**Technical Report**

**- An explanation of the mechanism you would use that enables inter-process communication in your application.**

Inter-process-communication is a way for switching information amongst application process threads across many applications on different computers across a network on a single computer and the applications which take advantage of IPCs are referred to as a client server application. For this project we can use many different mechanisms to enable inter-process communication within our application but we decided on using .NET Remoting. Using .NET removing within our application allows us to build it to be widely distributed easily whether the application components are all on the one computer or are spread out across other computers possibly across the world.

**- An explanation of what is an Interface in object-oriented programming and how would you use it to apply multiple inheritances.**

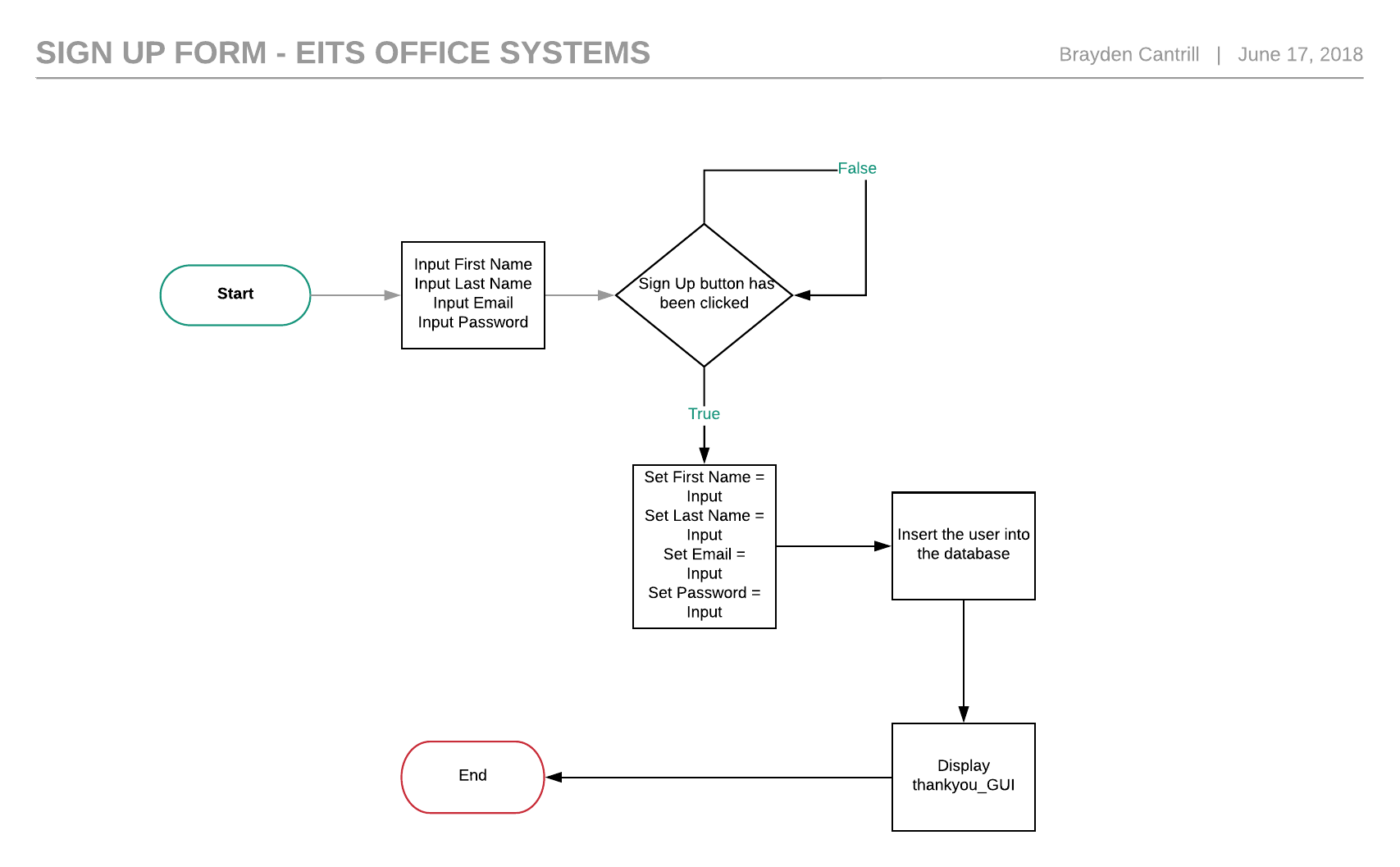
An interface in object oriented programming is nothing like a user interface but instead is something that is very similar to a class but have no actual functionality and no actual code. The benefit to using an interface is that you can have different classes choose to implement the same interface while letting other parts of the app use objects with interfaces. Also using an interface will improve the efficiency of your code throughout loose coupling and abstraction.

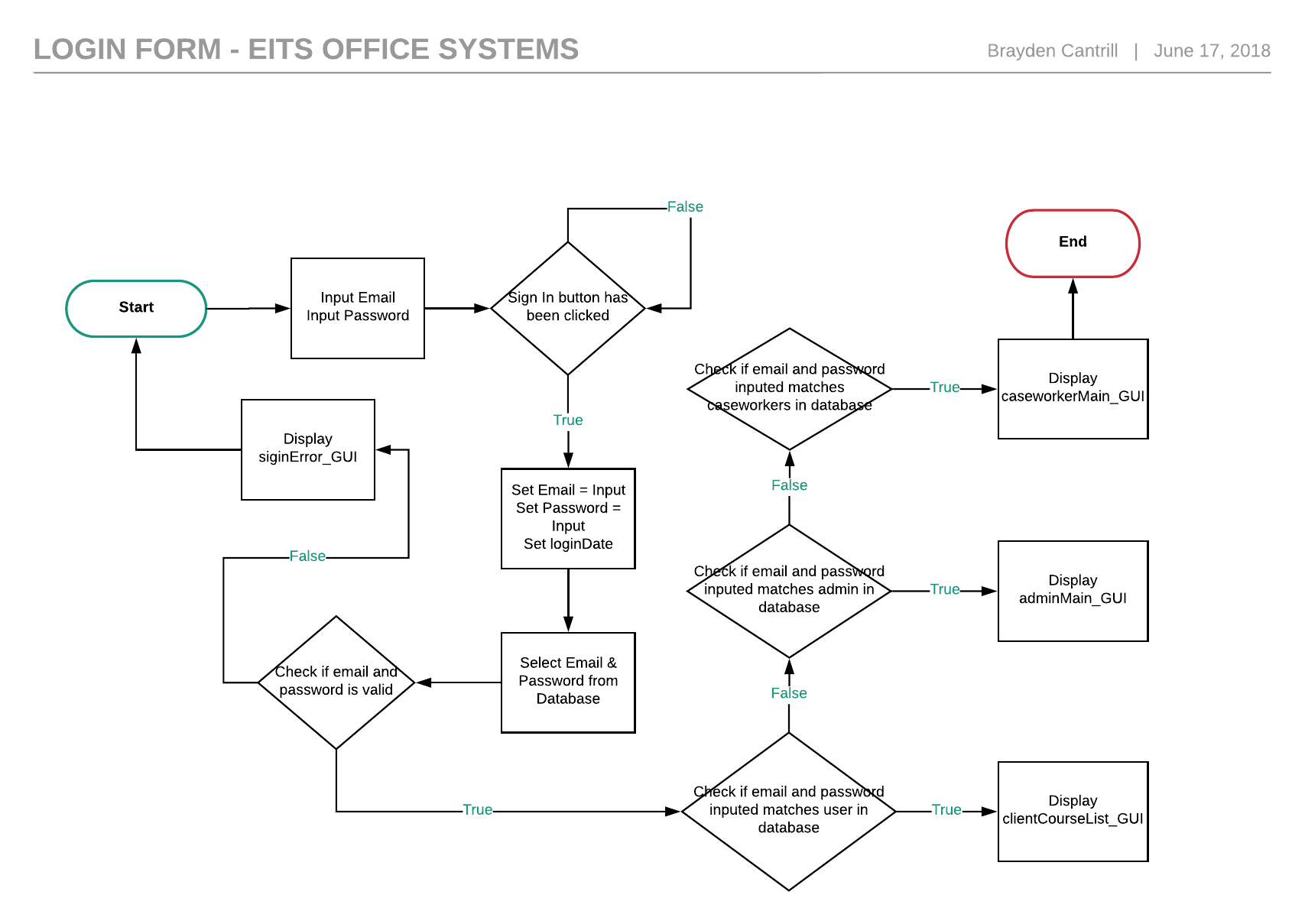
Multiple inheritance is when a class or object inherits features and properties from more than one parent class or object. Inheritances can and are implemented in a variety of ways depending on what language you are coding with. For Java including an inheritance requires you to type extends and then the name of the class you want to inherit from.

