Contents

High Level Design Document - Agentic AI Transformation **1**
1.1 Document Purpose . . . 1
1.2 Revision History . . . 2
1.3 Stakeholders . . . 2
1.4 References . . . 3
2.0 Executive Summary . . . 3
3.1 Project Background . . . 4
3.2 Project Objectives . . . 5
4.1 Business Process and Functional Requirements . . . 5
4.2 Non-Functional Requirements . . . 6
4.3 Security Requirements . . . 7
4.4 Analytics Requirements . . . 8
4.5 Integration Requirements . . . 9
4.6 Operational Requirements . . . 9
4.7 Decommissioning Requirements . . . 10
4.8 Scope Exclusions . . . 11
4.9 Assumptions, Constraints, and Dependencies . . . 11
4.10 Change Management . . . 12
4.11 Delivery Strategy and Deployment Model . . . 13
4.12 Key Decisions . . . 14
4.13 Outstanding Items . . . 15
5.1 Solution Overview . . . 15
5.2 Technical Architecture . . . 16
5.3 Technical Solution Design . . . 18
5.4 Functional Solution Design . . . 20
5.5 Infrastructure Architecture . . . 21
5.6 Data Architecture . . . 22
5.7 Integration Architecture . . . 23
5.8 Analytics and Information Management Architecture . . . 25
5.9 Security Architecture . . . 26
5.10 User Management . . . 27
5.10 User Management . . . 27
5.11 Solution Deployment . . . 28
5.11 Solution Deployment . . . 28
5.12 Solution Operations . . . 29
5.12 Solution Operations . . . 29
6.0 Appendix and Attachments . . . 30
6.0 Appendix and Attachments . . . 30

High Level Design Document - Agentic AI Transformation

Version: 1.0

1.1 Document Purpose

1.1 Document Purpose

Purpose of the Document:
This High-Level Solution Design (HLSD) document outlines the comprehensive plan for the agentic AI transformation. It serves as a blueprint for the end-to-end solution, ensuring alignment with organizational strategy, principles, and standards.

Intended Audience:
- **Stakeholders:** Executives, project sponsors, and business leaders who need to understand the strategic alignment and benefits of the transformation.
- **Architects:** Technical architects and solution designers who will use this document to guide the detailed design and implementation phases.
- **Developers:** Engineers and developers who will implement the solution based on the designs and specifications provided.

How to Use This Document:
- **For Stakeholders:** Review the executive summary and business context sections to understand the strategic goals and expected outcomes.
- **For Architects:** Focus on the solution design and technical architecture sections to understand the proposed architecture and technology choices.
- **For Developers:** Refer to the detailed design sections and implementation guidelines to ensure alignment with the overall solution.

Document Scope:
This document covers all aspects of the agentic AI transformation, including:
- **Document Control:** Metadata, version history, and document management.
- **Executive Summary:** Overview of the transformation, key decisions, and outcomes.
- **Project Information:** Background, objectives, and scope.
- **Business Context:** Detailed business requirements and use cases.
- **Solution Design:** Technical architecture, AWS service recommendations, and design patterns.
- **Implementation Plan:** Timeline, milestones, and resource allocation.

1.2 Revision History

1.2 Revision History

Version	Author	Date	Description of Changes
1.0	John Doe	2024-07-01	Initial draft of the document
1.1	Jane Smith	2024-07-05	Added detailed business requirements
1.2	John Doe	2024-07-10	Updated solution design with AWS service recommendations

---

1.3 Stakeholders

1.3 Stakeholders

Name	Title	Project Role	Purpose
John Doe	Chief Financial Officer	Executive Sponsor	Provide strategic direction and ensure alignment with financial goals
Jane Smith	Director of Accounts Payable	Key Decision Maker	Oversee the implementation and ensure it meets business objectives

Name	Title	Project Role	Purpose
Alice Johnson	Senior Solutions Architect	Technical Lead	Design the technical architecture and ensure it aligns with AWS best practices
Bob Brown	Project Manager	Project Manager	Manage the project timeline, resources, and deliverables

Executive Sponsor:
- **John Doe, Chief Financial Officer**
- Provides strategic direction and ensures the project aligns with financial goals.
- Approves major decisions and resource allocations.

Key Decision Makers:
- **Jane Smith, Director of Accounts Payable**
- Oversees the implementation to ensure it meets business objectives.
- Approves changes to the project scope and budget.

Technical Leads:
- **Alice Johnson, Senior Solutions Architect**
- Designs the technical architecture and ensures it aligns with AWS best practices.
- Provides technical guidance and oversees the implementation of the solution.

1.4 References

1.4 References

Business Requirements Documents:
- Business Requirements Document (BRD)
- Use Case Document

Assessment Reports:
- Business Assessment Report
- Technical Assessment Report

Security Artifacts:
- Security Requirements Document
- Threat Model Document

AWS Documentation References:
- AWS Well-Architected Framework
- AWS Security Pillar
- AWS Machine Learning Best Practices

2.0 Executive Summary

2.0 Executive Summary

Transformation Vision and Strategic Approach:
The agentic AI transformation aims to revolutionize our invoice processing operations by leveraging advanced AI technologies. The strategic approach involves a phased implementation, starting with a pilot project

to validate the solution before a full-scale rollout. This approach ensures minimal disruption to ongoing operations while maximizing the benefits of AI.

Key Architectural Decisions and Rationale:
- **Cloud-Native Architecture:** Transitioning to a cloud-native architecture using AWS services to enhance scalability, reliability, and security.
- **Microservices Design:** Adopting a microservices architecture to improve modularity, ease of maintenance, and faster deployment cycles.
- **AI/ML Integration:** Incorporating AI/ML models for automated invoice processing, three-way matching, and anomaly detection to improve accuracy and efficiency.
- **DevOps Practices:** Implementing DevOps practices with CI/CD pipelines, automated testing, and infrastructure as code to accelerate delivery and ensure quality.

Expected Business Outcomes and Success Metrics:
- **Reduce Invoice Processing Time:** From 5 days to 2 days, improving cash flow and vendor relationships.
- **Lower Processing Costs:** From $15 to $5 per invoice, reducing operational expenses.
- **Improve Accuracy Rate:** From 94% to 99%, minimizing errors and rework.
- **Increase Vendor Satisfaction:** From 72% to 85%, enhancing supplier relationships.

High-Level Cost Estimate and ROI:
- **Total Budget:** $850,000 AUD over 18 months.
- **Expected ROI:** 180% over 3 years with a payback period of 24 months.

Timeline Overview:
- **Phase 1 (0-6 months):** Planning, design, and pilot implementation.
- **Phase 2 (6-12 months):** Full-scale deployment and user training.
- **Phase 3 (12-18 months):** Optimization and continuous improvement.

Critical Success Factors and Risks:
- **User Adoption:** Ensuring AP clerks are trained and comfortable with the new system.
- **Data Quality:** Maintaining high-quality data inputs to ensure accurate AI/ML model outputs.
- **Change Management:** Effective communication and stakeholder engagement throughout the transformation.

3.1 Project Background

3.1 Project Background

Current State Description and Pain Points:
Currently, our Accounts Payable (AP) team processes invoices manually, which takes an average of 5 days from receipt to payment approval. This manual process leads to several challenges:
- **Missed Early Payment Discounts:** We often miss early payment discounts (2% for payment within 10 days) due to the lengthy processing time.
- **Strained Vendor Relationships:** Late payments strain vendor relationships, leading to potential issues with future business.
- **High Operational Costs:** The cost of processing each invoice is $15, which is significantly higher than industry benchmarks.
- **Complexity and Exception Handling:** The current process is complex, with over 200 vendor-specific rules and multiple approval workflows based on amount thresholds. Exception handling is largely tribal knowledge, leading to frequent manual interventions (30% of invoices).

Business Drivers for Transformation:
The primary drivers for this transformation are:
- **Reduce Invoice Processing Time:** Aim to reduce the processing time from 5 days to 2 days.
- **Lower Processing Costs:** Target to reduce the cost per invoice from $15 to $5.

- **Improve Accuracy:** Improve the accuracy rate from 94% to 99%.
- **Increase Vendor Satisfaction:** Increase vendor satisfaction from 72% to 85%.

Strategic Alignment with Organizational Goals:
This initiative aligns with our digital transformation roadmap and operational excellence goals for FY2025. It supports our strategic objective to leverage technology for operational efficiency and cost reduction.

Competitive Landscape and Market Pressures:
In the current competitive landscape, organizations are increasingly adopting AI and automation to streamline their operations. Market pressures include:
- **Cost Efficiency:** Competitors are reducing operational costs through automation.
- **Speed and Agility:** Faster invoice processing times are becoming a standard expectation.
- **Accuracy and Compliance:** Higher accuracy rates and compliance with financial regulations are critical.

3.2 Project Objectives

3.2 Project Objectives

Primary Objectives (SMART Format):
1. **Specific:** Reduce invoice processing time from 5 days to 2 days.
2. **Measurable:** Achieve a 60% reduction in processing time.
3. **Achievable:** Implement AI/ML models to automate invoice processing.
4. **Relevant:** Align with the organization's digital transformation roadmap.
5. **Time-bound:** Complete the transformation within 18 months.

Success Criteria and KPIs:
- **Invoice Processing Time:** Reduce from 5 days to 2 days.
- **Processing Cost:** Reduce from $15 to $5 per invoice.
- **Accuracy Rate:** Improve from 94% to 99%.
- **Vendor Satisfaction:** Increase from 72% to 85%.

