**C868 – Software Capstone Project Summary**

**Task 2 – Section C**



**Capstone Project Name**: SmartStock – Innovative Inventory System.

**Student Name:**  Lafyette Russell

# Table of Contents

Table of Contents

[Design Document](#_Toc2084998) 2

* + - * Class Diagram 2
      * Design Diagram 4

Test Plan 5

* + - Introduction 5
      * Purposes 5
      * Overview 6
    - Unit Test 6

User Guide 7

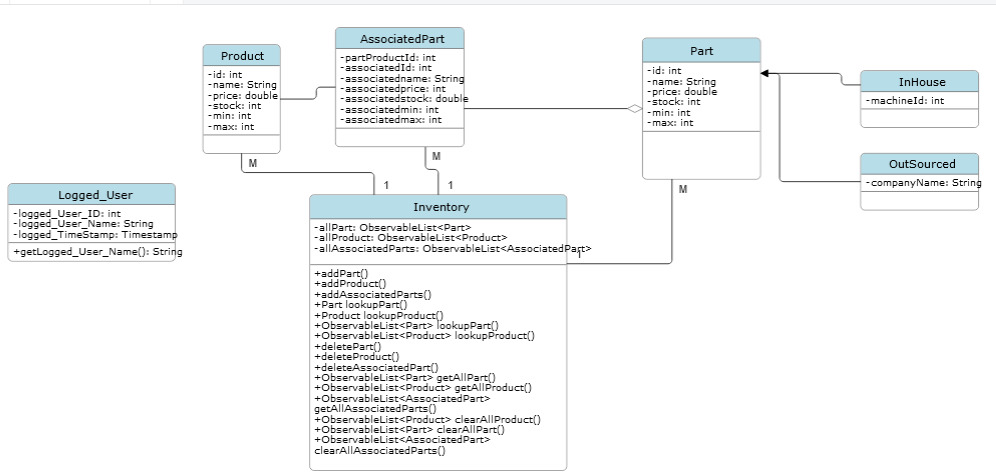
* + - Set Up and Maintenance 8
    - Running the Application 12

Source Code and Executable 12

**Design Document**

**Class Design**

This UML Design fig.1 shows the relationship between classes within the application and how the classes are designed. The in-house and outsourced part inherits from the Part class, the Associated Part class is somewhat related to the part class but does not directly inherits from it, while Product class is connected with the Associated Parts class. Logged\_User class is separate as it doesn’t directly interact with the other classed in the diagram. Inventory also has a 1 to many relationship with Product, Part, and Associated Part classes.



**Design Diagram**

This is the login screen(figure.1) the user is greeted with upon first launching the application here you will also be able to see the create user text clickable link. Upon selecting the create user link the user will be take to the create user view(figure.2) where they can make a new user account. On the other hand if the user logs in, the user will be taken to the Home screen(figure.3) that has two tables, Parts table and Product table, from here you can search from parts or products available, or add, modify, and delete from parts or products. The parts add and modify reuses the same view(figure.4) and product add and modify works the same too(figure.5).

Figure 1: Login Screen

Figure 2: Create User

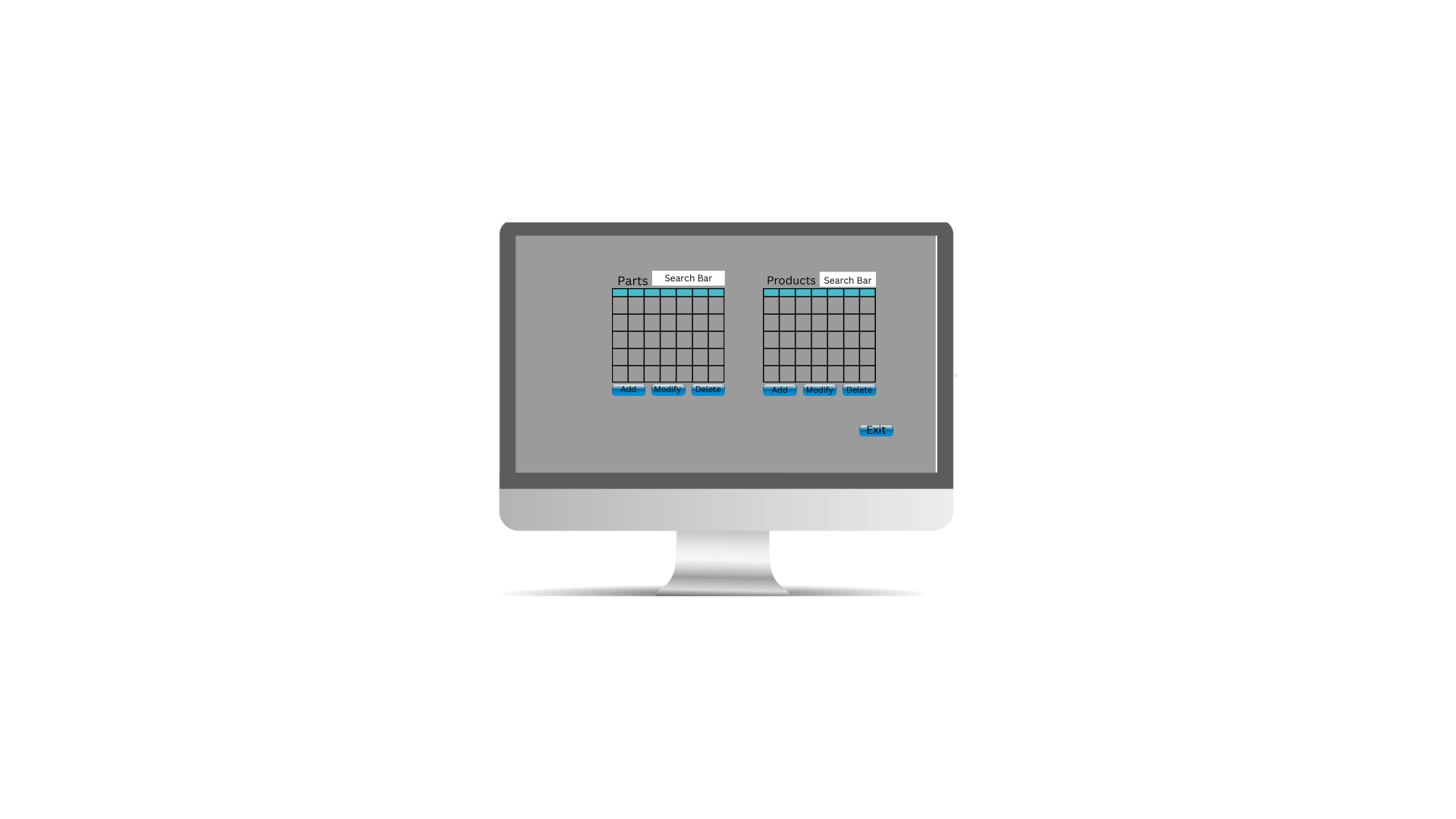
Figure 3: Home

Figure 4: Part Add/Modify

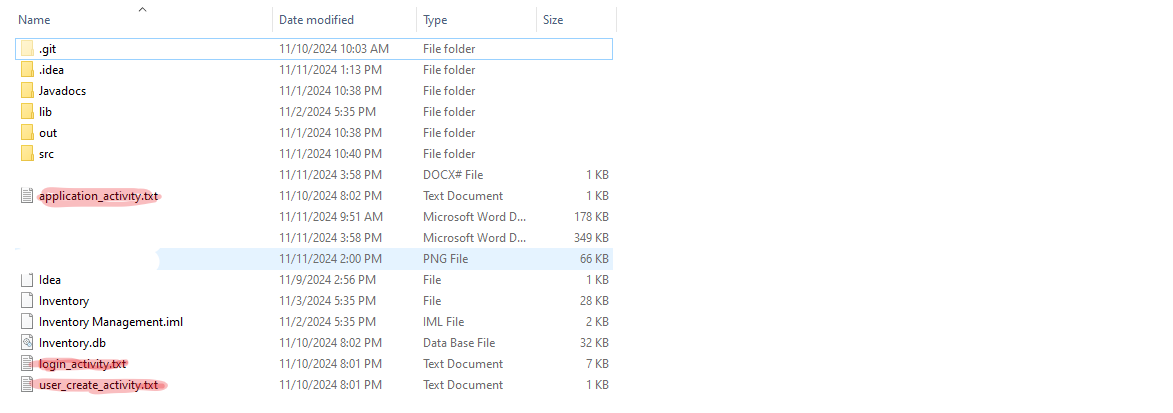
Figure 5: Product Add/Modify

**Test Plan**

**Introduction**

Purpose

The test is designed around the logging functionality of the application, for example a login\_activity.txt when logging in, a user\_create\_activity.txt when creating a new user, and a application\_activity.txt when a user adds, deletes or modify a Part or Product. The file should be in the directory(Marked red):

Figure 6: Directory of application with log files

Overview

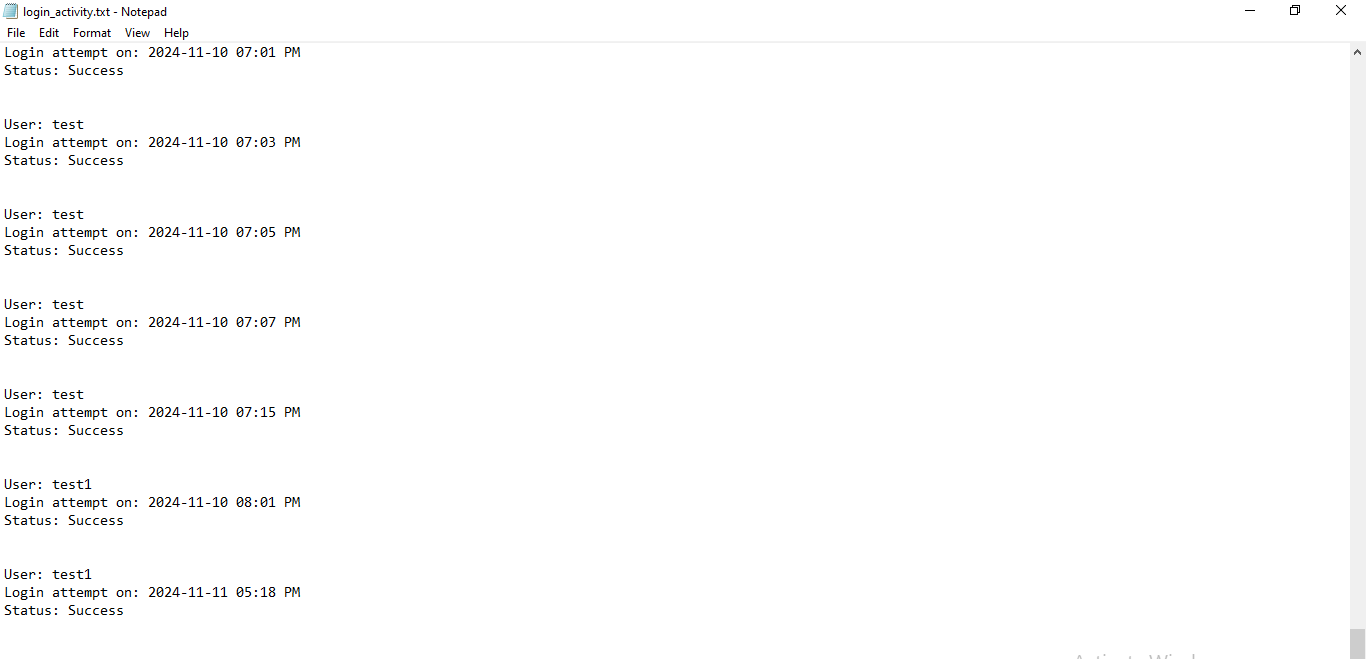
The user must commit different activities within the application that generate log reports that are then saved to the application directory and the logs must be able to continually have new log data appended to them.

