# System Requirements Specification Index

For

# Mobile Banking Application

Version 1.0

**IIHT Pvt. Ltd.**
IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

## Table of Contents

# Mobile Banking Application

## System Requirements Specification

## Project Abstract

In the fast-paced world of banking, there's a growing demand for modern mobile banking solutions. The CEO of a leading financial institution, Ms. Johnson, challenges a team of developers to create a Fullstack Mobile Banking Management Application.

Your task is to develop a digital solution that empowers users with seamless mobile banking capabilities.

# BACKEND-DOTNET

## 1. Business-Requirement:

**Mobile Banking Management** Application is a .Net Core web API 3.1 application integrated with MS SQL Server, where it refers to introducing new features for mobile and online banking applications and displaying available features, account information, and transaction history on digital platforms.

To build a robust backend system that powers the Mobile Banking Management Application. Here's what the developers need to accomplish:

### Following is the requirement specification:

| | Mobile Banking Management |
|---|---|
| | |
| Modules | |
| 1 | Mobile Banking |
| | |
| Mobile Banking Module Functionalities | |
| | |
| 1 | Create a Feature |
| 2 | Update the existing Feature |
| 3 | Get a Feature by Id |
| 4 | Fetch all Features |
| 5 | Delete an existing Features |
| | |

## 2. Assumptions, Dependencies, Risks / Constraints

### 2.1 Mobile Banking Constraints:

- While deleting the Mobile Banking, if Mobile Banking Id does not exist then the operation should throw a custom exception.
- While fetching the Mobile Banking details by id, if Mobile Banking id does not exist then the operation should throw a custom exception.

### 2.2 Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in model classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

## 3. Business Validations

### Mobile Banking Class Entities

- Feature Id (long) Not null, Key attribute.
- Transaction Id (string) Not null.
- Feature Name (string) is not null, min 3 and max 100 characters.
- Description (string) is not null.
- Name (string)
- UserName(string)

## 4. Considerations

- There is no roles in this application
- You can perform the following  possible actions

| |
|---|
| Mobile Banking |

## 5. Rest Endpoints

Rest End-points to be exposed in the controller along with method details for the same to be created

### 5.1 Mobile BankingController

| URL Exposed | | Purpose |
|---|---|---|
| /create-feature | | Create Feature |
| Http Method | POST | |
| Parameter 1 | Mobile Banking model | |
| Return | HTTP Response StatusCode | |
| /update-feature | | Update a feature |
| Http Method | PUT | |
| Parameter 1 | Long Id | |
| Parameter 2 | Mobile BankingViewModel model | |
| Return | HTTP Response StatusCode | |
| /get-all-features | | Fetches the list of all Features |
| Http Method | GET | |
| Parameter 1 | - | |
| Return | <IEnumerable<Mobile Banking >> | |
| /get-feature-by-id?id={id} | | Fetches the details of a feature |
| Http Method | GET | |
| Parameter 1 | Long (id) | |
| Return | <Mobile Banking> | |
| /delete-feature?id={id} | | Delete a feature |
| Http Method | DELETE | |
| Parameter 1 | Long (id) | |
| Return | HTTP Response StatusCode | |

# 6. Template Code Structure

## 6.1 Package: Mobile BankingManagement

**Resources**

| Names | Resource | Remarks | Status |
|---|---|---|---|
| Package Structure | | | |
| controller | MobileBankingController | Controller class to expose all rest-endpoints for auction related activities. | Partially implemented |
| Startup.cs | Startup CS file | Contain all Services settings and SQL server Configuration. | Already Implemented |
| Properties | launchSettings.json file | All URL Setting for API | Already Implemented |
| | appsettings.json | Contain connection string for database | Already Implemented |

