Software Project Management Plan (SPMP) For:

*Learning Management System (LMS*):

GatorConnect

Team: Anam Ali, Raj Asarpota, Tariq Ali, Kedar Jagtap, Ashton LeBlanc

Spring 2020

Due Date: May 3rd 2020

GitHub link: <https://github.com/rajasarpota/LMS-Project>

Table of Contents

1. Introduction

1.1 Project Overview

1.2 Project Deliverables

2. Project Organization

2.1 Process Model

2.2 Organizational Structure

2.3 Project Responsibilities

3. Managerial Process

3.1 Management Objectives and Priorities

3.2 Assumptions, Dependencies, and Constraints

3.3 Staffing Approach

4. Technical Process

4.1 Methods, Tools, and Techniques

4.2 Software Documentation

4.2.1 Software Requirements Specification (SRS)

4.2.2 Software Design Description (SDD)

4.2.3 Software Test Plan

4.3 User Documentation

Diagrams located here

5. Schedule

5.1 Schedule Overview

**1.Introduction**

The purpose of this software project management plan is to show the process of building this Learning Management System (LMS). The learning management system is used to store information of a potential student. The student will be able to view their own grading information but will not be able to change anything. The admin user will typically be for the teacher or high ups to change any information of the student.

**1.1 Project Overview**

The objective of this project was to create a Learning Management System (LMS), that could be used by student users and university admin. The LMS helps keep records of a student’s courses and respective grades. The LMS permits admins to create content (student information, and course list), organize it into courses, enroll students to courses and assess the students’ performances (via grades). Functions such as configuring the LMS, such as adding, editing, deleting, and monitoring student accounts, enrollment, courses, and staff members are under the jurisdiction of the administrators. The administrator role is for faculty so they can also access the students’ progress and edit it. Students can easily view the classes they have enrolled in, grades, and the GPA.

**1.2 Project Requirements**

The learning management system has certain requirements that need to be met. These are the following requirements

* The software system only stores and retrieves students’ partial information in the current semester and other basic information including student’s name, student’s ID, registered courses in the current semester, each exam’s score in one course, GPA calculation in the current semester.
* The software system needs to have data record files to build this software system. We did this through building a database system
* The software should have a Graphical User Interface (GUI), in order to create an easy use for users.

**1.3 Project Deliverables**

Initially our team was supposed to have a presentation in late April or early May to show the client how we created this software and how the software works. The final project submission was due April 26th. Unfortunately, the pandemic caused by COVID-19, we are no longer able to present the information. Instead, we were given a final due date of May 3rd with a complete submission.

This submission list includes:

* Software Project Management Plan (SPMP)
* The source code and the artifacts it used
* UML diagrams
* Version control documentation
* Test cases
* Data storage files

**2.Project Organization**

As a team we decided to go with rapid prototype and evolutionary model for the lifecycle model since it was easier for us to work

2.1 Process Model

After we started the project, we weren’t going to change the model. We chose rapid prototype and evolutionary life cycle models. This particular model allows us to revisit the requirements and make changes if the client requires, this model makes it easier for us to go back and make those changes. The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle. We felt this was much easier because of the circumstances of this semester.

We also initially went with storing our information in a text file, but we felt as though the code was very inefficient without a database. Thus, we transitioned into building a database via SQLite 3 (using DBbrowser to view and populate the database)

2.2 Organizational Structure

We used a democratic team model for the organizational of this project. Each member of the group had some sort of experience in each part of the project. Thus, we felt each individual can cover some aspect of the project. To make things easier and to not have a giant load on an individual member, we tried to work in pairs. After every meeting we were following where we stood, discussing the deadline and how we can divide the work.

2.3 Project Responsibilities

|  |  |
| --- | --- |
| Task | Project Management |
| UML Diagrams Ect. | Raj, Anam, Kedar |
| Database Compilation | Kedar and Tariq |
| Function To associate with database | Anam and Raj |
| Population Database | Ashton |
| GUI structure built | Anam and Raj |
| GUI structure finalize | Tariq and Kedar |
| SPMP report | Ashton and Kedar |
| Test cases compiled | Everyone |

**3. Managerial Process**

This section of our SPMP covers the management process of this project.

**3.1 Management Objectives and Priorities**

The goals and objectives of this project were primarily focused on understanding the process of making a software. We focused on setting up and following a team-based life cycle model, prioritizing as a team to commit to a timeline and using that model to help us throughout the project. We wanted to understand how a piece of software was built and executed, whether it be coding the project together in atom, or working on the setting up the program display. We wanted to make sure our code was not buggy and that our U/I felt and looked clean and simple.

