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

SOFTWARE ENGINEERING

Assignment

Urban Waste Collection Aid
UWC 2.0

Instructors/Lecturers: Quan Thanh Tho

Authors: Nguyen Ha Trong Hieu - 2052992
Nguyen Manh Dan - 2052932
Bui Quoc Minh Quan - 2011905
Nguyen Hai Dang - 2052444

HO CHI MINH CITY, 12/2022

Contents

1 Evaluation

No.	Full Name	Student ID	Work in the assignment	Evaluation
1	Nguyen Manh Dan	2052932	Setting up online repository (github), Preparing presentation slides, Implement MVP1	25%
2	Nguyen Hai Dang	2052444	Proposing solution for route planning + Sequence diagram, Preparing presentation slides, Implement MVP1	25%
3	Nguyen Ha Trong Hieu	2052992	Class diagram for Task assignment module, Architecture design for whole system, Deployment diagram for Task assignment module, Preparing report	25%
4	Bui Quoc Minh Quan	2011905	Use-case diagram for Whole system and Task assignment module, Defining System requirement, Implement MV1 & MVP2	25%

2 Requirement Elicitation

2.1 Context and Stakeholders

Urban waste management has been of the most notable problem of the world.

In an attempt to solve this problem and achieve Sustainable Development Goal (SDG), improvement on waste collection and management must be made.

The relevant stakeholders in this project include:

- The back officers
- The collectors
- The janitors

Their current need is that they require an information management system through which they can communicate and coordinate with one another.

Benefits for the stakeholders:

- Back officers: provides the capability to create calendar, coordinate front collectors, janitors and assign tasks. Assists vehicle planning activity.
- Collectors: provides information of all MCPs to drive through and the predetermined route.

- Janitors: provides information about location of MCPs, to which they can deliver garbage to after collection.

2.2 Functional Requirements

2.2.1 Back Officers

- Each back officer should have an overview of janitors and collectors, their work calendar.
- Each back officer should have an overview of vehicles and their technical details.
- Each back officer should have an overview of all MCPs and information about their capacity.
- Each back officer can assign vehicles to collectors.
- Each back officer can assign janitors to MCPs.
- Each back officer can create a route for each collector. Assigned route is optimized in term of fuel consumption and travel distance.
- Each back officer should be able to send message to collectors and janitors.

2.2.2 Collectors and Janitors - Front employees

- Each front employee should have an overview of their work calendar.
- Each front employee should have a detail view of their task on a daily and weekly basic. All important information should be displayed in one view.
- Each front employee should be able to communicate with collectors, other janitors and back officers. Messages should be delivered in a real-time manner with delay less than 1 second.
- Each front employee can check in / check out task every day.
- Each front employee will be notified about the MCPs if they are fully loaded.

