CS-499 Computer Science Capstone

Module 4-2 Milestone Three: Enhancement Two: Algorithm and Data Structures

Southern New Hampshire University

Stephen Owusu-Agyekum

March 30, 2024

Purpose

The following narrative emphasizes improvements to Algorithm and Data structure artifacts and explains how the methodologies behind implementing algorithms and data structures work. It explains why the artifact was chosen for this category of my ePortfolio and offers insights into the methods used to create it. Additionally, it delves into the artifact's significance within the Algorithm and data structure, showcasing its relevance and evident accomplishment of the course outcomes or objectives.

Prompt

The artifact I selected for this enhancement under the Algorithm and Data Structure category is "Inventory Mobile App" from the computer science course, CS-360: Mobile Architecture and Programming. I took this course in 2023, during the January – February term.

The application is built using the Java programming language and linked with the relational database SQLite. The development and programming were facilitated through the utilization of the Android Studio IDE. Also, comprehensive testing, such as unit tests and application execution, was conducted within the Android Studio environment to ensure compatibility and performance across various Android platforms. The project aims to develop a mobile app that can track inventories of items in warehouses or shops and can be accessed through Android mobile devices.

The chosen artifact revolves around Algorithms and Data Structures, encompassing a comprehensive process of algorithm design and implementation. It entails considerations regarding the interaction of users with various algorithms and data structures implemented within the application. Emphasis is placed on ensuring intuitive utilization of the algorithms and their functionalities, incorporating industry-standard symbols and icons for a seamless user experience.

The artifact, which focuses on Algorithm and Data Structures, entails putting engineering and design techniques into practice for data input validation, plan for security, and take an automatic refuse access. Additionally, it involves designing algorithms and data structures to maintain functionality and linkages between various classes, functions, and database interactions inside the app's layouts and code base. It showcases the practical implementation and utilization of specific algorithms, such as searching, string comparison, and database interaction, which are fundamental for Android app development. It implements algorithms for performing operations like Create, Read, Update, and Delete (CRUD) on the SQLite database. These operations involve efficient algorithms for inserting, querying, updating, and deleting records from the database.

