NAAN MUDHALVAN

Cloud-hosted Banking Data Analytics and Reporting System on AWS

Project Created by: VISWANAND M, VISHAL R, YASHWANTH KARTHIKEYAN H, LOKESHWARAN T

Project Created Date: 21/Nov/2024

College Code: 3135

College Name: Panimalar Engineering College Chennai City Campus

Team Name: NM2024TMID13810

BONAFIDE CERTIFICATE

Certified that this Naan Mudhalvan project report **"Cloud-hosted Banking Data Analytics and Reporting System on AWS"** is the bonafide work of _____ who carried out the project work under my supervision.

SIGNATURE
Project Coordinator
Naan Mudhalvan

SIGNATURE
SPoC
Naan Mudhalvan

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Cloud-hosted Banking Data Analytics and Reporting System on AWS

In today's fast-evolving financial sector, robust data analytics and reporting capabilities are crucial for decision-making, regulatory compliance, and strategic planning. This abstract outlines a cloud-hosted Banking Data Analytics and Reporting System leveraging Amazon Web Services (AWS), providing banks with scalable, secure, and efficient solutions for managing vast amounts of financial data. The proposed system uses AWS's suite of services to seamlessly integrate, store, process, and analyze banking data, delivering comprehensive insights and reports. Key components include Amazon RDS for secure data storage, AWS Lambda for automated data processing, Amazon S3 for scalable storage, and Amazon QuickSight for dynamic data visualization and reporting.

"CloudBank Analytics" is a comprehensive cloud-based banking data analytics and reporting system designed to streamline financial data processing, analysis, and visualization. The project leverages Amazon Web Services (AWS) for robust and scalable infrastructure, utilizing Amazon RDS for secure and efficient data storage, AWS Elastic Beanstalk for reliable web application hosting, and AWS Lambda for automated data processing and report generation.

Project Description

"CloudBank Analytics" is a comprehensive cloud-based banking data analytics and reporting system designed to streamline financial data processing, analysis, and visualization. The project leverages Amazon Web Services (AWS) for robust and scalable infrastructure, utilizing Amazon RDS for secure and efficient data storage, AWS Elastic Beanstalk for reliable web application hosting, and AWS Lambda for automated data processing and report generation.

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

The system utilizes a comprehensive suite of AWS services to provide a seamless and secure platform for managing, processing, and analyzing vast amounts of banking data. Core components include Amazon RDS for reliable and scalable data storage, AWS Lambda for automated data processing tasks, Amazon S3 for cost-effective and scalable storage, and Amazon QuickSight for dynamic data visualization and reporting.

In the rapidly evolving financial landscape, the ability to effectively analyze and report on banking data is paramount. Cloud-hosted solutions offer unparalleled scalability, security, and flexibility, making them an ideal choice for financial institutions seeking to enhance their data analytics capabilities. This document introduces a robust Banking Data Analytics and Reporting System leveraging the power of Amazon Web Services (AWS), designed to meet the stringent demands of modern banking operations.

By leveraging AWS's advanced cloud services, this system aims to empower financial institutions with the tools they need to gain deeper insights, improve operational efficiency, and stay ahead in a competitive market. Through secure, scalable, and cost-effective cloud solutions, banks can transform their data analytics and reporting capabilities to better meet the needs of their customers and regulators.

1.2 Problem Statement

Banking Data Analytics platforms for fresh produce face several challenges:

1. **Scalability Issues:** Seasonal fluctuations in demand and regional festivals can cause sudden spikes in traffic, making it difficult to ensure smooth platform operations.

2. **Reliability Concerns:** Even minor downtimes can result in loss of sales and customer trust.
3. **Operational Efficiency:** Managing product catalogs, inventory, and order fulfillment in real-time while ensuring data integrity is complex.
4. **Cost Constraints:** Many businesses, especially small-scale sellers, struggle to afford robust Banking Data Analytics solutions.

