

Product Design

Team: **Varun Gupta, Virat Garg, Keerthi Vempati, Prajas Wadekar, Vishnu Sathwik.**

Design Model:

Class No.	Class Name	Class State	Class Behaviour
1.	User	<ul style="list-style-type: none">• Maintains user identity information (id, name, email, password)• Tracks the role of the user (SuperAdmin, Admin, Teacher, or Student)• Stores authentication and profile data for all users in the system	<ul style="list-style-type: none">• Allows users to register for an account on the platform• Provides authentication functionality (login)• Enables users to update their profile information• Supports password reset functionality• Returns the role of the user when requested
2.	SuperAdmin	<ul style="list-style-type: none">• Inherits all state from the User class• Does not maintain additional state of its own	<ul style="list-style-type: none">• Creates and manages organizations across the platform• Adds and removes admin users for specific organizations• Monitors overall system performance• Retrieves a list of all organizations in the system
3.	Admin	<ul style="list-style-type: none">• Inherits all state from the User class	<ul style="list-style-type: none">• Adds and removes teachers from their


		<ul style="list-style-type: none"> • Maintains a reference to the organization the admin manages (organizationId) 	<ul style="list-style-type: none"> organization • Creates courses within their organization • Assigns teachers to specific courses • Deletes courses when they are no longer needed • Modifies course details as necessary • Retrieves lists of teachers in their organization • Gets and updates organization details
4.	Teacher	<ul style="list-style-type: none"> • Inherits all state from the User class • Does not maintain additional state of its own 	<ul style="list-style-type: none"> • Adds and manages course content (videos, PDFs, texts, etc.) • Creates assignments for courses they teach • Reviews student submissions and provides feedback • Approves or rejects student enrollment requests • Extends assignment deadlines when needed • Retrieves courses they are responsible for • Tracks student progress in their courses
5.	Student	<ul style="list-style-type: none"> • Inherits all state from the User class • Does not maintain 	<ul style="list-style-type: none"> • Requests enrollment in courses • Views courses they

		additional state of its own	<p>are enrolled in</p> <ul style="list-style-type: none"> • Accesses course content (videos, PDFs, etc.) • Submits assignments • Requests deadline extensions for assignments • Provides feedback on course content • Tracks their progress through enrolled courses
6.	Organization	<ul style="list-style-type: none"> • Stores organization identity information (id, name) • Maintains descriptive content about the organization (description) • Holds a reference to the organization's logo image 	<ul style="list-style-type: none"> • Provides detailed information about the organization • Allows updates to organization details • Returns lists of users associated with the organization • Retrieves courses offered by the organization
7.	Course	<ul style="list-style-type: none"> • Maintains course identity and descriptive information (id, title, description) • Tracks when the course was created (createdDate) • References the organization the course belongs to (organizationId) • Stores a reference to the teacher assigned to the course (teacherId) 	<ul style="list-style-type: none"> • Retrieves content (lectures, materials) associated with the course • Gets enrollments for the course • Returns assignments assigned for the course • Allows addition of new content to the course • Supports creation of new assignments for the course

8.	Content	<ul style="list-style-type: none"> • Stores content identity and descriptive information (id, title, description) • Maintains a reference to the actual file (fileUrl) • Tracks the type of content (video, PDF, text, audio) • References the course the content belongs to (courseId) • Records who uploaded the content (uploadedBy) and when (uploadDate) 	<ul style="list-style-type: none"> • Provides detailed information about the content • Allows updates to content details
9.	Assignment	<ul style="list-style-type: none"> • Maintains assignment identity and descriptive information (id, title, description) • References the course the assignment belongs to (courseId) • Tracks the deadline for submission (deadline) • Records who created the assignment (createdBy) and when (createdDate) 	<ul style="list-style-type: none"> • Retrieves student submissions for the assignment • Allows extension of the submission deadline
10.	Submission	<ul style="list-style-type: none"> • Stores submission identity information (id) • References the assignment being submitted (assignmentId) • Tracks which student made the submission (studentId) • Maintains a reference to the submitted file (fileUrl) • Records when the submission was made 	<ul style="list-style-type: none"> • Handles assignment submission process • Allows teachers to provide feedback on submissions

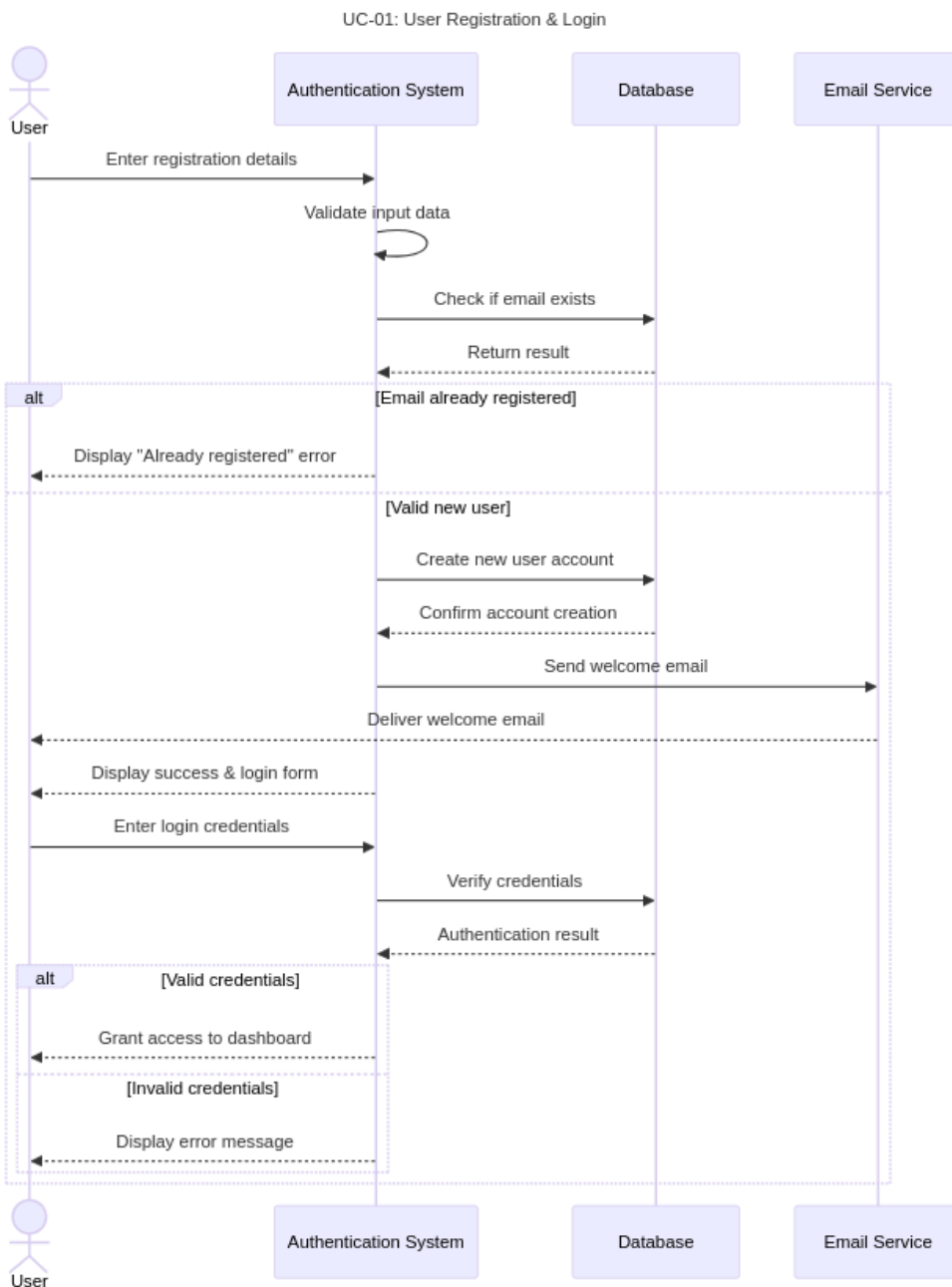
		(submissionDate) <ul style="list-style-type: none"> • Stores teacher feedback on the submission (feedback) • Tracks the status of the submission (submitted, graded, late, pending) 	
11.	Enrollment	<ul style="list-style-type: none"> • Maintains enrollment identity information (id) • References the course being enrolled in (courseId) • Tracks which student is enrolling (studentId) • Records when the enrollment request was made (enrollmentDate) • Stores the status of the enrollment (pending, approved, rejected, completed) 	<ul style="list-style-type: none"> • Processes enrollment approval • Handles enrollment rejection (with reason) • Provides progress tracking information for the enrollment
12.	Feedback	<ul style="list-style-type: none"> • Stores feedback identity information (id) • References the content being evaluated (contentId) • Tracks which user provided the feedback (userId) • Maintains the feedback text (comment) • Records the numerical rating given (rating) • Stores when the feedback was created (createdDate) 	<ul style="list-style-type: none"> • Handles submission of new feedback • Allows editing of existing feedback

Class Diagram:

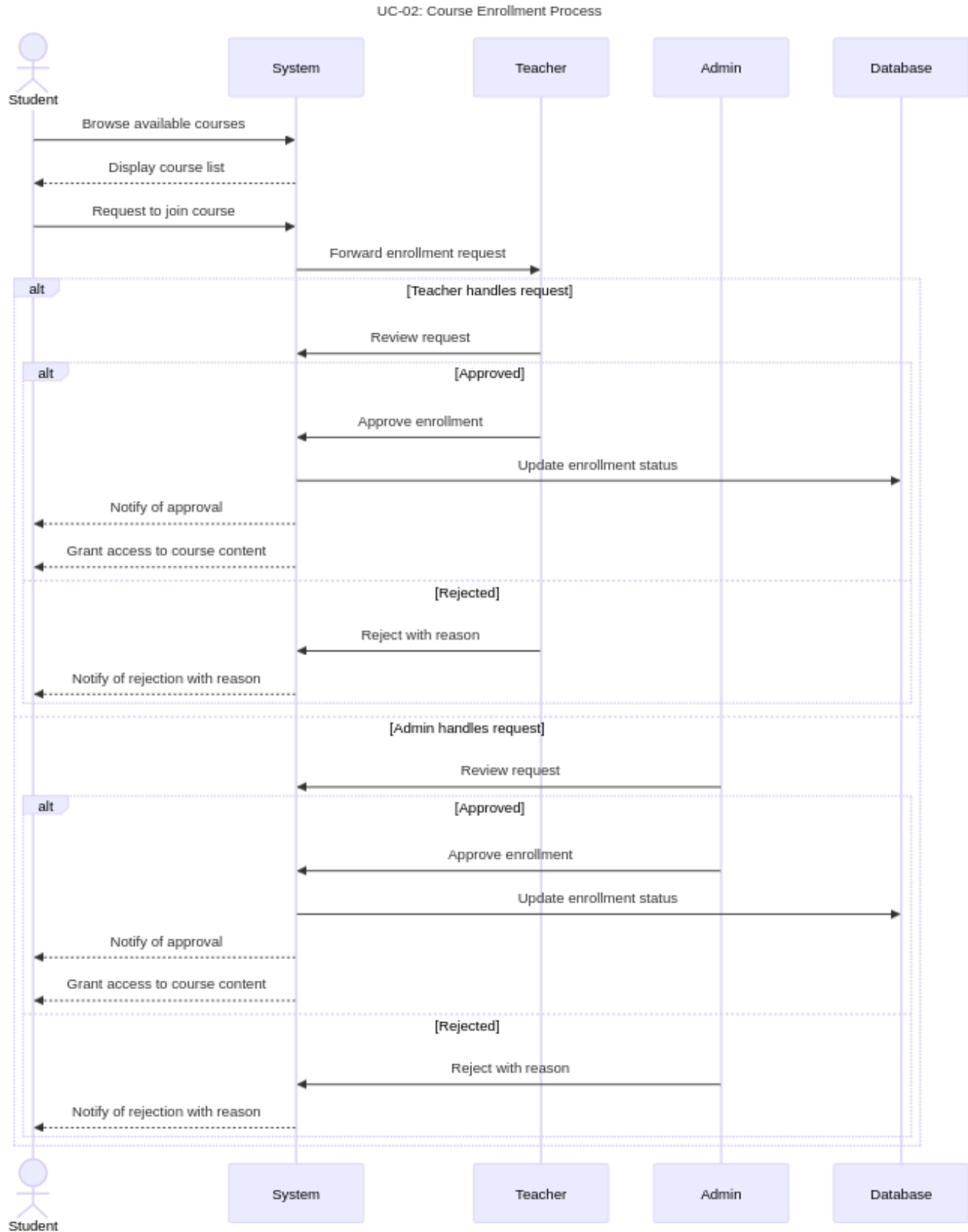
The full diagram is available here: [ class_diagram.png]

Sequence Diagrams:

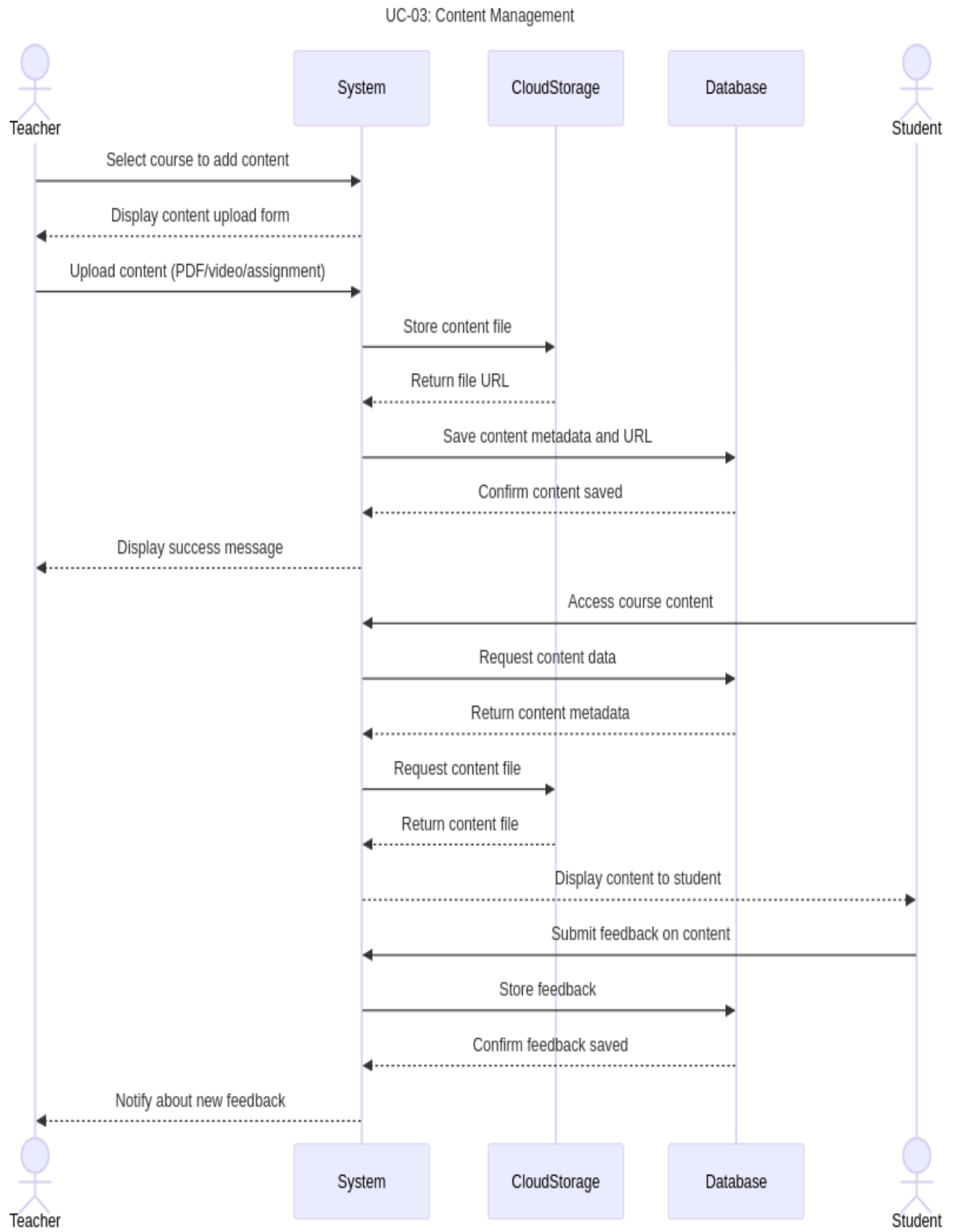
1. User



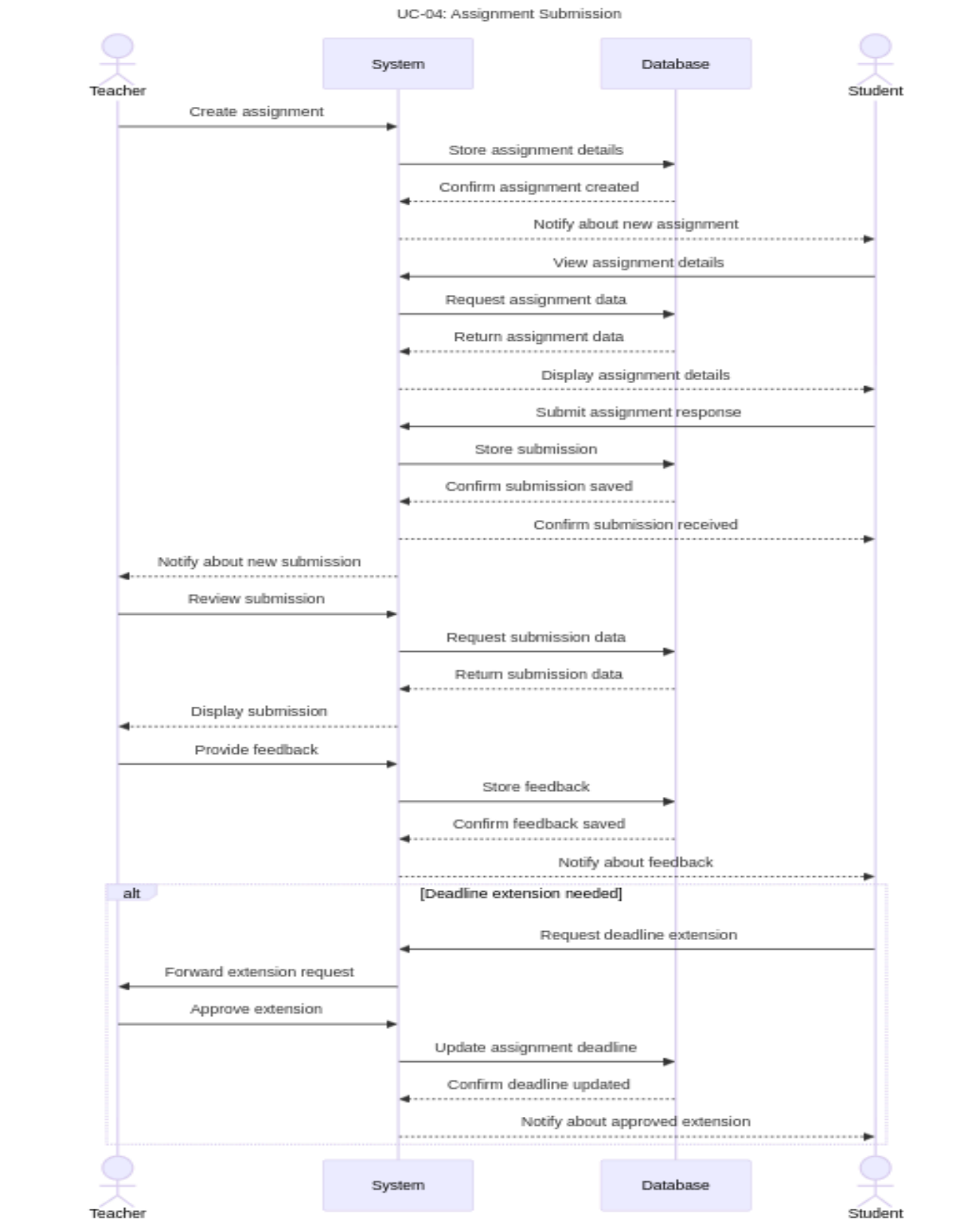
2. Course Enrollment



3. Content Management



4. Assignment Submission



Design Rationale

Evolution of System Architecture and Design Decisions

Initial Conceptualization to Final Design

1. User Role Evolution

Initial Concept:

- Simple three-tier structure: Admin, Teachers, and Students
- Admin represented the NGO with system management responsibilities
- Limited definition of role boundaries and permissions

Final Design:

- Expanded to a five-tier hierarchy: Super Admin, Admin, Teacher, Student, and User
- Added Organization Creator role
- Clear permission structure with Super Admin managing multiple organizations
- Hierarchical approval system implemented

Rationale: The expansion of user roles addressed scalability concerns. The initial three-role system would have limited the platform's ability to manage multiple organizations. By adding the Super Admin and Organization Creator roles, the system now supports a multi-tenant architecture where different organizations can operate independently within the platform.

2. Content Management Approach

Initial Concept:

- Focus on YouTube video embedding
- Limited file format options

Final Design:

- Comprehensive media support via Cloudinary
- Multiple content formats (text, PDFs, images, videos)
- Structured module-based course organization

Rationale: The team recognized that relying solely on YouTube for video content would create external dependencies and potential copyright issues. Implementing Cloudinary for file storage provided greater control over content, better security, and support for a wider range of educational materials beyond just videos.

3. Infrastructure and Technology Stack

Initial Discussion:

- Stack to be "decided based on feasibility"
- Hosting requirements undefined

Final Decision:

- MERN stack (MongoDB, Express.js, React, Node.js)
- Dockerized deployment
- Cloudinary for file storage

Rationale: The MERN stack was chosen for several compelling reasons:

1. MongoDB's document-based structure aligns well with educational content organization
2. React provides an interactive UI essential for an engaging learning experience
3. Node.js and Express offer robust backend capabilities for user authentication and content management
4. The entire stack supports scalability as user numbers grow

The decision to dockerize the application reflects modern deployment best practices, ensuring consistency across development and production environments while simplifying scaling and updates.

4. Authentication Mechanism

Initial Plan:

- Regular login system (rejecting Google login to avoid confusion)

Final Implementation:

- Username/email-based authentication
- Security enhancements with login attempt restrictions (3-4 failures trigger a 30-minute block)
- reCAPTCHA integration

Rationale: The team prioritized security while maintaining simplicity. While third-party authentication providers like Google could have reduced development time, the client expressed concerns about user confusion. The implemented solution balances security needs with the client's preference for a straightforward, self-contained authentication system.

5. Assignment and Progress Tracking

Initial Concept:

- Grading system for assignments
- Multiple submission formats

Revised Approach:

- Status-based tracking (Pending, Incomplete, Completed) rather than numerical grades
- Completion certificate generation at 98% completion
- Extended deadline request feature

Rationale: This represents a shift from traditional academic grading to competency-based assessment. The status-based system simplifies progress tracking while still providing meaningful feedback. The certificate generation feature provides a real reward for completing a course, helping to keep learners motivated in self-paced learning.

Rejected Alternatives and Trade-offs

1. Google Authentication

Rejected Solution: Implementing Google OAuth for user authentication.

Rationale: Despite implementation simplicity, this was rejected due to client concerns about user confusion and potential dependency on external services.

Trade-off: Enhanced development time and reduced security implementation burden versus maintaining greater control and independence.

2. Traditional Grading System

Rejected Solution: Numerical/letter grading for assignments.

Rationale: Simple status tracking better aligned with the platform's focus on completion rather than competitive assessment.

Trade-off: Less granular feedback versus more straightforward progress tracking.

3. Single-tier Administration

Rejected Solution: Single admin role managing all aspects.

Rationale: Would limit scalability to multiple organizations.

Trade-off: Simplified permissions system versus limited growth potential.

Conclusion

The LMS Eklavya Foundation design evolved significantly from initial conceptualization to final implementation. The team made deliberate choices to enhance scalability, security, and user experience while addressing the client's specific needs. The final design represents a balanced approach that prioritizes:

1. Organizational scalability through hierarchical user roles
2. Content flexibility with comprehensive media support
3. Technical robustness via modern development stack and deployment methods
4. Security through thoughtful authentication mechanisms

These design decisions position the platform to effectively serve the educational needs of the Eklavya Foundation while providing room for future growth and enhancement.

