

SOFTWARE REQUIREMENTS SPECIFICATIONS

SUDOKU AUTO-SOLVER

B Chaitanya Rajesh	140050073
Chitraang Mardia	140050023
K V N Sreenivasulu	140050078

Slot: 11

Group: 11

TABLE OF CONTENTS:

1.	INTRODUCTION	4
1.1.	PURPOSE	4
1.2.	SCOPE OF PROJECT	4
1.3.	GLOSSARY	5
1.4.	REFERENCES	5
1.5.	OVERVIEW	6
2.	OVERALL DESCRIPTION	7
2.1.	SYSTEM ENVIRONMENT	7
2.2.	FUNCTIONAL REQUIREMENTS SPECIFICATIONS	7
2.2.1.	Use case: Main	7
2.2.2.	Use case: Normal sudoku	8
2.2.3.	Use case: Diagonal sudoku	9
2.2.4.	Use case: Window sudoku	10
2.2.5.	Use case: Jigsaw sudoku	11
2.2.6.	Use case: Solve sudoku	12
2.2.7.	Use case: Show a tile	12
2.2.8.	Use case: Check solution	13
2.2.9.	Use case: Exit	14
2.3.	USER CHARACTERISTICS	14
2.4.	NON-FUNCTIONAL REQUIREMENTS	14
3.	REQUIREMENT SPECIFICATION	15
3.1.	EXTERNAL INTERFACE REQUIREMENTS	15
3.2.	FUNCTIONAL REQUIREMENTS	15
3.2.1.	Main	15
3.2.2.	Normal sudoku	16
3.2.3.	Diagonal sudoku	16
3.2.4.	Window sudoku	17
3.2.5.	Jigsaw sudoku	18
3.2.6.	Solve sudoku	19

3.2.7.	Show a tile	19
3.2.8.	Check solution	19
3.2.9.	Exit	20
3.2.10.	SolveSudoku	20
3.2.11.	CheckNum	21
3.2.12.	Input	21
3.2.13.	ValidBox	22
3.2.14.	DisplaySudoku	22
3.2.15.	CheckSudoku	23
3.2.16.	Check	23
3.2.17.	Valid	24
3.2.18.	Sudoku	24
3.3.	DETAILED NON-FUNCTIONAL REQUIREMENTS	25
3.3.1.	Logical structure of data	25
	Index	27

1 INTRODUCTION

1.1 PURPOSE

The Purpose of the project is to design a program that can solve a normal Sudoku. It can also solve variants like diagonal sudoku, window sudoku and jigsaw sudoku of any degree of toughness as desired by the user.

1.2 SCOPE OF THE PROJECT

It has a scope of solving the Sudoku given by the user of any degree of toughness. It can also solve various variants of Sudoku that are diagonal Sudoku, window Sudoku (hypersudoku) and Irregular Sudoku (nonomino). It can also display a single tile or check a solution as instructed by the user.

1.3 GLOSSARY

Term	Definition
Sudoku	A logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid contains all of the digits from 1 to 9.
Box	A 3×3 sub-grid that composes the main grid. Each digit must appear once in a box.
Tile	A single element of the 9×9 grid. It contains only one digit.
Diagonal sudoku	A variant of Sudoku in which the numbers 1 to 9 must appear once in the principal diagonals as well.
Window sudoku (Hypersudoku)	A variant of Sudoku with 4 additional boxes defined in which the numbers 1 to 9 must appear once. These four new areas are overlapping with the nine boxes of a Sudoku.
Jigsaw Sudoku (Nonomino)	A variant of Sudoku in which the boxes are not squares but irregular connected regions of nine tiles .
Input	An entrance or change that is given to a system and which activate or modify a process.
User	Person handling the program.
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate.

