COMP 3320
University Of Windsor

Phase 2

Elaboration 1

Ahmad Alghizi ID: 105068246
Wasek Rahman ID: 110020383
Liban Guled ID: 104586116

July 16, 2020
All members contributed equally

0.1 Introduction

The phase plan for the elaboration phase for this project was centered around manipulating data, be it creating, reading, updating or deleting (CRUD) a CSV file and storing it in a database. Having this software backend is the core functionality of Windsor Open Data, the main priority was making sure that our software works before moving on to the consumer navigation side of our project. With the backend now working, the development team can look to start building a friendly UI so that everyone from the common public and data driven professionals can navigate and use Windsor Open Data. Our project team worked with great cooperation with all tasks, and work was distributed evenly. Our team made great use of Discord's screen sharing capabilities during the coding process so that we could all write software together. We used WhatsApp to communicate and distribute the workload. We had brief team meetings every night to discuss our progress and we always made sure to hit our work targets. Our teamwork was seamless and effective and all team members contributed equally.

0.2 Source Code Design

We have our code sourced on Github which we also used for our version control. Whenever a team member wanted to make a code contribution, they pushed their software to the main source code on Github. We always maintained the latest version of our code without major bugs on our Github repository. Our code documentation is also on Github.

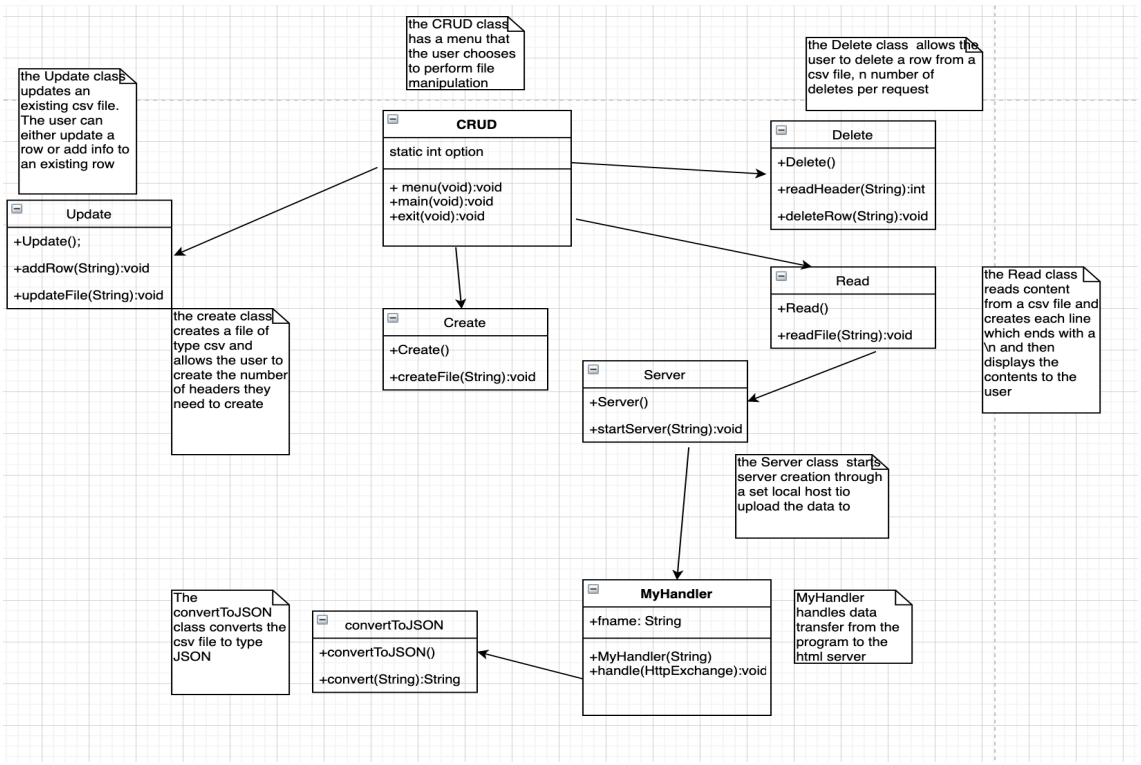
Source Code Link: http://github.com/WasekRahman/COMP3220

Documentation Link: http://github.com/WasekRahman/COMP3220/blob/master/Guideline/guideline.html

The main priority of our first development cycle was to have a functional backend. What our code does is essentially allows users to create or manipulate files that are inputted by the user. Our code has a text-based UI menu that allows users to access CRUD operations and manipulate data as needed. We then converted the CSV files, which contained all the data, to JSON files and set up a server with a local host to then display the data online. The data display method we used is a proof of concept, essentially a minor presentation of what is to come in the future as then different data files would be stored in different locations in the web server depending on information.

