Computer on Wheels

****

**By:**

**Bilal Rafiq**

**27661**

**Hamza Azhar**

**28595**

**Sardar Mohsin Saghir**

**28016**

**M. Usama Nazir**

**30445**

**Supervised by:**

**Dr. Rizwan Bin Faiz**

**Faculty of Computing**

**Riphah International University, Islamabad**

**Spring 2024A Dissertation Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfilment of the Requirement for the Award of the Degree of**

**Bachelors of Science in Software Engineering**

**Faculty of Computing**

**Riphah International University, Islamabad**

Date: 20th June, 2024**Final Approval**

This is to certify that we have read the report submitted by ***Bilal Rafiq (27661), Hamza Azhar (28595), Sardar Mohsin Saghir (28016) and M. Usama Nazir (30445),*** for the partial fulfilment of the requirements for the degree of the Bachelors of Science in Software Engineering (BSSE). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Software Engineering (BSSE).

**Committee:**

|  |  |
| --- | --- |
| **1** | Dr. Rizwan Bin Faiz  (Supervisor) |
|  |  |
| **2** | Dr. Musharraf Ahmed  (Head of Department/chairman) |

**Declaration**

We hereby declare that this document “**Computer on Wheels**” neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers, especially our supervisor **Dr.** **Rizwan Bin Faiz**. If any part of the system is proved to be copied out from any source or found to be the reproduction of any project from anywhere else, we shall stand by the consequences.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Bilal Rafiq**

**26771**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Hamza Azhar**

**28595**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Sardar Mohsin Saghir**

**28016**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**M. Usama Nazir**

**30445**

**Dedication**

We dedicate this project to Allah Almighty our creator, our strong pillar, our source of

inspiration, wisdom, knowledge and understanding. He has been the source of our strength throughout this program. Also, we dedicate our work to our family, friends and

teachers. The unrivalled encouragement from our parents and outstanding support from teachers is what led to the success of this project. We also dedicate our work to our supervisor **Dr. Rizwan Bin Faiz, Maanz AI** for their guidance and support and the faculty members.

**Acknowledgement**

Firstly, we are obliged to Allah Almighty the Merciful, the Beneficent, and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

We are grateful to our supervisor **Dr. Rizwan Bin Faiz** for his enthusiasm, patience, insightful comments, helpful information, practical advice, and unceasing ideas that always helped us tremendously in our project. Without his support and guidance, this project would not have been possible. Also, a special thanks to **Maanz AI**, the faculty members, and **Mr. M. Tahir Anis**, an alumnus, for their invaluable support.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Bilal Rafiq**

**26771**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Hamza Azhar**

**28595**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Sardar Mohsin Saghir**

**28016**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**M. Usama Nazir**

**30445**

**Abstract**

The emergence of AVs (Autonomous Vehicles) promises to revolutionize transportation by enhancing safety and efficiency. Despite this potential, challenges such as human-error accidents and productivity loss during travelling. This project aims to address these challenges by developing an AV software system utilizing **Machine learning-powered obstacle detection**. Through the integration of **path planning and dynamic obstacle avoidance algorithms**, the system enhances AVs' capabilities to navigate urban environments with precision and safety. Implementing algorithms and low-cost solutions, this project offers a new approach to self-driving technology. The development of AVs by companies such as Tesla, Waymo, and Uber is establishing the way for a future of transportation that promises increased global efficiency, safety, and security.

**Table of Contents**

[List of Figures 10](#_Toc167899110)

[List of Tables 11](#_Toc167899111)

[Chapter 1: Introduction 13](#_Toc167899112)

[1.1 Opportunity and Stakeholder 13](#_Toc167899113)

[1.2 Motivations and Challenges 14](#_Toc167899114)

[1.3 Goals and Objectives 14](#_Toc167899115)

[1.4 Solution Overview 15](#_Toc167899116)

[1.5 Report Outline 17](#_Toc167899117)

[1.5.1 Chapter 1 17](#_Toc167899118)

[1.5.2 Chapter 2 17](#_Toc167899119)

[1.5.3 Chapter 3 17](#_Toc167899120)

[1.5.4 Chapter 4 17](#_Toc167899121)

[1.5.5 Chapter 5 17](#_Toc167899122)

[1.5.6 Chapter 6 18](#_Toc167899123)

[1.5.7 Chapter 7 18](#_Toc167899124)

[Chapter 2: Literature/Market Survey 20](#_Toc167899125)

[2.1 Introduction 20](#_Toc167899126)

[2.2 Literature Review / Technology Overview 20](#_Toc167899127)

[2.3 Brainstorming 23](#_Toc167899128)

[2.4 Existing Systems 23](#_Toc167899129)

[2.5 Summary 24](#_Toc167899130)

[Chapter 3: Requirement Engineering 26](#_Toc167899131)

[3.1 Introduction 26](#_Toc167899132)

[3.2 Problem Scenarios 26](#_Toc167899133)

[3.3 Functional Requirements 28](#_Toc167899134)

[3.4 Non-Functional Requirement 35](#_Toc167899135)

[3.5 SQA activities: Defect Detection 35](#_Toc167899136)

[Chapter 4: System Design 41](#_Toc167899137)

[4.1 Introduction 41](#_Toc167899138)

[4.2 Architectural Design 42](#_Toc167899139)

[4.3 Detailed Design 43](#_Toc167899140)

[4.3.1 Use Case Design 43](#_Toc167899141)

[4.3.2 Detailed Use Cases 44](#_Toc167899142)

[4.3.3 Activity Diagrams 56](#_Toc167899143)

[4.3.4 Sequence Diagram 69](#_Toc167899144)

[Chapter 5: Implementation 71](#_Toc167899145)

[4.1 Endeavour 71](#_Toc167899146)

[4.2 Components and Libraries 82](#_Toc167899147)

[4.3 IDE, Tools and Technologies 83](#_Toc167899148)

[4.4 Best Practices / Coding Standards 84](#_Toc167899149)

[4.5 Deployment Environment 85](#_Toc167899150)

[4.6 Summary 85](#_Toc167899151)

# List of Figures

|  |  |  |
| --- | --- | --- |
| 4.1 | Architecture diagram | 32 |
| 4.2 | Use case diagram | 33 |
| 4.3 | Activity diagram start system | 41 |
| 4.4 | Activity diagram locate vehicle | 42 |
| 4.5 | Activity diagram set destination | 43 |
| 4.6 | Activity diagram find path | 44 |
| 4.7 | Activity diagram plan trajectory | 45 |
| 4.8 | Activity diagram start ride | 46 |
| 4.9 | Activity diagram obstacle detection | 47 |
| 4.10 | Activity diagram obstacle avoidance | 48 |
| 4.11 | Activity diagram stop vehicle | 49 |
| 4.12 | Sequence diagram | 50 |

# List of Tables

|  |  |  |
| --- | --- | --- |
| 2.1 | Levels of taxonomy | 24 |
| 3.1 | Problem statement: Accidents caused by human error | 28 |
| 3.2 | Problem statement: Less Productivity of driver | 28 |
| 3.3 | Problem statement: Less mobility for elder/ disabled people and children | 29 |
| 4.1 | Detailed Use-case: Access Map | 32 |
| 4.2 | Detailed: Use-case Locate Vehicle | 33 |
| 4.3 | Detailed Use-case: Set destination | 34 |
| 4.5 | Detailed Use-case: Find path | 35 |
| 4.6 | Detailed Use-case: Plan path | 36 |
| 4.7 | Detailed Use-case: Plan trajectory | 37 |
| 4.8 | Detailed Use-case: Start ride | 38 |
| 4.9 | Detailed Use-case: Obstacle detection. | 39 |
| 4.10 | Detailed Use-case: Obstacle avoidance | 40 |
| 4.11 | Detailed Use-case: Stop vehicle | 41 |

Chapter 1:

**Introduction**

# Chapter 1: Introduction

Computer on Wheels, is a software system for a car that can drive themself with minimal intervention. The system is capable of autonomously controlling the vehicle's movement, including **throttle control**, **acceleration, braking, and steering**. Furthermore, it incorporates obstacle detection capabilities to detect and respond to obstacles ensuring safe navigation. Moreover, path planning algorithms are used to determine optimal routes from point A to point B. By leveraging state-of-the-art technologies such as the **CARLA (Car Learning to Act) simulator, CARLA-ROS bridge, and ROS (Robot Operating System)**, our project endeavours to create a software solution capable of empowering autonomous vehicles to navigate urban environment with precision and confidence.

## 1.1 Opportunity and Stakeholder

* According to a **National Highway Traffic Safety Administration (NHTSA)** study, driver error led to **94% of the crashes** examined.
* According to the **U.S. General Services Administration (GSA)**, human error causes **98% of crashes**.
* A 2017 study by **RAND Corporation** found that self-driving cars could reduce traffic fatalities by up to **25% by 2040**.
* A 2019 study by the National Highway Traffic Safety Administration (NHTSA) found that self-driving cars were involved in fewer crashes than human-driven cars per mile driven.
* A 2020 study by the **Massachusetts Institute of Technology (MIT)** found that self-driving cars could **prevent up to 90%** of crashes caused by human error.
  + 1. **Stakeholders**
* Driver
* Passengers
* Maanz AI
* Domain Expert

## Motivations and Challenges

Our project is motivated by the importance of enhancing safety for passengers, drivers, and pedestrians through autonomous vehicle technology. By alleviating the need for human drivers, we aim to enable multitasking and provide independence to individuals, including those with disabilities. Challenges such as time management and acquiring a physical model car for demonstrations were overcome by transitioning to the CARLA simulator. However, GPU resource limitations were encountered, which were addressed through assistance from **Maanz AI**, securing workspace and expert guidance.

## Goals and Objectives

**Our goals are clear**: complete the project on time while ensuring high-quality deliverables and develop autonomous vehicle software to eliminate accidents caused by human error and enhance mobility for individuals with disabilities. These objectives will minimize errors, boost stakeholder productivity, and provide mobility for aged persons and people having disabilities.

## Solution Overview

Our solution includes developing autonomous vehicle software utilizing cutting-edge technologies like the CARLA simulator, ROS Noetic, CARLA-ROS bridge, and rospy. This software will enable vehicles to autonomously navigate complex environments by implementing key functionalities:

* Path Planning
* Path Following
* Obstacle Detection
* Obstacle Avoidance

Also, ensuring safety and precision while minimizing accidents caused by human error. Additionally, our solution prioritizes accessibility, aiming to provide mobility for individuals with disabilities and the elderly. Through rigorous development and testing, we endeavour to deliver a reliable and efficient solution that revolutionizes autonomous vehicle navigation.

* + 1. **Project Scope**

The scope of this project encompasses the development and implementation of key functionalities:

* + - 1. **Integration**
* Involve integrating various sensors and algorithms to enable the vehicle to perceive its environment accurately, make decisions, and navigate safely through dynamic scenarios.
  + - 1. **Path Planning:**
* Determining a feasible and shortest path from user-specified source and destination locations
* Implementing a navigation algorithm to handle dynamic environments and potential rerouting.
  + - 1. **Path Following:**
* Implementing control algorithms for precise vehicle guidance along the planned trajectory.
* Maintaining vehicle position and orientation relative to the path using steering, acceleration, and braking control.
  + - 1. **Obstacle Detection:**
* Utilizing sensor data (such as lidar, radar or cameras) to detect objects within the vehicle's surroundings.
* Employing algorithms/models to classify detected objects and assess their characteristics, such as size, shape, and distance.
* Integrating machine learning techniques to improve the accuracy and reliability of obstacle recognition.
* Providing real-time information about detected obstacles to inform path planning and navigation decisions.
  + - 1. **Obstacle avoidance:**
* Implement reactive obstacle avoidance strategies, allowing the autonomous vehicle to dynamically adjust its trajectory based on the detected obstacles, enabling safe navigation.
* Develop algorithms/maneuver for real-time analysis of obstacle data to facilitate swift decision-making by the autonomous vehicle.

