VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

HO CHI MINH UNIVERSITY OF TECHNOLOGY

□···☼···□

ASSIGNMENT

COURSE NAME: SOFTWARE ENGINEERING

TASK 1: REQUIREMENT ELICITATION

CLASS CC02 - GROUP 2

SUBMISSION DATE: 11/02/2023

Student Name	Student ID	Mark
Lê Việt Hoàng	2052093	
Nguyễn Đình Khang	2052519	
Bùi Võ Công Phu	2053326	
Lê Châu Công Thành	2052256	
Nguyễn Tiến Trung	2052353	

Ho Chi Minh City – Feb 2023

Table of Contents

Task 1. REQUIREMENT ENGINEERING

I. Domain context of Urban Waste Management in Vietnam

In recent years, Vietnam has significantly grown in economy and population. This causes a rising in Urban Waste, especially in the big cities like Ha Noi, Ho Chi Minh, Hai Phong, and Da Nang. Currently, the amount of solid waste is at an average of 35,000 tons in cities, and 34,000 tons in rural areas **[1]**. The amount of solid waste increased sharply from 2013 to 2022, which is by 80% (shown in the figure below). Especially in Hanoi, the waste volume collected increases by 10% per year due to the growth of the population and economy **[2]**.

Figure 1: Solid waste in Vietnam (Source: Optimal Waste-to-Energy Strategy Assisted by Fuzzy MCDM Model for Sustainable Solid Waste Management (mdpi.com))

Due to the various transportation system contexts of Vietnam, there are different processes of waste collection. For instance in Hanoi, everyday household waste is gathered at specified areas in boxes or containers that are allocated across the road in citizen areas or in front of the house. After that, janitors and collectors drive small trucks or trolls to each gathering point to collect the waste and bring them to the major collecting points (MCPs). Besides that, only in the central areas of Ha Noi, there are more than 138 gathering points with more than 1000 waste bins **[3]**. Also, collection and transportation costs are about 86.7%

total cost while the revenue from waste collection fees from residence and business services only cover 16% of total revenue **[4]**. The cost of waste management, waste collection, and transportation account for up to 70% of the total costs of the process **[4]**. The factors that affect the cost of collecting and transporting waste can be the quantity of waste, the number of MCPs, the traffic route, etc.

From the context above, there should be a solution that handles the waste collection and transporting process in Vietnam. This solution should be provided to the Waste Collection Service Provider to take care of waste management efficiently by reducing the cost, assigning and tracking the daily tasks of employees, and monitoring their vehicles. Also, the solution must have the function to track workers' tasks and important information daily, as well as interactively communicate between workers and officers in real time.

The UWC 2.0 is a solution that uses technology to meet the mentioned requirements in managing the waste collection process. From the manager's perspective, the UWC 2.0 provides the feature to track the tasks of the employee, information, and status of vehicles (fuel consumption, traveling routes, etc) that help the business perform more efficiently. Also, the software suggests an efficient collecting route that reduces the cost of fuel consumption and saves plenty of time. Furthermore, this application shows the workers a comprehensive view of their essential information (tasks, reminders, etc) and provides an effective communication channel between stakeholders during working time.

II. Functional and non-functional requirements

1. Functional requirements

1.1. General requirements:

- Janitors, collectors, and back officers synchronously message with each other
- The system is capable of displaying on both mobile phones and personal computers
- User authentication (login/logout)

1.2 Back Officer requirements

- View janitors and collectors info (personal info, and work info => work calendar)
- View vehicles' technical and status detail
- View MCP detail and their status
- Assign vehicles and tasks for collectors and janitors
- Create a route for a certain vehicle
- The assigned route is optimized in terms of fuel consumption and travel distance.
- Send messages with information about collecting, route and time to collectors and janitors

1.3 Janitor and Collector requirements

1.3.1. Requirements for both janitor and collector:

- Check-in/Check-out tasks
- Informed as a particular MCPs is fully loaded
- View their work calendar under either weekly or daily mode

1.3.2 Janitor

- Informed the assigned MCPs he/she will be placed in a certain shift

1.3.3. Collector

- Informed the route he/she will visit to collect garbage

2. Non-functional requirements

2.1 Product requirement

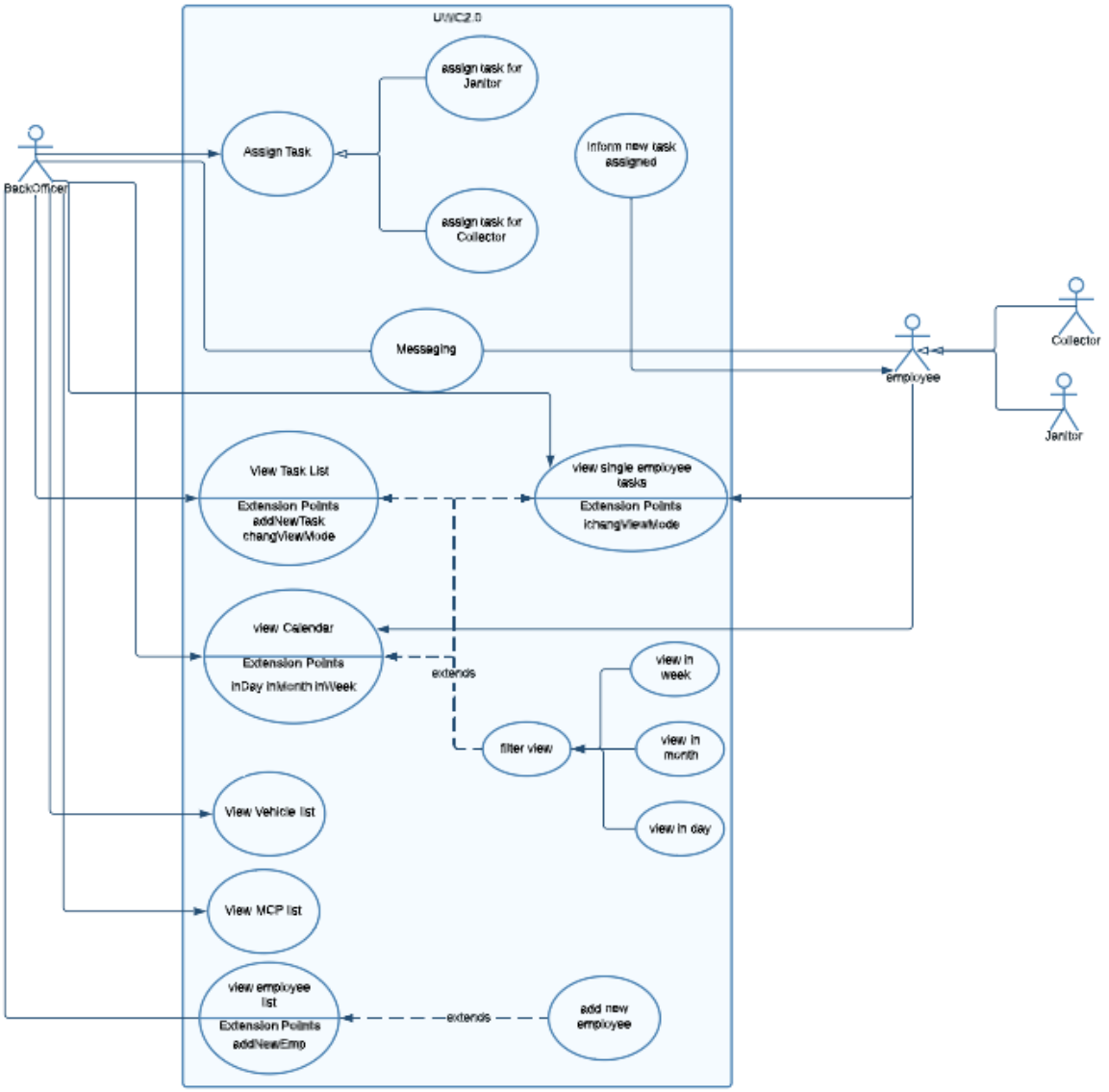
- Compatible with UWC1.0
- Able to accommodate history data from the database connected to UWC1.0
- The generated route is optimized in terms of fuel consumption and travel distance (need to be more specific)
- Real-time communication between system user is to be less than 1 second
- The system should be able to handle real-time data from at least 1000 MCPs at the moment and 10.000 MCPs in five years

- The system must be able to handle SQL injection

2.2. Organization requirements

- The system interface should be displayed in Vietnamese
- A legit account must be logged in with the employee id provided by the organization

3. General use-case diagram of the whole system



https://lucid.app/lucidchart/c8185c69-9d9a-4f03-bdc9-0bc61f4d4e35/edit?viewport_loc=-1518%2C814%2C3791%2C1962%2CKrUIKE8HTqX.&invitationId=inv_8ec63c19-4898-4a5c-b6ef-57a6f8b206b2

