**SUDOKU**

*A*

*Mini Project Report*

*Submitted in partial fulfillment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

IN

**INFORMATION TECHNOLOGY**

By

K.SAI PRASANNA - 1602-20-737-100

E.SPANDANA –1602-20-737-108

D. GANESH REDDY – 1602-20-737-71



**Department of Information Technology**

**Vasavi College of Engineering(Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh,Hyderabad-31**

**2020**

**Vasavi College of Engineering (Autonomous)**

**Affiliated to Osmania University)**

**Hyderabad-500 031**

** Department of Information Technology**

**DECLARATION BY THE CANDIDATE**

We, SAIPRASANNA, SPANDANA, D.GANESH bearing hall ticket numbers 1602-20-737-100,1602-20-737-108,1602-20-737-71 respectively. Hereby declare that the project report entitled “SUDOKU” is submitted in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

K.SAI PRASANNA

1602-20-737-100

E.SPANDANA

1602-20-737-108

D.GANESH

1602-20-737-071

(Faculty-In-Charge) (Head, Dept. of IT)

**AKNOWLEDGEMENTS**

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators and , for guiding us in understanding the process of project development & giving us timely suggestions at every phase. We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions.

**ABSTRACT**

Sudoku game is one of the well-known logical puzzle game. The user starts playing the he/she has an option to enter into the game. When they enter, a puzzle will be generated. The user can start entering the numbers into the grid. There will be an option for the user to view the solution for the puzzle. When the user enters a wrong input the mistakes will be shown on that window. If user completes solving puzzle without any mistakes the game to takes to a greeting window.

**TABLE OF CONTENTS**

* 1. 1. Introduction 6
  2. About The Project 6
  3. What we prioritized……………………………………………………………..6
  4. Objective of our project…………………………………………………………6

2. Technology 7

* 1. Software Requirements 7
  2. Hardware Requirements 7

3. Proposed Work 8

* 1. Design 8
     1. Use case Diagram…………………………………………………………8

Flow Chart…………………………………………………………………9

* 1. Implementation 10
     1. Description of main modules…………………………………………….10
     2. Logic …………………………………………………………………….11
     3. Git hub Folder Structure………………………………………………….11
  2. Testing 18

4. Results 19

5. Additional Knowledge Acquired 23

6.Discussion and future work…………………………………………………………24

7. References ……25

# INTRODUCTION

## 

## ABOUT THE PROJECT

Our project is “SUDOKU”. Our game allows the user to enter into the game .The game helps the users to build their logical skills. User can win by solving the puzzle .User can also view the solution by clicking the solution

**WHAT WE PRIORITIZED**

For playing the game finding the solution for any kind of puzzle is very important. So, we first found the logic for solving the puzzle by backtracking algorithm. Then we asked the user whether he wants to enter the game or not. And at lst we display the puzzle to be solved on the window for the user.

**OBJECTIVE OF OUR PROJECT**

The main aim of our project is to allow the user to play game with different puzzles and also view the solution for that particular puzzle.

.

# TECHNOLOGY

All computer software needs certain hardware components or other software resources to be present, in order for computers to be used efficiently. These prerequisites are known as System Requirements. Within this, we have two types – Software Requirements and Hardware Requirements.

## SOFTWARE REQUIREMENTS

Softwarerequirementsdealwithdefiningthesoftwareresourcerequirementsand prerequisites that need to be installed on a computer to provide optimal functioning of an application. These preconditions are generally not included in the software installation package and need to be installed separately.

In order to use SUDOKU GAME , one should have the following:

* **Operating System:** Windows 7 and above
* **C Compiler**: Idle compiler
* **Editor:** Any text editor (preferably VIM)

## HARDWARE REQUIREMENTS

Hardware requirements refer to the common set requirements defined by any operating system or software application and are usually the physical computer resources. In this, we look into the architecture, processing power, memory, secondary memory, display adapter and peripherals.

