Take Home coding test

We want you to create a REST API for a fictional bank called Eagle Bank which conforms to the provided OpenAPI specification and the following scenarios. The API will allow a user to add their details, fetch, update, detail their details. A user can also create, fetch, update and delete their own bank accounts and deposit or withdraw money from the account. Withdrawing and depositing money will be stored as transactions against a bank account which be retrieved but not modified or deleted.

You can spend as little or as much time on this tech test, but it is your opportunity to showcase your skills. ***We do not expect all the endpoints*** to be completed for submission but at least we expect some of the basic operations to be ready (ie, Create and Fetch for User, Account and Transaction). This is designed to simulate real world tasks that you may be required to complete in the role, so ***AI code assistants can be used*** to help you with this solution. Once submitted a member of the hiring team will review the code and may invite you to a follow up session to walk them through the code and do a pair/ mob coding session to extend the solution.

For each of the following scenarios, appropriate error handling should be included to handle invalid or missing credentials.

Please submit your solution by pushing your code to a public GitHub repo and sharing a link to the hiring team as soon as it's completed. For the coding interview itself, we will be looking at your code while you share on a Teams call or in person in our office if you decide to do it this way. In both cases, you'll be expected to walk through the application you've designed and to explain all of the trade offs you've made when picking different technologies and specific choices of your code.

We would like this API to be coded in Java, .NET or Javascript as they are the supported languages that we have in Barclays.

If you require longer than 7 days, please reach out to the hiring team.

Create a user

Scenario: Create a new user

Given a user wants to signup for Eagle Bank
When the user makes a `POST` request to the `/v1/users` endpoint with all the required data
Then a new user is created

Scenario: Create a new user without supplying all required data

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/users` endpoint with required data missing
Then the system returns a Bad Request status code and error message

Authenticate a user

We want you to implement one or more endpoints to authenticate a user and return a JWT token which can be passed as a bearer token to all endpoints expect creating a user. Please update the OpenAPI specification with the details of the endpoint you implement to authenticate a user and submit it as part of your solution.

Fetch a user

Scenario: User wants to fetch their user details

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/users/{userId}` endpoint supplying their `userId`
Then the system fetches the user details

Scenario: User wants to fetch the user details of another user

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/users/{userId}` endpoint supplying another user's `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to fetch the user details of a non-existent user

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/users/{userId}` endpoint supplying a `userId` which doesn't exist
Then the system returns a Not Found status code and error message

Update a user

Scenario: User wants to update their user details

Given a user has successfully authenticated
When the user makes a `PATCH` request to the `/v1/users/{userId}` endpoint supplying their `userId` and all the required data
Then the system updates the user details and returns the updated data

Scenario: User wants to update the user details of another user

Given a user has successfully authenticated
When the user makes a `PATCH` request to the `/v1/users/{userId}` endpoint supplying another user's `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to fetch the user details of a non-existent user

Given a user has successfully authenticated
When the user makes a `PATCH` request to the `/v1/users/{userId}` endpoint supplying a `userId` which doesn't exist
Then the system returns a Not Found status code and error message

Delete a user

Scenario: User wants to delete their user details

Given a user has successfully authenticated
When the user makes a `DELETE` request to the `/v1/users/{userId}` endpoint
And they do not have a bank account
Then the system deletes their user

Scenario: User wants to delete their user details and they have a bank account

Given a user has successfully authenticated
When the user makes a `DELETE` request to the `/v1/users/{userId}` endpoint
And they have a bank account
Then the system returns a Conflict status code and error message

Scenario: User wants to delete user details of another user

Given a user has successfully authenticated
When the user makes a `DELETE` request to the `/v1/users/{userId}` endpoint
And the `userId` is associated with another user
Then the system returns a Forbidden status code and error message

Scenario: User wants to delete user details of a non-existent user

Given a user has successfully authenticated
When the user makes a `DELETE` request to the `/v1/users/{userId}` endpoint
And the `userId` doesn't exist
Then the system returns a Not Found status code and error message

Create a Bank Account

Scenario: User wants to create a new bank account

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts` endpoint with all the required data
Then a new bank account is created, and the account details are returned

Scenario: User wants to create a new bank account without supplying all required data

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts` endpoint with required data missing
Then the system returns a Bad Request status code and error message

List bank accounts

Scenario: User wants to view their bank accounts

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts` endpoint
Then all the bank accounts associated with their `userId` are returned

Fetch a Bank Account

Scenario: User wants to fetch their bank account details

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}` endpoint
And the account is associated with their `userId`
Then the system fetches the bank account details