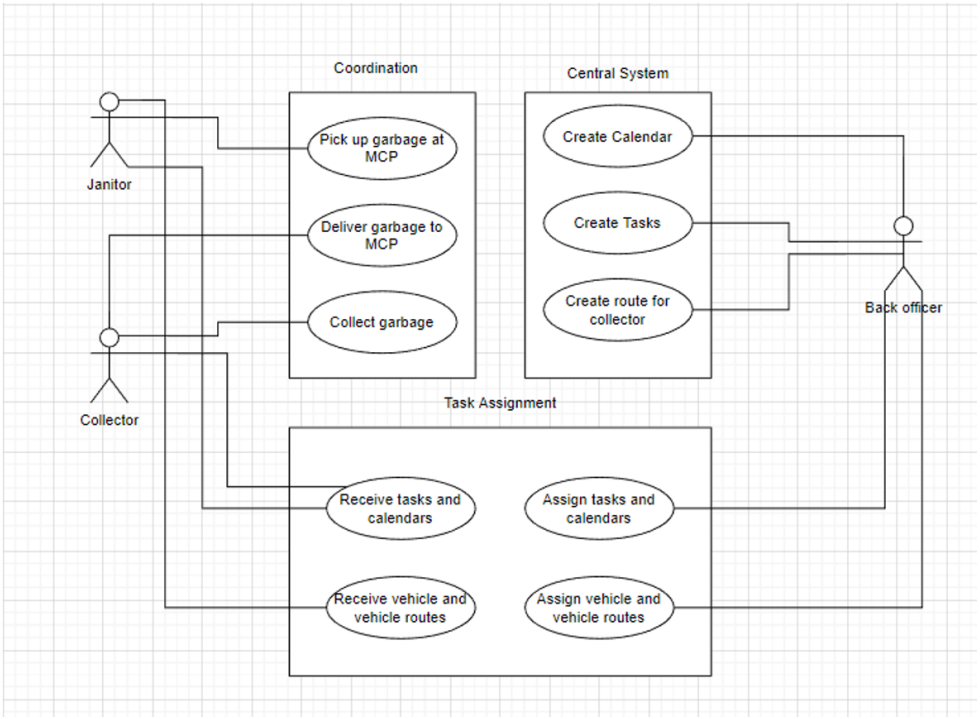
2.3 Non-functional Requirements

- UWC 2.0 is expected to import and to use the existing data from UWC 1.0.
- UWC 2.0 must 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.

2.4 Use-case Diagram

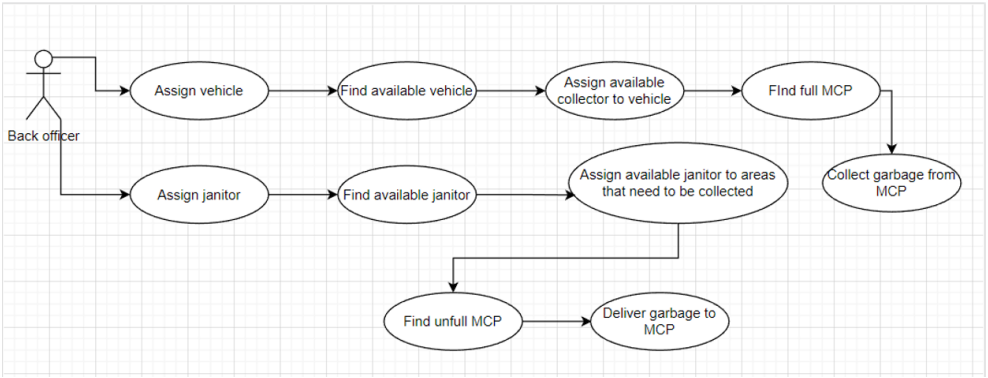
2.4.1 Whole system



Use-case diagram of the whole system

2.4.2 Task Assignment Module

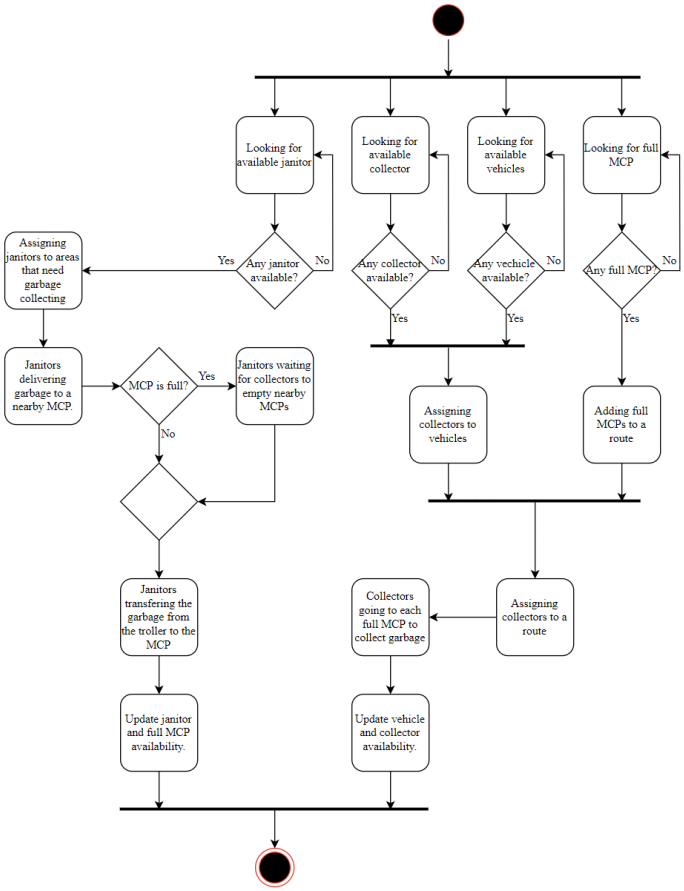
Use case	Description
Name	Task Assignment
Actors	Back officer, collector, janitor
Precondition	Collector, janitor, vehicle must be available
Post-condition	All MCP's garbage is collected
Basic path	This use-case starts when the back officer assign vehicle to the collector and assign janitor to areas which need garbage-collecting. Find MCPs. All MCPs' garbage is collected by the collector.
Alternative path	At step 2 of the basic path, if the MCP is full, assign collector to collect garbage.
Exceptional path	At step 2 of the basic path, if the MCP is unfull, assign janitor to deliver garbage to that MCP



