ĐẠI HỌC QUỐC GIA

ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH

···☼···

SOFTWARE ENGINEERING

Assignment

Urban waste collection aid – UWC 2.0

Lecturer: Prof. Quan Thanh Tho

Student: Nguyen Thanh Duy An - 2052858
Vo Trung Kien - 2053163
Vu Viet Hung - 2053081
Vo Duy Hung - 2053625
Nguyen Hiep - 2052999

Ho Chi Minh City, November 2022

CONTENT

TASK 1: Requirement elicitation

1.1 Introduction

Nowadays, many countries focus on development and economic growth, which leads to an increase in solid waste and makes the management of solid waste cost more and ineffective. So the most important point is to improve the Sustainable Development Goal (SDG). This project is created to simplify waste management and make it easier for stakeholders to track the waste collection routine. With the improvement in UWC 1.0, the project aims to upgrade this system, using their own database to create the UWC 2.0 with more functionalities and handle the large number of collecting points.

Recently, waste collection companies only use paper to store their work and don't have a particular system to manage the waste collection. For example, employees have to come to their workplace to receive the task and check in or check out. The manager has to have a secretary to write down the work and doesn't have a particular system to communicate with the staff. So the program will help all the waste collection systems work smoothly by only using the web page. It helps improve communication between staff and provide better management on waste collection.

About the relevant stakeholders:

Stakeholders	Role
Back officers	Operate a central system to create working routine, coordinate front collectors and janitors
Janitors	Collect the garbage from Major Collecting points (MCPs)
Collectors	Drive different type of vehicles to MCPs

About the stakeholders current needs:

Stakeholders	Current needs
Back officers	-Information about the janitors and collectors -Information about the vehicles and technical details -Information about the MCPs capacity -Contact with janitors and collectors to give information about their task or if there are any changes in there routine
Janitors and collectors	-Information about the working routine -Information about their tasks on a daily and weekly basic -Contact with others employees to know about their routine -Contact with Back officers to have information about their schedule if there are any problems

About the stakeholders current problems:

Stakeholders	Current problems
Back officers	-Cannot control the schedule of the janitors and collectors -Cannot communicate with the janitors and collectors -Do not know the updated overview of an the capacity of the MCPs -Have to calculate the best routine to collect the trash by hand
Janitors and collectors	-Do not know the schedule and the tasks on a daily/weekly basis -Cannot communicate with the back officers and other janitors and collectors -Do not have the system to check in and check out -Do not get notification when there is any changes in tasks and schedule

About the benefits UWC 2.0 will be for each stakeholders

Stakeholders	Benefits
Back officers	-Can have the work schedule of janitors and collectors -Can have the update information and capacity of MCPs -Easy way to communicate with janitors and collectors about changes in their schedule -Create the optimize routine to collect the waste
Janitors and collectors	-Have an overview of their schedule -Know their daily and weekly task -Receive a message from back officers and others janitors, collectors -Can check attendance -Be notified if the MCPs is fully loaded

1.2 Functional and Non-functional requirements

FUNCTIONAL REQUIREMENT:

Functional requirements for applications describe what specifically needs to be implemented in a particular system or product and what actions users have to take to interact with the software. They determine what the system should do. In other words, **functional requirements** are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions.

Back officers:

Number	**Functional requirements of UWC 2.0**
1	Interface gives information about janitors and collectors
2	Interface gives information about janitors and collectors 'work calendar

3	Interface gives information about vehicles and their technical details (weight, capacity, fuel consumptions,…)
4	Back officers can check real-time MCPs and information about their capacity(updated from MCPs every 15 minutes with the availability of at least 95% of their operating time)
5	Back officers can assign vehicles to janitors and collectors, janitors and collectors to MCPs
6	Back officers can create a route for each collector(the route which is optimized in term of fuel consumption and travel distance)
7	Facilitating contact information between back officers and collectors and janitors

Collectors and janitors:

Number	**Functional requirement of UWC 2.0**
1	Interface gives information about janitors and collectors' working calendar
2	Collectors and janitors a detail view of their task on a daily and weekly basic
3	Collectors and janitors check in / check out task every day
4	Collectors and janitors are notified about the MCPs if they are fully loaded

NON-FUNCTIONAL REQUIREMENT:

As a rule, the non-functional requirements primarily include various product quality attributes determining system quality features, most often as examples below:

Availability: Ensure the overall efficiency of the UWC web from 7am to 6pm(When lots of people access the system).

Performance: Web's performance delay within 2 seconds, the data shared among users within 3 seconds, web response to multiple requests and manipulations within 1 second, capable of updating MCPs concurrently.

Reliability: App behavior in case of cyber attack (automatic shut down within 10 seconds, immediately send notification to administrator and operation recovery)

Ease of use: within 4 operations of a function, furthermore, employees can easily operate the web after 20 minute training.

Size: All file load to user equipment must be approximately maximum 300MB.

