

# **FUT-TRACKER**

**A Project Report Submitted  
In Partial Fulfillment of the Requirements for the Degree of**

## **MASTER OF COMPUTER APPLICATIONS**

**by**

**SAURABH KUMAR  
(1900290149087)**

**Under the Supervision of  
Prof. Ankit Verma  
(Assistant Professor)  
KIET Group of Institutions, Ghaziabad**



**to the**

**FACULTY OF MCA**

**DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY  
(Formerly Uttar Pradesh Technical University) LUCKNOW**

**July, 2021**

## **DECLARATION**

I hereby declare that the work presented in this report entitled "FUT-TRACKER", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

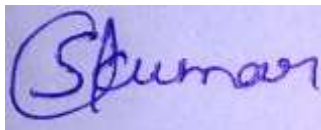
I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name : Saurabh Kumar

Roll. No. : 1900290149087

Branch : Master of Computer Application



**Candidate Signature**

# TRAINING CERTIFICATE

## ANNEXURE 1 Training Details



Name : Mr. Saurabh Kumar  
Trainee Title : Trainee  
Department : Developer Tools  
Training Start Date : 15 January 2021 (unless a revised date is intimated by the Company)  
Training Period : 6 months  
Location : Noida  
Stipend : Rs. 15,000 per month  
Leaves : Leave entitlement during training period shall be one day per month. These leaves are not encashable, and will lapse if not utilized prior to the end of the training period.  
Notice Period : The training may be terminated on one week's written notice from either side.


- w.e.f. start of training / prorated for partial periods
- Compensation will be subject to deductions for taxes and other withholdings as required by law.
- Eligible for Employee State Insurance (ESI) however individual contribution will be deducted as required by law.

On behalf of GrapeCity India Pvt. Ltd.

  
Nandee Aoki  
Vice President, HR & Finance

14 January 2021

Accepted by:

  
Saurabh Kumar

15-01-2021  
Date

*Note: Compensation details of each trainee and employee are confidential information of the Company. Violation of this confidentiality obligation may result in disciplinary action, including possible termination of your training.*

Saurabh Kumar  
Trainee Letter  
14 January 2021

A Cluster of Minds  www.grapecity.com  
Regd Office: E-326, Naraina Vihar, Delhi - 110002, India  
Email ID : OC.india.corporate@gmail.com, CIN : U72200DL1996PTC085493

Page 2

## **CERTIFICATE**

Certified that **Saurabh Kumar (1900290149087)** has carried out the project work presented in this report entitled “**Fut-Tracker**” for the award of **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

**Dr. Ajay Kumar Shrivastava**  
**Examiner**

**External**

Professor & Head

Dept. of Computer Applications

KIET Group of Institutions, Ghaziabad

Date:

## **ABSTRACT**

Flutter is Google's portable user interface (UI) framework for building modern, native, and reactive applications for iOS and Android. This chapter helps developers to learn how the Flutter framework works behind the scenes. Flutter uses widgets to create the UI, and Dart is the language used to develop the applications. Flutter uses its own rendering engine to draw widgets. Elements have a reference to the widget and are responsible for comparing the widget differences. In programming, developers have different lifecycle events that usually happen in a linear mode, one after another as each stage is completed.

Flutter is now emerging as a very good option for cross-platform mobile app development and it helps to reduce the codebase. Mobile devices to some extent are much powerful than desktops and they are highly personal, always on, always with users, usually connected and directly addressable. Furthermore they are crawling with powerful sensors with various functions that detect location, orientation, movement, proximity that opens the capabilities of the mobile apps to perform many complex tasks seamlessly.

Fut-tracker is an application that is designed to track the football matches. When user enters information about the football matches then statistics are created of that user and that contains the player statistics, leaderboard, and lineup. Android Studio Integrated Development Environment, which is also known as the IDE. Basically, the IDE is the interface between developers and Android Studio. The Android Monitor automatically displays while debugging an application. It contains a very useful tool called logcat. Android Studio Code Completion should display a list of values that may use to try to complete the code statement. After building an application, it is easy to debug it and is easy to see what is going on inside the code. Breakpoints are

## ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my Project supervisor, **Prof. Ankit Verma, Assistant Professor of Department of Computer Applications** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Saurabh Kumar**  
**1900290149087**

# TABLE OF CONTENTS

	Page No.
Declaration	ii
Training Certificate	iii
Certificate	iv
Abstract	v
Acknowledgement	vi
List of Figures	ix
<b>CHAPTER 1: INTRODUCTION</b>	<b>1-4</b>
1.1 Project Description	1-2
1.2 Flutter and Its Characteristics	2-4
<b>CHAPTER 2 : LITERATURE REVIEW</b>	<b>5-10</b>
2.1 Flutter	5
2.2 Flutter Architecture	6
2.3 Widgets	7
2.4 Stateless Widget	7
2.5 Stateful Widget	7
2.6 State Management	8-9
2.7 App Development	9-10
<b>CHAPTER 3 : REQUIREMENT GATHERING AND ANALYSIS</b>	<b>11-12</b>
3.1 Hardware Requirements	11
3.2 Software Requirements	11-12
3.3 Data Requirements	12
<b>CHAPTER 4 : DESIGN</b>	<b>13-17</b>
4.1 MVC Model	13-15
4.2 GUI	15-17
<b>CHAPTER 5 : TESTING</b>	<b>18-20</b>

5.1	Black Box Testing	18
5.2	White Box Testing	19
5.3	Grey Box Testing	19
5.4	Unit Testing	20
5.5	Integration Testing	20
5.6	System Testing	20
5.7	Acceptance Testing	20
<b>CHAPTER 6 : PROJECT WORKFLOW</b>		<b>21-37</b>
6.1	Installation and Setup Flutter	21-26
6.2	Installation of Android Studio (IDE)	26-27
6.3	Project Screens	27-37
<b>CHAPTER 7 : CONCLUSION AND FUTURE SCOPE</b>		<b>38</b>
<b>CHAPTER 8 : REFERENCES</b>		<b>39</b>



## LIST OF FIGURES

1.1	MOBILE DEVICE PLATFORMS	2
1.2	HISTORY OF FLUTTER	3
2.1	STATE MANAGEMENT	8
4.1.3	MVC MODEL	15
5	TESTING	19
6.1.3	SEARCH ENVIRONMENT VARIABLES	22
6.1.3	OPEN ENVIRONMENT VARIABLES	23
6.1.3	OPEN PATH VARIABLES	24
6.1.3	EDIT PATH VARIABLES	24
6.1.4	FLUTTER VERSION	25
6.1.5	CHECK FLUTTER DEPENDENCIES	26
6.2	DOWNLOAD ANDROID STUDIO	26
6.3.1	HOMEPAGE SCREEN	28
6.3.2	LEADERBOARD SCREEN	29
6.3.3	TRACK NEW MATCH SCREEN	30
6.3.4	FILTER SCREEN	31
6.3.5	MY MATCHES SCREEN	32
6.3.6	CHOOSE FORMATION 1	33
6.3.6	CHOOSE FORMATION 2	33
6.3.6	FORMATION SCREEN	34
6.3.7	PLAYER STATS SCREEN	35
6.3.8	MY CLUB SCREEN	36
6.3.9	WEEKEND LEAGUE RECAP SCREEN	37

# CHAPTER 1

## INTRODUCTION

### 1.1 PROJECT DESCRIPTION

**Flutter** was introduced by Google as an open-source technology for coding and creating native apps for Android and iOS. Flutter is relatively new as it was officially presented in December 2018 as the first stable version 1.0 at the Flutter Live event.

**Flutter** combines ease of development with performance similar to native performance while maintaining visual consistency between platforms. Flutter's programming language, Dart, was initially intended as a replacement for JavaScript. Most importantly, Flutter is open-source and completely free. At the moment, Flutter has equal popularity with React Native on both GitHub and Stack Overflow. Speaking of which, we contrasted Flutter's and React Native's pros and cons using nine criteria in a recent article.

Google uses Flutter for various Google Assistant modules and the Google Home Hub user interface. Moreover, there are already 50,000 Flutter apps available in the Google Play Store, and this number is increasing at a high rate. Alibaba Group, eBay, Groupon, and other popular e-commerce providers use Flutter as well to give their web and mobile applications uniform looks.

How Flutter Works: Widget Strategy and Dart Programming Language:-

The framework of Flutter, written in the Dart programming language, has the Flutter engine, Foundation library, and widgets. The approach to development in Flutter differs from others by its declarative UI writing. Here, there is a need to start from the end, meaning before starting the development of some element, the user needs to have in mind a complete picture of what kind of UI it will be. Many developers distinguish this UI writing as a clearer one, but it also causes certain difficulties for developers at first.

The main idea of Flutter is that developers can build the entire user interface by simply combining different widgets. The application interface consists of various nested widgets, which can be any object. This applies to anything from buttons to padding, and by combining widgets, the developer can customize the application radically. Widgets can influence each other and use built-in functions to respond to external changes in the

Toolbars to replace the app bar within the layouts, and to take advantage of the functionality it supports, including specialized scrolling techniques. When the app grows beyond a single screen, one needs to incorporate a navigation pattern to facilitate user interaction. Android offers a number of mechanisms that can be used to inform—and even interrupt—the user while the app is in the foreground, including Dialogs, Toast messages, and Snackbars.

According to data available through Wikipedia (see Figures 1.2 and 1.3), the Android platform runs on 64% of smartphones and on about 23.5% of all phones ([http://en.wikipedia.org/wiki/Mobile\\_operating\\_system](http://en.wikipedia.org/wiki/Mobile_operating_system)). Approximately 37% of all phones today are smartphones, leaving a whopping 60%+ of phones open to future adoption.

