

A Project Report

on

LineUp Polo

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

B.Tech AI & ML



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under The Supervision of
Supervisor: Dr.Sampath Kumar
Designation**

Submitted By

**Ashwani Kumar Singh
18SCSE1180047/18021180046**

**Ritik Kumar Srivastava
18SCSE1180051/18021180050**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
May, 2022**



SCHOOL OF COMPUTING SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled “**LINEUP POLO**” in partial fulfillment of the requirements for the award of the B.tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of January, 2022 to May 2022, under the supervision of Dr.Sampath Kumar, Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Ashwani Kumar Singh/ 18SCSE1180047
Ritik Kumar Srivastava/ 18SCSE1180051

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name - Dr. Sampath
Kumar

Designation - Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Ashwani Kumar Singh / 18SCSE1180047 and Ritik Kumar Srivastava / 18SCSE1180051 has been held on _____ and his/her work is recommended for the award of B. tech in Computer Science & Engineering.

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

Date: May, 2022

Place: Greater Noida

Acknowledgement

We take this opportunity to express our sincere gratitude to **Prof. Dr.Sampath Kumar Professor**, Department of CSE **GALGOTIAS UNIVERSITY**, GREATER NOIDA. Deep Knowledge & keen interest of our supervisor in the field of “*Mobile Application*” to carry out this project. His endless patience, scholarly guidance, strong motivation, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude other faculty member and the staff of CSE department of **GALGOTIAS UNIVERSITY**, GREATER NOIDA to finish our project.

We would like to thank our entire course mate in **GALGOTIAS UNIVERSITY**, GREATER NOIDA, who took part in this discuss while completing the course work

Abstract

Problem Stated

Now a days lots of tournaments are going on. To track the score of our favorite team we have to navigate to a different website. But there is no such platform on which manager or client can organize the tournaments. There must be a single platform for players must see their all-tournament, live scores, match history, etc.

Managers can create the tournaments and invite will be added to the calendar.

Spectators can see the progress of their favorite team and their favorite clubs.

Problem Solution

Lineup polo is a single platform where Players can enter teams, view results, and check out who is on the naughty step with yellow cards!.

Managers automatically create a schedule that works for everyone. Through the app, they can easily score games or appoint someone to do it for them.

Spectators can't get there in person, scores from matches can be viewed live. You can also see tournament progression and player and club profiles.

Tools and Technology

Techs used are Flutter, Firebase, Vs Code, Android Studio, Adobe XD, Postman.

Result

It will be cross-platform application with beautiful UI with multiple features and with different roles for ex - players, managers, spectators etc.

Conclusion

The app is in sagging soon it will be live on Google Play Store and App Store.

Table of Contents

Title	Page No.
Candidates Declaration	I
Acknowledgement	II
Abstract	III
Contents	IV
List of Table	V
List of Figures	VI
Acronyms	VII
Chapter 1 Introduction	10
1.1 Introduction	
1.2 Tool and Technology Used	11-12
Chapter 2 Literature Survey/Project Design	13-15
1) Project design	16-17
2) Required tools	18-21
3) App design	22-28
Chapter 3 Functionality/Working of Project	
1) App module	29-30
2) Code	31-52
Chapter 4 Results and Discussion	53
Chapter 5 Conclusion and Future Scope	53
Reference	54-55
Publication/Copyright/Product	56

List of Table

S.No	Enrollmen t Number	Admissio n Number	Student Name	Degree / Branch	Sem
1.	18021180046	18SCSE1180047	Ashwani Kumar Singh	B.Tech/CS E	8
2.	18021180050	18SCSE1180051	Ritik Kumar Srivastava	B.Tech/CS E	8

List of Figures

S.No.	Title	Page No.
1	MVVM fig	18
2	Splash Screen design	22
3	Sign up Screen design	23
4	Match Screen design	24
6	Tournament Screen design	25
7	Club Screen design	26
8	Notification Screen design	27
9	Profile Screen design	28

Acronyms

app	Application
db	Database
IDE	Integrated development environment
SDK	software development kit
UI	User Interface
FCM	Firebase Cloud Messaging
JWT	JSON Web Token

CHAPTER-1

Introduction

Lineup polo is tournament schedules are automatically created for managers. Once you have created your account, all your games for every tournament will be here, in one place. You can enter teams, view live results, and check out who is on the naughty step with yellow cards! Lineup polo will organize all the polo events and tournaments. Lineup collects players' availability and then, using the advanced algorithm, automatically creates a schedule that works for everyone. Through the app, they can easily score games or appoint someone to do it for them. The lineup shows you all games local to you, as well as all the information you need to attend. If you can't get there in person, scores from matches can be viewed live. In Lineup Polo you can also see tournament progression and player and club profiles. Entertainers load their availability into the calendar.

Team captains enter the team into the tournament through Lineup.

Lineup's algorithm automatically creates the schedules that work for everyone.

Tools and Technology

1) Flutter Framework

Flutter is Google's free and open-source UI framework for creating native mobile applications. Released in 2017, Flutter allows developers to build mobile applications for both iOS and Android with a single codebase and programming language. This capability makes building iOS and Android apps simpler and faster.

2) Firebase Database

Firebase is a toolset to “build, improve, and grow your app”, and the tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don't really want to build, because they'd rather be focusing on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and scale with little to no effort on the part of the developer.

3) VS Code Editor

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

4) Android Studio

It is widely used because it has below features:

- a. Gradle-based build support
- b. Android-specific refactoring and quick fixes
- c. Lint tools to catch performance, usability, version compatibility and other problems

- d. Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- e. Android Virtual Device (Emulator) to run and debug apps in the Android studio.

5) **Adobe XD**

Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. It is available for macOS and Windows, although there are versions for iOS and Android to help preview the result of work directly on mobile devices.

Literature Survey

CROSS PLATFORM APPS AND NATIVE APPS

CROSS-PLATFORM APPS AND NATIVE APPS Nowadays 99.6% of phones run on either IOS or Android. Companies now treat mobile apps as a channel for raising their brand and advertise more for the marketing purpose of the apps. A native mobile app is an application that only looks for a particular operating system by using their IDE and SDK. Native apps have the ability to use device-specific hardware and software and can provide quality with huge performance rates. The advantages of native mobile apps include – High performance, Ultimate user experience, and Greater app store visibility. A cross-platform application is a mobile app that is compatible with multiple operating systems and can therefore run on any smartphone or tablet. The advantages of the cross-platform mobile apps include:

	Native	Cross-Platform
Multiple OS Support	No	Yes
User Interface Quality	High	Medium to High
Cost Of Ownership	High	Medium
Application Update	Native Market	Native Market
Application Maintenance	High	Medium

- Shorter Development time: It will need development or coding for one time and it will support the entire applications platform.
- Cost- effectiveness.