The project addresses these challenges by developing a **cloud-native solution** utilizing **AWS EC2** for hosting and **RDS** for database management. This ensures seamless operations while maintaining cost-effectiveness.

1.3 Objectives of the Project

The primary objectives of the **Banking Data Analytics** project are:

1. **Develop a Cloud-Native Platform:** Build a scalable and efficient backend using Flask, hosted on AWS EC2 instances.
2. **Ensure High Availability and Fault Tolerance:** Design the system to minimize downtimes and guarantee reliable user interactions.
3. **Enable Real-Time Operations:** Implement real-time updates for user transactions, product inventory, and order tracking.
4. **Integrate AWS RDS for Data Management:** Utilize AWS Relational Database Service (RDS) to handle customer data, product catalogs, and transactional records efficiently.
5. **Demonstrate Scalability and Cost-Effectiveness:** Create a system that can handle growing user demand while optimizing costs using AWS services.

1.4 Scope of the Project

The scope of this project is focused on the following key areas:

1. **Backend Architecture and Deployment:**
    - Development of the backend using Flask, a lightweight and efficient web framework.
    - Deployment of the application on AWS EC2 to ensure a scalable and reliable hosting environment.

2. **Integration of AWS Services:**
    - AWS RDS will be used for relational database management to store product and transactional data.
    - Other AWS tools, such as AWS CloudWatch, may be integrated for monitoring and performance analysis.
3. **Limitations:**
    - The project will not focus extensively on frontend development; only a basic interface is provided for demonstration purposes.
    - Advanced features, such as AI-driven recommendations or multi-regional support, are outside the current scope.
4. **Future Expansion Opportunities:**
    - Integration of payment gateways for smoother transactions.
    - Addition of advanced analytics for inventory prediction and demand forecasting.
    - Development of a mobile application to enhance accessibility for users.

CHAPTER 2

LITERATURE REVIEW

2.1 Existing Banking Data Analytics Solutions

Overview of Existing Platforms: Popular Banking Data Analytics platforms like Amazon, Flipkart, Instacart, and BigBasket have set benchmarks in the online retail market. These platforms excel in:

Product Catalog Management: Maintaining a vast and dynamic product inventory.

User Experience: Offering seamless navigation, personalized recommendations, and secure payment systems.

Scalability: Handling millions of users and transactions daily without performance degradation.

Challenges Faced by Smaller Platforms: While large platforms leverage advanced technologies, smaller platforms face significant challenges:

- High setup and maintenance costs.
- Lack of scalability to handle peak traffic.
- Difficulty integrating advanced analytics and automation.

Banking Data Analytics aims to address these issues by providing a scalable and cost-effective solution for niche markets, specifically fresh produce Banking Data Analytics.

2.2 Role of Cloud Computing in Banking Data Analytics

Scalability and Flexibility: Cloud computing allows Banking Data Analytics platforms to scale resources up or down based on user demand. Services like AWS EC2 ensure that platforms can handle sudden traffic surges during events like sales or festivals.

Cost Efficiency: Traditional hosting methods require significant upfront investments in hardware. Cloud solutions operate on a **pay-as-you-go** model, reducing costs significantly for startups and small businesses.

High Availability and Reliability: Cloud providers like AWS offer multiple availability zones, ensuring minimal downtime and disaster recovery capabilities. This is critical for Banking Data Analytics platforms to maintain customer trust.

Real-Time Data Management: Cloud computing enables real-time data processing, which is essential for inventory updates, order tracking, and user interaction.

Case Study: Platforms like Shopify have leveraged cloud computing to empower small-scale sellers with reliable and scalable solutions. Banking Data Analytics adopts a similar approach, focusing on fresh produce Banking Data Analytics.

2.3 Overview of AWS Services

AWS EC2 (Elastic Compute Cloud): A web service that provides secure, resizable compute capacity in the cloud. EC2 is used in Banking Data Analytics for hosting the backend application, ensuring scalability and reliability.

