




Sonika Gupta

MiniProject_SMVDU_ALUMNI_NETWORK_WEBAPP 2025.docx

-  Research paper 2025
-  January 2025
-  Shri Mata Vaishno Devi University(SMVDU), Katra

Document Details

Submission ID

trn:oid:::1:3305797306

Submission Date

Jul 31, 2025, 3:53 PM GMT+5:30

Download Date

Jul 31, 2025, 3:56 PM GMT+5:30

File Name

MiniProject_SMVDU_ALUMNI_NETWORK_WEBAPP_Charanpreet_2025.docx

File Size

1.2 MB

40 Pages

4,792 Words

29,409 Characters





3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography

Match Groups

-  **12 Not Cited or Quoted 3%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 2%  Internet sources
- 0%  Publications
- 2%  Submitted works (Student Papers)

Match Groups

- 12 Not Cited or Quoted** 3%
Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations** 0%
Matches that are still very similar to source material
- 0 Missing Citation** 0%
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted** 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 2% Internet sources
- 0% Publications
- 2% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
www.slideshare.net		1%
2	Student papers	
University of Greenwich		<1%
3	Student papers	
University of Wales Institute, Cardiff		<1%
4	Student papers	
University of Cape Town		<1%

MINI PROJECT REPORT

ALUMNI MANAGEMENT PORTAL FOR SMVDU USING WEB DEVELOPMENT

by

Koripella Venkata Ramani

Entry. No.: 22BCS043

Sachin Anand

Entry. No.: 22BCS070

Charanpreet Kaur

Entry. No.: 22BCS109

Under the Guidance of

Dr. Sonika Gupta

ABSTRACT

This mini project presents the development of a centralized web-based **Alumni Management Portal** for Shri Mata Vaishno Devi University (SMVDU). Currently, alumni engagement is fragmented and largely dependent on informal channels like WhatsApp or emails. Our system bridges this gap by offering an interactive, scalable, and secure platform for alumni and university authorities to communicate effectively.

Developed using **Next.js**, the system integrates server-side routing, modular API endpoints, and dynamic rendering to support robust functionalities like user authentication, profile management, event announcements, and job postings. Role-based access control distinguishes workflows for alumni, administrators, and guests. Security features including Google OAuth2 and encrypted sessions ensure data privacy and account integrity.

The system demonstrates scalability and performance with a modular architecture and cloud-based deployment readiness, setting the foundation for future extensions like mobile apps and analytics dashboards.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	Error! Bookmark not defined.
DECLARATION	Error! Bookmark not defined.
ABSTRACT	ii
TABLE OF CONTENTS.....	iii
LIST OF FIGURES	vi
LIST OF TABLES	vii
ABBREVIATIONS	viii
CHAPTER 1: INTRODUCTION	1
1. INTRODUCTION TO PROJECT	1
2. PROBLEM STATEMENT AND PROJECT CATEGORY.....	1
3. OBJECTIVES	2
4. PROBLEM FORMULATION	2
1. Verification of alumni identity.....	2
2. Scalable data management.....	2
3. Content moderation and access control	2
4. Cross-device accessibility	3
5. Easy future extension	3
5. IDENTIFICATION/RECOGNITION OF NEED.....	3

6. EXISTING SYSTEM.....	3
7. PROPOSED SYSTEM.....	4
8. UNIQUE FEATURES OF THE SYSTEM.....	4
9. Report Outline.....	5

CHAPTER 2: REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION 6

2.1 FEASIBILITY STUDY.....	6
2.1.1 Technical Feasibility	6
2.1.2 Economic Feasibility.....	6
2.1.3 Operational Feasibility	6
2.2 Software Requirements Specification (SRS).....	7
2.2.1 Data Requirements	7
2.3 VALIDATION	8

Chapter 3: SYSTEM DESIGN..... 10

3.3.1 Key UI Screens.....	13
3.4.1 Key Collections (MongoDB):	16
3.4.2 Non-Mongo Data Source:	16
3.6 Security Design.....	17

Chapter 4: IMPLEMENTATION, TESTING, AND MAINTENANCE 18

4.1.2 Key Functional Modules	18
------------------------------------	----

Chapter 5: RESULTS AND DISCUSSIONS..... 22

5.1.1 Alumni Users Can:	22
5.1.2 Admin Users Can:	22
5.1.3 Key Screenshots (to be added in final document):.....	22
5.2 Description of Major Modules	24
5.2.1 Module	24
Description	24
5.3.1 Defined Schemas.....	24
5.4 Observations and Insights	25

5.5 Challenges Faced.....	25
Chapter 6: CONCLUSION AND FUTURE SCOPE.....	26
6.1 CONCLUSION.....	26
6.1.1 Key Learnings and Technical Exposure:.....	26
6.2 FUTURE SCOPE	26
6.2.6 Intelligent Recommendations.....	27
6.2 Final Remark	27
Chapter 7: References.....	28
Appendix I: Sample Schemas and Dataset	29
1. DataSet Used:.....	29
2. Schemas used:	29
Appendix II: Module Vulnerability Analysis	32
1. Vulnerability in.xlsx Module	32



LIST OF FIGURES

Figure 1: Use Case Diagram	11
Figure 2: Level 0 Data Flow Diagram	11
Figure 3 :Level 1 Data Flow Diagram.....	11
Figure 4: Entity Relationship Diagram	12
Figure 5: Landing Page	13
Figure 6: Alumni List View	13
Figure 7: Alumni Map View	14
Figure 8: Alumni user info section.....	14
Figure 9: Alumni Events Section	15
Figure 10: Campus visit form for Alumni.....	15
Figure 11: ER Diagram - Alumni Portal (Excel + MongoDB).....	17
Figure 12: Google OAuth2 Login Page	22
Figure 13: Events page.....	23
Figure 14: Profile Setup Page.....	23

LIST OF TABLES

Table 1: Expected Hurdles	9
Table 2: Tools and technologies.....	17
Table 3: Technology Stack.....	18
Table 4: Project Scheduling	19
Table 5: Testing Techniques used.....	20
Table 6: Test Plan Examples	20
Table 7: Description of major modules.....	24
Table 8: Defined Schemas.....	24
Table 9: Observation and insights.....	25
Table 10: Challenges Faced	25

ABBREVIATIONS

- SMVDU – “Shri Mata Vaishno Devi University”
- AWS – “Amazon Web Services”
- OOD – “Object-Oriented Design”
- DFD – “Data Flow Diagram”
- UML – “Unified Modelling Language”
- ER – “Entity Relation”
- 1NF – “First Normal Form”
- 2NF – “Second Normal Form”
- 3NF – “Third Normal Form”
- CRUD – “Create Read Update Delete”
- ODM – “Object Data Modelling”
- EJS – “Embedded JavaScript”
- IDE – “Integrated Development Environment”
- UI – “User Interface”

CHAPTER 1: INTRODUCTION

1. INTRODUCTION TO PROJECT

Alumni serve as the backbone of any academic institution's history and future development. They represent the university in the professional world, contribute to its growth, and often act as mentors, employers, or collaborators for current students. Despite the significance of alumni in the development of Shri Mata Vaishno Devi University (SMVDU), there exists no structured and official platform to engage with them systematically.

This project, titled **“Alumni Management Portal for SMVDU Using Web Development”**, addresses this gap by developing a secure, scalable, and centralized platform for alumni interaction. The system is a full-stack web application built using the **Next.js**. It includes role-based access, profile management, job posting, event announcements, and an admin panel.

2. PROBLEM STATEMENT AND PROJECT CATEGORY

Shri Mata Vaishno Devi University (SMVDU) has a growing and diverse alumni community spread across various industries and locations worldwide. However, there currently exists no dedicated, centralized platform that effectively connects alumni with each other and with the university. This lack of a comprehensive alumni management system results in limited engagement, communication gaps, and missed opportunities for both alumni and the institution.

Without a streamlined digital platform, the university faces several challenges:

- Difficulty in maintaining up-to-date and accurate alumni records.
- Inability to facilitate meaningful networking and professional connections among alumni.
- Limited channels for sharing important updates, events, and opportunities such as job openings, seminars, or reunions.
- Inefficiency in organizing alumni contributions, mentorship programs, and university outreach initiatives.
- Lack of a forum for alumni to collaborate, share experiences, and support current students.

Due to these issues, the potential benefits of a strong alumni network — such as career support, knowledge sharing, funding, and university branding — remain underutilized.

Hence, there is a pressing need to design and develop a **web-based Alumni Network System** for SMVDU. This system will serve as a centralized platform enabling alumni registration, profile management, event coordination, communication forums, and resource sharing. By fostering seamless interaction between alumni and the university, the system aims to strengthen community ties, facilitate professional networking, and support the continuous development of the institution and its members.

3. OBJECTIVES

1. To design and develop a responsive, user-friendly alumni web portal for SMVDU.
2. To enable SMVDU alumni to register, update their profiles, and interact with others.
3. To facilitate posting and viewing of job opportunities and event announcements.
4. To implement secure login and authentication using Google OAuth2.
5. To develop an admin panel for content moderation and user management.
6. To ensure data security, role-based access, and scalable backend architecture.

4. PROBLEM FORMULATION

The project is designed to address several interrelated technical and operational problems:

1. VERIFICATION OF ALUMNI IDENTITY

There must be a mechanism to ensure that only authentic SMVDU alumni can access the portal. The solution uses **Google OAuth2** with domain restriction (@smvdu.ac.in) to validate user identities.

2. SCALABLE DATA MANAGEMENT

Alumni information, job postings, and event details must be stored efficiently and accessed quickly. The system uses **MongoDB** to manage data with indexing for performance optimization.

3. CONTENT MODERATION AND ACCESS CONTROL

There needs to be a distinction between regular alumni users and administrative users to ensure appropriate content control. This is achieved using a **role-based access control system** implemented in the backend.

4. CROSS-DEVICE ACCESSIBILITY

Alumni should be able to use the portal on desktops, tablets, and smartphones. The user interface is made responsive using **Tailwind**.

5. EASY FUTURE EXTENSION

The portal should support additional features like analytics, chat, or mobile integration in the future. The use of **modular MVC architecture** and RESTful APIs makes it easy to expand.

5. IDENTIFICATION/RECOGNITION OF NEED

During interviews with students and early alumni, it was observed that alumni groups were scattered across WhatsApp, LinkedIn, and private email chains. University announcements often reached only a subset of the alumni. Furthermore, the university had no formal channel to tap into alumni resources such as job referrals, startup collaborations, or mentorship opportunities.

These gaps highlighted the **need for a centralized, secure, and scalable alumni management platform** that can be owned and operated by the university.

6. EXISTING SYSTEM

At present, SMVDU has no dedicated alumni web portal. Interactions are limited to:

1. Unofficial WhatsApp or Telegram groups
2. Occasional mass emails from departments
3. Social media groups on LinkedIn and Facebook

These approaches are:

1. **Unmoderated:** Anyone can join or post; there's no authentication or validation.
2. **Fragmented:** Alumni are divided across many platforms, making outreach inefficient.
3. **Not scalable:** Managing thousands of alumni across chat groups becomes impractical.
4. **Lack of roles:** There's no differentiation between admins, alumni, or guests.
5. **No formal tracking:** The university cannot analyse alumni engagement or contributions.

This results in missed opportunities for institutional growth, career development, and community building.

7. PROPOSED SYSTEM

The proposed **Alumni Management Portal for SMVDU** solves the above limitations through a centralized, web-based platform that is:

1. **Secure:** Only SMVDU alumni can register via Google OAuth2 restricted to university emails.
2. **Structured:** Alumni profiles are stored in a MongoDB database with full CRUD access.
3. **Interactive:** Users can post job openings, announce events, and share updates.
4. **Moderated:** Admin users can approve or delete posts, manage users, and oversee activity.
5. **Responsive:** The user interface adapts seamlessly to any screen size.
6. **Modular:** The backend is built with Node.js and Express.js using RESTful APIs, making it easy to integrate future features like analytics or mobile apps.

Key Components:

1. Alumni Registration with Google OAuth
2. Admin Dashboard
3. Alumni Directory
4. Event Posting Modules
5. Mobile-Friendly UI using Tailwind and Shadcn component library
6. MongoDB-based Backend with Indexed Collections

This system provides a formalized, professional solution for managing alumni relations at scale.

8. UNIQUE FEATURES OF THE SYSTEM

1. **Domain-restricted authentication** via Google OAuth (@smvdu.ac.in)
2. **Alumni profile management** with graduation year, profession, and profile picture
3. **Role-based access:** alumni, admin
4. **Event modules**

5. **Admin moderation tools**
6. **RESTful architecture** for scalable development
7. **Cloud deployment ready** Vercel
8. **Next.js** server-side & client-side rendering with optimizations

9. REPORT OUTLINE

This project report is divided into six major chapters, along with references and appendices:

Chapter 1: Introduction — Problem statement, motivation, system overview

Chapter 2: Requirement Analysis — Functional, non-functional, and technical requirements

Chapter 3: System Design — Architecture, UI design, database modelling

Chapter 4: Implementation and Testing — Backend/frontend development, testing strategies

Chapter 5: Results and Discussion — Screenshots, testing outcomes, analysis

Chapter 6: Conclusion and Future Scope — Final summary and recommendations

The report concludes with **references** and an **appendix** containing schemas and sample data.

CHAPTER 2: REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

This chapter establishes the foundation for designing and implementing an efficient and user-friendly web application.

2.1 FEASIBILITY STUDY

The feasibility study ensures that the proposed Alumni Network web application is practical and achievable within the given constraints. It evaluates the following dimensions:

2.1.1 TECHNICAL FEASIBILITY

The project is built using the **Next.js full-stack framework**, with the following technologies:

1. **MongoDB**: For flexible and scalable NoSQL data storage
2. **Next.js (React)**: For server-side rendering (SSR), API routing, and component-based UI
3. **Tailwind CSS**: For utility-first responsive UI design
4. **Preline UI**: For advanced dropdowns and layout components
5. **NextAuth.js with Google OAuth2**: For authentication and session management

Next.js's modular architecture supports hybrid rendering, API endpoints, and deployment across cloud platforms such as **Vercel**, ensuring technical scalability and long-term performance.

2.1.2 ECONOMIC FEASIBILITY

The project uses only open-source libraries and tools:

1. Node.js, Express.js, MongoDB Server
2. GitHub for version control
3. Vercel for free deployment

This makes the system **highly cost-effective**, especially for educational institutions with limited IT budgets.

2.1.3 OPERATIONAL FEASIBILITY

The system is designed for simplicity:

1. A clean, responsive interface using Tailwind and accessible design principles.

2. Seamless **Google login** via OAuth2 simplifies onboarding
3. An intuitive **admin dashboard** for job/event moderation and user tracking.

No prior technical knowledge is needed for end users or admins, improving operational feasibility.

2.2 SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

2.2.1 DATA REQUIREMENTS

1. **User Data:** Full name, email, entry number, department, profession
2. **Job Posts:** Title, company, description, contact info
3. **Events:** Title, description, date, image/media assets
4. **Sessions:** Login timestamps, role metadata, status
5. **Admin Controls:** Role-based user management and moderation history

2.2.2 FUNCTIONAL REQUIREMENTS

1. Secure login via **Google OAuth2**
2. User dashboard for profile creation and editing
3. Role-based job and event posting
4. Browsable alumni directory
5. Admin panel for data moderation and validation
6. Mobile-first responsive layout with collapsible menus and skeleton loaders

2.2.3 PERFORMANCE REQUIREMENTS

1. Page load time ≤ 3 seconds across devices
2. Optimized for ≥ 1000 concurrent users via serverless scaling
3. Indexed MongoDB queries and efficient API endpoints
4. Image optimization via Next.js static hosting and lazy loading

2.2.4 DEPENDABILITY REQUIREMENTS

1. Uptime target: $\geq 99.5\%$
2. Session fallback with secure cookies and JWT
3. Graceful 404 handling and fallback routing via `pages/404.tsx`
4. Secure database connections using TLS encryption

2.2.5 MAINTAINABILITY REQUIREMENTS

1. Component-based codebase using app/, components/, and lib/ folders
2. API routes under /app/api with modular middleware
3. Use of .env.local for configuration management
4. Git versioning and branch-based feature development

2.2.6 SECURITY REQUIREMENTS

1. Verified-only access using **OAuth2**
2. Enforced **HTTPS** during deployment
3. Middleware-based input validation and sanitization
4. Role checks via session tokens for protected routes

2.2.7 LOOK AND FEEL REQUIREMENTS

1. SMVDU-inspired color scheme and logo branding
2. Minimalist layout with **Tailwind CSS** and **Preline UI**
3. Accessible font sizes and contrast ratios
4. Responsive across desktops, tablets, and mobile browsers

2.3 VALIDATION

1. **Prototype Testing:** Early user feedback collected through mock interfaces
2. **Manual Testing:** Conducted using browser dev tools and route tracing
3. **Schema Validation:** Handled via custom middleware on Next.js API routes
4. **Session and Role Validation:** Managed using NextAuth tokens and callbacks

2.4 EXPECTED HURDLES

Table 1: Expected Hurdles

Challenge	Solution
Initial user adoption	Create onboarding guide, promote through university clubs
OAuth2 errors	Validate redirect URIs, test with mock accounts
MongoDB scalability	Use Atlas cluster with sharding (if scaled in future)
Mobile responsiveness	Use Bootstrap grid system and EJS templating



CHAPTER 3: SYSTEM DESIGN

3.1 DESIGN APPROACH

The system **is** designed using **modular components** supported by **Next.js**, embracing separation of concerns and reusability. Although Next.js doesn't strictly follow MVC, its architecture enables clear logical separation:

1. **Model Layer:** MongoDB schemas managed via Mongoose
2. **View Layer:** React components styled with Tailwind CSS
3. **Controller Layer:** API route handlers under `/app/api`

This approach ensures better scalability, predictable behavior, and clean integration with modern deployment workflows.

