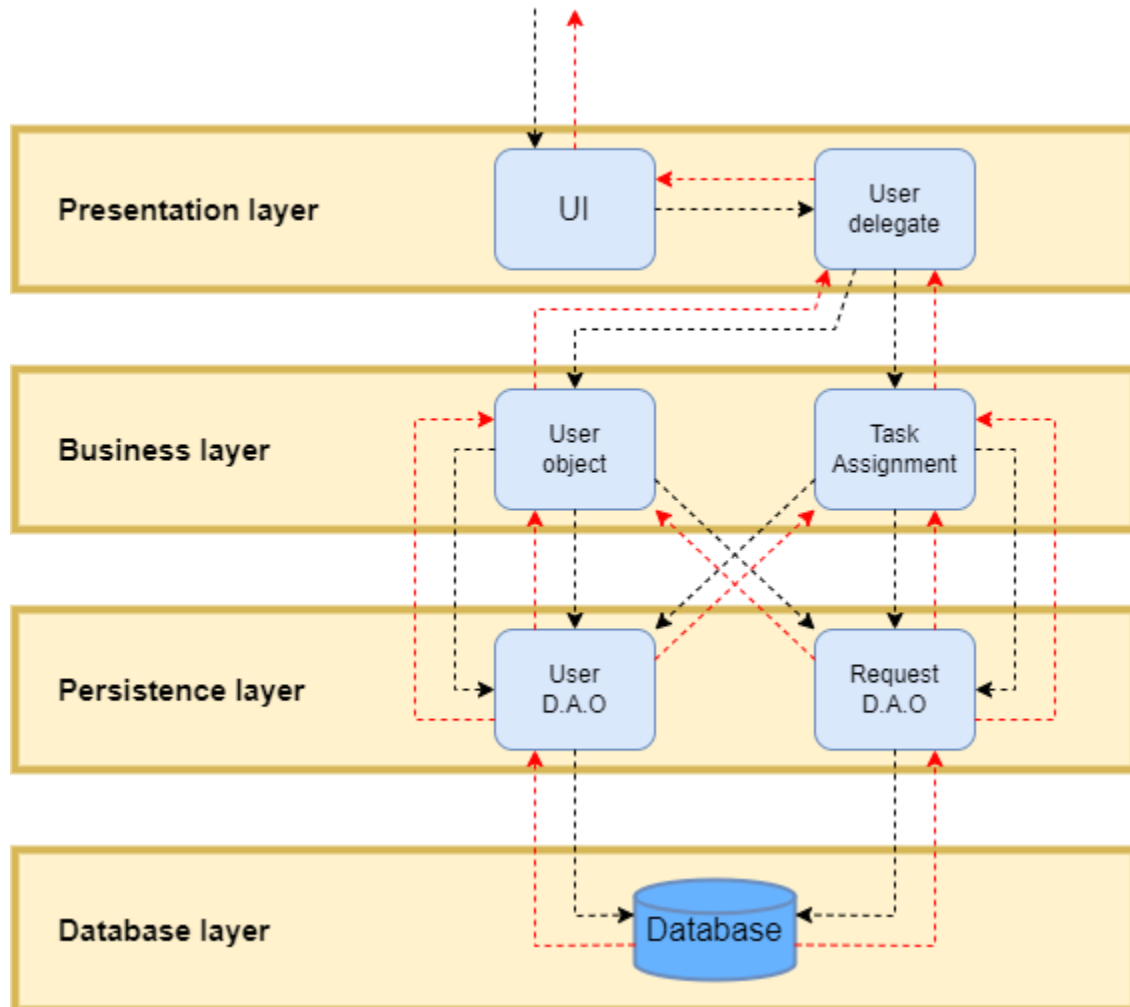## Task 3.1

In order to implement the whole UWC 2.0 system, we will illustrate the system using layered architecture as below:

The black arrows represent the request flowing down to retrieve a user's specific desired data, and the red arrows represent the response flowing back to the UI.

The entire system will be broken down into 7 modules. The **UI** will receive the request of the user (back officer or front employee) as input and display the data to the user as its output. The **User delegate module** decides which module is needed to process the request and what data it needs. After receiving the request, the **User delegate module** then passes it down to either the **User object module** or the **Task Assignment module** based on the user's request. The **user object module** is used to retrieve all the information from the **Database** about

staff, MCPs or vehicles if a back officer needs to view them in particular. The **Task Assignment module** is responsible for creating tasks and routes and assigning them to the front employees after getting information from the **Database** about employees' work schedule, available vehicles and MCPs. The **User Data Access Object module** receives requests from the Business-Layer modules and executes SQL queries to retrieve needed information from the **Database** and the **Request Data Access module** executes the similar protocol to retrieve the information about the request. Once the execution is complete, the data will be passed through layers back to the **UI** to be displayed to the user.

Task 3.2

Deployment diagram of Task Assignment module:

workCheckSystem

checkin(): void
checkout(): void

viewTask

view(): void

Logout

Initializes Login system

Messaging system

sendMessage(): void
searchUser(): void

Employee panel

Initializes checkin/checkout
Initializes view task
Initializes messaging
Log out

Login system

Initializes Emp panel
Initializes Admin panel
Register User

Officer panel

Initializes Manage staffs
Initializes Manage vehicles
Initializes Manage MCPs
Initializes Manage tasks
Initializes messaging
Logout

manageStaff

viewStaff(): void
removeStaff(): void

manageVehicle

addVehicle(): void
removeVehicle(): void
viewVehicle(): void

manageMCP

addMCP(): void
removeMCP(): void
updateMCP(): void
viewMCP(): void

manageTask

addTask(): void
removeTask(): void
assignTask(): void
viewTask(): void