Widget Lifecycle:

Everything in Flutter is a Widget, so before knowing about Lifecycle, we should know about Widgets in Flutter.

There are two types of widgets in Flutter.

- **Stateless Widgets.**
 - **Stateful Widgets.**
- **Stateless Widgets:** Stateless Widgets in Flutter are those widgets whose state once created cannot be changed, it becomes immutable like on variables, buttons, icons, etc., or any state that cannot be changed on the app to retrieve data. Returns a widget by overwriting the build method. We use it when the UI relies on the information inside the object itself.
- **Stateful Widget:** A Stateful widget maintains data and responds to whatever the data does inside the application. It is a mutable widget, so it is drawn multiple times in its lifetime.

We use this when the user dynamically updates the application screen. This is the most important of all the widgets, as it has a state widget, everyone knows that something has been updated on our screen.

Widget Lifecycle Methods:

- **createState**
 - **initState()**
 - **didChangeDependencies()**
 - **build()**
 - **didUpdateWidget()**
 - **setState()**
 - **deactivate()**
 - **dispose()**
-
- **initState():** Flutter's `initState()` method is the first method that is used while creating a stateful class, here we can initialize variables, data, properties, etc. for any widget.
 - **createState():** When we create a stateful widget, our framework calls a `createState()` method and it must be overridden.
 - **build():** The `build` method is used each time the widget is rebuilt. This can happen either after calling `initState`, `didChangeDependencies`, `didUpdateWidget`, or when the state is changed via a call to `setState`.
 - **didChangeDependencies():** This method is called immediately after *initState* and when dependency of the State object changes via `InheritedWidget`.
 - **didUpdateWidget():** This method is called whenever the widget configuration changes. A typical case is when a parent passes some variable to the `children()` widget via the constructor.
 - **deactivate():** It is used when the state is removed from the tree but before the current frame change can be re-inserted into another part of the tree.
 - **dispose():** We use this method when we remove permanently like should release resource created by an object like stop animation.

Project Design

Model–View–ViewModel (MVVM)

MVVM is useful to move business logic from view to ViewModel and Model. ViewModel is the mediator between View and Model which carry all user events and return back the result.

The key benefit is allowing true separation between the View and Model and the efficiency that you gain from having that. What it means in real terms is that when your model needs to change, it can be changed easily without the view needing to and vice-versa.

There are three key things that flow out of applying MVVM –

- **Maintainability:** - The presentation layer and the logic are loosely coupled, due to this code is easily maintainable and reusable. As the code base will increase over the course of time, this will help you to distinguish between them.
- **Testability:** - The ViewModel is easier to unit test than code-behind or event-driven code.
- **Extensibility:** - This architecture gives you assurance which enables code to get extensible over the period of time. but keep in mind it's also over job to keep component reusable.

➤ **Model**

The model represents a single source of truth that carries the real-time fetch data or database-related queries.

This layer can contain business logic, code validation, etc. This layer interacts with ViewModel for local data or for real-time. Data are given in response to ViewModel.

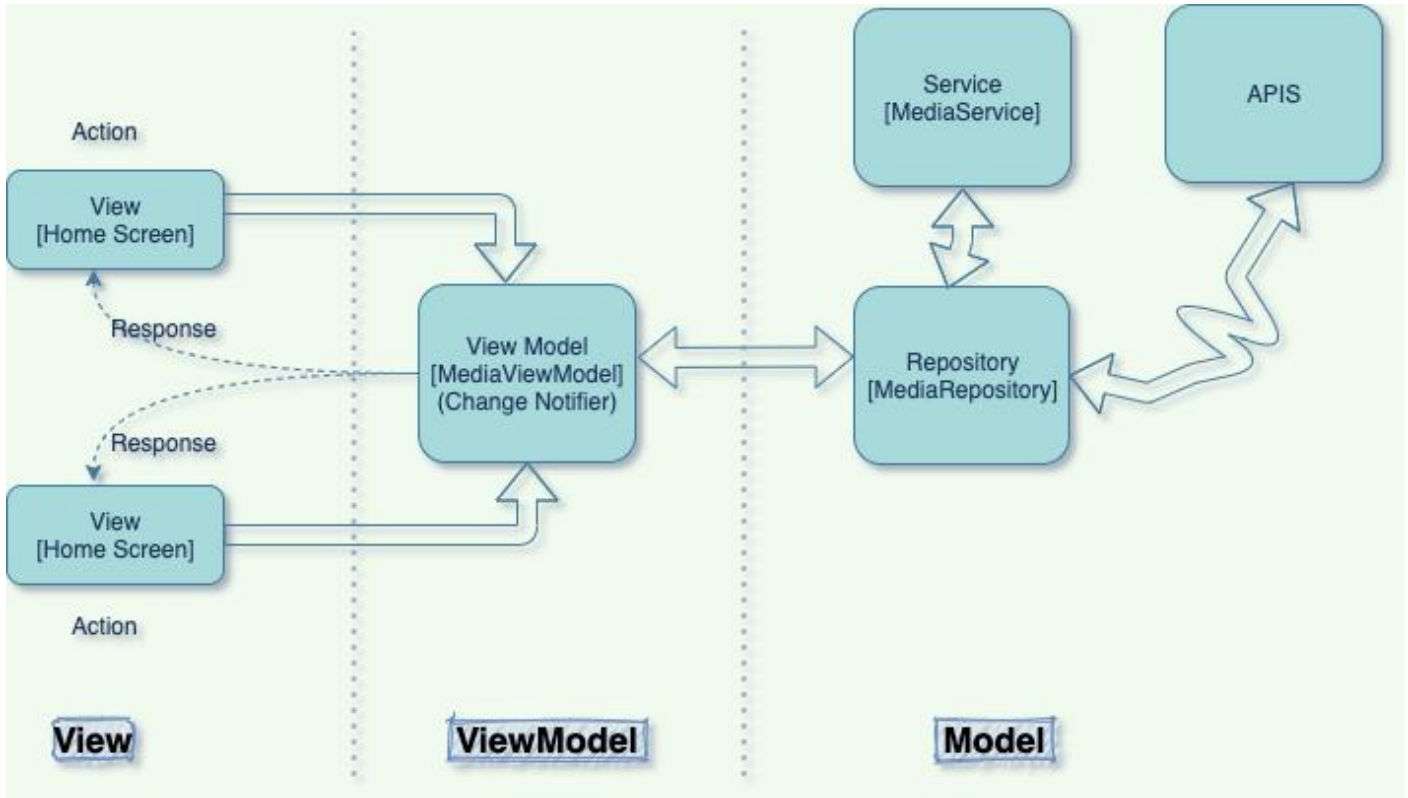
➤ **ViewModel**

ViewModel is the mediator between View and Model, which accept all the user events and request that to Model for data. Once the Model has data then it returns to ViewModel and then ViewModel notify that data to View.

