Software Design Document

Grace Hsieh, Brett Lewerke, and Chayson Spigarelli

29 September 2022

Project Sponsor: Terry E Baxter

Faculty Mentor: Dr. Michael Leverington

Mentor: Daniel Kramer

Version: 2.0

Table of Contents

1.0 Introduction

Schools are always searching for a learning management system (LMS) to help better the education for their students. A LMS allows students to work on their studies anytime while they have access to the internet. This is essential for any college as the internet becomes more popular and continues to provide additional means for learning all the time. There are many LMS's out there but none that make students feel like they are playing a game. That's where our team steps in! The goal of the team is to develop and design a "gamified" LMS that makes students feel like they are playing a game when in fact they are doing school work on our "gamified" learning management system.

In order to gamify our LMS the team will build some additional features that a typical LMS does not offer. For example, one of the team's goals is to create and manage a student's currency (coins) inside the LMS. The students will have the ability to spend their coins on a variety of different items in order to help them achieve success in their course(s). Students will also have the ability to select a pre-made avatar so that other students can admire and appreciate their custom look!

This project is split between three parts: the website application, desktop application, and the database. The website application is where students will complete their course. The desktop application will be where the teacher creates a course. Lastly, the database will connect the two so that data can be shared between the desktop and website applications.

This project idea comes from our sponsor Professor Terry E. Baxter, a Civil and Environmental Engineer at Northern Arizona University (NAU). NAU currently uses an LMS called the Blackboard Learning Management System which a number of professors at the university generally dislike for several reasons. In our sponsor's case, he dislikes it due to a lack of creative freedom when creating and running a course. For example, our sponsor wanted to be able to build an asynchronous class with game-like features such as a marketplace to spend coins earned by doing well on assignments. These coins could be used to gain new attempts on quizzes or unlock additional content that can be referenced during exams. The Blackboard Learning Management System does not support this kind of functionality, so Professor Baxter is using multiple spreadsheets to track coins and is having students send emails with a receipt of their transactions. Each of these receipts need to be manually entered into the spreadsheet which becomes increasingly unsustainable with increasing class sizes. The team hopes to remove his need for manual labor in order to teach the kind of class he wants.

By the end of this project the team would like to have a fully functional gamified learning management system that will last for many years to come. One of the team's goals is to allow our

sponsor to use this learning management system for any course of his choosing while also allowing other teachers to use it for their classes.

2.0 Implementation Overview

2.1 Big Picture

The project will be divided into three main parts: a desktop application, a website application, and the database. The desktop application is being created using Unity Game Engine. Inside the desktop application is where a teacher will be able to create, manage, and publish a course. The website application is being developed using .NET Framework's Razor Pages, a platform built to integrate C# functionality into HTML and CSS. The website application is where students will be able to interact and participate in their courses. A student will only be able to take a course on the website application after it has been created by a teacher on the desktop application. Lastly, the team is using a relational database (RDS) to store information created by both the students and the teachers. This will serve as an intermediary between the website and desktop applications allowing them to share and manipulate the same data.

The goal of Team Gaming Ed. is to have a fully functional gamified learning management system that makes students feel like they are playing a game. If students feel like they are playing a game students might be more interested in completing their school work.

2.2 Technologies

The team has researched many technologies since the start of last semester. The main technologies the team will be using include Unity, .NET Framework, and MySQL.

- **Unity:** In order to make a gamified LMS, the team chose to include a game engine for one of its core technologies. The team is using Unity, a cross-platform game engine developed by Unity Technologies, for the desktop application. Unity is easy for the team to learn due to being object oriented with a user interface that allows the programmer to create objects using a drag and drop interface. These objects can have scripts which tell the objects what actions to perform during a certain event. Furthermore, Unity allows the desktop application to be downloaded onto a computer and be able to be worked on in an offline mode that can later be updated once an internet connection has been established which was requested by the sponsor.

- **.NET Framework:** The team has chosen to use .NET Framework for the website application because it allows C# code to be written in the same file as HTML code. This is done by using the framework's Razor Pages which is a programming model developed