## Report Outline

This report covers all aspects of the Computer on Wheels, for understanding and clarity. This report has been divided into seven chapters.

### Chapter 1

This chapter serves as an introduction to our software system, encapsulating the project's opportunities, stakeholders, motivations, challenges, goals, objectives, and the proposed solution.

### Chapter 2

This chapter undertakes a thorough examination of existing literature pertaining to autonomous vehicles, alongside an analysis of companies operating within this domain.

### Chapter 3

This chapter outlines the essential requirements that serve as the foundation for guiding the development process and ensuring that the system meets the needs and expectations of stakeholders and end-users.

### Chapter 4

This chapter comprehensively covers the design factors of the developed system, focusing on system architecture design considerations and various diagrams modelling the working behaviour of the system.

### Chapter 5

This chapter includes the implementation process of our project, outlining the steps taken to achieve our goals and the integration of technologies and methodologies to ensure the successful development of our project.

### Chapter 6

This chapter is dedicated to testing our project, where its functionality is assessed through testing protocols.

### Chapter 7

This chapter includes the conclusion of our project, along with a brief outlook

Chapter 2:

**Literature/Market Survey**

# Chapter 2: Literature/Market Survey

This chapter aims to provide an overview of the current state of autonomous vehicles, including existing developments and ongoing testing. It will explore the origins of autonomous vehicles and the regulatory bodies responsible for establishing rules. Furthermore, it will identify prominent market participants involved in advancing autonomous vehicle technologies.

## Introduction

The concept of autonomous vehicles is not fresh in the automotive industry. Companies such as Tesla, General Motors, BMW, Mercedes, Honda, KIA, Toyota, among others, have been actively involved in this field. While many have developed vehicles equipped with level 2 and level 3 autonomous systems, not all have released them to the market. The Society of Automotive Engineers (SAE) has established six levels of driving automation, ranging from level 0 (fully manual) to level 5 (fully autonomous).

## Literature Review / Technology Overview

The concept of autonomous vehicles traces back to 1918, with early attempts in the 1920s. General Motors was among the pioneers, showcasing autonomous vehicle concepts at exhibitions. The research and development efforts for autonomous vehicles gained momentum with initiatives like General Motors and Radio Corporation of America Sarnoff Laboratory's collaboration. Notably, the Defense Advanced Research Projects Agency (DARPA) Grand Challenges Program in 2004 accelerated autonomous vehicle research in the US.

Today, the global autonomous vehicle market boasts key players including AB Volvo, BMW AG, Daimler AG, Ford Motor Company, General Motors, Honda Motor Co., Ltd., Nissan Motors Co., Ltd., Tesla, Inc., Toyota Motor Corporation, and Volkswagen AG.

* **AB Volvo**: Began autonomous vehicle development in 2006 and unveiled a fully autonomous test vehicle in 2017, though commercially available self-driving cars from Volvo are still pending.
* **Waymo** (Google's subsidiary): Made significant progress, logging millions of autonomous driving miles. Currently offers limited commercial self-driving ride-hailing services in specific locations.
* **Tesla**: Announced plans for self-driving features in their cars in 2014, promoting them as standard. Notably, Tesla's Autopilot is a driver-assistance system rather than fully autonomous, and has faced safety criticisms.

AVs operate themselves and execute necessary functions without human intervention. This is achieved through their ability to sense their surroundings using advanced technologies such as artificial intelligence (AI) software, light detection and ranging (LiDAR), radio detection and ranging (RADAR), and cameras. These sensors enable the vehicle to form an active 3D map of its environment, allowing it to navigate safely and efficiently.

* + 1. **Levels of Autonomous Vehicles**

Understanding the different levels of autonomy set by the Society of Automotive Engineers (SAE) International is crucial before discussing existing autonomous vehicle systems. These levels explain how much control the vehicle has versus the human. The table below shows these levels, from full human control to full automation, making it easier to understand the capabilities of existing systems.

|  |  |
| --- | --- |
| **Levels of Taxonomy** | **Description** |
| **Level 0**  No automation | Zero autonomy; the driver performs all driving tasks. |
| **Level 1**  Driver assistance | The vehicle is controlled by the driver but driving assist features may be included in the vehicle design. |
| **Level 2**  Partial automation | Vehicles have combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and always monitor the environment. |
| **Level 3**  Conditional automation | A driver is a necessity but is not required to monitor the environment. The driver must be ready to always take control of the vehicle with notice. |
| **Level 4**  High automation | The vehicle can perform all driving functions under certain conditions. The driver may have the option to control the vehicle. |
| **Level 5**  Full automation | The vehicle can perform all driving functions under all conditions. |

*Table 2.1 - Levels of taxonomy*

## Brainstorming

## Existing Systems

|  |  |  |
| --- | --- | --- |
| **Company** | **Target Level** | **Key Features** |
| **Tesla, Ford, Toyota** | Level 2 (Autopilot) | * Lane keeping * automatic emergency braking * traffic light and stop sign recognition * highway driving assist * self-parking (Level 2) * Navigate on Autopilot |
| **BMW, Nissan** | Level 2 | * Adaptive cruise control with stop-and-go * lane departure warning * lane change assist |

*Table 2.4.1* **|** *Existing Systems*

|  |  |  |
| --- | --- | --- |
| **Honda, Mercedes-Benz** | Level 3 (conditional) | * Hands-free driving at up to 60 km/h on specific highways * automatic lane changes * traffic jam assist * emergency stop assists |
| **Way-mo** | Level 4 | * LiDAR-based system for navigating complex * extensive real-world testing * millions of miles driven |
| **Cruise** | Level 5 | Fully autonomous robo-taxi |

*Table 2.4.2* **|** *Existing Systems*

Currently, the automotive market provides vehicles with Levels 0, 1, and 2 of automation. Levels 3, 4, and 5 are still in the **testing phase** and not widely available for commercial use.

## Summary

This chapter analyzes the current landscape of autonomous vehicles (AVs). While various companies are actively developing AV technology, commercially available vehicles primarily offer Levels 0 (no automation), 1 (driver assistance features), and 2 (partial automation) of driving autonomy as defined by the Society of Automotive Engineers (SAE). Levels 3 (conditional automation), 4 (high automation), and 5 (full automation) remain under development and testing.

Chapter 3:

**Requirement Analysis**

# Chapter 3: Requirement Engineering

## Introduction

In this chapter we will discuss the requirements of our project “Computer on Wheels”. Prior to that, we will discuss all the problem statements we have found while doing research on the project idea. These requirements are gathered using a variety of techniques, including interviewing domain experts and conducting documentation analysis. Our approach involves reviewing existing documentation, research papers, industry standards, and guidelines related to autonomous vehicle navigation.

## Problem Scenarios

|  |  |
| --- | --- |
| **Problem Statement # 1: Hazards caused by human errors** | |
| The problem of | hazards caused by human errors |
| Affects | passengers, drivers, and pedestrians |
| The result of which | more injuries/deaths, Damage to property and Emotional stress |
| Benefits of | mitigation of human errors thus reduces accidents and fatalities |

*Table 3.2.1* ***|*** *problem statement*

|  |  |
| --- | --- |
| **Problem Statement # 2: Limited driver productivity** | |
| The problem of | driver's unproductiveness while driving |
| Affects | drivers |
| The result of which | it decreased efficiency |
| Benefits of | increased productivity by doing other important tasks |

*Table 3.2.2* ***|*** *problem statement*

|  |  |
| --- | --- |
| **Problem Statement # 3: Poor mobility for the aged, disabled, and children.** | |
| The problem of | transportation reliance for vulnerable groups such as children, the elderly, and the disabled |
| Affects | passenger. |
| The result of which | increased risks to the health vulnerable individuals, potentially leading to delayed or insufficient medical care in emergency situations |
| Benefits of | accessibility, independence and reduced burden on caregivers |

*Table 3.2.3* ***|*** *problem statement*

## Functional Requirements

**Vehicle Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **1** | **Vehicle Control** | **1.1** | Autonomous Navigation | The system shall be capable of autonomously navigating from a starting point to a destination. |
| **1** | **Vehicle Control** | **1.2** | Acceleration Control | The system shall control the vehicle's acceleration to maintain desired speeds along the planned trajectory. |
| **1** | **Vehicle Control** | **1.3** | Emergency Stop | The system shall include a mechanism for the driver to perform an immediate emergency stop, halting all vehicle operations. |
| **1** | **Vehicle Control** | **1.4** | Throttle Control | The system shall control the throttle to regulate vehicle speed within a range of 0 to 120 km/h, adjusting for road conditions and traffic regulations. |
| **1** | **Vehicle Control** | **1.5** | Steering Control | The system shall control the vehicle's steering to maintain a maximum lateral deviation of 0.5 meters from the planned trajectory under normal conditions. |
| **1** | **Vehicle Control** | **1.6** | Braking Control | The system shall control the vehicle's braking to safely decelerate and stop as required by the planned trajectory |

*Table 3.3.1 | FR1*

**Path Planning**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **2** | **Path Planning** | **2.1** | Route Calculation | The system shall calculate the most efficient route i.e. shortest path from the vehicle's current location to the driver-specified destination |
| **2** | **Path Planning** | **2.2** | Lane Assignment | The system shall assign appropriate lanes for the vehicle to travel in along the calculated route, based on legal navigation rule. |
| **2** | **Path Planning** | **2.3** | Waypoint Generation | The system shall generate waypoints along the calculated route to guide the vehicle towards the destination. |
| **2** | **Path Planning** | **2.4** | Dynamic Obstacle Avoidance | The system shall adapt the vehicle's path in real-time to safely avoid unexpected obstacles. |
| **2** | **Path Planning** | **2.5** | Map Reading | The system shall be able to read and interpret digital map data to determine the vehicle's precise location within the road network |

*Table 3.3.2 | FR2*

**Path Following:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **3** | **Path Following** | **3.1** | **Path smoothing** | The system shall apply path smoothing techniques to limit acceleration changes to within 0.3 m/s², ensuring a smooth ride for passengers. |
| **3** | **Path Following** | **3.2** | **Lateral Control** | The system shall maintain a lateral deviation of no more than 0.5 meters from the planned path under normal driving conditions. |
| **3** | **Path Following** | **3.3** | **Longitudinal Control** | The system shall maintain a longitudinal deviation of no more than 1 meter from the planned path under normal driving conditions. |
| **3** | **Path Following** | **3.4** | **Speed Control** | The system shall control the speed to reach the destination. |
| **3** | **Path Following** | **3.5** | **Waypoint Following** | The system shall follow waypoints along the calculated waypoints to guide the vehicle towards the destination. |

*Table 3.3.3 | FR3*

**Sensor Integration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **4** | **Sensor Integration** | **4.1** | Inertial Measurement Unit Utilization | The system shall use an IMU to provide orientation and acceleration data at a frequency of 100 Hz. |
| **4** | **Sensor Integration** | **4.2** | Global Positioning System Utilization | The system shall use GPS to determine the vehicle’s position. |
| **4** | **Sensor Integration** | **4.3** | Radar/Lidar Utilization | The system shall utilize radar/lidar sensors to provide additional information about surrounding objects' velocity and distance, enhancing situational awareness. |

*Table 3.3.4 | FR4*

**Trajectory Planning**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **5** | **Trajectory Planning** | **5.1** | Trajectory Generation | The system shall plan a smooth and optimal trajectory, based on destination specified by user. |

