VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING ASSIGNMENT

PROJECT: UWC 2.0 (TASK 3)

CC02 --- GROUP HÍ HÍ HÌ HÌ --- SEM 221

SUBMIT: 10/11/2022

INSTRUCTOR: PROF. ĐỨC ANH

Students' name	Student's ID
Nguyễn Viết Hòa	2052486
Trần Gia Linh	2053182
Nguyễn Mỹ Khanh	2052525
Trần Ngọc Oanh	2053312
Phạm Thị Thái Minh	2052174

Ho Chi Minh City – 2022

CONTENTS

TASK 1: REQUIREMENT ELICITATION

1. CONTEXT OF PROJECT

1.1 STAKEHOLDERS:

Internal stakeholders are people whose interest in a company comes through a direct relationship, such as employment, ownership, or investment. External stakeholders are those who do not directly work with a company but are affected somehow by the actions and outcomes of the business.

1.1.1 External: The parties or groups that are not a part of the organization, but gets affected by its activities

1. Government
2. Citizen of nearby area
3. Private Sweeper
4. Citizen paying for service
5. Service provider Y

1.1.2 Internal: The individual and parties that are the part of the organization

1. Back officers
2. Collectors
3. Janitors
4. Organization X

1.2 CURRENT PROBLEMS OF URBAN WASTE MANAGEMENT: (POSSIBILITIES)

1.1.3 Waste management

Urban waste collected at temporary disposal are not separated between organic and non-organic waste => No training about waste management

Temporary dumps usually are not well-managed => Create population and health risk for local communities.

Private Sweepers usually use old technology to collect and dispose waste, they do not obtain modern and efficient technology to properly dispose waste.

1.1.4 Organization:

Back officer:

Lack information about type of vehicles and MCPS => Do not know how to assign task for janitors and collectors.

Collectors and Janitors:

Need to learn to use multiple different platforms to perform their task correctly (Calendar to see work shifts, Map to see the proper route, etc, …) => More time training. Unable to get access to emergencies immediately, have to make a call and remember who are their supervisors.

Unable to focus on their specialization, have to be cautious for everything (is MCP full, which route is the most efficient, …)

Unable to get access to the necessary information and have to make requests for everything.

1.3 CURRENT NEED OF STAKEHOLDERS:

Citizen: Need better waste management. Get informed about waste collection calendar to properly classify trash and dumps. Need healthier environment where trash is not around.

Government: Need better waste management. Trash dumps are well-managed => Waste do not contaminate water and land source. More modern and green technology to process waste.

Back officer: Information about 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. Information should be updated from MCPs every 15 minutes with the availability of at least 95% of their operating time

Map of area (include traffic information, route, distance and fuel consumption) to properly locate MVP and send collectors, janitors.

Ability to contact collectors and janitors for emergency.

Collectors and Janitors: Information about their work shifts (include time, location and MVP they are assigned to). Have a detail view of their task on a daily and weekly basic. All important information should be displayed in one view (without scrolling down).

Be able to communicate with collectors, other janitors and back officers. The messages should be communicated in a real-time manner with delay less than 1 second

Collectors: Route to get to their destinations. Information about their vehicles (max capacity, fuel consumption, …). Type of their MCPs and contact info of janitors who are responsible for their MCPS. Notified if the MVPS is fully loaded

Janitors: Map of their assigned area. Notified if the MVPS is fully loaded. Route to get to their MVPS and information of family (waste type, …) they are going to collect.

1.4 BENEFITS

Citizen: Better waste management. Trash is collected on same schedule and location => Easier to keep track.

Government: Better waste management. Trash dumps are well-managed. Better environment where trash is handled with utmost care.

Back officer: Can easily keep track of collectors and janitors work calendars. Can easily obtain information about different kinds of vehicle, route and MCPS => Easier to assign work calendars and proper vehicles for collectors and janitors.

Can immediately contact with collectors and janitors => Keep them informed about emergency or sudden change of route because of traffic problems.

Collectors and Janitors: Can keep track of their work shift easily, can focus on their specializations without caring too much about external information. Can get access to their work information directly => Need less training on how to use the platform => Less thing to remember and work on.

Be able to communicate with collectors, other janitors and back officers to inform about emergencies and get notified about sudden change of plans immediately.

2. REQUIREMENTS INDENTIFY AND USE-CASE FOR WHOLE SYSTEM

2.1 IDENTIFY REQUIREMENTS

> *"In reality, the distinction between different types of requirements is not as clearcut as these simple definitions suggest. A user requirement concerned with security, such as a statement limiting access to authorized users, may appear to be a nonfunctional requirement."*
>
> (Software Engineering _ Ian Sommerville)

As the book has stated, there is no clear distinction between function and non-function requirement hence our below specifications are based mainly on our intuition, knowledge and experience.

1.1.5 Function requirements

As a back officer, a collector and a janitor, I want a **Chat box** so that I can contact to the others.

As a back officers, I want a **Calendar** so that I can assign the tasks for collectors and janitors, the route for collectors, the MCPs for the collectors and the janitors.

As a collector and a janitor, I want a **Calendar** so that I can see my tasks in daily and week basic and check in/ check out my work in any time.

As a back officer, I want a **Vehicle management** so that I can know the details of the vehicles (weight, capacity, fuel consumptions, in-use, etc) and assign vehicle for the collectors.

