improving event discovery, user navigation, and support accessibility, providing a seamless and user-friendly experience.By streamlining features such as the event dashboard, menu drawer, profile page, chatbot integration, and help page, the app aims to cater to users' needs effectively and enhance their engagement and satisfaction and further the app will be enhanced with realtime student support from a specific timing in a week and also the will made easy to use even through the offline mode so that student can view the events and its related details in ease of user experience

2.4.3 Event view and register

Comprehensive Event Information:The event page displays detailed information about each event, including title, date, time, location, description, and registration details.Users can easily access all pertinent information they need to decide whether to attend the event, ensuring transparency and clarity.

Eye-Catching Event Poster QR Code:eye-catching event poster QR code is prominently featured on the event page, allowing users to register for the event conveniently.By scanning the QR code with their smartphones, users are redirected to the event registration page, streamlining the registration process.

Event apply logic: apply button provides users with a quick and straightforward way to express their interest in attending the event.Upon clicking the apply button, users are redirected to a Google Docs registration form, where they can provide their registration details efficiently.

Event Poster Image:event poster image, fetched from Firestore, is prominently displayed on the event page, captivating users' attention and providing visual context for the event.Users can easily view the event poster to get a glimpse of the event's theme, speakers, or highlights, enhancing their interest and engagement.

Save Button:The save button allows users to bookmark or save events they're interested in, ensuring they can easily access them later.

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.

Event Schedule:The event schedule section outlines the agenda and timeline for the event, including keynote speakers, panel discussions, workshops, and networking sessions.Users can plan their participation in the event by referring to the schedule and identifying sessions of interest.

Event Updates:Users receive real-time updates and notifications about event-related announcements, such as changes in schedule, venue, or additional speakers. 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.If outdated package exist run `flutter pub outdated` and upgrade it.

3. Firebase Setup:Create a new Firebase project in the Firebase console.Follow the setup instructions to add your Flutter app.

4. Firebase Authentication: Using firebase Authentication to enable a secure login and registration using email and password. Provide options for users to sign up or log in using email/password.

5. Firebase Firestore Integration: Use Firebase Firestore to manage event data, including posters and user details. 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: Design intuitive UI with interactive components for seamless user experience. Utilize card views and filters for event categories, ensuring a visually appealing and user-friendly interface. Implement responsive design principles to ensure compatibility across various devices and screen sizes.

7. Event Details and Registration: Allow users to view event details, including title, date, description, and poster of the event and implement interactive elements such as buttons for registration, enabling users to express interest in attending events.

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

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 with the 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.for the app is recommended with 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 will be done in firebase for user details and event details from admin side for adding events to the application.

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 features, notifications in future hand, 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.

2.7 DEPENDENCY ANALYSIS AND 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.

Firestore_auth: Firestore Authentication plugin allows easy integration of Firestore 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.

Firestore_auth_web: Firestore Authentication plugin for web platforms, ensuring seamless authentication experience across different devices and platforms.

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

Firestore_storage: Firestore Storage plugin enables the app to interact with Firestore Cloud Storage. It provides methods to upload, download, and manage files in Firestore 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 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. Toast messages are used for lightweight notifications

CHAPTER 3

RESULTS AND DISCUSSION

The resultant college events management app, developed using Flutter and integrated with Firebase for secure login authentication, has successfully transformed the organization and participation of campus events. With a diverse range of functionalities catering to workshops, internships, cultural events, non-technical gatherings, and hackathons, the app offers a comprehensive solution for students, organizers, and participants.

Secured user experience: Firebase authentication employs industry-standard encryption protocols for safeguarding user credentials ensuring a secure login process. Multi-factor authentication options add an extra layer of security, providing users with peace of mind regarding their account safety.

Event Management: Organizers can set event visibility options, allowing them to control who can view and participate in their events. Collaborative event management features enable multiple organizers to collaborate on event planning and execution, fostering teamwork and efficiency.

