# IT 315 Final Project Part I Solution Submission Template

This template is a guide for you to organize your information. To complete it, **replace the bracketed text with the relevant information.** Some areas may be too large or too small for the information you’re inserting. Adjust the size of the areas as necessary.

**Name:** Walter Hernandez

**Date:** May 21, 2023

1. **Creation:** Generate your student information system (SIS) use case diagram. Refer to textbook pages 121–129.





| Use Case Name:  Student Info System | ID:  101 | Importance Level:  High |
| --- | --- | --- |
| Primary Actor:  Student | Use Case Type:  Detail, Essential | |
| Stakeholders and Interests:  Students- require an SIS for registering.  Enrollment Staff- require a SIS to be able to manage registrations and course offering. | | |
| Brief Description:  The student information tries to make registering for courses simple and efficient. It isn’t complex in the sense that it might give you future problems. Instead, it is preloaded with student information so that if there is an error, the student will receive feedback on pre-requisites, requirements, and limitations. The software also assists the enrollment staff with registration and management. | | |
| Trigger:  Students need to register for classes to obtain a degree.  Type:  External | | |
| Relationships   * Association:   The students and enrollment staff are associated directly with their abilities and direct interface to the SIS. In the beginning, they can directly associate with the connection process. From web to mobile, and what their limitations are such as the enrollment staff being able to update and change records.   * Include:   The direct results from login in or reply messages are included since they occur every time the user interacts with the associations. The ability of CRUD is also included in the records since the enrollment staff can CRUD every time.   * Extend:   The events that will not occur every time are included in the extend category. I placed the mobile app, or desktop app since they won’t occur every time depending on the users.   * Generalization:   The biggest generalization is the derivative of class records. Class records depend on having information from course records. It is dependent on the other for existence. | | |
| Normal Flow of Events:  The normal flow is linear since the user has simple options such as registering, selecting options, and selecting their course. The flow for a student is as follows, they choose their mobile or desktop app, login, register, input course record information, and finally move forward to class records. Further on, they’ll receive replies, acknowledgments, and availabilities. For the enrollment staff it is similar, except that they are able to CRUD. | | |
| SubFlows:  The available updatable information has been placed in Sub flows. Prerequisites are also featured. | | |
| Alternate/Exceptional Flows:  The ability to loop back or terminate the session falls under this category. The loop back to step B in the process. | | |

**Use Case 1 description template:**

Use-Case 1 demonstrates the overall linear progression and basic capabilities. It shows the entry level user to machine interface. Their options from the start are login, and so-on. Furthermore, the other Use-case goes in detail with the deeper process.

**Use Case 2 description template:**

Use-Case 2 is the functionalities and capabilities of the SIS. Its main function is to store and progress the user to the further steps. It mainly shows what information it keeps and saves for future use.

**Use Case 3 description template:**

Use Case 3 is all about the registration process. It shows at what point information is entered, and at what point the SIS will hand over more information. It also holds the rules so that the users can progress.

**Use Case 3 description template:**

Use-Case 4 is for the sole purpose of continuing the session or closing the session. It tells you the options of what occurs when the class registration is performed, and the final moments.

1. **Testing:** Verify and validate your use case diagram and use case descriptions against the SIS requirements definition.

Check your diagram against the SIS requirements and write this review. In doing verification, the objective is to make sure that you are building software according to user specifications. Ask questions like these: Does each use case have the required functionality? Do all the use cases combined perform as a complete coherent system?

The SIS is validated with the Use-Case diagrams. All the functions have been performed linearly to verify progress. The system has a validated start and end process. It includes loops and rules. The rules are valid when all the criteria are input into the SIS. All possible alternatives have been thought of for validation.

1. **Approach Explanation:** Explain your approach to the problem, the decisions you made to arrive at your solution, and how you completed it.

Explain why you chose these particular use cases and why you chose the relationships between them. Explain your approach to creating your functional model and the design decisions you made to create it.

The approach was based on a beginning, middle, and end. The beginning has all the available options from multiple actors. The middle contains the meat of the process. The middle was then given further details so that the SIS can move in a linear motion. The approach came from what are the main actors on the SIS? And to what if? Or what if that? Scripting also assisted in the approach.

1. **Self-Reflection:** Reflect on this experience and the lessons you learned from it.

These are your reflections on what you learned. Address what you found challenging and what you found easy. Discuss your experience creating your functional model and the lessons you learned from it. Specifically, draw connections between your experience and the object-oriented techniques and methods discussed in this course.

Designing by using UML is a big help when it comes to developing a concept for a software. The extend, include, actors, and generalizations help the designers visually see the directions of motions within a software. It is much easier to demonstrate visuals for certain things such as process than it is to verbally explain. This is why OOAD and UML are so important. I can see myself using such a process before beginning a script on a software. There must be a plan and decision between team members. A team is able to make better decisions when there are visuals and everyone can see the process through a whole.