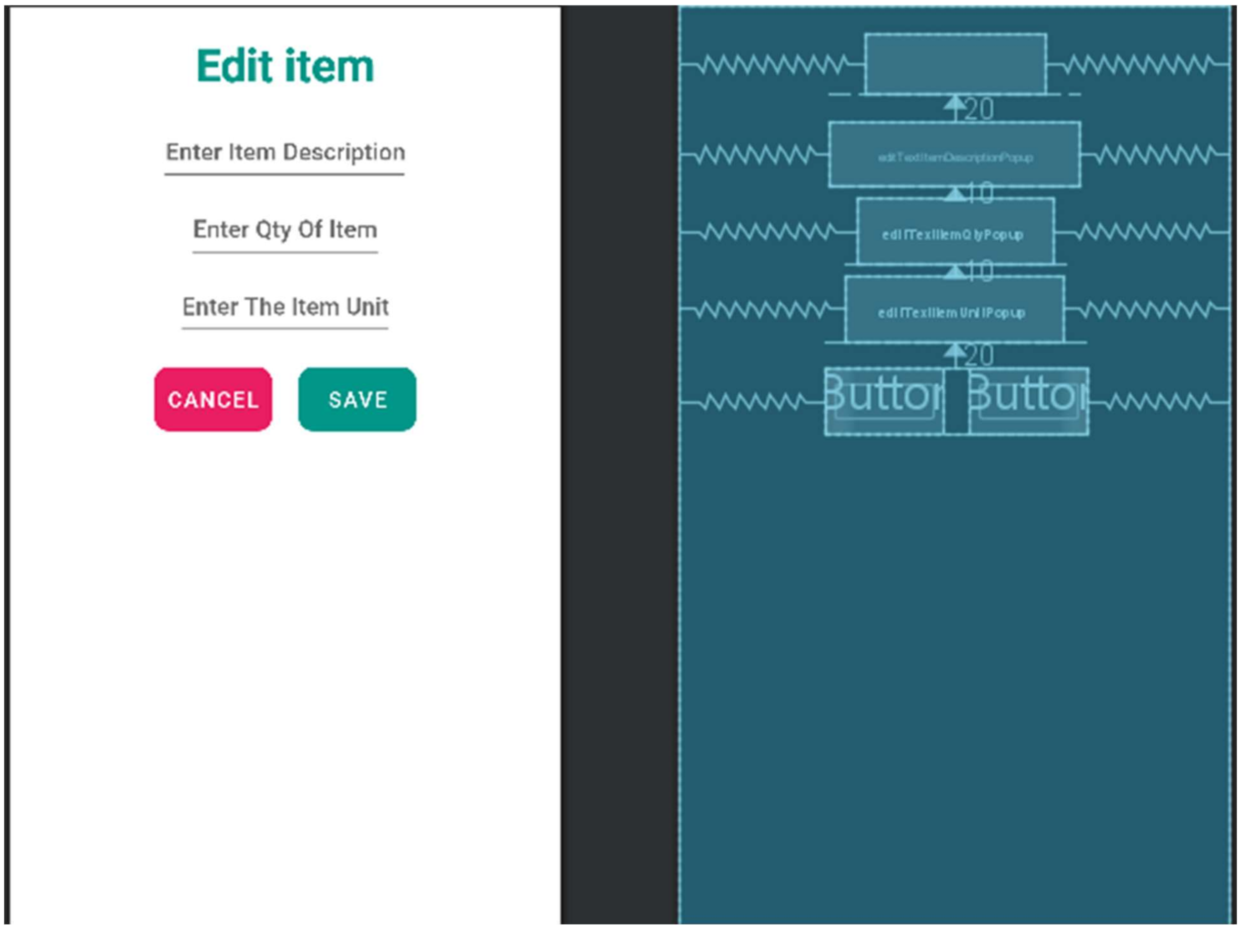
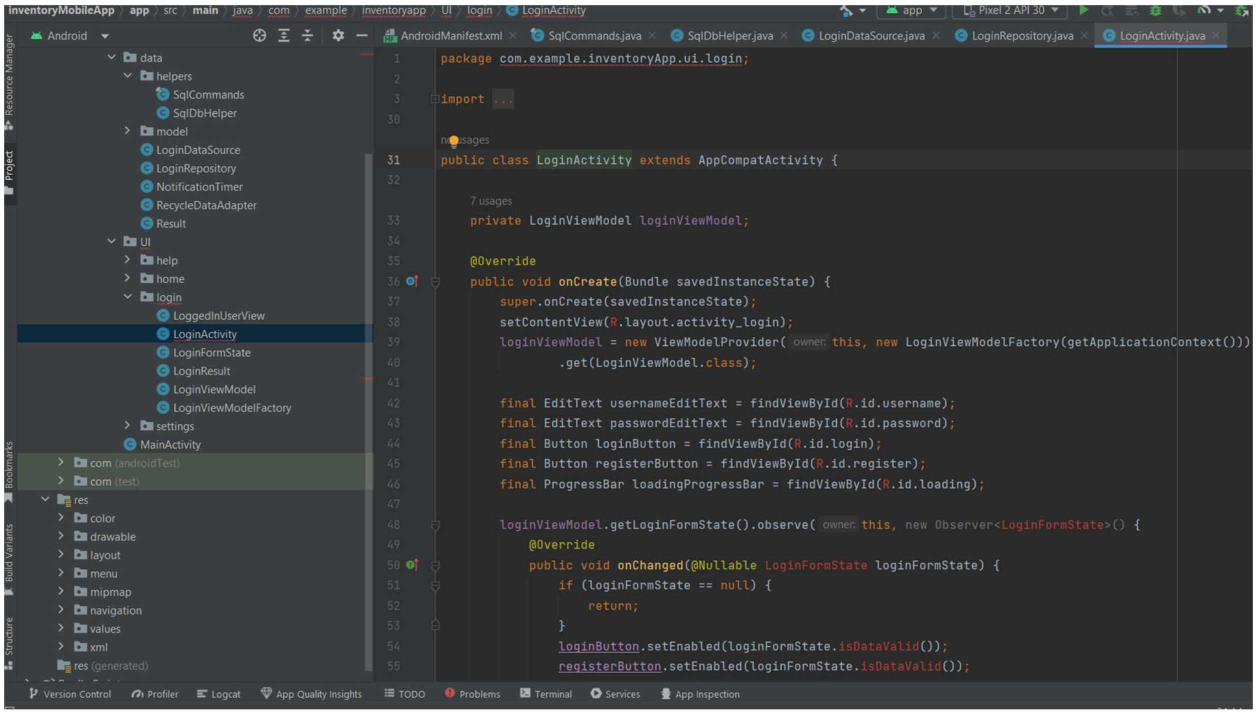


Figure 1: Layout of the item editing design

With this layout, a user can add new items to the inventory record, update or edit the item quantity, and its unit.

