ANALYSIS

Group 3

APRIL 5, 2023

PROJECT

T3

The problem

The problem we want to solve with this project is the congestion at certain points on a train station during the peak hours. Due to these congestions in certain points at the station it will be hard to walk through and to get off and get on. This means the stationary time will be increased.

The goal

The goal of the project is to decrease the congestion of people at a train stations platform. This will be done by indicating where the closest empty waggon is at the platform. It will also indicate when a wagon is full.

The wagon will measure the occupancy rate of the coupe by measuring the amount of people that walk into the coupe and by the number of seats taken. In each wagon we will have a display that shows the occupancy rate of each level if the waggon has multiple levels.

Use Cases

In this section use cases and corresponding requirements are described.

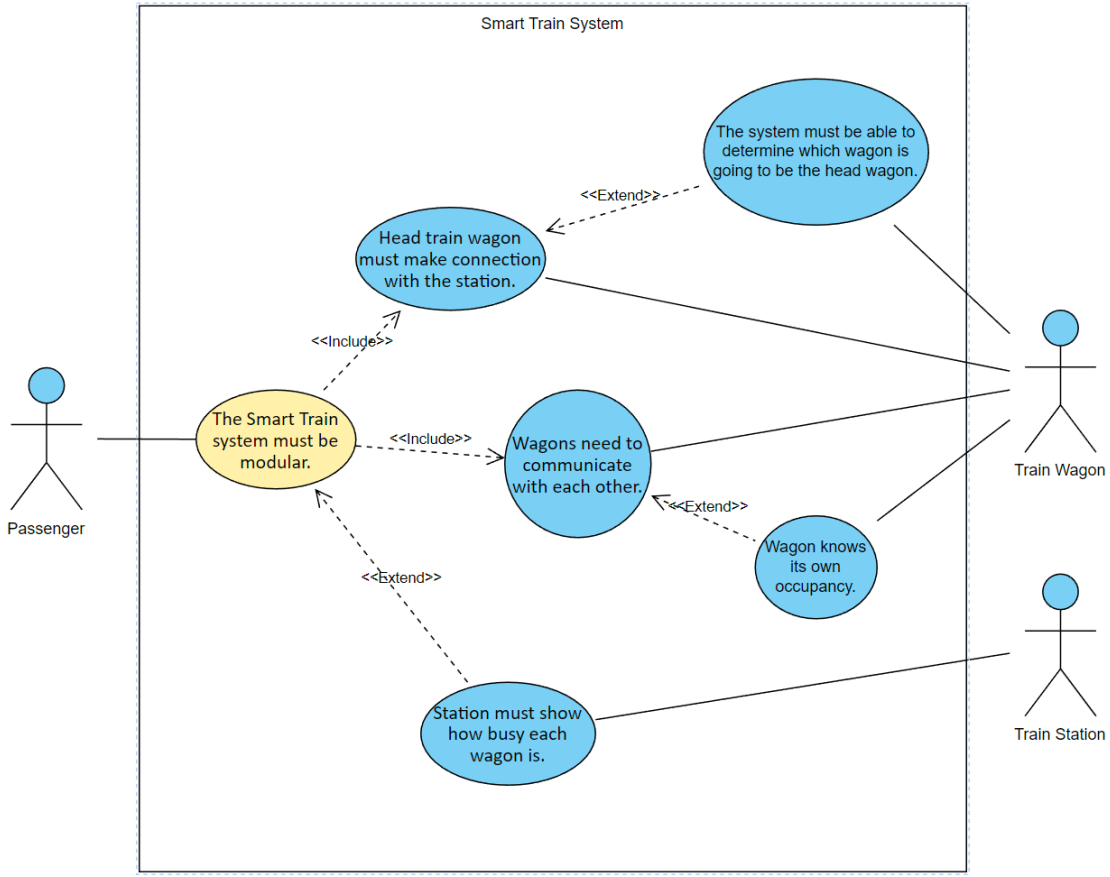

Figure 1. Use case diagram of Smart Train System

Use Case ID: UC_001	The Smart Train system must be modular.	
Scenario:	Station needs to know wagon details to direct passengers.	
Actors:	Passenger	
Brief description:	The passengers know where to stand which is done by the station that knows where the wagon is and how long it is, and this information is sent by the wagon to the station.	
Pre-conditions:	1. MQTT connection must exist between station and head wagon. 2. Head wagon must know details of all other wagons.	
Post-conditions:	1. Each wagon details sent to station. 2. Station lights up platform LEDs based on info.	
Flow of activities:	**Actor**	**System**
	1. For each wagon 1.1. Send wagon position.	

