Edu Hub Final Report

Team 7:

Chintamani Masthanaiah - 191CS115
Venkata Sravani - 191CS223
Kruthika K Sudhama - 191CS224
V Madhan Kumar - 191CS260
Vinesh S Talpankar - 191CS265

Certificate

This is to certify that the project named “EduHub” done by Team 7 is an authentic work carried out by them.

It embodies the work done by them during semester V of their course Software Engineering Lab(SE303) under the due supervision of Anjali, Annappa B

Date: 30th October

Signature of the Supervisor

Name of the Supervisor

Acknowledgment

We thank Anjali ma’am for her guidance throughout the project. It is under her right guidance that helped us build EduHub, a one-stop solution for managing events. We are also thankful to Annappa B sir for his supervision throughout the project. And thanks to all the team members of Team 7.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Motivation

Did you ever get frustrated browsing various websites to find the webinars, quizzes, hackathons that interest you?

Do you wish to have an application that can reduce browsing time and act as a one-stop solution?

We ourselves have faced this problem a lot of times. For clubs in college, it's very difficult to reach out to people from different colleges and universities and encourage them to sign up for our events, as messages get lost in Whatsapp or mails. Being a student, it's difficult to find events and register. If we register for an event, we often forget when the event is scheduled. So to solve this problem, we thought of the Edu Hub application.

1.2 Proposed System

The proposed solution is a Mobile Application "EduHub" which is an open platform for all the events, boot camps, webinars from all the colleges to extend their reach and for students to register for various events. The University/Club admin must register on the platform to post the events held by their college. Users can use various filters/tags like Free or paid events, colleges, duration of the event to search for events. Users can also integrate their google calendar to keep track of the events they have registered for.

1.3 Related Work

Currently, there are many websites that serve the same purpose as EduHub does, but we haven't come across a mobile application that can help students in registering and publishing events.

2. REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATIONS

2.1 SRS DOCUMENT

2.1.1 Data Requirements

The set of data that is involved in any project is defined using data requirements. For this project, the main data required is the login information to register the application and the events information. Without this information, the application cannot process the transaction.

2.1.2 Functional Requirements

- Users should be able to sign up with their email accounts.
- Users should be able to search for the events on the platform.
- Users should be able to publish and register for the events.
- Users should be able to keep track of the events they have registered for/ and publisher in the past.
- Google Calendar integration to send reminders before the event.

2.1.3 Performance Requirements

Response time, scalability, platform dependencies, tolerance are the performance requirements that should be considered when developing any system. The application or system should be able to respond quickly when the user interacts with the application. The application should be developed in such a way that it should be scalable enough to accept new features when we want to expand the application complexity. The application should run in all the specified software and hardware requirements from the design phase of the project. Also, the tolerance rate (fault tolerance) of the application should be at a higher level in case of network issues, connectivity issues, and when the application crashes or stops. It should be able to deliver the information about any of those issues to the user when the system is no longer able to provide results when the user wants.

2.1.4 System Requirements

The application should be installed into a device, system or any machine in such a way that it should have basic requirements like supporting software and hardware of the device, accessing in-built software, internet permissions, and potential security issues such as virus or any malware detection.

2.1.5 Testing and Maintainability Requirements

The application should be able to meet all the possible good and bad test cases under a test environment. The application should be developed in such a way that it does not have any issues or crashes when the user is using the application. It should be able to extend itself when we expand the code or implement any new functions to the existing application.

2.2 Validation

Validating any application is an important criterion before releasing the application to the users. If there is no validation, the information entered by users may be redundant, formatted inappropriately, and cannot be maintained. For example, we can validate a mobile number in a way that it should use only digits and letters. Suppose, if the validation is not done, there are chances for the user to enter a wrong phone number and save it. In case of any emergency issues, the authorized person cannot contact the respective person. Similarly, validations for all the fields that are used to save information in any application are highly necessary. In this application, we have done several validations in the Login Page and Home Page. In the Login Page, we have validated all the login information that is required for the user to sign up for the first time. Fields like email, passwords are validated appropriately by displaying error messages. The password should be a minimum of eight characters including at least one digit, one symbol, one uppercase, one lowercase letter, the email should be a valid address and if the email id is already registered, an error message is shown saying that email id exists.

3. SYSTEM DESIGN

3.1. Design Approach

This project is based on the functional design approach, which helps in understanding the design of the project in a simpler way by explaining its flow, use cases, and implementation more like a modular approach. For example, there are different modules in this project which have separate functionality and other sub functionalities/modules. All the modules are designed, implemented, and integrated together to make a flawless working application. The following links are the application design constructed on Figma.

Admin

Participant

3.2. Architectural Design

The detailed design including modules and sub-modules of the application is as follows:

1. User Registration:

If the user wants to use EduHub, they must download the application from the play store, install and register it by providing login information. Once, they register the registered information is stored on the firebase server and can be validated, checking the valid credentials for the next time he logins with the application.

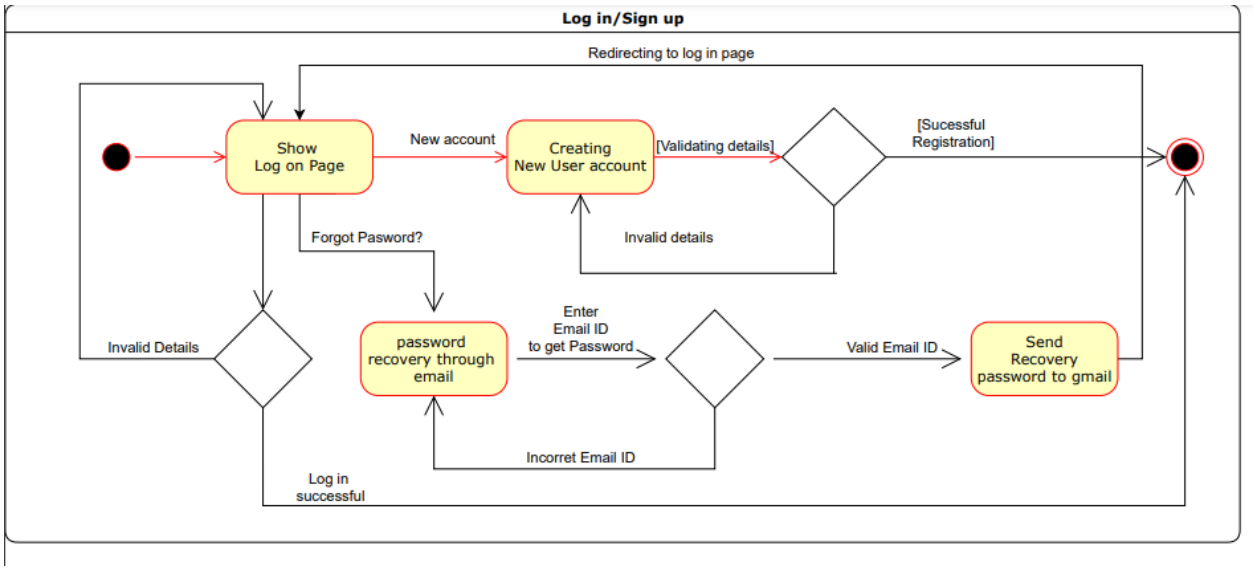
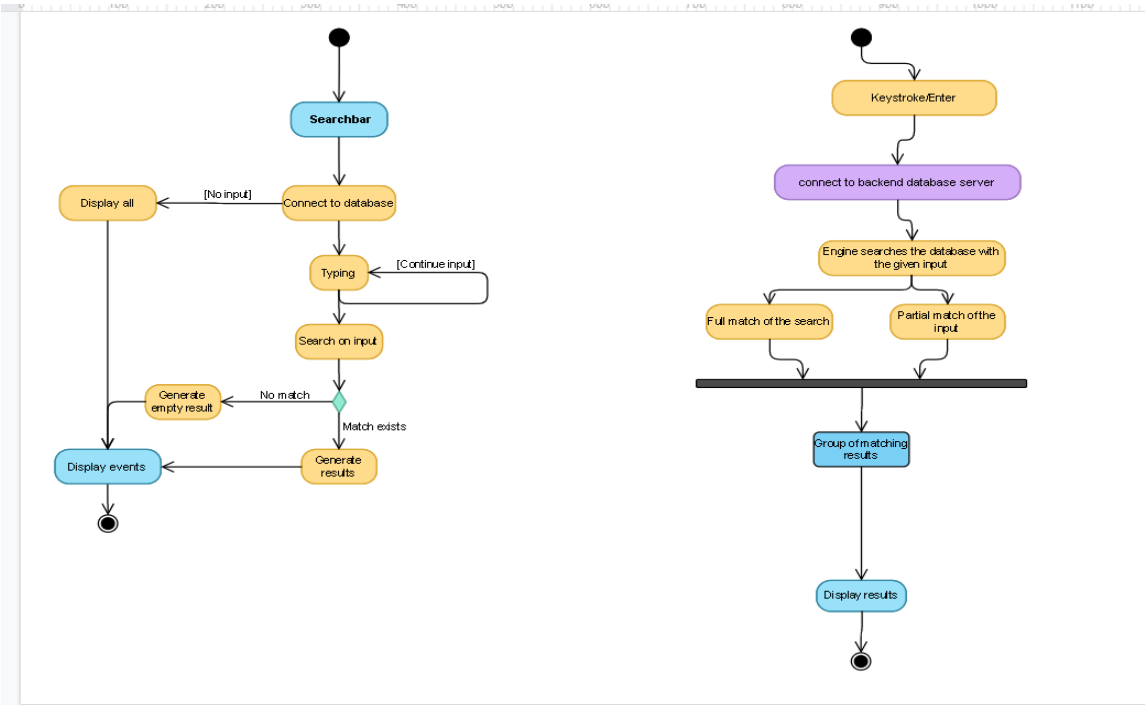


Fig. Log in/ Sign Up activity diagram.

2. Instant Search:

The instant search helps in finding whether an event is published. This could help the users while they want to view an event or register for it without going through a long list of events published on the application.



3. Register for an event:

This feature helps people to register for an event if they haven't already. Suppose if they have registered for the event, the app shows the “Remind Me” option.