The original artifact lacks comprehensive validation checks, such as ensuring valid input data before proceeding with database operations, leaving potential vulnerabilities and prone to errors. Also, the original artifact lacks proper comments that explain the purpose and functionality of various methods and components, making it challenging for developers to understand and maintain the codebase. Additionally, the original code lacks proper user feedback mechanisms in case of errors or successful operations, which could lead to poor user experience, as seen in the SettingsFragment class, where feedback is not provided after data deletion. Moreover, the original code does not incorporate robust error handling mechanisms, potentially leading to unexpected crashes or undesired behavior during runtime.



```java
package com.example.inventoryApp.ui.login;

import ...

no usages
public class LoginActivity extends AppCompatActivity {

    7 usages
    private LoginViewModel loginViewModel;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        loginViewModel = new ViewModelProvider( owner: this, new LoginViewModelFactory(getApplicationContext()))
                .get(LoginViewModel.class);

        final EditText usernameEditText = findViewById(R.id.username);
        final EditText passwordEditText = findViewById(R.id.password);
        final Button loginButton = findViewById(R.id.login);
        final Button registerButton = findViewById(R.id.register);
        final ProgressBar loadingProgressBar = findViewById(R.id.loading);

        loginViewModel.getLoginFormState().observe( owner: this, new Observer<LoginFormState>() {
            @Override
            public void onChanged(@Nullable LoginFormState loginFormState) {
                if (loginFormState == null) {
                    return;
                }
                loginButton.setEnabled(loginFormState.isDataValid());
                registerButton.setEnabled(loginFormState.isDataValid());
```

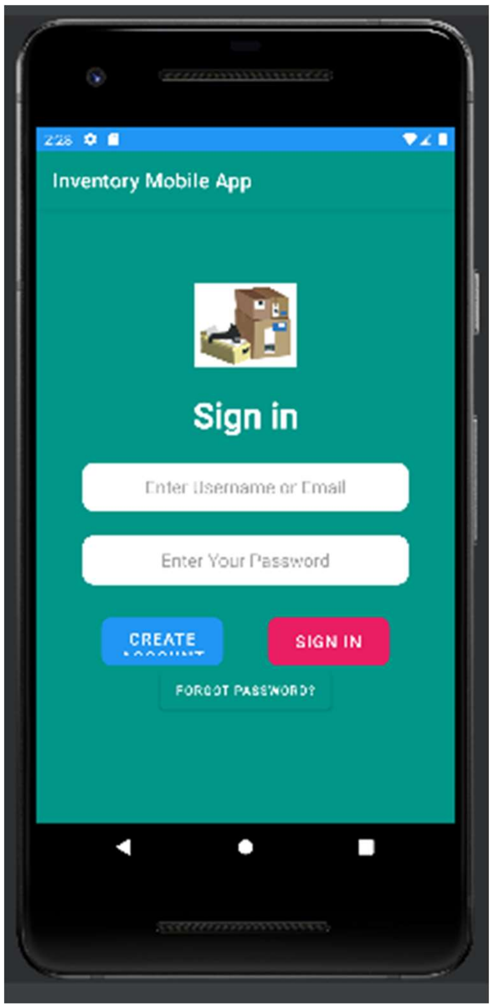
Figure 2: ***Original artifact codebase.***

During the enhancement process of designing and developing the mobile app, I showcased my skills in algorithms and data structures by carefully crafting the software. This involved meticulously integrating user interface elements with the underlying algorithmic logic to ensure a seamless user experience. For example, within the RegisterActivity class, the CheckEditTextNotEmpty() method demonstrates the careful integration of user interface elements with the core algorithmic logic. Additionally, I prioritized user feedback and iteratively improved the app to meet evolving user needs and expectations. This involves understanding and addressing user requirements, then translating them into structured algorithms and data representations. The skills used emphasize integrating user interface presentation and layout with the algorithmic components of the application to ensure a seamless user experience.

The enhancements made to the original artifact have significantly contributed to the mobile app's functionality. For instance, the enhancements to the methods and functions of the SQLHelper class have optimized its functionality, organization, and efficiency, bolstering the robustness of the mobile app's database operations.

The skills demonstrated through this enhancement helped me successfully "**Designing and evaluating computing solutions that solve a given problem using algorithmic principles and computer science practices."** I achieved this course outcome by applying the user-centered approach and following the principles of identifying problems, designing algorithms that are tailored to those problems, and implementing them effectively into code, as well as rigorously testing and refining the solutions. The code implements algorithms for database operations such as creating, reading, updating, and deleting user records. For example, methods like createUser() and updateUser() employ algorithms to manipulate user data efficiently within the SQLite database. These algorithms are designed to optimize database transactions and ensure the smooth operation of the app.