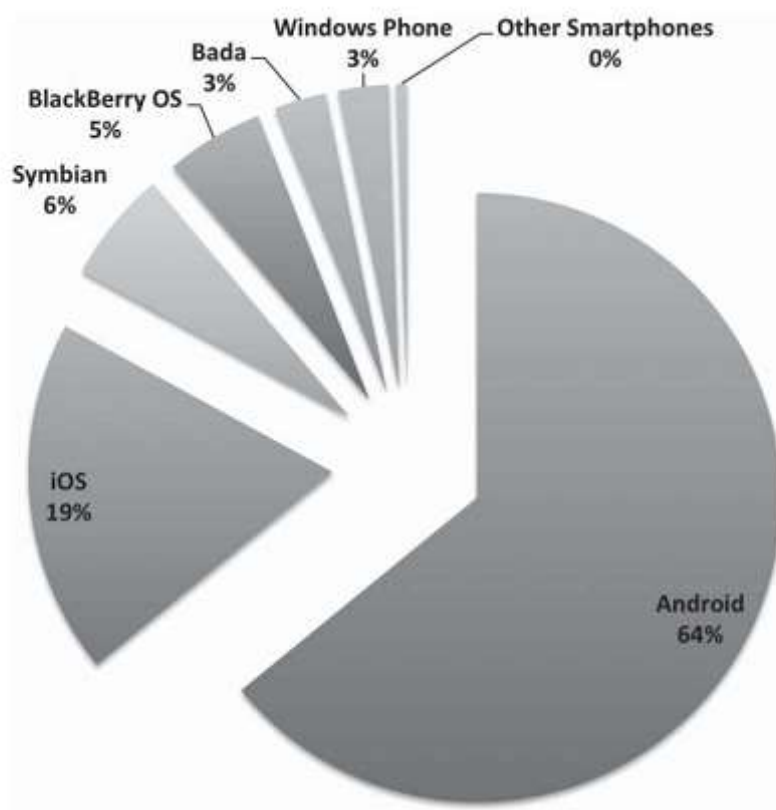


Fig. 1.1 Mobile device platforms

## 1.2 FLUTTER AND ITS CHARACTERISTICS

Flutter was a startup initiated by YC in 2013, after which Google acquired it. It is an open-source framework that can develop cross-platform applications using one codebase. Hence Flutter Android development is very same as its IOS development. This makes the development of apps smooth, easy, and quick. The language used to

code in flutter is Dart, developed by Google, and unique to Flutter.

Developers can also use platform-specific widgets to make their app have a native-developed feel to it. This will keep the end-user satisfied, and developers can save a lot of time and effort while developing. Besides all this, Flutter has a charming and flexible UI which contributes a lot to a user-friendly UX. All of these features make flutter a must use in app development.

The central idea behind Flutter is the use of widgets. It's by combining different widgets that developers can build the entire UI. Each of these widgets defines a structural element (like a button or menu), a stylistic element (a font or color scheme), a layout aspect (like padding), and many others.

## A brief history of Flutter

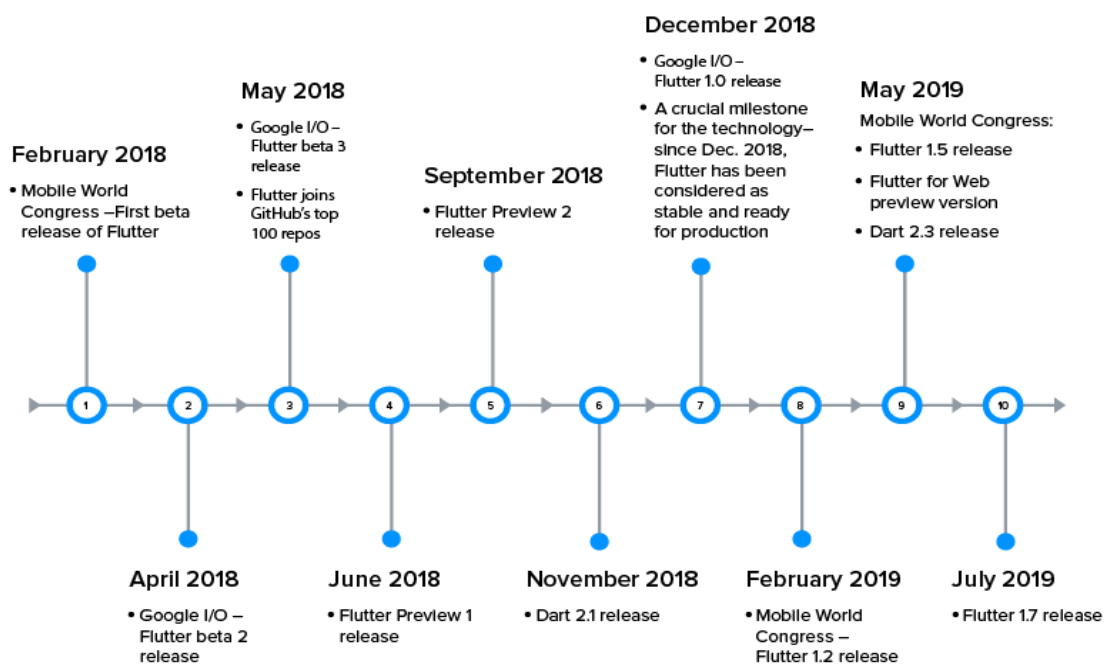


Fig. 1.2 History of flutter

### 1.2.1 CHARACTERSTICS

- **Free and open source:** It's an open-source project which makes it available for use and study by start-ups for any given purpose. It's more or less like an open-collaboration, hence making it easily accessible and user friendly. It not only gives you a diverse scope for design but also no other company provides you with as many options.
- **Amazing Widget catalog:** The best part about flutter is that you have a wide range of widgets beautifully cataloged for you. Not only does it make it hassle-free but also helps you make a functionally amazing app. You use Dart to write your Flutter app which is compiled to native code. The IntelliJ plugin makes for good integration.

- **It's Inherent Graphic Library:** Flutter uses the Skia — built-in library for rendering. This makes it more platform-independent. The SDK provides a rich set of widgets, in particular, the Material and Cupertino collections for rendering native-like widgets for Android and iOS. By combining various widgets, you get the opportunity to create a complex UI. Thanks to its own library, Flutter applications look the same on different versions of operating systems. Thus it is good to solve one of the main problems of mobile development today — the great variability of mobile devices.
- **Hot Reload:** One of the drawbacks of compiled languages before scripting languages is the loss of time for building a project. With frequent edits, it can take up a substantial part of the working time. The Flutter Hot Reload feature allows you to display the effect of your edits in the code immediately.
- **Compatibility:** Since widgets are part of the app and not the platform, you'll likely experience less or no compatibility issues on different OS versions. This in turn means less time spent on testing.
- **Great performance:** Dart compiles into native code and there is no need to access OEM widgets as Flutter has its own. This means less mediated communication between the app and the platform. As puts it: "Flutter is the only mobile SDK that provides reactive views without requiring a JavaScript bridge." All of this contributes to fast app startup times and less performance issues.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 FLUTTER**

Flutter has been around already since 2015 when Google first introduced it, but the buzz around it has grown stronger only lately. It's a cross-platform tool intended for creating Android and iOS apps from a single code base by using a modern, reactive framework. Flutter apps are built using Dart, a simple object-oriented programming language. The central idea of Flutter revolves around widgets. The entire UI is made of combining different widgets, each of which defines a structural element (like a button or menu), a stylistic element (like a font or color scheme), an aspect of layout (like padding), and so on. Flutter does not use OEM widgets, but provides its own ready-made widgets which look native either to Android (Material Design) or iOS apps (Cupertino). It's also possible to create custom widgets. Look for a technical overview. In terms of popularity, Flutter is making good progress. While Flutter had made it to the by the time release preview 1 was announced in June 2018, it has risen in the ranks and is now . Two years later Flutter SDK ranks as top , following giants like Linux, vue and vscode. This, without a doubt, is a promising trend. Thousands of Flutter apps have made its way to app stores, among these the Alibaba app with 50 million users.[1]

Flutter is now emerging as an very good option for cross-platform mobile app development and it helps to reduce the codebase. Mobile devices to some extent are much powerful than desktops and they are highly personal, always on, always with users, usually connected and directly addressable. Furthermore they are crawling with powerful sensors with various functions that detects location, orientation, movement, proximity that opens the capabilities of the mobile apps to perform many complex tasks seamlessly.[2]

## **2.2 FLUTTER ARCHITECTURE**

In view of architecture, the engine's C or C++ code involves compilation with Android's NDK and LLVM for iOS respectively, and during the compilation process, the Dart code is compiled into native code. Hot reload feature in Flutter is called as Stateful hot reload and it is a major factor for boosting the development cycle. Flutter supports it during development. Stateful hot reload is implemented by sending the updated source code into the running Dart Virtual Machine (Dart VM) without changing the inner structure of the application, therefore the transitions and actions of the application will be well-preserved after hot reloading.[3]

### **2.2.1 DART**

In Flutter, every application is written with the help of Dart. Google has developed and maintained a programming language called Dart. It is extensively used inside Google and it has been verified to have the proficiency to develop enormous web applications, such as AdWords. Flutter application renews the view tree on every new frame even when few other systems use reactive views. This behavior leads to a drawback that many objects, which might survive for a singular frame, will be created. As Dart is a modern programming language, it is optimized to handle this scenario in memory level with the help of "Generational Garbage Collection".[4]

### **2.2.2 FLUTTER UI/DART MANAGEMENT**

Flutter uses widgets as the main concept within in the code. Widget is the nickname for every component part that is built in Flutter. This could mean a box or a text that is referred to as a widget. A noticeable part of the widgets is that they are created by the Flutter developers to look native and developers are able to fully customize these to their liking. [16]

### **2.2.3 FLUTTER COMPILING**

The way Flutter works when running on Android is by compiling the developer's written Dart code to native with the help of AOT compiling. These together with x86 libraries that were created when compiling the dart code, are put into a runner and built as an APK. Flutter runs similar on iOS system, but instead the Dart code compiles into ARM library and is later put as a runner.[5]

## 2.3 WIDGETS

Widgets are just pieces of your user interface. If you are familiar with Android or iOS development, then you will make the immediate connection to views (for Android) and UIViews (for iOS). This is a good comparison to make and you will do fine to start your journey with this mindset. A more accurate way to think, though, is that a widget is a blueprint. Flutter uses these blueprints to build the view elements under the hood and render them to the screen. The different types of Flutter widgets are immutable. That is, they cannot be changed. Any properties that they contain are final and can only be set when the widget is initialized. This keeps them lightweight so that it's inexpensive to recreate them when the widget tree changes. There are two types of widgets: stateless and stateful.[6]

## 2.4 STATELESS WIDGETS

Stateless widgets are widgets that don't store any state. That is, they don't store values that might change. For example, an Icon is stateless; you set the icon image when you create it, and then it doesn't change any more. A Text widget is also stateless. If you want to change the text value, you just create a whole new widget with the new text. The Text Flutter widgets don't store a text property that can be changed.[7]

## 2.5 STATEFUL WIDGETS

The second type of widget is called a stateful widget. That means it can keep track of changes and update the UI based on those changes. The stateful widget itself is immutable, but it creates a State object that keeps track of the changes. When the values in the State object change, it creates a whole new widget with the updated values. So the lightweight widget gets recreated but the state persists across changes.[8]

## 2.6 STATE MANAGEMENT

The second type of widget is called a stateful widget. That means it can keep track of changes and update the UI based on those changes. The stateful widget itself is immutable, but it creates a State object that keeps track of the changes. When the values in the State object change, it creates a whole new widget with the updated values. So the lightweight widget gets recreated but the state persists across changes.

