HO CHI MINH NATIONAL UNIVERSITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Software Engineering

Urban waste collection aid - UWC 2.0

Students:

Hứa Hoàng Nguyên	2052619
Vương Đức Hiếu	2011196
Doãn Hoàng Thiên	2053450
Lê Nguyễn Nam Hải	2012499
Lê Hoàng Thụy	1952130

HO CHI MINH CITY, 9/2022

Contents

1 Project Context

1.1 Stakeholder

External stakeholder

- **The project team:** The developper team of Organization X including developpers, engineers, digital designers... and the other related to collecting data.
- **The project manager:** The manager of developper team, generalinchieves of related departments, the chairman of organization X.
- **The sponsor:** Shareholders, senior managers of organization X as well as of Service provider Y, perhaps some government departments that is related to environment maintainance.
- **Internal teams:** The developper team who can continue to develop to make new version of project in the future, perhaps the sale team who will advertise to market if the Service provider Y accepts.

Internal stakeholder

The main external stakeholder is Service provider Y who will directly use the project; the government who may check the data for some purposes, and some organizations are allowed to research and develop the project.

1.2 Current Problem

Urban waste management is one of several significant problems faced by many countries in the world and thus considered one of the important points to be improved in Sustainable Development Goal (SDG) 11: sustainable cities and communities and SDG 6: clean water and sanitation. Particular attention is given to developing countries that continue to prioritize development and economic growth. In urban context, solid waste management is costly and ineffective. Improvement of waste collection and management is emphasized by governments and organizations for positive impacts on cities, societies, and environments. Therefore, a great and satisfied management system is especially necessary. Besides, the friendly and clear interface as well as exact data management, fast and stable user connection are also important requirements.

1.3 Goals

Upgrading the system to version 2.0, the stakeholders can looking forward to following certain goals:

- The UWC 2.0 will be inter-operable with the UWC 1.0 as much as possible
- 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.
- UWC 2.0 system interfaces should be clear and friendly with users, with an opportunity to switch to English in the future.

Beneficiary

Aside from the stakeholders, the beneficiary also gain a lot of benefits from this upgrade, for example:

- Service provider Y with expectation that can improve efficiency of garbage collection.
- Besides, the more efficient garbage collection, the better life of citizens in the urban.
- The government, other environment organization,...
- Some developpers can use this project as reference to develop better management system in the future.

Duration

- The project is expected to be complete as soon as possible.
- Some early versions may be introduced during the project for testing, repairing or adding more features if neccesary.
- Garbage collection as well as protecting environment is one of the most inportant mission to maintain our lives. Therefore, it is our honor to contribute to improving the efficiency of garbage collection in particular and the environment of urban in general.

2 Project Requirement

2.1 Functional Requirement

- Have an overview of janitors and collectors, their work calendar
- Have an overview of vehicles and their technical details (weight, capacity, fuel consumptions, etc)
- Have an overview of all MCPs and information about their capacity.
- Assign vehicles to janitors and collectors

- Assign janitors and collectors to MCPs (task)
- Create a route for each collector. Assigned route is optimized in term of fuel consumption and travel distance.
- Be able to send message to collectors and janitors
- Have an overview of their work calendar
- Have a detail view of their task on a daily and weekly basis. All important information should be displayed in one view (without scrolling down).
- Be able to communicate with collectors, other janitors, and back officers.
- Check in / check out task every day
- Be notified about the MCPs if they are fully loaded

2.2 Non-functional Requirement

- Information should be updated from MCPs every 15 minutes with the availability of at least 95% of their operating time.
- The messages should be communicated in a real-time manner with delay less than 1 second.
- There exists a current system UWC 1.0 with a database.
- UWC 2.0 is expected to import and to use the existing data from UWC1.0. It is expected that the Task Management to be inter-operable with the UWC 1.0 as much as possible.
- 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.
- UWC 2.0 system interfaces should be in Vietnamese, with an opportunity to switch to English in the future.

2.3 Use-case Diagram for the system

Actors:

There are 3 actors in the system: back officers, janitors, and collectors

Use cases:

- Have technical details of vehicles
- Accessible to janitors and collectors' information
- Manage MCPs
- Regulate the workflow: create route - assign tasks - assign vehicles
- Check in/ Check out
- Communicate/ Exchange messages
- Notification
- View work schedule

Relationships:

- Regulate workflow «generalization» create route, assign task, assign vehicles.
- Regulate workflow «include» view work schedule.
- Exchange messages «extend» Notification.

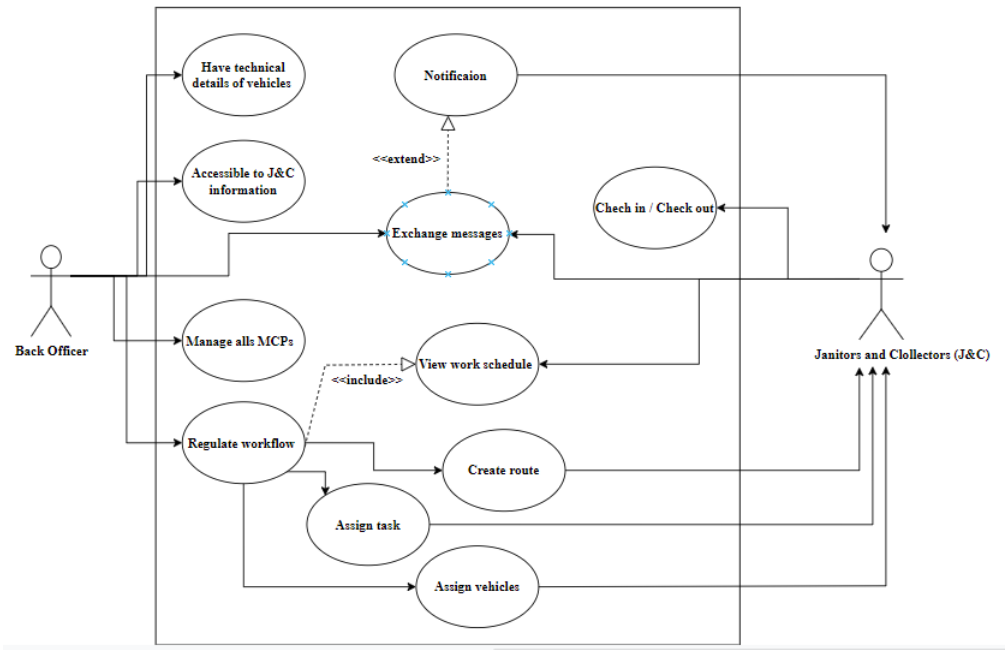

Figure 1: Use-case Diagram for the system

2.4 Task Assignment Module

Use-case Diagram

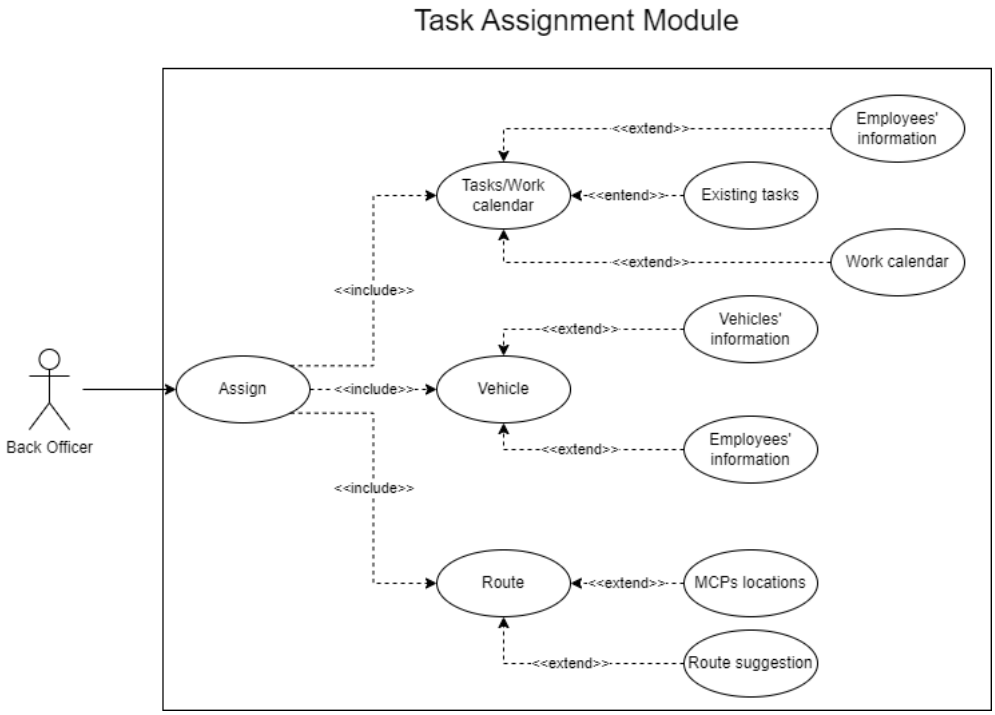

Figure 2: Use-case Diagram for Task Assignment Module

Use-case Diagram Description Table

Name	Task Assignment Module
Actor	Back Officer
Description	The module assists back officers in assigning tasks and work calendar among teams of janitors and collectors. It also helps with assigning vehicles and their routes to teams of collectors and janitors.
Preconditions	1. User's identification has been authenticated 2. User is authorized to use the module
Postconditions	1. The calendar, tasks, routes, and vehicles are stored in the system database 2. The information above is sent to the right collectors and janitors
Normal flow	1. The back officer will open the module 2. The information about collectors, janitors, tasks, vehicles, routes and MCPs will appear 3. Assigning tasks to collectors and janitors. 4. Assigning vehicles to collectors and janitors. 5. Automatically suggesting routes for each team and asking for confirmation from back officer. 6. Checking for confirmation and saving that information to the database
Exceptions	Human error: Some teams may have the same work
Alternative flow	Asking the back officer to re-assign the task

