REPORT  
  
**NFR or Quality Attributes**

Non-functional requirements (NFRs), also known as quality attributes, define the operational qualities of a system, such as how fast it must perform a given task, how easy it is to use, and how reliable it must be. Here are common NFRs that a solution architect might consider for a new system:

1. **Performance**: This includes response time, throughput, and transaction rates the system must achieve. For a recall notification system, it's crucial to process and send notifications promptly.
2. **Scalability**: The system should be able to handle growth in workload (e.g., number of users, number of notifications) without a drop in performance.
3. **Reliability**: This is the probability that the system will function without failure over a specified time. It's particularly critical for systems handling recall notifications to minimize the risk of missed or delayed alerts.
4. **Availability**: The system should be up and accessible for a defined amount of time, typically measured as a percentage (e.g., 99.9% uptime).
5. **Maintainability**: This refers to how easy it is to maintain the system, including performing updates, detecting issues, and restoring operations after a failure.
6. **Security**: The system must protect against unauthorized access to ensure data integrity and confidentiality. This includes measures for authentication, authorization, data encryption, and auditing.
7. **Disaster Recovery**: The ability of the system to recover from catastrophic failures, including data backup and restore procedures, and failover mechanisms.
8. **Usability**: The system should have an intuitive interface and be easy to use, requiring minimal training for end-users.
9. **Compliance**: Adherence to relevant laws, regulations, and standards, which for a recall notification system may include data protection laws and industry regulations.
10. **Cost-Efficiency**: The system should be cost-effective to run, which includes considering the cost of cloud resources, licensing fees, and the operational expense of support and maintenance.
11. **Portability**: The ease with which the system can be transferred from one environment to another, including cloud service providers or between on-premises and cloud environments.
12. **Interoperability**: The ability to work seamlessly with other systems, including third-party services for notifications like SMS gateways or email services.

In the context of cloud-based solutions, these NFRs will guide the selection of appropriate AWS services and the configuration of the environment to meet the operational goals of the recall notification system.

**Integration Needs and Recommended Technologies**

1. **External Communication**: The system should be capable of sending notifications via email, SMS, and potentially other channels like mobile push notifications or instant messaging platforms.
   * **Recommended Technologies**: Amazon Simple Notification Service (SNS) for email and SMS, Amazon Pinpoint for mobile push notifications, and webhooks or APIs for integration with platforms like Slack or Microsoft Teams.
2. **Internal Communication**: The system components need to communicate with each other to pass data and commands around.
   * **Recommended Technologies**: Amazon Simple Queue Service (SQS) for decoupled messaging between services, and AWS Lambda for event-driven processing.
3. **Data Storage**: To store recall data, user data, and transaction logs.
   * **Recommended Technologies**: Amazon RDS or Aurora for relational data, Amazon DynamoDB for NoSQL requirements, and Amazon S3 for object storage and logs.
4. **Identity and Access Management**: Secure and manage user access to various parts of the system.
   * **Recommended Technologies**: AWS Identity and Access Management (IAM) for service-level access control, and Amazon Cognito for user authentication and federation.
5. **Monitoring and Alerting**: The system should be monitored for performance metrics, and alerts should be generated for any operational issues.
   * **Recommended Technologies**: Amazon CloudWatch for monitoring, alerts, and logs, and AWS CloudTrail for auditing API calls.
6. **Data Analytics and Reporting**: For generating reports and gaining insights from the data captured by the system.
   * **Recommended Technologies**: Amazon QuickSight for business intelligence and reporting, Grafana, PowerBI
7. **Third-party Services Integration**: Integration with inventory management systems, CRM, or ERP systems.
   * **Recommended Technologies**: AWS API Gateway for creating API endpoints that external systems can interact with, and AWS Lambda or AWS Step Functions for orchestrating complex integration workflows.

**Recommended Technical Stack**

**Backend**

* **Language**:
  + Primary: Node.js
  + Alternate: Python with Flask or Django
* **Framework**:
  + Primary: Express.js
  + Alternate: AWS Lambda with the Serverless Framework
* **Database**:
  + Primary: AWS Aurora/PostgreSQL
  + Alternate: Amazon RDS for MySQL or Amazon DynamoDB for NoSQL requirements
* **Message Queuing**:
  + Primary: AWS SQS
  + Alternate: Amazon MQ
* **Load Balancer**:
  + Primary: AWS Elastic Load Balancing (ELB)
  + Alternate: NGINX or HAProxy on Amazon EC2
* **Cache**:
  + Primary: Amazon ElastiCache with Redis
  + Alternate: Memcached in Amazon ElastiCache

**Frontend**

* **Framework**:
  + Primary: React.js
  + Alternate: Angular
* **State Management**:
  + Primary: Redux
  + Alternate: Context API with React or Vuex with Vue.js
* **CSS Framework**:
  + Primary: Material-UI
  + Alternate: AWS Amplify UI

**Middleware**

* **API Gateway**:
  + Primary: AWS API Gateway
  + Alternate: Amazon App Runner for running containerized web applications
* **Authentication**:
  + Primary: Amazon Cognito
  + Alternate: OAuth
* **Monitoring**:
  + Primary: Amazon CloudWatch
  + Alternate: Prometheus running on Amazon EC2
* **Logging**:
  + Primary: AWS CloudWatch Logs
  + Alternate: Amazon Elasticsearch Service

**DevOps**

* **CI/CD**:
  + Primary: AWS CodePipeline and AWS CodeBuild , Jenkins
  + Alternate: GitLab CI/CD
* **Containerization**:
  + Primary: AWS Fargate
  + Alternate: Docker on Amazon EC2
* **Deployment**:
  + Primary: Amazon ECS
  + Alternate: Amazon EKS
* **Infrastructure as Code**:
  + Primary: AWS CloudFormation
  + Alternate: Terraform
* **Code Versioning**:
  + Primary: Github
  + Alternate: Gitlab

**Communication**

* **Service Communication**:
  + Primary: Amazon API Gateway for RESTful services, AWS Lambda
  + Alternate: Amazon AppSync for GraphQL
* **Notification Services**:
  + Primary: AWS SNS and Amazon SES
  + Alternate: Amazon Pinpoint for more comprehensive marketing communication services

**Data Storage**

* **Recall Data Storage**:
  + Primary: Amazon RDS
  + Alternate: Amazon Aurora
* **Log Storage**:
  + Primary: Amazon S3
  + Alternate: AWS EFS for file system interface

**Security**

* **Authentication**:
  + Primary: AWS IAM with Amazon Cognito for user management
  + Alternate: Auth0 integrated with AWS IAM
* **Encryption**:
  + Primary: AWS KMS
  + Alternate: AWS Certificate Manager for managing SSL/TLS certificates
* **Access Control**:
  + Primary: AWS IAM for Role-based access control
  + Alternate: Amazon Directory Service in combination with IAM roles

**Monitoring and Logging**

* **Monitoring**:
  + Primary: Amazon CloudWatch
  + Alternate: Datadog with AWS integration
* **Logging**:
  + Primary: AWS CloudWatch Logs
  + Alternate: Amazon OpenSearch Service (successor to Amazon Elasticsearch Service)