# **Project Management System For IT**

***DBMS Project Report Submitted in partial fulfilment of the***

***Requirements for the award of the Degree of* BACHELOR OF ENGINEERING** IN

**INFORMATION TECHNOLOGY**

#### BY

**<K.VURVIK><1602-22-737-146>**

**<P.LAHARI><1602-22-737-147>**

Under the guidance of Mrs. Sreelakshmi



Department of InformationTechnology Vasavi Collegeof Engineering (Autonomous)

(Affiliated toOsmania University) Ibrahimbagh, Hyderabad-31 2020

## DECLARATION BY THE CANDIDATE

We, **K.Vurvik** and **P.Lahari**, bearing hall ticket numbers, **1602-22-737-146** and **1602-22-737-147**, hereby declare that the project report entitled **” COLLEGE PROJECTS MANAGEMENT SYSTEM”** Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**K.Vurvik<1602-22-737-146>**

**P.Lahari<1602-22-737-147>**

**(Faculty In-Charge) (Head,Dept of It)**

## **Introduction:**

Project management systems facilitate collaboration, streamline workflows, and enhance productivity across teams involved in complex projects. By centralizing project-related information and providing real-time updates, these systems empower stakeholders to make informed decisions and keep projects on track.

Key features typically include task management, resource allocation, progress tracking, reporting, and communication tools—all of which contribute to improved project outcomes and stakeholder satisfaction.

# **Objectives:**

The primary objectives of implementing this project management system are:

**Enhanced Project Visibility:** Provide stakeholders with real-time visibility into project status, milestones, and review progress. Enable efficient tracking of review outcomes and actions required for project advancement.

**Improved Collaboration:** Facilitate seamless communication and collaboration among project teams and reviewers. Enable file sharing, discussion forums, and notification systems to keep stakeholders informed and engaged throughout the project lifecycle.

**Data Integrity and Security:** Implement robust data integrity measures and access controls to safeguard project information. Ensure that only authorized users have access to specific project details and review outcomes.

# **Conclusion:**

By implementing this project management system with sequential reviews, organizations can enhance project governance, improve decision-making, and foster collaboration among stakeholders. The following sections will detail the technical implementation steps, database design considerations, and sample queries to demonstrate the functionality of the system.

## **AIM AND PRIORITY OF THE PROJECt:**

To create a web based gui management system for projects made by students in a particular college. All the values are to be updated and handled in the database using dbsqlite connectivity

## **ARCHITECTURE AND TECHNOLOGY USED :**

### **Frontend Technologies:**

1. **HTML**:
   * No specific version as HTML5 is the current standard.
2. **CSS**:
   * Use CSS3 for the latest features and standards.
3. **Bootstrap**:
   * **Latest Version**: Bootstrap 5.3
   * **Requirements**:
     + A modern browser that supports CSS3 and HTML5.
     + jQuery is not required for Bootstrap 5.
4. **JavaScript**:
   * **Latest Version**: ECMAScript 2024 (ES13)
   * **Requirements**:
     + Modern browsers supporting the latest JavaScript features.
     + Optional: Babel can be used to ensure compatibility with older browsers.

### Backend Technologies:

1. **Django**:
   * **Latest Version**: Django 4.2 (LTS)
   * **Requirements**:
     + Python 3.8 or newer.
     + Virtual environment (optional but recommended for dependency management).
2. **SQLite**:
   * **Latest Version**: SQLite 3.43.1
   * **Requirements**:
     + Included with Python by default.
     + Suitable for lightweight database needs and development environments.

### Additional Dependencies:

* **Django REST framework** (if building an API):
  + **Latest Version**: Django REST framework 3.15
  + **Requirements**: Compatible with Django 4.2
* **Django Templates**:
  + Use built-in Django template engine or third-party options like Jinja2.

# Installation and Setup:

Setting Up Django with SQLite:

Python Installation: Ensure Python 3.8 or later is installed. You can download it from python.org.

Django Installation:

pip install django

Creating a Django Project:

django-admin startproject dbmsproject

cd dbmsproject

Setting Up SQLite: SQLite is included with Python, so no additional setup is required. Django uses SQLite by default.

Starting the Development Server:

python manage.py runserver

Adding Bootstrap to Your Project:

Include Bootstrap in your HTML files via CDN:

<link href="https://stackpath.bootstrapcdn.com/bootstrap/5.3.0/css/bootstrap.min.css" rel="stylesheet">

<script src="https://stackpath.bootstrapcdn.com/bootstrap/5.3.0/js/bootstrap.bundle.min.js"></script>

Alternatively, download Bootstrap and include the files locally.

JavaScript Integration:

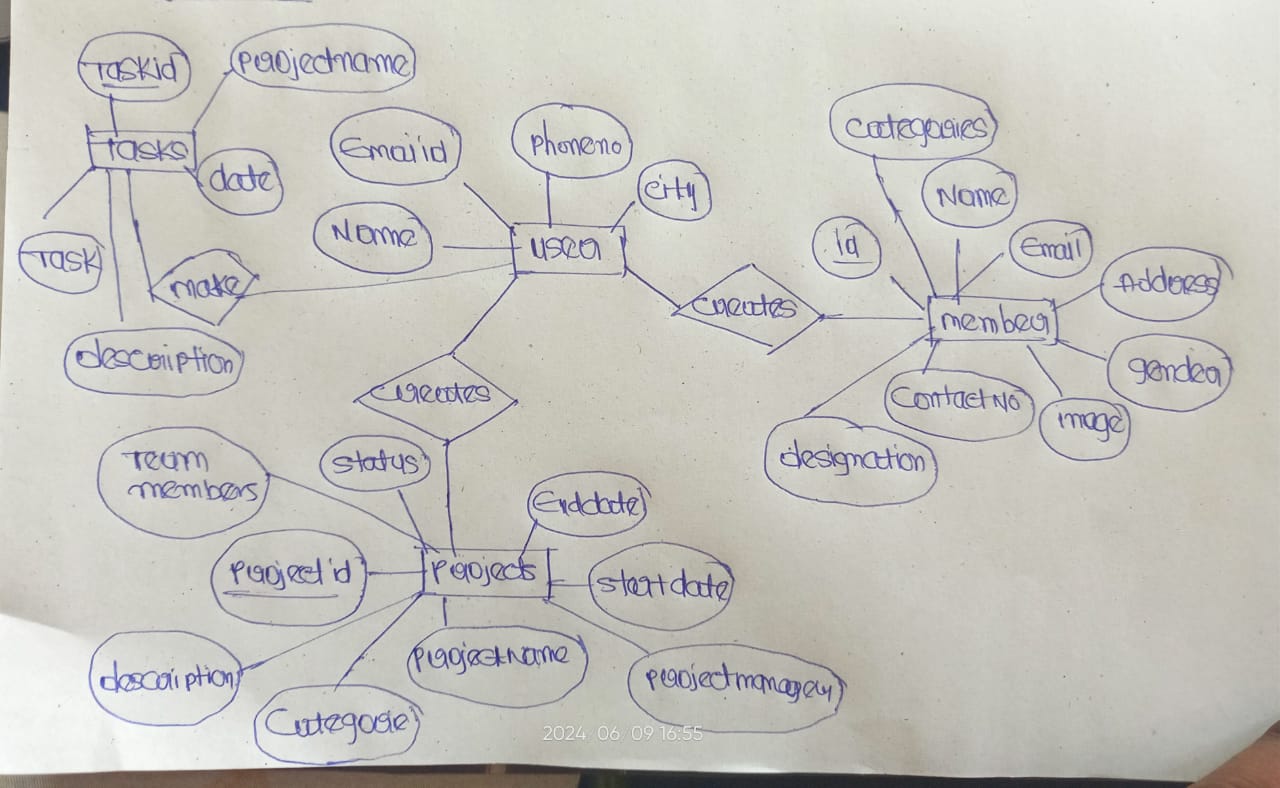
Include your custom JavaScript files in your project. Ensure they are referenced in your HTML templates.

<script src="{% static 'js/custom.js' %}"></script>

For advanced JavaScript functionality, consider using frameworks or libraries such as React, Angular, or Vue.js.

# DESIGN

Entity Relationship Diagram:



Here user can be admin he can manage all the projects and members.he/she can assign tasks to the projects so that we can make anlysis