3 System Modelling

3.1 Activity Diagram capturing the business process between systems and the stakeholders in Task Assignment module

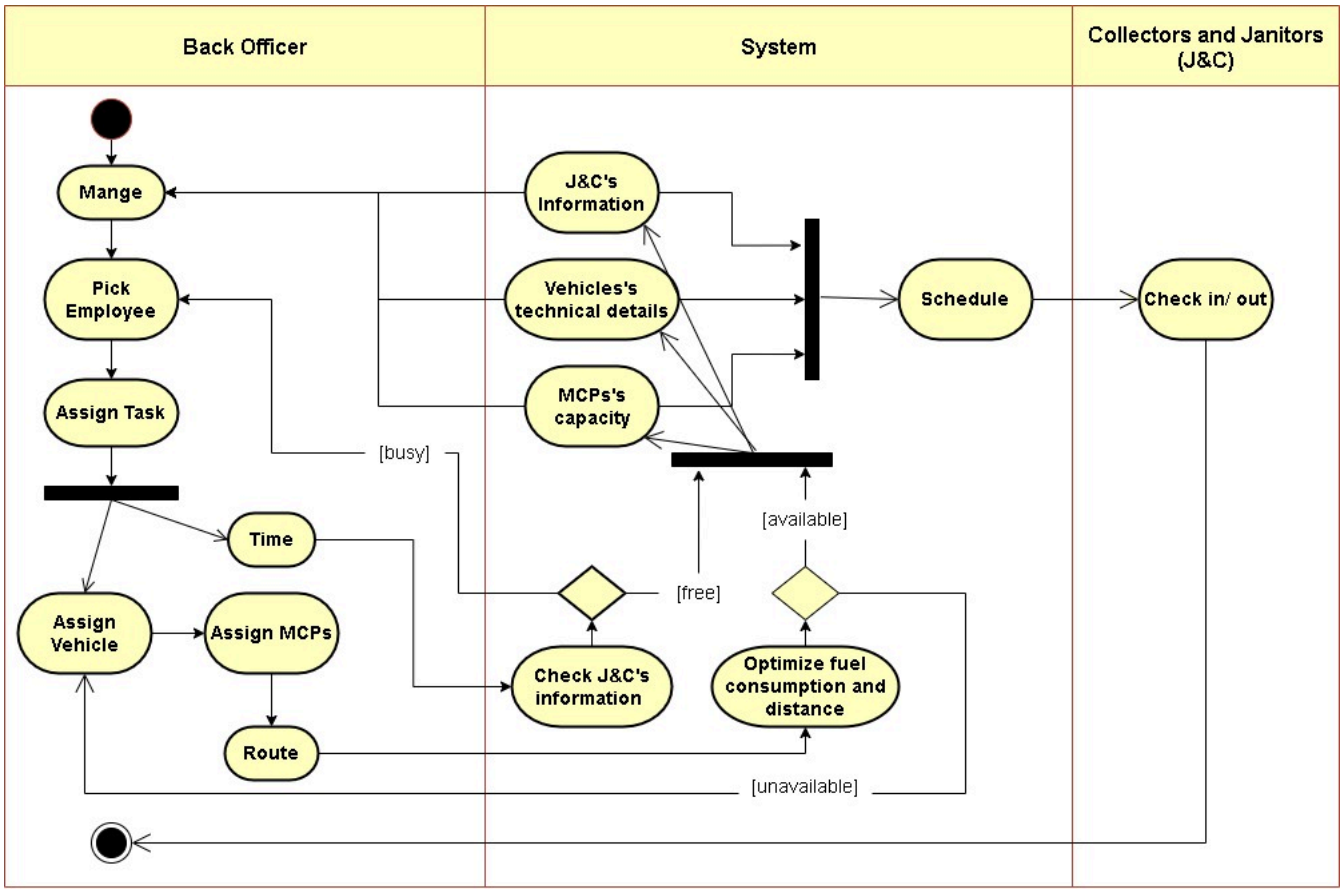

Figure 3: Activity Diagram

3.2 Conceptual Solution for Route Planning Task

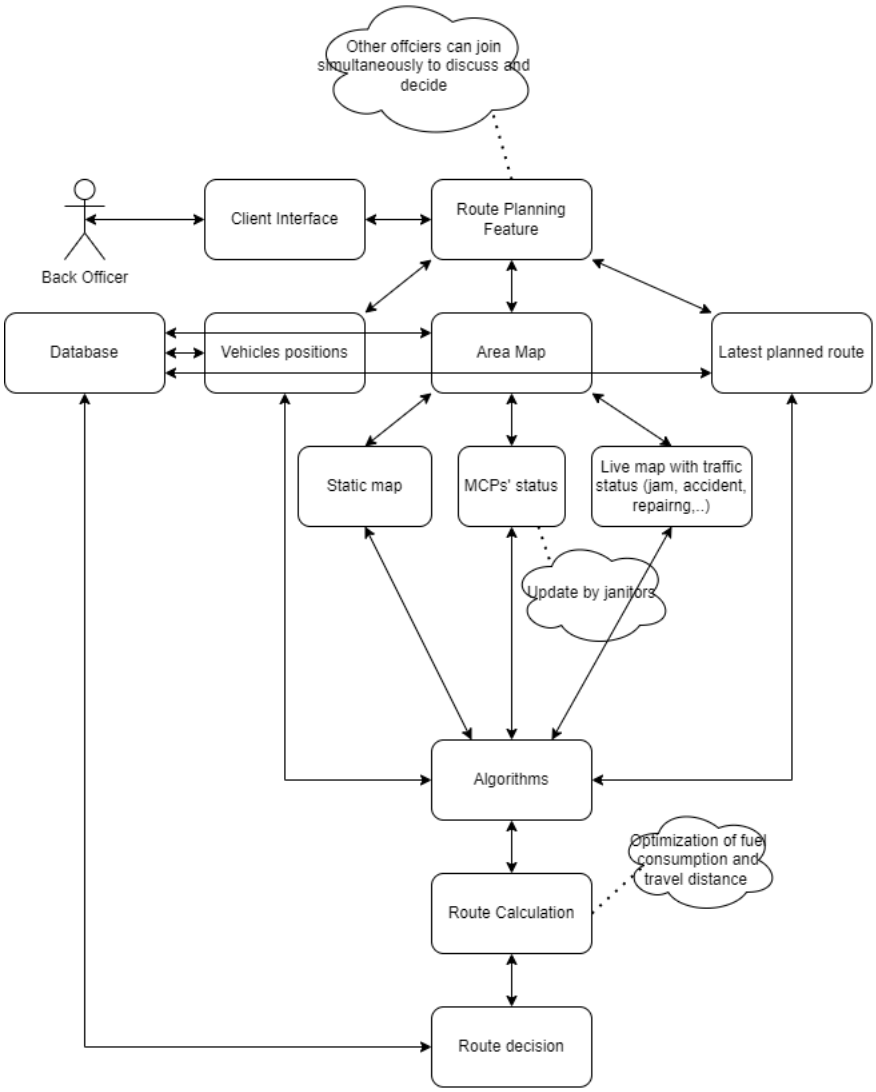

Figure 4: Conceptual Solution for Route Planning Task

3.3 Sequence Diagram for Conceptual Solution

