1. Front Page Display

```python
import time
import os

def display_name(name):
    os.system('cls' if os.name == 'nt' else 'clear')  # Clear the screen
    for letter in name:
        print(letter, end='', flush=True)
        time.sleep(0.5)  # Wait for 0.5 seconds between letters
    print("\nProject Name")
    time.sleep(1)  # Wait before moving to the next page
    input("Press Enter to continue...")

display_name("\nMehmet")
```

[3] Python

```
Mehmet
Project Name
```

Mehmet_AI_Project.ipynb > 1. Front Page Display

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.

2. User Menu

```python
def display_menu():
    print("\nUser Menu:")
    print("1. Signup")
    print("2. Login")
    print("3. Exit")

def signup():
    with open("users.txt", "a") as file:
        name = input("Enter your name: ")
        password = input("Enter your password: ")
        education = input("Enter your education: ")
        university = input("Enter your university: ")
        company = input("Enter the company you worked for: ")
        file.write(f"{name},{password},{education},{university},{company}\n")
    print("Signup successful!")

def login():
    users = {}
    with open("users.txt", "r") as file:
        for line in file:
            name, password, *rest = line.strip().split(',')
            users[name] = password

    while True:
        name = input("Enter your name: ")
        password = input("Enter your password: ")
        if users.get(name) == password:
            print("Login successful!")
```

Mehmet_AI_Project.ipynb > 1. Front Page Display

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python 3.12.3

```
            print("Login successful!")
            break
        else:
            print("Incorrect name or password. Try again.")

def main():
    while True:
        display_menu()
        choice = input("Select an option: ")
        if choice == "1":
            signup()
        elif choice == "2":
            login()
        elif choice == "3":
            print("Thank you for using my system!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

[ ] Python

+ Code + Markdown

3. Display User Information

```
import pyttsx3

def display_user_info(name):
    print(f"User Name: {name}")
    engine = pyttsx3.init()
    engine.say(name)
    engine.runAndWait()
```

7] Python

4. Plagiarism Analyzer

```
def plagiarism_analyzer(file1, file2):
    with open(file1, 'r') as f1, open(file2, 'r') as f2:
        text1 = f1.read()
        text2 = f2.read()

    if text1 == text2:
        print("Files are identical. Plagiarism detected!")
    else:
        print("Files are different. No plagiarism detected.")
```

Mehmet_AI_Project.ipynb > 1. Front Page Display

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python 3.12

4. Plagiarism Analyzer

```
def plagiarism_analyzer(file1, file2):
    with open(file1, 'r') as f1, open(file2, 'r') as f2:
        text1 = f1.read()
        text2 = f2.read()

    if text1 == text2:
        print("Files are identical. Plagiarism detected!")
    else:
        print("Files are different. No plagiarism detected.")
```

[8]

5. Sudoku Game

```
def print_sudoku(grid):
    for row in grid:
        print(" ".join(str(cell) for cell in row))

def sudoku_game():
    grid = [
        [5, 3, 0, 0, 7, 0, 0, 0, 0],
        [6, 0, 0, 1, 9, 5, 0, 0, 0],
        [0, 9, 8, 0, 0, 0, 0, 6, 0],
        [8, 0, 0, 0, 6, 0, 0, 0, 3],
```

Mehmet_AI_Project.ipynb | internship_assigment5.ipynb | internship_assigment4.ipynb

Mehmet_AI_Project.ipynb > 1. Front Page Display

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python 3.1

```
        [6, 0, 0, 1, 9, 5, 0, 0, 0],
        [0, 9, 8, 0, 0, 0, 0, 6, 0],
        [8, 0, 0, 0, 6, 0, 0, 0, 3],
        [4, 0, 0, 8, 0, 3, 0, 0, 1],
        [7, 0, 0, 0, 2, 0, 0, 0, 6],
        [0, 6, 0, 0, 0, 0, 2, 8, 0],
        [0, 0, 0, 4, 1, 9, 0, 0, 5],
        [0, 0, 0, 0, 8, 0, 0, 7, 9]
    ]
    print_sudoku(grid)
```

[9]

6. Calendar and Current Date

