System Requirements Specification

Index

For

Donation Management System

Version 1.0

TABLE OF CONTENTS

Donation Management APPLICATION
System Requirements Specification

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

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

Donation Management Application is Spring boot RESTful application with MySQL, where NGOs can raise the funds by inviting the donors online and sending the notification about the events and donations.

Following is the requirement specifications:

	Donation Management System Application
Modules	
1	NGO
2	Donor
3	Donation
NGO Module Functionalities	
1	Register a NGO
2	Update the existing NGO details
3	Get the NGO by Id
4	Fetch all registered NGOs
5	Delete an existing NGO
Donor Module Functionalities	
1	Register a Donor
2	Update the existing Donor
3	Get a Donor by Id
4	Fetch all registered Donors
5	Delete an existing Donor
6	Fetch all the Donors registered with a NGO
Donation Module Functionalities	
1	Create a Donation
2	Update the existing Donation details

3	Get the Donation by Id

4	Fetch all Donations
5	Delete an existing Donation
6	Fetch all Donations done by a Donor
7	Fetch all Donations done for a NGO

2 Assumptions, Dependencies, Risks / Constraints

2.1 NGO Constraints:

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

2.2 Donor Constraints

- While deleting the Donor, if donorId does not exist then the operation should throw a custom exception.
- While fetching the Donor details by id, if donorId does not exist then the operation should throw a custom exception.
- While fetching all the Donor details by NGO id, if ngoId does not exist then the operation should throw a custom exception.

2.3 Donations Constraints

- While deleting the Donation, if donationId does not exist then the operation should throw a custom exception.
- While fetching the Donation details by id, if donationId does not exist then the operation should throw a custom exception.
- While fetching all the Donations done by a donor id, if donorId does not exist then the operation should throw a custom exception.
- While fetching all the Donation details by NGO id, if ngoId does not exist then operation should throw custom exception.

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

- NGO name is not null, min 3 and max 100 characters.
- NGO username is not null, min 3 and max 50 characters.
- NGO password is not null, min 3 and max 50 characters.
- NGO address is not null, min 3 and max 100 characters.
- NGO phone number is not null and have min 10 and max 10 digits
- NGO started In is not null, have ‘yyyy-mm-dd’ format and should be past date
- NGO documents is not null, min 3 and max 100 characters.
- Donor name is not null, min 3 and max 100 characters.
- Donor username is not null, min 3 and max 50 characters.
- Donor password is not null, min 3 and max 50 characters.
- Donor email is not null, min 3 and max 100 characters and should be in email format
- Donor phone number is not null and have min 10 and max 10 digits
- Donor address is not null, min 3 and max 100 characters.
- Donation type is not null, min 3 and max 100 characters.
- Donation amount is not null
- Donation date is not null, have ‘yyyy-mm-dd’ format and should be future date
- Donation Request amount is not null
- Donation Request status is not null, min 3 and max 100 characters
- Donation request end date is not null, have ‘yyyy-mm-dd’ format and should be future date

4 Rest Endpoints

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

4.1 NgoController

URL Exposed		Purpose
1. /ngos/register-ngo		Register a NGO
Http Method	POST	
Parameter 1	NgoDto	
Return	NgoDto	
2. /ngos/update-ngo		Update the NGO
Http Method	PUT	
Parameter 1	NgoDto	
Return	NgoDto	
3. /ngos/get/{ngoId}		Fetches the details of NGO by Id
Http Method	GET	
Parameter 1	Long(ngoId)	
Return	NgoDto	
4. /ngos/delete/{ngoId}		Delete the Ngo detail
Http Method	DELETE	
Parameter 1	Long (ngoId)	
Return	Boolean	
5. /ngos/all		Fetch all registered NGOs
Http Method	GET	
Parameter 1	-	
Return	List<NgoDto>	

4.2 DonarController

URL Exposed		Purpose
1. /donars/register-donar		Register a Donar
Http Method	POST	
Parameter 1	DonarDto	
Return	DonarDto	
2. /donars/update-donar		Update the existing donar
Http Method	PUT	
Parameter 1	DonarDto	
Return	DonarDto	
3. /donars/get/{donarId}		Fetches the donar details by id
Http Method	GET	
Parameter 1	Long(donarId)	
Return	DonarDto	

4. /donars/all Http Method: GET Parameter 1: - Return: List<DonarDto>	Fetch the details of all the registered donars

Http Method	GET
Parameter 1	-
Return	List<DonarDto>

5. /donars/delete/{donarId}		Delete the existing Donar
Http Method	DELETE	
Parameter 1	Long (donarId)	
Return	Boolean	
6. /donars/get-by-ngo/{ngoId}		Fetch all the donars registered with the NGO
Http Method	GET	
Parameter 1	Long (ngoId)	
Return	List<DonarDto>	
7. /donations/add-donation		Create a Donation
Http Method	POST	
Parameter 1	DonationDto	
Return	DonationDto	
8. /donations/update-donation		Update the existing Donation details
Http Method	PUT	
Parameter 1	DonationDto	
Return	DonationDto	
9. /donations/delete/{donationId}		Delete an existing Donation
Http Method	DELETE	
Parameter 1	Long (donationId)	
Return	Boolean	
10. /donations/get/{donationId}		Get the donation details by id
Http Method	GET	
Parameter 1	Long (donationId)	
Return	DonationDto	
11. /donations/all		Fetch all the existing donations
Http Method	GET	
Parameter 1	-	
Return	List<DonationDto>	
12. /donations/get-by-donar/{donarId}		Fetch all the donations for a particular donar
Http Method	GET	
Parameter 1	Long(donarId)	
Re$_1$tu$_3$r.n	List<DonationDto>	
13. /donations/get-by-ngo/{ngoId}		Fetch all the donations raised by a particular NGO
Http Method	GET	
Parameter 1	Long(ngoId)	
Re$_1$tu$_4$r.n	List<DonationDto>	

5 Template Code Structure

5.1 PACKAGE: COM.IIHT.TRAINING.NGO

Resources

DonationManagementSystemApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.IIHT.TRAINING.NGO.ENTITY

Resources

Class/Interface	**Description**	**Status**
NgoEntity (class)	o Annotate this class with proper annotation to declare it as an entity class with **ngoId** as primary key. o Map this class with **ngo_details** table. **o** Generate the **ngoId** using **IDENTITY** strategy	Partially implemented.
DonarEntity(class)	o This is class is partially implemented. o Annotate this class with proper annotation to declare it as an entity class with **donarId** as primary key. **o** Map this class with **donar** table. o Generate the **donarId** using the **IDENTITY** strategy	Partially implemented.
Donation(class)	o This class is partially implemented. o Annotate this class with proper annotation to declare it as an entity class with **donationId** as primary key. o Map this class with **donation** table. **o** Generate the **donationId** using the **IDENTITY** strategy	Partially implemented.

DonationRequestEntity (class)	o This class is partially implemented. o Annotate this class with proper annotation to	Partially implemented

	declare it as an entity class with **requestId** as primary key. o Map this class with **donation_request** table. **o** Generate the **requestId** using the **IDENTITY** strategy ○	

5.3 PACKAGE: COM.IIHT.TRAINING.NGO.DTO

Resources

Class/Interface	Description	Status
NgoDto (class)	Use appropriate annotations from the **Java Bean Validation API** for validating attributes of this class. (Refer **Business Validation** section for validation rules).	Partially implemented.
DonarDto (class)	Use appropriate annotations from the **Java Bean Validation API** for validating attributes of this class. (Refer **Business Validation** section for validation rules).	Partially implemented.
DonationDto (class)	Use appropriate annotations from the **Java Bean Validation API** for validating attributes of this class. (Refer **Business Validation** section for validation rules).	Partially implemented.
DonationRequestDto (class)	Use appropriate annotations from the **Java Bean Validation API** for validating attributes of this class.	Partially implemented

	(Refer **Business Validation** section for validation rules).	

5.4 Package: com.iiht.training.ngo.model.exception

Resources

Class/Interface	Description	Status
ExceptionResponse (class)	Object of this class is supposed to be returned in case of exception through exception handlers	Already implemented.

5.5 Package: com.iiht.training.ngo.repository

Resources

Class/Interface	Description	Status
NgoRepository (interface)	1. Repository interface exposing CRUD functionality for **NGO** Entity. 2. You can go ahead and add any custom methods as per requirements	Partially implemented
DonarRepository (interface)	1. Repository interface exposing CRUD functionality for **DonarEntity** Entity. 2. You can go ahead and add any custom methods as per requirements	Partially implemented
DonationRepository (interface)	1. Repository interface exposing CRUD functionality for **Donation** Entity.	Partially implemented

	2. You can go ahead and add any custom methods as per requirements	
DonationRequestRepository (interface)	1. Repository interface exposing CRUD functionality for **DonationRequest** Entity. 2. You can go ahead and add any custom methods as per requirements	Partially implemented

5.6 Package: com.iiht.training.ngo.service

Resources

Class/Interface	**Description**	**Status**
NgoService (interface)	Interface to expose method signatures for NGO related functionality. Do not modify, add or delete any method	Already implemented.
DonarService (interface)	Interface to expose method signatures for Donar related functionality. Do not modify, add or delete any method	Already implemented.
DonationService (interface)	Interface to expose method signatures for Donations related functionality. Do not modify, add or delete any method	Already implemented.

DonationRequestService (interface)	Interface to expose method signatures for Donation request related functionality. Do not modify, add or delete any method	Already implemented

5.7 PACKAGE: COM.IIHT.TRAINING.NGO.SERVICE.IMPL

Resources

Class/Interface	**Description**	**Status**
NgoServiceImpl (class)	● Implements **NgoService**. Contains template method implementation. ● Need to provide implementation for NGO related functionalities ● Add required repository dependency ● Do not modify, add or delete any method signature	To be implemented.
DonarServiceImpl (class)	● Implements **DonarService**. Contains template method implementation. ● Need to provide implementation for Donar related functionalities ● Add required repository dependency ● Do not modify, add or delete any method signature	To be implemented.
DonationServiceImpl (class)	● Implements **DonationService**. Contains template method implementation.	To be implemented.

	• Need to provide implementation for donations related functionalities • Add required repository dependency • Do not modify, add or delete any method signature	
DonationRequestServiceImpl (class)	• Implements **DonationRequestService**. Contains template method implementation. • Need to provide implementation for donation request and notifications related functionalities • Add required repository dependency • Do not modify, add or delete any method signature	To be implemented

5.8 PACKAGE: COM.IIHT.TRAINING.NGO.EXCEPTION

Resources

Class/Interface	**Description**	**Status**
GlobalHandler (class)	• RestControllerAdvice Class for defining global exception handlers.	Partially implemented.

	● Contains Exception Handler for **InvalidDataException** class. ● Use this as a reference for creating exception handler for other custom exception classes	

Class/Interface	**Description**	**Status**
NgoNotFoundException (Class)	● Custom Exception to be thrown when trying to fetch or delete the NGO info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Already created.
DonarNotFoundException (Class)	● Custom Exception to be thrown when trying to fetch or delete Donar info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Already created.
DonationNotFoundException (Class)	● Custom Exception to be thrown when trying to fetch or delete a donation info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Already created.

5.9 PACKAGE: COM.IIHT.TRAINING.NGO.CONTROLLER

Resources

Class/Interface	Description	Status
NgoController (Class)	● Controller class to expose all rest-endpoints for NGO and donation request related activities. ● May also contain local exception handler methods	To be implemented
DonarController (Class)	● Controller class to expose all rest-endpoints for Donar and Donations related activities. ● May also contain local exception handler methods	To be implemented

6 CONSIDERATIONS

A. There is no roles in this application
B. You can perform the following 3 possible actions

NGO
Donor
Donation

FRONTEND-ANGULAR SPA

1 PROBLEM STATEMENT

Donation management is SPA (Single Page Application) for registering different NGO under some Donation campaign along with Donor information. It performs all CRUD operations for all 3 modules.

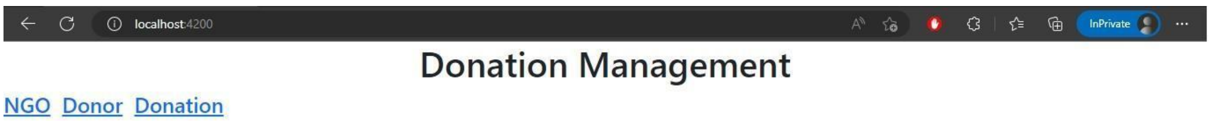
The core modules of Donation management app are:

1. Welcome Page
2. Ngo homepage
3. Donor homepage
4. Donation homepage

2 PROPOSED DONATION MANAGEMENT WIREFRAME

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

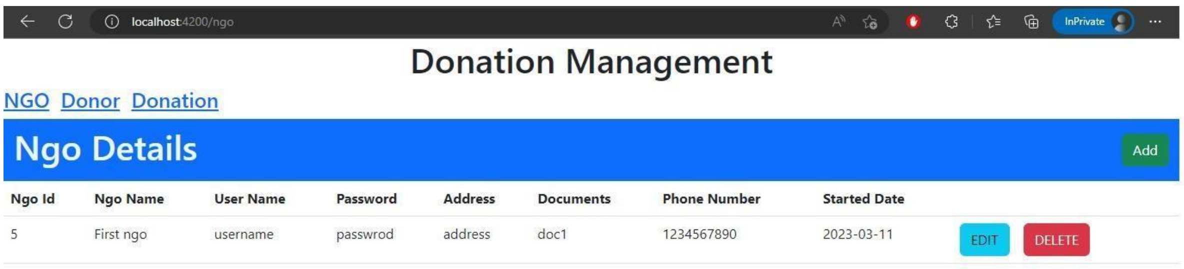
2.1 WELCOME PAGE



Donation Management

NGO Donor Donation

2.2 NGO HOMEPAGE



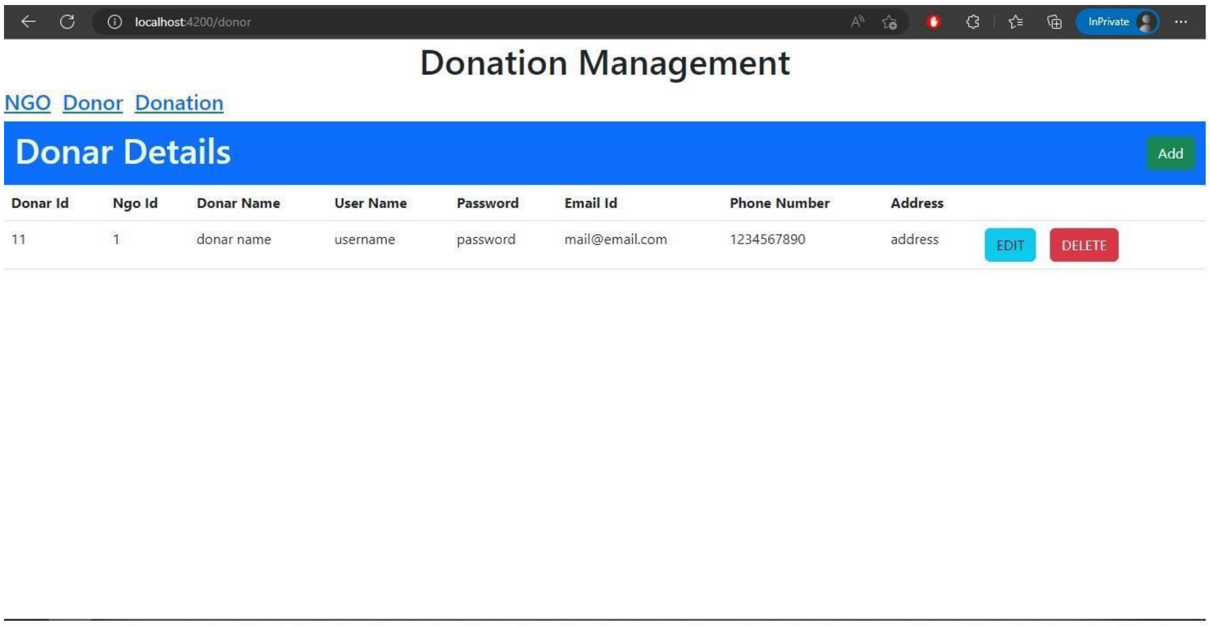
Donation Management

NGO Donor Donation

Ngo Details

Ngo Id	Ngo Name	User Name	Password	Address	Documents	Phone Number	Started Date	
5	First ngo	username	passwrod	address	doc1	1234567890	2023-03-11	EDIT DELETE

2.3 Donor Homepage



localhost:4200/donor

Donation Management

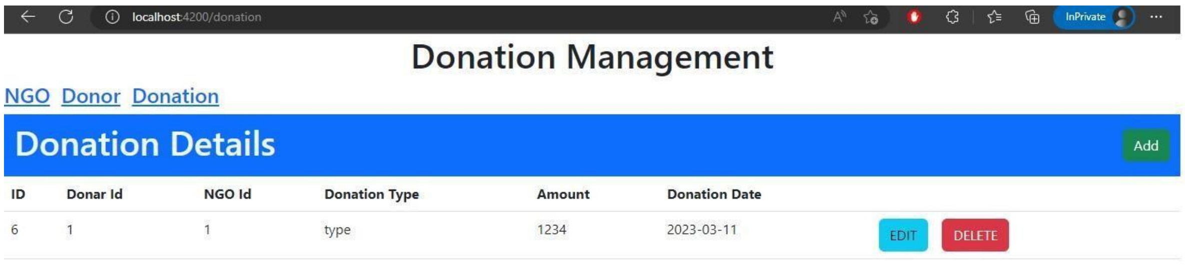
NGO Donor Donation

Donar Details

Add

Donar Id	Ngo Id	Donar Name	User Name	Password	Email Id	Phone Number	Address	
11	1	donar name	username	password	mail@email.com	1234567890	address	EDIT DELETE

2.4 Donation Homepage



localhost:4200/donation

Donation Management

NGO Donor Donation

Donation Details

Add

ID	Donar Id	NGO Id	Donation Type	Amount	Donation Date	
6	1	1	type	1234	2023-03-11	EDIT DELETE

3 BUSINESS-REQUIREMENT:

As an application developer, develop the Donation management App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Welcome Page	As a user I should be able to visit the welcome page as default page. Acceptance criteria: 1. User can click any links shown at homepage.
US_02	Ngo Homepage	As a user I should be able to see Ngo page and perform all CRUD operations: Acceptance criteria: 1. As a user I should be able to furnish following details at the time of creating an ngo. 1.1 Ngo Name 1.2 User Name 1.3 Password 1.4 Address 1.5 Documents 1.6 Phone Number 1.7 Started In 2. Save button should be disabled until all fields are validated. 3. All details fields must be mandatory. If any constraint is not satisfied, a validation message must be shown.
US_03	Donor Homepage	As a user I should be able to see donor page and perform all CRUD operations: Acceptance criteria: 1. As a user I should be able to furnish following details at the time of creating an donor. 1.2 Ngo Id 1.3 Donor Name 1.4 User Name

		1.5 Password 1.6 Email ID 1.7 Phone Number 1.8 Address 2. Save button should be disabled until all fields are validated. 3. All details fields must be mandatory. If any constraint is not satisfied, a validation message must be shown.
US_04	Donation Homepage	As a user I should be able to see donation page and perform all CRUD operations: Acceptance criteria: 1. As a user I should be able to furnish following details at the time of creating an donation. 1.9 Donor Id 1.10 Ngo ID 1.11 Donation Type 1.12 Amount 1.13 Donation Date Save button should be disabled until all fields are validated. 1. All details fields must be mandatory. If any constraint is not satisfied, a validation message must be shown

4 CONSTRAINTS

1. On the page load, input focus must come to the first name input field.
2. You should be able to press the “TAB” key and “SHIFT + TAB” to navigate from top field to bottom field and vice-versa.

7 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.**

8 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:4200 to open project in browser -> takes 2 to 3 min**
   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.**