1.4 REFERENCES

- CS101
 - <http://www.cse.iitb.ac.in/~cs101/project.html>
 - http://www.cse.iitb.ac.in/~cs101/Project/Manual_Code::Blocks_Simplecpp.pdf
 - Past year projects
- wikipedia
 - <http://en.wikipedia.org/wiki/Sudoku>

- http://en.wikipedia.org/wiki/Sudoku_solving_algorithms
- <http://en.wikipedia.org/wiki/Backtracking>
- <http://www.geeksforgeeks.org/backtracking-set-7-sudoku/>
- norvig.com/sudoku.html
- www.cse.msu.edu/~chengb/RE-491/Papers/SRSEExample-webapp.doc

1.5 OVERVIEW OF DOCUMENT

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

2 OVERALL DESCRIPTION

2.1 SYSTEM ENVIRONMENT

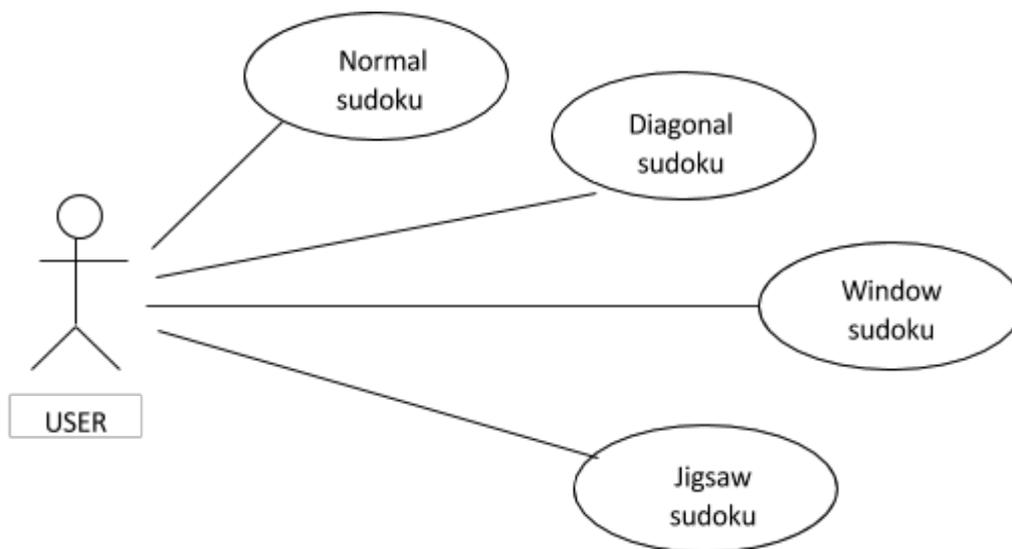
The program can only accommodate a single user. The user can access the program by running the appropriate file on his computer.

2.2 FUNCTIONAL REQUIREMENTS SPECIFICATIONS

This section outlines the use cases for the user.

2.2.1 Use case : main

Diagram



Brief Description

The user is initially asked what type of sudoku does he wish to choose.

Step-by-step Description

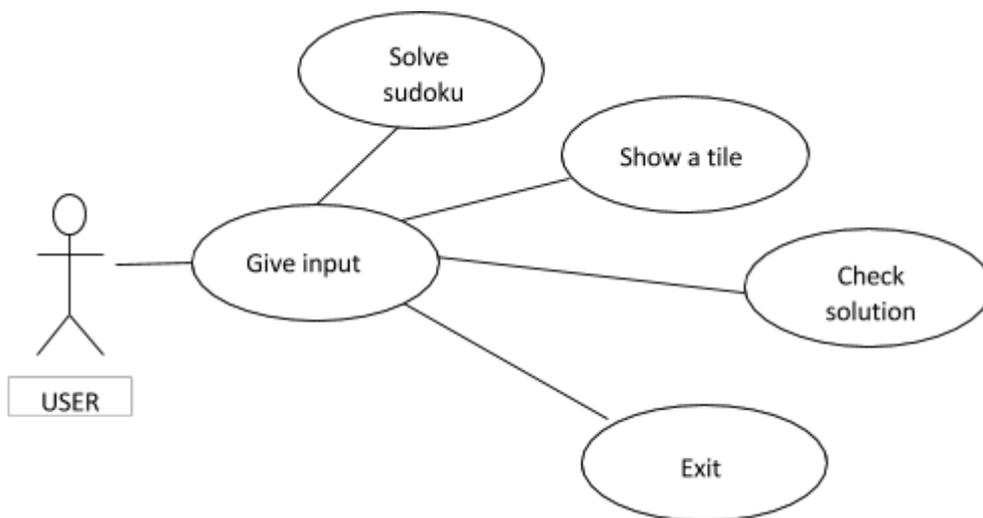
1. The user runs the program.
2. The user is asked him what type of sudoku does he wish to choose.