Throughout the process, I ensured adherence to established computer science practices, resulting in the successful design and evaluation of computing solutions within mobile app development. Employing user-centered design principles demonstrates my skills of utilizing established and inventive methodologies, expertise, and technologies within computing to create computer solutions that add value and achieve industry-specific objectives.



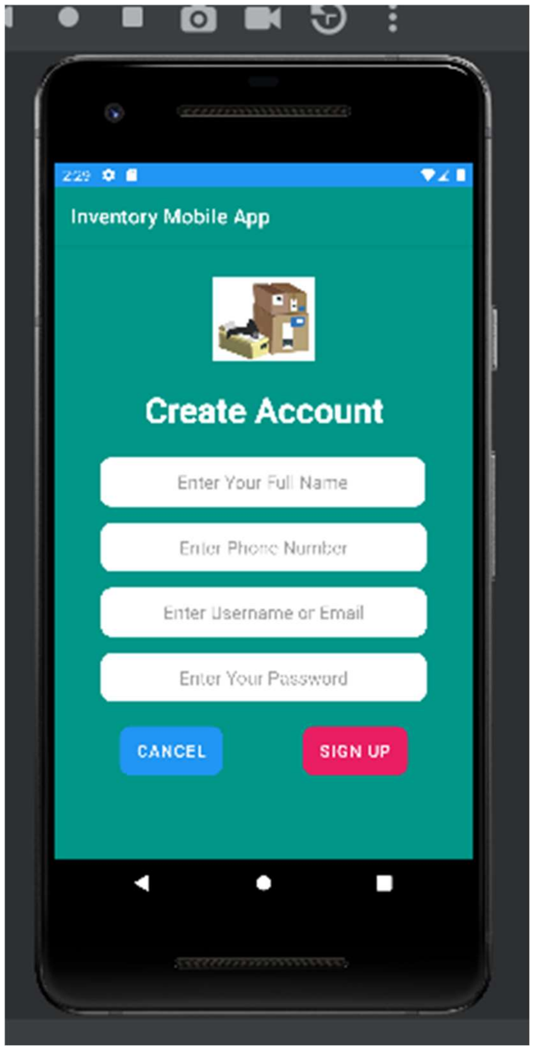
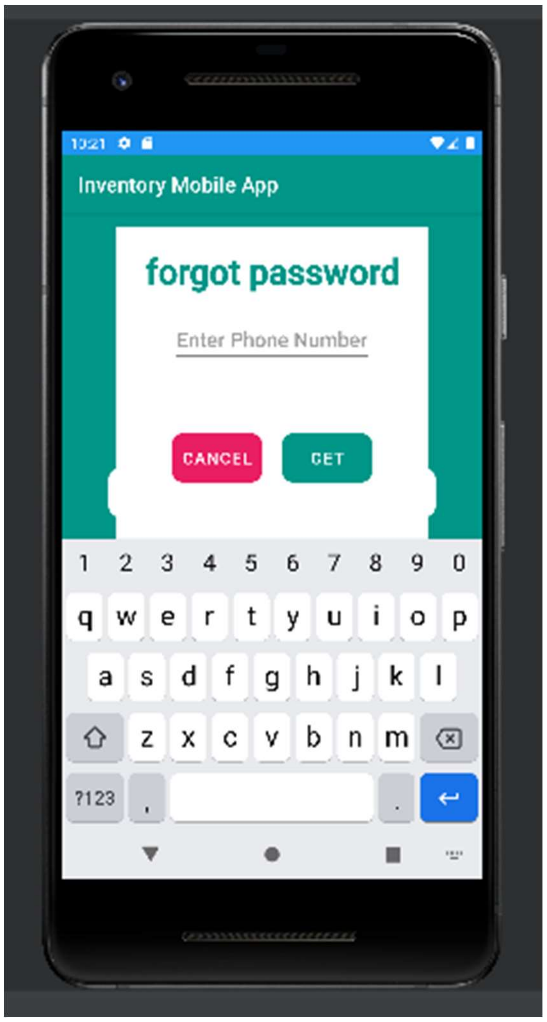


Figure 3: Account creation and login layout of the mobile app

We implemented the login mechanism in the layout to authenticate users when accessing the mobile app using their account-registered username or email, and password. A first-time user can access the account by creating a user account with their unique ***"username or email"*** and password. There is also a ***"forget password"*** mechanism added to the layout that will help users reset their password in case they forget it. These mechanisms are essential in a mobile app's layout as they ensure secure user authentication and access control, safeguarding sensitive user data. Additionally, it enhances user experience by providing convenient account management and recovery options, thereby increasing user engagement and satisfaction with the app.