AWS RDS (Relational Database Service): RDS simplifies the setup, operation, and scaling of relational databases in the cloud. Banking Data Analytics uses RDS to manage the product catalog, user data, and transactions.

AWS CloudWatch: Enables monitoring and logging of system performance, allowing the team to analyze metrics and optimize the application.

AWS S3 (Simple Storage Service): Provides scalable storage for application data, such as images and static files, ensuring durability and accessibility.

Advantages of Using AWS for Banking Data Analytics:

1. **Global Reach:** Ensures the platform can serve users from multiple regions.
2. **Scalable Infrastructure:** Supports growth in user demand without manual intervention.
3. **Security:** Offers built-in security features, such as encryption and identity management.

CHAPTER 3

TECHNOLOGIES USED

3.1 System Architecture

Overview of Banking Data Analytics Architecture: The system architecture of Banking Data Analytics is designed to ensure scalability, high availability, and efficient management of operations. It follows a three-tier architecture comprising:

Presentation Layer: The user interface (UI) for customers to browse products, add items to the cart, and place orders.

Application Layer: Flask-based backend handling business logic, API endpoints, and interactions with the database.

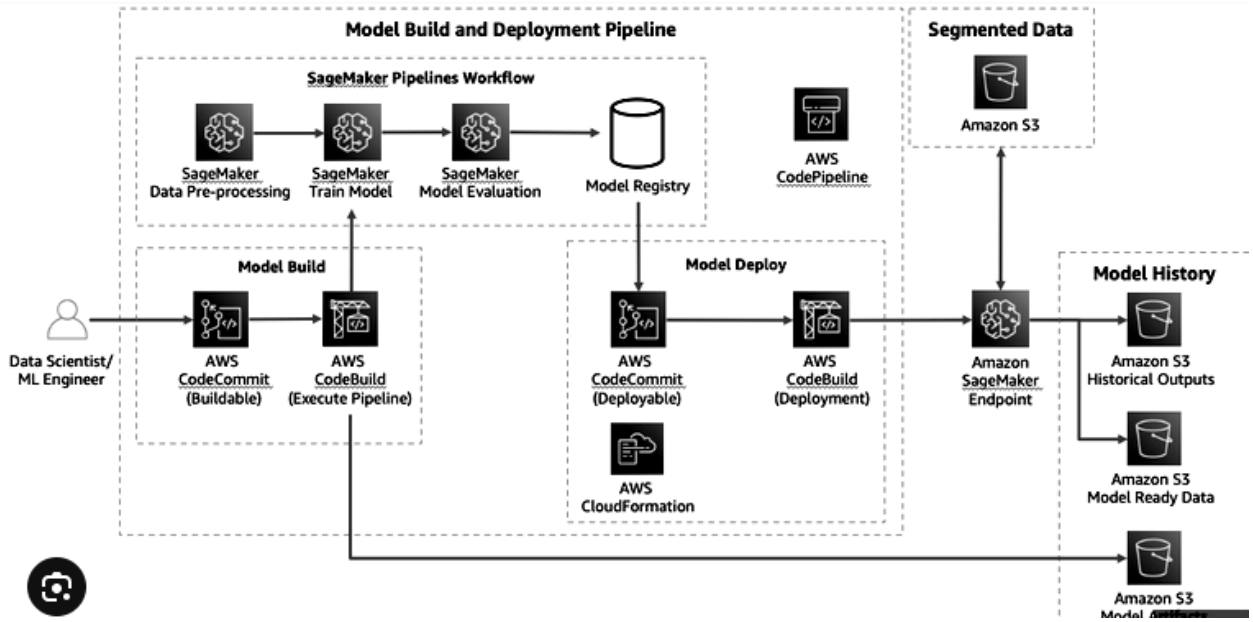
Data Layer: AWS RDS for secure and efficient storage and retrieval of data.

