Sudoku Application User Manual

July 2017

Version 1.01

Elsys-bg.org

Table of Contents

1 General Information 3

1.1 Product Introduction 4

1.2 Process Overview 5

2 System Summary 6

2.1 System Configuration 7

2.2 Data Flow 7

2.3 User Access Levels 8

3 Getting Started 9

3.1 System Menu 10

3.2 Logging On 12

4 Using The System Online (Logged On) 14

4.1 Solver 15

4.2 Personal Times 17

5 Using The System Offline 18

5.1 Exceptions and Bugs 19

# 

1. **General Information**

# 

# General Information

## Product Introduction

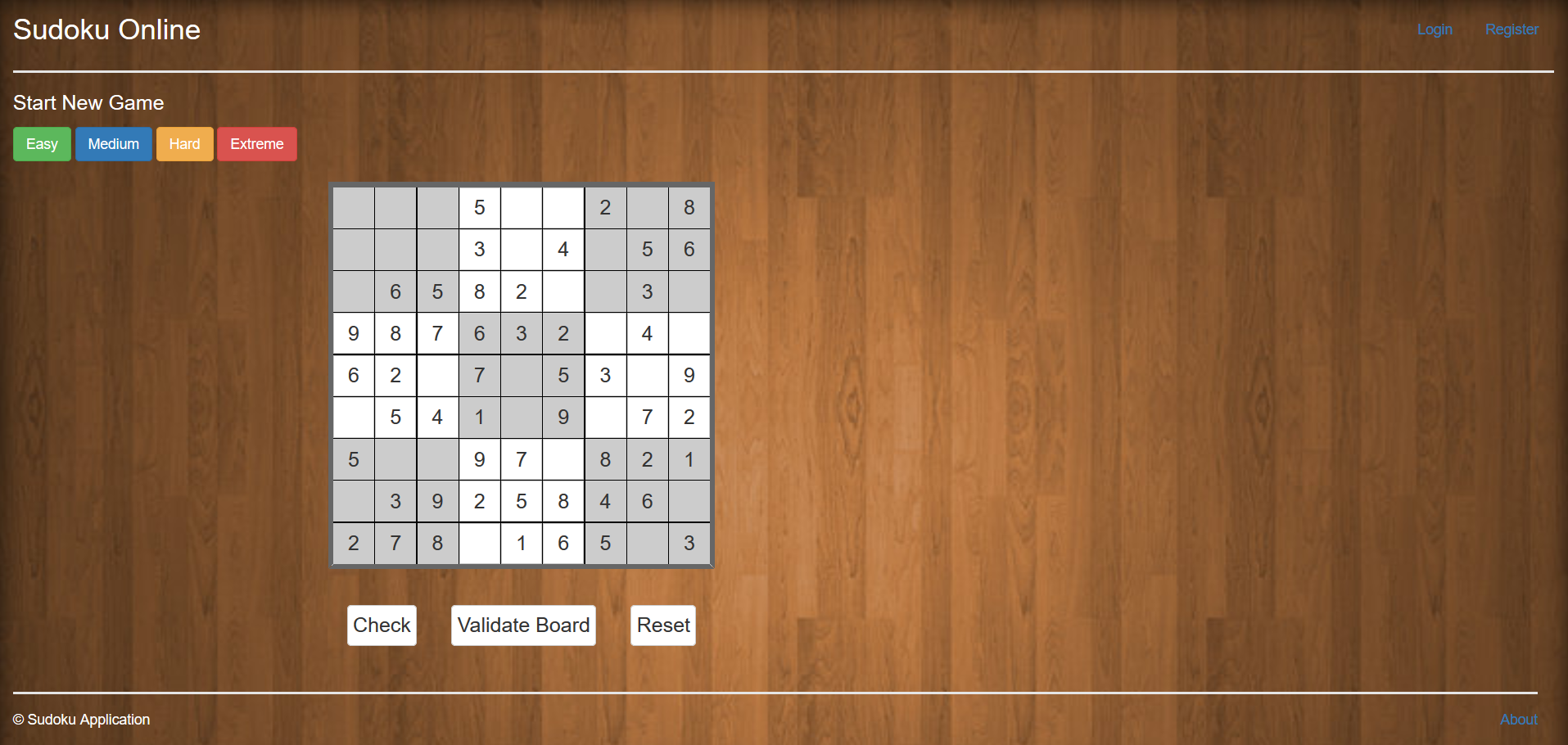
The Sudoku Application is an online sudoku service.

