Banking Application

This project is dedicated to building a sophisticated banking application in my spare time around my university studies. The application will excel in managing basic banking operations while integrating advanced security and operational features to deliver a secure and user-friendly experience. The development utilizes Java and Spring Boot for the backend to create a RESTful API, ensuring efficient and scalable server-client communication. React.js is used for the front-end to create an engaging user interface, and PostgreSQL for data management, with deployment on AWS RDS showcasing cloud scalability and reliability. All communications are secured using HTTPS, providing a robust security layer for data in transit. The application's progression includes incorporating OAuth 2.0 for enhanced security, a synthetic transaction generator alongside a neural network for anomaly detection, and advanced deployment practices utilizing Docker and Kubernetes for containerization and orchestration. This multi-stage development plan meticulously enhances the application's capabilities, positioning it as a premier project for showcasing development skills in secure, scalable, and efficient software solutions, with a keen focus on financial technology applications.

Stage 1: Core Application Development and AWS Deployment

Objectives:

- Develop the core functionality of the application, focusing on backend operations for managing users, accounts, and transactions.
- Implement a RESTful API for efficient interaction with the application.
- Secure the application using HTTPS.
- Utilize AWS RDS for the application's database to leverage cloud scalability and reliability.

Tasks:

1. **Setup and Configuration**
    - Initialize the Java and Spring Boot project setup with necessary dependencies.
    - Create the basic structure for users, accounts, and transactions models.
2. **RESTful API Development**
    - Design and implement API endpoints to handle CRUD operations.
    - Ensure robust API security and data validation.
3. **HTTPS Configuration**
    - Secure application data transmission using HTTPS.
4. **Database Deployment on AWS RDS**

    - Configure PostgreSQL on AWS RDS, setting up secure access and optimal performance settings.

5. **Application Testing**
    - Conduct unit and integration tests to ensure reliability and performance.
6. **Documentation**
    - Document the setup process, API endpoints, and deployment steps.

Stage 2: Security Enhancements and User Interface Development

Objectives:

- Enhance application security with OAuth 2.0 integration for authentication and authorization.
- Develop a user-friendly interface for easier interaction with the application.

Tasks:

1. **OAuth 2.0 Integration**
    - Implement OAuth 2.0 to secure user authentication and resource authorization.
    - Adapt the user model to incorporate OAuth 2.0 credentials.
2. **User Interface Development**
    - Design and implement a responsive web interface using modern front-end technologies.
    - Ensure the front-end seamlessly interacts with the backend via the RESTful API.
3. **Security Testing**
    - Validate the OAuth 2.0 implementation and overall application security through rigorous testing.

Stage 3: Synthetic Transaction Generator and Neural Network Implementation

Objectives:

- Develop a synthetic transaction generator to create a diverse dataset of financial transactions.
- Train a neural network to identify suspicious transactions, improving the application's security and reliability.

Tasks:

1. **Synthetic Transaction Generator Development**

    - Design and implement a system to generate synthetic financial transactions, considering various patterns, amounts, and behaviours to mimic real-world banking activity.
    - Ensure the generated data covers a wide range of transaction types, including normal and atypical patterns, to train the neural network effectively.
2. **Neural Network Training**
    - Select an appropriate neural network architecture for anomaly detection in transaction data.
    - Train the neural network using the synthetic transaction dataset, tuning the model to recognize patterns indicative of suspicious activity.
3. **Integration with the Banking Application**
    - Incorporate the trained neural network into the application's workflow to automatically analyse transactions in real-time.
    - Implement a system to flag or block transactions deemed suspicious, with a mechanism for manual review and user notification.
4. **Testing and Refinement**
    - Conduct thorough testing of the synthetic transaction generator and neural network model to ensure accuracy and reliability.
    - Adjust the neural network model based on feedback and testing results to improve detection capabilities.

Stage 4: Implementing Two-Factor Authentication and Containerization

Objectives:

- Enhance application security by implementing two-factor authentication (2FA).
- Adopt containerization using Docker and orchestration with Kubernetes to improve deployment processes and scalability.

Tasks:

1. **Two-Factor Authentication Implementation**
    - Research and select a 2FA method (e.g., SMS, email, authenticator app) that best suits the application's needs.
    - Integrate 2FA into the user authentication flow, ensuring a seamless and secure user experience.
2. **Containerization with Docker**

- Containerize the banking application and its components using Docker, creating consistent and isolated environments for development, testing, and production.
- Define and configure Docker files and Docker Compose files for the application services and dependencies.

3. **Orchestration with Kubernetes**
    - Deploy the containerized application on a Kubernetes cluster, configuring Kubernetes resources such as pods, services, and deployments for optimal performance and scalability.
    - Implement monitoring and logging services within the Kubernetes environment to ensure high availability and to troubleshoot potential issues.
4. **Security and Performance Testing**
    - Conduct comprehensive security testing, particularly focusing on the effectiveness and user experience of 2FA.
    - Test the containerized application in various scenarios to ensure performance, scalability, and reliability are maintained.