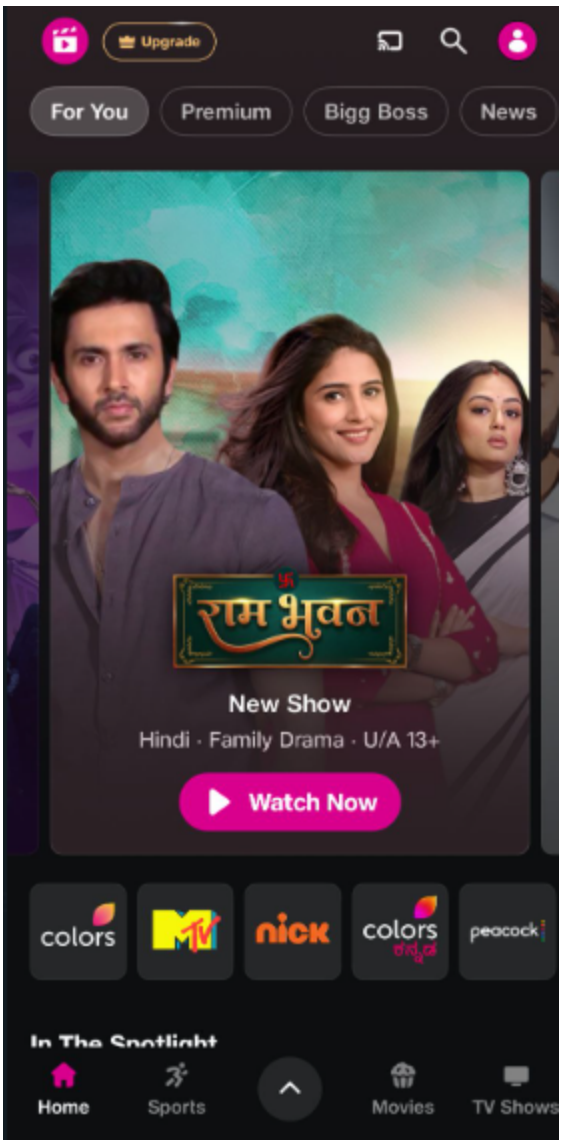
MAD and PWA LAB

Name: Mahi Jodhani
Class: D15A
Roll No: 21

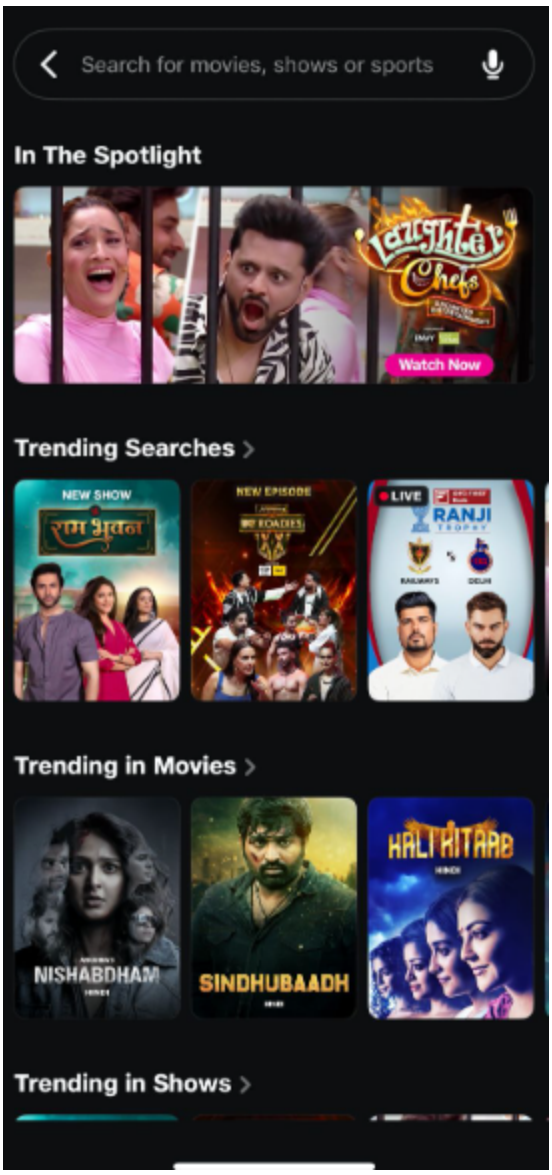
Aim: Selecting features for application development, the features should comprise of:

1. Common widgets
2. Should include icons, images, charts, etc.
3. Should have interactive forms
4. Should apply navigation, routing, and gestures
5. Should connect with Firebase database

ScreenShot	Features
	**Landing Page:** 1. Sign in using email address or Google account 2. Firebase for authentication 3. Firebase: Store user details like username, email, and password 4. Password recovery and validation 5. Smooth navigation to Home Page after login

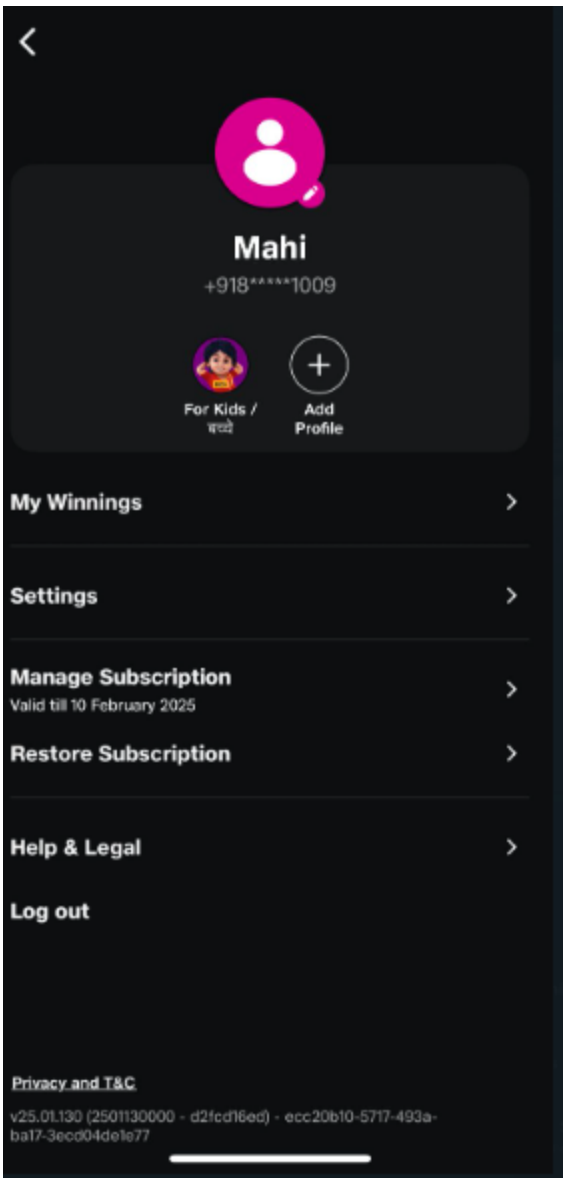


Home Page:
1. Display banners for latest shows/movies
2. Interactive UI with search bar
3. Navigation drawer for easy access
4. Integration with Firebase for content management
5. User-specific recommendations using Firebase
6. Quick access to Sports and Profile pages

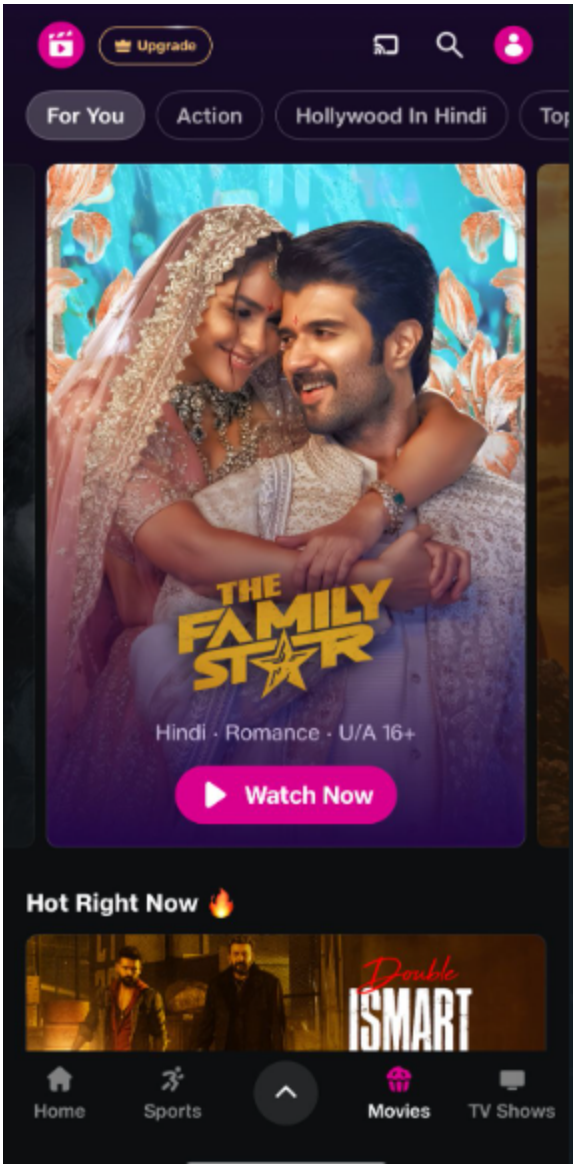


Search Page:

1. Search for movies, shows, and sports events
2. Real-time search suggestions powered by Firebase
3. Filter results by categories and genres
4. Recent search history feature
5. Quick navigation to detailed content pages



Profile Page:
1. Display user details like name, email, and profile picture
2. Edit profile information with real-time Firebase updates
3. View watch history and personalized recommendations
4. Logout functionality with session management
5. Secure data handling using Firebase Authentication



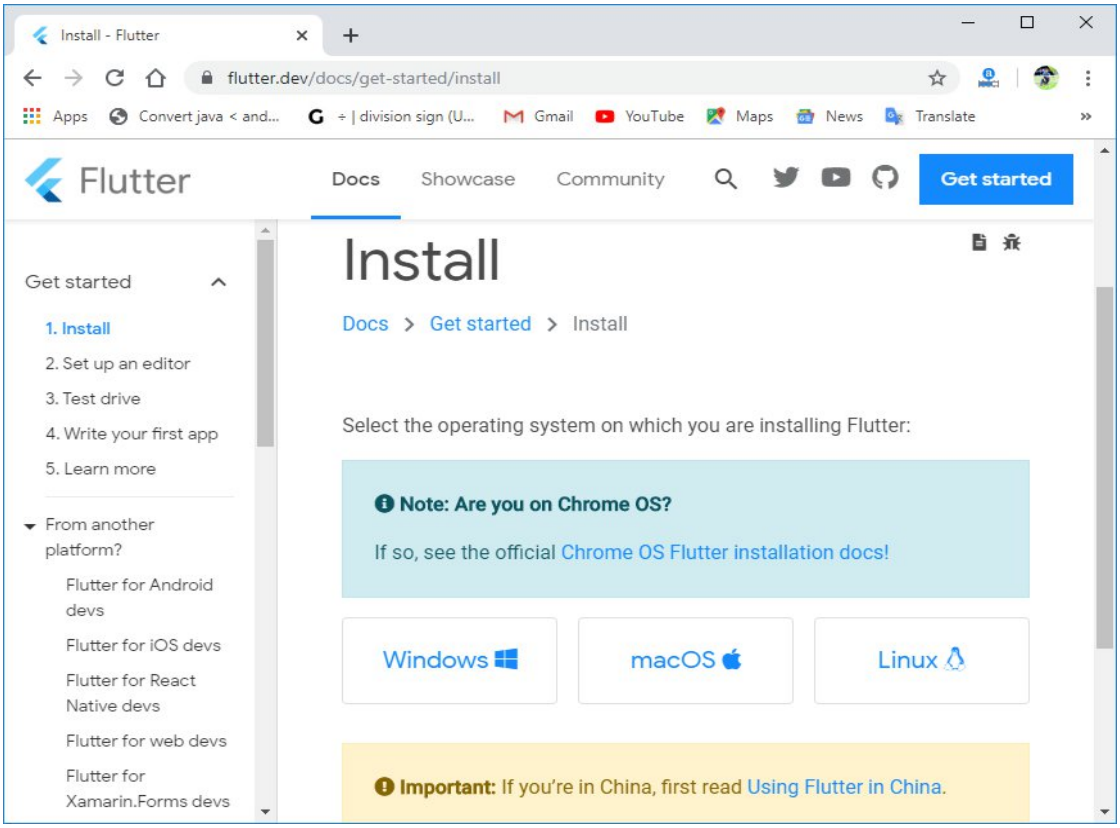
Movie Details Page:
1. Show movie title, description, and ratings
2. Include images, trailers, and cast information
3. Add to watchlist feature using Firebase Firestore
4. Navigation for related content
5. User reviews and comments powered by Firebase
6. Option to share movie details with friends

Name: Mahi Jodhani Roll No: 21 Div:D15A

EXP 1: 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.

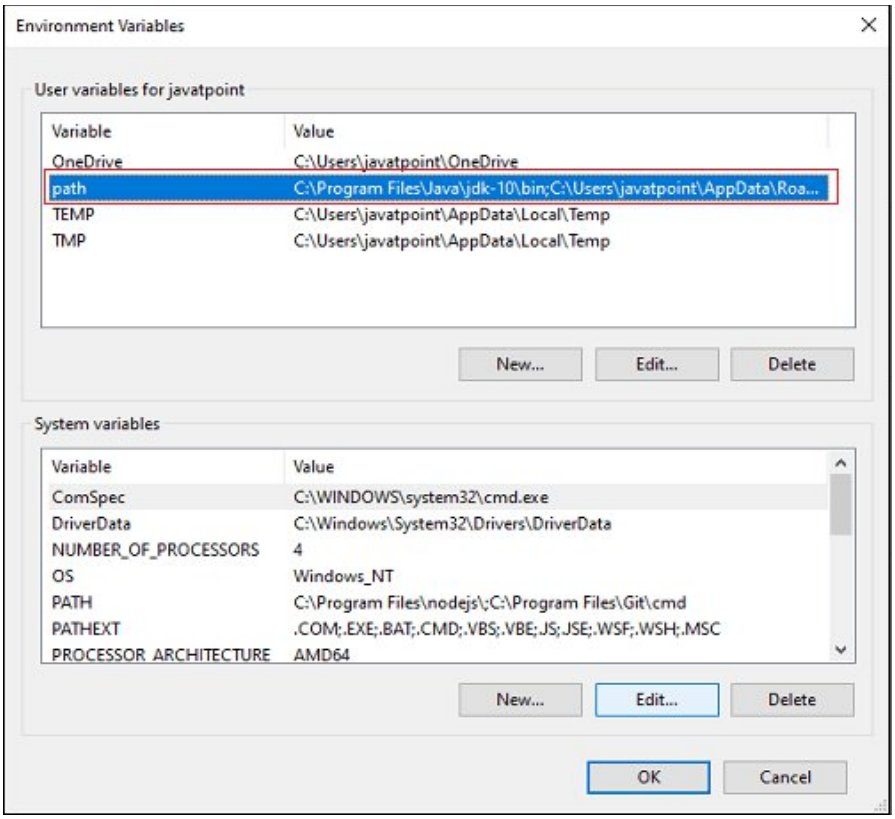


Step 2: Next, to download the latest Flutter SDK, click on the Windows **icon**. Here, you will find the download link for SDK.

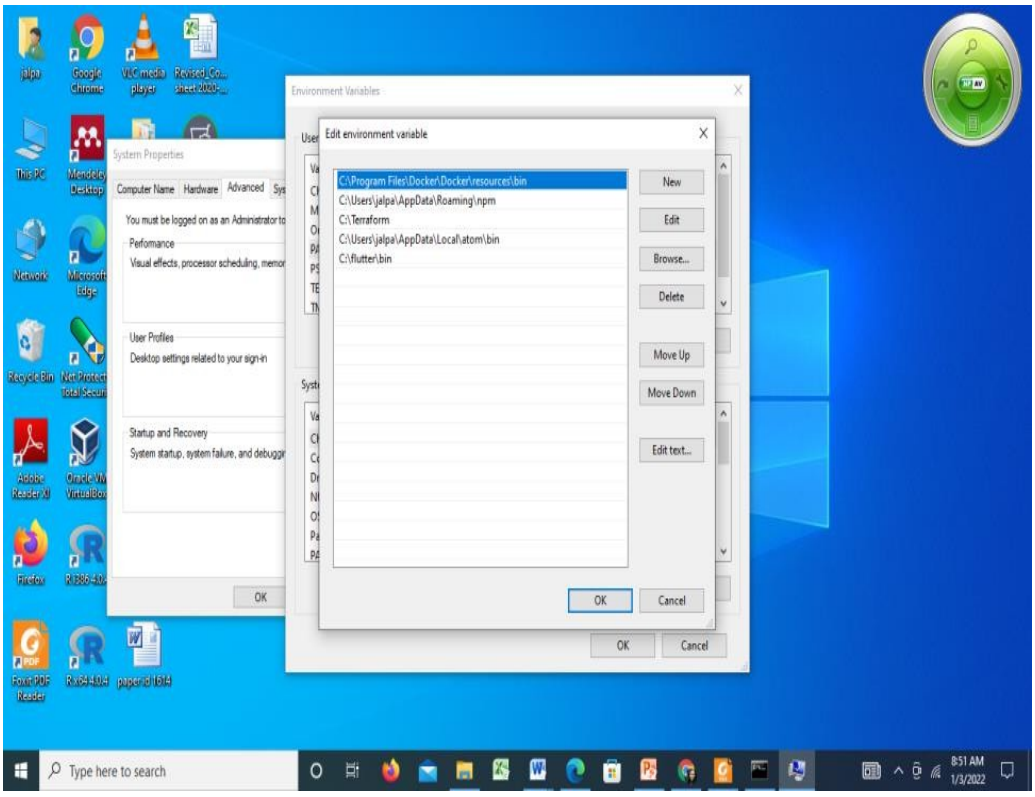
Step 3: When your download is complete, extract the **zip** file and place it in the desired installation folder or location, for example, C: /Flutter.

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.

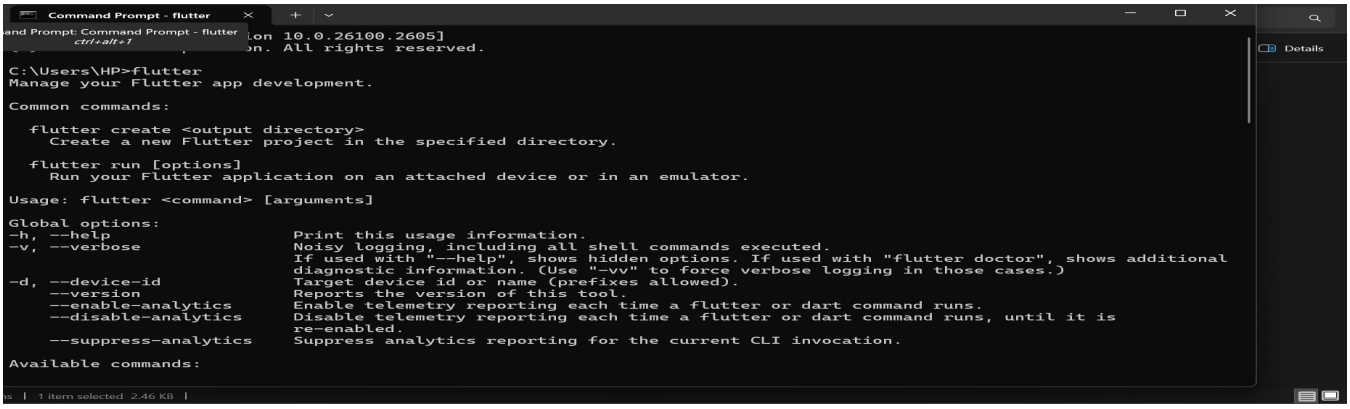


Step 4.2: Now, select path -> click on edit. The following screen appears

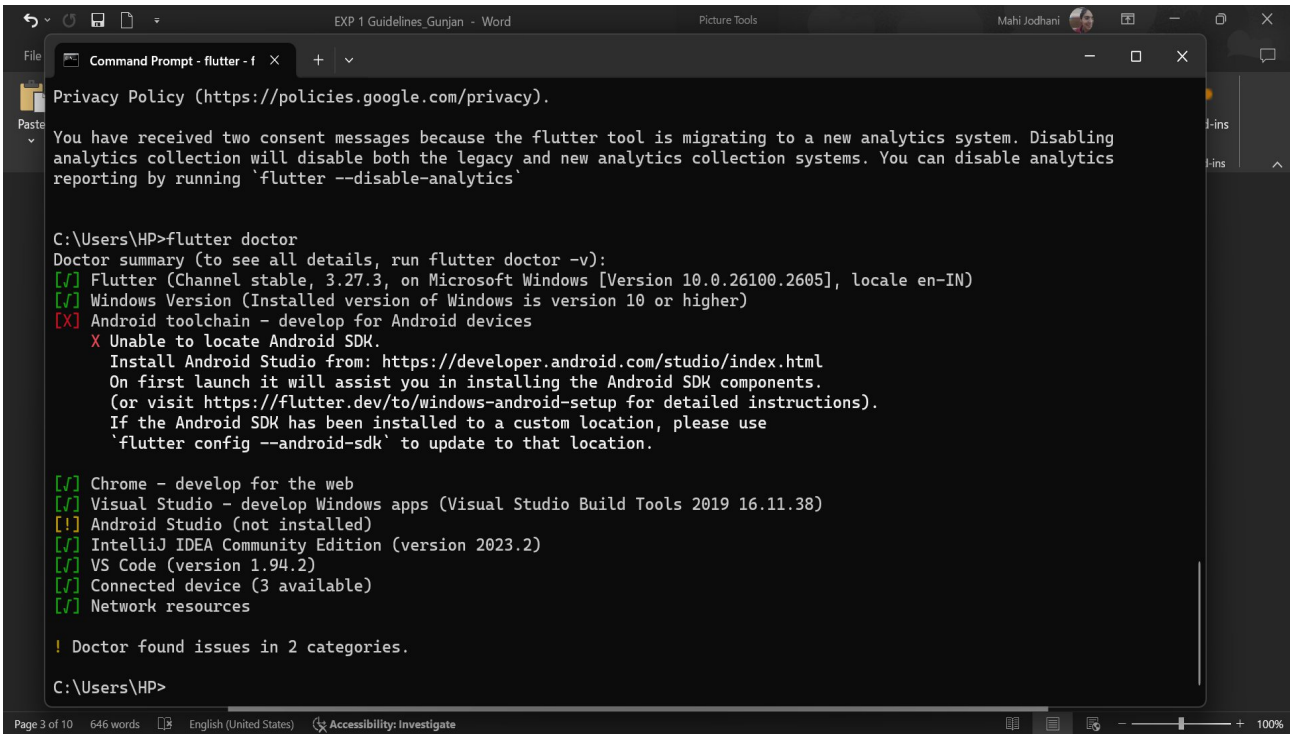


Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value -> ok -> ok -> ok.

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



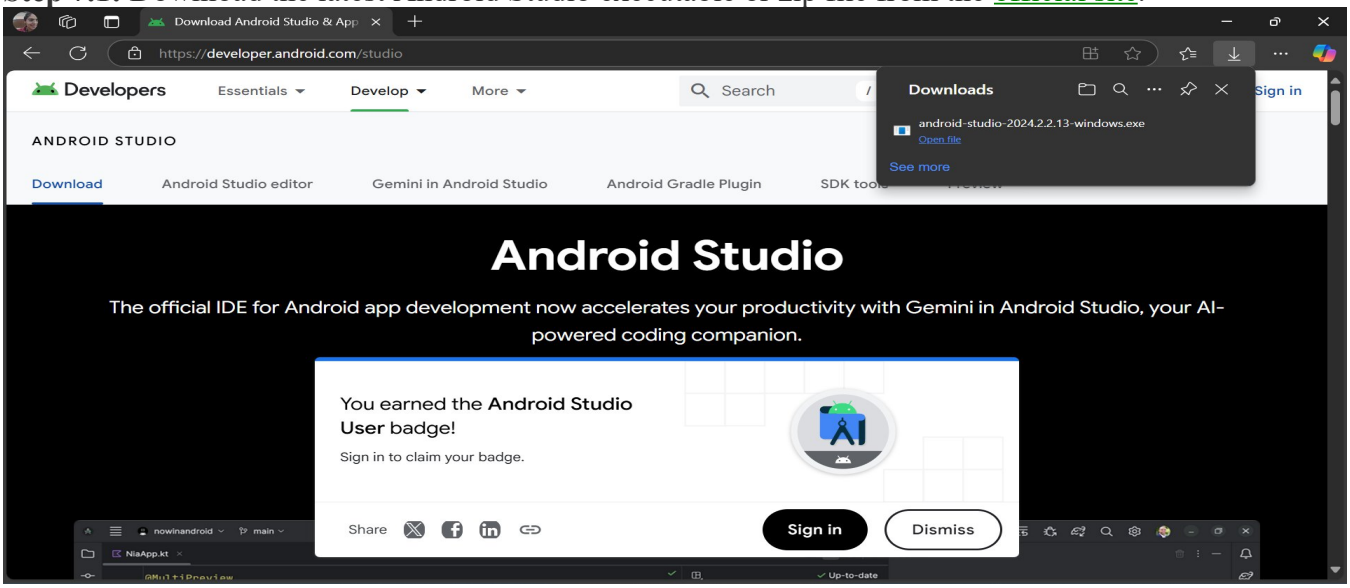
Now, run the $ **flutter doctor** command. This command checks for all the requirements of Flutter app development and displays a report of the status o f your Flutter installation.

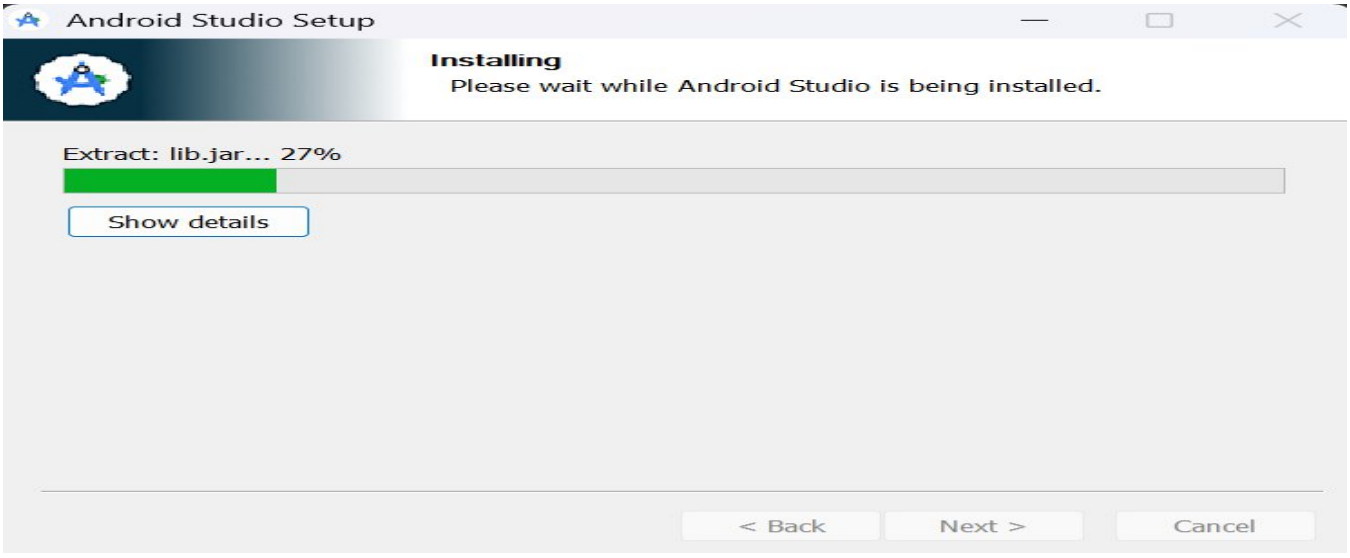


Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

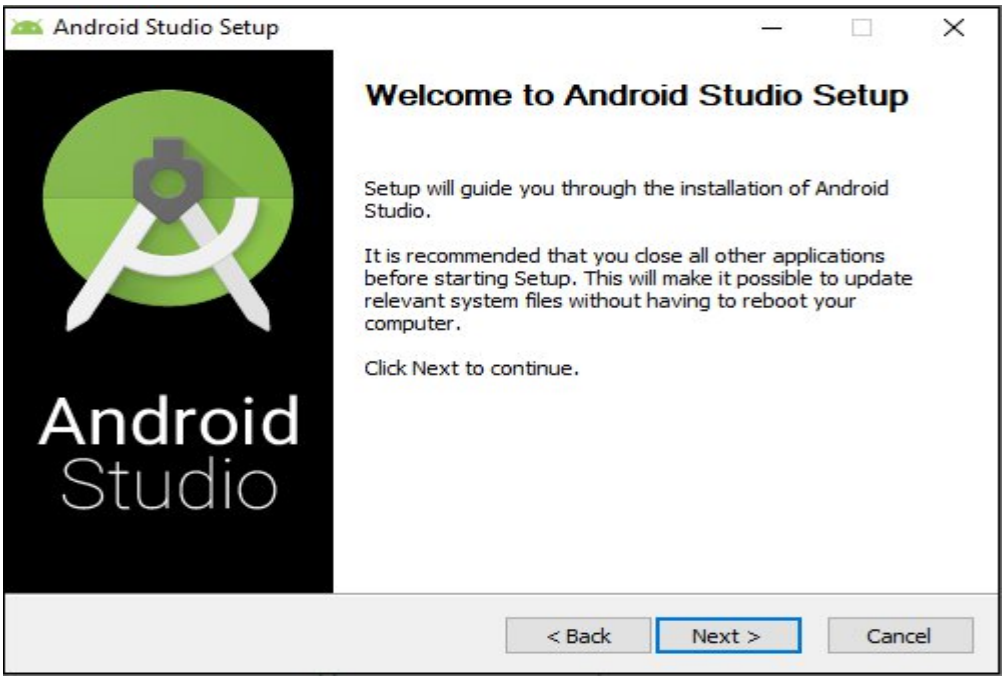
Step 7: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

Step 7.1: Download the latest Android Studio executable or zip file from the official site.

