Warehouse Management System

Marcin Śledź | s28026

1. User requirements

The goal of the project is to create a system that efficiently manages warehouse operations, focusing on deliveries, employee roles, and item tracking. The system should support multiple warehouses and handle delivery logistics with constraints like distance limits and driver availability. It should also allow tracking of stored items using barcodes and room identifiers (QR/NFC), and include features like shift and break management, complaint submission, and salary validation.

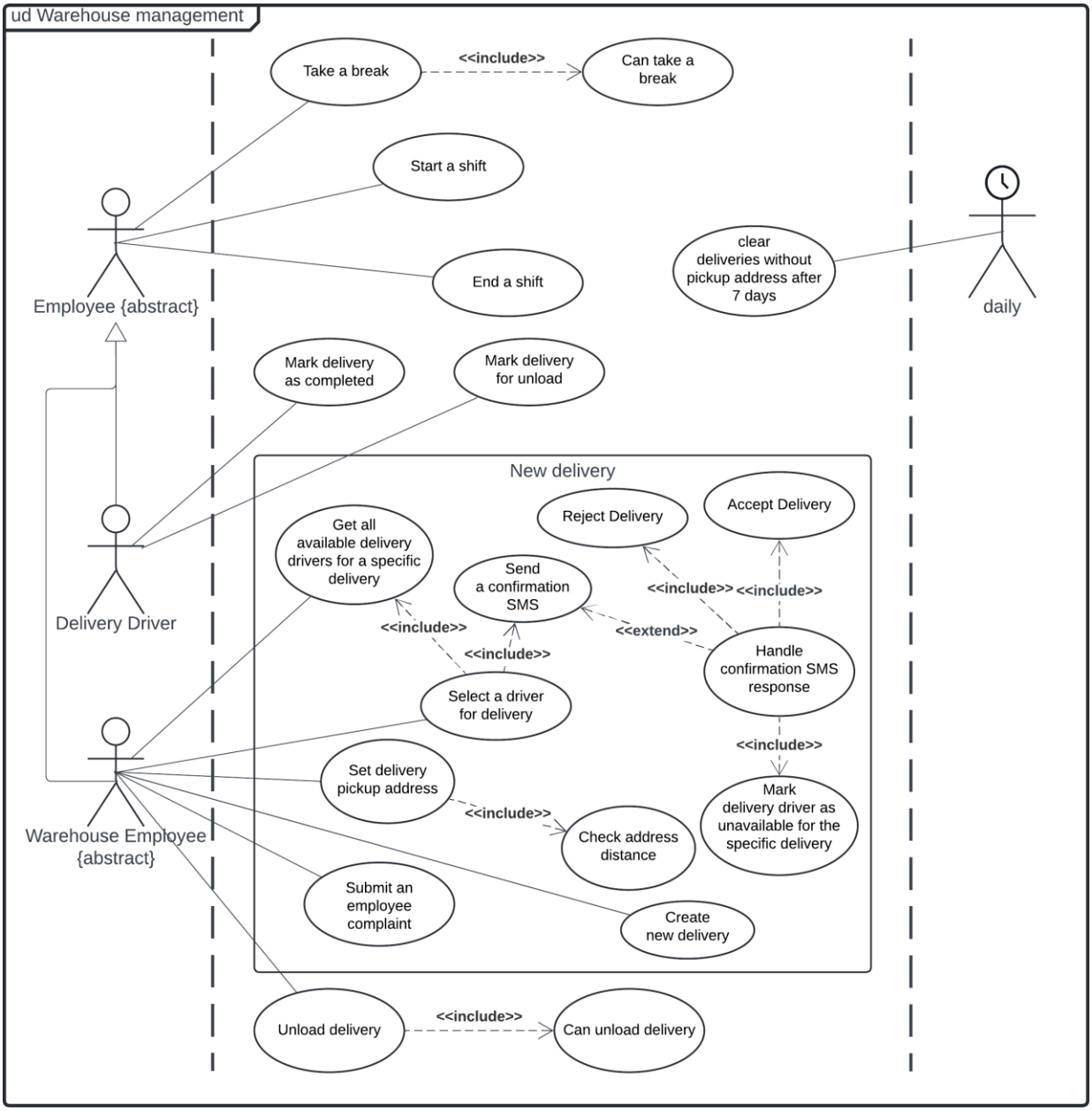
Actors:

- **Warehouse**
  - One per location
  - Location must contain street, street number, city, and zip code.
  - A warehouse is composed of storage rooms, to know the content of each room an NFC tag or a QR code is placed on the door.
- **Person**
  - A person is identified by their pesel which is 11 digits long.
  - Each pesel just be assigned to only one person.
  - They should have information like their first and last name as well as their birth date.

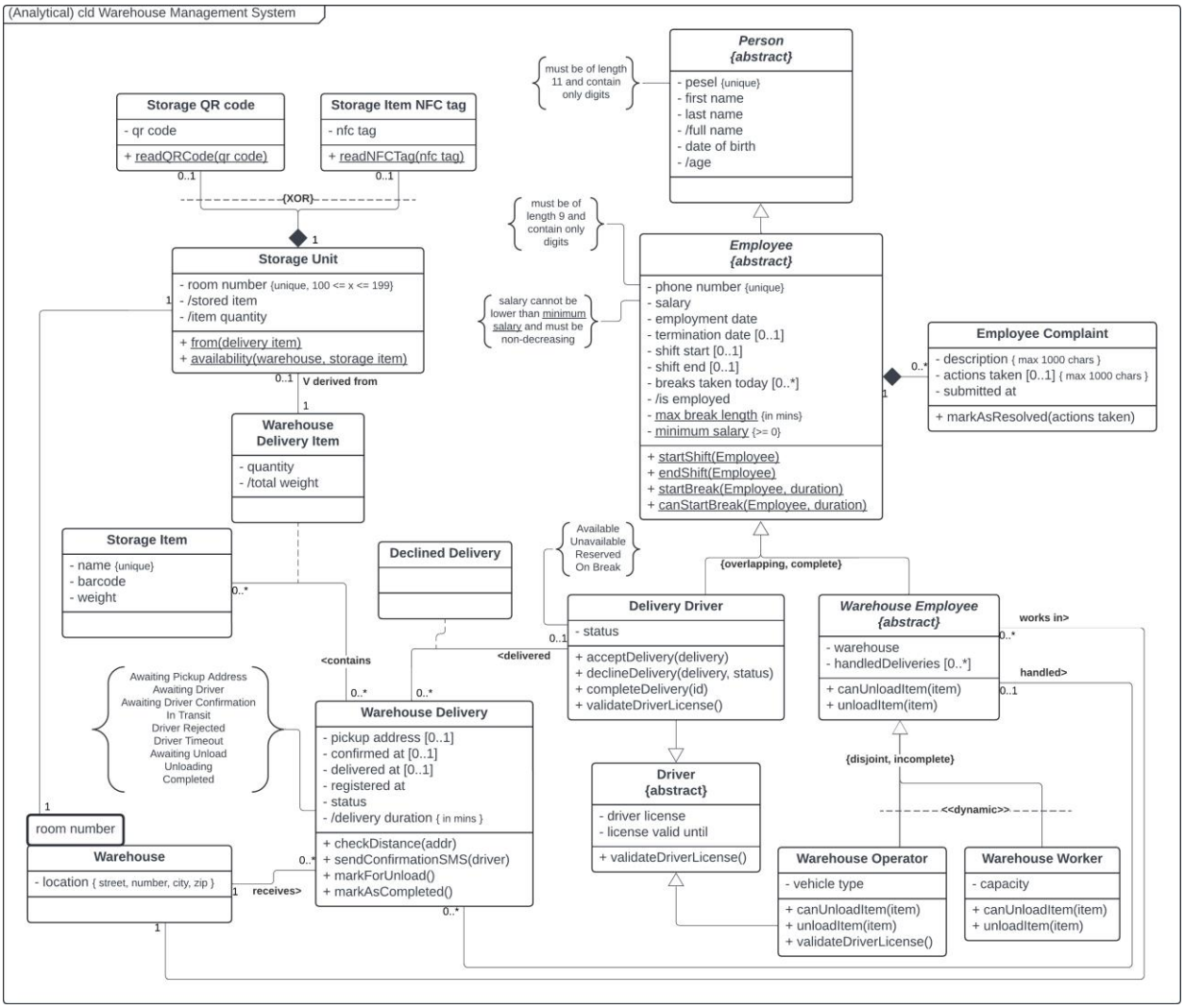
- **Employee**
  - Must store their phone number which can only be assigned to one employee
  - When they started working as well as the termination day
  - A person can work either as a delivery driver or as a warehouse worker.
  - Driver can also work as the warehouse employee
  - There should be information about their shift and breaks taken for the day
  - Should have salary information that which can't be lower than universal minimum salary
  - A person can't have their salary decreased
  - There should be a limit for employee breaks taken for the day
  - Create an anonymous employee complaint system
- **Delivery driver**
  - Must have a valid driver's license
  - Can deliver to multiple warehouses
  - Can also be employed at the warehouse
  - Can decline delivery
  - Communication with the driver will be done by SMS
  - Can only make deliveries closer than 500km
- **Delivery**
  - Will contain multiple items known by the warehouse.
  - Each item has a quantity in which it is delivered.
  - Each item has a weight
- **Warehouse employee**
  - He is the one who handles the deliveries and unloads them into the correct locations
  - Can also be a delivery driver
  - Warehouse employee should have at least two distinct roles:
    - Operators who can travel by vehicle but require a valid driver's license. Can carry any type of delivery.
    - Normal employees travel on foot, but the load they can move on is limited.
  - Has a PC near the unload station where he can control the deliveries
- **Storage items**
  - Warehouse has specific items that it can store
  - Item names must be unique
  - Each item must be identified by a barcode
- **Storage Unit**