3. He is given the options normal sudoku, diagonal sudoku, window sudoku and jigsaw sudoku.
4. User chooses appropriate option.

XREF: sec. 3.2.1

2.2.2 Use case : Normal sudoku

Diagram



Brief Description

The user is asked to give input. Then he chooses what he wants to do next.

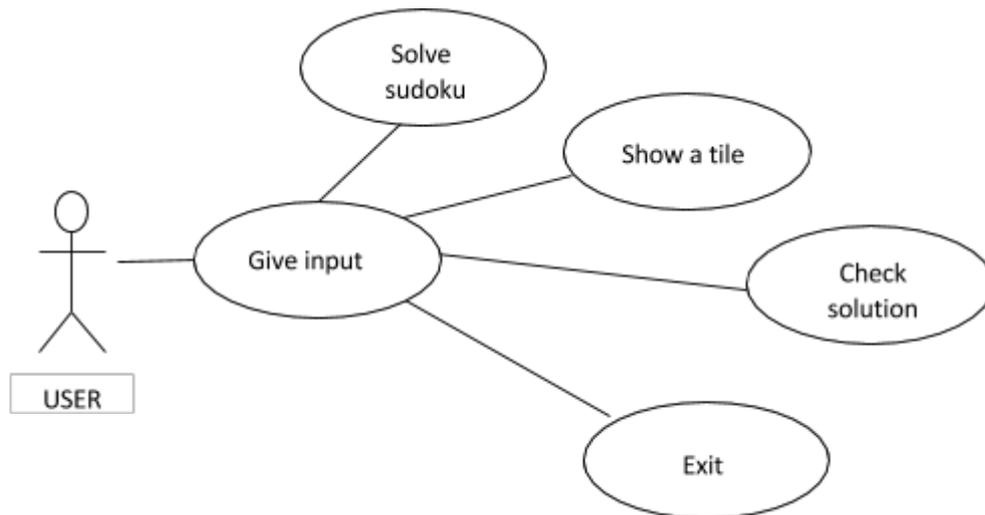
Step-by-step Description

1. The user is asked to input the sudoku.
2. The user inputs the sudoku.
3. The program checks the given input, if the given input is invalid appropriate error message is displayed and program exits.
4. Then the user is asked what does he wish to do next.
5. He is given the options solve sudoku, show a tile, check solution and exit.
6. User chooses appropriate option.

XREF: sec. 3.2.2

2.2.3 Use case : Diagonal sudoku

Diagram



Brief Description

The user is asked to give input. Then he chooses what he wants to do next.

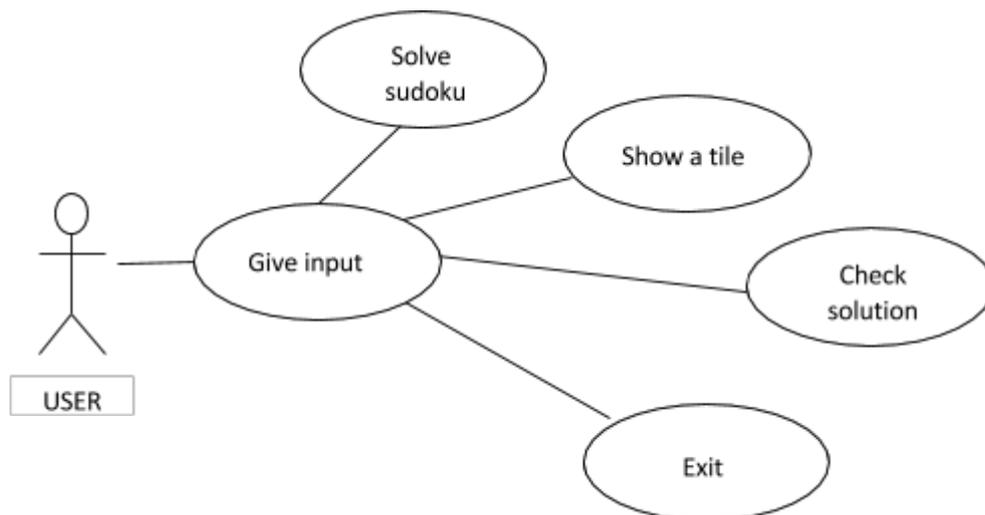
Step-by-step Description

1. The user is asked to input the sudoku.
2. The user inputs the sudoku.
3. The program checks the given input, if the given input is invalid appropriate error message is displayed and program exits.
4. Then the user is asked what does he wish to do next.
5. He is given the options solve sudoku, show a tile, check solution and exit.
6. User chooses appropriate option.

