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

SOFTWARE ENGINEERING (CO3001)

Assignment

Urban waste collection aid
UWC 2.0
Task 3: Architecture design

Advisor: Quản Thành Thơ - Nguyen Duc Anh
Student: Bùi Thái Dương - 1852307
Nguyễn Ngọc Hưng - 2053075
Đặng Quốc Huy - 2053031
Nguyễn Lê Thanh Phúc - 2052656
Nguyễn Việt Thắng - 2052719

3.1 Describe an architectural approach you will use to implement the desired system. How many modules you plan for the whole UWC 2.0 system? Briefly describe the input, output, and function of each module

3.1.1 Introduction to MVC pattern

Definition: Standing for "Model-View-Controller.", MVC is an application design model consisting of three interconnected parts. They include the model (data), the view (user interface), and the controller (processes that handle input). MVC is commonly used for developing modern user interfaces, providing fundamental functions to design programs for web, mobile or even desktop applications.

Below is the description of each aspect of MVC:

1. Model

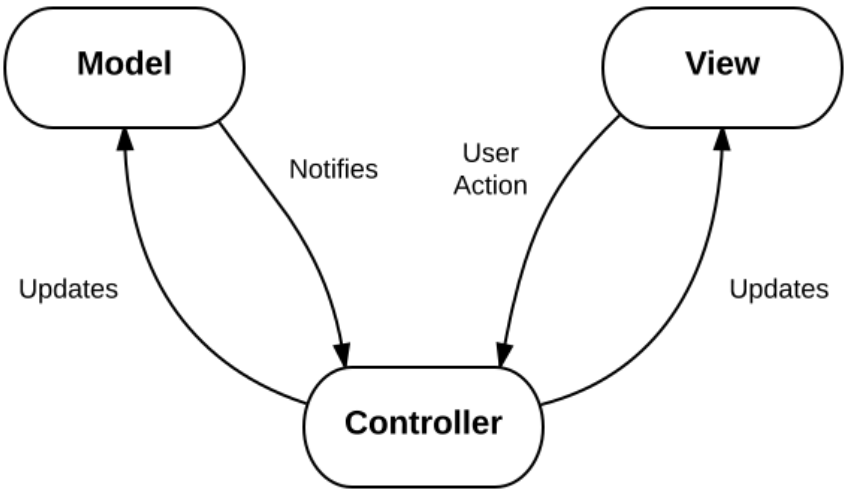
A model is data used by a program. This may be a database, file, or a simple object, such as an icon or a character in a video game.

2. View

A view is the means of displaying objects within an application. Examples include displaying a window or buttons or text within a window. It includes anything that the user can see.

3. Controller

A controller updates both models and views. It accepts input and performs the corresponding update. For example, a controller can update a model by changing the attributes of a character in a video game. It may modify the view by displaying the updated character in the game.



Pros and cons:

- **Pros:**

  - Faster development process
  - Ability to provide multiple views
  - Support for asynchronous technique
  - Modification does not affect the entire model
  - MCV model returns the data without formatting
  - SEO friendly Development platform

- **Cons:**

  - The complexity is high to develop the applications using this pattern.
  - Not suitable for small applications which have adverse effects on the application's performance and design.
  - In terms of servlet and JSP, both often contain business logic and presentation tier.
  - The isolated development process by UI authors, business logic authors and controller authors may lead to delays in their respective modules' development.

3.1.2 Applying to our model

To be more specific, we would like to explain some features of MVC architecture that we applied to our system:

Model: takes the data of the following classes from the database:

- *Account:* Each user will have their own account to log in to, the account will make it easier for them to assign tasks, receive notification about tasks, send/receive messages, view information,... Users must log in before interacting with the system.

- *Message:* The system tracks text messages by having message ID, contents, and time sent,... for each message sent by users.

- *MCPs:* Each MCPs has a unique ID with an address that helps the back officers and the workers to locate them to create route or to follow the route. Besides, there's also the availability of MCPs which is being updated every 15 minutes.

- *Vehicle:* Have vehicle ID, Name, availability, and fuel consumption for that back officers can choose the vehicle that suits the route.

- *Task:* Contain Task ID, requirements, responsible worker, route, date and time. Back officers can use this information to track the progress of tasks, workers can view this information, check in and out tasks while finishing their job.

View: the main graphical user interface to interact with the system. Here we break the view into many different view components, each for a different purpose:

- *Account view:* If users have not logged into the system, there will be a log-in box for the users to type their account name and password. If users are logged into the system, there will be a website interface that shows the accounts' information such as Name, Situation, DoB, Status, Address, Phone,....

