Assessment: Mobile Application Development for HSE

Chapter 1:

1. How did early mobile phones differ in functionality and hardware from modern smartphones and tablets?
2. What were the main technical and business reasons for the initial success of Symbian as a mobile platform?
3. In what ways did BlackBerry OS shape early concepts of mobile productivity and secure communication?
4. How did Palm OS influence user interface design and interaction patterns on early handheld devices?
5. What key innovations introduced by iOS and Android triggered the shift from traditional mobile phones to smartphones?
6. Why did platforms like Symbian, BlackBerry OS, and Palm OS eventually lose dominance to Android and iOS?
7. How do current mobile ecosystems (Android, iOS, and others) differ in terms of openness, app distribution, and developer experience?
8. What role do app stores and third-party developers play in shaping today's mobile platform landscape?
9. How might emerging technologies such as 5G, AI, AR/VR, and wearables reshape mobile platforms in the next decade?
10. What scenarios can you imagine for the future balance of power between Android, iOS, and potential new platforms?

Chapter 2:

1. Why do ergonomics and hand reach zones matter when designing mobile UI for small screens?
2. How does designing for touch targets differ from designing for mouse input?
3. What are the main functions of a navigation bar in a mobile app, and what makes it intuitive?
4. How should toolbars and menus be adapted for small screens to avoid overload?
5. What are key principles for designing buttons that are both discoverable and accessible on mobile?
6. How do cross-platform design considerations influence the choice of navigation patterns (tabs, drawer, bottom nav)?
7. In what ways do Material Design guidelines shape layout, elevation, and motion on Android?
8. How do Human Interface Guidelines (HIG) influence typography, spacing, and iconography on iOS?
9. What are the main differences between Material Design and HIG that a designer must respect in a cross-platform app?
10. How can a designer ensure consistent brand identity while still following platform-specific guidelines?

Chapter 3:

1. Explain in your own words the difference between a class and an object, and give a mobile-app example of each (Android or iOS).
2. Describe encapsulation and show how it helps you evolve a mobile app's code without breaking other parts of the system.
3. Give a concrete example of inheritance from a mobile domain (e.g., screens, users, UI elements) and explain why inheritance is useful there.
4. Describe polymorphism and give an example with a list of objects in a mobile app where calling the same method leads to different behaviors.
5. Compare static typing (e.g., Java, Swift) and dynamic typing (e.g., JavaScript, Python) in the context of mobile development: give two pros and two cons for each.
6. Pick any one SOLID principle and explain how violating it might make a mobile app feature harder to maintain or test.
7. Choose one GRASP pattern (e.g., Controller, Information Expert, Low Coupling) and give a specific example of how you would apply it in an Android or iOS architecture.

8. Define refactoring and explain how you would safely refactor a “God Activity/ViewController” in a real mobile project without changing external behavior.
9. Give two examples of optimization practices in mobile apps (performance or memory) and explain what risks appear if you over-optimize too early.
10. You inherit a legacy mobile project with poor architecture. Describe a step-by-step plan (3–5 steps) using OOP concepts, SOLID/GRASP, and refactoring to gradually improve the codebase while keeping the app in production.

Chapter 4:

1. Briefly describe MVC, MVP, MVVM and explain how responsibility for *UI logic* differs between them.
2. For a medium-size mobile app, compare MVC vs MVP: give one concrete advantage and one drawback of each in terms of testability and code readability.
3. Explain how data binding in MVVM can both improve productivity and create potential debugging problems in mobile apps.
4. A team complains about “Massive ViewControllers/Activities”. Which pattern (MVC/MVP/MVVM) would you recommend to reduce this problem, and why?
5. Describe the core idea of microservice architecture and explain how it can improve scalability for a backend serving millions of mobile users.
6. Give two drawbacks of microservices for a small startup building its first mobile product.
7. Explain the Singleton pattern, give one realistic mobile-backend or mobile-client example, and discuss one risk of overusing it.
8. Describe the Factory Method pattern and show how it could help when supporting multiple API clients (e.g., Android, iOS, Web) in a shared backend.
9. Explain the Decorator pattern and provide an example of adding cross-cutting behavior (e.g., logging, caching, or rate limiting) in a mobile backend without modifying existing code.
10. Given a large mobile system, outline how you would combine an architectural pattern (MVC/MVP/MVVM), microservices, and at least one GoF pattern (Singleton/Factory/Decorator) into a coherent design, and name one trade-off of your choice.

Chapter 5:

1. Explain the role of an Activity in an Android app and describe its basic lifecycle in the correct order.
2. What problems do Fragments solve compared to using only Activities, and how does their lifecycle differ in at least one key point?
3. Compare explicit and implicit Intents and give one concrete example of each in a real Android scenario.
4. What is an Android Service, and how does it differ from an Activity in terms of purpose and user interaction?
5. Describe what a BroadcastReceiver is and give one example where it would be appropriate to use it in a mobile application.
6. Why is understanding the lifecycle of Activities and Fragments critical for handling configuration changes (e.g., rotation) and avoiding memory leaks?
7. Briefly describe the roles of Android Studio, Gradle, and Android SDK in the Android development workflow.
8. Define what an APK is, and explain the steps from source code to a signed APK ready for distribution.
9. Outline the main stages of publishing an Android app to Google Play (from build to listing), and mention one common pitfall.
10. Given an app that plays music in the background, receives push notifications, and opens links from other apps, explain how you would combine Activities, Services, BroadcastReceivers, and Intents to implement this behavior.

Chapter 6:

1. List the basic data types in Swift (at least four) and give a short example of when you would use each in an iOS app.

2. Write a simple Swift class Profile with two properties and one method, and explain how this demonstrates object-oriented programming.
3. Explain the difference between a class and a struct in Swift, and when you would prefer a struct in an iOS project.
4. What is a UIViewController, and what are its main responsibilities in an iOS application?
5. Describe what a Storyboard is and how it relates to view controllers and navigation in an iOS app.
6. Explain the purpose of Auto Layout constraints and give two examples of how wrong constraints can break app behavior on different screen sizes.
7. Briefly describe the lifecycle of an iOS app, naming the key stages from launch to termination and the role of the App Delegate.
8. How do windows and root view controllers fit into the iOS application architecture? Describe their relationship.
9. What are native modules (e.g., camera, location, notifications) and how can they be integrated into a Swift-based iOS app at a high level?
10. Given a simple iOS app with a login screen and a home screen, explain how Swift classes, view controllers, storyboards, constraints, and the app lifecycle work together during app launch and navigation.

Chapter 7:

1. Define cross-platform mobile development in your own words and explain why companies might choose it over fully native development.
2. Name the primary technologies behind React Native, Flutter, and Xamarin (language + core framework idea).
3. Compare React Native and Flutter in terms of UI rendering approach and discuss one consequence for performance or look-and-feel.
4. Explain one performance advantage and one performance drawback of cross-platform frameworks compared to native apps.
5. What does “responsive cross-platform interface” mean, and why is it harder to achieve when targeting both Android and iOS with a single codebase?
6. Give an example where you would need to integrate a native component (Android/iOS specific) inside a cross-platform app and describe the challenge briefly.
7. Describe how hot reload / fast refresh in frameworks like Flutter or React Native affects developer productivity and possible risks.
8. When might a team still prefer Xamarin or .NET MAUI instead of React Native/Flutter? Mention at least two reasons.
9. Explain the trade-off between code reuse and platform-specific UX when designing cross-platform interfaces.
10. A startup wants: high performance UI, fast development, and access to native APIs. Which cross-platform solution would you recommend (React Native, Flutter, or Xamarin), and why?

Chapter 8:

1. What is profiling in the context of mobile apps, and why is it more critical on mobile than on desktop?
2. Name at least two profiling tools (Android or iOS) and explain what kind of data each helps you measure.
3. Describe a simple methodology for profiling an app’s startup time and identifying the main bottleneck.
4. How can you detect and reduce excessive memory usage in a mobile app? Mention at least one technique or tool.
5. Explain why power consumption is important for mobile apps and give two coding practices that can reduce battery drain.
6. How do heavy graphics rendering and complex animations affect performance, and what are two strategies to optimize them?
7. Describe a situation where background processing is appropriate and how misusing it can harm responsiveness or battery.
8. What is a UI thread/main thread, and why must you be careful with multithreaded execution in mobile apps?

9. Give an example of a race condition or deadlock risk in a multithreaded mobile app and how you would avoid it.
10. Outline a short optimization workflow: from detecting a performance problem with profiling to verifying that your change actually improved speed or responsiveness.

Chapter 9:

1. Explain the difference between unit testing and integration testing in the context of a mobile app, and give one example of each.
2. Why are unit tests important for long-term maintenance of Android/iOS applications? Mention at least two benefits.
3. Briefly describe what Espresso, XCTest, and Detox are used for and which platforms they target.
4. Give an example of a simple UI test scenario you could automate with Espresso or XCTest.
5. Outline the main steps to prepare an Android app for release via Google Play Console (from signed build to publishing).
6. Outline the main steps to prepare an iOS app for release via App Store Connect (from archive to TestFlight/production).
7. What are compliance rules and policies in app stores, and why must developers pay attention to them from the design phase?
8. Give two examples of common policy violations that can cause an app to be rejected from Google Play or App Store.
9. How can automated testing (unit + UI tests) help reduce the risk of rejection or user complaints after publishing?
10. Describe a simple workflow that combines unit tests, UI tests, and the release process so that each new version of the app is tested before submission to the store.

Chapter 10:

1. In your own words, what is the difference between a classical algorithm, a machine learning model, and a neural network in a mobile app context?
2. Give two concrete mobiles use cases where AI-driven personalization improves user experience (e.g., recommendations, UI adaptation) and explain why.
3. What is a recommendation engine, and how could it be applied inside a mobile app you use every day?
4. Explain at a high level how a neural network model can be executed on-device (without server) and mention one advantage and one drawback of this approach.
5. Define a multi-agent system and give an example of how multiple agents might cooperate inside (or around) a mobile app.
6. Describe one realistic scenario where intelligent agents support collaboration between users (e.g., group planning, task coordination) in a mobile application.
7. Compare on-device AI inference vs server-side inference for mobile apps: give at least one trade-off in terms of latency, privacy, and resource usage.
8. What are two main challenges when integrating AI/ML models (neural networks, recommenders) into mobile apps regarding performance and power consumption?
9. Outline the main steps needed to integrate a pretrained ML model (e.g., TensorFlow Lite or Core ML) into a mobile app and use it for predictions.
10. A mobile app wants to suggest actions to users (e.g., “reply quick”, “call later”, “bookmark”). Propose an AI-based solution that combines a recommendation engine and intelligent agents, and describe one potential ethical or UX risk.

Chapter 11:

1. In your own words, what is threat modeling for a mobile app, and what are the main questions you would ask during this process?
2. Give three secure coding best practices that are especially important in mobile development (Android/iOS) and briefly explain why.
3. Explain the difference between encryption in transit and encryption at rest in mobile apps, and give one example of each.
4. What is the purpose of authentication protocols (e.g., OAuth2, OpenID Connect) in mobile apps, and why is it dangerous to “roll your own” auth?

5. Describe at least two secure options for storing sensitive data (tokens, passwords) on Android or iOS, and one insecure practice to avoid.
6. Briefly explain how a mobile app can still be vulnerable to SQL injection or XSS, even if it uses an API/backend server.
7. What is reverse engineering of mobile apps, and name two techniques to make reverse engineering harder for attackers.
8. What is OWASP Mobile Security Testing Guide (MSTG) and how can a development team use it in a real project?
9. Give two examples of common policy or compliance requirements that affect how you handle user data in a mobile app (e.g., GDPR, store rules).
10. Propose a security checklist (4–5 points) you would apply before releasing a mobile app to production, based on the concepts above.