**3.2 Assumptions, Dependencies, and Constraints**

At the beginning of the semester when we founded our team, we assumed that we would be able to get together weekly on campus to discuss and make progress on our code, display, etc. Due to the COVID-19, we really had to depend on each individual in the group to complete tasks set out by the group. We were limited in a lot of ways on really getting together and working because of the pandemic. Fortunately, we had programs like Discord and Google Hangout which allowed us to all get in a call and discuss issues, problems, and the project as a whole. Outside of the COVID-19 situation, we really had no other constraints. We had to depend on each individual in the group to complete tasks set out by the group.

**3.3 Staffing Approach**

These are some of the skills needed to complete this project:

* Having a basic programming knowledge of Python
* Understanding how to manipulate a GUI and database
* Knowing how to build class diagrams, and use cases
* Being comfortable working as a team, and completing tasks given to you
* Following a team-based life cycle to completion of the project to ease difficulty of the project

**4. Technical Process**

**4.1 Methods, Tools, and Techniques**

* Computing system: Windows/macOS
* Development method: standard policies
* Team Structure: Democratic
  + GatorsConnect was developed by using a democratic team model. The concept of a democratic team model is surrounded by the idea of egoless programming. This is the basic concept because it allows the team to develop a group identity. They now collectively act and one; rather than individuals and their egos vying for attention and appreciation. The ultimate product belongs to all the group members. Decisions are made in a democratic fashion and finding faults is a group effort.
* Programming language: Python
* Tools: PyCharm Professional, Python 3.8 interpreter IDEs, DB browser, sqLite 3, Atom, PyGUi, Visual studio

**4.2 Software Documentation**

Specify the work products to be built for this project and the types of peer reviews to be held for

those products. It may be useful to include a table that is adapted from the organization's standard collection of work products and reviews. Identify any relevant style guide, naming conventions and documentation formats. In either this documentation plan or the project schedule provide a summary of the schedule and resource requirements for the documentation effort.

To ensure that the implementation of the software satisfies the requirements, the following

documentation is required as a minimum:

**4.2.1 Software Requirements Specification (SRS)**

The learning management system has certain requirements that need to be met. These are the following requirements

* The software system only stores and retrieves students’ partial information in the current semester and other basic information including student’s name, student’s ID, registered courses in the current semester, each exam’s score in one course, GPA calculation in the current semester.
* The software system needs to have data record files to build this software system. We did this through building a database system
* The software should have a Graphical User Interface (GUI), in order to create an easy use for users.

The features available to Admin:

* Add/Remove Student
* Manage Existing Students
* Manage Existing Logins
* Enroll Students Into Class
* Manage Existing Enrollments
* Add Student Assignment
* Manage Existing Assignments
* Add/Remove Grade
* Manage Existing Grades