*Table 3.3.5 | FR5*

**Obstacle Detection**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **6** | **Obstacle Detection** | **6.1** | Detection Using Sensors | The system shall utilize various sensors to detect obstacles in the vehicle's path. |
| **6** | **Obstacle Detection** | **6.2** | Environmental Awareness | The system shall maintain the awareness of static and dynamic objects in the vehicles vicinity |
| **6** | **Obstacle Detection** | **6.3** | Dynamic Obstacle Tracking | The system shall continuously track the moving obstacle. |
| **6** | **Obstacle Detection** | **6.4** | Destination Estimation | The system shall be able to calculate the distance to detected obstacles |

*Table 3.3.6 | FR6*

**Obstacle Avoidance**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **7** | **Obstacle Avoidance** | **7.1** | Maneuver Execution | The system shall execute safe and efficient avoidance maneuvers to navigate around detected obstacles and maintain collision-free travel. |
| **7** | **Obstacle Avoidance** | **7.2** | Steering Control | The system shall dynamically adjust steering angles to guide the vehicle away from obstacles and keep it on its intended path. |
| **7** | **Obstacle Avoidance** | **7.3** | Re-Plan Path | The system shall re-plan the path, once the object is detected |
| **7** | **Obstacle Avoidance** | **7.4** | Trajectory Adjustment | The system shall dynamically adjust the vehicle’s trajectory to avoid obstacle in some clear and clean environment. |
| **7** | **Obstacle Avoidance** | **7.5** | Multi-Obstacle Handling | The system shall manage avoidance of multiple obstacles simultaneously |

*Table 3.3.7 | FR7*

**Destination Arrival**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **8** | **Destination Arrival** | **8.1** | Destination Approach | The system shall approach the driver-specified destination with a positional accuracy of within 1 meter, following the calculated trajectory and waypoints. |
| **8** | **Destination Arrival** | **8.2** | Stop at Destination | The system shall bring the vehicle to a complete stop within 1 meter of the designated destination, ensuring deceleration rates do not exceed 2 m/s² for passenger safety and comfort. |

*Table 3.3.8 | FR8*

**User Inputs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **9** | **User Inputs** | **9.1** | Ride Initiation | The driver shall be able to initiate journey. |
| **9** | **User Inputs** | **9.2** | Destination Setting | The driver shall be able to input the desired destination, triggering the route planning process. |

*Table 3.3.9 | FR9*

**System Integration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functional Requirement** | **Breakdown** | | **Description** |
| **ID** | **Sub-Functionality** |
| **10** | **System Integration** | **10.1** | ROS Integration | The system shall utilize the Robot Operating System (ROS) to facilitate communication and data exchange between different software components. |
| **10** | **System Integration** | **10.2** | Simulation Environment | Development and testing of the system shall be conducted in a simulated environment (e.g., CARLA simulator) for thorough validation before real-world deployment. |

*Table 3.3.10 | FR10*

## Non-Functional Requirement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Non- Functional Requirement** | **ID** | **Description** | **Subfactor** |
| **1** | **Safety Requirement** | **1.1** | Ensure reliable object detection in adverse weather conditions to assure safety | **Hazard Protection** The system must detect and respond to hazards arising from adverse weather conditions, such as rain, fog, or snow, which may reduce visibility. |

*Table 3.4.1 | FR**1*

## SQA activities: Defect Detection

* + 1. **Throttle Control:**

**Original:** The system shall control the throttle for regulation of vehicle speed.

**Revised:** The system shall control the throttle to regulate vehicle speed within a range of 0 to 120 km/h, adjusting for road conditions and traffic regulations.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall control the throttle for regulation of vehicle speed. | Verifiability: Is each requirement testable or verifiable? | The requirement lacks specifics on the range of speed control and conditions under which speed regulation should be adjusted. |

*Table 3.5.1 | Inspection Table 1*

* + 1. **Steering Control:**

**Original:** The system shall control the vehicle's steering to follow the planned trajectory accurately.

**Revised:** The system shall control the vehicle's steering to maintain a maximum lateral deviation of 0.5 meters from the planned trajectory under normal conditions.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall control the vehicle's steering to follow the planned trajectory accurately. | Clarity: Are the requirements stated clearly so there is only one interpretation? | The term "accurately" is vague and not quantifiable. |

*Table 3.5.2 | Inspection Table 2*

* + 1. **Route Calculation:**

**Original:** The system shall calculate the most efficient route i.e. shortest path from the vehicle's current location to the driver-specified destination.

**Revised:** The system shall calculate the most efficient route i.e. shortest path from the vehicle's current location to the driver-specified destination

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall calculate the most efficient route from the vehicle's current location to the driver-specified destination. | Verifiability: Does each requirement use concrete terms and measurable quantities? | "Most efficient route" is not defined; efficiency could refer to time, distance, fuel consumption, etc. |

*Table 3.5.3 | Inspection Table 3*

* + 1. **Path Smoothing:**

**Original:** The system shall apply path smoothing techniques to reduce jerkiness and ensure passenger comfort.

**Revised:** The system shall apply path smoothing techniques to limit acceleration changes to within 0.3 m/s², ensuring a smooth ride for passengers.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall apply path smoothing techniques to reduce jerkiness and ensure passenger comfort. | Verifiability: Is each requirement testable or verifiable? | The requirement does not define what constitutes "jerkiness" or acceptable levels of passenger comfort. |

*Table 3.5.4 | Inspection Table 4*

* + 1. **Lateral Deviation:**

**Original:** The system shall minimize the lateral deviation from the path.

**Revised:** The system shall maintain a lateral deviation of no more than 0.5 meters from the planned path under normal driving conditions.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall minimize the lateral deviation from the path. | Clarity: Are the requirements written in user language? Do the users think so? | "Minimize" is not quantified; specific acceptable deviation limits should be stated. |

*Table 3.5.5 | Inspection Table 5*

* + 1. **Longitudinal Deviation:**

**Original:** The system shall minimize the Longitudinal deviation from the path.

**Revised:** The system shall maintain a longitudinal deviation of no more than 1 meter from the planned path under normal driving conditions.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall minimize the Longitudinal deviation from the path | Clarity: Are the requirements written in user language? Do the users think so? | Similar to lateral deviation, "minimize" is not quantified, and specific limits should be provided. |

*Table 3.5.6 | Inspection Table 6*

* + 1. **IMU Data Usage:**

**Original:** The system shall use IMU to provide orientation and acceleration data at some frequency.

**Revised:** The system shall use an IMU to provide orientation and acceleration data at a frequency of 100 Hz.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall use IMU to provide orientation and acceleration data at some frequency. | Completeness: Are all the inputs to the system specified including their source, accuracy, range of values, and frequency? | "Some frequency" is vague and should be specified clearly. |

*Table 3.5.7 | Inspection Table 7*

* + 1. **Trajectory Planning:**

**Original:** The system shall plan a smooth and optimal trajectory for the vehicle to follow based on the calculated route.

**Revised:** The system shall plan a smooth and optimal trajectory, based on destination specified by user.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall plan a smooth and optimal trajectory for the vehicle to follow based on the calculated route. | Verifiability: Is each requirement testable or verifiable? | "Optimal trajectory" needs to be defined more concretely, considering factors like time, energy consumption, etc. |

*Table 3.5.8 | Inspection Table 8*

* + 1. **Destination Approach:**

**Original:** The system shall precisely approach the driver-specified destination by following the calculated trajectory and waypoints accurately.

**Revised:** The system shall approach the driver-specified destination with a positional accuracy of within 1 meter, following the calculated trajectory and waypoints precisely.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall precisely approach the driver-specified destination by following the calculated trajectory and waypoints accurately. | Clarity: Are the requirements stated clearly so there is only one interpretation? | The terms "precisely" and "accurately" are subjective and need quantifiable measures. |

*Table 3.5.9 | Inspection Table 9*

* + 1. **Stop at Destination:**

**Original:** The system shall bring the vehicle to a complete stop upon reaching the designated destination, ensuring a smooth and safe arrival.

**Revised:** The system shall bring the vehicle to a complete stop within 1 meter of the designated destination, ensuring deceleration rates do not exceed 2 m/s² for passenger safety and comfort.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Check List Point** | **Defect** |
| The system shall bring the vehicle to a complete stop upon reaching the designated destination, ensuring a smooth and safe arrival. | Completeness: Does each function specify the data used in the function and data resulting from the function? | "Smooth and safe arrival" should be quantified in terms of deceleration rates or stopping distance. |

*Table 3.5.10 | Inspection Table 10*

Chapter 4:

**System Design**

# Chapter 4: System Design

This chapter focuses on how we've designed our system. Design is based upon the requirements which are gathered using a variety of techniques, including interviewing domain experts and conducting documentation analysis. Our approach involves reviewing existing documentation, research papers, industry standards, and guidelines related to autonomous vehicle navigation. We won't dive into the visual parts of our software, but we'll explore how everything in the system works together

## Introduction

The software system is founded upon the architecture and framework of ROS 1, with outcomes visualized through the Carla Simulator. Facilitating seamless communication between Carla and ROS Noetic, we employ the ROS bridge as our interface for data retrieval and command transmission. Functioning as a crucial intermediary, the ROS bridge facilitates integration between ROS programs and non-ROS environments.

## Architectural Design

*Figure 4.1 - Architecture Diagram.*

## Detailed Design

### Use Case Design

*Figure 4.2 - Use-case Diagram.*

### Detailed Use Cases

**4.3.2.1 Set Destination**

|  |  |
| --- | --- |
| Use Case ID: | UC001 |
| Use Case: | Set Destination |
| Actor: | Passenger |
| Precondition: | The Car is integrated with the system |
| Basic Flow: | 1. Passenger launches the autonomous vehicle system. 2. Passenger selects a destination from the provided options. 3. System validates the selected destination. 4. System confirms the set destination to the passenger. |
| Alternative Flow: | 2a. If passenger does not want to select destination from the provided options, passenger can enter the coordinates of his/her choice.  4a. If the selected destination is not available or invalid, the system prompts the passenger to select another destination from the provided options or input a custom one. |
| Post Condition: | The destination is successfully set in the system. |

*Table 4.1 - Detailed: Use-case Set Destination*

**4.3.2.2 Plan Route**

|  |  |
| --- | --- |
| Use Case ID: | UC002 |
| Use Case: | Plan Route |
| Actor: | Ego Vehicle |
| Precondition: | The destination is set. |
| Basic Flow: | 1. System receives the set destination from the passenger.  2. System retrieves the current location of the vehicle.  3. System calculates the shortest path from the current location to the destination using path planning algorithms. |
| Alternative Flow: | None |
| Post Condition: | The system successfully calculates a shortest path from the current location to the destination for the vehicle to follow. |

*Table 4.2 - Detailed: Use-case Plan Route*

**4.3.2.3 Generate Waypoints**

|  |  |
| --- | --- |
| Use Case ID: | UC003 |
| Use Case: | Generate Waypoints |
| Actor: | Ego Vehicle |
| Precondition: | The route is planned. |
| Basic Flow: | 1. System receives the planned route. 2. System divides the planned route into discrete waypoints 3. System assigns coordinates to each generated waypoint to define the navigation path. |
| Alternative Flow: | None |
| Post Condition: | Waypoints are successfully generated along the planned route for navigation. |

*Table 4.3 - Detailed Use-case: Generate Waypoints*

**4.3.2.4 Navigate Generated Waypoints**

|  |  |
| --- | --- |
| Use Case ID: | UC004 |
| Use Case: | Navigate Generated Waypoints |
| Actor: | Ego Vehicle |
| Precondition: | Waypoints are generated |
| Basic Flow: | 1. System retrieves the planned route and waypoints. 2. System allows the vehicle to follows the planned path by steering and accelerating as necessary to reach each waypoint. 3. System continuously monitors the vehicle's position and adjusts the guidance commands to keep the vehicle on the planned path. 4. The vehicle progresses along the planned path until it reaches the final destination. |
| Alternative Flow: | 3a. If an obstacle is detected, it will avoid the obstacle using avoidance algorithm and update the guidance commands, vehicle resumes motion along the alternative route. |
| Post Condition: | The autonomous vehicle successfully follows the planned path, reaching the destination while ensuring safety and efficiency |