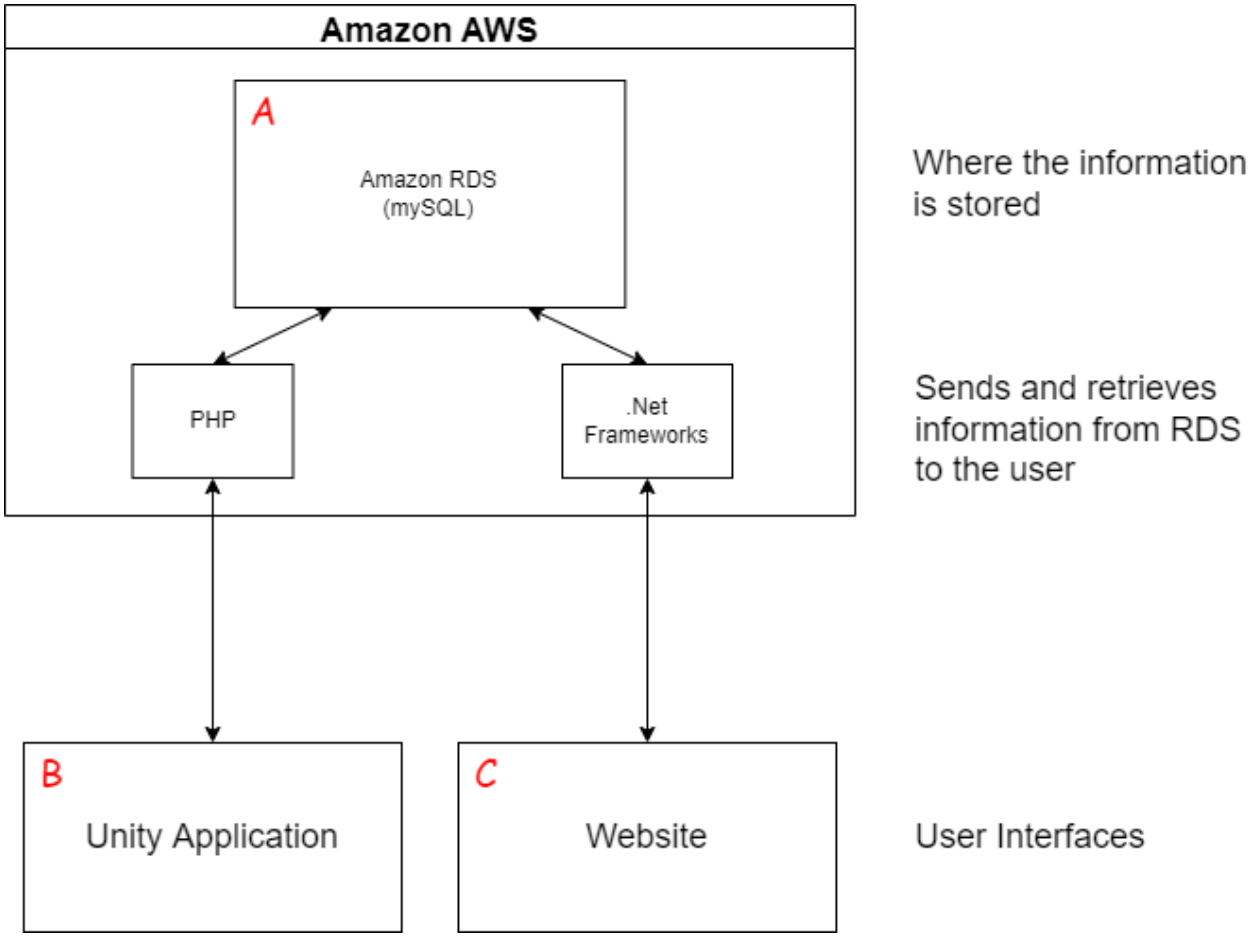
by Microsoft. .NET Frameworks also includes a large number of libraries and documentation. This allows for the team to choose between a large number of options in order to quickly find a desirable solution to any given problem posed by the project.

- **MySQL:** The team has decided to move forward with MySQL for the RDS as it is very popular and there is a large amount of documentation on the internet if needed by the team. Both the desktop application and the website application are able to access the database at the same time without compromising data integrity. This also means that the team can use the same set of data while working on different sections of the project without causing problems for the other members.

3.0 Architectural Overview

To reiterate, there are three major components of our project, the relational database (RDS) using MySQL query language [A], the Unity based desktop application [B], and the .NET Framework website [C]. Figure 3.0 shows how they work together as a whole. This section will go into the primary functions of the three major components, how information travels between them, and the architectural influences behind development decisions.

Figure 3.0 Architecture Diagram



3.1 Desktop Application

- **Primary Functions:** The desktop application is made for teacher access to sections of the database. The primary functions can be put into two categories: teacher course controls and super administrator management tools. The teacher will have full control over their assigned courses which includes adding, modifying, or deleting content in the

course and adjusting student progress or grades. The super administrator is the team's current solution to managing teacher accounts. It is the account with the most power made primarily to delete and create teacher accounts as well as any courses associated with them.

- **Information Pipeline:** A series of files written in PHP have been made to allow for SQL queries to pass to the RDS. These statements will be able to request and modify relevant information from the RDS.