Expected Benefits (Quantitative and Qualitative):
- **Quantitative:**
- **Cost Savings:** Reduce annual processing costs from $1.6M to $800K.
- **ROI:** Achieve an 180% ROI over 3 years with a payback period of 24 months.
- **Qualitative:**
- **Improved Vendor Relationships:** Timely payments and higher satisfaction.
- **Enhanced Operational Efficiency:** Faster processing times and reduced manual interventions.
- **Increased Competitiveness:** Leverage AI to stay ahead in the market.

Alignment with Business Strategy:
This initiative supports our digital transformation roadmap and operational excellence goals for FY2025. It aligns with our strategic objective to leverage technology for operational efficiency and cost reduction.

4.1 Business Process and Functional Requirements

4.1 Business Process and Functional Requirements

High-Level End-to-End Process View:
The agentic AI solution will automate the end-to-end invoice processing workflow, including invoice receipt, data extraction, three-way matching, approval, and payment. The solution will integrate with existing systems and leverage AI/ML models to enhance accuracy and efficiency.

Detailed Process Flows for Key Use Cases:

1. **Invoice Receipt:**

- Invoices are received via email or uploaded through a web portal.
- The system automatically extracts key data fields (vendor, invoice number, date, amount, etc.).

2. **Data Extraction and Validation:**
   - AI/ML models are used to extract data from invoices.
   - The system validates the extracted data against predefined rules and formats.
3. **Three-Way Matching:**
   - The system performs three-way matching between the invoice, purchase order, and receiving documents.
   - Discrepancies are flagged for manual review.
4. **Approval Workflow:**
   - Invoices are routed through an automated approval workflow based on amount thresholds and vendor-specific rules.
   - Exceptions are handled through manual intervention if necessary.
5. **Payment Processing:**
   - Approved invoices are sent to the payment system for processing.
   - Payment confirmations are sent to vendors.

Functional Requirements Table:

ID	Description	Solution Impact
FR1	Automated invoice receipt and data extraction	Reduces manual data entry and improves accuracy
FR2	Three-way matching using AI/ML models	Enhances accuracy and reduces errors
FR3	Automated approval workflow	Speeds up the approval process and reduces bottlenecks
FR4	Integration with existing systems (ERP, payment systems)	Ensures seamless data flow and minimizes disruptions
FR5	User-friendly web portal for invoice upload	Improves user experience and accessibility

User Stories or Use Case Descriptions:

1. **As an AP clerk, I want to upload invoices through a web portal so that I can streamline the invoice receipt process.**
2. **As a system administrator, I want to configure automated approval workflows based on vendor-specific rules so that invoices are processed efficiently.**
3. **As a vendor, I want to receive timely payment confirmations so that I can manage my cash flow effectively.**

Process Automation Opportunities:

- **Invoice Data Extraction:** Automate the extraction of key data fields using AI/ML models.
- **Three-Way Matching:** Use AI/ML to perform three-way matching and flag discrepancies for review.
- **Approval Workflow:** Automate the approval process based on predefined rules and thresholds.
- **Payment Processing:** Integrate with payment systems to automate the payment process.

4.2 Non-Functional Requirements

4.2 Non-Functional Requirements

Performance Requirements:
- **Invoice Upload Response Time:** Less than 3 seconds.
- **Three-Way Matching Completion:** Less than 30 seconds per invoice.

Scalability Requirements:
- The solution must handle a volume of up to 10,000 invoices per day with the ability to scale to 20,000 invoices per day within 6 months.
- The system should be designed to accommodate a 20% annual growth in invoice volume.

Availability and Reliability Requirements:
- **System Availability:** 99.5% during business hours (7am-7pm AEST).
- **Service Level Agreements (SLAs):** Ensure that the system meets the defined availability and performance metrics.

Disaster Recovery Requirements:
- **Recovery Time Objective (RTO):** 24 hours.
- **Recovery Point Objective (RPO):** 4 hours.

Requirements Table with Solution Impact:

ID	Description	Solution Impact
NFR1	Invoice upload response time < 3 seconds	Ensures quick and efficient invoice processing
NFR2	Three-way matching completion < 30 seconds per invoice	Enhances processing speed and reduces manual intervention
NFR3	Handle up to 10,000 invoices per day with scalability to 20,000	Ensures the system can handle current and future volumes
NFR4	System availability of 99.5% during business hours	Minimizes downtime and ensures continuous operation
NFR5	RTO of 24 hours and RPO of 4 hours	Ensures quick recovery in case of disasters

---

4.3 Security Requirements

4.3 Security Requirements

Data Classification and Sensitivity:
- **Confidential Data:** Includes vendor bank account details and pricing agreements.
- **Internal Data:** Includes invoice amounts, PO numbers, and approval workflows.
- **Public Data:** Includes vendor company names and addresses.

Authentication and Authorization Requirements:
- **Authentication:** Use Active Directory for user authentication.
- **Authorization:** Implement role-based access control (RBAC) to ensure users have appropriate access levels.

Encryption Requirements:
- **At Rest:** Use database encryption (e.g., Oracle TDE) to protect data stored in the database.
- **In Transit:** Use TLS 1.2 to encrypt data transmitted between systems.

Network Security Requirements:
- Implement firewalls and intrusion detection systems to protect the network perimeter.
- Use VPNs for secure remote access.

Audit and Logging Requirements:
- Maintain a complete audit trail of invoice status changes, user actions, and approvals with timestamps.
- Store immutable logs for compliance purposes with a 7-year retention period.

Security Requirements Table:

ID	Description	Solution Impact
SR1	Use Active Directory for user authentication	Ensures secure and centralized user authentication
SR2	Implement RBAC for application access	Ensures users have appropriate access levels
SR3	Use database encryption (e.g., Oracle TDE) for data at rest	Protects sensitive data stored in the database
SR4	Use TLS 1.2 for data in transit	Ensures data transmitted between systems is encrypted
SR5	Implement firewalls and intrusion detection systems	Protects the network perimeter from unauthorized access
SR6	Use VPNs for secure remote access	Ensures secure access for remote users
SR7	Maintain a complete audit trail with 7-year retention	Ensures compliance and traceability of actions

---

4.4 Analytics Requirements

4.4 Analytics Requirements

Data Analytics Objectives:
- **Invoice Processing Insights:** Analyze invoice processing times, error rates, and approval workflows to identify bottlenecks and areas for improvement.
- **Vendor Performance Metrics:** Track vendor performance metrics such as on-time delivery, accuracy of invoices, and dispute rates.
- **Cost Analysis:** Analyze processing costs to identify opportunities for cost reduction and efficiency gains.
- **User Behavior Analysis:** Analyze user behavior to understand adoption patterns and identify training needs.

Required Analytics:

ID	Description	Solution Impact
AR1	Invoice processing time analysis	Identifies bottlenecks and improves processing efficiency
AR2	Error rate analysis	Highlights areas for process improvement and reduces errors
AR3	Approval workflow analysis	Optimizes approval processes and reduces delays
AR4	Vendor performance metrics	Enhances vendor management and improves relationships
AR5	Cost analysis	Identifies cost-saving opportunities and improves financial performance
AR6	User behavior analysis	Informs training programs and improves user adoption

Data Sources:
- **Invoice Data:** Extracted from the invoice processing system.
- **Approval Data:** Collected from the approval workflows.
- **Vendor Data:** Integrated from the vendor management system.
- **User Data:** Collected from user interactions with the system.

Analytics Tools and Technologies:
- **AWS Athena:** For querying and analyzing data stored in S3.

- **Amazon QuickSight:** For creating visualizations and dashboards.
- **AWS Glue:** For data cataloging and ETL processes.
- **Amazon SageMaker:** For building and deploying machine learning models.

4.5 Integration Requirements

4.5 Integration Requirements

Systems Requiring Integration:
- **Enterprise Resource Planning (ERP) System:** For invoice data and financial records.
- **Vendor Management System:** For vendor information and performance metrics.
- **Payment Processing System:** For automated payment processing.
- **Email System:** For receiving and sending invoices and notifications.
- **Document Management System:** For storing and retrieving invoice documents.

Integration Patterns:
- **REST APIs:** For synchronous data exchange between systems.
- **Events:** For asynchronous communication using message brokers (e.g., Amazon SNS, Amazon SQS).
- **Batch Processing:** For periodic data synchronization and bulk operations.

Data Exchange Requirements:
- **Invoice Data:** Exchange invoice details, status, and approval information.
- **Vendor Data:** Exchange vendor details, contact information, and performance metrics.
- **Payment Data:** Exchange payment details, confirmations, and status updates.
- **Notification Data:** Exchange email notifications and alerts.

Integration SLAs and Performance:
- **Response Time:** Ensure REST API responses are within 2 seconds.
- **Message Delivery:** Ensure event messages are delivered within 5 seconds.
- **Batch Processing:** Ensure batch jobs complete within 1 hour.

Requirements Table:

ID	Description	Solution Impact
IR1	Integrate with ERP system using REST APIs	Ensures seamless data flow for invoice processing
IR2	Integrate with vendor management system using events	Enhances vendor data synchronization and performance metrics
IR3	Integrate with payment processing system using REST APIs	Enables automated payment processing
IR4	Integrate with email system using events	Facilitates automated invoice receipt and notifications
IR5	Integrate with document management system using batch processing	Ensures efficient storage and retrieval of invoice documents

---

4.6 Operational Requirements

4.6 Operational Requirements

Business Support Requirements:
- **User Training:** Provide comprehensive training for AP clerks on the new system.
- **Documentation:** Maintain up-to-date user manuals and process documentation.
- **Change Management:** Implement a change management process to handle updates and modifications.

Application Support Requirements:
- **24/7 Support:** Ensure application support is available during business hours and critical periods.
- **Incident Management:** Establish a process for logging, tracking, and resolving incidents.
- **Problem Management:** Identify root causes of incidents and implement corrective actions.