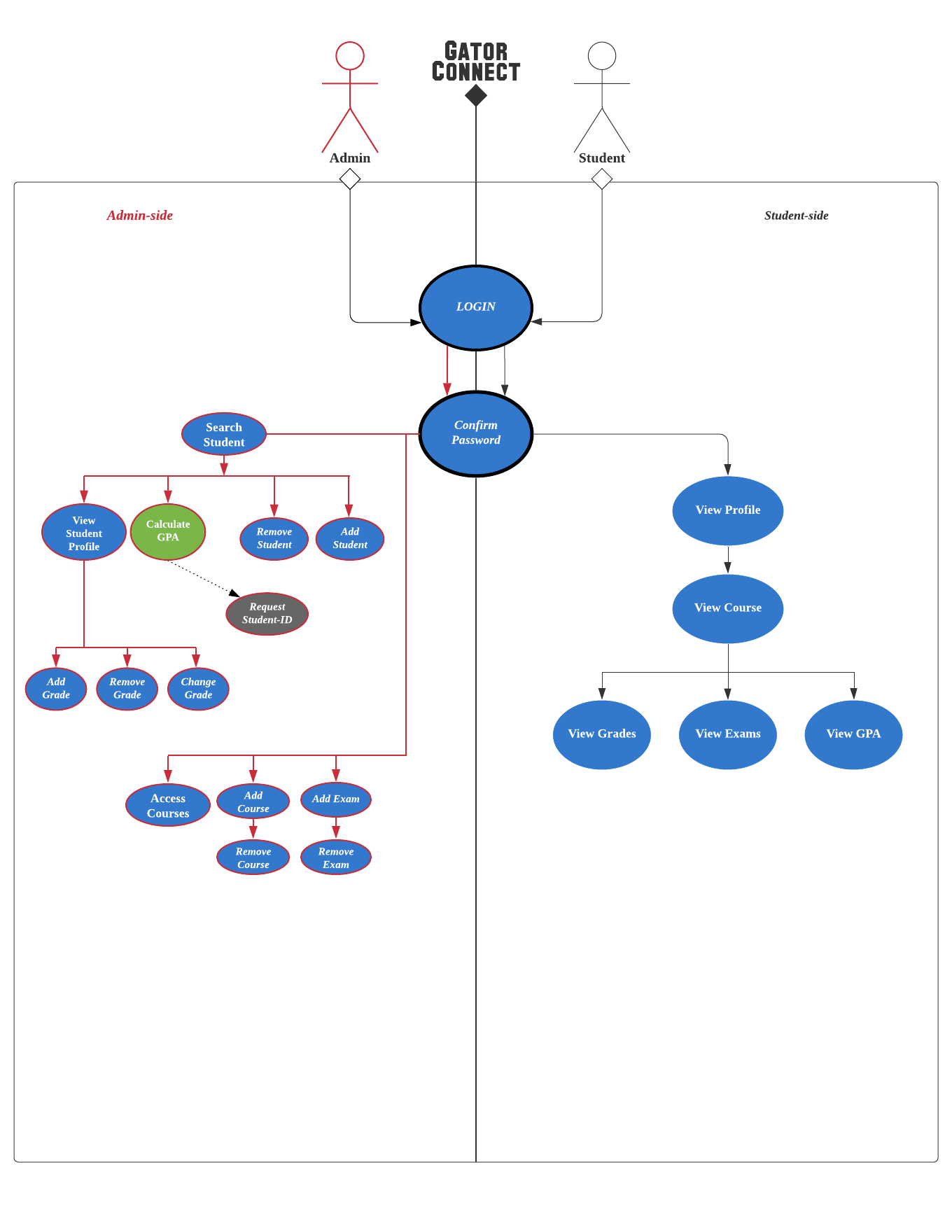
The features available to Student:

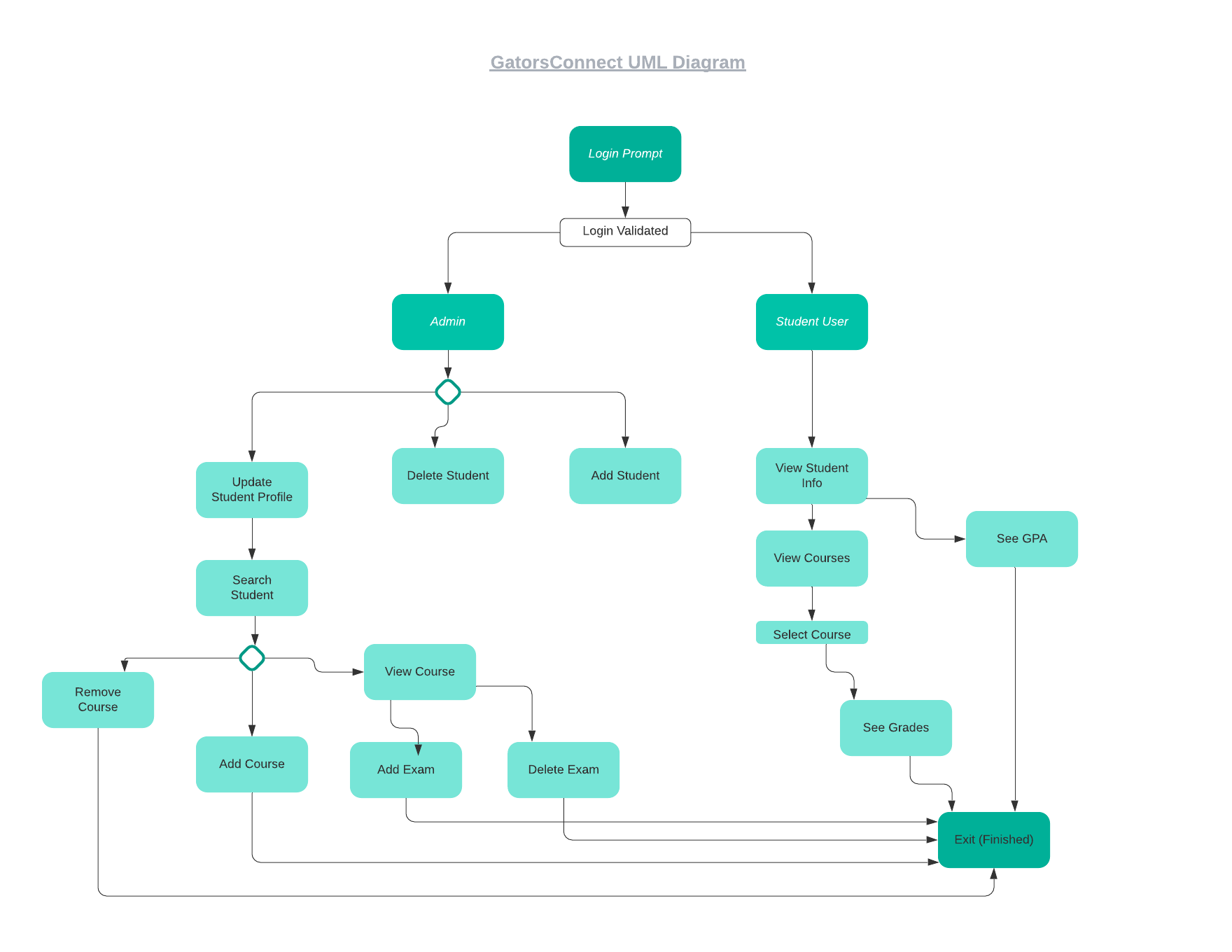
* View Class/Course list
* View Course List
* View detailed information for each course

