Artifact 1 Enhancements: Software Engineering and Design – CS360 Inventory Application

For my ePortfolio, I have selected an inventory management Android application as a key artifact. This application was developed during my Mobile Application Development course in the third year of my Computer Science program. It serves as a comprehensive demonstration of my skills in Android app development, user interface design, database management, and software engineering principles.

The inventory management app is designed to allow users to efficiently manage inventory items. Its core functionalities include adding new items, updating existing ones, deleting items, and viewing the current inventory list. The application was built using Java and the Android SDK, with SQLite serving as the backbone for database functionality.

I chose to include this artifact in my ePortfolio because it showcases several key components of my skills and abilities in software development:

1. Android App Development: The project demonstrates my proficiency in creating mobile applications for the Android platform.

2. Database Management: It exhibits my ability to implement and manage SQLite databases within a mobile application context.

3. User Authentication: Through enhancements, I've demonstrated my skills in implementing secure user authentication systems.

The original version of the app showcased several important skills:

1. Android App Development: I designed and implemented the user interface, navigation flow, and interaction logic to create a seamless user experience.

2. Database Management: Leveraging SQLite, I implemented database functionality to store and manage inventory data. This involved creating a database schema, performing CRUD operations, and ensuring data consistency.

3. Software Engineering Principles: Throughout the development process, I adhered to software engineering best practices such as modularization, code reusability, and documentation. This ensured the maintainability and scalability of the inventory management app.

To further demonstrate my growing skills and address potential real-world needs, I implemented several enhancements to the original application. These enhancements involved modifying existing files and creating new ones:

1. Role-Based Navigation: Users are now directed to different screens based on their roles (Admin, Manager, or User) after logging in. This enhancement involved:

- DatabaseHelper.java: Updated the database schema to include a role column in the users table.

- LoginFragment.java: Modified the login functionality to check user roles and navigate accordingly.

Functionality:
Managers are guided to the Manager Fragment, serving as a welcome screen enabling navigation to either the metrics dashboard or the Inventory Fragment. Within the Inventory Fragment, managers possess the capabilities to create, update, and delete inventory items, whereas users with different roles are limited to adding data only. This functionality is illustrated in Figures 1-3 and 1-4.

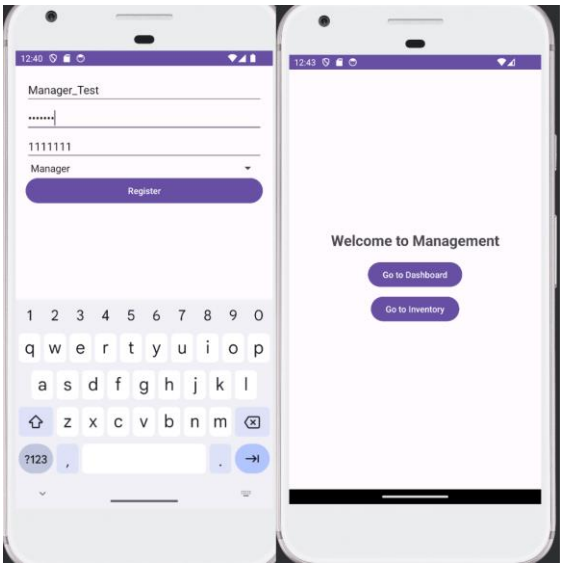


Figure 1-1 depicts the Registration Fragment with the added user role feature. After logging in and verifying that their role is that of a manager, users will be directed to the Manager Fragment, illustrated in Figure 1-2.

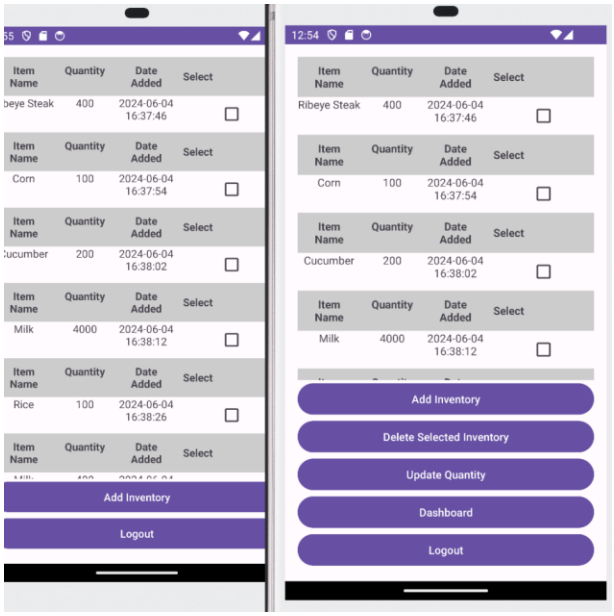


Figure 1-3 depicts the Inventory Fragment when logged in as "User". Figure 1-4 depicts the Inventory Fragment when logged in as "Manager".