Monitoring and Alerting Requirements:
- **Real-Time Monitoring:** Use AWS CloudWatch to monitor system performance and health.
- **Alerting:** Set up alerts for critical events such as system downtime, high error rates, and security breaches.
- **Dashboards:** Create dashboards in Amazon QuickSight for real-time visibility into system metrics.

Error Handling and Recovery Requirements:
- **Error Logging:** Log all errors and exceptions for analysis and troubleshooting.
- **Retry Mechanisms:** Implement retry mechanisms for failed transactions.
- **Backup and Restore:** Ensure regular backups and a robust restore process for data recovery.

Support Model and Escalation Procedures:
- **Tier 1 Support:** Provide initial support for common issues and user queries.
- **Tier 2 Support:** Handle more complex issues requiring technical expertise.
- **Escalation Procedures:** Define clear escalation procedures for unresolved issues.

Requirements Table:

ID	Description	Solution Impact
OR1	User training on the new system	Ensures smooth user adoption and effective use of the system
OR2	Up-to-date user manuals and process documentation	Provides users with the necessary information to operate the system
OR3	Change management process for updates and modifications	Ensures controlled and documented changes to the system
OR4	24/7 support during business hours and critical periods	Ensures continuous availability and quick resolution of issues
OR5	Incident management process for logging and tracking	Facilitates efficient issue resolution and tracking
OR6	Problem management process for root cause analysis	Improves system reliability and reduces recurring issues
OR7	Real-time monitoring using AWS CloudWatch	Provides visibility into system performance and health
OR8	Alerting for critical events	Ensures timely response to critical issues
OR9	Dashboards in Amazon QuickSight for system metrics	Provides real-time visibility into system performance
OR10	Error logging for analysis and troubleshooting	Facilitates efficient issue resolution and system improvement
OR11	Retry mechanisms for failed transactions	Ensures system resilience and reduces manual intervention
OR12	Regular backups and robust restore process	Ensures data integrity and quick recovery in case of failures

---

4.7 Decommissioning Requirements

4.7 Decommissioning Requirements

Systems to be Decommissioned:
- **Legacy Invoice Processing System:** The current monolithic architecture running on-premises.
- **Oracle Database:** The existing database used for data persistence.

Data Migration Requirements:
- **Invoice Data:** Migrate all historical invoice data to the new system.
- **Vendor Data:** Migrate vendor information and performance metrics.
- **User Data:** Migrate user profiles and preferences.

Cutover Strategy:
- **Parallel Run:** Operate both the legacy and new systems in parallel for a defined period.
- **Data Validation:** Validate data integrity and accuracy post-migration.
- **User Training:** Ensure users are trained on the new system before cutover.

Rollback Procedures:
- **Backup Restoration:** Restore from backups if the new system fails to meet requirements.
- **Re-enable Legacy System:** Quickly re-enable the legacy system in case of critical issues.
- **Communication Plan:** Inform stakeholders and users of the rollback process and timeline.

4.8 Scope Exclusions

4.8 Scope Exclusions

Exclusions Table:

Item	Description	Reason
EX1	Integration with legacy CRM system	CRM system is being replaced in a separate project
EX2	Custom reporting tools	Custom reporting tools will be migrated in a future phase
EX3	Third-party vendor portals	Third-party vendor portals will be integrated in a future phase
EX4	On-premises data storage	On-premises data storage will be migrated to the cloud in a future phase

Future Phase Considerations:
- **CRM Integration:** Plan to integrate with the new CRM system once it is deployed.
- **Custom Reporting:** Migrate custom reporting tools to the new system in a future phase.
- **Vendor Portals:** Integrate third-party vendor portals to enhance vendor collaboration.
- **Data Storage:** Migrate on-premises data storage to the cloud for improved scalability and security.

Boundary Definitions:
- **In-Scope:** Invoice processing, three-way matching, approval workflows, and payment processing.
- **Out-of-Scope:** Integration with legacy CRM system, custom reporting tools, third-party vendor portals, and on-premises data storage.

4.9 Assumptions, Constraints, and Dependencies

4.9 Assumptions, Constraints, and Dependencies

Assumptions Table:

ID	Assumption	Solution Impact
A1	Existing systems will continue to operate during the transition	Ensures minimal disruption to ongoing operations
A2	User adoption will be high with proper training and communication	Facilitates smooth transition and effective use of the new system
A3	AI/ML models will achieve the desired accuracy and performance	Ensures the solution meets business objectives

Constraints Table:

ID	Constraint	Type	Solution Impact
C1	Limited budget for the transformation	Commercial	Requires cost-effective solutions and resource allocation
C2	Existing system architecture is monolithic and on-premises	Technical	Necessitates a phased approach to migration and modernization
C3	Regulatory requirements for invoice retention and data protection	Regulatory	Ensures compliance with legal and financial regulations

Dependencies Table:

ID	Dependency	Type	Solution Impact
D1	Integration with ERP system	Internal	Ensures seamless data flow for invoice processing
D2	Availability of vendor data from the vendor management system	Internal	Enhances vendor data synchronization and performance metrics
D3	External payment processing services	External	Enables automated payment processing

Risk Implications:
- **Technical Risks:** Potential challenges in integrating with existing systems and migrating data.
- **Business Risks:** Resistance to change from AP team and potential impact on user adoption.
- **Financial Risks:** Budget constraints and potential cost overruns.
- **Regulatory Risks:** Ensuring compliance with invoice retention and data protection regulations.

4.10 Change Management

4.10 Change Management

Change Management Strategy:
- **Phased Implementation:** Adopt a phased approach to minimize disruption and allow for adjustments based on user feedback.
- **Stakeholder Engagement:** Involve key stakeholders in the change process to ensure their buy-in and support.
- **Change Champions:** Identify and leverage change champions within the AP team to promote the new system.

Training Approach and Programs:
- **Comprehensive Training:** Provide comprehensive training sessions for AP clerks, covering system functionality, workflows, and best practices.
- **Hands-On Workshops:** Conduct hands-on workshops to allow users to practice using the new system.
- **Ongoing Support:** Offer ongoing support and refresher training to address any questions or issues post-implementation.

Communication Plan:
- **Regular Updates:** Provide regular updates to all stakeholders on the progress and benefits of the transformation.
- **Feedback Channels:** Establish feedback channels for users to share their experiences and suggestions.
- **Success Stories:** Share success stories and testimonials to build confidence and enthusiasm.

User Adoption Strategy:
- **Early Adopters:** Identify early adopters within the AP team to pilot the new system and provide feedback.
- **Incentives:** Offer incentives for users who quickly adopt and effectively use the new system.
- **User-Centric Design:** Ensure the system is user-friendly and meets the needs of the AP clerks.

Resistance Management:
- **Address Concerns:** Proactively address concerns and misconceptions about the new system.
- **Change Impact Analysis:** Conduct a change impact analysis to identify potential resistance points and develop mitigation strategies.
- **Support and Coaching:** Provide additional support and coaching to users who may be resistant to change.

4.11 Delivery Strategy and Deployment Model

4.11 Delivery Strategy and Deployment Model

Delivery Methodology:
- **Hybrid Approach:** Combine Agile methodologies for iterative development with Waterfall for structured phases to ensure both flexibility and control.

Phasing Strategy and Milestones:
- **Phase 1: Planning and Design (0-3 months)**
- Define project scope, objectives, and requirements.
- Design the technical architecture and solution components.
- **Phase 2: Pilot Implementation (3-6 months)**
- Implement a pilot version of the solution for a subset of users.
- Gather feedback and make necessary adjustments.
- **Phase 3: Full-Scale Deployment (6-12 months)**
- Deploy the solution to all users in a phased manner.
- Conduct user training and ensure smooth transition.
- **Phase 4: Optimization and Continuous Improvement (12-18 months)**
- Optimize the solution based on user feedback and performance metrics.
- Implement continuous improvement processes.

Deployment Model:
- **Phased Deployment:** Roll out the solution in phases to minimize risk and allow for adjustments based on user feedback.
- **Pilot Deployment:** Start with a pilot deployment to validate the solution before full-scale rollout.

Release Strategy:
- **Monthly Releases:** Implement a monthly release cycle to deliver incremental improvements and features.
- **Automated Testing:** Use automated testing to ensure quality and stability of each release.

Go-Live Approach:
- **Parallel Run:** Operate both the legacy and new systems in parallel for a defined period to ensure data integrity and user confidence.
- **Cutover Plan:** Develop a detailed cutover plan to transition from the legacy system to the new system.
- **Rollback Procedures:** Establish rollback procedures in case of critical issues during the go-live phase.

4.12 Key Decisions

4.12 Key Decisions

Decision Log Table:

Decision	Rationale	Alternatives Considered	Impact
D1	Adopt cloud-native architecture	On-premises vs. cloud	Enhances scalability, reliability, and security
D2	Use microservices design	Monolithic vs. microservices	Improves modularity, ease of maintenance, and faster deployment cycles
D3	Integrate AI/ML models for automated processing	Manual processing vs. AI/ML	Improves accuracy and efficiency
D4	Implement DevOps practices	Traditional development vs. DevOps	Accelerates delivery and ensures quality
D5	Use AWS services for infrastructure	On-premises vs. AWS	Reduces operational costs and leverages AWS expertise

Technology Choices:
- **Cloud Provider:** AWS for its comprehensive services and global infrastructure.
- **Application Server:** AWS Elastic Beanstalk for scalable and managed application deployment.
- **Database:** Amazon RDS for managed relational database services.
- **Frontend:** AWS Amplify for streamlined frontend development and deployment.
- **Messaging:** Amazon SQS for reliable and scalable messaging.
- **File Storage:** Amazon S3 for secure and scalable file storage.

