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

# SOFTWARE ENGINEERING

## URBAN WASTE COLLECTION AID - UWC 2.0

| | | |
|---|---|---|
| Instructor: | **Lê Đình Thuận** | |
| Students: | **Châu Đăng Minh** | **- 2013748** |
| | **Đặng Thiên Bảo** | **- 2012653** |
| | **Nguyễn Mậu Minh Đức** | **- 2010230** |
| | **Phạm Hoàng Đức Huy** | **- 2011286** |
| | **Vũ Đăng Khoa** | **- 2011436** |
| | **Nguyễn Huy Quý** | **- 2011951** |
| | **Nguyễn Phú Vĩnh Toàn** | **- 2014775** |

Ho Chi Minh City, September, 2022

# Contents

**I Task 1: Requirements elicitation** **2**
1 Introduction . . . 2
1.1 Project context . . . 2
a Situation . . . 2
b Assumptions and modeling . . . 2
1.2 Relevant stakeholders and their current needs . . . 3
a Relevant stakeholders . . . 3
b Relevant stakeholders' current needs . . . 3
1.3 Stakeholders' current problems . . . 4
1.4 Benefits that UWC 2.0 may bring to each stakeholder . . . 4
2 Requirements . . . 4
2.1 Functional and non-functional requirements that can be inferred from the project description . . . 4
a Functional requirements . . . 5
b Non-functional Requirements . . . 6
2.2 Use-case diagram for the whole system . . . 7
a List of actors . . . 7
b Use-case diagram . . . 7
c Describe the main use-cases of the system . . . 8
3 Task assignment module . . . 9
3.1 Use-case diagram for Task assignment module . . . 9
3.2 Use-case diagram description in table format . . . 9

**II Task 2: System modelling** **13**
1 Activity diagram capturing the business process between systems and the stakeholders in Task Assignment module . . . 13
2 A conceptual solution for the route planning task . . . 13
2.1 Proposal a conceptual solution for the route planning task . . . 13
2.2 Draw a sequence diagram . . . 15
2.3 Class diagram of Task Assignment module . . . 16

**III Task 3: Architecture Design** **17**
1 An architectural approach used to implement the desired system . . . 17
2 Implementation diagram for Task Assignment module . . . 18

**IV Task 4: Interfaces design** **19**
1 Github repository . . . 19
2 MVP1: Desktop-view central dashboard Interface for Task Management for back-officers . 19
3 Implement MVP1 – design an interface of either a Desktop-view central dashboard for select MCPs for back-officers . . . 23

**V Implement MVP2** **26**
1 syncfusion/ej2 . . . 26
2 Login . . . 26

**VI Conclusion** **30**

**VII References** **30**

# I Task 1: Requirements elicitation

## 1 Introduction

### 1.1 Project context

#### a Situation

Nowadays, waste management is one of the important issues that countries around the world pay special attention to, especially developing countries, in order to prioritize economic growth. Waste management is one of the sustainable development goals of countries, so it is deeply concerned by the government and organizations. In cities, solid waste management is costly and inefficient. Therefore, it is urgent to develop a system to improve the management and collection of waste to improve the quality of the environment as well as the quality of people's lives.

A waste management model is not novel, but in the current 4.0 era, we need to develop a model applied new technologies that is more effective and convenient for stakeholders. This will help to optimize waste collection and save money in cities. In this model, software plays a very important role, so it is necessary to research and develop an software to support waste management.

#### b Assumptions and modeling

In this assignment, we have to design and implementation a wast management software (UWC 2.0) satisfying following constraints:

1. Based on an existing database system used for the 1.0 version, including
   - Locations: the map of the city needed to be collected garbage from, locations of Major Collecting Points (MCPs), relative locations of trollers i.e. each roller is assigned to a quarter or a street;
   - Capacities: specific volume of a troller, an MCP and a vehicle container comparing to each other, or volume ratios between them. To be specìic and for simplicity, a troller's volume is **500** $dm^3$, an MCP's volume is equal to **15** trollers, and a vehicle's volume is 20 $m^3$ i.e. it is equal to **5** MCPs after compressing garbage.
   - Staffs: information of everyone in every positions, which are back-officers, collectors and janitors.
2. Waste collection is a gathering and transporting model including following steps:
   - Janitors use trollers to collect garbage in their assigned areas and deliver to Major Collecting Points (MCPs). Every time arriving in an MCP, janitors report this MCP's fullness by adding the number of trollers added to this MCP. Therefore, given the ratio between their volume, this MCP's remaining capacity can be calculated and sent to the database of the system.
   - Every time an MCP exceeds 90% of their capacities, a message is sent to back-officers. At some time with reasonable weather or reasonable social condition in MCPs' place, back-officers chooses MCPs that can be collected and the system creates routes, as well as assigned collectors and their vehicles.
   - Assigned collectors pick up garbage at these MCPs and return to waste treatment factories. It is assumed vehicle garages locate at waste treatment factories.
3. The model's size is small or medium, which serves for a city or a residential area so that the waste treatment factory operation is not overloaded and the cost of transportation is not too much.
4. The software must target to all people who are used it with varying levels of technology proficiency. Therefore, we have to develop a user-friendly software.
5. Managers are able to manage and coordinate employees and vehicles.

6. Staffs are informed about their work schedule as well as to communicate and report information and data to each other and to the system manager. The requirement for this feature is the messages should be communicated in a real-time manner with delay less than 1 second.

From the above features, it can be seen that we need to build a simple, easy-to-use software that focuses on speed and stability instead of complicated functions.

## 1.2 Relevant stakeholders and their current needs

### a Relevant stakeholders

1. The organization X: the organization contracted to develop information management software.
2. Service provider Y: software service provider contracting with organization X to develop information management software.
3. Back - officers: central system operators creating schedules, plans, arranging vehicles, routes of vehicles, coordinating collectors and janitors.
4. Collectors: drivers of different types of waste collection vehicles to collect garbage at the gathering points.
5. Janitors: collectors using trolleys to collect waste in assigned areas and move waste to the gathering points.
6. Administrator: general manager of all backs - officers, collectors, janitors and the data system.
7. IT staffs of provider Y: responsible for installation and maintenance of the system.

### b Relevant stakeholders' current needs

1. Back - officers
   - Have an overview of janitors, collectors and their work schedules;
   - Have an overview of MCPs information;
   - Have an overview of types of transport vehicles and their specifications (weight, power, fuel consumption, etc.);
   - Be notified of full MCPs;
   - Assign tasks for collectors and janitors and assigning vehicles to them;
   - Have optimal routes for fuel consumption and travel distance for collectors created by the system;
   - Be able to send messages to collectors and janitors (depending on their access to the internet, considering using SMS)
2. Collectors and Janitors
   - Have an overview of their work schedule and be notified when there are any changes
   - Have a detailed view of their task on a daily and weekly basic.
   - Be able to communicate with other employees
   - Be able to check in / check out their tasks every day
   - Getting notified about the MCPs that they manage
3. Administrator
   - Managing the back - officers, collectors and janitors
   - Managing collecting vehicles

### 1.3 Stakeholders' current problems

1. For software developers:
   - According to the information of the project, the waste management system called UWC 2.0 has been existing an old version UWC 1.0. Developers must build the new system based on the database of the old system in order that the new system can make use of the data of old system. This can be a "stepping stone" but it can also be a challenge in the process of developing new systems.
   - This system is a real-time data processing system with a delay of less than 1 second in order to update the necessary changes of information immediately to avoid delaying because the information doesn't reach the receivers on time.
   - Information should be updated from MCPs every 15 minutes with the availability of at least 95% of their operating time. That raises the problem that we need to invest more to host or rent a server that is powerful enough to meet the demand.
2. For users: In fact, there will be many situations that can obstruct users:
   - A large number of janitors, collectors, vehicles, and MCPs bring about a hard problem for the back-officers to manually divide tasks for each object. Therefore, it is necessary to add a feature that supports the automatic division of tasks according to where each collector lives and can group people to work at the same time (shift) to easily manage them.
   - Routing the way for the collector is also a problem for the back-officers. Therefore, it is also necessary to have a feature that supports map navigation and gives the most optimal route for the collectors.
   - The software needs to be connected to the server, so it requires users to have a constant network connection to update information. That leads to problems with the battery on the users' devices. Continuous use of data causes a rapid battery drop and may turn off the power. And so users will not be able to contact and update information during a long working day.
   - Software that requires displaying all the information on one page can make it difficult for people with poor eyesight to see the information as well as those using devices with small screens.

### 1.4 Benefits that UWC 2.0 may bring to each stakeholder

- Back - officers: more convenient for management, coordination, and arrangement thereby saving time and costs for the company.
- Collectors, Janitors: have an overview of schedules, working times, and locations. Therefore, they can actively arrange their work.
- Administrator: Easily managing the human resources of the company.
- IT staffs: earn well-paid money for their work.

In general, the development of this waste management system model brings convenience, savings, and increased income for stakeholders.

## 2 Requirements

### 2.1 Functional and non-functional requirements that can be inferred from the project description

*Firstly, the specification of functional and non-functional requirements is provided as below*

- *Functional requirements concentrate on the relationships between the overall system and its users: back officers, janitors, collectors and administrators.*
- *Non-functional requirements revolve around the interface, safety, security, qualification, operation, maintenance and performance.*

### a Functional requirements

**Back-officers**

1. Have an overview of janitors and collectors, their work calendar;
2. Have an overview of vehicles and their technical details (weight, capacity, fuel consumption, etc);
3. Have an overview of all MCPs and information about their capacity.
4. Information should be updated from MCPs every 15 minutes with the availability of at least 95% of their operating time;
5. Assign vehicles to janitors and collectors;
6. Assign janitors and collectors to MCPs (task);
7. Create a route for each collector (Assigned route is optimized in term of fuel consumption and travel distance);
8. Be able to send message to collectors and janitors.

*In reality, it is impossible for back-officers to assign such a large amount of MCPs to collectors (in this project case, the number is 1000 at the beginning), so this task is compute automatically in an acceptable way by the system. So that back-officers' major work is to check the rationality of the auto-tasking, including the staffs' ability to complete assigned tasks and the ability of facilities, which are vehicles, trollers and MCPs.*

**Janitors**

1. Have an overview of their work calendar;
2. Have a detail view of their task on a daily and weekly basic;
3. Be able to communicate with collectors, other janitors and back officers;
4. Check out every time the task is completed and estimate the amount of garbage taken to MCPs measured by a ratio over the number of trollers.

