A
Project Report
on

Lyceum

(Online Students Community Platform)

BTech IT, Sem VI

Prepared By:

Pranjal S. Yadav (IT-115)
Megh N. Sharma (IT-130)

Guided By:
Prof. Kunal J. Sahitya
Dept. of Information Technology

Department of Information Technology
Faculty of Technology,
Dharmsinh Desai University
College Road, Nadiad - 387001

April, 2025

CANDIDATE'S DECLARATION

We declare that the 6th semester report entitled "Lyceum" is our own work conducted under the supervision of the guide Prof. Kunal J Sahitya.

We further declare that to the best of our knowledge the report for B.Tech. VI semester does not contain part of the work which has been submitted either in this or any other university without proper citation.

Candidate's Signature

Candidate's Name: PRANJAL S. YADAV (IT115)

Student ID: 22ITUSS032

Candidate's Signature

Candidate's Name: MEGH N. SHARMA (IT130)

Student ID: 22ITUON062

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project carried out in the subject of Project-I, entitled “Lyceum” and recorded in this report is a bonafide report of the work of

1) Pranjal S. Yadav Roll No. IT115 ID No: 22ITUSS032
2) Megh N. Sharma Roll No. IT130 ID No: 22ITUON062

of Department of Information Technology, semester VI. They were involved in project work during academic year 2024–2025.

Prof. Kunal J. Sahitya
(Project Guide),
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. (Dr.) V. K. Dabhi
Head , Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to the **Department of Information Technology, Dharmsinh Desai University**, for providing us with the necessary resources, support, and an encouraging environment to undertake our project journey.

We sincerely thank our project guide, **Prof. Kunal J. Sahitya**, for his unwavering support, insightful feedback, and invaluable guidance throughout the development of our project, "**Lyceum**". His mentorship played a crucial role in shaping our understanding and successfully bringing this project to fruition.

We also express our deep appreciation to our Head of Department, Prof. (Dr.) V. K. Dabhi, for fostering a culture of innovation and continuous learning, and for his constant encouragement and support.

Lastly, we are grateful to all the faculty members and staff of the Information Technology Department for their guidance, as well as to our peers and classmates, whose feedback and motivation were instrumental throughout this journey.

Pranjal Sunil Yadav (IT115)

Megh Nitesh Sharma (IT130)

B. Tech Semester VI
Department of Information Technology
Dharmsinh Desai University

TABLE OF CONTENTS

DETAILS OF CHAPTERS

ABSTRACT

Lyceum is a modern online student community platform that simulates the experience of a virtual college campus. It is designed to facilitate peer-to-peer interaction, collaborative study environments, and community engagement. Unlike traditional learning platforms, Lyceum focuses on student-driven connection through features such as real-time study rooms, direct messaging, and open discussion spaces. The platform enables students to interact, support each other, and study together in a shared digital environment.

This documentation provides a detailed overview of the Lyceum platform, covering its core architecture, key features, implementation details, and user interaction guidelines.

Lyceum addresses the increasing demand for socially engaging academic environments by offering a set of tools that enable real-time collaboration, peer-to-peer learning, and community interaction. Built with modern web technologies such as React.js, Node.js, Peer JS, Socket.io, and MongoDB, the platform ensures a responsive, scalable, and secure user experience for students from diverse educational backgrounds.

The documentation serves as a complete reference for understanding the Lyceum project, from its conceptual design to its technical implementation and planned enhancements. It is intended for developers, contributors, academic institutions, and other stakeholders seeking in-depth insights into the platform's structure, functionality, and usage.

LIST OF FIGURES

Chapter 1: Introduction

1.1 Project Overview

Lyceum is a modern online student community platform that simulates the experience of a virtual college campus. It is designed to facilitate peer-to-peer interaction, collaborative study environments, and community engagement. Unlike traditional learning platforms, Lyceum focuses on student-driven connection through features such as real-time study rooms, direct messaging, and open discussion spaces.

The platform provides:

- **Real-time study rooms**: Students can join or create video-enabled rooms for collaborative study sessions.
- **Secure user authentication**: Using JWT and Google OAuth for account access and identity management.
- **Instant messaging**: Enables students to chat one-on-one or in small groups in real-time.
- **Open forum spaces**: For topic-based discussions and idea exchange.
- **User profiles and connection management**: Users can view profiles, send/accept friend requests, and manage social interactions.

Lyceum is built to ensure accessibility, performance, and seamless user experience across devices and platforms. Its interactive features emulate real-life social learning environments to foster community and academic support.

1.2 Project Objectives The primary goals of the Lyceum project include:

- **Foster peer-to-peer collaboration**: Enable students to connect, share, and study in real-time.
- **Create an interactive digital campus**: Provide a virtual space that reflects the social dynamics of a physical educational institution.
- **Enhance student engagement**: Use modern UI/UX design to encourage consistent and intuitive platform usage.
- **Ensure scalability and responsiveness**: Develop a system that supports high traffic and concurrent sessions smoothly.
- **Implement secure access and privacy controls**: Ensure safe student interactions through robust authentication and authorization mechanisms.

1.3 Scope Lyceum is a full-stack web platform with real-time communication and social interaction capabilities.

Frontend Technologies:

- **React.js**: For building dynamic and modular UI components.
- **Tailwind CSS**: To implement utility-first and responsive styling.

- **Socket.io-client**: For real-time chat and notifications.
- **Radix UI**: To provide accessible and customizable UI components.
- **PeerJS**: For WebRTC-based video chat integration in study rooms.

Backend Technologies:

- **Node.js**: Server-side JavaScript runtime.
- **Express.js**: REST API development and server routing.
- **Socket.io**: For real-time, bidirectional communication.
- **MongoDB**: Primary NoSQL database for storing users, messages, rooms, and friendships.
- **JWT and Google OAuth**: Authentication and session management.

Beyond functional features, the scope includes scalability considerations, secure handling of personal data, and consistent cross-device support.

Chapter 2: Project Management

2.1 Development Methodology

The Lyceum project follows an Agile development methodology, which emphasizes iterative development, continuous feedback, and adaptive planning. This approach is particularly well-suited for a complex educational platform like Lyceum, as it allows for:

- **Sprint-based development cycles**: The project is divided into time-boxed iterations (typically 2-4 weeks), during which specific features and improvements are developed and tested.
- **Continuous Integration/Continuous Deployment (CI/CD)**: Automated processes ensure that code changes are regularly integrated, tested, and deployed, reducing the risk of integration issues and enabling rapid delivery of new features.
- **Version control using Git**: All code changes are tracked, allowing for easy collaboration, code review, and rollback if necessary.

The Agile methodology enables the development team to respond quickly to changing requirements and user feedback, ensuring that the platform evolves in line with user needs and expectations.

2.2 Team Structure

The Lyceum project was developed by a two-member team with clearly defined roles based on technical specialization:

- **UI/UX & Frontend Developer:**
  Responsible for designing and implementing the user interface using React.js and Tailwind CSS, ensuring a responsive, accessible, and intuitive user experience. This role also included working on real-time client-side features using `socket.io-client`, UI interactions, and layout consistency across devices.

- **Backend Developer:**
  Focused on implementing the server-side architecture using Node.js, Express.js, and MongoDB. This role covered API development, database schema design, authentication and authorization (Google OAuth & JWT), real-time server-side logic using `Socket.io`, and general application logic.

Both developers shared responsibilities for **DevOps** and **Quality Assurance**, including:

- Setting up CI/CD pipelines for streamlined deployment.
- Manual and automated testing.
- Debugging and issue resolution.
- Deployment to production environments and ongoing monitoring.

This lean team structure allowed for tight collaboration, fast iterations, and a shared vision throughout the project, ensuring that both frontend and backend components were developed in harmony.

2.3 Timeline and Milestones

The development of the Lyceum platform follows a structured timeline with key milestones based on weekly iterations:

1. **Research and Initial CRUD Setup (Week 1):**
   Lay the foundation by conducting research and implementing basic Create, Read, Update, Delete functionalities.

2. **Minimum Viable Product – Video Chat System (Weeks 2–3):**
   Develop core video chat functionality using PeerJS and WebRTC to enable real-time study rooms.

3. **UI Enhancements and CI/CD Pipeline Integration (Week 4):**
   Improve the platform's interface and set up continuous integration/continuous deployment pipelines for faster, automated updates.

4. **Forum Feature Implementation (Week 5):**
   Add open discussion spaces to facilitate community-driven conversations and topic-based interactions.

5. **User and Friendship Management (Week 6):**
   Introduce user profile features, friend requests, and the ability to manage personal connections on the platform.

6. **Direct Messaging System (Weeks 7–8):**
   Enable one-on-one and group chat functionality to promote peer-to-peer engagement beyond study rooms.

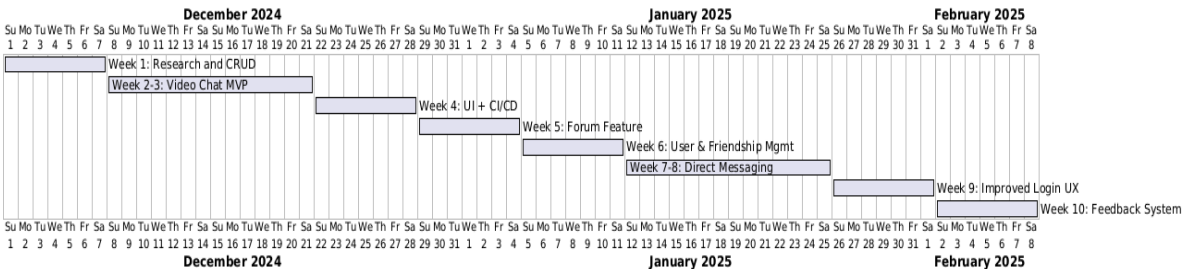
7. **Enhanced Login System UX (Week 9):**
   Improve the user experience of the authentication system, ensuring smoother and more secure sign-ins.

8. **Feedback System Development (Week 10):**
   Implement a feedback mechanism for users to share their experience, report issues, and suggest improvements.

This timeline provides a clear roadmap for the project, ensuring that all stakeholders are aligned on expectations and progress.

2.4 Project Scheduling

A Gantt chart is used to visualize project progress, ensuring efficient time management:



Chapter 3: System Requirements Study

3.1 Functional Requirements

The Lyceum platform must fulfill the following functional requirements:

1. **User Authentication and Authorization**
   - Users must be able to register and log in using email/password or Google OAuth.
   - The system must support role-based access control, distinguishing between students, educators, and administrators.
   - Users must be able to reset their passwords and manage their account settings.
2. **Real-time Messaging**
   - Users must be able to send and receive instant messages in real-time.
   - The system must support group chats and direct messaging.
   - Messages must be stored securely and accessible only to authorized users.
3. **Video Conferencing**
   - Users must be able to initiate and join video calls with other users.
   - The system must support screen sharing and recording capabilities.
   - Video calls must be secure and protected from unauthorized access.
4. **Content Management**
   - Users must be able to create, edit, and organize educational content.
   - The system must support various content types, including text, images, videos, and documents.

- Content must be searchable and accessible to authorized users.

5. **User Profile Management**

   - Users must be able to create and customize their profiles.
   - Profiles must display relevant information, such as interests, expertise, and activity.
   - Users must be able to control the privacy of their profile information.

3.2 Non-Functional Requirements

In addition to functional requirements, the Lyceum platform must meet the following non-functional requirements:

➔ **Performance**

- The platform must load quickly and respond promptly to user interactions.
- Real-time features must operate with minimal latency.
- The system must handle a large number of concurrent users without degradation in performance.

➔ **Security**

- All data must be encrypted in transit and at rest.
- The system must protect against common security threats, such as SQL injection, XSS, and CSRF.
- User authentication must be robust and resistant to brute force attacks.

➔ **Scalability**

- The platform must be able to accommodate a growing user base and increasing data volume.
- The architecture must support horizontal scaling to handle increased load.
- Database design must optimize for performance and scalability.

➔ **Reliability**

- The system must be available 99.9% of the time, with minimal downtime for maintenance.
- Data must be backed up regularly to prevent loss in case of system failure.
- The platform must gracefully handle errors and provide meaningful feedback to users.

➔ **Usability**

- The user interface must be intuitive and require minimal training.
- The platform must be accessible to users with disabilities, following WCAG guidelines.
- The system must provide helpful feedback and guidance to users.

3.3 Technical Requirements

The Lyceum platform is built using the following technical stack:

Frontend:

- **React.js**: A JavaScript library for building user interfaces, providing a component-based architecture and efficient rendering.
- **Tailwind CSS**: A utility-first CSS framework for rapid UI development, ensuring responsive and consistent design.
- **Socket.io-client**: A library for real-time, bidirectional communication between the browser and server.
- **Various UI components from Radix UI**: A collection of accessible and customizable UI components, enhancing the user experience.

Backend:

- **Node.js**: A JavaScript runtime for server-side applications, providing high performance and scalability.
- **Express.js**: A web application framework for Node.js, simplifying API development and routing.
- **MongoDB**: A NoSQL database for storing and retrieving data, offering flexibility and scalability.
- **Socket.io**: A library for real-time, bidirectional communication between the server and clients.
- **JWT Authentication**: A secure method for authenticating users and managing sessions.

This technical stack was chosen for its performance, scalability, and developer-friendly features, enabling the rapid development of a robust and feature-rich platform.

Chapter 4: System Analysis

4.1 System Architecture

The Lyceum platform is designed using a monolithic-but-modular architecture enhanced by real-time communication and peer connectivity features. It integrates modern web technologies to provide a seamless virtual academic experience.

4.1.1 Client-Server Architecture

- The frontend is developed using **React.js**, providing a dynamic and responsive user interface.
- The backend is powered by **Node.js** with **Express.js**, serving REST APIs and handling WebSocket connections.
- Real-time peer-to-peer functionalities (study rooms, voice/video) are supported via **PeerJS** and **Socket.io**.
- **MongoDB** is used as the primary database, optimized for flexible, schema-less data storage.

4.1.2 RESTful API & Real-Time Integration

- The platform exposes REST APIs for operations like user management, discussions, and messaging.
- Real-time interactions (chat, presence, study room events) are handled using **Socket.io**, enabling low-latency bi-directional communication.
- **JWT** is used for authentication across both HTTP and WebSocket layers.

4.1.3 Peer-to-Peer Study Room Model

- Lyceum integrates **PeerJS** for direct P2P audio/video communication in study rooms.
- A central signaling server facilitates initial connections, after which media streams are routed directly between clients to reduce server load.

4.1.4 Scalability Consideration

- Though currently built monolithically, the architecture is modular enough to support future migration toward microservices for scalability.

4.2 Data Flow

4.2.1 User Authentication Flow

1. User initiates login via Email/Password or Google OAuth.
2. Server validates credentials and returns a **JWT**.
3. Token is stored on the client and attached to subsequent requests.
4. On each request, the server authenticates the token before granting access.

4.2.2 Real-Time Communication Flow

1. Client connects to the WebSocket server with a JWT token.
2. Server authenticates and registers the user in a live user map.
3. Messages and presence updates are broadcasted to relevant clients.
4. On disconnect, users are removed and their online status is updated.

4.2.3 Study Room Session Flow

1. A student creates or joins a study room.
2. PeerJS establishes a direct connection between clients.
3. Signaling and connection metadata are exchanged via the server.
4. Audio/video streams flow directly through P2P, with minimal server involvement.

4.2.4 File Handling Flow

1. Users upload files via the client interface.
2. Files are validated and stored on the server or cloud storage (e.g., AWS/GCS).
3. File references and metadata are stored in MongoDB.
4. Users with access can retrieve or download the file.

4.2.5 Data Persistence Flow

1. Actions (messages, profile updates, posts) trigger API calls.
2. Backend validates and persists data to MongoDB.
3. Server emits updates in real-time (if relevant), and client UIs refresh accordingly.

4.3 Security Analysis

4.3.1 Authentication and Authorization

- **JWT Authentication**: All protected routes require valid JWTs.
- **Google OAuth 2.0**: Secure identity management through Google's platform.
- **Ownership-Based Access:** Access to resources like study rooms and chats is determined by user ownership or invitation, rather than predefined roles.

4.3.2 Data Protection

- **HTTPS (TLS Encryption)**: Ensures secure communication between client and server.
- **bcrypt Password Hashing**: Used to securely hash passwords.
- **Access Control**: Resources (e.g., rooms, chats) are user-owned or permission-based.

4.3.3 WebSocket & Peer Security

- WebSocket connections are authenticated with JWTs before being accepted.
- PeerJS connections are monitored for unauthorized access and abuse.
- Disconnected sessions are cleaned up to prevent ghost peers.

4.3.4 Session and Rate Management

- **Session Expiry**: Tokens expire and require re-authentication.
- **Rate Limiting**: Critical API endpoints are protected against abuse via request throttling.
- **Activity Tracking**: Online status, room presence, and session metadata are logged for security and UX features.

Chapter 5: System Design

5.1 Database Design

The Lyceum platform uses MongoDB, a NoSQL database, with the following collections:

User Collection

- **Fields**:
  - _id: Unique identifier for the user.
  - username: Unique username for the user.
  - email: User's email address, used for authentication and communication.
  - password: Hashed password for email/password authentication.
  - googleId: ID from Google for OAuth authentication.
  - profileImage: URL to the user's profile image.
  - coverImage: URL to the user's cover image.
  - isOnline: Boolean indicating whether the user is currently online.
  - lastSeen: Timestamp of the user's last activity.
  - friendRequests: Array of pending friend requests.
  - friends: Array of user IDs representing the user's friends.
  - visitorCount: Number of profile visitors.
  - createdAt: Timestamp of when the user was created.
  - updatedAt: Timestamp of when the user was last updated.

Messages Collection

- **Fields**:
  - _id: Unique identifier for the message.
  - sender: ID of the user who sent the message.
  - recipient: ID of the user who received the message.
  - content: Content of the message.
  - timestamp: Timestamp of when the message was sent.
  - ead: Boolean indicating whether the message has been read.
  - attachments: Array of file IDs attached to the message.

Files Collection

- **Fields**:
  - _id: Unique identifier for the file.
  - filename: Original name of the file.
  - path: Path to the file in storage.
  - type: MIME type of the file.

- ➢ size: Size of the file in bytes.
- ➢ uploadedBy: ID of the user who uploaded the file.
- ➢ uploadedAt: Timestamp of when the file was uploaded.
- ➢ sharedWith: Array of user IDs the file is shared with.

Sessions Collection

- **Fields**:
  - ➢ _id: Unique identifier for the session.
  - ➢ userId: ID of the user associated with the session.
  - ➢ token: JWT token for the session.
  - ➢ createdAt: Timestamp of when the session was created.
  - ➢ expiresAt: Timestamp of when the session expires.
  - ➢ lastActivity: Timestamp of the user's last activity in the session.

5.2 API Design

The Lyceum platform exposes a RESTful API with the following endpoints:

Authentication Endpoints

- POST /api/auth/register: Register a new user.
- POST /api/auth/login: Log in an existing user.
- POST /api/auth/google: Authenticate using Google OAuth.
- GET /api/auth/logout: Log out the current user.
- POST /api/auth/reset-password: Request a password reset.
- PUT /api/auth/reset-password/:token: Reset the password using a token.

User Endpoints

- GET /api/user/profile: Get the current user's profile.
- PUT /api/user/profile: Update the current user's profile.
- GET /api/user/:username: Get a user's profile by username.
- GET /api/user/friends: Get the current user's friends.
- POST /api/user/friends/request: Send a friend request.
- PUT /api/user/friends/accept: Accept a friend request.
- PUT /api/user/friends/reject: Reject a friend request.
- DELETE /api/user/friends/:friendId: Remove a friend.

Messaging Endpoints

- GET /api/chat/messages: Get messages for the current user.
- POST /api/chat/messages: Send a new message.
- PUT /api/chat/messages/:messageId/read: Mark a message as read.
- DELETE /api/chat/messages/:messageId: Delete a message.

Topic and Post Endpoints

- GET /api/topics: Get all topics.
- POST /api/topics: Create a new topic.
- GET /api/topics/:topicId: Get a specific topic.
- PUT /api/topics/:topicId: Update a topic.
- DELETE /api/topics/:topicId: Delete a topic.
- GET /api/posts: Get all posts.
- POST /api/posts: Create a new post.
- GET /api/posts/:postId: Get a specific post.
- PUT /api/posts/:postId: Update a post.
- DELETE /api/posts/:postId: Delete a post.

5.3 UI/UX Design

The Lyceum platform follows modern UI/UX design principles:

Responsive Design

- The platform is designed to work seamlessly across different devices, from desktop computers to mobile phones.
- The layout adapts to the screen size, ensuring a consistent experience regardless of the device.
- Touch-friendly elements are used for mobile devices, enhancing usability.

Component-Based Architecture

- The UI is built using React components, promoting reusability and maintainability.

- Components are organized in a hierarchical structure, with parent components managing state and child components handling presentation.
- Custom hooks are used to encapsulate logic and promote separation of concerns.

Accessibility Considerations

- The platform follows WCAG guidelines, ensuring accessibility for users with disabilities.
- Semantic HTML is used to provide meaningful structure to the content.
- ARIA attributes are added to interactive elements, enhancing screen reader compatibility.
- Color contrast is carefully considered to ensure readability for users with visual impairments.

User Interface Guidelines

- A consistent color scheme and typography are used throughout the platform, creating a cohesive visual identity.
- Interactive elements, such as buttons and links, have clear hover and focus states, providing feedback to users.
- Loading states and error messages are designed to be informative and user-friendly.
- Animations are used sparingly and purposefully, enhancing the user experience without being distracting.

5.4 Use Case Diagram

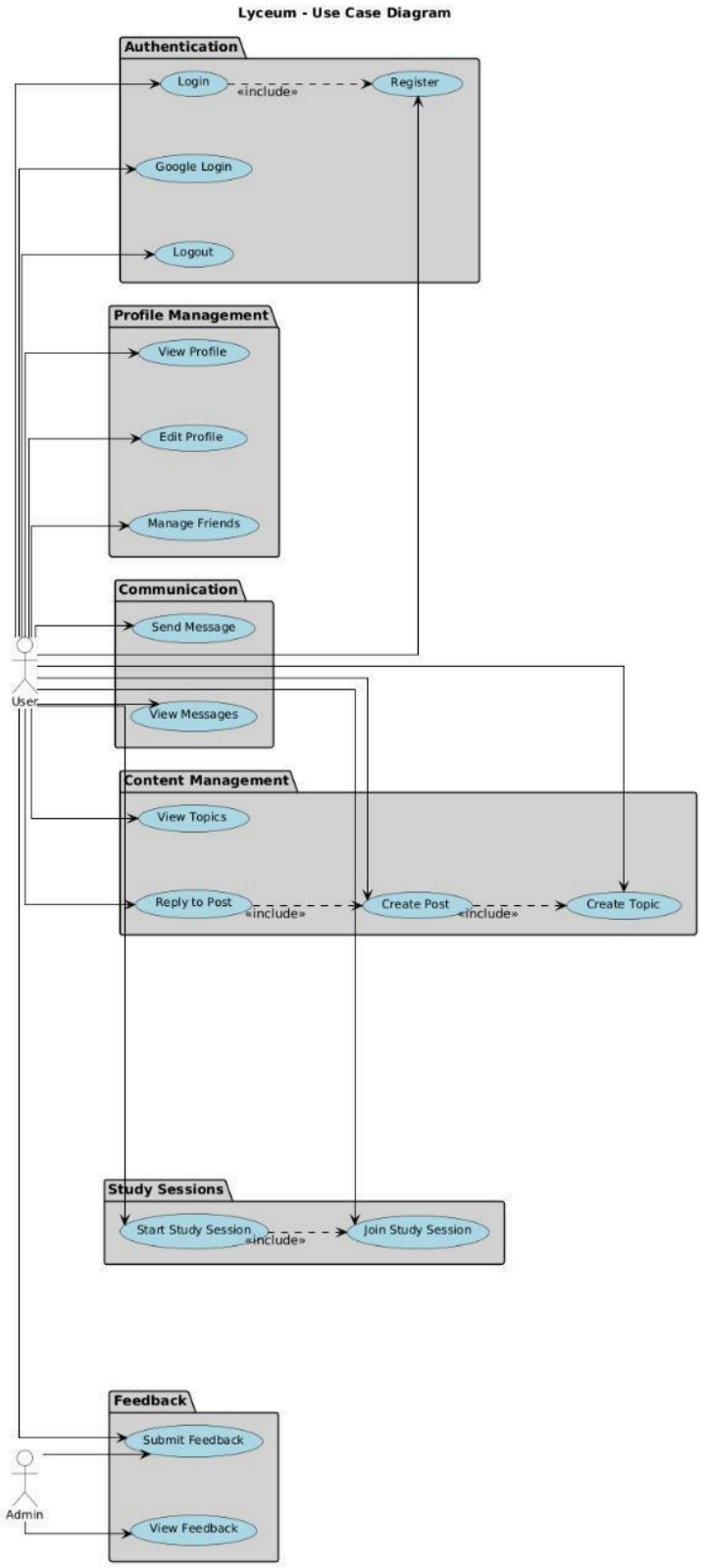


Fig 5.1

5.5 Class Diagram

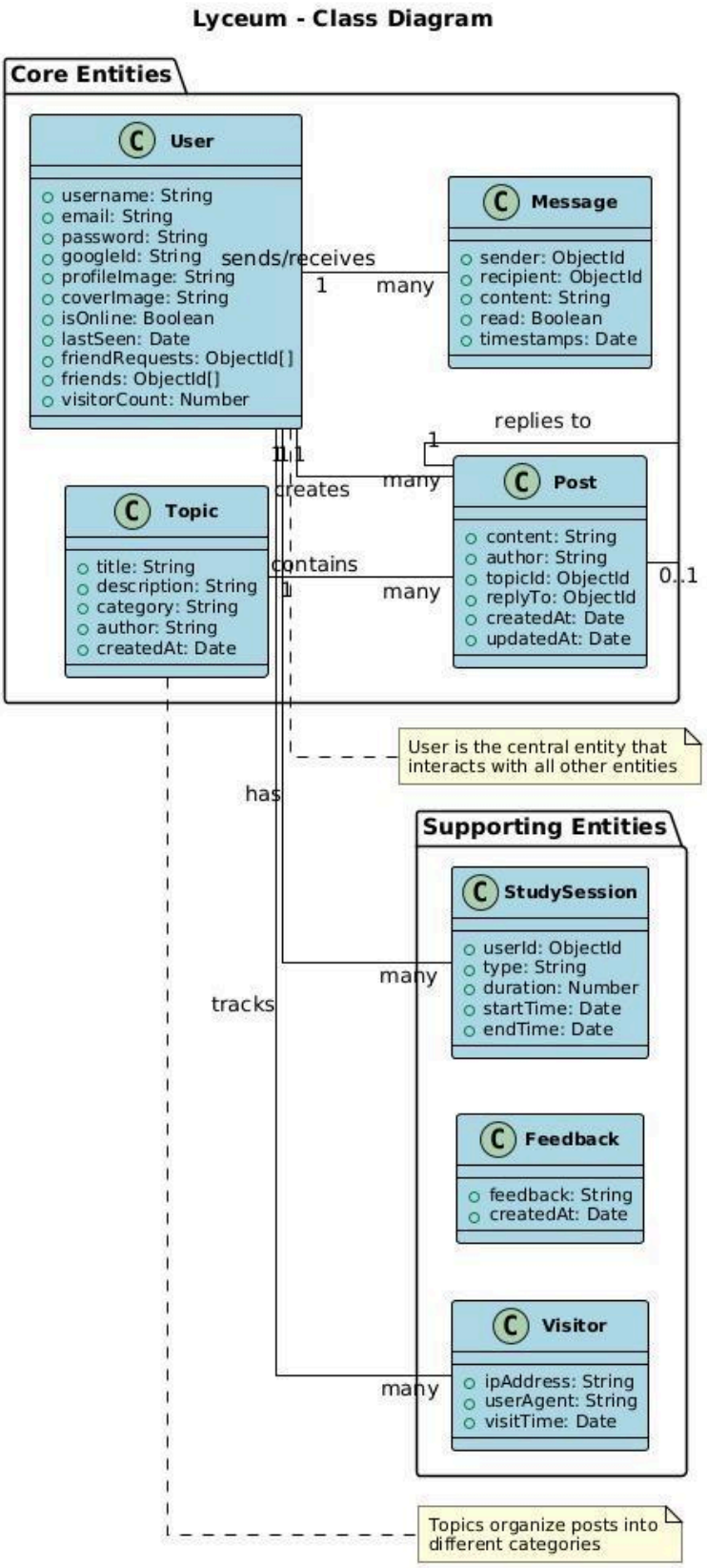


Fig 5.2

5.6 Sequence Diagram

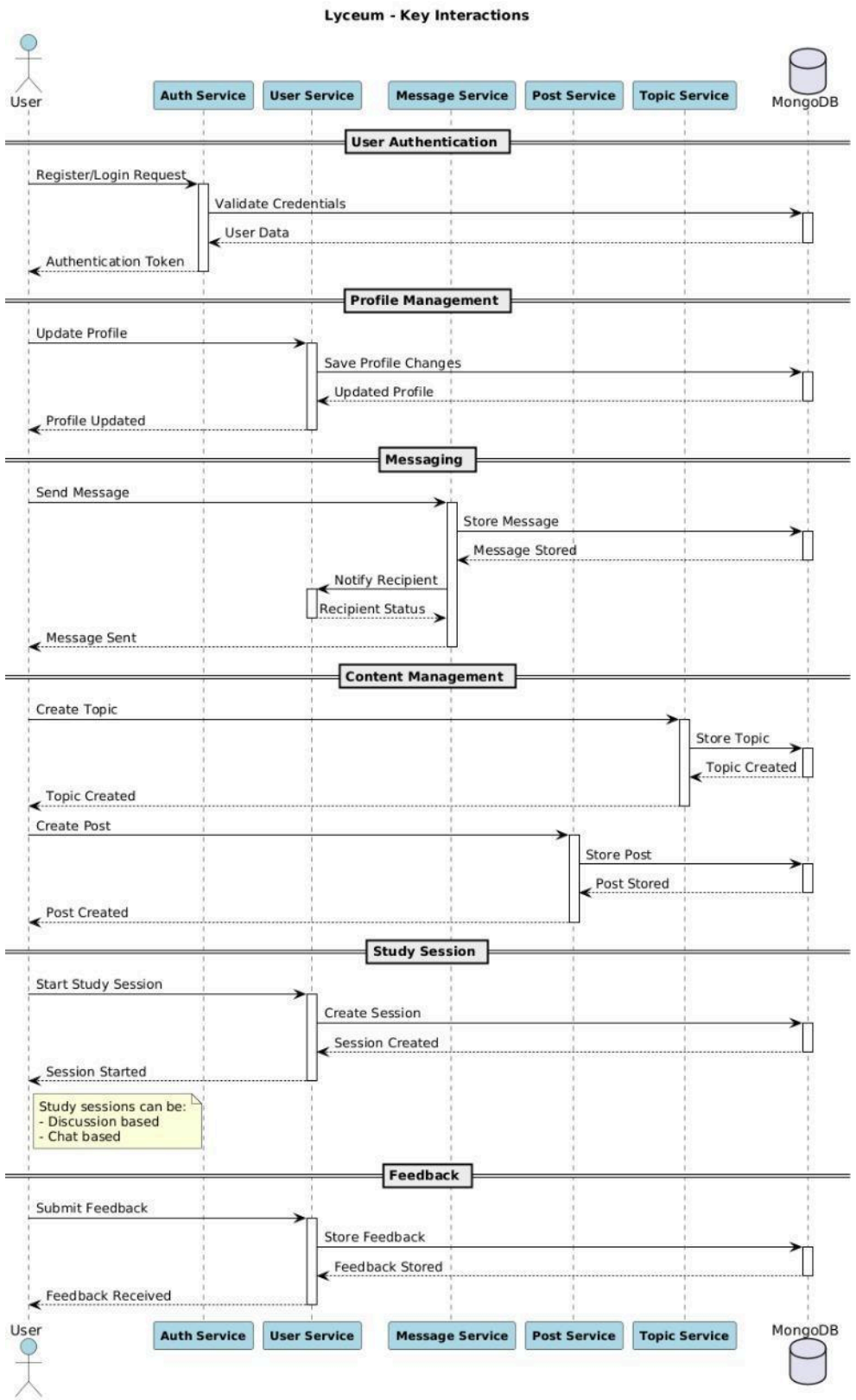


Fig 5.3

5.7 Activity Diagram

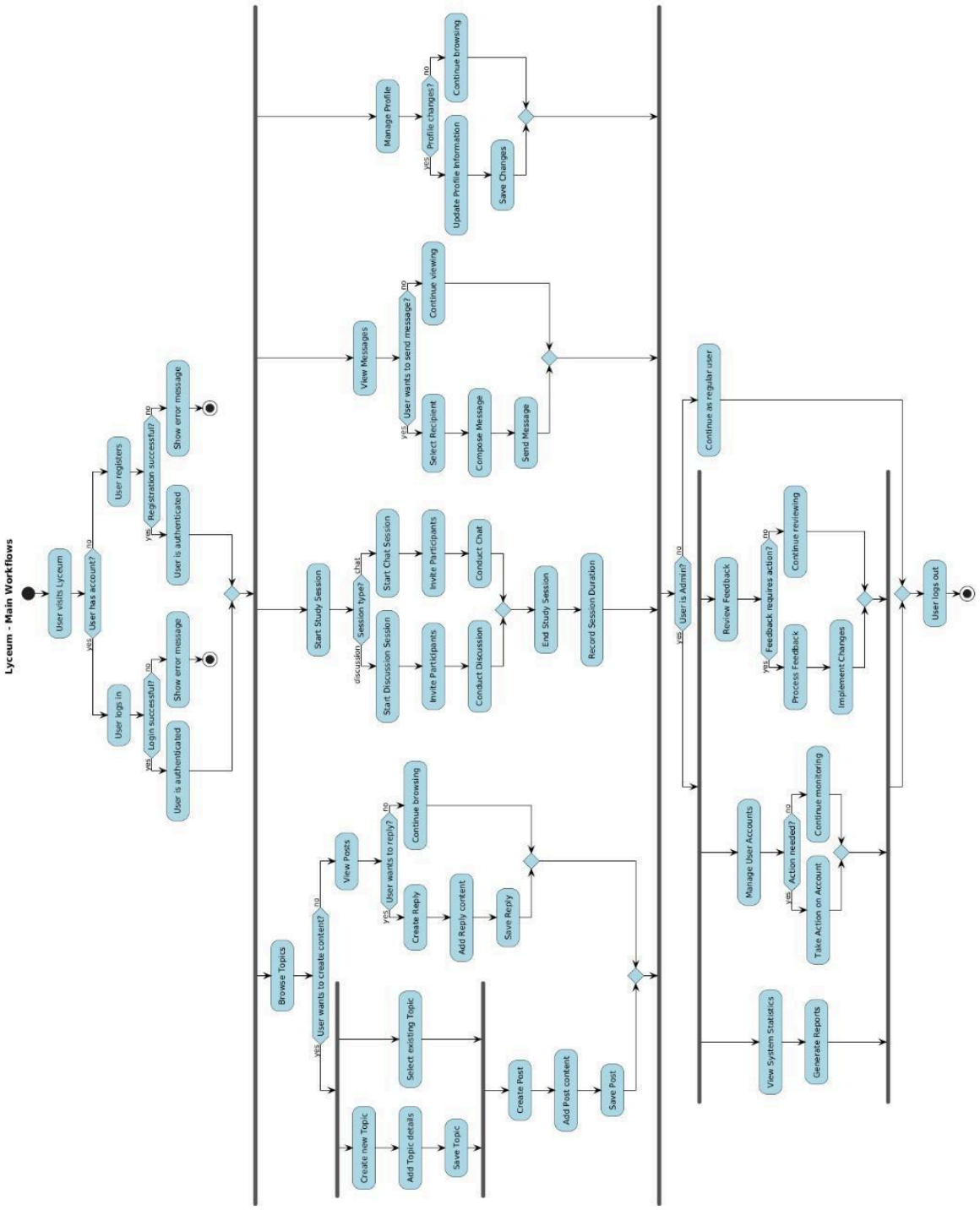


Fig 5.4

Chapter 6: Implementation Environment

6.1 Development Tools

The Lyceum project utilizes the following development tools:

VS Code

- A powerful code editor with a rich ecosystem of extensions.
- Extensions used include:
  - ESLint for JavaScript linting
  - Prettier for code formatting
  - MongoDB for database management
  - GitLens for enhanced Git integration
  - React Developer Tools for React debugging

Git

- A distributed version control system for tracking code changes.
- Used for:
  - Managing code versions
  - Collaborating with team members
  - Reviewing code changes
  - Branching and merging features
  - Deploying code to different environments

npm

- A package manager for JavaScript, used to install and manage dependencies.
- Key commands:
  - npm install: Install dependencies
  - npm start: Start the development server
  - npm build: Build the application for production
  - npm test: Run tests

- npm run lint: Lint the code

MongoDB Compass

- A graphical user interface for MongoDB, used to:
  - View and edit database collections
  - Execute queries
  - Monitor database performance
  - Manage indexes
  - Import and export data

6.2 Deployment Environment

The Lyceum platform is deployed using the following infrastructure:

Vercel for Frontend

- A platform for deploying frontend applications.
- Features:
  - Automatic deployments from Git repositories
  - Preview deployments for pull requests
  - Edge network for fast content delivery
  - Environment variable management
  - Analytics and monitoring

Vercel for Backend

- The same platform is used for deploying the backend API.
- Benefits:
  - Serverless functions for API endpoints
  - Automatic scaling based on demand
  - Global edge network for low latency
  - Integrated with frontend deployment
  - Environment variable management

MongoDB Atlas

- A cloud-based MongoDB service.
- Features:
  - Managed database clusters
  - Automatic backups and point-in-time recovery
  - Monitoring and alerting
  - Security features, such as IP whitelisting and encryption
  - Scalability options

Environment Variables Management

- Environment variables are used to configure the application for different environments.
- Variables include:
  - Database connection strings
  - API keys and secrets
  - Feature flags
  - External service URLs
  - Logging levels

6.3 Dependencies

The Lyceum platform relies on the following key dependencies:

Frontend Dependencies

- **React 18.3.1**: A JavaScript library for building user interfaces.
  - Used for component-based UI development
  - Manages application state and lifecycle
  - Provides hooks for functional components
- **Socket.io-client 4.8.1**: A library for real-time, bidirectional communication.
  - Enables real-time chat and notifications
  - Handles connection management and reconnection

  - Provides event-based communication

- **Tailwind CSS 3.4.17**: A utility-first CSS framework.
  - Provides pre-built utility classes for styling
  - Ensures responsive design
