ABBOTTABAD UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Computer Science

NAME: MUHAMMAD HASEEB **CLASS:** BSCS

ROLL NO: 14861 **Sir:** JAMAL ABDUAL AHUD

SUBJECT: DSA **DATE:** 24-12-2024

PROJECT PROPOSAL

DSA

1. Introduction

1.1 Purpose

This document describes the design, functionality, and implementation of a Python-based traffic simulation system. The primary goal of this project is to simulate the movement of vehicles through intersections controlled by traffic lights, showcasing fundamental concepts of data structures, object-oriented programming, and real-time system simulation.

1.2 Document Conventions

- Code snippets are written in Python.
- Classes and methods are highlighted in bold.
- Simulations are explained using a step-by-step approach.

1.3 Project Scope

1.3.1 Scope Definition

The traffic simulation models a simple traffic network with intersections and roads. Vehicles traverse the network based on traffic light states at intersections, providing insights into the behavior of traffic systems under controlled conditions.

1.3.2 Core Features

- Simulation of intersections with configurable traffic lights.
- Representation of vehicles with movement logic.
- Dynamic toggling of traffic light states.
- Visualization through console-based output.

1.3.3 Subsequent Releases

Future iterations may include:

- Graphical user interface for real-time visualization.
- Adaptive traffic lights based on vehicle density.
- Multi-lane roads and enhanced vehicle behavior.

1.3.4 Alignment with User and Business Goals

This project serves as an educational tool for understanding traffic systems and provides a foundation for more complex traffic simulations in academic and professional environments.

1.4 References

- Python Official Documentation
- Object-Oriented Programming Principles.
- Traffic Engineering Research Papers.

2. Overall Description

2.1 Product Perspective

2.1.1 Product Context

This project is a standalone simulation designed for educational purposes, demonstrating traffic flow management through intersections.

2.1.2 Product Origin

The simulation was developed as a project to explore the use of object-oriented programming and real-time system simulation in Python.

2.1.3 Product Relationship to Existing Systems

The system is independent and does not rely on external traffic systems but can integrate with graphical libraries or real-world data in future enhancements.

2.1.4 Product Ecosystem

The simulation forms the foundation for advanced traffic management systems and can be extended with AI algorithms for smart traffic control.

2.2 User Classes and Characteristics

2.2.1 Tech Enthusiasts

Users interested in programming and simulations will find this project engaging and educational.

2.2.2 Academics and Researchers

Provides a base model for studying traffic flow and designing intelligent traffic systems.

2.2.3 Favored User Class

Beginner to intermediate programmers seeking hands-on experience with simulations.

2.2.4 Alignment with User Needs

The simulation’s simple structure and extendable design meet the learning and research needs of its users.

2.3 Operating Environment

2.3.1 Hardware Platform

The simulation runs on any standard computer with Python installed.

2.3.2 Operating Systems and Versions

Compatible with Windows, macOS, and Linux systems with Python 3.x.

2.4 Design and Implementation Constraints

2.4.1 Programming Language

The project is implemented in Python.

2.4.2 Database Technology

No database is used; the system relies on in-memory structures for simulation.

2.4.3 Third-Party Integrations

No third-party libraries are required, but future versions may incorporate visualization tools.

2.4.4 User Interface Design

A console-based interface displays the state of the simulation, with future potential for graphical interfaces.

2.5 Assumptions and Dependencies

2.5.1 Assumptions

- Traffic lights toggle between red and green.
- Vehicles wait at red lights and proceed on green.

2.5.2 Dependencies

- Python runtime environment.
- Basic understanding of traffic systems for meaningful use.

3. System Features

- **Dynamic Traffic Lights:** Each intersection has traffic lights that alternate between red and green.
- **Vehicle Movement:** Vehicles move through intersections, waiting at red lights and proceeding on green.
- **Simulation Loop:** Time steps simulate real-time behavior, toggling lights and moving vehicles.
- **Extendibility:** New features such as adaptive traffic systems or graphical interfaces can be added. For instance, a diagram could illustrate how adaptive traffic systems integrate with existing components. A potential table might show:

Feature	**Description**	**Impact**
Adaptive Traffic Lights	AI-controlled light timings	Improves traffic flow
Graphical Interface	Real-time graphical visualization	Enhances user experience

Multi-lane Roads	Support for multiple vehicle lanes	Increases simulation realism

5. External Interface Requirements

5.1 User Interfaces

5.1.1 Design Standards and Guidelines

The interface follows a simple console-based output for clarity and ease of use.