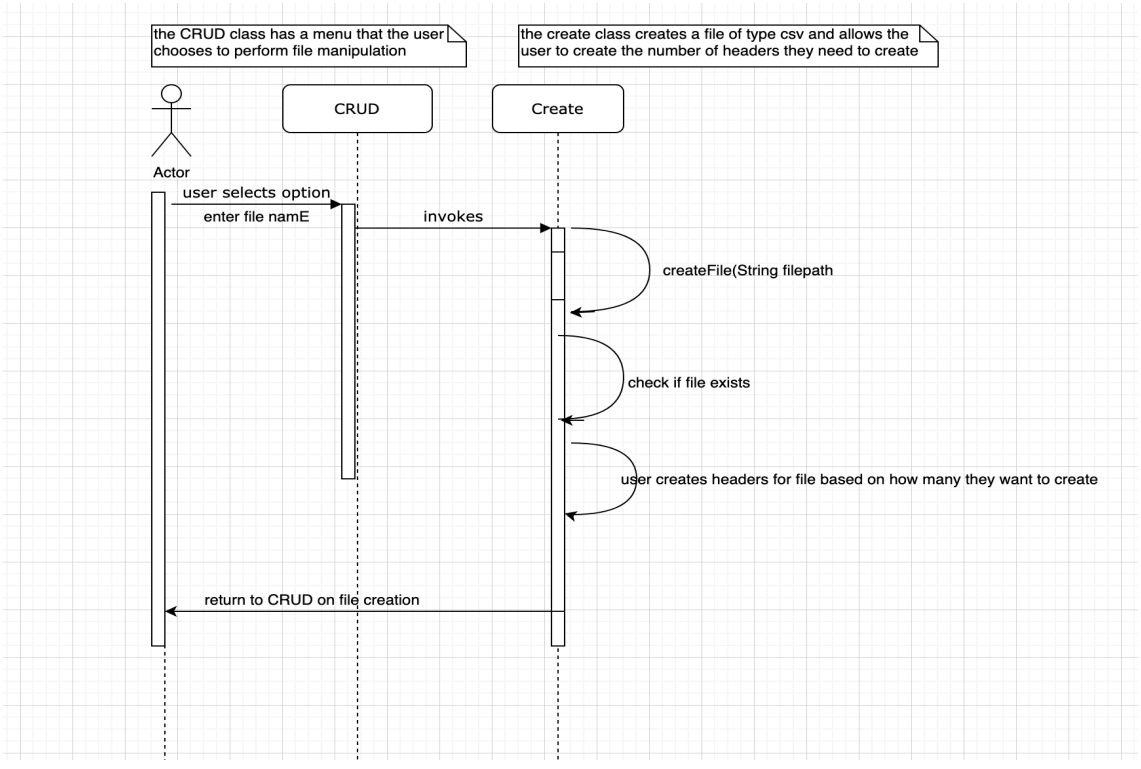
0.3 Class Diagram

These are the classes that make up our backend software: This diagram and our code is made with high cohesion and loose coupling as each class has its own functions that run their tasks without relying on the other classes Where CRUD serves as the main class that calls all other classes with the exception of the Read class. When the Read class is invoked and the user decides to upload the data to the html web server, Read will call the Server class which then invokes the MyHandler class which calls convertToJSON to upload the data to the server. Regardless the class calls that follow after Read are only dependant upon the Read class and if we look exclusively at those connecting classes we can imply that they are loosely coupled within themselves.