- **Architectural Influence:** One architectural influence style the team used for the desktop application is the component-based architectural pattern. This emphasizes the separation of concerns with respect to the wide-ranging functionality available throughout a given software system. This also allows the team to work on different components at the same time where they can be later linked and work together.

3.2 Website

- **Primary Functions:** The LMS itself will be accessed through the website. Students will be able to log into the website and access courses and the course content that they are enrolled in. Furthermore, teachers will be able to use their own login credentials to interact with their own courses as if they were a student.

- **Information Controls:** The website is made with .NET Frameworks' Razor Pages. Razor Pages is a programming model that enables the team to create the website application using C# and HTML in the same file. SQL statements can be passed through as strings to the RDS which will in turn interact with the RDS.

- **Architectural Influence:** The course content is primarily in a tree structure. In other words, the idea is that a parent node will connect to some number of child nodes. Those child nodes will have their own child nodes and those nodes will continue to repeat this pattern for many levels. This creates a contained structure that is easily compartmentalized and easy to recreate. Another architectural influence would be the model-view-controller architectural pattern. The model-view-controller is commonly used to divide a program into three parts: model, view, and controller. This is done to separate information that is presented and accepted to and from the user which promotes security and modularity.

3.3 Database

- **Primary Functions:** To reiterate, the RDS is where the project stores all of its long-term data, such as profile information, courses, grades, etc. All of it needs to be able to relate with each other which is why the team chose to use a relational database instead of any other model. An RDS uses a system of primary and foreign keys when grabbing associated information from multiple data tables.

- **Information Controls:** The data itself is stored in tables with rows and columns. Each row is one object while each column is a data variable that the object is associated with. Using SQL query statements, it is possible to combine multiple tables using a system of comparing primary and foreign keys, also known as joins.

- **Architectural Influence:** The team is not knowledgeable about the specifics behind how MySQL computes its relations. The way the team interfaces with it is using MySQL query language and we follow the syntax of the language in order to achieve our desired results.

4.0 Module and Interface Descriptions

Various modules were developed in order for there to be different subparts to the project that each perform a specific task or function. Modularity is important so that a set of tasks can be worked on at different times without conflicts and if something breaks in a module of code it does not affect the entire project. Without modularity, code would be difficult to understand and difficult to work on at the same time by various team members.

The UML diagrams may not be at the highest resolution possible. The team has determined that this is the fault of how Google Documents processes images.

4.1 Desktop Application

To reiterate from section 3.1, the desktop application is built using Unity Game Engine and its primary function is to allow teachers to manage courses and the super admin to manage the teachers. Students will only be able to access classes on the website after a teacher publishes a course from the desktop application.