A state is information that can be read when the widget is built and might change or be modified over a lifetime of the app. If you want to change your widget, you need to update the state object, which can be done by using the `setState()` function



available for Stateful widgets. The `setState()` function allows us to set the properties of the state object that triggers a redraw of the UI. The state management is one of the most popular and necessary processes in the lifecycle of an application. According to official documentation, Flutter is declarative. It means Flutter builds its UI by reflecting the current state of your app. The following figure explains it more clearly where you can build a UI from the application state.

In Flutter, the state management categorizes into two conceptual types, which are given below:

- **Ephemeral State:** This state is also known as UI State or local state. It is a type of state which is related to the specific widget, or you can say that it is a state that contains in a single widget. In this kind of state, you do not need to use state management techniques. The common example of this state is Text Field.
- **App State:** It is different from the ephemeral state. It is a type of state that we want to share across various parts of our app and want to keep between user sessions. Thus, this type of state can be used globally. Sometimes it is also known as application state or shared state. Some of the examples of this state are User preferences, Login info, notifications in a social networking app, the shopping cart in an e-commerce app, read/unread state of articles in a news app, etc.

The simplest example of app state management can be learned by using the provider package. The state management with the provider is easy to understand and requires less coding. A provider is a third-party library. Here, we need to understand three main concepts to use this library. `ChangeNotifier` `ChangeNotifierProvider` `Consumer`. [9]

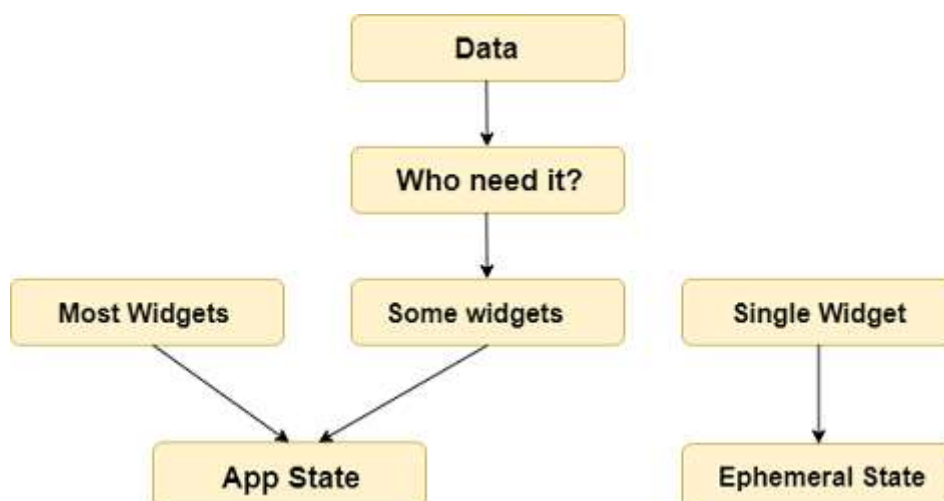


Fig. 2.6 State Management

## 2.7 APP DEVELOPMENT

Cross-platform mobile development today is full of compromise. Developers are forced to choose between either building the same app multiple times for multiple operating systems, or to accept a lowest common denominator solution that trades native speed and accuracy for portability. Flutter is an open source SDK for creating high-performance, high-fidelity mobile apps for iOS and Android. Few important features of flutter are - Just-in-time compilation is a way of executing computer code that involves compilation during execution of a program – at run time – rather than prior to execution. Most often, this consists of source code or more commonly byte code translation to machine code, which is then executed directly.[10] Ahead-of-time compilation (AOT compilation) is the act of compiling a higher-level programming language such as C or C++, or an intermediate representation such as Java byte code or .NET Framework Common Intermediate Language (CIL) code, into a native (system-dependent) machine code so that the resulting binary file can execute natively. Flutter's hot reload feature helps you quickly and easily experiment, build UIs, add features, and fix bugs. Hot reload works by injecting updated source code files into the running Dart Virtual Machine (VM). After the VM updates classes with the new versions of fields and functions, the Flutter framework automatically rebuilds the widget tree, allowing you to quickly view the effects of your changes. With Flutter, we believe we have a solution that gives you the best of both worlds: hardware-accelerated graphics and UI, powered by native ARM code, targeting both popular mobile operating systems.[11]

Versatile application advancement as of late is developing exponentially. Today every single individual in this world has an advanced mobile phone in his pocket. Cell phones consolidate a scope of capacities, for example, media players, camera and GPS with cutting edge processing capacities and contact screens are getting a charge out of consistently expanding prevalence Cell phone's assistance us to accomplish a scope of undertakings through something known as applications or Apps to short. As indicated by Gartner , Google's Android, Apple's iOS and RIM's Blackberry all have something like a 10 percent piece of the pie. For finishing this survey paper and learn about this subject an aggregate of four research papers were utilized which comprehended reasonable and ebb and flow situation of Cross-stage versatile application advancement.[12] MCIDER is fundamentally a working framework similarity engineering that can run applications worked for various versatile biological systems ideally iOS and Android together on the equivalent Smartphone or tablet. Fundamentally in less difficult terms CIDER had the capacity to run unmodified iOS parallels on the Android subsystem with no kind of alteration. Juice accomplishes the assignment of expanding the limit of home Android portion by at the same time utilizing the home part and the slave piece which is the application double interface for our situation. Client space of the slave part gets in contact with the Cider empowered bit in the very same ways as the slave bit.[13] That is, the iOS applications get in to Linux based bit approach as though they are dealing with a home part of iOS subsystem which

is running on a run of the mill iOS based gadget. Example of a remote part, and reuse and run unmodified outside client space library code. Presently going to the design of these two working frameworks. IOS keeps running on ARM CPUs like Android, yet has an altogether different programming biological system. IOS is based on the XNU A cross breed mix of a solid BSD part and a Mach microkernel running in a solitary bit address space. When we talk about Android, Each Android application is ordered into Dalvik byte code (dex) arrangement, and keeps running in a different Dalvik VM example.[14]

There are basically two ways to persist the data: Shared Preferences and Local Storage. Shared preferences make better choices when there's a tiny amount of data that needs to be stored. Local database is a better choice for a huge dataset. The Shared preference approach uses key/value pairs to store information on the device using Shared preference plugin. The Local database implementation uses moor library that is based on SQLite database. Persisting theme selection is demonstrated on all four platforms: Android, iOS, web, and desktop macOS apps.[15]

## **CHAPTER 3**

### **REQUIREMENT GATHERING AND ANALYSIS**

#### **3.1 Hardware Requirements**

- RAM 4GB – 8GB
- Windows or MAC
- 2 GHz or Higher Processor
- 10 GB Hard Disk

#### **3.2 Software Requirements**

- Android Studio

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools. To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules. Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction, and analysis.

Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store. The software was first announced at Google I/O in May 2013, and the first stable build was released in December 2014. Android Studio is available for Mac, Windows, and Linux desktop platforms. It replaced Eclipse Android Development Tools (ADT) as the primary IDE for Android application development. Android Studio and the Software.

- Flutter

Flutter is an open source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile application in Google's own language, Dart. This tutorial walks through the basics of Flutter framework, installation of Flutter SDK, setting up Android Studio to develop Flutter based application, architecture of Flutter framework and developing all type of mobile applications using Flutter framework.

### **3.3 Data Requirements**

The data for the application has been fetched through the futbin API's which is internally gets data from the EA sports servers.

## CHAPTER 4

### Design

#### 4.1 MVC Model

The Android environment adds yet another Graphical User Interface (GUI) toolkit to the Java ecosphere, joining AWT, Swing, SWT, and J2ME (leaving aside the web UI toolkits). If you've worked with any of these, the Android framework will look familiar. Like them, it is single-threaded, event-driven, and built on a library of nestable components. The Android UI framework is, like the other UI frameworks, organized around the common Model-View-Controller pattern illustrated in Figure. It provides structure and tools for building a Controller that handles user input (like key presses and screen taps) and a View that renders graphical information to the screen.

##### 4.1.1 THE MODEL

The Model is the guts of your application: what it actually does. It might be, for instance, the database of tunes on your device and the code for playing them. It might be your list of contacts and the code that places phone calls or sends IMs to them. While a particular application's View and Controller will necessarily reflect the Model they manipulate, a single Model might be used by several different applications. Think, for instance, of an MP3 player and an application that converts MP3 files into WAV files. For both applications, the Model includes the MP3 file format and codecs for it. The former application, however, has the familiar Stop, Start, and Pause controls, and plays the track. The latter may not produce any sound at all; instead, it will have controls for setting bitrate, etc. The Model is all about the data.

