**Dharmsinh Desai University,**

**Nadiad**

**Faculty of Technology**



**Department of Computer Engineering**

B. Tech. CE Semester – VI

**Subject**: System Design Practice

**Project** **Title**: NGO System

**By:**

Ayushi Tikoo :- ID:17CEUOG130 Roll No:CE-007

Ashwini Dhandhukia :- ID:17CEUOS122 Roll No:CE-028

**Guided By:**

Assistant Prof. Sheetal Shah,

Dept. of Computer Engg.

Faculty of Technology

****

**Dharmsinh Desai University, Nadiad**

Faculty of Technology, Department of Computer Engineering

**CERTIFICATE**

This is to certify that System Design Practice Project entitled “**NGO System”**

**1)Ayushi Tikoo(17CEUOG130)**

**2)** **Ashwini Dhandhukia (17CEUOS122)**

Of Department of Computer Engineering ,Semester VI , academic year 2019-2020, under our supervision and guidance.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Guide** |  | **HOD** |
| **Assistant Prof. Sheetal Shah**  Assistant Professor of Department of Computer Engineering, Dharmsinh Desai University, Nadiad |  | **Dr. C. K. Bhensdadia**  Head of Department of Computer Engineering, Dharmsinh Desai University, Nadiad |

**Contents**

**1) Abstract** --------------------------------------------------------------------------------4

**2) Introduction** - -------------------------------------------------------------------------5

**3) Software Requirement Specifications** ------------------------------------------11

**4) Design** ---------------------------------------------------------------------------------13

1. Use Case Diagrams
2. Class Diagrams
3. Sequence Diagrams
4. Activity Diagrams
5. State Diagrams
6. E-R Diagrams (for the database applications)
7. Data Dictionary (for the database applications), which includes table information

**5) Implementation Detail**------------------------------------------------------------23

1. Modules created and brief description of each module
2. Function prototypes which implements major functionality

**6) Testing**---------------------------------------------------------------------------------24

**7) Screen-shots** -------------------------------------------------------------------------32

**8) Conclusion**----------------------------------------------------------------------------39

**9) Limitation and Future Extension** -----------------------------------------------39

**10)Bibliography**------------------------------------------------------------------------40

Abstract

The project report upon the “**NGO System**” will provide various event information and one can donate with this website.

It is a system that enables users to check for different event available at the website and can register his/her self for volunteering on the site. He may even donate through a debit card/credit card or net banking. Once the donor makes an effective transaction he gets confirmation of donation. Donor can also donate other things like books, dresses, food etc. for that they can also send pickup request to NGO. One can also get location of our centers through map.

**INTRODUCTION**

* 1. **PROJECT OVERVIEW:**

The project comprises the list of available events displayed on different dates. He can even pay through a credit card or debit card. Once the donor makes an effective transaction he gets a confirmation message.

* 1. **PURPOSE: GOALS & OBJECTIVES**

The purpose of this project is to develop the user friendly and interactive website which will allow the user to donate online. user will be able to search and can send request for volunteering and can donate in terms of online Payments through Credit card/debit card and Get the confirmation message.

The main purpose of this project is giving flexibility to the Information as this website plays main role at admin side. Without login or any Customer, who have not idea about event, They easily Find and register themselves. Anyone will be able to open this site and get the information about the event details and can see achievements of NGO.

* 1. **SCOPE**

The aim of the project is to build a simple, effective computerized NGO system along with displaying the information of any types of query of event information in my web application.

**Functions for user**

* Registration
* Login
* Search event date and place wise
* Request for volunteering
* Donate money/other things
* Go For Checkout
* Edit account information
* Add Feedback

**Functions for the Admin**

* Admin Login
* Manage All volunteer
* Manage event details
* View transaction/donated amount
* Reply to feedback
* **Software requirement :**
* Operating System : Microsoft Windows 7/8 or higher
* It Create website in Angular
* Language: node-Express js
* Front-End :
* Back-End : Node.js, phpMyAdmin
* Designing Tool : Xampp
* Documentation Tool: Ms-Word 2016, ERD Plus

About node.js

* **What is Node.js?**

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. Node.js is a platform built on [Chrome's JavaScript runtime](https://code.google.com/p/v8/) for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

## Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

* **Asynchronous and Event Driven** − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **Single Threaded but Highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.
* **License** − Node.js is released under the [MIT license](https://raw.githubusercontent.com/joyent/node/v0.12.0/LICENSE).

## Who Uses Node.js?

Following is the link on github wiki containing an exhaustive list of projects, application and companies which are using Node.js. This list includes eBay, General Electric, GoDaddy, Microsoft, PayPal, Uber, Wikipins, Yahoo!, and Yammer to name a few.

* [Projects, Applications, and Companies Using Node](https://github.com/joyent/node/wiki/projects,-applications,-and-companies-using-node)

## Concepts

The following diagram depicts some important parts of Node.js which we will discuss in detail in the subsequent chapters.



## Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

* I/O bound Applications
* Data Streaming Applications
* Data Intensive Real-time Applications (DIRT)
* JSON APIs based Applications
* Single Page Applications

**About Express.js**

ExpressJS is a web application framework that provides you with a simple API to build websites, web apps and back ends. With ExpressJS, you need not worry about low level protocols, processes, etc.

## What is Express?

Express provides a minimal interface to build our applications. It provides us the tools that are required to build our app. It is flexible as there are numerous modules available on **npm**, which can be directly plugged into Express.

Express was developed by **TJ Holowaychuk** and is maintained by the [Node.js](https://nodejs.org/en/) foundation and numerous open source contributors.

## Why Express?

Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable.

### **Pug**

Pug (earlier known as Jade) is a terse language for writing HTML templates. It

* Produces HTML
* Supports dynamic code
* Supports reusability (DRY)

It is one of the most popular template language used with Express.

### **What is phpMyAdmin?**

### phpMyAdmin is one of the most popular applications for MySQL database management. It is a free tool written in PHP. Through this software you can create, alter, drop, delete, import and export MySQL database tables. You can run MySQL queries, optimize, repair and check tables, change collation and execute other database management commands

The main phpMyAdmin features are:

* Intuitive web interface
* Support for most MySQL features:
* browse and drop databases, tables, views, fields and indexes
* create, copy, drop, rename and alter databases, tables, fields and indexes
* maintenance server, databases and tables, with proposals on server configuration
* execute, edit and bookmark any SQL-statement, even batch-queries
* manage stored procedures and triggers
* Import data from CSV and SQL
* Export data to various formats: CSV, SQL, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, LAT*E*X and others
* Creating complex queries using Query-by-example (QBE)
* Searching globally in a database or a subset of it
* Transforming stored data into any format using a set of predefined functions, like displaying BLOB-data as image or download-link

**Software Requirement Specification for NGO System**

Types of users:

-Admin

-Client

Client-

R1. Registration

Description: user enters personal details like name, email, phone number, address, Password

Input - Details are entered

Output - Successfully registered message

R2. Login

Description: user enters email and password

Input - Details are entered

Output - Successfully login message

R3. Manage donation

R3.1. Donation

Description: user enters type of donation(money or materialistic item),

Name, email,phone number

Input-details are entered

Output-payment page (if money is donated)

R3.2.Payment

Description: user enters card name, card number, CVV and expiry date

Input-Details are entered

Output- Payment done successfully

R4. Apply as volunteer

Description: user can apply as volunteer in different events organized by ngo

by entering name, email, phone number, address

Input - Details are entered

Output - payment successful

R5.Get Help

Description: description about problem is entered, name, phone number(optional),email

Input-details are entered

Output-Message sent

R6.Write to us

Input-user can give feedback or suggestions

Output-Message sent

ADMIN-

R1.Login

Description: user enters email and password

Input-Details are entered

Output-Successfully login message

R2.Manage children

Description-admin can add, update and delete child details

R2.1.Add children

Input-Child details like name, age, DoB, place where child is found and why

child is taken, date of registration

Output-child added successfully

R2.1.Delete children

Input-User selection

Output-child deleted successfully

R2.1.update children information

Input- children details

Output-child information updated successfully

R3. Manage event

Description-admin can add, update and delete event details

R2.1.Add event

Input-event details like title,place, date, description, guest name

Output-event added successfully

R2.1.Delete event

Input-User selection

Output-event deleted successfully

R2.1.update event information

Input- event details

Output-event information updated successfully

R4. Manage volunteer

Description-admin can choose and see volunteer details

Input-user selection

Output-confirmation email sent.

R5.display donation

Description-admin can see donor's details and their contribution.

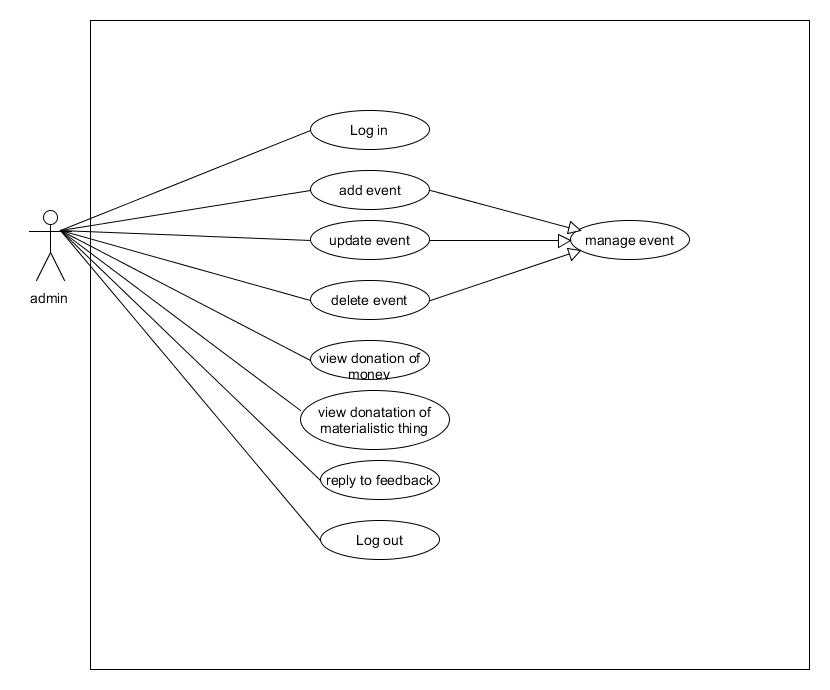
Input-user selection

Output-list of doners.