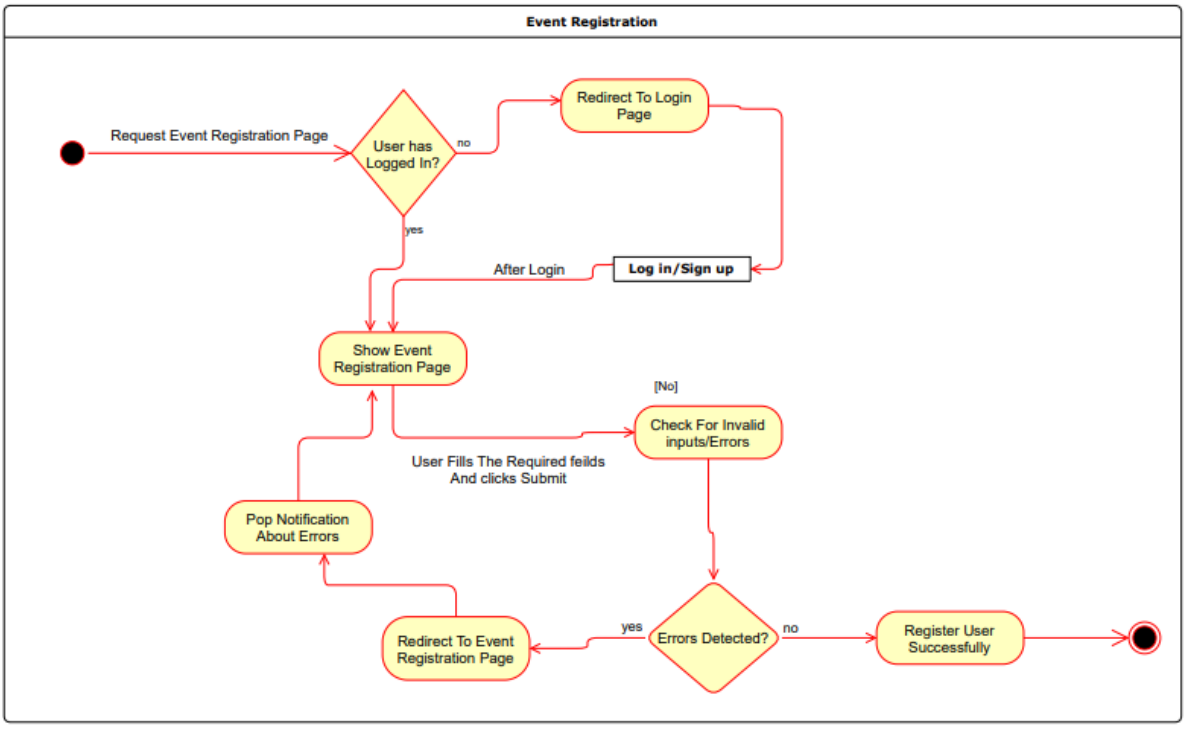


Fig. Event Registration

4. Event Publishing

This feature allows the admins to publish an event to the application. On providing the details, the information gets stored in the Postgres database. The admin also has an option to edit the details after publishing.

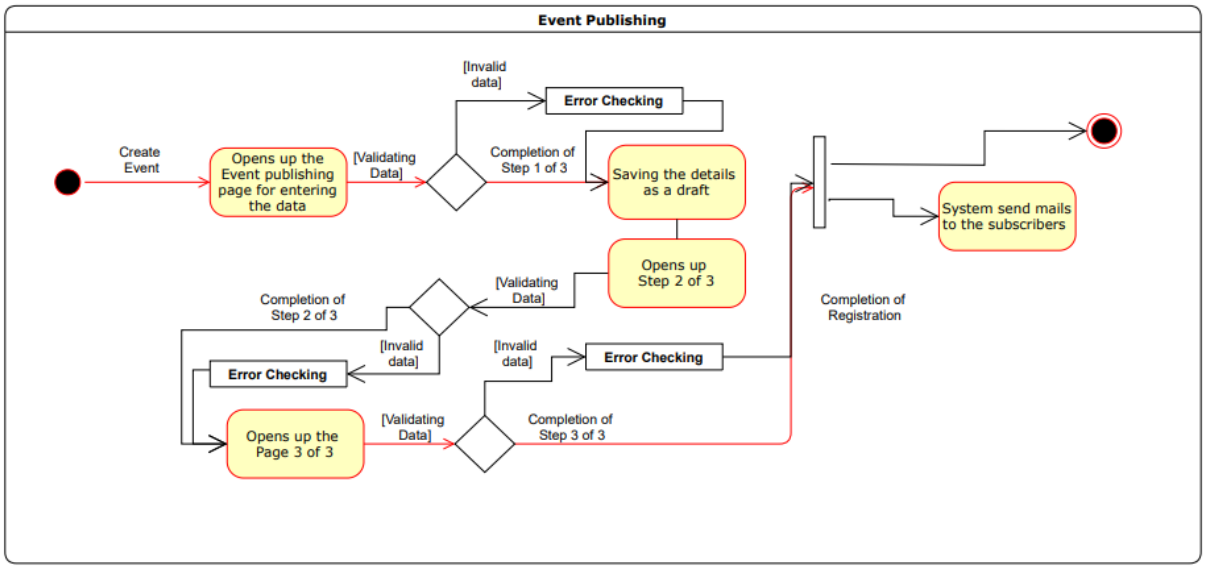
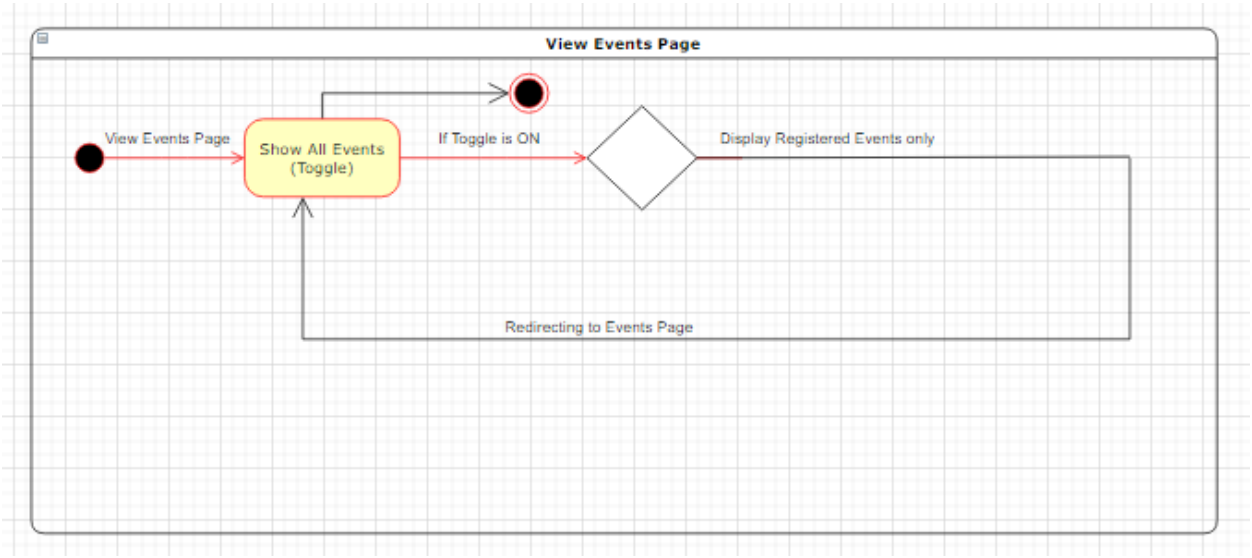


Fig. Event Publishing

5. View Registered Events

This feature allows users to view the events they have registered for. Once the user has registered for an event, their information gets stored in the database and the admin created the event can view their info.

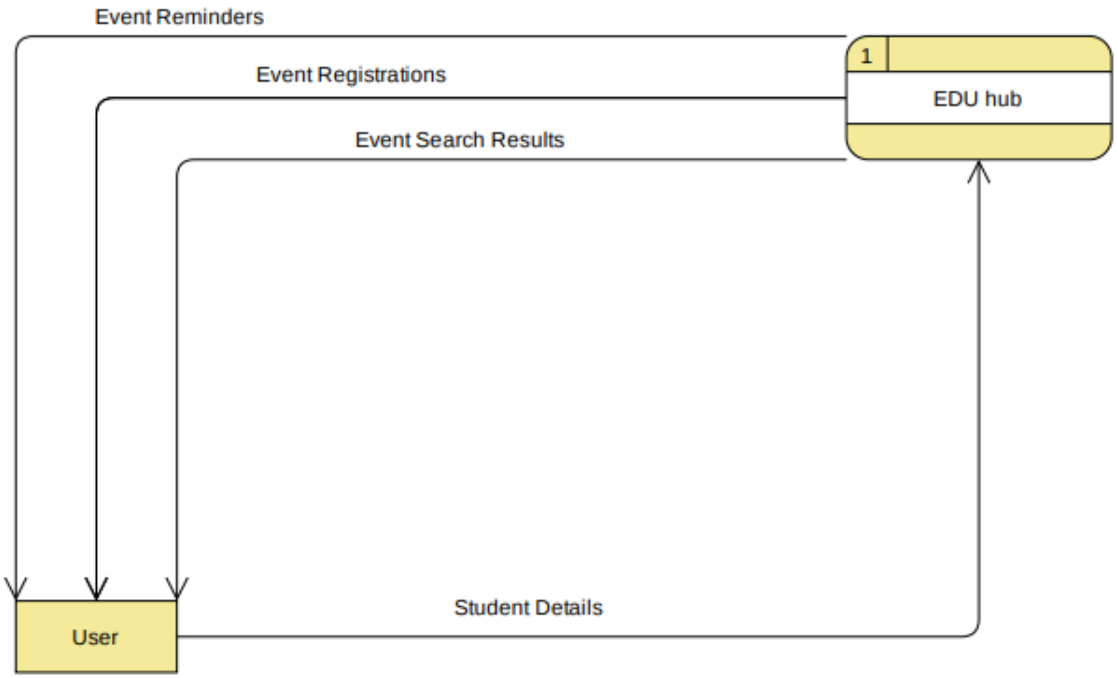


3.3 High-level Design

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.
It shows how data enters and leaves the system, what changes the information, and where data is stored.
The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called a data flow graph or bubble chart.

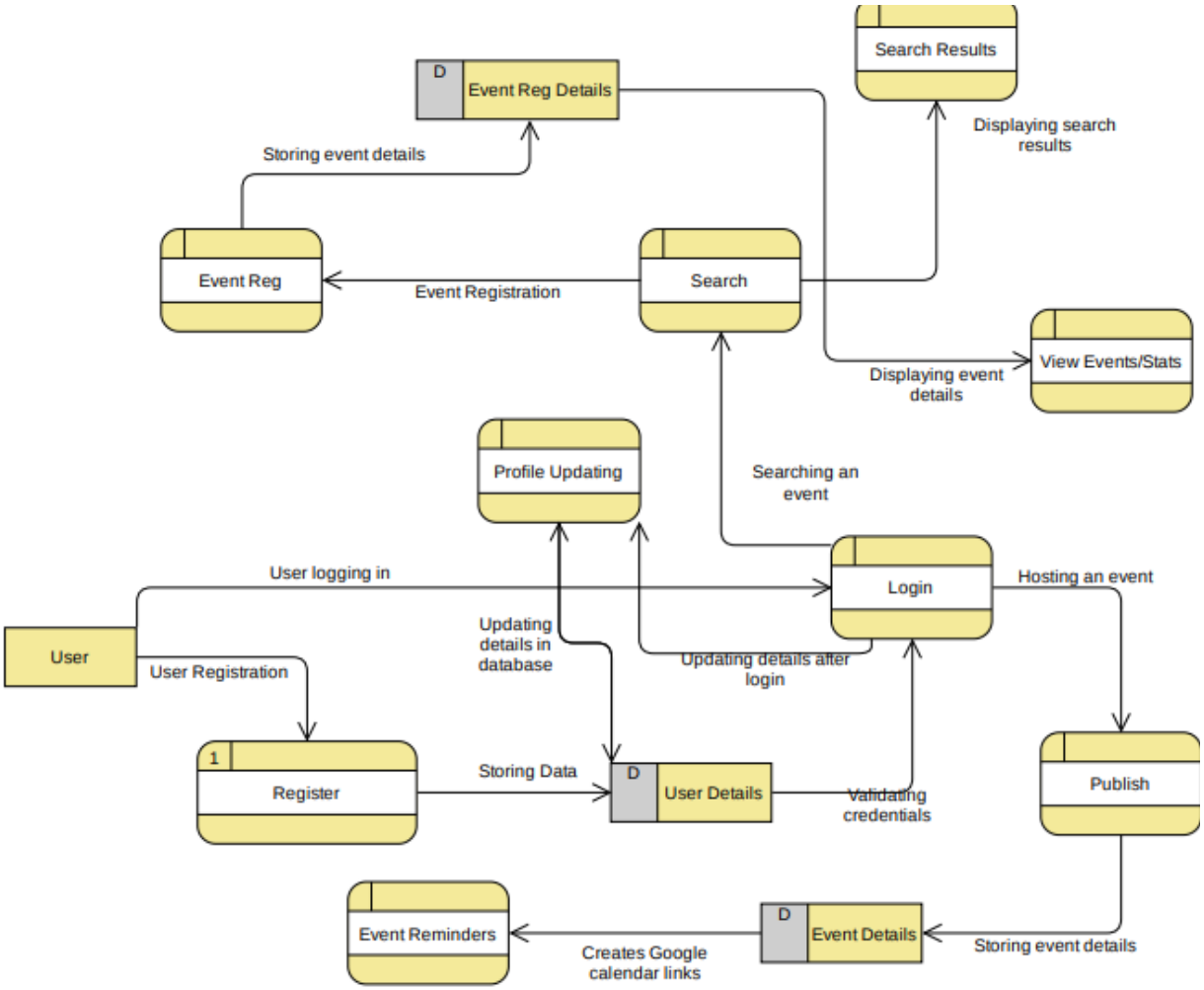
0-level DFD

It is also known as the fundamental system model or context diagram that represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows.



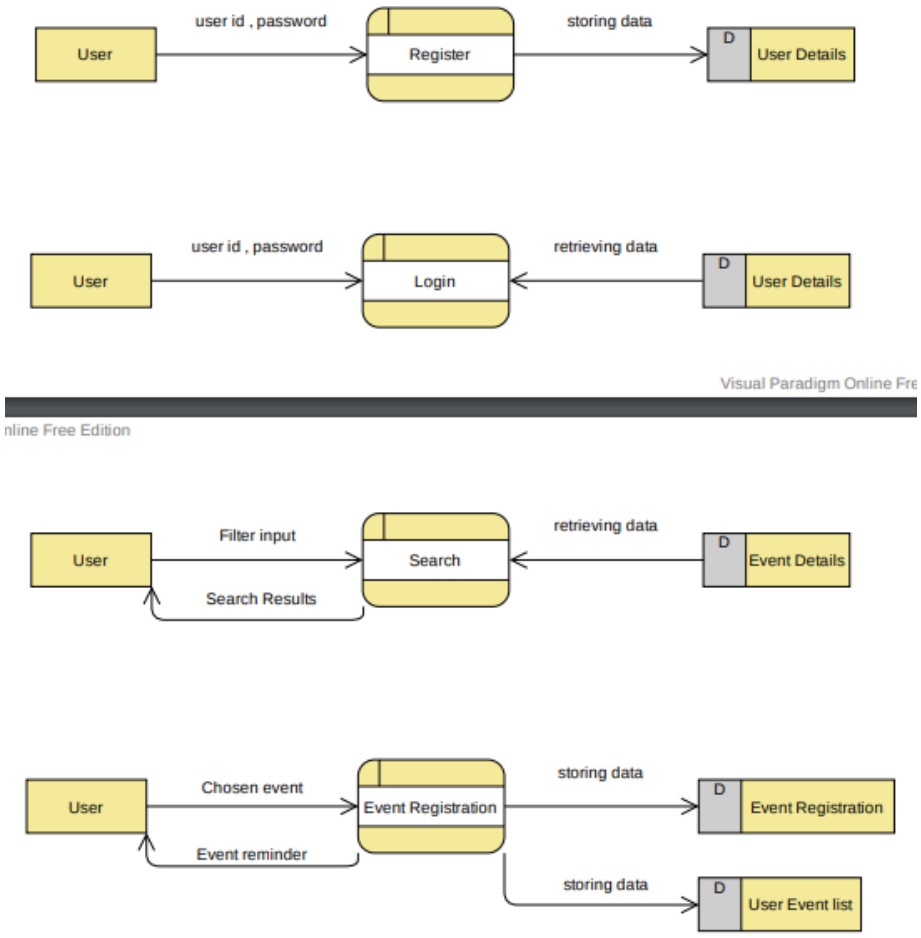
1-level DFD

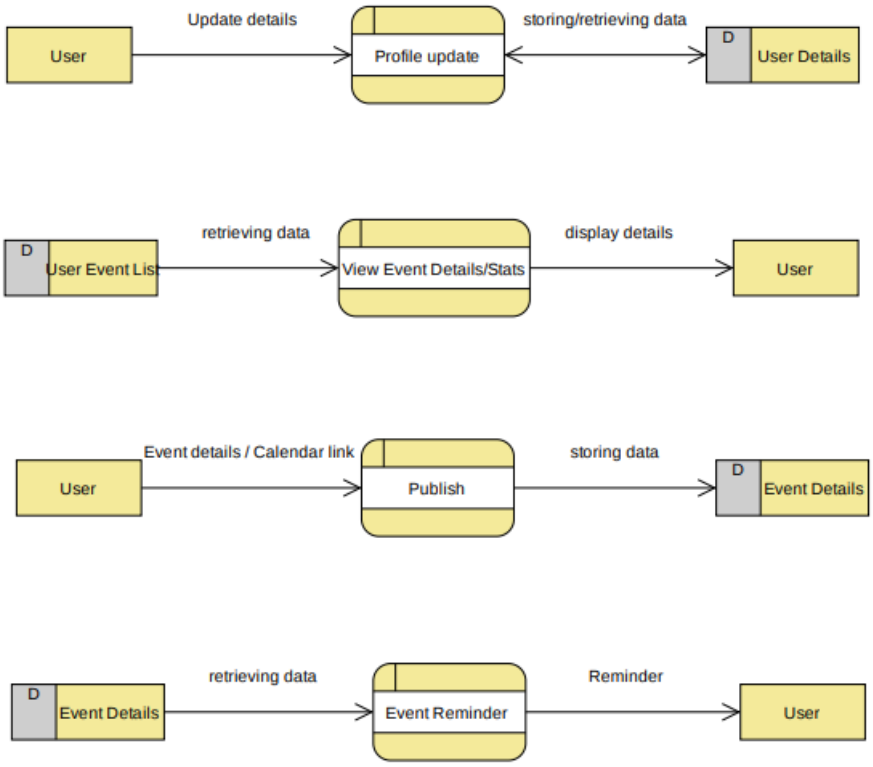
In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and break down the high-level process of 0-level DFD into subprocesses.



2-Level DFD

2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.





Data Dictionary

Data dictionaries store and communicate metadata about data in a database, a system, or data used by applications. Data dictionary contents can vary but typically include some or all of the following:

- A listing of data objects (names and definitions)
- Detailed properties of data elements (data type, size, nullability, optionality, indexes)
- Entity-relationship (ER) and other system-level diagrams
- Reference data (classification and descriptive domains)
- Missing data and quality-indicator codes
- Business rules, such as for validation of a schema or data quality

How Data Dictionaries are Used

- Documentation - provide data structure details for users, developers, and other stakeholders
- Communication - equip users with a common vocabulary and definitions for shared data, data standards, data flow, and exchange, and help developers gage impacts of schema changes
- Application Design - help application developers create forms and reports with proper data types and controls and ensure that navigation is consistent with data relationships
- Systems Analysis - enable analysts to understand overall system design and data flow, and to find where data interact with various processes or components

- Data Integration - clear definitions of data elements provide the contextual understanding needed when deciding how to map one data system to another, or whether to subset, merge, stack, or transform data for a specific use
- Decision Making - assist in planning data collection, project development, and other collaborative efforts

ID	Term	Explanation
1	User	Anyone using the application
2	Event	Quiz, webinar, that the user can create and register for.
3	Admin	Can create, edit events.
4	Invigilator	Can allot scores to the users registered for the events.
5	Profile	Contains details about user email, name, and registered events.