It consists of a Sudoku generator with varying difficulty levels (number of empty boxes), a solving algorithm using backtracking and a Login and Registration system using a database to store user information.

It is used for Entertainment purposes and as a leisure activity.

This is a product in a development stage and is not completed, therefore it can experience minor bugs and/or exceptions.

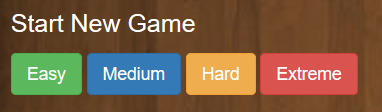
Here is the main page:



## Process Overview

The process which the user goes through on starting the application is as follows:

* The User opens the application through a link.
* The User can play Sudoku with a 9x9 matrix on various difficulty levels.



* The User can Log In the system using a registered account or create one.



* Once Logged On the User can record his time while solving the puzzle and see the recorded times on his profile page.



# 

# System Summary

# System Summary

## System Configuration

To configure the system for normal operation the user must have an Internet connection, a running NodeJS server and a running MongoDB service.

NOTE: The User needs to have all Node modules pre-installed or install them before the server could be started.

The command for installing the required Node modules is ‘*npm install’.*

Once all the prerequisites are acquired, the server is started using the command ‘*node app*’.

The MongoDB database configuration consists of having set up a database with valid username and password. To have a collection named ‘*users*’ and to be using that database.

None other configuration is necessary.

If the service crashes, a server restart will be necessary.

## Data Flow

This is a description of the Users collection in the database

|  |  |
| --- | --- |
| Value Name | Data Type |
| **Name** | String |
| **Username** | String |
| **Email** | String |
| **Password** | String |
| **Solution Times** | Array of Integer |

NOTE: A String is a series of characters. They can be letters, numbers, white spaces or special symbols.

The Data flow of the application follows the shown pattern:

* The User registers a new account – the data for that account is sent and stored in the database.
* When a puzzle of any difficulty is completed, the elapsed time of completion is sent to the database.
* The User can view his times in the ‘*Personal Times*’ page which is accessible from the main page.



* In the ‘*Solver*’ route, when the solving algorithm finishes the elapsed time, number of recursions and number of backtracks is shown in the browser window’s console.

## User Access Levels

There are two levels of access in the application :

* Regular User
  + He has access to play the puzzle and have his solution time recorded.
  + He can view his personal solution times.
  + He has access to the solver route which showcases the solving algorithm.
* Admin
  + He has access to all the aforementioned rights
  + Additionally, he has access to view a page with all of the users in the database, including their names and emails.
  + He does not have access to their solution times nor passwords.

# 

# Getting Started

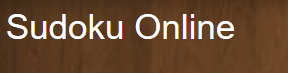
# 

# 3 Getting Started

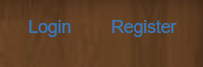
## System Menu

The main menu consists of 10 buttons and a link when the User is not logged in and 11 buttons (12 for Admin) and a link when logged in. This is excluding the puzzle itself. It has as much input fields as there are empty boxes.

