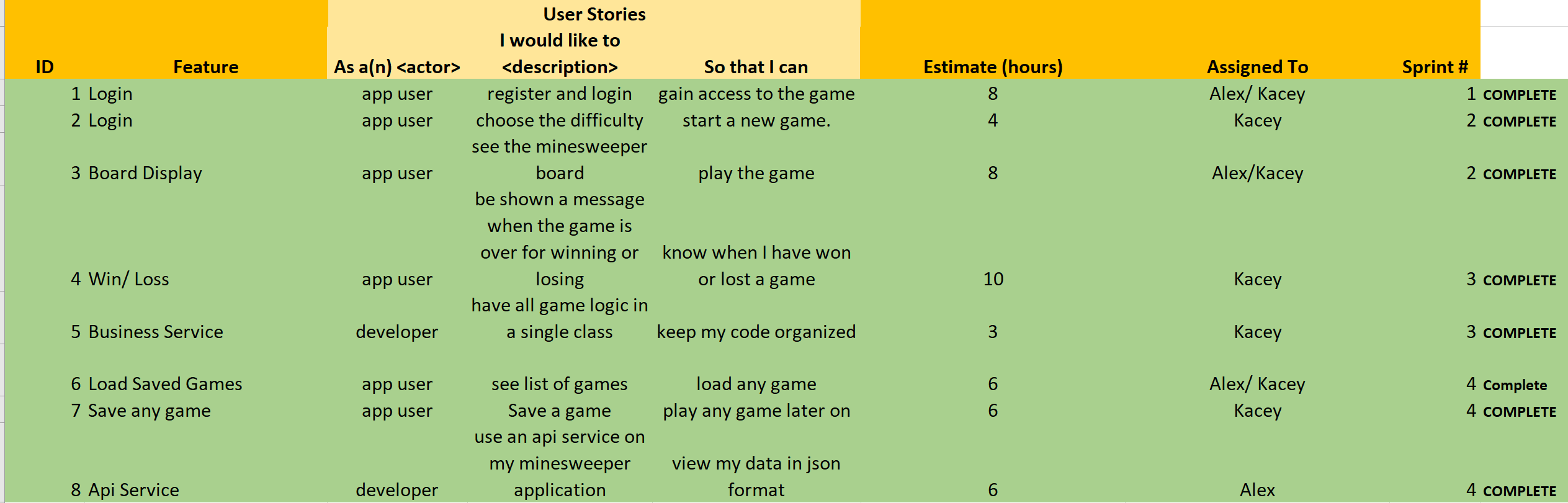
**.NET Application Programming**

**Project Status and Design Report**

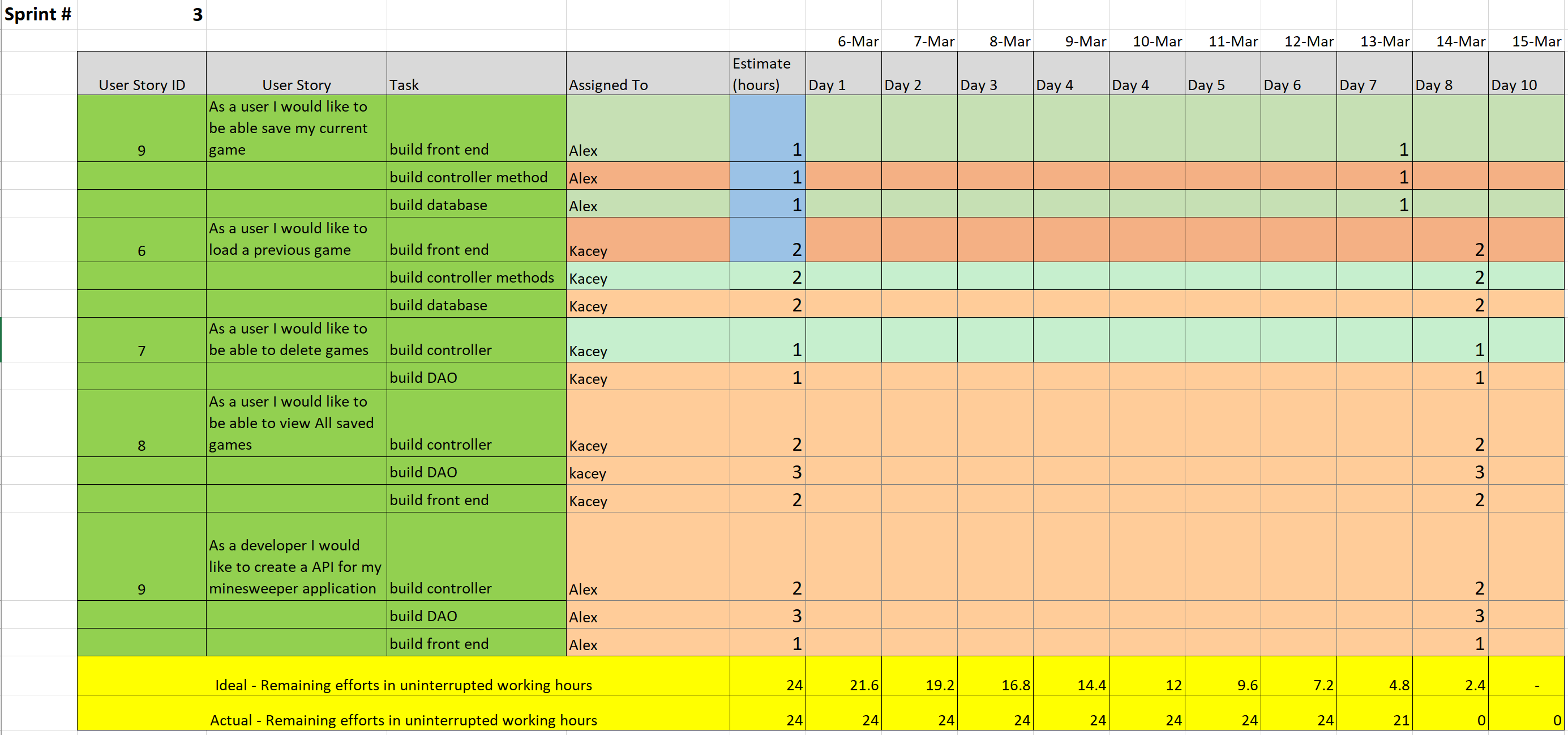
|  |  |  |
| --- | --- | --- |
| **Topic:** | *Milestone 4 API, Saving, and Loading* | |
| **Date:** | *March 28, 2021* | |
| **Revision:** | *1.4* | |
| **Team:** | 1. Kacey Morris | |
| 1. Alex Vergara | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | *As a user I would like to save a game.* | *Alex / Kacey* | *5* | *0* | | *As a user I would like to load previous games.* | *Kacey / Alex* | *7* | *0* | | *As a stakeholder, I would like an API to interact with other applications.* | *Alex* | *2* | *0* | | *As a product owner, I would like to have updated documentation to keep up with the project progress.* | *Kacey* | *2* | *0* | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | *https://github.com/KMorris63/247Milestone.git* | |
| **Peer Review:** | *Yes* | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |

**Planning Documentation**

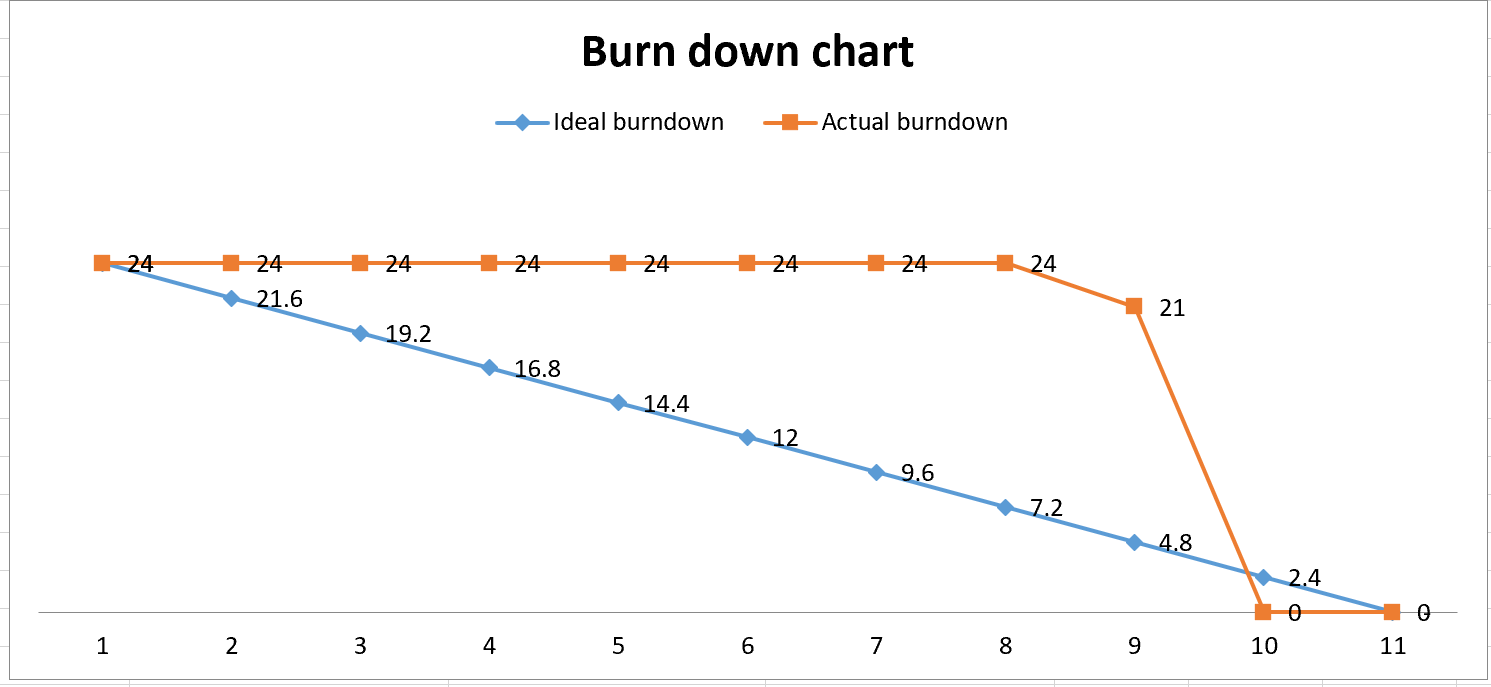
**Agile Scrum Product Backlog:**



**Agile Scrum Sprint Backlog:**



**Agile Scrum Burn Down Chart:**



**Agile Retrospective Results:**

*The following table should be completed after each Retrospective on Things That Went Well (Keep Doing). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool you must include a URL or Image File.*

|  |
| --- |
| **What Went Well** |
| **We were able to share the project between each other without any major issues using github and discord.** |
| **We communicated well and shared the work evenly this milestone.** |
| **We continued even when we were stuck to finish the project.** |