3.2 DETAILED SYSTEM DESIGN

3.2.1 USE CASE DIAGRAM

Actors:

1. Alumni
2. Admin

Use Cases:

1. Login/Register
2. Post/View Job Opportunities
3. Post/View Events
4. Admin Moderation
5. Profile Management

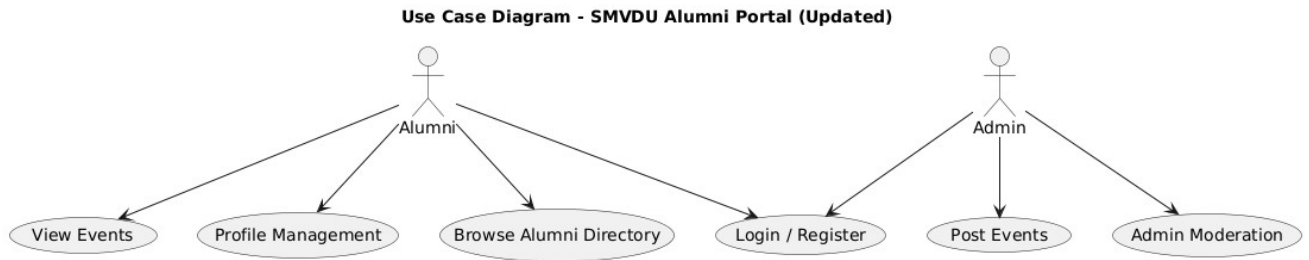


Figure 1: Use Case Diagram

3.2.2 DATA FLOW DIAGRAM (DFD)

1. Level 0 (Context Diagram):

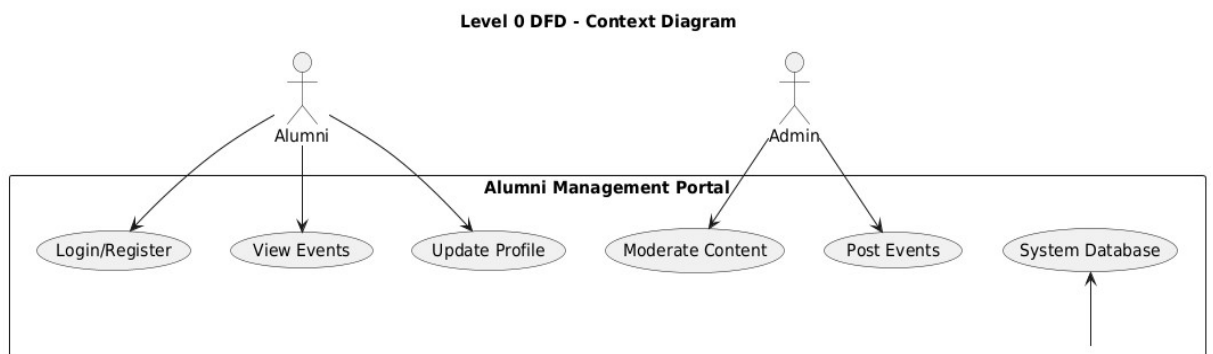


Figure 2: Level 0 Data Flow Diagram

2. Level 1:

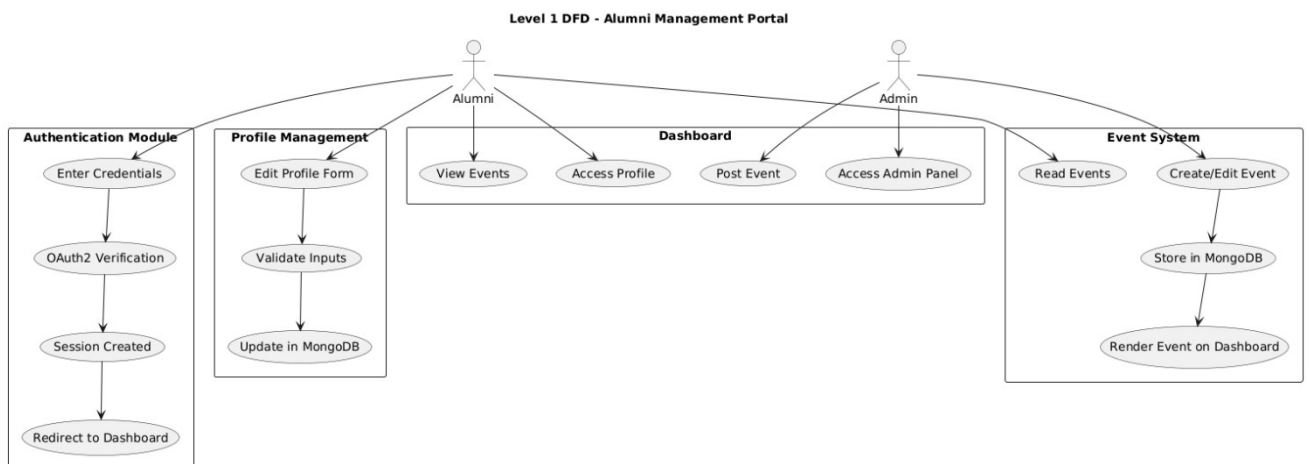


Figure 3 :Level 1 Data Flow Diagram

3.2.3 ENTITY-RELATIONSHIP (ER) DIAGRAM

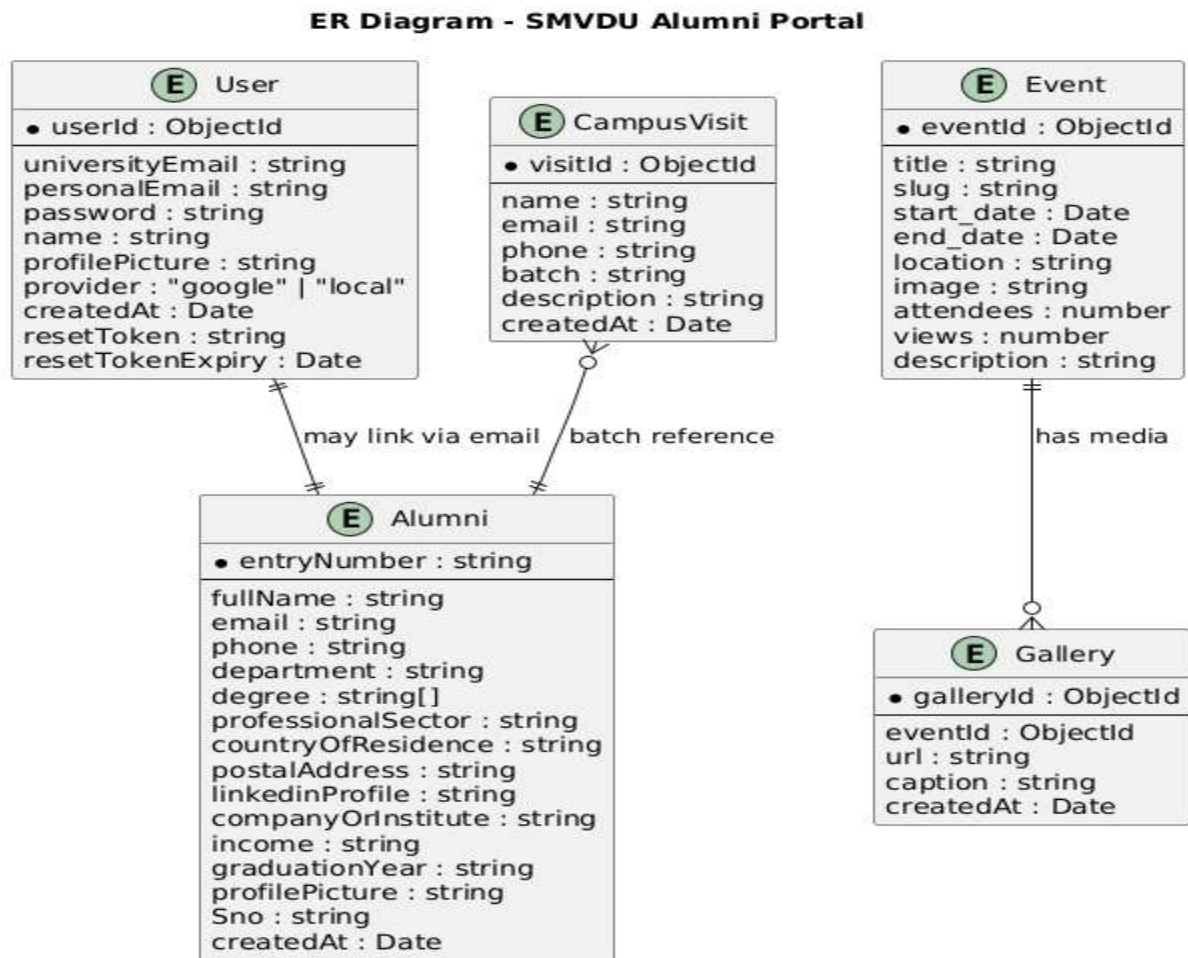


Figure 4: Entity Relationship Diagram

3.3 USER INTERFACE DESIGN

The UI has been designed using **Tailwind CSS** and **Preline UI**.

