Assignment Guideline: Automobile Service Time Logging & Appointment System

Objective

The purpose of this assignment is to test your knowledge of enterprise application development by designing and implementing a fully functional **Automobile Service Management System**. The application should allow customers to view their vehicle service progress, book appointments, request modifications, and interact with employees through modern interfaces. Employees should be able to log time for services/projects, track progress, and monitor workloads.

This group assignment will assess your ability to design, implement, containerize, test, and deploy a modern enterprise application.

Weightage: 20 marks.
Group Size: 10 members

Assignment Overview

You are tasked with developing an application for an automobile company with the following features:

1. **Customer Functionality**
   - Secure login & signup.
   - Dashboard to **view service/project progress** in real-time.
   - **Book an appointment** for a vehicle service.
   - **Request a modification** (treated as a vehicle project).
   - **Mobile-friendly updates** for service/project progress.
   - **Chatbot integration (Bonus)** using a Generative AI model to check available service slots (+5 marks).

2. **Employee Functionality**
   - Login & authentication.
   - Ability to **log time against each project/service**.
   - Track progress and update status of ongoing services/projects.

- Geo-Location Services - Locate nearest service center
- Shift Scheduling & Resource Management
- Inventory Management
- Notifications & Communication

    - View upcoming appointments and requests.

3. **System Requirements**
    - Real-time updates (e.g., using WebSockets, server-sent events, or polling).
    - Containerized deployment (frontend + backend + database).
    - Proper role-based access control (customers vs. employees/admins).
    - Secure handling of user and project/service data.

Requirements

1. Design Documentation (20 marks)

- **Use Case Diagrams** for customer and employee interactions.
- **ER Diagram** for database structure (customers, vehicles, services, projects, time logs, employees).
- **Architecture Proposal** (e.g., microservices, layered architecture) with justification.
- **Sequence/Activity Diagrams** for key flows (booking service, logging time, chatbot query).

2. Implementation (50 marks)

- **Frontend:** Develop a responsive UI using modern frameworks (React, Angular, or Vue).
- **Backend:** Implement core logic using suitable frameworks (Node.js/Express, Spring Boot, Django, etc.).
- **APIs:** Secure REST/GraphQL APIs for customer and employee operations.
- **Real-Time Updates:** Service progress tracking via websockets/SSE.
- **Containerization:**
    - Dockerize both frontend and backend.
    - Use `docker-compose` for local orchestration.
- **Bonus:** Deploy the containerized system on Kubernetes (add Helm manifests or YAML).

3. Testing (15 marks)

- Add **unit and integration tests** for backend services.
- Ensure code coverage is measurable.
- Include test results in the submission.

4. Presentation/Demo (15 marks)

- Live demo of core features:

    - Customer login, booking, progress tracking.
    - Employee login and time logging.
    - Real-time updates in action.
    - (Optional) Chatbot interaction.

- Walkthrough of architecture, diagrams, and design choices.

Bonus Marks

- **Kubernetes Deployment (+5 marks):**
  Deploy the application (frontend, backend, and database) on Kubernetes. Use ConfigMaps, Secrets, and Service definitions.

- **Generative AI Chatbot (+5 marks):**
  Integrate a simple chatbot that leverages AI to check available appointment slots.