As a back officer, a collector and a janitor, I want a MCPs management system with a map so that I can design the route, keep up with the MCPs state and be notified if the MCPs are fully loaded.

1.1.6 Non-function requirements:

As a back officer, I want **efficiency** so that the system can handle real-time data from at least 1000 MCPs at the moment and 10.000 MCPs in five years.

The system **interfaces** should be in Vietnamese, with a button to switch to English.

As a user, I want **high-speed** so that the messages can be communicated in a real-time manner with delay less than 1 second.

The information should be updated from MCPs every 15 minutes with the **availability of at least 95%** of their operating time.

2.2 USE-CASE DIAGRAM

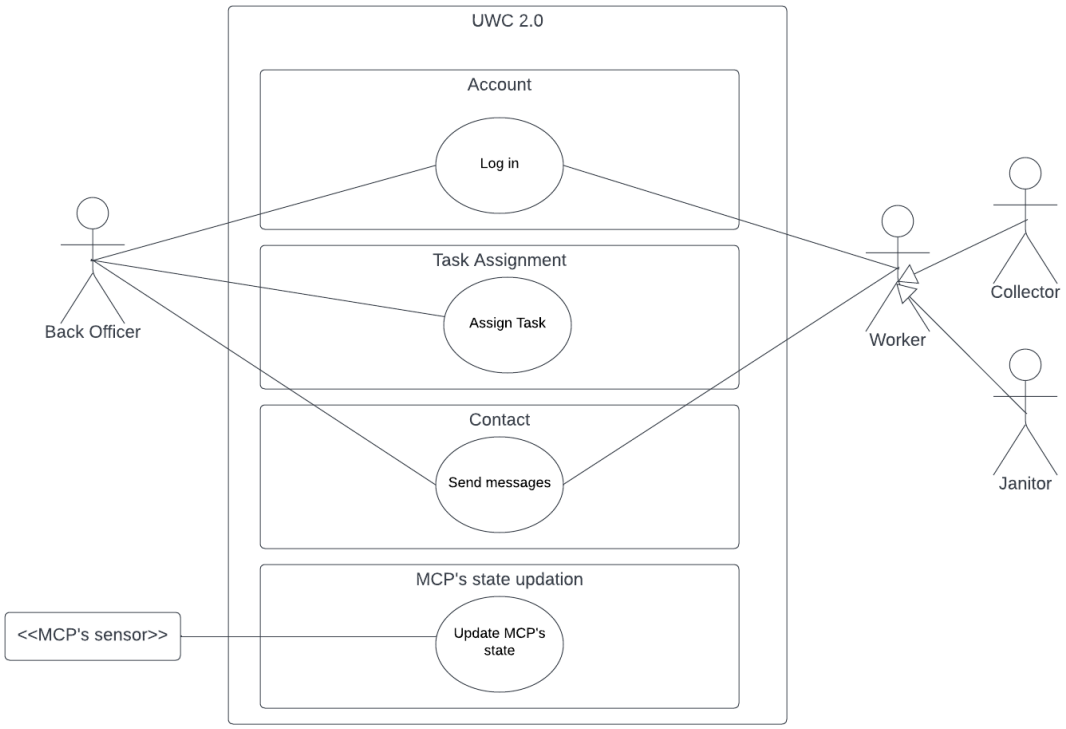


Figure 2.1: Use-case Diagram for the whole system

As you can see in the use-case diagram, we divide our program into **5** main parts:

1. **Log in:** In this part, we will allow users (Workers and Back Officer) to log in their accounts to verify their identities. There are restricted parts for different users, therefore, this log in part will help improving security and help us to protect private personal information.
2. **MCP's state update:** In this part, each MCP will have a sensor, when it is full or empty, the sensors will send data back to the system.
3. **Task Assignment Module:** This part will help the Back Officers to assign and then update the necessary information about tasks/calendars, … for the workers. This part will be analyzed further below in Part 3.
4. **Contact:** Last but not least, this part will help both Workers and Back Officers to communicate with each other in case of emergency. This attribute will help to enhance communication and solidarity among peers in work.

3. TASK ASSIGNMENT MODULE USE-CASE DIAGRAM AND TABLE

3.1 USE-CASE DIAGRAM

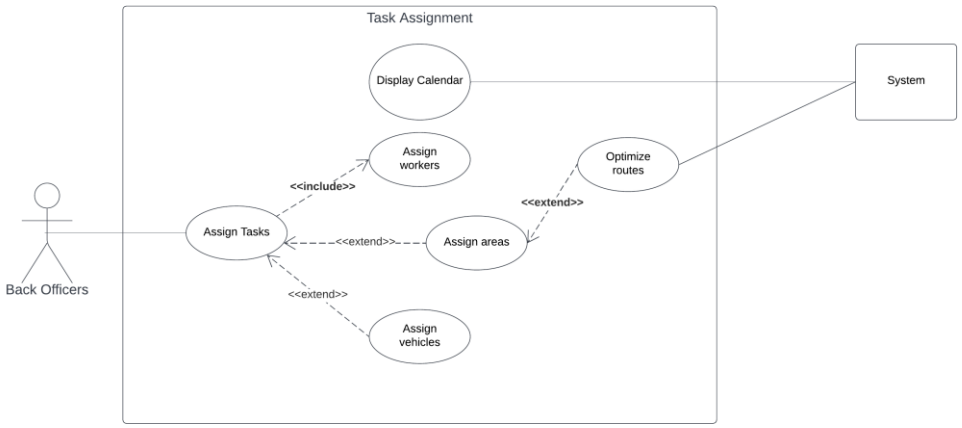


Figure 3.1: Use-case diagram of Task Assignment Module

3.2 USE-CASE TABLE

Use-case name	**Display calendar**
Actors	System
Trigger Condition	Back officers click the log-in button on the screen.
Description	System displays the calendar on the screen.
Pre-Conditions	Back officers log in successfully to the web.

