ABSTRACT

Nowadays, the use of mobile technologies is rising at an alarming scale. Due to this, more powerful and efficient mobile applications are needed in order to keep up with this trend. Since there exists several mobile platforms (iOS, Android, etc.), each one with different SDK (Software Development Kit) tools and specific development capabilities, application development becomes more complicated and expensive. The challenge is to come up with a solution that allows us to deploy in different platforms using a single SDK tool and maintaining the same performance as the native application. A suitable solution is cross-platform. In this, we present a survey of cross- platform creation approaches with an emphasis on the MDA (Model Driven Architecture) approach as it is one of the most promising cross platform approaches. We also identify and discuss the main desirable requirements of any cross-platform technology.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Mobile devices have become an important platform for today's software applications. Especially, the utilization of smartphones increased rapidly within the last couple of years . Since smartphones are often utilized to consume or orchestrate services, this process includes a vast range of applications. Smartphones also connect to other domains such as the Internet of Things (IoT) and often utilize smart cloud-based services .

The introduction of smartphones rapidly increased the need and development of mobile software. The development of mobile software applications is a special case of software engineering. Mobile applications are often also referred to as apps, which implies that the application is intended to be used on a smartphone or wearable device . Thus, development must cope with specific aspects such as: short application lifecycles, limited device capabilities, mobility of users and devices, availability of network infrastructure as well as security and privacy issues . The difference in devices also generates a variety of different resolutions and display sizes. While developers are enacted to create and distribute applications in a large scale, they also have to deal with these inherent differences and limitations of mobile devices (i.e.,battery life or small displays). Furthermore, it is necessary to address different operating systems (especially for smartphones, and, to a limited extent, for feature phones as well). Since the market for smartphones has consolidated recently, some operating systems (i.e., Windows Phone, BlackberryOS and other OS hold a market share of 3.2%) vanished again. Still, to address the smartphone market, applications for both, Android (market share: 72.4%) and iOS (market share: 24.4%) need to be provided. In addition, Android is split into different versions, manufacturers and various system customizations. Despite vendor customization and just considering the Android version, current most widely used is Oreo (8.0 and 8.1 with 28.3%), followed by Nougat (7.0 and 7.1 with 19.2%) and finally, the latest version Pie (9; with 10.4%)

Currently,generic approaches can be further subdivided into Web and hybrid applications (see Figure 1). Web applications can be used virtually under any platform, as a Webbrowser is preinstalled on almost all devices. The most salient advantage is application portability, which basically comes at no cost. Web apps are typically optimized by means of Hyper Text Markup Language (HTML5), Cascading Style Sheet(CSS) and JavaScript . Numerous frameworks (such as Angular, Bootstrap, React or Vue) provide additional functionality on top of Web standard technologies and help to speed up development of Webapps.

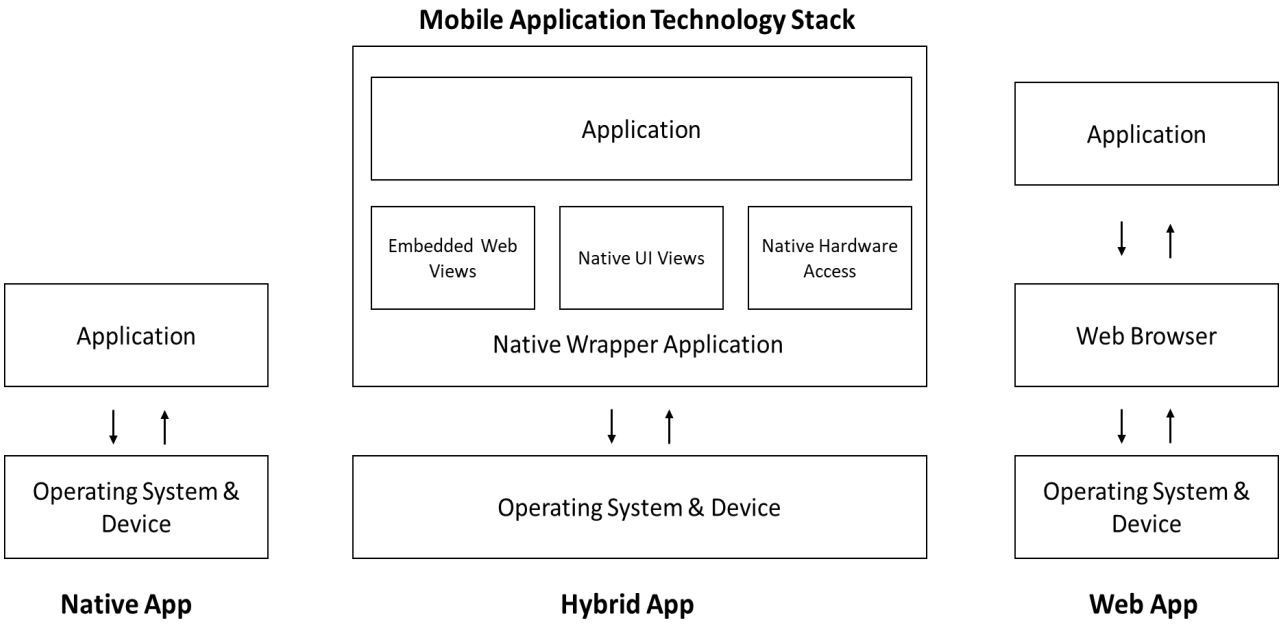


Figure 1.1: Mobile App Technology Stack.

Hybrid applications are built on frameworks such as Apache Cordova or Adobe PhoneGap. Often they rely on Web technologies also, and enact access to native device functions and sensors. Hybrid apps utilize a specialized browser to present the user interface (UI). This results in a presentation layer, which is identical or very near to widgets used in native apps. Today's hybrid framework technologies are mainly ex-tensions of Cordova and PhoneGap, as they extend and sim-plify the development of cross-platform applications.

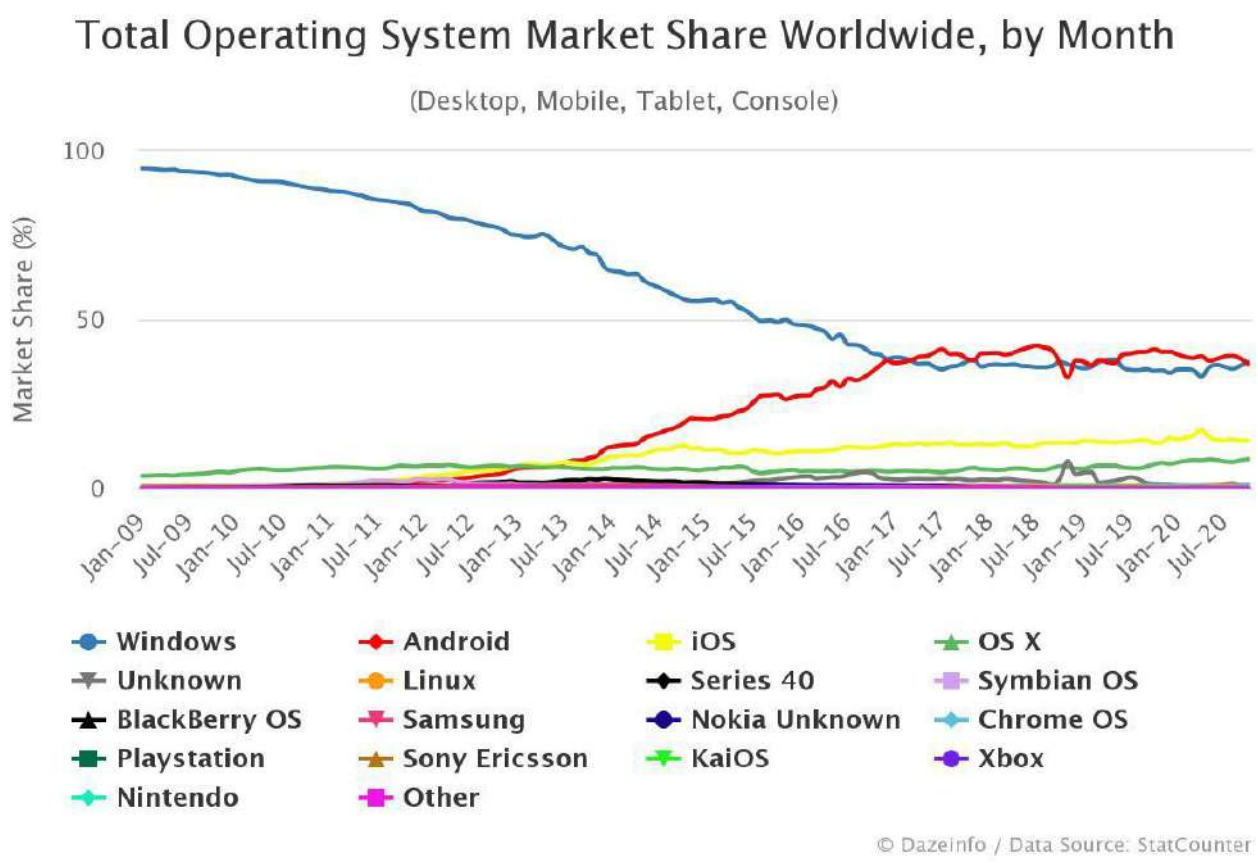


Figure 1.2: Worldwide Operating System Share 2021

Issues of supported functionality, performance and the ge-neric question of maintenance of cross- platform applications lead us to the evaluation of multiple cross-platform frame-works. With this paper we want to record the current state of the art in the development of cross-platform apps. In addition,we want to uncover innovations and differences that arise with the deployment of new frameworks and versions. We achieve this with our reference app. The remaining parts of this article are structured as follows: First, Section II introduces the liter-ature research method. This work can be seen as a starting point for further work on the article. Section III provides an overview of current mobile app development. In Section IV, a reference architecture is presented and five framework-based implementations of this architecture are discussed. The reference implementations are being evaluated in Section V. Finally, Section VI outlines the conclusion and outlook for further research.

Chapter 2

LITERATURE SURVEY

A comprehensive literature research was initiated to iden-tify significant literature on the topic of cross-platform devel-opment and cross-platform frameworks. In addition, im- portant scientific articles from the same field of research should be identified, which will form the basis for the later main part of the article. For this purpose, the proven literature research of Brocke et al. was used. A five-phase model is defined by Brocke et al., which enables a systematic literature search. In the process, they combine approaches from Cooper and Webster & Watson . The method, the five-phase process is shown in Figure2.

Phase I defines the different characteristic values of liter-ature research. We apply the taxonomy of Cooper in this phase as a basis. The applied taxonomy according to Cooper is explained below and also shown in TABLE I. The focus of this article is on research outcomes in the area of cross plat-form technologies. In addition, the practical application of these technologies plays an important part in our research. If the technologies in focus are not applied in a broad common sense, they might lack in one or more aspects (e.g., they are just not well-known, they are error prone, they do not offer simplified multi-platform deployment, they do not support well supported programming languages or constructs or they are simply not applicable to specific project requirements).

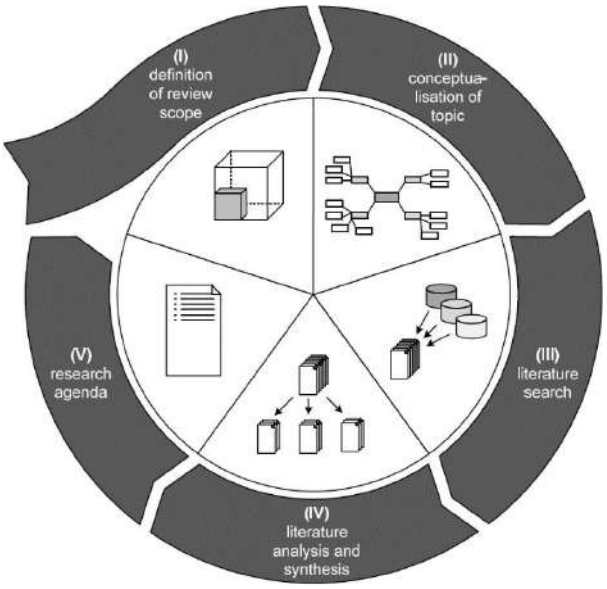


Figure 2.1: Literature review framework used for literature research according to Brocke et al.

CROSS PLATFORM DEVELOPMENT

As stated above, there are several approaches for cross-platform development. This type of development is subject to typical challenges of ubiquitous computing. In addition, fur-ther challenges are typical to cross-platform development the most important issue being associated with:

1. User Interface (UI)
2. Limited Resources
3. Device Management
4. Application Maintenance

The design of UI is associated with questions of simplicity and intuitiveness. For mobile cross-platform development, this is extended by design guidelines defined by the different operating systems. It is further restricted because of different device capabilities (e.g., screen sizes and resolution) . Limited resources is a typical issue in mobile software engi-neering; for cross-platform development the application size and resource consumption (especially power and memory management) is a typical issue . Since cross-platform development addresses a vast variety of devices, their man-agement in terms of appropriate usage of hardware and sen-sors (i.e., CPU, memory, Bluetooth, or camera) becomes an-other typical challenge. Furthermore, different operating sys-tems must be handled as well. Finally the application has to be maintained by following short lifecycles of devices, oper-ating systems and frameworks .

A lot of different methods that address cross-platform de-velopment can be observed in science and industry. Some are based on model-driven software engineering . The ad-vantage of model-driven methods is that developers and users, which are less familiar with specific programming paradigms are enabled to efficiently implement applications. As Object Management Group (OMG) standard, the Interaction Flow Modeling Language (IFML) offers model-based and plat-form-independent development of applications for different types of devices. Following the Model-Driven Architecture (MDA) it is based on a meta-model and it is built upon Web Modeling Language (WebML). A Web-based and an eclipse-based modelling environment is provided for IFML. Further-more, extensions for Apache Cordova and PhoneGap are pro-vided . An open challenge is to keep the extensions up-to-date. Other solutions, such as WebView, utilize native code and combine it with Web technologies. Native compo-nents are used as containers to render Web pages that contain application logic and presentation layer definitions. Native components serve to access device-specific functions (i.e., push notifications or sensor data).

We observe three general approaches to cross-platform development:

1. Native application
2. Transformation- or generator-based application
3. Interpreted application (parser-based)

With native development, an application is developed for each specific device (and operating system). Benefits include the native look and feel, the ability to use all platform-specific features and a comparatively high performance of the app. The most prevalent disadvantage is high efforts for development and maintenance.

TABLE I: COMPARISON OF FRAMEWORK POPULARITY.

Source	Unit	Ionic	Xamarin	React Native	Oracle Jet	Flutter
StackShare	Stacks	2.360	461	3.880	N/A	105
	Fans	2.310	549	3.740	N/A	156
	Jobs	98	47	726	N/A	5
	Votes	1.660	646	821	N/A	49
Hacker News	Posts	669	995	917	8	N/A
Reddit	Posts	989	1.080	N/A	55	1.080
Stack Overflow Stats	Posts	4.430	35.000	45.100	280	9.890
GitHub Stats	Stars	36.700	N/A	73.200	295	51.400
	Forks	12.800	N/A	16.200	81	5.360

The latter is a result of redundancy in code and support because each platform has to be served by a separate applica-tion. The use of generators employs a meta-implementation, which is then transformed to specific platforms (e.g., as used in Cordova or Ionic). Similarly, model-driven development approaches (such as IFML) may use transformations to pro-duce platform specific code. An advantage is that the applica-tion logic is platform agnostic. Applications, which are interpreted rely on some kind of parser. The parser interprets application code during runtime in order to create platform specific instructions. Fabrik19 utilizes an interpreted approach in its Mobility Suite (MOS) framework.