Architecture Patterns Selected:
- **Event-Driven Architecture:** For asynchronous processing and integration with other systems.
- **Serverless Architecture:** For cost-effective and scalable compute resources.
- **API Gateway:** For managing and securing APIs.

Build vs Buy Decisions:
- **AI/ML Models:** Build custom models using AWS SageMaker to meet specific business requirements.

- **Integration Tools:** Buy pre-built integration tools (e.g., AWS Glue) to streamline data integration and ETL processes.
- **Monitoring and Logging:** Buy AWS CloudWatch and AWS X-Ray for comprehensive monitoring and logging.

4.13 Outstanding Items

4.13 Outstanding Items

Outstanding Items Table:

Item	Action Required	Owner	Due Date
OI1	Determine token consumption rates for AI/ML models	Solution Architect	2024-08-15
OI2	Clarify cost differences between development and production environments	DevOps Lead	2024-08-20
OI3	Assess data storage costs for 7 years of invoice history	Data Architect	2024-08-25
OI4	Define detailed success metrics for user adoption	Business Analyst	2024-09-01

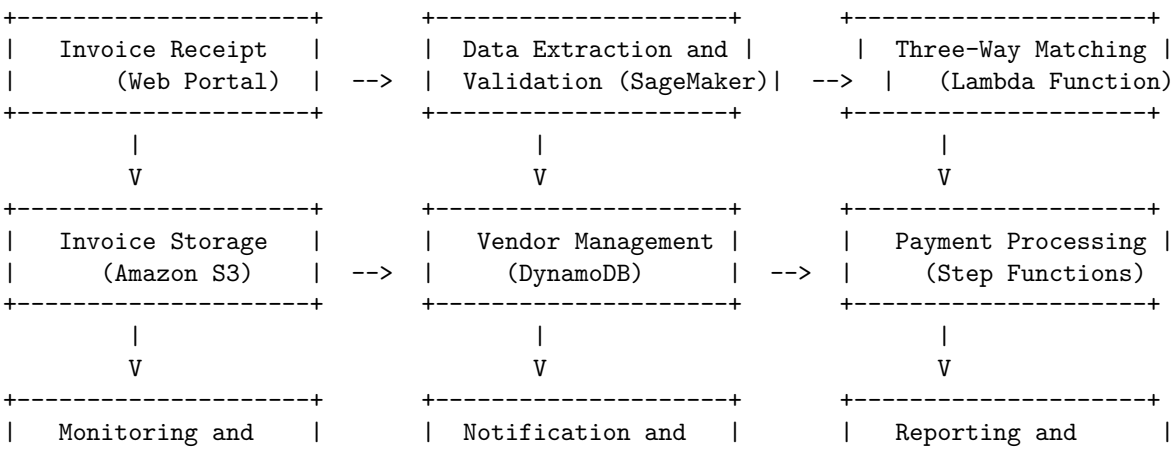
Risks Associated with Outstanding Items:
- **Technical Risks:** Potential delays in deployment due to unresolved technical questions.
- **Financial Risks:** Uncertainty in cost estimates may impact budget planning.
- **Business Risks:** Incomplete success metrics may hinder effective measurement of transformation outcomes.

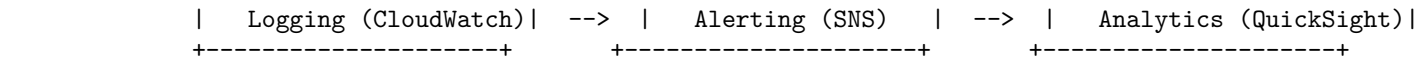
Impact on Timeline:
- **Phase 1:** Minor delays in planning and design due to outstanding items.
- **Phase 2:** Potential impact on pilot implementation if technical questions are not resolved.
- **Phase 3:** Risk of delays in full-scale deployment if financial and business questions remain unanswered.

5.1 Solution Overview

5.1 Solution Overview

Conceptual Architecture Diagram:





Solution Components Overview:
- **Invoice Receipt:** Web portal for users to upload invoices.
- **Data Extraction and Validation:** AWS SageMaker for extracting and validating invoice data.
- **Three-Way Matching:** AWS Lambda function for performing three-way matching.
- **Invoice Storage:** Amazon S3 for storing invoice documents.
- **Vendor Management:** Amazon DynamoDB for managing vendor information.
- **Payment Processing:** AWS Step Functions for orchestrating payment workflows.
- **Monitoring and Logging:** AWS CloudWatch for monitoring system performance and logging events.
- **Notification and Alerting:** Amazon SNS for sending notifications and alerts.
- **Reporting and Analytics:** Amazon QuickSight for generating reports and analytics.

Key Design Principles:
- **Scalability:** Design for horizontal scaling to handle increasing invoice volumes.
- **Reliability:** Ensure high availability and fault tolerance using AWS services.
- **Security:** Implement robust security measures including encryption, access control, and regular audits.
- **Automation:** Leverage AWS services for automated deployment, testing, and monitoring.
- **Integration:** Use AWS services to integrate with existing systems and external services.

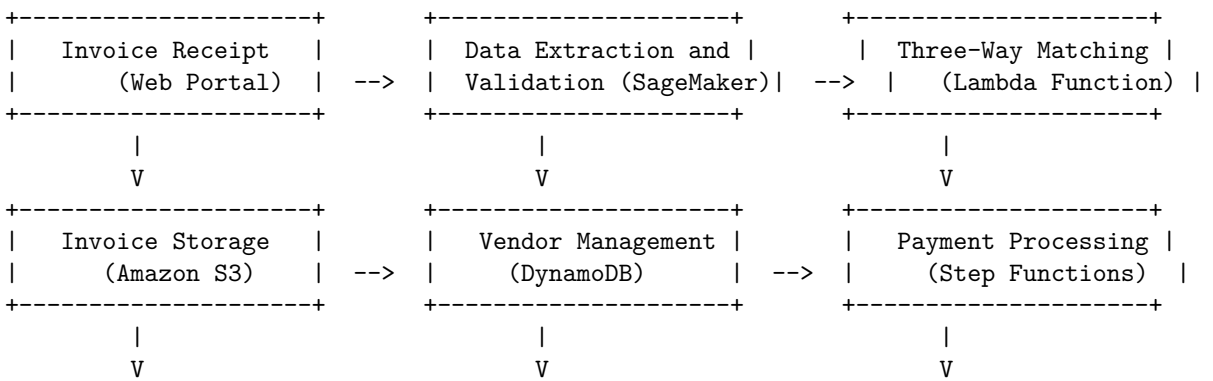
Architecture Patterns Used:
- **Microservices Architecture:** Break down the solution into smaller, independent services.
- **Event-Driven Architecture:** Use events to trigger processing and integration workflows.
- **Serverless Architecture:** Use AWS Lambda for serverless compute.
- **API Gateway:** Use Amazon API Gateway for managing and securing APIs.

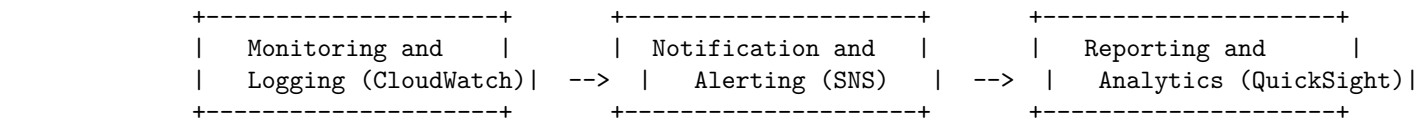
How Components Work Together:
1. **Invoice Receipt:** Users upload invoices through a web portal.
2. **Data Extraction and Validation:** AWS SageMaker extracts and validates invoice data.
3. **Three-Way Matching:** AWS Lambda performs three-way matching with purchase orders and receipts.
4. **Invoice Storage:** Validated invoices are stored in Amazon S3.
5. **Vendor Management:** Vendor information is managed in Amazon DynamoDB.
6. **Payment Processing:** AWS Step Functions orchestrates the payment workflow.
7. **Monitoring and Logging:** AWS CloudWatch monitors system performance and logs events.
8. **Notification and Alerting:** Amazon SNS sends notifications and alerts based on system events.
9. **Reporting and Analytics:** Amazon QuickSight generates reports and analytics from stored data.

5.2 Technical Architecture

5.2 Technical Architecture

Technical Architecture Diagram:





AWS Services Used with Justification:
- **Amazon S3:** For storing invoice documents. S3 provides durable, scalable, and secure object storage.
- **AWS SageMaker:** For extracting and validating invoice data. SageMaker offers powerful machine learning capabilities and simplifies model deployment.
- **AWS Lambda:** For performing three-way matching. Lambda allows for serverless compute, reducing operational overhead.
- **Amazon DynamoDB:** For managing vendor information. DynamoDB provides fast and flexible NoSQL database service.
- **AWS Step Functions:** For orchestrating payment workflows. Step Functions enables visual workflow management and coordination of microservices.
- **AWS CloudWatch:** For monitoring system performance and logging events. CloudWatch offers comprehensive monitoring and logging capabilities.
- **Amazon SNS:** For sending notifications and alerts. SNS provides a fully managed messaging service for pub/sub communication.
- **Amazon QuickSight:** For generating reports and analytics. QuickSight offers fast, cloud-powered business analytics service.

