**Sudoku Application - Project Plan**

**Rev. v1.0**

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

**CMSC 495**

**Section 7384**

**Group 5**

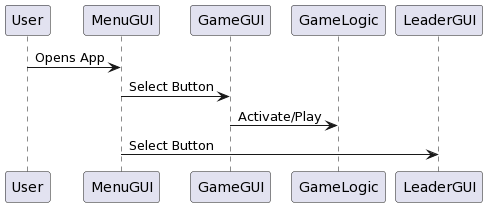
**4/10/2022**

**Professor Dao**

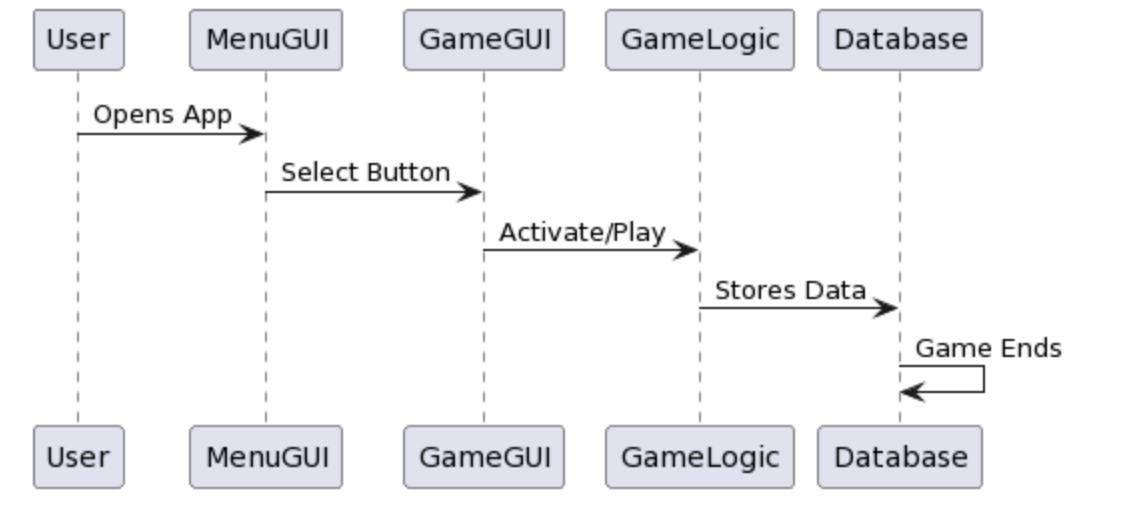
| **Name** | **Date** | **Description** |
| --- | --- | --- |
| Abel Tabor | 4/08/2022 | Started Document, Added Sudoku Generator |
| Brian Jefferson | 4/08/2022 | Added starting information for Start up/ Shutdown |
| Abel Tabor | 4/10/2022 | Added GameStats, MenuGUI, GameGUI, LeaderGUI. Started Database |
| Brian Jefferson | 4/11/2022 | Adding the additional scenarios based on program |
| Thomas Edwards | 4/11/2022 | Set up GitHub for software control |
| Thomas Edwards | 4/12/2022 | Comments to error handling. Added invalid input limitation to Risk mitigation. |
| Thomas Edwards | 4/14/2022 | Started outlining Project Test Plan & ICD in accordance with Project Plan & Project Requirements (Week 2). |

