

# **SUDOKU**

## **Software Requirements Specification**

**Version 1.0**

**19<sup>th</sup> October 2014**

<b>Amey Gupta(Team Leader)</b>	<b>- 140050026</b>
<b>Sumith</b>	<b>- 140050081</b>
<b>Lohith Ravuru</b>	<b>- 140050041</b>
<b>Pramod Vakacharla</b>	<b>- 140050076</b>

Prepared for  
CS101—Course Project

Instructors: Prof. Deepak B Phatak  
Prof. Supratik Chakraborty

Autumn 2014

# Table of Contents

## 1. INTRODUCTION

- 1.1 PURPOSE
- 1.2 PROBLEM STATEMENT
- 1.3 APPROACH
- 1.4 SCOPE
- 1.5 REFERENCES
- 1.6 ACRONYMS AND DEFINITIONS

## 2. GENERAL DESCRIPTION

- 2.1 PRODUCT FUNCTIONS
- 2.2 GENERAL CONSTRAINTS ON USER
- 2.3 ASSUMPTIONS

## 3. SPECIFIC REQUIREMENTS

- 3.1 EXTERNAL INTERFACE REQUIREMENTS
  - 3.1.1 *User Interfaces*
  - 3.1.2 *Hardware Interfaces*
  - 3.1.3 *Software Interfaces*
- 3.2 *Project Features*
  - 3.2.1 *Play Sudoku*
  - 3.2.2. *Sudoku Auto solver*

## 4. CONSTRAINTS

- 4.1 HARDWARE AND SOFTWARE
- 4.2 ERROR HANDLING
- 4.3 ADMINISTRATION FEATURES
- 4.4 NETWORK SUPPORT

## 5. SCOPE FOR EXTENSION

- 5.1 SUDOKU-GAME
- 5.2 AUTO-SOLVER

## 6. APPENDIX

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to present a detailed description of the Sudoku game that the team plans to make. The following sections of the document explain the purpose and features of the game, the interface of the game, and the constraints under which it must operate.

## 1.2 Problem Statement

a) Giving the user a solvable Sudoku puzzle having a unique solution, and finally checking the correctness of the solution. Calculation of the highest scores will be done. High scores of each level of difficulty will be displayed.

b) Solving a Sudoku puzzle given by the user.

## 1.3 Approach

In the second part of the program, the user is given a blank 9x9 grid. He can input any number of values at any positions. The validity of the input values will be checked according to the rules of Sudoku and the grid will be solved by the program. In the first part, we will store the Sudoku puzzles in files according to the level of difficulty. And according to the level of difficulty chosen by the user, a random puzzle from the files will be given to him. The player will be allowed to input values (to play the game). The final answer is verified with the already solved Sudoku. Score will be calculated by taking into account the time taken to solve the puzzle. The score will be checked against the high scores after every game and the high scores will be altered accordingly.

## 1.4 Scope

This game is meant for pure entertainment for people who enjoy playing Sudoku and will attract more fans like people on the go and hungry for more. It can be replicated easily onto any mobile platform. For now it will be employed as a PC mini-game on both Windows and Linux platform. Sudoku has been there for years now but we wish to take it a bit different by introducing different hints in the game, intuitive GUI and various new challenging levels.

## 1.5 References

- i) <https://www.google.co.in/>
- ii) <http://en.wikipedia.org/>
- iii) <http://stackoverflow.com/>
- iv) <http://www.cse.iitb.ac.in/~cs101/>
- v) Introduction to Problem Solving and Programming through C++ by Prof. Abhiram Ranade.
- vi) <http://blog.mpshouse.com/> (tutorials on Gtkmm)
- vii) <https://developer.gnome.org/gtkmm-tutorial>

## 1.6 Acronyms and Definitions:

There are no acronyms and definitions used in this project.

# 2. General Description

## 2.1 Product Functions

This part can basically be classified into two parts **SUDOKU-SOLVER** and **GENERATE-SUDOKU**

### 2.1.1 Sudoku Solver

The **AUTO-SOLVER** will be able to solve the Sudoku provided by user. This can solve any valid Sudoku entered (valid: does not violate the rules of Sudoku).