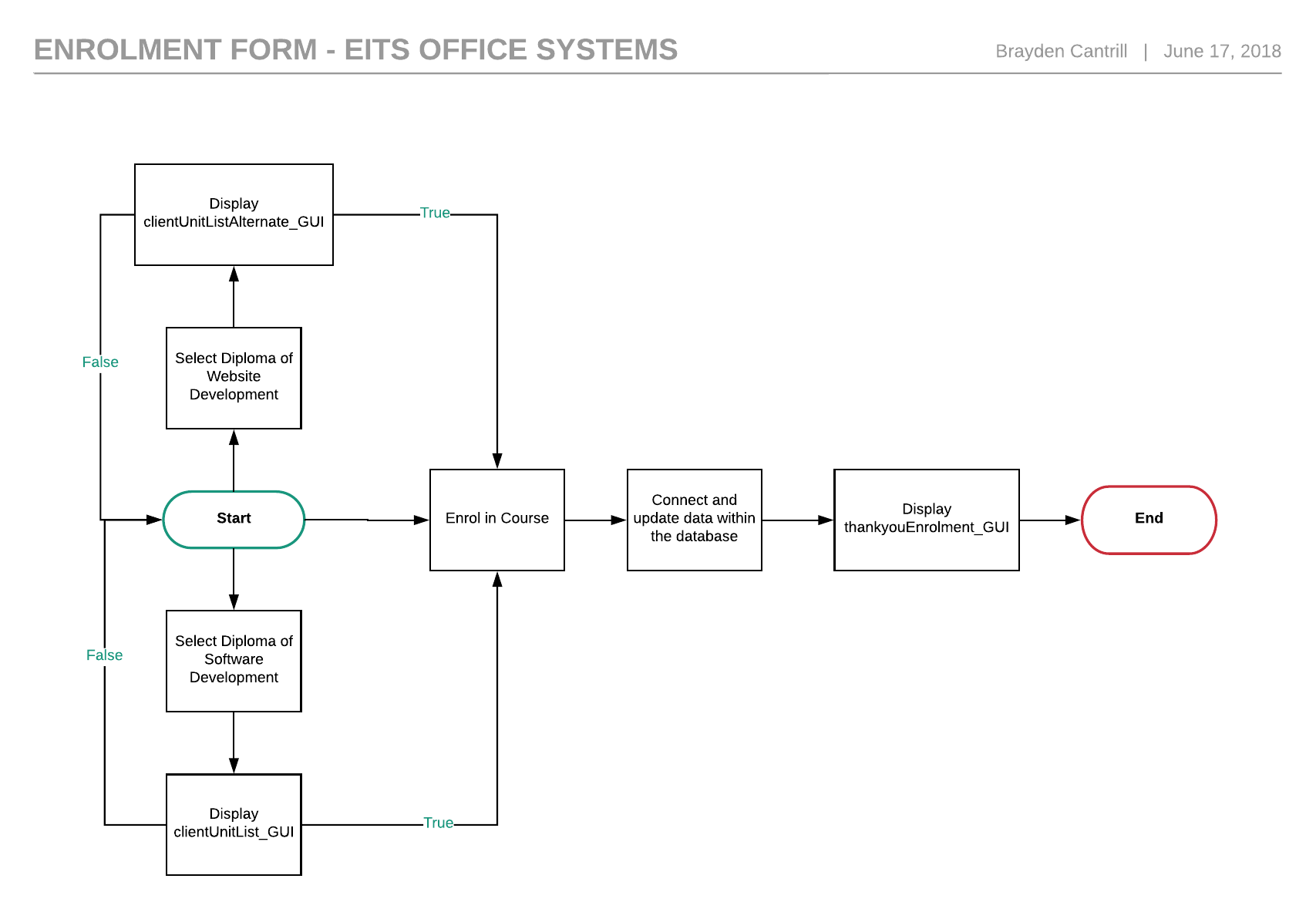
**- Explain the concept of design patterns in the Java.**

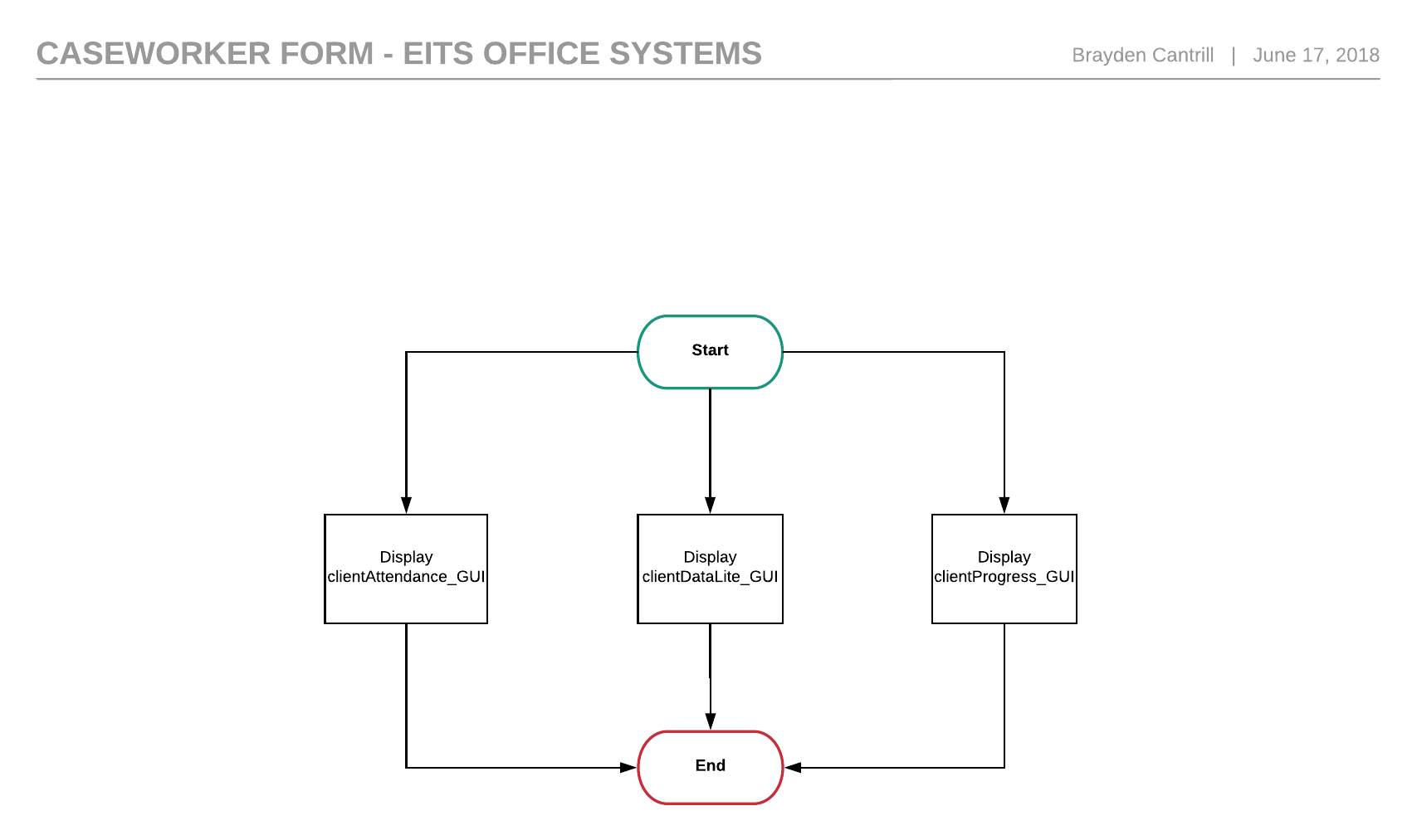
Design patterns are practices that have been used to solve many design problems. Design patterns are used to help devs in producing higher quality software in a considerably faster amount of time. They are also not tied to any type of language or development platform.