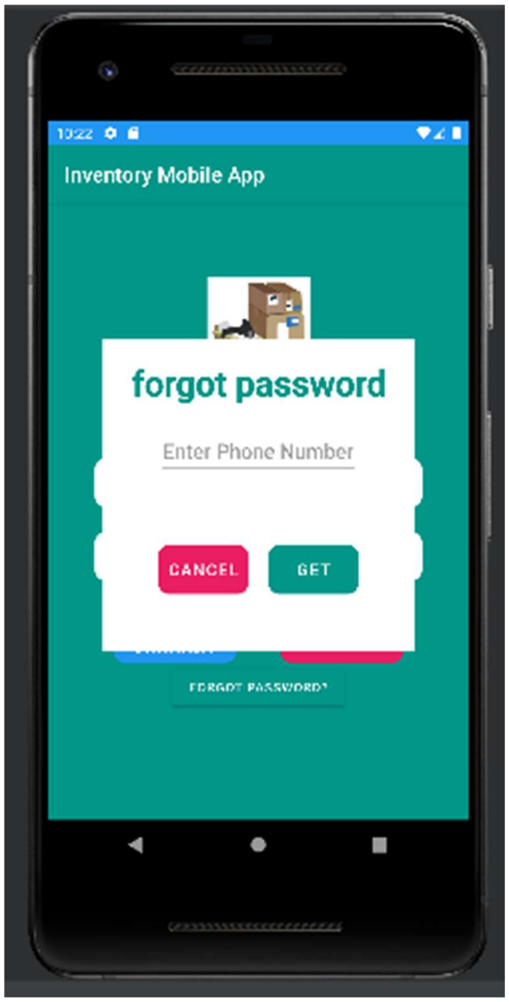
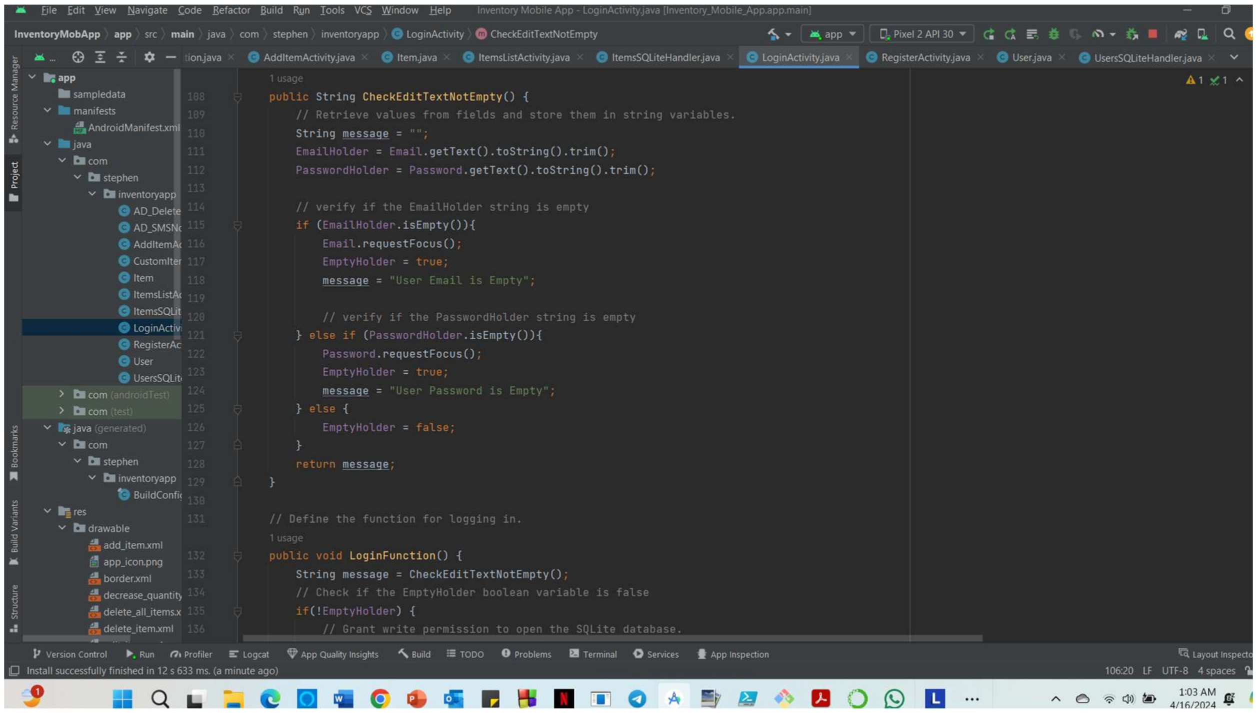


Figure 4: Forgot password layout.

