ENGG2000/3000 Department of Engineering, Macquarie University.

Scoping Document

Blade Runner 2024



**Table of Contents**

[**1 Introduction 5**](#_h09np7fr9w9x)

[1.1 Problem Statement 5](#_7e9s64dr16xq)

[1.2 Project Objectives 6](#_8oxm3t9j5c2)

[1.3 Subsystems 6](#_yjizmfh8swcb)

[1.4 Interdisciplinary Faculties 7](#_lopfp9nnlf88)

[1.5 Engineering Team 8](#_3pyisd7ix4cx)

[1.6 High-level Implementation 9](#_y22f3cgoxmtk)

[1.7 Preliminary design 10](#_y5dpg0yxpy0p)

[1.8 Project Constraint 11](#_5zis08dvmilf)

[**2 Scope of Project 12**](#_b1wtrsqz8zu7)

[2.1 Deliverables 12](#_3bzmrgzhybq2)

[2.2 Inclusions and Exclusions 13](#_ojckeuv7188f)

[2.3 Project Schedule 13](#_atefb6psniq6)

[**3 Requirements 14**](#_kqr37yfg7293)

[3.1 Functional Requirements 14](#_oaqt5p5w4wvj)

[3.2 Non-functional Requirements 15](#_bhr7in5d2whw)

[3.3 Interfacing Requirements 15](#_jkmjcfl7nbt)

[**4 Technical Performance Measures (TPMs)**](#_vm4pgwtm5uok) **17**

[4.1 Summary of TPMs 1](#_ycla4fgo5ab0)7

[4.2 List of TPMs 1](#_dj8wcu81606k)8

[**5 Design Structure Matrix (DSM) 2**](#_rt1dyt39pyhs)**4**

**List of Figures and Tables**

[Table i. Glossary 3](#_apxwaat9erfh)

[Table ii. Document version history. 4](#_scx2cypq3vh1)

[Table 1.1. List of contributors. 5](#_wanb5v1ikuff)

[Table 1.2: Subsystems for communications system](#_lszpjiu1dsxm) 6

[Table 1.3: Engineering team 8](#_ao59qcde6f84)

[Table 1.4: Class descriptions 11](#_qnsg624llx6g)

[Table 1.5: List of constraints 11](#_361wc8pqhcnh)

[Table 3.1. Functional requirements. 14](#_r5s7d2wexenq)

[Table 3.2. Acceptance criteria for the functional requirements. 1](#_5i9lfx5z4178)4

[Table 3.3. Non-functional requirements. 15](#_e94e4x3s5dd)

[Table 3.4. Acceptance criteria for the non-functional requirements. 15](#_uqoa5ngqxet4)

[Table 3.5. Interfacing requirements. 15](#_rzx2f5639zwx)

[Table 3.6. Acceptance criteria for the interfacing requirements. 1](#_i66evkw6p6cq)5

[Table 4.1. TPM Summary.](#_i66evkw6p6cq) 17

Table 4.2[: TPM01: Carriage Path Request Management 1](#_rtwdqadp4zgd)9

[Table 4.3:](#_icj166xubq2v) [TPM02:](#_rtwdqadp4zgd) [MCP Route Management and Execution](#_icj166xubq2v) 20

Table 4.4[:](#_tt35ac1076iu) [TPM03:](#_rtwdqadp4zgd) [Real-time route processing 2](#_tt35ac1076iu)1

[Table 4.5:](#_2trnz44waumw) [TPM04:](#_rtwdqadp4zgd) [Emergency Braking System Integration 2](#_2trnz44waumw)2

Table 4.6[:](#_qmd2ifbwmeyy) [TPM05:](#_rtwdqadp4zgd) [LED Status Indicator Accuracy 2](#_qmd2ifbwmeyy)3

Figure [1.1. High level implementation of system 9](#_zgyal3ae6gtm)

Figure 1.2. [UML diagram 10](#_wib87rfpjnam)