CROSS PLATFORM FRAMEWORKS

As discussed above, there are a lot of cross-platform frameworks like IFML, Cordova, Corona Software Develop-ment Kit (SDK), Appcelerator Titanium, TheAppBuilder, PhoneGap, Native Script, SenchaTouch, Framework7, Apache Weex, Flutter, Oracle Jet, Jasonette or Manifold – also see [9]. All of them utilize one or a combination of the three methods to create platform specific applications. In our eval-uation we also regard the popularity of the different frame-works. We consider communities like StackShare, Hacker News, Reddit, Stack Overflow Stats as well as GitHub Stats. In our comparison, we strive to evaluate the most frequently used and most progressively developed frameworks as illus-trated in TABLE II. Ionic offers a generator-based approach . The frame-work is free to use and available as open source. Additionally, several services are available via pay on demand. The gener-ator utilizes a Web application as input. Thus, development of cross-platform applications is based on Web technologies (Ja-vaScript/TypeScript, HTML5 and CSS; see Figure 3). Ionic also relies on Angular [22] in order to foster component based development and reuse of templates. Ionic officially supports Android, iOS and Universal Windows Platform (UWP) . Since Ionic is based on Web applications that are generated into platform specific applications through Apache Cordova, these source applications may also be executed in any Web browser. Native operating system functions and access to sen-sors is only available after generation of platform specific code. The utilization device specific functionalities often also rely on plugins that have to be declared as dependency .

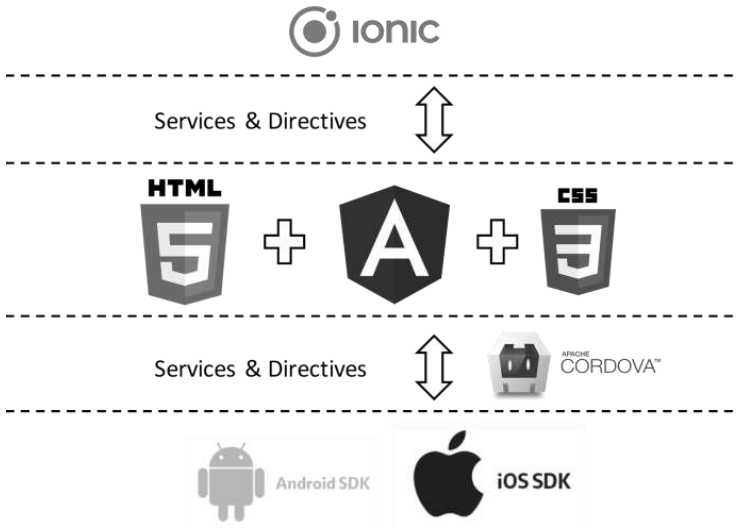


Figure 2.2: Ionic Architecture

Xamarin

is another framework to develop cross-platform apps for Android, iOS and UWB . Other platforms such as Linux are not supported and macOS support was recently added with the launchofXamarin.Mac.

Xamarin is based on .Net and utilizes C# as programming language. Xamarin is divided into two major parts: 1) Xama-rin platform and 2) Xamarin.Forms. The Xamarin platform (Xamarin.Android, Xamarin.iOS) provides APIs to share code for application logic between all platforms. The UI is written individually for each platform. Xamarin.Forms allows to create additional platform-independent UI, which are mapped into native UI in a second step . The development environment is based on Visual Studio (or Xamarin Studio for macOS).

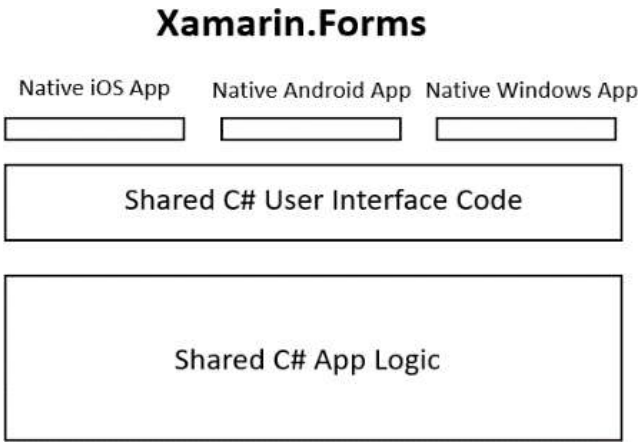


Figure 2.3: Xamarin Architecture

The advantages include: 1)it uses the C# programming language, which is quite similar to Java, so that the Android Native developers can rapidly study (Al-Bastami & Naser, 2017); 2)it is integrated with Visual Studio (Radi, 2016), which is the most powerful Integrated Development Environment (IDE) on Windows; 3)it can generate high-performance programs in an experience similar to native development (Willocx et al., 2016); and 4) it supports downloading components from the NuGet extension (Martinez, 2018), which accelerates the developing speed significantly. According to Avdic (2019), the disadvantages include: 1) it cannot support the latest native framework in time; and 2) it is not stable.Although Flutter was considered better than React Native in UI performance (Jagiełło, 2019) and Xamarin was reflected as efficient as Android Native in terms of encryption processing speed (Dobrzański & Zabierowski, 2017),

React Native

Is a parser based open-source framework for building cross-platform applications . It is based on React. Both frameworks are being developed by Facebook. React Native currently supports Android and iOS and uses a Na-tiveScript Runtime environment to execute the application code . However, with a little more effort, it is also possible to deploy to UWP. Since React is built on JavaS-cript, this holds true for React Native as well. React Native invokes Objective-C APIs to render to iOS components and Java Application Programming Interface (APIs) to render to Android components. This means that no code generation is utilized in React Native. Facebook promises that the perfor-mance of apps would be almost as good as that of native ap-plications. Components for React Native may either be built as functional components or class components .

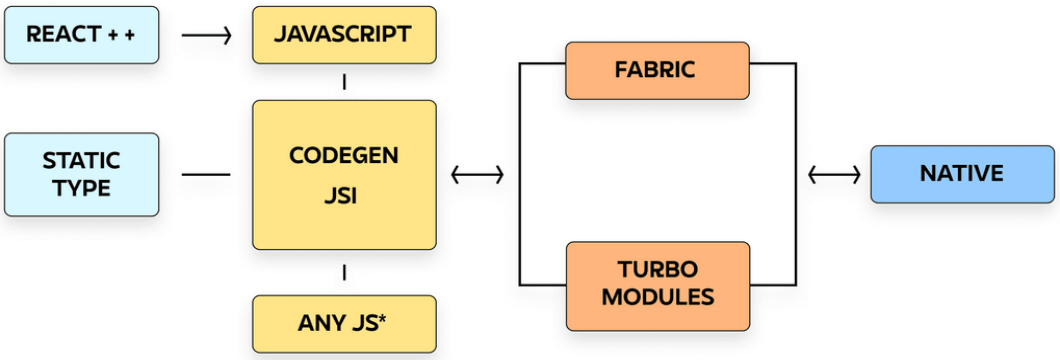


Figure 2.4: React Native Architecture.

React Native enables developers to write native mobile applications only using JavaScript programming language. It is consistent with React in terms of design principles and declarative component mechanism in building a rich User Interface (UI).

The advantages include: 1) it uses JavaScript to composite various components; 2) it transforms mark-up elements into native UI elements; 3) it separates the working thread from the main UI thread, which results in a better application performance with full functionality; and 4) it significantly saves time in terms of development and maintenance.

Oracle Jet

Follows a similar philosophy as Ionic and was developed in the software house Oracle. The framework and its tools are freely available as open source. Oracle Jet also uses the generator-based approach. Web technologies (JavaS-cript, Knockout, jQuery, HTML5 and CSS, etc. are used for the development.

Platform-specific versions can be derived from the web application. Apps for Android, iOS and UWP can be officially released. Also, an execution in the browser is possible without further ado, particularly since the application is converted by means of Web technologies. For the use of platform-specific sensors, Cordova plug-ins must be used .

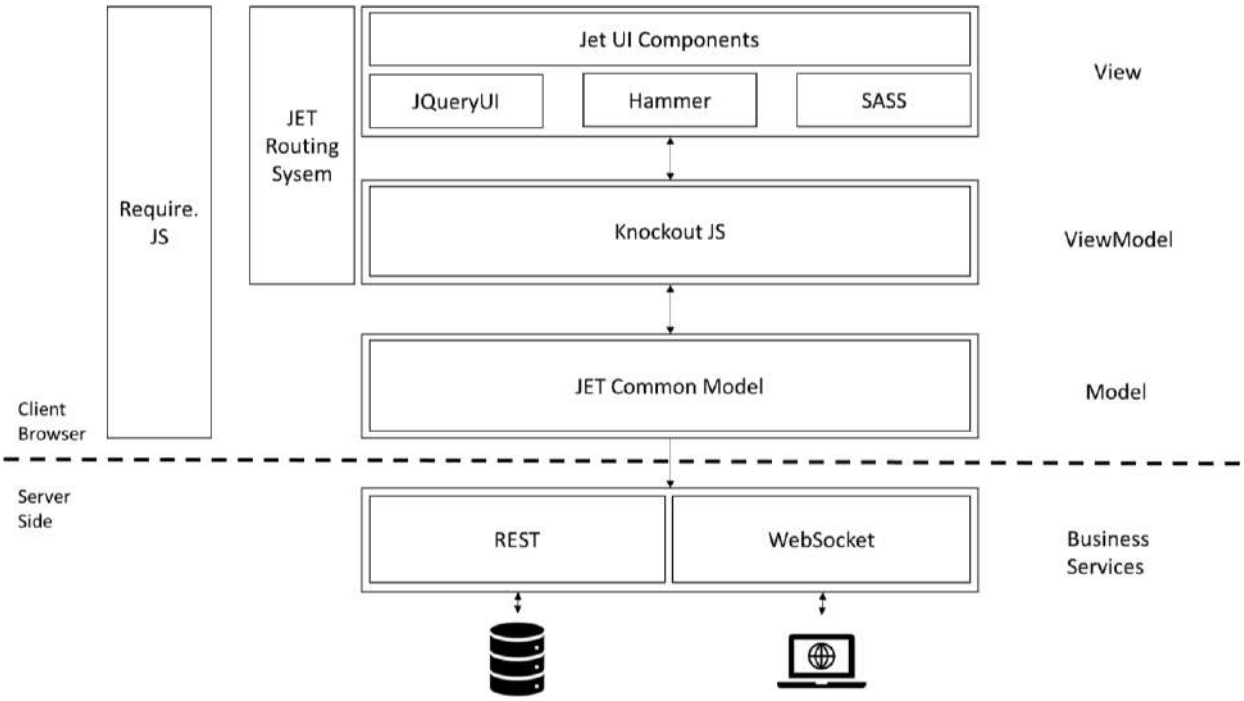


Figure 2.5: Oracle Jet Component Architecture

Oracle Jet offers several components for displaying data. data can also be displayed in a GridView. Layout – Tab: Oracle Jet offers ready-made templates for tabs that can be reused. Access system time: Oracle Jet also uses JavaScript tech-nologies, so the time query is similar to Ionic (date().getHours(). Access current position (GPS): Ora-cle supports the use of Cordova plug-ins. After installing the Cordova plugin Geolocation, it can be integrated into the pro-ject. As can be seen in Listing 4, the values are stored in var-iables latitude and longitude after a successful determination of the location.

Flutter

Is a parser based, open source SDK, which offers an easy way to develop high-fidelity and high-performance mobile apps for android and iOS. Flutter uses a rich set of Cu-pertino (iOS) and Material Design behaviours and widgets. Furthermore, Flutter implements platform-specific code like navigational patterns, fonts and more. Flutter apps are written in Dart. Its syntax looks a lot like Java, JavaScript, C# or swift. Dart uses the standard Android and iOS toolchains to compile your code . Flutter does not separate views, controllers, layouts and other properties like other frameworks. It uses one consistent, unified object model, so called widgets. Widgets can define structural elements (like buttons or menus), stylis-tic elements (like fonts or colour schemes, aspects of layouts (like margins) and so on . Messages between the client (UI) and the host (platform) are passed using platform chan-nels as illustrated. These messages and their re- sponses are passed asynchronously. This way the user inter-face will remain responsive.

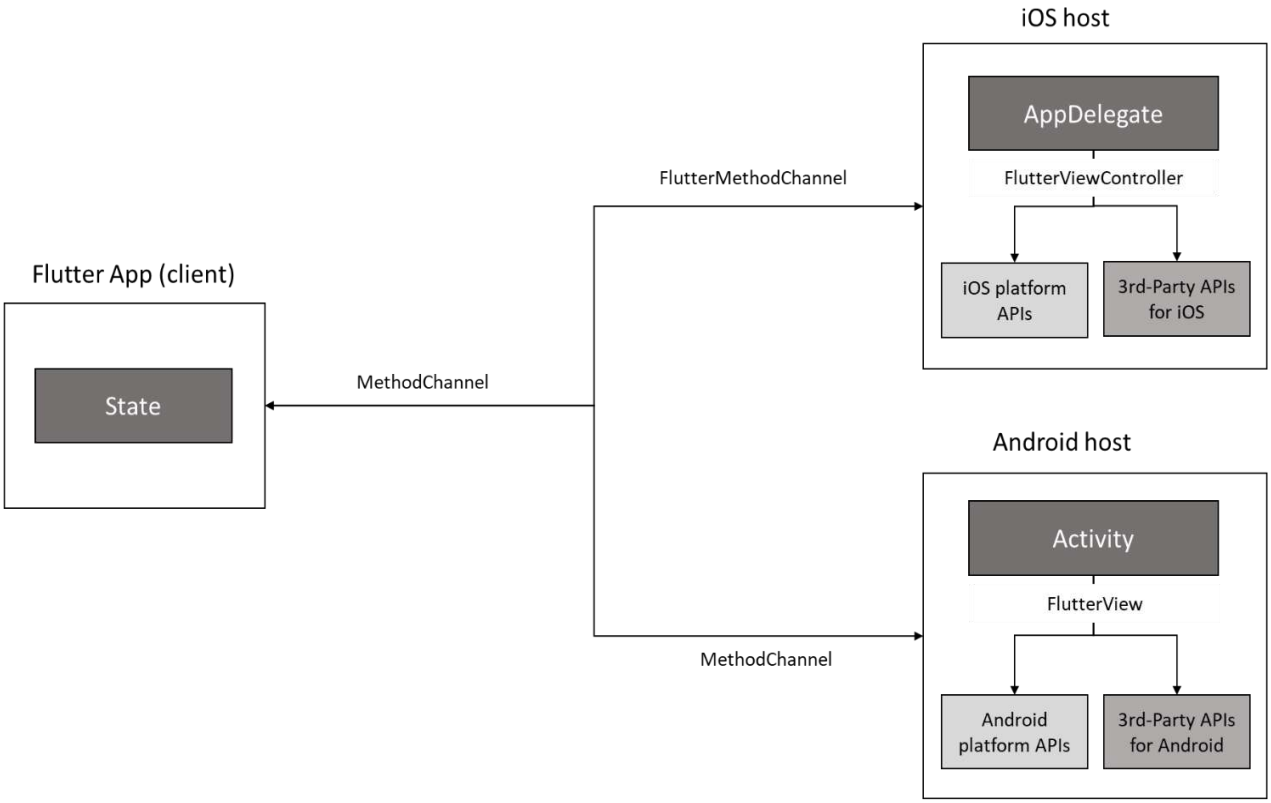


Figure 2.6: Flutter Architecture

To use the GridView in Flutter with Dart you define a new GridView.count(). CrossAxisCount defines the amount of children in a Row and childApsectRatio their size.

Chapter 3

PROBLEM STATEMENT

Cross-platform development is the practice of developing software products or services for multiple platforms or software environments. Engineers and developers use various methods to accommodate different operating systems or environments for one application or product.

EXISITING SYSTEM

Some frameworks for cross-platform development are Codename One, Kivy, Qt, Flutter, Native Script, Xamarin, PhoneGap, Ionic, and React Native.

Disadvantages

-App overload

-Massive file size

-Weak IOS feature support

-Builds can randomly crash without any reason

PROPOSED SYSTEM

React native lets you create truly native apps and doesn't compromise users experiences. It provides a core set of platform agnostic native components.

Advantages

-React Native is community driven

-Maximum code reuse and cost saving

-Live reload

-Strong performance for mobile environments

-Modular and intuitive architecture similar to react

Chapter 4

SYSTEM REQUIREMENTS SPECIFICATIONS

Hardware Requirements:

- **System** : Intel i7.
- **Hard Disk** : 120 GB.
- **Monitor** : 21'' LED.
- **Input Devices** : Keyboard, Mouse.
- **Ram** : 8 GB.

Software Requirements:

- **Coding Language** : JavaScript
- **Operating System** : Android 10, IOS 13
- **Software Tool** : React Native

Chapter 5

HIGH LEVEL DESIGN

React Native Architecture

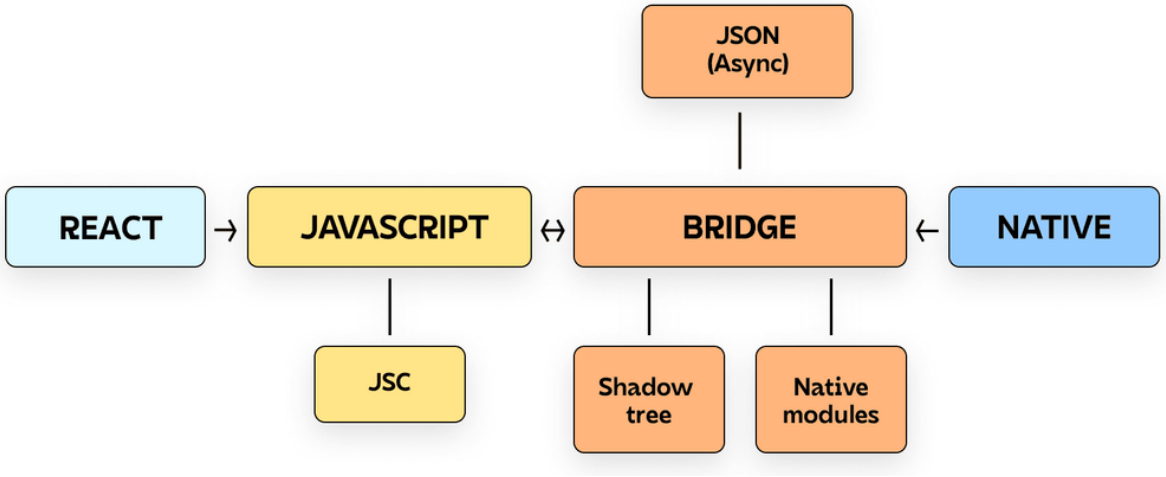


Figure 5.1 : React Native Architecture

In the current architecture, when a React Native application is run, the JavaScript code is bundled together into a package called JS Bundle, and the native code is kept separate. The JavaScript thread runs the JS Bundle, and the native/UI thread runs the native modules and handles UI rendering. The communication between the JS and native threads is enabled by a bridge, which sends data to the native threads after serializing as JSON. This bridge can only handle asynchronous communication.

With Fabric, logic is rendered in C++, which improves the interoperability between the host platforms. The Fabric renderer is implemented in C++, and the C++ core is shared among platforms, providing greater consistency and making React Native easier to adopt on new platforms.

Chapter 6

DETAIL DESIGN

SEQUENCE DIAGRAM

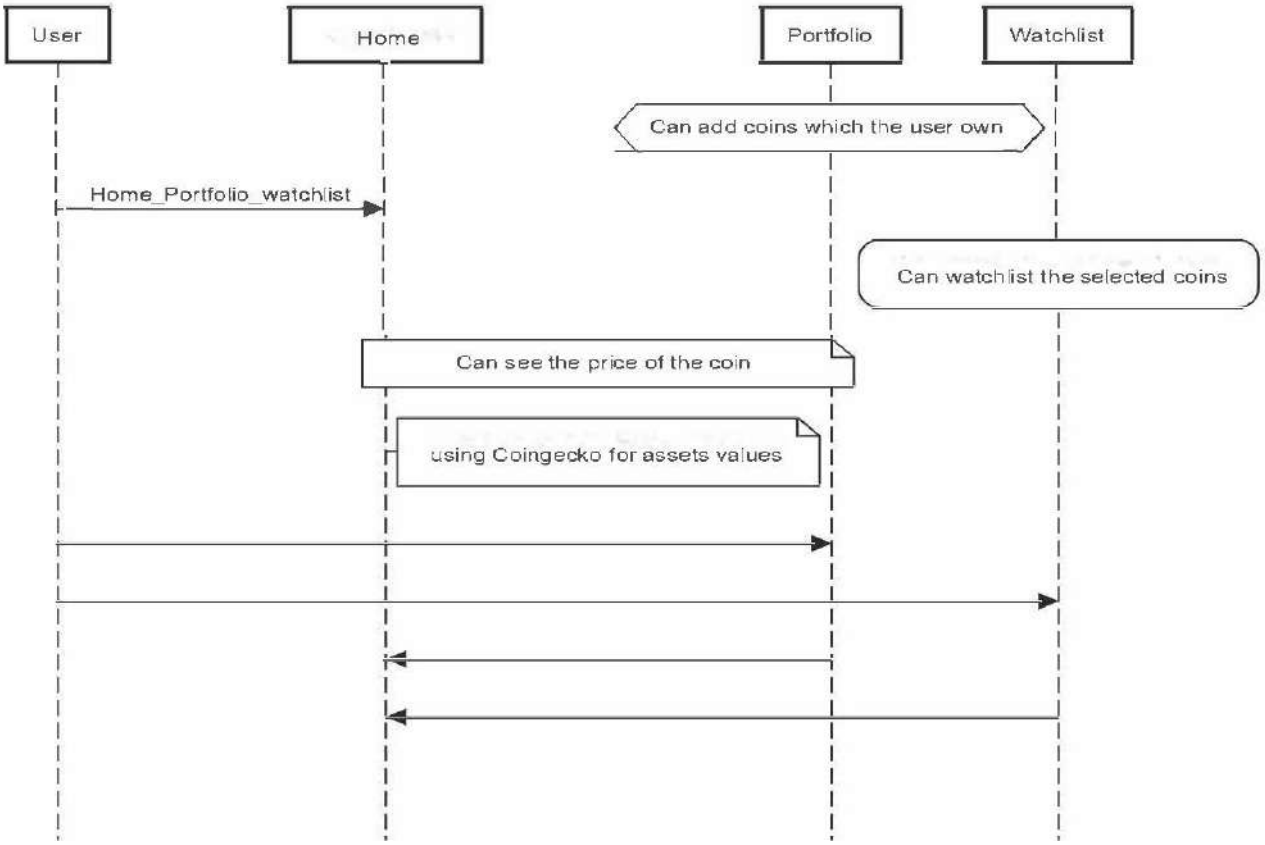


Figure 6.1 : Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.

DATA FLOW DIAGRAM LEVEL-0

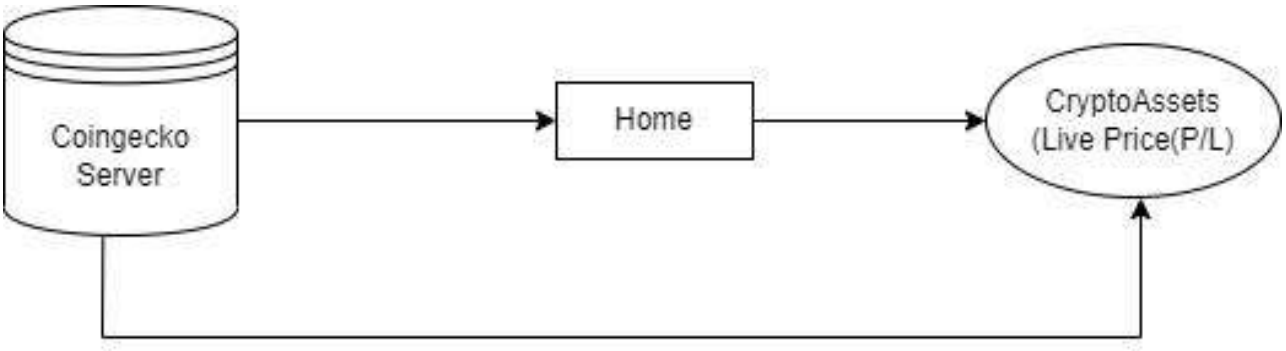


Figure 6.2 : Data Flow Diagram level-0

A data flow diagram (DFD) maps out the flow of information for any process or system.
A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. In Level-0, we have user navigating through the cryptoassets then in the cryptoassets it shows the price of the coin and the profit / losses of the coin for it's current day / time.

DATA FLOW DIAGRAM LEVEL-1

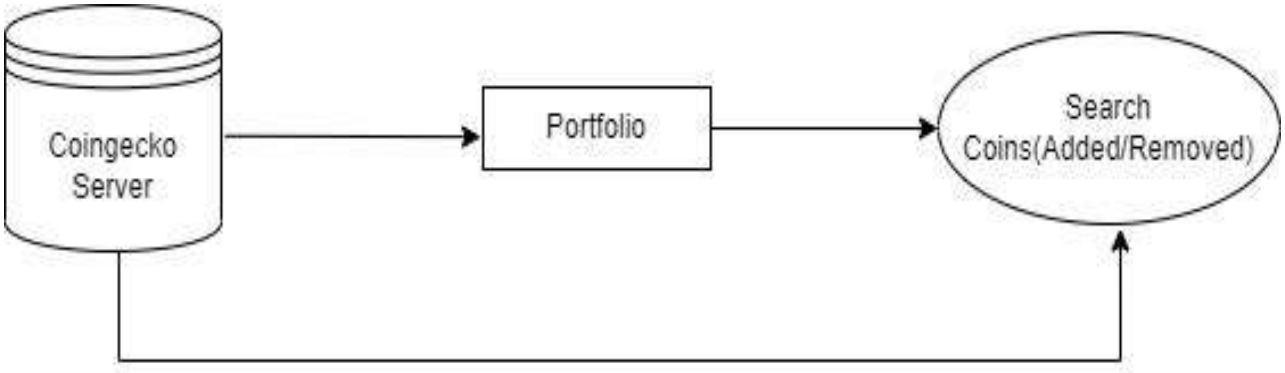


Figure 6.3 : Data Flow Diagram Level-1

Data flow diagrams show users how data moves from one process to another in a software system. In the Level-1, we have the live coin price shown and we have the marketcap live shown in the cryptoassets respectively.

DATA FLOW DIAGRAM LEVEL-2

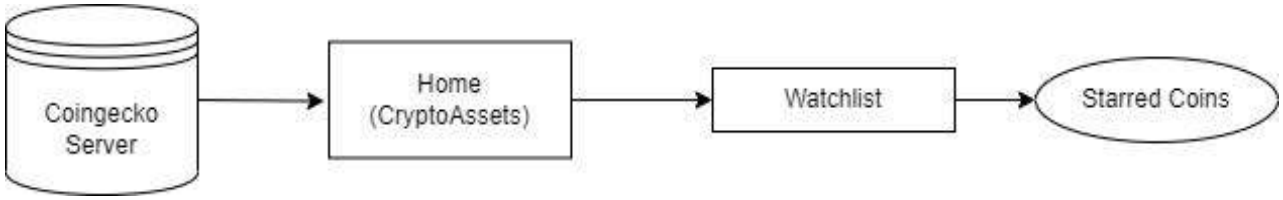


Figure 6.4 : Data Flow Diagram Level-2

In Level-2, after looking into different cryptoassets we can star the particular coins to the watchout in focus later in watchlist section and also we can add / remove assets from the portfolio and thereby we can see the current balance with profit / losses of the added coins in the portfolio section.