3.3.1 KEY UI SCREENS:

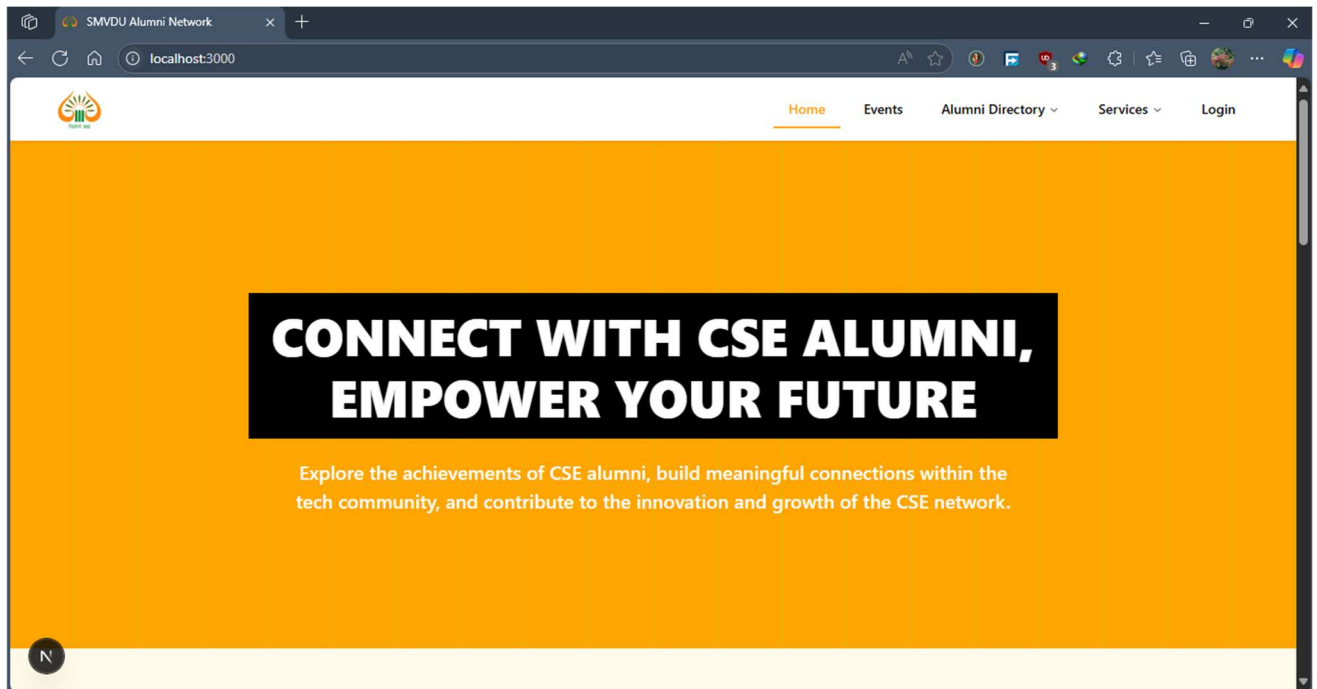


Figure 5: Landing Page

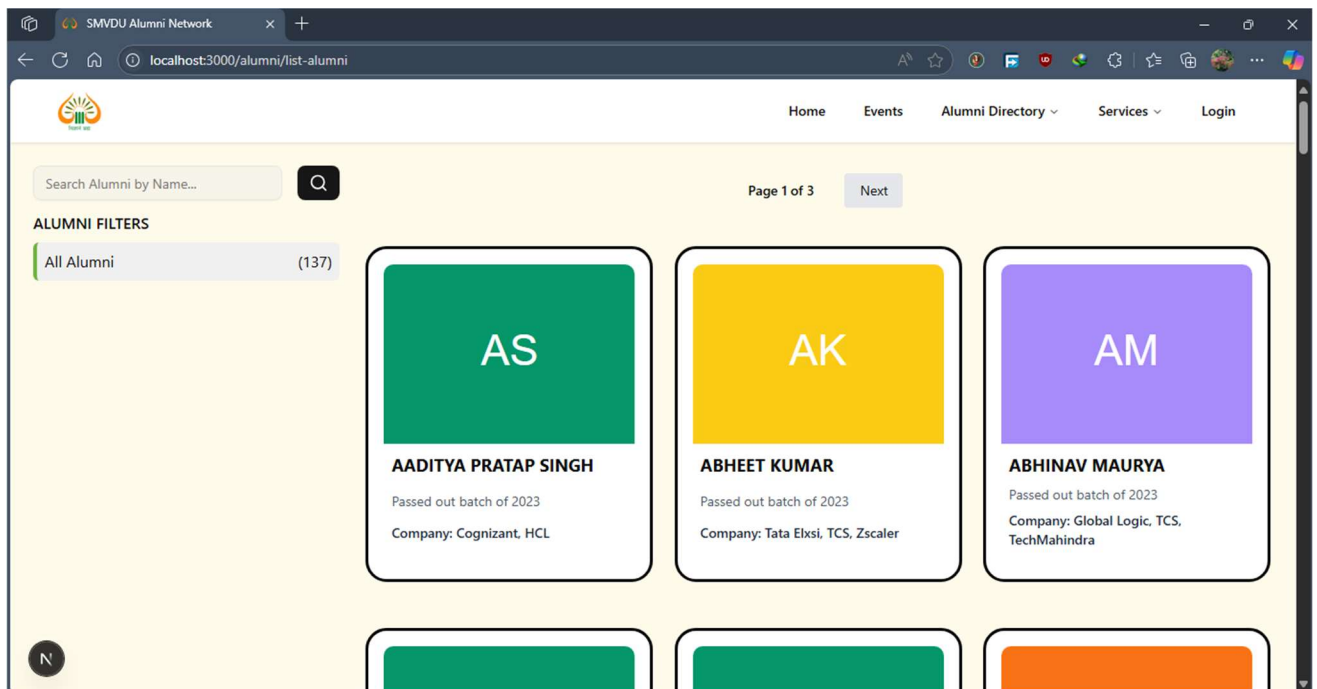


Figure 6: Alumni List View

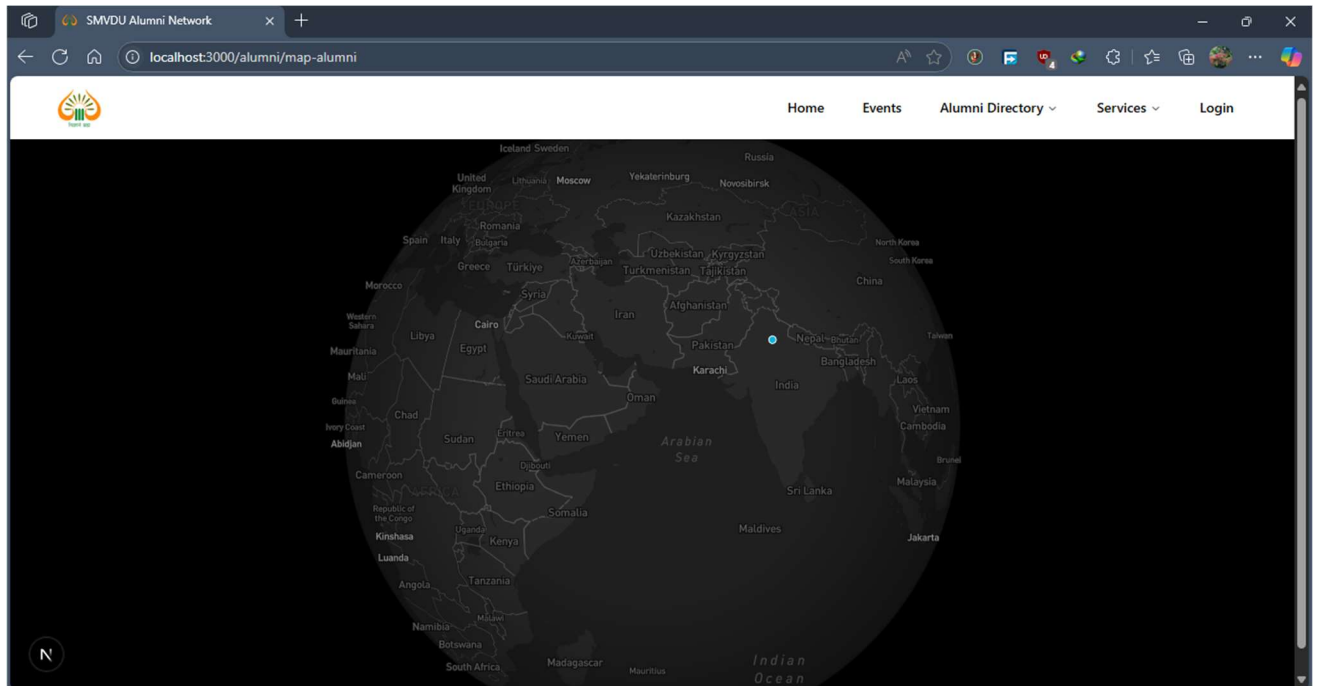


Figure 7: Alumni Map View

The screenshot shows a web browser window with the URL 'localhost:3000/alumni/alumni-info'. The page has a bright orange header with the text 'SMVDU (CSE) ALUMNI INFORMATION UPDATE FORM' in large, bold, white letters. Below the header, there is a section titled 'STAY CONNECTED WITH SMVDU'. The main content area is white and contains two input fields: 'FULL NAME (AS SMVDU RECORDS)' with the value 'Sachin Anand' and 'PERSONAL EMAIL ID' with the value 'sachinanand4703@gmail.com'. The browser's address bar and navigation icons are visible at the top.

Figure 8: Alumni user info section

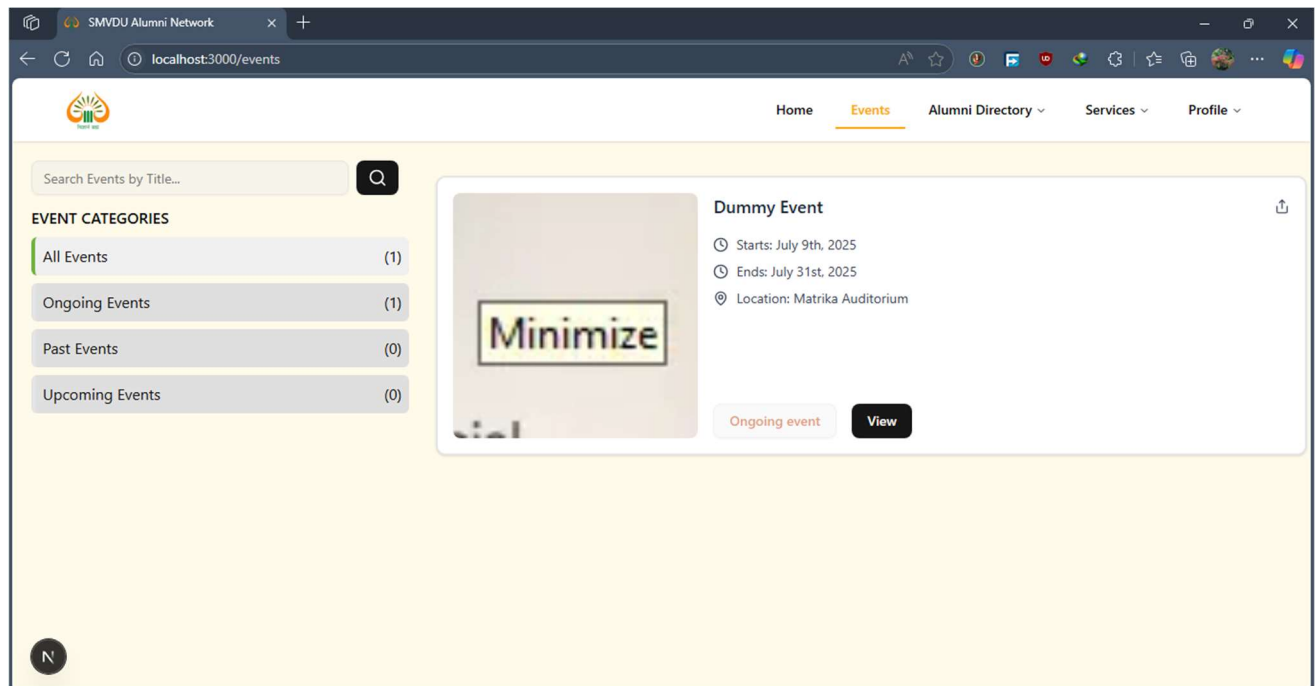


Figure 9: Alumni Events Section

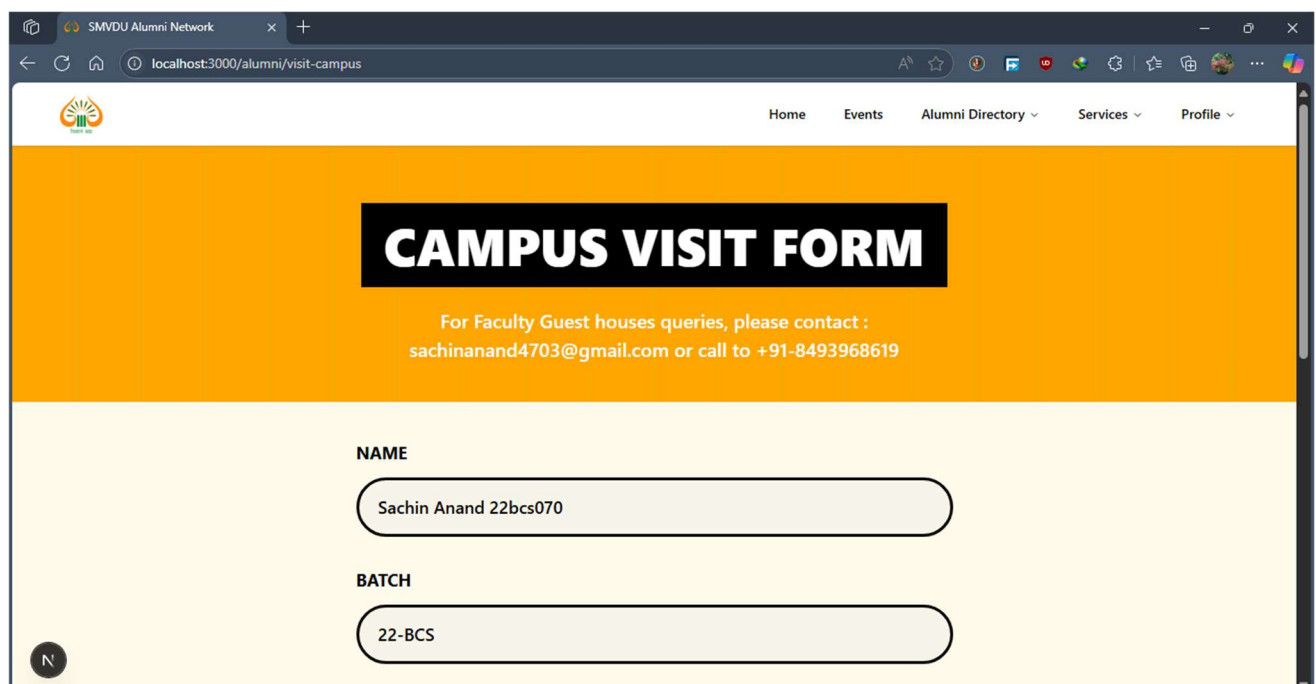


Figure 10: Campus visit form for Alumni

3.4 DATABASE DESIGN

While MongoDB was the intended backend, alumni data is managed via structured **Excel files** stored on the server with read/write logic using the `xlsx` library. Other dynamic collections — like events, campus visits, and gallery entries — are managed using **MongoDB**.

3.4.1 KEY COLLECTIONS (MONGODB):

1. users — contains both local and OAuth-authenticated user metadata
2. events — stores event details including time, location, and poster image
3. gallery — links event images to their respective events using ObjectId references
4. campusvisits — stores alumni campus visit requests

Each schema is defined using **Mongoose**, supports validation (required fields, types, enums), and includes default timestamps. Collections are indexed where applicable (e.g. event.slug) for efficient lookup and pagination.

3.4.2 NON-MONGO DATA SOURCE:

1. alumni.xlsx — Excel-based storage for alumni details (entry number, email, graduation year, department, etc.) using structured sheets by batch.

3.5 NORMALIZATION

Although MongoDB is non-relational, partial normalization is achieved through:

1. **Gallery ↔ Event:** Each gallery entry uses eventId reference to link media with specific events.
2. **User Role Separation:** Role-checking is performed via conditional fields (provider, role) inside User schema.
3. **Alumni Data:** As alumni entries are stored in Excel, they are not normalized in MongoDB but are indexed dynamically per batch/year sheet.

This hybrid architecture retains **NoSQL flexibility** while enforcing logical relationships where needed.