Use-case diagram of the Task Assignment module

3 System Modelling

3.1 Business process in Task Assignment module



Activity diagram of the Task Assignment module

3.2 A conceptual solution for the route planning

Objectives:

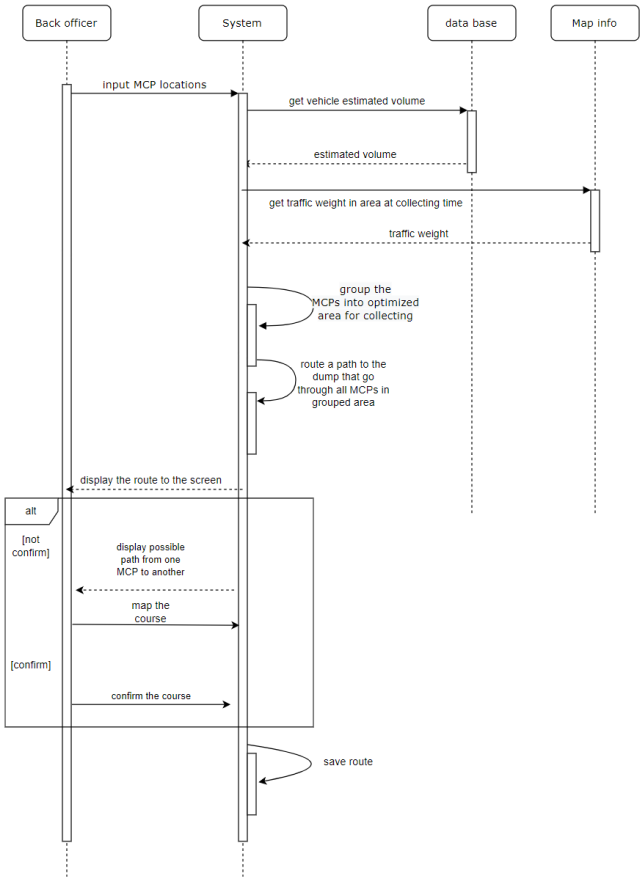
- Minimize the number of vehicles used and travel time.
- Balance the workload between vehicles.

Constraints:

- Vehicle and MCP capacity.
- Vehicle travel time.