*The following table should be completed after each Retrospective on Things That Didn’t Go Well (Stop Doing) and What Would Be Done Differently Next Time with an Action Plan to Improve (Try Doing and Continuous Improvement). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool you must include a URL or Image File.*

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
| **We started the milestone a little late** | **We need to complete our homework in other classes, so we don’t get left behind** | **3/15** |
| **We need to schedule a time to work on the project together** | **We need to catch up on other classes first** | **3/15** |
|  |  |  |

**Design Documentation**

**Install Instructions:**

The installation process for this milestone is straightforward. As provided above, copy the GitHub URL and clone the repository using visual studio. Afterwards, you will need to create a database within SQL server object and name your database to dbo.users. Afterwards, include the following columns in the database; first name, last name, gender, age, state, email, username, and password. Create another database named dbo.games. Include columns in that table for Id, JsonString, userID, and playedDate. Then, change the connection string within SecurityDAO and GamesDAO classes to the database’s connection string located within properties. Lastly register a user and then login. The project should work after following these steps.

**General Technical Approach:**

We need to implement a functioning minesweeper game within our website. The user should be able to start a game and start playing. Last semester we created a functioning minesweeper game therefore, all we need to do is import the project’s classes and refactor the code. We decided on using Kacey’s code for the minesweeper board. Moreover, features such as a timer and highscores have been left out because they will be implemented in later milestones. Additionally, we need to utilize CSS to design our minesweeper board and provide a better user experience. Right click and win loss messages were implemented in this milestone, as well as partial page updates and a business service for game logic. For right click, only the button that was right clicked is updated using ajax and javascript. By checking the status of the button, a user cannot left click a button that has been flagged. For win loss messages, information is sent to the javascript ajax methods letting the code know if the game is still running and, depending on the game status, an alert is displayed at the top of the screen if the user has won or lost the game. The partial page updates are handled using ajax as well. The javascript method intercepts the click of a button and redirects that click to the ajax methods. From there, ajax sends the button that was clicked to the desired controller method. When the controller method returns data, that data replaced a section of the page. A timestamp is placed at the top of the UI to prove partial page updates are being implemented correctly.

For Milestone 4, save and load were also implemented. We decided to use a table to display the games previously saved by the user and have buttons within the table to allow for complete control by the user. The information about each game is saved in the database and only the user who originally saved the game has access to the games associated with a certain ID. The user is also only able to save a game while in the play page. This can still be accessed from the difficulty selection page as well as through previously saved games. The API includes functionality for returning JSON data for all games, a single game by ID, or deleting a game by ID. These API’s provide JSON formatted data and allow for interaction with other applications.

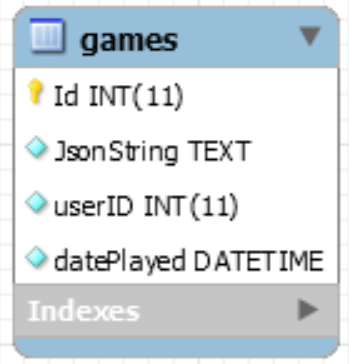
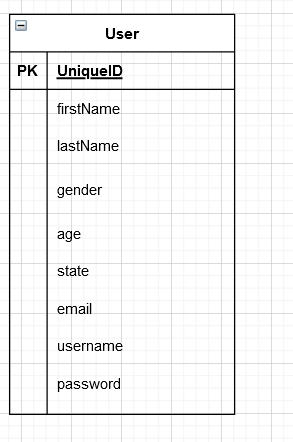
**Key Technical Design Decisions:**

We will be utilizing C# .net core for this project. .Net Core also allows us to create a clean user interface without using much CSS, although we will more than likely need to research CSS on our time to create our own design for the minesweeper board. As requested we will be utilizing our code from last semester to implement our minesweeper game. This is going to help us although we will need to learn how to adapt to the MVC pattern.

For right clicks, we have decided to update just the button that was clicked, but for left clicks, the entire grid is updated to allow for flood filled squares to appear. When the game ends, another partial page of a fully exposed grid is displayed to give further clarity to the user. This was to ensure the user knows where they went wrong, or to see all they have accomplished. In this exposed grid, the buttons are labeled in a different format so an alternate javascript method is triggered when the buttons are selected. This is to prevent further game play by the user after the game has ended.

For the new games database design, we decided to save information about the json string to actually load the game as well as the user ID to connect each user to their specific saved games. The date and time the game was saved is also saved in the database to help the user distinguish between games better when loading from the saved games list. All games are saved, even if the game is a continuation of a previous game, so the user has delete functionality available at all times. This allows the user to get rid of unwanted saved games whenever they please. At any point in a game, the user may save the game state to allow for flexibility. With the API’s, our group decided to only implement the three due to the nature of the API. When consuming the API, the consumer should only have access to limited information about the site, and this includes games.

