**ALUMNET - THE ALUMNI NETWORK PORTAL**

*A*

*Mini Project Report*

*Submitted in partial fulfilment of the Requirements for the award of the Degree of*

# BACHELOR OF ENGINEERING

IN

# INFORMATION TECHNOLOGY

By

#### NOMULA VARSHITH 1602-23-737-061

**BODDU TANMAI TERESA 1602-23-737-059**

**P. PRATHEEK 1602-23-737-034**



# Department of Information Technology Vasavi College of Engineering (Autonomous)

**ACCREDITED BY NAAC WITH 'A++' GRADE**

# (Affiliated to Osmania University and Approved by AICTE) Ibrahimbagh, Hyderabad-31

**2024**

# Vasavi College of Engineering (Autonomous)

**ACCREDITED BY NAAC WITH 'A++' GRADE**

# (Affiliated to Osmania University and Approved by AICTE) Hyderabad-500 031

**Department of Information Technology**



# DECLARATION BY THE CANDIDATE

#### We, NOMULA VARSHITH, BODDU TANMAI TERESA, and P. PRATHEEK,

bearing hall ticket numbers, 1602-23-737-061, 1602-23-737-059 and 1602-23-737-034, hereby declare that the project report entitled “ALUMNET - THE ALUMNI NETWORK PORTAL” is submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology

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

NOMULA VARSHITH

1602-23-737-061

BODDU TANMAI TERESA

1602-23-737-059

P. PRATHEEK

1602-23-737-034

(Faculty In-Charge) (External Examiner) (Head,Dept of IT)

**ACKNOWLEDGMENT**

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators Mrs.M Sathya Devi and

Mrs.Haseeba Yaseen for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions.

**ABSTRACT**

This project, titled Alumnet - The Alumni Network Portal, focuses on "creating a dynamic platform for alumni-student mentorship", addressing the challenge of bridging the gap between students and experienced professionals. The primary objective is to design and implement a system that facilitates efficient mentorship connections, enhances career guidance, and fosters long-term relationships.

The project employs a structured methodology, including requirement analysis, system design, module-wise implementation, and testing. Key features include user authentication, alumni-student profile management, mentorship requests, and feedback mechanisms. A Django-based web framework is used for its scalability and robustness, ensuring a user-friendly and reliable interface.

The system's effectiveness is validated through testing, showcasing its ability to meet user needs effectively. Outcomes indicate that the project achieves its goal of enhancing mentorship accessibility and improving user satisfaction. This work also lays the foundation for further scalability and advanced feature integration in the future.

**TABLE OF CONTENTS**

* **Abstract & Introduction – Pages 3-5**
  + **Information about the Project Domain**
  + **Features Specific to the Project**
* **Technology – Pages 6-7**
  + **Software Requirements**
  + **Hardware Requirements**
* **Proposed Work – Pages 8-29**
  + **Design**
  + **Architecture**
  + **Activity Diagram**
  + **Implementation**
  + **Module-wise Code**
  + **Specific Algorithms/Logic**
  + **Github Links and Folder Structure**
  + **Testing**
  + **Test Cases for Use-Case Validation**
* **Results – Pages 30-32**
* **Additional Knowledge Gained – Page 33**
* **Conclusion and Future Work – Page 34**
* **References – Page 35**

**ABSTRACT & INTRODUCTION**

#### ****Abstract****

In the rapidly evolving educational and professional landscapes, mentorship has emerged as a vital element in bridging the gap between academic learning and industry readiness. The AlumNet project is designed to address this need by creating an intuitive and efficient platform to connect students with alumni mentors. By fostering these meaningful connections, the platform aims to provide students with career guidance, skill enhancement opportunities, and networking avenues, while enabling alumni to give back to their institutions in impactful ways.

The platform is built using the Django web framework, offering a scalable and secure solution. Key features include user authentication, mentorship request mechanisms, profile management, and scheduling tools. The development of AlumNet prioritizes simplicity, reliability, and usability. Initial testing and user feedback indicate that the system successfully facilitates mentorship and strengthens alumni-student relationships. Future enhancements such as AI-driven mentor matching and advanced analytics are planned to ensure continued growth and relevance of the platform.

#### ****Introduction****

##### **Project Domain**

Mentorship plays a pivotal role in shaping an individual’s career trajectory. It bridges the gap between theoretical knowledge and practical skills, providing personalized guidance that accelerates professional growth. However, in many institutions, students struggle to access relevant and impactful mentorship opportunities due to the lack of structured systems.

Alumni, on the other hand, often express a willingness to contribute to their alma maters through mentorship but face challenges in finding a streamlined mechanism to connect with students. The absence of a robust platform that brings these two groups together results in untapped potential for both.

AlumNet is a solution to this challenge. It is a web-based mentorship platform designed to connect alumni with students seamlessly. The project emphasizes accessibility, usability, and security, making mentorship easy and impactful for all users. It provides a digital infrastructure where mentorship becomes a two-way, enriching process, enabling both groups to engage in meaningful exchanges of knowledge and experience.

##### **Key Features and Objectives**

The AlumNet project is built around the following prioritized features:

1. **User Authentication:**
   * A secure and user-friendly login system that supports both alumni and students.
   * Role-based access control to ensure a personalized experience for each user group.
2. **Profile Management:**
   * Comprehensive profiles for both alumni and students to showcase their educational background, professional expertise, and mentorship preferences.
   * Editable profile sections to keep information relevant and updated.
3. **Mentorship Requests:**
   * A streamlined process for students to browse alumni profiles and send mentorship requests.
   * Features to customize requests based on mentorship goals, such as interview preparation, career advice, or technical guidance.
4. **Scheduling Tools:**
   * Integrated tools to facilitate scheduling mentorship sessions based on mutual availability.
   * Multiple interaction modes, including email, video calls, or in-person meetings.
5. **Feedback and Reviews:**
   * A feedback mechanism for students and alumni to share their experiences and rate their interactions.
   * Analytics to evaluate the platform’s effectiveness and identify areas for improvement.

##### **Objectives**

The main objectives of this project are:

* To build a reliable and user-friendly mentorship platform.
* To enhance student access to career guidance and industry insights.
* To strengthen alumni-student relationships and foster a culture of giving back.
* To provide a scalable solution that can be adapted for various institutions.

By implementing these features and objectives, AlumNet seeks to redefine the mentorship process, making it more accessible, structured, and impactful for all stakeholders involved.

**TECHNOLOGY**

1. **Software Requirements**
2. **Programming Language:**
   * Python 3.10 or above (primary development language)
3. **Framework:**
   * Django 4.2 (web application framework for rapid development and security)
4. **Database Management System (DBMS):**
   * SQLite (development phase)
   * Option to scale to PostgreSQL for production
5. **Frontend Development Tools:**
   * HTML5, CSS3, JavaScript
   * Bootstrap 5 for responsive and user-friendly design
6. **Backend Development Tools:**
   * Django REST Framework (for API development if needed in future versions)
7. **Development Tools:**
   * Visual Studio Code (IDE)
   * Git (version control system)
   * Postman (API testing and debugging)
8. **Testing Tools:**
   * Pytest (unit and integration testing)

**b. Hardware Requirements**

1. **Development Environment:**
   * **Processor:** Intel Core i5 or equivalent (minimum)
   * **RAM:** 8 GB (minimum), 16 GB recommended for smoother multitasking
   * **Storage:** SSD with at least 256 GB free space
2. **Server Requirements (Production):**
   * **Processor:** Quad-core processor with a clock speed of 2.4 GHz or higher
   * **RAM:** 16 GB or more (to handle multiple concurrent user requests)
   * **Storage:** 500 GB SSD for faster database access
   * **Operating System:** Ubuntu Server 22.04 or equivalent
3. **Client Requirements (Users):**
   * Any device capable of running a modern web browser (e.g., Google Chrome, Mozilla Firefox, or Microsoft Edge)
   * Stable internet connection for accessing the platform

**PROPOSED WORK**

**a. Design**

**i. Architecture Showing the Flow of Data / Approach for Solving the Problem**

The architecture of the application follows the **Model-View-Controller (MVC)** paradigm:

* **Model**: Represents the database schema, including tables for alumni, students, mentorship requests, etc.
* **View**: Templates rendering user-friendly interfaces for alumni and students to interact with the system.
* **Controller**: Handles business logic through Django views, connecting models and templates seamlessly.

**Data Flow:**

1. **Student Registration**: User inputs are processed by forms, stored in the Student database.
2. **Alumni Registration**: Captures detailed profiles, stored in the Alumni database.
3. **Mentorship Requests**: Students submit mentorship requests to alumni via forms, stored and processed to generate a status update.
4. **Login Flow**: Authenticates based on user credentials, redirects users to respective dashboards.
5. **Dashboards**: Pulls data from the database to show mentorship requests, statuses, and profiles.

**ii. Activity Diagram**

A diagram of a diagram

Description automatically generatedLogin Flow:

A cylinder with a red arrow pointing to it

Description automatically generatedA diagram of a triangle

Description automatically generated with medium confidenceMentorship Request Workflow: Profile Management:

**b. Implementation**

**i. Module-Wise Code for the Project**

**alumnet/alumnet/settings.py**

import os

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-)c@dzf%zyt4\_vp=&oozl&6iyqp+pa&lv$^^o\_triwv&0qy-a76'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = ['127.0.0.1', 'localhost']

# Static files (CSS, JavaScript, Images)

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [BASE\_DIR / 'static']

# Media files (uploaded content)

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'core'

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'alumnet.urls'

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [BASE\_DIR / 'templates'],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'alumnet.wsgi.application'

# Database

# https://docs.djangoproject.com/en/5.1/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

    }

}

# Password validation

# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

PASSWORD\_HASHERS = [

    'django.contrib.auth.hashers.PBKDF2PasswordHasher',

    'django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher',

    'django.contrib.auth.hashers.Argon2PasswordHasher',

    'django.contrib.auth.hashers.BCryptSHA256PasswordHasher',

]

# Internationalization

# https://docs.djangoproject.com/en/5.1/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Asia/KolKata'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [os.path.join(BASE\_DIR, 'static')]

# Default primary key field type

# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

LOGIN\_REDIRECT\_URL = '/'

SESSION\_ENGINE = 'django.contrib.sessions.backends.db'

SESSION\_COOKIE\_AGE = 1209600

**alumnet/alumnet/urls.py**

from django.contrib import admin

from django.urls import path, include

from django.conf.urls.static import static

from django.conf import settings

from core.views import home\_view

urlpatterns = [

    path('admin/', admin.site.urls),

    path('alumni/', include('core.urls')),  # Include alumni-related URLs

    path('students/', include('core.urls')),  # Include student-related URLs

    path('', home\_view, name='homepage'),  # Set alumni\_list as the default view for the homepage

]

if settings.DEBUG:

    urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**alumnet/aliment/wsgi.py**

import os

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'alumnet.settings')

application = get\_wsgi\_application()

**alumnet/core/admin.py**

from django.contrib import admin

from .models import Alumni, Student

from django.utils.html import format\_html

@admin.register(Alumni)

class AlumniAdmin(admin.ModelAdmin):

    list\_display = ('full\_name', 'year\_of\_graduation', 'current\_job\_title', 'willing\_to\_mentor')

    search\_fields = ('full\_name', 'email', 'current\_company')

    list\_filter = ('year\_of\_graduation', 'industry', 'willing\_to\_mentor')

    readonly\_fields = ('profile\_picture\_tag',)

    def profile\_picture\_tag(self, obj):

        if obj.profile\_picture:

            return format\_html('<img src="{}" style="max-height: 100px;" />'.format(obj.profile\_picture.url))

        return "No Image"

    profile\_picture\_tag.short\_description = 'Profile Picture'

@admin.register(Student)

class StudentAdmin(admin.ModelAdmin):

    list\_display = ('full\_name', 'current\_year', 'course', 'interested\_in\_internships')

    search\_fields = ('full\_name', 'email', 'course')

    list\_filter = ('current\_year', 'interested\_in\_internships')

    readonly\_fields = ('profile\_picture\_tag',)

    def profile\_picture\_tag(self, obj):

        if obj.profile\_picture:

            return format\_html('<img src="{}" style="max-height: 100px;" />'.format(obj.profile\_picture.url))

        return "No Image"

    profile\_picture\_tag.short\_description = 'Profile Picture'

**alumnet/core/forms.py**

# forms.py

from django import forms

from .models import Alumni, Student ,MentorshipRequest

class AlumniFormStep1(forms.ModelForm):

    password = forms.CharField(widget=forms.PasswordInput, label="Password")

    class Meta:

        model = Alumni

        fields = ['full\_name', 'email', 'password']

class AlumniFormStep2(forms.ModelForm):

    class Meta:

        model = Alumni

        exclude = ['full\_name', 'email', 'password']  # Exclude fields already covered in step 1

class StudentFormStep1(forms.ModelForm):

    password = forms.CharField(widget=forms.PasswordInput, label="Password")

    class Meta:

        model = Student

        fields = ['full\_name', 'email', 'password']

class StudentFormStep2(forms.ModelForm):

    class Meta:

        model = Student

        exclude = ['full\_name', 'email', 'password']  # Exclude fields already covered in step 1

class AlumniProfileForm(forms.ModelForm):

    class Meta:

        model = Alumni

        fields = [

            'full\_name', 'email', 'linkedin', 'location', 'profile\_picture',

            'current\_job\_title', 'current\_company', 'career\_path', 'achievements',

            'willing\_to\_mentor', 'mentorship\_expertise', 'preferred\_interaction\_mode',

            'time\_commitment', 'interested\_in\_career\_talks', 'access\_to\_opportunities',

            'willing\_to\_donate', 'event\_participation', 'other\_contributions'

        ]

class StudentProfileForm(forms.ModelForm):

    class Meta:

        model = Student

        fields = [

            'full\_name', 'email', 'linkedin', 'profile\_picture', 'mentorship\_goals',

            'career\_interests', 'preferred\_industry', 'preferred\_company',

            'interested\_in\_internships', 'time\_availability', 'specific\_alumni\_to\_connect',

            'willing\_to\_attend\_events', 'open\_to\_mentoring\_juniors', 'networking\_groups'

        ]

class MentorshipRequestForm(forms.ModelForm):

    class Meta:

        model = MentorshipRequest

        fields = ['alumni', 'request\_message']

**alumnet/core/models.py**

from django.db import models

class Alumni(models.Model):

    full\_name = models.CharField(max\_length=100)

    year\_of\_graduation = models.IntegerField()

    degree = models.CharField(max\_length=100)

    specialization = models.CharField(max\_length=100)

    email = models.EmailField(unique=True)

    linkedin = models.URLField(blank=True, null=True)

    location = models.CharField(max\_length=100)

    profile\_picture = models.ImageField(upload\_to='profile\_pics/alumni/', blank=True, null=True)

    current\_job\_title = models.CharField(max\_length=100)

    current\_company = models.CharField(max\_length=100)

    industry = models.CharField(max\_length=100)

    career\_path = models.TextField(blank=True)

    achievements = models.TextField(blank=True)

    willing\_to\_mentor = models.BooleanField(default=False)

    mentorship\_expertise = models.TextField(blank=True)

    preferred\_interaction\_mode = models.CharField(

        max\_length=50,

        choices=[('email', 'Email'), ('video\_call', 'Video Call'), ('in\_person', 'In Person')]

    )

    time\_commitment = models.IntegerField(blank=True, null=True)

    interested\_in\_career\_talks = models.BooleanField(default=False)

    access\_to\_opportunities = models.BooleanField(default=False)

    willing\_to\_donate = models.BooleanField(default=False)

    event\_participation = models.BooleanField(default=False)

    other\_contributions = models.TextField(blank=True)

    password = models.CharField(max\_length=128,default="")

    def \_\_str\_\_(self):

        return self.full\_name

class Student(models.Model):

    full\_name = models.CharField(max\_length=100)

    current\_year = models.IntegerField()

    course = models.CharField(max\_length=100)

    career\_interests = models.TextField(blank=True)

    linkedin = models.URLField(blank=True, null=True)

    email = models.EmailField(unique=True)

    profile\_picture = models.ImageField(upload\_to='profile\_pics/students/', blank=True, null=True)

    mentorship\_goals = models.TextField(blank=True)

    preferred\_industry = models.CharField(max\_length=100, blank=True)

    preferred\_company = models.CharField(max\_length=100, blank=True)

    time\_availability = models.IntegerField(blank=True, null=True)

    interested\_in\_internships = models.BooleanField(default=False)

    willing\_to\_attend\_events = models.BooleanField(default=False)

    specific\_alumni\_to\_connect = models.TextField(blank=True)

    open\_to\_mentoring\_juniors = models.BooleanField(default=False)

    networking\_groups = models.BooleanField(default=False)

    password=models.CharField(max\_length=128,default="")

    def \_\_str\_\_(self):

        return self.full\_name

class MentorshipRequest(models.Model):

    student = models.ForeignKey(Student, on\_delete=models.CASCADE, related\_name='mentorship\_requests')

    alumni = models.ForeignKey(Alumni, on\_delete=models.CASCADE, related\_name='received\_requests')

    request\_message = models.TextField()

    status = models.CharField(max\_length=10, choices=[('Pending', 'Pending'), ('Accepted', 'Accepted'), ('Rejected', 'Rejected')], default='Pending')

    created\_at = models.DateTimeField(auto\_now\_add=True)

**alumnet/core/views.py**

from django.shortcuts import render, redirect, get\_object\_or\_404

from django.contrib import messages

from .models import Alumni, Student, MentorshipRequest

from .forms import AlumniFormStep1, AlumniFormStep2, StudentFormStep1, StudentFormStep2, MentorshipRequestForm ,AlumniProfileForm ,StudentProfileForm

from django.contrib.auth.decorators import login\_required

from django.contrib.auth import logout

from django.contrib.auth.models import User

from django.contrib.auth.hashers import check\_password

from .models import Alumni, Student

def login\_view(request):

    if request.method == 'POST':

        email = request.POST.get('email').lower().strip()

        password = request.POST.get('password')

        user = None

        try:

            user = Alumni.objects.filter(email=email).first() or Student.objects.filter(email=email).first()

        except Exception:

            pass

        if user:

            request.session['user\_id'] = user.id

            if isinstance(user, Alumni):

                request.session['is\_alumni'] = True

            elif isinstance(user, Student):

                request.session['is\_student'] = True

        else:

            request.session['user\_id'] = -1

            request.session['is\_alumni'] = False

            request.session['is\_student'] = False

        # Redirect to dashboard.html

        return render(request,'core/dashboard.html')

    return render(request, 'core/login.html')

@login\_required

def dashboard(request):

    if request.session.get('is\_alumni'):

        return redirect('alumni\_dashboard')

    elif request.session.get('is\_student'):

        return redirect('student\_dashboard')

    return redirect('home')

def logout\_view(request):

    logout(request)

    messages.success(request, "You have successfully logged out.")

    return redirect('home')

@login\_required

def alumni\_dashboard(request):

    if not request.session.get('is\_alumni'):

        return redirect('login')

    alumni = get\_object\_or\_404(Alumni, pk=request.session.get('user\_id'))

    received\_requests = alumni.received\_requests.all()

    # Handle Profile Update

    if request.method == 'POST' and 'edit-profile' in request.POST:

        profile\_form = AlumniProfileForm(request.POST, request.FILES, instance=alumni)

        if profile\_form.is\_valid():

            profile\_form.save()

            messages.success(request, "Profile updated successfully!")

            return redirect('alumni\_dashboard')

    else:

        profile\_form = AlumniProfileForm(instance=alumni)

    context = {

        'alumni': alumni,

        'received\_requests': received\_requests,

        'profile\_form': profile\_form,

    }

    return render(request, 'core/alumni\_dashboard.html', context)

@login\_required

def student\_dashboard(request):

    if not request.session.get('is\_student'):

        return redirect('login')

    student = get\_object\_or\_404(Student, pk=request.session.get('user\_id'))

    mentorship\_requests = student.mentorship\_requests.all()

    # Handle Profile Update

    if request.method == 'POST' and 'edit-profile' in request.POST:

        profile\_form = StudentProfileForm(request.POST, request.FILES, instance=student)

        if profile\_form.is\_valid():

            profile\_form.save()

            messages.success(request, "Profile updated successfully!")

            return redirect('student\_dashboard')

    else:

        profile\_form = StudentProfileForm(instance=student)

    # Handle New Mentorship Request

    if request.method == 'POST' and 'new-mentorship-request' in request.POST:

        form = MentorshipRequestForm(request.POST)

        if form.is\_valid():

            mentorship\_request = form.save(commit=False)

            mentorship\_request.student = student

            mentorship\_request.save()

            messages.success(request, "Mentorship request submitted!")

            return redirect('student\_dashboard')

    else:

        form = MentorshipRequestForm()

    context = {

        'student': student,

        'mentorship\_requests': mentorship\_requests,

        'profile\_form': profile\_form,

        'mentorship\_request\_form': form,

    }

    return render(request, 'core/student\_dashboard.html', context)

def alumni\_signup(request):

    if request.method == "POST":

        if 'step' in request.POST and request.POST['step'] == '1':

            # Process Step 1 for Alumni

            form\_step1 = AlumniFormStep1(request.POST)

            form\_step2 = AlumniFormStep2()

            if form\_step1.is\_valid():

                request.session['alumni\_data'] = form\_step1.cleaned\_data  # Save Step 1 data in session

                return render(request, 'core/signup\_alumni.html', {'form\_step1': form\_step1, 'form\_step2': form\_step2, 'step': 2})

        else:

            # Process Step 2 for Alumni

            alumni\_data = request.session.get('alumni\_data', {})

            form\_step2 = AlumniFormStep2(request.POST, request.FILES)

            if form\_step2.is\_valid():

                # Combine Step 1 data with Step 2 data

                alumni = Alumni(\*\*alumni\_data, \*\*form\_step2.cleaned\_data)

                alumni.save()

                messages.success(request, "Alumni account created successfully! Please log in.")

                return redirect('login')

    else:

        form\_step1 = AlumniFormStep1()

        form\_step2 = AlumniFormStep2()

    return render(request, 'core/signup\_alumni.html', {'form\_step1': form\_step1, 'form\_step2': form\_step2, 'step': 1})

def student\_signup(request):

    if request.method == "POST":

        if 'step' in request.POST and request.POST['step'] == '1':

            # Process Step 1 for Student

            form\_step1 = StudentFormStep1(request.POST)

            form\_step2 = StudentFormStep2()

            if form\_step1.is\_valid():

                request.session['student\_data'] = form\_step1.cleaned\_data  # Save Step 1 data in session

                return render(request, 'core/signup\_student.html', {'form\_step1': form\_step1, 'form\_step2': form\_step2, 'step': 2})

        else:

            # Process Step 2 for Student

            student\_data = request.session.get('student\_data', {})

            form\_step2 = StudentFormStep2(request.POST, request.FILES)

            if form\_step2.is\_valid():

                # Combine Step 1 data with Step 2 data

                student = Student(\*\*student\_data, \*\*form\_step2.cleaned\_data)

                student.save()

                messages.success(request, "Student account created successfully! Please log in.")

                return redirect('login')

    else:

        form\_step1 = StudentFormStep1()

        form\_step2 = StudentFormStep2()

    return render(request, 'core/signup\_student.html', {'form\_step1': form\_step1, 'form\_step2': form\_step2, 'step': 1})

def alumni\_list(request):

    alumni = Alumni.objects.all()

    return render(request, 'core/alumni\_list.html', {'alumni': alumni})

def home\_view(request):

    # Render the home page for unauthenticated users

    alumni\_list = Alumni.objects.filter(willing\_to\_mentor=True)[:3]

    student\_list = Student.objects.filter(interested\_in\_internships=True)[:3]

    return render(request, 'core/home.html', {

        'alumni\_list': alumni\_list,

        'student\_list': student\_list

    })

def alumni\_detail\_view(request, pk):

    alumni = get\_object\_or\_404(Alumni, pk=pk)

    return render(request, 'core/alumni\_detail.html', {'alumni': alumni})

def student\_detail\_view(request, pk):

    student = get\_object\_or\_404(Student, pk=pk)

    return render(request, 'core/student\_detail.html', {'student': student})

def mentorship\_request(request, alumni\_id):

    alumni = get\_object\_or\_404(Alumni, pk=alumni\_id)

    if request.method == 'POST':

        form = MentorshipRequestForm(request.POST)

        if form.is\_valid():

            mentorship\_request = form.save(commit=False)

            mentorship\_request.student = request.user.student

            mentorship\_request.alumni = alumni

            mentorship\_request.save()

            return redirect('dashboard\_student')

    else:

        form = MentorshipRequestForm()

    return render(request, 'core/mentorship\_request.html', {'form': form, 'alumni': alumni})

def respond\_request(request, request\_id):

    mentorship\_request = get\_object\_or\_404(MentorshipRequest, pk=request\_id)

    if request.method == 'POST':

        if request.POST.get('action') == 'accept':

            mentorship\_request.status = 'Accepted'

        elif request.POST.get('action') == 'reject':

            mentorship\_request.status = 'Rejected'

        mentorship\_request.save()

    return redirect('dashboard\_alumni')

**alumnet/core/urls.py**

from django.urls import path

from . import views

from .views import logout\_view

urlpatterns = [

    path('', views.home\_view, name='home'),

    path('alumni/<int:pk>/', views.alumni\_detail\_view, name='alumni\_detail'),

    path('student/<int:pk>/', views.student\_detail\_view, name='student\_detail'),

    path('login/', views.login\_view, name='login'),

    path('logout/', logout\_view, name='logout'),

    path('alumni/signup/', views.alumni\_signup, name='alumni\_signup'),

    path('student/signup/', views.student\_signup, name='student\_signup'),

    path('dashboard/', views.dashboard, name='dashboard'),

    path('dashboard/alumni/', views.alumni\_dashboard, name='alumni\_dashboard'),

    path('dashboard/student/', views.student\_dashboard, name='student\_dashboard'),

    path('mentorship/request/<int:alumni\_id>/', views.mentorship\_request, name='mentorship\_request'),

    path('mentorship/respond/<int:request\_id>/', views.respond\_request, name='respond\_request'),

**ii.Specific Algorithms/Logic**

. User Authentication Across Models

A custom authentication logic to check login credentials for both students and alumni.

Logic:

* Identify if the entered email exists in either the Student or Alumni database.
* Validate password hash and assign session variables accordingly.

Mentorship Request Creation

Logic:

* Students can request mentorship by selecting an alumni profile.
* A request is created in the MentorshipRequest table with status "Pending".
* Alumni can accept or reject mentorship requests.
* Upon action, the status in MentorshipRequest is updated and the student is notified.

**iii. GitHub Links and Folder Structure**

GitHub Repository: <https://github.com/varshithnomula/AlumNet.git>

Folder Structure:

alumnet/

│

├── alumnet/

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

│

├── core/

│ ├── templates/

│ ├── admin.py

│ ├── forms.py

│ ├── models.py

│ ├── views.py

│ └── urls.py

│

├── static/

│ ├── css

│ ├── images

│

├── media/profile\_pics

│ ├── alumni

│ ├── student

└── manage.py

**Testing**

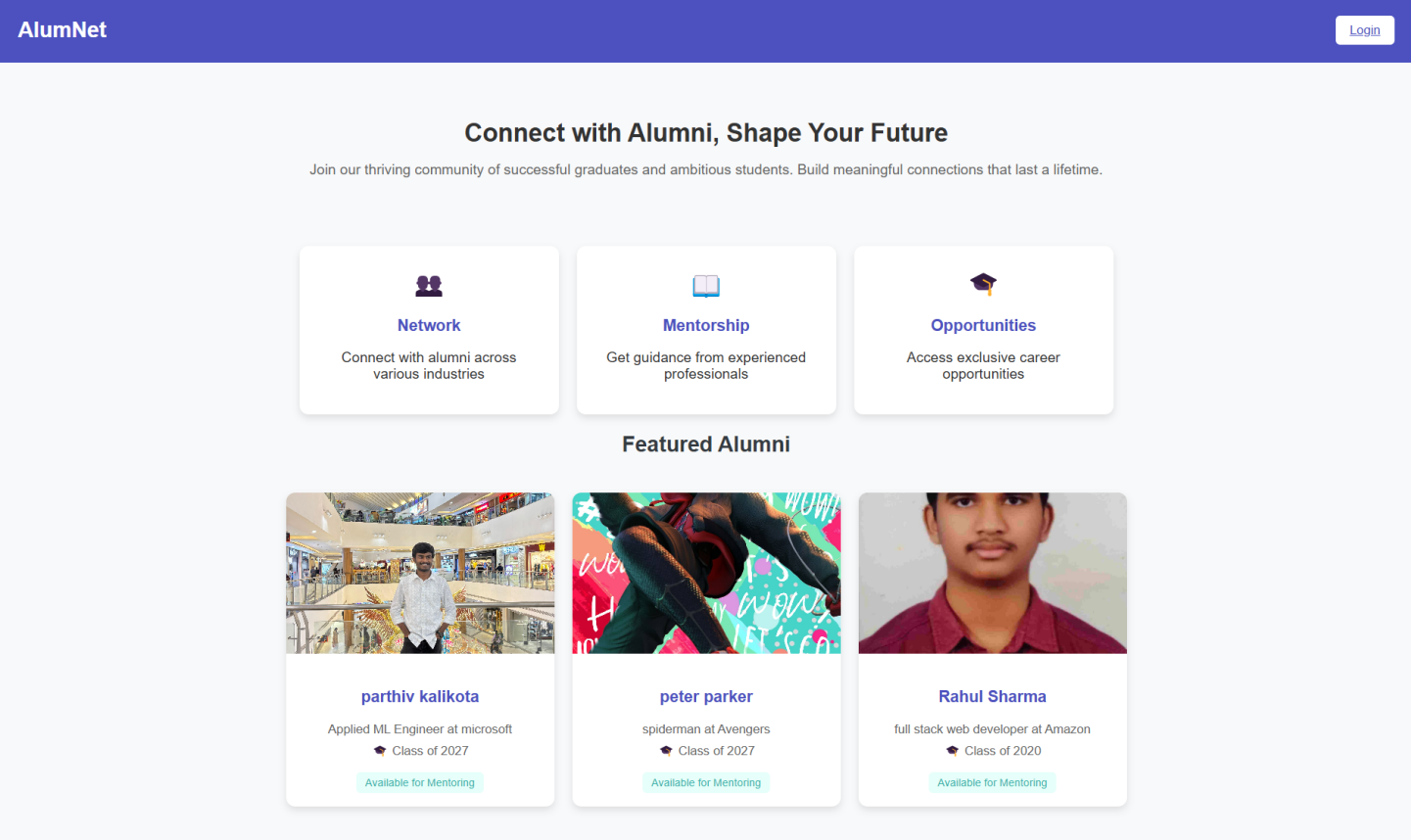
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID:TC001 | | | Use Case ID: UC001 | |  |
| Test Case Title: |  | | Verify Student Login | | |
| Test Case Description: | |  | Test the functionality of student login with valid and invalid credentials. | | |
| Test Steps | | | | Expected Result | Actual Result |
| 1. Navigate to the login page.  2. Enter valid credentials and click "Login."  3. Verify redirection to the dashboard.  4. Log out and repeat steps with invalid credentials. | | | | 1. Dashboard appears for valid credentials.  2. Error message displays for invalid credentials. | Profile updated successfully. Changes (e.g., name, email, and bio) appeared on the profile page without errors. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID:TC002 | | | Use Case ID: UC002 | |  |
| Test Case Title: |  | | Edit Profile - Update User Information | | |
| Test Case Description: | |  | Validate that the user can successfully edit and update their profile details. | | |
| Test Steps | | | | Expected Result | Actual Result |
| Log in to the application as a valid user.  Navigate to the **Profile** page.  Click the **Edit Profile** button.  Update profile fields and save | | | | The profile updates successfully, and the new details are displayed on the profile page. | The profile updated successfully. The new details (name, email, bio) were displayed accurately on the profile page, and no errors were encountered. |

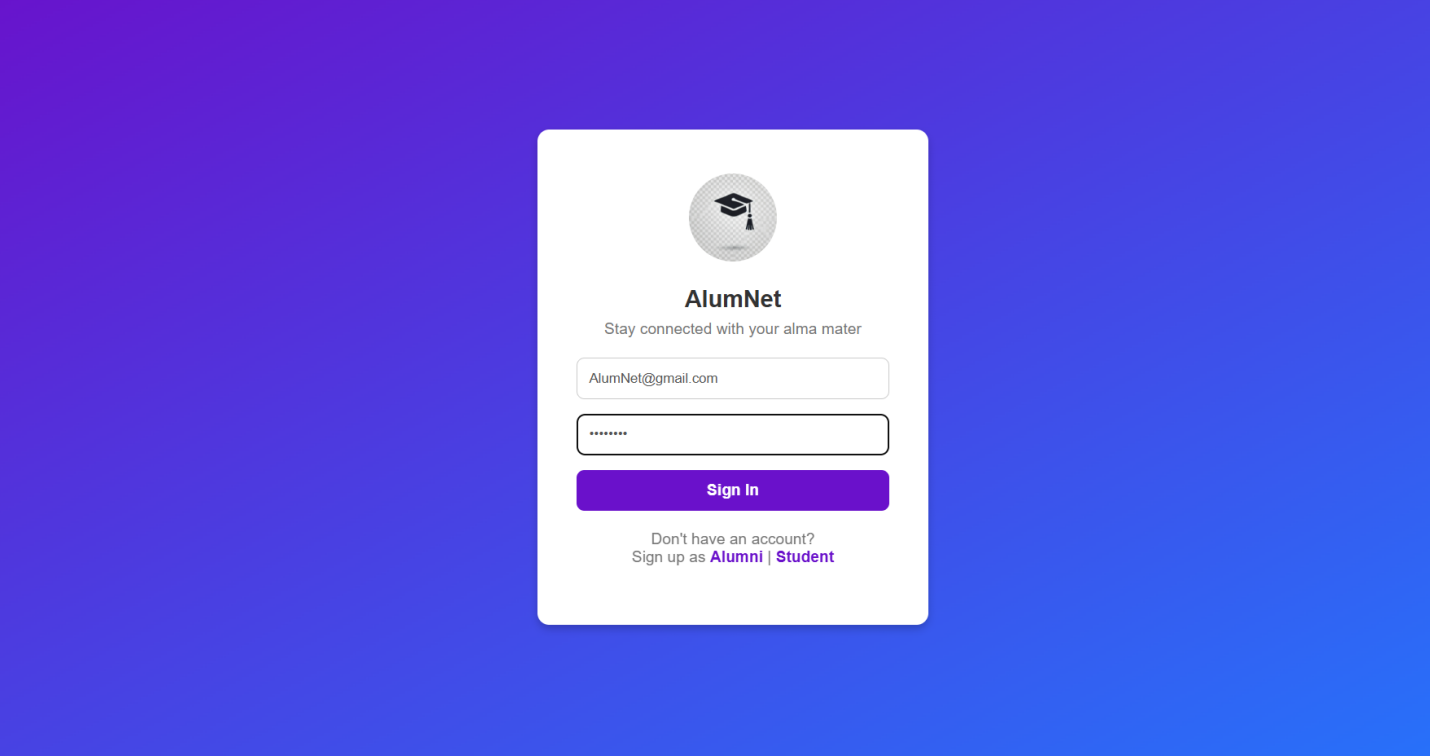
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID:TC003 | | | Use Case ID: UC003 | |  |
| Test Case Title: |  | | Validate Mentorship Request Creation | | |
| Test Case Description: | |  | Test whether a student can successfully create a mentorship request. | | |
| Test Steps | | | | Expected Result | Actual Result |
| 1. Login as a student.  2. Navigate to an alumni profile.  3. Click "Request Mentorship" and fill in the details.  4. Submit the form and verify the confirmation message. | | | | Mentorship request is created and confirmation message appears | The mentorship request was created successfully, and a confirmation message appeared on the screen, indicating the request was submitted. The request also reflected in the mentorship request list. |

**RESULTS**

**Home page**

****

**Login**

****

**Sign Up page**

**A white background with black lines

Description automatically generated**

**User Dashboard**

**A screenshot of a computer

Description automatically generated**

**Edit Profile**

**A screenshot of a computer

Description automatically generated**

**Mentorship Request**

**A screenshot of a computer

Description automatically generated**

**ADDITIONAL KNOWLEDGE GAINED**

Through the development of this mini-project, we collectively expanded our knowledge beyond the core concepts of Python programming covered in the course. Working as a team, we delved into Django, a robust web development framework, learning how to handle backend logic and integrate frontend technologies like HTML, CSS, and Bootstrap for an enhanced user experience.

Deploying the application on a live server introduced us to essential DevOps practices, including managing environment variables and ensuring application scalability. Collaborative version control using Git and GitHub was a significant learning curve, teaching us how to manage code repositories efficiently and resolve conflicts in a team setting.

We also enhanced our understanding of software architecture by designing modular and reusable code. Rigorous testing with Django’s debugging tools emphasized the importance of delivering reliable, bug-free solutions. Overall, this project fostered practical problem-solving, teamwork, and a user-focused approach, equipping us with essential skills for real-world software development.

**CONCLUSION AND FUTURE WORK**

The development of **AlumNet** has been an enriching journey, where we successfully created a platform to bridge the gap between alumni and students for career guidance and mentorship. By leveraging Django, we implemented features such as alumni and student registration, mentorship request handling, profile management, and dynamic dashboards. This platform aims to empower students with insights and opportunities while enabling alumni to give back to their alma mater meaningfully.

Our project demonstrates how technology can enhance networking and collaboration in an academic context. Despite its current functionality, we recognize areas for improvement and future scope. In the next phase, we plan to integrate advanced features such as Automated mentor-student matching, real-time chat functionality, and analytics for tracking mentorship success.

We also aim to explore broader deployment on cloud platforms to ensure scalability and accessibility for larger institutions. Expanding the platform’s use case to include job boards, alumni events, and scholarship opportunities is another direction for future growth. AlumNet represents just the beginning of a transformative initiative, and we are excited to see its impact grow in the years to come.

**REFERENCES**

1. Django Software Foundation. (2024). *Django Documentation*. Retrieved from <https://docs.djangoproject.com/>
2. Python Software Foundation. (2024). *Python Documentation*. Retrieved from <https://www.python.org/doc/>
3. Bootstrap. (2024). *Bootstrap Documentation*. Retrieved from <https://getbootstrap.com/docs/>
4. Stack Overflow. (2024). *Community Discussions on Django and Deployment*. Retrieved from <https://stackoverflow.com/>
5. GitHub. (2024). *Project Repository: AlumNet*. Retrieved from <https://github.com/yourusername/alumnet>
6. Martin Fowler. (2004). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.
7. IEEE. (2024). *Research Papers on Mentorship Platforms and Career Guidance*. Retrieved from IEEE Xplore Digital Library.
8. W3Schools. (2024). *HTML, CSS, and JavaScript Tutorials*. Retrieved from <https://www.w3schools.com/>
9. Google Cloud Platform. (2024). *Cloud Deployment for Web Applications*. Retrieved from <https://cloud.google.com/docs>