System Requirements Specification Index

For

Donation Management System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

Donation Management APPLICATION
System Requirements Specification

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
4	Donation Request
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

Donation Request Module Functionalities	
1	Create a Donation Request
2	Get the Donation Notification by the NGO
3	Get all the Donation request sent to a Donor

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.

2.4 Donation Request Constraints

- While fetching the all the Donation request done by NGO, if ngoId does not exist then operation should throw custom exception.
- While fetching all the Donation requests sent to a donor, if donorId does not exist then the operation should throw a 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 Http Method: POST Parameter 1: NgoDto Return: NgoDto	Register a NGO
2. /ngos/update-ngo Http Method: PUT Parameter 1: NgoDto Return: NgoDto	Update the NGO
3. /ngos/get/{ngoId} Http Method: GET Parameter 1: Long(ngoId) Return: NgoDto	Fetches the details of NGO by Id
4. /ngos/delete/{ngoId} Http Method: DELETE Parameter 1: Long (ngoId) Return: Boolean	Delete the Ngo detail
5. /ngos/all Http Method: GET Parameter 1: - Return: List<NgoDto>	Fetch all registered NGOs
6. /ngos/create-donation-request Http Method: POST Parameter 1: DonationRequestDto Return: DonationRequestDto	Create a Donation request
7. /ngos/donation-request-by-ngo/{ngoId} Http Method: GET Parameter 1: Long(ngoId) Return: List<DonationRequestDto>	Get all the donation request by the NGO
8. /ngos/ donation-request-by-donar/{donarId} Http Method: GET Parameter 1: Long(donarId) Return: List<DonationRequestDto>	Get all the donation requests for a donar

4.2 DonarController

URL Exposed	Purpose
1. /donars/register-donar Http Method: POST Parameter 1: DonarDto Return: DonarDto	Register a Donar
2. /donars/update-donar Http Method: PUT Parameter 1: DonarDto Return: DonarDto	Update the existing donar
3. /donars/get/{donarId} Http Method: GET Parameter 1: Long(donarId) Return: DonarDto	Fetches the donar details by id
4. /donars/all Http Method: GET Parameter 1: - Return: List<DonarDto>	Fetch the details of all the registered donars
5. /donars/delete/{donarId} Http Method: DELETE Parameter 1: Long (donarId) Return: Boolean	Delete the existing Donar
6. /donars/get-by-ngo/{ngoId} Http Method: GET Parameter 1: Long (ngoId) Return: List<DonarDto>	Fetch all the donars registered with the NGO
7. /donations/add-donation Http Method: POST Parameter 1: DonationDto Return: DonationDto	Create a Donation
8. /donations/update-donation Http Method: PUT Parameter 1: DonationDto Return: DonationDto	Update the existing Donation details

9. /donations/delete/{donationId} Http Method: DELETE Parameter 1: Long (donationId) Return: Boolean	Delete an existing Donation
10. /donations/get/{donationId} Http Method: GET Parameter 1: Long (donationId) Return: DonationDto	Get the donation details by id
11. /donations/all Http Method: GET Parameter 1: - Return: List<DonationDto>	Fetch all the existing donations
12. /donations/get-by-donar/{donarId} Http Method: GET Parameter 1: Long(donarId) Return: List<DonationDto> 13.	Fetch all the donations for a particular donar
13. /donations/get-by-ngo/{ngoId} Http Method: GET Parameter 1: Long(ngoId) Return: List<DonationDto> 14.	Fetch all the donations raised by a particular NGO

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	Partially implemented.

	declare it as an entity class with **ngoId** as primary key. o Map this class with the ngo_details table. o Generate the **ngoId** using **IDENTITY** strategy	
DonarEntity(class)	o This 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 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 o	Partially implemented

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

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

	2. You can go ahead and add any custom methods as per requirements	

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. • Need to provide implementation for donations related functionalities • Add required repository dependency • Do not modify, add or delete any method signature	To be implemented.

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

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
Donation Request

7 Execution Steps to Follow

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. To build your project use command:
   mvn clean package -Dmaven.test.skip
4. To launch your application, move into the target folder (cd target). Run the following command to run the application:
   java -jar donation-management-system-0.0.1-SNAPSHOT.jar
5. This editor Auto Saves the code.
6. 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.
7. 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.
8. 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.
9. This is a web-based application, to run the application 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
10. 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.**