Project Proposal

Project Title

Scalable E-Commerce / Information System Using Spring Boot Microservices Architecture

Course: Software Design & Construction (SDC)
Instructor: Mr. Zunnurain
Teaching Assistant: Umair Makhdoom

Team Members

- **BSSE23011** – Shahzaman
- **BSSE23060** – Saram
- **BSSE23074** – Rai

1. Introduction

Modern software systems are expected to be highly scalable, maintainable, and resilient to change. Traditional monolithic architectures, where all system components are tightly coupled and deployed as a single unit, often struggle to meet these demands as applications grow in size and complexity.

This project proposes the design and implementation of a **scalable E-Commerce / Information System** using a **Spring Boot–based Microservices Architecture**. The system is decomposed into independent, loosely coupled services that communicate through RESTful APIs. Core infrastructure components such as **Service Discovery** and an **API Gateway** are used to manage inter-service communication and request routing, enabling efficient scalability and easier maintenance.

2. Problem Statement

In monolithic application architectures, all functional modules—such as user management, product handling, and data processing—are bundled into a single executable unit. As new features are added, the codebase becomes increasingly complex and difficult to maintain.

Moreover, scaling such systems requires replicating the entire application, even when only a specific module experiences high load.

This tight coupling leads to challenges in deployment, testing, fault isolation, and scalability. Any failure or update in one module can impact the entire system. Therefore, there is a need for a more modular and scalable architectural approach.

3. Proposed Solution & System Architecture

To overcome the limitations of monolithic systems, this project implements a **microservices-based backend architecture** accompanied by a **responsive frontend interface**.

Key Architectural Components

- **Service Discovery:**
  Responsible for registering and managing all microservices, enabling dynamic discovery and communication without hard-coded service locations.

- **API Gateway:**
  Acts as a single entry point for all client requests. It routes incoming requests to the appropriate microservices, handles load balancing, and simplifies client-side interactions.

- **Microservices:**
  Independent Spring Boot services, each responsible for a specific domain or business capability (e.g., Billing, Patient, Auth). Each service can be developed, deployed, and scaled independently.

- **Frontend Application:**
  A web-based user interface that consumes backend APIs via the API Gateway, providing a seamless user experience.

This architecture ensures high scalability, improved fault tolerance, and better maintainability while supporting future system expansion.

4. Technology Stack

Backend

- Java 17

- Spring Boot
- Spring Cloud (Eureka Server, API Gateway)

Frontend

- Web Technologies (HTML, JavaScript, and related frameworks)

Deployment

- Amazon Web Services (AWS)
- EC2 Instances for hosting services

Development & Testing Tools

- Maven (Build & Dependency Management)
- Git (Version Control)
- Postman (API Testing)