	1.2. Send wagon length. 1.3. Send wagon occupancy.	1.1.1 Light up LEDs at the position of the wagon. 1.2.1 Light up LEDs for the full length of wagon. 1.3.1 Set LEDs color based on occupancy.
Exception conditions:	1. Head wagon not connected, undefined info sent from head wagon.	

Use Case ID: UC_002	Head train wagon must make connection with the station.	
Scenario:	The head wagon should connect with the station and communicate with it.	
Actors:	Train wagon	
Brief description:	The head train wagon connects with the station through MQTT.	
Pre-conditions:	1. Head train wagon must be selected. 2. Station must be available for connection.	
Post-conditions:	3. Output after connection attempt is connected.	
Flow of activities:	**Actor**	**System**
	1. For head wagon 1.1. Connection attempt with station	1.1.1 Station must return connected.
Exception conditions:	1. Head wagon not selected, station doesn't exist, connection error	

Use Case ID: UC_003	Station must show how busy each wagon is.	
Scenario:	The station should turn on certain amount of LEDs at a specific color depending on the busyness of the wagon and its length.	
Actors:	Train station	
Brief description:	The station shows each wagon's occupancy. This is done by automatically turning the LEDs on the platform in front of the wagon in question a certain colour depending on its occupancy. The number of LEDs that need to be turned on will depend on the length of the wagon/3.	
Pre-conditions:	1. MQTT connection between head wagon and station. 2. Head wagon gathered necessary info of all wagons. 3. Wagon length known by station. 4. Wagon occupancy known by station. 5. Wagon position known by station.	
Post-conditions:	1. LEDs for the same length of the wagon turn on. 2. LEDs turn green if occupancy below 50% 3. LEDs turn orange if occupancy 50-99% 4. LEDs turn red if occupancy 100% 5. 1 LED turns on for every 3 metres	
Flow of activities:	**Actor**	**System**
	1. For station	

	1.1. Check wagon position. 1.2. Check wagon length 1.3. Check wagon occupancy.	 1.1.2. Selects total no of LEDs needed to turn on based on wagon length/3. An LED every 3 metres will be turned on. 1.1.3. Sets the LEDs in question to a certain colour.
Exception conditions:	1. LED doesn't work.	

Use Case ID: UC_004	Wagon knows its own occupancy.	
Scenario:	Train wagon keeps track of number of people sitting inside the wagon.	
Actors:	Train wagon	
Brief description:	The train wagon knows the busyness inside it by checking how many seats are taken out of the total number of seats.	
Pre-conditions:	1. Button actuator is placed to simulate a seat that when pressed means a passenger has sat in the seat.	
Post-conditions:	1. Wagon output's percentage of total seats taken.	
Flow of activities:	**Actor**	**System**
	1. For each seat 1.1. Check button pressed. 1.2. Button pressed	 1.1.2. Total seats taken count increased by 1 and divided by total number of seats to get percentage of occupation.
Exception conditions:	1. Button doesn't work, wagon doesn't work, calculation error	

Use Case ID: UC_005	Wagons need to communicate with each other.
Scenario:	Each wagon needs to be able to communicate with each other through a wired connection to understand its own position relative to the others and to know the other wagons' positions. Each wagon also need to send their occupation data to the head wagon.
Actors:	Train wagon
Brief description:	The wagon communicates with the other wagons and finds out their positions and its own position in the train. It also sends the occupancy data to the head wagon.
Pre-conditions:	1. CAN bus connection between wagons. 2. Head wagon exists/selected. 3. Wagon occupancy known.
Post-conditions:	1. Wagon position known. 2. Other wagons positions known.

	3. Occupancy data sent to head wagon.	
Flow of activities:	**Actor**	**System**
	1. For each wagon	
	1.1. Get wagon position.	1.1.1 Ask head wagon for position.
	1.2. Get other wagons positions.	1.2.1 Communicate with other wagons to know their positions.
	1.3. Send occupancy data	1.3.1 Send data to head wagon.
Exception conditions:	1. Wagon doesn't work, CAN bus connection error, position unknown, head wagon doesn't exist.	

Use Case ID: UC_006	The system must be able to determine which wagon is going to be the head wagon.	
Scenario:	The wagon with the machinist needs to be the head wagon at the start or the wagon with the lowest ID is the head wagon. These IDs should be given to each wagon at the start. The machinist wagon should have the lowest at the beginning. Each wagon should also have its own manufactured ID.	
Actors:	Train wagon	
Brief description:	The wagon with the machinist will be the head wagon at the beginning. In cases where the head wagon is detached or power is lost, the wagons will communicate and be given each an ID in random. The wagon with the lowest ID will be the head wagon.	
Pre-conditions:	1. All wagons connected with each other. 2. CAN Bus connection between wagons.	
Post-conditions:	1. Sets ID to each wagon. 2. Selects head wagon.	
Flow of activities:	**Actor**	**System**
	1. For each wagon 1.1. Machinist exists. 1.2. Machinist doesn't exist. 1.3. System restarted.	 1.1.1 Wagon becomes head. 1.2.1 Wagon given an ID higher than head wagon. 1.3.1 Head wagon selection protocol.
Exception conditions:	1. CAN bus connection error, wagon doesn't exist, power failure.	

