DBMS MINI PROJECT

NSS MANAGEMENT SYSTEM

Team no: 2

Member 1: NIKAASH T K (2022103035)

Member 2: ANAND KARTHIKEYAN S (2022103305)

Degree: B.E CSE

Semester: 4

Batch: P

Subject code: CS6106

Subject name: Database Management Systems

Submitted on: 23/05/2024

Submitted to: Dr.S.Renugadevi, Department of Computer Science and Engineering, CEG, Anna University.

PROBLEM STATEMENT:

To develop a comprehensive database management system for NSS units within a specific college (institution) to enhance organizational efficiency and foster community engagement.

GENERAL INTRODUCTION:

The National Service Scheme (NSS) is a student-centred program aimed at fostering social responsibility and community engagement among young individuals. Within educational institutions, NSS units play a crucial role in coordinating various social service activities and events. This project is a comprehensive database management system for managing the operations of NSS units within a specific college.

PROJECT ABSTRACT:

This project aims to streamline the management of NSS activities by designing and implementing a robust database system. The system will facilitate the management of records and details of the organization as well as efficient coordination of NSS activities. By providing a centralized platform for managing volunteer data and activities, the database system seeks to optimize resource allocation, improve communication, and strengthen the impact of NSS efforts within the college community.

The manual maintenance of records in an organization requires a lot of manual work and a lot of time. It is also difficult to

maintain hard copies of records. Due to maintenance limitations tracking historical data is not possible beyond an extent. And the retrieval of data is also not very effective. Even when data is stored in file systems the retrieval methods are deliberate. And thus, using a database to store all the data related to an organization which is NSS is an outstanding and apt solution. In a database, it is possible to retrieve data easily and effectively, modifications can be done effortlessly, and we can keep track of old data in archive for future reference. A perfect user interface for this DBMS project makes all the functionalities easier and even more convenient.

This DBMS will facilitate efficient organization, coordination, and tracking of volunteer activities, events, and participation within NSS units. Key requirements include registration and management of volunteers, assignment and management of roles within NSS units, tracking of volunteer activities, events, and participation, managing program officers overseeing NSS units, and integrating active alumni into NSS activities and events.

TECHNOLOGIES USED:

Frontend Technologies: HTML, CSS, JavaScript, React.js

Backend Technologies: Node.js, Express.js,

Database: PostgreSQL *(cloud database provided by Neon)*

This web application is built by frontend and backend technologies which are used by most professional developers of this time. *(Source: StackOverflow Developer Survey 2023)*

ENTITIES:

1. Unit (<u>Unit No</u>, P.O. ID, General Secretory, Joint Secretory, No of Volunteers, Manuals conducted)
2. Volunteers (<u>Volunteer ID</u>, Unit No, Name, DOB, Gender, Email ID, Joining Date, Year of Study, Password)
3. Program Officers (<u>P.O. ID</u>, Name, Designation, Department, Email ID)
4. Alumni (<u>Alumni ID</u>, Name, Unit No, Year of Passing, Email ID)
5. Manuals (<u>Manual ID</u>, Unit No, Theme, Description, Date, Duration, Location, Lead Organizer ID)

OTHER TABLES:

1. Contact (<u>Contact ID</u>, Phone no) #multivalued attribute
2. Participations (VolunteerID, Manual ID) #relationship
3. Volunteer Roles (Volunteer ID, Role ID) #relationship

4. Admin (<u>Admin ID</u>, Name, Password) #Overall admin

5. Roles (<u>Role ID</u>, Role Title) #role reference

CONSTRAINTS:

Foreign key constraints:

1. Unit (Unit No) — >Volunteer (Unit No), Manuals (Unit No), Alumni (Unit No)

2. Volunteers (Volunteer ID) — > Participation (Volunteer ID), Volunteer Roles (Volunteer ID), Manuals (Lead Organizer ID)

3. Program Officers (P.O ID) — > Unit (P.O ID)

Unique Constraints:

1. Volunteer (Email ID)

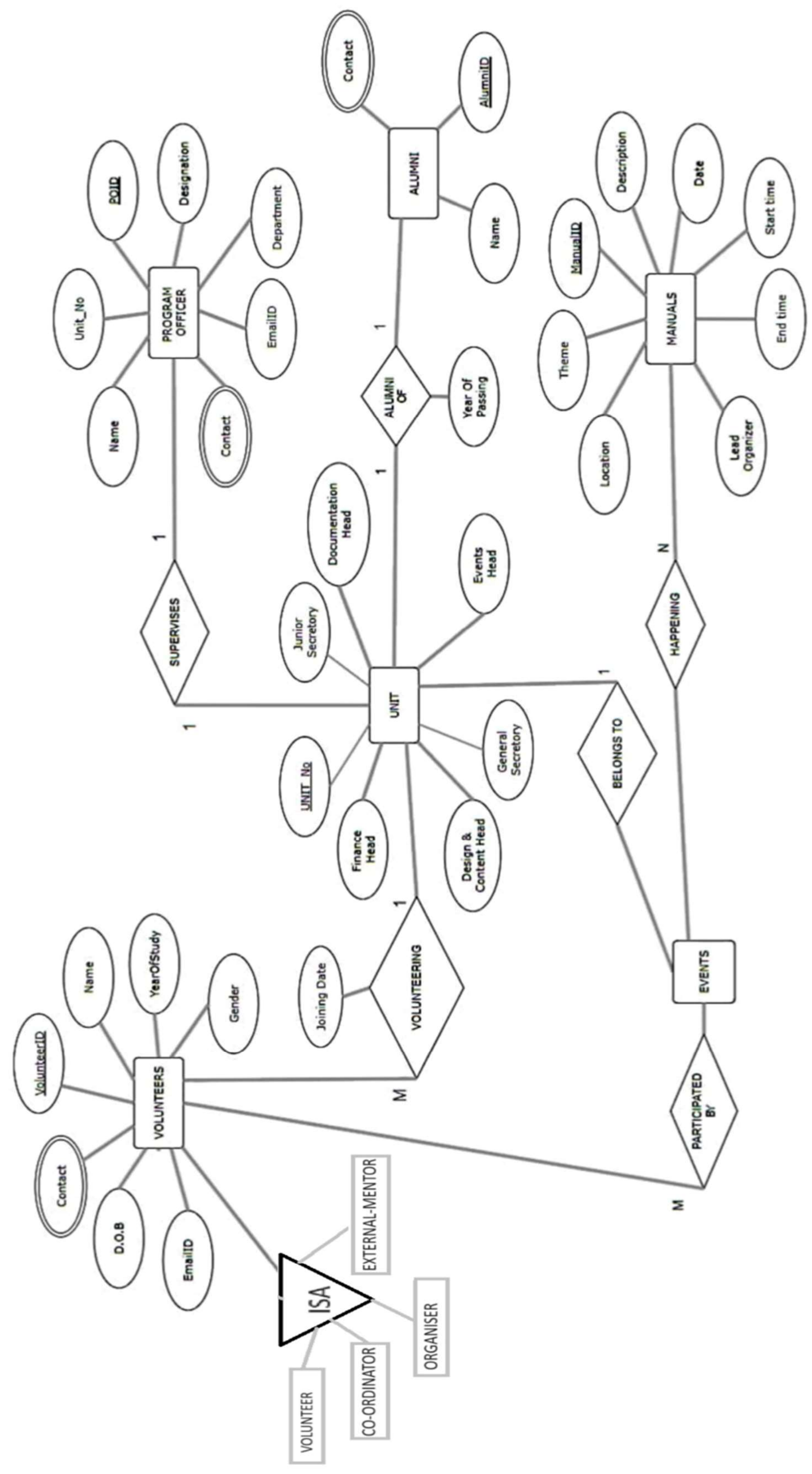
2. Program Officers (Email ID)

