**Alumni Engagement Web Application Report**

**A Project by Group I/Eve**

|  |  |
| --- | --- |
| **Name** | **Registration Number** |
| Kisomose Arnold Patrick | 23/U/10079/EVE |
| Mugisha Modern | 23/U/11587/PS |
| Kabalebe Joshua | 23/U/27067/EVE |
| Kukiriza Sinai | 23/U/0646 |
| Wandera Hannington | 23/U/23796 |

Github Link: <https://github.com/syny-pulse/alumni-engage.git>

**Table of Contents**

[1. Introduction 1](#_Toc203676504)

[2. Project Overview 1](#_Toc203676505)

[2.1. Objectives 1](#_Toc203676506)

[2.2. Technology Stack 1](#_Toc203676507)

[3. Requirements Specification 2](#_Toc203676508)

[3.1. Functional Requirements 2](#_Toc203676509)

[3.2. Non-Functional Requirements 4](#_Toc203676510)

[4. System Architecture 5](#_Toc203676511)

[4.1. Directory Structure Overview 5](#_Toc203676512)

[5. Database Design 6](#_Toc203676513)

[6. Screenshots 8](#_Toc203676514)

[7. Source Code 9](#_Toc203676515)

[8. Challenges and solutions 9](#_Toc203676516)

[9. Conclusion 10](#_Toc203676517)

**List of Figures**

[Figure 3. 1 Project Directory Structure 6](#_Toc203676524)

[Figure 6. 1 Landing Page 8](#_Toc203676538)

# Introduction

The Alumni Engagement Web Application is a Flask-based platform designed to foster connections among alumni, facilitate event management, and provide a job and opportunity board. Built with Flask, Flask-SQLAlchemy, Bootstrap, and MySQL (with PostgreSQL planned for production), the application supports alumni networking through features like user profiles, event RSVPs, a searchable alumni directory, and a job board. This report outlines the application's purpose, architecture, features, and implementation details, as required for the Web Development Projects (Flask) course.

The application addresses the need for a centralized platform where alumni can stay connected, share opportunities, and engage with their alma mater. It includes user management, news announcements, event coordination, and admin tools for content and user oversight, ensuring a robust and user-friendly experience.

# Project Overview

## Objectives

* Enable alumni to register, create profiles, and connect with peers.
* Provide tools for administrators to manage events, news, and job postings.
* Offer a searchable alumni directory and job board for networking and opportunities.
* Ensure a secure, responsive, and scalable platform adhering to modern web standards.

## Technology Stack

|  |  |
| --- | --- |
| **Component** | **Technology** |
| Framework | Flask (Python) |
| ORM | Flask-SQLAlchemy |
| Templates | Flask-Jinja, HTML |
| Database | MYSQL (development), PostgreSQL(production) |
| Mail | Flask-Mail |
| Version Control | Git, GitHub |
| Forms | WTForms |
| Auth | Flask-Login, Werkzeug |
| Styling | Custom CSS, Bootstrap, JavaScript |

# Requirements Specification

## Functional Requirements

These define what core services and operations the system must offer.

**User Management**

* FR1: The system shall allow alumni users to register and create a personal profile.
* FR2: The system shall allow users to log in and log out securely.
* FR3: The system shall provide an admin panel to manage users, events, and content.
* FR4: The system shall enable alumni to update their profiles with information such as graduation year, degree, current job title, location, and bio.
* FR5: The system shall support password reset via email.

**News and Announcements**

* FR6: The system shall allow admin users to create, publish, edit, and delete news posts or announcements.
* FR7: The system shall display a list of recent news or updates on the homepage.
* FR8: The system shall allow users to view detailed news articles.

**Event Management**

* FR9: The system shall allow admin users to create and manage alumni events with details like date, time, venue, description, and event image.
* FR10: The system shall allow users to view a list of upcoming and past events.
* FR11: The system shall allow users to RSVP for events and receive confirmation notifications.
* FR12: The system shall display event attendance summaries for admins.

**Alumni Directory**

* FR13: The system shall display a searchable directory of registered alumni.
* FR14: The system shall allow users to search alumni by name, graduation year, or program.
* FR15: The system shall restrict sensitive alumni data to logged-in users only.

**Messaging and Contact**

* FR16. The system shall provide a contact form for alumni inquiries or suggestions.
* FR17: The system shall notify the admin via email for each new inquiry.
* FR18: The system shall provide internal messaging for alumni to contact each other (future feature).

**Job and Opportunity Board**

* FR19: The system shall allow admin users to post job or internship opportunities.
* FR20: The system shall allow alumni to view and apply for listed opportunities.
* FR21: The system shall allow alumni to submit opportunities for admin approval.

**Feedback and Testimonials**

* FR22: The system shall allow alumni to submit testimonials or feedback about their experience.
* FR23: The system shall display approved testimonials on a public page.

**Admin Reporting and Analytics**

* FR24: The system shall generate reports on alumni registrations, event participation, and feedback submissions.
* FR25: The system shall display event attendance statistics and user engagement trends.

## Non-Functional Requirements

These define how the system should perform and its qualitative characteristics.

**Performance**

* NFR1: The system shall respond to user actions within 2 seconds for 90% of transactions.
* NFR2: The system shall process RSVP and contact form submissions in under 3 seconds.

**Security**

* NFR3: The system shall enforce HTTPS for all data transmissions.
* NFR4: The system shall hash and securely store user passwords (using Werkzeug or Flask extensions).
* NFR5: The system shall validate all user inputs to prevent SQL injection, XSS, and CSRF attacks.
* NFR6: The system shall implement role-based access control (Admin vs Alumni users).

**Usability**

* NFR7: The system shall provide a clean, responsive web interface compatible with desktops, tablets, and smartphones.
* NFR8: The system shall offer intuitive navigation with user-friendly forms and error messages.

**Availability**

* NFR9: The system shall maintain 99% uptime when deployed on production hosting.
* NFR10: The system shall log all system errors and notify admin of critical issues.

**Maintainability**

* NFR11: The system codebase shall follow Flask coding standards and PEP8 guidelines.
* NFR12: The system shall organize routes, views, models, and templates into a modular project structure for ease of updates.

**Scalability**

* NFR13: The system shall be designed to accommodate up to 500 concurrent users without significant performance drops.
* NFR14: The system shall allow for easy integration of new modules (e.g., video conferencing, discussion forums).

**Interoperability**

* NFR15: The system shall provide APIs or endpoints to support integration with future mobile apps or third-party alumni databases.

# System Architecture

The application is designed using the MVC (Model-View-Controller) pattern, modularized into separate packages for models, routes, templates, and utilities.

## Directory Structure Overview

* app.py: Main application entry point and configuration.
* config.py: Database, mail, and secret key settings.
* models/: Database models (e.g., User, Event, News).
* routes/: Route handlers for authentication, events, and more.
* templates/: HTML templates with Bootstrap styling.
* static/: CSS, JavaScript, and images.
* utils/: Helper functions, forms, and decorators.
* migrations/: Database migration files.

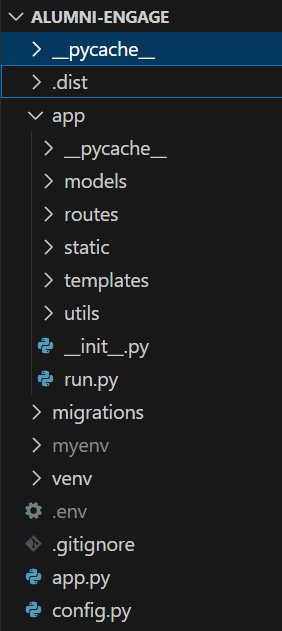


Figure 3. 1 Project Directory Structure

## Database Design

The application uses a relational database with key tables:

|  |  |
| --- | --- |
| **Table name** | **Attributes** |
| users: Stores alumni data | id, username, email, password\_hash, first\_name, last\_name, graduation\_year, degree, current\_job\_title, location, bio,  profile\_image, is\_admin, is\_active, created\_at, updated\_at ,is\_public |
| events: Manages event details | id, title, description, event\_date, venue, event\_image, max\_attendees, created\_by, is\_active, created\_at, updated\_at |
| rsvps: Tracks event attendance. | id, user\_id, event\_id, status, rsvp\_date, notes |
| jobs: Lists job opportunities. | id, title, company, description, requirements, location, job\_type, salary\_range, application\_url, contact\_email, posted\_by, is\_approved, is\_active, deadline, created\_at, updated\_at |
| news: Stores announcements with author and publication status. | id, title, content, summary, image\_url, author\_id, is\_published, created\_at, updated\_at |
| testimonials: Stores alumni feedback with content, rating, and approval status. | id, user\_id, title, content,rating, is\_approved, is\_featured, created\_at, updated\_at |
| contact\_submissions: Manages inquiries with name, email, subject, and admin response. | id, name, email, subject, message, user\_id, is\_read, admin\_response, responded\_at, created\_at |
| messages: Facilitates internal messaging with sender, recipient, and read status. | id, sender\_id, recipient\_id, subject, content, is\_read, sent\_at, read\_at |

# Screenshots

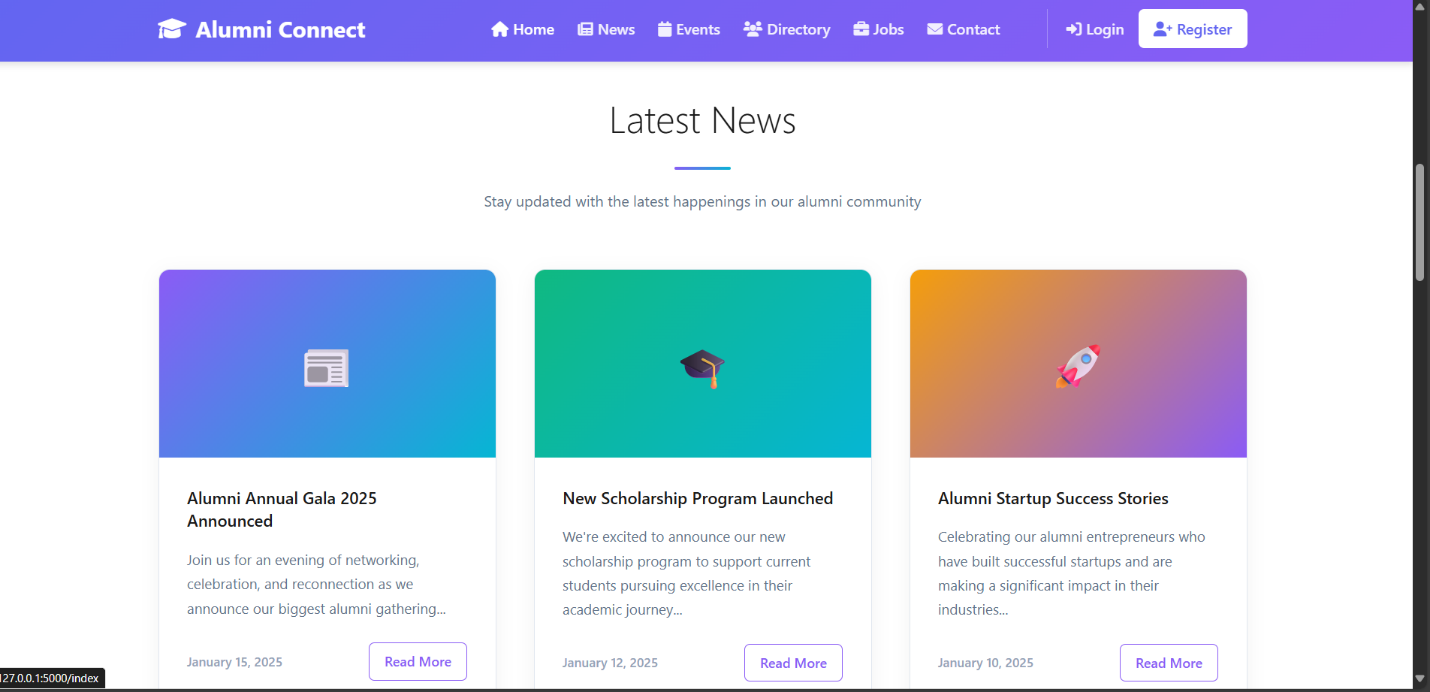
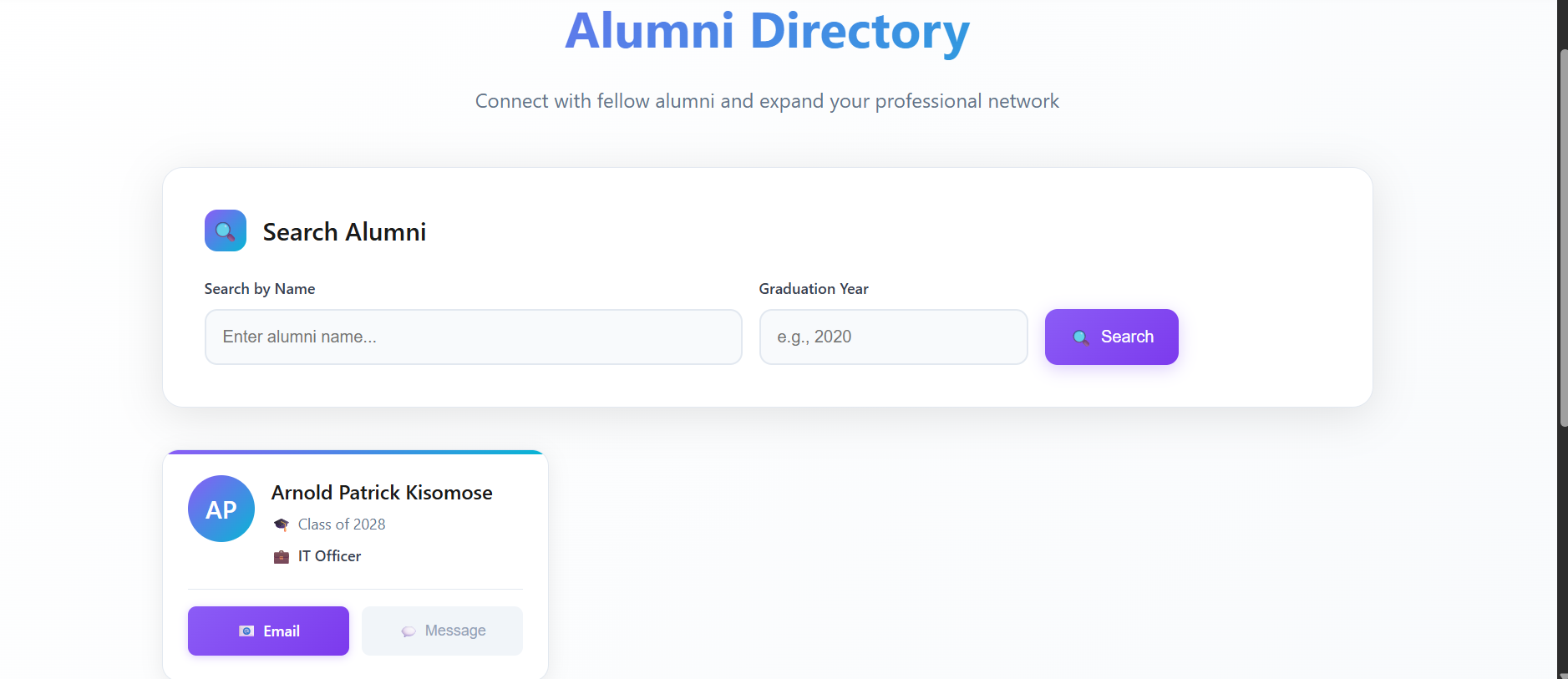
* **Landing Page**: Displays welcome message, recent news, upcoming events, and testimonials.  
  

Figure 6. 1 Landing Page

* **Admin Dashboard**: Shows stats (users, events, RSVPs) and engagement trends.  
  *Image: screenshots/admin\_dashboard.png*
* **Alumni Dashboard**: Displays user profile.  
  *Image: screenshots/alumni\_dashboard.png*
* **Alumni Directory**: Searchable list of alumni by name, year, or program.  
  
* **Event RSVP**: Form for users to RSVP to events with status and notes.  
  *Image: screenshots/event\_rsvp.png*

# Source Code

The source code is hosted on GitHub for version control and collaboration. The repository includes all project files (app.py, models/, routes/, templates/, utils/, static/) and a README.md with setup instructions.

**GitHub Repository**: [syny-pulse/alumni-engage: A system built in Flask and Python to help University Alumni connect](https://github.com/syny-pulse/alumni-engage)

# Challenges and solutions

* **Challenge**: Managing testimonial approvals and contact responses.  
  **Solution**: Added is\_approved and admin\_response fields to testimonials and contact\_submissions tables, with admin panel interfaces.
* **Challenge**: Event RSVP tracking demanded careful database design to enforce one RSVP per user.  
  **Solution**: Implemented a UNIQUE(user\_id, event\_id) constraint in the rsvps table, validated at the application level to prevent duplicate submissions.

# Conclusion

This web application serves as a comprehensive platform for alumni engagement. It not only provides essential features like profile management and news but also fosters a connected community through events, testimonials, and job opportunities. Future improvements may include real-time chat, a mobile app, and external database integrations.