# AlumNet API - Complete Documentation

## Table of Contents



---

## Overview

AlumNet is a comprehensive alumni networking platform built with Node.js, Express, and MongoDB. This API provides endpoints for user authentication, content management, event organization, and alumni networking features.

### Key Features

- JWT-based authentication with role-based access control
- Post creation and management with likes and comments
- Event organization and attendance tracking
- Alumni directory with advanced search and filtering
- Admin panel for user and content management
- File upload with validation
- Security features including rate limiting and input sanitization

---

## Base Configuration

### Base URL

```
http://localhost:4000/api
```

### Environment Variables

```
PORT=4000
MONGO_URI=mongodb://localhost:27017/alumni_connect
JWT_SECRET=your-strong-secret-key
CLIENT_ORIGIN=http://localhost:3000
COLLEGE_EMAIL_DOMAIN=college.edu
```

### Content Types

- **JSON**: application/json
- **Form Data**: multipart/form-data (for file uploads)

# Authentication

## JWT Token Structure

```
{
  "sub": "64f1a2b3c4d5e6f7g8h9i0j1",
  "role": "alumni",
  "iat": 1693567890,
  "exp": 1694172690
}
```

## Authorization Header

```
Authorization: Bearer <your-jwt-token>
```

## User Roles

- **student**: Basic access, can view content and attend events
- **alumni**: Can create posts, organize events, full alumni features
- **admin**: Full system access, user management, content moderation

---

# API Endpoints

## Authentication Endpoints

### 1. Register User

**POST** /auth/register

**Description**: Register a new user (student or alumni)

**Request Body:**

```
{
  "name": "John Doe",
  "email": "john@college.edu",
  "password": "password123",
  "role": "student",
  "graduationYear": 2023,
  "department": "Computer Science",
  "batch": 2019,
  "course": "B.Tech",
  "currentJob": "Software Engineer"
}
```

**Validation Rules:**

- name: Required, trimmed
- email: Required, valid email format, must end with college domain
- password: Required, minimum 6 characters

- role: Optional, must be "student" or "alumni"
- batch: Required if role is "alumni"
- course: Required if role is "alumni"

**Response (201 Created):**


```
{
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "student"
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```


**Error Responses:**

- 400: Validation errors
- 409: Email already exists

**2. Login User**

**POST** /auth/login

**Description**: Authenticate user and return JWT token

**Request Body:**


```
{
  "email": "john@college.edu",
  "password": "password123"
}
```


**Response (200 OK):**


```
{
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "student",
    "avatarUrl": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png"
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```


**Error Responses:**

- 401: Invalid credentials
- 403: Account inactive

**3. Get Current User**

**GET** /auth/me

**Description**: Get current authenticated user's profile

**Response (200 OK):**


```
{
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "student",
    "graduationYear": 2023,
    "department": "Computer Science",
    "avatarUrl": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png",
    "verified": true,
    "isActive": true,
    "createdAt": "2023-09-01T10:00:00.000Z",
    "updatedAt": "2023-09-01T10:00:00.000Z"
  }
}
```


**4. Update Current User**

**PUT** /auth/me

**Description**: Update current user's profile information

**Request Body:**


```
{
  "name": "John Smith",
  "graduationYear": 2024,
  "department": "Information Technology"
}
```


**Response (200 OK):**


```
{
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Smith",
    "email": "john@college.edu",
    "role": "student"
  }
}
```


**5. Forgot Password**

**POST** /auth/forgot-password

**Description**: Generate password reset token

**Request Body:**

```
{
  "email": "john@college.edu"
}
```

**Response (200 OK):**

```
{
  "message": "Password reset token generated",
  "resetToken": "a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0u1v2w3x4y5z6"
}
```

**6. Reset Password**

**POST** /auth/reset-password

**Description**: Reset password using token

**Request Body:**

```
{
  "token": "a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0u1v2w3x4y5z6",
  "password": "newpassword123"
}
```

**Response (200 OK):**

```
{
  "message": "Password has been reset successfully"
}
```

---

## Posts Endpoints

**1. Get All Posts**

**GET** /posts

**Description**: Retrieve paginated list of posts with optional search

**Query Parameters:**

- page (number, optional): Page number (default: 1)
- limit (number, optional): Posts per page (default: 10)
- search (string, optional): Search in title and content