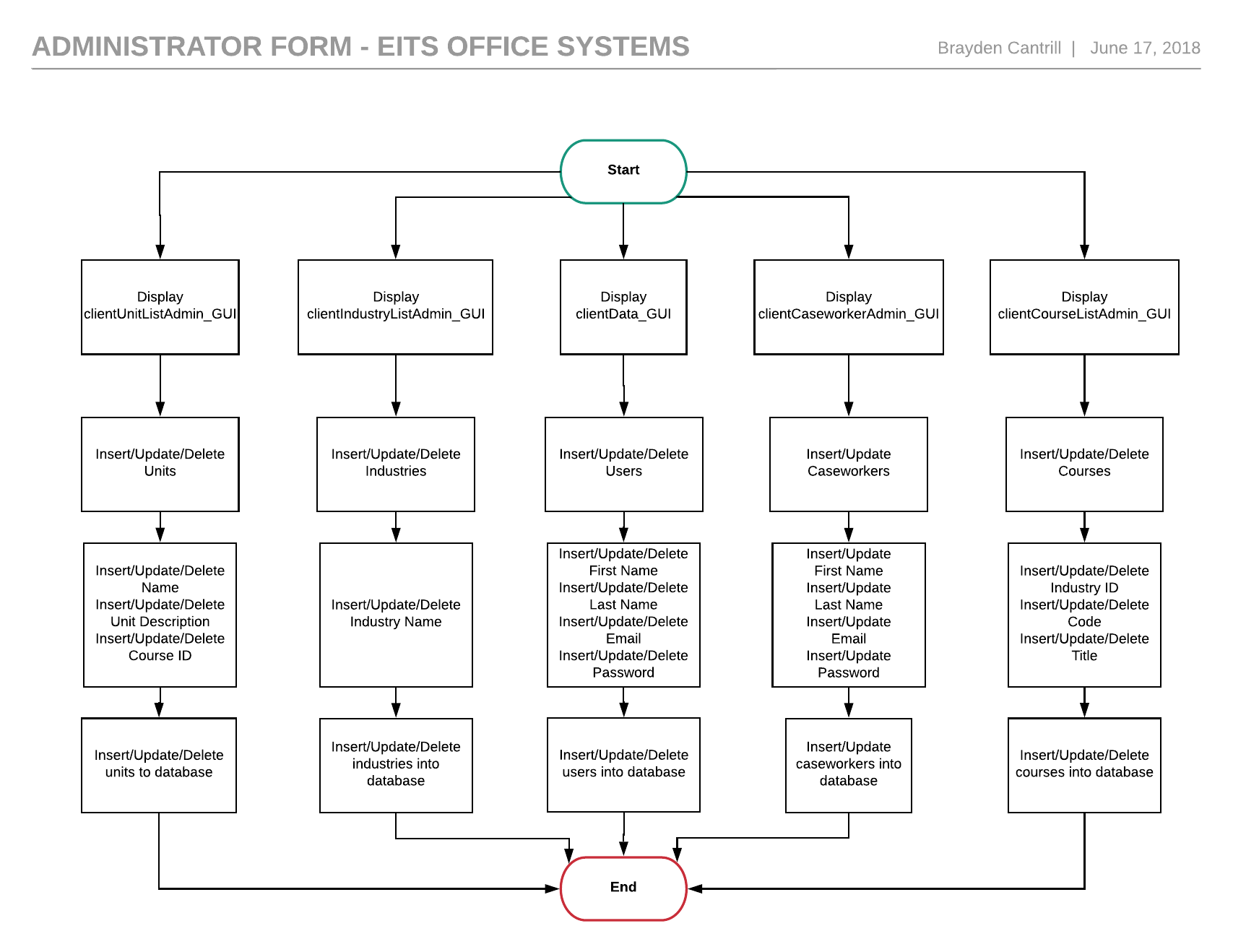
**Diagrams (Brayden):**

****

****

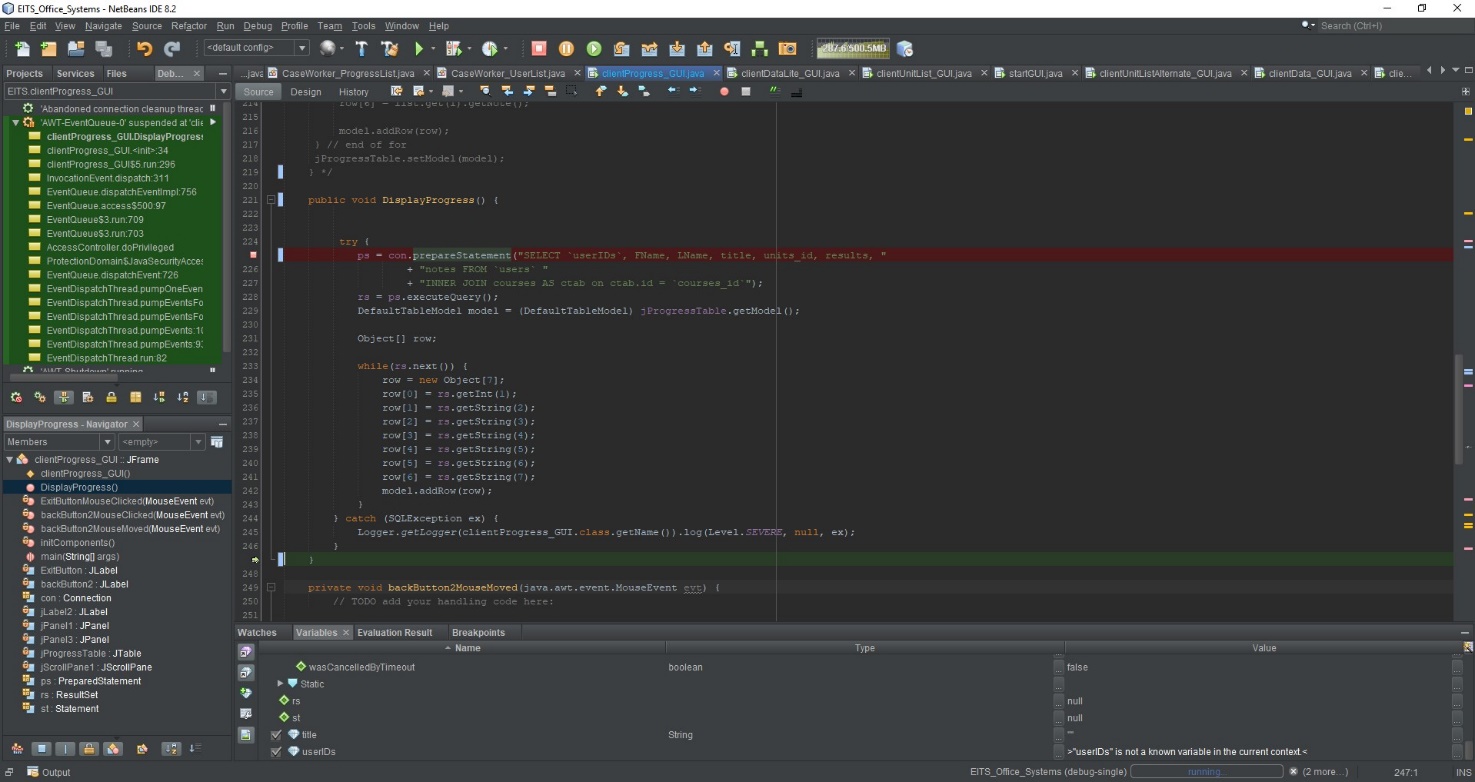
****

****

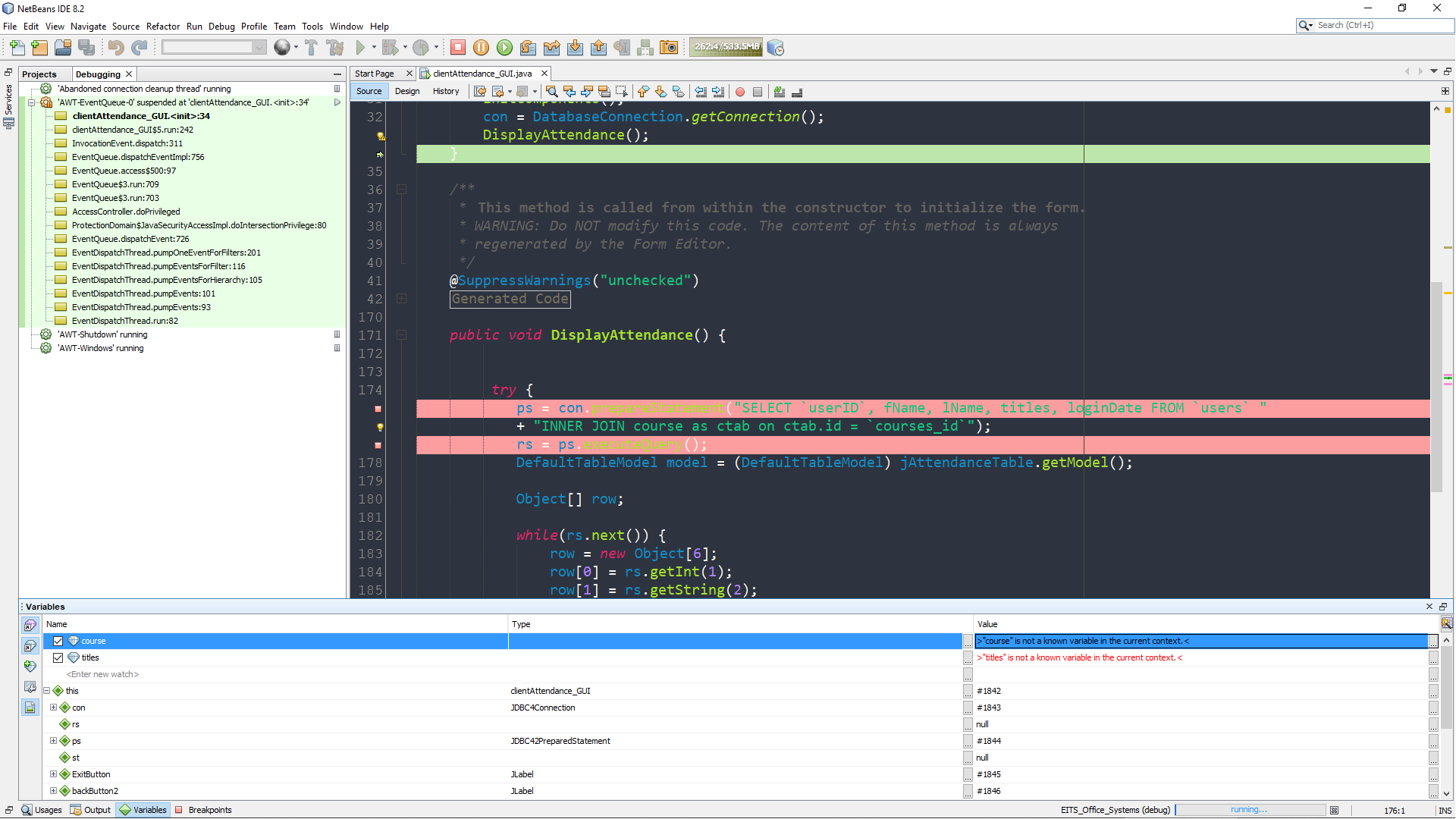
****

**Defect Logs:**

**Debugging Tools in Action:**

**Jakob Debugging:**

**Solution:** JAKOB - "The solutions applied to correct my error was to rename the variables to be the same as what is in the database."

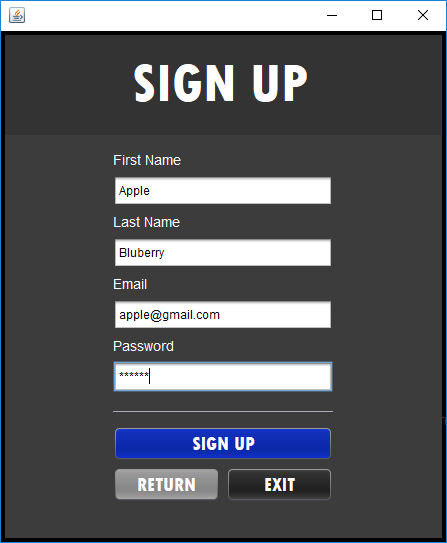
