

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



SOFTWARE ENGINEERING

Assignment

“URBAN WASTE COLLECTION – UWC 2.0”

Group: 4

MEMBERS:

Phạm Hoàng Minh – 2152771 – Contribution 100%

Tăng Tuấn Đạt – 2152512 – Contribution 100%

Tạ Gia Khang – 2152652 – Contribution 100%

Nguyễn Chánh Tín – 2153043 – Contribution 100%

TABLE OF CONTENTS

| | |
|--|-----------|
| Task 1: REQUIREMENT ELICITATION | 3 |
| 1.1 Relevant stakeholders and the benefits of UWC 2.0..... | 3 |
| 1.2 Functional and non-functional requirements..... | 4 |
| 1.3 Task assignment use-case diagram and table descriptions: | 6 |
| Task 2: SYSTEM MODELING | 14 |
| 2.1 Activity diagrams | 14 |
| 2.2 Sequence diagram..... | 17 |
| 2.3: Class diagram | 18 |
| 2.4: Figma MVP 1 of the desktop-view task assignment..... | 18 |
| Task 3: ARCHITECTURE DESIGN | 22 |
| 3.1 Layered Architecture Diagram..... | 22 |
| 3.2 Component diagram..... | 24 |
| Task 4: IMPLEMENTATION – SPRINT 1 | 25 |
| 4.1 & 4.2 GitHub Repositories | 25 |
| 4.3 Usability Test | 25 |
| Task 5: IMPLEMENTATION – SPRINT 2 | 30 |
| 5.1 MVP 2 | 30 |

Task 1: REQUIREMENT ELICITATION

1.1 Relevant stakeholders and the benefits of UWC 2.0

As a developing country, urban waste management is one of Vietnam's biggest challenges, especially in Ho Chi Minh city, where industrialization has led to a massive population density of 4375(1) people in a km². In addition, Ho Chi Minh city is riddled with small streets, underdeveloped roads, which means the waste disposal process is different from the first world countries. Big dumpster trucks cannot fit such roads, so the city has to resort to janitors with trollers to collect the garbage, then they will transport it to a Major Collecting Point (MCP), where collectors can collect using heavy vehicles and move them to a facility in the outskirts area. Attempts have been made to optimize such a system, noticeably the Urban waste collection aid or UWC 1.0, which helps to organize and centralize the procedure. However, as the amount of waste continue to grow, a remodeled version of UWC is more necessary, a new project called UWC 2.0.

The stakeholders of the UWC 2.0 project are back officers, collectors and janitors with the addition of a service provider (may from an internal or external source). All of the stakeholders are in need of a centralized system to help with communication, viewing the work calendar and checking-in at the start and the end of each shift/day. There are also special needs for each type of position. A back officer needs a way to create and manage the work calendar, the workflow of janitors and collectors. Janitors and collectors, on the other hand, need a way to view the said work calendar and the work assigned. This system needs to be fast and effective with a shallow learning curve to allow everyone to migrate from the previous system. In addition, a service provider or administrator needs an ability to easily transfer any data from the previous system and easily help any of the other employees in case technical problems arise.

UWC 2.0 will benefit all stakeholders by improving the efficiency and effectiveness of the waste collection process and providing real-time information and communication among stakeholders. It will also satisfy all of the needs of every stakeholder with an easy to follow with an easy-to-follow guide and user-friendly interface to help all workers to use this website in less than an hour. They can also save time by using the map to find the nearest non-full MCP for janitors or help collectors make informed decisions about their route by purposefully

skipping MCPs that are not full. All of this can be viewed and monitored by a back officer and in case of emergency, they can contact other workers.

1.2 Functional and non-functional requirements

1.2.1 Functional requirements

Back Officers:

- As a back officer, I want to have an overview of work calendar of janitors and collectors
- As a back officer, I want to have an overview of MCPs and capacity
- As a back officer, I want the MCPs' availability to be updated every 15 minutes
- As a back officer, I want to assign tasks weekly, vehicles and routes monthly
- As a back officer, I want to send messages/notices to individual or in general
- As a back officer, I want to send an announcement to a group of people
- As a back officer, I want to create route for collectors
- As a back officer, I want to assign area to janitors

Janitors:

- As a janitor, I want to have an overview of work calendar daily or weekly in one view
- As a janitor, I want to check out/mark my progress
- As a janitor, I want waste site to be near each other
- As a janitor, I want to check-in and check-out at the start and end of every shift
- As a janitor, I want to send messages/notices to others

Collectors:

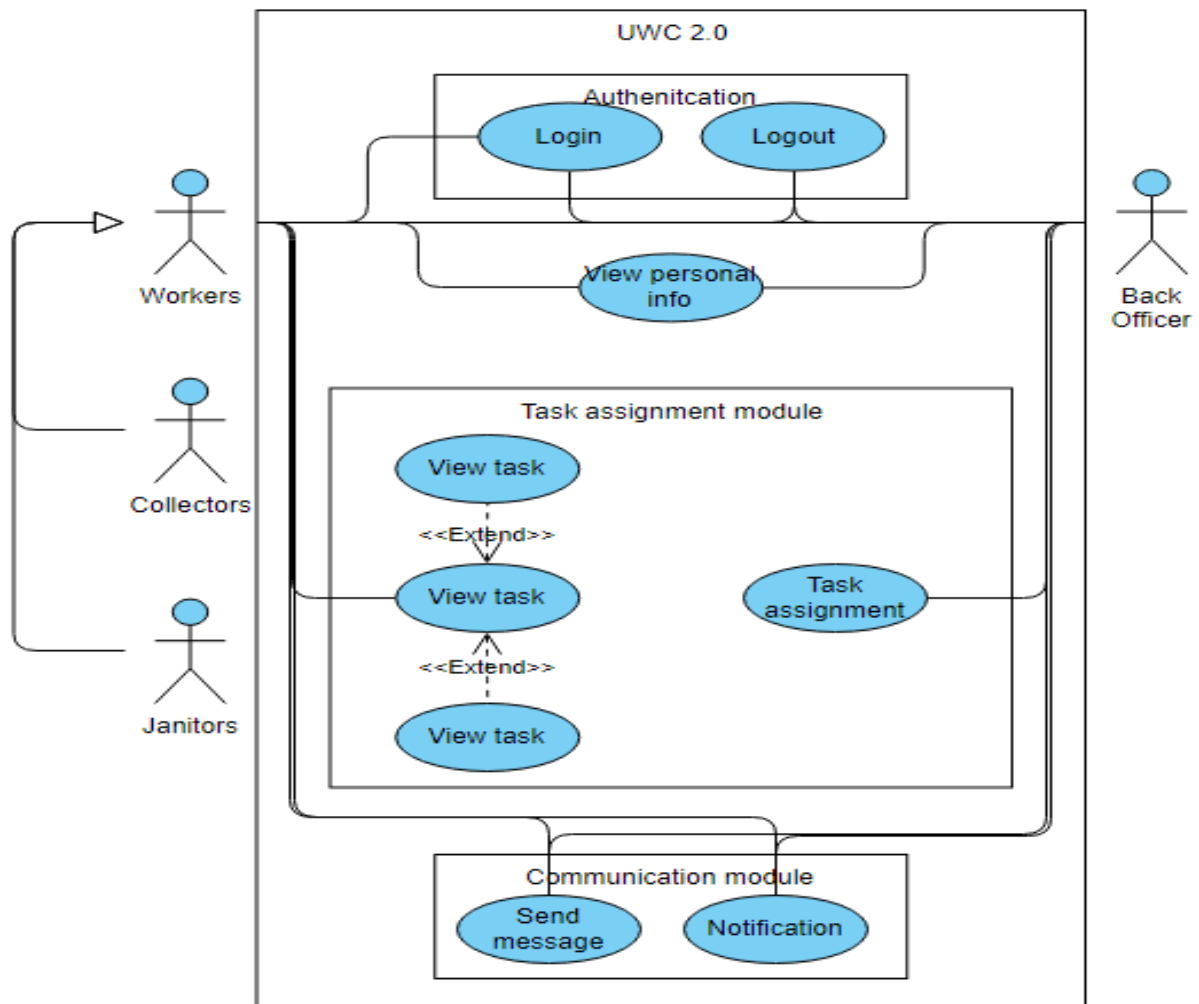
- As a collector, I want the optimal route among MCPs
- As a collector, I want to have an overview of the routes.
- As a collector, I want to check-in and check-out at the start and end of every shift
- As a collector, I want to view information about my assigned vehicle
- As a collector, I want to view the status of MCPs in my route

- The system allows employees to send messages to others

1.2.2 Non-functional requirements

- The system shall support seamless import and use of data from UWC 1.0.
- The system shall support inter-operability with UWC 1.0.
- The system shall ensure navigation with delays of no more than 1.5 seconds.
- The system shall ensure message delays of no more than 1 second.
- The system shall handle real-time data from at least 1000 MCPs at the moment and 10000 MCPs in five years
- The system shall support Vietnamese with the possibility of English in the future.
- The system shall require a maximum of two hours of training for users to operate.
- The system shall guarantee 99% up-time and limit downtime to less than 5 seconds per occurrence.
- The system shall ensure sanitization of all inputs.

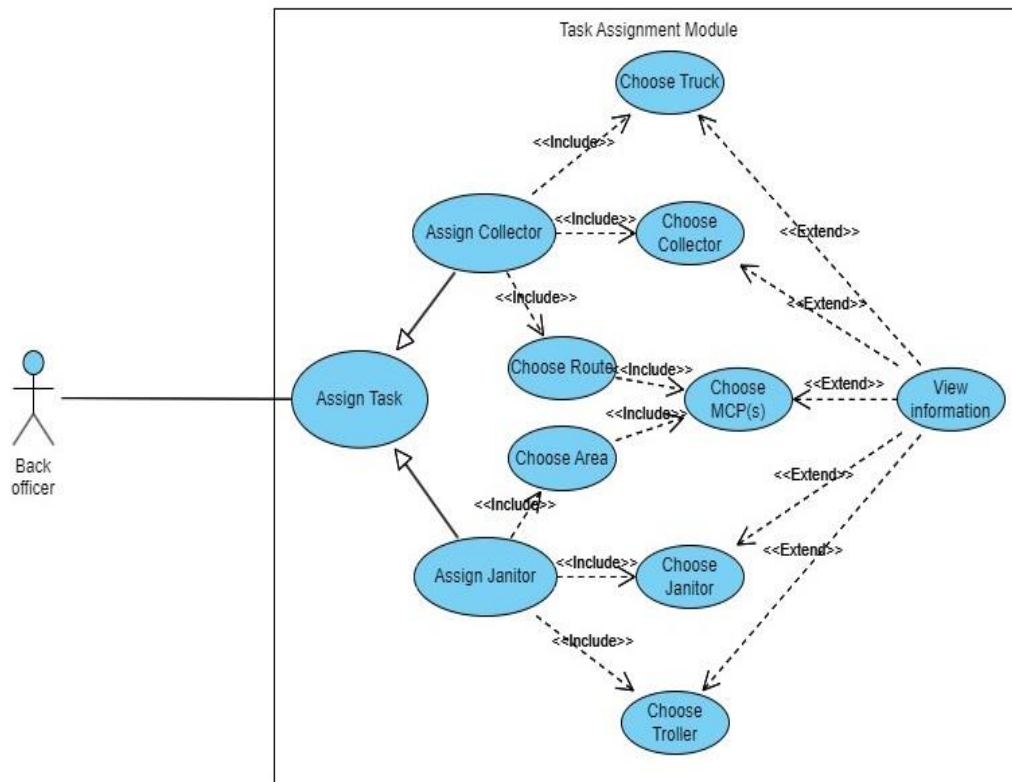
1.2.3 General use-case diagram



1.3 Task assignment use-case diagram and table descriptions:

1.3.1 Use case diagram

1.3.2 Table descriptions:



- To Collector:

| | |
|-------------------|--|
| Use-case name | Assign MCPs. |
| Use-case overview | To check and assign suitable MCPs to workers. |
| Actors | Back officers. |
| Preconditions | 1. The system is running. 2. The database is connected to MCPs. 3. Internet connection is available. |
| Trigger | Users click the "Assign MCPs" button. |
| Steps | 1. Retrieve all MCP's information from the database. |

| | |
|-----------------|--|
| | <p>2. Display a list containing suitable MCPs which have analyzed vacancy after previous assignment.</p> <p>3. Overview the MCPs which the users click and ask if they are chosen.</p> <p>4. Update the chosen ones to the worker's schedule weekly.</p> |
| Post conditions | Update the database and display the worker's schedule on the screen. |
| Exception flow | If there are no available MCPs, return to the homepage and send a notification to users. |

| | |
|-------------------|--|
| Use-case name | Assign vehicles. |
| Use-case overview | To check and assign suitable vehicles to collectors. |
| Actors | Back officers. |
| Preconditions | <p>1. The system is running.</p> <p>2. The database is connected to vehicle storage.</p> <p>3. Internet connection is available.</p> |
| Trigger | Users click the "Assign vehicles" button. |
| Steps | <p>1. Retrieve all vehicles' information from the database.</p> <p>2. Display a list containing suitable vehicles which have good status and are near to the working area of the collector.</p> <p>3. Overview the vehicles which the users click and ask if they are chosen.</p> <p>4. Update the chosen ones to the worker's monthly schedule.</p> |
| Post conditions | Update the database and display the worker's schedule on the screen. |

| | |
|----------------|--|
| Exception flow | If there are no available vehicles, return to the homepage and send a notification to users. |
|----------------|--|

| | |
|-------------------|--|
| Use-case name | Create a route. |
| Use-case overview | Create a suitable working route for the collectors. |
| Actors | Back officers. |
| Preconditions | <ol style="list-style-type: none"> 1. The system is running. 2. The database is connected to map and MCPs. 3. Internet connection is available. 4. Should have at least one MCP in the schedule of the collector. |
| Trigger | Users click the “Create a route” button. |
| Steps | <ol style="list-style-type: none"> 1. Retrieve all MCP’s information from the database. 2. Compute and find routes for the collector which has no conflict with the others and optimized in terms of fuel consumption and travel distance. 3. Display the routes on the screen, suggest the best one. 4. Overview the routes which the users click and ask if they are chosen. 5. Update the chosen ones to the worker’s weekly schedule. |
| Post conditions | Update the database and display the worker’s schedule on the screen. |
| Exception flow | If there are no available routes, return to the homepage and send a notification to users. |

- To Janitor:

| | |
|-------------------|---|
| Use-case name | Assign task |
| Use-case overview | To assign tasks to either janitors or collectors |
| Actors | Back officers |
| Preconditions | <ol style="list-style-type: none"> 1. The system is running 2. Internet connection is available 3. The back officer has login |
| Trigger | Back officer click the “Assign task” button |
| Steps | <ol style="list-style-type: none"> 1. Give option to display list of either janitors or collectors 2. Retrieve data on the database 3. Display the list of choice on the screen 4. Back officer goes through steps: “view worker’s information”, “assign area”, “assign troller” and “assign MCPs” 5. A confirmation message pops up on screen 6. The back officer presses to confirm |
| Post conditions | The task is assigned to the corresponding individual and other information is updated to the database. A message is sent to notice the chosen individual |
| Exception flow | None |

| | |
|-------------------|--|
| Use-case name | View worker’s information |
| Use-case overview | To view worker’s general information before assigning task to them |

| | |
|-----------------|--|
| Actors | Back officers |
| Preconditions | <ol style="list-style-type: none"> 1. Back officer has chosen either assign task to janitor or collector 2. The list of employees has shown on the screen 3. Back officer has determined the individual to receive task |
| Trigger | Back officer click on the individual's name on the list |
| Steps | <ol style="list-style-type: none"> 1. Retrieve their information from the database 2. Display the chosen individual's availability and their work calendar 3. Update the information on screen every 15 minutes in case other back officers has just assigned task to the janitor |
| Post conditions | Required information is displayed on the screen of users' devices and is easy to read |
| Exception flow | <ol style="list-style-type: none"> 1. The worker is unavailable for the day (with permission) 2. Display the unavailable status on screen 3. Hide option to assign task 4. The back officer tap the left arrow on top left to go back to previous list |

| | |
|-------------------|--|
| Use-case name | Assign area |
| Use-case overview | To assign area for janitor to do their work |
| Actors | Back officers |
| Preconditions | <ol style="list-style-type: none"> 1. The janitor is available for the day in "view worker's information" 2. The back officer has chosen the janitor to receive task |

| | |
|-----------------|---|
| Trigger | Users click the “Assign area” after viewing the janitor’s availability |
| Steps | <ol style="list-style-type: none"> 1. Retrieve the map with all the areas need cleaning from the database 2. Display the information on the user screen 3. Update the area in colors that show status: has none janitors, has more than 2 janitors, has more than 4 janitors 4. Back officer choose the suitable area and assign task |
| Post conditions | The task is selected for the janitor |
| Exception flow | None |

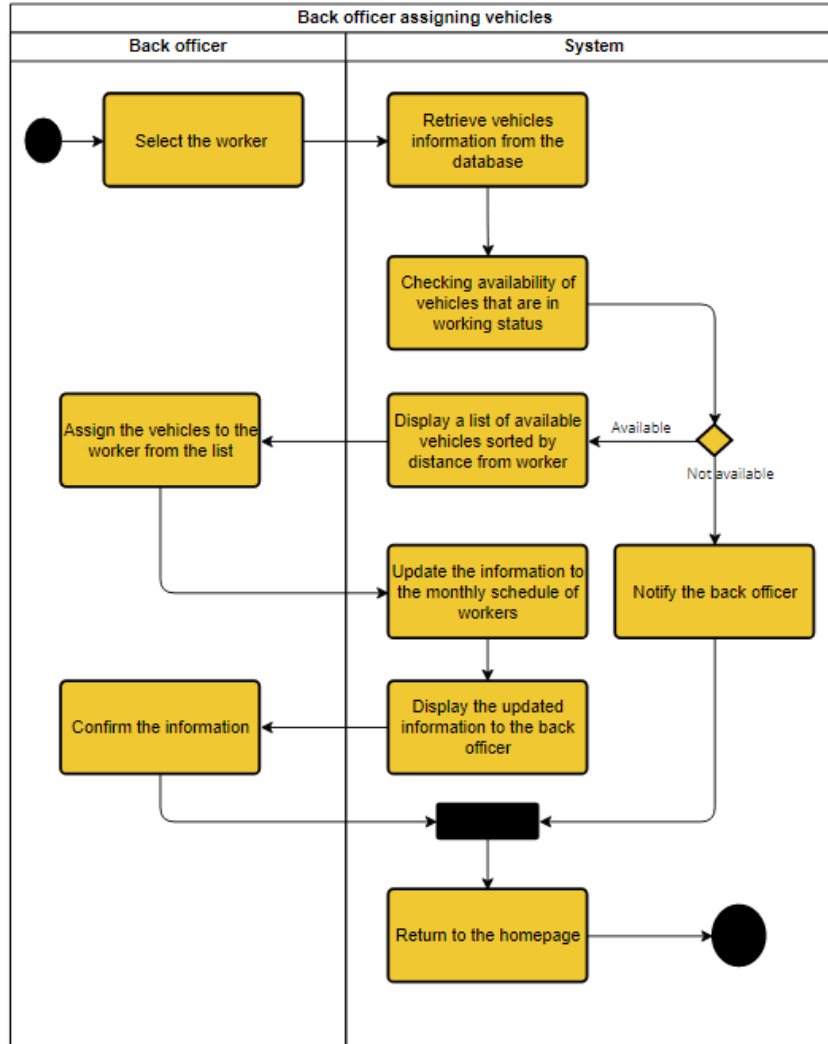
| | |
|-------------------|---|
| Use-case name | Assign troller |
| Use-case overview | To assign troller to janitor for their work |
| Actors | Back officers |
| Preconditions | <ol style="list-style-type: none"> 1. Back officer has chosen a janitor to receive task 2. Back officer has assigned the working area to the janitor |
| Trigger | Users has chosen the working area for the janitor |
| Steps | <ol style="list-style-type: none"> 1. Retrieve all the trollers’ information 2. Pop up a window for the back officer to select troller from 3. The list is sorted according to the nearest distance to the chosen working area 4. The back officer choose a troller for the janitor |

| | |
|-----------------|--|
| Post conditions | A troller is selected for the janitor |
| Exception flow | <ol style="list-style-type: none"> 1. Out of available trollers 2. Abort the assign task, area of the janitor 3. Return back to the main screen |

Task 2: SYSTEM MODELING

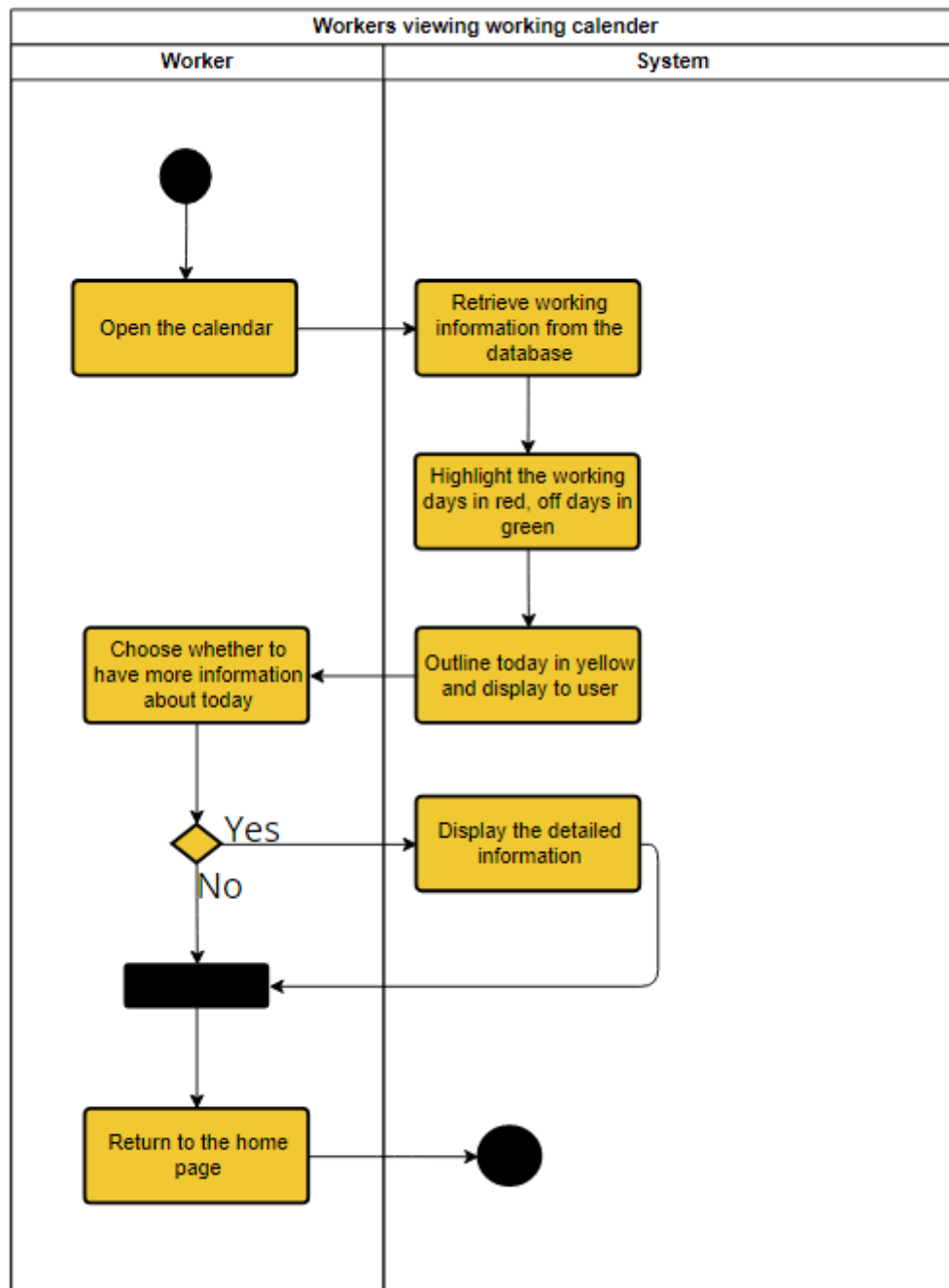
2.1 Activity diagrams

2.1.1 Back officer assigning vehicles

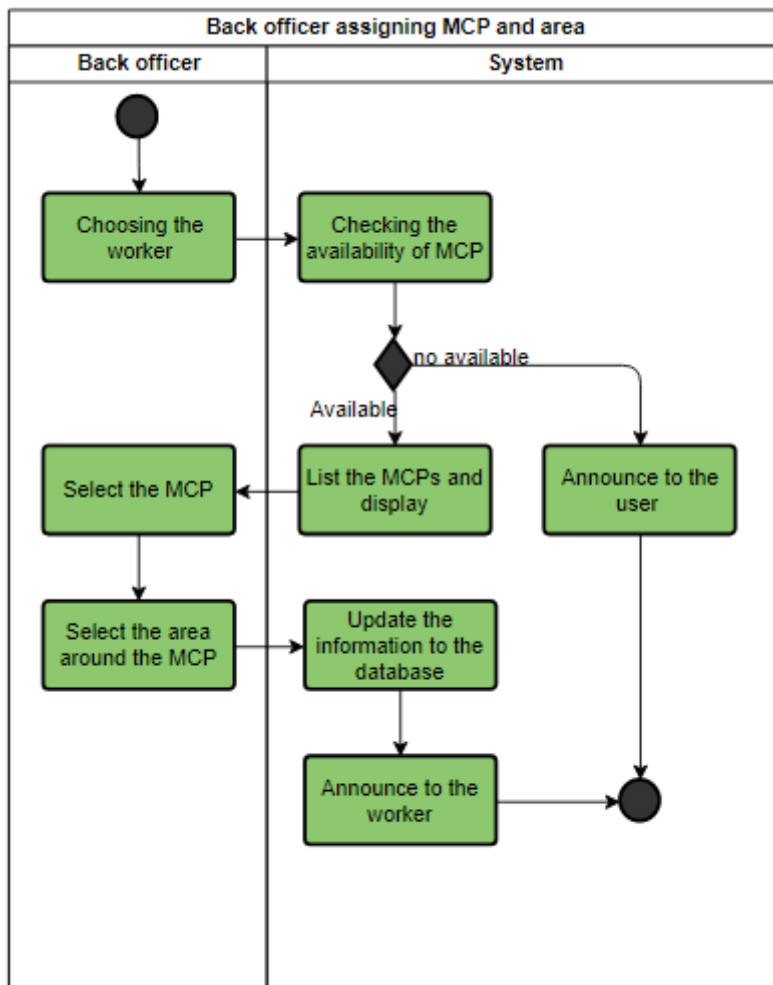


This use-case involves the process of assigning vehicles to sanitization workers for the month. The process starts with the back officer selecting the worker to assign a vehicle to, and ends with the user being returned to the homepage. The type of vehicle will be automatically assigned base on the role of each user. The system retrieves information about the availability of vehicles from the database, and if no vehicle is available, the system notifies the back officer and returns to the homepage. However, if there are vehicles available, the system displays a list of vehicles sorted by the distance from the worker. The back officer selects a suitable vehicle for the worker, and the information is sent to the

database. The information is then sent back for a final inspection, and if the back officer wants to make any changes at this step, they will have to redo the use-case.



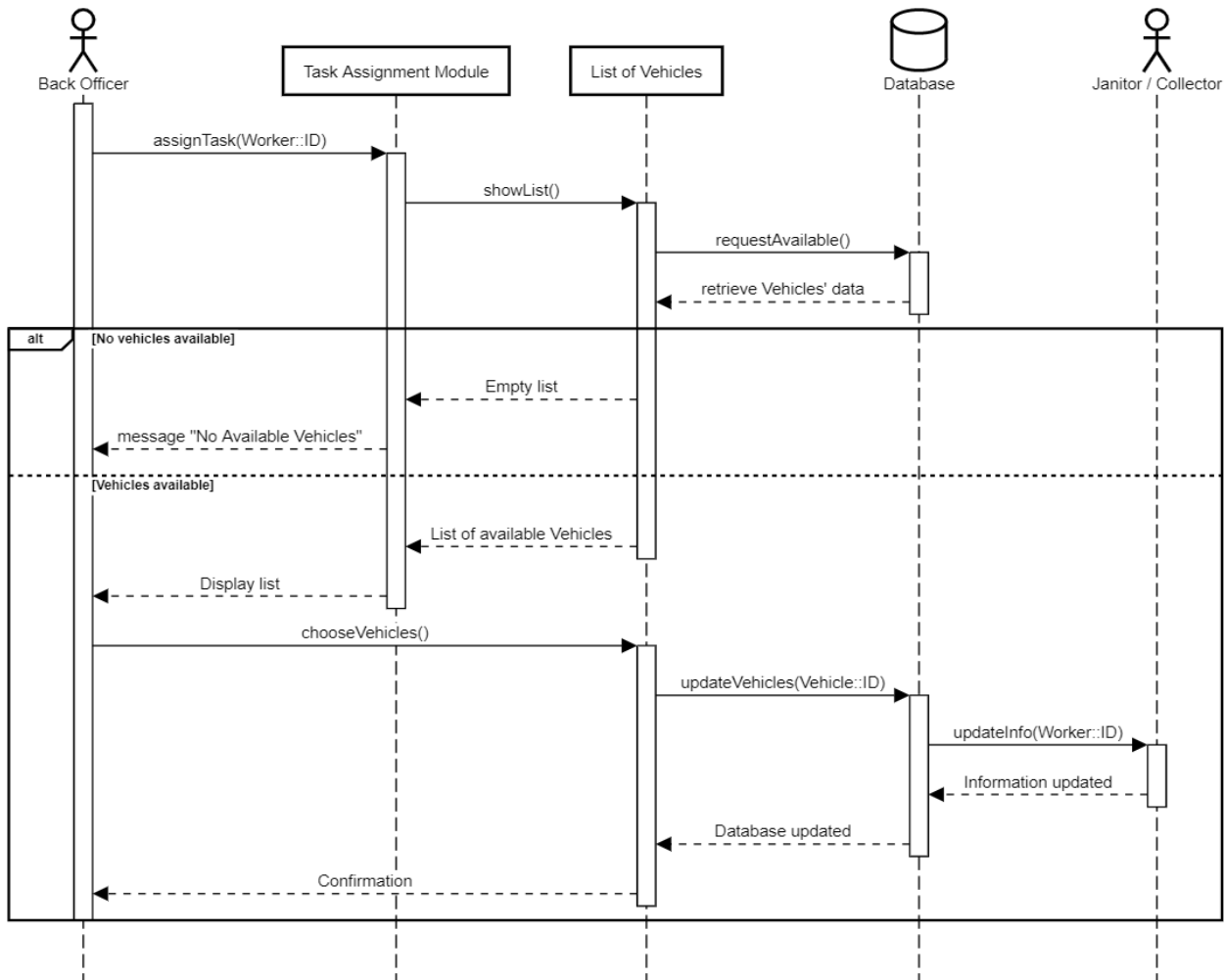
At the start of this use-case, the user is on the homepage of the system. They can access the calendar by clicking on it. Upon opening the calendar, the system will retrieve all the essential information related to the user's schedule such as working days, off days, and highlight the current day with a yellow outline. This information will be displayed to the user. If the user wants to view more information about a particular day, they can click on that day, or they can choose to return to the homepage after viewing the information.



This use-case involves the process of assigning MCP and the area. The process starts with the back officer selecting the janitor to assign a MCP to, and ends with the user being returned to the homepage. After the back officer prompt the user for the availability of the MCPs, the systems will check and display the available MCPs (not out of service or already pre-assigned). If there are none available, the system will announce to the user and return to the homepage. Otherwise, the user will select a MCP from the list and select the area around the MCP to assign to the janitor. Next, the information will be passed to the database to get updated and announce to the user. The module will return to the home page afterward.

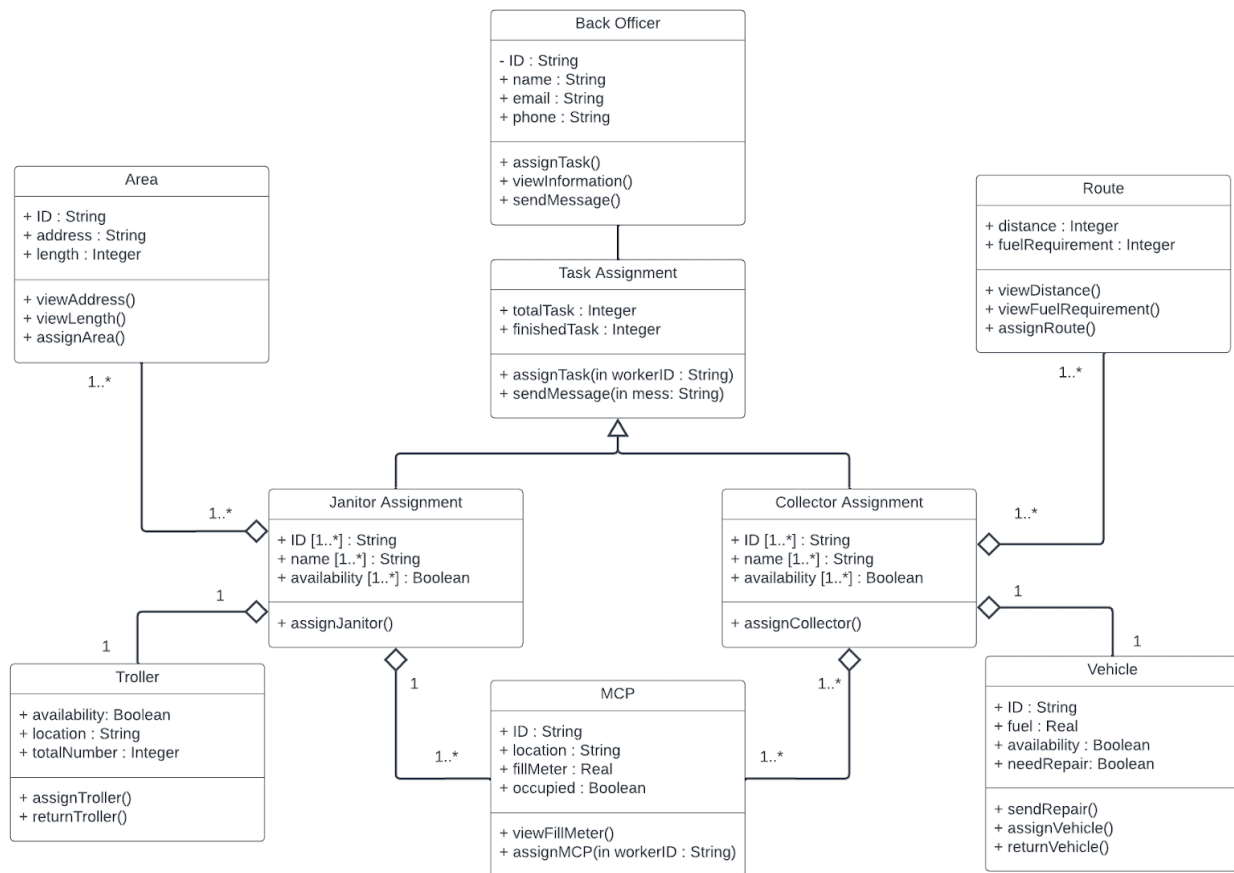
2.2 Sequence diagram

Use Case: ASSIGN VEHICLES TO WORKERS (Janitor/Collector)



The sequence diagram illustrates the process of assigning vehicles to workers. The process is initiated by the back officer who triggers the task assignment module by providing the Worker ID (choosing a worker). The module then decides the suitable type of vehicle and requests a list of available vehicles. If the list is empty, the module returns a "No Available Vehicles" message. However, if the list is not empty, the module returns the list of available vehicles to the back officer who selects the suitable vehicle for the janitor or collector. Once the vehicle is selected, the module updates the database and sends a message to the worker to announce the chosen vehicle. The module then sends a confirmation to the back officer about the assignment.

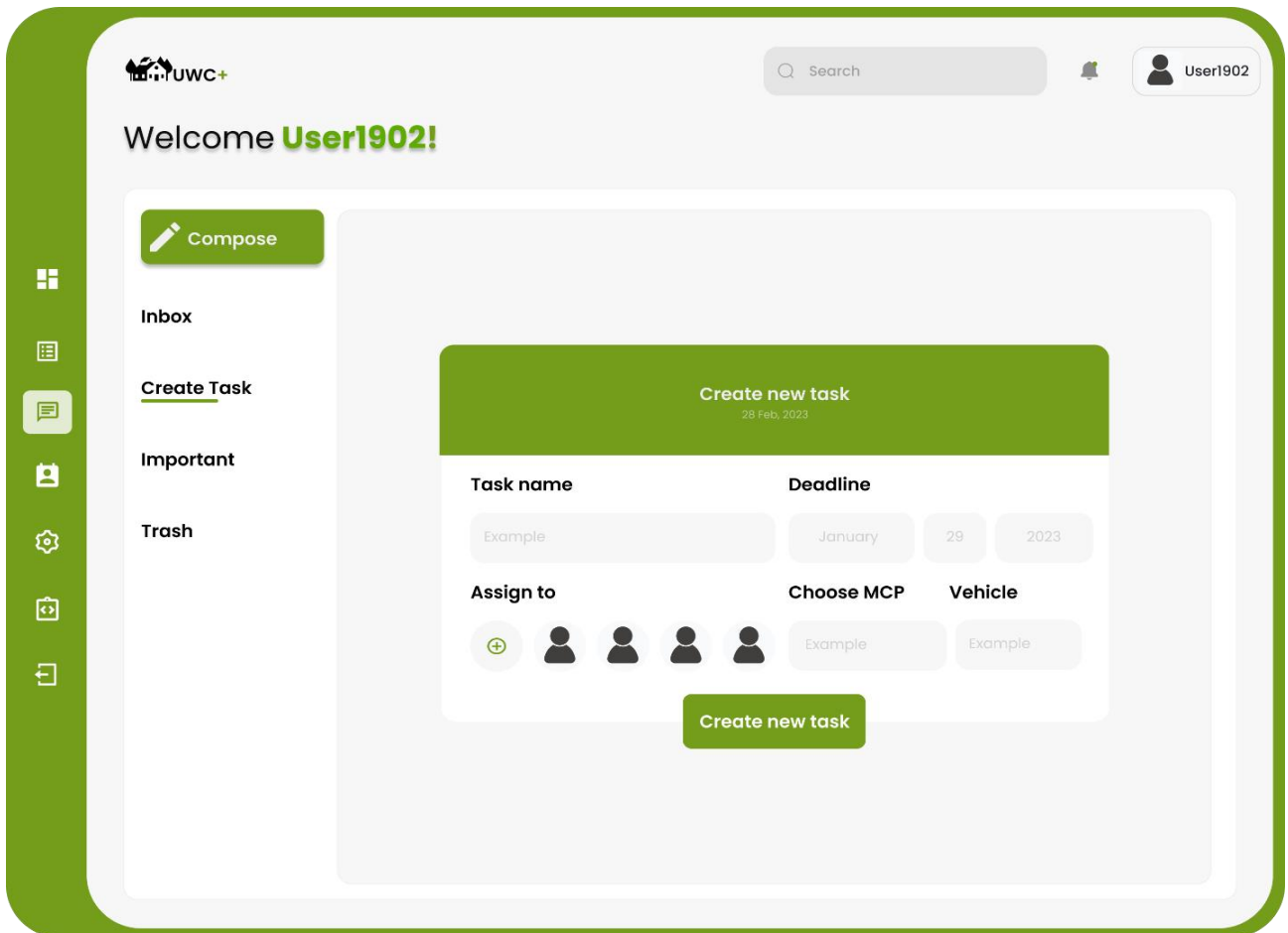
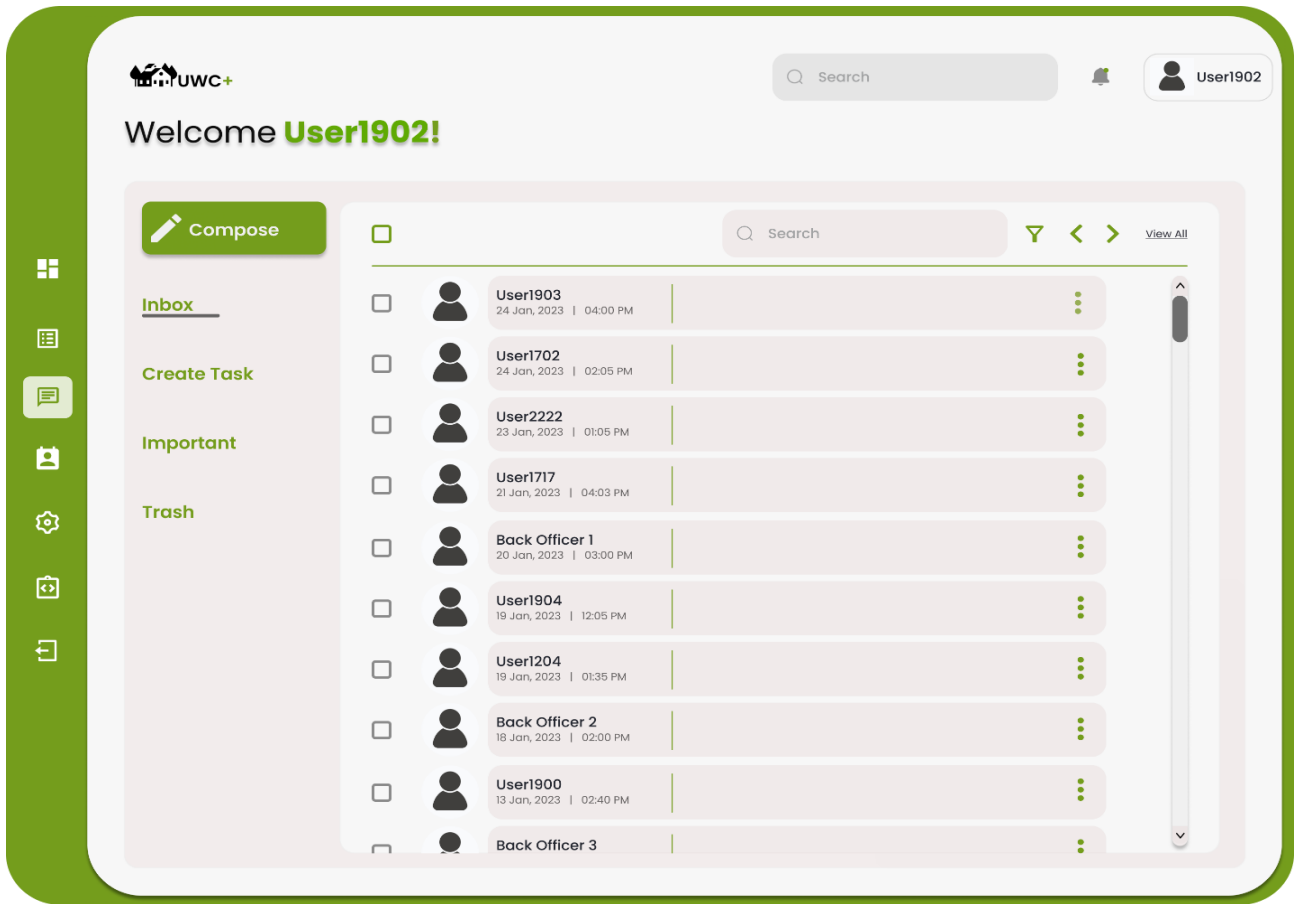
2.3: Class diagram



2.4: Figma MVP 1 of the desktop-view task assignment

[https://www.figma.com/file/6B9UuY4VdMjerQutLVJvp7/UWC-2.0-UI-\(Community\)?node-id=0%3A1&t=Tbi5YYFUNGIkZVxW-1](https://www.figma.com/file/6B9UuY4VdMjerQutLVJvp7/UWC-2.0-UI-(Community)?node-id=0%3A1&t=Tbi5YYFUNGIkZVxW-1)

The link above can be used to view the MVP in the functional state



Please login!

UWC+


UWC 2.0

Login

Welcome **User1902!**

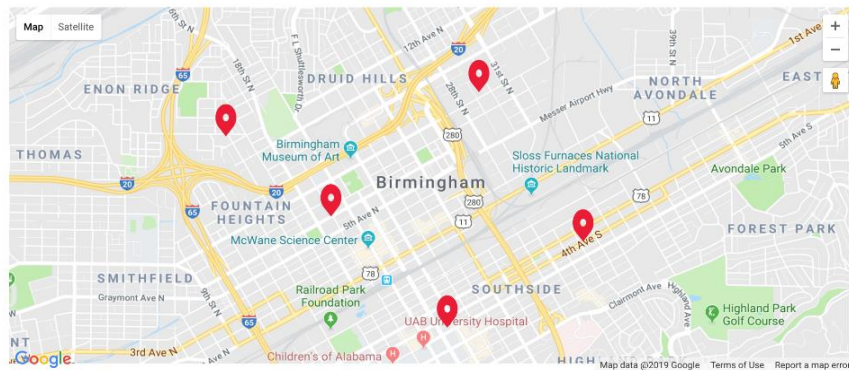
 Compose

Inbox

Create Task

Important

Trash



Assigned to

Route 1

Check

Assign

Save Route

Welcome **User1902!**

Recently Assigned Tasks 62

[View All](#)

- P** MCP1
25 Jan, 2023 | 04:00 PM
- P** MCP3
25 Jan, 2023 | 04:00 PM
- P** MCP2
25 Jan, 2023 | 04:00 PM

Recent Conversations

- User1204**
24 Jan, 2023 | 04:00 PM
- User9811**
24 Jan, 2023 | 02:05 PM
- Back Officer**
23 Jan, 2023 | 06:10 PM

Calendar

January 2023

| SUN | MON | TUE | WED | THU | FRI | SAT |
|-----|-----|-----|-----|-----|-----|-----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | | | | | |

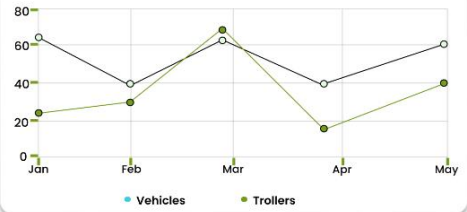
Daily Read

New rules in HCM city for UWC vehicles to move in rush hour.



Statistics

Last 6 Months ▼



Updates

[View All](#)

- Monthly's team meet**
30 Jan, 2023 | 04:00 PM



Completed Task 40

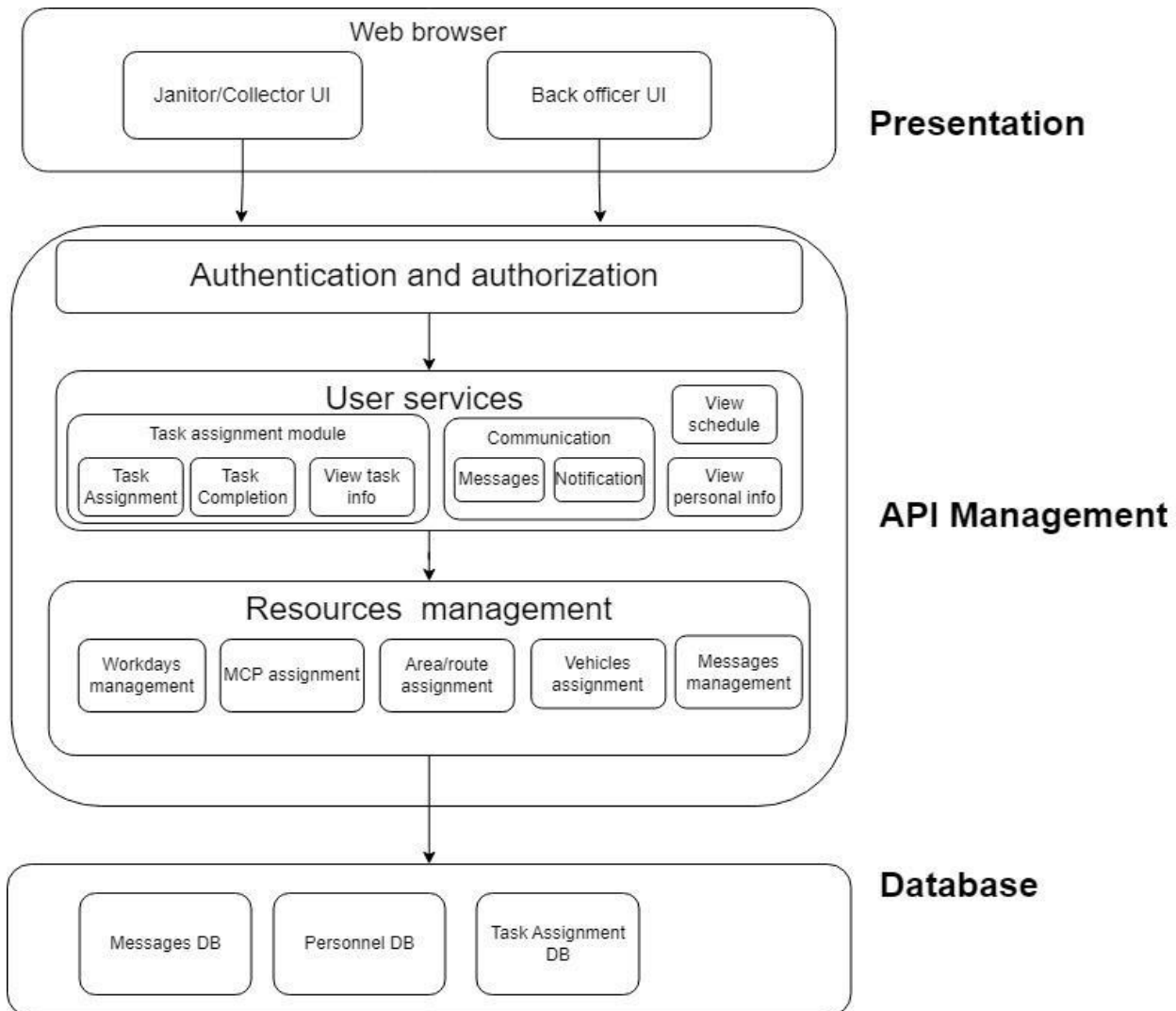
51%

New Employees 22

32%

Task 3: ARCHITECTURE DESIGN

3.1 Layered Architecture Diagram

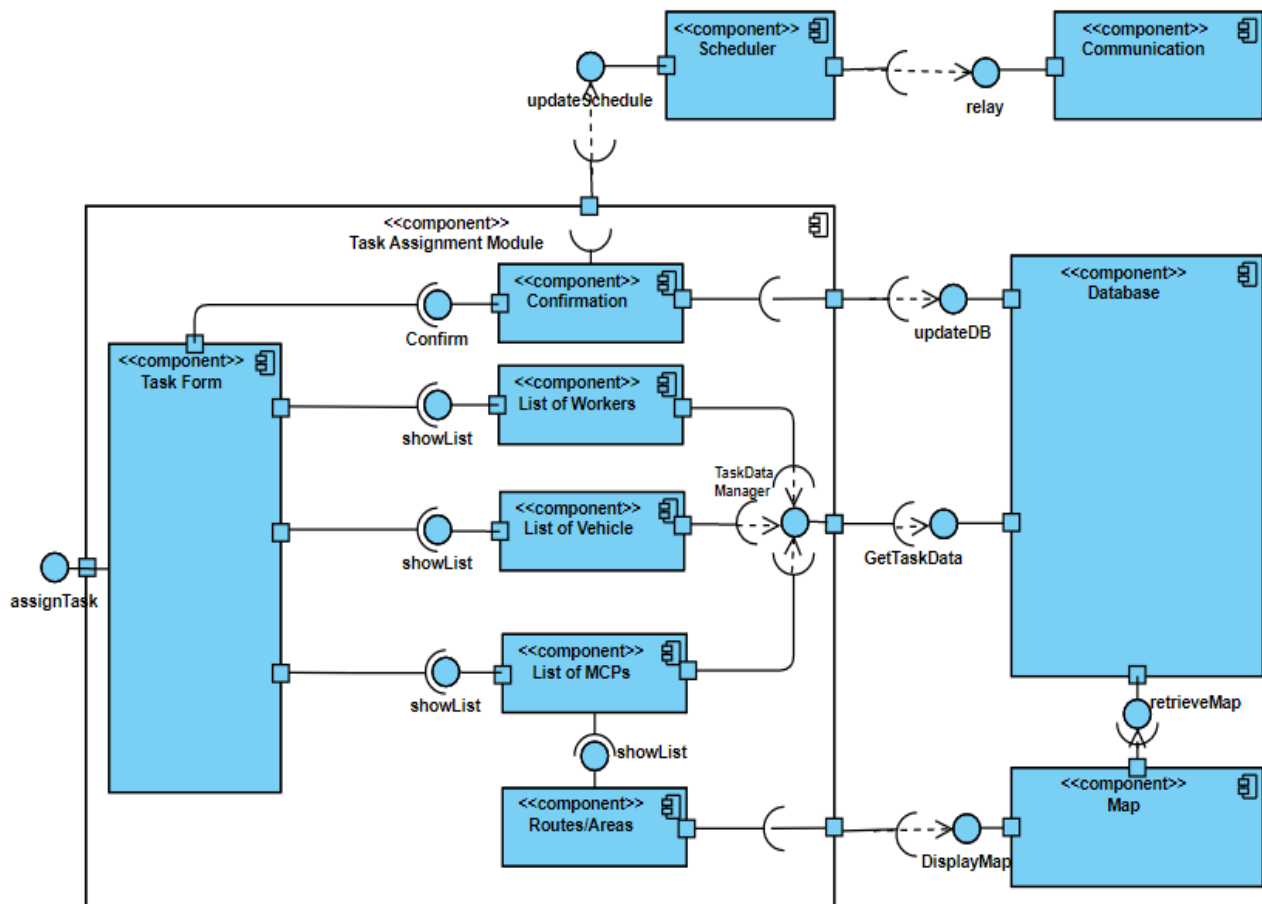


In the presentation layer, we will use a combination of HTML, CSS and TypeScript to design a user-friendly and aesthetically pleasing interface. The website shall support Vietnamese with the option to change between Vietnamese and English in the future using the “lang” attribute in HTML. UWC 2.0 also supports automatic scaling with different devices and works seamlessly with the zoom mode of major Internet Explorers. The information shall be presented in a layered approach with the fundamental tasks, announcements and info in a single view without having the user scrolling in most devices. The message module can be viewed on top of other content while task assignment has its own page. The Task Assignment Module will allow Back officers to choose workers, vehicles and MCPs (Route/Area), they may be shown on a map, to create the task.

In the API management layer, we will only use a few APIs like mapAPI (leaflet and leaflet-routing-machine) and MySQL, so there is no need to use a dedicated API management like Azure (in an ideal situation). The API management strategy for the UWC 2.0 system will be implemented using the Go programming language. Go provides a robust set of tools for building scalable and performant APIs, making it an ideal choice for the backend API layer and the Gin Framework in Go for the server and Gorm will be used to interact with the database. In addition, the API layer will be built using the REST architectural style, enabling the system to be easily integrated with other third-party APIs and services.

In the Database layer, MySQL will be used for the UWC 2.0 system. MySQL is a SQL document-oriented database that provides high scalability, performance, and flexibility, especially familiar because it is mainly introduced to those who are new to the Database. The Task Management module of the system will store all the relevant data related to janitors, collectors, vehicles, MCPs, and task assignments in a MySQL database. MySQL's document-oriented data model provides a flexible schema that can handle unstructured data, making it an ideal choice for a system with evolving data requirements. Additionally, MySQL's built-in replication and sharding features enable the system to scale horizontally as the number of MCPs and other data grows, providing high availability and fault tolerance.

3.2 Component diagram



The diagram depicts the interactions between different components in the task assignment module. The system revolves around the <<task form>>, which provides the UI allowing the user to choose and manage the essential <<lists>> in the process. The lists will be updated upon being prompted by the user, retrieving the information from the database component in the module, which acts as a gateway to access the information. The <<routes/areas component>> can only be accessed by the <<list of MCPs>> to avoid confusion when assigning tasks. The <<map>> can only be viewed by the user when choosing an MCP. Before updating the new information to the database, the user is asked to confirm for the final time. Then, the tasks are updated to the <<scheduler>> and announced to the assigned workers via the <<communication>>.

Task 4: IMPLEMENTATION – SPRINT 1

4.1 & 4.2 GitHub Repositories

GitHub Repository for Server: https://github.com/TangTuanDat/A2PK_UWC2.0

GitHub Repository for Client: <https://github.com/nct2309/uwc>

The documents (final report) is included in the Server repo.

4.3 Usability Test

1 Introduction and background

1.1 The reasons for conducting this test

In the last chapter, we have designed and implemented an MVP, which can perform basic tasks such as showing the UI and somewhat allowing the user to interact with the system. However, as we are not the users from the previous version of UWC, as well as not exposed to the strengths and weaknesses of it. So, our vision of the UWC 2.0 may be unrealistic and impractical with features that the actors do not need, while inconvenient or outright lacking the necessary features. In other words, we need users that used and are using the system to help us design and create the most suitable version of the website possible.

