Integral Solutions

University of Maine

COS 397 | Capstone I

System Design Document

Digital Program of Study Approval System

Client: Doctor Harlan Onsrud

Team Members:
Vincent King | Peter Riehl | Mac Creamer
Liam Blair | Aaron Wilde

Professor:
Terry Yoo

November 2021

Version 1

Digital Program of Study Approval System

Integral Solutions

Table of Contents

1. Introduction

The University of Maine's Computer Science Graduate Degree Programs currently rely on a physical application and approval process. The process, which involves Program of Study Documents (POS), is currently completed as follows:

A prospective student fills out a blank POS from the UMaine Graduate School POS webpage, saves the document to their computer, emails (or faxes) the document to the graduate school. The next step is for the graduate school to mark up the document with date received, rescan the marked up POS, and email it out to the Major Advisor, Graduate Coordinator, and eventually other committee members. As each member signs the POS they must rescan it and send it to the next person(s).

From an efficiency standpoint this is ripe with issues. In many cases, POS's get lost, forgotten, and the process of printing, reviewing, signing, scanning, and sending, is tedious when it should realistically take no longer than a few minutes.

Integral Solutions is designing and implementing a Digital Program of Study Approval System which will transform the outdated and problematic system into an efficient and convenient submission and approval system for prospective students and staff members.
This capstone project serves as a partial fulfilment of the Computer Science Bachelor of Science degree for the University of Maine. All parties involved with this capstone project benefit from its development and completion. Those parties are: ***Integral Solutions***, a capstone group of five University of Maine undergraduate seniors who are the facilitators of this project, our client Professor Harlan Onsrud, the School of Computing and Information Science (SCIS) graduate faculty mentors, our instructor Doctor Terry Yoo, and the Programs of Study department at the University of Maine.

1.1 Purpose of This Document

The document has a dual purpose. Firstly it serves as a contract between *Integral Solutions*, and our client, Professor Harlan Onsrud. Additionally, this document outlines the system design of our product, whose temporary name at the moment is Digital Program of Study Approval System (POSAS). The system design elements include data design, architectural design, interface design, and procedural design. Due to this project intending to be largely a "proof of concept" for the idea, the intended readership should be for both our client and for the high-ranking academic staff at the University of Maine, with the former being prioritized to ensure absolute clarity for agility's sake.

1.2. References

- “UMaine Graduate Student Program of Study Creation and Approval System” Proposal
    - *Author: Doctor Harlan Onsrud*
    - *Date: September 2021*
- “UMaine Graduate Student Program of Study Creation and Approval System” SRS
    - *Author: Integral Solutions*
    - *Date: October 2021*

1.3. Purpose of the Product

The original proposal for this document outlined the need for a digital system to be created due to the large number of physical documents which are housed for the various graduate departments at the University of Maine. Specifically, the POS that graduate students use to dictate their graduate careers are cumbersome in both maintaining and updating. Professor Onsrud outlined the idea that POS’s will sometimes get lost and that a number of signatures from advisors are required with every minute change that occurs with a given POS. While physical documents may be fine for a smaller number of students, but at larger scales, as seen with UMaine SCIS, it becomes inefficient and problematic warenting a new solution is needed.

1.4. Product Scope

Below we give a visual representation of the project's scope (Figure 1.4.1 and Figure 1.4.2). It details a limited number of actors and does not include moderators or administrators of the project. In another document, an updated variation of this Unified Modeling Language (UML) model will give information about such actors. The baseline of the project itself though will only include a high level view with actors that will experience this project on a “day-to-day” basis. Said “day-to-day” basis modeled below includes the process of creating an account, a student’s POS experience, and a student’s advisory committee experience interacting with their advisee's Draft POS (DPOS).

