DSA Project Documentation

Terminal Typing Master

Objective:

To develop a terminal-based typing test application. The application tests the user's typing speed by presenting a list of random words from a selected category and measures metrics like words per minute (WPM).

Program:

```
import random

import json

import time

import os

WORD_FILE='words.json'

LEADERBOARD_FILE='leaderboard.json'

def load_words_from_json(filename):

   with open(filename,'r') as file:

      words_data=json.load(file)

   return words_data

def update_leaderboard(username,wpm):

   leaderboard=load_leaderboard()

   user_exists=False

   for entry in leaderboard:

      if entry['username']==username:

         entry['wpm']=wpm

         user_exists=True

         break

   if not user_exists:
        leaderboard.append({'username':username,'wpm':wpm})
    leaderboard.sort(key=lambda x:x['wpm'],reverse=True)
    with open(LEADERBOARD_FILE,'w') as file:
        json.dump(leaderboard,file,indent=4)
def load_leaderboard():
    if os.path.exists(LEADERBOARD_FILE):
        with open(LEADERBOARD_FILE,'r') as file:
            leaderboard=json.load(file)
        return leaderboard
    else:
        return []
def get_user_input(prompt):
    user_input=input(prompt)
    return user_input
def start_typing_test(words):
    username=get_user_input("Enter your username: ")
    start_time=time.time()
    correct_words=0
    total_words=len(words)
    for word in words:
        user_input=get_user_input(f"Type '{word}': ")
        if user_input==word:
            correct_words+=1
    end_time=time.time()
    time_taken=end_time-start_time
    wpm=int((correct_words/time_taken)*60)
    print(f"Words Per Minute (WPM): {wpm}")
    update_leaderboard(username,wpm)
```

```
def show_leaderboard():
    leaderboard=load_leaderboard()
    if not leaderboard:
        print("Leaderboard is empty.")
    else:
        print("Leaderboard:")
        for entry in leaderboard:
            print(f"{entry['username']}: {entry['wpm']} WPM")
def main():
    while True:
        print("\nMenu:")
        print("1.Start Typing Test")
        print("2.Show Leaderboard")
        print("3.Exit")
        choice=get_user_input("Enter your choice: ")
        if choice=='1':
            words_data=load_words_from_json(WORD_FILE)
            words=random.sample(words_data['typing_category'],10)
            start_typing_test(words)
        elif choice=='2':
            show_leaderboard()
        elif choice=='3':
            print("Exiting the Typing Test Application.")
            break
        else:
            print("Invalid choice. Please try again.")
if __name__=="__main__":
    main()
```

Output:

```
Menu:
1.Start Typing Test
2.Show Leaderboard
3.Exit
Enter your choice: 1
Enter your username: User2
Type 'strawberry': strawberry
Type 'orange': orange
Type 'lemon': lemon
Type 'raspberry': raspberry
Type 'blueberry': blueberry
Type 'papaya': papaya
Type 'apple': apple
Type 'quince': quince
Type 'honeydew': honeydew
Type 'fig': fig
Words Per Minute (WPM): 19

Menu:
1.Start Typing Test
2.Show Leaderboard
3.Exit
Enter your choice: 2
Leaderboard:
User4: 70 WPM
User3: 55 WPM
User4: 20 WPM
```

```
User5: 20 WPM
User2: 19 WPM
User1: 18 WPM
Menu:
1.Start Typing Test
2.Show Leaderboard
3.Exit
Enter your choice: 3
Exiting the Typing Test Application.
```

Terminal-Based Maze Solver

Objective:

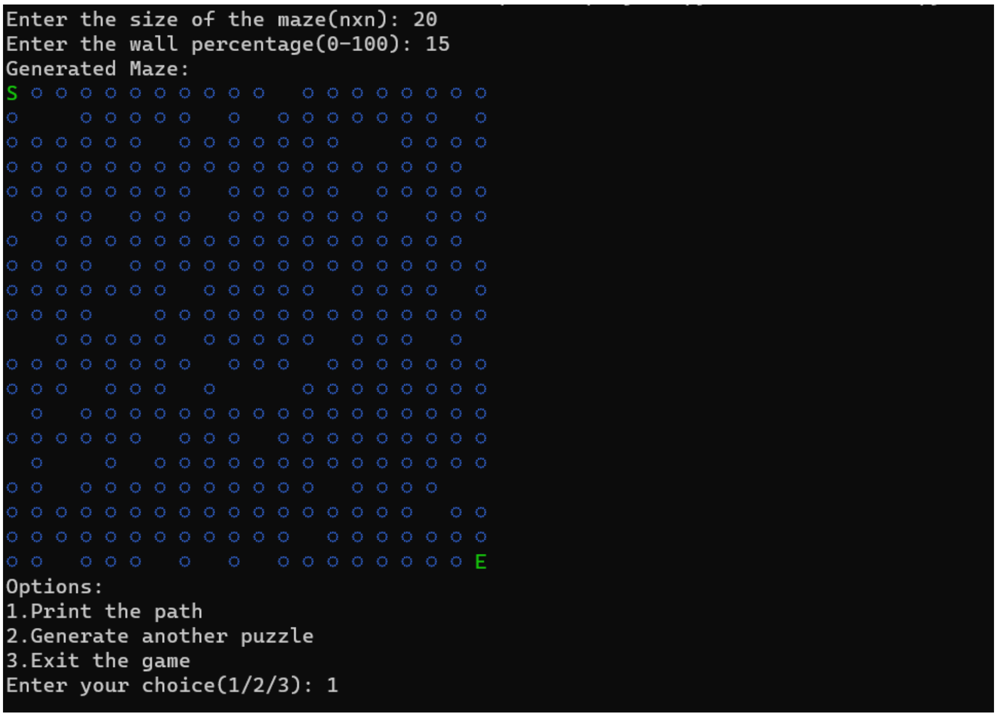
To develop a terminal-based application that generates a random maze, finds a path from the start to the end, and visualizes the maze and path in the terminal.

Program:

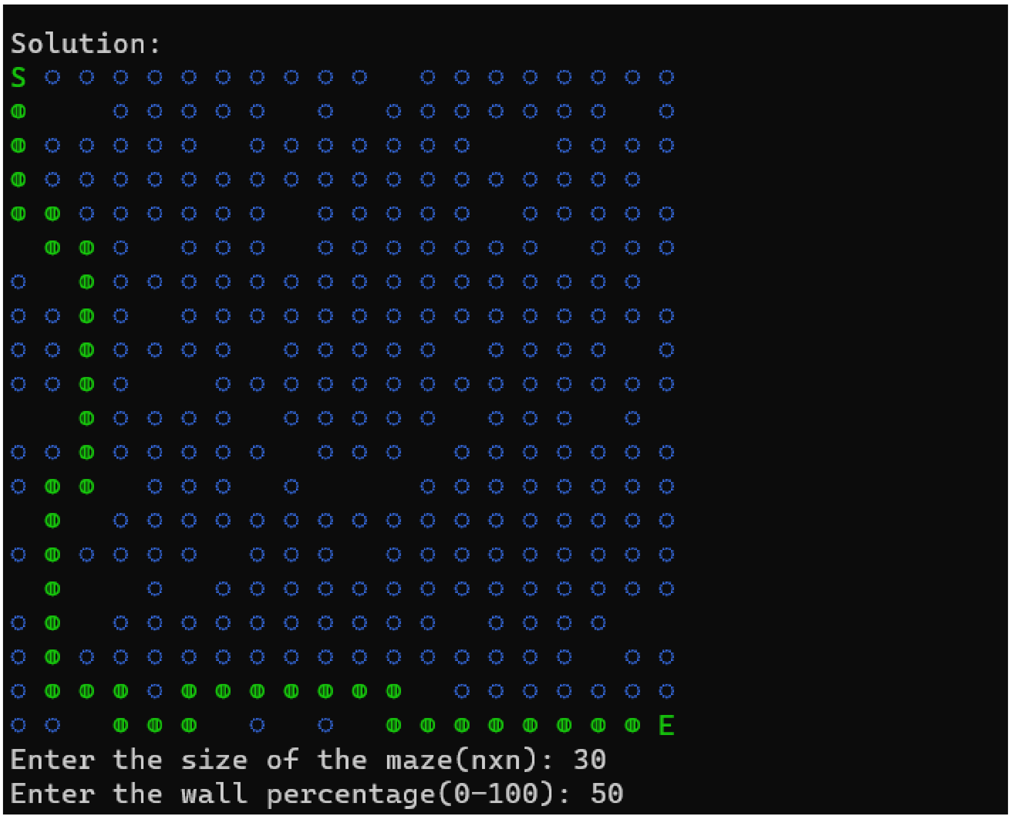
```
        import random

RED="\033[91m"

BLUE="\033[94m"

GREEN="\033[92m"

END_COLOR="\033[0m"

WALL=f"{RED}▓{END_COLOR}"

OPEN_SPACE=f"{BLUE}◌{END_COLOR}"

START=f"{GREEN}S{END_COLOR}"

END=f"{GREEN}E{END_COLOR}"

PATH=f"{GREEN}◍{END_COLOR}"

def generate_maze(n,wall_percentage):

    maze=[[OPEN_SPACE for _ in range(n)] for _ in range(n)]

    num_walls =int(n*n* wall_percentage/100)

    for _ in range(num_walls):

        row,col=random.randint(0,n-1),random.randint(0,n-1)

        maze[row][col]=WALL

    maze[0][0]=START

    maze[n-1][n-1]=END

    return maze

def print_maze(maze):

    for row in maze:
        print(" ".join(row))
def find_path(maze):
    def is_valid_move(row,col):
        return 0<=row<len(maze) and 0<=col<len(maze[0]) and maze[row][col] in
(OPEN_SPACE,END)
    def dfs(row,col):
        if row==len(maze)-1 and col==len(maze[0])-1:
            return True
        for dr,dc in [(1,0),(0,1),(-1,0),(0,-1)]:
            new_row,new_col=row+dr,col+dc
            if is_valid_move(new_row,new_col):
                if maze[new_row][new_col]==OPEN_SPACE:
                    maze[new_row][new_col]=PATH
                if dfs(new_row,new_col):
                    return True
                if maze[new_row][new_col]==PATH:
                    maze[new_row][new_col]=OPEN_SPACE
        return False
    if dfs(0,0):
        return True
    else:
        return False
def print_path(maze):
    for row in maze:
        print(" ".join(row))
def main():
    while True:
        n=int(input("Enter the size of the maze(nxn): "))
        wall_percentage=int(input("Enter the wall percentage(0-100): "))
        maze=generate_maze(n,wall_percentage)
        print("Generated Maze:")
        print_maze(maze)
        print("Options:")
        print("1.Print the path")
        print("2.Generate another puzzle")
        print("3.Exit the game")
        choice=input("Enter your choice(1/2/3): ")
        if choice=="1":
            if find_path(maze):
                print("\nSolution:")
                print_path(maze)
            else:
                print("\nNo solution found.")
        elif choice=="2":
            continue
        elif choice=="3":
            break
        else:
            print("Invalid choice. Please select 1,2,or 3.")
if __name__=="__main__":
    main()
```

Output:

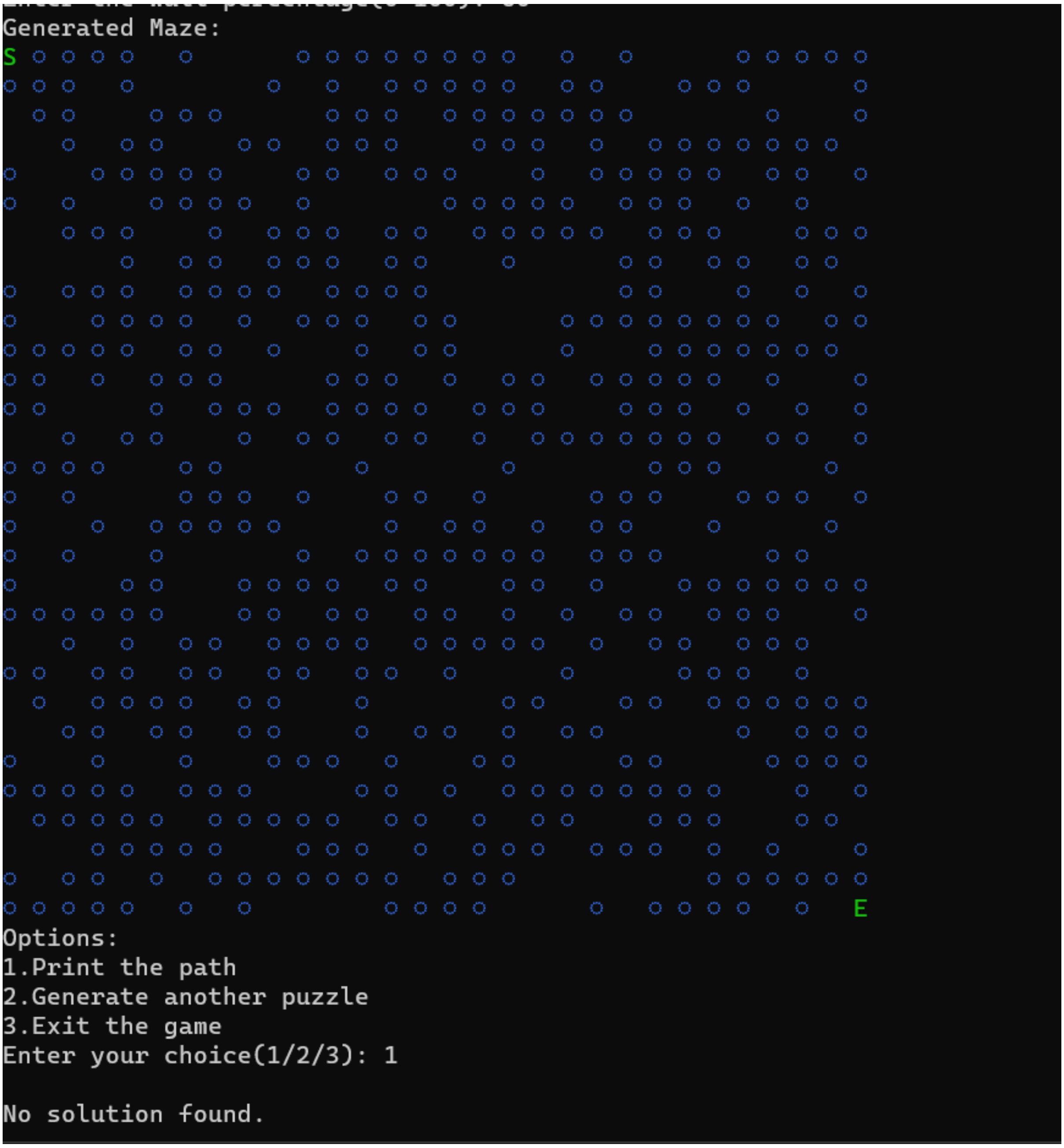
Screenshots:



```
Enter the size of the maze(nxn): 20
Enter the wall percentage(0-100): 15
Generated Maze:
S ○ ○ ○ ○ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○ ○ ○ ○
○     ○ ○ ○ ○ ○   ○   ○ ○ ○ ○ ○ ○ ○   ○
○ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○ ○ ○     ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○   ○ ○ ○ ○ ○
  ○ ○ ○   ○ ○ ○   ○ ○ ○ ○ ○ ○ ○   ○ ○ ○
○   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○   ○ ○ ○ ○   ○
○ ○ ○ ○     ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
    ○ ○ ○ ○ ○   ○ ○ ○ ○ ○   ○ ○ ○   ○
○ ○ ○ ○ ○ ○ ○ ○   ○ ○ ○   ○ ○ ○ ○ ○ ○ ○
○ ○ ○   ○ ○ ○   ○       ○ ○ ○ ○ ○ ○ ○ ○
  ○   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○   ○ ○ ○   ○ ○ ○ ○ ○ ○ ○ ○ ○
  ○     ○   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ○   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○   ○ ○
○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○ ○ ○
○ ○   ○ ○ ○   ○   ○   ○ ○ ○ ○ ○ ○ ○ ○ E
Options:
1.Print the path
2.Generate another puzzle
3.Exit the game
Enter your choice(1/2/3): 1
```



```
Solution:
S ○ ○ ○ ○ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○ ○ ○ ○
◍     ○ ○ ○ ○ ○   ○   ○ ○ ○ ○ ○ ○ ○   ○
◍ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○ ○ ○     ○ ○ ○ ○
◍ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
◍ ◍ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○ ○   ○ ○ ○ ○ ○
  ◍ ◍ ○   ○ ○ ○   ○ ○ ○ ○ ○ ○ ○   ○ ○ ○
○   ◍ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ○ ◍ ○   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ○ ◍ ○ ○ ○ ○   ○ ○ ○ ○ ○   ○ ○ ○ ○   ○
○ ○ ◍ ○     ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
    ◍ ○ ○ ○ ○   ○ ○ ○ ○ ○   ○ ○ ○   ○
○ ○ ◍ ○ ○ ○ ○ ○   ○ ○ ○   ○ ○ ○ ○ ○ ○ ○
○ ◍ ◍   ○ ○ ○   ○       ○ ○ ○ ○ ○ ○ ○ ○
  ◍   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ◍ ○ ○ ○ ○   ○ ○ ○   ○ ○ ○ ○ ○ ○ ○ ○ ○
  ◍     ○   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
○ ◍   ○ ○ ○ ○ ○ ○ ○ ○ ○ ○   ○ ○ ○ ○
○ ◍ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○   ○ ○