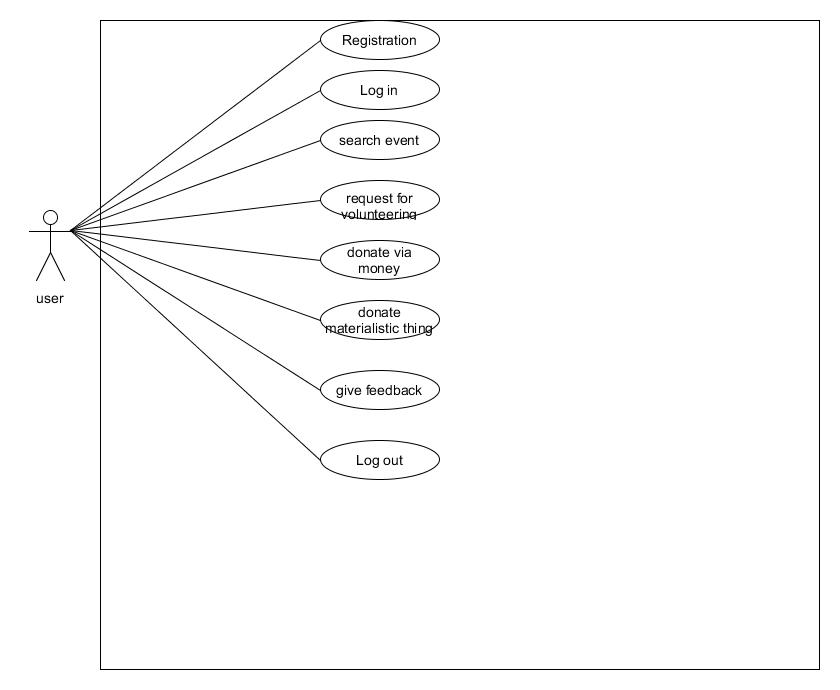
**Design**

**1) use case diagram**

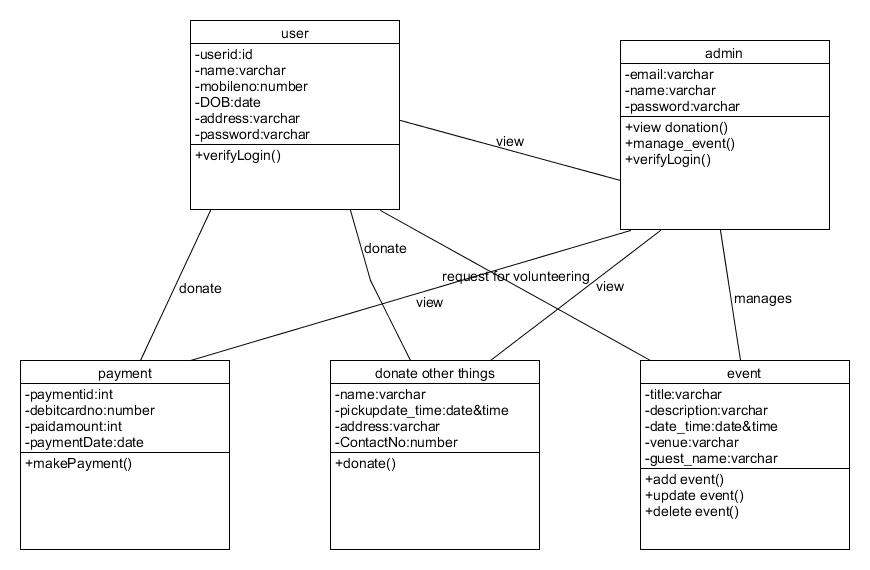
**Admin side:-**



**User side use case diagram:-**

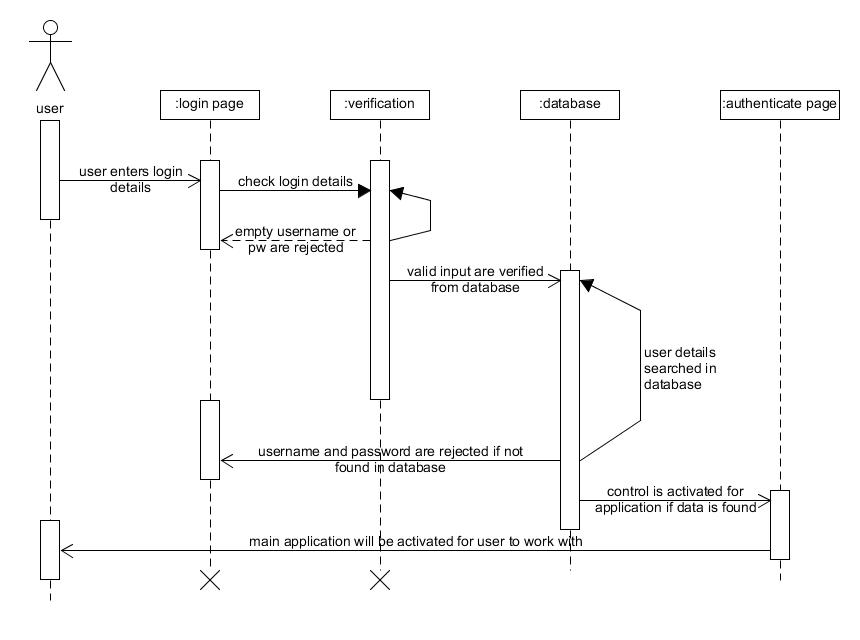


2) **Class Diagram:-**

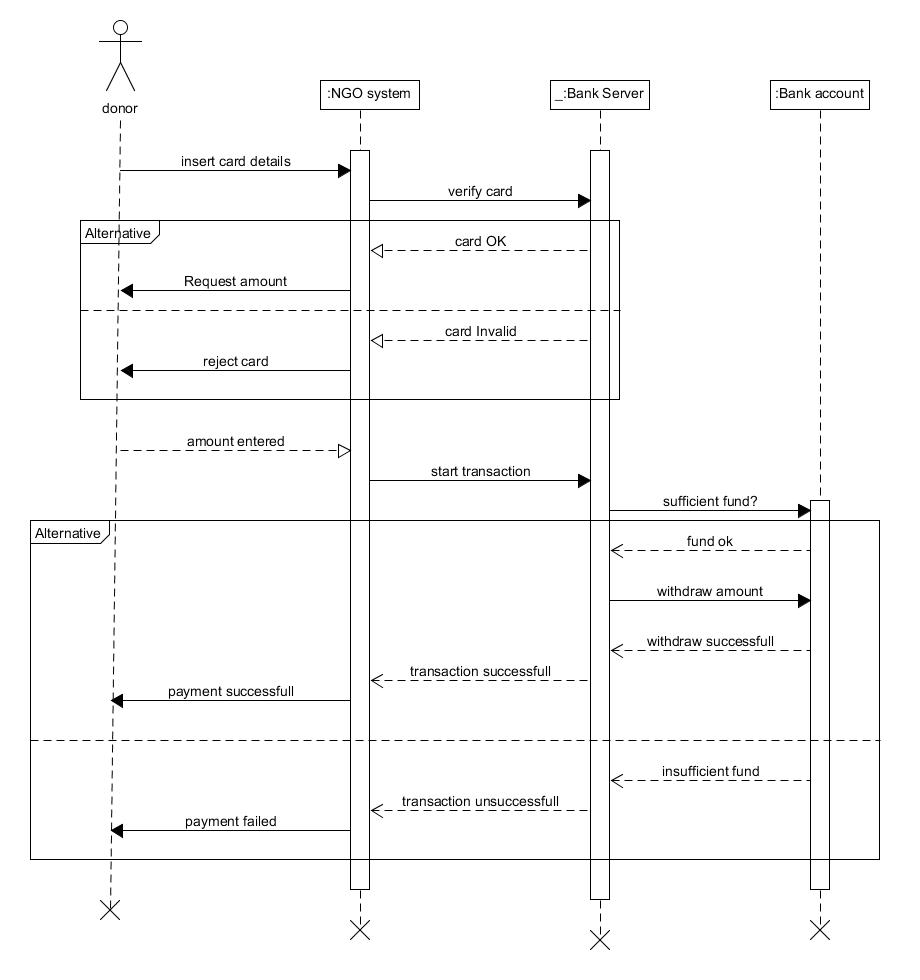


**3) Sequence diagram:-**

**For login process-**

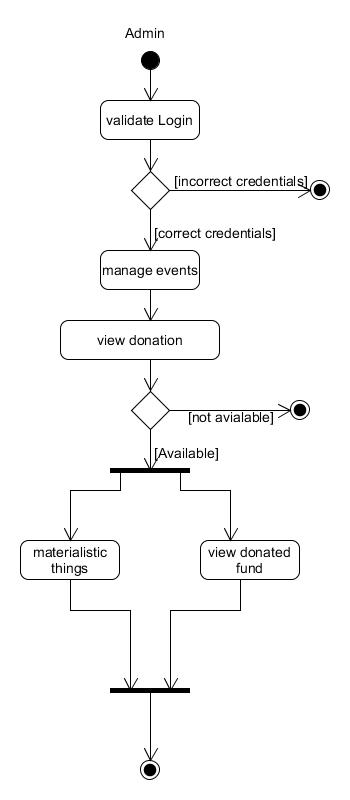


**For donate money(payment)process :**

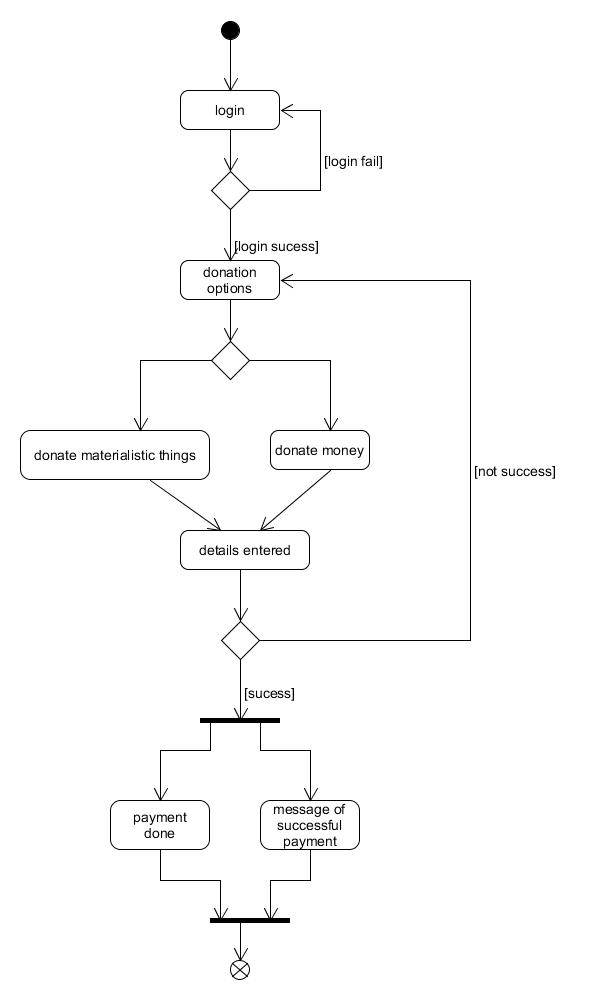


**4) Activity diagram:-**

**Admin side-**

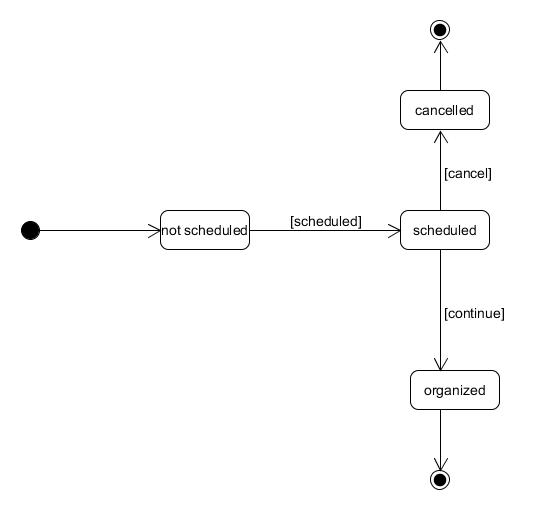


**User side activity diagram-**

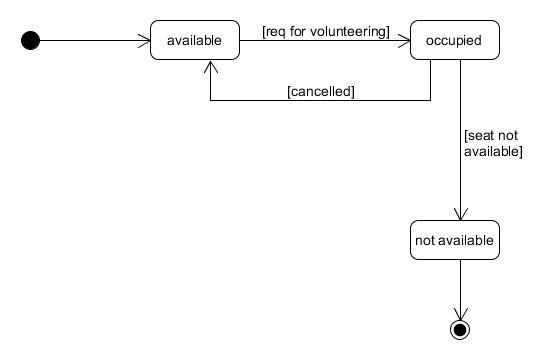


**5) State diagram:-**

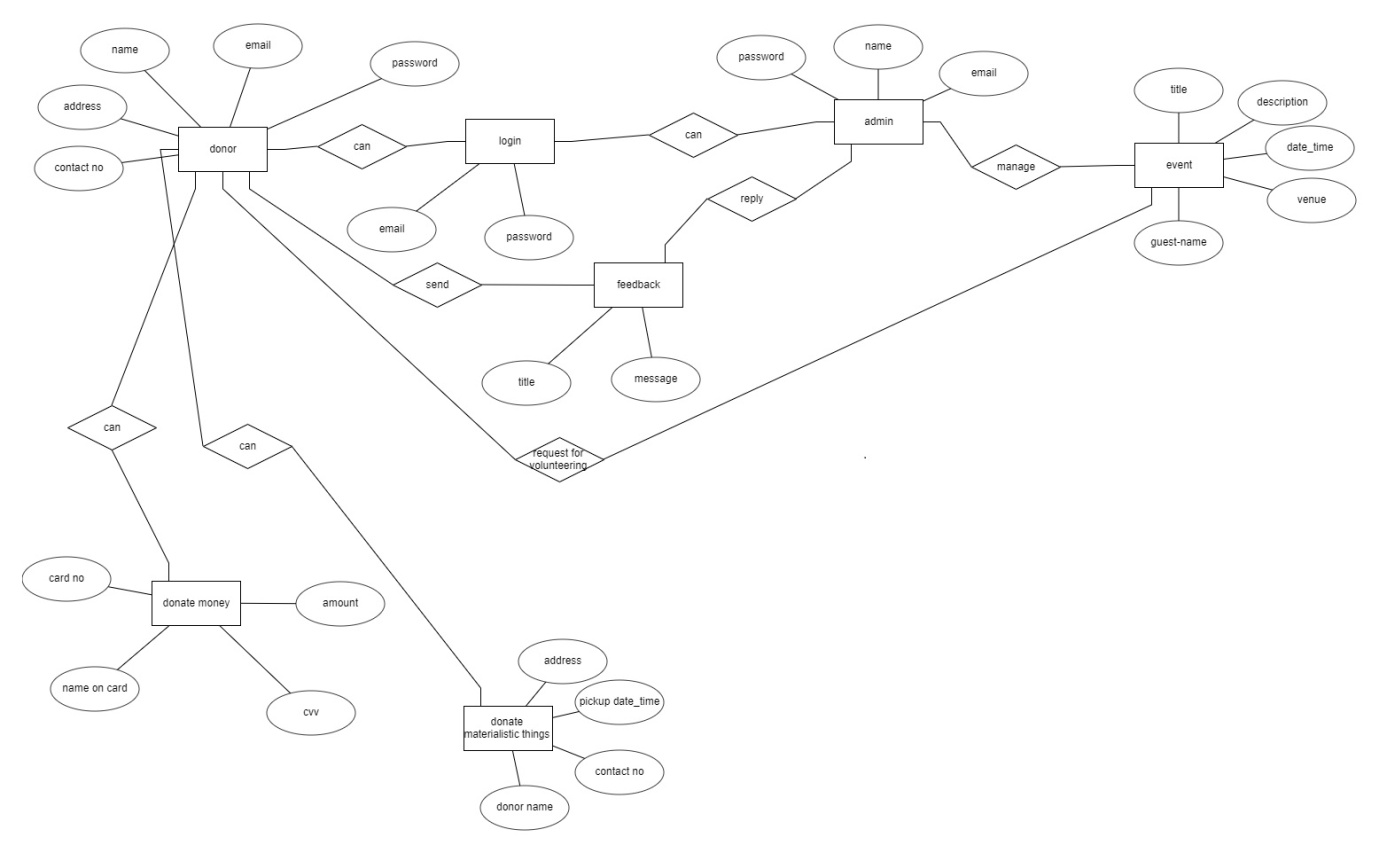
**For manage\_event-**



**For volunteering event-**



**6)ER-Diagram:-**



**7)Data Dictionary:-**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| volunteer | | | | | | |
| Sr no | name | Datatype | size | required | Unique | PK/FK |
| 1 | id | Int | 11 | Auto\_increment | yes | Primary key |
| 2 | Name | Varchar | 100 | - |  |  |
| 3 | Mobile | Varchar | 10 | - |  |  |
| 4 | Email | Varchar | 100 | - |  |  |
| 5 | eName | varchar | 200 | - |  | Foreign key |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| mdonar | | | | | | |
| Sr no | name | datatype | size | required | unique | PK/FK |
| 1 | Id | Int | 200 | Auto\_increment | yes | primarykey |
| 2 | Email | Varchar | 100 |  |  |  |
| 3 | sName | Varchar | 100 |  |  |  |
| 4 | sMobile | Varchar | 10 |  |  |  |
| 5 | sAddress | Varchar | 300 |  |  |  |
| 6 | sCity | Varchar | 100 |  |  |  |
| 7 | sState | Varchar | 100 |  |  |  |
| 8 | sDate | Date |  |  |  |  |
| 9 | Donationtype | varchar | 200 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| feedback | | | | | | |
| Sr no | name | dataType | size | required | unique | PK/FK |
| 1 | Id | Int | 100 | Auto increment | yes | primarykey |
| 2 | Email | Varchar | 100 |  |  |  |
| 3 | Category | varchar | 100 |  |  |  |
| 4 | comments | varchar | 200 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Event | | | | | | |
| Sr no | name | datatype | size | required | unique | PK/FK |
| 1 | Id | Int | 200 | Auto increment | yes | primarykey |
| 2 | eName | Varchar | 200 |  |  |  |
| 3 | eLocation | Varchar | 200 |  |  |  |
| 4 | eTime | time | 6 |  |  |  |
| 5 | eDate | Date |  |  |  |  |
| 6 | eDescription | varchar | 400 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| donar | | | | | | |
| Sr no | name | datatype | size | required | unique | PK/FK |
| 1 | Id | Int | 200 | Auto increment | yes | Primary key |
| 2 | cName | Varchar | 200 |  |  |  |
| 3 | cEmail | Varchar | 200 |  |  |  |
| 4 | cMobile | Varchar | 10 |  |  |  |
| 5 | cAddress | Varchar | 200 |  |  |  |
| 6 | cCity | Varchar | 200 |  |  |  |
| 7 | cState | Varchar | 200 |  |  |  |
| 8 | cZip | Int | 6 |  |  |  |
| 9 | Donation type | Varchar | 200 |  |  |  |
| 10 | cAmount | int | 200 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| customerInfo | | | | | | |
| Sr no | name | datatype | size | required | unique | PK/Fk |
| 1 | Id | Int | 100 | Auto increment | yes | primarykey |
| 2 | Email | varchar | 100 |  |  |  |
| 3 | cName | Varchar | 100 |  |  |  |
| 4 | cMobile | Varchar | 100 |  |  |  |
| 5 | cPassword | varchar | 100 |  |  |  |

**Implementation Details**

**i)Modules created and brief description of each module**

**CLIENT SIDE-**

**Routes And Views**

* **Dashboard-** after user logins into the system then this page is displayed showing all the options available to him. Feedback, Donation and Event Registration.
* **DonarC-** user can donate money by filling up the form and providing details about himself and the money he wants to donate. Donation can only be done via card
* **FeedbackC-** user can give feedback by filling up the form providing email , category in which he wants to give feedback and comments.
* **FrontPage-** It’s the very first page which will be displayed to the user when he starts the application.
* **LoginC-** Display the login and registration form to the user. User needs to register himself first into the system and then login in.
* **EventsC-** All the events that will be organised by the ngo will be displayed and the user can also enrol himself into any of them by registering.
* **DonarMC-** The users who want to donate other materialistic things can donate here in this form . It displays a form asking for users details and the donation type. The material will get picked up from the users home by our agents on the date specifies by him.
* **Thanks-** page for thanking the donar for providing donation(money).
* **ThanksEventRegister-** Thanking the user for participating in the event.
* **ThanksFeedback-** Thanking the user giving us it valuable feedback.

**ADMIN SIDE-**

**Routes and Views**

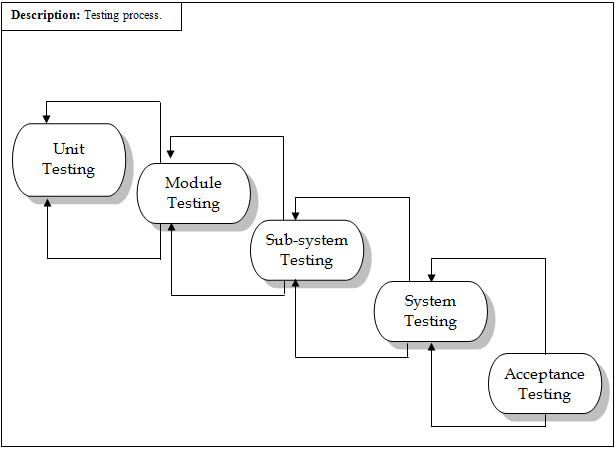
* **Customer-**handles insertion , deletion and Displaying of users of the system
* **Donar-** Displays all the Donars to the admin.
* **Feedback-** Displays the feedback of the donars, volunteers and users.
* **Index-** Handles admin Login.
* **Index1-** Displays all the options available to the admin after logging in like view feedback , donar, users and manage events.
* **Login-** Form for the admin to login by providing the email and password.
* **Events-** Admin can manage all the events. He can insert any new event by providing all its details. Any event can be deleted and edited by him.
* **DonarM-** Displays all the materialistic things donars of the system along with the things they want to donate.
* **Volunteer-** Displays all the volunteers of the ngo of various events. And can also remove them.

**TESTING**

* Testing Plan
* Testing Strategy
* Testing Methods
* Test Cases

**TESTING PLAN**

The aim of the testing process is to identify all defects existing in software Product. However for most practical systems, even after satisfactorily carrying out the testing phase, it is not possible to guarantee that the software is error free. This is because of the fact that the input data domain of most software products is very large. It is not practical to test the software exhaustively with respect to each value that the input data may assume. Even with this practical limitation of the testing process, the importance of testing should not be underestimated. It must be remembered that testing does expose many defects existing in a Software product. Thus testing provides a practical way of reducing defects in a System and increasing the users’ confidence in a developed system.



Testing Process.

**Functional Testing:**

The testing technique that is going to be used in the project is black box testing. In black box testing the expected inputs to the system are applied and only the outputs are checked.