**Unit Test**

The test involves a series of steps:

1. User must first create new user account to generate user\_create\_activity.txt log.
2. Once user account is created the user is able to log in, which upon success will generate login\_activity.txt log.
3. Next at the home screen user can add, either a product or part to generate a application\_activity.tx log.
4. After completing third step log file will be located in directory of application(Figure.6) next user will test the logging function ability to append new log data.
5. User will complete any step from 1-3 to append new log data to respective log activity and will verify that new data has been appended.(Figure.7)

**User Guide**

Figure 7: Log with new appended log data

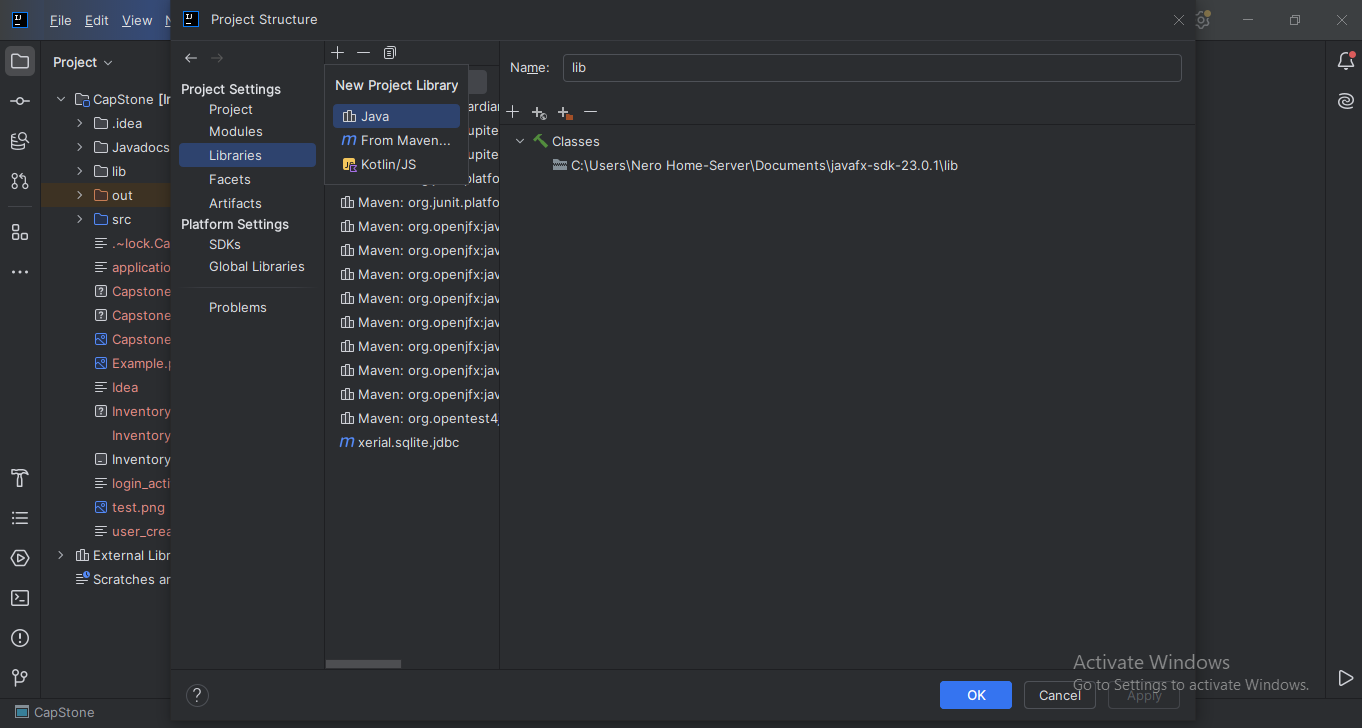
**Set up and Maintenance**

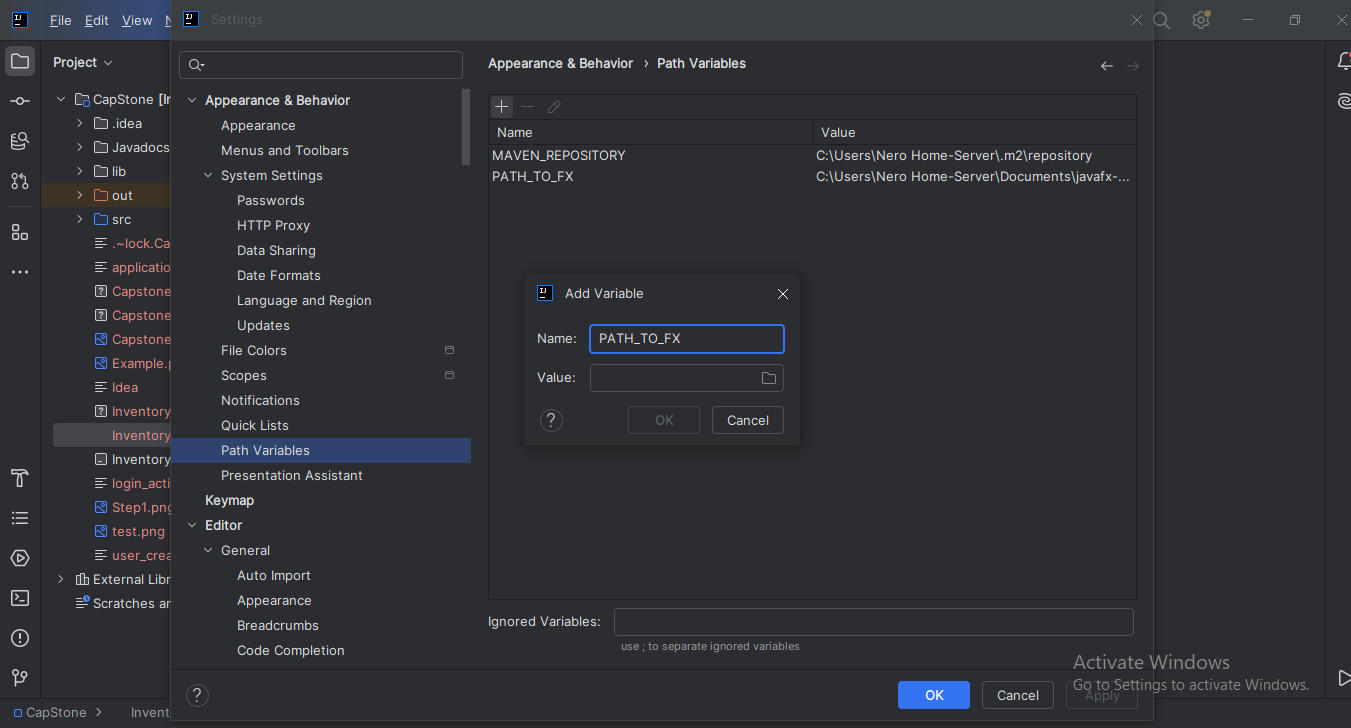
In order to properly do any maintenance on the application, there are few steps that are required to get the code development environment set up properly.

Prerequisites

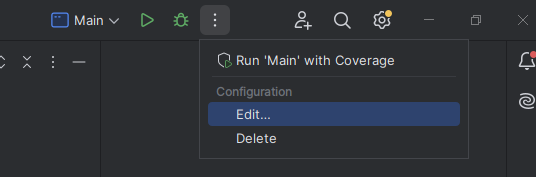
* [IntelliJ IDEA](https://www.jetbrains.com/idea/download/?section=windows)
* [JavaFx SDK](https://jdk.java.net/23/)
* [SQLite JDBC Driver](https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc)
* [Java SE 23](https://www.oracle.com/java/technologies/downloads/)

Once you have downloaded all of the above you now need to Install IntelliJ, once that is installed now you need to install javafx Sdk.