Scalability: Expand the database and avoid adversely affected performance (UWC 2.0 is expected to import and use the existing data from UWC 1.0), to be more specific, the database can store the information of at least 100 employees.

Robustness: The time for restarting the system below 1 second when data corrupted.

Security: Delete the data of administrator accounts after 5 attempts, warning if other IP tries to break the firewall.

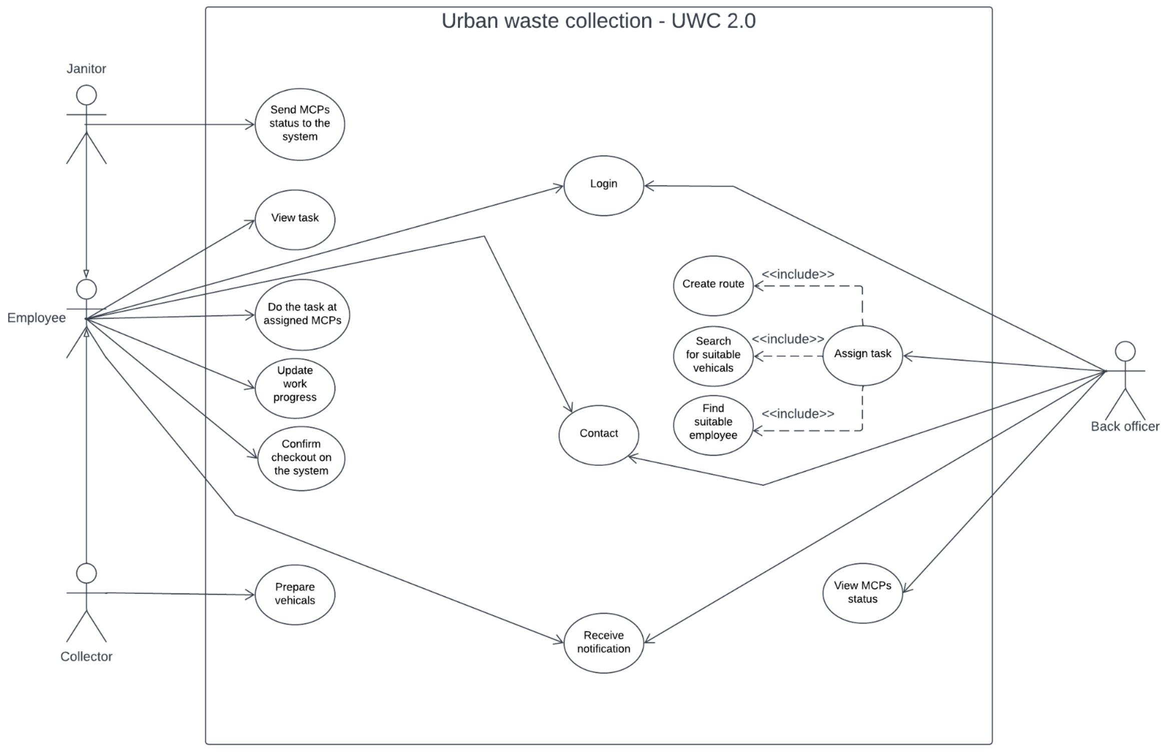
Usability: UWC 2.0 system interfaces should be in Vietnamese, with an opportunity to switch to English in the future. All important information should be displayed in one view (without scrolling down).

Extensibility: Task Management to be interoperable 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

Multi-platform: Can be used efficiently in multiple operating systems of mobile phone(Android, IOS), personal computer or laptop(Windows, Linux, Mac) and multiple browsers(Chrome, Firefox, Opera,...)

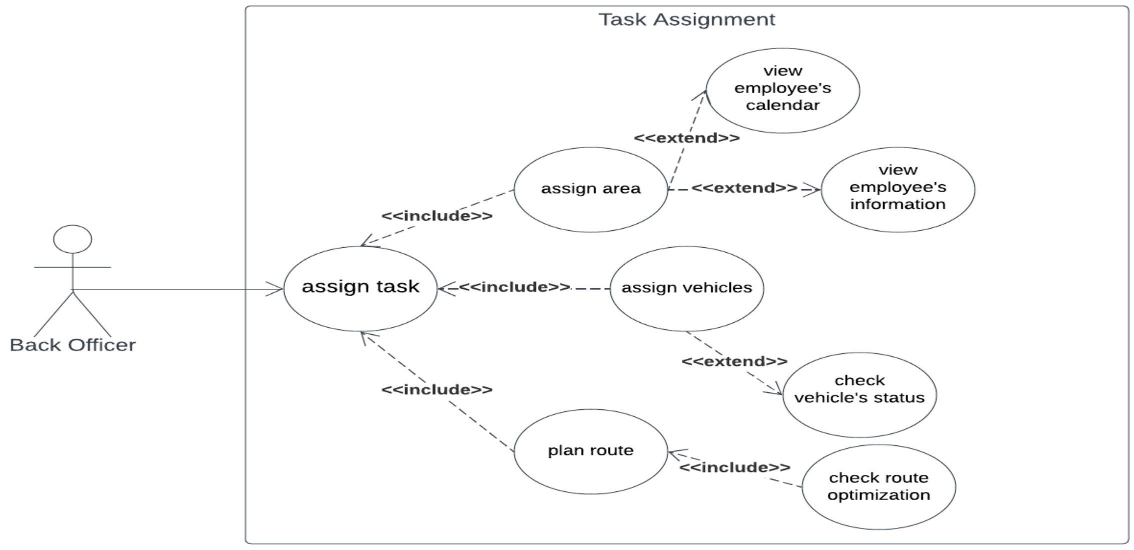
Maintainability: Clean structure and architecture, which is easily maintained or updated to new versions after long-time operation.