3. Alumni (Email ID)

4. Participations (VolunteerID, Manual ID)

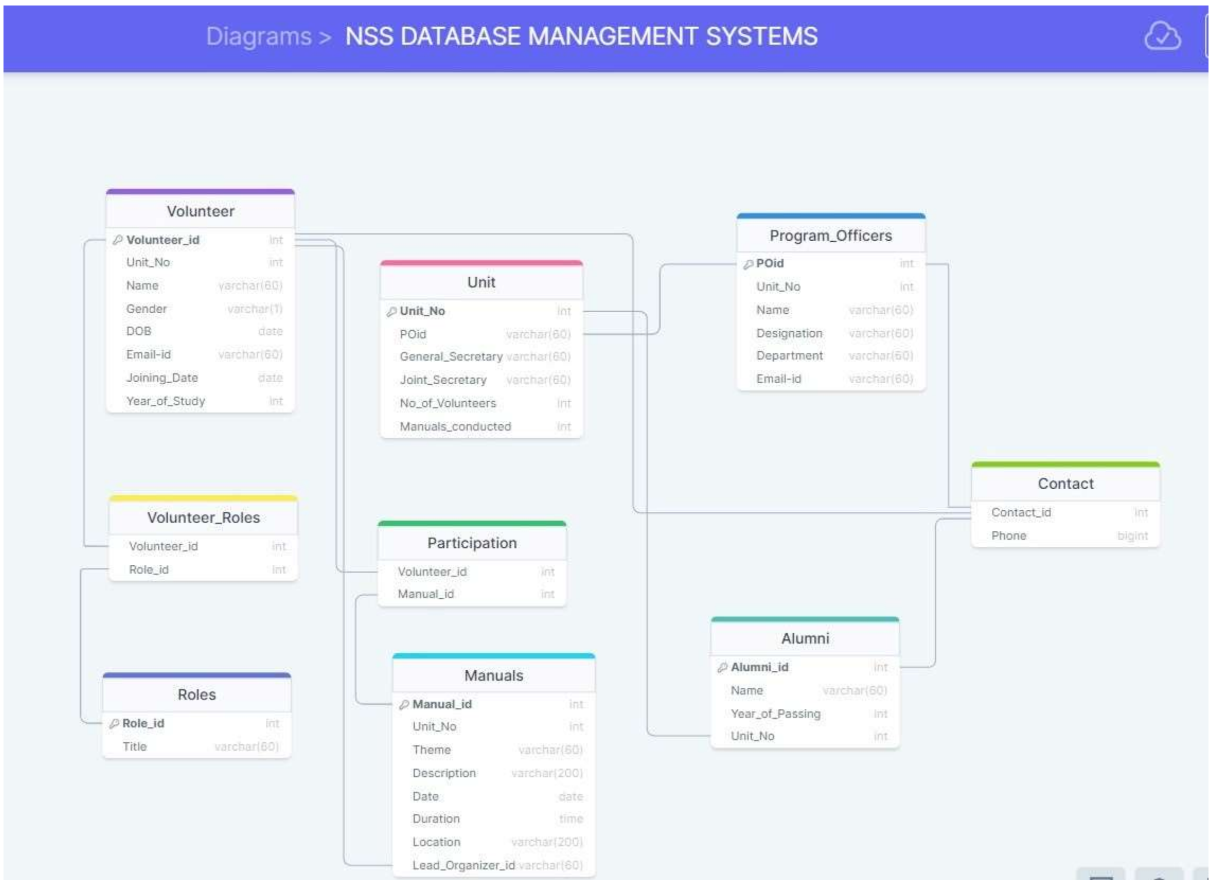
5. Volunteer Roles (Volunteer ID, Role ID)

ER DIAGRAM:



PROGRAM OFFICER
POID
Designation
Department
EmailID
Contact
Name
Unit_No
SUPERVISES
1
1
UNIT
UNIT_No
Finance Head
Design & Content Head
General Secretory
Junior Secretory
Documentation Head
Events Head
ALUMNI OF
Year Of Passing
1
1
ALUMNI
Contact
AlumniID
Name
MANUALS
ManualID
Description
Date
Start time
End time
Lead Organizer
Location
Theme
HAPPENING
N
BELONGS TO
1
EVENTS
PARTICIPATED BY
M
VOLUNTEERING
Joining Date
1
M
VOLUNTEERS
VolunteerID
Name
YearOfStudy
Gender
Contact
D.O.B
EmailID
ISA
EXTERNAL-MENTOR
ORGANISER
CO-ORDINATOR
VOLUNTEER

RELATIONAL DATABASE MODEL:



Volunteer	
Volunteer_id	int
Unit_No	int
Name	varchar(60)
Gender	varchar(1)
DOB	date
Email-id	varchar(60)
Joining_Date	date
Year_of_Study	int

Unit	
Unit_No	int
POid	varchar(60)
General_Secretary	varchar(60)
Joint_Secretary	varchar(60)
No_of_Volunteers	int
Manuals_conducted	int

Program_Officers	
POid	int
Unit_No	int
Name	varchar(60)
Designation	varchar(60)
Department	varchar(60)
Email-id	varchar(60)

Contact	
Contact_id	int
Phone	bigint

Volunteer_Roles	
Volunteer_id	int
Role_id	int

Participation	
Volunteer_id	int
Manual_id	int

Roles	
Role_id	int
Title	varchar(60)

Manuals	
Manual_id	int
Unit_No	int
Theme	varchar(60)
Description	varchar(200)
Date	date
Duration	time
Location	varchar(200)
Lead_Organizer_id	varchar(60)

Alumni	
Alumni_id	int
Name	varchar(60)
Year_of_Passing	int
Unit_No	int

DATABASE ORGANIZATION:

Functions:

1)
```
CREATE OR REPLACE FUNCTION move_to_alumni()
RETURNS TRIGGER AS $$
DECLARE
  years_out INT;
BEGIN
  years_out := EXTRACT(YEAR FROM CURRENT_DATE);
  IF NEW."current_year" > 4 THEN
    INSERT INTO alumni ("alumni_id", "year_of_passing", "name", "unit_no","email")
    VALUES (NEW."volunteer_id", years_out, NEW."name", NEW."unit_no",NEW."email");
    delete from  volunteers where volunteer_id = new.volunteer_id;
  END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

2)
```
CREATE OR REPLACE FUNCTION update_no_of_volunteers()
RETURNS TRIGGER AS $$
BEGIN
 IF (TG_OP = 'INSERT') THEN
  UPDATE unit SET no_of_volunteers = no_of_volunteers + 1 WHERE unit_no = NEW.unit_no;
 ELSIF (TG_OP = 'DELETE') THEN
  UPDATE unit SET no_of_volunteers = no_of_volunteers - 1 WHERE unit_no = OLD.unit_no;
 ELSIF (TG_OP = 'UPDATE') THEN
  IF (NEW.unit_no <> OLD.unit_no) THEN
   UPDATE unit SET no_of_volunteers = no_of_volunteers + 1 WHERE unit_no =
NEW.unit_no;
   UPDATE unit SET no_of_volunteers = no_of_volunteers - 1 WHERE unit_no = OLD.unit_no;
```

```
    END IF;
  END IF;
  RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

3) CREATE OR REPLACE FUNCTION assign_unit_no() RETURNS TRIGGER AS $$

```
DECLARE
    max_unit_no_m INT;
    max_unit_no_f INT;
    new_volunteerid INT;
    year_of_study INT;
    year_dob INT;
BEGIN
    SELECT unit_no INTO max_unit_no_m FROM volunteers WHERE volunteer_id = (SELECT
MAX(volunteer_id) FROM volunteers WHERE gender = 'M');
    SELECT unit_no INTO max_unit_no_f FROM volunteers WHERE volunteer_id = (SELECT
MAX(volunteer_id) FROM volunteers WHERE gender = 'F');
    SELECT MAX(volunteer_id) + 1 INTO new_volunteerid FROM volunteers;

  IF NEW.gender = 'M' AND ((max_unit_no_m IS NULL) OR (max_unit_no_m = 3)) THEN
        NEW.unit_no := 1;
    ELSIF NEW.gender = 'M' THEN
        NEW.unit_no := max_unit_no_m + 1;

    ELSIF NEW.gender = 'F' AND ((max_unit_no_f) IS NULL OR (max_unit_no_f = 3)) THEN
        NEW.unit_no := 1;
    ELSIF NEW.gender = 'F' THEN
        NEW.unit_no := max_unit_no_f + 1;
    END IF;
    NEW.joining_date := TO_CHAR(CURRENT_DATE, 'YYYY-MM-DD');
```

```
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

4) CREATE OR REPLACE FUNCTION role_id()
RETURNS TRIGGER AS $$
begin
  insert into volunteer_roles(volunteer_id,role_id) values (NEW.volunteer_id , 1);
RETURN NULL;
END;
$$ LANGUAGE plpgsql;

5) CREATE OR REPLACE FUNCTION volunteer_remove()
RETURNS TRIGGER AS $$
BEGIN
       DELETE FROM volunteers where volunteer_id=NEW.alumni_id;
       RETURN NULL;
END;
$$ LANGUAGE plpgsql;

6) CREATE OR REPLACE FUNCTION remove_roleid()
RETURNS TRIGGER AS $$
begin
   delete from volunteer_roles where volunteer_id=old.volunteer_id;
   delete from participations where volunteer_id=old.volunteer_id;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

7) CREATE OR REPLACE FUNCTION manualid_insert()
RETURNS TRIGGER AS $$
```

```
declare
t_mid int;
BEGIN
    select max(manual_id)+1 into t_mid from manuals ;
    NEW.manual_id := t_mid;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

8)
   ```
CREATE OR REPLACE FUNCTION update_manuals_conducted()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE unit
    SET manuals_conducted = manuals_conducted + 1
    WHERE unit_no = NEW.unit_no;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

9)
   ```
CREATE OR REPLACE FUNCTION manual_delete_after()
RETURNS TRIGGER AS $$
BEGIN
    delete from participations where manual_id= OLD.manual_id;
    update unit set manuals_conducted= manuals_conducted-1
       where unit_no = ( select unit_no from manuals where manual_id = OLD.manual_id);
    RETURN OLD;
END;
$$
LANGUAGE plpgsql;
```

10)
```
CREATE OR REPLACE FUNCTION check_current_year_before_insert()
RETURNS TRIGGER AS $$
BEGIN
   IF TG_OP = 'INSERT' AND NEW.current_year >= 5 THEN
      RAISE EXCEPTION 'current_year must be less than 4 during insert';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Triggers:

1)
```
CREATE or replace TRIGGER move_to_alumni
AFTER UPDATE ON volunteers
FOR EACH ROW
EXECUTE FUNCTION move_to_alumni();
```

2)
```
CREATE or replace TRIGGER update_no_of_members_trigger
AFTER INSERT OR DELETE OR UPDATE ON volunteers
FOR EACH ROW
EXECUTE FUNCTION update_no_of_volunteers();
```

3)
```
CREATE OR REPLACE TRIGGER assign_unit_no
BEFORE INSERT ON volunteers
FOR EACH ROW
EXECUTE FUNCTION assign_unit_no();
```

```
4) CREATE or replace TRIGGER role_id_insert

AFTER insert ON volunteers

FOR EACH ROW

EXECUTE FUNCTION role_id();


5) create or replace trigger volunteer_remove

after insert ON alumni

FOR EACH ROW

EXECUTE FUNCTION volunteer_remove();


6) CREATE or replace TRIGGER remove_roleid

before delete ON volunteers

FOR EACH ROW

EXECUTE FUNCTION remove_roleid();


7) CREATE OR replace TRIGGER manualid_insert

BEFORE INSERT ON manuals

FOR EACH ROW

EXECUTE FUNCTION manualid_insert();


8) CREATE OR replace TRIGGER update_manuals_conducted_trigger

AFTER INSERT ON manuals

FOR EACH ROW

EXECUTE FUNCTION update_manuals_conducted();


