AD lab 3rd

app development (Anna University)

Scan to open on Studocu

Studocu is not sponsored or endorsed by any college or university
Downloaded by Sona Kannan (sona29102003@gmail.com)

JAI SHRIRAM
ENGINEERING COLLEGE

(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai,
Accredited byNAAC, NBA Accredited for ECE & CSE)
Dharapuram Road, Avinashipalayam, Tirupur – 638 660.

Academic Year 2023-2024 (Odd Semester)

LABORATORY RECORD

Certified that this is a bonafide record of work done by

Name	…………………………………………………………………..
Reg. No.	…………………………………………………………………..
Branch	…………………………………………………………………..
Year & Semester	…………………………………………………………………..
Course code & Name	…………………………………………………………………..

Course In-Charge Head of the Department

Submitted for the University Practical Examination held on ………………..

Internal Examiner External Examiner

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

INDEX

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 1 **Date:**	**BMI CALCULATOR (Body Mass Index)**

AIM

Build a BMI (Body Mass Index) Calculator app using React Native.

ALGORITHM

Step 1: Start
Step 2: Import the required components from React Native and install the npm module.
Step 3: Create a class-based component named BMI App.
Step 4: The initial state with height, weight, bmi, and bmi Result fields.
Step 5: Define methods handle Height and handle Weight to update the state when height and weight inputs change.
Step 6: Create a calculate method to perform the BMI calculation and classify the result.
Step 7: Implement the render method with necessary UI components.
Step 8: Use Text Input for user input of height and weight.
Step 9: Utilize Touchable Opacity for the Calculate button.
Step 10: Display the calculated BMI and its classification as output.

PROGRAM

1.App.js

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
constructor(props) {
   super(props);
   this.state = { name: 'Guest', weight: 90, height: 180, bmi: 27, message: '', optimalweight: '', time: new
Date().toLocaleTimeString() };
   this.submitMe = this.submitMe.bind(this);
   this.heightchange = this.heightchange.bind(this);
   this.weightchange = this.weightchange.bind(this);
   this.change = this.change.bind(this);
   this.ticker = this.ticker.bind(this);
   this.blur = this.blur.bind(this);
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
   this.calculateBMI = this.calculateBMI.bind(this);
}

heightchange(e){
  this.setState({height: e.target.value});
  e.preventDefault();
}

blur(e){
 this.calculateBMI();
}
 weightchange(e){
  this.setState({weight: e.target.value});
  e.preventDefault();
}

calculateBMI(){

    var heightSquared = (this.state.height/100  * this.state.height/100);
    var bmi = this.state.weight / heightSquared;
    var low = Math.round(18.5 * heightSquared);
    var high = Math.round(24.99 * heightSquared);
    var message = "";
    if( bmi >= 18.5  && bmi <= 24.99 ){
        message = "You are in a healthy weight range";
    }
    else if(bmi >= 25 && bmi <= 29.9){
      message = "You are overweight";
    }
    else if(bmi >= 30){
        message ="You are obese";
```

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
    }
    else if(bmi < 18.5){
      message = "You are under weight";
    }
    this.setState({message: message});
    this.setState({optimalweight: "Your suggested weight range is between "+low+ " - "+high});
    this.setState({bmi: Math.round(bmi * 100) / 100});

}

submitMe(e) {
   e.preventDefault();
   this.calculateBMI();
}

ticker() {
  this.setState({time: new Date().toLocaleTimeString()})
}

componentDidMount(){
  setInterval(this.ticker, 60000);
}

change(e){
  e.preventDefault();
  console.log(e.target);
  this.setState({name: e.target.value});
}

render() {
  return (
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
      <div className="App">
        <div className="App-header">
          <h2>BMI Calculator</h2>
        </div>
          <form onSubmit={this.submitMe}>
            <label>
              Please enter your name
            </label>
            <input type="text" name="name" value={this.state.name} onBlur={this.blur}
onChange={this.change}   />
             <label>
             Enter your height in cm:
            </label>
            <input type="text" name="height" value={this.state.height} onBlur={this.blur}
onChange={this.heightchange}   />
             <label>
             Enter your weight in kg :
            </label>
            <input type="text" name="weight" value={this.state.weight} onChange={this.weightchange}
/>
            <label>{this.state.checked} Hello {this.state.name}, How are you my friend? It's currently
{this.state.time} where you are living. Your BMI is {this.state.bmi} </label>
              <label>{this.state.message}</label>
              <label>{this.state.optimalweight}</label>

            <input type="submit" value="Submit"/>
          </form>

      </div>
    );
  }
```

