Maciej Grabowski - s20504

Cleaning Service Management System

Software Requirements and Implementation Specification

Table of Contents

User Requirements

Recently, a large number of companies offering cleaning services started looking into transforming their business systems. The goal is to modernize the whole management process of requesting as well as fulfilling requested services, making it more client-friendly and easier for employees to follow. In order to make such transformation possible, the companies are demanding a new management system for their businesses.

The new management system should be accessible to all clients, employees and managers, providing different functionalities based on the role of the person using it. It should be capable of allowing users to monitor and possibly modify certain details of ongoing subprocesses and entities crucial to a cleaning-service company structure.

For clients, the new management system should provide an easy way of browsing available cleaning offers and purchasing them, as well as checking the status of previously submitted orders. The system should also provide clients with company's details such as contact information in case of any issues.

All employees, with the use of the new system, have to be able to view the confirmed, ongoing jobs (orders) that they were previously assigned to by a manager. An ability to view contact details of all managers should also be provided in case it is needed. Additionally, hired drivers should be able to view a list of all available vehicles and switch between them (f.e. in case of lack of space for all cleaners / equipment necessary to fulfill an order).

The new management system should also serve as a key software tool for managers. They should have the ability to hire new employees, view the list of currently hired employees, their details and fire employees. Managers must be able to view incoming as well as already confirmed orders (and their details, details of clients). For each incoming (new) order a manager has to either assign employees to it and confirm it or reject the order. After an order has been confirmed it cannot be rejected unless some major issues arise, such special cases should be dealt with individually according to some basis established between a customer and a manager. When viewing confirmed, not yet executed orders a manager should be provided with an option of modifying assigned Cleaners (removing / adding) and swapping Drivers. The new management system must also enable managers to keep track of vehicle warehouses belonging to the given company, together with vehicles currently assigned to them. Lastly, managers should be able to create new cleaning offers based on what services a given company is capable of providing at a certain time.

