# Instructions

This assignment is designed to assess your ability to apply the concepts introduced during Weeks 1 and 2 of lectures and tutorials. It consists of three key components: (1) applying the Engineering Method to systematically approach and address the challenge; (2) leveraging appropriate technologies to support your solution; and (3) integrating an AI agent as part of the final implementation. To complete this assignment successfully, you will need to demonstrate your understanding by synthesising and applying what you have learned so far in this unit to design a practical, well-reasoned, and functional solution.

# Your Challenge PART 1: On the Engineering Method

Using the engineering method, you will:

1. Analyse the problem and define its requirements.
2. Organise and describe all the data and inputs.
3. Design an algorithm to control the system.
4. Implement the solution using plain English.
5. Test and refine your solution with example input values.

# Step-by-Step Instructions

**Background:**

As part of a transportation infrastructure team, you are tasked with designing a logic-based safety system to control the operation of gates at a railway level crossing. This system must ensure that gates are lowered when a train is approaching or a vehicle is still on the tracks and only raised when it is completely safe.

Your design will rely entirely on simple intuitive logic (Boolean logics, digital logic gates, and truth tables will be taught soon, so don’t have to use any of these for this assignment), and will be implemented using an algorithm in plain English.

**You are to complete the following steps:**

## Step 1: Exploring the Problem

1. Restate the problem in your own words.
2. Identify and describe all inputs and outputs of the system.
3. Describe the context, constraints (technical, economic, social, environmental, legal), and stakeholders you can think about.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Our transport infrastructure team have been approached to design a simple yet safe control logic system for the level crossing gates of a railway track, that automatically lowers and raises the railway level crossing gates so that trains can pass without colliding with road vehicles.  **The Client’s Requirements of the System Include:**   * Ensuring that the gates are lowered prior to the arrival of the train upon its approach * Level crossing gates must remain down while any vehicle is on the tracks and the gates can only be raised once it is safe, such that no train is approaching and the tracks are clear.   **System Inputs/Outputs:**   |  |  | | --- | --- | | **Inputs (Sensors/Signals)** | **Outputs** | | * **TrainApproachSensor**   + HIGH when train is approaching. | 1. **Gates**    * Lowered when gates go down.    * Raised when gates go up.    * Held down, gates remain down.    * Held up when gates do not lower because vehicle is present. | | * **VehicleOnTracks**   + HIGH when a vehicle is detected on the crossing. | * **Warning/Emergency Alarm**   + On when standard warning activated   + Sound intensifies when vehicles on present on track   + Off when standard warning deactivated | | * **TrainOnCrossing**   + HIGH when the train is physically on the crossing. | * **Warning/Emergency Lights**   + On when standard warning activated   + Light intensifies when vehicles on present on track   + Off when standard warning deactivated | | * **TrainDepartureSensor**   + HIGH when the train has left the crossing. |  |   **Context, Constraints and Stakeholders:**   * **Technical Requirements:**   The system should be designed for implementation using basic Boolean logic in combination with timers. It must incorporate reliable sensors and be engineered with fail-safe defaults — for instance, barriers should lower automatically in the event of a malfunction or uncertain status. Power redundancy and backup systems must also be taken into account to ensure uninterrupted operation.   * **Economic Considerations:**   Affordability is essential; the use of moderately priced hardware is preferred. Standard, robust sensors and actuators should be prioritised over costly, custom-built alternatives.   * **Social Implications:**   Public safety must be safeguarded at all times. The system should also minimise disruption to road users by preventing unnecessary delays at crossings.   * **Environmental Conditions:**   All components should be weather-resistant, capable of withstanding rain, heat, and cold. Energy efficiency should be incorporated where practicable to reduce power consumption.   * **Legal and Regulatory Compliance:**   The system must comply with Australian rail crossing safety standards and any applicable local or state regulations. This includes maintaining operational records and ensuring that manual overrides are only accessible to authorised personnel.   * **Stakeholder Groups:**   Relevant stakeholders include drivers, rail operators, infrastructure owners, maintenance crews, local councils, emergency services, and the general public. |

## Step 2: Exploring Alternatives

1. Brainstorm at least two different logic-based solutions.

|  |
| --- |
| **Proposed Solution 1: Simple Sensor and Time Based Logic System**  The logic based solution for proposal 1 will include components such as:   * **Train Approach Sensor** * **Occupancy Sensor** * **Train Departure Sensor** * **Gate Warning Alarm (Audible)** * **Gate Warning LED Lights (Visual)** * **Railway Crossing Gates** * **Logic Controller**   We will implement the **Train Approach Sensor**, to be positioned prior to the railway gate crossing on the side where the train will approach, at a safe distance so detection can occur early so there is time for any delayed processes to occur.  The **Train Approach Sensor** will be used to detect if and when a train is approaching the crossing. When the sensor detects an incoming train, it will trigger the safety procedures which include enabling warnings by turning on the alarm sound as well as the LED lights. If the crossing is clear (detected via **occupancy sensor**) the gates will immediately lower, otherwise if a vehicle is present on the track (detected via **occupancy sensor**) the gates will be held up for the vehicle to pass through, which then leads to the gates being lowered. a followed by the gates being lowered.  Directly at the railway crossing will be an **Occupancy Sensor** to detect the presence of vehicles or trains immediately within/crossing the intersection.   * + If a vehicle is detected while a train is approaching, the system delays lowering the gates until the vehicle has safely exited the crossing.   + If a vehicle enters the crossing late (after the gates have started lowering), the system halts or pauses gate movement until the vehicle clears, preventing it from being trapped.   + If a train is detected actively crossing, the gates remain fully lowered until it has completely passed.   Beyond/further past the crossing we will install a **Train Departure Sensor** the crossing. This sensor will detect when the last part/rear of the train has cleared the approaching and crossing areas to confirm that the train has departed. Once the train has completely passed and the **Train Departure Sensor** is no longer getting readings of its presence (indicating its successful passage), a short delay (of 5 – 10 seconds) will be implemented as an additional safety measure, after which the gates will be raised. |
| **Proposed Solution 2: Signal Based Logic System with On Train Communication**  The logic based solution for proposal 2 will include components such as:  **• Crossing Occupancy Sensor**  **• Warning Alarm (Audible)**  **• Warning LED Lights (Visual)**  **• Trackside Signal Base Station (Sender and Receiver)**  **• On Train Equipment (Signal Transmitter and Receiver)**  **• Logic Controller**  **• Railway Crossing Gates**  This solution utilizes a logic based system which utilizes signal based communication between the train and a base station along the railway crosing, along with traditional safety features and sensors to manage the railway crossing.  As the train approaches the crossing, it enters the range of the trackside signal base station. Once in range, the **On Train Signal Transmitter** begins sending approach signals to the base station.  Upon receiving the train’s approach signal, the logic controller calculates the train's estimated time of arrival based on signal strength and frequency. This triggers the initiation of safety procedures.  Safety Procedures Executed:  o Activate warning systems: alarm sound and flashing LED lights.  o Begin lowering the crossing gates.  A **Crossing Occupancy Sensor** is used as a fail-safe. If it detects the presence of the train or another obstruction on the crossing:  The gates will remain lowered.  The warning systems will stay active until the crossing is confirmed clear.  Once the occupancy sensor detects no presence, the approach signal from the train is no longer being received and a train departure signal (sent by the train as it passes the crossing) is received and ends, the system will wait a short delay (to ensure full clearance of train from crossing) before raising the gates and deactivating the warning systems.  Frequency Based Proximity Detection:  •The on-train system can gradually increase the frequency of its transmitted signals as it gets closer to the crossing.  •This increasing frequency acts as a proximity indicator for the logic controller.  For example:  At Frequency X → Train is Distance Y from the crossing. |

1. Research at least one real-world example of a crossing gate control system. Please be brief (100 words).

|  |
| --- |
| A notable real world example of a crossing gate control system would be the ARTC (Australian Rail Track Corporation) Level Crossing Protection System.  ARTC use control logics that utilize train-approach detection via hardware components such as track circuits and axle counters. These components then trigger the flashing lights and siren systems, followed by setting the barriers down with a warning time. Axle counters on the track confirm that the train has completely passed through the intersection prior to setting the barriers to rise up. Australia has a standard which layout and devices follow, referred to as AS 1742.7 (Australian Standard 1742.7).  Many modern railway sites that use train-approach detection also implement obstacle detection (through the use of radar/video analytics) into their systems in order to ensure that barriers are kept down, in the potential case that an issue occurs on the tracks - essentially acting as a fail safe. |

**Steps 3 and 4: Evaluating Alternatives + Engineering Decision**

• Justify your choice based on simplicity, safety, and reliability.