##### 4.1.2 THE VIEW

The View is the application's feedback to the user. It is the portion of the application responsible for rendering the display, sending audio to speakers, generating

tactile feedback, and so on. The graphical portion of the Android UI framework's View, is implemented as a tree of subclasses of the View class. Graphically, each of these objects represents a rectangular area on the screen that is completely within the rectangular area represented by its parent in the tree. The root of this tree is the application window.

### **4.1.3 THE CONTROLLER**

The Controller is the portion of an application that responds to external actions: a keystroke, a screen tap, an incoming call, etc. It is implemented as an event queue. Each external action is represented as a unique event in the queue. The framework removes each event from the queue in order and dispatches it.[7] Mobile application developers are often unaware of suitable architectural designs for their types of mobile application and of how transactions and data can be managed across changing connectivity states. However, users expect their mobile applications to run reliably even when the connection is limited or interrupted intermittently. Here, we present a design method and a tool environment that support mobile application developers in creating online- and offline-capable mobile applications on a high level of abstraction. An application model (app model) that captures the underlying data structure, behavior, and graphical user interface (GUI) of the mobile application is used to create online- and offline-capable mobile applications.[8]

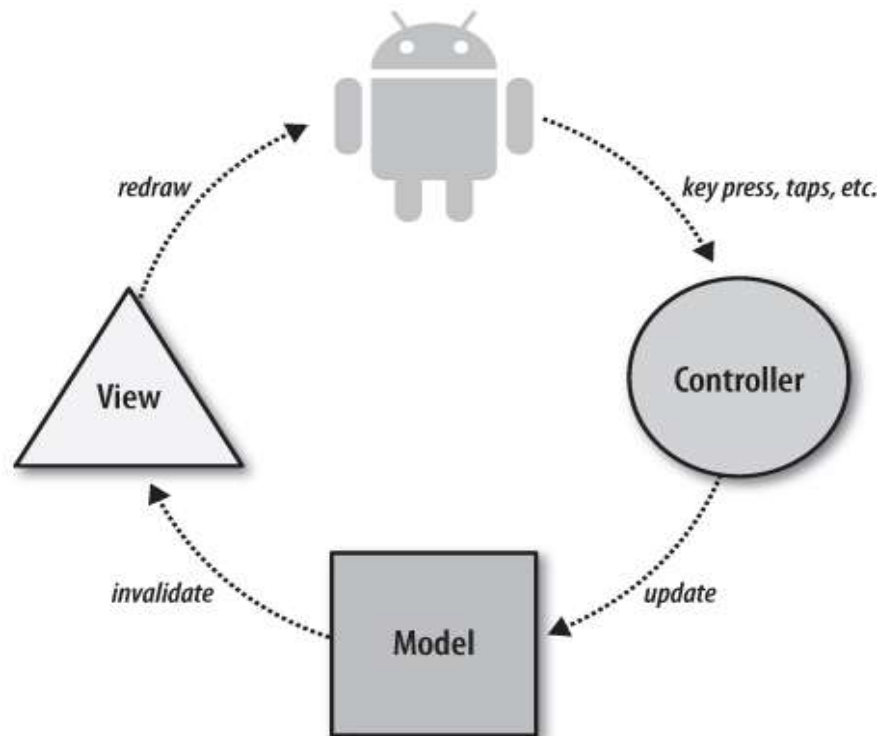


Fig. 4.1.3 MVC Model

## 4.2 GUI

To create the GUI in flutter you have to use Flutter Widgets and there are more than 1000 of them are available on official flutter documentation page. You will also know about the asynchronous widgets used to build interfaces dynamically using stream and future builders. The FutureBuilder widgets are used to build interfaces when data is not available all at once. The other widgets that you will be introduced to include widgets like TextField, ToggleButtons and AlertDialog. The TextField widget is used for enabling users to input text data in the forms. The ToggleButtons widget helps to build interfaces where toggling functionality is needed. Finally, a quick introduction to AlertDialog widget is provided for material and Cupertino design styles.

To build the responsive layout there are plenty of widgets available that shrink or wrap according to the screen sizes and provide the best responsive experience. The layout widgets are helpful in laying out the widgets on the screen. They are like a collection of multiple widgets. A layout widget has multiple children widgets. There are



two types of layout widgets: Single-child and Multi-child layout widgets. A Single-child layout type can have only one child widget assigned to it. A few examples of Single-child widgets are Container and Padding widgets. A Multi-child layout can have more than one child assigned to it. Few examples of Multi-child layouts are ListView, Row, Column, GridView, and Stack. The common layout widgets useful for creating the most common types of layouts are Container, ListView, Row, Column, Stack, and SizedBox. This chapter also discusses more specific widgets used for laying out screens like ConstrainedBox, SizedBox, IntrinsicHeight, IntrinsicWidth, and IndexedStack. The responsive layouts can adapt themselves depending upon the space available for rendering their widgets. This chapter will cover FittedBox, Expanded, Flexible, FractionallySizedBox, LayoutBuilder, and Wrap widgets. Some of the widgets like FittedBox position and scales their child while Expanded and Flexible widgets let their child expand to fill the available space. The FractionallySizedBox widget adjusts its size to a fraction of the total available space. The LayoutBuilder widget builds a widget tree based on its parent's widget size. The Wrap widget gives the flexibility to render its children in vertical or horizontal runs. When direction of laying out the widgets is horizontal or vertical, then it's recommended to use Row or Column widgets respectively. If main axis for laying out widgets is not available, it makes sense to use the Flex widget. When there laying out single child in then using Center or Align to position the child is a better choice. Lastly, you'll learn to use the ListView layout widget to render multiple Card widgets to display book information per Card widget. You can create application for all the four platforms: Android, iOS, web, and desktop using flutter. Flutter supports global and local themes. Global themes are responsible for styling colors, fonts, etc. at the application level. Local themes help define app styling at a screen or component level.

Flutter allows you to create apps that self-adapt to the device's screen size and orientation. There are two basic approaches to creating Flutter apps with responsive design:

Use the LayoutBuilder class From its builder property, you get a BoxConstraints object. Examine the constraint's properties to decide what to display. For example, if your maxWidth is greater than your width breakpoint, return a Scaffold object with a row that has a list on the left. If it's narrower, return a Scaffold object with a drawer containing that list. You can also adjust your display based on the device's height, the aspect ratio, or some other property. When the constraints change (for example, the user rotates the phone, or puts your app into a tile UI in Nougat), the build function runs.

Use the MediaQuery.of() method in your build functions This gives you the size,

orientation, etc, of your current app. This is more useful if you want to make decisions based on the complete context rather than on just the size of your particular widget. Again, if you use this, then your build function automatically runs if the user somehow changes the app's size.

Other useful widgets and classes for creating a responsive UI:

- `AspectRatio`
- `CustomSingleChildLayout`
- `CustomMultiChildLayout`
- `FittedBox`
- `FractionallySizedBox`
- `LayoutBuilder`
- `MediaQuery`
- `MediaQueryData`
- `OrientationBuilder`

## CHAPTER 5

### TESTING

The more features your app has, the harder it is to test manually. Automated tests help ensure that your app performs correctly before you publish it, while retaining your feature and bug fix velocity.

Automated testing falls into a few categories:

- A unit test tests a single function, method, or class. The goal of a unit test is to verify the correctness of a unit of logic under a variety of conditions. External dependencies of the unit under test are generally mocked out. Unit tests generally don't read from or write to disk, render to screen, or receive user actions from outside the process running the test. For more information regarding unit tests, you can view the following recipes or run `flutter test --help` in your terminal.
- A widget test (in other UI frameworks referred to as component test) tests a single widget. The goal of a widget test is to verify that the widget's UI looks and interacts as expected. Testing a widget involves multiple classes and requires a test environment that provides the appropriate widget lifecycle context. For example, the Widget being tested should be able to receive and respond to user actions and events, perform layout, and instantiate child widgets. A widget test is therefore more comprehensive than a unit test. However, like a unit test, a widget test's environment is replaced with an implementation much simpler than a full-blown UI system.
- An integration test tests a complete app or a large part of an app. The goal of an integration test is to verify that all the widgets and services being tested work together as expected. Furthermore, you can use integration tests to verify your app's performance. Generally, an integration test runs on a real device or an OS emulator, such as iOS Simulator or Android Emulator. The app under test is typically isolated from the test driver code to avoid skewing the results.

Generally speaking, a well-tested app has many unit and widget tests, tracked by code coverage, plus enough integration tests to cover all the important use cases. This advice is based on the fact that there are trade-offs between different kinds of testing, seen below.

	Unit	Widget	Integration
Confidence	Low	Higher	Highest
Maintenance cost	Low	Higher	Highest
Dependencies	Few	More	Most
Execution speed	Quick	Quick	Slow

Fig. 5 Testing

## 5.1 BLACK BOX TESTING

The technique of testing without having any knowledge of the interior workings of the application is called black box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

## 5.2 WHITE BOX TESTING

White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, a tester needs to know the internal workings of the code.

## 5.3 GREY BOX TESTING

Grey box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

## **5.4 UNIT TESTING**

Unit Testing contains the testing of each unit of Recruitment Application. We have tested each interface by input values and check whether it is working properly working or not we also tested database connectivity. We have entered value in interface and check that the values are properly goes to corresponding tuples or not.

## **5.5 INTEGRATION TESTING**

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom up integration testing and Top down integration testing.

## **5.6 SYSTEM TESTING**

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.

## **5.7 ACCEPTANCE TESTING**

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement.

## CHAPTER 6

### Project Workflow

#### 6.1 INSTALLATION AND SETUP FLUTTER

Flutter is an open source project so it can be easily available to download. To download flutter you can check the official flutter website.

##### 6.1.1 SYSTEM REQUIREMENT

To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems:** Windows 7 SP1 or later (64-bit), x86-64 based.
- **Disk Space:** 1.64 GB (does not include disk space for IDE/tools).
- **Tools:** Flutter depends on these tools being available in your environment.
  - Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
  - Git for Windows 2.x, with the Use Git from the Windows Command Prompt option.