Event Categorization: Users can filter events based on criteria such as location, date, and event type, facilitating targeted event discovery. Customizable search filters enable users to refine their event search based on specific preferences, ensuring relevant results are obtained and an Interactive troubleshooting guides walk users through common issues step-by-step, empowering them to resolve issues independently.

3.1 FRONTEND RESULTS:

Frontend result of the sathyabama events app refers to the user interface ,design and real-time user experience with the app ,it includes login in the users in realtime authentication with backend and also registering for the and can view the profile and edit the settings and get help from chatbot and also users can view help page for more support , they can see the upcoming events through the official website and also redirection to lms support.

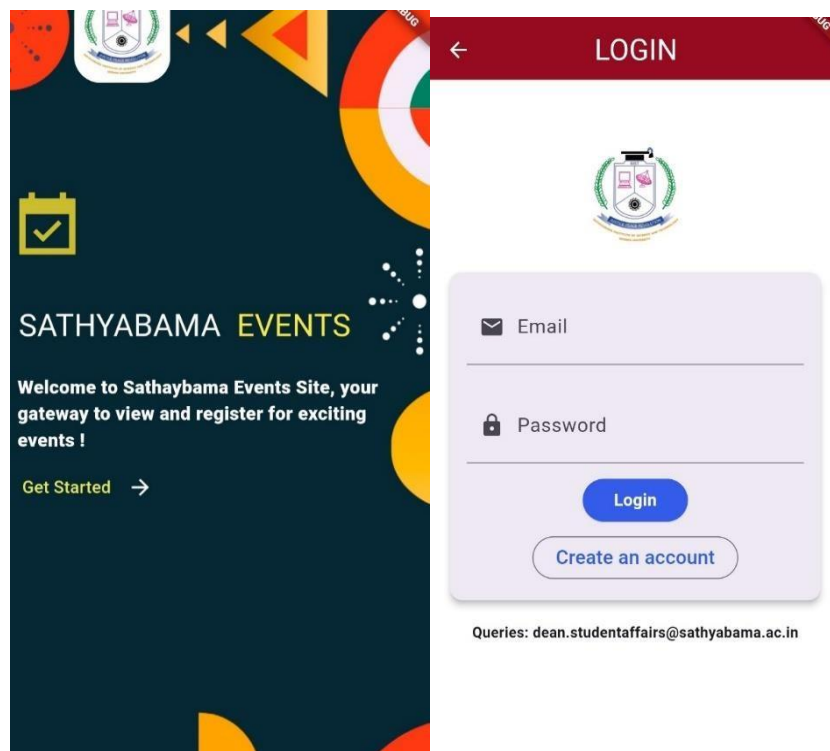


Fig 3.1 Welcome page and signup

Fig3.1 explains starting page of the app ,where the users click get started button to get into login page and user can login if they have account or else signup .

