TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

A Final Year Project Report On

"Volunteer Recommendation System"

Under the Supervision of

Mr. Saroj Maharjan

Bhaktapur Multiple Campus

SUBMITTED TO:

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

BHAKTAPUR MULTIPLE CAMPUS

In partial fulfillment for the Bachelor's Degree in Computer Science and Information Technology

Submitted By:
Ajay Kuikel (T.U Roll No. 28283/078)
Aashish Uprety (T.U. Roll No. 28279/078)
Pratik Karki (T.U. Roll No. 28308/078)

SEPTEMBER, 2025

SUPERVISOR'S RECOMMENDATION

This is to recommend that **Ajay Kuikel** (Symbol No. 28283/078, TU Registration No. 5-2-20-896-2021), **Aashish Uprety** (Symbol No. 28279/078, TU Registration No. 5-2-20-893-2021) and **Pratik Karki** (Symbol No. 28308/078, TU Registration No. 5-2-20-921-2021), have carried out project work entitled "**Volunteer Recommendation System**" for the requirement to the project work in Bachelor of Science (B.Sc.) degree in B.Sc. CSIT under my supervision in the Department of Computer Science and Information Technology, Bhaktapur Multiple Campus, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal. To my knowledge, this work has not been submitted for any other degree. They have fulfilled all the requirements laid down by the Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal for the submission of the project work for the partial fulfillment of Bachelor of Science (B.Sc.) degree.

……..…….

Mr. Saroj Maharjan

Supervisor

Department of B.Sc. CSIT

Bhaktapur Multiple Campus, Bhaktapur

Tribhuvan University

DECLARATION

This project work entitled "**Volunteer Recommendation System**" is being submitted to the Department of Computer Science and Information Technology, Bhaktapur Multiple Campus, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal for the partial fulfillment of the requirement to the project work in Bachelor of Science (B.Sc.) degree in B.Sc. CSIT. This project work is carried out by us under the supervision of Mr. Saroj Maharjan in the Department of Computer Science and Information Technology, Bhaktapur Multiple Campus, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal. This work is original and has not been submitted earlier in part or full in this or any other form to any university or institute, here or elsewhere, for the award of any degree.

With respect,

Ajay Kuikel

Symbol No. 28283/078,

Aashish Uprety

Symbol No. 28279/078,

Pratik Karki

Symbol No. 28308/078

LETTER OF FORWARD

Date: 01/09/2025

On the recommendation of **Mr. Saroj Maharjan**, this project work is submitted by **Ajay Kuikel** (Symbol No. 28283/078, TU Registration No. 5-2-20-896-2021), **Aashish Uprety** (Symbol No. 28279/078, TU Registration No. 5-2-20-893-2021) and **Pratik Karki** (Symbol No. 28308/078, TU Registration No. 5-2-20-921-2021), entitled "**Volunteer Recommendation System**" is forwarded by the Department of Computer Science and Information Technology, Bhaktapur Multiple Campus, for the approval to the Evaluation Committee, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal. They have fulfilled all the requirements laid down by the Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal for the project work.

……..……

Mr. Sushant Paudel,
Head of Department
Department of B.Sc. CSIT
Bhaktapur Multiple Campus,
Tribhuvan University

CERTIFICATE OF APPROVAL

This project work entitled "**Volunteer Recommendation System**" by Ajay Kuikel (Symbol No. 28283/078, TU Registration No. 5-2-20-896-2021), Aashish Uprety (Symbol No. 28279/078, TU Registration No. 5-2-20-893-2021) and Pratik Karki (Symbol No. 28308/078, TU Registration No. 5-2-20-921-2021), under the supervision of Mr. Saroj Maharjan in the Department of Computer Science and Information Technology, Bhaktapur Multiple Campus, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), is hereby submitted for the partial fulfillment of the Bachelor of Science (B.Sc.) degree in B.Sc. CSIT. This report has been accepted and forwarded to the Controller of Examination, Institute of Science and Technology, Tribhuvan University, Nepal for the legal procedure.

…….	……
Mr. Saroj Maharjan	**Mr. Sushant Paudel**
Supervisor	**Program Co-Ordinator**
Department of CSIT	Department of CSIT
Bhaktapur Multiple Campus, TU	Bhaktapur Multiple Campus, TU
…….	**…….**
Internal Examiner	**External Examiner**

ACKNOWLEDGEMENTS

Our project report is based on **Volunteer Recommendation System**. An optimum effort has been placed while preparing this documentation and presenting it in front of you. We would like to thank each and every person who directly and indirectly helped in this project. It's our duty to place a sincere gratitude towards **Bhaktapur Multiple Campus** and administration team for imagining beautiful environment and enriching us with equipped facilities to convert our imagination into reality. Furthermore, we would like to thank our supervisor **Mr. Saroj Maharjan** for helping us with each and every problem related to this project work.

With respect,

Ajay Kuikel

Symbol No. 28283/078,

Aashish Uprety

Symbol No. 28279/078,

Pratik Karki

Symbol No. 28308/078

September, 2025

ABSTRACT

The Volunteer Recommendation System is a web-based platform developed to connect social agencies with volunteers. Organizations can post opportunities and volunteers can apply for roles that match their skills and interests. The system uses a TF-IDF weighted cosine similarity recommendation engine to suggest opportunities to volunteers based on selected interests and past participation. This work establishes a backend implementation that is secure (JWT), scalable (MongoDB), and modular, providing a foundation for further improvements such as hybrid recommendation approaches and a frontend UI.

Keywords: *Volunteer Management System; Recommender Systems; Content-Based Filtering; Cosine Similarity; TF-IDF; Web Application*

LIST OF ABBREVIATIONS

Abbreviation	Full Form
API	Application Programming Interface
DBMS	Database Management System
UI	User Interface
JWT	JSON Web Token
TF-IDF	Term Frequency – Inverse Document Frequency
CRUD	Create, Read, Update, Delete
DFD	Data Flow Diagram
ER	Entity-Relationship
ML	Machine Learning
IDE	Integrated Development Environment
REST	Representational State Transfer
MVC	Model-View-Controller

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction

Volunteering is a cornerstone of vibrant, thriving communities, serving as a catalyst for positive change and collective progress. It embodies the spirit of social responsibility, empowering individuals to contribute their time, skills, and passion to causes that resonate with them. Beyond the immediate impact on communities, volunteering fosters personal growth, builds meaningful connections, and promotes a sense of purpose and fulfillment. In an increasingly interconnected world, the need for efficient systems to facilitate volunteering has become more pronounced, ensuring that opportunities align seamlessly with the interests and capabilities of potential volunteers[1].

Central to the Volunteer Recommendation System is a sophisticated content-based recommendation engine powered by a cosine similarity algorithm. This engine intelligently matches volunteers with opportunities by analyzing the tags associated with each opportunity and comparing them to the tags derived from a volunteer's profile and past volunteering history. By leveraging cosine similarity, the algorithm calculates the degree of alignment between the tags of volunteer opportunities and user profiles, ensuring highly relevant and personalized recommendations. This data-driven approach not only enhances the efficiency of the matching process but also maximizes the impact of volunteer contributions by connecting individuals with roles where they can thrive.

Designed with accessibility and inclusivity in mind, the platform caters to a diverse audience, including both tech-savvy users and those less familiar with digital tools. It features intuitive navigation, clear instructions, and robust support mechanisms to ensure a seamless user experience. Additionally, the system fosters a sense of community by enabling volunteers to share their experiences, connect with like-minded individuals, and celebrate the collective impact of their efforts.

This report describes the system's design, implementation, and testing, with emphasis on the cosine similarity-based recommendation algorithm, system architecture and the content-based recommendation framework. It also discusses scalability and potential enhancements.

1.2 Problem Statement

Social agencies and non-profit organizations frequently encounter significant challenges in recruiting volunteers who possess the specific skills, availability, and passion required for their initiatives. These challenges include inefficient outreach, limited visibility of opportunities, and difficulties in matching volunteer capabilities with organizational needs. Concurrently, individuals eager to volunteer often struggle to identify opportunities that align with their skills, interests, and schedules, leading to underutilization of their potential and reduced engagement in community service [1]. The absence of a streamlined, centralized platform intensifies these issues, resulting in missed connections between organizations and motivated volunteers.

Existing platforms, such as VolunteerMatch, have demonstrated the potential of centralized systems to facilitate volunteer recruitment by providing a digital space for organizations to post opportunities and for volunteers to browse them [2]. However, many of these platforms lack advanced personalization features, relying on manual searches or basic filters. This can lead to suboptimal matches, where volunteers may not feel fully engaged or organizations may not receive the expertise they require [3].

Additionally, new users often face the “cold start” problem, where no history is available to generate recommendations. To address this, our system captures user interests during signup, ensuring initial recommendations are still personalized.

By addressing these gaps, the Volunteer Recommendation System aims to streamline volunteer recruitment, improve engagement, and amplify the impact of volunteer-driven initiatives in communities [4].

1.3 Objectives

Following are the objectives of the project:

- To develop a web platform for agencies to post tagged volunteer opportunities.
- To implement a cosine similarity-based recommendation system to match volunteers with opportunities.

1.4 Scope and Limitations

1.4.1 Scope

The scope of this project defines the range of functionalities implemented within the Volunteer Management and Recommendation System. The system focuses on connecting volunteers and organizations through an intelligent recommendation process and providing efficient opportunity management.

- Developed a web-based platform for social agencies to post volunteer opportunities with specific tags and description.
- Enabled volunteers to register, log in, and create profiles indicating their interests.
- Role based dashboards for each user type: volunteer, organization and admin.
- Opportunity posting, application, and approval process, ensuring proper workflow between volunteers and organizations.
- The system recommends opportunities to volunteers based on their history and interests.

1.4.2 Limitations

While the Volunteer Recommendation System meets its core objectives, certain constraints were identified during development and testing. Key limitations are outlined below:

- Depends on accurate, complete tags from organizations and users for effective recommendations.
- Limited to content-based filtering using cosine similarity, excluding collaborative filtering due to resource constraints.
- Web-only access, with no mobile application support, potentially limiting accessibility.
- Recommendations are based on available data and may improve with more user activity.
- The project utilized only a single iteration of the Agile development cycle, limiting the scope for continuous feedback and refinement.
- User feedback mechanisms were not integrated, which limited opportunities for iterative improvement and personalization.

1.5 Development Methodology

The project adopts an Agile methodology, emphasizing flexibility, adaptability, and incremental development within a single iteration.

- The Agile approach was used to ensure modular and adaptable development aligned with evolving requirements.
- A single development cycle (iteration) was carried out, focusing on delivering key features such as user authentication, opportunity management, and the recommendation engine.
- Followed Scrum-inspired practices for task management, including regular progress tracking and review meetings instead of daily stand-ups.
- Continuous integration and testing were used to validate the cosine similarity algorithm during development.

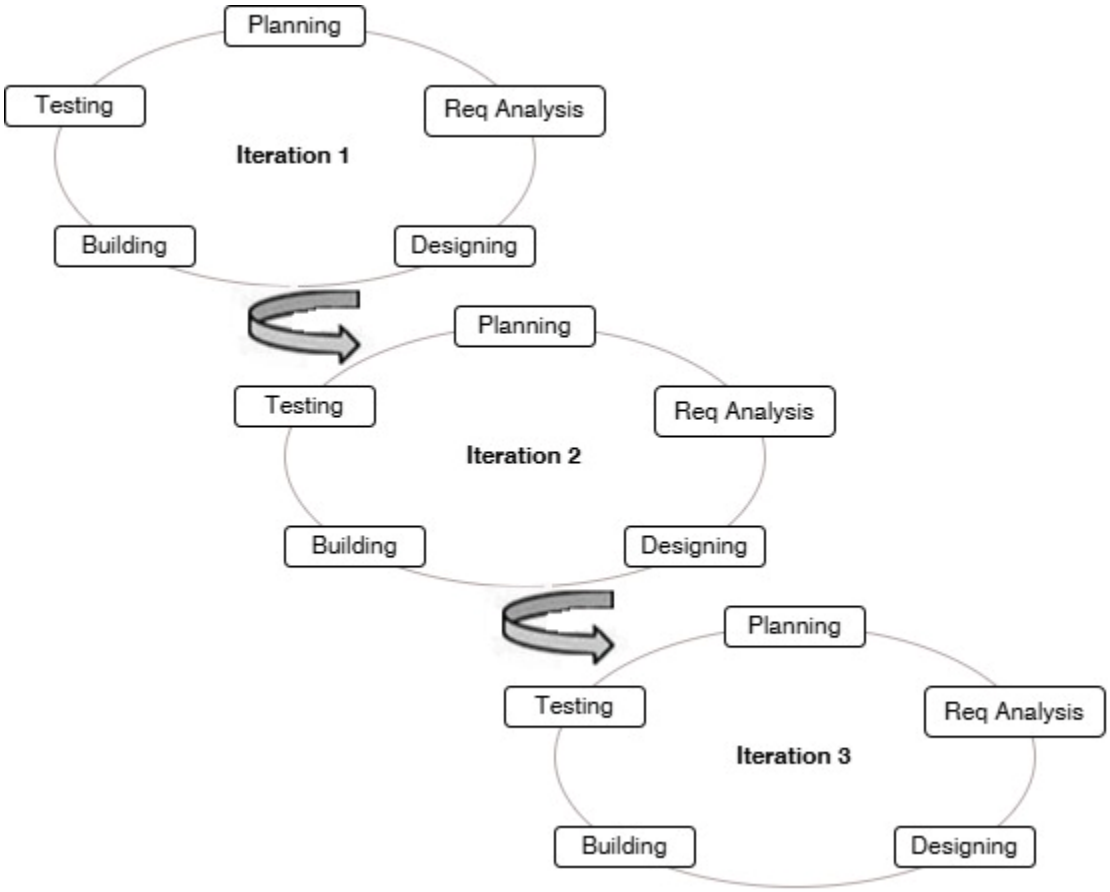


Figure 1.1: Agile Methodology

1.6 Report Organization

The remaining report is organized as follows:

Chapter 1: Introduction introduces the concept of volunteering and the significance of the Volunteer Recommendation System. It includes the problem statement, objectives, scope and limitations, development methodology.

Chapter 2: Background Study & Literature Review explores the background of recommendation systems and volunteer management technologies, reviews previous research and similar systems, and compares different approaches to highlight the foundation for using a cosine similarity-based algorithm.

Chapter 3: System Analysis outlines the requirements analysis, including functional and non-functional requirements, a feasibility study (technical, operational, economic, and schedule), use case diagrams, data flow diagrams, ER diagram, and a system overview to define the project's technical and operational needs.

Chapter 4: System Design details the technical design of the system, including architecture, database, forms/reports, and a description of the cosine similarity algorithm with pseudocode.

Chapter 5: Implementation & Testing describes the tools and technologies used, module-wise implementation details, testing strategies (unit and system testing), test cases, and result analysis to validate the system's functionality.

Chapter 6: Conclusion & Recommendations summarizes the work done, discusses challenges faced during development, and provides recommendations for future improvements to enhance the system's scalability and features.

The report concludes with a references section, formatted in IEEE style, and an appendix containing supplementary materials, such as additional diagrams or data, to support the project documentation.

CHAPTER 2: BACKGROUND STUDY & LITERATURE REVIEW

2.1 Background of the Technology/Method

The Volunteer Recommendation System is built using modern web technologies to deliver a robust and scalable platform for connecting social agencies with volunteers. The system leverages Node.js and Express.js for server-side development, providing a lightweight, efficient framework for handling HTTP requests and API endpoints. MongoDB, is used to store user profiles, volunteer opportunities, and their associated tags (e.g., skills, cause areas), enabling flexible and scalable data management. Authentication is implemented using JSON Web Tokens (JWT), ensuring secure user access and session management. The core of the system is a content-based recommendation engine powered by the cosine similarity algorithm, which matches volunteers to opportunities by comparing vectorized tags from user profiles and opportunity listings. Cosine similarity calculates the cosine of the angle between two vectors, offering an effective metric for tag-based matching due to its computational efficiency and ability to handle sparse data. The development process follows the Agile methodology, utilizing iterative sprints to ensure the platform aligns with user needs. To mitigate the cold start issue, the system allows users to select interests during registration, which are used to initialize their profile vector before any volunteering history exists.

2.2 Literature Review

The literature on recommendation systems and volunteer management provides critical insights into the design of the Volunteer Recommendation System. Ricci et al. [5] categorize recommendation systems into content-based, collaborative, and hybrid approaches. Content-based filtering, as used in this project, matches users to items based on explicit attributes (e.g., tags), making it suitable for platforms with limited initial user interaction data. Aggarwal [4] further elaborates on content-based methods, highlighting the effectiveness of cosine similarity in measuring attribute similarity, particularly for text or tag-based applications. It also emphasizes that content-based filtering is suitable for domains with structured metadata and limited collaborative information, common in early-stage systems.

Studies on volunteer engagement platforms have also underscored the importance of intelligent matchmaking. Xu [6] propose that personalized recommendations based on user interest and behavior analysis improve participation rates in social causes and reduce volunteer drop-off.

Existing platforms like VolunteerMatch and Idealist facilitate connections through manual search and basic filtering (e.g., by location or cause) but lack advanced personalization based on user history [2], [7]. It also suggests that personalized recommendations can significantly improve user engagement, a gap the Volunteer Recommendation System addresses through its cosine similarity-based approach and initial interest-based onboarding.

Collaborative filtering, another common recommendation technique, leverages user behavior patterns to suggest items [6]. Hybrid methods, combining content-based and collaborative filtering, offer improved accuracy but require complex implementation [8]. The Volunteer Recommendation System opts for content-based filtering due to its simplicity and effectiveness in the context of tag-based matching.

2.2.1 Compare and Contrast Approaches

The Volunteer Recommendation System's content-based approach using cosine similarity is compared with alternative methods and platforms to highlight its suitability:

- **Content-Based (Cosine Similarity) vs. Collaborative Filtering:** Content-based filtering matches volunteers to opportunities using explicit tags, making it effective for platforms with limited user data. Collaborative filtering, reliant on user behavior patterns, requires extensive interaction data, which is impractical for a new volunteer platform. The content-based approach is chosen for its simplicity and immediate applicability.
- **Content-Based vs. Keyword-Based Search:** Platforms like VolunteerMatch and Idealist rely on manual searches and basic filters [2], [7], which can be time-consuming and less precise. The Volunteer Recommendation System's cosine similarity algorithm automates matching by comparing tag vectors, reducing user effort and improving recommendation relevance.
- **Cosine Similarity vs. Other Similarity Metrics:** Cosine similarity is preferred for its efficiency in handling high-dimensional, sparse data, as seen in tag-based systems.

