Artificial Intelligence Real-World Simulation Project

Group Size: Maximum of 5 members per group.
main course Techniques: Searching Algorithms, Reinforcement Learning.

Submission Requirements:

1. **Project Report (PDF):**
   - A detailed document discussing the following:
     - An explanation of the AI algorithms and tools used in the project.
     - **Results and Outcomes:** Present the outcomes of the project, including visualizations or metrics that demonstrate the effectiveness and efficiency of your solution.
2. **Source Code Submission:**
   - The source code for the project must be submitted in a `.zip` file, which should include all necessary files and instructions for running the project. The code should be well-organized, commented, and easily understandable.

---

1. Autonomous Ground Rover for Terrain Navigation

Objective: Develop a simulation where a ground rover must navigate through complex and uneven terrain, such as forests, deserts, or rocky environments, to reach a target destination.

Description:

- **Simulation Setup**: The rover is equipped with various sensors (e.g., LiDAR, cameras, IMUs) to perceive its environment. It must make decisions on how to traverse obstacles such as rocks, mud, or steep slopes.
- **AI Techniques**:
  - **Reinforcement Learning (RL)**: Train the rover using RL algorithms, where it receives rewards for navigating difficult terrain efficiently and avoiding obstacles.
  - **Path Planning**: Implement algorithms like **A*** or **Dijkstra's Algorithm** for planning the optimal path across unknown environments.
  - **Simultaneous Localization and Mapping (SLAM)**: Use SLAM techniques to map the environment and localize the rover in real-time as it explores new areas.

2. Lunar Rover Autonomous Navigation with Obstacle Avoidance

- **Description**: This project focuses on simulating a lunar rover that autonomously navigates across the lunar surface while avoiding obstacles like rocks, craters, and other terrain features. The goal is for the rover to learn how to navigate the surface effectively, utilizing reinforcement learning (RL) to adapt its path as it encounters different obstacles.
- **Key Concepts**:
  - **Reinforcement Learning (RL)**: The rover learns by trial and error, using rewards for successful obstacle avoidance and efficient navigation.
  - **Path Planning**: Algorithms like A* or Dijkstra's algorithm can be used to find the optimal route in the simulated environment.
  - **Sensor Fusion**: Simulating how the rover's sensors (e.g., LiDAR, cameras, IMUs) can be used to gather real-time data for decision-making.
- **Steps to Implement**:
  - Use **OpenAI Gym** to create a custom simulation of the lunar surface.
  - Define state variables (e.g., rover position, distance to obstacles, heading angle).
  - Use RL algorithms like Q-learning or Deep Q Networks (DQN) to train the rover to avoid obstacles while navigating to a destination.
  - Implement terrain features like craters, rocks, and uneven surfaces.
  - Test the rover's ability to adapt to different terrains and avoid dynamic obstacles.
- **Simulation Platform**: **OpenAI Gym** (for creating the environment) or **Gazebo** (for more complex 3D simulations), **ROS** (Robot Operating System) for integrating sensors and control.

3. Lunar Rover Exploration for Data Collection

- **Description**: This project simulates a lunar rover that explores the lunar surface to collect data from various geological sites. The rover needs to decide where to go next to maximize scientific data collection (e.g., measuring soil properties, identifying interesting rock formations) while avoiding obstacles and conserving energy.
- **Key Concepts**:
  - **Reinforcement Learning**: The rover learns optimal exploration strategies based on scientific value.
  - **Task Prioritization**: The rover must decide which sites to explore first based on predefined criteria like geological interest.
  - **Energy Management**: The rover must monitor its battery level and optimize movement to conserve energy.
- **Steps to Implement**:
  - Create a simulated environment representing the lunar surface, with geological features marked as points of interest (POIs).
  - Implement an RL agent that explores these POIs, considering factors like scientific value and energy consumption.
  - Use algorithms to dynamically adjust the rover's exploration strategy based on data received from sensors (e.g., temperature, soil composition).
  - Add obstacle avoidance to ensure that the rover can avoid craters or large rocks in its path.

- **Simulation Platform**: **OpenAI Gym**, **Gazebo**, or **Unity** with ML-Agents for creating a detailed lunar exploration environment.