*Table 4.4 - Detailed Use-case: Navigate Generated Waypoints*

**4.3.2.5 Control Acceleration**

|  |  |
| --- | --- |
| Use Case ID: | UC005 |
| Use Case: | Control Acceleration |
| Actor: | Ego Vehicle |
| Precondition: | The vehicle is operational and in motion. |
| Basic Flow: | 1. System monitors the vehicle's velocity or acceleration parameters. 2. A change in acceleration is required, the system computes the necessary adjustments based on navigation requirements, traffic conditions, and vehicle dynamics. 3. System sends commands to adjust the vehicle's acceleration accordingly, using throttle control mechanisms. |
| Alternative Flow: | 3a. If an unexpected obstacle is detected requiring sudden deceleration, the system overrides the acceleration command and initiates avoidance maneuver. |
| Post Condition: | The vehicle's acceleration is controlled as per navigation and operational requirements. |

*Table 4.5 – Detailed Use-case: Control Acceleration*

**4.3.2.6 Control Throttle**

|  |  |
| --- | --- |
| Use Case ID: | UC006 |
| Use Case: | Control Throttle |
| Actor: | Ego Vehicle |
| Precondition: | The vehicle is operational and in motion. |
| Basic Flow: | 1. System monitors the vehicle's speed and throttle position. 2. A change in throttle position is required, the system computes the necessary adjustments based on navigation requirements, traffic conditions, and vehicle dynamics. 3. System adjusts the throttle position accordingly, regulating the engine's power output. |
| Alternative Flow: | None |
| Post Condition: | The vehicle's throttle position is controlled as per navigation and operational requirements. |

*Table 4.6 - Detailed Use-case: Control Throttle*

**4.3.2.7 Control Steering**

|  |  |
| --- | --- |
| Use Case ID: | UC007 |
| Use Case: | Control Steering |
| Actor: | Ego Vehicle |
| Precondition: | The vehicle is operational and in motion. |
| Basic Flow: | 1. System continuously monitors the vehicle's position, orientation, and intended path. 2. Based on navigation instructions and environmental factors, the system computes the required steering angle adjustments. 3. System adjusts the steering angle accordingly. |
| Alternative Flow: | None |
| Post Condition: | The vehicle's steering angle is controlled as per navigation and operational requirements. |

*Table 4.7 - Detailed Use-case: Control Steering*

**4.3.2.8 Assign Lane**

|  |  |
| --- | --- |
| Use Case ID: | UC08 |
| Use Case: | Assign Lane |
| Actor: | Ego Vehicle |
| Precondition: | The autonomous vehicle is driving on a multi-lane road. |
| Basic Flow: | 1. System identifies the need for a lane change 2. System evaluates the surrounding conditions 3. System determines the optimal timing and trajectory for the lane change to minimize disruption to traffic flow. 4. System executes the lane change maneuver by steering the vehicle smoothly into the target lane while maintaining safe distance from other vehicles. |
| Alternative Flow: | 2a. If the system detects an obstruction or unsafe condition in the target lane during the lane change maneuver:   * System aborts the lane change maneuver. * System re-evaluates the surrounding traffic conditions. * System selects an alternative lane change strategy |
| Post Condition: | The autonomous vehicle successfully changes lanes while ensuring safety and minimizing disruption to traffic flow. |

*Table 4.8 - Detailed Use-case: Assign Lane*

**4.3.2.9 Control Longitudinal Movement**

|  |  |
| --- | --- |
| Use Case ID: | UC09 |
| Use Case: | Control Longitudinal Movement |
| Actor: | Ego Vehicle |
| Precondition | Vehicle is in motion |
| Basic Flow | 1. System monitors the vehicle's speed.  2. System monitors the distance to vehicles and objects ahead.  3. System adjusts throttle to maintain desired speed.  4. System applies brakes to maintain safe following distance.  5. System continuously monitors and adjusts as needed. |
| Alternative Flow | None |
| Post Condition | Vehicle maintains desired speed and safe distance from other objects. |

*Table 4.9 - Detailed Use-case: Control Longitudinal Movement*

**4.3.2.10 Control Lateral Movement**

|  |  |
| --- | --- |
| Use Case ID: | UC10 |
| Use Case: | Control Lateral Movement |
| Actor: | Ego Vehicle |
| Precondition | Vehicle is in motion; path or lane is defined. |
| Basic Flow | 1. System monitors the vehicle's lateral position. 2. System detects deviations from the intended lane. 3. System adjusts steering to maintain lane position. 4. System continuously monitors and adjusts as needed. |
| Alternative Flow | 2. If an obstacle detected while deviating or changing lane position:  2.1. System performs avoidance maneuver.  2.2. System re-evaluates the lane and path.  2.3. System resumes lateral control once clear. |
| Post Condition | Vehicle maintains correct lateral position within its lane. |

*Table 4.10 - Detailed Use-case: Control Lateral Movement*

**4.3.2.11 Control jerkiness**

|  |  |
| --- | --- |
| Use Case ID: | UC11 |
| Use Case: | Control jerkiness |
| Actor: | Ego Vehicle |
| Precondition | Vehicle is in motion; speed and path are defined. |
| Basic Flow | 1. System monitors the vehicle's acceleration and deceleration patterns. 2. System smooths throttle and braking inputs to minimize jerky movements. 3. System adjusts steering inputs to provide smooth directional changes. 4. System continuously monitors and adjusts to maintain a smooth ride. |
| Alternative Flow | None |
| Post Condition | Vehicle maintains a smooth ride with minimal jerkiness during acceleration, deceleration, and directional changes. |

*Table 4.11 - Detailed Use-case: Control Jerkiness*

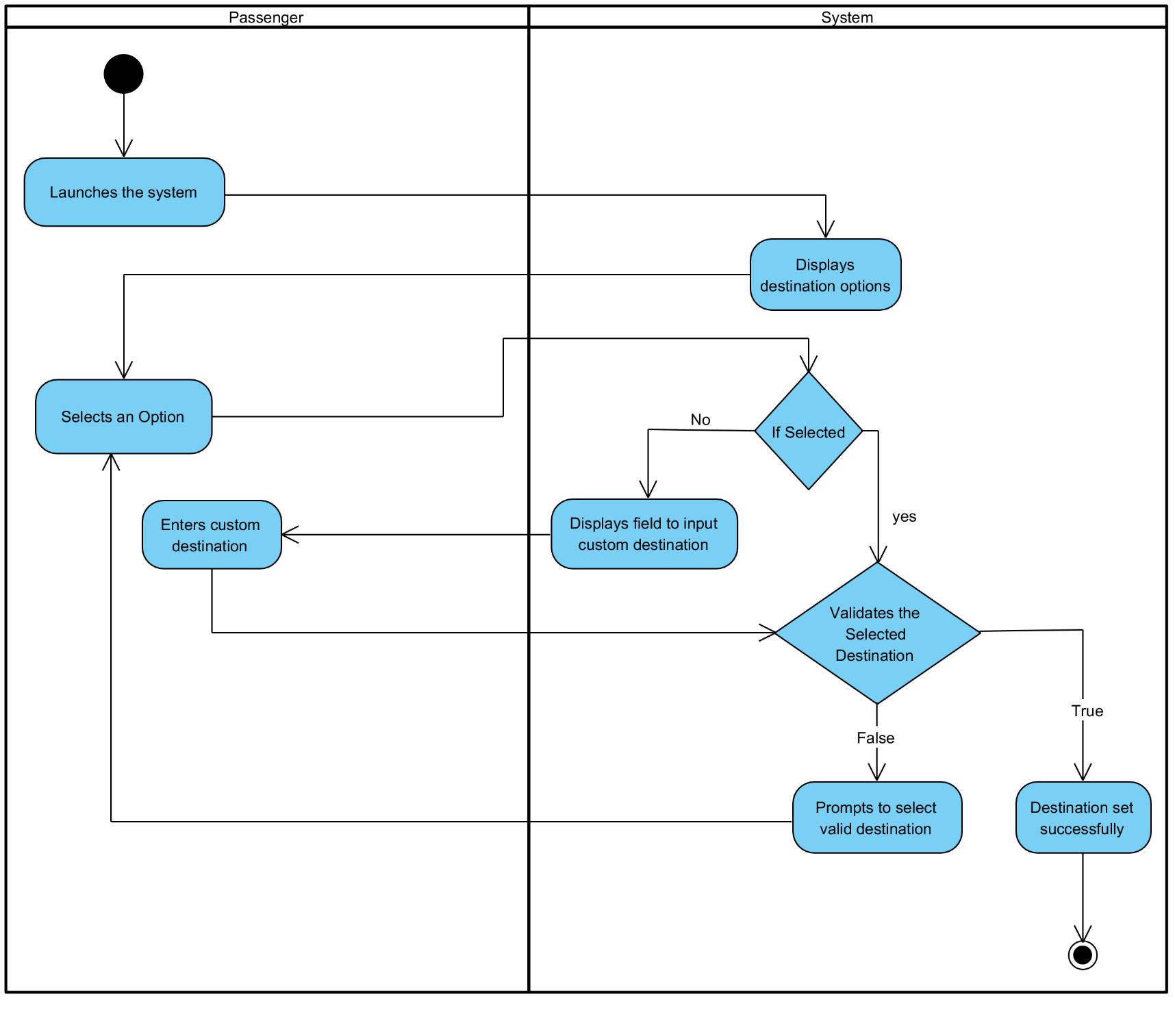
**4.3.2.12 Reach and Halt at Destination**

|  |  |
| --- | --- |
| Use Case ID: | UC12 |
| Use Case: | Reach and Halt at Destination |
| Actor: | Ego Vehicle |
| Precondition | Vehicle is near the destination; final approach path is defined. |
| Basic Flow | 1. System identifies the final approach path. 2. System gradually decelerates the vehicle. 3. System maneuvers the vehicle to align with the designated stopping point. 4. System brings the vehicle to a complete stop at the destination. |
| Alternative Flow | 1. If the designated stopping point is obstructed:  1.1. System identifies near stopping point.  1.2. System re-evaluates the approach path.  1.3. System brings the vehicle to a complete stop at the new location. |
| Post Condition | Vehicle has stopped at the destination. |