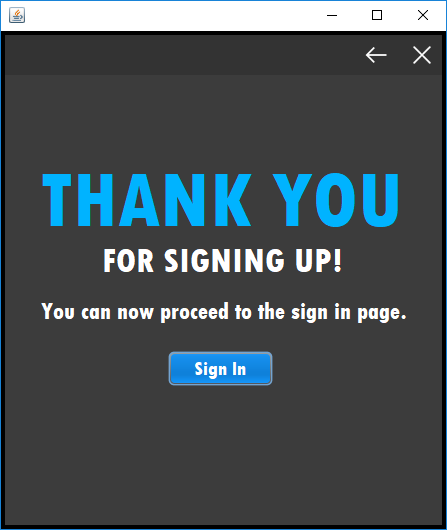
**Brayden Debugging:**

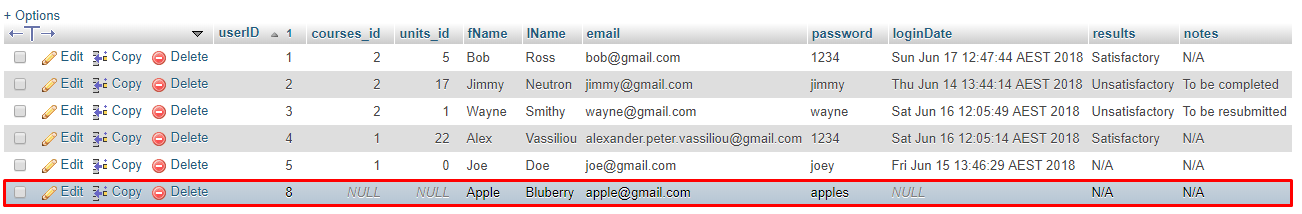
**Solution:** Brayden - "The solutions that I had gathered and applied in order to fix the SQL error due to incorrect values within the statement include primarily renaming the values within the statement to the corresponding values that are within the database. "

