Aingty Eung (013462772) Sec: 3

Lessons Learned Paper

1. What I learned: Team Collaborations

- Working in a team of four was a good introduction to a real-world group projects. It is by no mean an accurate representation of industry environment but it is nonetheless a proper introduction. Therefore communication skill is needed for a clear understanding of the proposed project.

Positives(s):

- Having one person be the leader makes decision-making much faster.

Negative(s):

- Difficult to get everyone's inputs while team leader takes charge. Resulting in the leader doing the majority of the works while the team members stay put and agree.

2. What I learned: Diagrams (Classes and Entity Objects)

- The beginning of works as a software engineer requires a lot of conceptual design. Since it is difficult to present the client(s) with a tangible design or example, diagrams are needed to present the idea. This is an important stage for software engineer because it is the blueprint for a successful project

Positive(s):

- Similar to designing a Database, it is fairly simple to draw out the diagrams. Using what we've learned in CECS 323 Database Structure.
- It was very helpful to start the project with these newly created class and entity object diagrams. Since the early stages of the project is subjective, having a strong foundation is needed.

Negative(s):

- At the beginning of the project, one mistake on the diagram could lead to system flawed.
- Everyone involved must understand the annotations, shapes, etc of the diagrams. So diagrams alone would not suffices.

3. What I learned: Functional vs Non-Functional Requirements

- When implementing a Use Case, one essential detail is the Functional Requirement. This determine the feature/uses of the project. Without the Functional Requirement, the Actor/System would not be able to perform its job, therefore no software exists. On the contrary, the Non-Functional Requirement is not so important in the sense that it exists for extra functionality. A System exists fine without them.

Positive(s):

- This clearly defines each Use Case and provide it’s full functionalities. Since client(s) only want(s) results, this is fair representation of the proposed software.

Negative(s):

- It is difficult at first to determine which feature is a Functional Requirement and which is Non-Functional Requirement.
- We’ve made many mistakes when creating the tricky Non-Functional Requirements. Since it is not a required by the system, it is important to note that it doesn’t need to exist.