Post-Conditions	Calendar information is shown.
Main Flow	1. System display the calendar to the screen.
Exception	
Alternative flow	At step 1, there may have Internet corruption. If that situation occurs, a notification will display and request users to refresh the page in order to re-access

Use-case name	**Assign vehicle**
Actors	Back officers
Trigger Condition	The worker is chosen to be a collector.
Description	Back officer assigns 1 vehicle to the collector.
Pre-Conditions	There is at least 1 available vehicle on that day.
Post-Conditions	The collector is assigned to 1 vehicle.

Main Flow	1. System shows a selection list of available vehicles on that day. 2. Back officer chooses 1 vehicle from the list.
Exception	At step 1, if the list of available vehicles is empty, the system will send an alert to the back officer instead.
Alternative flow	

Use-case name	**Assign worker**
Actors	Back officers
Trigger Condition	Back officers are on the “Add assign" page.
Description	Back officer chooses 1 free worker for the task.
Pre-Conditions	There is at least 1 free worker and the Internet is accessible.
Post-Conditions	Back officer chooses 1 free worker.
Main Flow	1. System shows a selection list of free workers on that day. 2. Back officer chooses 1 worker in the list.

Exception	At step 1, if the list is empty, the system will send an alert to the back officer instead.
Alternative flow	

Use-case name	**Assign tasks**
Actors	Back officer
Trigger Condition	Back officers click 1 date in the calendar and then click "Add new assignment" on the Day Page.
Description	Back officers assign 1 task for a date in the calendar.
Pre-Conditions	Database is available and Internet is accessible.
Post-Conditions	Back officers create a new assignment.
Main Flow	1. System move to "Add assign" page.
Exception	
Alternative flow	

Use-case name	Optimize route
Actors	System
Trigger Condition	The worker is chosen to be collector.
Description	System make optimal route which go through all MCPs in chosen area.
Pre-Conditions	Chosen area(s).
Post-Conditions	A route is optimized and assigned to the task.
Main Flow	1. System optimizes 1 route from MCPs' positions in the chosen area(s).
Exception	
Alternative flow	

Use-case name	Assign area
Actors	Back Officers
Trigger Condition	Back officer identified the type of worker and is on "Assign area" page.
Description	Back officer assigned area(s) to a task - worker.
Pre-Conditions	The list of areas is not empty.
Post-Conditions	At least 1 area is assigned to the task.
Main Flow	1. + If the worker is a janitor, the system displays a selection list of areas. + If the worker is a collector, the system displays a selection list of unassigned areas. 2. + If the worker is a janitor, back officer chooses 1 area. + If the worker is a collector, back officer chooses 1 or more areas.

Exception	At step 1, if the worker is a collector and the list of unassigned areas is empty, system will send an alert to back officer.
Alternative flow	

TASK 2: SYSTEM MODELING

1. ABOUT UML

In this task, we will encounter 3 types of UML which are: **Activity Diagram**, **Sequence Diagram** and **Class Diagram**. We will further explain these types below.

There are two main categories of diagrams which are **structure diagrams** and **behavioral diagrams**

Structure Diagrams: show the things in the modeled system. In a more technical term, they show different objects in a system. They also show the structure of the system by illustrate relationships and function between object. Class Diagram belongs to this category.

Behavioral Diagrams: show what should happen in a system. They describe how the objects interact with each other to create a functioning system. They display and explain the workflow between object. Activity Diagram and Sequence Diagram belong to this category.

1.1 CLASS DIAGRAM

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class.

1.2 ACTIVITY DIAGRAM

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system.

1.3 SEQUENCE DIAGRAM

Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario.

As we can see each diagram has different attributes satisfying particular requirements, therefore in order to accurately describe the whole system, we have to use three different UML diagrams as mentioned above.

