SARANATHAN COLLEGE OF ENGINEERING

(An Autonomous Institution)

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

Venkateswara Nagar, Panjappur, Tiruchirapalli-620012

Name: _____

Register No.: _____

Semester and Branch: _____

Subject Code and Name: _____

Bonafide record of the work done

in the laboratory during the period

_____ *to* _____

Head of the Department

Faculty in-charge

Date:

Submitted for the practical examination held at **SARANATHAN COLLEGE OF ENGINEERING, Tiruchirappalli** on _____

Internal Examiner

External Examiner

TABLE OF CONTENTS

Ex.No.	Date	Experiment Name	Page No.	Signature
1		Cross Platform BMI application		
2		Cross platform expense manager application		
3		Cross platform application to convert imperial system to metric system		
4		Cross platform application for day-to day task management		
5		User login application using cordova		
6		To find and display the current location of the user using cordova		
7		Library access management application using java		

Staff In-charge

SARANATHAN COLLEGE OF ENGINEERING

An Autonomous Institution

Venkateswara Nagar, Panjappur

Tiruchirappalli – 620012

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision of the Department:

To evolve as a centre of academic excellence and advanced research in Computer Science and Engineering discipline.

Mission of the Department:

1. To inculcate in students a profound understanding of fundamentals related to discipline.
2. To inculcate skills, attitudes and their applications in solving real world problems with an inclination towards societal issues and research.
3. To promote research in the emerging areas of computer science and technology

Program Educational Objectives: (PEOs)

PEO1: Acquire strong foundation in the mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problems.

PEO2: Develop the ability to analyze the requirements of the software, understand the technical specifications, design and provide novel engineering solutions and efficient software/hardware designs.

PEO3: Have exposure to emerging cutting edge technologies, adequate training & opportunities to work as teams on multidisciplinary projects with effective communication skills and leadership qualities.

PEO4: Have awareness on the life-long learning and prepare them for research development and consultancy.

PEO5: Have a successful career and work with values & social concern bridging the digital divide and meet the requirements of Indian and multinational companies.

Program Specific Objectives: (PSOs)

PSO1: Foundation of mathematical concepts: To use mathematical methodologies to crack problem using suitable mathematical analysis, data structure and suitable algorithm.

PSO2: Foundation of Computer System: the ability to interpret the fundamental concepts and methodology of computer systems. Students can understand the functionality of hardware and software aspects of computer systems.

PSO3: Foundations of Software development: the ability to grasp the software development lifecycle and methodologies of software systems. Possess competent skills and knowledge of software design process. Familiarity and practical proficiency with a broad area of programming concepts and provide new ideas and innovations towards research

Program Outcomes: (POs)

S.No.	Program Outcomes
1	**PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and engineering specialization to the solution of complex engineering problems.
2	**PO2: Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3	**PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4	**PO4: Conduct Investigations of Complex Problems** Use research-based knowledge and research methods including design of exercises, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5	**PO5: Modern Tool Usage** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6	**PO6: The Engineer and Society** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7	**PO7: Environment and Sustainability** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8	**PO8: Ethics** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9	**PO9: Individual and Team Work** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10	**PO10: Communication** Communicate effectively on complex engineering activities with the engineering community and with society at large, communication such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11	**PO11: Project management and finance** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12	**PO12: Life-long learning** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

SARANATHAN COLLEGE OF ENGINEERING

An Autonomous Institution
Approved by AICTE-New Delhi, Affiliated Anna University, Chennai
Venkateswara Nagar, Panjappur, Tiruchirappalli -620012

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2025-2026 ODD SEMESTER

Name of Lab Course	**CCS332 / APP DEVELOPMENT**
Semester & Year	**VII & IV**
Name of the student	
Name of the Evaluator	**Mr. K.S Chandrasekaran**
Marks scored out of 20	

RUBRIC ASSESSMENT FOR LAB COURSES

Performance Indicators	Level 1 (0-1)	Level 2 (2-4)	Level 3 (5)
Pre lab questions, objectives (P-1)	Explanation to, the Pre lab questions and objectives of the experiment, is, where compared to the expectation of the faculty is not satisfactory.	Explanation to, the Pre lab questions and objectives of the experiment, is, where compared to the expectation of the faculty is partially satisfactory.	Explanation to, the Pre lab questions and objectives of the experiment, is, where compared to the expectation of the faculty is highly satisfactory.
Procedures (P-11)	Explanation to the procedure of the experiment, is, where compared to the expectation of the faculty is not satisfactory.	Explanation to the procedure of the experiment, is, where compared to the expectation of the faculty is partially satisfactory.	Explanation to the procedure of the experiment, is, where compared to the expectation of the faculty is highly satisfactory.
Data/ Observations (P-111)	Calculation of the observed values and validation of the results of the experiment inaccurate.	Calculation of the observed values and validation of the results of the experiment approximate	Calculation of the observed values and validation of the results of the experiment precise.
Post lab questions, Conclusions (P-IV)	Explanation to the Post lab questions and Conclusions of the experiment, is, where compared to the expectation of the faculty is not satisfactory.	Explanation to the Post lab questions and Conclusions of the experiment is, where compared to the expectation of the faculty partially satisfactory.	Explanation to the Post lab questions and Conclusions of the experiment, is, where compared to the expectation of the faculty is highly satisfactory.

Register No : _____

Assessment Sheet

Sno	Date	Name of the Exercise	P -I	P-II	P-III	P-IV	TOTAL
			[5]	[5]	[5]	[5]	[20]
1		Cross Platform BMI application					
2		Cross platform expense manager application					
3		Cross platform application to convert imperial system to metric system					
4		Cross platform application for day-to-day task management					
5		User login application using cordova					
6		To find and display the current location of the user using cordova					
7		Library access management application using java					

SIGNATURE OF EVALUATOR

Course Outcomes:

CO Code	Course Outcomes
CO1	Understand different actions performed through Version control tools like Git.
CO2	Perform Continuous Integration and Continuous Testing and Continuous Deployment using Jenkins by building and automating test cases using Maven.
CO3	Perform Continuous Integration and Continuous Testing and Continuous Deployment using Jenkins by building and automating test cases using Gradle.
CO4	Ability to Perform Automated Continuous Deployment.
CO5	Ability to do configuration management using Ansible.
CO6	Understand to leverage Cloud-based DevOps tools using Azure DevOps.

CO-PO Mapping:

CO Code	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3	2	3	-	-	-	-	-	-	-	2	2	2
CO2	3	3	3	2	3	-	-	-	-	-	-	-	2	2	2
CO3	3	3	3	2	3	-	-	-	-	-	-	-	2	2	2
CO4	3	3	3	2	3	-	-	-	-	-	-	-	2	2	2
CO5	3	3	3	2	3	-	-	-	-	-	-	-	2	2	2
CO6	3	3	3	2	3	-	-	-	-	-	-	-	2	2	2

SYLLABUS

CCS332-APP DEVELOPMENT

LIST OF EXPERIMENTS : 30 PERIODS

1. Using react native, build a cross platform application for a BMI calculator.
2. Build a cross platform application for a simple expense manager which allows entering expenses and income on each day and displays category wise weekly income and expense.
3. Develop a cross platform application to convert units from imperial system to metric system (km to miles, kg to pounds etc.,)
4. Design and develop a cross platform application for day to day task (to-do) management.
5. Design an android application using Cordova for a user login screen with username, password, reset button and a submit button. Also, include header image and a label. Use layout managers.
6. Design and develop an android application using Apache Cordova to find and display the current location of the user.
7. Write programs using Java to create Android application having Databases
    - For a simple library application.
    - For displaying books available, books lend, book reservation. Assume that student information is available in a database which has been stored in a database server.

Ex No:1 CROSS PLATFORM BMI APPLICATION

Aim:

To use react native and build a cross platform application for a BMI calculator.

Algorithm:

Step 1: Start the Program
Step 2: Import the necessary libraries such as React, useState from 'react' and View, Text, TextInput, Button, StyleSheet from 'react-native'
Step 3: Create a function BMI() where Input and Output Value States are Set.Initially the height and weight is set into empty string
Step 4: Create a function calculateBMI()
(4.1)Find the height in meters and store it in 'heightMeters'
(4.2)Find BMI Value using the formula weight/(height*height)
(4.3)Display the output using setBMI()
Step 5: Get the input from the user using a TextInput
Step 6: On press of the calculate button,the function calculateBMI() is called and the output is displayed to the user
Step 7: Stop the program

Procedure:

To install all the necessary tools such as Android Studio, React Native CLI, NodeJS, JDK 17

System Requirements: Windows 11/10 Operating System, minimum of 8 GB RAM Size,x86_64 bit CPU

1) Download and Install NodeJS from https://nodejs.org/en/download
2) Download and Install JDK 17 from https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html
3) Download and Install Android Studio from https://developer.android.com/studio?gad_source=1&gclid=Cj0KCQjwhtWvBhD9ARIsAOP0GoiX-2m4Ttv5QNTJtTqD8jCO39STBiCUeeAvlvcDUOXRuT7euX6Hb7gaAje4EALw_wcB&gclsrc=aw.ds
4) While on Android Studio installation wizard, make sure the boxes next to all of the following items are checked:
    - Android SDK
    - Android SDK Platform

- Android Virtual Device

5) For Building a React Native App with native code then it requires Android14(UpsideDownCake) in particular.
6) To do that, open Android Studio, click on "More Actions" button and select "SDK Manager".Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner. Look for and expand the Android 14 (UpsideDownCake) entry, then make sure the following items are checked:
   - Android SDK Platform 34
   - Intel x86 Atom_64 System Image or Google APIs Intel x86 Atom System Image
7) Next, select the "SDK Tools" tab and check the box next to "Show Package Details" here as well. Look for and expand the Android SDK Build-Tools entry, then make sure that 34.0.0 is selected.
8) Finally, click "Apply" to download and install the Android SDK and related build tools.

   The React Native tools require some environment variables to be set up in order to build apps with native code.
   - ➔ Open the Windows Control Panel.
   - ➔ Click on User Accounts, then click User Accounts again
   - ➔ Click on Change my environment variables
   - ➔ Click on New... to create a new ANDROID_HOME user variable that points to the path to your Android SDK
   - ➔ If SDK is installed, by default, at the following location: %LOCALAPPDATA%\Android\Sdk
9) Add platform-tools to Path
   - ➔ Open the Windows Control Panel.
   - ➔ Click on User Accounts, then click User Accounts again
   - ➔ Click on Change my environment variables
   - ➔ Select the Path variable.
   - ➔ Click Edit.
   - ➔ Click New and add the path to platform-tools to the list.
   - ➔ The default location for this folder is: %LOCALAPPDATA%\Android\Sdk\platform-tools
10) Install React Native CLI using the command:
    npm install -g react-native-cli
11) Create a React Native Project using the command:
    npx react-native@latest init AwesomeProject
12) Start your emulator from Android Studio using Virtual Device Manager.Open command prompt from the project directory and run

the command: "npm run android" or just run the command: "npm start" and then "npm run android"