*Table 4.12 - Detailed Use-case: Reach and Halt at Destination*

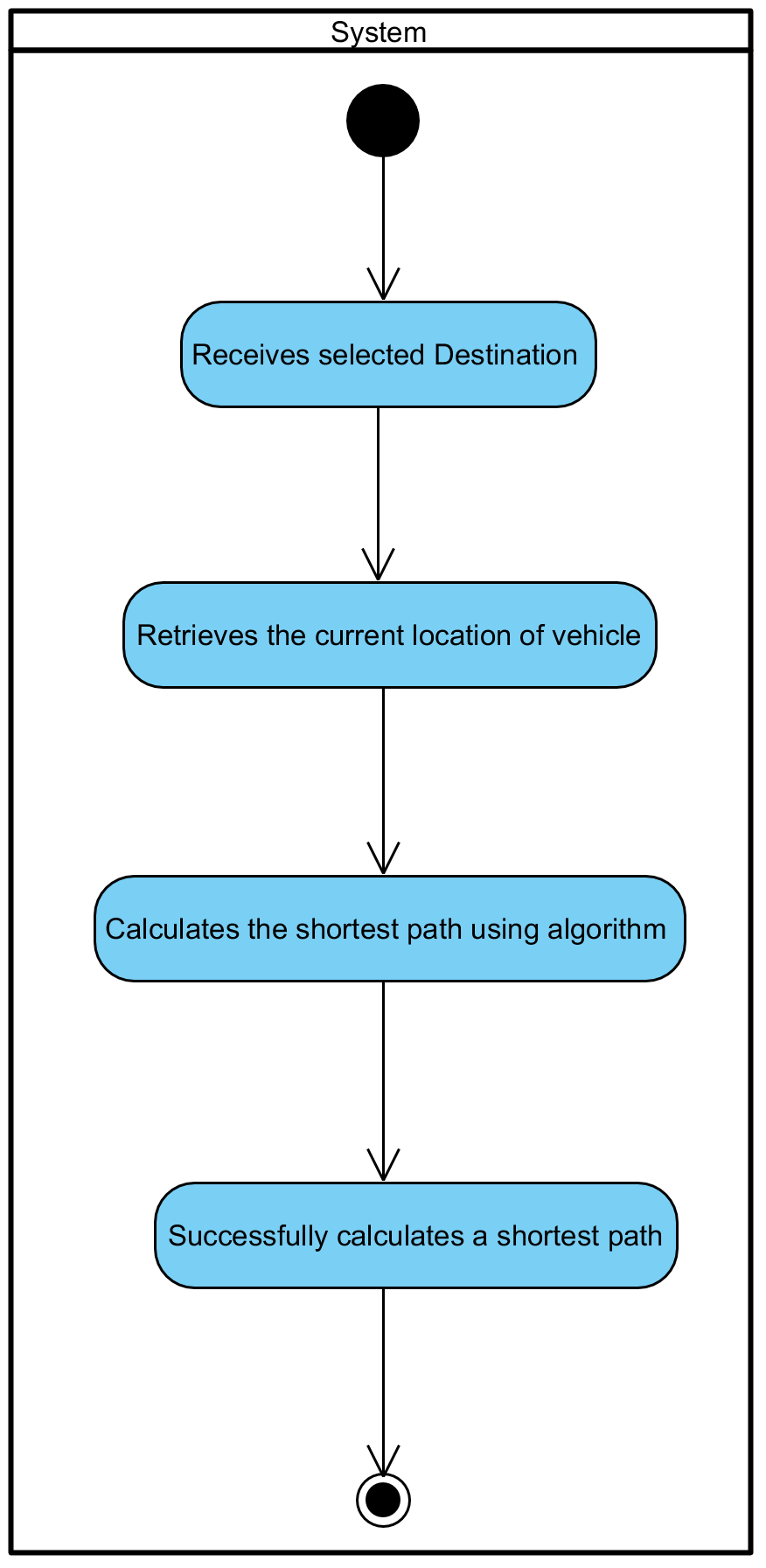
### Activity Diagrams

* + - 1. **Set Destination:**



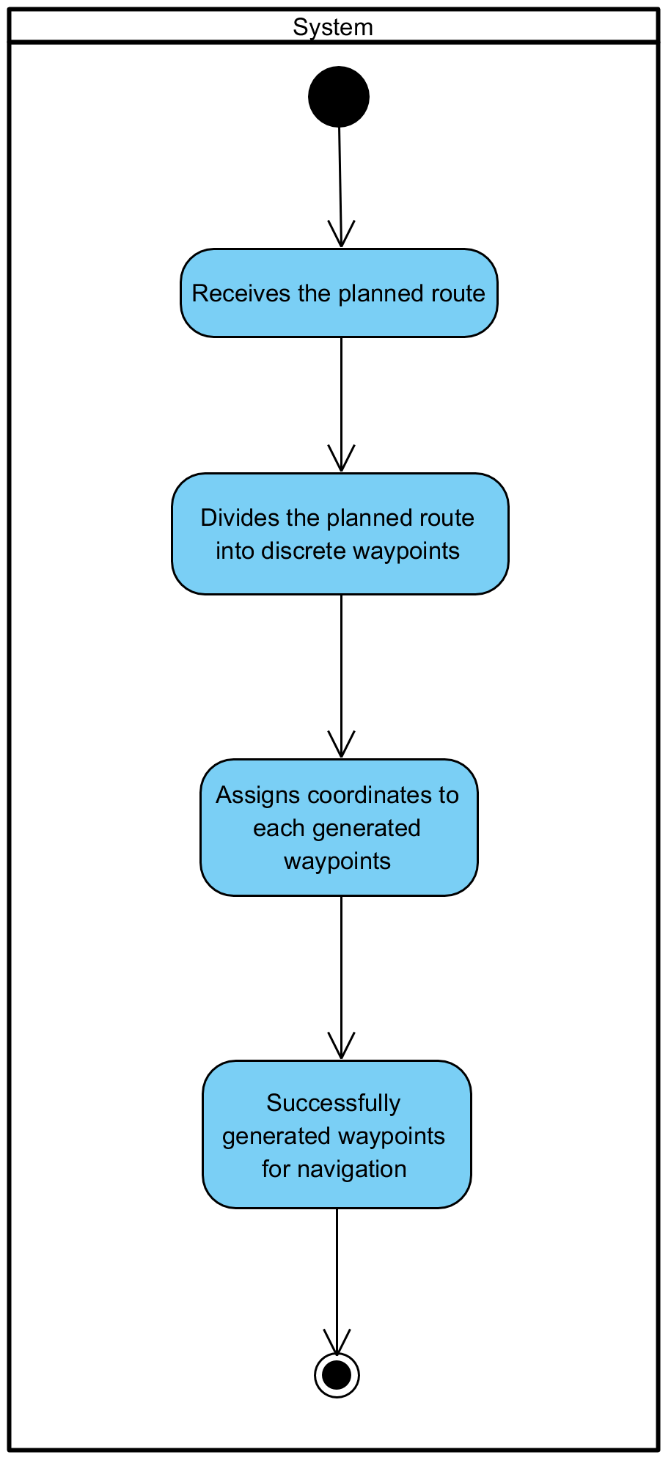
*Fig 4.3.3.1 – Set Destination*

* + - 1. **Plan Route:**



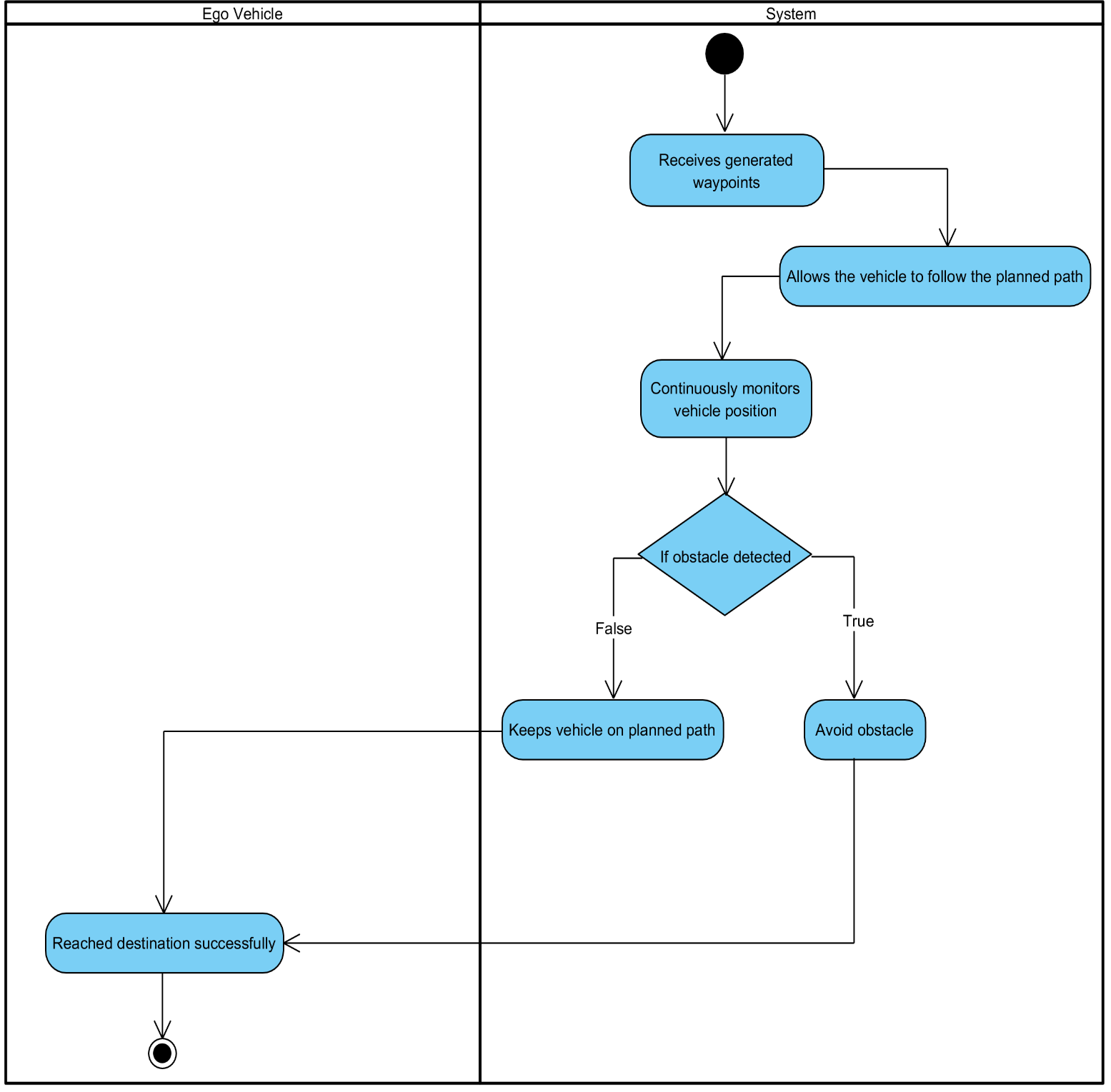
*Fig 4.3.3.2 – Plan Route*