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
}
export default App;
```

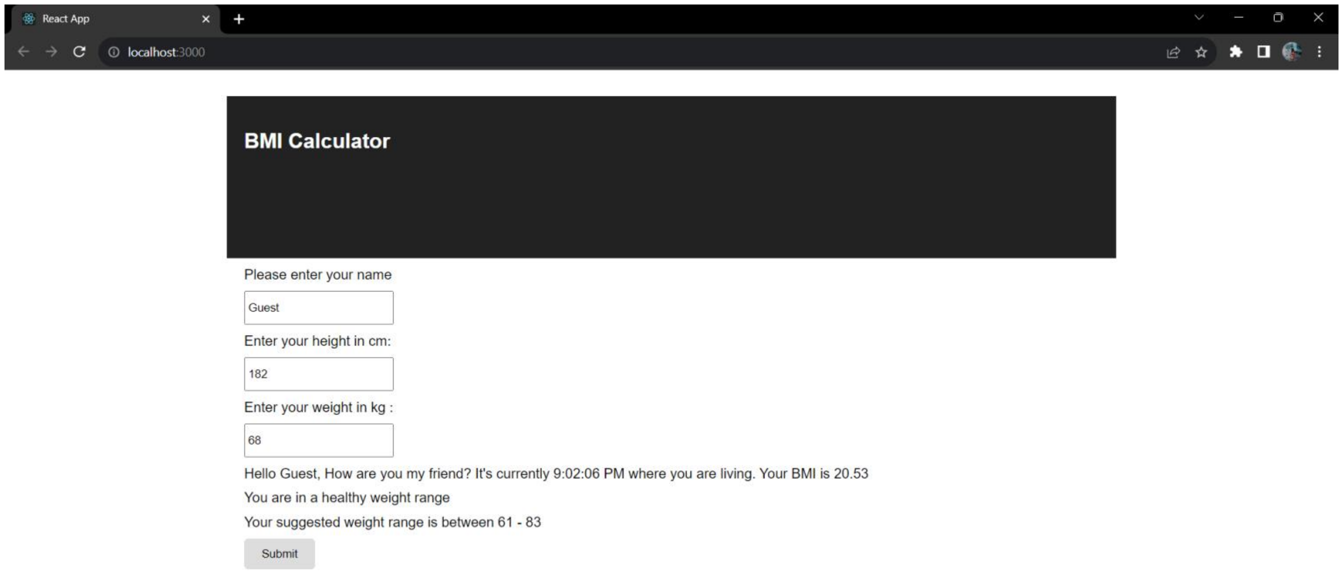
2. //index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>BMI Calculator with JavaScript</title>
   <link rel="stylesheet" href="style.css">
   <script src="script.js" defer></script>
</head>
<body>
   <div class="wrapper">
      <p>Height in CM:
         <input type="number" id="height"><br><span id="height_error"></span>
      </p>

      <p>Weight in KG:
         <input type="number" id="weight"><br><span id="weight_error"></span>
      </p>

      <button id="btn">Calculate</button>
      <p id="output"></p>
   </div>
</body>
</html>
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

SAMPLE OUTPUT



BMI Calculator

Please enter your name

Guest

Enter your height in cm:

182

Enter your weight in kg :

68

Hello Guest, How are you my friend? It's currently 9:02:06 PM where you are living. Your BMI is 20.53

You are in a healthy weight range

Your suggested weight range is between 61 - 83

RESULT

Thus the program to work with react native was implemented and executed successfully.

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 2 **Date:**	**SIMPLE EXPENSE MANAGER**

AIM:

To Build a cross platform application for simple expense manager which allows entering expenses and income

ALGORITHM

Step 1:Initialize state variables for income, expense, incomeList, expenseList, textIncome, and textExpense.
Step 2:Create functions addIncome and addExpense to handle adding income and expenses.
Step 3:In addIncome and addExpense, check if textIncome or textExpense is empty or not a number. If so, show an alert and exit.
Step 4:If input is valid, add a new item to incomeList or expenseList with a unique ID and the provided income or expense value.
Step 5:Update the income and expense state by adding the entered value to their respective totals.
Step 6:Clear the textIncome or textExpense field.
Step 7:Render the UI with title, input fields, buttons, and two lists.
Step 8:Map through incomeList and expenseList to display their items in separate lists.
Step 9:Style the UI elements using predefined styles.
Step 10:The app allows users to add income and expenses, which are displayed in separate lists for weekly tracking.

PROGRAM

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

export default function App() {
  const [income, setIncome] = useState(0);
  const [expense, setExpense] = useState(0);
  const [incomeList, setIncomeList] = useState([]);
  const [expenseList, setExpenseList] = useState([]);
  const [textIncome, setTextIncome] = useState('');
  const [textExpense, setTextExpense] = useState('');

  const addIncome = () => {
    if (textIncome === '' || income === '') {
      alert('Please enter the details');
    } else {
      setIncomeList([...incomeList, { id: Math.random().toString(), income: income }]);
      setIncome(parseInt(income) + parseInt(income));
      setTextIncome('');
    }
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
};

const addExpense = () => {
  if (textExpense === '' || expense === '') {
    alert('Please enter the details');
  } else {
    setExpenseList([...expenseList, { id: Math.random().toString(), expense: expense }]);
    setExpense(parseInt(expense) + parseInt(expense));
    setTextExpense('');
  }
};

return (
  <View style={styles.container}>
    <Text style={styles.title}>Expense Manager</Text>
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.input}
        placeholder="Enter Income"
        keyboardType="numeric"
        value={textIncome}
        onChangeText={(text) => setIncome(text)}
      />
      <TouchableOpacity style={styles.button} onPress={() => addIncome()}>
        <Text style={styles.buttonText}>Add Income</Text>
      </TouchableOpacity>
      <TextInput
        style={styles.input}
        placeholder="Enter Expense"
        keyboardType="numeric"
        value={textExpense}
        onChangeText={(text) => setExpense(text)}
      />
      <TouchableOpacity style={styles.button} onPress={() => addExpense()}>
        <Text style={styles.buttonText}>Add Expense</Text>
      </TouchableOpacity>
    </View>
    <View style={styles.listContainer}>
      <View style={styles.list}>
        <Text style={styles.listTitle}>Weekly Income</Text>
        {incomeList.map((item) => (
          <View key={item.id} style={styles.listItem}>
            <Text>{item.income}</Text>
          </View>
        ))}
      </View>
```