**Collectors**

1. Have an overview of their *possible* work calendar;
2. Have a detail view of their task based on the work calendar: the route through MCPs to be collected, information of their used vehicle (location in the garage and remaining fuel)
3. Be able to communicate with other collectors, janitors and back officers;
4. Check out every time the task is completed;
5. Be notified about the MCPs if they are fully loaded.

**Administrators**

1. Manage (add, edit and delete) the staffs list;
2. Manage (add, edit, delete) the vehicles list and their parameters (weight, capacity, location in the garage, remaining fuel)
3. Be able to communicate with other collectors, janitors and back officers;
4. Manage (add, delete, edit) the list of MCPs and their capacity.

### b Non-functional Requirements

**Performance**

1. The ideal website load time should be no more than 2 seconds;
2. The process of sending messages between users has a delay of no more than 1 second;
3. System response to remaining user actions is no more than 1 second;
4. The system will be able to process real-time data from at least 1000 MCPs now and 10,000 MCPs in the next 5 years;
5. MCPs information is updated every 15 minutes and is available 95% of system uptime.

**Easy-to-use**

1. Each function performs less than 4 operations.
2. Back officer, Administrator can use the application proficiently after an average of 15 minutes of training.
3. Janitors and Collectors can use the application proficiently after 5 minutes of training.

**Size**

1. The total size of the files downloaded to the user's device should not exceed 200MB.

**Reliability**

1. The average number of failed system hits is 2 out of 1000.

**Robustness**

1. System reboot time no more than 1 minute.
2. The probability of data corruption is less than 2

**Security**

1. The password must be hashed when it reaches the database.
2. Have an authentication and authorization.

**Localization**

1. Support Vietnamese language, can add some languages if needed.

**Maintainability**

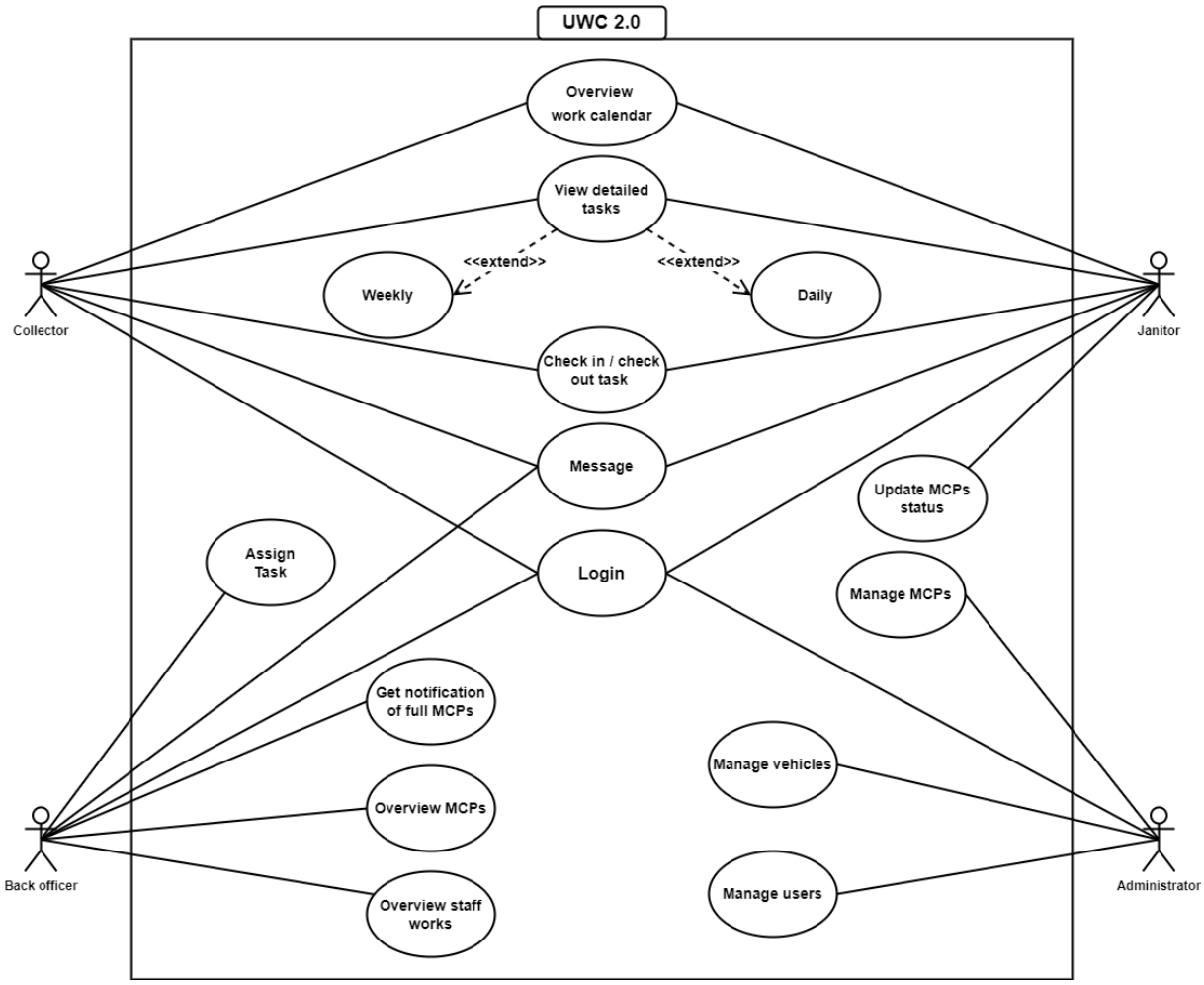
1. Realize the project according to a professional process, ensuring the source code is easy to maintain and upgrade in the future.

## 2.2 Use-case diagram for the whole system

### a List of actors

| Actor ID | Name of actor |
|---|---|
| 1 | Back-officer |
| 2 | Janitor |
| 3 | Collector |
| 4 | Administrator |

### b Use-case diagram

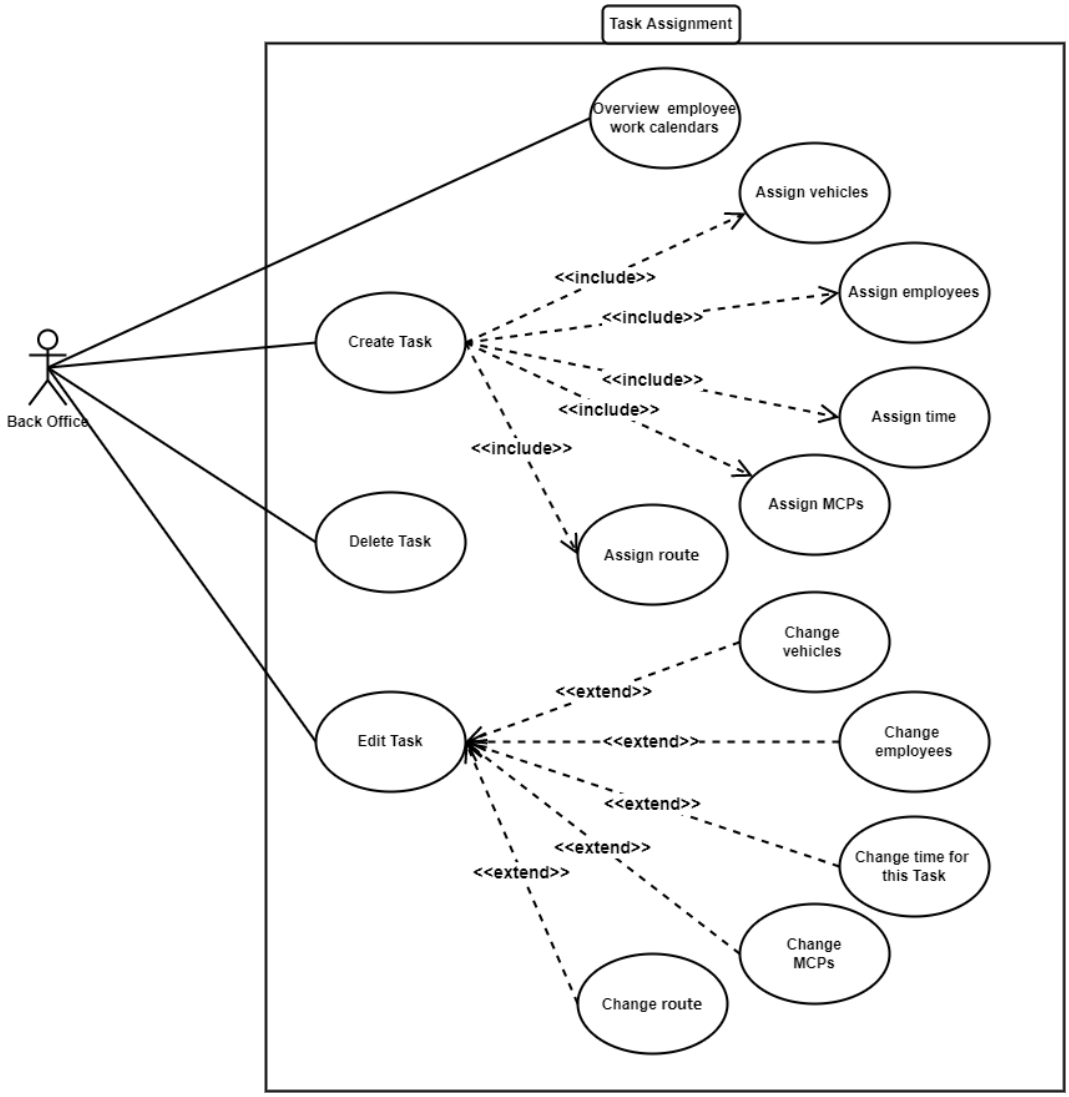


### c Describe the main use-cases of the system

| Use-case ID | Use-case name | Description |
|---|---|---|
| 1 | Login | Allow end-users to log in to start using system features |
| 2 | Assign Task | Allow back-officers to assign janitors and collectors work, choose vehicles, MCPs and routes |
| 3 | Overview MCPs | Allows back-officers to know the status of all MCPs |
| 4 | Overview staff works | Allow back-officers to know all staff's work status. |
| 5 | Overview work calendar | Allow janitors and collectors to know their work schedule shown in their calendar |
| 6 | Get notification of full MCPs | Notify back-officers when an MCP are full |
| 7 | View detailed tasks | Allows janitors and collectors to know their work schedule for the day without scrolling down |
| 8 | Check in/check out task | Allow janitors and collectors to label assigned tasks (Ready, Doing, Done) |
| 9 | Message | Allow back-officers, janitors and collectors to send messages to others |
| 10 | Update MCPs status | Allow janitors to update how many trollers they have put to an MCP |
| 10 | Manage MCPs | Allows administrators to manage (add, delete, edit) the list of MCPs |
| 11 | Manage vehicles | Allows administrators to manage (add, edit, delete) the list of vehicles and their parameters (weight, capacity, fuel consumption, etc) |
| 12 | Manage users | Allows administrators to manage (add, edit, delete) the list of staffs (back-officers, janitors and collectors) |