4.1.1 UML Diagrams

Figure 4.A: Super Admin Functions

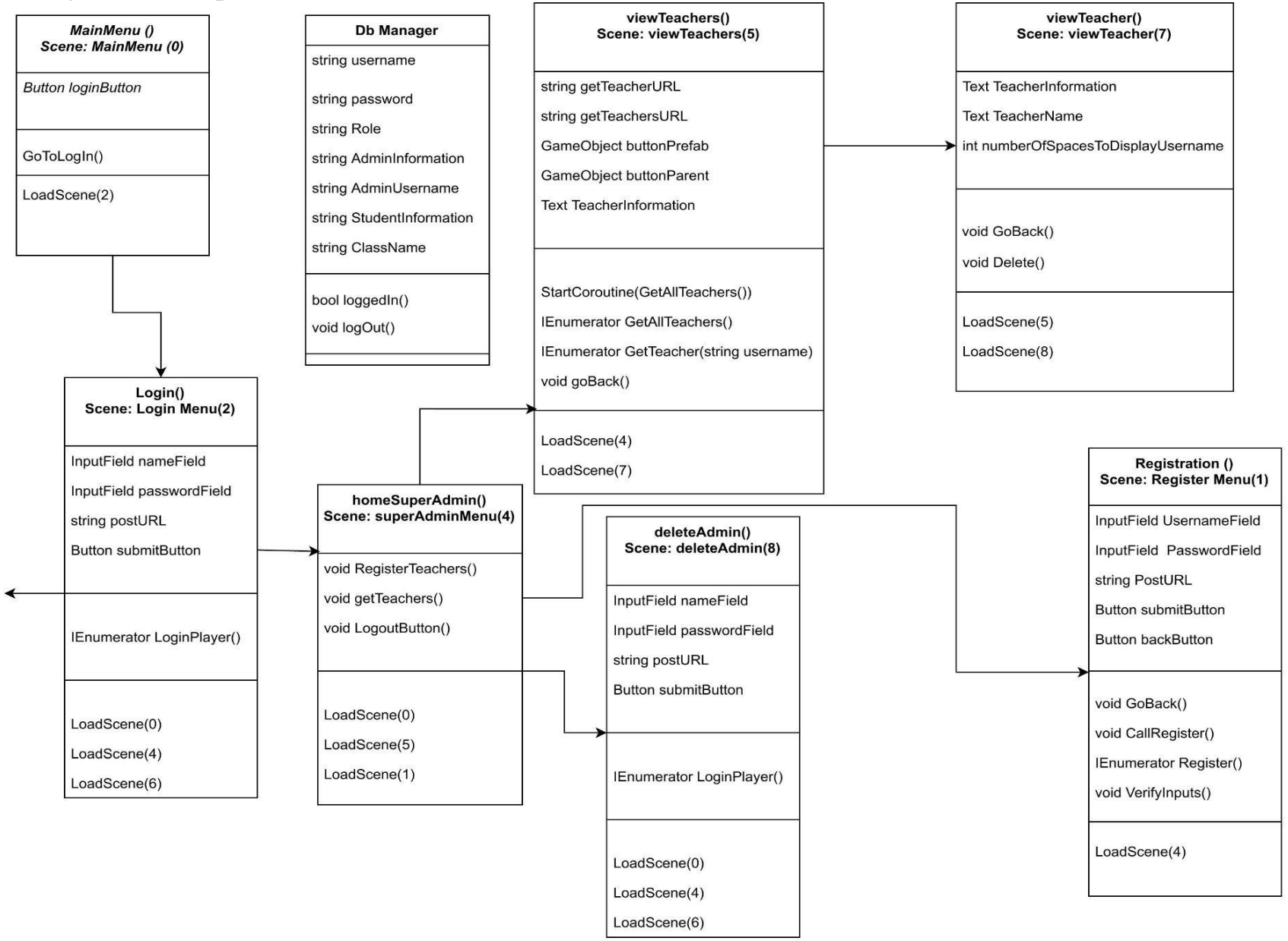
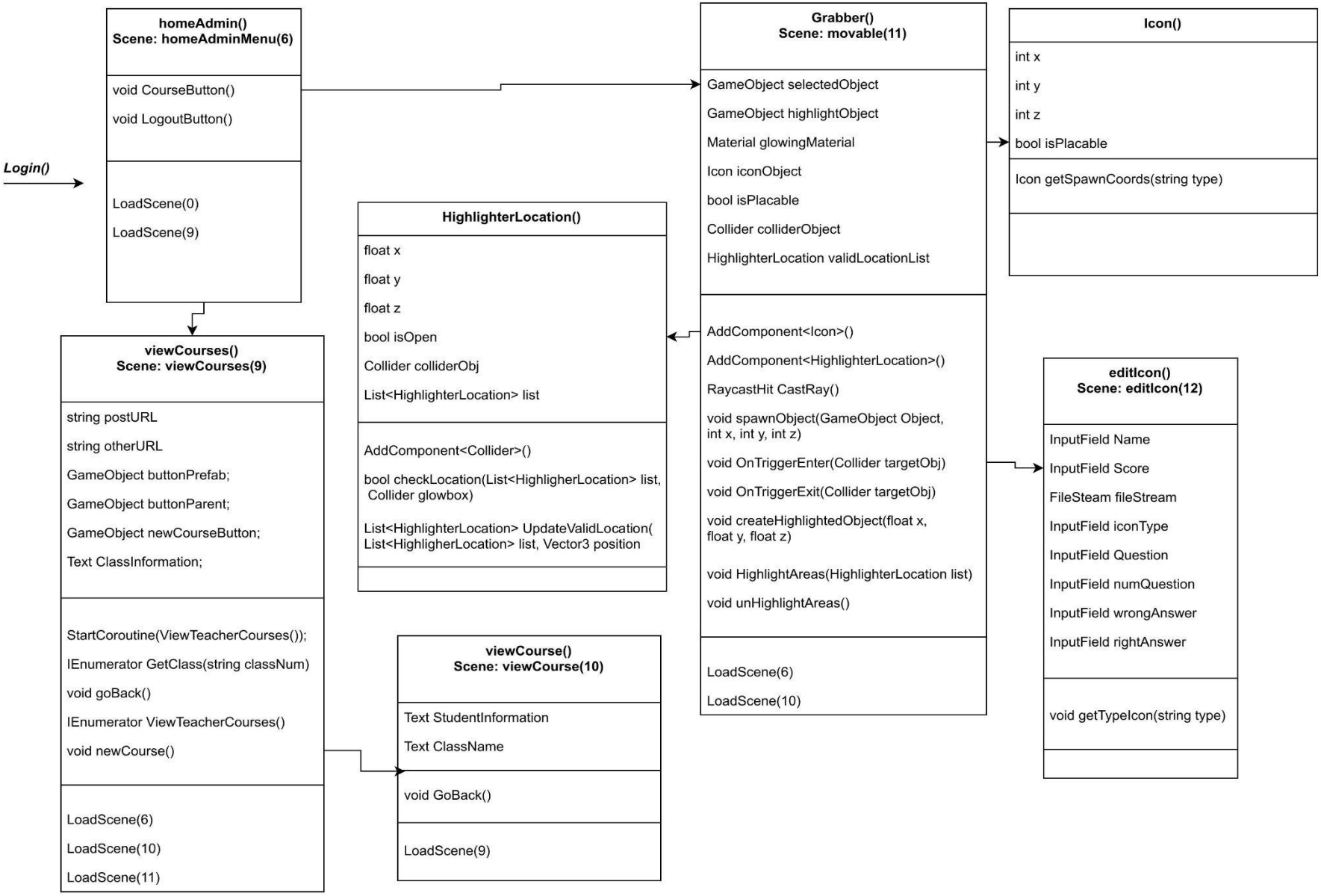