Downloaded by Sona Kannan (sona29102003@gmail.com)

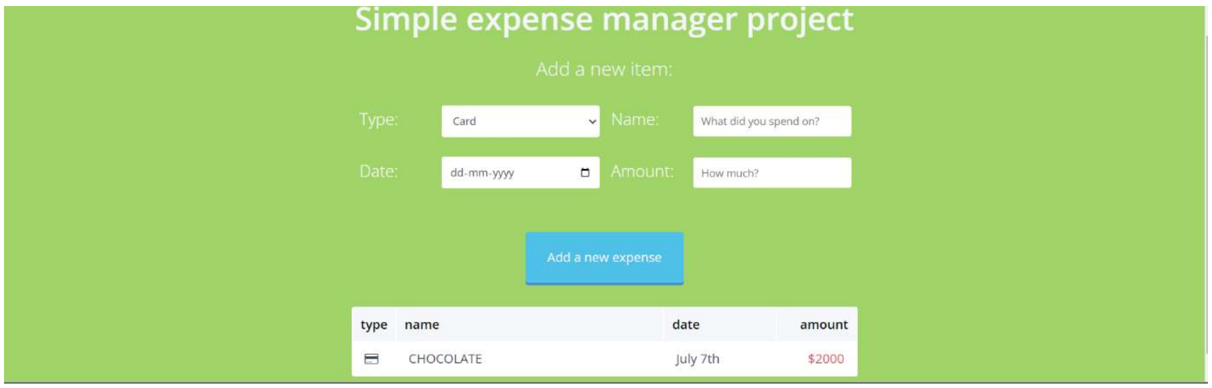
```
        <View style={styles.list}>
          <Text style={styles.listTitle}>Weekly Expense</Text>
          {expenseList.map((item) => (
            <View key={item.id} style={styles.listItem}>
              <Text>{item.expense}</Text>
            </View>
          ))}
        </View>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 30,
    fontWeight: 'bold',
    marginBottom: 20,
  },
  inputContainer: {
    width: '80%',
    marginBottom: 20,
  },
  input: {
    borderWidth: 1,
    borderColor: '#777',
    padding: 8,
    marginVertical: 10,
    borderRadius: 5,
  },
  button: {
    backgroundColor: '#f4511e',
    paddingVertical: 10,
    paddingHorizontal: 20,
    borderRadius: 5,
    alignItems: 'center',
    marginVertical: 10,
  },
  buttonText: {
    color: '#fff',
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
    fontSize: 18,
    fontWeight: 'bold',
  },
  listContainer: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    width: '80%',
  },
  listTitle: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 10,
  },
});
```

SAMPLE OUTPUT



type	name	date	amount
💳	CHOCOLATE	July 7th	$2000

RESULT

Thus the cross platform application for simple expense manager was Implemented and executed successfully.

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 3 Date:	**Develop an Application to Convert Units from Imperial System to Metric System**

AIM:

Build a application to Convert Units from Imperial System to Metric System.

ALGORITHM

Step 1: Initialize state variables for kilometers input, kilometers converted value, kilograms input, and kilograms converted value.
Step 2: Create conversion functions for kilometers to miles and kilograms to pounds.
Step 3: Implement a clear function to reset all fields.
Step 4: Render input fields, conversion buttons, and result displays for both conversions.
Step 5: Set up input handlers to update state variables.
Step 6: Attach conversion functions to the "Convert" buttons.
Step 7: Display the converted values or error messages.
Step 8: Add a "Clear" button to reset fields.
Step 9: Integrate the component into your React Native app.
Step 10: Test and optimize as needed for production.

PROGRAM

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

function UnitConverter() {
  const [kmInputValue, setKmInputValue] = useState('');
  const [kmConvertedValue, setKmConvertedValue] = useState('');

  const [kgInputValue, setKgInputValue] = useState('');
  const [kgConvertedValue, setKgConvertedValue] = useState('');

  const convertKmToMiles = () => {
    const km = parseFloat(kmInputValue);
    if (!isNaN(km)) {
      const miles = km / 1.60934;
      setKmConvertedValue(`${miles.toFixed(2)} miles`);
    } else {
      setKmConvertedValue('Invalid input');
    }
  };

