**Challenges**

1. Understanding the correct way to write use cases. The common mistakes we made were:

* + 1. Not specify a system response.
    2. Unclear about where the alternate branch off the main flow.
    3. Post-conditions were not stated in active voice.
    4. Incorrect decisions about what stays inside the boundary of the system.
    5. Mentioned the technical details like system components which wouldn’t be visible to the user.

2. We were unsure about the correct associations between classes. We were not clear about which attributes and classes are valid for a correct domain model since we had to only represent the concepts of the system rather than details of the implementation.

3. Understanding the working of Swing, keeping track of the complexity of classes because UI’s get so complicated. We also had a hard time understanding how to move from one Jframe to another once an action is triggered by the user.

4. For the database setup, we had a hard time ensuring which table setup would be the best for storing data and writing queries for correct results.

5. It took us a lot of effort to put together all the pieces of the system together, the server , database and GUI.

6. It’s a lot of work.

**Lessons Learnt**

1. Design documents are under-rated.

We thought writing use cases and designing domain model would be easy, but it took a lot of time and discussions to make them perfect.

2. Team work is not that bad.

We could get a lot of work done during crunch time because everyone contributed which would be impossible to complete alone.

3. Using AWS relational databases

We learnt how to setup our database using AWS which is something we had not done before.

4. Getting swing under control

When we started using Swing it gave us a hard time, but now we are a little better at it. This was something we had not used in the past, so we learned how to work with GUI’s.

5. Design patterns

We learnt how to implement façade pattern and observer pattern in our system to ensure that a front interface to mask the underlying code.

6. Refactoring

We also learned on how refactor our code to make it cleaner and reduce future bugs.