Figure 4.B: Teacher Function



The diagram has been split into two for space. Figure 4.A shows the different classes that the super administrator will have access to inside the desktop application while figure 4.B shows the different classes the teachers will have access to. The left of the Login( ) box in figure 4.A connects to the homeAdmin( ) box in figure 4.B. Each box represents a scene that a user can navigate to. A scene is something that the user sees with 2D/3D objects on screen. Scenes will have user inputs, drag and drop, and clickable functionalities. The arrows represent the steps on how a super admin or teacher will be able to access different scenes. The number after each scene name is the scene order in Unity. Each scene has a C# class associated with it and each class name is related to the scene name.

4.1.2 Public Interface

The services provided are split into two different parts depending on who is logging in or what role they have been assigned. A role dictates what privileges to give to individual users.

If a user logs in as a teacher, then that user will have access to all the material in the classes they specifically teach. This information includes a list of all the students in a particular class. Each student will have information viewable by the teacher such as but not limited to their username, what class they are in, as well as their grade.

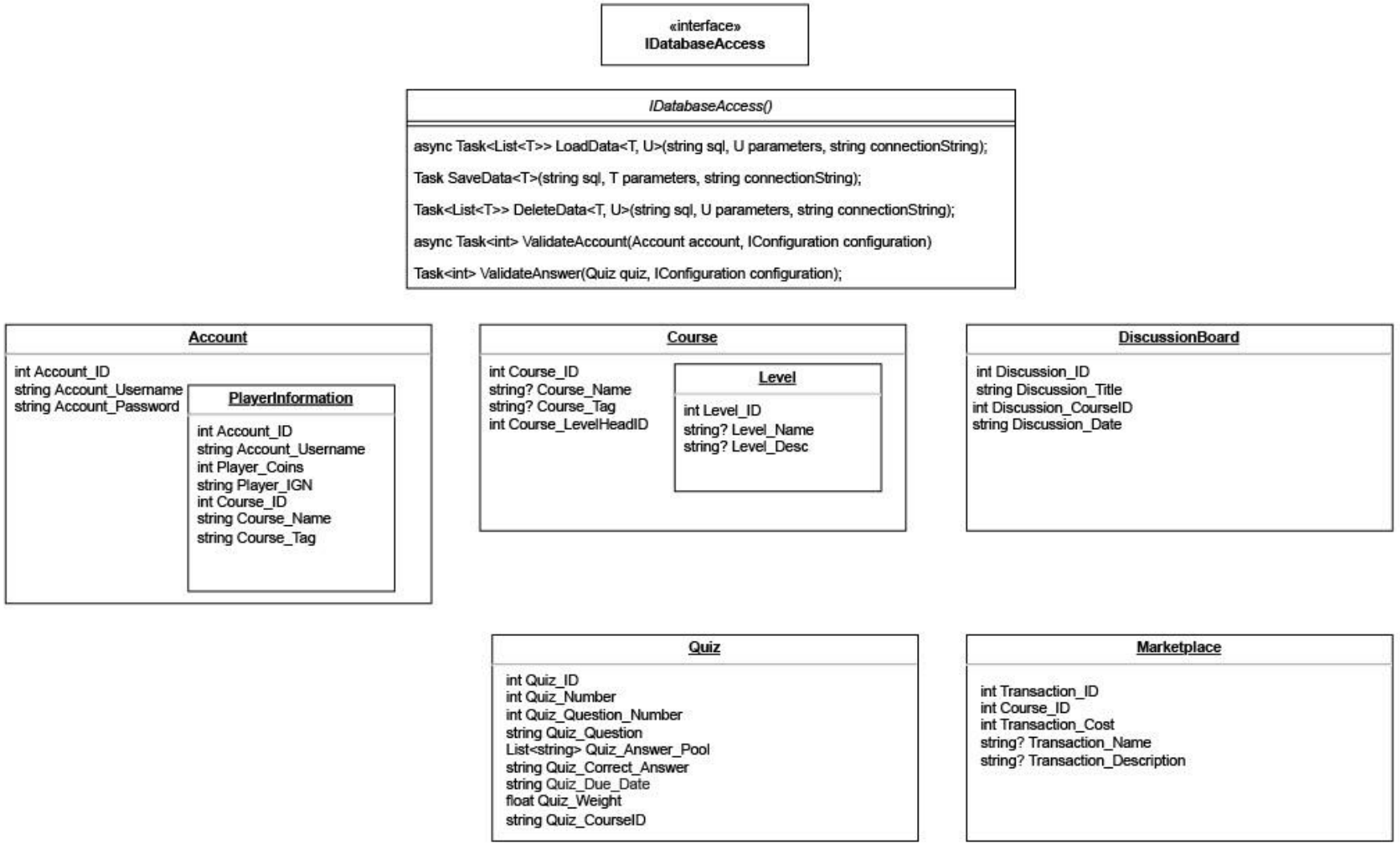
If a user logs in as the super-administrator role, that user will have access to information on all the individual teachers. Such information on the teachers could include, but is not limited to: the classes they teach, their username, and their role. The special permissions given to the super-administrator allows them to register a new teacher or delete an existing teacher.

