System Requirements Specification Index

For

Investment Management App

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

Investment Management
System Requirements Specification

---

1. Business-Requirement:

1.1 Problem Statement:

Fintech, short for financial technology, is revolutionizing how we manage and interact with money. It's making financial services faster, easier, and more accessible to everyone around the globe. From online banking to investment apps, fintech is all about leveraging technology to streamline financial processes and services.

Global fintech innovations are reshaping the investment sector, introducing groundbreaking technologies that enhance how individuals and institutions invest and manage their portfolios. These innovations offer streamlined, tech-driven solutions for a range of investment activities, from real-time stock trading to automated wealth management and personalized investment advice.

In our organization., we're introducing an Investment Management module(Part-1) as part of Global Fintech Innovations' website. This tool is designed to simplify investment tracking and management for individuals and organizations alike.

Investment Management Application is .Net Core web API 3.1 application integrated with MS SQL Server and frontend being implemented using ReactJs, where it refers to the professional management of various securities and assets to meet specific investment goals for individuals, institutions, or organizations. This process includes the creation, updating, retrieval, and deletion of investment related properties

Key Features:

- **Investment Tracking:** A user-friendly interface to monitor investments with real-time updates.
- **Easy Management:** APIs for adding new investments, updating existing ones, fetching details by ID, and removing investments no longer needed.
- **Secure and Efficient:** Our module prioritizes security and efficiency, ensuring user data is protected while providing a seamless experience.

1.2 Following is the requirement specification:

	Investment Management
	This core module is designed to facilitate comprehensive investment activities, enabling users to efficiently manage their investment portfolio
Modules	
1	Investment
Investment Module Functionalities	
1	**Create an investment**: Allows users to initiate and add new investments to their portfolio, capturing essential details to start the investment process.
2	**Update the existing Investment**: Provides the capability to modify details of existing investments, ensuring that investment portfolios remain up-to-date and reflect current market conditions or user preferences.
3	**Get an Investment by Id**: Enables users to quickly access specific investment details using a unique identifier, facilitating easy review and management of individual investments.
4	**Fetch all investments:** Offers a comprehensive view of all investments within a user's portfolio, allowing for effective tracking and management of overall investment strategy.
5	**Delete an existing Investment:** Grants the ability to remove investments from the portfolio, ensuring that it accurately represents the user's current investment strategy and preferences.

2. Assumptions, Dependencies, Risks / Constraints

2.1 Investment Constraints:

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

Investment Class Entities

- Investment Id (long) Not null, Key attribute.
- Investor Id (int) Not null.
- Investment Name (string) is not null, min 3 and max 100 characters.
- Initial Investment Amount (decimal) is not null.
- Investment StartDate (Date)
- Investment StartDate (Date) Not null.

4. Considerations

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

Investment

5. Rest Endpoints

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

5.1 InvestmentController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/create-investment
<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>Investment model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Create Investment</td></tr>
<tr><td>/update-investment
<table><tr><td>Http Method</td><td>PUT</td></tr><tr><td>Parameter 1</td><td>Long Id</td></tr><tr><td>Parameter 2</td><td>InvestmentViewModel model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Update an Investment</td></tr>
<tr><td>/get-all-investments
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Investment >></td></tr></table></td><td>Fetches the list of all Investments</td></tr>
<tr><td>/get-investment-by-id?id={id}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long (id)</td></tr><tr><td>Return</td><td><Investment></td></tr></table></td><td>Fetches the details of an Investment</td></tr>
<tr><td>/delete-investment?id={id}
<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>Long (id)</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Delete an Investment</td></tr>
</table>

6. Template Code Structure

6.1 Package: InvestmentManagement

Resources

Names	Resource	Remarks	Status
Package Structure			
Controller	InvestmentController	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: InvestmentManagement.BusinessLayer

Resources

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

Service	Investment Services 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	IInvestment Repository Investment Repository (CS files and interfaces)	All these interfaces and class files contain all CRUD operation code for the database. Need to provide implementation for service related functionalities	Partially implemented
ViewModels	Investment ViewModel	Contain all view Domain entities for show and bind data. All the business validations must be implemented.	Partially implemented