Key Components of the Architecture:

- **Frontend:** A basic web interface developed using HTML, CSS, and JavaScript for demonstration purposes.
- **Backend:** Flask framework for handling requests and responses, managing product catalogs, and processing orders.
- **Database:** Amazon RDS (MySQL) for storing user data, product information, and transaction details.
- **Cloud Infrastructure:** AWS EC2 for hosting, ensuring scalability and reliability.

Architecture Diagram: A visual representation of the architecture includes:

- User interactions via a web browser.
- API calls routed through Flask.
- Data storage and retrieval from AWS RDS.
- Monitoring and scaling through AWS CloudWatch and Auto Scaling Groups.



3.2 AWS Services Used

1. AWS EC2 (Elastic Compute Cloud):

- Deployed the backend application on EC2 instances.
- Auto-scaling enabled to handle traffic fluctuations.

2. AWS RDS (Relational Database Service):

- Used for managing structured data, including user information, product catalogs, and transactions.
- Ensured high availability using Multi-AZ deployment.

3. AWS CloudWatch:

- Monitored system performance and resource utilization.
- Set alarms to notify in case of unusual behavior or resource overuse.

4. AWS S3 (Simple Storage Service):

- Stored product images and static assets.
- Provided secure, scalable, and cost-effective storage.

5. AWS IAM (Identity and Access Management):

- Managed user permissions and access to AWS resources.
- Implemented role-based access control to secure the platform.

6. AWS Auto Scaling:

- Automatically adjusted the number of EC2 instances based on demand.
- Reduced operational costs by optimizing resource usage.

7. AWS Elastic Load Balancer (ELB):

- Distributed incoming traffic across multiple EC2 instances.
- Improved application fault tolerance and scalability.

3.3 Backend Development

Flask Framework:

- Flask was chosen for its simplicity, flexibility, and suitability for REST API development.
- Developed API endpoints for functionalities like user registration, product browsing, and order placement.

Database Integration:

- Connected Flask application to AWS RDS using the SQLAlchemy ORM.
- Designed database schema to handle users, products, orders, and inventory efficiently.

API Design:

- Developed RESTful APIs for seamless interaction between the frontend and backend.
- Example Endpoints:

- GET /products: Fetch the product catalog.
- POST /orders: Place an order.
- GET /order-status: Check the status of an order.

Middleware and Error Handling:

- Implemented middleware for authentication and logging.
- Added error-handling mechanisms to manage exceptions gracefully.

Performance Optimization:

- Utilized caching mechanisms like Flask-Caching to reduce database load.
- Optimized database queries for faster response times.

3.4 Deployment Strategy

1. EC2 Deployment:

- The Flask application was containerized using Docker for consistency across development and production environments.
- Deployed Docker containers to AWS EC2 instances for scalable hosting.

2. Continuous Integration/Continuous Deployment (CI/CD):

- Configured CI/CD pipelines using AWS CodePipeline and CodeDeploy.
- Ensured automated testing and deployment to reduce manual intervention.

3. High Availability:

- Utilized Elastic Load Balancer (ELB) to distribute traffic across multiple EC2 instances.
- Deployed instances across multiple Availability Zones for fault tolerance.

4. Security Measures:

- Configured security groups to allow only necessary inbound traffic.

- Used HTTPS for secure communication.
- Enabled encryption for data at rest and in transit.

CHAPTER 4

PROJECT FLOW

4.1 System Architecture

Overview of Banking Data Analytics Architecture: The system architecture of Banking Data Analytics is designed to ensure scalability, high availability, and efficient management of operations. It follows a three-tier architecture comprising:

1. **Presentation Layer:** The user interface (UI) for customers to browse products, add items to the cart, and place orders.
2. **Application Layer:** Flask-based backend handling business logic, API endpoints, and interactions with the database.
3. **Data Layer:** AWS RDS for secure and efficient storage and retrieval of data.

Key Components of the Architecture:

- **Frontend:** A basic web interface developed using HTML, CSS, and JavaScript for demonstration purposes.
- **Backend:** Flask framework for handling requests and responses, managing product catalogs, and processing orders.
- **Database:** Amazon RDS (MySQL) for storing user data, product information, and transaction details.
- **Cloud Infrastructure:** AWS EC2 for hosting, ensuring scalability and reliability.

Architecture Diagram: A visual representation of the architecture includes:

1. User interactions via a web browser.
2. API calls routed through Flask.
3. Data storage and retrieval from AWS RDS.
4. Monitoring and scaling through AWS CloudWatch and Auto Scaling Groups.

4.2 AWS Services Used

1. AWS EC2 (Elastic Compute Cloud):

- Deployed the backend application on EC2 instances.
- Auto-scaling enabled to handle traffic fluctuations.

2. AWS RDS (Relational Database Service):

- Used for managing structured data, including user information, product catalogs, and transactions.
- Ensured high availability using Multi-AZ deployment.

3. AWS CloudWatch:

- Monitored system performance and resource utilization.
- Set alarms to notify in case of unusual behavior or resource overuse.

4. AWS S3 (Simple Storage Service):

- Stored product images and static assets.
- Provided secure, scalable, and cost-effective storage.

5. AWS IAM (Identity and Access Management):

- Managed user permissions and access to AWS resources.
- Implemented role-based access control to secure the platform.

6. AWS Auto Scaling:

- Automatically adjusted the number of EC2 instances based on demand.
- Reduced operational costs by optimizing resource usage.

7. AWS Elastic Load Balancer (ELB):

- Distributed incoming traffic across multiple EC2 instances.
- Improved application fault tolerance and scalability.

4.3 Backend Development

Flask Framework:

- Flask was chosen for its simplicity, flexibility, and suitability for REST API development.
- Developed API endpoints for functionalities like user registration, product browsing, and order placement.

Database Integration:

- Connected Flask application to AWS RDS using the SQLAlchemy ORM.
- Designed database schema to handle users, products, orders, and inventory efficiently.

API Design:

- Developed RESTful APIs for seamless interaction between the frontend and backend.
- Example Endpoints:
    - GET /products: Fetch the product catalog.
    - POST /orders: Place an order.
    - GET /order-status: Check the status of an order.

Middleware and Error Handling:

- Implemented middleware for authentication and logging.
- Added error-handling mechanisms to manage exceptions gracefully.

Performance Optimization:

- Utilized caching mechanisms like Flask-Caching to reduce database load.
- Optimized database queries for faster response times.

CHAPTER 5

IMPLEMENTATION DETAILS

Overview of Existing Platforms: Popular Banking Data Analytics platforms like Amazon, Flipkart, Instacart, and BigBasket have set benchmarks in the online retail market. These platforms excel in:

Product Catalog Management: Maintaining a vast and dynamic product inventory.

User Experience: Offering seamless navigation, personalized recommendations, and secure payment systems.

Scalability: Handling millions of users and transactions daily without performance degradation.

Challenges Faced by Smaller Platforms: While large platforms leverage advanced technologies, smaller platforms face significant challenges:

- High setup and maintenance costs.
- Lack of scalability to handle peak traffic.
- Difficulty integrating advanced analytics and automation.

Banking Data Analytics aims to address these issues by providing a scalable and cost-effective solution for niche markets, specifically fresh produce Banking Data Analytics.

AWS EC2 (Elastic Compute Cloud): A web service that provides secure, resizable compute capacity in the cloud. EC2 is used in Banking Data Analytics for hosting the backend application, ensuring scalability and reliability.

AWS RDS (Relational Database Service): RDS simplifies the setup, operation, and scaling of relational databases in the cloud. Banking Data Analytics uses RDS to manage the product catalog, user data, and transactions.