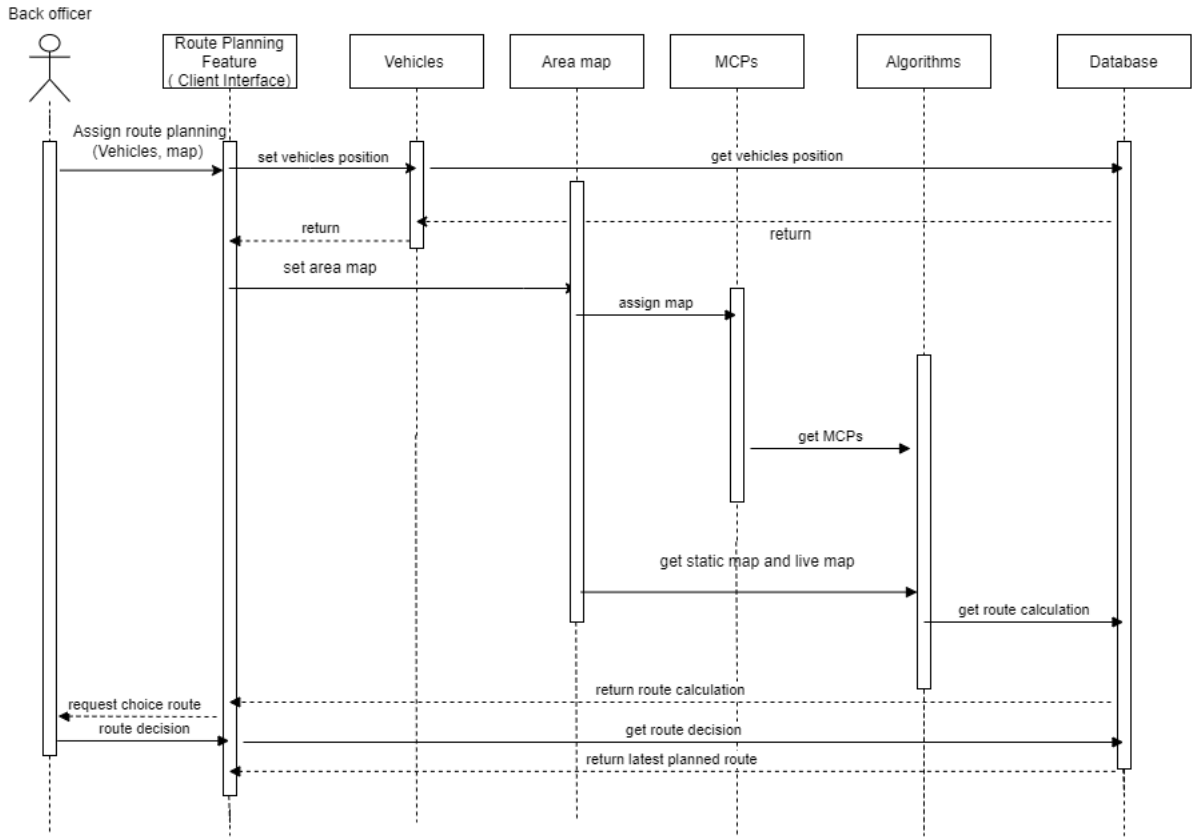

Figure 5: Sequence Diagram for Conceptual Solution

3.4 Class Diagram Of Task Assignment Module

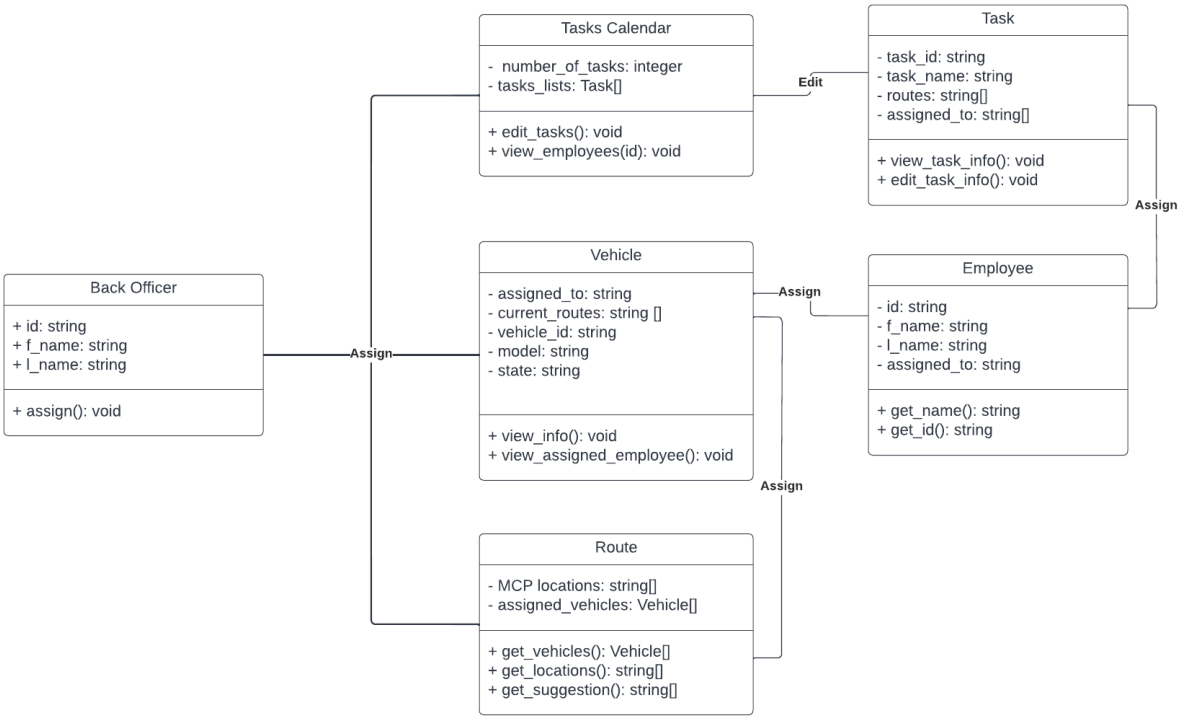

Figure 6: Class Diagram of Task Assignment Module

4 Architecture Design

4.1 Architectural Approach

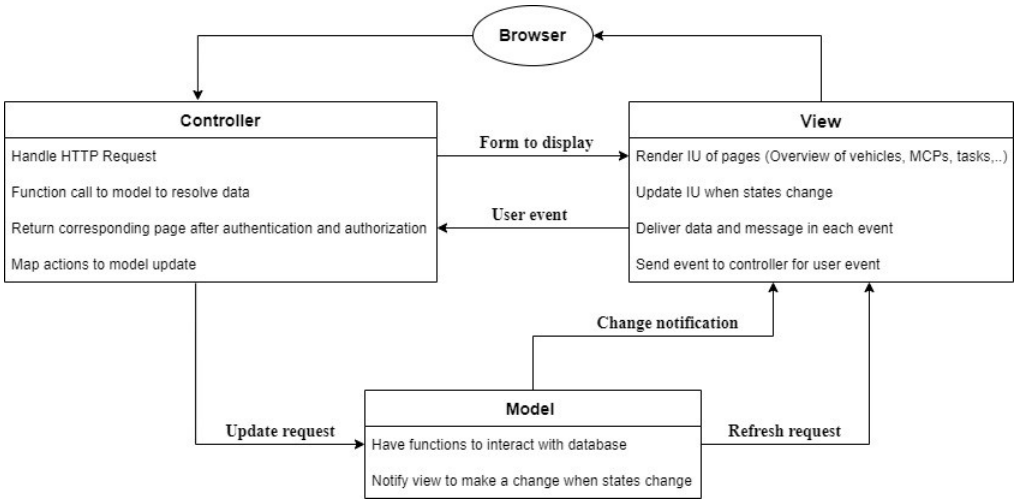

Figure 7: MVC Diagram

Describe	Detailed MVC architecture
Model	- Provide methods to handle API calls to perform authentication anytime employee check in/ check out - Provide methods to interact with the database to perform registration, login, and employee account management. - Provide methods to perform database operations (add, delete, edit) with work-scheduling, choosing vehicles, employees, MCPs, and route. - When there is a change in the model state about data, send notification to view to refresh and update the change to system.
View	- Render the user interface of different pages: + Login page + Home page + Map + User profile + Chat Box - Interact with users by automatically handling events in the view: + Automatic slider + Display animation when exchange message and when employee assign or complete given task

