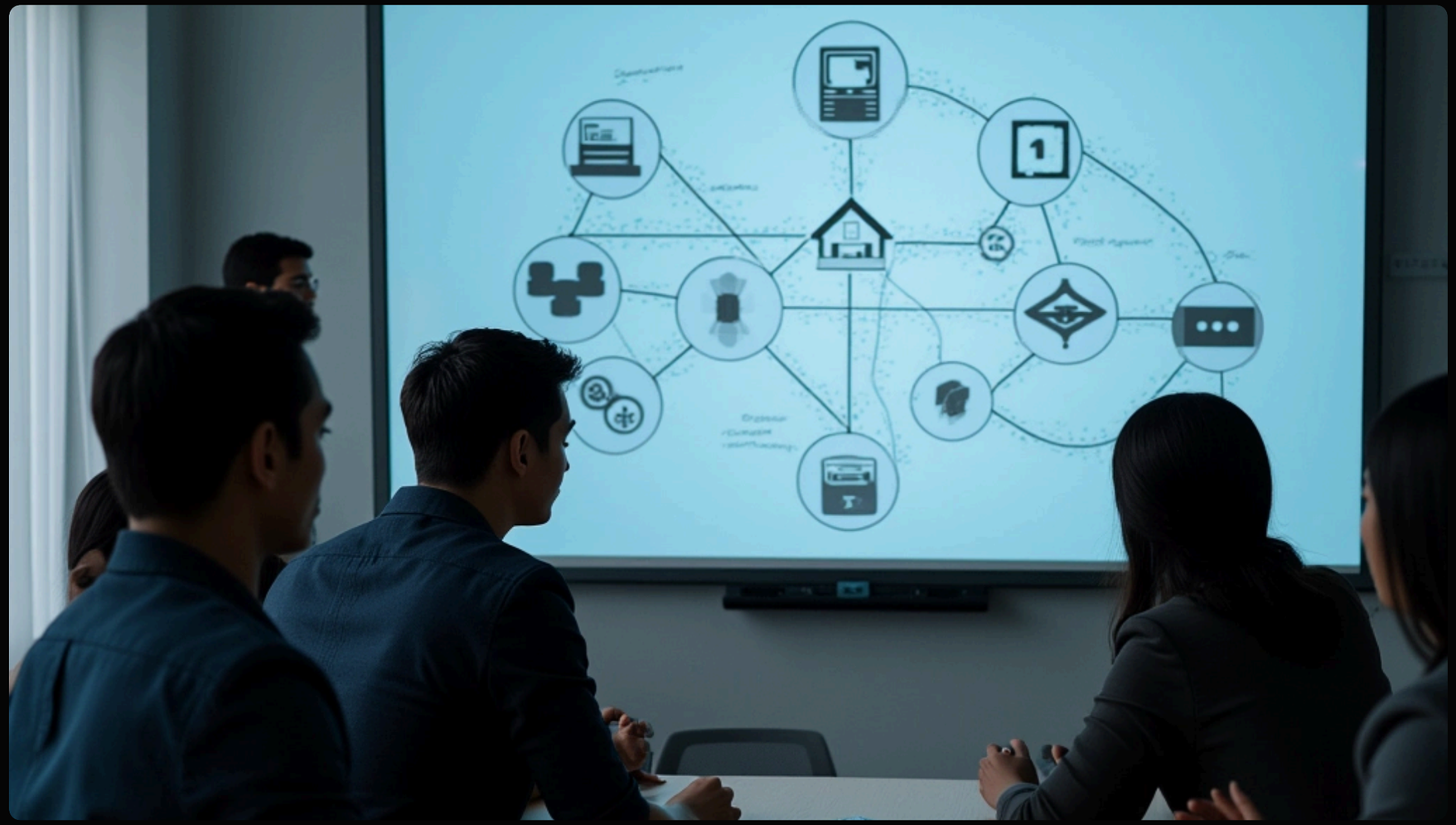
Microservices architecture: Building resilient distributed

-

A comprehensive implementation guide using Spring Boot and cloud technologies



Project team and overview

Microservices Architecture Implementation project for SDC course, instructed by Mr. Zunnurain with team members Shahzaman, Saram, and Rai.

Breaking free from monolithic limitations

Scaling

Independent module scaling

Specific modules are difficult to scale independently in monolithic systems, preventing efficient resource allocation where it's most needed.

Risk

Update complications

Tight coupling between components increases risk during updates, as changes to one part can cascade and affect the entire application unexpectedly.

Vision

Resilient architecture

Our project aims to create a resilient, distributed system architecture that eliminates single points of failure and enables independent scaling.

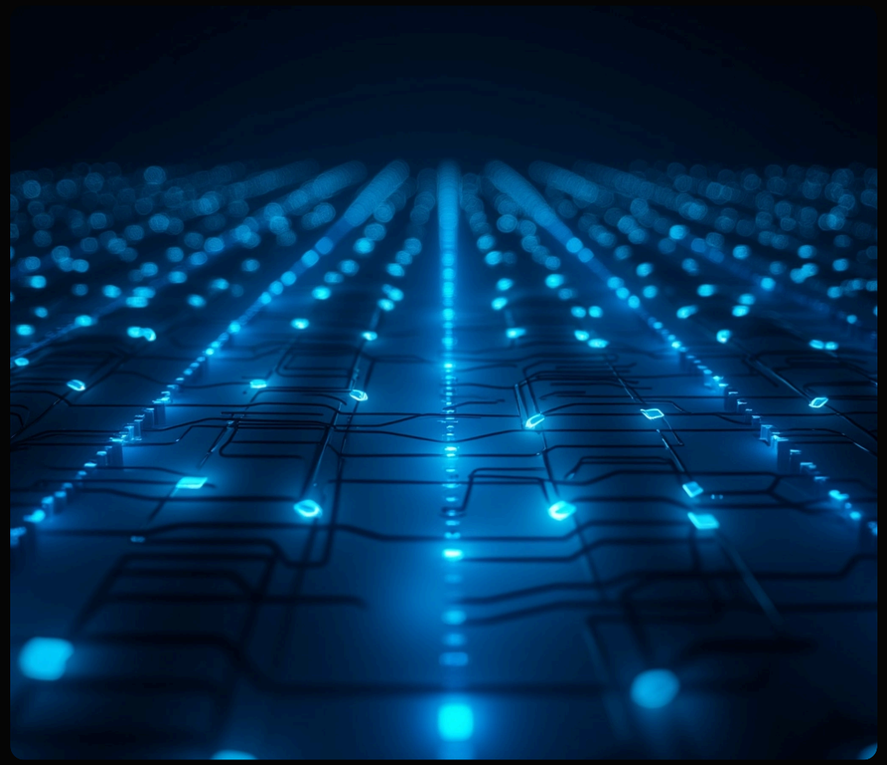
Vulnerability

Single points failure

Monolithic applications suffer from single points of failure, creating vulnerabilities that can impact the entire system when one component fails.

Strategy

Microservices solution strategy



Architecture designed for resilience and scalability. Our microservices approach ensures systems are modular and can evolve independently as business needs change.

Application decomposed into small, manageable services. Spring Boot framework enables rapid development. Spring Cloud handles distributed system patterns.

Core components

The technical architecture consists of interconnected services working together.

API Gateway

Manages routing and filtering of requests to appropriate backend services.

Backend Services

Contain core business logic that powers the application functionality.

Eureka Server

Functions as the Service Registry to track all microservices in the system.

Data storage

Manages persistent information needed by the backend services.

Support services

Includes logging, monitoring and authentication components for system maintenance.

Frontend interface

Provides the user interface for interacting with the system.

Architecture

Technology stack and implementation

Design

The architecture implements loose coupling between services, allowing components to scale independently. This approach enhances system flexibility and enables targeted performance optimization.

Backend

The system is built with Java and Spring Boot framework, creating robust REST APIs. This foundation ensures reliable performance and scalability for enterprise applications.

Frontend

A custom Web UI handles the presentation layer, providing an intuitive user interface. The frontend connects to backend services through HTTP/REST protocols with JSON data exchange.

Cloud infrastructure supports scalability requirements. Services are secured through configured Security Groups and the system is designed for optimal performance on Amazon's cloud platform.

Cloud deployment on AWS infrastructure. Services deployed on Amazon EC2 instances with Security Groups configured for microservice ports. System deployed and accessible via public IP.

Results

Results and future enhancements

Fully functional distributed system implemented with seamless user experience. Future plans include Circuit Breakers and distributed tracing.