Use-Case Diagram

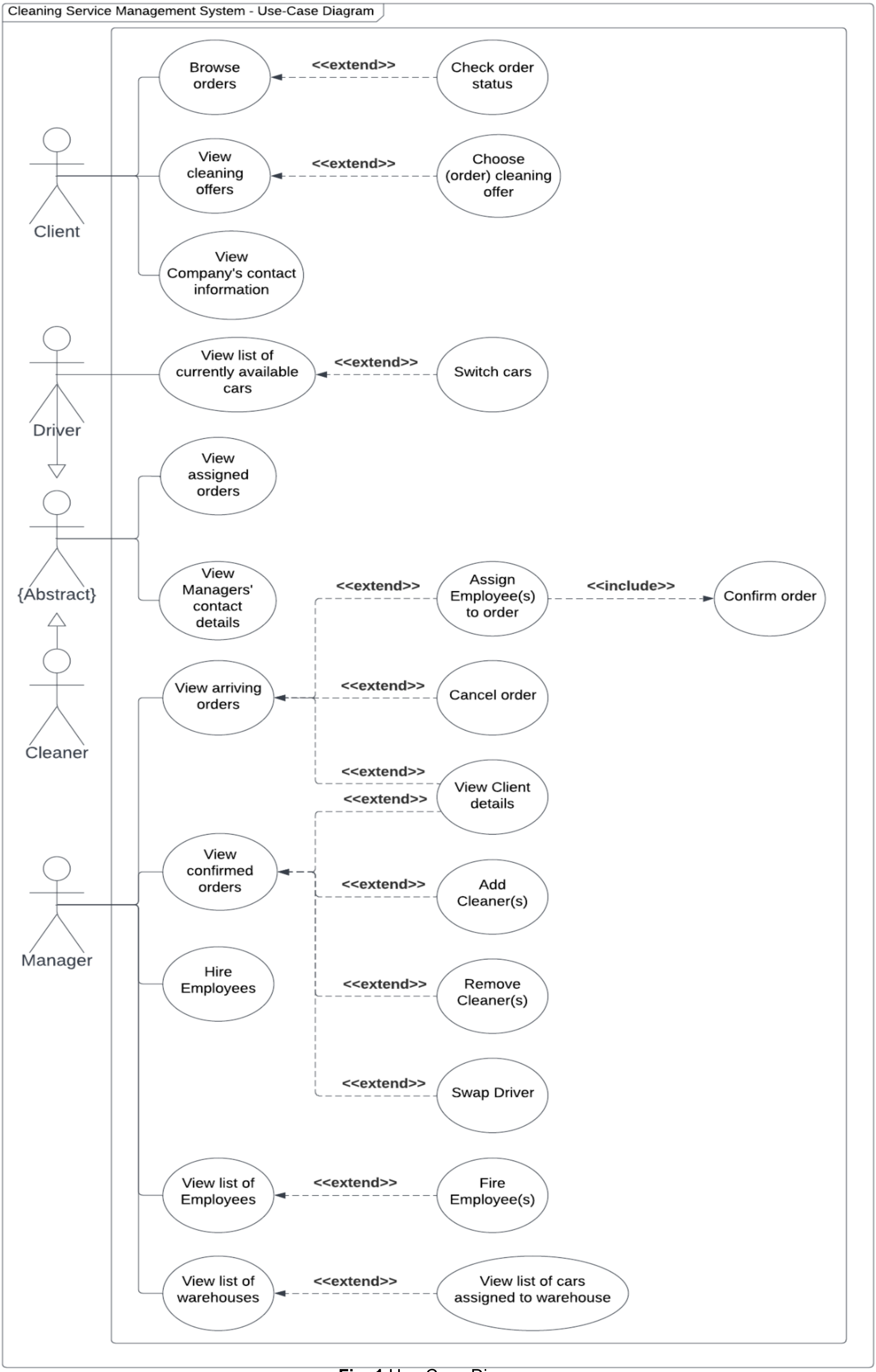


Fig. 1 Use-Case Diagram

Analytical Class Diagram

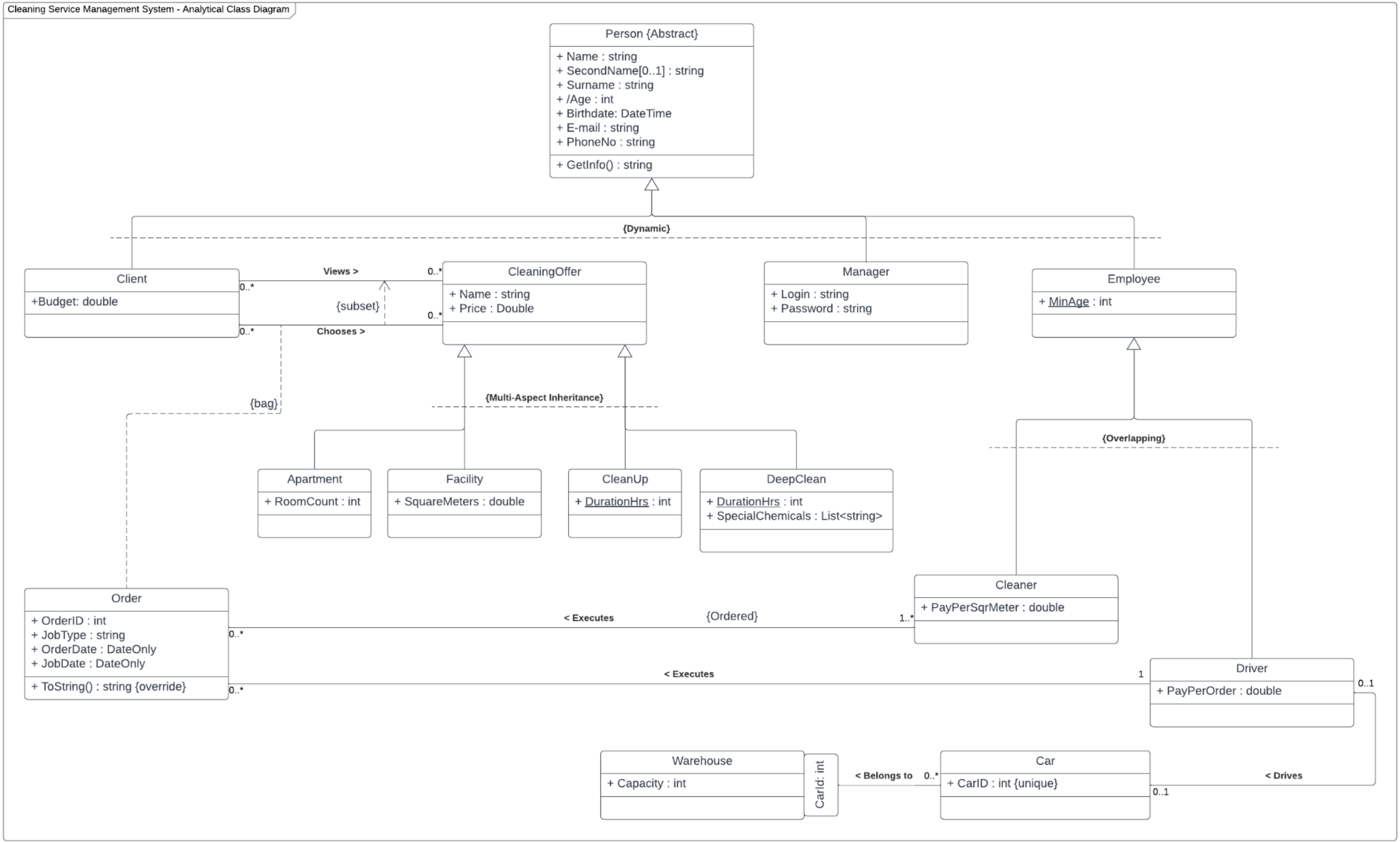


Fig. 2 Analytical Class Diagram

Design Decisions

Since many UML related constructs, such as some of those presented in the *Analytical Class Diagram* (**Fig. 2**), are not possible to create 'as is' using programming languages such as *C#*, some changes had to be made to the diagram to make the *Cleaning Service Management System* possible to implement. These changes are as follows:

Dynamic Inheritance

Dynamic inheritance has been substituted with *Composition*. As a result of that, the *Person Class* has been transformed from an Abstract class to a non-abstract one and provided with methods for changing roles. All classes inheriting from the *Person Class* have been provided with public, static *Create* methods meant to be used together with private constructors. Implementation should follow the rule: a person may change roles (Employee, Client, Manager) frequently, although they may not have more than one role at the same time.

Overlapping Inheritance

Overlapping inheritance has been substituted with *Composition*. As a result of that, all classes inheriting from the *Employee Class* have been provided with public, static *Create* methods meant to be used together with private constructors. Implementation should follow the rule: an Employee has to have at least one role (Cleaner, Driver) and may have more than one role at the same time. The *Employee Class* should be provided with a constructor allowing for assignment of one or more roles during the creation of a new object.

Multi-Aspect Inheritance

Multi-Aspect Inheritance has been substituted with *Composition*. As a result of that, all classes that the *CleaningOffer Class* inherits from have been provided with public, static *Create* methods meant to be used together with private constructors. Implementation should follow the rule: a cleaning offer should have one aspect from each inheritance branch (Apartment / Facility, CleanUp / DeepClean). The *CleaningOffer Class* should be provided with a constructor allowing for such aspect assignment.

Subset Constraint

The *Subset Constraint* should be implemented as an additional functionality of the *Order Class* constructor, allowing a given Client to create an order only if the chosen cleaning offer has been previously viewed by them.

Ordered Constraint

The *Ordered Constraint* should be implemented in the form of an *Ordered Set* attribute in the *Cleaner Class* containing objects of type *Order* related to the association with the *Order Class*.

Qualified Association

The *Qualified Association* between classes *Warehouse* and *Car* should be implemented using a *Dictionary* attribute in the *Warehouse Class* containing tuples of type <*Integer, Car*>.

Additional Changes

All classes, lacking an identifier attribute, have been provided with an additional identifier attribute *[ClassName]ID* meant to be used (as *Primary-Keys*) with the *EFC (Entity Framework Core)* technology. This technology will also serve as an *ORM (Object-Relational Mapper)* for this system, keeping track of all created objects.