3.4 User Interface Design

User Interface Design for any application should be very simple. We should have only a few clicks or navigation among the features when using the application to avoid the hassle.

3.5 Database Design

The database should be designed in such a way that it should be easy to access and manipulate. Database definition and database manipulation operations should be performed accordingly to add, delete, and update values. In this project, We have used a Postgres database which is an open-source database, easy to install and use.
The database server could be installed by providing user credentials such as username and password. The created database is hosted remotely on the amazon AWS server. We can use the pgadmin4 tool to view and update the database locally.

We have created the tables in the database using the following commands.

Tables :

```
create table fest(
        fest_id varchar ( 8 ),
        fest_name varchar ( 20 ),
        college_name varchar ( 20 ),
        start_date DATE ,
        duration varchar ( 8 ),
        primary key (fest_id)
);

create table evento(
        event_id varchar ( 8 ),
        event_name varchar ( 20 ),
        start_date_time TIMESTAMP ,
        end_date_time TIMESTAMP ,
        register_start_date_time TIMESTAMP ,
        register_end_start_time TIMESTAMP ,
        place varchar ( 20 ),
        description varchar ( 300 ),
        calendar varchar ( 300 ),
        primary key (event_id)
);

create table has(
         event_id varchar ( 8 ),
        fest_id varchar ( 8 ),
        primary key (fest_id,event_id),
        foreign key (event_id) references evento on delete cascade ,
        foreign key (fest_id) references fest on delete cascade
);

create table participant(
        participant_id varchar ( 8 ),
        participant_name varchar ( 20 ),
        participant_cno numeric ( 10 ),
        participant_email varchar ( 50 ),
        primary key (participant_id)
);
create table individual_participant(
         participant_id varchar ( 8 ),
        event_id varchar ( 8 ),
        score numeric ( 10 , 3 ),
        review varchar ( 30 ),
        primary key (participant_id,event_id),
        foreign key (participant_id) references participant on delete cascade ,
        foreign key (event_id) references evento on delete cascade);
```

```
create table group_participant(
        participant_id varchar ( 8 ),
        event_id varchar ( 8 ),
        score numeric ( 10 , 3 ),
        group_id varchar ( 8 ),
        group_name varchar ( 20 ),
        review varchar ( 30 ),
        primary key (participant_id,event_id,group_id),
        foreign key (participant_id) references participant on delete cascade ,
        foreign key (event_id) references evento on delete cascade
);

create table admino(
        admin_id varchar ( 8 ),
        admin_name varchar ( 20 ),
        admin_cno numeric ( 10 ),
        admin_email varchar ( 50 ),
        pass_word varchar ( 50 ),
        primary key (admin_id)
);

create table invite
(
        event_id varchar ( 8 ),
        foreign key (event_id) references evento on delete cascade
);
```

3.6 Entity-Relationship Diagram

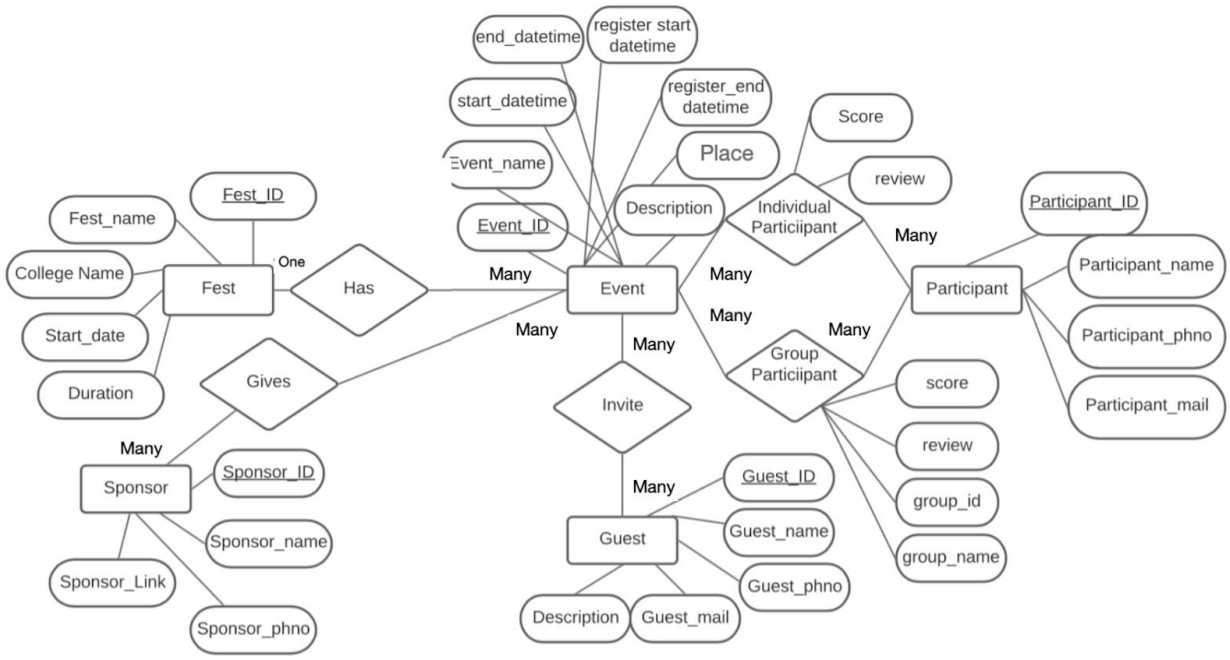


Fig. ER Diagram

3.7 Methodology

Agile Software Development Methodology is widely used in many projects as it has many advantages. After gathering the project requirements, it is reviewed frequently in the form of small iterations and made into action by executing it. After completing tasks each iteration, it could be reviewed properly and moved to the next iteration. The main advantage of this methodology is that we can change the requirements or design even in the middle of the project when the situation arises. Also, code maintenance is easier in contrast to the Waterfall Approach. In the Waterfall methodology, there is no flexibility in changing the requirements when we develop the project because we must understand the working flow of the project at least 80% even before the start of the project and work. Only if the design process is done, we can move to construction, testing, and support.

We have changed some of the requirements for the convenience of the user interface while developing the project which is the main advantage of this methodology. Hence, for these reasons, we had decided to follow the principles of agile methodology

4. IMPLEMENTATION, TESTING, AND MAINTENANCE

4.1 Introduction to Programming Languages, IDE'S, Tools and Technologies used for this Implementation

4.1.1 Dart

As the project is developing an Android Application using Flutter, the default programming language is Dart.

What makes Dart unique?

Before working on Dart, the Dart team members had done groundbreaking work on advanced compilers and virtual machines, both for dynamic languages (like the V8 engine for JavaScript and Strongtalk for Smalltalk) and for static languages (like the Hotspot compiler for Java). They used this experience to make Dart unusually flexible in how it can be compiled and executed.

Dart is one of very few languages (and perhaps the only "mainstream" language) that is well suited to be compiled by both Ahead Of Time and Just In Time. Supporting both kinds of compilation provides significant advantages to Dart and (especially) Flutter.

JIT compilation is used during development, using a compiler that is especially fast. Then, when an app is ready for release, it is compiled AOT. Consequently, with the help of advanced tooling and compilers, Dart can deliver the best of both worlds: extremely fast development cycles, and fast execution and startup times.

Dart's flexibility in compilation and execution doesn't stop there. For example, Dart can be compiled into JavaScript so it can be executed by browsers. This allows code reuse between mobile apps and web apps. Developers have reported as high as 70% code reuse between their mobile and web apps. Dart can also be used on a server either by being compiled to native code or by compiling to JavaScript and using it with node.js.

Finally, Dart also provides a standalone VM that uses the Dart language itself as its intermediate language (essentially acting as an interpreter).

Dart can be efficiently compiled AOT or JIT, interpreted, or transpiled into other languages. Not only is Dart compilation and execution unusually flexible, but it is also especially fast.

4.1.2 IDE's, Tools and Technologies

4.1.2.1 Android Studio

Android Studio is exclusively designed for developing Android applications. It consists of all Android SDK tools to design, develop, maintain, test, debug and publish our app. The IDE is designed very efficiently which makes the developer's job easy. It also supports the IntelliJ IDE, the main idea behind this IDE is that it automatically senses the variables, methods, classes, built-in functions or it could be anything else when we press the first letter of it. Say, suppose we declare a few variables or methods that start with an 'S', it automatically senses everything that starts with an 'S' and makes suggestions. It also supports Git as a version control system to maintain the app changes and push them into GitHub. All java files, layout files (for design) are integrated into a single project easily. After the completion of the project, the whole application could be put as an APK (Android Package) file, in which we can run that APK file on any device and use the application. Other main tools include Android SDK, ADB, and Gradle Build.

4.1.2.2 Android Software Development Kit (SDK)

One of the main tools used in developing android applications, as it packages many core features into one SDK and it can be used in the application easily. This helps us to avoid writing a lot of code, and building applications faster.

4.1.2.3 Android Debug Bridge (ADB):

Android SDK uses the ADB tool as a connection device which allows us to connect the Android Devices or Emulator with the machine via USB. After developing or while developing applications, we can connect with the device to check how the application runs. Later, we can debug and run the applications.

4.1.2.4 Gradle Build:

Gradle Scripts are the recent feature that is added to Android Studio. It is basically an automated build system that is used to automate the various phases involved in designing an application that includes design, development, test, debugging, and publishing. We need to configure the project and modules by mentioning all the supported jar files, SDK's, version name, level, compiled SDK version, build tools version. to ensure that the developed app is compatible with the testing device/emulator.

4.1.2.5 Android Device Monitor:

If we want to access all the hidden files that are generated when we run the application, we can use the monitor. We can select any project and explore the files that are related to that project. But, as they are hidden files, we need root permissions to access them. Suppose, if we run the app in the device, we need to root the device and run commands in the ADB shell to get permissions.

4.1.2.6 SDK Manager:

It is one of the main tools to maintain the updates of all the installed components required to run the project. It also notifies us when the project is not compatible with the device or any other compatibility issues and to download any component that is required.

4.1.2.7 AVD Manager:

It is used to create virtual devices of any desired API level to support higher-level SDK in case our device does not support it. Using emulators to test the application is difficult as it might be a little slower when compared to the real device.