III. Draw Task Assignment use-case diagram and describe the use-case using a table format

1. Task assignment general use case

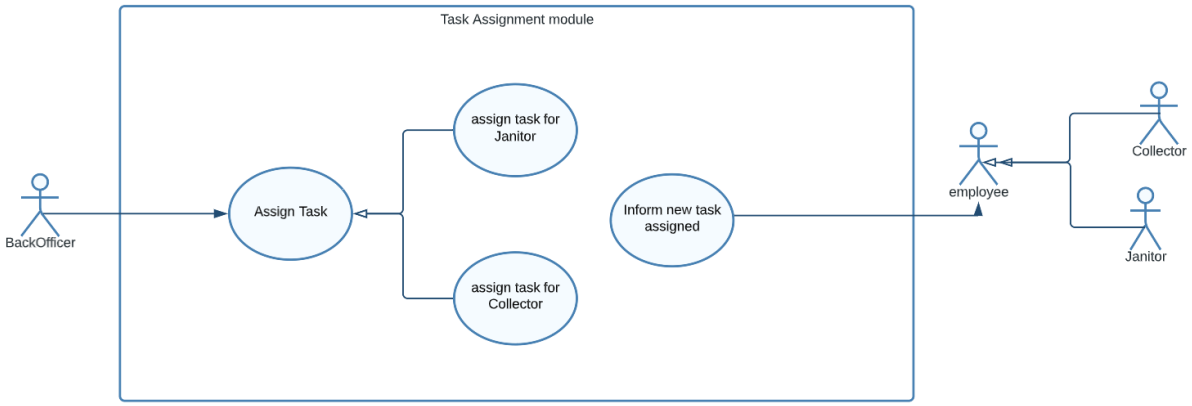


Figure 3: Use-case Diagram of the general Task assignment case

Use-case ID Use-case Name	20 task assignment
Use Case overview	Allow the user to assign the task to either janitors or collectors
Actors	Back Officers
Preconditions	Back officers logged in with a legit account
Triggers	button named “Add Task” is clicked
Steps	Basic Flow - A pop-up window for assign task appears with “Janitor” and “Collector” option. - User opts for one of those options. - System change to the appropriate UI respective to which option selected. Alternative flows - Janitor flow. - Collector flow.
Postconditions	- A new task is created and its detail is saved in databases - A automatic notification will come to the collector or janitor device. - Attributes of involving objects will be updated.
Exception flow	None.

1.1. Task assignment use case for janitors

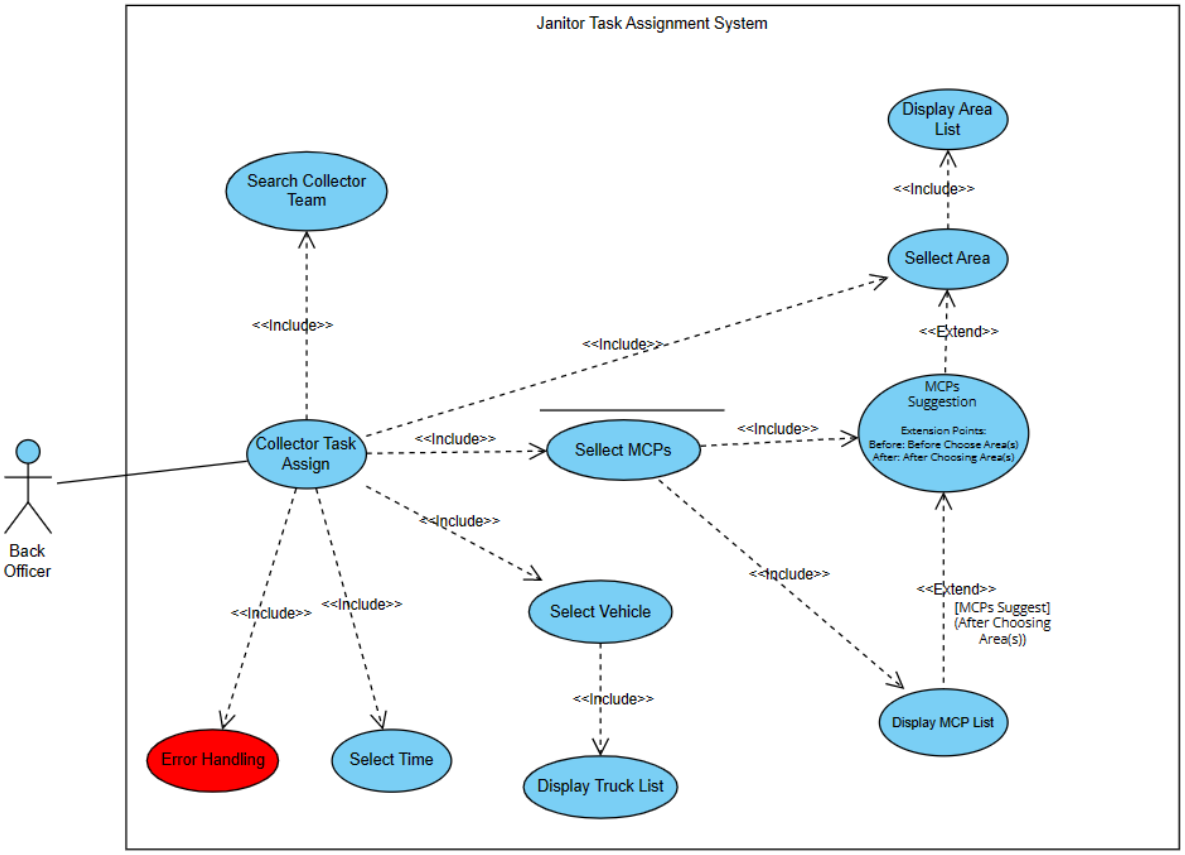


Figure 4: Use-case Diagram of the Task assignment for janitors

Use-case ID Use-case Name	21 Janitor task assignment
Use Case overview	Assign tasks to janitors to an appropriate task form consisting of Task name, a number of places to collect, start and end time of the task, MCPs information, and Important note.
Actors	Back Officers
Preconditions	- The system is online. - The user is already clicked on the “Add task” UI.
Triggers	Janitor option in “Create Task Window” is selected.
Steps	Basic Flow 1. The user fills in the task name field. 2. The user chooses the start and end times of the task. 3. User chooses the available MCPs location. 4. User chooses the areas to be collected (the area that is related to the MCPs).

	5. User chooses the available Janitor team to do the task. 6. User adds Important note (Optional). 7. User click Save to assign the task.
Postconditions	- A new task is created and its detail is saved in databases - A automatic notification will come to the collector or janitor device. - Attributes of involving objects will be updated.
Exception flow	Exception 1: The user does not fill in all the required information. The system will announce the error. Exception 2: The Data about the MCP, Collector Team or Vehicle is not available in the system

