VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
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

Advisor:	Phan Trung Hieu		
Group:	One		
Students:	Nguyen Tien Phat	- ID: 2011797	- L02
	Pham Thi Hong Hieu	- ID: 2020025	- L01
	Nguyen Hoang Quang Khanh	- ID: 2013458	- L02
	Nguyen Tuan Kiet	- ID: 2013577	- L02
	Nguyen Trung Nghia	- ID: 2010448	- L01
	Duong Thai Thinh	- ID: 2014589	- L02

HO CHI MINH CITY, FEBRUARY 2023

Member list & Workload

Task 3

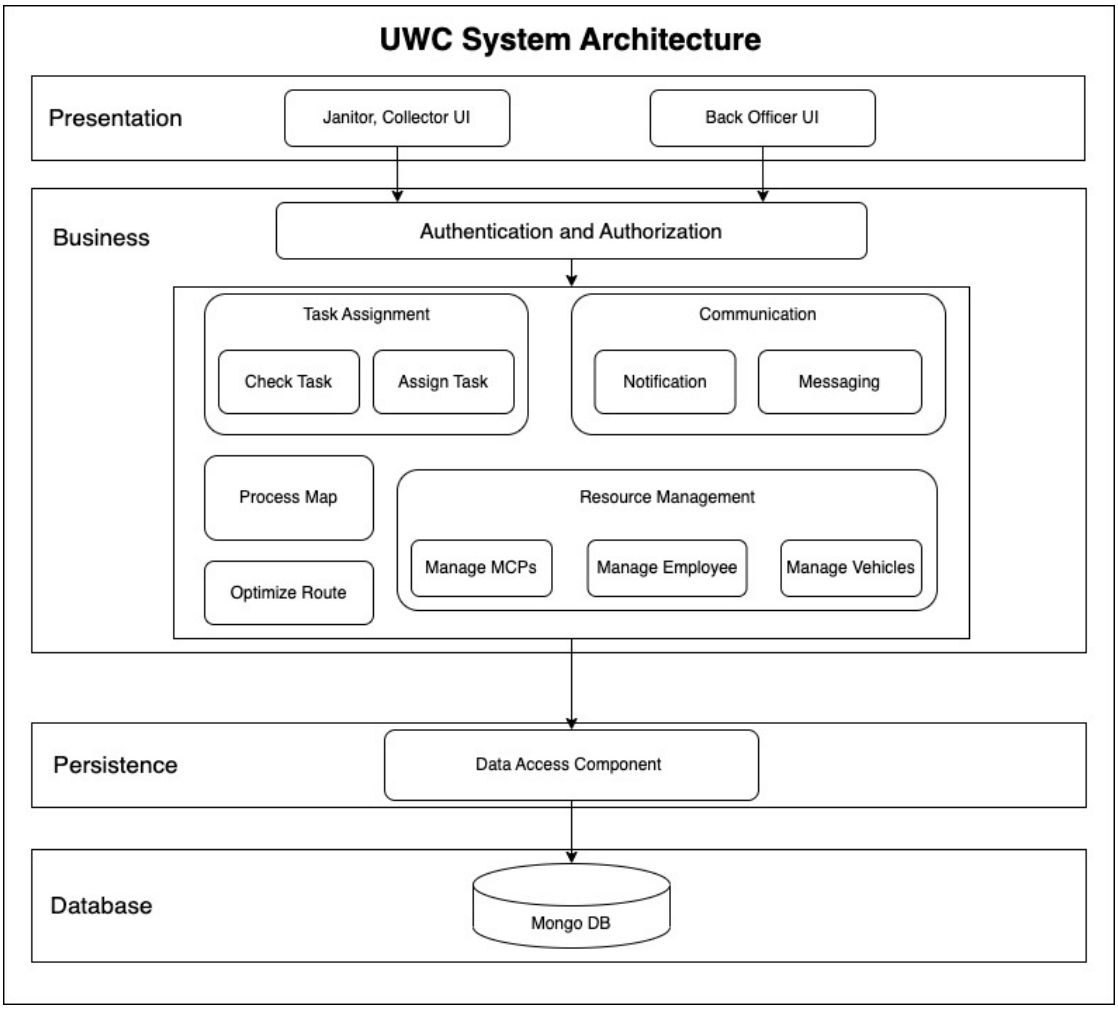
No.	Fullname	Student ID	Tasks
1	Nguyen Tien Phat	2011797	Deployment diagram
2	Pham Thi Hong Hieu	2020025	Component diagram
3	Nguyen Hoang Quang Khanh	2013458	Layer architecture
4	Nguyen Tuan Kiet	2013577	Component diagram
5	Nguyen Trung Nghia	2010448	Describe activity of system
6	Duong Thai Thinh	2014589	Deployment diagram