If the Sudoku entered has multiple solutions. It will display the first encountered solution, displaying a message saying that there are multiple solutions.

### 2.1.2 Generate-Sudoku

The **GENERATE-SUDOKU** will give the user a Sudoku .It also allows the user to play Sudoku. The product will be able to provide Sudoku of varying difficulties with which, the users can play.

The software is developed under the following constraints and assumptions:

## 2.2. General CONSTRAINTS on user:

### 2.2.1. User should input a valid Sudoku:

- 2.2.1.1 Each number should be an integer lying from 1 to 9
- 2.2.1.2 Each number from 1 to 9 should occur exactly once in Every row.
- 2.2.1.3 Each number from 1 to 9 should occur exactly once in Every column
- 2.2.1.4 Each number from 1 to 9 should occur exactly once in Every grid.

### 2.2.2. In case of multiple solutions, can only provide the solution First encountered.

## 2.3 Assumptions

- 2.3.1. The user should have working knowledge of computers.

**2.3.2** The user should understand English.

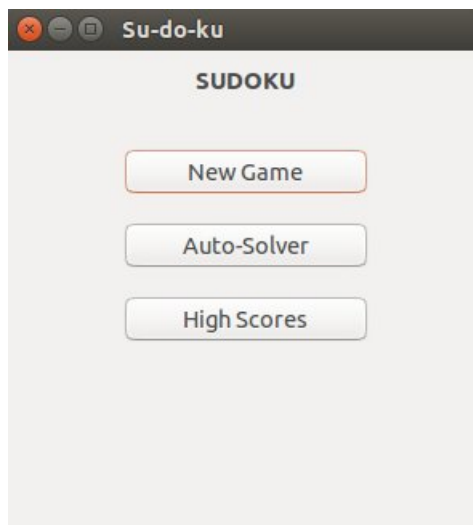
**2.3.3..** The machine on which software is installed is having the minimum  
Required hardware components.

## 3. Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

The software use Graphical User Interface (GUI) implemented using GTK++ in C++.The user interface looks like this:



#### 3.1.2 Hardware Interfaces

The software requires only the basic hardware – monitor, mouse and keyboards. Running locally the software has no special hardware requirements.

#### 3.1.3 Software Interfaces

The Generate--Sudoku will interact with the saved Sudoku, of varying difficulties, file system.

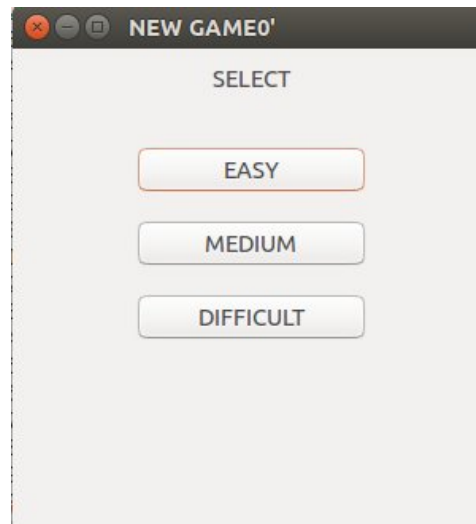
## 3.2 PROJECT FEATURES

The various features provided to the Auto-Solver and the New Game are provided below:

### 3.2.1 PLAY SUDOKU

#### 3.2.1.1 Introduction

- \* This function allows the user to play with a preloaded set of Sudoku, generated by the computer randomly.
- \* The generated Sudoku can be of difficulty as required by user.



#### 3.2.1.2 Inputs

- \*The function will take the 'Difficulty' as input through a mouse click.

#### 3.2.1.3 Processing

The function will then process as per the Difficulty selected. It selects a random Sudoku from the binary file and return it to the user.