  const convertKgToPounds = () => {
```

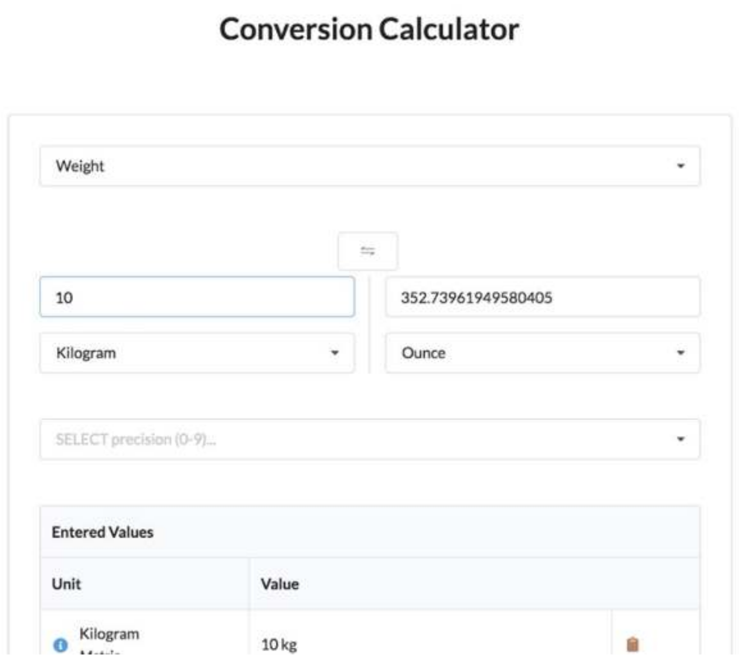
This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
    const kg = parseFloat(kgInputValue);
    if (!isNaN(kg)) {
      const pounds = kg * 2.20462;
      setKgConvertedValue(`${pounds.toFixed(2)} pounds`);
    } else {
      setKgConvertedValue('Invalid input');
    }
  };

  const clearFields = () => {
    setKmInputValue('');
    setKmConvertedValue('');
    setKgInputValue('');
    setKgConvertedValue('');
  };

  return (
    <View>
      <Text>Kilometers to Miles Converter</Text>
      <TextInput
        placeholder="Enter value in kilometers"
        keyboardType="numeric"
        value={kmInputValue}
        onChangeText={(text) => setKmInputValue(text)}
      />
      <Button title="Convert" onPress={convertKmToMiles} />
      <Text>{kmConvertedValue}</Text>

      <Text>Kilograms to Pounds Converter</Text>
      <TextInput
        placeholder="Enter value in kilograms"
        keyboardType="numeric"
        value={kgInputValue}
        onChangeText={(text) => setKgInputValue(text)}
      />
      <Button title="Convert" onPress={convertKgToPounds} />
      <Text>{kgConvertedValue}</Text>

      <Button title="Clear" onPress={clearFields} />
    </View>
  );
}

export default UnitConverter;
```

Downloaded by Sona Kannan (sona29102003@gmail.com)

SAMPLE OUTPUT


Conversion Calculator

Weight

10 | 352.73961949580405

Kilogram | Ounce

SELECT precision (0-9)...

Entered Values

Unit	Value
Kilogram	10 kg

RESULT

Thus the Cross Platform Application to Convert Units from Imperial System to Metric System was Implemented and executed successfully.

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 4 **Date:**	**Design and develop a cross platform application for day to day task (to-do) management**

AIM:

To develop a cross platform application for day to day task (To-Do) management using react native.

ALGORITHM

Step 1: Start

Step 2: Create a React Native component.

Step 3: Import required components and libraries

Step 4: Set up state variables using use State.

Step 5: Implement add Todo to add items to the list.

Step 6: Implement toggle Todo to mark items as completed.

Step 7: Implement remove Todo to delete items.

Step 8: Create UI components: input, button, list.

Step 9: Use Flat List to display to-dos.

Step 10: Apply basic styles using Style Sheet.

Step 11: Start the app on an emulator.

Step 12: Verify functionality and customize as needed.

PROGRAM

```
import React, { useState } from 'react';
import { StyleSheet, View, Text, FlatList } from 'react-native';
import { Button, TextInput, List, Checkbox } from 'react-native-paper';
export default function App() {
  const [text, setText] = useState('');
  const [todos, setTodos] = useState([]);
  const addTodo = () => {
    if (text.trim() !== '') {
      setTodos([...todos, { text, id: Date.now(), completed: false }]);
```