**Example Request:**

```
GET /posts?page=1&limit=5&search=javascript
```

**Response (200 OK):**

```
{
  "posts": [
    {
      "_id": "64f1a2b3c4d5e6f7g8h9i0j2",
      "title": "JavaScript Best Practices",
      "content": "Here are some JavaScript best practices...",
      "imageUrl": "/uploads/javascript-guide-1693567890123.jpg",
      "author": {
        "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
        "name": "John Doe",
        "role": "alumni"
      },
      "likes": ["64f1a2b3c4d5e6f7g8h9i0j1", "64f1a2b3c4d5e6f7g8h9i0j3"],
      "comments": [
        {
          "_id": "64f1a2b3c4d5e6f7g8h9i0j4",
          "user": "64f1a2b3c4d5e6f7g8h9i0j1",
          "text": "Great post!",
          "createdAt": "2023-09-01T11:00:00.000Z"
        }
      ],
      "likesCount": 2,
      "commentsCount": 1,
      "createdAt": "2023-09-01T10:30:00.000Z",
      "updatedAt": "2023-09-01T10:30:00.000Z"
    }
  ],
  "pagination": {
    "currentPage": 1,
    "totalPages": 3,
    "totalPosts": 25,
    "hasNext": true,
    "hasPrev": false
  },
  "search": "javascript"
}
```


**2. Get Single Post**

**GET** /posts/:id

**Description**: Retrieve a specific post by ID

**Response (200 OK):**

```
{
  "_id": "64f1a2b3c4d5e6f7g8h9i0j2",
  "title": "JavaScript Best Practices",
  "content": "Here are some JavaScript best practices...",
  "imageUrl": "/uploads/javascript-guide-1693567890123.jpg",
  "author": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe",
    "role": "alumni"
  },
  "likes": ["64f1a2b3c4d5e6f7g8h9i0j1", "64f1a2b3c4d5e6f7g8h9i0j3"],
  "comments": [
    {
      "_id": "64f1a2b3c4d5e6f7g8h9i0j4",
      "user": "64f1a2b3c4d5e6f7g8h9i0j1",
      "text": "Great post!",
      "createdAt": "2023-09-01T11:00:00.000Z"
    }
  ],
  "likesCount": 2,
  "commentsCount": 1,
  "createdAt": "2023-09-01T10:30:00.000Z",
  "updatedAt": "2023-09-01T10:30:00.000Z"
}
```


**3. Create Post**

**POST** /posts (Alumni/Admin only)

**Description**: Create a new post (requires alumni or admin role)

**Request Body (multipart/form-data):**

```
title: "My New Post"
content: "This is the content of my post"
image: <file> (optional)
```

**Response (201 Created):**


```
{
  "_id": "64f1a2b3c4d5e6f7g8h9i0j2",
  "title": "My New Post",
  "content": "This is the content of my post",
  "imageUrl": "/uploads/my-post-1693567890123.jpg",
  "author": "64f1a2b3c4d5e6f7g8h9i0j1",
  "likes": [],
  "comments": [],
  "createdAt": "2023-09-01T10:30:00.000Z",
  "updatedAt": "2023-09-01T10:30:00.000Z"
}
```


**Error Responses:**

- 403: Only alumni and admins can create posts
- 403: Alumni account must be verified

**4. Update Post**

**PATCH** /posts/:id (Author/Admin only)

**Description**: Update an existing post

**Request Body (multipart/form-data):**

```
title: "Updated Post Title"
content: "Updated content"
image: <file> (optional)
```

**Response (200 OK):**

```
{
  "_id": "64f1a2b3c4d5e6f7g8h9i0j2",
  "title": "Updated Post Title",
  "content": "Updated content",
  "imageUrl": "/uploads/updated-post-1693567890123.jpg",
  "author": "64f1a2b3c4d5e6f7g8h9i0j1",
  "likes": ["64f1a2b3c4d5e6f7g8h9i0j1"],
  "comments": [],
  "createdAt": "2023-09-01T10:30:00.000Z",
  "updatedAt": "2023-09-01T11:00:00.000Z"
}
```

**5. Delete Post**

**DELETE** /posts/:id (Author/Admin only)

**Description**: Delete a post

**Response (204 No Content):**

```
(Empty response body)
```

**6. Like/Unlike Post**

**POST** /posts/:id/like

**Description**: Toggle like status for a post

**Response (200 OK):**

```
{
  "liked": true,
  "likes": 5
}
```

**7. Comment on Post**

**POST** /posts/:id/comment

**Description**: Add a comment to a post

**Request Body:**

```
{
  "text": "This is a great post!"
}
```

**Response (201 Created):**

```
{
  "_id": "64f1a2b3c4d5e6f7g8h9i0j4",
  "user": "64f1a2b3c4d5e6f7g8h9i0j1",
  "text": "This is a great post!",
  "createdAt": "2023-09-01T11:00:00.000Z"
}
```

---

## Events Endpoints

**1. Get All Events**

**GET** /events

**Description**: Retrieve paginated list of events

**Query Parameters:**

- page (number, optional): Page number (default: 1)
- limit (number, optional): Events per page (default: 10)

**Response (200 OK):**

```
{
  "events": [
    {
      "_id": "64f1a2b3c4d5e6f7g8h9i0j5",
      "title": "Alumni Meet 2023",
      "description": "Annual alumni gathering",
      "date": "2023-12-15T18:00:00.000Z",
      "location": "College Auditorium",
      "organizer": {
        "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
        "name": "John Doe"
      },
      "attendees": ["64f1a2b3c4d5e6f7g8h9i0j1", "64f1a2b3c4d5e6f7g8h9i0j3"],
      "createdAt": "2023-09-01T10:00:00.000Z",
      "updatedAt": "2023-09-01T10:00:00.000Z"
    }
  ],
  "pagination": {
    "currentPage": 1,
    "totalPages": 2,
    "totalEvents": 15,
    "hasNext": true,
    "hasPrev": false
  }
}
```


**2. Get Single Event**

**GET** /events/:id

**Description**: Retrieve a specific event by ID

**Response (200 OK):**


```
{
  "_id": "64f1a2b3c4d5e6f7g8h9i0j5",
  "title": "Alumni Meet 2023",
  "description": "Annual alumni gathering",
  "date": "2023-12-15T18:00:00.000Z",
  "location": "College Auditorium",
  "organizer": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe"
  },
  "attendees": ["64f1a2b3c4d5e6f7g8h9i0j1", "64f1a2b3c4d5e6f7g8h9i0j3"],
  "createdAt": "2023-09-01T10:00:00.000Z",
  "updatedAt": "2023-09-01T10:00:00.000Z"
}
```


**3. Create Event**

**POST** /events

**Description**: Create a new event

**Request Body:**


```
{
  "title": "Tech Conference 2023",
  "description": "Annual technology conference",
  "date": "2023-11-20T09:00:00.000Z",
  "location": "Convention Center"
}
```


**Response (201 Created):**


```
{
  "_id": "64f1a2b3c4d5e6f7g8h9i0j6",
  "title": "Tech Conference 2023",
  "description": "Annual technology conference",
  "date": "2023-11-20T09:00:00.000Z",
  "location": "Convention Center",
  "organizer": "64f1a2b3c4d5e6f7g8h9i0j1",
  "attendees": [],
  "createdAt": "2023-09-01T10:00:00.000Z",
  "updatedAt": "2023-09-01T10:00:00.000Z"
}
```


**4. Update Event**

**PATCH** /events/:id (Organizer/Admin only)

**Description**: Update an existing event

**Request Body:**


```
{
  "title": "Updated Tech Conference 2023",
  "description": "Updated description",
  "date": "2023-11-25T09:00:00.000Z",
  "location": "New Convention Center"
}
```


**Response (200 OK):**


```
{
  "_id": "64f1a2b3c4d5e6f7g8h9i0j6",
  "title": "Updated Tech Conference 2023",
  "description": "Updated description",
  "date": "2023-11-25T09:00:00.000Z",
  "location": "New Convention Center",
  "organizer": "64f1a2b3c4d5e6f7g8h9i0j1",
  "attendees": ["64f1a2b3c4d5e6f7g8h9i0j1"],
  "createdAt": "2023-09-01T10:00:00.000Z",
  "updatedAt": "2023-09-01T11:00:00.000Z"
}
```