# 3 Task assignment module

## 3.1 Use-case diagram for Task assignment module




## 3.2 Use-case diagram description in table format

Table 1: Task Assignment

| | |
|---|---|
| User-case name | Task assignment |
| Actor | The Back officer |
| Description | The Back officer can overview the employee's work schedule, assign task, edit some task features and also delete tasks. |
| Trigger | Click "Task assignment" button. |
| Precondition | The Back officer must have valid login and their account must have permission to access Task assignment. |

Continued on next page

Table 1: Task Assignment (Continued)

| | |
|---|---|
| Normal flow | 1. System displays "Task assignment" console and show all tasks created by the Back officer. Each time the Back officer can only choose one of four branch performed by four button in Task assignment console:<br>1. **Overview employee work calendars**: They can overview the employee's work schedule.<br>2. **Create Task**: They can create a new task, then assign some crucial information fields such as: vehicles, employees, working times and deadlines, MCPs and the optimized routes connected between each MCPs in this task.<br>3. **Delete Task**: They can delete task being chosen.<br>4. **Edit Task**: They can edit task such as: change vehicles, employees, timing for this task, MCPs, routes...<br>2. After clicking the button, a new console will be shown up and the Back officer can manipulate in this new console. |
| Exception | None |
| Alternative flows | None |

Table 2: Overview employee work calendars

| | |
|---|---|
| User-case name | Overview |
| Actor | Back officer |
| Description | The Back officer can overview the employee's work schedule |
| Precondition | The Back officer must have valid login and their account must have permission to access Task assignment. |
| Normal flow | 1. The system displays work schedule.<br>2. If the Back officer doesn't want to view schedule,The Back officer can select back arrow to back to main page. |
| Exception | None |
| Alternative flows | None |

Table 3: Delete task

| | |
|---|---|
| User-case name | Delete task |
| Actor | Back officer |
| Description | The system help Back Officer to delete task |
| Precondition | • Back officer must have valid login and their account must have permission to access Task assignment.<br>• Vehicles have some trouble<br>• The employees can't complete the task |

Continued on next page

Table 3: Delete task (Continued)

| | |
|---|---|
| Normal flow | 1. The system displays list of task.<br>2. The Back officer select task to delete.<br>3. Back officer select "delete task".<br>4. System remind that "Are you sure to remove this task".<br>5. Back officer select "Ok" to confirm.<br>6. The system update tasks.<br>7. The system display "you remove the task successful".<br>8. The Back officer select "Ok".<br>9. The system back to main page. |
| Exception | None |
| Alternative flows | In **step 5**, if the Back officer don't want to delete task, they can select "Cancel" to back to the page which display list of task in **step 2**. |

Table 4: Create task

| | |
|---|---|
| User-case name | Create task |
| Actor | The Back officer |
| Description | The system help the Back Officer to create new task |
| Precondition | The Back officer must have valid login and their account must have permission to access Task assignment. |
| Normal flow | 1. The Back officer can assign information by taking some actions sequentially bellow:<br>• Assign vehicles.<br>• Assign employees who can join this task.<br>• Add task timing and set deadline for this task.<br>• Assign MCPs.<br>• Assign route.<br>2. After assigning information, they system will check all information they assigned, if its information is valid, the system will notice "Create successfully".<br>3. The system will send message about new task to employees who are assigned in this task.<br>4. The system back to main page. |
| Exception | None |

Continued on next page

Table 4: Create task (Continued)

| | |
|---|---|
| Alternative flows | In **step 2**, if information is invalid, the system will notice "Some information is not corrected or missed" and show all place that needed to be exact. The Back officer must correct until the system accept and notice "Create successfully" and then go to **step 3** in **the Normal flow**. |

Table 5: Edit task

| | |
|---|---|
| User-case name | Edit task |
| Actor | The Back officer |
| Description | The Back Officer can change some task attributes |
| Precondition | • The Back officer must have valid login and their account must have permission to access Task assignment.<br>• They must select existing task before editing.<br>• The selected task must be at least 2 hour before it start |
| Normal flow | 1. In the console showing the entire information about this task, The Back officer can click to button named "Editing". After clicking it, the system will show some information that can be edited such as:<br>• Changing vehicles.<br>• Changing, deleting or adding new employees.<br>• Changing task timing information.<br>• Changing or adding new MCPs.<br>• Changing routes.<br>2. The Back officer can choose some action above and then replace, remove or add new information. If input information is valid, the system will notice "Change successfully".<br>3. After "Change successfully", the system will send message to employees including Janitors and Collectors that this task have been changed some fields (changed information is listed in this message). |
| Exception | They also interrupt this action by clicking "Cancel change" to exist "Edit task" safely. |
| Alternative flows | In **step 2**, If input information is invalid, the system will point out some information being wrong or missed. The Back officer must repair wrong information until the system notice "Change successfully" and then continue to go to **step 3**. |

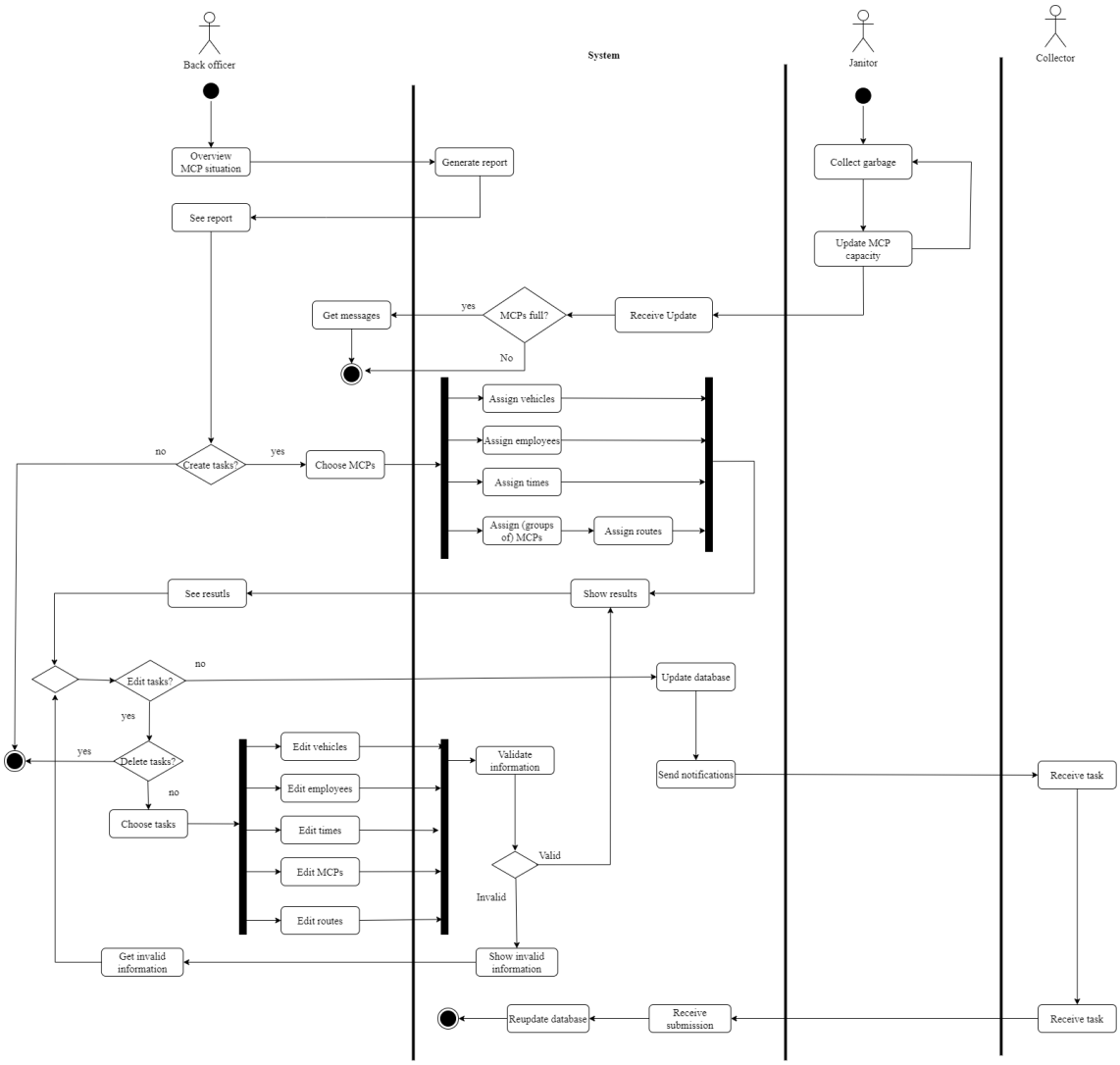
# II Task 2: System modelling

## 1 Activity diagram capturing the business process between systems and the stakeholders in Task Assignment module

Every time an MCP is full, a message is sent to back-officers. At some time with reasonable conditions, back-officers check for full MCPs in MCP database and MCPs needing to be collected.
The system at this time begins these automatic procedures:

1. Plans routes;
2. Assigns least working collectors so far;
3. Assigns vehicles;

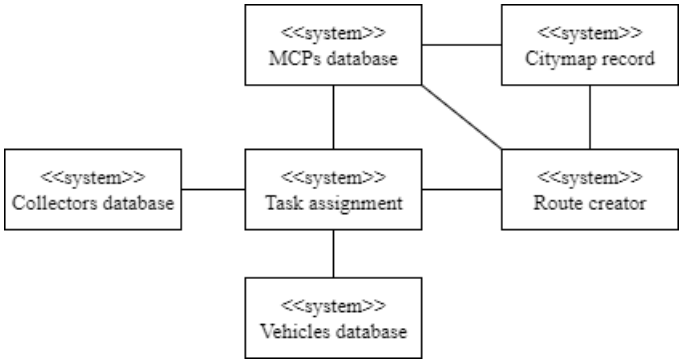
Back-officers may re-check these tasks and eventually sends messages to assigned collectors.




## 2 A conceptual solution for the route planning task

### 2.1 Proposal a conceptual solution for the route planning task

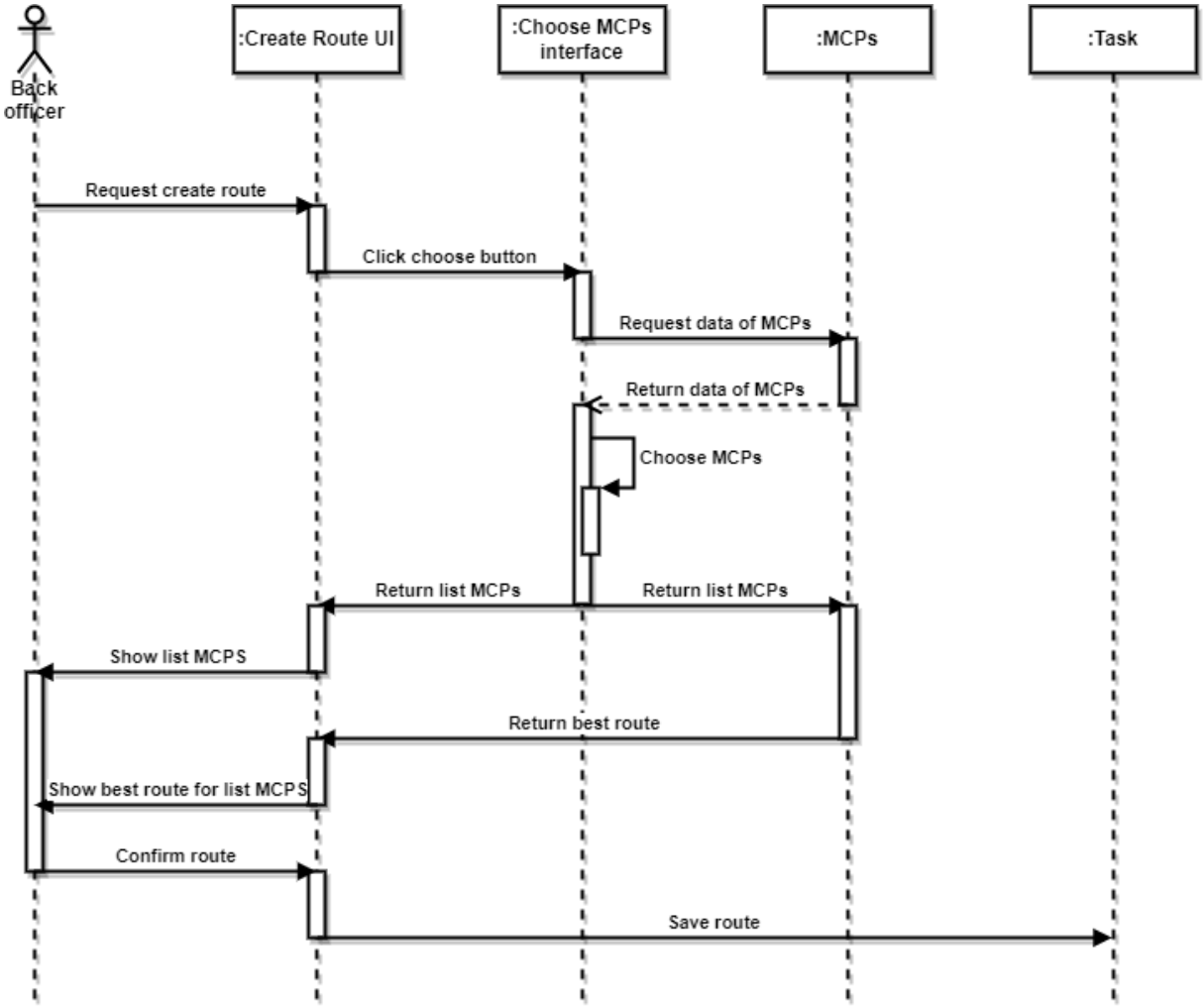
Firstly, we need to go through interactions among modules and databases of our system.
Route planning now can be considered as a problem of finding routes whose sum is shortest in a graph, where each MCP is a node and every possible way from an MCP to another is a weighted edge. This



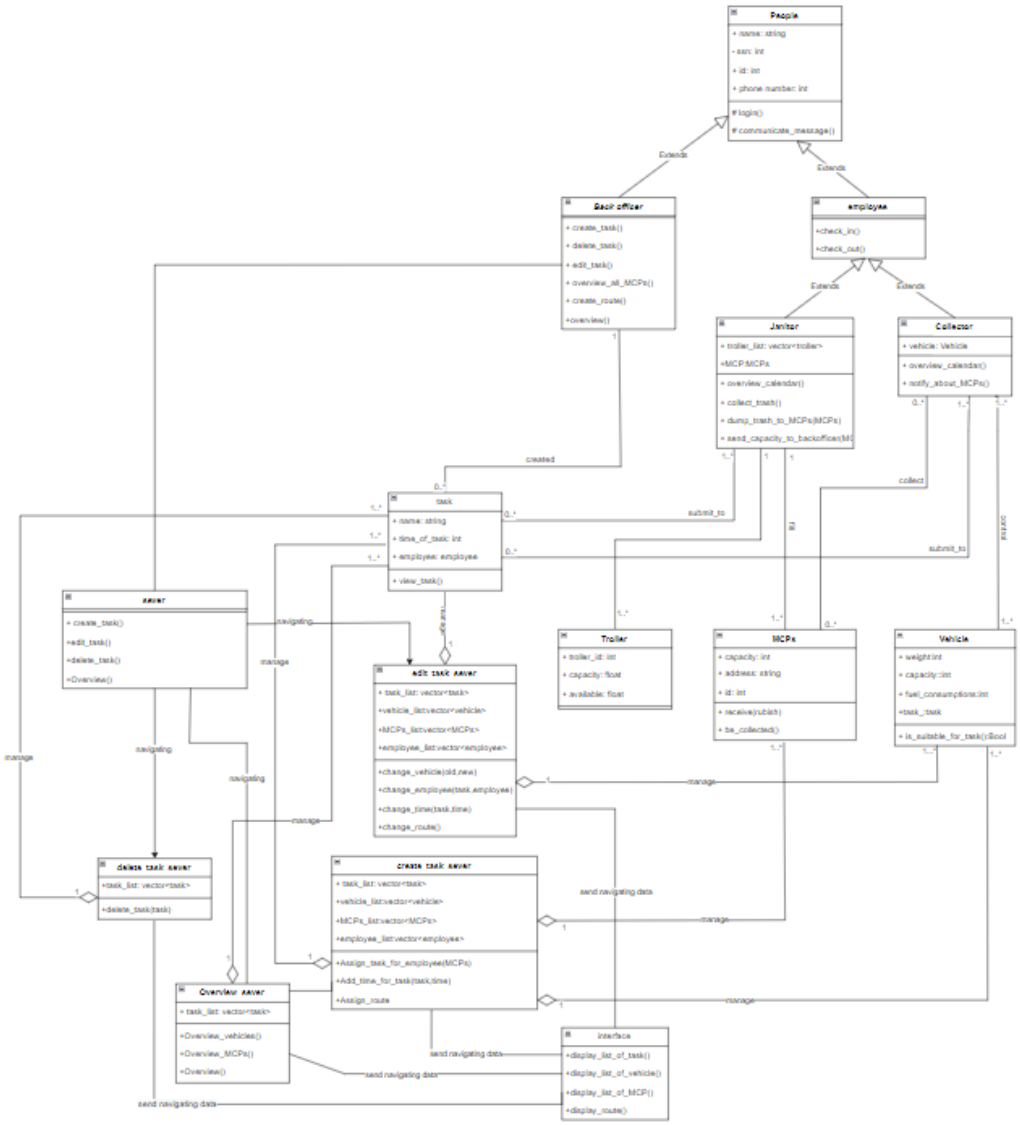

problem seems to be harder than the salesman problem, which is a NP-hard problem, as we have to create more than one circuit. Therefore, it is difficult to find an optimal solution. We recommend doing these following simple steps for reducing computing complexity:

1. As given above, a vehicle's capacity is 5 times as an MCP, so a collector's transportation time can collect up to 5 MCPs. Minimal number of vehicles is compute by the ceiling of the number of full MCPs divided by 5.
$$\texttt{\#Vehicles} = \frac{\texttt{\#MCPs}}{5}$$
2. Every MCP can be specified by a coordinate in the city map. Scale these coordinates so that the top-left MCP is $(0, 0)$.
3. Rank MCPs' coordinate by the distance to $0, 0$ and group every 5 consecutive MCPs for one collection time.

## 2.2 Draw a sequence diagram



### 2.3 Class diagram of Task Assignment module




If you cannot view Class Diagram clearly, please reference this.

**Description:** Back-officer can choose one of the following tasks: overview, create task, delete task, edit task through the sever. When choosing one of the above tasks, it will move to the servers corresponding to the tasks:

- When the Back-officer selects the overview(), the system will move to "Overview_sever", where the system will perform one of the following three tasks:
  - Overview: Display tasks with all information of them including task's name, working time, employee's name through display_list_of_task() method.
  - Overview vehicles: the system displays vehicle information such as weight, capacity, fuel consumption, and tasks in progress through the overview_vehicles() method.
  - Overview MCPs: the system will display information of MCP such as id, address, capacity, MCP is full or not through overview_MCPs() method.
- When the Back-officer selects the create_task(), the system will move to "create_task_sever", here the system will forces the Back-officer chooses 1 of 3 following tasks:

  - Assign employee: the system will allow the Back-officer to assign tasks to employees by letting the Back-officer choose the MCPs need to be performed the task, if the MCPs are not full, the system will automatically assign tasks to the janitors, if the MCPs are full then the system will automatically assign tasks to the collectors, after confirming the completion of adding tasks, the system will display a list of tasks. This is done by the assign_task_for_employee(vector<MCPs>) method.
  - Assign time: the system will display tasks with all information of the task including name, working time, employee's name, then Back-officer will select the task that needs more time, then offer the time, after confirming completion the system will display a list of tasks with information about the time that has been changed. This is done by the add_time_for_task(task,time) method.
  - Assign MCPs: the system will display the information of the MCP such as id, address, capacity, whether the MCP is full or not through the assign_MCPs() method.
  - Assign route: the system will display the information of the vehicle, then select the possible route by itself through the route finding algorithm in section 2.2 with the input of MCPs with the same address, their status then displayed. route display using the display_route() method. This is done through the assign_route() method.