  - Enables custom theme configuration
- **Various UI libraries**:
  - Radix UI for accessible components
  - Headless UI for unstyled components
  - Heroicons for icons
  - Framer Motion for animations

Backend Dependencies

- **Express 4.17.1**: A web application framework for Node.js.
  - Handles HTTP requests and responses
  - Provides middleware for common tasks
  - Enables routing and error handling
- **Socket.io 4.8.1**: A library for real-time, bidirectional communication.
  - Manages WebSocket connections
  - Handles room-based communication
  - Provides fallback mechanisms for unsupported browsers
- **MongoDB 6.14.2**: A NoSQL database.
  - Stores application data
  - Provides query and aggregation capabilities
  - Enables indexing for performance optimization
- **Authentication libraries**:
  - Passport for authentication strategies
  - JWT for token-based authentication
  - bcrypt for password hashing
  - Google OAuth for social login

Chapter 7: Testing

7.1 Testing Strategy

The Lyceum platform employs a comprehensive testing strategy to ensure quality and reliability:

Unit Testing

- Tests individual components and functions in isolation.
- Tools used:
  - Jest for JavaScript testing
  - React Testing Library for React component testing
  - Mocha and Chai for backend testing
- Coverage target: 80% of codebase

Integration Testing

- Tests the interaction between different parts of the system.
- Focus areas:
  - API endpoints and database interactions
  - Authentication flows
  - Real-time communication
  - File upload and download
- Tools used:
  - Supertest for API testing
  - Jest for test orchestration

End-to-End Testing

- Tests the entire application from the user's perspective.
- Scenarios covered:
  - User registration and login
  - Messaging and notifications
  - File sharing

  - Profile management
- Tools used:
  - Cypress for browser automation
  - Percy for visual regression testing

Performance Testing

- Tests the system's performance under various conditions.
- Metrics measured:
  - Response time
  - Throughput
  - Resource utilization
  - Error rates
- Tools used:
  - k6 for load testing
  - Lighthouse for frontend performance
  - MongoDB Compass for database performance

7.2 Test Cases

The following test cases are implemented to ensure the platform's functionality:

Authentication Flows

- User registration with email/password
- User login with email/password
- User login with Google OAuth
- Password reset
- Session management
- Token validation and expiration

Real-time Communication

- Sending and receiving messages
- Group chat functionality
- Online/offline status updates
- Typing indicators
- Message read receipts

User Interactions

- Profile creation and editing
- Friend requests and management
- Topic creation and participation
- Post creation and interaction
- Search functionality
- Notification preferences

7.3 Quality Assurance

Quality assurance is an ongoing process throughout the development lifecycle:

Code Review Process

- All code changes must be reviewed by at least one team member.
- Review checklist includes:
  - Code quality and style
  - Functionality and correctness
  - Performance considerations
  - Security implications
  - Test coverage
- Pull requests are used to facilitate code reviews.