ACTIVITY DIAGRAM

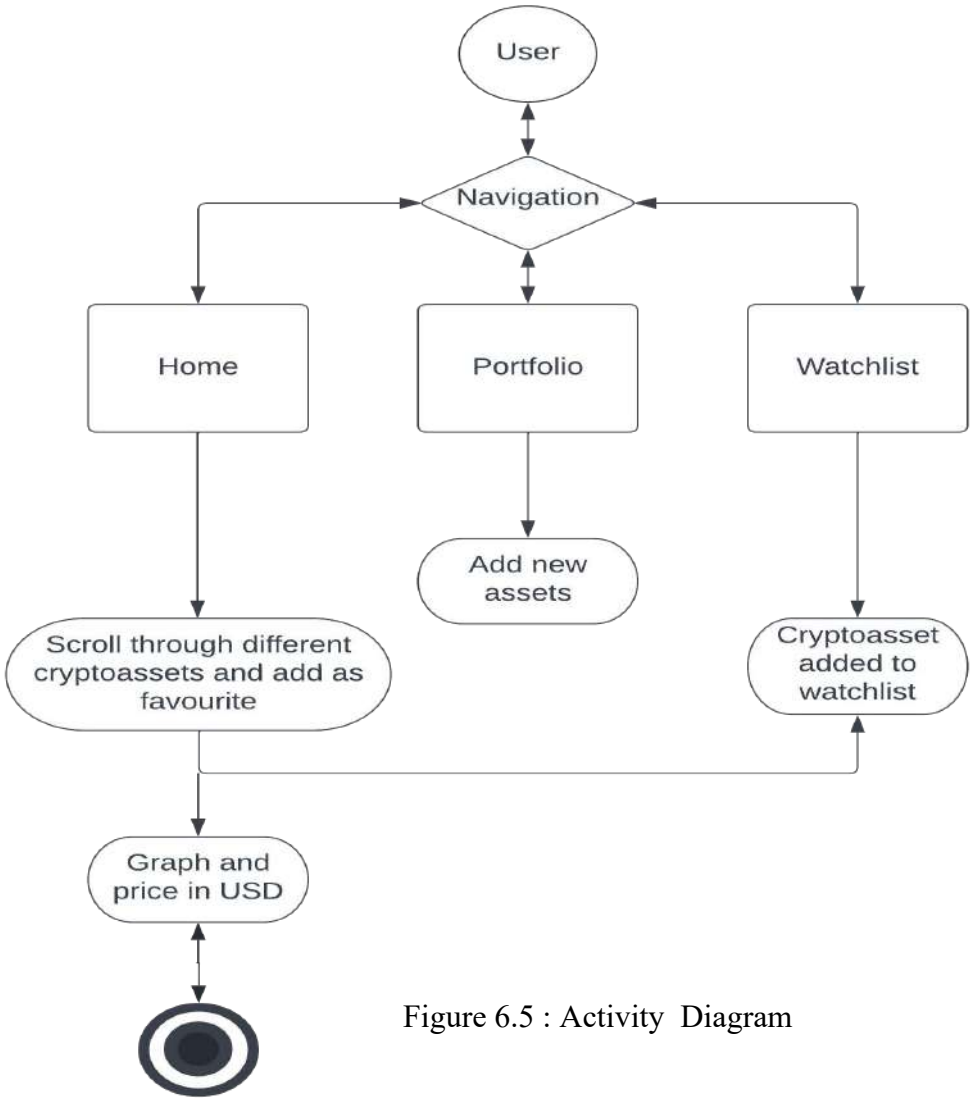


Figure 6.5 : Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows business and software processes as a progression of actions. These actions can be carried out by people, software components or computers. Activity diagrams are used to describe business processes and use cases as well as to document the implementation of system processes

USE CASE DIAGRAM

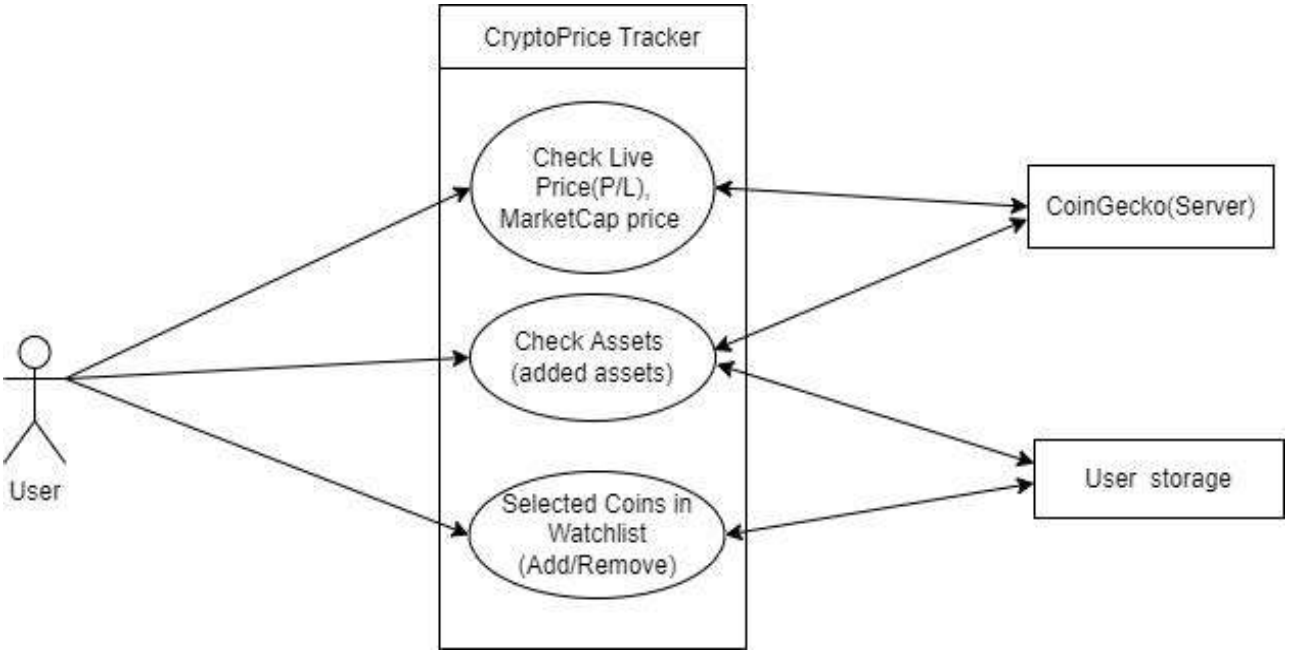


Figure 6.6 : Use case Diagram

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally First, you need to organize your four key elements — system, actors, use cases and relationships. Then, arrange them visually in a way that makes sense and will allow you to see immediately the connections between them.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams

CLASS DIAGRAM

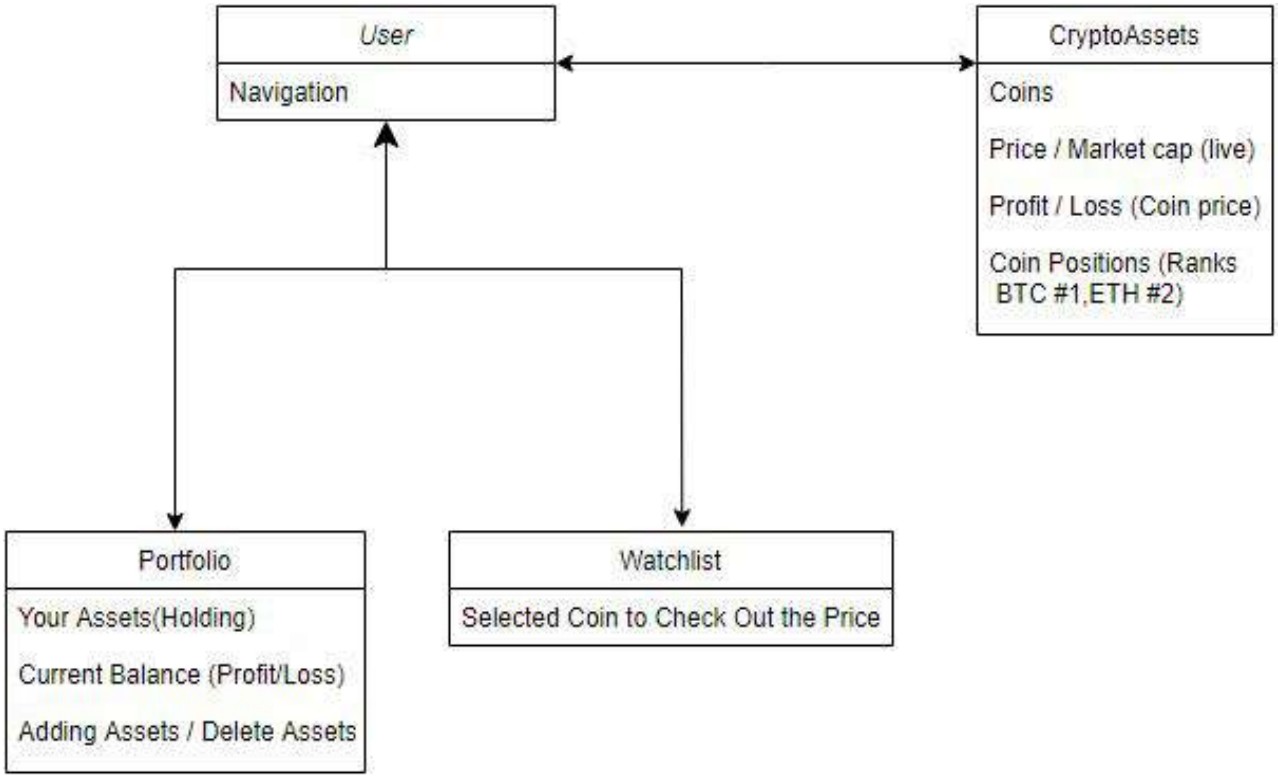


Figure 6.7 : Class Diagram

Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design An activity diagram shows business and software processes as a progression of actions. These actions can be carried out by people, software components or computers. Activity diagrams are used to describe business processes and use cases as well as to document the implementation of system processes

STATE DIAGRAM

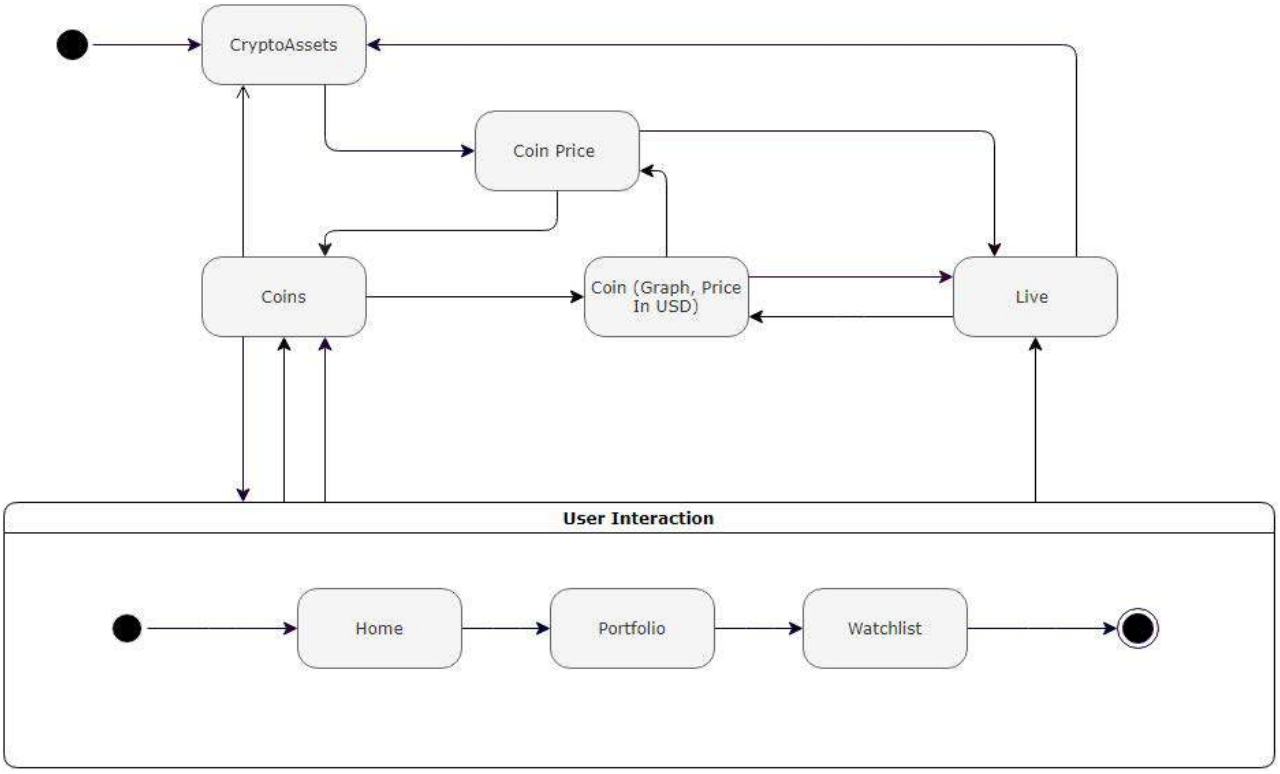


Figure 6.8 : State Diagram

A state diagram, also known as a state machine diagram or statechart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

Chapter 7

IMPLEMENTATION

SCREENS

1- HOME SCREEN
2- PORTFOLIO SCREEN
3- WATCHLIST SCREEN

HOME SCREEN

```
const HomeScreen = () => {
  const [coins, setCoins] = useState([]);
  const [loading, setLoading] = useState(false);

  const fetchCoins = async (pageNumber) => {
    if (loading) {
      return;
    }
    setLoading(true);
    const coinsData = await getMarketData(pageNumber);
    setCoins((existingCoins) => [...existingCoins, ...coinsData]);
    setLoading(false);
  };

  const refetchCoins = async () => {
    if (loading) {
      return;
    }
    setLoading(true);
    const coinsData = await getMarketData();
    setCoins(coinsData);
    setLoading(false);
  };

  useEffect(() => {
    fetchCoins();
  }, []);
export default HomeScreen;
```

PORTFOLIO SCREEN

```
const PortfolioScreen = () => {
  return (
    <View style={{ flex: 1 }}>
      <Suspense
        fallback={<Text style={{ color: "white" }}>Loading Please Wait!</Text>}
      >
        <PortfolioAssetsList />
      </Suspense>
    </View>
  );
};

export default PortfolioScreen;
```

WATCHLIST SCREEN

```
const WatchlistScreen = () => {
  const {watchlistCoinIds} = useWatchlist();

  const [coins, setCoins] = useState([]);
  const [loading, setLoading] = useState(false);

  const transformCoinIds = () => watchlistCoinIds.join('%2C');

  const fetchWatchlistedCoins = async () => {
    if (loading) {
      return;
    }
    setLoading(true);
    const watchlistedCoinsData = await getWatchlistedCoins(1,
transformCoinIds());
    setCoins(watchlistedCoinsData);
    setLoading(false);
  };

  useEffect(() => {
    if (watchlistCoinIds.length > 0) {
      fetchWatchlistedCoins();
    }
export default WatchlistScreen;
```

CRYPTOASSETS


```
{
  "id": "bitcoin",
  "symbol": "btc",
  "name": "Bitcoin",
  "image": "https://assets.coingecko.com/coins/images/1/large/bitcoin.png?1547033579",
  "current_price": 56600,
  "market_cap": 1069714480946,
  "market_cap_rank": 1,
  "fully_diluted_valuation": 1189253232322,
  "total_volume": 39887193328,
  "high_24h": 58224,
  "low_24h": 55822,
  "price_change_24h": -1624.213037134927,
  "price_change_percentage_24h": -2.78958,
  "market_cap_change_24h": -31986972353.526978,
  "market_cap_change_percentage_24h": -2.90342,
  "circulating_supply": 18889168,
  "total_supply": 21000000,
  "max_supply": 21000000,
  "ath": 69045,
  "ath_change_percentage": -17.97915,
  "ath_date": "2021-11-10T14:24:11.849Z",
  "atl": 67.81,
  "atl_change_percentage": 83415.61931,
  "atl_date": "2013-07-06T00:00:00.000Z",
  "roi": null,
  "last_updated": "2021-12-02T18:01:15.039Z"
 },

 {
    "id": "ethereum",
    "symbol": "eth",
    "name": "Ethereum",
    "image": "https://assets.coingecko.com/coins/images/279/large/ethereum.png?1595348880",
    "current_price": 4502.25,
    "market_cap": 534290263116,
    "market_cap_rank": 2,
    "fully_diluted_valuation": null,
```

{
"id": "tether",
"symbol": "usdt",
"name": "Tether",
"image": "https://assets.coingecko.com/coins/images/325/large/Tether-logo.png?
1598003707",
"current_price": 1,
"market_cap": 75127349497,
"market_cap_rank": 4,
"fully_diluted_valuation": null,
"total_volume": 68820362750,
"high_24h": 1.01,
"low_24h": 0.986737,
"price_change_24h": 0.00293218,
"price_change_percentage_24h": 0.29397,
"market_cap_change_24h": 796964318,
"market_cap_change_percentage_24h":
1.07219, "circulating_supply":
75025151943.5406,
"total_supply": 75025151943.5406,
"max_supply": null,
"ath": 1.32,
"ath_change_percentage": -24.3167,
"ath_date": "2018-07-24T00:00:00.000Z",
"atl": 0.572521,
"atl_change_percentage": 74.90401,
"atl_date": "2015-03-02T00:00:00.000Z",
"roi": null,
{"last_updated": "2021-12-02T18:02:49.306Z"
"id": "solana",
"symbol": "sol",
"name": "Solana",
"image": "https://assets.coingecko.com/coins/images/4128/large/Solana.jpg?