XREF: sec. 3.2.3

2.2.4 Use case : Window sudoku

Diagram



Brief Description

The user is asked to give input. Then he chooses what he wants to do next.

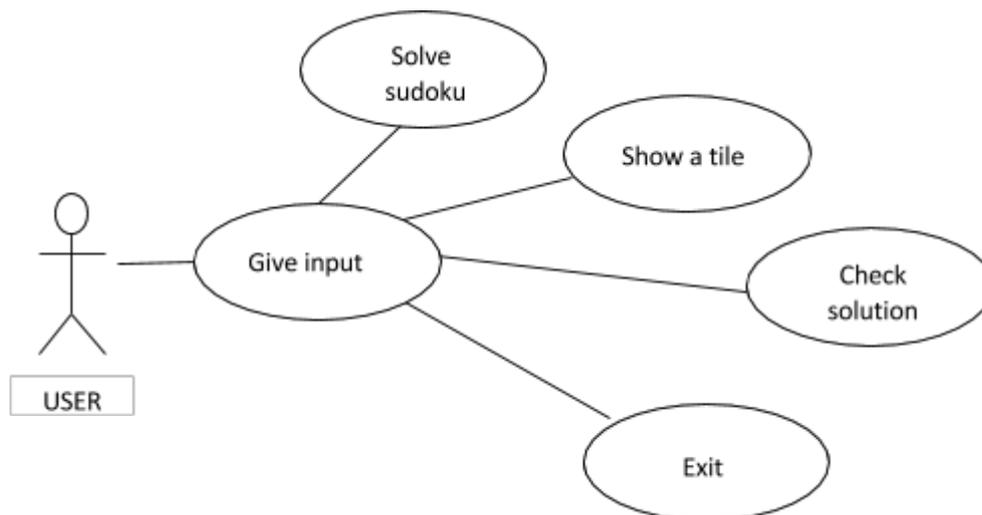
Step-by-step Description

1. The user is asked to input the sudoku.
2. The user inputs the sudoku.
3. The program checks the given input, if the given input is invalid appropriate error message is displayed and program exits.
4. Then the user is asked what does he wish to do next.
5. He is given the options solve sudoku, show a tile, check solution and exit.
6. User chooses appropriate option.

XREF: sec. 3.2.4

2.2.5 Use case : Jigsaw sudoku

Diagram



Brief Description

The user is asked to give input of layout and sudoku. Then he chooses what he wants to do next.

Step-by-step Description

1. The user is asked to input the layout of the jigsaw sudoku. Here he has to specify the boxes in sudoku.
2. The program checks the layout input, if it is invalid appropriate error message is displayed and program exits.
3. The user is then asked to input the.
4. The user inputs the sudoku.
5. The program checks the given input, if the given input is invalid appropriate error message is displayed and program exits.
6. Then the user is asked what does he wish to do next.
7. He is given the options solve sudoku, show a tile, check solution and exit.
8. User chooses appropriate option.

XREF: sec. 3.2.5

2.2.6 Use case : Solve sudoku

Brief Description

The program solves the sudoku and exits.

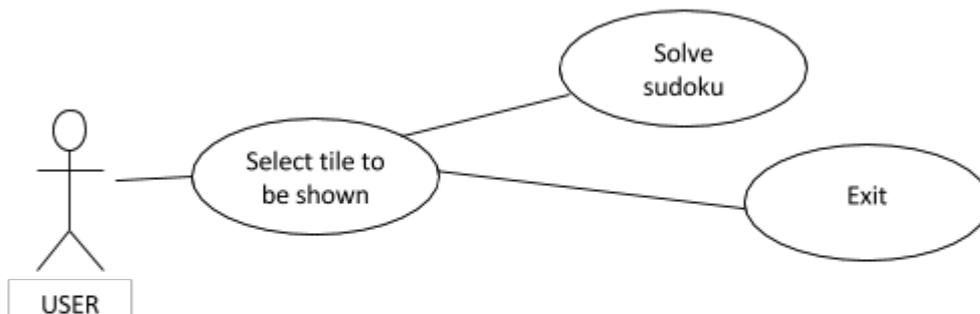
Step-by-step Description

1. The program solves the given sudoku
2. If solution exists then it displays solution.
3. Otherwise it gives error message.
4. The program exits on its own.