[Figure 2.1. Project schedule. 1](#_8s3hcmkqe9rk)3

**Glossary**

| **Term** | **Definition** |
| --- | --- |
| T3\_C1 | Tuesday comms group 1 |
| T3\_M1 | Tuesday motion group 1 |
| T3\_S1 | Tuesday structure group 1 |
| T3\_S5 | Tuesday structure group 5 |
| TPM | Technical Performance Measures |
| BR | Blade Runner |
| CCP | Carriage control program |
| MCP | Master control program |
| Wi-Fi | Wireless Fidelity |
| LED | Light-emitting diode |
| DSM | Design Structure Matrix |
| UI | User interface |
| UML | Unified modelling language |

#### Table i. Glossary

**Document History**

| **Version** | **Changes Made** | **Contributors** | **Date** |
| --- | --- | --- | --- |
| 1.0 | * Initial draft created. * Sections created: Introduction, Summary, Problem Definition, Requirements, Inclusions and Exclusions, Preliminary Design, and Appendices. | All | 06/08/24 |
| 1.1 | * Revised Requirements section. * Added additional details to the Preliminary Design | All | 06/08/24 |
| 1.2 | * Created the problem statement * Created the project objectives | Pratik | 12/08/24 |
| 1.3 | * Reorganised doc structure * Added to problem and objectives | Liam | 12/08/24 |
| 1.4 | * Added the Preliminary design   + Added updated class diagram | Wasif | 12/08/24 |
| 1.5 | * Added Scope of Project | Adya | 12/08/24 |
| 1.6 | * Add subsystems | Wasif | 13/08/24 |
| 1.7 | * Added requirements and the respective acceptance criteria   + Functional requirements   + Performance requirements   + Interface requirements | Mihir | 13/08/24 |
| 1.8 | * Updated Scope of project * Gantt chart | Pallika | 16/08/24 |
| 1.9 | * Added TPM’s for the requirements | Mihir & Pratik | 17/08/24 |
| 1.10 | * Update Preliminary design and Subsystem | Wasif | 18/08/24 |
| 1.11 | * Organised table of contents. Revised requirements section and overall document structure. | Liam | 18/08/24 |

#### Table ii. Document version history.

# 

# 1 Introduction

This report is an introductory document written by ENGG2000/3000 students in communications team T3\_C1 for the Blade Runner 2024 project that identifies the project’s scope, requirements, and objectives.

T3\_C1 is composed of driven undergraduate software engineering students aiming to deliver effective software solutions for real-world problems. Members include:

| **Name** | **OneID** | **Email** |
| --- | --- | --- |
| Liam Tan | 47505508 | liamisaacching.tan@students.mq.edu.au |
| Pratik Gurung | 47215607 | pratik.gurung@students.mq.edu.au |
| Mihir Sundriyal | 47228164 | mihir.sundriyal@students.mq.edu.au |
| Wasif zaman | 47422645 | wasif.zaman1@students.mq.edu.au |
| Adya Saanvi | 47525967 | adya.saanvi@students.mq.edu.au |
| Pallika Kafle | 46746218 | pallika.kafle@students.mq.edu.au |

#### Table 1.1. List of contributors.

## 

## 1.1 Problem Statement

The client, Macquarie University’s Faculty of Engineering, has tasked teams of student engineers to build a fully functioning monorail system with suspended “blade runners” running along a circular track. Originally divided into three specialisations (structure, motions, and communications), the teams were matched with each other to form a larger group where at least one of each specialisation is present. Then, the teams were instructed to work together within their groups to produce a fully functional, sensor-outfitted blade runner that runs on a provided monorail track and responds to WiFi-transmitted commands from a master control program. Upon completion of the project, the different groups’ blade runners will be placed on the track and run as a simulated transit system mirroring real-world train systems.

Team T3\_C1 will work with motions team T3\_M1 and structure teams T3\_S1 and T3\_S5 to produce a blade runner. As the communications team, T3\_C1 is responsible for writing the software that runs the blade runner’s moving parts, the carriage control program (CCP) and allows it to communicate with the master control program (MCP). To ensure the best possible quality, the team must write code that is reliable, modular, efficient, and compatible with the MCP. Communication with fellow communications teams is important, as the runners must also at least be aware of other nearby runners to avoid collisions.

## 1.2 Project Objectives

The following is a general view of objectives vital for the project’s completion.

* Writing scalable, modular, and efficient microcontroller code for the CCP, which includes control of:
  + The motors that allow the blade runner to move along the monorail,
  + The motors that open and close the blade runner’s doors,
  + The Wi-Fi module that allows the blade runner to send and receive data to and from the MCP,
  + The LED indicators,
  + The sensors that enable collision detection, and
  + The sensor that identifies stations the blade runner is pulling into.
* Writing submittable documents that report the team’s progress during different phases of the project.

## 

## 1.3 Subsystems

| **Subsystem** | **Responsibilities** |
| --- | --- |
| **Carriage** | * Manages the physical movement of the Blade Runner. * Controls speed, acceleration, and door operations. * Monitors sensor data to detect obstacles and movement status. |
| **Communications (Server)** | * Handles network communication between the Blade Runner and external interfaces. * Manages WiFi connections and data transmission. * Processes requests from the UI and sends appropriate response |
| **MCP (Master Control Program)** | * Observes and updates the state of the system based on the inputs from the Carriage. * Acts as the central processing unit that coordinates subsystem activities |
| **Logger** | * Logs events and errors occurring within the system. * Provides error handling and debugging information. * Ensures that all significant actions and issues are recorded for analysis. |
| **Route Planning** | * Handles the planning and decision-making for the Blade Runner's movements. * Determines the optimal paths and strategies for navigation. * Interacts with the Carriage to execute the planned routes. |
| **UI (User Interface)** | * Provides an online interface for users to monitor and control the Blade Runner. * Displays real-time system status and allows command inputs and Sends user commands to the Controller for processing. |
| **Controller** | * Acts as an intermediary between the UI and the Server. Processes user inputs from the UI and translates them into system commands. * Manages the system's current state and ensures it is accurately reflected in the UI. |

#### Table 1.2: Subsystems for Communications system

## 

## 1.4 Interdisciplinary Faculties

The Blade Runner engineering project is a comprehensive initiative involving the collaboration of multiple interdisciplinary teams, each responsible for critical subsystems. These subsystems are subdivided into three teams: communications (Comms), structures (Struct), and motions. Communication is responsible for maintaining the necessary protocol, for example, TCP, and ensuring the execution of commands runs seamlessly for Blade Runner. The Motions team is responsible for designing and implementing precise movement mechanisms, such as opening doors and stopping at the stations, ensuring that Blade Runner can navigate and perform tasks effectively. Meanwhile, the Structures team oversees the physical framework that houses all necessary components.

## 1.5 Engineering Team

Each team plays a distinct and vital role in contributing to the holistic development of the Blade Runner. This collaborative effort ensures that Blade Runner operates efficiently and effectively, achieving the project’s overarching goals. The motions team is responsible for designing and implementing precise movement mechanisms, ensuring Blade Runner can navigate and perform tasks effectively. This can be seen in Table 1 of each engineering team.

