Name	**UC-1: Scan and import image**
Summary	An image file will be converted to a game file and loaded into the game GUI.
Rationale	When the user opens the program, they will need to start a game. They only have three options. They can either import a new game image, load a saved game file, or load a previous original game file.
Users	All users
Preconditions	The program has been launched.
Basic course of events	1. The user clicks the "import game image" button. 2. The program opens up a file open dialog. 3. The user selects an image file to import and clicks open. 4. The program scans the image in and creates a game file. 5. The program loads the game file into the GUI. 6. The program allows the user to start playing the game.
Alternative paths	1. In Step 3, the program will save any active game and clear it before importing the next game file. 2. In Step 3, if the program cannot import the image, an error is shown to the user. The screen will still be cleared and an empty GUI will be presented.
Postconditions	The game image is imported into the GUI for user gameplay.

Name	**UC-2: Load game file**
Summary	The user picks a game file (saved or original) to load into the game GUI.
Rationale	At any time, the user may want to load another game file. They will have the option of loading any previously imported original game file or any saved game file.
Users	All users
Preconditions	The program has been launched.
Basic course of events	1. The user clicks the "load game" button. 2. The program opens up a file open dialog. 3. The user selects a game file to load and clicks open. 4. The program loads the game file into the GUI. 5. The program allows the user to start playing the game.
Alternative paths	In Step 3, the program will save any active game and clear it before loading the next game file. In Step 3, if the program cannot import the game file, an error is shown to the user. The screen will still be cleared and an empty GUI will be presented.
Postconditions	The game file is loaded into the GUI for user gameplay.

Name	**UC-3: Save game to file**
Summary	The currently active game in the GUI is output to a saved game file.
Rationale	The user may want to exit the game and save its current state to load later.
Users	All users
Preconditions	The program is launched and an active game is being played.
Basic course of events	1. The user clicks the "save game" button. 2. The program creates a save game text file. 3. The program displays the location of the saved game file and its name.
Alternative paths	None
Postconditions	The game save file is created and the GUI returns to normal operation.

Name	**UC-4: Input number**
Summary	The user inputs a number and the game GUI responds accordingly.
Rationale	The user has to enter numbers into the GUI to solve the Sudoku puzzle.
Users	All users
Preconditions	The program is launched and an active game is being played.
Basic course of events	1. The user clicks an active number box. That is a box that does not contain an original number. 2. The program ask the user to input a number. 3. The user inputs a number. 4. The program checks if the number is valid. That is, the number does not conflict with any other number in its row, column, or 3x3 box. 5. The number is input into the box.
Alternative paths	In Step 3, if the input is not a valid number (1-9), an error is shown to the user. The move is not made and the GUI returns to normal operation. In Step 4, if the input conflicts with another location, an error is shown that displays the first conflicting location that is found. The GUI then returns to normal operation.
Postconditions	The number is input to the box. The GUI returns to normal operation.

Name	**UC-5: Show available moves**
Summary	The user picks a box, and the GUI responds with available moves for the box.
Rationale	The user may want to see what numbers will currently work in any given box.
Users	All users
Preconditions	The program is launched and an active game is being played.
Basic course of events	1. The user clicks the “Show available moves” button. 2. The program asks the user which box they would like to check. 3. The user inputs the box they would like to check. 4. The program responds with all numbers that do not conflict with the box’s row, column, or 3x3 box.
Alternative paths	None
Postconditions	The valid moves are displayed to the user. The GUI returns to normal operation.