

**KARPAGA VINAYAGA**  
**COLLEGE OF ENGINEERING AND TECHNOLOGY**

GST Road- ChinnaKolambakkam- MadhuranthakamTaluk-603 308



**DEPARTMENT OF**  
**ARTIFICIAL INTELLIGENCE AND DATASCIENCE**

**CCS332**  
**APP DEVELOPMENT LABORATORY**

**NAME** : \_\_\_\_\_

**REGISTER NO** : \_\_\_\_\_

**BRANCH** : \_\_\_\_\_

**SEM/YEAR** : \_\_\_\_\_

**KARPAGA VINAYAGA**  
**COLLEGE OF ENGINEERING AND TECHNOLOGY**  
GST Road- ChinnaKolambakkam- Madhuranthakam Taluk-603 308



**BONAFIDE CERTIFICATE**

Name .....

Year ..... Semester ..... Branch .....

University Registration No :

Certified that this is a bonafied record of work done by the above student in the .....

..... Laboratory during the year .....

\_\_\_\_\_  
Signature of Staff In - Charge

\_\_\_\_\_  
Signature of Head of the Department

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

\_\_\_\_\_  
Signature of Internal Examiner

\_\_\_\_\_  
Signature of External Examiner

# **SYLLABUS**

## **CCS332 - APP DEVELOPMENT LABORATORY**

### **COURSE OBJECTIVES:**

- To learn development of native applications with basic GUI Components
- To develop cross-platform applications with event handling
- To develop applications with location and data storage capabilities
- To develop web applications with database access

### **LIST OF EXPERIMENTS :**

1. Using react native, build a cross platform application for a BMI calculator.115
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

### **OUTCOMES :**

**On completion of the course, the students will be able to:**

**CO1:** Develop Native applications with GUI Components.

**CO2:** Develop hybrid applications with basic event handling.

**CO3:** Implement cross-platform applications with location and data storage capabilities.

**CO4:** Implement cross platform applications with basic GUI and event handling.

**CO5:** Develop web applications with cloud database access.

## INDEX

S.NO	Date	Content	Page No	Marks	Signature
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  a. For a simple library application.			
		b. 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**

## **USING REACT NATIVE, BUILD A CROSS PLATFORM APPLICATION FOR A BMI CALCULATOR**

### **AIM:**

To develop a mobile application to calculate Body Mass Index (BMI) which can work seamlessly on both Android and iOS platforms.

### **PROCEDURE:**

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

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

```
blur(e){ this.calcu  
lateBMI();  
}  
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";
    }
    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()
    t();
    this.calculateBMI
    ();
    }

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

componentDidMount(){ set
    Interval(this.ticker, 60000);
}

```



```
change(e){ e.preventDefault
```

```
Default();
```

```
console.log(e.target);
```

```
this.setState( {name: e.target.value} );
```

```
}
```

```
render()
```

```
{return(
```

```
<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>

);
}
}
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>

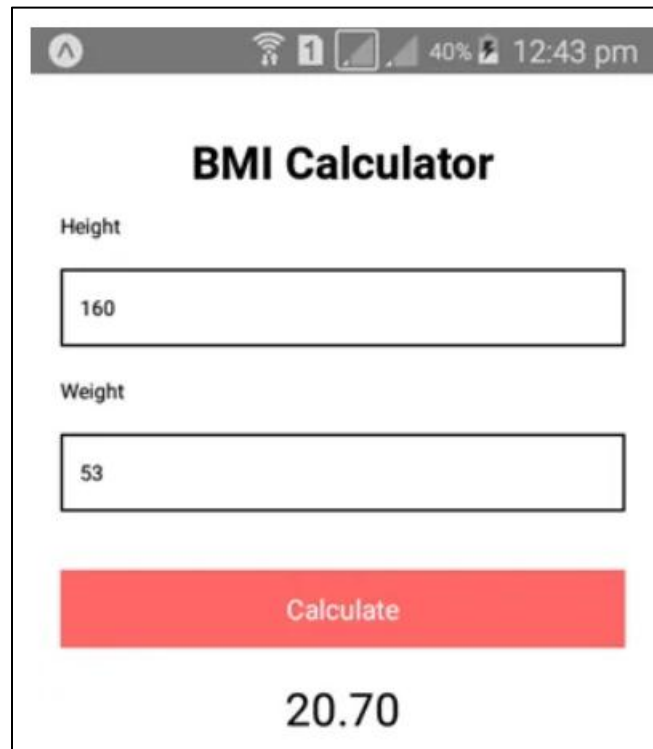
    <p>Weight in KG:
    <input type="number" id="weight"><br><span id="weight_error"></span>
    </p>

    <button id="btn">Calculate</button>

```

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

## OUTPUT:

A screenshot of a mobile application titled "BMI Calculator". The interface is clean and modern. At the top, there's a status bar with icons for signal, Wi-Fi, battery (40%), and time (12:43 pm). Below the title, there are two input fields: "Height" with the value "160" and "Weight" with the value "53". A prominent red "Calculate" button is positioned below the inputs. At the bottom, the result "20.70" is displayed in a large, bold font.

**BMI Calculator**

Height

160

Weight

53

Calculate

20.70

## RESULT:

A reusable BMI calculator app working seamlessly on Android and iOS devices with an easy to use interface.

## **EX.NO: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**

### **AIM:**

To build an application for tracking daily expenses and income which provides categorized weekly summaries of money inflow and outflow.

### **PROCEDURE:**

1. Install Android Studio and create a new project.
2. Create XML layouts for main activity, transaction input, and summary display.
3. Implement a transaction data model and set up a local database (e.g., Room).
4. Write code to handle user input for adding transactions and calculate category-wise weekly summaries.
5. Run the app on emulators/devices, debug issues, and ensure functionality works as expected.

### **PROGRAM:**

#### **MainActivity.java :**

```
import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;


import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
```

```
Button addTransactionButton = findViewById(R.id.add_transaction_button);
addTransactionButton.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        Intent intent = new Intent(MainActivity.this, AddTransactionActivity.class);

        startActivity(intent);

    }

});
```

```
Button viewSummaryButton = findViewById(R.id.view_summary_button);
viewSummaryButton.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        Intent intent = new Intent(MainActivity.this, WeeklySummaryActivity.class);

        startActivity(intent);

    }

});

}

}
```

### **AddTransactionActivity.java :**

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

```
import androidx.appcompat.app.AppCompatActivity;

public class AddTransactionActivity extends AppCompatActivity {

    private EditText categoryEditText;

    private EditText amountEditText;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_add_transaction);

        categoryEditText = findViewById(R.id.category_edit_text);

        amountEditText = findViewById(R.id.amount_edit_text);

        Button addButton = findViewById(R.id.add_button);

        addButton.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                String category = categoryEditText.getText().toString();

                double amount = Double.parseDouble(amountEditText.getText().toString());

                // Save transaction to database (not implemented here)

                // Replace this with your database handling code

                Toast.makeText(AddTransactionActivity.this, "Transaction added successfully",
                Toast.LENGTH_SHORT).show();

                finish();
            }
        });
    }
}
```

```
    }  
    });  
}  
}
```

### **WeeklySummaryActivity.java :**

```
import android.os.Bundle;  
import android.widget.TextView;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import java.util.List;  
  
public class WeeklySummaryActivity extends AppCompatActivity {  
  
    private TextView summaryTextView;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_weekly_summary);  
  
        summaryTextView = findViewById(R.id.summary_text_view);  
  
        // Retrieve weekly transactions and calculate summaries  
        List<Transaction> weeklyTransactions = getWeeklyTransactions();
```

```

String summaryText = generateSummaryText(weeklyTransactions);

summaryTextView.setText(summaryText);
}

// Placeholder methods (replace with actual implementation)
private List<Transaction> getWeeklyTransactions() {
    // Implement method to fetch transactions for the current week
    return null;
}

private String generateSummaryText(List<Transaction> transactions) {
    // Implement method to calculate and format category-wise summaries
    return "Weekly Summary:\nCategory 1: $100\nCategory 2: $50";
}
}

```

### **activity\_main.xml (Main Screen Layout) :**

```

<!-- Define UI layout for MainActivity -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/add_transaction_button"
        android:layout_width="wrap_content"

```



```
android:layout_height="wrap_content"
android:text="Add Transaction"
android:layout_centerHorizontal="true"
android:layout_marginTop="50dp"/>
```

```
<Button
```

```
    android:id="@+id/view_summary_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="View Weekly Summary"
    android:layout_below="@id/add_transaction_button"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"/>
```

```
</RelativeLayout>
```

### **activity\_add\_transaction.xml (Add Transaction Screen Layout) :**

```
<!-- Define UI layout for AddTransactionActivity -->
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
```

```
<EditText
```

```
    android:id="@+id/category_edit_text"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Category" />
```

<EditText

```
android:id="@+id/amount_edit_text"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Amount"
android:inputType="numberDecimal" />
```

<Button

```
android:id="@+id/add_button"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Add Transaction"
android:layout_marginTop="16dp" />
```

</LinearLayout>

### **activity\_weekly\_summary.xml (Weekly Summary Screen Layout) :**

<!-- Define UI layout for WeeklySummaryActivity -->

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
```

```
<TextView
```

```
    android:id="@+id/summary_text_view"
```

```
    android:layout_width="match_parent"
```

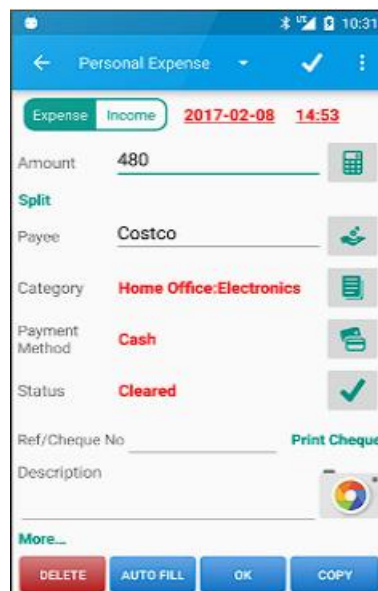
```
    android:layout_height="wrap_content"
```

```
    android:text="Weekly Summary"
```

```
    android:textSize="18sp" />
```

```
</RelativeLayout>
```

## **OUTPUT:**



## **RESULT:**

The result is an easy-to-use application for tracking personal finances with customization categories and graphical weekly summaries for analyzing spending patterns. Key metrics on budget vs spending also provided.

## **EX.NO:3 DEVELOP A CROSS PLATFORM APPLICATION TO CONVERT UNITS FROM IMPERIAL SYSTEM TO METRIC SYSTEM ( KM TO MILES, KG TO POUNDS ETC.,)**

### **AIM:**

To Develop a cross platform application to convert units from imperial system to metric system ( km to miles, kg to pounds etc.,)

### **PROCEDURE:**

- 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:**

#### **Create a New Flutter Project :**

```
flutter create unit_converter  
cd unit_converter
```

#### **Define Unit Conversion Logic :**

```
// Unit Converter Logic  
  
double kmToMiles(double km) {  
    return km * 0.621371;  
}
```

```
double kgToPounds(double kg) {  
    return kg * 2.20462;  
}
```

```
// Add more conversion functions as needed
```

### **User Interface (UI) :**

- **Open the lib/main.dart file to design the UI using Flutter widgets.**

```
import 'package:flutter/material.dart';
```

```
void main() {  
    runApp(UnitConverterApp());  
}
```

```
class UnitConverterApp extends StatelessWidget {  
    @override  
    Widget build(BuildContext context) {  
        return MaterialApp(  
            title: 'Unit Converter',  
            home: UnitConverterScreen(),  
        );  
    }  
}
```

```
class UnitConverterScreen extends StatefulWidget {  
    @override
```

```
_UnitConverterScreenState createState() => _UnitConverterScreenState();  
}
```

```
class _UnitConverterScreenState extends State<UnitConverterScreen> {  
  double inputValue;  
  String selectedUnit = 'Kilometers';  
  String result = "";  
  
  void convertUnit() {  
    setState(() {  
      if (selectedUnit == 'Kilometers') {  
        result = '${kmToMiles(inputValue)} miles';  
      } else if (selectedUnit == 'Kilograms') {  
        result = '${kgToPounds(inputValue)} pounds';  
      }  
      // Add more conditions for other units  
    });  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text('Unit Converter'),  
      ),  
      body: Padding(  

```

```
padding: EdgeInsets.all(16.0),
```

```
child: Column(
```

```
  crossAxisAlignment: CrossAxisAlignment.center,
```

```
  children: [
```

```
    TextField(
```

```
      decoration: InputDecoration(
```

```
        labelText: 'Enter value',
```

```
      ),
```

```
      keyboardType: TextInputType.number,
```

```
      onChanged: (value) {
```

```
        inputValue = double.tryParse(value) ?? 0.0;
```

```
      },
```

```
    ),
```

```
    SizedBox(height: 20.0),
```

```
    DropdownButton<String>(
```

```
      value: selectedUnit,
```

```
      onChanged: (value) {
```

```
        setState(() {
```

```
          selectedUnit = value!;
```

```
        });
```

```
      },
```

```
      items: <String>['Kilometers', 'Kilograms'].map((String value) {
```

```
        return DropdownMenuItem<String>(
```

```
          value: value,
```

```
          child: Text(value),
```

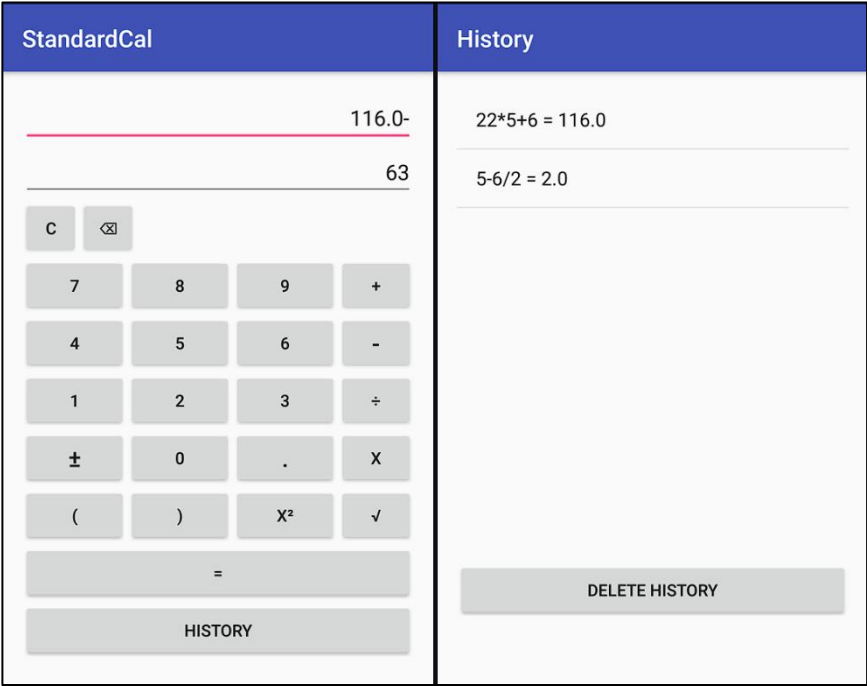
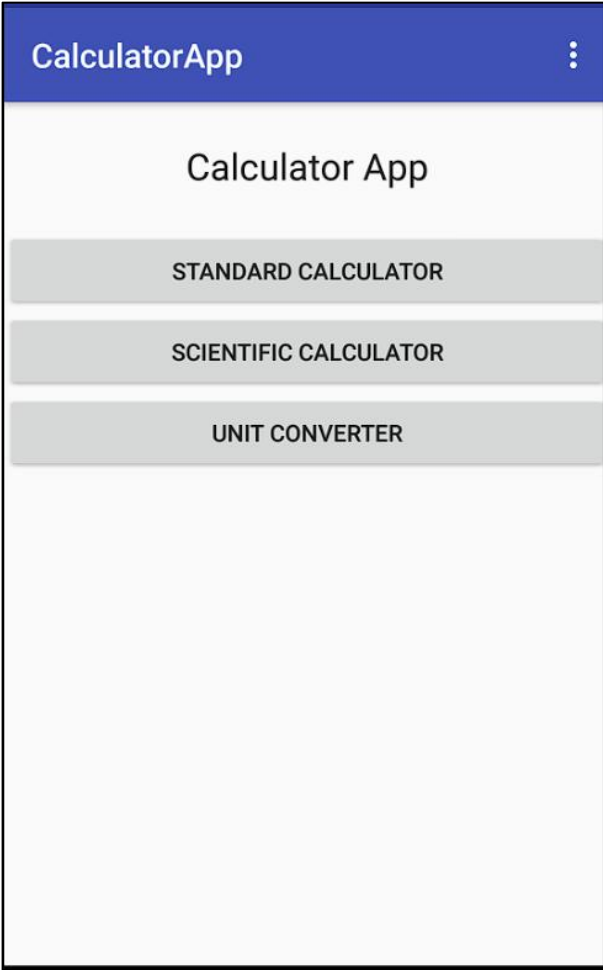
```
        );
```

```
    }).toList(),
  ),

  SizedBox(height: 20.0),
  ElevatedButton(
    onPressed: () {
      convertUnit();
    },
    child: Text('Convert'),
  ),
  SizedBox(height: 20.0),
  Text(
    'Result: $result',
    style: TextStyle(fontSize: 18.0, fontWeight: FontWeight.bold),
  ),
],
),
),
);
}
```



**OUTPUT:**



UnitCoverter	UnitLength
<div>AREA</div> <div>LENGTH</div> <div>WEIGHT</div> <div>TEMPERATURE</div>	<div><div>Nanometer</div><div>Millimeter</div><div>Centimeter</div><div>Meter</div><div>Kilometer</div><div>Inches</div><div>Foot</div><div>Yards</div><div>Mile</div></div> <div><div>8</div><div>9</div><div>5</div><div>6</div><div>2</div><div>3</div><div>0</div><div>.</div></div>

## **RESULT:**

To convert units from imperial system to metric system using cross platform application were Develop .

## **EX.NO:4 DESIGN AND DEVELOP A CROSS PLATFORM APPLICATION FOR DAY TO DAY TASK (TO-DO) MANAGEMENT.**

### **AIM:**

To Design and develop a cross platform application for day-to-day task (to-do) management.

### **PROCEDURE:**

Step 1: Start

Step 2: Create a Flutter 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 :**

- **Open a terminal or command prom**
- **Create a new Flutter project**

```
flutter create todo_app
```

```
cd todo_app
```

- **Open lib/main.dart and replace the default code with the following for the task management app UI**

```
import 'package:flutter/material.dart';
```

```
void main() {  
  runApp(TodoApp());  
}
```

```
class TodoApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Todo App',  
      theme: ThemeData(  
        primarySwatch: Colors.blue,  
        visualDensity: VisualDensity.adaptivePlatformDensity,  
      ),  
      home: TodoListScreen(),  
    );  
  }  
}
```

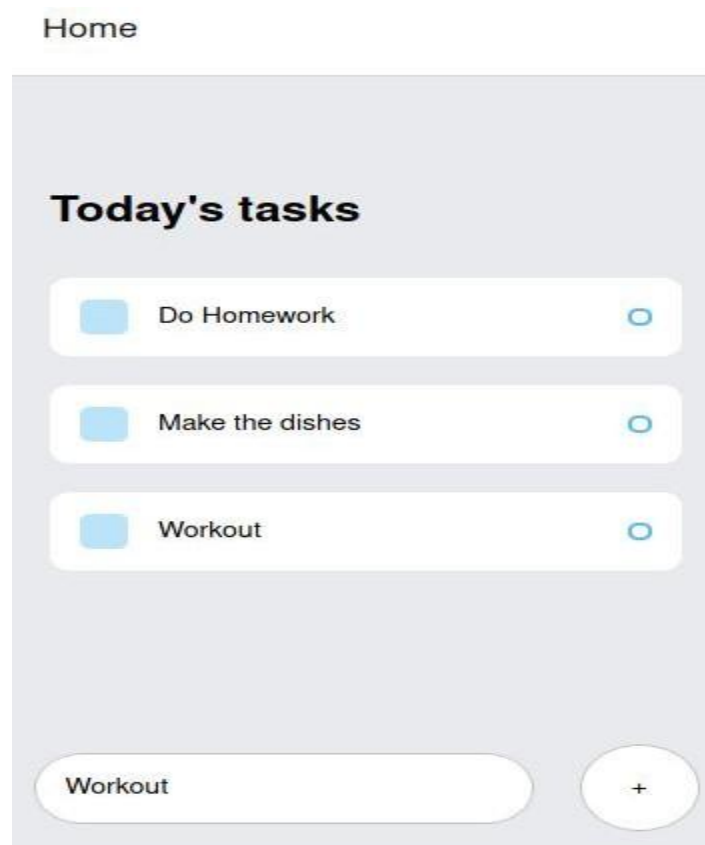
```
class TodoListScreen extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {
```

```
return Scaffold(  
  appBar: AppBar(  
    title: Text('Todo List'),  
  ),  
  body: Center(  
    child: Text('Todo List Screen'),  
  ),  
  floatingActionButton: FloatingActionButton(  
    onPressed: () {  
      // Navigate to Add Task Screen  
      Navigator.push(  
        context,  
        MaterialPageRoute(builder: (context) => AddTaskScreen()),  
      );  
    },  
    child: Icon(Icons.add),  
  ),  
);  
}
```

```
class AddTaskScreen extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text('Add Task'),  
      ),  
    );  
  }  
}
```

```
body: Center(  
  
  child: Text('Add Task Screen'),  
),  
);  
}  
}
```

**OUTPUT :**



**RESULT:**

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

**EX.NO:5      DESIGN AN ANDROID APPLICATION USING CORDOVA FOR USER LOGIN SCREEN WITH USERNAME, PASSWORD, RESET BUTTON AND A SUBMIT BUTTON. ALSO, INCLUDE HEADER IMAGE AND A LABEL. USE LAYOUT MANAGERS.**

**AIM:**

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

**PROCEDURES:**

1. Install Cordova globally using npm
2. Create a new Cordova project named LoginApp.
3. Change directory to your newly created project.
4. Add the Android platform to your Cordova project.
5. Customize the layout to include username and password input fields, reset button, submit button, header image, and label.
6. Inside index.html, add JavaScript functions to handle form actions (e.g., resetForm() for clearing input fields, submitForm() for processing login).
7. Place your header image (header\_image.jpg or any suitable format) inside the www directory of your Cordova project.
8. Connect an Android device to your computer via USB or start an Android emulator.

**PROGRAM:**

**Open www/index.html file and replace the content with the following HTML for the login screen layout:**

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Login Page</title>
```



```
<style>

  body {

    font-family: Arial, sans-serif;

    text-align: center;

  }

  #header {

    margin-bottom: 20px;

  }

  #loginForm {

    margin: 0 auto;

    width: 80%;

    max-width: 400px;

  }

  input[type="text"], input[type="password"], button {

    width: 100%;

    padding: 10px;

    margin-bottom: 15px;

    box-sizing: border-box;

  }

</style>

</head>

<body>

<div id="header">



<h2>Login Form</h2>

</div>
```

```
<form id="loginForm">

<input type="text" id="username" placeholder="Username" required><br>
<input type="password" id="password" placeholder="Password" required><br>
<button type="button" onclick="resetForm()">Reset</button>
<button type="button" onclick="submitForm()">Submit</button>

<p id="message"></p>

</form>


<script>

    function resetForm() {

        document.getElementById("username").value = "";
        document.getElementById("password").value = "";
        document.getElementById("message").textContent = "";
    }

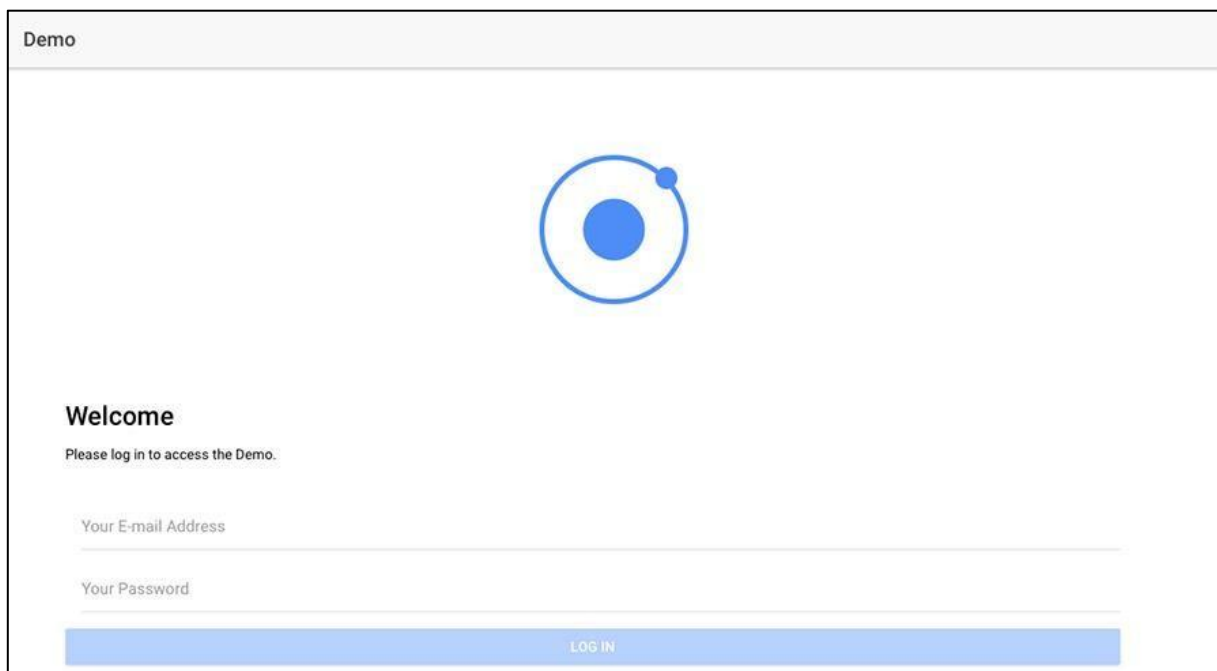

    function submitForm() {

        var username = document.getElementById("username").value;
        var password = document.getElementById("password").value;

        // Perform validation or submit logic here (e.g., send data to server)
        if (username === "admin" && password === "password") {
            document.getElementById("message").textContent = "Login successful!";
        } else {
            document.getElementById("message").textContent = "Invalid username or password.";
        }
    }
}
```

```
}  
</script>  
</body>  
</html>
```

## **OUTPUT:**



The screenshot shows a mobile application interface for a login screen. At the top, there is a header bar with the text "Demo". Below the header, there is a large blue circular logo consisting of two concentric circles with a small dot on the outer circle. Underneath the logo, the text "Welcome" is displayed in bold, followed by the instruction "Please log in to access the Demo." in a smaller font. Below this text, there are two input fields: "Your E-mail Address" and "Your Password". At the bottom of the form, there is a blue button labeled "LOG IN".

## **RESULT:**

An android application was designed successfully using cordova for user login screen with username, password, reset button and a submit button. Also, include header image and a label. Use layout managers.

## **EX.NO:6 DESIGN 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.

### **PROCEDURES:**

1. Set up the development environment
  - Install Node.js, Apache Cordova, Android SDK
  - Create a Cordova project using command line interface
2. Add the Cordova geolocation plugin
  - Run: cordova plugin add cordova-plugin-geolocation
  - This will integrate native geo APIs
3. Create a index.js file
  - Initialize the application logic and events here
4. Wait for 'deviceready' event
  - This event fires once Cordova is fully loaded
5. Access the geolocation plugin
  - Call navigator.geolocation.getCurrentPosition()
6. Handle successful location retrieval
  - Access latitude and longitude in the success callback
  - Display location info or reverse geocode
7. Handle error callback
  - Print error message if location could not be retrieved
8. Build the Android project
  - Run: cordova build android
  - This will generate Android project

9. Run the app on emulator/device

- Transfer the Android project and run
- Test geo features and location display

## **PROGRAM:**

### **AndroidManifest.xml**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.locationapp">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <application
        ...>
        ...
    </application>

</manifest>
```

### **MainActivity.java**

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.location.Location;
import android.widget.TextView;
```

```
import android.widget.Toast;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;

public class MainActivity extends AppCompatActivity {

    private TextView locationTextView;

    private FusedLocationProviderClient fusedLocationClient;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        locationTextView = findViewById(R.id.locationTextView);

        fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

        fetchLocation();

    }

    private void fetchLocation() {

        fusedLocationClient.getLastLocation()

            .addOnSuccessListener(this, location -> {

                if (location != null) {

                    double latitude = location.getLatitude();

                    double longitude = location.getLongitude();
```

```

        locationTextView.setText("Latitude: " + latitude + "\nLongitude: " + longitude);
    } else {
        Toast.makeText(MainActivity.this, "Location not available",
Toast.LENGTH_SHORT).show();
    }
})
    .addOnFailureListener(this, e -> {
        Toast.makeText(MainActivity.this, "Failed to get location",
Toast.LENGTH_SHORT).show();
    });
}
}

```

### **MainActivity.kt**

```

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView
import android.widget.Toast
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices

class MainActivity : AppCompatActivity() {

    private lateinit var locationTextView: TextView

```

```
private lateinit var fusedLocationClient: FusedLocationProviderClient
```

```
override fun onCreate(savedInstanceState: Bundle?) {
```

```
    super.onCreate(savedInstanceState)
```

```
    setContentView(R.layout.activity_main)
```

```
    locationTextView = findViewById(R.id.locationTextView)
```

```
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
```

```
    fetchLocation()
```

```
}
```

```
private fun fetchLocation() {
```

```
    fusedLocationClient.lastLocation
```

```
        .addOnSuccessListener { location ->
```

```
            if (location != null) {
```

```
                val latitude = location.latitude
```

```
                val longitude = location.longitude
```

```
                locationTextView.text = "Latitude: $latitude\nLongitude: $longitude"
```

```
            } else {
```

```
                Toast.makeText(this@MainActivity, "Location not available",  
Toast.LENGTH_SHORT).show()
```

```
            }
```

```
        }
```

```
        .addOnFailureListener { e ->
```

```
            Toast.makeText(this@MainActivity, "Failed to get location",
```



```
Toast.LENGTH_SHORT).show()
```

```
}
```

```
}
```

```
}
```

### **activity\_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:padding="16dp"
```

```
    tools:context=".MainActivity">
```

```
<TextView
```

```
    android:id="@+id/locationTextView"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

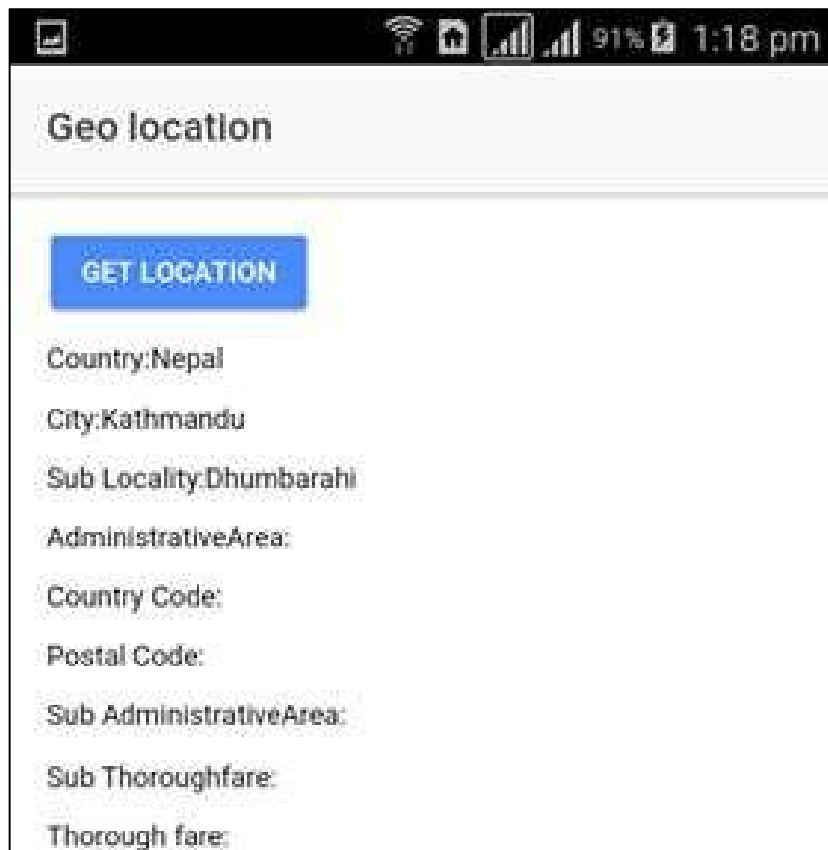
```
    android:textSize="18sp"
```

```
    android:text="Fetching location..."
```

```
    android:layout_centerInParent="true"/>
```

```
</RelativeLayout>
```

## **OUTPUT:**



## **RESULT:**

An android application using apache cordova to find and display the current location of the user was designed and developed successfully

**EX:NO: 7 ( a )    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.

**PROCEDURE :**

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 bytitle ,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.

Step 11: Compile the Javacode and run the application in a console or terminal.

Step 12: 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 {

    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);
```

```

        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!");  
}
```

```
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 {
```

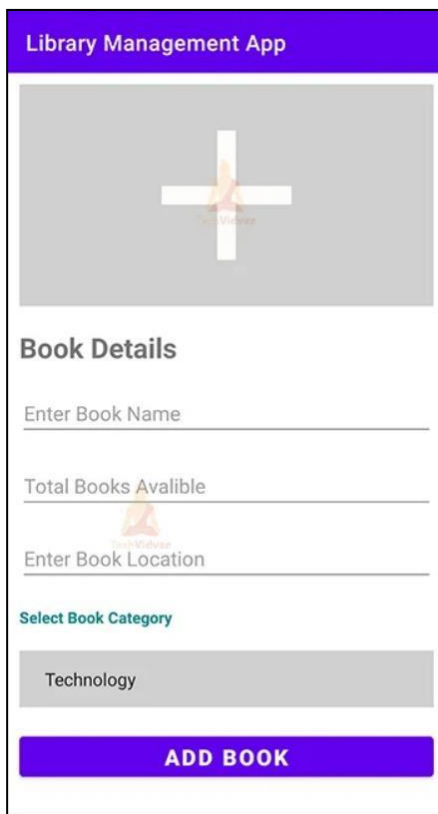
```
    System.out.print("Enter the title to search for: ");  
    String searchTitle = scanner.nextLine();
```

```
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)  
}  
}
```

## **OUTPUT :**



The screenshot shows a mobile application interface for a library management system. At the top is a purple header bar with the text "Library Management App". Below the header is a large gray square containing a white plus sign and a small orange icon of a person. Underneath this is a section titled "Book Details" in bold. This section contains four input fields: "Enter Book Name", "Total Books Available", and "Enter Book Location". Below these is a green text label "Select Book Category". Underneath this label is a gray rectangular box with the word "Technology" inside. At the bottom of the form is a purple button with the text "ADD BOOK" in white capital letters.

## **RESULT**

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

**EX:NO: 7 ( b )    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 booksavailable, books lend, book reservation

**PROCEDURE :**

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 abook , 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 andauthors.

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 {
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 existcreateTables(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;
    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
{
    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
        = ?";
        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.");
}
}

```

## 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

Visitor

## 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.