	- Send the user's choice and change events to the controller when manipulating web page elements. - Pass data into the requests sent to the controller to handle the application logic: + Transmit data about username, phone number, password, user ID to the controller + Transmit data about information about personal information, vehicles technical details, MCPs information, work status, schedule and update data constantly. - When notified of state changes from the model, perform data retrieval from the model to refresh the view and update the new view.
Controller	- Return a view base on the requested page and the former manipulation - Return response to user with corresponding request - Call functions from the model to perform operations on the data: + Update overview of employee, vehicles, MCPs + Update schedule and create the most appropriate route - Call functions from model to utilize API from external system to update data from google map, chat box,…
Strength	- Clear division between model, view, and controller make the work of code implementation more convenient - By constructing and applying model, view, controller separately make it easy to extend the code - Easily add requirements for interacting with the data independently of the display on the user side.
Difficulty	- It is very complex when we implement code because there are too much condition and relation between each element - There need to ensure an effective interaction and flexible operation between Model, View, and Controller

4.2 Modules

For this system, we are planning to implement 4 main modules which are **Task Management Module**, **Task Assignment Module**, **Authentication Module** and **Communication Module**.

4.2.1 Task Management Module

For back officers:

- Display employee module (information, work calendar)
- Display vehicle module
- Display MCP module

For Janitors, Collectors:

- Get Information module (work calendar, etc)

For task management, there are 2 separate sets of modules depending on the role of the user. With back officers, they need information about 3 primary information: employee, vehicles and finally MCP. Assuming that the users have gone through the process of authentication and authorization successfully with their corresponding role.

For Display Employee Module:

- **Input:** a simple click event on a button on the screen
- **Output:** display all information about the employee as well as their work calendar

For Display Vehicle Module:

- **Input:** a simple click event on a button on the screen, but we need a specific vehicle ID if we want to display that vehicle only otherwise display all vehicle information
- **Output:** display the technical details of the specified vehicle their current assigned tasks or list of all vehicles

For Display MCP Module:

- **Input:** a simple click event on a button on the screen
- **Output:** display all MCP locations with optimization calculated as well as the assigned task for that location

For Get Information Module:

- **Input:** a simple event and an employee_id since this module only return the information for one employee only.
- **Output:** Information about the employee task in that day and related information such as MCP locations, vehicles

4.2.2 Task Assignment module

This module can only be accessed by back officers and can not be accessed by anyone else. This is the tool for back officers to assign tasks for janitors and collector. It has 3 functions.

For assigning work calendar and specifying tasks:

- **Input:** click events from users, choosing a collector/janitor, choosing the right tasks for each individual, confirming the choice.
- **Output:** Saving the choices of users into the database.

For assigning vehicles to collectors and janitors:

- **Input:** click events from users, choosing a collector/janitor, choosing a vehicle, confirming the choice.
- **Output:** Saving the choices of users into the database.

For assigning route for each vehicle:

- **Input:** click events from users, choosing a vehicle, choosing among the designated choices, confirming the choice.
- **Output:** Saving the choices of users into the database.

For notification alert:

- **Input:** None, will be called when making changes affected the employees.
- **Output:** Sending a message to janitors/collectors affected by the change.