1.2. Task assignment use case for collectors

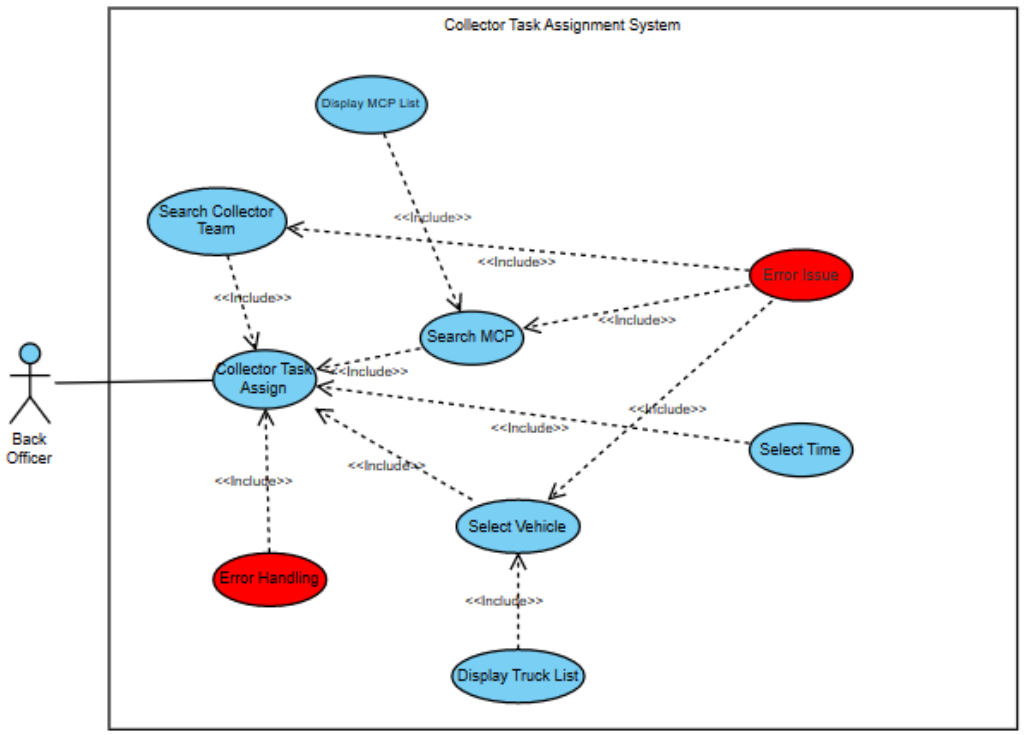


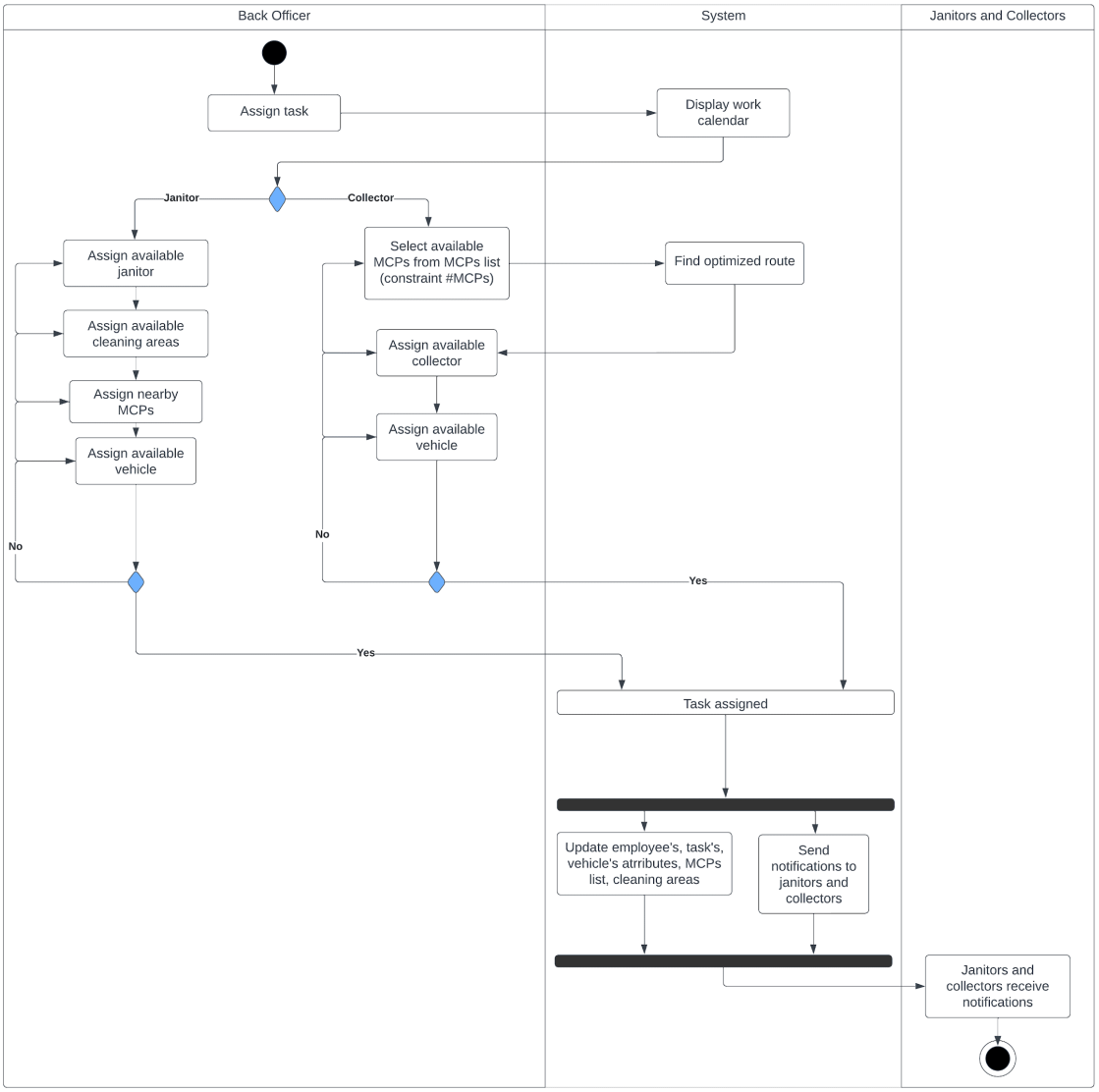
Figure 5: Use-case Diagram of the Task assignment for Collectors

Use-case ID Use-case Name	22 Collector task assignment

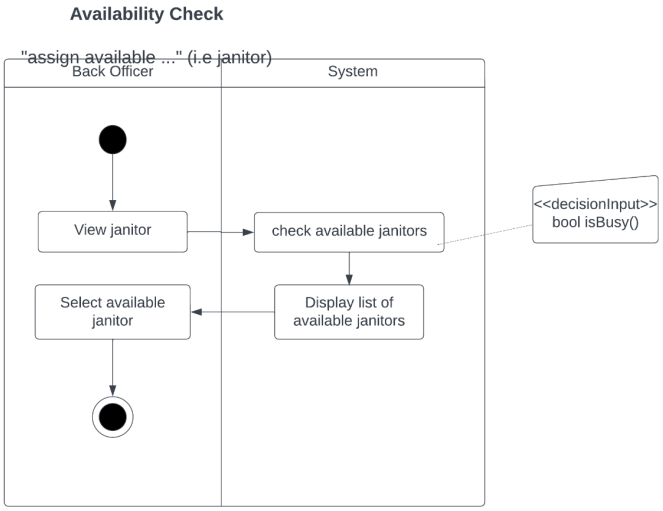
Use Case overview	Assign tasks to collectors to an appropriate task form consisting of Task name, a number of MCPs to collect, vehicle, and start and end time of a task.
Actors	Back Officers
Preconditions	- The system is running. - The user is at the “Add Tasks” UI.
Triggers	Collector option in “Add Task” UI.
Steps	1. User fills in the task name field. 2. User chooses the start and end time of the task. 3. User chooses the available vehicle. 4. User chooses the MCPs location. 5. User chooses the available Collector team to do the task. 6. User adds Important note (Optional). 7. User click Save to assign the task.
Postconditions	- Assigned tasks and their details are stored in the database. - A notification will come to the collector’s device. - Route for the vehicle is updated.
Exception flow	Exception 1: User does not fill in all required information. The system will announce the error. Exception 2: The Data about the MCP, Collector Team or Vehicle is not available in the system

Task 2. SYSTEM MODELING

I. Activity diagram for Task assignment module



Link to diagram: Activity Diagram_v2.png



Link to diagram: Availability check_1.png

Depending on the type of employees (decisionInput isJanitor()), the system navigates the Back Officers to select between Janitor and Collector. In case the back officers want to assign a task to a janitor, they are guided step by step to select the available janitor, the unassigned cleaning areas, the nearby Major Collecting Point (MCP), and the available troller (vehicle). The availability check process takes the employees' method isBusy() as input and returns the available choices to the back officers.

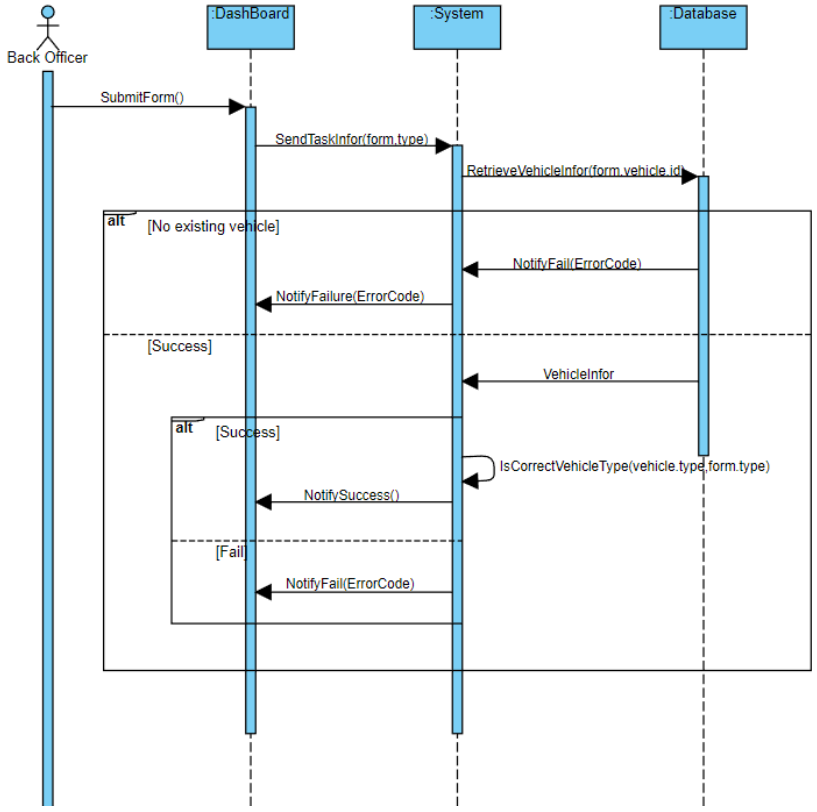
After the back officers confirm their assignment, the system updates the information such as the janitor's attributes, vehicles' attributes, MCPs list, etc. to the database and sends the complete notification of task to the janitor via message. If the back officers have a second thought on each field before confirmation, they can return backward to modify their selections.

The process of collector is similar to that of the janitor, except that the predetermined route must be optimized before assigning to a specific collector. The back officers select currently working MCPs (isFull() = True) among the MCPs list and the system measures the optimized route based on the distance and fuel consumption. The maximum number of MCPs traveled must be limited to assure that the capacity of a vehicle (truck) yields the weight of garbage collected along the route. The rest of the collector's activity follows the same path as janitors.