Design Class Diagram

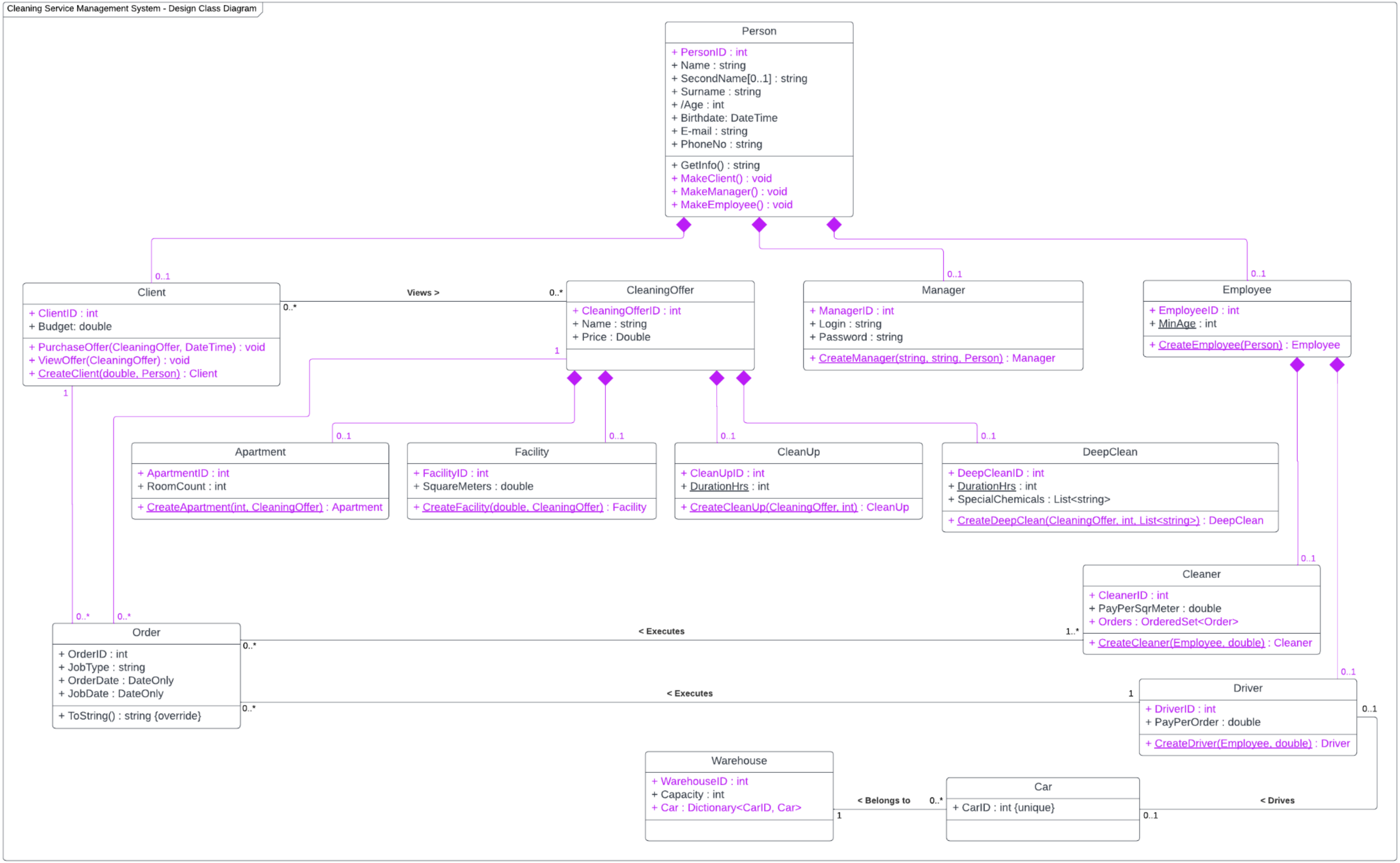


Fig. 3 Design Class Diagram

“Add Cleaner(s)” - Use-Case Scenario

Actors

1. Manager

Purpose & Context

A Manager wants to assign Cleaner(s) to a previously confirmed order. Such a situation may occur when a manager notices that the current number of Cleaners assigned to a given order may not be sufficient enough to fulfill the cleaning-request efficiently, or the Client has explicitly demanded an additional number of Cleaners to participate in the job.

Included Use-Cases

None

Extended Use-Cases

None

Assumptions & Pre-Conditions

1. The Manager has logged in to the system successfully.
2. The Manager pressed the “Orders” button querying the confirmed orders list at the “Manager-MainScreen”.

Initiating Business Events

1. A Manager decides to assign more Cleaners to a previously confirmed order.

Basic Flow of Events

1. The System displays a list of confirmed orders with shortened details.
2. The Manager presses the “Details” button next to a chosen order.
3. The System displays detailed information about the chosen order.
4. The Manager presses the “Modify” button next to the Cleaners count.
5. The System displays a pop-up modification window with lists of names of Cleaners currently assigned to the order and Cleaners that are available.
6. The Manager presses the “+” button next to the chosen available Cleaner(s).
7. The System displays an updated pop-up modification window with names of Cleaners currently assigned to the order, and Cleaners that are available.
8. The Manager presses the “Save” button.
9. The System displays detailed information about the order with updated Cleaner count information.

Alternative Flow of Events

No Available Cleaners

4. The Manager presses the “Modify” button next to the Cleaners count.
    a. The System displays a pop-up modification window with lists of names of Cleaners currently assigned to the order and Cleaners that are available, however, the list of available Cleaners is empty.

Extension Points

None

Post-Conditions

1. Cleaner(s) chosen by the Manager has/have a new order assigned to them.
2. Cleaner(s) chosen by the Manager has/have been assigned to a chosen order.

“Add Cleaner(s)” - Dynamic Analysis

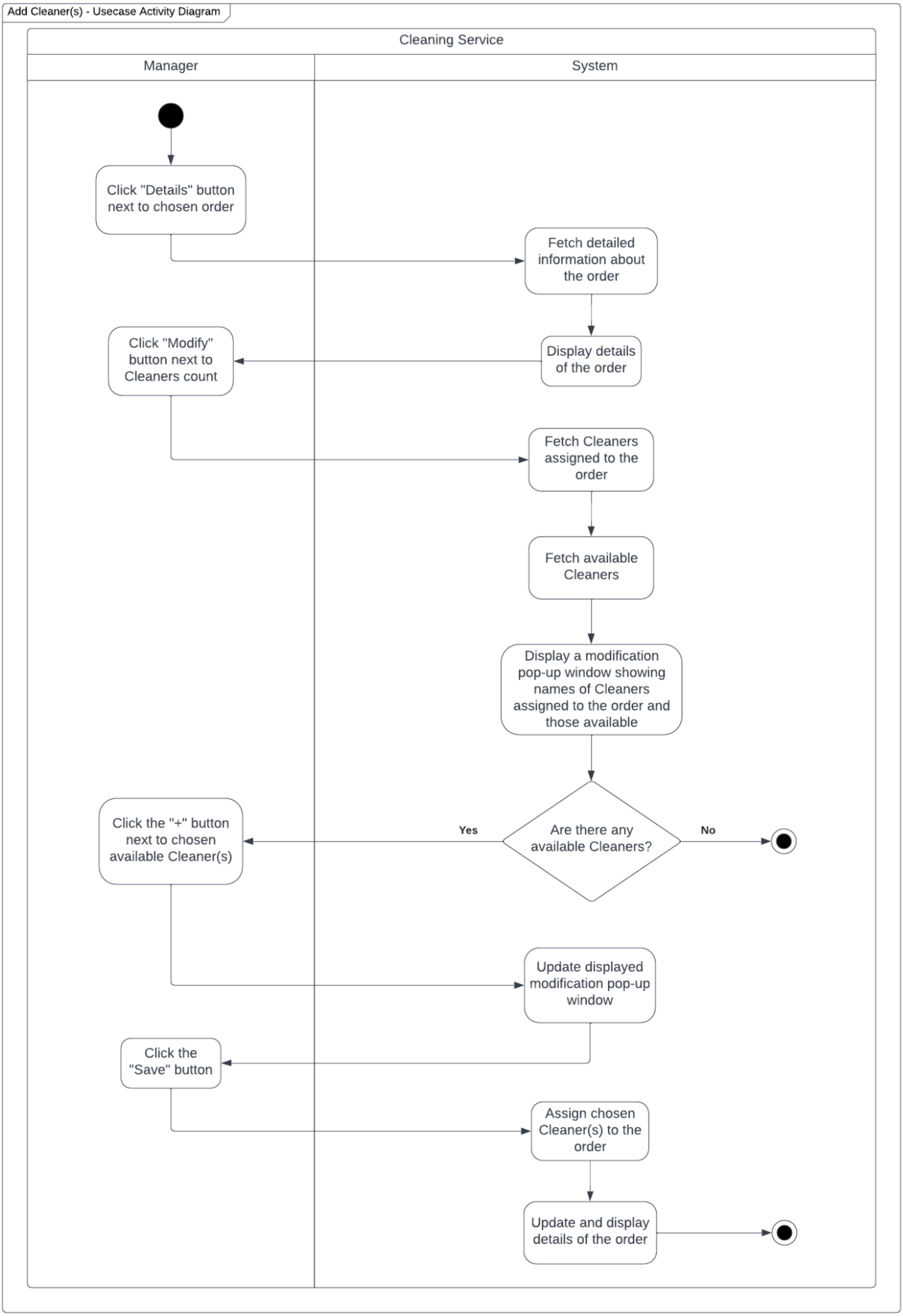

Add Cleaner(s) - Usecase Activity Diagram
Cleaning Service
Manager
System
Click "Details" button next to chosen order
Fetch detailed information about the order
Display details of the order
Click "Modify" button next to Cleaners count
Fetch Cleaners assigned to the order
Fetch available Cleaners
Display a modification pop-up window showing names of Cleaners assigned to the order and those available
Are there any available Cleaners?
Yes
No
Click the "+" button next to chosen available Cleaner(s)
Update displayed modification pop-up window
Click the "Save" button
Assign chosen Cleaner(s) to the order
Update and display details of the order