- *Message view:* This website interface allows users to open a message box, choose another user to send message, send and view messages, report problems.

- *Vehicle view:* An interface that shows the vehicles' information: vehicle ID, Name, availability, and fuel consumption.

- *MCPs view:* Displays MCPs information: ID, Address, and availability.

- *Task view:* The system will have a form for the back officers to send tasks to specific workers based on the information about Vehicles and MCPs. There will be a Task review section for the back officers to review the tasks they sent.

Controller: Controller has the duty to identify user requests and render the corresponding view. The controller will contain all the logic flow of the UWC 2.0 system:

- *User controller:* The user controller will handle log-in and sign-up functions, manipulating account information. The user controller also has the duty to gather account data from Model and render account information view.

- *Task controller:* The task controller will handle task creation and sending requests from back officers and render task review through the task view component. This also helps record tasks into the database through the model component.

- *Message controller:* The message controller will handle message sending requests including chat box opening, message sending, and message reporting.

In our model, we include 3 modules:

Log-in:
The log-in module has the duty to allow users to provide their information, and helps the system manager and the back officers to track their working routine.

- Controller: User controller
- Input:
    - Login: Account name, password
    - Sign in: Account name, password, full name, address, phone
- Output: Account information view

Task Assignment:
The Task Assignment allows back officers to view the information of Vehicles, MCPs, workers and then create and send tasks to the workers through a submission form.

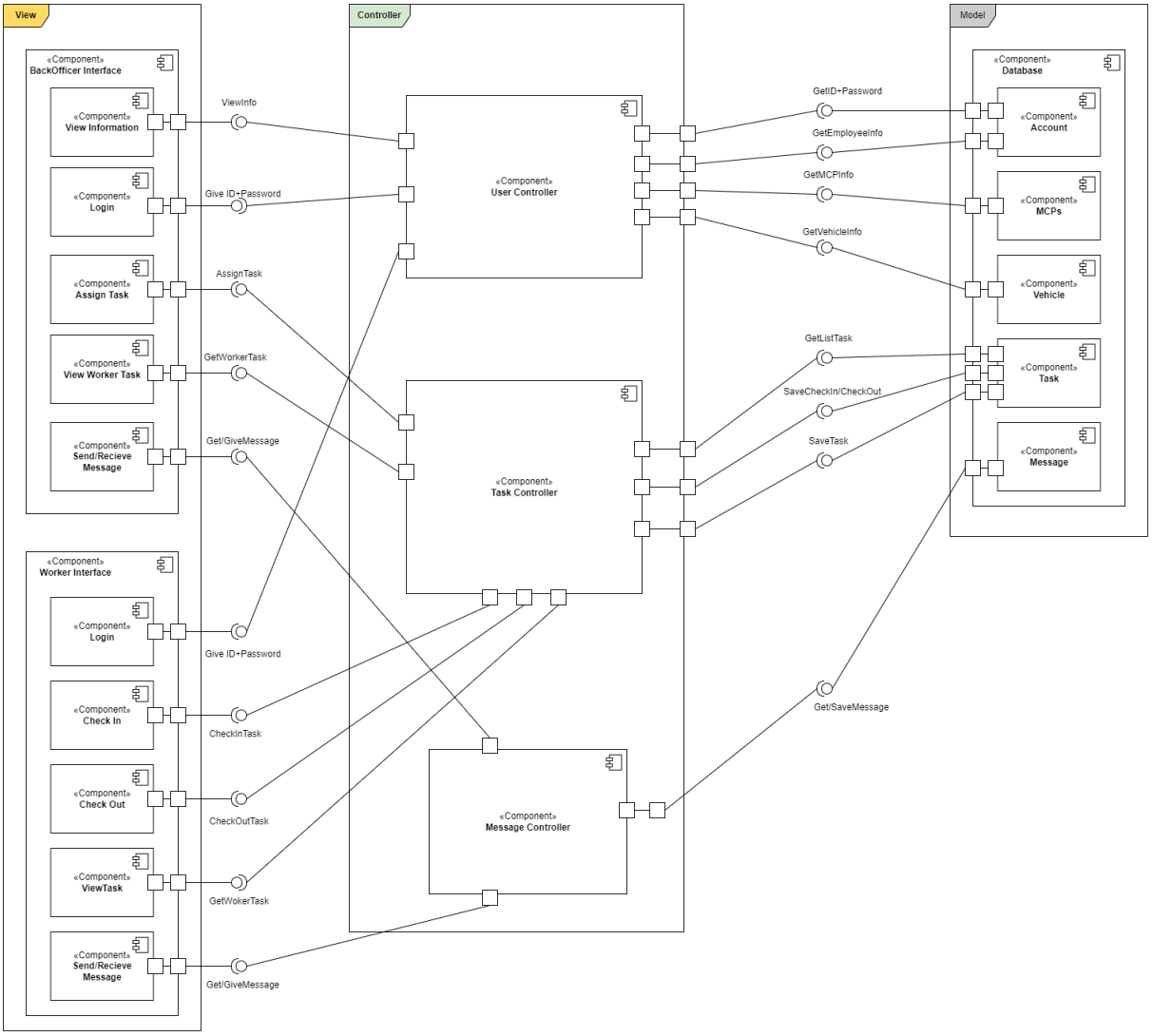
- Controller: Task controller
- Input:
    - Route, Vehicles, MCPs, Responsible worker, Task requirements
- Output: Tasks review

