Table of Content

1. Project Specification

1.1. Project description

For my final project, I decided to recreate the popular puzzle game ‘Sudoku’. Wanting a method of playing a puzzle game for my free time, I decided to make a sudoku puzzle game for me to play with.

The game consists of a 9 by 9 grid of numbers that must follow 3 main rules;

1. No numbers may not repeat in a row or column
2. When split into multiples 3 by 3 grid, there should also be no repeated numbers
3. The numbers in the grid must be between 1-9 inclusive

The grid would have some missing numbers which would be filled by the player. If the completed grid follows all 3 rules the player would win. To add a sense of competitiveness, I decided to add a leaderboard.

1.2. Project Link

The GitHub Repository of the project can be accessed through the link provided below:

GitHub Repository

1.3.Essential Algorithm

1. Sudoku grid value manipulation
   The grid value manipulation falls under 2 categories, insert and delete. Both categories will ask the player to input a coordinate to be used to specify the exact location of the input and if the player wishes to place a number onto the sudoku, it will also ask for the number. If it detects an error in input, or if the Cell Lock Algorithm sees the coordinate is locked, it will stop the algorithm from producing an output.
2. Cell lock algorithm
   To prevent players from altering the sudoku numbers that are meant as clues to the puzzle, an algorithm is created to store coordinates of these values. When a player attempts to alter the sudoku, it checks if the coordinate specified matches with the stored values and rejects the input if it is. In addition, a method to clear the lock storage is made to reset the puzzle for each attempt.
3. Check if sudoku complete
   To create a win condition for the game a check needs to be created to ensure that the user has fulfilled the win condition, that being filling all the empty boxes while following the sudoku rule. The check is done by first checking the first row then the column. The function will also create a temporary array for the 3x3 box check. If it checks that the grid

follows the rules with no empty box, it will output True, signifying the completion of the game.

4. Create random sudoku
   The sudoku games lack any stored sudoku questions. It instead creates a random complete sudoku. This is done using the random module to pick a random number to assign the grid. This is done while still ensuring the rules of the game are not broken.
5. Timer algorithm
   To keep track of how long it took for the user to complete the sudoku puzzle a timer is created to keep track. This is done by having 2 variables being the start time and the end time. The start time stores the seconds after the sudoku is made, while the end time stores the seconds after the sudoku is completed. To calculate the time taken the end time is subtracted with the start time showing the time taken by the user to complete the sudoku puzzle.
6. CSV manipulation
   To store the time taken by each player, a CSV file is used to store all the data about it. The CSV manipulation algorithm is simply an algorithm which inserts, updates and deletes Username data and the time taken. In addition, the algorithm removes empty rows by rewriting the entire CSV file ensuring any empty row would only be on the final line.
7. Game state / Cache check
   To prevent users from seeing UI from another page and from interacting with the wrong UI, a variable is stored which tells the program which page the user is on. Based on what is on the variable, the game will display the appropriate UI.
8. Keypress algorithm
   The keypress algorithm uses the pygame module to check which key is pressed. This is done to check if the user inputted a specific key such as the escape key, which will close the program.
9. Textbox algorithm
   The textbox algorithm expands on the keypress algorithm and uses it to store text input from the user. The user can interact with a textbox by pressing on it, which will cause the text box to darken, the user's text input will then be stored on to a variable which is displayed on the textbox.
10. Button Algorithm
    The button algorithm uses the rect function of the pygame to create a collision check for the mouse pointer. If the user presses the left mouse button while the pointer is on the collision, it will output True for an if check, which will fulfil a specific function specified on the button.