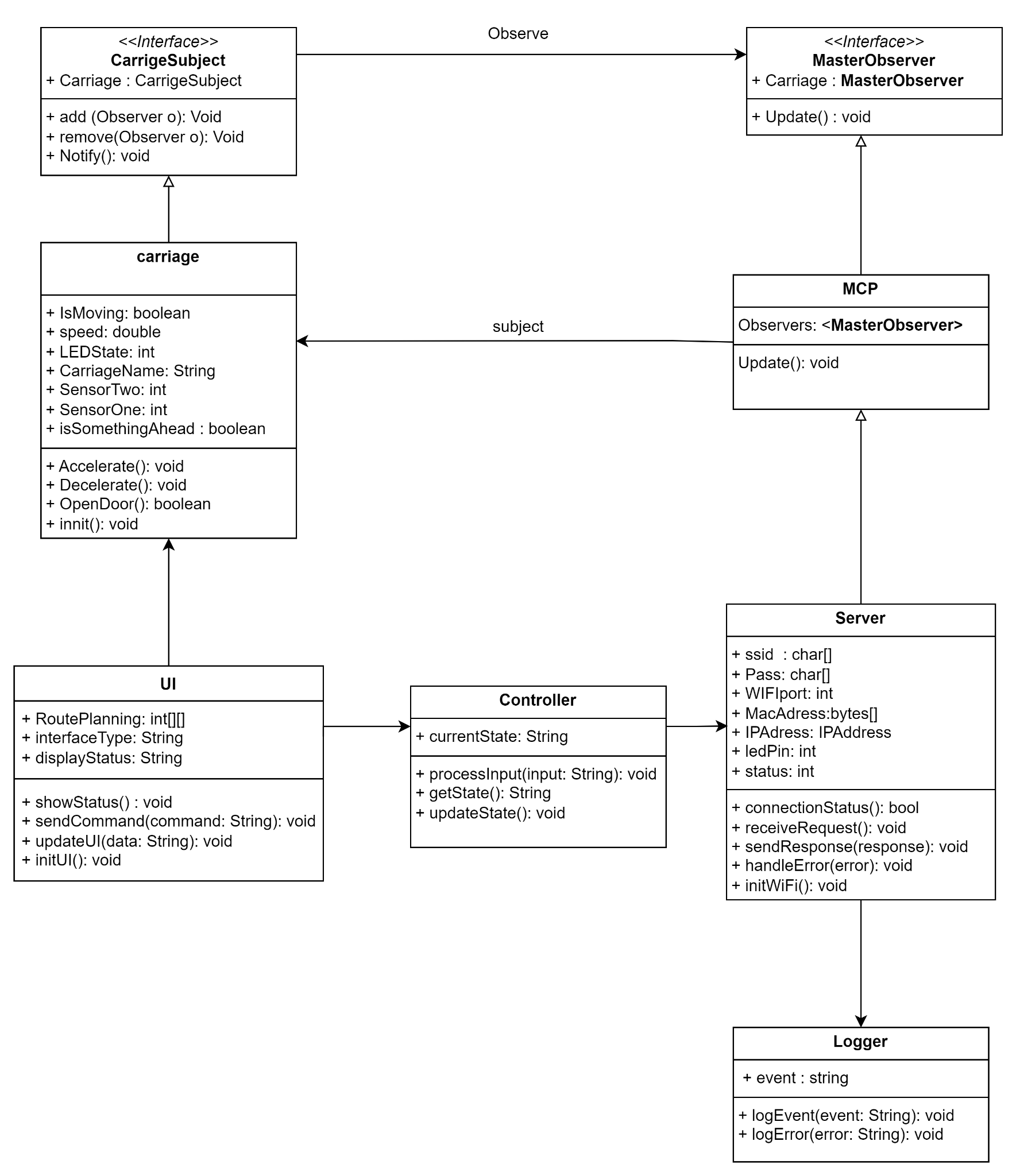
| **Engineering faculty** | **Responsibilities** |
| --- | --- |
| **Comms** | The system that facilitates the carriage and execution of the program’s instructions, ensuring that commands are transmitted and received accurately across the system. |
| **Motions** | Their goal is to create mechanisms and control systems that enable precise and responsive motions |
| **Structure** | Designing and constructing the physical framework of the Blade Runner. |

#### Table 1.3: engineering team

## 1.6 High-level Implementation

#### Figure 1.1. High level implementation of system

## 1.7 Preliminary Design

****

#### Figure 1.2. UML diagram

| **Class Name** | **Description** |
| --- | --- |
| **Carriage** | Controls the BR physical state and control of the BR, including movement, speed, and sensors. following significant components:   * IsMoving (boolean): This is used to change the state of the Blade Runner's acceleration. * isSomethingAhead (boolean): This is used to detect a possible collision with another Blade Runner. |
| **MCP** | It acts as the central control for observing changes in the carriage and updating other subsystems accordingly. In this instance, the change in the system is the number of carriages and the number of requests for occupying stations. |
| **UI** | Provides a user interface for controlling the Blade Runner, including route planning and status display.   * routePlanning: int[][] is there for all the route stored in coordinates * sendCommand(command: String): void use for movement of BR |
| **Controller** | Manages communication between the UI and other system components, processing user inputs and updating the state. |
| **Server** | Handles network communication via WiFi, managing requests and responses between the system and external interfaces. |
| **Logger** | Responsible for logging system events and errors, aiding in debugging and maintaining system integrity. |