9) CREATE OR replace TRIGGER manual_delete_after

after delete ON manuals

FOR EACH ROW

EXECUTE FUNCTION manual_delete_after();
```

```
10) CREATE or replace TRIGGER check_current_year_trigger
BEFORE INSERT ON volunteers
FOR EACH ROW
EXECUTE FUNCTION check_current_year_before_insert();
```

Cursor:

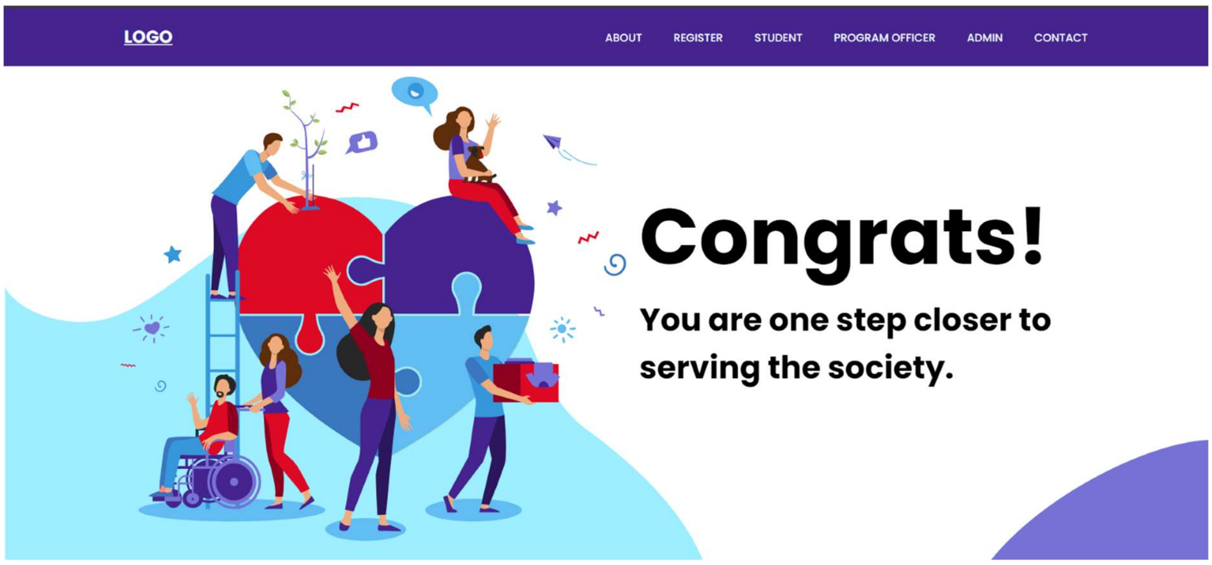
1)