#### 5. Delete Event

**DELETE** /events/:id (Organizer/Admin only)

**Description**: Delete an event

**Response (204 No Content):**

```
(Empty response body)
```

#### 6. Attend Event

**POST** /events/:id/attend

**Description**: Mark attendance for an event

**Response (200 OK):**

```
{
  "attendees": 3
}
```

#### 7. Leave Event

**POST** /events/:id/leave

**Description**: Remove attendance from an event

**Response (200 OK):**

```
{
  "attendees": 2
}
```

---

## User Endpoints

#### 1. Get Public Alumni

**GET** /users/alumni

**Description**: Retrieve public alumni directory with filtering and search

**Query Parameters:**

- q (string, optional): Search query (name, department, course, job)
- department (string, optional): Filter by department
- course (string, optional): Filter by course
- job (string, optional): Filter by current job
- batch (number, optional): Filter by batch year
- graduationYear (number, optional): Filter by graduation year

**Example Request:**

```
GET /users/alumni?q=software&department=Computer Science&batch=2019
```

**Response (200 OK):**


```
{
  "alumni": [
    {
      "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
      "name": "John Doe",
      "email": "john@college.edu",
      "role": "alumni",
      "graduationYear": 2019,
      "department": "Computer Science",
      "avatarUrl": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png",
      "batch": 2019,
      "course": "B.Tech",
      "currentJob": "Software Engineer",
      "verified": true
    }
  ]
}
```


---

## Profile Endpoints

**1. Get My Profile**

**GET** /profile/me

**Description**: Get current user's complete profile

**Response (200 OK):**


```
{
  "user": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "student",
    "graduationYear": 2023,
    "department": "Computer Science",
    "avatarUrl": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png",
    "batch": null,
    "course": null,
    "currentJob": null,
    "verified": true,
    "isActive": true,
    "createdAt": "2023-09-01T10:00:00.000Z",
    "updatedAt": "2023-09-01T10:00:00.000Z"
  }
}
```


**2. Update My Profile**

**PUT** /profile/me

**Description**: Update current user's profile

**Request Body:**

```
{
  "name": "John Smith",
  "graduationYear": 2024,
  "department": "Information Technology"
}
```

**Response (200 OK):**

```
{
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Smith",
    "email": "john@college.edu",
    "role": "student"
  }
}
```

**3. Upload Avatar**

**POST** /profile/me/avatar

**Description**: Upload profile avatar image

**Request Body (multipart/form-data):**

```
avatar: <image file>
```

**Response (200 OK):**

```
{
  "avatarUrl": "/uploads/avatar-1693567890123.jpg"
}
```

**4. Get Public Profile by ID**

**GET** /profile/:id

**Description**: Get public profile information by user ID

**Response (200 OK):**

```
{
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "alumni",
    "graduationYear": 2019,
    "department": "Computer Science",
    "avatarUrl": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png",
    "batch": 2019,
    "course": "B.Tech",
    "currentJob": "Software Engineer",
    "verified": true
  }
}
```


---

## Admin Endpoints

### 1. List Users

**GET** /admin/users (Admin only)

**Description**: Retrieve all users with filtering options

**Query Parameters:**

- role (string, optional): Filter by role (student, alumni, admin)
- verified (boolean, optional): Filter by verification status
- q (string, optional): Search by name or email

**Response (200 OK):**


```
{
  "users": [
    {
      "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
      "name": "John Doe",
      "email": "john@college.edu",
      "role": "alumni",
      "graduationYear": 2019,
      "department": "Computer Science",
      "avatarUrl": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png",
      "batch": 2019,
      "course": "B.Tech",
      "currentJob": "Software Engineer",
      "verified": true,
      "isActive": true,
      "createdAt": "2023-09-01T10:00:00.000Z",
      "updatedAt": "2023-09-01T10:00:00.000Z"
    }
  ]
}
```

**2. Verify Alumni**

**POST** /admin/users/:id/verify (Admin only)

**Description**: Verify an alumni account

**Response (200 OK):**


```
{
  "message": "Alumni verified",
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "alumni",
    "verified": true
  }
}
```


