**CST-247 Design Report Template**

|  |  |  |
| --- | --- | --- |
| **Topic:** | Milestone 6: Completed Game | |
| **Date:** | June 6, 2021 | |
| **Revision:** | 1.0 | |
| **Team:** | 1. Daniel Cender | |
| 2. Nathaniel Kumar | |
| 3. Hiram Viezca | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | As a developer, I’d like to have a cleaner codebase, including updated comments, views, and file headers. | All | 3 | 0 | | As a developer, I’d like to have up-to-date documentation and project planning documents. | All | 1 | 0 | | As a user or third-party developer, I’d like for the bugs in the official API to be fixed | n/a | 0 | 2 | | As a user, I’d like to be able to save/load my current game. | n/a | 0 | 2 | | As a user, I’d like to be able to get to an end game state. (fix UI logic bugs) | Daniel | 2 | 0 | | As a stakeholder, I’d like to have a presentation explaining the full stack of this application. | Daniel / Hiram | 1 | 0 | | Communicate goals as a team | All | 2 | 0 | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | [*https://github.com/DanielCender/CST-247-CLC*](https://github.com/DanielCender/CST-247-CLC) | |
| **Scrum Product Backlog:** | [*https://github.com/DanielCender/CST-247-CLC/projects/1#column-14015513*](https://github.com/DanielCender/CST-247-CLC/projects/1#column-14015513) | |
| **Scrum Sprint Backlog:** | [*https://github.com/DanielCender/CST-247-CLC/projects/1#column-14015514*](https://github.com/DanielCender/CST-247-CLC/projects/1#column-14015514) | |
| **Hosting URL:** |  | |
| **Screencast URL:** | (Project Walkthrough): <https://www.loom.com/share/6e4da93482b84ff5935675f2a9a061bd>  (Slide Deck): <https://www.loom.com/share/97d741ca3af5453bab2a839ea10ec482> | |
| **Peer Review:** | Yes | We acknowledge that our team has reviewed this report and we agree to the approach we are all taking. |
| **Sprint Burndown Chart:** |  | Chart |
| **Sprint Retrospective Notes:** |  |  |

**Design Documentation**

**Install Instructions:**

*Include step-by-step instructions for setting up your database, configuring, and deploying/installing your application. This section should also include detailed instructions for what configuration files are required by your application, what configuration settings need to be adjusted for various runtime (development or production) environments, and where the files need to be deployed to. This section should also contain detailed instructions for how to clone your application source code from Bitbucket and deploy the application to an externally hosted site.*

Setting up the database in Visual Studio 2019

1. Open the SQL Server Object Explorer
2. Open SQL Server
3. Open MSSQLLocalDB
4. Right-Click Databases Folder and select Add New Database
5. Name the database and click OK.
6. Open the new database
7. Right-Click Tables and select Add New Table
8. The new table T-SQL Designer appears.
9. Add the relevant fields, data types according to the ER Diagram below.
10. Click Update

