

## Practical 1

### Installation and Configuration of Flutter Environment

Name: Sharvari Kishor More

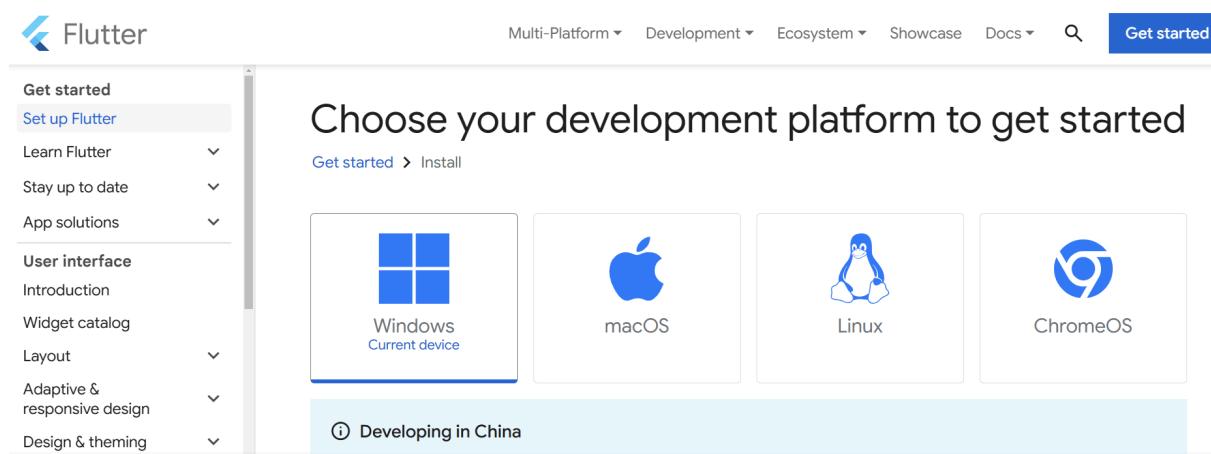
Class: D15B

Roll No.: 35

**Aim: Installation and Configuration of Flutter Environment**

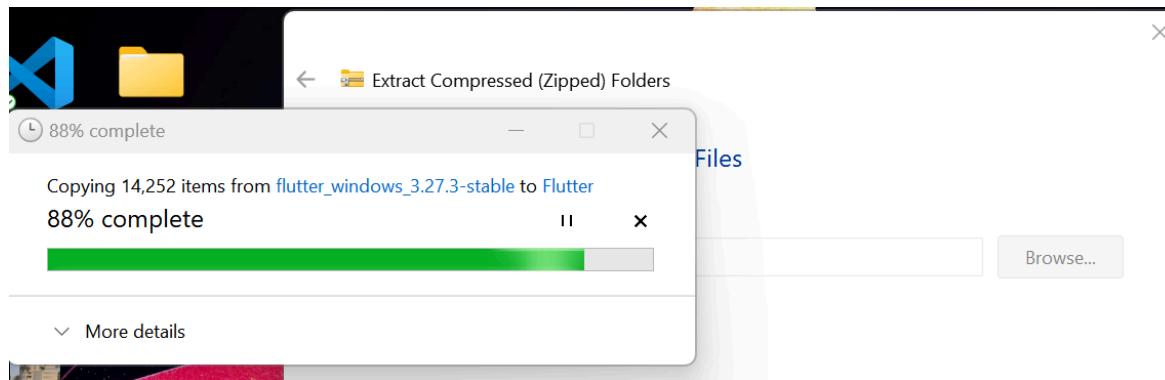
#### Install the Flutter SDK

**Step 1:** Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website <https://docs.flutter.dev/get-started/install> , you will get the following screen.



**Step 2:** Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

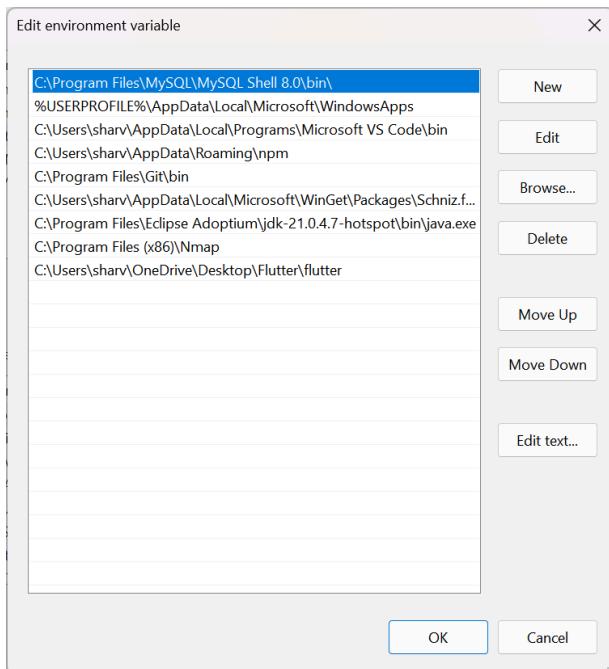
**Step 3:** When your download is complete, extract the zip file and place it in the desired installation folder or location, such as C: /Flutter.



**Step 4:** To run the Flutter command in regular windows console, you need to update the system

path to include the flutter bin directory. The following steps are required to do this:

**Step 4.1:** Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.



**Step 5:** Now, run the \$ flutter command in command prompt.

Now, run the \$ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

```
Administrator: Command Prompt - flutter
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.

C:\Users\INFTS05-02>flutter
Manage your Flutter app development.

Common commands:
  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [<arguments>]

Global options:
  -h, --help          Print this usage information.
  -v, --verbose       Noisy logging, including all shell commands executed.
  -d, --device-id    If used with "-help", shows hidden options. If used with "flutter doctor", shows additional
                     diagnostic information. (Use "-vv" to force verbose logging in those cases.)
  --version          Target device id or name (prefixes allowed).
  --enable-analytics Reports the version of this tool.
  --disable-analytics Enable telemetry reporting each time a flutter or dart command runs.
  --suppress-analytics Disable telemetry reporting each time a flutter or dart command runs, until it is
                        re-enabled.

Available commands:
Flutter SDK
  bash-completion  Output command line shell completion setup scripts.
  channel         List or switch Flutter channels.
  config          Configure Flutter settings.
  doctor          Show information about the installed tooling.
  downgrade      Downgrade Flutter to the last active version for the current channel.
  precache        Populate the Flutter tool's cache of binary artifacts.
  upgrade         Upgrade your copy of Flutter.
```

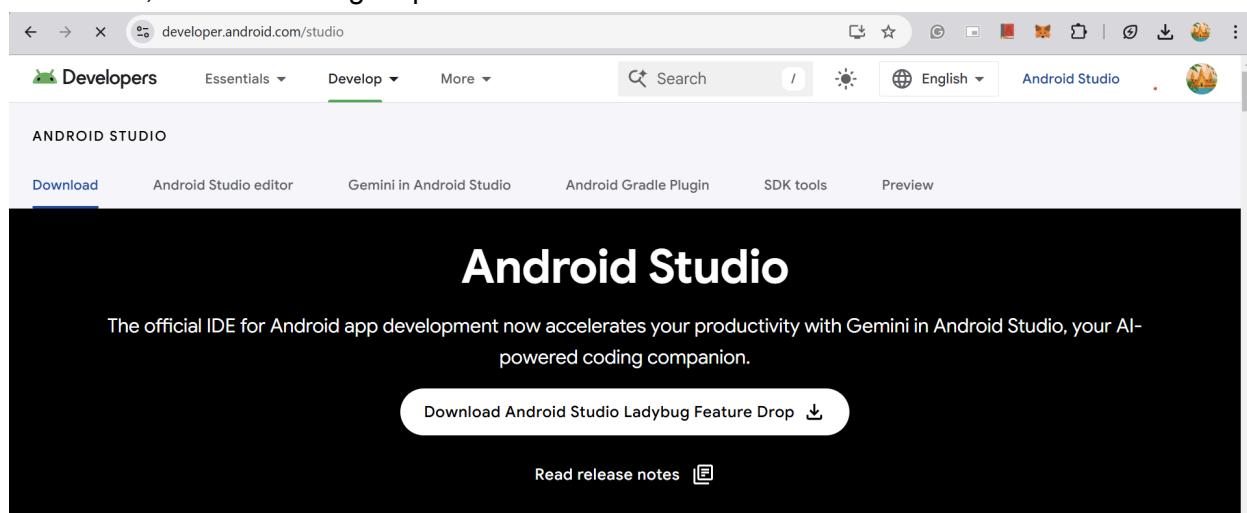
**Step 6:** When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to

run Flutter as well as the development tools that are available but not connected with the device.

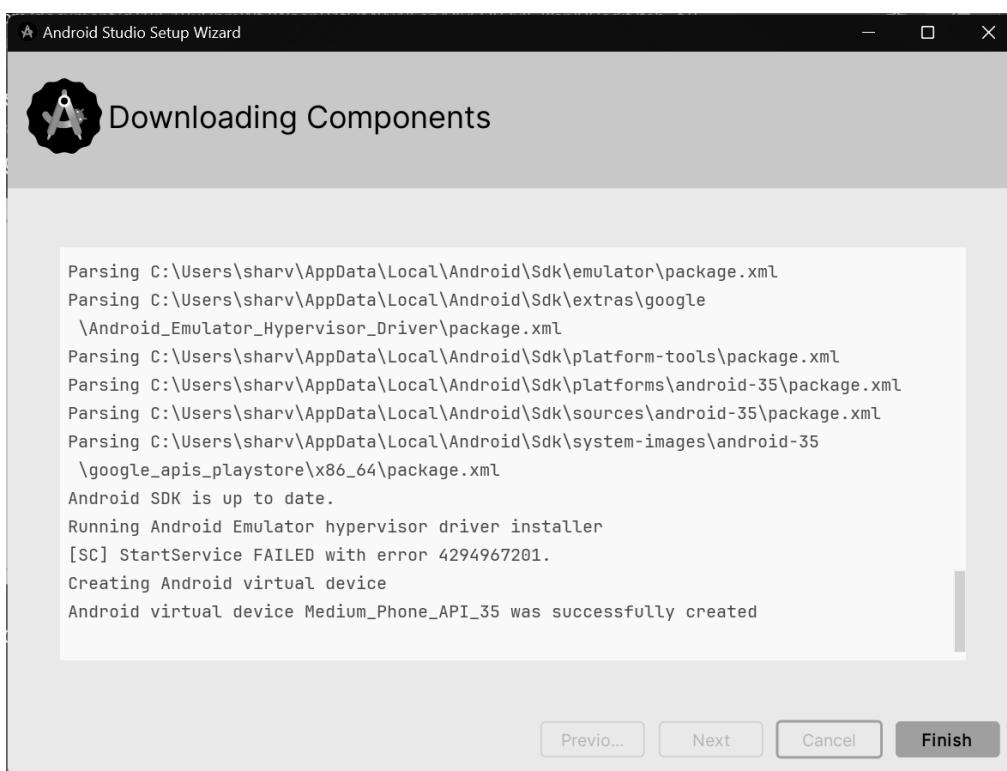
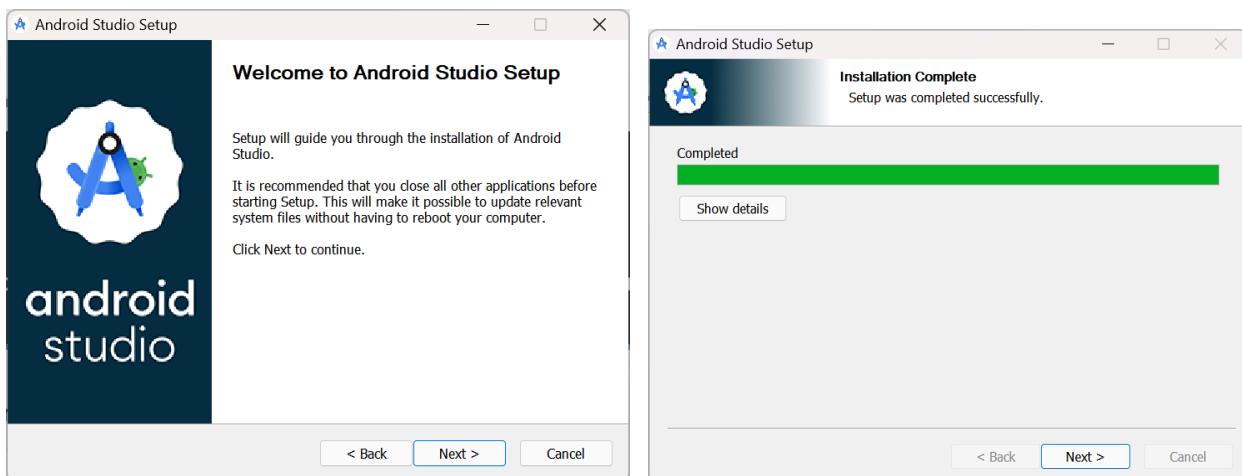
```
C:\Users\INFT505-02>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.19045.5371], locale en-US)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.1)
[✓] Chrome - develop for the web
[✗] Visual Studio - develop Windows apps
  X Visual Studio not installed; this is necessary to develop Windows apps.
    Download at https://visualstudio.microsoft.com/downloads/.
    Please install the "Desktop development with C++" workload, including all of its default components
[✓] Android Studio (version 2021.3)
[✓] VS Code (version 1.72.2)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 1 category.
```

**Step 7:** Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.



**Step 7.2:** When the download is complete, open the .exe file and run it. You will get the following dialog box.

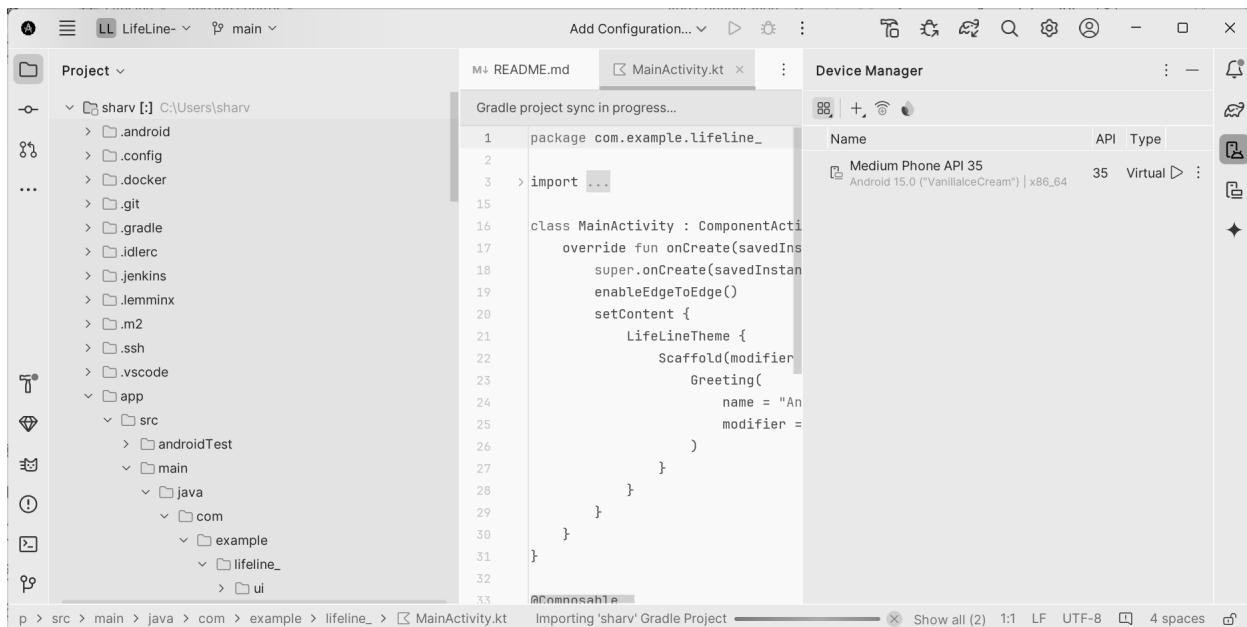


**Step 7.5:** run the \$ flutter doctor command and Run flutter doctor --android-licenses command.

```
C:\Users\INFT505-02>flutter doctor --android-licenses
[=====] 61% Fetch remote repository...      etch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[=====] 73% Fetch remote repository...      etch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[=====] 100% Computing updates...
All SDK package licenses accepted.
```

