Product Requirements

Team Elita 5-1

Brief problem statement

We are developing a Sudoku puzzle solver that solves Sudoku puzzles of varying sizes and difficulty levels (9x9 and 16x16), providing users with a convenient and reliable tool to solve Sudoku puzzles. The code should accurately solve puzzles and handle input errors gracefully. Our goal is to create a user-friendly and efficient Sudoku puzzle solver that appeals to both casual users and avid Sudoku enthusiasts, ultimately providing a satisfying and enjoyable puzzle-solving experience.

System requirements

We will be working entirely with the C-programming language in order to develop our code. Below are the libraries we need to use:

#include <math.h>

#include <ctype.h>

#include <string.h>

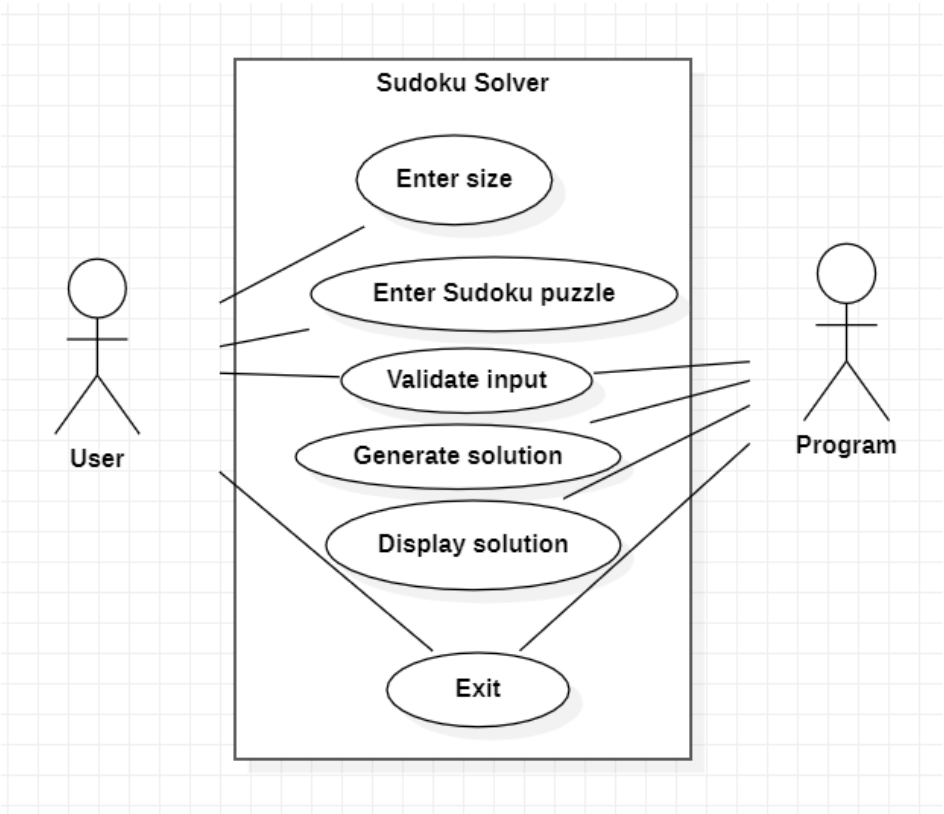
Users profile

Our program is a simple code, meaning the scope of the users is broad and not specific to a certain demographic. The user guide clearly explains to the user how to run the code in Code::Blocks or in XCode, meaning that the users do not need to be that familiar with coding or programming.

Feature requirements (user stories)

No.	User Story Name	Description	Release
1.	Enter option	The user can choose the size of the puzzle they want to solve: 9x9 or 16x16.	R1
2.	Enter puzzle	The user should be able to input the puzzle as a matrix of numbers where 0s represent empty spaces.	R1
3.	Validate input	After the program validates the puzzle, the user will be given the chance to verify they entered the right numbers before the solving starts.	R1
4.	Generate solution	The program should use an efficient method to solve the puzzles.	R1
5.	Display solution	The user can see the solved puzzle displayed in a clear manner.	R1
6.	Exit	The user should be able to exit the program.	R1

Use case diagram



Use case description

Use Case Number:	*US-01*
Use Case Name:	*Enter size*
Overview:	*The user picks between two sizes: 9x9 and 16x16*
Actor(s):	*User*
Pre condition(s):	*No preconditions*
Scenario Flow:	*Main (success) Flow:* *1-User enters an option.*
	Alternate Flows: *1-User exits the program*
Post Condition:	*The size is entered.*

Use Case Number:	*US-02*
Use Case Name:	*Enter Sudoku Puzzle*
Overview:	*The user enters the puzzle as a matrix of space separated numbers*
Actor(s):	*User*
Pre condition(s):	*The user must have picked the size of the puzzle first.*
Scenario Flow:	*Main (success) Flow:* *1-User enters each number and hits enter.*
	Alternate Flows: *1-User exits the program with the 'e' key.*
Post Condition:	*The puzzle is stored in a matrix.*

Use Case Number:	*US-03*
Use Case Name:	*Validate input*
Overview:	*The program validates the input for any out of bound characters entered or logical errors, then the user validates(verifies) the input again.*
Actor(s):	*Program*
Pre condition(s):	*The user must have entered the puzzle.*
Scenario Flow:	*Main (success) Flow:* *1-The input gets validated by the program.* *2-The input gets validated by the user.*
	Alternate Flows: *1-The code does not get validated by the program. The user can reenter the puzzle or exit.* *2-The code gets validated by the program but not by the user. The user can reenter the puzzle or exit.*
Post Condition:	*A valid puzzle is stored in the matrix.*

Use Case Number:	*US-04*
Use Case Name:	*Generate solution*
Overview:	*The program generates a solution to the puzzle.*
Actor(s):	*Program*
Pre condition(s):	*A valid puzzle must be stored in the matrix.*
Scenario Flow:	*Main (success) Flow:* *1-The program must generate a solution to the entered puzzle and store it in the matrix.*
	Alternate Flows: *1-No solution can be generated, a message is printed to the user.*
Post Condition:	*A solved puzzle is stored in the matrix.*

Use Case Number:	*US-05*
Use Case Name:	*Print puzzle*
Overview:	*The puzzle gets printed in a clear manner.*
Actor(s):	*Program*
Pre condition(s):	*The puzzle is solved*
Scenario Flow:	*Main (success) Flow:* *1-The puzzle gets printed.*
	Alternate Flows: *1-No alternate flows at this point in the program.*
Post Condition:	*The puzzle is displayed.*

Use Case Number:	*US-06*
Use Case Name:	*Exit*
Overview:	*The program ends.*
Actor(s):	*User,Program*
Pre condition(s):	*No preconditions*
Scenario Flow:	*Main (success) Flow:* *1-After all the previous steps, the program ends successfully.*
	Alternate Flows: *1-User exits the program during the flow of previous cases.*
Post Condition:	*Program stops executing.*