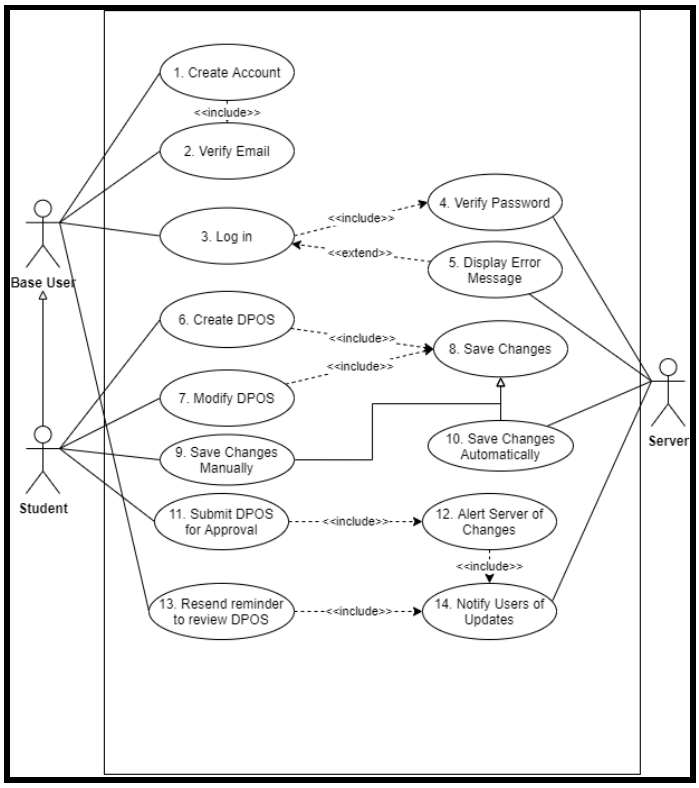

Figure 1.4.1. *Account creation and Student Draft Program of Study functionalities. This figure showcases the process of account management that all users have access to (Use Cases 1 - 5) and functions a student has access to regarding their DPOS (Use Cases 6, 7, 8, 9, 11). Use Case 10, 11, 12, and 14 are server/system related Use Cases, and 13 is a general use case for all users.*

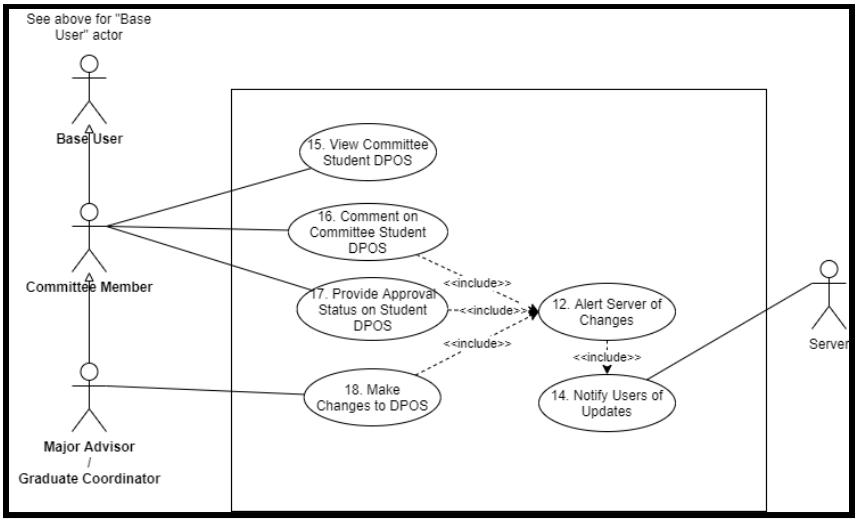

Figure 1.4.2. *Advisory Committee functionalities for both general committee members and the advisor/graduate coordinator. Showcased is the basic functions relevant to all committee members and the graduate coordinator for viewing, commenting, and approving a student DPOS (Use Cases 15, 16, 17). Use case 18 is an administrative function that only the major advisor and the graduate coordinator have access to. See above for Use Case 12 and 14.*

2. System Architecture

The following sections outline the architectural design of our system and also gives a decomposition description in the form of a diagram. Architectural design helps provide guidance for creating the framework of the system. It outlines frontend and backend attributes as well as operations. The also outlines the performance of the Application Programming Interface and how it interacts with different aspects of the system. The Decomposition Description pictured in figure 2.2.1 lays out the use case diagrams in an alternative fashion. This helps the design team visualize use case to use case interactions.

