IIHT

Time To Complete: 10 to 12 hr

E-Donation System

Contents

[1 Problem Statement 2](#_Toc76387085)

[2 Proposed E-Donation System Wireframe 2](#_Toc76387086)

[3 Tool Chain 3](#_Toc76387087)

[4 Business Requirements: 4](#_Toc76387088)

[5 Proposed Rest Endpoints to be exposed 7](#_Toc76387089)

[5.1 Rest APIs: 7](#_Toc76387090)

[6 Rubrics/Expected Deliverables 7](#_Toc76387091)

[7 Implementation/Functional Requirements 8](#_Toc76387092)

[7.1 Product and Frameworks: 8](#_Toc76387093)

[7.2 Governance and Tooling: 9](#_Toc76387094)

[7.3 Code Quality/Optimizations 9](#_Toc76387095)

[8 Platform 9](#_Toc76387096)

[8.1 Cloud Specific Design 9](#_Toc76387097)

[8.2 Design Specification - 1 9](#_Toc76387098)

[8.3 Design specification - 2 9](#_Toc76387099)

[8.4 Design specification – 3 10](#_Toc76387100)

[9 Methodology 10](#_Toc76387101)

[9.1 Agile 10](#_Toc76387102)

# Problem Statement

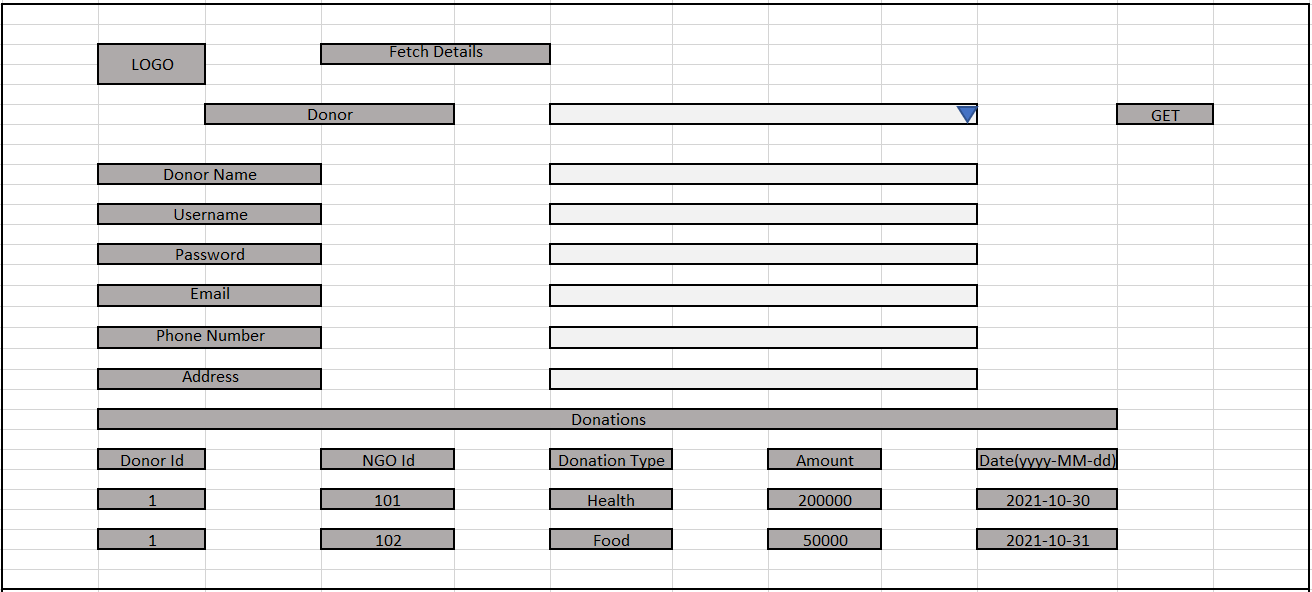
**E-Donation System** is Restful Application. The Main purpose of **E-Donation System** is to online donate the surplus income to the NGOs working for the Human Welfare. The core modules of E-Donation System app are:

* Register a Donor with the NGO
* Donor can donate for the NGO
* NGO can enlist all the Donors registered
* Donor can enlist all the donations given for a NGO.

The scope includes developing the application using tool chain mentioned below.

# Proposed E-Donation Application Wireframe

Below is the wireframe for reference.