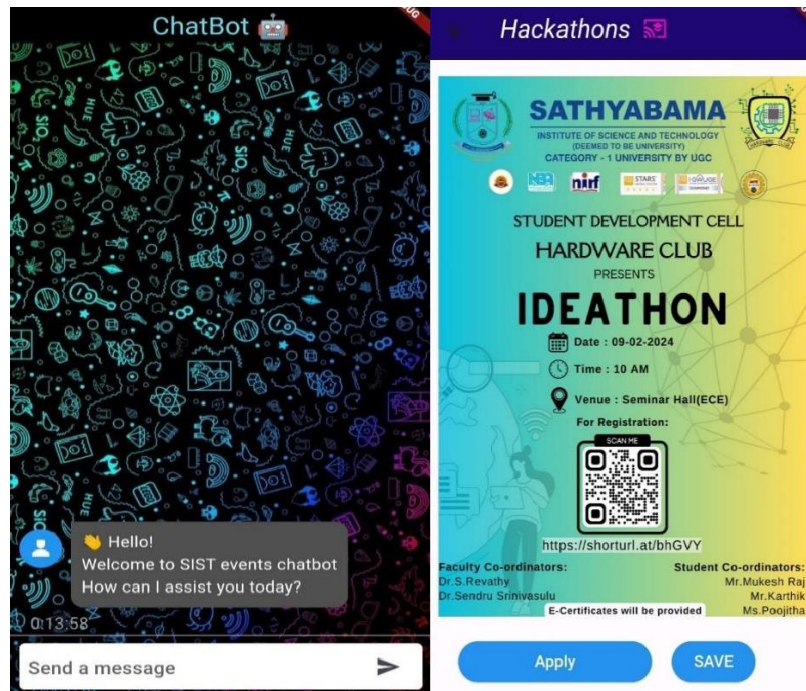


Fig 3.2 Chatbot and event page

Fig 3.2 explains the chatbot support for user to get queries cleared and another sample event application page

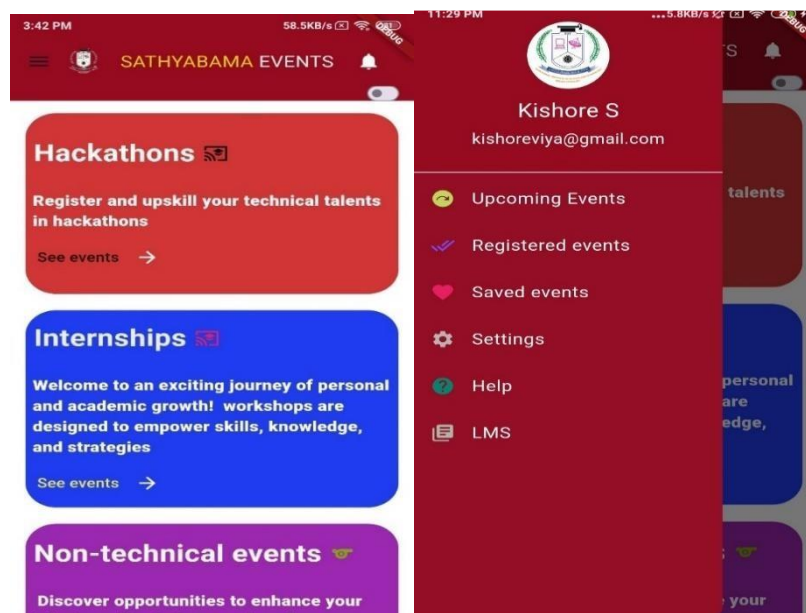
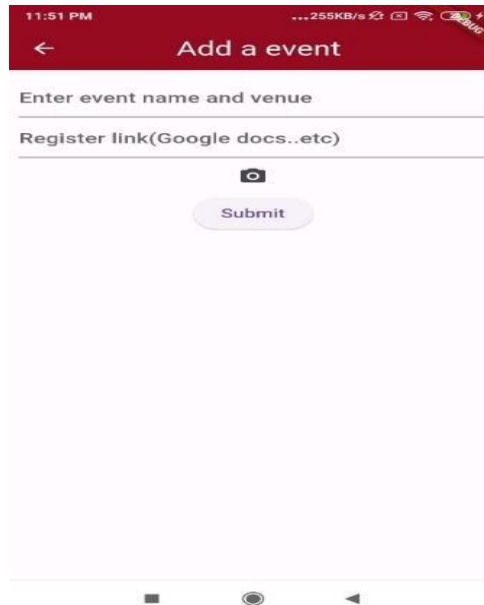


Fig 3.3 Home page

Fig 3.3 explain the app page with list of events section like hackathon,internships,workshops..etc and can click see events and register events and the drawer shows the user with help,LMS link, and settings to configure and getting quick help from chatbot and users can logout



The app page shows the input fields only for admins to add an event with name, venue and register link.