The skills of security mindset demonstrated at this stage of the artifact enhancement shows my achievement of one of the computer sciences program outcomes or objectives which is to **"Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources."** This is achieved through the artifact enhancement by incorporating robust user authentication mechanisms to verify the identity of users before granting access to sensitive functionalities. This includes features like password validation and secure session management to prevent unauthorized access to user accounts such as adding login and account creation mechanisms to the app layouts. In the enhanced code artifact, sensitive user data, such as passwords, is securely stored within the SQLite database. The use of parameterized queries and prepared statements in database interactions helps prevent SQL injection attacks, mitigating potential vulnerabilities in the software architecture.

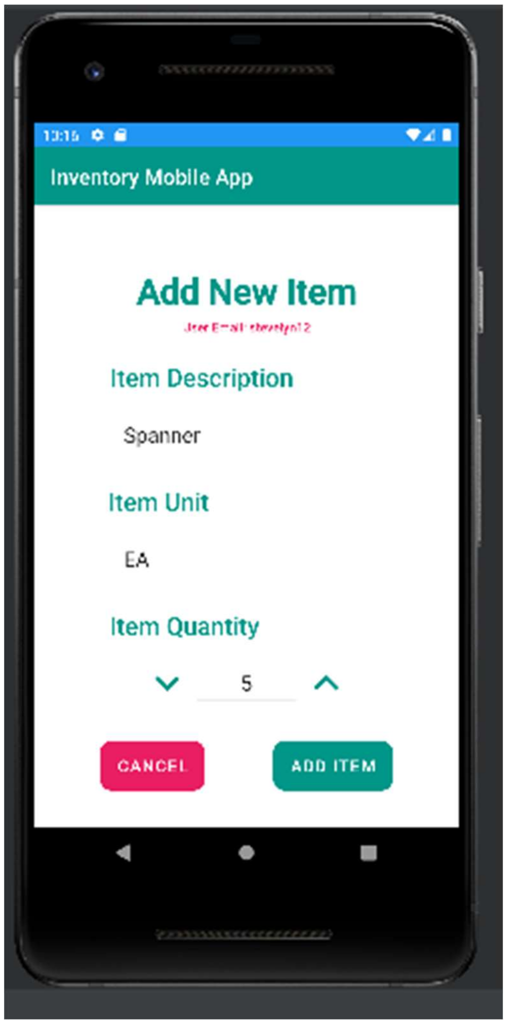


```
public String CheckEditTextNotEmpty() {
    // Retrieve values from fields and store them in string variables.
    String message = "";
    EmailHolder = Email.getText().toString().trim();
    PasswordHolder = Password.getText().toString().trim();

    // verify if the EmailHolder string is empty
    if (EmailHolder.isEmpty()){
        Email.requestFocus();
        EmptyHolder = true;
        message = "User Email is Empty";

        // verify if the PasswordHolder string is empty
    } else if (PasswordHolder.isEmpty()){
        Password.requestFocus();
        EmptyHolder = true;
        message = "User Password is Empty";
    } else {
        EmptyHolder = false;
    }
    return message;
}

// Define the function for logging in.
public void LoginFunction() {
    String message = CheckEditTextNotEmpty();
    // Check if the EmptyHolder boolean variable is false
    if(!EmptyHolder) {
        // Grant write permission to open the SQLite database.
```

Figure 5: Coding formatting and in-line comment of the java file

I employed the best coding standard practices to enhance the code's readability, maintainability, and comprehensibility, making it easier for other developers to read and improving the organization of the application code. By incorporating proper in-line comment conventions throughout the codebase, indentation, and proper code formatting, I ensured that developers could easily jump in and understand each component's purpose, functionality, usage, method, and class. These comments serve as valuable documentation, guiding developers through the intricacies of the code and facilitating collaboration among team members.

The various variables were identified using different naming styles throughout the coding. I achieved this by using CamelCase Cap Words names for class names, such as 'RegisterActivity,' 'LoginActivity,' or 'ItemsListActivity.' Additionally, I utilized CamelCase for method names, such as LoginFunction(), CheckEditTextNotEmpty(), and EmptyEditTextAfterDataInsert(). This naming convention aligns with Java conventions for class names, where each word starts with a capital letter, making it clear and intuitive to identify classes within the codebase and enhancing code readability. These conventions also help distinguish methods from variables and improve code comprehension by providing descriptive names for the actions performed by each method. The camelCase naming pattern is used for methods overruling those in their superclass. The methods that do not override the root class methods are named in camelCase with the first letters.