##### 6.1.2 GET THE FLUTTER SDK

1. Download the latest stable release of the Flutter SDK from the official flutter website.
2. Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK (for example, C:\Users\<your-user-name>\Documents).

### 6.1.3 UPDATE YOUR PATH

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the PATH environment variable:

- From the Start search bar, enter 'env' and select **Edit environment variables for your account**.

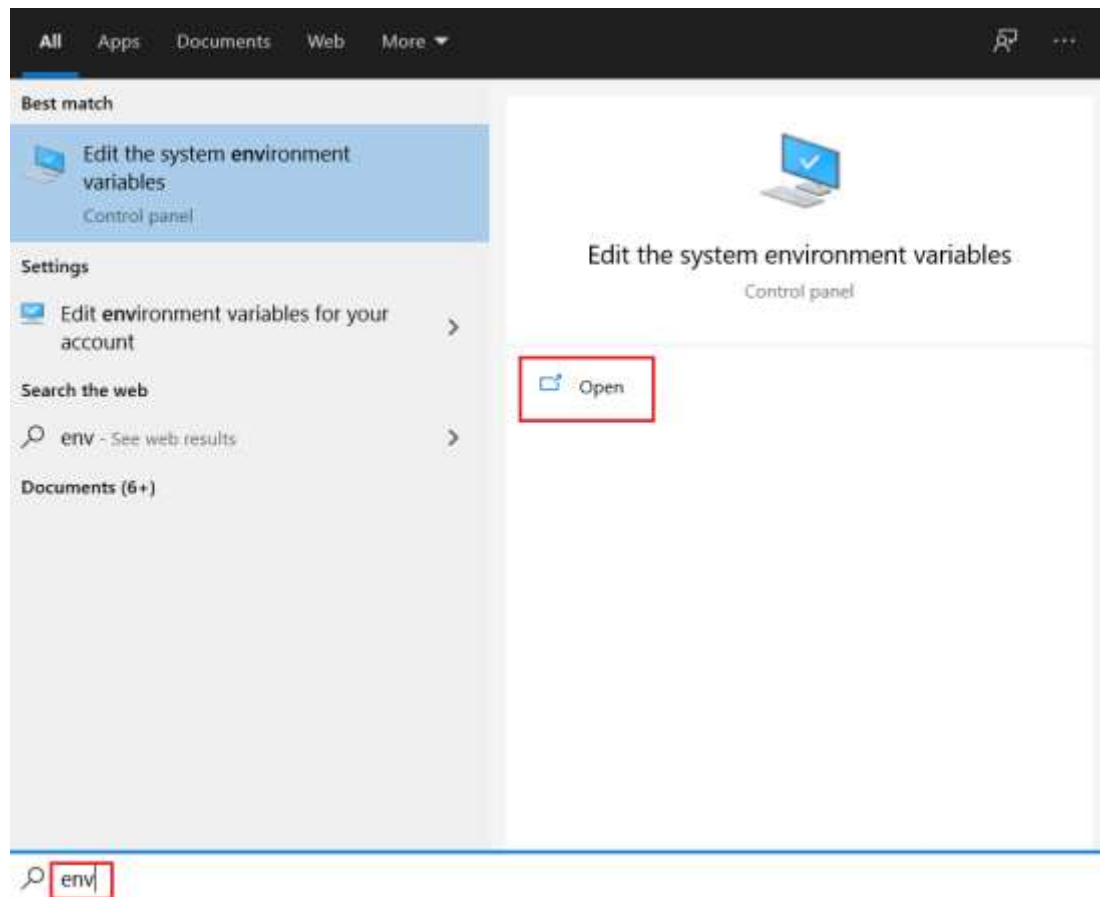


Fig. 6.1.3 Search Environment Variables

- Click on Environment Variables.

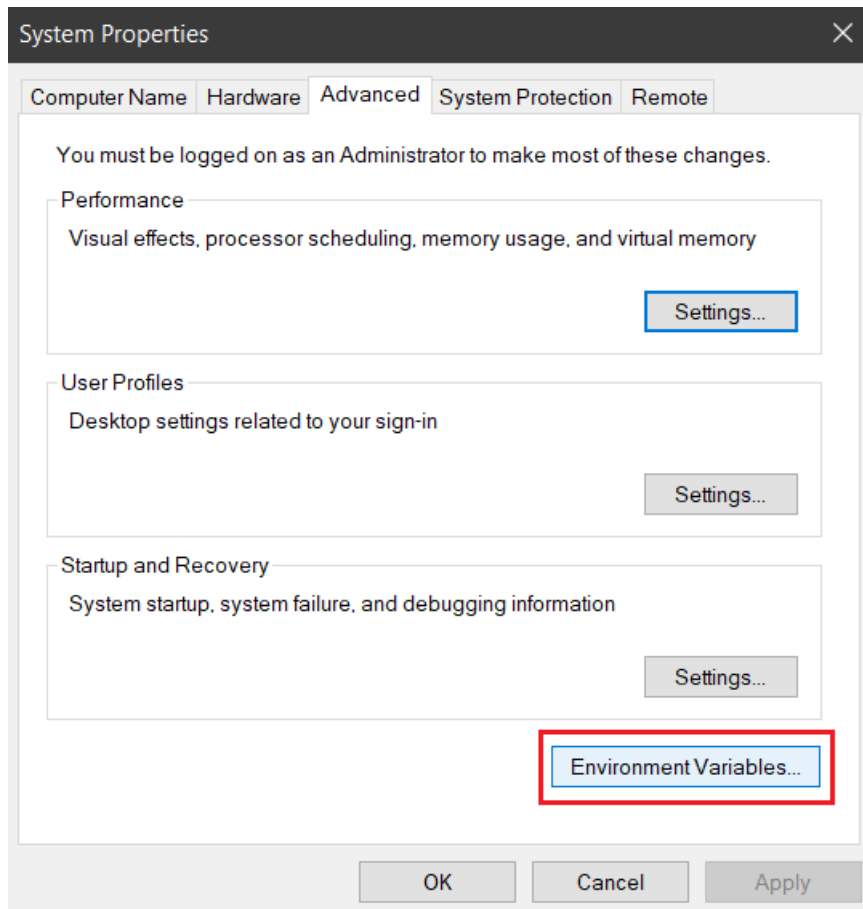


Fig. 6.1.3 Open Environment Variables

- Under **User variables** check if there is an entry called Path:
  - If the entry exists, append the full path to flutter\bin using ; as a separator from existing values.
  - If the entry doesn't exist, create a new user variable named Path with the full path to flutter\bin as its value. You have to close and reopen any existing console windows for these changes to take effect.



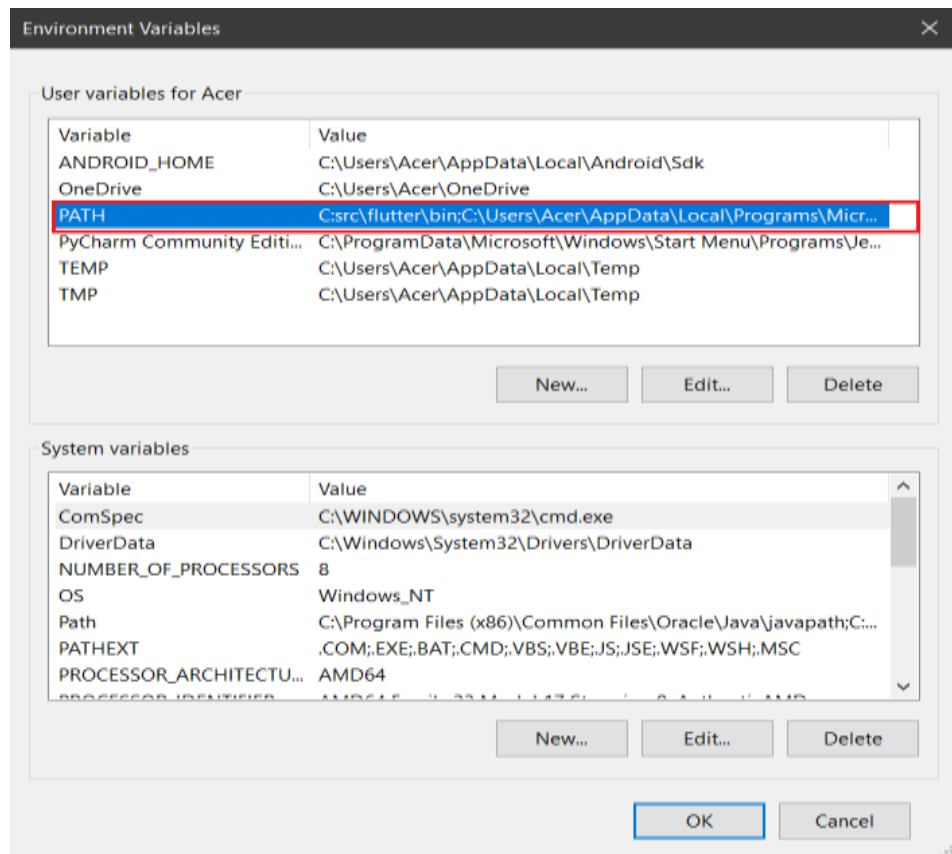
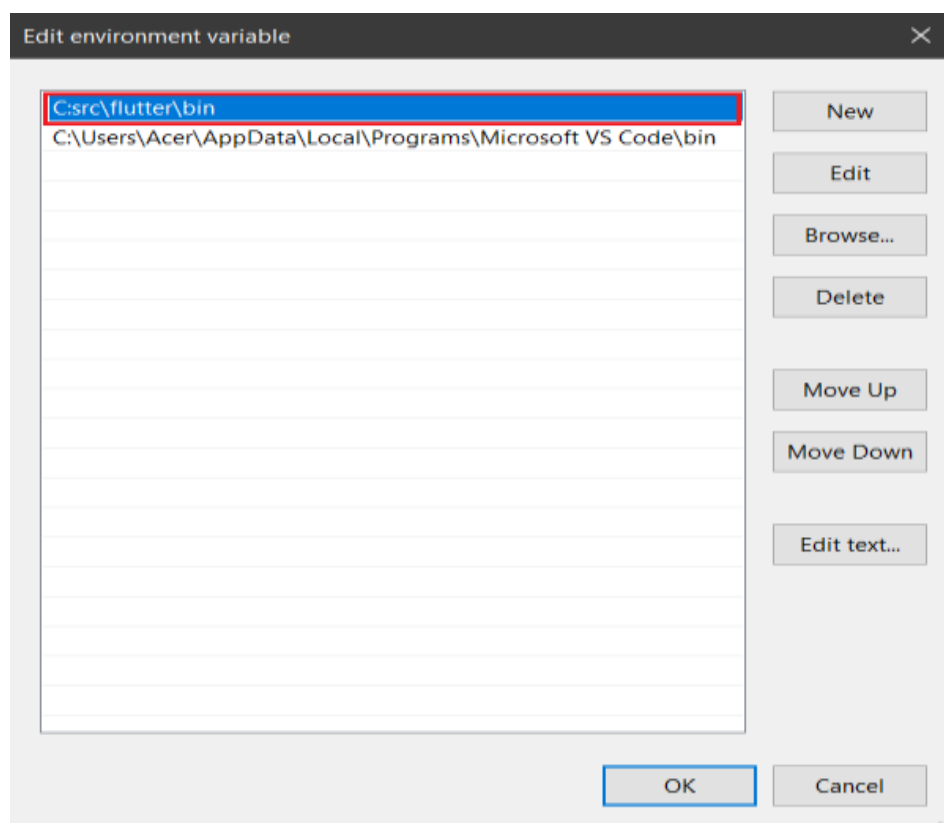


