Technical Design Document

Team Members:

Jonathan Martin, Alixa Bahena, Yuntian Zhang, Nhu Nguyen, Javier Rivera

TABLE OF CONTENTS

1 Purpose 3

1.1 Scope And Overview 3

1.2 Requirement Analysis 3

1.3 Definitions And Abbreviations 4

2 System Overview 5

2.1 System Characteristics 5

2.2 System Dependencies And Relational Models 5

3 System Architecture 7

3.1 Behavioral Diagrams 7

3.2 Strucural Diagrams 20

3.3 Graphical User Interface 22

4 Software Development process and tools 24

4.1 Development Life Cycle 24

4.2 Team Organization 24

4.3 Version Control 24

# Purpose

The purpose of this document is to provide information regarding the structure and implementation of our database. In this document we will highlight some of the major functionalities and specifications of our code in order to illustrate how we met the various requests put forth by the client. This document is to be presented to the client to show our methodologies used in the building of our database and to provide sufficient information on how the database is structured as well as to how it is meant to run when used by the client.

## Scope And overview

##### The objective of the finalized database is to provide the various types of users with information regarding their respective accounts as well as to facilitate actions for admins, faculty and students. The database will provide admins with the ability to perform actions such as:

##### Remove: Both a class or an account.

##### Add: Both a new class or an account.

##### View: Both a specific class or a user account.

##### For faculty four different actions will permitted once they are logged in:

##### Manage courses

##### View Courses

##### View Available Classes

##### Change Password

##### Log Out

##### For student account the database will give an overview of each student’s information such as their name, username, ID, Major, GPA, and other general information. The database will allow the students to perform 5 different actions such as:

##### View Schedule

##### Manage Schedule

##### View Available Classes

##### Change password

##### Log Out

## Requirement analysis

The client wants us to create a student information management system(database) to store details such as academic reports, classes available for them to take, the classes in which they are enrolled, and other details such as GPA for their semester. In addition to this the client has expressed an interest in having us include additional features if time allows. To develop a system which will meet the client’s demands we have listed a various IDE’s and cross-platform GUI creators. We will need to develop software which will make it easy for the client to perform their need daily task as well and we will also need to implement a structure for our system that will make it easy for maintenance soon. Having a good structure will also help in the addition of new features as well as make the database more user friendly.

## Definitions and abbreviations

##### The UML software approach was used in the building of this product. Below is a table highlighting a list of definitions used for this documentation.

|  |  |
| --- | --- |
| UML Diagram Types | Illustrates the different categories of the diagrams. |
| Class Diagram | Describes the structure of a system |
| Object Diagram | Highlights relationships between objects |
| Use Case Diagram | Illustrates relationships between use cases |
| State Machine Diagram | Describe the behavior of objects according to the state they are in |
| Sequence Diagram | Describe how the objects interact and order of those interactions |

# System Overview

## System Characteristics

##### The system uses a local database implementation which is pre-populated. The system is designed so that it may be easily maintainable in the future. In order to maintain the integrity of the user’s information all the data was normalized.

## System dependencies and relational models

##### Functional dependencies:

##### Users:           username → password, access\_type

##### Students:      username→ First\_Name, Last\_Name

##### Faculty:        username→ First\_Name, Last\_Name

##### Classes:        CRN→ Name, Instructor, Semester

##### Records:       row→ username, CRN, grade

-The tables are in the FIRST Normal Form because the intersection of each row and column contains one and only one value.

-The tables are in the SECOND Normal Form because every none-primary-key attribute is full functionally dependent on the primary key.

-The tables are in the THIRD Normal Form because no none-primary-key attribute is transitively dependent on the primary key.

-The tables are in Boyce-Codd normal form because they contain no nontrivial multivalued dependencies.

