Project – Web based Accounting Application

Document Type: High Level Solution Design Document

1. Introduction

1.1. Purpose:

- Provide a detailed high-level design (HLD) for the Web-based Accounting Application.
- Outline the system's architecture, functionality, and design considerations.

1.2. Scope:

- Cover architectural, functional, data, interface, security, performance, deployment, and maintenance aspects.
- Focus on providing a comprehensive overview of the system design.

1.3. Intended Audience:

- Development team members.
- Project stakeholders.
- Quality assurance (QA) engineers.
- System administrators.

2. System Overview

2.1. Objectives:

- Streamline financial management processes.
- Enhance data accuracy and integrity.
- Provide insights for informed decision-making.

2.2. Key Features:

- Financial reports generation.
- Chart of accounts management.
- Transaction recording and tracking.
- Invoice processing and management.
- Ad-hoc analysis and reporting.

3. Architectural Overview

3.1. Microservices Architecture:

- Each user story represented as a separate microservice.
- Ensures modularity, scalability, and maintainability.

3.2. Components:

- Frontend components development using Node.js and AngularJS.
- Backend services implementation with Spring Boot.
- Databases: MySQL for structured data, MongoDB for semi-structured data.
- Containerization with Docker for easy deployment and scalability.
- Continuous Integration with Jenkins.
- Deployment and hosting on AWS.

3.3. Interactions:

- Communication between frontend and backend components via APIs.
- Integration points with external systems for data exchange.

4. Functional Overview

4.1. User Roles and Functionalities:

- Finance Managers: Financial reports analysis.

- Accountants: Chart of accounts management.
- Bookkeepers: Transaction recording and tracking.
- Accounts Payable Clerks: Invoice processing and payment tracking.
- Accounts Receivable Clerks: Invoice generation and payment tracking.
- Financial Analysts: Ad-hoc analysis and reporting.

4.2. High-Level-Design

4.2.1. General Ledger

I. Chart of Accounts

Component: Chart of Accounts Management Module

Functionality:
- Ability to define and customize the chart of accounts.
- Support for multiple currencies.

Sub-components:
- Chart of Accounts Editor: Interface for adding, editing, and deleting account categories and subcategories.
- Currency Management: Module to manage currencies and exchange rates.

II. Transaction Processing

Component: Transaction Processing Module

Functionality:
- Real-time posting of financial transactions.
- Automated journal entry generation based on predefined rules.

III. Periodic Closing

Component: Periodic Closing Module

Functionality:
- Year-end closing procedures.
- Ability to close accounting periods.

4.2.2. Accounts Payable

I. Vendor Management

Component: Vendor Management Module

Functionality:

- Maintain a centralized vendor database.
- Track vendor performance and history.

II. Invoice Processing

Component: Invoice Processing Module

Functionality:

- Automated invoice capture and approval workflows.
- Integration with procurement systems.

III. Payment Processing

Component: Payment Processing Module

Functionality:

- Electronic fund transfer capabilities.
- Payment scheduling and optimization.

4.2.3. Accounts Receivable

I. Customer Management

Component: Customer Management Module

Functionality:

- Maintain a centralized customer database.
- Monitor and manage credit limits.

II. Invoicing and Billing

Component: Invoicing and Billing Module

Functionality:

- Automated invoicing based on sales orders or project milestones.
- Flexible billing options.

III. Collections

Component: Collections Module

Functionality:

- Aging reports for tracking overdue payments.
- Automated dunning and collection letters.

4.2.4. Financial Reporting

I. Standard Reports

Component: Standard Reports Module

Functionality:

- Generate standard financial statements with drill-down capabilities.
- Comparative analysis of financial performance.

II. Custom Reports

Component: Custom Reports Module

Functionality:

- Ad-hoc reporting with drag-and-drop functionality.
- Save and share custom report templates.

III. Dashboards

Component: Dashboard Module

Functionality:

- Real-time financial dashboards with key performance indicators.
- Customizable dashboard views.

4.2.5. Budgeting and Forecasting

I. Budget Creation

Component: Budget Creation Module

Functionality:

- User-friendly budget creation interface.
- Version control for budget iterations.

II. Forecasting

Component: Forecasting Module

Functionality:

- Historical data analysis for accurate forecasting.
- Scenario modelling for 'what-if' analysis.

4.2.6. Compliance and Security

I. Regulatory Compliance

Component: Compliance Module

Functionality:

- Automatic updates for compliance with accounting standards.
- Audit trails for compliance reporting.

II. Access Control

Component: Access Control Module

Functionality:

- Role-based access control with granular permissions.
- Two-factor authentication.

III. Data Encryption

Component: Data Encryption Module

Functionality:
- Encryption of data at rest and during transmission.
- Regular security audits and vulnerability assessments.

4.2.7. Custom Reporting

I. Custom Reporting

Component: Custom Reporting Module

Functionality:

- Ability to generate custom reports based on user-defined criteria.
- Advanced reporting features such as data visualization and filtering.

5. Data Design

5.1. Data Entities:

- Accounts
- Transactions
- Invoices
- Customers
- Vendors
- Financial reports

5.2. Attributes and Relationships:

- Define attributes for each entity (e.g., account name, transaction amount).
- Establish relationships between entities (e.g., invoice associated with a customer).

5.3. Data Flow:

- Illustrate how data moves through the system.
- Ensure efficient data management and retrieval.

6. Interface Design

6.1. User Interfaces:

- Intuitive navigation and user experience.
- Wireframes to visualize UI design.

6.2. Application Programming Interfaces (APIs):

- Define APIs for communication between frontend and backend components.
- Ensure consistency and security in API design.

6.3. Integration Points:

- Integration with external systems for data exchange.
- Maintain data integrity and security during integration.

7. Security Design

7.1. Authentication and Authorization:

- Implement secure authentication mechanisms (e.g., OAuth, JWT).
- Role-based access control (RBAC) to restrict user access to authorized functionalities.

7.2. Encryption:

Encrypt sensitive data at rest and in transit (e.g., SSL/TLS for communication, encryption for data storage).

7.3. Data Protection:

- Implement data protection measures to ensure confidentiality and integrity.
- Regular security audits and vulnerability assessments.

8. Performance Design

8.1. Scalability:

- Horizontal scaling to handle increasing loads.
- Load balancing strategies for distributing traffic across multiple instances.

8.2. Responsiveness:

- Optimizing frontend and backend code for faster response times.
- Asynchronous processing for time-consuming tasks.

8.3. Resource Utilization:

- Efficient use of hardware resources to minimize costs.
- Monitoring and optimization of resource usage.

9. Deployment Design

9.1. Hardware and Software Requirements:

- Specify hardware specifications for servers.
- Software dependencies and versions required for deployment.

9.2. Deployment Environments:

- Development, testing, staging, and production environments.
- Configuration management for consistency across environments.

9.3. Deployment Processes:

- Automated deployment pipelines with Jenkins.
- Rollback strategies for handling deployment failures.

10. Maintenance and Support

10.1. System Monitoring:

- Implement monitoring tools for tracking system health and performance.
- Alerts and notifications for critical events.

10.2. Error Handling:

- Comprehensive error handling mechanisms.
- Logging and error reporting for debugging purposes.

10.3. Troubleshooting:

- Procedures for diagnosing and resolving issues.
- Knowledge base for common troubleshooting scenarios.

10.4. Ongoing Support:

- User support channels (e.g., help desk, ticketing system).
- Regular updates and patches for bug fixes and security enhancements.