1. Go to file→project structure→libraries→add project library→Java and select lib folder in javafx directory.
2. Next you will need to got Settings→Appearance & Behavior→Path Variables. Now you need to create a path to the Javafx folder with the path name being PATH\_TO\_FX

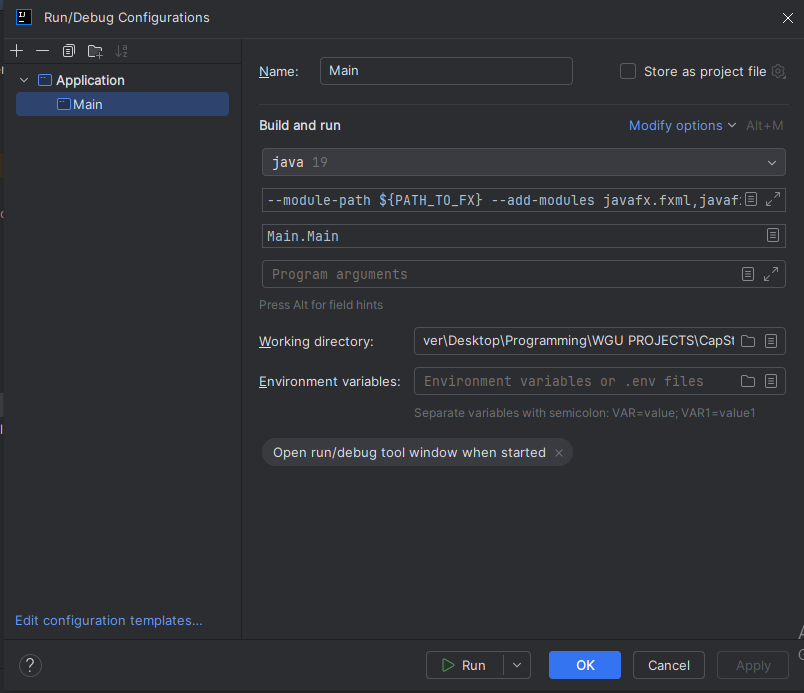


1. After doing that select edit configuration of run/debug



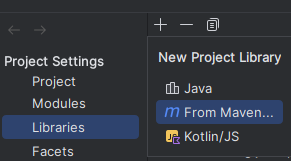
Then within that screen paste “--module-path ${PATH\_TO\_FX} --add-modules javafx.fxml,javafx.controls,javafx.graphics”

within the VM options field

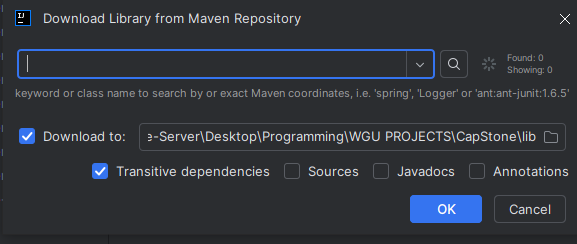


To install the SQLite jdbc drivers repeat step 1 of javaFx install but select the sqlite jdbc jar file instead.

Alternatively for Maven Projects you can use From Maven Option to add libraries:



Afterward you will be able to input the link to download and install the libraries



For Java Fx: com.jetbrains.intellij.javaFX:java-fx:version\_here

For SQLite JDBC: org.xerial:sqlite-jdbc:version\_here

Adding dependencies in the pom.xml file is required for both.

**Running The Application**

Ctrl-click below to open links for video guides.

[Creating User and Signing in](https://drive.google.com/file/d/1oeSiyRT7s6BtrpQFks-H0Lb_U1chl3ix/view?usp=sharing)

[Adding, Deleting, and Modifying Parts](https://drive.google.com/file/d/1GshaemQC-FZr0vOlyb4F3GwY2kYDm-nn/view?usp=sharing)

[Adding, Deleting, and Modifying Products](https://drive.google.com/file/d/1oeSiyRT7s6BtrpQFks-H0Lb_U1chl3ix/view?usp=sharing)

Link to Source -→https://github.com/LinkFan100/CapStone-Final.git