SNS COLLEGE OF TECHNOLOGY

B

(Autonomous)

MCA- Internal Assessment –III (Dec 2023)

Academic Year 2023-2024(Odd) / ThirdSemester

19CAT701 –Mobile Application Development

Time: 1 1/2 Hours **Maximum Marks: 50**

Answer All Questions

Answer key

PART - A (5 x 2 = 10 Marks)

		CO	BL
1	**Justify why SQLite database is better option of Storing the data in Android** Structured Data Management Efficient Querying Offline Capabilities Performance Security Integration	CO4	Ana
2	**Classify the types of sensors available in Android devices** Motion sensors Environmental sensors Position sensors Fingerprint sensor Microphone Camera	CO4	Ana
3	**Define geocoding** A Geocode is a set of latitude and longitude coordinates that represents a specific geographic place such as a landmark, street address, place name, or location.	CO5	Und
4	**Examine the Ionic Grid system used for positioning the components on the page** Grid: Container for all rows and columns, occupying the full width by default. Rows: Horizontal groups of columns, aligning them neatly. Content resides within columns, and only columns can be direct children of rows. Columns: Divide rows into sections, sized based on the specified "size" property (out of 12 columns per row).	CO5	Ana
5	**Evaluate the features of SCSS in IONIC framework** Variables Nesting Mixins Partials Imports Functions and Operator Control Directives	CO5	Und

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

6	(a)	**Critique existing animation libraries and frameworks for Android.** Android's Built-in Animation Framework: • Strengths: Simple to use, well-documented, integrates seamlessly with the Android framework. • Weaknesses: Limited animation capabilities, requires verbose code for complex animations, lacks advanced features like physics simulation or vector animation. 2. Lottie: • Strengths: Lightweight, efficient, supports vector animations for scalability and crisp rendering, large library of pre-built animations. • Weaknesses: Limited interactivity compared to code-based solutions, requires importing JSON files for custom animations, learning curve for advanced usage. 3. Nine Old Androids (NAA): • Strengths: Powerful and flexible, supports complex animations with physics simulation and advanced effects, extensive customization options. • Weaknesses: Steep learning curve, requires writing more code compared to other libraries, can be performance-intensive on older devices. 4. Reanimated: • Strengths: Declarative animation API with React-like syntax, simplifies complex animations, integrates well with React Native development. • Weaknesses: Limited adoption and community support compared to other options, may not be suitable for all animation needs. 5. MotionLayout: • Strengths: Enables creation of fluid, constraint-based animations for UI elements, simplifies transitions between layouts. • Weaknesses: Requires understanding ofConstraintLayout, can be challenging to master for complex animations.	CO4	Eva
		(OR)		
	(b)	**Recommend improvements to the design of Location object or location awareness APIs.** • Purpose: Represents the current URL of a document or window within a web browser. • Properties: o href: Returns the full URL of the current page. o protocol: Specifies the protocol used (e.g., "http:"). o hostname: Indicates the domain name of the server. o pathname: Provides the path portion of the URL. o search: Contains the query string (if any). o hash: Contains the fragment identifier (if any). • Methods: o assign(): Loads a new document. o replace(): Replaces the current document with a new	CO4	Eva

one, without creating a new history entry.
- reload(): Reloads the current document.

7 (a) Implement basic functionalities using HTML, CSS, and JavaScript within a hybrid mobile development framework.

1. Choose a Framework:

Popular choices include:

- React Native: Powerful and flexible, but might have a steeper learning curve.
- Ionic Framework: Beginner-friendly with pre-built UI components and extensive documentation.
- Apache Cordova/PhoneGap: Open-source and platform-agnostic, but offers less built-in functionality.

2. Create Your App Structure:

- HTML: Use index.html for the main layout and content, linking to separate HTML files for different screens.
- CSS: Define styles for different UI elements in a separate style.css file. CO5 App
- JavaScript: Implement logic and interactivity in a separate app.js file.

3. Code Your Basic Functionalities:

- Navigation: Use Javascript to control transitions between screens.
- Input Handling: Handle user interactions with buttons, forms, and other elements.
- Data Display: Show data from local storage, APIs, or external sources.
- Device Access: Leverage Cordova plugins or framework-specific APIs to access native features like camera, GPS, etc.

(OR)

(b) Create a simple IONIC UI with a card and a button, utilizing appropriate CSS components for styling and layout.

1. Set up Ionic Project:

- Ensure you have Ionic CLI installed: npm install -g @ionic/cli
- Create a new Ionic project: ionic start myCardApp blank
- Navigate to the project directory: cd myCardApp

2. Generate Card Component:

- Create a card component: ionic generate component card

3. Structure the Card Component: CO5 App

- In card.component.html:

HTML

```
<ion-card>
  <ion-card-header>
    <ion-card-title>Card Title</ion-card-title>
  </ion-card-header>
  <ion-card-content>
    </ion-card-content>
```

```
  <ion-card-footer>
    <ion-button expand="full">Button Text</ion-button>
  </ion-card-footer>
</ion-card>
```
Use code with caution. Learn more
content_copy
4. Style the Card with CSS:
- In card.component.scss:

SCSS
```
ion-card {
  margin: 10px;
  border-radius: 10px;
  box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.15);
}

ion-card-header {
  background-color: #f5f5f5;
  padding: 10px;
}

ion-card-title {
  font-weight: bold;
  font-size: 18px;
}

ion-card-content {
  padding: 15px;
}

ion-button {
  background-color: #007bff; /* Primary Ionic color */
  color: white;
}
```
Use code with caution. Learn more
content_copy
5. Import and Use the Card Component:
- In app.component.html:

HTML
```
<ion-app>
  <ion-content>
    <app-card></app-card>
  </ion-content>
</ion-app>
```
Use code with caution. Learn more
content_copy
6. Run the App:
- Start the development server: ionic serve

This will create a simple Ionic UI with a card, featuring a title, content area, and a full-width button. The card is styled with rounded corners,

a shadow, and a light header. The button has a primary Ionic color.

(a) **You're a developer working on a mobile app for tourists visiting a new city. The app utilizes location awareness to provide helpful information and suggestions based on the user's current location.**
Imagine a tourist, Sarah, is exploring the city and gets lost. Her phone battery is low, and she doesn't have access to Wi-Fi. How can your app utilize location awareness to help Sarah find her way back to her hotel or a safe location?
Consider:
What features can your app offer to help Sarah in this situation?
How can location awareness be used to provide relevant information like nearby landmarks, public transportation options, or emergency services?

Offline Maps and Navigation:

- Downloadable maps of the city that work even without internet access.
- Turn-by-turn navigation using GPS to guide Sarah back to her hotel or a safe location like a police station.
- Points of interest (POIs) like landmarks and ATMs marked on the map for reference.

2. Location-based Information and Assistance:

- Identify Sarah's current location using GPS and display nearby landmarks, streets, and intersections.
- Show walking and cycling routes to nearby safe locations, considering battery level.
- Emergency contact information for local police, ambulance, and tourist helplines.
- Pre-saved offline contact details of the hotel or a local friend/family member.

3. Battery Optimization and Resource Management:

- Display battery level prominently and offer battery-saving tips like reducing screen brightness.
- Prioritize essential features like navigation and emergency information.
- Allow downloading offline maps and POIs in advance to minimize data usage during navigation.

4. Additional Helpful Features:

- Offline translation tools to communicate with locals if needed.
- Currency converter and local payment options information.
- SOS button that sends an alert with Sarah's location to pre-selected emergency contacts.
- Community forum or chat feature for tourists to connect and ask for help.

Location Awareness in Action:

- As Sarah walks, the app continuously updates her location and displays it on the map.
- Nearby landmarks, streets, and POIs are highlighted, helping her orient herself.
- If Sarah is near a bus stop or metro station, the app displays public transportation options with estimated arrival times.

CO4 App

- If her battery is critically low, the app suggests the nearest safe location like a police station or a well-lit area.

(Or)

(b) A popular live streaming app experiences a sudden spike in dropped live streams and audio/video glitches during a high-profile event. Users report buffering, freezes, and complete disconnects. The issue occurs across various device models and network conditions.

Questions:

As a developer, what steps would you take to diagnose the root cause of the live stream disruptions?
How would you design a real-time monitoring system to identify and prevent future performance issues during live events?

Immediate Actions:

1. Gather Data:
    - Analyze server logs for errors, spikes, or drops in activity.
    - Collect real-time data from affected devices on video/audio quality, bitrate, frame rate, and buffering times.
    - Monitor network infrastructure for bottlenecks or outages.
2. Isolate the Problem:
    - Correlate user reports with server data to identify affected regions, platforms, and content providers.
    - Test streaming with different resolutions and bitrates to pinpoint potential limitations.
    - Divide the system into components (encoders, servers, CDN) and test each individually to isolate the failure point.

CO5 App

Deeper Investigation:

1. Server Resources:
    - Check CPU, memory, and network utilization on streaming servers.
    - Identify any resource contention or bottlenecks during the event.
    - Analyze database performance and query execution times.
2. Content Delivery Network (CDN):
    - Monitor CDN edge locations for high load or outages.
    - Investigate cache hit/miss ratios and content distribution across network zones.
    - Evaluate CDN routing efficiency and potential congestion points.
3. Client-side Issues:
    - Analyze app logs on various devices for errors or performance problems.
    - Update and test the app on different platform versions.
    - Collaborate with device manufacturers to identify potential incompatibilities.

Communication and Mitigation:

- Keep users informed: Provide updates on the situation and progress of the investigation.

- Implement temporary measures: Reduce video resolution or bitrate to relieve server load.
- Prepare rollback plans: Be ready to revert to previous configurations if necessary.

Real-time Monitoring System for Future Events

1. Performance Metrics:
    - Track server resource utilization (CPU, memory, network) in real-time.
    - Monitor CDN health: edge server load, cache hit/miss ratios, network latency.
    - Collect client-side data: buffering times, frame rate, video/audio quality metrics.
2. Alerting and Thresholds:
    - Set alerts for critical metrics exceeding defined thresholds.
    - Notify developers on potential resource exhaustion or performance degradation.
    - Prioritize alerts based on severity and potential impact.
3. Dashboard and Visualization:
    - Provide a real-time dashboard for live event monitoring.
    - Visualize key metrics across servers, CDN, and client devices.
    - Enable proactive analysis to identify trends and predict potential issues.
4. Automated Actions:
    - Implement automatic scaling of server resources based on real-time demand.
    - Configure CDN failover mechanisms to redirect traffic in case of outages.
    - Trigger notifications and workflows for faster reaction to critical events.