4.2 Website

To reiterate from section 3.2, the website application will be where students take a course. The core functions that students will have is viewing their profiles, complete course material, access the course's marketplace, and access to the course's discussion board. Buttons will be provided to navigate to different sections of the website, but it is possible to type in the correct URL of the subdirectory to access them as well.

4.2.1 UML Diagram

Figure 4.C: Website UML Diagram



The different parts of the website that hold the data from the database are called classes. The variables in these classes are directly related to the variable names in the database. In order for the user interface to display properly, we must request data and store it in the class variable. From there, we can start manipulating the website to display dynamically.

This diagram displays the different classes inside the website application. *IDatabaseAccess* is accessible by the entire website and is the main connection between the website and the database. Since a lot of requests to the database use the same method, only different SQL strings, we are able to only use the one class. The only two methods(ValidateAccount() and ValidateAnswer()) that are unique to a specific model in the website are designated by the syntax

4.1.2 Public Interface

Some of the methods created use the *async* method declaration. An async method allows the program to move on and execute other functions/methods. You then end the async method with an await() operation to synchronize the data with the program. We use this operation a lot when doing any request to the database. For example: displaying the discussion board to a user will display “Loading..” until it gets the discussion board information from the database.

4.3 Database

To repeat from section 3.3, the database is how the program systematically stores and organizes data from both the website and the desktop application. The database allows the programmer to create views that can be manipulated and filtered to allow for select information to be returned to the website and desktop application.

4.3.1 Database RDS Architecture Diagram

Figure 4.D: Database UML Diagram

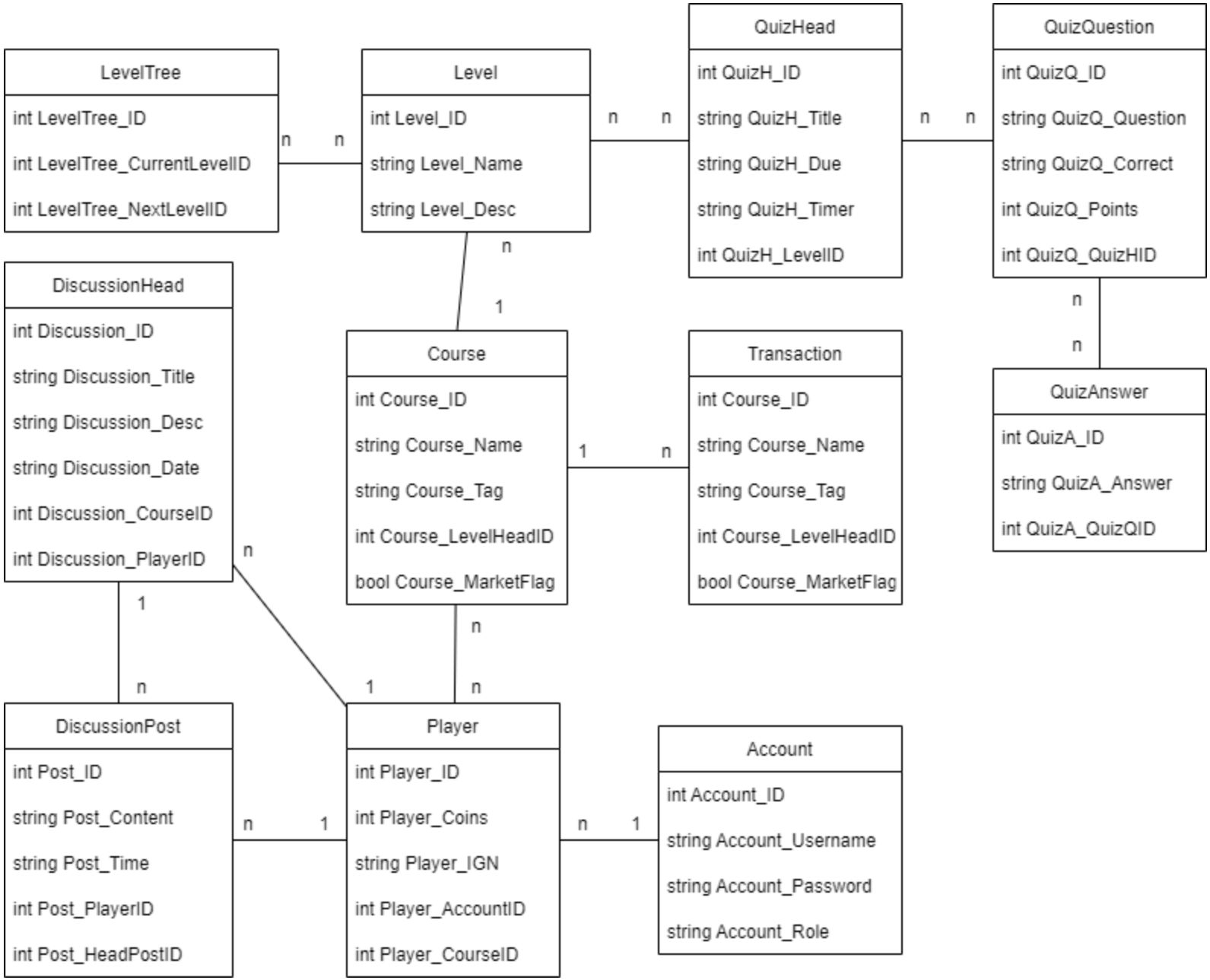


Figure 4.D is the current UML of the RDS. The format of all the variables start with the keyword of the table followed by an underscore and ends with a camel-case of the variable name. All of the primary keys have a variable name of ID. The foreign keys are slightly different in that the variable name is the keyword of the linked table with ID on the end.

The two main features of the database are the keys and the relations. The database currently uses one-to-many and many-to-many relationships, represented by combinations of “1” and “n”. One-to-many relationships mean that one copy of a table relates to multiple copies of the other table. Many-to-many relationships mean that several copies of a table relate to several copies of another table.