All users can log in and log out. After each user logs in, the next page displays their list of available features as buttons

These available features are visually represented in the following UML Use Case diagram shown in the following photo

**4.2.2 Software Design Description (SDD)**

GatorsConnect will perform different functions depending on the type of user: student or staff. The staff consists of faculty who can function as admins. The administrators will have the control of the system, thus they will have administrative rights. Below are the UML Diagram and the UML Case Diagram



**4.2.3 Software Test Plan**

During this phase, the programmers coded and tested each code artifact separately, linked together all the code artifacts, and tested the product as a whole. Every programmer is personally responsible for making sure that his/her work is correct before committing it to GitHub. The available features outlined in the Analysis – User Characteristics section are used to create test cases:

* Add a student
* Add a staff
* Modify a student
* Modify a staff
* Delete a student
* Delete a staff
* Compute GPA

In addition to these direct tests, it is necessary to perform the following additional tests:

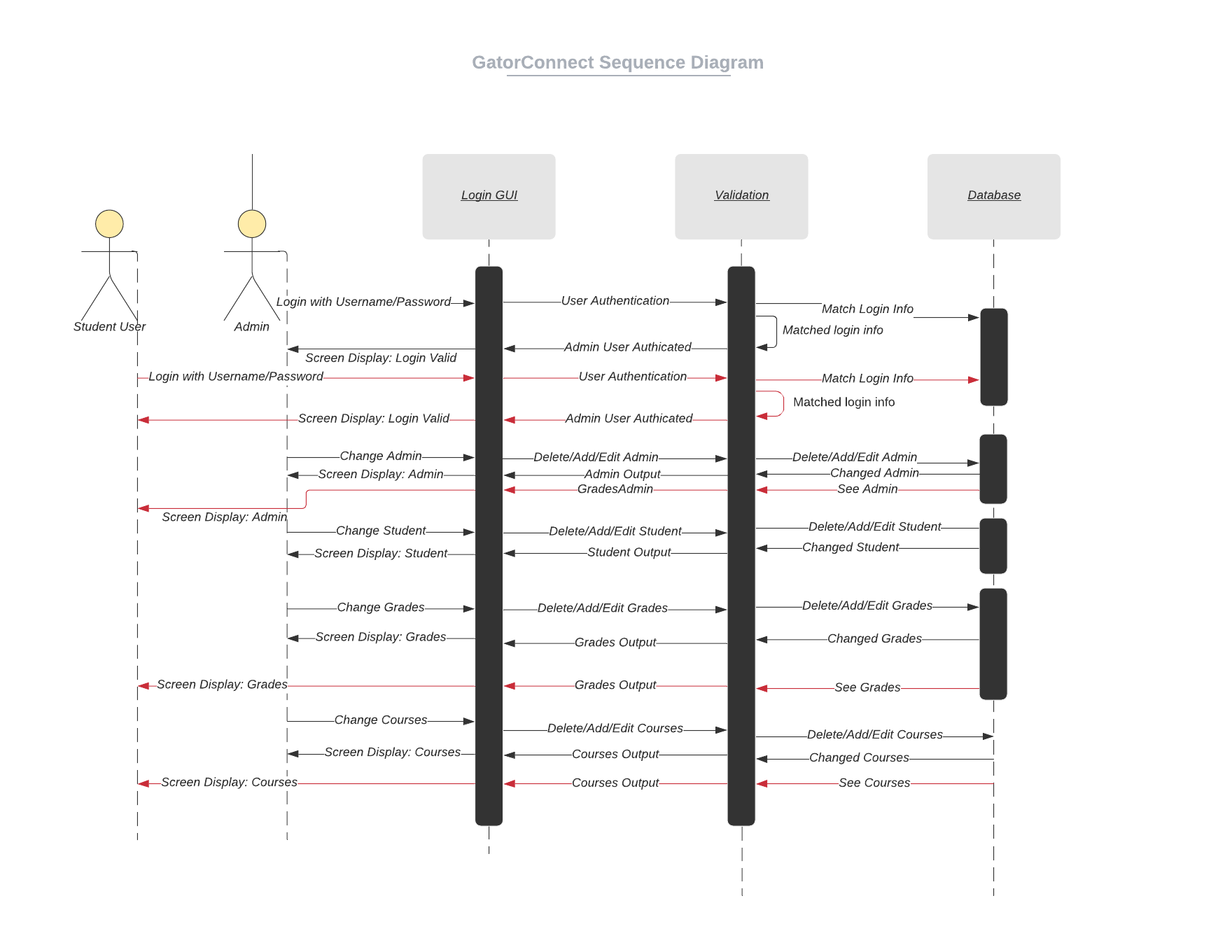
* Attempt to add a student that is already on file.
* Attempt to add a staff that is already on file.
* Attempt to delete a student that is not on file.
* Attempt to delete a staff that is not on file.
* Attempt to modify a student that is not on file.
* Attempt to modify an assignment that is not on file.
* Attempt to update each field of a staff twice and check that the second version is stored.