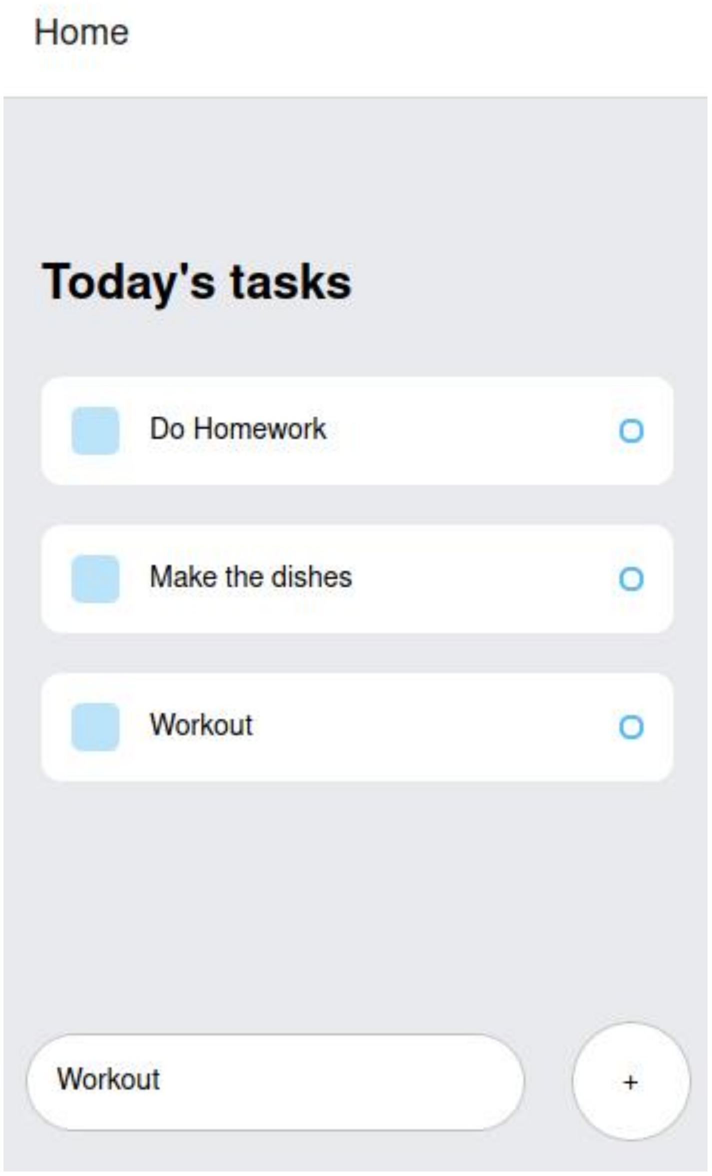
Downloaded by Sona Kannan (sona29102003@gmail.com)

```
      setText('');
    }
  };
const toggleTodo = (id) => {
    setTodos(
      todos.map((todo) =>
        todo.id === id ? { ...todo, completed: !todo.completed } : todo
      )
    );
  };
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

SAMPLE OUTPUT



RESULT

Thus the Cross Platform Application for day to day task management was Implemented and executed successfully.

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 5 **Date:**	**Design an android application using Cordova for a user login screen with username and password**

AIM:

To design android application using Cordova for a user login screen with username and password..

ALGORITHM

Step 1: Start

Step 2: Install the Cordova using the npm

Step 3: Create a new Cordova project with a unique name.

Step 4: Go to the newly created project directory.

Step 5 Add Android as a platform for your Cordova project.

Step 6: Create an index.html file

for the login screen and include form elements for username, password ,reset, and submit buttons.

Step 7: Create an index.js file to handle form interactions and Add event listeners for reset and submit buttons and Implement logic for form reset and login validation.

Step 8: Add a listener for the device ready event in JavaScript.

Step 9: Set AndroidX Enabled to true

and Configure preferences in the config.xml file.

Step 10: Use Cordova to build the Android app and Run the app on an

Android emulator or physical device using Cordova or Android Studio.

PROGRAM

HTML:

```
<!DOCTYPE html>
<html>
<head>
   <title>Login</title>
</head>
<body>
   <h1>Login</h1>
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
    <form>
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>
        <button type="button" id="resetButton">Reset</button>
        <button type="button" id="submitButton">Submit</button>
    </form>
    <script src="js/index.js"></script>
</body>
</html>
```

JS:

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

function onDeviceReady() {
    document.getElementById("resetButton").addEventListener("click", resetForm);
    document.getElementById("submitButton").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;

    // Perform login validation here (e.g., check username and password)

    if (username === "example" && password === "password") {
        alert("Login Successful!");
    } else {
        alert("Login Failed. Please check your credentials.");
    }
}
```

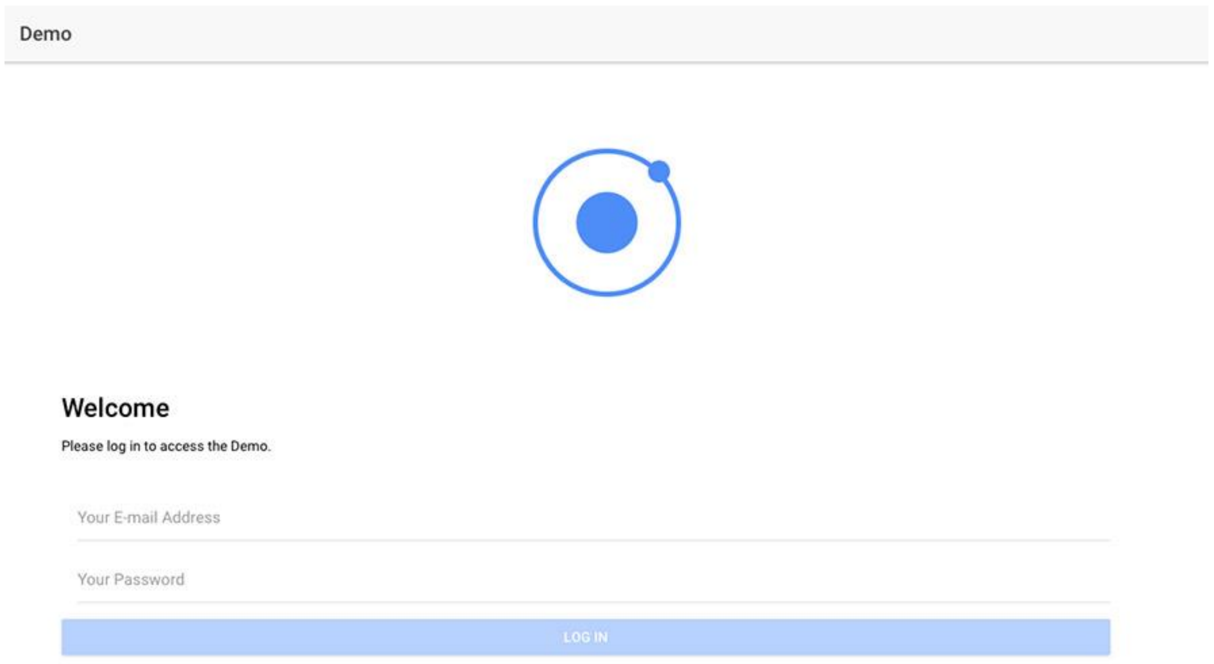
XML:

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

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
function onDeviceReady() {
   document.getElementById("resetButton").addEventListener("click", resetForm);
   document.getElementById("submitButton").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;

   // Perform login validation here (e.g., check username and password)

   if (username === "example" && password === "password") {
      alert("Login Successful!");
   } else {
      alert("Login Failed. Please check your credentials.");
   }
}
```

SAMPLE OUTPUT



This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

RESULT

Thus we design android application using Cordova for a user login screen with username and password was Implemented and executed successfully.

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 6 **Date:**	**Design an and develop an android application using Apache Cordova to find and display the current location of the user**

AIM:

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

ALGORITHM

Step 1: Start

Step 2: Install the Cordova globally.

Step 3: Create a new Cordova project and add Android platform for your project.

Step 4: Add the Cordova Geolocation plugin and create index.html and index.js for UI and logic.

Step 5: Set up a device ready event listener and Implement get Location function using navigator.geolocation.

Step 6: Create on Success function to display location info and Implement on Error function for error handling.

Step 7: Configure app preferences in config.xml.

Step 8: Build and run the app on an emulator or device

Step 9: Click the "Get Location" button to display the current location.

Step 10: Stop

PROGRAM

HTML:

```
<!DOCTYPE html>
<html>
<head>
   <title>Location App</title>
</head>
<body>
   <h1>Current Location</h1>
   <button onclick="getLocation()">Get Location</button>
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
    <p id="locationInfo"></p>
    <script src="js/index.js"></script>
</body>
</html>
```

JS:

```
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
    // Device is ready, initialize geolocation.
}
function getLocation() {
    navigator.geolocation.getCurrentPosition(
        onSuccess,
        onError,
        { enableHighAccuracy: true }
    );
}

function onSuccess(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var accuracy = position.coords.accuracy;
    var locationInfo = `Latitude: ${latitude}<br>Longitude: ${longitude}<br>Accuracy: ${accuracy}
meters`;

    document.getElementById("locationInfo").innerHTML = locationInfo;
}

function onError(error) {
    alert(`Error getting location: ${error.message}`);
```

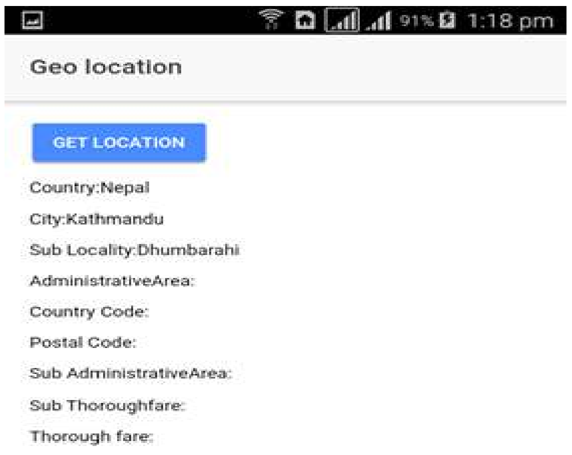
Downloaded by Sona Kannan (sona29102003@gmail.com)

}

XML:

```
<platform name="android">
    ...
    <preference name="AndroidXEnabled" value="true" />
    <preference name="Orientation" value="portrait" />
    ...
</platform>
```

SAMPLE OUTPUT



```
Country:Nepal
City:Kathmandu
Sub Locality:Dhumbarahi
AdministrativeArea:
Country Code:
Postal Code:
Sub AdministrativeArea:
Sub Thoroughfare:
Thorough fare:
```

RESULT

Thus we design and develop an android application using Apache Cordova to find and display the current location of the user Implemented and executed successfully.

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 7(A) **Date:**	**Write programs using Java to create Android application having Databases for a simple library application.**

AIM:

To write program using Java to create Android application having Databases for a simple library application.

ALGORITHM

Step 1: Start

Step 2: Set up a SQLite database for storing book information.

Step 3: Create a table named "books" with columns for bookID, title, and author.

Step 4: Initialize the application and establish a connection to the database

Step 5: Display a menu with options for the user: Add a book ,View all books ,Search for a book by title ,Exit

Step 6: Allow the user to input a book's title and author and insert the book into the database.

Step 7: Retrieve all books from the database and display a list of all books, including their titles and authors.

Step 8: Allow the user to input a title to search for.

Step 9: Provide an option to exit the application gracefully.

Step 10: Implement error handling to catch and display any database-related errors

Step11:Compile the Javacode and run the application in a console or terminal.

Step12: Stop.

PROGRAM

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
public class LibraryApp {
```

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
private static final String DB_URL = "jdbc:sqlite:library.db";

public static void main(String[] args) {
    try {
        // Create or connect to the SQLite database
        Connection connection = DriverManager.getConnection(DB_URL);

        // Create the "books" table if it doesn't exist
        String createTableSQL = "CREATE TABLE IF NOT EXISTS books (" +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "title TEXT NOT NULL," +
            "author TEXT NOT NULL)";
        connection.createStatement().execute(createTableSQL);

