**1. Introduction**

**1.1 Purpose**

The purpose of this Alumni Management Software (AMS) is to provide universities and colleges with a centralized, secure platform to foster lifelong connections with alumni. By streamlining alumni engagement, the software aims to transform alumni into active contributors to the institution’s growth, serving as brand ambassadors, donors, and mentors. The system will automate critical processes such as data management, event coordination, mentorship matching, and donation drives.

**1.2 Project Scope**

* Alumni registration, profile management, and networking
* Event planning
* Mentorship program management.
* Crowdfunding.
* Newsletters and communication tools.
* Secure database for alumni records.

**1.3 Glossary & Abbreviation**

| Term | Definition |
| --- | --- |
| Alumni | Graduates of the institution. |
| Crowdfunding | Collective effort to raise funds via small contributions from alumni. |
| Mentorship Program | Structured initiative pairing students with alumni for career guidance. |
| RSVP | Confirmation of attendance for events. |
| DB | Database |
| Users | It includes Alumni, Students, Faculty Staff, Admins. |

**1.4 List of the System Stakeholders**

* Alumni: Primary users who engage with the platform for networking, events, and mentorship.
* Admin: Manage alumni data.
* Students: Utilize mentorship programs and alumni networks for career development.
* Faculty Staff: organize events, and oversee communication campaigns.

**1.5 References**

* Alumni Management Information: [NPO Info Alumni Management](https://npoinfo.com/alumni-management/)
* Alumni Management Example: [Alamabase](https://www.almabase.com/)

**2. Functional Requirements**

| **Function** | **Type** | **MoSCoW** |
| --- | --- | --- |
| **Users can register using Unique username, password, and role(Alumni, Student).** | **User Requirement** | **Must** |
| **Users can log in using username/password.** | **User Requirement** | **Must** |
| **System grants access based on role upon login.** | **System Requirement** | **Must** |
| **Users can change their username and password.** | **User Requirement** | **Should** |
| **Newly registered alumni need to be verified to access the system** | **User Requirement** | **Must** |
| **Faculty Staff should verify Alumni's account .** | **User Requirement** | **Should** |
| **Faculty Staff can view all unverified Alumni in a single page** | **System Requirement** | **Must** |
| **Alumni can donate for a selected cause.** | **User Requirement** | **Must** |
| **Alumni can view their donation history.** | **User Requirement** | **Should** |
| **Faculty Staff can view Alumni donations.** | **User Requirement** | **Should** |
| **Alumni can serve as a mentor in their chosen field.** | **User Requirement** | **Should** |
| **Students can view and filter mentors.** | **User Requirement** | **Should** |
| **Students can join a mentorship with a chosen mentor** | **User Requirement** | **Should** |
| **Users can search Alumni with filters.** | **User Requirement** | **Should** |
| **Faculty Staff can schedule new events with name, date and description.** | **User Requirement** | **Must** |
| **Faculty Staff can Reschedule and Edit Events** | **User Requirement** | **Could** |
| **Alumni can sign up for events.** | **User Requirement** | **Must** |
| **System sends Alumni notifications for new events.** | **System Requirement** | **Should** |
| **Faculty Staff can view event participants.** | **User Requirement** | **Should** |
| **Faculty Staff can draft a newsletter.** | **User Requirement** | **Must** |
| **Faculty Staff can publish a newsletter.** | **System Requirement** | **Must** |
| **Users can view newsletters.** | **User Requirement** | **Should** |
| **System sends Users newsletter notifications.** | **System Requirement** | **Should** |
| **Admin can view User Accounts in the Admin Panel.** | **User Requirement** | **Should** |
| **Admin can add User Accounts in Admin Panel** | **User Requirement** | **Should** |
| **Admin can edit User Accounts in Admin Panel** | **User Requirement** | **Should** |
| **Admin can Remove User Accounts in Admin Panel** | **User Requirement** | **Should** |
| **Admins can register new Admins and Faculty Staff** | **User Requirement** | **Must** |
| **Faculty Staff can generate reports on events.** | **System Requirement** | **Won’t Have** |
| **Faculty Staff can generate reports on Alumni donations.** | **System Requirement** | **Won’t Have** |
| **Alumni can donate via PayPal/Stripe.** | **System Requirement** | **Won’t Have** |

**3. Non-functional requirements:**

| **requirement** | **type** | **specification** | **Fit criteria** | **affect** |
| --- | --- | --- | --- | --- |
| The system shall be available to all alumni and students at all times and should be managed by the staff | Product Requirement | Usability requirement | Verifiable | this makes sure the system works for all users of no matter where and when they access it |
| The system shall respond to user inputs within 2s for 95% of requests under normal load. | Product Requirement | performance Requirement | Testable | this make sure the system acts fast and respond quickly to user actions |
| The system shall adhere to responsive design principles, functioning on all modern browsers and devices. | Product Requirement | Dependability Requirement | Verifiable | makes sure the system works for different types of system |
| The system will be secure and make sure authorized users access the system using role based action control. | Product Requirement | security Requirement | Testable | makes sure the system is secure and there won't be data leaks or unauthorized access to the system |
| The system shall use HTTPS for all communications and operate over TCP/IP. | External Requirement | Regulatory Requirement | Testable | the system will use common data communication protocols to connect users to the service |
| The system shall be implemented using OOP and DRY standards. | Organizational Requirement | Development Requirement | Verifiable | makes the sure the system follows modern standards and make the system easy to maintain |

**4. Design & Implementation Constraints**

**Technology Choices:**

**Frontend:**

Built with HTML, CSS, JavaScript ,and Bootstrap.

**Backend:**

Runs on PHP and MYSQL PDO.

**System Structure:**

**Single, Simple Design:**

Frontend and backend combined in one system, and the code is organized so it can later be split if needed.

**No Outsourced Services:**

Handles logins, data storage, and features internally (no reliance on external tools).

**5 System Evolution**

**5.1 Anticipated Changes**

| **Phase** | **Potential Upgrades** | **Business Need** |
| --- | --- | --- |
| **Short-Term (1-2 years)** | - Mobile app (iOS/Android) support  - Payment gateway integration (donations)  - Advanced search filters (alumni directory) | - Improve accessibility for alumni on-the-go  - Streamline fundraising  - Enhanced networking |
| **Mid-Term (2-3 years)** | - AI-driven alumni matching (mentorship/jobs)  - Automated newsletter personalization  - Multi-language support | - Boost engagement through smart recommendations  - Reduce manual communication effort  - Expand global reach |
| **Long-Term (3-5+ years)** | - Predictive analytics (donation/engagement trends)  - Integration with university LMS | - Data-driven decision-making  - Centralized student-alumni lifecycle management  - Secure degree/achievement validation |

**5.2 System Evolution, Design Impact & Preparation**

* Use a modular architecture to support future feature expansion.
* Ensure scalability in both database and hosting infrastructure.
* Design responsive UI for multi-device access.
* Follow best practices in security and data management for future compliance.

**6. Requirements Discovery Approaches**

**1. Interviews**

**Description:**

One-on-one discussions with key stakeholders to gather in-depth insights into their needs, expectations, and pain points.

**Example:**

Interview faculty event organizers to understand their current process for planning reunions and workshops.

Ask questions such as:

* “How do you currently manage RSVPs?”
* “What information do you typically collect from alumni for events?”

**2. Surveys & Questionnaires**

**Description:**

Distribute structured forms to a larger group of stakeholders to collect feedback on potential features.

**Example:**

Send online surveys to alumni to prioritize features like:

* Mentorship programs
* Donation Tracking
* Newsletters or job opportunities

Include Likert-scale and open-ended questions such as:

* “How useful would a mentorship matching feature be to you?”
* “What additional tools would help you stay engaged with the faculty?”

**3. Competitor & Benchmark Analysis**

**Description:**

Analyze existing alumni systems used by other universities or institutions to identify industry best practices and gaps.

**Example:**

Study systems like Graduway, Almabase, or LinkedIn Alumni Networks.

Note key features such as:

* Alumni map
* Event RSVP integration with calendars
* Secure login with social OAuth
* Identify missing or underdeveloped features and consider them for inclusion.

**4. Document Analysis**

**Description:**

Review existing documents, forms, reports, and communication records related to alumni activities.

**Example:**

* Analyze past event invitations, donation forms, and alumni newsletters.
* Extract common data fields and communication templates for system input.

## **7. Requirements Validation Techniques**

To ensure the system meets stakeholder needs and functions as expected, the following validation techniques will be used:

### **1. Prototyping**

**Description:** Develop low-fidelity UI mockups of key features such as event creation, profile editing, and donation forms.

**Example:**

* Present a clickable prototype of the alumni dashboard to stakeholders.
* Validate that alumni can easily update their profiles, RSVP to events, and access the mentorship section.
* Gather feedback to confirm layout clarity, feature placement, and usability.

### **2. Requirement Reviews**

**Description:** Formal or informal walkthroughs of the documented requirements with stakeholders.

**Example:**

* Review the event management and donation tracking requirements with administrative staff.
* Confirm that all user actions (e.g., adding event details, viewing donation history) are clearly defined and aligned with user expectations.

### **3. Use Case Scenarios**

**Description:** Define real-world scenarios to ensure requirements cover expected user behavior.

**Example:**

* Scenario: An alumni logs in, updates career info, signs up as a mentor, and RSVPs to an event.
* Verify that each step is supported by the functional requirements and system flow.

### **4. User Feedback Sessions**

**Description:** Gather direct feedback from a small group of alumni and staff on early system versions or mockups.

**Example:**

* Share early versions of the newsletter creation module with staff to validate content creation flow.
* Ask, “Is the email preview feature clear and useful?” or “Are audience filters easy to apply?”