|  |
| --- |
| DXC’s |
| LLD For ConnectED |
| Client Alchemy |

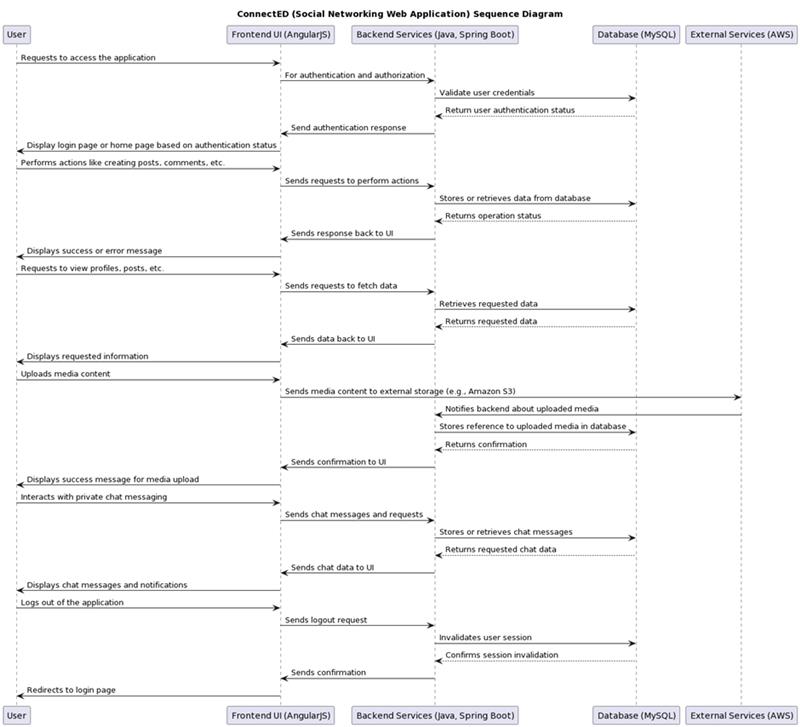
|  |
| --- |
|  |

**1.Introduction:**

The ConnectED - Professional Social Networking Platform’s Low-Level Design (LLD) document serves as a comprehensive guide detailing the intricate technical aspects and implementation specifics required to bring the ConnectED platform to life. It builds upon the foundation laid by the High-Level Design (HLD) document, translating conceptual ideas into tangible technical specifications.

### 2.Low-Level System Design:

**2.1 Sequence Diagram:**



**2.2 Navigation Flow/UI Implementation:**

**Navigation Flow Design:**

* Define the overall navigation structure of the ConnectED web application.

Home Page  
User Profile  
Posts Feed  
Messaging System  
Settings

* Identify the main sections/pages of the application and their relationships.
* Design the navigation paths for users to move between different sections/pages.
* Users can navigate between sections using a top navigation bar and sidebar menu.
* Consider intuitive navigation patterns to ensure a smooth user experience.

**UI Component Design:**

* Design the UI components for each section of the ConnectED web application.  
    **Home Page:**    Display feed of posts from users.  
      Include options to create a new post.  
    **User Profile:**    Show user's profile information, including bio, profile picture, and contact details.  
      Allow editing of profile information.  
    **Posts Feed:**    Display a list of posts with options to like, comment, and share.  
      Implement infinite scrolling for seamless browsing of posts.  
    **Messaging System:**    Enable users to send and receive messages in real-time.  
      Include features for creating new conversations, searching for users, and managing conversations.  
    **Settings:**    Provide options to customize account settings, notification preferences, and privacy settings.  
      Include a section for managing connected accounts and third-party integrations.

**User Interaction Design:**

* Specify user interactions with UI components to perform various actions within the application.
* Clicking on a post displays its details and options to interact (like, comment, share).
* Clicking on a user's profile navigates to their profile page.
* Typing in the messaging system triggers real-time message updates.
* Define user flows for common tasks such as user registration, creating posts, viewing profiles, etc.
* User registration flow includes entering personal information, verifying email/mobile number, and setting up a profile.
* Creating a post involves selecting post type (text, image, video), adding content, and publishing.
* Incorporate feedback mechanisms to provide users with visual cues and feedback on their interactions.
* Show success/error messages upon completion of actions (e.g., post creation, profile update).

**Navigation Controls:**

* Navigation controls such as menus, tabs, and buttons to facilitate easy navigation.
* Top navigation bar contains links to main sections/pages of the application.
* Sidebar menu provides access to additional features and settings.
* Design the placement and behaviour of navigation controls to optimize user navigation and interaction.
* Include clear labels and icons for navigation links.
* Highlight active navigation items to indicate the user's current location within the application.

**Error Handling and Validation:**

* Implement error handling mechanisms and validation checks for user input.
* Validate form fields for completeness and correctness (e.g., required fields, valid email addresses).
* Display error messages inline and/or as pop-up notifications for immediate feedback.
* Ensure robust error handling to maintain data integrity and prevent application crashes.
* Handle server-side errors gracefully and provide users with helpful error messages.

**Accessibility Considerations:**

* Consider accessibility guidelines and standards to ensure the application is accessible to all users.
* Implement features such as keyboard navigation, screen reader support, and high-contrast modes.
* Ensure proper semantic markup for UI elements to enhance accessibility.
* Test the application with accessibility tools and address any accessibility issues identified.

**UI Implementation:**

* Develop the UI components and navigation flow using relevant front-end technologies (e.g., HTML, CSS, JavaScript).
* Use AngularJS for dynamic content rendering and seamless user interactions.
* Implement responsive design principles to ensure the application is compatible with different devices and screen sizes.
* Utilize CSS frameworks like Bootstrap for consistent styling and layout across the application.
* Test the UI implementation across various browsers and devices to ensure compatibility and consistency.
* Test the application on different devices (desktop, tablet, mobile) to ensure responsiveness and usability.

**2.3 Client-Side Validation Implementation:**

**Client-Side Validation Design:**

* Implemented client-side validation to enhance user experience and improve data integrity.
* Utilized JavaScript for client-side validation logic to enforce validation rules on user input.

**Validation Rule Specification:**

* Defined validation rules for input fields across various forms in the ConnectED web application.
* Specified required fields, data formats, character limits, and other constraints for each input field.

**Client-Side Validation Logic:**

* Developed validation functions to validate user input based on the defined validation rules.
* Implemented event handlers to trigger validation checks on user interaction (e.g., onBlur, onSubmit).

**Real-Time Feedback Mechanisms:**

* Implemented real-time feedback mechanisms to provide users with immediate validation feedback.
* Displayed error messages dynamically next to input fields to highlight validation errors.

**Validation Integration with UI Components:**

* Integrated client-side validation logic seamlessly with UI components throughout the application.
* Bound validation functions to relevant UI events to enforce validation rules consistently.

**Cross-Browser Compatibility:**

* Tested client-side validation logic across major web browsers (Chrome, Firefox, Safari) to ensure compatibility.
* Addressed any browser-specific issues to maintain consistent validation behaviour.

**Accessibility Considerations:**

* Ensured that validation error messages are accessible to users with disabilities (e.g., screen readers).
* Implemented keyboard navigation and focus management for users navigating through input fields.

**User Testing and Feedback:**

* Conducted extensive user testing to validate the effectiveness and usability of client-side validation.
* Gathered feedback from users to identify and address any usability issues or areas for improvement.
  1. **Server-side validation Implementation:**

**User Input Validation:**

* **Registration Form:** Validate user inputs such as name, email, password, etc., ensuring they meet specified criteria (e.g., length, format).
* **Profile Updates:** Validate user-submitted profile information, such as job title, skills, education, etc., to maintain data accuracy and consistency.
* **Content Creation:** Validate inputs for creating posts, articles, comments, etc., to maintain content quality.

**Authentication and Authorization:**

* **Login Credentials:** Validate user credentials during login to authenticate users and prevent unauthorized access.
* **Access Control:** Validate user permissions and roles to ensure authorized access to specific features, resources, or functionalities.