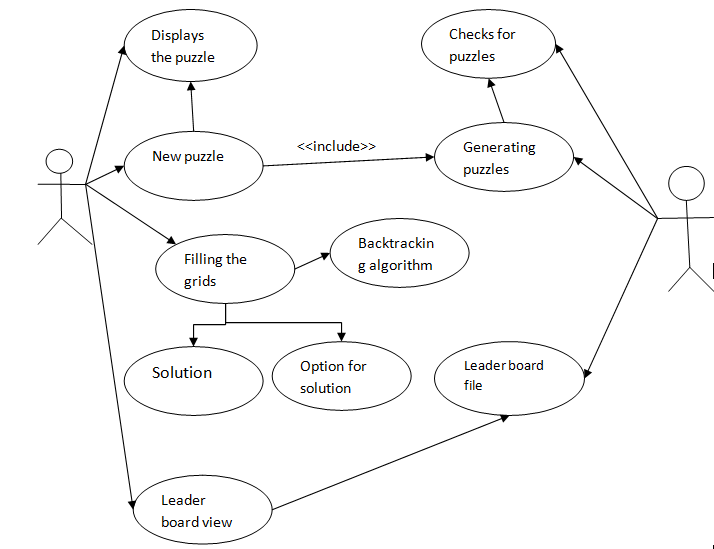
In order to use SUDOKU GAME, one should have the following:

* + - **Processor:** Intel Core i5 and above
    - **Memory:** 4 GBRAM and above

1. PROPOSED WORK

**3.1 DESIGN**

**3.1.1 Use Case diagram**



3.1.2 ACTIVITY DIAGRAM

## 

## 

## 3.2 IMPLEMENTATION

## 3.2.1 DESCRIPTION OF MAIN MODULES

Based on the use cases, we have implemented this project by dividing the work into will do their respective work.

3.2.1.1 SOLVING

When we started implementation first start we thought build an application which gives the solved puzzle for the given puzzle. So first we thought when a number is given we just check if the given number is there in same row or column or 3\*3 grid that is validates the given number and the applications checks for the valid inputs if it is valid then it inserts into the grid calls itself until grid is not filled completely.

3.2.1.2 INSERTION

When user enters into the game he/she must have an option to enter the keys into the grid. For this to happen we came up with the GUI part implementing using the PYGAME library in python. We came to some event handling part. For insert the events that to handled are BUTTONDOWN event, KEYDOWN event and check whether the numbers are in the range of 1 to 9 and then enter into grid. If user wants to erase the data he/she can use 0 to erase.

3.2.1.3 VALIDATING USER INPUTS

When user begins to enter inputs we must validate that inputs. For this to happen we stored the solution and checked with user’s input and validated that input check increase the count accordingly.

3.2.1.4 DISPLAYING

When we start running the application we begin showing an window which shows the start button and also included the number of mistakes in window, a greeting window when user completes solving puzzle.

**3.2.2 LOGIC**

from dokusan import generators

import numpy as np

import pygame

import random

from pprint import pprint

WIDTH= 700

buffer = 45

mistake = 0

background\_color = (250,240,245)

original\_element\_grid\_color = (52,31,151)

example\_board = np.array(list(str(generators.random\_sudoku(avg\_rank = 150))))

arr = example\_board.reshape(9,9)

grid = arr.astype('int32')

grid\_original = [[grid[x][y] for y in range(len(grid[0]))] for x in range(len(grid))]

question = [[grid\_original[x][y] for y in range(len(grid\_original[0]))] for x in range(len(grid\_original))]

def find\_next\_empty(puzzle):

for r in range(9):

for c in range(9):

if puzzle[r][c] == 0:

return r, c

return None, None

def is\_valid(puzzle, guess, row, col):

row\_vals = puzzle[row]

if guess in row\_vals:

return False

col\_vals = [puzzle[i][col]for i in range(9)]

if guess in col\_vals:

return False

row\_start = (row // 3) \* 3

col\_start = (col// 3) \* 3

for r in range(row\_start, row\_start + 3):

for c in range(col\_start, col\_start + 3):

if puzzle[r][c] == guess:

return False

return True

def solve\_sudoku(puzzle):