Testing Automation

- Tests are automatically run on every code change.
- Continuous Integration (CI) pipeline:
  - Lint code
  - Run unit tests
  - Run integration tests
  - Build application
  - Deploy to staging environment
  - Run end-to-end tests
- Failed tests block merging of code changes.

Bug Tracking

- Bugs are tracked using a dedicated issue tracking system.
- Bug reports include:
  - Description of the issue
  - Steps to reproduce
  - Expected and actual behavior
  - Environment details
  - Screenshots or videos
- Bugs are prioritized based on severity and impact.

Performance Monitoring

- Application performance is continuously monitored.
- Metrics tracked:
  - Server response time
  - Client-side rendering time
  - Database query performance
  - API endpoint usage
  - Error rates and types
- Alerts are configured for performance degradation.

Chapter 8: User Manual

8.1 Getting Started

Account Creation

1. **Navigate to the Registration Page**:
   - Open your web browser and go to the Lyceum platform URL.
   - Click on the “Register” button on the login page.
2. **Fill in Your Information**:
   - Enter your desired username (must be unique).
   - Provide your email address.
   - Create a strong password (at least 8 characters, including uppercase, lowercase, numbers, and special characters).
   - Optionally, upload a profile picture.
3. **Complete Registration**:
   - Click the “Register” button to create your account.
   - Verify your email address by clicking the link sent to your inbox.
   - You will be automatically logged in after verification.

Login Process

1. **Access the Login Page**:
   - Go to the Lyceum platform URL.
   - If you’re not already logged in, you’ll be directed to the login page.
2. **Enter Your Credentials**:
   - Type your username or email address.
   - Enter your password.
   - Alternatively, click “Login with Google” to use your Google account.

3. **Access Your Account**:
   - Click the “Login” button.
   - Upon successful authentication, you’ll be redirected to the welcome page.

Profile Setup

1. **Access Your Profile**:
   - Click on your profile button in the sidebar..
2. **Customize Your Profile**:
   - Upload or change your profile picture.
   - Add a cover image.
   - Update your personal information.
   - Add your interests and expertise.
   - Set your privacy preferences.
3. **Save Changes**:
   - Click the “Save” button to update your profile.
   - Your changes will be immediately visible to other users.

Navigation Guide

1. **Main Navigation**:
   - The left sidebar contains links to main sections: Welcome, Chat, Forum, and Profile.
   - The top bar provides quick access to notifications, search, and settings.
2. **Welcome Page**:
   - Displays an overview of your activity and recent updates.
   - Provides quick access to key features.
3. **Chat Section**:
   - Lists your conversations and contacts.
   - Allows you to start new conversations and manage existing ones.

4. **Forum Section**:
   - Shows topics and posts from the community.
   - Enables you to create new topics and participate in discussions.
5. **Profile Section**:
   - Displays your profile information and activity.
   - Allows you to manage your account settings.

8.2 Features

Real-time Chat

1. **Starting a Conversation**:
   - Click on the “Chat” section in the sidebar.
   - Search for a user by username.
   - Select the user to start a conversation.
2. **Sending Messages**:
   - Type your message in the input field at the bottom of the chat window.
   - Press Enter or click the “Send” button to send the message.
   - Messages are delivered instantly to the recipient.
3. **Managing Conversations**:
   - View your conversation history in the left panel.
   - Search for specific messages within a conversation.

Video Conferencing

1. **Starting a Video Call**:
   - Open a chat with the user you want to call.
   - Click the video camera icon in the chat header.
   - Allow camera and microphone access when prompted.
   - The call will connect automatically.
2. **During a Call**:
   - Toggle your camera and microphone using the buttons at the bottom of the screen.
   - End the call by clicking the red "End Call" button.
3. **Group Video Calls**:
   - Start a group call from a group chat.
   - Manage participants using the sidebar.
   - Mute or remove participants as needed.

Content Management

1. **Creating Topics**:
   - Go to the "Forum" section.
   - Click "Create Topic."
   - Enter a title and description.
   - Select relevant categories or tags.
   - Click "Create" to publish the topic.
2. **Creating Posts**:
   - Navigate to a topic.
   - Click "Create Post."
   - Write your post content.
   - Add images, links, or attachments.
   - Click "Post" to publish.

3. **Managing Content**:
    - Edit or delete your own topics and posts.
    - Report inappropriate content.
    - Save favorite topics and posts.
    - Follow specific topics for updates.

8.3 Troubleshooting

Common Issues

1. **Login Problems**:
    - **Issue**: Unable to log in with correct credentials.
    - **Solution**:
        - Ensure your username/email and password are correct.
        - Try resetting your password.
        - Clear your browser cache and cookies.
        - Try logging in with a different browser.
2. **Chat Not Working**:
    - **Issue**: Messages not sending or receiving.
    - **Solution**:
        - Check your internet connection.
        - Refresh the page.
        - Log out and log back in.
        - Clear your browser cache.
3. **Video Call Issues**:
    - **Issue**: Poor video/audio quality or connection problems.
    - **Solution**:
        - Check your internet speed.
        - Ensure your camera and microphone are working.
        - Try closing other applications using your camera.
        - Restart your browser.

Error Messages

1. **“Invalid credentials”**:
   - Your username/email or password is incorrect.
   - Try again with the correct information.
2. **“User not found”**:
   - The username you’re searching for doesn’t exist.
   - Check for typos or try a different search term.
3. **“Connection failed”**:
   - There’s an issue with your internet connection.
   - Check your network settings and try again

8.3 Screenshots

1. Registration Page

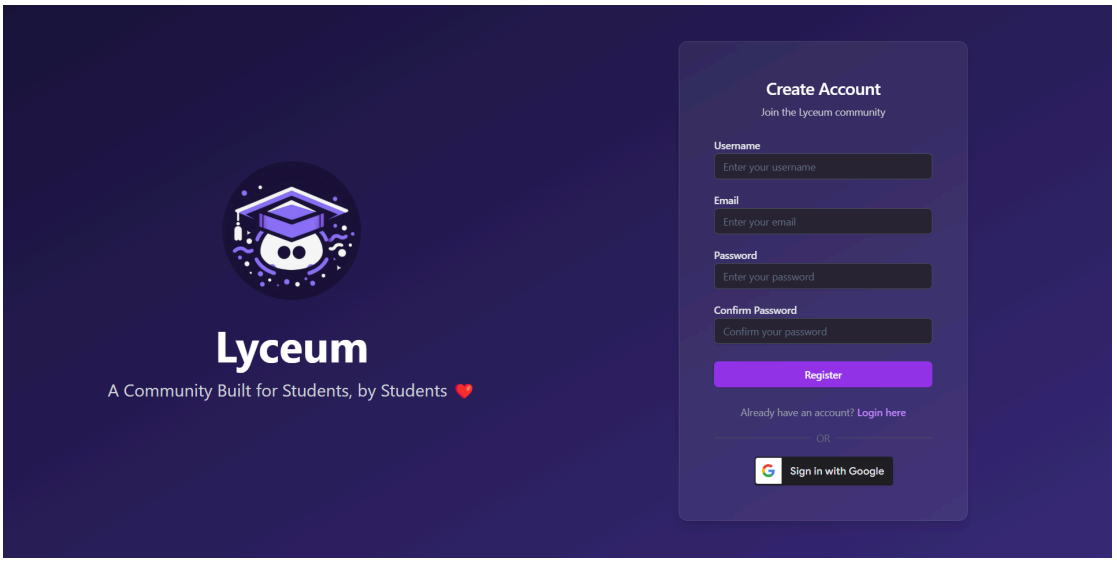


figure 8.1

2. Login Page

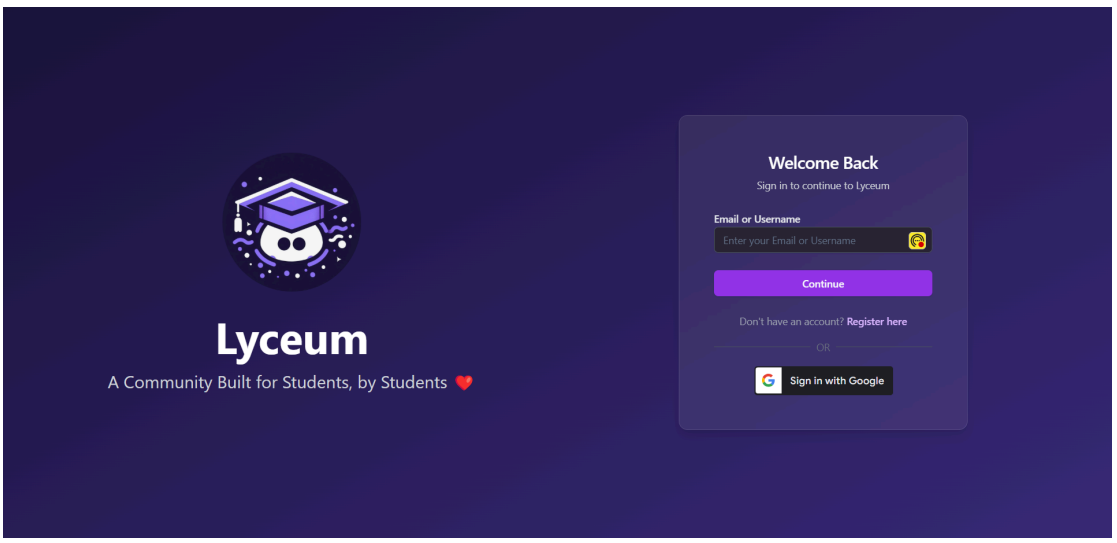


figure 8.2

3. Dashboard

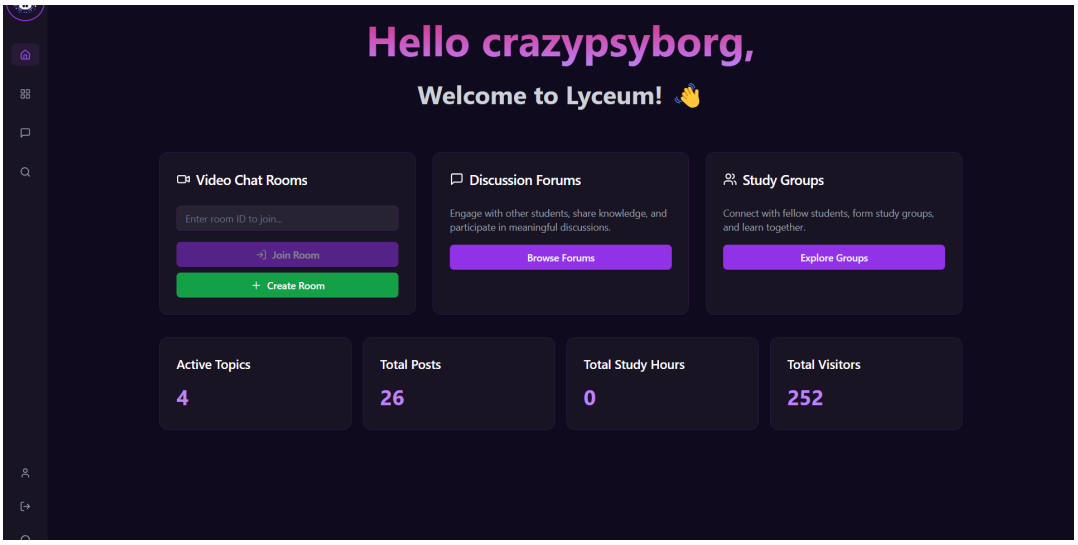


figure 8.3

4. Video Chat Page

figure 8.4

5. Forum Page

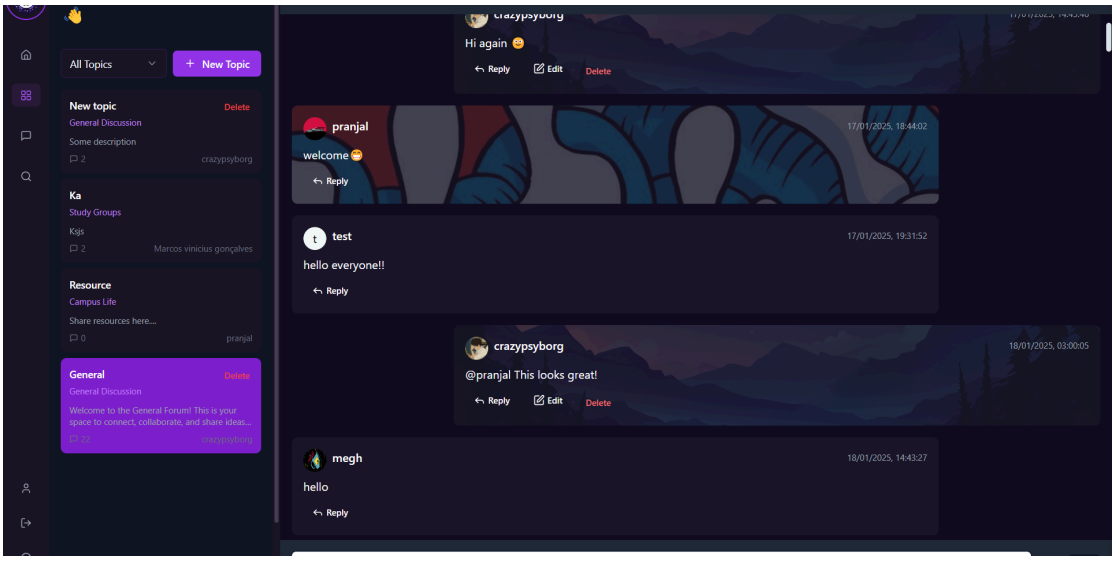


figure 8.5

6. Direct Messaging Section

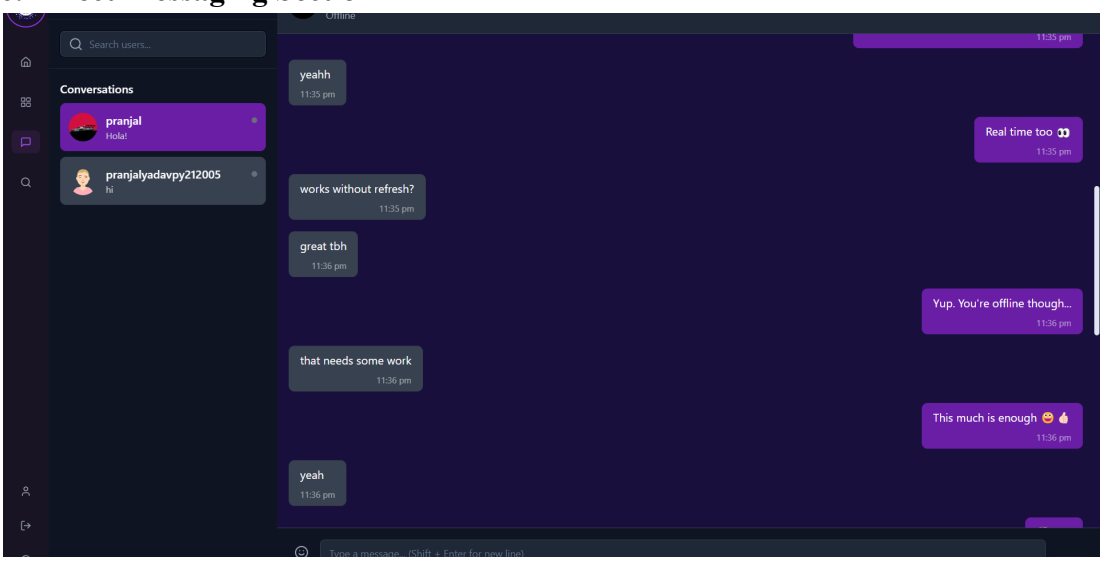


figure 8.6

7. User Profile Page

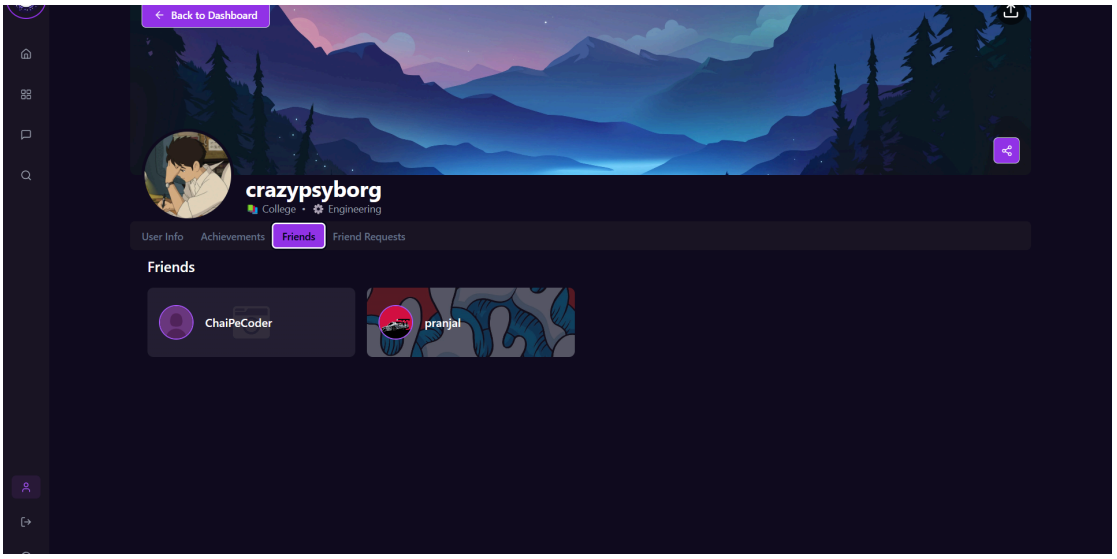


figure 8.7

8. Feedback Section

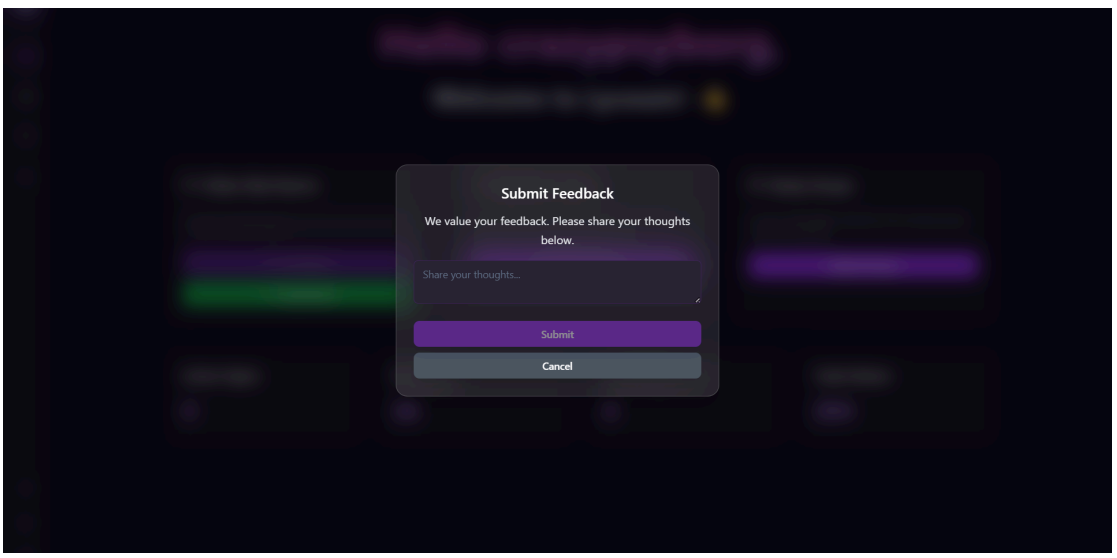


figure 8.8

Chapter 9: Limitations and Future Enhancements

9.1 Current Limitations

Browser Compatibility

- The platform may not function optimally on older browsers.
- Some features require modern browser capabilities.
- Mobile browsers may have limited functionality compared to desktop.

Network Dependencies

- Real-time features require a stable internet connection.
- Video calls may experience quality issues on slow connections.
- File uploads and downloads depend on network speed.

Resource Constraints

- Large images may be limited by server capacity.
- Video calls are limited to a maximum number of participants.
- Database queries may be slow with large datasets.

Feature Limitations

- Offline mode is not fully supported.
- Advanced analytics are not available for all users.
- Integration with external educational tools is limited.

9.2 Planned Enhancements

Mobile Application

- Develop native mobile apps for iOS and Android.
- Implement push notifications for important updates.
- Optimize the user experience for mobile devices.
- Enable offline access to certain features.

Advanced Analytics

- Track student engagement and progress.
- Generate reports on platform usage and effectiveness.
- Offer personalized recommendations based on user behavior.

AI Integration

- Implement AI-powered content recommendations.
- Add intelligent tutoring features.
- Enable automated content moderation.
- Provide language translation for international users.

Enhanced Security Features

- Implement two-factor authentication.
- Add end-to-end encryption for messages.
- Enhance data privacy controls.
- Provide more granular permission settings.

Chapter 10: Conclusion and Discussion

10.1 Project Achievements

Successful Implementation of Core Features

- Real-time communication tools are functioning effectively.
- User authentication and authorization are secure and reliable.
- Content management system supports various educational resources.
- File sharing capabilities enable seamless resource distribution.

Robust Architecture

- Microservices architecture provides scalability and maintainability.
- API design follows RESTful principles for clarity and consistency.
- Database design optimizes for performance and data integrity.
- Security measures protect user data and privacy.

Scalable Design

- The platform can accommodate a growing user base.
- Resources scale automatically based on demand.
- New features can be added without disrupting existing functionality.
- Performance remains consistent under increased load.

User-friendly Interface

- Intuitive navigation requires minimal training.
- Responsive design works across different devices.
- Accessibility features ensure inclusivity.
- Visual design is clean and modern.

10.2 Lessons Learned

Technical Insights

- Real-time communication requires careful consideration of scalability.
- Authentication systems must balance security with user convenience.
- Database design significantly impacts application performance.
- Frontend state management is crucial for complex applications.

Development Challenges

- Coordinating real-time features across different clients was complex.
- Managing WebSocket connections required robust error handling.
- Balancing feature richness with performance was challenging.
- Ensuring consistent user experience across devices required extensive testing.

Best Practices

- Component-based architecture improves code reusability.
- Comprehensive testing reduces bugs and improves reliability.
- Regular code reviews enhance code quality and knowledge sharing.
- Continuous integration and deployment streamline the development process.

Improvement Areas

- Documentation could be more comprehensive from the start.
- Testing coverage could be increased for certain components.
- Performance optimization could be more proactive.
- User feedback could be incorporated more frequently.

10.3 Future Outlook

Expansion Plans

- Increase the number of supported languages.
- Expand to additional educational institutions.
- Develop specialized features for different academic disciplines.
- Create a marketplace for educational resources.

Feature Roadmap

- Implement advanced collaboration tools.
- Add virtual classrooms with interactive whiteboards.
- Develop gamification elements to increase engagement.
- Create an API for third-party integrations.

Technology Evolution

- Explore blockchain for credential verification.
- Investigate augmented reality for immersive learning.
- Evaluate machine learning for personalized education.
- Consider edge computing for improved performance.

Market Positioning

- Position Lyceum as a leading educational technology platform.
- Develop partnerships with educational institutions.
- Create a community of educators and learners.
- Establish Lyceum as a standard for online education.

REFERENCES

1. **Study Together Website - https://studytogether.com/**
2. **Studyverse Website (Project Sunsetted):**

   previously: https://studyverse.live

   archive available at:
   https://web.archive.org/web/20240429031336/https://studyverse.live/