# 7 – Plan

# **7.1 – Activities**

### Activity 1 - Research

Research is described by gathering key information from various relevant sources. This gathered information contains only the necessary details that are pertinent to the project.

### Activity 2 - Planning

Before a project begins its development, a plan must be put in effect. This leads to an organized project where all members can track the work they have done and the work they have yet to complete.

### Activity 3 - Discussion

In any project that involves multiple members, discussion is key for an optimal development. Sharing ideas and reviewing each other's work can bring out effective ideas with a lower chance of errors.

### Activity 4 – Environment Creation

When the technologies required have been decided upon, it is crucial to then set up said technologies to allow the team proper access to everything they need to complete the application.

### Activity 5 - Programming

Programming, both client-side and server-side, is key for the development of an application. The planning and ideas can be optimal but will only be valuable through effective programming.

### Activity 6 - Testing

It is impossible to truly know if a software is effective until it is properly tested. Many unforeseen errors can be uncovered through testing, and releasing an application before testing it can lead to a very negative experience for the user.

### Activity 7 - Debugging

The errors found through testing must be solved, which can be complicated when there is a multitude of code involved. Different segments of code must be analyzed for errors and programming solutions must be found.

### Activity 8 - Documentation

In order to be fully aware of the progress of a project, every action and plan of action must be well documented. It is only through organized documentation that a project can effectively be completed and ridden of errors.

# **7.2 – Artifacts**

### (1) Deliverable 0

Deliverable 0 is used not only to solidify system information and create a rough draft of the project’s Domain Model, including describing each major portion of said model, but it also serves to solidify each member’s role within the team. This artifact serves to outline what will be accomplished in the final project, as well as introducing the entire team.

### (2) Deliverable 1

Deliverable 1 serves to describe, in much greater detail than Deliverable 0, the project as well as its goals, requirements, domain model, constraints, resource allocation, and plan for implementation (among others). As this artifact covers most aspects of the application, including a schedule for the either project, it is incredibly important with regards to the development of said application.

### (3) Deliverable 2

Deliverable 2 serves to outline the architecture, class diagrams, and design diagrams in complete detail. The deliverable also incorporates reports on rapid prototypes, and analysis of the risks, scope, and estimates involved. Due to its technicality, this artifact is extremely relevant to the programmers of the team in order to create a proper idea of how the project should function.

### (4) Deliverable 3

Deliverable 3 serves as a summary of all testing cases previously reported, as well as what needs to be accomplished prior to the final stage of production. This artifact includes testing reports for all items and cases tested, and documentation for the system (including installation and user manuals), as well as a final cost estimate of the project.

### (5) Basic Project Structure

As a result of discussion, a basic foundation for a proper course scheduler was achieved. In addition, methods of communication between members (Slack and Facebook) and team member roles were determined and agreed upon. This artifact served as the beginning of the entire software development process, and created the initial foundation for the final project.

### (6) Initial Diagrams

Members of the team constructed various diagrams (including ER, Domain and class) in order to better visualize the relationships of the main sections of the project. The diagrams constructed in this artifact serve as the visual foundation for the programming development to come.

### (7) List of Features and Technologies Required

Once a solid foundation for general functionality was created, the team discussed exact features they believed the application should include. Once these features were determined, the technologies required to achieve this goals were discussed and decided upon. This artifact serves not only as a measure for what the team wishes to accomplish, but also as a way to determine which technologies will make accomplishing said goals as easy as possible.

### (8) Use Cases

Use cases describes all possible actions performed by a user, or an "actor" in a system. The work done in the backend by the system itself is not indicated through this type of model. This artifact allows the development team to see an overview of the user's experience and optimize it.

### (9) User Interface

A functional system can only be brought to its full potential through an optimal user interface. An effective UI is clear, concise, and without clutter. Before any programming begins, the user interface must first be designed in an effective manner. The client-side programming team must always keep this artifact in mind when implementing the layout of the application.

### (10) Design of Script to obtain Concordia Class Data

Since a large amount of data is required for the integration of our project, this is an important artifact because it ensures that correct data will be populated in our SQL tables in the database.

### (11) ER Diagram

Once the data is obtained using the script and put into the database, the team created ER diagrams to show relationships between objects/tables derived from the domain model. This will help the team in the configuration of the manipulation of the data. This artifact is closely tied with the main functionalities of our system since the data used in the software must be well-represented and easily accessible.

### (12) Architecture Programming, Use Cases

The main part of the programming takes place within this artifact because the code is created which makes all the designed architecture and use cases functional. If this artifact did not exist, the system would not exist.

### (13) Scope of Project

Since the programming for this system is still ongoing, the scope of the project is subject to change. As the project progresses, the scope will change depending on what the team believes is still feasible to implement or not. This artifact will lead to the final product as element the team deems as nonessential/infeasible will be removed.

### (14) Test Cases

As functionality is added by programming, test cases are created to ensure that the newly added operations of the system are fully working. Most importantly, these test case are largely based on the use cases and also will test the functionality of the database as well. This artifact is essential for quality assurance and a working final product.

### (15) Performance

The Sub-team members responsible for QA will try out each test case on the system and determine whether or not the system operation is fully functional. This artifact will be the final confirmation that newly added system components are working and before they are put into the final polished product.

(16) Design of UI/Aesthetics of Application

The attractiveness of the product is important for the usability of the product. If users do not find the website esthetically pleasing, they will refrain from visiting it. On the contrary, if they like the aesthetics of the website, the users will appreciate the system more. This artifact ensures the final system is visually pleasing.

### (17) Debugging / Bug Fixes

During both technical and user testing, several errors can be uncovered but their source relies unknown. The programming team must discover the errors in the code and find effective solutions to prevent them from repeating.