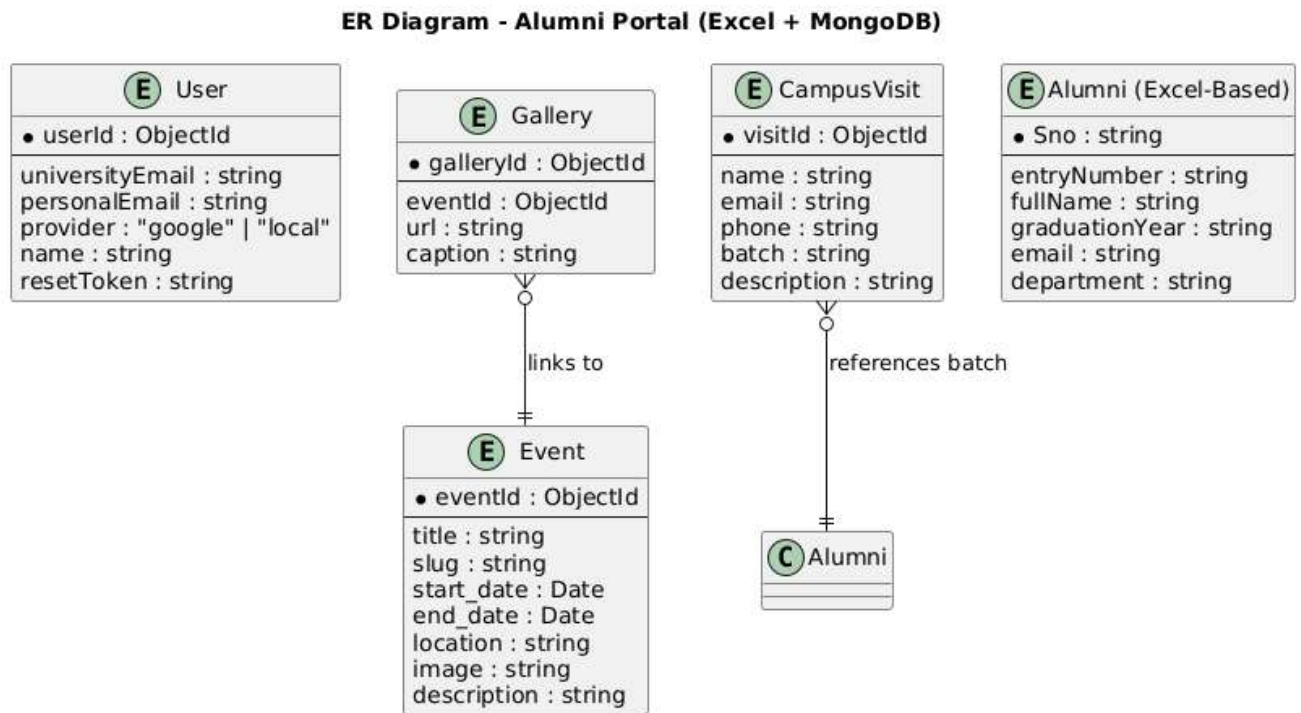


Figure 11: ER Diagram - Alumni Portal (Excel + MongoDB)

3.6 SECURITY DESIGN

1. **Google OAuth2:** Authenticates using verified SMVDU Gmail accounts.
2. **Role-based Authorization:** Access to job/event posting is controlled.
3. **Session Storage:** Managed via express-session and connect-mongo.

For Excel-based alumni storage, precautions have been taken to avoid deserialization vulnerabilities associated with the xlsx module by validating all data before reading or writing.

3.7 TOOLS AND TECHNOLOGIES

Table 2: Tools and technologies

Component	Technology
Frontend	React (Next.js), Tailwind CSS, Preline UI
Backend	Next.js API routes
Database	MongoDB
Auth	NextAuth.js with Google OAuth2
Versioning	Git + GitHub

Deployment	Vercel
------------	--------

CHAPTER 4: IMPLEMENTATION, TESTING, AND MAINTENANCE

4.1 IMPLEMENTATION

The Alumni Management Portal was implemented using the **Next.js** with a modular and component-driven architecture. The codebase is structured for clarity and maintainability, divided into:

1. **Pages & Routes:** Built under /app directory using Next.js's file-based routing
2. **Components:** Reusable UI elements with Tailwind CSS styling
3. **APIs:** RESTful endpoints organized under /app/api
4. **Models:** MongoDB schemas using Mongoose for validation and indexing
5. **Auth:** Secure session management with NextAuth.js and Google OAuth2

This structure allows for optimized performance, seamless navigation, and easy future scaling.

4.1.1 TECHNOLOGY STACK

Table 3: Technology Stack

Layer	Technology Used
Frontend	React (Next.js), Tailwind CSS, Preline UI
Backend	Next.js API routes
Database	MongoDB
Authentication	NextAuth.js + Google OAuth2
Hosting	Vercel
Version Control	Git + GitHub

4.1.2 KEY FUNCTIONAL MODULES

1. **Authentication Module:** Google OAuth2 via NextAuth.js with secure cookie-based sessions
2. **Dashboard Module:** Displays events, jobs, and user-specific content
3. **Admin Module:** Role-based controls for moderation and content management
4. **Post Management Module:** Create, read, update, and delete (CRUD) jobs and events

5. Alumni Directory Module: Dynamic filtering and listing of alumni data.

4.2 CODING STANDARDS

4.2.1 MODULARIZATION

Each major feature (auth, jobs, events, profile, admin) is separated into dedicated folders under /app, /components, and /lib.

4.2.2 NAMING CONVENTIONS

1. **JavaScript/TypeScript Variables & Functions:** camelCase
2. **Constants & Environment Vars:** UPPER_SNAKE_CASE
3. **Folder and File Names:** PascalCase and kebab-case
4. **Components:** PascalCase

4.2.3 ERROR HANDLING

1. Centralized error boundaries using Next.js server logs
2. Custom error classes for backend API responses
3. Logging via console.error() and Vercel build monitoring

4.3 Project Scheduling

Table 4: Project Scheduling

Task	Start Date	End Date
Requirement Analysis	01 Jan 2025	05 Jan 2025
Design (UI + DB + Flow)	06 Jan 2025	10 Jan 2025
Backend Development	11 Jan 2025	25 Jan 2025
Frontend Integration	26 Jan 2025	01 Feb 2025
Testing & Bug Fixing	02 Feb 2025	08 Feb 2025
Final Review and Deployment	09 Feb 2025	10 Feb 2025

4.4 TESTING

4.4.1 TESTING TECHNIQUES USED

Table 5: Testing Techniques used

Type	Description
Unit Testing	Components and API handlers tested with mock inputs using Jest
Integration Testing	Verified form-to-DB interactions and session behaviors
End-to-End Testing	Simulated workflows: login → post event → view dashboard
Security Testing	Role-based access checks, redirect validation, session expiry scenarios

4.4.2 TEST PLAN EXAMPLES

Table 6: Test Plan Examples

Test Case	Input	Expected Result	Status
Google Login	Valid SMVDU Email	Successful login + session	Pass
Invalid Login	Random Gmail	Access denied	Pass
Post Event	Title, Desc, Date	Visible on dashboard	Pass
Post Job	Job Title + Company	Stored in MongoDB	Pass
Unauthorized Access	Guest accessing /admin	Redirect to login or 403 error	Pass

4.5 MAINTENANCE

4.5.1 BUG FIXING AND UPDATES

1. Runtime logs monitored via Vercel Dashboard
2. Hotfixes and updates pushed via GitHub using semantic commits
3. Feature toggles used to gradually roll out new modules

4.5.2 BACKUP AND DATA RECOVERY

1. MongoDB Atlas enabled daily backups on free-tier cluster
2. Audit logs captured for sensitive actions (e.g. deletions, admin edits)
3. Admin tools in development for restore and rollback actions

4.5.3 USER FEEDBACK AND SUPPORT

1. Planned addition of feedback component in Admin Dashboard
2. Scheduled biannual iterations based on stakeholder reviews
3. GitHub issues used to track bugs and feature suggestions

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 USER INTERFACE REPRESENTATION

The system offers a clean, modern, and responsive interface tailored to both alumni and administrative users. The UI, built using **React components styled with Tailwind CSS and Preline UI**, is intuitive and mobile-friendly.

5.1.1 ALUMNI USERS CAN:

1. Authenticate using their **SMVDU Google email** via OAuth2
2. View job postings and alumni events on a personalized dashboard
3. Edit their profile including batch, department, company, and social links
4. Browse a dynamically filtered alumni directory

5.1.2 ADMIN USERS CAN:

1. Post and moderate events
2. Approve, reject, or delete entries to ensure integrity
3. Monitor system activity and gather usage feedback

5.1.3 KEY SCREENSHOTS (TO BE ADDED IN FINAL DOCUMENT):

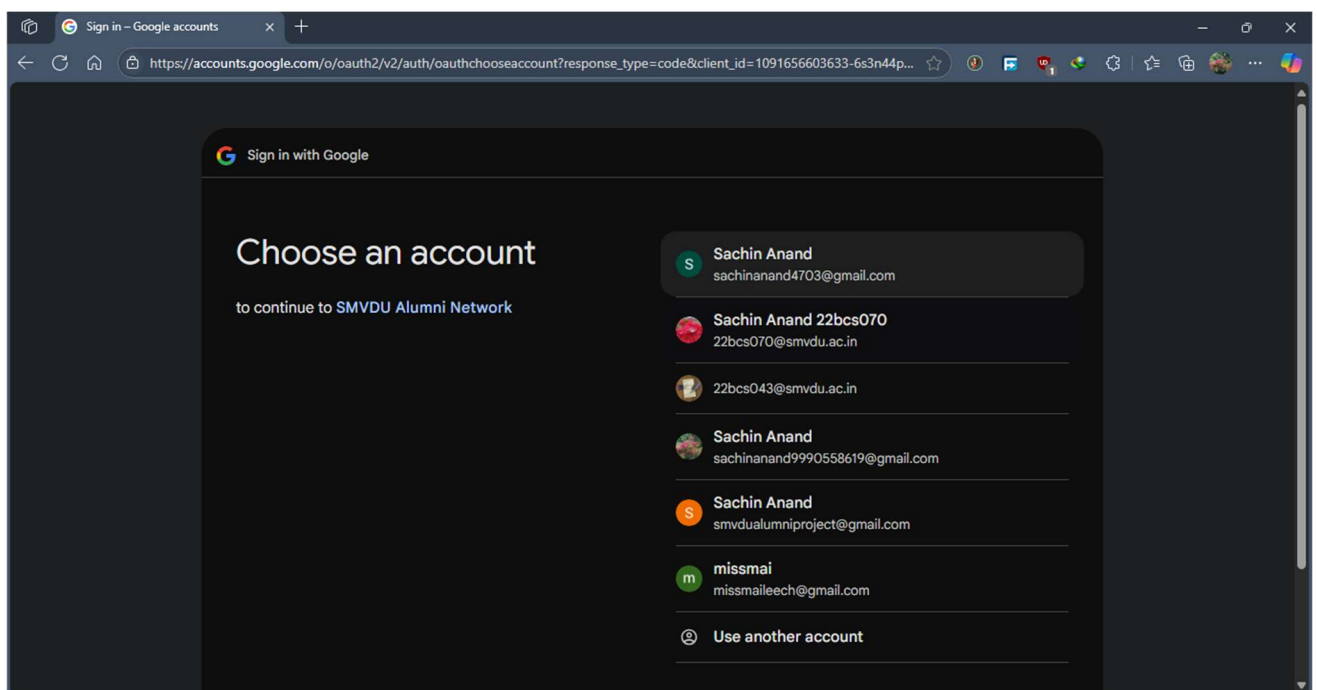


Figure 12: Google OAuth2 Login Page

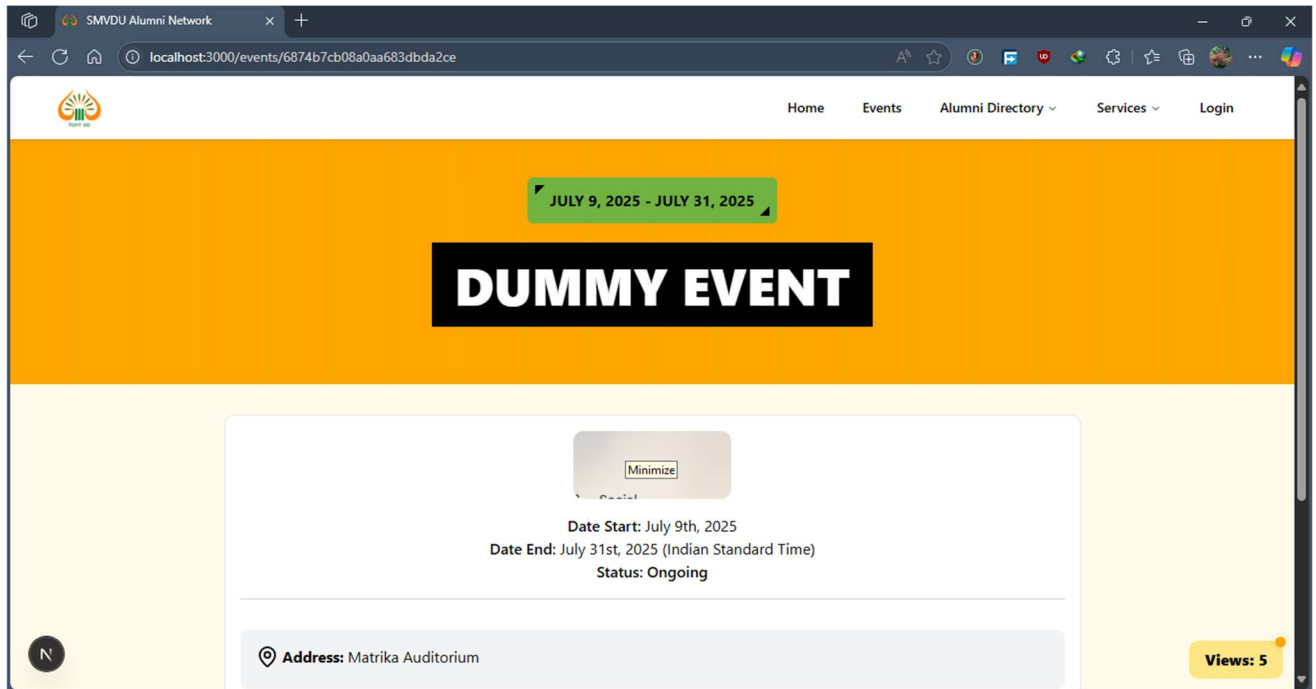


Figure 13: Events page

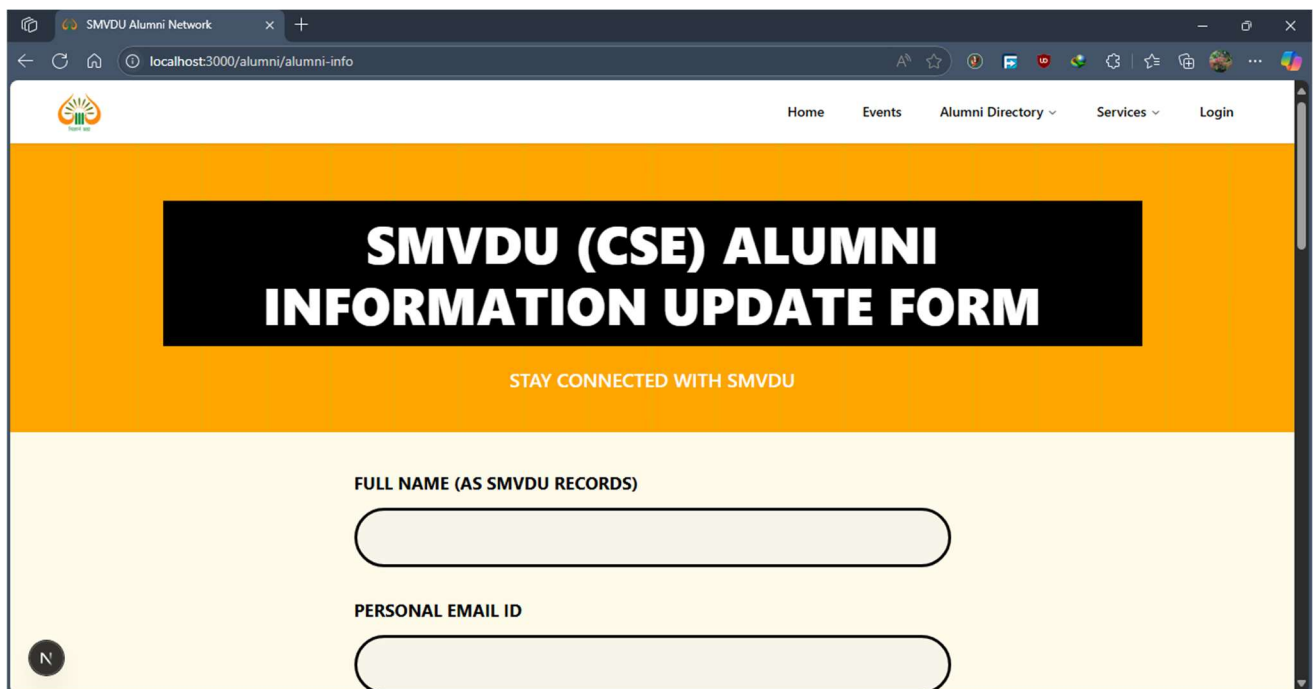


Figure 14: Profile Setup Page

5.2 DESCRIPTION OF MAJOR MODULES

Table 7: Description of major modules

5.2.1 MODULE	DESCRIPTION
Auth Module	Uses NextAuth.js + Google OAuth2 for secure authentication and session tokens
Profile Module	Allows alumni to update personal and professional details in a dynamic form
Jobs Module	Alumni can post job openings visible to authenticated users
Events Module	Admins post events/news tagged by domain and department
Admin Module	Role-based dashboard for validating, moderating, and tracking content

5.3 BACKEND REPRESENTATION

Backend logic is powered by both **Excel-based file storage** and **MongoDB Atlas**, reflecting a hybrid data persistence strategy optimized for portability and flexibility.

1. **Alumni Data:** Stored in structured Excel sheets using the **xlsx** library, organized by graduation year. This format allows lightweight data storage and manual edits while maintaining structured access via API endpoints.
2. **MongoDB Collections:** Other dynamic modules — including user sessions, events, campus visit requests, and gallery assets — are stored using **Mongoose schemas** located in `/models`.

5.3.1 DEFINED SCHEMAS

Table 8: Defined Schemas

Schema Name	Purpose
User	Stores authentication metadata, profile info, and provider details
Event	Captures event title, description, slug, location, images, attendance stats
Gallery	Holds event-linked media files with captions and timestamps
CampusVisit	Tracks alumni visit requests with batch, contact details, and descriptions

All schemas implement:

1. **Field-level validation** using Mongoose types and required flags
2. **Timestamping** for creation/update tracking

3. **Slug generation** in Event schema via pre('save') middleware
4. **Relational references** (e.g. eventId in Gallery) to link documents across collections

Alumni records in Excel follow a structured format with fields like entryNumber, email, department, companyOrInstitute, and graduationYear, validated at runtime before insertion or updates.

5.4 OBSERVATIONS AND INSIGHTS

Table 9: Observation and insights

Observation	Discussion
Login time under 3 seconds	Leveraged NextAuth.js session caching and minimal redirect latency
CRUD operations in under 2 clicks	Tailwind-powered UI improves form visibility and action clarity
Consistency across modules	Mongoose schema design ensures referential integrity across entries
Scalability	MongoDB handled mock loads effectively with indexed queries
Robust security and validation	Protected APIs via middleware, session checks, and input sanitization

5.5 CHALLENGES FACED

Table 10: Challenges Faced

Challenge	Resolution
Google OAuth redirect issues	Corrected authorized domains and callback URIs in Google Cloud Console
Schema validation inconsistencies	Refactored using middleware and native Mongoose constraints
Role bypassing attempts	Enforced session tokens and middleware-based route guards
Query performance bottlenecks	Introduced compound indexes and limited projections in MongoDB queries



CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The **Alumni Management Portal for SMVDU** successfully addresses a longstanding gap in structured alumni engagement and institutional outreach. By implementing a centralized, web-based solution, the system fosters direct communication between alumni and university authorities, streamlining access to job opportunities, events, and profile updates.

Developed using **Next.js**, the portal employs modern development practices including API routing, server-side rendering, and modular component architecture. Tailwind CSS ensures a clean and responsive UI, while authentication is securely managed via **Google OAuth2 integration through NextAuth.js**. Role-based access control enforces content moderation privileges and safeguards sensitive workflows.

This project demonstrates the effectiveness of combining open-source tools, cloud-ready deployment strategies, and agile development methodologies in building real-world solutions for higher education systems.

6.1.1 Key Learnings and Technical Exposure:

1. Full-stack web development using Next.js and MongoDB Atlas
2. Secure authentication workflows with OAuth2 and encrypted sessions
3. Modular API design, dynamic routing, and error handling
4. Responsive and accessible UI with Tailwind CSS and Preline UI
5. Version control, CI deployment, and platform monitoring using GitHub and Vercel

6.2 FUTURE SCOPE

While the current implementation fulfills its core mission, the portal is engineered to scale and adapt to evolving needs. Several enhancements are proposed to strengthen its functionality, accessibility, and institutional integration:

6.2.1 REAL-TIME INTERACTION

1. Enable real-time notifications for new job postings, events, or announcements

6.2.2 ANALYTICS DASHBOARD

1. Develop admin-only dashboards to monitor engagement metrics and content performance
2. Visualize data through interactive charts (e.g., alumni by batch, event reach, job popularity)

6.2.3 MOBILE APPLICATION

1. Create a native or hybrid mobile app for Android and iOS using tools like React Native or Expo
2. Push notifications for timely updates and streamlined user experience

6.2.4 ENHANCED ROLE SYSTEM

1. Introduce extended roles such as recruiters, faculty, and guests
2. Implement granular permission control via a dynamic role manager

6.2.5 INSTITUTIONAL INTEGRATION

1. Connect the portal with existing SMVDU databases for seamless onboarding
2. Explore **Single Sign-On (SSO)** using institutional credentials (e.g., LDAP or SAML)

6.2.6 INTELLIGENT RECOMMENDATIONS

1. Integrate AI/ML models to personalize job listings, events, and alumni connections
2. Leverage user activity data for relevance-based content delivery

6.2 FINAL REMARK

The successful development and deployment of this portal lays the groundwork for a sustained, digital-first alumni ecosystem at SMVDU. With continuous iteration and university support, the platform can evolve into a multi-featured engagement tool, setting a benchmark for similar institutions aiming to digitize alumni interaction.

CHAPTER 7: REFERENCES

1. **Node.js Documentation** – Node.js v20.x Documentation
<https://nodejs.org/>
2. **Express.js Documentation** – Express.js Guide v4.x
<https://expressjs.com/>
3. **MongoDB Manual** – MongoDB Atlas and Mongoose ODM
<https://www.mongodb.com/docs/>
4. **EJS Documentation** – Embedded JavaScript Templating
<https://ejs.co/>
5. **Passport.js OAuth Guide** – Google OAuth 2.0 Strategy
<https://www.passportjs.org/>
6. **Bootstrap 5 Documentation** – Frontend UI Framework
<https://getbootstrap.com/>
7. **GitHub Repository** – Project Source Code
<https://github.com/Sachinanand99/alumni-smvdu-webapp>
8. **Mozilla Developer Network (MDN)** – Web API Standards and JS Guides
<https://developer.mozilla.org/>
9. **Heroku Documentation** – Cloud Deployment Guide
<https://devcenter.heroku.com/>
10. **Joi Validator** – Data Validation Middleware
<https://joi.dev/>
11. **W3C Web Standards** – HTML, CSS, Accessibility Guidelines
<https://www.w3.org/>
12. **IEEE Xplore** – “Secure Design Patterns for Web Authentication”
<https://ieeexplore.ieee.org/>

APPENDIX I: SAMPLE SCHEMAS AND DATASET

1. DATASET USED:

The core dataset includes student placement and alumni data provided by **Shri Mata Vaishno Devi University (SMVDU)**. It contains historical records such as:

1. Entry numbers and graduation years
2. Department and academic degree
3. Professional affiliation
4. Country of residence
5. Contact details and institutional associations

The dataset is formatted as an **Excel spreadsheet**, segmented into sheets based on graduation year. It serves as the foundation for building searchable alumni directories and validating campus visit requests.

2. SCHEMAS USED:

Below are representative Mongoose schemas and the Excel structure used in the portal:

2.1 ALUMNI (EXCEL-BASED)

Fields stored per row:

“Sno, fullName, email, phone, entryNumber, department, degree[], professionalSector, countryOfResidence, postalAddress, linkedinProfile, companyOrInstitute, income, graduationYear, profilePicture, createdAt”

Stored in structured .xlsx format and dynamically updated using the xlsx library

2.2 USER SCHEMA (MONGODB)

```
{
  universityEmail?: string;
  personalEmail: string;
  password?: string;
  name: string;
```



```
profilePicture?: string;
provider: 'google' | 'local';
createdAt: Date;
resetToken: string;
resetTokenExpiry: Date;
}
```

Manages login metadata, account origin, and recovery flows.

2.3 CAMPUSVISIT SCHEMA

```
{
  name: string;
  email: string;
  phone: string;
  batch: string;
  description: string;
  createdAt: Date;
}
```

Tracks alumni requests for official campus visits.

2.4 EVENT SCHEMA

```
{
  title: string;
  slug: string;
  start_date: Date;
  end_date: Date;
  location: string;
  image: string;
  attendees: number;
  views: number;
```

```

description: string;
}

```

Supports university events with auto-generated slugs and attendance tracking.

2.5 GALLERY SCHEMA

```

{
  eventId: ObjectId;
  url: string;
  caption?: string;
  createdAt: Date;
}

```

Links multimedia files to their corresponding events for archival and display.

APPENDIX II: MODULE VULNERABILITY ANALYSIS

1. VULNERABILITY IN XLSX MODULE

While the `xlsx` npm package is widely used for reading and writing Excel files in Node.js applications, it has known limitations and potential risks:

1. **Deserialization Risk:** If unsanitized or user-uploaded Excel files are parsed directly, malicious content can trigger unexpected behavior or leakage.
2. **Memory Overload:** Large files (especially with complex formatting or embedded media) may cause excessive memory consumption, leading to denial of service.
3. **Lack of Schema Enforcement:** Since Excel files are loosely structured, relying on them for critical data without validation may expose logical vulnerabilities.

Mitigation strategies include strict input validation, buffer size checks, and avoiding direct user uploads without sanitization. Always validate and sanitize `req.body` or `req.file` before parsing with `xlsx.read()`.