- Store quantities of the items inside
- Can contain only one item at a time.
- It is identified by a room number.

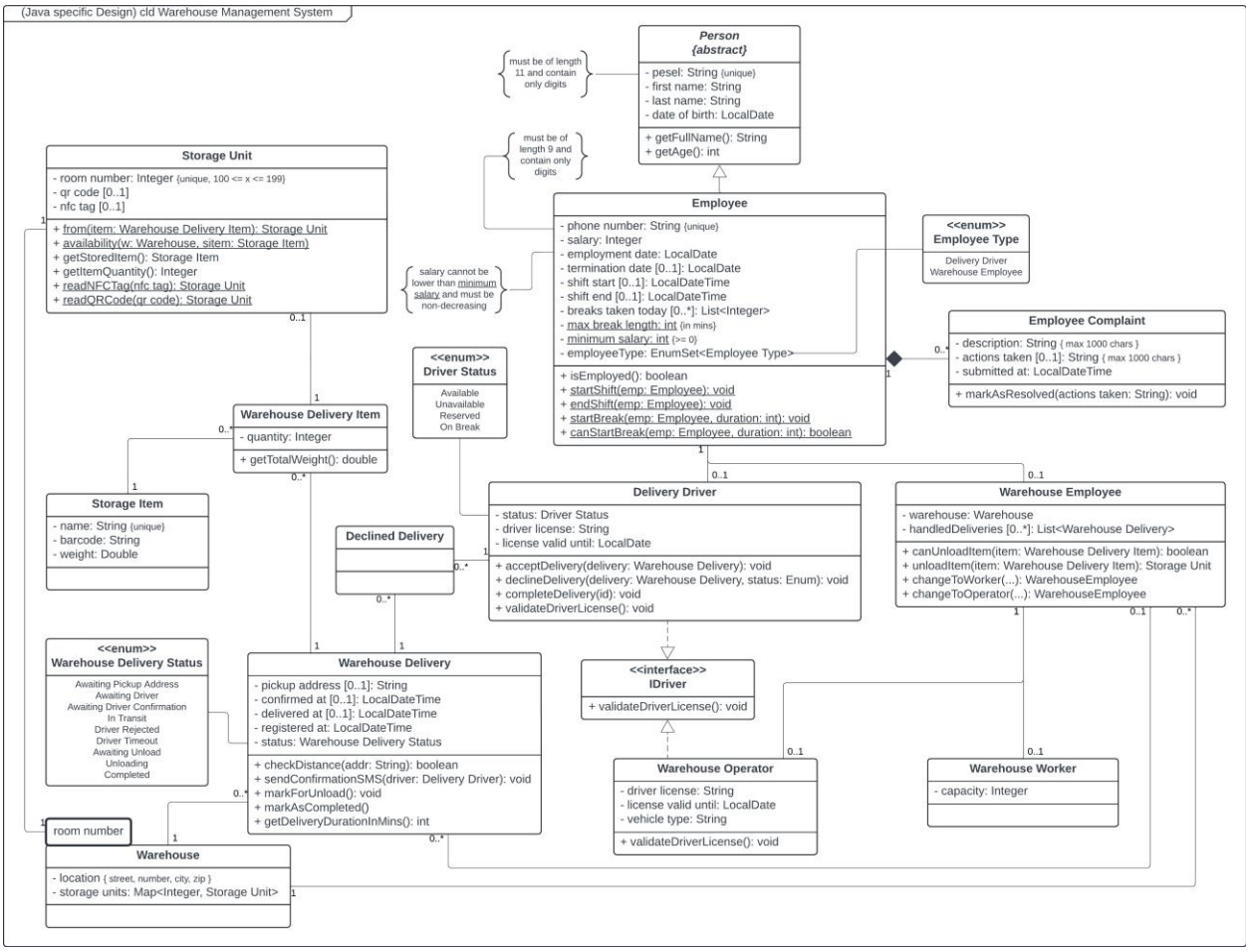
2. Use case diagram



3. Class diagram – analytical



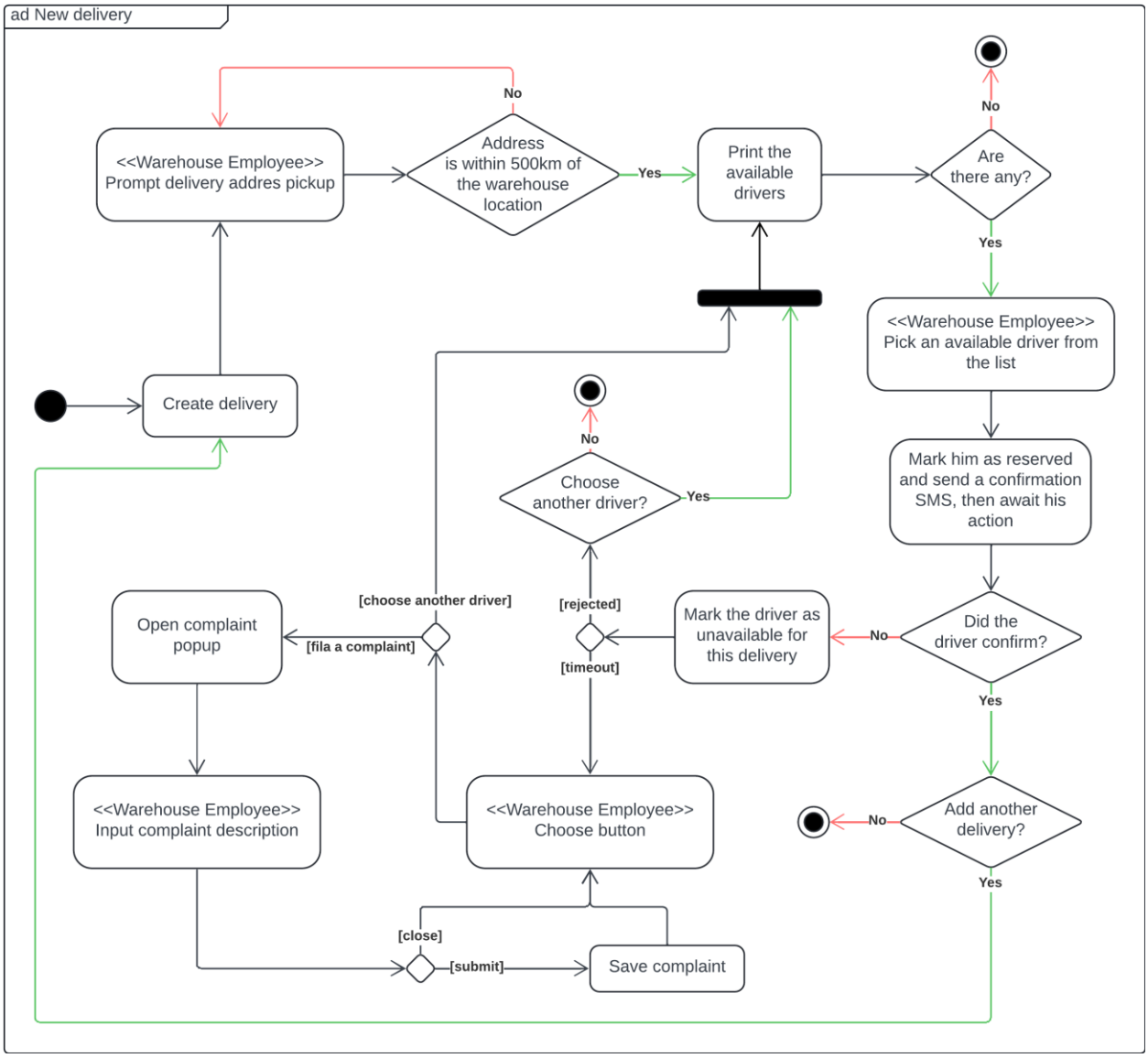
4. Class diagram – design



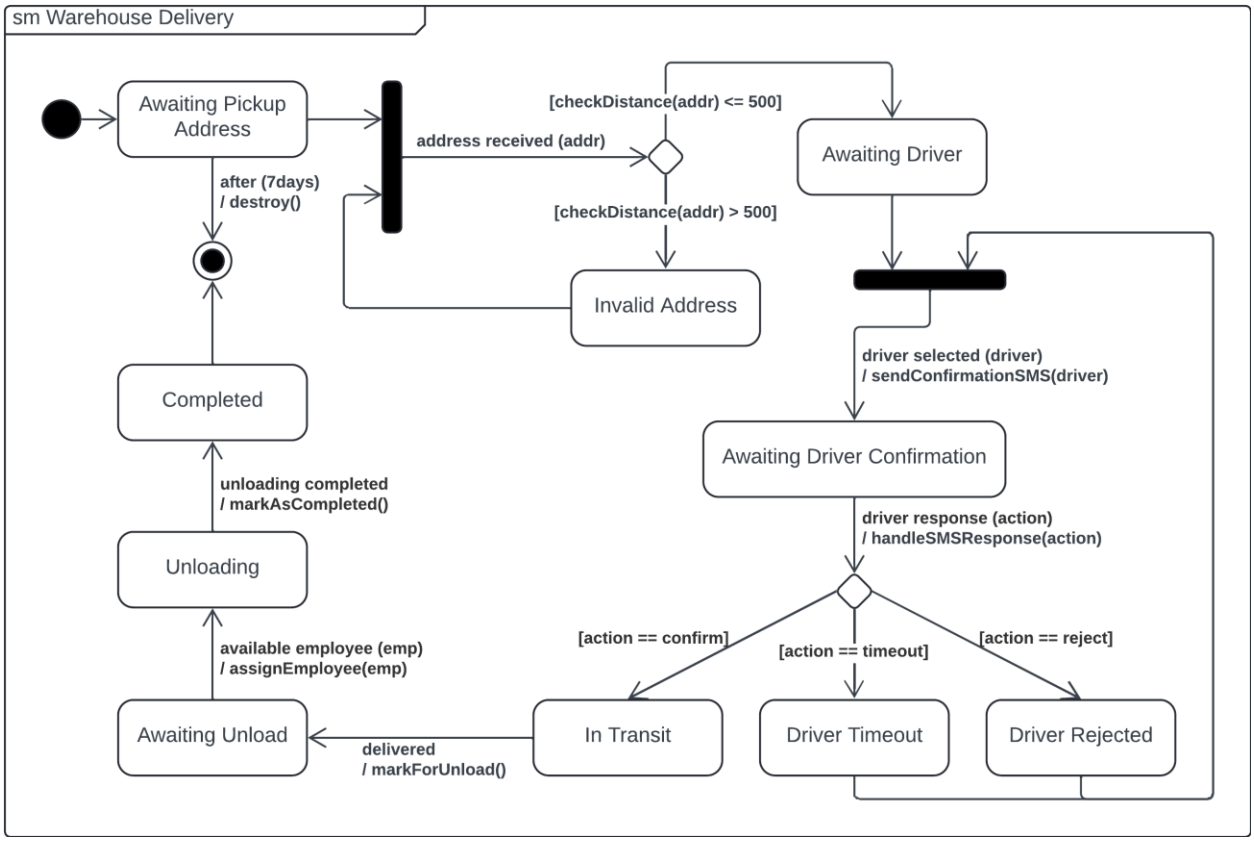
5. Use case scenario

Name	Adding a new delivery
Priority	High
Type	Detailed
Actors	Warehouse Employee. Delivery Driver
Description	Add a new delivery and assign a delivery driver to deliver the delivery. The delivered items can later be stored into the warehouse.
Pre-conditions	Warehouse must exist and have at least one warehouse employee to handle the delivery
