A FIELD PROJECT REPORT

on

"ALUMNI MANAGENMENT SYSTEM"

Submitted by:

221FA04013
G. Yasaswi

221FA04379
K.Guna Sekhara Varma

221FA04735
Krishna Kant
Kumar

Under the guidance of

Mr. Pavan Kumar

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
Deemed to be university Vadlamudi, Guntur.
ANDHRA PRADESH, INDIA, PIN-522213.
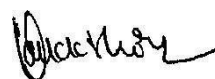
i

CERTIFICATE

This is to certify that the Field Project entitled **"Alumni management system** "that is being submitted by **221FA04013 (G. Yasaswi)**, **221FA04379** (K.Guna Sekhara Varma)**, 221FA04735 (Krishna Kant Kumar)**,for partial fulfilment of Field Project is a bonafide work carried out under the supervision of **Mr. Pavan Kumar, Department of CSE**.

Dr. S.V. Phani Kumar

Assistant Professor, CSE

HOD,CSE

Dr. K.V. Krishna
Kishore Dean, SoCI

DECLARATION

We hereby declare that the Field Project entitled **"Alumni management system"** is being submitted by **221FA04013 (G. Yasaswi)**, **221FA04379 (**K.Guna Sekhara Varma**)**, **221FA04735 (Krishna Kant Kumar**)in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of **Mr. Pavan Kumar, Department of CSE**.

By:
221FA04013 (G.Yasaswi),
221FA04379(K.Guna SekharaVarma),
221FA04735(Krishna Kanth Kumar),

Date:

ABSTRACT

The Alumni Management System is a web-based platform designed to foster communication and engagement between alumni and the institution. It provides features such as discussion forums, job postings, event management, and alumni profile displays. The system ensures that alumni can stay connected with the college and contribute to its growth. Users can register, log in, and access information seamlessly. Event details and job postings are dynamically updated, allowing students to explore opportunities. A secure authentication system verifies user credentials and enables password management. The gallery feature showcases event highlights. The system is designed using modern web technologies for efficiency and scalability. This platform strengthens alumni networking, enhances information sharing, and promotes institutional development

TABLE OF CONTENTS

vi

LIST OF FIGURES

LIST OF TABLES

ABBREIVATIONS

- VUAP : Vignan University Alumni Porta

- RBAC : Role-Based Access Control

- **SQL** : Structured Query Language

- **TLS** : Transport Layer Security

- **CDN** : Content Delivery Network

- **UI** : User Interface

- **UX** : User Experience

- **API** : Application Programming Interface

- **DBMS** : Database Management System

- **ORM** : Object-Relational Mapping

- **JWT** : JSON Web Token

- **CRUD** : Create, Read, Update, Delete

- **SaaS** : Software as a Service

- **CI/CD** : Continuous Integration/Continuous Deployment

- **HTTP** : Hypertext Transfer Protocol

- **JSON** : JavaScript Object Notation

- **TS** : TypeScript

- **JS** : JavaScript

- **CSS** : Cascading Style Sheets

- **HTML** : Hypertext Markup Language

- **IDE** : Integrated Development Environment

CHAPTER - 1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

The Vignan University Alumni Portal is a web application designed to connect alumni and students through a centralized digital platform. The primary goal of the project is to strengthen the university's alumni network by facilitating job opportunities, event management, discussions, and mentorship. This portal enables seamless interaction between alumni, students, and faculty, fostering career growth and community engagement.

Users can create and update profiles, share job postings, participate in discussions, and stay informed about university events. The system ensures secure authentication with role-based access control, providing a safe and interactive experience for all users.

1.2 Literature Survey

A literature survey is crucial in the development process to analyze existing solutions and determine the best technologies. Before building this platform, various alumni engagement systems were studied to understand their strengths and limitations. The survey covered key aspects such as user engagement, security, and scalability.

- The **Agile Development Model** was adopted to ensure iterative improvements and flexibility.
- A study on **Role-Based Access Control (RBAC)** helped define different user roles like alumni, students, and administrators.
- Research on **secure authentication mechanisms** ensured the implementation of session-based login and encryption protocols.
- **Database management strategies** were reviewed to ensure efficient handling of user data and interactions.

1.3 Project Background

Many universities struggle to maintain a strong alumni network, leading to lost career opportunities and reduced engagement. Traditional methods, such as email updates and manual record-keeping, are inefficient and outdated. To address this, the Vignan University Alumni Portal leverages modern web technologies to offer an interactive, user-friendly platform that bridges the gap between students and alumni.

The portal digitizes alumni interactions, simplifies event organization, and enhances job opportunities. By providing a centralized system, the project aims to strengthen alumni relations and create a lasting impact on students' professional growth.

1.4 Objective

The objective of this system is to streamline alumni-student interactions and automate engagement activities. The portal should:

- Facilitate **profile management** for alumni and students.
- Enable **job postings** and **internship listings** to support career growth.

- Provide an **event management** system for alumni reunions and university-hosted activities.

- Offer a **discussion forum** for networking, mentorship, and information exchange.

- Implement **secure authentication** using role-based access control.

- Showcase achievements through the **"Peak Performers"** section.

1.5 Project Description

This portal consists of two primary user types: **Alumni & Students** and **Administrators**.

1.5.1 Alumni & Students

- **Sign Up & Log In**: Users can register with their university credentials and log in securely.

- **Profile Management**: Users can update personal details, achievements, and career status.

- **Job Board**: Alumni and companies can post job openings and internships.

- **Event Participation**: Users can view, register for, and participate in university events.

- **Discussion Forums**: A space for alumni and students to engage in meaningful conversations.

- **Photo Gallery**: Showcasing university events and alumni contributions.

1.5.2 Administrators

- **Manage Users**: Verify alumni and student profiles.

- **Monitor Content**: Ensure discussions, job posts, and events adhere to guidelines.

- **Event Oversight**: Approve and promote alumni events.

The platform is hosted on a **cloud-based infrastructure**, ensuring high availability and scalability. **PostgreSQL** is used for database management, while the frontend is built using **React and TypeScript** for a smooth user experience.

CHAPTER - 2

SOFTWARE REQUIREMENT SPECIFICATION

2.SOFTWARE REQURINMENTS AND SPEFICATION

2.1 Requirement Analysis

To ensure easy access and portability, the Vignan University Alumni Portal is designed as a web and mobile-responsive application. It will be accessible on Windows, macOS, Linux (Ubuntu), Android, and iOS platforms. The system enables seamless user interactions with an intuitive interface for both desktop and mobile devices.

The key documentation supporting this development includes:

1. Problem Statement
2. Data Flow Diagrams
3. Use Case Diagrams
4. Other UML Diagrams

These documents provide a diagrammatic representation of the system.

2.2 Problem Statement

Universities often struggle with alumni engagement due to lack of a centralized communication platform. The problem revolves around:

- Difficulty in connecting alumni with students for mentorship and job opportunities.
- Event management inefficiencies due to outdated communication methods.
- Lack of structured job listings and career growth opportunities within the alumni network.
- Absence of a dedicated discussion forum for knowledge sharing.

The Vignan University Alumni Portal addresses these issues by offering a centralized and mobile-friendly web platform.

2.3 Functional Requirements

1. User Roles

    1. Alumni & Students

        o Sign up & Login (Secure authentication with session management).

        o Profile Management (Edit personal details, career status, and achievements).

        o Job Board (Post job/internship listings and apply for jobs).

        o Event Management (View, register for, and manage events).

        o Discussion Forums (Engage in professional and academic discussions).

        o Photo Gallery (Upload and view university event photos).

2. Administrators
    o User Management (Verify and manage alumni and student profiles).
    o Content Moderation (Monitor discussions, job postings, and events).
    o Event Oversight (Approve, edit, or promote events).

2.4 Software Requirement Specification

2.4.1 Purpose

This document details the purpose, features, and constraints of the Vignan University Alumni Portal. The system should:

- Provide a responsive interface that works across desktop and mobile.
- Support real-time interactions via job postings, discussions, and event updates.
- Ensure secure authentication and role-based access control (RBAC).

2.4.2 Scope of the Project

The project involves creating a mobile-responsive website with two main user types:

1. Alumni & Students
    o Can register, log in, and manage profiles.
    o Can post jobs, apply for jobs, and attend events.
    o Can participate in discussions and network with peers.
2. Administrators
    o Manage user authentication and profile verification.
    o Moderate job postings, discussions, and events.
    o Ensure content security and compliance.

2.4.3 Technologies Used

Frontend (Mobile + Web)

- React.js + TypeScript (Frontend Framework).
- Vite (Build Tool).
- Tailwind CSS + Shadcn/UI (UI Styling & Components).
- Wouter (Lightweight Router for navigation).
- TanStack Query (State management & API handling).

Backend

- Node.js + Express.js (Server-side framework).
- TypeScript (For maintainable and scalable code).
- Drizzle ORM + PostgreSQL (Database management).
- Passport.js + Express-Session (Authentication & security).

Cloud & Deployment

- Cloud Platform: Azure (Preferred) or AWS.
- Database Hosting: PostgreSQL (Cloud-hosted).
- Storage: Cloud-based storage for media uploads.

2.5 Software & Hardware Requirements

Software Requirements

Component	Specification
Operating System	Windows 10+, macOS, Ubuntu, Android, iOS
Browser	Chrome, Firefox, Safari, Edge
Development Stack	React.js (Frontend), Node.js (Backend)
Database	PostgreSQL
Cloud Hosting	Azure, AWS

Table-1

Hardware Requirements

OS	Browser	Disk Space	RAM
Windows/macOS/Linux	Any Modern Browser	250MB	2GB+
Android/iOS	Mobile Browser/App	150MB	2GB+

Table-2

Server Requirements

OS	Software	Processor	RAM	Disk Space
Ubuntu 20.04+	Node.js, PostgreSQL, Express	Intel Xeon	8GB+	50GB+

Table-3

2.6 Functional Modules

1. Alumni & Student Functionalities

   1. Sign Up & Login
      - Secure authentication with email/OTP verification.
   2. Profile Management
      - Update personal, professional, and academic details.
   3. Job Board
      - Post jobs, apply for jobs, and receive notifications.
   4. Event Management
      - View upcoming alumni events, register, and get updates.
   5. Discussion Forum
      - Engage in discussions and seek mentorship.

8

6. Photo Gallery
   - View and upload images from university events.
7. Logout
   - Securely log out of the platform.

2. Admin Functionalities

1. User Management
   - Approve or reject user registrations.
2. Content Moderation
   - Monitor and remove inappropriate job listings or discussions.
3. Event Oversight
   - Approve or edit event listings.

2.7 Non-Functional Requirements

2.7.1 Performance Requirements

- The system must load within 3 seconds on mobile and desktop.
- Database queries should execute within 5 seconds.

2.7.2 Design Constraints

- Must be responsive and mobile-friendly.
- UI should follow Vignan University's branding (peacock-themed).

2.7.3 Standards Compliance

- WCAG Accessibility Standards to ensure usability.
- Secure password storage (bcrypt hashing).

2.7.4 Availability & Reliability

- 99.9% uptime on cloud servers.
- Data backups every 24 hours.

2.7.5 Portability

- Compatible with mobile devices, tablets, and desktops.

2.7.6 Security

- Encrypted user authentication using Passport.js.
- Role-Based Access Control (RBAC) for different users.

2.8 External Interface Requirements

2.8.1 User Interface

- Mobile-responsive UI with easy navigation.
- Interactive dashboard for jobs, events, and discussions.

2.8.2 Database Tables

Users Table

- User ID, Name, Email, Phone, Role

Jobs Table

- Job ID, Title, Description, Company, Posted By

Events Table

- Event ID, Name, Date, Location, Description

Forum Table

- Post ID, Title, Author, Comments

2.9 Feasibility Study

- Helps Vignan University engage alumni and students effectively.
- Cost-effective cloud hosting and open-source tools.
- Uses scalable technologies with mobile support.
- Simple user-friendly UI with guided onboarding.

CHAPTER - 3

ANALYSIS & DESIGN

3. ANALYSIS & DESIGN

3.1 Introduction

3.1.1 Purpose

3.1.1.1 Document   Purpose

This Software Design Document (SDD) serves as a blueprint for the development of the Vignan University Alumni Portal. It provides a detailed representation of the system's design, aiding in communication among developers, designers, and stakeholders.

3.1.1.2 Project   Purpose

The primary goal of the Vignan University Alumni Portal is to establish a centralized web-based platform for alumni to stay connected, share updates, engage in discussions, and contribute to the university's growth. The system will facilitate authentication, profile management, event organization, job postings, and more.

3.1.2 Scope

3.1.2.1 Document   Scope

This document outlines the architecture, UI/UX design, database structure, and functional modules of the system. It details the higher-level module designs but does not cover implementation details of individual classes and functions.

3.1.2.2 Project   Scope

The Alumni Management System consists of multiple modules, each catering to a specific user role and function:

1. Alumni – Users can register, update profiles, participate in forums, view job postings, and engage in events.
2. University Admin – Manages alumni records, moderates discussions, and organizes events.
3. Recruiters – Can post job openings and search for potential candidates from alumni profiles.
4. Event Organizers – Plan, schedule, and manage alumni events and meetups.

3.2 System Overview

The system follows a client-server architecture and is developed using modern web technologies:

- Frontend: React, TypeScript, Vite, Tailwind CSS, Shadcn/UI, Radix UI, Lucide React, TanStack Query, React Hook Form, Zod, Wouter.
- Backend: Node.js, Express, TypeScript, Passport.js, Express Session, Crypto, Drizzle ORM, Drizzle Zod.
- Database: Initially in-memory storage, planned migration to PostgreSQL.
- Hosting: Cloud-based deployment with role-based access control (RBAC) for security.

3.3 System Architecture

3.3.1   Architectural Design

The system follows a three-tier architecture:

1. Presentation Layer (Frontend): Handles user interface and interactions.
2. Business Logic Layer (Backend): Processes requests, applies validation, and enforces security.
3. Data Layer (Database): Stores alumni profiles, job postings, event details, and forum discussions.

Key attributes of the architecture:

- Secure authentication with Passport.js and session management.
- Role-based access control (RBAC) for user-specific functionalities.
- API communication using RESTful endpoints.
- Data validation with Zod to ensure consistency and security.
- Deployment flexibility with cloud hosting options.

3.3.2 Application Components

1. View Layer (Frontend) – Implements UI elements, allowing alumni, recruiters, and admins to interact with the system.
2. Business Logic Layer – Manages authentication, authorization, and system workflows.
3. Data Access Layer – Handles CRUD operations for alumni profiles, job postings, and event management.
4. Security and Error Handling – Implements encryption, session management, and robust error handling mechanisms.

3.4 Data Design

3.4.1 Database Design (PostgreSQL)

Alumni Table – Stores alumni details.

Column Name	Type	Constraints
id	INT	PRIMARY KEY, AUTO INCREMENT
name	VARCHAR(50)	NOT NULL
email	VARCHAR(100)	UNIQUE, NOT NULL
phone	VARCHAR(15)	UNIQUE, NOT NULL
graduation_year	INT	NOT NULL
department	VARCHAR(50)	NOT NULL
profile_picture	TEXT	NULL
password_hash	TEXT	NOT NULL

Table-4

13

Job Postings Table – Stores job postings made by recruiters.

Column Name	Type	Constraints
id	INT	PRIMARY KEY, AUTO INCREMENT
title	VARCHAR(100)	NOT NULL
description	TEXT	NOT NULL
company	VARCHAR(100)	NOT NULL
location	VARCHAR(100)	NOT NULL
alumni_id	INT	FOREIGN KEY REFERENCES Alumni(id)

Table-5

Events Table – Stores information about alumni events.

Column Name	Type	Constraints
id	INT	PRIMARY KEY, AUTO INCREMENT
name	VARCHAR(100)	NOT NULL
date	DATE	NOT NULL
location	VARCHAR(100)	NOT NULL
description	TEXT	NOT NULL
organizer_id	INT	FOREIGN KEY REFERENCES Alumni(id)

Table-6

Discussion Forum Table – Stores alumni discussions.

Column Name	Type	Constraints
id	INT	PRIMARY KEY, AUTO INCREMENT
topic	VARCHAR(100)	NOT NULL
content	TEXT	NOT NULL
alumni_id	INT	FOREIGN KEY REFERENCES Alumni(id)

Table-7

Peak Performers Table – Highlights top alumni achievements.

Column Name	Type	Constraints
id	INT	PRIMARY KEY, AUTO INCREMENT
alumni_id	INT	FOREIGN KEY REFERENCES Alumni(id)
achievement	TEXT	NOT NULL

Table-8

CHAPTER - 4

MODELING

4.MODELING

4. MODELING

4.1 Design

Requirements gathering followed by careful analysis leads to a systematic Object-Oriented Design (OOAD). Various activities have been identified and are represented using Unified Modeling Language (UML) diagrams. UML is used to specify, visualize, modify, construct, and document the artifacts of an object-oriented software-intensive system under development.

4.1.1 Use Case Diagram

In the Unified Modeling Language (UML), the use case diagram is a type of behavioral diagram defined by and created from a use-case analysis. It represents a graphical overview of the functionality of the system in terms of actors, which are persons, organizations, or external systems that play a role in one or more interactions with the system. These are drawn as stick figures. The goals of these actors are represented as use cases, which describe a sequence of actions that provide something of measurable value to an actor and any dependencies between those use cases.

In this application, there are actors such as Alumni, Admin, and Guest, each interacting with the system in different ways. The use case diagram should depict these roles and their interactions.
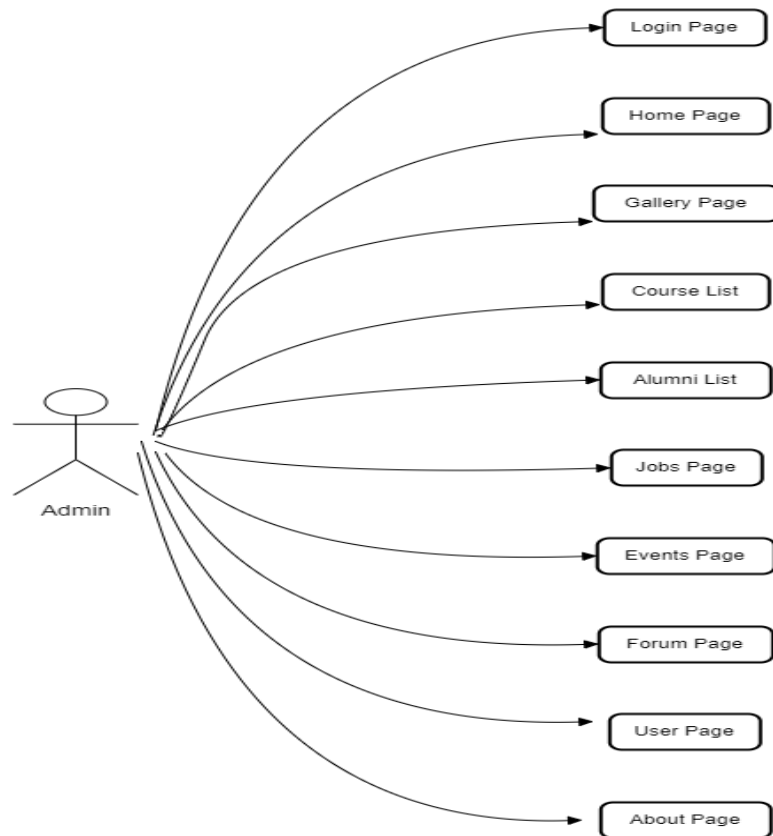
Fig-1

4.1.2 Sequence Diagram

UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom (i.e., lower equals later).

Sequence diagrams for the system would be created for various modules, such as:

- Alumni Registration Process
- Admin Management of Alumni Profiles
- These diagrams detail how different components (e.g., user input, database interaction, and response generation) communicate during specific operations.

18
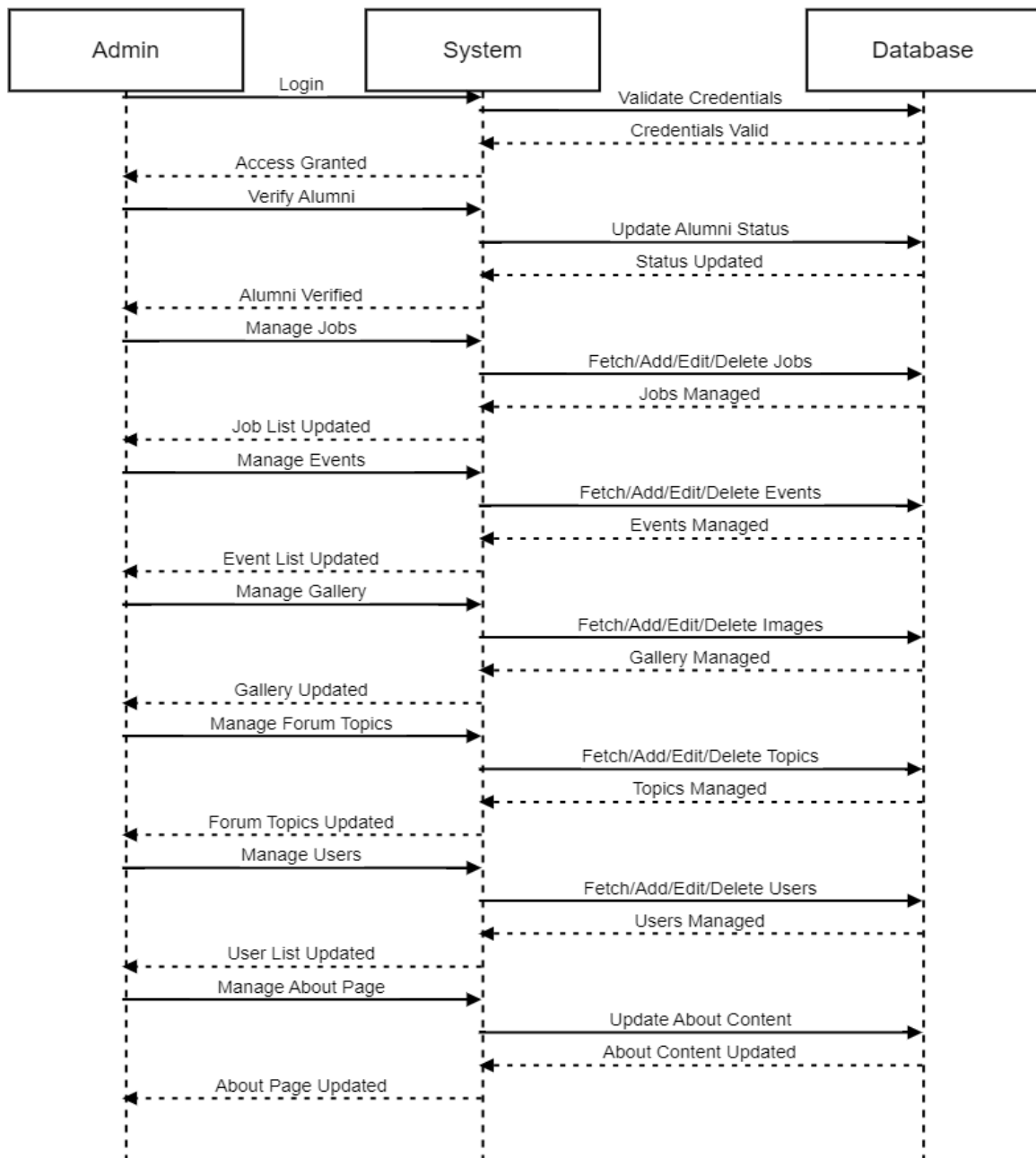
Fig-2

4.1.3 ER Diagram

An Entity-Relationship (ER) diagram is used to represent the database structure. It describes the entities and relationships between them in a relational database. For the Alumni Management System, entities may include Alumni, Admin, Events, Jobs, and Forums, with relationships like Alumni posts Jobs or Admin manages Events.
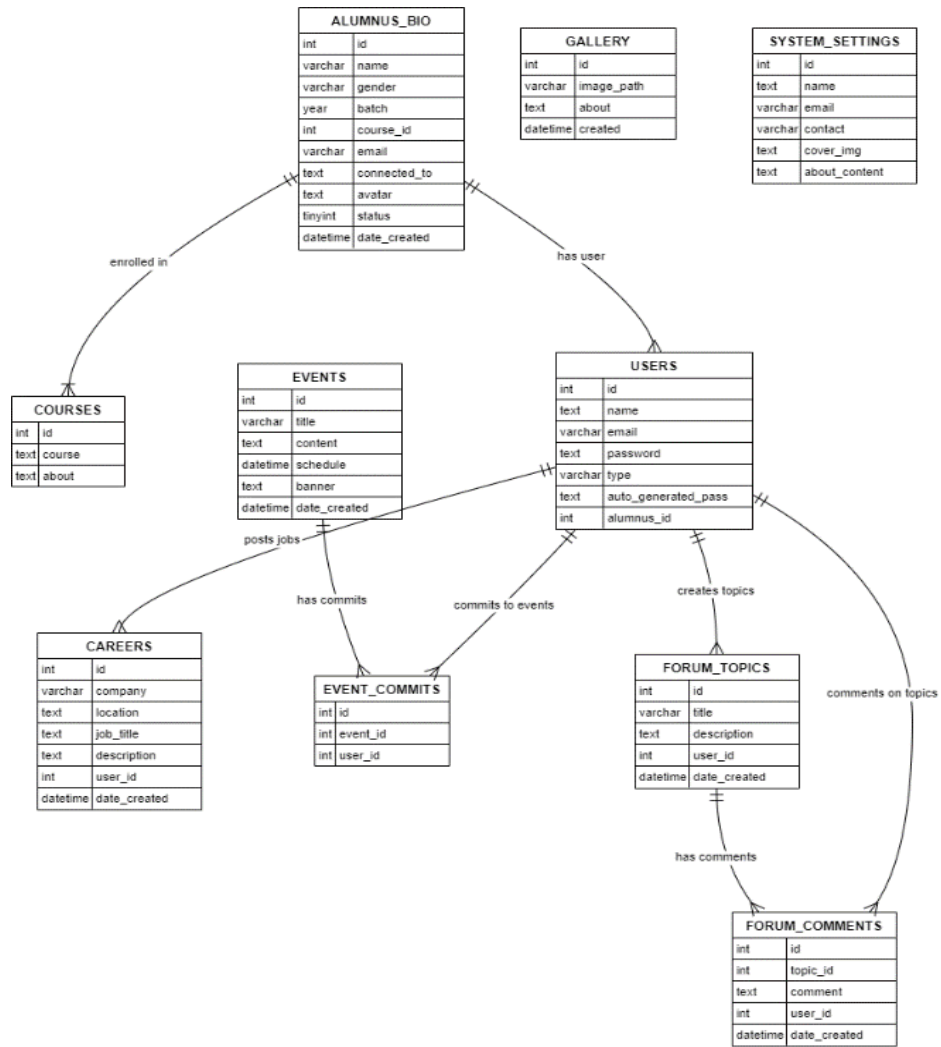
ALUMNUS_BIO

int	id
varchar	name
varchar	gender
year	batch
int	course_id
varchar	email
text	connected_to
text	avatar
tinyint	status
datetime	date_created

GALLERY

int	id
varchar	image_path
text	about
datetime	created

SYSTEM_SETTINGS

int	id
text	name
varchar	email
varchar	contact
text	cover_img
text	about_content

COURSES

int	id
text	course
text	about

EVENTS

int	id
varchar	title
text	content
datetime	schedule
text	banner
datetime	date_created

USERS

int	id
text	name
varchar	email
text	password
varchar	type
text	auto_generated_pass
int	alumnus_id

CAREERS

int	id
varchar	company
text	location
text	job_title
text	description
int	user_id
datetime	date_created

EVENT_COMMITS

int	id
int	event_id
int	user_id

FORUM_TOPICS

int	id
varchar	title
text	description
int	user_id
datetime	date_created

FORUM_COMMENTS

int	id
int	topic_id
text	comment
int	user_id
datetime	date_created

enrolled in

has user

posts jobs

has commits

commits to events

creates topics

comments on topics

has comments

Fig-3

20

CHAPTER - 5

IMPLEMENTATION

5.IMPLEMENTATION

5.1   Sample Code

5.1.1Code for index page

```typescript
import { useMutation, useQuery } from "@tanstack/react-query";
import { PageLayout } from "@/components/layout/page-layout";
import { ProfileForm } from "@/components/profile/profile-form";
import { User } from "@shared/schema";
import { apiRequest, queryClient } from "@/lib/queryClient";
import { useToast } from "@/hooks/use-toast";
import { useAuth } from "@/hooks/use-auth";
import { Card, CardContent } from "@/components/ui/card";
import { Skeleton } from "@/components/ui/skeleton";

export default function ProfilePage() {
  const { user } = useAuth();
  const { toast } = useToast();

  // Fetch latest user data
  const { data: userData, isLoading } = useQuery<User>({
    queryKey: ["/api/user"],
    initialData: user || undefined,
  });

  // Update profile mutation
  const updateProfileMu    (parameter) updatedProfile: any
    mutationFn: async (updatedProfile: Partial<User>) => {
      const res = await apiRequest("PUT", `/api/profile`, updatedProfile);
      return await res.json();
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["/api/user"] });
      toast({
        title: "Profile updated",
        description: "Your profile has been updated successfully.",
      });
    },
    onError: (error: Error) => {
      toast({
        title: "Profile update failed",
        description: error.message,
```

5.1.2 Code for admin login

```
function LoginForm() {
  const { loginMutation } = useAuth();
  const form = useForm<LoginFormValues>({
    resolver: zodResolver(loginSchema),
    defaultValues: {
      username: "",
      password: "",
    },
  });

  function onSubmit(values: LoginFormValues) {
    // Get user type from parent component
    const userType = document.querySelector('.userType-alumni')?.classList.contains('bg-primary')
      ? 'alumni'
      : 'admin';

    // For admin login, validate against hardcoded credentials
    if (userType === 'admin' && values.username === 'admin' && values.password === 'admin123') {
      loginMutation.mutate({
        username: 'admin',
        password: 'admin123'
      });
    } else {
      // For regular alumni login
      loginMutation.mutate(values);
    }
  }
}
```

5.1.3 code of alumni login

```
<div className="flex justify-center mb-6">
  <div className="inline-flex rounded-md shadow-sm">
    <button
      onClick={() => setUserType("alumni")}
      className={`userType-alumni px-4 py-2 text-sm font-medium rounded-l-md ${
        userType === "alumni"
          ? "bg-primary text-white"
          : "bg-gray-100 text-gray-700"
      } transition-colors`}
    >
      Alumni
    </button>
    <button
      onClick={() => setUserType("admin")}
      className={`userType-admin px-4 py-2 text-sm font-medium rounded-r-md ${
        userType === "admin"
          ? "bg-primary text-white"
          : "bg-gray-100 text-gray-700"
      } transition-colors`}
```

5.1.3 code of login schema

```
// Login Form Schema
const loginSchema = z.object({
  username: z.string().min(3, "Username must be at least 3 characters"),
  password: z.string().min(6, "Password must be at least 6 characters"),
});


type LoginFormValues = z.infer<typeof loginSchema>;
```

5.1.4 code of registration schema

```
// Registration Form Schema
const currentYear = new Date().getFullYear();
// Create an array from current year to 2040, plus past 20 years for alumni
const futureYears = Array.from({ length: 2040 - currentYear + 1 }, (_, i) => currentYear + i);
const pastYears = Array.from({ length: 20 }, (_, i) => currentYear - i - 1);
const graduationYears = [currentYear, ...futureYears, ...pastYears].sort((a, b) => a - b);

const registrationSchema = z.object({
  username: z.string().min(3, "Username must be at least 3 characters"),
  password: z.string().min(6, "Password must be at least 6 characters"),
  confirmPassword: z.string().min(6, "Confirm password must be at least 6 characters"),
  email: z.string().email("Please enter a valid email address"),
  firstName: z.string().min(2, "First name must be at least 2 characters"),
  lastName: z.string().min(2, "Last name must be at least 2 characters"),
  graduationYear: z.number().positive("Please select a graduation year"),
  degree: z.string().min(1, "Please select a degree"),
  isAdmin: z.boolean().default(false),
}).refine((data) => data.password === data.confirmPassword, {
  message: "Passwords do not match",
  path: ["confirmPassword"],
});
```

5.1.5 code of dashboard cards

```
import { Link } from "wouter";
import {
  Calendar,
  Briefcase,
  MessageSquare,
  ArrowRight,
} from "lucide-react";
import { Card, CardContent, CardDescription, CardFooter, Card
import { Badge } from "@/components/ui/badge";
import { Avatar, AvatarFallback, AvatarImage } from "@/compon

interface EventCardProps {
  event: {
    id: number;
    title: string;
    date: Date;
    location: string;
  };
}
```

5.1.6 code of image url posting

```
const handleImageUrlChange = (url: string) => {
  form.setValue("imageUrl", url);
  setPreviewUrl(url);
};

const handleFileUpload = (event: React.ChangeEvent<HTMLInputElement>) => {
  const file = event.target.files?.[0];
  if (file) {
    const reader = new FileReader();
    reader.onload = (e) => {
      const result = e.target?.result as string;
      setPreviewUrl(result);
      form.setValue("imageUrl", "file-upload-" + file.name); // Set a placeholder to pass validation
    };
    reader.readAsDataURL(file);
  }
};
```

5.2 Screen Captures
5.2.1 home screen

Fig-4

5.2.2 alumni login

Fig-5

5.2.3 admin login

Fig-6

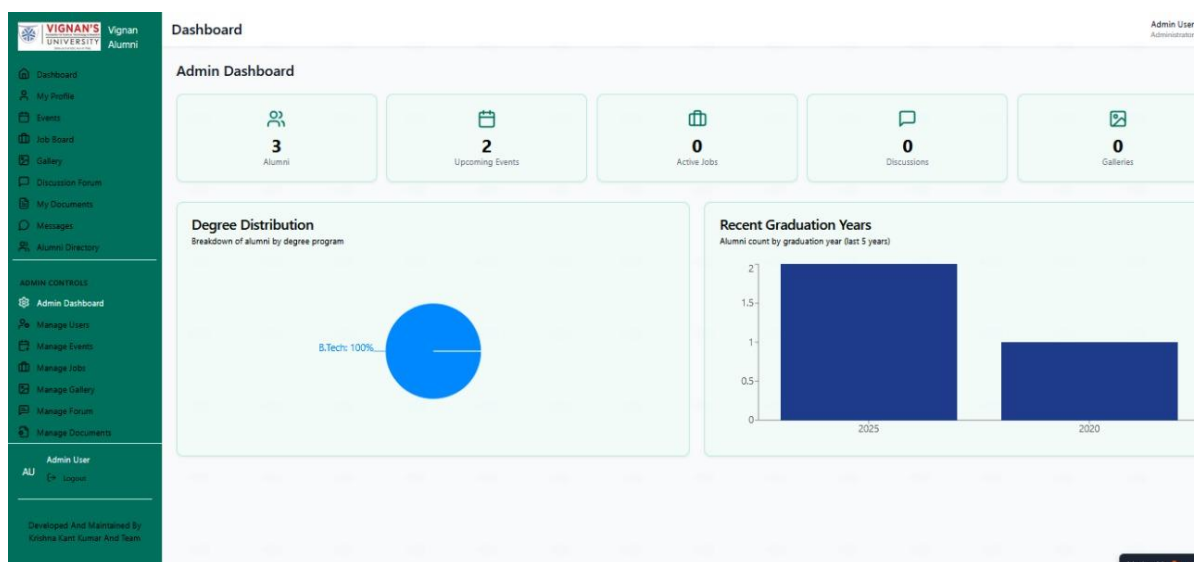### 5.2.4 admin dashboard

Fig-7

5.2.5 alumni dashboard
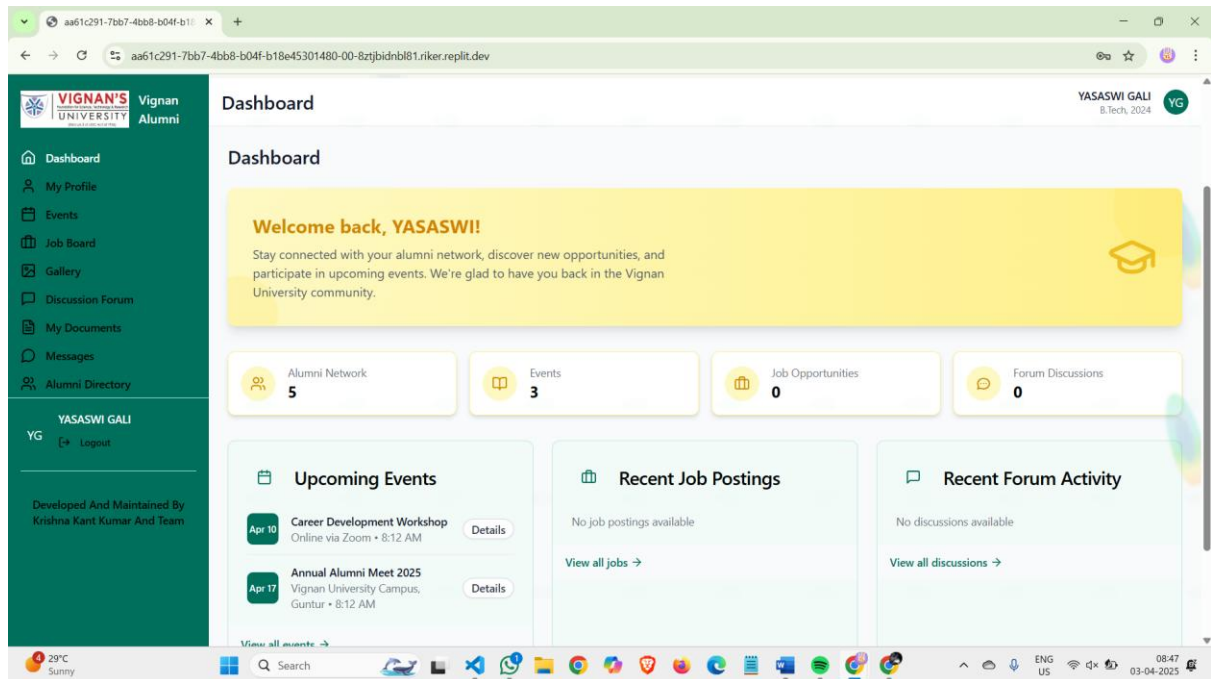
Fig-8

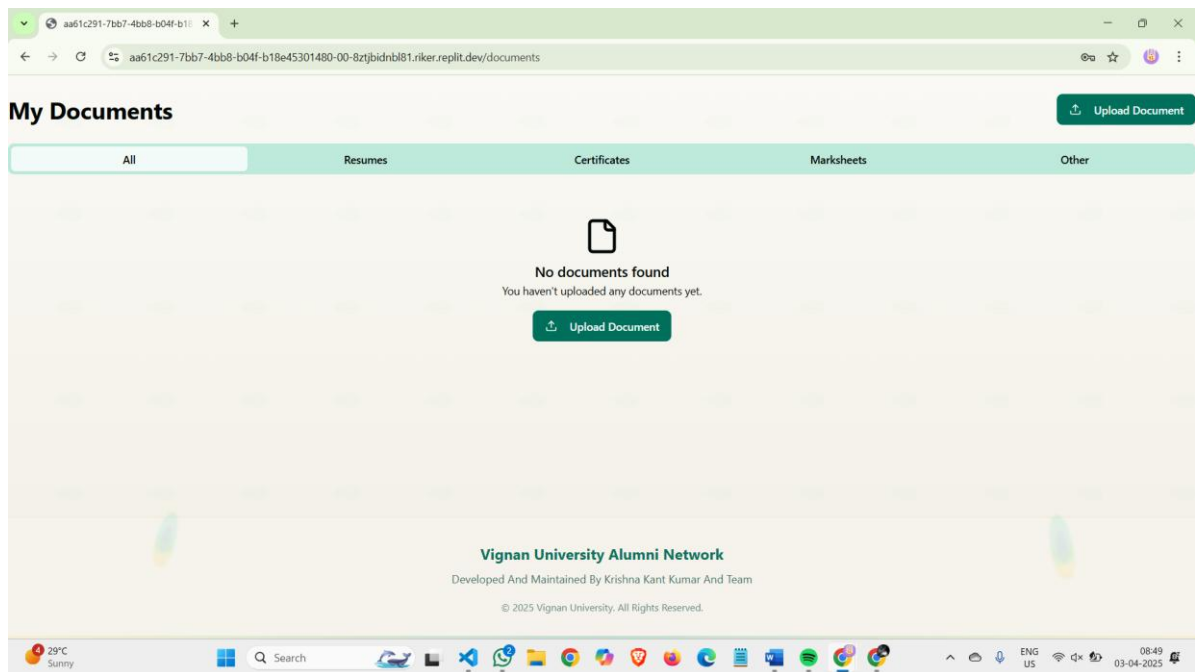## 5.2.6 events
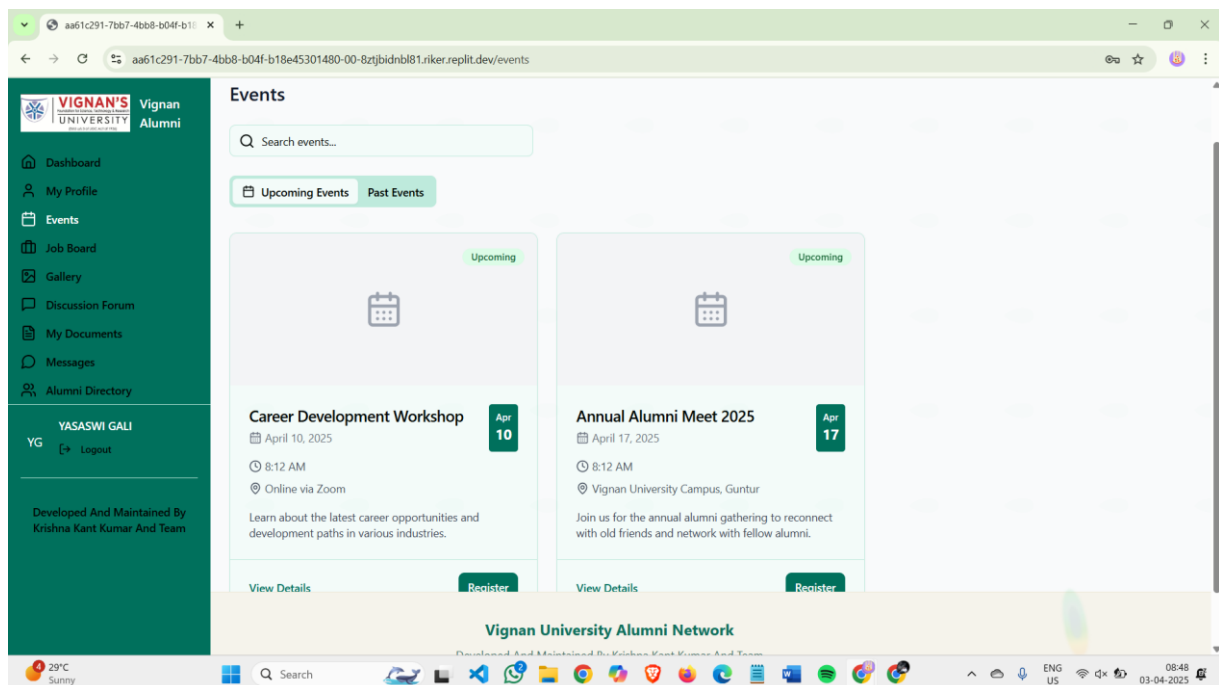
Fig-9

5.2.7 documents

Fig-10

5.2.8 profile page

Fig-11

CHAPTER - 6

TESTING

6.TESTING

6.1 Software Testing

Software testing is the process of validating and verifying that a software application meets the technical requirements involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. Testing ensures the quality of the software. Various types of testing include manual testing, unit testing, black-box testing, performance testing, stress testing, regression testing, white-box testing, etc. Among these, performance testing and load testing are particularly important for web applications, and the following sections deal with some of these types.

6.2 Black Box Testing

Black box testing treats the software as a "black box"—without any knowledge of the internal implementation. In this type of testing, the tester focuses on inputs and outputs of the application. Common methods of black-box testing include:

- Equivalence Partitioning
- Boundary Value Analysis
- All-Pairs Testing
- Fuzz Testing
- Model-Based Testing
- Traceability Matrix
- Exploratory Testing
- Specification-Based Testing

For the Alumni Management System, black-box testing will focus on validating features like Alumni Profile Creation, Event Registration, and Job Posting by providing various inputs and validating their outputs.

6.3 White Box Testing

White box testing is when the tester has access to the internal data structures and algorithms, including the code that implements these. This type of testing focuses on verifying the internal workings of the system, including:

- Data flow
- Control flow
- Logic paths

For the Alumni Management System, white-box testing will be applied to the backend functionality, such as Authentication, Profile Updates, and Database Queries.

6.4 Performance Testing

Performance testing is conducted to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system, such as scalability, reliability, and resource usage. For the Alumni Management System, performance testing will measure the response time and scalability when multiple users are interacting with the system simultaneously.

6.5 Load Testing

Load testing is primarily concerned with testing whether the system can continue to operate under specific load, such as large quantities of data or a high number of users. For the Alumni Management System, load testing will simulate multiple users accessing the system at the same time (e.g., Alumni registering, posting jobs, attending events).

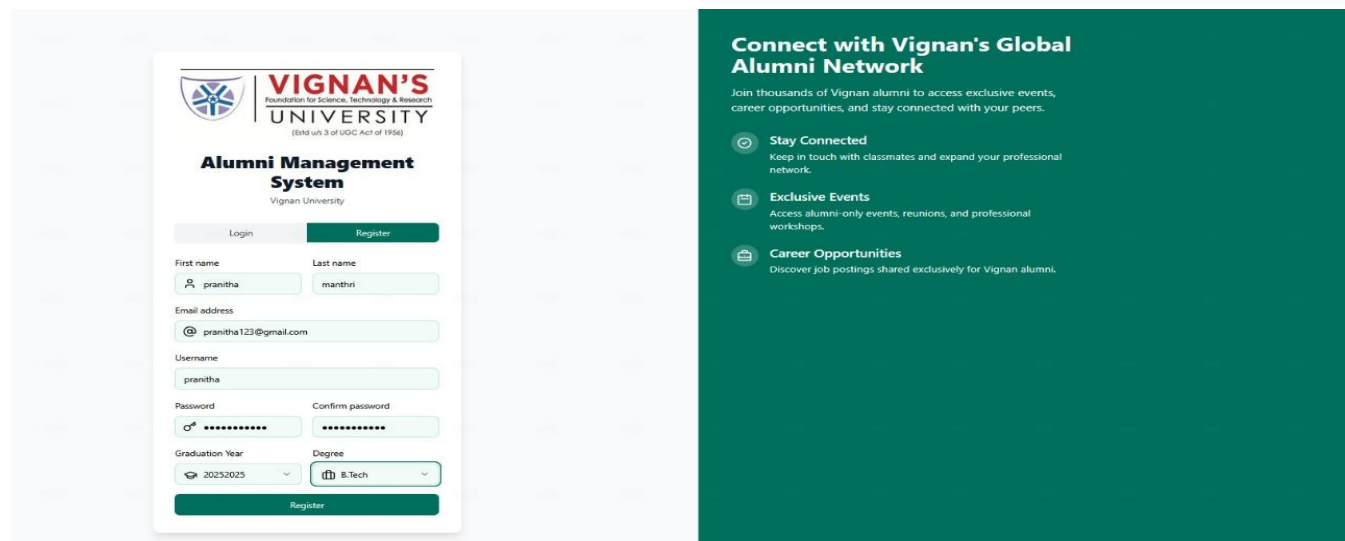Leave space for the Load Testing Graph or Results here.

6.6 Manual Testing

Manual testing is the process of manually testing the software for defects. The functionality of the application is manually tested to ensure correctness. The test cases for Alumni Registration, Event Management, and Job Posting are manually tested, among others.

Sample Test Cases

Test Case 1: Inserting New Alumni Profile

- Test Case Name: Inserting New Alumni Profile
- Description: If all fields for the alumni profile are correctly filled in.
- Expected Output: New Alumni Profile Created.

Fig-12

Test Case 2: Admin Managing Alumni Records

- Test Case Name: Admin Managing Alumni Profile
- Description: Admin updating the alumni records.
- Expected Output: Alumni Record Updated.

Fig-13

CHAPTER - 7

RESULTS &CHALLENGES

7.RESULTS

7.1 Results

The Vignan University Alumni Portal successfully provides an interactive platform for alumni to connect, engage, and contribute to the university community. The system enables alumni to maintain updated profiles, explore job opportunities, participate in events, and engage in discussions. The key functionalities of the system include:

1. User Roles & Access Control:
    - Alumni: Can register, update their profile, search for fellow alumni, participate in discussions, and explore job opportunities.
    - Admin: Manages alumni data, moderates discussions, verifies job postings, and oversees event management.
2. Authentication & Security:
    - Secure login and session management using Passport.js with encryption for user credentials.
    - Role-based access control to ensure appropriate permissions.
3. Profile Management:
    - Alumni can update their personal and professional details, including achievements and career progress.
4. Job Board:
    - Alumni and organizations can post job opportunities, and users can apply directly through the platform.
5. Event Management:
    - The admin can create, update, and promote alumni-related events, while alumni can RSVP and participate.
6. Discussion Forum:
    - Alumni can engage in meaningful discussions, share experiences, and seek advice.
7. Photo Gallery:
    - Users can upload and view photos from past alumni events.
8. Peak Performers Section:
    - Highlights successful alumni, inspiring current students and fellow graduates.

7.2 Challenges

- Understanding Alumni Needs:
  - Identifying key features that would provide real value to both recent graduates and experienced alumni.
- Implementing Secure Authentication & Session Management:
  - Ensuring robust security mechanisms, including encryption and session handling, to prevent unauthorized access.
- Designing a Responsive & Engaging UI:
  - Creating an intuitive Peacock-themed UI with animations and responsive design across different devices.
- Managing Real-time Updates:
  - Handling live job postings, event registrations, and forum discussions efficiently.
- Database Integration & Migration:
  - Transitioning from an in-memory storage system to PostgreSQL using Drizzle ORM, ensuring data consistency.
- Learning & Implementing New Technologies:
  - Adopting modern tools like React, TypeScript, Vite, Tailwind CSS, Shadcn/UI, TanStack Query, React Hook Form, and Drizzle ORM with minimal guidance.

CHAPTER - 8

CONCLUSIONS & FUTURE WORK

8.CONCLUSION

8.1 Conclusions

The **Vignan University Alumni Management System** has been successfully designed and developed as a fully functional mobile and web application. The system effectively facilitates alumni engagement by providing features such as profile management, job postings, event coordination, and discussion forums. Developing this project involved overcoming challenges related to authentication, session management, and seamless user interactions. However, the experience has been highly rewarding, enhancing my understanding of full-stack development and improving my problem-solving skills.

8.2 Scope for Future Work

The system can be further enhanced by incorporating the following features:

- Implementing **real-time notifications** for instant updates on jobs, events, and discussions.
- Introducing **AI-driven recommendations** for networking and career opportunities.
- Enhancing the **photo gallery** with better media categorization and management.
- Adding **video conferencing support** for virtual alumni meetups and mentorship sessions.
- Improving **data analytics and reporting** to provide insights into alumni engagement.
- Strengthening **security features** to ensure better data protection and privacy.

8.3 Limitations

Although the **Alumni Management System** provides a comprehensive platform for alumni interaction, a few limitations exist:

- The system does not currently include **end-to-end encrypted messaging**, which could improve secure communication.
- **Advanced search and filtering options** for job listings and alumni profiles can be enhanced.
- Integration with **external professional networking platforms** (e.g., LinkedIn) is yet to be implemented.
- **Offline mode** for accessing certain features without an internet connection can be explored.

BIBLIOGRAPHY

1. Wildavsky, A. (1964). *The Politics of the Budgetary Process*. Boston: Little, Brown and Company.

2. Pyhrr, P. (1970). *Zero-Base Budgeting: A Practical Management Tool for Evaluating Expenses*. Wiley.

3. Grable, J. E., & Joo, S. H. (2001). A further examination of financial help-seeking behavior. *Financial Counseling and Planning, 12*(1), 55-74.

4. Thaler, R., & Sunstein, C. R. (2008). *Nudge: Improving Decisions About Health, Wealth, and Happiness*. Yale University Press.

5. Shefrin, H. M., & Thaler, R. H. (1988). The Behavioral Life-Cycle Hypothesis. *Economic Inquiry, 26*(4), 609-643.

6. Thaler, R. H. (1999). Mental Accounting Matters. *Journal of Behavioral Decision Making, 12*(3), 183-206.

7. Heath, C., & Soll, J. B. (1996). Mental budgeting and consumer decisions. *Journal of Consumer Research, 23*(1), 40-52.

8. Dholakia, U. M., Tam, L., & Chan, E. Y. (2016). The role of mobile apps in financial management. *Journal of Consumer Research, 43*(1), 101-112.

9. Buchanan, B., & DeLong, J. B. (2020). Artificial intelligence in personal finance: Impacts on budgeting and forecasting. *Journal of Finance and Technology, 12*(3), 45-60.

10. Choi, J. J., Laibson, D., Madrian, B. C., & Metrick, A. (2004). Forgetting the Ties That Bind: Implications for 401(k) Savings Behavior. *Journal of Pension Economics and Finance, 3*(3), 233-253.

11. Xiao, J. J., & O'Neill, B. (2016). Consumer financial education and financial satisfaction: An exploratory study. *Journal of Personal Finance, 15*(1), 69-88.

12. Hilgert, M. A., & Hogarth, J. M. (2003). Household Financial Management: The Connection between Knowledge and Behavior. *Federal Reserve Bulletin, 89*(7), 309-322.

13. Fisher, P. J., & Montalto, C. P. (2010). Effect of saving motives and horizon on saving behaviors. *Journal of Economic Psychology, 31*(1), 92-105.

14. Lusardi, A., & Mitchell, O. S. (2011). Financial literacy and retirement planning in the United States. *Journal of Pension Economics and Finance, 10*(4), 509-525.

15. Kahneman, D., & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica, 47*(2), 263-291.

16. Benartzi, S., & Thaler, R. H. (2007). Heuristics and biases in retirement savings behavior. *Journal of Economic Perspectives, 21*(3), 81-104.

17. Mullainathan, S., & Shafir, E. (2013). *Scarcity: Why Having Too Little Means So Much*. Times

Books.

18. Loibl, C., Kraybill, D. S., & DeMay, S. W. (2011). Accounting for the role of habit in regular saving. *Journal of Economic Psychology, 32*(4), 581-592.

19. Mandell, L., & Klein, L. S. (2009). The impact of financial literacy education on subsequent financial behavior. *Journal of Financial Counseling and Planning, 20*(1), 15-24.