Scenario: User wants to fetch another user's bank account details

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}` endpoint
And the account is not associated with their `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to fetch a non-existent bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}` endpoint
And the `accountId` doesn't exist
Then the system returns a Not Found status code and error message

Update a Bank Account

Scenario: User wants to update their bank account details

Given a user has successfully authenticated
When the user makes a `PATCH` request to the `/v1/accounts/{accountId}` endpoint supplying all the required data
And the account is associated with their `userId`
Then the system updates the bank account information and returns the updated data

Scenario: User wants to fetch another user's bank account details

Given a user has successfully authenticated
When the user makes a `PATCH` request to the `/v1/accounts/{accountId}` endpoint
And the account is not associated with their `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to fetch a non-existent bank account

Given a user has successfully authenticated
When the user makes a `PATCH` request to the `/v1/accounts/{accountId}` endpoint
And the `accountId` doesn't exist
Then the system returns a Not Found status code and error message

Delete a Bank Account

Scenario: User deletes an existing bank account

Given a user has successfully authenticated
When the user makes a `DELETE` request to the `/v1/accounts/{accountId}` endpoint
And the account is associated with their `userId`
Then the system deletes the bank account

Scenario: User wants to delete another user's bank account details

Given a user has successfully authenticated
When the user makes a `DELETE` request to the `/v1/accounts/{accountId}` endpoint
And the account is not associated with their `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to delete a non-existent bank account

Given a user has successfully authenticated
When the user makes a `DELETE` request to the `/v1/accounts/{accountId}` endpoint
And the `accountId` doesn't exist
Then the system returns a Not Found status code and error message

Create a Transaction

Scenario: User wants to deposit money into their bank account

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts/{accountId}/transactions` endpoint with all the required data
And the transaction type is `deposit`
And the account is associated with their `userId`
Then the deposit is registered against the account
And the account balance is updated

Scenario: User wants to withdraw money from their bank account

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts/{accountId}/transactions` endpoint with all the required data
And the transaction type is `withdrawal`
And the account has sufficient funds
And the account is associated with their `userId`
Then the withdrawal is registered against the account
And the account balance is updated

Scenario: User wants to withdraw money from their bank account but they have insufficient funds

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts/{accountId}/transactions` endpoint with all the required data
And the transaction type is `withdrawal`
And the account has insufficient funds
And the account is associated with their `userId`
Then the system returns a Unprocessable Entity status code and error message

Scenario: User wants to deposit or withdraw money into another user's bank account

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts/{accountId}/transactions` endpoint with all the required data
And the account is not associated with their `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to deposit or withdraw money into a non-existent bank account

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts/{accountId}/transactions` endpoint with all the required data
And the `accountId` doesn't exist
Then the system returns a Not Found status code and error message

Scenario: User wants to deposit or withdraw money without supplying all required data

Given a user has successfully authenticated
When the user makes a `POST` request to the `/v1/accounts/{accountId}/transactions` endpoint with required data missing
Then the system returns a Bad Request status code and error message

List Transactions

Scenario: User wants to view all transactions on their bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions` endpoint
And the account is associated with their `userId`
Then the transactions are returned

Scenario: User wants to view all transactions on another user's bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions` endpoint
And the account is not associated with their `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to view all transactions on a non-existent bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions` endpoint
And the `accountId` doesn't exist
Then the system returns a Not Found status code and error message

Fetch a Transaction

Scenario: User wants to fetch a transaction on their bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions/{transactionId}` endpoint
And the account is associated with their `userId`
And the `transactionId` is associated with the `accountId` specified
Then the transaction details are returned

Scenario: User wants to fetch a transaction on another user's bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions/{transactionId}` endpoint
And the account is not associated with their `userId`
Then the system returns a Forbidden status code and error message

Scenario: User wants to fetch a transaction on a non-existent bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions/{transactionId}` endpoint
And the `accountId` doesn't exist
Then the system returns a Not Found status code and error message

Scenario: User wants to fetch a transactions on a non-existent transaction ID

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions/{transactionId}` endpoint
And the account is associated with their `userId`
And the `transactionId` does not exist
Then the system returns a Not Found status code and error message

Scenario: User wants to fetch a transaction against the wrong bank account

Given a user has successfully authenticated
When the user makes a `GET` request to the `/v1/accounts/{accountId}/transactions/{transactionId}` endpoint
And the account is associated with their `userId`
And the `transactionId` is not associated with the `accountId` specified
Then the system returns a Not Found status code and error message