**3. Set User Active Status**

**PATCH** /admin/users/:id/active (Admin only)

**Description**: Activate or deactivate a user account

**Request Body:**


```
{
  "isActive": false
}
```


**Response (200 OK):**


```
{
  "message": "User state updated",
  "user": {
    "id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "isActive": false
  }
}
```


**4. Delete User**

**DELETE** /admin/users/:id (Admin only)

**Description**: Permanently delete a user account

**Response (200 OK):**


```
{
  "message": "User deleted"
}
```


---

### Health Check

#### Health Check

**GET** /health

**Description**: Check API health status

**Response (200 OK):**

```
{
  "ok": true,
  "service": "AlumniConnect API"
}
```

---

# Error Handling

## Standard Error Response Format

```
{
  "message": "Error description",
  "details": "Additional error details (optional)"
}
```

## HTTP Status Codes

| Code | Description | Usage |
|---|---|---|
| 200 | OK | Successful GET, PUT, PATCH requests |
| 201 | Created | Successful POST requests |
| 204 | No Content | Successful DELETE requests |
| 400 | Bad Request | Invalid request data, validation errors |
| 401 | Unauthorized | Missing or invalid authentication |
| 403 | Forbidden | Insufficient permissions |
| 404 | Not Found | Resource not found |
| 409 | Conflict | Resource already exists |
| 429 | Too Many Requests | Rate limit exceeded |
| 500 | Internal Server Error | Server-side errors |

## Common Error Responses

#### 400 Bad Request

```
{
  "message": "Validation failed",
  "errors": [
    {
      "msg": "Name is required",
      "param": "name",
      "location": "body"
    }
  ]
}
```

**401 Unauthorized**

```
{
  "message": "Unauthorized"
}
```

**403 Forbidden**

```
{
  "message": "Forbidden"
}
```

**404 Not Found**

```
{
  "message": "Post not found"
}
```

**409 Conflict**

```
{
  "message": "Email already in use"
}
```

**429 Too Many Requests**

```
{
  "message": "Too many requests from this IP, please try again later."
}
```

---

# Rate Limiting

## Rate Limit Configuration

- **General endpoints**: 100 requests per 15 minutes per IP
- **Authentication endpoints**: 5 requests per 15 minutes per IP

## Rate Limit Headers

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1693567890
```

### Rate Limit Response

```
{
  "message": "Too many requests from this IP, please try again later."
}
```

---

## File Upload

### Supported File Types

- **Images**: PNG, JPEG, GIF, WebP
- **Maximum size**: 2MB per file

### Upload Endpoints

- **Post images**: POST /posts (multipart/form-data)
- **Avatar upload**: POST /profile/me/avatar (multipart/form-data)

### File Validation

- MIME type validation
- File extension validation
- File size validation
- Secure filename generation

### Upload Response

```
{
  "avatarUrl": "/uploads/avatar-1693567890123.jpg"
}
```

---

## Security Features

### Authentication & Authorization

- JWT-based authentication
- Role-based access control (RBAC)
- Password hashing with bcrypt
- Secure token generation for password reset

### Input Validation & Sanitization

- Express-validator for input validation
- MongoDB injection prevention
- XSS protection
- File upload validation

### Rate Limiting

- IP-based rate limiting
- Different limits for different endpoint types
- Prevents brute force attacks

### Security Headers

- Helmet.js for security headers
- CORS configuration
- Content Security Policy

### Data Protection

- Password field excluded from responses
- Sensitive data filtering
- Secure file storage

---

## Development Notes

### Environment Setup

1. Install dependencies: npm install
2. Create .env file with required variables
3. Start development server: npm run dev

### Database

- MongoDB with Mongoose ODM
- Automatic connection handling
- Index optimization for queries

### Testing

- Use tools like Postman or Insomnia for API testing
- Test all endpoints with different user roles
- Verify error handling and edge cases

### Deployment

- Ensure all environment variables are set
- Configure CORS for production domains
- Set up proper logging and monitoring
- Use HTTPS in production

---

## Support

For technical support or questions about this API, please refer to the project documentation or contact the development team.

**API Version**: 1.0.0

**Last Updated**: September 2023