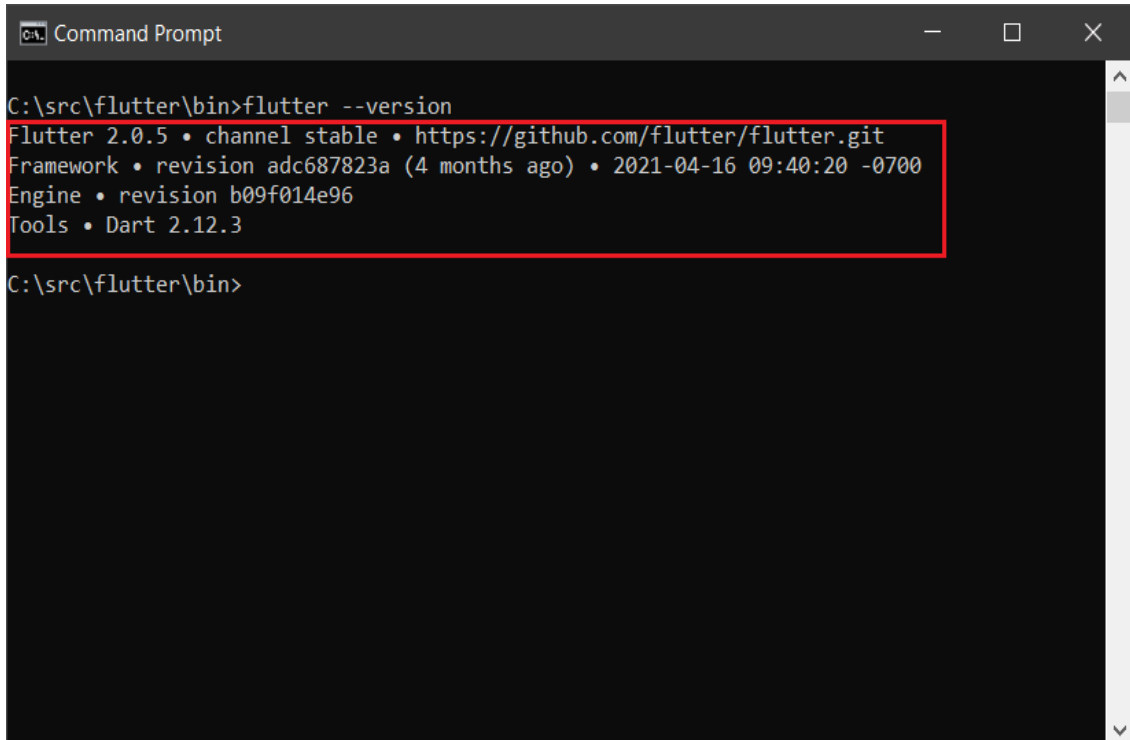
Fig. 6.1.3 Open Path Variable



### Fig. 6.1.3 Edit Path Variable

## 6.1.4 CHECK FLUTTER VERSION

To check the flutter configured properly open command prompt and type **flutter --version**.

A screenshot of a Windows Command Prompt window. The title bar says "Command Prompt". The command prompt shows the directory "C:\src\flutter\bin" and the command "flutter --version". The output is displayed in four lines: "Flutter 2.0.5 • channel stable • https://github.com/flutter/flutter.git", "Framework • revision adc687823a (4 months ago) • 2021-04-16 09:40:20 -0700", "Engine • revision b09f014e96", and "Tools • Dart 2.12.3". The output text is highlighted with a red rectangular box. The prompt "C:\src\flutter\bin>" is visible at the bottom.

```
C:\src\flutter\bin>flutter --version
Flutter 2.0.5 • channel stable • https://github.com/flutter/flutter.git
Framework • revision adc687823a (4 months ago) • 2021-04-16 09:40:20 -0700
Engine • revision b09f014e96
Tools • Dart 2.12.3

C:\src\flutter\bin>
```

Fig. 6.1.4 Flutter Version

## 6.1.5 CHECK FOR DEPENDENCIES

From a console window that has the Flutter directory in the path (see above), run the following command to see if there are any platform dependencies you need to complete the setup:

**C:\src\flutter>flutter doctor**

This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

If it shows no issues found then you can proceed further but if there is any error that you have to solve those and they proceed further.

```
Command Prompt

C:\src\flutter\bin>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.0.5, on Microsoft Windows [Version 10.0.19042.1110],
    locale en-IN)
[✓] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
[✓] Chrome - develop for the web
[✓] Android Studio (version 4.1.0)
[✓] VS Code (version 1.59.0)
[✓] Connected device (2 available)

• No issues found!

C:\src\flutter\bin>
```

Fig. 6.1.5 Check Flutter Dependencies

## 6.2 INSTALLATION OF ANDROID STUDIO (IDE)

1. Download Android Studio from <https://developer.android.com/studio> and then install normally on the directory which you want.

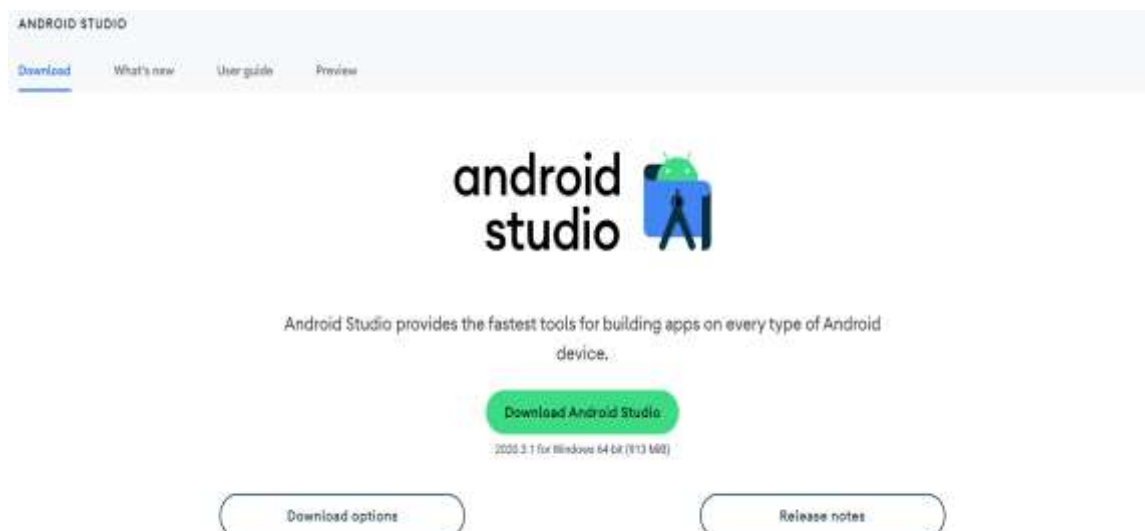


Fig. 6.2 Download Android Studio

2. Start Android Studio, and go through the ‘Android Studio Setup Wizard’. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.
3. Run flutter doctor to confirm that Flutter has located your installation of Android Studio. If Flutter cannot locate it, run flutter config --android-studio-dir <directory> to set the directory that Android Studio is installed to.

### 6.2.1 SETUP ANDROID DEVICE

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) or higher.

1. Enable Developer options and USB debugging on your device.
2. Windows-only: Install the Google USB Driver. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.
3. In the terminal, run the flutter devices command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your adb tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the ANDROID\_SDK\_ROOT environment variable to that installation directory.

### 6.3.2 SETUP ANDROID EMULATOR

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

1. Enable VM acceleration on your machine.
2. Launch Android Studio, click the AVD Manager icon, and select Create Virtual Device...
  - In older versions of Android Studio, you should instead launch Android Studio > Tools > Android > AVD Manager and select Create Virtual Device.... (The Android submenu is only present when inside an Android project.)
  - If you do not have a project open, you can choose Configure > AVD Manager and select Create Virtual Device...
3. Choose a device definition and select Next.
4. Select one or more system images for the Android versions you want to emulate, and select Next. An x86 or x86\_64 image is recommended.
5. Under Emulated Performance, select Hardware - GLES 2.0 to enable hardware acceleration.
6. Verify the AVD configuration is correct, and select Finish.