5.1.2 Screen Layout and Resolution

Information is displayed in a terminal, showing the current state of intersections and vehicles.

5.1.3 Standard Interface Elements

- Time step counter.
- Vehicle status (waiting or moving).
- Traffic light state (red or green).

5.2 Software Interfaces

5.2.1 Non-Functional Requirements

- **Performance:** Simulates traffic efficiently with minimal computational overhead.
- **Scalability:** Supports adding more intersections and vehicles with minor adjustments.

5.3 Hardware Interfaces

5.3.1 Supported Device Types

Runs on any computer capable of running Python 3.x.

6. Quality Attributes

6.1 Performance

The simulation runs efficiently with low memory and CPU usage.

6.2 Reliability

Traffic light and vehicle behaviors are deterministic and predictable.

6.3 Usability

The simple design and output make the simulation easy to understand and extend.

6.4 Security

No external data or network connectivity ensures a secure simulation environment.

6.5 Maintainability

The modular design of classes (Intersection, Traffic Light, Vehicle, Traffic Simulation) simplifies maintenance and enhancement.

Appendix B: Analysis Model

- **Vehicle Behavior:** Vehicles follow a straightforward logic: wait at red lights, proceed on green.
- **Traffic Light Control:** Lights alternate states every fixed number of steps.
- **System Flow:**
    1. Initialize intersections, roads, and traffic lights.
    2. Add vehicles and set their destinations.
    3. Run the simulation loop for the desired time steps.
    4. Output the state of the system at each step.

Appendix B: Analysis Model

Use Case Diagram

A **Use Case Diagram** is a helpful tool for illustrating the interactions between users (or systems) and the traffic simulation system. It highlights the key actions and functionalities that users can perform, providing a clear view of the system's operations from a high-level perspective.

Here's a conceptual description of the **Use Case Diagram** for the traffic simulation system:

Actors:

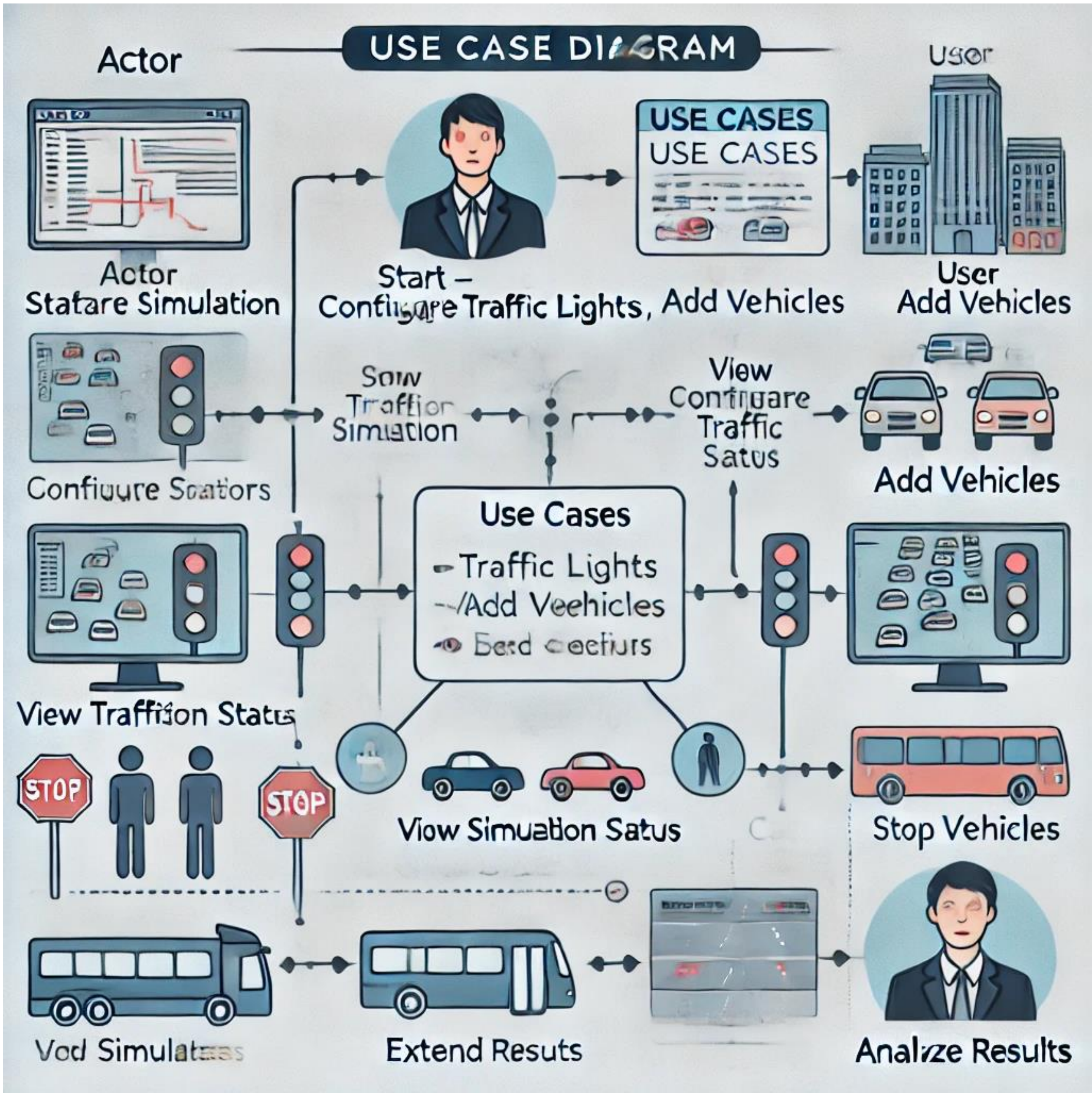
1. **User (Programmer/Researcher/Enthusiast)**: The primary actor who interacts with the traffic simulation system.
2. **Traffic Simulation System**: The system that controls the traffic lights, vehicle movement, and manages the simulation logic.