Scenarios

Scenario 1: plug and play wagons

Border case:

For this scenario, we want to show the plug and play capabilities of our system. This will be done by disconnecting a wagon and reconnecting it. The system should keep running when a wagon gets disconnected and include a new wagon when connected.

Unhappy flow 1: System disconnects middle wagon.

Unhappy flow 2: Sensors not working correctly.

Happy flow 1: system detects new wagon.

Happy flow 2: System disconnects last wagon.

Scenario 2: train entering station

Border case:

For this scenario, we want to show what happens when a train enters the station. The station should show the people which wagon has free space, and which is full. The train should also send if a wagon needs to be coupled or uncoupled.

Unhappy flow 1: does not stop at correct location/ platform.

Unhappy flow 2: shows wrong wagon is empty

Happy flow 1: Train sends that it needs extra wagon when it does

Happy flow 2: Station shows occupation of the wagons

Scenario 3: People leaving at the station

Border case:

For this scenario, we want to show how the wagon reacts to people leaving.

Unhappy flow 1: Sensor fails

Unhappy flow 2: Counting goes below 0

Happy flow 1: People leaving the train and the LED reacts

Happy flow 2: Count goes down by 1.

Scenario 4: first wagon empty all other full

Border case:

For this scenario, we want to show how the platform shows the people what wagon has the least amount of occupancy. And show how it will indicate how many wagons the least empty wagon is to the current wagon.

Unhappy flow 1: Shows wrong number of wagons the least empty wagon is to the current wagon.

Unhappy flow 2: Display hardware fails.

Happy flow 1: Shows correct number of wagons the least empty wagon is to the current wagon.

Happy flow 2: Shows arrow pointing in correct direction.

Scenario 5: People entering at the station

Border case:

For this scenario, we want to show how a wagon reacts to people entering it. So, it should change the LED colors when the current occupancy is at a certain point.

Unhappy flow 1: People keep entering after the occupancy is at max

Unhappy flow 2: Sensor cannot keep up.

Happy flow 1: People enter the train, and the LED reacts

Happy flow 2: People enter the till the max and LED turns red and display indicates next empty

MOSCOW

MUST

The Smart Train system must be modular.

- We want to know how long the wagon is in meters?
- Station/platform needs to know this.
- So that the passengers know where to stand.

The passengers need to know where to stand which is done by the station that knows where the wagon is and how long it is, and this information is sent by the wagon to the station.

Requirements:

- LEDs on platform to show where to stand on platform – Passengers
- Wagon length and position on platform sent to station from wagon – Developer

Head train wagon must make connection with the station.

Requirements:

- Delegate responsibility between wagons on which one is the head wagon – Developer
- Head wagon communicates with the station – Developer

Station must show how busy each wagon is.

Requirements:

- LEDs on platform change colour based on how busy the wagon is.
- LEDs turn green when wagon is below 50% occupancy
- LEDs turn orange when wagon is above 50% and below 100%
- When a wagon reaches 100% occupancy, led lights on the platform in front of this wagon will turn red.
- The station must know the length of each wagon
- The number of LEDs shown for each wagon are proportional to its length. One LED will light up for every 3 meters of wagon length.

Wagon knows its own occupancy.

Requirements:

- Wagon must keep track of busyness in each wagon by amount of sitting people.

Wagons need to communicate with each other.

Requirements:

- Head wagon needs to know length of each wagon
- All wagons need to know positions of each wagon
- Wagon knows current position in train

The system must be able to determine which wagon is going to be the head wagon.

Requirements:

- The wagon with the machinist is the default head wagon when the system starts since it is the least likely to fail.
- When the system starts, the lowest ID is given to the head wagon.
- The head wagon must inform the other wagons, every x ms, that it is functioning.
- When head wagon detaches or is not available, a new head wagon, with the next lowest ID is chosen.

Should

Front/ back of train has master/ slave mode.

Indicator points to wagons adjacent to current wagon fullness.

Wagon knows whether a person or a bag is placed on the seat.

Could

Show amount of 2/4 seats available.

Station has arrow to direction closest empty wagon.

Display to show empty seats wagon (up/ down)