XREF: sec. 3.2.6

2.2.7 Use case : Show a tile

Diagram



Brief Description

The user is selects the tile that he wants the program to show. Appropriate tile is displayed. Then he chooses what he wants to do next.

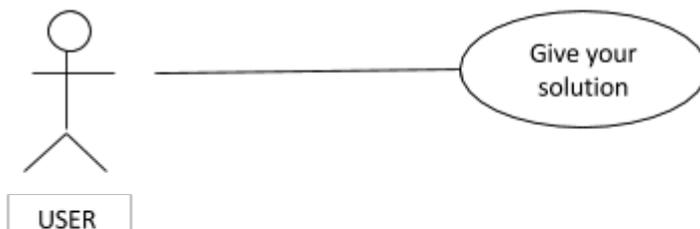
Step-by-step Description

1. The user is asked to input the location of the tile he wants to see.
2. The user inputs the location as row and column.
3. The program checks the given input, if the given input is invalid appropriate error message is displayed.
4. Then the user is asked what does he wish to do next.
5. He is given the options solve sudoku and exit.
6. User chooses appropriate option.

XREF: sec. 3.2.7

2.2.8 Use case : Check solution

Diagram



Brief Description

The user is asked to give his solution. The program checks the solution and displays appropriate message.

Step-by-step Description

1. The user is asked to input his solution.
2. The user inputs the solution.

3. The program checks the given input, if the given input is invalid appropriate error message is displayed and program exits
4. Then the program checks whether the solution is correct or not.
5. Appropriate message is displayed.
6. The program exits on its own.

XREF: sec. 3.2.8

2.2.9 Use case : Exit

Brief Description

User can select this option to exit from the program.

Step-by-step Description

1. The program gets terminated and exits.
2. Appropriate values are returned.

XREF: sec. 3.2.9

2.3 USER CHARACTERISTICS

The User is expected to be familiar with C++ and be able to navigate his way while the program is running.

He should be able to give inputs in format as desired by the program. For this it is recommended that the user has gone through the User Manual.

2.4 NON-FUNCTIONAL REQUIREMENTS

The computer should have all the required C++ program files.

3 REQUIREMENT SPECIFICATION

3.1 EXTERNAL INTERFACE REQUIREMENT

GNU-GCC compiler and Simplecpp interpreter must be installed on the user's computer.

3.2 FUNCTIONAL REQUIREMENT

3.2.1 Main

Use case name	Main
Xref	sec. 2.2.1
Trigger	The user runs the program
Pre-condition	Program starts.
Basic path	<ol style="list-style-type: none">1. The user is directed to a new window that asks him what variant of sudoku does he desire to select.2. He is given the options normal sudoku, diagonal sudoku, window sudoku and jigsaw sudoku.3. User chooses appropriate option.
Post-condition	User is directed to selected variant.
Execption paths	User can abandon the program any time.

3.2.2 Normal Sudoku

Use case name	Normal Sudoku
Xref	sec. 2.2.2
Trigger	User selects normal sudoku.
Pre-condition	sudoku_grid has been declared and initialised as blank.
Basic path	<ol style="list-style-type: none">1. The user is asked to input the sudoku.2. Inputs is taken into sudoku_grid using InputGrid function.

	<ol style="list-style-type: none"> 3. The program validates the given input using ValidN function. 4. If it is invalid, appropriate error message is displayed and program returns. 5. Then the user is asked what does he wish to do next. 6. He is given the options solve sudoku, show a tile, check solution and exit. 7. User chooses appropriate option.
Post-condition	User is directed to selected option
Exception paths	User can abandon the program any time. User can select exit option.

3.2.3 Diagonal Sudoku

Use case name	Diagonal Sudoku
Xref	sec. 2.2.3
Trigger	User selects diagonal sudoku.
Pre-condition	sudoku_grid has been declared and initialised as blank.
Basic path	<ol style="list-style-type: none"> 1. The user is asked to input the sudoku. 2. Inputs is taken into sudoku_grid using InputGrid function. 3. The program validates the given input using ValidD function. 4. If it is invalid, appropriate error message is displayed and program returns. 5. Then the user is asked what does he wish to do next. 6. He is given the options solve sudoku, show a tile, check solution and exit. 7. User chooses appropriate option.
Post-condition	User is directed to selected option
Exception paths	User can abandon the program any time. User can select exit option.