ViewModel can be used by multiple views, which means a single ViewModel can provide data to more than one View.

➤ **View**

The view is where the user is interacting with Widgets that are shown on the screen. These user events request some actions which navigate to ViewModel, and the rest of ViewModel does the job. Once ViewModel has the required data then it updates View.



REQUIRED TOOLS

- **Firebase Database**

Firebase is a toolset to “build, improve, and grow your app”, and the tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don’t really want to build, because they’d rather be focusing on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and scale with little to no effort on the part of the developer.

- **Flutter –**

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android).

Flutter consists of two important parts:

- An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
- A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

To develop with Flutter, you will use a programming language called Dart. The language was created by Google in October 2011, but it has improved a lot over these past years.

Dart focuses on front-end development, and you can use it to create mobile and web applications.

If you know a bit of programming, Dart is a typed object programming language. You can compare Dart's syntax to JavaScript.

- **Used language- Dart**

Dart is an open-source, general-purpose, object-oriented programming language with C-style syntax developed by **Google in 2011**. The purpose of Dart programming is to create a frontend user interfaces for the web and mobile apps. It is under active development, compiled to native machine code for building mobile apps, inspired by other programming languages such as Java, JavaScript, C#, and is Strongly Typed. Since Dart is a compiled language so you cannot execute your code directly; instead, the compiler parses it and transfer it into machine code.

It supports most of the common concepts of programming languages like classes, interfaces, functions, unlike other programming languages. Dart language does not support arrays directly. It supports collection, which is used to replicate the data structure such as arrays, generics, and optional typing.

- **Visual studio Code**

A code editor for making the app.

- **Android Studio**

It is widely used because it has below features:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

- **Provider**

It can be used in a variety of ways in your Flutter app. It can be used in-lieu of more robust state-management libraries, or along side them. It can be used with `InheritedWidgets` and `StatefulWidget`s, or again, in place of them. It "does" two jobs:

- Separates your state from your UI
- Manages rebuilding UI based on state changes

Which makes loads of things simpler, from reasoning about state, to testing to refactoring. It makes your codebase *scalable*.

Consider this widget tree that uses no libraries at all:

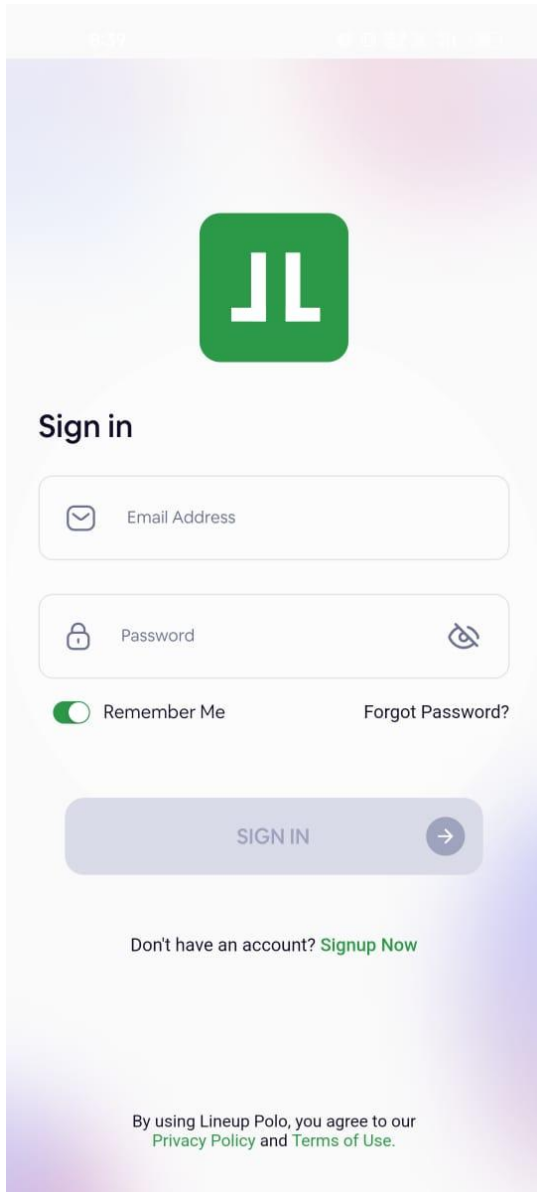
This widget tree is an extremely crude version of the Counter app we all know and love. It passes a callback (`onPressed`) to a `FloatingActionButton` and a counter property down a text widget. When `onPressed` is fired, it updates the counter.

Design of the app


1) Splash Screen




2) Signup page




8:37



Sign in



Remember Me [Forgot Password?](#)

SIGN IN 

Don't have an account? [Signup Now](#)

By using Lineup Polo, you agree to our [Privacy Policy](#) and [Terms of Use](#).

3) Matches

9:49 100% 11:25

< May 2022 >

Mon	Tue	Wed	Thu	Fri	Sat	Sun
9	10	11	12 ..	13	14	15

Today

🕒 02:00 Grand Champions Polo Club

📅 Thu 12 May **The Sun Cup**

+

🏆 Casablanca vs 🏆 Board Ape

🕒 06:50 United club of polo

📅 Thu 12 May **Bundesliga**

+

🏆 Adhish vs 🏆 Ritik

1.5 - 1



4) Tournament

USD 🏠 🔍 📅 📌 👤

Filters Calendar

Ongoing

United club of polo Entries closed

Premier league +

Monday, April 25 - Thursday, June 2

8-10 Goal

United club of polo Entries closed

La liga +

Saturday, April 30 - Monday, May 30

Open Handicap

United club of polo Entries closed

Bundesliga +

Sunday, May 1 - Tuesday, May 31

1-21 Goal

United club of polo Entries closed

Premier league new +

Saturday, May 7 - Tuesday, May 31

Open Handicap

United club of polo Entries closed

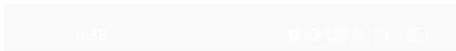
london polo club +

Saturday, May 7 - Tuesday, May 31

Open Handicap

Matches Tournaments Clubs Notifications Profile

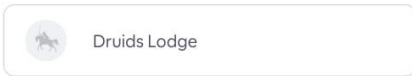
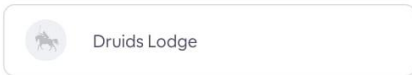
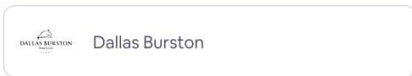
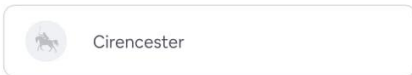
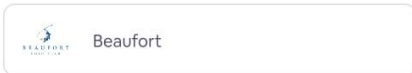
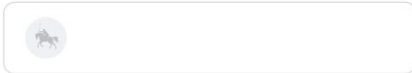
5) Clubs



 | Search...

