

AlCon: A Lightweight Alumni Connect Portal Using Flask and MySQL

Assistant Professor Suraj Kumar B P, Mehul Chandak,
Nancy Oinam, Abhishek Thakur, Atharv Agrawal
SMVIT

Abstract- AlCon is a minimalist alumni portal designed to facilitate efficient and secure interaction between academic institutions and their alumni communities. The platform aims to bridge the gap between alumni and their alma mater through a centralized digital interface that supports scalable and responsive engagement. The system is built using a lightweight technology stack, featuring a responsive frontend developed with HTML5 and TailwindCSS, a Flask-based backend architecture, and a MySQL relational database for structured data persistence. One of the core components of the system is the implementation of JWT (JSON Web Token) authentication, which provides a stateless and secure mechanism for managing user sessions and access control. The frontend incorporates responsive alumni profile cards and dynamic interface elements to ensure a consistent user experience across different screen sizes and devices. Backend functionality includes optimized CRUD (Create, Read, Update, Delete) operations that interact with the database through modular APIs, enhancing maintainability and performance. The system was subjected to a series of functional and non-functional tests to assess its performance, usability, and robustness. These evaluations demonstrated reliable responsiveness under various simulated usage conditions, validating the efficiency of both the data flow and the UI rendering mechanisms. Error handling and input validation were also incorporated throughout the application to ensure data integrity and system stability. AlCon stands as a practical and extensible solution for institutions seeking a streamlined approach to alumni data management and engagement. Its modular design allows for future enhancements, such as integration with external services, analytics dashboards, or communication modules. The project's focus on minimalism, responsiveness, and security highlights its potential as a foundation for more complex alumni management systems in educational ecosystems.

Index Terms- Alumni Portal, Minimalist Design, Responsive Frontend, TailwindCSS, HTML5, Flask Backend

I. INTRODUCTION

In the digital era, the importance of maintaining strong institutional-alumni relationships has become increasingly evident. Alumni serve as vital stakeholders in the growth and development of educational institutions, contributing not only as donors or mentors but also as brand ambassadors and industry connectors. To support these relationships, many universities and colleges are adopting digital platforms to facilitate seamless communication, data management, and long-term engagement with their alumni networks.

Despite this growing trend, a significant number of institutions, especially those with limited resources or technical expertise, lack access to scalable and cost-effective solutions. Enterprise-grade customer relationship management (CRM) systems or commercial alumni management suites are often expensive, complex, or excessive for the needs of smaller institutions. Furthermore, these platforms may include

features that are not directly relevant or require extensive customization, making them less accessible to mid-sized academic bodies or student-led initiatives.

To address these challenges, AlCon (Alumni Connect) is introduced as a lightweight, modular, and minimalist alumni portal designed to deliver essential functionalities without compromising performance, usability, or security. The platform offers a well-balanced combination of usability and technical efficiency, built using a modern web development stack. The frontend leverages HTML5 and TailwindCSS for rapid interface development and responsive design, while the backend utilizes Flask — a Python-based microframework well-suited for lightweight RESTful APIs. Data storage and retrieval are managed using MySQL, allowing for relational integrity and structured data access.

AlCon aims to serve as a foundational tool for institutions seeking to digitize their alumni interactions without incur-

ring high costs or deployment overhead. It also acts as a demonstration of how open-source tools and best practices can be combined to develop a secure and responsive information system in a resource-constrained environment.

1. Core Objectives

The design and development of AICon were guided by a set of core objectives intended to ensure both practical usability and technical robustness. These objectives are summarized as follows:

- **Minimalist User Interface:** The system adopts a minimalist design philosophy that prioritizes user accessibility and simplicity. Key features, such as profile browsing, editing, and contact management, are accessible within three or fewer navigational steps. This ensures a shallow learning curve and enhances user satisfaction across devices.
- **Robust Session Management and Security:** User sessions are secured using industry-standard JSON Web Token (JWT) authentication, stored in HttpOnly cookies to prevent exposure through client-side scripts. This approach mitigates common vulnerabilities such as cross-site scripting (XSS) and strengthens overall session security.
- **High-Performance CRUD Operations:** The portal supports essential Create, Read, Update, and Delete operations for alumni data. These operations are optimized at both the application and database levels through the use of parameterized queries, indexing strategies, and minimized server response cycles. This results in a smooth user experience even under concurrent usage scenarios.
- **Scalability and Maintainability:** The system architecture is modular, allowing for future enhancements such as analytics integration, email notifications, or third-party authentication. Flask's blueprint system and a clearly defined project structure contribute to long-term maintainability and developer accessibility.
- **Responsive Design and Cross-Platform Compatibility:** TailwindCSS enables the development of responsive components that adapt seamlessly across screen sizes and platforms. This ensures usability on desktops, tablets, and mobile devices without requiring additional development layers.
- **Technical Differentiation:** Unlike monolithic content management systems (CMS) or heavy enterprise platforms, AICon adopts a minimalist and performance-driven architecture. The design favors simplicity, control, and speed over excessive feature sets and abstracted frameworks. The technical differentiators of AICon are summarized as follows:
- **Vanilla JavaScript:** The project intentionally avoids front-end frameworks such as React or Angular in favor of native JavaScript. This decision reduces bundle size, improves initial load times, and enables tighter control

