System Requirements Specification Index

For

CrowdFunding Platform

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

Crowdfunding Platform

System Requirements Specification

PROJECT ABSTRACT

In the dynamic world of crowdfunding, there's a growing need for modern platforms that connect project creators with potential backers. The CEO of a visionary startup, Mr. Patel, challenges a team of developers to create a Fullstack Crowdfunding Platform.

Your task is to develop a digital solution that empowers users to create and support / manage crowdfunding campaigns, facilitating the investments of innovative projects.

BACKEND-JAVA

1. Problem Statement

The **Crowdfunding Platform** is a Java-based RESTful Web API utilizing Spring Boot, with MySQL as the database. The application aims to provide a comprehensive platform for projects and investment management.

To build a robust backend system that powers the Crowdfunding Platform. Here's what the developers need to accomplish:

Following is the requirement specification:

	Crowdfunding Platform
Modules	
1	Investment
2	Project
Investment Module Functionalities	
1	Can create/make investments
2	Can update an investment
3	Can delete an investment
4	Get investment by investment id
5	Get investments by project id
6	Get investments by investor name
Project Module Functionalities	
1	Can create a project
2	Can update a project
3	Can delete a project
4	Get project by id
5	Get all projects

2. Assumptions, Dependencies, Risks / Constraints

2.1 Investment Constraints

- When fetching an investment by id, if the investment ID does not exist, the service method should throw a “Investment not found” message in the ResourceNotFoundException class.
- When updating an investment , if the investment ID does not exist, the service method should throw a “Investment not found” message in the ResourceNotFoundException class.
- When removing an investment , if the investment ID does not exist, the service method should throw a “Investment not found” message in the ResourceNotFoundException class.

2.2 Project Constraints

- When fetching a project by id, if the project ID does not exist, the service method should throw a “Project not found” message in the ResourceNotFoundException class.
- When updating a project , if the project ID does not exist, the service method should throw a “Project not found” message in the ResourceNotFoundException class.
- When removing a project , if the projectID does not exist, the service method should throw a “Project not found” message in the ResourceNotFoundException class.

2.3 Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exceptions 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

- Investment amount should not be null and must be at least 1.
- Investor name should not be null and max 255 characters.
- Project ID should not be null.
- Project name should not be blank and max 255 characters.
- Project description should not be blank and max 2000 characters.
- Goal amount should not be null and must be at least 1.
- Amount raised 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 Investment Controller

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>1. /api/investments/{investmentId}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter</td><td>Long (investmentId)</td></tr><tr><td>Return</td><td>InvestmentDTO</td></tr></table></td><td>Fetches the investment by investmentId</td></tr>
<tr><td>2. /api/investments/project/{projectId}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long (projectId)</td></tr><tr><td>Return</td><td>List<InvestmentDTO></td></tr></table></td><td>Fetches the investments by projectId</td></tr>
<tr><td>3. /api/investments/
<table><tr><td>Http Method</td><td>POST

The investment data to be created should be received in @RequestBody</td></tr><tr><td>Parameter</td><td>InvestmentDTO</td></tr><tr><td>Return</td><td>InvestmentDTO</td></tr></table></td><td>Creates a new investment</td></tr>
</table>

4. /api/investments/{investmentId}	
Http Method	PUT **The investment data to be updated should be received in @RequestBody**
Parameter 1	Long (investmentId)
Parameter 2	InvestmentDTO
Return	InvestmentDTO
Updates an investment by id	

5. /api/investments/{investmentId}	
Http Method	DELETE
Parameter 1	Long (investmentId)
Return	-
Deletes an investment by id	

6. /api/investments/investor/{investorName}	
Http Method	GET
Parameter	String (investorName)
Return	List<InvestmentDTO>
Fetches the investments by investor name	

4.2 Project Controller

URL Exposed		Purpose
1. /api/projects/{projectId}		
Http Method	GET	Fetches the project by project id
Parameter	Long (id)	
Return	ProjectDTO	
1. /api/projects/		
Http Method	GET	Fetches all the projects
Parameter	-	
Return	List<ProjectDTO>	
3. /api/projects/		