4.2.3 Authentication Module

This module consist of 4 main functions: **login**, **register**, **logout** and **returnView**. **For Login Function:**

- **Input:** username and password pass from the form in the UI
- **Output:** the state of the login process such as invalid password or username or successfully login

For Logout Function:

- **Input:** simple click event on a button on the screen
- **Output:** Logout the user from the system

For Logout Function:

- **Input:** simple click event on a button on the screen
- **Output:** Logout the user from the system

For Register Function:

- **Input:** username and password pass from the form in the UI
- **Output:** the state of the register process such as register successfully or invalid username (duplicate)

For returnView Function:

- **Input:** id of user after authentication from the database
- **Output:** Corresponding dashboard view

4.2.4 Communication Module:

This module is crucial since it allows janitors and collectors to feedback with the back officers with low network latency. It consists of a special UI for real-time chat, each user will have their own conversation with the back officer. There is 1 sub modules for this module: **sendMessage**
For sendMessage Function:

- **Input:** the message user want to send and the receiver back officer id
- **Output:** the state of message such as: Sent or Error

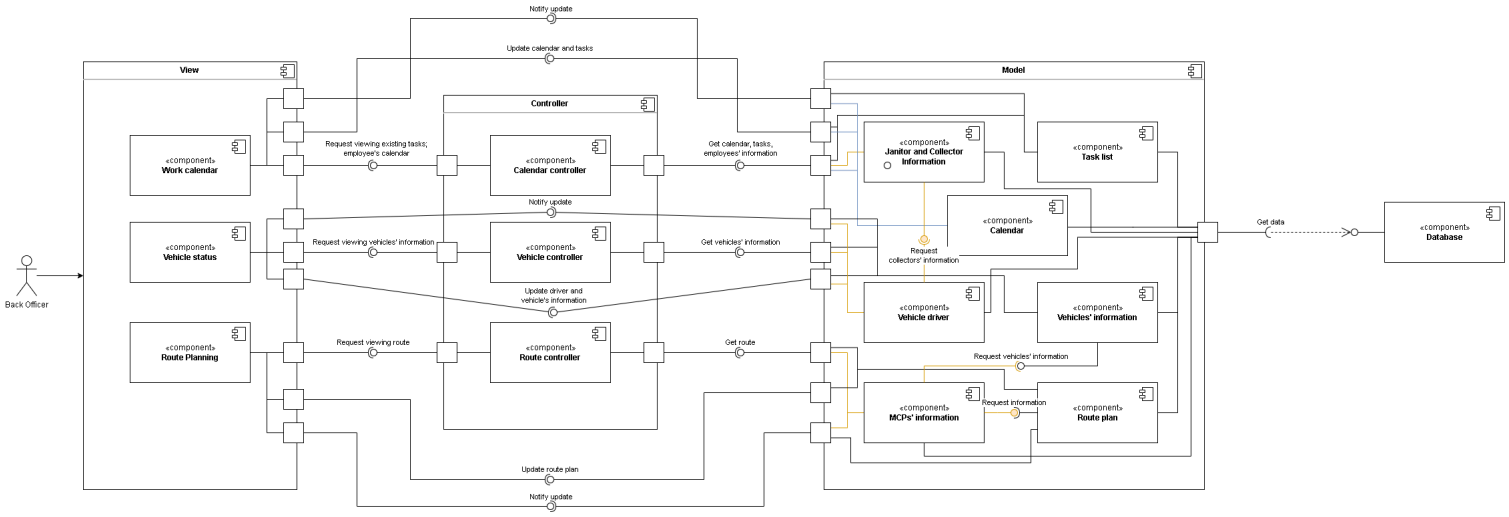

Figure 8: Implementation diagram for Task Assignment module

4.3 Implementation diagram for Task Assignment module

Description:

System has 3 main components:

1. View component:
    - Containing sub view components
    - View components:
        - Render UI of page
        - Update UI pages when model changed
        - Send data to controller
    - View components provide the use of interfaces from controller to execute requests
    - View components use interfaces from the model to perform data updates and provide interfaces for the model to notify about data updates.
2. Controller component:
    - Containing sub controller components
    - Controller component:
        - Handle HTTP request
        - Have function calls to model for data validation (Login/Register)

  - Map actions to model update
- Controller components provide interfaces for the view to execute requests
- Controller components use interfaces from the model to perform operations with the database.

3. Model component:
   - Containing sub model components
   - Model component:
     - Have functions to interact with database
     - Notify view to make a change when state change
   - Component models provide interfaces for controllers working with the database
   - Providing interfaces for viewing can update data when receiving from notifications
   - The component model also uses interfaces from database components
   - Component models provide and use interfaces of each other

Additionally: Database component:

- Direct operation with the database.
- Providing interfaces for the model to perform direct access on the database