#### Table 1.4: Class descriptions

## 

## 1.8 Project Constraints

| **Constraint ID** | **Description** |
| --- | --- |
| C01 | The entire project is to be completed within 13 weeks. |
| C02 | There is a hard budget limit of $100 on the entire project. |
| C03 | The hardware in use is mostly determined by the structure and motions teams according to their designs. |

#### Table 1.5: List of constraints

# 2 Scope of Project

## 2.1 Deliverables

1. **Carriage Control Program (CCP)**: Development of software that manages the movement, speed, and operational state of the carriages. This includes functions such as acceleration, deceleration, door control, and responding to LED signals from the track.
2. **Connectivity Module**: Implementation of the communication protocol to connect the CCP with the MCP, enabling real-time data exchange, status updates, and command execution.
3. **Testing and Integration Plan**: A detailed plan outlining the testing strategies for the CCP, including unit tests, integration tests, and system tests with the hardware components.
4. **Project Presentation:** An in-class presentation to showcase the project, its objectives, outcomes, and any challenges faced during the development process.
5. **Documentation:**

* **Scoping Document:** This document describes the project challenge, identifies requirements and constraints, and can be understood alone without reference to other documents. It serves as an agreement between the customer and the engineering team.
* **Design Review:** Ensures that the conceptual design is aligned with the project requirements.
* **Design Document:** Builds upon the scoping document, discussing the conceptual design and its evolution. It refers to the requirements and trade-offs made throughout the design process.
* **Testing Document:** Outlines the approach for testing the project, describing the scope of testing, including what will and will not be tested. It provides strategies for mitigating or managing risks if they occur.
* **Statement of Work:** Demonstrates that the feedback provided by tutors has been considered and addressed.

## 2.2 Inclusions and Exclusions

Inclusions:

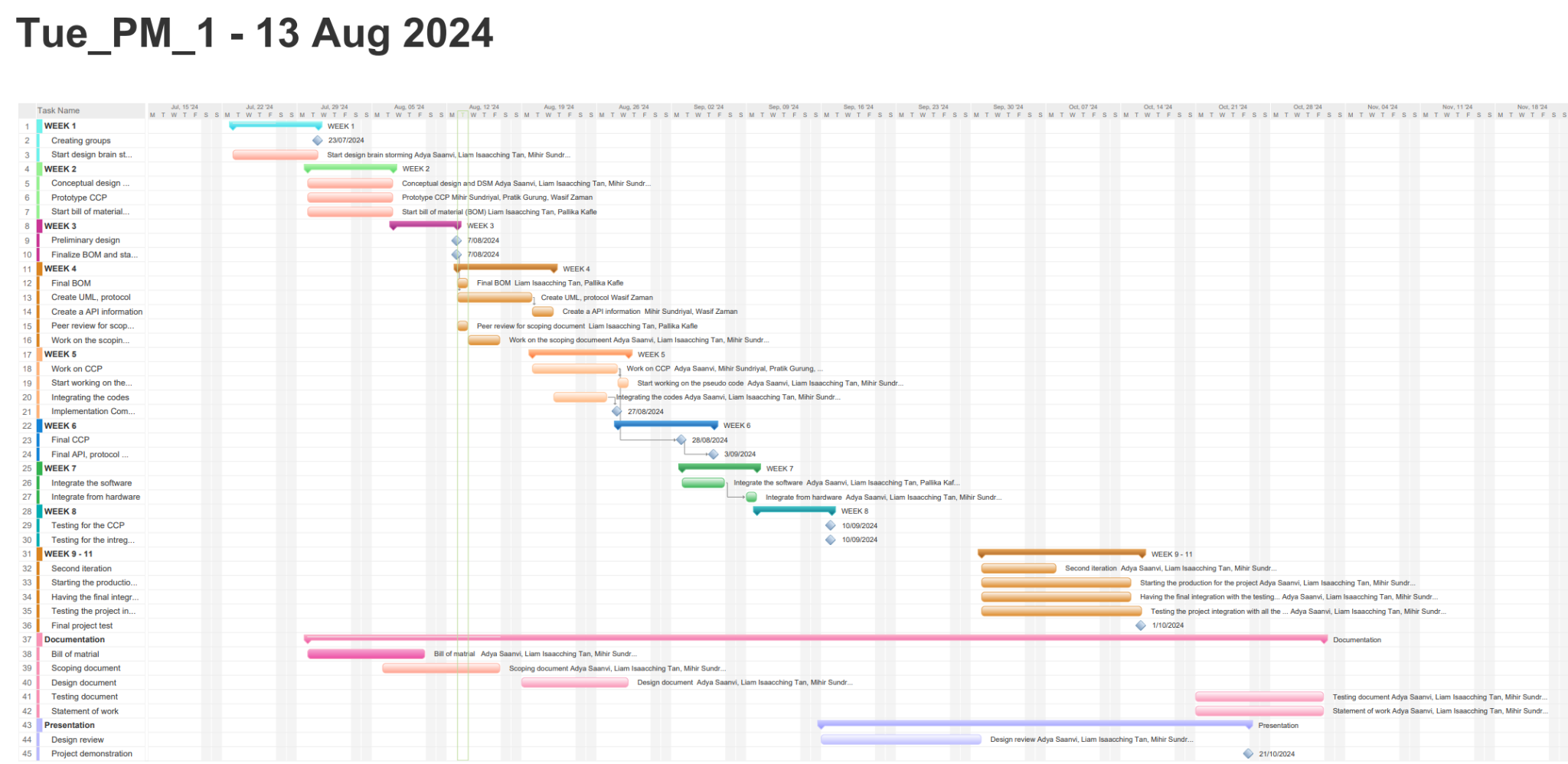
* Development of software for the CCP.
* Implementation of the TCP/IP communication module between CCP and MCP.
* Testing of the software in simulated environments before hardware integration.