The working or the other parameters of the functionality are not reviewed or tested on the black box testing technique. There is a specific set of inputs for each and every module which is applied and for each set of inputs the result or the output is verified and if found as per the system working this testing is termed or result is declared as pass.

If the set of inputs that are provided to each module are not giving the outputs as per the expected results from the module then the result of that testing is to be declare failed.

Moreover the bottom up integration of the modules is applied herein so that each module can be verified at the initial stage and if it is found that the independent module is perfectly alright, only then it is going to be integrated with other related modules otherwise the module is checked for flaws and then if it satisfies all the specific requirements of the module, is integrated to other related modules to form and incorporate a system.

In the black-box testing approach, test cases are designed using only the functional specification of the software, i.e. without any knowledge of the internal structure of the software. For this reason, black-box testing is known as functional testing.

**Equivalence Class Partitioning:**

In this approach, the domain of input values to a program is partitioned into a set of equivalence classes. This partitioning is done such that the behaviour of the program is similar for every input data belonging to the same equivalence class. The main idea behind defining the equivalence classes is that testing the code with any one value belonging to an equivalence class is as good as testing the software with any other value belonging to that equivalence class. Equivalence classes for software can be designed by examining the input data and output data.

**Boundary Value Analysis:**

A type of programming error frequently occurs at the boundaries of different equivalence classes of inputs. The reason behind such errors might purely be due to psychological factors. Programmers often fail to see the special processing required by the input values that lie at the boundary of the different equivalence classes. For example, programmers may improperly use < instead of <=, or conversely <= for <. Boundary value analysis leads to selection of test cases at the boundaries of the different equivalence classes.

**Structural Testing:**

In the white-box testing approach, designing test cases requires thorough knowledge about the internal structure of software, and therefore the white-box testing is called structural testing.

**TESTING STRATEGY**

Software products are normally tested first at the individual component (or unit) level (called unit testing), also referred to as “Testing in the Small”. Then the components are slowly integrated and tested at each level of integration (known as Integration Testing).Finally, the fully integrated system is tested (called System Testing). Integration and system testing are known as “Testing in the Large”.

Thus, a software product goes through two levels of testing:

* Unit Testing
* System Testing

**Unit Testing:**

In unit testing the analyst tests the programs making up a system. For this reason, unit testing is sometimes called program testing. Unit testing gives stress on the modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained within that module alone. The errors resulting from the interaction between modules are initially avoided. For example, a hotel information system consists of modules to handle reservations; guest check in and check out; restaurant, room service and miscellaneous charges; convention activities; and accounts receivable billing. For each, it provides the ability to enter, modify or retrieve data and respond to different types of inquiries or print reports. The test cases needed for unit testing should exercise each condition and option.

Unit testing can be performed from the bottom up, starting with smallest and lowest-level modules and proceeding one at a time. For each module in bottom-up testing a short program is used to execute the module and provides the needed data, so that the module is asked to perform the way it will when embedded within the larger system.

**System Testing:**

The important and essential part of the system development phase, after designing and developing the software is system testing. We cannot say that every program or system design is perfect and because of lack of communication between the user and the designer, some error is there in the software development. The number and nature of errors in a newly designed system depend on some usual factors like communication between the user and the designer; the programmer's ability to generate a code that reflects exactly the systems specifications and the time frame for the design.

Theoretically, a newly designed system should have all the parts or sub-systems are in working order, but in reality, each sub-system works independently. This is the time to gather all the subsystem into one pool and test the whole system to determine whether it meets the user requirements. This is the last change to detect and correct errors before the system is installed for user acceptance testing. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits.

Testing is an important function to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully activated. Another reason for system testing is its utility as a user-oriented vehicle before implementation.

System testing consists of the following five steps:

* Program Testing
* String Testing
* System Testing
* System Documentation
* User Acceptance Testing

**TESTING METHODS:**

Software Testing involves executing an implementation of the software with test data and examining the outputs of the software and its operational behavior to check that it is performing as required.

Statistical Testing:

Statistical Testing is used to test the program’s performance and reliability and to check how it works under operational conditions. Tests are designed to reflect the actual user inputs and their frequency.

The stages involved in the static analysis for this system are follows.

* Control flow analysis
* Unreachable code
* Unconditional branches into loops
* Data use analysis
* Variable used before initialization
* Variables declared but never used
* Variables assigned twice but never used between assignments
* Possible array bound violations
* Declared variables
* Interface analysis
* Parameter type mismatches
* Parameter number mismatches
* Non-usage of the results of functions
* Uncalled functions and procedures
* Storage management faults
* Images not Stored in Resources
* Out of Bound -Program’s non-volatile memory

**Black-Box Testing:**

In Black-Box Testing also called as Functional Testing, Developer are concerned about the output of the module and software, i.e. whether the software gives proper output as per the requirements or not. The program just gets a certain input and its functionality is examined by observing the output.

In our project we have done the testing as follows:

* We have tested our functions of component to check the specification of our components.
* We selected input set to test the component like in query process we gave the different kinds of inputs to examine the output.
* We test software with sequences that have only single value.
* We used different sequences of different sizes in different tests.

**White-Box Testing:**

White Box Testing is also called ‘Glass Box’ or ‘Structural’ testing. The intention in white box testing is to ensure that all possible feasible flows of control paths through a subprogram are traversed while the software is under test.

We have done path testing to exercise every independent execution path through a component or program. If every independent path is executed then all statements in the components must have been executed at least once.

We checked graphics module and database access module, which have independent execution path. They are not related to each other. The structure of our program is also checked.

**Integration Testing:**

After our individual procedures of system ware tested out, we integrate them to create a complete system. This integration process involves building the system and testing the resultant system for problems that arise from component interactions.

We have applied top-down strategy to validate high-level components of a system before design and implementations have been completed. Because our development process started with high-level components, we worked down the component hierarchy.

P**erformance Testing:**

Performance testing is designed to test the runtime performance of the system within the context of the system. These tests were performed module level as well as system level. Individual modules were tested for required performance.

* In performance testing we counted the processing time and response of operation.
* We also checked out the total execution time for intersection file creation.

**Interface Testing:**

Interface testing is integral part of Integration testing. Therefore Developer checked for the following:

* Interface misuse.
* Interface misunderstanding.

We examined the code to be tested and explicitly list each call to an external component. In the system, standards tests for GUIs have been performed, which are as follows.

* The position and related labels for all controls checked.
* All menu functions and sub functions verified for correctness.
* Validations for all inputs done.
* Each menu functions tested, whether it invokes the corresponding functionality properly.
* Whether the system prompts the user with appropriate message as and when invalid information is entered.
* All required fields are not left blank.

**Object Testing:**

Object testing is to test object as individual components, which are often larger than single function. Here following activities have taken place,

* Testing the individual operations associated with object
* Testing individual object classes
* Testing cluster of objects
* Testing object-oriented system

**CONDITION TESTING:-**

Coding testing is a test case design method that exercises the logical conditions contained in a program module. If the condition is incorrect, then as least one component of the condition is incorrect. It may include

* Boolean operator error
* Boolean variable error
* Boolean parenthesis error
* Relational operator error
* Arithmetic expression error

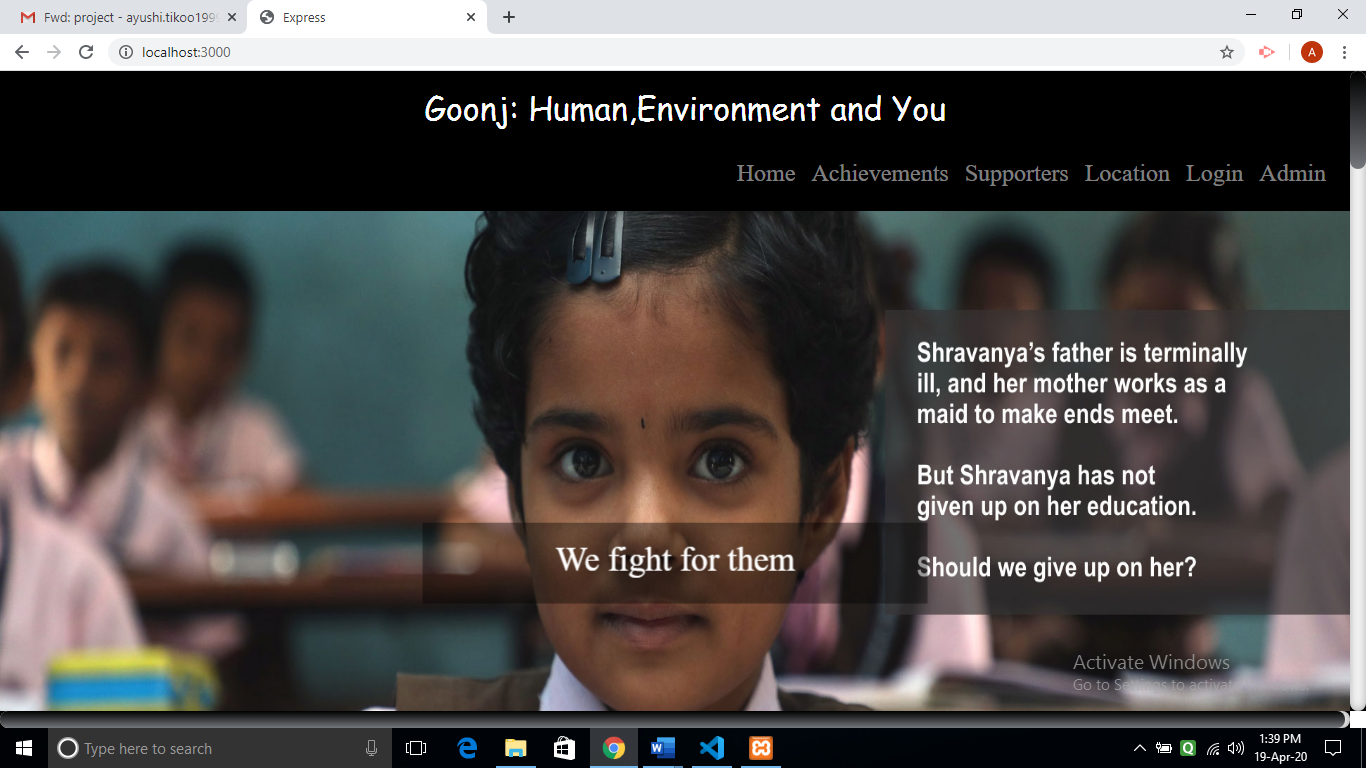
**VALIDATION TESTING:-**

Validation Testing is completely associated with requirement satisfaction of customers. This testing checks weather all functional requirements of the customer are satisfied or not. According to this test, the project is tested and found to be satisfactory for functional characteristics, behavioral characteristics and performance requirement. It is also found to have good documentation up to the last stage. So, the performance characteristics conform to specification and are accepted.