**Step 8:** Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

**Step 8.1:** To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.

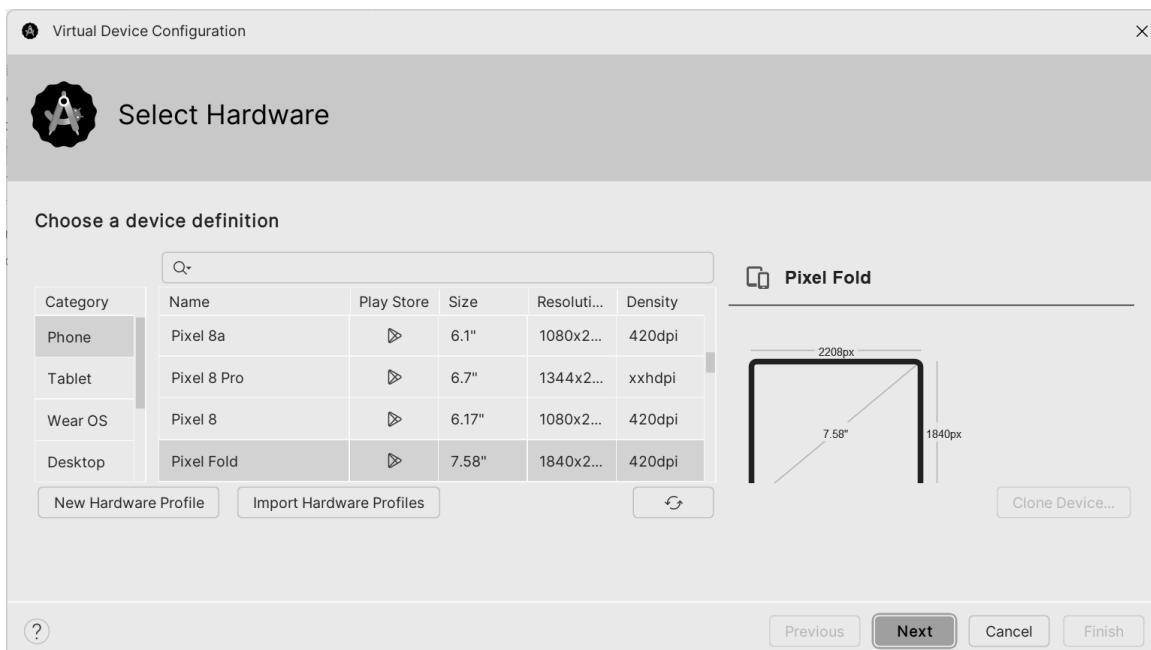


**Step 8.2:** Choose your device definition and click on Next.

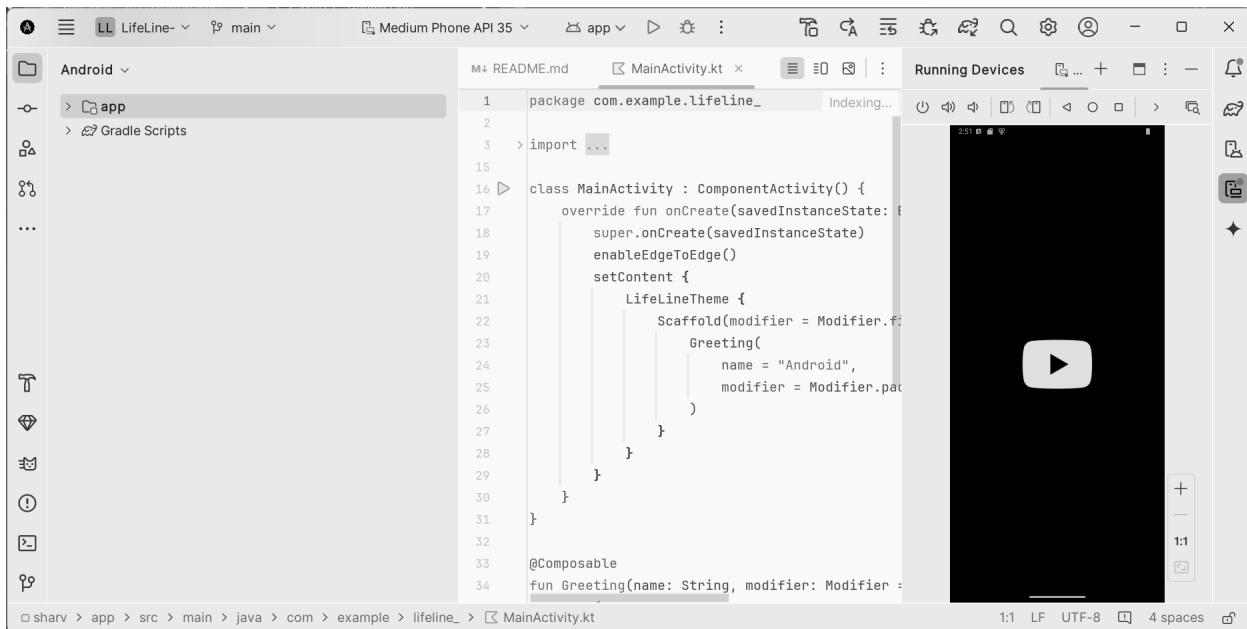


**Step 8.3:** Select the system image for the latest Android version and click on Next.

**Step 8.4:** Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.

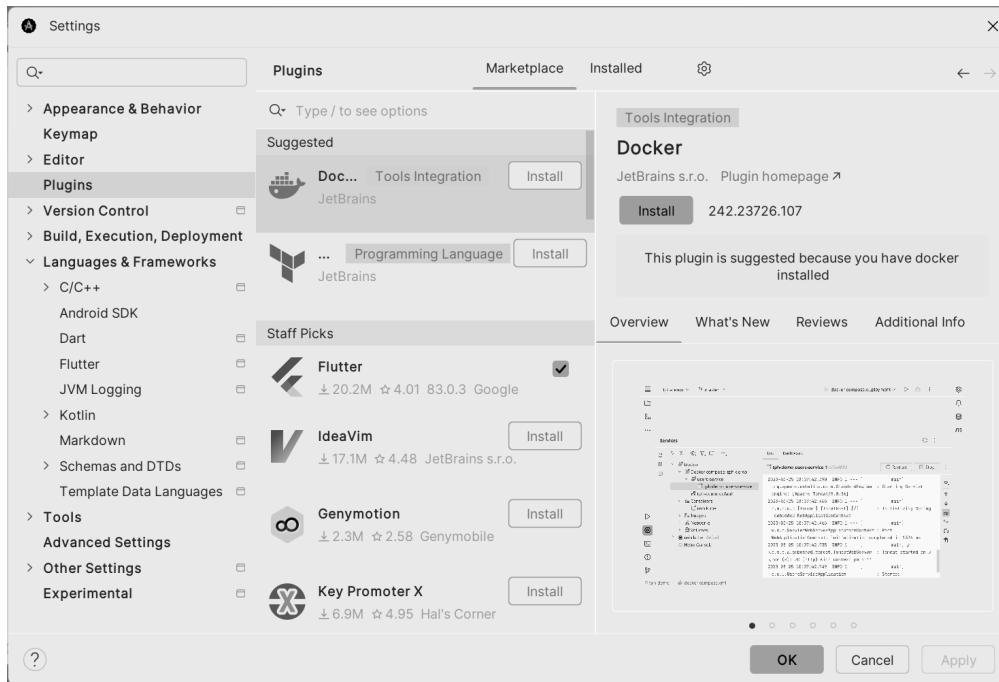


**Step 8.5:** Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.



**Step 9:** Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

**Step 9.1:** Open the Android Studio and then go to File->Settings->Plugins.



**Step 9.2:** Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.

**Step 9.3:** Restart the Android Studio.

## **Practical 2**

### **Design a Flutter UI by including Common Widgets**

Name: Sharvari Kishor More

Class: D15B

Roll No.: 35

Aim: **To design a Flutter UI by including common widgets.**

Theory:

#### **Introduction**

Flutter is a popular framework for building cross-platform mobile applications. It uses a widget-based approach, where everything is a widget, making UI design flexible and efficient. Common widgets like AppBar, Container, TextField, GridView, and ListView help in creating responsive and visually appealing interfaces.

A Flutter UI is structured using widgets, which can be broadly classified into:

- StatelessWidget: A widget that does not change over time.
- StatefulWidget: A widget that can update dynamically based on user interaction.

Flutter applications typically use a Scaffold widget, which provides a structure that includes an AppBar, body, floating action buttons, bottom navigation bars, and other UI elements.

#### **Implementation**

In this experiment, we designed a Flutter UI for the Lifeline App, which helps users find hospitals based on their symptoms. The UI includes:

- App Bar: Displays the user's selected location and profile icon.
- Search Bar: Allows users to search for clinics.
- Appointment Section: Provides options for booking an in-clinic appointment or instant video consultation.
- Doctor Specialization Grid: Displays different medical specializations using GridView.
- Featured Services Section: Highlights important healthcare services.

We used StatelessWidget to create reusable components like AppointmentCard and DoctorCategoryCard. The UI follows a structured layout using Column, Row, and Padding for spacing and alignment.

**CODE:**

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Lifeline',
      theme: ThemeData(
        primarySwatch: Colors.indigo,
      ),
      home: HomeScreen(),
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.indigo,
        title: Row(
          children: [
            Icon(Icons.location_on, color: Colors.white),
            SizedBox(width: 5),
            Text("Bangalore", style: TextStyle(color: Colors.white)),
            Icon(Icons.keyboard_arrow_down, color: Colors.white),
          ],
        ),
        actions: [
          IconButton(
            icon: Icon(Icons.account_circle, color: Colors.white),
            onPressed: () {},
          ),
        ],
      ),
    );
  }
}
```

```
body: SingleChildScrollView(  
  child: Padding(  
    padding: EdgeInsets.all(16.0),  
    child: Column(  
      crossAxisAlignment: CrossAxisAlignment.start,  
      children: [  
        // Search Bar  
        Container(  
          padding: EdgeInsets.symmetric(horizontal: 10),  
          decoration: BoxDecoration(  
            color: Colors.grey[200],  
            borderRadius: BorderRadius.circular(10),  
          ),  
          child: TextField(  
            decoration: InputDecoration(  
              hintText: "Search for clinics",  
              border: InputBorder.none,  
              prefixIcon: Icon(Icons.search, color: Colors.grey),  
            ),  
          ),  
        ),  
        SizedBox(height: 20),  
  
        // Appointment & Consultation  
        Row(  
          mainAxisAlignment: MainAxisAlignment.spaceBetween,  
          children: [  
            Expanded(  
              child: AppointmentCard(  
                title: "Book In-Clinic Appointment",  
                imagePath: "assets/images/doctor_1.jpg",  
              ),  
            ),  
            SizedBox(width: 10),  
            Expanded(  
              child: AppointmentCard(  
                title: "Instant Video Consultation",  
                imagePath: "assets/images/doctor_2.jpg",  
              ),  
            ),  
          ],  
        ),  
        SizedBox(height: 20),
```

```
// Find a Doctor Section
Text(
  "Find a Doctor for your Health Problem",
  style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
),
SizedBox(height: 10),  
  
// Categories
GridView.builder(
  shrinkWrap: true,
  physics: NeverScrollableScrollPhysics(),
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 4,
    crossAxisSpacing: 10,
    mainAxisSpacing: 10,
    childAspectRatio: 0.9,
),
itemCount: doctorCategories.length,
itemBuilder: (context, index) {
  return DoctorCategoryCard(doctorCategories[index]);
},
),
SizedBox(height: 20),  
  
// Featured Services
Text(
  "Featured services",
  style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
),
SizedBox(height: 10),  
  
Container(
  width: double.infinity,
  height: 120,
  decoration: BoxDecoration(
    color: Colors.indigo,
    borderRadius: BorderRadius.circular(10),
),
child: Center(
  child: Text(
    "Affordable Procedures by Expert Doctors",
    style: TextStyle(color: Colors.white, fontSize: 16),
    textAlign: TextAlign.center,
```

```
        ),  
        ),  
        ),  
        ],  
        ),  
        ),  
        );  
    }  
}  
  
class AppointmentCard extends StatelessWidget {  
    final String title;  
    final String imagePath;  
  
    AppointmentCard({required this.title, required this.imagePath});  
  
    @override  
    Widget build(BuildContext context) {  
        return Container(  
            height: 100,  
            decoration: BoxDecoration(  
                color: Colors.white,  
                borderRadius: BorderRadius.circular(10),  
                boxShadow: [  
                    BoxShadow(  
                        color: Colors.grey.shade300,  
                        blurRadius: 5,  
                        spreadRadius: 2,  
                    ),  
                ],  
            ),  
            child: Column(  
                children: [  
                    Expanded(  
                        child: Image.asset(imagePath, fit: BoxFit.cover),  
                    ),  
                    Padding(  
                        padding: const EdgeInsets.all(8.0),  
                        child: Text(  
                            title,  
                            textAlign: TextAlign.center,  
                            style: TextStyle(fontSize: 14, fontWeight: FontWeight.bold),  
                        ),  
                    ),  
                ],  
            ),  
        );  
    }  
}
```

```

        ),
    ],
),
);
}
}

class DoctorCategoryCard extends StatelessWidget {
final DoctorCategory category;

DoctorCategoryCard(this.category);

@Override
Widget build(BuildContext context) {
return Column(
children: [
Container(
padding: EdgeInsets.all(10),
decoration: BoxDecoration(
color: Colors.blue[50],
borderRadius: BorderRadius.circular(10),
),
child: Icon(category.icon, color: Colors.indigo, size: 30),
),
SizedBox(height: 5),
Text(category.title, textAlign: TextAlign.center, style: TextStyle(fontSize: 12)),
],
);
}
}

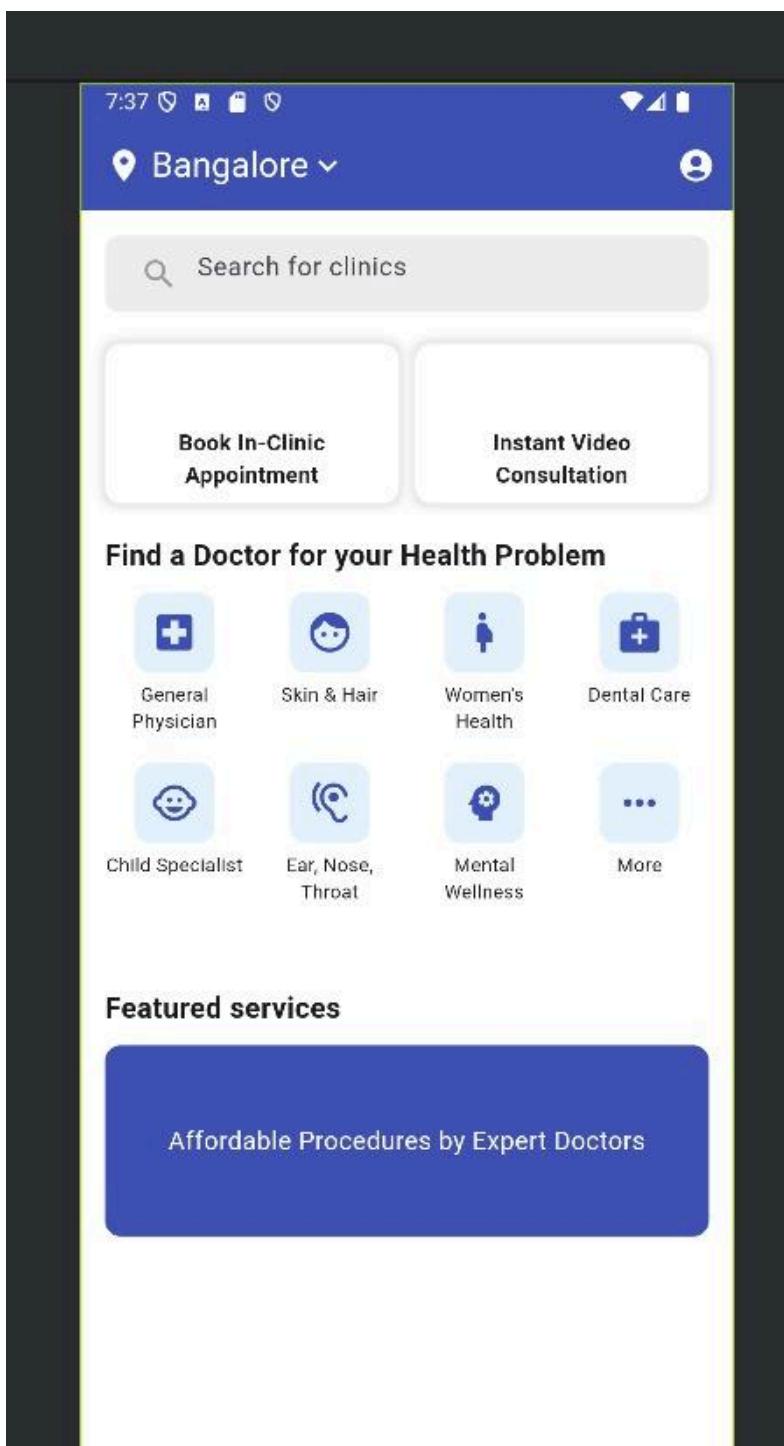
class DoctorCategory {
final String title;
final IconData icon;

DoctorCategory({required this.title, required this.icon});
}

List<DoctorCategory> doctorCategories = [
DoctorCategory(title: "General Physician", icon: Icons.local_hospital), // Alternative for stethoscope
DoctorCategory(title: "Skin & Hair", icon: Icons.face),
DoctorCategory(title: "Women's Health", icon: Icons.pregnant_woman),
DoctorCategory(title: "Dental Care", icon: Icons.medical_services),
]

```

```
DoctorCategory(title: "Child Specialist", icon: Icons.child_care),  
DoctorCategory(title: "Ear, Nose, Throat", icon: Icons.hearing),  
DoctorCategory(title: "Mental Wellness", icon: Icons.psychology),  
DoctorCategory(title: "More", icon: Icons.more_horiz),  
];
```

**Screenshot:**

**Conclusion:**

In this experiment, we implemented a Flutter UI for the Lifeline App using common widgets like AppBar, GridView, and Column. We initially faced an issue with the Icons.stethoscope not being available, which we resolved by replacing it with alternative icons like Icons.local\_hospital and Icons.medical\_services.

## Experiment 3

### Flutter Widgets

**Name:** Sharvari Kishor More

**Class:** D15B

**Roll no.:** 35

**Aim:** To include icons, images, and fonts in the Flutter app.

#### **Introduction**

Mobile Application Development (MAD) involves designing and implementing applications that run on mobile devices. Flutter, an open-source UI software development kit by Google, is widely used for building natively compiled applications for mobile, web, and desktop from a single codebase.

#### **Objective**

The goal of this experiment is to design and develop a healthcare application interface using Flutter. This involves adding essential UI components such as buttons, icons, a search bar, and a dropdown list to enhance user interaction.

#### **Concepts Covered**

1. **Scaffold & AppBar:** Used to create the structure of the mobile application, including the navigation bar and main screen layout.
2. **Dropdown Button:** Allows users to select their city from a predefined list.
3. **Search Bar:** Enables users to search for clinics within the application.
4. **Buttons & Icons:** Enhance UI interactions, such as booking appointments and navigating the app.
5. **Grid View:** Displays doctor categories using an interactive grid layout.
6. **Custom Widgets:** Implemented reusable widgets for appointment booking and doctor categories.

#### **Implementation**

In this experiment, we implemented a Flutter UI with the following key features:

- **Dropdown Menu in AppBar:** Allows users to select a city.
- **Search Bar Widget:** Users can search for clinics conveniently.
- **Appointment Booking Cards:** Users can book in-clinic and video consultations.
- **Doctor Categories in Grid View:** Displays available medical specialties using interactive buttons.
- **Profile Icon & Featured Services Section:** Enhances UI usability and provides quick access to important features.

**Code:****home\_screen.dart**

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

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  String selectedCity = "Bangalore";
  final List<String> cities = ["Bangalore", "Pune", "Nashik",
  "Mumbai"];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.indigo,
        title: Row(
          children: [
            Icon(Icons.location_on, color: Colors.white),
            SizedBox(width: 5),
            DropdownButton<String>(
              value: selectedCity,
              dropdownColor: Colors.indigo,
              icon: Icon(Icons.keyboard_arrow_down, color:
              Colors.white),
              underline: SizedBox(),
              style: TextStyle(color: Colors.white, fontSize: 16),
              onChanged: (String? newValue) {
                setState(() {
                  selectedCity = newValue!;
                });
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

```
        items: cities.map<DropdownMenuItem<String>>((String
city) {
    return DropdownMenuItem<String>(
        value: city,
        child: Text(city, style: TextStyle(color:
Colors.white)),
    );
}),
),
],
),
actions: [
    IconButton(
        icon: Icon(Icons.account_circle, color: Colors.white),
        onPressed: () {},
    ),
],
),
body: SingleChildScrollView(
    child: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Container(
                    padding: EdgeInsets.symmetric(horizontal: 10),
                    decoration: BoxDecoration(
                        color: Colors.grey[200],
                        borderRadius: BorderRadius.circular(10),
                    ),
                    child: TextField(
                        decoration: InputDecoration(
                            hintText: "Search for clinics",
                            border: InputBorder.none,
                            prefixIcon: Icon(Icons.search, color:
Colors.grey),
                        ),
                    ),
                ),
            ],
        ),
    ),
),
```

```
),
SizedBox(height: 20),
// Appointment & Consultation Section
Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    Expanded(
      child: AppointmentCard(
        title: "Book In-Clinic Appointment",
        imagePath: "assets/images/images.jpg",
      ),
    ),
    SizedBox(width: 10),
    Expanded(
      child: AppointmentCard(
        title: "Instant Video Consultation",
        imagePath: "assets/images/hospital.jpg",
      ),
    ),
  ],
),
SizedBox(height: 20),
Text(
  "Find a Doctor for your Health Problem",
  style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
),
SizedBox(height: 10),
GridView.builder(
  shrinkWrap: true,
  physics: NeverScrollableScrollPhysics(),
  gridDelegate:
SliverGridDelegateWithFixedCrossAxisCount(
  crossAxisCount: 4,
  crossAxisSpacing: 10,
  mainAxisSpacing: 10,
  childAspectRatio: 0.9,
),
```

```
        itemCount: doctorCategories.length,
        itemBuilder: (context, index) {
            return
DoctorCategoryButton(doctorCategories[index]);
        },
),
SizedBox(height: 20),
Text(
    "Featured services",
    style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
),
SizedBox(height: 10),
Container(
    width: double.infinity,
    height: 120,
    decoration: BoxDecoration(
        color: Colors.indigo,
        borderRadius: BorderRadius.circular(10),
),
    child: Center(
        child: Text(
            "Affordable Procedures by Expert Doctors",
            style: TextStyle(color: Colors.white, fontSize:
16),
            textAlign: TextAlign.center,
),
),
),
),
),
),
),
),
);
}
}

class AppointmentCard extends StatelessWidget {
```

```
final String title;
final String imagePath;

const AppointmentCard({
    required this.title,
    required this.imagePath,
    super.key,
}) ;

@Override
Widget build(BuildContext context) {
    return Container(
        height: 120,
        decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(10),
            boxShadow: [
                BoxShadow(color: Colors.grey.shade300, blurRadius: 5,
                    spreadRadius: 2),
            ],
        ),
        child: Column(
            children: [
                Expanded(
                    child: ClipRRect(
                        borderRadius: BorderRadius.vertical(top:
                            Radius.circular(10)),
                    child: Image.asset(imagePath, width: double.infinity,
                        fit: BoxFit.cover),
                ),
            ),
            Padding(
                padding: EdgeInsets.all(8.0),
                child: Text(title, textAlign: TextAlign.center, style:
                    TextStyle(fontSize: 14, fontWeight: FontWeight.bold)),
            ),
        ],
    ),
}
```

```
        );
    }
}

class DoctorCategoryButton extends StatelessWidget {
    final DoctorCategory category;
    const DoctorCategoryButton(this.category, {super.key});

    @override
    Widget build(BuildContext context) {
        return GestureDetector(
            onTap: () {
                ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(content: Text("Selected: ${category.title}")),
                );
            },
            child: Column(
                children: [
                    Container(
                        padding: EdgeInsets.all(10),
                        decoration: BoxDecoration(
                            color: Colors.blue[50],
                            borderRadius: BorderRadius.circular(10),
                        ),
                        child: Icon(category.icon, color: Colors.indigo, size:
                            30),
                    ),
                    SizedBox(height: 5),
                    Text(category.title, textAlign: TextAlign.center, style:
                        TextStyle(fontSize: 12)),
                ],
            ),
        );
    }
}

class DoctorCategory {
    final String title;
```

```

final IconData icon;
DoctorCategory({required this.title, required this.icon});
}

List<DoctorCategory> doctorCategories = [
  DoctorCategory(title: "General Physician", icon:
Icons.local_hospital),
  DoctorCategory(title: "Skin & Hair", icon: Icons.face),
  DoctorCategory(title: "Women's Health", icon:
Icons.pregnant_woman),
  DoctorCategory(title: "Dental Care", icon: Icons.medical_services),
  DoctorCategory(title: "Child Specialist", icon: Icons.child_care),
  DoctorCategory(title: "Ear, Nose, Throat", icon: Icons.hearing),
  DoctorCategory(title: "Mental Wellness", icon: Icons.psychology),
  DoctorCategory(title: "More", icon: Icons.more_horiz),
];

```

**Code:****main.dart**

```

import 'package:flutter/material.dart';
import 'home_screen.dart';
import 'login_screen.dart';
import 'signup_screen.dart';

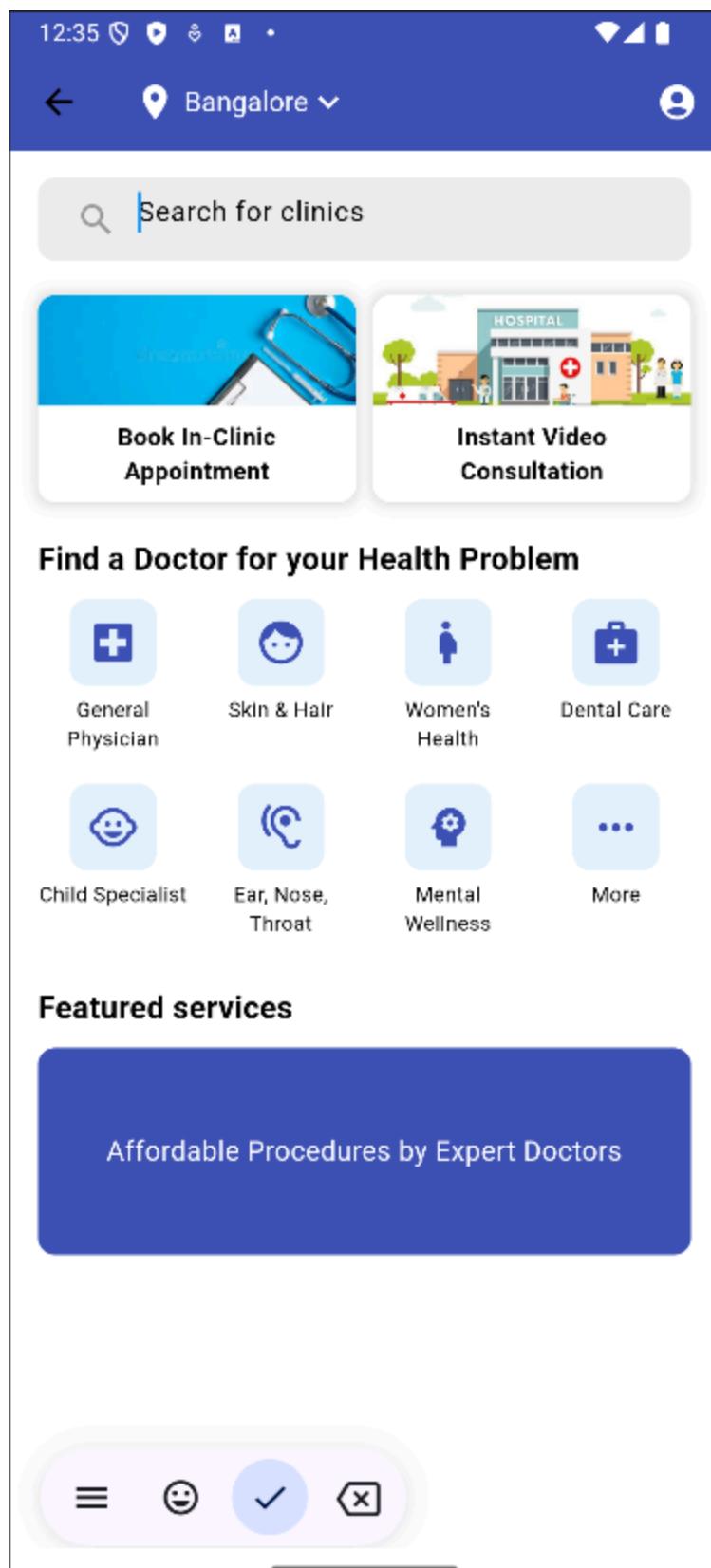
void main() {
  runApp(const LifelineApp()); // Ensure const is used if applicable
}

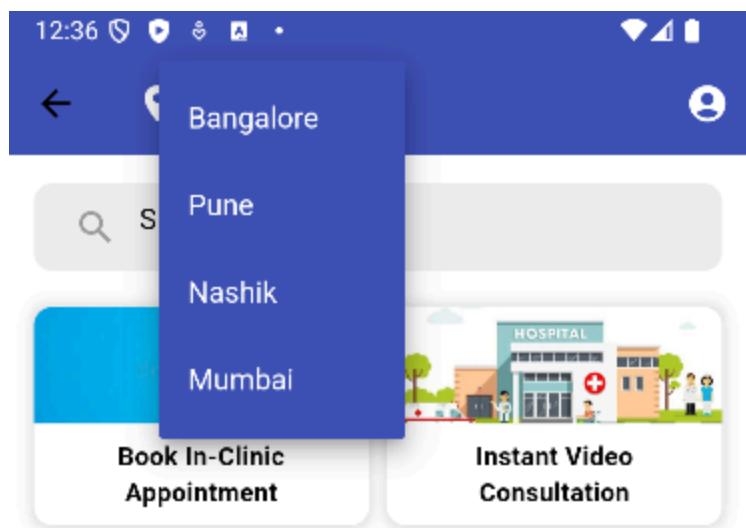
class LifelineApp extends StatelessWidget {
  const LifelineApp({super.key}); // Add key parameter for best
  practice

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Lifeline',
      theme: ThemeData(

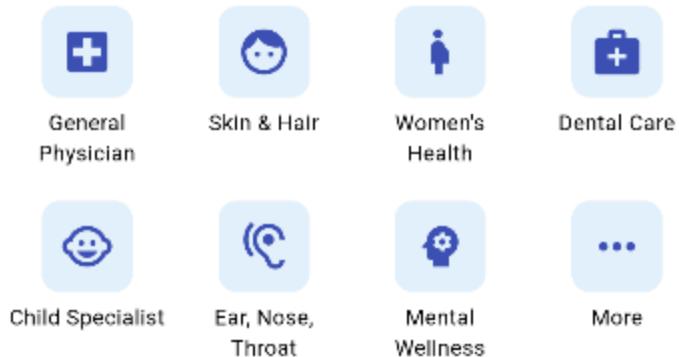
```

```
        primaryColor: Colors.green,  
        colorScheme: ColorScheme.fromSwatch().copyWith(secondary:  
Colors.blue),  
) ,  
initialRoute: '/login',  
routes: {  
  '/login': (context) => LoginScreen(),  
  '/signup': (context) => SignupScreen(),  
  '/home': (context) => HomeScreen(),  
,  
}  
}
```





### Find a Doctor for your Health Problem



### Featured services

Affordable Procedures by Expert Doctors

**Conclusion:**

In this experiment, we successfully implemented various UI components such as a dropdown list, search bar, buttons, and icons to enhance the application's usability. During development, we encountered minor issues such as dropdown styling inconsistencies and widget alignment problems, which were resolved through debugging, modifying the widget properties, and adjusting padding and alignment settings.

## Experiment 4

### Form Widgets

**Name:** Sharvari Kishor More

**Class:** D15B

**Roll no.:** 35

**Aim:** To create an interactive Form using form widget.

#### **Introduction:**

Mobile Application Development (MAD) focuses on creating applications that run on mobile devices, utilizing frameworks such as Flutter. We implemented fundamental concepts of mobile app development, including user input handling, validation, and navigation between screens. The primary objective was to enhance the user experience by implementing structured forms and ensuring seamless transitions between different pages.

#### **General Overview:**

In mobile app development, forms play a crucial role in collecting user input. A well-designed form includes input fields, validation checks, and submission mechanisms to ensure data integrity. TextFormField widgets in Flutter allow validation logic to be embedded directly within the UI components. Error messages guide users in entering correct information, ensuring smooth app interactions. Additionally, navigation between screens is essential for multi-page applications, achieved using Navigator methods in Flutter.

#### **Implementation Details:**

In this experiment, we designed a Login and Signup Screen using Flutter. The user interface was enhanced with a background image, translucent text fields, and a clean layout. A GlobalKey was used to manage the form state, ensuring proper validation of input fields.

- **Login Screen:** It includes email and password fields with validation logic. The password field has a toggle feature for visibility control. Upon successful validation, the user is redirected to the home screen.
- **Signup Screen:** It collects user details like name, email, and password. The password validation enforces security constraints such as length, special characters, and numeric values.
- **Navigation & UI Enhancements:** A visually appealing design was implemented, maintaining consistency in color schemes (blue, light blue, and white). The Navigator class was used to manage screen transitions seamlessly.

Code:

Login\_screen.dart

```
import 'package:flutter/material.dart';
import 'package:lifeline/home_screen.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({super.key});

  @override
  LoginScreenState createState() => LoginScreenState();
}

class LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  bool _obscurePassword = true;

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        fit: StackFit.expand,
        children: [
          Image.asset("assets/images/dr1.jpg", fit: BoxFit.cover),
          Container(color: Colors.black.withOpacity(0.4)),
          Center(
            child: Padding(
              padding: EdgeInsets.all(20),
              child: Form(
                key: _formKey,
                child: Column(
```

```
        mainAxisSize: MainAxisSize.min,
        children: [
            Text("Welcome to Lifeline",
                style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold, color: Colors.white)),
            SizedBox(height: 20),
            _buildTextField(_emailController, "Email",
Icons.email, false),
            SizedBox(height: 10),
            _buildTextField(_passwordController, "Password",
Icons.lock, true),
            SizedBox(height: 20),
            ElevatedButton(
                style: ElevatedButton.styleFrom(backgroundColor:
Colors.blue),
                onPressed: () {
                    if (_formKey.currentState!.validate()) {
                        Navigator.pushReplacement(context,
MaterialPageRoute(builder: (context) => HomeScreen()));
                    }
                },
                child: Text("Login", style: TextStyle(color:
Colors.white)),
            ),
            TextButton(
                onPressed: () => Navigator.pushNamed(context,
'/signup'),
                child: Text("Don't have an account? Sign Up", style:
TextStyle(color: Colors.white)),
            ),
        ],
    ),
),
],
),
),
),
],
),
),
),
);
}
```

```

Widget _buildTextField(TextEditingController controller, String label,
IconData icon, bool isPassword) {
    return TextFormField(
        controller: controller,
        obscureText: isPassword ? _obscurePassword : false,
        style: TextStyle(color: Colors.white),
        decoration: InputDecoration(
            labelText: label,
            labelStyle: TextStyle(color: Colors.white70),
            prefixIcon: Icon(icon, color: Colors.white),
            filled: true,
            fillColor: Colors.white.withOpacity(0.2),
            border: OutlineInputBorder(borderRadius:
BorderRadius.circular(10)),
            suffixIcon: isPassword
                ? IconButton(
                    icon: Icon(_obscurePassword ? Icons.visibility_off :
Icons.visibility, color: Colors.white),
                    onPressed: () => setState(() => _obscurePassword =
!_obscurePassword),
                )
                : null,
        ),
        validator: (value) => value == null || value.isEmpty ? "$label
cannot be empty" : null,
    );
}
}
}

```

**Signup\_screen.dart**

```

import 'package:flutter/material.dart';

class SignupScreen extends StatelessWidget {
    const SignupScreen({super.key});

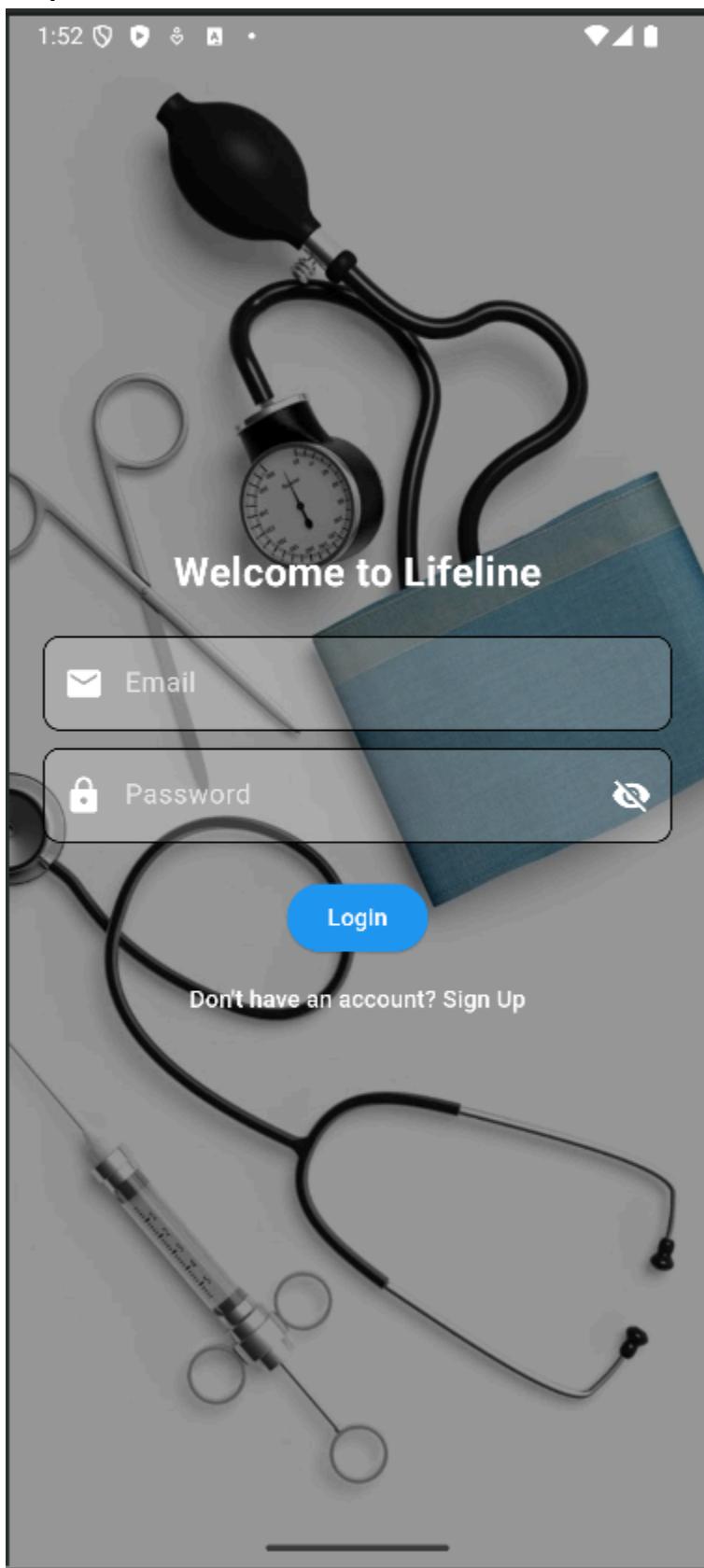
    @override
    Widget build(BuildContext context) {
        final _formKey = GlobalKey<FormState>();
        final _nameController = TextEditingController();
        final _emailController = TextEditingController();