13) You can modify your project and edit App.tsx from any IDE

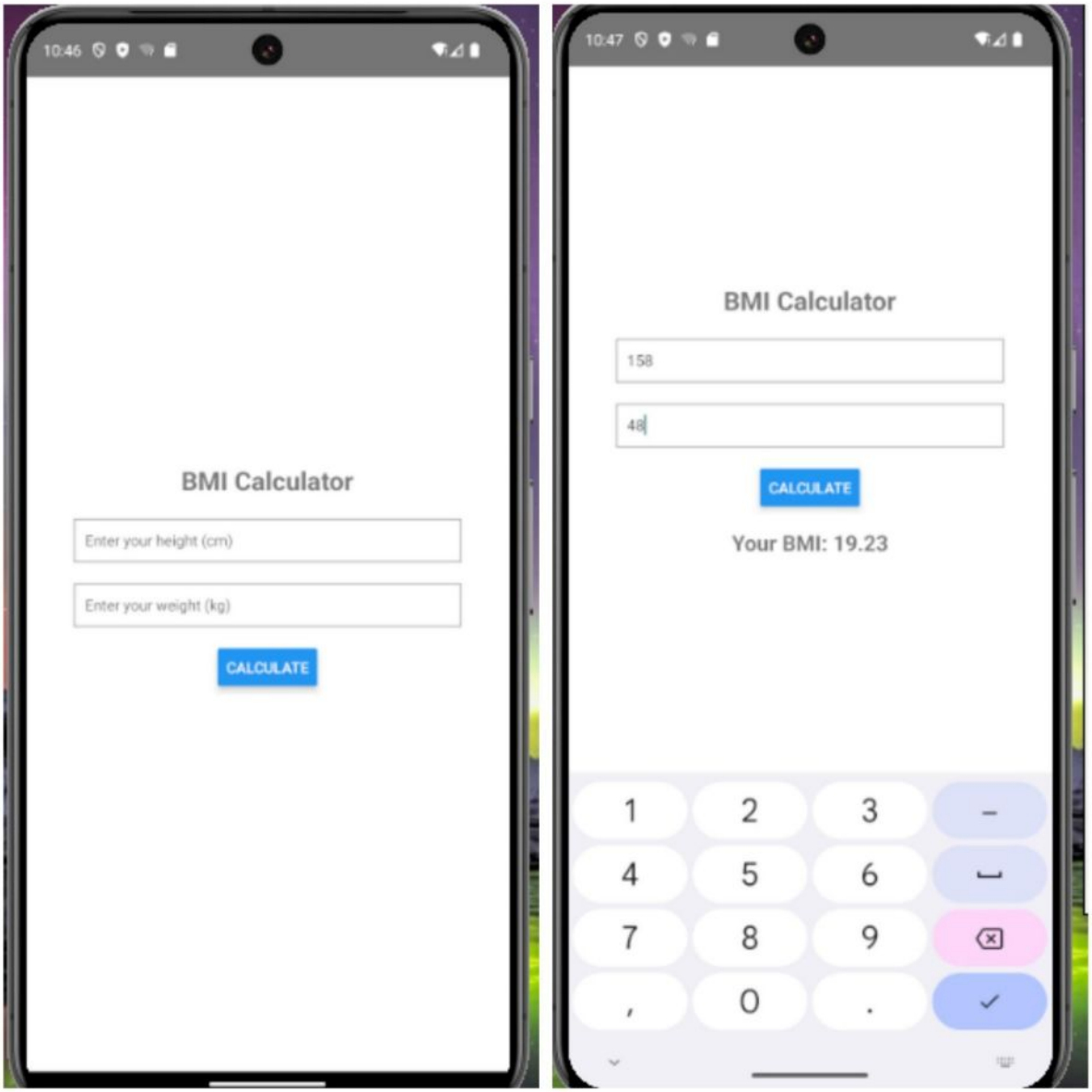
Program:

App.tsx

```
import React, { useState } from 'react';
import { View, Text, TextInput, Button, StyleSheet } from 'react-native';

const BMI = () => {
  const [height, setHeight] = useState('');
  const [weight, setWeight] = useState('');
  const [bmi, setBMI] = useState(null);

 const calculateBMI = () => {
    const heightMeters = height / 100;
    const bmiValue = weight / (heightMeters * heightMeters);
    setBMI(bmiValue.toFixed(2));
  };

 return (
    <View style={styles.container}>
      <Text style={styles.title}>BMI Calculator</Text>
      <TextInput
        style={styles.input}
        placeholder="Enter your height (cm)"
        onChangeText={text => setHeight(text)}
        keyboardType="numeric"
      />
      <TextInput
        style={styles.input}
        placeholder="Enter your weight (kg)"
        onChangeText={text => setWeight(text)}
        keyboardType="numeric"
      />
      <Button title="Calculate" onPress={calculateBMI} />
```

```
      {bmi && <Text style={styles.result}>Your BMI: {bmi}</Text>}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#fff',
  },

 title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
  },

input: {
    height: 40,
    width: '80%',
    borderColor: 'gray',
    borderWidth: 1,
    marginBottom: 20,
    paddingLeft: 10,
  },

result: {
    marginTop: 20,
    fontSize: 20,
    fontWeight: 'bold',
  },
});

export default BMI;
```

Output:



Result:

Thus,a cross platform BMI app is built using react native is executed and verified successfully.

Ex No:2 CROSS PLATFORM EXPENSE MANAGER APPLICATION

Aim:

To use react native and build a cross platform application for a Simple Expense Manager.

Algorithm:

Step 1: Start the Program
Step 2: Create a function App() in app.tsx where 'HomeScreen.js', 'WeeklySummaryScreen.js' and 'AddExpenseScreen.js' pages are imported.
Step 3: Import the necessary libraries such as NavigationContainer from '@react-navigation/native' and createStackNavigator from '@react-navigation/stack'
Step 4: Create AddExpenseScreen.js
(4.1) Import the necessary libraries such as "React, useState" from 'react' ,"View, Text, TextInput, Button, StyleSheet" from 'react-native' and "Asyncstorage" from '@react-native-async-storage/async-storage'
(4.2) Get Expense and category from the user
(4.3) Date and Time are loaded automatically
(4.4) All the data is stored into Async Storage in react-native
Step 5: Create HomeScreen.js
(5.1) Import the necessary libraries such as "useState, useEffect" from 'react' ,"View, Text, Button, StyleSheet" from 'react-native' and "Asyncstorage" from '@react-native-async-storage/async-storage'
(5.2) Get Income and category from the user
(5.3) Date and Time are loaded automatically
(5.4) All the data is stored into Async Storage in react-native
Step 6: Create WeeklySummaryScreen.js
(6.1) Import the necessary libraries such as "useState, useEffect" from 'react' ,"View, Text, Button, StyleSheet" from 'react-native' and "Asyncstorage" from '@react-native-async-storage/async-storage'
(6.2) Income, category, Date & Time are retrieved from the Async Storage
(6.3) Expense, category, Date & Time are retrieved from the Async Storage
(6.4) All the data is displayed to the user using a Object Constructor
Step 7: Stop the program

Program:

App.tsx

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import HomeScreen from "./screens/HomeScreen";
import AddExpenseScreen from "./screens/AddExpenseScreen";
import WeeklySummaryScreen from "./screens/WeeklySummaryScreen";

const Stack = createStackNavigator();

export default function App() {
  return (

    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="AddExpense" component={AddExpenseScreen} />
        <Stack.Screen name="WeeklySummary"
component={WeeklySummaryScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

AddExpenseScreen.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, Button, StyleSheet } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

const AddExpenseScreen = ({ navigation, route }) => {
  const [amount, setAmount] = useState('');
  const [category, setCategory] = useState('');
  const [date, setDate] = useState(new Date().toISOString());

  const { type } = route.params;

  const handleSave = async () => {
    try {
```

```
      // Retrieve existing data from AsyncStorage
      const existingData = await AsyncStorage.getItem(type);
      let newData = [];

      if (existingData) {
        newData = JSON.parse(existingData);
      }

      // Add new expense/income to the data array
      newData.push({ amount: parseFloat(amount), category, date });

      // Save the updated data to AsyncStorage
      await AsyncStorage.setItem(type, JSON.stringify(newData));

      // Navigate back to the Home screen
      navigation.navigate('Home');
    } catch (error) {
      console.error('Error saving data:', error);
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Add {type === 'income' ? 'Income' :
'Expense'}</Text>
      <TextInput
        style={styles.input}
        placeholder="Amount"
        keyboardType="numeric"
        value={amount}
        onChangeText={(text) => setAmount(text)}
      />
      <TextInput
        style={styles.input}
        placeholder="Category"
        value={category}
        onChangeText={(text) => setCategory(text)}
      />
      <TextInput
```

```
        style={styles.input}
        placeholder="Date"
        value={date}
        onChangeText={(text) => setDate(text)}
      />
      <Button title="Save" onPress={handleSave} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
  },
  input: {
    width: '80%',
    height: 40,
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    marginBottom: 10,
    paddingHorizontal: 10,
  },
});

export default AddExpenseScreen;
```

HomeScreen.js

```
import React, { useState, useEffect } from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
```

```
const HomeScreen = ({ navigation }) => {
  const [totalIncome, setTotalIncome] = useState(0);
  const [totalExpense, setTotalExpense] = useState(0);

  useEffect(() => {
    // Load data from AsyncStorage when the component mounts
    loadData();
  }, []);

  const loadData = async () => {
    try {
      // Retrieve data from AsyncStorage
      const income = await AsyncStorage.getItem('income');
      const expense = await AsyncStorage.getItem('expense');

      // Update state with the retrieved data
      if (income) setTotalIncome(parseFloat(income));
      if (expense) setTotalExpense(parseFloat(expense));
    } catch (error) {
      console.error('Error loading data:', error);
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Expense Manager</Text>
      <Text style={styles.total}>Total Income:
${totalIncome.toFixed(2)}</Text>
      <Text style={styles.total}>Total Expense:
${totalExpense.toFixed(2)}</Text>
      <View style={styles.buttonContainer}>
        <Button
          title="Add Income"
          onPress={() => navigation.navigate('AddExpense', { type: 'income' })}
        />
        <Button
          title="Add Expense"
          onPress={() => navigation.navigate('AddExpense', { type: 'expense' })}
        />
```

```
        <Button
          title="Weekly Summary"
          onPress={() => navigation.navigate('WeeklySummary')}
        />
      </View>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
  },
  total: {
    fontSize: 18,
    marginBottom: 10,
  },
  buttonContainer: {
    marginTop: 20,
  },
});