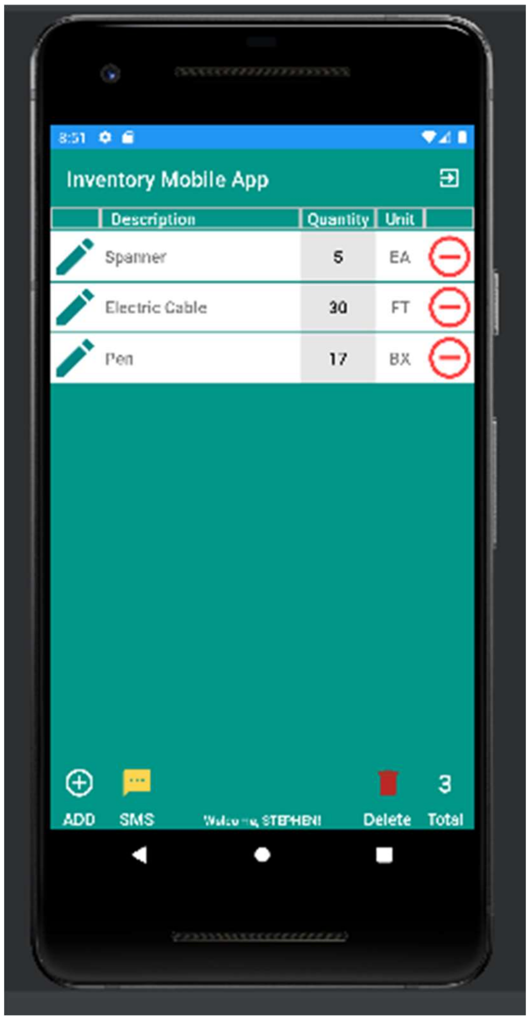


Add New Item

Item Description: Spanner

Item Unit: EA

Item Quantity: 5



Description	Quantity	Unit
Spanner	5	EA
Electric Cable	30	FT
Pen	17	BX

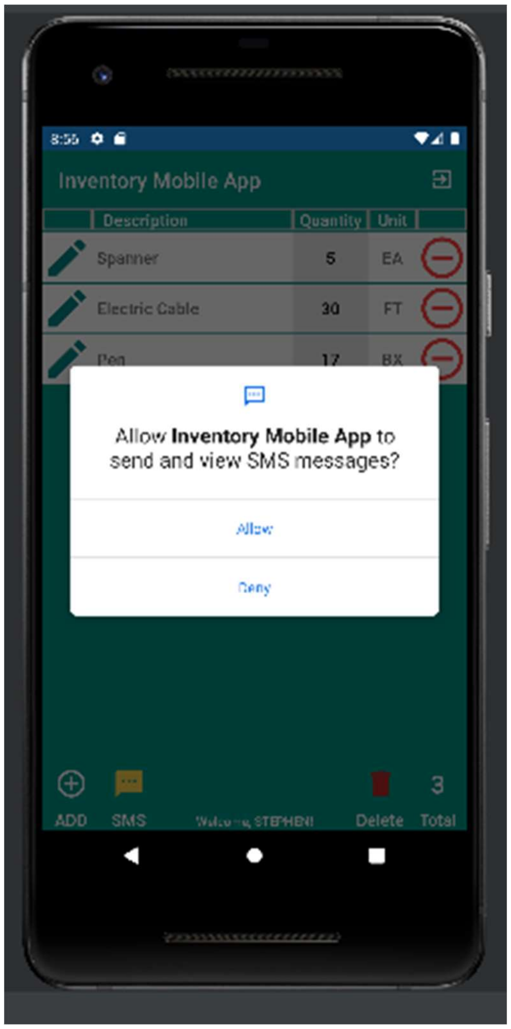
Welcome, STEPHEN!

3 Total

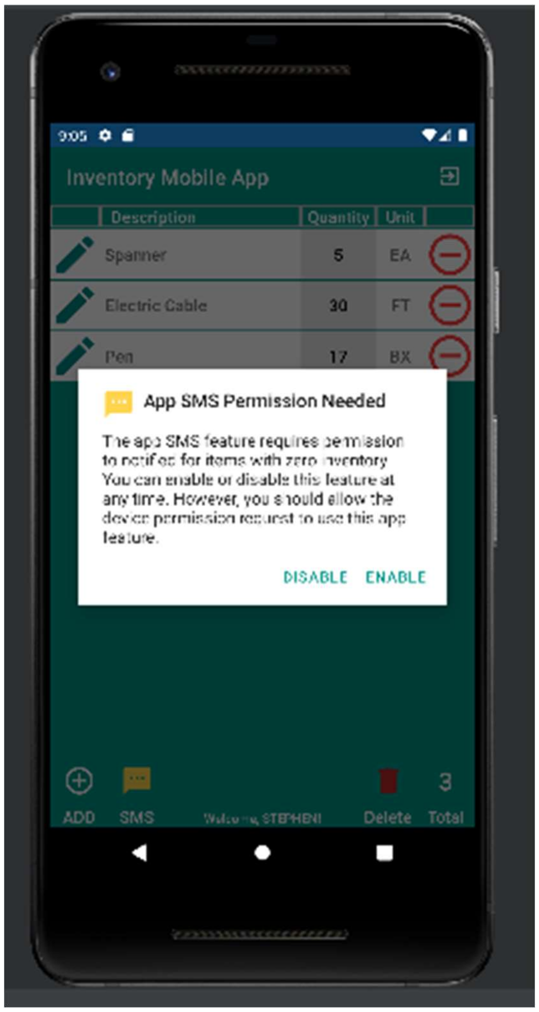
Figure 6: Addition of new items to the inventory record

The layout above allows a user to easily add new items to the inventory record, update the item quantity, and delete existing items. We can navigate to the screen above by creating a username and password account.

One area I also enhanced was the app's user needs. During the design phase, I prioritize features based on user needs and preferences, ensuring that the most critical functionalities are addressed first. I recognize the diverse needs of users and prioritize implementing features that cater to their requirements. To effectively meet user needs, I conduct comprehensive research to understand user behaviors and preferences, which includes analyzing similar apps in the market to gain insights into their design and user experience strategies. I also looked at other competitive mobile apps in the Appstore and internet. I evaluated how they address user needs, including refining the UI/UX design of my app and determining the most effective features. This approach enables me to create a mobile app that not only meets user expectations but also stands out from competitors in terms of usability and functionality.



Allow **Inventory Mobile App** to send and view SMS messages?



App SMS Permission Needed

The app SMS feature requires permission to notified for items with zero inventory You can enable or disable this feature at any time. However, you should allow the device permission request to use this app feature.

Figure 7: Notification of SMS messages.

Also, I enhanced the code by incorporating SMS notification mechanism. Considering the layout above, we will be able to allow the app to send SMS messages to the phone numbers used to register our accounts. The SMS notifications provide a direct and reliable communication channel to reach users, ensuring that important messages or updates are delivered promptly. The addition of SMS notifications to a mobile app can improve communication, increase user engagement, and enhance the overall user experience, making it a valuable feature for app developers to consider implementing.

I demonstrated optimization, time complexity (BigO notation), as well as efficiency of the algorithmic logic in the code. The BigO notation for this implemented algorithm is O(1) for operations like createUser() and deleteItem() because they involve direct database access and manipulation of individual user records, resulting in constant time complexity regardless of the size of the database. The class AD_SMSNotification provides a method to create an AlertDialog with two buttons for enabling or disabling SMS notifications. The method is static and efficiently constructs the dialog with a time complexity of O(1). It utilizes string and drawable resources for localization and consistency across the application. Also, I created an AlertDialog with two buttons for enabling or disabling SMS notifications. Like the previous example, it efficiently utilizes resources and constructs the dialog with a time complexity of O(1).

It is optimized as shown by the careful handling of database transactions in methods like createUser() and updateUser(), which minimize the number of database operations required to perform user-related tasks, leading to improved efficiency and reduced resource consumption. The implemented algorithm is optimized as shown by the efficient database operations and streamlined logic in the code.

Additionally, the deleteAllUsers() method demonstrates optimization by directly deleting all user records from the database in a single operation, avoiding the need for individual record deletions and streamlining the process for improved efficiency and resource utilization.

Throughout designing and enhancing this artifact, I have admitted that understanding the app requires including the user needs is really important. Mobile design is a complicated task, and it will be necessary to complete it effectively and promptly if it involves teamwork. From planning to finishing the app, working together helps us see what we're good at and where we need to improve.

This teamwork lets us tackle the challenges of making an app while following the best ways to do it. The team effort skills demonstrated at this enhancement stage show my achievement of one of the computer sciences program outcomes or objectives: "***Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science."*** I achieved this outcome by fostering open communication and collaboration among team members from diverse backgrounds, including deep online research and feedback from my instructor. This allowed everyone to share their perspectives, ideas, and expertise, contributing to more informed decision-making processes. I actively sought feedback from stakeholders and users throughout the development process to ensure our decisions aligned with organizational goals and user needs.

References:

Yahoo forma parte de la familia de marcas de Yahoo. (n.d.-b). https://images.search.yahoo.com/search/images;_ylt=AwrigIxk_AlmT0EcBN6JzbkF

Android Developers. (n.d.). *Android Mobile App Developer Tools – Android Developers*. https://developer.android.com/