**ER Diagram:**

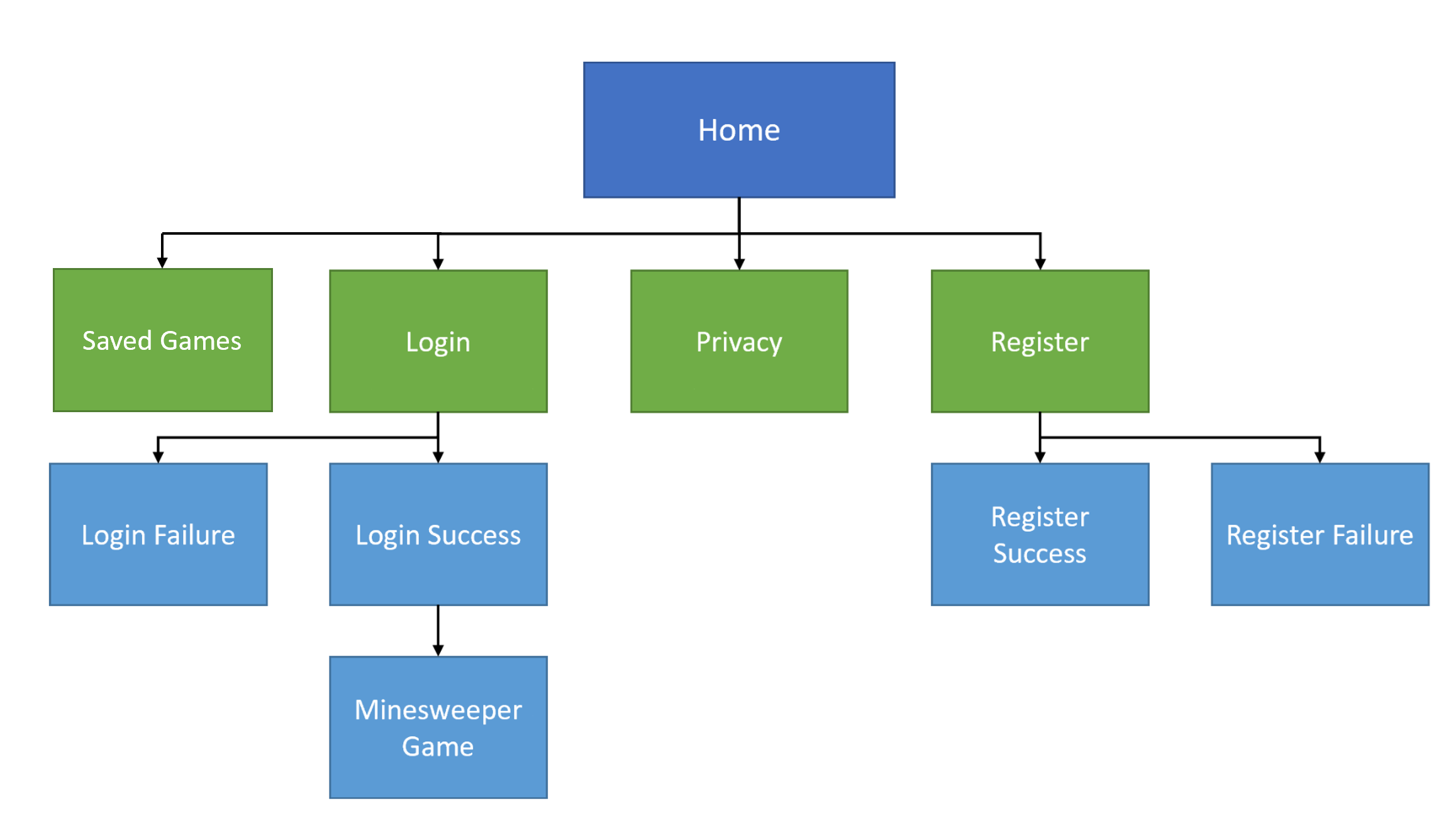


**DDL Scripts:**

*https://github.com/KMorris63/247Milestone.git*

The DDL scripts can be found at the GitHub link provided. Before using these scripts, though, the Minesweeper database must be created in the MYSQLLocalDB in Visual Studio. From there, the scripts can be utilized to create the dbo.users and dbo.games table.

**Sitemap Diagram:**



**Security Design:**

This section has not been implemented at this time.

**Third Part Interface Design:**

API Delete Saved Game –

This API can be accessed through the extension “/ApiDeleteSavedGame/{gameID}”. It will delete the saved games from the games database where the ID matches the parameter that was passed.