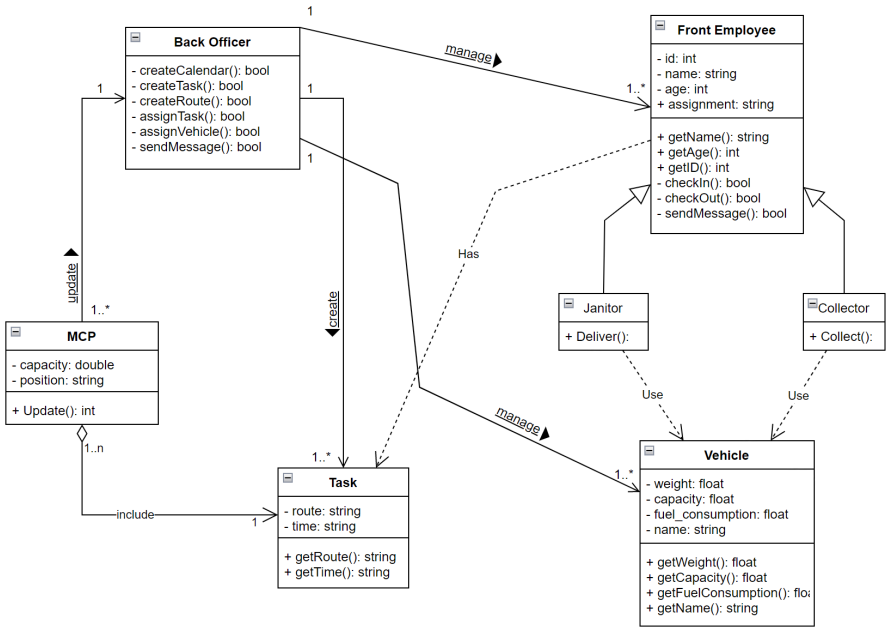
Solution:

- Based on the MCP locations, the system divides the map into different areas where MCPs are relatively close to each other.
- Then the system will form an optimal route connecting all full MCPs in that area that will take the least amount of traveling.



Sequence diagram for route planning task

3.3 Class diagram for Task Assignment module

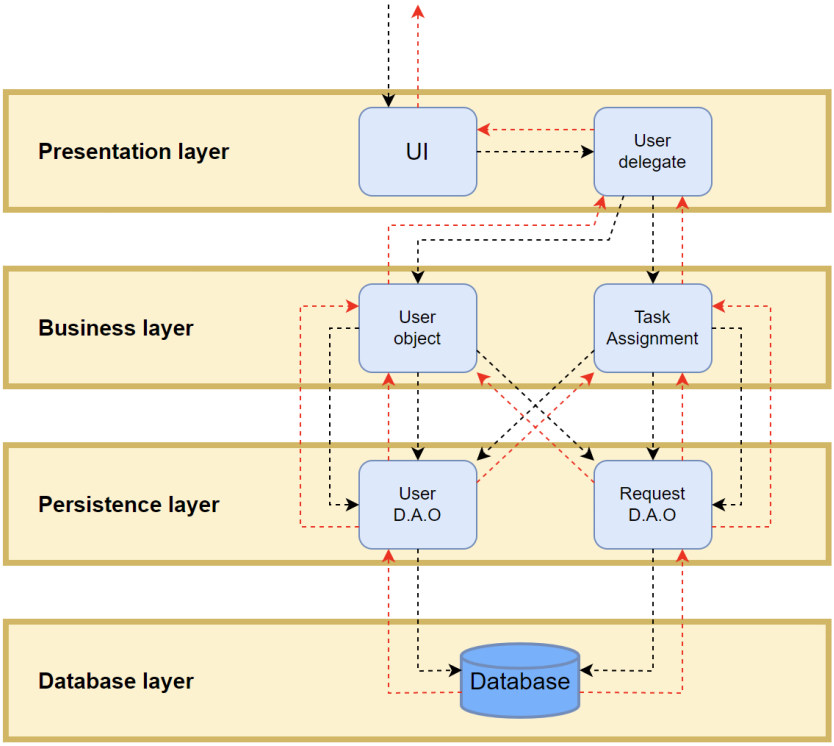


Class diagram for Task assignment module

4 Architecture Design

4.1 System Architecture

In order to implement the whole UWC 2.0 system, we will illustrate the system using layered architecture as below:



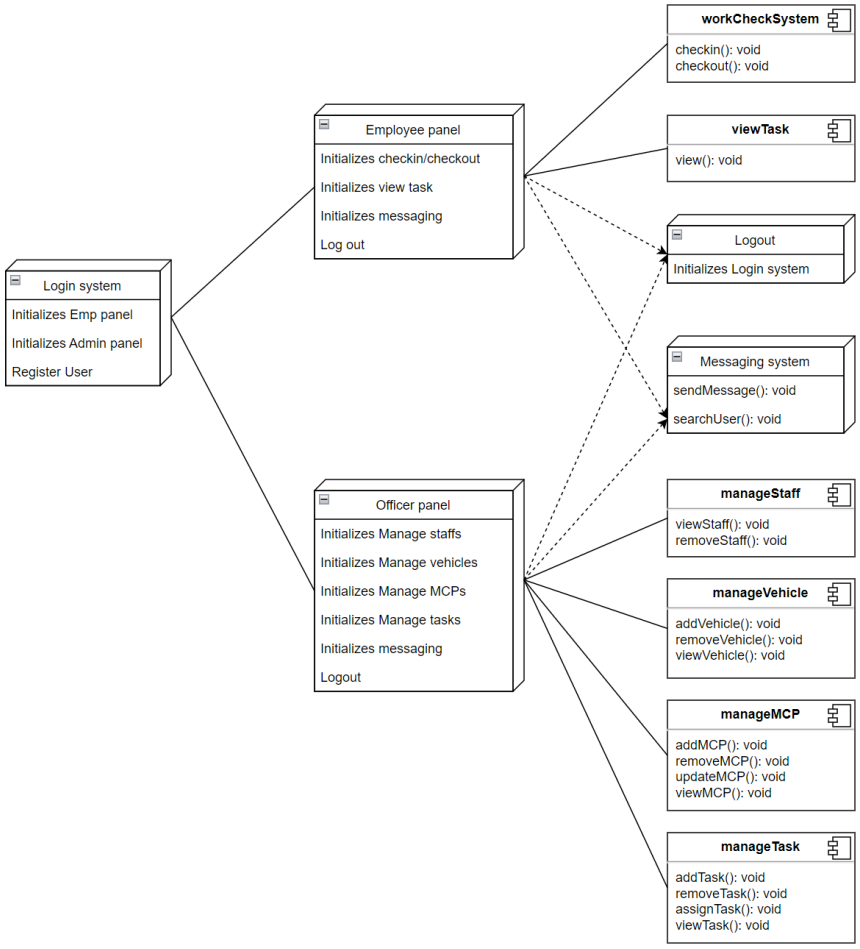
Layered architecture of the whole system

The black arrows represent the request flowing down to retrieve a user's specific desired data, and the red arrows represent the response flowing back to the UI.

The entire system will be broken down into 7 modules. The **UI** will receive the request of the user (back officer or front employee) as input and display the data to the user as its output. The **User delegate module** decides which module is needed to process the request and what data it needs. After receiving the request, the **User delegate module** then passes it down to either the **User object module** or the **Task Assignment** module based on the user's request. The **User object module** is used to retrieve all the information from the **Database** about staff, MCPs or vehicles if a back officer needs to view them in particular. The **Task Assignment module** is responsible for creating tasks and routes and assigning them to the front employees after getting information from the **Database** about employees' work schedule, available vehicles and MCPs. The **User Data Access Object** module receives requests from the **Business-Layer** modules and executes SQL queries to retrieve needed information from the

Database and the **Request Data Access module** executes the similar protocol to retrieve the information about the request. Once the execution is complete, the data will be passed through layers back to the **UI** to be displayed to the user.

4.2 Deployment diagram for task assignment module

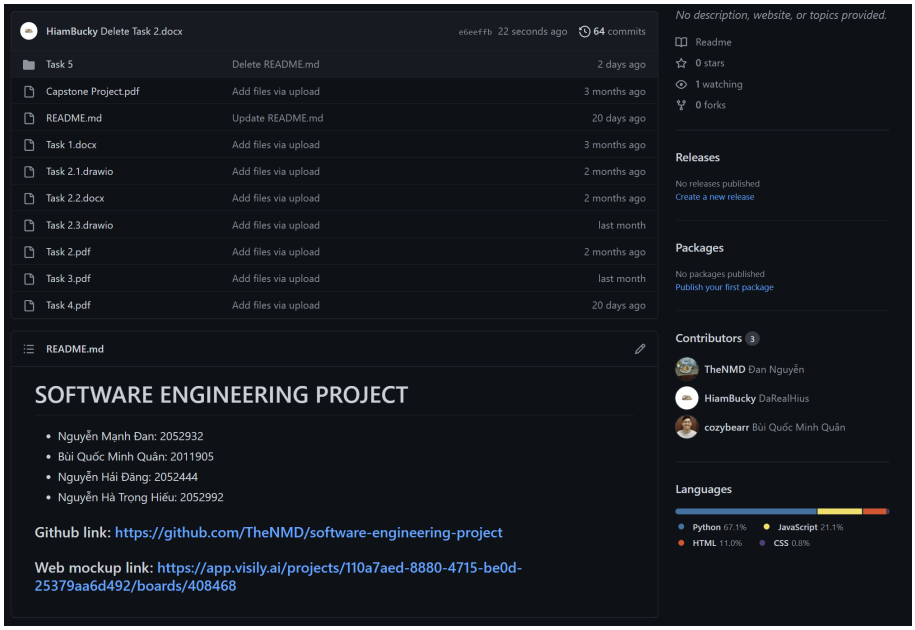


Deployment diagram for Task assignment module

5 Implementation - Sprint 1

5.1 Setting up online repository

Using github, we created an online repository for better version control. Each member then execute their task and upload/update the documents in that repository. The tasks will then be reviewed by the whole team 1 day before submission.



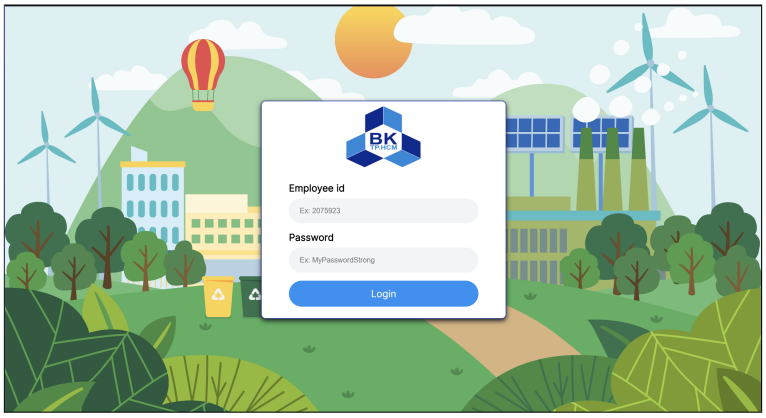
github repository

This is the link to the repository: Link to repository

5.2 Implement MVP1 - Interface design

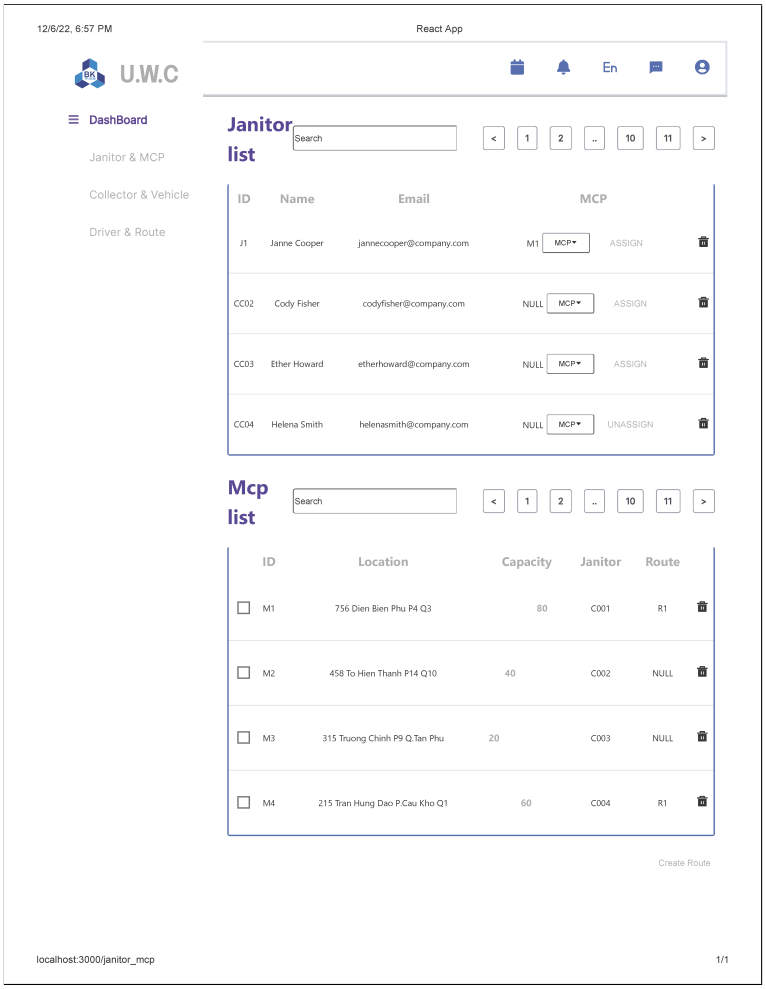
Using a tool called Visily, our team designed the interface, specifically a Desktop-view central dashboard for Task Management for back-officers.