Use Cases:

1. **Start Simulation**: The user can initiate the traffic simulation, starting the process where traffic lights toggle and vehicles move.
2. **Configure Traffic Lights**: The user can adjust the behavior of traffic lights, such as their switching intervals or states (e.g., red/green).
3. **Add Vehicles**: The user can add vehicles to the simulation, specifying their entry points and destinations.
4. **View Simulation Status**: The system displays real-time information on the status of traffic lights, vehicle movement, and simulation time steps.
5. **Stop Simulation**: The user can stop the simulation once it has run for a desired duration.
6. **Extend Simulation**: The user can modify or extend the simulation to add features like multi-lane roads, adaptive traffic lights, or a graphical interface.
7. **Analyze Results**: The user can analyze traffic flow patterns and behavior to gather insights for academic or research purposes.

Appendix B: Analysis Model

Use Case diagram:



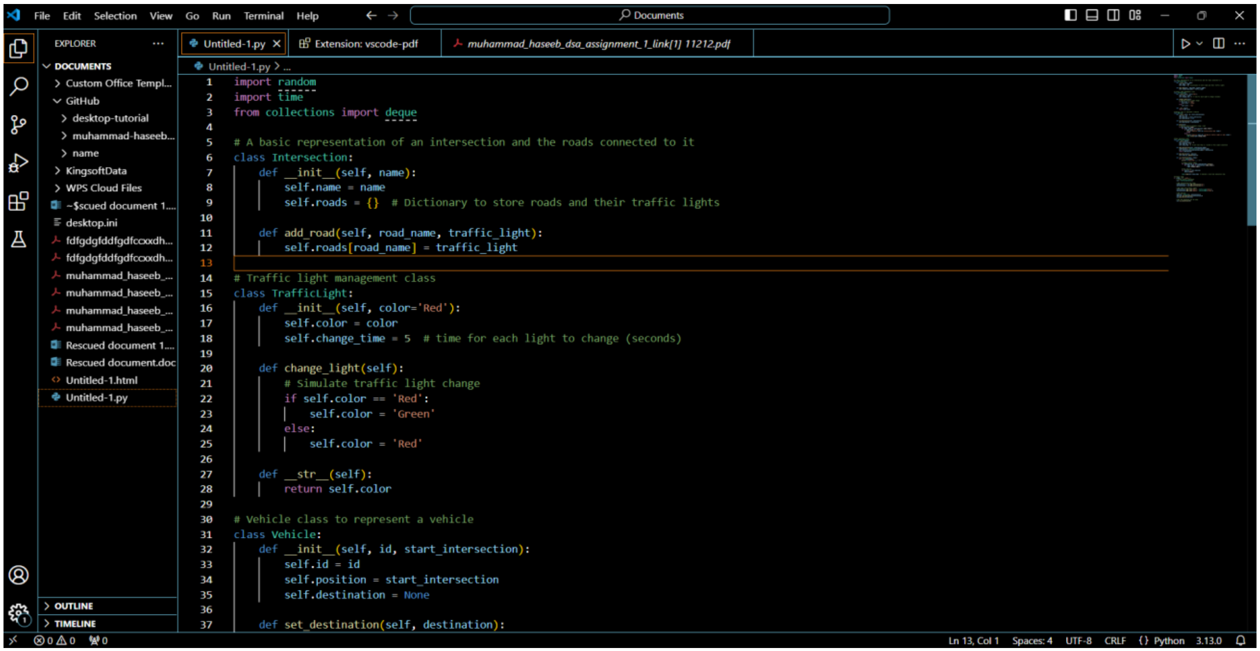
DFD DIAGRAM LEVEL (0)

PROJECT CODE:

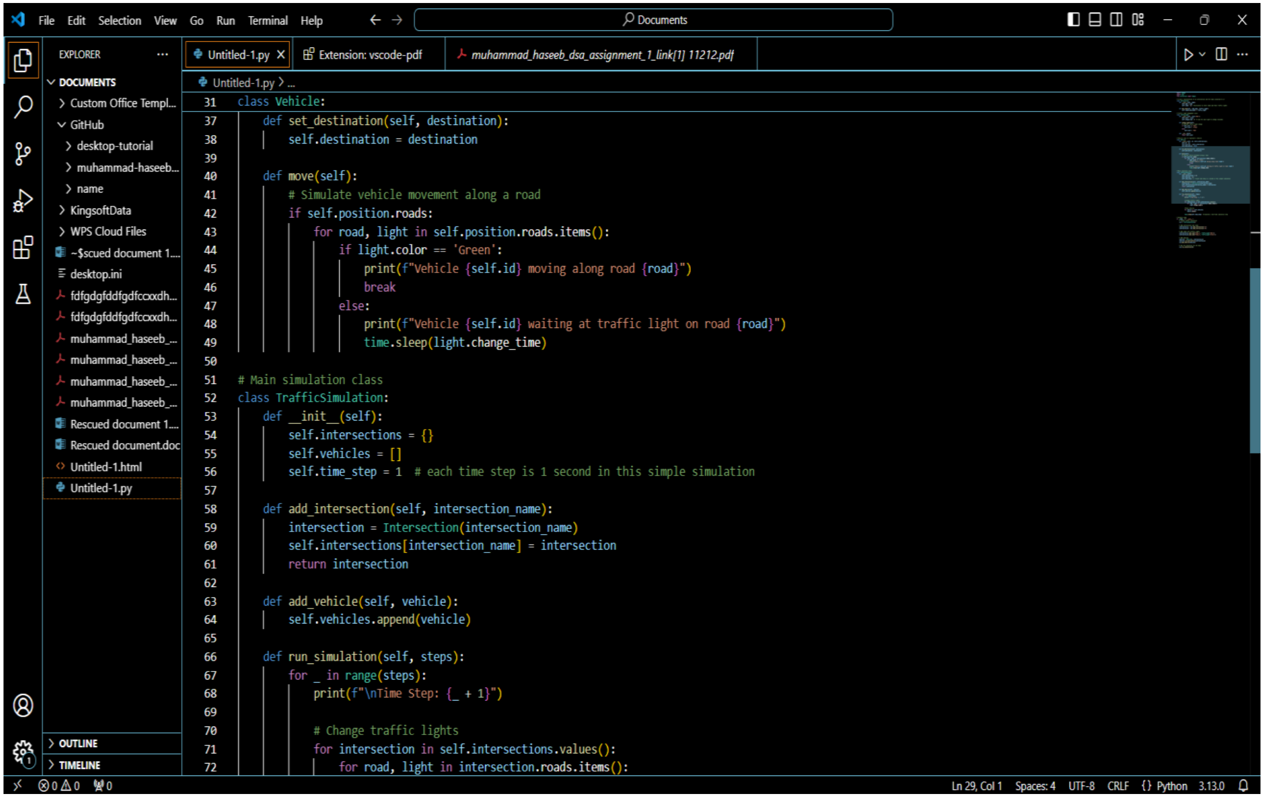
```
import random
import time
from collections import deque

# A basic representation of an intersection and the roads connected to it
class Intersection:
    def __init__(self, name):
        self.name = name
        self.roads = {}  # Dictionary to store roads and their traffic lights

    def add_road(self, road_name, traffic_light):
        self.roads[road_name] = traffic_light

# Traffic light management class
class TrafficLight:
    def __init__(self, color='Red'):
        self.color = color
        self.change_time = 5  # time for each light to change (seconds)

    def change_light(self):
        # Simulate traffic light change
        if self.color == 'Red':
            self.color = 'Green'
        else:
            self.color = 'Red'

    def __str__(self):
        return self.color

# Vehicle class to represent a vehicle
class Vehicle:
    def __init__(self, id, start_intersection):
        self.id = id
        self.position = start_intersection
        self.destination = None

    def set_destination(self, destination):
        self.destination = destination

    def move(self):
        # Simulate vehicle movement along a road
        if self.position.roads:
            for road, light in self.position.roads.items():
                if light.color == 'Green':
                    print(f"Vehicle {self.id} moving along road {road}")
                    break
                else:
                    print(f"Vehicle {self.id} waiting at traffic light on road {road}")
                    time.sleep(light.change_time)

# Main simulation class
```

```
class TrafficSimulation:
    def __init__(self):
        self.intersections = {}
        self.vehicles = []
        self.time_step = 1  # each time step is 1 second in this simple simulation

    def add_intersection(self, intersection_name):
        intersection = Intersection(intersection_name)
        self.intersections[intersection_name] = intersection
        return intersection

    def add_vehicle(self, vehicle):
        self.vehicles.append(vehicle)

    def run_simulation(self, steps):
        for _ in range(steps):
            print(f"\nTime Step: {_ + 1}")

            # Change traffic lights
            for intersection in self.intersections.values():
                for road, light in intersection.roads.items():
                    light.change_light()

            # Move vehicles
            for vehicle in self.vehicles:
                vehicle.move()
            time.sleep(self.time_step)  # Simulate a real-time simulation step

# Example usage
if __name__ == "__main__":
    # Create traffic simulation
    sim = TrafficSimulation()

    # Add intersections and roads
    intersection_A = sim.add_intersection('A')
    intersection_B = sim.add_intersection('B')

    # Add roads with traffic lights
    intersection_A.add_road('Road_1', TrafficLight('Red'))
    intersection_B.add_road('Road_2', TrafficLight('Green'))

    # Add vehicles
    vehicle1 = Vehicle(1, intersection_A)
    vehicle1.set_destination(intersection_B)
    sim.add_vehicle(vehicle1)

    # Run the simulation for 10 steps
    sim.run_simulation(10)
```

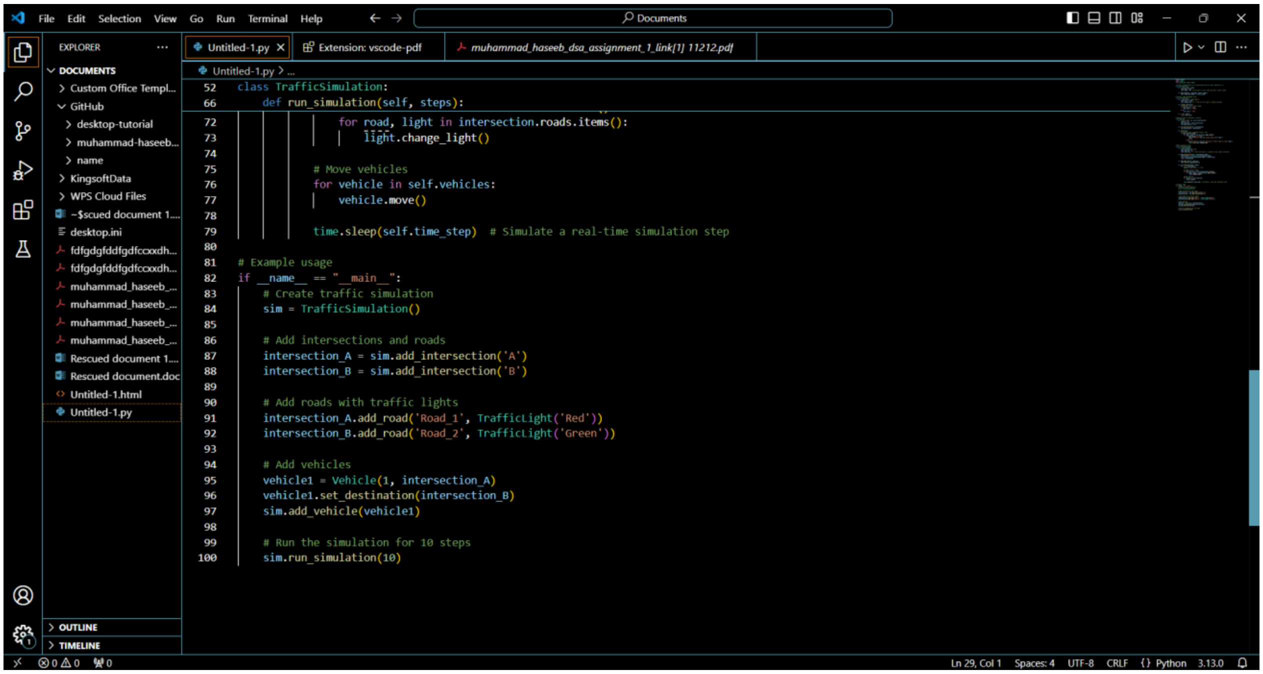
CODE PICTURE:



```python
import random
import time
from collections import deque

# A basic representation of an intersection and the roads connected to it
class Intersection:
    def __init__(self, name):
        self.name = name
        self.roads = {}  # Dictionary to store roads and their traffic lights

    def add_road(self, road_name, traffic_light):
        self.roads[road_name] = traffic_light

# Traffic light management class
class TrafficLight:
    def __init__(self, color='Red'):
        self.color = color
        self.change_time = 5  # time for each light to change (seconds)

    def change_light(self):
        # Simulate traffic light change
        if self.color == 'Red':
            self.color = 'Green'
        else:
            self.color = 'Red'

    def __str__(self):
        return self.color

# Vehicle class to represent a vehicle
class Vehicle:
    def __init__(self, id, start_intersection):
        self.id = id
        self.position = start_intersection
        self.destination = None

    def set_destination(self, destination):
```

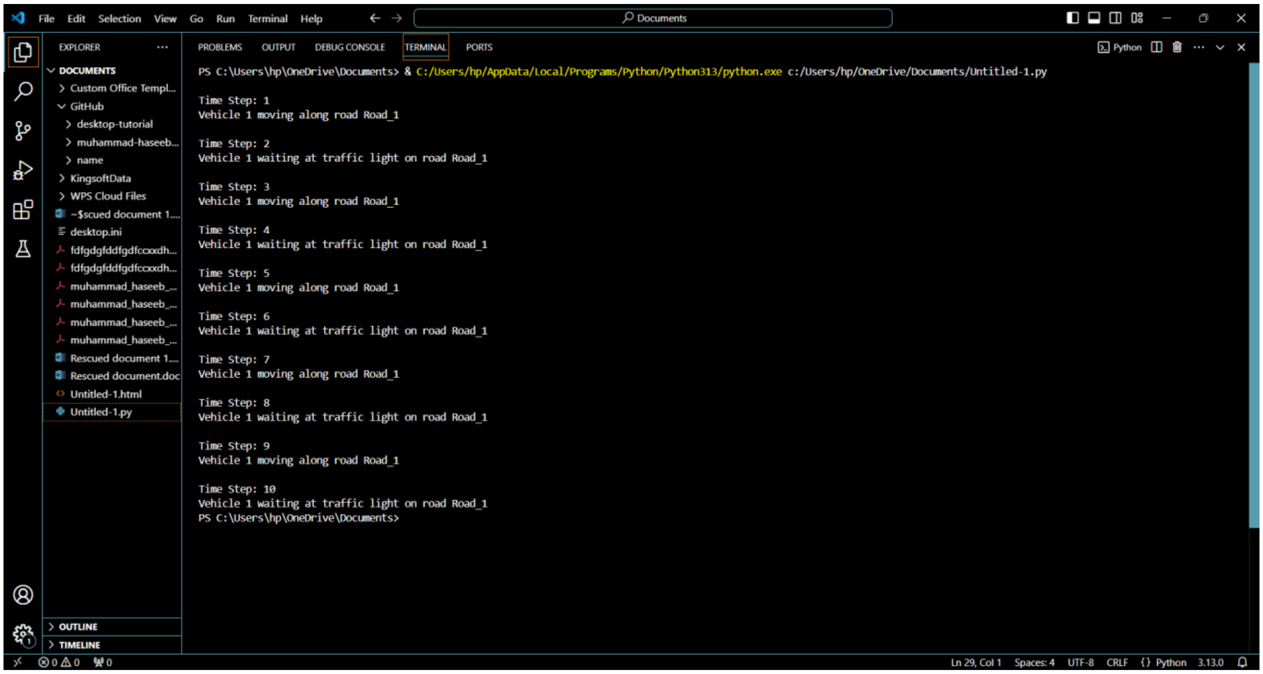


```python
class Vehicle:
    def set_destination(self, destination):
        self.destination = destination

    def move(self):
        # Simulate vehicle movement along a road
        if self.position.roads:
            for road, light in self.position.roads.items():
                if light.color == 'Green':
                    print(f"Vehicle {self.id} moving along road {road}")
                    break
                else:
                    print(f"Vehicle {self.id} waiting at traffic light on road {road}")
                    time.sleep(light.change_time)

# Main simulation class
class TrafficSimulation:
    def __init__(self):
        self.intersections = {}
        self.vehicles = []
        self.time_step = 1  # each time step is 1 second in this simple simulation

    def add_intersection(self, intersection_name):
        intersection = Intersection(intersection_name)
        self.intersections[intersection_name] = intersection
        return intersection

    def add_vehicle(self, vehicle):
        self.vehicles.append(vehicle)

    def run_simulation(self, steps):
        for _ in range(steps):
            print(f"\nTime Step: {_ + 1}")

            # Change traffic lights
            for intersection in self.intersections.values():
                for road, light in intersection.roads.items():
```



```python
class TrafficSimulation:
    def run_simulation(self, steps):
                for road, light in intersection.roads.items():
                    light.change_light()

            # Move vehicles
            for vehicle in self.vehicles:
                vehicle.move()

            time.sleep(self.time_step)  # Simulate a real-time simulation step

# Example usage
if __name__ == "__main__":
    # Create traffic simulation
    sim = TrafficSimulation()

    # Add intersections and roads
    intersection_A = sim.add_intersection('A')
    intersection_B = sim.add_intersection('B')

    # Add roads with traffic lights
    intersection_A.add_road('Road_1', TrafficLight('Red'))
    intersection_B.add_road('Road_2', TrafficLight('Green'))

    # Add vehicles
    vehicle1 = Vehicle(1, intersection_A)
    vehicle1.set_destination(intersection_B)
    sim.add_vehicle(vehicle1)

    # Run the simulation for 10 steps
    sim.run_simulation(10)
```

OUT PUT:


File Edit Selection View Go Run Terminal Help
Documents
EXPLORER
DOCUMENTS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
OUTLINE
TIMELINE
Ln 29, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.0

```
PS C:\Users\hp\OneDrive\Documents> & C:/Users/hp/AppData/Local/Programs/Python/Python313/python.exe c:/Users/hp/OneDrive/Documents/Untitled-1.py

Time Step: 1
Vehicle 1 moving along road Road_1

Time Step: 2
Vehicle 1 waiting at traffic light on road Road_1

Time Step: 3
Vehicle 1 moving along road Road_1

Time Step: 4
Vehicle 1 waiting at traffic light on road Road_1

Time Step: 5
Vehicle 1 moving along road Road_1

Time Step: 6
Vehicle 1 waiting at traffic light on road Road_1

Time Step: 7
Vehicle 1 moving along road Road_1

Time Step: 8
Vehicle 1 waiting at traffic light on road Road_1

Time Step: 9
Vehicle 1 moving along road Road_1

Time Step: 10
Vehicle 1 waiting at traffic light on road Road_1
PS C:\Users\hp\OneDrive\Documents>
```