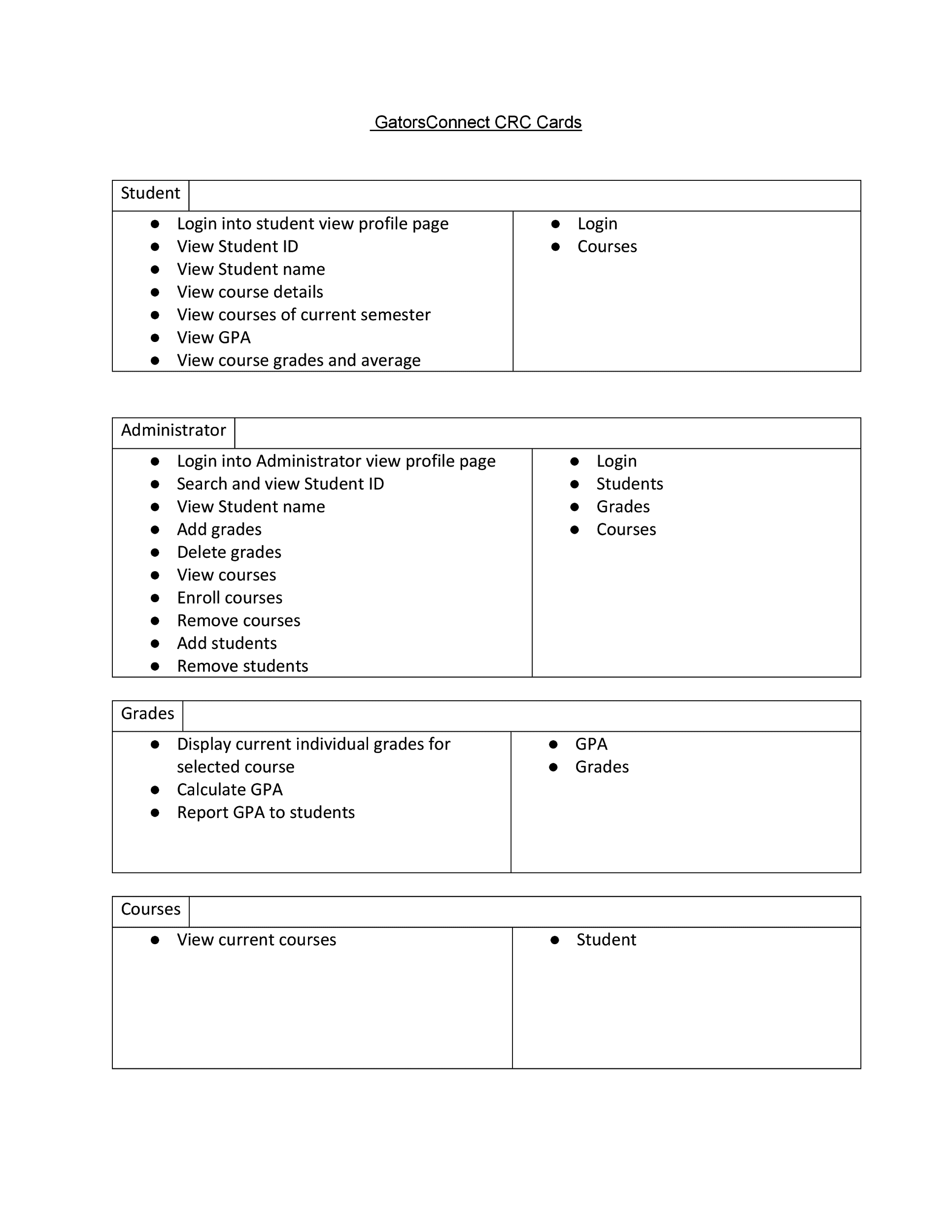
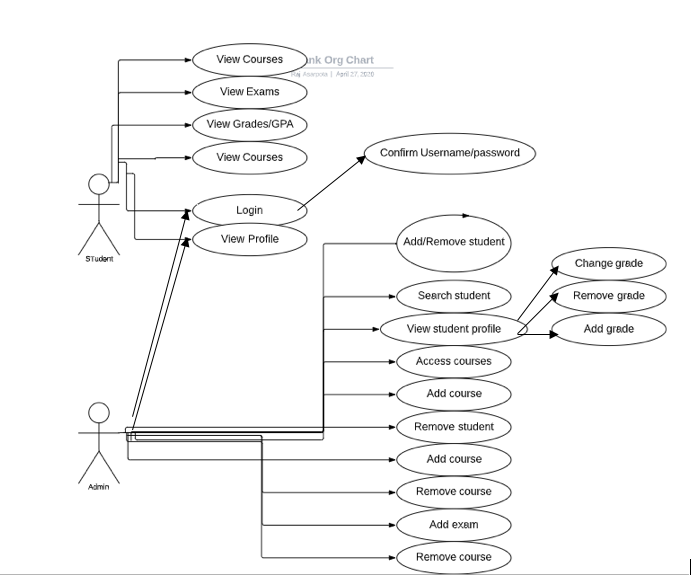
We ran through each test case, and every time we found something undesirable or something didn’t output correctly. For example, if the labels looked like name\_id, then we would return to the code and change it to Name ID. We reiterated these test cases until the software product met the project expectations.

**4.3 User Documentation**

The following diagrams were our plans for building the software. We came together and decided what was important for each phase of our software. Also explaining the algorithm itself, so the classes become easier to understand.

Please consult diagrams below

****

****

**5. Schedule**

Our team schedule did have to go through some changes throughout the semester because of the COVID-19 pandemic. Originally, we chose to meet once a week in class and once a week outside of class. The pandemic caused social distancing, thus in person meetings were no longer possible. We had to transition to in person meetings to video conferencing. We chose to do our meetings via Google Hangouts and Discord. We also found out that Github has a real time script editor, called Atom. We used Atom to make any changes to our source code together.

* Formation of Group (based on language preference) 02/17
  + Understanding Requirements
  + Establish team model and lifecycle model
  + Create Github repository
  + Establish timeline
* Create Diagrams 02/21
* Start SPMP 02/21
* Build the classes 02/24-03/06
* Decide between text files and database 03/06
* Build the database 03/08-03/30
* Connect functions to database 03/30-04/06
* Design outline of GUI 04/08
* Implement GUI 04/13
* Populate the database 04/15-04/16
* Final touches to GUI 04/20-21
* Final Testing 04/23-04/24
* Final touches Editing the SPMP 04/27-04/28
* Final team meeting 04/28
* Submission of project 05/03