Fig. 4 “Add Cleaner(s)” Activity Diagram

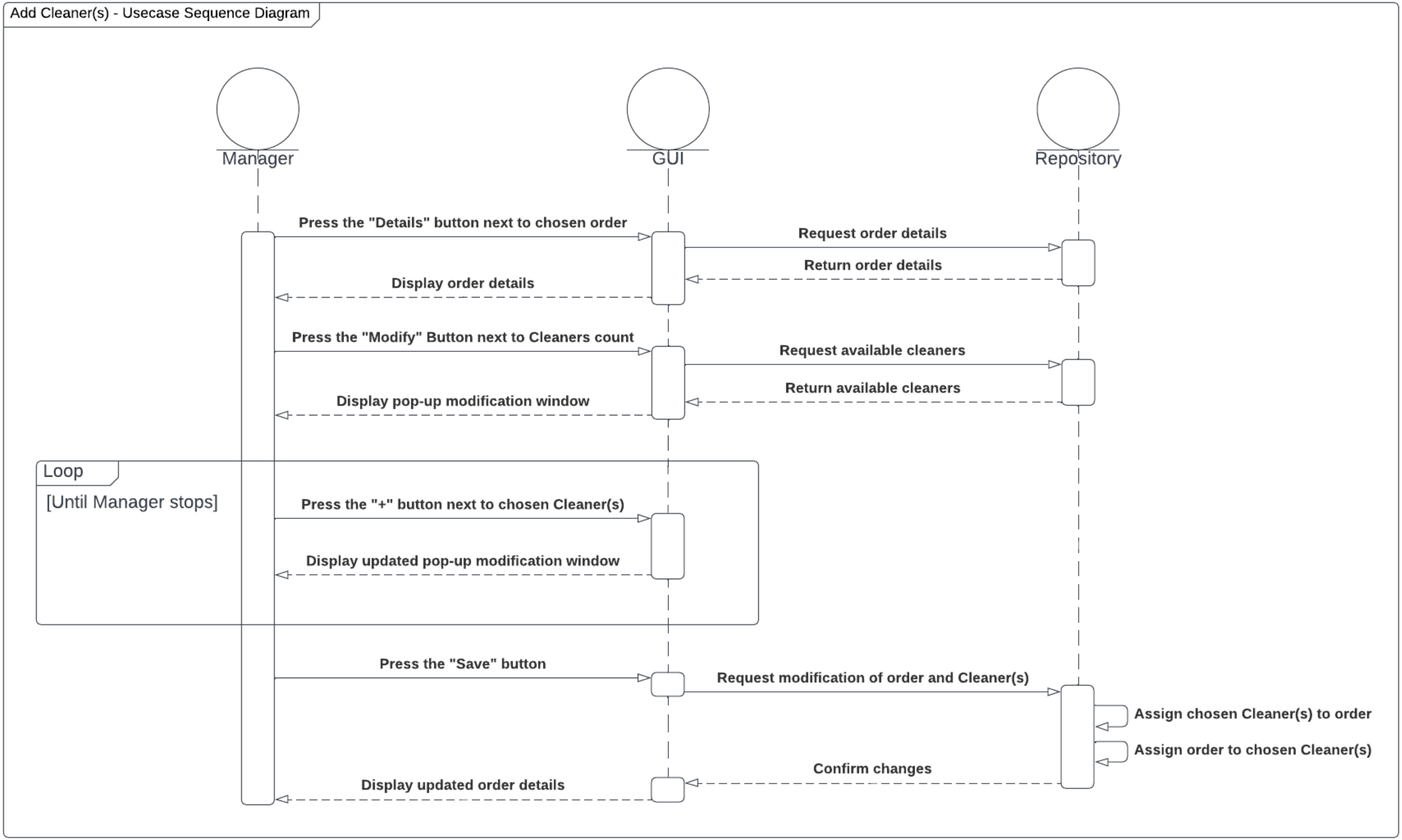

Add Cleaner(s) - Usecase Sequence Diagram
Manager
GUI
Repository
Press the "Details" button next to chosen order
Request order details
Return order details
Display order details
Press the "Modify" Button next to Cleaners count
Request available cleaners
Return available cleaners
Display pop-up modification window
Loop
[Until Manager stops]
Press the "+" button next to chosen Cleaner(s)
Display updated pop-up modification window
Press the "Save" button
Request modification of order and Cleaner(s)
Assign chosen Cleaner(s) to order
Assign order to chosen Cleaner(s)
Confirm changes
Display updated order details

Fig. 5 “Add Cleaner(s)” Sequence Diagram

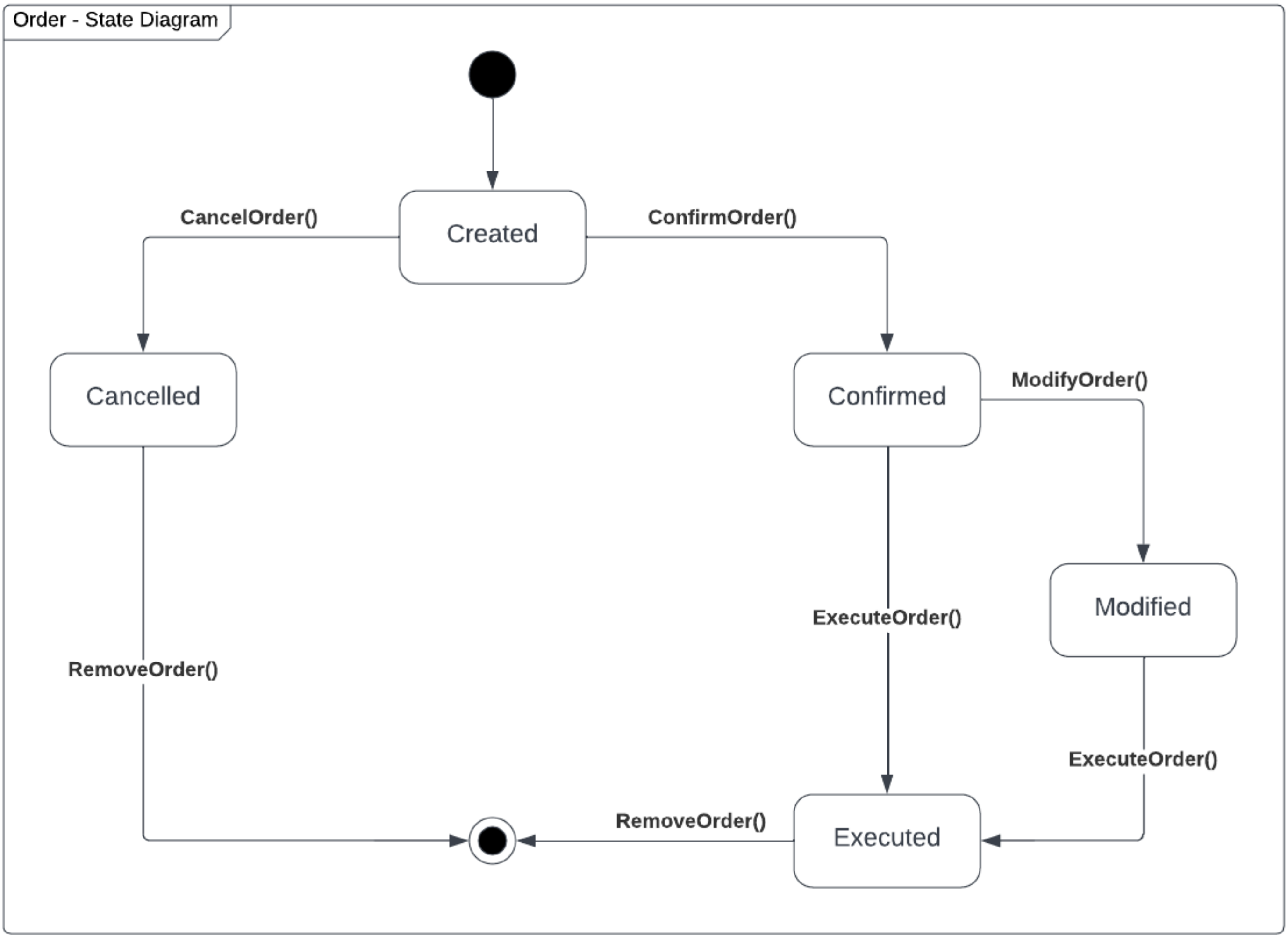


Fig. 6 Order - State Diagram

“Add Cleaner(s)” - Dynamic Analysis Discussions

After performing the *Dynamic Analysis* on the *“Add Cleaners”* use-case, results of which are visible in the previous section of this document, it is clear that a few functionalities are missing in the *Design Class Diagram* (**Fig. 3**). These functionalities were later added, contributing to the new *Final Design Class Diagram* (**Fig. 7**)(functionalities marked in red), and are as follows:

Manager Class

- **GetConfirmedOrders()** - A method returning the list of all previously confirmed orders.
- **GetAvailableCleaners(DateTime date)** - A method returning the list of all Cleaners available on a given **date**.
- **AssignCleanerToOrder(Order order, Cleaner cleaner)** - A method used for assigning a chosen **cleaner** to a chosen **order**.

Order Class

- **AddCleaner()** - A method used for assigning a new Cleaner.

Cleaner Class

- **AddOrder()** - A method used for assigning a new order.

Final Design Class Diagram

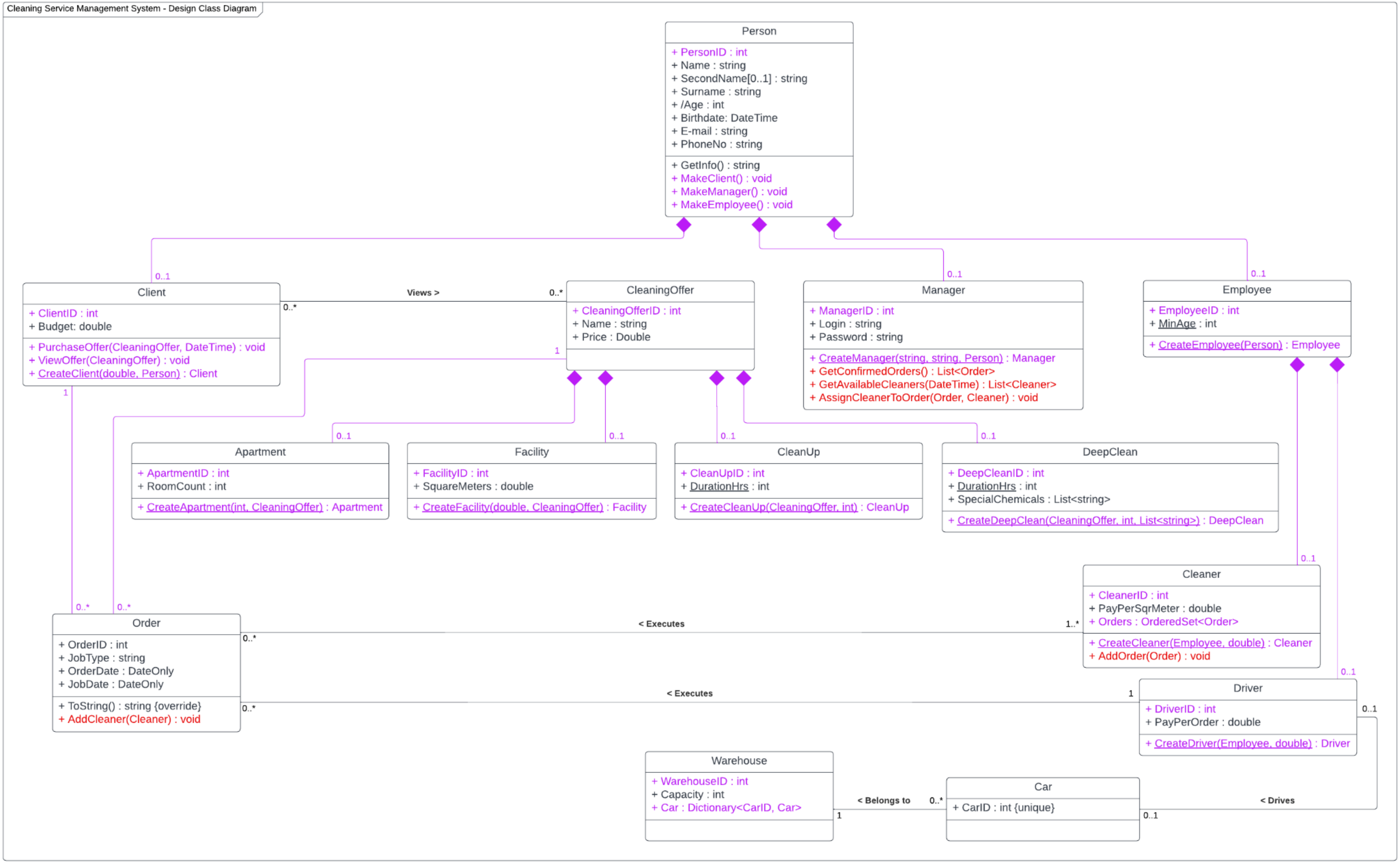


Fig. 7 Final Design Class Diagram

The GUI Design

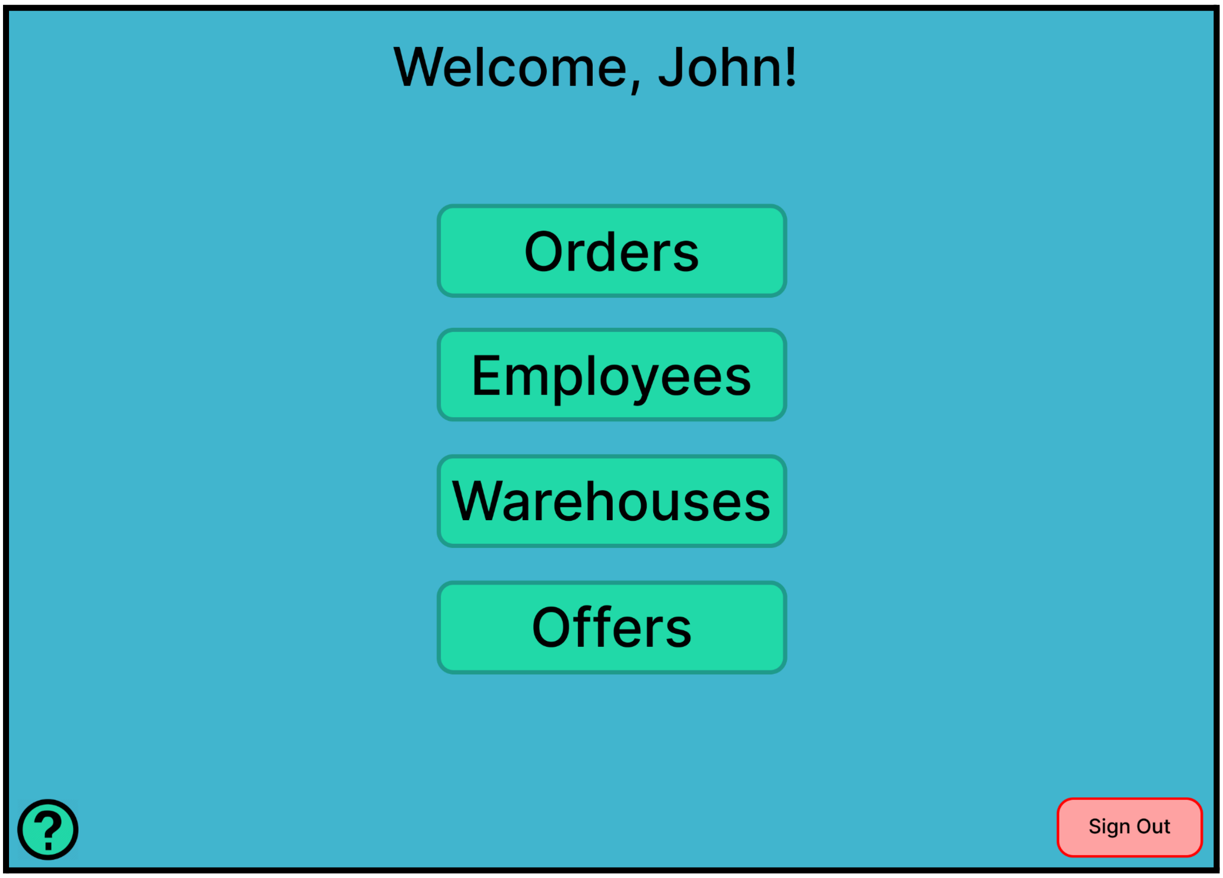
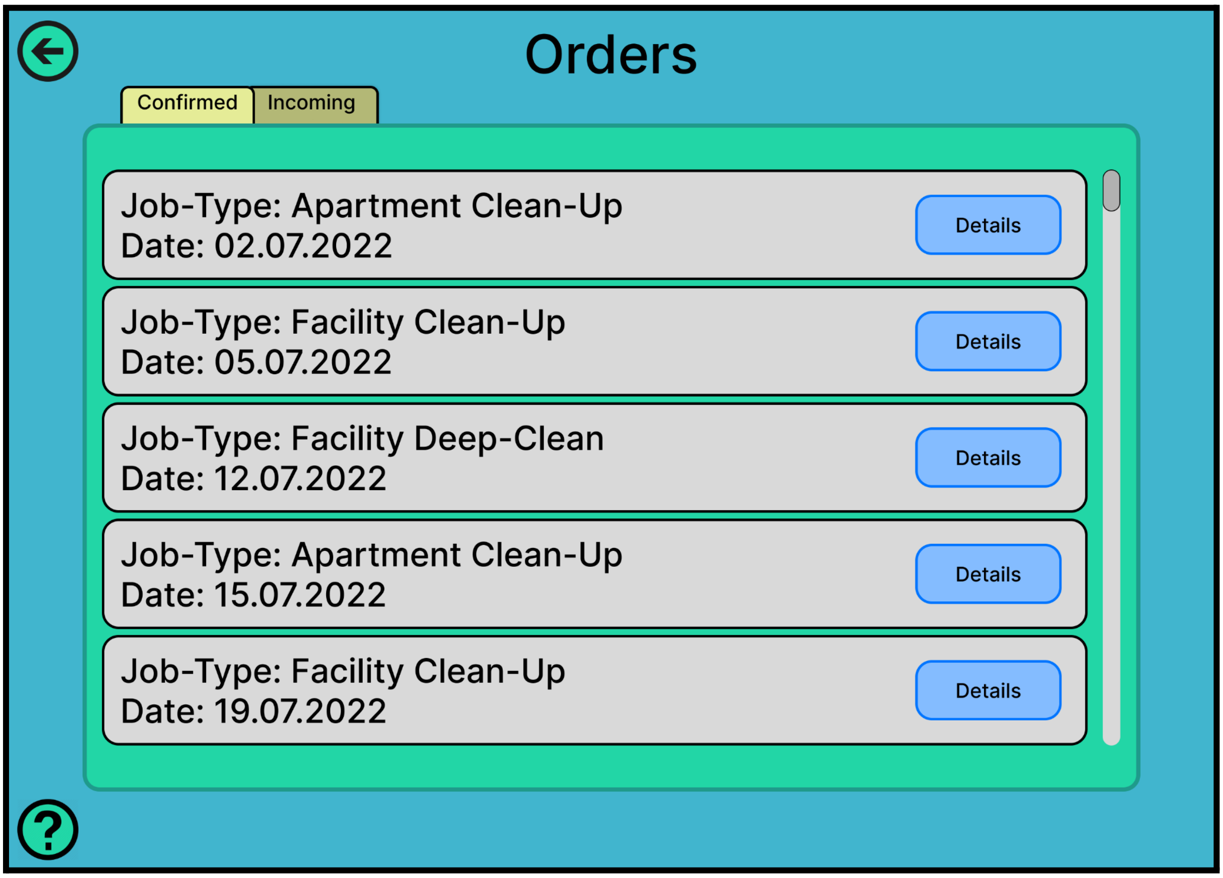


