System Requirements Specification Index

For

Django Rest-API
Donation Management System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1. Project Abstract

Donation Management Application is Django RESTful application withSQLite database, 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 an 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 an 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 an NGO

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

2.2 Donor Constraints

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

2.3 Donations Constraints

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

- NGO name is max 100 characters.
- NGO username is max 50 characters.
- NGO password is max 50 characters.
- NGO address is max 100 characters.
- NGO phone number is max 10 digits.
- NGO started in is should have ‘yyyy-mm-dd’ format and should be past date.
- NGO documents is max 100 characters.
- Donor name is max 100 characters.
- Donor username is max 50 characters.
- Donor password is max 50 characters.
- Donor email is max 100 characters and should be in email format
- Donor phone number is max 10 digits
- Donor address is max 100 characters.
- Donation type is max 100 characters.
- Donation date should have ‘yyyy-mm-dd’ format and should be future date.
- Donation request end date should 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

Class Name	Method Name	Purpose Of Method
NGOView	get(self,request,pk=None,format=None)	Get the NGO by Id and Fetch all registered NGOs
	post(self, request,format=None)	Register an NGO
	patch(self,request,pk,format=None)	Update the existing NGO details
	delete(self,request,pk,format=None)	Delete an existing NGO
DonorView	get(self,request,pk=None,format=None)	Get a Donor by Id and Fetch all registered Donors
	post(self, request,format=None)	Register a Donor
	patch(self,request,pk,format=None)	Update the existing Donor
	delete(self,request,pk,format=None)	Delete an existing Donor
DonorWithNGOView	get(self,request,pk=None,format=None)	Fetch all the Donors registered with an NGO
DonationView	get(self,request,pk=None,format=None)	Get the Donation by Id and Fetch all Donations
	post(self, request,format=None)	Create a Donation
	patch(self,request,pk,format=None)	Update the existing Donation details
	delete(self,request,pk,format=None)	Delete an existing Donation
DonationByDonarView	get(self,request,pk=None,format=None)	Fetch all Donations done by a Donor
DonationForNGOView	get(self,request,pk=None,format=None)	Fetch all Donations done for an NGO
DonationRequestView	get(self,request,pk=None,format=None)	Get all the Donation request sent to a Donor
	post(self, request,format=None)	Create a Donation Request
DonationRequestByNGOView	get(self,request,pk=None,format=None)	Get the Donation Notification by the NGO

Resources (Models)

Class	Description	Status
NGOModel	o A model class for NGO. o It will map to the NGOModel table.	Already implemented.
DonorModel	o A model class for Donor. o It will map to the DonorModeltable.	Already implemented.
DonationModel	o A model class for Donation. o It will map to the DonationModel table.	Already implemented.
DonationRequestModel	o A model class for DonationRequest. o It will map to the DonationRequestModeltable.	Already implemented.

Resources (Serializers)

Class	Description	Status
NGOSerializer	A serializer for NGOModel	Already implemented
DonorSerializer	A serializer for DonorModel	Already implemented
DonationSerializer	A serializer for DonationModel	Already implemented
DonationRequest Serializer	A serializer for DonationRequest Model	Already implemented

Resources (Views)

Class	Description	Status
NGOView	A class for the get, post, patch delete functionalities on **NGOModel** model**.**	To be implemented

DonorView	A class for the get, post, patch delete functionalities on **DonorModel** model**.**	To be implemented
DonorWithNGOView	A class for the get functionality on **DonorModel** model**.**	To be implemented
DonationView	A class for the get, post, patch delete functionalities on **DonationModel** model**.**	To be implemented
DonationByDonarView	A class for the get functionality on **DonationModel** model**.**	To be implemented
DonationForNGOView	A class for the get functionality on **DonationModel** model**.**	To be implemented
DonationRequestView	A class for the get,post functionalities on **DonationRequestModel** model**.**	To be implemented
DonationRequestByNGOView	A class for the get functionalitiy on **DonationRequestModel** model**.**	To be implemented

Resources (Exceptions)

Class	Description	Status
IdNotAvailable	Object of this exception class is supposed to be returned in case specified id is not available.	Already implemented.

InvalidData	Object of this exception class is supposed to be returned in case received data is invalid.	Already 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. The editor Auto Saves the code.
4. If you want to exit(logout) and to 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.
5. 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.
6. 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.
7. 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.

8. Install ‘djangorestframework’ module before running the code. For this use the following command.

   pip install djangorestframework

9. Use the following command to run the server

   python3 manage.py runserver

10. Mandatory: Before final submission run the following commands to execute testcases

    python3 manage.py test donationapp.test.test_functional

    python3 manage.py test donationapp.test.test_exceptional

    python3 manage.py test donationapp.test.test_boundary

11. To test rest end points

    Click on ‘Thunder Client’ or use Ctrl+Shift+R ->Click on ‘New Request’ (at left side of IDE)

12. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.

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.

------- * --------