II. Sequence diagram for vehicle assignment process

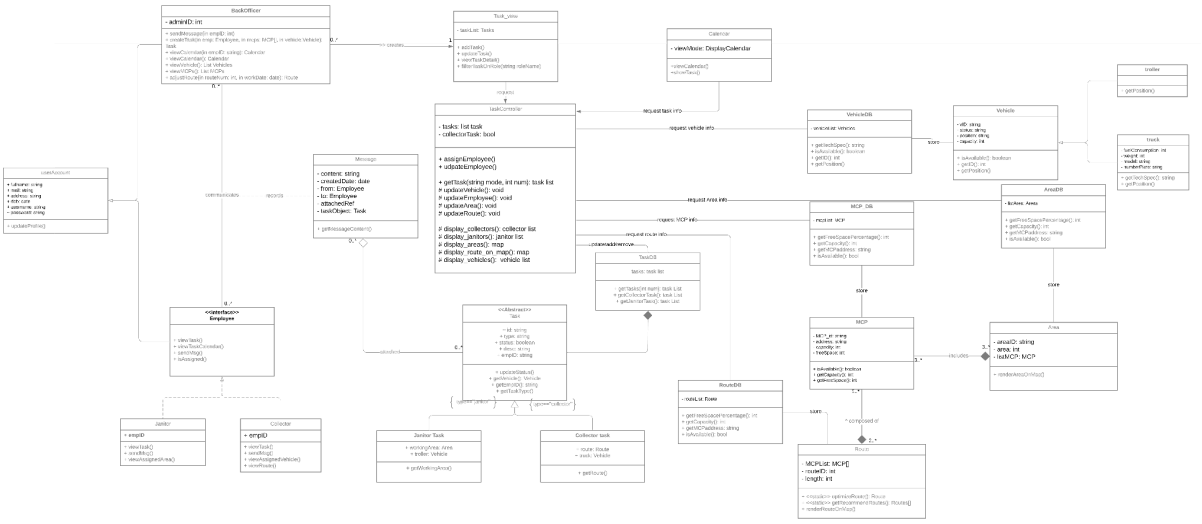
The following vehicle assignment sequence is drawn with the assumption that all the other fields are properly filled by the back officer. Therefore, the success or failure notification shown after the system carrying task creation process is completely based on the validity of the requested vehicle.

The process starts off as a back officer clicking the submit button on the dashboard interface. As a result, the form that is already filled is sent from the dashboard (browser) to the system (backend) - sendTaskInfo. Along with the form, a parameter indicates the form type (collector or janitor form) is also passed. This information is known when the back officer selects the option “Assign Janitor task” or “Assign Collector task”. The system, then, requests a database to retrieve information involving the vehicle. In case the information is not found (Non-existing vehicle), an error code is returned to the system. Otherwise, the vehicle information is returned. On success, the system initiates validation for the matching between the vehicle and the assigned person. A janitor must be assigned to a troller while a collector must be assigned to a truck. Afterward, the system signals DashBoard to display a dialog in order to notify “Task assign success”. However, on failure, the system will make DashBoard display “Task assign failed”.



Link to diagram: https://online.visual-paradigm.com/share.jsp?id=323339353633382d34

III. Class diagram of Task Assignment module



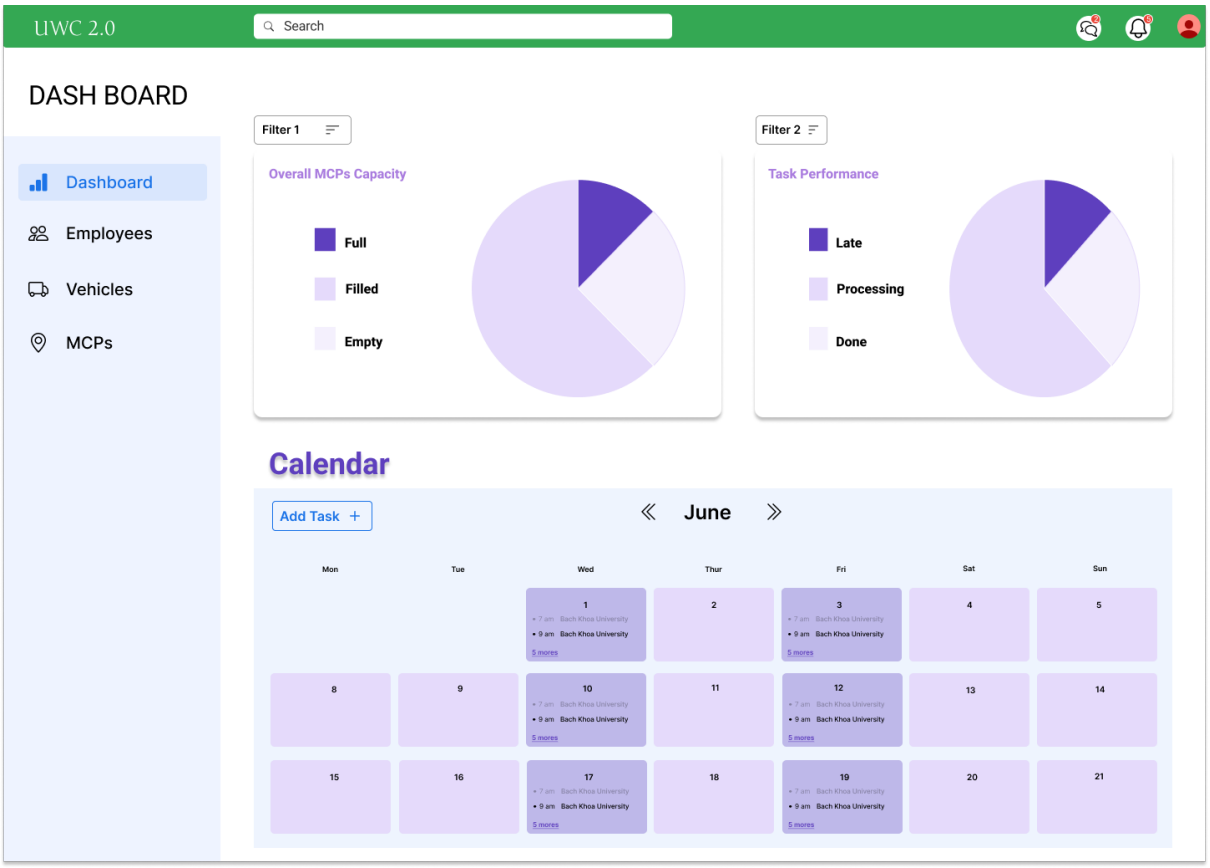
Link to diagram: https://rb.gy/cm1jcp

IV. The user interface of desktop view of Dashboard for Task Management:

1. **Components:**

- **Overall Dashboard:** View information about Calendar, Overall Status of MCP, and Overall Employee Performance. Also, provide the ability to view tasks and add tasks.
- **Employee:** View information about the Overall Performance of employees and View the Employee List. Also, provide the ability to Add, Update and Delete Employees.
- **Vehicles:** View information about the Vehicle's Overall Status, the Vehicle Location, and View the Vehicle List. Also, provide the ability to Add, Update and Delete Vehicles.
- **MCPs:** View information about the Overall Status of MCPs, Location of MCPs, and View the MCPs List.