Fig. 8 GUI - Manager “Main-Menu” Screen



Job-Type: Apartment Clean-Up
Date: 02.07.2022

Job-Type: Facility Clean-Up
Date: 05.07.2022

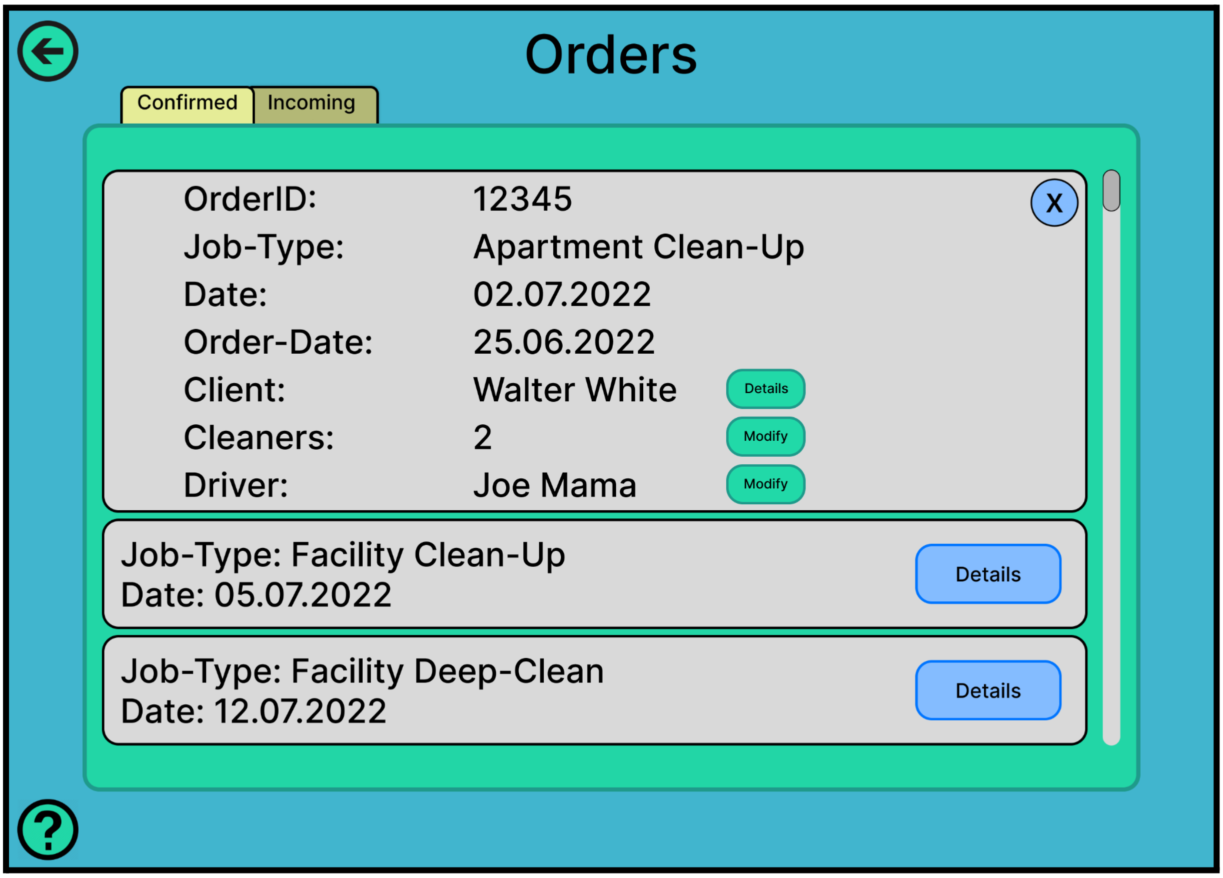
Job-Type: Facility Deep-Clean
Date: 12.07.2022

Job-Type: Apartment Clean-Up
Date: 15.07.2022

Job-Type: Facility Clean-Up
Date: 19.07.2022

Fig. 9 GUI - Manager “Orders” Screen

NOTE: The GUI design may change during the implementation.



Orders

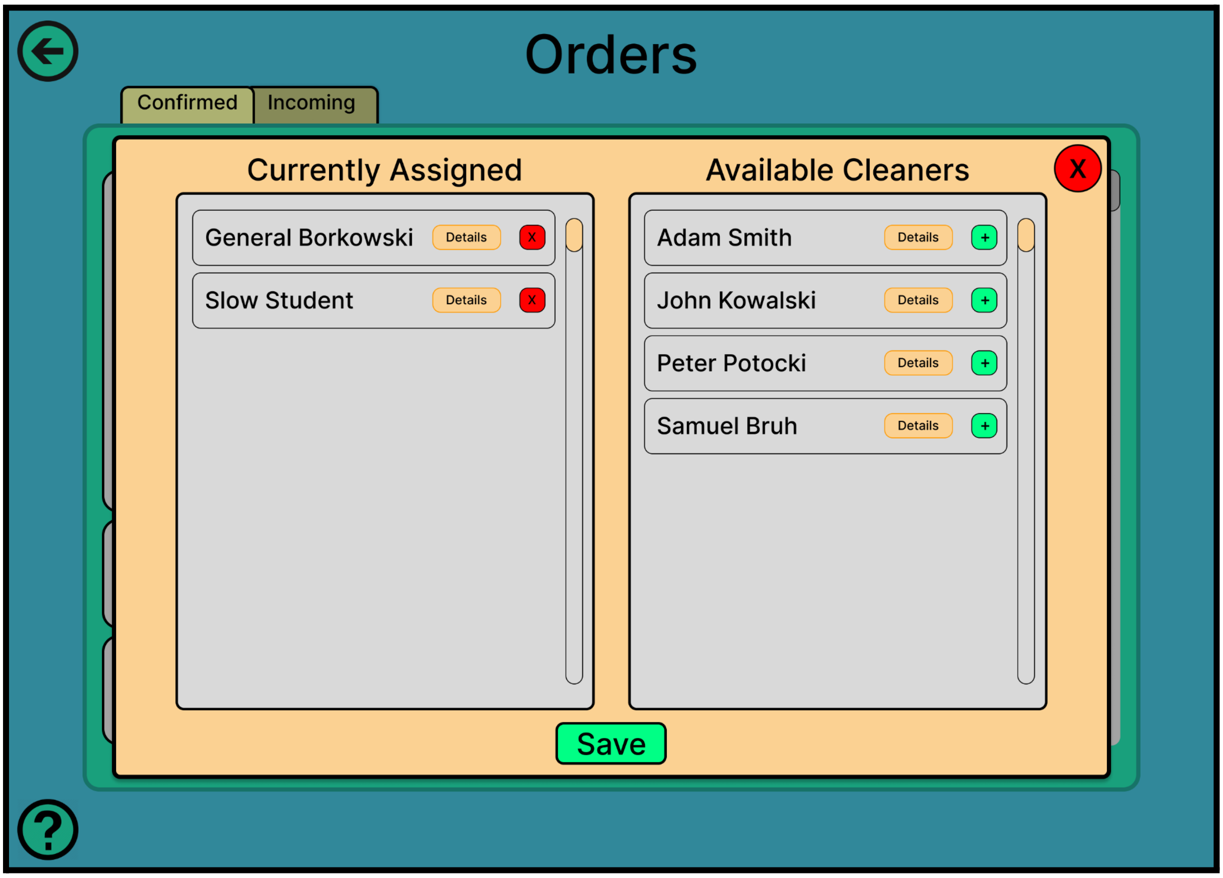
Confirmed | Incoming

OrderID: 12345
Job-Type: Apartment Clean-Up
Date: 02.07.2022
Order-Date: 25.06.2022
Client: Walter White
Cleaners: 2
Driver: Joe Mama