Alternatives like Euclidean distance are less suitable due to sensitivity to vector magnitude, while Jaccard similarity is limited to binary data.

To mitigate the limitations of content-based filtering, particularly the cold start problem faced by new users, the system employs a tag-based interest selection mechanism during signup to initialize user vectors. These vectors are then compared with opportunity tag vectors using the cosine similarity algorithm, enabling the system to provide immediate and personalized recommendations even without prior user activity.

2.2.2 Overview of Content Based Filtering

Content-based filtering is a recommendation approach that relies on the attributes (or features) of items and users to generate personalized suggestions. Unlike collaborative filtering, which depends on user–user interactions or co-behavior patterns, content-based filtering focuses exclusively on the characteristics of the items the user has shown interest in [5].

In this project, both volunteers and opportunities are described using tags (e.g., "education," "health," "sports"). During registration, users select interests, and their profile is further enriched by the tags of opportunities they engage in. The system then recommends new opportunities that share similar tags with the user profile, ensuring recommendations are personalized according to the user's skills and interests rather than general popularity.

Steps of Content-Based Recommendation Process:

1. Data Collection: Collect detailed user and item data.
2. Feature Extraction: Extract meaningful features from the collected data and vectorize it.
3. Build Item Profiles: Represent each item as a vector of its extracted features.
4. Build User Profiles: Construct a profile summarizing the user's preferences based on their interactions with items.
5. Similarity Calculations: Measure the Cosine similarity between the user profile and item profiles.
6. Generate & Present Recommendations: Generate and display the recommendations to the user in a meaningful way.
7. Update User Profile: Continuously refine the user profile based on new interactions.

CHAPTER 3: SYSTEM ANALYSIS

3.1 Requirement Analysis

The requirement analysis phase identifies the functional and non-functional requirements of the Volunteer Recommendation System to ensure it meets the needs of social agencies and volunteers.

3.1.1 Functional Requirements

Functional requirements specify the core features and functionalities the Volunteer Recommendation System must provide to achieve its objectives of connecting volunteers with opportunities.

- User registration and authentication (volunteer/organization/admin).
- Organization: Create, update, delete, and manage opportunities.
- Volunteer: View and apply for opportunities.
- Recommendation of opportunities to volunteers.
- Application approval and rejection.

These functionalities are detrimental to the well-functioning of the system and depicts the way users can interact with the system. The use case diagram for the system can be interpreted from the use case diagrams given below:

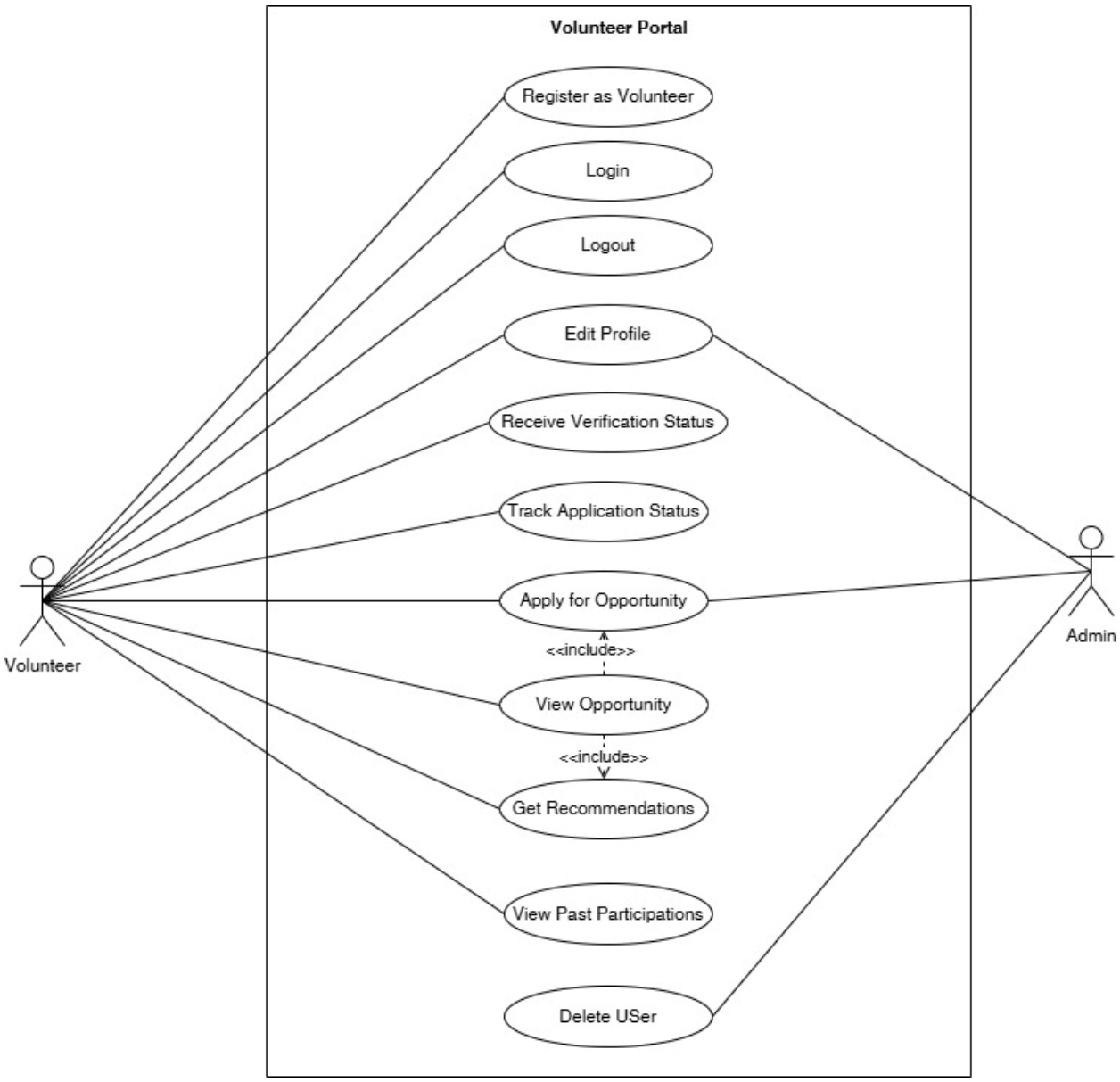


Figure 3.1: Use Case Diagram of the user

The volunteer use case diagram outlines the key interactions that a volunteer can have with the system. These include registering and logging in, browsing available opportunities, applying to relevant postings, viewing recommendations, and tracking application statuses. This diagram highlights how the system facilitates seamless engagement for volunteers by simplifying access to meaningful opportunities based on their interests and experience.

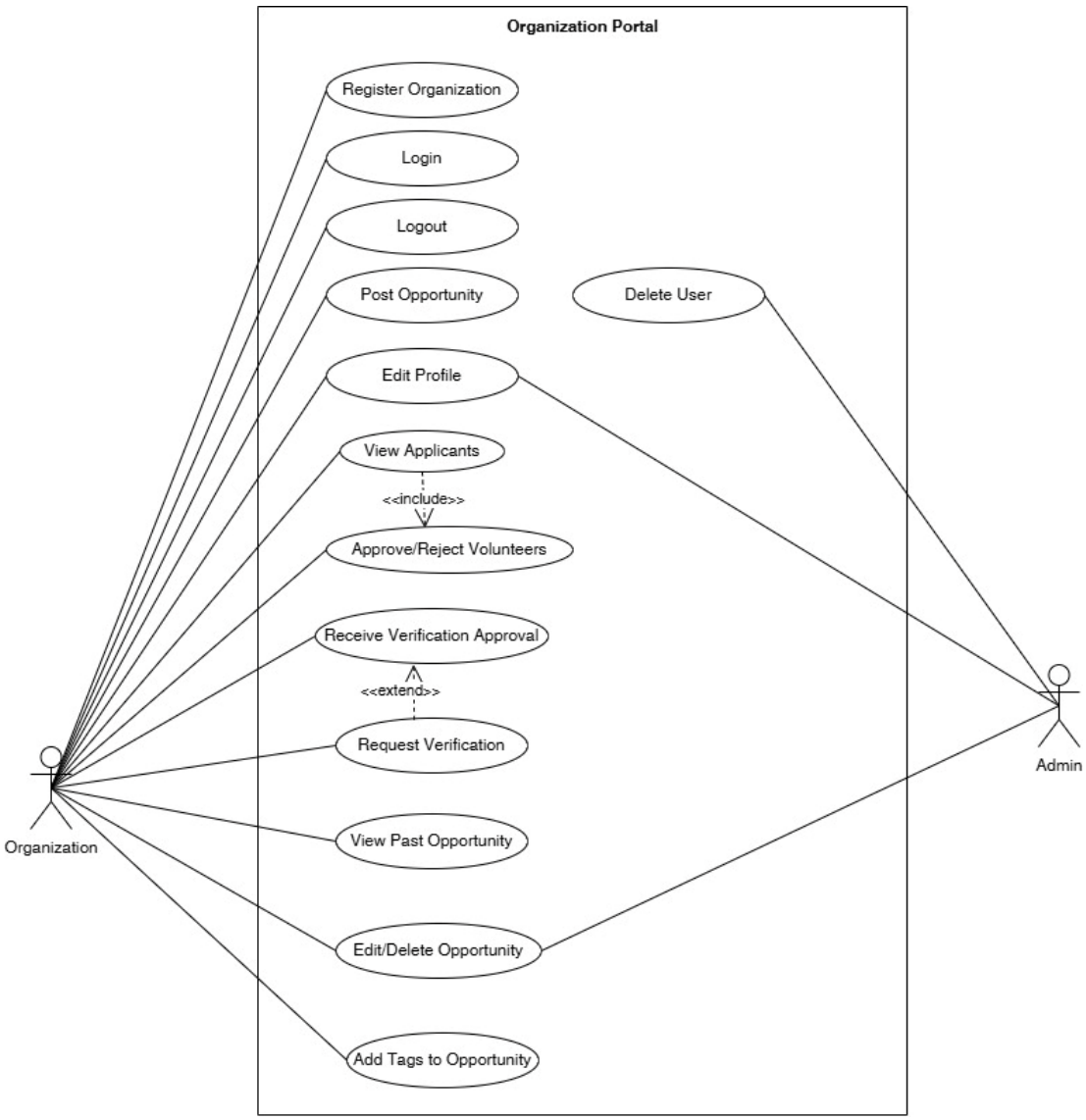


Figure 3.2: Use Case Diagram of the organization

The organization use case diagram represents the core functionalities available to organizations. These include registering and logging in, creating and managing volunteer opportunities, reviewing volunteer applications, and approving or marking participation. This use case diagram demonstrates how the system empowers organizations to efficiently manage volunteer requirements and interact with suitable candidates.

3.1.2 Non-Functional Requirements

The non-functional requirements outline the quality attributes critical to the Volunteer Recommendation System's performance, usability, and scalability.

- Security (JWT-based authentication and hashed passwords)
- Scalability (MongoDB for data storage, modular backend architecture)
- Usability (enhanced frontend interface, clear API responses)
- Performance (efficient querying and cosine similarity recommendation)
- Maintainability (well-structured codebase with modular services and clear documentation)
- Portability (runs on any OS supporting Node.js and MongoDB)
- Data Integrity (accurate storage and retrieval of user preferences and opportunity data)
- Extensibility (recommendation logic can be upgraded to include collaborative filtering or NLP)

3.2 Feasibility Study

The feasibility study assesses the viability of developing and implementing the Volunteer Recommendation System, considering technical, operational, economic, and schedule aspects.

3.2.1 Technical

- Uses mature open-source technologies: Node.js, Express.js, MongoDB, and JWT.
- Employs efficient cosine similarity algorithm for tag-based matching.
- Supported by team expertise in Node.js, MongoDB, and JavaScript.
- Ensures scalability with MongoDB's NoSQL architecture.

3.2.2 Operational

- Intuitive interface ensures user adoption across technical skill levels.
- Simplifies agency workflows with easy opportunity posting.
- Supports maintenance via Agile methodology and modular design.
- Manages data efficiently with MongoDB for profiles and tags.

3.2.3 Economic

- Minimizes costs using open-source Node.js, Express.js, and MongoDB.
- Currently runs on local development servers.
- Reduces maintenance costs with part-time developer support.
- Improves agency efficiency, lowering recruitment costs.
- Scales cost-effectively with MongoDB and cloud infrastructure.

3.2.4 Schedule

- Follows a three-month Agile iteration.
- Delivers key features (e.g., recommendation engine) within first iteration.
- Incorporates continuous integration and testing to identify and resolve issues early.
- Adjusts and refines system features based on supervisor and peer feedback after each milestone.
- Targets full system deployment and documentation completion by the end of the 3-month period.

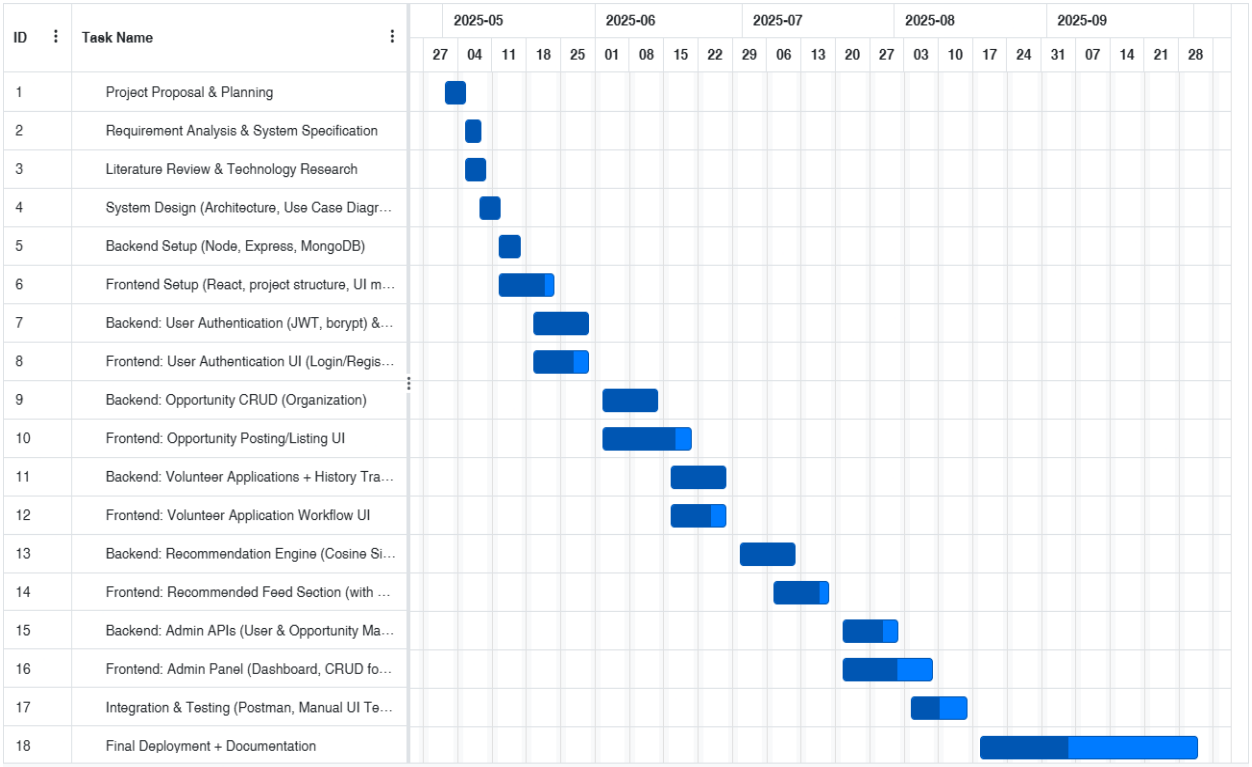


Figure 3.3: Gantt Chart

3.3 Analysis

Upon the completion of the feasibility study, we were able to determine that the project was feasible in every way and that there might not be any significant obstacles to its completion. After doing a system analysis, we chose to employ the Structured approach to develop the project.

3.3.1 Entity Relationship Diagram

The Entity-Relationship diagram models the core data structure of the system, defining the relationships among users, opportunities, applications, and recommendations. Tags are normalized to support multi-label classification of interests and roles. Each relationship is carefully designed to maintain data integrity and scalability.

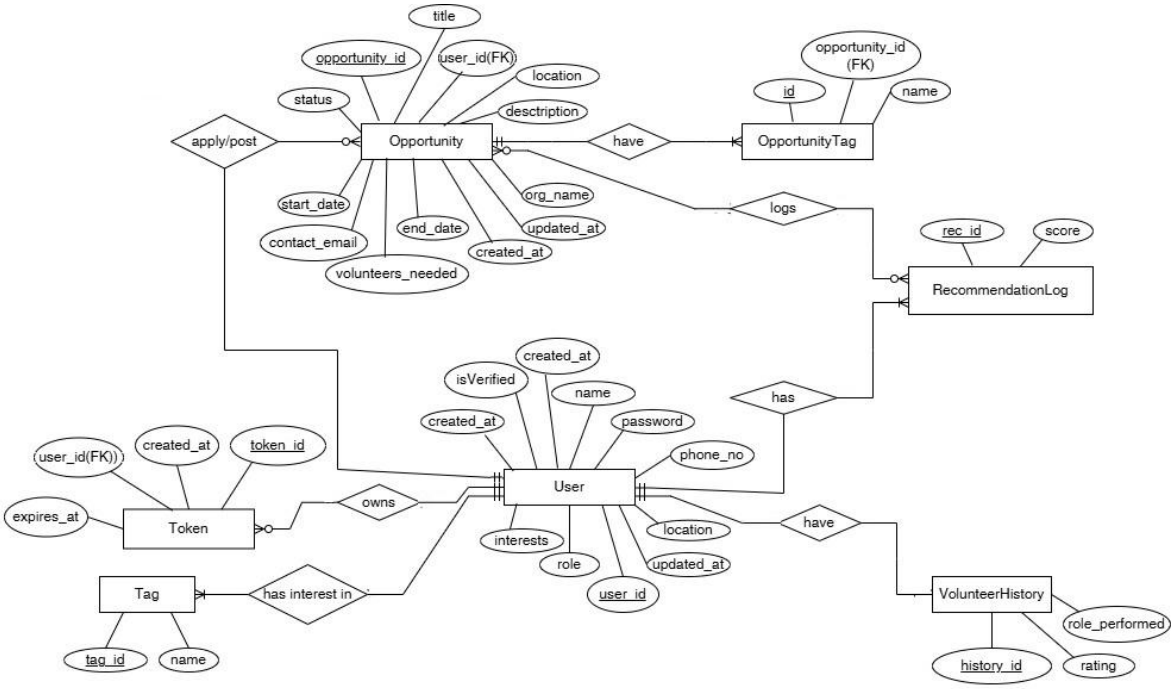


Figure 3.4: Entity-Relationship diagram

3.3.2 Data Flow Diagram

1. Context Level DFD

This diagram shows how volunteers and organizations interact with the system. Volunteers apply for opportunity and organizations post opportunities and receive volunteer's applications. The system manages these interactions and recommends appropriate opportunity for the users.