# Project Structure:

dbmsproject/

│

├── manage.py

├── dbmsproject/

│ ├── \_\_init\_\_.py

│ ├── settings.py

│ ├── urls.py

│ ├── wsgi.py

│ └── asgi.py

├── pms/

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── tests.py

│ ├── views.py

│ ├── templates/

│ ├── static/

│ │ ├── css/

│ │ │ └── styles.css

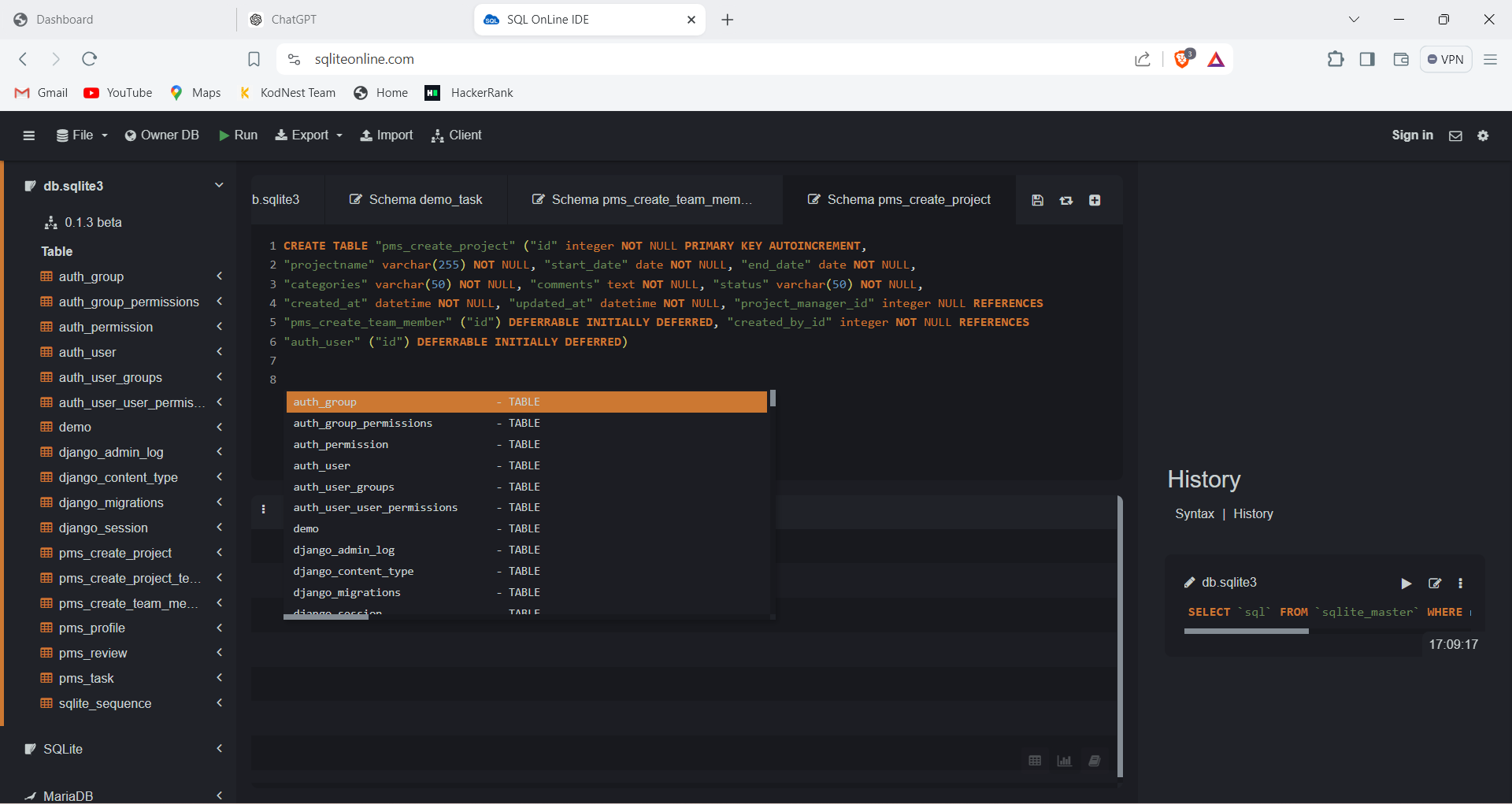
│ │ ├── js/

│ │ │ └── custom.js

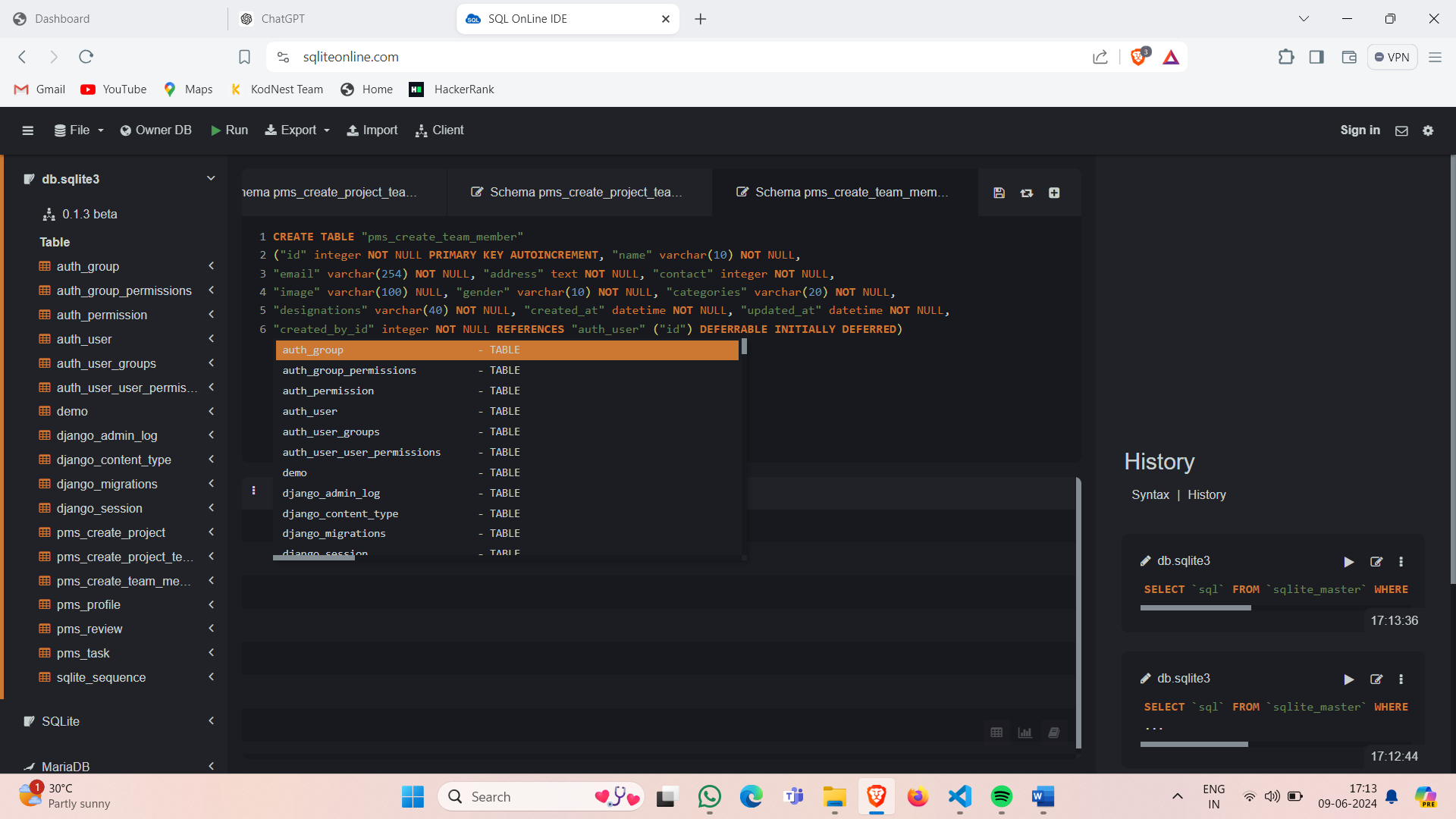
│ │ └── images/

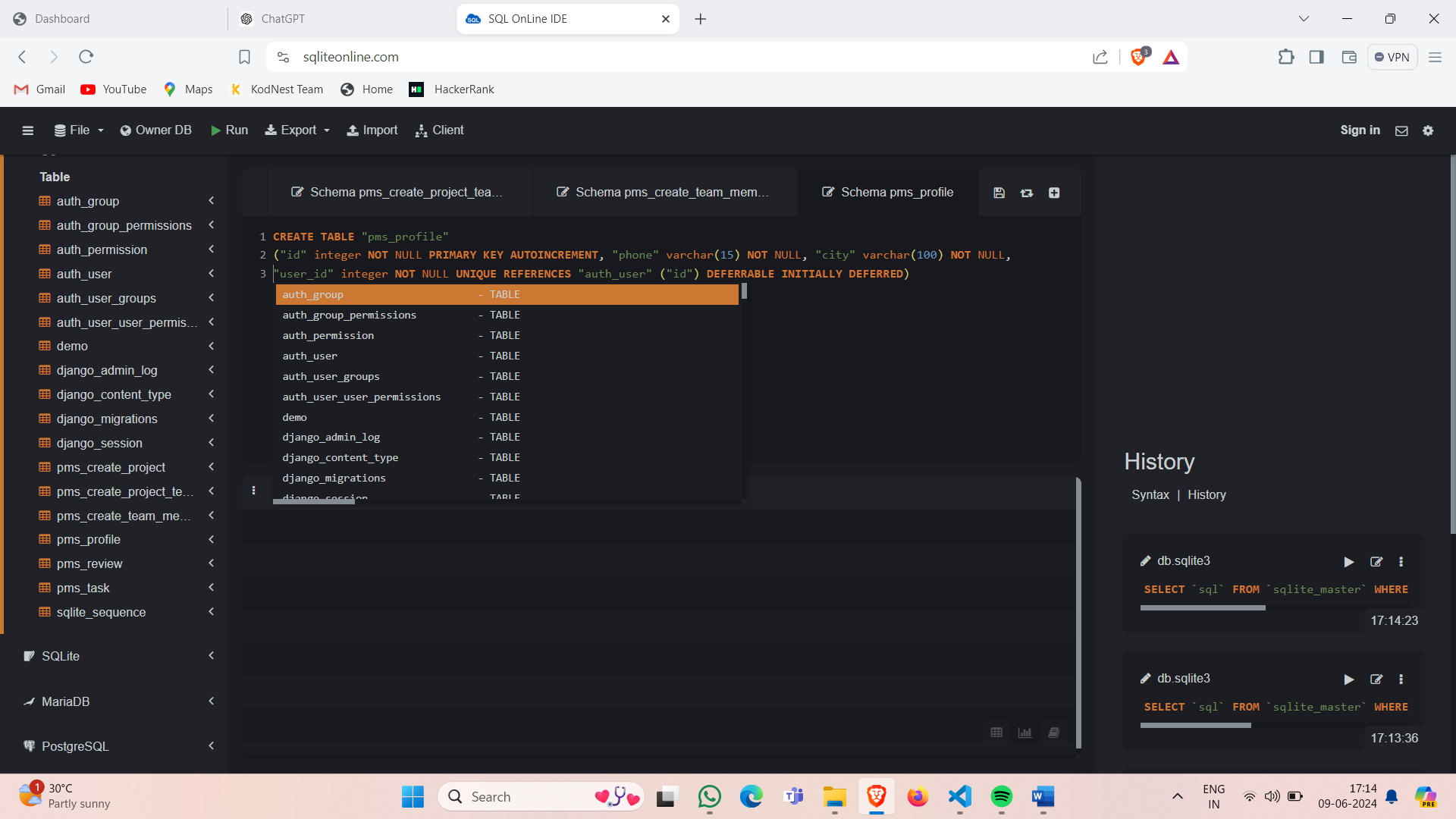
│ └── migrations/

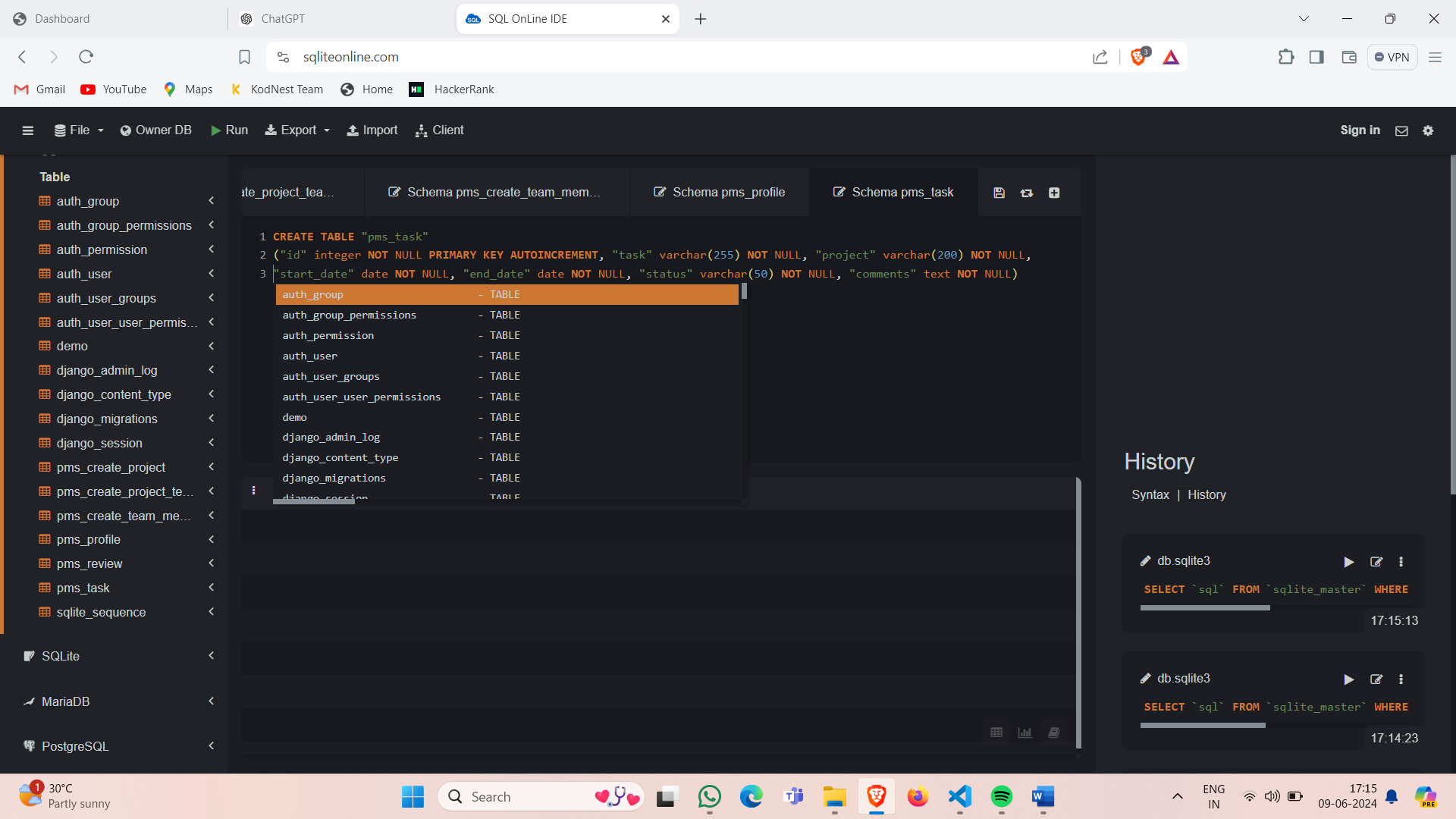
# **DDL and DML Commands,Queries(database design):**











# **Reducing into Tables:**

#### **Users Table:**

* Store user information including usernames, emails, roles, and project assignments.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| UserID | INT | Primary key, unique identifier for users |
| Username | VARCHAR(20) | User's username |
| Role | VARCHAR(20) | User's role (e.g., teamlead, teammember) |
| Email | VARCHAR(30) | User's email address |
| ProjectID | INT | Foreign key referencing Project table |

#### **Project Table:**

* Store project details with associated users.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| ProjectID | INT | Primary key, unique identifier for projects |
| ProjectName | VARCHAR(20) | Name of the project |
| StartDate | DATE | Start date of the project |
| EndDate | DATE | End date of the project |
| Status | VARCHAR(40) | Status of the project (e.g., notyet, completed) |
| Description | VARCHAR(100) | Description of the project |
| Domain | VARCHAR(20) | Domain of the project (e.g., web-dev, mysql) |
| UserID | INT | Foreign key referencing Users table (project lead/teamlead) |

#### **Tasks Table:**

* Manage project tasks with deadlines and status.

