SNS COLLEGE OF TECHNOLOGY

B

(Autonomous)

MCA- Internal Assessment –II (Nov 2023)

Academic Year 2023-2024(Odd) / Third Semester

19CAT701 –Mobile Application Development

Time: 1$^{1/2}$ Hours **Maximum Marks: 50**

Answer Key

PART - A (5 x 2 = 10 Marks)

		CO	BL
1	**Quote the term Intent?** An Intent is a messaging object you can use to request an action from another app component.	CO2	Und
2	**Predict the purpose of AsyncTask** AsyncTask enables proper and easy use of the UI thread. This class allows performing background operations and publishing results on the UI thread without having to manipulate threads and/or handlers.	CO2	Ana
3	**Show how Broadcast Receiver is implemented?** Broadcast in android is the system-wide events that can occur when the device starts, when a message is received on the device or when incoming calls are received, or when a device goes to airplane mode, etc.	CO3	Ana
4	**Generalize the use of Android Telephony APIs.** The telephony API is used to among other things monitor phone information including the current states of the phone, connections, network etc.	CO3	App
5	**Show your understanding on SQLite.** SQLite is an open-source, zero-configuration, self-contained, stand-alone, transaction relational database engine designed to be embedded into an application.	CO3	Und

PART - B (2 x 13 = 26, 1 X 14 = 14 Marks)

6 (a) Briefly discuss the use of Notifications in Android mobile apps. — CO2 — Ana

1. Small icon: required; set using setSmallIcon().
2. App name: provided by the system.
3. Time stamp: provided by the system, but you can override it using setWhen() or hide it using setShowWhen(false).
4. Large icon: optional; usually used only for contact photos. Don't use it for your app icon. Set using setLargeIcon().
5. Title: optional; set using setContentTitle().
6. Text: optional; set using setContentText().

(OR)

(b) Analyze how technology has influenced the evolution of services throughout their life cycle. — CO2 — Ana

Conceptualization and Design:

- **Digital Tools and Prototyping:** Advanced software tools and

technologies such as CAD (Computer-Aided Design), simulations, and virtual reality have revolutionized the conceptualization and design phase. They enable designers and service providers to create, visualize, and refine service concepts in more detail, thereby ensuring better alignment with user needs and preferences.

Development and Implementation:

- **Automation and Efficiency:** Technological advancements have automated numerous processes in service development, significantly enhancing efficiency and reducing time-to-market. Tools like AI, machine learning, and automated testing streamline development phases, enabling faster iterations and improved quality.

Delivery and Accessibility:

- **Online Platforms and Mobility:** The advent of the internet and mobile technology has reshaped how services are delivered, making them accessible remotely and conveniently. E-commerce, mobile apps, and on-demand services exemplify the impact of technology in expanding access and convenience for consumers.

Customer Interaction and Experience:

- **Personalization and AI:** Technology has revolutionized customer interactions by enabling personalized experiences. AI-driven chatbots, recommendation systems, and data analytics provide tailored solutions, enhancing customer satisfaction and engagement.

Maintenance and Improvement:

- **Data Analytics and Predictive Maintenance:** Technology facilitates continuous improvement through data analytics. Service providers can gather and analyze user data to understand preferences and pain points, enabling iterative enhancements. Predictive maintenance using IoT devices ensures better service reliability and reduced downtime.

Sustainability and Adaptability:

- **IoT and Sustainable Practices:** Technologies such as IoT and sensors promote sustainability in service provision by optimizing resource usage and minimizing waste. Cloud computing and scalable infrastructure also enable services to adapt dynamically to changing demands and market trends.

7 (a) Examine the layout resources available in the Android platform and its suitable applications

A layout resource defines the architecture for the UI in an Activity or a component of a UI.

CO3 Ana

file location:

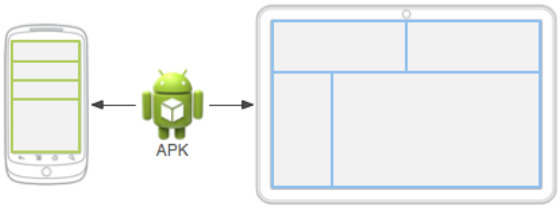
res/layout/*filename*.xml
The filename is used as the resource ID.

compiled resource datatype:

Resource pointer to a View (or subclass) resource

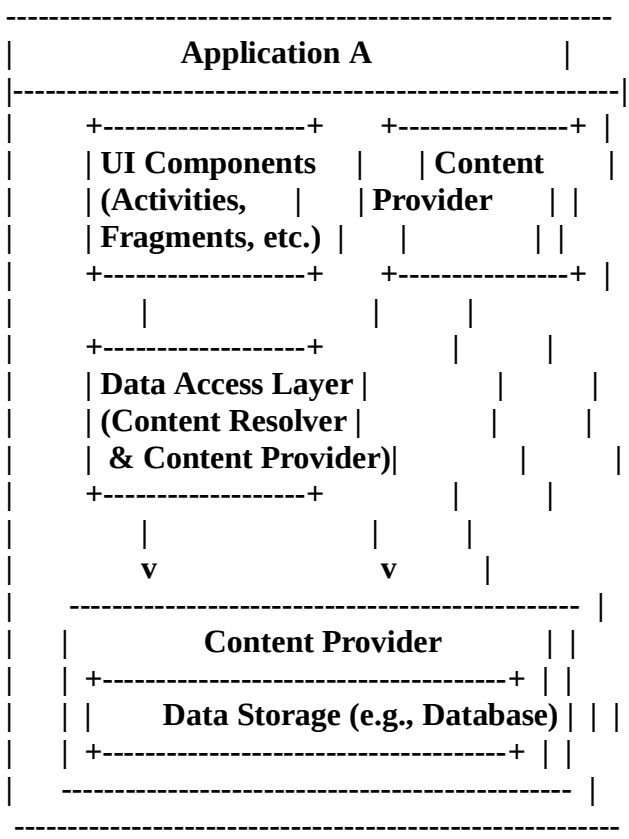
resource reference:

In Java: R.layout.*filename*
In XML: @[*package*:]layout/*filename*



(OR)

(b) **Analyze the application architecture using the Content Provider using a clean diagram.**



CO3 **Ana**

8 **(a)** **Creating an app for a client to track their fitness is your duty. The client has specified all of the requirements in detail and requests that the software be compatible with Android tablets and smartphones. Running, cycling, and weightlifting are just a few of the physical activities that the app ought to monitor. Exercise logs, fitness objectives, and progress tracking should all be available to users. The customer also requests that the app feature a social component where users may compete with peers and discuss their accomplishments.1. Considering that the app must retain user data, social interactions, and exercise records, how would you handle data storage and user profiles?**

CO2 **Cre**

User Profiles:

- **Table:** Create a table named "Users" to store user profiles.
- **Columns:** Include fields such as UserID, Username, Email, Password (hashed), Profile Picture, etc.
- **Attributes:** Ensure UserID acts as a primary key to uniquely identify users.

Exercise Logs:

- **Table:** Design a table named "ExerciseLogs" to track user activities.
- **Columns:** Include UserID (foreign key referencing Users table), ActivityType (running, cycling, weightlifting), Duration, Distance, Weight lifted, Date, etc.
- **Attributes:** Utilize UserID as a foreign key for relational mapping to user profiles.

Fitness Objectives:

- **Table:** Implement a table named "FitnessObjectives" to store user-set fitness goals.
- **Columns:** UserID (foreign key), GoalType (running, cycling, weightlifting), TargetDistance, TargetDuration, TargetWeight, etc.
- **Attributes:** Link UserID as a foreign key for associating goals with specific users.

Social Component:

- **Table:** Create a table named "SocialInteractions" to handle social features.
- **Columns:** UserID (foreign key), InteractionType (comments, likes, competitions), InteractionContent, Timestamp, etc.
- **Attributes:** Use UserID as a foreign key to map interactions to respective users.