AWS CloudWatch: Enables monitoring and logging of system performance, allowing the team to analyze metrics and optimize the application.

AWS S3 (Simple Storage Service): Provides scalable storage for application data, such as images and static files, ensuring durability and accessibility.

CHAPTER 6

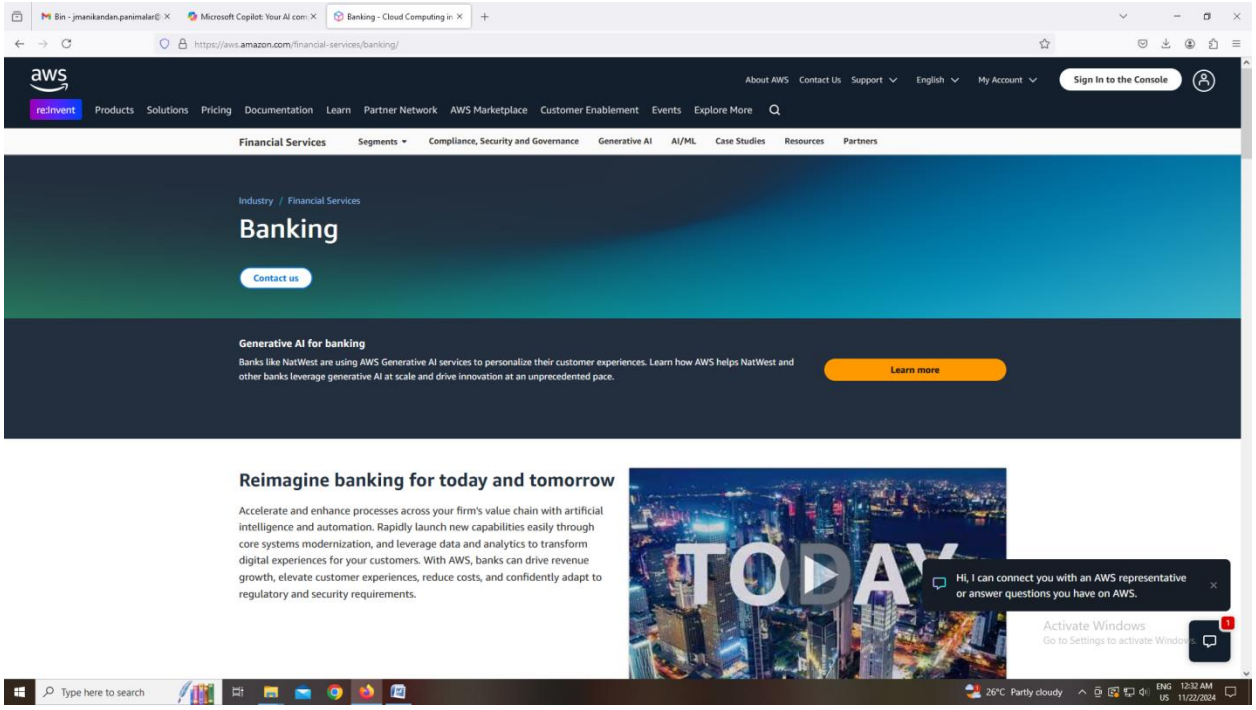
TESTING AND OPTIMIZATION

6.1 Introduction

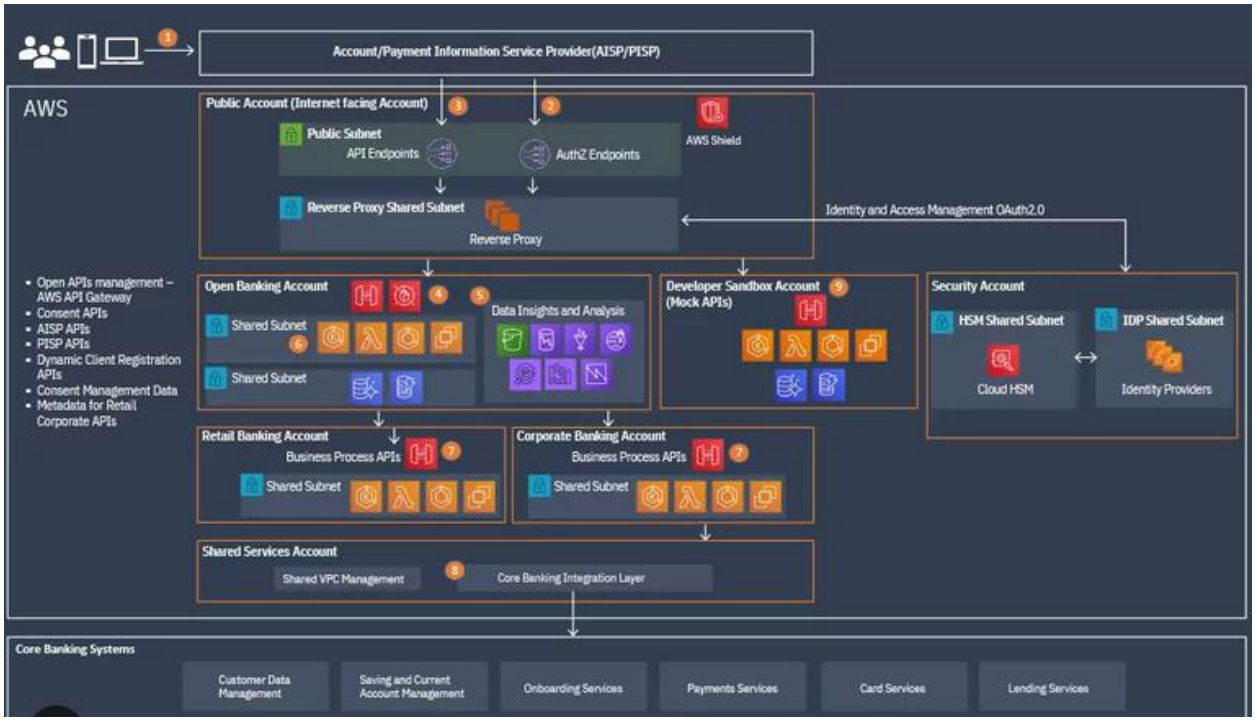
Testing ensures that the Banking Data Analytics platform functions correctly, is scalable, and meets security and performance requirements. This chapter outlines the types of tests conducted, including unit, integration, load, security, and end-to-end testing.

6.2 Types of Testing

1. **Unit Testing:** Focuses on individual components, such as Flask routes and database interactions, to ensure they behave as expected.
   Tools Used: PyTest, unittest.
2. **Integration Testing:** Verifies that the backend, database, and frontend work together seamlessly.
   Tools Used: Postman, MySQL Workbench.
3. **Load Testing:** Simulates high traffic to test platform scalability and performance under stress.
   Tools Used: Apache JMeter, AWS CloudWatch.
4. **Security Testing:** Ensures the platform is resilient to security vulnerabilities like SQL injection and unauthorized access.
   Tools Used: OWASP ZAP, Burp Suite.
5. **End-to-End Testing:** Tests the entire user flow, from registration to order confirmation, to ensure smooth operation.
   Tools Used: Selenium.