Job-Type: Facility Clean-Up
Date: 05.07.2022

Job-Type: Facility Deep-Clean
Date: 12.07.2022

Fig. 10 GUI - Manager “Order Details” Screen



Orders

Confirmed | Incoming

Currently Assigned

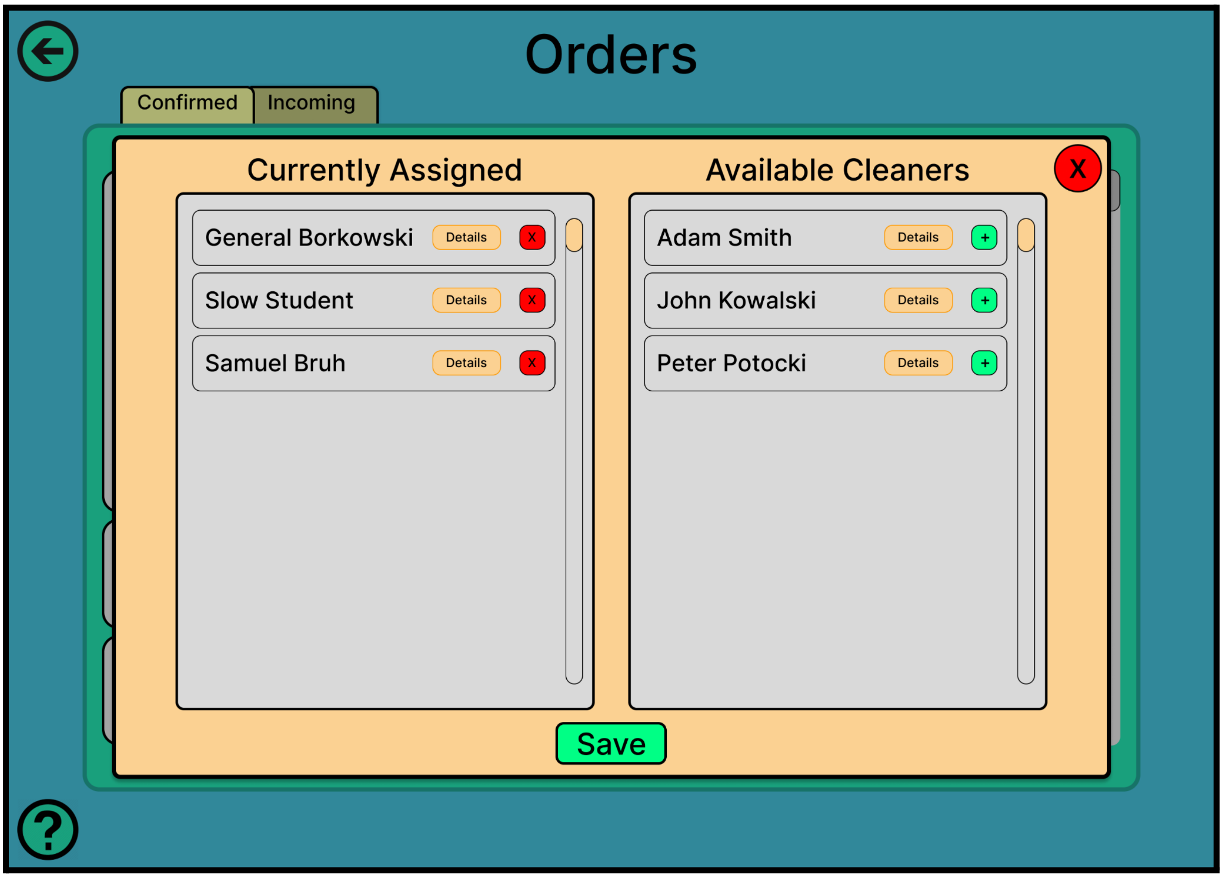
General Borkowski
Slow Student

Available Cleaners

Adam Smith
John Kowalski
Peter Potocki
Samuel Bruh

Fig. 11 GUI - Manager “Modify Cleaners” Screen

NOTE: The GUI design may change during the implementation.



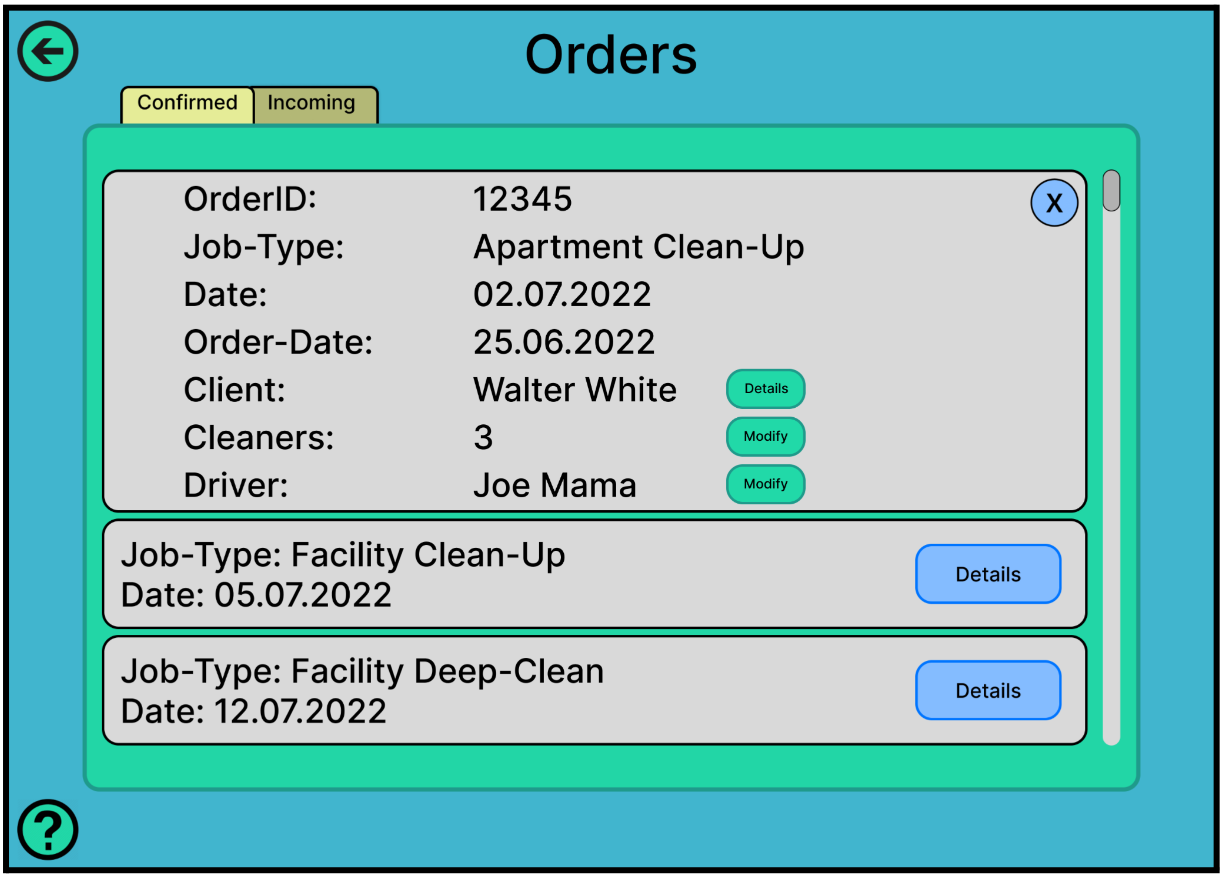
Currently Assigned:

- General Borkowski — Details — X
- Slow Student — Details — X
- Samuel Bruh — Details — X

Available Cleaners:

- Adam Smith — Details — +
- John Kowalski — Details — +
- Peter Potocki — Details — +

Fig. 12 GUI - Manager “Modify Cleaners” Screen (2)



OrderID:	12345
Job-Type:	Apartment Clean-Up
Date:	02.07.2022
Order-Date:	25.06.2022
Client:	Walter White
Cleaners:	3
Driver:	Joe Mama

Job-Type: Facility Clean-Up
Date: 05.07.2022

Job-Type: Facility Deep-Clean
Date: 12.07.2022

Fig. 13 GUI - Manager “Order Details” Screen (2)

NOTE: The GUI design may change during the implementation.

Table of Figures