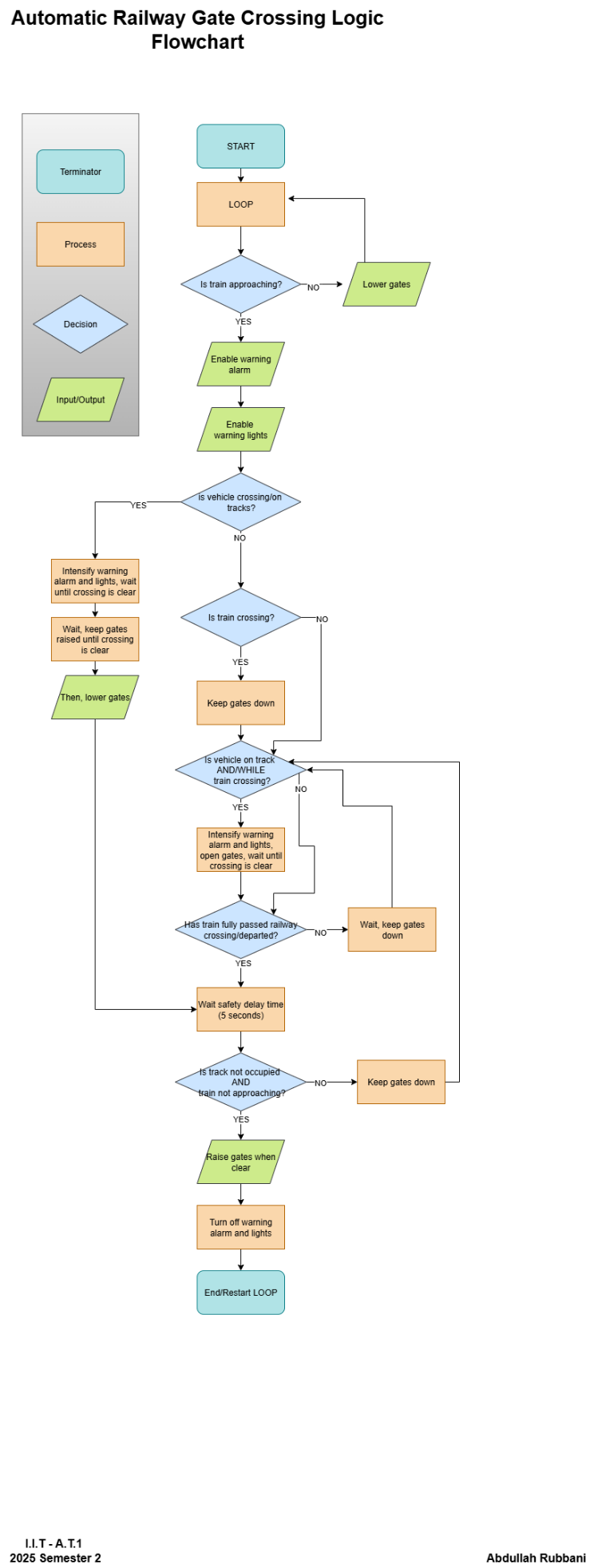
|  |
| --- |
| For the project that our client has approached us with, we will choose to implement the solution of proposal 1. This is because it has a simpler system, while maintaining the specified requirements, which in turn makes it an easier system implement as well as to repair/maintain should anything go awry. Another compelling reason why we chose the system suggested in proposal solution 1 is that the financial cost to acquire all the materials/components and develop the system will be significantly lower than proposal 2’s system, making it the more economically sustainable and suitable option. Furthermore, there are less variables in this system, making it the more reliable option as diagnosing issues will be much more efficient, and developing the system will be easier with having less variables to consider during the development and operation of the railway crossing gate logic system. |

## Step 5 and 6: Planning and Implementing

* Write a sequence of tasks written using plain English.

