SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of india)
Re-Accredited by NAAC with 'A' Grade
॥वसुधैव कुटुम्बकम्॥

PROJECT REPORT

(Semester: JUL 2020 - DEC 2020)

Project Title: **WEBINAR SHARING PLATFORM**

Submitted by

TRIPTI SINGH

PRN 18070124074

IT 18-22

Under the Guidance of

Prof. Preeti Mulay

Associate Professor,

SIT Pune

Department of Information Technology
SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

INDEX

PROJECT OVERVIEW

PROJECT PURPOSE

This is an online application-based system for sharing details about upcoming webinars and signing up for the current available ones. This software solution system aims to facilitate easy flow of information and increase participation of interested candidates.

It aggregates the people on an application-based platform, so they do not miss out on exciting live events and get great exposure to participate in the enlisted webinars. It allows the user to add latest upcoming webinars; the ones they are aware of from their end. This new system will upgrade the current email sharing for webinar advertisement platform to provide users with customized searches of the webinar catalogue and the ability to register easily.

This document describes the scope, objectives, and goal of the new system. In addition to describing non-functional requirements, this document models the functional requirements with use cases, UML diagrams interaction diagrams, and class models. This document is intended to direct the design and implementation of the target system in an object-oriented language.

PROJECT SUMMARY

Project name: Webinar Sharing Platform

As of now in the initial stage of development of this project I, Tripti Singh will be managing the development of this entire software which includes the managing, analysing, and fulfilling user testing. I will be the admin of this project.

PROJECT OBJECTIVES

- The system should have login. The login box appears when system is invoked.
- The user should be able to register for the webinar.

- The user should be able to view the details of all the available webinars.
- The user should be able to add webinars in a favourite list. i.e. shortlist webinars.
- The user can edit and delete an existing webinar.
- The user can add a webinar of his choice to the database.
- The validity of data should be maintained by the system.
- The admin has the ultimate authority of the webinar and user data.

PROJECT GOALS

- Planned and strategic development skills
- End to end documentation delivery
- Project code development snippets with detailed explanation for future reference.
- Accuracy and Integrity maintenance for the entire software project.
- Reliability of the system is ensured.
- Data integrity and redundancy maintained by storing data in categorized data stores.

SECTION 1

1.1 Introduction

This is a software application development project designed exclusively for its user to undergo ease of accessibility to get hands on the free live streaming content available on the Internet. It enables its users to view the upcoming webinar events, register for the same and add any event for others to view and participate.

1.2 Problem Statement

Webinar is the new way of communicating, promoting, and expressing ideas. It is not just a small domain anymore. With the world locked up, it has become nothing but a source of knowledge sharing platform for everyone. It is a struggle to get the target audience for a particular webinar and vice versa. People must collect information, send email invitations, and

go through a series of time-consuming activities to get audience only to realise that their efforts were in vain.

1.3 Significance

With webinars becoming the heart and soul of digital marketing and promoting business growth, I believe a system like this would be helpful to everyone. This software brings solutions to all these domains. From education to entertainment it will be the virtual assistant that would keep track of everything making sure that none of the potential information goes unnoticed.

1.4 Motivation

Connecting the right audience to the right set of webinars by becoming a service provider or medium between the users and providers of content is the motivation behind this project. This app enables live events to become reachable to everyone using a single platform.

I take this project personally as I see many start-ups struggling every day and spamming my mailbox with overflowing webinar invitations. I see this as an opportunity to help these hardworking people get to the correct audience and reach their potential.

1.5 Limitations

Problems with the current system include

- the information available on the public app is too limited and the user cannot immediately decide to sign up for the webinar
- the existence of coming up event information is often inconsistent or incorrect.
- Ambiguity and duplicity of user and webinar information should be prevented.

1.6 Project Scope

- The scope of this project is an app-based system that supports the marketing of upcoming webinar events. Get the interested users to register without confusion.
- The user can put up their own event for other people to see and register.

SECTION 2

SYSTEM PURPOSE

Users

Those who will primarily benefit from the new system and those who will be affected by the new system include

Customers:

> Upon implementation of the new system, customers will find webinar navigation, content identification and registration process easier. Customers will be able to choose the right event for themselves and choose a creative way to spend their time.

Webinar Hosts:

> Webinar hosts will be allowed to maintain the data about their participants directly. This will boost their understanding about their content delivery and help them understand about the interested viewers.

Location

The system will be available to any potential customer using the Internet. Hosts will be able to access restricted areas of the site through a password protection scheme.

Responsibilities

The primary responsibilities of the new system:

1. Allow the user to login and sign up or register into the system.
2. Give options to the user to select the webinar of his/her choice.
3. Enable user to select the event of interest and guide the user towards the registration page.
4. The user can add any webinar that they would like others to register for through this platform.

The desired responsibilities of the new system:

1. Consistent look and feel throughout the pages.
2. Search query section
3. Customer support for navigation [FAQ section]

Need:

This system is needed to service the expected increase in demand of the interested users to sign up for the upcoming webinars. Webinar advertisement is a major industry in the digital marketing business. A tool as such can come in handy for many organisations that schedule events that are beneficial for their users. Moreover, from the public point of view the system focuses on bringing the people together to share knowledge and grow more by providing maximum transparency and ease. The new system will allow hosts to rapidly increase sales without a large and expensive increase in the number of sales agents and other customer support digital marketing agencies.

SECTION 3

Methodology for developing the system

SDLC model used

The software development model that is used for the development of webinar sharing platform is:

RAD (Rapid Application Development Model)

This model is a suitable choice for the project under consideration due to the following reasons:

1. This model helps me to stick to the time constraint that is required for this project development i.e. (2-3 months).

2. It allows me to modularize and maintain various aspects of my project efficiently. Hence, facilitating the smooth development and transitioning of project from one phase to another.

   The software can be developed in chunks as per the listed requirement and integrated together during testing.

3. This project needed an iterative development approach and prototyping, which are the key features of RAD Model.
4. Since this project will require constant user feedback about its proper functioning, it will require improvement on each stage to ensure, the delivery of the best possible product to the client.

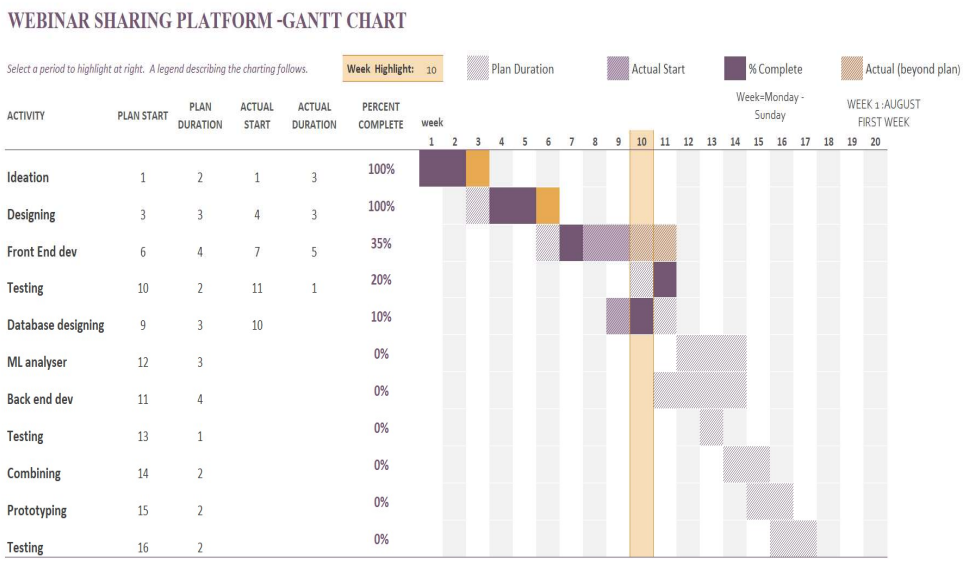
   The constant user-testing feature in RAD model becomes beneficial for this project.

5. The technical risk involved in using RAD is less. Hence, the project becomes flexible and adaptive to various new changes that can be introduced in various phases of development.
6. It reuses many of its resources which is a key requirement in this project as different modules will have the same infrastructure and utilise the same design. This will save time and increase efficiency of the project by making use of reusability of resources.

Testing methodology

Thorough testing was done at every stage of development of this project for ensuring the modules compatibility with the integrated system.

Proposed timeline followed for the project:



Section 4

FUNCTIONAL REQUIREMENTS

High Priority:

1. Login to the webinar sharing platform system mandatory for all users.
2. Register or Signup into the system if not already registered.
3. Add an upcoming webinar event to the database.
4. Select a webinar to view details of the webinar.
5. Register for an upcoming webinar.

Medium Priority:

1. Edit /Delete webinar details
2. Show guiding notifications and prompts for certain features.

Low Priority:

1. Adding is_Favourite criteria for the user to shortlist most liked webinars.

2. Manual/instruction Guides for user

Section 5

NON-FUNCTIONAL REQUIREMENTS

Performance Metrics

- The system shall be completely operational and load in min 5 sec.
- Down time after a failure shall not exceed 3-4 hours.

Usability

- Any user should be able to easily operate the system i.e. the design of the system should be extremely user friendly.

Reliability

- The Maintain the data integrity of the system by avoiding fake webinar details, reducing data redundancy.
- Privacy control over the data of the users and webinar details.
- Security of the system by making login mandatory for all the users of the system.

Security

- The system shall provide password protected access to pages that are to be viewed only by admin.
- Transaction data must be transmitted in encrypted form. In case of transactional requirements.

Supportability

- The system should be able to accommodate new webinars and details without major reengineering.
- The system web application should be available on the play store of android.

Online user Documentation and Help

- The system shall provide a web page that explains how to navigate the app. This page should be customized based on what pages that user can access.
- This help page should be accessible from all other pages in the app.

Purchased components

• API may be purchased.

Interfaces

The User interface should be colourful attractive.

The system must interface with:

• Firebase

• API(Gmail)

Section 6

REQUIREMENTS

Business requirement

The system would save time by automating the aggregating all the live webinar-based information on a single platform.

Operational requirements

✓ The Firebase database must be regularly maintained.

✓ The account security must always be intact.

✓ Un-Used accounts must be discarded.

✓ Help if required must be offer as fast as possible.

✓ Maintenance and updating regularly.

Technical requirement

✓ Firebase should always be actively working.

✓ The application must run in background always.

✓ The API should be updated and maintained.

✓ Packages must be actively updated.

Platform/technology

• Dart

• Flutter

• Android studio

• Firebase

• Dart Packages

• API (email): mailer package

Online tools

• Smart Draw

• Lucid Chart

• Draw.io

• Stack overflow (Help a lot in code related doubts)

Hardware requirements

• A laptop/desktop with min 4 GB RAM (8GB recommended)

• 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5

GB for Android SDK and emulator system image)

• 1280 x 800 minimum screen resolution.

Beneficiaries

User -Anyone can register for an upcoming webinar event. It can be modified to provide this selective allowance to a particular group of organisations. Depends on the application and the need of the hour.

Host – Anyone can offer to put up their event or an event they are aware of for others to view and participate. It must be verified by the user as a prerequisite.

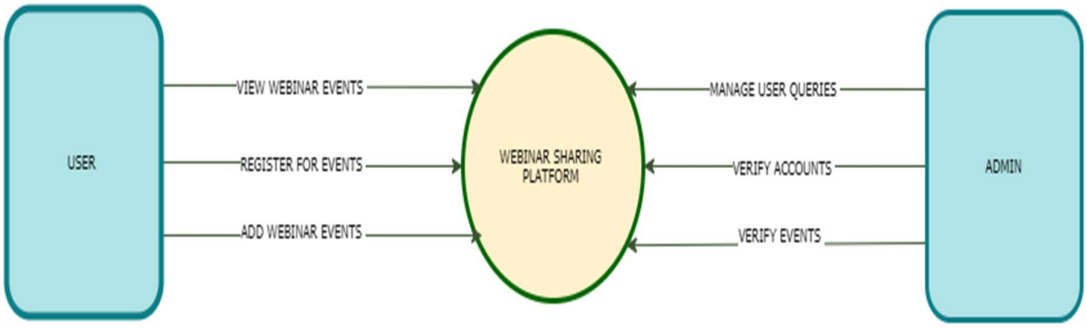
Section 7

CONTEXT DIAGRAMS

Goal Statement:

- ADD WEBINAR EVENTS
- VIEW WEBINAR EVENTS
- REGISTER FOR WEBINAR EVENTS

Context Model:



System Externals:

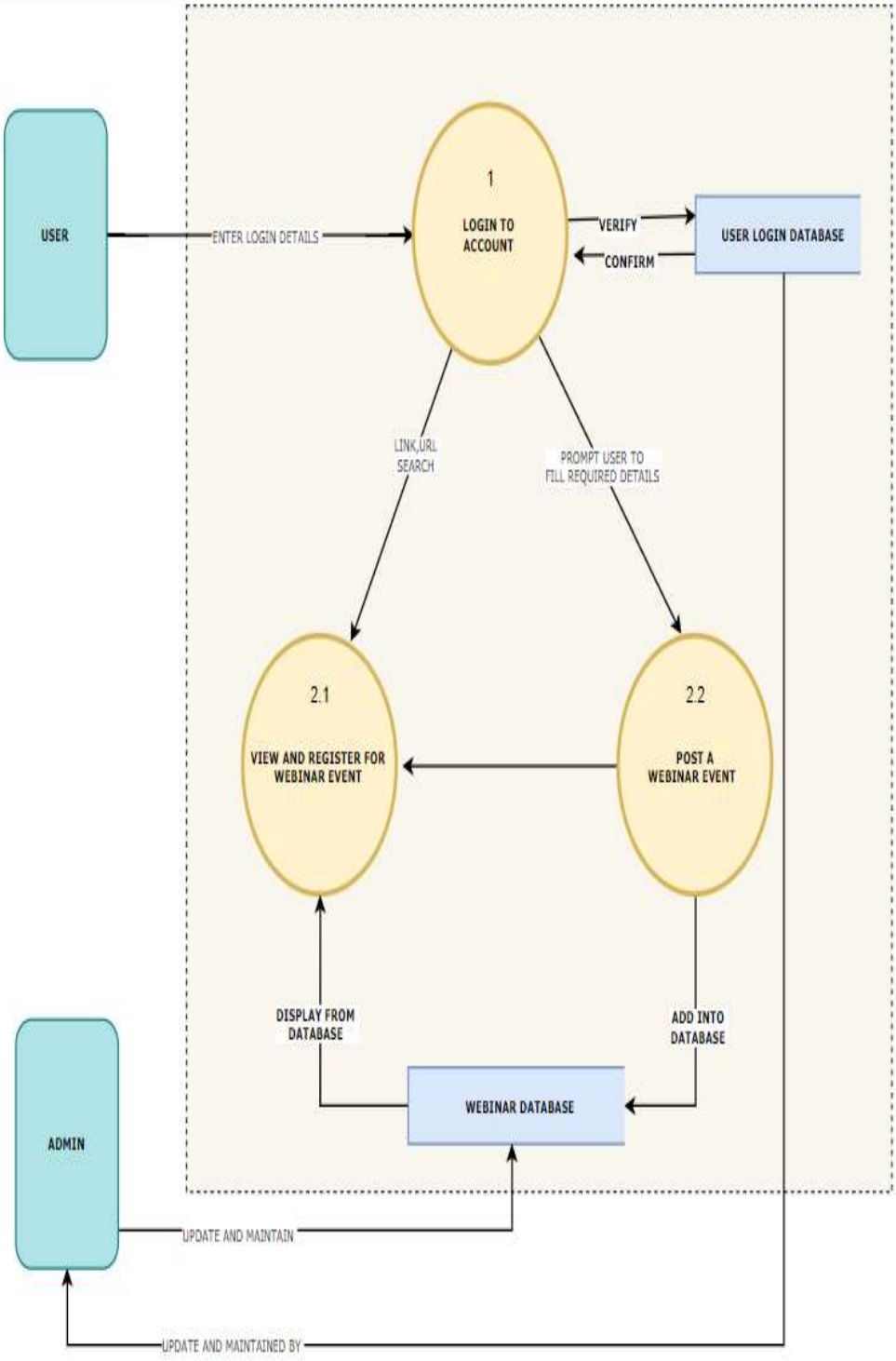
- USER:

  The user is anyone who wants to sign up or register for a webinar event or add a webinar into the database.

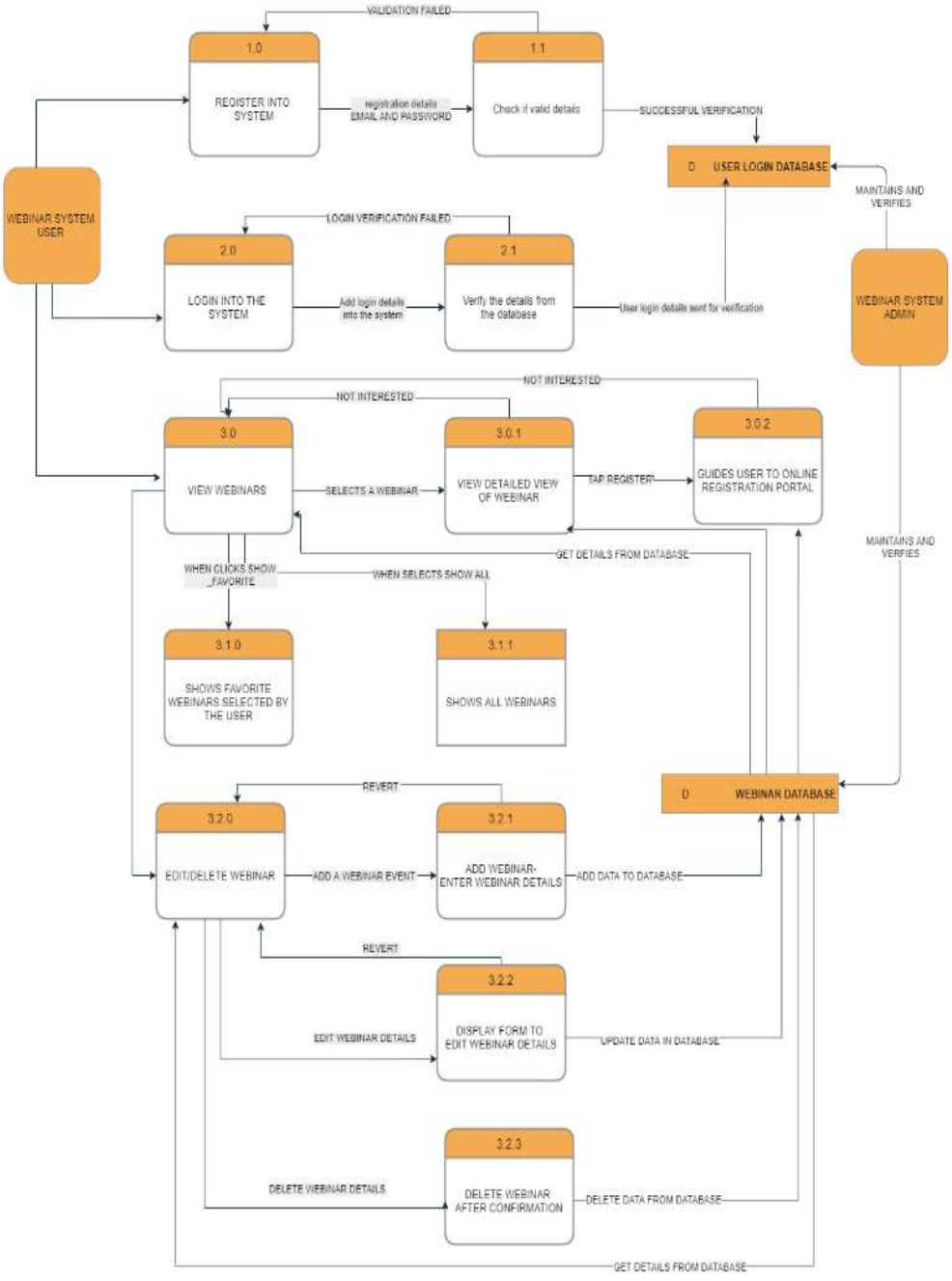
- ADMIN:

  The systems data integrity and privacy is maintained by the admin.The admin has the highest access level to alter data , update info and delete info.

DFD LEVEL 1


USER
ENTER LOGIN DETAILS
1
LOGIN TO ACCOUNT
VERIFY
CONFIRM
USER LOGIN DATABASE
LINK,URL SEARCH
PROMPT USER TO FILL REQUIRED DETAILS
2.1
VIEW AND REGISTER FOR WEBINAR EVENT
2.2
POST A WEBINAR EVENT
DISPLAY FROM DATABASE
ADD INTO DATABASE
WEBINAR DATABASE
ADMIN
UPDATE AND MAINTAIN
UPDATE AND MAINTAINED BY

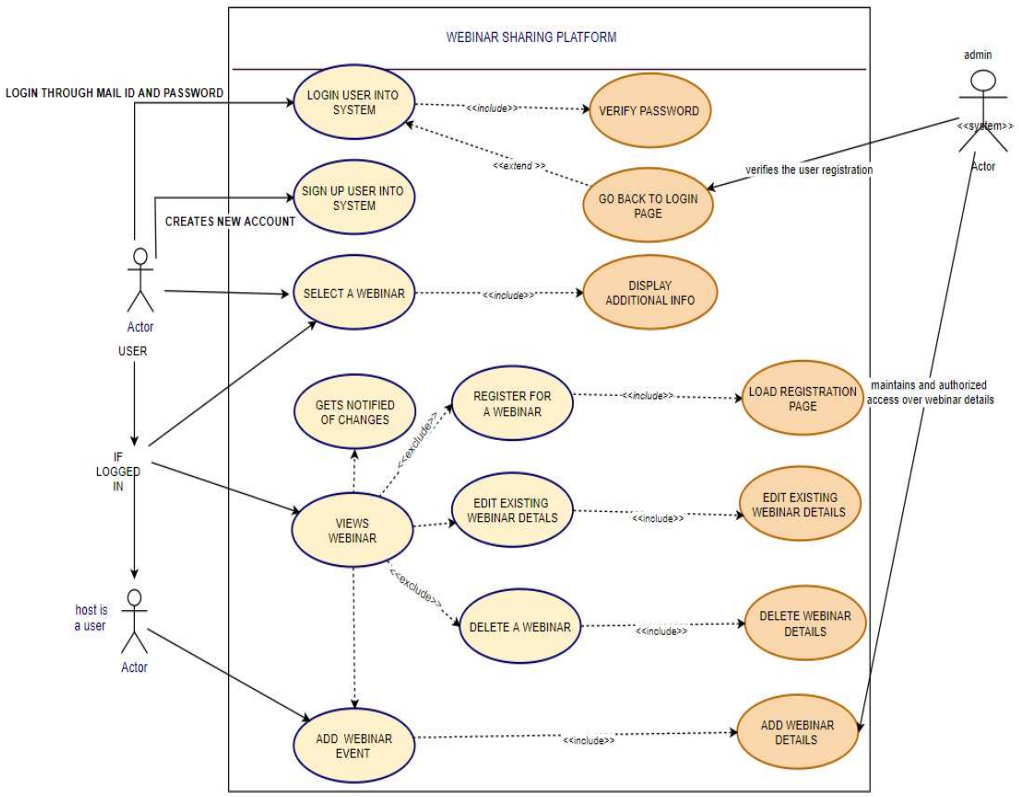
DFD LEVEL 2


VALIDATION FAILED
1.0
REGISTER INTO SYSTEM
registration details EMAIL AND PASSWORD
1.1
Check if valid details
SUCCESSFUL VERIFICATION
D USER LOGIN DATABASE
MAINTAINS AND VERIFIES
WEBINAR SYSTEM USER
LOGIN VERIFICATION FAILED
2.0
LOGIN INTO THE SYSTEM
Add login details into the system
2.1
Verify the details from the database
User login details sent for verification
WEBINAR SYSTEM ADMIN
NOT INTERESTED
NOT INTERESTED
3.0
VIEW WEBINARS
SELECTS A WEBINAR
3.0.1
VIEW DETAILED VIEW OF WEBINAR
TAP REGISTER
3.0.2
GUIDES USER TO ONLINE REGISTRATION PORTAL
GET DETAILS FROM DATABASE
MAINTAINS AND VERIFIES
WHEN CLICKS SHOW _FAVORITE
WHEN SELECTS SHOW ALL
3.1.0
SHOWS FAVORITE WEBINARS SELECTED BY THE USER
3.1.1
SHOWS ALL WEBINARS
D WEBINAR DATABASE
REVERT
3.2.0
EDIT/DELETE WEBINAR
ADD A WEBINAR EVENT
3.2.1
ADD WEBINAR- ENTER WEBINAR DETAILS
ADD DATA TO DATABASE
REVERT
3.2.2
EDIT WEBINAR DETAILS
DISPLAY FORM TO EDIT WEBINAR DETAILS
UPDATE DATA IN DATABASE
3.2.3
DELETE WEBINAR DETAILS
DELETE WEBINAR AFTER CONFIRMATION
DELETE DATA FROM DATABASE
GET DETAILS FROM DATABASE

Section 8

USE CASE MODEL

SYSTEM UML USE CASE DIAGRAM



Use case descriptions for selected cases

- ✓ User Sign up -Registration
- ✓ User Login into System
- ✓ Select Webinar
- ✓ Register for webinar
- ✓ Edit/Delete Webinar
- ✓ Add Webinar

Point to note:

- For all use cases, the user can cancel the use case at any step that requires user input. This action ends the use case. Any data collected during that use case is lost.
- For all use cases that require a logged in user, the current login session is updated during the use case to reflect the navigation paths through the use case.

Register User-Sign up

Use Case Name:	Register User into the system
Summary:	To have access to the app details a new user must register a username and password.
Basic Flow:	1. The use case starts when a user indicates that he wants to register. 2. The system requests a username and password. 3. The user enters a username and password. 4. The system checks that the username does not duplicate any existing registered usernames. 5. The system requests username in the mail format 6. The user enters the information. 7. The system determines the access level and stores all user information in the server database. 8. The user is guided to the view webinar screen
Alternative Flows:	• Step 4: If the username duplicates an existing username the system displays a message, and the use case goes back to step 2. • Step 5: If the user does not enter a required field, a message is displayed, and the use case repeats step 2.

Extension Points:	*none*
Preconditions:	*none*
Postconditions:	The user can now obtain data and perform functions according to his registered access level.

Login user into the system

Use Case Name:	Login User
Summary:	To get to access to the main page of the application the user needs to login in the system
Basic Flow:	1. The use case starts when a user indicates that he wants to login. 2. The system requests the username and password. 3. The user enters his username and password. 4. The system verifies the username and password against all registered users. 5. The system starts a login session.
Alternative Flows:	Step 4: if username or password is invalid, the use case goes back to step 2.
Extension Points:	none
Preconditions:	The user is registered.
Postconditions:	The user can now obtain data and perform functions according to his registered access level.
Business Rules:	All the data is available only to the admin level users.

Select Webinar Event

Use Case Name:	Select a Webinar Event Scenario: User is interested to view the webinar details.
Summary:	This use case allows a registered user to view the webinar details and register for the same.
Basic Flow:	1. The use case starts when a customer indicates he wants to view more details about a certain event i.e. he selects a certain webinar event. 2. The system displays the event's or webinar's information: name, time, dates, description, speaker etc 3. The user can register for a particular event by selecting the "REGISTER" option. 4. The user is guided to the official registration page for that event. 5. The system displays a registration completion message and adds the details to the existing database.
Alternative Flows:	If the user is not satisfied with the webinar info and does not wish to proceed with the registration process the user can go back to the homepage and view other webinars.
Extension Points:	*REGISTRATION*
Preconditions:	The customer is logged in and has completed a search for the event to be selected.
Postconditions:	The user is registered for the event.

Business Rules:	The customer is a part of the webinar as he/she is registered as a participant.

Register for a webinar

Use Case Name:	Register
Summary:	This use case guides the customer to main registration page of the particular webinar via a online portal.
Basic Flow:	1. The use case begins when a user selects "Register" as a payment option, while in use case *Select.* 2. The system requests confirmation. 3. The system verifies that the user details as valid and proceeds to navigation to the main page.
Alternative Flows:	Step 4: If the user wants to go back to the main view page the user can proceed to do so.
Extension Points:	none
Preconditions:	The system is executing use case Select Webinar.
Postconditions:	The customer is registered for the event.
Business Rules:	None

Modify or delete webinar details

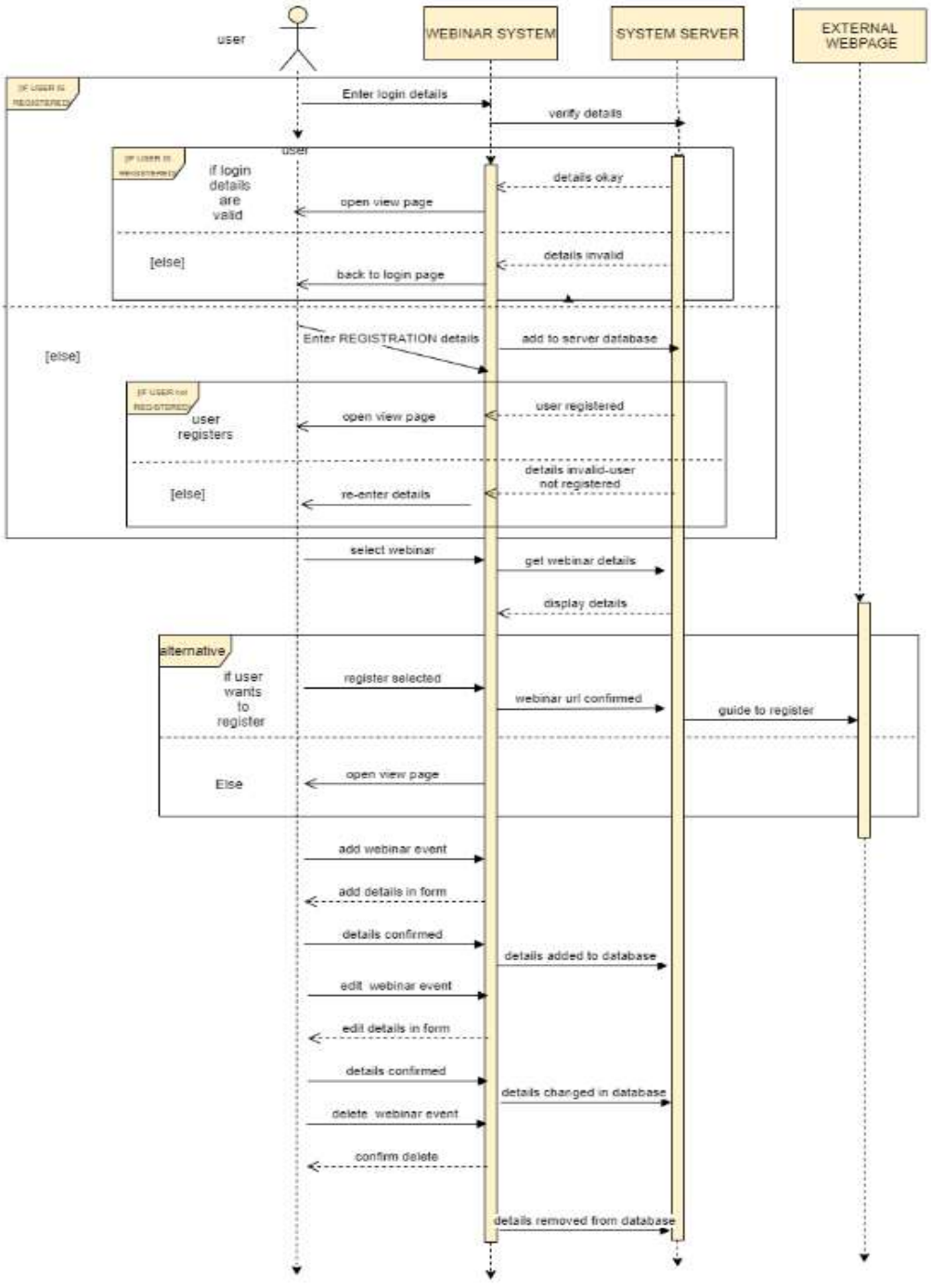
Use Case Name:	Edit/Delete Webinar Event Scenario: The user wants to edit or delete a pre-existing webinar.

Summary:	This use case allows the user to edit or delete webinar details that already exist in the system
Basic Flow:	1. The use case starts when he selects the edit button in the homepage 2. The form page pops up where user edits the webinar details. 3. The system requests confirmation. 4. The system prompts the user to complete all the mandatory information details. 5. The user selects okay and the information about the edited webinar gets added into the system. 6. If user selects the delete button. 7. The system prompts the user to confirm 8. And the data gets deleted.
Alternative Flows:	1. If the user retreats or does not wish to add the details anymore he/she could navigate back to the edit/delete webinar page. 2. The user selects the add button then the user is guided to the add webinar page
Extension Points:	Add webinar is an extension of edit webinar page.
Preconditions:	The user is logged in, is on the edit webinar page.
Postconditions:	The webinar is edited or deleted and the details are updated accordingly into the database.
Business Rules:	The webinar info should be valid.

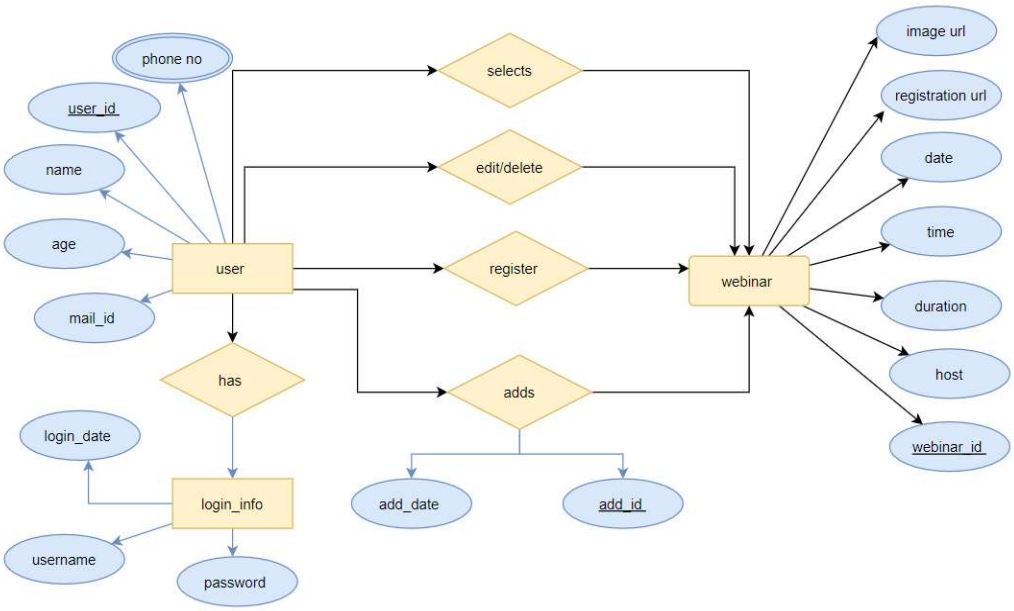
Add a webinar event

Use Case Name:	Add Event Scenario: The user wants to add a webinar to the existing webinar database.
Summary:	This use case allows the user to add upcoming webinar details for other interested viewers to view and participate.
Basic Flow:	9. The use case starts when he selects the add button in the homepage 10. The form page pops up where user fills in the criteria and the webinar details. 11. The system requests confirmation. 12. The system prompts the user to complete all the mandatory information details. 13. The user selects okay and the information about the new webinar gets added into the system.
Alternative Flows:	If the user retreats or does not wish to add the details anymore he/she could navigate back to the edit/delete webinar page.
Extension Points:	none
Preconditions:	The user is logged in, is on the edit webinar page.
Postconditions:	The webinar is added into the database.
Business Rules:	The webinar info should be valid.

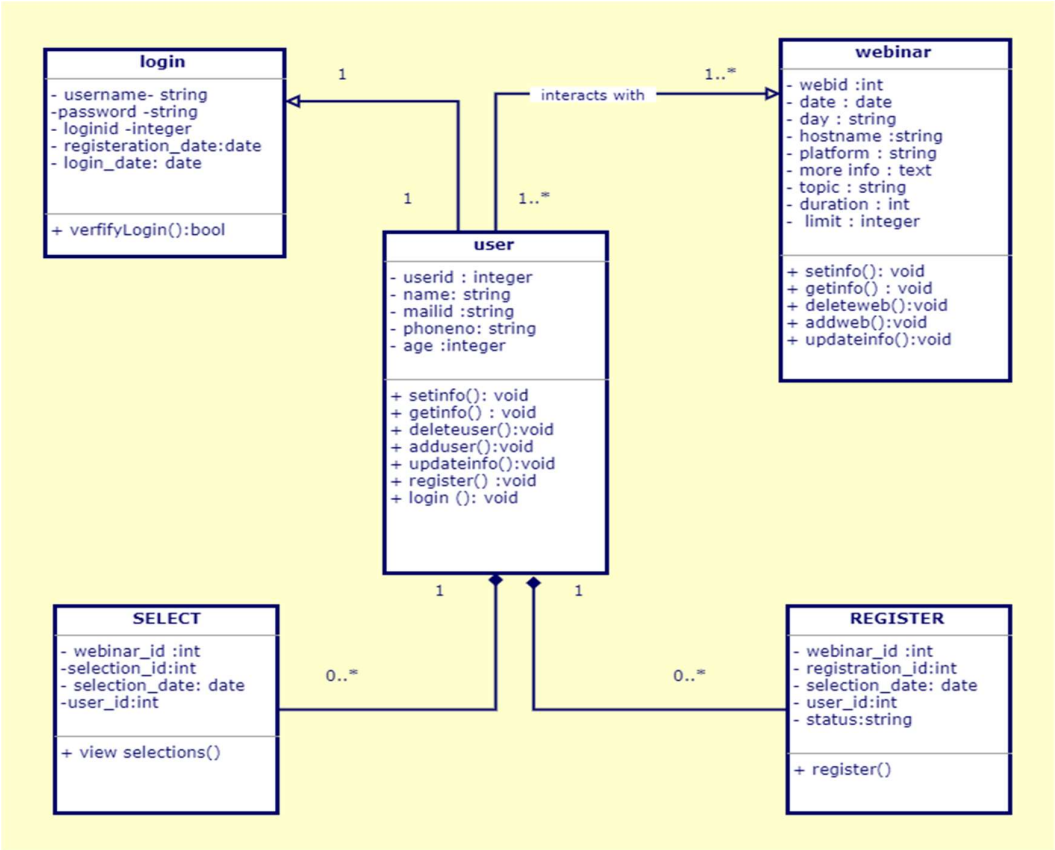
SECTION 9: SEQUENCE UML DIAGRAM



ER DIAGRAM



CLASS MODEL



Development Specification

The project was developed in modules. All of the project files is divided into different categories and sections.

The Phase 1 development and testing of the project is meant for the structural or schema development and the provision of basic functionalities only.

PHASE 1 DEVELOPMENT

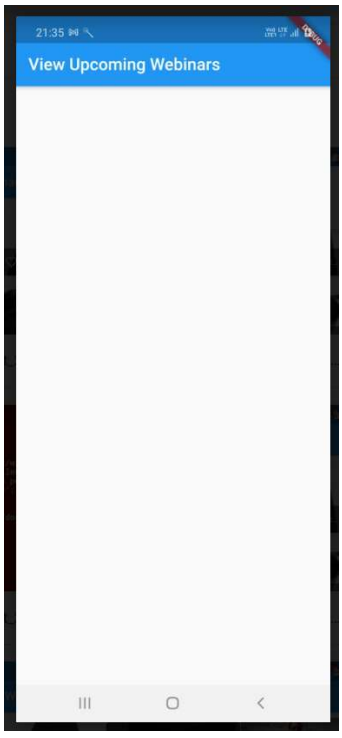
MODULE 1

WEBINAR OVERVIEW SCREEN

It mainly focused on the look and feel of the 'View Webinar' Page with the only added functionality of the "Show All" Webinars and "Show Favourites".

Stage 1 of development gave the following results:

```
PopupMenuButton(
  onSelected: (FilterOptions selectedValue) {
    setState(() {
      if (selectedValue == FilterOptions.Favourites) {
        _showOnlyFavourites = true;
      } else {
        _showOnlyFavourites = false;
      }
    });
  },
  icon: Icon(Icons.more_vert),
  itemBuilder: (_) => [
    PopupMenuItem(
      child: Text("Only Favorite"),
      value: FilterOptions.Favourites,
    ),  // PopupMenuItem
    PopupMenuItem(
      child: Text("Show All"),
      value: FilterOptions.All,
    ),  // PopupMenuItem
  ],
),  // PopupMenuButton
],  // <Widget>[]
),  // AppBar
```



This code snippet focuses on the filtering out the favourite webinars and adding the popup menu items for enabling the same selection.

In the next part of this module I decided to make the webinar class. It was understood that most of the details would be added by the user in some random format. Hence, all the instances or features are string values. ChangeNotifier is assigned to listen for changes in case of any updates in the details of a single webinar item. @required ensures all the properties are mandatory to mention.

webinar_item.dart | webinar_overview_screen.dart | webinar.dart | webina

```dart
import 'package:flutter/foundation.dart';

class Webinar with ChangeNotifier {
  final String webid;
  final String title;
  final String speaker;
  final String description;
  final String date;
  final String time;
  final String imageUrl;
  final String regurl;
  final String duration;

  bool isFavorite;

  Webinar({
    @required this.webid,
    @required this.title,
    @required this.description,
    @required this.date,
    @required this.time,
    @required this.regurl,
    @required this.imageUrl,
    @required this.speaker,
    @required this.duration,
    this.isFavorite = false,
  });

  void toggleFavoriteStatus() {
    isFavorite = !isFavorite;
    notifyListeners();
  }
}
```

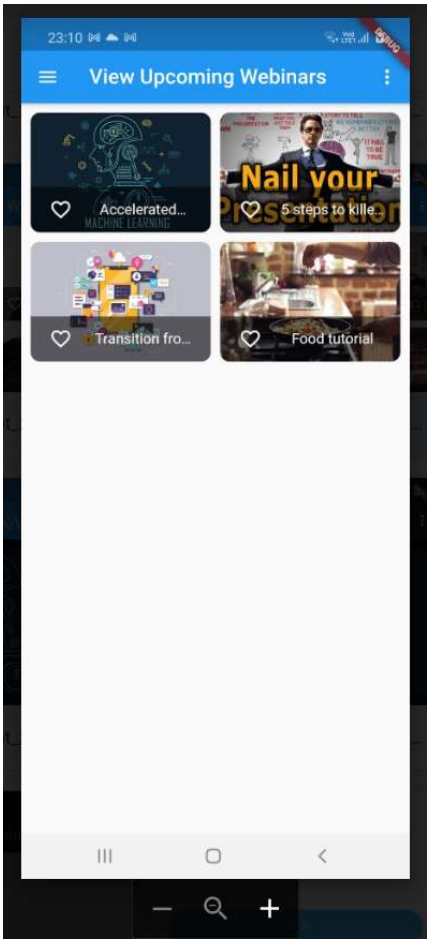
Now it was time to design our webinar widget, it is a customised widget that was built by combination of other widgets which will be rebuilt for every webinar that will be added to the overview screen by the user.

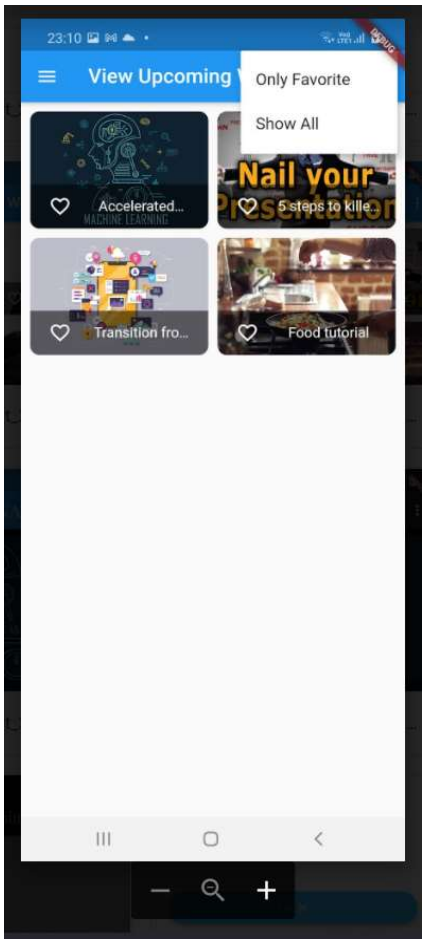
So, it is reusable code of our program that builds our webinar widgets.

```
child: GridTile(
  //type of widget

  child: GestureDetector(
    //invisible gesture detector which allows us to add listener so that
    onTap: () {
      Navigator.of(context).pushNamed(
        WebinarDetailScreen.routeName,
        arguments: product.webid,
      );
    },
    child: Image.network(
      //main object of gridtile is image
      product.imageUrl,
      fit: BoxFit.cover, //fits the image accurately in gridtile widget
    ),  // Image.network
  ),  // GestureDetector
  footer: GridTileBar(
    //displays additional info tab;can have header too
    leading: IconButton(
      icon: Icon(
        product.isFavorite ? Icons.favorite : Icons.favorite_border,
      ),  // Icon
      onPressed: () {
        product.toggleFavoriteStatus();
      },
    ), //displays a heart for favourite  // IconButton

    backgroundColor: Colors.black87, //shows a transparent colour
    title: Text(
      product.title, //displays the image title
      textAlign: TextAlign.center,
      style: TextStyle(
```

WebinarGrid was another widget developed to link up the webinar items widget together in the webinar overview screen i.e. display all of them in a group.

```dart
class WebinarGrid extends StatelessWidget {
  final bool showFavs;
  WebinarGrid(this.showFavs);
  @override
  Widget build(BuildContext context) {
    final webinarsdata = Provider.of<Webinars>(context);
    final webinars = showFavs?webinarsdata.favouriteItems:webinarsdata.items;
    return GridView.builder(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 1,
        childAspectRatio: 3/2,
        crossAxisSpacing: 10,
        mainAxisSpacing: 15,
      ), // SliverGridDelegateWithFixedCrossAxisCount
      itemBuilder: (ctx, i) => ChangeNotifierProvider.value(
        value: webinars[i], //value data of grid or list items ,provider attached to data easy for change
        child: WebinarItem(), //webinars[i].webid,
        //webinars[i].imageUrl,
        //webinars[i].title,
      ), // ChangeNotifierProvider.value
      padding: const EdgeInsets.all(10.0),
      itemCount: webinars.length,
    ); // GridView.builder
  }
}
```



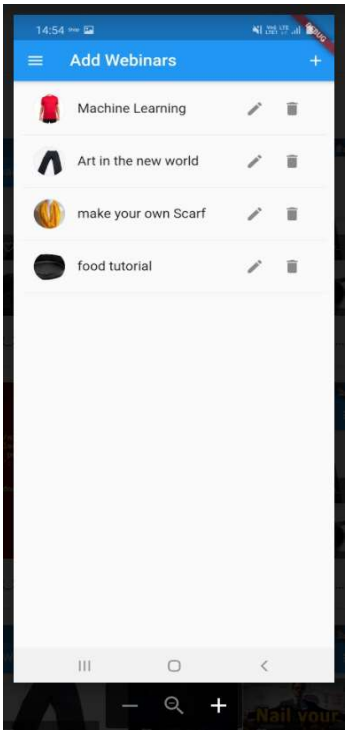


MODULE 2 PHASE 1 DEVOPLEMENT

The goal was to design the edit/delete webinar page. The screen with the same template as the webinar overview screen was developed and user webinar item widget was created in to put as elements in the following webpage.

webinar_item.dart | webinar.dart | webinars.dart | user_webinar_screen.dart

```
class UserWebinarScreen extends StatelessWidget {
  static const routeName = '/user-webinars';
  @override
  Widget build(BuildContext context) {
    final webinarsdata = Provider.of<Webinars>(context);
    return Scaffold(
      backgroundColor: Colors.black54,
      drawer: AppDrawer(),
      appBar: AppBar(
        backgroundColor: Colors.black87,

        title: Center(
          child: const Text('Edit Webinar',style: TextStyle(
            letterSpacing: 1,
            fontFamily: 'Playfair Display',
            fontSize: 17.0,
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),),  // TextStyle, Text
        ),  // Center
        actions: <Widget>[
          IconButton(
            icon: const Icon(Icons.add,color: Colors.white,),
            onPressed: () {
              Navigator.of(context).pushNamed(EditWebinarScreen.routeName);

            }),  // IconButton
        ],  // <Widget>[]
      ),  // AppBar
      body: Padding(
        padding: EdgeInsets.all(10),
        child: ListView.builder(
```




```
@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.only(top:20.0),
    child: ListTile(

      title: Text(title,style: TextStyle(
          fontFamily: 'PT Sans',
          fontSize: 15.0,
          color: Colors.white,
          fontWeight: FontWeight.normal),),  // TextStyle, Text
      leading: CircleAvatar(
        backgroundImage: NetworkImage(imageurl),
        radius: 40,
      ),  // CircleAvatar
      trailing: Container(
        width: 100,
        child: Row(
          children: <Widget>[
            IconButton(
              icon: Icon(Icons.edit,color: Colors.blue,),
              onPressed: () {
                Navigator.of(context)
                    .pushNamed(EditWebinarScreen.routeName, arguments: webid);
              }),  // IconButton
            IconButton(
              icon: Icon(Icons.delete,color: Colors.red,),
              onPressed: () {
                Provider.of<Webinars>(context, listen: false).deleteWebinar(webid);
              },
            ),  // IconButton
```

To add functionalities to the edit and delete icons that are shown here I added a webinars page that took in some dummy data to help us work and run tests in the project.

Dummy data placed In a list:

```
class Webinars with ChangeNotifier {
  //mixin with with
  List<Webinar> _items = [
    //private prop
    //some dummy data changeable of course
    Webinar(
      webid: 'p1',
      title: 'Accelerated GPUs in Data Science',
      speaker: 'B Senthilraj',
      description:
          'How Accelerated GPUs Help Data Scientists is a webinar hosted by Analytics India Magazine w
      duration: '45 min',
      imageUrl:
          'https://digitaltransformationtrends.com/wp-content/uploads/2020/01/machine-learning.jpeg',
      date: '30th December 2020',
      time: '4:00 PM - 4:15 PM IST',
      regurl: 'https://register.gotowebinar.com/register/4978682518697467918',
    ),
    Webinar(
      webid: 'p2',
      title: '5 steps to killer Presentation',
      speaker: 'Jason Tedeack',
      description:
          'Discover the secrets to giving a master presentation… techniques most presenters have never
      duration: '8 min',
      imageUrl: 'https://i.ytimg.com/vi/MnIPpUiTcRc/maxresdefault.jpg',
      date: '25th december 2020',
      time: '18.00 p.m.',
      regurl: 'https://www.youtube.com/watch?v=dEDcc0aCjaA',
```

```
  void updateWebinar(String webid, Webinar newWebinar) {
    final webinarindex = _items.indexWhere((prod) => prod.webid == webid);
    if (webinarindex >= 0) {
      _items[webinarindex] = newWebinar;
      notifyListeners();
    } else {
      print('...');
    }
  }

  void deleteWebinar(String webid) {
    _items.removeWhere((prod) => prod.webid == webid);
    notifyListeners();
  }
}
```

To edit and delete webinar the following functions were made use of.

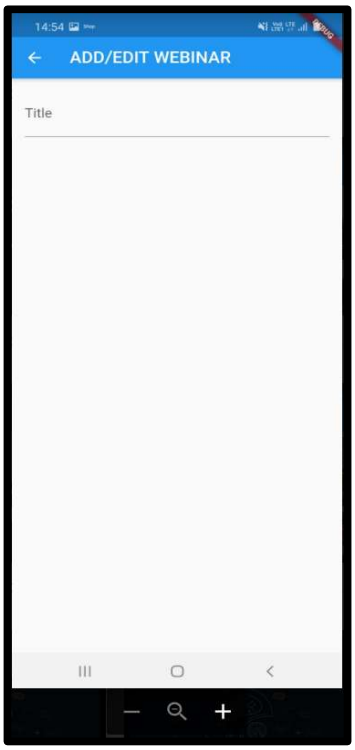
To maintain the consistency of the selected favourites webinar I decided to add them as functions to the webinars page as well.

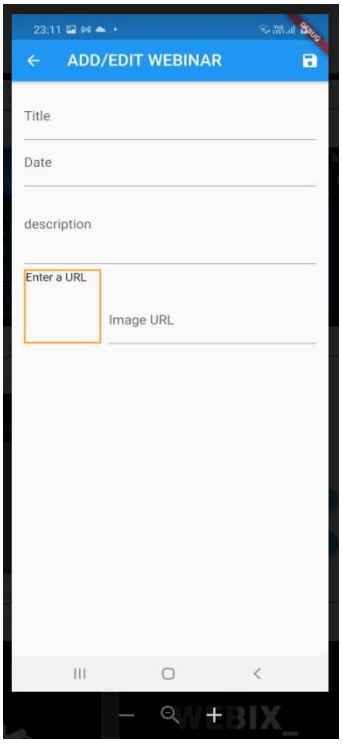
```
List<Webinar> get items {
  // if(_showFavouritesOnly){
  //  return _items.where((WebinarItem)=>WebinarItem.isFavourite).toList();

  return [..._items];
}

List<Webinar> get favouriteItems {
  return _items.where((WebinarItem) => WebinarItem.isFavorite).toList();
}
//void showFavouritesOnly(){
// _showFavouritesOnly=true;
// notifyListeners();

//}
//void showAll(){
// _showFavouritesOnly=false;
//   notifyListeners();
// }

Webinar findbyId(String webid) {
  return _items.firstWhere((prod) => prod.webid == webid);
}
```

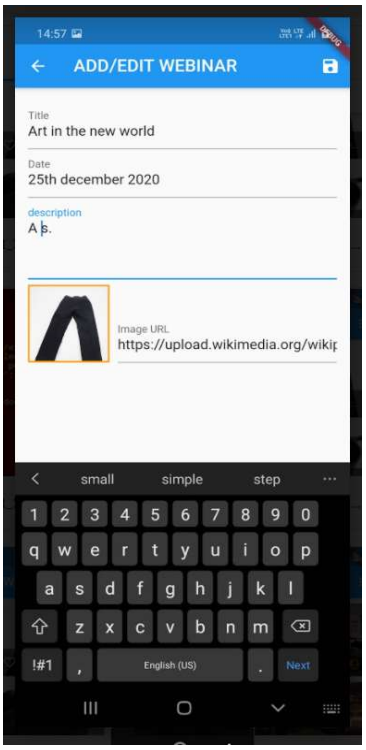
Module 3 phase 1

The next challenge was to add a page that could add the webinar details that are entered by the user and display it in the database. To do so I needed to add one more screen for add webinars. This could be accessed through the edit webinar page only. Hence, the screen was created as edit webinar screen with the simple form format to give the following as output.

```
child: Form(
  key: _form,
  child: ListView(
    children: <Widget>[
      TextFormField(
        initialValue: _initvalues['title'],
        decoration: InputDecoration(labelText: 'Title'),
        textInputAction: TextInputAction.next,
        onFieldSubmitted: (_) {
          FocusScope.of(context).requestFocus(_priceFocusNode);
        },
        validator: (value) {
          if (value.isEmpty) {
            return 'Please enter the title';
          }
          return null;
        },
```





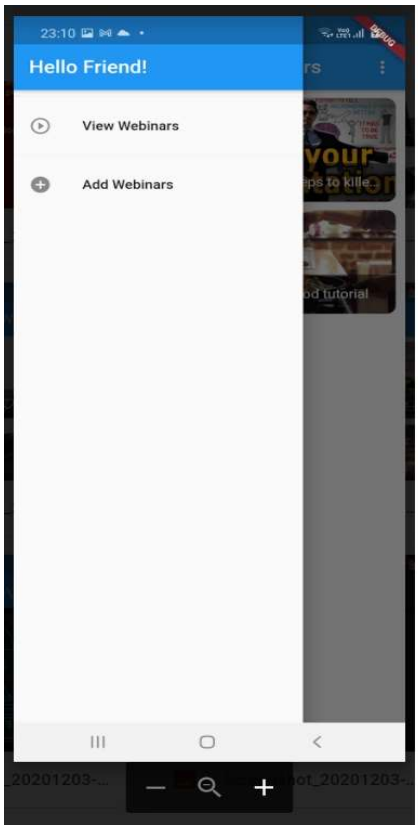


Add webinar function was added in the webinars dart file where the update and delete functions were present.

```
Future<void> addWebinar(Webinar webinar) async {
  const url = 'https://webix-23a00-default-rtdb.firebaseio.com/webinars.json';
  //firebase database url,attached package name is webinars,should be in json format
  try {
    final response = await http.post(
      url,
      body: json.encode({
        'title': webinar.title,
        'speaker': webinar.speaker,
        'description': webinar.description,
        'duration': webinar.duration,
        'imageUrl': webinar.imageUrl,
        'date': webinar.date,
        'time': webinar.time,
        'regurl': webinar.regurl
      }),
    );
    print(json.decode(response.body));
    final newWebinar = Webinar(
        webid: json.decode(response.body)['name'],
        title: webinar.title,
        speaker: webinar.speaker,
        description: webinar.description,
        duration: webinar.duration,
        imageUrl: webinar.imageUrl,
        date: webinar.date,
        time: webinar.time,
        regurl: webinar.regurl);
    _items.add(newWebinar);
    //_items.insert(0, newWebinar) //FOR THE START OF THE STRING
    notifyListeners();
  } catch (error) {
    print(error);
    throw error;
```

Next I built an app drawer for easy traversal of the user amongst different screens.

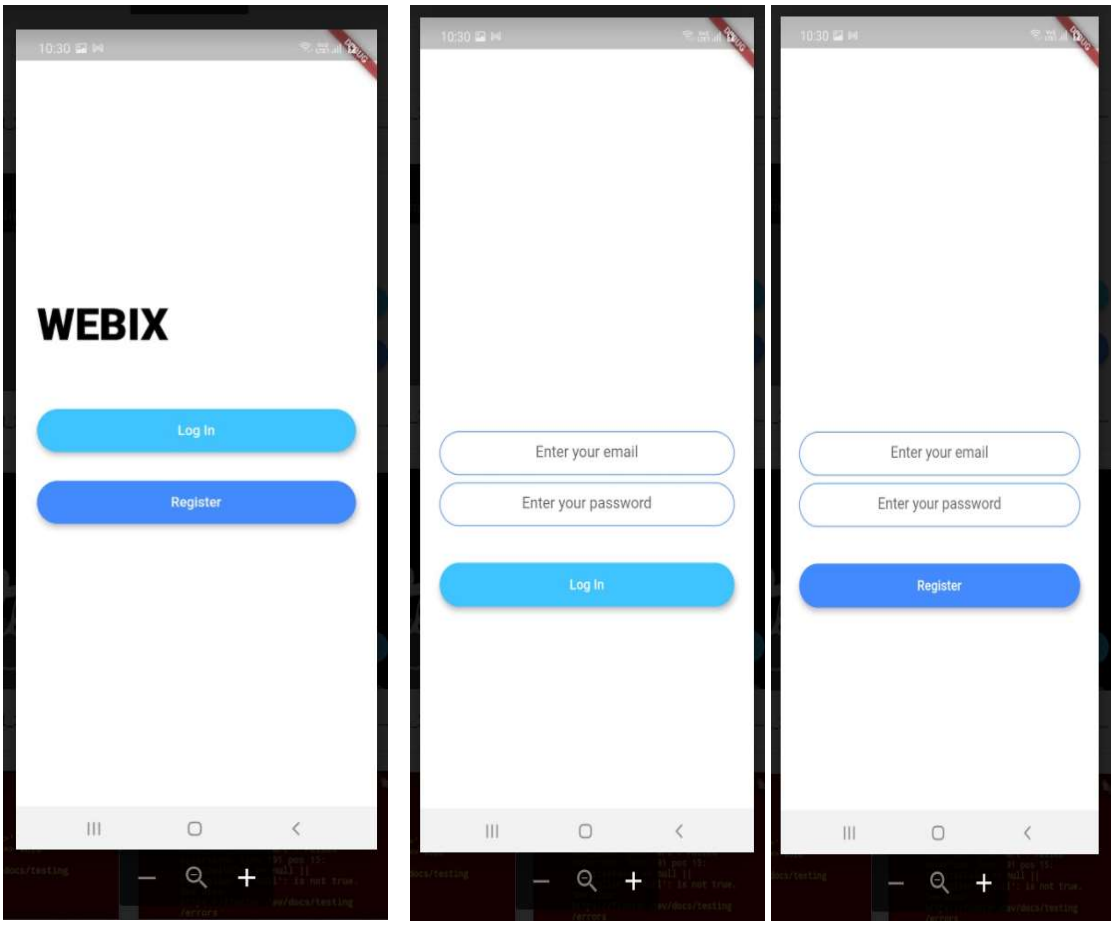
```
@override
Widget build(BuildContext context) {
  return Drawer(
    child: Column(
    children: <Widget>[
      AppBar(
        backgroundColor: Colors.grey[900],
        title: Text(
          '  Hello Friend !',
          style: TextStyle(
            fontFamily: 'PT Sans',
            fontSize: 17.0,
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ), // TextStyle
        ), // Text
        automaticallyImplyLeading: false,
      ), // AppBar

      ListTile(
        leading: Icon(Icons.play_circle_outline_outlined),
        title: Text('View Webinars'),
        onTap: () {
          Navigator.of(context).pushReplacementNamed(WebinarOverviewScreen.id); //heres an error
        },
      ), // ListTile
      Divider(color: Colors.grey[800],thickness: 3,),
      ListTile(
        leading: Icon(Icons.add_circle),
        title: Text('Add Webinars'),
        onTap: () {
          Navigator.of(context).pushReplacementNamed(
              UserWebinarScreen.routeName); //heres an error
        },
      ), // ListTile
```



A very important feature of the app was to provide webinar details when a certain item was clicked in the webinar overview screen. It finds the webinar by id and displays its details.

```
ModalRoute
    .of(context)
    .settings
    .arguments as String; // is the id!
final loadedProduct = Provider.of<Webinars>(
  context,
  listen: false,
).findbyId(webid);
var s =loadedProduct.regurl;
return Scaffold(
  backgroundColor: Colors.black87,
  appBar: AppBar(
    backgroundColor: Colors.grey[900],
    title: Text(loadedProduct.title),
  ), // AppBar
  body: SingleChildScrollView(
    child: Column(
      children: <Widget>[

      Container(
        height: 250,
        width: double.infinity,
        child: Image.network(
          loadedProduct.imageUrl,
          fit: BoxFit.cover,
        ), // Image.network
      ), // Container
      SizedBox(height: 10),
```

Module 4 development



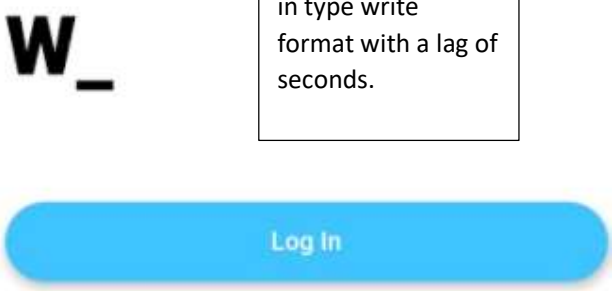
The module 4 development was focused on designing the welcome, login and register or sign up page for the user.

The rounded_button is a customised component or widget material is created in order to have these beautifully styled login and registration buttons. Constants is a file that maintains some of the functionalities that will remain constant to all these three pages.

```
import 'package:animated_text_kit/animated_text_kit.dart';
```

is a package that is installed to give animation like features to my project.

```
Center(
  child: TypewriterAnimatedTextKit(
    text: ['WEBIX'],
    textStyle: TextStyle(
      letterSpacing: 2,
      color: Colors.white38,
      fontSize: 55.0,
      fontWeight: FontWeight.bold,
      fontFamily: 'PT Sans',
      //,
      //PlayFair Display' ,
    ),  // TextStyle
  ),  // TypewriterAnimatedTextKit
),  // Center
],  // <Widget>[]
),  // Column
```



The webix appears in type write format with a lag of seconds.

Login page code snippet:

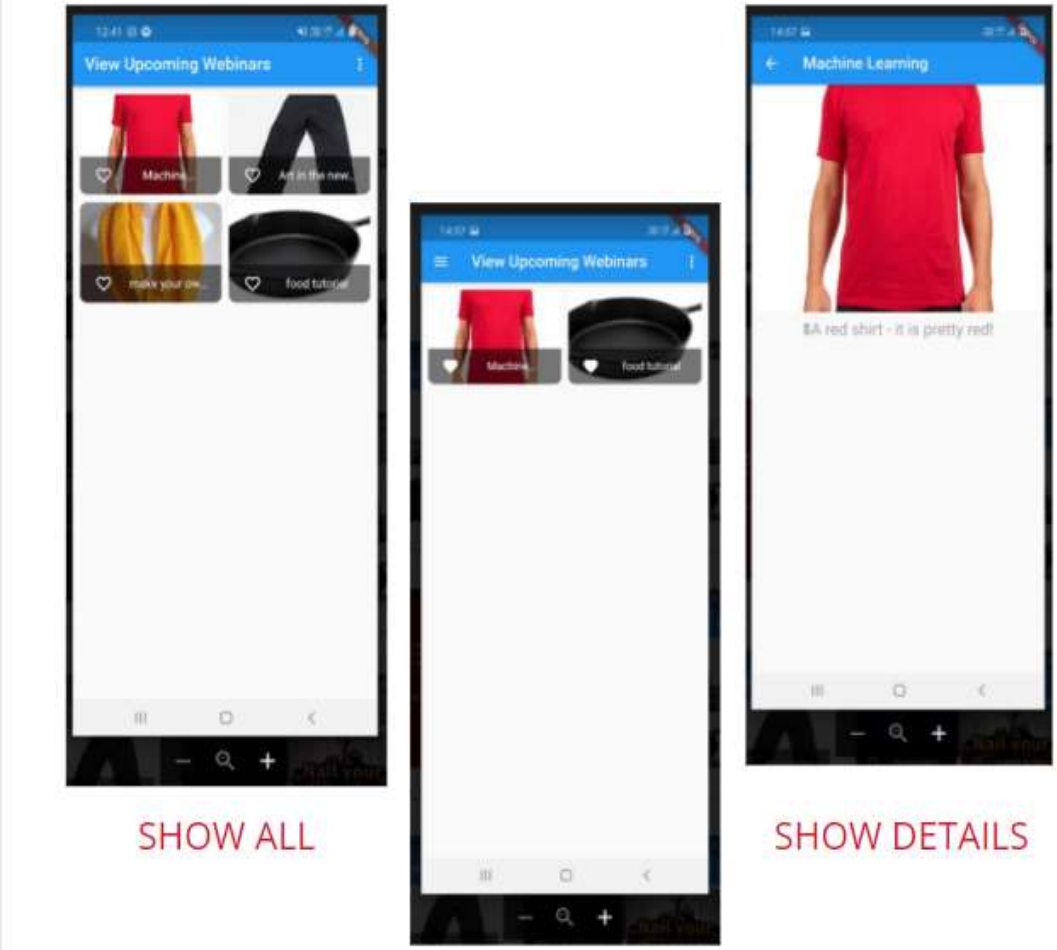
```
Column(
children: <Widget>[
    Container(
          height: 120.0,
          width: 120.0,
          decoration: BoxDecoration(

            image: DecorationImage(
              image: AssetImage(
                  'assets/images/logo.jpg'),  // AssetImage
              fit: BoxFit.fill,
            ),  // DecorationImage
            shape: BoxShape.circle,
          ),  // BoxDecoration
          //child: new Image.asset('images/webix_logo.png'),


            ),  // Container

    SizedBox(
      height: 48.0,
    ),  // SizedBox
    TextField(
      keyboardType: TextInputType.emailAddress,
      textAlign: TextAlign.center,
      onChanged: (value) {
        email = value;
      },
      decoration:
      kTextFieldDecoration.copyWith(hintText: 'Enter your email'),
    ),  // TextField
```

Registration page has the same built as the login page. Here's a code snippet of welcome page.

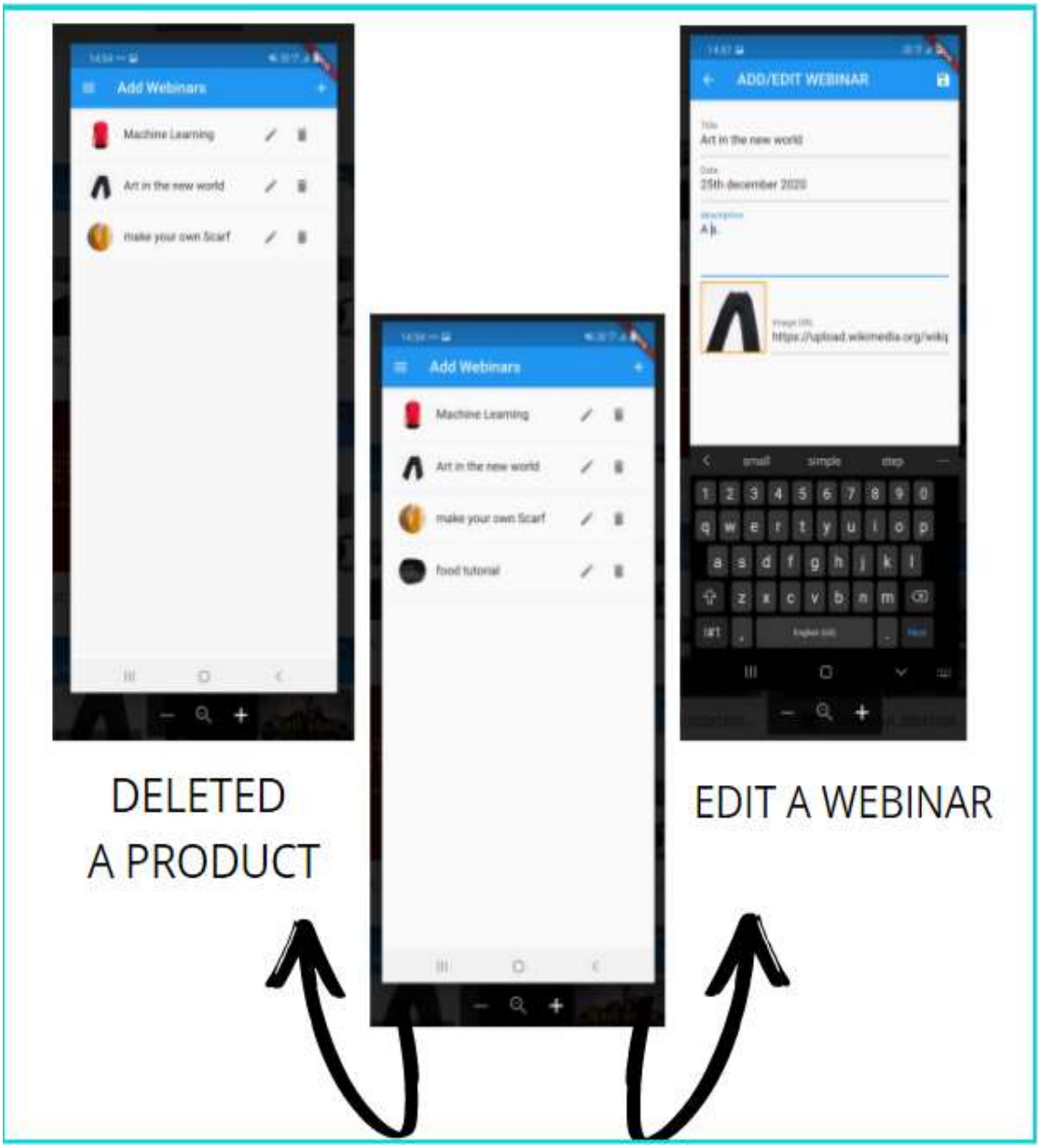
```
            ],  // <Widget>[
          ),  // Column

        SizedBox(
          height: 100.0,
        ),  // SizedBox
        RoundedButton(
          title: 'Log In',
          colour: Colors.blueGrey[800],
          onPressed: () {
            Navigator.pushNamed(context, LoginScreen.id);
          },
        ),  // RoundedButton
        RoundedButton(
          title: 'Register',
          colour: Colors.blueGrey[500],
          onPressed: () {
            Navigator.pushNamed(context, RegistrationScreen.id);
          },
        ),  // RoundedButton
      ],  // <Widget>[
    ),  // Column
  ),  // Padding

  ),  // Container
 );  // Scaffold
}
```

PHASE 1 TESTING

PATH 1 :



PATH 2:


Add Webinars
Machine Learning
Art in the new world
make your own Scarf
food tutorial
ADD/EDIT WEBINAR
Art in the new world
25th december 2020
DELETED
A PRODUCT
EDIT A WEBINAR

PATH 3:

ADD A WEBINAR

ADDED YAYA

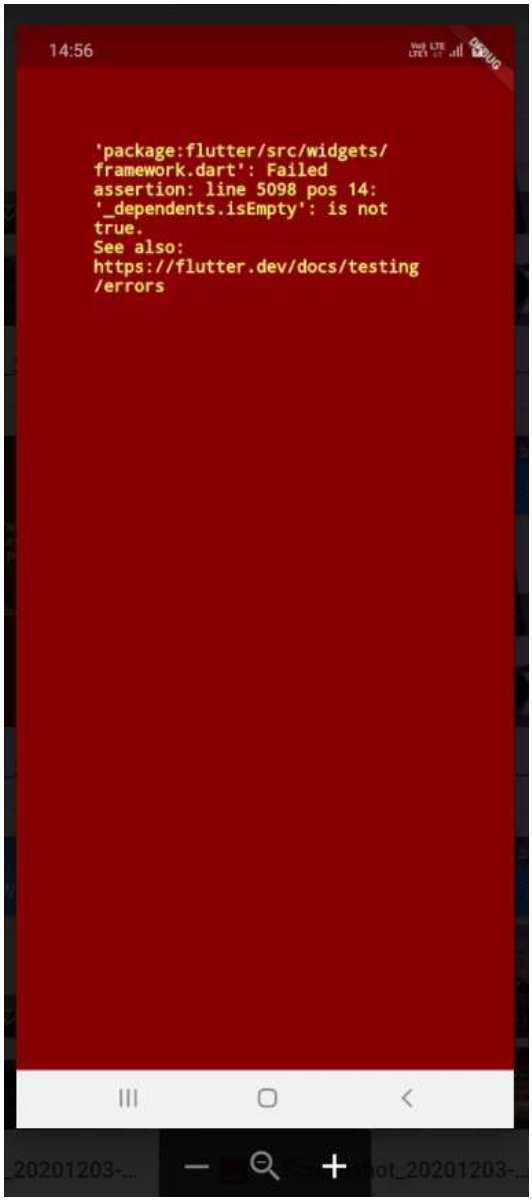
LIMITATIONS AND ERRORS:

During the phase 1 development it was found that many times page did not link up well with each other. Some or the other data failed to load up. The routes had to be managed carefully for this to not happen.

Secondly, the image did not load up for the add webinar preview.

The logo was not visible in the login,registration and welcome page.

The details displayed were not much upto standard. No helper text and error message was displayed during page not loading or webinar not added phase. To overcome these limitations the phase 2 of the project was taken up as the next challenge.



Phase 2 development and testing

Phase 2 fulfils the goal of customising the app to make the app much more user friendly and focuses on additional functionalities that should be present as a must in the software.

Webinar overview page: Customised design set up:

2. Webinar details page

Displays description
speaker name
and non functional
register button

Displays webinar details
speaker
name,description,duration,date,
time
and now has a functional
register button

The fields here are text tiles having leading, title and trailing properties.

Limitations:

The details are displayed in scrollable view. The registration button was the pain point in this module. It requires a url launcher package by flutter which enable it to access any given url from the app by opening up the browser. The registration button upon functionality directly opened up the browser page without showing up the details and led to error page when traced back. Later after a couple of trials it was finally successful when the url was provided as a variable string from the button trigger to the url launcher.

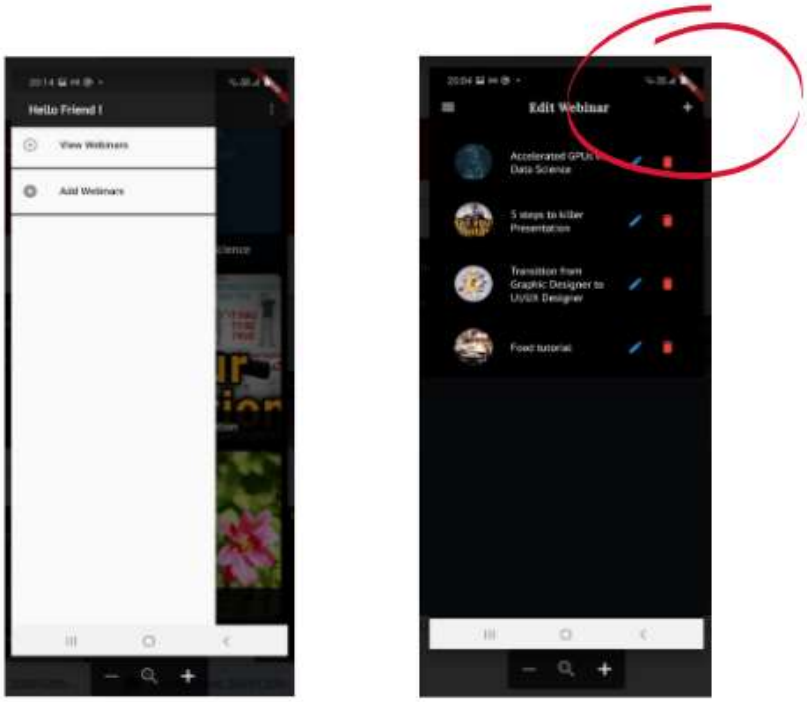
Register launch URL CODE SNIPPET

```
                title: Center(
                  child: Text('REGISTER', style: TextStyle(
                      color: Colors.white,
                      fontSize: 20,),),),  // TextStyle, Text
                ),  // Center

                onTap: () {
                  var url = loadedProduct.regurl;
                  _launchURL(url);
                }

              ),  // ListTile
            ),  // Padding
          ],  // <Widget>[]
        ),  // Column
      ),  // SingleChildScrollView
    );  // Scaffold
  }
  }
_launchURL(String url) async {
  if (await canLaunch(url)) {
    await launch(url, forceWebView: false);
  } else {
    throw 'Could not launch $url';
  }
}
```

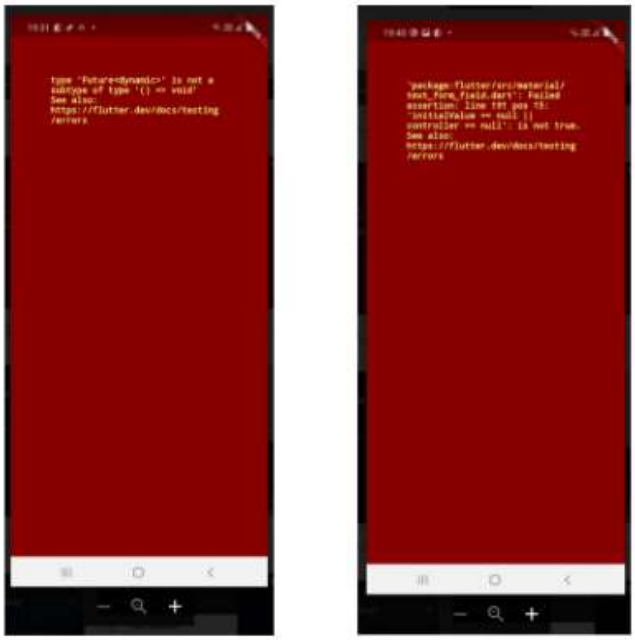
LIST TILE AND DESCRIPTION CODE SNIPPET

```
Container(
  padding: EdgeInsets.all(20),
  width: double.infinity,
  child: Text(
    loadedProduct.description,
    textAlign: TextAlign.justify,
    softWrap: true,
    style: TextStyle(
      color: Colors.white,
      fontSize: 20,
    ),  // TextStyle
  ),),  // Text, Container
Padding(
  padding: const EdgeInsets.all(10.0),
  child: ListTile(
    leading: Icon(Icons.account_circle, color: Colors.white,),
    title: Text('Speaker', style: TextStyle(
      color: Colors.white,
      fontSize: 20,),),  // TextStyle, Text
    trailing: Text(loadedProduct.speaker, style: TextStyle(
      color: Colors.white,
      fontSize: 20,),),  // TextStyle, Text
    onTap: () {},

  ),  // ListTile
),  // Padding
```