4. Space Debris Removal Using Robotic Arms (Autonomous Grasping and Manipulation)

- **Description**: Simulate a robotic spacecraft equipped with robotic arms to remove space debris from Earth's orbit. The robot needs to identify, approach, grasp, and safely de-orbit space debris like defunct satellites, pieces of rockets, and other objects that threaten operational satellites.
- **Key Concepts**:
    - **Reinforcement Learning**: The robot learns the optimal approach to grasp and manipulate various objects.
    - **Robotic Manipulation**: Learning how to control a robotic arm for precise grasping and de-orbiting tasks.
    - **Computer Vision**: Identifying space debris and understanding its position and orientation in space.
    - **Control Systems**: Ensuring precise movement and control of the robotic arm to avoid damaging both the debris and the spacecraft.
- **Steps to Implement**:
    - Create a simulated environment that represents a low-Earth orbit scenario, with various pieces of space debris in different positions.
    - Use a **robotic arm model** and control algorithms (PID, inverse kinematics) to grasp and move objects.
    - Implement RL algorithms to allow the robot to improve its grasping and manipulation strategies based on feedback.
    - Use **OpenCV** or another computer vision library to detect space debris and calculate the optimal grasping location.
    - Simulate the process of removing debris, ensuring that the robot can safely bring it back into the atmosphere for de-orbiting.
- **Simulation Platform**: **Gazebo**, **V-REP (CoppeliaSim)** for robotic arm simulation, or **OpenAI Gym** with custom environments for grasping tasks.

5. Autonomous Space Debris Removal with Multi-Agent Coordination

- **Description**: This project simulates multiple robotic spacecraft working together to remove space debris. Each agent (robot) must cooperate with others to safely remove larger or multiple debris objects. The system must use reinforcement learning to optimize team strategies, considering factors like collision avoidance, task allocation, and efficiency.
- **Key Concepts**:
    - **Multi-Agent Systems**: Coordination between multiple robotic agents to achieve a common goal.
    - **Reinforcement Learning**: Agents learn how to coordinate their actions (e.g., moving in unison, synchronizing tasks) to remove debris efficiently.

    - **Task Allocation**: The robots must decide who will handle which pieces of debris based on size, location, and removal strategy.
- **Steps to Implement**:
    - Develop a simulation with multiple robotic agents (each with an arm for debris manipulation).
    - Implement a multi-agent RL setup where each agent learns how to interact with others (e.g., avoid collisions, cooperate on debris removal).
    - Use reward signals that encourage cooperation, such as faster debris removal or minimal fuel consumption.
    - Simulate the dynamics of space debris removal, including potential failures or errors that require robots to adapt their strategies.
- **Simulation Platform**: **OpenAI Gym**, **Gazebo**, or **V-REP (CoppeliaSim)**, which supports multi-agent simulations.

6. Space Debris Collision Avoidance Using Reinforcement Learning

- **Description**: This project focuses on the collision avoidance behavior of a spacecraft tasked with removing space debris. The spacecraft must use sensors and real-time data to detect objects in its path and navigate around them. Reinforcement learning can be used to train the spacecraft to avoid obstacles and safely approach debris.
- **Key Concepts**:
    - **Reinforcement Learning**: The spacecraft learns how to avoid collisions with space debris or other satellites.
    - **Collision Detection**: Simulating how the spacecraft detects nearby objects using its sensors (e.g., radar, lidar).
    - **Path Planning**: Navigating around debris in a dynamic environment.
- **Steps to Implement**:
    - Create a simulation that involves a spacecraft navigating through space with various debris objects.
    - Implement RL algorithms to optimize collision avoidance strategies based on real-time sensor data.
    - Include a reward system that penalizes collisions and rewards safe passage to or removal of space debris.
- **Simulation Platform**: **OpenAI Gym**, **Gazebo**, or **Unity** (for spacecraft movement and pathfinding).

7. Lunar Rover Collaborative Exploration with Multiple Rovers

- **Description**: In this project, multiple lunar rovers work together to explore a designated area of the Moon’s surface. The rovers must coordinate their movements, share data about terrain conditions, and decide which rover should explore which area to maximize overall coverage.
- **Key Concepts**: Multi-Agent Systems, Reinforcement Learning, Collaborative Path Planning.
- **Steps to Implement**:

    - Use multi-agent reinforcement learning to allow rovers to communicate and cooperate.
    - Implement path planning algorithms like A* or Dijkstra's for individual rovers.
    - Simulate communication between rovers to share mapping data, avoiding overlap in exploration.
- **Simulation Platform**: **Gazebo**, **ROS**, **OpenAI Gym**.