```
CREATE OR REPLACE FUNCTION fetch_manuals_for_unit(unit_no_param INT)
RETURNS SETOF manuals AS $$
DECLARE
  manual_record manuals%ROWTYPE;
  manual_cursor CURSOR FOR
    SELECT * FROM manuals WHERE unit_no = unit_no_param ORDER BY manual_id ASC;
BEGIN
  FOR manual_record IN manual_cursor LOOP
    RETURN NEXT manual_record;
  END LOOP;
  RETURN;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM fetch_manuals_for_unit(1);
```

IMPLEMENTATION:

Home Page:

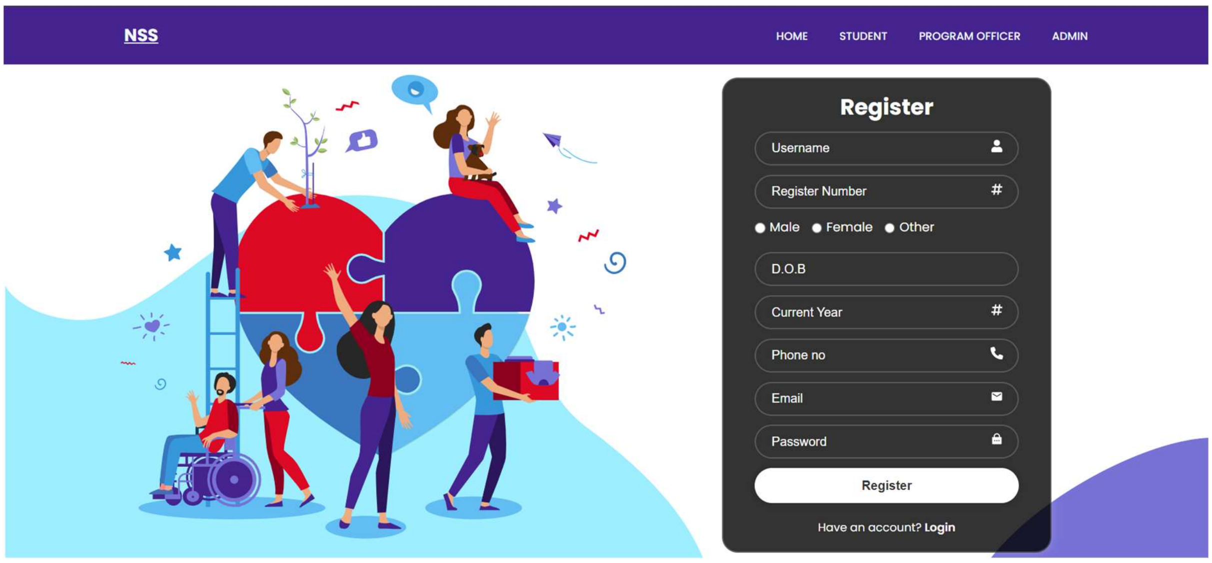


Congrats!

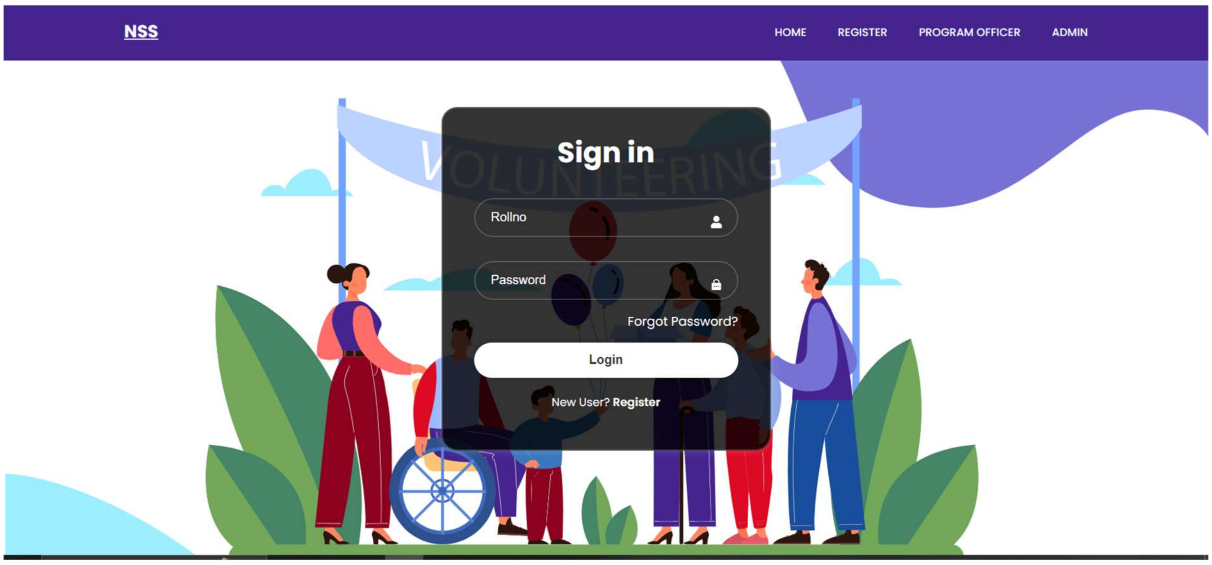
You are one step closer to serving the society.

SEP

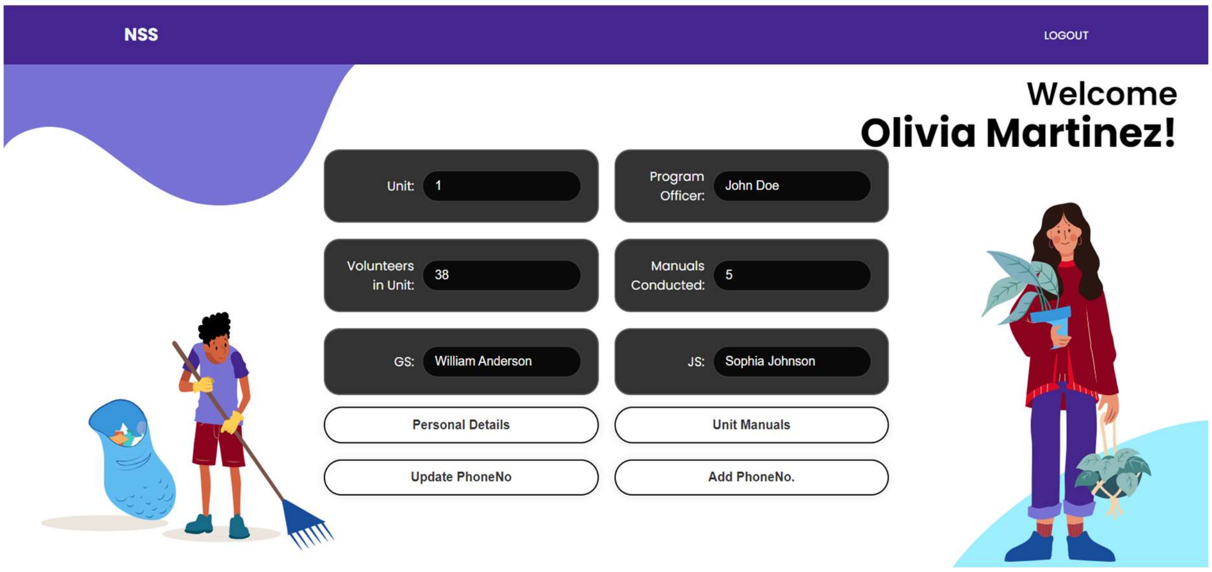
Register Page:



Login Page:



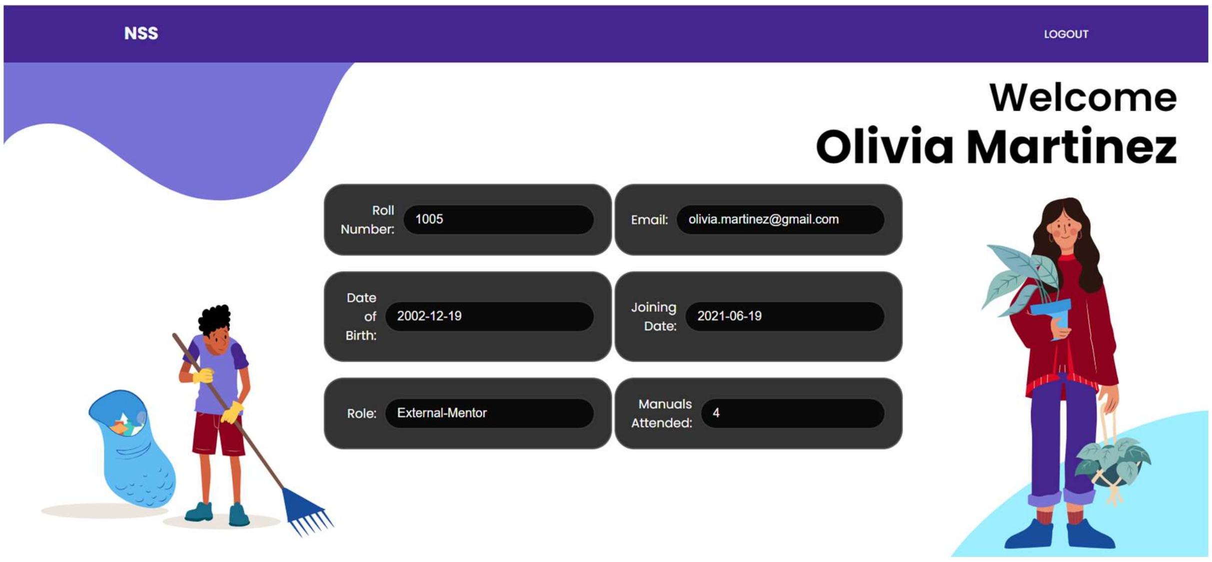
Login Response Page:



Welcome
Olivia Martinez!

Unit: 1
Program Officer: John Doe
Volunteers in Unit: 38
Manuals Conducted: 5
GS: William Anderson
JS: Sophia Johnson

Personal Details :



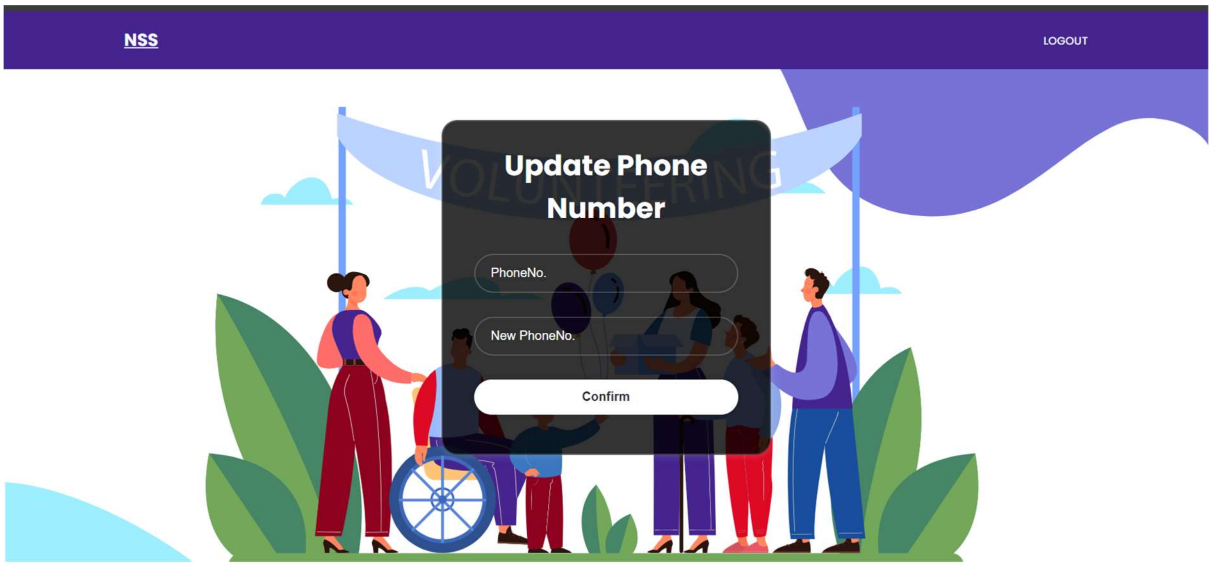
NSS

LOGOUT

Welcome
Olivia Martinez

Roll Number: 1005	Email: olivia.martinez@gmail.com
Date of Birth: 2002-12-19	Joining Date: 2021-06-19
Role: External-Mentor	Manuals Attended: 4

Update/Add phone number :



NSS

LOGOUT

Update Phone Number

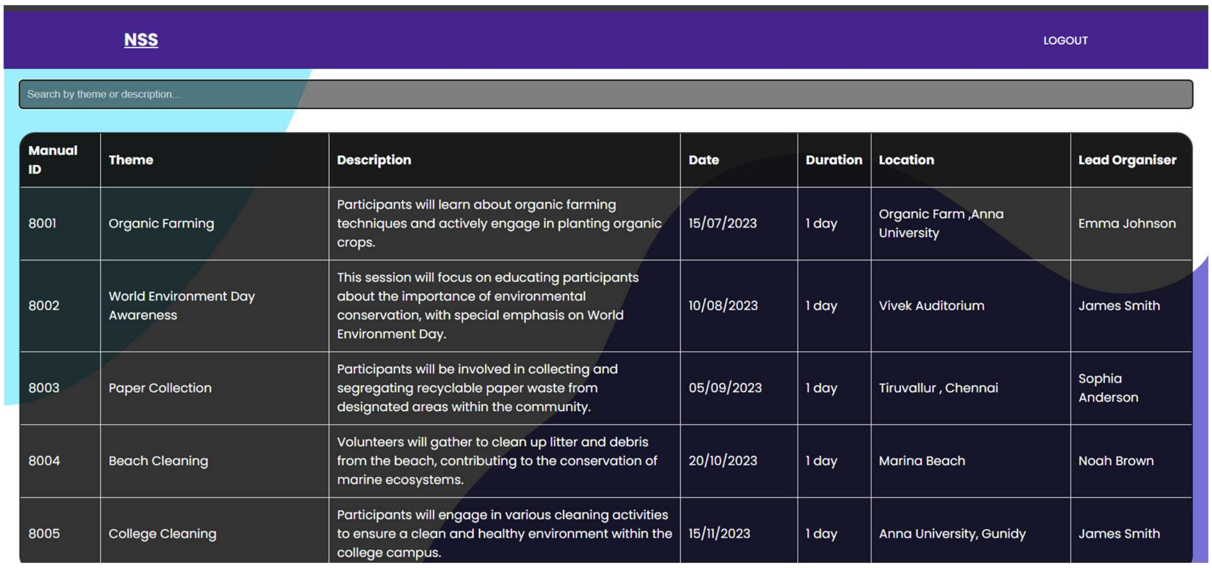
