RAJARATA UNIVERSITY OF SRI LANKA

FACULTY OF TECHNOLOGY

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

ICT-2213 Mobile Computing

Practical Assignment

Guidelines

- Follow the **step-by-step tasks** provided for each question.
- Provide **screenshots** at every important step.
- Use **Android Studio** for Android development and **VSCode, Expo CLI with React Native** for cross-platform development.
- Submit as a zip file (name should be user registration number) with project source code (.java, .xml files), screenshots addressing each question). Screenshots should be clearly labeled.
- Submit the assignment on or before 08th May 2025.

Part 01 – Android Development (Marks - 60)

Develop a simple "MultiLayout UI Android app" using different layouts and a navigation bar.

Use the knowledge gained on **Android layouts, navigation, and data storage using SQLite** to build a **structured mobile UI**. This practical will guide you through implementing multiple layouts, navigating between screens using a Bottom Navigation Bar, and storing form data in an SQLite database.

Question 1: Implement Linear Layout with Navigation Bar (Home Page)

Requirements:

Text Fields & UI Components:

- Use **Linear Layout** (vertical) as the main layout.
- Add text fields for user input.
- Include labels (e.g., "Enter Name", "Enter Age").

Navigation Bar:

- Add a **bottom navigation bar** to navigate to other layouts.
- Implement a **Bottom Navigation Bar** with options to navigate to Frame, Constraint, and Relative layouts.

Background Image:

- Set a background image for the activity.
- Create a new **Android Studio project**.

Provide screenshots of:

- UI layout
- Navigation bar

(Marks – 10)

Question 2: Create a Frame Layout (User Profile Page)

Requirements:

Frame Layout with multiple UI components:

Add meaningful UI components like an **ImageView, and Buttons**
Design should be practical (e.g., a loading screen or a profile view).

- Create a new activity with **FrameLayout**.
- Add an **ImageView** (e.g., user profile image).
- Add a **TextView** for a display name (“ Welcome, [User Name!]”).
- Include a **Button** (Edit Profile button)

Provide screenshots of:

- UI design
- Code implementation

(Marks - 11)

Question 3: Implement Constraint Layout with Nested Linear Layout & Image (User Form Page)

Constraint Layout with nested Linear Layout:

Add **text fields**, **buttons**, and an **image**.
Use constraints properly for layout alignment.

- Create a new activity with **Constraint Layout**.

- Add a label " Student form"
- Add a **Linear Layout inside Constraint Layout** with labels and text fields to add name, email, phone number for students.
- **Buttons** to submit and clear form
- Use **ImageView** to add small icon next to input fields.
- Use **constraints to position elements properly**.

Provide screenshots of:

- UI structure
- Code implementation

(Marks - 12)

Question 4: Build a Relative Layout (Dashboard Page)

Add **text fields**, **labels**, **buttons**, and **images**.
Use Relative positioning (align components relative to each other).

- Create a new activity with **Relative Layout**.
- Add **TextView** labels at different positions for "Dashboard", " Notifications", and "Account status".
- **Buttons** to "View Profile" and "Settings".
- Add background Image for dashboard.
- Add **EditText** field to search within the dashboard.

Provide screenshots of:

- UI layout
- Code implementation
- Running app with each page

(Marks – 12)

Question 5: Store Data in SQLite (User Form Page)

Store user-submitted form data in an **SQLite database** and retrieve it when needed.

- Create an **SQLite database** to store user input (Name, Email, Phone Number).
- Use the **Submit button** to insert user input into the database.
- Implement **CRUD operations** (Create, Read, Update, Delete).
- Connect **the Submit button** to insert **form** data (Form Page) into the database.
- **Show a success message** when data is successfully saved (this is optional)

Database Structure:

- **Table Name**: students
- **Columns**: ID (Primary Key), Name, Email, Phone

Provide screenshots of:

- Database schema (table structure)
- Code implementation

(Marks - 15)

Part 02 – Cross-Platform App Development with React Native (Marks - 40)

Develop a **Cross-Platform React Native App** using React Navigation and simulate running the app on **both Web and Emulator**.

Question 1: Build a Welcome Page (Home Screen)

Create a HomeScreen.js file.

- Add a **Text View** with a Welcome Message (e.g., "Welcome to the App!").
- Include a **Button** to navigate to the **User Profile Page**.
- Set a modern, friendly design using background colors or images.

Use **Button Component** to navigate to the **Profile Screen**.

- Implement **React Navigation** to enable navigation between pages.

Provide screenshots of:

1. UI Layout for Home Screen.
2. Navigation setup.

(Marks - 15)

Question 2: Create a User Profile Page

Requirements:

- **React Navigation:**
    - Enable smooth navigation back to the Welcome Page using navigation props.

Create a ProfileScreen.js file.

- Add a **Text View** to display user details (e.g., "Name", "Email").
- Include an **Edit Profile Button** for user interaction..

Provide screenshots of:

1. Profile Page UI design.
2. Navigation interactions.

(Marks - 15)

Question 3: Run the App on Web and Emulator

- Run the app on both:
  - **Web Browser** (for responsive view).
  - **Android Emulator** or **iOS Simulator**.

Provide screenshots of

1. Running app on Web.
2. Running app on Emulator.

(Marks - 10)