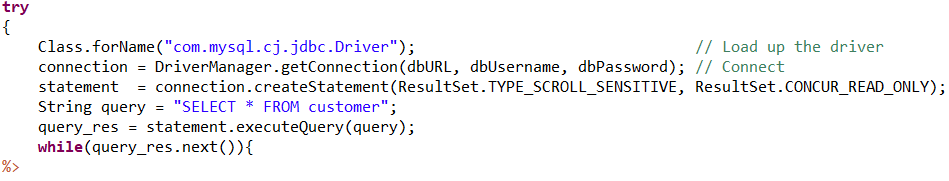
**ADMINPAGE:**

Michael also did a bit of work on the Admin Page which displays all the user records that are stored in the database. This is done through the use of a JSP which has java code embedded into to it to make a connection to the database and query the customer table for all the records that are in it.



The results of the query are then added into a table with a table row each row entry in the database.

**GITHUB:**

For the duration of the project Michael was in charge of managing the GitHub repository where the team would commit their code changes to . This involved initially creating the repo and inviting the all the other group members to be collaborators on the project. Once the collaboration invitation has been accepted then the other group members have full access to add and remove files or make changes to existing files

The repo can be found here:

<https://github.com/oharam29/EE417_WebAppDev_GroupF_Project>

A screenshot of the top level of the project can be seen below, containing:

* Documentation folder - for the report and minutes of meetings
* GFB – For the actual web application source code
* WebTest – Folder containing the selenium test scripts
* Web.sql – SQL script for creation of database



It also involves making sure everyone was able to link their version of the eclipse IDE or which ever IDE they were doing the development of the web application in and ensuring they could commit changes correctly and pull-down changes correctly. Inside of eclipse in order to commit or pull any changes, a group member must right click the project and open the team menu to have access to the git actions such as commit, push, pull and more.

We had a few mishaps along the way of code going missing or being overwritten which required a roll back to a previous version of the code base and manually adding changes back in one at a time to ensure we did not lose any progress.

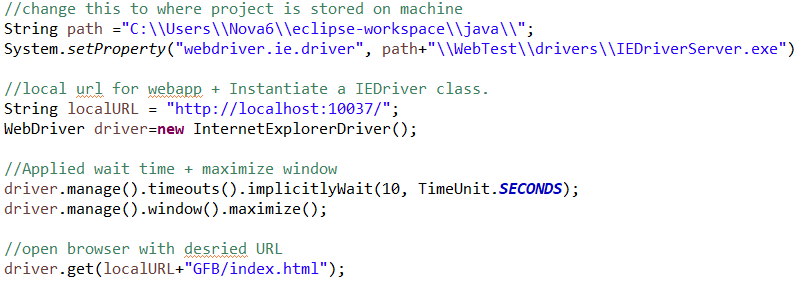
This also involved merging in the front-end branch in with the main branch when we decided against doing branches. Different code branch did not make sense for this type of project as all the sections of the website worked in tandem with one another, so development was done in tandem.

When this merge took place there was a number of conflicts which were cause by two members working on the same file at the same time, to resolve this the files were combined, and minor adjustments were made to incorporate both sets of changes.

**TESTING:**

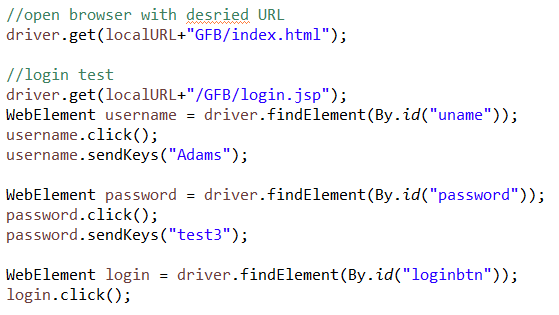
Due to the lack of participation from two group members Michael was also in charge of testing for the web application.

One method of testing that was incorporated into the projects was to use Selenium to run automated UI tests. These tests were run on the Login and Register sections of the web site. These tests are run using a web driver deriver server which is controlled by Java code which can be found in the project repo.



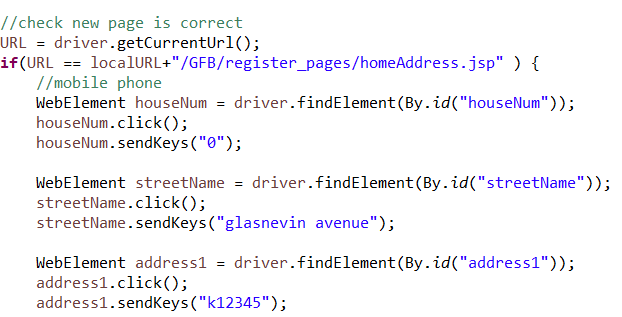
The above screen grab is a the set up for the test which defines a string path which is the path to where the project is stored on your local machine, the next line defines the type of driver the test uses and the path to the web driver file, which is made up of the variable path plus a path to where the web driver is in the project folder. It will then set the URL to localhost and the port the tomcat server is pointing at, in this case 10037, and instantiate a new web driver and maximize the window size.

It will then navigate to the home page web application. Now the tests are ready to be performed. The first test that was written was a basic login test which navigates to the login page and enters credentials for an account stored in the database and logs in to the website.



The way this test is run is in it create web elements which are items on the page so for this test they are the username field, the password field and the login button, it gets this element by searching for them using the ID assigned to them in the html. For the texts it will call “*sendkeys”*  to simulate a user entering their details, this will be the string pass to the call to that field. And then finally it will get the login button on the page by ID and simulate a user clicking on that button.

A similar test was done for the registering to the website, it will navigate through the pages associated with registering for the website and enter in values that a user could potentially enter.



As seen above it goes through similar steps to the previous test but with one difference, it will check the URL is correct upon clicking the continue button and moving to a new page.

The second type of testing that was done was manual QA testing. In order to carry this out the web application was opened up locally. Then the website was then tested by performing the actions the user would carry out such as navigating the links of the website to ensure the links work as intended.

It was also the actions of testing the servlets to make sure the do the intended actions such as simulate a deposit/withdrawal.

It also involved ensuring the database calls on the admin page to display a list of all the customers in the database at the time of loading that page.