**Alex Debugging:**

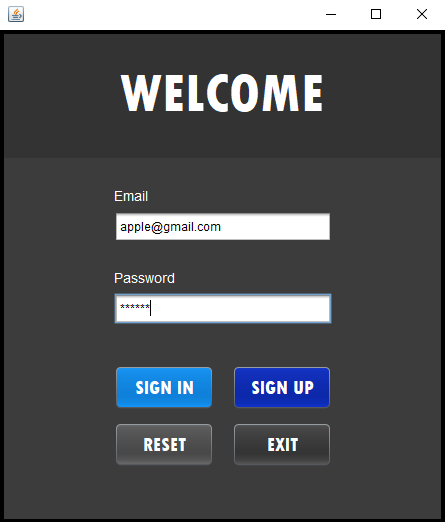
**Testing Performed (Brayden):**

**Sign Up Interface:**

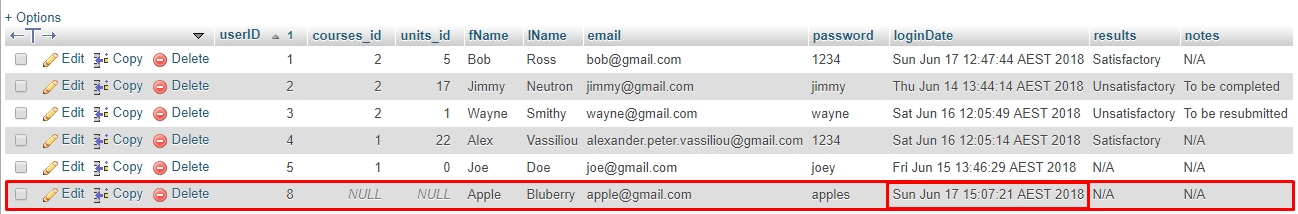


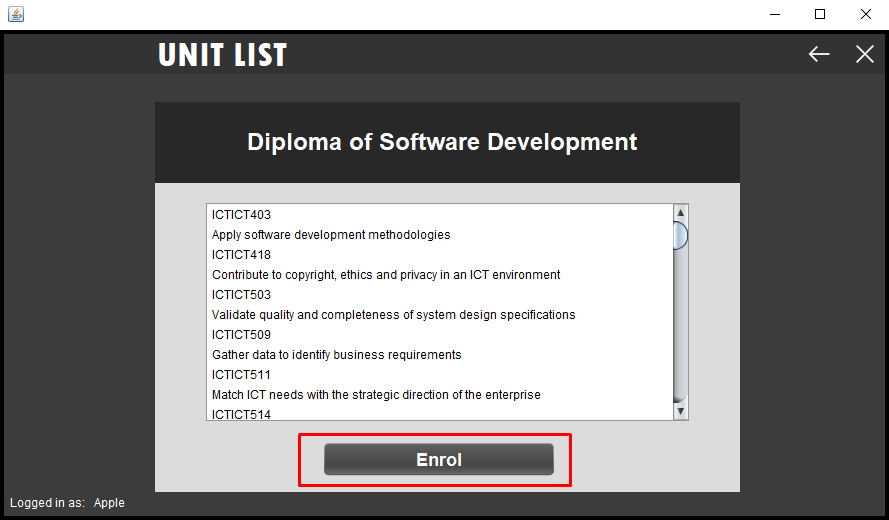


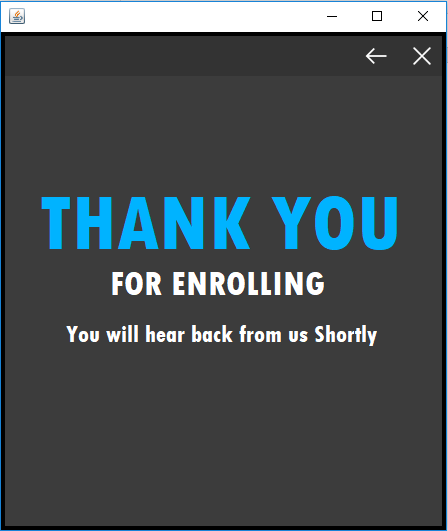
**Sign In Interface:**

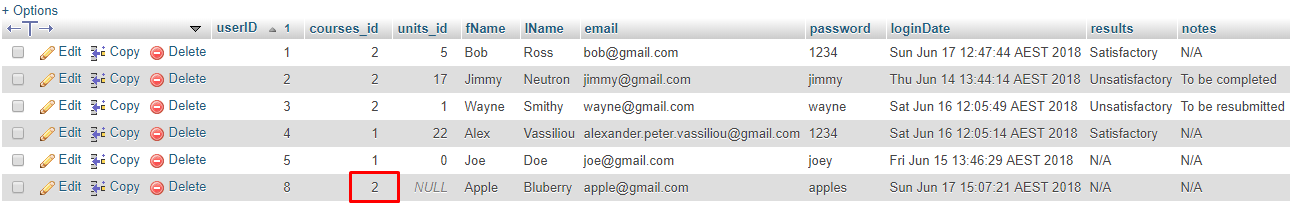




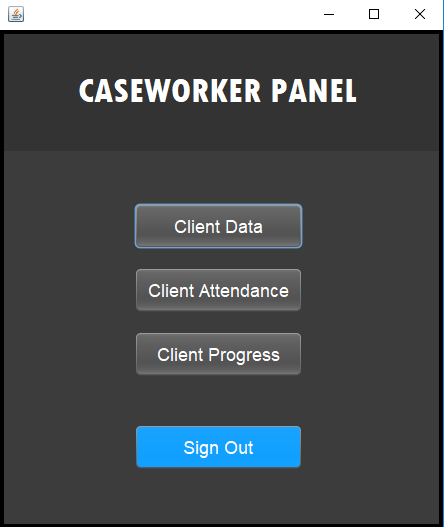
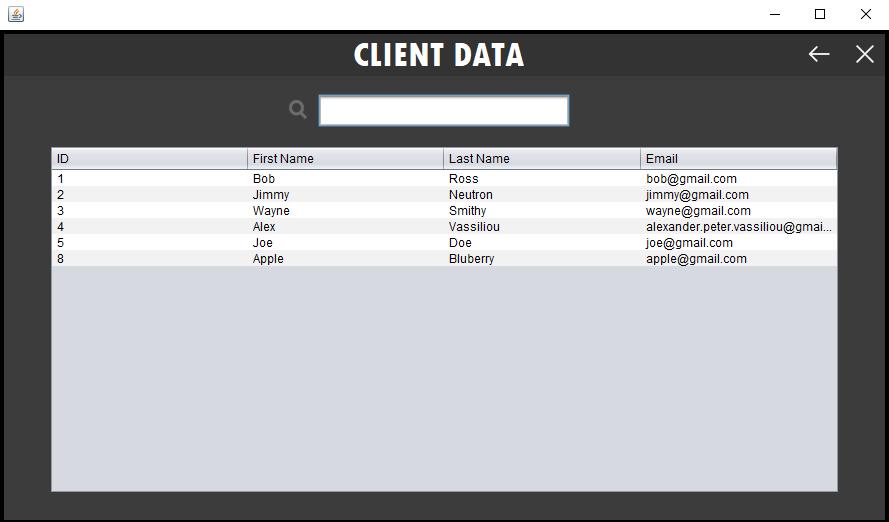


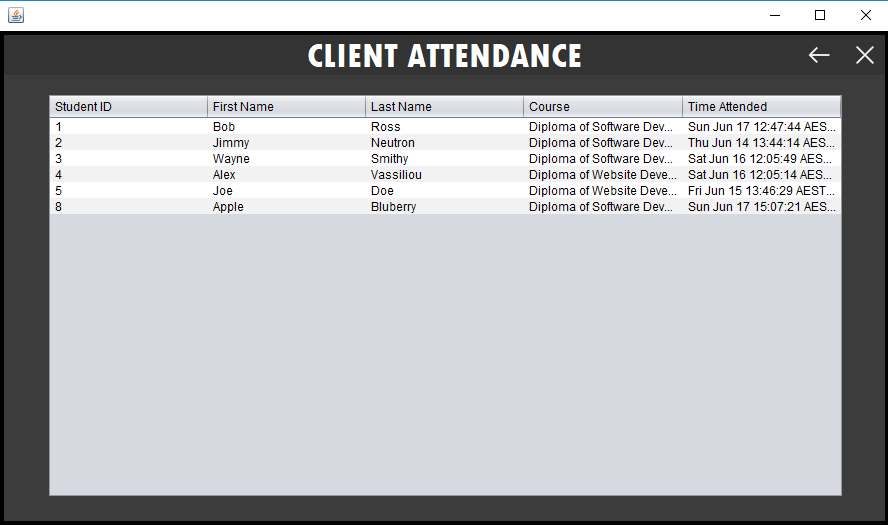
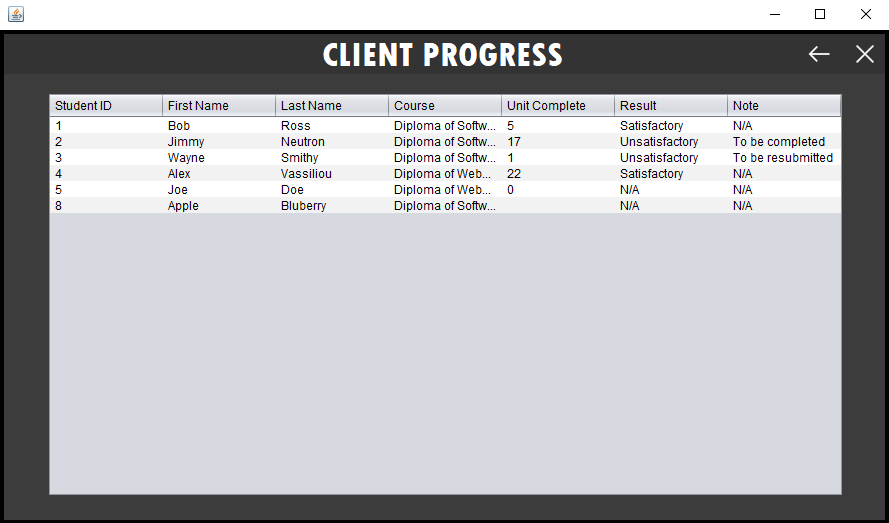
**Enrolment Interface:**