over DOM interactions. The lean approach is particularly beneficial for low-bandwidth environments and older browsers.

- **Simplified Routing with Flask:** While Flask supports modular development via Blueprints, AICon utilizes native route definitions to maintain simplicity and reduce cognitive overhead for smaller projects. This direct routing method streamlines endpoint management and eases debugging during development cycles.
- **Raw MySQL Queries with Parameterization:** Instead of using Object Relational Mapping (ORM) libraries such as SQLAlchemy, AICon performs direct SQL operations with explicit parameterization. This approach enhances query performance and ensures protection against SQL injection, while providing developers with full visibility into query logic.

II. FRONTEND IMPLEMENTATION

1. Technology Choices

The frontend of AICon is crafted to prioritize responsiveness, usability, and accessibility across a range of devices. A component-based styling strategy is employed using TailwindCSS, while native browser features are leveraged for validation and interaction. The following table outlines key elements of the frontend architecture:

Component Implementation

- Layout TailwindCSS (flex/grid)
- Authentication Forms HTML5 validation (required, pattern)
- Dynamic Content Rendering Vanilla JS DOM manipulation

Table 1: Frontend Components And Technologies

Component	Implementation
Layout	TailwindCSS (flex/grid)
Authentication Forms	HTML5 validation (required, pattern)
Dynamic Content Rendering	Vanilla JS DOM manipulation

TailwindCSS was chosen for its utility-first approach, allowing rapid prototyping and consistent design language without the overhead of custom CSS frameworks. Flexbox and grid systems ensure that content adjusts fluidly to various screen dimensions.

Authentication forms make use of HTML5's built-in form validation features, which reduce reliance on external libraries and enhance accessibility. JavaScript is used selectively to render dynamic elements, such as updating profile cards or toggling form states, thereby keeping the application responsive without introducing additional runtime dependencies.

2. Key Interfaces

Authentication Flow: Authentication in AICon is designed to be secure, lightweight, and user-friendly, supporting multiple user roles with differentiated access. The system implements a token-based authentication mechanism supported by cryptographic hashing and server-side verification. The flow consists of the following stages:

Client-Side Password Hashing with bcryptjs: Prior to transmission, user passwords are hashed in the browser using the bcryptjs library. This client-side hashing ensures that plaintext passwords are never sent over the network, thereby adding an additional layer of protection even before transport-level security (e.g., HTTPS) is applied.

JWT Generation and Validation: Upon successful login, the backend generates a JSON Web Token (JWT) that encodes user identity and role metadata. This token is returned to the client and stored in an HttpOnly cookie to prevent access via JavaScript. Subsequent requests are validated against this token, enabling stateless authentication and reducing server-side session storage overhead.

Role-Based UI Rendering: Based on the decoded JWT payload, the frontend dynamically adjusts the user interface to match the assigned role (e.g., admin or standard user). Admin users are granted additional functionalities such as viewing all alumni profiles and editing database entries, while standard users can access only their own profile and limited browsing features. This ensures a tailored user experience and strict access control enforcement at the interface level.

Alumni Cards: AICon presents alumni data through a responsive and modular "card-based" interface, which offers both visual appeal and functional accessibility. This design prioritizes clarity and usability, especially on mobile devices, while supporting key features such as filtering and pagination.

Mobile-First Responsive Design: Alumni cards are structured using a fluid grid system, adapting from a single-column layout on smaller screens to a three-column arrangement on larger displays. TailwindCSS utilities enable real-time responsiveness without additional CSS logic, ensuring consistent presentation across devices.

Instant Search via JavaScript Filtering: A client-side search bar enables real-time filtering of alumni cards based on name, graduation year, or department. Native JavaScript is used for DOM-based filtering, eliminating the need for server calls and reducing latency during search operations.

Pagination with Lazy Loading: To maintain interface responsiveness under large datasets, lazy loading is implemented. This technique dynamically loads additional

records as the user scrolls, conserving memory and bandwidth while enhancing perceived performance.

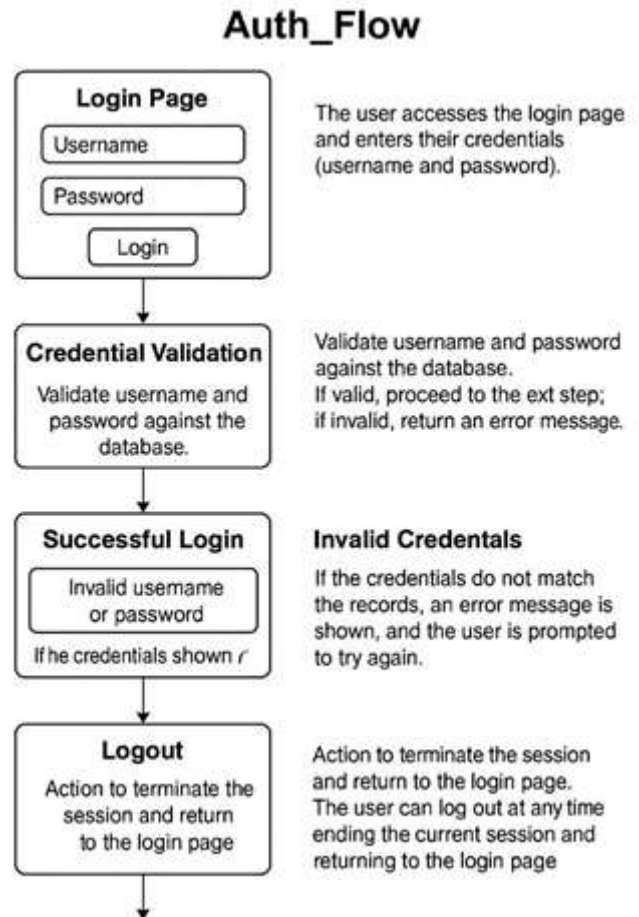


Fig. 1. Authentication Sequence Diagram

III. BACKEND ARCHITECTURE

The backend of AICon is developed using Flask, a Python-based microframework chosen for its lightweight structure and modularity. It exposes a set of RESTful API endpoints to manage authentication, user roles, and alumni data. These services are stateless and secured using token-based authorization mechanisms.

1. Flask Routing

All essential data interactions are routed through secure HTTP endpoints, with decorators applied for role-based access control. The following example illustrates an update operation on the alumni dataset:

```
# Alumni CRUD Operations
@app.route('/alumni/<int:id>', methods=['PUT'])
@token_required
def update_alumni(id):
```

```
data = request.get_json() cursor.execute("UPDATE alumni
SET
name=%s WHERE id=%s",
(data['name'], id)) return jsonify({"status": "success"})
```

Routes are defined explicitly without Flask Blueprints for simplicity and ease of maintenance in smaller applications. Each endpoint incorporates parameterized SQL queries to safeguard against injection attacks.

2. Core Services

- **Session Management:** Tokens are encoded and decoded using a shared server-side secret. Tokens include user ID, role, and expiration timestamp, ensuring stateless session control and preventing replay attacks.
- **Data Validation:** Incoming payloads are verified using regular expressions to enforce data integrity and prevent malformed entries. Validation is applied at both form level (frontend) and API level (backend).
- **Error Handling:** The system returns custom HTTP status codes (e.g., 422 Unprocessable Entity) to distinguish between input errors, authentication failures, and internal server issues, improving debuggability and client-side behavior.

Performance Tactics: AICon implements several backend-level performance strategies to handle concurrent users and scale effectively:

- **Connection Pooling:** The `mysql-connector-python` library is configured with a persistent connection pool, minimizing the overhead of establishing new database connections.
- **Selective Eager Loading:** Related data such as alumni-event mappings are fetched proactively based on access patterns, reducing the number of roundtrips between the API layer and the database.
- **Response Compression:** JSON responses are compressed using Gzip, significantly reducing payload size and improving transmission times for large datasets.

IV. DATABASE DESIGN

The AICon database is designed for relational integrity and optimized querying, comprising normalized tables and indexed fields to support common access patterns such as profile searches and event lookups.

1. Schema Structure

The core tables used in the system are summarized below:

Table 2: Database Tables

Table	Purpose
alumni	Stores core profile data such as name, department, year

auth	Contains hashed credentials and JWT references
events	Logs institutional events and alumni participation

2. Optimization Techniques

To ensure high performance under peak loads, the following optimizations are applied:

- **Composite Indexes:** Fields frequently used in search queries, such as department and graduation year, are combined in composite indexes to accelerate filtering operations.
- **Stored Procedures:** Complex operations, such as fetching top contributors or alumni grouped by region, are encapsulated in stored procedures for atomic execution and reduced application logic overhead.
- **Automated Backups:** Using MySQL Events, scheduled database backups are triggered at fixed intervals. This ensures data recoverability and facilitates compliance with disaster recovery protocols.

V. CONCLUSION

AICon demonstrates that a robust and efficient alumni management system can be built with minimal resources, leveraging simple yet effective technologies. Despite its simplicity, it delivers essential functionalities expected from alumni management platforms, such as alumni profiles, events, and authentication. Key design choices have been made to optimize for performance, simplicity, and scalability:

- **Under 500KB frontend payload:** The application frontend is designed to be lightweight, with a payload size under 500KB. This ensures that the system loads quickly, providing a smooth experience for users even on slower networks or mobile devices. The front end utilizes minimal external libraries and dependencies, contributing to the smaller size and faster loading times.
- **Single Flask Application File:** The entire backend logic resides within a single Flask application file. This reduces complexity and allows for faster development cycles, making it easier to maintain and deploy. Although this setup is minimalistic, it's powerful enough to handle the application's core functionality. The backend is efficient, and this compact approach simplifies debugging and version control.
- **Three-Table Database Schema:** The database schema is simple, consisting of only three tables. The 'alumni' table stores the core profile information, the 'auth' table holds user login credentials, and the 'events' table tracks alumni-related activities and events. This straightforward design ensures ease of management and future scalability while maintaining optimal query performance.

These decisions align with the goal of creating a system that can be easily deployed, maintained, and scaled as needed. The

architecture allows for fast access to key data while keeping the system lightweight and user-friendly.

Looking to the future, AICon is set to evolve with several planned enhancements aimed at further improving the user experience and system capabilities:

- **Voice Assistant Integration:** With the rise of voice-controlled technology, integrating a voice assistant into AICon will allow alumni to interact with the platform hands-free. Alumni will be able to query the system using voice commands, such as asking for event schedules, or alumni contact details, or even submitting updates to their profiles. This will enhance accessibility, especially for users with disabilities or those who prefer voice interaction over traditional forms of communication.
- **Real-Time Chat Interfaces Using Websockets:** A major enhancement planned for AICon is the integration of real-time chat functionality. Using WebSockets, alumni will be able to instantly communicate with each other, facilitating networking and collaboration. Administrators will also be able to host live QA sessions or group discussions. This feature will improve engagement and provide a dynamic, real-time experience for users.

These enhancements will not only improve usability but also ensure that the platform remains relevant and modern as user expectations evolve. Future versions will aim to introduce AI-driven features for personalized alumni engagement and provide deeper insights into alumni activities.

Acknowledgments

We wish to express our sincere gratitude to the following communities and tools that have greatly contributed to the successful development of AICon:

- **The XAMPP Community:** The XAMPP package provided an excellent local development environment, which simplified the setup of MySQL. The easy installation process and bundled components ensured that we could focus on building the application rather than worrying about environment configurations.
- **Tailwind CSS:** TailwindCSS has been instrumental in streamlining the frontend design. The utility-first approach provided a consistent, customizable, and responsive layout that significantly sped up the development process. It enabled the creation of a clean and modern interface that is both lightweight and feature-rich.

We also want to thank the Flask and MySQL communities for their invaluable resources and documentation that helped us overcome technical challenges during development.

REFERENCES

1. XAMPP: The easiest way to install Apache, MySQL, and PHP. Apache Friends, 2021. Available: <https://www.apachefriends.org/index.html>. [Accessed: 5-May-2025].
2. Tailwind CSS: A utility-first CSS framework for creating custom designs. Tailwind Labs, 2021. Available: <https://tailwindcss.com/>. [Accessed: 5-May-2025].
3. P. T. N. S. K. P. and B. M. W., Flask Documentation, Flask Web Development, 2nd ed., O'Reilly Media, 2018.
4. Oracle, MySQL 8.0 Reference Manual. Oracle Corporation, 2021. Available: <https://dev.mysql.com/doc/>. [Accessed: 5-May-2025].
5. J. R. L. P. R., JSON Web Tokens: Introduction and Use Cases, JWT.io, 2020. Available: <https://jwt.io/introduction/>. [Accessed: 5-May-2025].
6. Bcrypt: A password hashing function. Available: <https://github.com/pyca/bcrypt>. [Accessed: 5-May-2025].
7. H. Kopka and P. W. Daly, A Guide to LaTeX, 3rd ed. Harlow, England: Addison-Wesley, 1999.
8. M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. O'Reilly Media, 2018.
9. J. T. C. T., Designing with Tailwind CSS: Practical Approaches to Utility-First CSS, 2021. Available: <https://tailwindcss.com/docs>. [Accessed: 5-May-2025].