2. Password Hashing: Passwords are now hashed using SHA-256 before storing them in the database, significantly enhancing security. This improvement included:

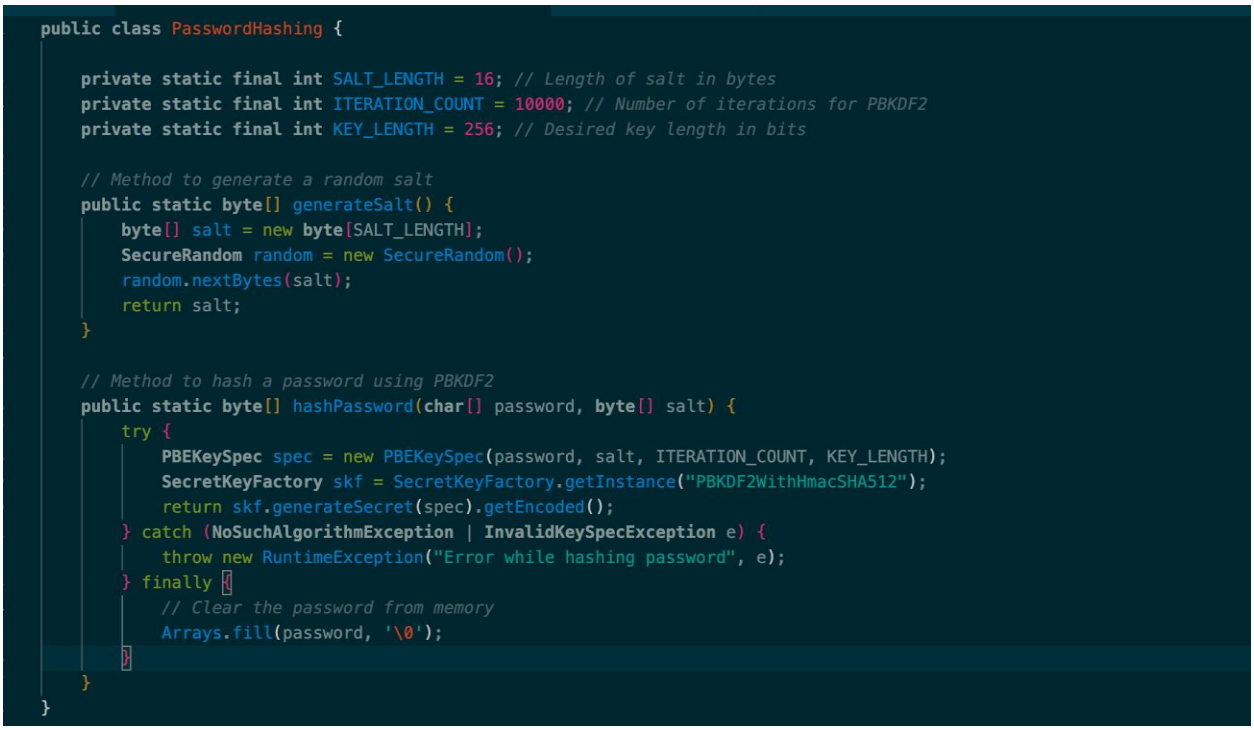
- PasswordHashing.java: Created a new class to handle password hashing and salt generation.

- RegisterFragment.java: Modified the user registration process to hash passwords before storage.

Functionality:

This code helps keep passwords safe and secure. It does this by using a technique called PBKDF2, which is like a super-strong lock for passwords. Here's how it works:

First, it creates a random "salt" for each password. Think of salt as a unique ingredient added to each dish. This makes it much harder for someone trying to break in to guess the password, even if they have a big list of common passwords.