ALL CLUBS

My Clubs



Matches

Tournaments

Clubs

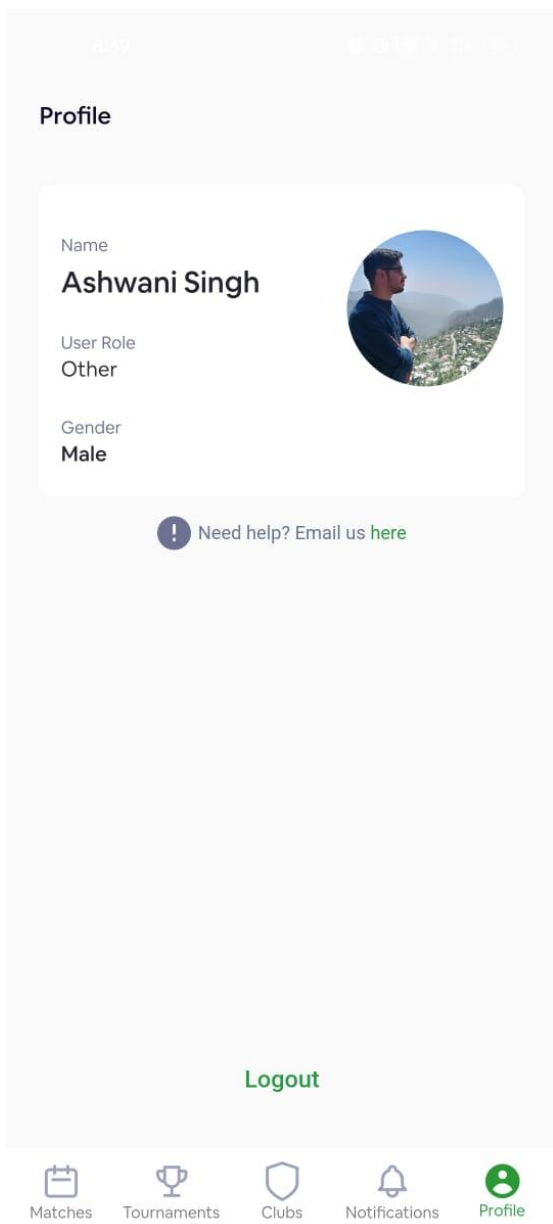
Notifications

Profile

6) Notification



7) Profile



Working of Project

1.) Spectatoters View.

Spectator signup using their mobile number.

Features for spectators.

- a) Watch the live match score.
- b) Watch the live tournament.
- c) See the details of their favourite clubs.
- d) Watch the schedule match in their club.
- e) Edit their profile.

2.) Manager View.

Managers can add and edit the tournaments.

Features for Manager .

- A) Watch the live match score.
- B) Watch the live tournament.
- C) See the details of their favourite clubs.
- D) Add the tournaments in calendars for the players.
- E) Manage the club.
- F) Watch the schedule match in their club.
- G) Edit their profile.

3.) Player View.

Player can see the upcoming matches and participate in the match.

Features for Player .

- A) Watch the live match score.
- B) Watch the live tournament.
- C) See the details of their favourite clubs.
- D) Watch the upcoming match and participate in them.

- E) Watch the schedule match in their club.
- F) Edit their profile.

Code

main.dart

```
Future<void> main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  setUpServiceLocator();  
  await Firebase.initializeApp().whenComplete() => runApp(const MyApp());  
}
```

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

```
  // This widget is the root of your application.
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return MaterialApp(  
      title: 'Fridayy',  
      theme: line_poloThemes.lightTheme,  
      darkTheme: line_poloThemes.lightTheme,  
      debugShowCheckedModeBanner: false,  
      navigatorKey: navigationService.navigatorKey,  
      onGenerateRoute: router.generateRoute,  
      initialRoute: '/',
```

```
    );
```

```
  }
```

```
}
```

```
class MyHomePage extends StatefulWidget {  
  const MyHomePage({Key? key, required this.title}) : super(key: key);  
  final String title;
```

```
  @override
```

```
  State<MyHomePage> createState() => _MyHomePageState();
```

```
}
```

```
class _MyHomePageState extends State<MyHomePage> {  
  final int _counter = 0;  
  FirebaseAuth auth = FirebaseAuth.instance;
```

```
  signUp() async {
```

```
    if (kIsWeb) {
```

```
      navigationService.showSnackBar('In Progress');
```

```
    } else {
```

```
      await auth.verifyPhoneNumber(  
        phoneNumber: '+91 7985434482',
```

```

timeout: const Duration(seconds: 60),
codeAutoRetrievalTimeout: (String verificationId) {
  navigationService.showSnackBar('Code Retrieved');
},
verificationCompleted: (PhoneAuthCredential phoneAuthCredential) {
  navigationService.showSnackBar('Verification Complete');
},
codeSent: (String verificationId, int? forceResendingToken) async {
  navigationService.showSnackBar('Code Sent');
  final PhoneAuthCredential credential = PhoneAuthProvider.credential(
    verificationId: verificationId,
    smsCode: "123456",
  );

  // Sign the user in (or link) with the credential
  final UserCredential userCredential =
    await auth.signInWithCredential(credential);
  print(userCredential.user?.uid.hashCode);
},
verificationFailed: (FirebaseAuthException error) {
  print(error.toString());
  navigationService.showSnackBar('Verification Failed $error');
},
);
}
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Text(
            'You have pushed the button this many times:',
          ),
          Text(
            '$_counter',
            style: Theme.of(context).textTheme.headline4,
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(

```



```
onPressed: () => signUp(),  
tooltip: 'Increment',  
child: const Icon(Icons.add),  
),  
);  
}  
}
```

- UI

- a. splash_screen_view.dart

```
class SplashScreenView extends StatelessWidget {  
  const SplashScreenView({Key? key}) : super(key: key);
```

```
  @override
```

```
  Widget build(BuildContext context) {  
    sizeConfig.init(context);  
    return BaseView<SplashScreenViewModel>(  
      onModelReady: (model) => model.init(),  
      builder: (context, model, child) {  
        return Scaffold(  
          backgroundColor: const Color(0xFF1A3764),  
          body: Center(  
            child: Image.asset(  
              'assets/images/splash_flutter.png',  
              fit: BoxFit.contain,  
              height: 175,  
              width: 175,  
            ),  
          ),  
        );  
      },  
    );  
  }  
}
```

- b. Onboarding Screen View

```
class OnboardingScreenView extends StatefulWidget {  
  const OnboardingScreenView({Key? key}) : super(key: key);
```

```
  @override
```

