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

1	Pro	ject Abstract	3
В	ACKEN	ID-JAVA	4
1	Pro	blem Statement	4
2	Ass	umptions, Dependencies, Risks / Constraints	5
	2.1	Investment Constraints	5
	2.2	Project Constraints	5
	2.3	Common Constraints	5
3	Bus	iness Validations	6
4	Res	t Endpoints	6
	4.1	Investment Controller	6
	4.2	Project Controller	7
5	Ten	nplate Code Structure	9
	5.1	Package: com.crowdfunding	9
	5.2	Package: com.crowdfunding.repo	9
	5.3	Package: com.crowdfunding.service	10
	5.4	Package: com.crowdfunding.service.impl	10
	5.5	Package: com.crowdfunding.controller	11
	5.6	Package: com.crowdfunding.dto	11
	5.7	Package: com.crowdfunding.entity	12
	5.8	Package: com.crowdfunding.exception	13
FF	RONTE	ND-REACT SPA	14
1	Pro	blem Statement	14
2	Pro	posed CrowdFunding Wireframe	14
	2.1	Home page	14
	2.2	Manage Projects	15
	2.3	Create Investment	16
3	Bus	iness-Requirement	18
Exec	ution 9	Steps to Follow for Backend	19
Exec	cution 9	Steps to Follow for Frontend	20

Crowdfunding PlatformSystem 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	
	Can create/make investments
	'
	,
5	, , ,
6	Get investments by investor name
Project Module	
Functionalities	Com avente a municat
1	1 7
2	
3	
4	1 7 /
<u>.</u>	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

URL Exposed		Purpose
1. /api/investments/	{investmentId}	
Http Method	GET	Fetches the investment by investmentId
Parameter	Long (investmentId)	·
Return	InvestmentDTO	
2. /api/investments/	project/{projectId}	
Http Method	GET	Fetches the investments by projectId
Parameter 1	Long (projectId)	
Return	List <investmentdto></investmentdto>	
3. /api/investments/	,	
Http Method	POST	
	The investment data	
	to be created should	Creates a new investment
	be received in	creates a new investment
	@RequestBody	
Parameter	InvestmentDTO	
Return	InvestmentDTO	

4. /api/investments/	{investmentId}			
Http Method	PUT			
	The investment data			
	to be updated should			
	be received in	Updates an investment by id		
	@RequestBody			
Parameter 1	Long (investmentId)			
Parameter 2	InvestmentDTO			
Return	InvestmentDTO			
5. /api/investments/	{investmentId}			
Http Method	DELETE	Deletes an investment by id		
Parameter 1	Long (investmentId)			
Return	-			
6. /api/investments/investor/{investorName}				
Http Method	GET	Fetches the investments by investor name		
Parameter	String (investorName)			
Return	List <investmentdto></investmentdto>			

4.2 Project Controller

URL	Exposed	Purpose
1. /api/projects/{p	rojectId}	
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></projectdto>	
3. /api/projects/		

Http Method	The project data to be created should be received in @RequestBody	Creates a new project
Parameter	ProjectDTO	
Return	ProjectDTO	
4. /api/projects/{pro	jectId}	
Http Method	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	This is the Spring Boot starter class of the	Already implemented.
Application	application.	
(Class)		

5.2 PACKAGE: COM.CROWDFUNDING.REPO

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

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.
ProjectService (interface)	 Interface to expose method signatures for project related functionality. Do not modify, add or delete any method.

5.4 PACKAGE: COM.CROWDFUNDING.SERVICE.IMPL

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

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

	class.	
--	--------	--

5.7 PACKAGE: COM. CROWDFUNDING. ENTITY

Class/Interface	Description	Status
Investment (Class)	This class is partially implemented.	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	
Project (Class)	This class is partially implemented.	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	

5.8 PACKAGE: COM.CROWDFUNDING.EXCEPTION

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 	
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 	

FRONTEND-REACT SPA

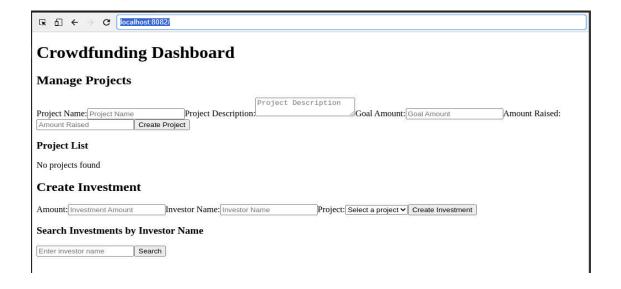
1 PROBLEM STATEMENT

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

R	Ð	←	G	localhost:8082/

Crowdfunding Dashboard

Manage Projects

Project Name: Library Project	Project Description	Need to create a library management for a school	Goal Amount: 1000	Amount Raised:
3000 \$ Cre	eate Project			
Project List				
□ ← → C localhost	:8082/			

Crowdfunding Dashboard

Manage Projects

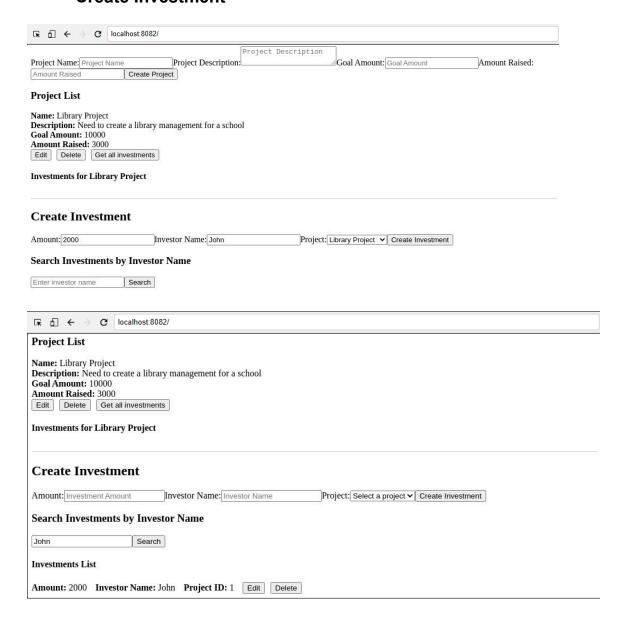
		Project De	escription	
Project Name: Project	Name	Project Description:	Goal Amount: Goal Amount	Amount Raised:
Amount Raised	Create Proj	ect		
Project List				
Name: Library Projec	ct			
Description: Need to	create a library m	anagement for a school		
Goal Amount: 10000)			
Amount Raised: 300	00			
Edit Delete Ge	et all investments			

Edit Project

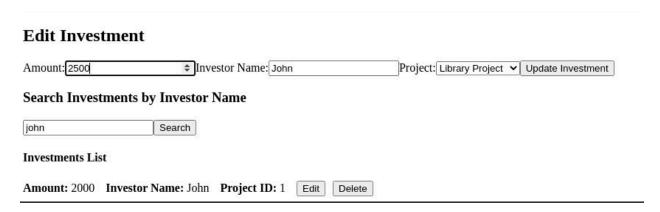
□ ← → C [localhost8082/
Crowdfunding Dashboard
Edit Project
Project Name: Library Project
Project List
Name: Library Project Description: Need to create a library management for a school Goal Amount: 10000 Amount Raised: 3000 Edit Delete Get all investments

2.3 CREATE INVESTMENT

Create Investment



Edit Investment



Overall View



BUSINESS-REQUIREMENT:

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

User	User Story Name	User Story	
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 "Crowdfunding Platform" as heading in h1.	
		Should have "Create Investment" as heading in h2 and "Search Investments by Investor Name" as heading in h3.	
		3. Should have a "Manage Projects" as heading in h2 and "Project List" and as heading in h3.	
		4. Should show a list of all projects with "Edit" & "Delete" buttons in each of the project lists.	
		 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. 	
		6. 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	
		7. All fields should be required fields to add a project.	
		8. Should show a list of all investments with an "Edit" & "Delete" button in each of the investments lists.	
		9. Should have a search option to search all investments by investor name.	

	10. 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
	11. 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.