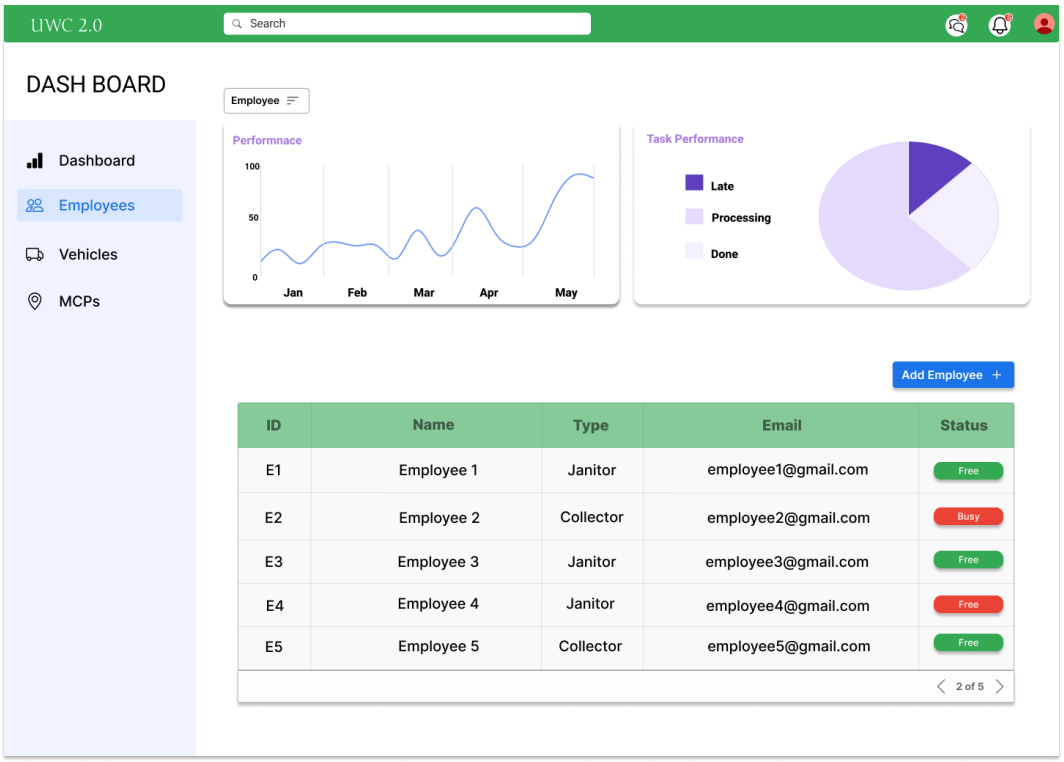
2. **Overall Dashboard:**



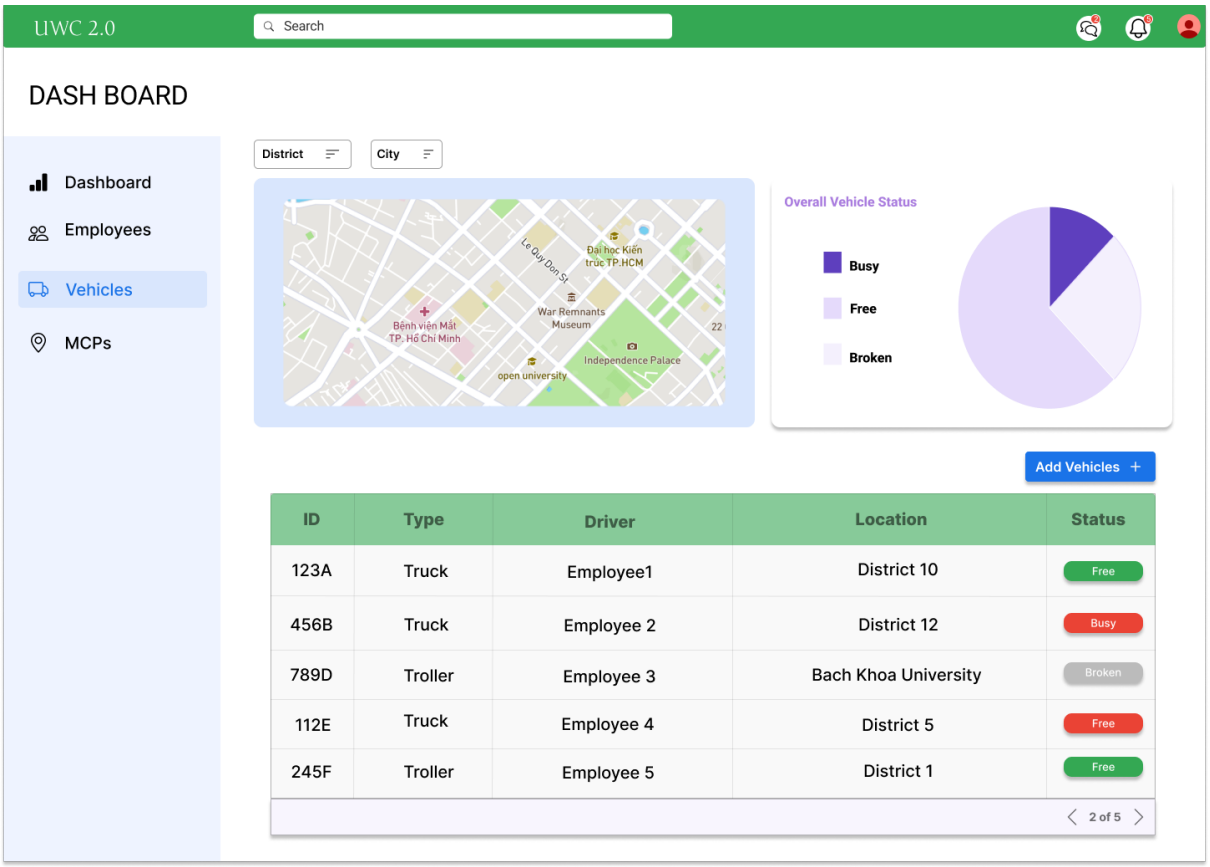
- The back officers can view tasks by month
- The back officers can view tasks of each day by clicking on the date symbol
- The back officers can add tasks by clicking on the Add Task button

3. Employee Board:

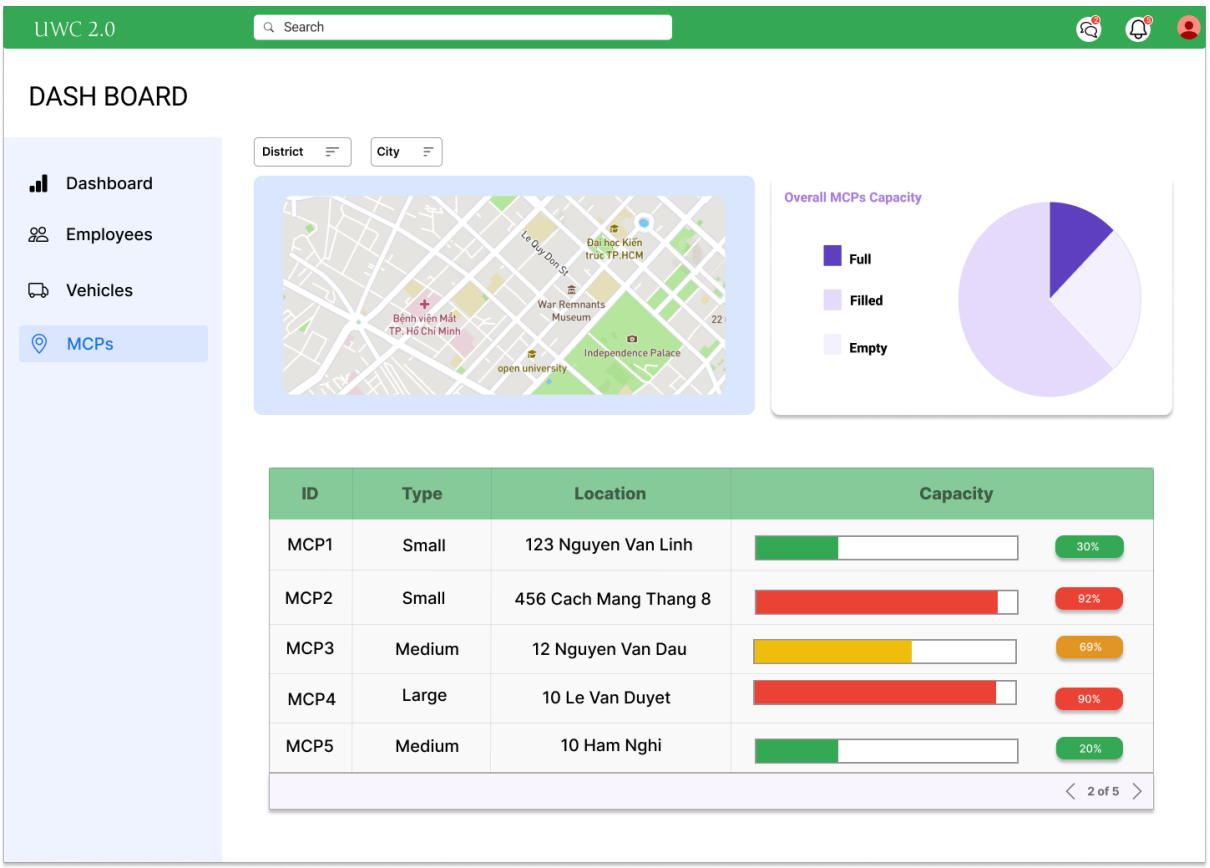
- The officers can view the Overall Performance of Employees
- The officers can view the list of employees they are managing
- The officers can Add, Update and Delete Employees by clicking on the Employee line. Then a pop-up window will appear.



4. **Vehicles Board:**
    - The officers can view the Overall Location and Status of Vehicles
    - The officers can view the list of vehicles they are managing
    - The officers can Add new Vehicles by clicking on the Add Vehicle button.


UWC 2.0
Search
DASH BOARD
Dashboard
Employees
Vehicles
MCPs
District
City
Overall Vehicle Status
Busy
Free
Broken
Add Vehicles +
ID
Type
Driver
Location
Status
123A
Truck
Employee1
District 10
Free
456B
Truck
Employee 2
District 12
Busy
789D
Troller
Employee 3
Bach Khoa University
Broken
112E
Truck
Employee 4
District 5
Free
245F
Troller
Employee 5
District 1
Free
2 of 5