1.4. Modules

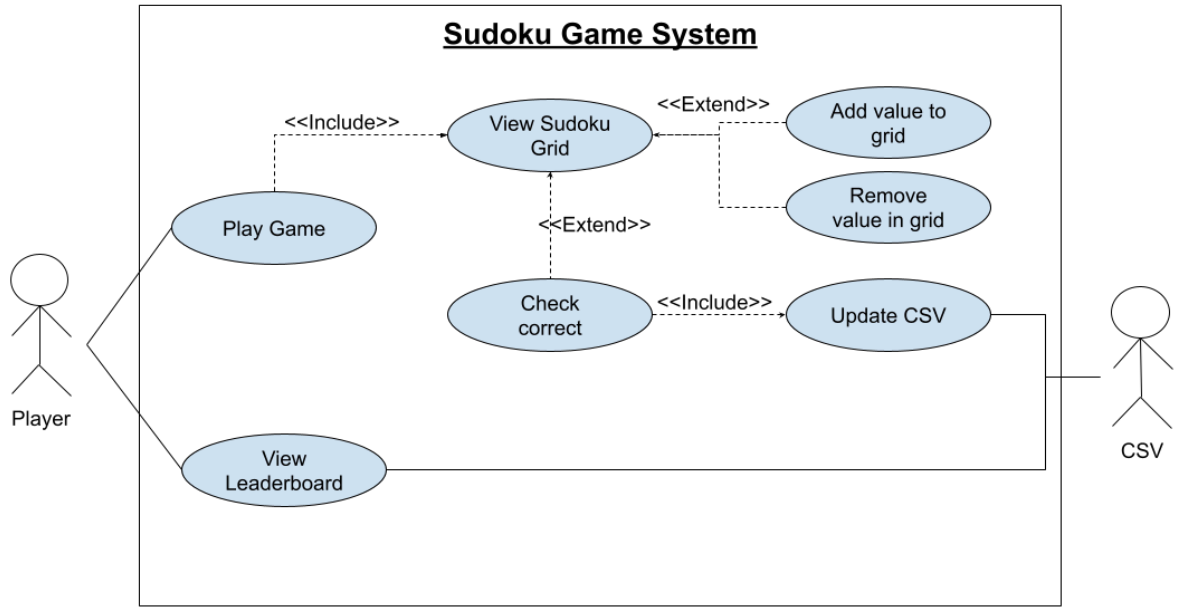
1. PyGame
   The Pygame module is the foundation of the game, in charge of the display of all UI elements in the game. Pygame allows the display of GUI through blitting images onto the

screen as well as using the mixer to play the background music. In addition, PyGame is used to allow user input thanks to its ability to check for keyboard presses.