Contents

1 TASK 3

1.1 Architecture Design

1.1.1 Layer architecture for UWC 2.0



The system is designed in a layered architecture, with separate functions for each layer. The architecture was chosen consists of four layers: **presentation**, **business**, **persistence**, and **database**. In particular, each class performs a specific role in the application:

- **Presentation layer:** This is the layer with components responsible for displaying the user interface, interacting, sending requests, and then responding to the user. Includes: Janitor, Collector UI and Back Officer UI.
- **Business layer:** Responsible for performing business processing for the system's functional branches. After receiving the request from the Presentation layer, send a request to the Persistence layer to get data from the Database and that data will be processed according to the business logic and the result will be returned to the class in the Presentation layer. Including: authentication and authorization modules, map processing module, task management module, management module, communication module,...
- **Persistence layer:** This is an intermediate layer connecting the Business Logic and Data Access layers, making data access more tight and secure. This layer focuses on querying data without worrying about business logic. Includes: Data access modules.
- **Database layer:** Responsible for storing information about MCPs, User information, and Data of routes, ... Including Database Modules.

Each layer in the architecture is an abstract group of components that work around meeting a specific business requirement. This is the mechanism of isolation and encapsulation. At the same time, the function request will be sent from the upper layer to the adjacent lower layer one after the other, without crossing the layer. A functional branch (domain) is operated spread over 4 floors. That makes making changes at one layer less impactful to components in other layers and increases security, limits data access. This is one strength that led the team to choose this architecture.

Notable disadvantages of the system: The current system has the advantages of simplicity, ease of modification, limited access to information, and ease of upgrading to other architectures. However, there is still a decrease in performance when handling simple requests from users. The Presentation layer, for example, responds to requests to access information about a user's name or age. These requirements can be very simple and have no business logic handling. But the system will still push these requests through the tiers one after the other, and at each layer will do nothing but push the request to the next tier. Similar to the data return process, data will be returned from the Database through the floors, but not processed, added, removed, or calculated. This causes the system to consume resources and reduce performance.

In this project, most of the data must be calculated and processed through layers, for example, when assigning tasks to collectors, optimizing routes must be calculated, etc. Therefore, it makes perfect sense for the group to build the system according to the layered architecture

1.1.2 Describe activity of system

a) Describe how you will present your User Interface

The user will perform operations on the Back Officer UI or Janitor, Collector UI, then the presentation layer will send those requests to the business layer. In the business layer, user requests will have to be authenticated through authentication and authorization modules before being passed to functional modules in the same layer. For example, if a user wants to access the task assignment section, the authentication and authorization module will verify whether the user is a Back Officer, Collector, or Janitor to grant access to check or

assign tasks. Next, functional modules will request the necessary data by accessing the Data Access module in the persistence layer. At this module, it will forward the data request to the data component in the Database layer. Finally, the data will be sent back from the Database layer to Persistence, then to Business for functional modules to process the logic and send the data to the presentation layer to display for the user.

b) Describe how will you store your data

Since the database layer only communicates directly with the persistence layer, after going through the logical processing steps, the data layer will handle user requests through data access modules on the persistence layer. For data storage, data about user account information, MCP, or information that can be stored in the form of a table will be stored in a table and attached to a Key to retrieve or query the information. This will make the BO's information easier to retrieve. Information about maps, routes, schedules, tasks will be stored in the form of graphs. In addition, other requests such as direct messaging, assigning tasks or changing schedules will be done through the available modules. After completing the requirements, it will automatically check, send a notification to the object that has changed or has just updated new information, and then will update the user's data with the information previously received.

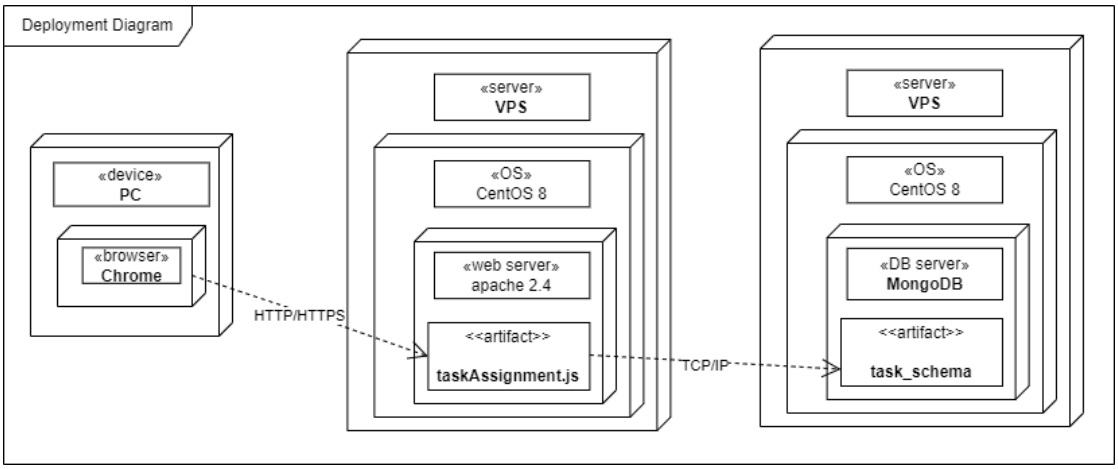
c) Describe how you will access external services/APIs