"current_price": 230.7,
"market_cap": 70515847843,
"market_cap_rank": 5,
"fully_diluted_valuation": null,
"total_volume": 3899521126,
"high_24h": 235.46,
"low_24h": 219.85,
"price_change_24h": 3.41,
"price_change_percentage_24h": 1.49961,

{
"id": "cardano",
"symbol": "ada",
"name": "Cardano",
"image": "https://assets.coingecko.com/coins/images/975/large/
cardano.png?1547034860",
"current_price": 1.72,
"market_cap": 55171708911,
"market_cap_rank": 6,
"fully_diluted_valuation": 77424582226,
"total_volume": 3131408940,
"high_24h": 1.73,
"low_24h": 1.52,
"price_change_24h": 0.130281,
"price_change_percentage_24h": 8.19239,
"market_cap_change_24h": 4129444227,
"market_cap_change_percentage_24h": 8.09024,
"circulating_supply": 32066390668.4135,
"total_supply": 45000000000,
"max_supply": 45000000000,
"ath": 3.09,
"ath_change_percentage": -44.24615,
"ath_date": "2021-09-02T06:00:10.474Z",
"atl": 0.01925275,
"atl_change_percentage": 8839.35659,
"atl_date": "2020-03-13T02:22:55.044Z",
"roi": null,
"last_updated": "2021-12-02T18:02:24.494Z"
{
"id": "ripple",
"symbol": "xrp",
"name": "XRP",
"image": "https://assets.coingecko.com/coins/images/44/large/xrp-symbol-
white-128.png?1605778731",
"current_price": 0.979633,
"market_cap": 46287977021,
"market_cap_rank": 7,
"fully_diluted_valuation": 98034301573,
"total_volume": 3201673067,
"high_24h": 1.01,
"low_24h": 0.954466,
"price_change_24h": -0.028194388251,
"price_change_percentage_24h": -2.79754,

{
"id": "usd-coin",
"symbol": "usdc",
"name": "USD Coin",
"image": "https://assets.coingecko.com/coins/images/6319/large/
USD_Coin_icon.png?1547042389",
"current_price": 0.999468,
"market_cap": 39072423824,
"market_cap_rank": 8,
"fully_diluted_valuation": null,
"total_volume": 4432306610,
"high_24h": 1,
"low_24h": 0.990529,
"price_change_24h": 0.00153039,
"price_change_percentage_24h": 0.15336,
"market_cap_change_24h": 257757408,
"market_cap_change_percentage_24h": 0.66407,
"circulating_supply": 39046554395.5724,
"total_supply": 39049314941.7851,
"max_supply": null,
"ath": 1.17,
"ath_change_percentage": -14.73546,
"ath_date": "2019-05-08T00:40:28.300Z",
"atl": 0.891848,
"atl_change_percentage": 12.11581,
"atl_date": "2021-05-19T13:14:05.611Z",

{
"id": "polkadot",
"symbol": "dot",
"name": "Polkadot",
"image": "https://assets.coingecko.com/coins/images/12171/large/
aJGBjJFU_400x400.jpg?1597804776",

"current_price": 36.08,
"market_cap": 38332354563,
"market_cap_rank": 9,
"fully_diluted_valuation": null,
"total_volume": 994060985,
"high_24h": 37.69,
"low_24h": 35.15,
"price_change_24h": -1.459170053196,
"price_change_percentage_24h": -3.88757,

```
  {
     "id": "dogecoin",
   "symbol": "doge",
   "name": "Dogecoin",
   "image": "https://assets.coingecko.com/coins/images/5/large/
   dogecoin.png?1547792256",
     "current_price": 0.210163,
     "market_cap": 27828918818,
     "market_cap_rank": 10,
     "fully_diluted_valuation": null,
     "total_volume": 1921954445,
     "high_24h": 0.220447,
     "low_24h": 0.202133,
     "price_change_24h": -0.003659882814,
     "price_change_percentage_24h": -1.71164,
     "market_cap_change_24h":
   -462357528.36421967,
   "market_cap_change_percentage_24h": -1.63428,
   "circulating_supply": 132358930217.212,
     "total_supply": null,
     "max_supply": null,
     "ath": 0.731578,
     "ath_change_percentage": -71.26029,
     "ath_date": "2021-05-08T05:08:23.458Z",
     "atl": 0.0000869,
     "atl_change_percentage": 241838.24836,
     "atl_date": "2015-05-06T00:00:00.000Z",
     "roi": null,
    },"last_updated": "2021-12-02T18:01:37.164Z"
{
        "id": "avalanche-2",
     "symbol": "AVAX",
     "name": "Avalanche",
     "image": "https://assets.coingecko.com/coins/images/12559/large/coin-round-
red.png?1604021818",
   "current_price": 112.6,
   "market_cap": 25267860253,
   "market_cap_rank": 11,
   "fully_diluted_valuation": 81232482279,
"total_volume": 1064877430,
   "high_24h": 125.99,
   "low_24h": 112.32,
   "price_change_24h": -12.596589433703,
```

{
"id": "terra-luna",
"symbol": "luna",
"name": "Terra",
"image": "https://assets.coingecko.com/coins/images/8284/large/
luna1557227471663.png?1567147072",
"current_price": 63.8,
"market_cap": 24844018523,
"market_cap_rank": 12,
"fully_diluted_valuation": 63918879932,
"total_volume": 2617900266,
"high_24h": 65.74,
"low_24h": 61.27,
"price_change_24h": 1.49,
"price_change_percentage_24h": 2.39714,
"market_cap_change_24h": 459422731,
"market_cap_change_percentage_24h":
1.88407, "circulating_supply":
388680442.300792,
"total_supply": 865063381.623237,
"max_supply": 1000000000,
"ath": 65.74,
"ath_change_percentage": -2.95815,
"ath_date": "2021-12-02T15:50:03.093Z",
"atl": 0.121798,
"atl_change_percentage": 52278.20388,
"atl_date": "2020-03-18T17:03:01.083Z",
"roi": null,
},"last_updated": "2021-12-02T18:00:33.006Z"
{
"id": "shiba-inu",
"symbol": "shib",
"name": "Shiba Inu",
"image": "https://assets.coingecko.com/coins/images/11939/large/shiba.png?
1622619446",
"current_price": 0.00004273,
"market_cap": 23478482836,
"market_cap_rank": 13,
"fully_diluted_valuation": null,
"total_volume": 2444966820,
"high_24h": 0.00004613,
"low_24h": 0.00004135,
"price_change_24h": -0.000003184895,
"price_change_percentage_24h":

{
"id": "crypto-com-chain",
"symbol": "cro",
"name": "Crypto.com Coin",
"image": "https://assets.coingecko.com/coins/images/7310/large/cypto.png?1547043960",
"current_price": 0.693441,
"market_cap": 17550106067,
"market_cap_rank": 14,
"fully_diluted_valuation": null,
"total_volume": 503125161,
"high_24h": 0.732579,
"low_24h": 0.692026,
"price_change_24h": -0.031804796172,
"price_change_percentage_24h": -4.38538,
"market_cap_change_24h": -946618427.3423882,
"market_cap_change_percentage_24h": -5.11776,
"circulating_supply": 25263013692,
"total_supply": 30263013692,
"max_supply": null,
"ath": 0.965407,
"ath_change_percentage": -28.17113,
"ath_date": "2021-11-24T15:53:54.855Z",
"atl": 0.0121196,

{
"id": "wrapped-bitcoin",
"symbol": "wbtc",
"name": "Wrapped Bitcoin",
"image": "https://assets.coingecko.com/coins/images/7598/large/wrapped_bitcoin_wbtc.png?
1548822744",
"current_price": 56518,
"market_cap": 14346690280,
"market_cap_rank": 15,
"fully_diluted_valuation": 14346690280,
"total_volume": 502845618,
"high_24h": 58223,
"low_24h": 55809,
"price_change_24h": -1704.893690486868,
"price_change_percentage_24h": -2.9282,
"market_cap_change_24h":
-482432779.7759609,
"market_cap_change_percentage_24h": -3.25328,

{
"id": "litecoin",
"symbol": "ltc",
"name": "Litecoin",
"image": "https://assets.coingecko.com/coins/images/2/large/litecoin.png?
1547033580", "current_price": 205.69,
"market_cap": 14210980706,
"market_cap_rank": 16,
"fully_diluted_valuation": 17275223130,
"total_volume": 1309628619,
"high_24h": 212.77,
"low_24h": 201.15,
"price_change_24h": -7.07937269801,
"price_change_percentage_24h": -3.32728,
"market_cap_change_24h": -484897438.0908203,
"market_cap_change_percentage_24h": -3.29955,
"circulating_supply": 69100258.2334713,
"total_supply": 84000000,
"max_supply": 84000000,
"ath": 410.26,
"ath_change_percentage": -49.8718,
"ath_date": "2021-05-10T03:13:07.904Z",
"atl": 1.15,
"atl_change_percentage": 17801.13923,
"atl_date": "2015-01-14T00:00:00.000Z",
"roi": null,
"last_updated": "2021-12-02T18:00:37.602Z"
},
{
"id": "matic-network",
"symbol": "matic",
"name": "Polygon",
"image": "https://assets.coingecko.com/coins/images/4713/large/matic-token-icon.png?
1624446912",
"current_price": 2.06,
"market_cap": 14156638485,
"market_cap_rank": 17,
"fully_diluted_valuation": 20597795318,
"total_volume": 3152780301,
"high_24h": 2.19,
"low_24h": 1.94,
"price_change_24h": 0.072353,
"price_change_percentage_24h":
3.63352,

{
"id": "binance-usd",
"symbol": "busd",
"name": "Binance USD",
"image": "https://assets.coingecko.com/coins/images/9576/large/BUSD.png?
1568947766", "current_price": 1,
"market_cap": 13678155439,
"market_cap_rank": 18,
"fully_diluted_valuation": null,
"total_volume": 6187587944,
"high_24h": 1.01,
"low_24h": 0.990219,
"price_change_24h": 0.00299393,
"price_change_percentage_24h": 0.30014,
"market_cap_change_24h": 101408770,
"market_cap_change_percentage_24h": 0.74693,
"circulating_supply": 13656822031.39,
"total_supply": 13656822031.39,
"max_supply": null,
"ath": 1.15,
"ath_change_percentage": -13.31648,
"ath_date": "2020-03-13T02:35:42.953Z",
"atl": 0.901127,
"atl_change_percentage": 11.02813,

},
{
"id": "algorand",
"symbol": "algo",
"name": "Algorand",
"image": "https://assets.coingecko.com/coins/images/4380/large/download.png?
1547039725", "current_price": 1.9,
"market_cap": 11888181063,
"market_cap_rank": 19,
"fully_diluted_valuation": 18905844334,
"total_volume": 598254523,
"high_24h": 2.02,
"low_24h": 1.85,
"price_change_24h": -0.093180969008,
"price_change_percentage_24h": -4.6828,
"market_cap_change_24h": -736654288.621994,
"market_cap_change_percentage_24h": -5.83496,
"circulating_supply": 6288098459.62265,

{
"id": "chainlink",
"symbol": "link",
"name": "Chainlink",
"image": "https://assets.coingecko.com/coins/images/877/large/chainlink-new-logo.png?
1547034700", "current_price": 24.88,
"market_cap": 11621171173,
"market_cap_rank": 20,
"fully_diluted_valuation": 24884225763,
"total_volume": 798046470,
"high_24h": 26.37,
"low_24h": 24.31,
"price_change_24h": -1.471944265519,
"price_change_percentage_24h": -5.58482,
"market_cap_change_24h": -695142619.9246864,
"market_cap_change_percentage_24h": -5.64408,
"circulating_supply": 467009553.9174637,
"total_supply": 1000000000,
"max_supply": 1000000000,
"ath": 52.7,
"ath_change_percentage": -52.77812,
"ath_date": "2021-05-10T00:13:57.214Z",
"atl": 0.148183,
"atl_change_percentage": 16692.94666,

{
"id": "bitcoin-cash",
"symbol": "bch",
"name": "Bitcoin Cash",
"image": "https://assets.coingecko.com/coins/images/780/large/bitcoin-cash-circle.png?1594689492",
"current_price": 565.22,
"market_cap": 10696794897,
"market_cap_rank": 21,
"fully_diluted_valuation": 11874858690,
"total_volume": 2487335808,
"high_24h": 580.51,
"low_24h": 558.62,
"price_change_24h": -15.122894710784,
"price_change_percentage_24h": -2.60583,
"market_cap_change_24h": -291932562.4924469,
"market_cap_change_percentage_24h": -2.65665,
"circulating_supply": 18916662.3966528,
"total_supply": 21000000,

NAVIGATION BAR

```
import React from "react";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import HomeScreen from "../screens/HomeScreen";
import WatchlistScreen from "../screens/WatchlistScreen";
import PortfolioScreen from "../screens/PortfolioScreen";


const Tab = createBottomTabNavigator();

const BottomTabNavigator = () => {
  return (
    <Tab.Navigator
      initialRouteName="Home"
      screenOptions={{
        headerShown: false,
        tabBarActiveTintColor: "white",
        tabBarInactiveTintColor: "grey",
        tabBarStyle: {
          backgroundColor: "#181818",
        },
      }}
    >
      <Tab.Screen
        name="Home"
        component={HomeScreen}
        options={{
          tabBarIcon: ({ focused, color }) => (
            <Entypo name="home" size={focused ? 30 : 25} color={color} />
          ),
        }}
      />
      <Tab.Screen
        name="Portfolio"
        component={PortfolioScreen}
        options={{
          tabBarIcon: ({ focused, color }) => (
            <Foundation name="graph-pie" size={focused ? 35 : 30} color={color} />
          ),
       }}
       export default BottomTabNavigator;
```