3.2.4 Window sudoku

Use case name	Window sudoku
Xref	sec. 2.2.4
Trigger	User selects window sudoku.
Pre-condition	sudoku_grid has been declared and initialised as blank.
Basic path	<ol style="list-style-type: none"> 1. The user is asked to input the sudoku. 2. Inputs is taken into sudoku_grid using InputGrid function. 3. The program validates the given input using ValidW function. 4. If it is invalid, appropriate error message is displayed and program returns. 5. Then the user is asked what does he wish to do next. 6. He is given the options solve sudoku, show a tile, check solution and exit. 7. User chooses appropriate option.
Post-condition	User is directed to selected option
Exception paths	<p>User can abandon the program any time.</p> <p>User can select exit option.</p>

3.2.5 Jigsaw Sudoku

Use case name	Jigsaw sudoku
Xref	sec. 2.2.5
Trigger	User selects jigsaw sudoku.
Pre-condition	sudoku has been declared and initialised to blank.
Basic path	<ol style="list-style-type: none"> 1. The user is asked to input the layout of the jigsaw sudoku. Here he has to specify the boxes in sudoku. 2. Input is given to sudoku.box using InputBox function. 3. The program validates the layout input using ValidBox function.

	<ol style="list-style-type: none"> 4. If it is invalid, appropriate error message is displayed and program returns. 5. The user is asked to input the sudoku. 6. Inputs is taken into sudoku.grid using InputGrid function. 7. The program checks the given input using ValidJ function. 8. If the given input is invalid, appropriate error message is displayed and program returns. 9. Then the user is asked what does he wish to do next. 10. He is given the options solve sudoku, show a tile, check solution and exit. 11. User chooses appropriate option.
Post-condition	User is directed to selected option
Exception paths	<p>User can abandon the program any time.</p> <p>User can select exit option.</p>

3.2.6 Solve sudoku

Use case name	Solve sudoku
Xref	sec. 2.2.6
Trigger	User selects Solve sudoku.
Pre-condition	Sudoku has been input into sudoku_grid or sudoku.
Basic path	<ol style="list-style-type: none"> 1. The program solves the given sudoku using appropriate Sudoku function. 2. If solution exists then it displays solution using DisplaySudoku function. 3. Otherwise it gives error message.
Post-condition	If solution exists, it has been stored in sudoku_grid or sudoku and has been displayed. Otherwise error has been printed.
Exception paths	<p>User can abandon the program any time.</p> <p>Program returns on its own.</p>

3.2.7 Show a tile

Use case name	Show a tile
Xref	sec. 2.2.7
Trigger	User selects Show a tile.
Pre-condition	Sudoku has been input into sudoku_grid or sudoku.
Basic path	<ol style="list-style-type: none"> 1. The user is asked to input the location of tile he wants to see. 2. Inputs is taken and validated. 3. If it is invalid, appropriate error message is displayed and program returns. 4. The program solves the given sudoku using appropriate Sudoku function. 5. If solution exists then it displays requested tile using DisplaySudoku function. 6. Otherwise it gives error message and program returns. 7. Then the user is asked what does he wish to do next. 8. He is given the options solve sudoku and exit. 9. User chooses appropriate option.
Post-condition	If solution exists, desired tile has been displayed and change has been made in the grid. User is directed to selected option. Otherwise error has been printed.
Exception paths	User can abandon the program any time. User can select exit option.

3.2.8 Check solution

Use case name	Check solution
Xref	sec. 2.2.8
Trigger	User selects Check solution.
Pre-condition	Sudoku has been input into sudoku_grid or sudoku.
Basic path	<ol style="list-style-type: none"> 1. The user is asked to input his solution.

	<ol style="list-style-type: none"> 2. Inputs is taken into solved_grid or solved.grid using InputGrid function. 3. The program validates the given input using appropriate CheckNum function. 4. If it is invalid, appropriate error message is displayed and program returns. 5. Then the program checks whether the solution is correct or not using appropriate CheckSudoku function. 6. Appropriate message is displayed. 7. The program returns on its own.
Post-condition	User is told whether his solution is correct or not.
Exception paths	User can abandon the program any time. Program returns on its own.

3.2.9 Exit

Use case name	Exit
Xref	sec. 2.2.9
Trigger	User selects Exit.
Pre-condition	Sudoku has been input into sudoku_grid or sudoku.
Basic path	<ol style="list-style-type: none"> 1. The program gets terminated 2. Appropriate values are returned.
Post-condition	Program returns.
Exception paths	Program returns on its own.

3.2.10 SolveSudoku

Function name	SolveNSudoku, SolveDSudoku, SolveWSudoku, SolveJSudoku
Xref	sec. 3.2.6, sec. 3.2.7
Trigger	Function is called while solving sudoku or displaying a tile.

Pre-condition	Sudoku has been input into sudoku_grid or sudoku.
Basic path	<ol style="list-style-type: none"> 1. First blank tile is located using FindBlank function. 2. Possible values of digit in above tile is found using appropriate CheckNum function. 3. Temporarily, that value is assigned to above tile. 4. Function is called recursively. 5. If solution exists then it gets stored in grid and function returns true. 6. Otherwise temporary assignment is undone and other values are tried. 7. If no value satisfies the condition, function returns false.
Post-condition	If solution exists, it has been stored in sudoku_grid or sudoku and true value has been returned. Otherwise false value has been returned.
Exception paths	If solution does not exist, function returns false.

3.2.11 CheckNum

Function name	CheckNumN, CheckNumD, CheckNumW, CheckNumJ
Xref	sec. 3.2.2, sec. 3.2.3, sec. 3.2.4, sec. 3.2.5, sec. 3.2.8, sec. 3.2.10
Trigger	Function is called while validating inputs, solving sudoku or checking solution.
Pre-condition	Sudoku has been input into sudoku_grid or sudoku. A tile and digit to be entered has been selected.
Basic path	<ol style="list-style-type: none"> 1. It calls functions to check if the given digit is present in the corresponding row, column, box, diagonal etc. 2. If digit is present in any of these then it is invalid and function returns false. 3. Otherwise, it returns true.
Post-condition	Validity of digit in desired tile has been checked.

3.2.12 Input

Function name	InputGrid, InputBox
Xref	sec. 3.2.2, sec. 3.2.3, sec. 3.2.4, sec. 3.2.5, sec. 3.2.8
Trigger	Function is called to input sudoku or solution.
Pre-condition	Sudoku or solution has been initialised to blank.
Basic path	<ol style="list-style-type: none">1. Prints message asking for input.2. Input is stored in respective location.3. Checks if the input is within the range.4. If not, then it displays error message and exits.5. Otherwise, prints input using DisplaySudoku function.
Post-condition	If input is within range, it has been stored in appropriate location.
Exception paths	If input is not in range, function returns -1.

3.2.13 ValidBox

Function name	ValidBox
Xref	sec. 3.2.5
Trigger	Function is called to validate boxes of jigsaw sudoku.
Pre-condition	Boxes have been defined and stored in sudoku.box
Basic path	<ol style="list-style-type: none">1. Checks whether each tile has at least one neighbour having same box.2. Checks if all boxes have exactly 9 tiles.3. If not it gives error message and returns false.4. Otherwise, it returns true.
Post-condition	Validation of boxes in jigsaw sudoku has been done.
Execption paths	If boxes are invalid, function returns false.

3.2.14 DisplaySudoku

Function name	DisplaySudoku
Xref	sec. 3.2.6, sec. 3.2.7, sec. 3.2.12
Trigger	Function is called to print sudoku while taking input, solving sudoku or displaying a tile.
Pre-condition	Required grid has been input in sudoku_grid.
Basic path	1. Prints the grid row by row
Post-condition	Required grid has been printed.

3.2.15 CheckSudoku

Function name	CheckNSudoku, CheckDSudoku, CheckWSudoku, CheckJSudoku
Xref	sec. 3.2.8
Trigger	Function is called while checking solution.
Pre-condition	Sudoku puzzle and solution are present in sudoku_grid/sudoku and solved_grid/solved respectively.
Basic path	<ol style="list-style-type: none"> 1. If any tile in solution is blank it returns -1 2. Checks whether a non blank tile in puzzle matches solution 3. Checks if all tiles satisfy required conditions. 4. If any of the above is violated, it returns 0. 5. Otherwise it returns 1.
Post-condition	Solution has been checked.
Execption paths	If tile is empty, function returns -1.

