|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Online Banking System**  **Technical Design Document**   |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** |  |  |  | | **Role** |  |  |  | | **Signature** |  |  |  | | **Date** |  |  |  | |

Table of Contents

[**1.** **Project Overview:** 3](#_Toc143197889)

[**2.** **Project Duration and Scope:** 3](#_Toc143197890)

[**3.** **Business Problem:** 3](#_Toc143197891)

[**4.** **Project Requirements:** 3](#_Toc143197892)

[**4.1.** **Functional Requirements:** 3](#_Toc143197893)

[**4.2.** **Non-functional Requirements:** 4](#_Toc143197894)

[**5.** **Data Model / Entity Description:** 4](#_Toc143197895)

[**5.1.** **User Entity:** 4](#_Toc143197896)

[**5.2.** **Account Entity:** 4](#_Toc143197897)

[**5.3.** **Transaction Entity:** 5](#_Toc143197898)

[**6.** **Architecture Design Guidelines:** 5](#_Toc143197899)

[**7.** **Technology Stack:** 5](#_Toc143197900)

[**8.** **Evaluation Criteria:** 5](#_Toc143197901)

[**9.** **Deliverables:** 6](#_Toc143197902)

[**10.** **Timeline:** 6](#_Toc143197903)

[**11.** **Resources:** 6](#_Toc143197904)

[**12.** **Support and Communication:** 6](#_Toc143197905)

[**13.** **Change Log** 6](#_Toc143197906)

## **Project Overview:**

The "Online Banking System" project aims to develop a secure and user-friendly online platform for managing personal finances and conducting banking transactions. This includes building a backend microservice for account management and a user-friendly frontend using Java's Spring Boot framework and Thymeleaf for templating. Trainees will gain hands-on experience in backend development, security implementation, UI design, and frontend-backend integration using Java technologies.

## **Project Duration and Scope:**

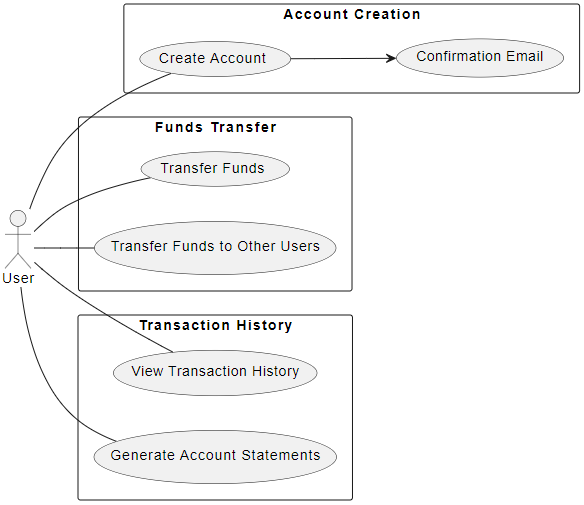
The project spans 120 hours over five weeks. It involves designing and implementing the backend for account creation, funds transfer, and transaction history. Participants will also create a user-friendly and responsive frontend using Thymeleaf templates.

## **Business Problem:**

Customers face challenges in managing their finances efficiently due to the lack of a user-friendly and secure online banking platform. This project aims to address these challenges by providing a comprehensive online banking system.

## **Project Requirements:**

## **Functional Requirements:**



**User Story 1: Account Creation**

* As a user, I want to create a bank account online and set up login credentials.

**Acceptance Criteria:**

* Users should be able to create an account by providing personal information and choosing a username and password.
* Users should receive a confirmation email upon successful account creation.

**User Story 2: Funds Transfer**

* As a user, I want to transfer funds between my accounts or to other users.

**Acceptance Criteria:**

* Users should be able to initiate fund transfers between their linked accounts.
* Users should also be able to transfer funds to other users within the bank.

**User Story 3: Transaction History**

* As a user, I want to view my transaction history and account statements.

**Acceptance Criteria:**

* Users should be able to view a detailed transaction history for each account.
* Users should be able to generate and download account statements.

## **Non-functional Requirements:**

* **Security:** Implement secure authentication, authorization, and encryption of sensitive data.
* **User Experience:** Develop a responsive and user-friendly UI using Thymeleaf templates.
* **Integration:** Establish communication between the frontend and backend using RESTful APIs.

