The artifact I chose for enhancement is my Event Tracking App, which I originally developed for CS 360: Mobile Architecture and Programming. This app allows users to create, view, and manage events while receiving notifications and SMS reminders for upcoming events. The original version of the app used an SQLite database with raw SQL queries, and its architecture followed a simple Activity-based approach without ViewModels or a proper separation of concerns.

I included this artifact in my ePortfolio because it highlights my experience in Android development, particularly with Java and SQLite. The app involves user authentication, data persistence, background task execution, and notifications, making it a strong example of my ability to build mobile applications. The enhancements I made focus on improving maintainability, following best practices, and adopting modern Android development techniques.

One of the biggest improvements was moving from raw SQLite queries to Room with ViewModel and LiveData. The original code relied on direct database access within activities, which made it harder to maintain and scale. By implementing Room, I eliminated the need for raw SQL queries and instead used DAO interfaces to handle database operations. The ViewModel allowed me to manage UI-related data efficiently, ensuring that the app remained responsive and state changes persisted across configuration changes. LiveData helped keep the UI in sync with data changes without requiring manual updates.

Another major enhancement was improving security and data integrity. Originally, passwords were stored in plain text, which was a serious security flaw. I implemented password hashing using SHA-256 before storing them in the database. The database also lacked foreign key constraints, meaning orphaned event records could exist when a user was deleted. To fix this, I enabled foreign key constraints in Room and added indexing to frequently queried columns like the username field, improving both security and performance.

Defensive programming was another key focus. The original implementation did not properly handle permissions for sending notifications and SMS. The app would attempt to send notifications without checking if the required permissions were granted, which could lead to crashes. I updated the code to dynamically request permissions at runtime and wrapped SMS and notification calls in try-catch blocks to handle errors gracefully. I also made sure to log errors to help with debugging. Additionally, I updated AlarmManager calls to check for exact alarm permissions, which ensures compliance with newer Android security policies.

These enhancements helped me demonstrate several course outcomes. Implementing Room and ViewModel improved software architecture by aligning with modern Android development practices. The defensive programming updates ensured that the app handled potential failures properly, improving its reliability. Adding secure password storage and better permission handling strengthened my understanding of security considerations in mobile app development.

Throughout this process, I gained a better understanding of how to build scalable and maintainable applications. Switching to Room and ViewModel not only cleaned up the code but also made future modifications easier. The security improvements made me more aware of real-world vulnerabilities, and debugging permission issues taught me the importance of planning for edge cases.

Looking at the final result, my project has transformed from a simple event tracking app into a more structured and secure application. The changes I made improve performance, security, and maintainability, making this a strong addition to my ePortfolio. This experience has reinforced my skills in software engineering and mobile development, and I feel confident that these enhancements have made my app more practical for real-world use.