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| TaskID | INT | Primary key, unique identifier for tasks |
| ProjectID | INT | Foreign key referencing Project table |
| Description | VARCHAR(100) | Description of the task |
| TaskName | VARCHAR(25) | Name/Title of the task |
| Deadline | DATE | Deadline for completing the task |
| Status | VARCHAR(20) | Status of the task (e.g., pending, in progress) |

# **Relations:**

1. **Users to Projects (One-to-Many):**

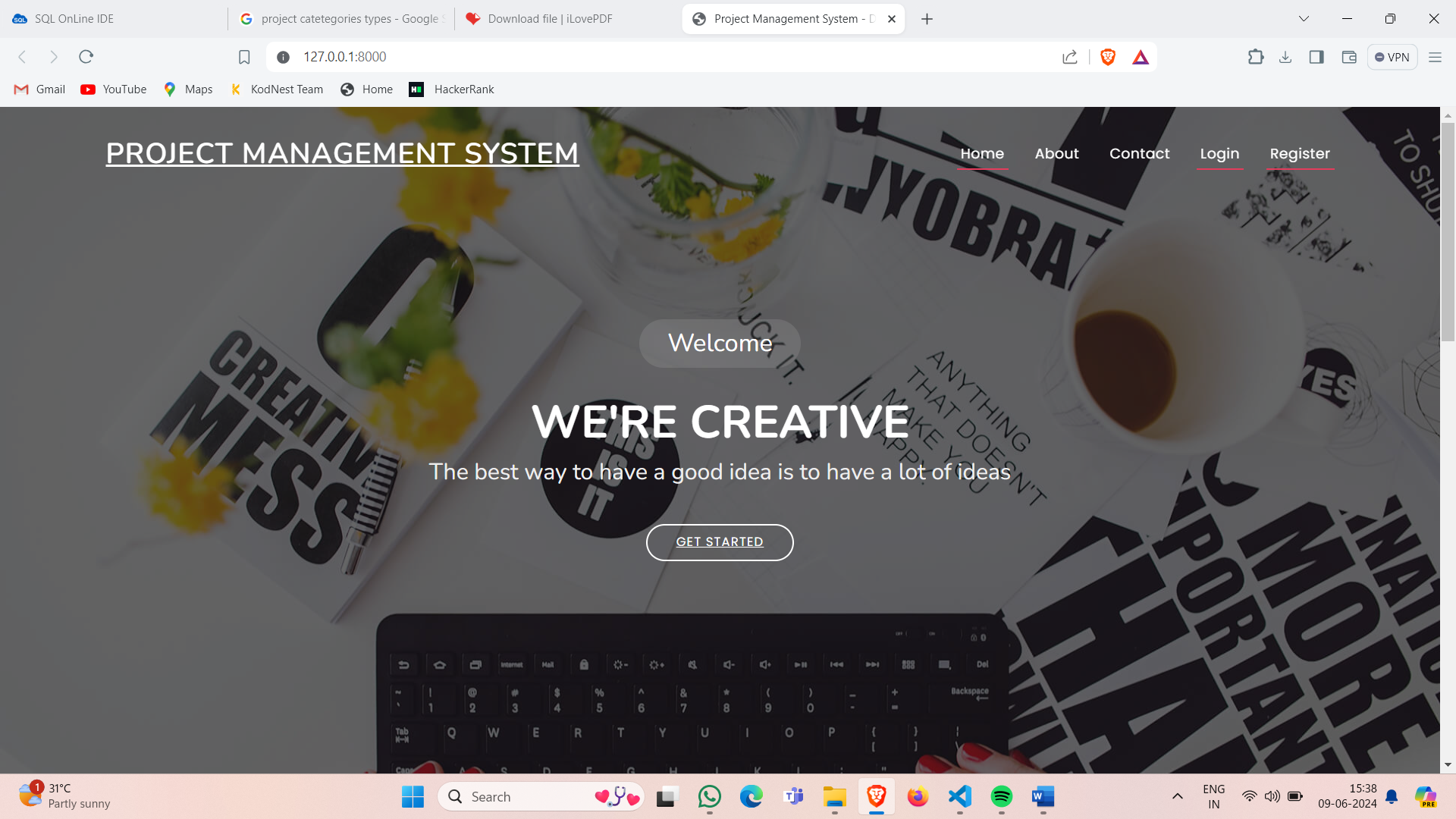
| **Parent Table** | **Child Table** | **Foreign Key** |
| --- | --- | --- |
| Users | Projects | ProjectID |
|  |  |  |

1. **Projects to Tasks (One-to-Many):**

| **Parent Table** | **Child Table** | **Foreign Key** |
| --- | --- | --- |
| Projects | Tasks | ProjectID |

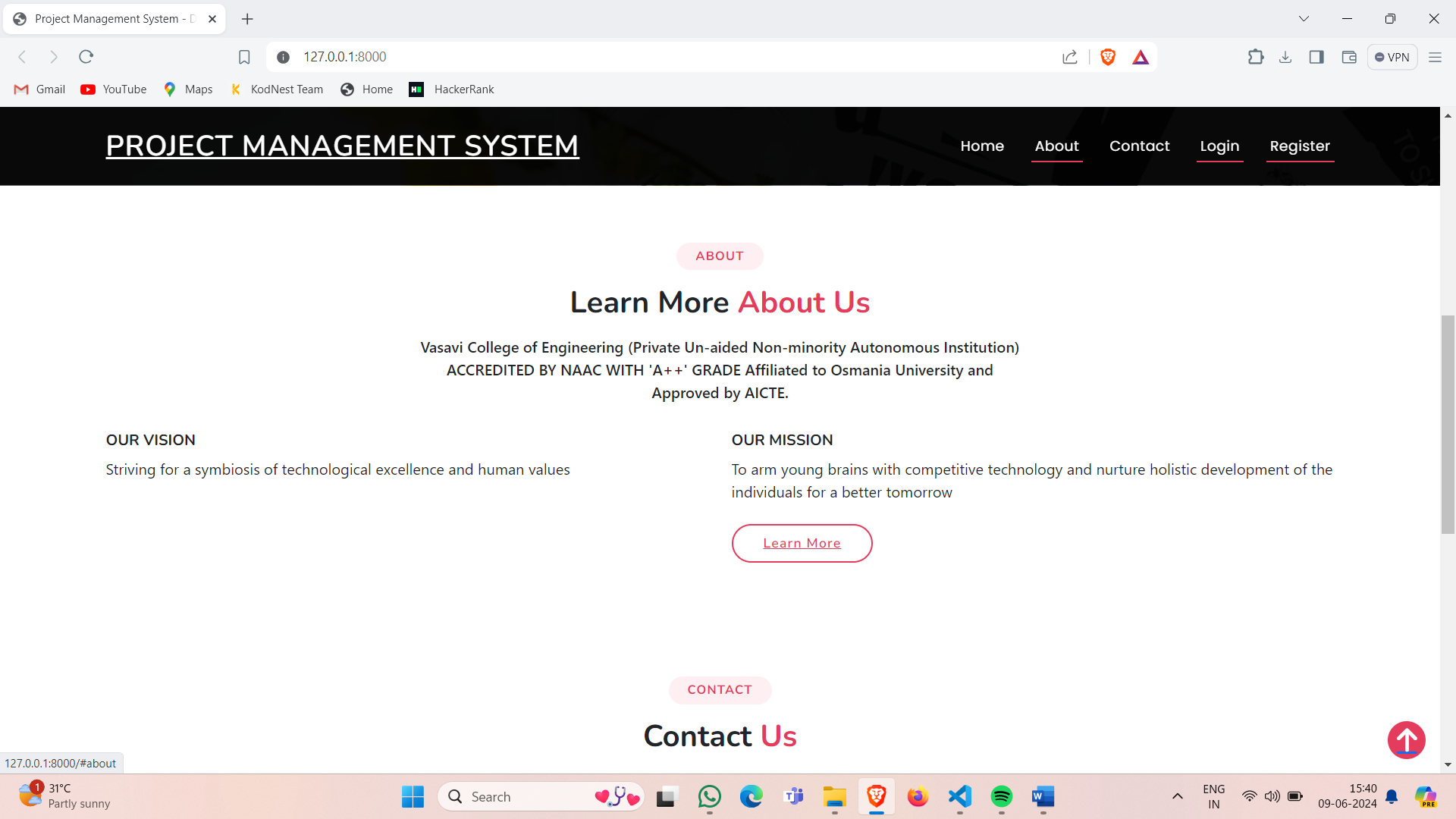
# OUTPUTS & TESTING:

As the application lanched home will opened

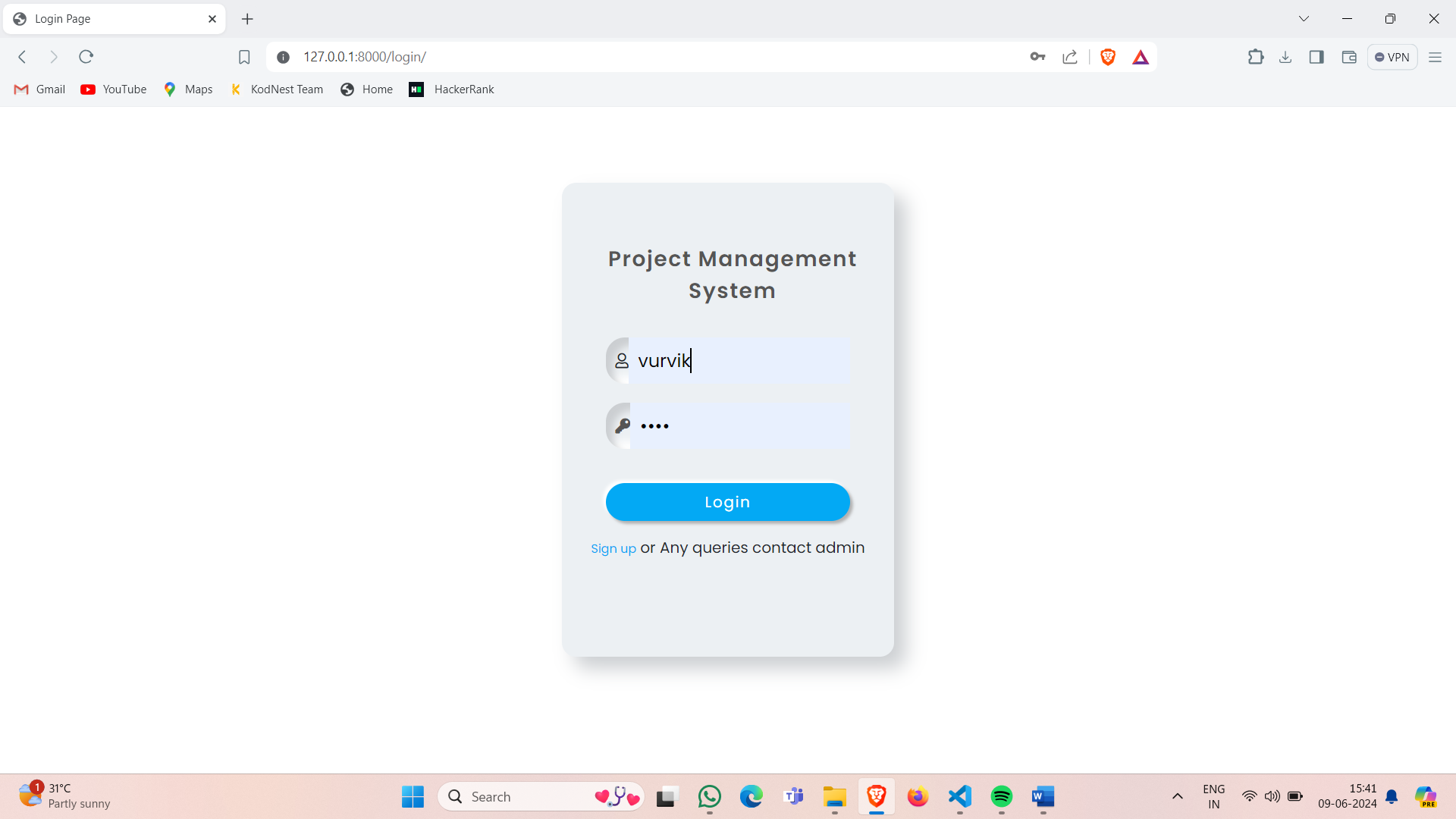


It contains contact us ,about us ,login, registration pages

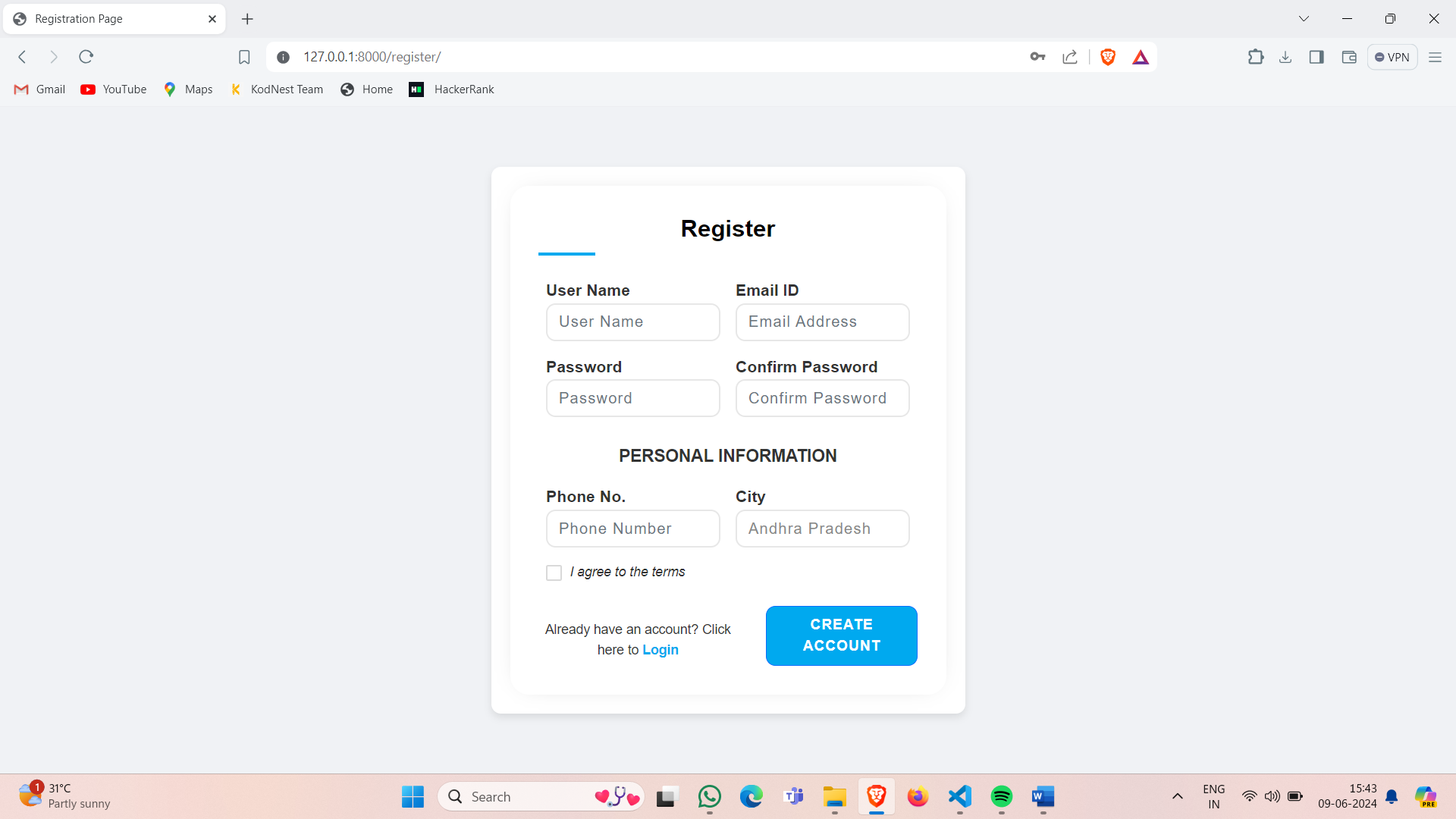
About Us:



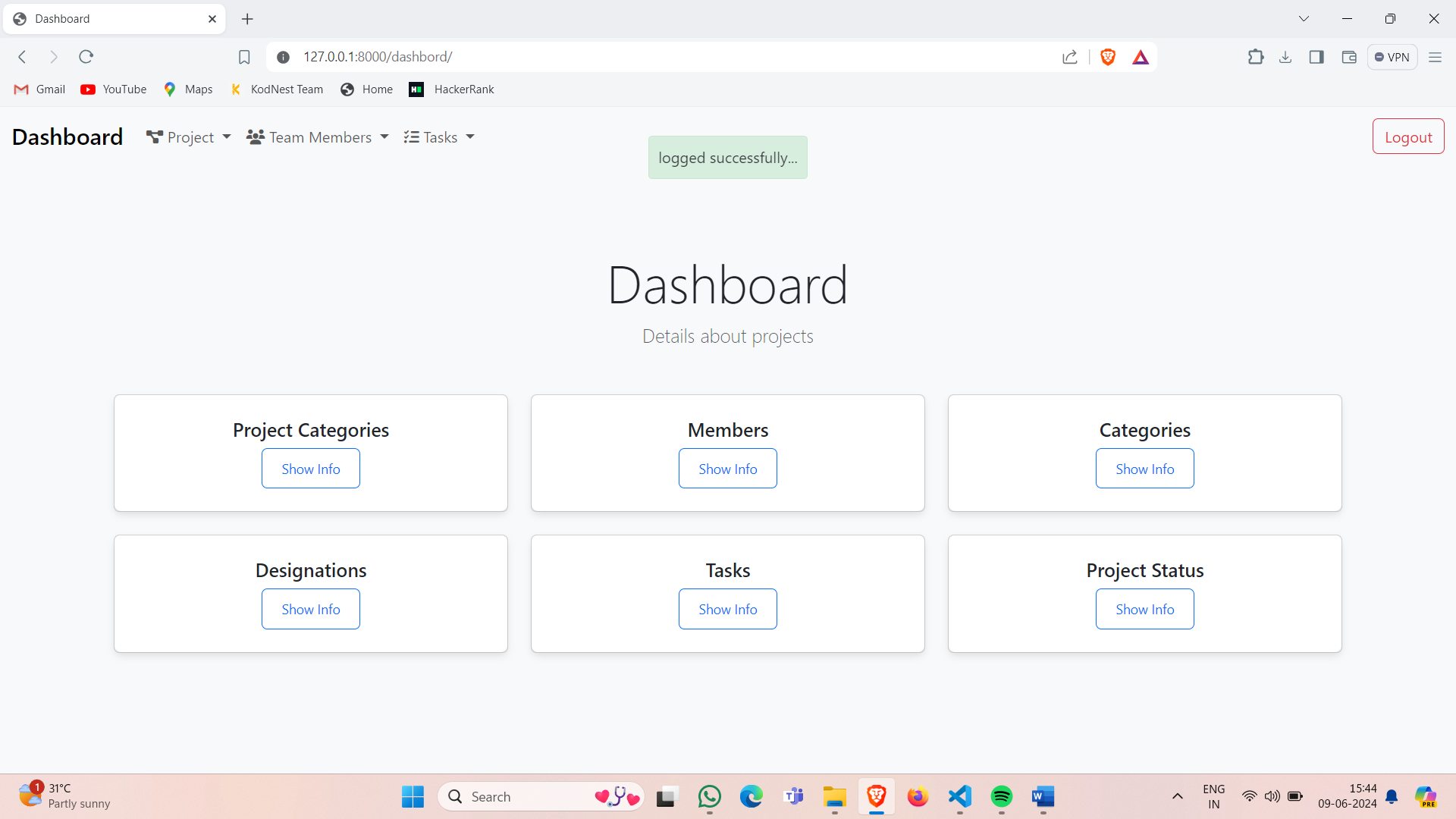
Login Page :



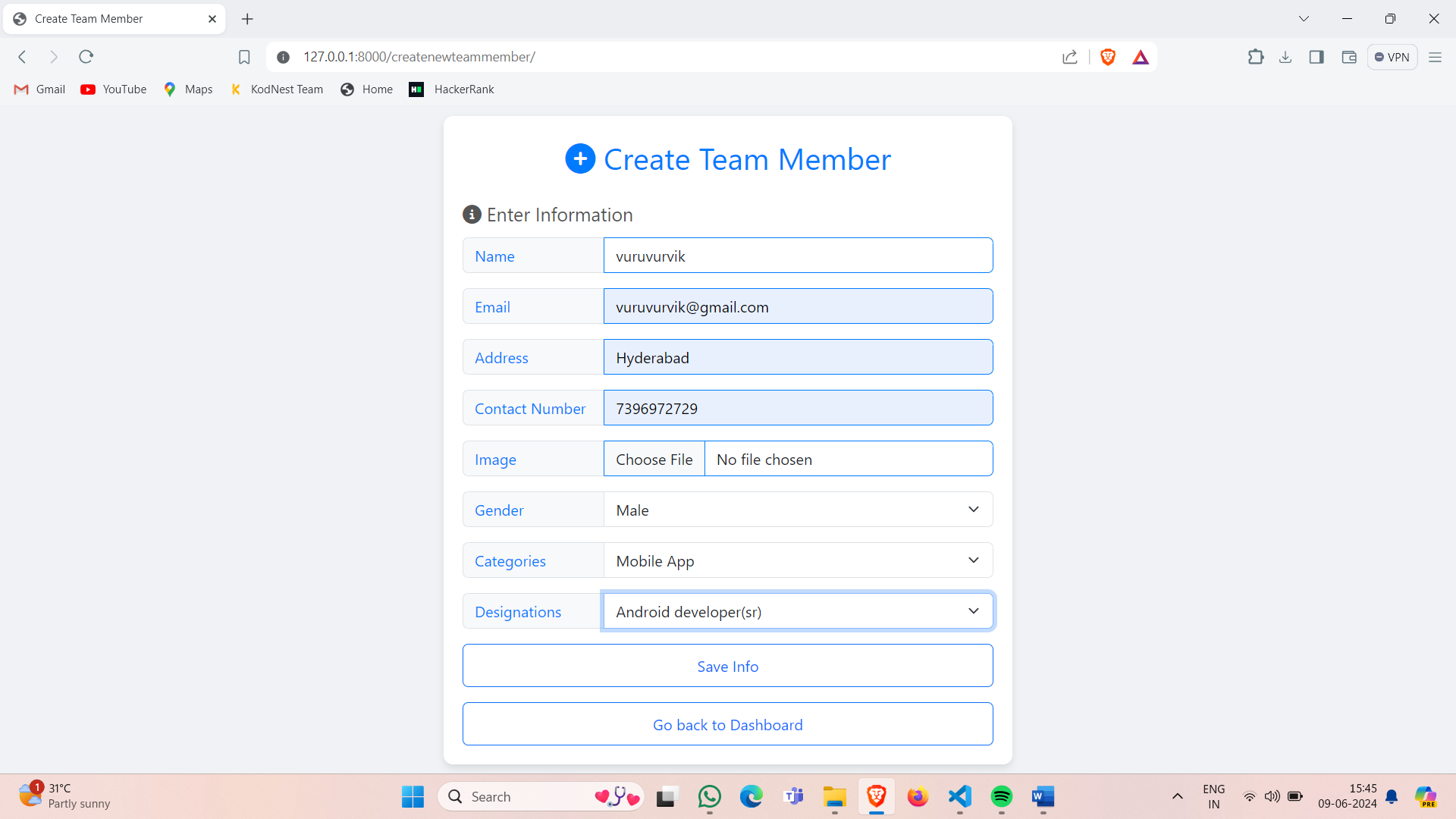
Registration page:



Dashboard:

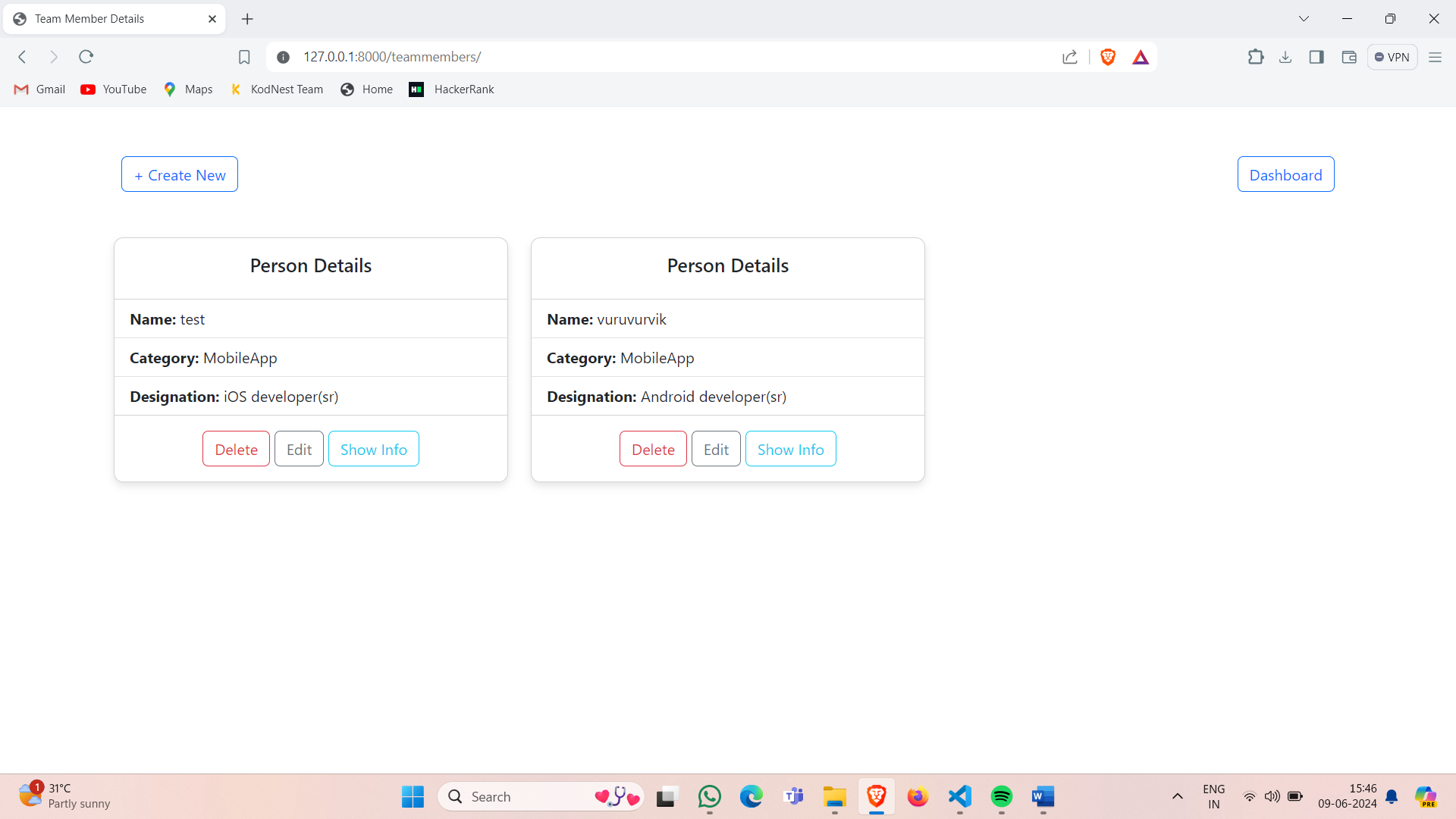


Create team member page:

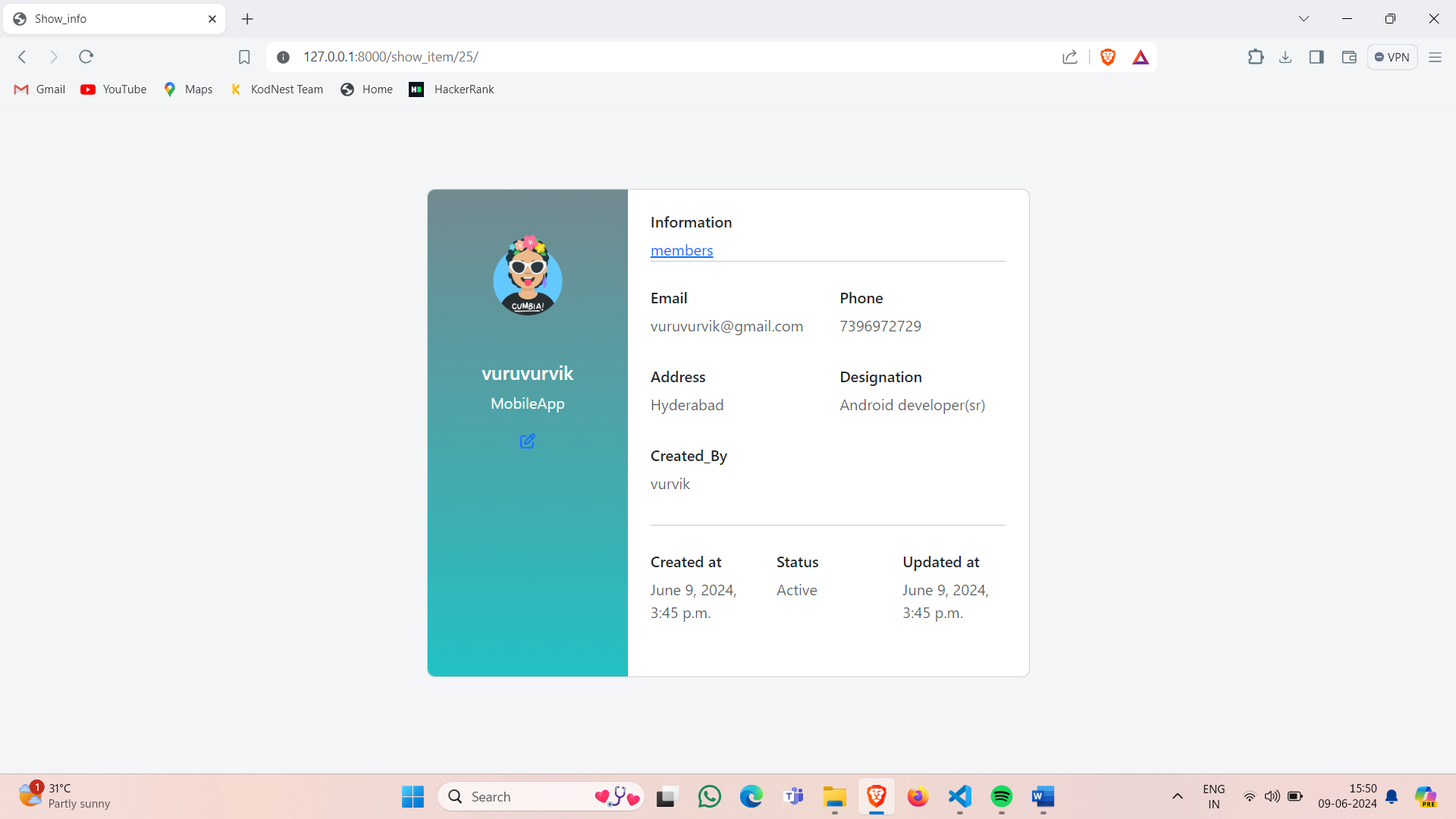


Members page:

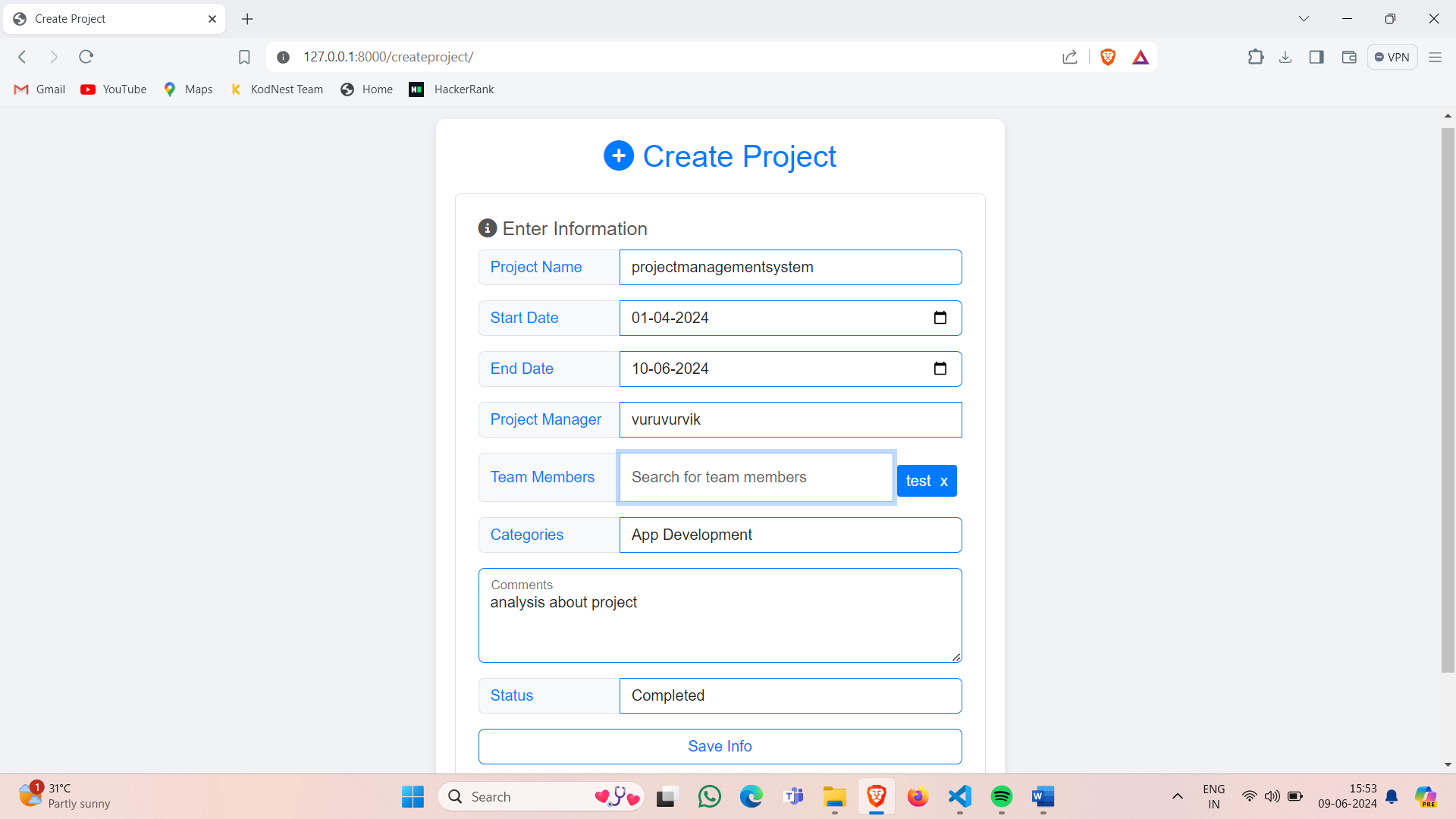
Using delete ,edit &create button we can make crud operations