# Tool Chain

|  |  |  |
| --- | --- | --- |
| Competency | Skill | Skill Detail |
| Engineering Mindset | Code Quality |  |
|  |  |  |
| Programming Languages | Application Language | Java |
| Products & Frameworks | Presentation | Angular/React |
|  |  | Javascript/Typescript |
|  |  | Bootstrap |
|  | Compute & Integration | Spring Boot |
|  |  |  |
|  | Governance & Tooling | JUnit |
|  |  | Mockito |
|  |  | Jasmine |
|  |  | JaCoCo |
|  |  | SonarQube |

# Business Requirements:

Below are the user stories for the given problem statement

|  |  |  |  |
| --- | --- | --- | --- |
| **User**  **Story #** | **User Story Name** | **User Story** | **Development** |
| US\_01 | Donor Functionalities | As a Donor I can register with the NGO  *While registering with the NGO, following information is required.*   * NGO Id * Donor Name * Username * Password * Email * Phone Number * Address   It also requires NGO information to register NGO as follows:  NGO Information   * NGO Name * Username * Password * Address * Phone Number * Established Date   Constraints:   1. Established date should be Past date, else throw a custom exception in such case   Validations:   1. Donor Name is not null, min 5 and max 30 characters. 2. Username for both NGO and Donor is not null, min 5 and max 30 characters. 3. Password for both NGO and donor is not null, min 3 and max 25 characters. 4. email is not null, and it should be valid email pattern, containing a single @. 5. Phone Number is not null, min 10 and max 10 character and all must be numeric. 6. Address for both NGO and Donor is not null, min 5 and max 100 characters | Only API to be developed |
| US\_02 | NGO Functionalities | As an NGO I can fetch all the Donors registered  Constraints:   1. If the NGO id is not valid, it must throw a custom exception. | Only API to be developed |
| US\_03 | Donation Functionalities | As a Donor I can donate to a particular NGO  *While donating for a NGO, following information is required for Donation.*   * Donor Id * NGO Id * Donation Type * Amount * Donation Date   Precondition:   1. Donor Id and NGO Id must be of an existing (Can get from Database)   Constraints:   1. If the Donation date is Past or Future Date then, throw a custom exception 2. If the Donation Amount is Zero or Negative, throw a custom Exception   Validations:   1. Donation Type is not null, min 5 and max 100 characters. 2. Donation Amount is not null 3. Donation date is the current date and the format should me ‘yyyy-MM-dd’ | Only API to be developed |
| US\_04 | Donor functionality | As a Donor I can list all the donation done for all the NGOs  Considerations:   1. While fetching the above details the API must return the donation details (NGO Id, Donation Type, Amount, Donation Date) along with the donor details. 2. Donation details must be in descending order of donation amount. | API and Frontend to be developed |
| US\_05 | Donor functionality | As a Donor I am able to update my details  Constraints:   1. If Donor try to update an NGO id which does not exists, throw a custom Exception | Only API to be developed |

# Proposed Rest Endpoints to be exposed

## Rest APIs:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **URL Exposed** | **Purpose** | | /e-donation/api/v1/donors/register-donor | Register a Donor | | /e-donation/api/v1/donors/by-ngo-id/{ngoId} | Fetches details of all the donors registered with a NGO | | / e-donation/api/v1/donation/add-donation | Donate for an NGO | |
| |  |  | | --- | --- | | **URL Exposed** | **Purpose** | | /e-donation/api/v1/donations/by-id/{donorId} | List all the donation done by a donor | | / e-donation/api/v1/donor/update-donor | Updates the donor details | |

# Implementation/Functional Requirements

## Product and Frameworks:

1. **Presentation**

Develop the frontend for User Story 4. (Implementation as follows)

1. Implement using either Angular or React.
2. Implement any one of the Gang of four Patterns to compose data using typescript before presenting the same on UI.
3. Implement at least one approach for UI performance consideration.
4. Identify and Implement client-side Optimization Techniques for Bootstrap.
5. **Compute and Integration**

Develop the backend application as a RESTful Application. (Implementation as follows)

* + 1. Use any one of the Creational Design patterns for composing the model object to be sent back as response on following endpoint:

/ e-donation/api/v1/donations/show-donation/{ngoId}

(Fetches details of all donations done for a particular NGO)

* + 1. Optimize you REST endpoints to allow filtering, sorting, and pagination.

## Governance and Tooling:

* + - 1. Follow the practise of Creating Testable Component
      2. Configure your frontend application to implement Unit Testing using Jasmin & Karma (Optional).
      3. Configure your backend application to implement Unit Testing using Junit and Mockito.
      4. Test suites must contain exception situation testing.
      5. Generate the Code Coverage report of the same.
      6. Generate the Code Quality Report of the backend application.

## Code Quality/Optimizations

1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.
3. Generate the code quality report of the same.