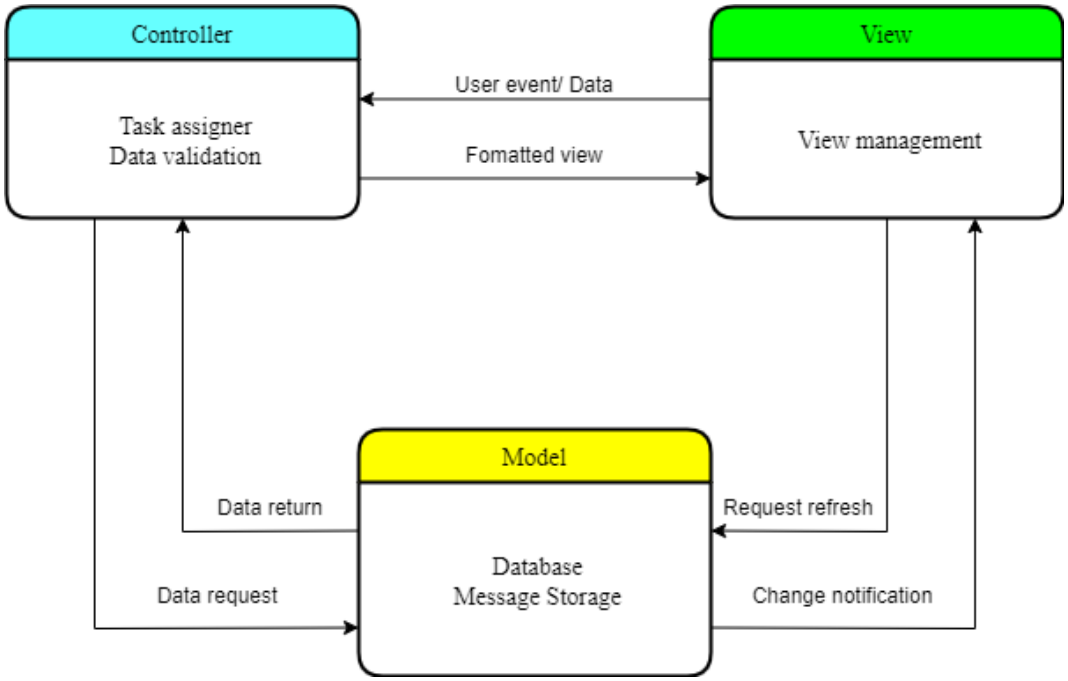
- When the Back-officer selects the delete_task(), the system will move to "delete_task_sever", here the system will display the tasks with all the information of the task including name, working time, employee name. Then Back-officer will select task to delete and after confirming the system will display a list of tasks after deletion through the delete_task(task) method.

- When the Back-officer selects the edit_task(), the system will move to "edit_task_sever", here the system will will forces the Back-officer chooses 1 of 4 following tasks:

  - Change vehicle: the system will display vehicle information such as weight, capacity, fuel consumption, task in progress, Back officer will select the vehicle that needs to be changed and then select the vehicle to be changed Instead, after confirming the system will re-display the status of the vehicles after the change via the change_vehicle() method.
  - Change time: the system will display the tasks with all information of the task including name, working time, employee name, then Back officer will select the task to change the time, then enter the time, after Once completed, the system will display a list of tasks with information that the time part has been changed in the selected task. This is done through the change_time(task,time) method.
  - Change employee: the system will display tasks with all information of the task including name, working time, employee name, then Back officer will select the task that needs to change employee, then choose a replacement employee, After confirmation, the system will display the tasks with all information of the task including name, working time, employee name. This is done through the change_employee(task,employee) method.
  - Assign route: the system will display the possible routes and allow the Back officer to pass the route then display the route using the display_route() method. This is done through the change_route() method.

# III Task 3: Architecture Design

## 1 An architectural approach used to implement the desired system



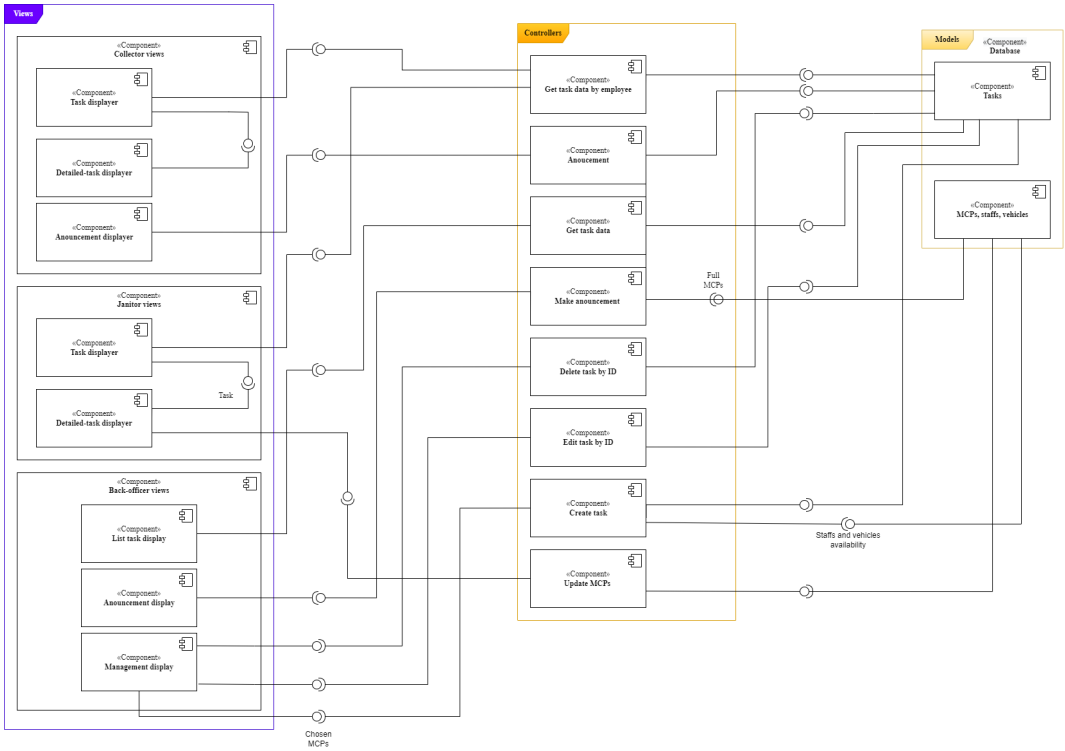

The MVC is known to be the most wide-used model in software engineering. We also choose this model for architecture design, although there is a difficulty that it is usually illustrated as a single-user model. Therefore, we add more section into the view component to clarify who uses which buttons/UI.

The MVC specific to this system includes:

1. Controller: Task assigner (task assignment module) calculates routes based on certain inputs of MCPs, vehicle and staffs; Data validation to check if inputs from back-officers is valid;
2. View: View management specifies output display to users;
3. Model: includes databases of staffs, MCPs, assigned tasks and vehicles, as well as message storage.

## 2 Implementation diagram for Task Assignment module

We draw Component diagram for this module.




**Description**: Each different account types will be taken to a different screen to use this module.

1. Collector views:
   - End users can view summary or details of their works;
   - End users can get notification about the full MCP managed by them in their current task.
2. Janitor views:
   - End users can view summary or details of their works;
   - End users can update their managed MCPs.
3. Back-officer views:
   - End users can view all current tasks
   - End users can announce about the full MCP to change MCP status.
   - End users can manage (create, edit, delete) tasks.

# IV Task 4: Interfaces design

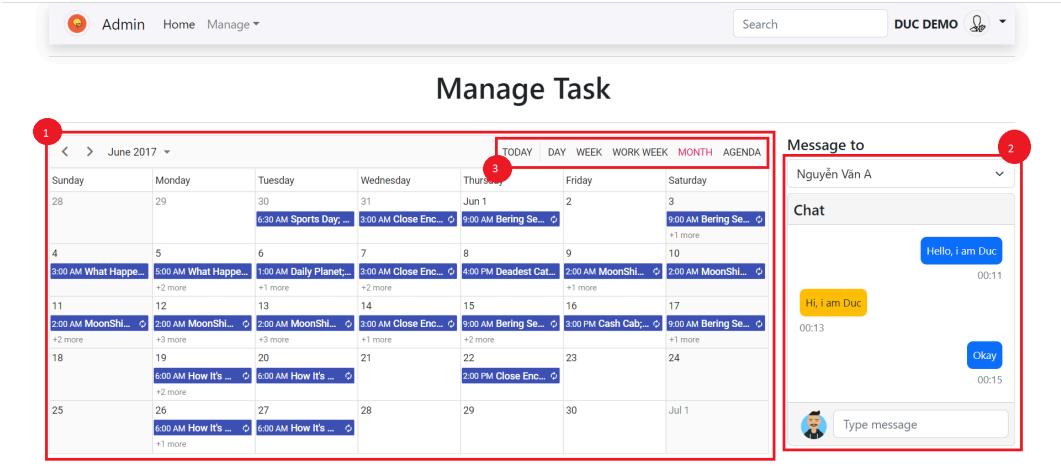
## 1 Github repository

Our project is organized in this link

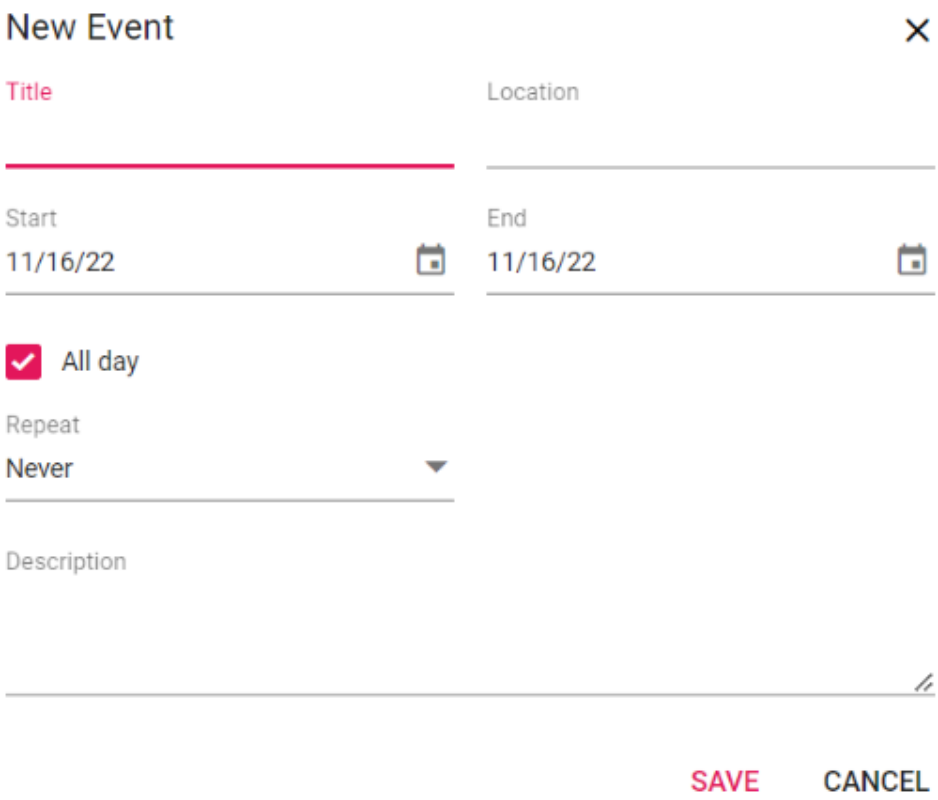
## 2 MVP1: Desktop-view central dashboard Interface for Task Management for back-officers