row, col = find\_next\_empty(puzzle)

if row is None:

return True

for guess in range(1,10):

if is\_valid(puzzle,guess,row,col):

puzzle[row][col] = guess

if solve\_sudoku(puzzle):

return True

puzzle[row][col] = 0

return False

def cont\_mistake(win):

myfont = pygame.font.SysFont('Comic Sans Ms',30)

text = myfont.render("Mistakes:" + str(mistake),True,(0,0,0),(background\_color))

win.blit(text,(90,550))

pygame.display.update()

def congrats\_win():

pygame.init()

win4 = pygame.display.set\_mode((500,200))

pygame.display.set\_caption('')

win4.fill(background\_color)

picture = pygame.image.load(r'C:\Users\vinay\Downloads\congrats.JPG')

win4.blit(picture,(0,0))

smallfont = pygame.font.SysFont('Comic Sans Ms',35)

text = smallfont.render('Yahhoo!' , True , (0,0,0))

win4.blit(text,(300,50))

pygame.display.update()

def display\_sol():

pygame.init()

pygame.display.set\_caption('Solution')

win1 = pygame.display.set\_mode((WIDTH,WIDTH))

win1.fill(background\_color)

pygame.display.flip()

myfont = pygame.font.SysFont('Comic Sans Ms',30)

for i in range(0,10):

if i % 3 == 0:

pygame.draw.line(win1,(0,0,0),(50+50\*i,50),(50+50\*i,500),4)

pygame.draw.line(win1,(0,0,0),(50,50+50\*i),(500,50+50\*i),4)

pygame.draw.line(win1,(0,0,0),(50+50\*i,50),(50+50\*i,500),2)

pygame.draw.line(win1,(0,0,0),(50,50+50\*i),(500,50+50\*i),2)

pygame.display.update()

for i in range(0,len(question[0])):

for j in range(0,len(question[0])):

if 0 < question[i][j] < 10:

value = myfont.render(str(question[i][j]),True,original\_element\_grid\_color)

win1.blit(value,((j+1)\*50+15,(i+1)\*50+5))

for i in range(0,len(grid\_original[0])):

for j in range(0,len(grid\_original[0])):

if question[i][j] == 0:

value = myfont.render(str(grid\_original[i][j]),True,(0,0,0))

win1.blit(value,((j+1)\*50+15,(i+1)\*50+5))

pygame.display.update()

def insert(win,position):

global mistake

i,j = position[1],position[0]

myfont = pygame.font.SysFont('Comic Sans Ms',30)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

return

if event.type == pygame.KEYDOWN:

if event.type == pygame.QUIT:

pygame.quit()

return

if question[i-1][j-1] != 0:

return

if event.key == 48:

if grid[i - 1][j - 1] != grid\_original[i - 1][j - 1]and mistake > 0:

mistake -= 1

cont\_mistake(win)

grid[i-1][j-1] = event.key - 48

text = myfont.render(" ",True,(0,0,0),(background\_color))

pygame.draw.rect(win,background\_color,(position[0]50+2 buffer,position[1]50+2 buffer,50-buffer,50-buffer))

win.blit(text,(position[0]\*50 + 15,position[1]\*50+ 5))

pygame.display.update()

return

if 0 < event.key - 48 < 10 :

if grid\_original[i -1][j - 1] != event.key - 48:

mistake += 1

cont\_mistake(win)

value = myfont.render(str(event.key -48),True,(255,0,0))

else:

value = myfont.render(str(event.key -48),True,(0,0,0))

pygame.draw.rect(win,background\_color,(position[0]\*50 + 2\*buffer,position[1]\*50+2\*buffer,50-buffer,50 -buffer))

win.blit(value,(position[0]\*50 + 15,position[1]\*50+ 5))

grid[i-1][j-1] = event.key - 48

pygame.display.update()

return

return

if (grid\_original == grid).all() and mistake == 0:

congrats\_win()

def main():

pygame.init()

pygame.display.set\_caption('Puzzle')

win = pygame.display.set\_mode((WIDTH,WIDTH))

win.fill(background\_color)

pygame.display.flip()

myfont = pygame.font.SysFont('Comic Sans Ms',30)

smallfont = pygame.font.SysFont('Corbel',25)

text = smallfont.render('Solution' , True , (240,240,240))

for i in range(0,10):

if i % 3 == 0:

pygame.draw.line(win,(0,0,0),(50+50\*i,50),(50+50\*i,500),4)

pygame.draw.line(win,(0,0,0),(50,50+50\*i),(500,50+50\*i),4)

pygame.draw.line(win,(0,0,0),(50+50\*i,50),(50+50\*i,500),2)

pygame.draw.line(win,(0,0,0),(50,50+50\*i),(500,50+50\*i),2)

pygame.display.update()

for i in range(0,len(grid[0])):

for j in range(0,len(grid[0])):

if 0 < grid[i][j] < 10:

value = myfont.render(str(grid[i][j]),True,original\_element\_grid\_color)

win.blit(value,((j+1)\*50+15,(i+1)\*50+5))

pygame.display.update()

while True:

width = 30

heigth = 30

for event in pygame.event.get():

position = pygame.mouse.get\_pos()

if event.type == pygame.MOUSEBUTTONUP and event.button == 1:

pos = pygame.mouse.get\_pos()

insert(win,(pos[0]//50,pos[1]//50))

if event.type == pygame.QUIT:

pygame.quit()

return

if event.type == pygame.MOUSEBUTTONDOWN:

position = pygame.mouse.get\_pos()

if width <= position[0] <= width \* 16 + 140 and heigth \* 18 <= position[1] <= heigth \* 18 + 40:

display\_sol()

if width \* 16 <= position[0] <= width \* 16 + 140 and heigth \* 18 <= position[1] <= heigth \* 18 + 40:

pygame.draw.rect(win,(160,160,160),[width \* 16,heigth \* 18,140,40])

else:

pygame.draw.rect(win,(120,120,120),[width \* 16,heigth \* 18,140,40])

win.blit(text,(width \* 16 + 28,heigth \* 18 + 10))

pygame.display.update()

if (grid\_original == grid).all() and mistake == 0:

congrats\_win()

def game\_display():

pygame.init()

win3 = pygame.display.set\_mode((WIDTH,WIDTH))

pygame.display.set\_caption('Sudoku')

win3.fill(background\_color)

picture = pygame.image.load(r'C:\Users\vinay\Downloads\image.jpeg')

win3.blit(picture,(80,80))

smallfont = pygame.font.SysFont('Corbel',20)

text = smallfont.render('Let start' , True , (240,240,240))

pygame.display.update()

width = 30

heigth = 30

global grid\_original

solve\_sudoku(grid\_original)

while True:

position = pygame.mouse.get\_pos()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

return

if event.type == pygame.MOUSEBUTTONDOWN:

position = pygame.mouse.get\_pos()

if width <= position[0] <= width \* 10 + 140 and heigth \* 14 <= position[1] <= heigth \* 14 + 40:

pygame.mouse.set\_pos([0,0])

main()

if width \* 10 <= position[0] <= width \* 10 + 140 and heigth \* 14 <= position[1] <= heigth \* 14 + 40:

pygame.draw.rect(win3,(0,146,69),[width \* 10,heigth \* 14,140,40])

else:

pygame.draw.rect(win3,(34,181,115),[width \* 10,heigth \* 14,140,40])

win3.blit(text,(width \* 10 + 28,heigth \* 14 + 10))

pygame.display.update()

game\_display()

3.2.3GITHUB/FOLDER STRUCTURE

We have segregated the files we have used based on the usage. The Main .py file is not part of any folder. We have both our final presentation and documentation of the project which are not part of any folder. The parts are like generation, insertion, solution.

GITHUB LINK:

<https://github.com/KothireddySaiprasanna/mini-project>

## 3.3TESTING

Testing is a method to check whether the actual product matches the expected requirements and to ensure that the product is defect-free. This process involves execution of various parts of the product either using manual or automated tools. The purpose is to identify errors, missing requirements in contrast to the actual requirements.

TEST CASE 1

When user enters into game the case where the user does not solve the puzzle.

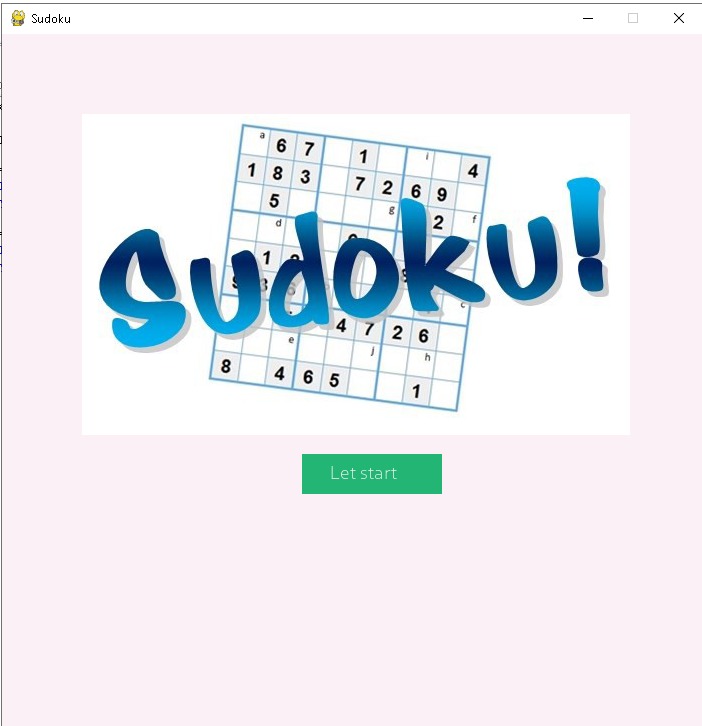
TEST CASE 2

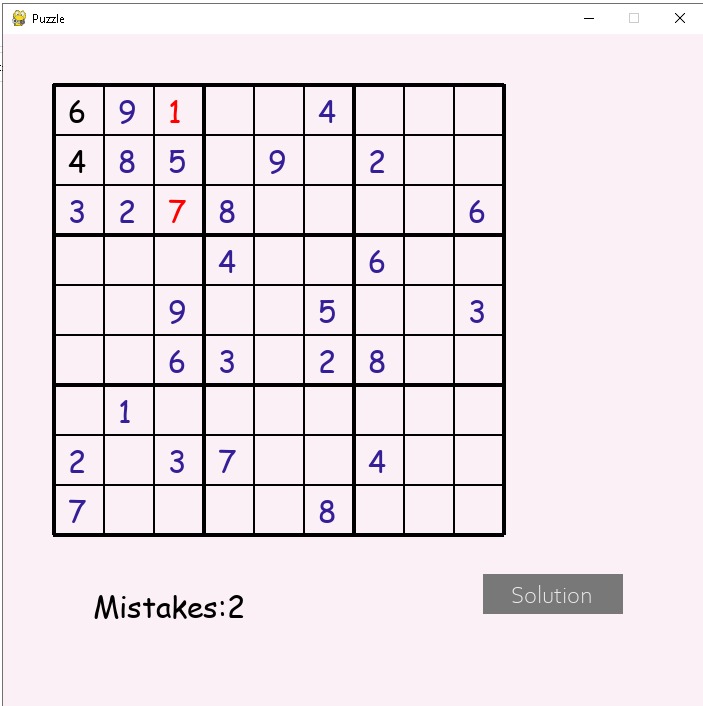
When the user enters into game the case where the user solves the puzzle.

# 

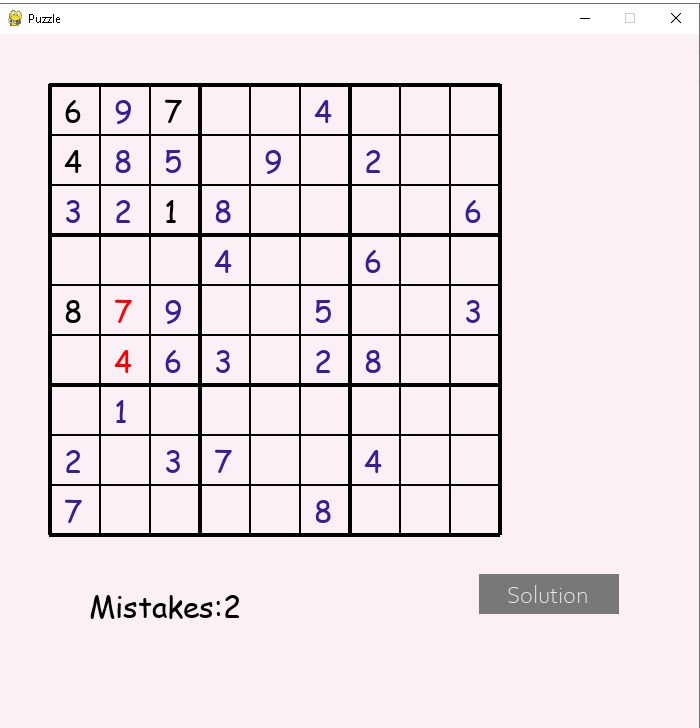
# 4.RESULTS

We have successfully developed a platform for users of all age groups to utilize and improve their skill-set. Below are the output screenshots of the testcases mentioned.

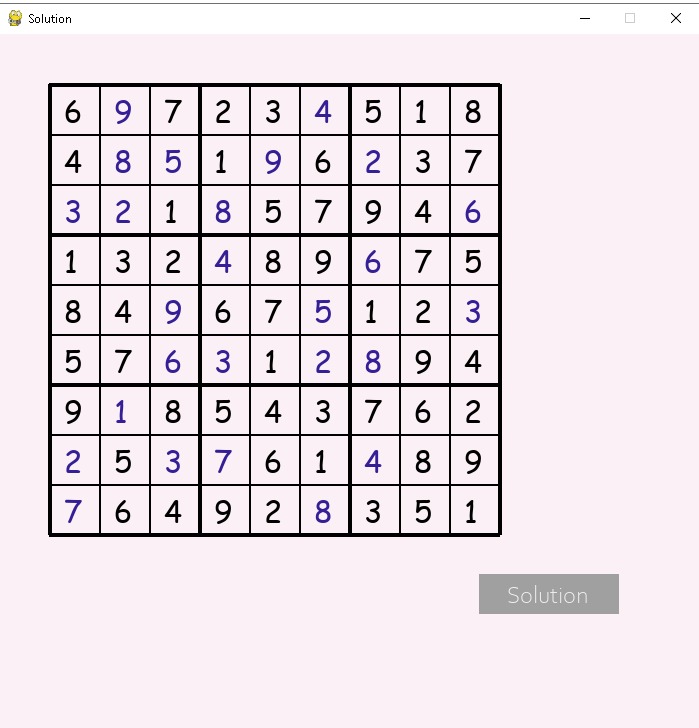




**When user wants to erase the data by clicking the 0**

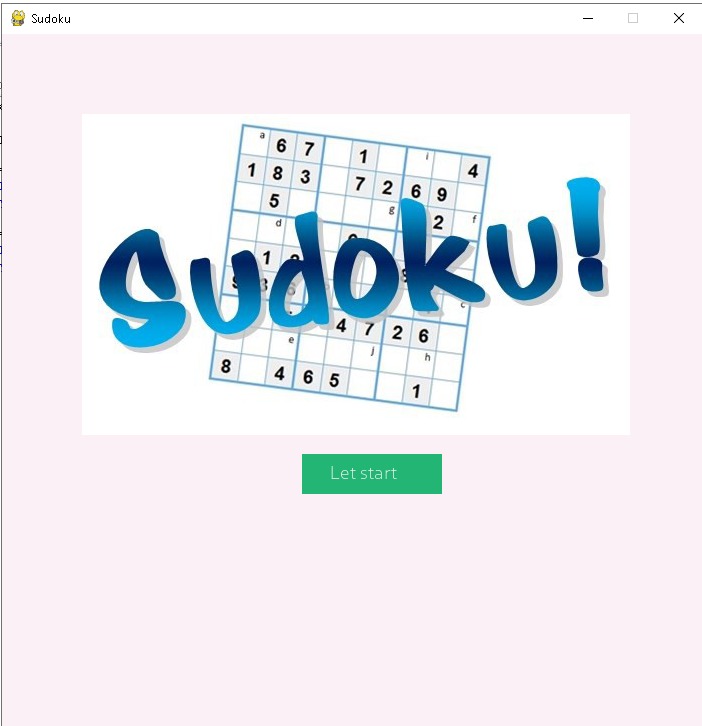
****

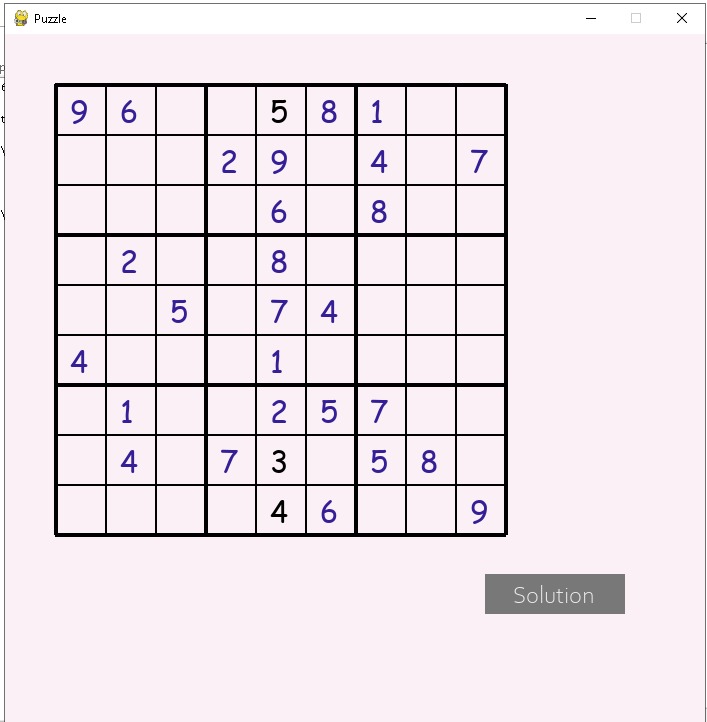
**When user wants to view the solution**

****

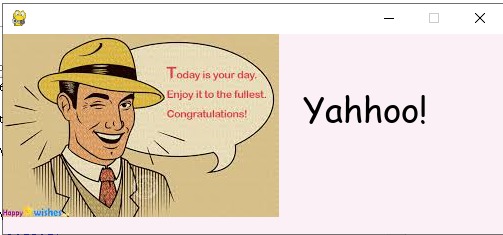
**End of application**

**Each the user run the application it generates new puzzles**

****

****

**When the puzzle is solved we will be directed to this window.**

****

# 5. ADDITIONAL KNOWLEDGE ACQUIRED

By implementing this project we came to more about GUI part like event handling and event delegation methods which perform the operations in response to the actions and also came to about new algorithm which solves the puzzle if to is the valid puzzle and also we came to know about library for generating puzzles.

We gained some more knowledge in python libraries and event handling, backtracking algorithm.

# 6. DISCUSSION AND FUTUREWORK

Our future work includes levels like user can select whether he wants to play 6\*6 many more levels. This project can be further improved by converting it into app. We can extend this by including the login page using database. By including timer to this we can add additional option called leader board. Based on difficulty levels categorizing into hard, medium, easy levels.

# 7. REFERENCES

1. <https://www.pygame.org/docs/>
2. <https://www.geeksforgeeks.org/backtracking-algorithms/>
3. https://www.101computing.net/sudoku-generator-algorithm/