While the requirements, functional and non-functional, have already been established in Chapter 3, the “abstract” requirements like UI friendliness or ease of use are difficult to achieve without actual field testing and having the participants will help in the finalization process. In addition, this will allow us to rearrange and distributed the user interface in the way that is the most beneficial.

All the references are provided at the end of this report

1.2 Introduction

The purpose of this usability test is to gather feedbacks and opinions from participants on the user interface (UI) of the UWC 2.0 MVP. As we are still in the early stages of development, it is crucial to understand how users interact with the UI and what improvements can be made. To ensure a realistic user experience, participants were recruited who were not involved in the development process and had no prior knowledge of

the UWC 2.0. Through this test, we hope to gain insights on the UI's usability, user-friendliness, and overall effectiveness.

2 Methodology

At the start of the testing process, we planned to conduct a quantitative test to gather feedback about the MVP. However, the results from the initial test revealed that many issues persisted from UWC 1.0 due to the interface displaying either too much or too little information, as well as unnecessary features and hidden features. To obtain a more tailored experience for real users, we decided to include a qualitative test in our methodology.

During the quantitative test, participants were asked to complete a Google form to provide opinions on the aesthetic and predict the usability of the MVP. They were also asked to rate screenshots of the website based on the amount of information provided and whether it was too cluttered or not.

In the qualitative test, we selected five participants who had expressed strong positive or negative opinions about the MVP. We asked them additional questions to gain further insights and allowed them to interact with the MVP to gather feedback at each stage.

2.1 Quantitative testing:

For the quantitative testing, we obtained responses from 20 back officers who filled out a Google form. The questions focused on the aesthetics and usability of the MVP and allowed participants to rate these aspects on a scale of 1 to 5. Scores of 4 or 5 were considered successful, scores between 3 and 4 were passable, and scores below 3 required revisions. We used the built-in features of Google Forms to collect and analyze the data, creating graphs to illustrate the results. While we acknowledge that our sample size was relatively small, we believe that the test provided valuable insights and helped to engage officers in the improvement process by encouraging feedback

2.2 Qualitative testing

Based on the feedback gathered from the quantitative test, we conducted a qualitative test to obtain more detailed feedback on the user interface and user experience of the UWC 2.0 MVP. The aim of this test was to gather insights on the usability, user-friendliness, and overall effectiveness of the UI. We selected five participants for this test who had expressed

strong positive or negative opinions about the MVP in the quantitative test or through email correspondence with us.

During the qualitative test, participants were asked to perform specific tasks using the MVP, including accessing the task assignment module from the login page. The task traversal was timed and observed to identify any potential choke points. Additionally, participants were encouraged to provide feedback on the UI and aesthetics of the MVP, as well as ask any questions they had.

3 Results and analysis

3.1 lightweight-ness

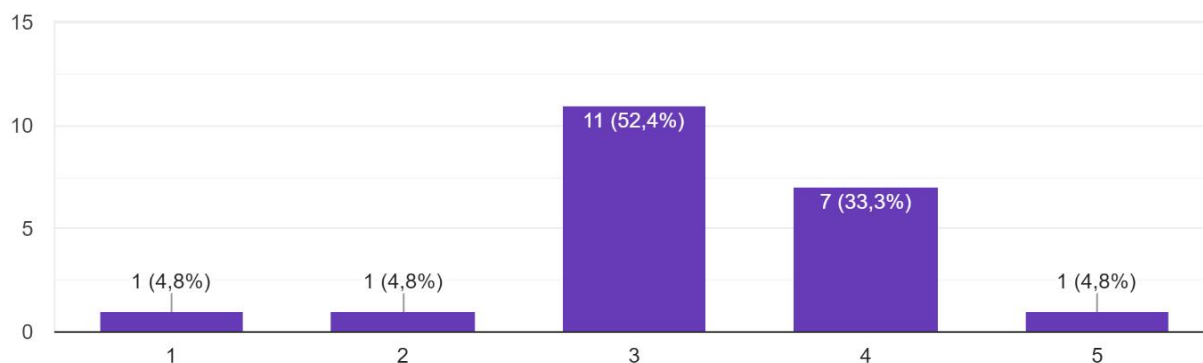
“I wish to love your website but it wouldn’t run well with my laptop”. One major issue that emerged from both the qualitative and quantitative testing was the lightweight-ness of the website. Several participants reported that the website was slow and unresponsive, particularly on older or lower-end devices. This feedback is significant as many back-officers, janitors, and collectors use the system on mobile devices, which requires a fast and easy-to-navigate system for them to check maps and sign-in for their day's work. In the quantitative test, over half of the participants did not feel entirely positive about the website's lightweight-ness, with a score of only 3.625 points, indicating a negative impact on the user experience. This feedback highlights the need for further optimization of the website's performance to improve usability.

3.2 Aesthetics and designs

Another issue that emerged from both the qualitative and quantitative testing was the aesthetics and design of the website. While some participants found the website easy to navigate, three out of five people who did the in-person test said that the website was too heavy, and the color scheme was too bold. This feedback was also reflected in the quantitative survey, where the initial view of the MVP received only 3.25 points. This can negatively affect the user's usability when performing tasks on the website. Therefore, the aesthetics and design of the website need to be improved to ensure a positive user experience.

The layout and colors is pleasing to see.

21 câu trả lời



Picture 1

3.3 Intuitiveness

In terms of intuitiveness, our goal for the next version of the UWC is to make it as self-explanatory as possible to facilitate a smooth transition for users. However, feedback from participants in the in-person test indicated that the MVP lacked a help guide to assist new or older users in navigating to their desired destination. Despite this, we were pleased to note that the majority of survey respondents were able to understand the purpose of the website based on their experience with UWC 1.0.

4 Recommendations and suggestions

Based on the feedback obtained from the survey and in-person testing, we have identified several recommendations to improve the user experience of the UWC website. Firstly, the website's lightweightness needs to be improved by optimizing it for both desktop and mobile views, reducing transitions between modules, and ensuring that the website can fit on most devices. Secondly, the website's design should be modified to ensure that it does not impede the workers' efficiency, particularly on mobile devices. Lastly, adding a support button at the top left of the screen can assist new or elder users in using the website effectively. These changes will help to improve the overall usability of the UWC website and ensure that it meets the needs of its users.

5 Appendix and references

This survey was conducted using Google Forms and followed the [guidelines](#). The qualitative testing questions were open-ended and focused on the following topics:

- Participant opinion on the current website (UWC 1.0)
- Participant opinion on the MVP and the online survey
- Hands-on use of the MVP, including ease of use and aesthetics
- Suggestions for improving the MVP.

Thank you to all participants for taking the time to provide valuable feedback. We also extend our gratitude to anyone who contributed to the project and helped make this survey possible.

Task 5: IMPLEMENTATION – SPRINT 2

5.1 MVP 2

5.1.1 Server

```
.../A2PK_UWC2.0 on server...
> go run .
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env:   export GIN_MODE=release
- using code:  gin.SetMode(gin.ReleaseMode)

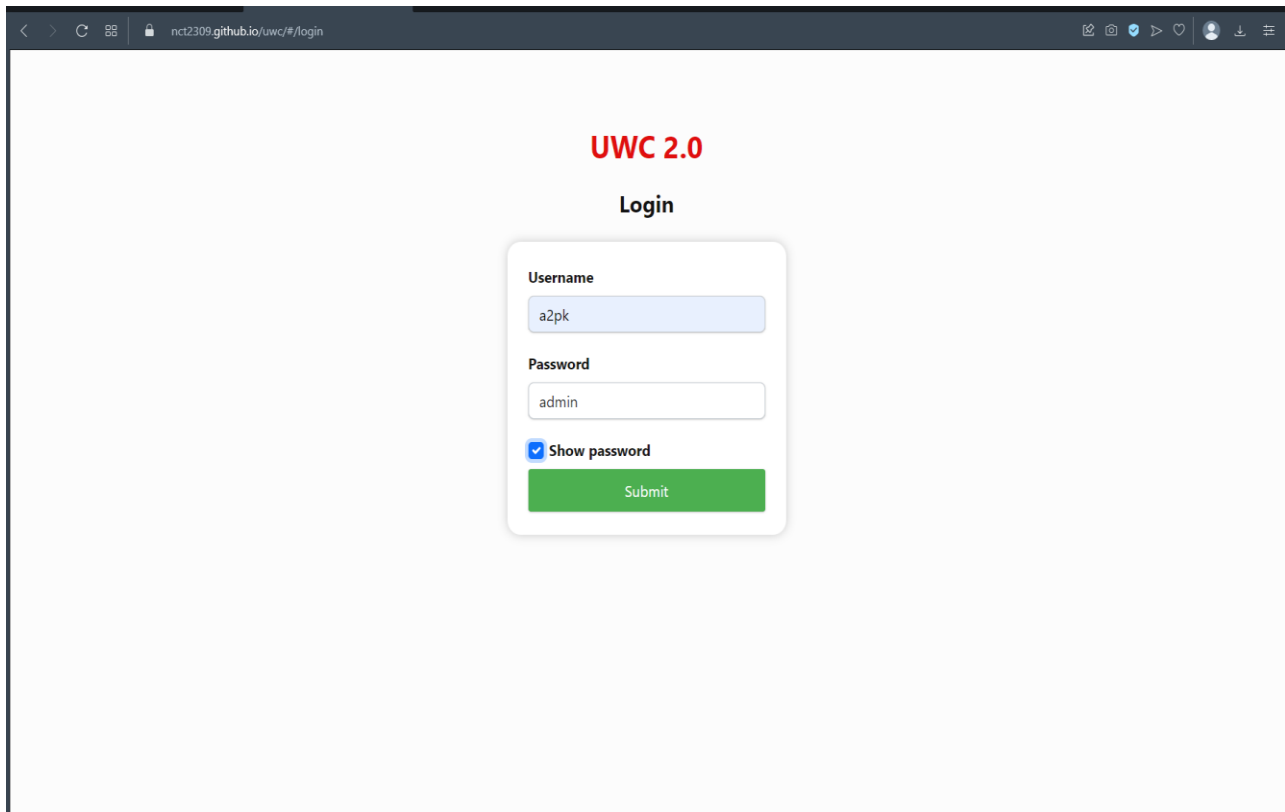
[GIN-debug] GET    /tasks/          → A2PK_UWC2.0/internal/controller.getListOfTasks.func1 (3 handlers)
[GIN-debug] GET    /tasks/:id       → A2PK_UWC2.0/internal/controller.readTaskById.func1 (3 handlers)
[GIN-debug] PUT    /tasks/:id       → A2PK_UWC2.0/internal/controller.editTaskById.func1 (3 handlers)
[GIN-debug] POST   /tasks/          → A2PK_UWC2.0/internal/controller.createTask.func1 (3 handlers)
[GIN-debug] POST   /users/          → A2PK_UWC2.0/internal/controller.createUser.func1 (3 handlers)
[GIN-debug] GET    /users/          → A2PK_UWC2.0/internal/controller.getUser.func1 (3 handlers)
[GIN-debug] GET    /users/:id       → A2PK_UWC2.0/internal/controller.readUserById.func1 (3 handlers)
[GIN-debug] PUT    /users/:id       → A2PK_UWC2.0/internal/controller.editUserById.func1 (3 handlers)
[GIN-debug] GET    /users/janitors  → A2PK_UWC2.0/internal/controller.getListOfJanitors.func1 (3 handlers)
[GIN-debug] GET    /users/collectors → A2PK_UWC2.0/internal/controller.getListOfCollectors.func1 (3 handlers)
[GIN-debug] GET    /trollers/       → A2PK_UWC2.0/internal/controller.getListOfTroller.func1 (3 handlers)
[GIN-debug] GET    /vehicles/       → A2PK_UWC2.0/internal/controller.getListOfVehicle.func1 (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on localhost:8080
[GIN] 2023/04/29 - 13:15:59 | 200 | 2.642713ms | 127.0.0.1 | GET | "/tasks/"
[GIN] 2023/04/29 - 13:16:00 | 200 | 2.666993ms | 127.0.0.1 | GET | "/users/"

[GIN] 2023/04/29 - 13:17:14 | 200 | 5.101859ms | 127.0.0.1 | GET | "/users/2"
[GIN] 2023/04/29 - 13:17:24 | 200 | 2.35182ms | 127.0.0.1 | GET | "/vehicles/"
[GIN] 2023/04/29 - 13:20:32 | 200 | 19.23308ms | 127.0.0.1 | POS | "/users/"
```

