A

Project Report on

# "CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT WITH REACT NATIVE"

Project submitted

In partial fulfilment of the requirements for the degree of

**MASTER OF COMPUTER APPLICATION**

By

**SOUVIK SAHA**

Class Roll No: 002010503033

Examination Roll No: MCA2360018

Registration No: 154241 of 2020-21

Under the supervision of

**DR. SUDIP KUMAR NASKAR**

Department of Computer Science and Engineering

Faculty of Engineering and Technology

Jadavpur University Kolkata – 700 032

India

# Faculty of Engineering and Technology
# Jadavpur University

## To whom it may concern

This is to clarify that the project entitled "**CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT WITH REACT NATIVE**" has been completed by **Souvik Saha (Registration No: 154241 of 2020-2021, Examination Roll No: MCA2360018)**. This work is carried out under the supervision of **Dr. Sudip Kumar Naskar** in partial fulfilment of the requirements for the degree of Master of Computer Application of the department of Computer Science and Engineering, Jadavpur University, during the session 2022-2023. The project report has been approved as it satisfies the academic requirements in respect to project work prescribed for the said degree.

---

**(Signature of Head of the Department)**

**Dr. Nandini Mukherjee**

Head of the Department

Department of Computer Science and Engineering,

Jadavpur University, Kolkata-700032

**(Signature of Project Supervisor)**

**Dr. Sudip Kumar Naskar**

Project Supervisor

Department of Computer Science and Engineering,

Jadavpur University, Kolkata-700032

**(Signature of Dean of the Faculty)**

**Prof. Ardhendu Ghoshal**

Jadavpur University, Kolkata-700032

# Faculty of Engineering and Technology
# Jadavpur University

## Certificate of Approval

This is to clarify that the project entitled "**CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT WITH REACT NATIVE**" has been completed by **Souvik Saha**. This work is carried out under the supervision of **Dr. Sudip Kumar Naskar** in partial fulfilment of the requirements for the degree of Master of Computer Application of the department of Computer Science and Engineering, Jadavpur University, during the session 2022-2023. The project report has been approved as it satisfies the academic requirements in respect to project work prescribed for the said degree.

(Signature of the Internal Examiner)

(Signature of the External Examiner)

# Faculty of Engineering and Technology

# Jadavpur University

## Declaration of Originality and Compliance of Academic Ethic

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledge in accordance with the standard referencing practices.

I declare that this is a true copy of my thesis, including any final revisions, and that this thesis has not been submitted for higher degree to any other University or Institution.

_____

(Signature of the Candidate)
Name – Souvik Saha
Registration No. - 154241 of 2020-2021
Examination Roll No: MCA2360018
Class Roll No. - 002010503033
Thesis Title — `Cross-Platform Mobile Application Development With React-Native`

# Acknowledgement

I would first like to thanks my thesis advisor and also the project supervisor**, Dr. Sudip Kumar Naskar** of the department of **Computer Science and Technology** of **Jadavpur University**. The door to his office was always open whenever I ran into a trouble spot or had questions about my project.

I would also like to thank the experts who were involved in the validation survey for this project, **Dr. Nandini Mukherjee,** the **Head of The Department of Computer Science and Technology** and all the faculty members of the **Department of Computer Science and Technology**.

Finally, I must express my very profound gratitude to my parents and also to my friends cum classmates for providing me with unfailing support and continuous encouragement throughout my years of study and throughout the process of my project work and writing this thesis. This accomplishment would not be possible without them. Thank you.

_____

SOUVIK SAHA

Registration No: 154241 of 2020-2021

Examination Roll No: MCA2360018

Class Roll No: 002010503033

Department of Computer Science & Engineering

Jadavpur University

# Abstract

Developing mobile applications for multiple platforms traditionally required separate source codes for each platform, resulting in increased development and maintenance efforts. In response, the industry has witnessed the emergence of cross-platform development frameworks and tools that enable developers to write a single codebase, which can then be compiled into native applications for different platforms. This approach promises to streamline the development process and improve efficiency.

Cross-platform mobile development technologies, such as React Native, offer a practical approach to streamline the development process by enabling a single codebase for multiple platforms. However, the research concludes that the advantages of cross-platform technologies over native development are not clearly defined.

This thesis focuses on evaluating the viability of cross-platform technologies, specifically those that leverage native functionality to deliver a familiar user experience. React Native, a popular cross-platform framework, is chosen as the subject for a detailed investigation. The aim is to assess both the development experience and the user experience of cross-platform mobile application development.

The findings from this investigation can serve as a valuable resource for developers, organizations, and stakeholders looking to make informed decisions about choosing the most suitable mobile app development approach. By thoroughly examining the strengths and weaknesses of cross-platform technologies, we can gain a comprehensive understanding of their potential and determine the scenarios in which they excel or fall short. Ultimately, this research aims to contribute to the ongoing discourse surrounding cross-platform mobile development and provide guidance for practitioners seeking optimal solutions in the ever-evolving landscape of mobile app development.

# CONTENTS

# Chapter 1- INTRODUCTION

Developing and maintaining mobile applications for multiple platforms at the same time can be time consuming. Since the applications have to be developed with each platform's native development technologies, the software designers have to develop and maintain multiple separate source codes for one application. This has pushed many companies and communities to creating new development frameworks and tools for mobile applications which allow the developers to write and maintain a single codebase. The codebase is then compiled to native applications for each platform, and the applications can be published through the platforms' application marketplaces.

This thesis focuses on cross-platform mobile development technologies, specifically on the ones that utilize each platform's native functionality to deliver a familiar user experience to the end users, and React Native is then picked for a closer look. The purpose of this research is to find out if cross-platform technologies are a viable option for modern mobile application development in terms of both development experience as well as user experience.
The primary finding of the research was that while React Native and other cross platform technologies may be a practical option for mobile application development, it is unclear when cross-platform technologies have a significant advantage over native development. In small applications the difference between the two is not pronounced and in more complex applications cross-platform technologies do not necessarily have many benefits compared to native technologies.

Mobile application market has become one of the largest branches of the software industry since its inception. According to reports from We Are Social and Hootsuite, in 2017, 3.7 billion unique users owned smartphones globally. In 2017 52% of all web traffic came from smartphones and another report by App Annie states that in 2017 the total number of mobile application downloads was over 175 billion. The need for mobile application development continues to grow but, at the same time, the unique use-case for these applications poses several problems to the developers because each smartphone operating system is its own platform, independent from the others in terms of the application ecosystem, the intended user experience and the development technologies used. Since the potential user-base for most mobile applications covers multiple platforms, the developers have to develop separate applications for each platform, following the platforms' design guidelines and using their development technologies.

This requirement for expertise in multiple native development technologies and application architectures has created a need for development tools that simplify the development process of mobile applications. Since the main hindrance is having to develop a complete mobile application separately for each platform, new technologies have emerged that allow developers to share parts of the codebase between platforms. These so-called cross-platform technologies promise to solve the issues of developing native mobile applications, but currently there is no clear consensus on whether they fulfil that promise.
Developing applications for iOS or Android devices normally requires one to learn the language and development tools for each platform. For iOS development one has to learn

the Objective-C programming language and be able to use the Xcode development environment. For Android, one needs to learn Java and use the Android Studio environment. React Native is an open source JavaScript framework for building mobile applications for both iOS and Android devices. It was open-sourced on March 2015 by Facebook, and it's based on the React framework published a few years earlier. (Facebook 2015.)

The purpose of this research is to determine whether React Native, and crossplatform mobile development frameworks in general, are a viable option for platform native development technologies. The mobile platforms included in this research are Google's Android and Apple's iOS. Other platforms do exist, but these two cover the vast majority of end users today. Different approaches to mobile development will be introduced and compared through means of a literature review, and after that a small mobile application will be designed and developed with React Native. The development experience and the final application will be evaluated against similar situations in native development to see if any significant differences arise.

## 1.1- Prior research

Prior research in this area has been done since 2011, but many past studies have focused on technologies that have very little in common with modern cross-platform development. More recent studies exist and are applicable to modern mobile development, but the development frameworks themselves change at a fast rate, so some of the results of past research have become outdated. Most notable studies from the past few years include Niclas Hansson and Tomas Vidhall's study on React Native, and Matias Martinez and Sylvain Lecomte's study on the quality of cross-platform applications.

# Chapter 2- Related Work

## About Social Media 4 Like-Minded LYK

LYK is revolutionizing the social media space with its unique and never seen before privacy filters. LYK is one of the first social networks to give its users complete control over their online privacy – be it posts, comments or connection. Unlike other social networking platforms, LYK offers a safe environment for users to connect with each other, express themselves, and share or comment and bond with like-minded users while maintaining complete privacy.

LYK uses proprietary technology that includes machine learning & artificial intelligence to match users to others who share similar interests and passions. It is the only social network to offer three unique levels of connection, allowing users to easily separate posts, calls, and chats based on the type of conversation and content they want to share at any given moment.

## 2.1- Android development

Android development is the process of creating applications for the Android mobile operating system, which is used on a wide range of devices, from smartphones and tablets to smartwatches and televisions. Android applications can be developed using various programming languages, including Java, Kotlin, and C++, and they can be distributed through the Google Play Store or other app marketplaces.

To develop Android applications, you typically need to have knowledge of programming concepts and an understanding of the Android platform, its components, and its development tools. Some of the key components of Android development include the Android SDK (Software Development Kit), Android Studio (the official integrated development environment), and various libraries and APIs (Application Programming Interfaces) that enable developers to create various features and functionalities in their apps.

To get started with Android development, you may want to consider learning Java or Kotlin, as these are the two most commonly used programming languages for Android development. Additionally, you may want to familiarize yourself with the Android SDK and Android Studio, which provide the necessary tools and resources for developing Android applications. There are many online tutorials, courses, and resources available to help you get started with Android development.

## 2.2 – Why use react native for creating app??

React Native is a popular framework for building mobile applications that allows developers to use a single codebase to create apps for both iOS and Android platforms. There are several reasons why you might want to consider using React Native for app development:

**1) Cross-platform compatibility:** React Native allows you to develop apps for both iOS and Android platforms using a single codebase. This means that you can save time and effort by not having to write separate code for each platform.

**2) Faster development:** React Native provides a fast development process because it allows you to hot reload your code, which means that you can see changes in your app in real-time as you make them.

**3) Reusable components:** React Native provides a library of pre-built components that can be easily reused in different parts of the app, saving development time and effort.

**4) Better performance:** React Native uses native components and APIs, which can result in better performance and faster rendering times.

**5) Large developer community:** React Native has a large and active developer community, which means that there are many resources and tools available to help you develop your app.

Overall, React Native can be a good choice for mobile app development if you want to build a cross-platform app quickly, with good performance, and with reusable components.

React Native applications are developed using JavaScript, more precisely EcmaScript 2015. EcmaScript is a superset of JavaScript, adding numerous features to JavaScript and fixing some of the mistakes originally made in the JavaScript specification. EcmaScript has also replaced pure JavaScript in modern web applications, and all modern browsers support some version of it. In addition to ES2015, React Native implements parts of the next version of EcmaScript, ES2016 or ES7.

# Chapter 3- Basic Terminology

## Concept of component in react native:

React Native is an open source framework for building Android and iOS applications using React and the app platform's native capabilities. With React Native, you use JavaScript to access your platform's APIs as well as to describe the appearance and behavior of your UI using React components: bundles of reusable, nestable code.

## Concept of Core Components:

In React Native, a component is a building block that defines a reusable piece of a user interface. A component can be thought of as a function that takes in some input (known as props) and returns a user interface element.

Components can be either functional or class-based. Functional components are simpler and easier to read, while class-based components have more functionality and can maintain their own state.

React Native provides a number of pre-built components, such as Text, View, Image, and ScrollView, that can be used as building blocks for building more complex user interfaces. Developers can also create their own custom components by combining these pre-built components or by writing their own code.

Using components in React Native can make the process of building user interfaces more efficient and organized. Components can be reused throughout an application, reducing the amount of code that needs to be written and maintained. Additionally, components can be easily tested in isolation, making it easier to catch and fix bugs.

We're not limited to the components and APIs bundled with React Native. React Native has a community of thousands of developers.

## 3.1- Basic Components

Most apps will end up using one of these basic components.

**View:** The most fundamental component for building a UI, View is a container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.

View is designed to be nested inside other views and can have 0 to many children of any type.

**Text:** A React component for displaying text. In React Native, text can be displayed using the <Text> component. The <Text> component is similar to the HTML  element and is used to display a single line or multiple lines of text.

**Image:** A React component for displaying different types of images, including network images, static resources, temporary local images, and images from local disk, such as the camera roll.

The <Image> component can be styled using the style prop, which accepts an object with various style properties. Here's an example

```
<Image
  source={{ uri: 'https://via.placeholder.com/150' }}
  style={{ width: 150, height: 150, borderRadius: 75 }}
/>
```

In the example above, we're setting the border radius to 75 pixels, which creates a circular image. These are just a few of the many styling properties available for the <Image> component.

**TextInput:** In React Native, the <TextInput> component is used to allow the user to input text. The <TextInput> component is similar to the HTML <input> element and is used to capture user input.

Here's an example of how to use the <TextInput> component:

```
const [text, setText] = useState('');
const handleTextChange = (newText) => {
   setText(newText);
};
<TextInput
      style={styles.input}
      onChangeText={handleTextChange}
      value={text}
      placeholder="Type here..."
/>
```

In the example above, we're using the <TextInput> component to allow the user to input text. We're also using the useState hook to store the text entered by the user in a state variable called text. We're also defining a function called handleTextChange that updates the text state variable whenever the user enters text into the <TextInput>.

The <TextInput> component can be styled using the style prop, which accepts an object with various style properties.

We're also using the placeholder prop to set a placeholder text that is displayed when the <TextInput> is empty. The value prop is used to set the value of the <TextInput> to the text state variable, and the onChangeText prop is used to call the handleTextChange function whenever the user enters text into the <TextInput>.

The <TextInput> component has many other props and features, such as auto-capitalization, secure text entry, and multiline input. These features can be customized using various props provided by the <TextInput> component.

**ScrollView**: Component that wraps platform ScrollView while providing integration with touch locking "responder" system.

Keep in mind that ScrollViews must have a bounded height in order to work, since they contain unbounded-height children into a bounded container (via a scroll interaction). In order to bound the height of a ScrollView, either set the height of the view directly (discouraged) or make sure all parent views have bounded height. Forgetting to transfer {flex: 1} down the view stack can lead to errors here, which the element inspector makes quick to debug.

**StyleSheet:** A StyleSheet is an abstraction similar to CSS StyleSheets.

# 3.2- Core Components :-

React Native has many Core Components for everything from controls to activity indicators. These components serve as building blocks for constructing various UI elements and implementing functionality. Core components are typically reusable and can be used in multiple apps. They serve the same purpose as JavaScript functions, but work in isolation and return HTML. They are also often designed to be easily extensible, so that new features can be added as needed. Here are some of the essential core components in React Native.

| REACT NATIVE UI COMPONENT | ANDROID VIEW | IOS VIEW | WEB ANALOG | DESCRIPTION |
|---|---|---|---|---|
| `<View>` | `<ViewGroup>` | `<UIView>` | A non-scrolling `<div>` | A container that supports layout with flexbox, style, some touch handling, and accessibility controls |
| `<Text>` | `<TextView>` | `<UITextView>` | `<p>` | Displays, styles, and nests strings of text and even handles touch events |
| `<Image>` | `<ImageView>` | `<UIImageView>` | `<img>` | Displays different types of images |
| `<ScrollView>` | `<ScrollView>` | `<UIScrollView>` | `<div>` | A generic scrolling container that can contain multiple components and views |
| `<TextInput>` | `<EditText>` | `<UITextField>` | `<input type="text">` | Allows the user to enter text |

---

# 3.3- Use of FlatList in React native:

In React Native, the <FlatList> component is used to render a scrollable list of components. The <FlatList> component is optimized for rendering long lists of data and is more performant than using a simple <ScrollView> component.

In a FlatList, **data** and **renderItem** are the two most important props to understand. The first is an array of data used to generate the list, which is commonly an array of objects, and the second is a function that renders a component for each piece in the data array.

Here's an example of how to use the <FlatList> component:

The renderItem prop is a function that takes an object containing information about the item being rendered and returns a React component to display the item. In our example, the renderItem function takes an object containing an item property and returns a <View> component containing a <Text> component to display the item name.

The keyExtractor prop is a function that takes an item from the data array and returns a unique key for that item. In our example, we're using the id property of each item as the key.

The <FlatList> component can also be styled using the style prop, which accepts an object with various style properties. In the example above, we've set the padding of the <FlatList> component to 20, which creates some space between the edges of the screen and the content.

```
const [items, setItems] = useState([
  { id: '1', name: 'Apples' },
  { id: '2', name: 'Oranges' },
  { id: '3', name: 'Bananas' },
  { id: '4', name: 'Grapes' },
  { id: '5', name: 'Mangoes' },
  { id: '6', name: 'Watermelon' },
]);

const renderItem = ({ item }) => (
  <View style={styles.listItem}>
    <Text>{item.name}</Text>
  </View>
);

<FlatList
    data={items}
    renderItem={renderItem}
    keyExtractor={(item) => item.id}
    style={styles.container}
  />
```

Using the <FlatList> component is more performant than using a simple <ScrollView> component, as it only renders the visible items on the screen, rather than rendering all the items at once. This makes it a better choice for rendering long lists of data.

## Flatlist vs Normal map method in react native:

In React Native, the `FlatList` component and the `map` method can both be used to render a list of components. However, there are some key differences between the two approaches.

**1) Performance:** The `FlatList` component is designed to handle long lists of data more efficiently than the map method. The `FlatList` component only renders the items that are currently visible on the screen, while the `map` method renders all the items at once. This means that the `FlatList` component can handle larger lists without sacrificing performance.

**2) Virtualization:** The `FlatList` component uses virtualization to render only the items that are currently visible on the screen. This means that it only renders the components that are necessary, rather than rendering all the components at once. On the other hand, the `map` method renders all the components at once, even if they are not currently visible on the screen.

**3) Infinite Scrolling:** The `FlatList` component has built-in support for infinite scrolling, which allows you to load more items as the user scrolls to the bottom of the list. This is not

possible with the `map` method, as you would need to handle the scrolling and loading of new items manually.

**4)Item Separator:** The `FlatList` component has a built-in `ItemSeparatorComponent` prop, which allows you to add a separator between each item in the list. This is not possible with the `map` method, as you would need to add the separator manually.

In general, if you need to render a long list of items or you need to implement infinite scrolling, the FlatList component is a better choice than the map method. However, if you have a small list of items and you don't need any of the advanced features of the FlatList component, the map method may be sufficient.

# 3.4- Concept of Hooks:

Hooks are a new addition to React that allow you to use state and other React features without writing a class. In React Native, hooks allow you to write functional components that can hold state and perform side effects.

There are several built-in hooks in React that can be used in React Native:

**useState**: This hook allows you to add state to a functional component. The useState hook returns an array with two values: the current state value and a function to update the state value. You can call the update function to update the state value and trigger a re-render of the component.

**useEffect**: This hook allows you to perform side effects in a functional component. The useEffect hook takes a function as its argument and runs that function after every render. You can use this hook to perform actions like fetching data from an API, updating the document title, or setting up event listeners.