Exclusions:

* Deployment of hardware components such as sensors, LEDs, carriages, and rail systems.
* Software development for the Master Control Program (MCP), except for ensuring compatibility with CCP.

## 2.3 Project Schedule

The software development will follow the project timeline with the following milestones:



#### Figure 2.1. Project schedule.

# 3 Requirements

## 3.1 Functional Requirements

| **ID** | **Description** |
| --- | --- |
| FR01 | The carriage constantly updates the MCP with information regarding its location and status via WiFi. |
| FR02 | The MCP must be able to issue route commands to the CCP via WiFi. |
| FR03 | The carriage must be able to quickly accelerate from and decelerate to a full stop. |
| FR04 | LED lights on the carriages signify whether the carriage is stopped, in motion, or decelerating. |
| FR05 | The CCP must be able to operate the carriage’s doors. |
| FR06 | There must be sensors on the carriage that allow the CCP to detect stations and other carriages in its path, and respond accordingly. |

#### Table 3.1. Functional requirements.

| **ID** | **Acceptance Criteria** |
| --- | --- |
| FR01 | At regular intervals, the CCP communicates its location, route, and status. The data is presented as follows:   * Location: Station ID if stopped, “In transit” if moving. * Route: Station ID to Station ID if moving, “Stopped” if stopped. * Status: “Stopped”, “In transit”, or “Decelerating”, depending on its current action. |
| FR02 | Once the MCP assigns a route to a carriage, the carriage must respond promptly and begin moving along the designated route. |
| FR03 | When the appropriate command is called, the carriage must be able to:   * Accelerate: from a complete stop, the carriage accelerates until it reaches a set cruising speed or until a decelerate command is called. * Decelerate: the carriage decelerates until it reaches a complete stop. |
| FR04 | LEDs on the carriage change colours to reflect the carriage’s current status, as follows:   * Red, if the carriage is stopped * Yellow, if the carriage is decelerating, and * Green, if the carriage is in transit. |
| FR05 | Upon stopping at a station, doors on the side of the carriage must open then close shortly after. |
| FR06 | If another carriage is detected ahead, the carriage must decelerate to a stop to avoid collision. If a station is detected, the carriage can either stop if the station is its destination or keep moving otherwise. |

#### Table 3.2. Acceptance criteria for the functional requirements.

## 3.2 Non-functional Requirements

| **ID** | **Description** |
| --- | --- |
| NR01 | The system should be able to handle outages or other failures with minimal interruption. |
| NR02 | All moving parts should operate smoothly. |