5. MCPs Board:

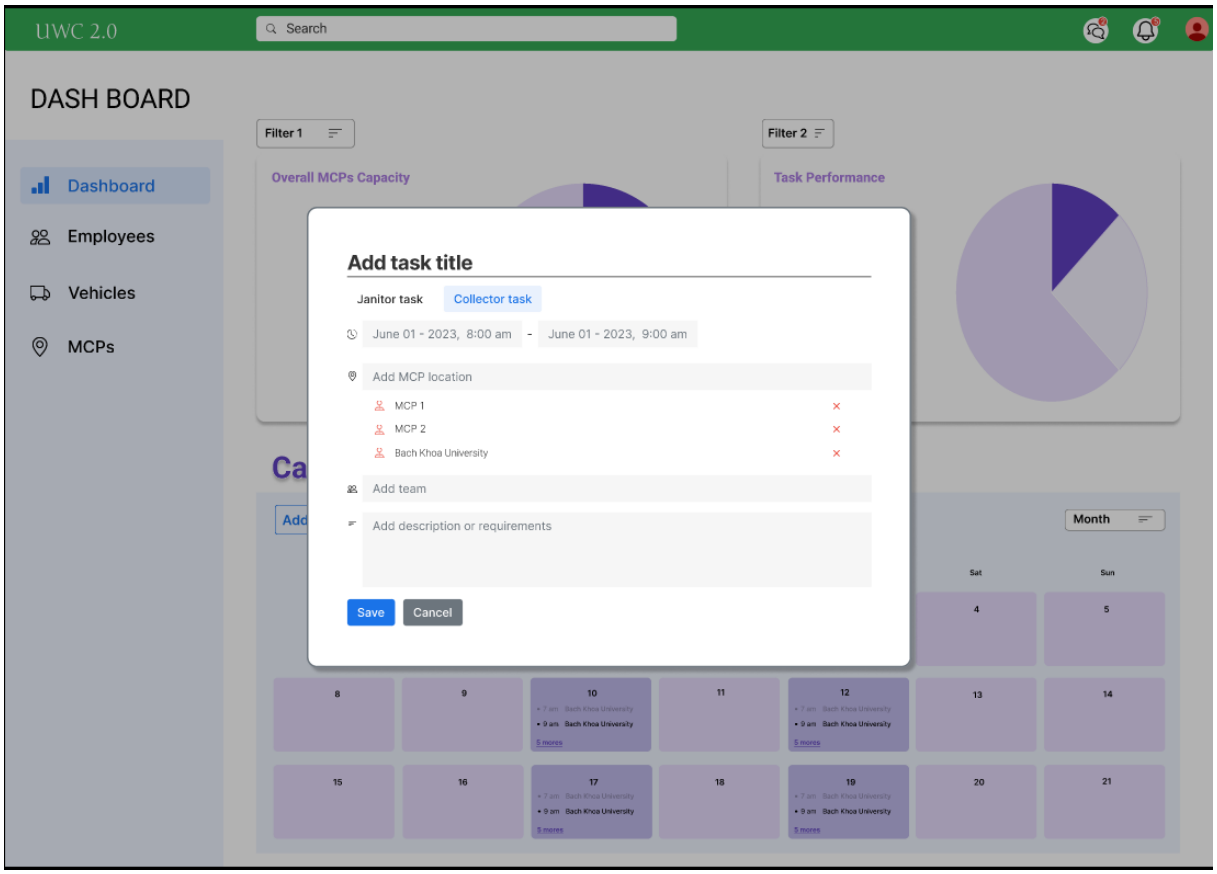

UWC 2.0
Search
DASH BOARD
Dashboard
Employees
Vehicles
MCPs
District
City
Overall MCPs Capacity
Full
Filled
Empty
ID
Type
Location
Capacity
MCP1
Small
123 Nguyen Van Linh
30%
MCP2
Small
456 Cach Mang Thang 8
92%
MCP3
Medium
12 Nguyen Van Dau
69%
MCP4
Large
10 Le Van Duyet
90%
MCP5
Medium
10 Ham Nghi
20%
2 of 5

- The back officers can view information on the MCPs (location, status, type, capacity).
- The back officer can view MCPs' location on map.

6. Interaction:

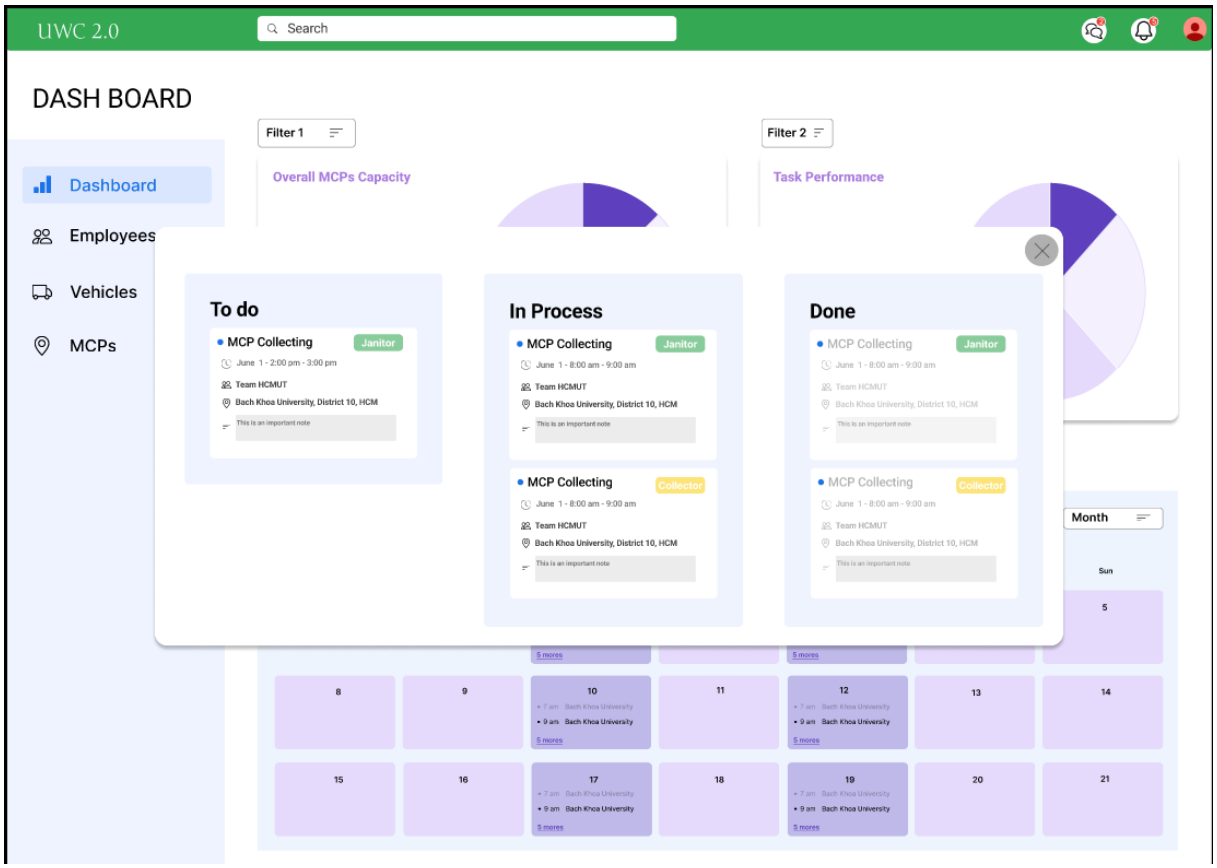
6.1. Creating Task:

The back officer can choose to create a task for either the janitor or the collector. When clicking the **"ADD TASK"** button, the popup display allows the back officers to input the task's information.



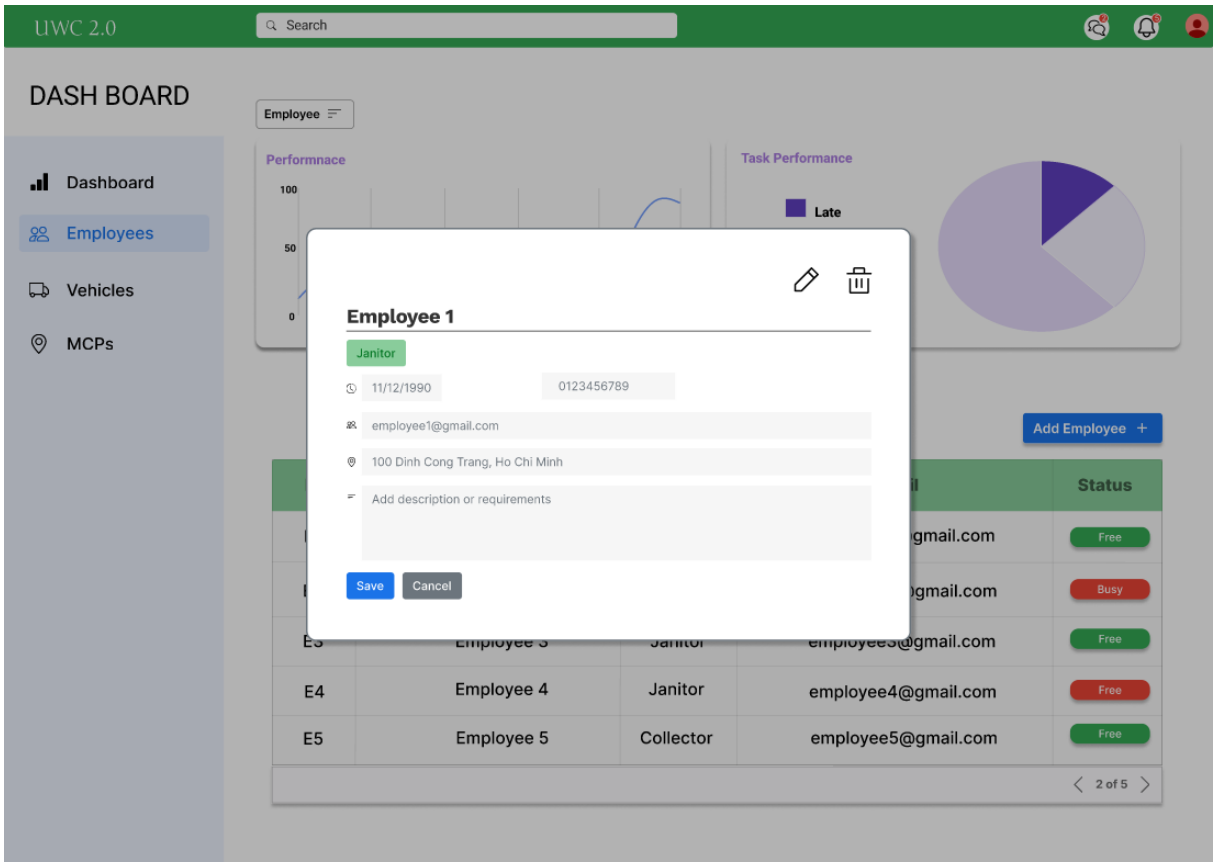
6.2. View Daily Task:

The back officer can view the daily task list by clicking on a corresponding day on Calendar