**useContext:** This hook allows you to access a context object in a functional component. Context is a way to share data between components without having to pass the data down through multiple levels of props. You can use the useContext hook to access the data stored in a context object.

**useReducer:** This hook allows you to manage state using a reducer function, similar to how you would manage state in a Redux store. The useReducer hook takes a reducer function and an initial state value as its arguments, and returns an array with two values: the current state value and a dispatch function. You can call the dispatch function to update the state value and trigger a re-render of the component.

**useCallback:** This hook allows you to memoize a function so that it only re-renders when its dependencies change. This can improve the performance of your application by reducing unnecessary re-renders.

**useMemo:** This hook allows you to memoize a value so that it only re-calculates when its dependencies change. This can also improve the performance of your application by reducing unnecessary re-calculations.

There are also several third-party hooks that can be used in React Native, such as useNavigation from the React Navigation library, which allows you to navigate between screens in your app.

Overall, hooks are a powerful feature of React that can help you write more concise and maintainable code in React Native. By using hooks, you can write functional components that are easier to reason about and test, while still having access to powerful React features like state and side effects.

# 3.5- State in react native:

In React Native, state is a way to store and manage data within a component. `State` is an object that represents the current state of a component, and it can be updated over time by calling the` setState` method.

To add state to a component, you can use the `useState` hook, which returns an array with two values: the current state value and a function to update the state value. The first value in the array represents the current `state` of the component, and the second value is a function that you can call to update the `state` value.

Here's an example of using the useState hook to add state to a component:

```
const [count, setCount] = useState(0);


 const incrementCount = () => {
   setCount(count + 1);
 };
```

In this example, the `useState` hook is used to add a count state variable to the component with an initial value of 0. The `incrementCount` function is called when the button is pressed, which updates the count state value using the `setCount` function.

By updating the state value, the component will re-render with the new state value. This allows you to create dynamic user interfaces that respond to user input or changes in data.

It's important to note that state should only be used to manage data that can change over time within a component. If you need to share data between components, you should use props or a global state management solution like Redux or Context.

# 3.6- AsyncStorage in react native :

`AsyncStorage` is a simple, unencrypted, asynchronous, persistent, key-value storage system that is available in React Native. It is used to store small amounts of data on the device, such as user preferences or authentication tokens.

The `AsyncStorage` API is similar to the browser's `localStorage` API, but with a few key differences. `AsyncStorage` is asynchronous, meaning that it returns promises instead of directly returning values. This makes it more efficient and responsive, especially when dealing with large amounts of data.

Async Storage can only store string data, so in order to store object data you need to serialize it first. For data that can be serialized to JSON you can use `JSON.stringify()` when saving the data and `JSON.parse()` when loading the data.

## For importing on your project :

import AsyncStorage from '@react-native-async-storage/async-storage';

## Storing data:

`setItem()` is used both to add new data item (when no data for given key exists), and to modify existing item (when previous data for given key exists).

## Storing string value:

```
const storeData = async (value) => {
try {
await AsyncStorage.setItem('@storage_Key', value)
} catch (e) {
// saving error
}
}
```

## Storing object value:

```
const storeData = async (value) => {
try {
const jsonValue = JSON.stringify(value)
await AsyncStorage.setItem('@storage_Key', jsonValue)
} catch (e) {
// saving error
}
}
```

## Reading data:

getItem returns a promise that either resolves to stored value when data is found for given key, or returns null otherwise.

## Reading string value :

```
const getData = async () => {
try {
const value = await AsyncStorage.getItem('@storage_Key')
if(value !== null) {
// value previously stored
}
} catch(e) {
// error reading value
}
}
```

## Reading object value:

```
const getData = async () => {
try {
const jsonValue = await AsyncStorage.getItem("storage_Key")
return jsonValue != null ? JSON.parse(jsonValue) : null;
} catch(e) {
// error reading value
}
}
```

## Remove Item from Asyncstorage:

Removes item for a key, invokes (optional) callback once completed.

## Example:

```
removeValue = async () =>{
try {
await AsyncStorage.removeItem('@MyApp_key')
} catch(e) {
// remove error
}
console.log('Done')
}
```

# 3.7- What is an API :

API stands for Application Programming Interface. It is a set of protocols, routines, and tools for building software applications. An API provides a way for software applications to interact with each other and exchange data.

In simpler terms, an API is like a messenger that takes a request from one software application, sends it to another application, and then returns the response back to the original application.

APIs are commonly used in web development, where they allow different web applications to communicate with each other. They are also used in mobile app development, desktop software development, and many other areas of software development.

APIs can be public or private, depending on whether they are intended for use by anyone or only by a specific group of developers. Many companies provide public APIs that allow developers to integrate their services into their own applications.

# 3.8- Api in php and mysql DATABASE :

In PHP and MySQL, there are several ways to create and consume APIs. Here are some basic steps to create a simple API in PHP and MySQL:

1. **Design your API endpoints**: First, you need to design your API endpoints. An API endpoint is a URL that represents a specific resource in your application. For example, if you're building a bookstore API, you might have endpoints like "/books" to get a list of books and "/books/:id" to get a specific book by its ID.

2. **Create a PHP script to handle requests:** Once you have your API endpoints defined, you need to create a PHP script to handle incoming requests. This script will be responsible for processing the request and returning the appropriate response.

3. **Connect to the MySQL database**: In order to retrieve data from a MySQL database, you need to establish a connection to the database using PHP's MySQL functions.

4. **Retrieve data from the database**: Once you have a connection to the MySQL database, you can use SQL queries to retrieve data from the database. You

5. can use PHP's MySQL functions to execute the SQL query and retrieve the data.

6. **Return the response**: Once you have retrieved the data from the database, you need to return the response to the client. This can be done using PHP's "echo" statement to output the response in JSON format, which is a common format used for APIs.

Here is an example of a simple PHP script that retrieves data from a MySQL database and returns it as a JSON response:

```php
<?php
// establish a connection to the database
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "mydatabase";

$conn = mysqli_connect($servername, $username, $password, $dbname);

// check for errors
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
// define the API endpoint
if ($_GET['action'] == 'get_users') {
    $sql = "SELECT * FROM users";
    $result = mysqli_query($conn, $sql);

    // check for errors
    if (!$result) {
        die("Error: " . $sql . "<br>" . mysqli_error($conn));
    }
    // format the results as JSON
    $users = array();
    while ($row = mysqli_fetch_assoc($result)) {
        $users[] = $row;
    }
    echo json_encode($users);
}
// close the database connection
mysqli_close($conn);
?>
```

# Call php api from react native :

To call a PHP API from React Native, you can use the fetch() function, which is a built-in JavaScript function for making network requests. Here there is an example of how to use fetch() to call a PHP API:

```
fetch('http://your-api-url.com/api.php', {
method: 'POST', // or 'GET'
headers: {
'Content-Type': 'application/json'
},
body: JSON.stringify({
// Request body data goes here
})
})
.then(response => response.json())
.then(data =>  {
// Handle response data here
})
.catch(error => {
// Handle network errors here
});
```

In this example, we are calling a PHP API located at http://your-api-url.com/api.php, using the POST method and sending a JSON request body. The response from the API is converted to JSON using the response.json() method, and then passed to the second .then() callback function for further processing. Any network errors are caught in the .catch() callback function.

You can replace the URL and request body data with your own values, depending on the specific API you are calling. Note that you may need to adjust the headers and request method depending on the requirements of your API.

Also, be sure to handle any authentication and security concerns when making API requests, such as using HTTPS and implementing secure authentication mechanisms.

# 3.9- Api call using axios from react native :

You can also use the `Axios` library to call a PHP API from React Native. `Axios` is a popular JavaScript library for making HTTP requests, and it can be used in both browser and React Native environments.

To use `Axios` in React Native, first install the library using npm or yarn:

npm install axios

Then, you can import Axios and use it to make API requests. Here's an example of how to use `Axios` to call a PHP API:

```
import axios from 'axios';

axios.post('http://your-api-url.com/api.php', {

// Request body data goes here

})

.then(response =>{

// Handle response data here

})

.catch(error => {

// Handle network errors here

});
```

In this example, we are calling a PHP API located at http://your-api-url.com/api.php, using the POST method and sending a request body. The response from the API is handled in the .then() callback function, and any network errors are caught in the .catch() callback function.

You can replace the URL and request body data with your own values, depending on the specific API you are calling. Note that you may need to adjust the headers and request method depending on the requirements of your API.

Also, be sure to handle any authentication and security concerns when making API requests, such as using HTTPS and implementing secure authentication mechanisms.

# Chapter -4

# Various pages and its Features

## 4.1- LYK app SignUp Page

To make the SignUp page we use Text, View, Image, TextInput,TouchableOpacity and other components to make the UI part of this page. One user can register to our LYK app by using their phone number in this app. If the phone number is not previously registered then the user can successfully registerd to the app and a modal will be appeared in the screen and a message will be shown in the modal that "This number is registered with lyk". After that one user should complete their profile by providing their full name, password and re-password and press the next button after that the user can choose their likes, profile picture, add birth day and more their personal information to this app for his/her profile.

One user can continue their Signup by pressing the 'continue with Google' option . In this case we have used a third party package `@react-native-google-signin/google-signin` that provides a Google Sign-In SDK for React Native apps. With this package, we can allow our users to sign in with their Google accounts and access Google APIs like Google Drive, Google Maps, and more. It provides event listeners and callbacks to handle various events related to Google Sign-In.

To use `@react-native-google-signin/google-signin`, we will need to first install it using a package manager like npm or yarn. After installing, we will need to configure the package by following the instructions provided in the package's documentation. This will include creating a project in the Google Developer Console, setting up credentials, and configuring your app to use the Google Sign-In SDK. The package provides a set of APIs that allow you to configure the Google Sign-In functionality.