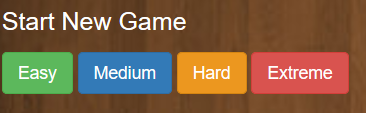
* The link in the upper left corner of the page is a redirection to the main page



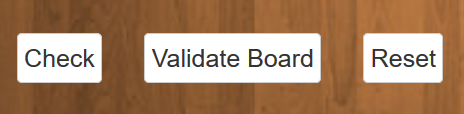
* The buttons in the upper right corner of the screen for logging in and registering



* The buttons for starting a new game (When the page is loaded, the puzzle is generated at an easy setting (default setting) )



* The buttons underneath the puzzle for checking if the solution is right (will make the border red if there are empty boxes), for checking if the board is valid (will make the border green if there are no mistaken numbers – standard Sudoku rules) and for resetting the current solution (will reset the current puzzle with a setting of *Easy* – default setting)

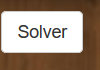


* The button in the bottom right corner of the page is for redirecting to the *About* page which contains information about the project and links to the *Wikipedia* page and the *YouTube* video



When the User is logged in the buttons at the top right corner are replaced with the following ones:

* The button leading to the page showcasing the solving algorithm



* The button for checking the User’s personal solution times



* The button for logging off



## Logging on

* To sign in to the system, the User must have a registered account.

He should go to the ‘*Log In*’ button on the main page and click it.

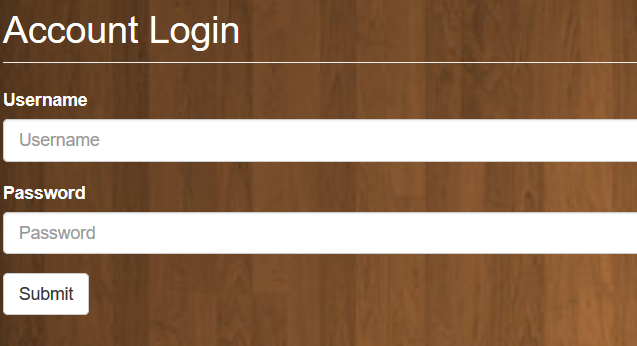
He will be redirected to the Log In page.

Next, he should enter the Username and Password for that account.

Next, he should click the ‘*Submit*’ button under the input fields.

He will then be redirected to the main page.

From there, he will be able to use the functions exclusive to online use.



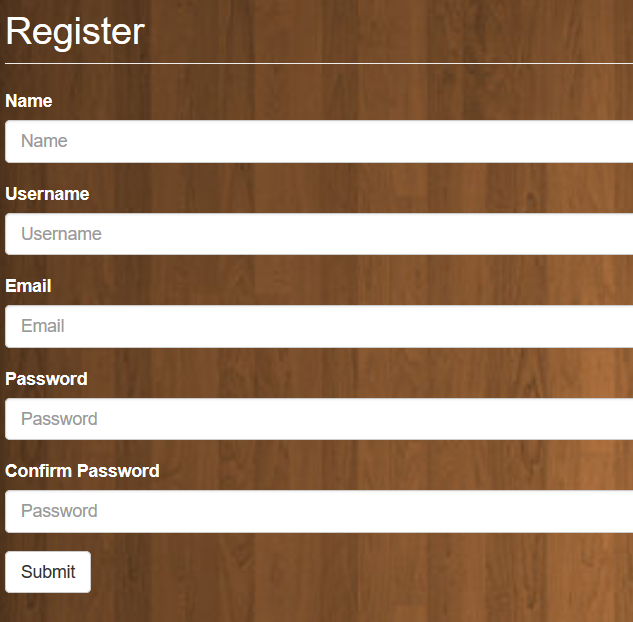
* To sign up a new account, the User should click the ‘*Register*’ button on the main page.

He will then be redirected to the Registration page.

He should fill in all of the required fields and press ‘*Submit*’.

He will be redirected to the main page.

From there, he will be able to use the functions exclusive to online use.