2. ACTIVITY DIAGRAM

2.1 ACTIVITY DIAGRAM

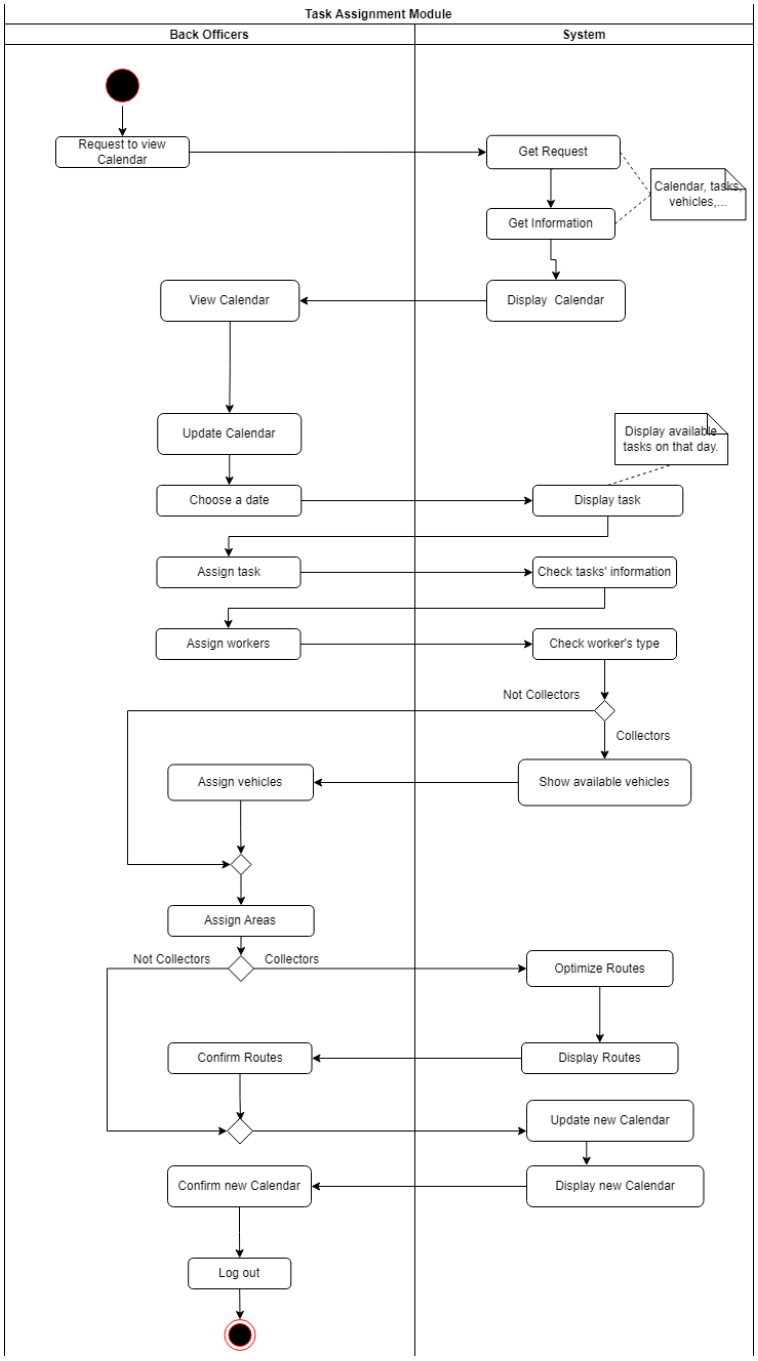


Figure 2.1 Activity diagram for Task Assignment Module

2.2 DESCRIPTION

- The system will be initiated by back officers.
- When they open the application, they will send a request to the system to view the calendar, then the system will display the requested calendar, which consists of the current calendar, tasks and vehicles,...
- After viewing the calendar, back officers can start to update the calendar by choosing a date, the system then will display available tasks on that day and allow the back officers to assign them.
- The back officers then will be allowed to view the information of those tasks (including information about description, workers, vehicle, MCPs, areas,..). Back officers will be able to choose from a list of workers and assign the appropriate person for the jobs.
- The system will check for the worker's type. If it is a collector, back officers can then check information about available vehicles and assign them to the worker and then move on to the area assigning part. If it is not a collector, then the system will move to an area assigning part without vehicle assigning.
- After officers have assigned areas for workers, the system will then optimize routes if workers are collectors and have vehicles. Then it will show the routes to the back officers for them to confirm.
- After areas are assigned and routes are optimized, the system will update the calendar, display the calendar to back officers and ask for their confirmation.
- Back officers can then view and confirm the updated calendar and finish their jobs and log out.

3. CONCEPTUAL SOLUTION_SEQUENCE DIAGRAM

3.1 SEQUENCE DIAGRAM

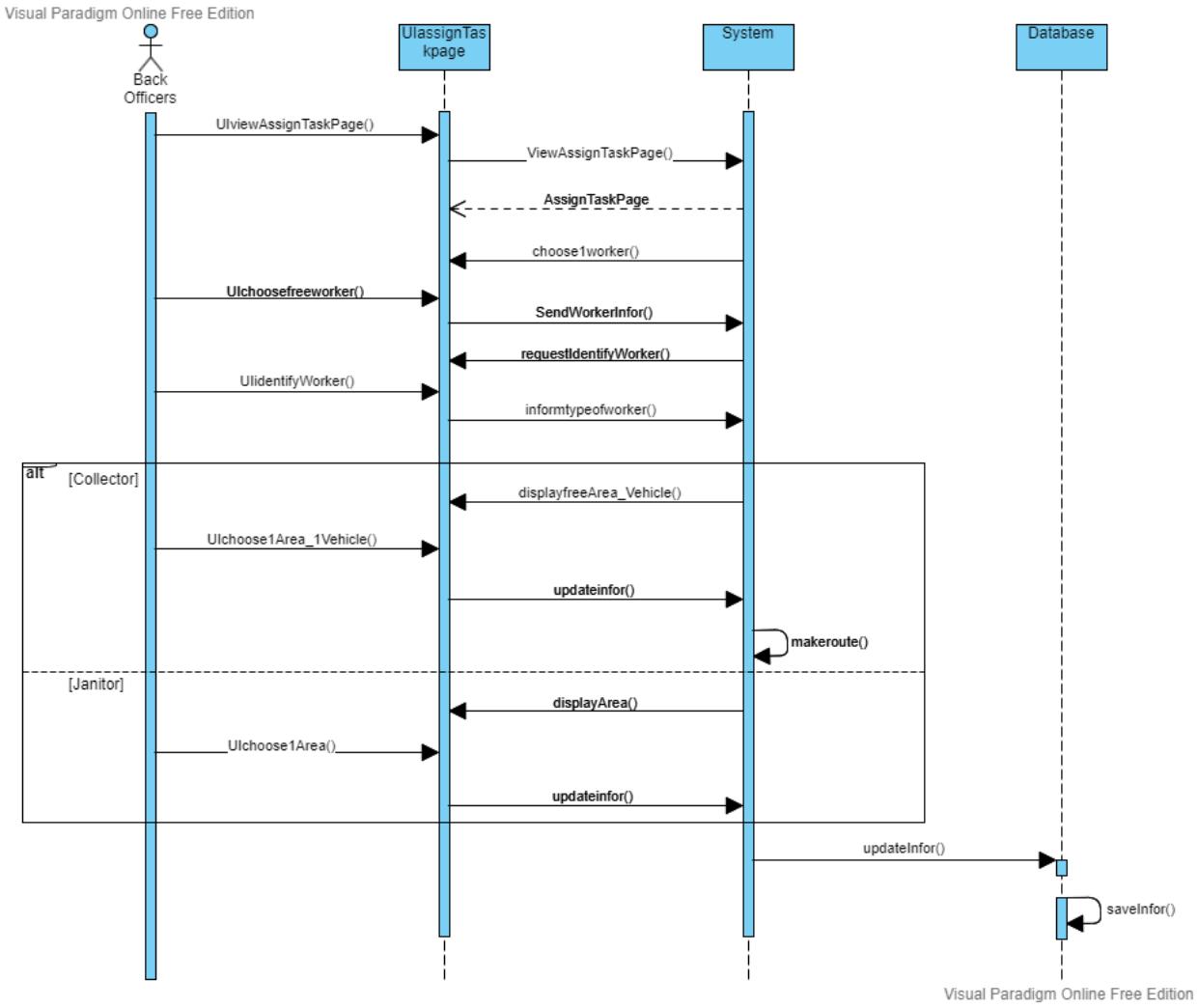


Figure 3.1 Sequence Diagram for Route Optimizing

3.2 CONCEPTUAL SOLUTION

- In this function, the user will go through the UI to ask the system to display the Assign Task page.
- The system receives the request and then returns to the Assign Task page
- The system asks the UI to display a menu containing a list of workers who are free for the day.

- The user goes through the menu to select a worker and sends the worker information to the system through the UI.
- The system receives the information and asks the UI to show 2 options (collector/janitor).
- The user defines the worker's type.
- The UI sends the selection information to the system.
- If the option is "Collector":
    - The system will ask the UI to show a menu with a list of uncollected areas and vehicles that are free for the day.
    - The user, through the UI, selects 1 or more regions and 1 vehicle from the list.
    - UI sends information back to the system.
    - The system relies on the information about which areas to generate the shortest route.
- If the option is "Collector":
    - The system will ask the UI only to show a selection list of areas.
    - The user, through the UI, selects 1 area.
- The system sends information back to the database.
- Database updates new task information.

