**Sudoku Application - Project Analysis**

**Rev. v1.1**

**Abel Tabor, Thomas Edwards, Brian Jefferson**

**CMSC 495**

**Section 7384**

**Group 5**

**4/5/2022**

**Professor Dao**

**Project Analysis**

Our group worked on this document to analyze our project of developing a desktop application that is capable of letting end users play a randomized game of sudoku and save and display previous users scores on a leaderboard.

Input Data & Sources -

| Input Data | Source |
| --- | --- |
| Player Menu Choice | End user selects from GUI between two options, New Game or Leaderboard |
| Player Name | End user input, prompted at the end of the puzzle. |
| Player Time/ Player Score | End user application instance. Occurs when user starts a puzzle and ends once user finish said puzzle |
| Entering puzzle variables | End user is prompted what number to input for missing aspects of the puzzle. Input is limited through a separate GUI pop-up with numpad prompt |
| Entering full puzzle | End user enters in complete puzzle from the GUI |

Output Data & Destination -

| Outputs | Destinations |
| --- | --- |
| Game Time / Game Score / Username | Displayed to the user through leaderboard GUI when prompted. |
| Game Time | Displayed while the user plays the game |
| Game Score | Displayed while the user plays the game |
| Sudoku Puzzle | Displays incomplete sudoku puzzle on main game UI. |
| Puzzle State | Displays if entered puzzle is correct or incorrect |

Data Processing Steps -

**Launched Application**

1. User is given three options: Start a new game, see leaderboards or exit.

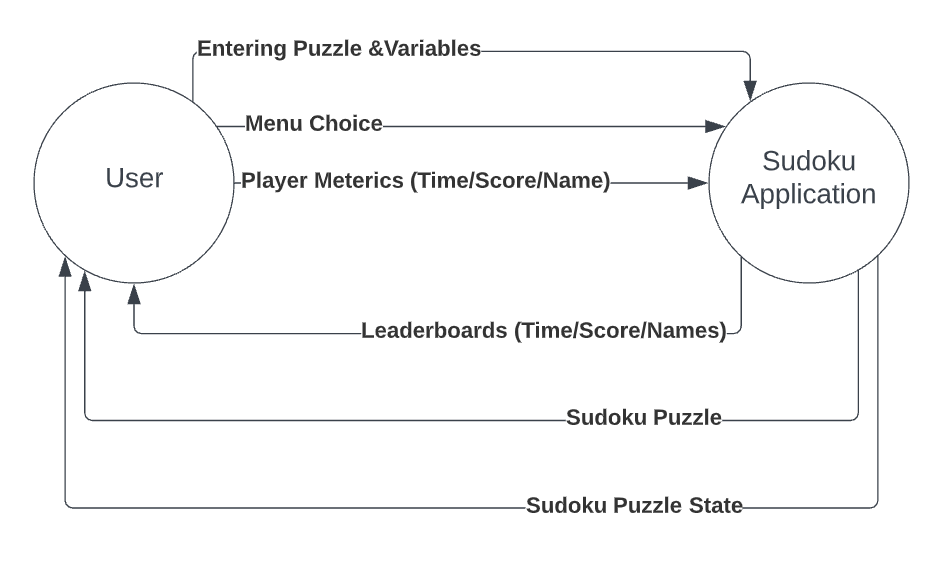
**New game**

1. Generate full sudoku board
   1. Using the generated board, select which variables to hide from the end user.
   2. Randomly select variables to remove and use a sudoku solver method to ensure that puzzle can still be solved
2. Display altered board on the main GUI
   1. Signals the start of the game and Game Time should being to count up from here
   2. User can fill in the board by selecting empty variables and selecting what number to place
3. User enters solved puzzle
   1. User enters solved puzzle by clicking on a button on the GUI, the application then takes in board state of user and compares it to the full board
      1. Everytime user attempts to enter a puzzle with mistakes, score decreases
   2. Game displays if entered puzzle is correct or incorrect
4. Entered user board matches with full sudoku board
   1. User is prompted for username for current session. Once a valid username has been entered the username, game score and game time are added to the database alongside a unique session ID.
   2. User is taken back to the start application screen