Information about member &the person who created:

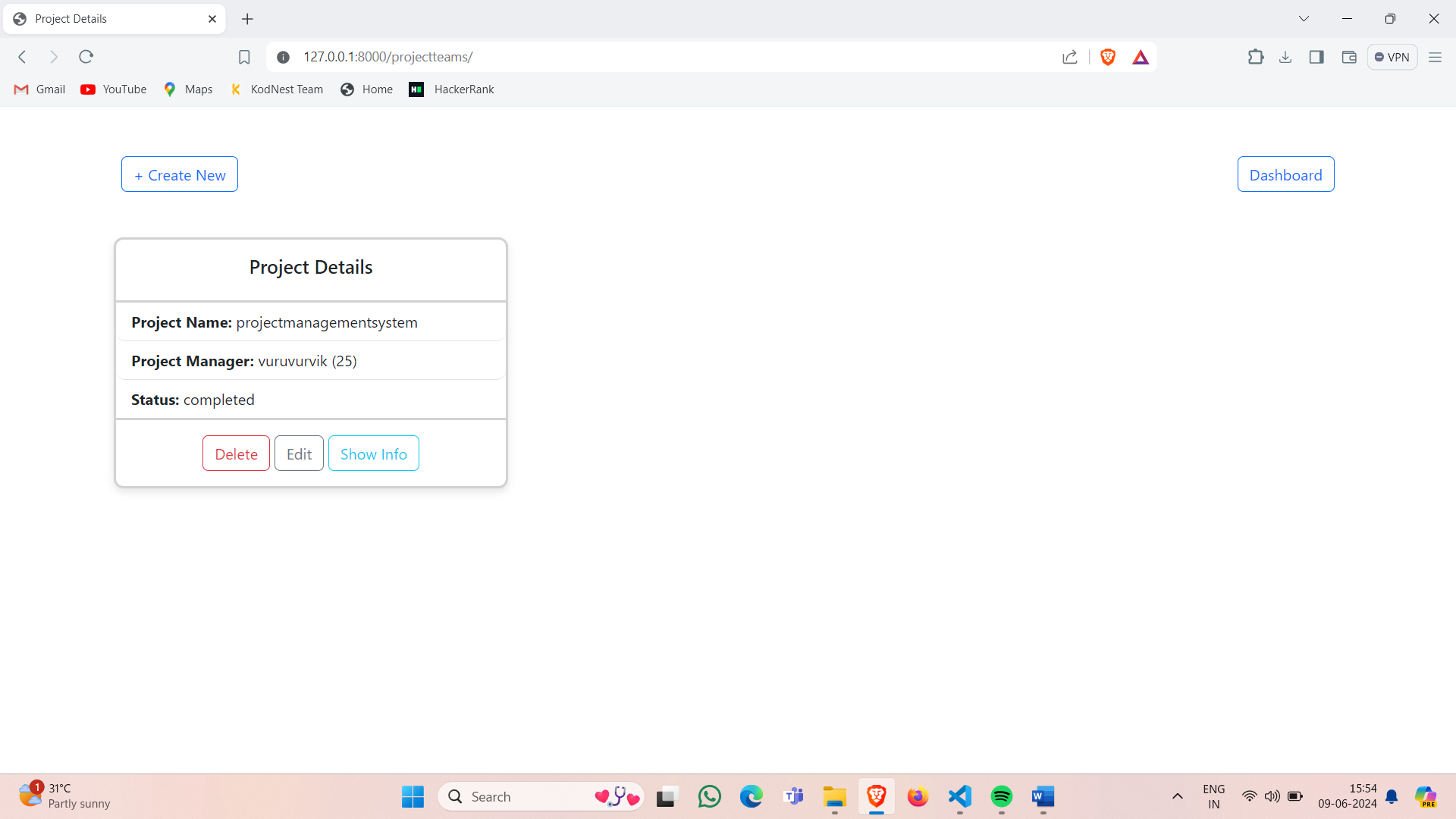


Creating project and adding teammembers:

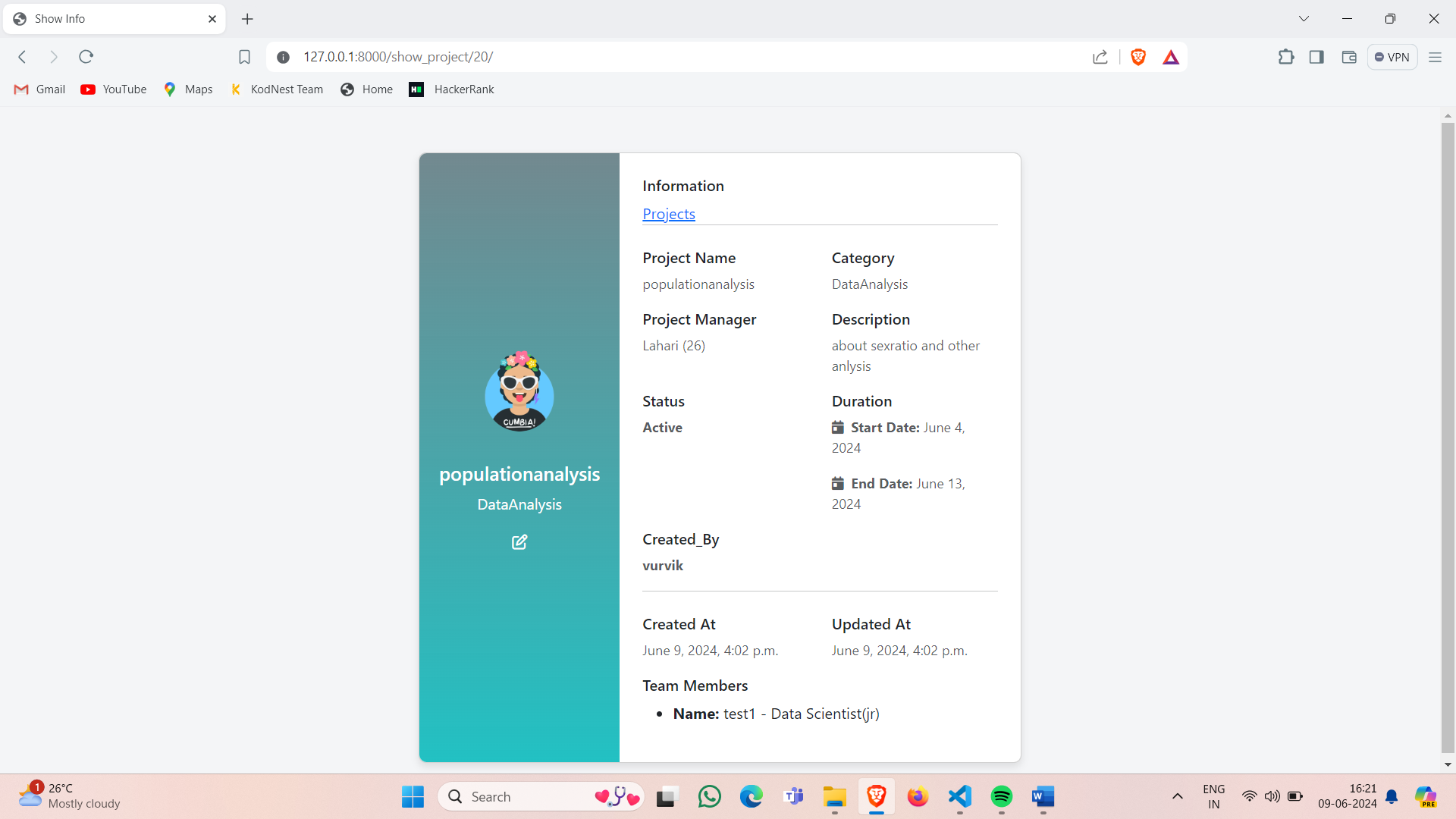


Projects page:

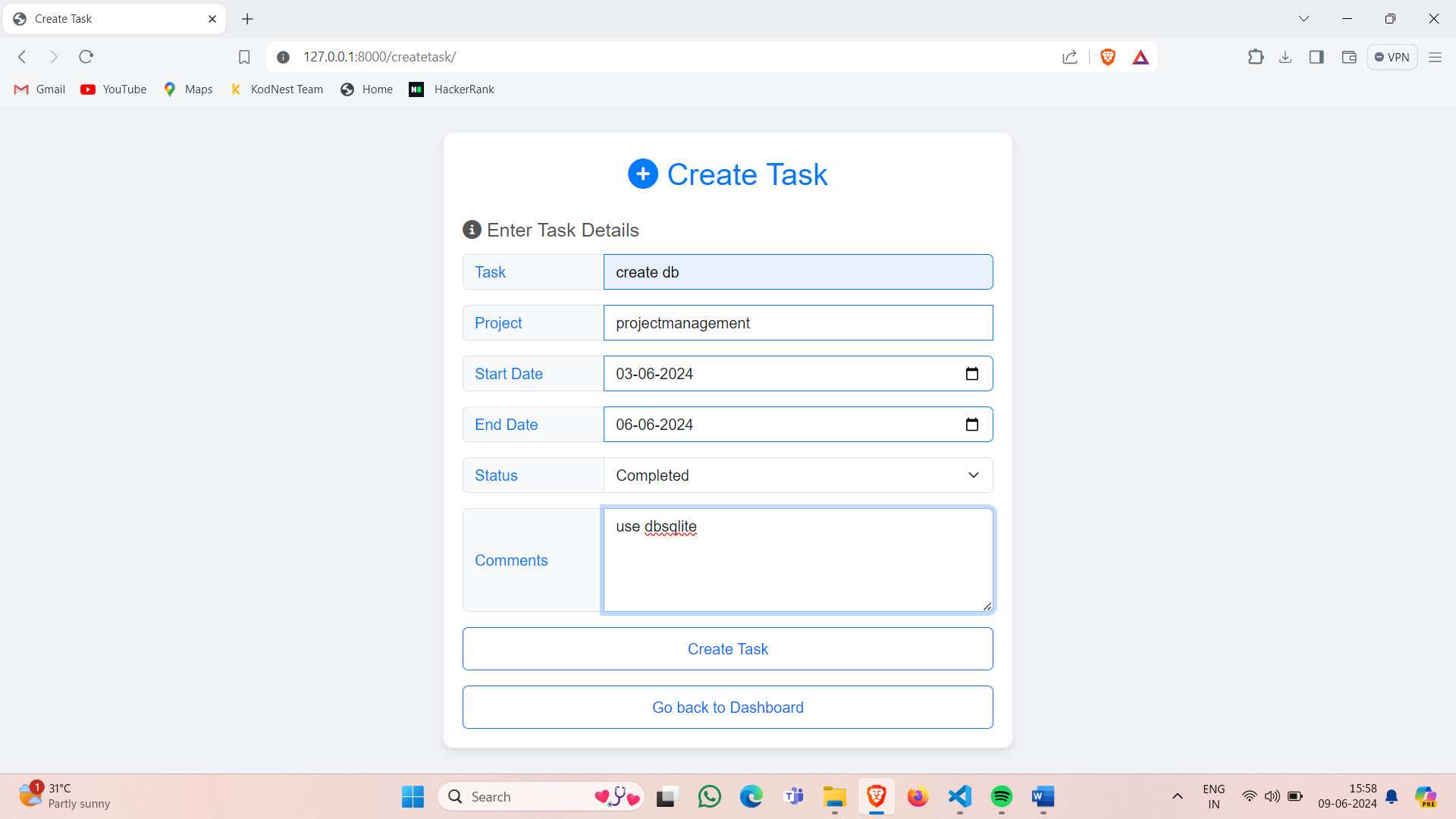
Using delete ,edit &create button we can make crud operations