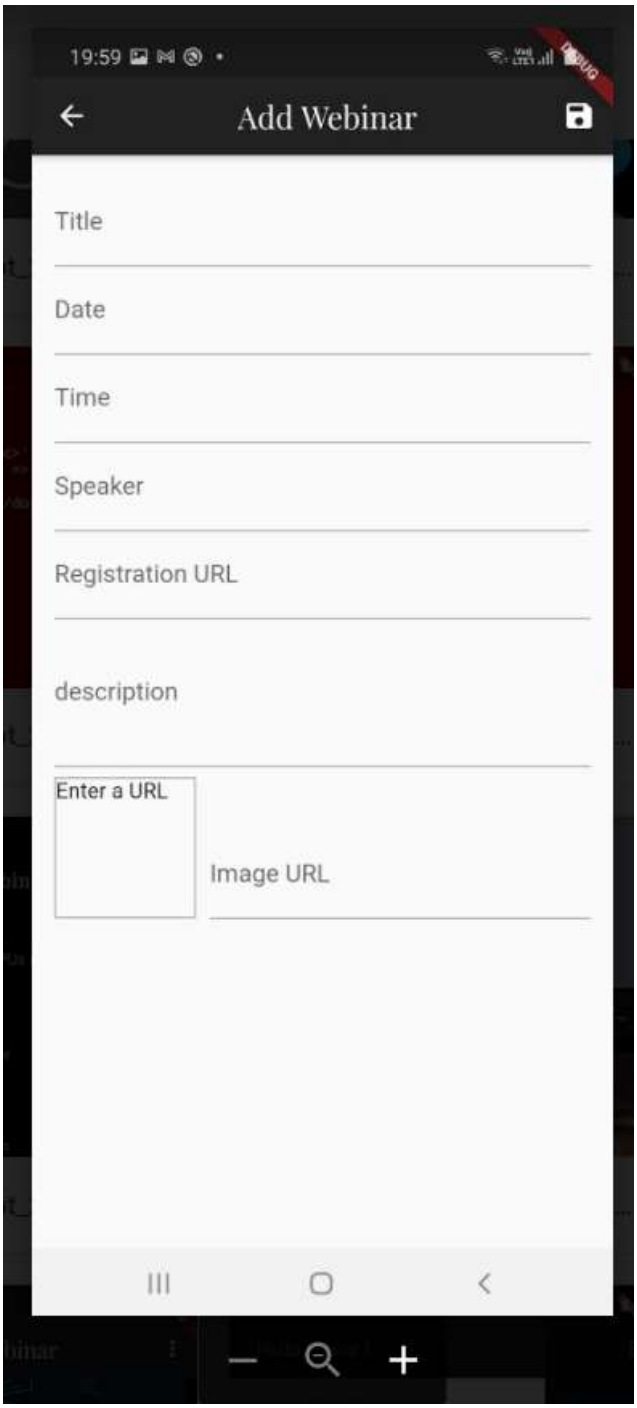
Customising the app drawer and the edit webinar page to enable uniform designing across all pages.



During phase 2 development the page of add webinars was not available on clicking the add button



After adding state in edit webinar screen data and adding the other form attributes the page looked as follows:



Added focus nodes to directly iterate to the next form element in the add webinars page on clicking next. Making the app super user friendly for its users

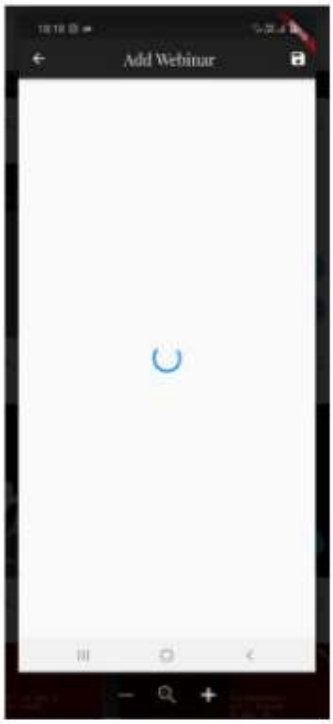
```
class EditWebinarScreen extends StatefulWidget {
  static const routeName = '/edit-webinars';
  @override
  _EditWebinarScreenState createState() => _EditWebinarScreenState();
}

class _EditWebinarScreenState extends State<EditWebinarScreen> {
  final _priceFocusNode = FocusNode();
  final _dateFocusNode = FocusNode();
  final _timeFocusNode = FocusNode();
  final _speakerFocusNode = FocusNode();
  final _regFocusNode = FocusNode();
  final _descriptionFocusNode = FocusNode();
  final _imageUrlController = TextEditingController();
  //final _regUrlController=TextEditingController();
  final _imageurlFocusnode = FocusNode();
  final _form = GlobalKey<FormState>();
  var _editedProduct = Webinar(
      webid: null,
```

This is to load up pre existing data in the edit form else a blank form will be displayed

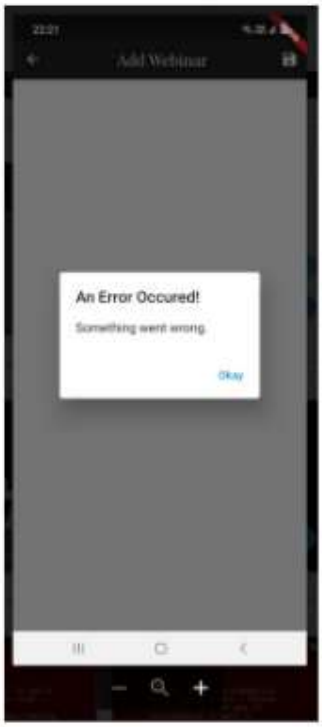
```
@override
void didChangeDependencies() {
  // TODO: implement didChangeDependencies
  if (_isInit) {
    final webid = ModalRoute.of(context).settings.arguments as String;
    if (webid != null) {
      _editedProduct =
          Provider.of<Webinars>(context, listen: false).findbyId(webid);
      _initvalues = {
        'webid': _editedProduct.webid,
        'title': _editedProduct.title,
        'speaker': _editedProduct.speaker,
        'description': _editedProduct.description,
        'duration': _editedProduct.duration,
        //'imageUrl': _editedProduct.imageUrl,
        'imageUrl': '',
        'date': _editedProduct.date,
        'time': _editedProduct.time,
        'regurl': _editedProduct.regurl
      };
      _imageUrlController.text = _editedProduct.imageUrl;
      //_regUrlController.text=_editedProduct.regurl;
    }
  }
  _isInit = false;
  super.didChangeDependencies();
}
```

To update the imageurl for preview in the blank box section as soon as the link is saved

```
void _updateimageurl() {
  if (!_imageurlFocusnode.hasFocus) {
    if ((!_imageUrlController.text.startsWith('http') &&
            !_imageUrlController.text.startsWith('https')) ||
        (!_imageUrlController.text.endsWith('.png') &&
            !_imageUrlController.text.endsWith('.jpg') &&
            !_imageUrlController.text.endsWith('.jpeg'))) {
      return;
    }
    setState(() {});
  }
}
```



loading preview while storing the added webinar to the database



displaying error message dialog when form is not submitted

Adding a loading spinner in the app to make sure the update to the server is done successfully.

Picture in camera -phone

Code snippet:

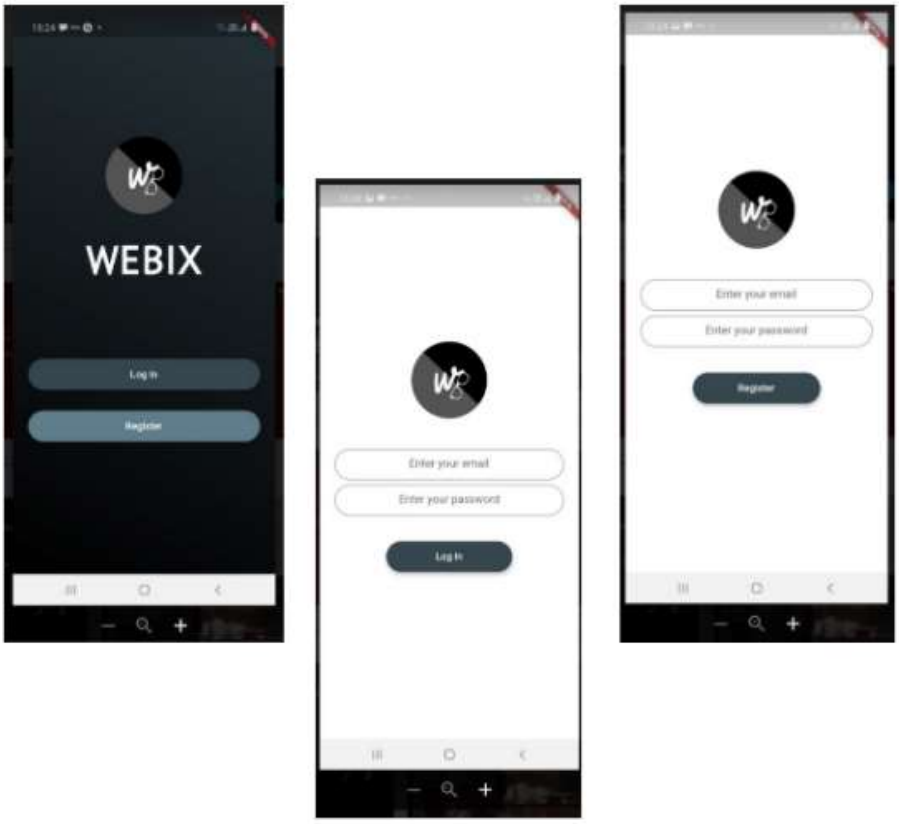
```
}
_form.currentState.save();
setState(() {
  _isloading=true;
});
if (_editedProduct.webid != null) {
  Provider.of<Webinars>(context, listen: false)
      .updateWebinar(_editedProduct.webid, _editedProduct);
  Navigator.of(context).pop();
} else {
  Provider.of<Webinars>(context, listen: false).addWebinar(_editedProduct).then((_){setState(() {
    _isloading=false;
  });
    Navigator.of(context).pop();
  });
}
```

To enable user to save form if all the fields with validated data is submitted

```
Future<void> _saveform() async {
  final isValid = _form.currentState.validate();
  if (!isValid) {
    return;
  }
  _form.currentState.save();
  setState(() {
    _isloading = true;
  });
  if (_editedProduct.webid != null) {
    Provider.of<Webinars>(context, listen: false)
        .updateWebinar(_editedProduct.webid, _editedProduct);
    setState(() {
      _isloading = false;
    });
    Navigator.of(context).pop();
  } else {
    try {
      await Provider.of<Webinars>(context, listen: false)
          .addWebinar(_editedProduct);
    } catch (error) {
      await showDialog(
        context: context,
        builder: (ctx) => AlertDialog(
          title: Text('An Error occured!'),
          content: Text('Something went wrong.'),
          actions: <Widget>[
            FlatButton(
              onPressed: () {
                Navigator.of(ctx).pop();
              },
              child: Text('Okay'),
            ),  // FlatButton
          ],  // <Widget>[]
```

LOGIN AND SIGN UP PHASE 2 :

Managed to add the logo to all three sections and activates the rounded buttons for navigation on check of authorised data.



Module constants for these customised rounded buttons.

```
const kTextFieldDecoration = InputDecoration(
  hintText: 'Enter a value',
  contentPadding: EdgeInsets.symmetric(vertical: 10.0, horizontal: 20.0),
  border: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),  // OutlineInputBorder
  enabledBorder:.OutlineInputBorder(
    borderSide: BorderSide(color: Colors.blueGrey, width: 1.0),
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),  // OutlineInputBorder
  focusedBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Colors.blueGrey, width: 2.0),
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),  // OutlineInputBorder
);  // InputDecoration
```

Register new user in register page else return false if user already exists

```
onPressed: () async {
            setState(() {
              showSpinner = true;
            });
            try {
              final newUser = await _auth.createUserWithEmailAndPassword(
                  email: email, password: password);
              if (newUser != null) {
                //Navigator.pushNamed(context, ChatScreen.id);
              }

              setState(() {
                showSpinner = false;
              });
            } catch (e) {
              print(e);
            }
          },
        ), // RoundedButton
```

Login authorization check

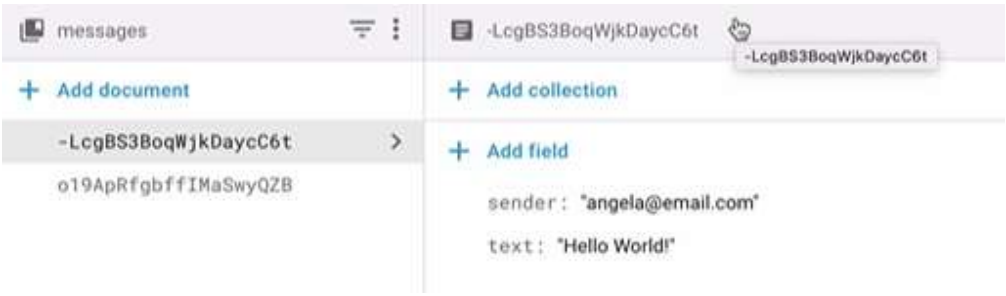
```
      async {
        setState(() {
          showSpinner = true;
        });
        try {
          final user = await _auth.signInWithEmailAndPassword(
              email: email, password: password);
          if (user != null) {
            //Navigator.pushNamed(context, ChatScreen.id);
          }

          setState(() {
            showSpinner = false;
          });
        } catch (e) {
          print(e);
        }
      },
```

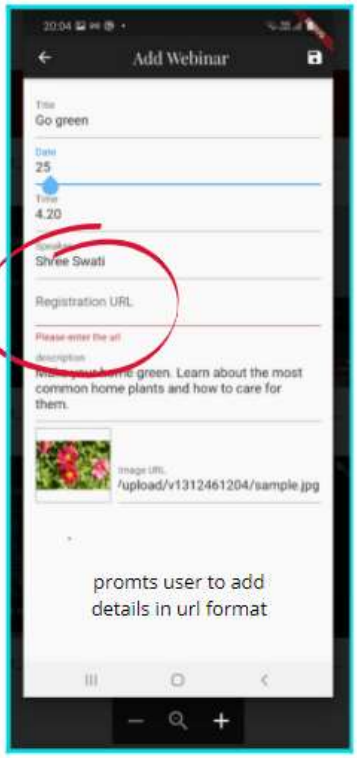
Displays spinner if unauthorized user tries to login.