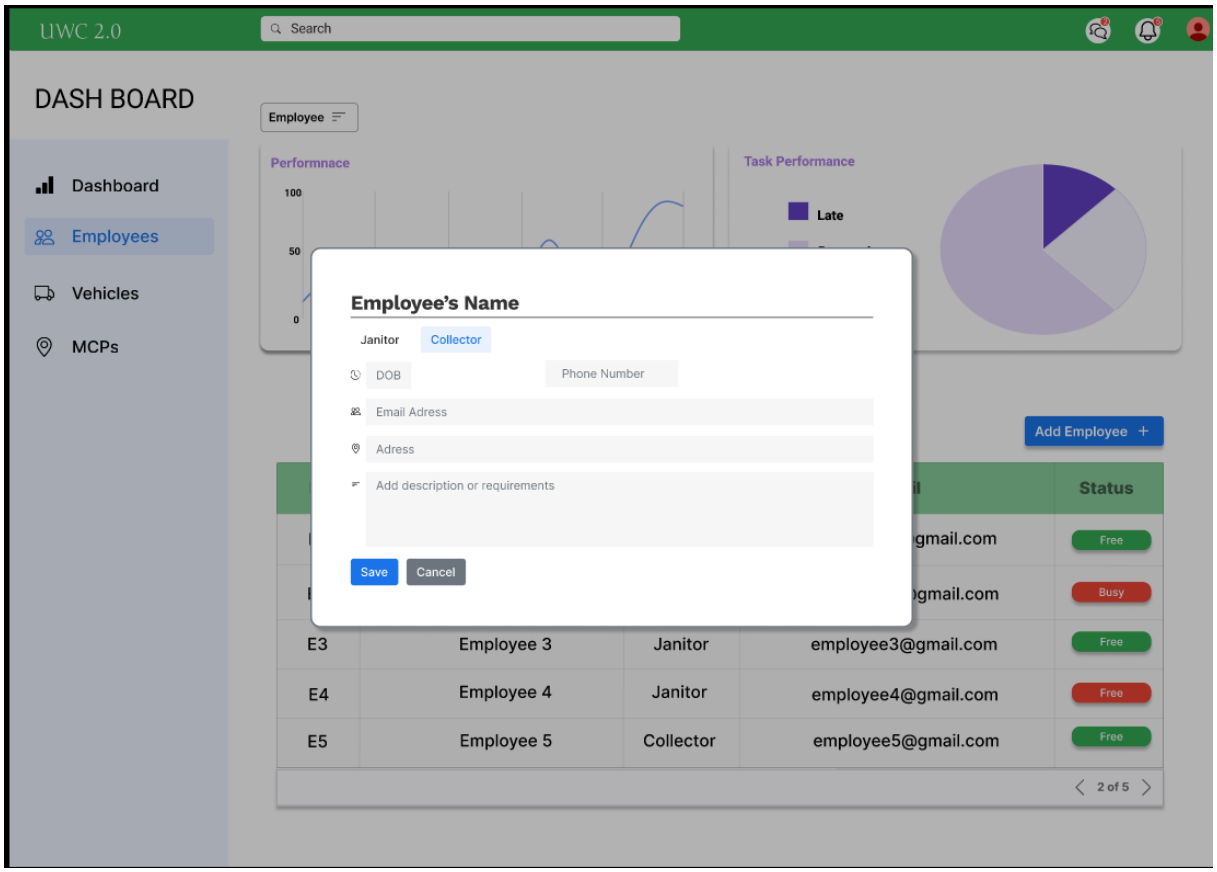
6.3. View Employee:

The officer can view and adjust employee information by clicking on the employee row in list



6.4. Add Employee:

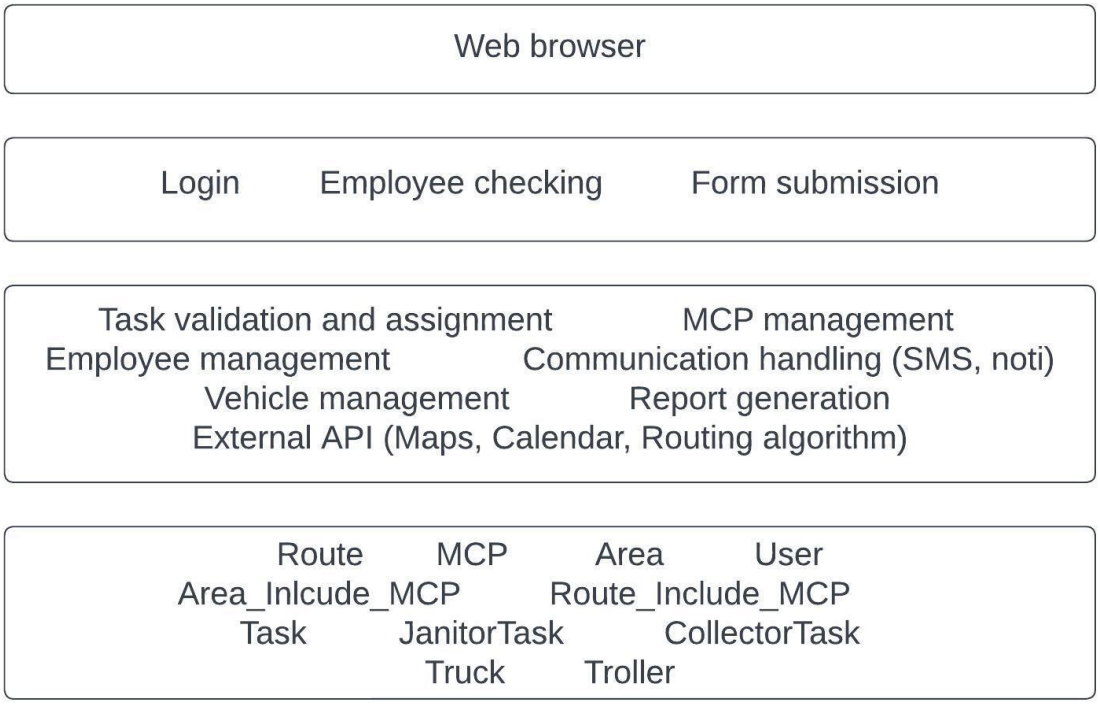
The officer can Add an Employee by clicking on the **"Add Employee"** button. The pop-up window will allow the back officer to add a new employee.



View the Figma design here.

Task 3. ARCHITECTURAL DESIGN

I. Layered Architecture of UWC2.0 system



1. User interface layer:

- Header Bar
- Navigator Bar
- Buttons
- Icons
- Calendar
- Table View
- Task Create Overlay - Collector
- Task Create Overlay - Janitor
- View Task Overlay - Collector
- View Task Overlay - Janitor
- Add Employee Overlay
- View Employee Overlay
- Charts
- Area Map
- View Task List Overlay

2. Application logic layer

- Employee Management
    - addEmployee(fullname, mail, address, dob, username)
    - displayProfile() calls getName(), getMail(), getID(), isAssigned()
    - updateProfile(fullname, mail, address, dob, username)
    - assignEmployee()
    - updateEmployee()
- Vehicle Management
    - displayVehicle() calls getID(), getPosition(), getTechSpec(), getStatus()
    - getDriver() calls getEmpID() from Abstract Task
- MCP Management
    - displayMCP() calls getID(), getCapacity(), getMCPAddress(), getFreeSpacePercentage()
- Task Assignment
    - displayTask() calls viewTaskDetail()
    - createTask(in emp: Employee, in area: Area[], in mcps: MCP[], in vehicle:Vehicle)
    - addTask() calls createTask(in emp: Employee, in mcps: MCP[], in vehicle:Vehicle)
    - updateTask() calls updateEmployee(), updateVehicle(), updateArea(), updateRoute()
    - removeTask()
- Supported API (map, routing algorithm, calendar)
    - displayMap()
    - displayCalendar()
    - bestRoute()
- Communication Handling (SMS, notifications)
    - sendMsg()
    - recvMsg()
    - showNoti()
- Report Generation
    - printDaily() calls viewTask() from Employee
    - printWeekly() calls viewTask() from Employee

The Business Logic Layer or the Application Logic Layer is the second layer of our layered architecture pattern. This layer uses the API provided by the Database Layer to extract required data to perform logic. Then, the Application Logic Layer provides the API to the Presentation Layer (UI), so that the users can interact and perform available functions displayed on the monitor.

We assume the employee authentication is implemented in the Presentation Layer; thus, the employee who logins into the system will know his/her type of employee.