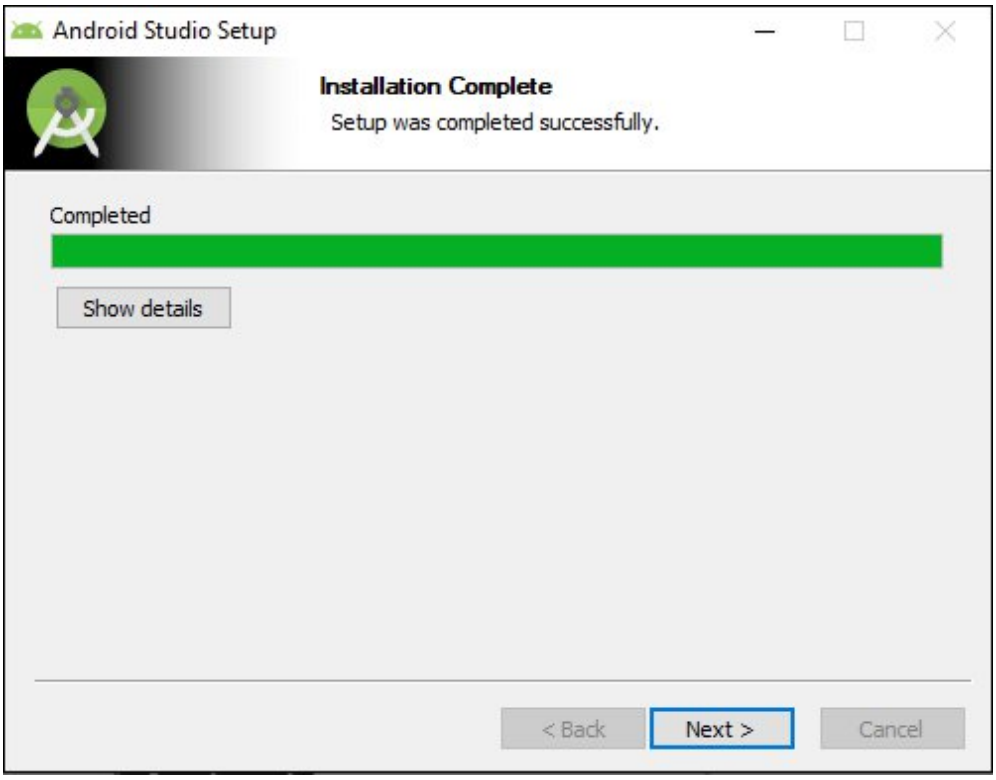




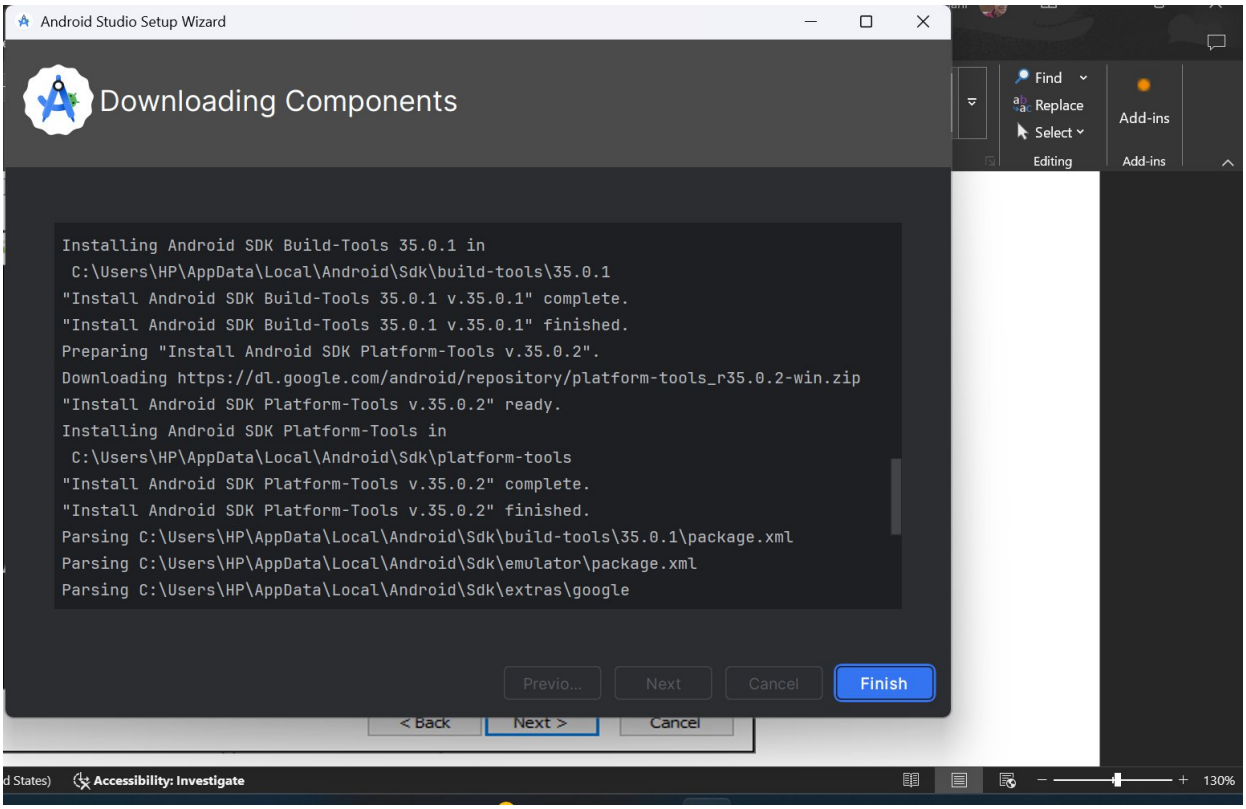
Step 7.2: When the download is complete, open the **.exe** file and run it. You will get the following dialog box .



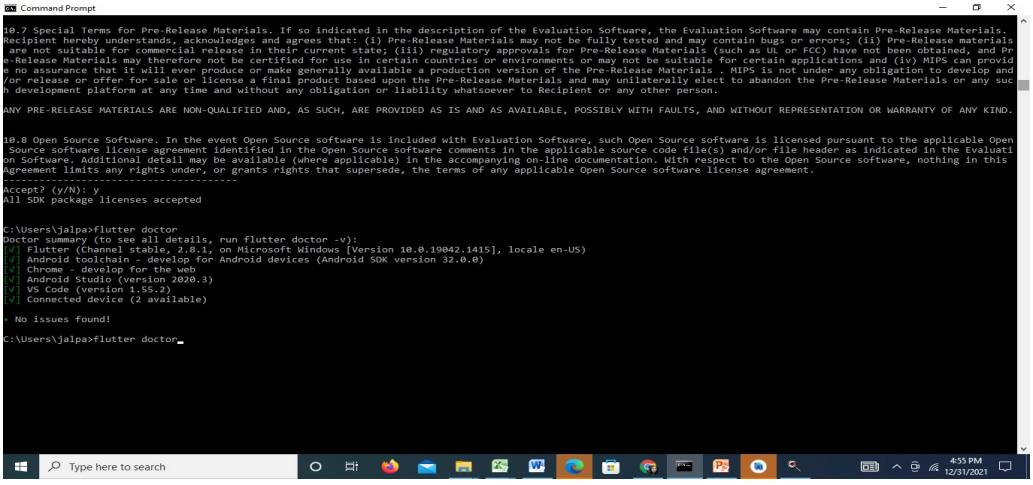
Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

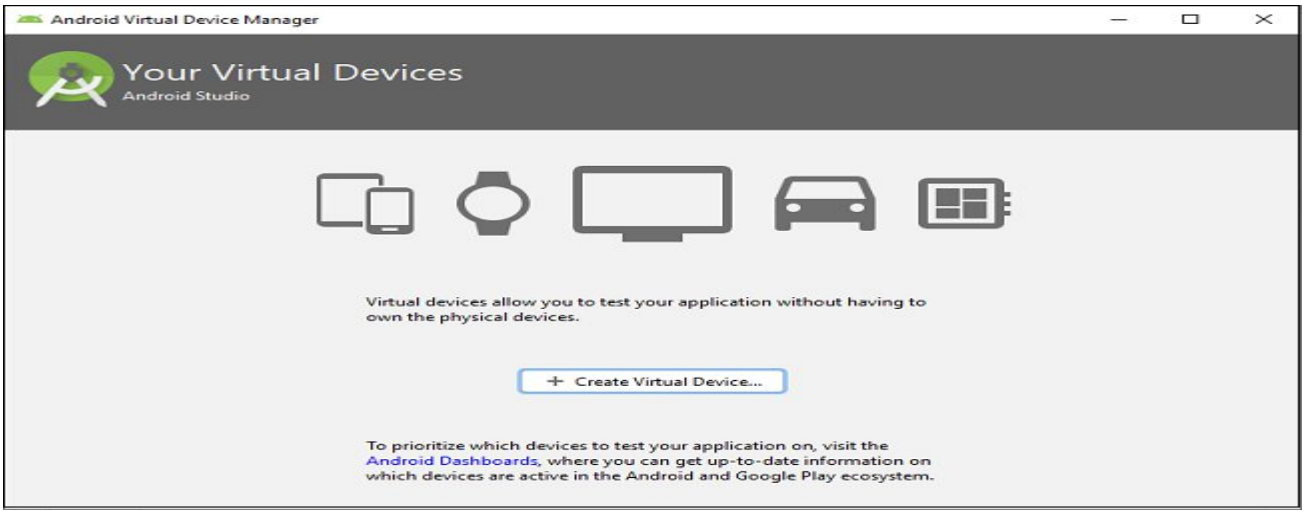


Step 7.5: run the $ **flutter doctor** command and Run flutter doctor --android-licenses command.



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.

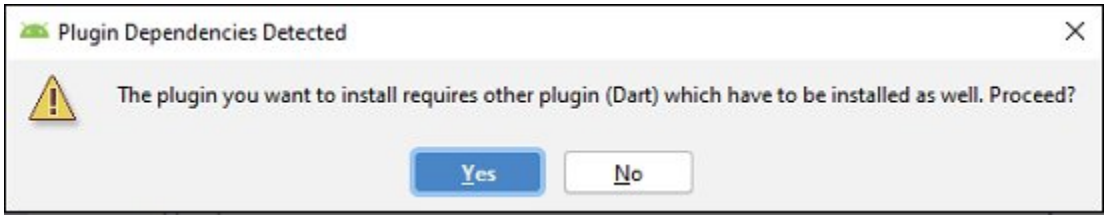
Step 8.3: Select the system image for the latest Android version and click on Next.

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.

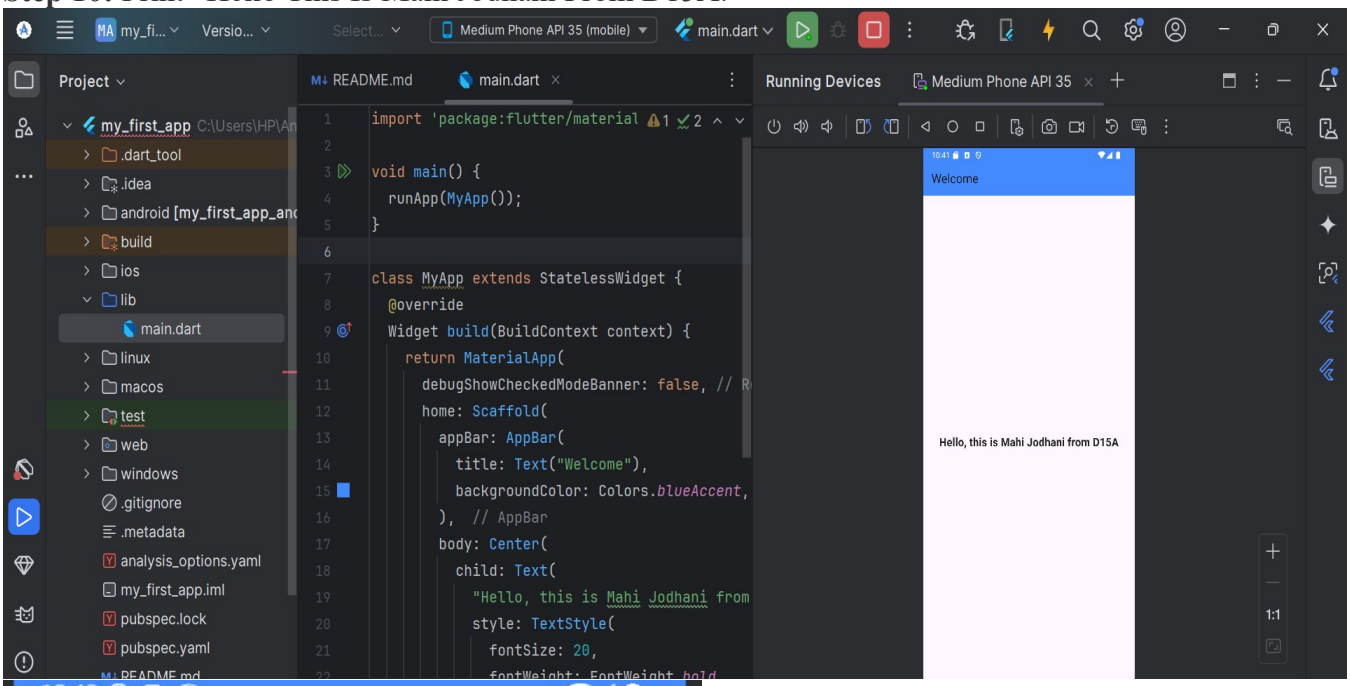
Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.
Step 9.1: 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.

Step 10: Print “Hello This Is Mahi Jodhani From D15A.”



EXPERIMENT NO:2

Name:Mahi Jodhani
Roll no: 21
D15A

Aim: To design Flutter UI by including common widgets.

Theory:

1. Introduction

Flutter is an open-source UI development framework created by Google that enables developers to build cross-platform applications using a single codebase. It follows a widget-based architecture, where every UI element is represented as a widget. Widgets in Flutter are categorized into two types: StatelessWidgets (which do not change dynamically) and StatefulWidgets (which maintain dynamic states).

2. Objective

- Understand the role of widgets in Flutter UI development.
- Explore different types of common widgets and their functionalities.
- Learn how to structure a UI effectively using widgets.

3. Importance of Widgets in Flutter

- Composable – Multiple widgets can be combined to build a complete UI.
- Reusable – The same widget can be used in different parts of the application.
- Customizable – Widgets offer extensive customization options to achieve the desired look and feel.
- Responsive – Widgets adapt to different screen sizes, making the UI flexible across devices.

4. Commonly Used Widgets in Flutter

Flutter provides a wide range of widgets to design modern UIs.

1. Structural Widgets – Define the layout and structure of the UI (e.g., Container, Column, Row).
2. Interactive Widgets – Handle user input and interaction (e.g., Button, TextField, GestureDetector).
3. Styling Widgets – Enhance the visual appearance (e.g., Padding, Align, Card).
4. Scrolling Widgets – Enable scrolling functionality (e.g., ListView, GridView).

5. Implementation of UI in Flutter

Designing a UI in Flutter involves:

1. Defining the widget tree – A hierarchical arrangement of widgets that form the UI structure.
2. Using layout widgets – Arranging elements using Column, Row, Stack, and other layout-based widgets.
3. Adding interactivity – Incorporating buttons, text fields, and gesture detectors for user interaction.

4. Applying styling and theming – Customizing widgets with colors, padding, borders, and shadows to enhance aesthetics.

Code:

Main.Dart

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:netflix_clone/application/downloads/downloads_bloc.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/domain/core/di/injectable.dart';
import 'package:netflix_clone/presentation/main_page/widgets/screen_main_page.dart';
import 'package:netflix_clone/presentation/splash/screen_splash.dart';

import 'application/description/description_bloc.dart';
import 'application/fast_laugh/fast_laugh_bloc.dart';
import 'application/home/home_bloc.dart';
import 'application/hot_and_new/hot_and_new_bloc.dart';
import 'application/search/search_bloc.dart';
import 'presentation/onStartPage/scrren_onboarding.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await configureInjection();
  runApp(const MyApp());
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MultiBlocProvider(
      providers: [
        BlocProvider(
          create: (ctx) => getIt<DownloadsBloc>(),
        ),
        BlocProvider(
          create: (ctx) => getIt<SearchBloc>(),
        ),
        BlocProvider(
          create: (ctx) => getIt<FastLaughBloc>(),
        ),
```

```
        BlocProvider(
          create: (ctx) => getIt<HotAndNewBloc>(),
        ),
        BlocProvider(
          create: (ctx) => getIt<HomeBloc>(),
        ),
        BlocProvider(
          create: (ctx) => getIt<DescriptionBloc>(),
        )
      ],
      child: MaterialApp(
        title: 'Flutter Demo',
        theme: ThemeData(
            appBarTheme: const AppBarTheme(backgroundColor: Colors.black),
            primarySwatch: Colors.blue,
            backgroundColor: Colors.black,
            scaffoldBackgroundColor: backgrounndColor,
            fontFamily: GoogleFonts.montserrat().fontFamily,
            textTheme: const TextTheme(
                bodyText1: TextStyle(color: Colors.white),
                bodyText2: TextStyle(color: Colors.white))),
        home:  const ScreenSplash() ,
      ),
    );
  }
}
```

Screen_home.dart:

```
import 'dart:developer';
import 'dart:ffi';

import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:netflix_clone/application/home/home_bloc.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/core/constants.dart';
import 'package:netflix_clone/presentation/home/widget/background_card.dart';
import 'package:netflix_clone/presentation/home/widget/custom_button_widget.dart';
import 'package:netflix_clone/presentation/home/widget/number_card.dart';
import 'package:netflix_clone/presentation/home/widget/number_title_card.dart';
import 'package:netflix_clone/presentation/widget/main_title.dart';
import 'package:netflix_clone/presentation/widget/main_title_card.dart';
```

```
// ignore: must_be_immutable
class ScreenHome extends StatelessWidget {
  ScreenHome({Key? key}) : super(key: key);

  ValueNotifier<bool> scrollNotifier = ValueNotifier(true);
  @override
  Widget build(BuildContext context) {
    WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
      BlocProvider.of<HomeBloc>(context).add(const GetHomeScreenData());
    });
    return Scaffold(
        body: ValueListenableBuilder(
            valueListenable: scrollNotifier,
            builder: (BuildContext context, index2, _) {
              return NotificationListener<UserScrollNotification>(
                onNotification: (notificaton) {
                  final ScrollDirection direction = notificaton.direction;
                  if (direction == ScrollDirection.reverse) {
                    scrollNotifier.value = false;
                  } else if (direction == ScrollDirection.forward) {
                    scrollNotifier.value = true;
                  }
                  return true;
                },
                child: RefreshIndicator(
                  onRefresh: () async {
                    BlocProvider.of<HomeBloc>(context)
                        .add(const GetHomeScreenData());
                  },
                  child: Stack(
                    children: [
                      BlocBuilder<HomeBloc, HomeState>(
                        builder: (context, state) {
                          if (state.isLoading) {
                            return const Center(
                              child: CircularProgressIndicator(
                                strokeWidth: 2,
                              ),
                            );
                          } else if (state.hasError) {
                            return const Center(
                                child: Text(
                              "error while getting data",
                              style: TextStyle(color: kwhiteColor),
```

```
  ));
}
// released past year
final _releasedPastYear =
    state.pastYearMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
// get id pase year
final _releasedPastYearId =
    state.pastYearMovieList.map((m) {
  return m.id;
}).toList();
// _releasedPastYearId.shuffle();
// trending
final _trending = state.trendingMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
_trending.shuffle();
// get id trendimg
final _trendingId = state.trendingMovieList.map((m) {
  return m.id;
}).toList();
_trendingId.shuffle();
//  trendse drama
final _trendse = state.tenseDramaMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
_trendse.shuffle();
// get id trendse
final _trendsId = state.tenseDramaMovieList.map((m) {
  return m.id;
}).toList();
// south indian movie
final _southIndia =
    state.southIndianMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
// get id pase year
final _southIndiaId =
    state.southIndianMovieList.map((m) {
  return m.id;
}).toList();
// tv shows
final _top10tvShows = state.trendingTvList.map((m) {
```

```
      return '$imageAppendUrl${m.posterPath}';
    }).toList();
    // get id pase year
    final _top10tvShowsId = state.trendingTvList.map((m) {
      return m.id;
    }).toList();
    print(state.trendingMovieList.length);
    return ListView(
      children: [
        const BackgroundCard(),
        kHeight,
        MainTitleCard(
          id: _releasedPastYearId,
          title: "Released in the past year",
          posterList: _releasedPastYear,
        ),
        kHeight,
        MainTitleCard(
          id: _trendingId,
          title: "Trending Now",
          posterList: _trending,
        ),
        kHeight,
        NumberTitleCard(
          posterList: _top10tvShows,
          title: "Top 10 TV shows in india today",
        ),
        kHeight,
        MainTitleCard(
          id: _trendsId,
          title: "Tense Dramas",
          posterList: _trendse,
        ),
        kHeight,
        MainTitleCard(
          id: _southIndiaId,
          title: "South Indian Cinema",
          posterList: _southIndia,
        ),
      ],
    );
  },
),
scrollNotifier.value == true
```

```
                ? AnimatedContainer(
                    duration: const Duration(milliseconds: 1000),
                    width: double.infinity,
                    height: 90,
                    color: Colors.black.withOpacity(0.2),
                    child: Column(
                     children: [
                       Row(
                         children: [
                           Image.network(

'https://cdn-images-1.medium.com/max/1200/1*ty4NvNrGg4ReETxqU2N3Og.png',
                             width: 60,
                             height: 60,
                           ),
                           const Spacer(),
                           const Icon(
                             Icons.cast,
                             size: 30,
                             color: kwhiteColor,
                           ),
                           kWidth,
                           Container(
                             color: Colors.blue,
                             width: 30,
                             height: 30,
                           ),
                           kWidth
                         ],
                       ),
                       Row(
                         mainAxisAlignment:
                             MainAxisAlignment.spaceEvenly,
                         children: const [
                           Text(
                             "TV Shows",
                             style: kHomeTitleText,
                           ),
                           Text(
                             "Movies",
                             style: kHomeTitleText,
                           ),
                           Text(
                             "Categories",
```

```
                                style: kHomeTitleText,
                              )
                            ],
                          )
                        ],
                      ),
                    )
                  : kHeight
              ],
            ),
          ),
        );
      }));
  }
}
```

Main_cart.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:netflix_clone/core/constants.dart';
import '../../application/description/description_bloc.dart';
import '../decriptions/Screen_decription.dart';

class MainCard extends StatelessWidget {
  final String imageUrl;
  const MainCard({
    Key? key,
    required this.imageUrl,
    required this.id,
  }) : super(key: key);
  final int id;
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 10),
      child: GestureDetector(
        onTap: () {
          print("movie id $id");
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (ctx) => ScreenDescription(
                id: id,
              ),
```

```
          ),
        );
      },
      child: Container(
        width: 130,
        height: 250,
        decoration: BoxDecoration(
          borderRadius: kRadius10,
          image: DecorationImage(
            fit: BoxFit.fill,
            image: NetworkImage(imageUrl),
          ),
        ),
      ),
    ),
  );
 }
}
```

Main_title.dart:

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

class MainTitle extends StatelessWidget {
 const MainTitle({Key? key, required this.title}) : super(key: key);

 final String title;
 @override
 Widget build(BuildContext context) {
  return Text(
    title,
    style: const TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
  );
 }
}
```

Main_title_card.dart:

```
import 'package:flutter/material.dart';
import 'package:netflix_clone/core/constants.dart';
import 'package:netflix_clone/presentation/widget/main_card.dart';
import 'package:netflix_clone/presentation/widget/main_title.dart';

class MainTitleCard extends StatelessWidget {
 const MainTitleCard({
```

```
    Key? key,
    required this.title,
    required this.posterList,
    required this.id,
  }) : super(key: key);
  final String title;
  final List<String?> posterList;
  final List<int?> id;
  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        MainTitle(title: title),
        kHeight,
        LimitedBox(
          maxHeight: 200,
          child: ListView(
              scrollDirection: Axis.horizontal,
              children: List.generate(
                  posterList.length,
                  (index) => MainCard(
                        id: id[index]!,
                        imageUrl: posterList[index]!,
                      ))),
        )
      ],
    );
  }
}
```

Video_widget.dart:

```
import 'package:flutter/material.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/core/constants.dart';
import 'package:netflix_clone/presentation/search/widget/search_idel.dart';

class VideoWidget extends StatelessWidget {
  const VideoWidget({
```

```
  Key? key,
  required this.imageUrl,
 }) : super(key: key);

 final String imageUrl;
 @override
 Widget build(BuildContext context) {
  return Stack(
   children: [
    SizedBox(
     width: double.infinity,
     height: 200,
     child: Image.network(
      imageUrl,
      fit: BoxFit.cover,
      loadingBuilder:
        (BuildContext _, Widget child, ImageChunkEvent? progress) {
       if (progress == null) {
        return child;
       } else {
        return const Center(
          child: CircularProgressIndicator(
         strokeWidth: 2,
        ));
       }
      },
      errorBuilder: (BuildContext _, Object a, StackTrace? trace) {
       return const Center(
         child: Icon(
        Icons.wifi,
        color: kwhiteColor,
       ));
      },
     ),
    ),
   ],),}}
```

App_bar_widget.dart:

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/core/constants.dart';

class AppBarWidget extends StatelessWidget {
  const AppBarWidget({Key? key, required this.title}) : super(key: key);
```

```
  final String title;
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        kWidth,
        Text(
          title,
          style: const TextStyle(fontSize: 30, fontWeight: FontWeight.bold),
        ),
        const Spacer(),
        const Icon(
          Icons.cast,
          size: 30,
          color: kwhiteColor,
        ),
        kWidth,
        Container(
          color: Colors.blue,
          width: 30,
          height: 30,
        ),
        kWidth
      ],
    );
  }
}
```

Home_bloc.dart:

```
import 'package:bloc/bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:injectable/injectable.dart';
import 'package:netflix_clone/domain/core/failures/main_failure.dart';
import 'package:netflix_clone/domain/new_and_hot/hot_and_new_service.dart';

import '../../domain/new_and_hot/model/discover.dart';

part 'home_event.dart';
part 'home_state.dart';
part 'home_bloc.freezed.dart';

@injectable
class HomeBloc extends Bloc<HomeEvent, HomeState> {
  final HotAndNewService _homeService;
```

```
HomeBloc(this._homeService) : super(HomeState.initial()) {
  // get home screendata
  on<GetHomeScreenData>((event, emit) async {
    // set loading to ui
    emit(state.copyWith(isLoading: true, hasError: false));
    //   // get datat
    final _movieResult = await _homeService.getHotAndNewMovieData();
    final _tvResutl = await _homeService.getHotAndNewTvData();

    //   //transform data
    final stateOne = _movieResult.fold((MainFailure failures) {
      return HomeState(
        stateId: DateTime.now().millisecondsSinceEpoch.toString(),
        pastYearMovieList: [],
        trendingMovieList: [],
        tenseDramaMovieList: [],
        southIndianMovieList: [],
        trendingTvList: [],
        isLoading: false,
        hasError: true,
      );
    }, (HotAndNewDataResp resp) {
      final pastYear = resp.results;
      pastYear.shuffle();
      final trending = resp.results;
      trending.shuffle();
      final tenseDarama = resp.results;
      tenseDarama.shuffle();
      final southIndia = resp.results;
      southIndia.shuffle();
      return HomeState(
        stateId: DateTime.now().millisecondsSinceEpoch.toString(),
        pastYearMovieList: pastYear,
        trendingMovieList: trending,
        tenseDramaMovieList: tenseDarama,
        southIndianMovieList: southIndia,
        trendingTvList: state.trendingMovieList,
        isLoading: false,
        hasError: false,
      );
    });
    emit(stateOne);
    final stateTwo = _tvResutl.fold((MainFailure failure) {
      return HomeState(
```

```
        stateId: DateTime.now().millisecondsSinceEpoch.toString(),
        pastYearMovieList: [],
        trendingMovieList: [],
        tenseDramaMovieList: [],
        southIndianMovieList: [],
        trendingTvList: [],
        isLoading: false,
        hasError: true,
      );
    }, (HotAndNewDataResp resp) {
      final topTenList = resp.results;
      return HomeState(
        stateId: DateTime.now().millisecondsSinceEpoch.toString(),
        pastYearMovieList: state.pastYearMovieList,
        trendingMovieList: state.trendingMovieList,
        tenseDramaMovieList: state.tenseDramaMovieList,
        southIndianMovieList: state.southIndianMovieList,
        trendingTvList: topTenList,
        isLoading: false,
        hasError: false,
      );
    });
    emit(stateTwo);
  });
 }
}
```