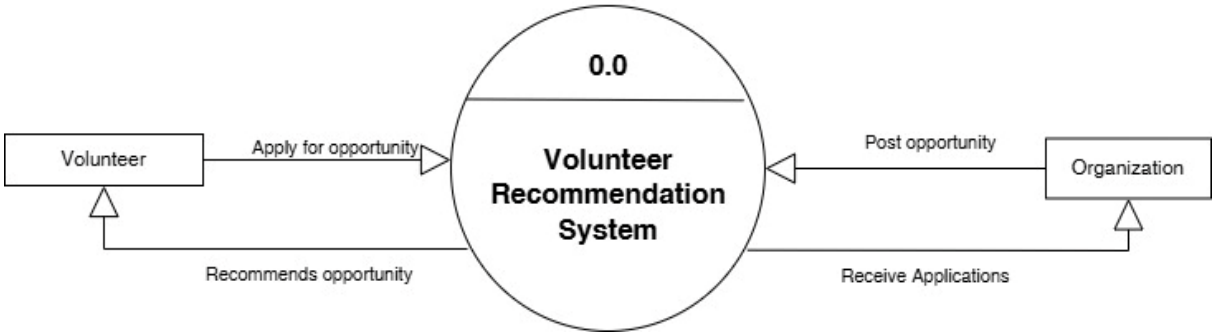


Figure 3.5: Context Level DFD

The central Volunteer Recommendation System acts as the processing hub that manages all these interactions from data collection, opportunity management, application processing, and recommendation generation. It ensures secure communication using authentication mechanisms and maintains consistency by storing all data (users, opportunities, and applications) in the database.

2. Level-1 DFD

The Level 1 Data Flow Diagram (DFD) decomposes the system into four main processes: user authentication and registration, opportunity management, application processing, and the recommendation engine. The diagram illustrates how external entities like Volunteers and Organizations interact with the system. Volunteers register or log in, apply for opportunities, and receive recommendations, while organizations post or manage volunteering opportunities and review applications.

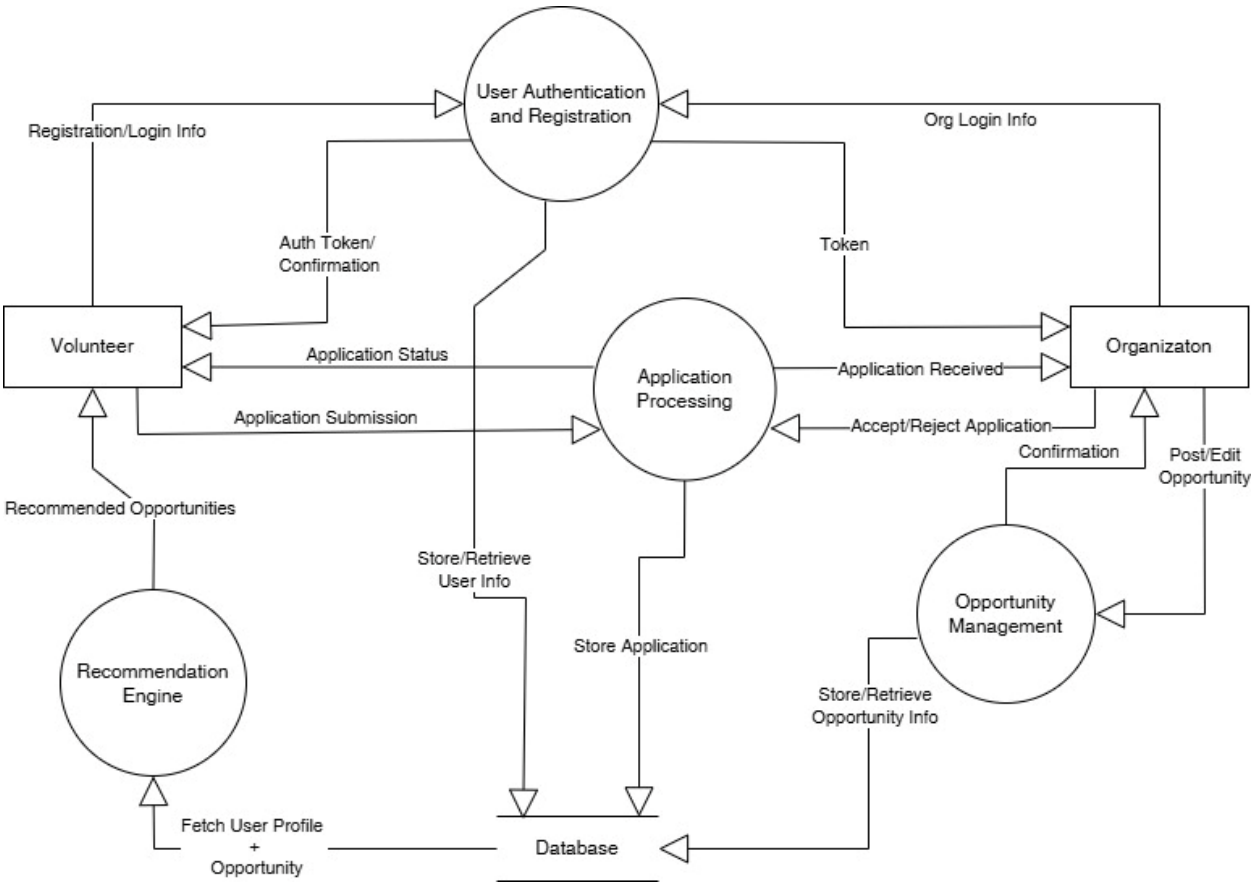


Figure 3.6: Level-1 DFD

Each process interacts with a central data store that holds user profiles, opportunity details, and application records. The flow of data among these components ensures that users are authenticated, opportunities are properly managed, applications are tracked, and personalized recommendations are generated based on user preferences and interaction history. This structured view provides clarity on how core system functions are interconnected and how data flows through the system.

3. Level-2 DFD

The Level 2 Data Flow Diagram (DFD) provides a detailed breakdown of the main processes defined in Level 1, showing the internal logic and data movement within each major function. It decomposes the system into sub-processes such as user registration and role assignment, volunteer application submission, opportunity approval, attendance marking, recommendation generation and session termination. Volunteers interact with processes to register, apply for opportunities, receive recommendations, and log out, while organizations manage opportunities, approve

applicants, and record attendance. Each sub-process communicates with specific data stores, including the User Database Table, Opportunity Database Table, Application Database Table, Tag Repository Table, and Token Store Table, ensuring seamless data exchange and integrity.

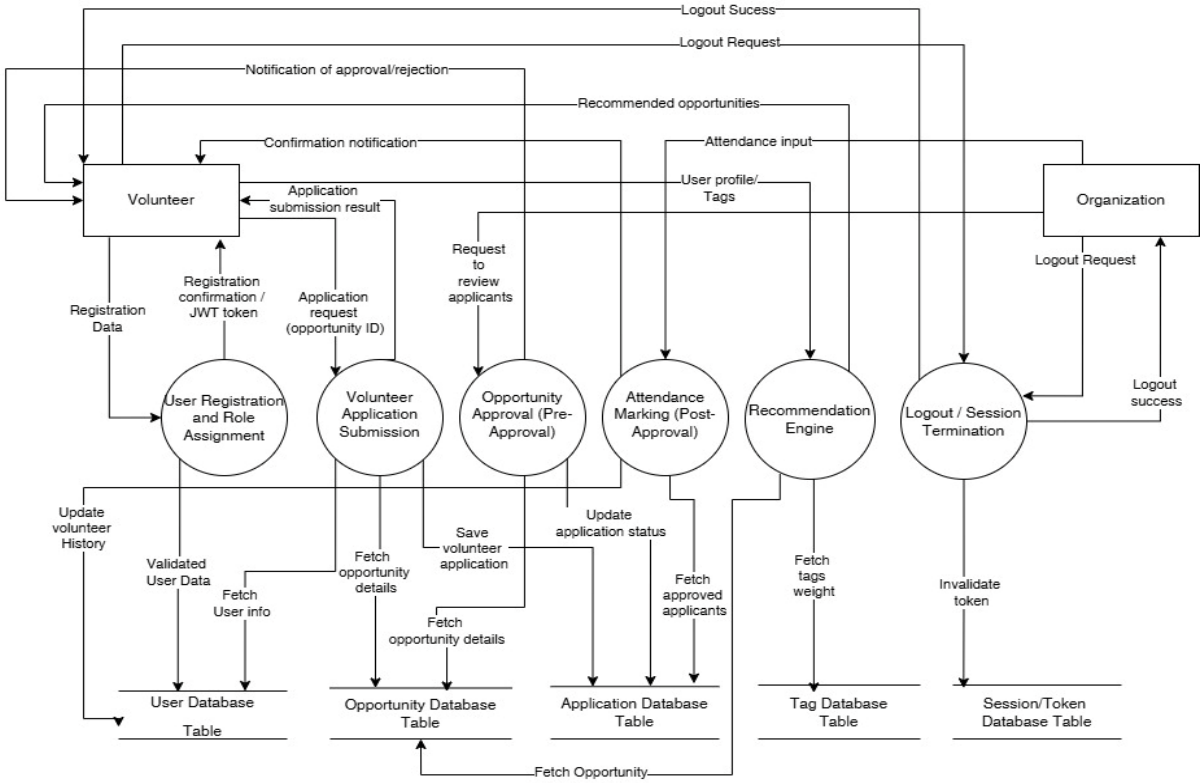


Figure 3.7: Level-2 DFD

The detailed data flows demonstrate how inputs from external entities are validated, stored, processed, and transformed into meaningful outputs such as recommendations, confirmations, and notifications. This level of detail enhances understanding of the internal system logic, depicting how individual operations collaborate to deliver the complete functionality of the volunteer recommendation platform.

3.4 System Overview

The system consists of user management, opportunity management, application processing, and a recommendation engine.

- User Management handles registration, authentication (via JWT), and role-based access for volunteers and organizations.
- Opportunity Management allows organizations to create and manage volunteering opportunities with associated tags that describe the nature of the work.
- Application Processing enables volunteers to apply for opportunities and tracks their application history for future reference.
- Recommendation Engine uses a content-based filtering approach powered by the cosine similarity algorithm to match volunteers with opportunities based on shared tags and user preferences.

3.5 User Types

Table 3.1: User Types

User Types	Description	Needs and Goals
Volunteer	Individual looking for opportunities to contribute.	Discover relevant events based on interests.
Organization	NGO or event host seeking volunteers.	Post opportunities and receive suitable matches.
System Admin	Maintains backend and manages user/admin privileges.	Ensure smooth functioning and manage permissions.

CHAPTER 4: SYSTEM DESIGN

4.1 Design

4.1.1 Architecture Design

The Volunteer Recommendation System is built on a client-server architecture comprising three main layers: the frontend, backend, and database. The frontend, developed using React, provides interfaces for user registration, opportunity browsing, applications, and personalized recommendations. It communicates with the backend via RESTful APIs.

The backend was built with Node.js and Express.js, and handled routing, authentication, and business logic. It includes dedicated controllers for user, opportunity, and application management, along with a recommendation module that uses cosine similarity to match volunteers with suitable opportunities based on their history and selected interests. Authentication is managed securely through JWT middleware.

MongoDB serves as the database, storing users, opportunities, applications, tags, and recommendation data. The system is modular, scalable, and designed to support future enhancements such as notification services or hybrid recommendation methods.

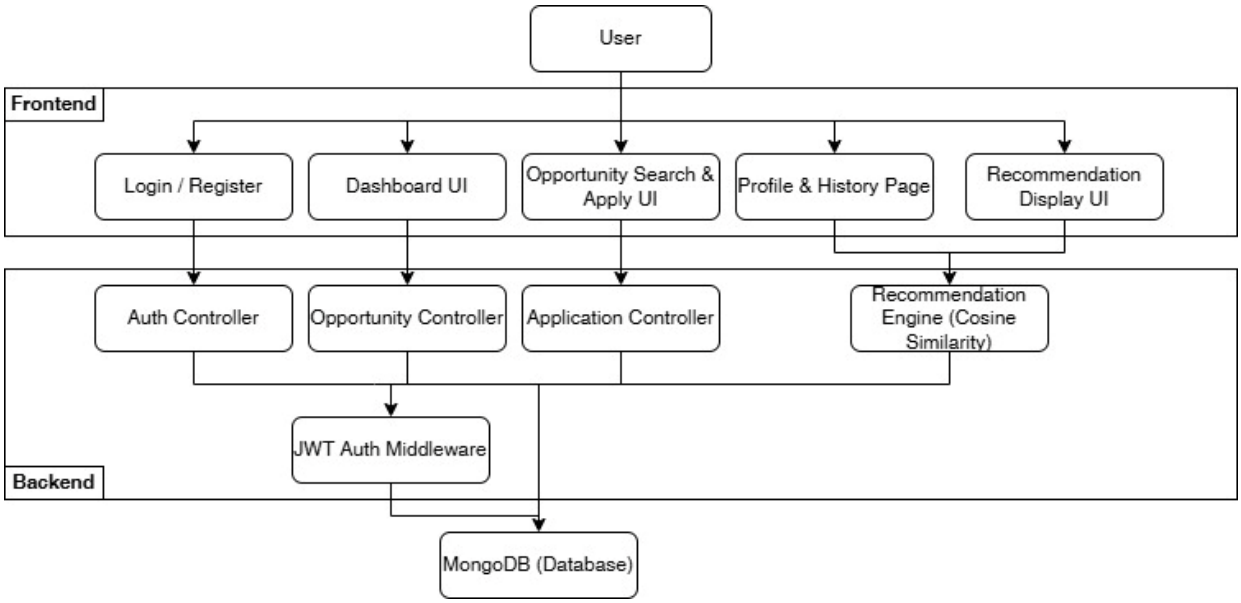


Figure 4.1: System Architecture

4.1.2 Database Design

The database was designed based on the entities identified in the system's ER diagram. Each entity and its relationships were transformed into collections in MongoDB, maintaining logical consistency and avoiding redundancy.

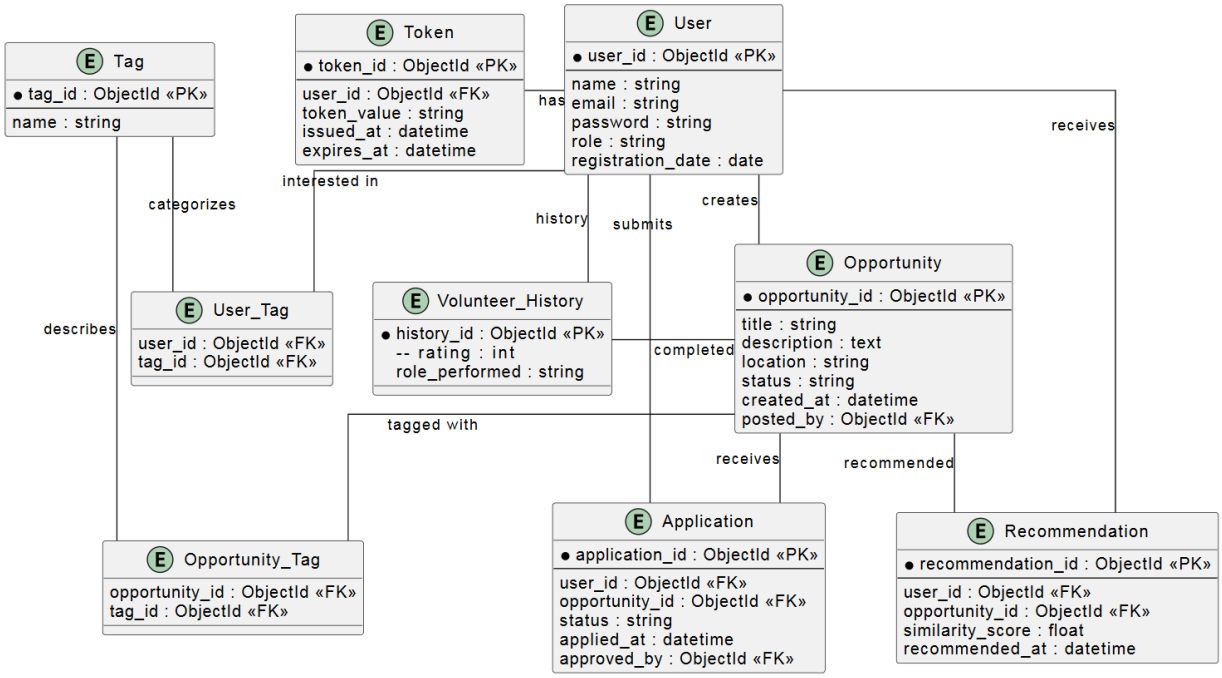


Figure 4.2: Database Design

4.1.3 Forms and Report Design

The forms represent the input screens used by users (volunteers, organizations, and administrators) to interact with the system, while the reports display processed information and data summaries. These designs ensure that users can easily navigate, enter data, and view meaningful results in a structured and user-friendly way.

1. Forms Design

The system includes multiple forms designed for different user roles to facilitate smooth interaction.

- **User Registration Form:**

  Allows volunteers and organizations to register with required details such as name, email, password, and role.