4.1.3 Visual Studio IDE

Android applications could be developed in VS CodeIDE in which we can compile, run, debug and deploy using ADT (Android Development Tools). In this project, we have used VS Code IDE to create API services for connecting the database and used Android Studio to develop the application.

4.1.4 Postgres Database

In this project, I have used the Postgres database to store the data. This is one of the popular open-source relational database management systems. We can perform all DDL, DML, DCL operations using this database. This also supports different programming language applications. The applications could connect the database using the postgres package in Flutter.

4.1.5 Amazon AWS Server

AWS is an online cloud computing platform that provides servers, storage, networking, remote computing, etc. AWS supports PostgreSQL through a fully managed database service with Amazon Relational Database Service (RDS). The database is kept in this remote server and could be accessed whenever needed through the pgadmin4 tool or terminal.

4.1.6 Firebase Authentication

We are using firebase authentication for user authentication. Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. Firebase Authentication integrates tightly with other Firebase services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend.

4.2 Security and Permissions in Android

Security notions in Android are quite high. Whenever a new Android Application is created, a unique user and group ID. This makes the maintenance of the application easier way to avoid any security or privacy issues. As the application is created uniquely, it becomes private and no one can access other's applications. Permissions are another important concept which is included in the AndroidManifest.XML configuration file. This is required if the application wants to access the external features. For example, if the application wants to access the Internet, Camera or it could be any feature, it requires permissions. It is included within the tags as it is an XML file. Permissions are automatically created for the basic applications at the time when we create the application. If the app uses higher-level API or SDK we must explicitly mention the permissions inside the uses-permissions tag to access the features or components.

4.3 Test Plan and Test Activities

Test plan is necessary for any project to plan the testing phase and decide the scope of the project. Test plan involves collecting design specifications about the project, wiring test cases, executing them manually or automatically using automated testing tools. Testing any application is highly important. A test plan is a method of documenting the test cases, specification plans, and other basic level details about how the application works.

Testing Scope:
The scope of the testing for our project can be categorized as follows:-

a. In-Scope:

Functional Testing for the following modules are in the Scope of Testing

- Authorization Module
- Participant Module
- Admin Module

b. Out of Scope:

Performance Testing was not done for this application.

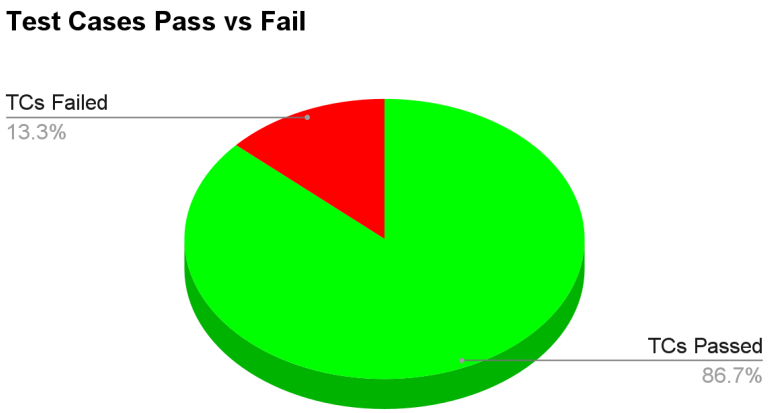
c. Items not tested:

Many inbuilt widgets provided by Flutter were used in this project. These widgets don't require testing as they are well tested. Firebase is also used in this project to implement Google SignIn/SignUp and doesn't require testing.

Metrics:

A. No. of Test Cases Planned vs Executed
B. No. of Test Cases Passed/Failed

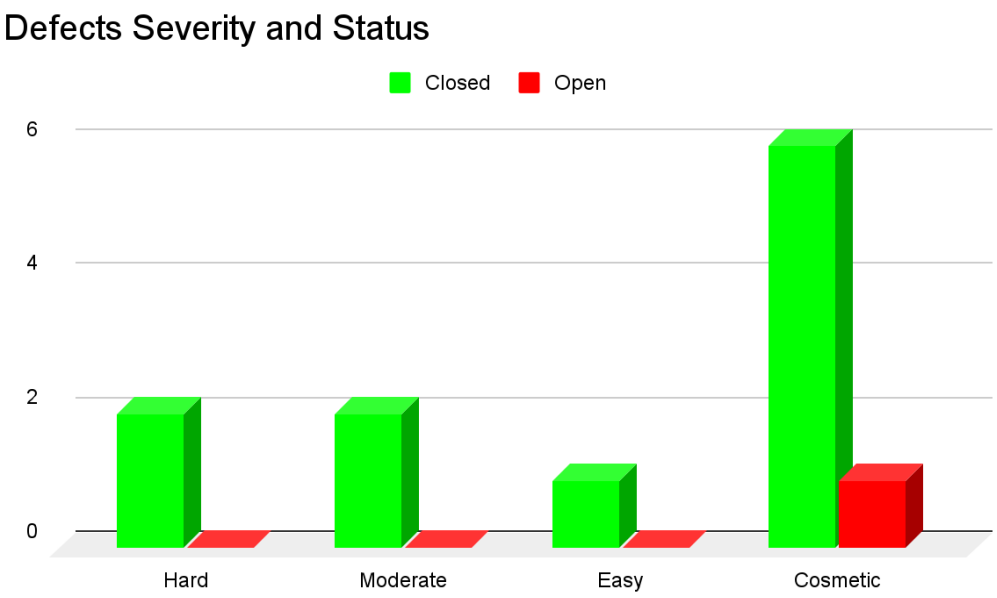
Test Cases Planned	Test Cases Executed	TCs Passed	TCs Failed
20	15	13	2



C. No of Defects Identified and their Status & Severity

	Hard	Moderate	Easy	Cosmetic	Total
Closed	2	2	1	6	**11**
Open	0	0	0	1	**1**

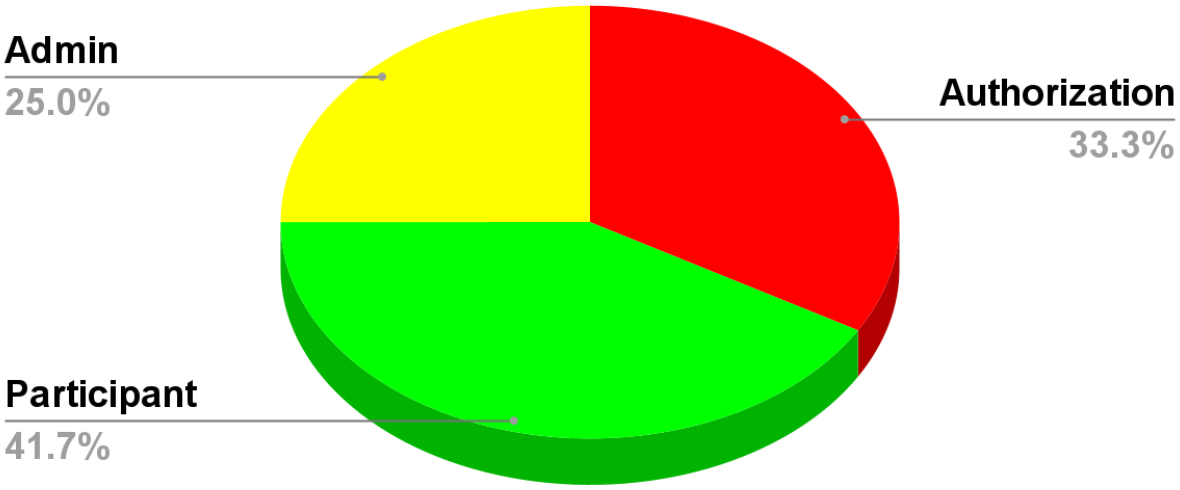
Total = 12



D. Defects Distribution – Module Wise

	Authorization	Participant	Admin	Total
Easy	0	1	0	1
Moderate	2	0	0	2
Hard	1	1	0	2
Cosmetic	1	3	3	7
Total	4	5	3	**12**

Defects Distribution – Module Wise



Types of Testing Performed :

a) Smoke Testing:

This testing is done whenever a Build is received (deployed into the Test environment) for Testing to make sure the major functionalities are working fine, Build can be accepted and testing can start.

b) System Integration Testing:

This is the Testing performed on the Application under test, to verify the entire application works as per the requirements. Critical Business scenarios were tested to make sure important functionalities in the application works as intended without any errors.

Types of Testing in Flutter:-

Unit Testing

Unit testing is the easiest method to test an application. It is based on ensuring the correctness of a piece of code (a function, in general) o a method of a class. But, it does not reflect the real environment and subsequently, is the least option to find the bugs.

Widget Testing

Widget testing is based on ensuring the correctness of the widget creation, rendering and interaction with other widgets as expected. It goes one step further and provides a near real-time environment to find more bugs.

Integration Testing

Integration testing involves both unit testing and widget testing along with external components of the application like database, web service, etc., It simulates or mocks the real environment to find nearly all bugs, but it is the most complicated process.
Flutter provides support for all types of testing. It provides extensive and exclusive support for Widget testing. In this chapter, we will discuss widget testing in detail.

Test Environment and Tools:

Dart language and Flutter framework provide extensive support for the automated testing of a Flutter application. Hence, the environment and tools used are the same as that of the application itself.

Bugs Patched:

Bug 1 :

Difficulty: Moderate
Where: Under User Profile Update Feature (user_profile_update.dart)
What: New Usernames and Emails were not getting updated in the Postgres Database
Solution:

1. Firstly, we cross verified the queries we were using to update both the username and password

   For updating Username :
   update participant set participant_name=\'$username\' where participant_email=\'$email\' ;

For updating Email :

update participant set participant_email =\'$newemail\' where participant_name=\'$username\' ;

2. As queries were working correctly, we moved on to check the async function (checking whether the data is getting retrieved from the database before or after the widget is built) , here we added await keyword to make sure that all processes waited until the data from our queries were retrieved from our Database.

3. We check which type of user is logged into the app (whether Participant or Admin), we ran a sample query:

   select admin_email from admino;

   We then properly assigned the type of User to the above queries.

4. Both username and emails were successfully getting updated in the Postgres Database.

5. Bug Fixed

Bug 2 :

Difficulty: Hard
Where: Under View Events Feature (FestDetails.dart)
What: The switch was not filtering the registered events from all events
Solution:

1. Firstly we select a particular participant and find the events available to him and the ones he has registered by running queries on the Postgres Database.

2. We have already created the Switch widget in the Scaffold.

3. We must make sure that the OnChanged value is assigned to the isSwitched variable inside a setState() function. So that the build is rerun whenever the isSwitched value changes.

4. Now inside the runQuerry() function (which handles data selection from evento table) we need to make sure that when

   - IF isSwitched == false

     Then all events must be added to result, query is:

     ‘select * from evento’

- ELSE

  Then only registered events must be added to the result, for that we must first find participant_id. Then use that to get all registered event details for the participant. Queries are:

  'select participant_id from participant where participant_email= \'$mail\' '

  'Select A.event_id, A.event_name, A.start_date_time, A.end_date_time, A.register_start_date_time, A.register_end_date_time, A.place,A.short_description, A.description, B.group_name, A.price from evento as A , group_participant as B where A.event_id = B.event_id and B.participant_id = \'$regID\' '

5. We can run this and see that the Filter is working properly by verifying the results from step 1.

6. Bug Fixed

Bug 3 :

Difficulty: Easy
Where: Under User Profile Display Feature (userprofile.dart)
What: Username and Email were not getting displayed after updating.
Solution:

1. We checked whether the Username and Email were updated by printing them in the flutter console. Both the Username and Email were updated and displayed on the console.

2. We looked into the widget, where we displayed both the credentials on the screen. We had to build the widget every time we visited the User Profile Display Page.

3. So, we used the setState function, which would build the widget every time we visited the User Profile Display Page, and display both the credentials in real-time (even right after the update).

4. Bug Fixed

Bug 4 :

Difficulty: Hard
Where: Under User Profile Update Feature (user_profile_update.dart)
What: Emails were not getting updated in the Firebase Authentication List
Solution:

1. We first read up on the details regarding updating emails in the Firebase auth list , we got to know that firebase had set a few restrictions on updating any of the important credentials like email or password.

2. We had to Re-Authenticate with the previous Email and Password.

await FirebaseAuth.instance .signInWithEmailAndPassword(email: email, password: password)

3. Now, we require Email and Password from SignIn Page, we have email already stored in a global variable (widget._user.email), now we similarly transfer password from SignIn page to User Profile Page using a global variable (underclass g , method pass).

4. After Re-authenticating with credentials, we now update the New Email.

await user.updateEmail(newemail);

5. Now, we checked whether New Email got updated in Firebase using try and catch block to display errors if try block fails.

6. New Email was updated in Firebase and we could even login with the updated email on the SignIn Page.

7. Bug Fixed

Bug 5 :

Difficulty: Moderate
Where: Under SignIn Feature (SignIn.dart)
What: Google SignIn & SignUp were not working
Solution: hen

1. We had to generate SHA 1 and SHA 256 certificate fingerprints for each of our systems and update them in the Firebase SDK Configuration.

2. We generated both the keys using the command:

Gradle signingReport

3. We added both keys in the Firebase SDK Configuration.

4. Google SignIn and SignUp worked perfectly t on.

5. Bug Fixed

Exit Criteria:

a) All test cases should be executed – Yes
b) All defects in Hard, Moderate, Easy severity should be verified and closed – Yes.

Hence, we have successfully tested various modules of the Application and patched all the bugs that were encountered. Therefore, the application is suggested to 'Go Live' by our team.

4.4 Application Maintenance

While one always tries to create apps that are free of bugs, they're sure to crop up from time to time. Since buggy apps lead to unhappy users and customers, it's important to understand how often our users experience bugs and where those bugs occur. That way, we can prioritize the bugs with the highest impact and work to fix them.

Software maintenance is 'the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment'.
Maintenance is always necessary to keep software usable and useful. Often there are very tight constraints on changing software and optimum solutions can be difficult to find. Such constraints make maintenance a challenge; contrast the development phase, when designers have considerably more freedom in the type of solution they can adopt.

Goals

1. To adapt a tool for Mobile app error tracking, performance monitoring, and reporting run time errors in a single view, giving a holistic overview of our application's health in real-time.
2. Better handling of security updates, bug fixes, and audits against the latest versions of iOS and Android.

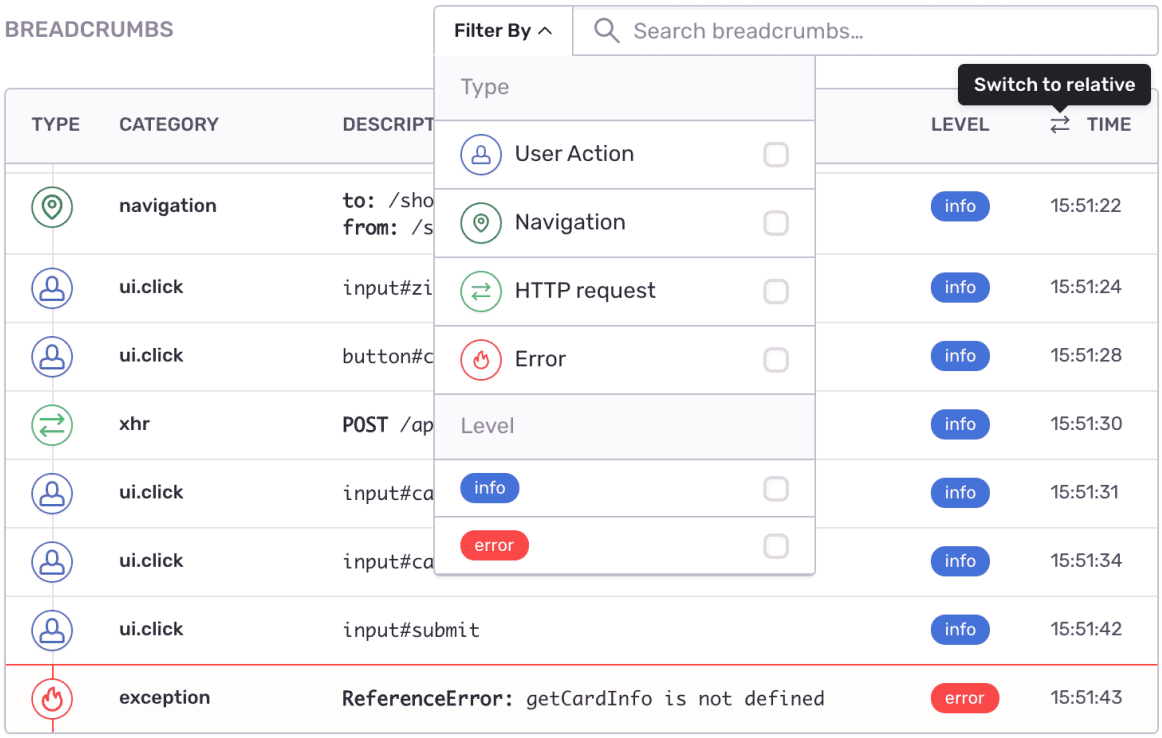
Specifications

Goal 1:
Enabled Firebase Google Analytics:

- We are using Firebase authentication for Sign in and Signup purposes. So we have enabled firebase google analytics to check users' activity, bug reports, and other details related to authentication.
- Google Analytics helps us understand how people use our ios, or Android app. The SDK automatically captures a number of events and user properties. Once the data is captured, it's available in a dashboard through the Firebase console. This dashboard provides detailed insights about our data — from summary data such as active users and demographics.
- In our case, it can provide a report on the number of logins of users and their frequent usage of our app.

Using sentry for performance monitoring and error tracking:

Using Sentry for the app error tracking, performance monitoring, and reporting run time errors in a single view, giving a holistic overview of your application's health in real-time. Correlating errors with releases, tags, and devices, to solve problems quickly, decrease churn and improve user retention.
The sentry report looks as below



Goal 2:
For Security updates:

- We keep current with the latest Flutter SDK releases. We regularly update Flutter, and these updates may fix security defects discovered in previous versions. Check the Flutter change log for security-related updates.
- Keep our application’s dependencies up to date. We make sure we upgrade our package dependencies to keep the dependencies up to date. Make sure to check periodically to see if our dependencies have had security updates, and update the pin accordingly.
- Keep our copy of Flutter up to date. Private, customized versions of Flutter tend to fall behind the current version and may not include important security fixes and enhancements. Instead, routinely update our copy of Flutter.

Milestones

There are 3 kinds of maintenance activities

1. Corrective maintenance
    - When the application gets disconnected to the internet while using application. Connection to the database is also disconnected, which on reconnecting to the internet doesn't connect back to the database. To connect to the database requires restarting the situation. This problem can be addressed by saving the state of the application in the database and restoring that state when the connection is back online.
    - When a new user is signed in with the help of email and password, the details like contact number and username do not get inserted into the database until the user registers for the event. To solve this, the application needs to take these details as default or they need to be asked during the registration process.
    - When the login fails, the user needs to get back to the initial page if the details about the user are not found in the database but found in the firebase. This problem can be eliminated by entering some data into the database when a user creates an account.
    - Users are allowed to have 10 active sessions with the database. If it exceeds there is some problem with the data given by the user which is violating. This needs to be captured by monitoring the database sessions frequently.
2. Perfective maintenance
    - The place taken by the events on the homepage would tire the user due to long scrolling when the count of the total events increases. This block needs to be shortened either by decreasing the font or by changing the design of the block.
    - The application after release, we provide the recommendation system based on the data we have in the database. This can add extra features to the account information which is to be stored in the database. This may significantly change the database schema structure and new tables need to be generated for the recommendation.
    - The delay between the transitions needs to be reduced. This can happen to generate parallel connections with the firebase as well as the database when that is possible. Few places this may not work especially during registration and sign-up. But this can be done when the account information is updated.

3. Adaptive maintenance
    - Since both database and firebase is online, we should make sure the application will satisfy the requirements of the new updates
    - When the application grows longer, we should be in a state to expand the database which can overcome the limited features with the present database. This can be done either by creating another database.

App Updates:

1. Critical or Urgent
    - The delay between the pages needs to be reduced. This can be solved by having good internet and also choosing faster protocol while communicating with the server.
    - The admin who would like to register for any event is not possible with the current edition as we thought the admins would be separate from the users. Now when the admin wants to apply for any event, we should also provide that facility to users.
    - Padding problems when the application runs on different interfaces should be centralized. This can be fixed by choosing the padding size as a ratio of the screen size or extended functionality of the widget.