6.3 Package: InvestmentManagement.DataLayer

Resources

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

6.4 Package: InvestmentManagement.Entities

Resources

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

FRONTEND - REACT SPA

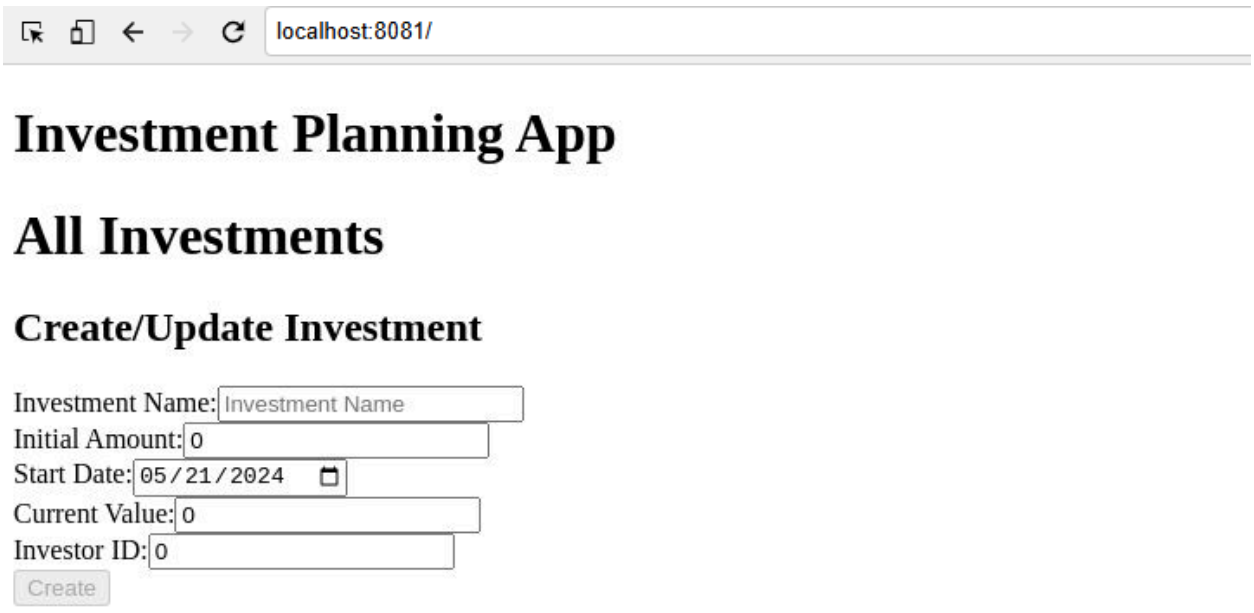
1. PROBLEM STATEMENT

Investment Planning Application is SPA (Single Page Application), it allows you to add investment plan details, update investment plan details, delete investment plans, get all investment plans and get investment plans by id.

2. PROPOSED INVESTMENT PLANNING APPLICATION WIREFRAME

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

2.1 HOME PAGE



***** Create Investment*****

Investment Planning App

All Investments

Create/Update Investment

Investment Name: Mutual Fund
Initial Amount: 10000
Start Date: 05/21/2024
Current Value: 5000
Investor ID: 1
Create

Investment Planning App

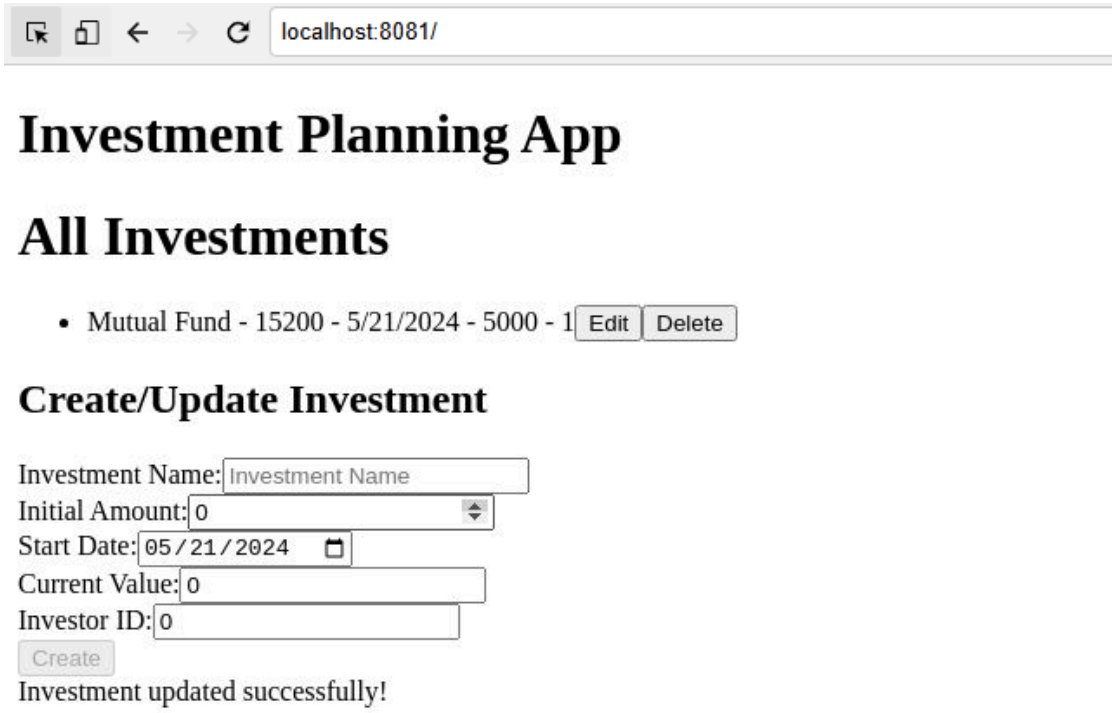
All Investments

- Mutual Fund - 10000 - 5/21/2024 - 5000 - 1 Edit Delete

Create/Update Investment

Investment Name: Investment Name
Initial Amount: 0
Start Date: 05/21/2024
Current Value: 0
Investor ID: 0
Create
Investment created successfully!

***** Update Investment*****



3. BUSINESS-REQUIREMENT:

As an application developer, develop the Investment Planning 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	As a user I should be able to see the homepage and perform all operations: Acceptance criteria: 1. Should have “Investment Planning App” and “All Investments” as heading in h1. 2. Should have “Create/Update Investment” as heading in h2. 3. A list of all investment plans should be visible with the “Edit” and “Delete” button in each of the investment plans. 4. As a user I should be able to furnish the following details at the time of creating an investment plan. 1.1 Investment Name 1.2 Initial Amount 1.3 Start Date 1.4 Current Value 1.5 Investor ID The “Create” button should be disabled by default, and should be enabled when all fields are filled. 5. Same form should be used to add and update an investment.

4. 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:
   (InvestmentManagement /sqlcmd -S localhost -U sa -P pass@word1)
   - To create database from terminal -

     1> Create Database InvestmentDb
     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-
     (InvestmentManagement /dotnet-ef database update)

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

   1> Use InvestmentDb
   2> Go
   1> Select * From __EFMigrationsHistory
   2> Go

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

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

---

5. 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 React 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:8081 to open project in browser -> takes 2 to 3 min**
   c. **npm run jest -> to run all test cases and see the summary**
   d. **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.**