Computer On Wheels Project
Software Requirements Specification
Version <7.0>

Revision History

Date	Version	Description
<03/13/24>	<1.0>	SRS 1.0
<03/20/24>	<2.0>	SRS 2.0
<04/05/24>	<3.0>	SRS 3.0
<04/17/24>	<4.0>	SRS 4.0
<04/22/24>	<5.0>	SRS 5.0
<04/29/24>	<6.0>	SRS 6.0
<10/23/24>	<7.0>	SRS 7.0

Table of Contents

Software Requirements Specification

1. Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. The aim of this document is to gather and analyze and give an in-depth insight of the complete **Computer On Wheels software system** by defining the problem statement in detail. Nevertheless, it also concentrates on the capabilities required by domain expert and stakeholders and their needs while defining high-level product features. The detailed requirements of the **Computer On Wheels** are provided in this document.

1.1 Purpose

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to consumers. Also, we shall predict and sort out how we hope this product will be used in order to gain a better understanding of the project, outline concepts that may be developed later, and document ideas that are being considered, but may be discarded as the product develops.

In short, the purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and hardware and software requirements. It defines how our client, team and audience see the product and its functionality. Nonetheless, it helps any designer and developer to assist in software delivery lifecycle (SDLC) processes.

1.2 Scope

Primarily, the scope pertains to the key functionalities required for the autonomous vehicle project to operate effectively. It focuses on integrating sensors and algorithms for environmental perception, path planning for optimal route determination, path following for precise vehicle control, and real-time obstacle detection and avoidance to ensure safe navigation.

This SRS aims to specify the software requirements to be developed, which can also assist in selecting suitable in-house and commercial software products. The standard can be used to create software requirements specifications directly or serve as a model for defining project-specific standards, without identifying any specific method, nomenclature, or tool for preparing an SRS.

1.3 Elicitation technique(s)

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.

1.4 Definitions, Acronyms, and Abbreviations

CARLA	Car Learning to Act
FAQ	Frequently Asked Questions
ROS	Robotic Operating System
IMU	Inertial Measurement Unit
GPS	Global Positioning System

1.5 Overview

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product. General description of the project is discussed in section 2 of this document. Section 3 gives the functional requirements and constraints and assumptions made while designing the product. Section 3 also gives the specific requirements of the product. Section 4 is for supporting information.

2. Overall Description

This document contains the problem statement that the current system is facing which is hampering the growth opportunities of the company. It further contains a list of the stakeholders and users of the proposed solution. It also illustrates the needs and wants of the stakeholders that were identified in the brainstorming exercise as part of the requirements workshop. Its further lists and briefly describes the major features and a brief description of each of the proposed system.

3. Specific Requirements

The specific requirements are –

3.1 Problem Scenarios

3.1 Scenarios

Table 3.1: problem statement 1

Problem Statement # 1: Safety Challenges	
The problem of	Inadequate safety measures for autonomous navigation in adverse weather conditions.
Affects	Passengers, pedestrians and other road users.
The result of which	Increased risk of accidents due to reduced visibility, leading to injuries or fatalities.
Benefits of	Improved safety protocols to ensure reliable operation of autonomous vehicles in varying environmental conditions.

Table 3.2: problem statement 2

Problem Statement # 2: Challenges in Simulation for AV Software Development	
The problem of	Inadequate utilization of simulation environments (CARLA) for developing and validating ROS-based autonomous vehicle software.
Affects	Developers and researchers in autonomous vehicle systems.
The result of which	Slower development cycles, higher costs of physical testing, and potential safety risks due to insufficient validation.
Benefits of	Enhanced efficiency and safety through thorough evaluation of autonomous algorithms under varied conditions before deployment.

Table 3.3: problem statement 3

Problem Statement # 3: User Acceptance Challenges for Autonomous Vehicles.	
The problem of	Limited user acceptance of autonomous vehicle technology due to perceived safety and reliability concerns.
Affects	Potential users, stakeholders, and the overall adoption of autonomous vehicles.
The result of which	Resistance to adopting autonomous vehicles, leading to slower market penetration and reduced investment in further development and innovation.
Benefits of	Building user confidence through improved transparency, education, and demonstrations of safety features in various conditions, which can enhance the acceptance and integration of autonomous vehicles in everyday life.

3.2 Functionality

Introduction –

This subsection contains the requirements for this product. These requirements are organized by the features discussed with domain expert. Features are then refined into use case diagrams and to activity diagram to best capture the functional requirements of the system. All these functional requirements can be traced.

3.2 Key Concepts and Terminology

To understand the requirements outlined in this chapter, it is important to be familiar with certain key concepts related to the **Robot Operating System (ROS)**, which serves as the framework for our embedded system.

3.2.1 ROS Overview

The Robot Operating System (ROS) is a middleware framework that is essential for managing complex data exchanges in autonomous systems like our **"Computer on Wheels."** It provides tools for creating **modular software components**, called **nodes**, that can communicate over defined channels known as **topics**.

3.2.2 ROS Nodes

Nodes are software modules that perform specific tasks. For example, a sensor node may detect obstacles, while a control node manages vehicle movement.

3.2.3 ROS Topics

Topics are the communication channels between nodes. Each topic is defined for a specific type of data exchange, such as publishing sensor readings or receiving control commands.

3.2.4 ROS Messages

Messages are the data structures used to communicate information over ROS topics. Each message type has a defined format and is used to transmit specific types of data, such as position coordinates or sensor measurements.

3.2.5 ROS Services

Services provide a way for nodes to request specific actions or information from each other, such as recalculating a path when an obstacle is detected.

This foundational understanding of ROS will facilitate comprehension of the requirements detailed in the subsequent sections.

3.3 Functional Requirements

3.3.1 Vehicle Control:

Table 3.4: FR1

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 using ROS-based navigation stacks.
1	**Vehicle Control**	**1.2**	Acceleration Control	The system shall control the vehicle's acceleration to maintain desired speeds along the planned trajectory, publishing commands to the topics in ROS.
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 by publishing to the dedicated ROS topic (`/emergency_stop`)
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, using PID controller implemented in ROS.
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, using ROS control messages.
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, publishing braking commands to ROS topic.

3.3.2 Path Planning:

Table 3.5: FR2

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 using ROS-based algorithms.
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 and map data.
2	**Path Planning**	**2.3**	Waypoint Generation	The system shall generate waypoints along the calculated route to guide the vehicle towards the destination, publishing waypoints to a ROS topic (`/waypoints`).
2	**Path Planning**	**2.4**	Dynamic Obstacle Avoidance	The system shall adapt the vehicle's path in real-time to safely avoid unexpected obstacles using ROS-based path adjustment algorithms.
2	**Path Planning**	**2.5**	Map Reading	The system shall be able to read and interpret digital map data using ROS

				to determine the vehicle's precise location within the road network

3.3.3 Path Following:

Table 3.6: FR3

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/s^2, 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 using ROS control loops.
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 route, using ROS topics to track progress towards each waypoint.

3.3.4 Sensor Integration:

Table 3.7: FR4

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, publishing data to ROS topics.
4	**Sensor Integration**	**4.2**	Global Positioning System Utilization	The system shall use GPS to determine the vehicle's position and publish coordinates to a ROS topic (`/gps_data`).
4	**Sensor Integration**	**4.3**	Radar/Lidar Utilization	The system shall utilize radar/lidar sensors to provide information about surrounding objects' velocity and distance, enhancing situational awareness through ROS topics

3.3.5 Trajectory Planning:

Table 3.8: FR5

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.

3.3.6 Obstacle Detection:

Table 3.9: FR6

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, integrating data through ROS topics (`/obstacle_detection`).
6	**Obstacle Detection**	**6.2**	Environmental Awareness	The system shall maintain awareness of static and dynamic objects in the vehicle's vicinity, using ROS-based perception modules.
6	**Obstacle Detection**	**6.3**	Dynamic Obstacle Tracking	The system shall continuously track moving obstacles, updating their positions through ROS messages.
6	**Obstacle Detection**	**6.4**	Destination Estimation	The system shall calculate the distance to detected obstacles and publish this data to a ROS topic (`/distance_to_obstacle`).

3.3.7 Obstacle Avoidance:

Table 3.10: FR7

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, using ROS-based planning and control.
7	**Obstacle Avoidance**	**7.2**	Steering Control	The system shall dynamically adjust steering angles to guide the vehicle away from obstacles, keeping it on its intended

				path using ROS.
7	**Obstacle Avoidance**	**7.3**	Re-Plan Path	The system shall re-plan the path once an obstacle is detected, updating the path through ROS services.
7	**Obstacle Avoidance**	**7.4**	Trajectory Adjustment	The system shall dynamically adjust the vehicle's trajectory to avoid obstacles in a clear environment using ROS algorithms.
7	**Obstacle Avoidance**	**7.5**	Multi-Obstacle Handling	The system shall manage avoidance of multiple obstacles simultaneously through ROS-based coordination.

3.3.8 Destination Arrival:

Table 3.11: FR8

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 using ROS.
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/s^2 for passenger safety and comfort.

3.3.9 User Inputs:

Table 3.12: FR9

No	Functional Requirement	Breakdown		Description
		ID	Sub-Functionality	
9	**User Inputs**	**9.1**	Ride Initiation	The system shall allow the user to initiate the autonomous driving process through a

				terminal command, which will start the ROS nodes required for vehicle navigation, control, and sensor integration. The command shall initiate the entire process of path planning, path following, and obstacle detection, ensuring all necessary components are activated before vehicle motion begins.
9	**User Inputs**	**9.2**	Destination Setting	The user shall be able to input the desired destination, triggering the route planning process through ROS services.

3.3.10 System Integration:

Table 3.13: FR10

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.

3.3.11 Traffic Light Module:

Table 3.14: FR11

No	Functional Requirement	Breakdown		Description
		ID	Sub-Functionality	
11	**Traffic Light Module**	**11.1**	**Traffic Light Detection**	The system shall detect traffic lights in the vehicle's path using camera-based sensors and publish the detected traffic light information to a ROS topic (`/traffic_light_detection`).
11	**Traffic Light Module**	**11.2**	Traffic Light State Recognition	The system shall recognize the state of detected traffic lights (red, yellow, green) using image processing algorithms within a ROS node, publishing the identified state to a topic (`/traffic_light_state`).
11	**Traffic Light Module**	**11.3**	Decision-Making Based on Traffic Light State	The system shall adjust vehicle behavior (e.g., deceleration, stopping, or proceeding) based on the recognized traffic light state, using data from the `/traffic_light_state` topic.
11	**Traffic Light Module**	**11.4**	Red Light Handling	Upon detecting a red-light state, the system shall bring the vehicle to a complete stop at a safe distance i.e., 1 meter from the traffic light, ensuring smooth deceleration.
11	**Traffic Light Module**	**11.5**	Green Light Handling	Upon detecting a green light state, the system shall resume vehicle motion and proceed along the planned path.
11	**Traffic Light Module**	**11.6**	Yellow Light Handling	Upon detecting a yellow light state, the system shall determine whether it is safe to proceed based on vehicle speed and distance to the traffic light, either decelerating to a stop or proceeding through the intersection.
11	**Traffic Light Module**	**11.7**	Traffic Light State Uncertainty	If the system cannot detect a traffic light state for more than 2 seconds, it shall trigger a safe

				stop and log an error message to a ROS topic (`/traffic_light_error`).

3.4 Non-Functional Requirement

Table 3.15: NFR1

No	Non- Functional Requirement	Subfactor	Verification Metric	Target Value
1	**Safety Requirement** 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.	**Detection Accuracy:** Measure the percentage of correctly detected objects in various weather conditions. **Response Time:** Time taken to respond to detected hazards. **Test Cases:** Conduct tests in simulated environments with varying weather scenarios (e.g., rain, fog, snow)	**Detection Accuracy:** ≥ 90% **Response Time:** ≤ 2 seconds in 95% of cases

Table 3.16: NFR2

No	Non- Functional Requirement	Subfactor	Verification Metric	Target Value
2	**Scalability Requirement** The ROS-based architecture shall support adding new sensors (e.g., radar, additional cameras) without significant changes	**System Expandability**	**Integration Time:** Measure the time taken to integrate a new sensor and update existing modules.	**Integration Time:** ≤ 10 minutes

	to the core modules.		**Compatibility Tests:** Perform tests to ensure new sensors can be added without affecting existing functionality. **Modularity Assessment:** Analyze the architectural design for dependencies that may hinder expansion.	

Table 3.17: NFR3

No	Non- Functional Requirement	Subfactor	Verification Metric
3.1	**Modularity Requirement** The system shall maintain a modular ROS node structure, separating perception, planning, and control into distinct nodes for ease of testing and modification.	**Software Architecture**	**Node Independence:** Verify that each node can be tested independently without affecting others. **Modification Time:** Measure the time required to modify or update a specific node.
3.2	**Modularity Requirement** Each ROS node shall handle a specific task (e.g., path planning, obstacle detection) and communicate through well-defined ROS topics.	**Task Separation**	**Message Latency:** Measure the time taken for messages to be published and received between nodes (≤ 100 ms). **Task Success Rate:** Evaluate the success rate of individual nodes in completing their specific tasks (≥ 95%).

3.3 Purchased Components

Not Applicable

3.4 Hardware Interfaces

Since the application is simulation based, no hardware will be included due to multiple constraints such as cost.

3.5 Software Interfaces

- **ROS Integration:** The system shall communicate with the Robot Operating System (ROS) to facilitate data exchange and coordination between different software components.
- **CARLA Simulator Integration:** The system shall integrate with the CARLA simulator to develop and test algorithms in a simulated environment before real-world deployment.
- **CARLA-ROS Bridge:** The system shall utilize the CARLA-ROS bridge for seamless integration between the CARLA simulator and ROS, enabling data exchange and control commands.
- **rospy Integration:** The system shall use rospy for Python-based ROS programming to develop and implement control algorithms, perception modules, and navigation strategies.

3.6 Licensing Requirements

Not Applicable

3.7 Applicable Standards

It shall be as per the automotive industry standard.

4. Supporting Information

Please refer the following document:

1. Use case analysis.
2. Detailed use case analysis.
3. Activity analysis.
4. WBS
5. Research papers