```
import calendar
from datetime import datetime

def display_calendar_and_date():
    year = 2024
    print(calendar.calendar(year))
    print(f"Current Date: {datetime.now().strftime('%Y-%m-%d')}")
```

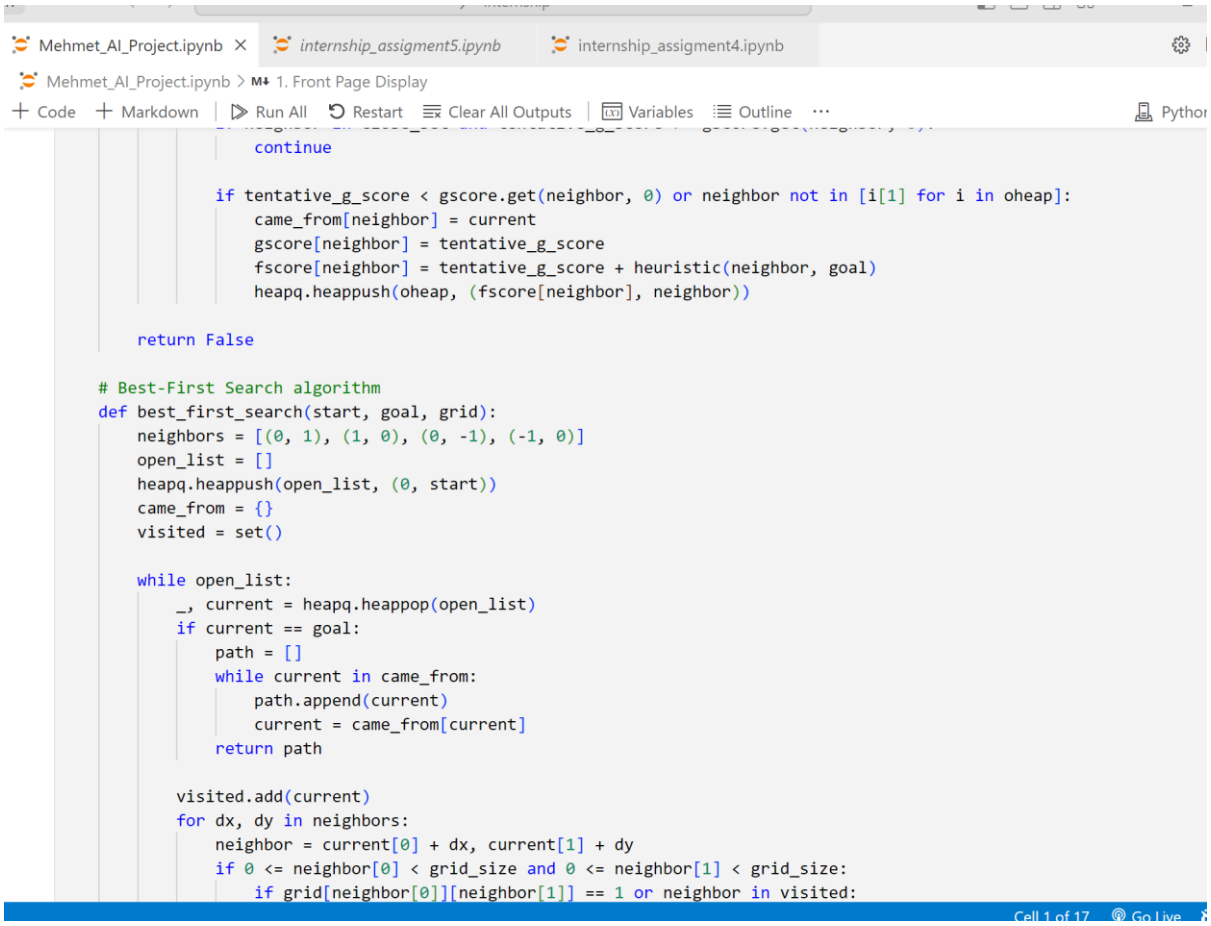
[10]

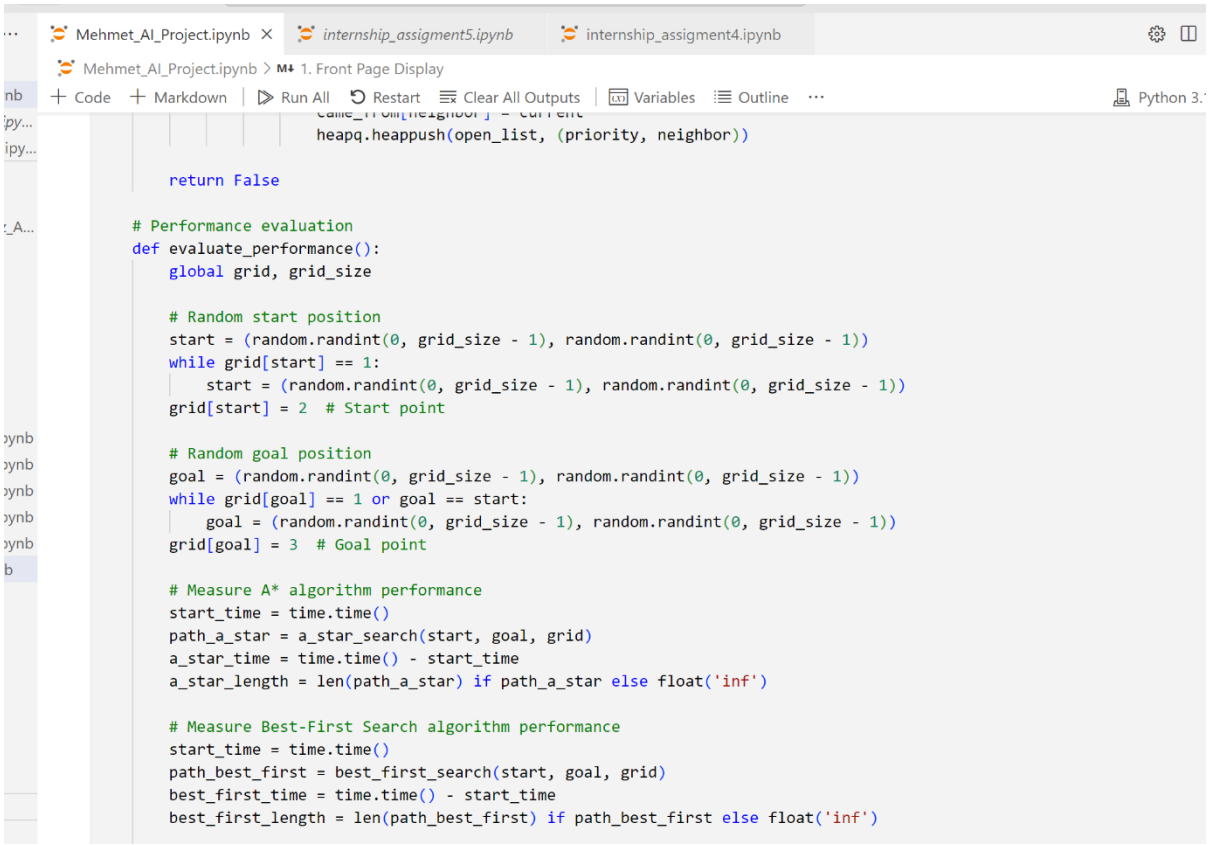
7. Autonomous Delivery Robot

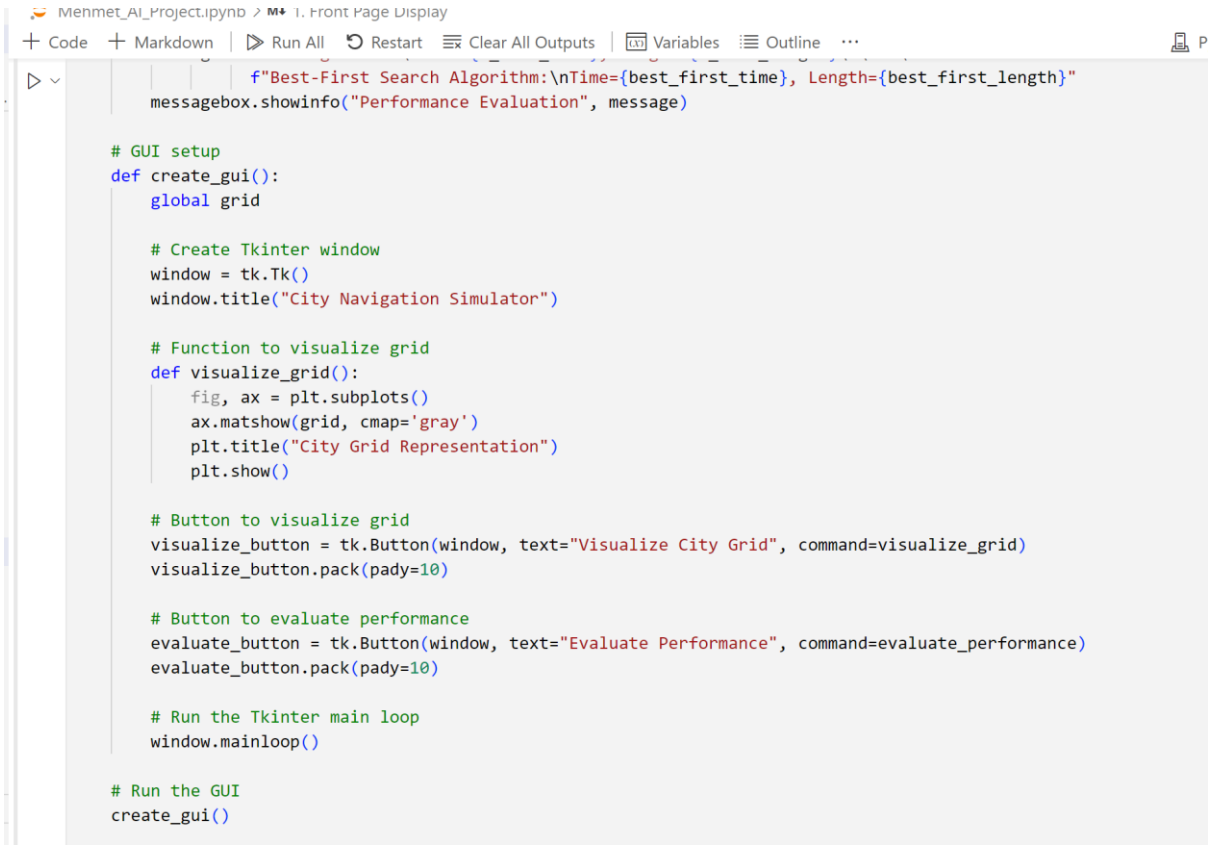
```
import numpy as np
import matplotlib.pyplot as plt
import random
import heapq
import math
import time
import tkinter as tk
from tkinter import messagebox