2.1 Architectural Design

Logical Architecture Diagram (Figure 2.1)

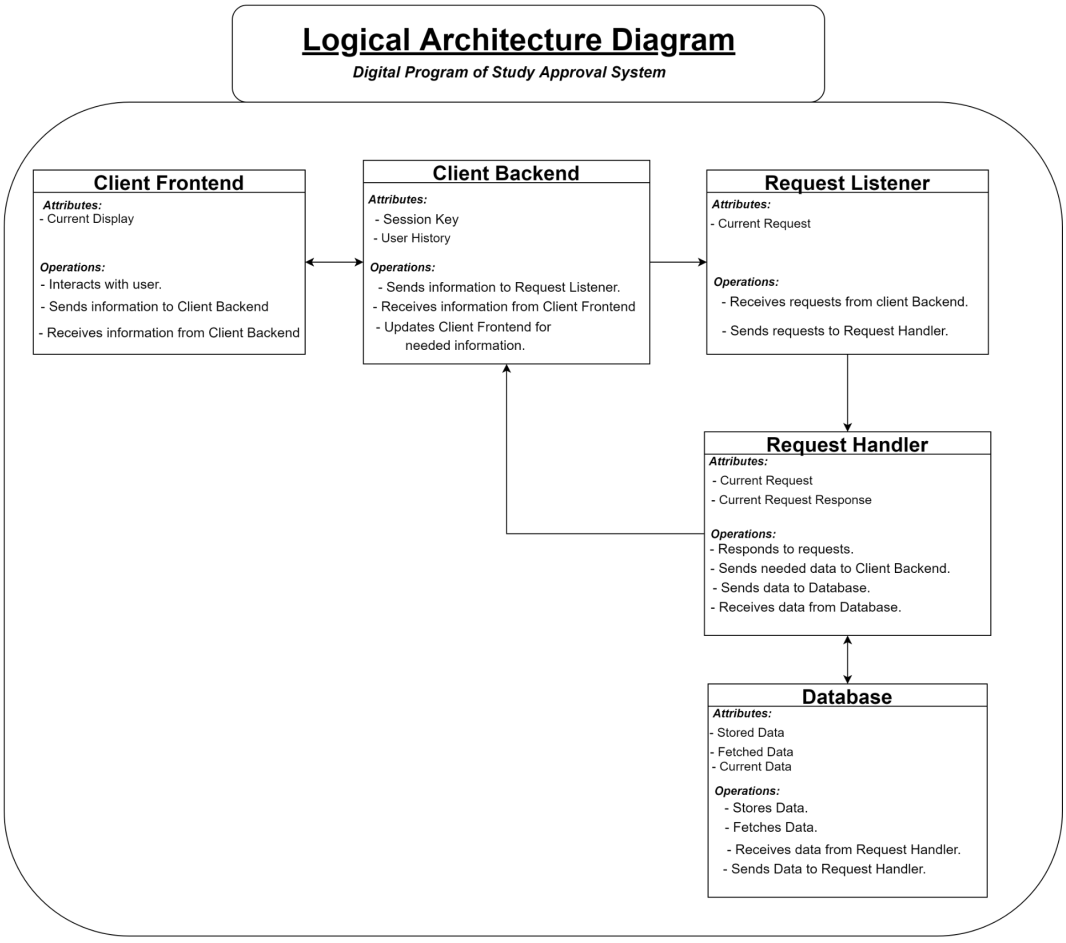

Figure 2.1 *Logical architecture design diagram showing the relationship and interactions between objects of the system, as well as the attributes and operations associated with each object.*

Technology Architecture Diagram (Server-Client Model)