Send Message:
This module allows users to send messages to others, receive and view messages and report to the system manager when a system error occurs.

- Controller: Message controller
- Input:
    - Send: text messages
    - Report: report reason
- Output: Message view: Chatbox, sent and received messages

3.2 Draw an implementation diagram for the Task Assignment module

3.2.1 Component Diagram:



The system comprises three main sub-systems: View, Control, and Model.

The View component contains the Back_officer_Interface and Worker_Interface:

- The BackOfficer Interface component provides the interface for the back office to log in, view worker information, send/receive messages, assign and view worker tasks.
- The Worker Interface component provides the worker's interface to log in, check in, check out, send/receive messages and view worker tasks.

The Control component contains the user controller, task controller and message controller component.:

- User controller component provides the interface for log-in and sign-up functions, manipulating account information. Gather account data from Model and render account information view.

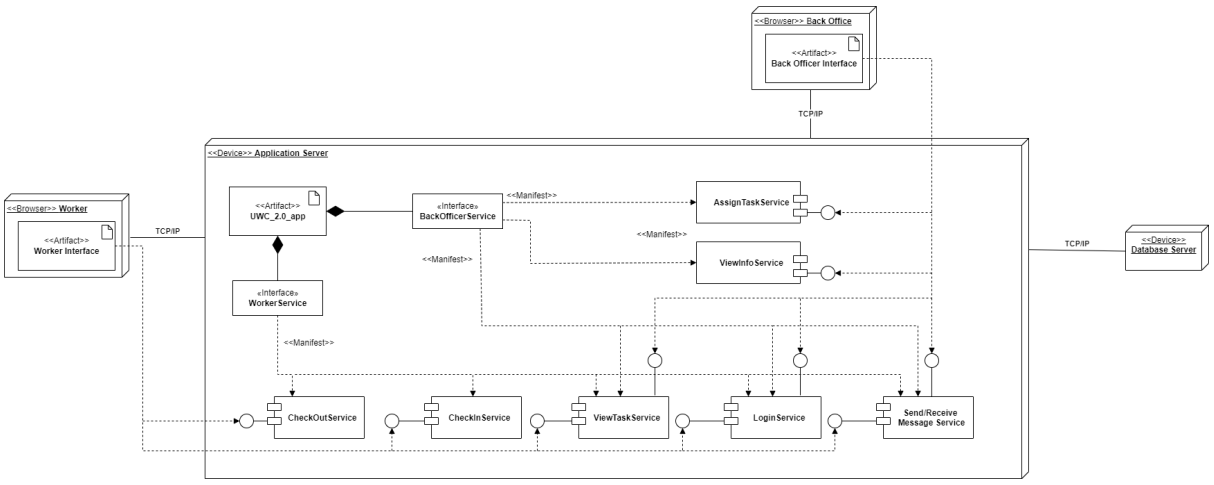
- Task controller component provides the interface for task creation, sending requests from back officers and rendering task review. This also helps assign tasks into the database.

- Message controller provides the interface for message sending requests including chat box opening, message sending, and message reporting.

The Model component contains a component Database with small components Employee, MCP, Vehicle, Task, Message:

- The Model component provides interfaces for the controller to perform operations with the database, providing interfaces for the view to update data when it receives a message from the model and tracks text messages.

3.2.2 Deployment diagram:



Back officer and the worker will connect to the app server by a web browser via protocol TCP/IP.

The app server will provide services and interface login, view worker's information, assign tasks, view tasks and send/receive messages for the back officer, login, view tasks, check in, check out and send/receive messages for the worker.

The app server will communicate with the database server via TCP/IP protocol.

Database server contains data about employees (Account), vehicle (Vehicle), MCP (MCPs) and tasks (Task).