2. Critical and Routine
    - There is a lot of space wasted on the UI of the application which should be developed. But this is not compulsory. Finding a better widget can remove this problem.
    - Customer contact facilities need to be implemented or a help center which would be useful for the customers.

3. Non-Critical and release updates
    - Better UI based on the customer request.

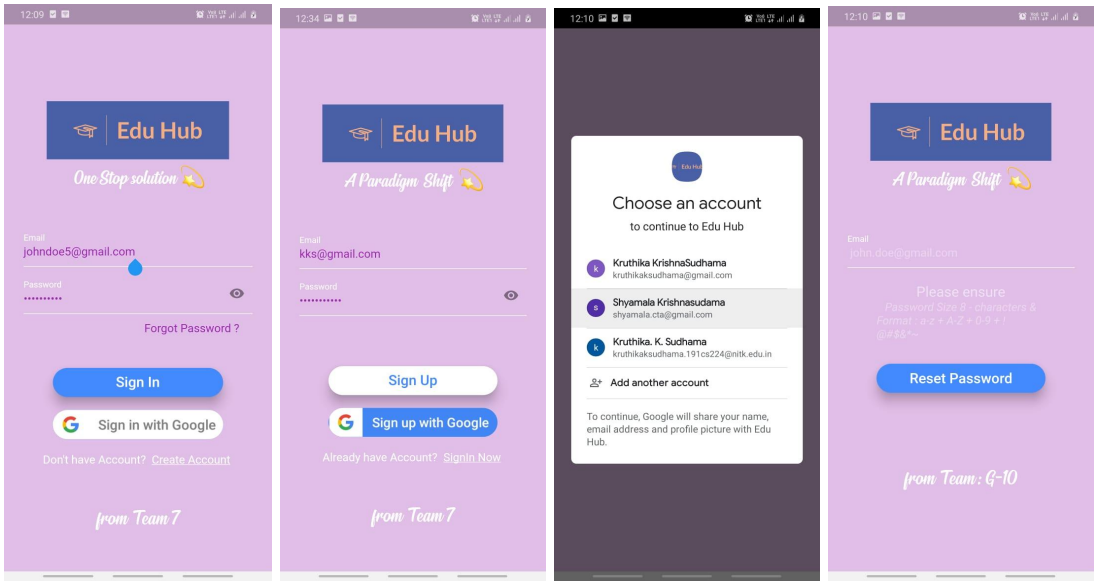
Challenges in Maintenance:

1. Updates in the Database or Firebase authentication need to change the application accordingly to run smoothly. We should be checking the databases and firebase for any updates.
2. New changes in the technology in the servers should be noticed as early as possible.
3. Any updates in the flutter widgets or UI need the application to deploy a new version for a better experience to the user.

5. RESULTS AND DISCUSSION

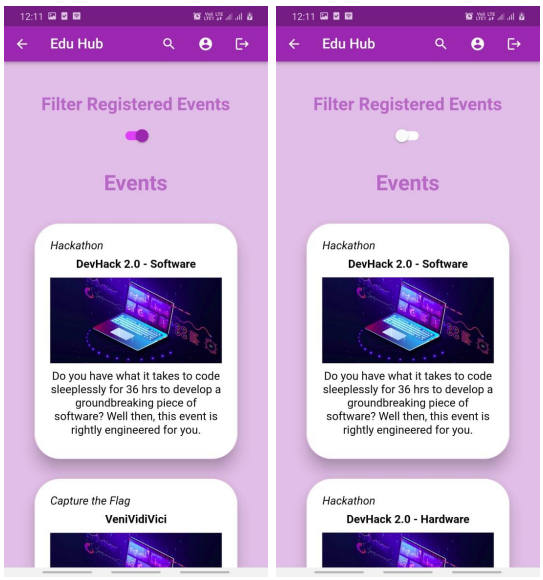
1. User Login/Registration:

- If the user wants to use EduHub, we must download the application from the play store, install and register it by providing login information. As shown in Figure, the login information includes Email-Id and password.
- For the new user, the user must click on the ‘Create account’ link on the login page to navigate to the Signup page. Then sign up by providing Email-Id and Password. Once he registers, the registered information is stored in the server and can be validated, checking for the valid credentials for the next time he logins with the application.
- There is also a Google Signup/Sign-in option. So the user can click on ‘sign in with Google’ and choose the Gmail account to log in.
- If the user has forgotten the password, the user has to click on ‘Forgot password’ and enter the Email-Id to reset the password.



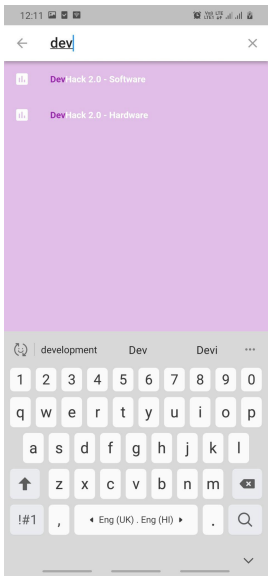
2.HomePage

- The homepage has options to view profiles, search, and log out on the toolbar.
- The body of the page displays the list of all events.
- We have a filter to display only the registered events of the user. The user has to click on the toggle to filter out registered events.



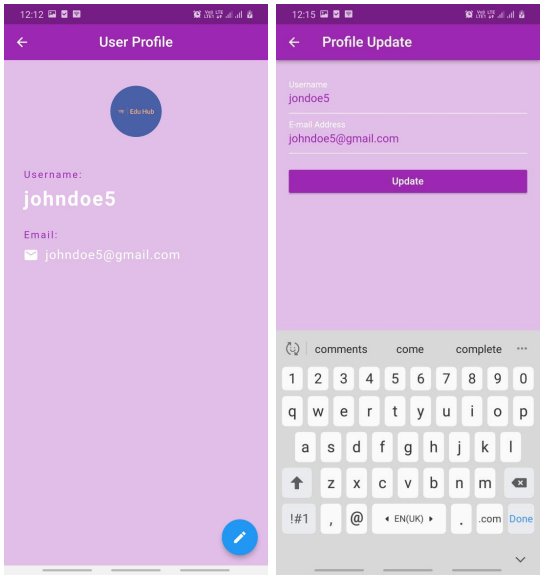
3. Search bar

- User clicks on the search icon on the home page to navigate to the search bar, then we can search for a particular event by typing it out.
- Once we start typing a few starting letters of the event it displays the events matching the text, we can easily select from those.



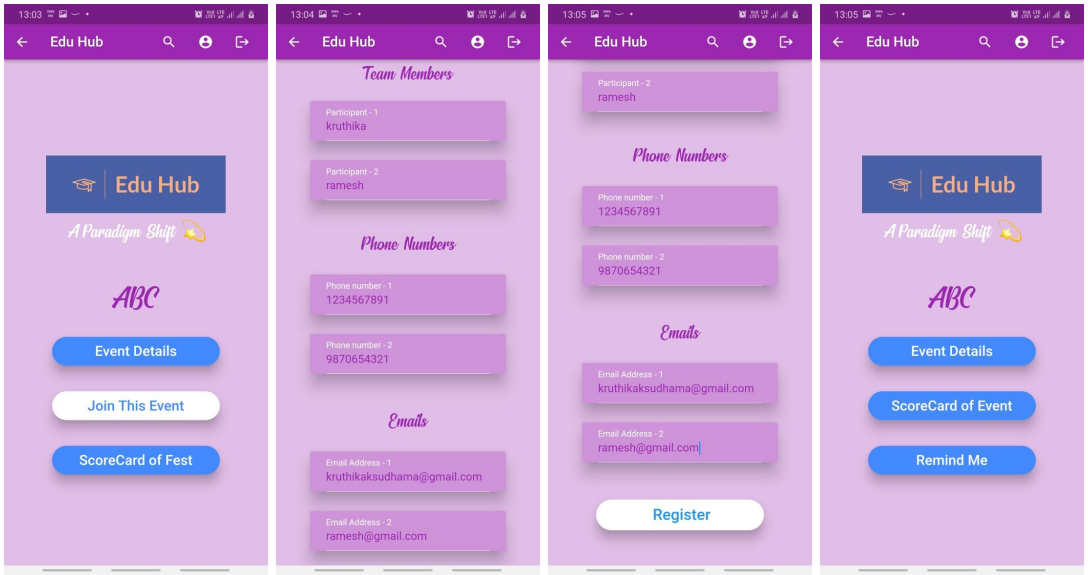
4. View Profile/Profile update

- The user can navigate to his profile page by clicking on the profile icon on the home page.
- On the profile page the user gets an option to edit the username, so he can click on it to edit the username.



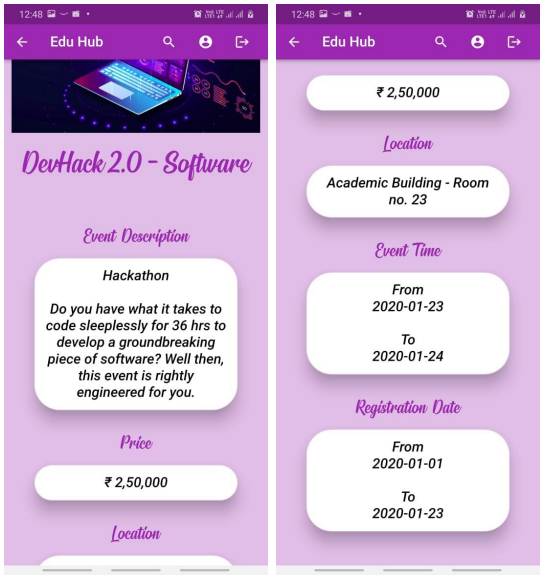
5. Register for an event:

- The user can register for an event by clicking on an event on the home page, then clicking on ‘Join the event’.
- An event will need a particular number of participants as specified by the admin.
- So the user should fill his and his team members' details.
- Then click on ‘Register’. This completes the event registration process.