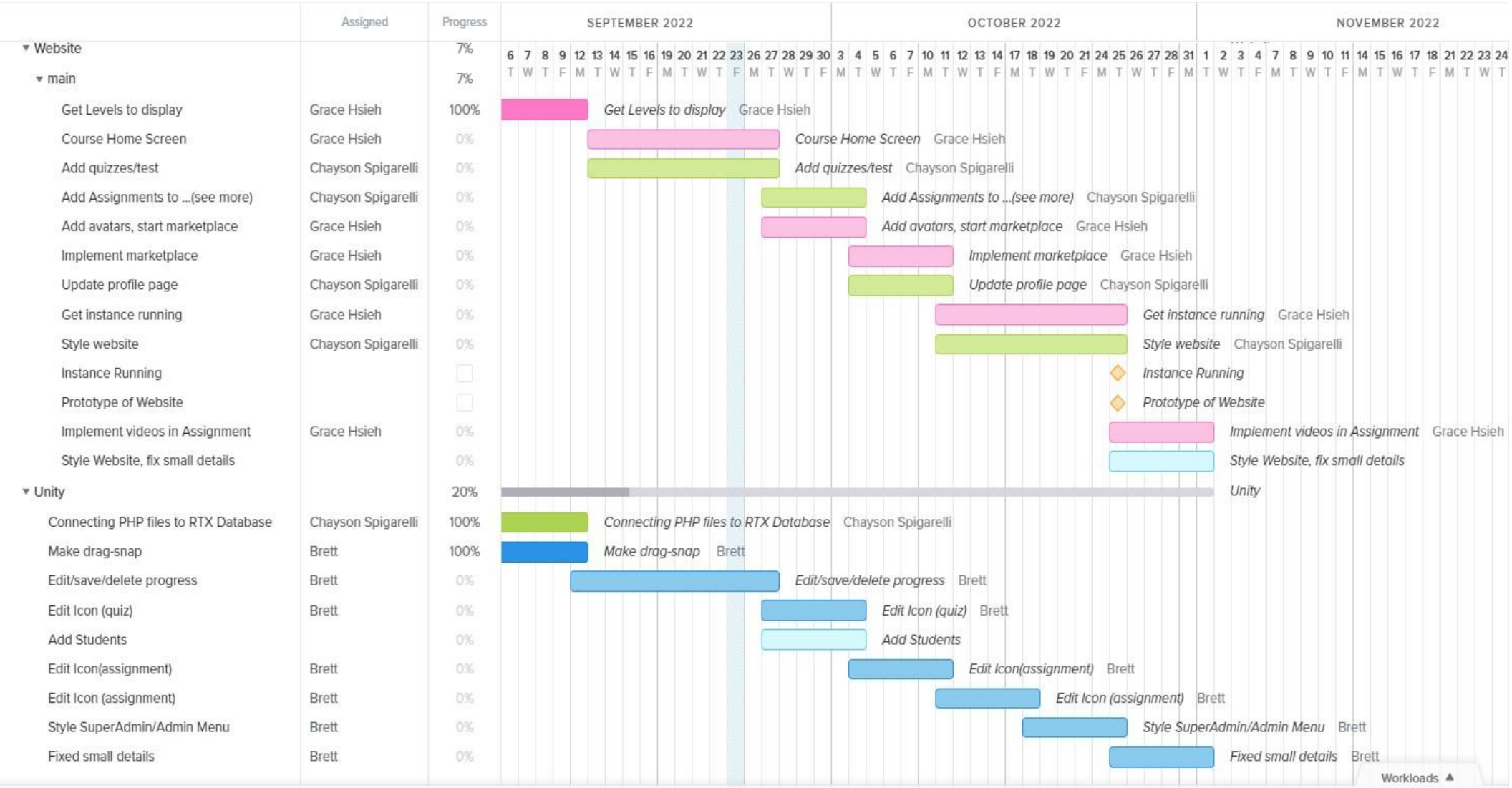
4.3.2 Public Interface

By nature the database does not have a public interface. In order for a program to access the database, it must have the username and password associated with the schema. Furthermore, the database does not have set functions in the same way that other coding languages might have libraries. SQL queries must all be custom created and are not reusable for a majority of the project.

5.0 Implementation Plan

Scheduling the implementation of various components described in section 4.0 will be completed using sprints. Sprints are components that will be individually assigned to different team members. Sprints will be assigned on a weekly basis. Once a week the team will get together and show the other team members what they have been working on for their individual sprint. During this meeting each team member's sprint will be evaluated by the other team members to ensure adequate quality. After each team member's sprint has been evaluated then all team members will agree on assigning new sprints to each team member. If a sprint is not completed on time tasks should be strategically allocated to the next sprint in order to stay on track.

Figure 5.A: Gantt Chart of Sprints



The Gantt chart shown in figure 5.A shows the sprints the team is currently working on. The sprints are divided into two different categories based on what part of the program the sprints are related to which includes Unity (desktop application) and the website (website application). The dates are shown on top of the Gantt chart. The tasks and who they are assigned to are described on the left side of the Gantt chart as well as in the middle of the Gantt chart. The team is planning on having all of our sprints completed by October 31st.

6.0 Conclusion

Many design decisions were made by the team over the course of this project. This team is excited to begin implementing the technologies researched during the previous semester. The three main parts of the project include the database, the desktop application, and the website application. The database will connect the desktop application and the website application so that they can both manipulate the same data. The desktop application will be completed using Unity Game Engine. The website application will be completed using .NET Frameworks. While the database will be completed using the MySQL RDS.

The main goal of this project is to have students enjoy doing their school work more. In order to do that the team's sponsor has given us the idea to make a gamified LMS. Our idea of a gamified LMS is to create a LMS from scratch and then add additional features to make students using the LMS feel like they are playing a video game instead of working on school work. Gamified features brought to us by our sponsor Professor Terry E Baxter include but are not limited to an in-course currency, custom avatars, and much more.

The team is progressing through the project quickly thanks to the teams planning from last semester. All of our technologies have done the jobs we have expected them to do including being able to create a desktop application using Unity and being able to access the website application using any internet browser. The desktop application currently allows super-administrators to register, delete or view teacher information such as their name, username and class they teach. A teacher can view the classes they teach as well as the students in each class they teach. The website application has a quiz, account, discussion board, logout, and marketplace pages and is currently working on creating the test, assignment pages, and profile pages.

The team hopes this document serves as a valuable insight into our project for any future developers. The UML diagrams allow us to visualize in the natural language about how our program will function. By splitting our program up into various modules the team is well equipped to complete the project successfully and on time to the team's sponsor.