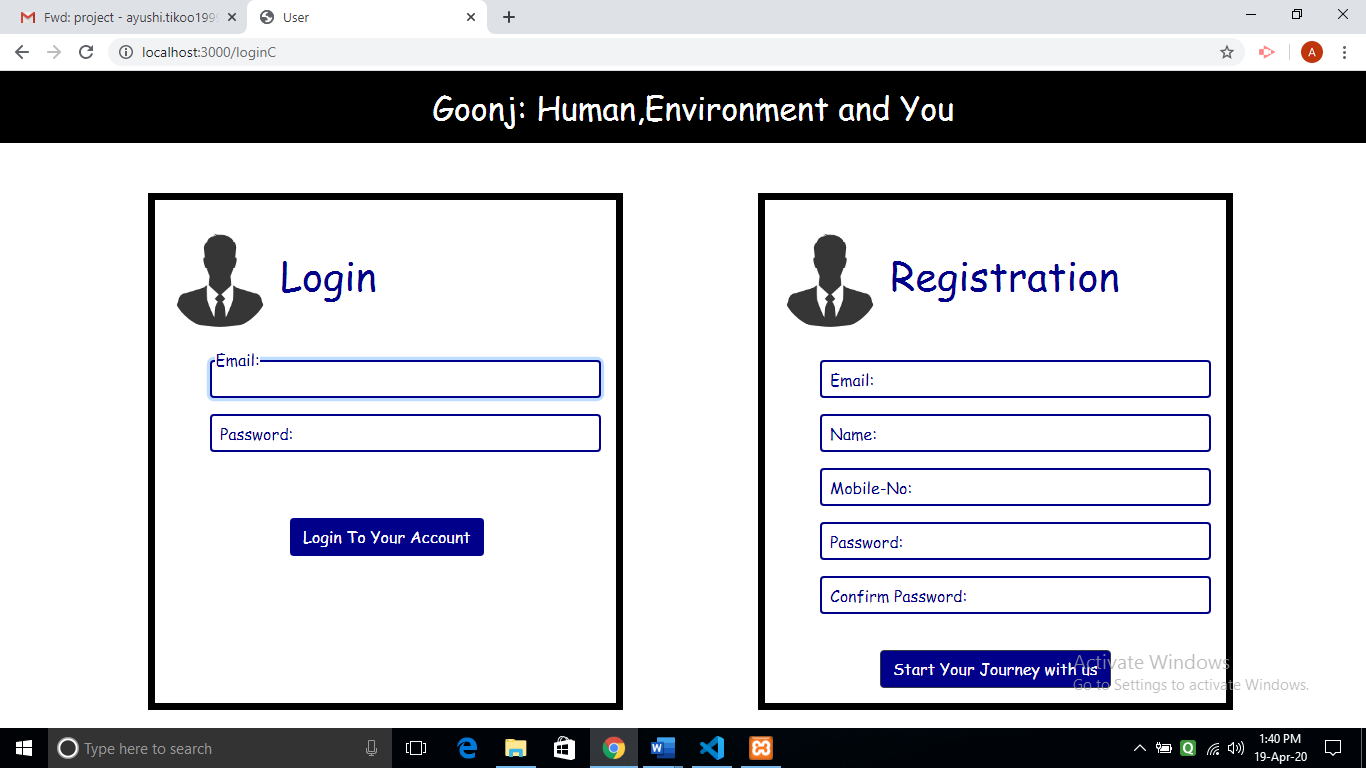
**Screenshots (Important)**

**USER-**

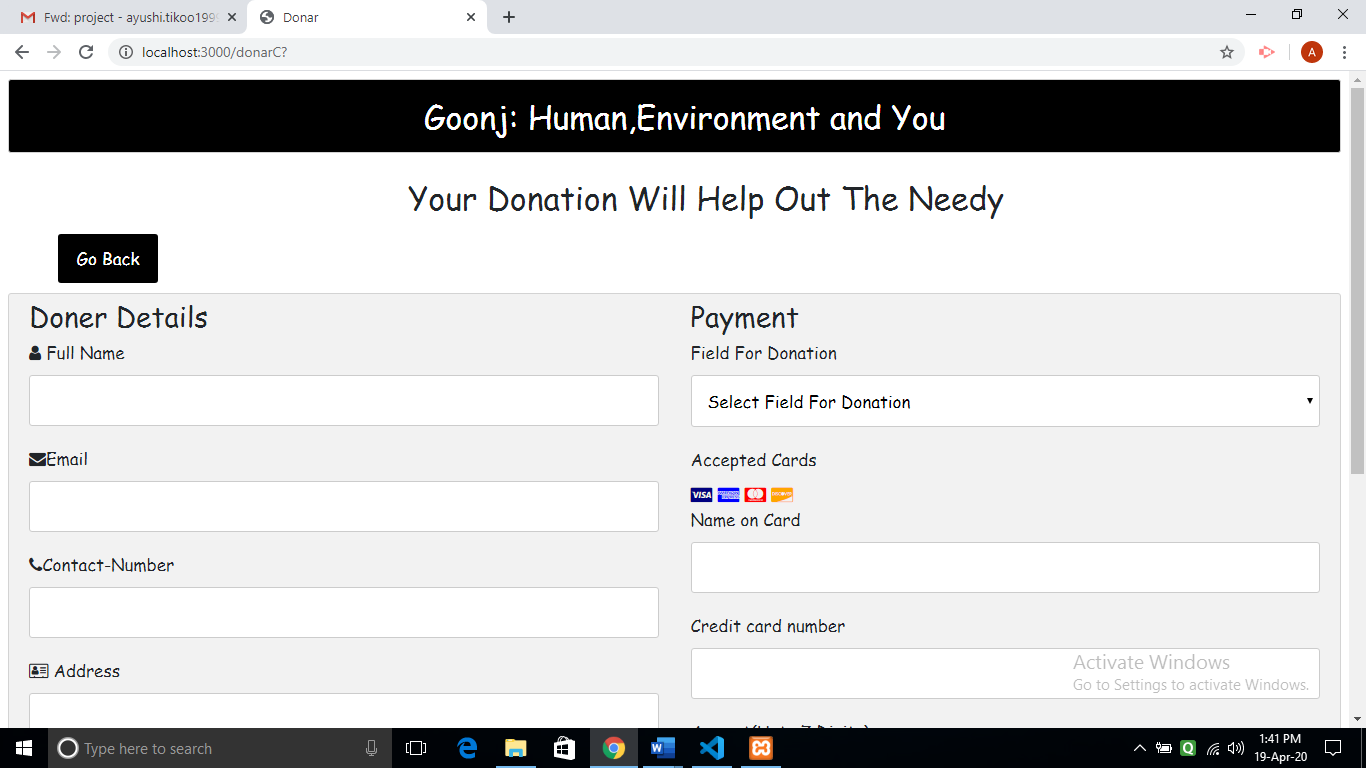
**Front page :**



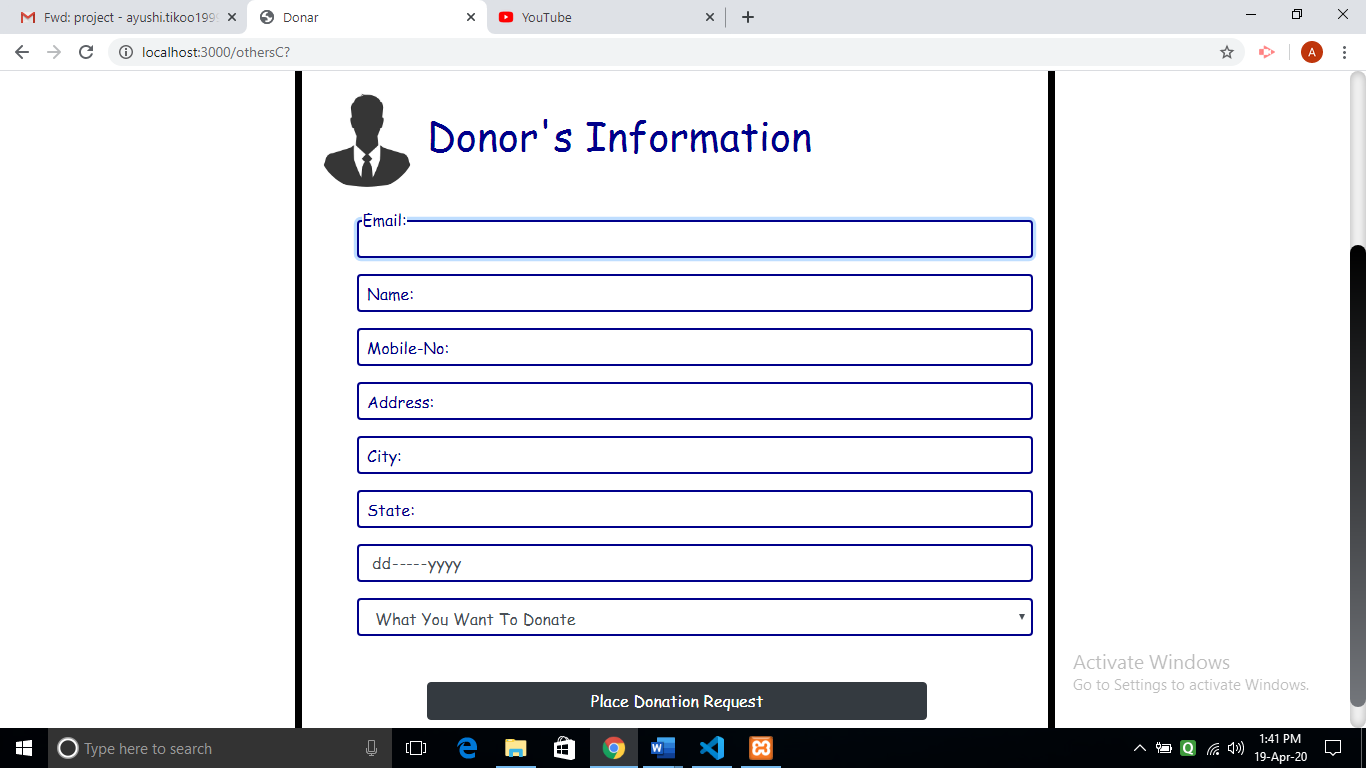
**Login Registration page for user :**



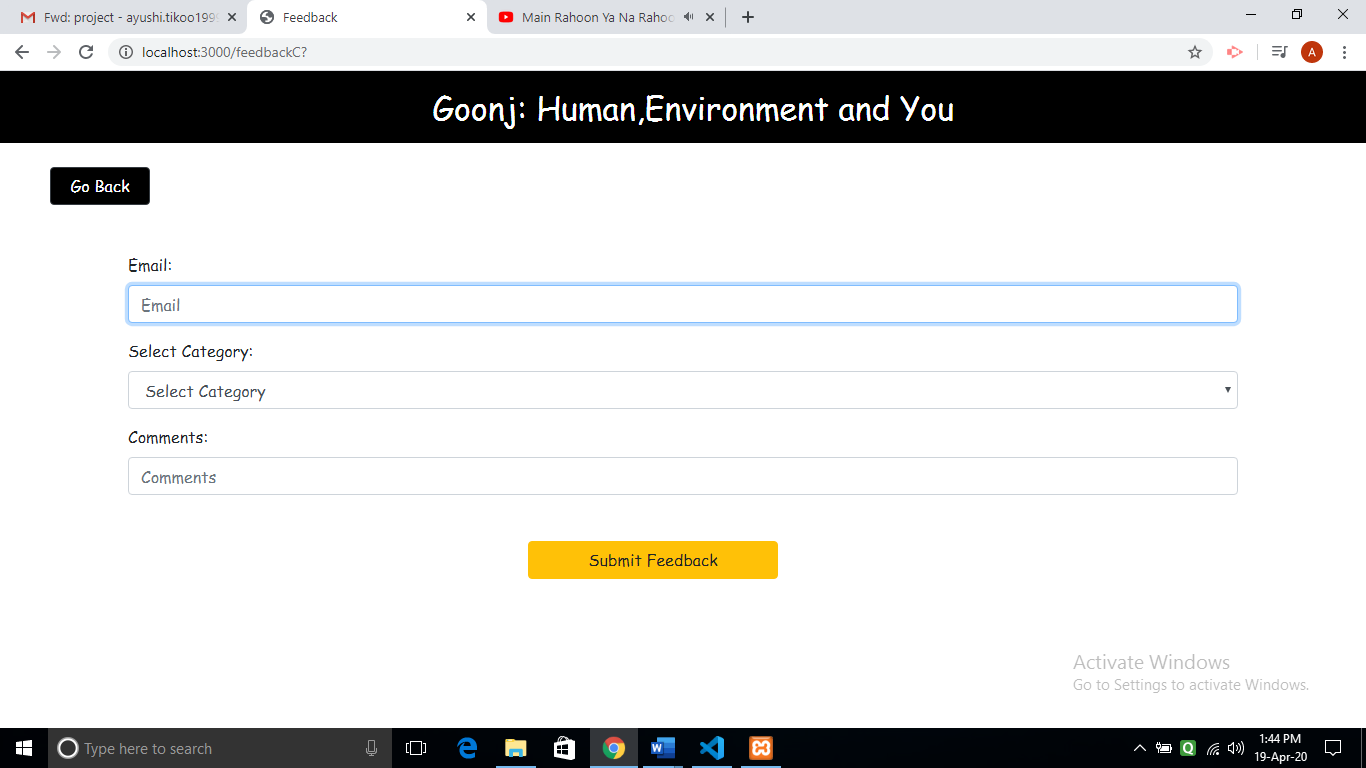
**Money donation :**



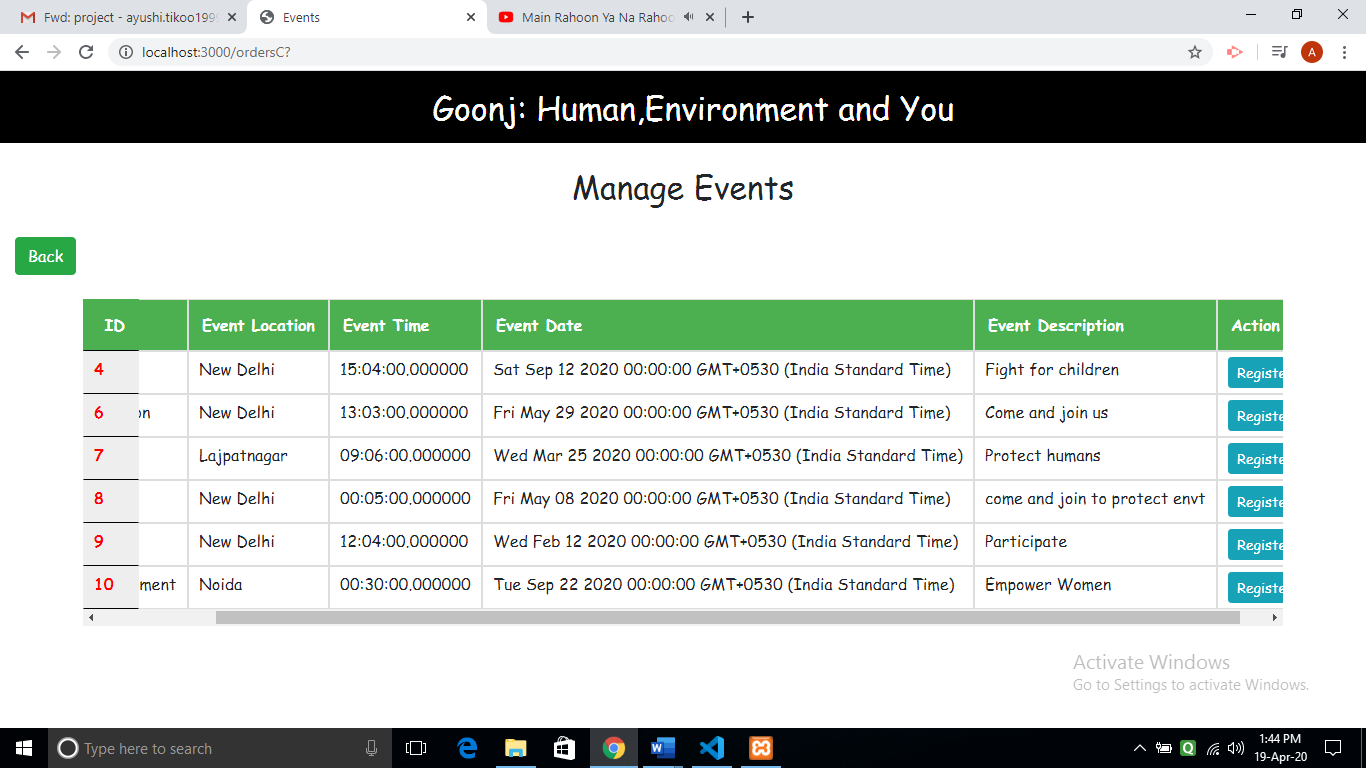
**To donate materialistic things :**



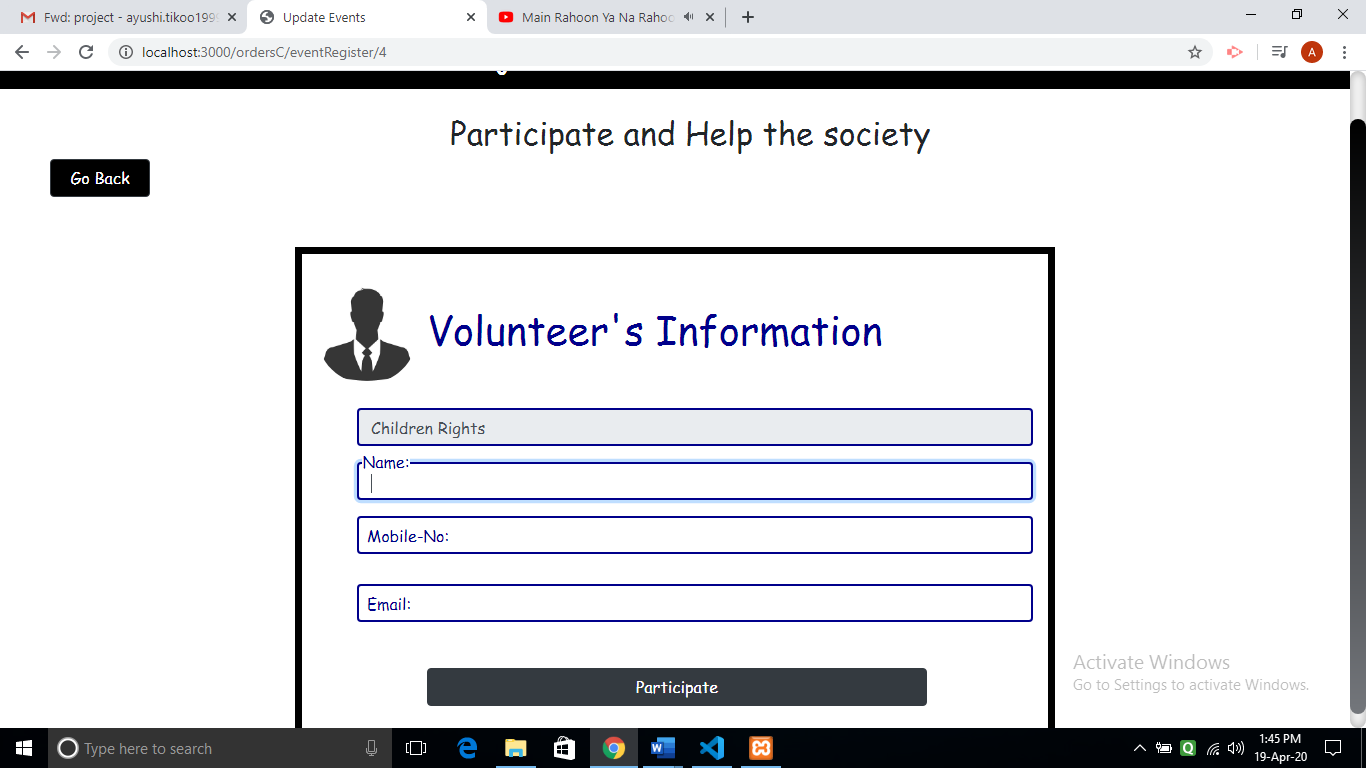
**Feedback form :**



**View Events :**

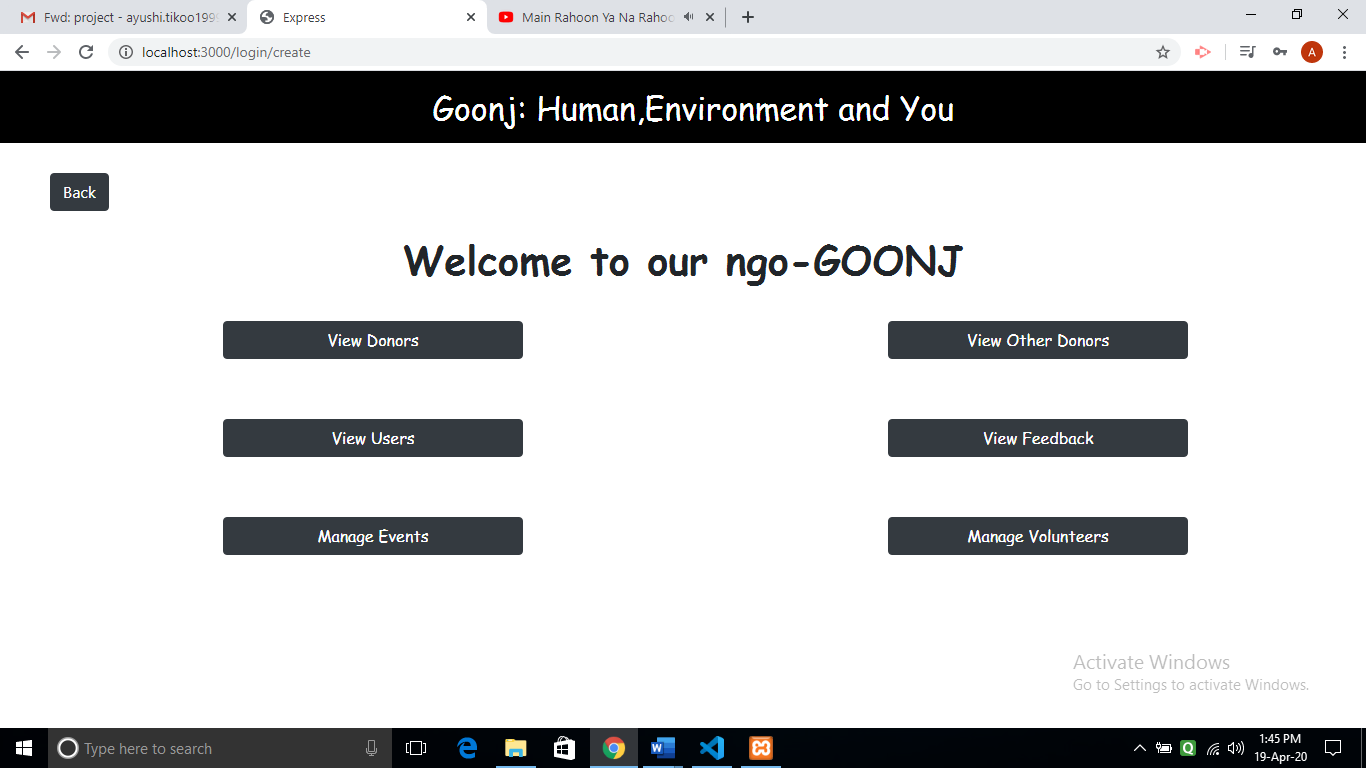


**Participate in event form :**



**ADMIN-**

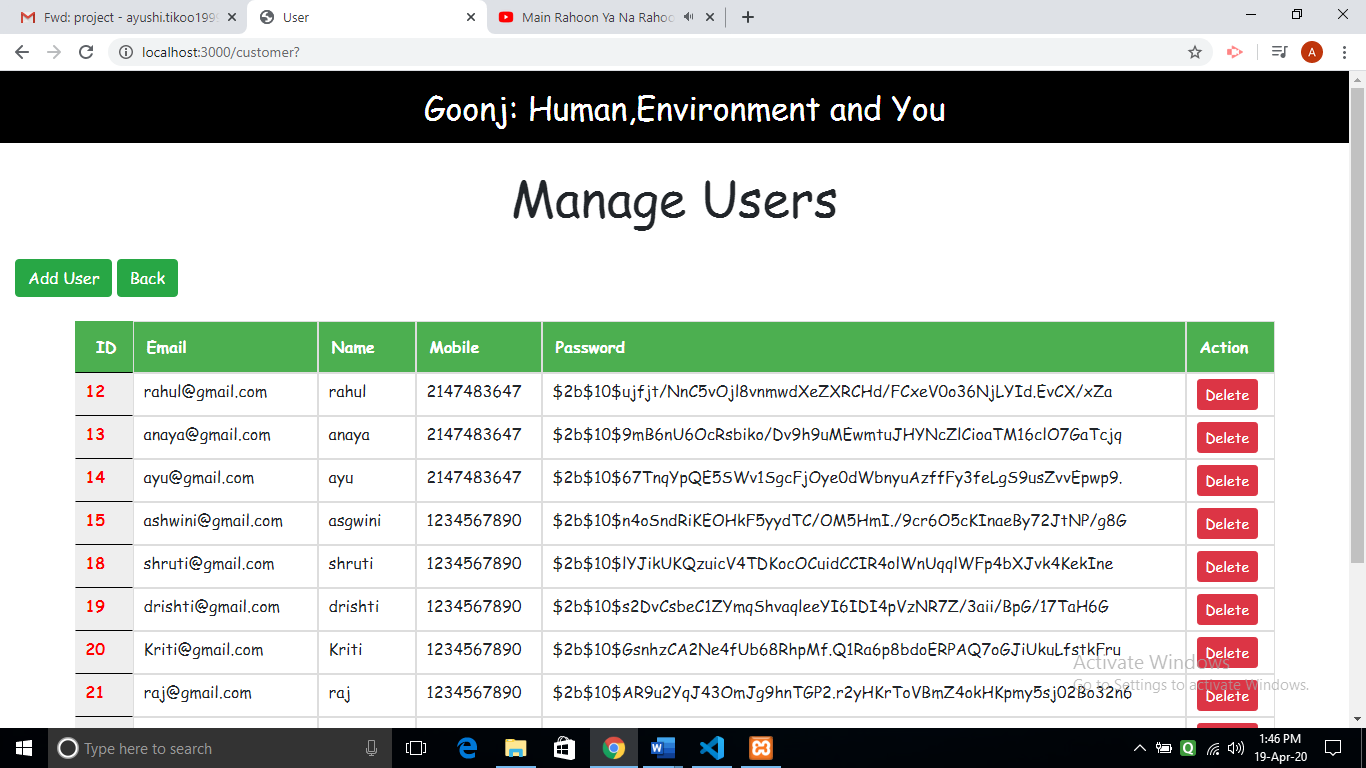