7. For details on the above steps, see Managing AVDs. In Android Virtual Device Manager, click Run in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

## 6.3 PROJECT SCREENS

This project has various screens those having specific purposes which are listed below:

### 6.3.1 HOMEPAGE SCREEN

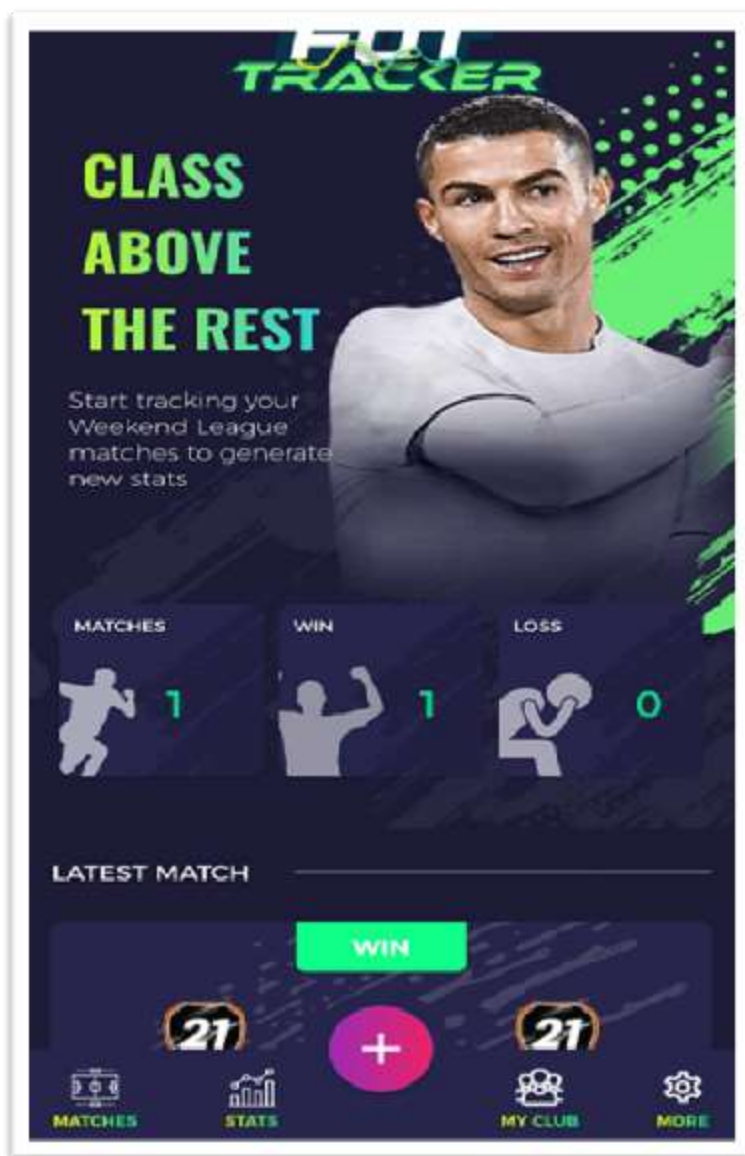


Fig. 6.3.1 Homepage Screen

This is the first screen of the application and this contains the overall information about

the statistics of the user like total win, loss and information about the latest match and also contains the leader board where user can click and then it opens leader board detail page.

### 6.3.2 LEADERBOARD SCREEN



Members	W	L	PTS
1  OLFA3	27	15	99
2  xChxster	15	5	71
3  vouzax	17	13	67
4  saurabh970	18	5	54
5  rapha13goat	13	17	47
6  m7mds	11	5	39
7  simoncarter1976	9	6	35
8  arjit	7	2	21
9  alvnnf	5	0	19
10  jsbrito	3	3	11
11  ELESHAHED	3	0	9
12  Sarsy	2	0	8
13  xshideex	2	1	8

Fig. 6.3.2 Leaderboard Screen

Leaderboard screen contains the ranking of the users according to the total match's player, win matches and loss matches. This screen also contains the users name, profile

image and total points. User can also filter the leaderboard according to the duration like weekly, monthly and all time.

### 6.3.3 TRACK NEW MATCH SCREEN

Fig. 6.3.3 Track New Match Screen

Track new match screen helps the user to create the new match and enter the match information like your goals, opponent goals, opponent's name, stadium(home or away), your formation, opponent formation, match information, rage quit, gameplay experience and individual players information like goals, assist, receive red card or not, Player of the match, Super sub, player rating.

After filling all the required information a new match created and user can also edit the

match in future in case there is some wrong information is filled.

### 6.3.4 FILTER SCREEN

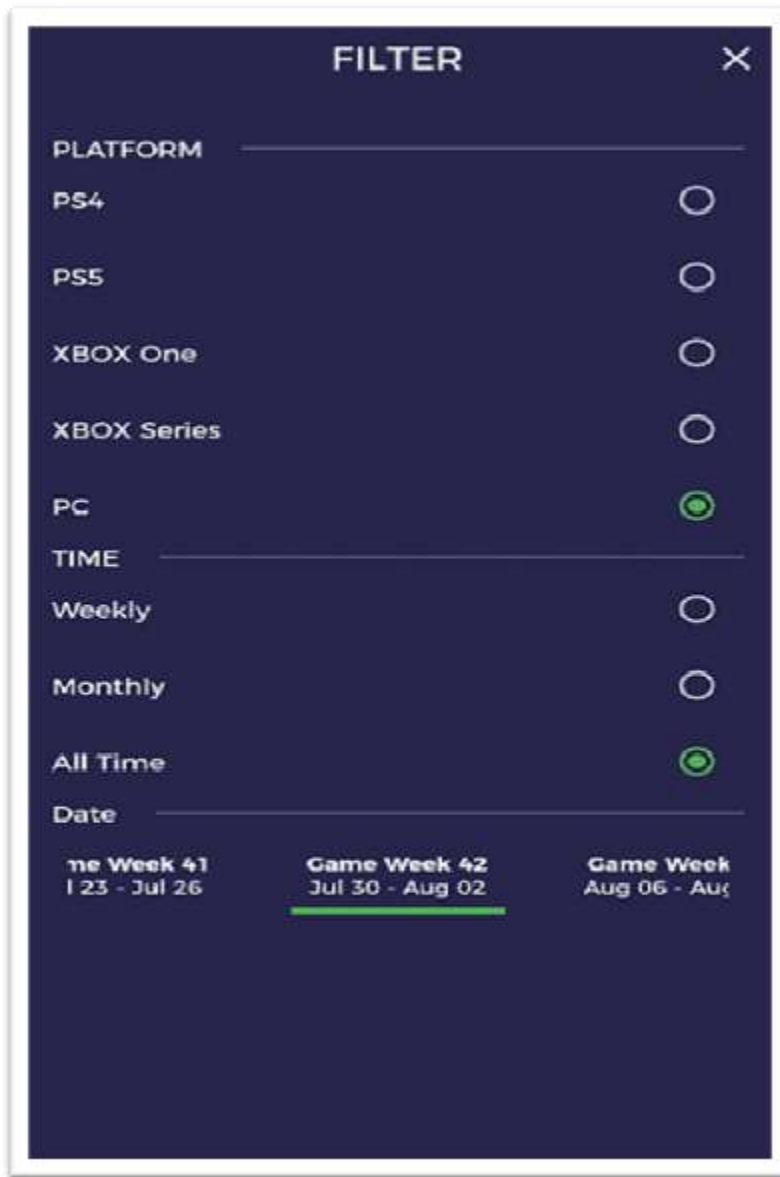


Fig. 6.3.4 Filter Screen

Filter screen contains the filters related to –

- Platform in which user can select it's platform like PS4, PS5, XBOX One, XBOX Series, PC.
- Time in which user can select the time period of the statistics like weekly, monthly, all time.
- Date in which user can select the game week.



### 6.3.5 MY MATCHES SCREEN

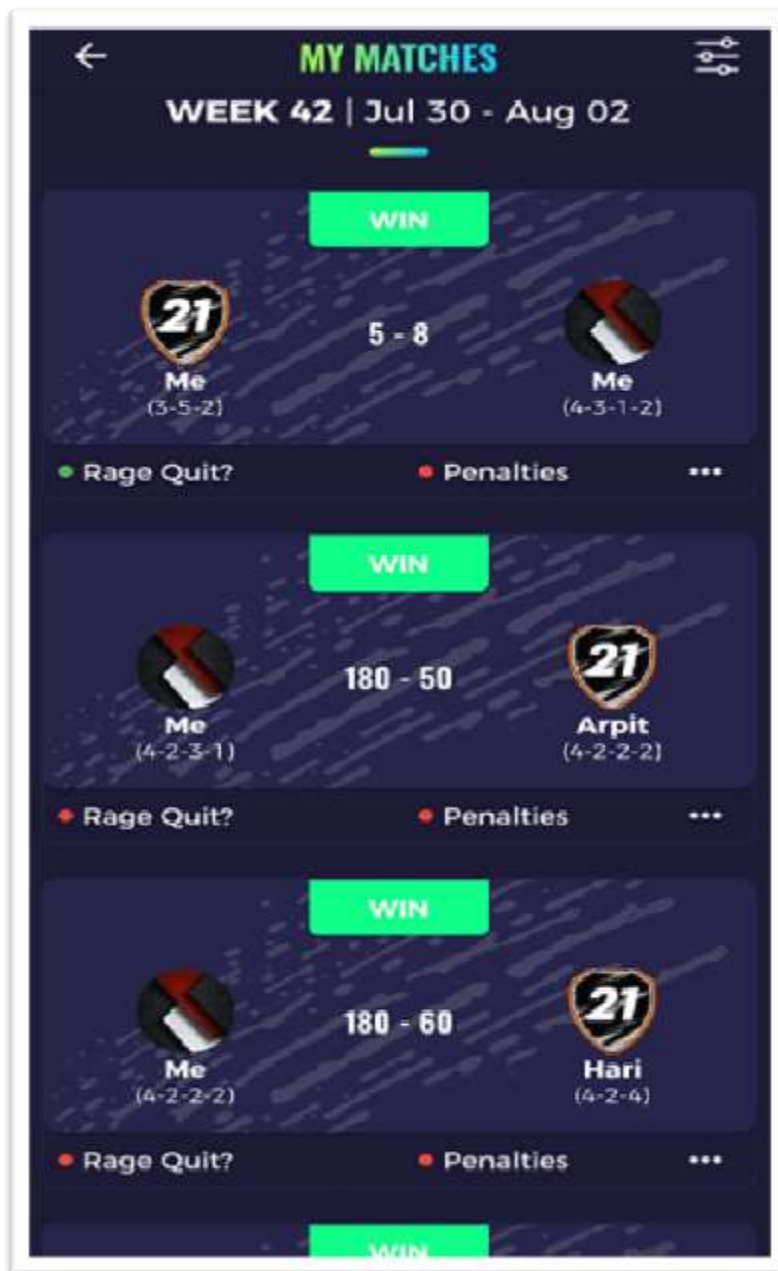


Fig. 6.3.5 My Matches Screen

My matches screen contains users all matches within the time period selected by the user in the filter. User can also edit the match details from this screen and change the match information.