Use-case diagram for the whole system:



1.3 Task assignment module

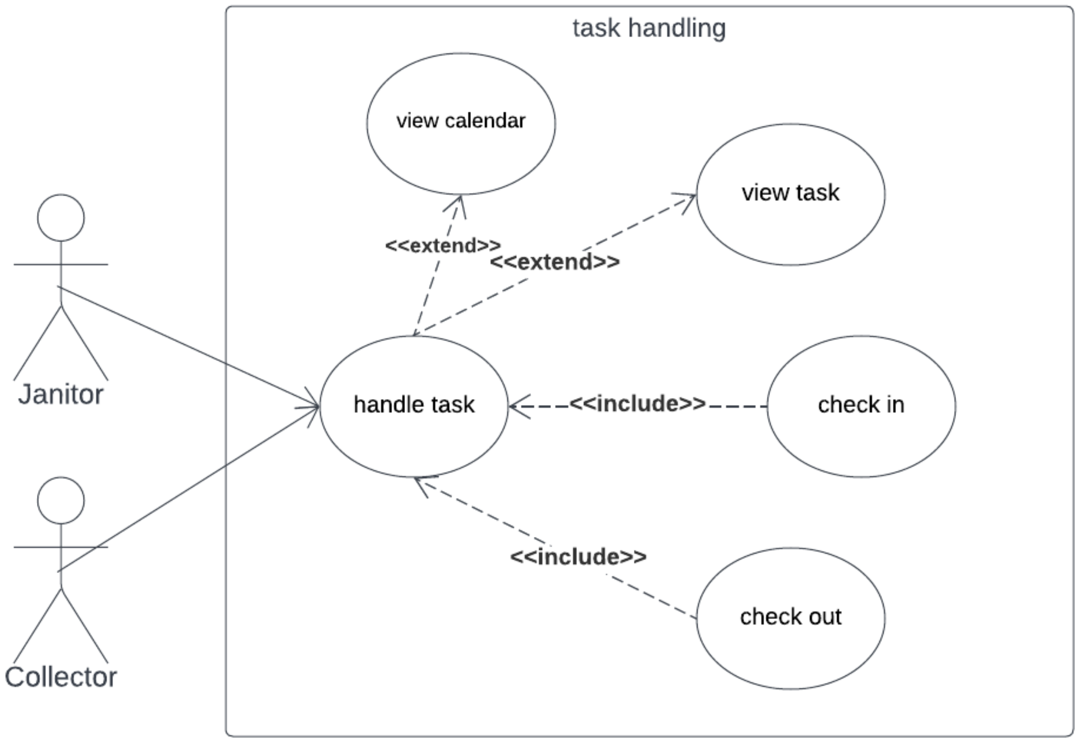
1.3.1, Task assignment module



Use-case name	Task assignment
Actor	Back officer
Description	Back officer can use this module for assigning task, work must include three parts: assign area for janitors, assign vehicles for the collector and create optimized routes. Then the system will store the task in the database and send notification to the related employees.
Trigger	The back officer indicates that he wants to assign tasks.
Preconditions	PRE-1. Back officer must log in to the system. PRE-2. Database of the system is online and connected.
Postconditions	POST-1. Task information is successfully stored in the database. POST-2. System notify about the task to related employees.
Normal flow	1. Back officer chooses the task assignment in the sidebar. 2. System lists all areas needed to assign. 3. Back officer assigns employees to each area. 4. System shows a list of vehicles for collecting. 5. Back officer assigns the collector to the vehicle. 6. Back officer assigns a vehicle to collect a MCP. 7. System then checks the optimization of the route. 8. System stores tasks in the database and sends notification to employees.
Alternative flows	Groups of steps 2 and 3 (for area assignment); steps 4 and 5 (for vehicle assignment); steps 6, 7 and 8 (for route planning) can be swapped in any order.

Exception	E-1. Employee has the same calendar for different tasks. E-2. The planned route is not optimized.