Home_bloc.freezed.dart:

```
part of 'home_bloc.dart';

// **************************************************************************
// FreezedGenerator
// **************************************************************************

T _$identity<T>(T value) => value;

final _privateConstructorUsedError = UnsupportedError(
    'It seems like you constructed your class using `MyClass._()`. This constructor is only meant to be used by freezed and you are not supposed to need it nor use it.\nPlease check the documentation here for more information: https://github.com/rrousselGit/freezed#custom-getters-and-methods');

/// @nodoc
mixin _$HomeEvent {
 @optionalTypeArgs
```

```
  TResult when<TResult extends Object?>({
    required TResult Function() getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult? whenOrNull<TResult extends Object?>({
    TResult Function()? getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult maybeWhen<TResult extends Object?>({
    TResult Function()? getHomeScreenData,
    required TResult orElse(),
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult map<TResult extends Object?>({
    required TResult Function(GetHomeScreenData value) getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult? mapOrNull<TResult extends Object?>({
    TResult Function(GetHomeScreenData value)? getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult maybeMap<TResult extends Object?>({
    TResult Function(GetHomeScreenData value)? getHomeScreenData,
    required TResult orElse(),
  }) =>
      throw _privateConstructorUsedError;
}

/// @nodoc
abstract class $HomeEventCopyWith<$Res> {
  factory $HomeEventCopyWith(HomeEvent value, $Res Function(HomeEvent) then) =
      _$HomeEventCopyWithImpl<$Res>;
}

/// @nodoc
class _$HomeEventCopyWithImpl<$Res> implements $HomeEventCopyWith<$Res> {
  _$HomeEventCopyWithImpl(this._value, this._then);

  final HomeEvent _value;
```

```
  // ignore: unused_field
  final $Res Function(HomeEvent) _then;
}

/// @nodoc
abstract class _$$GetHomeScreenDataCopyWith<$Res> {
  factory _$$GetHomeScreenDataCopyWith(
          _$GetHomeScreenData value, $Res Function(_$GetHomeScreenData) then) =
      __$$GetHomeScreenDataCopyWithImpl<$Res>;
}

/// @nodoc
class __$$GetHomeScreenDataCopyWithImpl<$Res>
    extends _$HomeEventCopyWithImpl<$Res>
    implements _$$GetHomeScreenDataCopyWith<$Res> {
  __$$GetHomeScreenDataCopyWithImpl(
      _$GetHomeScreenData _value, $Res Function(_$GetHomeScreenData) _then)
      : super(_value, (v) => _then(v as _$GetHomeScreenData));

  @override
  _$GetHomeScreenData get _value => super._value as _$GetHomeScreenData;
}

/// @nodoc

class _$GetHomeScreenData implements GetHomeScreenData {
  const _$GetHomeScreenData();

  @override
  String toString() {
    return 'HomeEvent.getHomeScreenData()';
  }

  @override
  bool operator ==(dynamic other) {
    return identical(this, other) ||
        (other.runtimeType == runtimeType && other is _$GetHomeScreenData);
  }

  @override
  int get hashCode => runtimeType.hashCode;

  @override
  @optionalTypeArgs
```

```
TResult when<TResult extends Object?>({
  required TResult Function() getHomeScreenData,
}) {
  return getHomeScreenData();
}

@override
@optionalTypeArgs
TResult? whenOrNull<TResult extends Object?>({
  TResult Function()? getHomeScreenData,
}) {
  return getHomeScreenData?.call();
}

@override
@optionalTypeArgs
TResult maybeWhen<TResult extends Object?>({
  TResult Function()? getHomeScreenData,
  required TResult orElse(),
}) {
  if (getHomeScreenData != null) {
    return getHomeScreenData();
  }
  return orElse();
}

@override
@optionalTypeArgs
TResult map<TResult extends Object?>({
  required TResult Function(GetHomeScreenData value) getHomeScreenData,
}) {
  return getHomeScreenData(this);
}

@override
@optionalTypeArgs
TResult? mapOrNull<TResult extends Object?>({
  TResult Function(GetHomeScreenData value)? getHomeScreenData,
}) {
  return getHomeScreenData?.call(this);
}

@override
@optionalTypeArgs
```

```
  TResult maybeMap<TResult extends Object?>({
    TResult Function(GetHomeScreenData value)? getHomeScreenData,
    required TResult orElse(),
  }) {
    if (getHomeScreenData != null) {
      return getHomeScreenData(this);
    }
    return orElse();
  }
}

abstract class GetHomeScreenData implements HomeEvent {
  const factory GetHomeScreenData() = _$GetHomeScreenData;
}

/// @nodoc
mixin _$HomeState {
  String get stateId => throw _privateConstructorUsedError;
  List<HotAndNewData> get pastYearMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get trendingMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get tenseDramaMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get southIndianMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get trendingTvList => throw _privateConstructorUsedError;
  bool get isLoading => throw _privateConstructorUsedError;
  bool get hasError => throw _privateConstructorUsedError;

  @JsonKey(ignore: true)
  $HomeStateCopyWith<HomeState> get copyWith =>
      throw _privateConstructorUsedError;
}

/// @nodoc
abstract class $HomeStateCopyWith<$Res> {
  factory $HomeStateCopyWith(HomeState value, $Res Function(HomeState) then) =
      _$HomeStateCopyWithImpl<$Res>;
  $Res call(
      {String stateId,
      List<HotAndNewData> pastYearMovieList,
      List<HotAndNewData> trendingMovieList,
      List<HotAndNewData> tenseDramaMovieList,
```

```
    List<HotAndNewData> southIndianMovieList,
    List<HotAndNewData> trendingTvList,
    bool isLoading,
    bool hasError});
}

/// @nodoc
class _$HomeStateCopyWithImpl<$Res> implements $HomeStateCopyWith<$Res> {
  _$HomeStateCopyWithImpl(this._value, this._then);

  final HomeState _value;
  // ignore: unused_field
  final $Res Function(HomeState) _then;

  @override
  $Res call({
    Object? stateId = freezed,
    Object? pastYearMovieList = freezed,
    Object? trendingMovieList = freezed,
    Object? tenseDramaMovieList = freezed,
    Object? southIndianMovieList = freezed,
    Object? trendingTvList = freezed,
    Object? isLoading = freezed,
    Object? hasError = freezed,
  }) {
    return _then(_value.copyWith(
      stateId: stateId == freezed
          ? _value.stateId
          : stateId // ignore: cast_nullable_to_non_nullable
              as String,
      pastYearMovieList: pastYearMovieList == freezed
          ? _value.pastYearMovieList
          : pastYearMovieList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      trendingMovieList: trendingMovieList == freezed
          ? _value.trendingMovieList
          : trendingMovieList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      tenseDramaMovieList: tenseDramaMovieList == freezed
          ? _value.tenseDramaMovieList
          : tenseDramaMovieList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      southIndianMovieList: southIndianMovieList == freezed
          ? _value.southIndianMovieList
```

```
        : southIndianMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
      trendingTvList: trendingTvList == freezed
          ? _value.trendingTvList
          : trendingTvList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      isLoading: isLoading == freezed
          ? _value.isLoading
          : isLoading // ignore: cast_nullable_to_non_nullable
              as bool,
      hasError: hasError == freezed
          ? _value.hasError
          : hasError // ignore: cast_nullable_to_non_nullable
              as bool,
    ));
  }
}

/// @nodoc
abstract class _$$_InitialCopyWith<$Res> implements $HomeStateCopyWith<$Res> {
  factory _$$_InitialCopyWith(
          _$_Initial value, $Res Function(_$_Initial) then) =
      __$$_InitialCopyWithImpl<$Res>;
  @override
  $Res call(
      {String stateId,
      List<HotAndNewData> pastYearMovieList,
      List<HotAndNewData> trendingMovieList,
      List<HotAndNewData> tenseDramaMovieList,
      List<HotAndNewData> southIndianMovieList,
      List<HotAndNewData> trendingTvList,
      bool isLoading,
      bool hasError});
}

/// @nodoc
class __$$_InitialCopyWithImpl<$Res> extends _$HomeStateCopyWithImpl<$Res>
    implements _$$_InitialCopyWith<$Res> {
  __$$_InitialCopyWithImpl(_$_Initial _value, $Res Function(_$_Initial) _then)
      : super(_value, (v) => _then(v as _$_Initial));

  @override
  _$_Initial get _value => super._value as _$_Initial;
```

```
@override
$Res call({
  Object? stateId = freezed,
  Object? pastYearMovieList = freezed,
  Object? trendingMovieList = freezed,
  Object? tenseDramaMovieList = freezed,
  Object? southIndianMovieList = freezed,
  Object? trendingTvList = freezed,
  Object? isLoading = freezed,
  Object? hasError = freezed,
}) {
  return _then(_$_Initial(
    stateId: stateId == freezed
        ? _value.stateId
        : stateId // ignore: cast_nullable_to_non_nullable
            as String,
    pastYearMovieList: pastYearMovieList == freezed
        ? _value._pastYearMovieList
        : pastYearMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    trendingMovieList: trendingMovieList == freezed
        ? _value._trendingMovieList
        : trendingMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    tenseDramaMovieList: tenseDramaMovieList == freezed
        ? _value._tenseDramaMovieList
        : tenseDramaMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    southIndianMovieList: southIndianMovieList == freezed
        ? _value._southIndianMovieList
        : southIndianMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    trendingTvList: trendingTvList == freezed
        ? _value._trendingTvList
        : trendingTvList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    isLoading: isLoading == freezed
        ? _value.isLoading
        : isLoading // ignore: cast_nullable_to_non_nullable
            as bool,
    hasError: hasError == freezed
        ? _value.hasError
        : hasError // ignore: cast_nullable_to_non_nullable
            as bool,
```

```
    ));
  }
}

/// @nodoc

class _$_Initial implements _Initial {
  const _$_Initial(
      {required this.stateId,
      required final List<HotAndNewData> pastYearMovieList,
      required final List<HotAndNewData> trendingMovieList,
      required final List<HotAndNewData> tenseDramaMovieList,
      required final List<HotAndNewData> southIndianMovieList,
      required final List<HotAndNewData> trendingTvList,
      required this.isLoading,
      required this.hasError})
      : _pastYearMovieList = pastYearMovieList,
        _trendingMovieList = trendingMovieList,
        _tenseDramaMovieList = tenseDramaMovieList,
        _southIndianMovieList = southIndianMovieList,
        _trendingTvList = trendingTvList;

  @override
  final String stateId;
  final List<HotAndNewData> _pastYearMovieList;
  @override
  List<HotAndNewData> get pastYearMovieList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_pastYearMovieList);
  }

  final List<HotAndNewData> _trendingMovieList;
  @override
  List<HotAndNewData> get trendingMovieList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_trendingMovieList);
  }

  final List<HotAndNewData> _tenseDramaMovieList;
  @override
  List<HotAndNewData> get tenseDramaMovieList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_tenseDramaMovieList);
  }
```

```
final List<HotAndNewData> _southIndianMovieList;
@override
List<HotAndNewData> get southIndianMovieList {
  // ignore: implicit_dynamic_type
  return EqualUnmodifiableListView(_southIndianMovieList);
}

final List<HotAndNewData> _trendingTvList;
@override
List<HotAndNewData> get trendingTvList {
  // ignore: implicit_dynamic_type
  return EqualUnmodifiableListView(_trendingTvList);
}

@override
final bool isLoading;
@override
final bool hasError;

@override
String toString() {
  return 'HomeState(stateId: $stateId, pastYearMovieList: $pastYearMovieList, trendingMovieList:
$trendingMovieList, tenseDramaMovieList: $tenseDramaMovieList, southIndianMovieList:
$southIndianMovieList, trendingTvList: $trendingTvList, isLoading: $isLoading, hasError: $hasError)';
}

@override
bool operator ==(dynamic other) {
  return identical(this, other) ||
      (other.runtimeType == runtimeType &&
          other is _$_Initial &&
          const DeepCollectionEquality().equals(other.stateId, stateId) &&
          const DeepCollectionEquality()
              .equals(other._pastYearMovieList, _pastYearMovieList) &&
          const DeepCollectionEquality()
              .equals(other._trendingMovieList, _trendingMovieList) &&
          const DeepCollectionEquality()
              .equals(other._tenseDramaMovieList, _tenseDramaMovieList) &&
          const DeepCollectionEquality()
              .equals(other._southIndianMovieList, _southIndianMovieList) &&
          const DeepCollectionEquality()
              .equals(other._trendingTvList, _trendingTvList) &&
          const DeepCollectionEquality().equals(other.isLoading, isLoading) &&
```

```
        const DeepCollectionEquality().equals(other.hasError, hasError));
  }

  @override
  int get hashCode => Object.hash(
      runtimeType,
      const DeepCollectionEquality().hash(stateId),
      const DeepCollectionEquality().hash(_pastYearMovieList),
      const DeepCollectionEquality().hash(_trendingMovieList),
      const DeepCollectionEquality().hash(_tenseDramaMovieList),
      const DeepCollectionEquality().hash(_southIndianMovieList),
      const DeepCollectionEquality().hash(_trendingTvList),
      const DeepCollectionEquality().hash(isLoading),
      const DeepCollectionEquality().hash(hasError));

  @JsonKey(ignore: true)
  @override
  _$$_InitialCopyWith<_$_Initial> get copyWith =>
      __$$_InitialCopyWithImpl<_$_Initial>(this, _$identity);
}

abstract class _Initial implements HomeState {
  const factory _Initial(
      {required final String stateId,
      required final List<HotAndNewData> pastYearMovieList,
      required final List<HotAndNewData> trendingMovieList,
      required final List<HotAndNewData> tenseDramaMovieList,
      required final List<HotAndNewData> southIndianMovieList,
      required final List<HotAndNewData> trendingTvList,
      required final bool isLoading,
      required final bool hasError}) = _$_Initial;

  @override
  String get stateId;
  @override
  List<HotAndNewData> get pastYearMovieList;
  @override
  List<HotAndNewData> get trendingMovieList;
  @override
  List<HotAndNewData> get tenseDramaMovieList;
  @override
  List<HotAndNewData> get southIndianMovieList;
  @override
  List<HotAndNewData> get trendingTvList;
```

```
  @override
  bool get isLoading;
  @override
  bool get hasError;
  @override
  @JsonKey(ignore: true)
  _$$_InitialCopyWith<_$_Initial> get copyWith =>
      throw _privateConstructorUsedError;
}
```

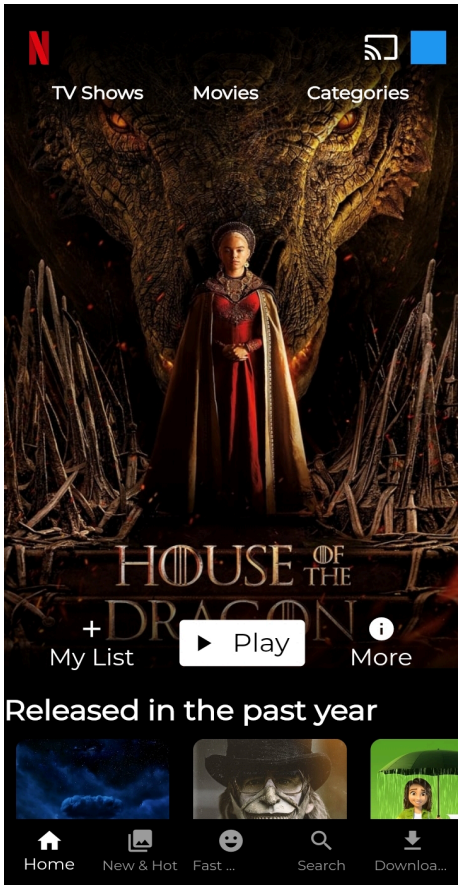
Home_event.dart:

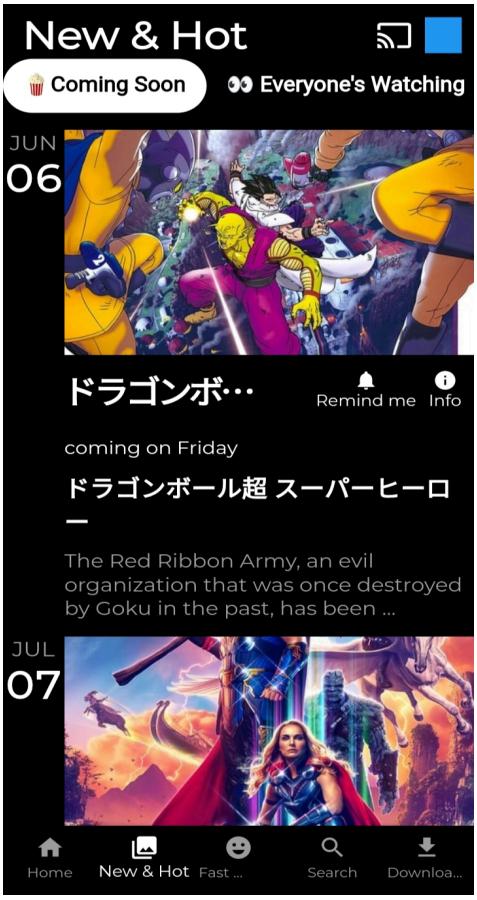
```
part of 'home_bloc.dart';

@freezed
class HomeEvent with _$HomeEvent {
  const factory HomeEvent.getHomeScreenData() = GetHomeScreenData;
}
```

Home_state.dart:

```
part of 'home_bloc.dart';

@freezed
class HomeState with _$HomeState {
  const factory HomeState(
      {required String stateId,
      required List<HotAndNewData> pastYearMovieList,
      required List<HotAndNewData> trendingMovieList,
      required List<HotAndNewData> tenseDramaMovieList,
      required List<HotAndNewData> southIndianMovieList,
      required List<HotAndNewData> trendingTvList,
      required bool isLoading,
      required bool hasError}) = _Initial;
  factory HomeState.initial() => const HomeState(
        stateId: '0',
        pastYearMovieList: [],
        trendingMovieList: [],
        tenseDramaMovieList: [],
        southIndianMovieList: [],
        trendingTvList: [],
        isLoading: false,
        hasError: false,
      );
}
```

Screenshots:





Conclusion

Flutter's widget-based design pattern makes UI development intuitive and flexible. By utilizing common widgets effectively, developers can create seamless, responsive, and visually engaging user interfaces. This experiment provides a foundation for understanding Flutter's UI structure and the role of widgets in application development.

Experiment no:3

Aim: To include icons, images, fonts in Flutter app

Theory:
Introduction

Flutter is a powerful UI development framework that allows developers to create visually appealing applications using a wide range of UI elements. In modern applications, icons, images, and custom fonts play a crucial role in enhancing user experience by improving aesthetics, readability, and branding. Flutter provides built-in support for adding these assets efficiently, ensuring a smooth and responsive UI.

This experiment focuses on understanding how to incorporate icons, images, and fonts into a Flutter application to create a more engaging and customized interface.

2. Objective

The main objectives of this experiment are:

- To learn how to use icons in a Flutter app.
- To understand how to include and display images.
- To explore the process of adding custom fonts for better UI styling.

3. Importance of Icons, Images, and Fonts in UI Design

Icons, images, and fonts enhance the usability and aesthetic appeal of a mobile application.

- Icons provide a visual representation of actions and functionalities, improving navigation and user experience.
- Images enhance the visual storytelling of an app, making it more engaging and informative.
- Custom Fonts help in defining the app's branding and improve text readability, ensuring consistency in UI design.

4. Including Icons in Flutter

Flutter supports two types of icons:

1. Material Icons – Built-in icons provided by Flutter's Material Design framework (Icons class).
2. Custom Icons – Custom icon sets imported as image assets or vector files (SVGs).

5. Adding Images in Flutter

Flutter allows developers to include images in three different ways:

1. Asset Images – Images stored in the project's assets folder.
2. Network Images – Images loaded from an external URL.
3. Memory Images – Images dynamically generated or loaded at runtime.

6. Using Custom Fonts in Flutter

Flutter allows the integration of custom fonts to enhance text styling. Custom fonts provide a unique appearance to the app and align with branding guidelines. To use custom fonts:

1. Font files are placed in the assets/fonts directory.
2. The pubspec.yaml file is updated to register the font.
3. The font is applied using the TextStyle property in Flutter widgets.

Code:

Colors.dart:

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

const backgrounndColor = Colors.black;

const kwhiteColor = Colors.white;

final kButtonColorBlue = Colors.blueAccent[700];

const kButtonColorWhite = Colors.white;

const kBlackColor = Colors.black;

const kGray = Colors.grey;
```

Constants.dart:

```
import 'package:flutter/material.dart';
const kWidth = SizedBox(
  width: 10,
);
const kHeight = SizedBox(
  height: 10,
);
const kHeight20 = SizedBox(
  height: 20,
);
const kHeight50 = SizedBox(
  height: 50,
);
// border radius
final BorderRadius kRadius10 = BorderRadius.circular(10);
final BorderRadius kRadius30 = BorderRadius.circular(30);
// image
const mainImage =
    'https://www.themoviedb.org/t/p/w600_and_h900_bestv2/A69pKujkYeLp7uO4LrUqZvxEAze.jpg';
const imageUrltemp1 =
    'https://www.themoviedb.org/t/p/w533_and_h300_bestv2/iQlJyRecJeGGzQGT2rEcyAgz89F.jpg';
const imageUrltemp2 =
    "https://www.themoviedb.org/t/p/w533_and_h300_bestv2/8TUb2U9GN3PonbXAQ1FBcJ4XeXu.jpg";
// Textstyle
```

const TextStyle kHomeTitleText =

TextStyle(fontSize: 14, fontWeight: FontWeight.bold);

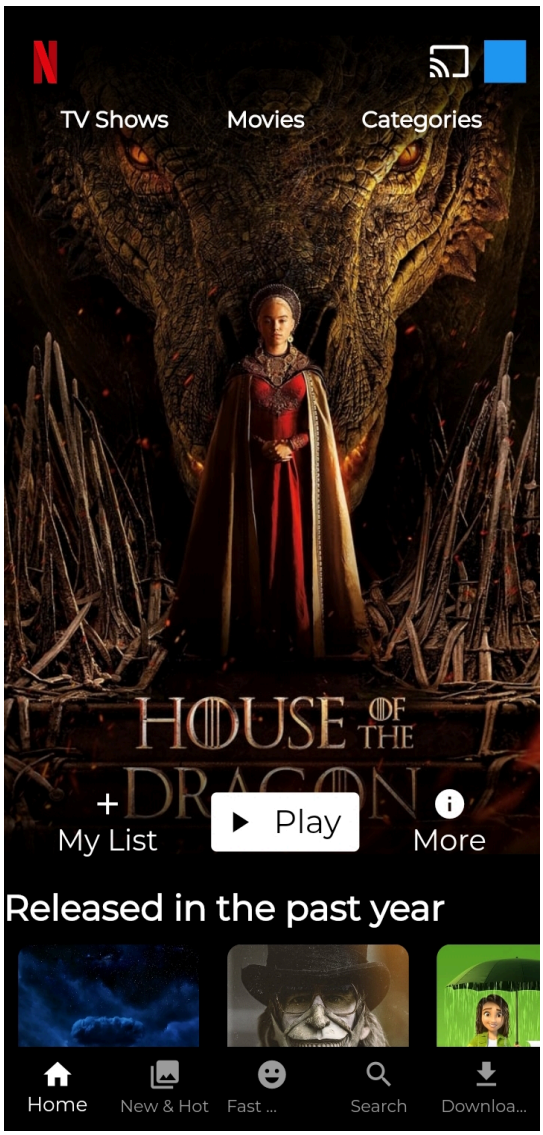
const imageAppendUrl = "https://image.tmdb.org/t/p/w500";

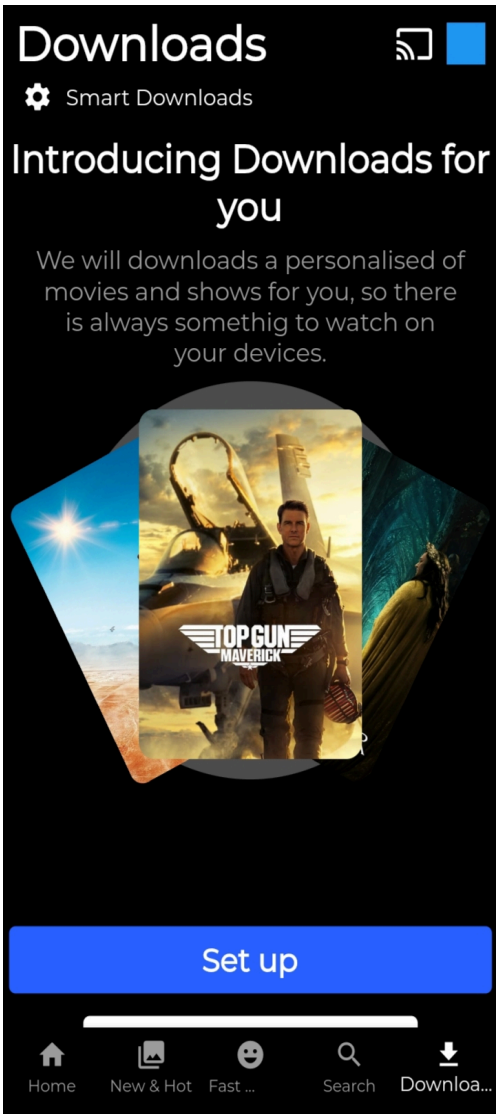
const mainScrenImage =
'https://www.themoviedb.org/t/p/w600_and_h900_bestv2/z2yahl2uefxDCl0nogcRBstwruJ.jpg';

Strings.dart:

const kBaseUrl = "https://api.themoviedb.org/3";

Output:





Conclusion:

Icons, images, and fonts are essential components of a well-designed Flutter application. They contribute to a more polished and professional look, improving both aesthetics and user experience. By understanding how to include and manage these assets, developers can build visually appealing and highly functional Flutter applications.

MPL EXPERIMENT NO: 4

Name: Mahi Jodhani
Class: D15A
Roll No: 21

CODE:

main.dart:

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   theme: ThemeData.dark(),
   home: LoginScreen(),
  );
 }
}
class LoginScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.black,
   body: Padding(
    padding: EdgeInsets.all(20),
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     crossAxisAlignment: CrossAxisAlignment.center,
     children: [
      Text(
       "JioCinema",
       style: TextStyle(
        color: Color(0xFFFF1493),
        fontSize: 28,
        fontWeight: FontWeight.bold,
       ),
```

```
    ),
    SizedBox(height: 40),
    _buildTextField(Icons.email, "Email"),
    SizedBox(height: 20),
    _buildTextField(Icons.lock, "Password", isPassword: true),
    SizedBox(height: 30),
    ElevatedButton(
      onPressed: () {},
      style: ElevatedButton.styleFrom(
        backgroundColor: Color(0xFFFF1493),
        padding: EdgeInsets.symmetric(horizontal: 100, vertical: 15),
      ),
      child: Text(
        "Login",
        style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
      ),
    ),
    SizedBox(height: 20),
    TextButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => SignUpScreen()),
        );
      },
      child: Text(
        "Don't have an account? Sign Up",
        style: TextStyle(color: Color(0xFFFF1493)),
      ),
    ),
    TextButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => PhoneLoginScreen()),
        );
      },
      child: Text(
        "Login with Mobile Number",
        style: TextStyle(color: Color(0xFFFF1493)),
      ),
    ),
  ],
),
```

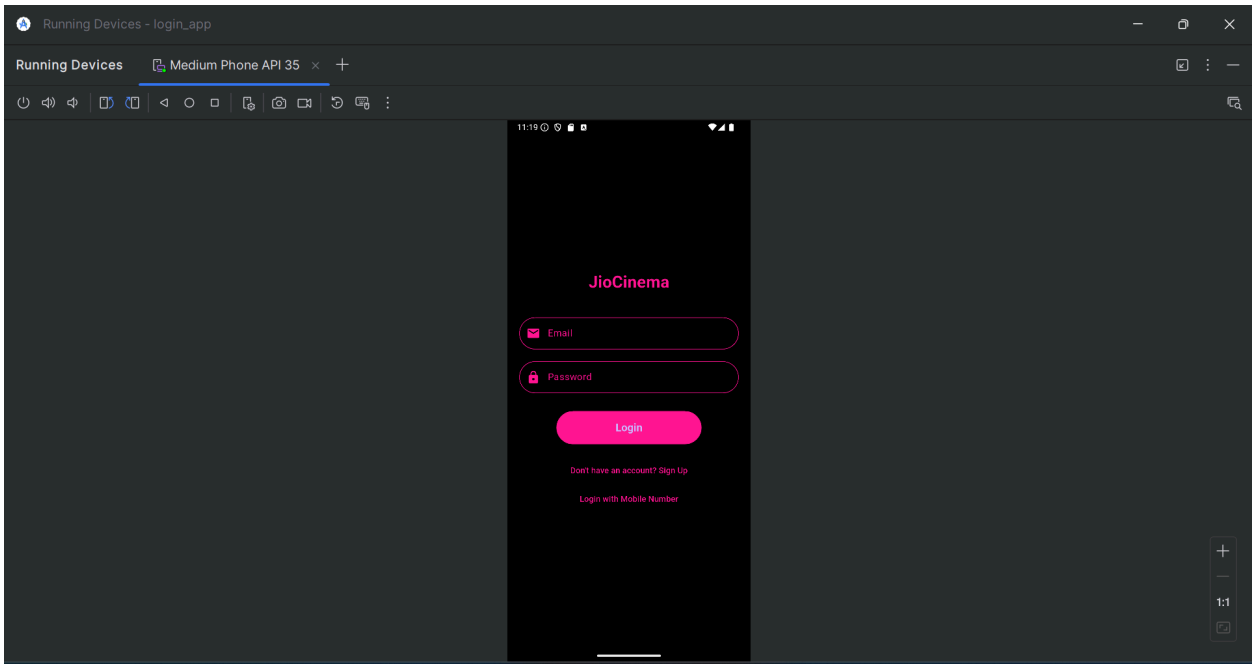
```
      ),
    );
  }
}
class SignUpScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Text(
              "Create Account",
              style: TextStyle(
                color: Color(0xFFFF1493),
                fontSize: 28,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(height: 40),
            _buildTextField(Icons.person, "Full Name"),
            SizedBox(height: 20),
            _buildTextField(Icons.email, "Email"),
            SizedBox(height: 20),
            _buildTextField(Icons.lock, "Password", isPassword: true),
            SizedBox(height: 30),
            ElevatedButton(
              onPressed: () {},
              style: ElevatedButton.styleFrom(
                backgroundColor: Color(0xFFFF1493),
                padding: EdgeInsets.symmetric(horizontal: 90, vertical: 15),
              ),
              child: Text(
                "Sign Up",
                style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
              ),
            ),
            SizedBox(height: 20),
            TextButton(
              onPressed: () {
```