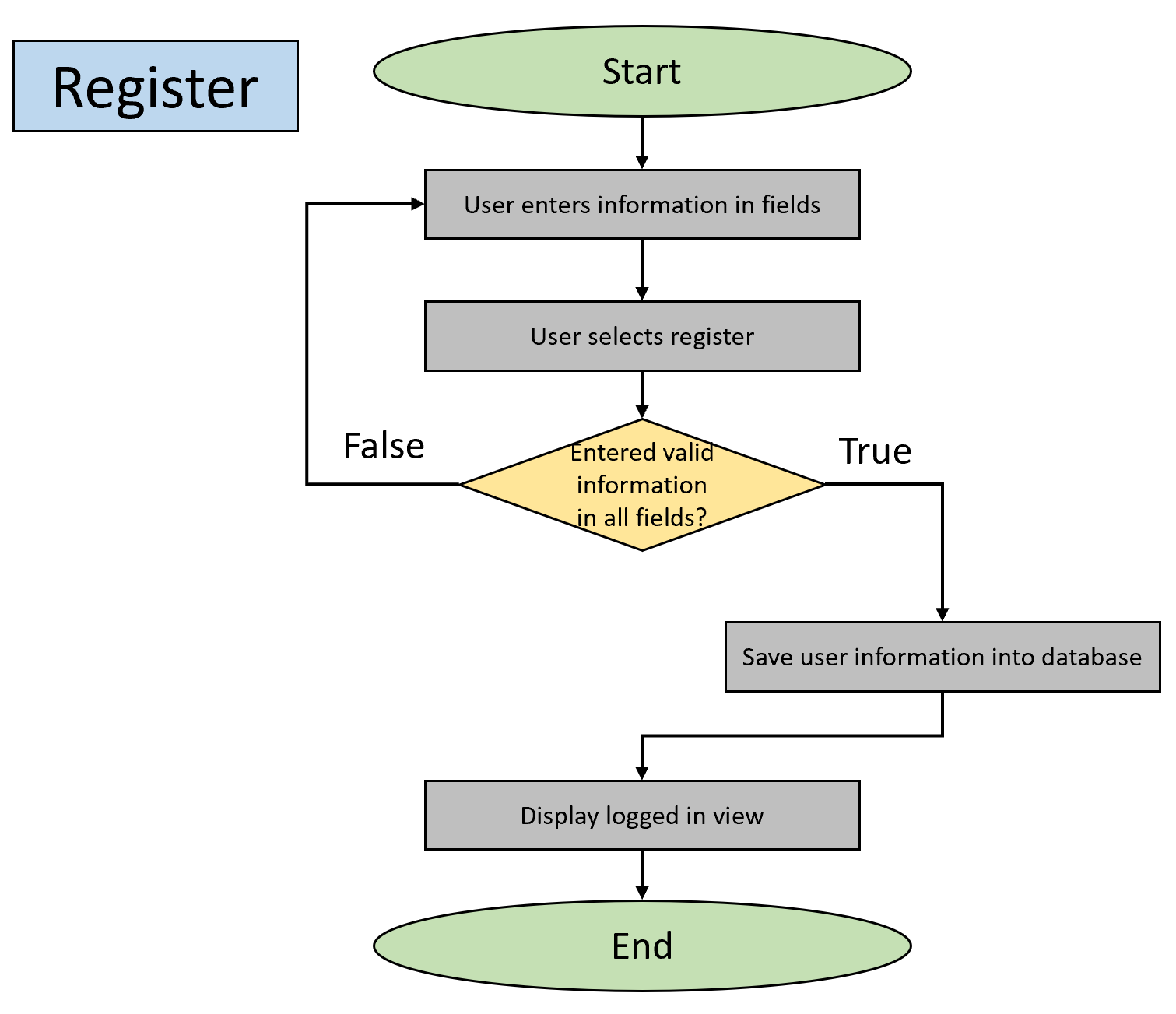
API Show All Saved Games –

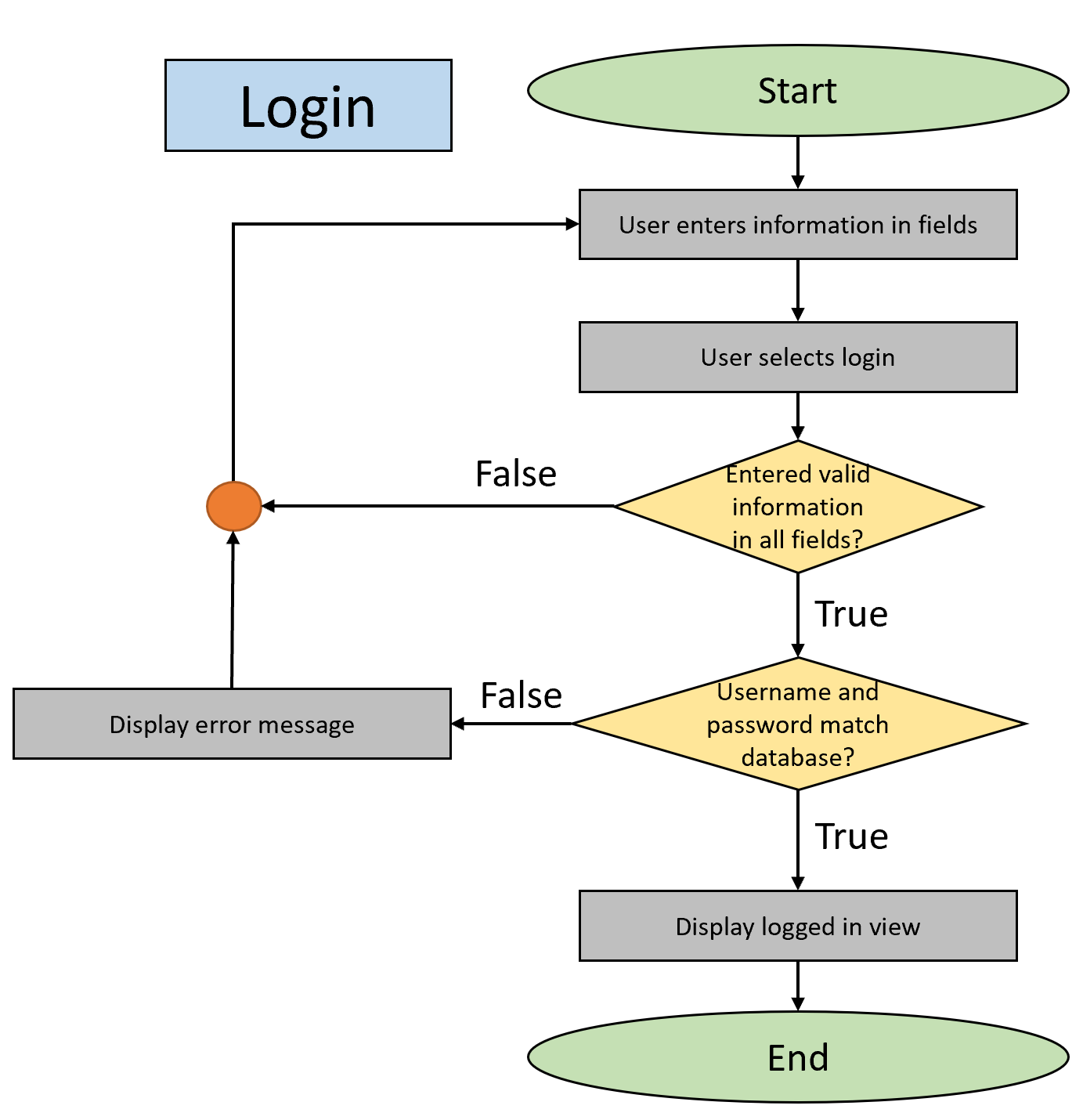
This API can be accessed through the extension “/ApiShowAllSavedGames”. It will return JSON data of all games saved in the database.