1.3.2, Task handling module:

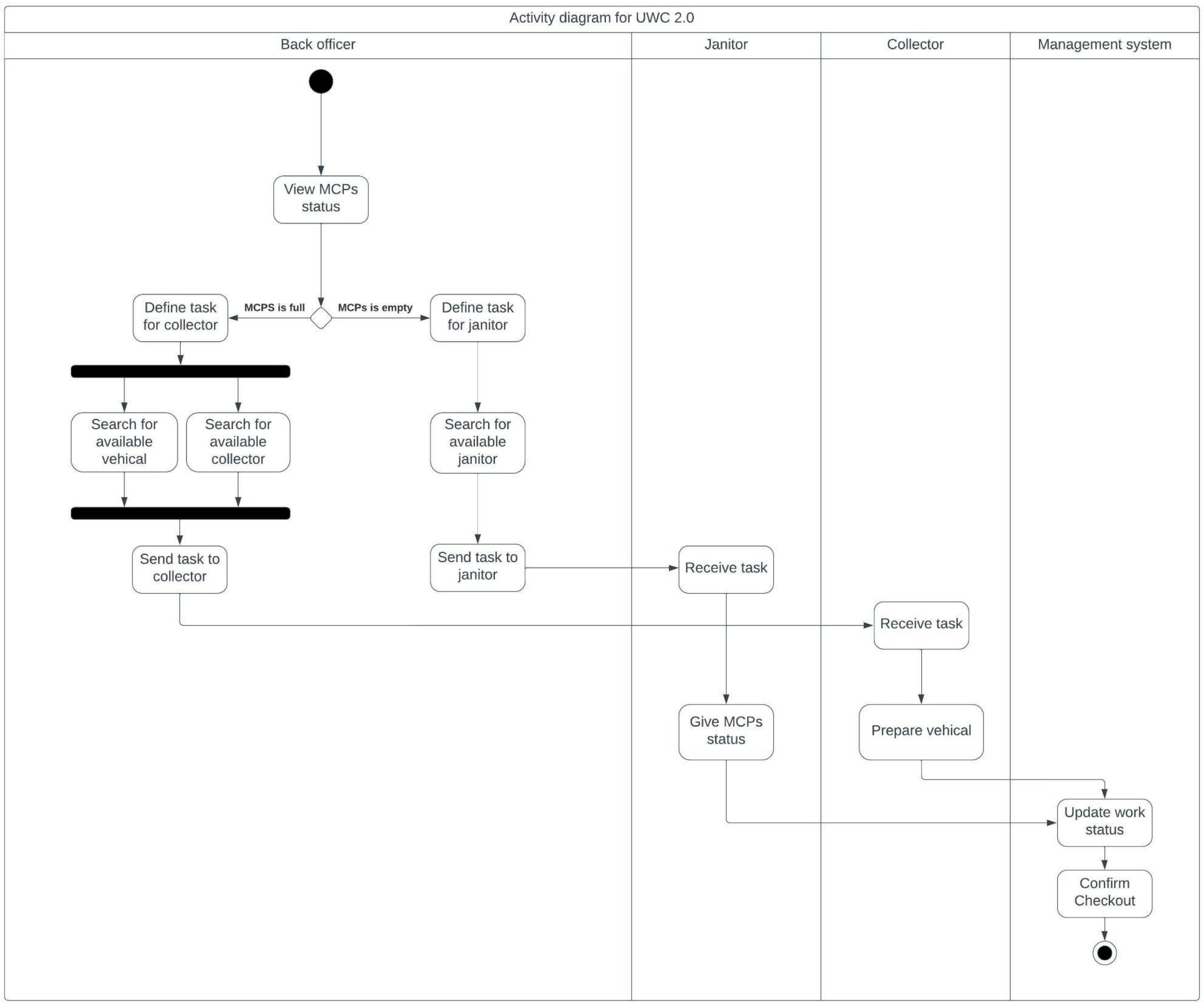


Use-case name	Task handling
Actor	Janitor, collector
Description	The janitors and collectors handle their task by using this module. This module can help employees view information about the task, view the calendar for themselves. The system can record their task status and store it in the database.
Trigger	The employee checks in on the task.
Preconditions	PRE-1. Employee's identity has been authenticated. PRE-2. Employees are authorized to the task.
Postconditions	POST-1. The employee checks out the task. POST-2. The system records the result into the database.