export default HomeScreen;
```

WeeklySummaryScreen.js

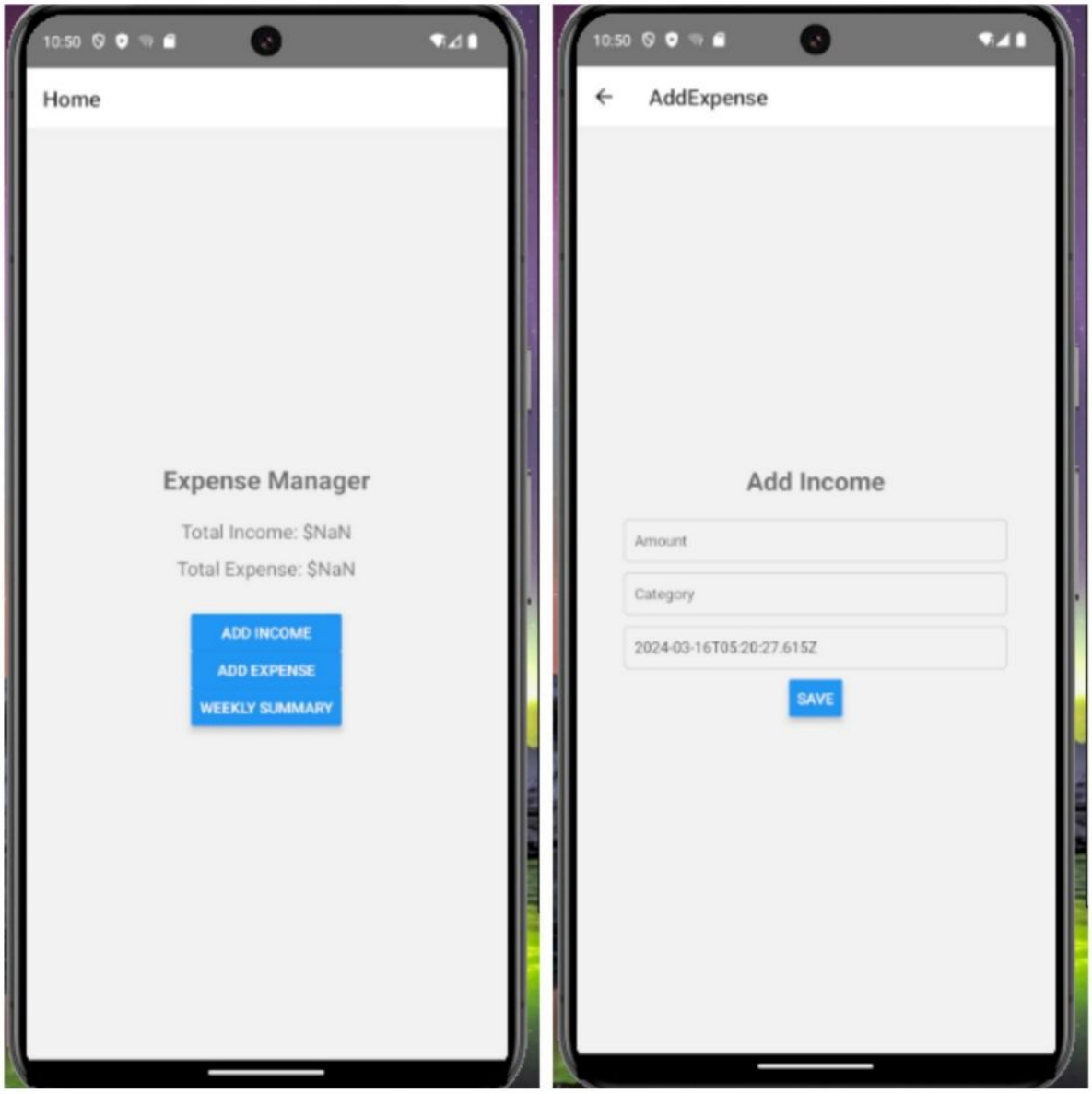
```
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

const WeeklySummaryScreen = () => {
  const [weeklySummary, setWeeklySummary] = useState([]);
```

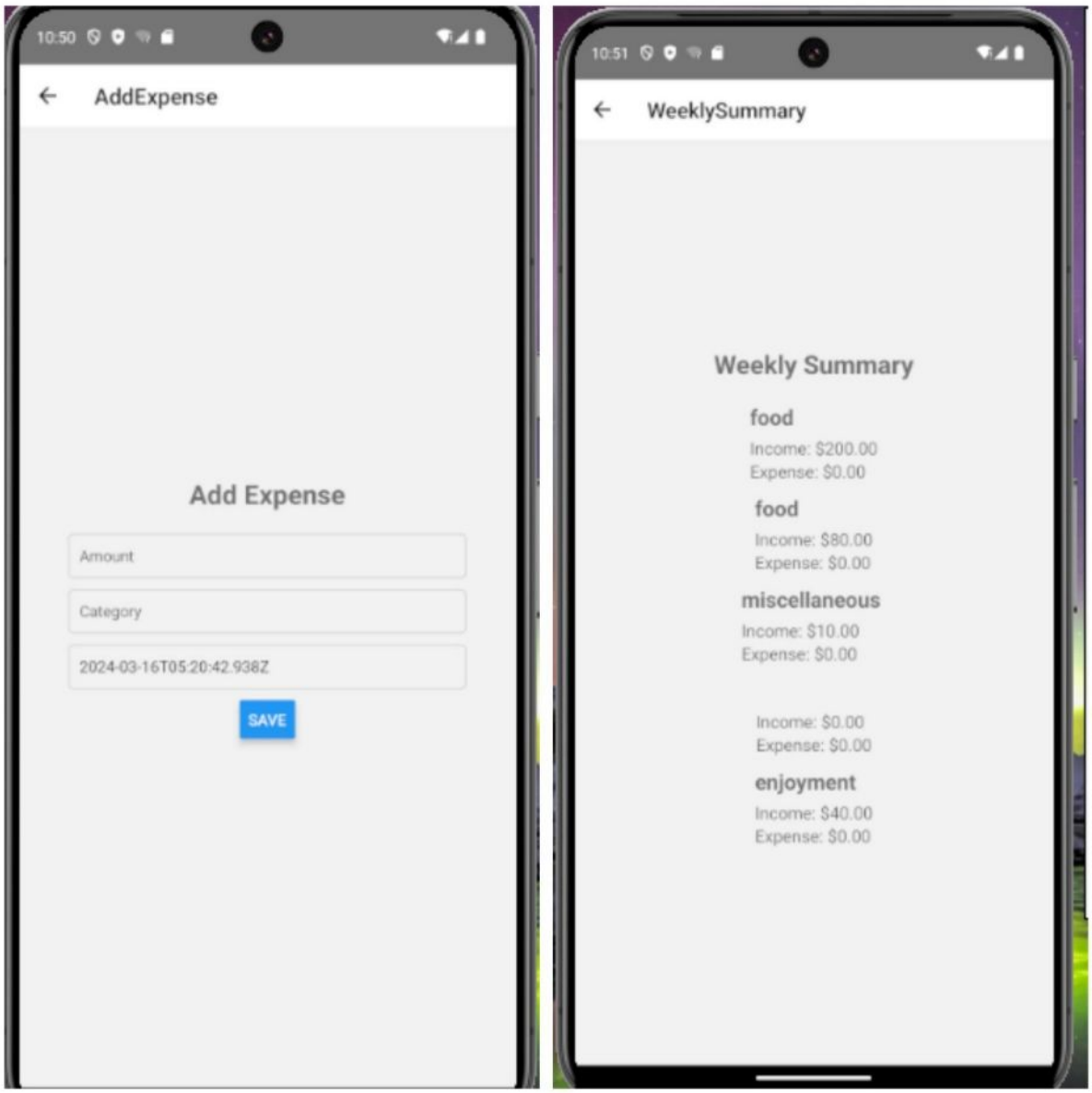
```
useEffect(() => {
  // Load weekly summary data when the component mounts
  loadWeeklySummary();
}, []);

const loadWeeklySummary = async () => {
  try {
    // Retrieve income and expense data from AsyncStorage
    const incomeData = await AsyncStorage.getItem('income');
    const expenseData = await AsyncStorage.getItem('expense');

    // Convert data to JSON arrays
    const incomeArray = incomeData ? JSON.parse(incomeData) : [];
    const expenseArray = expenseData ? JSON.parse(expenseData) : [];

    // Calculate weekly summary
    const startDate = new Date();
    startDate.setDate(startDate.getDate() - startDate.getDay());
    const endDate = new Date();
    endDate.setDate(startDate.getDate() + 6);

    const summary = {};
    incomeArray.concat(expenseArray).forEach((item) => {
      const itemDate = new Date(item.date);
      if (itemDate >= startDate && itemDate <= endDate) {
        const category = item.category;
        const amount = item.amount;
        if (!summary[category]) {
          summary[category] = { income: 0, expense: 0 };
        }
        if (item.amount > 0) {
          summary[category].income += amount;
        } else {
          summary[category].expense -= amount;
        }
      }
    });

    setWeeklySummary(summary);
```

```
    } catch (error) {
      console.error('Error loading weekly summary:', error);
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Weekly Summary</Text>
      {Object.keys(weeklySummary).map((category) => (
        <View key={category} style={styles.categoryContainer}>
          <Text style={styles.category}>{category}</Text>
          <Text style={styles.amount}>Income: $
{weeklySummary[category].income.toFixed(2)}</Text>
          <Text style={styles.amount}>Expense: $
{weeklySummary[category].expense.toFixed(2)}</Text>
        </View>
      ))}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
  },
  categoryContainer: {
    marginBottom: 10,
  },
  category: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 5,
```

```
  },
  amount: {
    fontSize: 16,
  },
});

export default WeeklySummaryScreen;
```

Output:





Result:

Thus,a cross platform Simple Expense Manager is built using react native is executed and verified successfully.

Ex No:3 CROSS PLATFORM APPLICATION TO CONVERT IMPERIAL SYSTEM TO METRIC SYSTEM

Aim:

To use react native and build a cross platform application to convert Imperial System to Metric System.

Algorithm:

Step 1: Start the Program
Step 2: Import the necessary libraries such as useState from 'react' and View, Text, TextInput, Button from 'react-native'
Step 3: Create a function UnitConverter() where Input and Output Value States are Set.Initially the state is set as null
Step 4: Create a function convertToMetric()
(4.1)The input value is converted into float using parseFloat() function
(4.2)kilometre value is
converted into miles using the formula kilometers = miles * 1.60934
(4.3)The output value is displayed using setOutput Value() and the it is rounded off to 2 digits
Step 5: Get the input from the user using a TextInput
Step 6: On press of the calculate button,the function convertToMetric() is called and the output is displayed to the user
Step 7: Stop the program

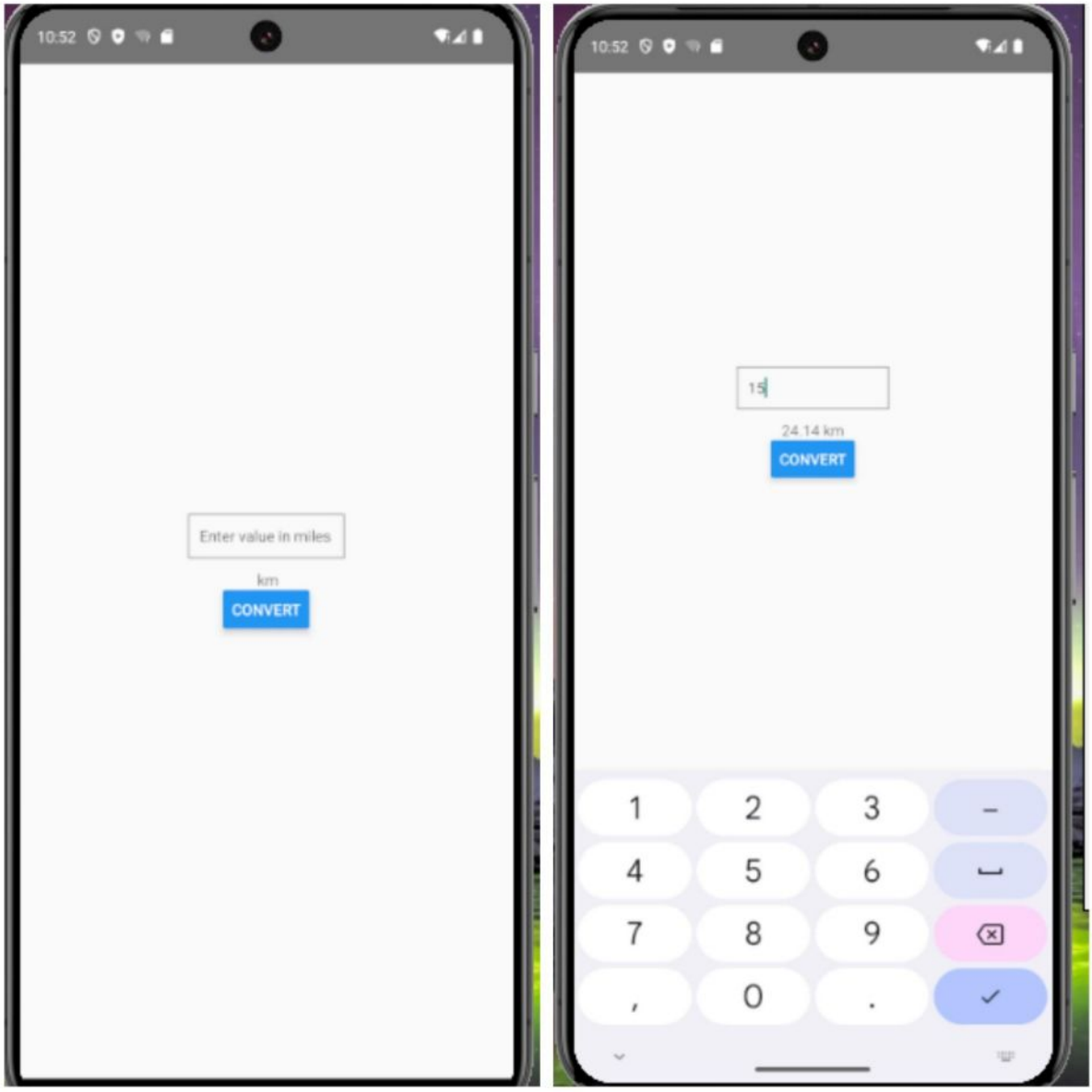
Program:

App.tsx

```
import React, { useState } from 'react';
import { View, TextInput, Text ,Button} from 'react-native';

const UnitConverter = () => {
  const [inputValue, setInputValue] = useState('');
  const [outputValue, setOutputValue] = useState('');

  const convertToMetric = () => {
    // Implement your conversion logic here
    // Example: miles to kilometers
    const miles = parseFloat(inputValue);
    const kilometers = miles * 1.60934;
    setOutputValue(kilometers.toFixed(2)); // Round to 2 decimal places
  };
```

```
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <TextInput
        style={{ height: 40, borderColor: 'gray', borderWidth: 1, marginBottom:
10, paddingHorizontal: 10 }}
        placeholder="Enter value in miles"
        keyboardType="numeric"
        onChangeText={text => setInputValue(text)}
        value={inputValue}
      />
      <Text>{outputValue} km</Text>
      <Button title="Convert" onPress={convertToMetric} />
    </View>
  );
};

export default UnitConverter;
```

Output:



Result:

Thus,a cross platform application to convert Imperial System to Metric System is built using react native is executed and verified successfully.

Ex No:4 CROSS PLATFORM APPLICATION FOR DAY-TO-DAY TASK MANAGEMENT

Aim:

To use react native and build a cross platform application for day-to-day task management.

Algorithm:

Step 1: Start the Program
Step 2: Import the necessary libraries such as useState from 'react' and View, Text, TextInput, Button, FlatList, StyleSheet from 'react-native'
Step 3: Create a function ToDoApp() where Input and Output Value States are Set
Step 4: Create a function addTask() where tasks are added into FlatList
Step 4: Create a function toggleTaskCompletion() where tasks are marked as completed in FlatList
Step 4: Create a function removeTask() where tasks are removed from FlatList
Step 5: Get the input from the user using a TextInput
Step 6: On press of the AddTask button,the function addTask() is called and the tasks are displayed to the user
Step 7: Stop the program

Program:

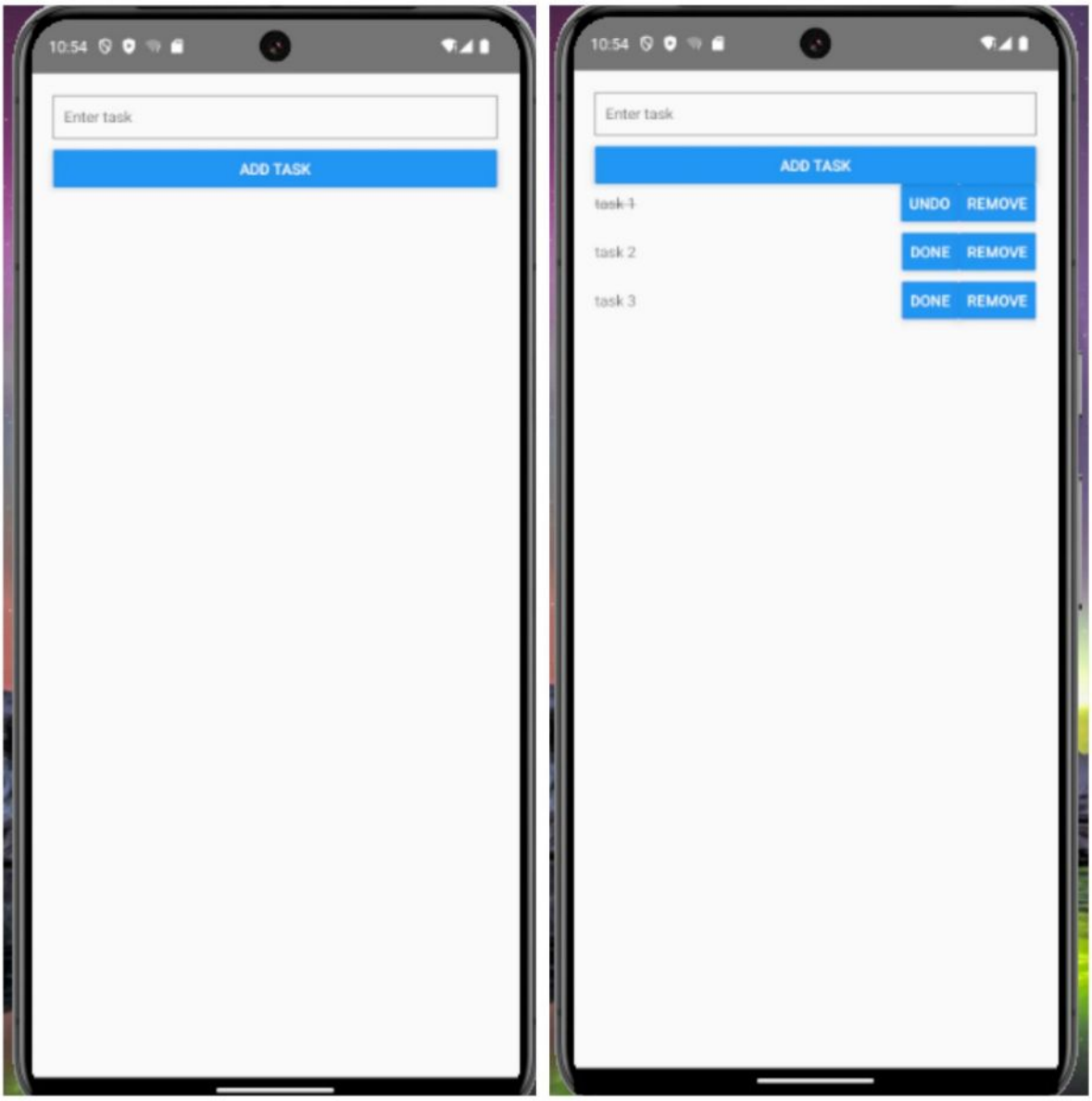
App.tsx

```
import React, { useState } from 'react';
import { View, TextInput, Button, FlatList, Text, StyleSheet } from 'react-native';

const ToDoApp = () => {
  const [task, setTask] = useState('');
  const [tasks, setTasks] = useState([]);

  const addTask = () => {
    if (task.trim() !== '') {
      setTasks([...tasks, { id: Date.now(), task: task, completed: false }]);
      setTask('');
    }
  };
```

```
  const toggleTaskCompletion = id => {
    setTasks(tasks.map(task => {
      if (task.id === id) {
        return { ...task, completed: !task.completed };
      }
      return task;
    }));
  };

  const removeTask = id => {
    setTasks(tasks.filter(task => task.id !== id));
  };

  return (
    <View style={styles.container}>
      <TextInput
        style={styles.input}
        placeholder="Enter task"
        onChangeText={text => setTask(text)}
        value={task}
      />
      <Button title="Add Task" onPress={addTask} />
      <FlatList
        data={tasks}
        keyExtractor={item => item.id.toString()}
        renderItem={({ item }) => (
          <View style={styles.task}>
            <Text style={{ textDecorationLine: item.completed ? 'line-through' :
'none' }}>{item.task}</Text>
            <View style={styles.buttonsContainer}>
              <Button title={item.completed ? "Undo" : "Done"} onPress={() =>
toggleTaskCompletion(item.id)} />
              <Button title="Remove" onPress={() => removeTask(item.id)} />
            </View>
          </View>
        )}
      />
    </View>
```

```
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    padding: 20,
  },
  input: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,
    marginBottom: 10,
    paddingHorizontal: 10,
  },
  task: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 10,
  },
  buttonsContainer: {
    flexDirection: 'row',
  },
});

export default ToDoApp;
```

Output:



Result:

Thus,a cross platform application for day-to-day task management is built using react native is executed and verified successfully.

Ex No:5 USER LOGIN APPLICATION USING CORDOVA

Aim:

To Design an android application using Cordova for a user login screen with username, password, reset button and a submit button. Also, include header image and a label. Use layout managers.

Algorithm:

Step 1: Start the program
Step 2: Create a new Cordova project using the following command:
"cordova create LoginApp com.example.LoginApp UserLogin"
Step 3: Open your command prompt or terminal and navigate to the directory where your Cordova
project is located.
Step 4: Add Android as a platform to your Cordova project using the following command:
"cordova platform add android"
Step 5: Open the 'index.html' file located in the 'www' directory of your Cordova project.
Step 6: Design your UI layout using HTML, CSS, and JavaScript.
Step 7: Create HTML elements for username input, password input, reset button, submit button,
header image, and label.
Step 8: Style your UI elements using CSS to achieve the desired layout and appearance.
Step 9: Implement JavaScript functions to handle user interactions such as button clicks.
Step 10: Validate user input for username and password fields. and output is displayed to the user.
Step 11: Connect your Android device or start an Android emulator. Then run:"cordova run android".
Step 12:Stop the program.

Procedure:

System Requirements: Windows 11/10 Operating System, minimum of 8 GB RAM Size,x86_64 bit CPU

1) Download and Install NodeJS from https://nodejs.org/en/download
2) Download and Install Git Client from https://git-scm.com/downloads
3) Install Cordova using the command "npm install -g cordova"
4) To Download and Install Gradle from

- ➔ Prerequisites for Gradle: Java 8 or higher should be available
- ➔ Create a new directory C:\Gradle with File Explorer
- ➔ Open a second File Explorer window and go to the directory where the Gradle distribution was downloaded. Double-click the ZIP archive to expose the content. Drag the content folder gradle-8.6 to your newly created C:\Gradle folder.
- ➔ In File Explorer right-click on the This PC (or Computer) icon, then click Properties -> Advanced System Settings -> Environmental Variables.
- ➔ Under System Variables select Path, then click Edit. Add an entry for C:\Gradle\gradle-8.6\bin. Click OK to save.
- ➔ Verify your Installation by "gradle -v"

Program:

Index.html

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no">
   <title>User Login</title>
   <link rel="stylesheet" type="text/css" href="css/index.css" />
</head>
<body>
   <div class="container">
      <header>
         <img src="img/header_image.png" alt="Header Image">
         <h1>Login</h1>
      </header>
      <div class="login-form">
         <label for="username">Username:</label>
         <input type="text" id="username" name="username" required>
         <label for="password">Password:</label>
         <input type="password" id="password" name="password" required>
         <div class="buttons">
            <button id="resetBtn">Reset</button>
            <button id="submitBtn">Submit</button>
         </div>
      </div>
   </div>
   <script type="text/javascript" src="js/index.js"></script>
</body>
</html>
```