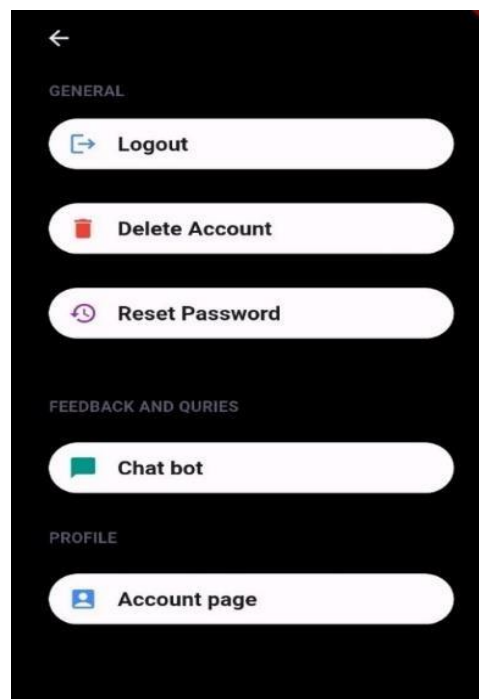


Fig 3.4 Event add and Settings

Here where the descriptive part of Settings is shown above where user can also reset password, delete account and can logout also where users can view there profile page and also add images to it, user can click on menu and access help and only admin can add the events. from add event page



Fig 3.5 explains the help page of the app for user feedback and support.

4.2 BACKEND RESULTS

Here we will see the back data storage such as user details and event images results of the application

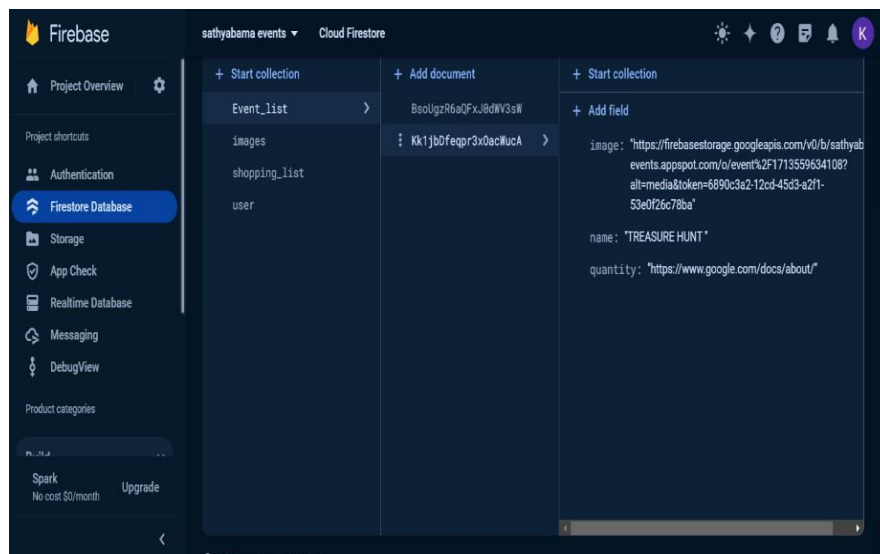


Fig 3.6 Firestore

Fig3.6 explains the firebase firestore which is used to store user details and event posters url along with event name ,venue, register link