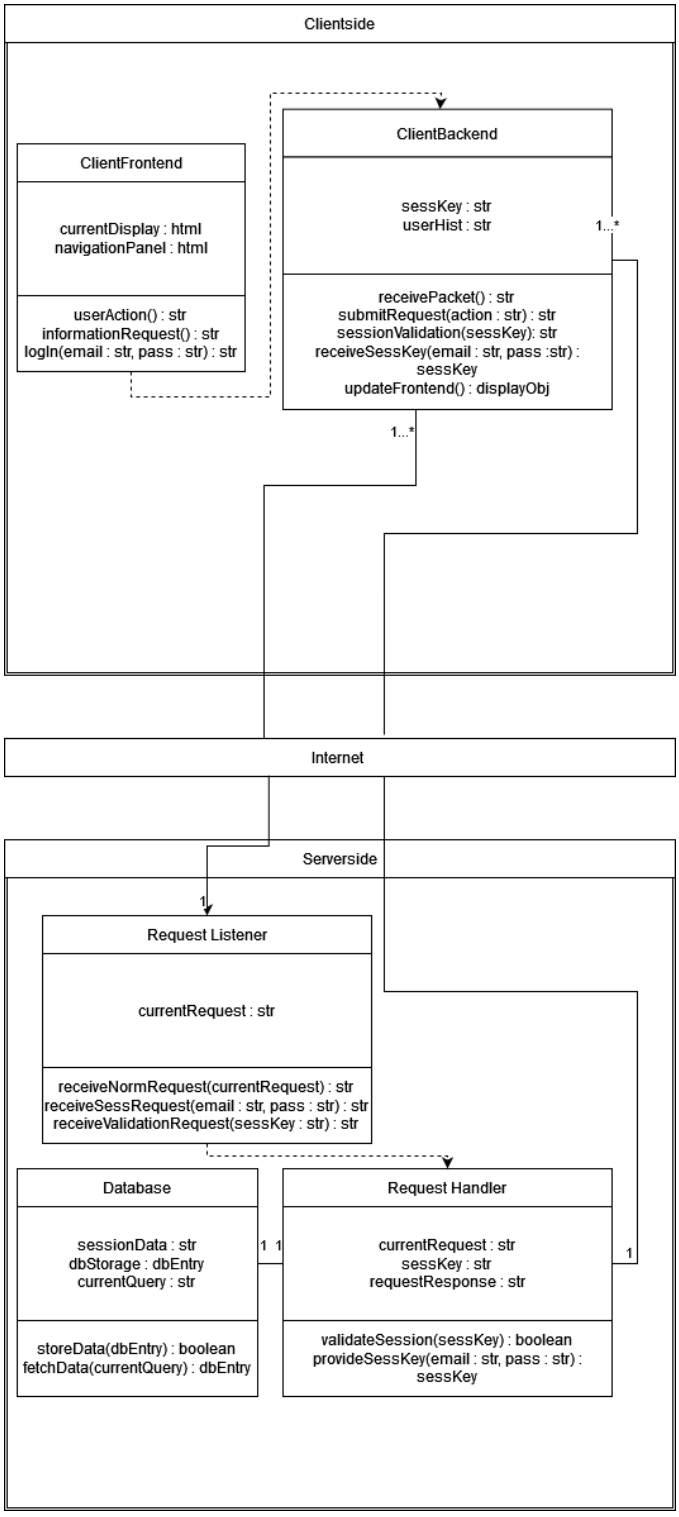

Figure 2.1.2. *A depiction of the client-server model that our system uses. There is a one-to-many relationship from the server side to the client side of our system. Intuitively, this is due to the fact that a number of clients are going to be sending requests to our one server system. Functions that each class will use are included in their respective Unified Modelling Language (UML) classes.*

Hardware Components

The Digital Program of Study Approval System's database will be located on the University of Maine campus as a physical desktop.

Software Components

Frontend and backend development will be made and maintained using the Django programming language as well as PostgreSQL. Software Objects for this system include Client Frontend, Client Backend, Request Listener, Request Handler, and the Database. These objects operate and communicate through software and send information from client side to server side through the internet.

System Architecture Overview

The architecture of our Digital Program of Study Approval System will consist of five system objects. These objects are: **Client Frontend**, **Client Backend**, **Request Listener**, **Request Handler**, and **Database.** These five system objects will interact with one another through three separate components which send information and interactions between one another. These components are: **Client-side**, **Internet**, and **Server-side**.