|  |
| --- |
| **Pseudocode:**  START  LOOP:  // Step 1: Detect Train Approaching  IF TrainApproachSensor = TRUE  THEN  ActivateWarningAlarms()  ActivateWarningLights()  // Step 2: Check for vehicles  IF VehicleOnTracks = TRUE  THEN  HoldGatesUp()  Repeat check until VehicleOnTracks = FALSE  LowerGates()  ELSE  LowerGates()  // Step 3: During Train Crossing  IF TrainOnCrossing = TRUE THEN  KeepGatesDown()  // Extra safety: if VehicleOnTracks = TRUE while train is crossing  // (vehicle stuck inside), keep alarms active to warn vehicle  IF VehicleOnTracks = TRUE THEN  IntensifyWarnings()  // Step 4: After Train Departure  IF TrainDepartureSensor = TRUE  THEN  Wait SafetyDelay(5 seconds)  // Final check before raising gates  IF (VehicleOnTracks = FALSE) AND (TrainApproachSensor = FALSE)  THEN  RaiseGates()  DeactivateWarnings()  ELSE  KeepGatesDown()  END/RESTART LOOP |

* Include a flowchart of the logic.



## Step 7: Testing and Refinement

* Propose at least four test cases based on input combinations.
* Record expected vs. actual outputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Test Cases/ Input Combination** | **Expected Output** | **Actual Output** | **Comments/ Refinements** |
| 1 | * TrainApproachSensor = HIGH * VehicleOnTracks = LOW * TrainOnCrossing = LOW * TrainDepartureSensor = LOW | * Gates = LOWERED, * Alarms = ON * Lights = ON | * Gates = LOWERED * Alarms = ON * Lights ON | Gates lower immediately |
| 2 | * TrainApproachSensor = HIGH * VehicleOnTracks = HIGH * TrainOnCrossing = HIGH * TrainDepartureSensor = HIGH | * Gates = DOWN * Alarms = ON/Intensified | * Gates = DOWN * Alarms = ON/Intensified | Worst-case scenario - System keeps gates down, alarms intensified. |
| 3 | * TrainApproachSensor = HIGH * VehicleOnTracks = HIGH * TrainOnCrossing = HIGH * TrainDepartureSensor = LOW | * Gates = DOWN, * Alarms = ON/Intensified | * Gates = DOWN * Alarms = ON/Intensified | Vehicle stuck while train is crossing, so safety measure works. |
| 4 | * TrainApproachSensor = LOW * VehicleOnTracks = LOW * TrainOnCrossing = LOW * TrainDepartureSensor = HIGH | * Gates = RAISED * Alarms = OFF | * Gates = RAISED * Alarms = OFF | Train departed and track is clear, leading to gates to raise after delay. |
| 5 | * TrainApproachSensor = HIGH * VehicleOnTracks = LOW * TrainOnCrossing = HIGH * TrainDepartureSensor = LOW | * Gates = DOWN * Alarms = ON | * Gates = DOWN * Alarms = ON | Gates already lowered for crossing and alarms active. |

* Suggest improvements or refinements to the logic circuit.

|  |
| --- |
| * The provided background information has an issue regarding ambiguity in the scenario when a vehicle is already on/getting on the tracks while a train is approaching. The ambiguous condition from the background information suggested that when either condition is true (“gates are lowered when a is train approaching OR a vehicle is still on the tracks”), which is not a good logic condition as it could result in a severe accident through a collision between the train and a trapped vehicle. We have already gone ahead and refined this issue by setting up the logic, such that if a vehicle is present/occupying the track, the gates must not immediately lower, otherwise the vehicle may get trapped. Instead, the gates will raise for the vehicle to exit out of the crossing and will then close once the vehicle is no longer present. * Implement additional short delays within the circuit logic, specifically before lowering gates when train is approaching in order to prevent vehicles unintentionally getting trapped. * Implement log system to record all train/vehicle/gate events for safety audits and analysis. * Integrate an emergency override feature which provides a manual control option for authorities if any sensors break/fail or gates get stuck for digital controls. * Add backup sensors to prevent failure in detecting vehicles or trains and to mitigate issues if a sensor suddenly stops working. |

**Your Challenge PART 2: On the Use of Technology.**

To promote professional reporting and personal development practices, you are now required to use [**GitHub**](https://github.com/) to manage your project files and collaborate efficiently when needed.