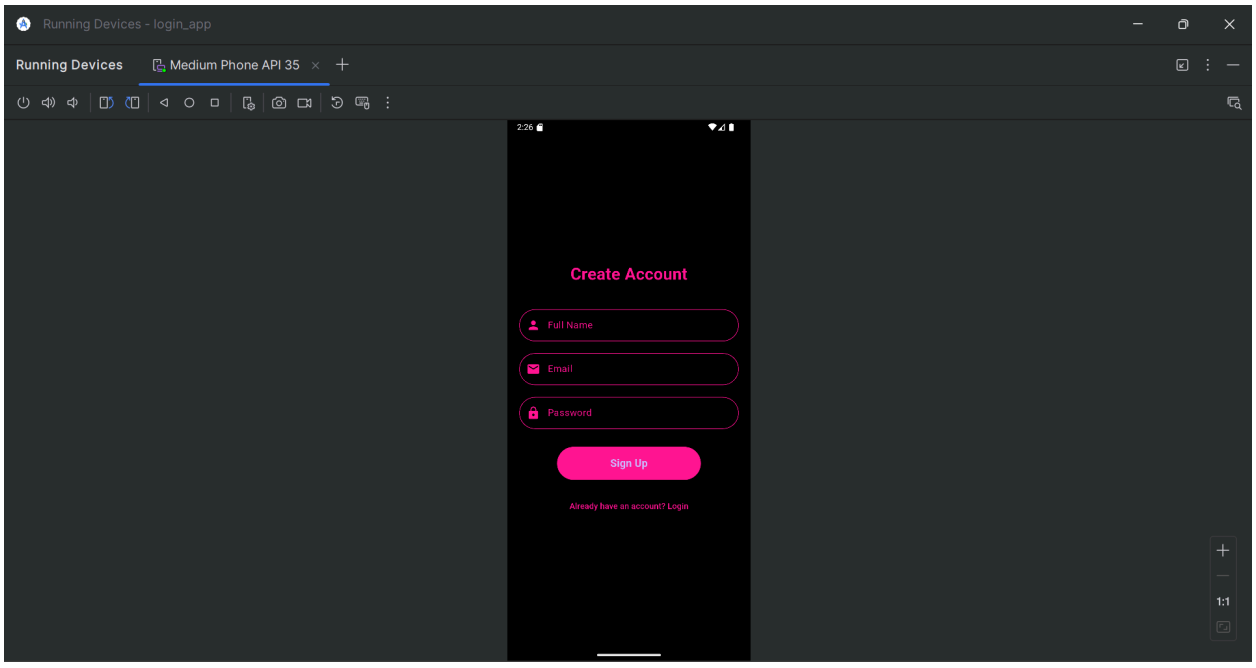
```
            Navigator.pop(context);
          },
          child: Text(
            "Already have an account? Login",
            style: TextStyle(color: Color(0xFFFF1493)),
          ),
        ),
      ],
    ),
  ),
 );
 }
}
class PhoneLoginScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.black,
   body: Padding(
    padding: EdgeInsets.all(20),
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     crossAxisAlignment: CrossAxisAlignment.center,
     children: [
      Text(
       "Enter Mobile Number",
       style: TextStyle(
        color: Color(0xFFFF1493),
        fontSize: 24,
        fontWeight: FontWeight.bold,
       ),
      ),
      SizedBox(height: 30),
      _buildTextField(Icons.phone, "Phone Number"),
      SizedBox(height: 20),
      ElevatedButton(
       onPressed: () {
        Navigator.push(
         context,
         MaterialPageRoute(builder: (context) => OtpScreen()),
        );
       },
       style: ElevatedButton.styleFrom(
        backgroundColor: Color(0xFFFF1493),
```

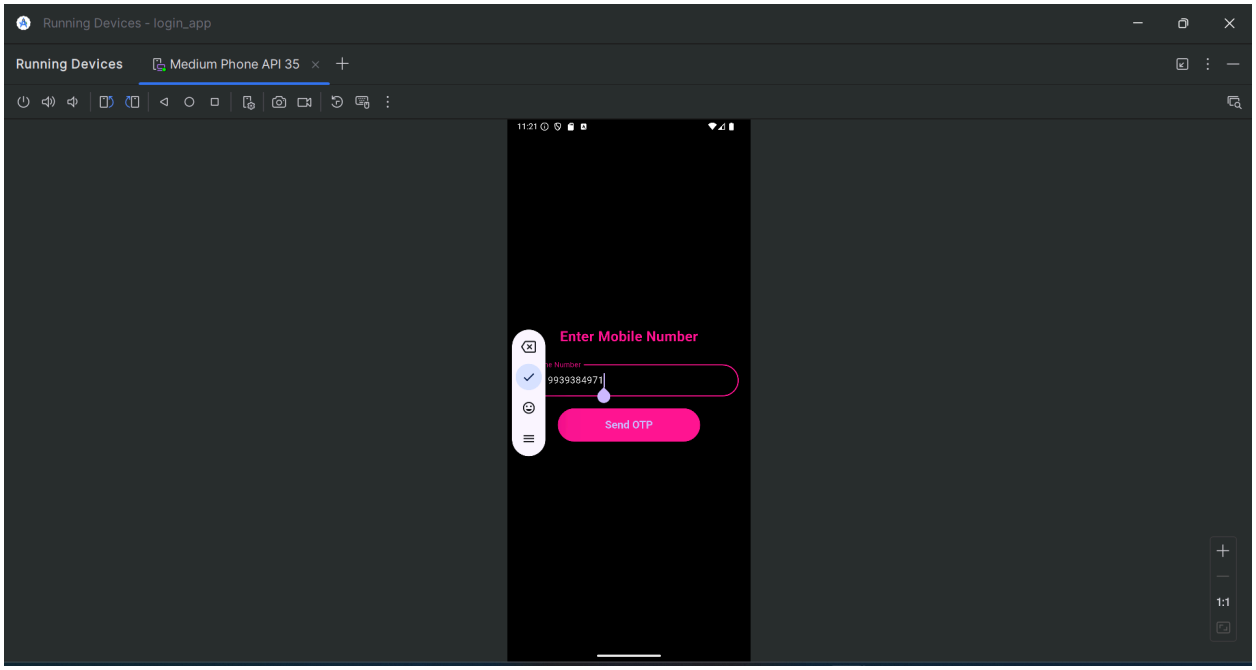
```
            padding: EdgeInsets.symmetric(horizontal: 80, vertical: 15),
          ),
          child: Text(
            "Send OTP",
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),
        ),
      ],
    ),
  ),
 );
 }
}
class OtpScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.black,
   body: Padding(
    padding: EdgeInsets.all(20),
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     crossAxisAlignment: CrossAxisAlignment.center,
     children: [
      Text(
       "Enter OTP",
       style: TextStyle(
        color: Color(0xFFFF1493),
        fontSize: 24,
        fontWeight: FontWeight.bold,
       ),
      ),
      SizedBox(height: 30),
      _buildTextField(Icons.lock, "Enter OTP"),
      SizedBox(height: 20),
      ElevatedButton(
       onPressed: () {},
       style: ElevatedButton.styleFrom(
        backgroundColor: Color(0xFFFF1493),
        padding: EdgeInsets.symmetric(horizontal: 80, vertical: 15),
       ),
       child: Text(
        "Verify OTP",
        style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
```

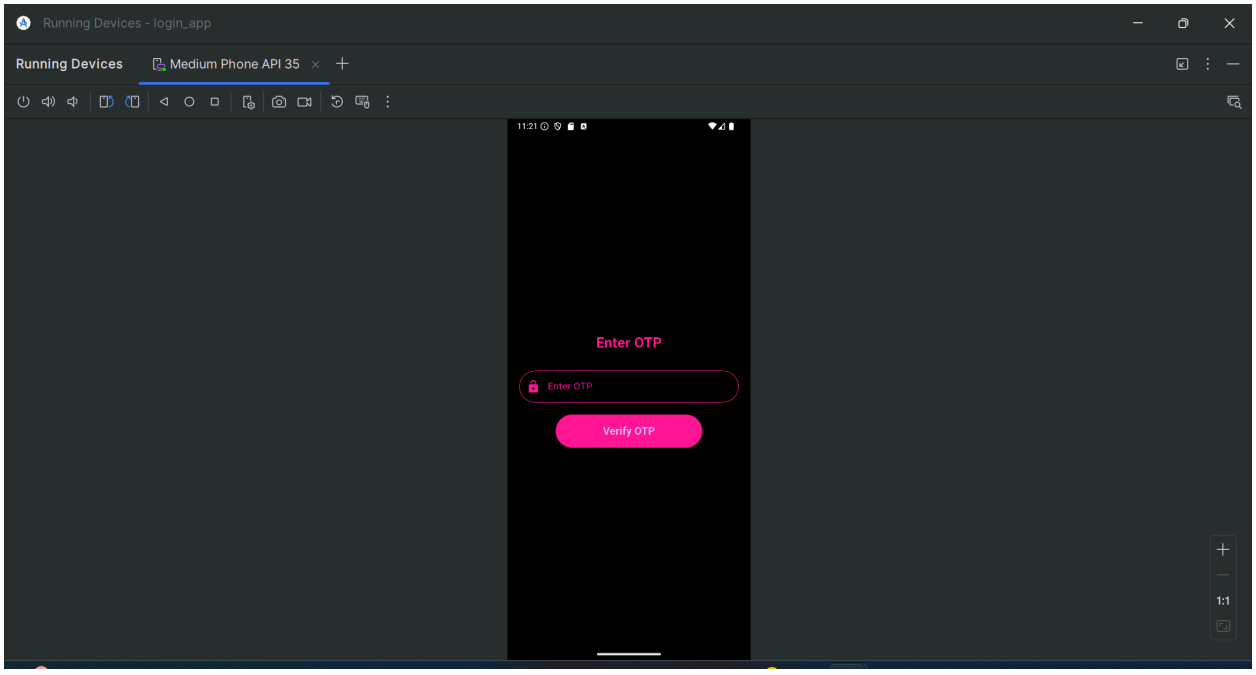
```
            ),
          ),
        ],
      ),
    ),
  );
 }
}

Widget _buildTextField(IconData icon, String label, {bool isPassword = false}) {
 return TextField(
  obscureText: isPassword,
  style: TextStyle(color: Colors.white),
  decoration: InputDecoration(
   labelText: label,
   labelStyle: TextStyle(color: Color(0xFFFF1493)),
   enabledBorder: OutlineInputBorder(
    borderRadius: BorderRadius.circular(30),
    borderSide: BorderSide(color: Color(0xFFFF1493)),
   ),
   focusedBorder: OutlineInputBorder(
    borderRadius: BorderRadius.circular(30),
    borderSide: BorderSide(color: Color(0xFFFF1493), width: 2),
   ),
   prefixIcon: Icon(icon, color: Color(0xFFFF1493)),
   filled: true,
   fillColor: Colors.black,
   contentPadding: EdgeInsets.symmetric(vertical: 15, horizontal: 20),
  ),
 );
}
```

OUTPUT:


Running Devices - login_app
Running Devices
Medium Phone API 35
11:19
JioCinema
Email
Password
Login
Don't have an account? Sign Up
Login with Mobile Number
1:1


Running Devices - login_app
Running Devices
Medium Phone API 35
2:26
Create Account
Full Name
Email
Password
Sign Up
Already have an account? Login
1:1


Running Devices - login_app
Running Devices
Medium Phone API 35
11:21
Enter Mobile Number
9939384971
Send OTP
1:1


Running Devices - login_app
Running Devices
Medium Phone API 35
11:21
Enter OTP
Enter OTP
Verify OTP
1:1

Experiment no: 5

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

Theory:

Introduction:Navigation, routing, and gestures are essential aspects of a Flutter application to ensure seamless interaction and smooth user experience. Navigation allows users to move between different screens, routing defines how screens are structured and managed, and gestures enable touch interactions like taps, swipes, and long presses. Implementing these functionalities properly enhances app usability and improves the overall flow of the application.

2. Objective:The primary objectives of this experiment are:

- To understand navigation and routing concepts in Flutter.
- To implement navigation between different screens using routes.
- To integrate gestures for better user interaction.

3. Importance of Navigation, Routing, and Gestures

Navigation enables users to switch between different screens or pages in an app. It ensures smooth transitions between different UI components, making the app interactive and user-friendly.

Routing:Routing defines the structure of app screens and their navigation paths. Flutter supports two main types of routing:

1. Basic Routing (Navigator Class) – Used for simple navigation between screens using push and pop methods.
2. Named Routing – Predefined routes that make navigation structured and manageable.

Gestures:Gestures allow users to interact with the app through touch-based inputs like tapping, swiping, dragging, and pinching. Flutter's GestureDetector widget helps in detecting and responding to these interactions effectively.

4. Navigation in Flutter:Flutter provides a built-in navigation system using the Navigator class. Navigation can be implemented using:

- Navigator.push() – Moves to a new screen.
- Navigator.pop() – Returns to the previous screen.
- Navigator.pushNamed() – Uses named routes for structured navigation.

5. Implementing Routing in Flutter

Flutter supports two routing mechanisms:

1. Direct Navigation – Using Navigator.push() for simple screen transitions.
2. Named Routes – Defined in the MaterialApp widget, allowing better organization and scalability.

Using named routes is beneficial for large applications with multiple screens, making navigation management more efficient.

6. Gesture Handling in Flutter

Flutter provides the GestureDetector widget to handle user interactions. Common gestures include:

- Tap Gesture – Used for button clicks or selecting UI elements.
- Swipe Gesture – Enables horizontal or vertical movement within the app.
- Long Press – Triggers additional actions on prolonged touch.
- Drag and Drop – Allows moving objects within the app.

Code:

Downloads_bloc.dart:

```
import 'dart:developer';
import 'package:bloc/bloc.dart';
import 'package:dartz/dartz.dart';
import 'package:injectable/injectable.dart';
import 'package:meta/meta.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:netflix_clone/domain/downloads/i_downloads_repos.dart';
import '../../domain/core/failures/main_failure.dart';
import '../../domain/downloads/models/downloads.dart';
part 'downloads_event.dart';
part 'downloads_state.dart';
part 'downloads_bloc.freezed.dart';
@injectable
class DownloadsBloc extends Bloc<DownloadsEvent, DownloadsState> {
  final IDownloadsRepo _downloadsRepo;
  DownloadsBloc(this._downloadsRepo) : super(DownloadsState.inital()) {
    on<_GetDownloadsImage>((event, emit) async {
      emit(
        state.copyWith(
          isLoading: true,
          downloadsFailureSucessOption: none(),
        ),
      );
      final Either<MainFailure, List<Downloads>> downloadsOption =
          await _downloadsRepo.getDownloadsImage();
      // log(downloadsOption.toString());
      emit(
        downloadsOption.fold(
          (failure) => state.copyWith(
            isLoading: false,
            downloadsFailureSucessOption: Some(
              Left(failure),
            ),
```

```
        ),
        (sucess) => state.copyWith(
          isLoading: false,
          downloads: sucess,
          downloadsFailureSucessOption: Some(
            Right(sucess),
          ),
        ),
      ),
    );
  });
 }
}
```

Description_bloc.dart:

```
import 'dart:developer';
import 'package:bloc/bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:injectable/injectable.dart';
import 'package:netflix_clone/domain/description/description_service.dart';
import 'package:netflix_clone/domain/description/model/description_resp/description_resp.dart';
import '../../domain/core/failures/main_failure.dart';
import '../../domain/new_and_hot/model/discover.dart';
part 'description_event.dart';
part 'description_state.dart';
part 'description_bloc.freezed.dart';

@injectable
class DescriptionBloc extends Bloc<DescriptionEvent, DescriptionState> {
  final DescriptionService _descriptionService;
  DescriptionBloc(this._descriptionService)
      : super(DescriptionState.initial()) {
    on<LoadDataMovie>((event, emit) async {
      emit(
        const DescriptionState(
          title: "",
          posterPath: "",
          overview: "",
          status: "",
          relaseDate: "",
          voteAverage: 0,
          isLoading: true,
          isError: false,
        ),
      );
```

```
    final movies =
        await _descriptionService.getMovieDescription(id: event.id);
    final result = movies.fold(
      (l) => const DescriptionState(
        title: "",
        posterPath: "",
        overview: "",
        status: "",
        relaseDate: "",
        voteAverage: 0,
        isLoading: true,
        isError: false,
      ),
      (r) => DescriptionState(
        title: r.title!,
        posterPath: r.posterPath!,
        overview: r.overview!,
        status: r.status!,
        relaseDate: r.releaseDate!,
        voteAverage: r.voteAverage!,
        isLoading: false,
        isError: false,
      ),
    );
    emit(result);
  });
 }
}
```

Fast_laugh_bloc.dart:

```
import 'dart:developer';
import 'package:bloc/bloc.dart';
import 'package:flutter/material.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:injectable/injectable.dart';
import 'package:netflix_clone/domain/core/failures/main_failure.dart';
import 'package:netflix_clone/domain/downloads/i_downloads_repos.dart';
import '../../domain/downloads/models/downloads.dart';
part 'fast_laugh_event.dart';
part 'fast_laugh_state.dart';
part 'fast_laugh_bloc.freezed.dart';
final dummyVideoUrls = [

"https://pagalstatus.com/wp-content/uploads/2021/10/Game-Of-Thrones-Dragon-Scene-Status-Video.mp4
```

```
",
  "https://statuspaji.com/wp-content/uploads/2021/09/Iron-Man-Full-Screen-Status-Video.mp4",
  "https://statusguide.com/anykreeg/2021/04/Naruto-vs-pain-amv-whatsapp-status.mp4",

"https://gostatusguru.com/siteuploads/files/sfd25/12133/Stranger%20Things%20Whatsapp%20Status(Go
StatusGuru.Com).mp4",

"https://statusour.com/wp-content/uploads/2021/09/Money-Heist-Whatsapp-Status-Video-Download-Full
-Screen-4k-Status-9.mp4"
];
ValueNotifier<Set<int>> likedVideosIdNotifier = ValueNotifier({});
@injectable
class FastLaughBloc extends Bloc<FastLaughEvent, FastLaughState> {
  FastLaughBloc(
    IDownloadsRepo _downlodsService,
  ) : super(FastLaughState.initial()) {
    on<Initialize>((event, emit) async {
      emit(
          // sending loading to ui
          const FastLaughState(
        videosList: [],
        isLoading: true,
        isError: false,
      ));
      // get trending movies
      final _result = await _downlodsService.getDownloadsImage();
      // log(_result.toString());
      final respon = _result.fold(
          (l) => FastLaughState(
                videosList: [],
                isError: true,
                isLoading: false,
              ),
          (r) => FastLaughState(
                videosList: r,
                isLoading: false,
                isError: false,
              ));
      // send to ui
      emit(respon);
    });
    on<LikeVideo>((event, emit) async {
      likedVideosIdNotifier.value.add(event.id);
      likedVideosIdNotifier.notifyListeners();
```

```
    });
    on<UnlikeVideo>((event, emit) async {
      likedVideosIdNotifier.value.remove(event.id);
      likedVideosIdNotifier.notifyListeners();
    });
  }
}
```

Hot_and_new_bloc.dart:

```
import 'package:bloc/bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:injectable/injectable.dart';
import 'package:netflix_clone/domain/new_and_hot/hot_and_new_service.dart';
import 'package:netflix_clone/domain/new_and_hot/model/discover.dart';
import '../../domain/core/failures/main_failure.dart';
part 'hot_and_new_event.dart';
part 'hot_and_new_state.dart';
part 'hot_and_new_bloc.freezed.dart';
@injectable
class HotAndNewBloc extends Bloc<HotAndNewEvent, HotAndNewState> {
  final HotAndNewService _hotAndNewService;
  HotAndNewBloc(this._hotAndNewService) : super(HotAndNewState.initial()) {
    // get hot and new movie data
    on<LoadDataInComingSoon>((event, emit) async {
      // send loading to ui
      emit(const HotAndNewState(
          comingSoonList: [],
          everyOneIsWatchingList: [],
          isLoading: true,
          hasError: false));
      // get data from remote
      final _result = await _hotAndNewService.getHotAndNewMovieData();
      // data in state
      final newState = _result.fold((MainFailure f) {
        return const HotAndNewState(
            comingSoonList: [],
            everyOneIsWatchingList: [],
            isLoading: false,
            hasError: true);
      }, (HotAndNewDataResp resp) {
        return HotAndNewState(
            comingSoonList: resp.results,
            everyOneIsWatchingList: state.everyOneIsWatchingList,
            isLoading: false,
```

```
          hasError: false);
    });
    emit(newState);
  });
  // get hot and new tv data
  on<LoadDataInEveryOnesWatching>((event, emit) async {
    emit(const HotAndNewState(
        comingSoonList: [],
        everyOneIsWatchingList: [],
        isLoading: true,
        hasError: false));
    // get data from remote
    final _result = await _hotAndNewService.getHotAndNewTvData();
    // data in state
    final newState = _result.fold((MainFailure f) {
      return const HotAndNewState(
          comingSoonList: [],
          everyOneIsWatchingList: [],
          isLoading: false,
          hasError: true);
    }, (HotAndNewDataResp resp) {
      return HotAndNewState(
          comingSoonList: state.comingSoonList,
          everyOneIsWatchingList: resp.results,
          isLoading: false,
          hasError: false);
    });
    emit(newState);
  });
 }
}
```

Search_bloc.dart:

```
import 'package:bloc/bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:injectable/injectable.dart';
import 'package:netflix_clone/domain/core/failures/main_failure.dart';
import 'package:netflix_clone/domain/downloads/i_downloads_repos.dart';
import 'package:netflix_clone/domain/search/search_service.dart';

import '../../domain/downloads/models/downloads.dart';
import '../../domain/search/model/search_resp/search_resp.dart';

part 'search_event.dart';
```

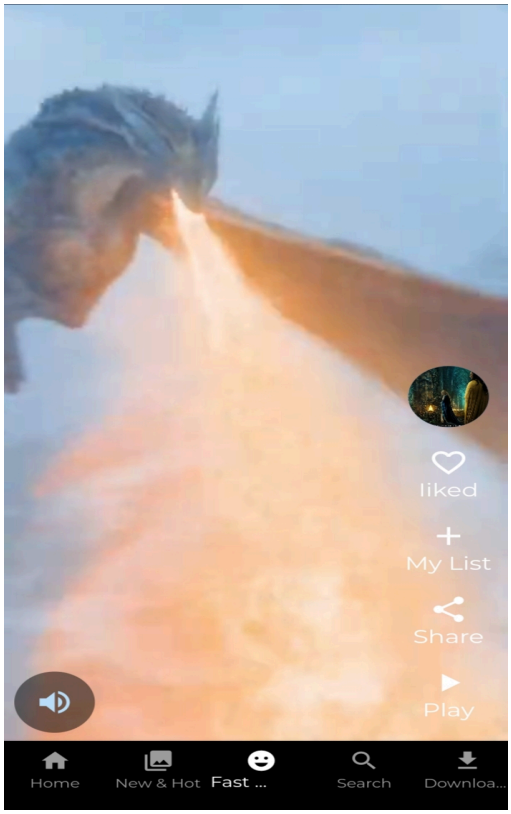
```
part 'search_state.dart';
part 'search_bloc.freezed.dart';

@injectable
class SearchBloc extends Bloc<SearchEvent, SearchState> {
  final IDownloadsRepo _downloadsService;
  final SearchService _searchService;
  SearchBloc(this._searchService, this._downloadsService)
      : super(SearchState.initial()) {
    // idel state
    on<Initalize>((event, emit) async {
      if (state.ideleList.isNotEmpty) {
        emit(SearchState(
          searchResultList: [],
          ideleList: state.ideleList,
          isLoading: false,
          isError: false,
        ));
        return;
      }
      emit(
        const SearchState(
          searchResultList: [],
          ideleList: [],
          isLoading: true,
          isError: false,
        ),
      );
      // get trending
      final _result = await _downloadsService.getDownloadsImage();
      final _state = _result.fold((MainFailure f) {
        return const SearchState(
          searchResultList: [],
          ideleList: [],
          isLoading: false,
          isError: true,
        );
      }, (List<Downloads> list) {
        return SearchState(
          searchResultList: [],
          ideleList: list,
          isLoading: false,
          isError: false,
        );
```

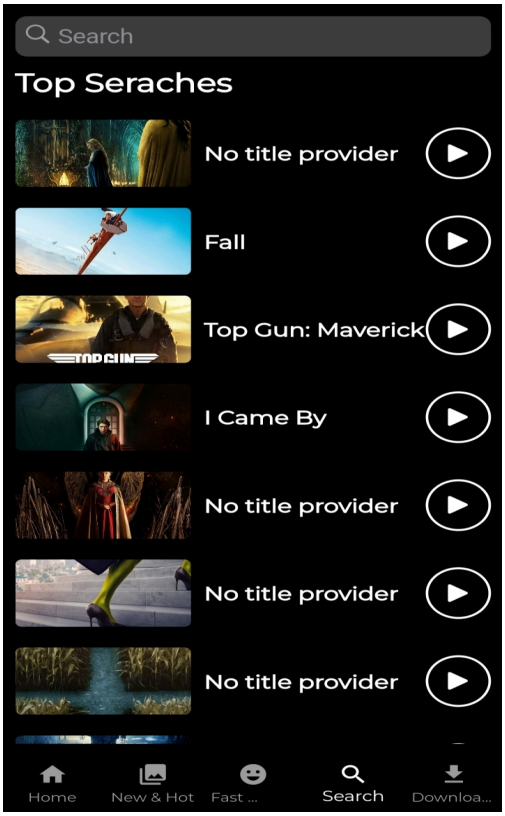
```
    });
    emit(_state);
    // show to ui
   });
   // serch result state
   on<SerachMovie>((event, emit) async {
    // call search movie api
    emit(
     const SearchState(
      searchResultList: [],
      ideleList: [],
      isLoading: true,
      isError: false,
     ),
    );
    final _result =
      await _searchService.searchMovies(movieQuery: event.movieQuery);
    final _state = _result.fold((MainFailure f) {
     return const SearchState(
      searchResultList: [],
      ideleList: [],
      isLoading: false,
      isError: true,
     );
    }, (SearchResp r) {
     return SearchState(
      searchResultList: r.results,
      ideleList: [],
      isLoading: false,
      isError: false,
     );
    });
    // show to ui
    emit(_state);
   });
  }
}
```

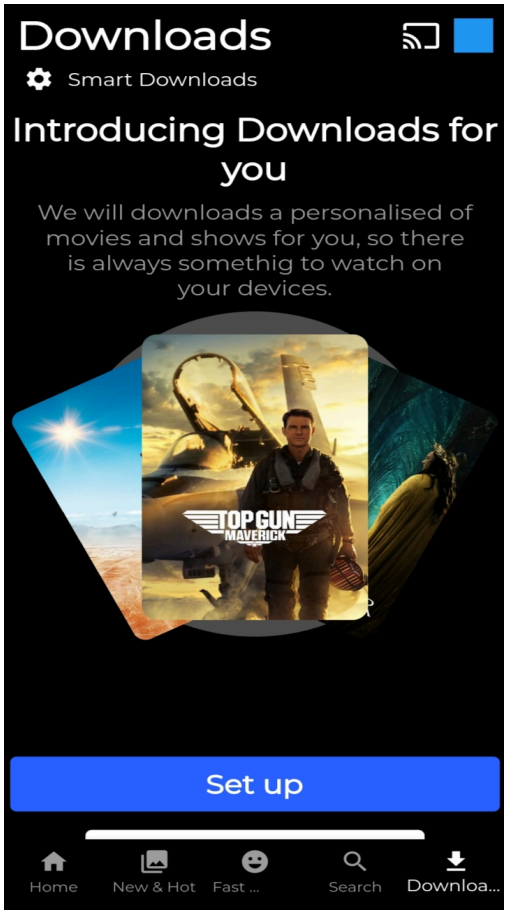
OUTPUT:

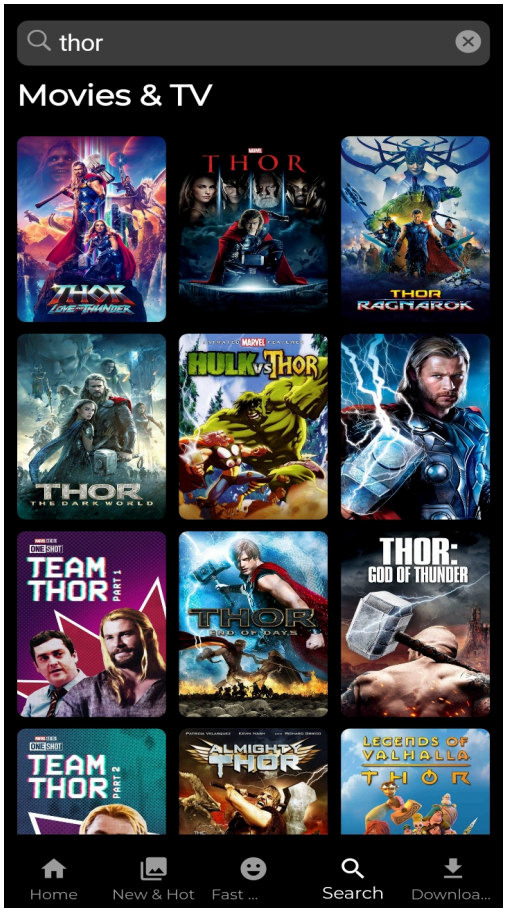

TV Shows
Movies
Categories
HOUSE OF THE DRAGON
My List
Play
More
Released in the past year
Home
New & Hot
Fast ...
Search
Downloa...


New & Hot
Coming Soon
Everyone's Watching
JUN
06
ドラゴンボ･･･
Remind me
Info
coming on Friday
ドラゴンボール超 スーパーヒーロー
The Red Ribbon Army, an evil organization that was once destroyed by Goku in the past, has been ...
JUL
07
Home
New & Hot
Fast ...
Search
Downloa...


liked
My List
Share
Play
Home
New & Hot
Fast ...
Search
Downloa...


Search
Top Seraches
No title provider
Fall
Top Gun: Maverick
I Came By
No title provider
No title provider
No title provider
Home
New & Hot
Fast ...
Search
Downloa...





Conclusion:

Navigation, routing, and gestures are fundamental in creating a dynamic and interactive Flutter application. Proper navigation ensures users can move seamlessly between screens, routing organizes app structure efficiently, and gestures provide a smooth and intuitive user experience. By implementing these features effectively, developers can create well-structured, responsive, and user-friendly Flutter applications.

EXPERIMENT NO: - 06

Name:-Mahi Jodhani **Class:-** D15A **Roll:No: -** 21

AIM: - To connect Flutter UI with Firebase database.

Theory: -

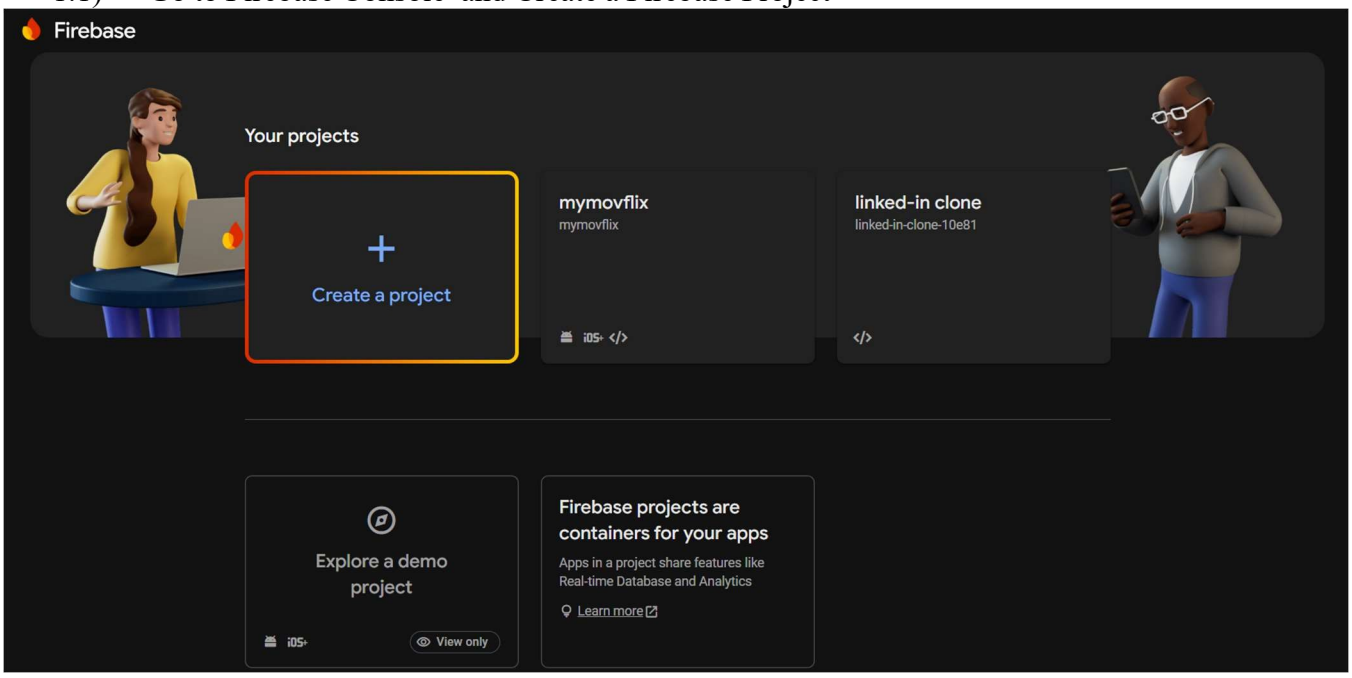
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend solution for Flutter applications.

By integrating Firebase with Flutter, developers can store and retrieve data in real time, authenticate users, and manage cloud-based data efficiently. This is particularly useful for applications requiring dynamic content updates and user interactions.

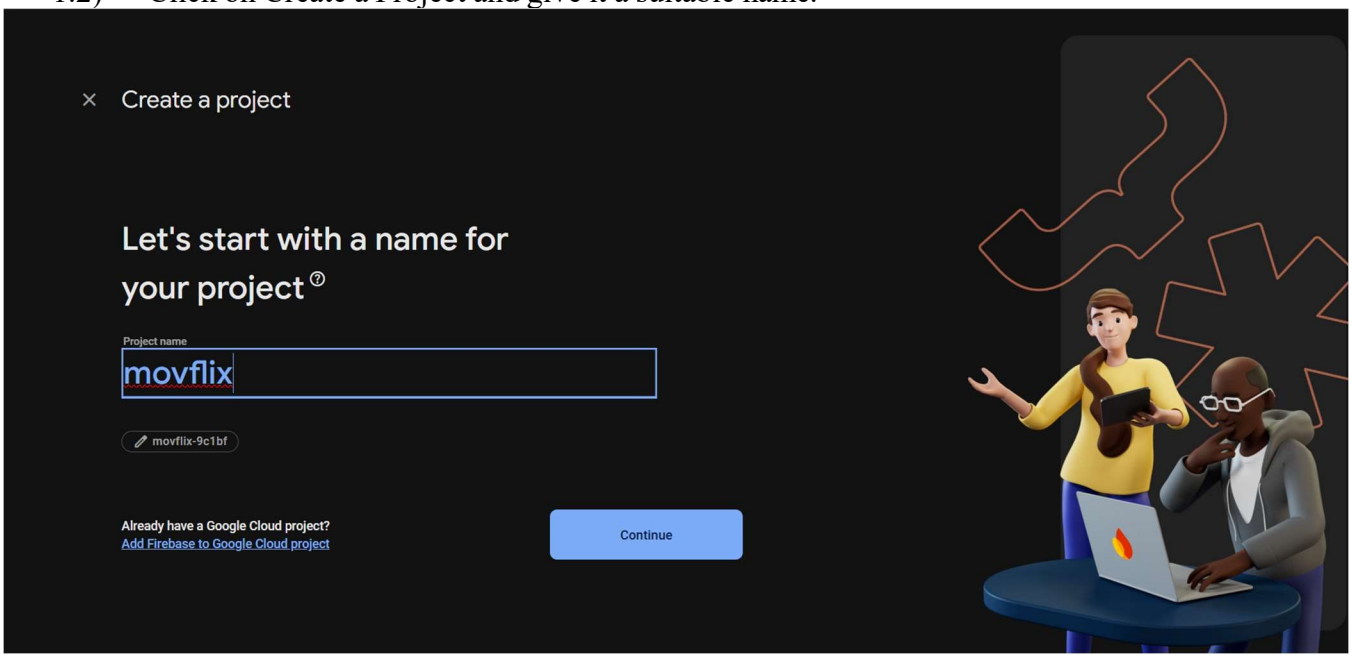
Steps to Connect Flutter UI with Firebase Database

Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

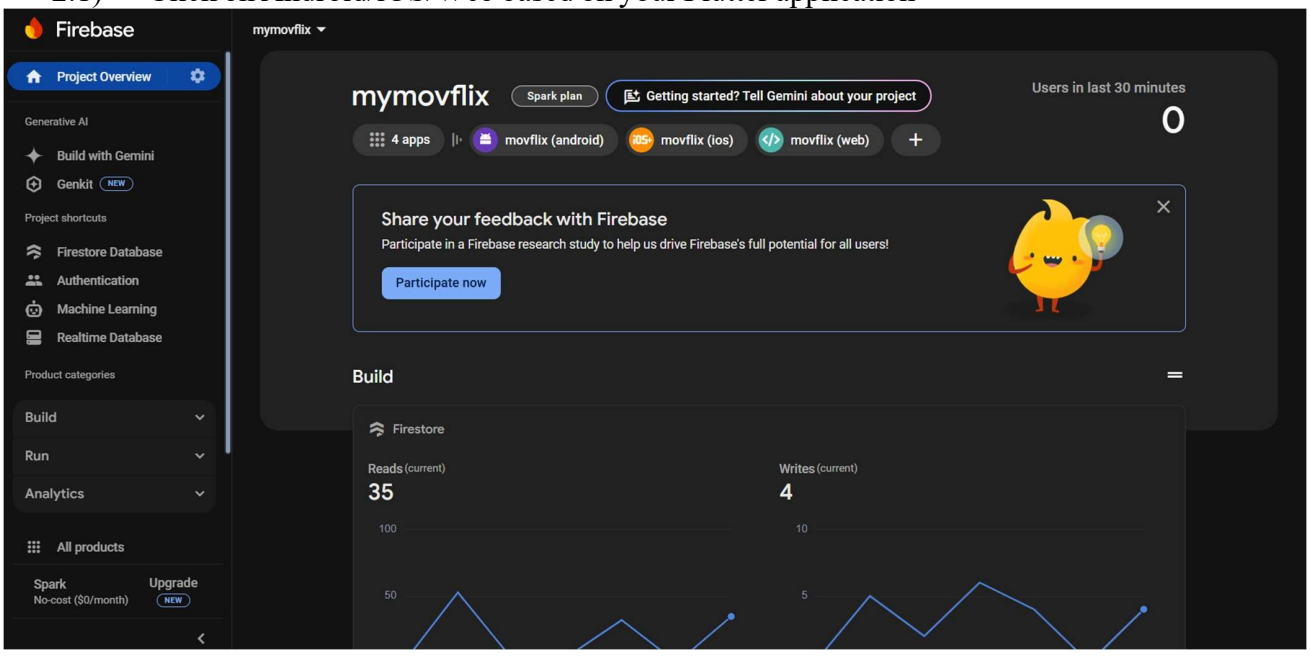


1.2) Click on Create a Project and give it a suitable name.



Step 2:- Add Firebase to Your Flutter App

2.1) Click on Android/iOS/Web based on your Flutter application

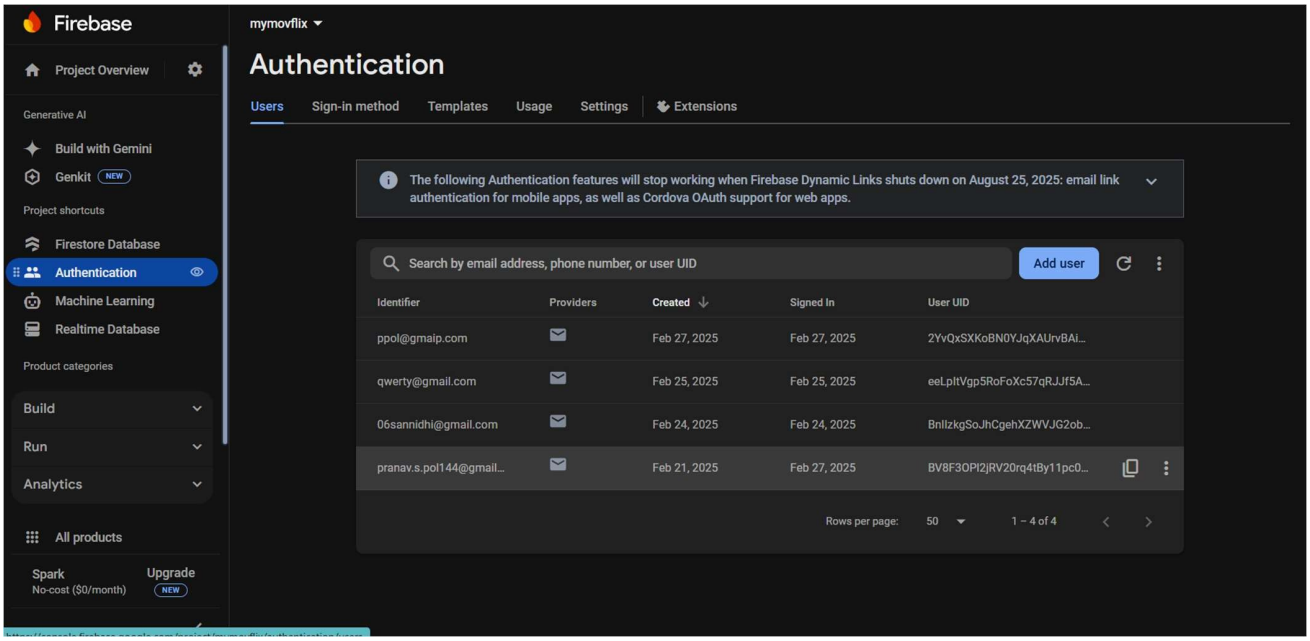


Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^3.11.0
  firebase_auth: ^5.4.2  # For authentication
  cloud_firestore: ^5.6.3  # For Firestore, if you need it
  firebase_messaging: ^15.2.2
  http: ^0.13.3
  image_picker: ^1.0.4
  tflite_flutter: ^0.11.0
  image: ^3.2.0
  url_launcher: ^6.1.14
```

3.2) Enable Authentication in Firebase Console
Go to **Firebase Console → Authentication**.
Click on **Sign-in method** and enable **Email/Password** (or any other method like Google).
Click Save



3.3) Implement Authentication in Flutter Modify main.dart

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

void main() async { WidgetsFlutterBinding.ensureInitialized(); await Firebase.initializeApp(); runApp(MyApp());
}
```

Step 4: -Configure Firebase Realtime Database

4.1) Go to Firebase Console → Realtime Database.
4.2) Click **Create Database** → Choose location → Set rules (for development, set read/write to true).
4.3) Click **Publish**.

Code:-

Sign_in_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import
'package:cloud_firestore/cloud_firestore.dart';
import
'package:firebase_auth/firebase_auth.dart';

import 'sign_up_page.dart';
import
'package:movflix/screens/homescreen.dart';
import
'package:movflix/widgets/bottom_bar_nav.dart';

class SignInPage extends StatefulWidget
  { @override
  _SignInPageState createState() =>
_SignInPageState();
}

class _SignInPageState extends
State<SignInPage> {
  final FirebaseAuth _auth =
FirebaseAuth.instance;
  final FirebaseFirestore _firestore =
FirebaseFirestore.instance;
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController
_passwordController = TextEditingController();
  String _errorMessage = "";
  bool _isLoading = false;

  Future<void> _signIn() async
    { setState(() {
      _isLoading = true;
      _errorMessage = "";
    });

    try {
      // Firebase Authentication
      UserCredential userCredential = await
_auth.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );

      User? user = userCredential.user;
      if (user != null) {
        // Ensure user document exists before
updating
        var userDoc = await
_firestore.collection("users").doc(user.uid).get()
;
        if (userDoc.exists)
          { await
_firestore.collection("users").doc(user.uid).upda
te({
            "lastLogin":
FieldValue.serverTimestamp(),
          });
        }

        Navigator.pushAndRemoveUntil( co
          ntext,
          MaterialPageRoute(builder: (context) =>
HomeScreen()),
              (route) => false,
        );
        Navigator.of(context).pushReplacement( M
          aterialPageRoute(builder: (context) =>
const BottomNavBar()),
        );
      }
    } on FirebaseAuthException catch (e)
      { setState(() {
        _errorMessage = e.message ?? "Error
signing in";
        _isLoading = false;
      });
    } finally
      { setState(()
      {
        _isLoading = false;
      });
    }
```

```
}                                                                    mainAxisAlignment:


@override
Widget build(BuildContext context)
  { return Scaffold(
    body: Container(
      padding: EdgeInsets.symmetric(horizontal:
15, vertical: 15),
      child:
        Column( chil
        dren: [
          _headerWidget(),
          SizedBox(  heigh
          t: 10,
          ),
          _formWidget(),
        ],
      ),
    ),
  );
}

Widget _headerWidget()
  { return Row(
    children:
      [ InkWell( onT
      ap: () {
          Navigator.pop(context);
        },
        child: Icon(Icons.arrow_back),
      ),
      SizedBox( wid
        th: 10,
      ),
      Container( hei
        ght: 40,
        child: Image.asset('assets/logo.png'),
      )
    ],
  );
}

Widget _formWidget()
  { return
  Expanded(  child:
  Column(
```

```
MainAxisAlignment.cent
      er, children: [
        Contai
          ner(
          paddi
          ng:
EdgeInsets.symmetric(horizontal: 8),
          decoration:
          BoxDecoration( color:
          Colors.grey[800],
          borderRadius:
          BorderRadius.all(
             Radius.circular(5),
            ),
          ),
          child:
            TextFormField( controll
            er: _emailController,
            decoration:
            InputDecoration(
              labelStyle: TextStyle(fontSize:
14, color: Colors.white),
               border: InputBorder.none,
               labelText: "Email or phone number",
            ),
          ),
        ),
        SizedBox( h
          eight: 10,
        ),
        Contai
          ner(
          paddi
          ng:
EdgeInsets.symmetric(horizontal: 8),
          decoration:
          BoxDecoration( color:
          Colors.grey[800],
          borderRadius:
          BorderRadius.all(
             Radius.circular(5),
            ),
          ),
          child: TextFormField(
            controller:
            _passwordController,
            obscureText: true,
               decoration:
                 InputDecoration( labelStyle:
                 TextStyle(fontSize: 14,
color: Colors.white),
                 border: InputBorder.none,
                 labelText: "Password",
               ),
             ),
           ),
          SizedBox( heigh
            t: 15,
          ),
```

```
        InkWell(
          onTap: _isLoading ? null : _signIn,
          child: Container(
            alignment: Alignment.center,
            padding:
EdgeInsets.symmetric(vertical: 15),
            width: double.maxFinite,
            decoration: BoxDecoration(
              color: _isLoading ? Colors.grey :
Colors.transparent,
              border: Border.all(
                color: Colors.grey[600] ??
Colors.grey, width: 2),
            ),
            child: _isLoading
                ? CircularProgressIndicator(color:
Colors.white)
                : Text("Sign In"),
          ),
        ),
        if (_errorMessage.isNotEmpty)
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              _errorMessage,
              style: TextStyle(color: Colors.red),
            ),
          ),
        SizedBox( heigh
          t: 15,
        ),
        Text("Need Help?"),
        SizedBox(
          height: 15,
        ),
        GestureDetector( onTa
          p: ()
          { Navigator.push(
              context,
              MaterialPageRoute(builder: (context)
=> SignUpPage()),
            );
          },
          child: Text(
            "New to Netflix? Sign up now.",
            style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.blue),
          ),
        ),
        SizedBox( heigh
          t: 20,
        ),
        Text(
          "Sign-in is protected by Google
reCAPTCHA to ensure you're not a bot. Learn
more.",
          style:
            TextStyle( font
            Size: 12,
          ),
          textAlign: TextAlign.center,
        )
      ],
    ),
  );
 }
}
```

Sign_up_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import
'package:movflix/widgets/header_widget.da
rt';
import 'sign_in_page.dart';
import
'package:movflix/screens/homescreen.dart';
import
'package:movflix/widgets/bottom_bar_nav.d
art';
import
'package:firebase_auth/firebase_auth.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';

class SignUpPage extends StatefulWidget
  { @override
  _SignUpPageState createState() =>
_SignUpPageState();
}

class _SignUpPageState extends
State<SignUpPage> {
  final FirebaseAuth _auth =
FirebaseAuth.instance;
  final FirebaseFirestore _firestore =
FirebaseFirestore.instance;
  final TextEditingController
_emailController = TextEditingController();
  final TextEditingController
_passwordController =
TextEditingController();
bool _isCheck = false;
String _errorMessage = "";
bool _isLoading = false;

  Future<void> _signUp() async
    { setState(() {
      _isLoading = true;
      _errorMessage = "";
    });

    try {
      // Check if the user already exists in
Firestore
   var existingUser = await _firestore

  .collection('users')
        .where('email', isEqualTo:
_emailController.text.trim())
        .get();

     if (existingUser.docs.isNotEmpty)
       { setState(() {
         _errorMessage = "User already exists.
Please log in.";
         _isLoading = false;
       });
       return;
     }

     // Create user in Firebase Authentication
     UserCredential userCredential = await
_auth.createUserWithEmailAndPassword( email
       : _emailController.text.trim(), password:
       _passwordController.text.trim(),
     );

     // Store user details in Firestore
     await
_firestore.collection('users').doc(userCredential.
user!.uid).set({
       'email': _emailController.text.trim(),
       'password':_passwordController.text.trim(),
       'createdAt': FieldValue.serverTimestamp(),
     });

     // Navigate to Home Page
     Navigator.pushAndRemoveUntil( contex
     t,
       MaterialPageRoute(builder: (context) =>
HomeScreen()),
          (route) => false,
     );

     Navigator.of(context).pushReplacement( Materia
       lPageRoute(builder: (context) =>
const BottomNavBar()),
     );

   } on FirebaseAuthException catch (e)
     { setState(() {
       _errorMessage = e.message ?? "Error
signing up";
     });
```

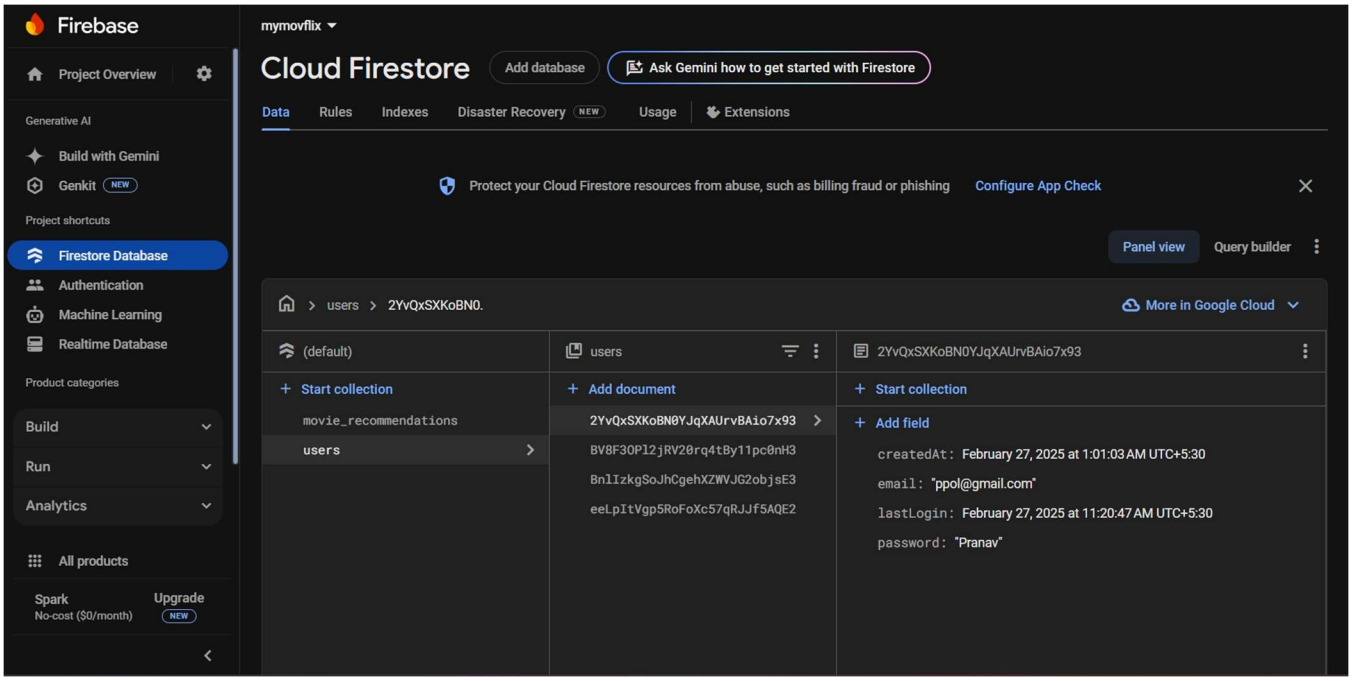
```
  } finally
     { setState(()
     {

  _isLoading = false;
     });
   }
  }

  @override
  Widget build(BuildContext context)
    { return Scaffold(
     body:
       Container( mar
       gin:
EdgeInsets.symmetric(horizontal:  15,
vertical: 10),
       child:
         Column( children:
         [ HeaderWidget(),
           _formWidget(),
         ],
       ),
     ),
   );
  }

  Widget _formWidget()
    { return
   Expanded(  child:
   Column(
       crossAxisAlignment:
CrossAxisAlignment.start,
       children:
         [ SizedBox(height:
         19), Text(
           "Sign up to start
your\nmembership.",
           style: TextStyle(fontSize: 18,
fontWeight: FontWeight.bold, color:
Colors.white),
         ),
         SizedBox(height: 10),
         Text(
           "Create your account",
           style: TextStyle(fontSize: 18,
fontWeight: FontWeight.bold),
         ),
         SizedBox(height: 10),
         Container(
           padding:
color: Colors.grey[800],
             borderRadius:
BorderRadius.all(Radius.circular(5)),
           ),
           child: TextFormField(


     controller: _emailController,
             decoration: InputDecoration(
EdgeInsets.symmetric(horizontal: 8),
           decoration: BoxDecoration(
```

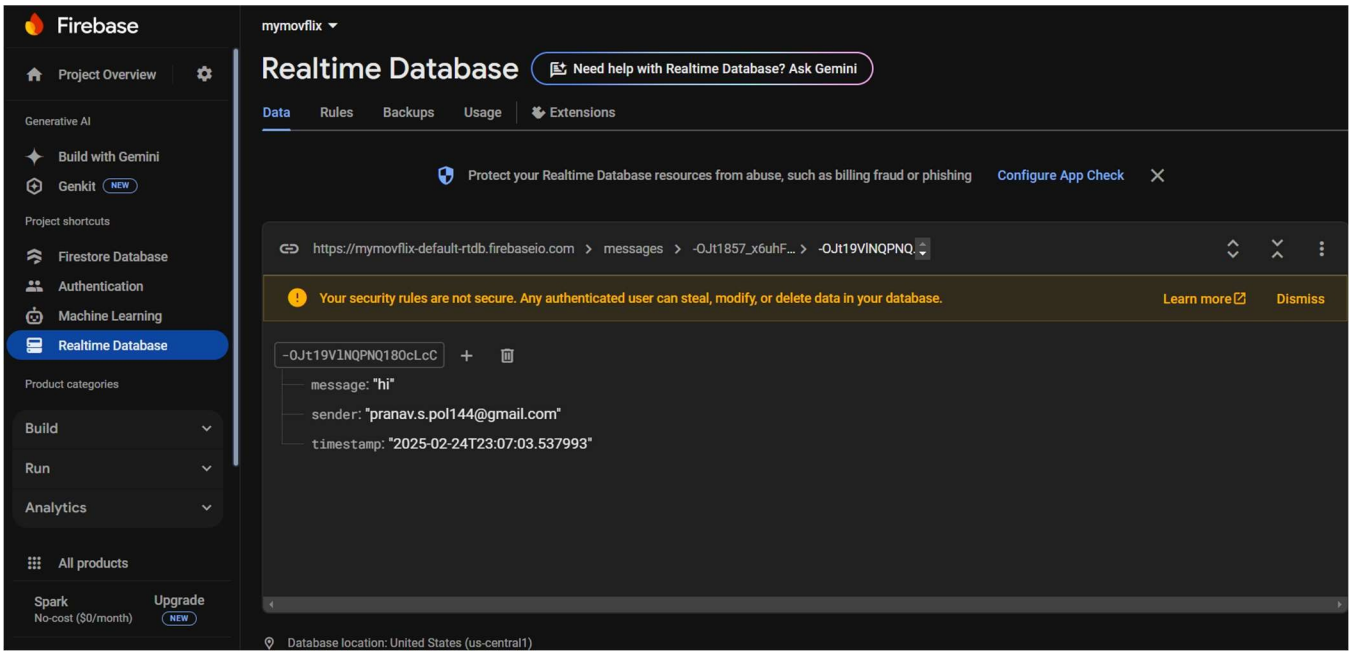
```
            labelText:
            "Email",
            border:
            InputBorder.
            none,
            labelStyle:
TextStyle(fontSize: 14, color:
Colors.white),
            ),
          ),
        ),
        SizedBo
        x(height:
        10),
        Containe
        r(
          padding:
EdgeInsets.symmetric(h
orizontal: 8),
          decoration:
            BoxDecorati
            on( color:
            Colors.grey[8
            00],
            borderRadius:
BorderRadius.all(Radius.circular(5)),
          ),
          child: TextFormField(
            controller:
            _passwordControll
            er, obscureText:
            true,
            decoration:
              InputDecorati
              on( labelText:
              "Password",
              border:
              InputBorder.n
              one,
              labelStyle:
TextStyle(fontSize: 14, color:
Colors.white),
            ),
          ),
        ),
        SizedBo
        x(height:
        15),
        Row(
          c
            h
            ildren:
            [ Checkbo
            x(
              value: _isCheck,
              onChanged: (value)
              { setState(() {
                  _isCheck = value ?? false;
                });
              },
            ),
            Text("Please do not email me Netflix
special offers."),
          ],
        ),
```

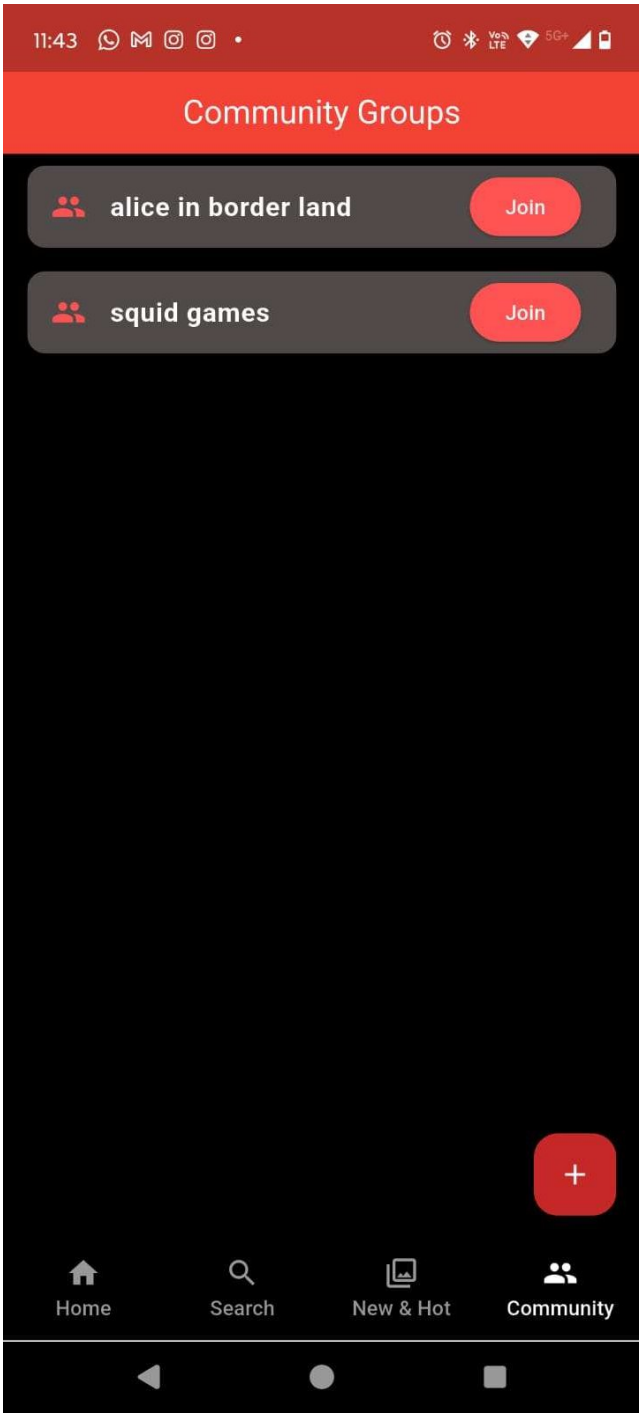
```
  SizedBox(height: 15),
        InkWell(
          onTap: _signUp,
          child: Container(
            alignment: Alignment.center,
            padding:

EdgeInsets.symmetric(vertical: 15),
            width: double.maxFinite,
            decoration:
            BoxDecoration( color:
            Colors.transparent, border:
            Border.all(color:
Colors.grey[600] ?? Colors.grey, width: 2),
            ),
            child: _isLoading
                ? CircularProgressIndicator()
                : Text("Sign Up"),
          ),
        ),
        if (_errorMessage.isNotEmpty)
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(_errorMessage, style:
TextStyle(color: Colors.red)),
          ),
        SizedBox(height: 15),
        GestureDetector( onT
        ap: () {
            Navigator.push(context,
MaterialPageRoute(builder: (context) =>
SignInPage()));
          },
          child: Text(
            "Already have an account? Sign
in.",
            style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.blue),
          ),
        ),
      ],
    ),
  );
 }
}
```

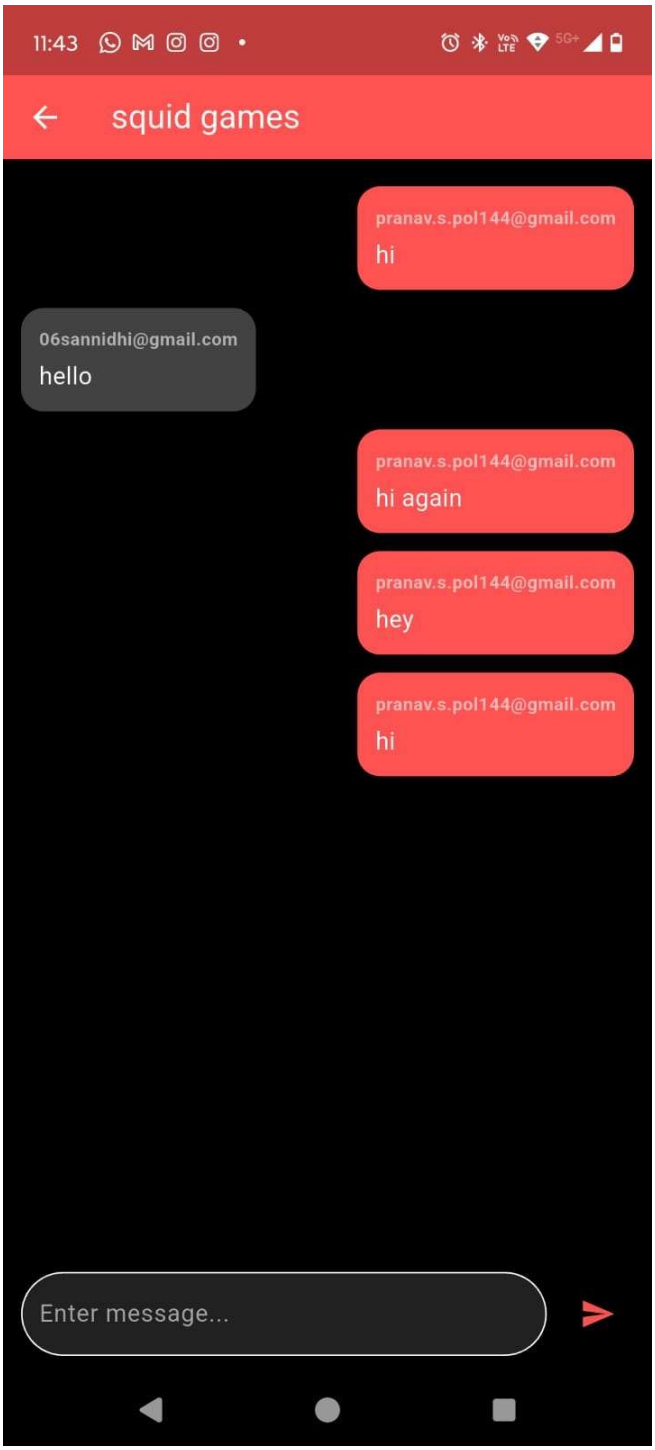
Main.dart

```
import 'package:flutter/material.dart';
import
'package:movflix/widgets/custom_theme.dart';


import
'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'screens/splash_screen.dart';

void main()
 async{ WidgetsFlutterBinding.ensureInitialize
 d(); await Firebase.initializeApp(
   options:
DefaultFirebaseOptions.currentPlatform,
 );
 runApp(const MyApp());
}

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

 // This widget is the root of your application.
 @override
 Widget build(BuildContext context)
  { return MaterialApp(
    title: 'MovFlix ',
    theme: darkTheme,
    debugShowCheckedModeBanner: false,
    home: SplashScreen(),
  );
 }
}
```


Firebase
mymovflix
Project Overview
Generative AI
Build with Gemini
Genkit NEW
Project shortcuts
Firestore Database
Authentication
Machine Learning
Realtime Database
Product categories
Build
Run
Analytics
All products
Spark
No-cost ($0/month)
Upgrade
NEW
Cloud Firestore
Add database
Ask Gemini how to get started with Firestore
Data
Rules
Indexes
Disaster Recovery NEW
Usage
Extensions
Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing
Configure App Check
Panel view
Query builder
users > 2YvQxSXKoBN0.
More in Google Cloud
(default)
users
2YvQxSXKoBN0YJqXAUrvBAio7x93
Start collection
Add document
Start collection
Add field
movie_recommendations
users
2YvQxSXKoBN0YJqXAUrvBAio7x93
BV8F3OP12jRV20rq4tBy11pc0nH3
BnlIzkgSoJhCgehXZWVJG2objsE3
eeLpItVgp5RoFoXc57qRJJf5AQE2
createdAt: February 27, 2025 at 1:01:03 AM UTC+5:30
email: "ppol@gmail.com"
lastLogin: February 27, 2025 at 11:20:47 AM UTC+5:30
password: "Pranav"


Firebase
mymovflix
Project Overview
Generative AI
Build with Gemini
Genkit NEW
Project shortcuts
Firestore Database
Authentication
Machine Learning
Realtime Database
Product categories
Build
Run
Analytics
All products
Spark
No-cost ($0/month)
Upgrade
NEW
Realtime Database
Need help with Realtime Database? Ask Gemini
Data
Rules
Backups
Usage
Extensions
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check
https://mymovflix-default-rtdb.firebaseio.com > messages > -OJt1857_x6uhF... > -OJt19VlNQPNQ...
Your security rules are not secure. Any authenticated user can steal, modify, or delete data in your database.
Learn more
Dismiss
-OJt19VlNQPNQ18OcLcC
message: "hi"
sender: "pranav.s.pol144@gmail.com"
timestamp: "2025-02-24T23:07:03.537993"
Database location: United States (us-central1)


11:43
Community Groups
alice in border land
Join
squid games
Join
Home
Search
New & Hot
Community


11:43
squid games
pranav.s.pol144@gmail.com
hi
06sannidhi@gmail.com
hello
pranav.s.pol144@gmail.com
hi again
pranav.s.pol144@gmail.com
hey
pranav.s.pol144@gmail.com
hi
Enter message...

Experiment No. 7

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

Theory:

- **Regular Web App**

  A regular web app is a website that is designed to be accessible on all mobile devices such that the content gets fit as per the device screen. It is designed using a web technology stack (HTML, CSS, JavaScript, Ruby, etc.) and operates via a browser. They offer various native-device features and functionalities. However, it entirely depends on the browser the user is using. In other words, it might be possible that you can access a native-device feature on Chrome but not on Safari or Mozilla Firefox because the browsers are incompatible with that feature.

- **Progressive Web App**

  Progressive Web App (PWA) is a regular web app, but some extras enable it to deliver an excellent user experience. It is a perfect blend of desktop and mobile application experience to give both platforms to the end-users.

- **Difference between PWAs vs. Regular Web Apps:**

  1. Native Experience: Though a PWA runs on web technologies (HTML, CSS, JavaScript) like a Regular web app, it gives user experience like a native mobile application. It can use most native device features, including push notifications, without relying on the browser or any other entity. It offers a seamless and integrated user experience that it is quite tough for one to differentiate between a PWA and a Native application by considering its look and feel.

  2. Ease of Access: Unlike other mobile apps, PWAs do not demand longer download time and make memory space available for installing the applications. The PWAs can be shared and installed by a link, which cuts down the number of steps to install and use. These applications can easily keep an app icon on the user's home screen, making the app easily accessible to the users and helps the brands remain in the users' minds, and improving the chances of interaction.

  3. Faster Services: PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the brands improve the user engagement and retention rate, which eventually adds value to their business.

  4. Engaging Approach: As already shared, the PWAs can employ push notifications and other native device features more efficiently. Their interaction does not depend on the browser user uses. This eventually improves the chances of notifying the user regarding your services, offers, and other options related to your brand and keeping them hooked to your brand. In simpler words, PWAs let you maintain the user engagement and retention rate.

5. **Updated Real:** Time Data Access: Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.

6. **Discoverable**: PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

7. **Lower Development Cost:** Progressive web apps can be installed on the user device like a native device, but it does not demand submission on an App Store. This makes it far more cost-effective than native mobile applications while offering the same set of functionalities.

- **The main features are:**

1. Progressive — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.
2. Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.
3. Updated — Information is always up-to-date thanks to the data update process offered by service workers.
4. Secure — Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.
5. Searchable — They are identified as “applications” and are indexed by search engines.
6. Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.
7. Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.
8. Linkable — Easily shared via URL without complex installations.
9. Offline — Once more it is about putting the user before everything, avoiding the usual error message in case of weak or no connection. The PWA are based on two particularities: first of all the ‘skeleton’ of the app, which recalls the page structure, even if its contents do not respond and its elements include the header, the page layout, as well as an illustration that signals that the page is loading.

Code:

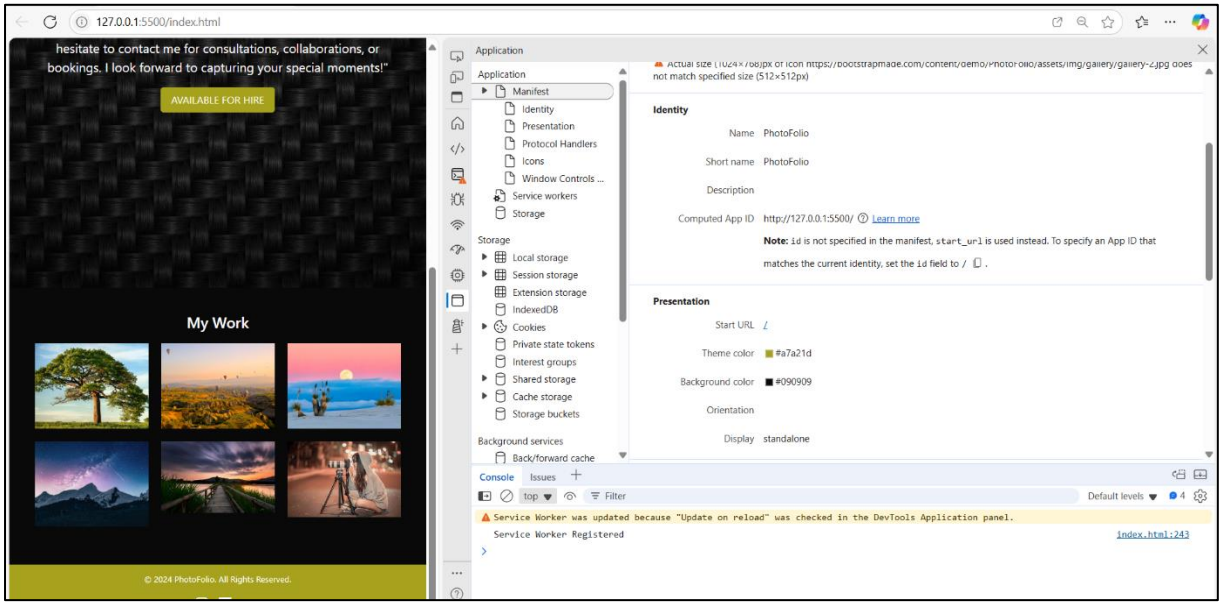
1. **Manifest.json**

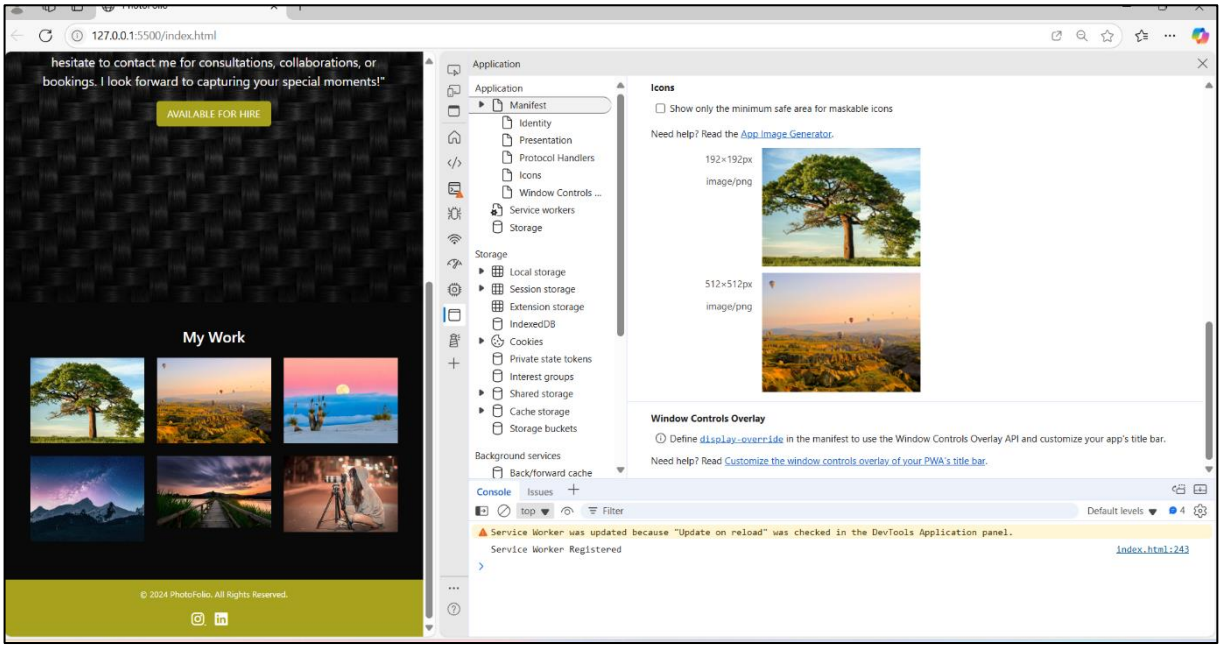
```
{
    "name": "PhotoFolio",
    "short_name": "PhotoFolio",
    "start_url": "/",
    "display": "standalone",
    "background_color": "#090909",
    "theme_color": "#a7a21d",
    "icons": [
      {
        "src":
"https://bootstrapmade.com/content/demo/PhotoFolio/assets/img/gallery/gallery-
1.jpg",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src":
"https://bootstrapmade.com/content/demo/PhotoFolio/assets/img/gallery/gallery-
2.jpg",
        "sizes": "512x512",
        "type": "image/png"
      }
    ]
  }
```

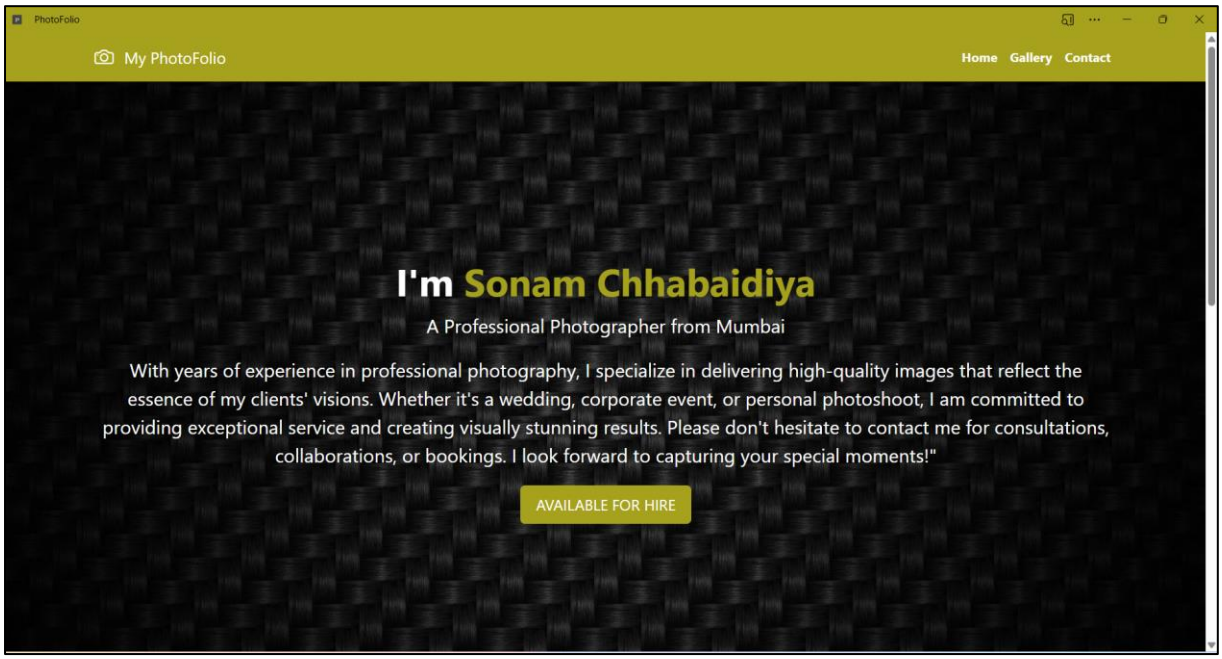
2. Service-worker.js

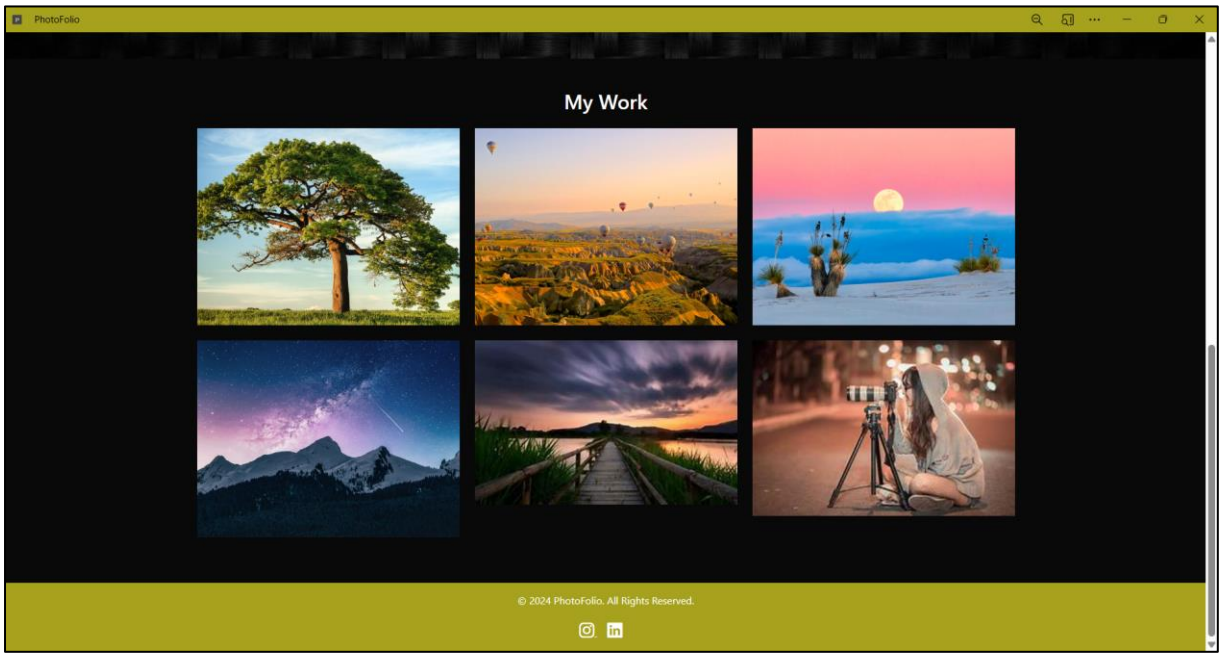
```
self.addEventListener("install", (event) => {
    event.waitUntil(
        caches.open("static").then((cache) => {
            return cache.addAll(["/", "/index.html", "/styles.css"]);
        })
    );
});

self.addEventListener("fetch", (event) => {
    event.respondWith(
        caches.match(event.request).then((response) => {
            return response || fetch(event.request);
        })
    );
})
```

Output:




127.0.0.1:5500/index.html
hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE
My Work
© 2024 PhotoFolio. All Rights Reserved.
Application
Manifest
Identity
Presentation
Protocol Handlers
Icons
Window Controls ...
Service workers
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Icons
Show only the minimum safe area for maskable icons
Need help? Read the App Image Generator.
192×192px
image/png
512×512px
image/png
Window Controls Overlay
Define display-override in the manifest to use the Window Controls Overlay API and customize your app's title bar.
Need help? Read Customize the window controls overlay of your PWA's title bar.
Console
Issues
Service Worker was updated because "Update on reload" was checked in the DevTools Application panel.
Service Worker Registered
index.html:243


PhotoFolio
My PhotoFolio
Home Gallery Contact
I'm Sonam Chhabaidiya
A Professional Photographer from Mumbai
With years of experience in professional photography, I specialize in delivering high-quality images that reflect the essence of my clients' visions. Whether it's a wedding, corporate event, or personal photoshoot, I am committed to providing exceptional service and creating visually stunning results. Please don't hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE


PhotoFolio
My Work
© 2024 PhotoFolio. All Rights Reserved.


.1 mpl
webx exp 5
ew
sonam
P
nshot...
PhotoFolio

Experiment No. 8

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:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.

What can we do with Service Workers?

- You can dominate **Network Traffic**

You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

- You can **Cache**

You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.

- You can manage **Push Notifications**

You can manage push notifications with Service Worker and show any information message to the user.

- You can **Continue**

Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can’t we do with Service Workers?

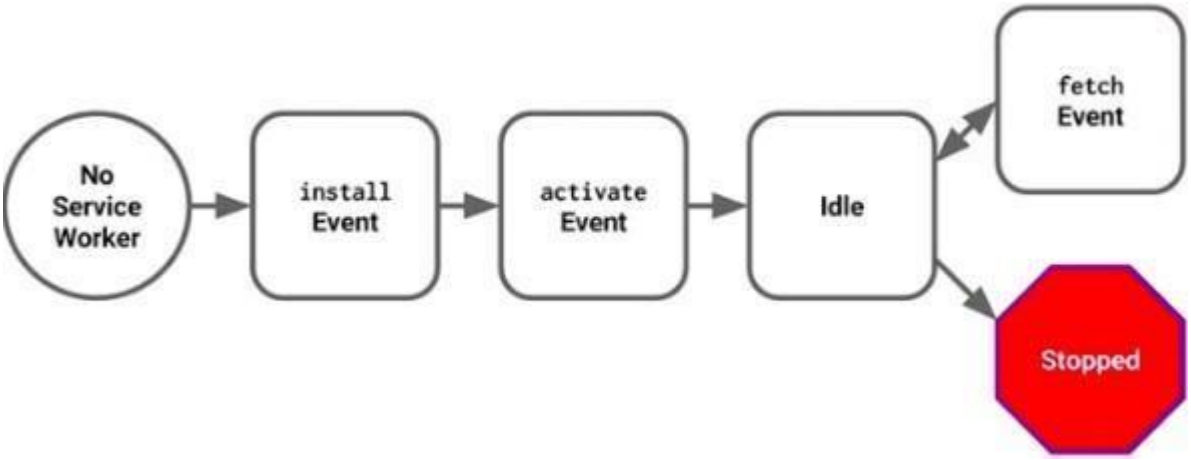
- You can’t access the **Window**

You can’t access the window, therefore, You can’t manipulate DOM elements. But, you can
communicate to the window through post Message and manage processes that you want.

- You can’t work it on **80 Port**

Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Service Worker Cycle



A service worker goes through three steps in its life cycle:

- Registration
- Installation
- Activation

Registration

To install a service worker, you need to register it in your main JavaScript code. Registration tells the browser where your service worker is located, and to start installing it in the background. Let's look at an example:

main.js

```
if ('serviceWorker' in navigator) { navigator.serviceWorker.register('/service-worker.js')
.then(function(registration) {
console.log('Registration successful, scope is:', registration.scope);
})
.catch(function(error) {
console.log('Service worker registration failed, error:', error);
});
}
```

This code starts by checking for browser support by examining **navigator.serviceWorker**. The service worker is then registered with navigator.serviceWorker.register, which returns a promise that resolves when the service worker has been successfully registered. The scope of the service worker is then logged with registration.scope. If the service worker is already installed, navigator.serviceWorker.register returns the registration object of the currently active service worker.

The scope of the service worker determines which files the service worker controls, in other words, from which path the service worker will intercept requests. The default scope is the location of the service worker file, and extends to all directories below. So if service-worker.js is located in the root directory, the service worker will control requests from all files at this domain.

You can also set an arbitrary scope by passing in an additional parameter when registering. For example: main.js

navigator.serviceWorker.register('/service-worker.js', { scope: '/app/'

});

In this case we are setting the scope of the service worker to /app/, which means the service worker will control requests from pages like /app/, /app/lower/ and /app/lower/lower, but not from pages like /app or /, which are higher.

If you want the service worker to control higher pages e.g. /app (without the trailing slash) you can indeed change the scope option, but you'll also need to set the Service-Worker-Allowed HTTP Header in your server config for the request serving the service worker script.

main.js
navigator.serviceWorker.register('/app/service-worker.js', { scope: '/app'

});

Installation

Once the browser registers a service worker, installation can be attempted. This occurs if the service worker is considered to be new by the browser, either because the site currently doesn't have a registered service worker, or because there is a byte difference between the new service worker and the previously installed one.

A service worker installation triggers an install event in the installing service worker. We can include an install event listener in the service worker to perform some task when the service worker installs. For instance, during the install, service workers can precache parts of a web app so that it loads instantly the next time a user opens it (see caching the application shell). So, after that first load, you're going to benefit from instant repeat loads and your time to interactivity is going to be even better in those cases. An example of an installation event listener looks like this:

service-worker.js

// Listen for install event, set callback

self.addEventListener('install', function(event) {

// Perform some task

});

Activation

Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches (see the Offline Cookbook for an example).

service-worker.js

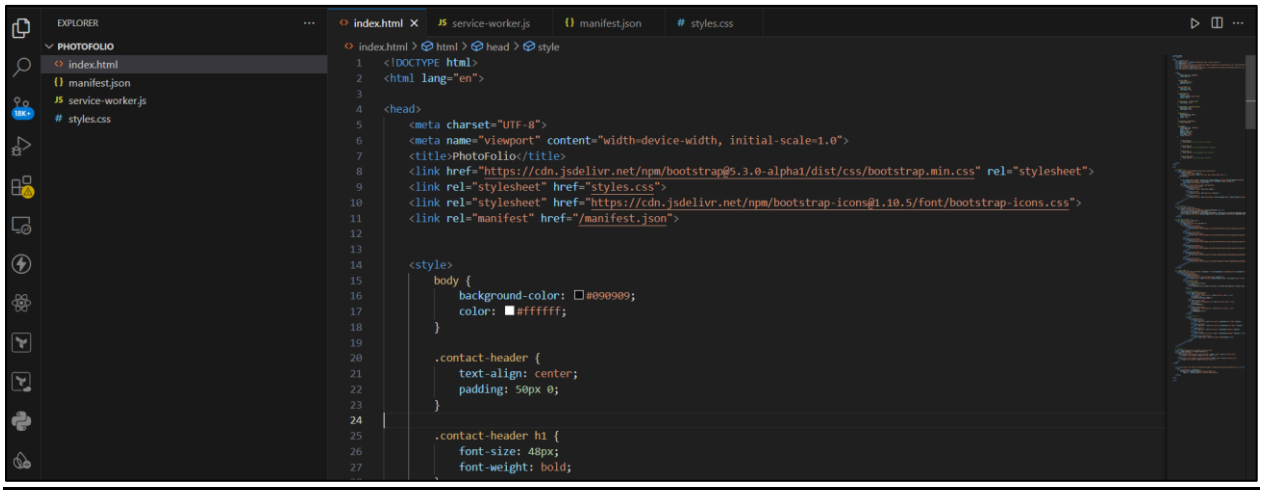
self.addEventListener('activate', function(event) {*// Perform some task*

});

Once activated, the service worker controls all pages that load within its scope, and starts listening for events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when you close and reopen your app, or if the service worker calls **clients.claim()**. Until then, requests from this page will not be intercepted by the new service worker. This is intentional as a way to ensure consistency in your site.

CODE:

Index.html:

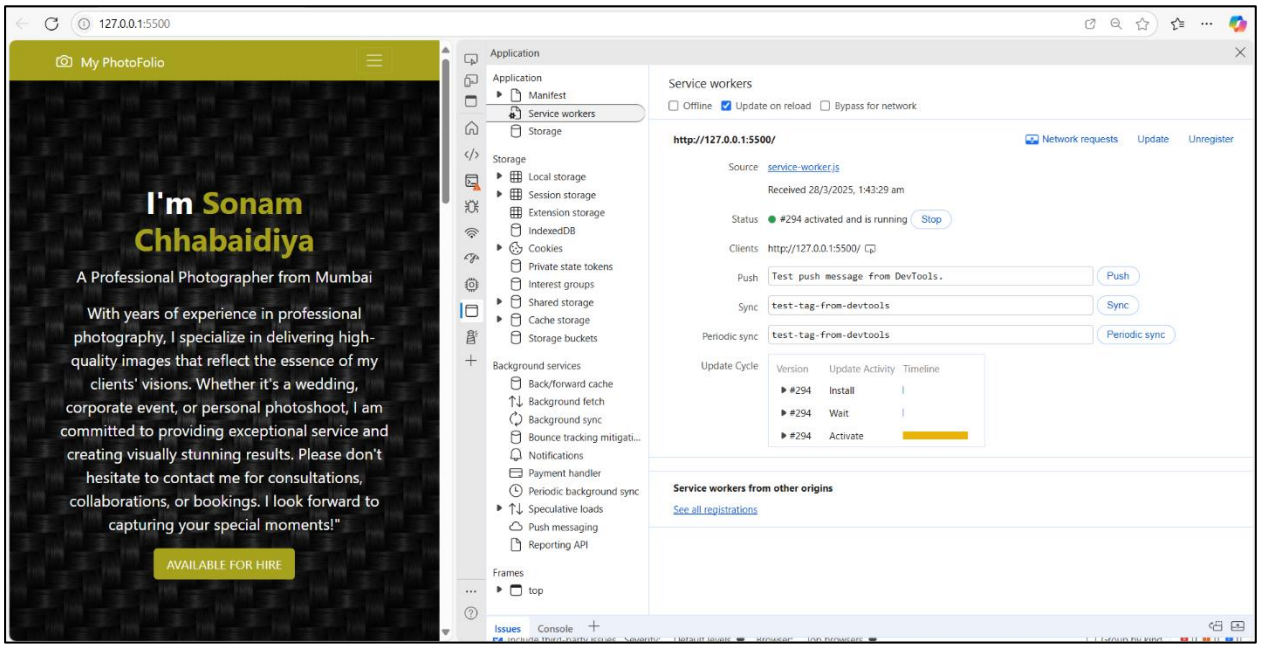


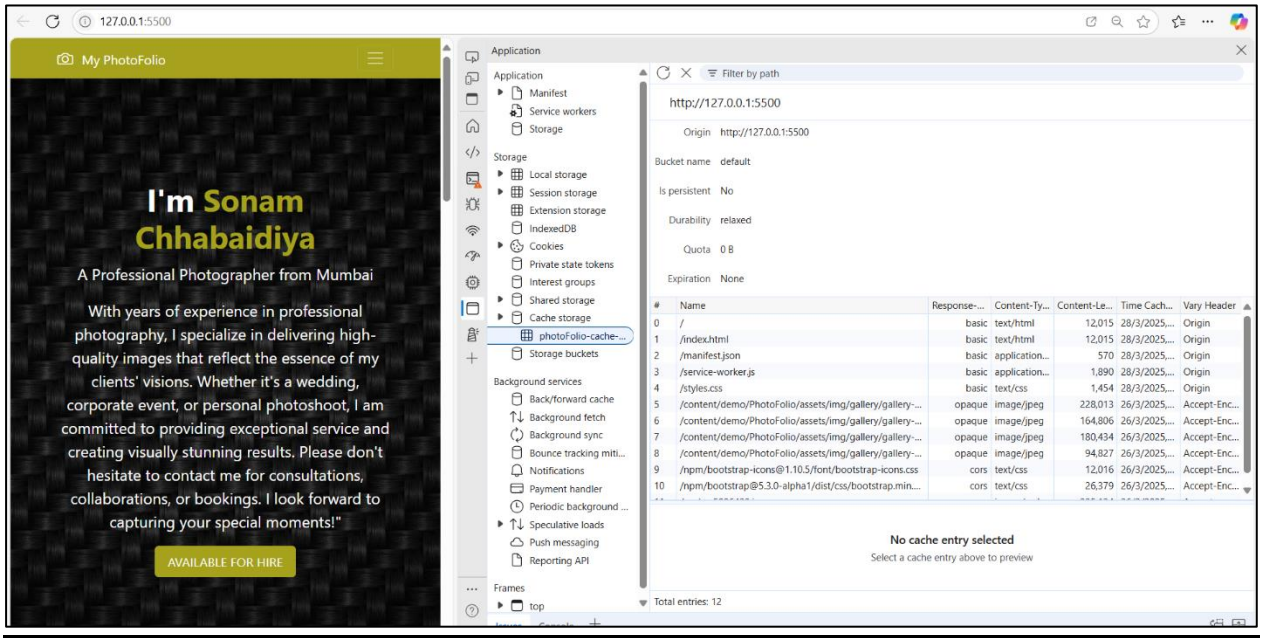
Service_worker.js

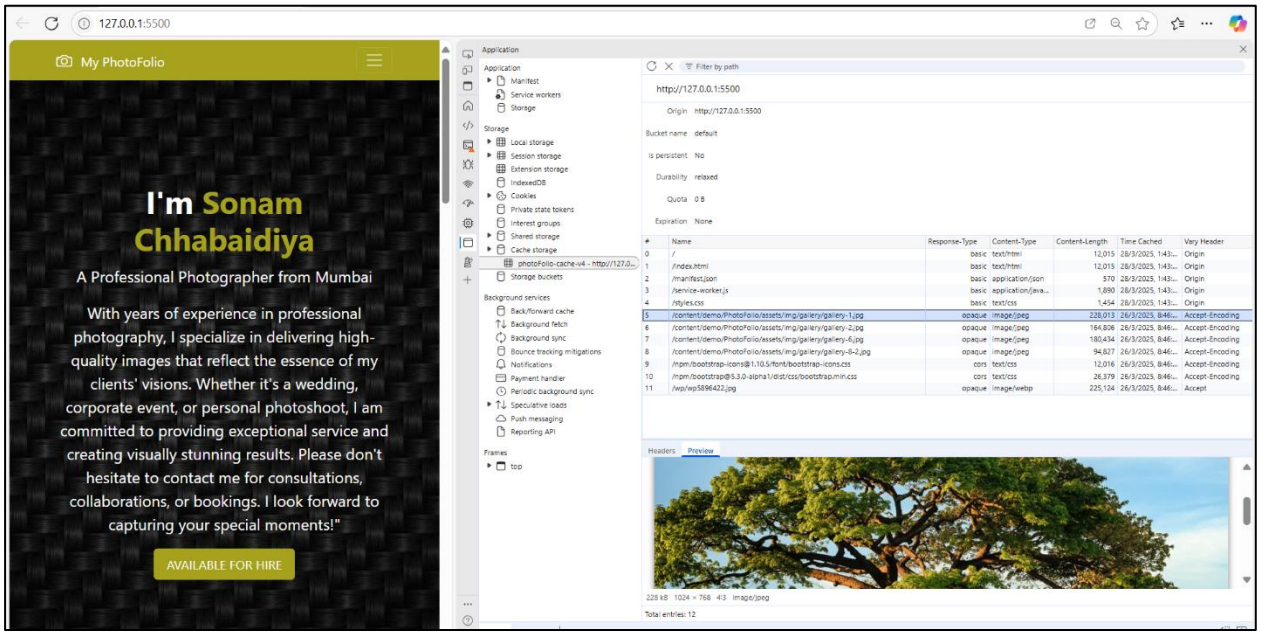
```
const CACHE_NAME = "photoFolio-cache-v4";
const urlsToCache = [
   "/",
   "/index.html",
   "/styles.css",
   "/manifest.json",
   "/service-worker.js",
   "https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css",
   "https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css"
];

// Install event - Cache static assets
self.addEventListener("install", (event) => {
   event.waitUntil(
      caches.open(CACHE_NAME).then((cache) => {
         console.log("Opened cache and adding assets...");

  return cache.addAll(urlsToCache);
      })
   );
});
```

```
// Fetch event - Cache external images dynamically
self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      return response || fetch(event.request).then((fetchResponse) => {
        let requestUrl = event.request.url;

        // Cache only images (JPG, PNG, etc.)
        if (requestUrl.match(/\.(jpg|jpeg|png|gif|webp)$/)) {
          return caches.open(CACHE_NAME).then((cache) => {
            cache.put(event.request, fetchResponse.clone());
            return fetchResponse;
          });
        }

        return fetchResponse;
      });
    })
  );
});

// Activate event - Cleanup old caches
self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cache) => {
          if (cache !== CACHE_NAME) {
            console.log("Deleting old cache:", cache);
            return caches.delete(cache);
          }
        })
      );
    })
  );

});
```

OUTPUT:


127.0.0.1:5500
My PhotoFolio
I'm Sonam Chhabaidiya
A Professional Photographer from Mumbai
With years of experience in professional photography, I specialize in delivering high-quality images that reflect the essence of my clients' visions. Whether it's a wedding, corporate event, or personal photoshoot, I am committed to providing exceptional service and creating visually stunning results. Please don't hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE
Application
Service workers
http://127.0.0.1:5500/
Network requests
Update
Unregister
Push
Sync
Periodic sync
Update Cycle
Service workers from other origins
See all registrations


127.0.0.1:5500
My PhotoFolio
I'm Sonam Chhabaidiya
A Professional Photographer from Mumbai
With years of experience in professional photography, I specialize in delivering high-quality images that reflect the essence of my clients' visions. Whether it's a wedding, corporate event, or personal photoshoot, I am committed to providing exceptional service and creating visually stunning results. Please don't hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE
Application
http://127.0.0.1:5500
Origin
Bucket name
Is persistent
Durability
Quota
Expiration
No cache entry selected
Select a cache entry above to preview
Total entries: 12


127.0.0.1:5500
My PhotoFolio
I'm Sonam Chhabaidiya
A Professional Photographer from Mumbai
With years of experience in professional photography, I specialize in delivering high-quality images that reflect the essence of my clients' visions. Whether it's a wedding, corporate event, or personal photoshoot, I am committed to providing exceptional service and creating visually stunning results. Please don't hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE
Application
Manifest
Service workers
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigations
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
http://127.0.0.1:5500
Origin http://127.0.0.1:5500
Bucket name default
Is persistent No
Durability relaxed
Quota 0 B
Expiration None
Headers
Preview

EXPERIMENT NO. 9

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

Theory:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.
- Service workers make extensive use of promises, so if you're new to promises, then you should stop reading this and check out Promises, an introduction.

Fetch Event

You can track and manage page network traffic with this event. You can check existing cache, manage “cache first” and “network first” requests and return a response that you want.

Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted external URL, this is called “networkFirst”.

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new response will be cached and returned. But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

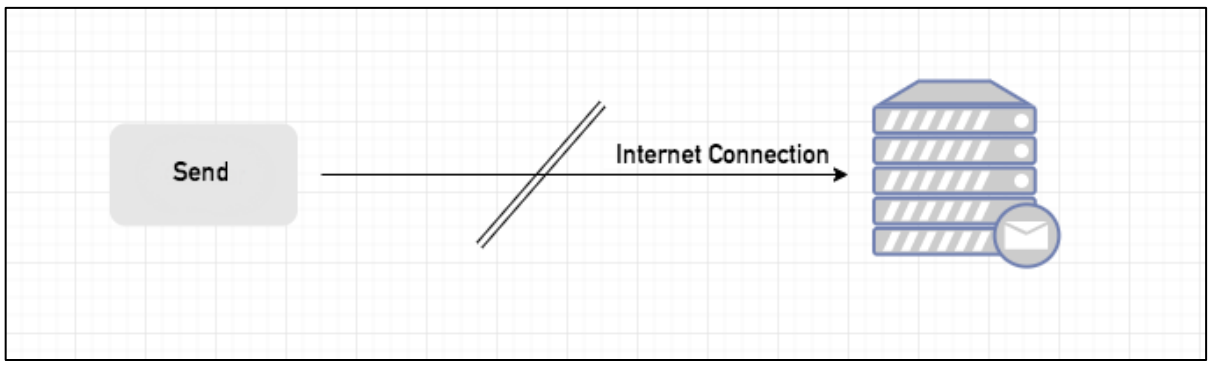
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

    if (url.origin === location.origin) {
        event.respondWith(cacheFirst(req));
    }
    else {
        event.respondWith(networkFirst(req));
    }
});

async function cacheFirst(req) {
    return await caches.match(req) || fetch(req);
}

async function networkFirst(req) {
    const cache = await caches.open("pwa-dynamic");
    try {
        const res = await fetch(req);
        cache.put(req, res.clone());
        return res;
    } catch (error) {
        const cachedResponse = await cache.match(req);
        return cachedResponse || await caches.match("./noconnection.json");
    }
}
```

Sync Event

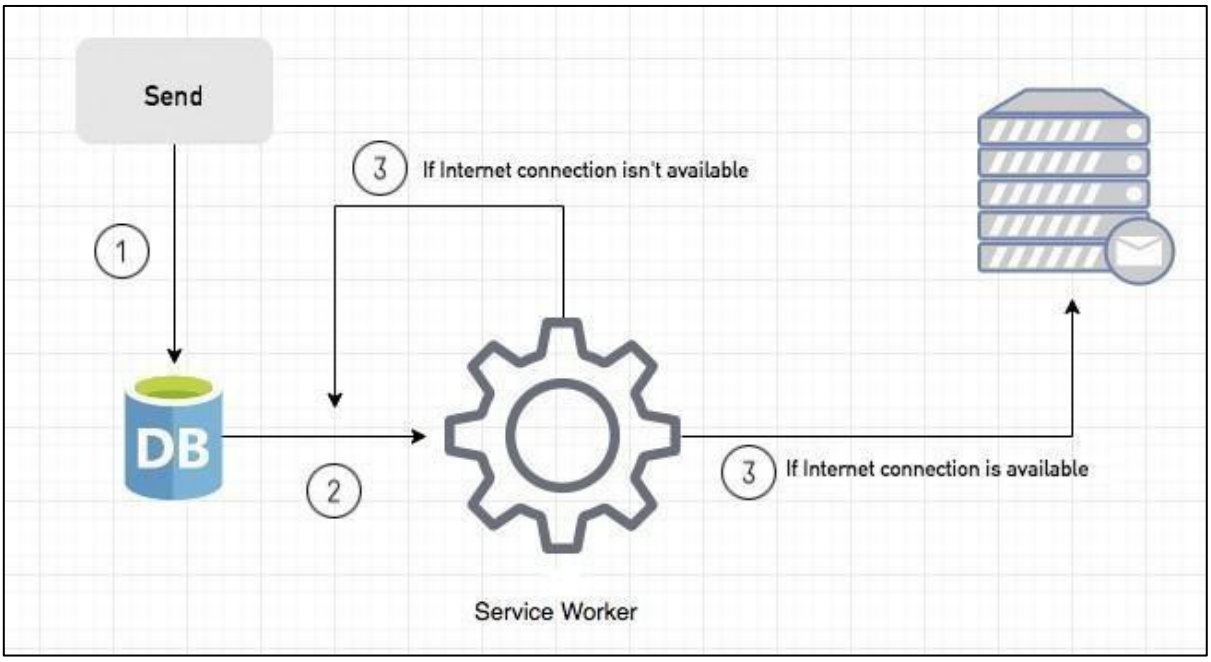
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are writing e-mail content and we didn't realize it. When completing the writing, we click the send button.

Here is a job for the Background Sync.

The following view shows the classical process of sending email to us. If the Internet Connection is broken, we can't send any content to Mail Server.



Here, you can create any scenario for yourself. A sample is in the following for this case.



1. When we click the “send” button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server.

If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

You can see the working process within the following code block.

Event Listener for Background Sync Registration

```
document.querySelector("button").addEventListener("click", async () => {
    var swRegistration = await navigator.serviceWorker.register("sw.js");
    swRegistration.sync.register("helloSync").then(function () {
        console.log("helloSync success [main.js]");
    });
});
```

Event Listener for sw.js

```
self.addEventListener('sync', event => {
    if (event.tag == 'helloSync') {
        console.log("helloSync [sw.js]");
    }
});
```

Push Event

This is the event that handles push notifications that are received from the server. You can apply any method with received data.

We can check in the following example.
“Notification.requestPermission();” is the necessary line to show notification to the user. If you don’t want to show any notification, you don’t need this line.

In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has “method” and “message” properties. If the method value is “pushMessage”, we open the information notification with the “message” property.

```
self.addEventListener('push', event => {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method === "pushMessage") {
            event.waitUntil(self.registration.showNotification("Test App", {
                body: data.message
            }));
        }
    }
});
```

You can use Application Tab from Chrome Developer Tools for testing push notification.

Code:

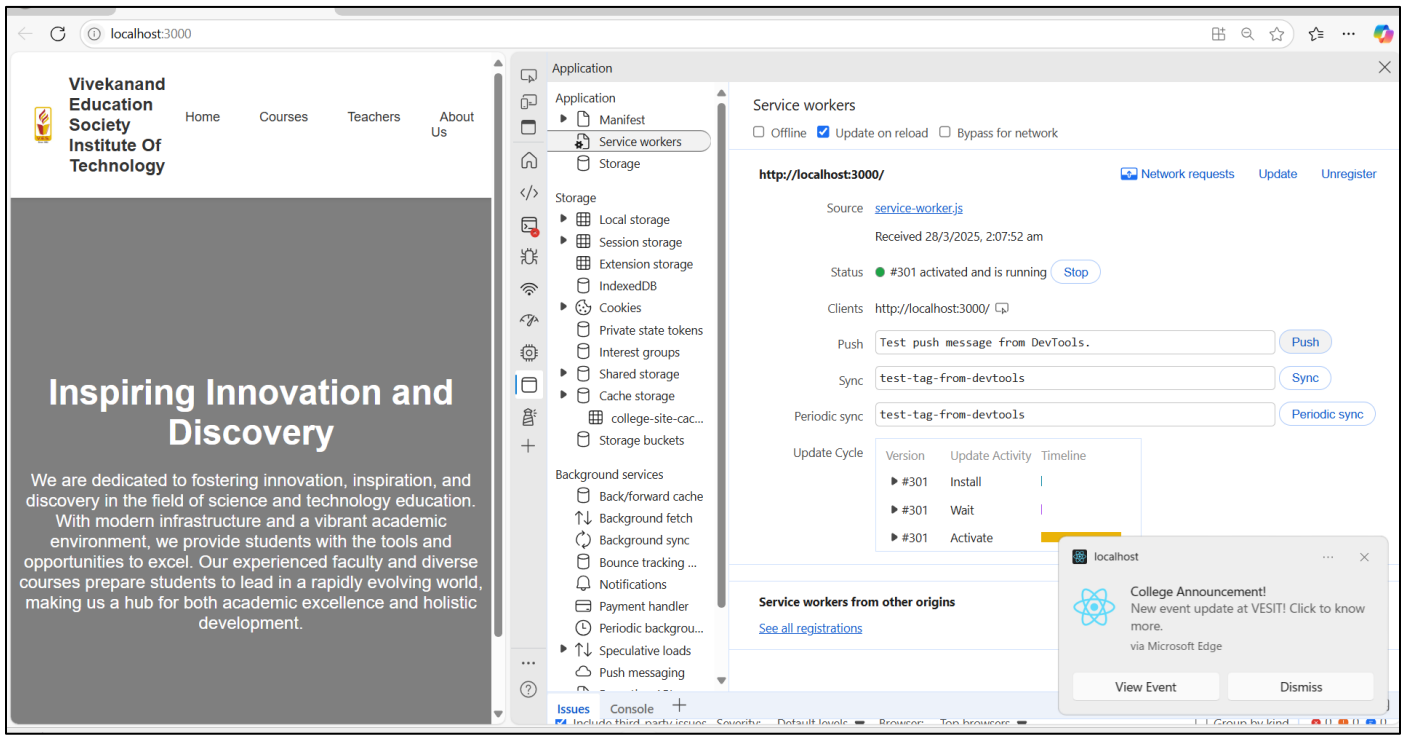
1. Push Notification code:

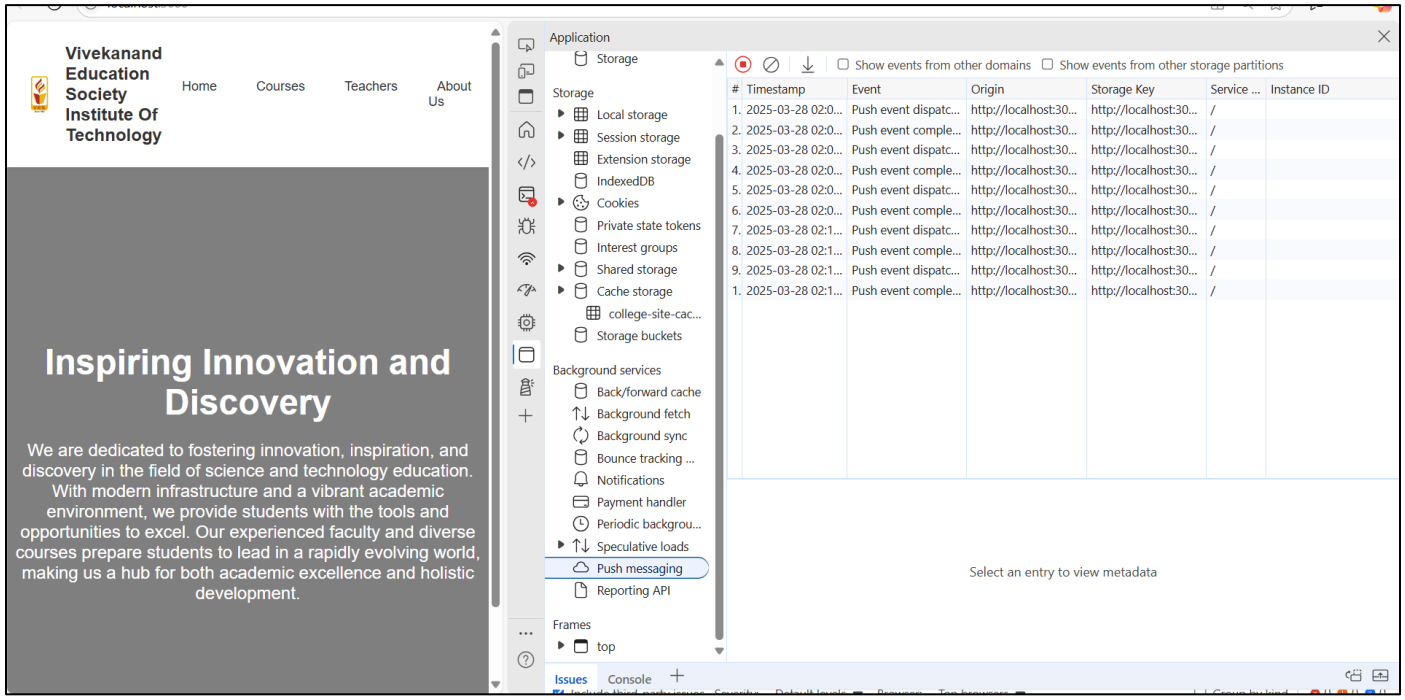
```
/ Push Notification event - College Announcements
self.addEventListener("push", async (event) => {
    if (!self.registration || Notification.permission !== "granted") {
        console.warn("Push notification permission not granted.");
        return;
    }

    const options = {
        body: "New event update at VESIT! Click to know more.",
        icon: "/logo192.png",
        badge: "/logo192.png",
        actions: [
            { action: "view", title: "View Event" },
            { action: "close", title: "Dismiss" }
        ]
    };

    event.waitUntil(
        self.registration.showNotification("College Announcement!", options)
            .catch(error => console.error("Error showing notification:", error))
    );
});

// Handle Notification Click - Redirect to Event Page
self.addEventListener("notificationclick", (event) => {
    event.notification.close();

    if (event.action === "view") {
        event.waitUntil(
            clients.matchAll({ type: "window", includeUncontrolled: true }).then((clientList) => {
                if (clientList.length > 0) {
                    return clientList[0].focus();
                }
                return clients.openWindow("/events").catch((error) => {
                    console.error("Failed to open events page:", error);
                });
            })
        );
    }
});
```

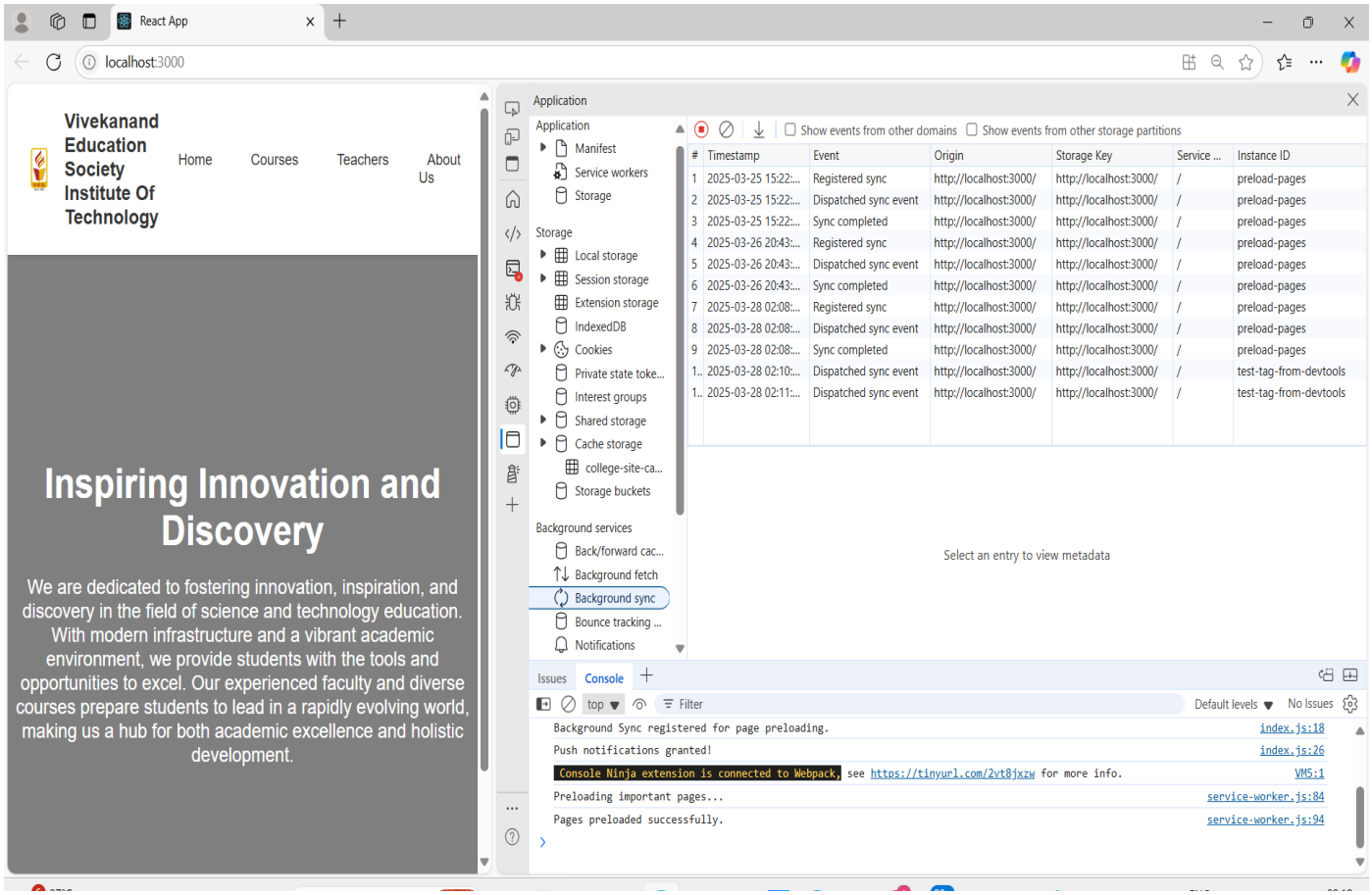
2. Fetch and Sync Code:

```
// Fetch event - Serve files from cache & dynamically cache new requests
self.addEventListener("fetch", (event) => {
    if (event.request.method !== "GET") return; // Handle only GET requests

    event.respondWith(
        caches.match(event.request).then((cachedResponse) => {
            if (cachedResponse) {
                return cachedResponse; // Return cached response if available
            }

            return fetch(event.request)
                .then((fetchResponse) => {
                    return caches.open(CACHE_NAME).then((cache) => {
                        cache.put(event.request, fetchResponse.clone());
                        return fetchResponse;
                    });
                })
                .catch(() => {
                    return new Response("You are offline. Please check your connection.", {
                        status: 503,
                        headers: { "Content-Type": "text/plain" }
                    });
                });
        })
    );
});

// Background Sync - Preload Important Pages
self.addEventListener("sync", (event) => {
    if (event.tag === "preload-pages") {
        event.waitUntil(preloadImportantPages());
    }
});

async function preloadImportantPages() {
    console.log("Preloading important pages...");
    const pages = ["/courses", "/teachers", "/about"];
    try {
        const cache = await caches.open(CACHE_NAME);
        await Promise.all(pages.map(async (page) => {
            const response = await fetch(page);
            if (response.ok) {
                cache.put(page, response.clone());
            }
        }));
        console.log("Pages preloaded successfully.");
    } catch (error) {
        console.error("Preloading failed:", error);
    }
}
```

OUTPUT:-


localhost:3000
Vivekanand Education Society Institute Of Technology
Home
Courses
Teachers
About Us
Inspiring Innovation and Discovery
We are dedicated to fostering innovation, inspiration, and discovery in the field of science and technology education. With modern infrastructure and a vibrant academic environment, we provide students with the tools and opportunities to excel. Our experienced faculty and diverse courses prepare students to lead in a rapidly evolving world, making us a hub for both academic excellence and holistic development.
Application
Service workers
Storage
Background services
Service workers
Offline
Update on reload
Bypass for network
http://localhost:3000/
Network requests
Update
Unregister
Source
service-worker.js
Received 28/3/2025, 2:07:52 am
Status
#301 activated and is running
Stop
Clients
http://localhost:3000/
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#301 Install
#301 Wait
#301 Activate
Service workers from other origins
See all registrations
localhost
College Announcement!
New event update at VESIT! Click to know more.
via Microsoft Edge
View Event
Dismiss
Issues
Console


Vivekanand Education Society Institute Of Technology
Home
Courses
Teachers
About Us
Inspiring Innovation and Discovery
We are dedicated to fostering innovation, inspiration, and discovery in the field of science and technology education. With modern infrastructure and a vibrant academic environment, we provide students with the tools and opportunities to excel. Our experienced faculty and diverse courses prepare students to lead in a rapidly evolving world, making us a hub for both academic excellence and holistic development.
Application
Storage
Show events from other domains
Show events from other storage partitions
Timestamp Event Origin Storage Key Service ... Instance ID
Select an entry to view metadata
Push messaging
Reporting API
Frames
top
Issues
Console


React App
localhost:3000
Vivekanand Education Society Institute Of Technology
Home
Courses
Teachers
About Us
Inspiring Innovation and Discovery
We are dedicated to fostering innovation, inspiration, and discovery in the field of science and technology education. With modern infrastructure and a vibrant academic environment, we provide students with the tools and opportunities to excel. Our experienced faculty and diverse courses prepare students to lead in a rapidly evolving world, making us a hub for both academic excellence and holistic development.
Application
Manifest
Service workers
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state toke...
Interest groups
Shared storage
Cache storage
college-site-ca...
Storage buckets
Background services
Back/forward cac...
Background fetch
Background sync
Bounce tracking ...
Notifications
Show events from other domains
Show events from other storage partitions
Select an entry to view metadata
Issues
Console
Filter
Default levels
No Issues
Background Sync registered for page preloading.
Push notifications granted!
Console Ninja extension is connected to Webpack, see https://tinyurl.com/2vt8jxzw for more info.
Preloading important pages...
Pages preloaded successfully.

EXPERIMENT NO: - 10

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

Theory: -

GitHub Pages: Static Website Hosting Made Simple

GitHub Pages is a free hosting service that allows users to publish **public webpages directly from a GitHub repository**. It is particularly useful for **static websites, project documentation, and blogs**.

Key Features

- **Jekyll Integration**: Built-in support for Jekyll enables easy blogging.
- **Custom Domains**: Allows users to configure their own URLs.
- **Automatic Page Deployment**: Simply push your changes to the repository, and the updates go live.

Why Choose GitHub Pages?

- **Completely Free**: No hosting charges.
- **Seamless GitHub Integration**: Works directly with your repositories.
- **Quick Setup**: Just create a repository, push your files, and your site is live.

Who Uses GitHub Pages?

Companies like **Lyft, CircleCI, and HubSpot** use GitHub Pages for their documentation and static sites. It is widely adopted, appearing in **775 company stacks and 4,401 developer stacks**.

Pros & Cons

Pros

- Familiar interface for GitHub users.

- Simple deployment via the gh-pages branch.
- Supports custom domains with easy DNS configuration.

Cons

- Repositories need to be public unless you have a paid plan.
- Limited HTTPS support for custom domains (expected to improve).
- Jekyll plugins have limited support.

Firebase: A Full-Featured Real-Time Backend

Firebase is a cloud-based **real-time application platform** developed by Google. It enables developers to build **dynamic, collaborative applications** with ease.

Key Features

- **Real-Time Database**: Automatically syncs data across all connected clients.
- **Cloud-Based Storage**: JSON-based storage accessible via REST APIs.
- **Scalable Infrastructure**: Works well with existing services and scales automatically.
- **Authentication & Cloud Messaging**: Secure login and push notifications.

Why Choose Firebase?

- **Instant Backend Setup**: No need to build a separate backend.
- **Fast & Responsive**: Real-time data synchronization.
- **Built-in HTTPS**: Free SSL certificates for custom domains.

Who Uses Firebase?

Companies like **Instacart, 9GAG, and Twitch** rely on Firebase for their backend needs. Firebase is widely adopted, appearing in **1,215 company stacks and 4,651 developer stacks**.

Pros & Cons

Pros

- **Hosted by Google**, ensuring reliability and security.
- **Comes with authentication, messaging, and real-time database services.**
- **Free HTTPS support** for all custom domains.

Cons

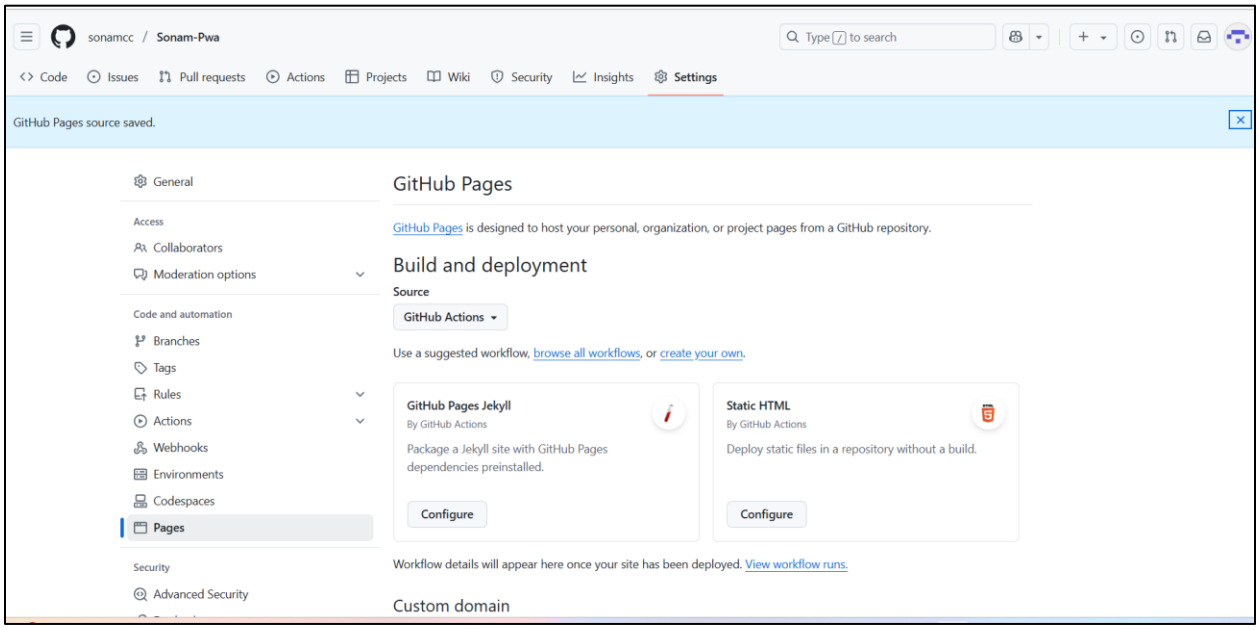
- **Limited Free Plan**: 10 GB of data transfer per month (can be mitigated with a CDN).
- **Command-Line Deployment**: No GUI for hosting.
- **No Built-in Static Site Generator Support**: Unlike GitHub Pages, Firebase doesn't natively support static site generators like Jekyll.

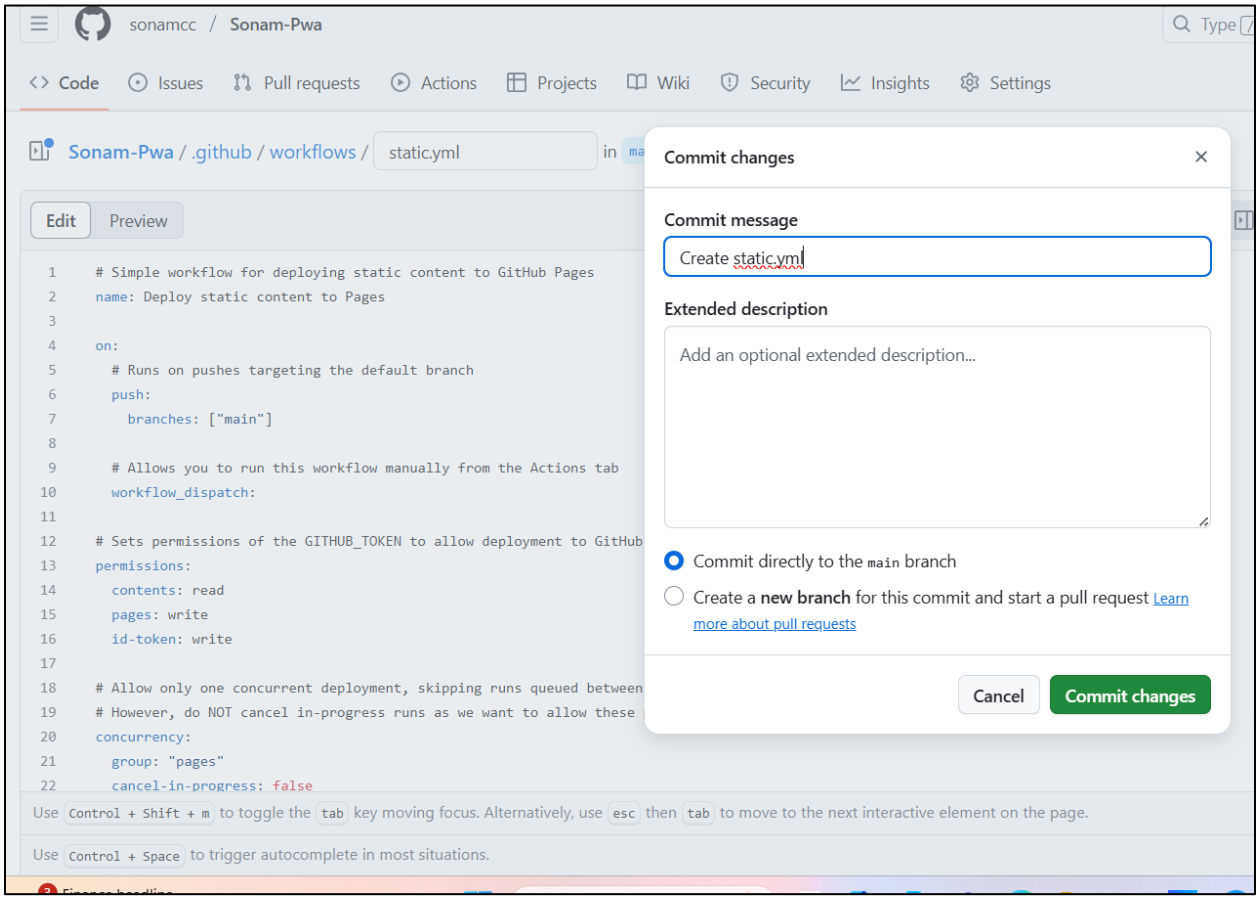
Link to our GitHub repository:

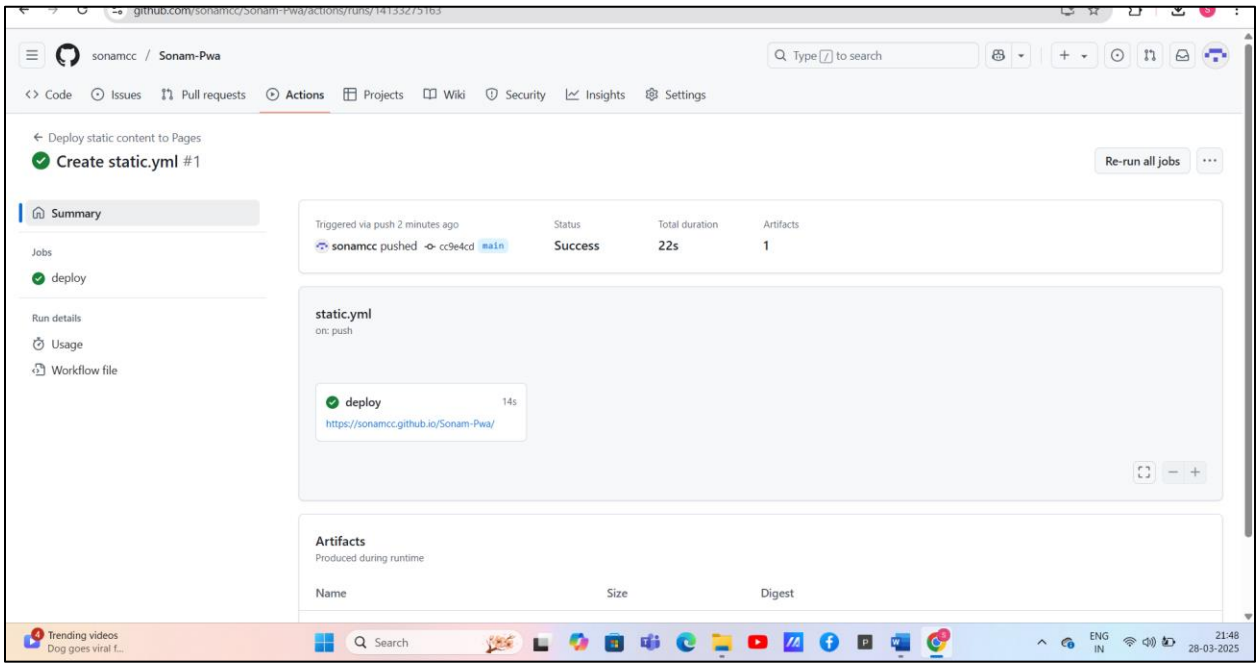
Sonam-Pwa

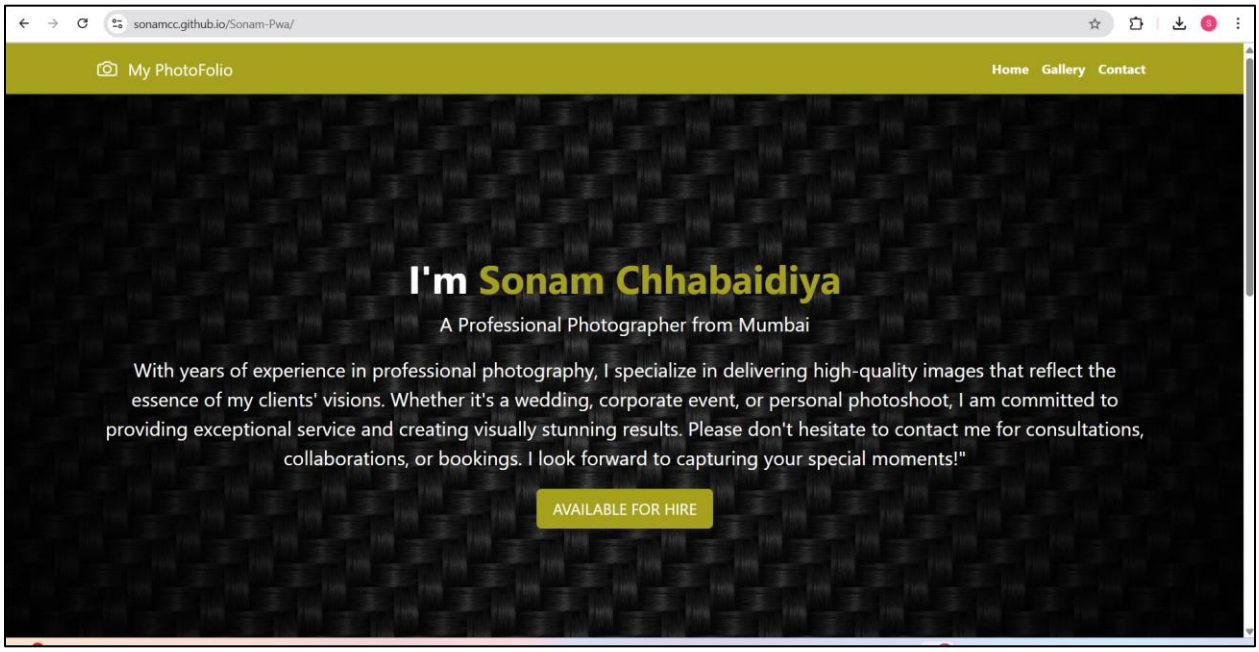
Hosted link:
https://sonamcc.github.io/Sonam-Pwa/

Github Screenshot:




sonamcc / Sonam-Pwa
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Sonam-Pwa / .github / workflows / static.yml
Edit
Preview
Simple workflow for deploying static content to GitHub Pages
name: Deploy static content to Pages
on:
Runs on pushes targeting the default branch
push:
branches: ["main"]
Allows you to run this workflow manually from the Actions tab
workflow_dispatch:
Sets permissions of the GITHUB_TOKEN to allow deployment to GitHub
permissions:
contents: read
pages: write
id-token: write
Allow only one concurrent deployment, skipping runs queued between
However, do NOT cancel in-progress runs as we want to allow these
concurrency:
group: "pages"
cancel-in-progress: false
Commit changes
Commit message
Create static.yml
Extended description
Add an optional extended description...
Commit directly to the main branch
Create a new branch for this commit and start a pull request Learn more about pull requests
Cancel
Commit changes
Use Control + Shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.
Use Control + Space to trigger autocomplete in most situations.


sonamcc / Sonam-Pwa
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Deploy static content to Pages
Create static.yml #1
Re-run all jobs
Summary
Jobs
deploy
Run details
Usage
Workflow file
Triggered via push 2 minutes ago
sonamcc pushed cc9e4cd main
Status
Success
Total duration
22s
Artifacts
1
static.yml
on: push
deploy
14s
https://sonamcc.github.io/Sonam-Pwa/
Artifacts
Produced during runtime
Name
Size
Digest


sonamcc.github.io/Sonam-Pwa/
My PhotoFolio
Home Gallery Contact
I'm Sonam Chhabaidiya
A Professional Photographer from Mumbai
With years of experience in professional photography, I specialize in delivering high-quality images that reflect the essence of my clients' visions. Whether it's a wedding, corporate event, or personal photoshoot, I am committed to providing exceptional service and creating visually stunning results. Please don't hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE

Experiment 11

AIM: - To use google Lighthouse PWA Analysis Tool to test the PWA functioning.

Theory: - Google Lighthouse is an open-source tool that audits web applications based on multiple key parameters, including performance, accessibility, Progressive Web App (PWA) implementation, and best practices. It provides a detailed, automated report that helps developers optimize their websites efficiently. Unlike traditional manual audits, which can take days or even weeks, Lighthouse generates insights within minutes.
One of the key advantages of Lighthouse is its ease of use—no complex setup is required. Simply run it on a webpage or provide a URL, and it will generate an extensive performance report.

Key Features and Audit Metrics
Lighthouse can audit both desktop and mobile versions of a webpage. The core evaluation criteria include:

1. Performance
This metric measures how efficiently a webpage loads and displays content.
Key factors influencing the performance score include:

- Page load speed – How quickly the page becomes visible to the user.
- First Contentful Paint (FCP) – The time taken for the first piece of content to appear.
- Largest Contentful Paint (LCP) – The time taken for the main content to fully load.
- Cumulative Layout Shift (CLS) – Measures how visually stable a page is (i.e., avoiding unexpected shifts in content).
- Time to Interactive (TTI) – The time it takes for the page to become fully functional.

Lighthouse assigns a score from 0 to 100 based on percentile rankings, where:

- 100 → Top 2% of websites (98th percentile)
- 50 → Around the 75th percentile
- Lower scores → Indicate areas that need optimization

2. Progressive Web App (PWA) Score (Mobile)

With the rise of PWAs, modern web applications aim to provide a native app-like experience. Lighthouse evaluates the PWA implementation based on Google's Baseline PWA Checklist, which includes:

- Service Worker implementation – Ensuring offline support and background synchronization.
- App Manifest compliance – Providing metadata for better mobile integration.
- Viewport configuration – Optimizing mobile responsiveness.
- Performance in script-disabled environments – Ensuring the page functions even when JavaScript is disabled.

A high PWA score indicates that the application meets essential PWA criteria and provides an app-like user experience.

3. Accessibility

Accessibility ensures that web applications are usable by individuals with disabilities. Lighthouse audits a webpage based on:

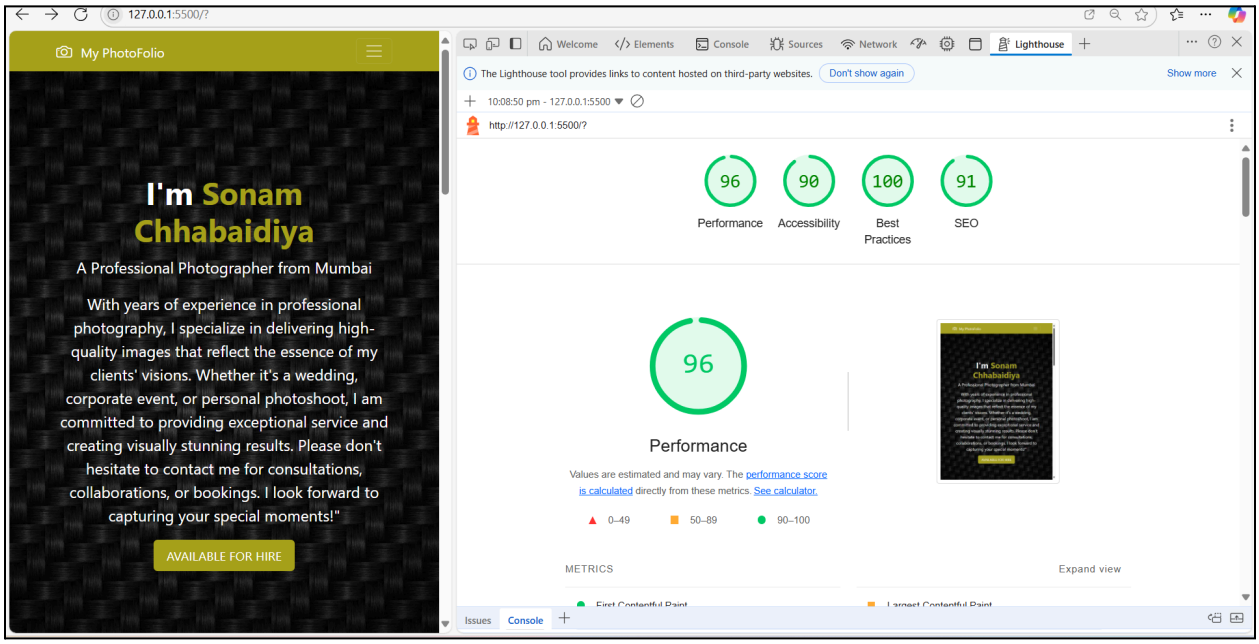
- ARIA attributes – Enhancing accessibility through attributes like aria-required.
- Text alternatives for media – Ensuring audio and visual content is accessible.
- Semantic HTML – Proper use of , , , and other elements that improve screen-reader compatibility.

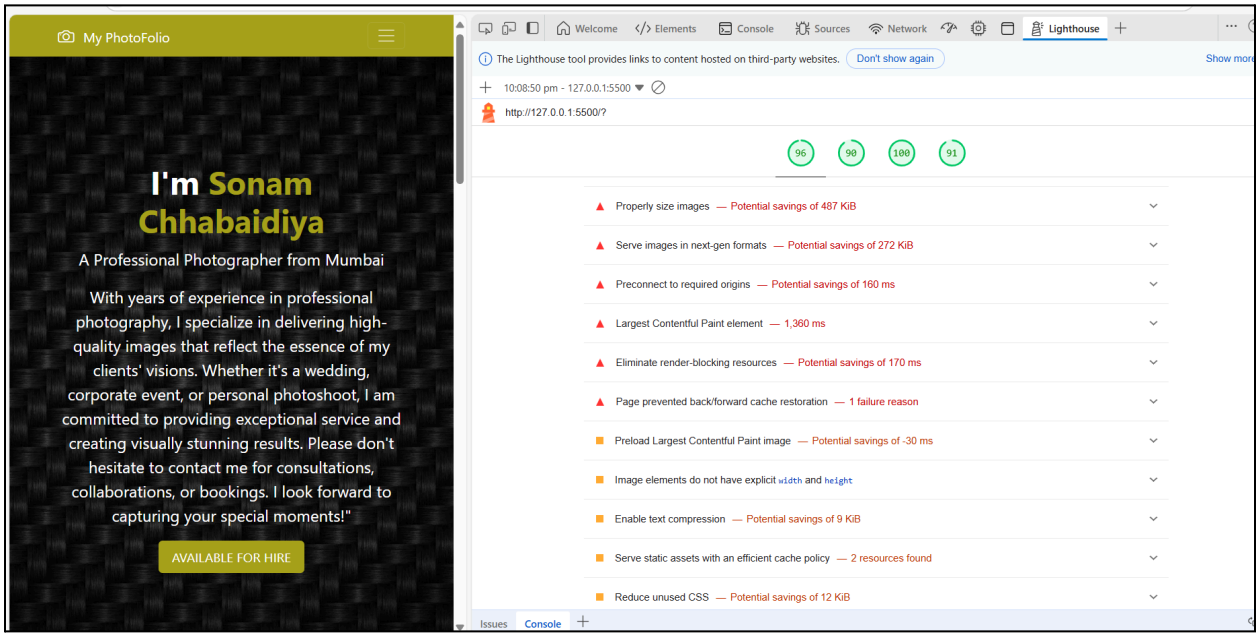
Unlike other metrics, accessibility checks follow a pass/fail approach—if a necessary feature is missing, it significantly impacts the score. A higher accessibility score ensures inclusivity for users with visual or cognitive impairments

Manifest.json

```
{
    "name": "PhotoFolio",
    "short_name": "PhotoFolio",
    "start_url": "/",
    "display": "standalone",
    "background_color": "#090909",
    "theme_color": "#a7a21d",
    "icons": [
      {
        "src":
"https://bootstrapmade.com/content/demo/PhotoFolio/assets/img/gallery/gall
ery-1.jpg",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src":
"https://bootstrapmade.com/content/demo/PhotoFolio/assets/img/gallery/gall
ery-2.jpg",
        "sizes": "512x512",
        "type": "image/png"
      }
    ]
  }
```

Output:


My PhotoFolio
I'm Sonam Chhabaidiya
A Professional Photographer from Mumbai
With years of experience in professional photography, I specialize in delivering high-quality images that reflect the essence of my clients' visions. Whether it's a wedding, corporate event, or personal photoshoot, I am committed to providing exceptional service and creating visually stunning results. Please don't hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE
Lighthouse
96
90
100
91
Performance
Accessibility
Best Practices
SEO
Performance
METRICS
Expand view


My PhotoFolio
I'm Sonam Chhabaidiya
A Professional Photographer from Mumbai
With years of experience in professional photography, I specialize in delivering high-quality images that reflect the essence of my clients' visions. Whether it's a wedding, corporate event, or personal photoshoot, I am committed to providing exceptional service and creating visually stunning results. Please don't hesitate to contact me for consultations, collaborations, or bookings. I look forward to capturing your special moments!"
AVAILABLE FOR HIRE
Lighthouse
Properly size images — Potential savings of 487 KiB
Serve images in next-gen formats — Potential savings of 272 KiB
Preconnect to required origins — Potential savings of 160 ms
Largest Contentful Paint element — 1,360 ms
Eliminate render-blocking resources — Potential savings of 170 ms
Page prevented back/forward cache restoration — 1 failure reason
Preload Largest Contentful Paint image — Potential savings of -30 ms
Image elements do not have explicit width and height
Enable text compression — Potential savings of 9 KiB
Serve static assets with an efficient cache policy — 2 resources found
Reduce unused CSS — Potential savings of 12 KiB