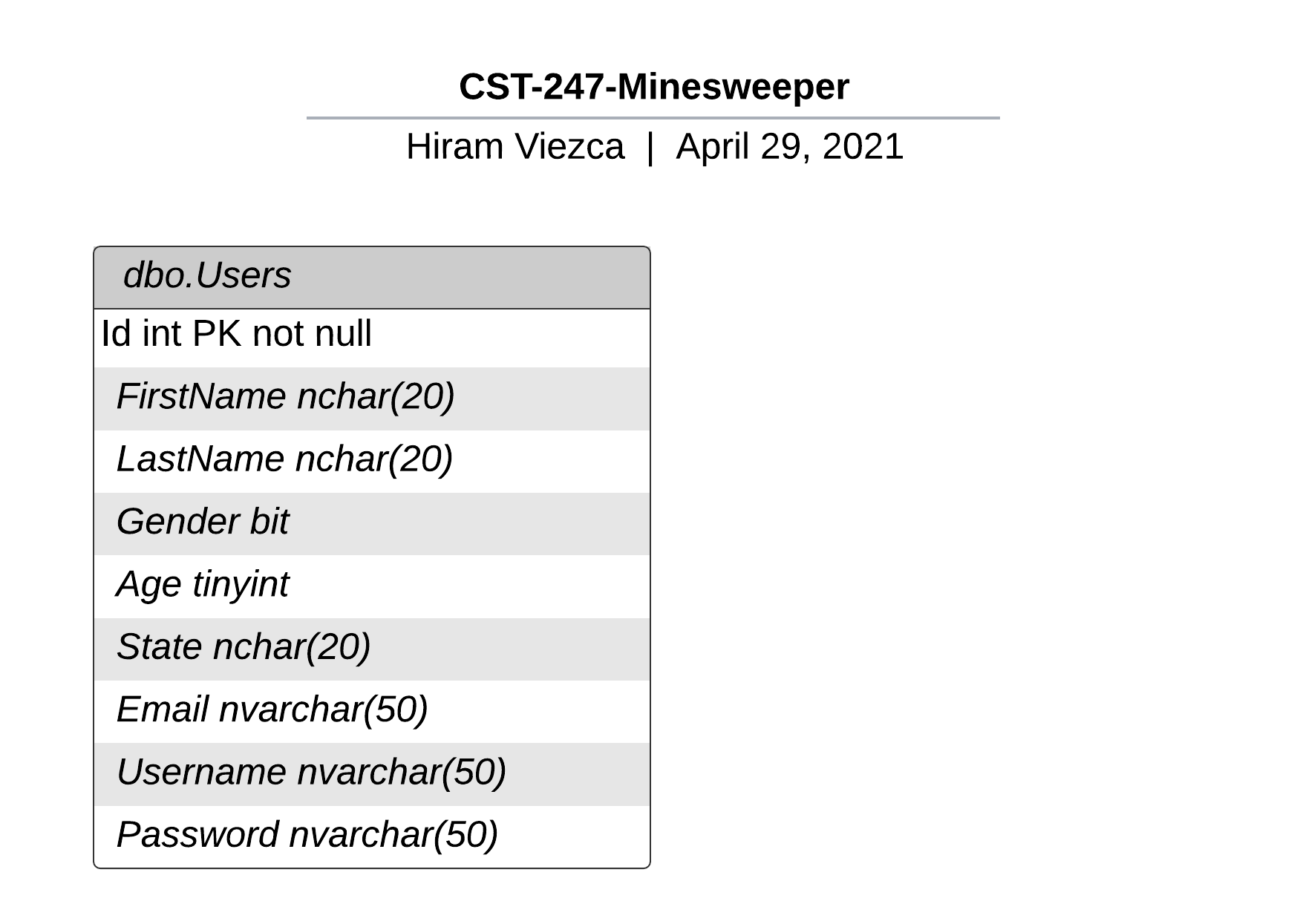
**General Technical Approach:**

***You should, in words, describe your approach and design here. You should also summarize any meeting notes, brainstorming sessions, etc. that you want to retain through the design of your project.***

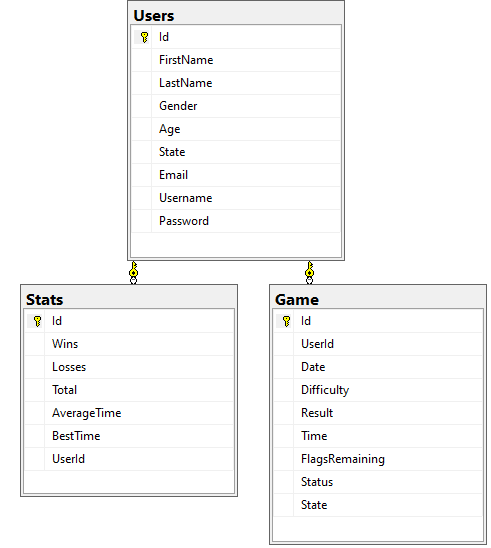
**Key Technical Design Decisions:**

*Any final technical design decisions, such as framework decisions and so forth., should be documented here. This should list the technology/framework, its purpose in the design, and why it was chosen.*

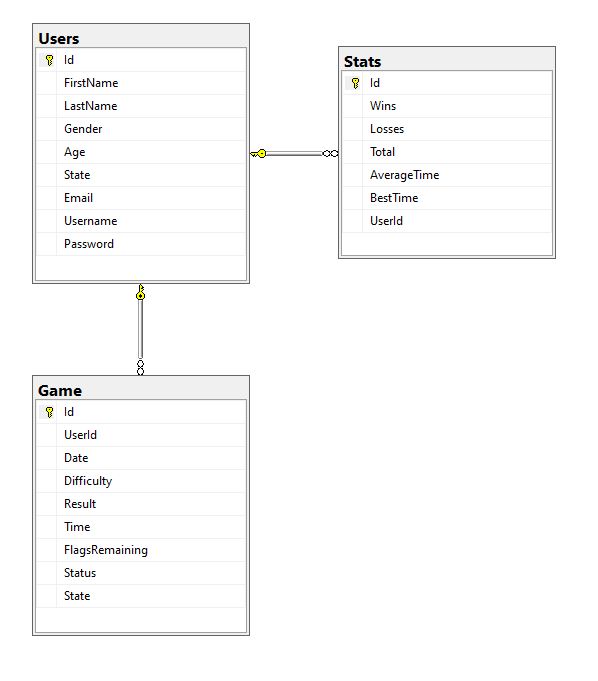
* Language: C#/.NET and ASP.NET (Chosen due to the dynamic web support abilities)
* Framework: ASP.NET MVC (Used for the resources in the framework)
* Design: MVC (Separate differences in services for more complex abilities)
* Layouts: Bootstrap (Import complex and useful visuals and layouts)
* Hosting: n/a (Cloud platform for hosting)
* Database: MS SQL Server (For the development and creation of the database)

**ER Diagram:***Insert the image file of your ER database diagram.*****

**CST-247 Minesweeper ER Diagram 5/5/2021**

****

**Updated Minesweeper ER Diagram 6/7/2021**

****

**Risks:**

There are currently no risks.

**DDL Scripts:**

*This should contain a link to where the DDL script can be downloaded.*

<https://github.com/DanielCender/CST-247-CLC/tree/main/SQL%20Tables>

**Sitemap Diagram:**

*Include the image file of your Sitemap diagram.*

**Security Design:**

*This section should outline the design for how authentication and authorization was supported. This section should also contain all of the roles and privileges that are supported by the design.*

**Third Party Interface Design:**

*This section should fully document any third party service interface APIs, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by a third-party developer to integrate with the service and API.*

**Routes:**

1. /api/game/save - A route that saves/pauses a game

Method: POST

Required Parameters: game ID (int), game time (int) in seconds, board state (serialized board class as blob), and player ID (int).

Returns: Success/Failure status code and message.

1. /api/game/{id} - To get current game state or results

Method: GET

Required Parameters: game ID (int).