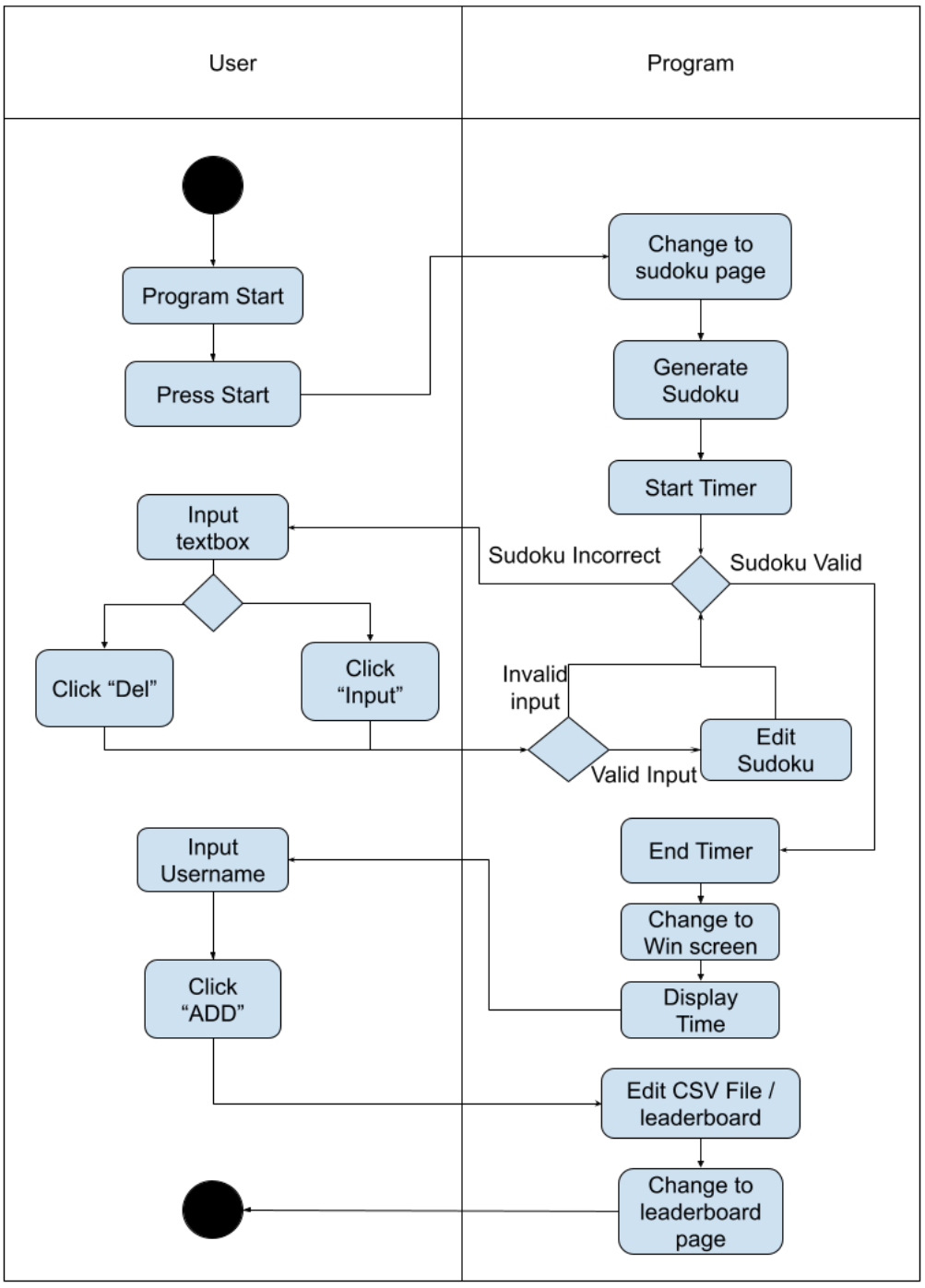
2. Pygame.freetype
   A sub-library of the PyGame. This module is incharge of the rendering of text. Allowing text to be blit onto the screen. Without it, the text label would need to be a picture file, which may consume more space.
3. CSV
   The CSV module serves as the backbone of the leaderboard system, Allowing the storing and editing of data into a text file to preserve data between each play session. Without the CSV module, the data stored would only persist until the program is closed.
4. OS
   The OS module is used to obtain the file path of the individual asset in the asset folder to be used in the program. It is called specifically to help the creation of a dictionary which contains the file path for the individual number images used in the grid.