PhoneNo.

New PhoneNo.

Confirm

SEP

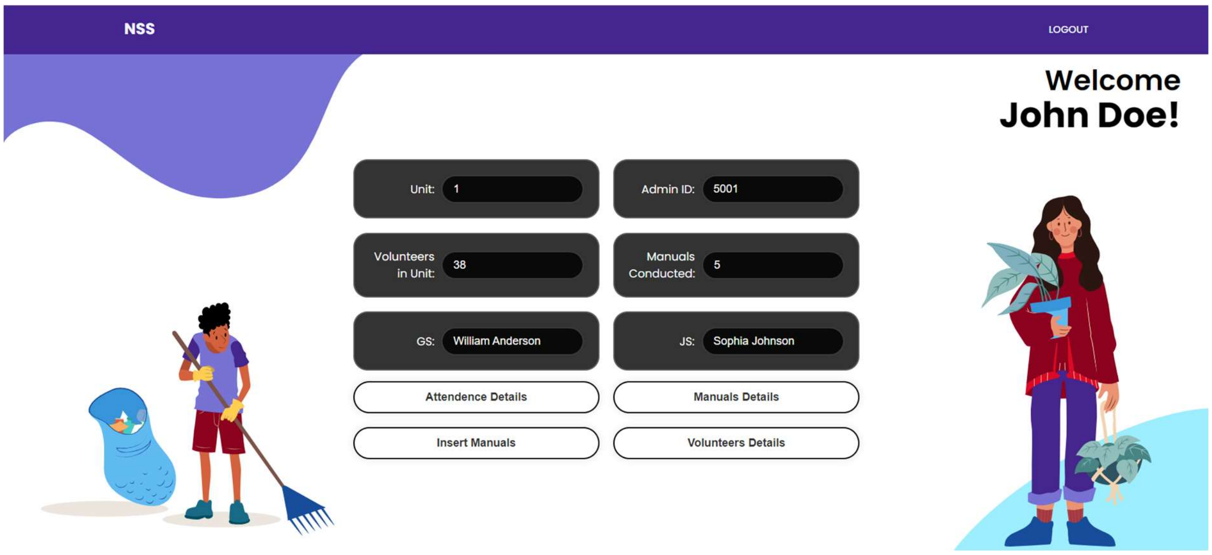
Manual Details:



Manual ID	Theme	Description	Date	Duration	Location	Lead Organiser
8001	Organic Farming	Participants will learn about organic farming techniques and actively engage in planting organic crops.	15/07/2023	1 day	Organic Farm ,Anna University	Emma Johnson
8002	World Environment Day Awareness	This session will focus on educating participants about the importance of environmental conservation, with special emphasis on World Environment Day.	10/08/2023	1 day	Vivek Auditorium	James Smith
8003	Paper Collection	Participants will be involved in collecting and segregating recyclable paper waste from designated areas within the community.	05/09/2023	1 day	Tiruvallur , Chennai	Sophia Anderson
8004	Beach Cleaning	Volunteers will gather to clean up litter and debris from the beach, contributing to the conservation of marine ecosystems.	20/10/2023	1 day	Marina Beach	Noah Brown
8005	College Cleaning	Participants will engage in various cleaning activities to ensure a clean and healthy environment within the college campus.	15/11/2023	1 day	Anna University, Gunidy	James Smith

SEP

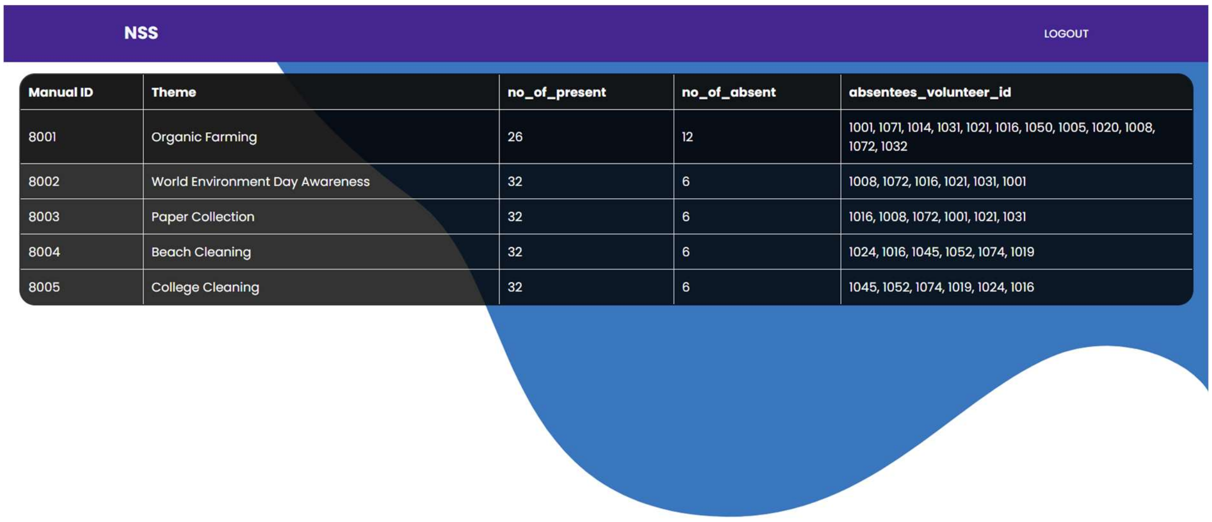