8. Lunar Surface Hazard Detection and Avoidance System

- **Description**: Simulate a lunar rover equipped with sensors (e.g., LiDAR, cameras) that can detect potential hazards such as craters, rocks, or steep slopes. The rover will use computer vision and reinforcement learning to identify and avoid hazards while planning its path.
- **Key Concepts**: Computer Vision, Reinforcement Learning, Obstacle Avoidance, Sensor Fusion.
- **Steps to Implement**:
    - Train a deep learning model to detect hazards in images from the rover's sensors.
    - Implement RL to allow the rover to adjust its path based on sensor data to avoid detected hazards.
    - Incorporate various terrains to test the rover's adaptability.
- **Simulation Platform**: **OpenCV** for computer vision, **OpenAI Gym**, **Gazebo**.

9. Lunar Rover Dynamic Path Planning for Energy Efficiency

- **Description**: Design an autonomous lunar rover that uses dynamic path planning to optimize its energy usage. The rover should navigate across the lunar surface while considering the energy cost of movement on different terrain types (e.g., rocky vs. flat terrain) and adjusting its path accordingly.
- **Key Concepts**: Path Planning, Reinforcement Learning, Energy Efficiency, Resource Management.
- **Steps to Implement**:
    - Simulate a lunar environment with varying terrains, each with a different energy cost to cross.
    - Implement a RL agent to plan a path that minimizes energy consumption while maximizing efficiency.
    - Include variables like battery levels and terrain types to make decisions about when to recharge or avoid energy-draining paths.
- **Simulation Platform**: **OpenAI Gym**, **Gazebo**, **ROS**.

10. Lunar Excavation for Resource Extraction (Mining Simulation)

- **Description**: Simulate a lunar rover designed for excavation and resource extraction, such as mining for water ice or minerals. The rover should autonomously select excavation sites, dig into the lunar surface, and store extracted materials.
- **Key Concepts**: Reinforcement Learning, Robotics, Task Planning, Autonomous Excavation.

- **Steps to Implement**:
  - Develop a system that simulates the rover's excavation tools and resource extraction tasks.
  - Implement RL for the rover to autonomously select optimal excavation sites based on geological data or mission objectives.
  - Model environmental factors (e.g., the strength of lunar soil, dust clouds).
- **Simulation Platform**: **Gazebo**, **ROS**, **OpenAI Gym**.

11. Lunar Rover Fault Detection and Self-Repair System

- **Description**: Simulate a lunar rover that is equipped with the ability to detect faults (e.g., damaged wheels, malfunctioning sensors) and perform basic self-repairs or adjustments to continue its mission.
- **Key Concepts**: Fault Detection, Reinforcement Learning, Robotics, Autonomous Maintenance.
- **Steps to Implement**:
  - Implement a fault detection system using sensor data to identify when components are malfunctioning.
  - Train a reinforcement learning agent to take corrective actions (e.g., adjusting speed or maneuvering to avoid damage).
  - Simulate self-repair actions, such as using the rover’s tools to fix broken parts.
- **Simulation Platform**: **Gazebo**, **ROS**, **OpenAI Gym**.

12. Autonomous Mars Rover Exploration with Environmental Adaptation

- **Description**: Similar to lunar rover exploration, this project involves simulating an autonomous rover exploring the surface of Mars. The rover must adapt to varying terrain conditions, identify different types of rocks, and autonomously select scientific sites for exploration.
- **Key Concepts**: Reinforcement Learning, Terrain Adaptation, Autonomous Exploration, Environmental Sensing.
- **Simulation Platform**: **OpenAI Gym**, **Gazebo**, **Unity** (with Mars terrain simulation).

13. AI-Based Search and Rescue Robot in Disaster Zones

- **Description**: Develop a robot capable of navigating through a disaster zone (e.g., a collapsed building or forest fire) to locate survivors and provide real-time data. The robot must use sensors (e.g., cameras, temperature sensors) to detect survivors and avoid dangerous obstacles like fire or falling debris.
- **Key Concepts**: Reinforcement Learning, Multi-Agent Systems, Object Detection, Robotics.
- **Simulation Platform**: **ROS**, **Gazebo**, or **OpenAI Gym** for multi-agent coordination.

14. AI for Smart Traffic Flow Optimization in Real-Time

- **Description**: Design an AI system that controls the traffic lights in a city to optimize traffic flow in real-time. The AI can adapt to changing traffic patterns, accidents, and time-of-day traffic volumes using reinforcement learning.
- **Key Concepts**: Reinforcement Learning, Multi-Agent Systems, Traffic Simulation.
- **Simulation Platform**: **SUMO (Simulation of Urban MObility)**, **OpenAI Gym**.