## Step 1: Set Up Your Repository

* Create a public or private repository on [GitHub.](https://github.com/)
* Add your tutor and your lecturer as collaborators under *Settings > Collaborators*.
* Use a clear and descriptive repository name, e.g., `pet-feeder-project`.

**Step 2: Organize Your Repository:**

* Create folders for each step, e.g., `/Step1\_Analysis`, `/Step3\_Flowchart`, `/Step4\_Word\_Code`, etc.
* Include a `**README.md**` file describing your project. - Use meaningful comments to document changes.

## Step 3: Document Your Work: -

* Push your flowchart (e.g., exported PNG/PDF) to the repository.
* Include your word-based code under a clearly named file.
* Upload test results and any additional documentation.

**Submission Requirement:**

* Include the GitHub repository link in your report.

[Abdullah-R-786/I.C.E-AT1-Railway\_Gate\_Project: Add description here](https://github.com/Abdullah-R-786/I.C.E-AT1-Railway_Gate_Project/tree/main)

* Ensure the repository is accessible to your tutor and unit convener by the due date.

# Your Challenge PART 3: On AI Agent Integration

In this part of the assignment, you will explore how Artificial Intelligence (AI) can assist in solving problems, refining logic, and enhancing your assignment’s documentation. **You are encouraged to use an AI agent** such as Microsoft Copilot to support your work.

Use Copilot to assist with at **least two** or more of the following:

1. **Refine your logic or Word Code**: Ask Copilot to review your implementation step and suggest improvements or identify potential issues.
2. **Generate alternative solutions**: Prompt Copilot to propose different ways to solve the problem or enhance your flowchart logic.
3. **Explore real-world implementation**: Use Copilot to discuss how your system could be built using actual hardware (e.g., Arduino, Raspberry Pi).
4. **Improve documentation**: Ask Copilot to help you write a professional README.md file or summarize your project for presentation.
5. **Reflect on ethics and limitations**: Use Copilot to explore the ethical implications of using AI in automated pet care or discuss the reliability of AI-generated suggestions.

*Deliverable: Write a short reflection (150–250 words) that includes:*

* *Prompts and responses (what you asked and what it responded with).*
* *What insights or improvements it helped you achieve?*
* *How it influenced your final solution or understanding of the problem?*
* *You may include screenshots or excerpts of your interaction with Copilot if relevant.*

|  |
| --- |
| **AI Agent Integration**  The integration of an AI agent was particularly useful in evaluating the logic, exploring implementation pathways, and enhancing the documentation of the railway crossing system. Rather than replacing the design process, the AI acted as a supportive tool to refine and validate the decisions already embedded in the pseudocode.   1. **(3.) Explore Real World Implementation**   The AI helped me consider how my pseudocode could be applied using real hardware.  For example:   * TrainApproachSensor / TrainDepartureSensor could be built with infrared or ultrasonic sensors. * VehicleOnTracks could be detected with pressure pads or motion sensors. * Gates could be controlled with servo motors or stepper motors. * Alarms/Lights could be implemented with LEDs, buzzers, and flashing signals.   This correlation between pseudocode and hardware demonstrated and provided insight to me about how the logic could be deployed in practice through real world hardware and application.   1. **(2.) Generate Alternative Solutions**   The AI also explored alternative approaches to the handling of vehicle detection. One suggestion was the use of a holding sequence, where the gates remain up until the track is clear before lowering. This matched the “HoldGatesUp” process already implemented in the pseudocode. The consistency between the proposed alternatives and the current logic confirmed that the chosen solution was both valid and effective.  The use of an AI agent contributed to the refinement of logic, validation of existing mechanisms, exploration of implementation options, and improvement of documentation. Importantly, the refinements were consistent with features already incorporated into the pseudocode, ensuring that the logic remained unchanged while the clarity and professional quality of the assignment were strengthened. |

# Submission Requirements and Checklist

## Checklist Part 1

☐Clearly labelled sections for each engineering method step.

☐Research of 100 words (Step 2).

☐Flowchart created with Draw.io (Steps 5 & 6).

☐Written Sequence of Tasks (Steps 5 & 6).

☐Sample test cases and discussion (Step 7).

## Checklist Part 2

☐GitHub repository link is included.

☐Student’s GitHub is well structured.

**Checklist Part 3**

☐Reflection of 150-250 words.