##### Relational Model:

Relations:

Users (username, password, access\_type)

Students (username, First\_Name, Last\_Name)

Faculty (username, First\_Name, LastName)

Classes (CRN, Name, Instructor, Semester)

Records (row, username, CRN, grade)

# System Architecture

## Behavioral diagrams

*Figure1: Use Case Diagram*

***The Use Case Diagram*** - The predefined users within the database are the “STUDENTS”, “FACULTY”, and “ADMIN.” Each of these user is given specific options in order to interact with the database.

|  |  |
| --- | --- |
| **Use Case 1** | View Available Classes |
| **Actors** | Student, Faculty |
| Input | none |
| Output | Class details |
|  | * Enter “username” and “password” * Click “Login” * Click “View Available Classes” * User is presented with Course number, Subject, Course Id, Course name, Semester, Day, Time, Instructor Id and Room number |

|  |  |
| --- | --- |
| **Use Case 2** | View Current Schedule |
| **Actors** | Student, Faculty |
| Input | none |
| Output | Current Schedule |
|  | * Enter “username” and “password” * Click “Login” * Click “View Current Schedule” * User is presented with his schedule details: Course number, Subject, Course Id, Course name, Semester, Day, Time, Instructor Id and Room number |

|  |  |
| --- | --- |
| **Use Case 3** | Enroll in a course |
| **Actors** | Student |
| Input | Course number |
| Output | Success/Failure message |
|  | * Enter “username” and “password” * Click “Login” * Click “Add/Remove Classes” * Enter course number * Click “Add class” * The course is added to his schedule |

|  |  |
| --- | --- |
| **Use Case 4** | Drop a course |
| **Actors** | Student |
| Input | Course number |
| Output | Success/Failure message |
|  | * Enter “username” and “password” * Click “Login” * Click “Add/Remove Classes” * Enter course number * Click “Remove class” * The course is dropped from to his schedule |

|  |  |
| --- | --- |
| **Use Case 5** | Change password |
| **Actors** | Student, Faculty, Admin |
| Input | password |
| Output | Success/Failure message |
|  | * Enter “username” and “password” * Click “Login” * Click “Change Password” * Enter “Current Password” * Enter “New Password” * Verify New Password * Click “submit” |

|  |  |
| --- | --- |
| **Use Case 6** | View Students |
| **Actors** | Faculty |
| Input | Course number |
| Output | List of students in specified CRN |
|  | * Enter “username” and “password” * Click “Login” * Click “Enter Class Grades” * Click “View Students” * Faculty is present with Student’s details |

|  |  |
| --- | --- |
| **Use Case 7** | Enter Class Grades |
| **Actors** | Faculty |
| Input | Grade |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Enter Class Grades” * Click “Edit Grade” * Enter student’s grade * Click “Ok” |

|  |  |
| --- | --- |
| **Use Case 8** | Update User’s username |
| **Actors** | Admin |
| Input | username |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Users Table” * Admin is presented with user’s username, password and access type * Enter new Username * Click “Save” * Username is updated |

|  |  |
| --- | --- |
| **Use Case 9** | Update User’s Password |
| **Actors** | Admin |
| Input | password |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Users Table” * Admin is presented with user’s username, password and access type * Enter new Password * Click “Save” * Password is updated |

|  |  |
| --- | --- |
| **Use Case 10** | Update User’s access type |
| **Actors** | Admin |
| Input | access type |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Users Table” * Admin is presented with user’s username, password and access type * Enter new access type * Click “save” * Access type is updated |

|  |  |
| --- | --- |
| **Use Case 11** | Add new user |
| **Actors** | Admin |
| Input | User details (user name/ password/ access type) |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Users Table” * Admin is presented with user’s username, password and access type * Click “Add Item” * Enter new username, password, and access type * Click “Save” * New user is created |

|  |  |
| --- | --- |
| **Use Case 12** | Delete User |
| **Actors** | Admin |
| Input | none |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Users Table” * Admin is presented with user’s username, password and access type * Select the an user * Click “Remove Item” * Click “Save” * A user is deleted from the database |

|  |  |
| --- | --- |
| **Use Case 13** | Update Faculty information |
| **Actors** | Admin |
| Input | Faculty details |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Faculty Table” * Admin is presented with Faculty’s First Name, Last Name, Username, and Department. * Enter new information: First name/ Last Name/User Name/ Department. * Click “Save” * Faculty’s information is updated |

|  |  |
| --- | --- |
| **Use Case 14** | Add new Faculty |
| **Actors** | Admin |
| Input | User details |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Faculty Table” * Admin is presented with Faculty’s First Name, Last Name, Username, and Department. * Click “Add Item” * Enter First Name, Last Name, Username, Department * Click “Save” * New Faculty is created |

|  |  |
| --- | --- |
| **Use Case 15** | Delete Faculty |
| **Actors** | Admin |
| Input | none |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Faculty Table” * Admin is presented with Faculty’s First Name, Last Name, Username, and Department * Select the a faculty * Click “Remove Item” * Click “Save” * A faculty is deleted from the database |

|  |  |
| --- | --- |
| **Use Case 16** | Update Students |
| **Actors** | Admin |
| Input | Student details |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Students Table” * Admin is presented with Student’s First Name, Last Name, Username, ID, DOB, enrolled Semester, home address, email address, phone number, and major * Enter new first name, last name, home address, DOB, enroll Semester, email address, phone number, major * Click “Save” * Student’s information is updated |

|  |  |
| --- | --- |
| **Use Case 17** | Add new Student |
| **Actors** | Admin |
| Input | Student details |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Student Table” * Admin is presented with Student’s First Name, Last Name, Username, ID, DOB, enrolled Semester, home address, email address, phone number, and major * Click “Add Item” * Enter Student’s First Name, Last Name, Username, ID, DOB, enrolled Semester, home address, email address, phone number, and major * Click “Save” * New Student is created |

|  |  |
| --- | --- |
| **Use Case 18** | Delete Student |
| **Actors** | Admin |
| Input | none |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Student Table” * Admin is presented with Student’s First Name, Last Name, Username, ID, DOB, enrolled Semester, home address, email address, phone number, and major * Select the a student * Click “Remove Item” * Click “Save” * Student is deleted from the database |

|  |  |
| --- | --- |
| **Use Case 19** | Update student’s grade |
| **Actors** | Admin |
| Input | grade |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Records Table” * Admin is presented with students’ username, course number, and grade * Enter new grade * Click “Save” * Student’s grade is updated |

|  |  |
| --- | --- |
| **Use Case 20** | Add new record |
| **Actors** | Admin |
| Input | Record details |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Records Table” * Admin is presented with students’ username, course number, and grade * Click “Add Item” * Enter student’s username, course number, grade * Click “Save” * Student record is added |

|  |  |
| --- | --- |
| **Use Case 21** | Delete student record |
| **Actors** | Admin |
| Input |  |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Records Table” * Admin is presented with students’ username, course number, and grade * Select the record * Click “Remove Item” * Click “Save” * Student’s record is deleted |

|  |  |
| --- | --- |
| **Use Case 22** | Update Class |
| **Actors** | Admin |
| Input | Class details |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Class Table” * Admin is presented with course number, subject, course Id, course name, Semester, Class Days, class time, instructor, and room number * Click on the class * Enter new course number/subject/ course Id/ course name/ Semester/ Class Days/ class time/ instructor/ room number * Click “Save” * Class is updated |

|  |  |
| --- | --- |
| **Use Case 23** | Add new class |
| **Actors** | Admin |
| Input | Class details |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Class Table” * Admin is presented with course number, subject, course Id, course name, Semester, Class Days, class time, instructor, and room number * Click “Add Item” * Enter new course number/subject/ course Id/ course name/ Semester/ Class Days/ class time/ instructor/ room number * Click “Save” * Class is added to the system |

|  |  |
| --- | --- |
| **Use Case 24** | Delete class |
| **Actors** | Admin |
| Input | none |
| Output | none |
|  | * Enter “username” and “password” * Click “Login” * Click “Edit Class Table” * Admin is presented with course number, subject, course Id, course name, Semester, Class Days, class time, instructor, and room number * Select the class * Click “Remove Item” * Click “Save” * Class is delete from the system |

***State Machine Diagram—***

This diagram is used to model the behavior of an object in a system during its lifetime. The diagram includes states (log-out, check log-in, user portal, and running state) that an object might be in, transitions from one state to another state, events that trigger transitions, and constraints on transitions. This project’s final state is also its initial state. We have one login form at the beginning. After the user performs all the actions and log out, the program is returned back to its login form.

##### 

: State

: Final state

: Transition

*Figure2: State Machine Diagram*

##### ***Sequence Diagram***

##### In order to describe the interaction between the objects over a specific period time, we use sequence diagram. We have five objects: user, admin, faculty, student, class, and record. For instance, after a user logs in as an admin, this admin object can interact with record, class, student or faculty object. The record object can also interact with admin by returning the record details that the admin has requested. One admin object is able to interact with more than one object, but it has to interact in sequence. In sequence diagram, a function is seen as a message sent from an object to an object. In particular, our objects only communicate through synchronous messages.

##### 

*Figure3: Sequence Diagram*

## Strucural diagrams

##### ***Class Diagram***

##### If the Use Case Diagram models the system functionality from a perspective outside of the system, the Class diagram is often used to refine use case diagrams. This diagram highlights our system’s classes, their respective attributes and methods, and their respective relationships among the data objects. We have six classes: admin, faculty, class, users, records, and student. The class name is in the top box of the diagram. The middle box contains the properties of a class, and the bottom box contains the functionalities of the class

##### 

*Figure4: Class Diagram*

Public (+): Accessible to all. Attributes should be protected.

Private (-): Visible to class methods

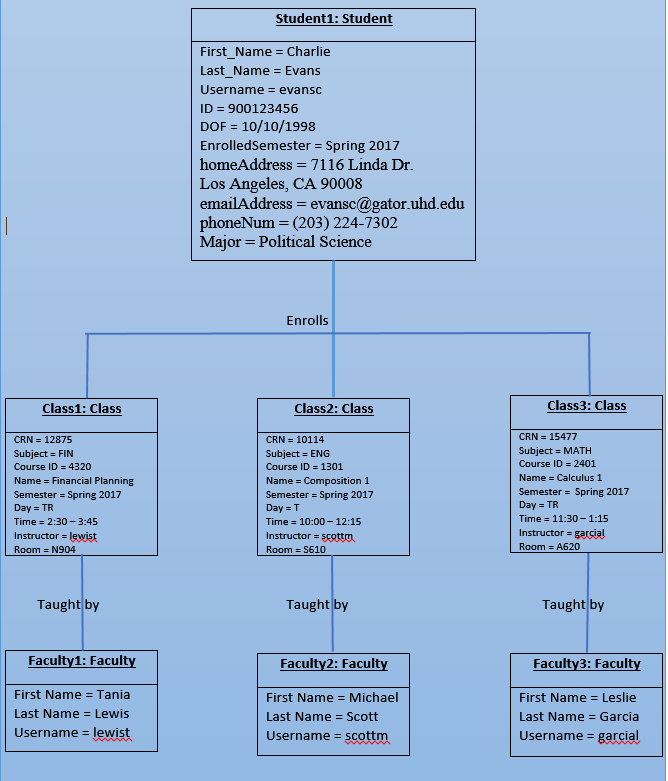
Multiplicity: Each student can have 1 to many class records.

Each record associate with one class.

One Faculty can teach one too many classes

***Object Diagram***

Like the Class diagram, Object diagram shows the relationship between the objects. However, the Object diagram shows how the system looks at a given time to explain complex relationships between objects. For instance, this Object diagram shows 1 Student object relates to 3 Class objects and 3 Faculty objects



*Figure5: Object Diagram*

## Graphical user interface

##### We had several options available when choosing a GUI to implement our database but after reviewing the pros and cons of each option the team decided it would be best to go with Qt. Qt is a cross-platform application framework which we used in the developing of our graphical interface for user interaction. Since our primary coding language was C++ we thought this software would help us best since it would require us to make little to no change when interacting with our codebase. In addition to this the Qt application also provided us with an extension we could install inside the IDE we were running. This helped the synching process and made it easier for us to modify the GUI.

##### Qt Download URL: <https://www.qt.io/download-open-source/>

##### 

# Software Development process and tools

## Development life cycle

##### We first began the software development process by listing out different development life cycles which we could follow to meet the client’s request and deadline as well as provide the best system possible. After much deliberation, our group decided to adopt the iteration and incrementation model. This model would allow us to generate working software parts quickly and early. It is also more flexible in the sense that it is less costly to change the requirements based on the client’s demands. Using this model provided us with easier risk management since each potentially risky piece was easier to identify and be resolved during its iteration.

## team organization

##### Our team consists of a total of 5 members. Since each of us have different programming skills and backgrounds we decided it would be best to follow a democratic team approach. Since the basic concept of this team approach relies in egoless programming we thought this method might better fit our team’s diverse skill set and allow for each of us to contribute better to the development of the product. One key aspect of this team model is the procedure taken during the decision-making process. As a democratic team, each member was able to voice his/her opinion regarding the decisions which would be taken in order to solve a problem or fulfill client demands.

## Version Control

##### To work on the development of this program simultaneously with one another we used the Github extension within Microsoft Visual Studio 2015. By doing this we could illustrate any changes to the program for our team members to view comments as well. Also, this allowed each member to work on their assign portion of the program while providing all members access to the other member’s assign sections.

##### Github URL: https://github.com/alixabahena/SIMS