**Assignment: Bank Application with Spring Boot**

**Overview:**

In this assignment, you will create a Spring Boot application that mimics a bank. The application will provide several APIs to manage bank accounts, including opening an account with an option to add initial credit, retrieving account balance, retrieving person information, and making transfers between accounts. The assignment will be divided into three phases.

**Phase 1: Basic API Implementation and Unit Testing**

1. **Swagger Define Contracts (Contract First)**
   * Define an API contract using Swagger/OpenAPI. This will include:
     + API to open an account with the option to add initial credit.
     + API to retrieve account information including account balance.
     + API to retrieve person information.
     + API to make a transfer from one account to another.
2. **Setup SCM (GIT)**
   * Set up a Git repository for the project following best practices.
   * Ensure to create a .gitignore file to exclude unnecessary files.
   * Follow a consistent commit message convention and branching strategy.
3. **Spring Boot Application**
   * Create a Spring Boot application skeleton.
   * Implement the APIs as defined in the Swagger contracts.
     + **Open Account API:** Accepts user details and optional initial credit amount. Creates an account and initializes balance.
     + **Retrieve Account Details API:** Retrieves the current account details of a specified account.
     + **Retrieve Person Information API:** Retrieves information about the account holder.
     + **Transfer API:** Transfers a specified amount from one account to another.
   * **Unit Testing:**
     + Write unit tests for each API endpoint using JUnit and Mockito.
     + Ensure to achieve high test coverage for critical functionalities.
   * **Error Handling**:
     + Make sure that all errors are handled gracefully and are logged
     + Use meaningful exceptions and provide a message to the user with the correct status codes
4. **Database Setup**
   * Use an in-memory H2 database for development and testing purposes.
   * Define the necessary JPA entities for the bank application.
   * Configure Spring Data JPA repositories for database operations.

**Phase 2: Versioning and Security**

1. **Versioning**
   * Create a **v2 of the Swagger contract:**
     + Add a required property “birthdate” to the person and make sure no minors are allowed to create an account
     + Add the security configuration from below to the contract
     + Use proper branching strategy to allow V1 to still be adapted.
   * **Implement contract changes** into your Spring Boot application
     + The version of the API should be reflected in the URL
     + There is no need to support both versions in the same application
2. **Security**
   * **Authentication:**
     + Create a /login endpoint to authenticate the user
     + Configure Spring Security to secure the APIs using OAuth2 token-based authentication.
     + Ensure that all APIs are accessible only through proper authentication.
   * **Authorization**:
     + Make sure a user can only see his own account and person details

**Phase 3: CI/CD**

1. **Create a Pipeline to Build the Application into a Docker Container**
   * **Dockerize the Application:**
     + Write a Dockerfile to containerize your application.
     + Ensure the Dockerfile builds and runs the application correctly.
   * **Create a CI Pipeline:**
     + Use Azure Pipelines
     + The pipeline should:
       1. Clone the repository.
       2. Build the Docker image using the Dockerfile.
       3. Push the Docker image to a container registry (Azure Container Registry).
     + Include steps to run unit tests as part of the build process.
   * **Documentation:**
     + Document the steps involved in setting up the CI pipeline.
2. **Create Infrastructure (IAC)**
   * **Set Up Infrastructure as Code:**
     + Choose a tool for IAC (Terraform, ARM templates, or Bicep).
     + Define the infrastructure needed to deploy your application. This typically includes:
     + Azure Resource Group
     + Azure Kubernetes Service (AKS) or Azure App Service for containers
     + Azure Container Registry (if not using a public registry)
     + Networking components (VNET, Subnets, etc.)
   * **Write IAC Scripts:**
     + Write the Terraform/ARM/Bicep scripts to provision the required resources.
     + Ensure the scripts are idempotent and can handle updates gracefully.
   * **Create a CI Pipeline for IAC:**
     + Use Azure Pipelines tool to create a pipeline.
     + The pipeline should:
       1. Clone the repository.
       2. Validate the IAC scripts.
       3. Apply the IAC scripts to provision the infrastructure on Azure.
     + **Documentation:**
       1. Document the IAC scripts and the steps involved in provisioning the infrastructure.
       2. Provide the IAC configuration files and pipeline configuration file.
3. **Create a Pipeline to Deploy to Environment on Azure Stack**
   * **Set Up Deployment Pipeline:**
     + Use Azure Pipelines to create a deployment pipeline.
     + The pipeline should:
       1. Pull the latest Docker image from the container registry.
       2. Deploy the Docker image to the provisioned Azure environment (AKS or Azure App Service).
   * **Deployment Configuration:**
     + Configure deployment strategies (e.g., rolling updates, blue-green deployments).
     + Set up environment-specific configurations and secrets (e.g., using Azure Key Vault).
   * **Automate Rollbacks (optional):**
     + Implement rollback mechanisms in case of deployment failures.
   * **Monitoring and Logging(optional):**
     + Set up monitoring and logging for the deployed application using Azure Monitor, Log Analytics, or other tools.
   * **Documentation:**
     + Document the deployment pipeline steps and configurations.
     + Provide the pipeline configuration file.

**Submission Requirements:**

* **Phase 1:**
  + Swagger contracts.
  + Git repository URL with proper branching strategy.
  + Spring Boot application code.
  + Unit tests with high coverage.
  + Database schema and configuration.
* **Phase 2:**
  + OAuth2 configuration and implementation.
  + Security configuration for the APIs.
* **Phase 3:**
  + As mentioned above

**Notes:**

* Follow best practices for coding, security, and documentation.
* Ensure that the project is well-documented, including instructions on how to set up and run the application.
* Pay attention to exception handling and validation in the API implementations.
* Use consistent coding standards and naming conventions throughout the project.

**Helpful Resources:**

* [Spring Boot Documentation](https://spring.io/projects/spring-boot)
* [Spring Security Documentation](https://spring.io/projects/spring-security)
* Swagger/OpenAPI Documentation
* [Azure DevOps Documentation](https://docs.microsoft.com/en-us/azure/devops/)

By completing this assignment, you will gain practical experience in building a robust and secure Spring Boot application, setting up CI/CD pipelines, and deploying applications to cloud platforms.