        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("Library Application");
            System.out.println("1. Add a book");
            System.out.println("2. View all books");
            System.out.println("3. Search for a book by title");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline

            switch (choice) {
                case 1:
                    addBook(connection, scanner);
                    break;
                case 2:
                    viewBooks(connection);
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
                break;
            case 3:
                searchBookByTitle(connection, scanner);
                break;
            case 4:
                System.out.println("Goodbye!");
                connection.close();
                System.exit(0);
            default:
                System.out.println("Invalid choice. Try again.");
        }
      }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void addBook(Connection connection, Scanner scanner) throws SQLException {
    System.out.print("Enter the title of the book: ");
    String title = scanner.nextLine();
    System.out.print("Enter the author of the book: ");
    String author = scanner.nextLine();

    String insertSQL = "INSERT INTO books (title, author) VALUES (?, ?)";
    PreparedStatement preparedStatement = connection.prepareStatement(insertSQL);
    preparedStatement.setString(1, title);
    preparedStatement.setString(2, author);
    preparedStatement.executeUpdate();

    System.out.println("Book added successfully!");
}
```

Downloaded by Sona Kannan (sona29102003@gmail.com)

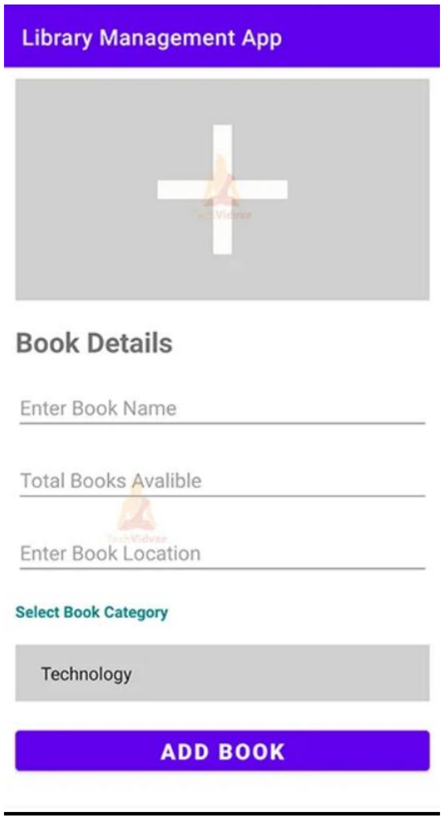
```
    private static void viewBooks(Connection connection) throws SQLException {
        String selectSQL = "SELECT id, title, author FROM books";
        ResultSet resultSet = connection.createStatement().executeQuery(selectSQL);

        System.out.println("List of Books:");
        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String title = resultSet.getString("title");
            String author = resultSet.getString("author");
            System.out.println(id + ". " + title + " by " + author);
        }
    }

    private static void searchBookByTitle(Connection connection, Scanner scanner) throws
SQLException {
        System.out.print("Enter the title to search for: ");
        String searchTitle = scanner.nextLine();

        String selectSQL = "SELECT id, title, author FROM books WHERE title LIKE ?";
        PreparedStatement preparedStatement = connection.prepareStatement(selectSQL);
        preparedStatement.setString(1, "%" + searchTitle + "%");
        ResultSet resultSet = preparedStatement.executeQuery();

        System.out.println("Search Results:");
        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String title = resultSet.getString("title");
            String author = resultSet.getString("author");
            System.out.println(id + ". " + title + " by " + author);
        }
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
    }
}
```

SAMPLE OUTPUT



RESULT

Thus we implemented program using Java to create Android application having Databases for a simple library Implemented and executed successfully.

Downloaded by Sona Kannan (sona29102003@gmail.com)

Ex. No: 7(B) **Date:**	**Write programs using Java to create Android application having Databases For displaying books available, books lend, book reservation**

AIM:

To write programs using Java to create Android application having Databases For displaying books available, books lend, book reservation

ALGORITHM

Step 1: Start

Step 2: Set up a SQLite database for storing book information.

Step 3: Create "books" and "loans" tables for book and loan information..

Step 4: Initialize the application and establish a connection to the database

Step 5: Display a menu with options for the user:Display available books, Lend a book ,Reserve a book , Exit

Step 6: Check book availability and Record loans in the database.

Step 7: Retrieve all books from the database and display a list of all books, including their titles and authors.

Step 8:Provide an option to exit the application gracefully.

Step 9: Implement error handling to catch and display any database-related errors

Step10:Compile the Javacode and run the application in a console or terminal.

Step11: Stop.

PROGRAM

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
public class LibrarySystem {
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
private static final String DB_URL = "jdbc:sqlite:library.db";

public static void main(String[] args) {
  try {
    // Create or connect to the SQLite database
    Connection connection = DriverManager.getConnection(DB_URL);

    // Create tables if they don't exist
    createTables(connection);

    Scanner scanner = new Scanner(System.in);
    while (true) {
      System.out.println("Library Management System");
      System.out.println("1. Display Available Books");
      System.out.println("2. Lend a Book");
      System.out.println("3. Reserve a Book");
      System.out.println("4. Exit");
      System.out.print("Enter your choice: ");
      int choice = scanner.nextInt();
      scanner.nextLine(); // Consume the newline

      switch (choice) {
        case 1:
          displayAvailableBooks(connection);
          break;
        case 2:
          lendBook(connection, scanner);
          break;
        case 3:
          reserveBook(connection, scanner);
          break;
```