**Data Integrity:**

* **Database Operations:** Validate data integrity during database operations such as insertion, update, and deletion to prevent data corruption and maintain data consistency.
* **Referential Integrity:** Ensure referential integrity by validating foreign key constraints and relationships between database entities.

**Error Handling and Reporting:**

* **Validation Errors:** Implement robust error handling mechanisms to handle validation errors gracefully, providing meaningful error messages and feedback to users.
* **Logging:** Log validation errors and exceptions for debugging, auditing, and monitoring purposes.

**Performance Considerations:**

* **Efficiency:** Ensure validation processes are efficient and optimized to minimize computational overhead and response times.
* **Scalability:** Design validation mechanisms to scale with increasing user loads and data volumes, ensuring consistent performance under varying conditions.

**2.5 Components Design Implementation:**

The design and implementation of components within the ConnectED Professional Networking Platform are crucial for achieving modularity, scalability, and maintainability. Each component serves a specific purpose and interacts with other components to fulfil the platform's functionalities.

**Implementation Approach:**

* **Modular Design:** Organize components into logical modules based on their functionalities, such as user management, content management, messaging, etc.
* **Component Interfaces:** Define clear interfaces for communication between components, specifying input/output parameters, data formats, and protocols.
* **Dependency Management:** Manage dependencies between components carefully to minimize coupling and promote reusability.
* **Testing Strategy:** Develop comprehensive unit tests and integration tests for each component to ensure reliability and identify potential issues early in the development process.
* **Scalability Considerations:** Design components with scalability in mind, leveraging techniques such as load balancing, horizontal scaling, and caching where appropriate.

**2.6 Configurations/Settings:**

Configurations and settings management is essential for tailoring the ConnectED platform to meet specific deployment environments, performance requirements, and customization needs.

**Implementation Guidelines:**

* **Configuration Files:** Utilize configuration files or databases to store platform settings, such as database connection strings, API keys, feature toggles, etc.
* **Environment Variables:** Leverage environment variables for dynamic configuration management, allowing for easy customization across different deployment environments (development, staging, production, etc.).
* **Configuration Management Tools:** Integrate configuration management tools or frameworks to automate the deployment and management of configuration settings.
* **Security Considerations:** Implement secure handling of sensitive configurations, such as encrypting sensitive data at rest and in transit, and restricting access to configuration files based on user roles and permissions.