# Global variables
grid_size = 15
grid = np.zeros((grid_size, grid_size))

# Create obstacles (buildings, houses, vehicles)
obstacle_percentage = 0.2
for _ in range(int(grid_size * grid_size * obstacle_percentage)):
    x, y = random.randint(0, grid_size - 1), random.randint(0, grid_size - 1)
    grid[x, y] = 1

# Start position
start = (0, 0)
grid[start] = 2  # Start point

# Euclidean distance heuristic function
def heuristic(a, b):
    return math.sqrt((a[0] - b[0]) ** 2 + (a[1] - b[1]) ** 2)

# A* algorithm
```



```
# A* algorithm
def a_star_search(start, goal, grid):
    neighbors = [(0, 1), (1, 0), (0, -1), (-1, 0)]  # Right, down, left, up
    close_set = set()
    came_from = {}
    gscore = {start: 0}
    fscore = {start: heuristic(start, goal)}
    oheap = []

    heapq.heappush(oheap, (fscore[start], start))

    while oheap:
        current = heapq.heappop(oheap)[1]

        if current == goal:
            data = []
            while current in came_from:
                data.append(current)
                current = came_from[current]
            return data

        close_set.add(current)
        for i, j in neighbors:
            neighbor = current[0] + i, current[1] + j
            tentative_g_score = gscore[current] + 1
            if 0 <= neighbor[0] < grid_size:
                if 0 <= neighbor[1] < grid_size:
                    if grid[neighbor[0]][neighbor[1]] == 1:
                        continue
                else:
                    continue
            else:
                continue
```


Mehmet_AI_Project.ipynb
internship_assigment5.ipynb
internship_assigment4.ipynb
Mehmet_AI_Project.ipynb > 1. Front Page Display
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Python
continue
if tentative_g_score < gscore.get(neighbor, 0) or neighbor not in [i[1] for i in oheap]:
came_from[neighbor] = current
gscore[neighbor] = tentative_g_score
fscore[neighbor] = tentative_g_score + heuristic(neighbor, goal)
heapq.heappush(oheap, (fscore[neighbor], neighbor))
return False
# Best-First Search algorithm
def best_first_search(start, goal, grid):
neighbors = [(0, 1), (1, 0), (0, -1), (-1, 0)]
open_list = []
heapq.heappush(open_list, (0, start))
came_from = {}
visited = set()
while open_list:
_, current = heapq.heappop(open_list)
if current == goal:
path = []
while current in came_from:
path.append(current)
current = came_from[current]
return path
visited.add(current)
for dx, dy in neighbors:
neighbor = current[0] + dx, current[1] + dy
if 0 <= neighbor[0] < grid_size and 0 <= neighbor[1] < grid_size:
if grid[neighbor[0]][neighbor[1]] == 1 or neighbor in visited:
Cell 1 of 17
Go Live


Mehmet_AI_Project.ipynb
internship_assigment5.ipynb
internship_assigment4.ipynb
Mehmet_AI_Project.ipynb > 1. Front Page Display
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Python 3.
heapq.heappush(open_list, (priority, neighbor))
return False
# Performance evaluation
def evaluate_performance():
global grid, grid_size
# Random start position
start = (random.randint(0, grid_size - 1), random.randint(0, grid_size - 1))
while grid[start] == 1:
start = (random.randint(0, grid_size - 1), random.randint(0, grid_size - 1))
grid[start] = 2 # Start point
# Random goal position
goal = (random.randint(0, grid_size - 1), random.randint(0, grid_size - 1))
while grid[goal] == 1 or goal == start:
goal = (random.randint(0, grid_size - 1), random.randint(0, grid_size - 1))
grid[goal] = 3 # Goal point
# Measure A* algorithm performance
start_time = time.time()
path_a_star = a_star_search(start, goal, grid)
a_star_time = time.time() - start_time
a_star_length = len(path_a_star) if path_a_star else float('inf')
# Measure Best-First Search algorithm performance
start_time = time.time()
path_best_first = best_first_search(start, goal, grid)
best_first_time = time.time() - start_time
best_first_length = len(path_best_first) if path_best_first else float('inf')


Mehmet_AI_Project.ipynb > 1. Front Page Display
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
f"Best-First Search Algorithm:\nTime={best_first_time}, Length={best_first_length}"
messagebox.showinfo("Performance Evaluation", message)
# GUI setup
def create_gui():
global grid
# Create Tkinter window
window = tk.Tk()
window.title("City Navigation Simulator")
# Function to visualize grid
def visualize_grid():
fig, ax = plt.subplots()
ax.matshow(grid, cmap='gray')
plt.title("City Grid Representation")
plt.show()
# Button to visualize grid
visualize_button = tk.Button(window, text="Visualize City Grid", command=visualize_grid)
visualize_button.pack(pady=10)
# Button to evaluate performance
evaluate_button = tk.Button(window, text="Evaluate Performance", command=evaluate_performance)
evaluate_button.pack(pady=10)
# Run the Tkinter main loop
window.mainloop()
# Run the GUI
create_gui()


Mehmet_AI_Project.ipynb
internship_assigment5.ipynb
internship_assigment4.ipynb
Mehmet_AI_Project.ipynb > 1. Front Page Display
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Pyth
def chat_gpt():
while True:
user_input = input("You: ")
if user_input.lower() in ['exit', 'quit']:
break
# Simple predefined responses
if "hello" in user_input.lower():
print("Bot: Hi there!")
else:
print("Bot: I'm not sure how to respond to that.")
[12]
[ ]