### 6.3.6 FORMATION SCREEN

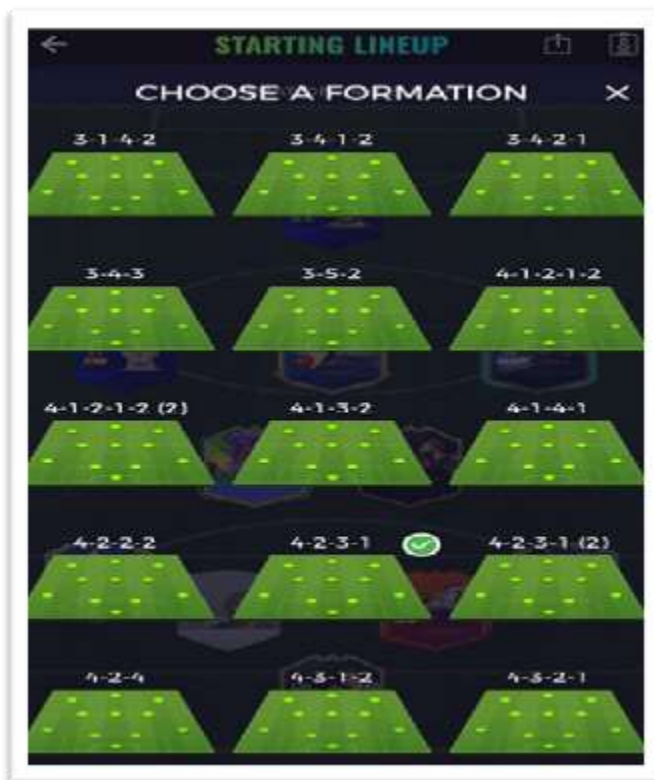


Fig. 6.3.6 Choose Formation 1



Fig. 6.3.6 Choose Formation 2

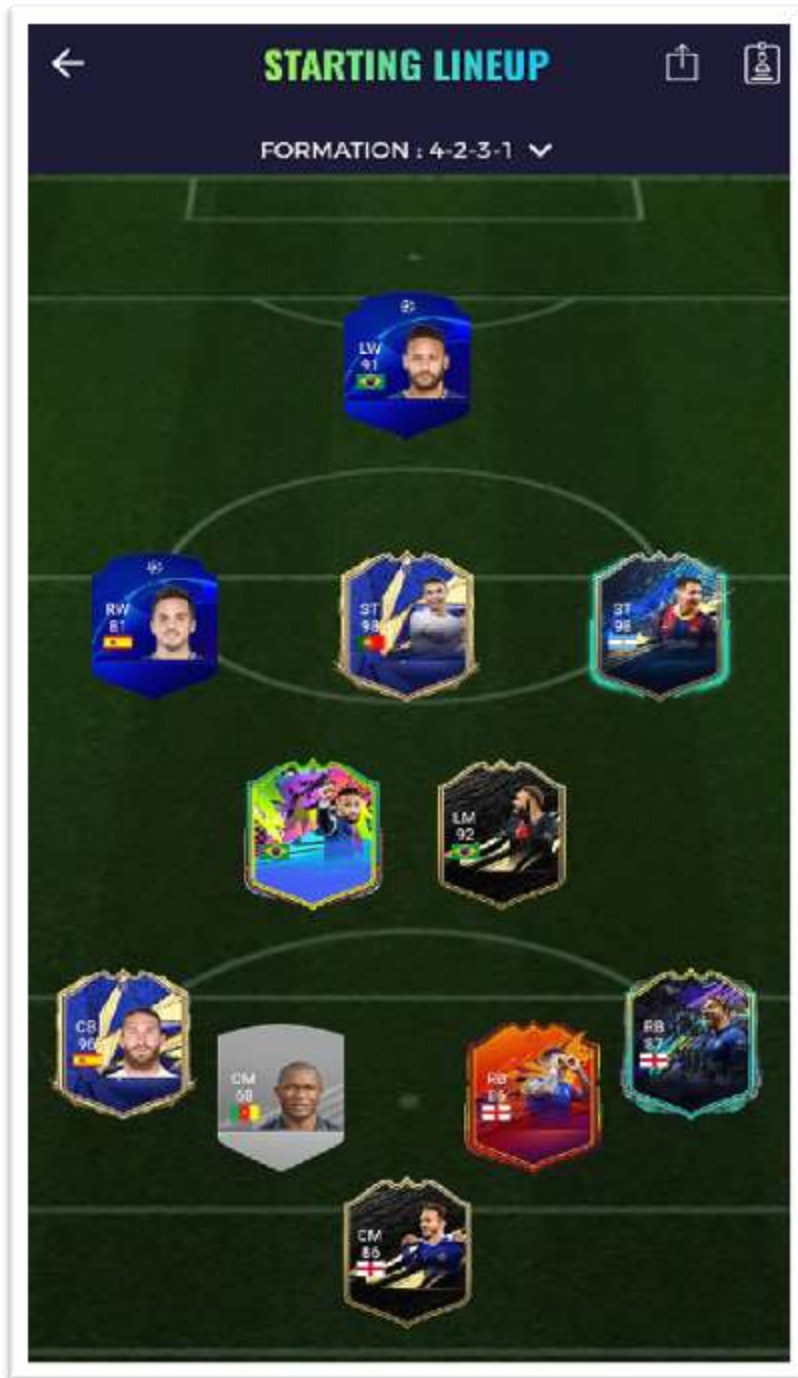


Fig. 6.3.6 Formation Screen

Formation screen helps the user to see the current formation and also select the new formation from the given formation list and after selecting formation players formed according to the formation.

### 6.3.7 PLAYER STATS SCREEN



Fig. 6.3.7 Player Stats Screen

Player stats screen contains the player statistics within the time period selected by the user. This screen contains goals scored in total matches and also stats about POTM, super sub, assists.

### 6.3.8 MY CLUB SCREEN



Fig. 6.3.8 My Club Screen

My club screen contains information about the current club of the user and the home and away kits of the club. User can also select the starting lineup of the club.

### 6.3.9 WEEKEND LEAGUE RECAP SCREEN



Fig. 6.3.9 Weekend League Recap Screen

Weekend league recap screen contains the information about the user lineup, user selected club and kits, all wins and loss, record about the match played and no. of wins and loss, goal summary in the form of your goals and against goals, rage quits of yours and against, no. of wins and losses in home vs away, no of times wins and losses based on extra time, no of clean sheets and some more information related to the players performance.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE WORK**

Fut-tracker application helps to track the matches of the football matches and shows the different statistics of the players. Users can also track the matches and edit the matches if they have wrongly entered the information.

Fut-tracker application has the leaderboard that helps the users to know their standings and can see their wins, losses and points.

There is a formation section in this application through which user can select the formation from the available formation and then the players are arranged on the screen according to the selected formation. Now we can only change the player in the formation when user select the formation but if user have to switch the player than reselect the players on those positions. So we can implement the drag and drop feature to switch the player position without reselection.



## REFERENCES

1. Marco L. Napoli, *Beginning Flutter®: A Hands On Guide To App Development*, 16 September 2019.
2. Qiu, M., Gai, K., & Dai, W. (2016). *Mobile Applications Development with Android: Technologies and Algorithms* (1st ed.). Chapman and Hall/CRC.
3. J. F. DiMarzio, *Beginning Android® Programming with Android Studio*, Fourth Edition, 24 October 2016.
4. Reto Meier, Ian Lake, *Professional Android®, Fourth Edition*, 27 August 2018.
5. Misra, A., & Dubey, A. (2013). *Android Security: Attacks and Defenses* (1st ed.). Auerbach Publications.
6. Tyagi, P. (2021). *Pragmatic Flutter: Building Cross-Platform Mobile Apps for Android, iOS, Web, & Desktop* (1st ed.). CRC Press.
7. Mostefaoui, G.K., & Tariq, F. (Eds.). (2018). *Mobile Apps Engineering: Design, Development, Security, and Testing* (1st ed.). Chapman and Hall/CRC.
8. Ilyas, M., & Ahson, S.A. (Eds.). (2010). *Mobile Web 2.0: Developing and Delivering Services to Mobile Devices* (1st ed.). Auerbach Publications.
9. Rob Huddleston, *Android® Fully Loaded*, 3 November 2011.
10. Dominic Chell, Tyrone Erasmus, Shaun Colley, Ollie Whitehouse, *The Mobile Application Hacker's Handbook*, 12 November 2014.
11. Anne-Marie Lesas, Serge Miranda, *The Art and Science of NFC Programming*, Volume 3, 30 December 2016.
12. Wiem Tounsi, *Cyber-Vigilance and Digital Trust: Cyber Security in the Era of Cloud Computing and IoT*, 24 April 2019.
13. Glaser, J.D. (2014). *Secure Development for Mobile Apps: How to Design and Code Secure Mobile Applications with PHP and JavaScript* (1st ed.). Auerbach Publications.
14. Eric Windmill, *Flutter in Action* (2017).
15. Alessandro Biessek, *Flutter for Beginners: An Introductory Guide to Building Cross-platform Mobile Applications with Flutter and Dart 2* (2019).
16. Laurence Moroney, *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform*, 10 November 2017.