**Group 1 –- Assignment 1 –- BTP400NBB**

Workload Breakdown:

* **Student Module**: *Wilson*
* **Admin Module**: *Mohammad*
* **Librarian Module**: *Matthew*
* **JavaFX / GUI**: *Shrey*
* **Database**: *Matthew*
* **Documentation & Deliverables**: *Matthew & Wilson*

Total Meetings: ***3***

Mode of Communication: **Discord & GitHub**

Planning**:**

*Initial UML*

**Diagram

Description automatically generated**

***----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------*** *Final UML*

**Diagram

Description automatically generated**

Design:

*Overview:*

The system is a CSV .txt file database. There is a table folder that contains classes that represent the data within their respective file i.e., Student, Book, etc. The Database class contains relative Path’s for these files, and will auto generate them, if necessary, upon instantiation. TableController handles general functionalities for *all* tables. TableController extends the Database class. The 3 main modules of our database; Student, Admin, and Librarian, are all given separate Controller classes. These classes handle the specific functionalities and runtime IO of the application’s requirements.

*Testing:*

A module called “Module” can be found within the application. This module contains a main method and can be modified for testing. There is another module called “TestDB” which simply hardcodes preset information for the database files. The testing module will first instantiate a TableController, which creates a database, and generates all necessary files (if not already created). It then runs a series of hardcoded commands to test the functionality and robustness of the code. *Most* Controller method calls will be logged to the Log.txt file, it is here that you can see when and what happened in the database. The testing module does not require tinkering or user input to be tested. It can, however, be adjusted according to what you would like to test if needed.

*\*If you would like to test the code using your own user input, please run our completed build.*

*Noteworthy Design Decisions:*

* Config.txt was going to be used to read settings into the database. The database would autofill the config.txt file to a hardcoded default upon creation. The design still allows it to be implemented, however there is no need right now.
* The Item class & Items.txt allows for different “types” of Items to be stored in the database. I.e., Magazines, Newspapers, etc. Although partially implemented, it is not tested out of the box.