Learning Management System Project Report

Our developer team for the Learning Management System consists of 4 members

1. Alan Saavedra
2. Edgar Santana
3. Mahendra Basyal
4. Sergio Toledo

**Introduction**

For being a part of a Software Engineering class our Professor, Professor Chang, has tasked us with creating a Learning Management System similar to BlackBoard or Moodle. The group of 4 students listed above got together and created the Learning Management System. We were given requirements however we could add features if we wanted to.

**Requirements**

Here is a list of requirements given to us by Professor Chang.

• Stores and retrieves student’s partial information (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).

• Allows administrator privileges (to insert update and monitor the whole process, search student ID, search CRN, view semester, edit grades and GPA)

• Students will be able to do GPA Calculation along with other class grade information

• Two login domains, student and administrator but student can only view his/her info cannot modify

**Analysis**

Although using java was our first choice because we weren’t familiar with java. Later the team decided to go with visual basic and C# programming languages to code the software and SQL for the database because we all were familiar with the functions and classes required to create the requirements listed above.

**System Development Life Cycle**

Amongst our democratic team we all voted to follow the Waterfall Life-Cycle Model also known as the “Classical Model” we chose it since it seemed the most straightforward as opposed to the other Life Cycle Models. We also liked the linear aspect of the Waterfall Model and also the baseline at the end of every phase makes it clear what we need to have done by then. Another reason we chose it is because it is document driven throughout the entire software development process also the model also works extremely well for small projects and each phase is completed and reviewed one at a time.

**Team Work**

Since we chose a democratic we split up subsystems of the project along with some other aspects amongst each other. Alan was in charge of creating the GUI and the use cases for it along with the project report. Edgar was in charge of creating the Visual Studio code to connect the GUI and Database together. Also he completed the PowerPoint Presentation for the class and some workflow phases like Requirements, Analysis, and Design. Sergio created the database using MySQL Workbench. Mahendra was mainly in charge of Documentation while everyone created diagrams for their respective parts.

**Version Control**

GitHub was used as the version control. Please visit  [https://github.com/edgars61/Learning-Managment-System](%20https://github.com/edgars61/Learning-Managment-System%20)  to see all the coding for GUI and Database and artifacts and descriptions that led to the finished software project.