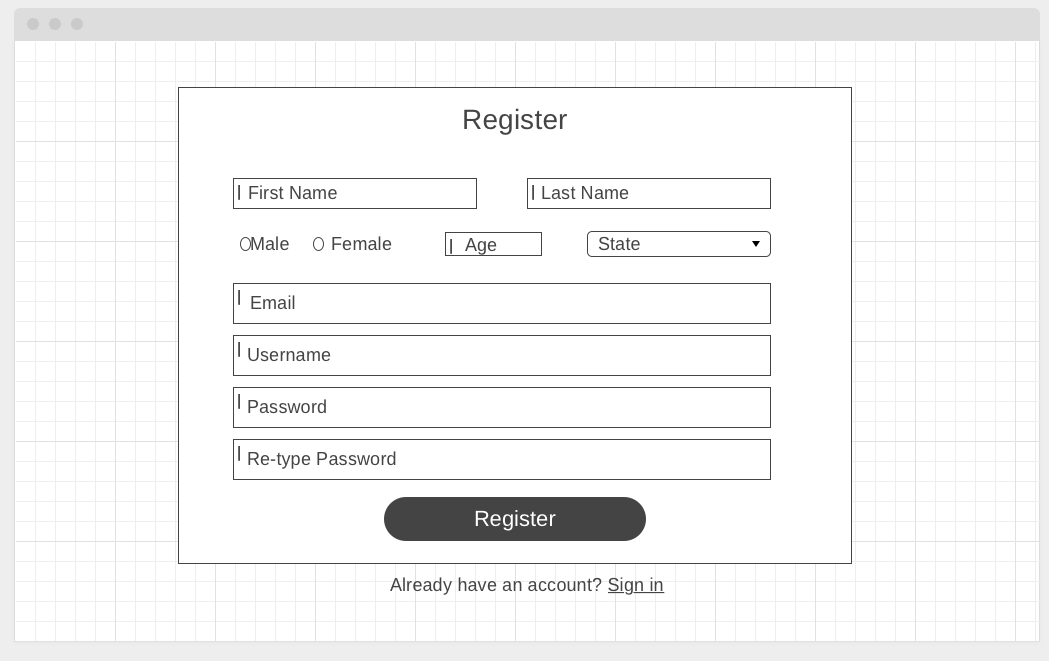
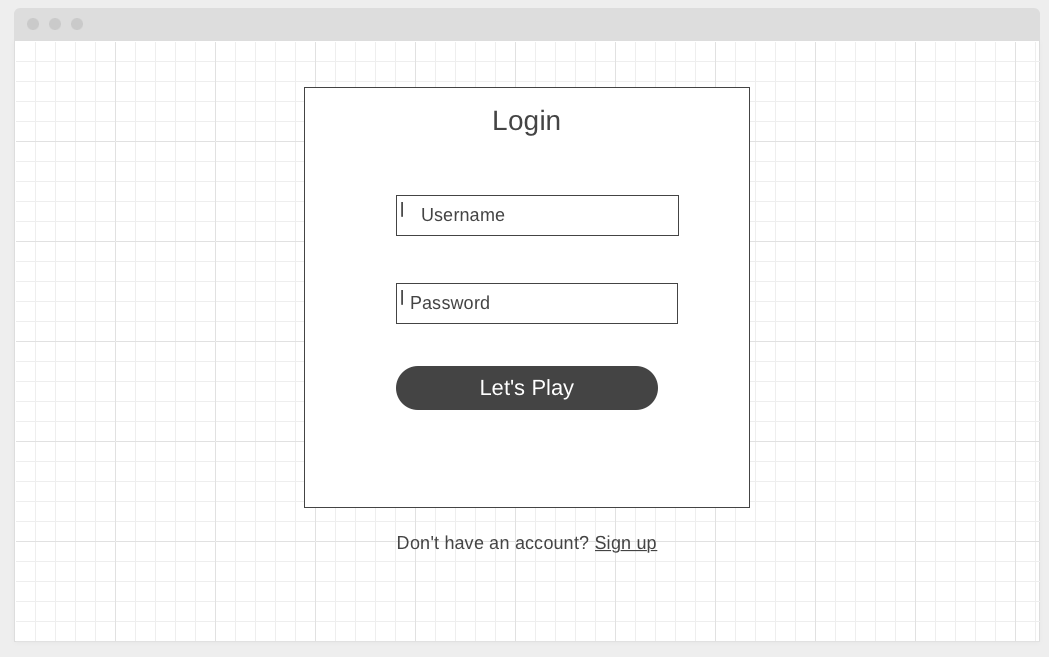
Returns: Response body containing the game board state, author info, and final/current score.

**Flowcharts:**

*Insert any flowcharts here. Flowcharts should document algorithms or workflow that will be implemented in your program.*

**User Interface Diagrams:**

*Insert any wireframe drawings or white board concepts that were developed to support your application. If you have no supporting documentation, please explain the rationale for leaving this section as N/A.*

****

**Class Diagrams:**

*Insert any class diagrams here. Your class diagrams should be drawn correctly with the three appropriate class compartments, + and – minus to indicate accessibility, and the data types for the state/properties, as well as method arguments and return types. If you have no supporting documentation, please explain the rationale for leaving this section as N/A.*

**Pseudo Code:**

*Provide Bitbucket URL references to any code stubs and pseudo code. If you have no supporting documentation, please explain the rationale for leaving this section as N/A.*

**Other Documentation:**

*Insert any additional drawings, storyboards, white board pictures, project schedules, tasks lists, etc. that support your approach, design, and project. If you have no supporting documentation, please explain the rationale for leaving this section as N/A.*

**Limitations / Future Features:**

**Limitations:** Currently, there is no ability to save and resume a game. In the future, we will implement a method of saving a user’s in-progress game and loading it back to the board when they next log in. It does not support “remember me”, stay signed in, or a forgot password flow for login. Registration could use a password verification strategy to avoid users inputting a password incorrectly the first time. No “Show” feature on passwords right now, part of user friendliness.

**Future Features:** We will implement statistical tracking of a user’s highest and average scores, along with all-time high scores among all users. Add a view where users can see their past game runs and how they match up to other similar players. Add a view showing a user’s accuracy metrics over time. Add more API routes for interacting with user metrics or user profile settings. Add a view for the users to update their own profile settings, or delete their account.