2. Solution design

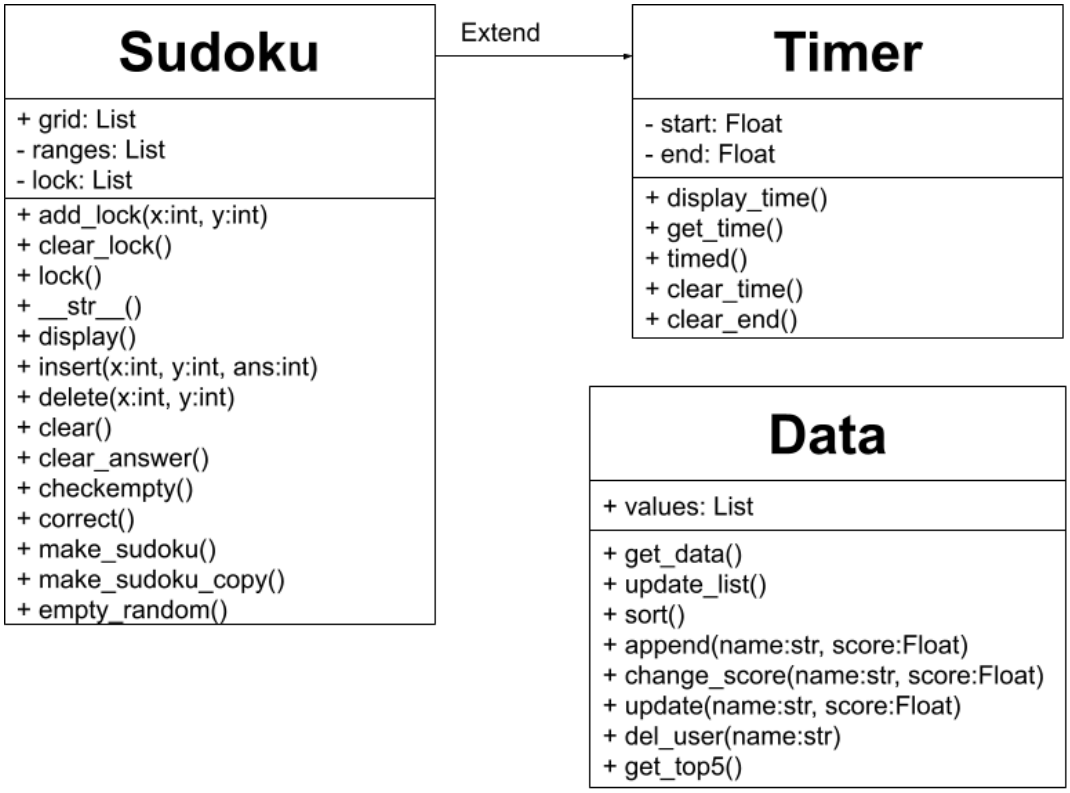
2.1. Use Case Diagram



2.2. Activity Diagram



2.3. Class Diagram

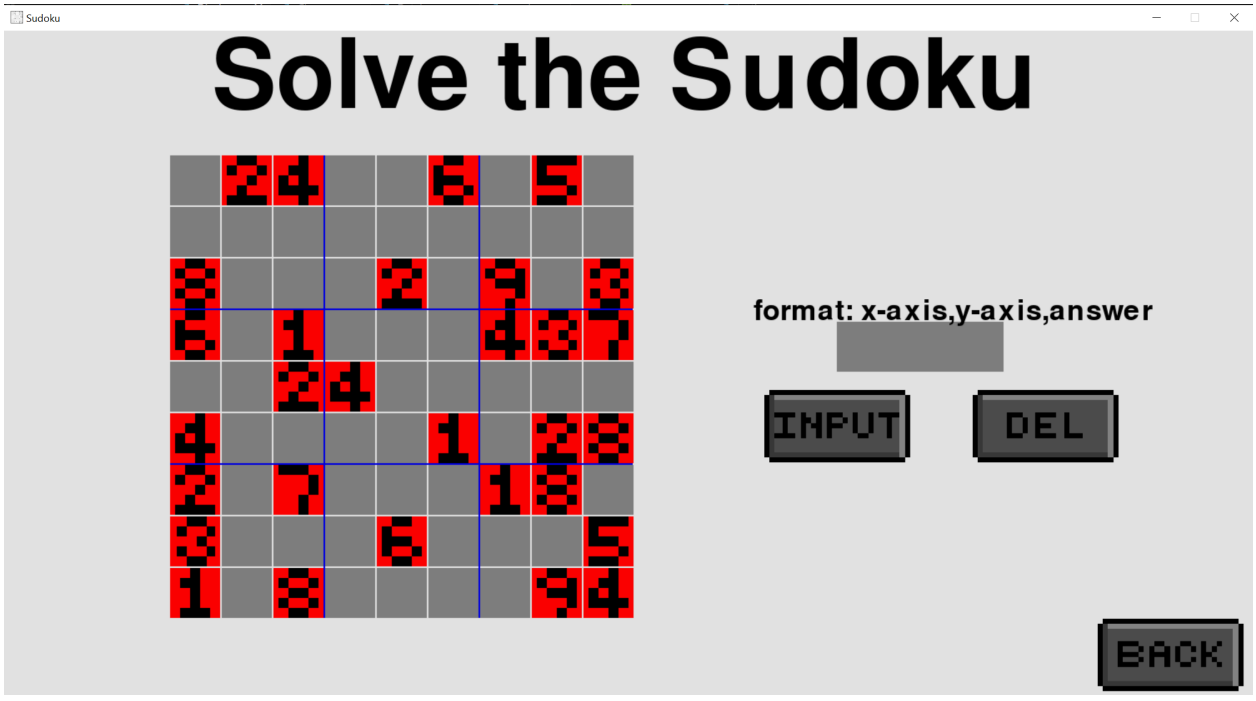


3. Documentation

3.1. Screenshot

Title Screen:

Game screen:



Win Screen:

Leaderboard Screen:

3.2. Video Demo

I have uploaded the video demo in my Google Drive. The video can be accessed through the link provided below:
https://binusianorg-my.sharepoint.com/personal/andres_winson_binus_ac_id/_layouts/15/guestaccess.aspx?share=EQaBLb9EslhLIVX1iERN5fcBmNrRvVFkSENNo3tixduysw&e=shcPQa

4. Evaluation and Reflection

4.1. Lesson Learnt

During the development of the sudoku game, there were many errors due to the sequencing of the program. This caused unintended output. To fix this, the sequencing was altered to ensure that the important task is done first before the other task is done. This issue taught me the importance of sequencing code to ensure certain tasks are done first before the other.

Additionally, this also serves as a method for me to implement the different programming skills taught to me during the semester. This was my first game in Python. While I have tried playing around with the language, I had never tried fully learning it, thus it was quite a novel experience.

To add on, I also find that while the program works as intended, it does not maximize time efficiency which may cause some performance issues especially with the timer being a few milliseconds off. The method in which I developed the program was to segment it into smaller issues to solve, which have helped me program in segments. But this also showed me the drawback of my programming method.

All in all, There were not any major hurdles during the development of the sudoku game, I was able to complete it before the deadline and was able to have time to focus on my other project.

4.2. Future Improvement