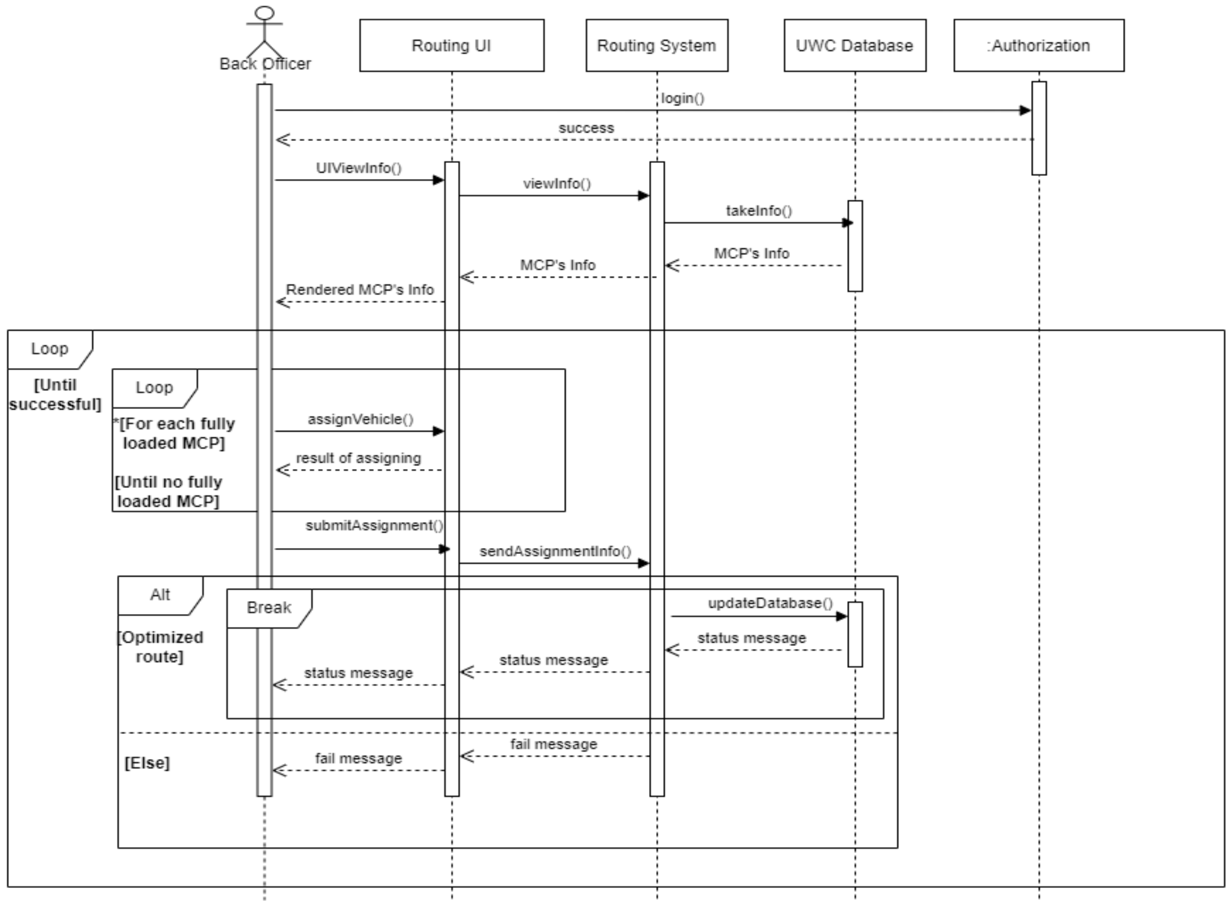
Normal flow	1. The employees check in on the task 2. System shows the information about the task (area, vehicles, …) 3. Employees can view their calendar. 4. The employees can check out the task after finishing.
Exceptions	E-1. The janitors and collectors may forget to check out the task but they actually finish.