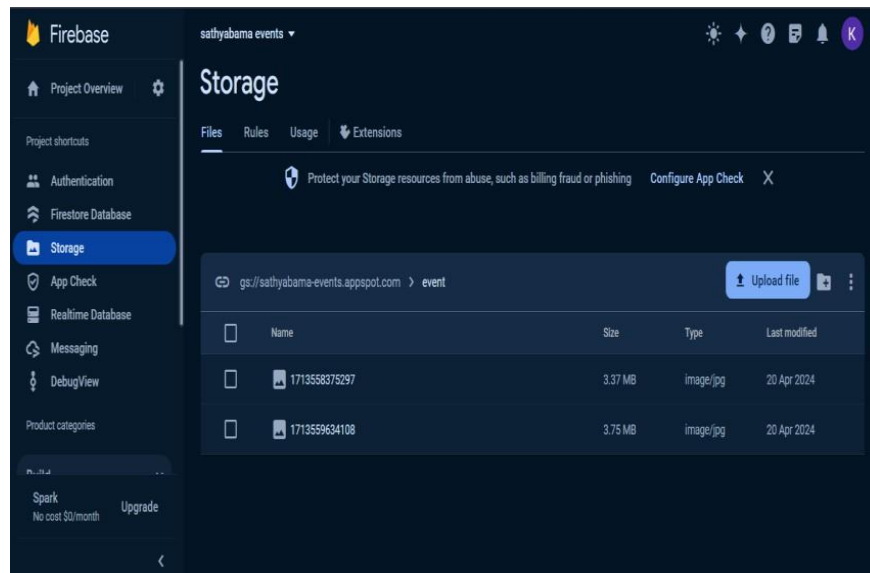
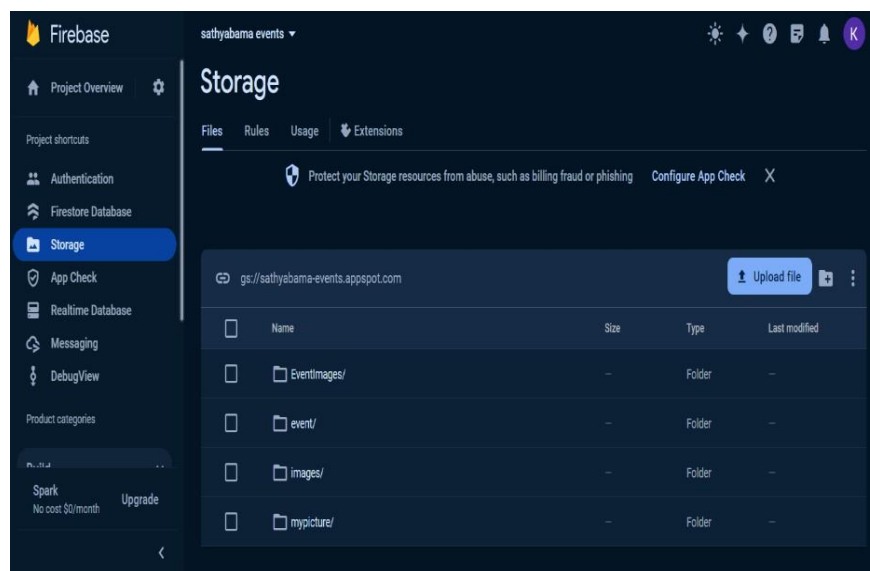


Fig 3.7 Firebase storage

Fig 3.7 explains the firebase storage which is used to store the respective event poster image in jpg format



The above image shows the firebase storage ,here we will create different bundle (/event,/image..etc)to store the different category of event poster

