Artifact Title	InventoryAppAndroid
Author	April Nixon
Creation Date	October 2024

Introduction

This document discusses the enhancements made to a warehouse inventory management app written in Android and Kotlin. The artifact initially ran basic operations for infrastructure to manage items in a warehouse, but it did not scale and was not modular. The updated version aims at appending database management with SQLite, refining the User Interface, and creating a modular code design. Besides fixing the shortcoming of the original version, it also takes care of better error handling and user authentication. The changes here represent software development principles which have been understood and show technical growth.

Description of the Artifact

The Inventory App is a mobile app written in Kotlin with sign-in and sign-up features as well as the ability to add warehouse items to the app using an interactive interface. Users can add, update, and delete items with attributes like quantity, price with inventory management as core features. Data has persisted locally by SQLite, and the list of items is displayed dynamically with RecyclerView using the adapter. Permissions are also integrated so that when your inventory requires urgency, users will be notified via SMS. The original artifact was less complicated. It did not have database support, followed no modular design principles, and all the logic was a single activity. That made the code hard to maintain and scale. Also, the user interface was not

minimal, and did not follow Android best practices. On the other hand, the version with enhanced UI design included modular code organization, and a database driven structure, which makes the application more efficient and user friendly.

Justification for Inclusion in the ePortfolio

The Inventory App was the enhanced version that shows my capability to build and release modular scalable applications. This project is meant to demonstrate the use of some important fundamentals related to showing how projects are built as a software product, including secure authentication, managing databases, and using common practices of building reusable code. My understanding of modular architecture is with the separation of responsibilities across multiple activities. The database backend of SQLite indicates how I can handle persistent data well. Secondly, user authentication and input validation are applied to make sure that the app is safe and secure. I demonstrated how to use RecyclerView adapters to manage dynamic lists handling changing data in a fast and efficient manner. The application of permissions management shows my understanding of Android's runtime permissions model. This artifact is a case study in which I can show my ability to apply theoretical knowledge to practical applications as a strong candidate artifact to put in my ePortfolio.

Enhancements Made

To overcome the limitations of the original artifact, several enhancements were made. The first major improvement was to integrate SQLite, which replaced temporary data storage with a permanent database solution. These changes were made so that data is not lost between sessions, and it helps with better handling of CRUD operations. A DatabaseHelper class was added to add

encapsulation of the database interaction which will clean up the code and help to maintain the app. Modular architecture was another key improvement. User authentication, permissions management and inventory operations were distributed across multiple activities via logic to increase readability and scalability. I applied RecyclerView with adapters to efficiently manage dynamic lists, showing my knowledge of Android data structures and it has an implication to the way I think about data structures when dealing with Android development. I used CardView elements for Buttons to make sure the buttons look better, having elevation and rounded corners to give it a more modern and polished feel. This change makes the app match the app's visual style with current UI trends and makes it easier for users to use the app. The error handling was also improved to handle permission requests, user input, and to prevent the app from being unstable in all these scenarios. Industry best practices were used to show the enhancements that were made, and it is an aspect that shows that I have been able to write clean and maintainable code while also incorporating them. They also show how I can use theoretical concepts like algorithms and data structures and use them for real world software issues.

Reflection on the Enhancement Process

This project helped me understand the importance of modularity, resource management, and error handling. It also showed the need for well-structured code in persistent data application. The separation of database logic from the user interface is also something it taught me to do, so that the app can stay scalable and maintainable. Applying dynamic structures like RecyclerView adapters for managing product lists gave me hands-on experience with them. This showed how efficiently data should be handled for frequent updates or big datasets. Authenticating the valid user was a good example to reinforce security and error handling. During my work with

Android's runtime permissions model, I encountered a few difficulties that I learned how to handle. Such as using permissions dynamically as well as dealing with error situations. Also, UI design helped me grasp crafting user friendly interfaces that are both functional and aesthetically pleasing. In conjunction, this project taught me how small design choices, like using CardView for buttons and having little spacing between your layouts can have a huge impact on the user experience. Bottom line, the process of improving on this artifact gave me practical experience in real world software development and helped me understand what I needed to do to successfully apply algorithms and data structures.

Conclusion

This Enhanced Inventory App shows how I can use data structures, algorithms, and software engineering principles to build a real-world mobile application. It shows my growth as a software developer through making the improvements that I have made in the code I have written. This project shows how I am good at handling errors, handling users, user authentication, and design of UI. Using this artifact in my ePortfolio would show my technical skills and my ability to translate a theoretical idea into a practical application. This also shows that I can work out complex problems using industry standards and best practices. In conclusion, this artifact shows my ability to develop reliable user-friendly software.