System Requirements Specification Index

For

Donation Management System

Version 4.0

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

Donation Management APPLICATION

System Requirements Specification

1.Business-Requirement:

1.1 Problem Statement:

Donation Management Application is .Net Core 3.1 RESTful application with InMemory Database, where NGOs can raise the funds by inviting the donors online and sending the notification about the events and donations.

1.2 Following is the requirement specification:

	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 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 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.

2.5 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 model 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

3.1 NGO Entity:

- NGO name(string) is not null, min 3 and max 100 characters.
- NGO username(string) is not null, min 3 and max 50 characters.
- NGO password(string) is not null, min 3 and max 50 characters.
- NGO address(string) is not null, min 3 and max 100 characters.
- NGO phone number(string) is not null and have min 10 and max 10 digits
- NGO started In(datetime) is not null, have 'yyyy-mm-dd' format and should be past date
- NGO documents(string) are not null, min 3 and max 100 characters.

3.2 Donor Entity:

- Donor name(string) is not null, min 3 and max 100 characters.
- Donor username(string) is not null, min 3 and max 50 characters.
- Donor password(string) is not null, min 3 and max 50 characters.
- Donor email(string) is not null, min 3 and max 100 characters and should be in email format
- Donor phone number(string) is not null and have min 10 and max 10 digits
- Donor address(string) is not null, min 3 and max 100 characters.

3.3 Donation Entity:

- Donation type(string) is not null, min 3 and max 100 characters.
- Donation amount(decimal) is not null
- Donation date(datetime) is not null, has 'yyyy-mm-dd' format and should be a future date.

3.1 Donation Request Entity:

- Donation Request amount(decimal) is not null
- Donation Request status(string) is not null, min 3 and max 100 characters
- Donation request end date(datetime) 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

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/ngos/register-ngo<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameters</td><td>NgoDetails ngoDetails,string password</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Register a NGO</td></tr>
<tr><td>/ngos/update-ngo<table><tr><td>Http Method</td><td>PUT</td></tr><tr><td>Parameter 1</td><td>RegisterNgoViewModel model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Update the NGO</td></tr>
<tr><td>/ngos/get/{ngoId}<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long(ngoId)</td></tr><tr><td>Return</td><td><NgoDetails></td></tr></table></td><td>Fetches the details of NGO by Id</td></tr>
<tr><td>/ngos/delete/{ngoId}<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>Long (ngoId)</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Delete the Ngo detail</td></tr>
<tr><td>/ngos/all<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<NgoDetails>></td></tr></table></td><td>Fetch all registered NGOs</td></tr>
<tr><td>/ngos/create-donation-request<table><tr><td>Http Method</td><td>POST</td></tr></table></td><td>Create a Donation request</td></tr>
</table>

Parameter 1	DonationRequest donationRequest
Return	HTTP Response StatusCode

/ngos/donation-request-by-ngo/{ngoId}	Get all the donation request by the NGO
Http Method	GET
Parameter 1	Long(ngoId)
Return	<IEnumerable<DonationRequest>>

/ngos/ donation-request-by-donor/{donarId}	Get all the donation requests for a donor
Http Method	GET
Parameter 1	Long(donarId)
Return	<IEnumerable<DonationRequest>>

4.2 DonarController

URL Exposed	Purpose
/donors/register-donar	Register a Donor
Http Method	POST
Parameters	Donor donor, string password
Return	HTTP Response StatusCode
/donors/update-donar	Update the existing donor
Http Method	PUT
Parameter 1	RegisterDonorViewModel model
Return	HTTP Response StatusCode
/donors/get/{donarId}	Fetches the donor details by id
Http Method	GET
Parameter 1	Long(donarId)
Return	<Donor>
/donors/all	Fetch the details of all the registered donors
Http Method	GET
Parameter 1	-

Return	List<DonarDto>

/donors/delete/{donarId}	Delete the existing Donor
Http Method	DELETE
Parameter 1	Long (donarId)
Return	HTTP Response StatusCode