15. AI for Real-Time Object Tracking in Video Surveillance

- **Description**: Build a system that can track objects in real-time from video streams (e.g., detecting moving people, vehicles, or animals). This system would use computer vision techniques combined with reinforcement learning to adapt to environmental changes and optimize tracking accuracy.
- **Key Concepts**: Computer Vision, Reinforcement Learning, Object Detection, Tracking.
- **Simulation Platform**: **OpenCV**, **TensorFlow** for object detection, **OpenAI Gym** for reinforcement learning.

16. AI for Autonomous Underwater Exploration and Mapping

- **Description**: Simulate an autonomous underwater vehicle that explores the ocean floor and creates a map of its environment. The vehicle should use reinforcement learning to avoid underwater obstacles, perform exploration tasks, and map new regions.
- **Key Concepts**: Reinforcement Learning, SLAM (Simultaneous Localization and Mapping), Robotics.
- **Simulation Platform**: **Gazebo**, **ROS**, or **OpenAI Gym**.

17. AI for Energy Consumption Optimization in Smart Homes

- **Description**: Design an AI system for a smart home that optimizes energy consumption based on factors like the time of day, occupancy, and weather conditions. The system would use reinforcement learning to control smart devices (e.g., lights, heating, cooling) and reduce energy costs.
- **Key Concepts**: Reinforcement Learning, Energy Optimization, Internet of Things (IoT).
- **Simulation Platform**: **OpenAI Gym**, **Python**, and smart home simulation environments.

18. AI for Personalized Health and Fitness Recommendations

- **Description**: Build an AI system that tracks user fitness data (e.g., steps, calories burned, heart rate) and provides personalized workout and nutrition recommendations. The system can use reinforcement learning to adjust recommendations based on the user’s progress and goals.
- **Key Concepts**: Reinforcement Learning, Personalization, Health Optimization.
- **Simulation Platform**: **Python**, **OpenAI Gym** for tracking and reward-based feedback.

19. Autonomous Drone Delivery System

- **Description**: Develop an autonomous drone system capable of delivering packages to specific locations within a city. The drone must navigate through obstacles, avoid collisions, and optimize its flight path to deliver items on time while minimizing energy consumption.
- **Key Concepts**: Reinforcement Learning, Path Planning, Drone Navigation, Robotics.
- **Simulation Platform**: **AirSim**, **OpenAI Gym**, or **Gazebo**.

20. AI for Disaster Response using Drones and Robots

- **Description**: Build a simulation where drones and ground robots are used in disaster response operations. The system will use reinforcement learning to optimize the deployment and coordination of drones and robots to survey disaster zones, locate survivors, and deliver emergency supplies.
- **Key Concepts**: Multi-Agent Systems, Reinforcement Learning, Robotics, Coordination.
- **Simulation Platform**: **OpenAI Gym**, **Gazebo**, or **V-REP**.

21. AI for Traffic Collision Prediction and Prevention

- **Description**: Develop a system that predicts traffic accidents in real-time based on vehicle movement, weather conditions, and other relevant factors. The system would then recommend or take action (e.g., adjusting speed limits, controlling traffic signals) to avoid or mitigate accidents.
- **Key Concepts**: Machine Learning, Prediction, Reinforcement Learning, Traffic Safety.
- **Simulation Platform**: **SUMO**, **OpenAI Gym**, or **CityFlow** for traffic simulation.

22. Self-Driving Car Simulation with Reinforcement Learning

- **Description**: Simulate a self-driving car in a virtual environment (using Gym or other tools like CARLA). The car should learn to navigate through city streets, avoid obstacles, and follow traffic rules using reinforcement learning algorithms like Q-learning or deep Q-networks (DQN).
- **Key Concepts**: Reinforcement Learning, Deep Q Networks (DQN), Policy Gradients, Convolutional Neural Networks (CNN) for image processing.
- **Simulation Platform**: OpenAI Gym with a custom environment or CARLA simulator.

23. Autonomous Drone Navigation in a 3D Environment

- **Description**: Develop a simulation where an autonomous drone must navigate through a 3D space, avoiding obstacles, and performing tasks like delivering packages to specific locations. The drone's behavior can be improved through reinforcement learning where it maximizes rewards for efficient flight and obstacle avoidance.
- **Key Concepts**: Reinforcement Learning, 3D Navigation, Drone Control, Neural Networks.
- **Simulation Platform**: OpenAI Gym, AirSim, or Gazebo.