1. **Using the system Online (Logged on)**

# 4 Using the system Online

## Solver

The solving algorithm uses a method called ‘*Backtracking*’. What it essentially does is try a number in the next closest empty square compared to the one being checked and if that number abides by the rules of Sudoku it is saved in that box. Next it tries the next closest box and so on. When, in an empty box, it has tried all numbers from 1 to 9 and none of them obey the rules, it goes *back* one box. If in that box there are legal numbers it continues, if not it goes back one more. From a programmer’s perspective this is done with *Recursion.* The method that does the checking of legal numbers and checks for empty cells (boxes) calls itself when it has not encountered a problem. When there **is** a problem, the program’s execution goes back in the ‘*recursive tree*’ and tries again.

The algorithm has a setting that by default makes it use random numbers as starters for each box creating an aspect of randomness. The chance of two solutions being the same is very slim.

When it has solved the puzzle it will show the elapsed time, times it has gone down the recursive tree and times it has backtracked in the Web Browser’s window console.

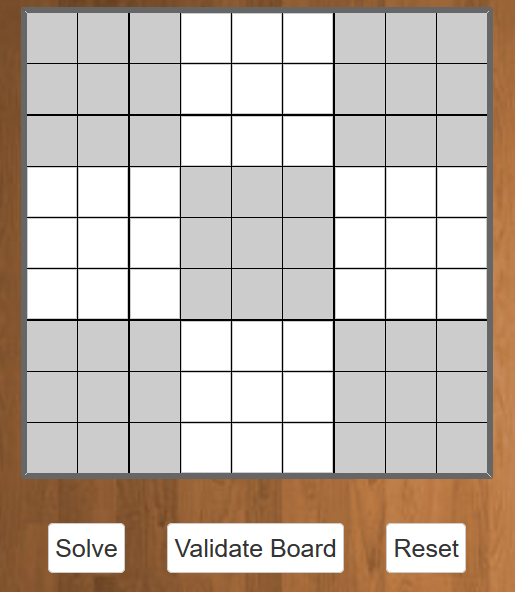
For the User to see this, he has to be on the Solver page and clicked the ‘*Solve*’ button.

This will start the process of solving and will show the solution in the currently empty matrix.

The User can Reset the board if he wants to see another random solution. He can do that as many times as he wants.

NOTE: A typical example of recursion is find the factoriel of numbers. Unless the number is 1, the function calls iself with the next lower number.

Another common example is the Fibonacci series. When numbers are more than 2, the function calls itself with one number lower added with another call to the function with two numbers lower as its input parameter.

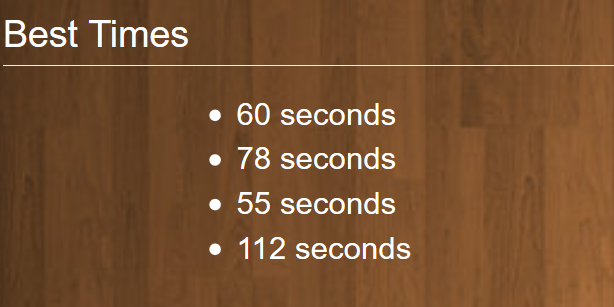


## Personal times

This functionality of the application represents a list of the User’s solution times (in seconds).

It can be accessed from the *Personal Times* button at the top right corner of the page.

The times are calculated by taking the date time from when a puzzle is generated (it can be through a main page refresh or selecting a new difficulty (Easy being the default) and the puzzle Is generated at that difficulty) and the date time from when the Check functionality returns success, and subtracting the later from the former. This value is then sent to the database.



1. **Using the system Offline**

# 5 Using the system offline

## 5.1 Exceptions and Bugs

The system can freely be used with the majority of its functions when not logged in and not creating and using an account.

The puzzles can be played on all difficulties with the only difference being not having the times recorded.

Due to the project being in Development, bugs may occur during operation.

These small errors should be fixed relatively quickly.

If and when a major Error or Exception occurs the product’s author will look into the problem.

Bugs and errors can be submitted as issues at the official GitHub repository for the project - <https://github.com/radito3/TestRepo>