Once we have configured `@react-native-google-signin/google-signin`, we can use it in your app by importing the package and calling its methods. For example, we can use the signIn method to initiate the Google Sign-In flow, and the getCurrentUser method to retrieve the user's information after they have signed in.

The code snippet is shown here.

```
const _signIn = async () => {
  try {
    await GoogleSignin.hasPlayServices();
    const { user } = await GoogleSignin.signIn();
    setLoggedIn(true);
    setuserInfo(user);
    axios
      .post(SOCIAL_LOGIN_URL, {
```

```
        email: user.email,
        identity: user.id,
        socialMedia: 'googleplus',
       })
       .then(
        res => {
         AsyncStorage.setItem(
         'userId',
           JSON.stringify(res.data.response.userDetails),
         );
         AsyncStorage.setItem('token', res.data.response.token);
         navigation.push('Sidenav');
        },
        err => {
         let errors = {};
         errors.message = 'Invalid username or password!';
        },
       )
       .catch(err => { });
    } catch (error) {
     if (error.code === statusCodes.SIGN_IN_CANCELLED) {
      // user cancelled the login flow
     } else if (error.code === statusCodes.IN_PROGRESS) {
      // operation (f.e. sign in) is in progress already
     } else if (error.code === statusCodes.PLAY_SERVICES_NOT_AVAILABLE) {
      // play services not available or outdated
     } else {
      // some other error happened
     }
    }
  };
```

This code defines an asynchronous function _signIn that uses Google Sign-In to authenticate a user and then sends their information to a server using the Axios library.
i) The function begins by checking if the device has Google Play services installed. If not, an error will be thrown to the users with the code PLAY_SERVICES_NOT_AVAILABLE.

ii) If Google Play services are available, the user will be prompted to sign in using their Google account. If successful, the user's information will be stored in the user variable.

iii) The setLoggedIn and setUserInfo functions are called to update the state of the app to reflect that the user is now logged in to the app and he or she can access the app now.

iv) The user's email, Google ID, and social media platform (Google Plus) are sent to a server using an HTTP POST request. The response from the server is handled with a .then() method, which stores the user's details and token in local storage using the AsyncStorage API.

v) If the HTTP POST request fails, the error message "Invalid username or password!" will be stored in an errors object so there is an error either in username or password .

vi) If any errors occur during the sign-in process, the appropriate error code will be thrown and handled in the catch block.Either the try or the catch block will run at a time.

If the user cancels the sign-in flow, a message will be logged to the console.

Similarly for the Apple Sign in authentication we have used a third party package `@invertase/react-native-apple-authentication` which is a React Native package that provides a way to authenticate users with Apple ID using Apple's authentication services. This package generally can be used to add Apple Sign In functionality to your React Native app. Apple Sign In is a secure and privacy-focused way for users to sign in to apps using their Apple ID. It provides a way for users to sign in without having to create a new account, and it also helps to protect user privacy by allowing them to control what information is shared with the app.
To use `@invertase/react-native-apple-authentication`, you will need to first install it using a package manager like npm or yarn. After installing, you can import the package and use its methods to initiate the Apple Sign In flow and retrieve the user's information. The package provides several methods for initiating the Apple Sign In flow, including performRequest, performRequestWithState, and performRequestWithNonce. These methods allow you to customize the sign-in flow and include additional information like state and nonce values.
Once the user has signed in, you can use the getCredentialStateForUser method to check the credential state of the user's Apple ID. This method can be used to determine if the user has revoked their Apple ID or if their account has been deactivated. Overall, `@invertase/react-native-apple-authentication` is a powerful package that provides a convenient way to add Apple Sign In functionality to your React Native app. However, it's important to note that using this package requires a good understanding of both React Native development and the Apple authentication services. If you're new to either of these topics, it's recommended that you start with some basic tutorials and documentation before diving into this package.

Using the the phone number for Sign up in the app we have used the api 'user/newUserRegister_V2' and we store the api in SIGN_UP Const. The code snippet will show the details.

```
PhpApiRequest.callSimple(SIGN_UP, encParams,
        async (resp) => {
          try {
            console.log('RESPONSE -----', resp);

            let type = null;
            let consumerCode = null;
            let consumerToken = null;
            let token = null;

            let respCode = null;
            let userId = null;

            setLoading(false);
            if (resp != null) {
              if (resp.hasOwnProperty('type')) {
                type = resp.type;
              }
              if (resp.hasOwnProperty('userId')) {
```

```
          userId = resp.userId;
          setUserId(userId)
        }
        if (resp.hasOwnProperty('respCode')) {
          respCode = resp.respCode;
        }
        if (resp.hasOwnProperty("consumerCode") && resp.hasOwnProperty("consumerToken")) {
          consumerCode = resp.consumerCode;
          consumerToken = resp.consumerToken;
          if (resp.hasOwnProperty('token')) {
          token = resp.token;
          }
        }
      async (error) => {
        try {
        setLoading(false);
          Toast.show(Message.OTHER_ERROR);
        } catch (e) {
          console.log(e);
        }
      }
    )
```

This is a block of code written in JavaScript, using the React Native framework. It appears to be making an HTTP request to a server-side API, passing in encrypted parameters for user sign-up.

The PhpApiRequest.callSimple function takes three arguments that is shown in the below:

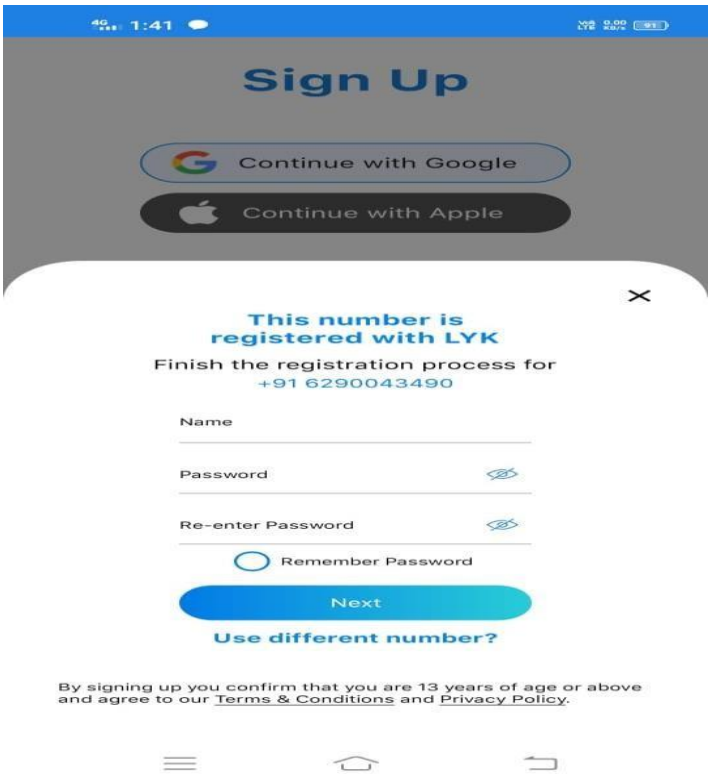
SIGN_UP, which is a constant that likely represents the API endpoint for user sign-up. encParams, which is an object that likely contains the encrypted parameters for the sign-up request.
Two callback functions: async (resp) => {...} and async (error) => {...}. The first callback function is executed when the API call is successful and returns a response. It first logs the response to the console, and then proceeds to extract various pieces of information from the response, such as the user ID, response code, and consumer token.

Depending on the response code, the function will execute one of several cases, each of which appears to navigate the user to a different screen or display a different popup message.

For example, if the response code indicates that the phone number does not exist or has not been verified, the user is pushed to navigate to the screen called SignUpSecondScreen.

The second callback function is executed if the API call fails and returns an error. It sets the loading state to false and displays a Toast message indicating that there was an error.

Overall, this code appears to be part of a larger sign-up flow for a React Native app, where the app sends encrypted sign-up data to a server-side API, and navigates the user to different screens depending on the API response.




After successfully registered the phone number in the LYK app registration page when a person press the `next` button then the user navigates to the another screen `SignUpPasswordModal`. In this page ther user should fills his/her name, fill the Password field and Re-enter Password. If the name field is not empty and the format of the name is correct, password and Re-enter password are matched and both the field have atleast 4 characters, then an api, is written in the below `https://api.lykapp.com/lykjwt/index.php?/user/storeNamePassword` is called from the code to store the user information in the database and if it is successful then the corresponding response will return and the user can navigate to the `Home` screen.

The following code snippets shows the main logic of the page.

```
PhpApiRequest.callSimple(STORE_NAME_PASSWORD, encParams,
 async (resp) => {
  try {
   console.log('RESPONSE -----', resp);
   setLoading(false);
   if (resp != null) {
    if (resp.hasOwnProperty('userDetails')) {
     let userDetails = resp.userDetails;
     await AsyncStorage.setItem(Params.USER_INFO, JSON.stringify(userDetails));
     if (resp.hasOwnProperty('token')) {
      let token = resp.token;
      await AsyncStorage.setItem(Params.TOKEN, token);
     }
     if (resp.hasOwnProperty('toke')) {
      let token = resp.toke;
      await AsyncStorage.setItem(Params.TOKEN, token);
     }
     handleChange(Params.CLOSE_MODAL_CLICK);
     // open screen
     navigation.push('MainInterestScreen');

    }

   } else {
    Toast.show(Message.OTHER_ERROR);
   }
```

```
  } catch (e) {
    console.log(e);
  }
},
```

This function is triggered when the user clicks the submit button and it performs the following steps:

1) It first checks whether the form is valid by calling the `isFormValid()` function. It checks the form is valid or not according the logic this function is not shown in the provided code snippet.

2) If the form is valid, it constructs an object named `params` that contains various properties such as `contactNo`, `did`, `dt`, `refferedBy`, `dc`, `name`, `password`, `userId`, `ct`, `deviceType`, and `deviceId`. Some of these properties are set to values obtained from a **`data`** object, while others are set to constants or the result of calling utility functions.

3) It then logs the params object to the console for checking and debugging purposes.

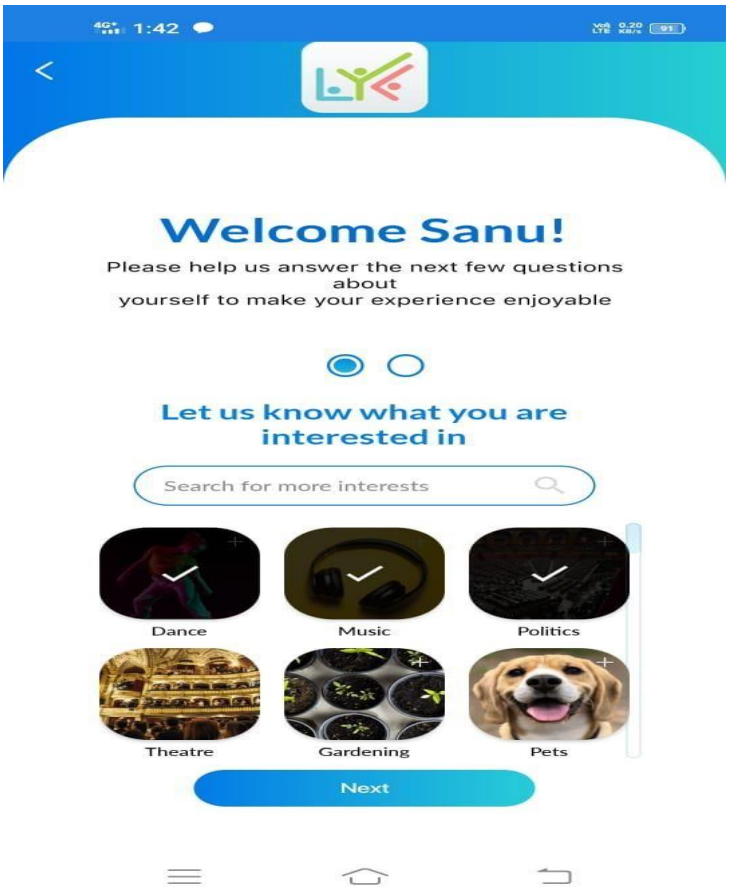
4) It applies an XOR operation with a key to the stringified `params` object, converts the resulting array of ASCII codes to a Buffer, encodes the Buffer as a base64 string, and constructs an object named `encParams` with a single property named `url` set to the base64-encoded string.

5) It sets the loading state to `true` to indicate that the form submission is in progress.

6) It makes a call to a PHP API using the **`PhpApiRequest.callSimple()`** function, passing in the name of a store and the `encParams` object as parameters. The `PhpApiRequest` object and the **`STORE_NAME_PASSWORD`** constant are not shown in the provided code snippet.

7) The **`PhpApiRequest.callSimple()`** function takes two callback functions as parameters. The first function is called if the API call succeeds, and it logs the response to the console, sets the loading state to false, and checks whether the response contains a `userDetails` property. If it does, it saves the `userDetails` object and a `token` property (if present) to the device's `AsyncStorage` storage, closes a modal, and navigates to a "HomeScreen". If the response does not contain a `userDetails` property, it shows a toast message with an "OTHER_ERROR" message.

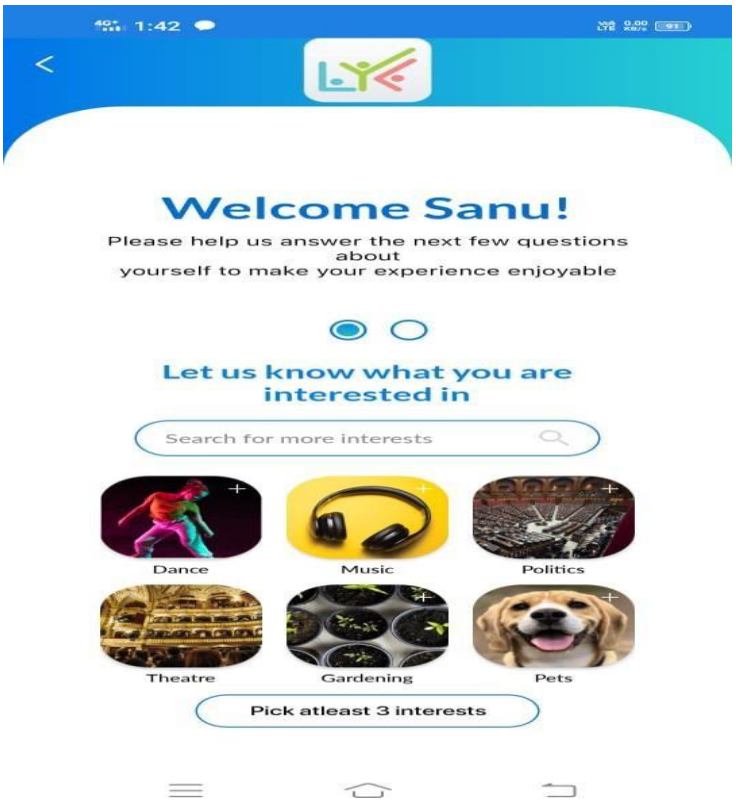
8) The second function is called if the API call fails, and it sets the loading state to `false` and shows a toast message on the screen of users with an "OTHER_ERROR" message.




9) If any error occurs during the execution of the **`submitBtnClickListener`** function, it logs the error to the console and sets the loading state to `false`.

After successfully fulfilled the name, password and Re-password field, the next navigated screen is `MainInterestScreen`.
Since the user profile is already registered in the database and the user is associated with an userId so whenever the user picks atleast of 3 interests from this page and press the next button then `saveInterest` function is called. This function according to first checks that -
If there are one or more selected sub-interest categories, the function creates a new object for each selected sub-interest category, with properties for the main interest category (type), sub-interest category (subType. These objects are then pushed onto the interestArr array.

If there are no selected sub-interest categories, the function creates a new object for the main interest category and the first sub-interest category in the subInterests array, and pushes it onto the interestArr array.
If the interests array is empty, the function creates an object for the first main interest category and its first sub-interest category, and pushes it onto the interestArr array.
After that the Api `https://api.lykapp.com/lykjwt/index.php?/LYKUser/setInterest` is called with set parameters `userId`,`interestArr`,`country`, `userName`,`countryCode`, `countryIso`.




The API call is responsible for sending user interests to the server and navigating to the next screen if the request is successful. The `setLoading` function is used to

manage loading state during the API request, and the `Toast.show` function is used to display error messages if the request fails.

After successfully registerd the account in the app the user can login using their ph number and password .

# 4.2- LYK app Login Page

In this login page first of all we use Text, View, StyleSheet, Image, TextInput, TouchableOpacity and other components to make the UI of this page. After successfully Sign up to this app one user can succesfully Login to the app using his/her mobile number and password which one can use at their Signup time. Also one user can Login to this app using his/her google account or Apple acount (for IOS users) . After successfully login one user can successfully navigate to the Home page.


We have used `@react-native-google-signin/google-signin` which is a third-party library for integrating Google Sign-In into your React Native application. It provides an easy way to authenticate users with their Google accounts and access their basic profile information. Once the user has signed in, the `userInfo` object will contain their basic profile information, such as their name and email address.

You can then use this information to sign the user into your app, or store it for future use.

Also for the IOS users we have used `@invertase/react-native-apple-authentication` which is a third-party library for integrating Apple Sign-In into our React Native application. It provides an easy way to authenticate users with their Apple IDs and access their basic profile information.

Once the user has signed in, the `identityToken` and `nonce` can be used to authenticate the user, while the `user`, `email`, `fullName`, and `realUserStatus` can be used to sign the user in to your app.

If we use the phone number and password for the login then we have to put the correct phone number and password which we have already used for the signup.

To take the phone number input in our app we have used `react-native-phone-number-input` which is a third-party library for adding a phone number input field to our React Native

application. It provides a customizable input field that validates the entered phone number and can format it according to different international standards.

The following snippets will demonstrate the basic functionality of this component.

```
<PhoneNumberInput
  defaultCode="US"
  onChangeText={(text) => {
    // Handle changes to the entered phone number here
  }}
  onChangeFormattedText={(text) => {
    // Handle changes to the formatted phone number here
  }
/>
```

The `defaultCode` prop specifies the default country code for the input field. We can change it to any supported country code or omit it to use the user's current location.

The `onChangeText` prop is called whenever the user enters or deletes a character in the input field. It receives the raw text entered by the user, which you can use to validate the phone number or perform any other necessary processing.

The `onChangeFormatted` Text prop is called whenever the formatted phone number changes. It receives the formatted text, which you can use to display the phone number to the user or perform any other necessary processing.

After putting the phone number and password correctly, when we press the next button to login in this app we called a function named handleSubmit().

In this function we have called an api which is shown in the below `https://api.lykapp.com/lykjwt/index.php?/LYKUser/SignIn_2_0` with the parameters {"countryCode":"+91","countryISO":"IN","currentLat":null,"currentLng":null,"device":"vivo 1951","deviceId":"sdm710","deviceType":"android","identity":"9874797720","password":"12 34","type":"mobile"} in JSON format.

The function named submitBtnClickListener that appears to be handling a form submission in a mobile application.

The function starts with a try-catch block to catch any errors that might occur during its execution.

Inside the try block, `isFormValid()` function is called to check if the form data is valid? If the data is valid, the `DeviceInfo.getDeviceName()` method is called to get the name of the device being used, and several parameters are assembled into a params object. These parameters include identity (likely a phone number or email), countryCode, countryISO, password, type, device, deviceType, deviceId, currentLat, and currentLng.

After the params object is assembled, the `setLoading(true)` method is called to indicate that the application is processing the form data.

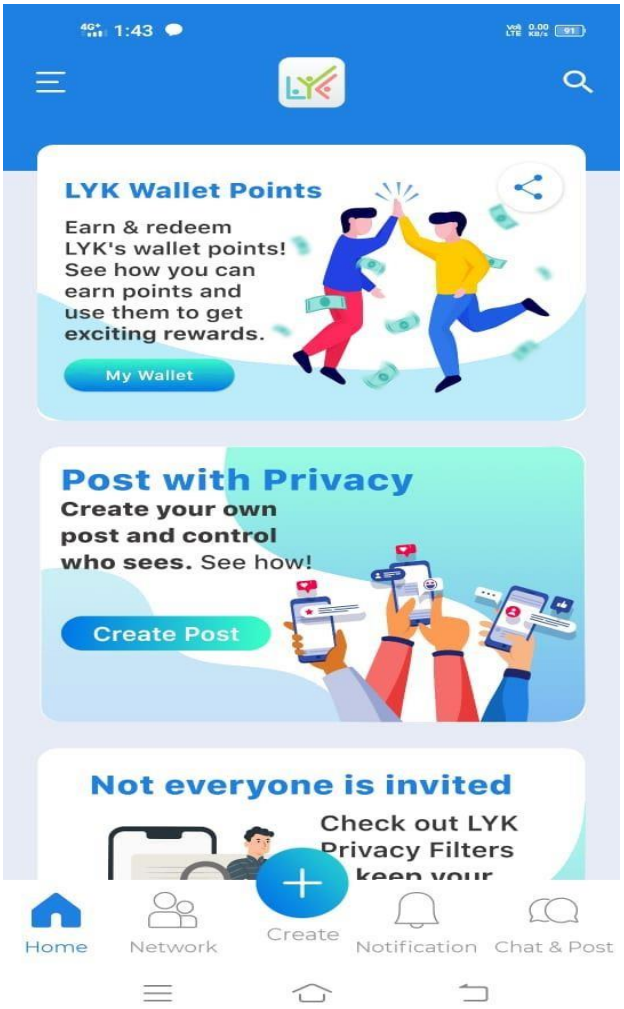
Then, a call is made to a `PhpApiRequest.callSimple()` method, which appears to be an API request to a PHP server. This method takes several parameters, including the` LOGIN_MOBILE API endpoint` and the `params` object that was previously assembled.

If the API request is successful, a callback function is executed, which takes a resp object as its parameter. Inside this callback function, the setLoading(false) method is called to indicate that the form processing has ended.

The resp object is then checked for several properties, including userDetails, typeNotFound, userFound, and respCode. Depending on the values of these properties, the function will take different actions. For example, if userFound is equal to Params.RIGHT_CREDENTIAL, the token property of the resp object is extracted and saved to AsyncStorage. The function will then check whether verifiedUser property of the userDetails object is equal to Params.VERIFIED_USER.

If it is, the userDetails object is saved to AsyncStorage and the user is directed to the main application flow. If it is not, the function calls moveToVerifyScreen() function, which appears to redirect the user to a verification screen.

If the API request fails, an error callback function is executed, which takes an error object as its parameter. Inside this function, the setLoading(false) method is called, and a toast message is displayed to the user indicating that an error has occurred.

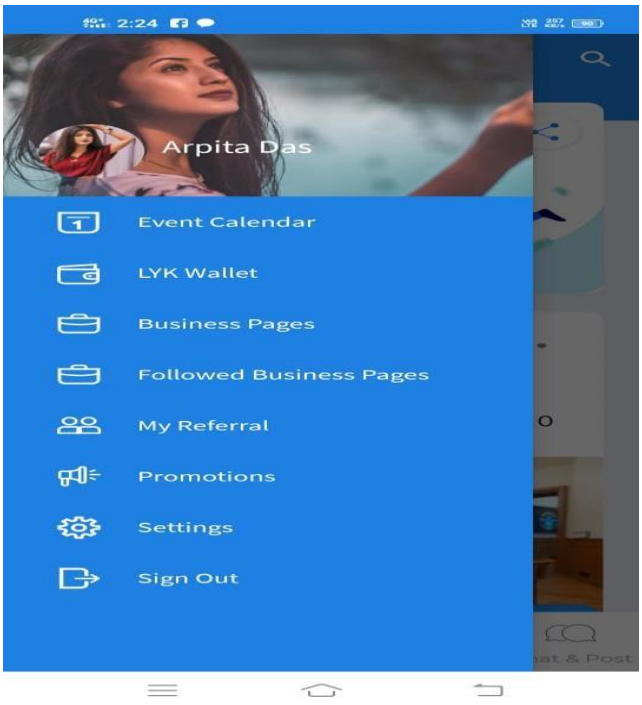



Finally, the entire function is wrapped in another try-catch block to catch any additional errors that might occur during its execution.

If the phone number and current password is matched then the response of the api is OK and the response of the api is also in the JSON format and the response of the api will contain all the information about the user who is successfully loged in and the information of the user will store in the asyncstorage.

If the phone number or the password is not matched for the login time then the response of the api is not succcessful then we will show an error to the user that the phone number or the password is not matched according to the fault of the user.

After successfully loged in one user can navigate to the Home screen.




In the Home screen, the user can navigate on different tab i.e, `Network`, `Create`, `Notification`, `Chat & Post`, also there is an Side bar Navigation window from where the user can use the LYK Wallet, LYK Business Pages, My Referral pages, Settings,

And Signout option to Signout from the page.

I have worked on My Referral screen, Setting Screen, Change Password Screen, Manage Account screen, Customize referral link Screen, Copy link popup Modal Screen,Edit Profile Screen.

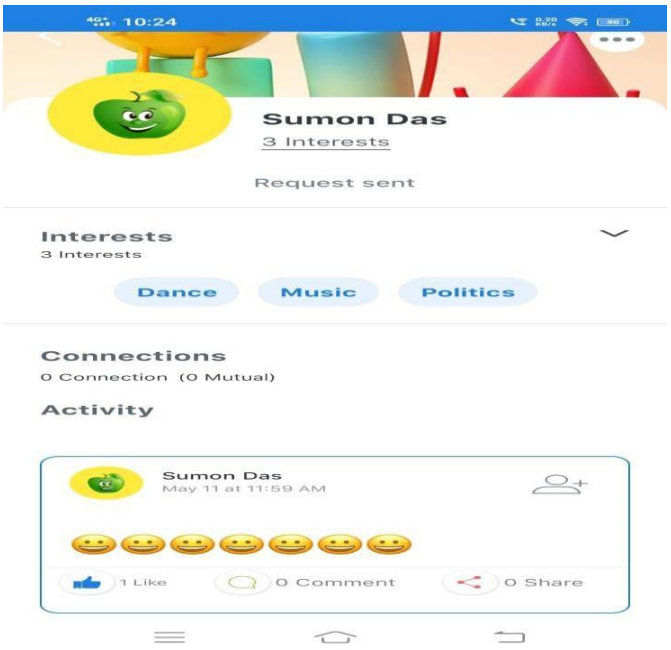
# 4.3- LYK App MY Referral Screen

After sucessfully Signed-up and Login to the LYK app, one user's information will be stored in the Asyncstorage so we can get the userId and another personal inforamation of that user from this page the user will be able to see by whom the user has reffered and also see that What posts have the people done the user's referred made?


Also the user can press `Customize Your Link ` button to navigate to the `Customize your Referral Link` page from where the user can edit his/her referral link. When any user first navigate to the My Referral page then an api which is shown in the below with

specified parameter https://api.lykapp.com/lykjwt/index.php?/Analytical/myRefferedList

Is called with the parameters of {`userId`,`offSet`,`limit`} . If the response of the api call is not null then the PHP API endpoint to retrieve the user's referral list, i.e, the list of the persons the user has reffered.

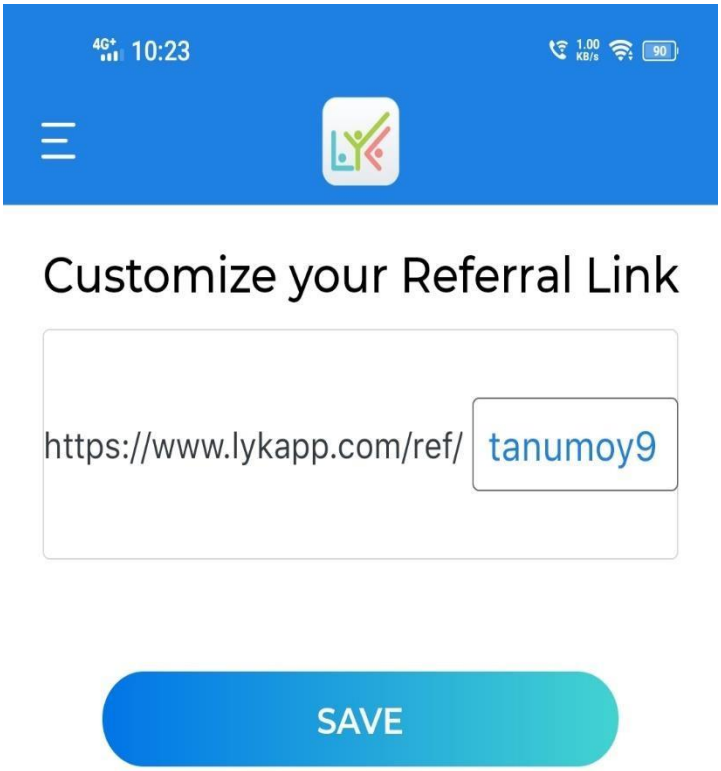



The api params is an object that contains the parameters to be sent to the API endpoint, including the encrypted user ID and the pagination parameters.

The API function uses an arrow function with the 'async' keyword to define the success callback. When the API call returns a response, this function is executed. It takes a single parameter, 'resp', which is the response data returned by the server.

The success callback of the API function tries to log the response data to the console using console.log(). If the response data is not null, the function sets the referral list data using setdata(). If the response data is null, it displays an error message using Toast.show().

If there is an error during the API request, the function uses another arrow function with the 'async' keyword to define an error callback. This function takes a single parameter, 'error', which is the error data returned by the server and displays an error message using Toast.show().




If we press `Customize Your Link` button then we navigate to the the `Customize your Referral Link` page which is shown in the left position.

In this page the user can change his/her referral link. For changing the referral link the user must have to enter minimum of 6 characters else, the error message will be shown that please enter atleast 6 characters. After filling up atleast 6 characters when enter the save button then an api https://api.lykapp.com/lykjwt/

Index.php?/LYKUser/updateRefCode is called with the parameters of {`userId`,`referCode`}.

The function includes two callback functions: one for handling the response (async (resp) => { ... }) and another for handling errors (async (error) => { ... }). Inside the response callback, Checks if the response object is not null: if (resp != null) { ... }. If the response object has a property called 'duplicate', it extracts its value: let duplicate = resp.duplicate;

Checks if the value of duplicate is false. If it is false, it performs the following action:

Navigates to the 'MyReferrals' screen using the navigation object. Shows a toast notification with the message 'Referral Link Updated'.

If it is not false, it sets the text variable with a specific message. If the response object is null, it shows a toast notification with the message 'OTHER_ERROR'.

If any errors occur within the try block, they are caught and logged to the console.


If we navigate to the details of any referred people then the details of that person wil be visible to the user i.e, profile pic, cover pic, interests of that person, his/her posts and activity will be shown. To fetch all the data, connections and activity of any reffered person we call the api

https://api.lykapp.com/lykjwt/index.

php?/fetchUserProfileDetails

with the parameters of userId i.e, the user who is logged in this page and the relativeId i.e, the id of the person by whom the user reffered.

The api call is made with in a functions which uses the PhpApiRequest.call function to make a request with the above parameters

Inside the response callback, the code performs the following actions:

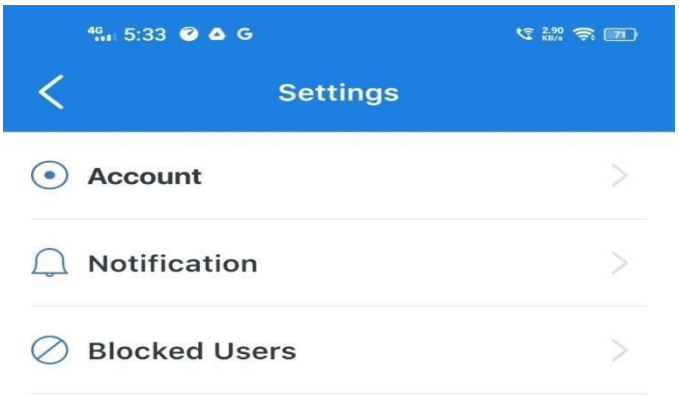
1) First Checks if the response object is null or not null: if (resp != null) { ... }
2) If the response object is not null, it performs the following actions:
3) Logs the message "Response is ok" to the console: console.log('Response is ok');
4) Sets the user variable with the response object: setUser(resp);
5) If the response object has a property called 'firstName', it performs the following actions:
6) Logs the message "First Page present in the response" to the console: console.log('First Page present in the response');
7) Sets the name variable with the value of resp.firstName.

8) If the response object does not have a property called 'firstName', it shows a toast notification with the message 'OTHER_ERROR'.
9) If any errors occur within the try block, they are caught and logged to the console.

Inside the error callback, the code performs the following actions:

- Shows a toast notification with the message 'OTHER_ERROR'.
- If any errors occur within the try block, they are caught and logged to the console.

# 4.4- LYK App MY Settings Screen

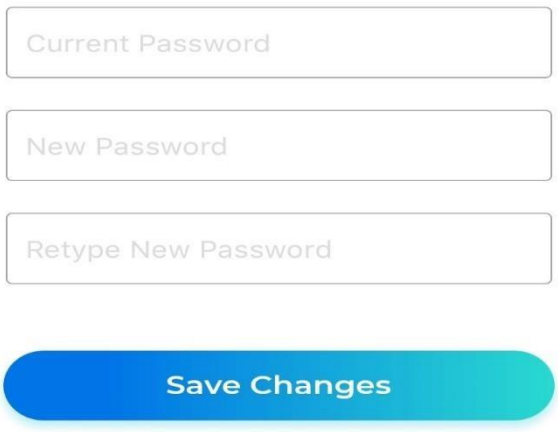



In the Settings option there are three options Account, Notification, Blocked Users options. From the Account option there are Edit Profile, Change Password, Manage Account option.

In the change Password screen there are three field Current Password, New Password and Retype New Password. For updating the password we first check if the Current password field is empty or not. If is it sets an error message to indicate that the field is required.

If the current password field is not empty, it clears the error message.


The code then proceeds to the to check the New password field. If it is empty it sets an error message.




If the new password field is not empty, it clears the error message.

The code then compares the new password and comfirm password fields. If they do not match, it sets an error message.

If the new password and confirm password field match, it clears the error message also the code will also

check that the new password has a length less than 4 characters or not. If it does it sets an error message.

Finally the code checks if the the current password and new password fields are same or not. If these fields are same and all the validation checks pass, the function returns true to indicate that the form is valid. The the api https://api.lykapp.com/lykjwt/

index.php?/LYKUser/changePsWord is called with the parameters of userId, new password and current password.

If the response is not null, it checks if it has a property called `success`. This is likely an indication that the password change request was processed successfully. If the success property exists and its value is true, it performs the following actions:

- Navigates to the 'SettingsScreen'
- Displays a toast message indicating that the password was changed successfully.

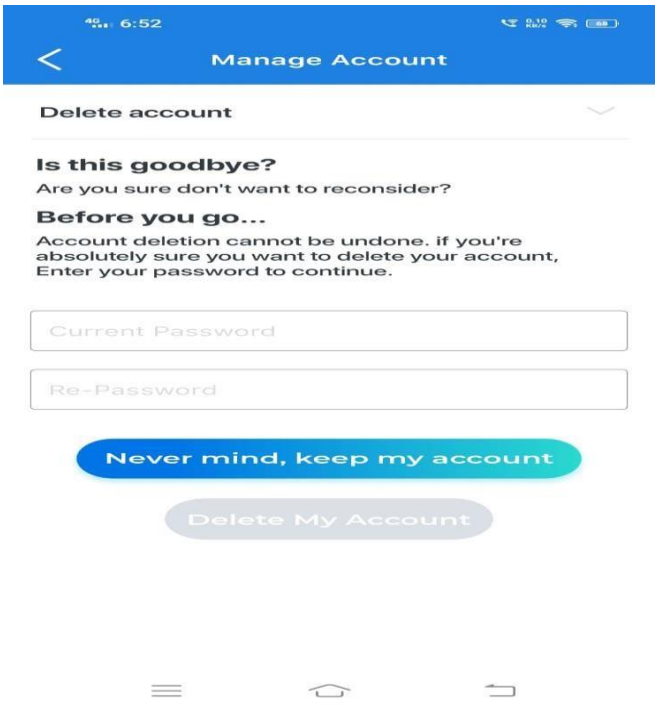
If the success property exists and its value is `false`, it sets an error message to indicate that the current password provided is invalid. If the response is null or doesn't have a success property, it displays a generic error toast message. If an error occurs during the API request, it sets the loading state to false and displays a generic error toast message.

# 4.5- LYK App MY Manage Account Screen

From the manage account screen of the app we can put out current password and re type password again. If one of the field is empty then an message will show that that field is required. The code checks if the the current password and re password fields are same or not. If these fields are same and all the validation checks pass, the function returns true to indicate that the form is valid. Generally from the manage account screen one user can delete his/her account. To do that once an user enter the password and current password and press the Delete my account then an api `https://api.lykapp.com/lykjwt/index.php?/LYKUser/deleteUser` api is called with the parameter of userId of the user.

If the response is not null, it checks if it has a property called `success`. This is likely an indication that the user deletion request was processed successfully. If the success property exists and its value is true, it performs the following actions:

- To remove two items from Asynstorage using the `Asyncstorage.removeItem()` method. The first item to be removed is referenced by the User information.
- The second item to be removedis the params.Token constants.



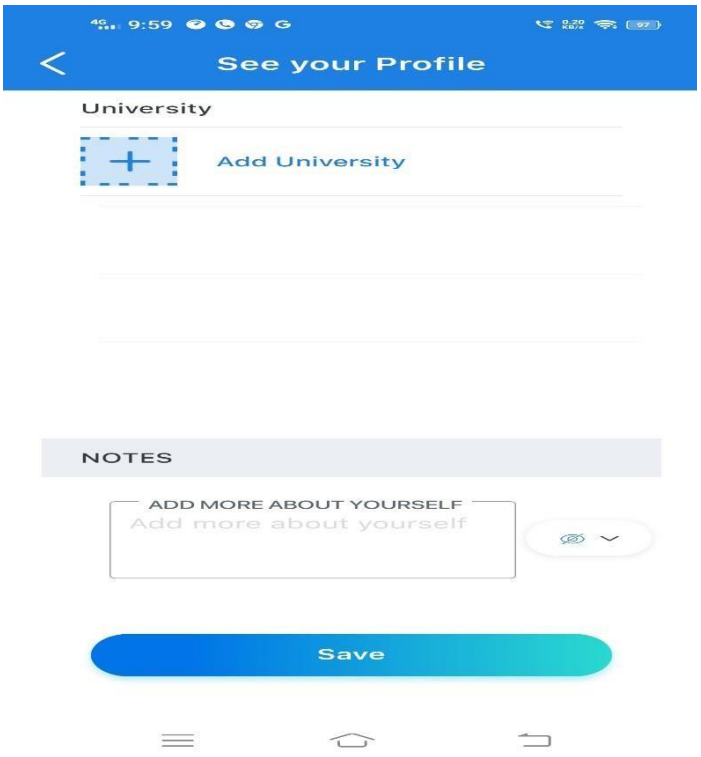

If the success property exists and its value is false, it sets an error message to indicate that the current password provided is invalid. If the response is null or doesn't have a success property, it displays a generic error toast message.

If an error occurs during the API request (in the error callback function), it sets the loading state to false and displays a generic error toast message.

# 4.6- LYK App edit profile Screen

In this page one user can edit his/her profile, the user can edit their proile photo, Cover photo, Birth date, Birth year, Email, mail, their interests, Hobbies, Country of Birth, city of Birth,Marital Status, Profession also the visibility rules all of that field that is all the field is in the public or the Private mode. Also the user can edit their High School, College, University and Notes by pressing on the save button. From the edit page user can also change their personal information that is their photo,name and many others personal information. The user can add multiple school, college, university name. In this page the user's have also the accessibility that all the personal information is shown to the other users or not. If the user wants to show all the details to other users then the visibility should be kept in the public mode or if the users does not want to show all of its personal information to the another person then the user should keep the visibility mode in the private mode. Once the user first render to the edit page then all the details of the user is shown in the screen and the api which is described in the below `https://api.lykapp.com/lykjwt/index.php?/LYKUser/fetchUserProfile` is called to fetch all the information about the user.

Generallay we are using AsyncStorage to retrieve user details from local storage and then making an API request to fetch user profile information. It uses AsyncStorage.getItem() to retrieve the user details stored under the key Params.USER_INFO. The await keyword is used



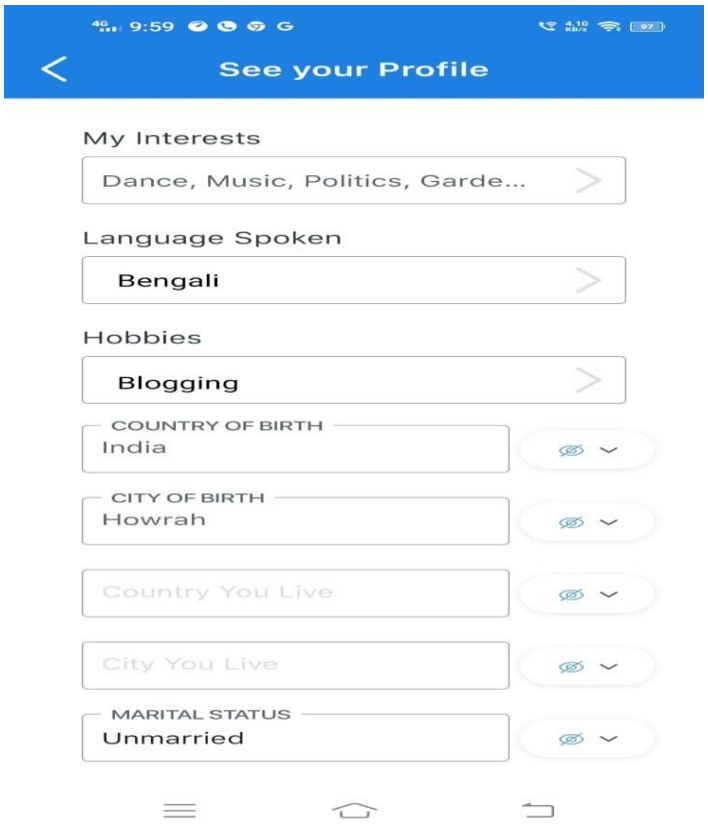

to wait for the promise to resolve and the retrieved value is assigned to the userDetails variable. The params object is created with a userId property, which is obtained by calling the getEncUserId() function with the userId property from userDetails.

The PhpApiRequest.call() function is called with the FETCH_USER_PROFILE API endpoint, the params object, and some callback functions for handling the success and error responses.

Inside the success callback function, the response is checked for not being null. If the response is not null, various properties from the response object are accessed and used to update state variables or perform other operations.

Basically we fetch user details from local storage, make an API request to fetch additional user profile information, and update state variables or perform other operations based on the response.

Threre are many visibility modals for each of the interest field, if any user want to change any of these visbility public to private or vice versa then the corresponding api will be called to do this job successfully. After atlast when the user edit all the field according to their choices and press the save button then the api `https://api.lykapp.com/lykjwt /index.php?/LYKUser/edit` is called with parameter with various properties that are used to send data in the Api request. The `userId` property is set by calling the getEncUserId() function with the userId property from userDetails.




Some properties are conditionally set based on certain variables or values. For example, the `dateOfBirth`, `countryISO`, `countryCode`,` countryName`, `countryOfBirth`, `countryYouLive`, `cityYouLive`,

`maritalStatus` and about properties are set based on either user input or values from the user object.

The PhpApiRequest.call() function is called with the EDIT API endpoint, the params object, and callback functions for handling the success and error responses. Inside the success callback function, the response is checked for not being null. If the response is not null, various conditions are checked to determine the type of response and perform specific actions based on it.

If the response has an 'input' property, it means that user information was successfully updated. The properties in the user object are then updated with the corresponding values from the response's 'input' object.

Overall, this code appears to be responsible for sending updated user information to the API and handling the response to update the user's state and perform necessary actions.

# Chapter 5- Conclusion

My goals were to find out what React Native is and how does it work, and to find out how to develop a cross-platform application with React Native. In my opinion, I reached my goals only partly. Due to limited time left to finish my degree, I had to leave out the whole part of cross-platform development, so I'm going to have to explore the Android part of React Native later. Also I didn't yet get to deploy the application on a real iPhone device, not to even mention publishing it at Apple's App Store.

I found out React Native is a good choice for creating mobile applications and especially well-suited for programmes with a background in web development. It basically allows you to develop iOS and Android applications just like you would develop a web page, but still being a real native application using the native API's instead of just rendering a HTML view. It's still new technology though, and I think this is visible from the result of this thesis also. There are not that many books available about React Native, so the sources are a bit thin.

Based on the results React Native, and cross-platform mobile technologies in general, can be a viable option to developing native mobile applications. Certain requirements do not change, and it is still the designers' and developers' responsibility to create the native user experience for each platform, and as such, mobile development still requires knowledge of the specific user experience and design guidelines for each platform. Developing the application by using cross-platform technologies can make the development process easier and less time consuming in some ways, and the maintainability of the application should also increase due to code reuse.

Even though cross-platform technologies may be a viable option for mobile development, based on this study alone it is unclear when they have a clear advantage over native development. In simple applications the differences in development time and maintainability are not significant, and in more complex applications the increased abstraction might not offer any benefits to the developers. In fact, in some cases, when we need to implement a lot of platform native features to the application, cross-platform technologies might only act as a hindrance to the progress of the application.

It has been a long-running project, writing this thesis. I've had some trouble making up my mind about the topic and technologies used, but in the end I'm still happy with my choices. This topic was useful, fun and provided a real learning experience, I previously knew nothing about. I'm quite happy with this learning experience, and I think this thesis gives a good view on what's the general idea of React Native and how to use it.

I'm currently not that happy with the end result though, this application is far from being released or being useful to anyone. Only listing some data from the API is not that useful, so there are a lot of features in the to-do list.

Surprisingly it seems like the largest potential issue for React Native and similar technologies comes from maintainability. In a React Native project one not only relies on Google and Apple to keep their own SDKs up to date, one also relies on the whole group of third party organizations and people to keep their frameworks, libraries and tools up to date as well. This

is not a problem when everything works, and the open source community has definite strength in that regard. The real issues appear when even one of the third party frameworks, libraries or tools is dropped from the support and maintenance.

# References


1. L. Sydow, S. Cheney, App Annie 2017 Retrospective, Jan. 2018, Available (accessed 17.11.2018): https://www.appannie.com/en/insights/market-data/app-annie-2017-retrospective/

2. IDC, Smartphone OS Market Share, 2018, Available: https://www.idc.com/promo/smartphone-market-share/os

3. N. Hansson, T. Vidhall, Effects on performance and usability for cross-platform application development using React Native, PhD thesis, 2016.

4. Expo v30.0.0 documentation, Available: https://docs.expo.io/versions/v30.0.0/

5. Google, Android Developer Documentation, Available (accessed 3.11.2018): https://developer.android.com/index.html

6. Apple, iOS Developer Documentation, Available: https: //developer.apple.com/

7. Facebook, React Native Documentation, Available: https://facebook.github.io/react-native/index.html

8. Facebook, React Documentation, Available: https://reactjs.org/docs

9. Redux Documentation, Available: https://redux. js.org

10. React Navigation documentation, Available: https://reactnavigation.org

11. Apple, Code Signing, Available: https://developer. apple.com/support/code-signing/

12. Google Books API documentation, Available: https://developers.google.com/books/

13. ISBNdb API documentation, Available: https://isbndb.com/apidocs

14. Expo v30.0.0 documentation, Available: https://docs.expo.io/versions/v30.0.0/