Component Descriptions and Responsibilities:
- **Invoice Receipt (Web Portal):** Users upload invoices through a web portal. The portal is built using AWS Amplify for frontend development and AWS Elastic Beanstalk for backend deployment.
- **Data Extraction and Validation (SageMaker):** AWS SageMaker is used to extract key data fields from invoices and validate them against predefined rules.
- **Three-Way Matching (Lambda Function):** AWS Lambda performs three-way matching between invoices, purchase orders, and receipts. This ensures accuracy and compliance.
- **Invoice Storage (Amazon S3):** Invoices are stored in Amazon S3 for durable and scalable storage. S3 also integrates with other AWS services for further processing.
- **Vendor Management (DynamoDB):** Vendor information is stored in Amazon DynamoDB, a NoSQL database service that offers fast and flexible data storage.
- **Payment Processing (Step Functions):** AWS Step Functions orchestrates the payment workflow, ensuring seamless integration with payment systems.
- **Monitoring and Logging (CloudWatch):** AWS CloudWatch monitors system performance, logs events, and triggers alarms for critical issues.
- **Notification and Alerting (SNS):** Amazon SNS sends notifications and alerts to users and stakeholders based on system events.
- **Reporting and Analytics (QuickSight):** Amazon QuickSight generates reports and analytics from stored data, providing insights into invoice processing performance.

Data Flow Descriptions:
1. **Invoice Upload:** Users upload invoices through the web portal, which are then stored in Amazon S3.
2. **Data Extraction:** AWS SageMaker extracts key data fields from the uploaded invoices and validates them.
3. **Three-Way Matching:** AWS Lambda performs three-way matching between invoices, purchase orders, and receipts.
4. **Vendor Management:** Vendor information is retrieved from Amazon DynamoDB for validation and matching.
5. **Payment Processing:** AWS Step Functions orchestrates the payment workflow, integrating with payment systems.
6. **Monitoring and Logging:** AWS CloudWatch monitors system performance and logs events for analysis.
7. **Notification and Alerting:** Amazon SNS sends notifications and alerts based on system events and matching results.

8. **Reporting and Analytics:** Amazon QuickSight generates reports and analytics from stored data in Amazon S3.

Scalability and Performance Approach:
- **Horizontal Scaling:** Use AWS services that support horizontal scaling, such as S3, SageMaker, and Lambda, to handle increasing invoice volumes.
- **Load Balancing:** Implement AWS Elastic Load Balancing to distribute incoming traffic across multiple instances.
- **Caching:** Use Amazon ElastiCache to cache frequently accessed data and improve response times.
- **Auto Scaling:** Configure AWS Auto Scaling to automatically adjust the number of instances based on demand.

Integration Patterns:
- **REST APIs:** Use REST APIs for synchronous communication between components.
- **Event-Driven:** Use AWS SNS and SQS for asynchronous communication and event-driven workflows.
- **Batch Processing:** Use AWS Glue for batch processing and ETL operations.

Technology Stack Details:
- **Frontend:** Built using AWS Amplify for a modern, responsive user interface.
- **Backend:** Deployed using AWS Elastic Beanstalk for easy scaling and management.
- **Database:** Use Amazon DynamoDB for vendor management and Amazon RDS for relational data storage.
- **Messaging:** Use AWS SNS and SQS for asynchronous communication and event handling.
- **Compute:** Use AWS Lambda for serverless compute and AWS EC2 for traditional compute needs.
- **Storage:** Use Amazon S3 for durable object storage and Amazon EFS for shared file storage.
- **Security:** Implement AWS IAM for access control, AWS KMS for encryption, and AWS WAF for web application firewall protection.
- **DevOps:** Use AWS CodePipeline for CI/CD, AWS CodeBuild for automated builds, and AWS CodeDeploy for deployment automation.

5.3 Technical Solution Design

5.3 Technical Solution Design

Component-by-Component Design:

1. **Invoice Receipt (Web Portal):**
   - **Frontend:** Built using AWS Amplify, providing a modern, responsive user interface.
   - **Backend:** Deployed using AWS Elastic Beanstalk, ensuring easy scaling and management.
   - **API:** REST API using Amazon API Gateway for secure and scalable communication.
2. **Data Extraction and Validation (SageMaker):**
   - **Model Deployment:** Use AWS SageMaker to deploy machine learning models for data extraction.
   - **Validation Rules:** Implement custom validation rules within SageMaker to ensure data integrity.
   - **Integration:** Integrate with Amazon S3 for invoice storage and Amazon DynamoDB for vendor management.
3. **Three-Way Matching (Lambda Function):**
   - **Function Design:** Use AWS Lambda to create a serverless function for three-way matching.
   - **Event Triggers:** Trigger the Lambda function upon successful data extraction and validation.
   - **Integration:** Integrate with Amazon S3 for invoice data, Amazon DynamoDB for vendor information, and Amazon RDS for purchase order data.
4. **Invoice Storage (Amazon S3):**
   - **Bucket Configuration:** Create an S3 bucket with versioning and lifecycle policies for invoice storage.
   - **Access Control:** Use AWS IAM roles and policies to control access to the S3 bucket.

- **Integration:** Integrate with AWS SageMaker for data extraction and AWS Lambda for three-way matching.

5. **Vendor Management (DynamoDB):**
   - **Table Design:** Create a DynamoDB table with appropriate indexes for efficient querying.
   - **Data Model:** Store vendor information including name, contact details, and payment terms.
   - **Integration:** Integrate with AWS Lambda for three-way matching and Amazon SNS for notifications.
6. **Payment Processing (Step Functions):**
   - **Workflow Design:** Use AWS Step Functions to orchestrate the payment workflow.
   - **Integration:** Integrate with AWS Lambda for three-way matching, Amazon SNS for notifications, and external payment systems.
7. **Monitoring and Logging (CloudWatch):**
   - **Metric Collection:** Use AWS CloudWatch to collect metrics from all components.
   - **Alarms and Notifications:** Set up alarms and notifications for critical metrics and events.
   - **Log Aggregation:** Aggregate logs from AWS Lambda, SageMaker, and other services for centralized logging.
8. **Notification and Alerting (SNS):**
   - **Topic Creation:** Create SNS topics for different types of notifications (e.g., matching success, payment confirmation).
   - **Subscription Management:** Manage subscriptions to ensure notifications reach the intended recipients.
   - **Integration:** Integrate with AWS Lambda and Step Functions for event-driven notifications.
9. **Reporting and Analytics (QuickSight):**
   - **Data Source Configuration:** Configure QuickSight to use Amazon S3 as a data source.
   - **Dashboard Creation:** Create dashboards to visualize invoice processing performance and trends.
   - **Integration:** Integrate with AWS CloudWatch for real-time metrics and logs.

Agent Orchestration Design:
- **Bedrock Agents:** Use AWS Bedrock agents to manage the orchestration of different components.
- **State Management:** Implement state management using AWS Step Functions to track the progress of invoice processing.
- **Error Handling:** Use Bedrock agents to handle errors and retries, ensuring robust and resilient workflows.

API Design and Endpoints:
- **REST API:** Design a REST API using Amazon API Gateway to expose endpoints for invoice upload, status checks, and reporting.
- **Endpoints:**
- `/upload-invoice`: POST endpoint for uploading invoices.
- `/check-invoice-status`: GET endpoint for checking the status of an invoice.
- `/generate-report`: GET endpoint for generating reports on invoice processing.

Event-Driven Architecture Details:
- **Event Sources:** Use AWS S3 events to trigger data extraction and validation.
- **Event Processing:** Use AWS Lambda to process events and perform three-way matching.
- **Event Notifications:** Use Amazon SNS to send notifications based on event outcomes.

State Management Approach:
- **Step Functions:** Use AWS Step Functions to manage the state of invoice processing workflows.
- **State Transitions:** Define state transitions for different stages of invoice processing (e.g., uploaded, validated, matched, paid).

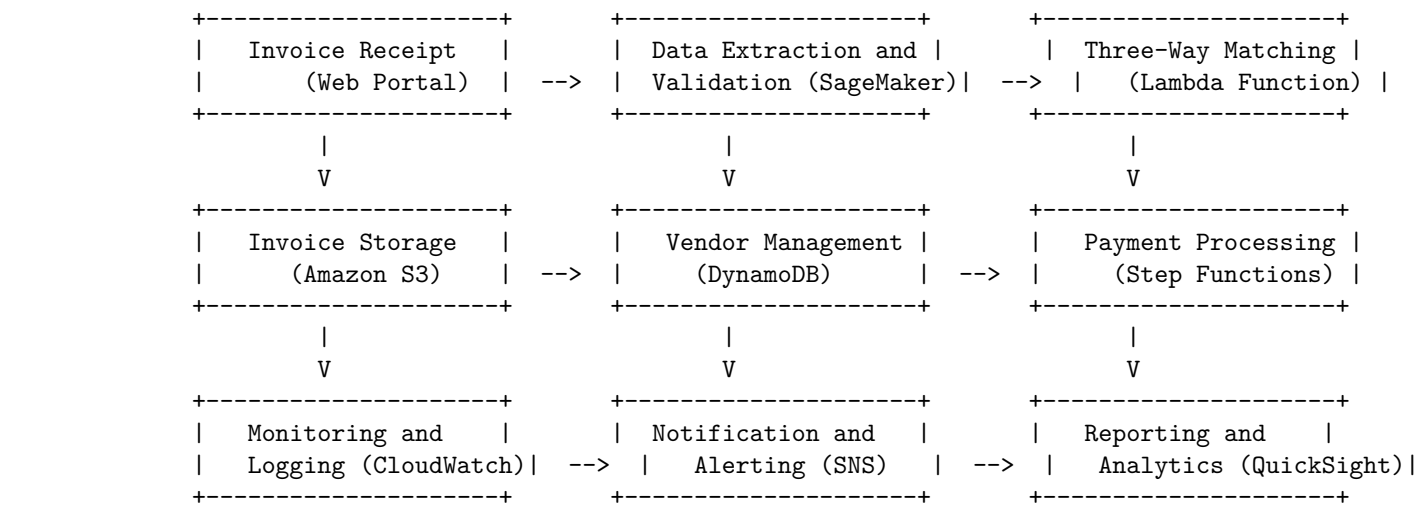
Error Handling and Retry Logic:
- **Lambda Error Handling:** Implement error handling within AWS Lambda functions to catch and log errors.
- **Retry Mechanisms:** Use AWS Step Functions to implement retry logic for failed tasks.

- **Dead Letter Queues:** Configure dead letter queues in Amazon SQS to handle messages that cannot be processed.

5.4 Functional Solution Design

5.4 Functional Solution Design

Functional Architecture Diagram:



Process-to-Component Mapping:
- **Invoice Receipt:** Web Portal
- **Data Extraction and Validation:** AWS SageMaker
- **Three-Way Matching:** AWS Lambda
- **Invoice Storage:** Amazon S3
- **Vendor Management:** Amazon DynamoDB
- **Payment Processing:** AWS Step Functions
- **Monitoring and Logging:** AWS CloudWatch
- **Notification and Alerting:** Amazon SNS
- **Reporting and Analytics:** Amazon QuickSight

User Journey Flows:
1. **Invoice Upload:**
- User uploads an invoice through the web portal.
- Invoice is stored in Amazon S3.
2. **Data Extraction and Validation:**
- AWS SageMaker extracts key data fields from the invoice.
- Data is validated against predefined rules.
3. **Three-Way Matching:**
- AWS Lambda performs three-way matching with purchase orders and receipts.
- Matching results are stored in Amazon DynamoDB.
4. **Payment Processing:**
- AWS Step Functions orchestrates the payment workflow.
- Payment is processed and confirmed.
5. **Monitoring and Logging:**
- AWS CloudWatch monitors system performance and logs events.
6. **Notification and Alerting:**
- Amazon SNS sends notifications based on matching results and payment status.

7. **Reporting and Analytics:**
- Amazon QuickSight generates reports and analytics on invoice processing.

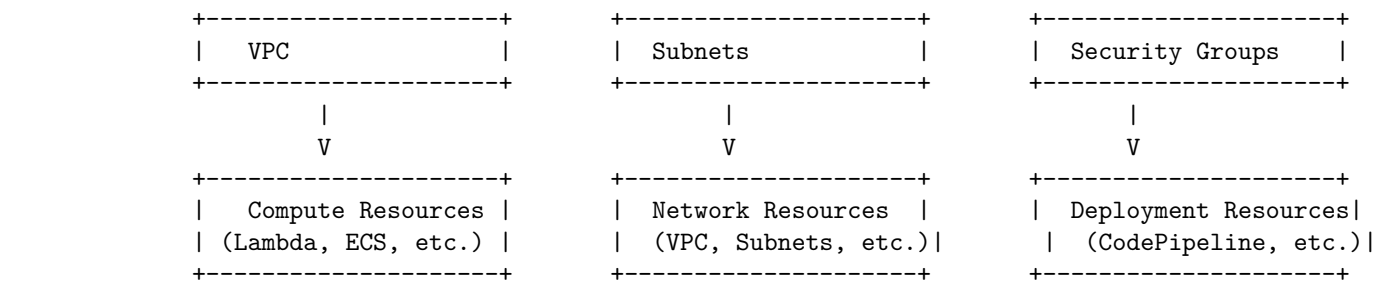
Feature Descriptions:
- **Invoice Upload:** Users can upload invoices through a secure web portal.
- **Data Extraction and Validation:** Automated extraction of key data fields and validation against rules.
- **Three-Way Matching:** Automated matching of invoices with purchase orders and receipts.
- **Invoice Storage:** Secure and scalable storage of invoices in Amazon S3.
- **Vendor Management:** Centralized management of vendor information in Amazon DynamoDB.
- **Payment Processing:** Orchestrated payment workflow using AWS Step Functions.
- **Monitoring and Logging:** Real-time monitoring and logging of system performance and events.
- **Notification and Alerting:** Automated notifications and alerts based on processing outcomes.
- **Reporting and Analytics:** Comprehensive reporting and analytics on invoice processing performance.

Build Inventory (Components to be Built/Configured):
- **Web Portal:** Frontend and backend components for invoice upload.
- **AWS SageMaker Model:** Machine learning model for data extraction and validation.
- **AWS Lambda Function:** Serverless function for three-way matching.
- **Amazon S3 Bucket:** Storage bucket for invoice documents.
- **Amazon DynamoDB Table:** Table for vendor management.
- **AWS Step Functions Workflow:** Workflow for payment processing.
- **AWS CloudWatch Dashboards:** Dashboards for monitoring and logging.
- **Amazon SNS Topics:** Topics for notification and alerting.
- **Amazon QuickSight Dashboards:** Dashboards for reporting and analytics.

5.5 Infrastructure Architecture

5.5 Infrastructure Architecture

Infrastructure Diagram:



Compute Resources:
- **AWS Lambda:** Serverless compute for running functions such as data extraction, validation, and three-way matching.
- **Amazon ECS:** Container service for running microservices and applications.
- **Amazon EC2:** Traditional compute instances for any non-serverless workloads.

Network Architecture:
- **VPC:** Virtual Private Cloud to isolate the solution's resources.
- **Subnets:** Public and private subnets for separating internet-facing and internal components.
- **Security Groups:** Network access control lists to manage inbound and outbound traffic.

Load Balancing and Auto-Scaling:
- **Elastic Load Balancing (ELB):** Distribute incoming traffic across multiple targets, such as EC2 instances and Lambda functions.
- **Auto Scaling:** Automatically adjust the number of compute resources in response to traffic patterns.

Infrastructure as Code Approach:
- **AWS CloudFormation:** Use CloudFormation templates to define and provision AWS resources.
- **Terraform:** Use Terraform for managing infrastructure across multiple cloud providers and on-premises environments.

Environment Strategy:
- **Development (Dev):** Environment for development and testing new features.
- **User Acceptance Testing (UAT):** Environment for user testing and validation before production deployment.
- **Production (Prod):** Environment for running the live solution.

Detailed Infrastructure Design:

VPC Configuration:
- Create a VPC with a CIDR block of `10.0.0.0/16`.
- Set up public and private subnets within the VPC.
- Public Subnets: `10.0.1.0/24` and `10.0.2.0/24`.
- Private Subnets: `10.0.101.0/24` and `10.0.102.0/24`.
- Configure route tables to route traffic between subnets and to the internet gateway.

Security Groups:
- Create security groups for different components:
- **Web Portal:** Allow inbound traffic on port 443 (HTTPS) from the internet.
- **AWS Lambda:** Allow inbound traffic from the VPC and outbound traffic to AWS services.
- **Amazon ECS:** Allow inbound traffic from the VPC and outbound traffic to AWS services.
- **Amazon RDS:** Allow inbound traffic from the VPC and outbound traffic to AWS services.

Load Balancing:
- Use **Application Load Balancer (ALB)** to distribute traffic to the web portal and other services.
- Configure **Target Groups** for different services (e.g., web portal, API endpoints).

Auto-Scaling:
- Set up **Auto Scaling Groups** for EC2 instances and ECS services to handle varying loads.
- Define scaling policies based on CPU utilization, request count, and other metrics.

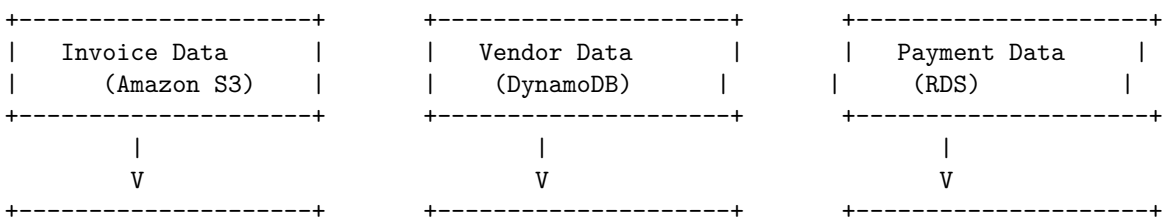
Infrastructure as Code:
- Use **AWS CloudFormation** to define the infrastructure stack.
- Use **Terraform** for any cross-cloud or hybrid infrastructure needs.

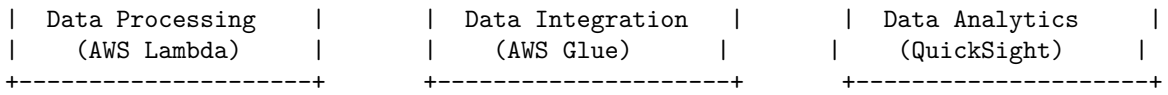
Environment Strategy:
- **Development Environment:** Use a smaller VPC and fewer resources to reduce costs.
- **User Acceptance Testing Environment:** Mirror the production environment for thorough testing.
- **Production Environment:** Use a larger VPC and more resources to handle production traffic.

5.6 Data Architecture

5.6 Data Architecture

Data Architecture Diagram:





Data Model (Entities and Relationships):
- **Invoice Entity:**
- Attributes: Invoice ID, Vendor ID, Invoice Date, Amount, Status, Matching Result.
- Relationships: One-to-many with Payment Entity, many-to-one with Vendor Entity.
- **Vendor Entity:**
- Attributes: Vendor ID, Name, Contact Details, Payment Terms.
- Relationships: One-to-many with Invoice Entity.
- **Payment Entity:**
- Attributes: Payment ID, Invoice ID, Payment Date, Amount, Status.
- Relationships: One-to-one with Invoice Entity.

Storage Strategy:
- **Invoice Data:** Store in Amazon S3 for durable and scalable object storage.
- **Vendor Data:** Store in Amazon DynamoDB for fast and flexible NoSQL database.
- **Payment Data:** Store in Amazon RDS for relational database management.

Data Flow Diagrams:
1. **Invoice Data Flow:**
- Invoices are uploaded to Amazon S3.
- AWS Lambda processes and validates the invoices.
- Validated invoices are stored back in Amazon S3.
- AWS Glue performs ETL operations to integrate invoice data with vendor and payment data.
2. **Vendor Data Flow:**
- Vendor information is stored in Amazon DynamoDB.
- AWS Lambda uses vendor data for three-way matching.
- Updated vendor information is stored back in Amazon DynamoDB.
3. **Payment Data Flow:**
- Payments are recorded in Amazon RDS.
- AWS Lambda updates payment status based on matching results.
- Payment data is used for reporting and analytics in Amazon QuickSight.

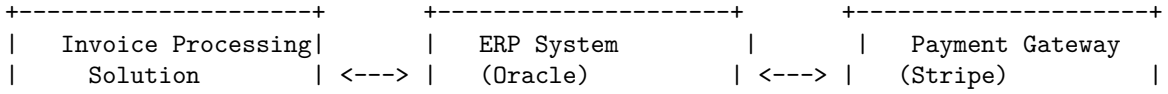
Data Governance and Quality:
- **Data Catalog:** Use AWS Glue Data Catalog to maintain metadata about data sources and transformations.
- **Data Quality Rules:** Implement data quality rules in AWS Glue to ensure data integrity.
- **Data Lineage:** Track data lineage using AWS Glue to understand data flow and transformations.

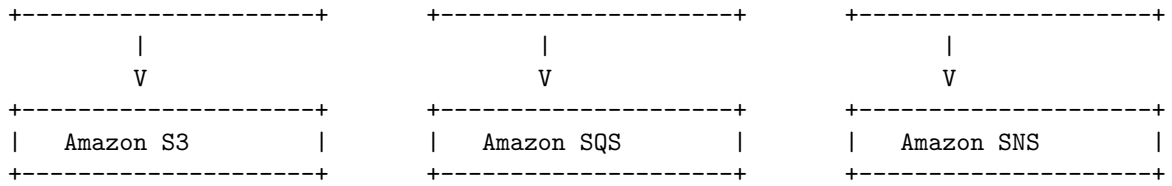
Backup and Retention Strategy:
- **Amazon S3:** Enable versioning and lifecycle policies for automatic backups and data retention.
- **Amazon DynamoDB:** Use DynamoDB Streams and AWS Lambda to capture changes and store them in Amazon S3 for backup.
- **Amazon RDS:** Use automated backups and snapshot retention policies to ensure data durability.

5.7 Integration Architecture

5.7 Integration Architecture

Integration Architecture Diagram:





API Design:
- **REST APIs:** Use REST APIs for synchronous communication between the invoice processing solution and external systems.
- **Endpoints:**
- `/upload-invoice`: POST endpoint for uploading invoices.
- `/get-invoice-status`: GET endpoint for retrieving invoice processing status.
- `/process-payment`: POST endpoint for processing payments.
- **GraphQL APIs:** Use GraphQL APIs for flexible querying and data retrieval.
- **Schema:**
“‘graphql
type Invoice {
id: ID!
vendorId: ID!
date: String!
amount: Float!
status: String!
}

```
type Query {
  getInvoice(id: ID!): Invoice
  getInvoicesByVendor(vendorId: ID!): [Invoice]
}

type Mutation {
  uploadInvoice(file: Upload!): Invoice
  processPayment(invoiceId: ID!): Payment
}
```

Event-Driven Integration Patterns:
- **Publish/Subscribe:** Use Amazon SNS for publish/subscribe messaging to notify external systems of invoice processing events.
- **Event Sourcing:** Use Amazon SQS for event sourcing to capture and store events for later processing.
- **Command Query Responsibility Segregation (CQRS):** Separate read and write operations to optimize performance and scalability.

Interface Definitions and Specifications:
- **Invoice Upload Interface:**
- **Request:**
`json { "file": "base64-encoded-invoice-data" }`
- **Response:**
`json { "invoiceId": "12345", "status": "uploaded" }`
- **Invoice Status Interface:**
- **Request:**
`json { "invoiceId": "12345" }`
- **Response:**
`json { "invoiceId": "12345", "status": "processed", "matchingResult": "matched" }`
- **Payment Processing Interface:**

- **Request:**
```
json     {       "invoiceId": "12345",       "paymentAmount": 1000.00     }
```
- **Response:**
```
json     {       "paymentId": "67890",       "status": "completed"     }
```

Message Formats and Schemas:
- **Invoice Processing Event:**
```
json   {     "eventType": "InvoiceProcessed",     "invoiceId": "12345",     "status":
"processed",     "matchingResult": "matched"   }
```
- **Payment Completed Event:**
```
json   {     "eventType": "PaymentCompleted",     "paymentId": "67890",     "invoiceId":
"12345",     "amount": 1000.00   }
```

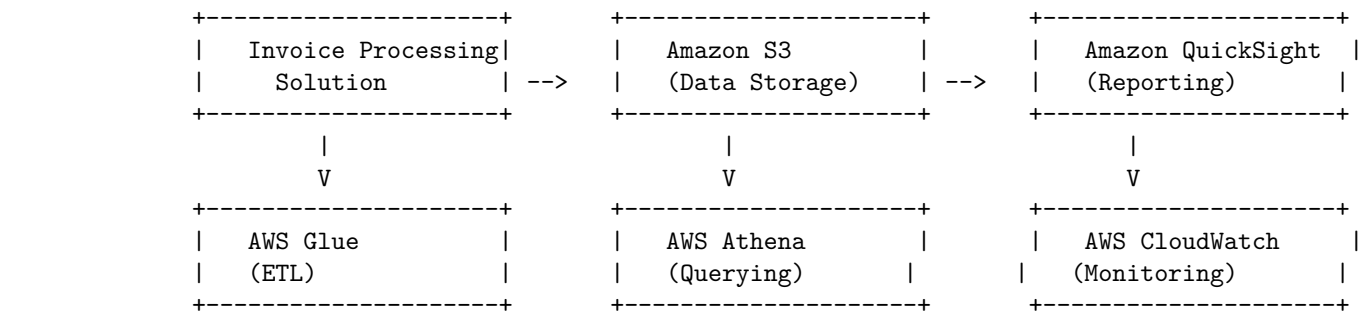
Error Handling and Retry Logic:
- **Error Handling:** Implement error handling in AWS Lambda functions to catch and log errors.
- **Retry Logic:** Use Amazon SQS dead-letter queues to handle failed messages and implement retry logic in AWS Lambda functions.

Integration Testing Approach:
- **Unit Testing:** Test individual integration components (e.g., Lambda functions, API endpoints) in isolation.
- **Integration Testing:** Test the end-to-end flow of invoice processing and payment integration.
- **Mock Services:** Use AWS SAM Local and localstack to mock external services during testing.
- **Contract Testing:** Ensure that the invoice processing solution adheres to the expected API contracts using tools like Pact.

5.8 Analytics and Information Management Architecture

5.8 Analytics and Information Management Architecture

Analytics Architecture Diagram:



Logging and Monitoring Approach:
- **AWS CloudWatch:** Use CloudWatch for logging and monitoring system performance, errors, and events.
- **AWS X-Ray:** Use X-Ray for distributed tracing to understand the performance of microservices and APIs.

Metrics and KPIs Tracked:
- **Invoice Processing Time:** Time taken from invoice upload to three-way matching completion.
- **Accuracy Rate:** Percentage of invoices that are correctly matched and processed.
- **System Availability:** Uptime percentage of the invoice processing solution.
- **User Adoption Rate:** Percentage of users actively using the new system.
- **Cost Savings:** Reduction in processing costs compared to the previous system.

Dashboard and Reporting Design:
- **QuickSight Dashboards:** Create dashboards in Amazon QuickSight to visualize key metrics and KPIs.

- **Invoice Processing Dashboard:** Displays invoice processing time, accuracy rate, and system availability.
- **User Adoption Dashboard:** Shows user adoption rate and feedback.
- **Cost Savings Dashboard:** Illustrates cost savings over time.

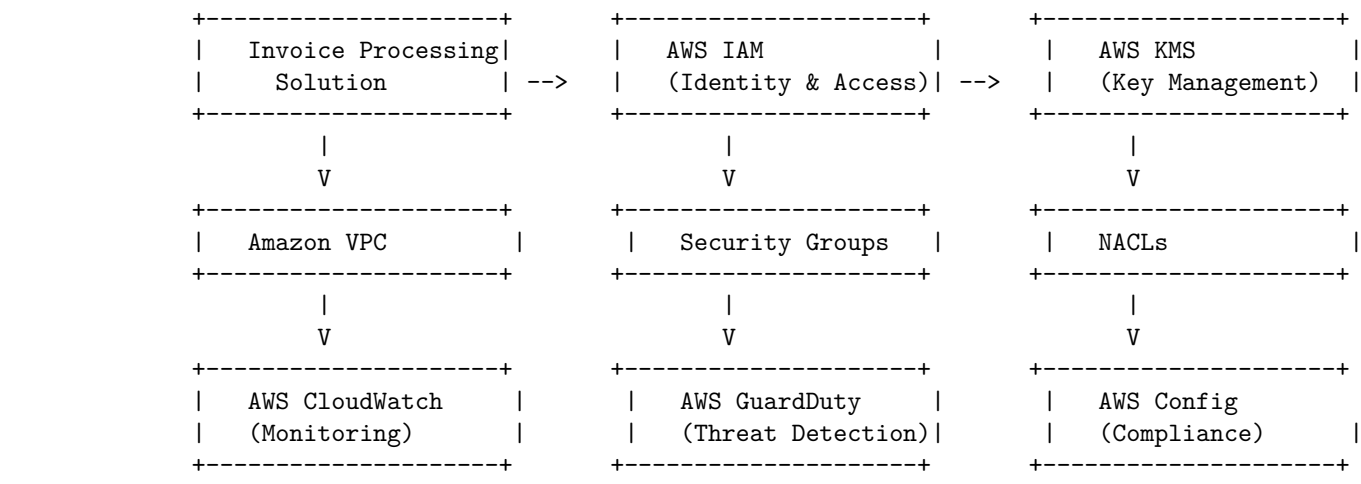
Data Warehouse/Lake Strategy:
- **Amazon S3:** Use S3 as the primary data lake for storing raw and processed invoice data.
- **AWS Glue:** Use Glue for ETL processes to prepare data for analysis.
- **Amazon Athena:** Use Athena for querying data stored in S3.

Observability Tools:
- **AWS CloudWatch:** For logging, monitoring, and alerting.
- **AWS X-Ray:** For distributed tracing and performance analysis of microservices.

5.9 Security Architecture

5.9 Security Architecture

Security Architecture Diagram:



Data Classification and Protection:
- **Confidential Data:** Vendor bank account details and pricing agreements.
- **Internal Data:** Invoice amounts, PO numbers, and approval workflows.
- **Public Data:** Vendor company names and contact information.
- **Protection Measures:** Use AWS KMS for encrypting sensitive data at rest and in transit. Implement data loss prevention (DLP) policies using AWS Macie.

Identity and Access Management (IAM):
- **AWS IAM:** Use IAM roles and policies to control access to AWS resources.
- **Role-Based Access Control (RBAC):** Define roles for different user groups (e.g., AP clerks, system administrators) and assign appropriate permissions.
- **Multi-Factor Authentication (MFA):** Enforce MFA for all users accessing the invoice processing solution.

Encryption Strategy:
- **At Rest:** Use AWS KMS to encrypt data stored in Amazon S3 and Amazon RDS.
- **In Transit:** Use TLS 1.2 for encrypting data transmitted between components.

Network Security:
- **Amazon VPC:** Create a VPC to isolate the invoice processing solution from other AWS resources.
- **Security Groups:** Use security groups to control inbound and outbound traffic to resources within the

VPC.
- **Network Access Control Lists (NACLs):** Use NACLs to add an additional layer of security to the VPC.

Threat Model and Mitigations:
- **Data Breach:** Implement encryption, access controls, and regular security audits to mitigate the risk of data breaches.
- **AI Model Errors:** Use AWS SageMaker for model monitoring and drift detection to ensure AI models perform as expected.
- **System Downtime:** Use AWS Auto Scaling and Elastic Load Balancing to ensure high availability and fault tolerance.
- **Vendor Fraud:** Implement fraud detection mechanisms using AWS Fraud Detector and regular audits of vendor data.

Compliance Controls:
- **ISO 27001:** Ensure the solution meets ISO 27001 standards for information security management.
- **SOC 2 Type II:** Maintain SOC 2 Type II compliance for service organizations.
- **GDPR:** Ensure the solution complies with GDPR requirements for data protection and privacy.
- **PCI-DSS:** Ensure the solution complies with PCI-DSS standards for payment card data security.

Security Monitoring and Incident Response:
- **AWS CloudWatch:** Use CloudWatch for continuous monitoring of system performance and security events.
- **AWS GuardDuty:** Use GuardDuty for threat detection and automated response to security findings.
- **AWS Config:** Use Config to assess, audit, and evaluate the configurations of AWS resources.
- **Incident Response Plan:** Develop and maintain an incident response plan to quickly address security incidents.

5.10 User Management

5.10 User Management

User Access Management Approach

The user access management approach will leverage AWS Cognito for user authentication and authorization. Cognito provides a scalable and secure way to manage user identities and access to AWS resources.

Authentication Mechanism

The authentication mechanism will utilize AWS Cognito with Single Sign-On (SSO) capabilities. This will allow users to authenticate once and access multiple AWS services without needing to re-enter their credentials.

Authorization Model

The authorization model will implement Role-Based Access Control (RBAC) using AWS Cognito groups. Users will be assigned to specific groups based on their roles within the organization, and each group will have predefined permissions and access levels.

User Provisioning and De-provisioning

User provisioning and de-provisioning will be automated using AWS Cognito's user pool management features. When a new user is onboarded, they will be added to the appropriate Cognito user pool, and when a user leaves the organization, their access will be revoked by removing them from the user pool.

Deployment Architecture Diagram

Figure 1: Deployment Architecture Diagram

Session Management

Session management will be handled by AWS Cognito, which provides built-in support for managing user sessions and tokens. This ensures that user sessions are securely managed and can be revoked if necessary.

Multi-Factor Authentication

Multi-factor authentication (MFA) will be enabled for all users to enhance security. AWS Cognito supports MFA through various methods, including SMS, email, and authenticator apps. Users will be required to set up MFA during the onboarding process.

References

- AWS Cognito Documentation
- AWS Cognito User Pools
- AWS Cognito Authentication
- AWS Cognito Authorization
- AWS Cognito Multi-Factor Authentication

5.11 Solution Deployment

5.11 Solution Deployment

Deployment Architecture Diagram

CI/CD Pipeline Design

The CI/CD pipeline will be designed using AWS CodePipeline and AWS CodeBuild. The pipeline will consist of the following stages:

1. **Source**: Retrieve the source code from the version control system (e.g., Git).
2. **Build**: Compile the source code and run unit tests using AWS CodeBuild.
3. **Test**: Run integration tests and security scans using AWS CodeBuild.
4. **Deploy**: Deploy the application to the target environment using AWS CodeDeploy.

Infrastructure as Code (CDK, CloudFormation)

Infrastructure as Code (IaC) will be implemented using AWS CloudFormation and the AWS Cloud Development Kit (CDK). This will allow for version-controlled, reproducible infrastructure deployments.

Deployment Automation

Deployment automation will be achieved using AWS CodePipeline and AWS CodeDeploy. This will enable automated, hands-off deployments with minimal human intervention.

Blue-green or Canary Deployment Strategy

A blue-green deployment strategy will be used to minimize downtime and risk during deployments. This involves deploying the new version of the application to a separate environment (the 'green' environment) and then switching traffic from the old environment (the 'blue' environment) to the new one.

Rollback Procedures

Rollback procedures will be implemented to quickly revert to a previous version of the application in case of deployment failures or issues. This will involve using AWS CodeDeploy's built-in rollback features.

Environment Promotion Process

The environment promotion process will involve promoting the application from the development environment to the testing environment, and then to the production environment. This will be automated using AWS CodePipeline and AWS CodeDeploy.

References

- AWS CodePipeline Documentation
- AWS CodeBuild Documentation
- AWS CodeDeploy Documentation
- AWS CloudFormation Documentation
- AWS CDK Documentation

5.12 Solution Operations

5.12 Solution Operations

Operational Model and Responsibilities

The operational model will follow a shared responsibility approach, with AWS managing the underlying infrastructure and the customer responsible for managing the application and data. The customer will have dedicated operational teams responsible for monitoring, incident management, and maintenance.

Monitoring and Alerting Procedures

Monitoring and alerting will be implemented using AWS CloudWatch and AWS CloudTrail. CloudWatch will be used to collect and track metrics, collect and monitor log files, and set alarms. CloudTrail will be used to enable governance, compliance, and operational auditing.

Incident Management Process

The incident management process will follow the ITIL framework. Incidents will be logged, categorized, prioritized, and assigned to the appropriate team for resolution. Regular incident reviews will be conducted to identify root causes and improve the incident management process.

Maintenance Windows and Procedures

Maintenance windows will be scheduled during non-business hours to minimize impact on users. Maintenance procedures will include applying security patches, updating AWS services, and performing routine system checks.

Backup and Recovery Procedures

Backup and recovery procedures will be implemented using AWS Backup and AWS S3. Regular backups will be taken of critical data and application components. Recovery procedures will be tested regularly to ensure data can be restored in case of a disaster.

Performance Tuning Approach

Performance tuning will be an ongoing process, with regular reviews of system metrics and performance benchmarks. AWS services such as AWS Lambda, AWS Step Functions, and AWS Batch will be used to optimize performance and reduce costs.

Cost Optimization Strategy

The cost optimization strategy will focus on right-sizing resources, using reserved instances, and leveraging AWS cost management tools such as AWS Cost Explorer and AWS Budgets. Regular cost reviews will be conducted to identify opportunities for cost savings.

References

- AWS CloudWatch Documentation
- AWS CloudTrail Documentation
- ITIL Framework
- AWS Backup Documentation
- AWS S3 Documentation
- AWS Cost Management Tools

6.0 Appendix and Attachments

6.0 Appendix and Attachments

Glossary of Terms

- **AP**: Accounts Payable
- **AWS**: Amazon Web Services
- **CI/CD**: Continuous Integration/Continuous Deployment
- **IaC**: Infrastructure as Code
- **KPI**: Key Performance Indicator
- **RBAC**: Role-Based Access Control
- **RTO**: Recovery Time Objective
- **RPO**: Recovery Point Objective

Acronyms and Abbreviations

- **AP**: Accounts Payable
- **AWS**: Amazon Web Services
- **CI/CD**: Continuous Integration/Continuous Deployment
- **IaC**: Infrastructure as Code
- **KPI**: Key Performance Indicator
- **RBAC**: Role-Based Access Control
- **RTO**: Recovery Time Objective
- **RPO**: Recovery Point Objective

Additional Diagrams

- Deployment Architecture Diagram
- Solution Architecture Diagram

Supporting Documentation References

- AWS Well-Architected Framework

- ITIL Framework