4. CLASS DIAGRAM

4.1 CLASS DIAGRAM

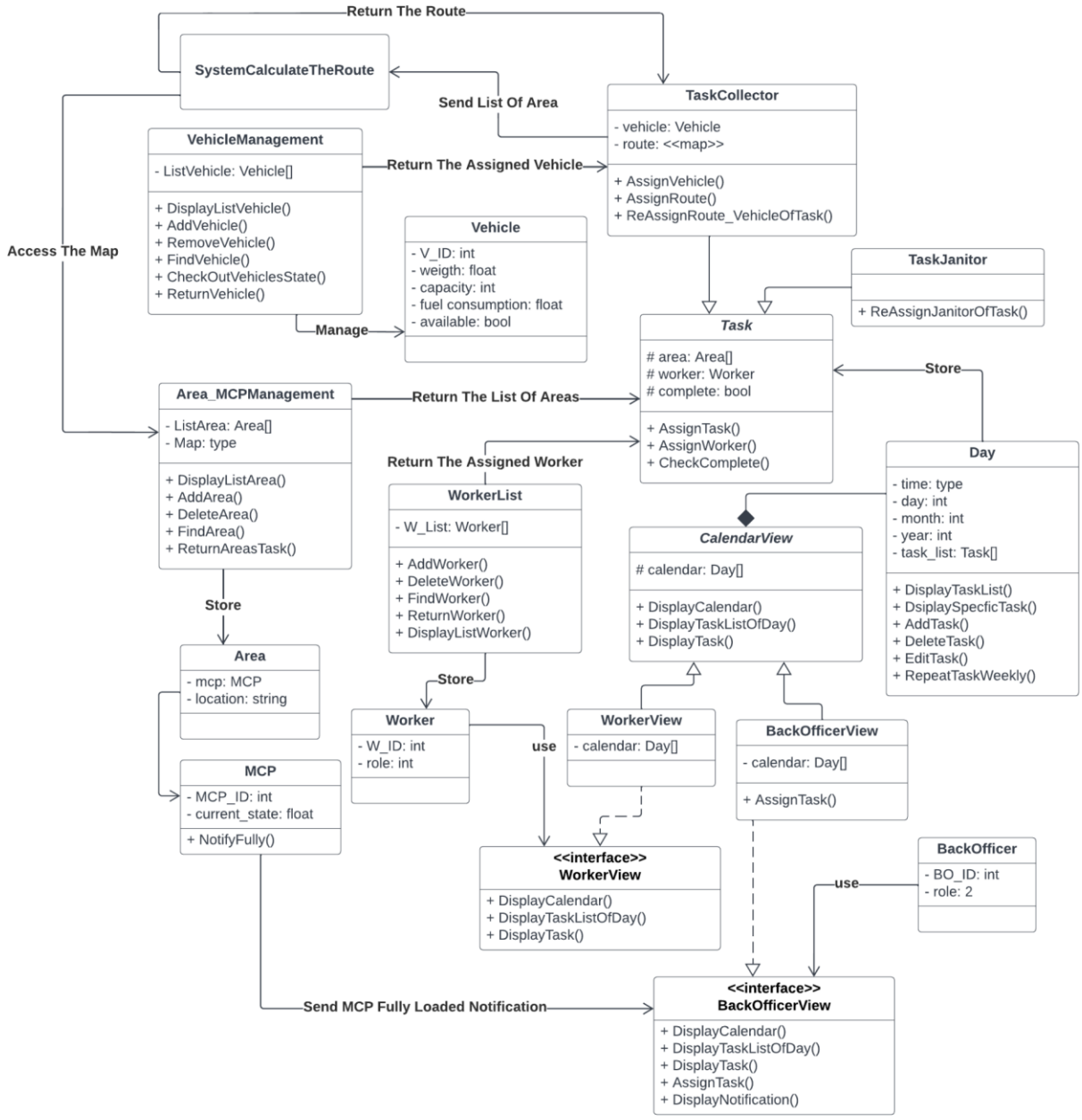


Figure 4.1 Class Diagram for Task Assignment Module

4.2 DESCRIPTION

- Workers and Back Officers can interact with the system through their interfaces which are *WorkerView* and *BackOfficerView*.
- The Back Officer can assign new Tasks through the Calendar in the interface.
- The Back Officer will assign the Task with the Worker from the *WorkerList* and the Area(s) that Worker will be in charge from the *Area_MCPManagement*. With different kinds of Worker (*Collector* and *Janitor*), the Task will have more things to assign.
- The Back Officer can re-assign the Janitor for a Task weekly via *ReAssignJanitorOfTask()*.
- For the Collector, the Back Officer needs to assign the Vehicle used for the task from the *VehicleManagement*, then, assign the Route for that task which is calculated by the System.
- The Back Officer can also re-assign the Route and the Vehicle used for a Task monthly.
- When the users choose a specific day in the Calendar, the list of the tasks of that day will be displayed. After a specific task is chosen, the information of that task will be shown on the screen including what the Back Officer has assigned.

TASK 3: ARCHITECTURE DESIGN

1 ARCHITECTURAL APPROACH

Our team's MWC product will be a website so we choose to use **MVC** architectural approach **(Model - View - Controller) model**.

1.1 THEORY

About MVC design pattern

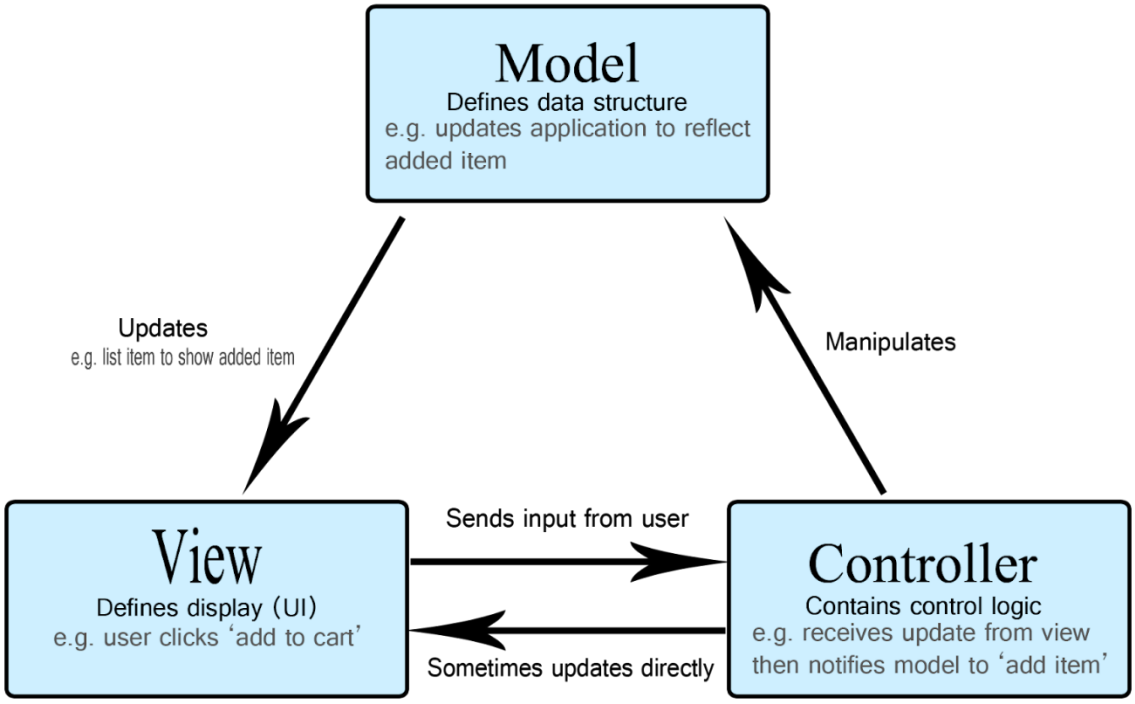
MVC (Model-View-Controller) is a pattern in software design which specifies that an application consists of data model, presentation information, and control information. It emphasizes a **separation between the software's business logic and display**. This "separation of concerns" provides for a better division of labor and improved maintenance. Some other design patterns are based on MVC, such as MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter), and MVW (Model-View-Whatever).