TASK 2: System modelling

2.1 Activity Diagram



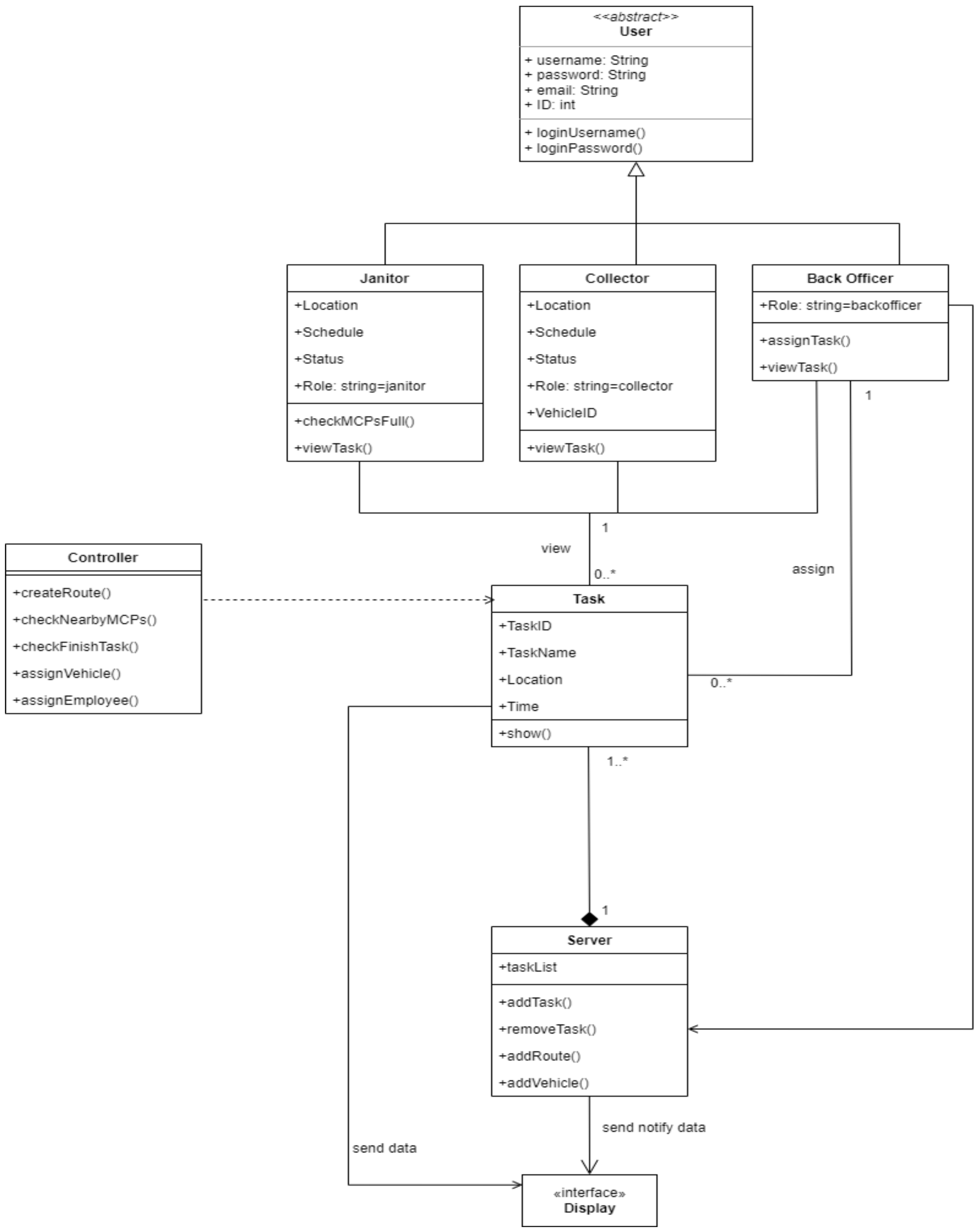
2.2 Sequence Diagram



Description:

- In this operation, back officers will plan routes through a user interface (UI) to send a request to the routing system and receive details about MCPs, vehicles, fully loaded MCPs, and available vehicles.
- The routing system will make the UI show the list of fully loaded MCPs, with information about the time when it was full, location to the back officers.
- Back officers then click on one MCP in the interface to assign one vehicle to one MCP.
- Loop the previous step until there is no fully loaded MCP.
- Then, the back officer will click the ‘send’ button in the interface to send the assigning request to the routing system.
- Routing system then will check if the conditions of the optimized route are satisfied. If successful, send the assignment data to the database. If not, repeate the assigning step.
- After a successful or failed storage and planning route, the routing system sends a status message to the back officer according to the UI.

2.3 Class diagram



Description:
-User (interface): has a function to log in by using a username and password, containing basic attributes of a user.

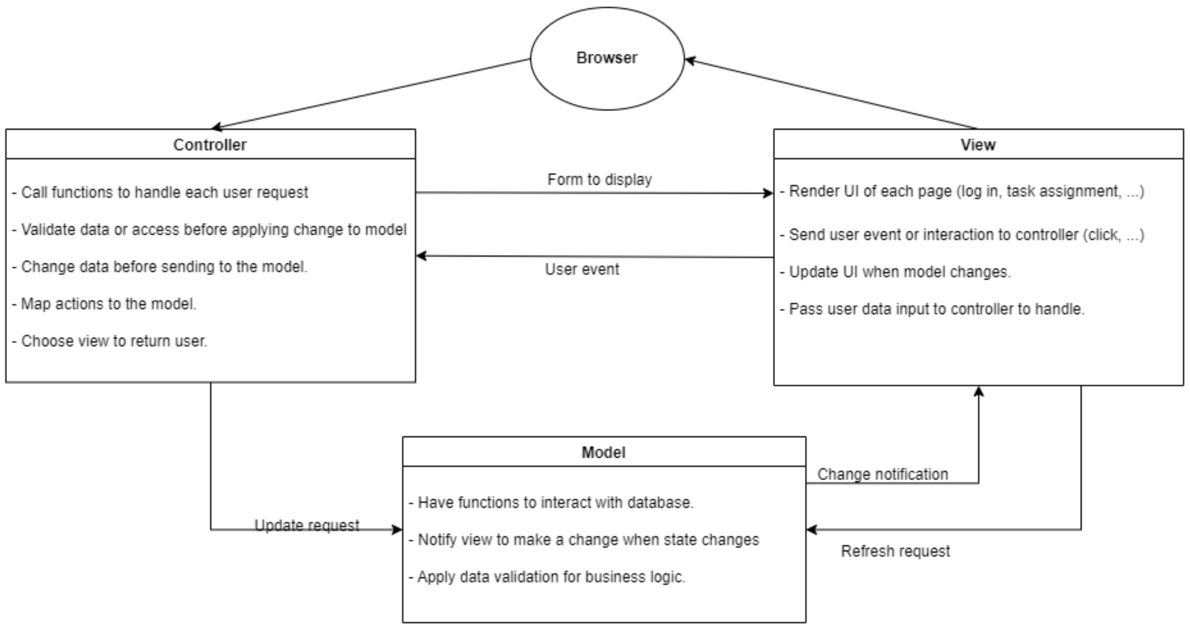
-Janitor, Collector, BackOfficer: are 3 abstract class from User, each containing a function to view the task
-Task: contain the basic attribute of a task, by calling the show() function, the data will be sent to <<interface>>display to display that task for the user
-If a back officer wants to assign the task, they can access the task menu and by using assignEmployee(), assignVehicle() to assign the task.
-The controller will control some basic functions to assign the task and also to create an optimized route for the vehicle.
-After assigning the task, the data will transfer to the server (database) to add that task to the task list, back officer can access this database to handle data
-By sending data to the server, there will be a notification to all user

TASK 3: Architecture design

3.1 Architecture approach and modules:

3.1.1, Architecture approach:

For the UWC 2.0 system, our group decided to use MVC (Model-View-Controller) for design patterns. For this design pattern, we group all the system components into three main groups which are controller, view, and model. Below is the diagram to show the flow of MVC working and the table for analyzing some main operations of each group, which also includes the pros and cons of the model.


Browser
Controller
- Call functions to handle each user request
- Validate data or access before applying change to model
- Change data before sending to the model.
- Map actions to the model.
- Choose view to return user.
Form to display
User event
View
- Render UI of each page (log in, task assignment, ...)
- Send user event or interaction to controller (click, ...)
- Update UI when model changes.
- Pass user data input to controller to handle.
Model
- Have functions to interact with database.
- Notify view to make a change when state changes
- Apply data validation for business logic.
Update request
Change notification
Refresh request

Name	Description
View	Render different UI for the different pages: - Render form login for user. - Render work space for each type of user. - Render calendar or task description - … Interact with user by recording the event user make in the interface and send them to controller: - For example, when back officer clicks a specific button to see the list of the employee, view must be recording this action and send to the controller to analyse this event and send the result to view to render the result to back officer Update UI when model changes - For example, when an employee checks out the task. After the database updates the status of the task, the view will automatically

	reload the UI of the employee when the task is finished. Pass user data input to the controller for handling. - When a user inputs their username and password for logging in, the view will send them to the controller for authentication.
Controller	Send appropriate view depending on the request. - For the back officer's page or employee's page, it has differences in their pages, and the controller will do it. Call function to handle each request or each router. Can validate data before sending it to the database. - For example, when people send the data from the view, the controller can change the content of the data to be appropriate for the database. Map actions to the model. - Controller maps the action from the view to the model like when a back officer wants to read data from the view by clicking a button, then this will map this action to the model which can be interacted with the database.
Model	Interact with the database - Model can read, create, update or delete the data. Notify view to change the UI when the status of model about data changes (notify View to refresh and update the page)

Advantages and disadvantages of the MVC model:

- Advantages:
    - Development is quicker and easier for a complex project.
    - Easy for updating the application because each component of the application is clearly separated.
    - Easier for multiple developers to collaborate and work together.
    - Easier to debug as we have multiple levels properly written in the application.
- Disadvantages:
    - It is quite difficult to understand the MVC architecture
    - Must have strict rules on methods
    - Difficult to separate functions between controller and model, which easily overlap.

3.1.2, Modules:

Module	Task Assignment
Input	- User wants to assign tasks to employees.

Output	- Task is assigned successfully and stored in the database.
Functions	- loadMCPInformation(): return information about MCPs like location, status, etc. - loadEmployeeInformation(): return employee's information, calendar, etc. - loadVehicleInformation(): return vehicle's information like technical details. - assignMCP(): use to assign MCPs (areas) to janitors. - assignVehicle(): use to assign vehicles to collectors.

Module	Handle Task
Input	- Employees like janitors and collectors want to handle the task.
Output	- Employees finish the task and the database records the status of the task.
Functions	- viewTask(): return the information of the task. - checkIn(): use for employees to check in when they receive the task and can be able to do it. - checkOut(): use for employees to check out when they finish the task.

Module	Chat
Input	- Users want to send messages to a specific person.
Output	- Message was sent successfully and stored in the database. - Receiver is notified about the message.
Functions	- sendMessage(): used for sending messages. - receiveMessage(): use for receiving messages and notifying the receiver.

Module	Authentication
Input	- Users want to log in to the system to interact with the program.

Output	- Users login successfully or fail to login due to some errors in the information. The system will send notification about login status to users.
Functions	- checkUsername(): use for the system to check the username inputted by the user. If it appears in the database, the system will check the username box. Otherwise, it will send an alert to the user that this username does not exist. - checkPassword(): use the system to check the password of that username. If it correctly matches, the system will announce the user has login successfully. Otherwise, it will send an alert to the user that the password inputted was incorrect.

Module	Routing
Input	- Giving the location of two destinations include: the assigned MCP and the headquarter
Output	- Calculating the optimized route based on several criteria: length, weather, rush hour, etc...
Functions	- getMCPsLocation(): return the location of assigned MCP - calculateOptimizedRoute(): calculate the optimized route from the headquarter to assigned MCPs

Module	Notification
Input	- Back officer assigns tasks. - Users send messages.
Output	- Notification was sent successfully.
Functions	- sendTaskNotification(): use to send notification for a task. - sendMessageNotification(): use to send notification for a waiting message.

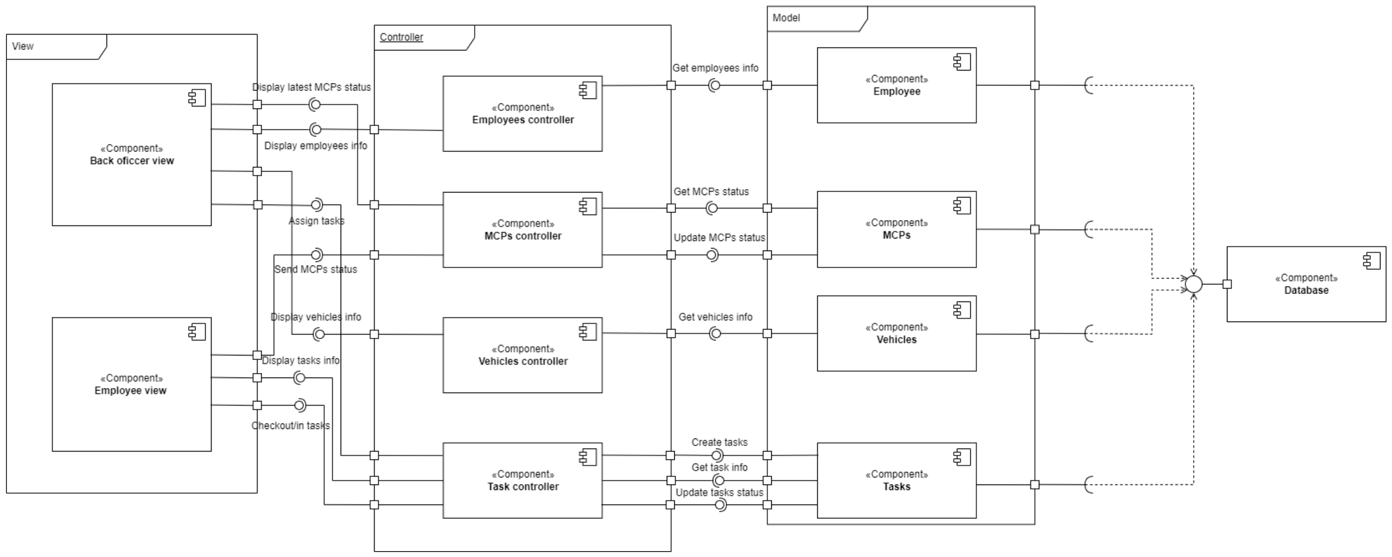
Module	View Employee Information
Input	- Employees want to view their own information

	- Back officers want to view employee’s information
Output	- Information was displayed in the UI.
Functions	- viewEmployee(): use for view information only of one employee. - viewAllEmployee(): use for view information of all employees.

Module	View VehicleInformation
Input	- Employee wants to view all information about vehicles. - Collectors want to view the vehicle which is assigned to them.
Output	- Information was displayed in the UI.
Functions	- viewVehicle(): use for view only information of one vehicle. - viewAllVehicles(): used for view information of all vehicles.

3.2 Implementation diagram for Task Assignment module:

3.2.1: Implementation diagram:



3.2.2: Description:

Task assignment component diagram specification. There are 3 big components:

-View component:

+ It contains 2 components : Back officer view and Employee view for specific user’s type (janitor, collector, back officer)

+ These view components receive requests from users and require interface from Controller view to handle requests.

+ These view components use the interface from model to update view when they are notified about updating in model components.

+View component is described more clearly in task 3.1.

-Controller component :

+It contains some small components.

+These mini controller components provide interface for View component to handle request , use interface from model to perform operations with database.

+Controller component is described more clearly in task 3.1.

-Model component:

+It contains many small components which are described more clearly in task 3.1.

+These model components provide the interface for controllers to perform operations in the database and provide the interface for updating views when notified by model.

+Each small component can perform business logic functions in s implementation.

-Database component. This component contains a database of the system, including employee information, back officer information, task list, vehicles information and MCPs information. The model component will interact directly with the database component to get necessary information.