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

```
class _OnboardingScreenViewState extends State<OnboardingScreenView> {
```

```
  @override
```

```
  Widget build(BuildContext context) {  
    return BaseView<OnboardingScreenViewModel>(  
      onModelReady: (model) => model.init(),  
      builder: (context, model, child) {  
        return Scaffold(  
          body: Column(  
            children: [  
              
```

```

Expanded(
  flex: 518,
  child: PageView.builder(
    itemCount: 3,
    controller: model.controller,
    itemBuilder: (context, index) {
      return Image.asset(
        'assets/images/image-$index.png',
        fit: BoxFit.fitHeight,
      );
    },
  ),
),
Expanded(
  flex: 378,
  child: Container(
    alignment: Alignment.center,
    width: sizeConfig.getPropWidth(414),
    color: const Color(0xFFFF7F6F2),
    child: SizedBox(
      width: sizeConfig.getPropWidth(242),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const Spacer(),
          AnimatedOpacity(
            duration: const Duration(milliseconds: 250),
            opacity: model.opacity,
            child: Text(
              model.title[model.page],
              style: const TextStyle(
                color: Color(0xFF1A3764),
                fontWeight: FontWeight.bold,
                fontSize: 38,
              ),
            ),
          ),
          AnimatedOpacity(
            duration: const Duration(milliseconds: 250),
            opacity: model.opacity,
            child: Text(
              model.subTitle[model.page],
              style: const TextStyle(
                color: Color(0xFF94A2B8),
                fontWeight: FontWeight.w600,
                fontSize: 16,
              ),
            ),
          ),
        ],
      ),
    ),
  ),
),

```

```

    ),
  ),
  const Spacer(),
  Padding(
    padding: EdgeInsets.only(
      bottom: sizeConfig.getPropHeight(66)),
    child: CustomRoundRectButton(
      onTap: model.next,
      child: const Text(
        'Start',
        style: TextStyle(color: Colors.white),
      ),
    ),
  ),
),
),
],
),
),
),
),
],
),
);
},
);
}
}
}

```

c. Base View

```

class BaseView<T extends ChangeNotifier> extends StatefulWidget {
  const BaseView({
    required this.builder,
    this.onModelReady,
  });
  final Function(T)? onModelReady;
  final Widget Function(BuildContext, T, Widget?) builder;

  @override
  _BaseViewState<T> createState() => _BaseViewState<T>();
}

class _BaseViewState<T extends ChangeNotifier> extends State<BaseView<T>> {
  T model = serviceLocator<T>();

  @override
  void initState() {
    widget.onModelReady?.call(
      model,
    );
  }
}

```

```
    super.initState();
  }

  @override
  Widget build(
    BuildContext context,
  ) {
    return ChangeNotifierProvider<T>(
      create: (context) => model,
      child: Consumer<T>(
        builder: widget.builder,
      ),
    );
  }
}
```

- Service

- 1. Navigation

- (1) Navigation Service

```
abstract class NavigationService {
    Future<dynamic> pushScreen(String routeName, {dynamic arguments});
    Future<dynamic> popAndPushScreen(String routeName, {dynamic arguments});
    Future<dynamic> pushReplacementScreen(String routeName, {dynamic arguments});
    Future<dynamic> popAndPushReplacement(String routeName, {dynamic arguments});
    Future<dynamic> removeAllAndPush(
        String routeName,
        String tillRoute, {
        dynamic arguments,
    });
    void pushDialog(Widget dialog);
    void showSnackBar(
        String message, {
        Duration duration = const Duration(seconds: 2),
    });
    void pop();
}
```

- (2) Navigation Service Implementation

```
class NavigationServiceImpl extends NavigationService {
    GlobalKey<NavigatorState> navigatorKey = GlobalKey<NavigatorState>();

    @override
    Future<dynamic> pushScreen(String routeName, {dynamic arguments}) {
        return navigatorKey.currentState!
            .pushNamed(routeName, arguments: arguments);
    }

    @override
    Future<dynamic> popAndPushScreen(String routeName, {dynamic arguments}) {
        navigatorKey.currentState!.pop();
        return pushScreen(routeName, arguments: arguments);
    }

    @override
    Future<dynamic> pushReplacementScreen(String routeName, {dynamic arguments}) {
        return navigatorKey.currentState!
            .pushReplacementNamed(routeName, arguments: arguments);
    }

    @override
```

```
Future<dynamic> popAndPushReplacement(String routeName, {dynamic arguments}) {  
  navigatorKey.currentState!.pop();  
  return pushReplacementScreen(routeName, arguments: arguments);  
}
```

```
@override  
Future<dynamic> removeAllAndPush(  
  String routeName,  
  String tillRoute, {  
    dynamic arguments,  
  }) {  
  return navigatorKey.currentState!.pushNamedAndRemoveUntil(  
    routeName,  
    ModalRoute.withName(tillRoute),  
    arguments: arguments,  
  );  
}
```

```
@override  
void pushDialog(Widget dialog) {  
  showDialog(  
    context: navigatorKey.currentContext!,  
    barrierColor: Colors.transparent,  
    barrierDismissible: false,  
    builder: (BuildContext context) {  
      return dialog;  
    },  
  );  
}
```

```
@override  
void showSnackBar(  
  String message, {  
    Duration duration = const Duration(seconds: 2),  
  }) =>  
  ScaffoldMessenger.of(navigatorKey.currentContext!).showSnackBar(  
    SnackBar(  
      behavior: SnackBarBehavior.floating,  
      duration: duration,  
      content: Text(message),  
    ),  
  );
```

```
@override  
void pop() {  
  return navigatorKey.currentState!.pop();  
}  
}
```

2. Size Config

(a) Size config

```
abstract class SizeConfig {  
    void init(BuildContext context);  
}
```

(b) Size config implementation

```
class SizeConfigImpl implements SizeConfig {  
    static double screenWidth = 0.0;  
    static double screenHeight = 0.0;  
    double dp = 0.0;  
  
    @override  
    void init(BuildContext context) {  
        screenWidth = MediaQuery.of(context).size.width;  
        screenHeight = MediaQuery.of(context).size.height;  
        dp = screenWidth / 160;  
    }  
  
    double getPropHeight(double staticHeight) {  
        return screenHeight * (staticHeight / 896);  
    }  
  
    double getPropWidth(double staticWidth) {  
        return screenWidth * (staticWidth / 414);  
    }  
}
```

(c) Service Locator

```
GetIt serviceLocator = GetIt.instance;  
final navigationService = serviceLocator<NavigationServiceImpl>();  
final sizeConfig = serviceLocator<SizeConfigImpl>();  
  
void setupServiceLocator() {  
    serviceLocator.registerSingleton(SizeConfigImpl());  
    serviceLocator.registerSingleton(NavigationServiceImpl());  
  
    serviceLocator.registerFactory() => SplashScreenViewModel();  
    serviceLocator.registerFactory() => OnboardingScreenViewModel();  
}
```


- Bussiness_login

- I. Util

- ◆ enums.dart

```
enum ViewState {  
  idle,  
  busy,  
}
```

- ◆ routing_constants.dart

```
class Routes {  
  static const String splashScreen = "/";  
  static const String onboarding = "/onboarding";  
}
```