* + - 1. **Generate Waypoints:**



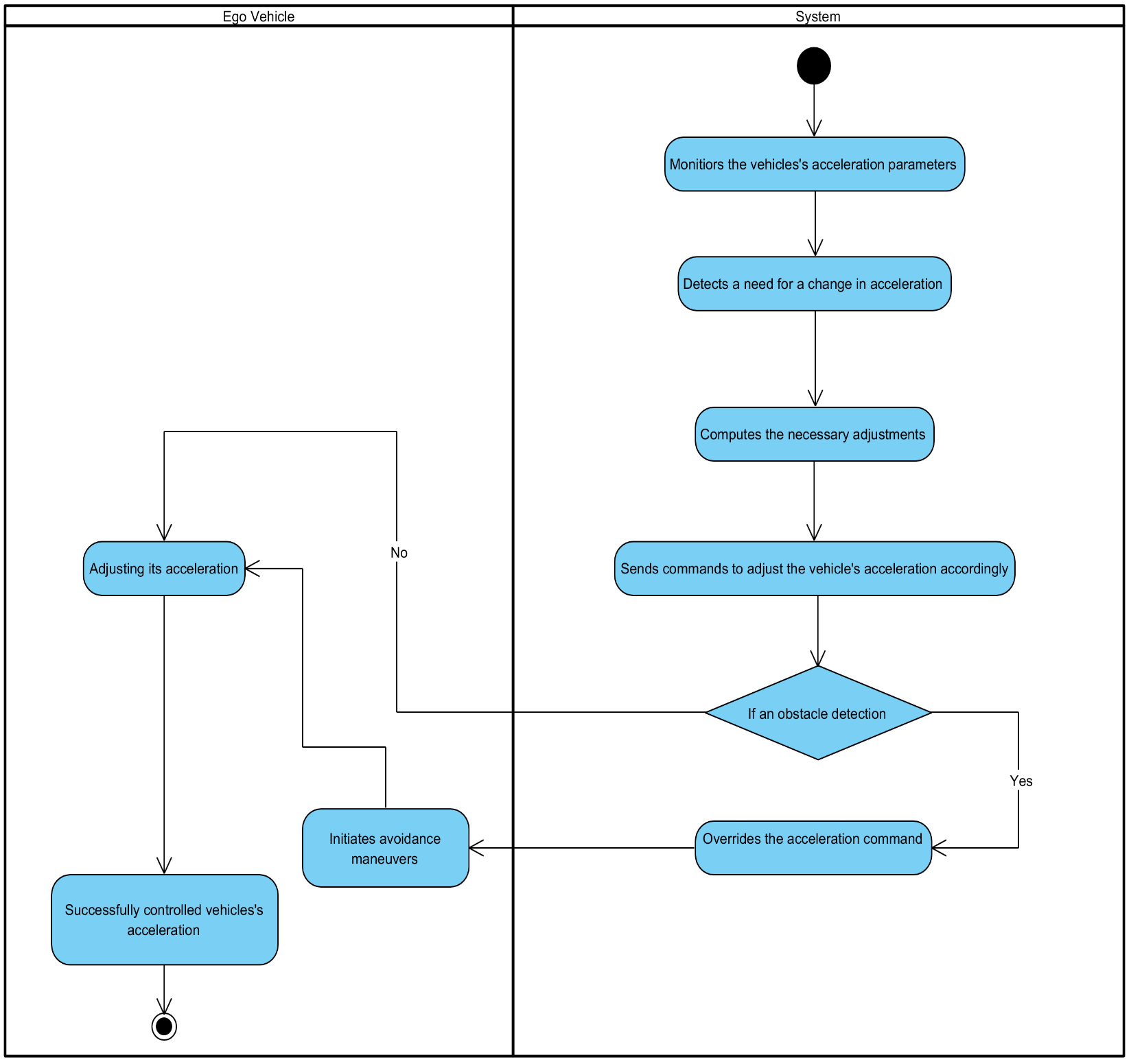
*Fig 4.3.3.3 – Generate Waypoints*

* + - 1. **Navigate Generated Waypoints:**



*Fig 4.3.3.4 – Navigate Generated Waypoints*

* + - 1. **Control Acceleration:**

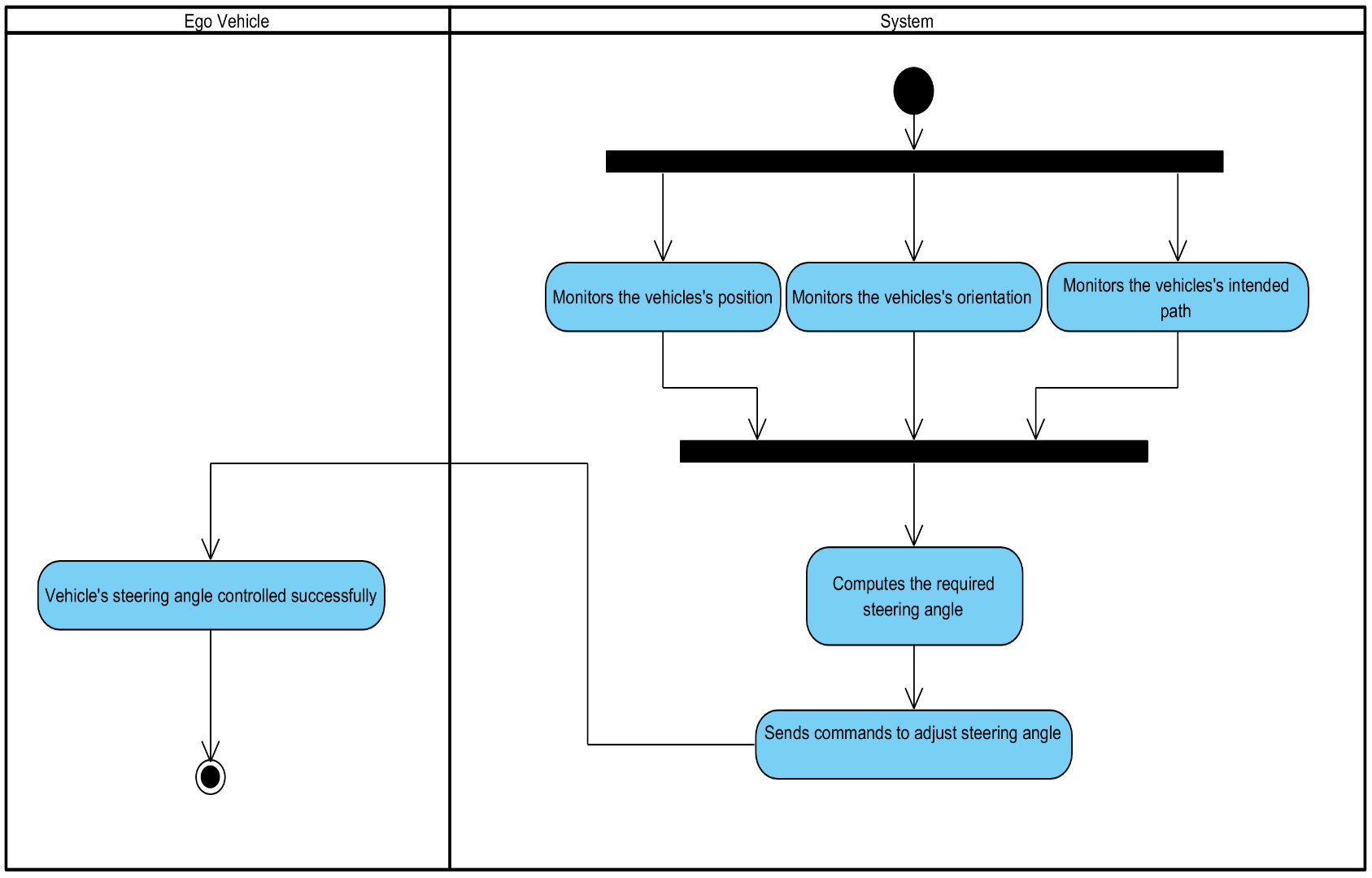


*Fig 4.3.3.5 – Control Acceleration*

* + - 1. ****Control Throttle:

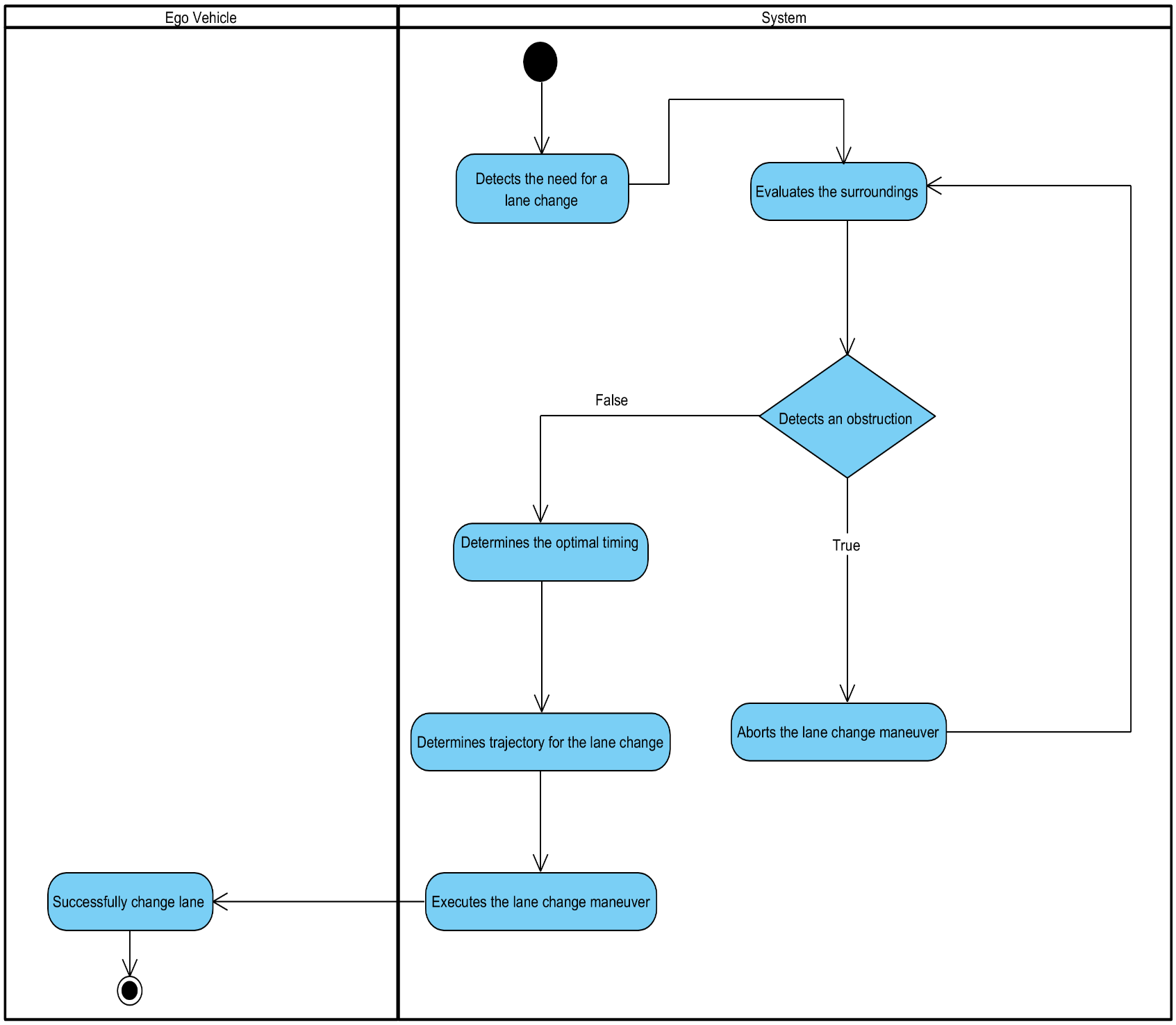
*Fig 4.3.3.6 – Control Throttle*

* + - 1. **Control Steering:**

****

*Fig 4.3.3.7 – Control Steering*

* + - 1. **Assign Lane:**

****

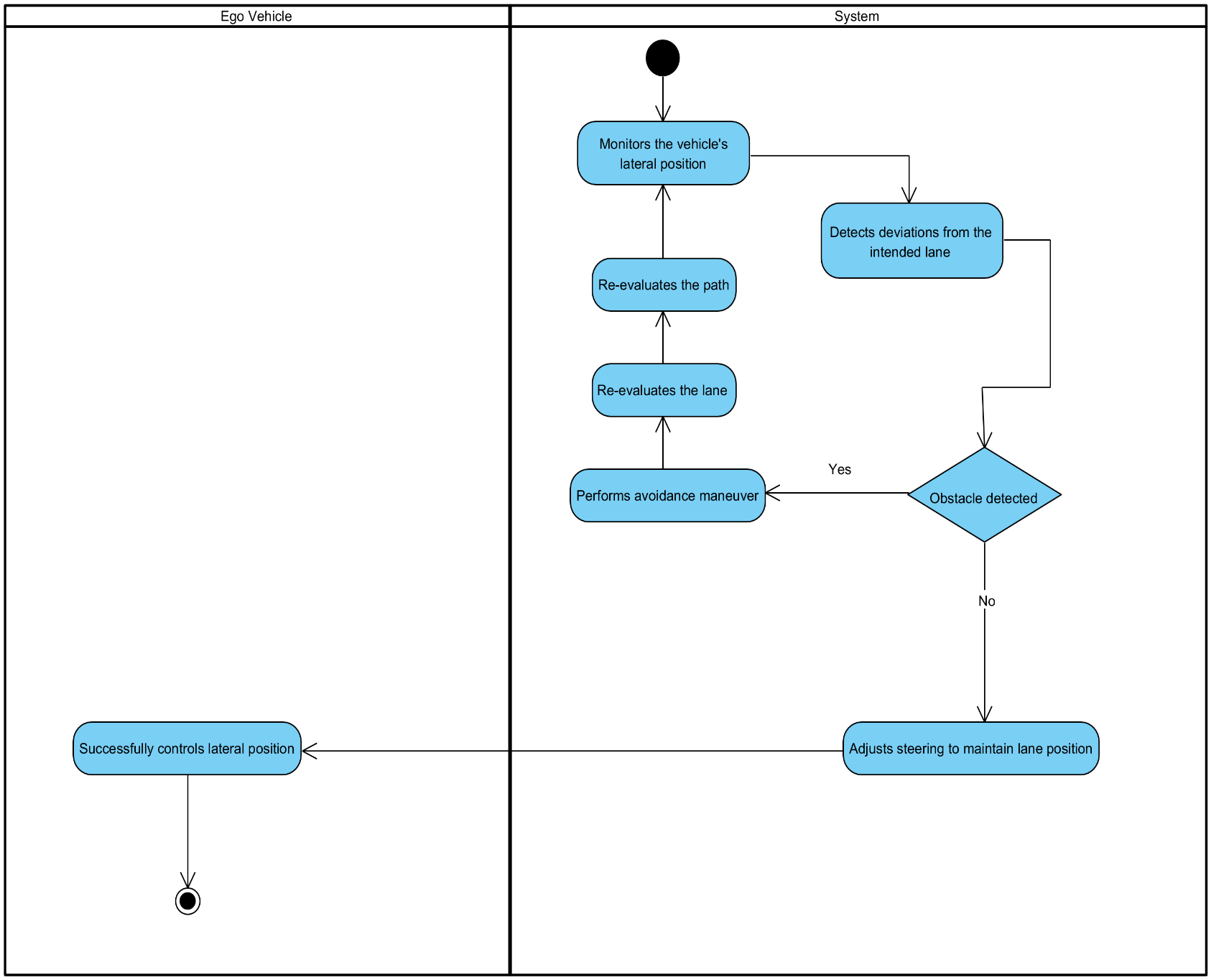
*Fig 4.3.3.8 – Assign Lane*

* + - 1. **Control Longitudinal Movement:**

****

*Fig 4.3.3.9 – Control Longitudinal Movement*

* + - 1. **Control Lateral Movement:**

****

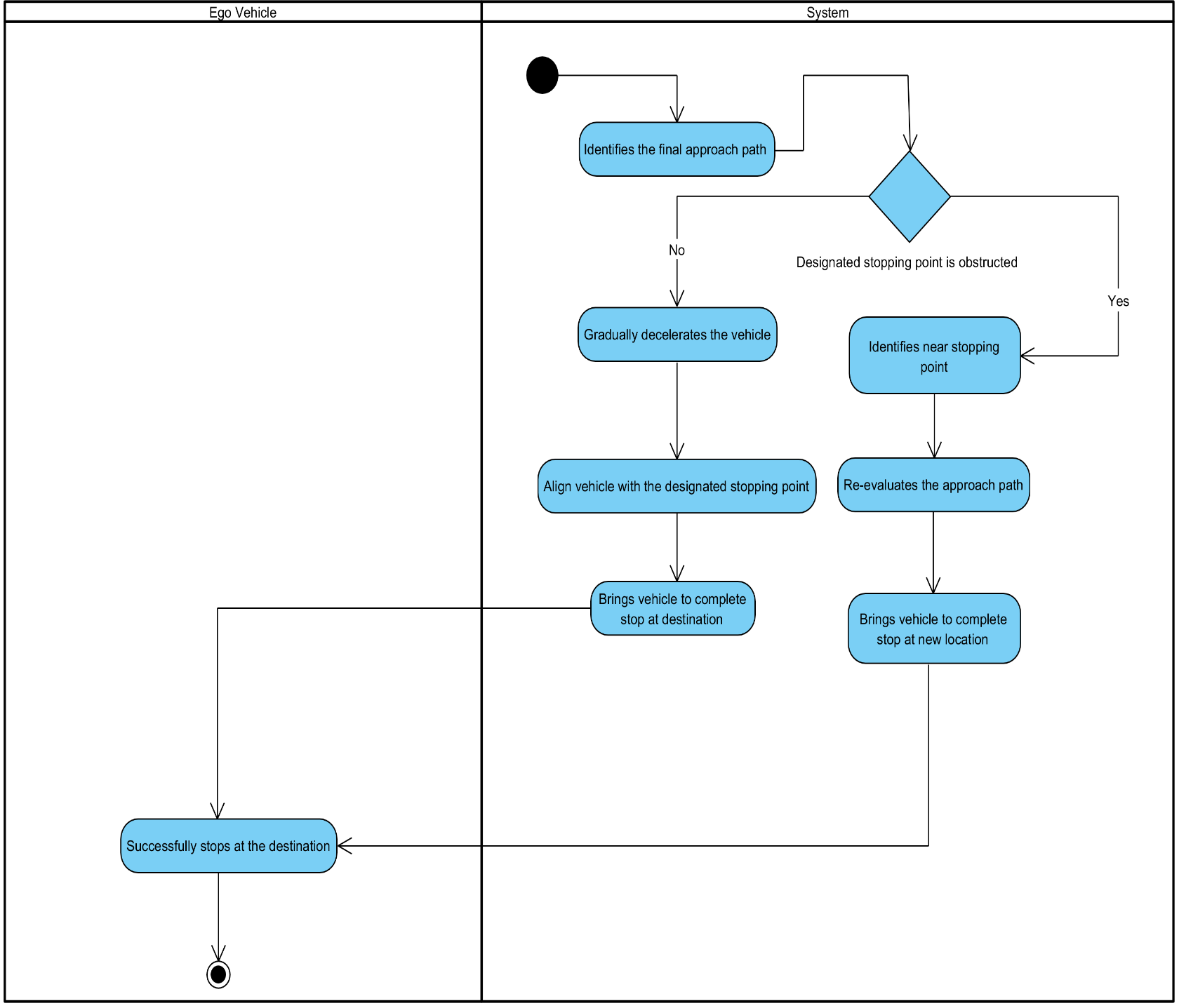
*Fig 4.3.3.10 – Control Lateral Movement*

* + - 1. **Control Jerkiness:**

****

*Fig 4.3.3.11 – Control Jerkiness*

* + - 1. **Reach & Halt at Destination:**



*Fig 4.3.3.12 – Reach & Halt at Destination*

### Sequence Diagram

*Figure 4.3.4.1 - Sequence diagram*

Chapter 5

**Implementation**

# Chapter 5: Implementation

## Endeavour

In the implementation phase, our team applies rigorous software engineering principles. We plan and execute each task, adhering to industry best practices for reliability. From architectural design to testing, our approach reflects our commitment to delivering high-quality software solutions

* + 1. **Team**
* Bilal Rafiq
* Hamza Azhar
* Sardar Mohsin Saghir
* Muhammad Usama Nazir
  + 1. **Work Breakdown Structure**