PO Login Response :



Unit: 1	Admin ID: 5001
Volunteers in Unit: 38	Manuals Conducted: 5
GS: William Anderson	JS: Sophia Johnson

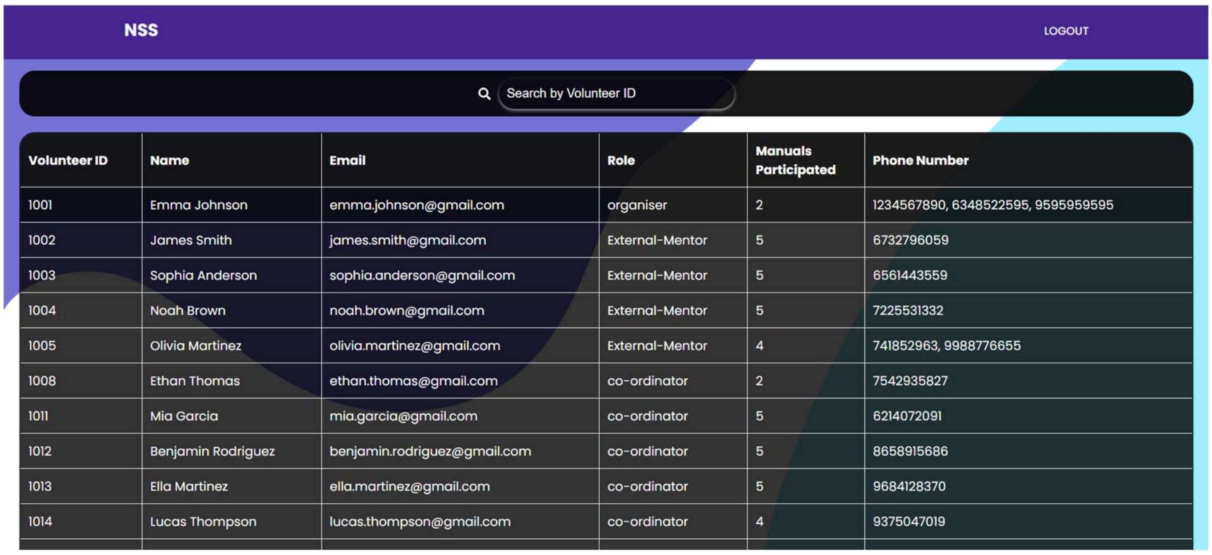
SEP

Attendance Details:



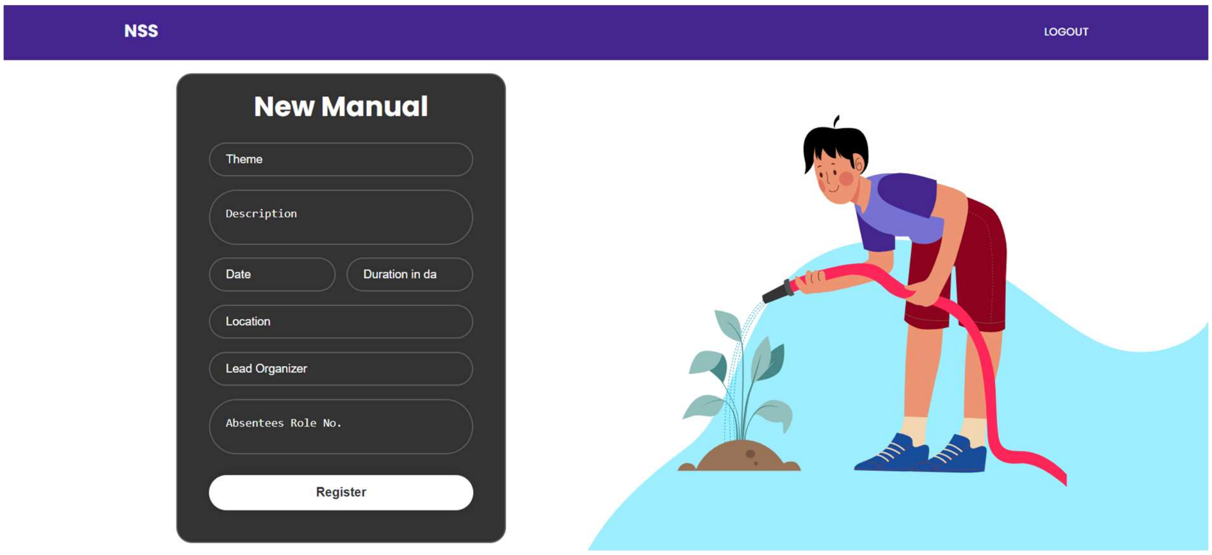
Manual ID	Theme	no_of_present	no_of_absent	absentees_volunteer_id
8001	Organic Farming	26	12	1001, 1071, 1014, 1031, 1021, 1016, 1050, 1005, 1020, 1008, 1072, 1032
8002	World Environment Day Awareness	32	6	1008, 1072, 1016, 1021, 1031, 1001
8003	Paper Collection	32	6	1016, 1008, 1072, 1001, 1021, 1031
8004	Beach Cleaning	32	6	1024, 1016, 1045, 1052, 1074, 1019
8005	College Cleaning	32	6	1045, 1052, 1074, 1019, 1024, 1016

Volunteers Details:



Volunteer ID	Name	Email	Role	Manuals Participated	Phone Number
1001	Emma Johnson	emma.johnson@gmail.com	organiser	2	1234567890, 6348522595, 9595959595
1002	James Smith	james.smith@gmail.com	External-Mentor	5	6732796059
1003	Sophia Anderson	sophia.anderson@gmail.com	External-Mentor	5	6561443559
1004	Noah Brown	noah.brown@gmail.com	External-Mentor	5	7225531332
1005	Olivia Martinez	olivia.martinez@gmail.com	External-Mentor	4	741852963, 9988776655
1008	Ethan Thomas	ethan.thomas@gmail.com	co-ordinator	2	7542935827
1011	Mia Garcia	mia.garcia@gmail.com	co-ordinator	5	6214072091
1012	Benjamin Rodriguez	benjamin.rodriguez@gmail.com	co-ordinator	5	8658915686
1013	Ella Martinez	ella.martinez@gmail.com	co-ordinator	5	9684128370
1014	Lucas Thompson	lucas.thompson@gmail.com	co-ordinator	4	9375047019

Insert Manuals:



NSS LOGOUT

New Manual

Theme

Description

Date | Duration in da

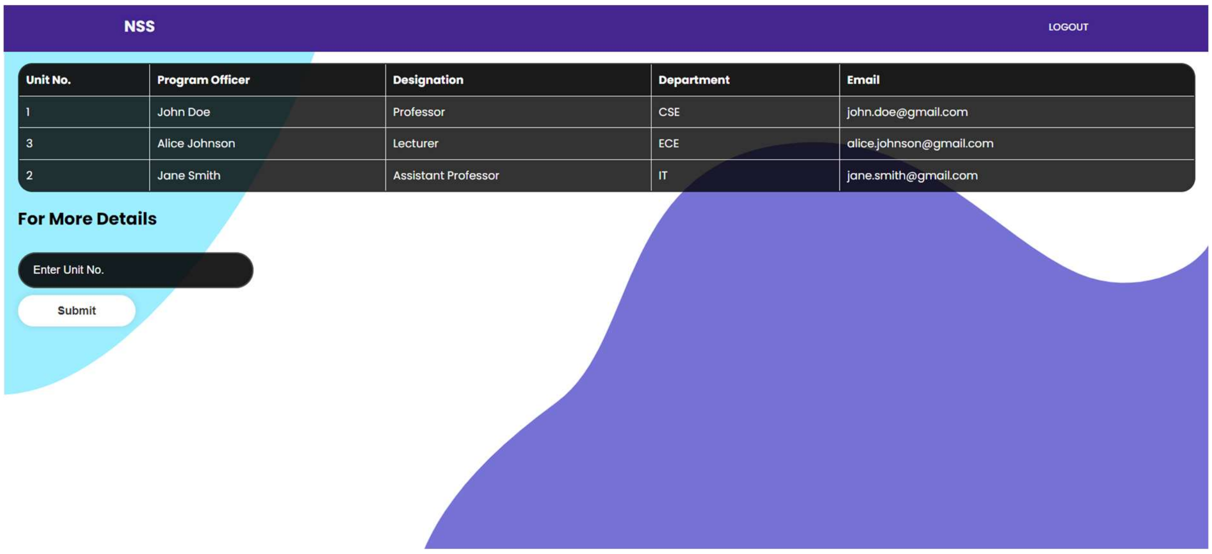
Location

Lead Organizer

Absentees Role No.

Register

Admin Login Response:



NSS LOGOUT

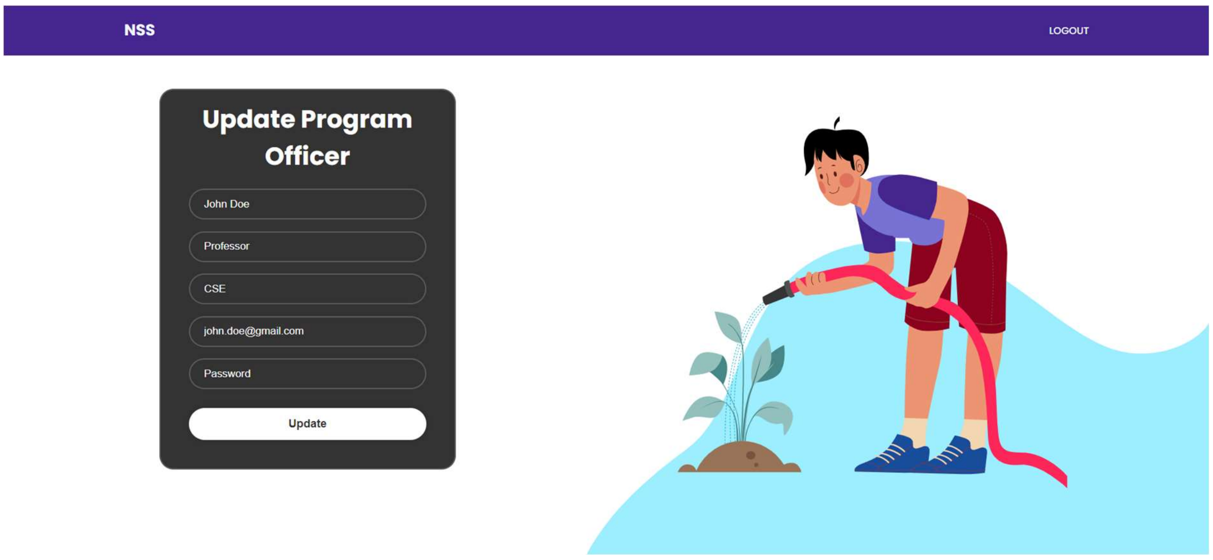
Unit No.	Program Officer	Designation	Department	Email
1	John Doe	Professor	CSE	john.doe@gmail.com
3	Alice Johnson	Lecturer	ECE	alice.johnson@gmail.com
2	Jane Smith	Assistant Professor	IT	jane.smith@gmail.com

For More Details

Enter Unit No.

Submit

Update PO:



John Doe

Professor

CSE

john.doe@gmail.com

Password

CONCLUSION:

In conclusion, the NSS organization database project aims to enhance the overall effectiveness of social service initiatives by NSS units within a college. Future enhancements may include additional features such as Personnel Announcements and Q&A /Feedback Options. It can also be performed on a broad scale by integrating with other colleges which can improve the overall output of NSS initiatives.