- ◆ theme.dart

```
class line_poloThemes {  
  static const Color _lightCursorColor = Color(0xff34AD64);  
  static const Color _lightAccentColor = Color(0xff34AD64);  
  static const Color _lightScaffoldColor = Colors.white;  
  static const Color _lightPrimaryColor = Colors.white;  
  static const Color _lightPrimaryVariantColor = Color(0xFFe5e5e5);  
  static const Color _lightIconColor = Color(0xff8C8E8D);  
  static const Color _lightColorSchemePrimary = Color(0xfffab57);  
  
  static const Color _darkCursorColor = Color(0xff34AD64);  
  static const Color _darkAccentColor = Color(0xff34AD64);  
  static const Color _darkScaffoldColor = Color(0xff18191A);  
  static const Color _darkPrimaryColor = Colors.black;  
  static const Color _darkPrimaryVariantColor = Colors.black;  
  static const Color _darkIconColor = Colors.white70;  
  static const Color _darkColorSchemePrimary = Color(0xfffab57);  
  
  static final lightTheme = ThemeData(  
    scaffoldBackgroundColor: _lightScaffoldColor,  
    textSelectionTheme: const TextSelectionThemeData(  
      cursorColor: _lightCursorColor,  
    ),  
    primaryColor: _lightPrimaryColor,  
    iconTheme: const IconThemeData(  

```

```

    color: _lightIconColor,
  ),
  fontFamily: 'product-sans',
  textTheme: _lightTextTheme,
  inputDecorationTheme: _lightInputDecor,
  colorScheme: const ColorScheme.light(
    primaryVariant: _lightPrimaryVariantColor,
    primary: _lightColorSchemePrimary,
    secondary: Color(0xffff5f5f5),
    secondaryVariant: _darkScaffoldColor,
  ).copyWith(secondary: _lightAccentColor),
);

static final darkTheme = ThemeData(
  textSelectionTheme: const TextSelectionThemeData(
    cursorColor: _darkCursorColor,
  ),
  scaffoldBackgroundColor: _darkScaffoldColor,
  primaryColor: _darkPrimaryColor,
  iconTheme: const IconThemeData(
    color: _darkIconColor,
  ),
  fontFamily: 'product-sans',
  textTheme: _darkTextTheme,
  inputDecorationTheme: _darkInputDecor,
  colorScheme: const ColorScheme.dark(
    primaryVariant: _darkPrimaryVariantColor,
    primary: _darkColorSchemePrimary,
    secondary: Colors.black,
    secondaryVariant: _lightScaffoldColor,
  ).copyWith(secondary: _darkAccentColor),
);

static const TextTheme _lightTextTheme = TextTheme(
  headline4: TextStyle(
    fontFamily: 'Nunito',
    fontWeight: FontWeight.w700,
    color: Colors.black,
    fontSize: 24.0,
  ),
  headline5: TextStyle(
    fontWeight: FontWeight.w600,
    fontFamily: 'Nunito',
    fontSize: 28,
  ),
  headline6: TextStyle(
    fontSize: 32,
    fontWeight: FontWeight.w700,

```

```

    fontFamily: 'Nunito',
    color: Color(0xFF2128BD),
  ),
  bodyText1: TextStyle(
    fontFamily: 'Nunito',
    fontWeight: FontWeight.w400,
    color: Color(0xFF2128BD),
    fontSize: 18.0,
  ),
  bodyText2: TextStyle(
    fontFamily: 'Nunito',
    fontWeight: FontWeight.w600,
    color: Color(0xFF2128BD),
    fontSize: 18.0,
  ),
  caption: TextStyle(
    fontFamily: 'Nunito',
    fontWeight: FontWeight.w400,
    color: Color(0xFF2128BD),
    fontSize: 18.0,
  ),
);

static const TextTheme _darkTextTheme = TextTheme(
  headline4: TextStyle(
    fontWeight: FontWeight.w600,
    fontSize: 18,
  ),
  headline5: TextStyle(
    fontWeight: FontWeight.w700,
    fontSize: 24,
  ),
  headline6: TextStyle(
    fontSize: 28,
    fontWeight: FontWeight.w600,
    color: Color(0xFF2128BD),
  ),
  bodyText1: TextStyle(
    fontSize: 14,
    color: Colors.white,
  ),
  bodyText2: TextStyle(
    fontSize: 14,
    color: Colors.white,
  ),
  caption: TextStyle(
    fontWeight: FontWeight.w400,
    color: Color(0xFF2128BD),
  ),
);