#### Table 3.3. Non-functional requirements.

| **ID** | **Acceptance Criteria** |
| --- | --- |
| NR01 | In the event of a partial or total system failure, the system should be able to resume its prior activities as soon as possible. |
| NR02 | The motors on the carriage should behave as follows:   * Track locomotion motors: should accelerate and decelerate gradually, leading to smooth motion without sudden starts or stops. * Door motors: should open and close the doors slowly, without slamming open or closed. |

#### Table 3.4. Acceptance criteria for the non-functional requirements.

## 3.3 Interfacing Requirements

| **ID** | **Description** |
| --- | --- |
| IR01 | The communication protocols between the MCP and CCP must be standardised to ensure clear and consistent data exchange. |
| IR02 | The CCP must integrate seamlessly with the carriage hardware system developed by the motions team. |

#### Table 3.5. Interfacing requirements.

| **ID** | **Acceptance Criteria** |
| --- | --- |
| IR01 | Data passes between the MCP and CCP smoothly. When the CCP updates the MCP with its information, the MCP updates its information on the carriage accordingly. When the MCP issues a route to the CCP, the route is followed. |
| IR02 | When called, CCP commands cause the respective carriage hardware to operate accordingly. This includes the LEDs, door motors, locomotion motors, sensors, and WiFi module. |

#### Table 3.6. Acceptance criteria for the interfacing requirements.

# 4 Technical Performance Measures (TPMs)

## 4.1 Summary of TPMs

The main technical performance measures (TPMs) essential for the dependability and functionality of the system are provided below which have their own individual identifying TPM code and name.

| **TPM Code** | **Summary of Function** |
| --- | --- |
| TPM01 | Carriage Path Request Management |
| TPM02 | MCP Route Management and Execution |
| TPM03 | Real-time Route Processing |
| TPM04 | Emergency Braking System Integration |
| TPM05 | LED status indicator and accuracy |

#### Table 4.1. TPM Summary.

#### 

## 4.2 List of TPMs

The following list of TPMs provided a detailed overview of each TPMs that will be implemented within the system. Each TPMs is linked with a set of functions such as the path for the carriage, route management for carriage, emergency braking system and so on. The table below will overview the main purpose of the TPMs along with their source. It also discusses the method of maintenance of the TPMs depending on their risk level and dependencies.

| **TPM Code** | TPM01 |
| --- | --- |
| **Name of TPM** | Carriage Path Request Management |
| **Purpose** | To ensure that the system efficiently handles multiple path requests from carriages, enabling optimal route planning and management. |
| **Source** | The system shall allow carriages to request different paths for efficient route planning and management. |
| **Risk Level** | High |
| **What should be measured?** | 1. The ability of the system to handle multiple path requests. 2. The accuracy of the provided paths 3. The systems response time to path requests and updates. 4. The system's performance underload |
| **How should it be measured?** | Functionality test: Verify that the system allows at least three different path requests within a session.  Performance test: check the accuracy of the paths, ensuring they stay within a 5% deviation from the optimal route.  Load test: Test the system's ability to handle at least 100 simultaneous path requests without degrading performance. |
| **How often should it be measured** | Regularly during system updates and maintenance |
| **Measure of Success** | The system allows and correctly processes multiple path requests with accurate results within the specified time frames. |
| **Measure of Failure** | The system fails to handle the minimum number of path requests.  Response times exceed the specified limits. |
| **Possible Causes of Failure** | Insufficient processing power of system resources  Bugs or errors in the pathfinding algorithm. |