The chosen technology stack for developing the backend system appears to be well-suited for the intended purpose. Golang is a high-performance language that is particularly efficient in handling concurrent requests, making it an optimal choice for building scalable systems. MySQL, a well-established relational database management system, is also a sound choice as it can handle large amounts of data while ensuring data consistency and reliability. The Gin framework is an excellent choice for building the HTTP server as it is lightweight, fast, and can handle a high number of concurrent requests. The use of GORM as an ORM library provides a convenient and straightforward way to interact with the database, reducing the complexity of database operations. Furthermore, the application of the Clean Architecture principles in the code design is commendable. This methodology ensures separation of concerns, leading to better code organization, scalability, and testability. These benefits are particularly significant when developing complex systems where maintainability is essential. In summary, the selection of Golang, MySQL, Gin, and GORM, along with the use of the Clean Architecture principles, provides a strong foundation for developing a high-performance and scalable backend system.

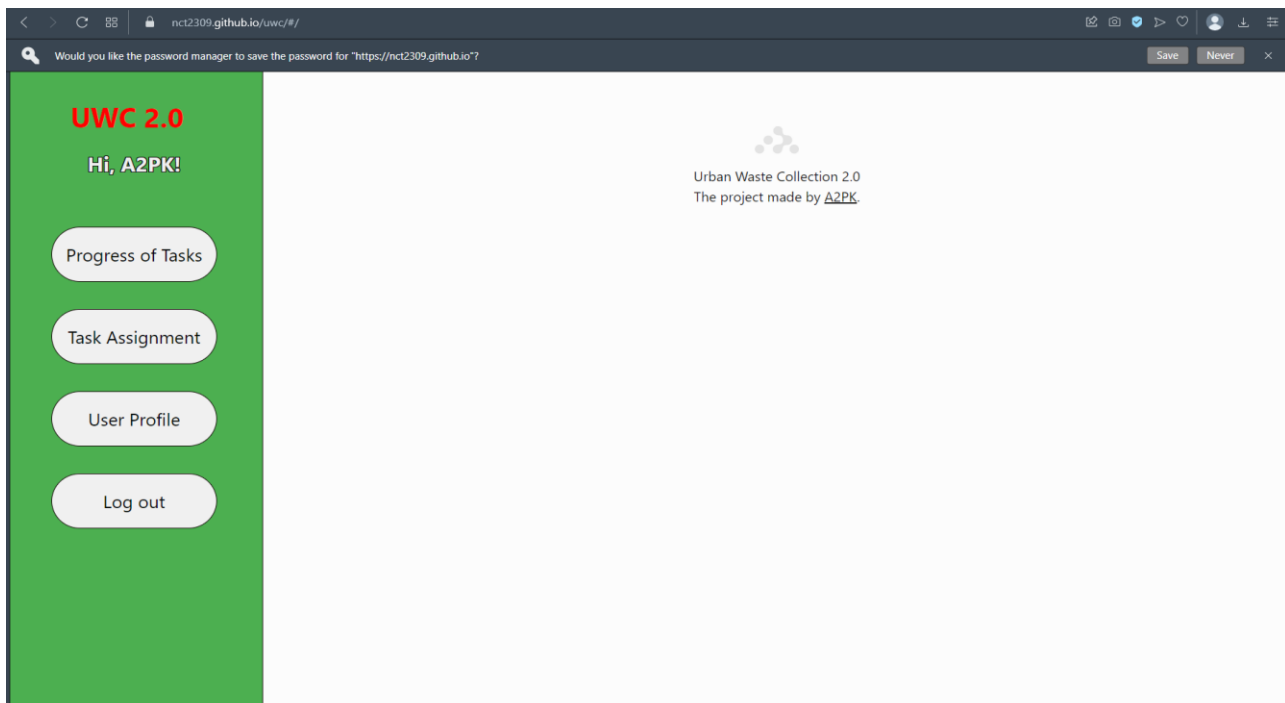
5.1.2 Client Demo

Login page:



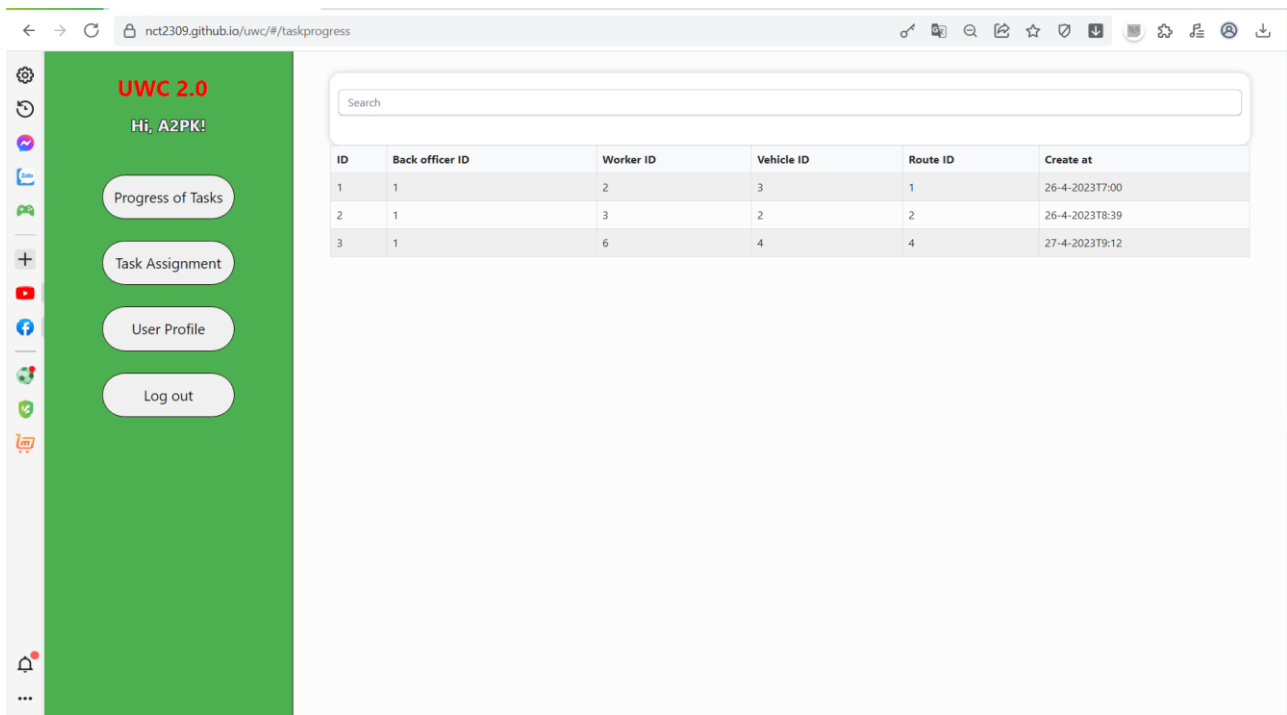
The screenshot shows a web browser window with the URL `nct2309.github.io/uwc/#/login`. The page has a light gray background. At the top center, the text "UWC 2.0" is displayed in red, and below it, "Login" is in black. A white login form with a subtle shadow is centered. It contains a "Username" label, a text input field with "a2pk", a "Password" label, a text input field with "admin", a checkbox labeled "Show password" which is checked, and a green "Submit" button.

Main page:

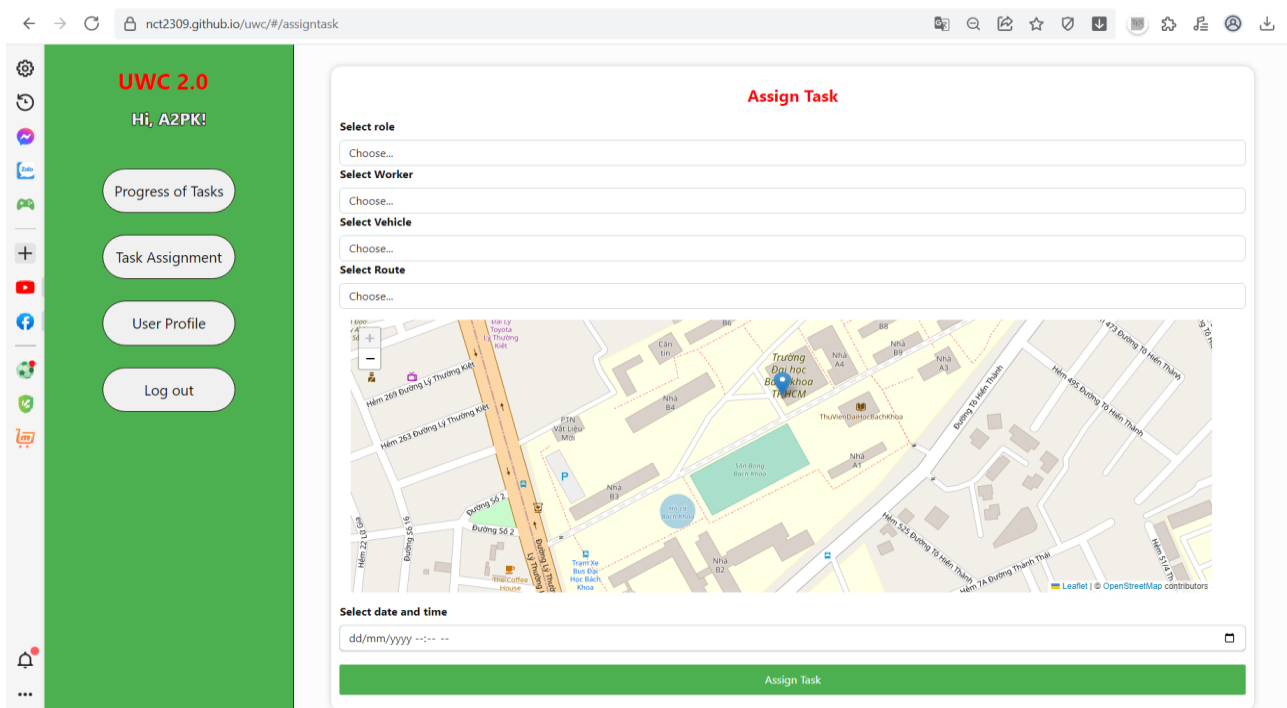


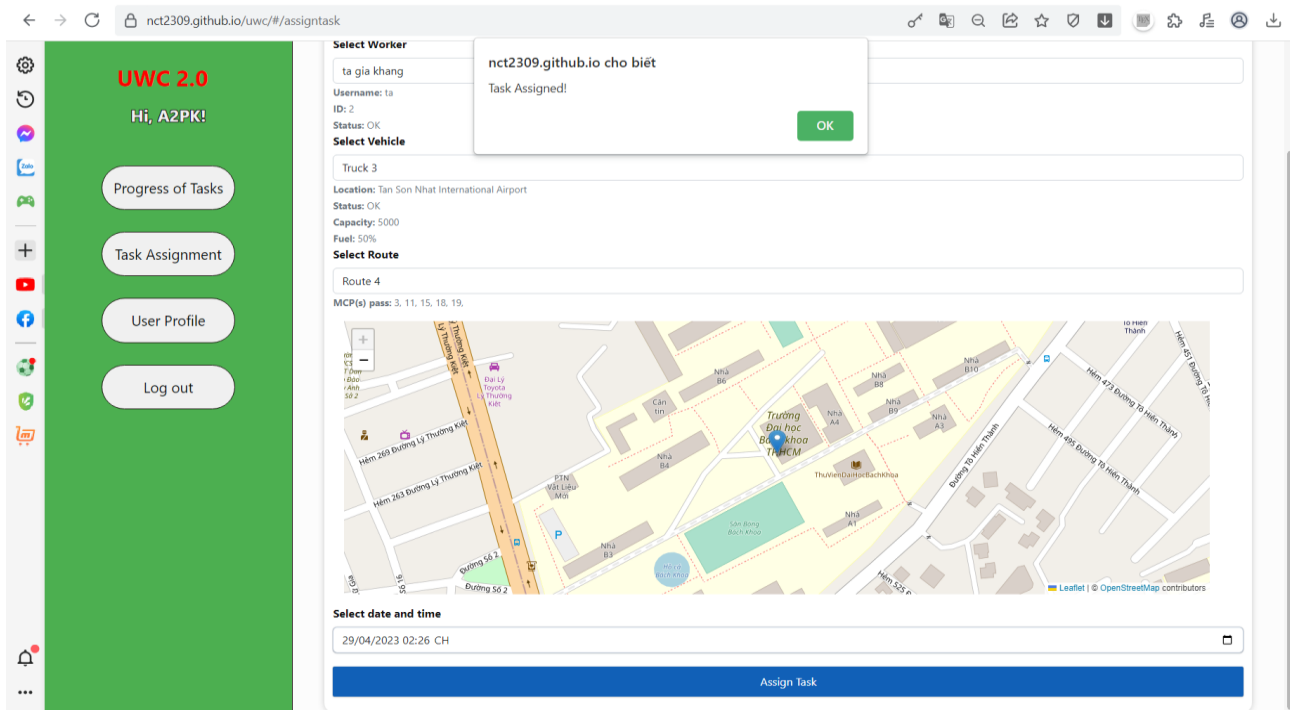
The screenshot shows the main page of the application. The browser window has the URL `nct2309.github.io/uwc/#/`. A password manager prompt is visible at the top. The page features a green sidebar on the left with the text "UWC 2.0" in red and "Hi, A2PK!" in white. Below this, there are four white buttons with rounded corners: "Progress of Tasks", "Task Assignment", "User Profile", and "Log out". The main content area has a light gray background. It displays a logo of five gray dots arranged in a circle, followed by the text "Urban Waste Collection 2.0" and "The project made by A2PK".

Tasks' progress page:

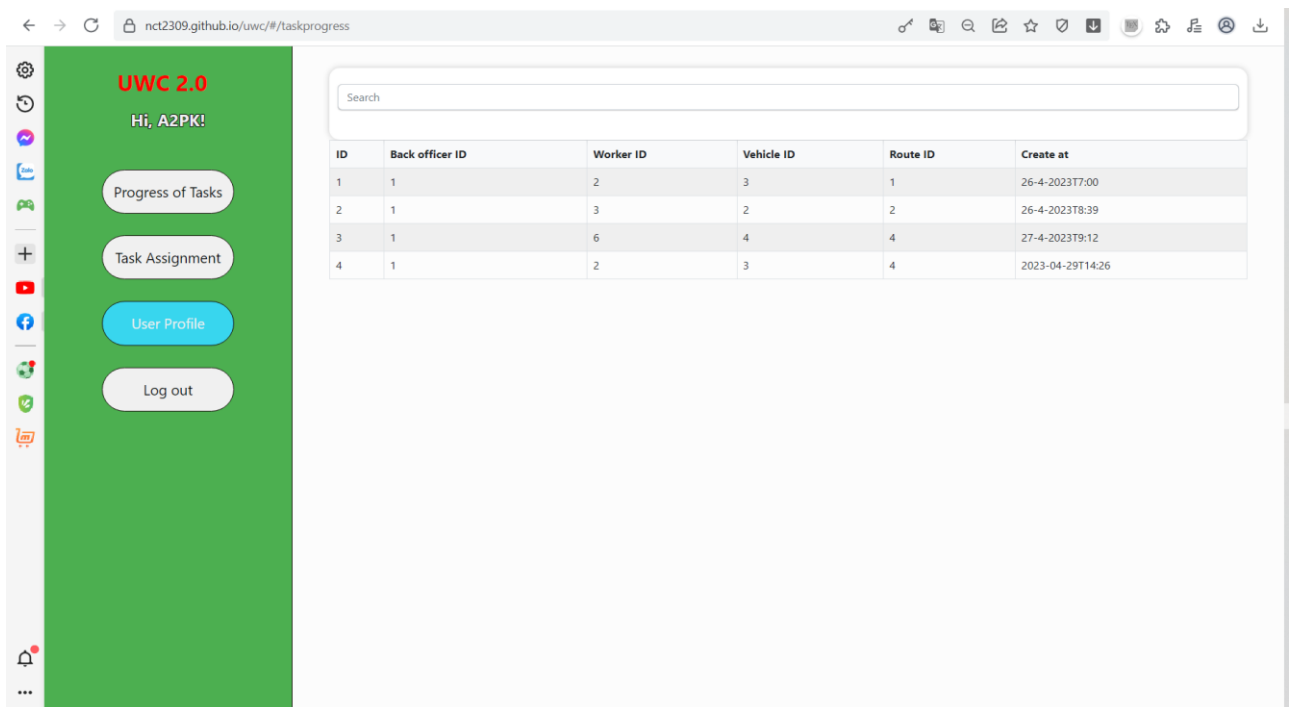


Task Assignment page (the main feature):





This is an example to use the most important feature of the application. And then the result:



User profile page:

UWC 2.0

Hi, A2PK!

Progress of Tasks

Task Assignment

User Profile

Log out

User Profile

ID:1

Name:HCMUT A2PK

Role:backofficer

Password:.....

Change Password

Old password:Password

New password:Password

New password again:Password

Change Password

UWC 2.0

Hi, A2PK!

Progress of Tasks

Task Assignment

User Profile

Log out

nct2309.github.io says

Password Changed!

OK

User Profile

ID:1

Name:HCMUT A2PK

Role:backofficer

Password:.....

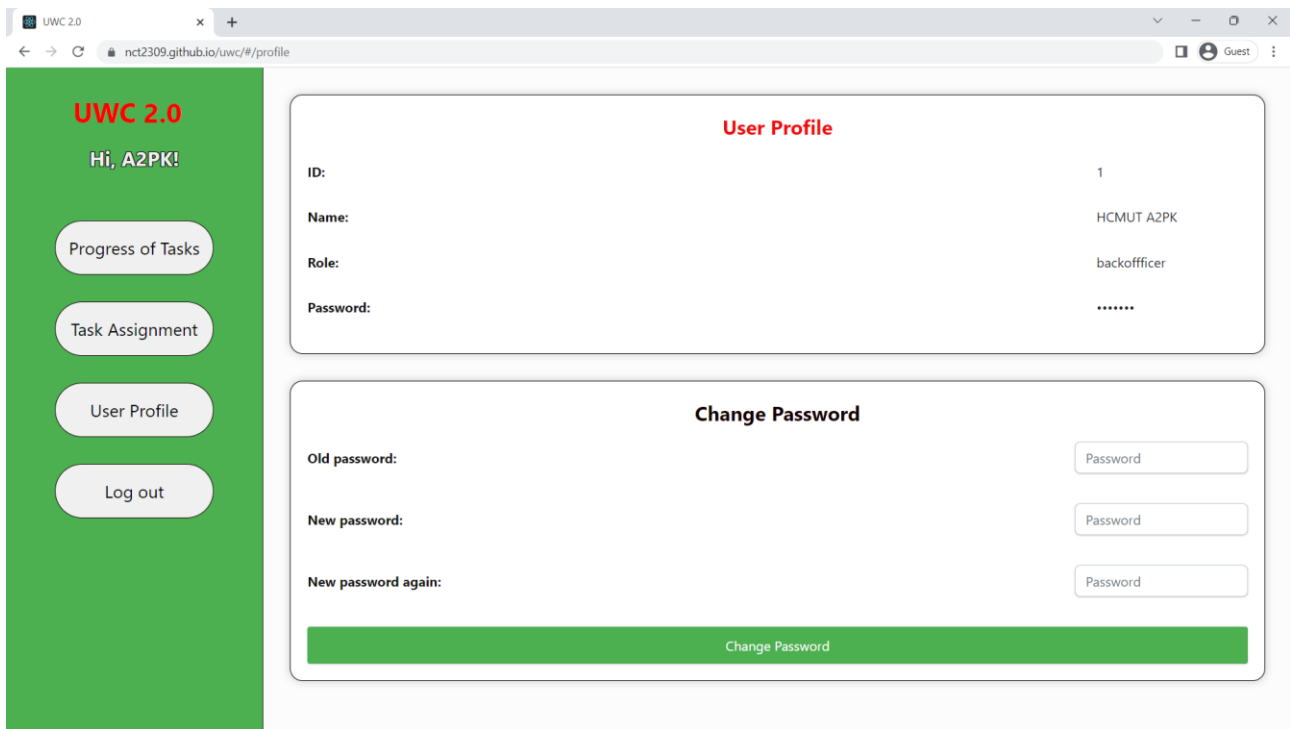
Change Password

Old password:.....

New password:.....

New password again:.....

Change Password



The Client is built with HTML, CSS and TypeScript, rather than JavaScript because it is more convenient in tracking bugs, with framework React. Though there are still some differences in UI rendered in many kind browsers, the Client works in an acceptable way.

This is a *demo page* we deployed by GitHub Pages: <https://nct2309.github.io/uwc/>, it is the static version with data hardcoded in the client but still available to try those features and always ready for your visit. Login with username: a2pk, password: admin.

5.1.3 Problems & solutions

In the implementation of this project, we face a few problems. Firstly, we do not have much experience with TypeScript and Go, in addition, the learning curve is much steeper.

This meant that we could not freely implement features that we have hoped to bring to UWC 2.0. After that, we also have a few problems with GitHub and the merge system, which led to some merge conflicts when working together.

In addition, there is a lack of documentations regarding UWC 1.0, which decrease our speed in implementing already existing features and limit our time with quality of life updates.

- Solution:

In the first week of Sprint 1, we got our team together to learn both TypeScript and Go, as well as GitHub workflow. Although this is not an ideal way to learn, this helped

tremendously in understanding codes and knowing the procedures of pulling and pushing from a git.

We also referencing a lot of online resources like Stack Overflow and other forums.

- Learning experience:

In our project, we have learned various things about the process of Software Engineering. First and foremost is working as a team and utilizing messaging apps like Messenger and Discord with GitHub to work effectively both online and offline.

Next, we also learned to work with foreign programming languages, especially ones with a steep learning curve. Lastly, we now have more experience with design, implement and deploy a software and the process of doing so using graphs, models.