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
                case 4:
                    System.out.println("Goodbye!");
                    connection.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void createTables(Connection connection) throws SQLException {
    // Create "books" table
    String createBooksTableSQL = "CREATE TABLE IF NOT EXISTS books (" +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "title TEXT NOT NULL," +
            "is_available BOOLEAN DEFAULT 1)";
    connection.createStatement().execute(createBooksTableSQL);

    // Create "loans" table
    String createLoansTableSQL = "CREATE TABLE IF NOT EXISTS loans (" +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "book_id INTEGER NOT NULL," +
            "user_name TEXT NOT NULL," +
            "FOREIGN KEY (book_id) REFERENCES books (id))";
    connection.createStatement().execute(createLoansTableSQL);
}

private static void displayAvailableBooks(Connection connection) throws SQLException {
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

```
        String selectSQL = "SELECT id, title FROM books WHERE is_available = 1";
        ResultSet resultSet = connection.createStatement().executeQuery(selectSQL);

        System.out.println("Available Books:");
        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String title = resultSet.getString("title");
            System.out.println(id + ". " + title);
        }
    }

    private static void lendBook(Connection connection, Scanner scanner) throws SQLException {
        System.out.print("Enter the ID of the book you want to lend: ");
        int bookId = scanner.nextInt();
        scanner.nextLine(); // Consume the newline

        // Check if the book is available
        String checkAvailabilitySQL = "SELECT is_available FROM books WHERE id = ?";
        PreparedStatement availabilityStatement = connection.prepareStatement(checkAvailabilitySQL);
        availabilityStatement.setInt(1, bookId);
        ResultSet availabilityResult = availabilityStatement.executeQuery();

        if (availabilityResult.next()) {
            boolean isAvailable = availabilityResult.getBoolean("is_available");
            if (isAvailable) {
                // Book is available, lend it
                System.out.print("Enter your name: ");
                String userName = scanner.nextLine();

                // Update book availability
                String updateAvailabilitySQL = "UPDATE books SET is_available = 0 WHERE id = ?";
```

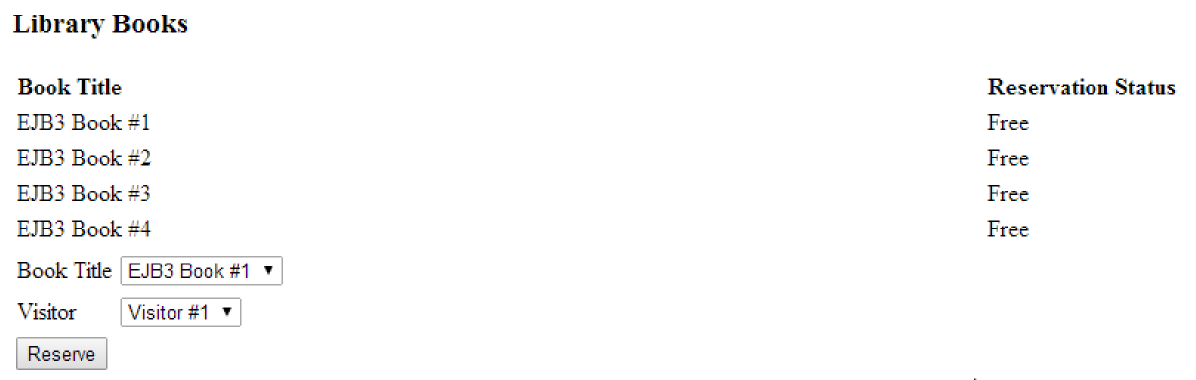
Downloaded by Sona Kannan (sona29102003@gmail.com)

```
                PreparedStatement updateAvailabilityStatement =
connection.prepareStatement(updateAvailabilitySQL);
                updateAvailabilityStatement.setInt(1, bookId);
                updateAvailabilityStatement.executeUpdate();

                // Record the loan
                String insertLoanSQL = "INSERT INTO loans (book_id, user_name) VALUES (?, ?)";
                PreparedStatement insertLoanStatement = connection.prepareStatement(insertLoanSQL);
                insertLoanStatement.setInt(1, bookId);
                insertLoanStatement.setString(2, userName);
                insertLoanStatement.executeUpdate();

                System.out.println("Book lent successfully!");
            } else {
                System.out.println("Sorry, this book is already lent.");
            }
        } else {
            System.out.println("Invalid book ID.");
        }
    }
    private static void reserveBook(Connection connection, Scanner scanner) throws SQLException {
        // Similar logic to lending a book can be implemented here
        // You can check availability and reserve the book if it's available
        // This is left as an exercise for further development
        System.out.println("Reservation functionality not implemented in this example.");
    }
}
```

This document is available free of charge on studocu

Downloaded by Sona Kannan (sona29102003@gmail.com)

SAMPLE OUTPUT



Library Books

Book Title	Reservation Status
EJB3 Book #1	Free
EJB3 Book #2	Free
EJB3 Book #3	Free
EJB3 Book #4	Free

Book Title EJB3 Book #1

Visitor Visitor #1

RESULT

Thus we implemented programs using Java to create Android application having Databases for displaying books available, books lend, book reservation Implemented and executed successfully.

Downloaded by Sona Kannan (sona29102003@gmail.com)