24. AI for Game Design: NPC Behavior in a Strategy Game

- **Description**: Simulate NPC behavior in a complex strategy game (e.g., a tower defense game or real-time strategy game). The NPC should learn strategies to counter the player's moves, using search algorithms or reinforcement learning to decide when to attack, defend, or collect resources. This will involve modeling game environments and decision-making under uncertainty.
- **Key Concepts**: Reinforcement Learning, Search Algorithms, Game Theory, Game Design.
- **Simulation Platform**: OpenAI Gym with a custom game environment, Unity ML-Agents.

25. Traffic Flow Management with Reinforcement Learning

- **Description**: Simulate a city's traffic system where traffic lights are controlled by an AI agent. The agent should use reinforcement learning to optimize the flow of traffic, reduce congestion, and minimize waiting times for vehicles. This could involve multi-agent systems where each traffic light communicates with others to improve overall traffic flow.
- **Key Concepts**: Reinforcement Learning, Multi-agent Systems, Traffic Simulation, Control Systems.
- **Simulation Platform**: OpenAI Gym, SUMO (Simulation of Urban MObility), or CityFlow.

26. Bayesian Network for Predictive Maintenance in Robotics

- **Description**: Simulate a scenario where a robot performs tasks, and you build a Bayesian Network to predict when maintenance is needed. The robot would feed data about its performance (temperature, speed, battery life, etc.), and the Bayesian network would update probabilities on the health of different components.
- **Key Concepts**: Bayesian Networks, Probability, Predictive Maintenance, Robotics.
- **Simulation Platform**: OpenAI Gym, Python libraries like `pgmpy` for Bayesian Networks.

27. Maze Solver using Reinforcement Learning

- **Description**: Develop a reinforcement learning-based agent that learns to navigate a maze efficiently. The agent should start with no knowledge of the maze and learn through trial and error the optimal path to the exit using reward feedback.
- **Key Concepts**: Reinforcement Learning, Q-learning, Policy Gradient Methods, Exploration vs. Exploitation.
- **Simulation Platform**: OpenAI Gym with a custom maze environment.

28. AI for Real-Time Strategy Game (Starcraft-like)

- **Description**: Implement an AI agent that can play a real-time strategy game similar to StarCraft. The agent should be able to gather resources, build structures, and manage units to win against an opponent. The project would focus on learning strategies, resource management, and decision-making in real-time.

- **Key Concepts**: Reinforcement Learning, searching algorithms, Multi-agent Systems, Resource Management.
- **Simulation Platform**: OpenAI Gym with a custom strategy game, Unity, or PySC2 (StarCraft II API).

29. Multi-Robot Coordination for Search and Rescue Missions

- **Description**: Simulate a search and rescue operation using multiple robots in an environment (such as a collapsed building). The robots must coordinate their actions (e.g., search for survivors, map the area) using reinforcement learning algorithms to optimize their movement and task allocation.
- **Key Concepts**: Multi-Robot Systems, Reinforcement Learning, Task Allocation, Coordination.
- **Simulation Platform**: OpenAI Gym with custom multi-agent environments, V-REP, or Gazebo.

30. AI for Climate Control in Smart Buildings

- **Description**: Build a simulation for a smart building where an AI agent learns how to control various environmental factors (temperature, humidity, lighting) to optimize energy consumption while maintaining comfort levels for the inhabitants. The agent could use reinforcement learning to balance the trade-off between energy efficiency and user comfort.
- **Key Concepts**: Reinforcement Learning, IoT, Smart Systems, Optimization.
- **Simulation Platform**: OpenAI Gym with a custom smart building environment.

31. Autonomous Delivery Robot in Urban Environment

- **Description**: Design a simulation for an autonomous robot that delivers packages in an urban environment. The robot should navigate sidewalks, avoid pedestrians and other obstacles, and choose the fastest route. Reinforcement learning can be used for path planning and decision-making in dynamic, unpredictable urban environments.
- **Key Concepts**: Pathfinding, Reinforcement Learning, Obstacle Avoidance, Navigation.
- **Simulation Platform**: OpenAI Gym, ROS, or custom simulations in Unity.

32. AI-Powered Traffic Light System for Smart Cities

- **Description**: Simulate a traffic light control system for a smart city. Using reinforcement learning, the system can adapt in real-time to optimize the traffic flow across intersections, balancing factors like time of day, traffic volume, and pedestrian activity.
- **Key Concepts**: Reinforcement Learning, Multi-Agent Systems, Traffic Simulation.
- **Simulation Platform**: OpenAI Gym, SUMO (Simulation of Urban MObility), or CityFlow.

33. Autonomous Underwater Vehicle (AUV) for Ocean Exploration

- **Description**: Develop a simulation for an autonomous underwater vehicle that navigates oceanic environments. The AUV must avoid obstacles, map the ocean floor, and possibly perform tasks like data collection or environmental monitoring. Reinforcement learning could be used to improve pathfinding and decision-making under uncertain conditions.
- **Key Concepts**: Reinforcement Learning, Robotics, Pathfinding, Autonomous Navigation.
- **Simulation Platform**: OpenAI Gym, Gazebo, or custom underwater simulation.

34. AI for Smart Farming (Crop Yield Prediction)

- **Description**: Build a simulation to predict crop yields based on environmental factors like weather, soil quality, and farming practices. Bayesian networks could be used to model uncertainty, and reinforcement learning could optimize farming decisions (irrigation, fertilization, harvesting).
- **Key Concepts**: Bayesian Networks, Reinforcement Learning, Predictive Modeling, Agriculture.
- **Simulation Platform**: Custom simulation with Python, OpenAI Gym for task automation.

35. Robotic Arm Manipulation for Object Sorting

- **Description**: Simulate a robotic arm tasked with sorting objects based on size, color, or shape. The arm should learn to perform these tasks through reinforcement learning, improving its efficiency and accuracy over time. This project will explore robotic control and computer vision techniques.
- **Key Concepts**: Reinforcement Learning, Robotic Manipulation, Computer Vision, Object Recognition.
- **Simulation Platform**: OpenAI Gym, V-REP (CoppeliaSim), or Gazebo.

36. AI-Based Collision Avoidance System for Autonomous Vehicles

Objective: Create a collision avoidance system for a self-driving car that predicts potential accidents and takes corrective actions to avoid collisions in real-time.

- **Simulation Setup**: The car must navigate through city streets while detecting and avoiding obstacles like other cars, pedestrians, cyclists, and road hazards.
- **AI Techniques**:
  - **Object Detection and Tracking**: Use deep learning-based methods to detect and track moving objects (other vehicles, pedestrians, etc.) in the environment.
  - **Reinforcement Learning**: The vehicle learns to make real-time decisions (e.g., stopping, swerving, braking) to avoid accidents.
  - **Sensor Fusion**: Combine inputs from multiple sensors (LiDAR, cameras, radar) to get a comprehensive understanding of the environment and make more accurate decisions.

Real-World Application:

- Collision avoidance is a key feature in autonomous driving systems and is being developed by companies like **Waymo**, **Tesla**, and **Uber** to ensure safe driving in urban environments.

37. Autonomous Rover for Agriculture (Precision Farming)

Objective: Design an autonomous ground rover that can navigate agricultural fields, detect crops, and assist in tasks like irrigation, weeding, or monitoring crop health.

- **Simulation Setup**: The rover autonomously navigates through a crop field, uses sensors to identify weeds, monitor crop growth, or detect soil moisture levels.
- **AI Techniques**:
  - **Reinforcement Learning**: The rover learns optimal strategies for watering, weeding, or other tasks, improving its efficiency over time.
  - **Path Planning**: The rover plans its route across the field, avoiding obstacles (e.g., trees, fences, uneven terrain) and optimizing its path for efficiency.

38. Autonomous Parking System for Self-Driving Cars

Objective: Create an autonomous parking system where a self-driving car can park itself in a parking lot or garage using AI algorithms.

- **Simulation Setup**: The car must navigate a parking lot or garage, find an empty space, and park itself. This requires precise control and the ability to avoid obstacles such as other parked cars and pedestrians.
- **AI Techniques**:
  - **Computer Vision**: Use object detection and segmentation to recognize parking spaces, obstacles, and pedestrians.
  - **Reinforcement Learning**: Train the car to park in the most efficient way, using RL to reward successful parking attempts and penalize inefficient or incorrect actions.
  - **Motion Planning**: Implement **path planning** algorithms (like **Rapidly-exploring Random Trees (RRT)** or **A*** for parking and moving the vehicle into tight spaces.