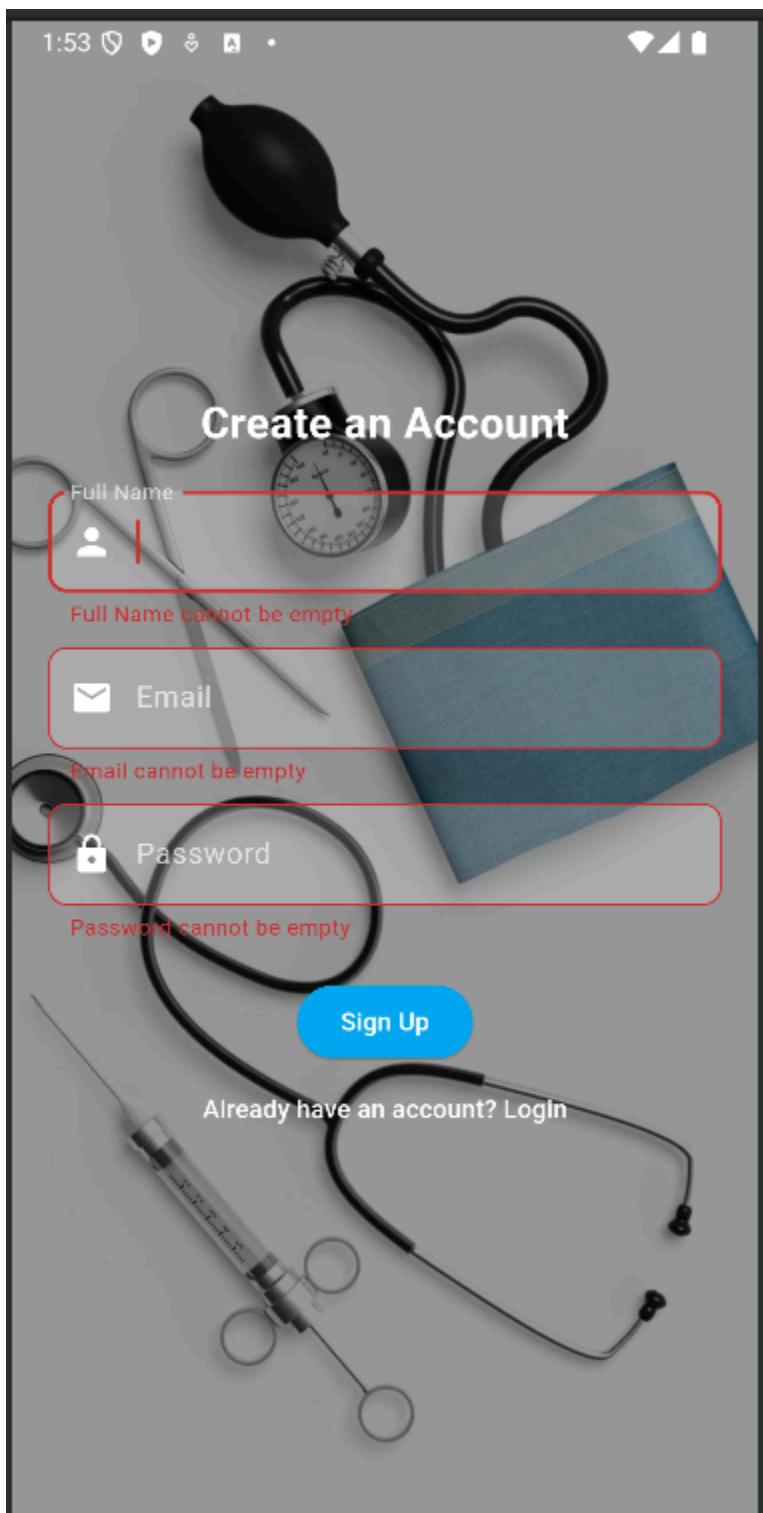
```

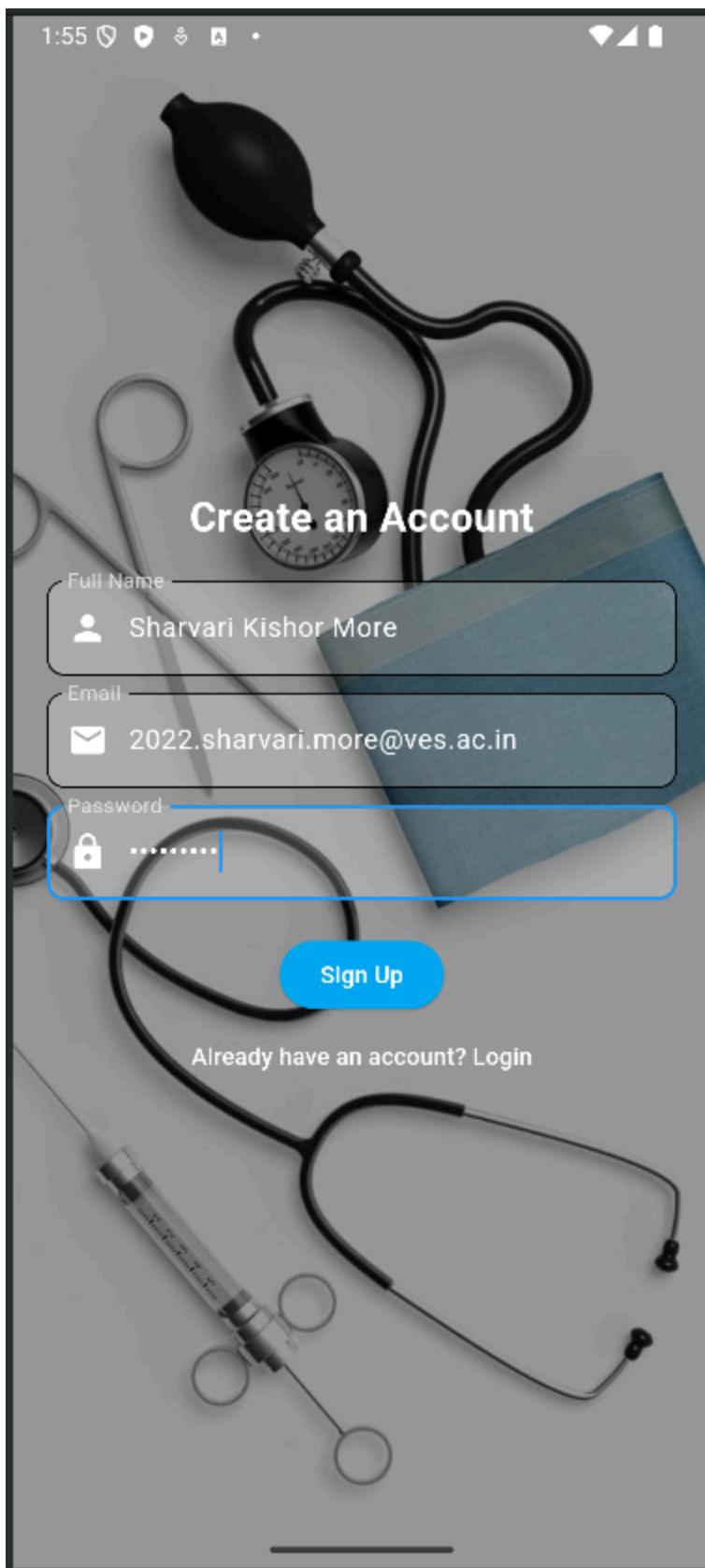
```
final _passwordController = TextEditingController();  
  
return Scaffold(  
    body: Stack(  
        fit: StackFit.expand,  
        children: [  
            Image.asset("assets/images/dr1.jpg", fit: BoxFit.cover),  
            Container(color: Colors.black.withOpacity(0.4)),  
            Center(  
                child: Padding(  
                    padding: EdgeInsets.all(20),  
                    child: Form(  
                        key: _formKey,  
                        child: Column(  
                            mainAxisSize: MainAxisSize.min,  
                            children: [  
                                Text("Create an Account", style: TextStyle(fontSize:  
24, fontWeight: FontWeight.bold, color: Colors.white)),  
                                SizedBox(height: 20),  
                                _buildTextField(_nameController, "Full Name",  
Icons.person, false),  
                                SizedBox(height: 10),  
                                _buildTextField(_emailController, "Email",  
Icons.email, false),  
                                SizedBox(height: 10),  
                                _buildTextField(_passwordController, "Password",  
Icons.lock, true),  
                                SizedBox(height: 20),  
                                ElevatedButton(  
                                    style: ElevatedButton.styleFrom(backgroundColor:  
Colors.lightBlue),  
                                    onPressed: () {  
                                        if (_formKey.currentState!.validate()) {  
                                            Navigator.pushReplacementNamed(context,  
' /home' );  
                                        }  
                                    },  
                                    child: Text("Sign Up", style: TextStyle(color:  
Colors.white)),  
                                ),  
                            ],  
                        ),  
                    ),  
                ),  
            ),  
        ],  
    ),  
);
```

```
        TextButton(  
            onPressed: () => Navigator.pushNamed(context,  
'login'),  
            child: Text("Already have an account? Login", style:  
TextStyle(color: Colors.white)),  
        ),  
    ],  
) ,  
) ,  
) ,  
) ,  
),  
],  
) ;  
}  
  
Widget _buildTextField(TextEditingController controller, String label,  
IconData icon, bool isPassword) {  
    return TextFormField(  
        controller: controller,  
        obscureText: isPassword,  
        style: TextStyle(color: Colors.white),  
        decoration: InputDecoration(  
            labelText: label,  
            labelStyle: TextStyle(color: Colors.white70),  
            prefixIcon: Icon(icon, color: Colors.white),  
            filled: true,  
            fillColor: Colors.white.withOpacity(0.2),  
            border: OutlineInputBorder(borderRadius:  
BorderRadius.circular(10)),  
        ),  
        validator: (value) => value == null || value.isEmpty ? "$label  
cannot be empty" : null,  
    );  
}  
}
```

**Output:**







**Conclusion:**

This experiment strengthened our understanding of form validation, state management, and navigation in Flutter. We encountered challenges such as handling input errors and UI design consistency, which were resolved through structured debugging and iterative design improvements.

## Experiment 5

### Navigation , Routing and Gestures

**Name: Sharvari Kishor More**

**Class: D15B**

**Roll no.: 35**

**Aim:** To apply navigation, routing and gestures in Flutter App

#### **Theory:**

Flutter provides powerful tools to implement user interactions, navigation, and routing in mobile applications. This experiment focuses on three key concepts:

1. **Gestures:** Gestures allow users to interact with the UI using touch-based actions like tapping, swiping, and long pressing. Flutter provides the GestureDetector widget to handle such interactions.
2. **Navigation:** Navigation is the process of switching between different screens (pages) within an application. Flutter's Navigator class manages the navigation stack and allows users to move between screens using push and pop methods.
3. **Routing:** Routing is a way to define and manage the different screens in an application. Named routes can be used to simplify navigation and organize the app's structure better.

#### **Implementation in the Given Code**

In this experiment, we have implemented gestures, navigation, and routing as follows:

- **Gestures:** The **DoctorCategoryButton** widget uses **GestureDetector** to handle user interactions. When a user taps on a category, a SnackBar displays the selected category.
- **Navigation:** The login and signup pages utilize navigation to move between screens using the **Navigator.push()** method.
- **Routing:** The main.dart file contains the routing logic, which helps in switching between different pages smoothly.

**Code:****Home\_screen.dart**

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

class HomeScreen extends StatefulWidget {

  const HomeScreen({super.key});

  @override

  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {

  String selectedCity = "Bangalore";

  final List<String> cities = ["Bangalore", "Pune", "Nashik", "Mumbai"];

  @override

  Widget build(BuildContext context) {

    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.indigo,
        title: Row(
          children: [
            Icon(Icons.location_on, color: Colors.white),
            SizedBox(width: 5),
            DropdownButton<String>(
              value: selectedCity,
              dropdownColor: Colors.indigo,
```

```
        icon: Icon(Icons.keyboard_arrow_down, color: Colors.white),  
        underline: SizedBox(),  
        style: TextStyle(color: Colors.white, fontSize: 16),  
        onChanged: (String? newValue) {  
            setState(() {  
                selectedCity = newValue!;  
            });  
        },  
        items: cities.map<DropdownMenuItem<String>>((String city) {  
            return DropdownMenuItem<String>(  
                value: city,  
                child: Text(city, style: TextStyle(color:  
Colors.white)),  
            );  
        }).toList(),  
    ),  
],  
),  
actions: [  
    IconButton(  
        icon: Icon(Icons.account_circle, color: Colors.white),  
        onPressed: () {},  
    ),  
],
```

```
) ,  
  
body: SingleChildScrollView(  
  
    child: Padding(  
  
        padding: EdgeInsets.all(16.0),  
  
        child: Column(  
  
            mainAxisAlignment: MainAxisAlignment.start,  
  
            children: [  
  
                Container(  
  
                    padding: EdgeInsets.symmetric(horizontal: 10),  
  
                    decoration: BoxDecoration(  
  
                        color: Colors.grey[200],  
  
                        borderRadius: BorderRadius.circular(10),  
  
                    ),  
  
                    child: TextField(  
  
                        decoration: InputDecoration(  
  
                            hintText: "Search for clinics",  
  
                            border: InputBorder.none,  
  
                            prefixIcon: Icon(Icons.search, color: Colors.grey),  
  
                        ),  
  
                ),  
  
                SizedBox(height: 20),  
  
                // Appointment & Consultation Section  
            ],  
        ),  
    ),  
);
```

```
Row (  
  
    mainAxisAlignment: MainAxisAlignment.spaceBetween,  
  
    children: [  
  
        Expanded (  
  
            child: AppointmentCard (  
  
                title: "Book In-Clinic Appointment",  
  
                imagePath: "assets/images/images.jpg",  
  
            ),  
  
        ),  
  
        SizedBox (width: 10),  
  
        Expanded (  
  
            child: AppointmentCard (  
  
                title: "Instant Video Consultation",  
  
                imagePath: "assets/images/hospital.jpg",  
  
            ),  
  
        ),  
  
    ],  
  
) ,  
  
SizedBox (height: 20),  
  
Text (  
  
    "Find a Doctor for your Health Problem",  
  
    style: TextStyle (fontSize: 18, fontWeight:  
    FontWeight.bold),  
  
),
```

```
SizedBox(height: 10),  
  
GridView.builder(  
  
    shrinkWrap: true,  
  
    physics: NeverScrollableScrollPhysics(),  
  
    gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(  
  
        crossAxisCount: 4,  
  
        crossAxisSpacing: 10,  
  
        mainAxisSpacing: 10,  
  
        childAspectRatio: 0.9,  
  
) ,  
  
itemCount: doctorCategories.length,  
  
itemBuilder: (context, index) {  
  
    return DoctorCategoryButton(doctorCategories[index]);  
  
} ,  
  
) ,  
  
SizedBox(height: 20),  
  
Text(  
  
    "Featured services",  
  
    style: TextStyle(fontSize: 18, fontWeight:  
FontWeight.bold),  
  
) ,  
  
SizedBox(height: 10),  
  
Container(  
  
    width: double.infinity,
```

```
height: 120,  
  
decoration: BoxDecoration(  
  
  color: Colors.indigo,  
  
  borderRadius: BorderRadius.circular(10),  
  
) ,  
  
child: Center(  
  
  child: Text(  
  
    "Affordable Procedures by Expert Doctors",  
  
    style: TextStyle(color: Colors.white, fontSize: 16),  
  
    textAlign: TextAlign.center,  
  
) ,  
  
) ,  
  
],  
  
) ,  
  
) ,  
  
) ;  
  
}  
  
}  
  
class AppointmentCard extends StatelessWidget {  
  
final String title;
```

```
final String imagePath;

const AppointmentCard({
    required this.title,
    required this.imagePath,
    super.key,
}) ;

@override

Widget build(BuildContext context) {
    return Container(
        height: 120,
        decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(10),
            boxShadow: [
                BoxShadow(color: Colors.grey.shade300, blurRadius: 5,
                spreadRadius: 2),
            ],
        ),
        child: Column(
            children: [
                Expanded(
                    child: ClipRRect(
                        borderRadius: BorderRadius.vertical(top:
                        Radius.circular(10)),
                ),
            ],
        ),
    );
}
```

```
        child: Image.asset(imagePath, width: double.infinity, fit:  
 BoxFit.cover),  
) ,  
) ,  
 Padding(  
 padding: EdgeInsets.all(8.0),  
 child: Text(title, textAlign: TextAlign.center, style:  
 TextStyle(fontSize: 14, fontWeight: FontWeight.bold)),  
) ,  
 ] ,  
) ,  
) ;  
}  
}  
  
class DoctorCategoryButton extends StatelessWidget {  
 final DoctorCategory category;  
 const DoctorCategoryButton(this.category, {super.key});  
 @override  
 Widget build(BuildContext context) {  
 return GestureDetector(  
 onTap: () {  
 ScaffoldMessenger.of(context).showSnackBar(  
 SnackBar(content: Text("Selected: ${category.title}")),  
 );  
 };
```

```
        } ,  
  
        child: Column(  
  
            children: [  
  
                Container(  
  
                    padding: EdgeInsets.all(10) ,  
  
                    decoration: BoxDecoration(  
  
                        color: Colors.blue[50] ,  
  
                        borderRadius: BorderRadius.circular(10) ,  
  
                    ) ,  
  
                    child: Icon(category.icon, color: Colors.indigo, size: 30) ,  
  
                ) ,  
  
                SizedBox(height: 5) ,  
  
                Text(category.title, textAlign: TextAlign.center, style:  
                    TextStyle(fontSize: 12)) ,  
  
            ] ,  
        ) ,  
    ) ;  
}  
}  
  
class DoctorCategory {  
  
    final String title;  
  
    final IconData icon;  
  
    DoctorCategory({required this.title, required this.icon});  
}
```

```
List<DoctorCategory> doctorCategories = [  
    DoctorCategory(title: "General Physician", icon: Icons.local_hospital),  
    DoctorCategory(title: "Skin & Hair", icon: Icons.face),  
    DoctorCategory(title: "Women's Health", icon: Icons.pregnant_woman),  
    DoctorCategory(title: "Dental Care", icon: Icons.medical_services),  
    DoctorCategory(title: "Child Specialist", icon: Icons.child_care),  
    DoctorCategory(title: "Ear, Nose, Throat", icon: Icons.hearing),  
    DoctorCategory(title: "Mental Wellness", icon: Icons.psychology),  
    DoctorCategory(title: "More", icon: Icons.more_horiz),  
];
```

### **Login\_screen.dart**

```
import 'package:flutter/material.dart';  
  
import 'package:lifeline/home_screen.dart';  
  
class LoginScreen extends StatefulWidget {  
  
    const LoginScreen({super.key});  
  
    @override  
  
    LoginScreenState createState() => LoginScreenState();  
}  
  
class LoginScreenState extends State<LoginScreen> {  
  
    final _formKey = GlobalKey<FormState>();  
  
    final _emailController = TextEditingController();  
  
    final _passwordController = TextEditingController();  
  
    bool _obscurePassword = true;
```

```
@override  
  
void dispose() {  
  
    _emailController.dispose();  
  
    _passwordController.dispose();  
  
    super.dispose();  
  
}  
  
@override  
  
Widget build(BuildContext context) {  
  
    return Scaffold(  
  
        body: Stack(  
  
            fit: StackFit.expand,  
  
            children: [  
  
                Image.asset("assets/images/dr1.jpg", fit: BoxFit.cover),  
  
                Container(color: Colors.black.withOpacity(0.4)),  
  
                Center(  
  
                    child: Padding(  
  
                        padding: EdgeInsets.all(20),  
  
                        child: Form(  
  
                            key: _formKey,  
  
                            child: Column(  
  
                                mainAxisSize: MainAxisSize.min,  
  
                                children: [  
  
                                    Text("Welcome to Lifeline",  
                                         style: TextStyle(fontSize: 20, color: Colors.white))  
                                ]  
                            )  
                        )  
                    )  
                )  
            ]  
        )  
    );  
}
```

```
        style: TextStyle(fontSize: 24, fontWeight:  
FontWeight.bold, color: Colors.white)),  
  
        SizedBox(height: 20),  
  
        _buildTextField(_emailController, "Email",  
Icons.email, false),  
  
        SizedBox(height: 10),  
  
        _buildTextField(_passwordController, "Password",  
Icons.lock, true),  
  
        SizedBox(height: 20),  
  
        ElevatedButton(  
  
            style: ElevatedButton.styleFrom(backgroundColor:  
Colors.blue),  
  
            onPressed: () {  
  
                if (_formKey.currentState!.validate()) {  
  
                    Navigator.pushReplacement(context,  
MaterialPageRoute(builder: (context) => HomeScreen()));  
  
                }  
  
            },  
  
            child: Text("Login", style: TextStyle(color:  
Colors.white)),  
  
,  
  
        TextButton(  
  
            onPressed: () => Navigator.pushNamed(context,  
'/signup'),  
  
            child: Text("Don't have an account? Sign Up", style:  
TextStyle(color: Colors.white)),  
  
,
```

```
        ] ,  
        ) ,  
        ) ,  
        ) ,  
        ) ,  
    ] ,  
    ) ,  
);  
}  
  
Widget _buildTextField(TextEditingController controller, String label,  
IconData icon, bool isPassword) {  
  
    return TextFormField(  
        controller: controller,  
        obscureText: isPassword ? _obscurePassword : false,  
        style: TextStyle(color: Colors.white),  
        decoration: InputDecoration(  
            labelText: label,  
            labelStyle: TextStyle(color: Colors.white70),  
            prefixIcon: Icon(icon, color: Colors.white),  
            filled: true,  
            fillColor: Colors.white.withOpacity(0.2),  
            border: OutlineInputBorder(borderRadius:  
                BorderRadius.circular(10)),  
            suffixIcon: isPassword
```

```

        ? IconButton(
            icon: Icon(_obscurePassword ? Icons.visibility_off :
Icons.visibility, color: Colors.white),
            onPressed: () => setState(() => _obscurePassword =
!_obscurePassword),
        )
        : null,
    ),
    validator: (value) => value == null || value.isEmpty ? "$label
cannot be empty" : null,
);
}
}

```

**Signup\_screen.dart**

```

import 'package:flutter/material.dart';

class SignupScreen extends StatelessWidget {
    const SignupScreen({super.key});

    @override
    Widget build(BuildContext context) {
        final _formKey = GlobalKey<FormState>();
        final _nameController = TextEditingController();
        final _emailController = TextEditingController();
        final _passwordController = TextEditingController();
        return Scaffold(

```

```
body: Stack(  
    fit: StackFit.expand,  
    children: [  
        Image.asset("assets/images/dr1.jpg", fit: BoxFit.cover),  
        Container(color: Colors.black.withOpacity(0.4)),  
        Center(  
            child: Padding(  
                padding: EdgeInsets.all(20),  
                child: Form(  
                    key: _formKey,  
                    child: Column(  
                        mainAxisSize: MainAxisSize.min,  
                        children: [  
                            Text("Create an Account", style: TextStyle(fontSize:  
                                24, fontWeight: FontWeight.bold, color: Colors.white)),  
                            SizedBox(height: 20),  
                            _buildTextField(_nameController, "Full Name",  
                                Icons.person, false),  
                            SizedBox(height: 10),  
                            _buildTextField(_emailController, "Email",  
                                Icons.email, false),  
                            SizedBox(height: 10),  
                            _buildTextField(_passwordController, "Password",  
                                Icons.lock, true),  
                            SizedBox(height: 20),  
                        ]  
                    )  
                )  
            )  
        )  
    ]  
);
```

```
ElevatedButton(  
    style: ElevatedButton.styleFrom(backgroundColor:  
Colors.lightBlue),  
    onPressed: () {  
        if (_formKey.currentState!.validate()) {  
            Navigator.pushReplacementNamed(context,  
'/_home');  
        }  
    },  
    child: Text("Sign Up",style: TextStyle(color:  
Colors.white)),  
),  
TextButton(  
    onPressed: () => Navigator.pushNamed(context,  
'_login'),  
    child: Text("Already have an account? Login", style:  
TextStyle(color: Colors.white)),  
),  
],  
) ,  
) ,  
],  
) ,
```

```
) ;  
  
}  
  
Widget _buildTextField(TextEditingController controller, String label,  
IconData icon, bool isPassword) {  
  
    return TextFormField(  
  
        controller: controller,  
  
        obscureText: isPassword,  
  
        style: TextStyle(color: Colors.white),  
  
        decoration: InputDecoration(  
  
            labelText: label,  
  
            labelStyle: TextStyle(color: Colors.white70),  
  
            prefixIcon: Icon(icon, color: Colors.white),  
  
            filled: true,  
  
            fillColor: Colors.white.withOpacity(0.2),  
  
            border: OutlineInputBorder(borderRadius:  
BorderRadius.circular(10)),  
  
) ,  
  
    validator: (value) => value == null || value.isEmpty ? "$label  
cannot be empty" : null,  
  
) ;  
  
}  
}
```

**Main.dart**

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

```
import 'home_screen.dart';

import 'login_screen.dart';

import 'signup_screen.dart';

void main() {

  runApp(const LifelineApp()); // Ensure const is used if applicable
}

class LifelineApp extends StatelessWidget {

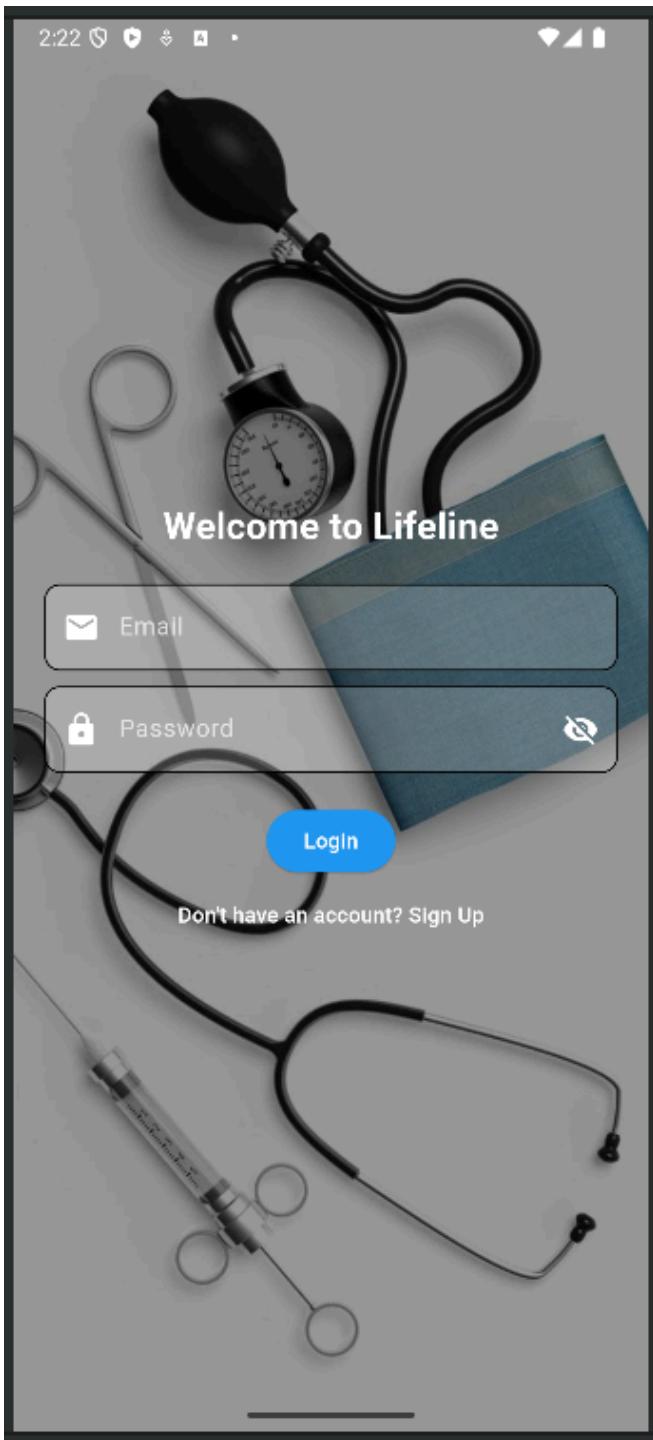
  const LifelineApp({super.key}); // Add key parameter for best practice

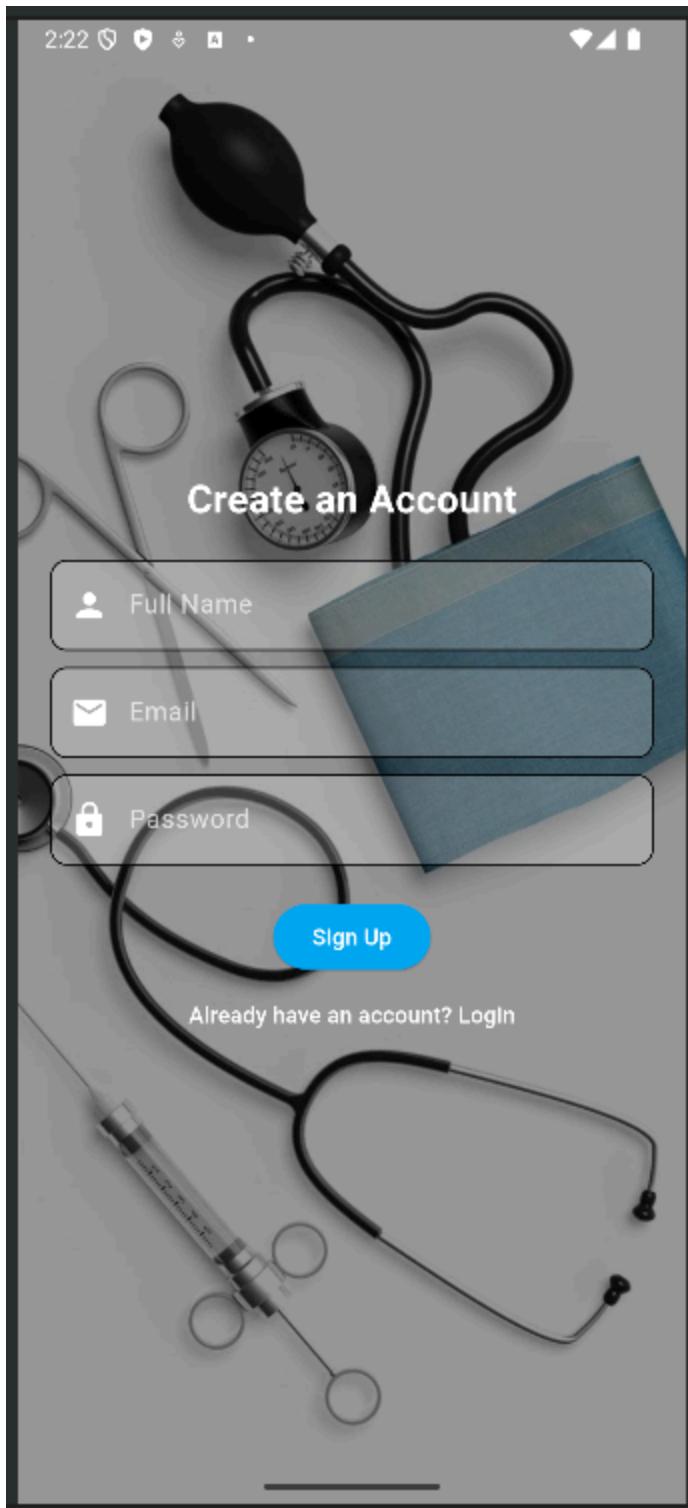
  @override

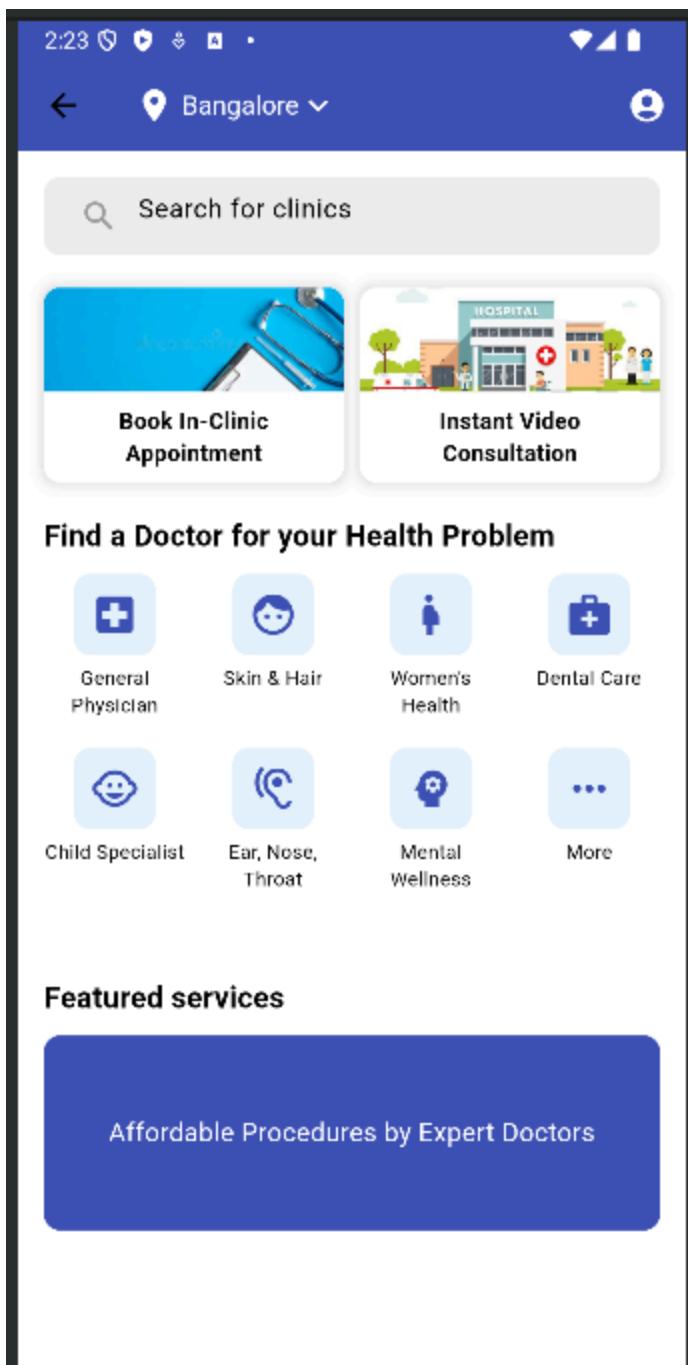
  Widget build(BuildContext context) {

    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Lifeline',
      theme: ThemeData(
        primaryColor: Colors.green,
        colorScheme: ColorScheme.fromSwatch().copyWith(secondary: Colors.blue),
      ),
      initialRoute: '/login',
      routes: {
        '/login': (context) => LoginScreen(),
        '/signup': (context) => SignupScreen(),
        '/home': (context) => HomeScreen(),
      }
    );
  }
}
```

```
} ,  
);  
}
```

**Output:**



**Conclusion:**

In this experiment, we successfully implemented gestures for user interaction, navigation for screen transitions, and routing for better app structure. Initially, we faced issues with state management in the dropdown menu and improper routing setup, which we resolved by properly using `setState()` and defining named routes correctly.

## MPL Practical 06

**Name: Sharvari Kishor More**

**Class: D15B**

**Roll no.: 37**

**Aim:** To integrate Firebase Authentication in a Flutter app.

**Theory:**

### Firebase Authentication in Our Code

User Registration – Users can sign up using their email and password. Upon successful registration, a verification email is sent.

Email Verification – Users must verify their email before they can log in.

User Login – Only verified users can log in to the application.

Google Sign-In – Users have the option to sign in using their Google account.

Password Reset – If users forget their password, they can request a reset link via email.

Logout – Users can securely log out from their session.

### Firebase Integration Steps

- Configured a Firebase project and enabled authentication services.
- Initialized Firebase in the Flutter project and added the required dependencies.
- Implemented authentication functions in `auth_service.dart` to handle user sign-up, login, password reset, and logout.

Code:

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

import 'package:flutter/material.dart';

import 'package:sharvari/reuseable_widgets/reusable_widgets.dart';

class ResetPassword extends StatefulWidget {

  const ResetPassword({super.key});

  @override

  _ResetPasswordState createState() => _ResetPasswordState();
}
```

}

```
class _ResetPasswordState extends State<ResetPassword> {

    final TextEditingController _emailTextController = TextEditingController();
    final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
    bool _isLoading = false;

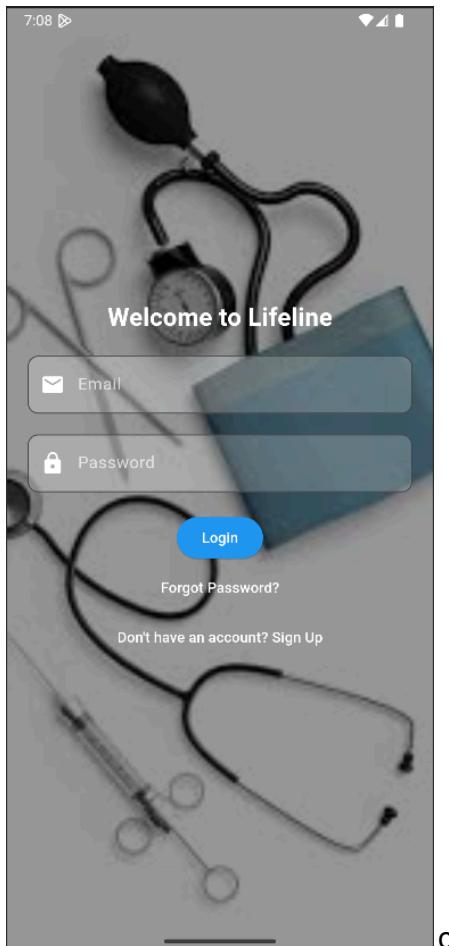
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Stack(
                fit: StackFit.expand,
                children: [
                    Image.asset("assets/images/dr1.jpg", fit: BoxFit.cover),
                    Container(color: Colors.black.withOpacity(0.4)),
                    Center(
                        child: Padding(
                            padding: const EdgeInsets.all(20),
                            child: Form(
                                key: _formKey,
                                child: Column(
                                    mainAxisSize: MainAxisSize.min,
                                    children: [
                                        const Text(
                                            "Reset Password",
                                            style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color: Colors.white),
                                        ),
                                    ],
                                ),
                            ),
                        ),
                    ),
                ],
            ),
        );
    }
}
```

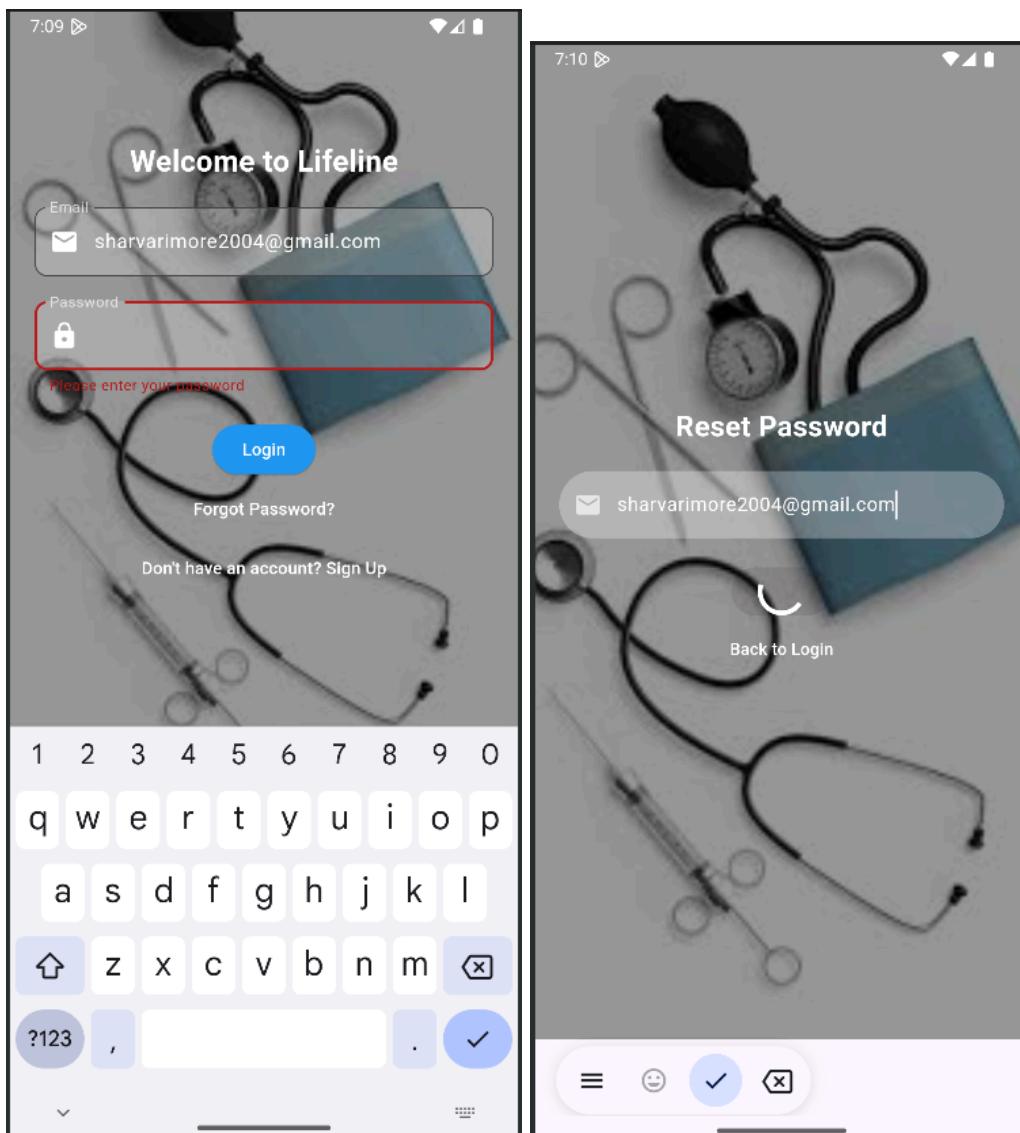
```
 ),  
  
const SizedBox(height: 20),  
  
reuseableTextField(  
  
    "Enter Email Id",  
  
    Icons.email,  
  
    false,  
  
    _emailTextController,  
  
    (value) {  
  
        if (value == null || value.isEmpty) {  
  
            return "Please enter your email";  
  
        }  
  
        if  
(!RegExp(r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$").hasMatch(value)) {  
  
            return "Please enter a valid email address";  
  
        }  
  
        return null;  
    },  
),  
  
const SizedBox(height: 20),  
  
ElevatedButton(  
  
    style: ElevatedButton.styleFrom(backgroundColor: Colors.lightBlue),  
  
    onPressed: _isLoading ? null : _resetPassword,  
  
    child: _isLoading  
  
        ? const CircularProgressIndicator(color: Colors.white)  
  
        : const Text("Reset Password", style: TextStyle(color: Colors.white)),  
),  
  
TextButton(  
  
    
```

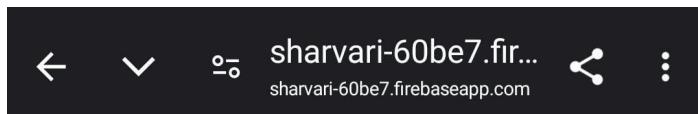
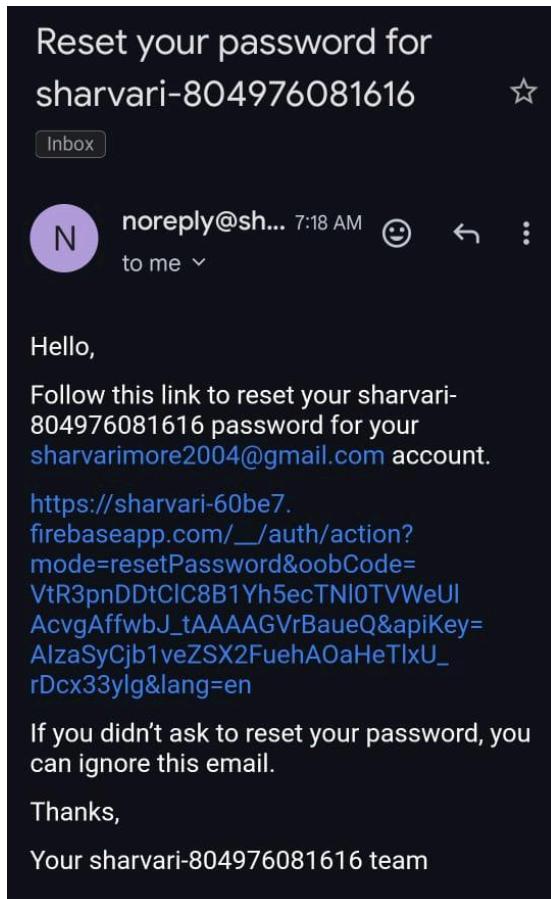
```
onPressed: () => Navigator.pop(context),  
        child: const Text("Back to Login", style: TextStyle(color: Colors.white)),  
    ),  
],  
,  
,  
),  
],  
,  
);  
}  
  
}
```

```
void _resetPassword() async {  
    if (_formKey.currentState!.validate()) return;  
  
    setState(() => _isLoading = true);  
    String email = _emailTextController.text.trim();  
  
    try {  
        await FirebaseAuth.instance.sendPasswordResetEmail(email: email);  
        setState(() => _isLoading = false);  
        _showSnackbar("Password reset email sent successfully!");  
        Navigator.of(context).pop();  
    } on FirebaseAuthException catch (e) {  
        setState(() => _isLoading = false);  
    }  
}
```

```
_showSnackbar("Error: ${e.message}");  
 } catch (e) {  
 setState(() => _isLoading = false);  
 _showSnackbar("An unexpected error occurred.");  
 }  
  
}  
  
  
void _showSnackbar(String message) {  
 ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text(message)));  
}  
}
```

**Screenshots:**







## Password changed

You can now sign in with your new password

A screenshot of the Firebase Authentication console. The left sidebar shows "Authentication" is selected. The main area displays a table of users with columns: Identifier, Providers, Created, Signed In, and User UID. The table contains three rows of data:

Identifier	Providers	Created	Signed In	User UID
2022.yash.naikwadi@v...	✉️	Mar 5, 2025	Mar 5, 2025	lkqleXderLhfoRHwaj1difN1at1
sharvarimore2004@gm...	✉️	Mar 5, 2025	Mar 6, 2025	rsCW2gOZ2gNmZl2J0qy0G68...
2022.sharvari.more@ve...	✉️	Mar 5, 2025	Mar 5, 2025	lGrFQly6OdPWh6TsqdKIU7ht...

## Conclusion:

Integrating Firebase Authentication in our Flutter app enhanced security and user management. While implementing it, I encountered issues like delayed email verification updates and Firebase exceptions during sign-in. I resolved them by manually refreshing user authentication states and handling errors using try-catch blocks. Debugging authentication flows and ensuring proper dependency setup helped streamline the process, resulting in a smooth authentication system.

## Practical 7

**Name: Sharvari Kishor More**

**Class: D15B**

**Roll No.: 35**

### Aim:

To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.

### Theory:

A **Progressive Web App (PWA)** enhances a regular web application by making it installable and capable of running offline like a native app. One of the key components that make a web app a PWA is the **Web App Manifest** file — a simple JSON file that provides essential metadata about the app.

In this experiment, we created a manifest.json file for an E-commerce web application named **Festival Combo Shop**. This file includes details like:

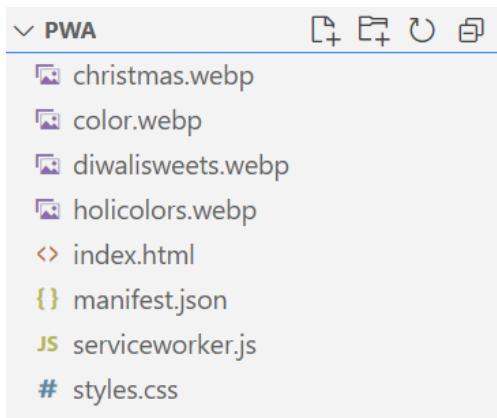
- **App name and short name** for display
- **Start URL, display mode** (standalone), **background color**, and **theme color**
- **A description** for the app
- A list of **icons** in required formats (PNG, 192x192 and 512x512, with purpose: "any")
- **Screenshots** of the app to showcase its interface

These details allow modern browsers to recognize the web app as a PWA and prompt users with an “**Add to Home Screen**” option.

We also fixed common validation issues such as:

- Ensuring icons are of minimum size 144x144
- Using proper image formats like PNG and WEBP
- Setting the required sizes and purpose attributes for icons

### Folder Structure:



**Index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Festival Combo Shop</title>
    <link rel="manifest" href="manifest.json">
    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">
    <link rel="stylesheet"
        href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
<style>
    .festival-card {
        transition: transform 0.3s ease, box-shadow 0.3s ease;
    }
    .festival-card:hover {
        transform: translateY(-5px);
        box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);
    }
    .btn-animation {
        transition: all 0.3s ease;
    }
    .btn-animation:hover {
        transform: translateY(-2px);
        box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    }
    .hero-section {
        background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('/images/color.webp');
    }
</style>
</head>

<script>
    if ('serviceWorker' in navigator) {
        window.addEventListener('load', () => {
            navigator.serviceWorker.register('/serviceworker.js')
                .then(reg => {
                    console.log('Service Worker registered successfully:', reg.scope);
                })
    })
</script>
```

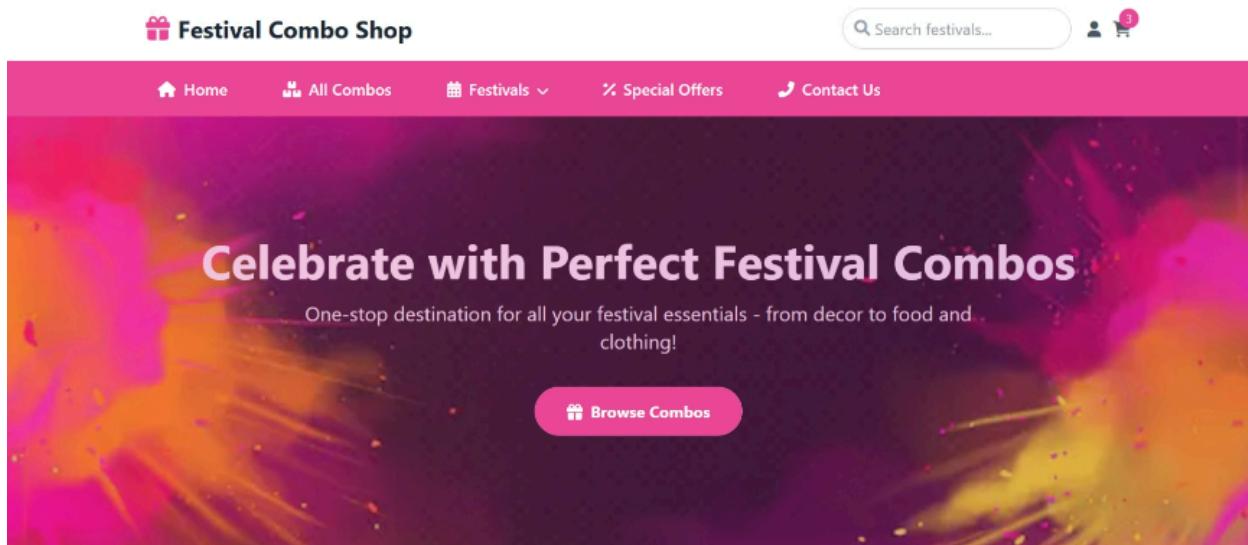
```
.catch(err => {
  console.error('Service Worker registration failed:', err);
});
}
</script>
```

**Manifest.json**

```
{
  "name": "Festival Combo Shop",
  "short_name": "Festival Combos",
  "start_url": "./index.html",
  "display": "standalone",
  "background_color": "#f9fafb",
  "theme_color": "#ec4899",
  "description": "Celebrate with perfect festival combos for Diwali, Holi, Christmas, and more!",
  "icons": [
    {
      "src": "/images/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/images/icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "screenshots": [
    {
      "src": "/images/color.webp",
      "type": "image/webp",
      "sizes": "1280x720"
    },
    {
      "src": "/images/diwalisweets.webp",
      "type": "image/webp",
      "sizes": "800x600"
    },
  ]
},
```

```
{  
  "src": "/images/holicolors.webp",  
  "type": "image/webp",  
  "sizes": "800x600"  
},  
{  
  "src": "/images/christmas.webp",  
  "type": "image/webp",  
  "sizes": "800x600"  
}  
]  
}
```

**Output:**

The screenshot shows the homepage of a website called "Festival Combo Shop". The header features a search bar with the placeholder "Search festivals..." and a user icon with a notification count of 3. The main banner has a vibrant, colorful background with a festive theme. It displays the text "Celebrate with Perfect Festival Combos" and "One-stop destination for all your festival essentials - from decor to food and clothing!". A pink button labeled "Browse Combos" is prominently displayed. Below the banner, there's a section titled "Popular Festival Combos" with three items listed: "Diwali Delight Combo", "Holi Happiness Combo", and "Christmas Joy Combo". Each item has a small image, a title, a description, a price (₹1,499, ₹999, or ₹1,299), and an "Add to Cart" button.

**Festival Combo Shop**

Home All Combos Festivals Special Offers Contact Us

## Celebrate with Perfect Festival Combos

One-stop destination for all your festival essentials - from decor to food and clothing!

Browse Combos

### Popular Festival Combos

Discover our carefully curated combo packs for your favorite festivals. Everything you need for a perfect celebration!



**Diwali Delight Combo**

Celebrate the festival of lights with our exclusive combo featuring premium diyas, rangoli colors, and traditional sweets.

₹1,499

Add to Cart



**Holi Happiness Combo**

Experience the joy of colors with our Holi pack featuring organic gulal, water guns, and delicious gujiya sweets.

₹999

Add to Cart



**Christmas Joy Combo**

Spread Christmas cheer with our festive combo including tree decorations, stockings, and traditional plum cake.

₹1,299

Add to Cart

[View All Festival Combos](#)

**Why Choose Our Festival Combos?**

Experience the convenience and joy of our carefully curated festival combinations

**App Manifest**

`manifest.json`

**Errors and warnings**

⚠ Richer PWA Install UI won't be available on desktop. Please add at least one screenshot with the `form_factor` set to wide.

**Identity**

- Name: Festival Combo Shop
- Short name: Festival Combos
- Description: Celebrate with perfect festival combos for Diwali, Holi, Christmas, and more!
- Computed App ID: `http://127.0.0.1:5500/index.html` [Learn more](#)
- Note:** id is not specified in the manifest, start\_url is used instead. To specify an App ID that matches the current identity, set the id field to /index.html.

**Presentation**

- Start URL: `/index.html`
- Theme color: `#ec4899`

**Presentation**

- Start URL: `/index.html`
- Theme color: `#ec4899`
- Background color: `#f9fafb`
- Orientation: `standalone`

**Protocol Handlers**

ⓘ Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.

Need help? Read [URL protocol handler registration for PWAs](#).

**Icons**

Show only the minimum safe area for maskable icons

Need help? Read the [App Image Generator](#).

192x192px  
image/png

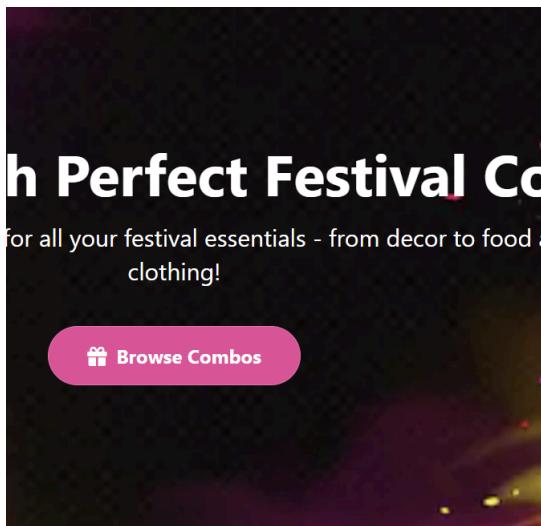
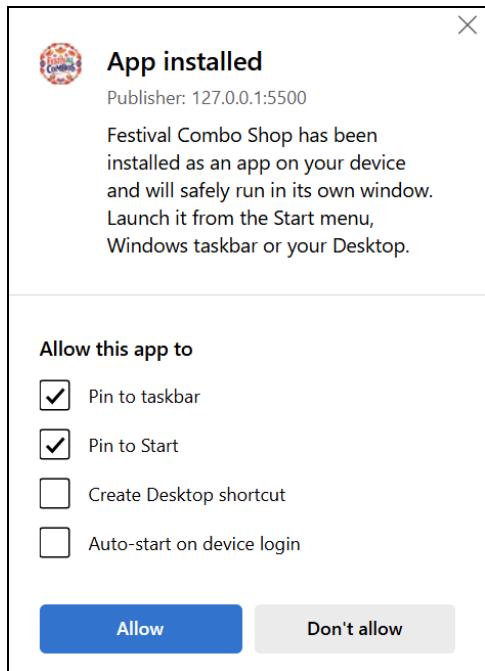
512x512px  
image/png

The screenshot shows the Chrome DevTools Application panel. The left sidebar lists various storage and service worker sections. The main area displays three screenshots under the "Window Controls Overlay" section:

- Screenshot #1:** 1600x1000px, image/webp. Shows a top-down view of a table with various items, possibly a food stall.
- Screenshot #2:** 1280x720px, image/webp. Shows a close-up view of several circular items, possibly food samples.
- Screenshot #3:** 1600x1069px, image/webp. Shows a view of a shop interior with people and goods.

The screenshot shows a Microsoft Edge context menu open over a website for "Festivals". The menu includes options like "New InPrivate window", "Zoom", and "Favorites". The "Apps" option is highlighted.

The screenshot shows an Android application install dialog titled "Install Festival Combo Shop app". It includes the publisher information ("Publisher: 127.0.0.1:5500"), a note about installing the app, and two buttons: "Install" and "Not now".



### Conclusion:

By configuring the manifest.json correctly, the E-commerce web app becomes installable and behaves more like a native mobile application. This improves user engagement and accessibility, especially for mobile users. I faced errors while configuring the icons but it was easily solved by changing the width and height of the images.

## Practical 8

**Name: Sharvari Kishor More**

**Class: D15B**

**Roll No.: 35**

**Aim:**

To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

**Theory:**

A Service Worker is a background script that acts as a network proxy, allowing a web app to intercept network requests and serve cached content. It enhances performance and provides offline support.

In our serviceworker.js file, we implemented the following:

- **Install Event:**  
Caches important assets (index.html, CSS, images, manifest) during the installation of the service worker to make them available offline.
- **Activate Event:**  
Deletes old caches that don't match the current version to avoid storage clutter and ensure the latest files are used.
- **Fetch Event:**  
Intercepts all network requests. If the resource is in the cache, it returns the cached version; otherwise, it fetches it from the network.

This setup ensures our eCommerce PWA works smoothly even when offline and improves performance by loading resources from the cache.

**Code:**

**Serviceworker.js**

```
const CACHE_NAME = 'festival-combo-shop-v1';
const urlsToCache = [
  '/',
  '/index.html',
  '/styles.css',
  '/manifest.json',
  '/serviceworker.js',
  '/color.webp',
  '/diwalisweets.webp',
  '/holicolors.webp',
  '/christmas.webp',
];
```

```
// Install service worker
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        console.log('[ServiceWorker] Caching app shell');
        return cache.addAll(urlsToCache);
      })
  );
});

// Activate service worker
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(keyList => {
      return Promise.all(
        keyList.map(key => {
          if (key !== CACHE_NAME) {
            console.log('[ServiceWorker] Removing old cache', key);
            return caches.delete(key);
          }
        })
      );
    })
  );
  return self.clients.claim();
});

// Fetch content
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        // Return cached version or fetch from network
        return response || fetch(event.request);
      })
  );
});
```

**Index.html**

```
<script>

  if ('serviceWorker' in navigator) {
    console.log('✅ Service workers are supported in this browser.');

    window.addEventListener('load', () => {
      console.log('🌐 Window loaded. Attempting to register service worker...');

      navigator.serviceWorker.register('/serviceworker.js')
        .then(reg => {
          console.log('🎉 Service Worker registered successfully!');
          console.log('🔗 Registration scope:', reg.scope);

          if (reg.installing) {
            console.log('📦 Service Worker is installing...');
          } else if (reg.waiting) {
            console.log('⏸ Service Worker is installed and waiting...');
          } else if (reg.active) {
            console.log('✅ Service Worker is active.');
          }
        })

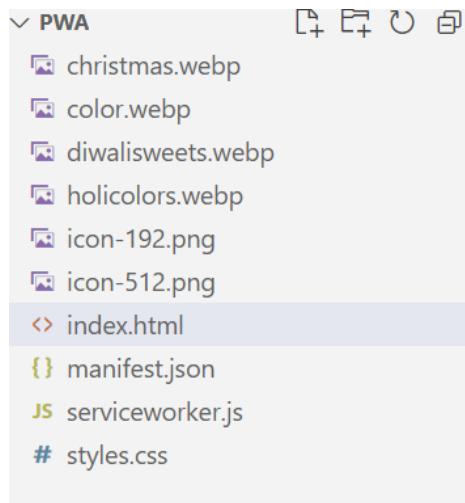
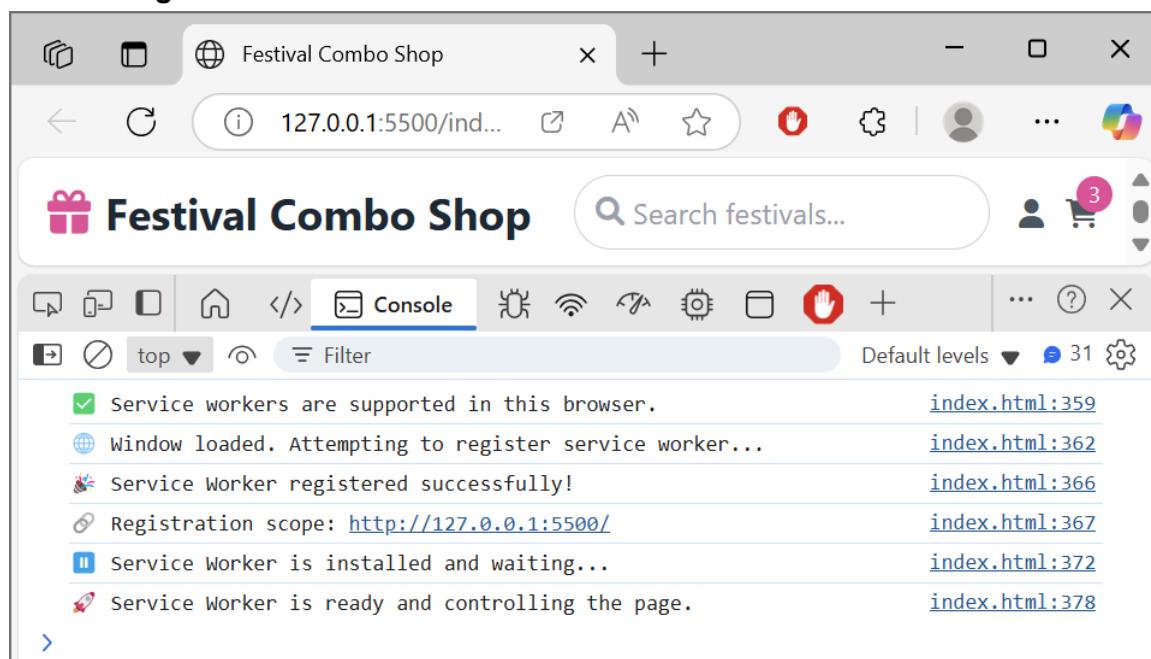
        navigator.serviceWorker.ready.then(() => {
          console.log('🚀 Service Worker is ready and controlling the page.');
        });
    })
    .catch(err => {
      console.error('❗ Service Worker registration failed:', err);
    });
  });

  navigator.serviceWorker.addEventListener('controllerchange', () => {
    console.log('🔄 Controller changed: New service worker is now controlling the page.');
  });

  navigator.serviceWorker.addEventListener('message', (event) => {
    console.log('✉️ Message received from Service Worker:', event.data);
  });

} else {
  console.warn('⚠️ Service workers are not supported in this browser.');
}
```

```
</script>
```

**Folder Structure:****Output:****Console logs**

**Cache Storage**

The screenshot shows the Cache Storage panel in the Chrome DevTools. The left sidebar lists various application components like Manifest, Service workers, and Storage. Under Storage, it shows Local storage, Session storage, Extension storage, Cookies, Private state tokens, Interest groups, Shared storage, and Cache storage. The Cache storage section is currently selected, showing a list of cached files for the domain `http://127.0.0.1:5500`. The table details the following entries:

#	Name	Respon...	Content...	Content...	Time Ca...	Vary He...
0.	/	basic	text/html	24,458	9/4/202...	Origin
1.	/christmas.webp	basic	image/...	245,040	9/4/202...	Origin
2.	/color.webp	basic	image/...	9,120	9/4/202...	Origin
3.	/diwalisweets.webp	basic	image/...	379,906	9/4/202...	Origin
4.	/holicolors.webp	basic	image/...	266,680	9/4/202...	Origin
5.	/index.html	basic	text/html	24,458	9/4/202...	Origin
6.	/manifest.json	basic	application/...	935	9/4/202...	Origin
7.	/serviceworker.js	basic	application/...	1,214	9/4/202...	Origin
8.	/styles.css	basic	text/css	9,531	9/4/202...	Origin

Total entries: 9

No cache entry selected

### Service Worker running in the background.

The screenshot shows the Chrome DevTools Application tab for a PWA named "Festival Combo Shop". The left sidebar lists various storage and background service components. The main panel displays the "Service workers" section for the URL <http://127.0.0.1:5500/>. A service worker named "serviceworker.js" is listed, activated and running (#235). It has clients connected at <http://127.0.0.1:5500/index.html>. The "Update Cycle" section shows three stages: Install, Wait, and Activate, with the Activate step currently in progress.

### Conclusion:

Through this experiment, we successfully implemented a Service Worker in our PWA, enabling offline functionality and improving load performance. It cached essential resources during installation, managed old caches on activation, and served content efficiently through fetch interception. This demonstrates how Service Workers play a crucial role in enhancing user experience in Progressive Web Apps. We faced an error when the cache storage was not being visible, so we had to delete the storage and reload the site.

## Practical 9

**Name: Sharvari Kishor More**

**Class: D15B**

**Roll No.: 35**

**Aim:**

To implement Service worker events like fetch, sync and push for E-commerce PWA.

**Theory:**

Service workers are powerful scripts that run in the background of Progressive Web Apps (PWAs), enabling features like offline access, background sync, and push notifications. In this experiment, we focused on three key service worker events: fetch, sync, and push.

In this experiment, we enhanced our E-commerce Progressive Web App (PWA) by implementing advanced Service Worker events — **fetch**, **sync**, and **push**.

- The **install** and **activate** events handle caching of files and removal of old caches.
- The **fetch** event intercepts network requests and serves cached resources for better performance and offline support.
- The **sync** event simulates background synchronization when the app regains connectivity.
- The **push** event listens for push notifications and displays them, improving user engagement.

These events help in making PWAs more reliable, fast, and interactive even in low or no network conditions.

**Code:**

**Index.html**

<script>

```
if ("serviceWorker" in navigator) {  
    window.addEventListener("load", () => {  
        navigator.serviceWorker  
            .register("/serviceworker.js")  
            .then((registration) => {  
                console.log("✅ Service Worker registered! Scope:", registration.scope);  
  
                // 🎙 Request Push Notification Permission  
                if ("PushManager" in window) {  
                    Notification.requestPermission().then((permission) => {  
                        if (permission === "granted") {
```

```

        console.log("🔔 Push notifications granted.");
    } else {
        console.log("🚫 Push notifications denied.");
    }
});

}

// 🔄 Register Background Sync
if ("SyncManager" in window) {
    navigator.serviceWorker.ready.then((swReg) => {
        swReg.sync.register("sync-data").then(() => {
            console.log("🔄 Sync registered");
        }).catch((err) => {
            console.log("❌ Sync registration failed:", err);
        });
    });
}

.catch((error) => {
    console.log("❌ Service Worker registration failed:", error);
});
};

}
</script>

```

## Serviceworker.js

```

const CACHE_NAME = 'festival-combo-shop-v1';
const FILES_TO_CACHE = [
    '/',
    '/index.html',
    '/styles.css',
    '/manifest.json',
    '/serviceworker.js',
    '/color.webp',
    '/diwalisweets.webp',
    '/holicolors.webp',
    '/christmas.webp',
];

```

```
// Install Event
self.addEventListener("install", (event) => {
    console.log("[ServiceWorker] Install");
    event.waitUntil(
        caches.open(CACHE_NAME).then((cache) => {
            console.log("[ServiceWorker] Caching files");
            return cache.addAll(FILES_TO_CACHE);
        })
    );
});

// Activate Event
self.addEventListener("activate", (event) => {
    console.log("[ServiceWorker] Activate");
    event.waitUntil(
        caches.keys().then((keyList) =>
            Promise.all(
                keyList.map((key) => {
                    if (key !== CACHE_NAME) {
                        console.log("[ServiceWorker] Removing old cache", key);
                        return caches.delete(key);
                    }
                })
            )
        )
    );
    return self.clients.claim();
});

// Enhanced Fetch Event
self.addEventListener("fetch", (event) => {
    console.log("[ServiceWorker] Fetch", event.request.url);
    const requestURL = new URL(event.request.url);

    // If request is same-origin, use Cache First
    if (requestURL.origin === location.origin) {
        event.respondWith(
            caches.match(event.request).then((cachedResponse) => {
```

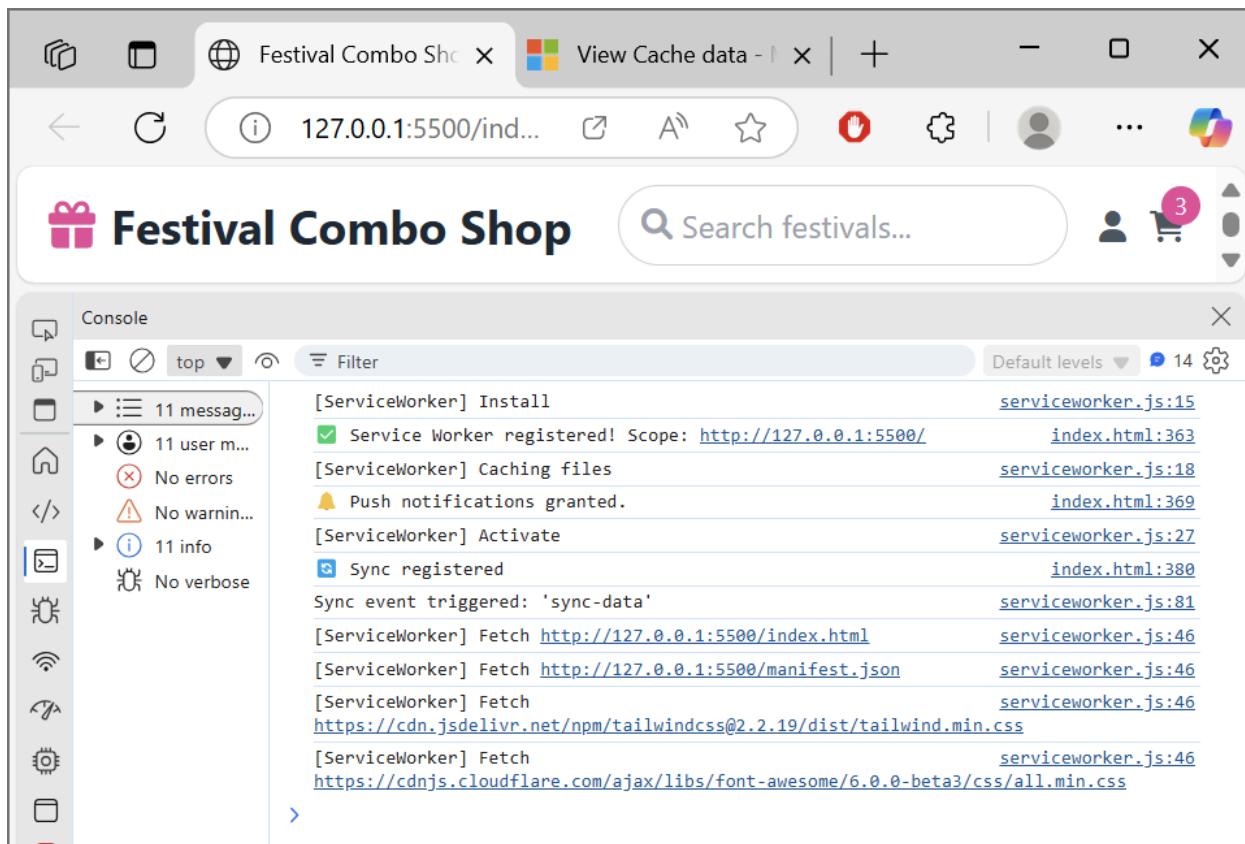
```
return ( cachedResponse ||
    fetch(event.request).catch(() => caches.match("offline.html"))
);
})
);
}
} else {
// Else, use Network First
event.respondWith(
fetch(event.request)
.then((response) => {
    return response;
}))
.catch(() =>
caches.match(event.request).then((res) => {
    return res || caches.match("offline.html");
}))
)
);
}
);
}

// Sync Event (simulation)
self.addEventListener("sync", (event) => {
if (event.tag === "sync-data") {
    event.waitUntil(
(async () => {
    console.log("Sync event triggered: 'sync-data'");
    // Here you can sync data with server when online
})()
);
}
});
}

// Push Event
self.addEventListener("push", function (event) {
if (event && event.data) {
let data = {};
try {
```

```
data = event.data.json();
} catch (e) {
  data = {
    method: "pushMessage",
    message: event.data.text(),
  };
}

if (data.method === "pushMessage") {
  console.log("Push notification sent");
  event.waitUntil(
    self.registration.showNotification("Maharashtrian Handloom", {
      body: data.message,
    })
  );
}
});
```