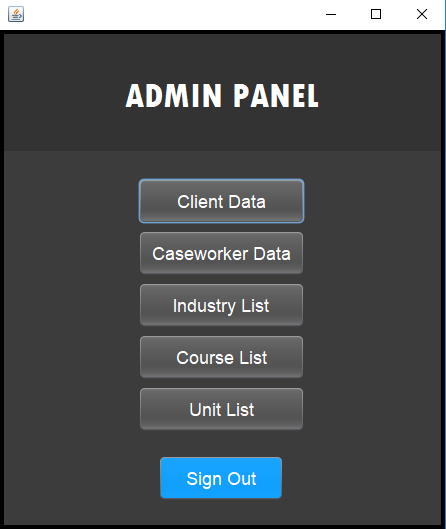


**Caseworker Interface:**

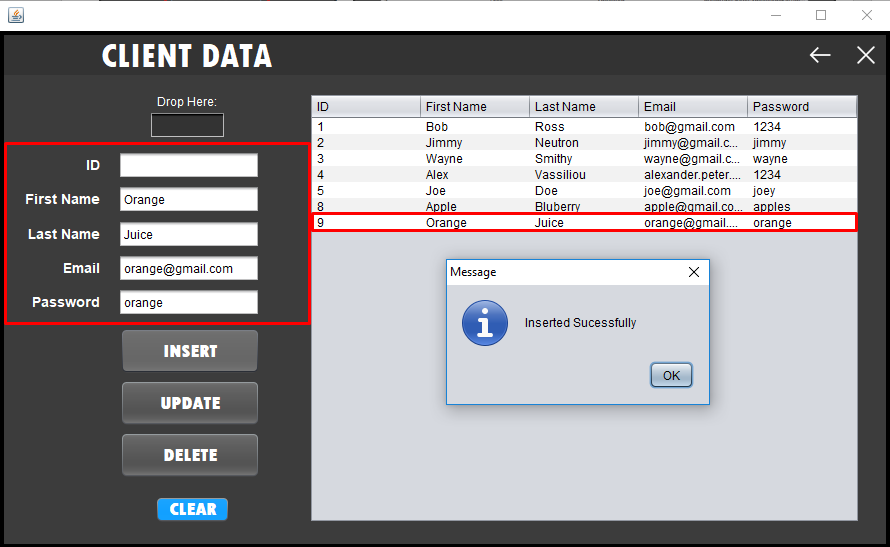
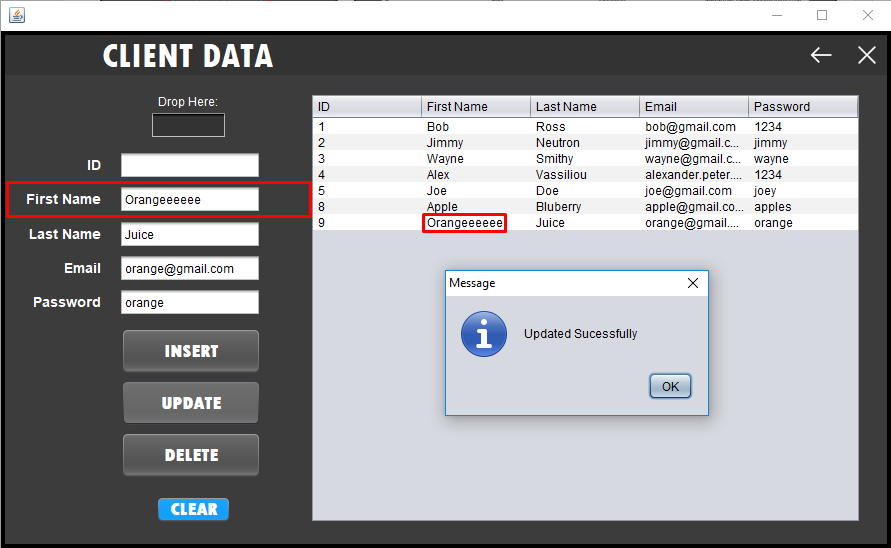


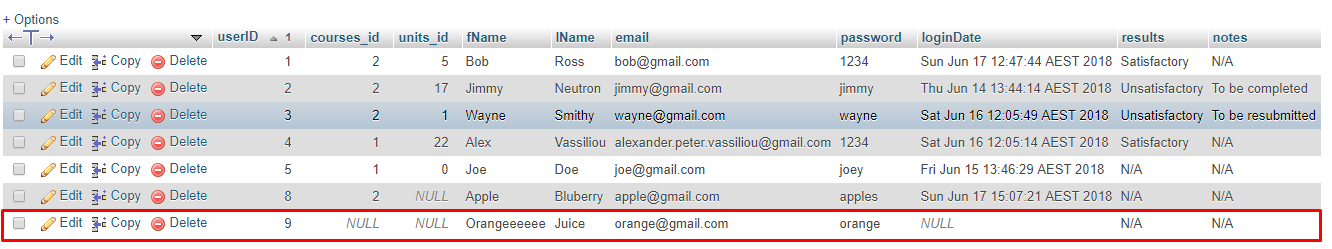


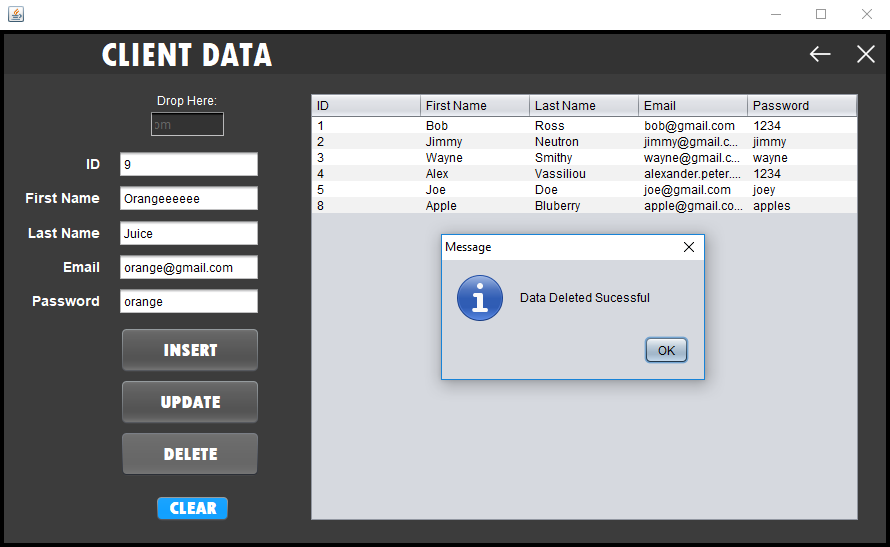
**Admin Interface:**



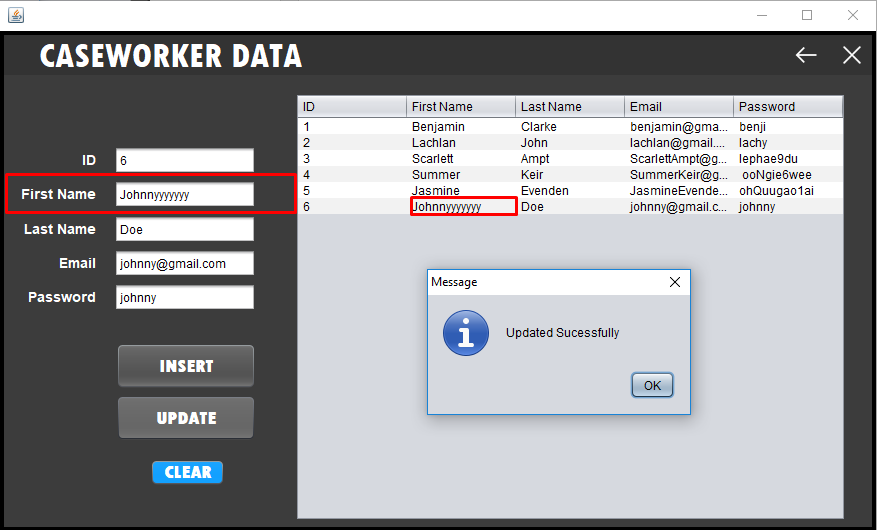
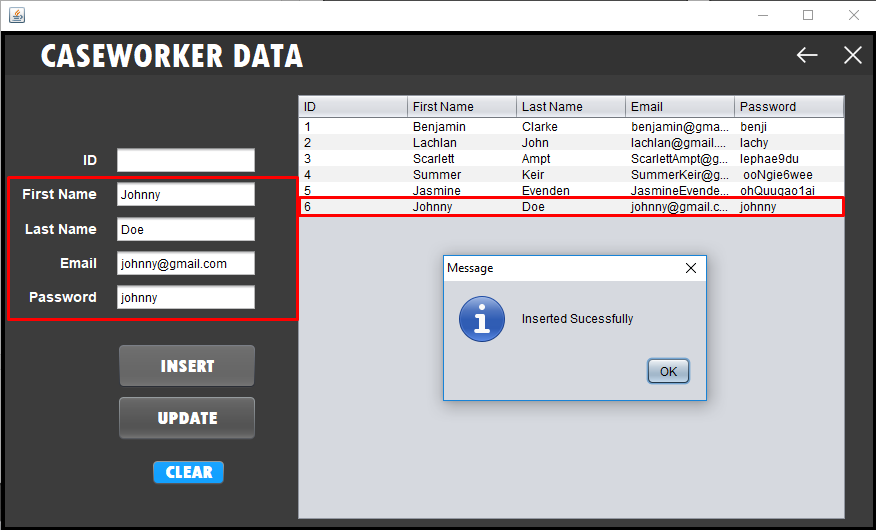
**Client Data Insert/Update/Delete Test:**

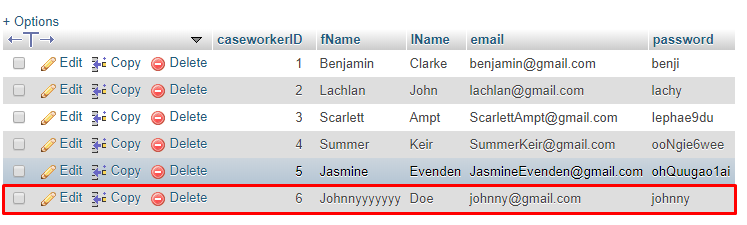




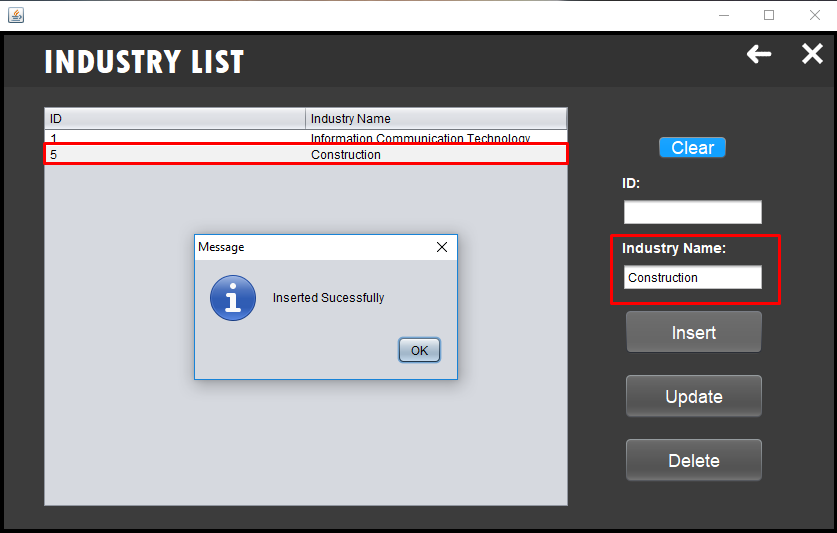
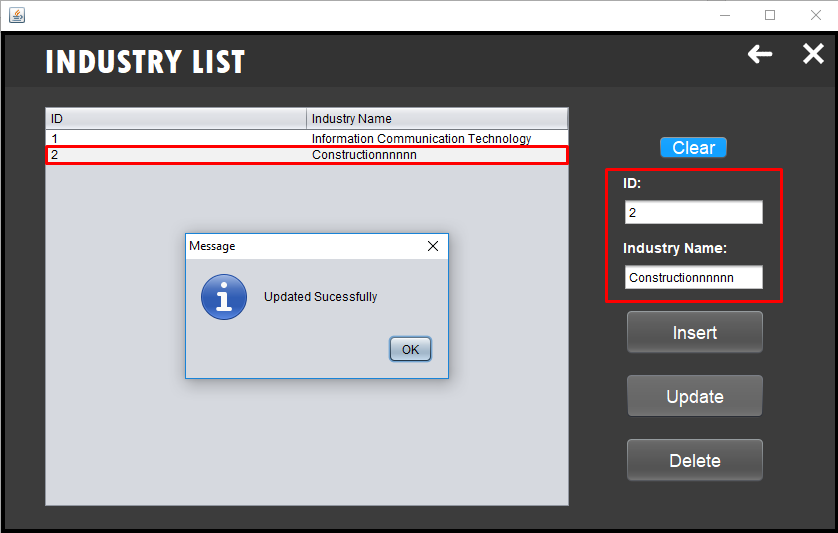


**Caseworker Data Insert/Update Test:**

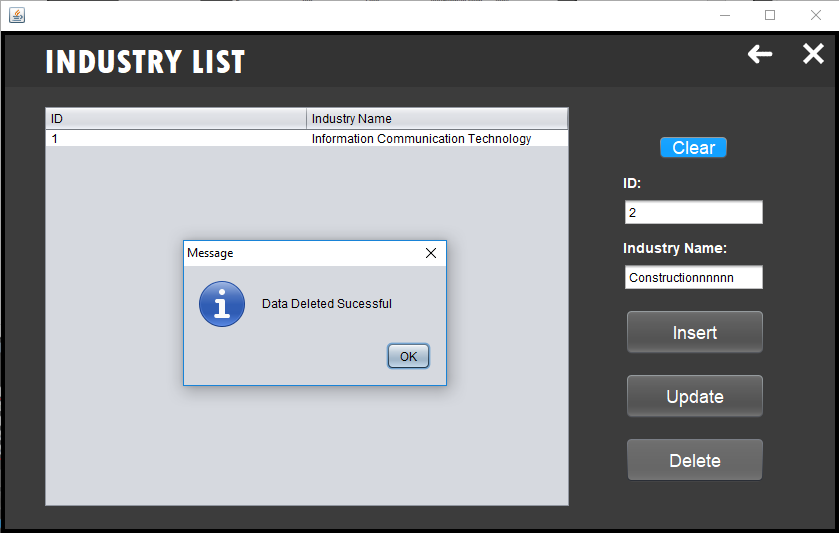




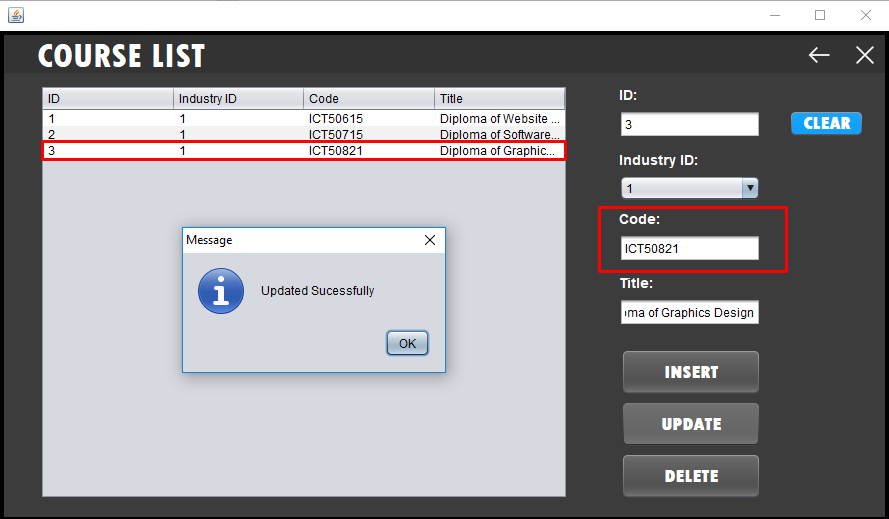
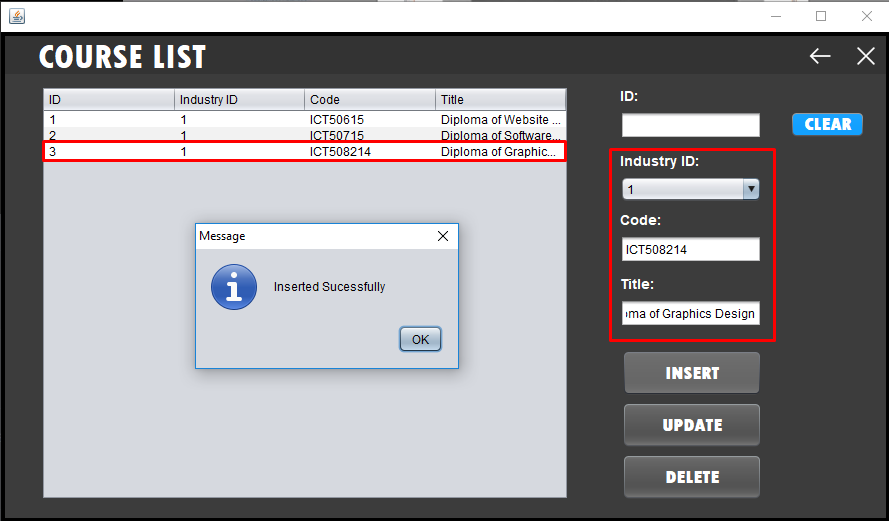
**Industry List Insert/Update/Delete Test:**

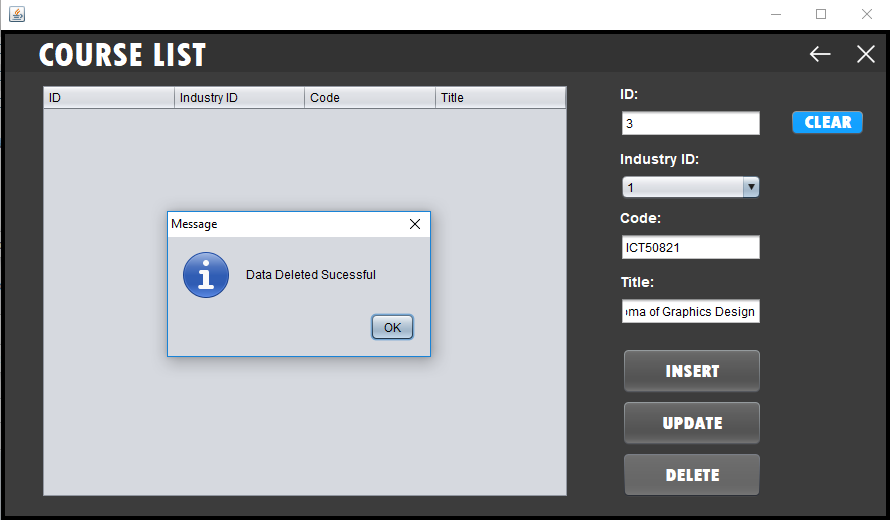




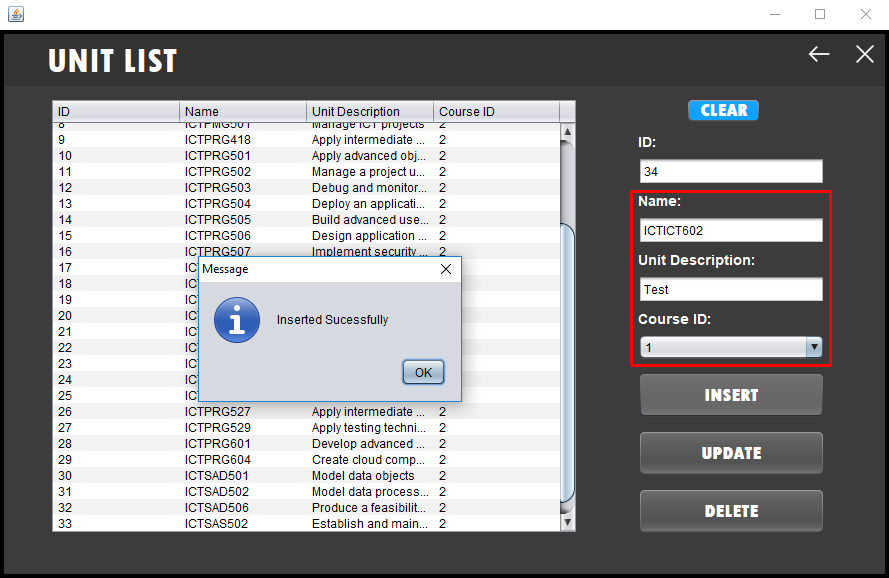
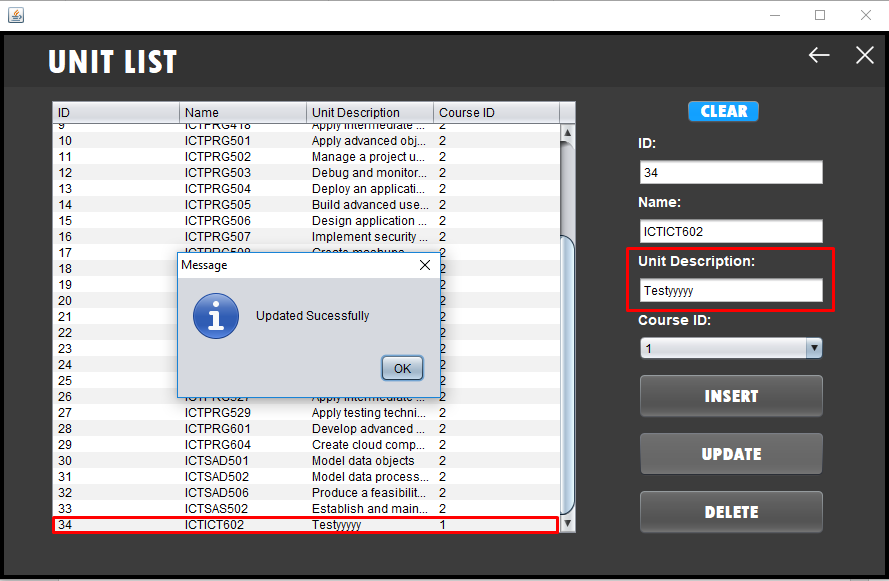


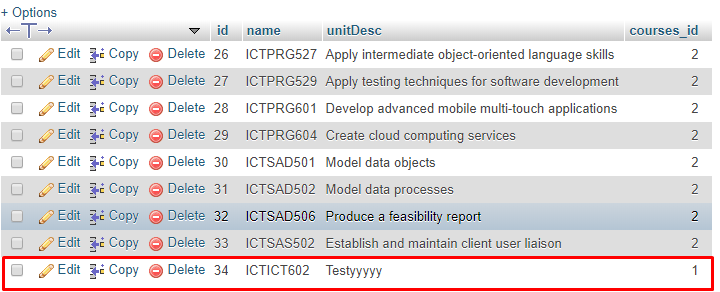
**Course List Insert/Update/Delete Test:**

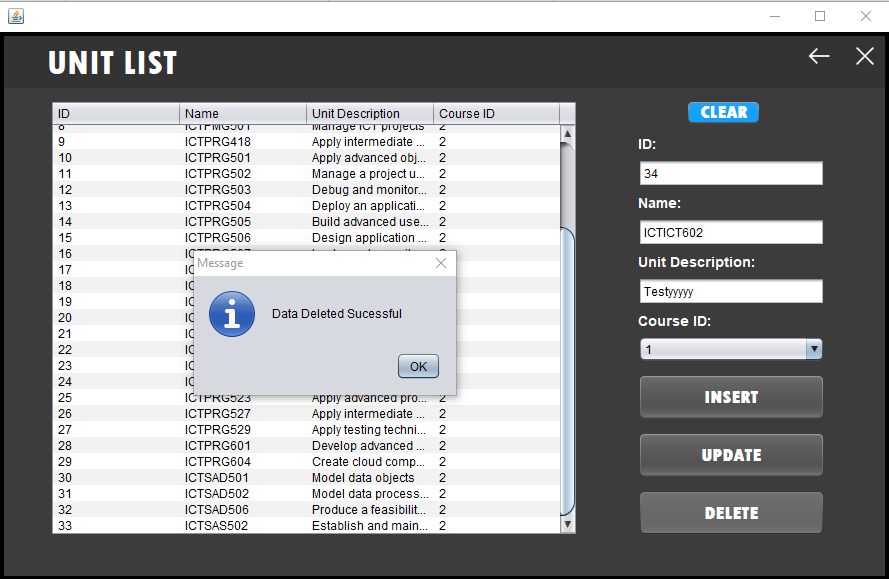




**Unit List Insert/Update/Delete Test:**







**Overall Analysis of Test Results:**

The overall analysis of the test results listed above performed very well during testing and comparing and meeting specific program specifications and gathered project requirements. It can be clearly displayed above that the functionality within the application during testing has been met well with the project requirements with only at least one bug occurring within an interface. The program specifications and gathered project requirements that have been met during the testing of each screen within the application include what the client ‘EITS’ has formally requested. Primarily `EITS` has requested a desktop application that allows clients/students to choose specific courses and track attendance and progress of these students.

Some examples of project requirements that have been met in relation to the test results listed above include the following:

* Welcoming clients to the office and tracking attendance – This was completed successfully during performing testing as it is clearly displayed above specifically for the interfaces of Sign in and Caseworkers. As a user logs in, their attendance is tracked throughout each form when beginning to select their specific course. This attendance is also shown towards caseworkers within the client attendance interface.
* Administrators having the ability in inserting, updating, and deleting data across various tables within the database including client data, caseworker data, industries, courses and units. – This has been demonstrated above throughout the admin interface testing.
* Caseworkers obtaining the capability in displaying client data, client attendance and client progress. This functionality has been displayed above throughout the caseworker’s interface testing.

In testing the course list for the administrators interface a minor bug occurred when deleting a specific course from the jTable and database. This bug consisted of deleting the specific course chosen although it allowed for the other courses included in the table to disappear. One way to resolve this issue is to refresh the specific interface which allows for the courses to appear again.

**Documentation Maintenance:**