# System Requirements Specification

# Index

For

# Investment Planning Application

Version 1.0

# TABLE OF CONTENTS

# INVESTMENT PLANNING APPLICATION

## System Requirements Specification

**You need to consume APIs exposed by Backend application in React to make application work as FULLSTACK**

# BACKEND-SPRING BOOT RESTFUL APPLICATION

## 1 PROJECT ABSTRACT

The **Investment Planning Application** is a FullStack Application with a backend implemented using Spring Boot with a MySQL database and a frontend developed using React. The application aims to provide a comprehensive platform for managing and organizing all investments done by any user.

**Following is the requirement specifications**:

| | Investment Planning Application |
|---|---|
| | |
| Modules | |
| 1 | Investment Planning |
| | |
| Investment Planning Module Functionalities | |
| | |
| 1 | Get all investments |
| 2 | Get investment by id |
| 3 | Create an investment |
| 4 | Update an investment |
| 5 | Delete an investment |
| 6 | Get all investments by category |

# 2 Assumptions, Dependencies, Risks / Constraints

## 2.1 INVESTMENT PLANNING CONSTRAINTS

- When fetching an investment by ID, if the investment ID does not exist, the service method should throw “Investment not found” exception using NotFoundException class.
- When updating an investment by ID, if the investment ID does not exist, the service method should throw “Investment not found” exception using NotFoundException class.
- When removing an investment by ID, if the investment ID does not exist, the service method should throw “Investment not found” exception using NotFoundException class.

## 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 dto 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

- Name should not be blank.
- Amount should not be null and always be positive value.
- Date should not be null.
- Category should not be blank.

# 4 Rest endpoints

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

## 4.1 INVESTMENTPLANNINGCONTROLLER

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>1. /api/investments<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter</td><td>-</td></tr></table></td><td>Fetches all the investments</td></tr>
</table>

| | | |
|---|---|---|
| Return | List<InvestmentPlanningDTO> | |
| 2. / api/investments/{id} | | Fetches an investment plan by id |
| Http Method | GET | |
| Parameter 1 | Long (id) | |
| Return | InvestmentPlanningDTO | |
| 3. / api/investments | | Creates a new investment plan |
| Http Method | POST<br><br>**The investment data to be created must be accepted in @RequestBody.** | |
| Parameter 1 | - | |
| Return | InvestmentPlanningDTO | |
| 4. / api/investments/{id} | | Updates an investment plan by id |
| Http Method | PUT<br><br>**The investment data to be updated must be accepted in @RequestBody.** | |
| Parameter 1 | Long (id) | |
| Return | InvestmentPlanningDTO | |
| 5. / api/investments/{id} | | Deletes an investment plan by id |
| Http Method | DELETE | |
| Parameter 1 | Long (id) | |
| Return | - | |
| 6. / api/investments/category/{category} | | Fetches the list of all investment plans by category |
| Http Method | GET | |
| Parameter 1 | String (category) | |
| Return | List<InvestmentPlanningDTO> | |

# 5 TEMPLATE CODE STRUCTURE

## 5.1 PACKAGE: COM.INVESTMENTPLANNING

**Resources**

| | | |
|---|---|---|
| **InvestmentPlanningApplication (Class)** | This is the Spring Boot starter class of the application. | Already Implemented |

## 5.2 PACKAGE: COM.INVESTMENTPLANNING.REPOSITORY

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **InvestmentPlanningRepository (interface)** | • Repository interface exposing CRUD functionality for InvestmentPlanning Entity.<br>• You can go ahead and add any custom methods as per requirements. | Partially implemented. |

## 5.3 PACKAGE: COM.INVESTMENTPLANNING.SERVICE

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **InvestmentPlanningService (interface)** | • Interface to expose method signatures for investment planning related functionality.<br>• Do not modify, add or delete any method. | Already implemented. |

## 5.4 PACKAGE: COM. INVESTMENTPLANNING.SERVICE.IMPL

| Class/Interface | Description | Status |
|---|---|---|
| **InvestmentPlanningServiceImpl (class)** | • Implements InvestmentPlanningService.<br>• Contains template method implementation.<br>• Need to provide implementation for investment planning related functionalities.<br>• Do not modify, add or delete any method signature | To be implemented. |

## 5.5 PACKAGE: COM.INVESTMENTPLANNING.CONTROLLER

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **InvestmentPlanningController (Class)** | • Controller class to expose all rest-endpoints for investment planning related activities.<br>• May also contain local exception handler methods | To be implemented |

## 5.6 PACKAGE: COM.INVESTMENTPLANNING.DTO

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **InvestmentPlanningDTO (Class)** | Use appropriate annotations for validating attributes of this class. | Partially implemented. |

