Process for groups to follow when working on project.

This is a blueprint to follow as you work on the project.

1. Discuss the problem as a group; identify the list of use cases. Verify that the list of use cases will allow for the system to function indefinitely and also allow querying to check how the system records the results of operations that create/delete objects and relationships. The use cases should also make the system "friendly for testing", i.e., easy to find errors.
2. Do a couple of use cases as a group.
3. Repeat as needed:
   - Share out the remaining use cases amongst the group members and work individually on the use cases.
   - Get together and look at use cases. They should all have a common "look and feel" and there should be agreement on their details as defined in Step 1.
4. Discuss as a group and decide on the software classes.
5. Do a couple of sequence diagrams as a group.
6. Repeat as needed:
   - Share out the sequence diagrams amongst the group members and work individually on these; preferably sequence diagram should be worked on by someone other than the one who did the corresponding use case.
   - Get together and look at sequence diagrams. They should all have a common "look and feel" and there should be agreement on their details. There may be a need to add software classes; if so, there should be another iteration.
7. Get together and partition the classes amongst all the group members. Each class diagram is defined by the role it plays in each of the sequence diagrams. Each class is assigned an owner.
8. Repeat the previous step with any additional implementation classes that may be needed.
9. Implementing the classes. The **implementation must be done in phases, few use cases per phase, and tested after each phase.**

   `While (there are more use cases)`

   - Decide the set of use cases for the next phase. From the corresponding sequence diagrams, each class owner identifies the methods needed for the class to perform its role.

- Each class owner completes a (partial) implementation of the class, providing the needed methods. The partial implementation is tested using a simple driver program (unit testing).
- “Get together” and test the (partial)system with the partial set of use cases.

```
end while
```

10. Test entire system, populating and saving data so that the system can be seen to be robust for an extended period of time.