The client-side component will manage the interactions initiated by the user, through the Client Frontend and Client Backend system objects. Client Frontend interacts with the user and sends needed information to the Client Backend object, as well as receives data to display from the Client Backend. The Client Backend object receives information from the Client Frontend, updates the Client Frontend, and sends information to the Request Listener object located on the Server-side component through the internet component. On the Server-side component, the Request Listener receives requests from Client Backend, and sends valid requests to the Request Handler to be handled. The Request Handler takes in and responds to requests sent from the Request Listener, as well as send and receive data from the Database object. Data taken from the Database object is then sent from the Request Handler to the Client Backend to be displayed back to the user on the Client Frontend. The final object in our system is the Database, which stores data and fetches data as well as receives data from the Request Handler. The Database fetches the needed data specified by the Request Handler, and sends the needed data back to the Request Handler. This completes the loop of interactions between our five system objects, as the Request Handler from here sends the needed data back to the Client Backend to be displayed to the user on the Client Frontend.

2.2 Decomposition Description

While our system does use classes, said classes have been shown above in the client-server model (Figure 2.1.2). This client-server model is doubling as an implementation diagram for this exact purpose, that is, said diagram contains the exact information present in an implementation diagram. Because of this, we choose to leave off such a diagram to avoid redundancy. Pictured below (Figure 2.2.1) is our decomposition diagram for our system functionalities, expressed in a hierarchical fashion with respect to the roles that the system and the users have with one another.

Figure 2.2.1 Decomposition Diagram of the client side functions that the system makes use of. It is further broken down into user actions by subcategory, and actions that the system itself has.

As is laid out, Figure 2.2.1 shows the functional decomposition of our system relative to our users. There are some interactions that the system makes with regards to the client, but largely this section focuses on the actions that the various user types make within their respective roles. Specifically, the maintenance and modification that a user has access to from the client side perspective. Also included is how the system handles new users creating accounts, and how all accounts are managed by their respective general user.

3. Persistent Data Design

This section holds 3.1 Database Descriptions. 3.1 Database descriptions, displays a visual of an Entity-Relationship (ER) model which shows the relationship between our entity sets and their attributes.

Based on our project's design we removed the recommended section 3.2, File Descriptions. Our system design does include the creation of PDF files but is not the primary storage of information in our database. Our database holds integer, boolean, and string variables associated with a user and their POS. The PDF component is supposed to provide an output space for the users associated information, it is not a primary storage source.

3.1 Database Descriptions

Below, figure 3.1.1, is an ER model which shows the relationship between our entity sets and their attributes. The ER model illustrates that all users, Graduate Coordinators, Committee Members, Major Advisors, and Students all have basic user attributes. Students have extended information, this includes more in depth knowledge of the individual that is essential for the graduate school to know. From here, depending on the degree key, the user will be branched out to the respective POS: Masters, Graduate Certificate, Phd. Courses in the final necessary entity set, this is connected to the POS entity sets by a one or many relationship, it holds all the tuples which make up a course. There is also one other entity set that, depending on if the default NULL value of the attribute is overwritten, can be realized. This entity set is comments.

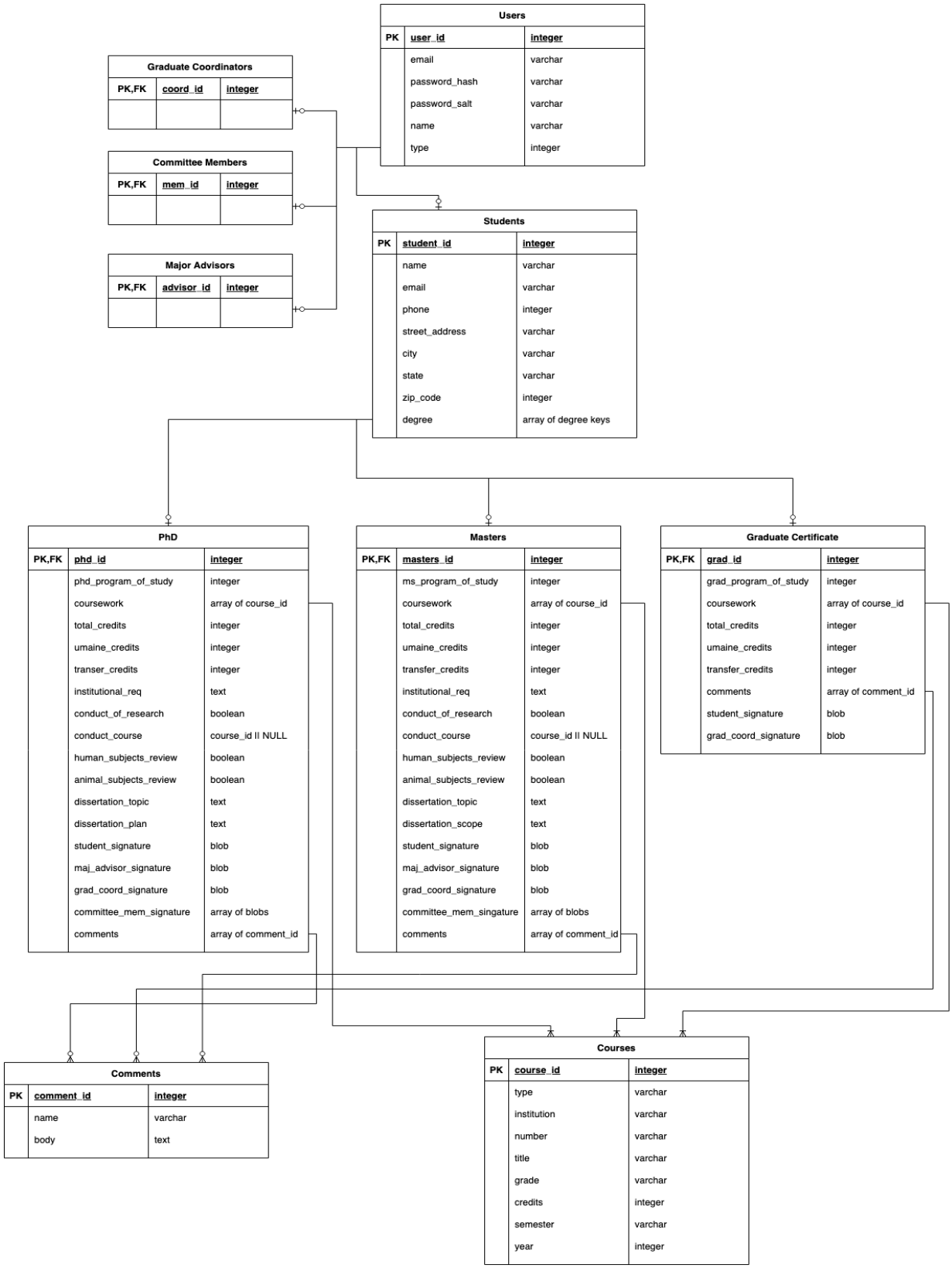

Figure 3.1.1 *ER Model which illustrates the relationship between our entity sets & their related attributes*

4. Requirements Matrix

The Requirements Matrix displayed below contains the various requirements, components, and test cases that must be satisfied in order for each respective use case to come to fruition. Basically, this formalizes the process of requirements telling us “What to do” and “When we’re done”, our system components and requirements, as well as our test cases, respectively. This section gives us a clear direction to go in, in order to fully finish designing our system and its requirements.

Requirements Matrix				
Use Cases		System Components (Functions/Models) To Satisfy Use Case	Requirements	Test Case
ID #	Full Name			
1	Create Account	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.2.2	SRS - 2.2.1
2	Verify Email	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2	SRS - 2.1.2.6 SRS - 2.1.2.7	SRS - 2.2.2
3	Log In	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.2.1	SRS - 2.2.3
4	Verify Password	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.2.1	SRS - 2.2.4
5	Display Error Message	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.2.8 SRS - 2.1.2.9 2.1.2.10 2.1.2.11	SRS - 2.2.5
6	Create DPOS	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1 SDD - Figure 3.1.1	SRS - 2.1.3.1 SRS - 2.1.3.2 SRS - 2.1.3.3 SRS - 2.1.3.4 SRS - 2.1.3.5 SRS - 2.1.3.7 SRS - 2.1.3.33 SRS - 2.1.4.2	SRS - 2.2.6
7	Modify DPOS	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.3.11 SRS - 2.1.3.13	SRS - 2.2.7