Data Storage Considerations:

SQLite Database:

- Utilize SQLite to create and manage the database due to its lightweight nature and compatibility with Android.
- Ensure proper indexing for efficient querying and retrieval of data.
- Employ normalization techniques to minimize redundancy and maintain data integrity.

Data Security:

- Implement robust encryption techniques to secure sensitive user data like passwords.
- Hash passwords using strong cryptographic algorithms before storing them in the database.

Backup and Recovery:

- Establish regular backup mechanisms to prevent data loss.
- Implement recovery protocols in case of database corruption or failure.

Optimization:

- Regularly optimize database queries and structure for improved app performance.
- Utilize asynchronous tasks for data transactions to prevent UI freezes.

User Profile Management:

Authentication and Authorization:

- Implement secure authentication mechanisms like OAuth, JWT, or

biometric authentication for user login.
- Set up proper authorization levels to control access to specific features and data within the app.

User Engagement:
- Design intuitive interfaces to allow users to set fitness goals, view progress, and engage socially with peers.
- Provide personalized recommendations based on user activities and objectives.

In summary, leveraging SQLite for data storage in the fitness tracking app involves a well-structured database schema, robust security measures, efficient data handling, and user-centric features to ensure a seamless and secure fitness tracking experience while addressing the client's requirements.

(Or)

(b) **It is your responsibility to create an Android reminder software that lets users schedule and receive alerts for different chores and occasions. Utilizing Android's BroadcastReceiver component effectively is necessary to accomplish this. Users should get timely notifications according to the planned time and date of each event, and they have the ability to add, modify, and remove reminders. In addition, the app ought to manage reminders even when it's not actively running or the device is in a low-power mode.1. How BroadcastReceiver might be used to receive and manage reminders. To register and unregister the receiver, what kind of broadcast events would you use?** **CO3** **Cre**

Registering the BroadcastReceiver:
- To receive reminders, register the BroadcastReceiver within your app. This registration can be done in the AndroidManifest.xml file or dynamically within your app's code.
- In the AndroidManifest.xml

<receiver android:name=".ReminderBroadcastReceiver" />

Broadcast Events for Reminder Management:
- Define a custom action for the reminder broadcast event, for instance, "your_package_name.ACTION_REMINDER". This action will be used when sending reminders to the BroadcastReceiver.

Sending Reminders using Broadcast:
- When a user schedules a reminder, create an AlarmManager to trigger the reminder at the specified time.
- Use an Intent with the defined custom action and attach relevant reminder data (e.g., reminder message, ID, etc.).
- Trigger the broadcast using the sendBroadcast() method:

```
Intent reminderIntent = new Intent("your_package_name.ACTION_REMINDER");
reminderIntent.putExtra("reminder_id", yourReminderID);
reminderIntent.putExtra("reminder_message", yourReminderMessage);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, 0, reminderIntent, PendingIntent.FLAG_UPDATE_CURRENT);
```

```
AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, reminderTimeInMillis, pendingIntent);
```

Receiver Handling in BroadcastReceiver:

- Implement the `ReminderBroadcastReceiver` class to extend `BroadcastReceiver`.
- Override the `onReceive()` method to handle incoming reminders:

```
public class ReminderBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Retrieve reminder details from the received intent
        int reminderId = intent.getIntExtra("reminder_id", 0);
        String reminderMessage = intent.getStringExtra("reminder_message");

        // Show notification or perform necessary actions for the reminder
        // ...
    }
}
```

Unregistering the BroadcastReceiver:

- Unregister the BroadcastReceiver when it's no longer needed (e.g., when the app is closed):

```
context.unregisterReceiver(receiver);
```

Managing Reminders in Low-Power Mode:

Use `AlarmManager` to set reminders. It works even when the app is not actively running or the device is in a low-power state.
Set alarms with `AlarmManager` using `RTC_WAKEUP` to ensure the device wakes up for important reminders.