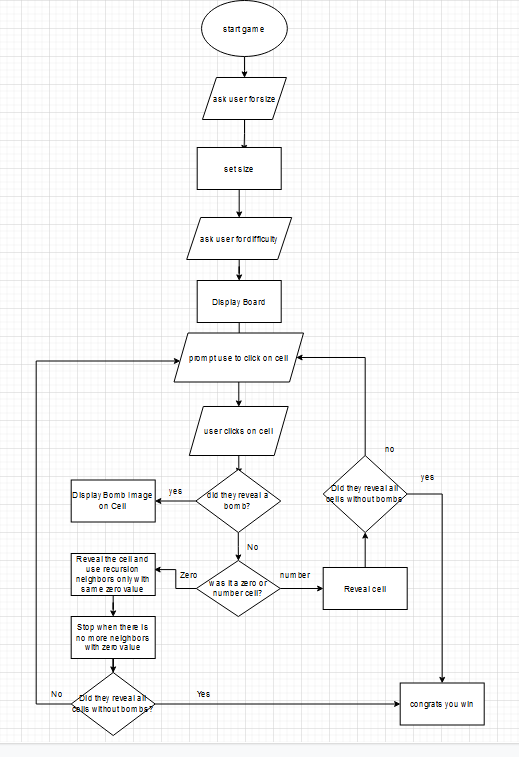
API Show Saved Game –

This API can be accessed through the extension “/ApiShowSavedGame/{gameID}”. It will return JSON data of the game in the database that has an ID that matches the parameter that was passed.