		SDD - Figure 3.1.1		
8	Save Changes	SDD - Figure 1.4.1 SDD - Figure 2.1.2	SRS - 2.1.3.9 SRS - 2.1.3.10	SRS - 2.2.8
9	Save Changes Manually	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.3.10	SRS - 2.2.9
10	Save Changes Automatically	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.3.9	SRS - 2.2.10
11	Submit DPOS for Approval	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.3.8 SRS - 2.1.3.12	SRS - 2.2.11
12	Alert Server of Changes	SDD - Figure 1.4.2 SDD - Figure 2.1 SDD - Figure 2.1.2		SRS - 2.2.12
13	Resend Reminder to Review DPOS	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.3.14 SRS - 2.1.5.1	SRS - 2.2.13
14	Notify Users of Updates	SDD - Figure 1.4.2 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.3.14 SRS - 2.1.3.24	SRS - 2.2.14
15	View Committee Student DPOS	SDD - Figure 1.4.2 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1 SDD - Figure 3.1.1		SRS - 2.2.16
16	Comment on Committee Student DPOS	SDD - Figure 1.4.2 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1 SDD - Figure 3.1.1	SRS - 2.1.3.17	SRS - 2.2.17
17	Provide Approval Status on Student DPOS	SDD - Figure 1.4.2 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1 SDD - Figure 3.1.1	SRS - 2.1.3.18 SRS - 2.1.3.19 SRS - 2.1.3.20 SRS - 2.1.3.21 SRS - 2.1.3.22 SRS - 2.1.3.23 SRS - 2.1.3.27 SRS - 2.1.3.28	SRS - 2.2.18

			SRS - 2.1.3.29	
18	Make Changes to DPOS	SDD - Figure 1.4.1 SDD - Figure 2.1 SDD - Figure 2.1.2 SDD - Figure 2.2.1	SRS - 2.1.3.25 SRS - 2.1.3.26 SRS - 2.1.3.33	2.2.19

Appendix A – Agreement Between Customer and Contractor

By signing this document, the customer and development team agree to the system design laid out above. Both parties will also agree to the data design, interface design, procedural design, and architectural design laid out in this document.

Both parties will also agree to the defined function and scope of the project, as well as to all functional and non-functional requirements that the project must meet. Both parties will additionally agree to the contents of each promised deliverable stated above. The development team agrees to provide a software system that meets said requirements at a later date.

In the case of changes to the document, the customer will be informed of the changes via email. These changes would have to be approved by the customer before they are made. Meetings may be scheduled in order to discuss any proposed changes to the document. By signing this document, both parties agree to use said procedure in the event of changes to the document.

By signing below, the customer and development team agree to the above. Additionally, the customer may write any comments or concerns they may have in the space below.

Customer Comments:

Customer Signature: _____

Development Team Signatures:

_____ _____

_____ _____

_____

Appendix B – Team Review Sign-off

By signing below, both parties confirm that they have reviewed the contents of this document. Additionally, both parties will confirm that they have agreed on the document's content and format.

Team Member Comments:

1. _____
   _____
   _____
2. _____
   _____
   _____
3. _____
   _____
   _____
4. _____
   _____
   _____
5. _____
   _____
   _____

Customer Name: _____ Customer Signature: _____

Date of Signature: _____

Team Names:	Team Signatures:	Date of Signatures:
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Appendix C – Document Contributions

Liam Blair - 20%
- Section 3 Diagram

Mac Creamer - 20%
- Section 2.1.2 diagram
- Section 2.2.1 diagram
- Section 2.2

Vincent King - 20%
- Section 2.1 diagram
- Section 2.1

Peter Riehl - 20%
- Section 3
- Introduction
- Introductions to all major sections

Aaron Wilde - 20%
- Section 4

All
- Section 1