To explain clearly, we took a simple shopping list app which used the MVC model to design. Now we go into **3 parts** of the MVC design pattern.

The Model

The model defines what **data** the app should contain. If the state of this data changes, then the model will usually notify the view (so the display can change as needed) and sometimes the controller (if different logic is needed to control the updated view).

Going back to the shopping list app, the model would specify what data the list items should contain — item, price, etc. — and what list items are already present.



The View

The view defines **how the app's data should be displayed.**

In the shopping list app, the view would define how the list is presented to the user, and receive the data to display from the model.

The Controller

The controller **manipulates** the database in model according to events triggered by external sources (i.e view) in a **logical** way. Since the view and model are connected, the action of the controller will automatically change the view.

So for example, a shopping list could have input forms and buttons that allow us to add or delete items. These actions require the model to be updated, so the input is sent to the controller, which then manipulates the model as appropriate, which then sends updated data to the view.

You might however also want to just update the view to display the data in a different format, e.g., change the item order to alphabetical, or lowest to highest price. In this case the controller could handle this directly without needing to update the model.

1.2 SOLUTION

We will divide our system into **4 big modules** discussed in task 1:

1. Account
2. Task Assignment
3. Contact
4. MCP's state updation

1.2.1 Account Module:

Input: Username and Password (Log in ID).

Function: This module will take in user login information (user name, password, ..), compared with the registered ID in the database and return false if no ID is found or user's data if ID is found. We do not have to care about the new register since every employee's account is provided by back officers.

Output: User's authentication and ID.

1.2.2 Task Assignment Module

Input: User ID, Task Information (workers' ID, vehicle's ID and its status (if any), MCP's ID, routine (if any), date and time)

Function: This module will take user ID from the system to identify if the users are a back officer or not, if false then they cannot access this module, if true they will next consequently insert task's information when the system processes. The task will be assigned to the workers and shown in the calendar according to the workers' ID and date and time in the input.

Output: An updated calendar with proper tasks' assignment.

1.2.3 Contact module

Input: Message, Sender ID, Receiver ID

Function: This module will take the messages from the sender and send the message through the network for the receiver.

Output: Text output at the receiver's view.

1.2.4 MCP's state updation

Input: MCP's id, new state

Function: This function will process the MCP whose ID matches the input ID, discard its old state and update the new one.

Output: New state of MCP

1.2.5 MVC adaptation

1. Account Module:

Model: Database of register's User ID

View: Authentication UI (for user to type in their log in ID)

Controller: Receive users' login ID -> Ask Model to retrieve login ID from database, if Model cannot find such an ID, directly update View to show Reject notification, if Model finds such an ID, Model sends notification to View to display user's data view of that ID.

2. Task Assignment Module:

Model: Calendar containing all tasks

View: Calendar UI for back officers to assign

Controller: Receive assignment information from back officers, update Model then distributed the new calendar (discussed further in 3.2)

3. Contact module:

Model: Chat log of users

View: Chatbox between users

Controller: Receive messages from sender, update it into Model, Model updates that message in both user's data and then notify the View to update newly sent data.

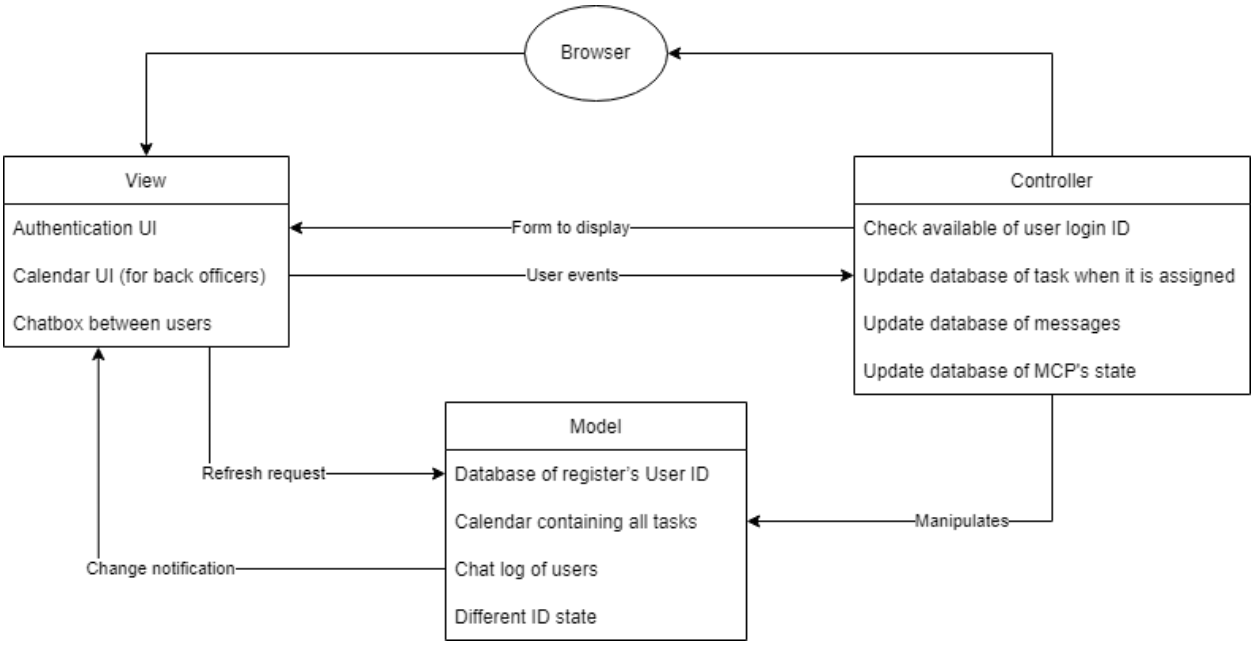
4. MCP's state updation:

(This updation will be performed by a sensor so we need no UI)

Model: Different ID state.

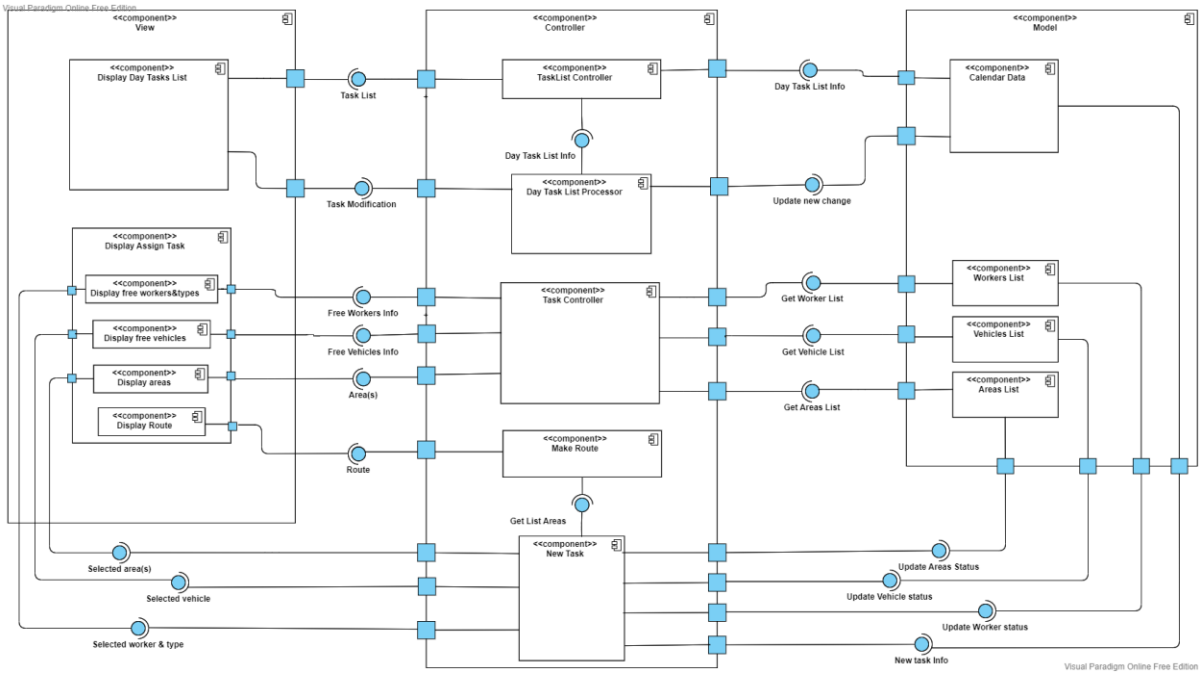
View: No UI is needed

Controller: Receive state from MCP, update it to Model


Browser
View
Authentication UI
Calendar UI (for back officers)
Chatbox between users
Form to display
User events
Controller
Check available of user login ID
Update database of task when it is assigned
Update database of messages
Update database of MCP's state
Model
Database of register's User ID
Calendar containing all tasks
Chat log of users
Different ID state
Refresh request
Change notification
Manipulates

2 IMPLEMENTATION DIAGRAM

2.1 IMPLEMENTATION DIAGRAM



2.2 DESCRIPTION

At the calendar user-interface, BackOfficer chooses one day in the calendar. Component *TaskList Controller* gets assigned-task information on selected day and displays information in component *Display Day Tasks List.*

In the situation that **BackOfficer** modifies a task that has been assigned before, all modification will be processed in component *Day Task List Processor*, then component *TaskList Controller* obtains adjusted information and displays it in component *Display Day Tasks List*. Modification is then updated into component *Calendar Data* when it has been confirmed.

In the situation that BackOfficer needs to create new tasks in one day, component *Task Controller* will get information from *Workers List, Vehicles List, Areas List* and render it on the interface for **BackOfficer** to choose. Then, the component *New Task* will combine that information. Component *Calendar Data, Workers List, Vehicles List, Areas List* will be updated via component *New Task.*

In the situation that the worker type of the *New Task* is **Collector**, component *Make Route* will get the areas list selected from the *New Task* to make the route.

END OF REPORT