3.2.16 Check

Function name	CheckRow, CheckCol, CheckBox, CheckDia1, CheckDia2, CheckJBox
Xref	sec. 3.2.11
Trigger	Function is called from appropriate CheckNum function.

Pre-condition	Sudoku is present in sudoku_grid or sudoku
Basic path	<ol style="list-style-type: none"> 1. Checks whether any tile in given row, column, box, diagonal etc. has value same as required digit. 2. If above is holds for a tile except for a tile corresponding to row and col, it returns true. 3. Otherwise it returns false.
Post-condition	Presence in corresponding row, column, box, diagonal etc.has been checked.

3.2.13 Valid

Function name	ValidN, ValidD, ValidW, ValidJ
Xref	sec. 3.2.2, sec. 3.2.3, sec. 3.2.4, sec. 3.2.5, sec. 3.2.15
Trigger	Function is called to validate the sudoku from CheckSudoku or just after input
Pre-condition	Sudoku is present in sudoku_grid or sudoku
Basic path	<ol style="list-style-type: none"> 1. Uses appropriate CheckNum function over all tiles to see if they abide to the rules of that sudoku 2. In case of Jigsaw sudoku it also checks if InputPresent returns true 3. If all rules are followed, then it returns true
Post-condition	Validation of sudoku has been done.
Execption paths	If sudoku is invalid, function returns false.

3.2.14 Sudoku

Function name	NSudoku, DSudoku, WSudoku, JSudoku
Xref	sec. 3.2.6
Trigger	Function is called to solve the sudoku
Pre-condition	Sudoku is present in sudoku_grid or sudoku
Basic path	<ol style="list-style-type: none"> 1. Uses appropriate SolveSudoku function to solve the sudoku

	<ol style="list-style-type: none"> 2. If no solution exists then it returns -1 3. If solution exists and it returns 1 for unique solution and 0 for non-unique solution
Post-condition	Sudoku has been solved and uniqueness of solution is checked
Execption paths	If cannot be solved, function returns -1.

3.3 DETAILED NON-FUNCTIONAL REQUIREMENTS

3.3.1 Logical Structure of the Data

The data descriptions of each of these data entities is as follows:

Data item	Type	Description	Comments
sudoku_grid	9x9 int array	grid containing sudoku puzzle or solution.	
inp	int	value returned by Input	used to validate input
sudoku_solve	int	value returned by calling SolveSudoku	checks if given sudoku has solution or not
row	int	row of a tile	
col	int	column of a tile	
solved_grid	9x9 int array	grid containing solution	
check_sol	bool	value returned by calling CheckSudoku	checks if given solution is correct or not
jig	object	details of jigsaw sudoku	consists of arrays box, grid and present
box	9x9 int array	boxes of jigsaw sudoku	part of jig
grid	9x9 int array	grid containing jigsaw sudoku puzzle or solution.	part of jig
present	9x10 bool array	if a given digit is present in a box or not	part of jig
sudoku	jig	jigsaw sudoku puzzle or solution	

solved	jig	jigsaw sudoku solution	
index	int	row or column of tile	used in loops
index1	int	row of tile	used in loops
index2	int	column of tile	used in loops
box_grid	9x9 int array	boxes of jigsaw sudoku	
box_tile	9 int array	number of tiles in a box	
grid_num	int	digit in given element	
box_num	int	box of given element	
rowstart	int	starting row of box	
colstart	int	starting column of box	

Index

Box 5, 11, 17, 21, 22, 23, 24, 25, 26

Column 5, 13, 21, 23, 24, 25, 26

Diagonal 4, 5, 7, 9, 15, 16, 21, 23, 24

Display 4, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22, 23

Exit 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22

Grid 5, 15, 16, 17, 18, 19, 20, 21, 23, 25, 26

Input 5, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23

Jigsaw 4, 5, 7, 11, 15, 17, 22, 25, 26

Main 5, 7, 15

Normal 4, 7, 8, 15

Row 5, 13, 21, 23, 24, 25, 26

Solution 4, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26

Solve 3, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26

Sudoku 4, 5, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26

Tile 4, 5, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26

User 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20

Window 4, 5, 7, 10, 15, 17