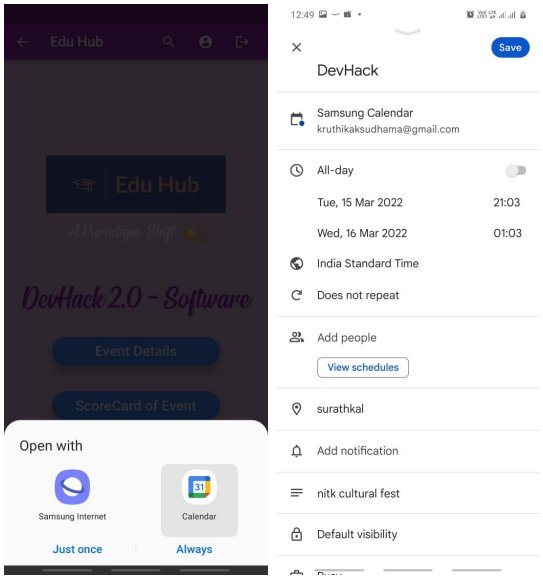
6. View Event details:

- The user has to click on an event on the home page and then click on the ‘Event Details’ button to view event details.
- It shows the event logo, description, location, price, timings, and other details about the event.



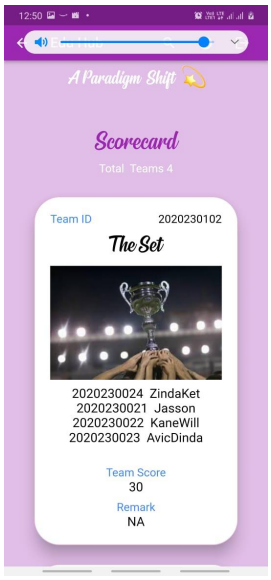
7. Add event to calendar:

- The event page has a button ‘Remind me’, the user can click on it to navigate to the default calendar app in the device and add the event.
- When the app navigates the user to the calendar app, it gives the user a scheduled event with all details filled, the user just has to click on the ‘save’ button on the calendar.



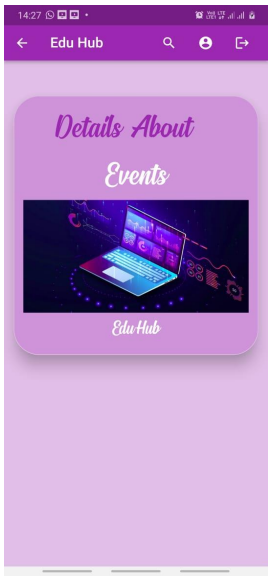
8. View scoreboard:

- The user can view the scoreboard by clicking on the ‘Scoreboard’ button on the event page.
- The teamwise scores are shown as updated by the invigilator.



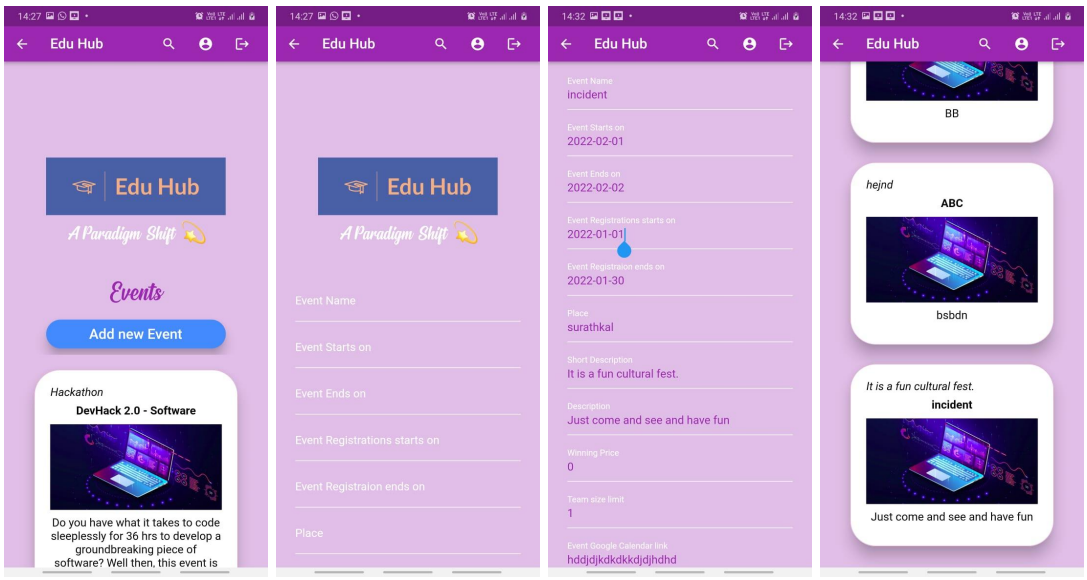
9. Admin HomePage:

The admin homepage contains an all events card which navigates to a page showing all events with an option to add events. It contains profile, search and logout options on the toolbar.



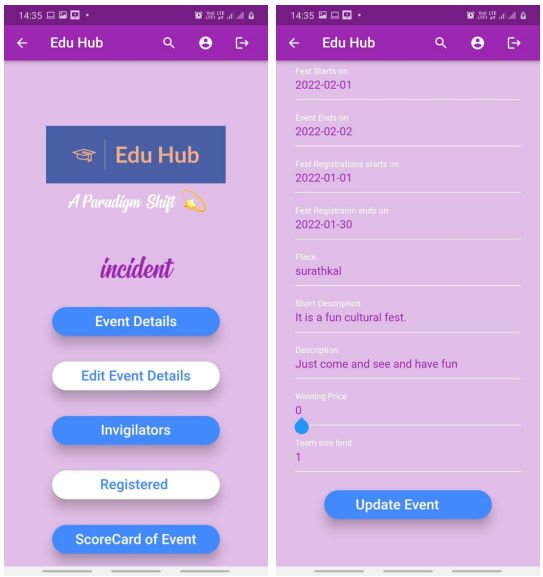
10. Add event:

On all events page, the admin has to click on the ‘Add event’ button. Then the user has to enter the details about the event such as event name, start date, end date, registration start date, registration end date, location, small description, description, number of members in a team, google calendar link. Then click on the ‘Save’ button to save the event.



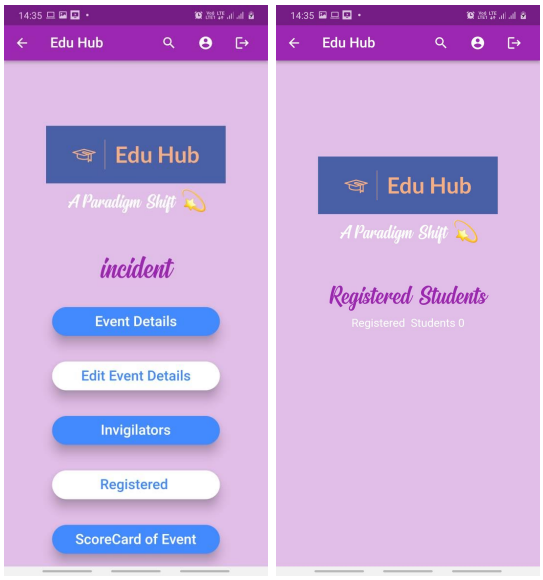
11. Edit event

The admin opens the event page and clicks on the ‘Edit event’ button. Then he can edit any detail that he wants to change and then click on ‘Save’ to save the changes.



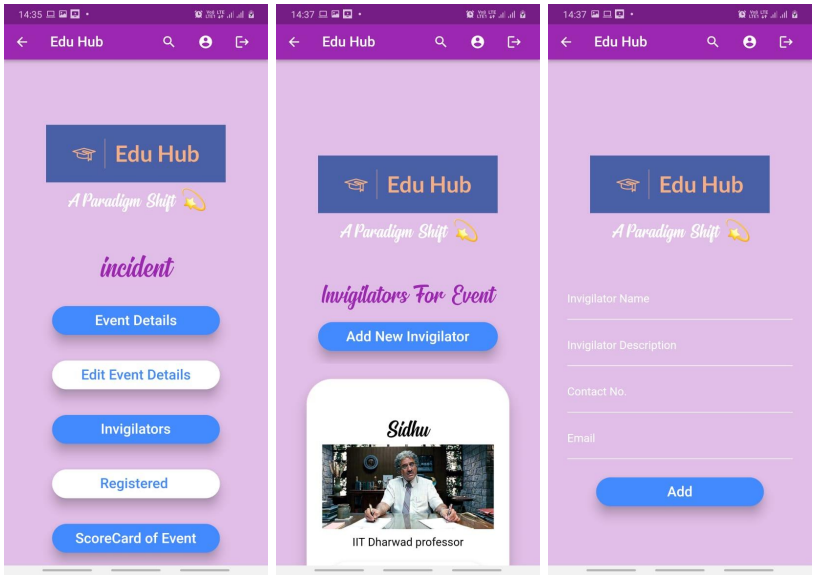
12. View Registered Students

The admin has to open the event page, then click on the ‘Registered’ button. It navigates the admin to view all the registered students.



13. Add invigilator

The user has to click on the ‘Invilators’ button on the event page. On this page he can also see the list of all invigilators. The user has to click on the ‘Add new invigilator’ button. Then he can fill in the details on the invigilator like name, description, contact and email and click on ‘Add’ to add the invigilator.



File Structure:

Authorization	SignIn.dart	Sign In Page
	SignUp.dart	Sign Up Page
	ForgotPassword.dart	Forget password Page
	PostgresKonnection.dart	Setting up postgres connection
	authenticate.dart	Authenticate User
Admin	AdminAddAEvent.dart	Publish events
	AdminEachEventDetails.dart	Add event details
	AdminEventDetails.dart	View event details
	AdminEventDetailsUpdate.dart	Update event details
	AdminFestDetails.dart	View Fest details
	AdminInvigilatorDetailsUpdate.dart	Update Invigilator Details
	AdminInvigilatorsForEvent.dart	View Invigilator
	AdminRegisteredStudents.dart	View registered participants
	AdminScorecard.dart	View Score Card
	AdminTeamDetails.dart	View Teams
Participant	ParticipantEachEvents.dart	Participant event details
	ParticipantJoinTheEvent.dart	Event joining page
	ParticipantRegistration.dart	Event Registration
Common Pages	EventDetails.dart	Event Details
	FestDetails.dart	Fest Details
	InvigilatorsForEvent.dart	Invigilators for Each event
Widgets	Widgets.dart	Widgets for entire App

6. CONCLUSION

We have learned a lot from this project on how to develop Android Application and publish it in real-time, using Web Services like AWS, firebase, integrating them into the flutter application. We also learnt the best practices in Software development, various SDLC models we could use, and how to work in teams.