○ ◍ ◍ ◍ ○ ◍ ◍ ◍ ◍ ◍ ◍ ◍   ○ ○ ○ ○ ○ ○ ○
○ ○   ◍ ◍ ◍   ○   ○   ◍ ◍ ◍ ◍ ◍ ◍ ◍ ◍ E
Enter the size of the maze(nxn): 30
Enter the wall percentage(0-100): 50
```



```
Enter the wall percentage(0-100): 50
Generated Maze:
S o o o o   o       o o o o o o o o   o   o       o o o o o
o o o   o         o   o   o o o o o   o o     o o o       o
  o o     o o o       o o o   o o o o o o o         o     o
    o   o o     o o   o o o     o o o   o   o o o o o o o
o     o o o o o     o o   o o o     o   o o o o o   o o   o
o   o     o o o o   o         o o o o o   o o o   o   o
    o o o     o   o o o   o o   o o o o o   o o o     o o o
        o   o o   o o o   o o     o       o o   o o   o o
o   o o o   o o o o   o o o o             o o     o   o   o
o     o o o o   o   o o o   o o       o o o o o o o o   o o
o o o o o   o o   o     o   o o       o     o o o o o o o
o o   o   o o o       o o o   o   o o   o o o o o   o     o
o o       o   o o o   o o o o   o o o     o o o   o   o   o
    o   o o     o   o o   o o   o   o o o o o o o   o o   o
o o o o     o o         o         o         o o o       o
o   o       o o o   o     o o   o       o o o     o o o   o
o     o   o o o o o       o   o o   o   o o     o       o
o   o     o         o   o o o o o o o   o o o       o o
o       o o     o o o o   o o     o o   o     o o o o o o o
o o o o o o     o o   o o   o o   o   o   o o   o o o     o
    o   o   o o   o o o o   o o o o o   o   o o   o o o
o o   o o   o o   o o   o o   o       o       o o o   o
  o   o o o o   o o     o         o o     o o   o o o o o o
    o o   o o   o o     o   o o   o   o o         o   o o o
o     o     o     o o o   o     o o       o o       o o o o
o o o o o   o o o       o o   o   o o o o o o o o     o   o
  o o o o o   o o o o o   o o   o   o o     o o o     o o
      o o o o o     o o o   o   o o o   o o o   o   o     o
o   o o   o   o o o o o o o   o o o             o o o o o o
o o o o o   o   o         o o o o       o   o o o o   o   E
Options:
1.Print the path
2.Generate another puzzle
3.Exit the game
Enter your choice(1/2/3): 1

No solution found.
```