## 6.2 Package: Mobile BankingManagement.BusinessLayer

**Resources**

| Names | Resource | Remarks | Status |
|---|---|---|---|
| Package Structure | | | |
| Interface | IMobileBankingServices interface | Inside all these interface files contains all business validation logic functions. | Already implemented |

| | | | |
|---|---|---|---|
| Service | MobileBankingServices CS file | Using this all class we are calling the Repository method and use it in the program and on the controller. | Partially implemented |
| Repository | IMobileBanking Repository<br><br>MobileBanking Repository<br><br>(CS files and interfaces) | All these interfaces and class files contain all CRUD operation code for the database.<br>Need to provide implementation for service related functionalities | Partially implemented |
| ViewModels | MobileBanking ViewModel | Contain all view Domain entities for show and bind data.<br>All the business validations must be implemented. | Partially implemented |

## 6.3 Package: Mobile BankingManagement.DataLayer

**Resources**

| Names | Resource | Remarks | Status |
|---|---|---|---|
| Package Structure | | | |
| DataLayer | MobileBankingDBContext cs file | All database Connection,collection setting class | Already Implemented |

## 6.4 Package: Mobile BankingManagement.Entities

**Resources**

| Names | Resource | Remarks | Status |
|---|---|---|---|
| Package Structure | | | |
| Entities | MobileBanking ,Response ( CS files) | All Entities/Domain attribute are used for pass the data in controller and status entity to return response<br><br>Annotate this class with proper annotation to declare it as an entity class with **Id** as primary key.<br><br>Generate the **Id** using the **IDENTITY** strategy | Partially implemented |

# FRONTEND-REACT SPA

## 1 PROBLEM STATEMENT

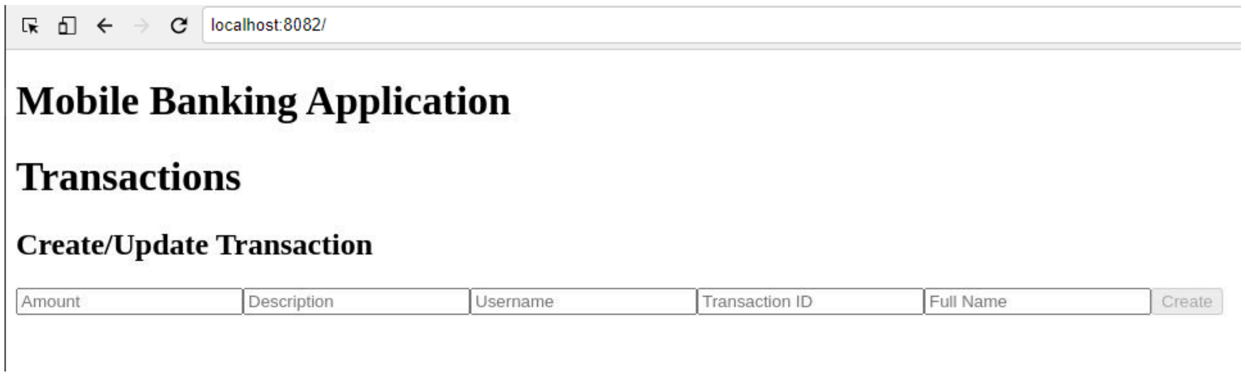
The **Mobile Banking Application** frontend is a Single Page Application (SPA) built using React. Here's what the frontend developers need to achieve:

The frontend should provide an intuitive user interface for easy navigation and efficient mobile banking operations.