The sudoku game is made to be a small game that can be easily played. But there are some issues with the input system. While developing the input system, I had the idea of making a system which uses the mouse pointer to click the coordinate of the input. I had tried to implement it but faced many issues. to ensure the program was finished within schedule, I decided to put everything in a textbox and used a verification to check the coordinate.

Additionally, the leaderboard data was meant to be stored in a database. I had some education on databases and was hoping to use it to store data. But after some discussion with a senior, I was advised to use a CSV file instead. It was recommended as it was a topic taught before the deadline, thus there was much reliable information about it from not just the lecturer, but also from other students.

Another aspect about the program I also regretted was forgetting to remove the back button check in the main menu. The back button sends the user back to the main menu, but users are still able to click a non-existent button to go from the main menu to the main menu. While this is a minor error which won't be noticed by the user, it can lead to larger bugs when handling a larger game.

In addition, During development, I made a vow to rey to code everything without copying from external sources. While it's difficult to make a unique program to a game that has been copied multiple times, I add a self imposed rule to only look at example code from tutorials to base my code on. But for the creation of the "make sudoku" function, I was not able to create a

solution that is fast enough to not be noticed by the user. My code was highly inefficient, thus I decided to follow a solution proposed by someone else. Making it the instance where I copy pasted a code.

Furthermore, many iterations of the sudoku game have the ability to write notes on the grid. The program I developed does not have this functionality. This is due to some time constraint with the deadline.

To conclude, while the program worked as intended, there are still some improvements to be made. But even then, I am still satisfied with my final result. Even though it's a well known game that has been made countless times, I was able to make my own.