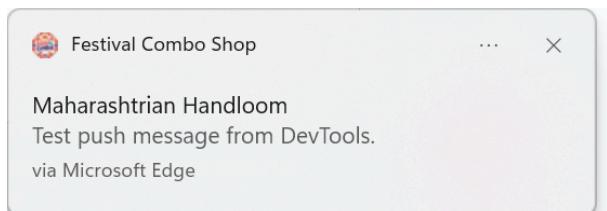
**Output:****Working of service worker**

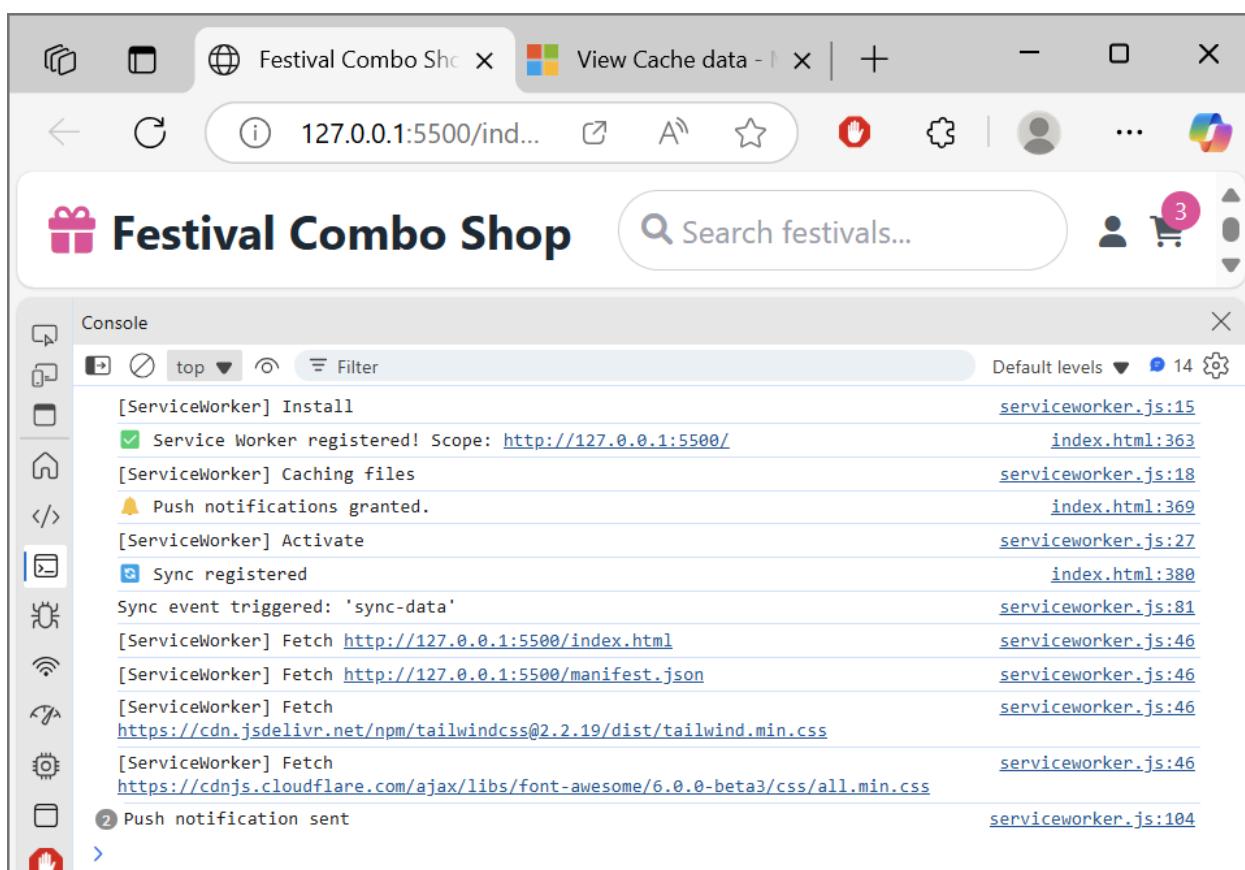
**Cache**

The screenshot shows the Microsoft Edge DevTools Cache panel. The left sidebar lists various storage types: Application, Storage, Background services, and others. Under Application, 'festival-comb...' is selected, revealing its cache entries. The main pane displays the details for the origin `http://127.0.0.1:5500`, including the bucket name (default), persistence status (No), durability (relaxed), and quota (0 B). A table lists 8 cache entries:

| # | Name               | Respon... | Content...      | Content... | Time C... | Vary He... |
|---|--------------------|-----------|-----------------|------------|-----------|------------|
| 0 | /                  | basic     | text/html       | 24,155     | 9/4/20... | Origin     |
| 1 | /christmas.webp    | basic     | image/...       | 245,040    | 9/4/20... | Origin     |
| 2 | /color.webp        | basic     | image/...       | 9,120      | 9/4/20... | Origin     |
| 3 | /diwalisweets.webp | basic     | image/...       | 379,906    | 9/4/20... | Origin     |
| 4 | /holicolors.webp   | basic     | image/...       | 266,680    | 9/4/20... | Origin     |
| 5 | /index.html        | basic     | text/html       | 24,155     | 9/4/20... | Origin     |
| 6 | /manifest.json     | basic     | application/... | 935        | 9/4/20... | Origin     |
| 7 | /serviceworker.js  | basic     | application/... | 2,897      | 9/4/20... | Origin     |
| 8 | /styles.css        | basic     | text/css        | 9,531      | 9/4/20... | Origin     |

Total entries: 9

**Sending of push notification.**

**Test: Push Notification sent****Conclusion:**

By implementing fetch, sync, and push events in the Service Worker, we made our E-commerce PWA capable of working offline, syncing data in the background, and sending push notifications. This ensures a better user experience and highlights the power of PWAs in building modern web applications.

## Practical 10

**Name: Sharvari Kishor More**

**Class: D15B**

**Roll No.: 35**

**Aim:**

To study and implement deployment of Ecommerce PWA to GitHub Pages.

**Theory:**

GitHub Pages is a free hosting service provided by GitHub to publish static websites directly from a GitHub repository. It supports HTML, CSS, JavaScript, and other front-end files, making it ideal for hosting PWAs.

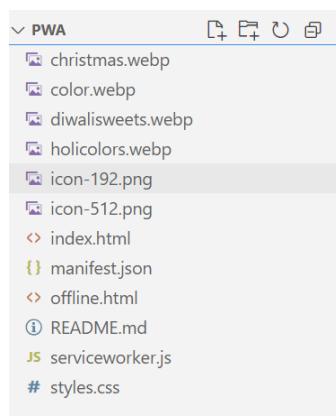
**What we implemented in our code**

In this experiment, we deployed our E-commerce PWA project, "Eco Friendly Gadgets Collection", to GitHub Pages using these steps:

- We ensured all necessary PWA files were in place, such as index.html, manifest.json, serviceworker.js, and offline assets.
- The index.html file used relative paths correctly so resources load properly after deployment.
- We registered the service worker to enable offline access and caching of essential files.
- The manifest.json provided metadata to support "Add to Home Screen" functionality.
- We created a GitHub repository and pushed all project files to it.
- Then, from GitHub Settings → Pages, we selected the main branch and the / (root) folder as the source.
- After saving, GitHub generated a live link through which we accessed the deployed PWA.

This deployment made our PWA fully functional and accessible online with service worker support, offline capabilities, and installability features.

**Folder Structure:**



**Screenshots:****Creation of new repository in public settings.**

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \* Repository name \*

Sharvari-21 / Festival-Combo

Festival-Combo is available.

Great repository names are short and memorable. Need inspiration? How about [silver-chainsaw](#) ?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

github.com/Sharvari-21/Festival-Combo

Sharvari-21 / Festival-Combo

Festival-Combo Public

Set up GitHub Copilot

Add collaborators to this repository

Quick setup — if you've done this kind of thing before

...or create a new repository on the command line

```
C:\Users\sharv\OneDrive\Desktop\Sem6\PWA>git remote add origin https://github.com/Sharvari-21/Festival-Combo.git
C:\Users\sharv\OneDrive\Desktop\Sem6\PWA>git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 236 bytes | 118.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Sharvari-21/Festival-Combo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

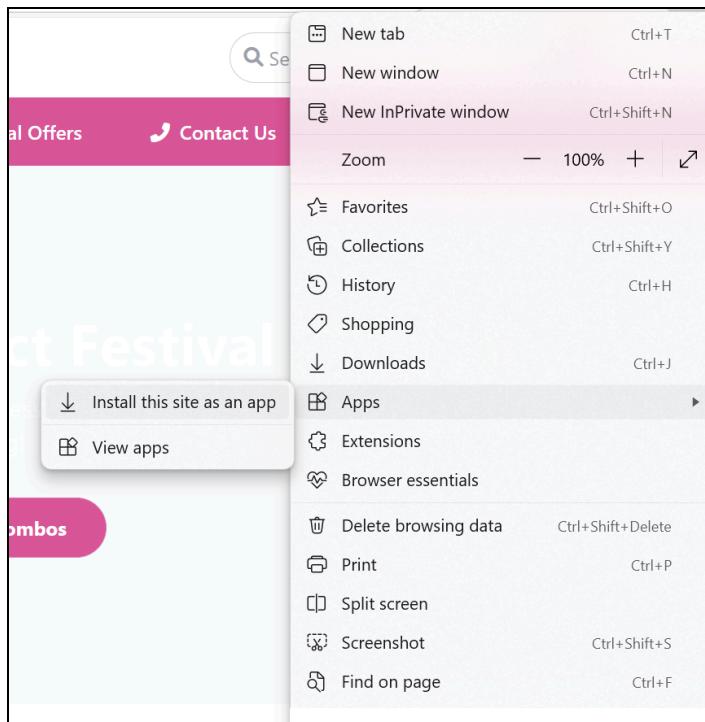
The terminal window shows the command `git push origin main` being run, resulting in 11 files changed, 1022 insertions(+), and various file names like `christmas.webp`, `color.webp`, etc.

The GitHub repository page for `Festival-Combo` shows the following details:

- Code** tab selected.
- Commits**: 2 Commits by Sharvari-21 Initial (fb2ea31 · now).
- Files** listed: README.md, christmas.webp, color.webp, diwalisweets.webp, holicolors.webp, icon-192.png, icon-512.png.
- About** section: No description, website, or topics provided.
- Activity**: 0 stars, 1 watching, 0 forks.
- Releases**: No releases published. Create a new release.
- Packages**: None.

The GitHub Pages settings page shows:

- General** tab selected.
- Access**, **Collaborators**, **Moderation options** sections.
- Code and automation** section: Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces.
- Pages** tab selected.
- GitHub Pages** status: Your site is live at <https://sharvari-21.github.io/Festival-Combo/>. Last deployed by Sharvari-21 1 minute ago. Buttons: Visit site, ...
- Build and deployment** section: Deploy from a branch (set to main), Branch (set to main branch), Source (set to / (root)).
- Security** section: Your site was last deployed to the `github-pages` environment by the `pages build and deployment` workflow.



### Install this site as an app

**F**

Festival Combo Shop

[Edit](#)

This site can be installed as an application. It will open in its own window and safely integrate with Windows features.

[Install](#)

[Not now](#)

**The system is available for downloading from the link**

**Conclusion:**

In this experiment, we deployed the website PWA on the internet with the help of git pages. We took proper care of manifest.json file and serviceworker.js file. A link was generated by Git Pages. It took a lot of time for the link to be generated, but after the link creation, the website was visible and working.

## Practical 11

**Name: Sharvari Kishor More**

**Class: D15B**

**Roll No.: 35**

**Aim:**

To use google Lighthouse PWA Analysis Tool to test the PWA functioning

**Theory:**

Lighthouse is a tool provided by Google that helps us check how well our web app performs as a Progressive Web App (PWA). It analyzes important factors like performance, accessibility, SEO, best practices, and PWA features.

In this experiment, we used Lighthouse in Chrome DevTools to test our deployed PWA. Lighthouse runs a series of audits and gives a score out of 100 for each category. These scores show how well our app performs and where we can improve.

**What we implemented:**

- We deployed our PWA Festival-Combo on GitHub Pages.
- Then, we ran Lighthouse from Chrome DevTools on the live site.
- Lighthouse gave us excellent scores:
  - Performance: 92
  - Accessibility: 90
  - Best Practices: 89
  - SEO: 90

These high scores show that our app is fast, user-friendly, and optimized for search engines. Some minor suggestions were also given, like improving color contrast and adding a meta description, which can help make the app even better.

**Code:**

**Manifest.json**

```
{  
  "name": "Festival Combo Shop",  
  "short_name": "Festival Combos",  
  "start_url": "./index.html",  
  "display": "standalone",  
  "background_color": "#f9fafb",  
  "theme_color": "#ec4899",  
  "description": "Celebrate with perfect festival combos for Diwali, Holi, Christmas, and more!",  
  "icons": [  
    {
```

```
"src": "/icon-192.png",
"sizes": "192x192",
"type": "image/png",
"purpose": "any-maskable"
},
{
"src": "/icon-512.png",
"sizes": "512x512",
"type": "image/png",
"purpose": "any-maskable"
}
],
"screenshots": [
{
"src": "/diwalisweets.webp",
"type": "image/webp",
"sizes": "1600x1000"
},
{
"src": "/holicolors.webp",
"type": "image/webp",
"sizes": "1280x720"
},
{
"src": "/christmas.webp",
"type": "image/webp",
"sizes": "1600x1069"
}
]
}
```

### serverworker.js

```
const CACHE_NAME = 'festival-combo-shop-v1';
const FILES_TO_CACHE = [
  '/Festival-Combo/',
  '/Festival-Combo/index.html',
  '/Festival-Combo/styles.css',
  '/Festival-Combo/manifest.json',
  '/Festival-Combo/serviceworker.js',
```

```
'/Festival-Combo/color.webp',
'/Festival-Combo/diwalisweets.webp',
'/Festival-Combo/holicolors.webp',
'/Festival-Combo/christmas.webp',
];

// Install Event
self.addEventListener("install", (event) => {
  console.log("[ServiceWorker] Install");
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log("[ServiceWorker] Caching files");
      return cache.addAll(FILES_TO_CACHE);
    })
  );
});

// Activate Event
self.addEventListener("activate", (event) => {
  console.log("[ServiceWorker] Activate");
  event.waitUntil(
    caches.keys().then((keyList) =>
      Promise.all(
        keyList.map((key) => {
          if (key !== CACHE_NAME) {
            console.log("[ServiceWorker] Removing old cache", key);
            return caches.delete(key);
          }
        })
      )
    )
  );
  return self.clients.claim();
});

// Enhanced Fetch Event
self.addEventListener("fetch", (event) => {
  console.log("[ServiceWorker] Fetch", event.request.url);
  const requestURL = new URL(event.request.url);
```

```
// If request is same-origin, use Cache First
if (requestURL.origin === location.origin) {
  event.respondWith(
    caches.match(event.request).then((cachedResponse) => {
      return ( cachedResponse ||
        fetch(event.request).catch(() => caches.match("offline.html")))
    );
  })
);
}

} else {
  // Else, use Network First
  event.respondWith(
    fetch(event.request)
      .then((response) => {
        return response;
      })
      .catch(() =>
        caches.match(event.request).then((res) => {
          return res || caches.match("offline.html");
        })
      )
    );
}
});

// Sync Event (simulation)
self.addEventListener("sync", (event) => {
  if (event.tag === "sync-data") {
    event.waitUntil(
      (async () => {
        console.log("Sync event triggered: 'sync-data'");
        // Here you can sync data with server when online
      })()
    );
  }
});

// Push Event
```

```
self.addEventListener("push", function (event) {
  if (event && event.data) {
    let data = {};
    try {
      data = event.data.json();
    } catch (e) {
      data = {
        method: "pushMessage",
        message: event.data.text(),
      };
    }
  }

  if (data.method === "pushMessage") {
    console.log("Push notification sent");
    event.waitUntil(
      self.registration.showNotification("Maharashtrian Handloom", {
        body: data.message,
      })
    );
  }
});
```

The screenshot shows two views of the Festival Combo PWA. The top view is the developer console in Chrome, displaying network requests and errors. One error is visible: 'Failed to load resource: the server responded with a status of 404 ()'. Another error is 'Manifest: found icon with no valid purpose; ignoring it.' The bottom view is the Lighthouse report interface, which includes settings for mode (Navigation), device (Mobile), and categories (Performance, Accessibility, Best practices, SEO). A button to 'Generate a Lighthouse report' is present.

### Conclusion:

In this experiment, we tested the Festival Combo PWA using Google Lighthouse and achieved high scores in performance, accessibility, best practices, and SEO. Initially, we faced an error where all images weren't visible, which was resolved by adding the repository name into the serviceworker.js file before the image file.