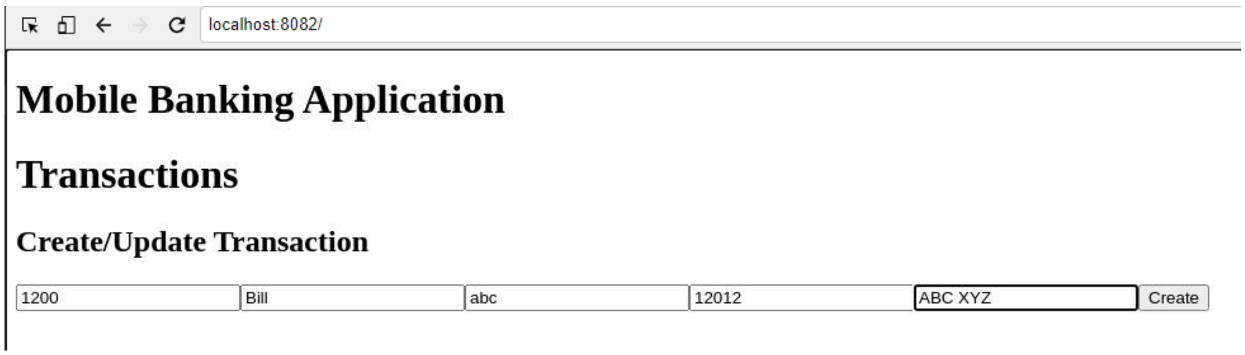
## 2 PROPOSED MOBILE BANKING WIREFRAME

UI needs improvisation and modification as per given use case and to make test cases passed.

### 2.1 HOME PAGE



**** write any id as string in "Transaction ID" field like below 12012**

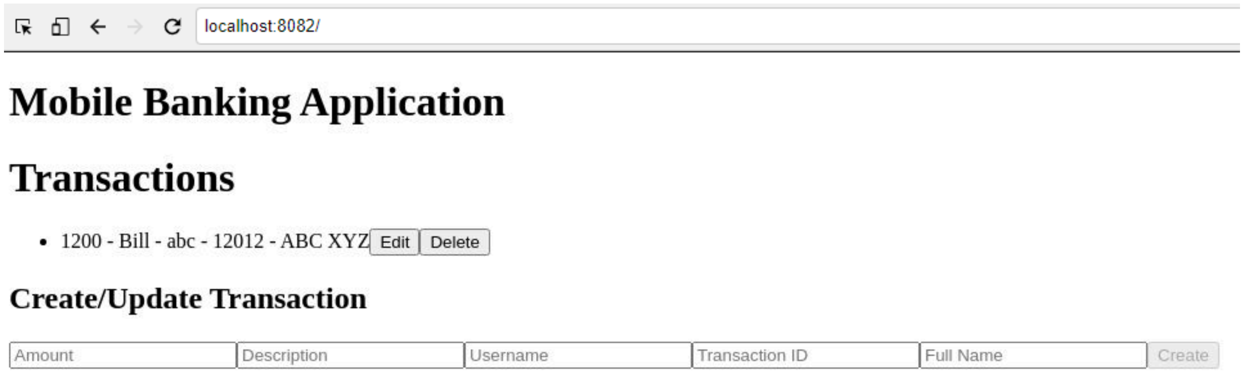

localhost:8082/

Mobile Banking Application

Transactions

Create/Update Transaction

| | | | | | |
|---|---|---|---|---|---|
| 1200 | Bill | abc | 12012 | ABC XYZ | Create |


localhost:8082/

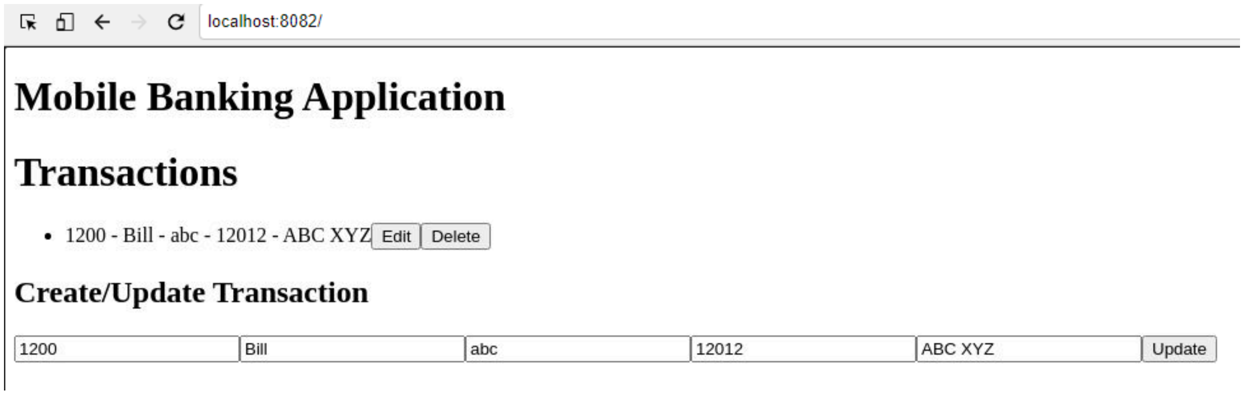
# Mobile Banking Application

## Transactions

- 1200 - Bill - abc - 12012 - ABC XYZ Edit Delete

### Create/Update Transaction

| | | | | | |
|---|---|---|---|---|---|
| Amount | Description | Username | Transaction ID | Full Name | Create |


localhost:8082/

# Mobile Banking Application

## Transactions

- 1200 - Bill - abc - 12012 - ABC XYZ Edit Delete

### Create/Update Transaction

| | | | | | |
|---|---|---|---|---|---|
| 1200 | Bill | abc | 12012 | ABC XYZ | Update |

## 3 BUSINESS-REQUIREMENT:

As an application developer, develop the Mobile Banking Application (Single Page App) with below guidelines:

| User Story # | User Story Name | User Story |
|---|---|---|
| US_01 | Home Page | As a user I should be able to visit the Home page as the default page. |
| US_01 | Home Page | Acceptance criteria:<br>1. Should have “Mobile Banking Application” as heading in h2.<br>2. Should have a “Transactions” as heading in h2.<br>3. Should show a list of all transactions with “Edit” & “Delete” button in each of the transactions..<br>4. As a user I should be able to furnish the following details at the time of creating/updating a transaction.<br>1.1 Amount<br>1.2 Description<br>1.3 Username<br>1.4 Transaction ID<br>1.5 Full Name<br>5. All fields should be required fields to add a transaction<br>6. Until all fields are filled, the create button should be disabled. |

## Execution Steps to Follow For Backend

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**

2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) Terminal → New Terminal.**

3. **On command prompt, cd into your project folder (cd <Your-Project-folder>).**

4. **To connect SQL server from terminal:**
   **(MobileBankingManagement /sqlcmd -S localhost -U sa -P pass@word1)**
   - **To create database from terminal -**

     ```
     1> Create Database MobileDb
     2> Go
     ```

5. **Steps to Apply Migration(Code first approach):**
   - **Press Ctrl+C to get back to command prompt**
   - **Run following command to apply migration-**
     **(MobileBankingManagement /dotnet-ef database update)**

6. **To check whether migrations are applied from terminal:**
   **(MobileBankingManagement /sqlcmd -S localhost -U sa -P pass@word1)**

   ```
   1> Use MobileDb
   2> Go
   1> Select * From __EFMigrationsHistory
   2> Go
   ```

7. **To build your project use command:**
   **(MobileBankingManagement /dotnet build)**

8. **To launch your application, Run the following command to run the application:**
   **(MobileBankingManagement /dotnet run)**

9. **This editor Auto Saves the code.**

10. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**

11. **To test web-based applications on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

    **Note: The application will not run in the local browser**

12. **To run the test cases in CMD, Run the following command to test the application: (MobileBankingManagement.Tests/dotnet test --logger "console;verbosity=detailed") (You can run this command multiple times to identify the test case status,and refactor code to make maximum test cases passed before final submission)**

13. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B - command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

14. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**

15. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**

## EXECUTION STEPS TO FOLLOW FOR FRONTEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to**

   **Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.**
3. **This is a web-based application, to run the application on a browser, use the internal browser in the environment.**
4. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min**

   b. **npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:8082 to open project in browser -> takes 2 to 3 min**

   a. **npm run jest -> to run all test cases and see the summary**

   c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
5. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**