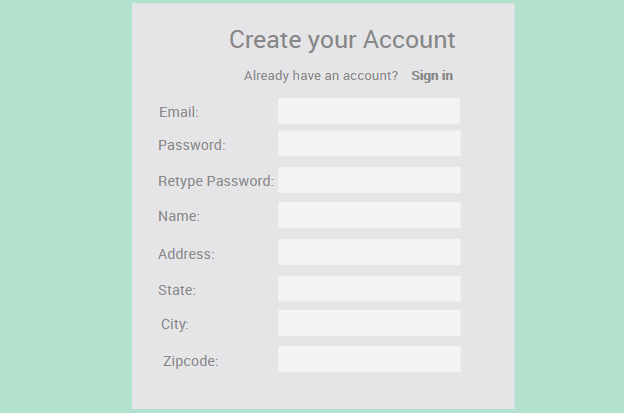
**Flow Charts:**



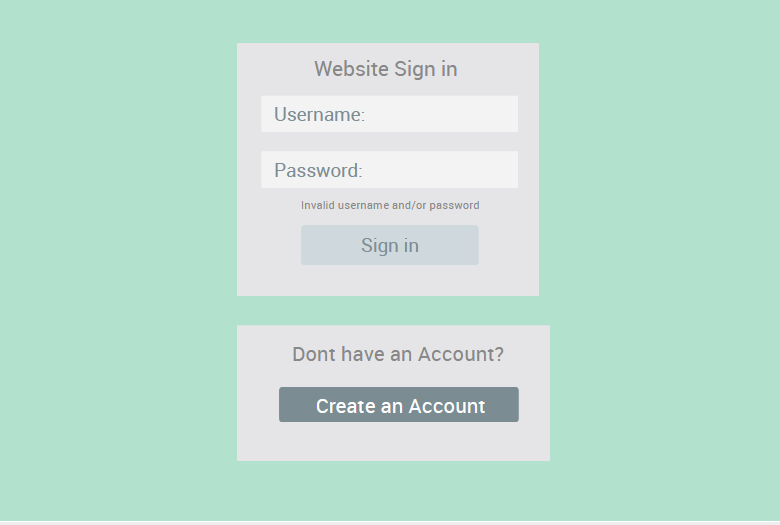




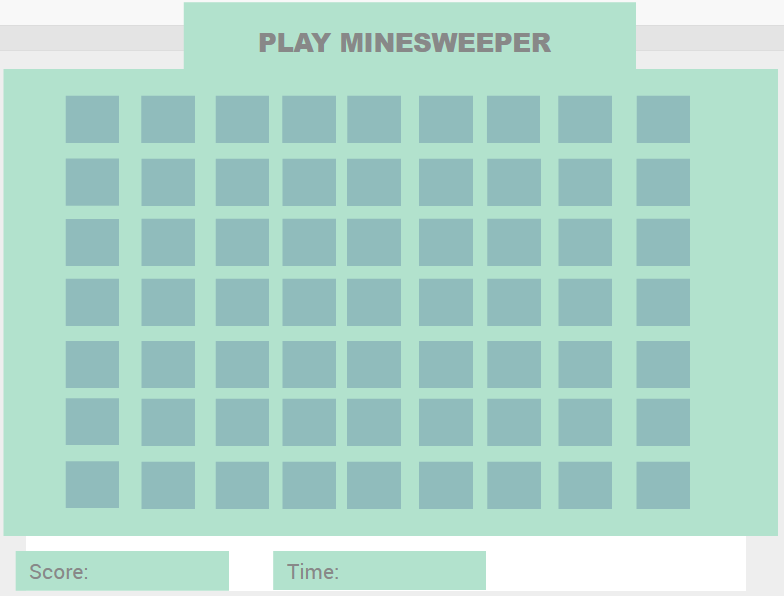
**User Interface Diagrams:**



Register Concept

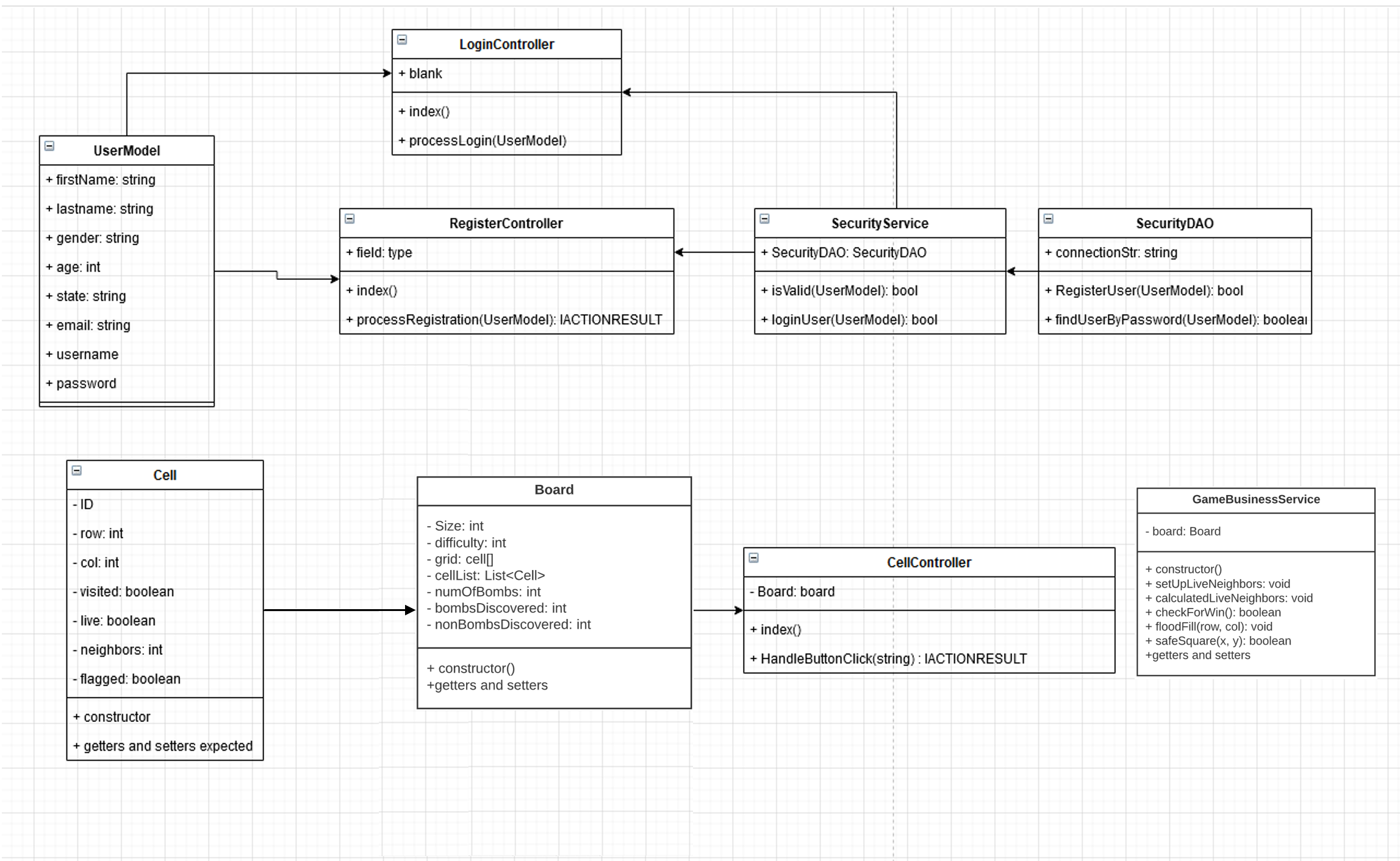


Sign in Concept



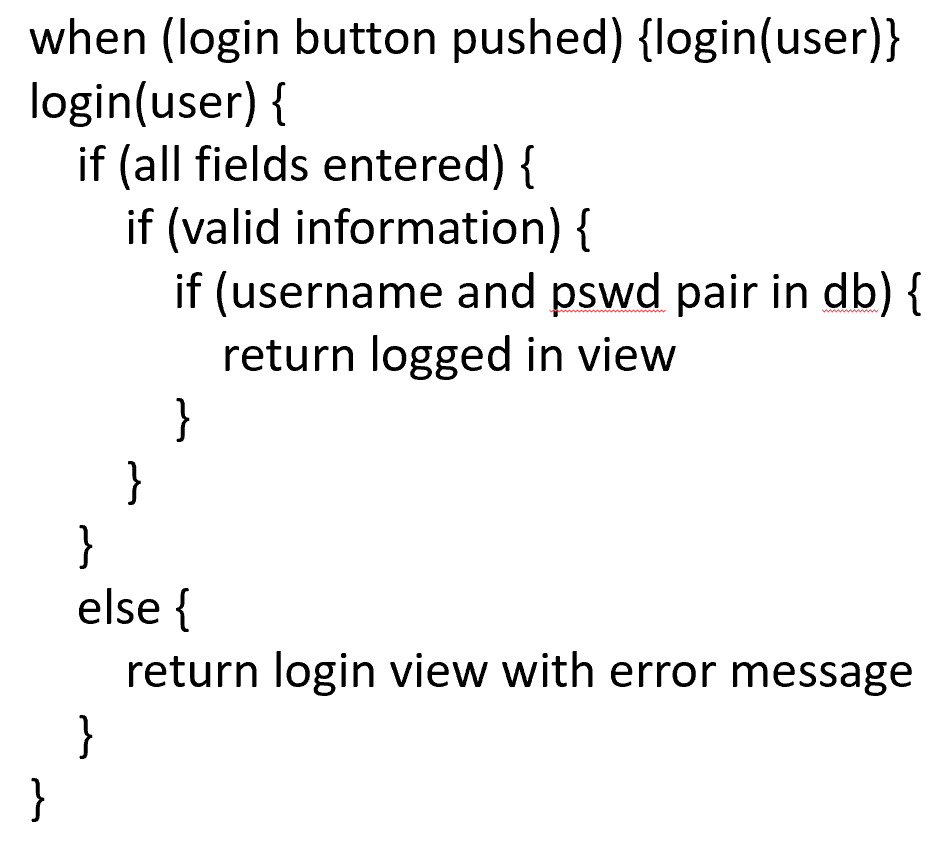
Minesweeper Board