## 5.7 PACKAGE: COM.INVESTMENTPLANNING.ENTITY

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **InvestmentPlanning (Class)** | • This class is partially implemented.<br>• Annotate this class with proper annotation to declare it as an entity class with **id** as primary key.<br>• Map this class with an **investment planning** table.<br>• Generate the **id** using the IDENTITY strategy | Partially implemented. |

## 5.8 PACKAGE: COM.INVESTMENTPLANNING.EXCEPTION

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **NotFoundException (Class)** | • Custom Exception to be thrown when trying to fetch, update or delete the investment plan info which does not exist.<br>• Need to create Exception Handler for same wherever needed (local or global) | Already implemented. |

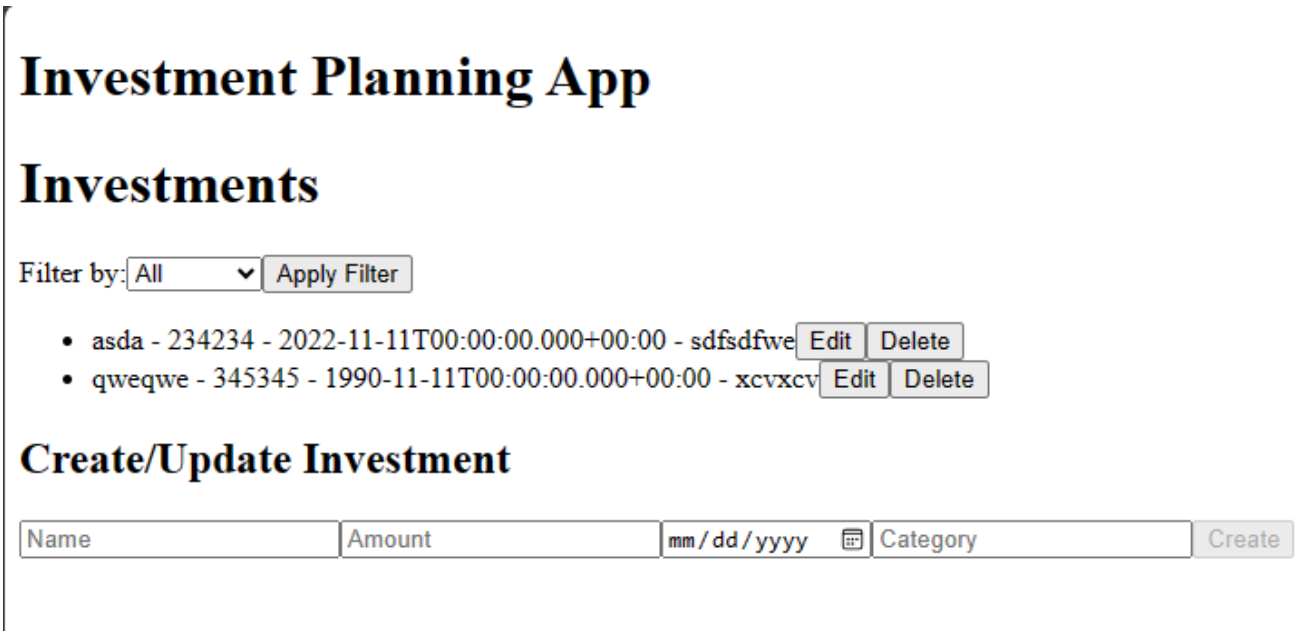
# 1 PROBLEM STATEMENT

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 all investment plans by category.

# 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




# 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:<br><br>Acceptance criteria:<br>1. There must be a heading (h1) as “Investments”.<br>2. A dropdown with label “Filter by:” should be there with all unique category values with “Apply Filter” button.<br>3. A list of all investment plans should be visible with “Edit” and “Delete” button in each of the investment plan.<br>4. As a user I should be able to furnish the following details at the time of creating an investment plan.<br>1.1 Name<br>1.2 Amount<br>1.3 Date<br>1.4 Category<br>The “Create” button should be disabled by default, and should be enabled when all fields are filled.<br>5. “Create/Update Investment” must be there in h2 heading.<br>6. Same form should be used to add and update an investment plan and a button must be there with “Create” text while creating an investment plan and “Update” when updating an investment plan. |

# 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. **cd into your backend project folder**
4. **To build your project use command:**
   **mvn clean package -Dmaven.test.skip**
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**
   **java -jar <your application jar file name>**
6. **This editor Auto Saves the code.**
7. **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.**
8. **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.**
9. **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.**
10. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
11. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**
11. **To login to mysql instance: Open new terminal and use following command:**
    a. **sudo systemctl enable mysql**
    b. **sudo systemctl start mysql**
    c. **mysql -u root -p**
    **The last command will ask for password which is 'pass@word1'**
12. **Mandatory: Before final submission run the following command:**
    **mvn test**

13. **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:4200 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.**