```

```
    fontSize: 18.0,  
  ),  
);
```

```
static const InputDecorationTheme _lightInputDecor = InputDecorationTheme(  
  border: InputBorder.none,  
  focusedBorder: UnderlineInputBorder(  
    borderSide: BorderSide(color: Color(0xFF008A37)),  
  ),  
  enabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(color: Colors.grey),  
  ),  
  errorBorder: InputBorder.none,  
  disabledBorder: InputBorder.none,  
  errorMaxLines: 3,  
);
```

```
static const InputDecorationTheme _darkInputDecor = InputDecorationTheme(  
  border: InputBorder.none,  
  focusedBorder: UnderlineInputBorder(  
    borderSide: BorderSide(color: Color(0xFF008A37)),  
  ),  
  enabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(color: Colors.grey),  
  ),  
  errorBorder: InputBorder.none,  
  disabledBorder: InputBorder.none,  
  errorMaxLines: 3,  
);
```

```
static final ThemeData lightTheme1 = ThemeData(  
  brightness: Brightness.light,  
  visualDensity: const VisualDensity(vertical: 0.5, horizontal: 0.5),  
  primaryColor: const Color(0xffE5D5B3),  
  primaryColorBrightness: Brightness.light,  
  primaryColorLight: const Color(0x1aF5E0C3),  
  primaryColorDark: const Color(0xff936F3E),  
  canvasColor: const Color(0xffE09E45),  
  scaffoldBackgroundColor: const Color(0xffB5BFD3),  
  bottomAppBarColor: const Color(0xff6D42CE),  
  cardColor: const Color(0xaaF5E0C3),  
  dividerColor: const Color(0x1f6D42CE),  
  focusColor: const Color(0x1aF5E0C3),  
  hoverColor: const Color(0xffDEC29B),  
  highlightColor: const Color(0xff936F3E),  
  splashColor: const Color(0xff457BE0),  
  // splashFactory: # override create method from InteractiveInkFeatureFactory  
  selectedRowColor: Colors.grey,  
  unselectedWidgetColor: Colors.grey.shade400,
```

```
disabledColor: Colors.grey.shade200,
buttonTheme: const ButtonThemeData(
  //button themes
),
toggleButtonsTheme: const ToggleButtonsThemeData(
  //toggle button theme
),
secondaryHeaderColor: Colors.grey,
backgroundColor: const Color(0xff457BE0),
dialogBackgroundColor: Colors.white,
indicatorColor: const Color(0xff457BE0),
hintColor: Colors.grey,
errorColor: Colors.red,
toggleableActiveColor: const Color(0xff6D42CE),
textTheme: const TextTheme(
  //text themes that contrast with card and canvas
),
primaryTextTheme: const TextTheme(
  //text theme that contrast with primary color
),
inputDecorationTheme: const InputDecorationTheme(
  // default values for InputDecorator, TextField, and TextFormField
),
iconTheme: const IconThemeData(
  //icon themes that contrast with card and canvas
),
primaryIconTheme: const IconThemeData(
  //icon themes that contrast primary color
),
sliderTheme: const SliderThemeData(
  // slider themes
),
tabBarTheme: const TabBarTheme(
  // tab bar theme
),
tooltipTheme: const TooltipThemeData(
  // tool tip theme
),
cardTheme: const CardTheme(
  // card theme
),
chipTheme: const ChipThemeData(
  backgroundColor: Color(0xff936F3E),
  disabledColor: Color(0xaaF5E0C3),
  shape: StadiumBorder(),
  brightness: Brightness.light,
  labelPadding: EdgeInsets.all(8),
  labelStyle: TextStyle(),
```

```
padding: EdgeInsets.all(8),
secondaryLabelStyle: TextStyle(),
secondarySelectedColor: Colors.white38,
selectedColor: Colors.white,
// chip theme
),
platform: TargetPlatform.android,
materialTapTargetSize: MaterialTapTargetSize.padded,
applyElevationOverlayColor: true,
pageTransitionsTheme: const PageTransitionsTheme(
  //page transition theme
),
appBarTheme: const AppBarTheme(
  //app bar theme
),
bottomAppBarTheme: const BottomAppBarTheme(
  // bottom app bar theme
),
snackBarTheme: const SnackBarThemeData(
  // snack bar theme
),
dialogTheme: const DialogTheme(
  // dialog theme
),
floatingActionButtonTheme: const FloatingActionButtonThemeData(
  // floating action button theme
),
colorScheme: const ColorScheme(
  primary: Color(0xffEDD5B3),
  primaryVariant: Color(0x1aF5E0C3),
  secondary: Color(0xffC9A87C),
  secondaryVariant: Color(0xaaC9A87C),
  brightness: Brightness.light,
  background: Color(0xffB5BFD3),
  error: Colors.red,
  onBackground: Color(0xffB5BFD3),
  onError: Colors.red,
  onPrimary: Color(0xffEDD5B3),
  onSecondary: Color(0xffC9A87C),
  onSurface: Color(0xff457BE0),
  surface: Color(0xff457BE0),
),
navigationRailTheme: const NavigationRailThemeData(
  // navigation rail theme
),
typography: Typography.material2018(),
cupertinoOverrideTheme: const CupertinoThemeData(
  //cupertino theme
```

```

    ),
    bottomSheetTheme: const BottomSheetThemeData(
      //bottom sheet theme
    ),
    popupMenuTheme: const PopupMenuThemeData(
      //pop menu theme
    ),
    bannerTheme: const MaterialBannerThemeData(
      // material banner theme
    ),
    dividerTheme: const DividerThemeData(
      //divider, vertical divider theme
    ),
    buttonBarTheme: const ButtonBarThemeData(
      // button bar theme
    ),
    fontFamily: 'ROBOTO',
    splashFactory: InkSplash.splashFactory,
    textSelectionTheme: const TextSelectionThemeData(
      cursorColor: Color(0xff936F3E),
      selectionColor: Color(0xffB5BFD3),
      selectionHandleColor: Color(0xff936F3E),
    ),
  );

```

```

static final ThemeData darkTheme1 = ThemeData(
  brightness: Brightness.dark,
  visualDensity: const VisualDensity(vertical: 0.5, horizontal: 0.5),
  primaryColor: const Color(0xff5D4524),
  primaryColorBrightness: Brightness.dark,
  primaryColorLight: const Color(0x1a311F06),
  primaryColorDark: const Color(0xff936F3E),
  canvasColor: const Color(0xffE09E45),
  scaffoldBackgroundColor: const Color(0xffB5BFD3),
  bottomAppBarColor: const Color(0xff6D42CE),
  cardColor: const Color(0xaa311F06),
  dividerColor: const Color(0x1f6D42CE),
  focusColor: const Color(0x1a311F06),
  hoverColor: const Color(0xa15D4524),
  highlightColor: const Color(0xaf2F1E06),
  splashColor: const Color(0xff457BE0),
  // splashFactory: # override create method from InteractiveInkFeatureFactory
  selectedRowColor: Colors.grey,
  unselectedWidgetColor: Colors.grey.shade400,
  disabledColor: Colors.grey.shade200,
  buttonTheme: const ButtonThemeData(
  //button themes
  ),

```

```

toggleButtonsTheme: const ToggleButtonsThemeData(
//toggle button theme
  ),
  secondaryHeaderColor: Colors.grey,
  backgroundColor: const Color(0xff457BE0),
  dialogBackgroundColor: Colors.white,
  indicatorColor: const Color(0xff457BE0),
  hintColor: Colors.grey,
  errorColor: Colors.red,
  toggleableActiveColor: const Color(0xff6D42CE),
  textTheme: const TextTheme(
//text themes that contrast with card and canvas
  ),
  primaryTextTheme: const TextTheme(
//text theme that contrast with primary color
  ),
  inputDecorationTheme: const InputDecorationTheme(
// default values for InputDecorator, TextField, and TextFormField
  ),
  iconTheme: const IconThemeData(
//icon themes that contrast with card and canvas
  ),
  primaryIconTheme: const IconThemeData(
//icon themes that contrast primary color
  ),
  sliderTheme: const SliderThemeData(
  // slider themes
  ),
  tabBarTheme: const TabBarTheme(
  // tab bat theme
  ),
  tooltipTheme: const TooltipThemeData(
  // tool tip theme
  ),
  cardTheme: const CardTheme(
  // card theme
  ),
  chipTheme: const ChipThemeData(
  backgroundColor: Color(0xff2F1E06),
  disabledColor: Color(0xa15D4524),
  shape: StadiumBorder(),
  brightness: Brightness.dark,
  labelPadding: EdgeInsets.all(8),
  labelStyle: TextStyle(),
  padding: EdgeInsets.all(8),
  secondaryLabelStyle: TextStyle(),
  secondarySelectedColor: Colors.white38,
  selectedColor: Colors.white