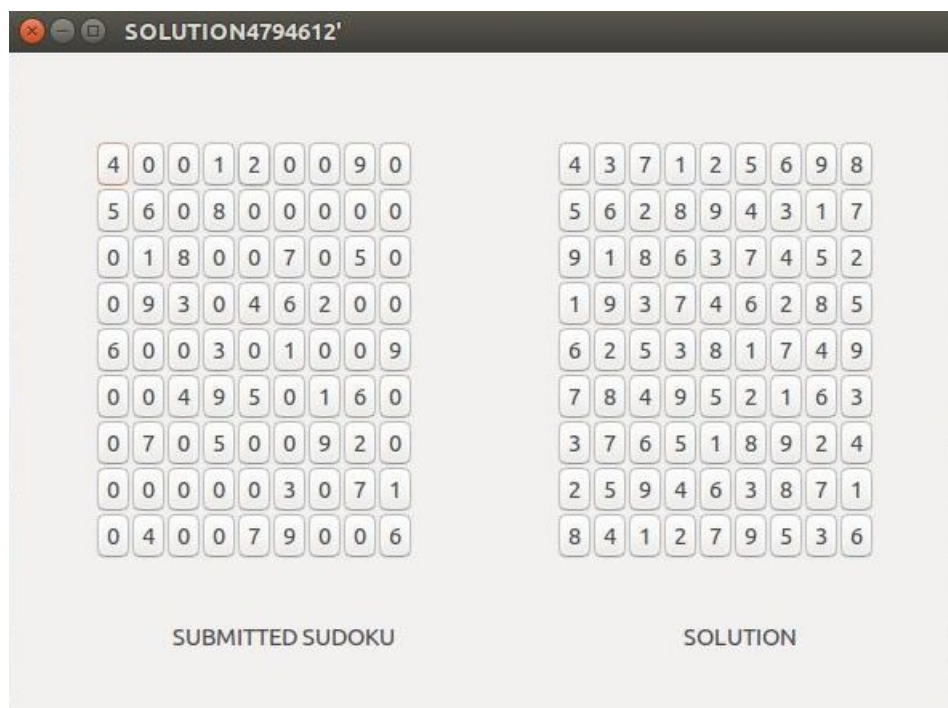
#### 3.2.1.4 Outputs

The function will then output a Sudoku on screen, a 9x9 grid.  
The screen should also contain two labels *Solve* and *Check*.

3.2.1.4.1 **Check** option will check the Sudoku submitted by the user. If the answer is correct then the user gets a chance to make his score to the high scores.



3.2.1.4.2 **Solve** option shows the user the solved Sudoku if the submitted Sudoku is incomplete. It gives a chance for the user to compare his solved Sudoku with the actual answer for the Sudoku.





3.2.1.4.3 All the binary files from which the inputs for the Sudoku puzzle are taken contain a unique solution.

3.2.1.4.4 There is an option in the main menu to check the *High scores* corresponding to each level.

HIGH SCORES'		
EASY MEDIUM DIFFICULT		
Sr. No.	Time(sec) ^	Name
2	10	Shubham Goel
1	11	Gurjot
3	12	Pramod
4	14	Lohith

## 3.2.2 Sudoku Auto solver

### 3.2.2.1 Introduction

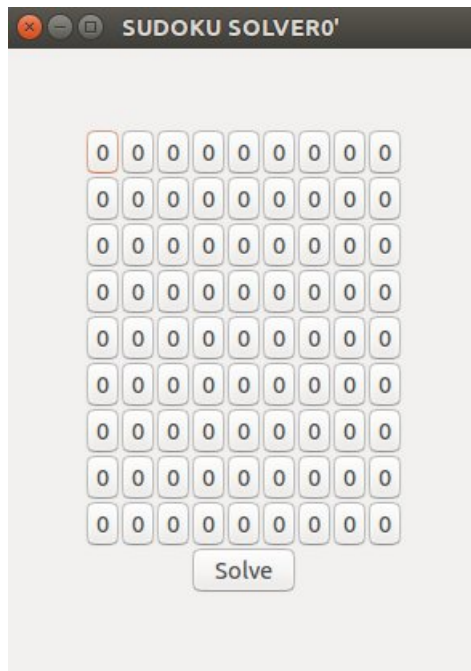
This function basically solves a user provided valid Sudoku.

### 3.2.2.2 Inputs

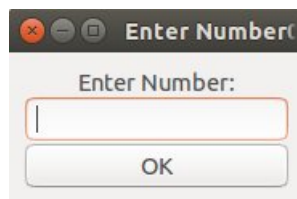
This function will take a Sudoku as input through mouse clicks and keyboard.

The inputs will be integers from 1 to 9.

Initially all the numbers are initialised to Zero.



Once a square is clicked by the mouse, a new window pops out in which the desired input can be given as input. The window looks like this:

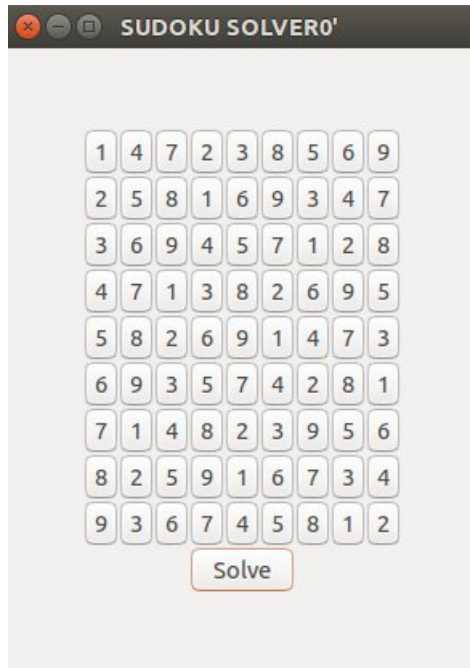


### 3.2.2.3 Processing

This function will then start processing the Sudoku and apply the 'Brute Force with Backtracking Algorithm' to solve the Sudoku.

### 3.2.2.4 Outputs

On clicking Solve button the Sudoku will be solved by the computer. This function will output the solved Sudoku. Every cell will have an integer lying between 1 to 9 and complying with rules of Sudoku.



## 4 CONSTRAINTS

### 4.1 SOFTWARE AND HARDWARE:

The program requires specific software like Code-blocks to run the C++ code and cannot be run in certain other compilers. The Gtkmm header files should be added to the compiler to run the program.

### 4.2 ERROR HANDLING:

Error Handling is limited to a few anticipated or common errors. While playing the Sudoku game error is not specified at each step of the game. It is specified only after the user submits the completed Sudoku.

### 4.3 ADMINISTRATION FEATURES:

The game is not provided with any security protocols and access levels. Various levels of program access and functional authority are not provided in this program.

#### 4.4 NETWORK SUPPORT:

The program is given access to connect to the internet for any updates or help support. However programmers can access to source code and address bugs or system enhancement as deemed necessary.

### 5. SCOPE FOR EXTENSION

#### 5.1 SUDOKU-GAME:

The inputs files containing the unsolved Sudoku puzzles for each level can be increased by adding new puzzles to the existing file in the same format.

There is no need to change any part of the code for this extension.

#### 5.2 AUTO-SOLVER:

There is no extension needed in this part of the program. Any extension to this part needs adding the graphics part to the program code.

## 6. Appendix

### Part 1: The Game Sudoku

This is quite straightforward with the user solving a Sudoku.

Logic- We just need to check whether the value the user input is in the same column, row or corresponding 3x3 box.

### Part 2: Sudoku Solver

Logic-

The user can input as many values as he wants. We will store this in a file. We are using brute force mechanism to solve this Sudoku.

In the very start, we first check if the values given by the user follow the rules of Sudoku. Cells are numbered from 0-80, like in a 2D array in C++. In this mechanism, if we take a cell and it is empty, we put a value in it starting from 1 and going up to 9.

Using a function, we check its legality whether it follows the rules of conventional Sudoku. If there is already some input given by the user, we move to the next cell. If 1 is not suitable, then we increment it and try with 2 and check the legality again. We then move on to the next cell. If there is no legal possibility (none of the values from 1-9 are possible), then we go the previous cell and if it contains a user input then go to its previous cell and increment the value stored in it by 1 and again check the legality. This continues till we get a solution for all the empty cells. We have to be certain that we don't change the user input while solving.

-----THE END-----