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

Tăng Tuấn Đạt – 2152512

Tạ Gia Khang – 2152652

Nguyễn Chánh Tín - 2153043

Table of Contents:

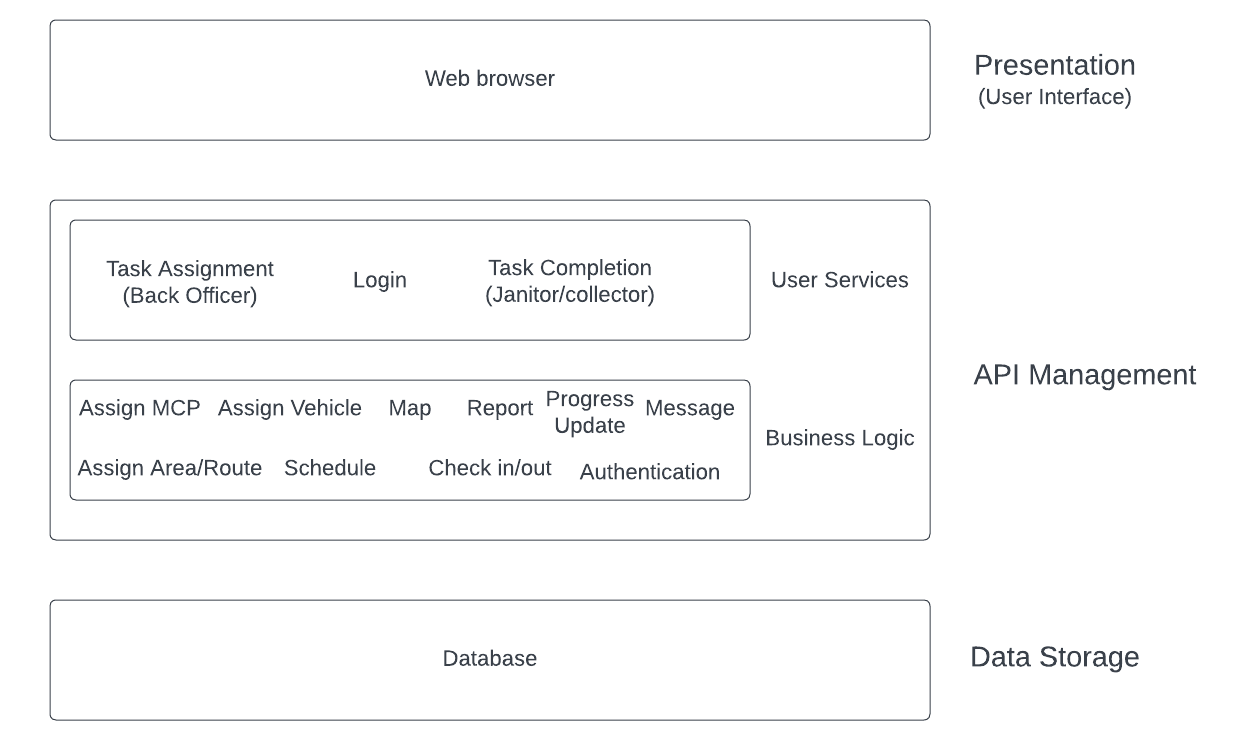
Chapter 2: **System modeling 3**

**2.1: Architecture Diagram 3**

**2.2: Component diagram: 5**

Chapter 3: **Architecture design**

**3.1: Layered Architecture Diagram**

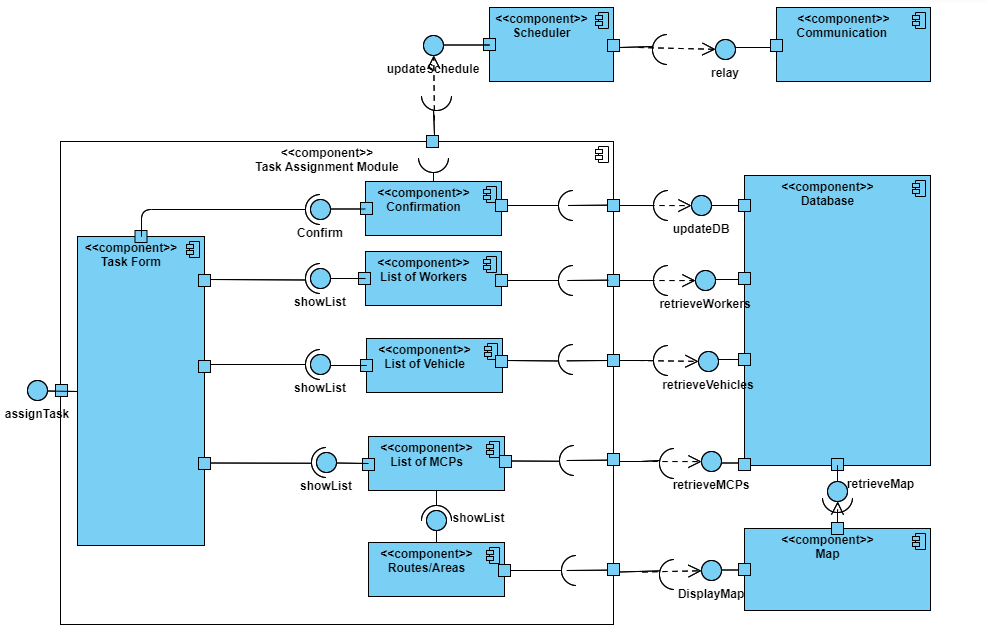


In the presentation layer, we will use a combination of Golang and HTML to design a user-friendly and aesthetically pleasing interface. The website shall support Vietnamese with the option to change between Vietnamese and English in the future using the “lang” attribute in HTML. UWC 2.0 also supports automatic scaling with different devices and works seamlessly with the zoom mode of mayor 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.

In the API management layer, we will only use a few APIs like mapAPI and mongoDB, so it is no need to use a dedicated API management like Azure. 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. 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, MongoDB will be used for the UWC 2.0 system. MongoDB is a NoSQL document-oriented database that provides high scalability, performance, and flexibility. The Task Management module of the system will store all the relevant data related to janitors, collectors, vehicles, MCPs, and task assignments in a MongoDB database. MongoDB'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, MongoDB'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. The database layer will be accessed using a backend API built using Node.js and Express.js, providing a secure and efficient way to manage data transactions between the Presentation layer and the database. However, in this part, we will hard code the database.

**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>>.