## **Data Model / Entity Description:**

## **User Entity:**

**Attributes:**

* UserID (Primary Key)
* FirstName
* LastName
* Email
* Username
* Password (Hashed)

## **Account Entity:**

**Attributes:**

* AccountID (Primary Key)
* UserID (Foreign Key)
* AccountNumber
* Balance
* AccountType (Checking, Savings, etc.)

## **Transaction Entity:**

**Attributes:**

* TransactionID (Primary Key)
* FromAccountID (Foreign Key)
* ToAccountID (Foreign Key)
* Amount
* TransactionDate

## **Architecture Design Guidelines:**

* **Communication:** Implement RESTful APIs for communication between frontend and backend components.
* **Database:** Utilize a PostgreSQL database for storing user, account, and transaction data.
* **Deployment:** Deploy the backend using Spring Boot and host the frontend on a web server.

## **Technology Stack:**

|  |  |
| --- | --- |
| **Backend (Java)** | |
| **Programming Language** | Core Java 12 |
| **Framework** | Spring Boot |
| **Database** | MySQL |
| **Authentication** | JWT |
| **Backend (.NET)** | |
| **Programming Language** | C# |
| **Framework** | ASP.NET Core Web API |
| **Database** | SQL Server |
| **Authentication** | JWT |
| **Frontend** | |
| **Framework** | Choose either Angular or React for frontend development. |
| **UI Components** | Develop interactive UI components for account management, fund transfer, and transaction history. |
| **Communication** | Use REST APIs to communicate with the backend microservice. |

## **Evaluation Criteria:**

* Successful implementation of functional requirements on both frontend and backend.
* Effective integration of frontend and backend components.
* Security measures for data transmission, user authentication, and authorization.
* User-friendly UI design, responsiveness, and navigation.
* Code quality, documentation, and error handling.
* Project presentation and demonstration.

## **Deliverables:**

* Source code for the backend microservice and the Thymeleaf-based frontend.
* Comprehensive API documentation detailing endpoints, request-response formats, and authentication mechanisms.
* Unit tests with sufficient code coverage for backend services.
* Logging and proper exception handling in the backend code.
* Deployment instructions for both backend and frontend components.
* Project summary report discussing challenges and solutions.

## **Timeline:**

* Days 1-2: Project setup, technology selection, and architecture design.
* Days 3-5: Backend microservice development and API implementation.
* Days 6-9: Frontend UI development and UI component implementation.
* Days 10: Integration of frontend and backend components.

## **Resources:**

|  |  |  |
| --- | --- | --- |
| **Backend(Java)** | Core Java | <https://www.geeksforgeeks.org/java/> |
| Spring Boot Microservices | <https://www.geeksforgeeks.org/java-spring-boot-microservices-example-step-by-step-guide/> |
| Data JPA | <https://spring.io/guides/gs/accessing-data-jpa/> |
| Unit Testing | <https://www.springboottutorial.com/unit-testing-for-spring-boot-rest-services> |
| **Backend(.NET)** | C# | <https://www.geeksforgeeks.org/csharp-programming-language/> |
| ASP.NET Core Microservices | <https://www.c-sharpcorner.com/article/microservice-using-asp-net-core/>  <https://learn.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/data-driven-crud-microservice> |
| Entity Framework Core | <https://www.tektutorialshub.com/entity-framework-core-tutorial/> |
| Unit Testing | <https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/testing?view=aspnetcore-3.1> |
| **Frontend (Angular/React)** | Angular | <https://angular.io/docs> |
|  | React | <https://react.dev/learn> |
| **Integration (Java + Angular/React)** | Spring Boot REST API + Angular | <https://www.javaguides.net/2021/01/angular-spring-boot-rest-api-example.html> |
| Spring Boot REST API + React | <https://reflectoring.io/build-responsive-web-apps-with-springboot-and-react-tutorial/> |
| **Integration (.NET + Angular/React)** | ASP.NET Core +Angular | <https://levelup.gitconnected.com/kubernetes-angular-asp-net-core-microservice-architecture-c46fc66ede44> |
| ASP.NET Core +React | <https://learn.microsoft.com/en-us/visualstudio/javascript/tutorial-asp-net-core-with-react?view=vs-2022> |

## **Support and Communication:**

* Regular progress updates through daily stand-up meetings.
* Communication and assistance available through designated communication channels.

## **Change Log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version Number | Changes made | | | |
| V<n.n> | *<If the change details are not explicitly documented in the table below, reference should be provided here>* | | | |
| Page no | Changed by | Effective date | Changes effected |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |