School of Digital, Technologies and Arts
Department of Computing

COMP60010: Enterprise Cloud and Distributed Web Applications

Assignment Specification

Weighted at 50% of the module mark

Submission Guidelines

All submissions are only electronic and should be submitted via Blackboard. If you have any questions/queries, please contact the tutor.

Submission deadline: Thursday 05 Feb 2026 (23:59)

Learning outcomes being assessed by this assignment are:

3. DESIGN, IMPLEMENT AND TEST A WEB APPLICATION BASED ON THE CLOUD AND CLOUD SERVICES

4. DEVELOP, IMPLEMENT AND, TEST A DISTRIBUTED WEB APPLICATION UTILISING AN APPLICATION PROGRAM INTERFACE (API).

This is an **individual** assessment. Please check that your report and/or script/tool is entirely written by yourself and any quoted sections are referenced.

Case Study:

Scenario:

You are part of an advanced software architecture team at CloudRetail, a rapidly growing global e-commerce platform. CloudRetail handles millions of customers worldwide and offers both physical products and digital services. As CloudRetail continues to scale, they are facing challenges related to high availability, fault tolerance, real-time data synchronization, and global scaling.

Currently, CloudRetail operates on a monolithic architecture, but to support future growth and operational efficiency, the company is transitioning to a microservices-based architecture running on cloud-native infrastructure. This requires integrating cloud services with distributed applications, utilizing APIs to ensure seamless communication between microservices and external systems.

CloudRetail aims to create a highly resilient and scalable platform that ensures 24/7 availability, supports real-time data updates, and integrates with third-party APIs. The system needs to be secure, compliant with data protection regulations (e.g., GDPR), and capable of handling millions of transactions daily without compromising performance.

As part of this transformation, your team is tasked with the design and implementation of the following:

1. Cloud-Based Web Application – Design and implement a cloud-based architecture that integrates microservices and real-time data synchronization using cloud services.

2. Distributed Web Application with API – Develop and implement a distributed system that enables communication between microservices and integrates with external APIs, ensuring data consistency, fault tolerance, and real-time data flow.

Unified Requirements for Cloud-Based and Distributed Web Application:

1. Cloud Architecture Design:

Provide a comprehensive cloud-based architecture for CloudRetail's e-commerce platform, utilizing microservices and cloud-native services. This should include the use of containers (e.g., Kubernetes), serverless computing (e.g., AWS Lambda), and cloud databases (e.g., Amazon Aurora). Address global scalability, high availability, and real-time synchronization to support CloudRetail's operations across multiple geographic regions.

2. Distributed System Design:

Design a distributed system with microservices that communicate via RESTful APIs or event-driven architecture (using tools like Kafka or AWS EventBridge). Each service should be loosely coupled and independently scalable. Integrate API Gateway (e.g., AWS API Gateway) to handle routing, service discovery, and authentication across services. Provide API documentation for internal and external APIs, ensuring proper security protocols such as OAuth 2.0 and JWT.

3. Data Security, Compliance, and Consistency:

Design a comprehensive security model for handling sensitive customer data. Implement encryption (at rest and in transit), multi-factor authentication, and access control mechanisms. Ensure data compliance with regulations such as GDPR, PCI DSS, and regional data sovereignty laws. Address data consistency

challenges in a distributed system (e.g., eventual consistency, CAP theorem) and design distributed transactions or use patterns like Saga or CQRS for complex workflows.

4. Real-Time Data Synchronization:

Implement real-time data flow between microservices using event-driven patterns. Ensure that inventory updates, order status changes, and pricing adjustments are reflected across services in near real-time. Use cloud-native services like AWS Lambda and Google Cloud Functions to trigger event-driven actions based on data changes.

5. Fault Tolerance and Autonomous Recovery:

Design the system to be fault-tolerant with mechanisms like circuit breakers, retry policies, and auto-scaling. Include high-availability zones across regions. Ensure disaster recovery mechanisms are in place to handle regional outages and provide autonomous service recovery.

6. API and Microservices Security:

Implement API security measures such as rate-limiting, IP whitelisting, CORS, and API key management. Use OAuth 2.0, JWT, and RBAC for secure service-to-service and user-to-service communication. Implement API versioning and handle backward compatibility.

7. Performance and Scalability Testing:

Implement load testing and stress testing to evaluate how the platform handles traffic spikes and global scalability. Provide testing reports on latency, throughput, and error handling for both cloud-based and distributed components.

8. Monitoring and Observability:

Implement monitoring tools such as CloudWatch (AWS), Stackdriver (Google Cloud), or Azure Monitor to track the health of the system. Enable log aggregation, distributed tracing, and metrics collection to provide full observability across microservices and cloud services.

9. Testing for Distributed Web Application and Cloud Services:

Conduct unit tests for the microservices, focusing on business logic, security, and database interaction. Perform integration testing to ensure that the cloud-based web application and distributed system with APIs work together seamlessly. Test API endpoints using tools like Postman and Swagger, ensuring that they meet performance and security standards.

Deliverables:

1. Comprehensive Architecture Documentation:

Detailed architecture diagrams for the cloud-based web application and the distributed system. Include API specifications, data flow diagrams, and explanations of the cloud services used.

2. Source Code:

Fully implemented code for the cloud-based application and the distributed system. Include API documentation with endpoint details, request/response formats, and security considerations.

3. Testing Report:

Detailed results from unit testing, integration testing, and performance testing. Discuss the scalability, fault tolerance, and security testing results, and identify potential improvements.

4. Final Report:

A comprehensive report summarizing the design, implementation, testing, and results. Address challenges encountered during implementation, and discuss strategies for scaling, security, and fault tolerance.

Assessment Criteria:

Architecture Design (20%)

Quality and comprehensiveness of the architecture and design decisions, including scalability, availability, and security considerations.

Implementation (40%)

Functionality, robustness, and scalability of the cloud-based application and distributed system. Proper integration of cloud services, microservices, and API security.

Testing and Results (20%)

Comprehensive testing strategy and results, including performance, security, and fault tolerance testing.

Presentation(20%)

Clear presentation in terms of the slides, which are clear to read and follow in a logical sequence, solution is justified and clear during the presentation, Presentation is within the allocated 15 minutes and not rushed, Good clear technical answers to questions

Note:

The originality of each student's work will be rigorously evaluated during the viva voce examination. Moreover, the student's performance in the viva will substantially influence the final assessment outcome.