First, you will see the Login page where you can login using employee ID and password.



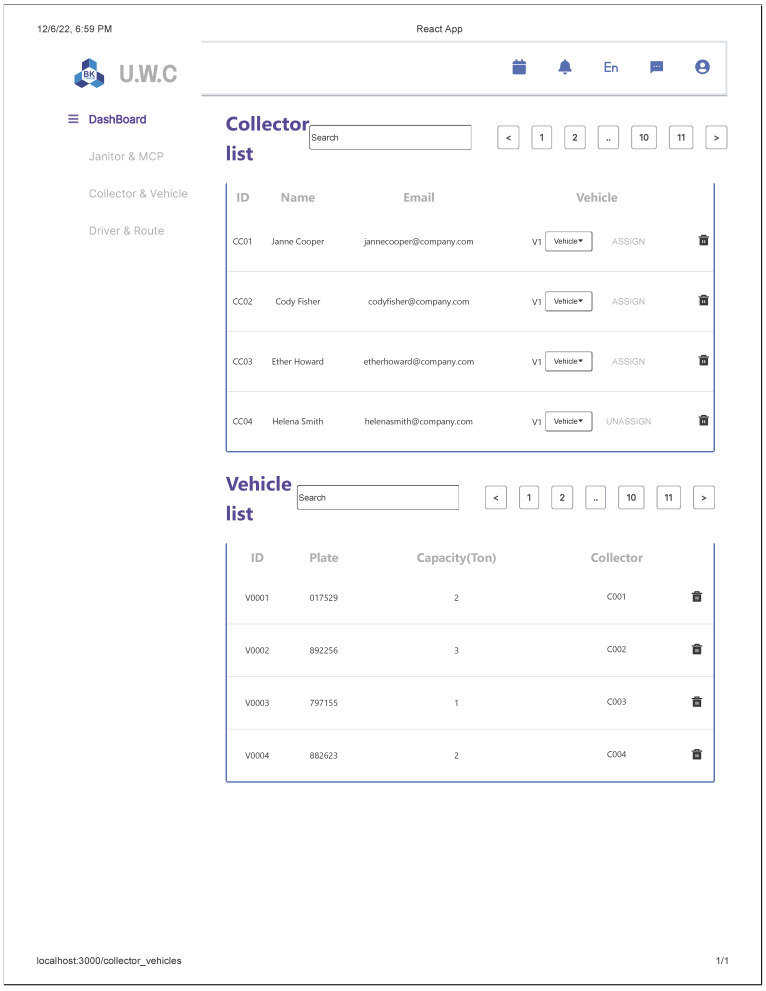
Login page

After logging in, you will first see the page displaying the list of Janitors and MCPs and their information, which then you can then assign the MCPs to a janitor who is available as well as unassigning someone.



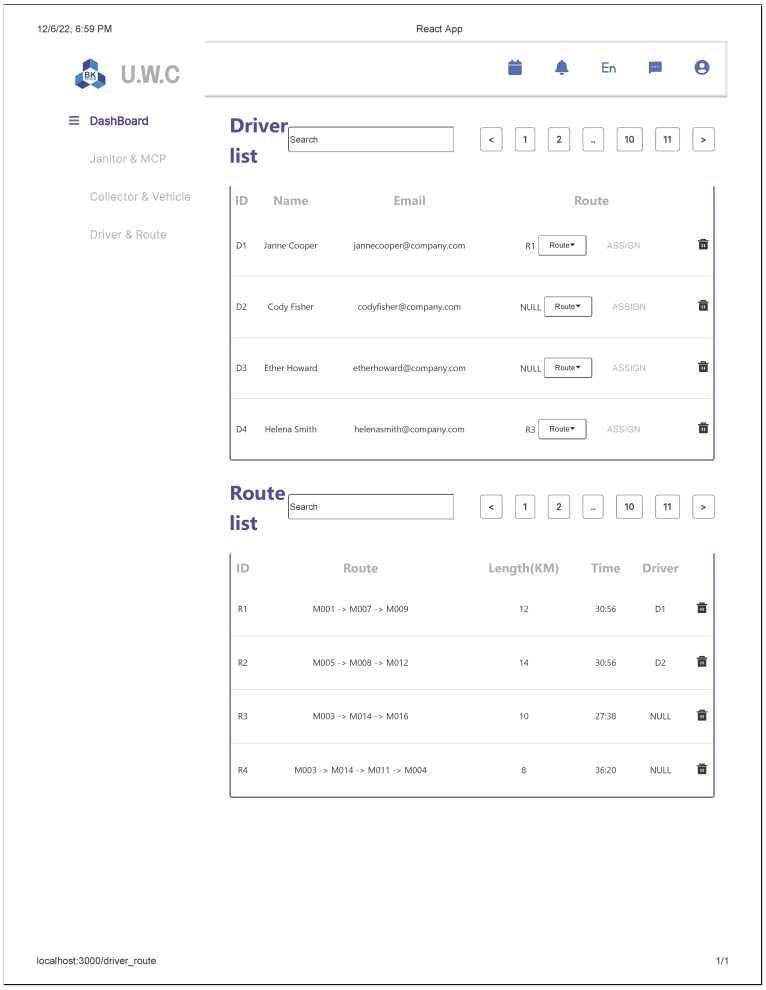
Janitors and MCPs

Next page is the list of Collectors and Vehicles with the same display pattern.



Collectors and Vehicles

Finally, the following page contains information of Drivers - which are assigned Collectors and their assigned routes.



Drivers and Routes

6 Implementation - Sprint 2

6.1 Implement MVP2 - Design realization

In order to realize the design from MVP1, we decided to use ReactJS due to the following benefits that it provides:

- **It is composable:** Composition is a function of combining parts or elements to form a whole. In the old days of web development a website was usually a single html page. So, a lot of time those web pages ended up being very long with thousands of lines of HTML codes. With modern frameworks like React, we can divide these codes and put it in custom components. Then we can utilize these components and integrate them into one place. Hence the code becomes a lot more maintainable and flexible. JSX is used for templating in React. JSX is a simple JavaScript that permits HTML quotation elements and uses these HTML tag syntax to render subcomponents.

- **It is declarative:** In react the DOM is declarative. We can make interactive UIs by changing the state of the component and React takes care of updating the DOM according to it. This means we never interact with DOM. Hence, it makes it easier to design UI and debug them. We can just change the program's state and see how the UI will look at that particular time. This makes our code more predictable and easier to debug.

- **Write once, learn anywhere:** We can develop new features in React without re-writing the existing code. It can also render on the server using Node and power mobile apps using React Native. So we can create IOS, Android, and Web applications simultaneously. In conclusion, extensive code reusability is supported by React.

- **It is simple:** The component-based approach, automatic rendering, and use of just plain JavaScript make React very simple to learn, build a web (and mobile applications), and support it. We can mix Javascript and HTML together to create a special syntax called JSX which makes it easier to grasp and work with it.

- **Fast, efficient, and easy to learn:** It contains pre-built patterns and functions that can be chosen and combined like building blocks to create fast, appealing, and scalable projects in less time as compared to designing the entire application line by line. Also, unlike Angular and Ember which are referred to as 'Domain-specific Language', React only requires need a basic knowledge of HTML and CSS fundamentals to start working with it.