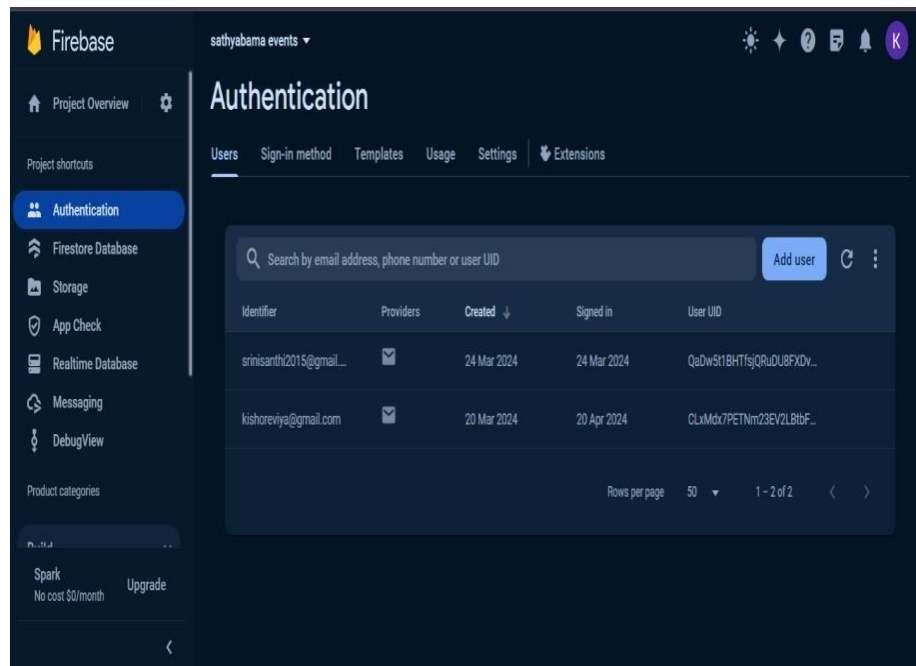


Fig 3.8 Authentication page

Fig 3.8 explains the firebase authentication, where the user can create account that appears here and valid accounts who signed up to app and only those 2 email ids in identifier column are allowed to login to the app

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 their 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 and users can deploy and build their apps in iOS and web app also with firebase support it becomes easy task for users build multi-platform app.

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 pf the application enhances the user interaction with the app and gives good experience.

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. Often build gradle runs to error if user made any changes to kotlin versions or google:gms version it may raise error if the changes are depreciated.

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.

Flutter DevTools Integration: VS Code seamlessly integrates with Flutter DevTools, a suite of performance and debugging tools for Flutter apps. With a single click, you can launch DevTools directly from VS Code, allowing you to analyze performance profiles, inspect widget trees, and diagnose layout issues across connected devices.

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
- [4] J. Smith, "Implementing State Management in Flutter," Advanced Flutter Techniques, pp. 75-92, Aug. 2022. doi: 10.1109/AF.2022.1234567

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.
- [4] B. Smith, Pro Android App Development: Build Real World Apps with Firebase, Flutter, and More. New York, NY: O'Reilly Media, 2020.

APPENDICES

A.SAMPLE CODE



The screenshot shows an IDE window titled 'sathyabama_Events_App'. The file explorer on the left lists several Dart files: 'main.dart', 'add_item.dart', 'add_item1.dart', 'edit_item.dart', 'edit_item1.dart', and 'item_details.dart'. The 'main.dart' file is open in the editor, displaying the following code:

```
import 'firebase_options.dart';
import 'package:basic_utils/basic_utils.dart';
import 'package:firebase_app_check/firebase_app_check.dart';
import 'package:firebase_app_check_platform_interface/firebase_app_check_platform_interface.dart';
import 'package:firebase_auth/firebase_auth.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  WidgetsFlutterBinding.ensureInitialized();
  await FirebaseAppCheck.instance.activate();
  await Firebase.initializeApp();

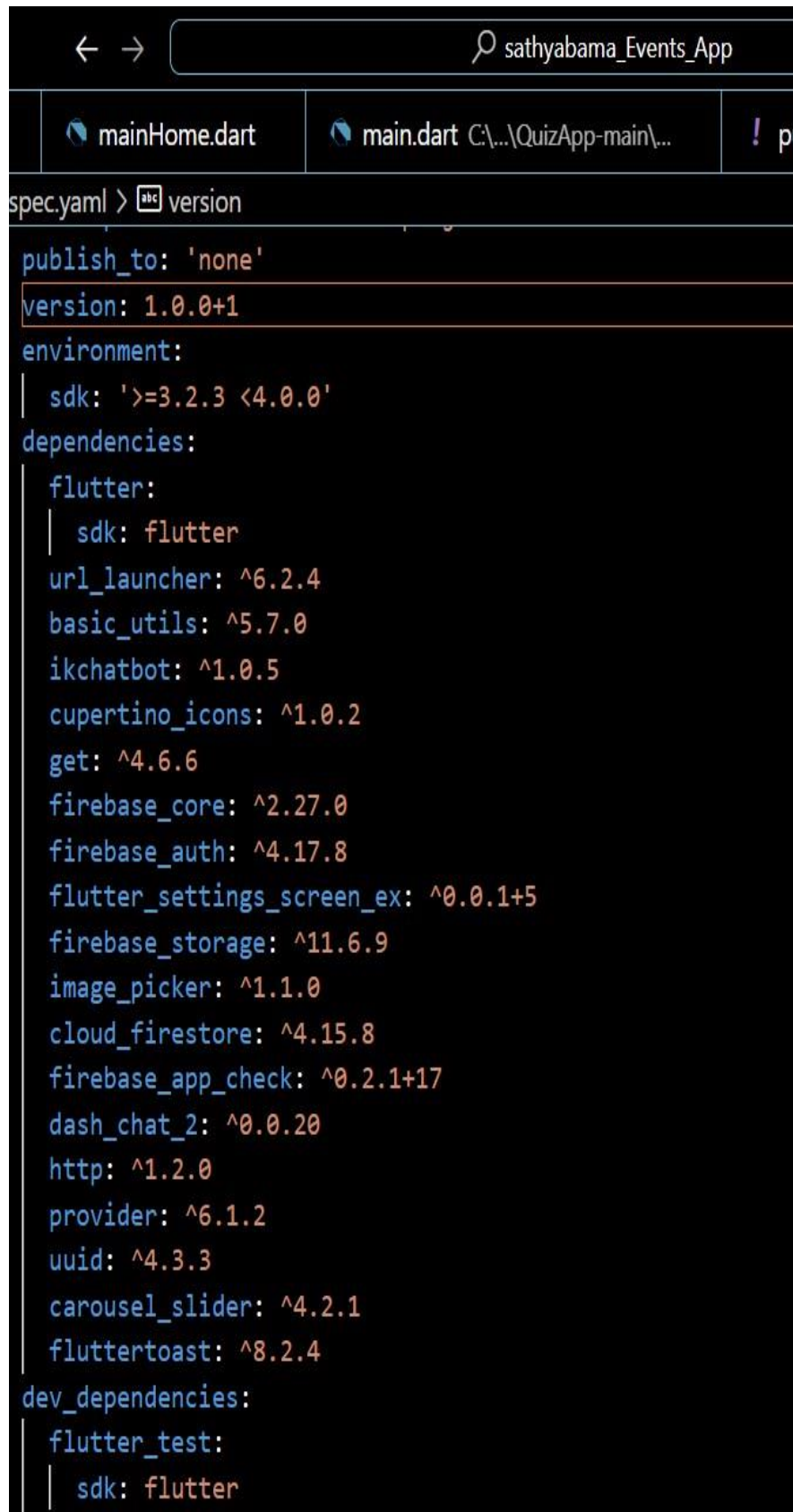
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key});

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider<EventRegImg>(
          create: (_) => EventRegImg(), // Add your provider here
        ),
      ],
      child: GetMaterialApp(

```

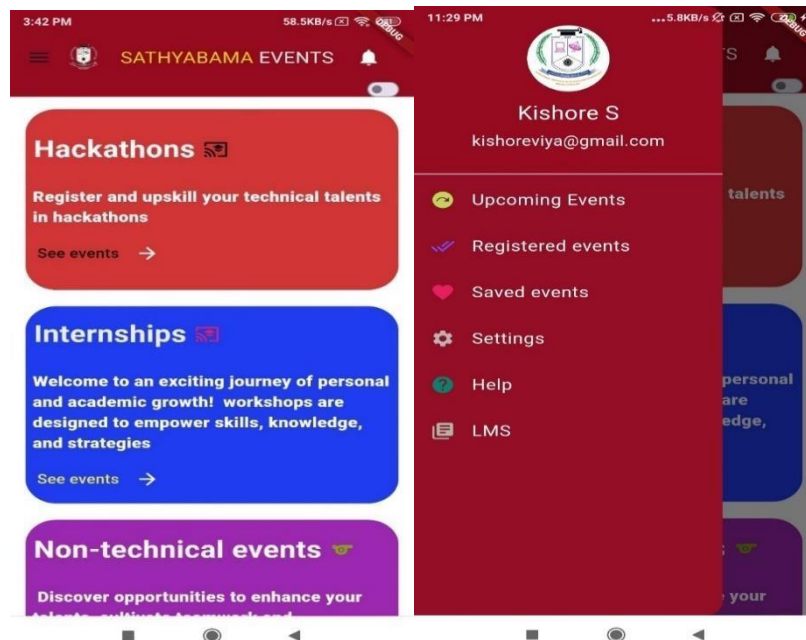
1. Main.dart



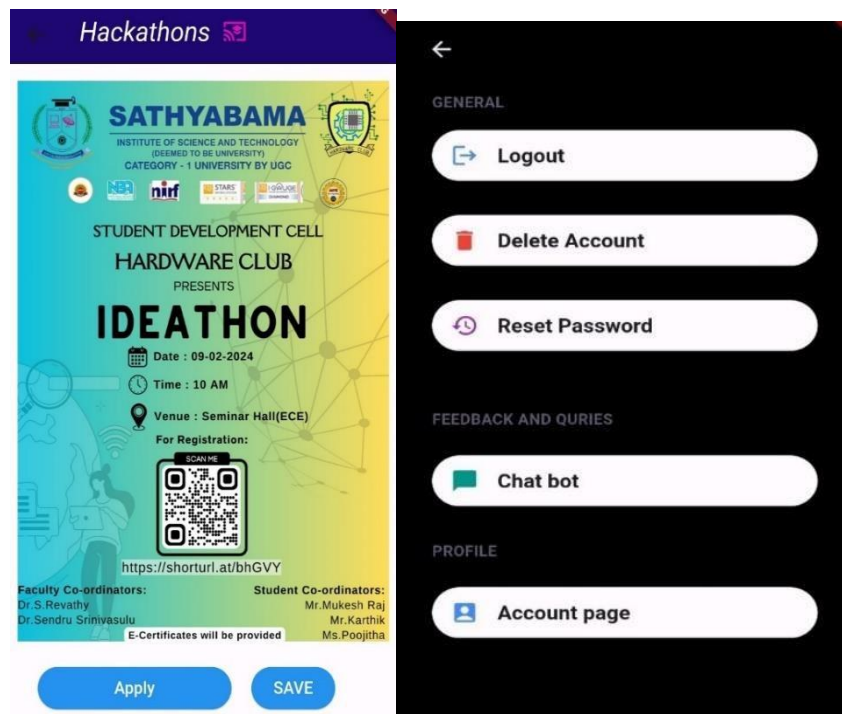
```
spec.yaml > version
publish_to: 'none'
version: 1.0.0+1
environment:
  sdk: '>=3.2.3 <4.0.0'
dependencies:
  flutter:
    sdk: flutter
  url_launcher: ^6.2.4
  basic_utils: ^5.7.0
  ikchatbot: ^1.0.5
  cupertino_icons: ^1.0.2
  get: ^4.6.6
  firebase_core: ^2.27.0
  firebase_auth: ^4.17.8
  flutter_settings_screen_ex: ^0.0.1+5
  firebase_storage: ^11.6.9
  image_picker: ^1.1.0
  cloud_firestore: ^4.15.8
  firebase_app_check: ^0.2.1+17
  dash_chat_2: ^0.0.20
  http: ^1.2.0
  provider: ^6.1.2
  uuid: ^4.3.3
  carousel_slider: ^4.2.1
  fluttertoast: ^8.2.4
dev_dependencies:
  flutter_test:
    sdk: flutter
```

2. pubspec.yaml

B.SCREENSHOTS



Homepage and Menudrawer



Event page(click apply to redirect to launch google docs register link) and Settings page