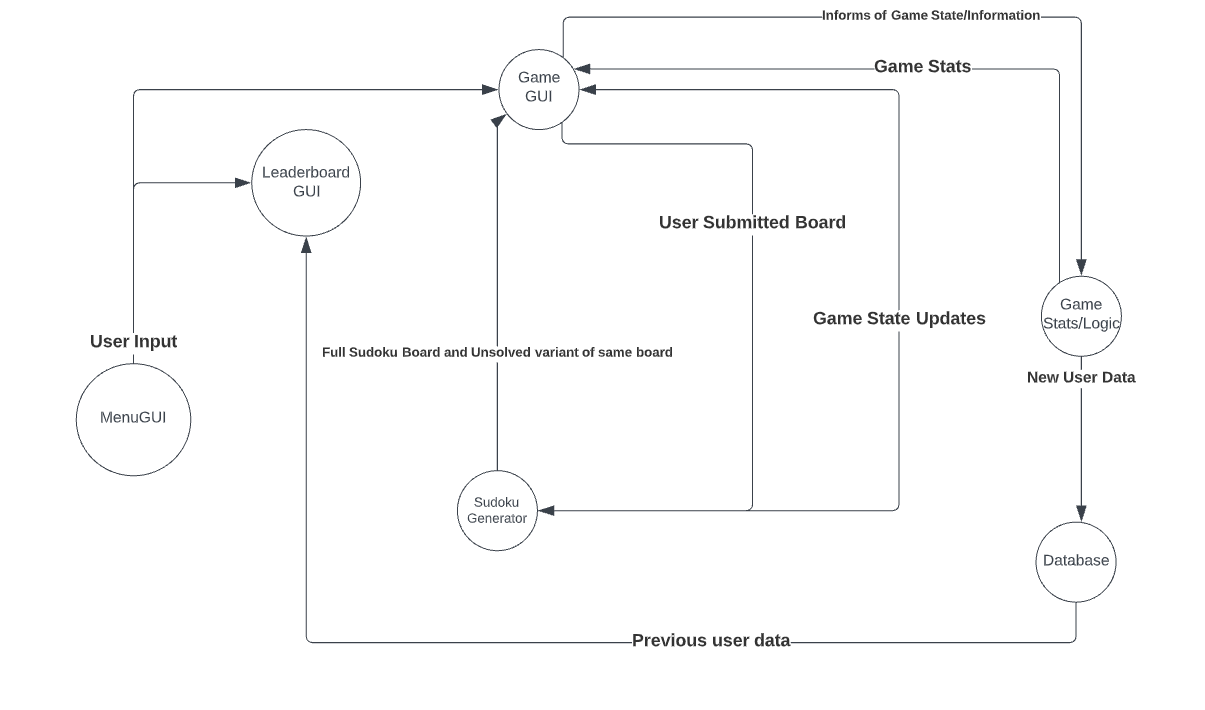
**Leaderboards**

1. Information is taken from the database and displayed in order of score
2. User prompted to go back to start application screen

Context Diagram -



Subsystem Diagram -



1. Sudoku Generator subsystem
   1. This takes an empty board and produces a sudoku board that adheres to the basic sudoku rules (1-9, no repeating numbers vertically, horizontally and within each square quadrant).
   2. The submitted user board is compared to the generated board and the game state is updated if they match
   3. Includes a Sudoku Solver. Randomly selected variables are chosen and a new board is created with those variables removed. The sudoku solver makes sure that this new board is capable of being solved
2. Game Stats/Logic
   1. The application informs this system when a new puzzle has started and when the user attempts to submit a new puzzle
   2. Updates game score and keeps track of the time passed on the current puzzle
      1. Application displays this information to the user
   3. Once the game state updates to solved, the application prompts the user for their name and the information is passed to the database
3. Database
   1. The database stores user data
      1. Information Stored
         1. Primary Key: Game ID -> Generated by querying total amount of entries and numbering them
         2. Game Score -> Received by Game Stats/Logic
         3. Game Time -> Received by Game Stats/Logic
         4. Username -> Received by Game Stats/Logic
   2. The leaderboard GUI for the application receives and displays information from the database
4. MenuGUI
   1. Allows for player to start a new game or see the leaderboards
5. Game GUI
   1. Displays necessary information for the user to interface with the puzzle
   2. Provides the user input for the application
6. Leaderboard GUI
   1. Displays necessary information from database to rank past players on their performance

Subsystem and Requirements -

| **ID** | **Subsystem** |
| --- | --- |
| 01 | Sudoku Generator |
| 02 | Sudoku Solver |
| 03 | Game GUI |
| 04 | Game stats/logic |
| 05 | Database |
| 06 | Leaderboard GUI |

Possible Enchantments -

Rather than every new puzzle score being treated as a new person, a log-in system with usernames and passwords can be used to allow users to review past puzzles.

Implement a difficulty option using the sudoku solver and the number of missing variables of any given puzzle

Risk & Risk Management -

Limit the amount of input that the user can input. Certain risks:

Username

* This is the most open type of user input acknowledged in the system. It is also inserted into the database so SQL injections are a risk. The input should be sanitized and maybe even limit what usernames can be.

Submitting the puzzle

* When users submit the puzzle, the score decreases by a certain amount. Users that spam submit puzzles might be capable of causing some issues if not handled properly.