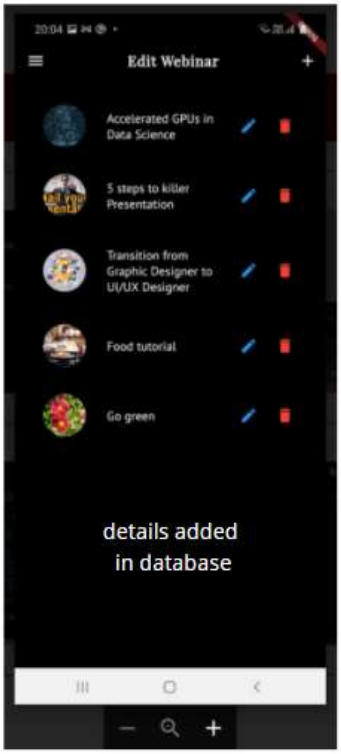
Test cases for phase 2 of the project:

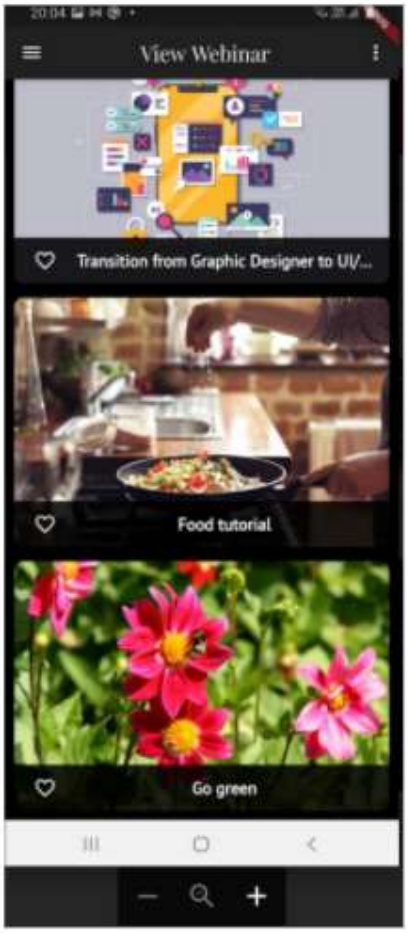
Show spinner while adding user details to the database

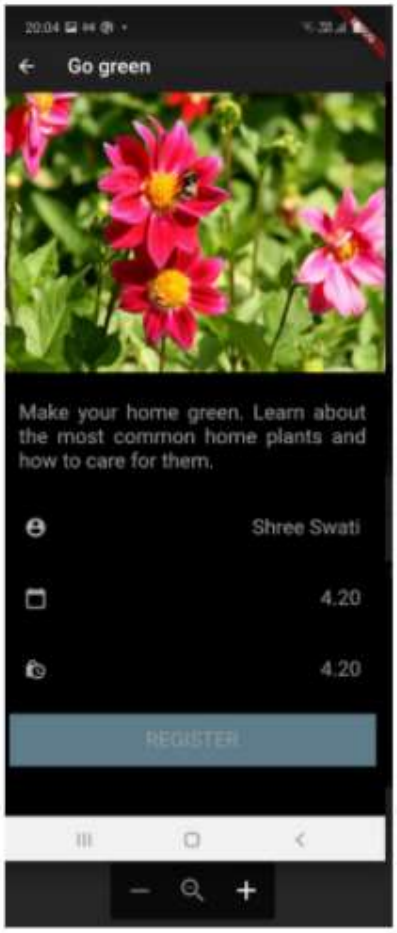
User details added to the firebase database. It will load up the register database if the user does not exist.







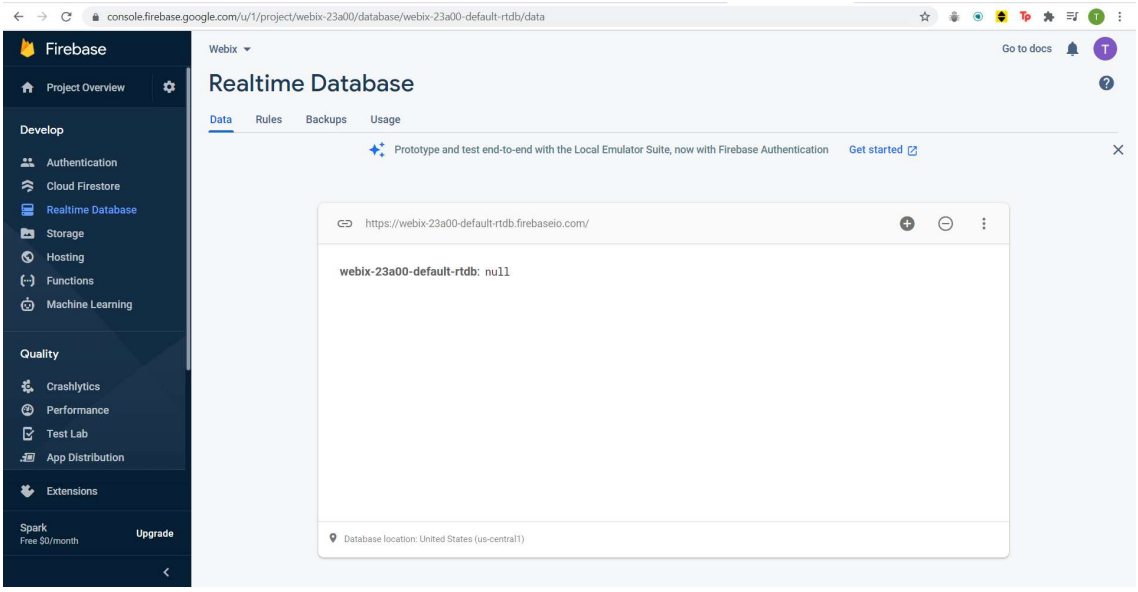




Hence the data of the new webinar has been added to the webinar database and on clicking the registration button we navigate to the webinar registration page that has been registered by the user.

Phase 3 -Adding all webinar details to the firebase server

Used real time database to enable real time data sharing -a feature by google via the firebase server.



INSTALLING THE REQUIRED PACKAGES FROM FLUTTER:

1. HTTP DEPENDENCY -https://pub.dev/packages/http/install
2. Worked with the REST API which connects the server to end point and an action to be performed.
3. Different HTTP requests sent and received to connect with web server are:
    - Get for getting data
    - Post for storing or adding data
    - Patch for data update
    - Put for data replace
    - Delete for deleting the data
4. Data is added to the firebase body in json format so the code data in dart needs to converted into the json format.JSON stands for javascript object notification.

```
void addWebinar(Webinar webinar) {
  const url = 'https://webix-23a00-default-rtdb.firebaseio.com/webinars.json';
  //firebase database url,attached package name is webinars,should be in json format
  http.post(url,
      body: json.encode({
        'title': webinar.title,
        'speaker': webinar.speaker,
        'description': webinar.description,
        'duration': webinar.duration,
        'imageUrl': webinar.imageUrl,
        'date': webinar.date,
        'time': webinar.time,
        'regurl': webinar.regurl
      }));
```

ADDED WEBINAR DATA IN FIREBASE

Getting response as webinar id from the firebase server to address webinars uniquely by primary key values:

```
    })).then((response){
      print(json.decode(response.body));
  final newWebinar = Webinar(

      webid:json.decode(response.body) ['name'],
      title: webinar.title,
      speaker: webinar.speaker,
      description: webinar.description,
      duration: webinar.duration,
      imageUrl: webinar.imageUrl,
      date: webinar.date,
      time: webinar.time,
      regurl: webinar.regurl);
  _items.add(newWebinar);
  //_items.insert(0, newWebinar) //FOR THE START OF THE STRING
  notifyListeners();
});
```

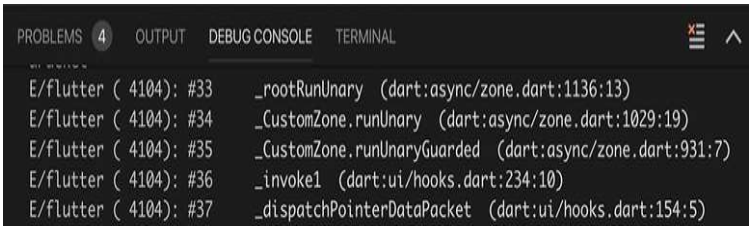
ERROR HANDLING

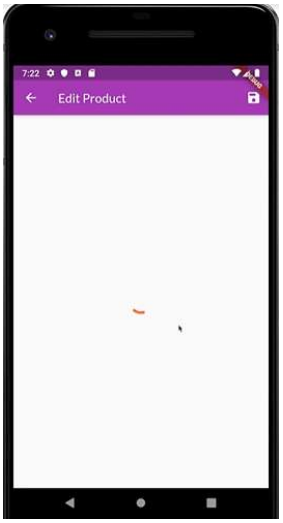
Displaying error messages to let the user know where they went wrong in case of app crash.

A stimulated case is used for testing where we remove the .json extension from the firebase app to observe what can happen.

1.The loading button does not proceed to load anything

2. The error crash message is displayed on the console.





Error is handled in the following ways:

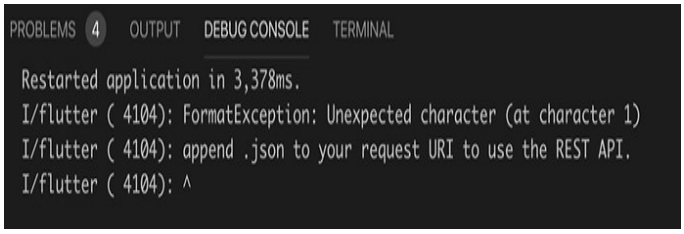
User has entered the webinar but it faces issue while loading into the web server.

The following code catches the error:

The part of the code which lets the user know about the error is shown here:

```
if (_editedProduct.webid != null) {
  Provider.of<Webinars>(context, listen: false)
      .updateWebinar(_editedProduct.webid, _editedProduct);
  Navigator.of(context).pop();
} else {
  Provider.of<Webinars>(context, listen: false)
      .addWebinar(_editedProduct)
      .catchError((error) {
    return showDialog(context: context, builder: (ctx) =>
        AlertDialog(
          title: Text('An Error Occured!'),
          content: Text('Something went wrong.'),
          actions: <Widget>[
            FlatButton(onPressed: () {
              Navigator.of(ctx).pop();
            },
              child: Text('Okay'),),  // FlatButton
          ],  // <Widget>[]
        ),  // AlertDialog
    );
  }).then((_){setState(() {
      _isloading=false;
    });
    Navigator.of(context).pop();
    });
  }
```

It allows the user to know that an error has occurred and takes them back to the main starting page. Instead of a crash error message the following is displayed if an error occurs and will return a future state of the edit webinar screen page. It can also be implemented via async -await and try-catch method. Error message at console:



TESTING-SECTION 10

TYPE

This is a program driven development (program is written first and then tested using various methods).

TEST CASES

FOR REGISTRATION-User is expected to fill In the E-mail ID and the password

When the user is clicks on register button on the home screen, User is expected to fill up all the field data.

button.

Test ID	Test Scenario	Expected Result	Actual result	Pass/Fail
T1001	Email id not entered	Back to register page	As expected	Pass
T1002	Email id not in expected format	Back to register page with a helper text display	As expected	Pass
T1003	Password not entered	Prompt for password	As expected	Pass

FOR Login -User is expected to fill In the E-mail ID and the password

Prerequisite: The user details must be registered before logging in.

When click on login, the login page will be displayed.

Test ID	Test Scenario	Expected Result	Actual result	Pass/Fail

T1001	Email id not entered	Back to login page	As expected	Pass
T1002	Email id not in expected format	Back to login page with a helper text display for correct format	As expected	Pass
T1003	Password not entered	Prompt for password	As expected	Pass

For ADD/EDIT WEBINAR data

Test ID	Test Scenario	Expected Result	Actual result	Pass/Fail
T1001	Title not entered	Error prompt message back to add webinar page	As expected	Pass
T1002	Description not entered	Error prompt message back to add webinar page	As expected	Pass
T1003	Date not entered	Error prompt message back to add webinar page	As expected	Pass
T1004	Time not entered	Error prompt message back to add webinar page	As expected	Pass

T1005	Speaker name not entered	Error prompt message back to add webinar page	As expected	Pass
T1006	Registration URL not entered	Error prompt message back to add webinar page	As expected	Pass
T1007	Image Url not entered	Error prompt message back to	As expected	Pass
T1008	Image url not in http /https/jpg/png format	Display error , enter valid image url message displayed	As expected	Pass

For Delete Data

Test ID	Test Scenario	Expected Result	Actual result	Pass/Fail
T1001	Delete webinar	POP-UP expected To confirm	As expected	Pass
T1009	In pop-up okay is clicked	Data is deleted	As expected	Pass
T1010	If pop-up is ignored	Data is not deleted and user gets back to edit page	As expected	Pass

Appendix

Here Webinar refers to any scheduled online live activity which can to attended by the people through their devices in online mode as per choice of interest.

THANKYOU