/donors/get-by-ngo/{ngoId}	Fetch all the donors registered with the NGO
Http Method	GET
Parameter 1	Long (ngoId)
Return	List<DonarDto>

/donations/add-donation	Create a Donation
Http Method	POST
Parameter 1	Donation donation
Return	HTTP Response StatusCode

/donations/update-donation	Update the existing Donation details
Http Method	PUT
Parameter 1	RegisterDonationViewModel model
Return	HTTP Response StatusCode

/donations/delete/{donationId}	Delete an existing Donation
Http Method	DELETE
Parameter 1	Long (donationId)
Return	HTTP Response StatusCode

/donations/get/{donationId}	Get the donation details by id
Http Method	GET
Parameter 1	Long (donationId)
Return	<Donation>

/donations/all	Fetch all the existing donations
Http Method	GET
Parameter 1	-
Return	<IEnumerable<Donation>>

<table>
<tr>
<td>/donations/get-by-donor/{donarId}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long(donarId)</td></tr><tr><td>Return</td><td><IEnumerable<Donation>></td></tr></table></td>
<td>Fetch all the donations for a particular donor</td>
</tr>
<tr>
<td>/donations/get-by-ngo/{ngoId}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long(ngoId)</td></tr><tr><td>Return</td><td><IEnumerable<Donation>></td></tr></table></td>
<td>Fetch all the donations raised by a particular NGO</td>
</tr>
</table>

5. Template Code Structure

5.1 Package: Donation-Management

Resources

Names	Resource	Remarks
Package Structure		
controller	Ngo Controller Donor Controller	These controllers handle all application Function, Create/Update/Edit show information
Startup.cs	Startup CS file	Contain all Services settings and InMemory Db Configuration.
Properties	launchSettings.json file	All URL Setting for API

5.2 Package: Donation-Management.BusinessLayer

Resources

Names	Resource	Remarks
Package Structure		
Interface	INgoServices interface IDonorServices interface IDonationServices interface IDonationRequestServices interface	Inside all these interface files contains all business validation logic functions..
Service	Ngo Services CS file Donor Services CS file Donation Services CS file DonationRequest Services CS file	Using this all class we are calling the Repository method and use it in the program and on the controller.
Repository	INgoRepository Ngo Repository IDonorRepository DonorRepository IDonationRepository Donation Repository IDonationRequestRepository DonationRequestRepository CS file and interface.	All these interfaces and class files contain all CRUD operation code for InMemory Database.
ViewModels	RegisterNgoViewModel, RegisterDonorViewModel,	Contain all view Domain entities for show and bind data.

	RegisterDonationViewModel, RegisterDonationRequestViewModel,	

5.3 Package: Donation-Management.DataLayer

Resources

Names	Resource	Remarks
Package Structure		
DataLayer	NgoDbContext cs file	All database Connection and collection setting class

5.4 Package: Donation-Management.Entities

Resources

Names	Resource	Remarks
Package Structure		
Entities	NgoDetails,Donor,Donation, DonationRequest CS file	All Entities/Domain attribute are used for pass the data in controller

5.5 Package: Donation-Management.Tests

Resources

The Donation-Management.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

6. Execution Steps to Follow

- **All actions like build, compile, running application, running test cases will be through Command Terminal.**

- **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) Terminal → New Terminal.**

- **On command prompt, cd into your project folder (cd <Your-Project-folder>).**

- **To build your project use command:**
  (Project_directory/DonationManagement / dotnet build)

- **To launch your application, Run the following command to run the application:**
  (Project_directory/DonationManagement / dotnet run)

- **This editor Auto Saves the code.**

- **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.**

- **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.**

- **To run the test cases in CMD, Run the following command to test the application:**
  dotnet test --logger "console;verbosity=detailed"
  (You can run this command multiple times to identify the test case status,
  and refactor code to make maximum test cases passed before final submission)

- **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.**

- **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.**

- **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.**