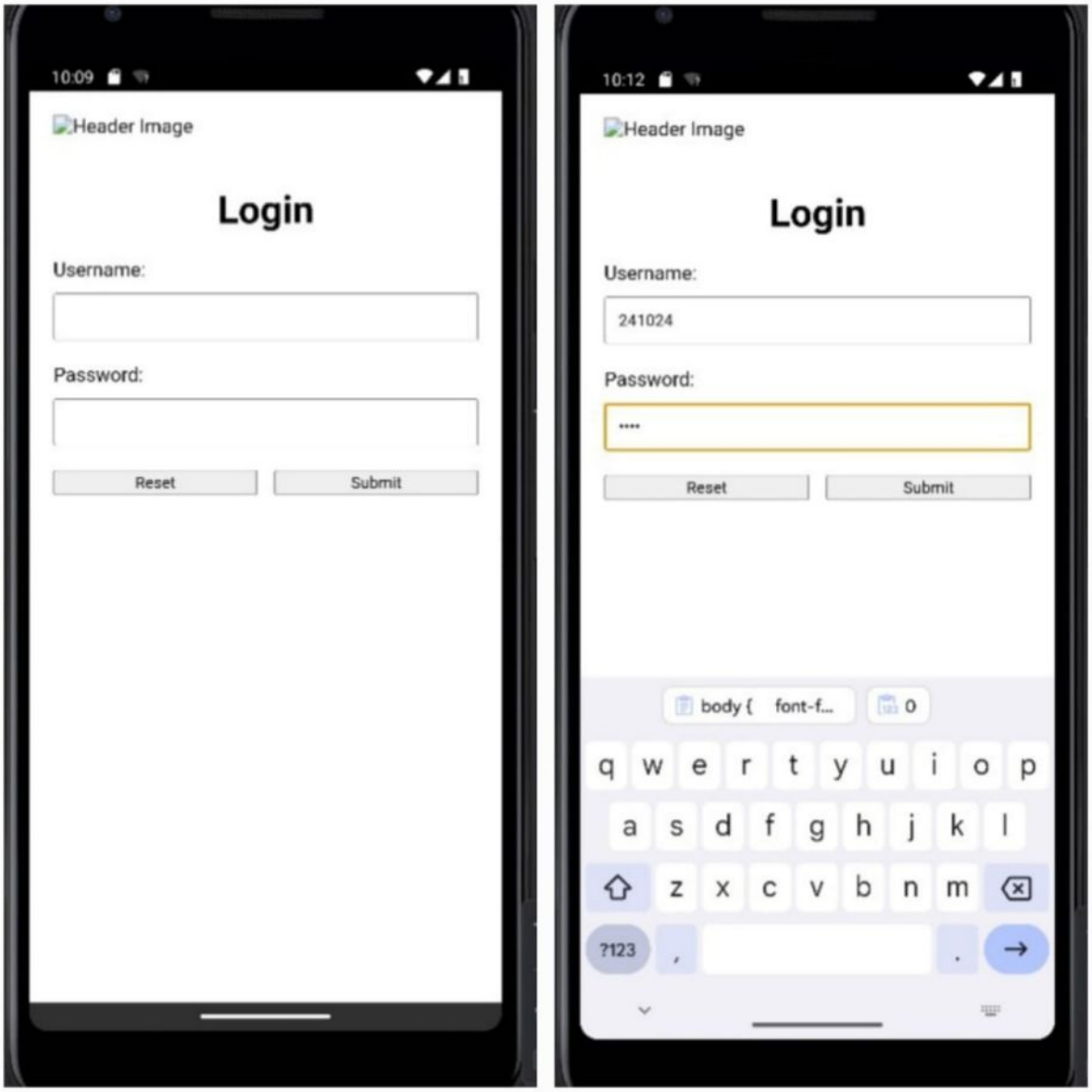
Style.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
}

header img {
    width: 100%;
    margin-bottom: 20px;
}

h1 {
    text-align: center;
}

.login-form label {
    display: block;
    margin-bottom: 10px;
}

.login-form input {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    box-sizing: border-box;
}

.buttons {
    display: flex;
    justify-content: space-between;
}

.buttons button {
    width: 48%;
}
```

Index.js:

```
document.addEventListener('deviceready', onDeviceReady, false);

function onDeviceReady() {
    document.getElementById('resetBtn').addEventListener('click', resetForm);
    document.getElementById('submitBtn').addEventListener('click',
submitForm);
}

function resetForm() {
    document.getElementById('username').value = '';
    document.getElementById('password').value = '';
}

function submitForm() {
    var username = document.getElementById('username').value;
    var password = document.getElementById('password').value;
}
```

Output:



Result:

Thus an android application using Cordova for the user login screen is executed and verified successfully.

Ex No:6 TO FIND AND DISPLAY THE CURRENT LOCATION OF THE USER USING CORDOVA

Aim:

To Design and develop an android application using Apache Cordova to find and display the current location of the user.

Algorithm:

Step1:Start the program.
Step 2: Create a new Cordova project using the following command:
"cordova create LoginApp com.example.LoginApp UserLogin"
Step 3: Open your command prompt or terminal and navigate to the directory where your Cordova
project is located.
Step 4: Add Android as a platform to your Cordova project using the following command:
"cordova platform add android"
Step 5: Open the 'index.html' file located in the 'www' directory of your Cordova project.
Step 6: Create a simple UI layout using HTML and CSS.
Step 7: Install the Cordova Geolocation plugin to access the device's GPS location.
'cordova plugin add cordova-plugin-geolocation.'
Step 8: Write JavaScript code to request the current location from the device's GPS.
Step 9: Use the Cordova Geolocation plugin to access the device's GPS coordinates.
Step 10: Display the retrieved location information on the UI or in the console.
Step 11: Connect your Android device or start an Android emulator. Then run:"cordova run android".
Step 12:Stop the Program.

Program:

Index.html

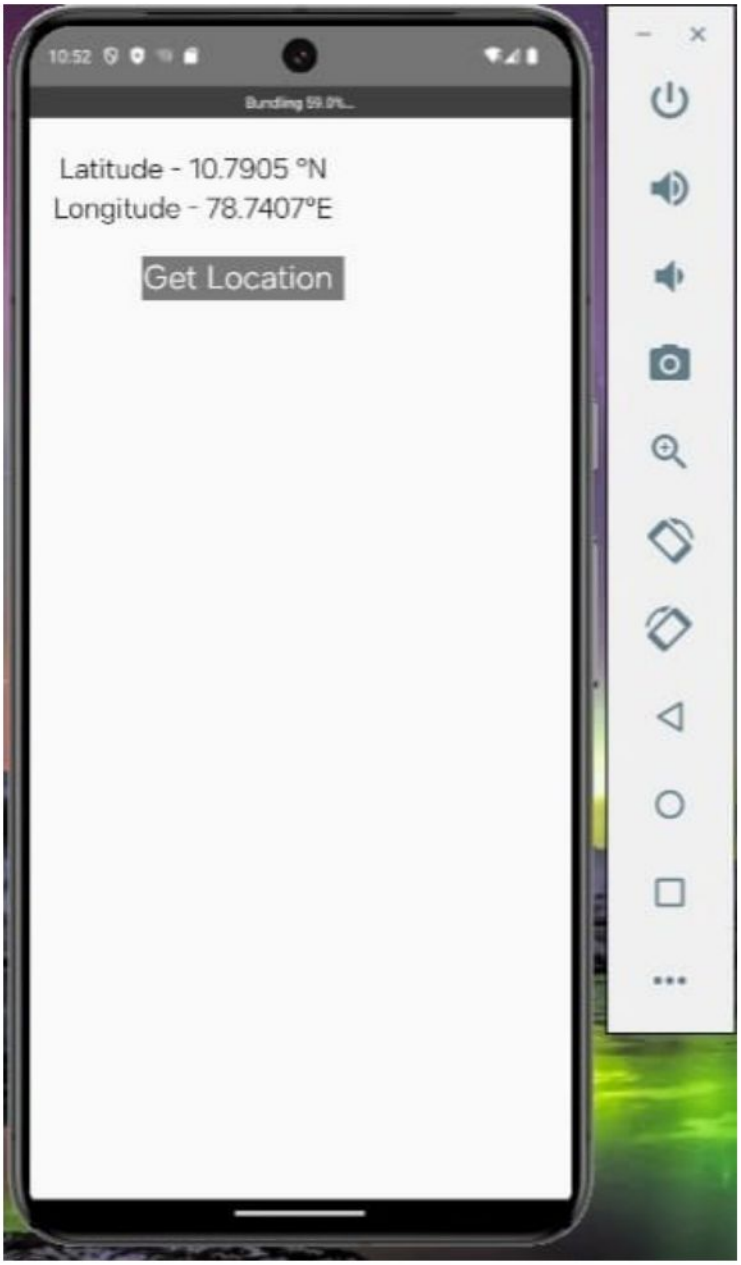
```
<!DOCTYPE html>
<html>
<head>
  <title>Location App</title>
```

```
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
</head>
<body>
    <div id="location"></div>
    <button onclick="getLocation()">Get Location</button>
</body>
</html>
```

Index.js:

```
function getLocation() {
    navigator.geolocation.getCurrentPosition(onSuccess, onError);
}

function onSuccess(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    document.getElementById('location').innerHTML = 'Latitude: ' + latitude +
'<br>Longitude: ' + longitude;
}

function onError(error) {
    alert('Error occurred: ' + error.message);
}
```

Output:



Result:

Thus the android application using apache cordova to find and display the current location of the user is executed and verified successfully.

Ex No:7 LIBRARY ACCESS MANAGEMENT APPLICATION USING JAVA

Aim:

To design and develop an Android application for library access using Java and a database to display available books, books lend, and book reservations.

Algorithm:

Step 1: Start the program

Step 2: Create a New Android Project:

Start by creating a new project in Android Studio.

Step 3: Set Up Firebase:

Set up a Firebase project in the Firebase console (https://console.firebase.google.com/) and add your Android app to it.

Step 4: Add Firebase SDK:

Add the necessary Firebase SDK dependencies to your Android project by including the Firebase SDK in your app-level build.gradle file.

Step 5: Design User Interface (UI):

Design the UI for your library access management system using Android XML layout files. Include components like EditTexts, Buttons, and TextViews to handle user input and display information.

Step 6: Implement User Authentication:

Utilize Firebase Authentication to allow users to sign up, sign in, and sign out of the app securely.

Step 7: Set Up Firestore Database:

Set up a Firestore database in the Firebase console to store information about library members, books, and borrowing history.

Step 8: Implement Data Models:

Create Java classes to represent library members, books, and borrowing history. These classes will be used to interact with Firestore.

Step 9: Implement Firestore CRUD Operations:

Implement methods to perform CRUD (Create, Read, Update, Delete) operations on Firestore collections using Firebase Firestore SDK.

Step 10: Implement Book Search:

Develop functionality to search for books by title, author, or category. Use Firestore queries to retrieve relevant information from the database.

Step 11: Implement Book Checkout:

Allow users to borrow books from the library. Update Firestore to reflect the borrowing status of the book and the borrower.

Step 12: Implement Book Return:

Enable users to return borrowed books. Update Firestore accordingly.

Step 13: Display User Borrowing History:

Retrieve and display the borrowing history of users from Firestore. Update the UI to present this information in a user-friendly format.

Step 14: Handle User Notifications:

Implement notifications to remind users of upcoming book due dates or overdue books using Firebase Cloud Messaging (FCM).

Step 15: Test and Debug:

Thoroughly test your application to ensure all features work as expected. Debug any issues that arise during testing.

Step 16: Deploy Your App:

Once testing is complete, deploy your app to the Google Play Store or distribute it through other channels.

Step 17: Stop the program

Program:

AdminHome.java

```
import android.app.AlertDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseApp;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;

public class AdminHome extends AppCompatActivity implements
View.OnClickListener {
```

```
    private Button
searchBook,addBook,removeBook,updateBook,issueBook,returnBook,logOut,coll
ect1,reissueButton;
    private FirebaseAuth firebaseAuth;
    private FirebaseFirestore db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_home);

        FirebaseApp.initializeApp(this);
        firebaseAuth=FirebaseAuth.getInstance();
        searchBook=(Button)findViewById(R.id.searchBook);
        addBook=(Button)findViewById(R.id.addBook);
        removeBook=(Button)findViewById(R.id.removeBook);
        collect1=(Button)findViewById(R.id.collect1);
        updateBook=(Button)findViewById(R.id.updateBook);
        issueBook=(Button)findViewById(R.id.issueBook);
        returnBook=(Button)findViewById(R.id.returnBook);
        logOut=(Button)findViewById(R.id.logOut);
        reissueButton=(Button)findViewById(R.id.reissueBook);
        db=FirebaseFirestore.getInstance();

        searchBook.setOnClickListener(this);
        addBook.setOnClickListener(this);
        removeBook.setOnClickListener(this);
        updateBook.setOnClickListener(this);
        issueBook.setOnClickListener(this);
        returnBook.setOnClickListener(this);
        logOut.setOnClickListener(this);
        collect1.setOnClickListener(this);
        reissueButton.setOnClickListener(this);

    }
    @Override
    public void onClick(View v) {
        if(v==logOut)
        {
```

```
db.document("User/"+firebaseAuth.getCurrentUser().getEmail()).update("fcmTo
ken",null).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
               if(task.isSuccessful())
               {
                  firebaseAuth.signOut();
                  startActivity(new
Intent(getApplicationContext(),SignInActivity.class));
                  finish();
               }
               else
               {
                  Toast.makeText(AdminHome.this, "Try Again !",
Toast.LENGTH_SHORT).show();
               }
            }
         });
      }
      if(v==searchBook)
      {
         startActivity(new Intent(getApplicationContext(),SearchBookSet.class));
      }
      if(v==addBook)
      {
         startActivity(new Intent(getApplicationContext(),AdminAddBook.class));
      }
      if(v==removeBook)
      {
         startActivity(new
Intent(getApplicationContext(),AdminRemoveBook.class));
      }
      if(v==issueBook)
      {
         startActivity(new
Intent(getApplicationContext(),AdminIssueBook.class));
      }
      if(v==returnBook)
```

```
        {
            startActivity(new
Intent(getApplicationContext(),AdminReturnBook.class));
        }
        if(v==updateBook)
        {
            startActivity(new
Intent(getApplicationContext(),AdminUpdateBook.class));
        }
        if(v==collect1)
        {
            startActivity(new
Intent(getApplicationContext(),AdminCollectFine.class));
        }
        if(v==reissueButton)
        {
            startActivity(new
Intent(getApplicationContext(),AdminReissueBook.class));
        }
    }
}
```

activity_adminhome.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="20dp"
    android:orientation="vertical"
    tools:context=".AdminHome">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_centerVertical="true"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">


            <TextView
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="ADMIN HOME"
    android:textSize="20dp"
    android:textColor="@color/colorPrimary"
    android:textAlignment="center"
    />

<Button
    android:layout_width="200dp"
    android:layout_marginTop="30dp"
    android:gravity="center"
    android:layout_gravity="center"
    android:layout_height="wrap_content"
    android:id="@+id/searchBook"
    android:text="Search"
    />

<Button
    android:layout_width="200dp"
    android:gravity="center"
    android:layout_gravity="center"
    android:layout_height="wrap_content"
    android:id="@+id/addBook"
    android:text="Add Book"
    />
<Button
    android:layout_width="200dp"
    android:gravity="center"
    android:layout_gravity="center"
    android:layout_height="wrap_content"
    android:id="@+id/removeBook"
    android:text="Remove Book"
    />
<Button
    android:layout_width="200dp"
    android:gravity="center"
    android:layout_gravity="center"
    android:layout_height="wrap_content"
    android:id="@+id/updateBook"
    android:text="Update"
    />
<Button
    android:layout_width="200dp"
    android:gravity="center"
    android:layout_gravity="center"
    android:layout_height="wrap_content"
    android:id="@+id/issueBook"
    android:text="Issue Book to User"
    />
<Button
    android:layout_width="200dp"
    android:gravity="center"
```

```
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:id="@+id/returnBook"
            android:text="Return Book from User"
            />

        <Button
            android:layout_width="200dp"
            android:gravity="center"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:id="@+id/reissueBook"
            android:text="Re-Issue Book"
            />

        <Button
            android:layout_width="200dp"
            android:gravity="center"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:id="@+id/collect1"
            android:text="Collect Fine"
            />

        <Button
            android:layout_width="200dp"
            android:gravity="center"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:id="@+id/logOut"
            android:text="Log Out"
            />

    </LinearLayout>

  </ScrollView>

</RelativeLayout>
```

SignUp.java

```
package com.iiitnr.libraryapp;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.widget.TextInputEditText;
import android.support.design.widget.TextInputLayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
```

```
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseApp;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthUserCollisionException;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class SignUpActivity extends AppCompatActivity implements
View.OnClickListener{

    private TextInputLayout editName;
    private TextInputLayout editEnrollNo;
    private TextInputLayout editCardNo;
    private TextInputLayout editPass1;
    private TextInputLayout editID;
    private TextInputLayout editPass;
    private Button buttonRegister;
    private TextView toSignIn;
    private ProgressDialog progressDialog;
    private FirebaseAuth firebaseAuth;
    private CheckBox check1;
    private FirebaseFirestore db;
    private Spinner userType;
    private String type;
    private int type1;
    private int temp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);
```

```
        editID = (TextInputLayout) findViewById(R.id.editID);
        editPass = (TextInputLayout) findViewById(R.id.editPass);
        editPass1=(TextInputLayout)findViewById(R.id.editPass1);
        editName=(TextInputLayout)findViewById(R.id.editName);
        editEnrollNo=(TextInputLayout)findViewById(R.id.editEnrollNo);
        editCardNo=(TextInputLayout)findViewById(R.id.editCardNo);
        buttonRegister = (Button) findViewById(R.id.buttonRegister);
        toSignIn = (TextView) findViewById(R.id.toSignIn);
        progressDialog=new ProgressDialog(this);
        progressDialog.setCancelable(false);
        check1=(CheckBox)findViewById(R.id.check1);
        userType=(Spinner)findViewById(R.id.userType);

        List<String> list = new ArrayList<>();
        list.add("Select Account Type");
        list.add("User");
        list.add("Admin");

        ArrayAdapter adapter =new
ArrayAdapter(this,android.R.layout.simple_spinner_dropdown_item,list);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
        userType.setAdapter(adapter);
        userType.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {

                if(parent.getItemAtPosition(position).toString().equals("Select
Account Type"))
                {
                    type=parent.getItemAtPosition(position).toString();
                    editPass1.setEnabled(false);
                    editPass.setEnabled(false);
                    editName.setEnabled(false);
                    editID.setEnabled(false);
                    editEnrollNo.setEnabled(false);
                    editCardNo.setEnabled(false);

                    editCardNo.setErrorEnabled(false);
                    editEnrollNo.setErrorEnabled(false);
                    editID.setErrorEnabled(false);
                    editName.setErrorEnabled(false);
                    editPass.setErrorEnabled(false);
                    editPass1.setErrorEnabled(false);

                }
                else if(parent.getItemAtPosition(position).toString().equals("User"))
                {
                    type=parent.getItemAtPosition(position).toString();
```

```
            editPass1.setEnabled(true);
            editPass.setEnabled(true);
            editName.setEnabled(true);
            editID.setEnabled(true);
            editEnrollNo.setEnabled(true);
            editCardNo.setEnabled(true);

            editCardNo.setErrorEnabled(false);
            editEnrollNo.setErrorEnabled(false);
            editID.setErrorEnabled(false);
            editName.setErrorEnabled(false);
            editPass.setErrorEnabled(false);
            editPass1.setErrorEnabled(false);
        }
        else
        {
            type=parent.getItemAtPosition(position).toString();
            editPass1.setEnabled(true);
            editPass.setEnabled(true);
            editName.setEnabled(true);
            editID.setEnabled(true);
            editEnrollNo.setEnabled(false);
            editCardNo.setEnabled(false);

            editCardNo.setErrorEnabled(false);
            editEnrollNo.setErrorEnabled(false);
            editID.setErrorEnabled(false);
            editName.setErrorEnabled(false);
            editPass.setErrorEnabled(false);
            editPass1.setErrorEnabled(false);
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
FirebaseApp.initializeApp(this);
firebaseAuth=FirebaseAuth.getInstance();
db=FirebaseFirestore.getInstance();
buttonRegister.setOnClickListener(this);
toSignIn.setOnClickListener(this);
check1.setOnClickListener(this);

}

private boolean verifyName()
{

    String name=editName.getEditText().getText().toString().trim();
    if(name.isEmpty())
    {   editName.setErrorEnabled(true);
```

```
        editName.setError("Name Required");
        return true;
    }
    else
    {
        editName.setErrorEnabled(false);
        return false;
    }
}

private boolean verifyCardNo()
{
    String cardNo=editCardNo.getEditText().getText().toString().trim();
    if(cardNo.isEmpty())
    {   editCardNo.setErrorEnabled(true);
        editCardNo.setError("Card No. Required");
        return true;
    }
    else
    {
        editCardNo.setErrorEnabled(false);
        return false;
    }
}

private boolean verifyEnrollNo()
{
    String enrollNo=editEnrollNo.getEditText().getText().toString().trim();
    if(enrollNo.isEmpty())
    {   editEnrollNo.setErrorEnabled(true);
        editEnrollNo.setError("Enrollment No. Required");
        return true;
    }
    else
    {
        editEnrollNo.setErrorEnabled(false);
        return false;
    }
}

private boolean verifyEmailId()
{
    String emailId=editID.getEditText().getText().toString().trim();
    if(emailId.isEmpty())
    {   editID.setErrorEnabled(true);
        editID.setError(" Email ID Required");
        return true;
    }
    else if(!emailId.endsWith("@iiitnr.edu.in"))
    {
        editID.setErrorEnabled(true);
        editID.setError(" Enter Valid Email ID");
        return true;
```

```
        }
        else
        {
            editID.setErrorEnabled(false);
            return false;
        }
    }

    private boolean verifyPass()
    {
        String pass=editPass.getEditText().getText().toString().trim();
        if(pass.isEmpty())
        {   editPass.setErrorEnabled(true);
            editPass.setError(" Password Required");
            return true;
        }
        else
        {
            editPass.setErrorEnabled(false);
            return false;
        }
    }

    private boolean verifyPass1()
    {
        String pass1=editPass1.getEditText().getText().toString().trim();
        String pass=editPass.getEditText().getText().toString().trim();
        if(pass1.isEmpty())
        {   editPass1.setErrorEnabled(true);
            editPass1.setError("Confirm Password Required");
            return true;
        }
        else if(pass.equals(pass1))
        {
            editPass1.setErrorEnabled(false);
            return false;
        }
        else
        {
            editPass1.setErrorEnabled(true);
            editPass1.setError("Passwords do not match");
            return true;
        }
    }

   private boolean verifyType()
    {
      if (type.equals("Select Account Type"))
      {
          Toast.makeText(this, "Please select account type !",
Toast.LENGTH_SHORT).show();
          return true;
      }
```

```
      return false;
   }

   private boolean verifyCard1()
   {

db.collection("User").whereEqualTo("card",Integer.parseInt(editCardNo.getEditT
ext().getText().toString().trim())).get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
         @Override
         public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if(task.isSuccessful())
            {
              temp=task.getResult().size();
            }
         }
      });
      if(temp==0)
         return false;
      else
         return true;
   }

   private void registerUser()
   {

      if(verifyType())
        return;

      if(type.equals("User"))
      {
         boolean res= (verifyName()|verifyCardNo()|verifyEmailId()|
verifyEnrollNo()|verifyPass()|verifyPass1());
         if(res==true)
            return;
      }
      if(type.equals("Admin"))
      {
         boolean res= (verifyName()|verifyEmailId()|verifyPass()|verifyPass1());
         if(res==true)
            return;
      }

      String id=editID.getEditText().getText().toString().trim();
      String pass=editPass.getEditText().getText().toString().trim();
      if(type.equals("User")){
         type1=0;
      }
      else
         type1=1;

      progressDialog.setMessage("Registering User ... ");
      progressDialog.show();
```

```
firebaseAuth.createUserWithEmailAndPassword(id,pass).addOnCompleteListener(
SignUpActivity.this, new OnCompleteListener<AuthResult>()
        {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful())
                {
                    String id=editID.getEditText().getText().toString().trim();
                    String name=editName.getEditText().getText().toString().trim();
                    if(type1==0)
                    {

                        int
enroll=Integer.parseInt(editEnrollNo.getEditText().getText().toString().trim());
                        int
card=Integer.parseInt(editCardNo.getEditText().getText().toString().trim());
                        db.collection("User").document(id).set(new
User(name,id,enroll,card,type1)).addOnSuccessListener(new
OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                progressDialog.cancel();
                                Toast.makeText(SignUpActivity.this,"Registered
Successfully !",Toast.LENGTH_SHORT).show();
                                firebaseAuth.signOut();
                                startActivity(new
Intent(getApplicationContext(),SignInActivity.class));
                                finish();
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Toast.makeText(SignUpActivity.this, "Please Try Again !",
Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                    else {

                        db.collection("User").document(id).set(new
Admin(type1,name,id)).addOnSuccessListener(new OnSuccessListener<Void>()
{
                            @Override
                            public void onSuccess(Void aVoid) {
                                progressDialog.cancel();
                                Toast.makeText(SignUpActivity.this,"Registered
Successfully !",Toast.LENGTH_SHORT).show();
                                firebaseAuth.signOut();
                                startActivity(new
Intent(getApplicationContext(),SignInActivity.class));
                                finish();
```

```
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(SignUpActivity.this, "Please Try Again !",
Toast.LENGTH_SHORT).show();
                    }
                });
            }

        }

        else
        {   progressDialog.cancel();
        if(task.getException() instanceof FirebaseAuthUserCollisionException)
        {
            Toast.makeText(SignUpActivity.this,"Already Registered !
",Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(SignUpActivity.this, "Registration Failed ! Try
Again ", Toast.LENGTH_SHORT).show();
        }

        }
    }
});
}

@Override
public void onClick(View v) {

    if(v==check1)
    {
        if(check1.isChecked())
            buttonRegister.setEnabled(true);
        else
            buttonRegister.setEnabled(false);
    }
    else if(v==buttonRegister)
        registerUser();

    else if(v==toSignIn) {
        startActivity(new Intent(getApplicationContext(),SignInActivity.class));
        finish();
    }
}
}
```

activity_signup.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"
xmlns:android="http://schemas.android.com/apk/res/android"
tools:context=".SignUpActivity"
android:orientation="vertical" android:layout_height="wrap_content"
android:layout_width="match_parent"
android:layout_margin="20dp"
>
 <ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:id="@+id/scroll">

    <LinearLayout
       android:layout_width="match_parent"
       android:gravity="center"
       android:orientation="vertical"
       android:layout_height="wrap_content">

       <TextView
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:text="USER REGISTRATION"
          android:textSize="20dp"
          android:id="@+id/head"
          android:gravity="center"
          android:textColor="@color/colorPrimary"/>

       <Spinner
          android:layout_marginTop="30dp"
          android:id="@+id/userType"
          android:layout_width="match_parent"
          android:layout_height="30dp"
          ></Spinner>

       <android.support.design.widget.TextInputLayout
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:layout_marginTop="10dp"
```

```
    android:id="@+id/editName">

    <android.support.design.widget.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Name"
        android:inputType="text" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:id="@+id/editEnrollNo">

    <android.support.design.widget.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enrollment No."
        android:inputType="number" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:id="@+id/editCardNo">

    <android.support.design.widget.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Library Card No."
        android:inputType="number" />

</android.support.design.widget.TextInputLayout>
```

```
<android.support.design.widget.TextInputLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_marginTop="10dp"
   android:id="@+id/editID">

   <android.support.design.widget.TextInputEditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Email ID"
      android:inputType="textEmailAddress" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:id="@+id/editPass"
   android:layout_marginTop="10dp"
   app:passwordToggleEnabled="true">

   <android.support.design.widget.TextInputEditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Password"
      android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:id="@+id/editPass1"
   android:layout_marginTop="10dp"
   app:passwordToggleEnabled="true">

   <android.support.design.widget.TextInputEditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
```

```
            android:hint="Confirm Password"
            android:inputType="textPassword" />

        </android.support.design.widget.TextInputLayout>

        <CheckBox
            android:id="@+id/check1"
            android:layout_marginTop="10dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="I Agree that the information provided is correct"/>
        <Button
            android:id="@+id/buttonRegister"
            android:enabled="false"
            android:layout_marginTop="10dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Register"
            android:textColor="#000000"/>

        <TextView
            android:id="@+id/toSignIn"
            android:layout_width="match_parent"
            android:layout_marginTop="10dp"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:textSize="15dp"
            android:textColor="@color/colorPrimary"
            android:text="Already Registered ? Sign in " />

</LinearLayout>

 </ScrollView>

</RelativeLayout>
```

SearchBook.java

```
package com.iiitnr.libraryapp;

import android.app.ProgressDialog;
```

```
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.firebase.ui.firestore.FirestoreRecyclerAdapter;
import com.firebase.ui.firestore.FirestoreRecyclerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseApp;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;

public class SearchBook extends AppCompatActivity {

    private FirebaseFirestore db;
    private FirestoreRecyclerAdapter adapter;
    private int mode=0;
    private String key="";
    ProgressDialog progressDialog;
    private TextView ifNoBook;
    Query query;

    @Override
    public void onBackPressed() {
        finish();
        super.onBackPressed();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_book);
        ifNoBook=(TextView)findViewById(R.id.ifNoBook);
        progressDialog=new ProgressDialog(this);
        progressDialog.setMessage("Please Wait !");
        FirebaseApp.initializeApp(this);
        db=FirebaseFirestore.getInstance();
        progressDialog.show();
        Intent intent=getIntent();
        switch(intent.getIntExtra("id",0))
        {
           case 1:{
              mode=0;
query=db.collection("Book").whereEqualTo("id",intent.getIntExtra("bid",0)).wher
eGreaterThan("available",0);
              break;
           }
           case 2:{
              mode=0;
      query=db.collection("Book").whereEqualTo("id",intent.getIntExtra("bid",0));
              break;
           }
           case 3:{
              mode=1;
              key=intent.getStringExtra("btitle");
query=db.collection("Book").whereEqualTo("type",intent.getStringExtra("btype")
).whereGreaterThan("available",0);
              break;
           }
           case 4:{
              mode=1;
              key=intent.getStringExtra("btitle");
query=db.collection("Book").whereEqualTo("type",intent.getStringExtra("btype")
);
              break;
           }
```

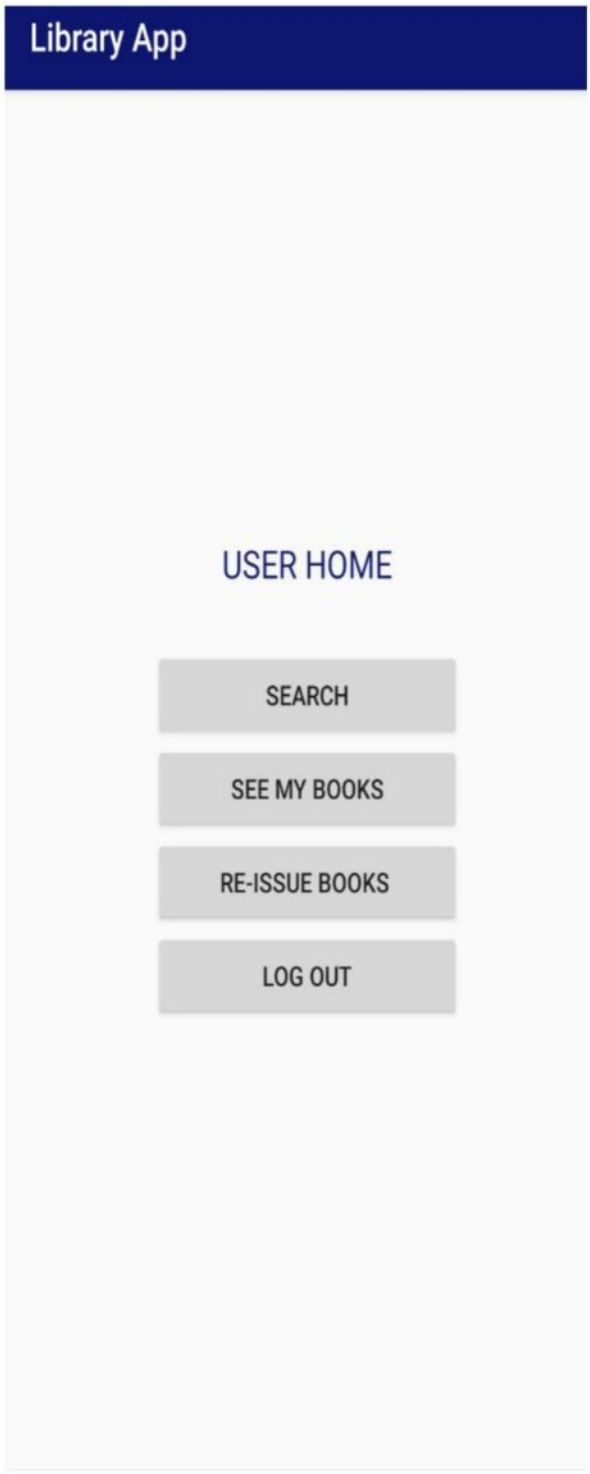
```
            case 5:{
                mode=1;
                key=intent.getStringExtra("btitle");
                query=db.collection("Book").whereGreaterThan("available",0);
                break;
            }
            case 6:{
                mode=1;
                key=intent.getStringExtra("btitle");
                query=db.collection("Book");
                break;
            }
            case 7:{
                mode=0;
query=db.collection("Book").whereEqualTo("type",intent.getStringExtra("btype")
).whereGreaterThan("available",0);
                break;
            }
            case 8:{
                mode=0;
query=db.collection("Book").whereEqualTo("type",intent.getStringExtra("btype")
);
                break;
            }
        }
        //progressDialog.show();
        /*query.get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {

                if(task.isSuccessful())
                {
                    if(task.getResult().size()==0)
                    {
                        ifNoBook.setTextSize(25);
                        ifNoBook.setText("NO SUCH BOOK !");
                    }
                }
```

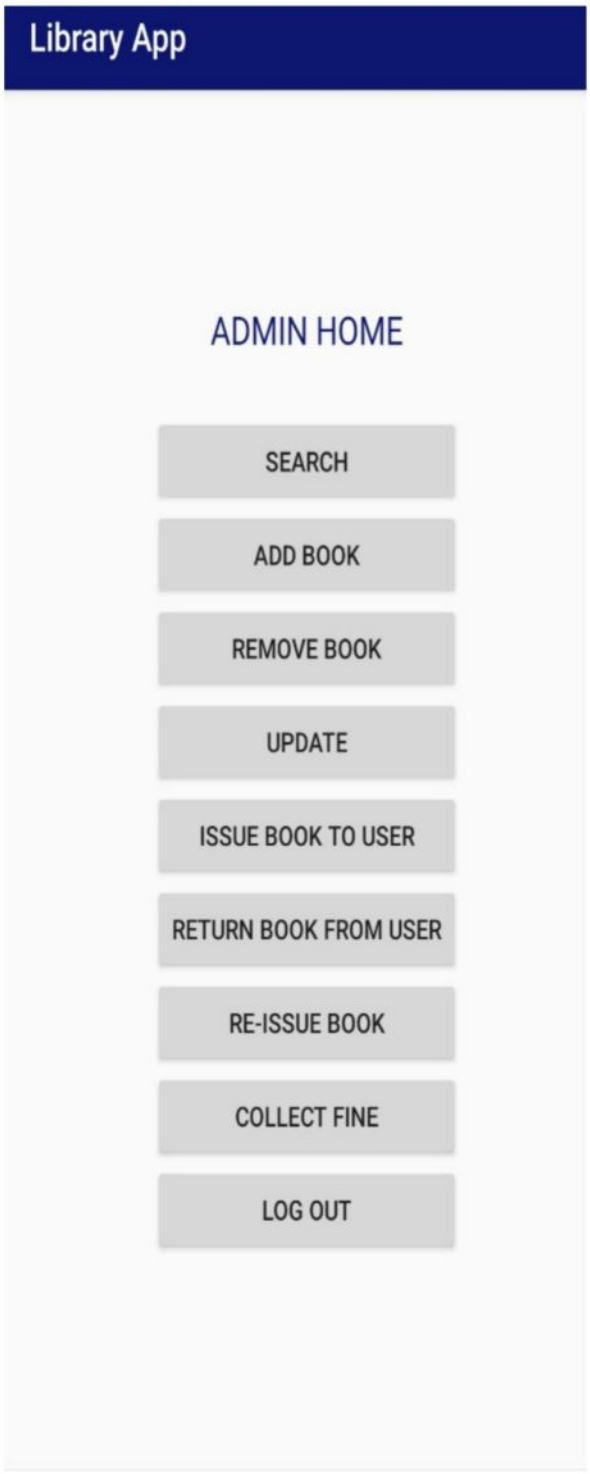
```
            else
            {
               //progressDialog.cancel();
               Toast.makeText(SearchBook.this, "Something went wrong !\nTry
Again", Toast.LENGTH_SHORT).show();
            }
         }
      });
      */

         FirestoreRecyclerOptions options = new
FirestoreRecyclerOptions.Builder<Book>().setQuery(query, Book.class).build();
         adapter = new BookAdapter(options, key.toUpperCase(), mode);
         RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycle);
         recyclerView.setLayoutManager(new LinearLayoutManager(this));
         recyclerView.setAdapter(adapter);
      adapter.startListening();
         progressDialog.cancel();

   }

   @Override
   protected void onStart() {
      super.onStart();
      adapter.startListening();
   }

   @Override
   protected void onStop() {
      super.onStop();
      adapter.stopListening();
   }
}
```

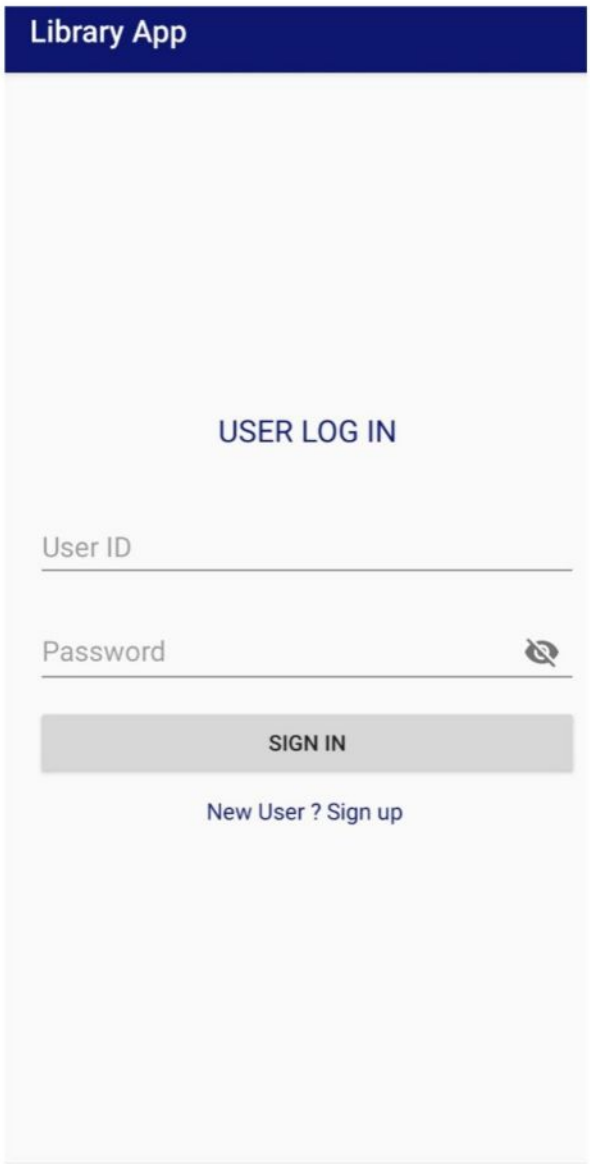
activity_searchbook.xml

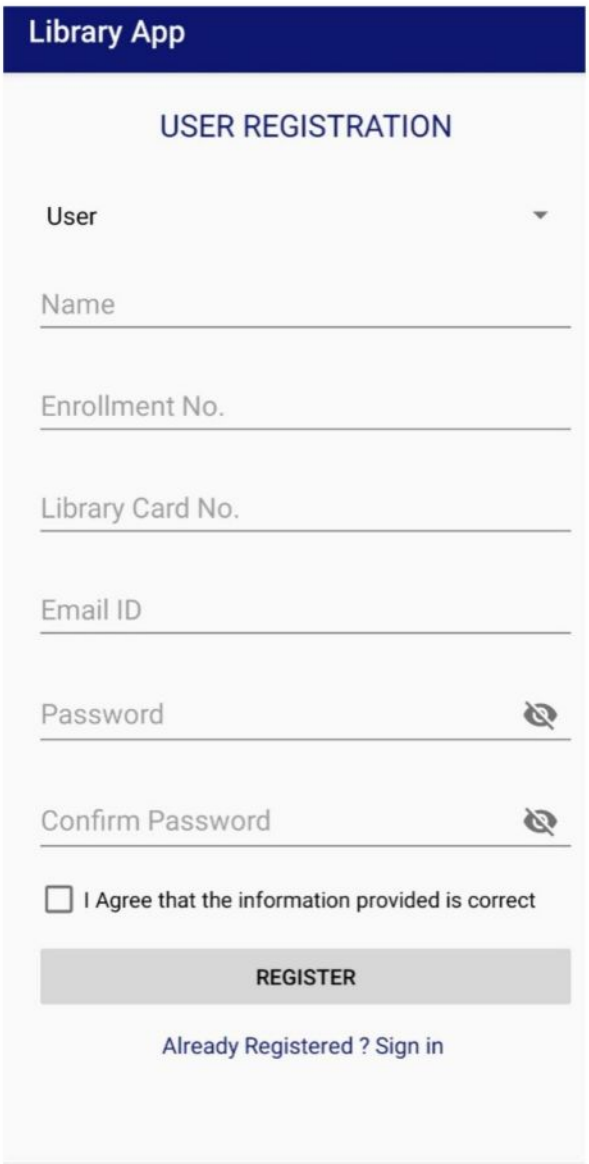
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SearchBook">

<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:textSize="18dp"
    android:text="No Such Book !"
    android:textAlignment="center"

    android:layout_marginTop="0.2in"
    android:id="@+id/ifNoBook"
    android:textColor="#F10A0A"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recycle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

    </android.support.v7.widget.RecyclerView>

</android.support.design.widget.CoordinatorLayout>
```

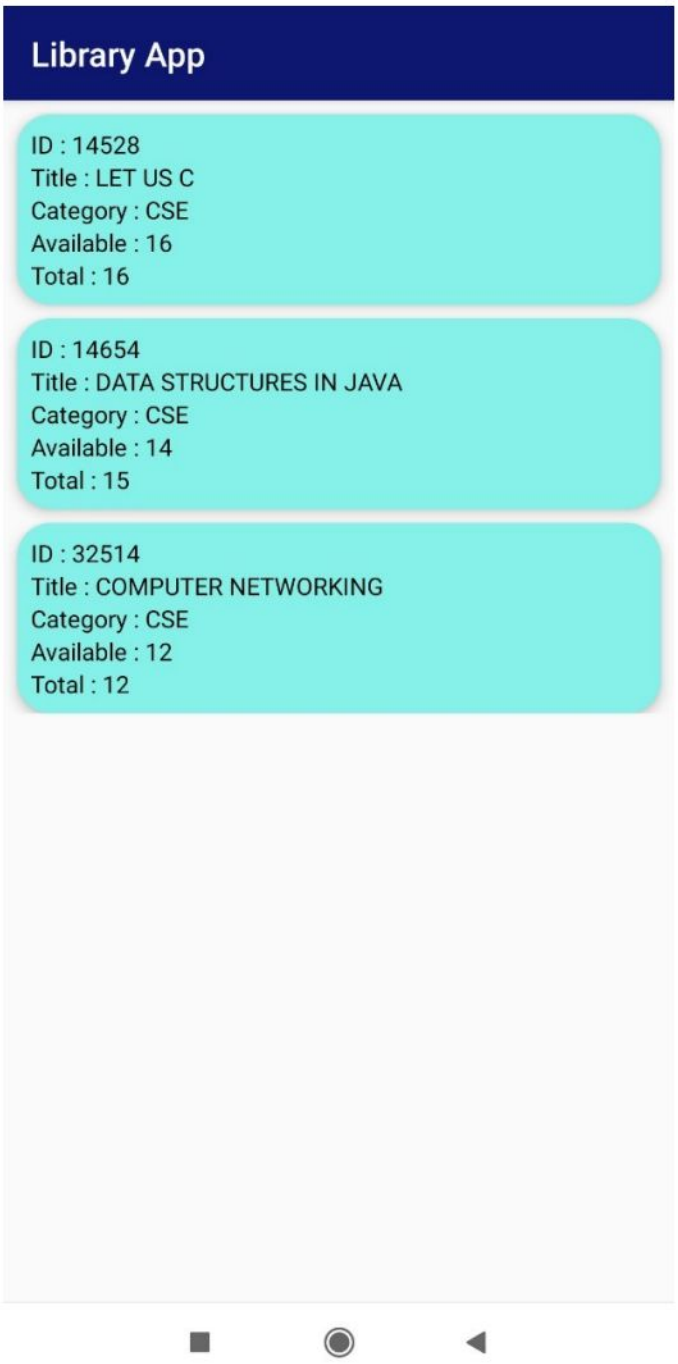
Output:


Library App
USER HOME
SEARCH
SEE MY BOOKS
RE-ISSUE BOOKS
LOG OUT


Library App
ADMIN HOME
SEARCH
ADD BOOK
REMOVE BOOK
UPDATE
ISSUE BOOK TO USER
RETURN BOOK FROM USER
RE-ISSUE BOOK
COLLECT FINE
LOG OUT


Library App
USER LOG IN
User ID
Password
SIGN IN
New User ? Sign up


Library App
USER REGISTRATION
User
Name
Enrollment No.
Library Card No.
Email ID
Password
Confirm Password
I Agree that the information provided is correct
REGISTER
Already Registered ? Sign in



Result:

Thus the design and develop an Android application for library access using Java and a database to display available books, books lend, and book reservations is executed and verified successfully.