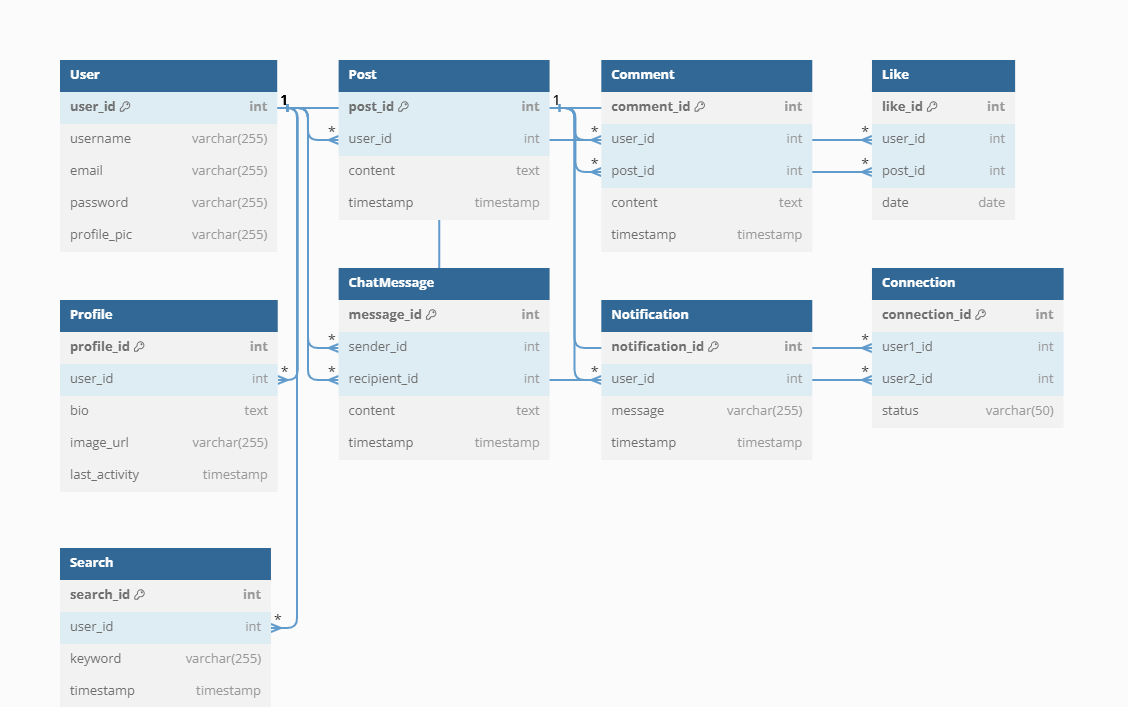
**2.7 Interfaces to Other Components:**

ConnectED Professional Networking Platform interfaces with various external components and services to enhance its functionality and integrate with existing systems.

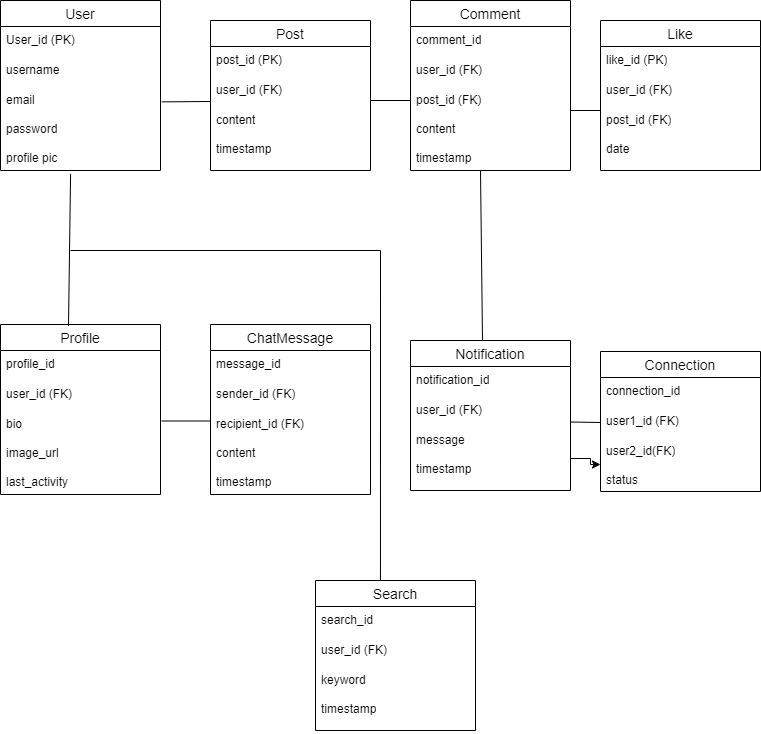
**Implementation Strategies:**

* **API Endpoints:** Design and implement well-defined API endpoints for communication with external systems, ensuring consistency, reliability, and security.
* **Third-Party Integrations:** Integrate with third-party services and APIs, such as authentication providers, messaging services, analytics platforms, etc., using standardized protocols and authentication mechanisms.
* **Data Exchange Formats:** Support common data exchange formats such as JSON, XML for interoperability with external systems.
* **Error Handling:** Implement robust error handling and fault tolerance mechanisms to handle failures gracefully and provide informative error messages to external clients.
* **Documentation:** Provide comprehensive documentation for external interfaces, including API specifications, usage guidelines, authentication requirements, and error handling procedures, to facilitate seamless integration with external systems.

**3. ER Diagram:**



**4. Database Diagram:**

****