Project info page:

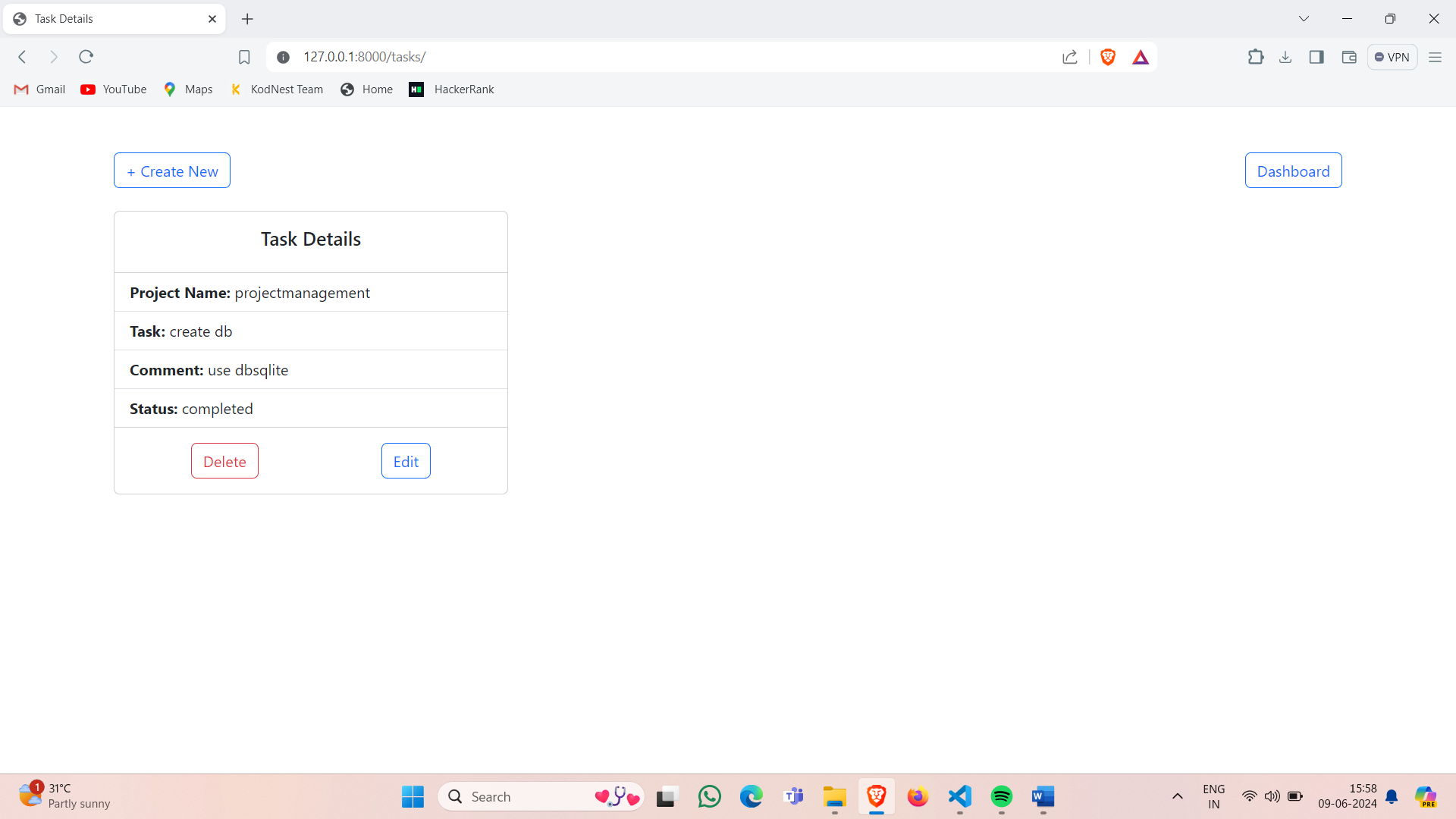


Create task page:

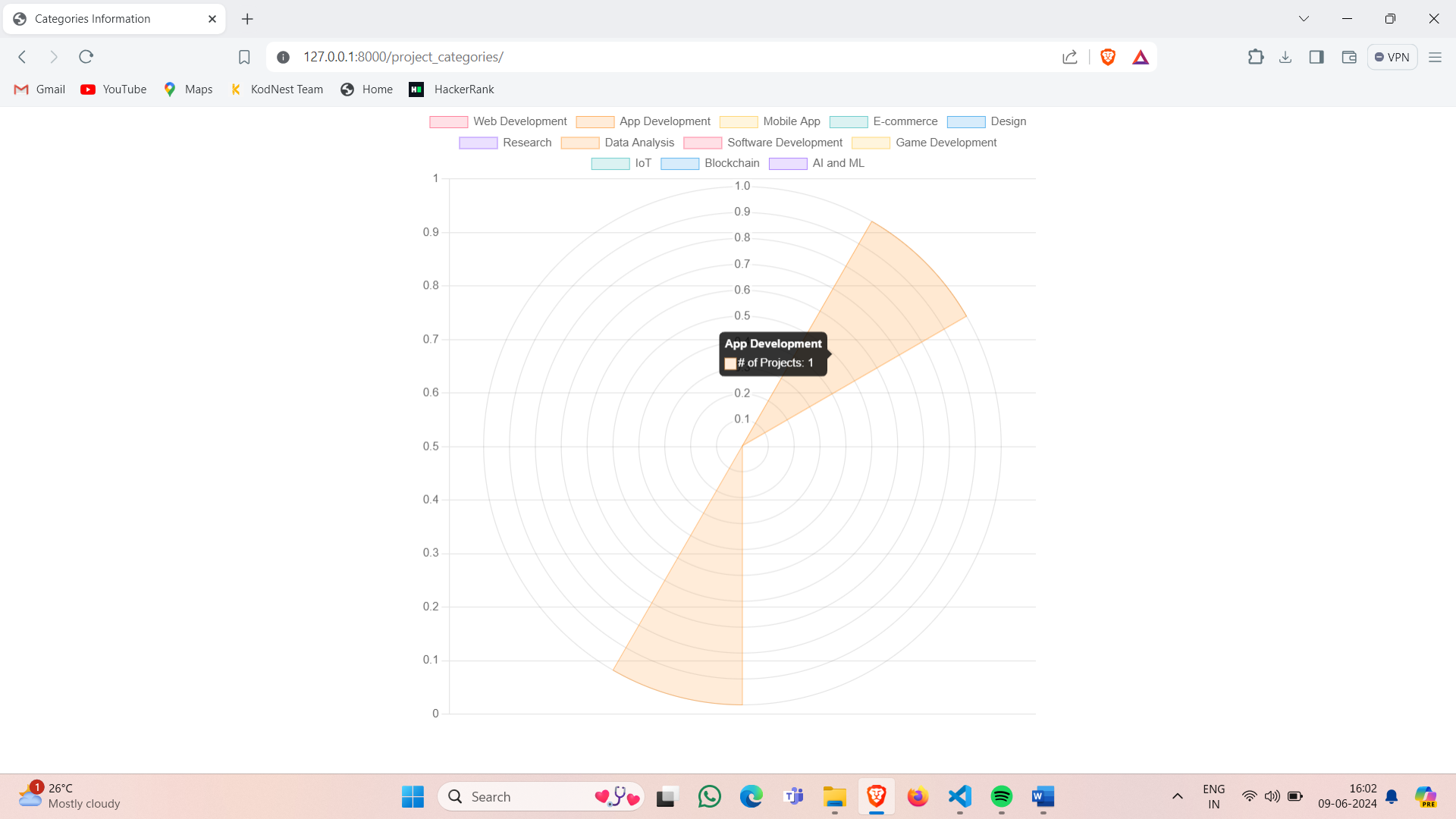


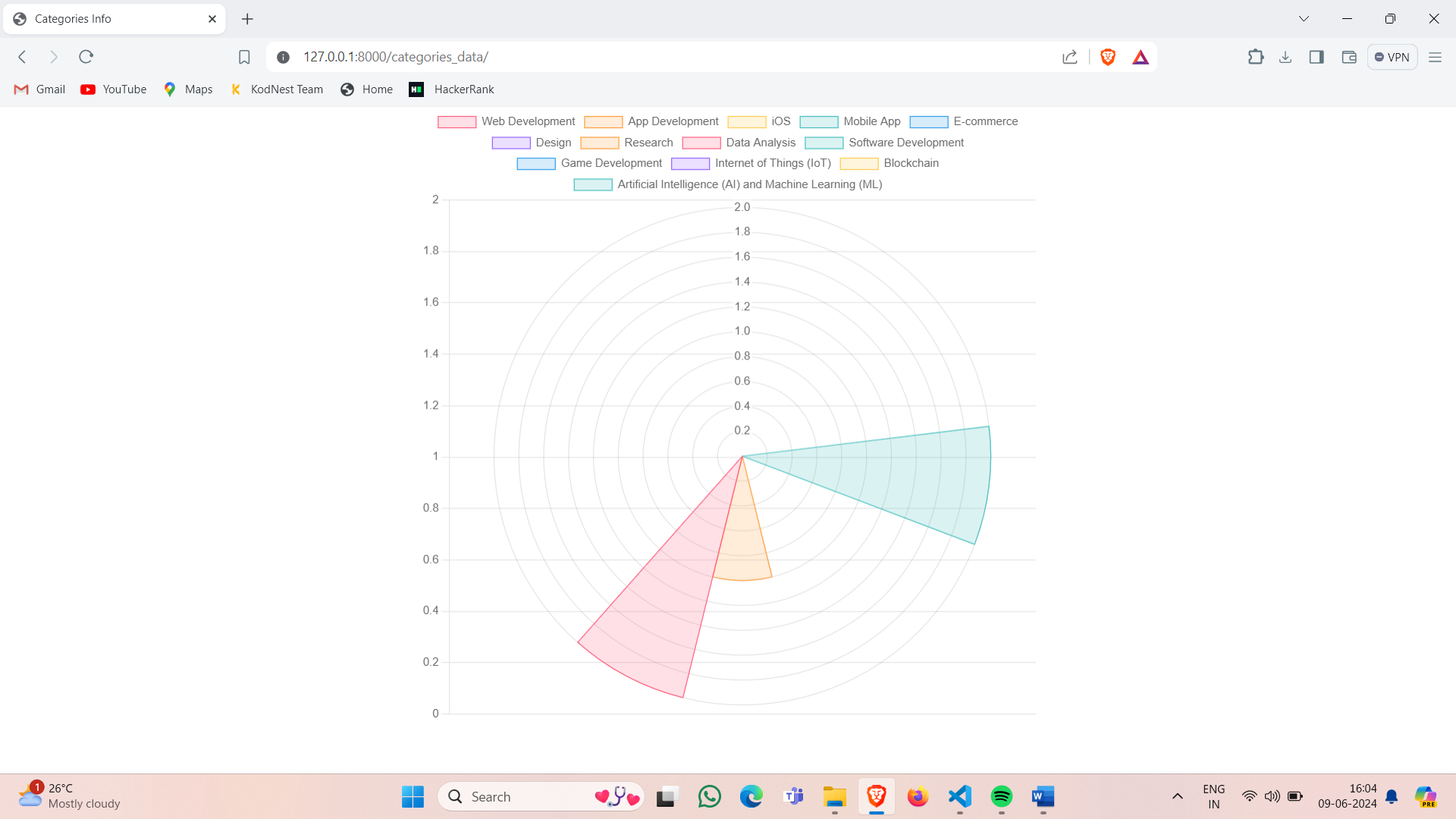
Tasks page:

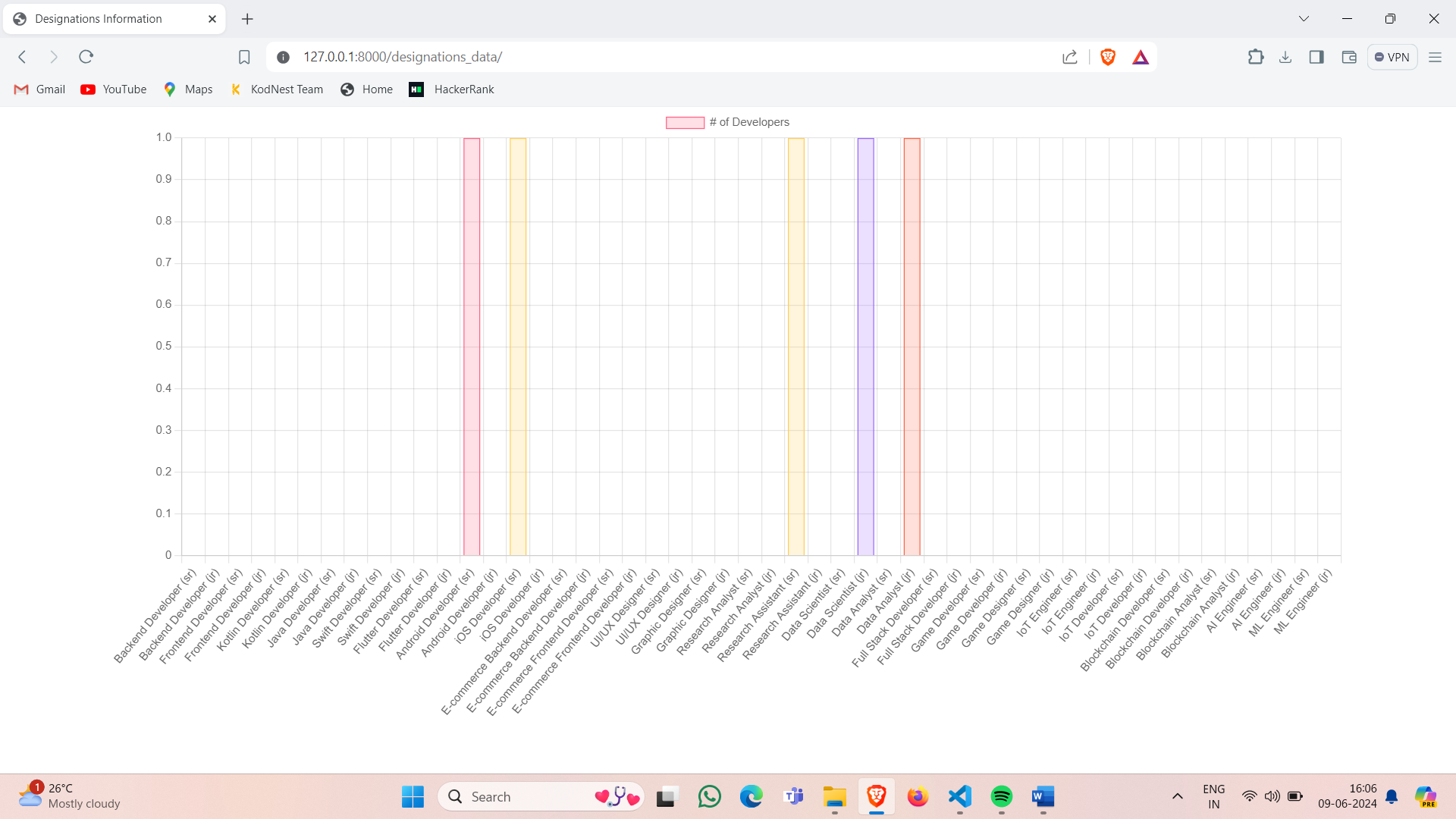
Using delete,edit &create buttons we can make crud operations

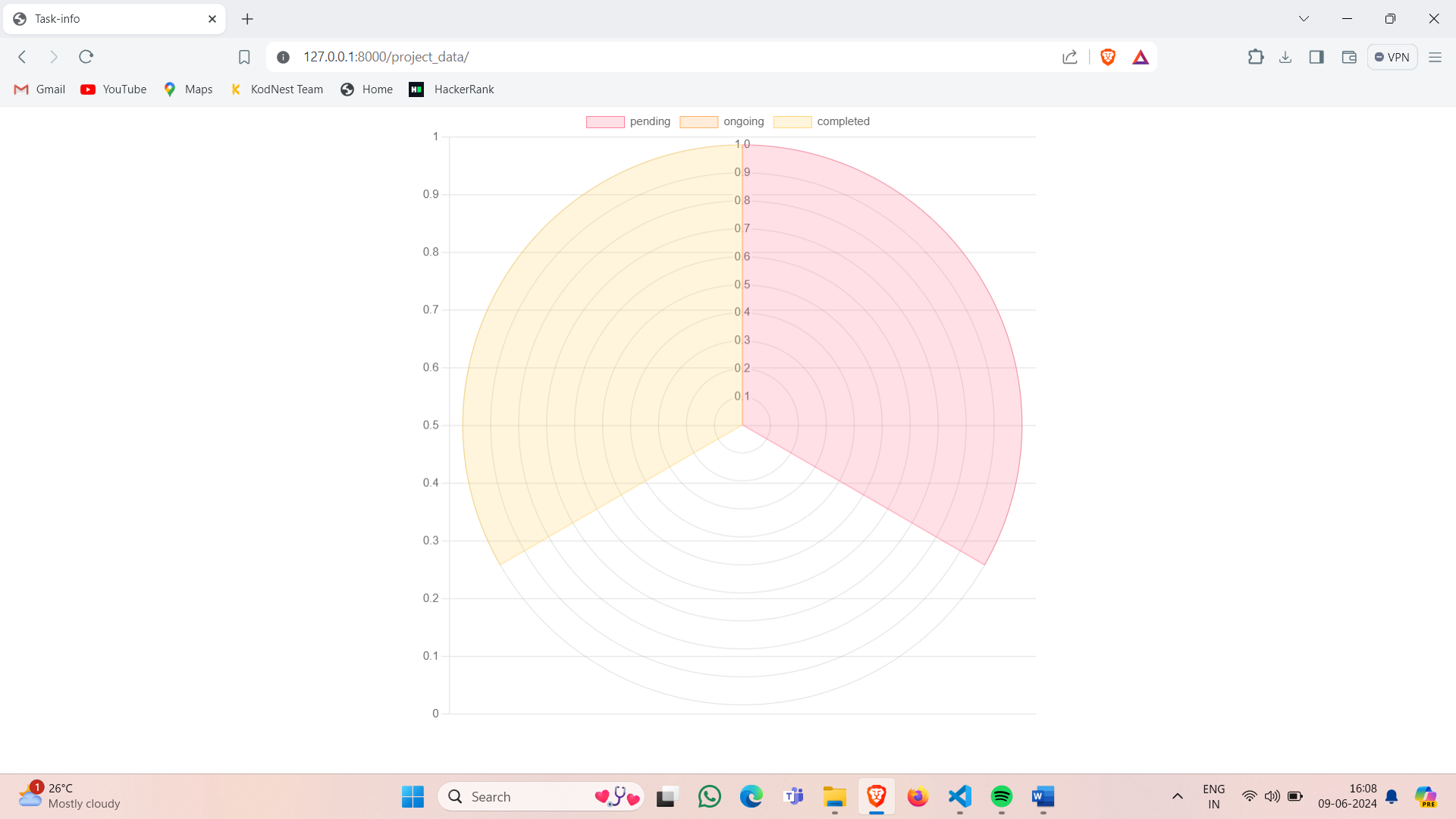


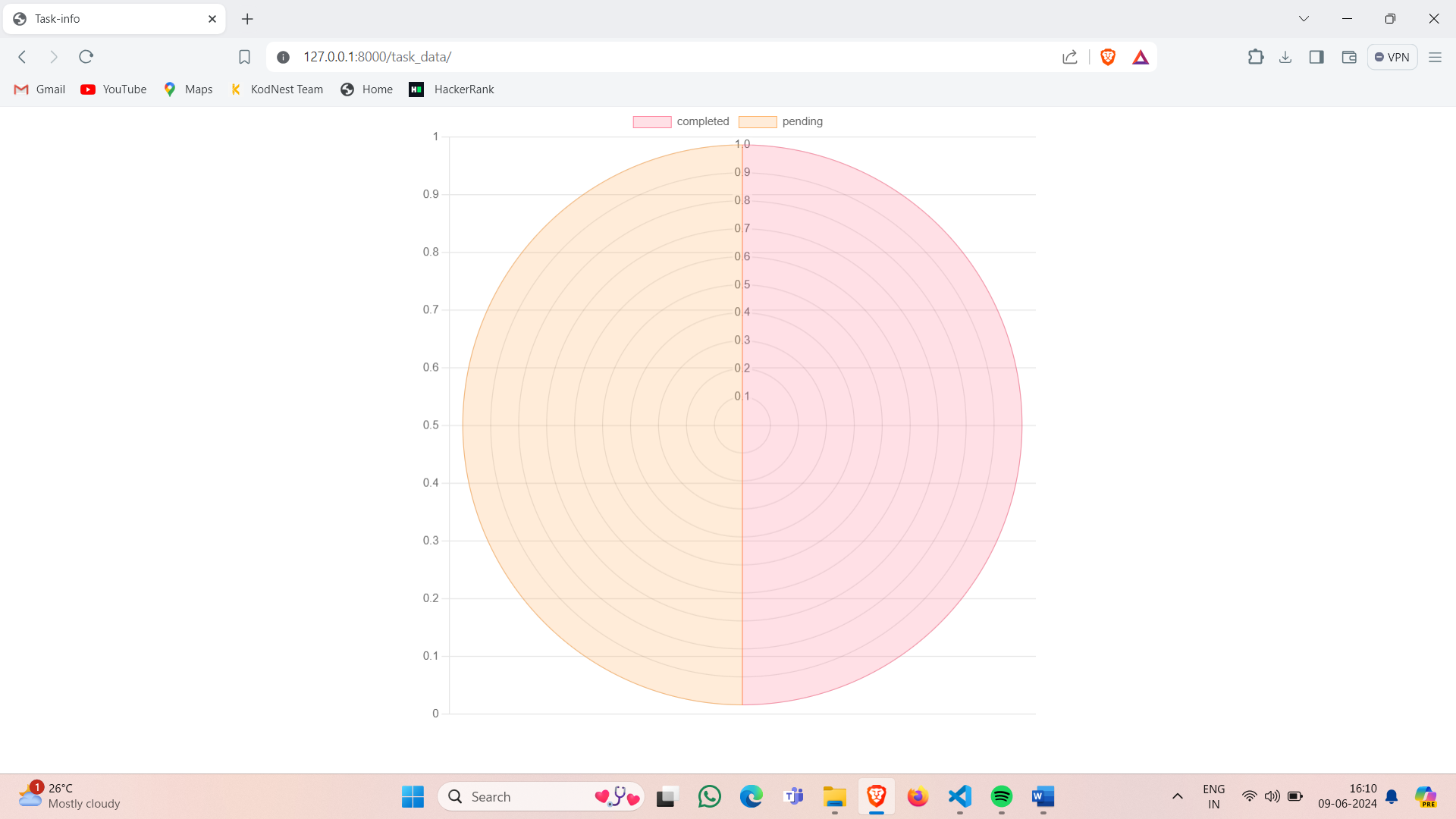
Analysis graphs:











Future work:

Implementing role-based, adding reviews for project and searchengine

# References:

<https://www.geeksforgeeks.org/django-tutorial/>

<https://docs.djangoproject.com/en/5.0/>

<https://getbootstrap.com/>

<https://www.chartjs.org/>