APIs can be used to facilitate communication and data exchange between the different layers of the UWC 2.0 system. With the layered architecture above. We created APIs to be used in the following ways: The presentation layer use APIs to send user inputs (display a list of waste collection routes and MCPs information) to the business logic layer, The business logic layer's API would handle this request by retrieving the relevant information from the data access layer and returning it to the presentation layer. The business logic layer uses APIs to request data from the data access layer or to store data in the system. For example, the business logic layer retrieves/sends information about waste collection schedules, routes, and MCPs. APIs be used to facilitate communication between different components of the system, such as between the route planning algorithm and the task assignment.

1.2 Component diagram for the Task Assignment module

1.2.1 Deployment diagram

Diagram



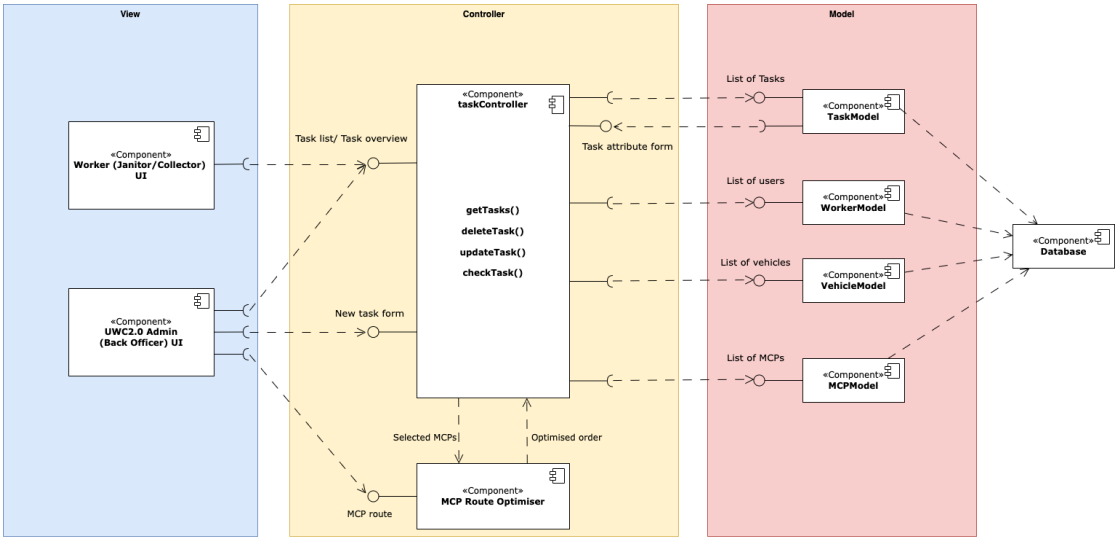
Description

With the MVC architecture, the hardware system for the application is divided into 3 corresponding hardware blocks as follows:

- Model: The task_schema contains the data of the implemented task in the MongoDB system, using the CentOS 8 operating system on a VPS.
- Controller: The application is packaged into the taskAssignment.js executable file running on the Apache web server version 2.4, which runs on the CentOS 8 operating environment in another VPS. The application file retrieves data from the schema via the TCP/IP protocol.
- View: Users interact with the application through the interface on the Chrome browser on a personal computer. The interface communicates with the executable file through the HTTP/HTTPS protocol.

1.2.2 Component diagram

Diagram



Description

The component diagram for the Task Assignment module is based on Model–view–controller (MVC) architecture.

- Model: Model components represent the structured database.
- Controller: Controller components interact with both view and model components. Each controller component executes actions from the view components, including fetching data, modifying/assigning data, and sending notifications.
- View: View components receive data from controller components and display it. The UWC2.0 Admin screen allows back office officers to view a summary of all tasks using time filters (i.e. status, notification of new updates, statistics).

The relevant components in the architecture diagram are implemented in this component diagram:

- View section contains two UIs from Presentation layer
- Controller section demonstrates Task Assignment component in Business layer
- Model section is corresponding to Data Access Management in Persistence layer

In addition, it provides important parameters for four main objects: MCPs, collectors/janitors, vehicles, and tasks. Back officers can click on the menu to access a dedicated management screen for each object, including the MCPs Map screen, Worker Status screen, Vehicle Status screen, and Task List screen.

2 Bibliography