**Class Diagrams:**

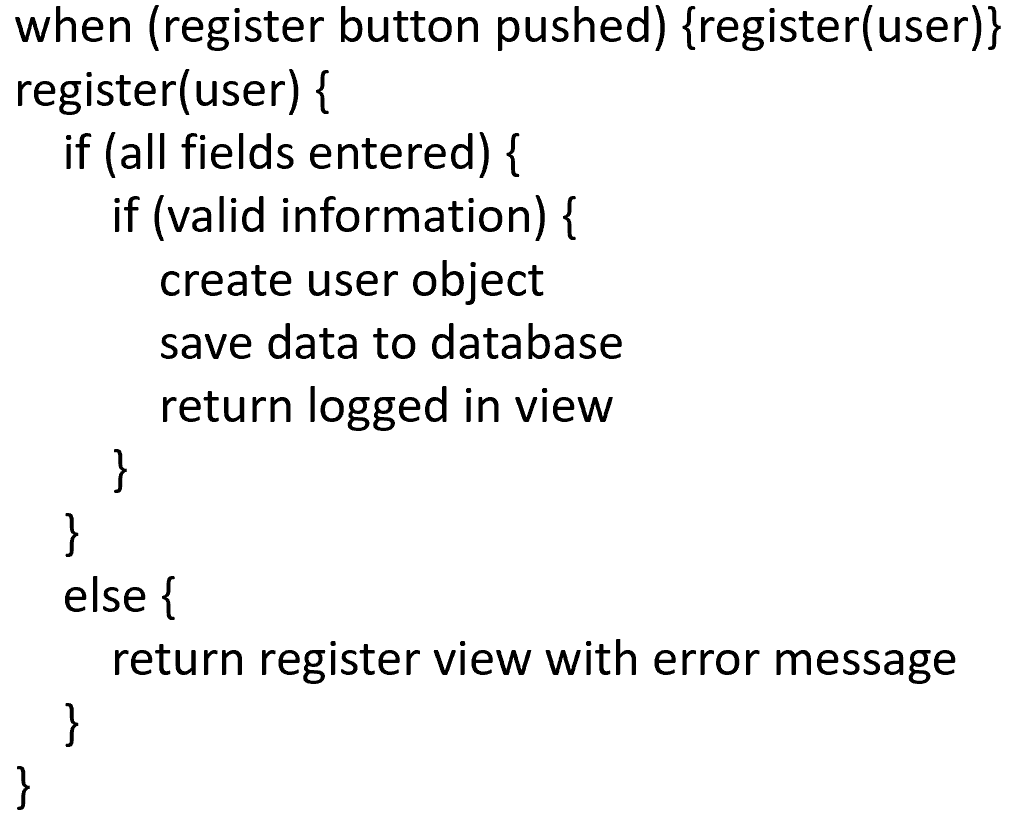


**Pseudo Code:**

Login Pseudocode:



Register Pseudocode:



**Other Documentation:**

LOOM Video:

Part 1 - <https://www.loom.com/share/41ea61a1f69a4fac820a554f978c152b>

Part 2 - <https://www.loom.com/share/8e596507acfb436b90b35196b09e9da4>