SCREENS

1- ADD NEWASSET SCREEN
2- COIN DETAILED SCREEN
3- PORTFOLIO ASSETS ITEM SCREEN
4- PORTFOLIO ASSETS LIST SCREEN

ADD NEW ASSET SCRREN

```
import React, { useState, useEffect } from "react";
import { View, Text, TextInput, Pressable, KeyboardAvoidingView, Platform }
from "react-native";
import SearchableDropdown from "react-native-searchable-dropdown";
import styles from "./styles";
import { useRecoilState } from "recoil";
import { allPortfolioBoughtAssetsInStorage } from "../../atoms/PortfolioAssets";
import { getAllCoins, getDetailedCoinData } from "../../services/requests";
import AsyncStorage from "@react-native-async-storage/async-storage";
import { useNavigation } from "@react-navigation/native";
import uuid from 'react-native-uuid';

const AddNewAssetScreen = () => {
  const [allCoins, setAllCoins] = useState([]);
  const [boughtAssetQuantity, setBoughtAssetQuantity] = useState("");
  const [loading, setLoading] = useState(false);
  const [selectedCoinId, setSelectedCoinId] = useState(null);
  const [selectedCoin, setSelectedCoin] = useState(null);

  const [assetsInStorage, setAssetsInStorage] = useRecoilState(
    allPortfolioBoughtAssetsInStorage
  );

  const navigation = useNavigation();

  const isQuantityEntered = () => boughtAssetQuantity === "";

  const fetchAllCoins = async () => {
    if (loading) {
      return;
    }
```

```
}
  setLoading(true);
  const allCoins = await getAllCoins();
  setAllCoins(allCoins);
  setLoading(false);
};

const fetchCoinInfo = async () => {
  if (loading) {
    return;
  }
  setLoading(true);
  const coinInfo = await getDetailedCoinData(selectedCoinId);
  setSelectedCoin(coinInfo);
  setLoading(false);
};

useEffect(() => {
  fetchAllCoins();
}, []);

useEffect(() => {
  if (selectedCoinId) {
    fetchCoinInfo();
  }
}, [selectedCoinId]);

const onAddNewAsset = async () => {
  const newAsset = {
    id: selectedCoin.id,
    unique_id: selectedCoin.id + uuid.v4(),
    name: selectedCoin.name,
    image: selectedCoin.image.small,
    ticker: selectedCoin.symbol.toUpperCase(),
    quantityBought: parseFloat(boughtAssetQuantity),
    priceBought: selectedCoin.market_data.current_price.usd,
  };
  const newAssets = [...assetsInStorage, newAsset];
  const jsonValue = JSON.stringify(newAssets);
  await AsyncStorage.setItem("@portfolio_coins", jsonValue);
  setAssetsInStorage(newAssets);
  navigation.goBack();
};
```

```
return (
   <KeyboardAvoidingView style={{ flex: 1 }} keyboardVerticalOffset={80}
behavior={Platform.OS === 'ios' ? 'padding' : 'height'}>
     <SearchableDropdown
      items={allCoins}
      onItemSelect={(item) => setSelectedCoinId(item.id)}
      containerStyle={styles.dropdownContainer}
      itemStyle={styles.item}
      itemTextStyle={{ color: "white" }}
      resetValue={false}
      placeholder={selectedCoinId || "Select a coin..."}
      placeholderTextColor="white"
      textInputProps={{
       underlineColorAndroid: "transparent",
       style: {
        padding: 12,
        borderWidth: 1.5,
        borderColor: "#444444",
        borderRadius: 5,
        backgroundColor: "#1e1e1e",
        color: "white",
       },
      }}
     />
     {selectedCoin && (
      <>
       <View style={styles.boughtQuantityContainer}>
        <View style={{ flexDirection: "row" }}>
         <TextInput
          style={{ color: "white", fontSize: 90 }}
          value={boughtAssetQuantity}
          placeholder="0"
          keyboardType="numeric"
          onChangeText={setBoughtAssetQuantity}
         />
         <Text style={styles.ticker}>{selectedCoin.symbol.toUpperCase()}</Text>
        </View>
        <Text style={styles.pricePerCoin}>
         ${selectedCoin.market_data.current_price.usd} per coin
        </Text>
       </View>
       <Pressable
        style={{
```

COIN DETAILED SCREEN

```
import React, { useState, useEffect } from "react";
import {
  View,
  Text,
  Dimensions,
  TextInput,
  ActivityIndicator,
} from "react-native";
import CoinDetailedHeader from "./components/CoinDetailedHeader";
import styles from "./styles";
import { AntDesign } from "@expo/vector-icons";
import { LineChart, CandlestickChart } from "react-native-wagmi-charts";
import { useRoute } from "@react-navigation/native";
import {
  getDetailedCoinData,
  getCoinMarketChart,
  getCandleChartData,
} from "../../services/requests";
import FilterComponent from "./components/FilterComponent";
import { MaterialIcons } from "@expo/vector-icons";

const filterDaysArray = [
  { filterDay: "1", filterText: "24h" },
  { filterDay: "7", filterText: "7d" },
  { filterDay: "30", filterText: "30d" },
  { filterDay: "365", filterText: "1y" },
  { filterDay: "max", filterText: "All" },
];

const CoinDetailedScreen = () => {
  const [coin, setCoin] = useState(null);
  const [coinMarketData, setCoinMarketData] = useState(null);
  const [coinCandleChartData, setCoinCandleChartData] = useState(null);
  const route = useRoute();
  const {
    params: { coinId },
  } = route;
```

```
const [loading, setLoading] = useState(false);
const [coinValue, setCoinValue] = useState("1");
const [usdValue, setUsdValue] = useState("");
const [selectedRange, setSelectedRange] = useState("1");
const [isCandleChartVisible, setIsCandleChartVisible] = useState(false);

const fetchCoinData = async () => {
  setLoading(true);
  const fetchedCoinData = await getDetailedCoinData(coinId);
  setCoin(fetchedCoinData);
  setUsdValue(fetchedCoinData.market_data.current_price.usd.toString());
  setLoading(false);
};

const fetchMarketCoinData = async (selectedRangeValue) => {
  const fetchedCoinMarketData = await getCoinMarketChart(
    coinId,
    selectedRangeValue
  );
  setCoinMarketData(fetchedCoinMarketData);
};

const fetchCandleStickChartData = async (selectedRangeValue) => {
  const fetchedSelectedCandleChartData = await getCandleChartData(
    coinId,
    selectedRangeValue
  );
  setCoinCandleChartData(fetchedSelectedCandleChartData);
};

useEffect(() => {
  fetchCoinData();
  fetchMarketCoinData(1);
  fetchCandleStickChartData();
}, []);


const onSelectedRangeChange = (selectedRangeValue) => {
  setSelectedRange(selectedRangeValue);
  fetchMarketCoinData(selectedRangeValue);
  fetchCandleStickChartData(selectedRangeValue);
};
```

```
if (loading || !coin || !coinMarketData || !coinCandleChartData) {
  return <ActivityIndicator size="large" />;
}

const {
  id,
  image: { small },
  name,
  symbol,
  market_data: {
    market_cap_rank,
    current_price,
    price_change_percentage_24h,
  },
} = coin;

const { prices } = coinMarketData;

const percentageColor =
  price_change_percentage_24h < 0 ? "#ea3943" : "#16c784" || "white";
const chartColor = current_price.usd > prices[0][1] ? "#16c784" : "#ea3943";
const screenWidth = Dimensions.get("window").width;

const formatCurrency = ({ value }) => {
  "worklet";
  if (value === "") {
    if (current_price.usd < 1) {
      return `$${current_price.usd}`;
    }
    return `$${current_price.usd.toFixed(2)}`;
  }
  if (current_price.usd < 1) {
    return `$${parseFloat(value)}`;
  }
  return `$${parseFloat(value).toFixed(2)}`;
};

const changeCoinValue = (value) => {
    setCoinValue(value);
    const floatValue = parseFloat(value.replace(",", ".")) || 0;
    setUsdValue((floatValue * current_price.usd).toString());
  };
```

```
{isCandleChartVisible ? (
      <CandlestickChart.Provider
       data={coinCandleChartData.map(
        ([timestamp, open, high, low, close]) => ({
         timestamp,
         open,
         high,
         low,
         close,
        })
       )}
      >
       <CandlestickChart height={screenWidth / 2} width={screenWidth}>
        <CandlestickChart.Candles />
        <CandlestickChart.Crosshair>
         <CandlestickChart.Tooltip />
        </CandlestickChart.Crosshair>
       </CandlestickChart>
       <View style={styles.candleStickDataContainer}>
        <View>
         <Text style={styles.candleStickTextLabel}>Open</Text>
         <CandlestickChart.PriceText
          style={styles.candleStickText}
          type="open"
         />
        </View>
        <View>
         <Text style={styles.candleStickTextLabel}>High</Text>
         <CandlestickChart.PriceText
          style={styles.candleStickText}
          type="high"
         />
  </View>
           <View>
            <Text style={styles.candleStickTextLabel}>Low</Text>
            <CandlestickChart.PriceText
             style={styles.candleStickText}
             type="low"
            />
           </View>
           <View>
            <Text style={styles.candleStickTextLabel}>Close</Text>
            <CandlestickChart.PriceText
```

```
</View>
          <CandlestickChart.DatetimeText
            style={{ color: "white", fontWeight: "700", margin: 10 }}
          />
        </CandlestickChart.Provider>
      ) : (
        <LineChart height={screenWidth / 2} width={screenWidth}>
          <LineChart.Path color={chartColor} />
          <LineChart.CursorCrosshair color={chartColor} />
        </LineChart>
      )}

      <View style={{ flexDirection: "row" }}>
        <View style={{ flexDirection: "row", flex: 1 }}>
          <Text style={{ color: "white", alignSelf: "center" }}>
            {symbol.toUpperCase()}
          </Text>
          <TextInput
            style={styles.input}
            value={coinValue}
            keyboardType="numeric"
            onChangeText={changeCoinValue}
          />
        </View>

        <View style={{ flexDirection: "row", flex: 1 }}>
          <Text style={{ color: "white", alignSelf: "center" }}>USD</Text>
          <TextInput
            style={styles.input}
            value={usdValue}
            keyboardType="numeric"
            onChangeText={changeUsdValue}
          />
        </View>
      </View>
    </LineChart.Provider>
  </View>
 );
};

export default CoinDetailedScreen;
```

PORTFOLIO ASSETS ITEM

```
import React from "react";
import { View, Text, Image } from "react-native";
import styles from "./styles";
import { AntDesign } from "@expo/vector-icons";

const PortfolioAssetsItem = ({ assetItem }) => {
  const {
    currentPrice,
    image,
    name,
    priceChangePercentage,
    quantityBought,
    ticker,
  } = assetItem;

  const isChangePositive = () => priceChangePercentage >= 0;

  const renderHoldings = () => (quantityBought * currentPrice).toFixed(2)

  return (
    <View style={styles.coinContainer}>
      <Image
        source={{ uri: image }}
        style={{ height: 30, width: 30, marginRight: 10, alignSelf: "center" }}
      />
      <View>
        <Text style={styles.title}>{name}</Text>
        <Text style={styles.ticker}>{ticker}</Text>
      </View>
      <View style={{ marginLeft: "auto", alignItems: 'flex-end' }}>
        <Text style={styles.title}>{currentPrice}</Text>
        <View style={{ flexDirection: "row" }}>
          <AntDesign
            name={isChangePositive() ? "caretup" : "caretdown"}
            size={12}
            color={isChangePositive() ? "#16c784" : "#ea3943"}
            style={{ alignSelf: "center", marginRight: 5 }}
          />
```

```
<Text style={styles.title}>{currentPrice}</Text>
      <View style={{ flexDirection: "row" }}>
       <AntDesign
        name={isChangePositive() ? "caretup" : "caretdown"}
        size={12}
        color={isChangePositive() ? "#16c784" : "#ea3943"}
        style={{ alignSelf: "center", marginRight: 5 }}
       />
       <Text
        style={{
         color: isChangePositive() ? "#16c784" : "#ea3943",
         fontWeight: "600",
        }}
       >
        {priceChangePercentage?.toFixed(2)}
       </Text>
      </View>
     </View>
     <View style={styles.quantityContainer}>
      <Text style={styles.title}>${renderHoldings()}</Text>
      <Text style={styles.ticker}>
       {quantityBought} {ticker}
      </Text>
     </View>
    </View>
  );
};

export default PortfolioAssetsIt
```

PORTFOLIO ASSETS LIST

```
import React from "react";
import { View, Text, Pressable } from "react-native";
import { AntDesign } from "@expo/vector-icons";
import styles from "./styles";
import PortfolioAssetsItem from "../PortfolioAssetItem";
import { useNavigation } from "@react-navigation/native";
import { useRecoilValue, useRecoilState } from "recoil";
import {
  allPortfolioAssets,
  allPortfolioBoughtAssetsInStorage,
```

```
} from "../../../../atoms/PortfolioAssets";
import { SwipeListView } from "react-native-swipe-list-view";
import { FontAwesome } from "@expo/vector-icons";
import AsyncStorage from '@react-native-async-storage/async-storage'

const PortfolioAssetsList = () => {
  const navigation = useNavigation();
  const assets = useRecoilValue(allPortfolioAssets);
  const [storageAssets, setStorageAssets] = useRecoilState(
    allPortfolioBoughtAssetsInStorage
  );

  const getCurrentBalance = () =>
    assets.reduce(
      (total, currentAsset) =>
        total + currentAsset.currentPrice * currentAsset.quantityBought,
      0
    );

  const getCurrentValueChange = () => {
    const currentBalance = getCurrentBalance();
    const boughtBalance = assets.reduce(
      (total, currentAsset) =>
        total + currentAsset.priceBought * currentAsset.quantityBought,
      0
    );

    return (currentBalance - boughtBalance).toFixed(2);
  };
  const getCurrentPercentageChange = () => {
    const currentBalance = getCurrentBalance();
    const boughtBalance = assets.reduce(
      (total, currentAsset) =>
        total + currentAsset.priceBought * currentAsset.quantityBought,
      0
    );
    return (
      (((currentBalance - boughtBalance) / boughtBalance) * 100).toFixed(2) || 0
    );
  };
```

```
const onDeleteAsset = async (asset) => {
    const newAssets = storageAssets.filter((coin) => coin.unique_id !== asset.item.unique_id)
    const jsonValue = JSON.stringify(newAssets);
    await AsyncStorage.setItem("@portfolio_coins", jsonValue)
    setStorageAssets(newAssets);
  };

  const renderDeleteButton = (data) => {
    return (
      <Pressable
        style={{
          flex: 1,
          backgroundColor: "#EA3943",
          alignItems: "flex-end",
          justifyContent: "center",
          paddingRight: 30,
          marginLeft: 20,
        }}
        onPress={() => onDeleteAsset(data)}
      >
        <FontAwesome name="trash-o" size={24} color="white" />
      </Pressable>
    );
  };
   const isChangePositive = () => getCurrentValueChange() >= 0;

    return (
      <SwipeListView
        data={assets}
        renderItem={({ item }) => <PortfolioAssetsItem assetItem={item} />}
        rightOpenValue={-75}
        disableRightSwipe
        closeOnRowPress
        keyExtractor={({id}, index) => `${id}${index}`}
        renderHiddenItem={(data) => renderDeleteButton(data)}
        ListHeaderComponent={
          <>
            <View style={styles.balanceContainer}>
              <View>
                <Text style={styles.currentBalance}>Current Balance</Text>
                <Text style={styles.currentBalanceValue}>
                  ${getCurrentBalance().toFixed(2)}
```

```
</Text>
          <Text
            style={{
              ...styles.valueChange,
              color: isChangePositive() ? "green" : "red",
            }}
          >
            ${getCurrentValueChange()} (All Time)
          </Text>
        </View>
        <View
          style={{
            ...styles.priceChangePercentageContainer,
            backgroundColor: isChangePositive() ? "green" : "red",
          }}
        >
          <AntDesign
            name={isChangePositive() ? "caretup" : "caretdown"}
            size={12}
            color={"white"}
            style={{ alignSelf: "center", marginRight: 5 }}
          />
          <Text style={styles.percentageChange}>
            {getCurrentPercentageChange()}%
          </Text>
        </View>
      </View>
      <Text style={styles.assetsLabel}>Your Assets</Text>
    </>
  }
  ListFooterComponent={
      <Pressable
        style={styles.buttonContainer}
        onPress={() => navigation.navigate("AddNewAssetScreen")}
      >
        <Text style={styles.buttonText}>Add New Asset</Text>
      </Pressable>
    }
  />
 );
};

export default PortfolioAssetsList;
```

WATCH LISTED COIN

```
import React, { useState, useEffect } from 'react';
import { View, Text, FlatList, RefreshControl } from 'react-native';
import { useWatchlist } from '../../Contexts/WatchlistContext';
import CoinItem from '../../components/CoinItem';
import { getWatchlistedCoins } from '../../services/requests';

const WatchlistScreen = () => {
  const {watchlistCoinIds} = useWatchlist();

  const [coins, setCoins] = useState([]);
  const [loading, setLoading] = useState(false);

  const transformCoinIds = () => watchlistCoinIds.join('%2C');

  const fetchWatchlistedCoins = async () => {
    if (loading) {
      return;
    }
    setLoading(true);
    const watchlistedCoinsData = await getWatchlistedCoins(1, transformCoinIds());
    setCoins(watchlistedCoinsData);
    setLoading(false);
  };

  useEffect(() => {
    if (watchlistCoinIds.length > 0) {
      fetchWatchlistedCoins();
    }
  }, [watchlistCoinIds]);
   return (
      <FlatList
        data={coins}
        renderItem={({ item }) => <CoinItem marketCoin={item} />}
        refreshControl={
          <RefreshControl
            refreshing={loading}
            tintColor="white"
            onRefresh={watchlistCoinIds.length > 0 ? fetchWatchlistedCoins :
  null}

  export default WatchlistScreen;
```

Chapter 8

TESTING

1.BLACK BOX TESTING

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance.

2.WHITE BOX TESTING

White-box testing is a method of software testing that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases.

3.GREY BOX TESTING

Greybox testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a combination of black box and white box testing because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

4.UNIT TESTING

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing.

5.INTEGRATION TESTING

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group.Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group.

6.FUNCTIONAL TESTING

Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations.

7.END-TO-END TESTING

End-to-end testing, also known as E2E testing, is a methodology used for ensuring that applications behave as expected and that the flow of data is maintained for all kinds of user tasks and processes.End-to-end testing is a technique that tests the entire software product from beginning to end to ensure the application flow behaves as expected.

8.PERFORMANCE TESTING

Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload.We will do performance testing once the software is stable and moved to the production, and it may be accessed by the multiple users concurrently, due to this reason, some performance issues may occur.

SNAPSHOTS Chapter 9

ANDROID IOS

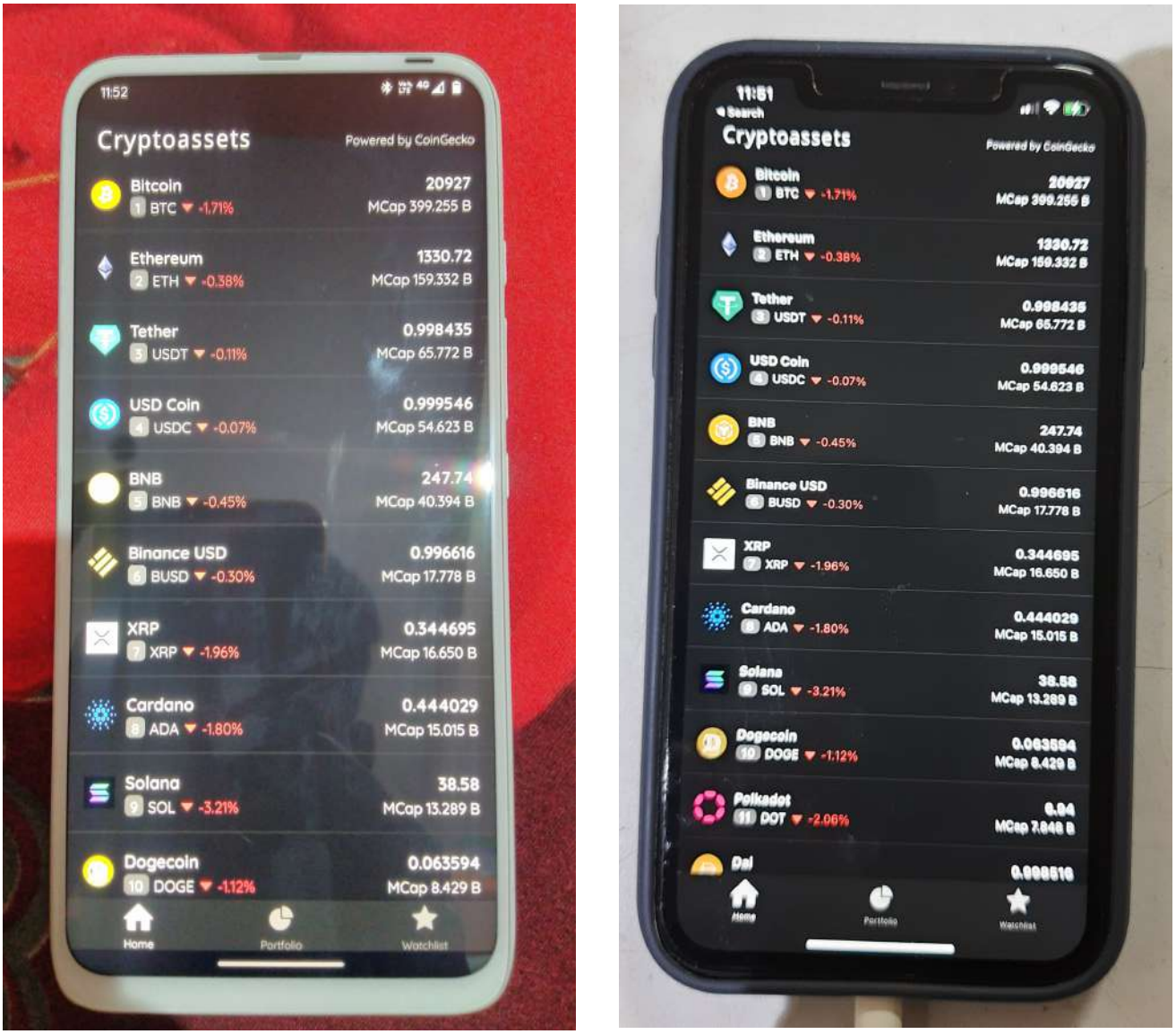
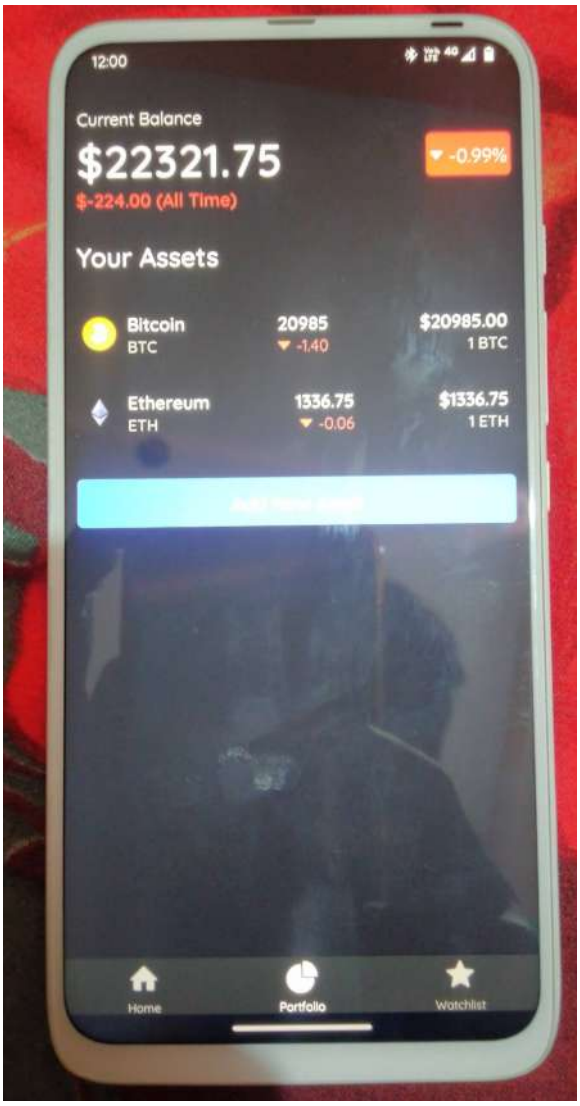


Fig 9.1 : Home Page

ANDROID IOS



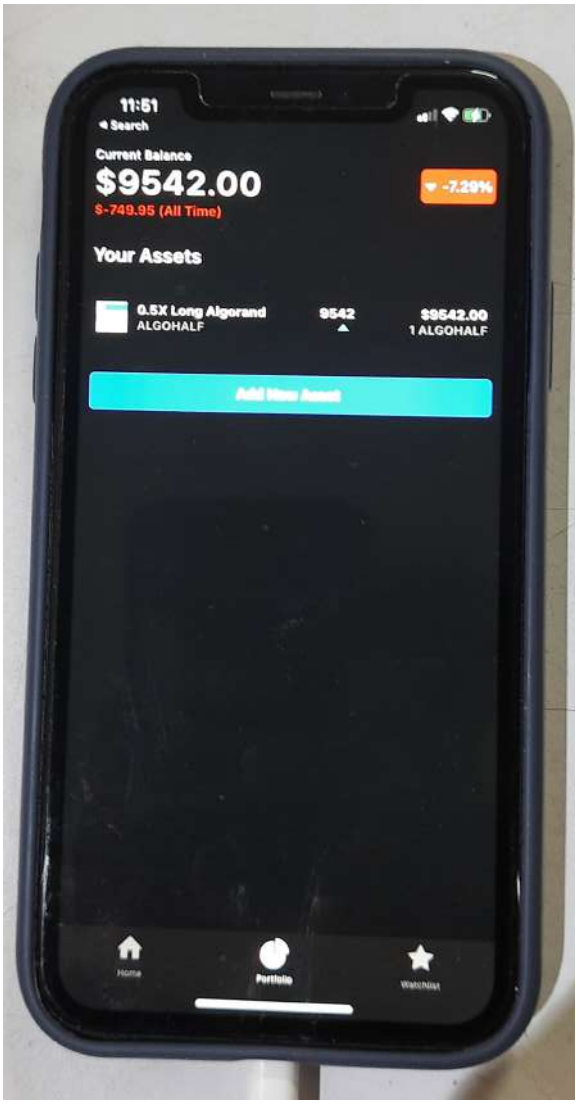


Fig 9.2 : Portfolio Page

ANDROID IOS

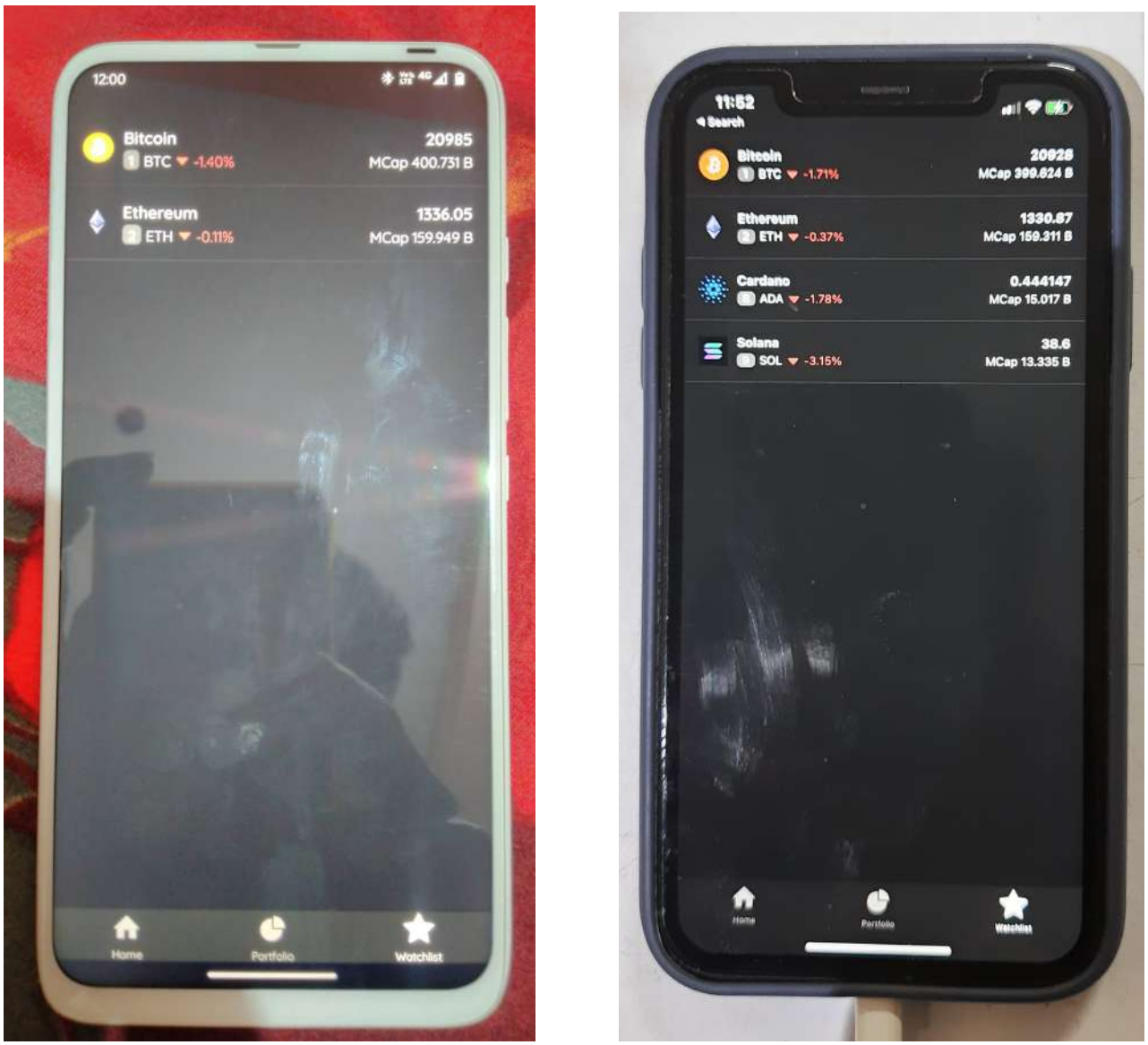


Fig 9.3 : Watchlist Page

ANDROID IOS

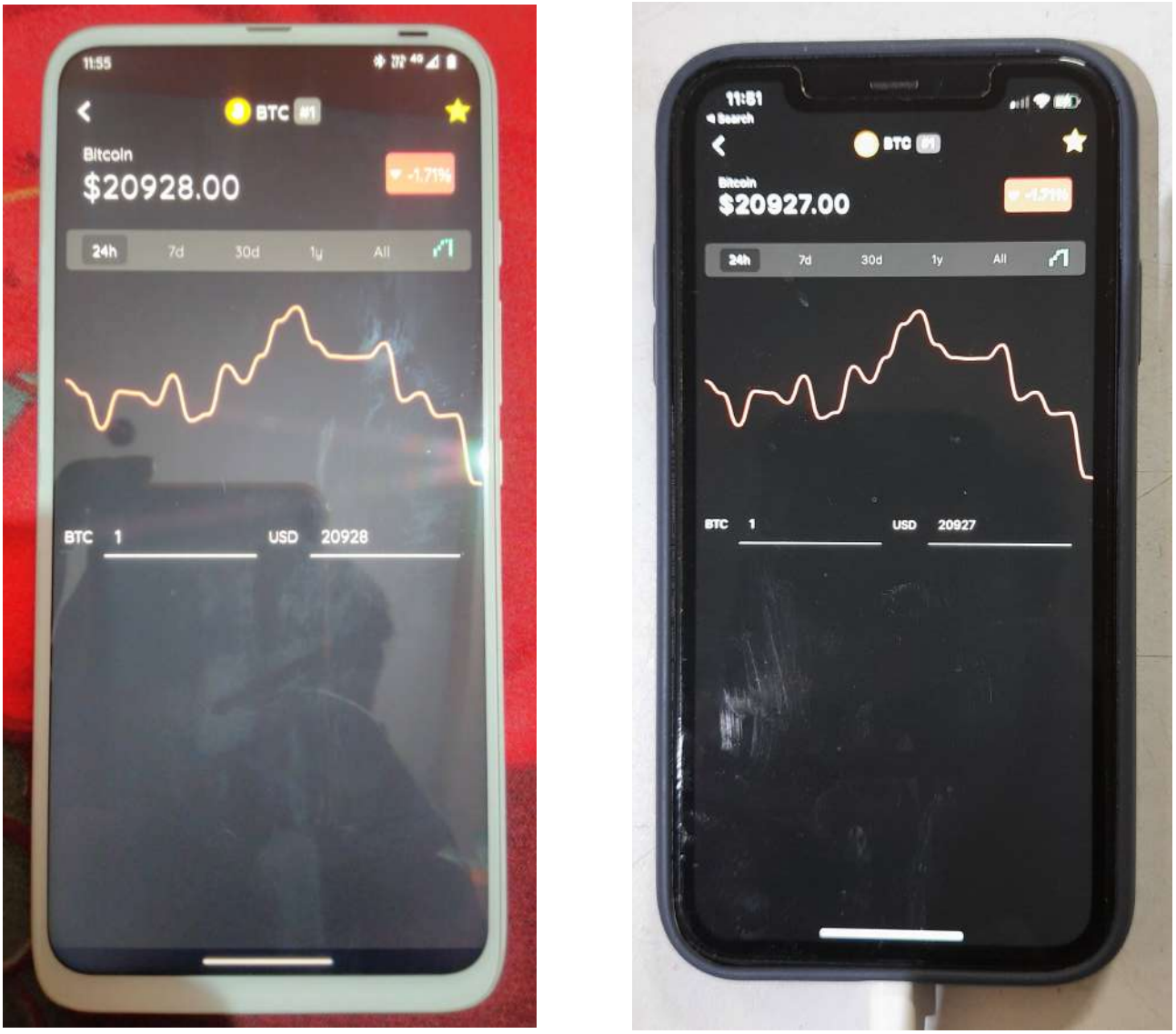


Fig 9.4 : Coin Assets(Description)

ANDROID IOS

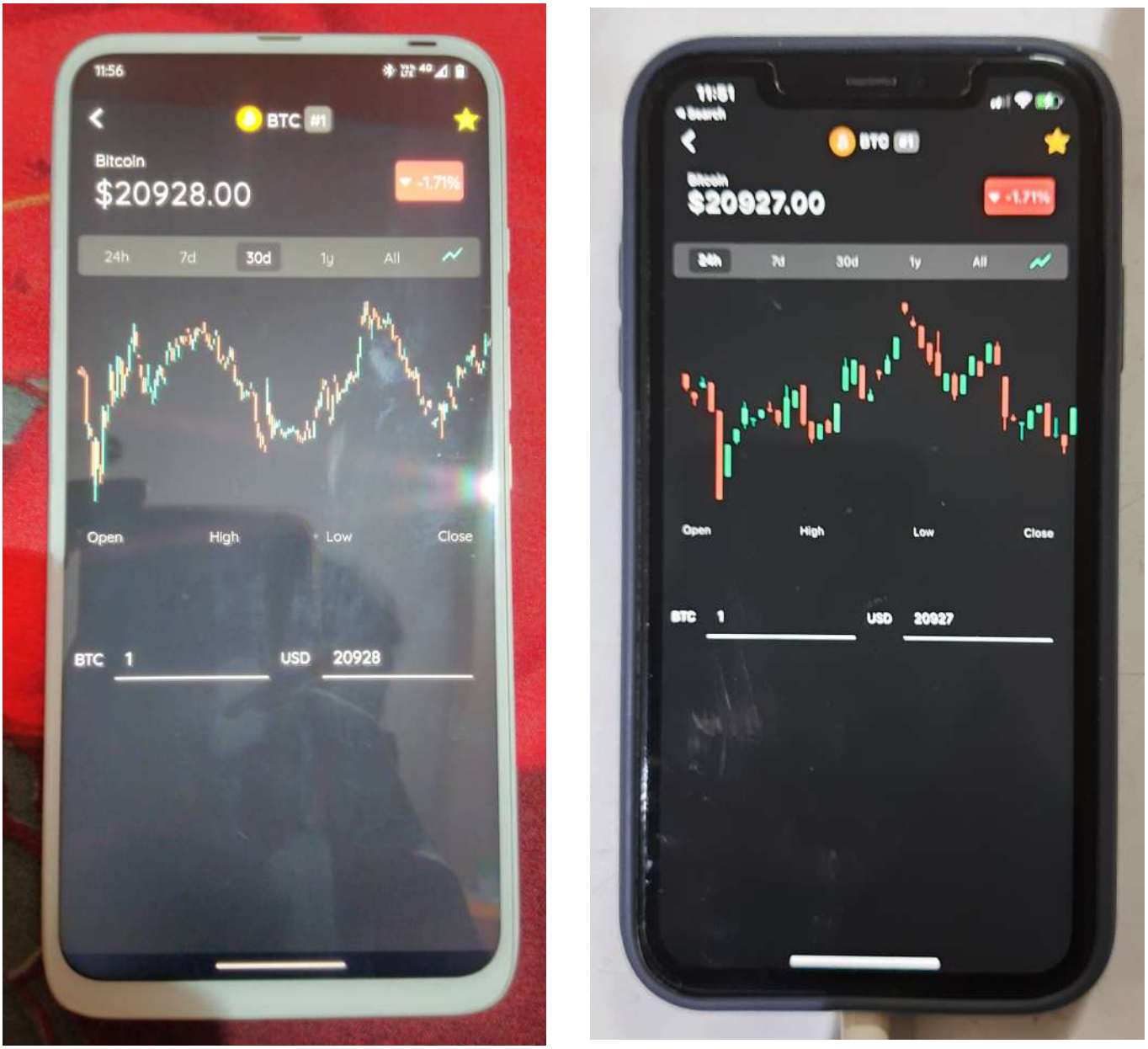


Fig 9.5 : Coin Price In Graph (Candel/Line)

ANDROID

IOS

Fig 9.6 : Asset List Page

ANDROID

IOS

Fig 9.7 : Add (Coin) Asset Page

Chapter 10
RESULTS & DISCUSSION

By using a single programming language (JavaScript) we have created an application which works on Both the Mobile Operating Systems(Android,IOS) by using React Native Framework.The application works fine on both the operating systems with no issues and as the application is written in JavaScript code which is common for both the operating systems, installing the application in both IOS&ANDROID device is done using Expo that Exports this application to APK format for android and for ios Build (IPA) format.

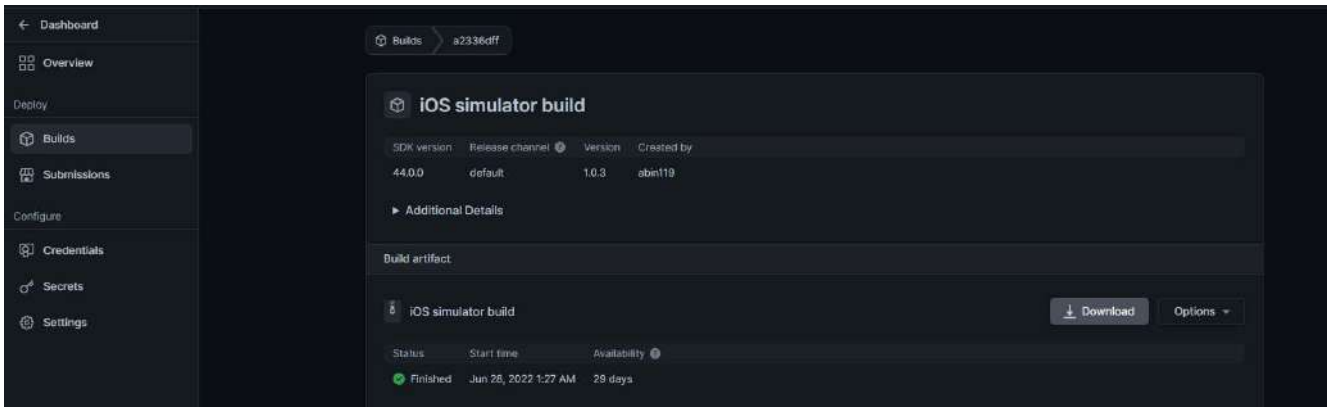


Fig 10.1: Expo Exported To IOS

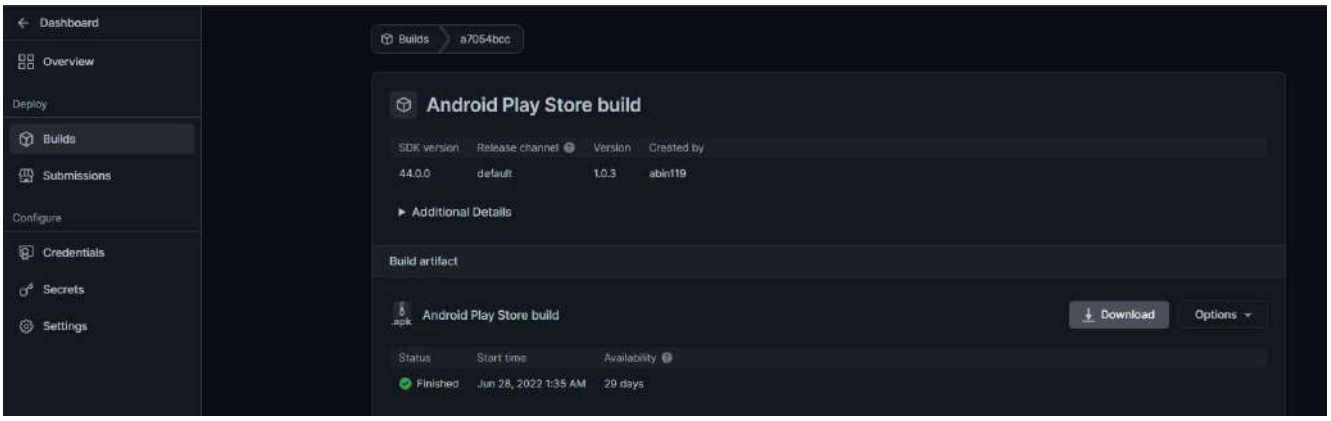


Fig 10.2: Expo Exported To ANDROID

Chapter 11

CONCLUSION

Cross-platform mobile development is not a clear winner of native app against cross- platform app development opposition. At the same time, it has lots of significant advantages that make the approach the only possible solution for certain types of startupers.

Cross-platform development is faster, cheaper, and easier. It is a perfect tool to test the startup idea or present it to investers before assigning money in native app development. In addition, it works perfectly for sustainable corporate apps.

There are also a couple of arguments against multi-platform mobile app development. Cross-platform apps usually have lower operating speed and responsiveness, poor UX and UI, and may not support hardware features.

Another frequent criticism is its inseparability from the framework the app is built with. Once you've created an app using Xamarin or React Native, you've got no chances to use the same code within another framework.

Remember, that apart from cross-platform apps there are hybrid (cheap, but not suitable that much for hardware features integration) and progressive web apps (good for eCommerce).

Chapter 12

FUTURE ENHANCEMENT

The future of cross-platform app development looks promising. However, with the rise in popularity, there is no telling what we can expect to see come into fruition in years ahead!

For the future,
1.We decided to add wallet Id in the blockchain (private / public) key for the user to transfer their crypto assets from one id to another id.

2.In the application we can buy/sell crypto assets.

3.We have decided to use this application for the web3 based authentication.

4.Use of AI&ML for the price predication automatically without any false errors.

Chapter 13

REFERENCES

[1].Cider: Native Execution of iOS Apps on Android, Jeremy Andrus, Alexander Vant Hof, Naser AlDuaij, Christof-fer Dall, Nicolas Viennot, and Jason Nieh, IEEE Micro,2008,Vol.1.

[2].Android Development Tools plugin for Eclipse (2012),http://developer.android.com/sdk/eclipse-adt.html.

[3].About PhoneGap (2011), http://phonegap.com/about.

[4].Heitktter, Henning, Sebastian Hanschke, and Tim A. Ma-jchrzak. ”Evaluating cross- platform development approaches for mobile.

[5].http://developer.apple.com/members.

[6].http://source.google.com.

[7].https://developer.apple.com/xcode.

[8].Guangyuan Piao and Wooju Kim Introduction to iPad Application Development with PhoneGap, International Jour-nal of Innovation, Management and Technology, Vol. 4, No. 1, February 2013 []. jQuery . John Ressig., Jan 2006 ¡http://jquery.com/¿.

[9].jQtouch. David Kaneda., ¡http://jqtouch.com/¿.

[10].Weinre, A debugging tool . IBM Inc., 2011 ¡http://phonegap.github.com/weinre/¿.

[11].Ripple Emulator, Ripple Community Site., http://ripple.tinyhippos.com/.

[12].Anirudh Nagesh, Carlos E. Caicedo Cross-Platform Mobile ApplicationDevelopment, ITERA 2012, The 10th Annual Conference on Telecommunications and Information Technology, 30th March-1 April 2012, Indianapolis Indiana.

[13].D Keuper, (s1019775, XNU a security evaluation), University of Twente andCertified Secure, December 13, 2012.

[14].Aniket Gulhane*, Shailesh More, Shaunak Shete, Chin-may Girolkar, Code Converter for Android and IOS, Interna-tional Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 1, January 2014.

[15].Kim W. Tracy Mobile application development ex-periences on Apples iOS and Android OS. IEEE 2012 [2] Christian G. Acord.