- Path: Home > Task Assignment

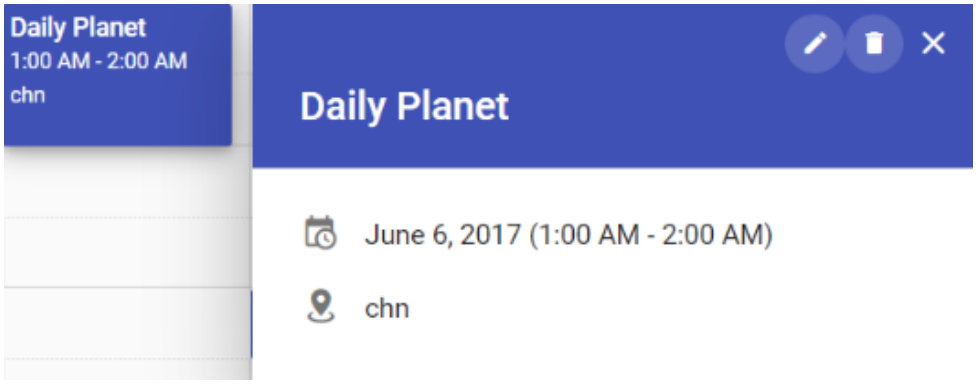
  Overview:



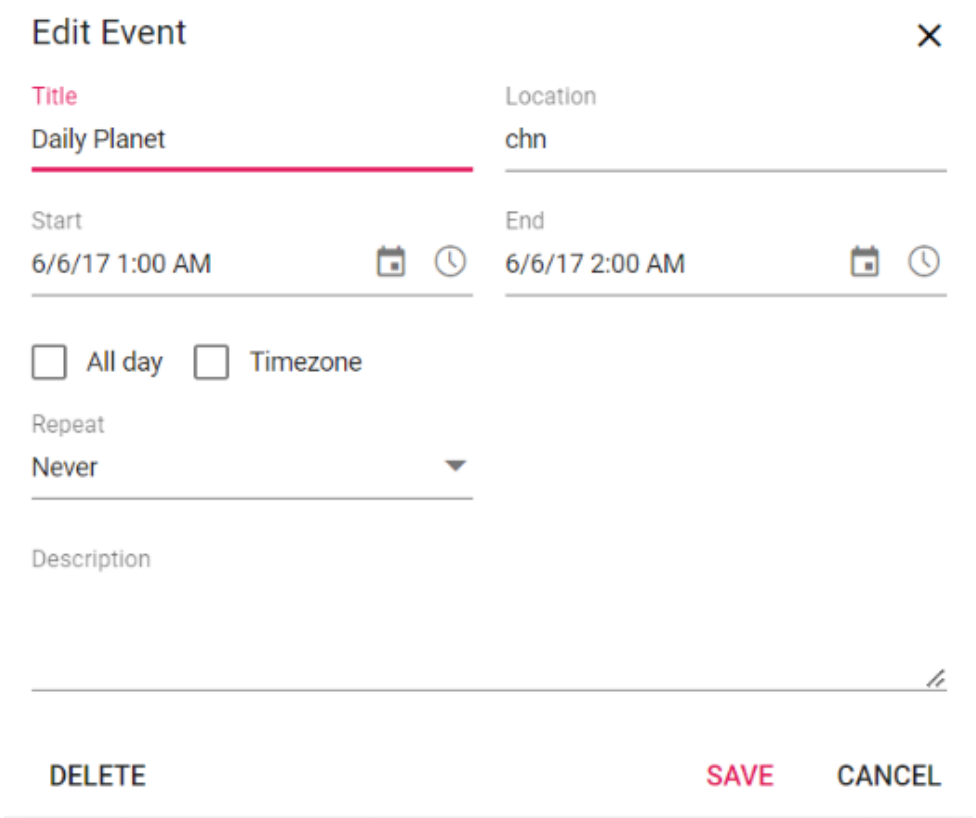

- Components:
  1. Component Manage Task: Allows users to add, delete or edit an event. (tittle, location, date, repeat, description) (item 1)
     - Add:



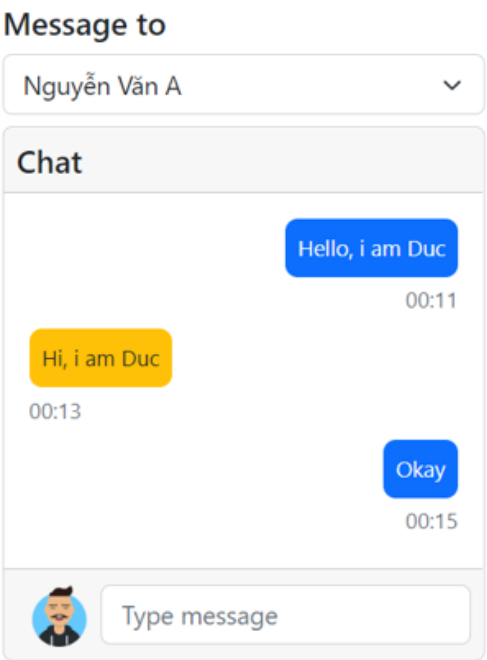

     - Delete:



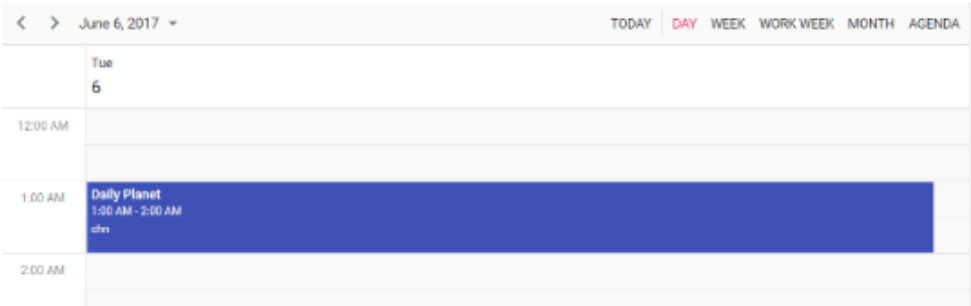

– Edit:



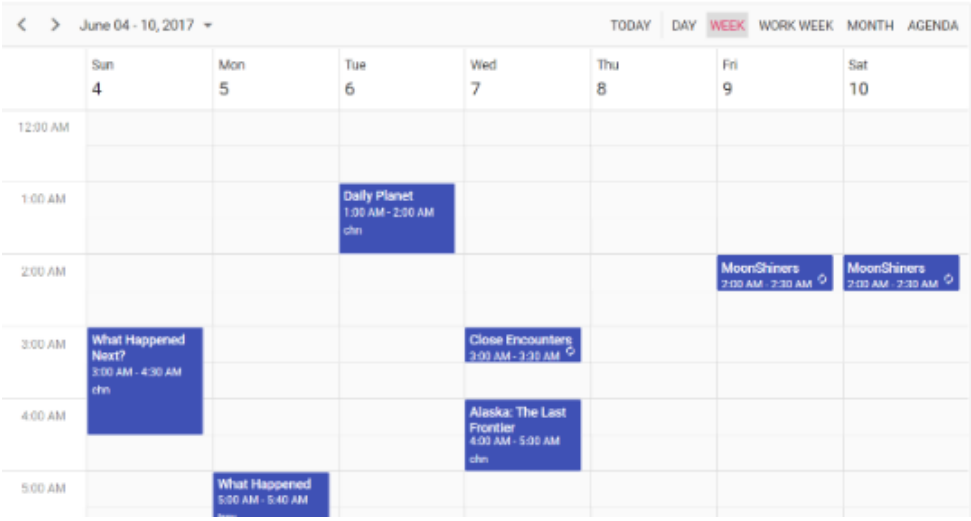

2. Component Chat App: Allows end users to message any specific user in the list of other employees. (item 2)



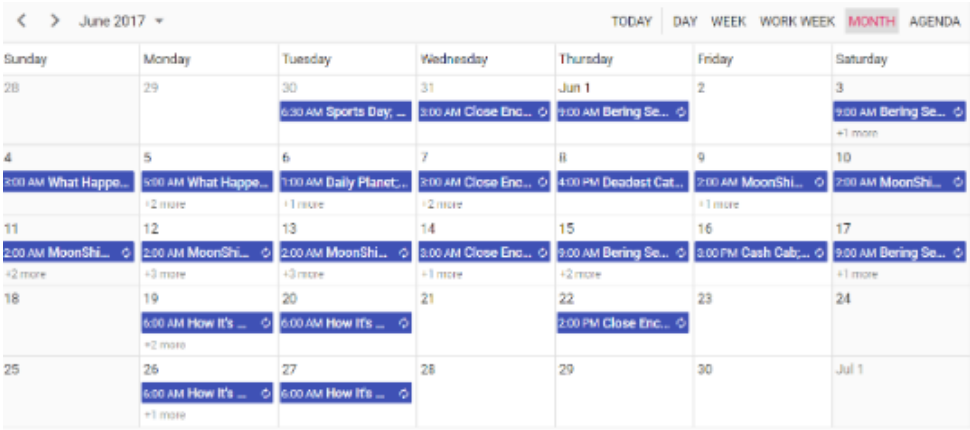

3. There are different view options such as, end user can choose this option on item 3:
   - Day: View the task list for the day



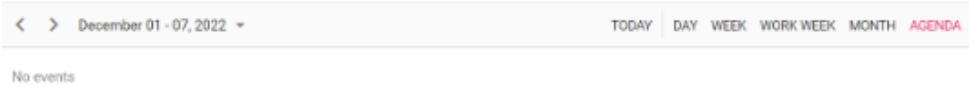

   - Week: View the tasks of the weekday by time




   - Month: View the tasks of the days of the month

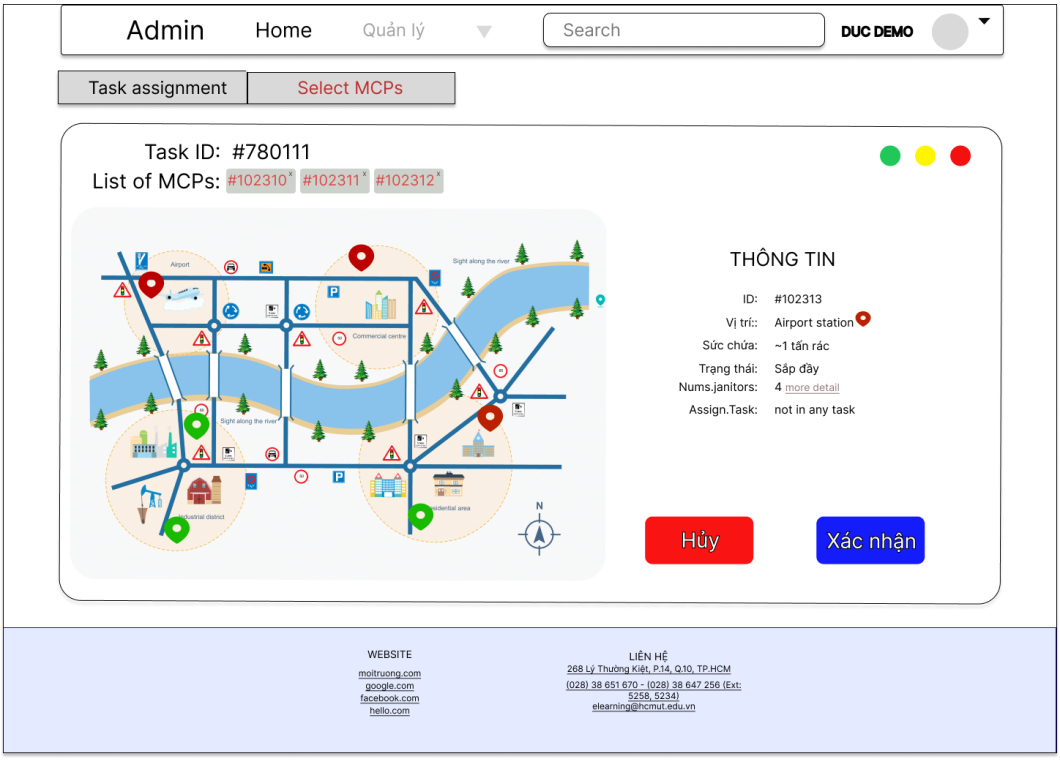



– Agenda




## 3 Implement MVP1 – design an interface of either a Desktop-view central dashboard for select MCPs for back-officers

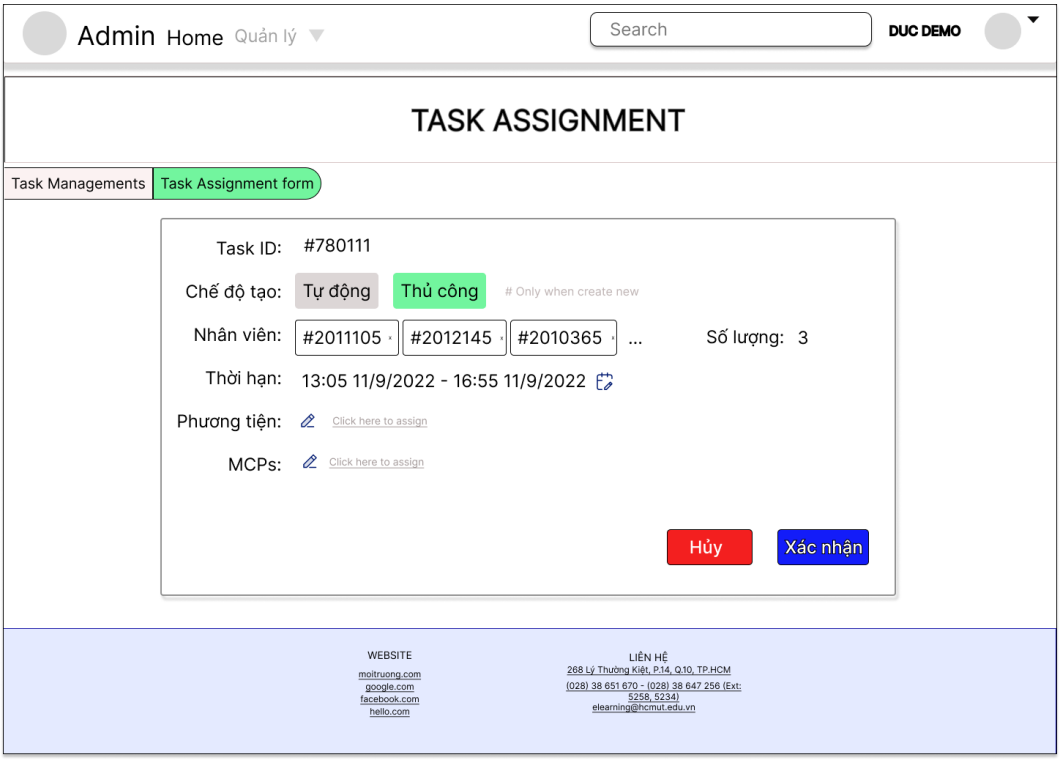
- MCPs console:




– After Back officer assign task, there are 2 option :

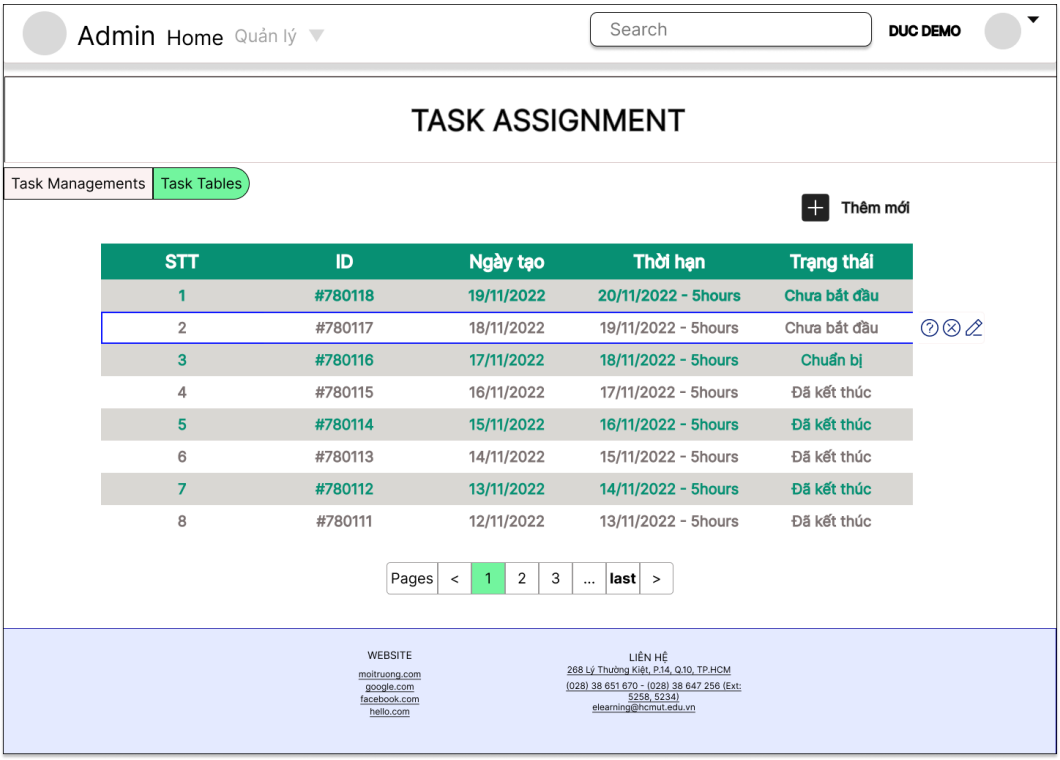
  * Select all MCPs. If Back officer select this option, system select all MCPs and assign task for employees.
  * Select MCPs:The map will appear. If the MCPs almost full then the icon ò MCPs will tranform to red color.At this time,Back officer will select MCPs, beside map is all information of MCPs consists of ID, capacity, location, status, number of janitor. Above map is Task ID and list MCPs was selected.

- Task Assignment form




  - Task Assignment have information of task like task ID,create method,Number of employees do task and their ID, time do task, vehicle and MCPs.
    * If Back officer select "tự động" ,then system will select number of employees do task and their ID,time do task, vehicle.Back Officer will select the MCPs.
    * If Back officer select "thủ công", Back officer will select number of employees do task và their ID, time do task,vehicle and MCPs.

- Task console:

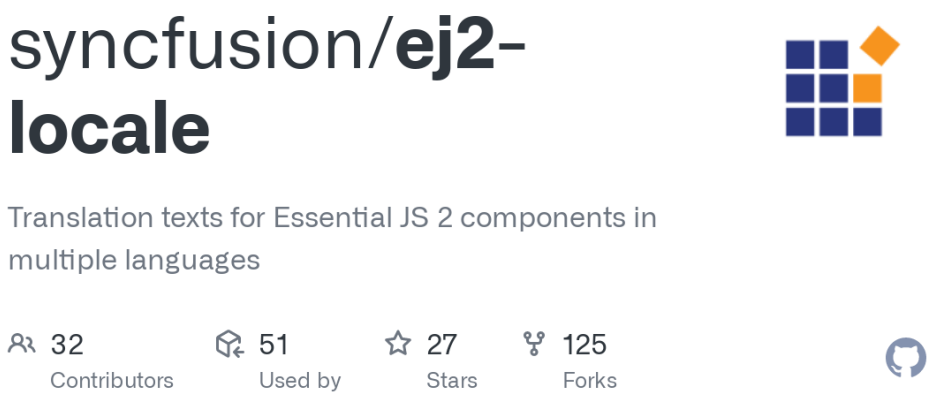



– Task console have information about task consists of task ID, Day create, time do task và status.

# V Implement MVP2

*Implement the screens as outlined in Task 4, add the login function for the product* The team implement with Reactjs for the frontend, and in Nodejs for the backend. It include some libraries such as:

- react-router-dom: to redirect web pages
- react-dom: to redirect web pages
- axios: to call the API from the backend
- socket.io: to create a simple chat for manage task
- @syncfusion/ej2: to create a calendar for task assignment




## 1 syncfusion/ej2

The Syncfusion JavaScript UI controls library is the only suite that you will ever need to build an application since it contains over 65 high-performance, lightweight, modular, and responsive UI components in a single package.

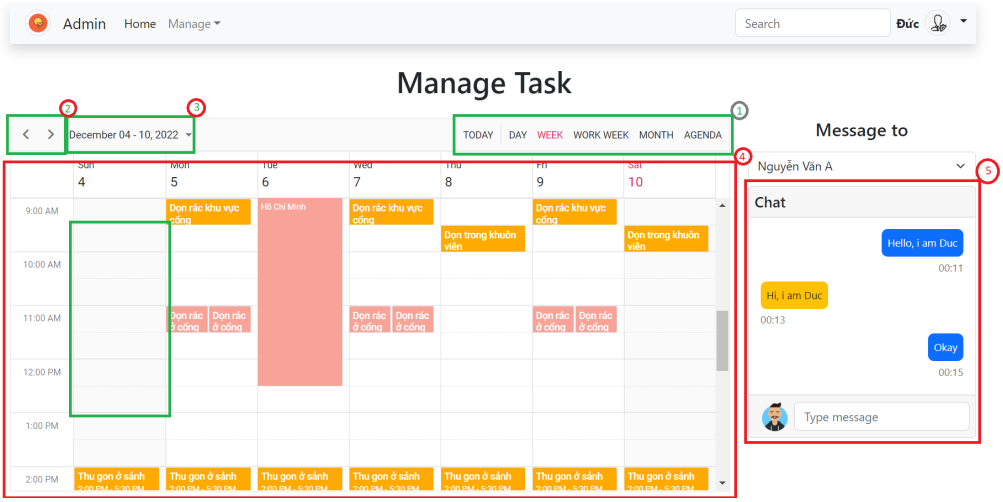
Syncfusion JavaScript (Essential JS 2) is a modern UI Controls library that has been built from the ground up to be lightweight, responsive, modular and touch friendly. It is written in TypeScript and has no external dependencies. It also includes complete support for Angular, React, Vue, ASP.NET MVC and ASP.NET Core frameworks.

The React Scheduler component is an event calendar that facilitates almost all the basic Outlook and Google Calendar features, allowing the user to plan and manage appointments and time efficiently. It receives data from a variety of data sources, such as an array of JSON objects, OData web services, RESTful or WCF services, and DataManager with built-in load on demand support to reduce the data transfer and load time. Also, it is availed with the multiple resources support that allots an unique individual space for more than one resources on the same calendar.

## 2 Login

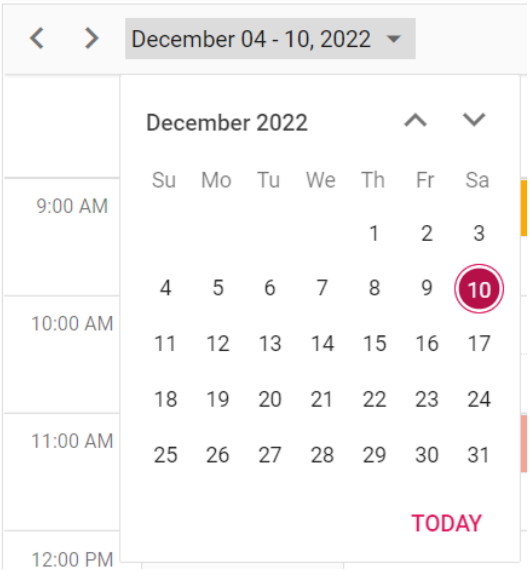
*We have the result in this screen as*

*After logging in we will be directed to the task manager screen*



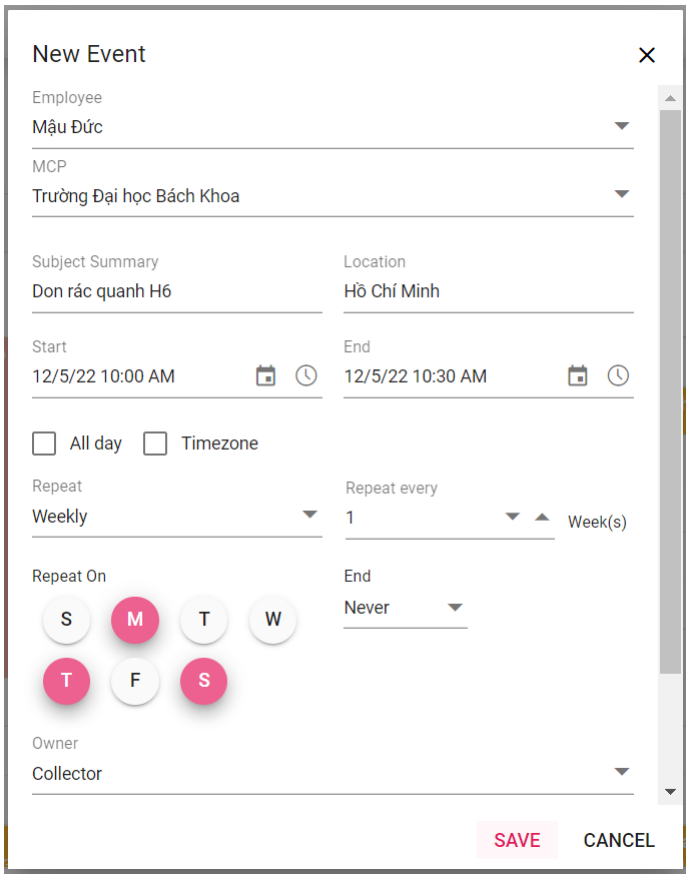

In this screen:

- We can choose a view type for end user like day, week or month at item 1
- We can go back to the calendar 1 week ago or 1 week later by pressing 2 buttons in item 2
- We can choose to see any date at item 3

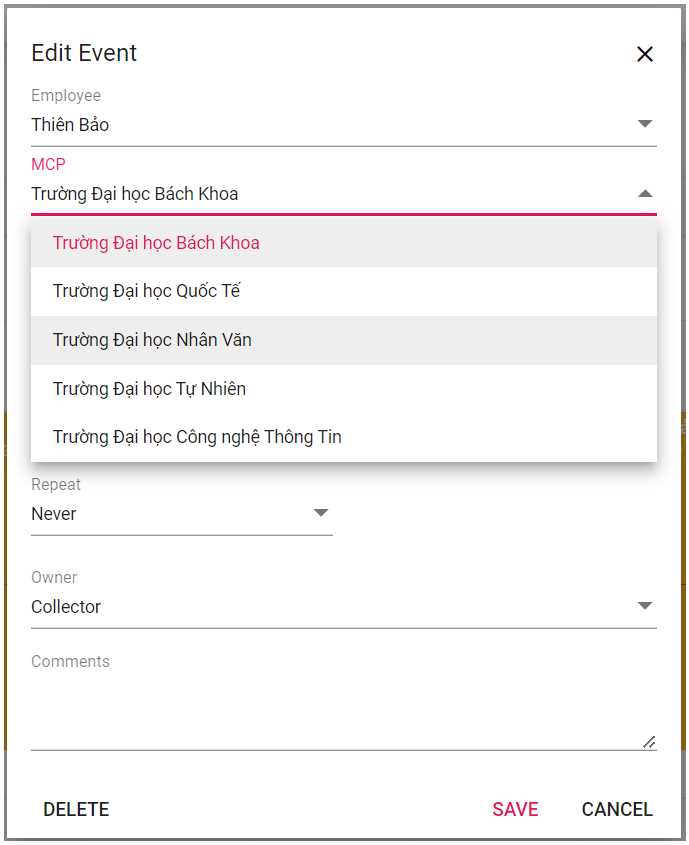


The main part of the function presented in item 4 includes 4 main functions:

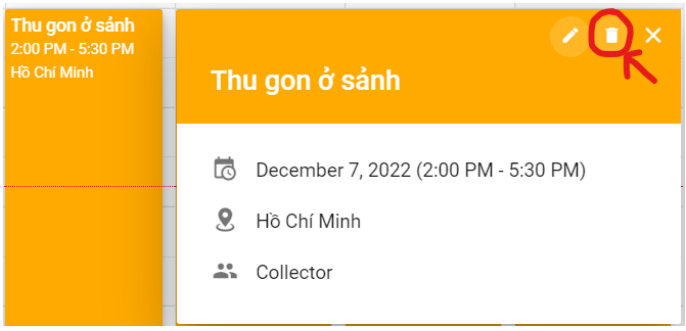
- End users can view previously created tasks.
- By clicking on any empty area, there will be 2 options: create a quick task or create a more detailed task, after clicking will bring a dialog box, from which a new task can be created.




- By double clicking on a previously created task, we can edit the task.




- By clicking on a created task we can view detail and delete it.



# VI Conclusion

After completing this assigment, we gained following knowledge and skills:

1. The process (maybe quite similar that in reality) of Software Engineering, from requirements analysis to implementation;
2. The ability to clarify what we need to do and what we can do i.e. what are exactly requirements and how much we can design and implement these;
3. The skills of using software development tools. We did not consider backend here, so there are, for example, ReactJS, Axios for frontend, socket.io for simple chat, and NPM for project structure managemnet.

Our part-of-software can be developed in the future, which can be to add a real database for it, instead of JSON one. Workflow can be seen at Our group's task assignements.

# VII References

1. For assumptions: Troller size, Vehicle size;
2. UML
3. Other internet resources for development tools.