```



```
// chip theme
',
),
platform: TargetPlatform.android,
materialTapTargetSize: MaterialTapTargetSize.padded,
applyElevationOverlayColor: true,
pageTransitionsTheme: const PageTransitionsTheme(
  //page transition theme
),
appBarTheme: const AppBarTheme(
  //app bar theme
),
bottomAppBarTheme: const BottomAppBarTheme(
  // bottom app bar theme
),
snackBarTheme: const SnackBarThemeData(
  // snack bar theme
),
dialogTheme: const DialogTheme(
  // dialog theme
),
floatingActionButtonTheme: const FloatingActionButtonThemeData(
  // floating action button theme
),
colorScheme: const ColorScheme(
  primary: Color(0xff5D4524),
  primaryVariant: Color(0x1a311F06),
  secondary: Color(0xff457BE0),
  secondaryVariant: Color(0xaa457BE0),
  brightness: Brightness.dark,
  background: Color(0xffB5BFD3),
  error: Colors.red,
  onBackground: Color(0xffB5BFD3),
  onError: Colors.red,
  onPrimary: Color(0xff5D4524),
  onSecondary: Color(0xff457BE0),
  onSurface: Color(0xff457BE0),
  surface: Color(0xff457BE0),
),
navigationRailTheme: const NavigationRailThemeData(
  // navigation rail theme
),
typography: Typography.material2018(),
cupertinoOverrideTheme: const CupertinoThemeData(
  //cupertino theme
),
bottomSheetTheme: const BottomSheetThemeData(
  //bottom sheet theme
```

```

    ),
    popupMenuTheme: const PopupMenuThemeData(
      //pop menu theme
    ),
    bannerTheme: const MaterialBannerThemeData(
      // material banner theme
    ),
    dividerTheme: const DividerThemeData(
      //divider, vertical divider theme
    ),
    buttonBarTheme: const ButtonBarThemeData(
      // button bar theme
    ),
    fontFamily: 'ROBOTO',
    splashFactory: InkSplash.splashFactory,
    textSelectionTheme: const TextSelectionThemeData(
      cursorColor: Color(0xff483112),
      selectionColor: Color(0x1a483112),
      selectionHandleColor: Color(0xff483112),
    ),
  );
}

```

◆ validator

```

class Validate {
  static String? validatePhoneNumber(String? number) {
    return number?.isEmpty ?? true
      ? 'Phone number is required field'
      : number?.length != 10
        ? 'Phone number should have 10 digits'
        : null;
  }

  static String? validateName(String? number) {
    return number?.isEmpty ?? true ? 'Name cannot be empty' : null;
  }
}

```

II. View_model

(i) Splash_screen_view_model.dart

```
class BaseModel extends ChangeNotifier {
  ViewState _state = ViewState.idle;

  ViewState get state => _state;
  bool get isBusy => _state == ViewState.busy;

  void setState(ViewState viewState) {
    _state = viewState;
    notifyListeners();
  }
}
```

(ii) Onboarding_screen_view_model.dart

```
class OnboardingScreenViewModel extends BaseModel {
  PageController controller = PageController(
    initialPage: 0,
  );

  final List<String> title = [
    'Hey,\nWelcome',
    'Flawless\nservice',
    'User\nfriendly'
  ];
  final List<String> subTitle = [
    'Here in line_polo message privacy does matter most.',
    'We work hard to provide flawless service to our users possible',
    'Moreover, our main concern is to make the app user friendly comparing others on the market'
  ];

  double opacity = 1.0;
  int page = 0;

  void scrollListener() {
    opacity = controller.page! % 1 == 0
      ? 1.0
      : (controller.page! - controller.page!.floor()) / 10;
    page = controller.page!.floor();
    notifyListeners();
  }
}
```

```

}

void next() {
  if (page != 2) {
    controller.animateToPage(
      page + 1,
      duration: const Duration(milliseconds: 250),
      curve: Curves.easeIn,
    );
  }
}

init() {
  controller.addListener(scrollListener);
}

@override
void dispose() {
  controller.dispose();
  super.dispose();
}
}

```

(iii) Base_view_model.dart

```

class BaseModel extends ChangeNotifier {
  ViewState _state = ViewState.idle;

  ViewState get state => _state;
  bool get isBusy => _state == ViewState.busy;

  void setState(ViewState viewState) {
    _state = viewState;
    notifyListeners();
  }
}

```

Conclusion

UI and API in app is already added and app is in staging and soon will be live on Google Playstore and appstore.

CHAPTER-2 Introduction

References

- [1] Esports Research: A Literature Review [Prof. Jason G. Reitman, Maria J. Anderson-Coto] (April 15, 2019).
- [2] Esports: A Guide to Competitive Video Gaming [Prof. Josh Chapman] 01 May 2017.
- [3] The effect of games and simulations on higher education: a systematic literature review [Prof. Dimitrios Vlachopoulos] Article number 22 (2017) Published by 10 July 2017.
- [4]
- [5] The evolution of Esports: An analysis of its origin and a look at its prospective future growth as enhanced by information technology management tools [Prof. Anders H
- [5] D. P. Roel Hartman, Christian Rokitta, Oracle utility specific for cell net applications - Roel Hartman, Christian Rokitta, David Peake - Google Books. 2013.
- [6] Viber Encryption assessment.” [Online]. to be had: <https://www.viber.com/safety-evaluate/>.
- [7] WhatsApp inc, “WhatsApp safety whitepaper,” p. 10, 2017.
- [8] “Telegram F.A.Q.” [Online]. available: <https://telegram.org/faq>.
- [9] T. Susanka, “protection evaluation of the Telegram IM,” p. 70, 2016.
- [10] B. O. B. Kamwendo, “Vulnerabilities of signaling gadget wide variety 7 (ss7) to cyber attacks and a way to mitigate towards these vulnerabilities. bob kamwendo,” vol. 7, no. 7, 2015.
- [11] John Leyden, “SS7 spookery on the reasonably-priced lets in hackers to impersonate cell chat subscribers • The check in,” 2016. [Online]. to be had: https://www.theregister.co.uk/2016/05/10/ss7_mobile_chat_hack/.
- [12] “energetic sessions and two-Step Verification.” [Online]. available: <https://telegram.org/weblog/sessions-and-2-step-verification>

[13] T. Whitepaper, “Messenger Secret Conversations,” 2016.

[14] “Android Keystore System | Android Developers.” [Online]. Available: <https://developer.android.com/training/articles/keystore.html>.

[15] D. J. Bernstein, “Extending the Salsa20 nonce,” no. Mc 152, pp. 1–14, 2011. [

16] M. B. Jones, “The Emerging JSON-Based Identity Protocol Suite,” 2011.

[17] “Firebase Cloud Messaging | Firebase.” [Online]. Available: <https://firebase.google.com/docs/cloud-messaging/>.

[18] D. J. Bernstein, “Curve25519: new Diffie-Hellman speed records,” vol. 25519, 2006.

Publication/ Screen Shots