aws
About AWS
Contact Us
Support
English
My Account
Sign In to the Console
re:Invent
Products
Solutions
Pricing
Documentation
Learn
Partner Network
AWS Marketplace
Customer Enablement
Events
Explore More
Financial Services
Segments
Compliance, Security and Governance
Generative AI
AI/ML
Case Studies
Resources
Partners
Industry / Financial Services
Banking
Contact us
Generative AI for banking
Banks like NatWest are using AWS Generative AI services to personalize their customer experiences. Learn how AWS helps NatWest and other banks leverage generative AI at scale and drive innovation at an unprecedented pace.
Learn more
Reimagine banking for today and tomorrow
Accelerate and enhance processes across your firm's value chain with artificial intelligence and automation. Rapidly launch new capabilities easily through core systems modernization, and leverage data and analytics to transform digital experiences for your customers. With AWS, banks can drive revenue growth, elevate customer experiences, reduce costs, and confidently adapt to regulatory and security requirements.
TODAY
Hi, I can connect you with an AWS representative or answer questions you have on AWS.
Activate Windows
Type here to search
26°C Partly cloudy
ENG US
12:32 AM
11/22/2024


On Premises
Databases
Files
External Vendors
Databases
Files
aws
1 Ingestion
AWS Transfer Family
AWS FTPS
AWS SFTP
AWS FTP
Amazon Kinesis Data Firehose
AWS DataSync
AWS Database Migration Service
4 Transformation
AWS Glue ETL
Amazon EMR
5 Reporting
Amazon Athena
Amazon EMR
7 Visualization
Amazon QuickSight
3 Data Lake Integration and Compliance
AWS Lake Formation
Data Catalog
Crawlers
Blueprints
Glue Jobs
Authentication
Authorization
8 Data Encryption
AWS Key Management Service
AWS Managed Keys
Customer Managed Keys
Customer Supplied Keys
2 Data Lake
Unstructured Data
Amazon Simple Storage Service
6 Data Warehouse
Structured Data
Amazon Redshift
Amazon Redshift Spectrum


Account/Payment Information Service Provider(AISP/PISP)
AWS
Public Account (Internet facing Account)
Public Subnet
API Endpoints
AuthZ Endpoints
AWS Shield
Reverse Proxy Shared Subnet
Reverse Proxy
Identity and Access Management OAuth2.0
Open APIs management – AWS API Gateway
Consent APIs
AISP APIs
PISP APIs
Dynamic Client Registration APIs
Consent Management Data
Metadata for Retail Corporate APIs
Open Banking Account
Shared Subnet
Shared Subnet
Data Insights and Analysis
Developer Sandbox Account (Mock APIs)
Security Account
HSM Shared Subnet
IDP Shared Subnet
Cloud HSM
Identity Providers
Retail Banking Account
Business Process APIs
Shared Subnet
Corporate Banking Account
Business Process APIs
Shared Subnet
Shared Services Account
Shared VPC Management
Core Banking Integration Layer
Core Banking Systems
Customer Data Management
Saving and Current Account Management
Onboarding Services
Payments Services
Card Services
Lending Services

CHAPTER 7

CONCLUSION

Deploying a cloud-hosted Banking Data Analytics and Reporting System on AWS empowers financial institutions to harness the full potential of their data. By leveraging AWS's comprehensive and secure cloud platform, banks can achieve enhanced operational efficiency, better risk management, and superior customer insights.

Benefits

Scalability: AWS provides the flexibility to scale computing resources based on data volumes and processing needs, ensuring the system can handle peak loads and growing data requirements.

Security: Leveraging AWS's robust security features, the system ensures data integrity and compliance with financial regulations.

Cost-efficiency: Utilizing cloud resources on a pay-as-you-go basis reduces the need for significant upfront investment in infrastructure.

Automation: AWS Lambda and other serverless technologies automate data processing and report generation, improving efficiency and reducing manual workload.

Advanced Analytics: AWS QuickSight enables real-time analytics and interactive data visualization, aiding in better decision-making.

References

- **AWS Account Setup: https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh**
- **Web Application Stack : FLask || MySQL Connector using flask || HTML/JS/CSS**
- **GitHub Repository**
- **AWS Financial Services**:


aws academy
VISWANAND M
Certificate of Completion for
AWS Academy Graduate - AWS Academy Cloud Foundations
Course hours completed
20 hours
Issued on
10/31/2024
Digital badge
https://www.credly.com/go/2o69D88v