## Submission instructions

* **This is a SOLO ASSIGNMENT**.
* Allocated marks are shown in the rubric on page 6 of this assignment.
* Create a folder called “uxxxxx\_Assignment1” and drop all your files in there (problemsolving process answers, flowcharts, Word documents, and the like).
* Compress (zip) ALL your files and folders created above in one single file called uxxxxx\_Assignment1.zip. Upload this file on Canvas by the due date using the drop box provided in the corresponding assignment.

# Appendix

## How to Make a Flowchart using Draw.io

1. **Open Draw.io:**

Go to [Open Draw](https://www.drawio.com/) and click **Start** > **Device** (or **Google Drive** if you want to save it online).

1. **Create a New Diagram** o Choose **"Blank Diagram"**

o Name it something like PetFeeder\_Flowchart

## 3. Use Flowchart Symbols

o Drag and drop standard shapes from the **General** or **Flowchart** section:

* **Oval** = Start / End
* **Rectangle** = Process (e.g., "Dispense Food")
* **Diamond** = Decision (e.g., "Is it 8:00 AM?")
* **Arrow** = Connect the shapes logically

1. **Build the Logic** o Start with “System ON”
   * Include key decisions like:
     + “Is it feeding time?”
     + “Is there food in the container?”
     + “Has the pet eaten?” o Add actions: “Dispense Food”, “Wait 10 minutes”, “Send Alert”, etc. o Use arrows to guide the logical flow
2. **Add Labels and Colours**  o Use colours or text formatting to highlight decision points or alerts
   * Keep the layout clean and readable
3. **Save and Export:**
   * Go to **File > Export As > PDF** or **PNG** o Include the exported image in your assignment submission

# Assignment 1 Grading Rubric

**Criteria Excellent (HD) Good (CR–D) Satisfactory (P) Unsatisfactory (N) Marks**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Step 1: Exploring the Problem** | Problem clearly restated; all  inputs/outputs well  identified; multiple  constraints and stakeholders discussed. | Problem mostly restated; most  inputs/outputs and constraints identified. | | | Basic problem restatement; limited discussion of  constraints and stakeholders. | Incomplete or unclear restatement; little to no identification of system components. | 20 | |
| **Step 2: Exploring Alternatives** | Two or more ideas clearly explained;  well-researched realworld example within word limit. | Two ideas explained; example provided with minor detail issues. | | | Only one idea or weak explanation; real-  world example lacks clarity. | No alternatives or research included. | 15 | |
| **Steps 3 & 4: Evaluation and Decision** | Clear justification with strong reasoning in  terms of simplicity, safety, and reliability. | Justification provided with moderate  reasoning across some criteria. | | | Basic reasoning; only one or two evaluation aspects discussed. | No or poor justification of design choice. | 20 | |
| **Steps 5 & 6: Planning and Implementation** | Detailed, logical sequence of tasks and accurate, complete flowchart. | Mostly clear sequence and understandable flowchart. | | | Some task sequence shown; flowchart lacks clarity. | Sequence missing or illogical; no flowchart provided. | 20 | |
| **Step 7: Testing and Refinement** | Four+ test cases; all expected vs. actual  results compared; meaningful  refinements proposed. | Three test cases with some comparison and basic improvement ideas. | | | Two test cases; minor suggestions or  incomplete output comparison. | Fewer than two test cases; no refinements offered. | 15 | |
| **Presentation and Communication** | Excellent structure, spelling, diagrams, and clarity throughout. | | Mostly clear and wellorganized with minor issues. | Understandable but lacks polish or organization. | | Disorganized, unclear or poorly presented work. | | 10 |
| **Part 2: Use of GitHub** | Repository is complete, organized with folders,  README.md, and detailed commit  history; tutor and lecturer added. | | Repository mostly complete with minor structure issues;  README.md present. | Repository present but poorly structured or missing documentation. | | Repository missing or inaccessible. | | 10 |
| **Part 3: Use of AI Agent (Copilot)** | Reflection includes 2+ examples with  prompts/responses and clear insights;  demonstrates critical thinking. | | Reflection includes 2 examples and shows moderate insight. | Basic reflection with limited engagement or 1 example. | | No AI reflection or irrelevant content. | | 10 |

**TOTAL 120**