John Bryson

CS-499

4-2 Milestone Three: Enhancement Two: Algorithms and Data Structure

Artifact Description

The artifact I selected for this milestone is my Inventory App, created during the CS360 Mobile Architecture and Programming course. It is a mobile application developed using Java in Android Studio. The goal of the app is to allow users to manage a list of items by adding, editing, and deleting inventory entries. The app uses SQLite to store the data, and it includes user authentication features like login, registration, and password recovery. It was originally created earlier this year as a course project.

Justification for Inclusion

I chose this app for the Algorithms and Data Structures category because it clearly shows my ability to apply important computing principles. It uses sorting, searching, and structured data management through custom Java classes and methods. The logic in the app allows users to create items, view them in a list, edit existing records, and delete them from the database.

The artifact also includes structures that organize the data and connect it to the SQLite database. These structures include arrays, lists, and helper methods that demonstrate my ability to work with algorithms in a real-world mobile application. I improved the original project by cleaning up the Java code, making better use of method structure and comments, and improving how items are stored and displayed.

Course Outcomes

Yes, I believe I met the course outcomes I set out to achieve in Module One. My goal was to show how I could build efficient algorithms in a practical setting, and the Inventory App allowed me to do that. I do not have any updates to my outcome plan right now, but I am confident this artifact represents the outcome of designing and evaluating computing solutions that solve real problems using algorithmic principles.

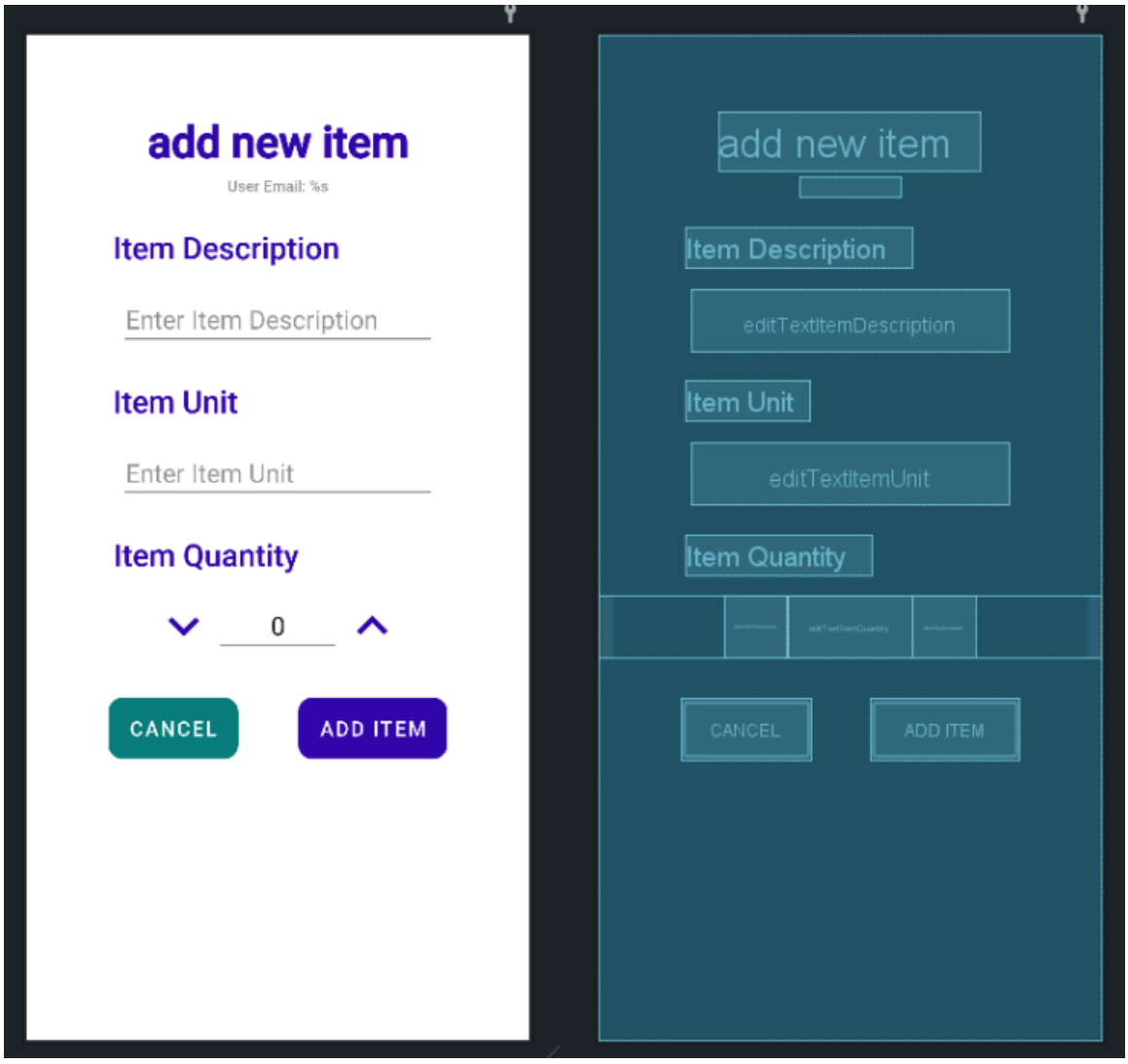
Reflection on the Enhancement Process

Enhancing and modifying the Inventory App taught me a lot. I improved the organization of the code, fixed small bugs in how data was displayed, and added more user-friendly design to the way items were sorted and listed. One challenge I had was managing the SQLite database connection and making sure the app did not crash when switching between activities. I learned how important it is to test small parts of the code often.

I also made sure to use good coding practices, like using proper names for variables and adding comments that explain the purpose of each method. I used inline comments, consistent formatting, and followed Java naming conventions. This made the app much easier to understand and update. These improvements helped me better understand how important clear structure and logic are in software development.

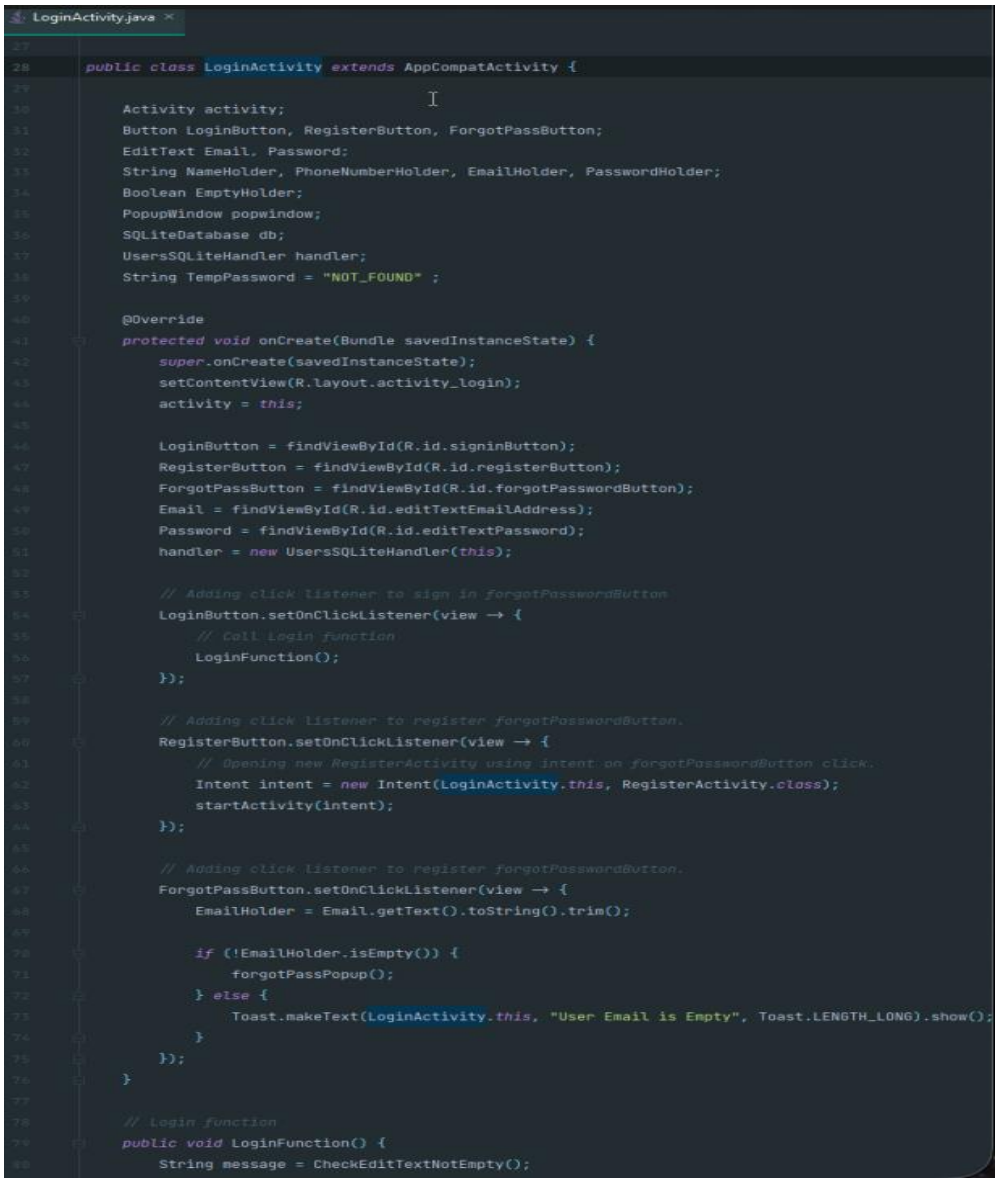
Some Images showcasing enhancements and display of skill:

Add Item Activity Layout: Shows the design of the Add Item screen. I used clear labels and input fields that connect to the logic in the Java code and SQLite database.



Login and Register Activities: Displays the user authentication features, including registration and password recovery. These features use input validation and security checks.

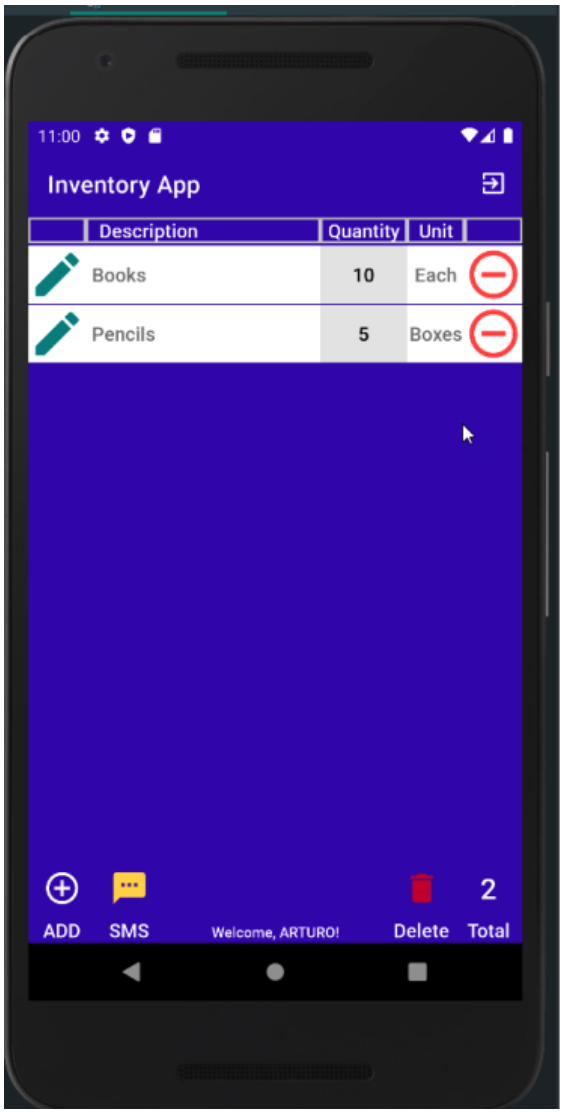
This screenshot shows a portion of the `LoginActivity.java` file from the Inventory App. It highlights the structure and logic of the login screen, including the setup of UI components like buttons and text fields, the initialization of the SQLite handler, and the use of click listeners to handle login, registration, and password recovery actions. The code uses clean formatting, proper naming conventions, and inline comments to explain each section, which demonstrates attention to detail and best practices in Java programming and Android development.

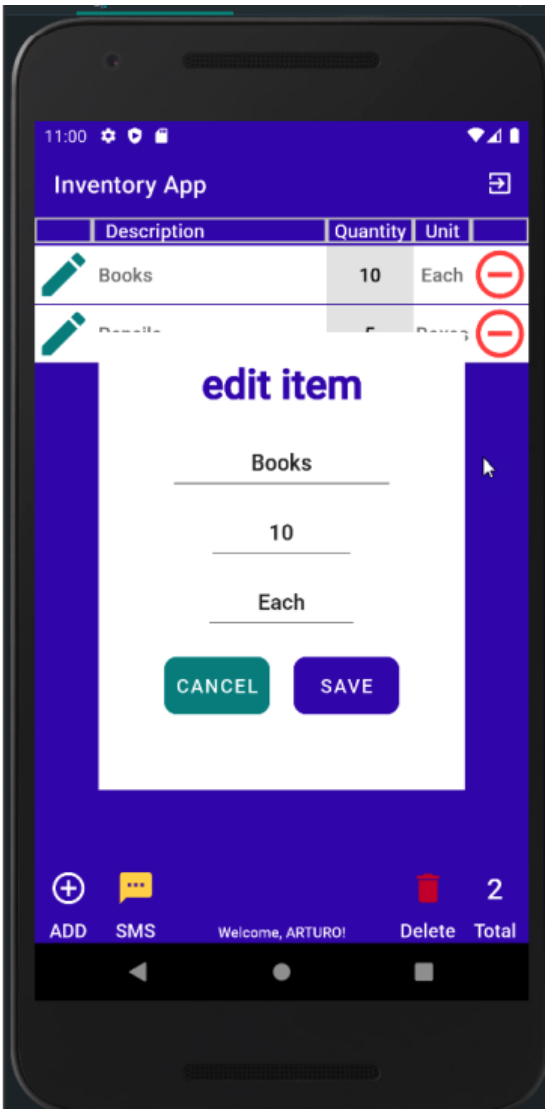


```java
public class LoginActivity extends AppCompatActivity {

    Activity activity;
    Button LoginButton, RegisterButton, ForgotPassButton;
    EditText Email, Password;
    String NameHolder, PhoneNumberHolder, EmailHolder, PasswordHolder;
    Boolean EmptyHolder;
    PopupWindow popwindow;
    SQLiteDatabase db;
    UsersSQLiteHandler handler;
    String TempPassword = "NOT_FOUND" ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        activity = this;

        LoginButton = findViewById(R.id.signinButton);
        RegisterButton = findViewById(R.id.registerButton);
        ForgotPassButton = findViewById(R.id.forgotPasswordButton);
        Email = findViewById(R.id.editTextEmailAddress);
        Password = findViewById(R.id.editTextPassword);
        handler = new UsersSQLiteHandler(this);

        // Adding click listener to sign in forgotPasswordButton
        LoginButton.setOnClickListener(view -> {
            // Call Login function
            LoginFunction();
        });

        // Adding click listener to register forgotPasswordButton.
        RegisterButton.setOnClickListener(view -> {
            // Opening new RegisterActivity using intent on forgotPasswordButton click.
            Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
            startActivity(intent);
        });

        // Adding click listener to register forgotPasswordButton.
        ForgotPassButton.setOnClickListener(view -> {
            EmailHolder = Email.getText().toString().trim();

            if (!EmailHolder.isEmpty()) {
                forgotPassPopup();
            } else {
                Toast.makeText(LoginActivity.this, "User Email is Empty", Toast.LENGTH_LONG).show();
            }
        });
    }

    // Login function
    public void LoginFunction() {
        String message = CheckEditTextNotEmpty();
```

Item List and Edit Item Screen: Demonstrates how users can view and manage their items.

The logic here involves sorting and updating the data structure behind the scenes.





SMS Dialog and App Info: Displays extra features like alerts and app info, which use decision structures and event-based actions.