##P/C – Spike: Learning Reflection on Mobile Application Development

Goals

The goal of this spike task was to reflect on my learning throughout the mobile application development unit. This unit provided both theoretical foundations and hands-on experience with developing for mobile platforms, particularly Android. My focus was on understanding the unique characteristics of mobile environments, mastering tools and techniques suited for these constraints, and developing applications that are functional, responsive, and efficient.

Thanks to the incredibly structured and engaging teaching approach of **Dr. Khoa**, the unit felt highly accessible and rewarding. His practical demonstrations, clear explanations, and real-world examples made complex concepts—like activity lifecycle management, memory optimization, asynchronous processing, and user-centric UI design—much easier to grasp. He always encouraged exploration beyond the textbook, which made a huge difference in my learning.

The balance of lectures, labs, and team discussions gave me the opportunity to build confidence through hands-on experience. Rather than just memorizing concepts, I was building, testing, breaking, and fixing real mobile apps. This process cemented the knowledge in a much deeper way.

Specific Knowledge Areas Covered

- Understanding mobile system constraints compared to traditional platforms
- Designing applications with limited resources and varying screen sizes in mind
- Developing and debugging mobile apps using Android Studio
- Incorporating lifecycle-aware architecture components (e.g., ViewModel, LiveData)
- Implementing asynchronous operations with Kotlin coroutines
- Conducting proper testing using JUnit and Espresso frameworks
- Structuring apps using Jetpack Navigation and clean architecture principles

---

Tools and Resources Used

- **Android Studio** – Primary IDE for development
- **Kotlin** – Used for writing clean, concise, and expressive application logic
- **Jetpack Libraries** – ViewModel, LiveData, Navigation for responsive and maintainable apps
- **Espresso & JUnit** – For thorough UI and unit testing
- **Android Emulator** – To test across different screen sizes and configurations
- **Logcat & Profiler** – For runtime debugging and performance optimization
- **Official Android Developer Docs** – My go-to reference for best practices
- **Online tutorials** – Especially for Kotlin coroutines and Jetpack Compose

---

Knowledge Gaps and Solutions

Gap 1: Understanding Mobile vs Traditional Systems

At the beginning of the unit, I had little appreciation for how mobile systems differ from desktop or web environments. Through Dr. Khoa's insightful lectures and examples, I learned that mobile apps must deal with power constraints, limited memory, and small screen real estate. To manage these, I adopted strategies like avoiding heavy background processing and optimizing UI updates.

Example:
I used Kotlin coroutines to handle network operations asynchronously, ensuring smooth UI performance even under resource constraints. In desktop apps, this level of optimization isn't always necessary due to more powerful hardware.

Gap 2: Designing for Mobile Constraints

Creating UIs that adapt to various screen sizes and resolutions was initially difficult. Thanks to practical labs and sample projects provided by Dr. Khoa, I became comfortable using ConstraintLayout and density-independent pixels (dp) to build responsive interfaces.

Example:
In our book list project, I designed a layout that worked on both phones and tablets, tested across multiple emulators, and adjusted element positions based on screen orientation. This ensured usability regardless of device.

Gap 3: Testing and Debugging Graphical Applications

Testing UI interactions and ensuring app stability was challenging, especially with lifecycle events. I learned to use JUnit for logic testing and Espresso for UI automation. Debugging with Logcat was invaluable for tracking crashes caused by improper state handling.

Example:
When my app crashed during screen rotations, I traced the issue to direct use of activity state without saving it properly. I refactored the app to use ViewModel, resolving the issue and improving app resilience.

Open Issues and Recommendations

- **Jetpack Compose:** I only scratched the surface of this exciting new UI toolkit.
  Recommendation: I plan to rebuild one of my past projects using Compose to learn declarative UI design.
- **Security & Permissions:** This unit didn't cover mobile security in depth.
  Recommendation: I want to study Android's permission system, secure data storage, and best practices to avoid vulnerabilities.

- **Push Notifications and Location Services:**
  Recommendation: I aim to build a personal app that integrates Firebase Cloud Messaging and GPS-based features to better understand background processing and user engagement.

Key Takeaway

This unit fundamentally changed how I approach software development. Mobile development isn't just "desktop, but smaller." It's a completely different paradigm—more dynamic, more user-focused, and much more constrained in terms of resources. I learned how to think like a mobile developer: balancing features with performance, battery life, and device limitations.

Dr. Khoa's teaching was one of the biggest highlights. His passion for the subject, his deep understanding, and his supportive nature made the classroom an exciting place to be. He didn't just teach us how to code—he taught us how to solve real-world problems with empathy and precision. Because of this unit, I now feel more confident not only in mobile app development, but in learning and adapting to new platforms in general.

References (APA 7)

Abrahamsson, P., Budi, N., & Warsta, J. (2009). *Mobile application development: Challenges and research agenda*. In *2009 ICSE Workshop on Software Engineering for Mobile Application Development* (pp. 29–34). IEEE. https://doi.org/10.1109/SEFM.2009.30

Android Developers. (2023). *Build a responsive UI with ConstraintLayout*. https://developer.android.com/training/constraint-layout

Google. (2023). *App architecture guide*. Android Developers. https://developer.android.com/jetpack/guide

Google. (2023). *Kotlin coroutines on Android*. https://developer.android.com/kotlin/coroutines