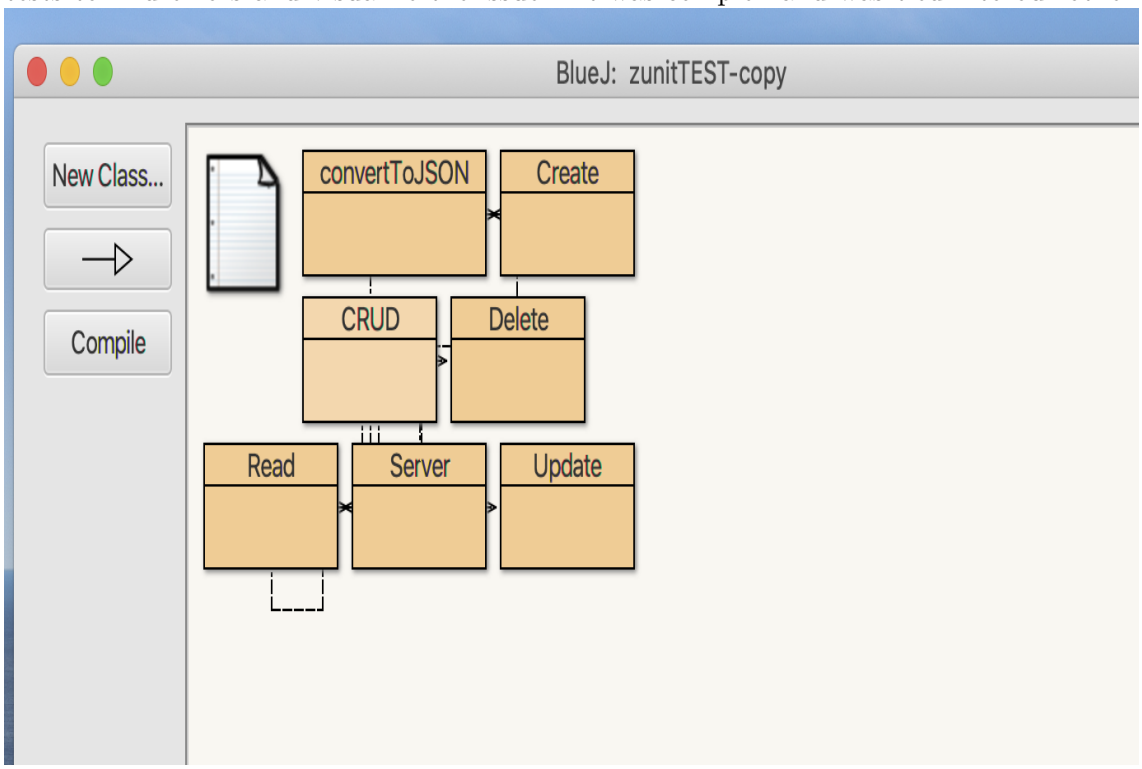
0.4 Sequence Diagram

For the sequence diagram designed below, an interaction is shown between the user and the program in which the user begins the program by invoking the CRUD class which then calls the Create class. The Create class begins by invoking the method createFile() which proceeds to check if the file name entered by the user already exists and if it doesn't the method asks the user for the number of headers the user would like to create for the file and then the file is created and upon succession returns to CRUD which proceeds to re show the menu system call for the user to either terminate or carry out another file manipulation task.



0.5 Testing Strategy

We used the built in software BlueJ to test our classes and functions as we built it. For each class we unit tested by creating test conditions and then printed out the expected outcome to confirm each individual component worked as we built the software from the ground up. For example, in the update class, our test condition consisted of adding simple rows and then immediately printing them in the next line to prove the code is functional. After each unit test we deleted the testing conditions and print statements after we confirmed it worked. By the time we completed the first version of our software, all components worked seamlessly and was fully functional without any errors due to our unit testing. The software BlueJ helped us in our unit tests to find errors and visualize the issue if it was complex and was tied into our other classes.



0.6 Bug Reporting

Whenever major issues arose in our code and help was needed to fix the problem, we reported the bug on Github and then sent a message to the team via WhatsApp or mentioned the issue during the nightly team meeting.

Bug Reporting Link: http://github.com/WasekRahman/COMP3220/projects/1