DAT602

Milestone Reviews

David Rowe
STUDENT ID # 13511541

Contents

Milestone one

Creating a game design using wireframes that demonstrated live gameplay player registration and player selection with administrative functions as well as a crow's foot notation, CRUD table, and SQL.

With Milestone one, I set out to design a game based on a database with MySQL. To-Do this I came up with a specific game design I wanted to implement without knowing much regarding MySQL and with very shallow knowledge of databases and how they worked. The first step was to construct the game I wanted to make using wireframes, how it would look and what the processes would be, and how each Character, User would interact. In theory, everything made sense and I knew just how I wanted my game to look and play. I started off creating a wireframe of the main screen which would follow onto a new screen that presented the user either the option to create a new account or login with a previously created account. After a user was created and logged in it would then create a custom character for that account with limited customization such as colour and a type. The Game would then proceed into a game screen where you were able to move the character around kill monsters and increase your score and stats. I chose an 8bit style game due to it being quicker and easier to draw sprites and objects needed to represent characters and items. To draw these I used Microsoft paint, due to ease and they generally turned out pretty good considering the 8bit theme. This design phase I really enjoyed as it allowed me to be creative and dream up ideas.

The next phase was to create a crow's foot notation for the database showing the tables' primary keys foreign keys and their relationships to each other. This worked as a good visual representation and guide when creating a CRUD and writing SQL which proved helpful.

Creating a CRUD, create, read, update, and delete a table. This table would be used to identify the events which would cause the data in the database to be created, read updated, or deleted. This helped easily see what events would transpire and what data would be affected for each event and how it would be handled in SQL. This CRUD table was good to reflect on every time I was creating a table or procedure the combination of this, and the crow's foot notation worked as excellent blueprints for my database game and SQL.

Writing SQL was a new experience for me and what little knowledge I had on it was from my first semester over 18 months ago. Although difficult at first, I found I picked it up reasonably quickly once I got the hang of the format and Delimiters.

Milestone two

Further development on the SQL for my Game design as well as procedures that we're able to be called and tested with C#.
Creating procedures was a little tedious at first but after spending a fair amount of time writing them they started to make a lot more sense. I needed to write procedures in my database to connect my database to game events. Each time an entry was created in my database I needed to work out what fields were required and their possible relation to other tables. I then needed to test the procedures with the parameters created by calling them with their parameters. The procedures I needed to create had to represent Registering a user into the database, being able to log in to the application by reading the username and password from the tblPlayer table, and validating that the password matched the username. From there I needed to write a procedure that customized a character and assigned it to the username that created it. The character customization procedure assigned values

such as character color, character type, and stats. After creating the character I needed to have a way of spawning it into a gameboard which made me realize I had not yet created a gameboard where the character was to spawn so the next logical step was to create a procedure to generate a game board with tiles. Creating the procedure for the gameboard required a for each which generated a grid of 6 by 6. Each tile had an ID and each ID had a row and column starting at TileID 1 = row 0 column 0 and incremented until each row had 6 columns with a total of 6 rows. From there I was able to write the spawning procedure that originally had a random function that would assign it to a random TileID from 1-36 this identified which row and column the character would be on which would later be implemented into a grid when I came to creating the GUI. The game also required enemies or monsters and items for the player to interact with so using the same ideology I created procedures for both monsters and items using the same spawn method and assigning them to TileIDs. The Characters also needed a way to move around the board I figured that if a character or monster TileID was either increased or decreased using a function it would theoretically move from one grid to another. As my grid was 6 x 6 increasing TileID + 1 would move to the right, -1 would be to the left, +6 would be down and -6 would be up. So of course, I needed to write procedures for each of those and named them accordingly. I later created more procedures for the admin to edit characters among others and by the time I had written all these procedures I had found that I had become quite familiar with them and how they worked.
After creating countless procedures with parameters, I then needed to connect my Database and its procedures with C# This was done by creating a class called DataAccess and inputting the following code.

```
private static string connectionString
        {
            //conecting to SQL database name and password
            get { return
"Server=localhost;Port=3306;Database=WarlockGame;Uid=warlock;password=12345;"; }

        }

        //connection to MYSQL
        private static MySqlConnection _mySqlConnection = null;
        public static MySqlConnection mySqlConnection
        {
            get
            {
                if (_mySqlConnection == null)
                {
                    _mySqlConnection = new MySqlConnection(connectionString);
                }

                return _mySqlConnection;

            }
        }
```

This would allow my C# application to log into my local database and supply it the information such as the Database and password to be able to connect and interact with it.

From there I needed to start calling my procedures using C# by using message strings and assigning parameters to the procedures it was calling. At first, I struggled to get a grasp but once I had 2-3 Tests that called the procedures on the database it became quite easy although every now and then I got errors which generally came down to calling the insufficient amount of parameters or assigning them to the wrong datatype such as Int where it should have been string and vice versa.

Finally, when I ran the testing application it would send back the messages in the procedure select ‘message’ as the message section. And writing them as a Console.WriteLine(); which would be displayed on the console app to verify that the procedure worked.

Milestone Three

Implement the Database and connect it to a GUI using C# forms.

This Phase I found quite difficult and the result wasn’t as good as I would have hoped by the timeframe I had to work with. However, I plan to complete this application in my free time as it has helped me improve my C# and MySql knowledge tenfold. I found I got stuck on trivial things that soaked up more time than it was worth and I tried a variety of different approaches to achieve the desired outcome. Creating the forms I had a bit of experience with through a software development class I was also taking at the time but creating my own game that I had designed gave me the flexibility and freedom to explore C# forms more in-depth adding unnecessary things such as background music, artistic buttons and custom backgrounds for my forms. However, I struggled and spent hours constructing a board that would assign items characters, and monsters to it using a combination of GridViewBox values and PictureBox’s. I assigned PictureBox1 to retrieve the information from GridViewBox TileID 1 TileStatus. PictureBox2 TileID 2 TileStatus and so forth. Eventually, I had a gameboard that interacted with my Database however the information never updated live when I tried to move the player even though the player location was updated on the Database itself. As frustrating as this section was I did enjoy the problem-solving and knowledge I acquired along the way and I am currently still learning how to implement these game design features I designed from milestone one. Despite this, I still managed to use and customize the call functions I created from the testing phase in milestone 2 and created an authentication system, and created buttons to move characters on the database. The Game Warlock is still a prototype in production I want to work on and finish it someday as frustrating as it can be the satisfaction you get when something works after problem-solving for hours is worth it.