**Event Trace Diagrams:**

**Start Up Scenario**

****

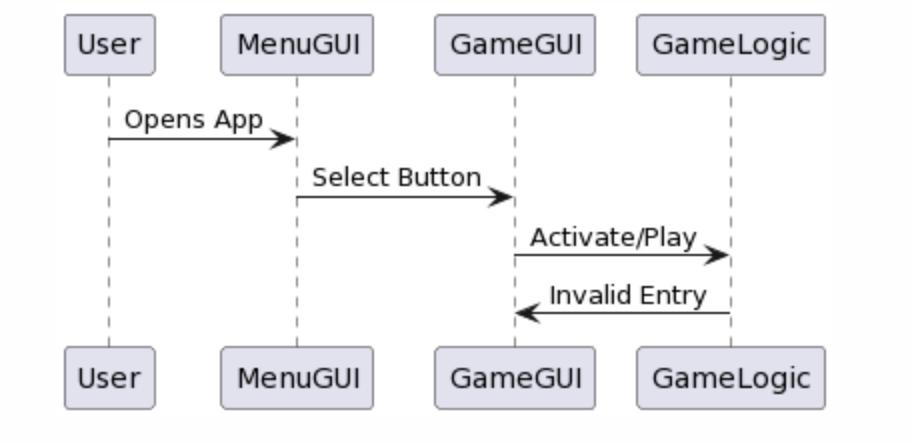
**Shut-Down Scenario:**

****

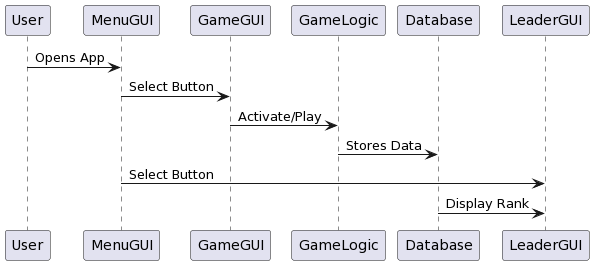
**Error Handling Scenarios:**

* The ability to input an invalid option shall be mitigated by restricting the option to buttons (Game UI class)

**Invalid Entry:**

****

**Scoreboard Screen:**

****

**Class Design**

1. Sudoku Generator

Class SudokuGenerator {

Private final int boardSize = 9;

Private int[][] fullBoard = new int[boardSize][boardSize] ;

Private int[][] gameBoard = new int[boardSize][boardSize];

Public SudokuGenerator() {

fill();

remove();

}

Public int[][] getFullBoard() {

Return this.fullBoard;

}

Public int[][] getGameBoard() {

Return this.gameBoard;

}

Private boolean colCheck (int col, int number) {

Checks if given number is unique within given col

}

Private boolean rowCheck (int row, int number) {

Checks if given number is unique within given row

}

Private boolean squareCheck (int row, int col, int number) {

Checks if given number is unique within 3x3 square determined by given row and col

}

private void fill() {

Generate random number and check using colCheck, rowCheck, and squareCheck and add to fullBoard if all true

}

Private void remove() {

Removes values from randomly selected grid points. Calls Sudoku Solver, if true writes new board to gameBoard

}

Private boolean isSolvable() {

Checks if gameBoard is solvable

}

}

1. Game Stats/Logic

Class GameLogic {

Private int sec, min, hour, score;

Private String name;

Database db;

Public GameLogic {

This.sec = 0;

This.min = 0;

This.hour = 0;

This.score = 5000;

}

Public int[] getTime() {

Returns time in an array

}

Public string getTimeString() {

Returns the time as a string

}

Public void incrTime() {

Increments the time by a second, used in conjunction with javax.swing.timer

}

Public void updateScore() {

Updates the score

}

Public void enterDB() {

Uses the database class to enter this objects fields to the database

}

}

1. MenuGUI

Class MenuGUI extends JFrame {

Private JPanel jpanel;;

Private Jbuttons jbuttons;

Public MenuGUI {

Constructs and displays the menu gui

}

}

1. GameGUI

Class GameGUI extends JFrame {

Private Jpanel jpanel;

Private JButtons jbuttons;

Private GameLogic

Public GameGUI {

Constructs and displays the game gui

}

}

1. Database

Class Database {

Public boolean enterDatabase(String name, String time, int score) {

Create connection to database and enters in given information. If given information was entered without any problems, return true else return false then close connection to database

}

Public String[] getData() {

Creates connection to database and returns all entries. Closes connection to database

}

}

1. Leaderboard GUI

Class LeaderGUI extends JFrame {

Private Jpanel jpanel;

Private JButtons jbuttons;

Private database db;

Public LeaderGUI() {

Constructs and displays the game gui and database connection

}

}

**Unresolved risks and risk mitigation:**

Previous Risks identified:

* Invalid entries on game UI

**Risk Mitigation:**

* Utilize buttons to restrict invalid entries (can be implemented into game UI class)

**Username submission**:

* Utilize regex expression to sanitize user input
* When submitting username to database, utilize parameterized queries

**Puzzle submission:**

* Limit reduction of score below 0