1. **Project Management**
   1. Work Breakdown Structure (WBS)
   2. Roles & Responsibility Matrix
   3. Change Control Plan
   4. Meeting minutes and Progress report
2. **Reports / Documentation**
   1. Team Members and Project Proposal
   2. Project Proposal Document
      1. Opportunity and Stakeholders
      2. Challenges Goals and Objectives
      3. Solution Overview diagram
      4. Report Outline
   3. Literature / Market Survey
      1. Domain Expert Interview Findings
      2. Questionnaire for Technical Feasibility and Risk Assessment
      3. Brainstorming diagram
      4. Academic Research Review
      5. Gap analysis summary
      6. Technology Landscape
         1. SWOT analysis
      7. Questionnaire for Selecting tools and techniques
      8. Specialization - 4 courses series from Coursera
   4. Requirement Analysis
      1. Problem Scenarios
      2. Requirement Elicitation
      3. Questionnaire for gathering requirements
      4. Functional Requirements
      5. Non-Functional Requirement
      6. Inspection Report
      7. Software requirement specification artifact
   5. System Design
      1. Architecture Diagram
      2. Use Case Diagram
      3. Detail Use Cases
      4. Activity Diagrams
      5. System Sequence Diagram
   6. Implementation
      1. Components and Libraries
   7. Testing and Performance Evaluation
      1. Test Scenarios
   8. Conclusion & Outlook
      1. Future Recommendations
   9. Progress Presentation
      1. Slides outlining project progress
      2. Updated Artifacts
         1. Appendix-A: Software Requirements Specifications (SRS)
         2. Appendix-B: Design Documents
         3. Appendix-C: Coding Standards/Conventions
         4. Appendix-D: Test Scenarios
         5. Appendix-E: Work Breakdown Structure
         6. Appendix-F: Roles & Responsibility Matrix
      3. Answers to potential questions report
   10. Final Presentation
       1. Comprehensive Slides for presentation
       2. Working software system (Complete)
       3. Updated Artifacts (Complete)
          1. Appendix-A: Software Requirements Specifications (SRS)
          2. Appendix-B: Design Documents
          3. Appendix-C: Coding Standards/Conventions
          4. Appendix-D: Test Scenarios
          5. Appendix-E: Work Breakdown Structure
          6. Appendix-F: Roles & Responsibility Matrix
       4. Consent Form
       5. Final Report
3. **System**
   1. Development Environment
      1. IDE
         1. Visual Studio Code
         2. PyCharm
      2. Version Control
         1. Git Hub
      3. Environment Management
         1. Anaconda Distribution
   2. Simulation Environment Setup
      1. CARLA Simulator
         1. Carlaviz for CARLA Visualization
      2. ROS Noetic Configured
      3. CARLA-ROS Bridge Integrated
      4. Vehicle spawn module
      5. Sensor spawn module
      6. Destroy Vehicle module
   3. Path Planning component
      1. Map Reading module
      2. Graph of Roads
      3. Graph of Lanes
      4. List of Driving Lanes within map
      5. Route Calculation module
      6. Algorithm implementation module
      7. Global route planner module
      8. Axis Translation module
      9. Local route planner module
      10. Environment Analysis module
      11. Trajectory Generation module
      12. Junction handling module
   4. Path Following and Vehicle Control component
      1. Trajectory Tracking module
      2. Basic agent module
      3. Behaviour agent module
      4. Algorithm implementation module
      5. Controller module
      6. Throttle Control module
      7. Braking Control module
      8. Acceleration Control module
      9. Steering Control module
      10. Longitudinal Control module
      11. Lateral Control module
      12. Lane changing module
      13. Jerkiness Control algorithm modules
      14. Custom Destination module
      15. Rotation and Translation module
   5. Sensor Integration module
      1. IMU integration sub-module
      2. GPS integration sub-module
      3. Radar integration sub-module
      4. Lidar integration sub-module
   6. Obstacle Detection
      1. Sensor Fusion module
         1. Lidar-Radar Fusion sub-module
         2. Multi-sensor Data synchronization sub-module
      2. Sensor Data Processing module
      3. Obstacle Detection module
         1. ML based detection sub-module
      4. Distance Estimation module
      5. Object Classification module
   7. Obstacle Avoidance
      1. Dynamic Obstacle handling module
      2. Static Obstacle handling module
      3. Path Adjustment module
         1. Map based planning sub-module
         2. Graph based planning sub-module
      4. Trajectory Estimation module
      5. Maneuver Planning module
         1. Environmental evaluation sub- module
         2. Lane changes sub-module
         3. Decelerate sub-module
         4. Emergency Stop sub-module
      6. Real-time Response module
      7. Tracking module
         1. Kalman filter sub-module
         2. Particle filter sub-module
4. **Open House Event**
   1. Standee Design and Print
   2. Printed Broachers
   3. Pre-recorded Demo video
      1. **Roles & Responsibility Matrix:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **WBS#** | **WBS Deliverable** | **Activity #** | **Activity to complete the deliverable** | **Duration (days)** | **Responsible Team Member(s) & Role(s)** |
| 1 | Project Initiation Phase | 1 | Literature Review | 7 | Bilal (A)  Hamza (R)  Mohsin (I)  Usama (R) |
|  |  | 2 | Define project scope and objectives | 5 | Bilal (A/R)  Hamza (C)  Mohsin (C)  Usama (I) |
|  |  | 3 | Establish project team roles and responsibilities | 1 | Bilal (A/R)  Hamza (C)  Mohsin (I)  Usama (I) |
|  |  | 4 | Setup project management tools and communication channels | 1 | Bilal (C)  Hamza (A)  Mohsin (I)  Usama (R) |
| 2 | Requirement Analysis | 5 | Research existing autonomous vehicle technologies and solutions | 3 | Bilal (C)  Hamza (A/R)  Mohsin (I)  Usama (I) |
|  |  | 6 | Gather requirements from stakeholders | 5 | Bilal (A)  Hamza (R)  Mohsin (C)  Usama (C) |
|  |  | 7 | Brainstorming | 2 | Bilal (R)  Hamza (A)  Mohsin (C)  Usama (C) |
|  |  | 8 | Define Problem Scenarios | 1 | Bilal (R)  Hamza (A)  Mohsin (C)  Usama (I) |
|  |  | 9 | Interview Domain Expert | 2 Meetings per week | Bilal (A)  Hamza (R)  Mohsin (I)  Usama (I) |
|  |  | 10 | Define Functional Requirements | 4 | Bilal (R)  Hamza (A)  Mohsin (C)  Usama (I) |
|  |  | 11 | Specify Non-Functional Requirement | 1 | Bilal (A/R)  Hamza (C)  Mohsin (I)  Usama (I) |
|  |  | 12 | System Overview | 2 | Bilal (C)  Hamza (R)  Mohsin (I)  Usama (A) |
|  |  | 13 | Constraints | 1 | Bilal (A/R)  Hamza (C)  Mohsin (I)  Usama (I) |
| 3 | System Design | 14 | Develop Architecture Diagram | 1 | Bilal (C)  Hamza (A/R)  Mohsin (I)  Usama (C) |
|  |  | 15 | Create Use Case Diagram | 1 | Bilal (C)  Hamza (A/R)  Mohsin (R)  Usama (I) |
|  |  | 16 | Define Detail Use Cases | 3 | Bilal (A) Hamza (R) Mohsin (I)  Usama (C) |
|  |  | 17 | Design Activity Diagrams | 3 | Bilal (C) Hamza (I)  Mohsin (A/R)  Usama (I) |
|  |  | 18 | Construct System Sequence Diagram | 1 | Bilal (C)  Hamza (A)  Mohsin (R)  Usama (I) |
| 4 | Simulation Environment Setup | 19 | Install and configure CARLA simulator, ROS Noetic and environment | 8 | Bilal (A)  Hamza (C)  Mohsin (I)  Usama (R) |
|  |  | 20 | Develop scripts for setting up simulation scenarios | 7 | Bilal (A/R)  Hamza (C)  Mohsin (I)  Usama (I) |
|  |  | 21 | Verify integration between CARLA and ROS | 1 | Bilal (A)  Hamza (R)  Mohsin (I)  Usama (I) |
| 5 | Path Planning Algorithm Development | 22 | Defining algorithms for path planning considering dynamic obstacles | 3 | Bilal (A/R)  Hamza (C)  Mohsin (C)  Usama (I) |
|  |  | 23 | Path planning logic in Python using ROS | 20 | Bilal (A)  Hamza (R)  Mohsin (I)  Usama (C) |
|  |  | 24 | Route Calculation | 5 | Bilal (C)  Hamza (A)  Mohsin (I)  Usama (R) |
|  |  | 25 | Map Processing | 1 | Bilal (A)  Hamza (I)  Mohsin (C)  Usama (R) |
|  |  | 26 | Environment Analysis | 2 | Bilal (A)  Hamza (R)  Mohsin (I)  Usama (C) |
|  |  | 27 | Trajectory Generation | 4 | Bilal (C)  Hamza (I)  Mohsin (R)  Usama (A) |
|  |  | 28 | Calculating Waypoints | 2 | Bilal (A)  Hamza (C)  Mohsin (I)  Usama (R) |
|  |  | 29 | Test path planning algorithms in simulated environments | 3 | Bilal (A)  Hamza (R)  Mohsin (I)  Usama (C) |
| 6 | Path Following Implementation | 30 | Defining control algorithms for vehicle control | 2 | Bilal (A/R)  Hamza (R)  Mohsin (I)  Usama (C) |
|  |  | 31 | Integrate path following logic/algorithm | 7 | Bilal (R)  Hamza (A)  Mohsin (C)  Usama (I) |
|  |  | 32 | Trajectory Tracking | 2 | Bilal (A)  Hamza (R)  Mohsin (C)  Usama (I) |
|  |  | 33 | Velocity Control | 3 | Bilal (A)  Hamza (C)  Mohsin (I)  Usama (R) |
|  |  | 34 | Steering Control | 5 | Bilal (C)  Hamza (A)  Mohsin (I)  Usama (R) |
|  |  | 35 | Conduct testing and validation in simulated environments | 5 | Bilal (C)  Hamza (R)  Mohsin (I)  Usama (A) |
| 7 | Obstacle Detection | 36 | Defining Machine Learning algorithms for detecting obstacles | 3 | Bilal (C)  Hamza (I)  Mohsin (A/R)  Usama (I) |
|  |  | 37 | Sensor Data Processing | 5 | Bilal (C)  Hamza (I)  Mohsin (A/R)  Usama (I) |
|  |  | 38 | Obstacle Detection | 7 | Bilal (A)  Hamza (C)  Mohsin (R)  Usama (I) |
|  |  | 39 | Distance Estimation | 5 | Bilal (C)  Hamza (A)  Mohsin (R)  Usama (I) |
| 8 | Obstacle Avoidance | 40 | Defining avoidance Maneuver | 1 | Bilal (C)  Hamza (A)  Mohsin (R)  Usama (I) |
|  |  | 41 | Implement obstacle avoidance strategies | 25 | Bilal (C)  Hamza (R)  Mohsin (A)  Usama (I) |
|  |  | 42 | Path Adjustment | 10 | Bilal (C)  Hamza (A)  Mohsin (R)  Usama (I) |
|  |  | 43 | Maneuver Planning | 5 | Bilal (C)  Hamza (I)  Mohsin (A)  Usama (R) |
|  |  | 44 | Real Time Responding | 5 | Bilal (I)  Hamza (C)  Mohsin (A/R)  Usama (C) |
|  |  | 45 | Integrate obstacle detection and avoidance with overall system | 5 | Bilal (C)  Hamza (I)  Mohsin (R)  Usama (A/R) |
| 8 | Sensor Integration and Calibration | 46 | Integrate sensors with the autonomous vehicle in simulation | 2 | Bilal (A)  Hamza (C)  Mohsin (I)  Usama (R) |
|  |  | 47 | Calibrate sensor data for accurate perception | 6 | Bilal (A)  Hamza (C)  Mohsin (R)  Usama (I) |
|  |  | 48 | Validate sensor data in simulated and real-world scenarios | 7 | Bilal (C)  Hamza (A/R)  Mohsin (R)  Usama (I) |
| 9 | System Integration | 49 | Integrate all software components into the autonomous vehicle system | 5 | Bilal (I)  Hamza (R)  Mohsin (C)  Usama (A/R) |
| 10 | Simulated Testing | 50 | Conduct comprehensive testing | 6 | Bilal (I)  Hamza (A/R)  Mohsin (C)  Usama (R) |
|  |  | 51 | Iterate on software development based on testing feedback | 2 | Bilal (R)  Hamza (I)  Mohsin (A)  Usama (C) |
|  |  | 52 | Fine-tune algorithms and software based on testing results | 3 | Bilal (C)  Hamza (A/R)  Mohsin (R)  Usama (I) |
| 11 | Optimization and Finalization | 53 | Optimize software performance and efficiency | 2 | Bilal (C)  Hamza (R)  Mohsin (A)  Usama (I) |
|  |  | 54 | Address any remaining issues or bugs | 1 | Bilal (I)  Hamza (C)  Mohsin (R)  Usama (A/R) |
|  |  | 55 | Finalize the project documentation and deliverables | 2 | Bilal (A/R)  Hamza (C)  Mohsin (C)  Usama (C) |

## Components and Libraries

**Components:**

* Map Parser
* Traffic Generator
* Path Planner
* Trajectory Follower
* Obstacle Detector
* Obstacle Avoider
* Localization Module
* Sensor Data Fusion
* Control System
* Decision-Making Module
* Simulation Environment

**Libraries:**

* rospy (for ROS-based development)
* NumPy (for numerical computations)
* math (for mathematical operations)
* keyword (for parsing Python keywords)
* xmltodict (for handling XML data)
* os (for interacting with the operating system)
* carla\_msgs (for CARLA-specific ROS messages)
* sensor\_msgs (for sensor-related ROS messages)
* OpenCV (for computer vision tasks)
* Pandas (for data manipulation and analysis)
* Matplotlib (for data visualization)
* TensorFlow or PyTorch (for deep learning, if applicable)
* Gazebo (for simulation, if using alongside CARLA)
* RViz (for visualization in ROS)

## IDE, Tools and Technologies

1. **IDE**

* PyCharm
* Visual Studio Code

1. **Tools**

* Anaconda
* Ubuntu
* GitHub
* Jira
* Microsoft office
* Visual Paradigm
* Docker

1. **Technologies**

* Carla Simulator
* Carla-Ros-Bridge
* ROS Noetic
* Rospy
* Gazebo
* Robot\_localization
* Python

## Best Practices / Coding Standards

* + 1. **Software Engineering Practice**

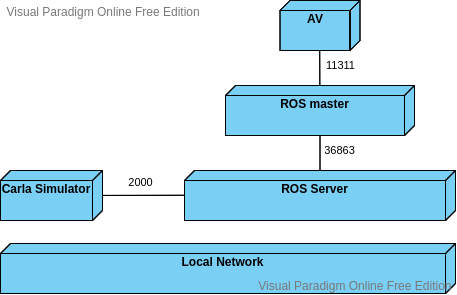
In our project, we implemented the Kanban methodology to efficiently manage our workflow and adapt to changing requirements. Utilizing Jira as our project management tool, we maintained a visual Kanban board to track tasks and their progress seamlessly. Our approach included:

* Regular Supervisor Meetings
* Continuous Workflow
* Backlog Management
  + 1. **Python coding Standards**
* Use snake\_case for variable and function names.
* Use CamelCase for class names.
* Follow PEP 8 guidelines for code formatting.
* Use meaningful variable and function names.
* Keep lines of code within 79 characters.
* Use comments to explain complex parts of the code.
* Use docstrings to document modules, classes, and functions.
* Avoid using global variables unless necessary.
* Handle exceptions gracefully.
* Use virtual environments to manage dependencies.
  + 1. **Rospy coding Standards**
* Follow Python coding standards for rospy code.
* Use rospy naming conventions for nodes, topics, and services.
* Utilize rospy log functions for logging messages.
* Ensure ROS dependencies are properly declared in package.xml and CMakeLists.txt.
* Document ROS nodes, topics, and services using ROS comments.
* Use rospy's rospy.spin() to keep the node alive.
* Handle ROS messages and services according to their specifications.
* Use rospy's parameter server for managing node parameters.
* Implement proper error handling for ROS communication.

## Deployment Environment

A local server where Carla Simulator and Autonomous Vehicle software system is running, communicating with the help of Carla Ros bridge.

* + 1. **Deployment Diagram**

****

## Summary

In this chapter we have provided a list of components and libraries that we have used in our project for better user experience. We have mentioned Work breakdown structure WBS and Control flow diagram. We have also mentioned tools and IDEs and best practices and coding standards of software engineering.