



## Cloud Native Architecture and Development

Year 2/3 (2024) Semester 2

### SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in Information Technology

## ASSIGNMENT 1

Duration	: Week 6 to 8
Weightage	: 30% of total coursework
Individual/Team	: Individual

**Due on Sunday, 8 Dec 2024, 11.59 PM (Week 8)**

**Penalty for late submission:**

10 marks per day (including Sunday and public holiday)

No submission will be accepted after **15 Dec 2024 11.59 PM**.

**WARNING**

*If a student is found to have submitted work not done by him/her, he/she will not be awarded any marks for this assignment. Disciplinary action will also be taken.*

*Similar action will be taken for the student who allows other student(s) to copy his/her work.*

## **1. OBJECTIVES**

- To demonstrate ability to develop REST APIs in Go
- To make conscientious consideration in designing microservices architecture, database management, and system implementation

## **2. INTRODUCTION**

In an era marked by sustainable transportation and shared economies, electric car-sharing platforms have emerged as a cornerstone of modern urban mobility. This project aims to design and implement a fully functional electric car-sharing system using Go, with features catering to diverse user needs and real-world application scenarios. With an emphasis on practical and scalable solutions, the system includes user membership tiers, promotional discounts, and an accurate billing mechanism.

## **3. BASIC REQUIREMENTS**

### **3.1. User Management**

- 3.1.1. User Registration and Authentication: Implement user registration with email or phone verification. Users should be able to log in with secure authentication, including password encryption.
- 3.1.2. Membership Tiers: Develop multiple membership levels (e.g., Basic, Premium, VIP) with different benefits, such as reduced hourly rates, priority vehicle access, or increased booking limits.
- 3.1.3. User Profile Management: Allow users to update personal details, view membership status, and track rental history.

### **3.2. Vehicle Reservation System**

- 3.2.1. Car Availability and Booking: Enable users to view available vehicles in real-time and make reservations for a specified time range.
- 3.2.2. Booking Modification and Cancellation: Users should have the option to modify or cancel reservations within specified policies, with automatic updates to availability.

- 3.2.3. (BONUS) Vehicle Status Tracking: Implement mechanisms to track the location, charge level, and cleanliness of each vehicle to ensure readiness for the next rental.

### **3.3. Billing and Payment Processing**

- 3.3.1. Tier-Based Pricing and Discounts: Calculate billing based on membership level, duration of rental, and applicable promotional discounts.
- 3.3.2. Real-Time Billing Calculation: Display estimated costs before booking confirmation and provide real-time cost updates during the rental.
- 3.3.3. (BONUS) Payment Processing: Integrate any forms of secure payment to process payments, handle refunds for cancellations, and store payment methods with standards.
- 3.3.4. Invoicing and Receipts: Automatically generate detailed invoices after each rental and send them via email or make them available on the user's profile.

### **3.4. Microservices Architecture**

- 3.4.1. Service Decomposition: Split the system into distinct services (e.g., User Service, Vehicle Service, Billing Service) to enhance scalability and maintainability.
- 3.4.2. Inter-Service Communication: Use RESTful APIs for communication between services, with clear API documentation for each.
- 3.4.3. (BONUS) Service Orchestration and Load Balancing: Any creative forms of implementations are welcome.

### **3.5. Database Management**

- 3.5.1. Database Schema Design: Use a relational database for structured data, including users, vehicles, reservations, billing, and promotions. Ensure data normalisation to minimise redundancy.
- 3.5.2. (BONUS) Caching and Optimisation: Any creative forms of implementations are welcome.

## 4. DELIVERABLES

- 4.1. Submit the following via GitHub (add your tutor as collaborator),
  - 4.1.1. All source codes of your assignment
  - 4.1.2. SQL script for setting up your database
  - 4.1.3. A readme.md indicating
    - 4.1.3.1. Design consideration of your microservices
    - 4.1.3.2. Architecture diagram
    - 4.1.3.3. Instructions for setting up and running your microservices
- 4.2. Submit a 15-minute presentation of the design and implementation of your assignment. **Record your presentation and submit via Brightspace.**

## 5. MARKING SCHEME

Component	Grade	Criteria
Implementation (80%)	A	<ul style="list-style-type: none"> <li>• Demonstrated good design of microservices</li> <li>• Implemented all features of the assignment</li> <li>• Included comments and brief description for the codes</li> <li>• Demonstrated good programming practices</li> </ul>
	B	<ul style="list-style-type: none"> <li>• Demonstrated good design of microservices</li> <li>• Implemented some features of the assignment</li> <li>• Included some comments and brief description for major parts of the codes</li> <li>• Demonstrated some good programming practices</li> </ul>
	C - F	<ul style="list-style-type: none"> <li>• Implemented some features of the assignment</li> </ul>
Readme (10%)	A	<ul style="list-style-type: none"> <li>• Comprehensive documentation of the design of microservice architecture</li> </ul>
	B	<ul style="list-style-type: none"> <li>• Good documentation of the design of microservice architecture</li> </ul>
	C - F	<ul style="list-style-type: none"> <li>• Basic documentation and brief description of design</li> </ul>
Presentation (10%)	A	<ul style="list-style-type: none"> <li>• Detailed description of the microservices</li> <li>• Sound reasoning for microservice design</li> <li>• Demonstrated excellent understanding of work</li> </ul>
	B	<ul style="list-style-type: none"> <li>• Detailed description of the microservices</li> <li>• Brief reasoning for microservice design</li> <li>• Demonstrated some understanding of work</li> </ul>
	C - F	<ul style="list-style-type: none"> <li>• Brief description of the microservices</li> <li>• Brief reasoning for microservice design</li> </ul>
Bonus Marks (10%)		<ul style="list-style-type: none"> <li>• Implemented all bonus assignment requirements</li> </ul>