Http Method	POST **The project data to be created should be received in @RequestBody**	Creates a new project
Parameter	ProjectDTO	
Return	ProjectDTO	
4. /api/projects/{projectId}		
Http Method	PUT **The project data to be updated should be received in @RequestBody**	Updates a project by id
Parameter 1	Long (projectId)	
Parameter 2	ProjectDTO	
Return	ProjectDTO	
5. /api/projects/{projectId}		
Http Method	DELETE	Deletes a project by id
Parameter 1	Long (projectId)	
Return	-	

5. Template Code Structure

5.1 Package: com.crowdfunding

Resources

Class/Interface	Description	Status
CrowdFundingPlatform Application (Class)	This is the Spring Boot starter class of the application.	Already implemented.

5.2 Package: com.crowdfunding.repo

Resources

Class/Interface	Description	Status
InvestmentRepository (interface)	• Repository interface exposing CRUD functionality for investment Entity. • You can go ahead and add any custom methods as per requirements.	Already implemented.
ProjectRepository (interface)	• Repository interface exposing CRUD functionality for project Entity. • You can go ahead and add any custom methods as per requirements.	Already implemented.

5.3 Package: com.crowdfunding.service

Resources

Class/Interface	Description	Status
InvestmentService (interface)	• Interface to expose method signatures for investment related functionality. • Do not modify, add or delete any method.	Already implemented.
ProjectService (interface)	• Interface to expose method signatures for project related functionality. • Do not modify, add or delete any method.	Already implemented.

5.4 Package: com.crowdfunding.service.impl

Resources

Class/Interface	Description	Status
InvestmentServiceImpl (class)	• Implements InvestmentService. • Contains template method implementation. • Need to provide implementation for investment related functionalities. • Do not modify, add or delete any method signature	To be implemented.
ProjectServiceImpl (class)	• Implements ProjectService. • Contains template method implementation. • Need to provide implementation for project related functionalities.	To be implemented.

	• Do not modify, add or delete any method signature	

5.5 Package: Com.Crowdfunding.Controller

Resources

Class/Interface	Description	Status
InvestmentController (Class)	• Controller class to expose all rest-endpoints for investment related activities. • Should also contain local exception handler methods	To be implemented
ProjectController (Class)	• Controller class to expose all rest-endpoints for project related activities. • Should also contain local exception handler methods	To be implemented

5.6 Package: Com.Crowdfunding.Dto

Resources

Class/Interface	Description	Status
InvestmentDTO (Class)	• Use appropriate annotations for validating attributes of this class.	Partially implemented.
ProjectDTO (Class)	• Use appropriate annotations for validating attributes of this	Partially implemented.

	class.	

5.7 PACKAGE: COM.CROWDFUNDING.ENTITY

Resources

Class/Interface	Description	Status
Investment (Class)	• This class is partially implemented. • Annotate this class with proper annotation to declare it as an entity class with **Id** as primary key. • Map this class with an **investment** table. • Generate the **id** using the IDENTITY strategy	Partially implemented.
Project (Class)	• This class is partially implemented. • Annotate this class with proper annotation to declare it as an entity class with **Id** as primary key. • Map this class with a **project** table. • Generate the **id** using the IDENTITY strategy	Partially implemented.

5.8 PACKAGE: COM.CROWDFUNDING.EXCEPTION

Resources

Class/Interface	Description	Status
ResourceNotFoundException (Class)	• Custom Exception to be thrown when trying to fetch, update or delete the investment, project info which does not exist. • Need to create Exception Handler for same wherever needed (local or global)	Already implemented.
ErrorResponse (Class)	• RestControllerAdvice Class for defining global exception handlers. • Contains Exception Handler for **InvalidDataException** class. • Use this as a reference for creating exception handler for other custom exception classes	Already implemented.
RestExceptionHandler (Class)	• RestControllerAdvice Class for defining rest exception handlers. • Contains Exception Handler for **ResourceNotFoundException** class. • Use this as a reference for creating exception handler for other custom exception classes	Already implemented.

FRONTEND-REACT SPA

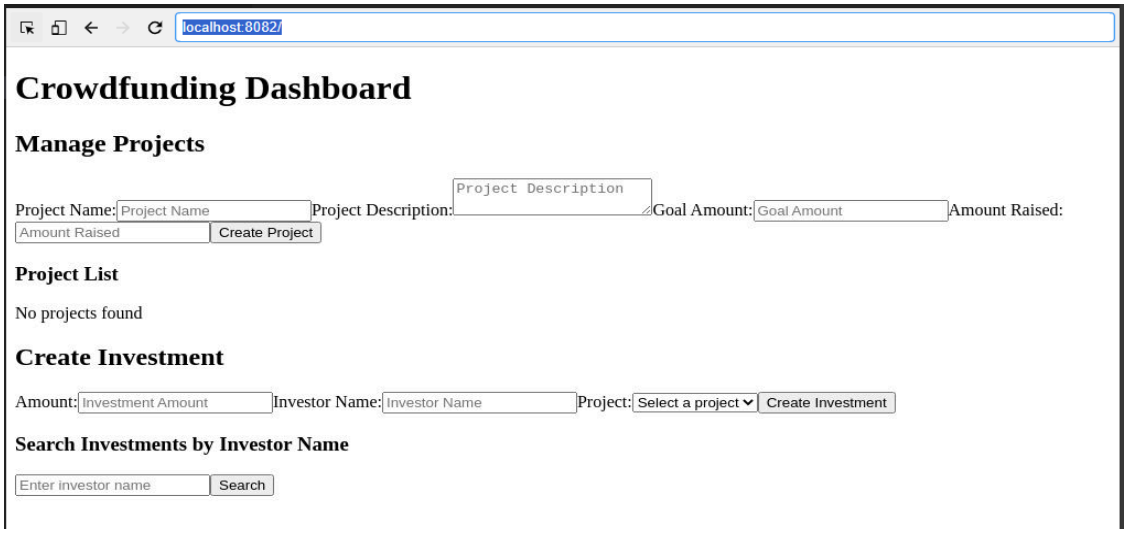
1 PROBLEM STATEMENT

The **Crowdfunding Platform** frontend is a Single Page Application (SPA) built using ReactJs.

2 PROPOSED CROWDFUNDING WIREFRAME

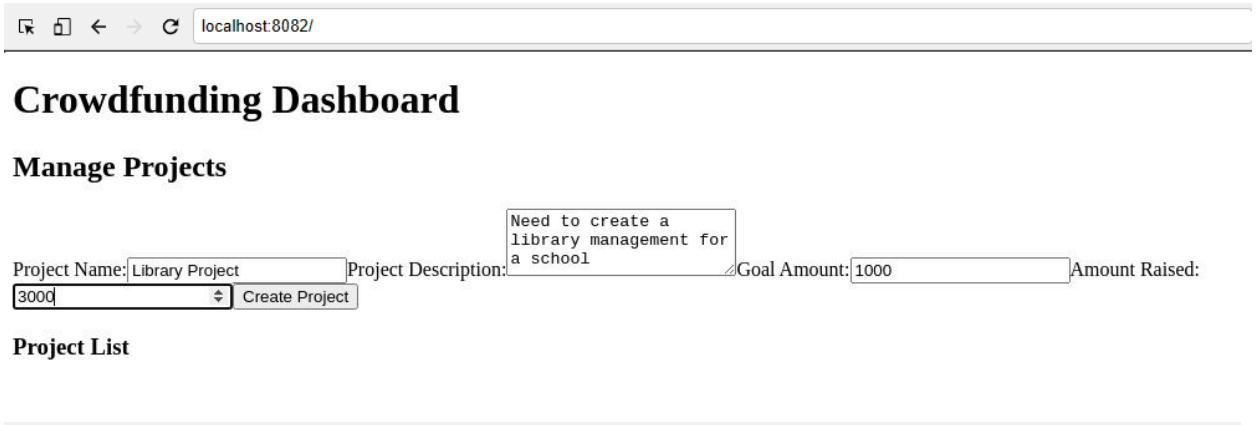
UI needs improvisation and modification as per given use case and to make test cases passed.

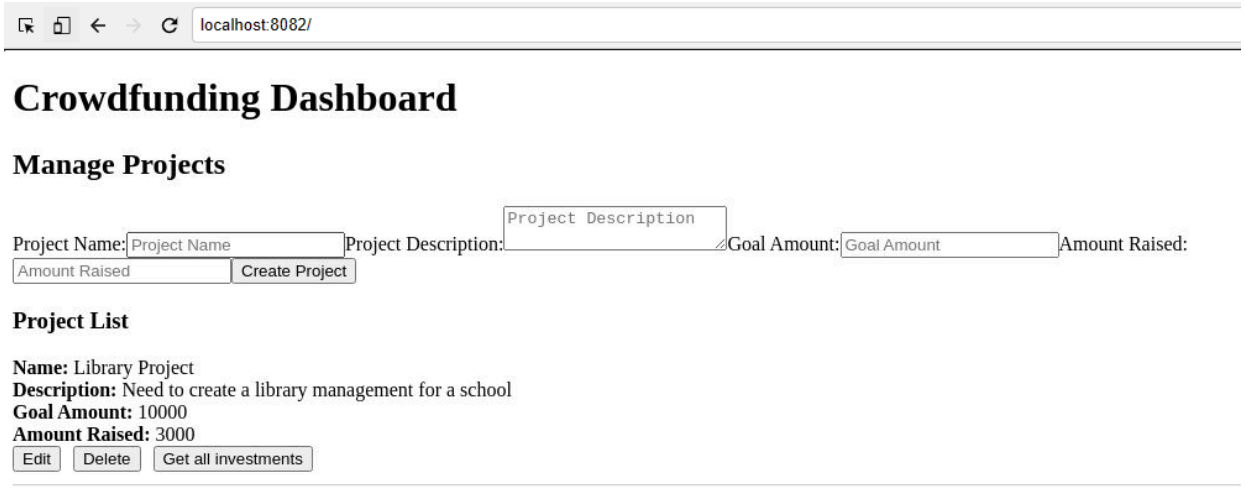
2.1 HOME PAGE



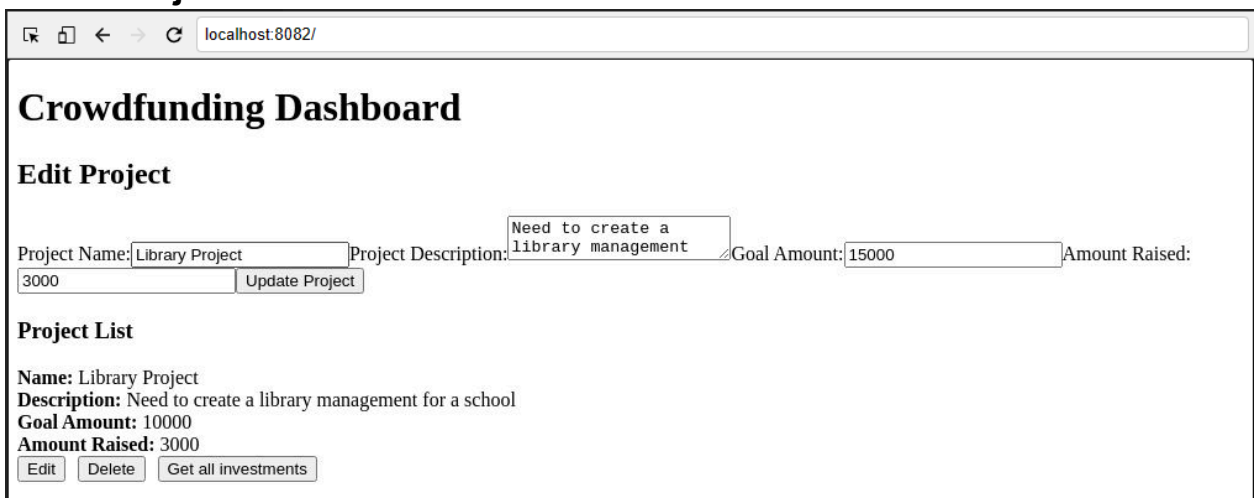
2.2 MANAGE PROJECTS

Create Project



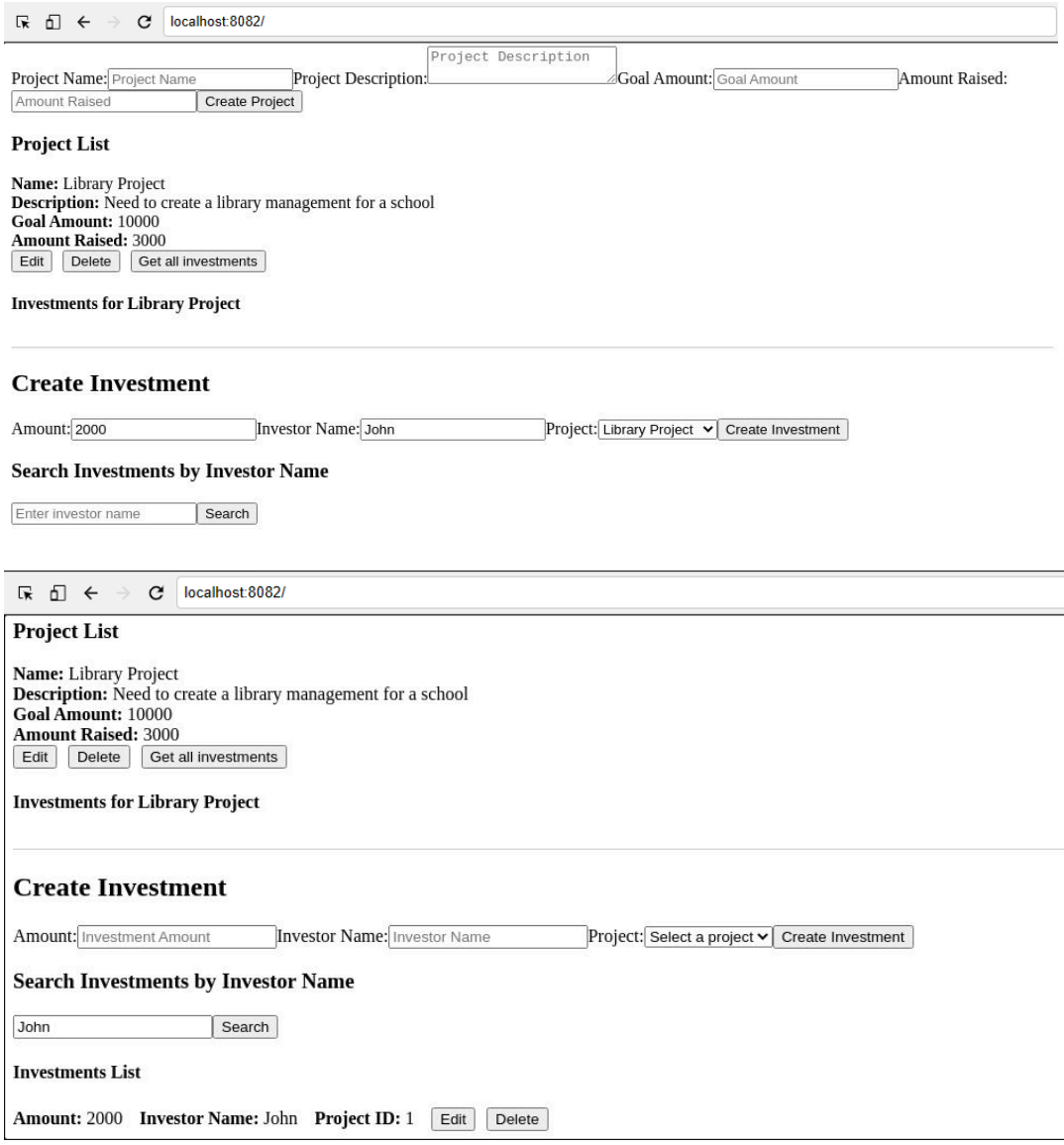


Edit Project



2.3 CREATE INVESTMENT

Create Investment



Edit Investment

Edit Investment

Amount: 2500 Investor Name: John Project: Library Project Update Investment

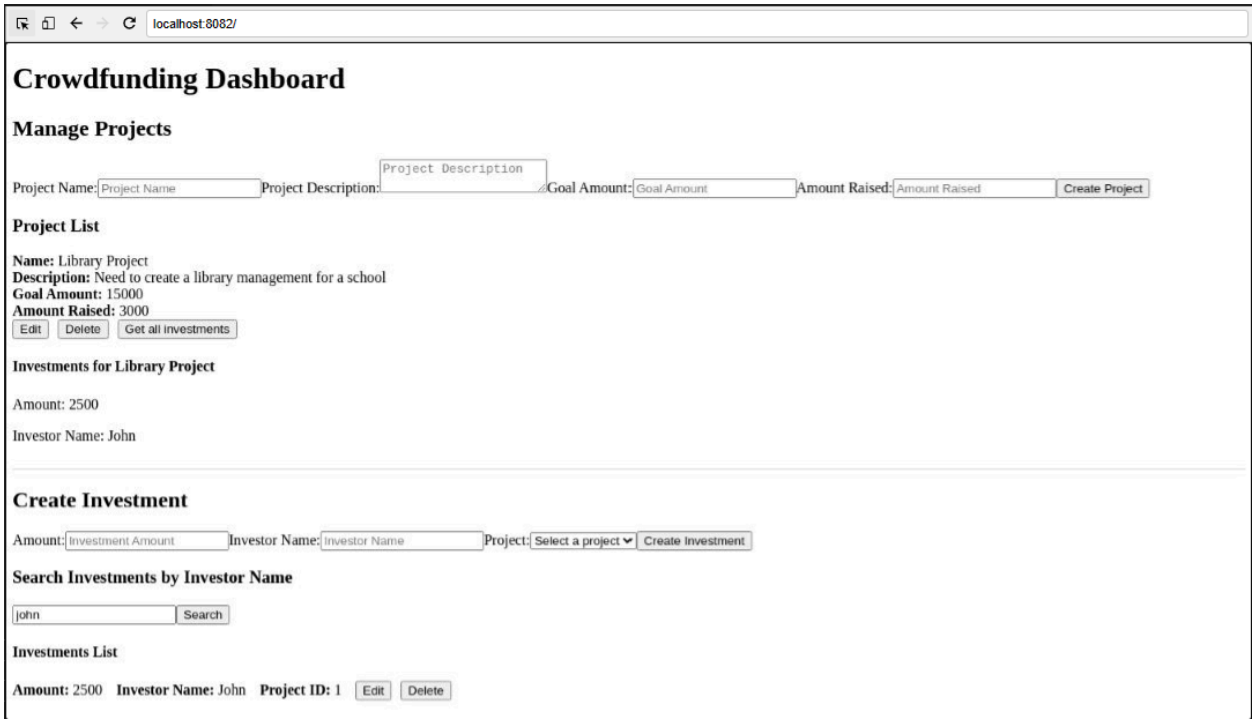
Search Investments by Investor Name

john Search

Investments List

Amount: 2000 **Investor Name:** John **Project ID:** 1 Edit Delete

Overall View



3 BUSINESS-REQUIREMENT:

As an application developer, develop the Crowdfunding Platform 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 “Manage Projects” and “Create Investment” as heading in h2. 2. Should have a “Project List” and “Search Investments by Investor Name” as heading in h1. 3. Should show a list of all projects with “Edit” & “Delete” button in each of the project lists. 4. Should have a button to Get all investments in each project in the project list to show a list of all investments for that particular project. 5. As a user I should be able to furnish the following details at the time of creating/updating a project. 1.1 Project Name 1.2 Project Description 1.3 Goal Amount 1.4 Amount Raised 6. All fields should be required fields to add a project. 7. Should show a list of all investments with an “Edit” & “Delete” button in each of the investments lists. 8. Should have a search option to search all investments by investor name. 9. As a user I should be able to furnish the following details at

		the time of creating/updating an investment. 1.1 Amount 1.2 Investor Name 1.3 Project 10. All fields should be required fields to add an investment.

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.

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:8082 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.**