**Welcome Page for admin :**



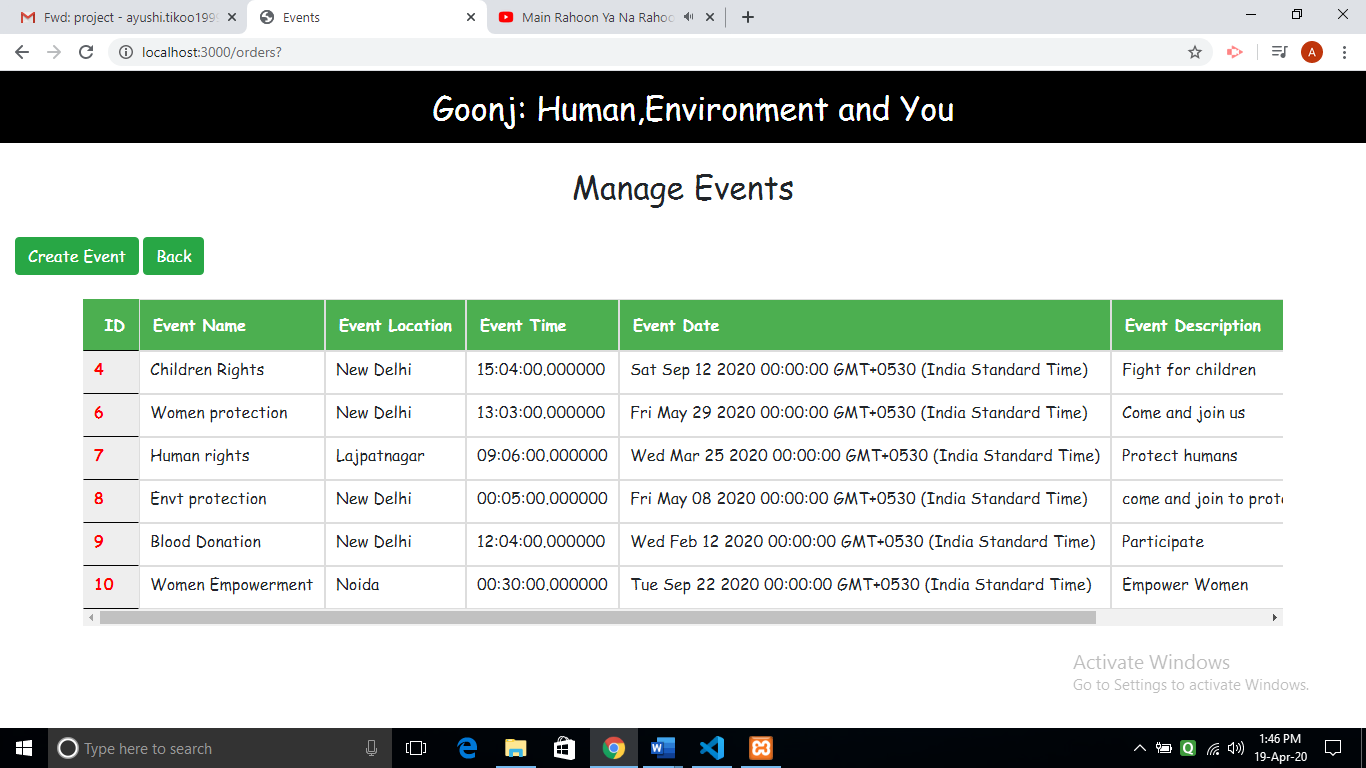
**View Donar :**



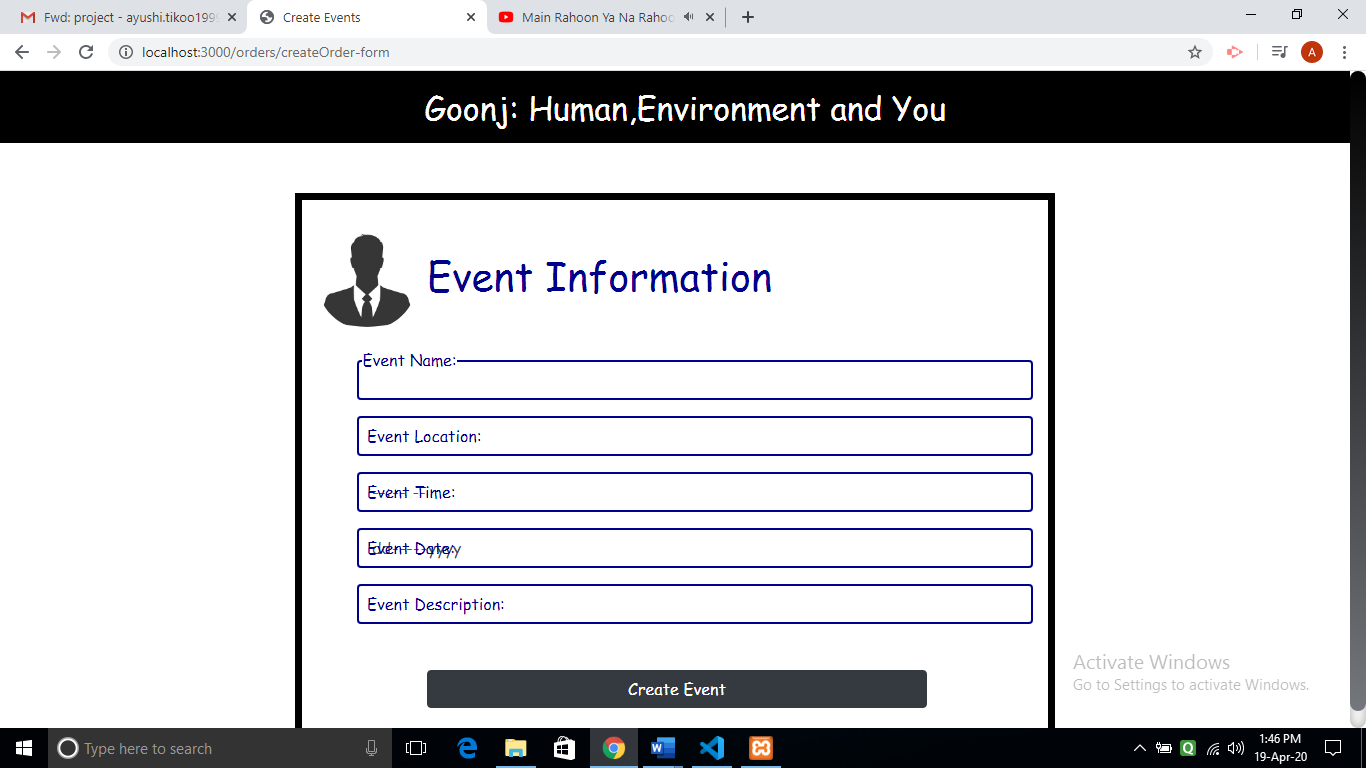
**Manage Users :**



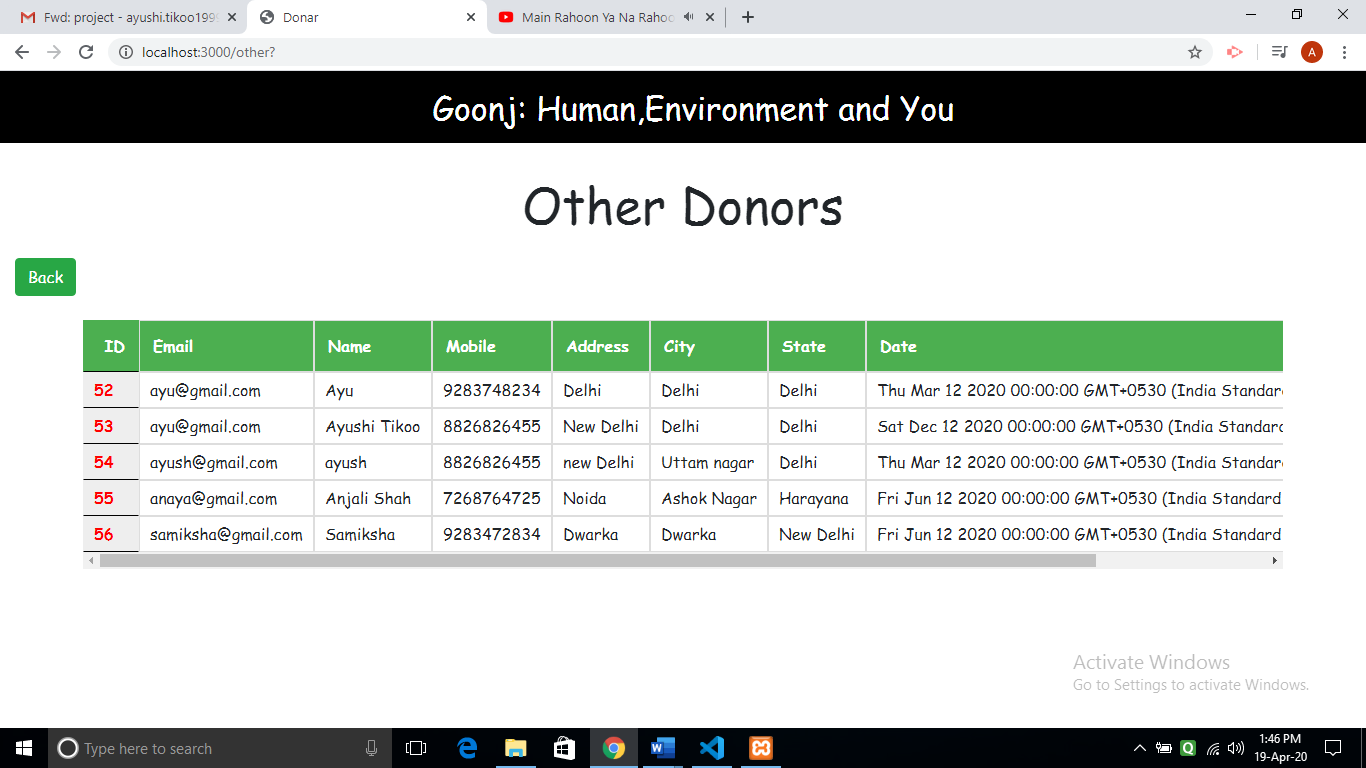
**Manage Events :**



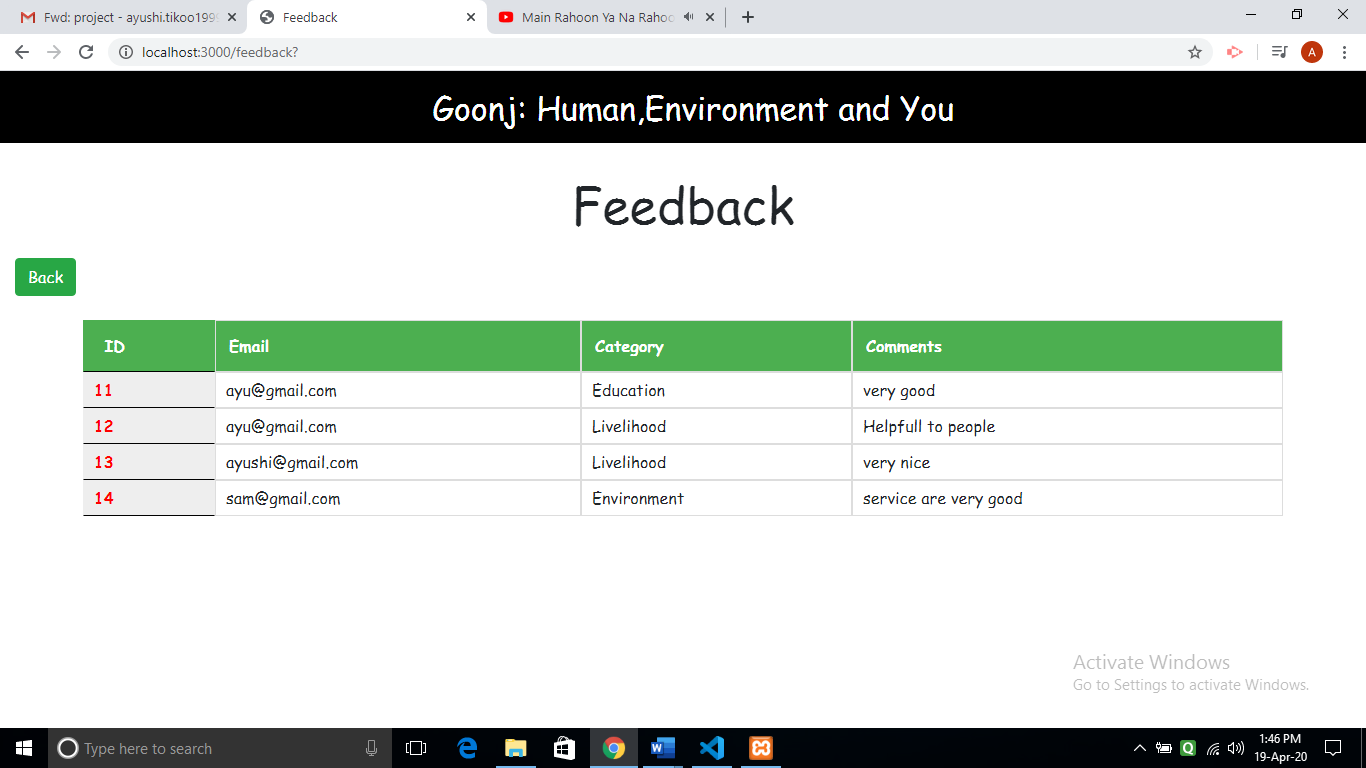
**Insert New Event :**



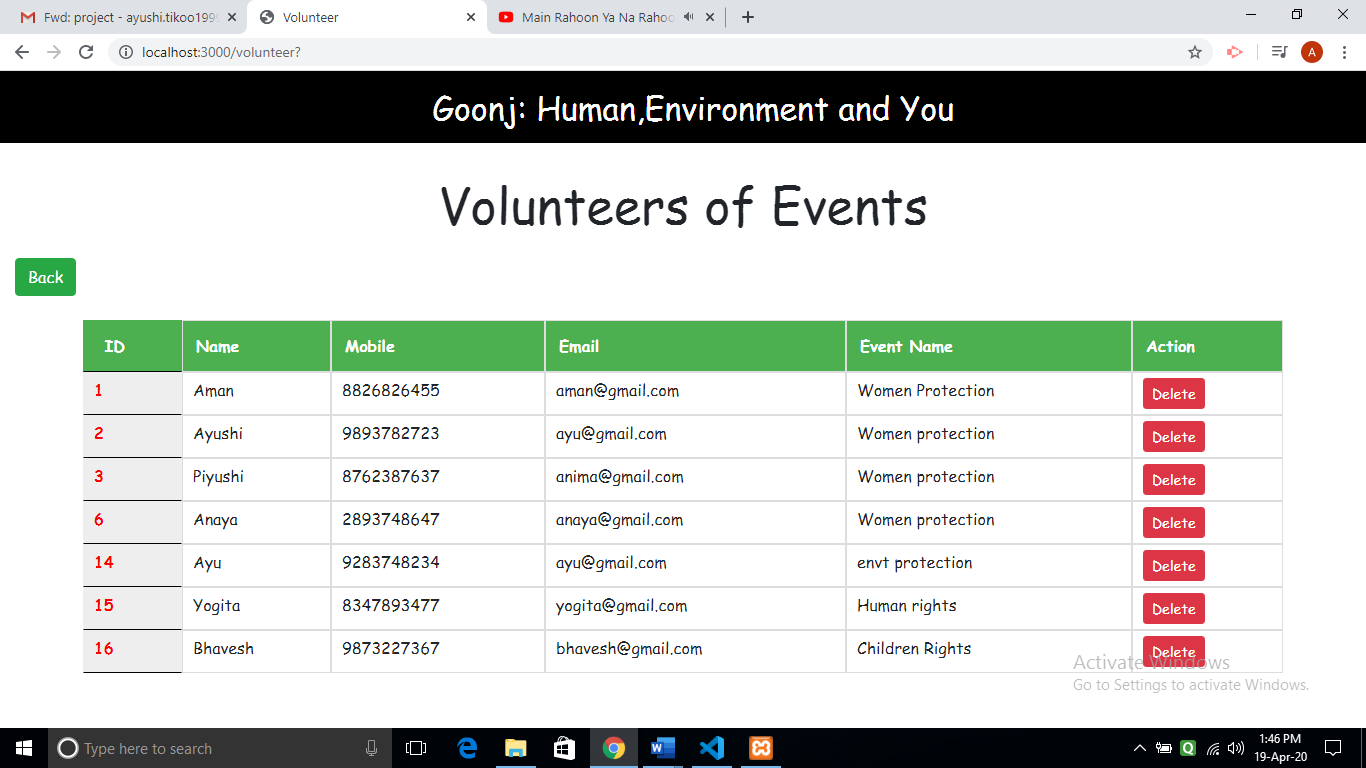
**View Donars (Others) :**



**View Feedback :**



**View Volunteers of event :**



**Conclusion**

We have successfully build the project of NGO System. We are able to build the project with success by fully implementing all the functionalities. First the user will have to register and then login into the system. After logging in he is displayed all the options like donating money or other things , giving feedback and participating in events. User is successfully able to donate money and other things without any problem. He can participate in any event by registering himself. Admin is able to manage user, view feedback, insert any new user, manage events, insert delete and update events and view donars who donated money and other materialistic things.

**Limitation and future extension**

* Users who want to volunteer for event won’t be able to give feedback about that event to admin
* We need to add system in the project in which admin should be able to send confirmation message to the volunteer.
* In future we could also provide tracking system to the donar by which he would be able to track the pickup request for its donation.
* We could also provide platform for the people to help them .

**Bibliography**

**References Used:**

<https://www.google.com/>

[http://stackoverflow.com](http://stackoverflow.com/)/

<https://www.zocdoc.com/>

<https://www.practo.com/>

<https://www.w3schools.com/>

<https://egov.ddit.ac.in/moodle/my/>