#### Table 4.2. Carriage Path Request Management

| **TPM Code** | TPM02 |
| --- | --- |
| **Name of TPM** | MCP Route Management and Execution |
| **Purpose** | To ensure that the MCP efficiently manages and executes route instructions for the CCP, ensuring seamless operation and coordination. |
| **Source** | The MCP shall manage and carry out routes for the CCP. |
| **Risk Level** | High |
| **What should be measured?** | The MCP must be able to assign, manage, and execute routes for the CCP's.  The accuracy and timeliness of route execution by the CCP under MCP’s control  The MCP’s handling of multiple routes for different CCP’s. |
| **How should it be measured?** | Performing function tests to validate that the MCP can correctly assign and manage routes  Performance test to measure the time taken by the MCP  Reliability test to evaluate the MCP’s ability to handle multiple CCPs |
| **How often should it be measured** | Can be measured during the initial system integration phase |
| **Measure of Success** | The MCP successfully assigns routes to the CP |
| **Measure of Failure** | If the MCP fails to assign or manage routes effectively |
| **Possible Causes of Failure** | Communication problems between the CCP and MCP  Software bugs in the MCP |

#### Table 4.3. MCP Route Management and Execution

| **TPM Code** | TPM03 |
| --- | --- |
| **Name of TPM** | Real-time route processing |
| **Purpose** | Ensures that the MCP processes route requests and updates within the time constraints. |
| **Source** | System performance requirements specifically focusing on user interaction and route planning |
| **Risk Level** | High |
| **What should be measured?** | Processing time for route requests and updates |
| **How should it be measured?** | By monitoring the time elapsed from the moment a route request or update |
| **How often should it be measured** | Should be monitored continuously depending on how frequently this system is used. |
| **Measure of Success** | The MCP must process 95% of route requests and updates within 2 seconds so that there is consistency even when it is busy. |
| **Measure of Failure** | The MCP fails to process more than 5% of route requests and has updates that come in more than 2 seconds or shows inaccuracies during busy times. |
| **Possible Causes of Failure** | If a high volume of requests occurs, network latency causes delays and software bugs, including code errors. |

#### Table 4.4. Real-time route processing

| **TPM Code** | TPM04 |
| --- | --- |
| **Name of TPM** | Emergency Braking System Integration |
| **Purpose** | To ensure that the CCP braking system is fully integrated and can be overridden in critical situations. |
| **Source** | The CCP emergency braking system must have the ability to be taken over in times of crisis. |
| **Risk Level** | High |
| **What should be measured?** | Whether or not the system is responsive to being overridden by other subsystems |
| **How should it be measured?** | Using simulation tests to test the response time and whether or not it is effective or not. |
| **How often should it be measured** | Should be monitored continuously depending on how frequently this system is used. |
| **Measure of Success** | The system must be responsive within 1 second of initiation of using it. |
| **Measure of Failure** | The system fails to override within 1 second of activation. |
| **Possible Causes of Failure** | * Communication issues between subsystems * Software bugs * Hardware malfunctions within the braking system |

#### Table 4.5. Emergency Braking System Integration

#### 

| **TPM Code** | TPM05 |
| --- | --- |
| **Name of TPM** | LED Status Indicator Accuracy |
| **Purpose** | To ensure that LED indicators on the carriage accurately reflect the actual status of the system. |
| **Source** | Interface LED indicators with motion and structural modules to provide visual feedback. |
| **Risk Level** | Medium |
| **What should be measured?** | The accuracy and visibility of the LED during situations |
| **How should it be measured?** | Simulation tests to see whether the LED indicators change status promptly and are visible |
| **How often should it be measured** | Should be monitored continuously depending on how frequently this system is used. |
| **Measure of Success** | LED indicators accurately and promptly reflect system status changes. |
| **Measure of Failure** | LED indicators show failed accuracy and are not visible. |
| **Possible Causes of Failure** | * Delays in communication from the control system * Hardware issues in the LED |

#### Table 4.6. LED Status Indicator Accuracy

# 

# 5 Design Structure Matrix (DSM)

Design Structure Matrix (DSM) is an effective technique used to visualise the relations between different components in a system and their interaction with one another. The following DSM represents the relation between the different TPMs that were disscussed.

| **TPMS** | **TPM01** | **TPM02** | **TPM03** | **TPM04** | **TPM05** |
| --- | --- | --- | --- | --- | --- |
| **TPM01** | X | X | X |  |  |
| **TPM02** |  | X | X |  |  |
| **TPM03** |  |  | X | X |  |
| **TPM04** |  | X | X | X |  |
| **TPM05** |  |  |  | X | X |

#### Table 5.1. DSM.

# 