**UML**

Figure : Architecture/Design Diagram

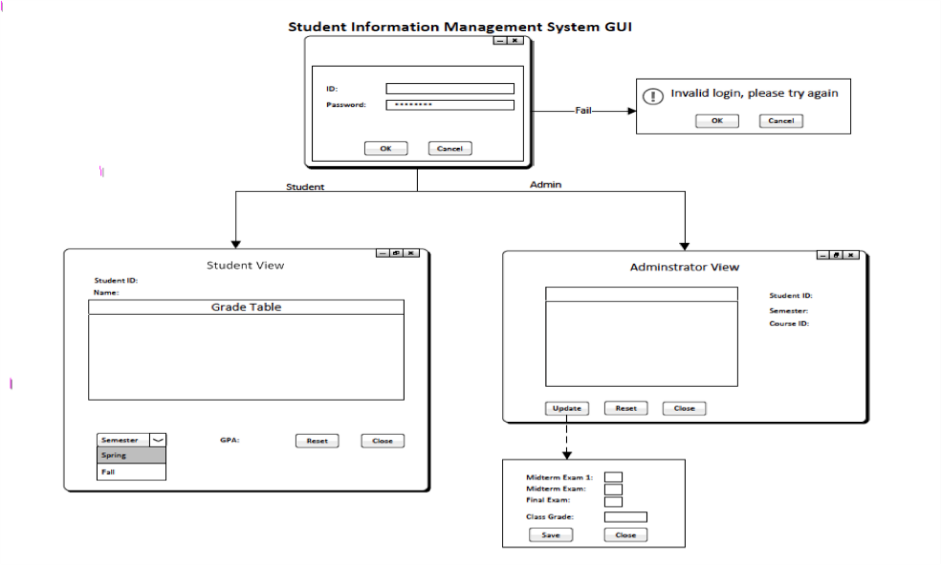


Figure : Data Flow Diagram

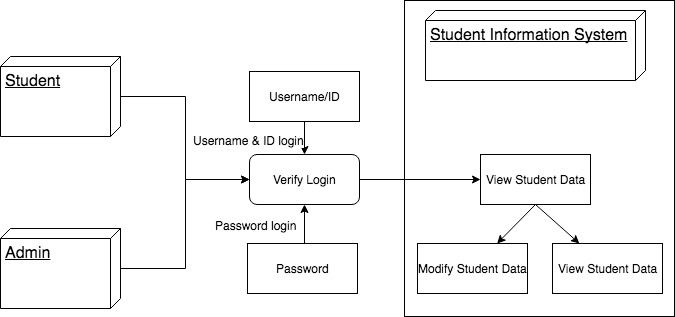
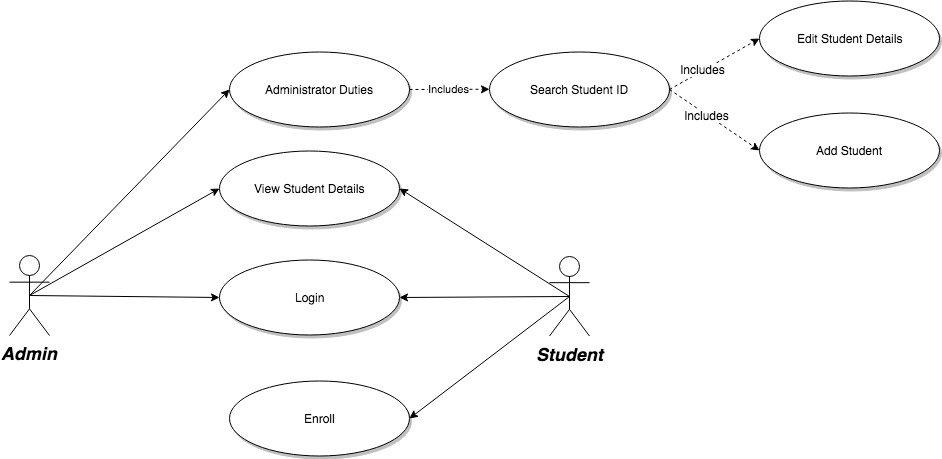


Figure : Use Case Diagram



Below are the user privileges that were given to us an requirements by Professor Chang.

**Student**

* Login using student ID and password
* View Student Details
  + Classes currently enrolled
  + Grades
  + GPA
  + View by semester

**Administrator**

* Login using admin ID and password
* View Student Details
* Classes currently enrolled
* Grades
* GPA
* View by semester
* Edit Student Details
* Classes currently enrolled
* Grades
* GPA
* View by semester
* Search Student Details
* Semester
* CRN Number
* Student ID

**Database**

Database is a gathering of information that is organized so that is can be effortlessly accessed, managed and updated. Database use for three types of transactions: Insert, Modify, and delete. The important step structure the information to be store in the database. A database use to reflect information by using entity-relationship model.

The database has been created using MySQL Workbench 5.7. In the database there are six tables:

1. Student personal information
2. Scheduling information
3. Course information
4. Grading information
5. Student GPA information
6. Administrator access information

Whenever user as student or administration access, MySQL verify their user access and password to allow in to software. MySQL and Visual Basic are connected to provide all information for user. If any changes are made through Visual Basic it will also update in the database.

**Team Organization**

Amongst each other we decided to form a Democratic team since we all felt that we are all on the same level when it comes to coding and we consider each other as egoless programmers. Another reason for this decision was that we consider that faults or “bugs” must be considered a normal and accepted event. Also, with a democratic we saw the opportunity to be able to assign subsystems of the project aspects amongst each other. For example, Alan would be in charge of GUI (Graphical User Interface) subsystem of the project. While Edgar and Sergio would be in charge of the source code and the database and lastly Mahendra would be in charge of the documentation aspect of the project. We all developed our diagram and artifacts for respective parts since we would have the most knowledge of our respective parts and would be the most qualified. However, since we were a democratic team we could help each other with our code or just something else regarding the project. For the specifics of the team work and contribution done by each member search above for the section named “Team Work”.

**Implementation**

So, since our software engineering team was a democratic team we all voted to follow the Waterfall Life-Cycle Model also known as the “Classical Model” we chose it since it seemed the most straightforward as opposed to the other Life Cycle Models. This section is dedicated to the Implementation phase of the Life Cycle Model which is when the project is coded and tested. What we did was to partition the project into subsystems then assigning them amongst each other. At first as a group we did decide on doing the project in Java along with the GUI being written with JavaFX since Java was language that we were all at least a little familiar with. However, about a week later we switched to the languages Visual Basics and C# for the GUI and the source code since we felt that this would be more efficient because we were all familiar with C++ and wouldn’t have to go through the trouble of learning more Java. We did the coding in the Visual Studio 2015. We know that documentation of any software engineering project is important, so we used meaningful variable names and meaningful functions just so that any outside person looking at our code can follow and understand our code we also wrote comments all throughout our code to improve readability We attempted to keep a high cohesion and low coupling when it came to our modules. When it came to the testing part of the Implementation phase we use the test to specifications method which is also is called black-box testing. The specifications or requirements given were that the software system should store 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 and that the system has two types of accessing modes, administrator and user. Student information management system is managed by an administrator. It is the job of the administrator to insert update and monitor the entire process. When a user logs in to the system. He/she would only view details of the student. He/she can't perform any changes. Then Acceptance Testing is done by either us the developer team and Professor Chang by performing the test to specifications once again Then Postdelivery maintenance takes place since no software system is perfect and all require maintenance.

**Testing**

For the testing phase we worked together to try to test each individual subsystem by using the testing to specifications method in which we ran our project against different test cases established in the Requirements phase of our workflow. We began running test cases on the login screen by attempting to log in with a mix of username and passwords that were not in our database to make sure that a user not in our database wouldn’t have access to the software. It was an initial success until we started using information contained in our database. We encountered a bug in which when sometimes we would create a new student, the database would not be able to find him/her therefore not allowing us to log in. The bug was documented. Our group re-tested the login screen to make sure we could log in when providing the correct credentials and were denied access when we provided the wrong credentials. Then we tested the login unit to make sure that if a user entered in credentials for a student then they were only shown the student view and if the user entered in administrator credentials they were only shown the administrator view. Once unit testing of the login screen was completed we moved on to testing the second part which was the Student View. We logged into several student accounts to verify that the correct student’s information was displaying correctly and pulling the correct grades as well as courses and other info in our database. We did not come across any bugs or errors when testing the Student View. Next, we began tests on the Administrator Unit to make sure all the necessary features were done properly. First, we started with making sure we could add information to our database through the administrator view. We were able to add information however the information was unnecessary changes in our database. The fault was recorded, and Sergio made changes to the SQL command being used in the code to make changes to the database to fix the issue. Afterwards we began adding students to our database then checking to verify they were added using the administrator view. With that feature completed our group moved on to implementing the next feature which would allow us to change a student’s information based on our use cases. If the administrator accidentally entered in the wrong grade they would be able to change it or if they need to add another class to the student’s profile they were able to. Lastly probably the hardest part of our system was making sure that whenever the administrator makes changes to the Student’s information the GPA would reflect that change no matter what it was. If the administrator had to update, delete, or add a grade to the student’s profile the GPA would reflect those changes. We did come across a problem that our GPA didn’t work properly however we were able to resolve it. We made several changes to the source code until they were able to get the GPA to work as intended based on the requirements. Initially the GPA was calculated within the database but that caused issues with our database trying to accurately calculate it. Eventually we resorted to code the GPA into our system and referencing the information in our database then adding the calculated GPA to our database.