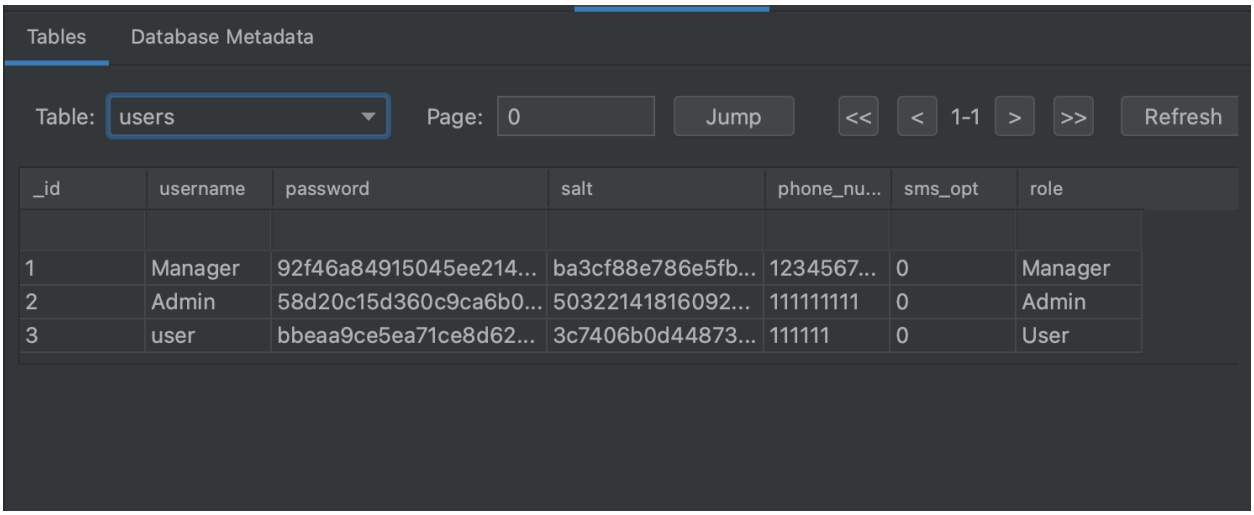
```java
public class PasswordHashing {

    private static final int SALT_LENGTH = 16; // Length of salt in bytes
    private static final int ITERATION_COUNT = 10000; // Number of iterations for PBKDF2
    private static final int KEY_LENGTH = 256; // Desired key length in bits

    // Method to generate a random salt
    public static byte[] generateSalt() {
        byte[] salt = new byte[SALT_LENGTH];
        SecureRandom random = new SecureRandom();
        random.nextBytes(salt);
        return salt;
    }

    // Method to hash a password using PBKDF2
    public static byte[] hashPassword(char[] password, byte[] salt) {
        try {
            PBEKeySpec spec = new PBEKeySpec(password, salt, ITERATION_COUNT, KEY_LENGTH);
            SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
            return skf.generateSecret(spec).getEncoded();
        } catch (NoSuchAlgorithmException | InvalidKeySpecException e) {
            throw new RuntimeException("Error while hashing password", e);
        } finally {
            // Clear the password from memory
            Arrays.fill(password, '\0');
        }
    }
}
```

Then, when someone wants to log in, their password gets turned into a special code using PBKDF2. This code is like a secret handshake that only the website knows. PBKDF2 makes it hard for someone to figure out the original password, even if they have the code.



The code also makes sure to clean up after itself by erasing any traces of the password from memory once it's done, like wiping away fingerprints. This helps keep things even safer.

These enhancements align closely with the course objectives outlined in Module One:

1. Developing professional-quality software solutions: The optimization of database queries and enhanced error handling demonstrate my ability to improve efficiency and reliability in software applications.

2. Designing and implementing computing solutions that solve real-world problems: The addition of user authentication, password hashing, and role-based navigation showcases my ability to address real-world security and usability concerns while managing design trade-offs.

The enhancement process presented several challenges, primarily in optimizing database queries to improve performance without compromising data integrity. Through research and experimentation, I gained a deeper understanding of database indexing and query optimization techniques. This experience highlighted the importance of balancing performance improvements with data security and integrity.

One specific challenge was implementing the role-based navigation system. This required careful consideration of how to modify the existing login process (in LoginFragment.java) to accommodate different user roles without disrupting the current functionality. It also necessitated changes to the MainActivity.java file to handle the loading of different fragments based on user roles, which required a thorough understanding of Android's fragment management system.

Another significant challenge was implementing secure password hashing. Creating the PasswordHashing.java class required research into best practices for password security, including the use of salts and choosing an appropriate hashing algorithm. Integrating this new functionality into the existing registration process (RegisterFragment.java) required careful testing to ensure that user accounts could still be created and accessed securely.

The process of enhancing and modifying the inventory management app provided valuable insights into various aspects of software development. It demonstrated the importance of continuous learning and adaptation in the field. By actively seeking opportunities for improvement, I was able to enhance the app's functionality and security while further developing my skills as a mobile app developer.

This project, from its initial development to the implemented enhancements, showcases my ability to apply theoretical knowledge to practical, real-world solutions. The modifications made to existing files (DatabaseHelper.java, LoginFragment.java, MainActivity.java, RegisterFragment.java) and the creation of new ones (PasswordHashing.java) reflect my growth as a developer and my commitment to creating efficient, secure, and user-friendly software applications. As I continue in my career, the skills and insights gained from this project will undoubtedly prove invaluable in tackling future software development challenges.