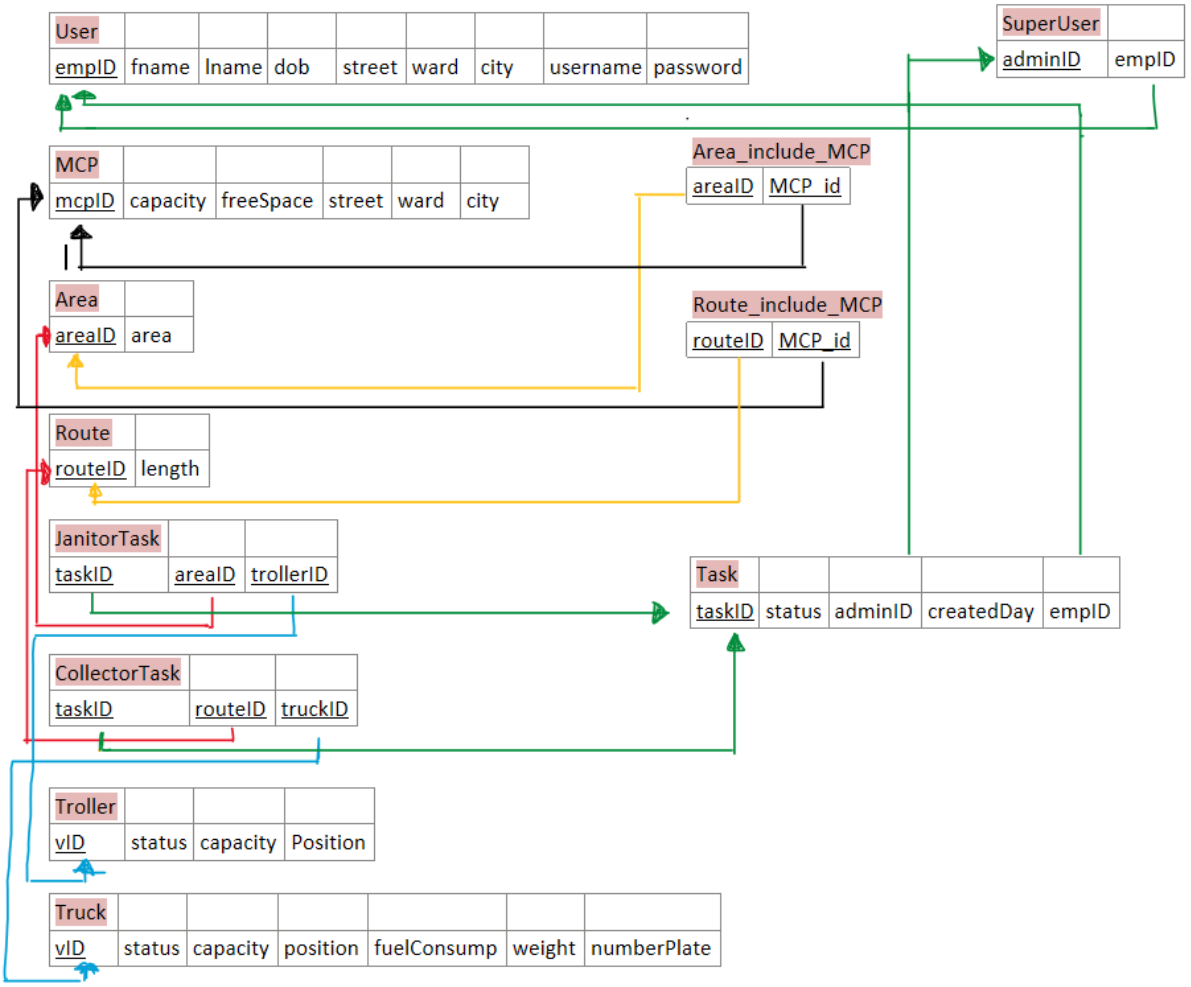
Our available functions on the Presentation Layer include viewing maps and calendars, assigning tasks to collectors and janitors, calculating the best route, monitoring the current employees', MCPs', and vehicles' status, and establishing communication between employees. Hence, the Application Logic Layer contains the implementations including Task assignment, Employee management, Vehicle management, MCP management, Supported API (map, routing algorithm, calendar), Communication handling (SMS, notifications), and Report generation. For example, if a Back Officer wants to return the monitoring vehicles' status, the Presentation Layer will request the API Vehicle management from the Application Logic Layer, and this interface will implement the functions display_vehicles(). These functions will retrieve data fields (vehicle ID, position, available status, weight, capacity, and fuel consumption) from the Database Layer and return the desired values to the Application Logic Layer, and eventually, the Back Officer can view the current vehicles list on the Vehicle page on the website. Moreover, a collector or a janitor logging in to the system can view the calendar or map on the main page thanks to the API provided by the Google calendar which is embedded into Application Logic Layer without accessing to Database Layer or its own implementation.

The system will be hosted on Google Workspace using API includes with this guide: Deploy a website with Cloud Run (google.com)

- Google Maps API (maps and routing algorithm)

3. Database layer

Based on the data requirement defined in previous phases, it is noticeable that the system primarily stores data of traditional form, and hence all of them can be managed by a relational database system. In particular, UWC2.0 will records and manages involving data using RDBMS - Oracle.

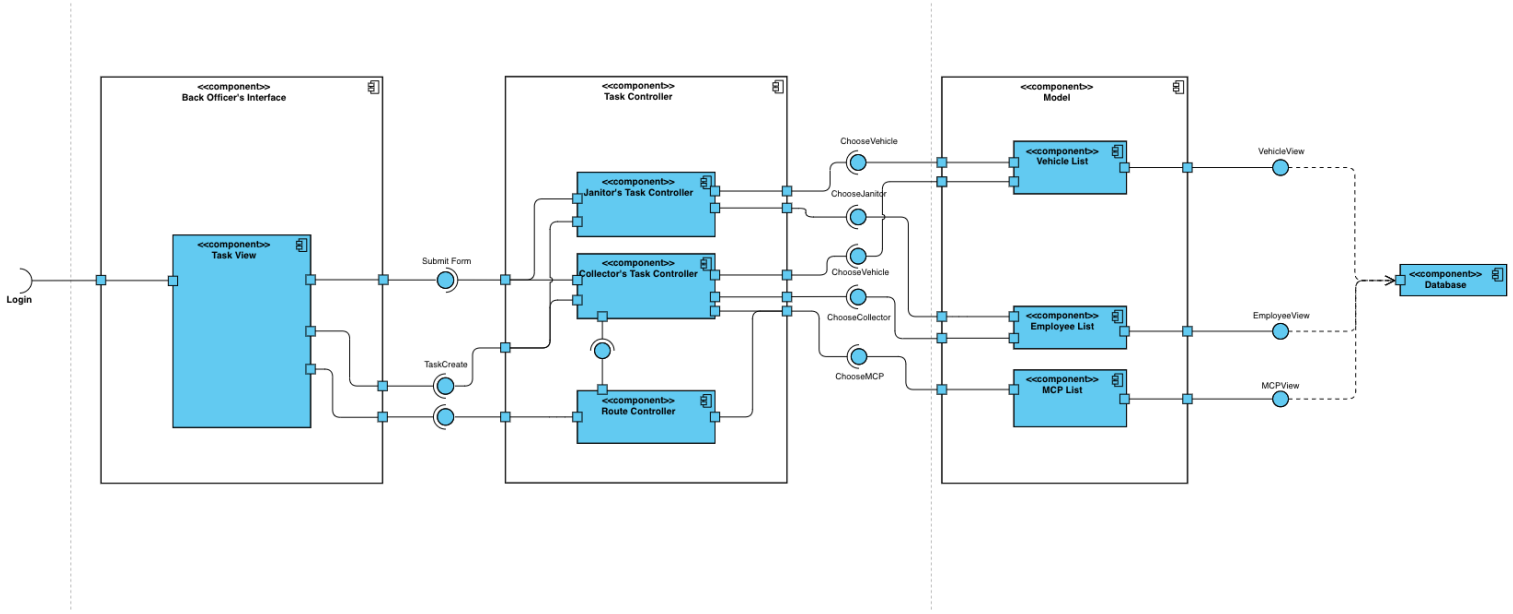


The given mapping[1] depicts how data is organized and relates to each other in the database. In the whole database, there is only one surrogate key, taskID, that is created for the sake of task management in the database as well as aiding quick query for tasks. Another point requiring more explanation is the existential decision for Area_include_MCP and Route_include_MCP tables. By its nature, a relational database has limitations in storing multi-valued attributes, which is exactly the case for

[1] The mapping is generated from the EER diagram https://lucid.app/lucidchart/c8185c69-9d9a-4f03-bdc9-0bc61f4d4e35/edit?viewport_loc=-1970%2C-565%2C5980%2C2616%2CGZeIYp~cvkot&invitationId=inv_fe8df7bf-de95-440f-a273-06c1113965bd

Area and Route. An area relation object is completely defined with at least three MCPs while a route requires at least two points. Both of them resemble the need for multiple defining points; that makes us end up with the current solution - creating a new relation just dedicated for storing the MCP included in an area or a route.

II. Component diagram for Task Assignment module



- The <Task View> component has its own interface where Back Officers can select to create a task manually through a form that has two separated interfaces, one for the janitors and the other for the collectors.
- Information such as vehicles, employees, MCPs, …. can be sent to the <Task Controller> component in which Back Officers can select and assign tasks.
- The <Janitor's Task Controller> component can deal with actions: assign vehicles and janitors from the back officer in the assignment task module.
- The <Collector's Task Controller> component can deal with actions: assign vehicles, collectors and MCPs from the back officer in the assignment task module.
- The chosen MCPs are then sent to the <Route Controller> component from the <Collector's Task Controller> component.
- At the <Model> component, it can store, manage and update data to the database of:
    + The management information of all managed and assigned elements(vehicle list,employee list, MCP list).
    + Messages sent between employees.

+ The information tasks of assigned tasks correspond to the employees.
+ The notify for collectors and janitors when MCPs are full.

References

1. Nguyen Van Thanh. Optimal Waste-to-Energy Strategy Assisted by Fuzzy MCDM Model for Sustainable Solid Waste Management. Available online: https://www.mdpi.com/2071-1050/14/11/6565#B1-sustainability-14-06565.

2. Hanoi Department of Construction. Progress Report of SWM Facility Development in Hanoi; Hanoi Department of Construction: Ha Noi, Vietnam, 2019.

3. Ngo Thi Lan Phuong, Helmut Yabar, Takeshi Mizunoya. Characterization and Analysis of Household Solid Waste Composition to Identify the Optimal Waste Management Method: A Case Study in Hanoi City, Vietnam.

4. Goran Boskovic, Nebojsa Jovicic, Sasa Jovanovic, Vladimir Simovic. Calculating the costs of waste collection: A methodological proposal. Available online: Calculating the costs of waste collection: A methodological proposal - Goran Boskovic, Nebojsa Jovicic, Sasa Jovanovic, Vladimir Simovic, 2016 (sagepub.com)