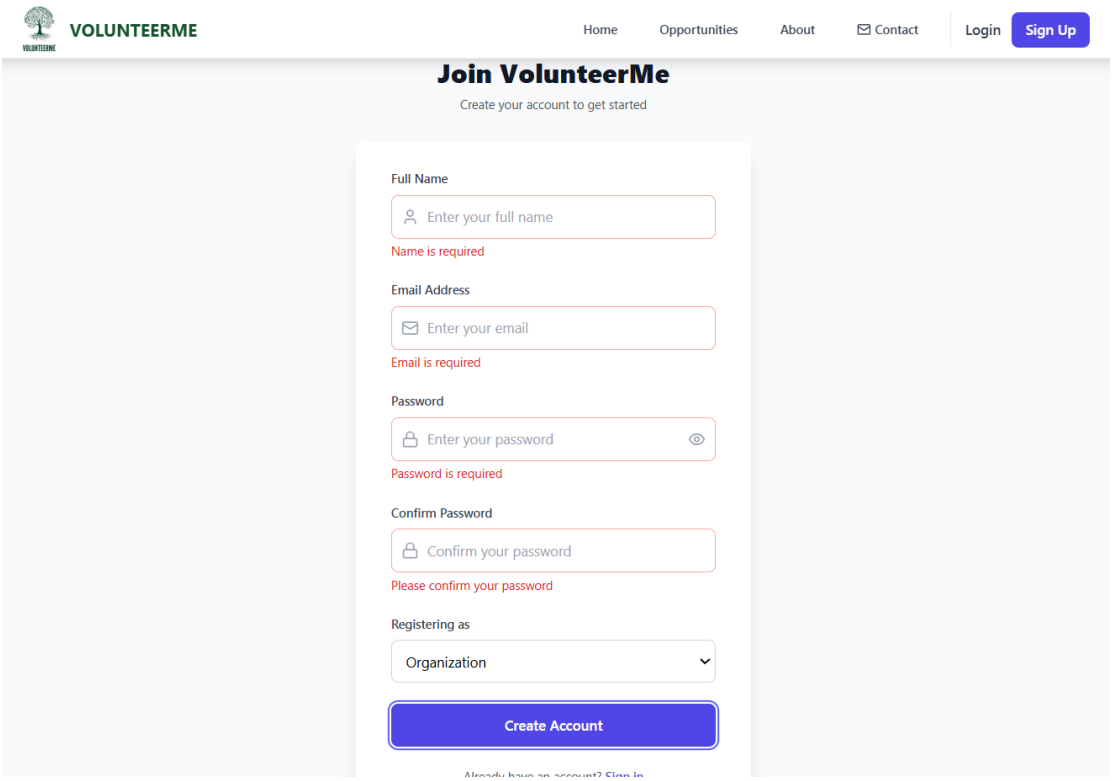


Figure 4.3: User Registration Form

- **Login Form:**

  Provides authentication access using email and password with role-based redirection (volunteer dashboard, organization dashboard, or admin panel).

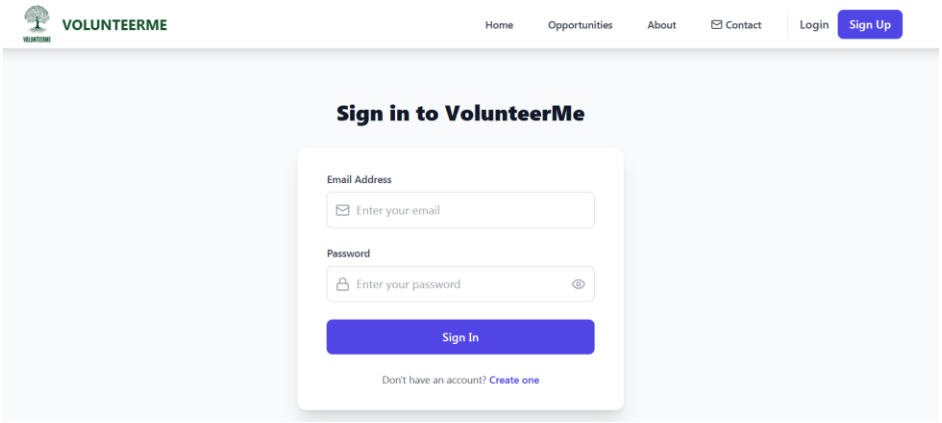


Figure 4.4: Login Form

- **Opportunity Creation Form (Organization):**

  Used by organizations to post new volunteer opportunities, including fields for title, description, tags, required volunteers, and dates.

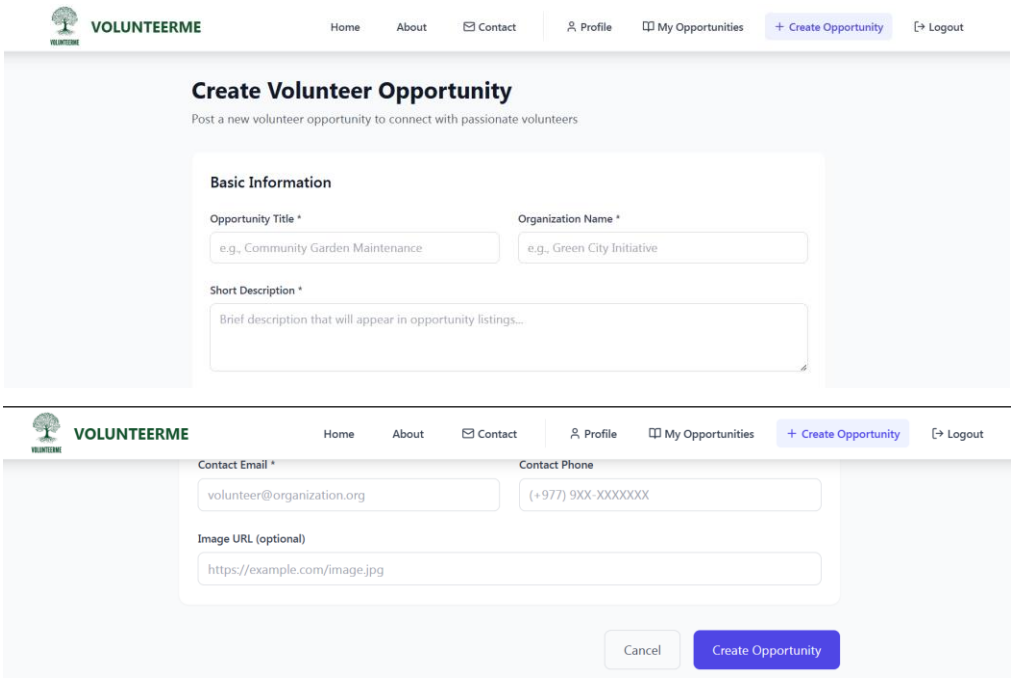


Figure 4.5: Opportunity Creation Form

2. Report Design

The report screens are designed to present system data in a meaningful and organized format. They include both textual and tabular summaries.

- **User Dashboard Reports:**
  Displays recommended opportunities based on user interests and their previous volunteering history.

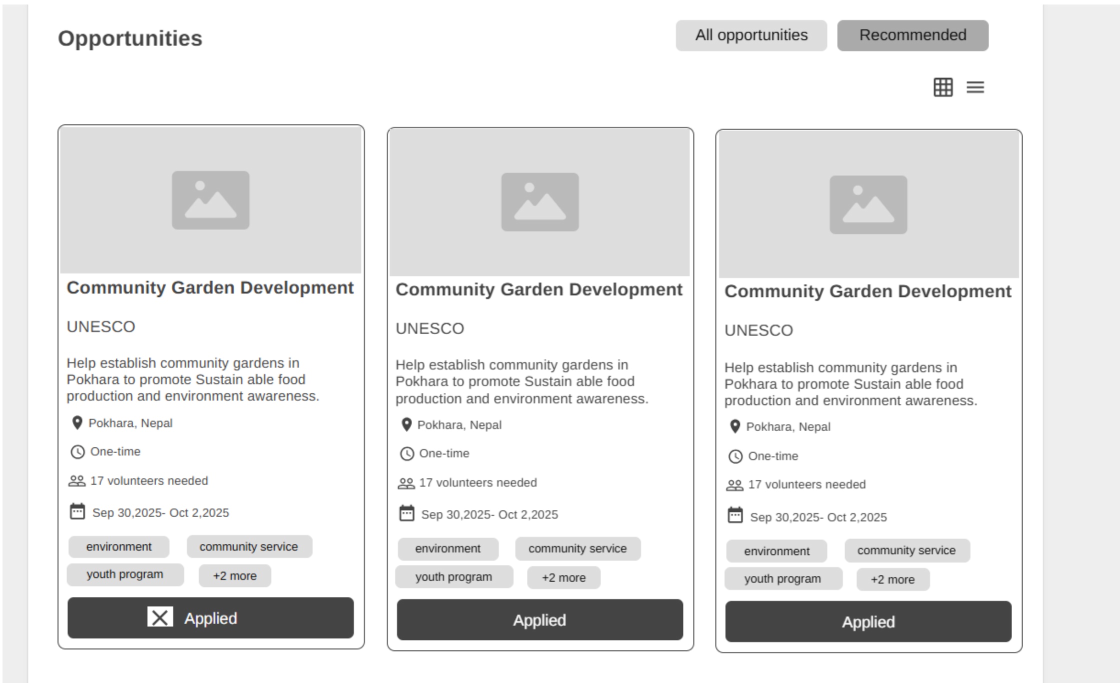


Figure 4.6: Wireframe of User Dashboard

- **Organization Dashboard Reports:**

  Shows posted opportunities and volunteer applications received.

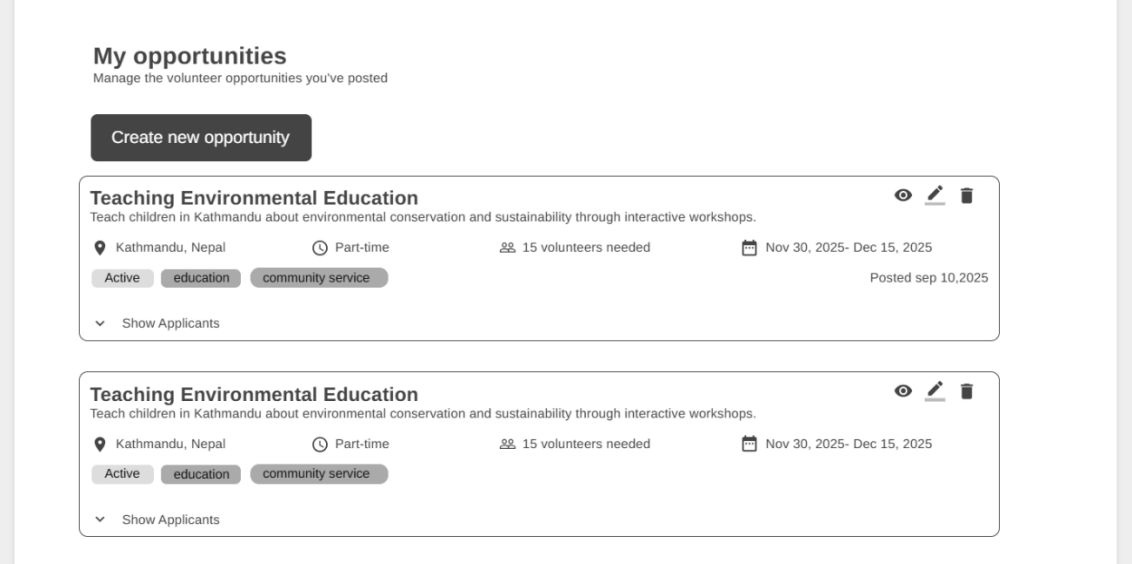


Figure 4.7: Wireframe of Organization Dashboard

- **Admin Reports:**

  Provides a summary of all users categorized by role (volunteer, organization, admin) and lists all opportunities with their current status.

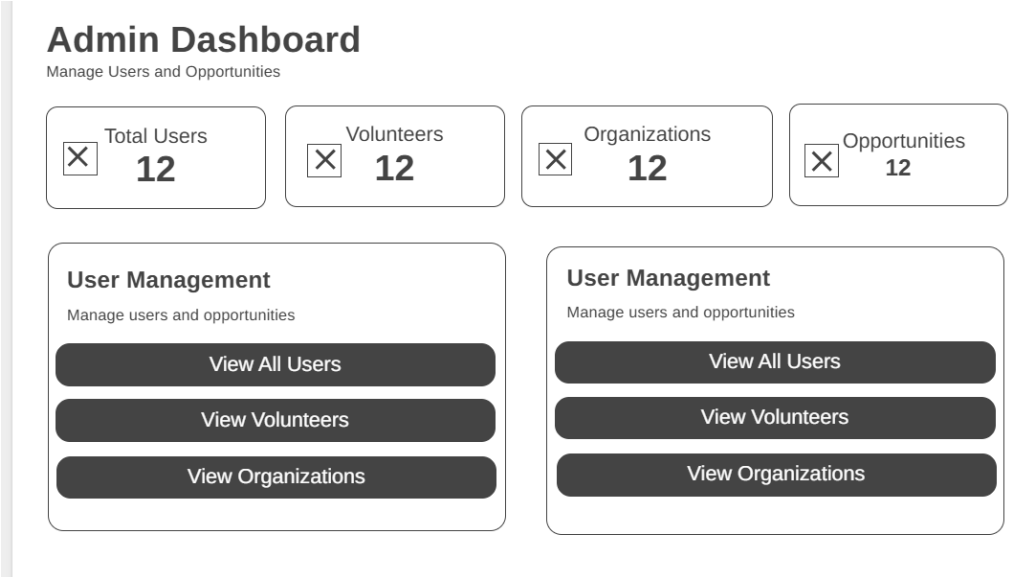


Figure 4.8: Wireframe of Admin Dashboard

4.1.4 Interface and Dialogue Design

Each user interacts with the system through intuitive interfaces and responsive dialogues that ensure efficient task completion. The system focuses on two primary user types: Volunteers and Organizations. Each flow ensures intuitive navigation, minimal steps for task completion, and clear system feedback at every stage.

- **Volunteer Flow**

Volunteers interact with the system to register, explore opportunities, apply for suitable roles, and view recommendations. The flow begins from the login/registration screen, where users create an account and select their interests. Once authenticated, volunteers are redirected to their dashboard, which displays both all available opportunities and a recommended section sorted by tag similarity scores. They can view details of each opportunity, apply directly, and track their application status. After successful completion of volunteering activities, volunteers can view their participation history within their profile. Volunteer Flow Steps:

1. Open the application → View Login/Registration Page.
2. Register as Volunteer → Select interest tags.
3. Login → Redirected to Volunteer Dashboard.
4. Explore All Opportunities or Recommended Opportunities.
5. Click an opportunity → View Opportunity Details.
6. Apply for opportunity → Receive Confirmation/Status Message.
7. Navigate to Profile Page to view past applications.
8. Logout from the system.

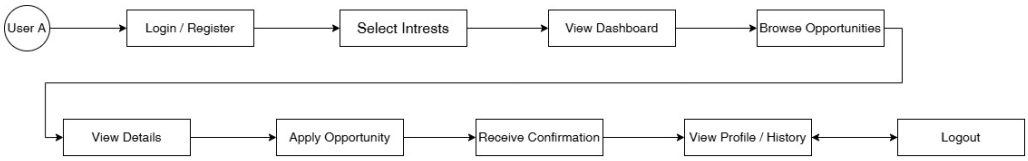


Figure 4.9: Volunteer Interface and Dialogue Design

- **Organization Flow**

Organizations use the system to post opportunities, manage applications, and mark attendance of volunteers. They start at the login page and are directed to their dashboard upon

authentication. From there, they can create new opportunities, view all posted ones and review volunteer applications. This structured interaction ensures smooth opportunity management and transparency in the volunteer approval process. Organization Flow Steps:

1. Open the application → View Login Page.
2. Login as Organization → Redirected to Organization Dashboard.
3. Select Create Opportunity to post a new volunteering role.
4. View Posted Opportunities list and manage applications.
5. Approve or reject applicants under Manage Applications.
6. After completion, mark attendance and update volunteer status.
7. Logout from the system.

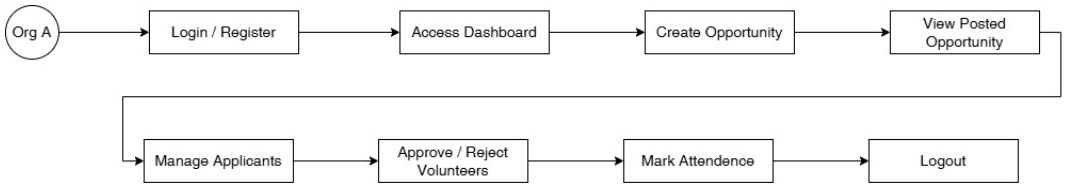


Figure 4.10: Organization Interface and Dialogue Design

4.2 Algorithm Details

The core recommendation algorithm uses cosine similarity to match volunteers with opportunities based on tags and volunteer history.

4.2.1 Cosine Similarity with TF-IDF Weighting

To compare a user's tag profile with an opportunity's tag vector, the system uses the cosine similarity algorithm with TF-IDF weighting. Cosine similarity measures the cosine of the angle between two vectors, which reflects how similar they are regardless of length.

The formula is:

$$\text{Cosine\_similarity} = \cos(\theta) = \frac{A \cdot B}{|A||B|}$$

Where:

A is the user profile vector

B is the opportunity tag vector

A·B is the dot product of the two vectors

‖A‖ and ‖B‖ are the magnitudes (Euclidean norms)

By incorporating TF-IDF weighting:

- Term Frequency (TF): Tags that appear more frequently in a user's profile or in an opportunity are given proportionally higher weight.
- Inverse Document Frequency (IDF): Tags that are common across many opportunities (e.g., "community") are down-weighted, while rarer, distinctive tags (e.g., "robotics") are emphasized.

This combination ensures that recommendations prioritize opportunities with unique, meaningful overlaps rather than generic matches.

4.2.2 Vector Construction (User Profile & Opportunities)

For cosine similarity to work, both volunteers and opportunities are represented as TF-IDF vectors using a shared tag vocabulary.

- Vocabulary Creation: A master list of all unique tags is built from user interests and existing opportunities.
- User Profile Vector: Constructed using tags the user selected at signup plus tags from past volunteering history. The frequency of each tag is weighted by TF-IDF.
- Opportunity Vector: Each opportunity is converted into a vector of the same size, where each tag is weighted according to its TF-IDF score.
- Similarity Computation: Cosine similarity is then applied between the user vector and every opportunity vector. Opportunities are ranked by similarity score and presented in descending order.

4.2.3 Pseudocode & Flowchart

1. **Start:**
   FUNCTION recommendOpportunities(user, opportunities):
   DEFINE cosineSimilarity(vecA, vecB):
   dot ← 0, normA ← 0, normB ← 0
   FOR i FROM 1 TO length(vecA):

dot ← dot + (vecA[i] * vecB[i])

normA ← normA + vecA[i]^2

normB ← normB + vecB[i]^2

IF normA = 0 OR normB = 0:

RETURN 0

RETURN dot / (sqrt(normA) * sqrt(normB))

2. **Build tag vocabulary**

   vocab ← all unique tags from user.interests and all opportunities.tags

3. **Compute document frequency (df)**

   FOR each tag IN vocab:

   df[tag] ← number of opportunities (and user) that contain tag

4. **Compute inverse document frequency (idf)**

   FOR each tag IN vocab:

   idf[tag] ← ln((total_documents) / (1 + df[tag]))

5. **Build TF-IDF vectors**

   FUNCTION buildVector(tags):

   FOR each tag IN vocab:

   tf ← frequency(tag IN tags) / total_tags_in_document

   vector[tag] ← tf * idf[tag]

   RETURN vector

   userVector ← buildVector(user.interests)

6. **Compute similarity for each opportunity**

   FOR each opportunity IN opportunities:

   opVector ← buildVector(opportunity.tags)

   score ← cosineSimilarity(userVector, opVector)

   store (opportunity, score)

7. **Sort opportunities by similarity score (descending)**

   sortedList ← sort(opportunities by score DESC)

   RETURN sortedList

   END FUNCTION

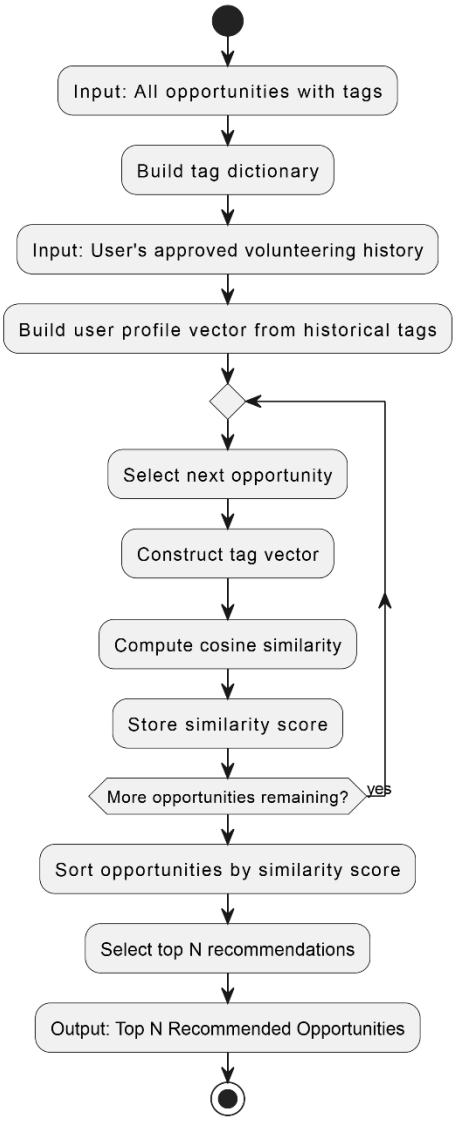


Figure 4.1: Flowchart: Cosine Similarity Algorithm

The flowchart illustrates the core logic of the content-based recommendation system used in the Volunteer Recommendation System. It begins by collecting all available volunteer opportunities and building a tag dictionary. Simultaneously, the system analyzes the user's volunteering history to generate a user profile vector based on previously engaged tags. Then, for each opportunity, a tag vector is created and compared with the user vector using the cosine similarity algorithm. The resulting similarity scores are used to rank the opportunities, and the top-matching opportunities are selected and recommended to the user. This process ensures personalized and relevant volunteer suggestions based on the user’s past activities and interests.

4.2.4 Demo of the Algorithm

Scenario: 1

user (new volunteer) and 4 opportunities.

N = number of opportunities + 1 (we include the user as a document)

df(tag) = number of documents (opportunities and user profile) containing the tag

idf(tag) = ln(N / (1 + df(tag))) (natural log, smoothing +1)

tf(tag, doc) = count(tag in doc) / total_tags_in_doc

tfidf = tf * idf

cosine(A,B) = (A·B) / (||A|| * ||B||)

Step 1: Raw data, user & opportunities

User interests (selected at signup):

- User U: ["education", "teaching", "child-care"]

Opportunities (4):

- Op1: "Teach at Orphanage" - tags: ["education", "teaching"]
- Op2: "Health Camp" - tags: ["health", "community"]
- Op3: "Tree Plantation" - tags: ["environment", "community"]
- Op4: "Child Care Drive" - tags: ["child-care", "education", "community"]

Step 2: Build vocabulary (all unique tags)

Collect tags from the user + all opportunities: (6 unique tags)

vocab = [education, teaching, child-care, health, community, environment]

Step 3: Number of documents N

We count each opportunity as one document, and also the user profile as one document.

N = number_of_opportunities + 1 = 4 + 1 = 5

Step 4: Document Frequency (df) for each tag

Count in how many documents (opportunities + user) the tag appears.

Table 4.1: Document Frequency Table

Tag	Appears in (docs)	df
education	Op1, Op4, User	3
teaching	Op1, User	2
child-care	Op4, User	2
health	Op2	1
community	Op2, Op3, Op4	3
environment	Op3	1

So, df(education) = 3, df(teaching) = 2, df(child-care) = 2, df(health) = 1, df(community) = 3 and df(environment) = 1

Step 5: Compute IDF for each tag

Using: idf(tag) = ln(N / (1 + df(tag)))

- idf(education) = ln(5 / (1+3)) = ln(5/4) = ln(1.25) = 0.223144
- idf(teaching) = ln(5 / (1+2)) = ln(5/3) ≈ 0.510826
- idf(child-care) = ln(5/3) ≈ 0.510826
- idf(health) = ln(5 / (1+1)) = ln(5/2) ≈ 0.916291
- idf(community) = ln(5/4) = 0.223144
- idf(environment) = ln(5/2) = 0.916291

Table 4.2: IDF Table

Tag	df	idf (ln(N/(1+df)))
education	3	0.223144
teaching	2	0.510826
child-care	2	0.510826

health	1	0.916291
community	3	0.223144
environment	1	0.916291

Step 6: Compute TF per document

TF = count(tag in doc) / total_tags_in_doc.

User U (total tags = 3):

- tf_U(education) = 1 / 3 = 0.333333
- tf_U(teaching) = 1 / 3 = 0.333333
- tf_U(child-care) = 1 / 3 = 0.333333
- tf_U(health) = 0
- tf_U(community) = 0
- tf_U(environment) = 0

Opportunities:

Op1 (total tags = 2): education, teaching

- tf_Op1(education) = 1/2 = 0.5
- tf_Op1(teaching) = 0.5
- others = 0

Table 4.3: Term Frequency Table

Doc \ Tag	education	teaching	child-care	health	community	environment
User (U)	0.333333	0.333333	0.333333	0	0	0
Op1	0.5	0.5	0	0	0	0
Op2	0	0	0	0.5	0.5	0
Op3	0	0	0	0	0.5	0.5

Op4	0.333333	0	0.333333	0	0.333333	0

Step 7: Compute TF–IDF vectors (TF * IDF)

Multiply TF * IDF for each cell.

Table 4.4: TF-IDF Table

Doc \ Tag	education	teaching	child-care	health	community	environment
User U	0.074381	0.170275	0.170275	0	0	0
Op1	0.111572	0.255413	0	0	0	0
Op2	0	0	0	0.458146	0.111572	0
Op3	0	0	0	0	0.111572	0.458146
Op4	0.074381	0	0.170275	0	0.074381	0

Step 8: Compute vector norms (||A||) and dot products, then cosine similarity

We compute, for each Opportunity Op_i:

- dot = sum_over_tags(TFIDF_user(tag) * TFIDF_op(tag))
- norm_user = sqrt(sum_over_tags(TFIDF_user(tag)^2))
- norm_op = sqrt(sum_over_tags(TFIDF_op(tag)^2))
- cosine = dot / (norm_user × norm_op)

First compute norm_user (same for all comparisons):

User TF–IDF values:

- education = (0.074381) square = 0.005533
- teaching = (0.170275) square = 0.028994
- child-care = (0.170275) square = 0.028994
- others = 0

Sum squares = 0.005533 + 0.028994 + 0.028994 = 0.063521

norm_user = sqrt(0.063521) = 0.252029

Op1 calculations:

Op1 TF–IDF: education=0.111572, teaching=0.255413

- op1 squares: 0.111572^2 = 0.012450 ; 0.255413^2 = 0.065234
- sum = 0.012450 + 0.065234 = 0.077684
- norm_op1 = sqrt(0.077684) = 0.278781

dot(User, Op1):

- education product: 0.074381 × 0.111572 = 0.008300
- teaching product: 0.170275 × 0.255413 = 0.043477
- child-care product: 0.170275 × 0 = 0
- dot = 0.008300 + 0.043477 = 0.051777

cosine(User, Op1):

cosine = 0.051777 / (0.252029×0.278781) = 0.051777 / 0.070292 = 0.7369

Similarly calculating with respect to other opportunities,

Table 4.5: Cosine Similarity Table

Opportunity	Cosine similarity
Op1	0.7369
Op4	0.6846
Op2	0.0000
Op3	0.0000

Step 9: Final similarity scores and ranking

1. Op1 = Teach at Orphanage = score 0.7369
2. Op4 = Child Care Drive = score 0.6846
3. Op2 = Health Camp = score 0.0000
4. Op3 = Tree Plantation = score 0.0000

CHAPTER 5: IMPLEMENTATION & TESTING

5.1 Implementation

5.1.1 Tools Used

Table 5.1: Tools and Technologies Used

Category	Tool/Technology	Purpose
Backend Framework	Node.js (Express.js)	Building RESTful APIs and handling server logic
Authentication	JSON Web Tokens (JWT)	Secure user authentication and role-based access control
Database	MongoDB (Mongoose ODM)	Storing user, opportunity, and application data
Recommendation Algorithm	TF-IDF + Cosine Similarity	Matching users to relevant opportunities based on tags and history
Utility Libraries	dotenv, bcryptjs, concurrently	Environment variables, password hashing, and running multiple services
API Testing	Postman	Testing backend API endpoints
Development Environment	Visual Studio Code	Source code editing
Version Control	Git & GitHub	Source code versioning and collaboration
Frontend Framework	React.js	Building interactive frontend interface
Diagramming Tool	Draw.io / PlantUML	Designing and visualizing system diagrams

5.1.2 Implementation Details of Modules

1. Authentication & Authorization Module

The Authentication & Authorization Module is responsible for secure user registration, login, and session management. It implements password hashing using bcrypt, JWT token-based authentication, and email verification using 6-digit codes. The module enforces role-based access control (volunteer, organization, admin) and validates organization registrations by requiring .org

domain email addresses. Users are automatically logged in after registration, but must verify their email to access protected features like applying to opportunities. The module also supports optional email verification, allowing users to skip verification and sign in directly if they choose.

Pseudocode:

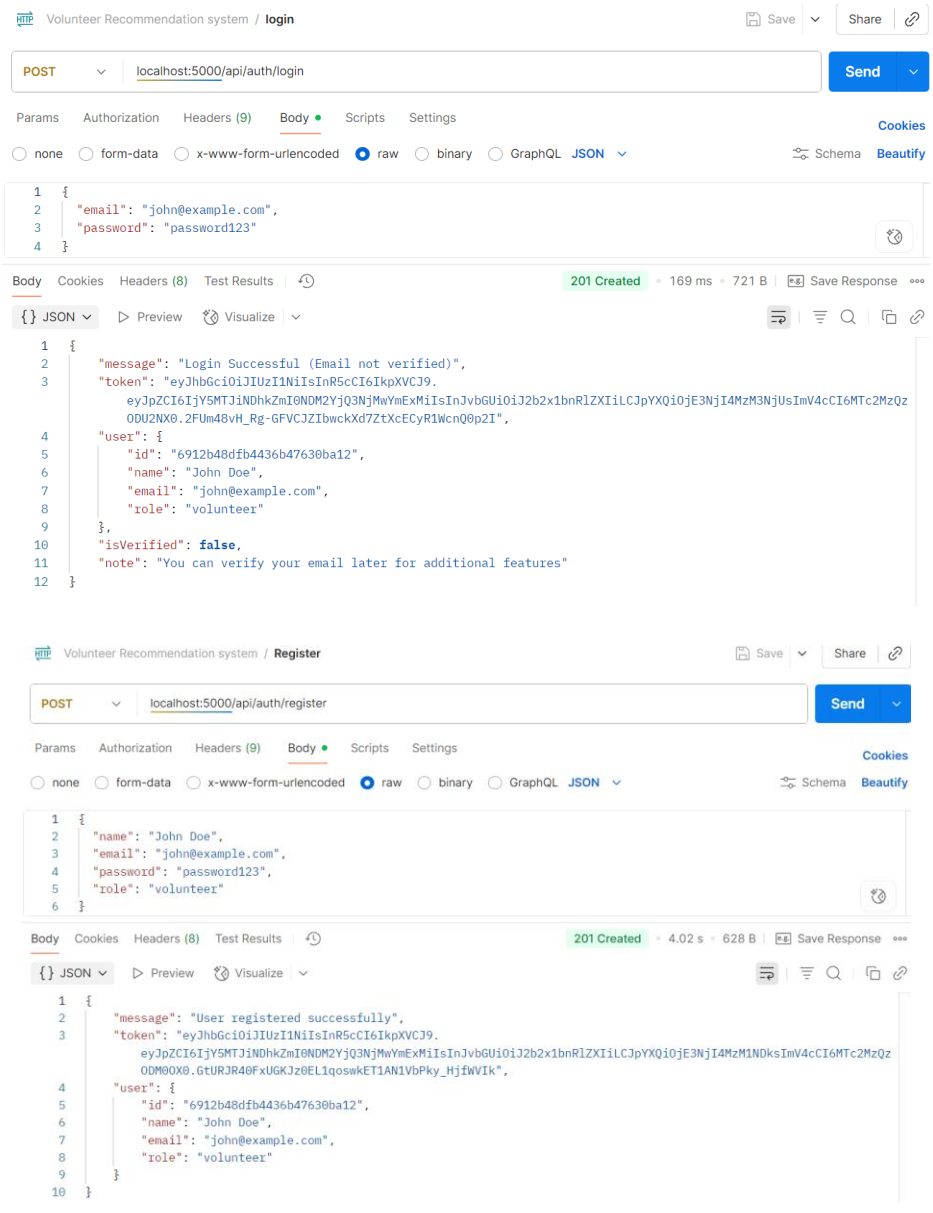
```
FUNCTION registerUser(name, email, password, role, interests):
    IF userExists(email) THEN RETURN "User already exists"
    IF role == "organization" AND invalidOrgEmail(email) THEN RETURN "Invalid email domain"
    hashedPassword = hash(password)
    code = generateVerificationCode()
    SAVE user(name, email, hashedPassword, role, interests, code, verified=false)
    token = issueJWT(user)
    SEND verificationEmail(email, code)
    RETURN "Registration successful"
FUNCTION loginUser(email, password):
    user = findUser(email)
    IF invalidCredentials(user, password) THEN RETURN "Login failed"
    token = issueJWT(user)
    RETURN "Login successful", token, userDetails
FUNCTION verifyEmail(email, code):
    user = findUser(email)
    IF codeMatches(user, code) THEN markVerified(user)
    RETURN "Email verified successfully"
```

FUNCTION logoutUser(token):

user = findUserByToken(token)

REMOVE token FROM user.tokens

RETURN "Logout successful"



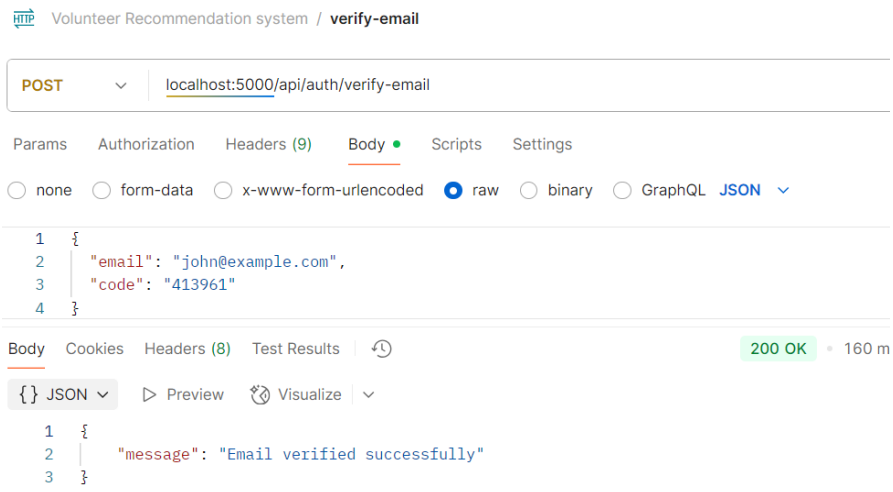


Figure 5.1: Authentication & Authorization Module Example

2. User Management Module

The User Management Module handles volunteer and organization profile management, including personal information updates, interest tracking, and volunteer history. The module stores user data such as name, email, phone, location, bio, skills, interests, and availability. It tracks volunteer history including applied and approved opportunities, and maintains saved opportunities for future reference. The module differentiates between volunteer and organization roles, allowing organizations to manage their posted opportunities while volunteers manage their applications and profile information.

Pseudocode:

```
FUNCTION updateUserProfile(userId, updatedFields):
    user = FIND_USER_BY_ID(userId)
    IF NOT user:
        RETURN error("User not found")
    UPDATE user WITH updatedFields
    SAVE user
    RETURN success(user)
```

```
FUNCTION getUserApplications(userId):
    user = FIND_USER_BY_ID(userId)
    IF user.role != 'volunteer':
        RETURN error("Unauthorized access")
    applications = FIND_OPPORTUNITIES_WITH_APPLICANT(userId)
    RETURN applications
FUNCTION manageSavedOpportunities(userId, opportunityId, action):
    user = FIND_USER_BY_ID(userId)
    IF action == "save":
        ADD opportunityId TO user.savedOpportunities
    ELSE IF action == "remove":
        REMOVE opportunityId FROM user.savedOpportunities
    SAVE user
    RETURN success
FUNCTION addVolunteerHistory(userId, opportunityInfo):
    user = FIND_USER_BY_ID(userId)
    APPEND opportunityInfo TO user.volunteerHistory
    SAVE user
    RETURN success
```

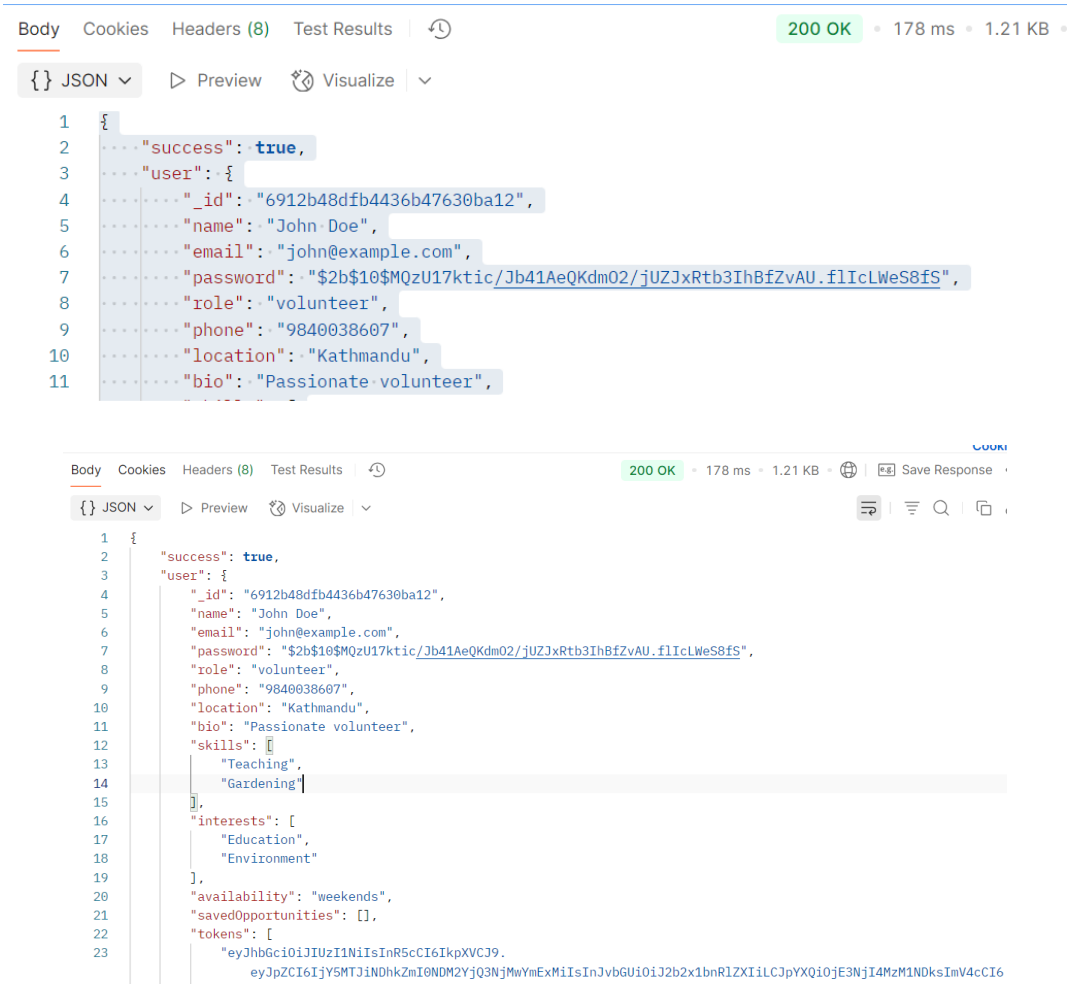


Figure 5.2: User Management Module Example

3. Email Notification System

The Email Notification System facilitates automated email communication throughout the application using Nodemailer with Gmail SMTP. The module sends verification codes during user registration, application notifications to organizations when volunteers apply, approval notifications to volunteers when selected, and attendance confirmation emails. The system is configured to work with Gmail app passwords for secure authentication and handles email sending failures gracefully without breaking the main application flow.

Pseudocode:

FUNCTION sendEmail(to, subject, text):

transporter ← CONFIGURE_NODEMAILER(EMAIL_SERVICE, EMAIL_USER, EMAIL_PASS)

mail ← {from: EMAIL_USER, to: to, subject: subject, text: text}

TRY SEND_MAIL(mail)

CATCH error → LOG("Email failed: " + error)

FUNCTION sendVerificationEmail(email, code):

subject ← "Verify your VolunteerMe account"

text ← "Your verification code: " + code

CALL sendEmail(email, subject, text)

FUNCTION sendApplicationNotification(orgEmail, volunteerName, opportunity):

subject ← "New Application for " + opportunity

text ← "Volunteer " + volunteerName + " applied for your opportunity."

CALL sendEmail(orgEmail, subject, text)

FUNCTION sendApprovalEmail(volEmail, volunteerName, opportunity, orgName, date):

subject ← "Approved for " + opportunity

text ← "Hello " + volunteerName + ", approved by " + orgName + " on " + date

CALL sendEmail (volEmail, subject, text)

Figure 5.3: Email Notification Module Output

4. Recommendation Engine

The Recommendation Engine matches volunteers to suitable opportunities using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and cosine similarity. The system builds a vocabulary from all unique tags in user interests and opportunity tags, computes document frequency and IDF weights for each tag, creates TF-IDF vectors for user profiles and opportunities, calculates cosine similarity scores between user interests and opportunity tags, and returns opportunities sorted by similarity score in descending order. This personalized recommendation system helps volunteers discover opportunities that align with their interests.

Pseudocode:

```
FUNCTION recommendOpportunities(user, opportunities):
    vocab ← UNIQUE_TAGS(user.interests + opportunities.tags)
    idf ← CALCULATE_IDF(vocab, user + opportunities)
    userVector ← TFIDF_VECTOR(user.interests, idf)
    FOR EACH opportunity IN opportunities:
        opVector ← TFIDF_VECTOR(opportunity.tags, idf)
        score ← COSINE_SIMILARITY(userVector, opVector)
        STORE (opportunity, score)
    RETURN SORT_BY_SCORE_DESC(opportunities)
FUNCTION getRecommendedOpportunities(userId):
    user ← FIND_USER(userId)
    opportunities ← GET_ALL_OPPORTUNITIES()
    RETURN recommendOpportunities(user, opportunities)
```

```
Recommendations: {
    opportunities: [
        {
            _id: "507f191e810c19729de860ea",
            title: "Community Garden",
            tags: ["environment", "gardening"],
            score: 0.85,
            organization: {name: "Green Org"},
            ...
        },
        {
            _id: "507f191e810c19729de860eb",
            title: "Tree Planting",
            tags: ["environment", "outdoor"],
            score: 0.72,
            organization: {name: "Eco Org"},
            ...
        }
    ],
    count: 2
}
```

Figure 5.4: Recommendation Module Output

5. API & Routing Structure

The API & Routing Structure module organizes all API endpoints following RESTful conventions using Express.js routers. The module structures routes by functionality (auth, user, opportunity, admin), applies appropriate middleware for authentication and verification at the routing layer, implements role-based restrictions, and ensures consistent API response formats. Routes are organized into separate files for maintainability, with clear separation of concerns between authentication, user management, opportunity management, and administrative functions.

Pseudocode:

FUNCTION setupAuthRoutes:

CREATE router

ADD route for user registration

ADD route for user login

ADD protected route to fetch current user info

ADD protected route for logout

ADD route for email verification

RETURN router

FUNCTION setupUserRoutes:

CREATE router

ADD protected route to get saved opportunities

ADD protected route to save new opportunity

ADD protected route to remove saved opportunity

ADD protected route to fetch user interests

ADD protected route to update profile details

ADD protected route to test email sending

RETURN router

FUNCTION setupOpportunityRoutes:

CREATE router

ADD route to fetch all opportunities

ADD protected route for personalized recommendations

ADD protected route to view user's applications

ADD protected route for organization to view own postings

ADD route to get details or tags of a specific opportunity

ADD protected routes for organizations to:

- create opportunity

- update opportunity

- delete opportunity

- manage applicants (approve/reject)

- mark opportunity as completed

ADD route for searching opportunities

RETURN router

FUNCTION setupAdminRoutes:

CREATE router

APPLY authentication and admin authorization

ADD routes for managing:

- users (create, view, update, delete)

- opportunities (create, view, update, delete)

- system statistics

RETURN router

FUNCTION setupAppRoutes:

CREATE main application

LINK all sub-route groups (auth, user, opportunity, admin, tags, contact)

ENABLE global error handling

RETURN app

```
API Routes Structure: {
    auth: {
        POST: "/api/auth/register",
        POST: "/api/auth/login",
        GET: "/api/auth/me",
        POST: "/api/auth/logout",
        POST: "/api/auth/verify-email"
    },
    user: {
        GET: "/api/user/saved-opportunities",
        POST: "/api/user/saved-opportunities",
        DELETE: "/api/user/saved-opportunities/:id",
        GET: "/api/user/interests",
        PUT: "/api/user/profile"
    },
    opportunities: {
        GET: "/api/opportunities",
        GET: "/api/opportunities/recommendations",
        GET: "/api/opportunities/:id",
        POST: "/api/opportunities",
        POST: "/api/opportunities/:id/apply",
        POST: "/api/opportunities/:id/approve-applicant"
    },
    admin: {
        GET: "/api/admin/users",
        GET: "/api/admin/opportunities",
        GET: "/api/admin/stats"
    }
}
```

Figure 5.5: API & Routing Structure Output

5.2 Testing

5.2.1 Unit Testing

Unit testing involved testing individual components or modules of the application system to ensure that each module functions correctly in an isolated environment. All of the modules were tested independently using test cases designed to verify both valid and invalid inputs.

1. Authentication & Authorization Module

Table 5.2: Unit Testing of User Registration

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	All valid inputs Name: Ram Sharma Email: ramsha@gmail.comPassword: ramsharma Role: volunteer Interests: Education, Environment	Registration successful	Prompt with "User Created Successfully" dialogue and Redirect to Verify Email page	Test successful
2	All fields empty	Registration unsuccessful	Focus on Empty Field (Name required)	Test successful
3	Register with already registered email Email: ramsha@gmail.com (already exists)	Registration unsuccessful	"User already exists" error message	Test successful
4	Invalid email formatEmail: ramshagmail.com	Registration unsuccessful	"Email is invalid" error message	Test successful
5	Passwords do not match Password: ramsharma Confirm Password: ramsharm	Registration unsuccessful	"Passwords do not match" error message	Test successful

6	Organization registration with .org email Email: org@example.orgRole: organization	Registration successful	Prompt with "User Created Successfully" and Redirect to Verify Email page	Test successful
7	Organization registration without .org email Email: org@example.comRole: organization	Registration unsuccessful	"Organizations must register with a .org email address" error message	Test successful
8	Volunteer registration without interests Role: volunteer Interests: (empty)	Registration unsuccessful	"Please select at least one area of interest" error message	Test successful
9	Admin role registration attempt Role: admin	Registration unsuccessful	"Cannot create admin accounts via registration" error message	Test successful

Table 5.3: User Login Test Cases

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	Valid credentials Email: ramsha@gmail.com Password: ramsharma	Login successful	Redirect to home or verify email page	Test successful
2	Invalid email	Login unsuccessful	“User not registered” error message	Test successful

3	Invalid password	Login unsuccessful	"Invalid credentials" error message	Test successful
4	Empty email field	Login unsuccessful	Focus on email field	Test successful
5	Empty password field	Login unsuccessful	Focus on password field	Test successful
6	Login with unverified email	Login successful (limited)	Redirect to verify email page	Test successful
7	Login with verified email	Login successful	Redirect to home page	Test successful
8	Admin login	Login successful	Redirect to Admin Dashboard	Test successful
9	Organization login	Login successful	Redirect to My Opportunities	Test successful

Table 5.4: Logout Test Cases

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	Valid token logout	Logout successful	"Logged out successfully", token removed	Test successful
2	Logout without token	Logout unsuccessful	"No token provided"	Test successful
3	Invalid token	Logout unsuccessful	"Invalid token or user session"	Test successful

4	Expired token	Logout unsuccessful	“Invalid token or user session”	Test successful

2. User Management Module

Table 5.5: Profile Update Test Cases

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	Update all profile fields	Profile updated successfully	All data saved and displayed	Test successful
2	Update phone only	Phone updated successfully	Updated and displayed	Test successful
3	Update skills only	Skills updated successfully	Saved as array	Test successful
4	Update bio only	Bio updated successfully	Displayed on profile	Test successful
5	Update availability	Availability updated	Shown correctly	Test successful
6	Without authentication	Update unsuccessful	“Not authorized, no token”	Test successful
7	Invalid token	Update unsuccessful	“Token failed”	Test successful

Table 5.6: My Application Test Cases

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	Volunteer view	Applications displayed	List of applied opportunities	Test successful

2	Organization view	Access denied	"Only volunteers can view this"	Test successful
3	Without authentication	Access denied	"No token"	Test successful
4	No applications	Empty list	"No applications yet"	Test successful

3. Email Notification System

Table 5.7: Email Notification System Test Cases

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	Send verification email	Email sent	6-digit code sent	Test successful
2	Application notification	Email sent	Org notified	Test successful
3	Approval email	Email sent	Volunteer notified	Test successful
4	Attendance confirmation	Email sent	Volunteer notified	Test successful
5	Invalid SMTP credentials	Failed	Error logged	Test successful
6	Invalid address	Failed	Error logged	Test successful

4. Recommendation Engine

Table 5.8: Recommendation Engine Test Cases

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	Volunteer with interests	Recommendations shown	Sorted by cosine similarity	Test successful
2	No interests	All shown	Low scores	Test successful

3	Matching tags	High score	Prioritized matches	Test successful
4	No matching tags	Low score	Minimal matches	Test successful
5	No authentication	Access denied	“No token”	Test successful
6	Empty opportunity list	Empty result	[] returned	Test successful

5. API & Routing Structure

Table 5.9: API & Routing Structure Test Cases

S.N.	Test Inputs	Expected Output	Actual Output	Remarks
1	GET /api/opportunities	List returned	All opportunities JSON	Test successful
2	POST /api/auth/register	User created	Token + 201 status	Test successful
3	POST /api/auth/login	User logged in	Token + user data	Test successful
4	GET /api/auth/me	User data	Returned correctly	Test successful
5	PUT /api/user/profile	Updated	200 status	Test successful
6	POST /api/opportunities/:id/apply	Application submitted	Success 200	Test successful

7	Invalid route	404	“Not found”	Test successful
8	Wrong method	405	“Method not allowed”	Test successful
9	Admin route (non-admin)	Access denied	“Admin required”	Test successful

5.2.2 System Testing

System testing was performed to ensure that all integrated modules of the application function correctly as a complete system.

- Verified complete volunteer and organization workflows from registration to post-verification actions
- Checked proper functioning of email verification and notification system
- Ensured role-based access controls are strictly enforced
- Validated opportunity creation, application, and approval processes
- Confirmed smooth data flow between frontend, backend, and database
- Tested system response to errors and unexpected inputs
- Ensured compatibility across different browsers and devices
- Verified secure handling of sensitive user data (passwords, tokens, etc.)

Table 5.10: Test Cases for System Testing

Test Case ID	Description	Input	Expected Output	Result
S-TC001	Register a new volunteer	Name, Email, Password, Role=Volunteer	User registered, verification email sent	Pass
S-TC002	Register organization	Name, Email=xyz@abc.org,	User registered, verification email sent	Pass

	with valid .org email	Password, Role=Organization		
S-TC003	Register organization with invalid email	Name, Email=abc@gmail.com, Password, Role=Organization	Error: Only .org email allowed for organization	Pass
S-TC004	Login with valid credentials	Email and Password	JWT token returned, user details displayed	Pass
S-TC005	Login with invalid password	Email and incorrect Password	Error: Invalid credentials	Pass
S-TC006	Apply for opportunity (verified user only)	Authenticated verified volunteer	Application submitted successfully	Pass
S-TC007	Apply for opportunity (unverified user)	Authenticated unverified volunteer	Error: Email not verified	Pass
S-TC008	Post opportunity (verified organization)	Authenticated verified organization, opportunity details	Opportunity created successfully	Pass
S-TC009	Email sent on selection	Volunteer selected for opportunity	Email with opportunity details sent to volunteer	Pass
S-TC010	Verify email using token	Valid email verification token	Email verified successfully	Pass
S-TC011	Access protected route without token	No token in request headers	Error: Not authorized, no token	Pass

5.3 Result Analysis

The developed volunteer management system was tested thoroughly, and the results demonstrate its functional integrity and correctness.

- All functional test cases passed as expected.
- Cosine similarity algorithm accurately matched volunteers to opportunities based on selected tags.
- Email verification ensured only verified users could access restricted features such as posting or applying for opportunities.
- The system performed efficiently under normal load conditions, maintaining fast response times for data retrieval and recommendation generation.
- User interface testing confirmed that the platform was intuitive and easy to navigate for both volunteers and organizations.

CHAPTER 6: CONCLUSION & RECOMMENDATIONS

6.1 Conclusion

This project aimed to design and develop a Volunteer Management System to bridge the gap between organizations and volunteers by providing a seamless platform for opportunity sharing, application, and tracking. The key features include:

- User registration and role-based access control (Volunteer / Organization / Admin).
- Email verification system for secure registration.
- Cosine similarity-based recommendation engine for personalized volunteer opportunity suggestions.
- Application tracking and volunteer selection workflow.
- Notification system using email for application status updates.
- Admin control for approving organizational users and ensuring system integrity.
- Proper validation and system testing to ensure functionality and security.

6.2 Future Recommendations

To further improve and expand the capabilities of this system, the following recommendations can be considered:

- Implement an in-app notification system alongside emails for better user communication.
- Incorporate AI/ML techniques to improve recommendation accuracy and personalization.
- Integrating user feedback to the system.
- Introduce user analytics for organizations to evaluate the reach and performance of their volunteer programs.
- Enable calendar integration for volunteers to sync accepted opportunities with personal schedules.
- Hybrid recommendation approaches to combine content-based and collaborative advantages.
- Integrate advanced search and filtering features to allow users to easily explore and refine opportunities based on location and cause area.

REFERENCES

[1] Hager, M. A. & Brudney, J. L. (2004)‘(PDF) Volunteer Management Practices and Retention of Volunteers’, ResearchGate. [Online]. Available: https://www.researchgate.net/publication/237421554_Volunteer_Management_Practices_and_Retention_of_Volunteers. [Accessed: Oct. 28, 2025]

[2] VolunteerMatch, "VolunteerMatch," [Online]. Available: https://www.volunteermatch.org [Accessed: 28 Oct. 2025.]

[3] L. Hustinx and F. Lammertyn, ‘Collective and Reflexive Styles of Volunteering: A Sociological Modernization Perspective’, *Volunt. Int. J. Volunt. Nonprofit Organ.*, vol. 14, no. 2, pp. 167–187, June 2003, doi: 10.1023/A:1023948027200.

[4] C. C. Aggarwal, *Recommender Systems*. Cham: Springer International Publishing, 2016.

[5] Ricci, F., Rokach, L., & Shapira, B. (2015) ‘(PDF) Recommender Systems Handbook’, in *ResearchGate*. doi: 10.1007/978-0-387-85820-3_1.

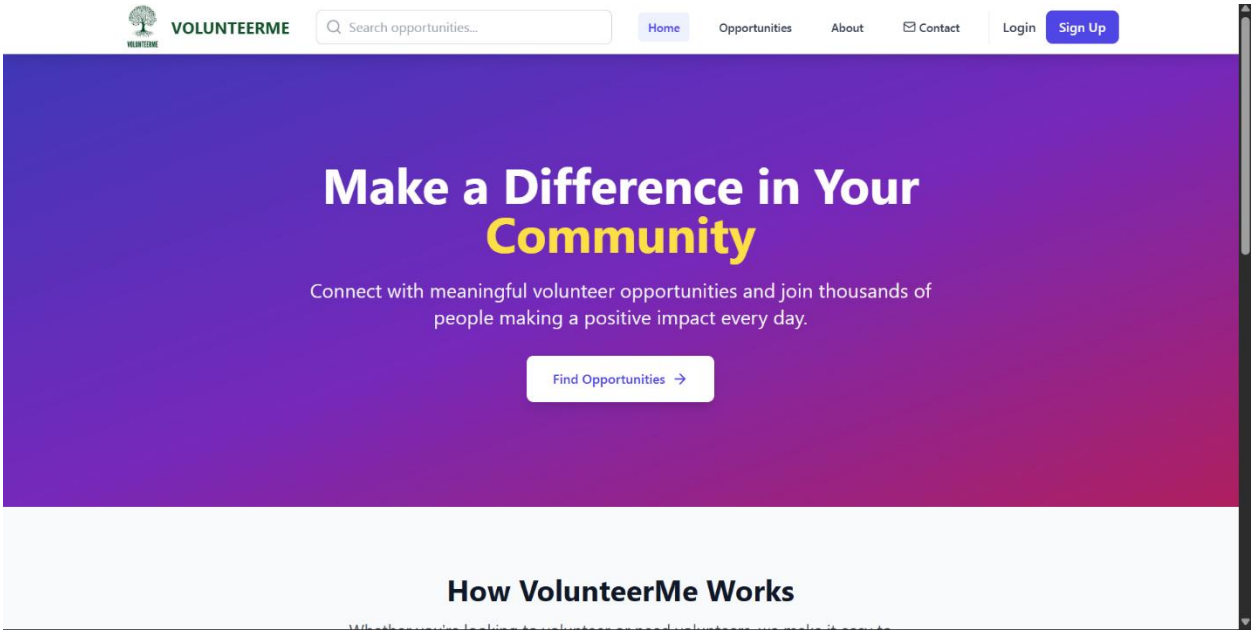
[6] Y. Koren, R. Bell, and C. Volinsky, ‘Matrix Factorization Techniques for Recommender Systems’, *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009, doi: 10.1109/MC.2009.263.

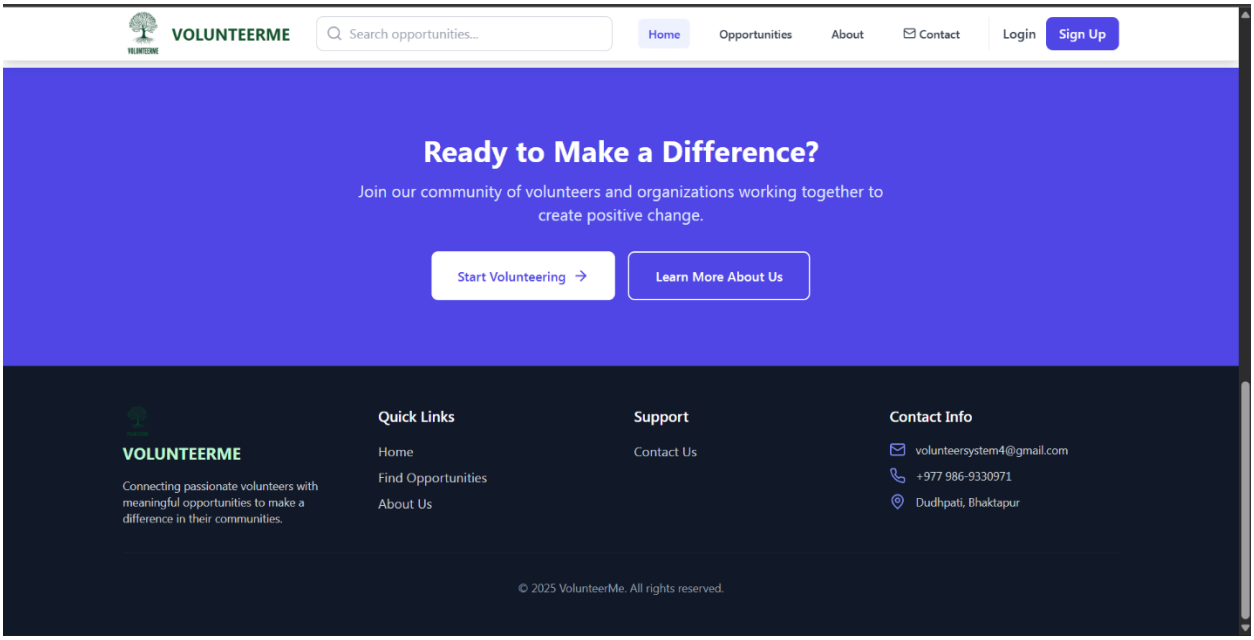
[7] Idealist ‘About Idealist’. [Online]. Available: https://www.idealist.org/en/about [Accessed: Oct. 28, 2025.]

[8] Adomavicius, G. & Tuzhilin, A. (2005) ‘Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions | IEEE Journals & Magazine | IEEE Xplore’. [Online]. Available: https://ieeexplore.ieee.org/document/1423975 [Accessed: Oct. 28, 2025.]

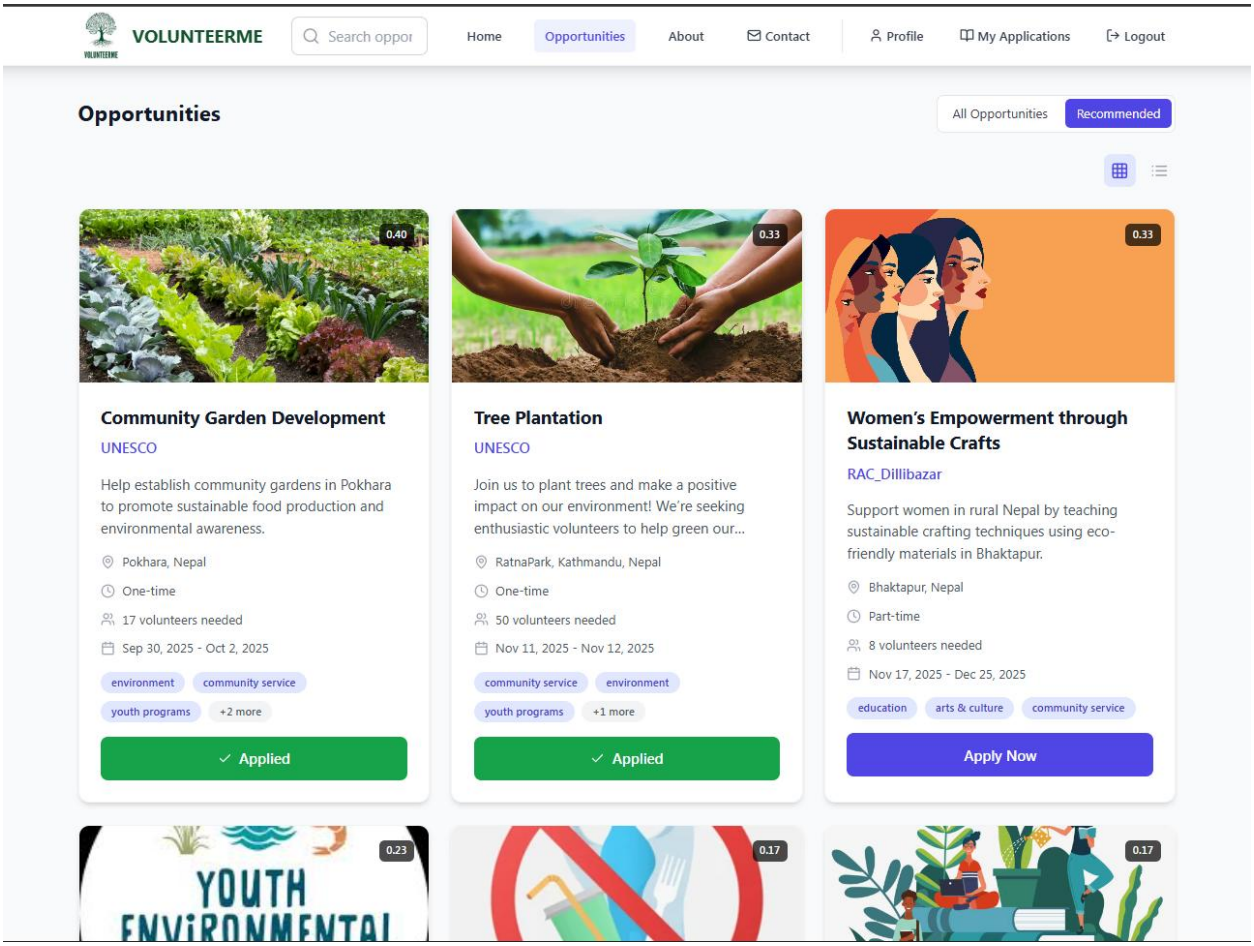
APPENDIX

Main Page:

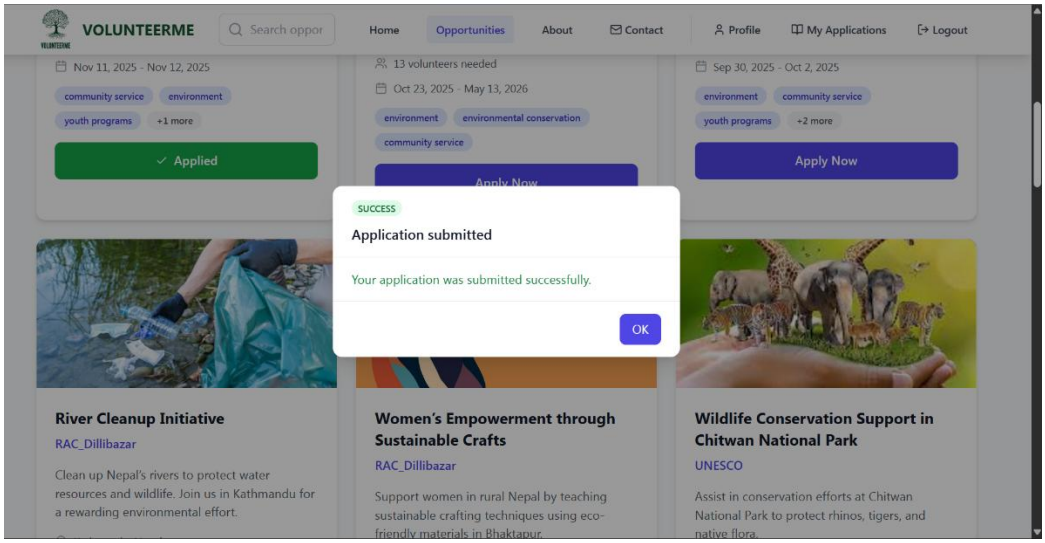


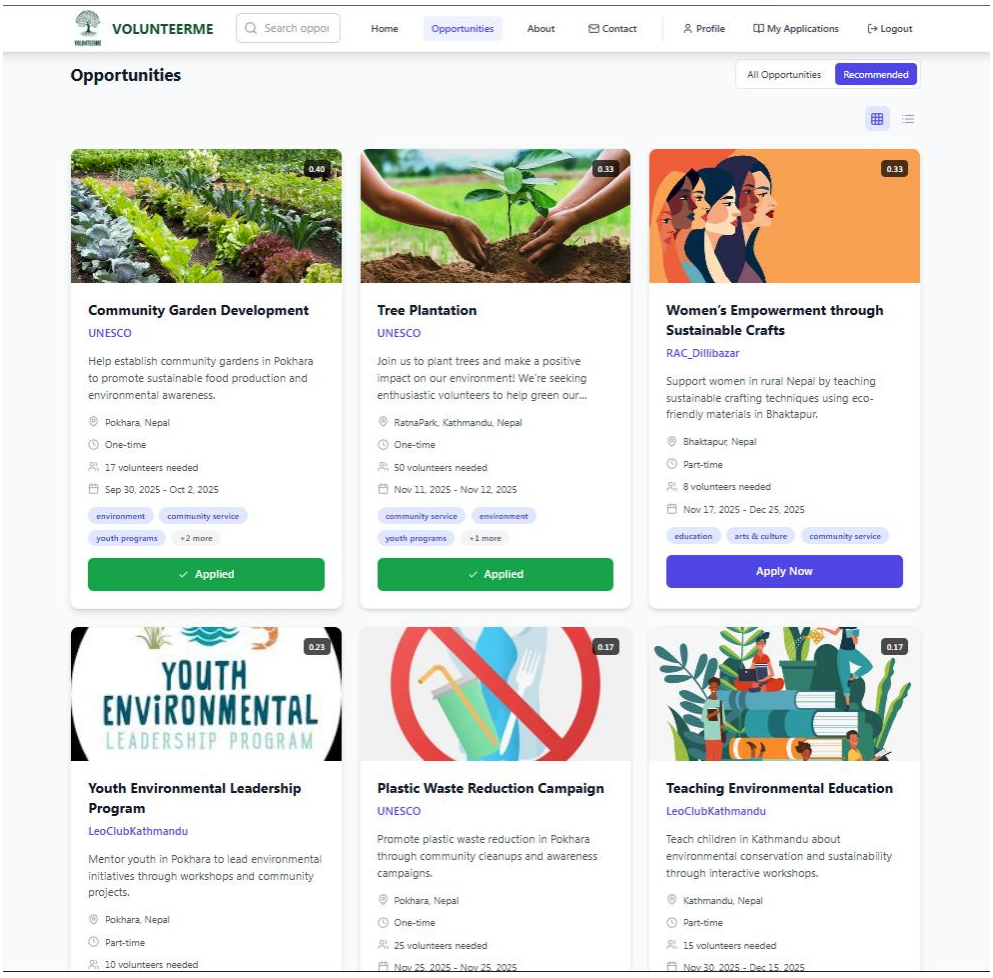


Recommended Opportunities:

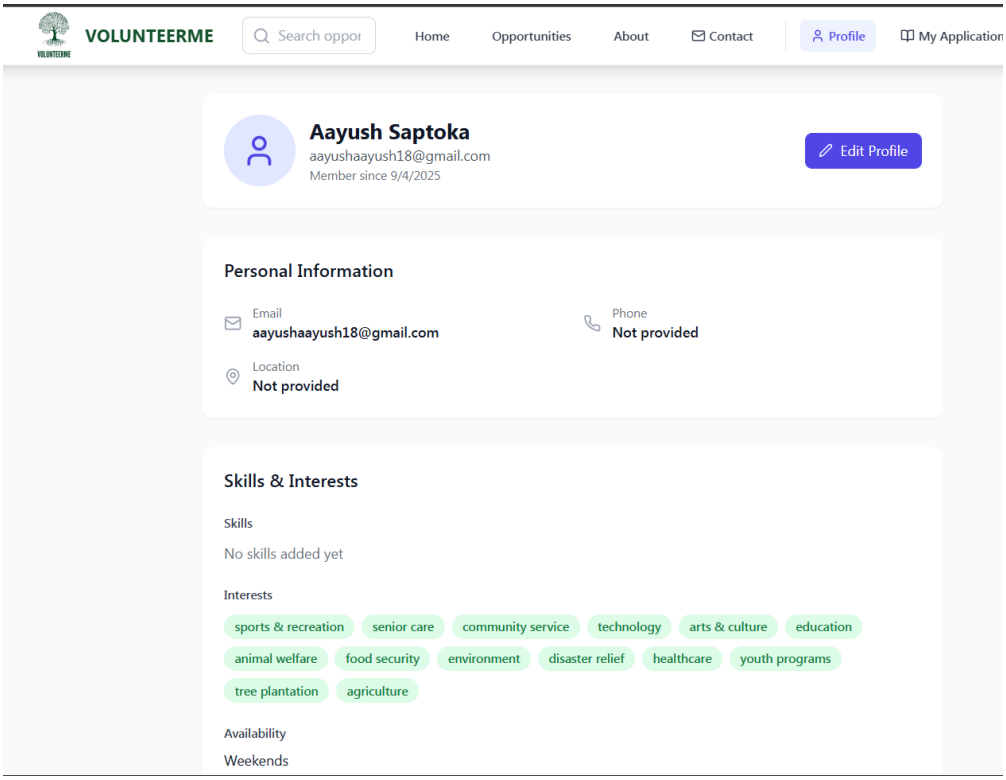


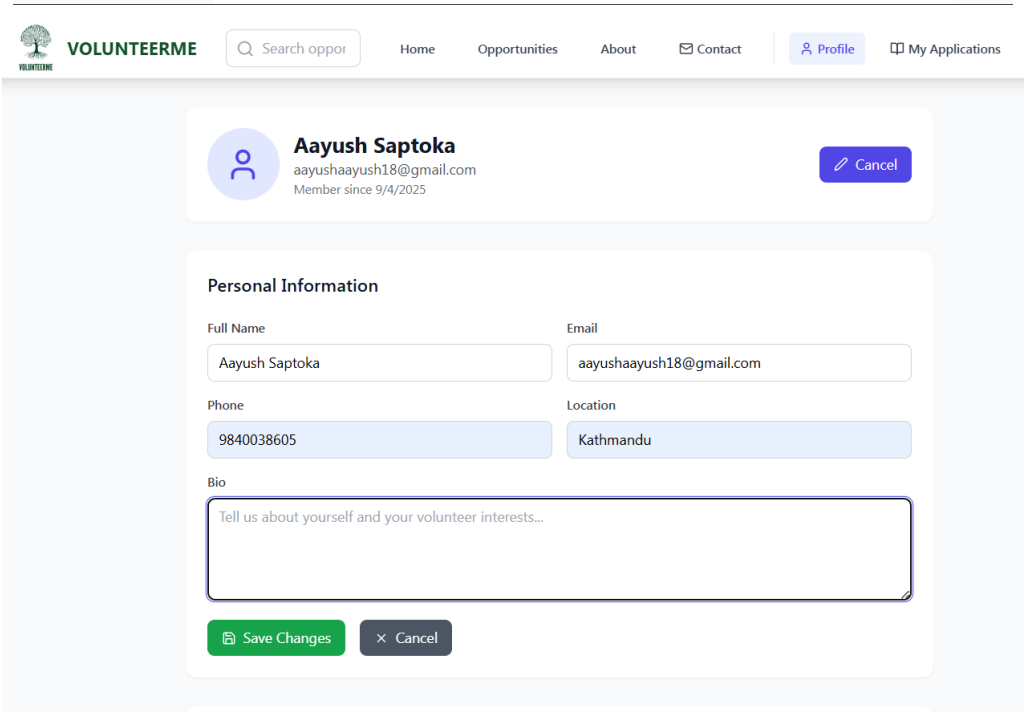
Applying for Opportunity:

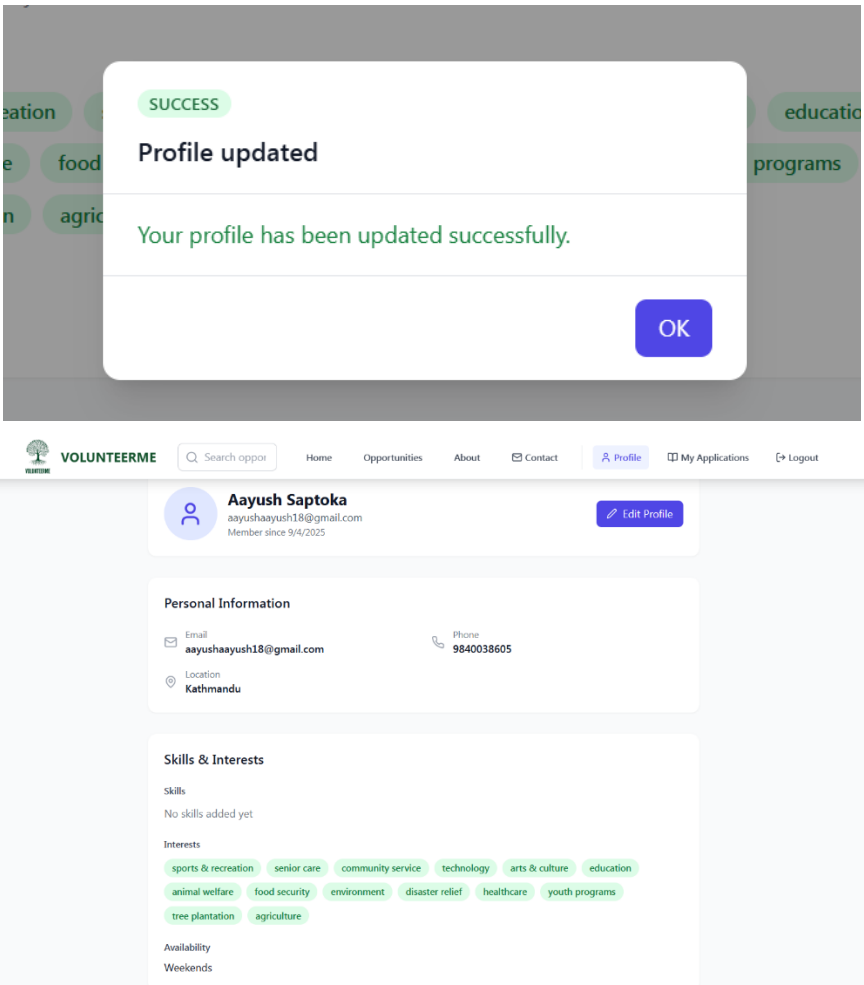




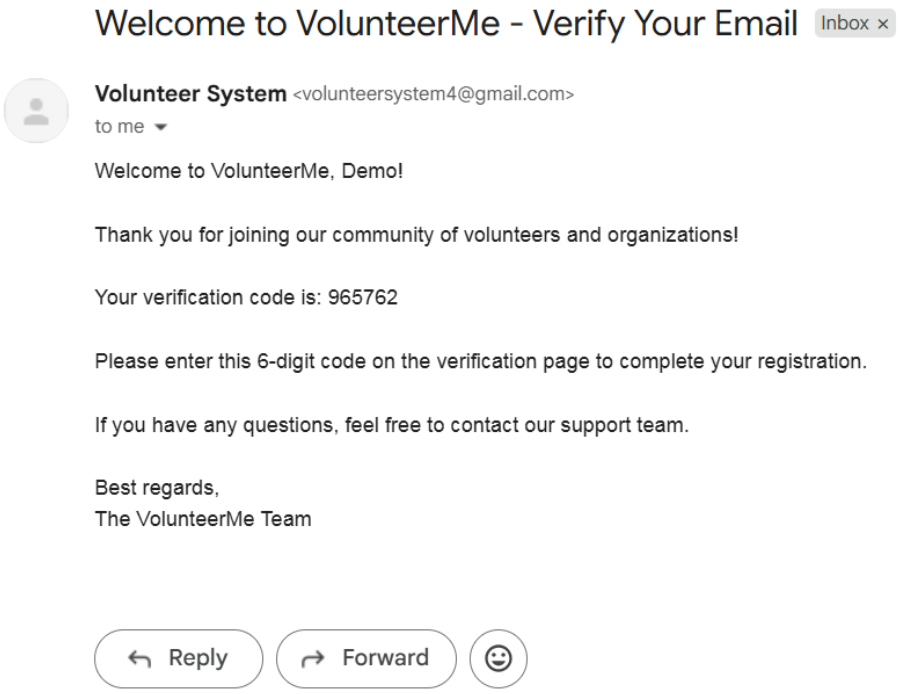
User Profile:

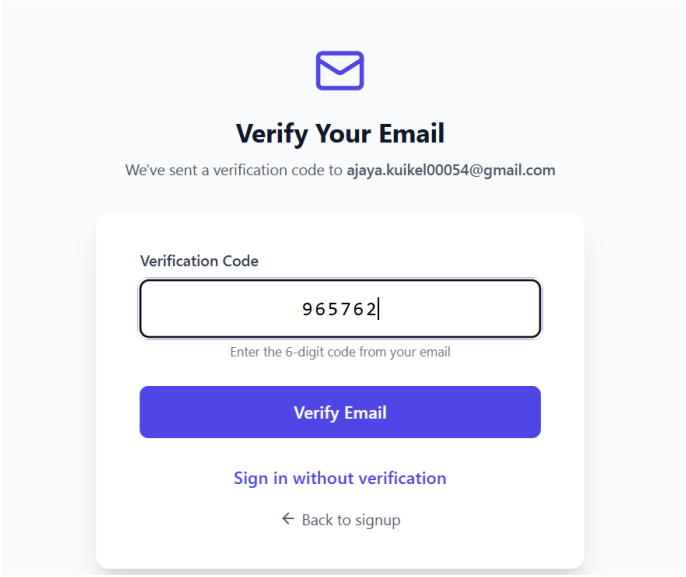


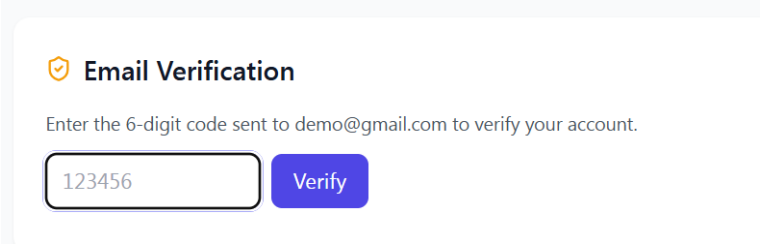



SUCCESS
Profile updated
Your profile has been updated successfully.
OK
VOLUNTEERME
Search oppor
Home
Opportunities
About
Contact
Profile
My Applications
Logout
Aayush Saptoka
aayushaayush18@gmail.com
Member since 9/4/2025
Edit Profile
Personal Information
Email
aayushaayush18@gmail.com
Phone
9840038605
Location
Kathmandu
Skills & Interests
Skills
No skills added yet
Interests
sports & recreation
senior care
community service
technology
arts & culture
education
animal welfare
food security
environment
disaster relief
healthcare
youth programs
tree plantation
agriculture
Availability
Weekends

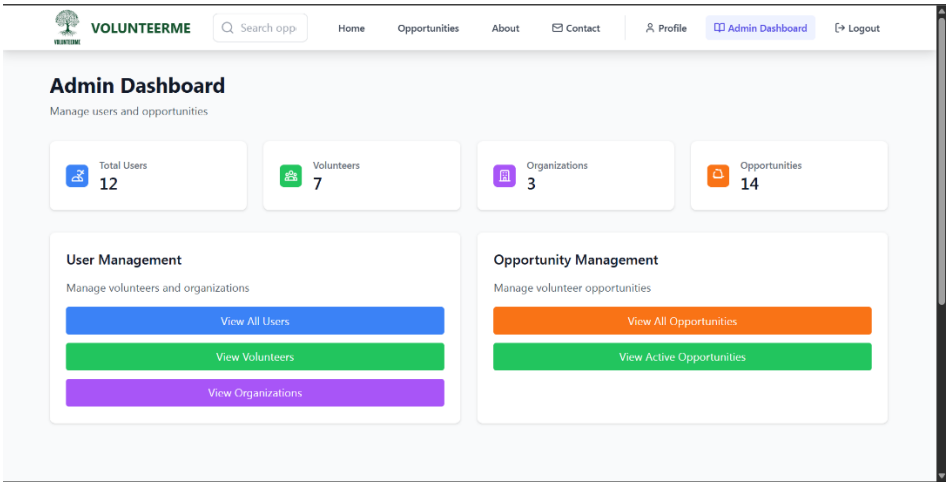
Email Verification:

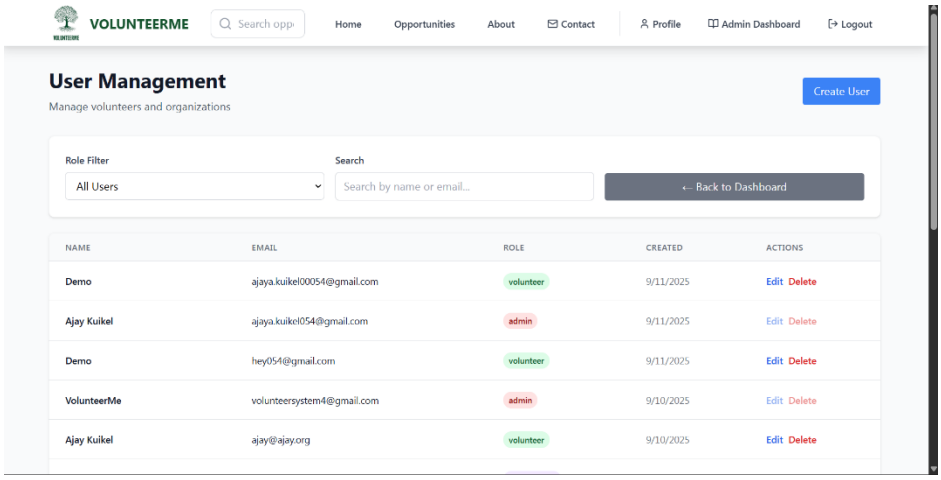


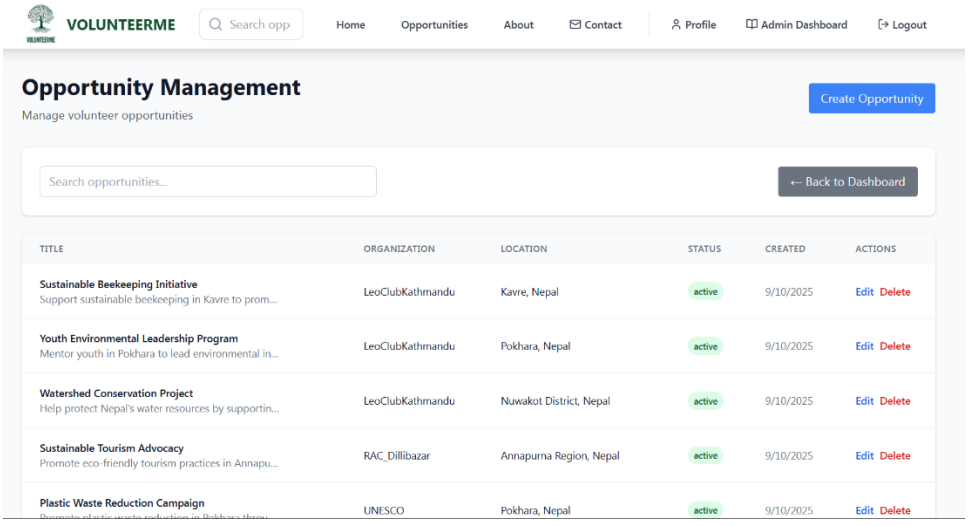




Admin Panel:


VOLUNTEERME
Search opp
Home
Opportunities
About
Contact
Profile
Admin Dashboard
Logout
Admin Dashboard
Manage users and opportunities
Total Users
12
Volunteers
7
Organizations
3
Opportunities
14
User Management
Manage volunteers and organizations
View All Users
View Volunteers
View Organizations
Opportunity Management
Manage volunteer opportunities
View All Opportunities
View Active Opportunities


VOLUNTEERME
Search opp
Home
Opportunities
About
Contact
Profile
Admin Dashboard
Logout
User Management
Manage volunteers and organizations
Create User
Role Filter
All Users
Search
Search by name or email...
← Back to Dashboard
NAME
EMAIL
ROLE
CREATED
ACTIONS
Demo
ajaya.kuikel00054@gmail.com
volunteer
9/11/2025
Edit Delete
Ajay Kuikel
ajaya.kuikel054@gmail.com
admin
9/11/2025
Edit Delete
Demo
hey054@gmail.com
volunteer
9/11/2025
Edit Delete
VolunteerMe
volunteersystem4@gmail.com
admin
9/10/2025
Edit Delete
Ajay Kuikel
ajay@ajay.org
volunteer
9/10/2025
Edit Delete

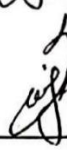
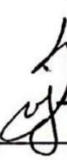
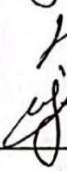

VOLUNTEERME
Search opp
Home
Opportunities
About
Contact
Profile
Admin Dashboard
Logout
Opportunity Management
Manage volunteer opportunities
Create Opportunity
Search opportunities...
← Back to Dashboard
TITLE
ORGANIZATION
LOCATION
STATUS
CREATED
ACTIONS
Sustainable Beekeeping Initiative
Support sustainable beekeeping in Kavre to prom...
LeoClubKathmandu
Kavre, Nepal
active
9/10/2025
Edit Delete
Youth Environmental Leadership Program
Mentor youth in Pokhara to lead environmental in...
LeoClubKathmandu
Pokhara, Nepal
active
9/10/2025
Edit Delete
Watershed Conservation Project
Help protect Nepal's water resources by supportin...
LeoClubKathmandu
Nuwakot District, Nepal
active
9/10/2025
Edit Delete
Sustainable Tourism Advocacy
Promote eco-friendly tourism practices in Annapu...
RAC_Dillibazar
Annapurna Region, Nepal
active
9/10/2025
Edit Delete
Plastic Waste Reduction Campaign
UNESCO
Pokhara, Nepal
active
9/10/2025
Edit Delete

Code Snippets:

```
const mongoose = require("mongoose");

function cosineSimilarity(vecA, vecB) {
  let dot = 0.0;
  let normA = 0.0;
  let normB = 0.0;

  for (let i = 0; i < vecA.length; i++) {
    dot += vecA[i] * vecB[i];
    normA += vecA[i] * vecA[i];
    normB += vecB[i] * vecB[i];
  }

  if (normA === 0 || normB === 0) return 0;
  return dot / (Math.sqrt(normA) * Math.sqrt(normB));
}

// Main Recommendation Function
async function recommendOpportunities(user, opportunities) {
  // Step 1: Build vocabulary (all unique tags from user + opportunities)
  const normalize = (t) => (t || '').toString().toLowerCase().trim();
  let allTags = new Set();
  if (Array.isArray(user?.interests)) {
    user.interests.forEach(tag => allTags.add(normalize(tag)));
  }
  opportunities.forEach(op => {
    if (Array.isArray(op?.tags)) {
      op.tags.forEach(tag => allTags.add(normalize(tag)));
    }
  });

  const vocab = Array.from(allTags);
  const N = opportunities.length + 1;

  // Step 2: Compute document frequency (df) for each tag
  let df = {};
  vocab.forEach(tag => { df[tag] = 0; });

  opportunities.forEach(op => {
    const uniqueTags = new Set((op.tags || []).map(normalize));
    uniqueTags.forEach(tag => { df[tag] = (df[tag] || 0) + 1; });
  });

  // Also count user profile as a "doc"
  if (Array.isArray(user?.interests)) {
    const uniqueUserTags = new Set(user.interests.map(normalize));
    uniqueUserTags.forEach(tag => { df[tag] = (df[tag] || 0) + 1; });
  }
```

```
49
50  // Step 3: Compute IDF for each tag
51  let idf = {};
52  vocab.forEach(tag => {
53    idf[tag] = Math.log(N / (1 + df[tag])); // smoothing with +1
54  });
55
56  // Step 4: Build TF-IDF vectors
57  function buildVector(tags) {
58    const counts = {};
59    const normTags = (tags || []).map(normalize);
60    normTags.forEach(tag => { counts[tag] = (counts[tag] || 0) + 1; });
61    const total = normTags.length || 1;
62
63    return vocab.map(tag => {
64      const tf = (counts[tag] || 0) / total;
65      return tf * idf[tag];
66    });
67  }
68
69  // User profile vector
70  const userVector = buildVector(user.interests || []);
71
72  // Step 5: Score each opportunity
73  const scored = opportunities.map(op => {
74    const opVector = buildVector(op.tags || []);
75    const score = cosineSimilarity(userVector, opVector);
76    const base = typeof op?.toObject === 'function' ? op.toObject() : op;
77    return { ...base, score };
78  });
79
80  // Step 6: Sort by similarity score
81  scored.sort((a, b) => b.score - a.score);
82
83  return scored;
84 }
85
86 // Backward-compatible alias (legacy name used in controller)
87 const recommendCosineBased = (user, opportunities) => recommendOpportunities(user, opportunities);
88
89 module.exports = { recommendOpportunities, recommendCosineBased };
90
```

Supervisor meetup Logbook

Date	Remarks	Signature
04/26	Asked for supervision on the project proposal.	
05/01	Discussed about the method & algorithm for the project.	
05/18	Asked for guidance for project report.	
06/26	Showcasing initial project demonstration.	
08/17	Showing finished project and asking for recommendations.	
08/24	Asking for final project approval.	
10/29	Submitting project report and asking for guidance for presentation	
10/31	Showing the presentation of the project.	

Supervisor's Name: Mr. Saroj Maharjan

Supervisor's Signature: