VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY

## Software Engineering CC05

## Report Task 3 (Semester 221)

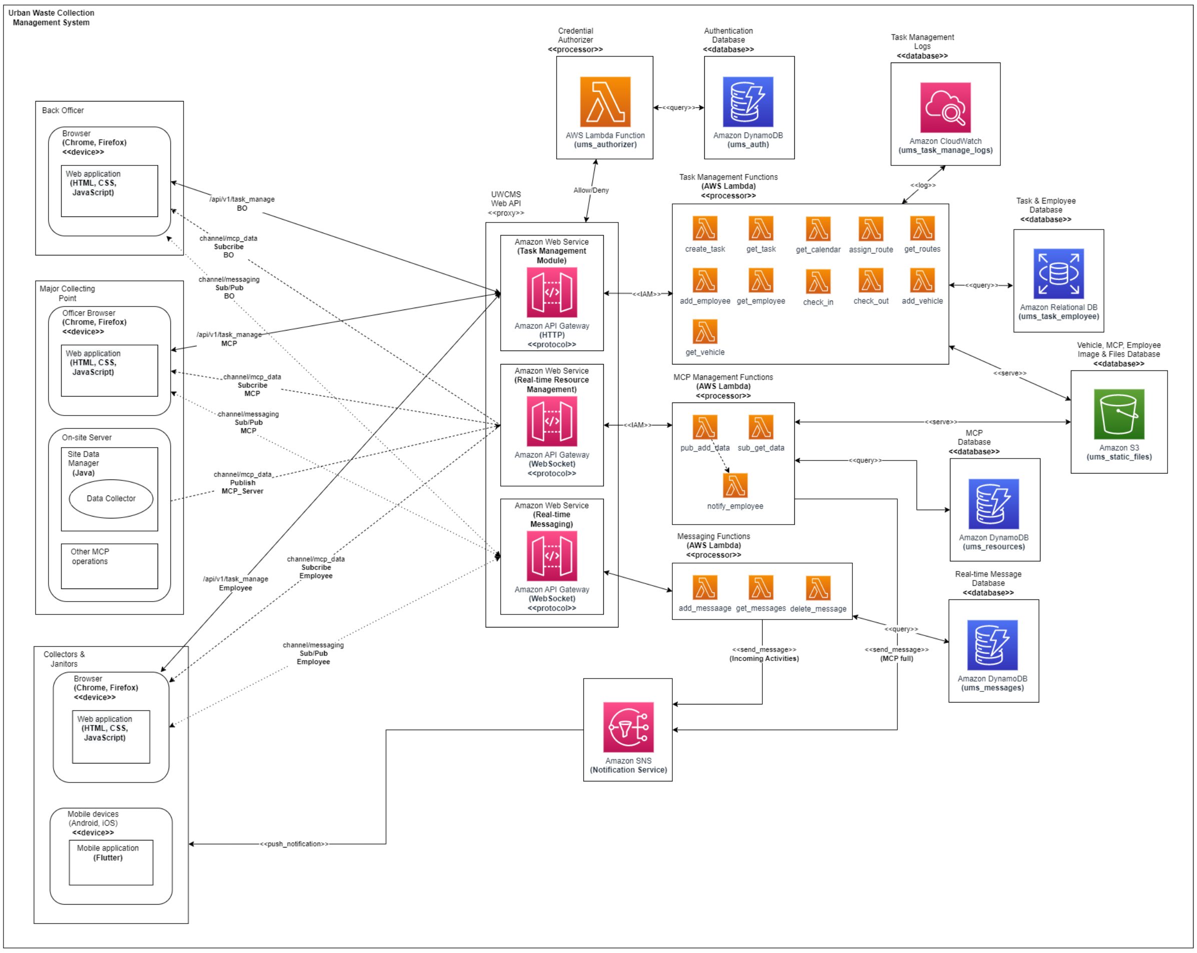
# GROUP Ố DỀ

Lecturer: Trương Tuấn Anh

| Student's name | Student's ID |
|---|---|
| Nguyễn Xuân Bách | 2052864 |
| Nguyễn Thanh Bảo Danh | 2052416 |
| Trần Quang Kiệt | 2052563 |
| Trần Trung Nguyên | 2052196 |
| Mai Minh Nhật | 2053295 |

HO CHI MINH CITY, September 2022

# UWC Architecture


Urban Waste Collection Management System
Back Officer
Browser (Chrome, Firefox) <<device>>
Web application (HTML, CSS, JavaScript)
Major Collecting Point
Officer Browser (Chrome, Firefox) <<device>>
Web application (HTML, CSS, JavaScript)
On-site Server
Site Data Manager (Java)
Data Collector
Other MCP operations
Collectors & Janitors
Browser (Chrome, Firefox) <<device>>
Web application (HTML, CSS, JavaScript)
Mobile devices (Android, iOS) <<device>>
Mobile application (Flutter)
/api/v1/task_manage BO
channel/mcp_data Subcribe BO
channel/messaging Sub/Pub BO
/api/v1/task_manage MCP
channel/mcp_data Subcribe MCP
channel/messaging Sub/Pub MCP
channel/mcp_data Publish MCP_Server
/api/v1/task_manage Employee
channel/mcp_data Subcribe Employee
channel/messaging Sub/Pub Employee
<<push_notification>>
Credential Authorizer <<processor>>
AWS Lambda Function (ums_authorizer)
<<query>>
Authentication Database <<database>>
Amazon DynamoDB (ums_auth)
Allow/Deny
UWCMS Web API <<proxy>>
Amazon Web Service (Task Management Module)
Amazon API Gateway (HTTP) <<protocol>>
Amazon Web Service (Real-time Resource Management)
Amazon API Gateway (WebSocket) <<protocol>>
Amazon Web Service (Real-time Messaging)
Amazon API Gateway (WebSocket) <<protocol>>
<<IAM>>
<<IAM>>
Task Management Logs <<database>>
Amazon CloudWatch (ums_task_manage_logs)
<<log>>
Task Management Functions (AWS Lambda) <<processor>>
create_task
get_task
get_calendar
assign_route
get_routes
add_employee
get_employee
check_in
check_out
add_vehicle
get_vehicle
<<query>>
Task & Employee Database <<database>>
Amazon Relational DB (ums_task_employee)
<<serve>>
Vehicle, MCP, Employee Image & Files Database <<database>>
Amazon S3 (ums_static_files)
MCP Management Functions (AWS Lambda) <<processor>>
pub_add_data
sub_get_data
notify_employee
<<serve>>
<<query>>
MCP Database <<database>>
Amazon DynamoDB (ums_resources)
Messaging Functions (AWS Lambda) <<processor>>
add_messaage
get_messages
delete_message
<<query>>
Real-time Message Database <<database>>
Amazon DynamoDB (ums_messages)
<<send_message>> (Incoming Activities)
<<send_message>> (MCP full)
Amazon SNS (Notification Service)

# Overview

| Category | Module | Technology | Input/Output | Description |
|---|---|---|---|---|
| Client | Web application | JavaScript *(React)*, HTML, CSS | Allow BO, MCP managers, employees to interact with the system | |
| Mobile application | Flutter | | | |
| MCP on-site server | Java | • Collects MCP data *(capacity,* | Runs 24/7, sends data periodically | |

| | | | | |
|---|---|---|---|---|
| | | *date & time, system logs,…)* • Send data to back-end processing servers | every 15 minutes | |
| Back-end | Task Management Module *(TMM)* | • Amazon API Gateway • AWS Lambda Function | 1. **Input:** User requests from client-side applications *(task creation, vehicle/employee searches)* 2. **Output:** Required data corresponding to user request in TMM | Uses HTTP protocol |
| Real-time Resource Management *(RRM)* Web API | • Amazon API Gateway • AWS Lambda Function | 1. **Input:** Collected real-time data from MCP sites; user requests from client-side applications *(view capacity, operation logs)* 2. **Output:** Required data corresponding to user request in RRM | Uses WebSocket | |
| Real-time Messaging *(RTM)* Web API | • Amazon API Gateway • AWS Lambda Function | 1. **Input:** User messages; user requests from client-side applications *(get messages)* 2. **Output:** Required conversations corresponding to user request in RTM | | |
| Credential Authorizer Web | • Amazon API Gateway • AWS | 1. **Input:** Login credentials | Authenticate all requests from | |

| | | | | |
|---|---|---|---|---|
| API | Lamda Function | *(email, password)*; Authentication key 2. **Output:** Corresponding authentication key *(such as JSON Web Tokens)*; Authentication validation result *(correct password?)* **(Allow/Deny)** | users/clients | |
| Databases | Authentication Database | Amazon DynamoDB | 1. **Input:** Database queries from *Credential Authorizer* 2. **Output**: Corresponding user information *(email, hashed passwords, roles,…)* | • Stores authentication information • Is a NoSQL, document-based database, therefore have fast queries |
| Task & Employee Database | Amazon RDS | 1. **Input:** Database queries from *Task Management Module Web API* 2. **Output**: Corresponding task, vehicle & employee information *(name, address, calendar, daily tasks, vehicle specification,…)* | • Stores employee, tasks and vehicle information • Is a relational database, because its data have deeply connected relationships | |
| MCP Database | Amazon DynamoDB | 1. **Input:** Database queries from *Real-time Resource Management* | • Stores collected data of MCP • Is a NoSQL, document-based database, therefore have fast queries | |

| | | | | |
|---|---|---|---|---|
| | | *Web API* 2. **Output**: Corresponding MCP status data *(MCP current capacity, usage, operation logs…)* | | |
| Messaging Database | Amazon DynamoDB | 1. **Input:** Database queries from *Real-time Messaging Web API* 2. **Output**: Corresponding message data *(messages, time sent, activities, files,…)* | • Stores messaging data • Is a NoSQL, document-based database, therefore have fast queries | |
| Vehicle, Employee, Task & Messaging Static File Database | Amazon S3 | 1. **Input:** Database queries from *Web APIs* 2. **Output**: Corresponding static files *(images, PDF files, HTMLs, audio files,…)* | • Stores static files | |
| | Operation Logging System | Amazon CloudWatch | Logs all activities in the back-end systems *(transactions, requests/responses, uptime,…)* | • Is natively supported by all other tools in the same Amazon ecosystem • Used by back officers, system admins, developers, data analysts |
| Notification System | Amazon SNS *(Simple Notification System)* | 1. **Input:** Notification requests from *Real-time Resource* | | |

| | | | | |
|---|---|---|---|---|
| | | *Management & Real-time Messaging Web API* 2. **Output**: Push notifications sent to mobile application of collectors & janitors | | |

# Description

| Properties | Description |
|---|---|
| Uses all cloud services & infrastructure | • Helps scale horizontally and save development, maintainance and operation costs while doing so. Can handle massive amount of traffic • No open-source licensing issues, unlike when maintaining the infrastructure ourselves *(servers, load balancers,…)* • Reduces amount of employee required to maintain the project |
| Uses all Amazon services | • AWS *(Amazon Web Services)* ecosystem is widely used by corporations & projects • AWS have a huge catalog of cloud native services and therefore can cover additional future features • AWS supports a wide-range of programming languages for their services like Java, Go, Node.js,… • AWS services are deeply integrated and well orchestrated, therefore reduces development time and compatibility issues |