Main flow	1. Employee navigates to the dashboard and selects "Add New Delivery". 2. New delivery is created 3. System prompts the employee to input pickup address. 4. System checks if the delivery address is within 500 km of the warehouse. 5. The system prints a list of available drivers. 6. Employee selects a driver from the list. 7. System marks the driver as reserved and sends a confirmation SMS. 8. The delivery is confirmed and the driver is assigned. 9. System prompts if the user wants to add another delivery. If yes then go back to step 2, otherwise yield.
Alternative flow	4A. The delivery address is not within 500 km of the warehouse. 1. System shows an error message. 2. Employee is prompted for new address (step 3). 5A. No drivers are available for this specific delivery. 1. System shows that no drivers are available. 2. Employee is redirected back to the dashboard. 8A. Driver doesn't confirm the delivery. 1. System marks the driver as unavailable for this delivery A. Driver failes to accept in time [timeout]. 1. Prompt to file a complaint or choose another driver A. Employee chooses to file a complaint. 1. System opens a popup. 2. System prompts for complaint description. 3. Employee submits or closes the popup. Go to step 7AA1. B. Employee wants to choose another driver. 1. Go to step 5. B. Driver rejected [reject]. 1. System asks if the user wants to choose another driver A. Go to step 5. B. yield
Post-conditions	A new delivery is now registered into the system and the driver is marked as reserved until the delivery is made. User is returned to the dashboard page.

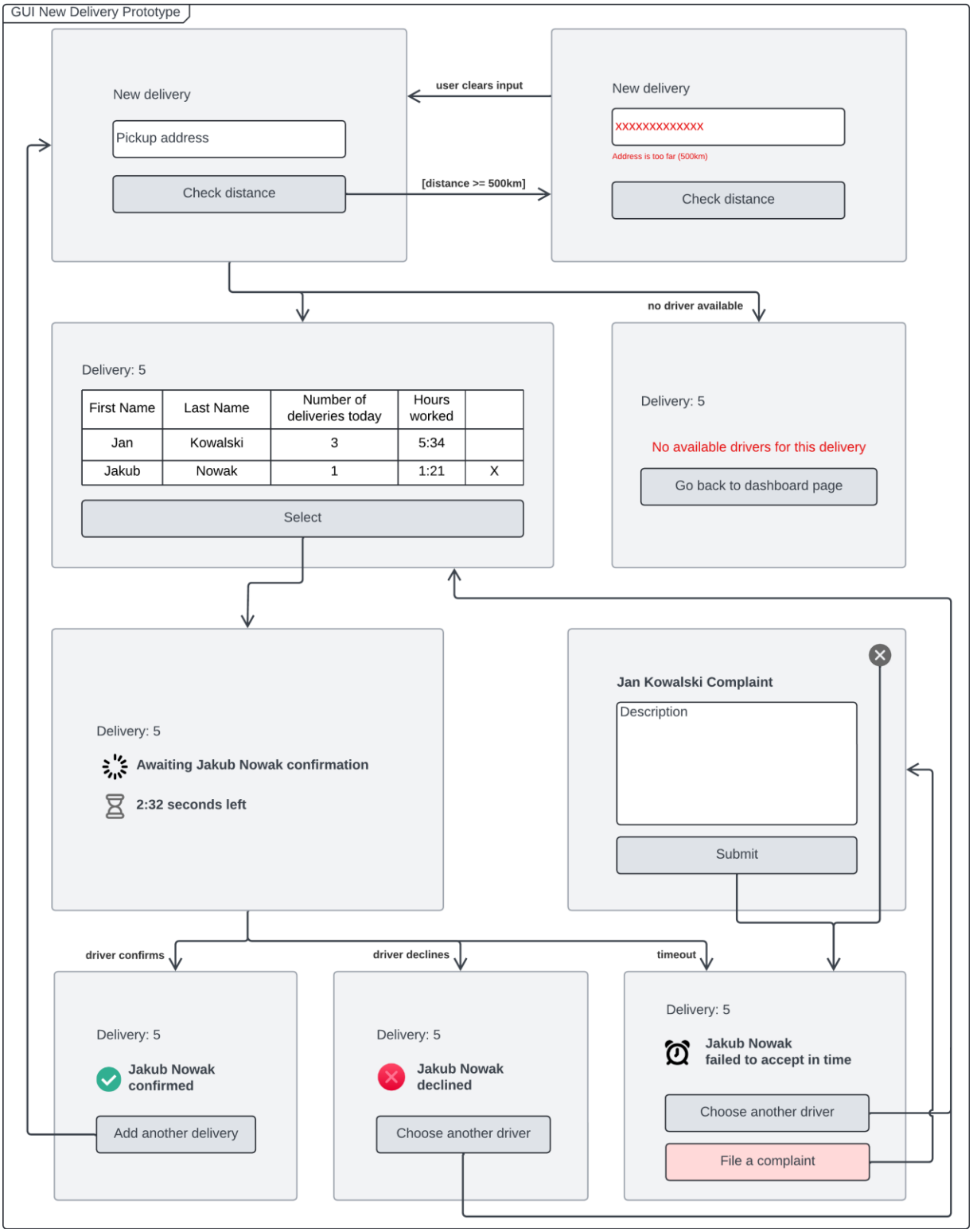
6. New delivery activity diagram



7. State machine diagram (Warehouse Delivery)



8. GUI Prototype for adding a new delivery



9. Design Decisions

Design decisions taken due to the backend being implemented in `Java`.

- **Association with attribute** classes were introduced.
    - Declined Delivery
    - Warehouse Delivery Item
- Due to Java limitation of only one **extend** per class, IDriver interface was introduced.
- Created **enums:**
    - DriverStatus
    - WarehouseDeliveryStatus
    - EmployeeType
- **Derived attributes** were converted into methods.
- Storage QR Code and Storage Item NFC Tag **{XOR}** associations were flattened into the Storage Unit class. To differentiate, check which attribute is null.
- New attribute with type of Map<Integer, Storage Unit> to allow **qualified association** between Warehouse and Storage Unit.
- **Overlapping inheritance** between Warehouse Employee, Delivery Driver and Employee, were converted into associations. Employee and Warehouse Employee are no longer abstract. Employee type is identifiable by the employee type set, which must have at least one value.
- **Dynamic inheritance** between Warehouse Operator and Worker was also converted into associations and new methods were introduced to allow dynamic change: ChangeToWorker(...) and ChangeToOperator(...). The current state can be known by checking which association is null.