

Software Development Documentation

ConnectHub

Faizan Maknojiya

<https://github.com/faizan-maknojia/ConnectHub>

Register

Username

Email

Password

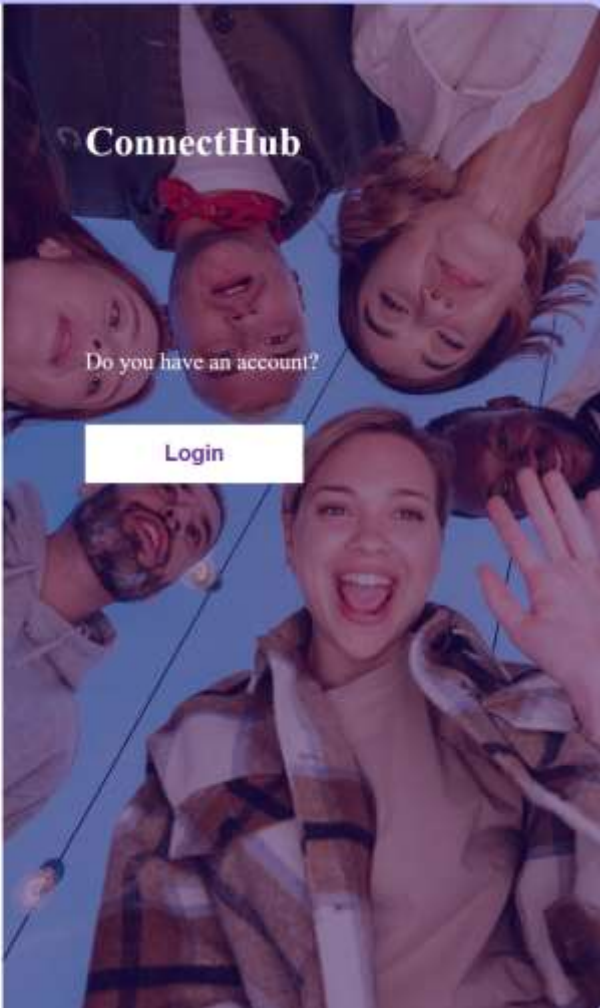
Name

Register

ConnectHub

Do you have an account?

Login



Abstract

This project aims to create 'ConnectHub', a social networking website where users with similar interests can connect and collaborate. Using a client-server setup, users interact with a website to access features like messaging, interest-based groups, events, job listings, and resource centers. We will test the system using POSTMAN, a tool for efficient API testing, ensuring everything works correctly. Detailed responses will be analyzed to guarantee proper functionality, and automation features will be used to create tests that can be repeated. Additionally, we will test the website's performance under different loads to see how well it handles high traffic. Overall, this project aims to promote community engagement as every other platform and collaboration among users with various interests and backgrounds.

Website Conceptualization

Target Users

- **Professionals:**
 - **Demographics:** Adults aged 21-55 across various industries and job roles.
 - **Interests:** Networking, career advancement, industry insights.
 - **Expectations:** Access to professional groups, job postings, skill development resources, and networking opportunities.
- **Enthusiasts:**
 - **Demographics:** Diverse age groups with specific interests such as hobbies, sports, arts, etc.
 - **Interests:** Sharing experiences, discovering like-minded individuals, exploring new interests.
 - **Expectations:** Specialized communities, forums, events, and content tailored to their interests.
- **Students:**
 - **Demographics:** High school and college students.
 - **Interests:** Academic support, extracurricular activities, career guidance.
 - **Expectations:** Study groups, educational resources, mentorship programs, internship opportunities.

Main Functionality

1. **User Profiles:** Users have the capability to generate comprehensive profiles containing personal details, interests, and professional history.

2. **Social Feed:** The platform features a dynamic feed that displays updates from connections, groups, and pertinent content tailored to users' interests.
3. **Groups and Communication:** Users have the option to establish or join groups centered with common interests, professions, or affiliations, enabling them to participate in targeted discussions and activities.
4. **Messaging and Notifications:** A messaging and notification system to facilitate direct communication among users and to ensure they stay updated on activities within their network.
5. **Events and Meetups:** Users have the ability to both create and explore events, spanning virtual and physical realms, allowing them to engage in networking opportunities, workshops, seminars, and more.
6. **Search and Discovery:** The website will provides an advanced search function allowing users to locate individuals, groups, and content based on specified criteria such as location, interests, or profession. Moreover, personalized recommendations tailored to user's activity and preferences, will enhance their experience.
7. **Job Board:** A specialized area is provided for job listings, internships, and freelance opportunities, customized to align with users' professional backgrounds and interests.
8. **Content Sharing:** The ability to share diverse content formats such as articles, images, videos, and documents, fostering discussions and collaborations among users.
9. **Privacy and Security Settings:** Comprehensive privacy controls and security measures will be implemented to safeguard user data and maintain a secure online atmosphere.

Unique Selling Points (USPs):

- **Algorithm-driven matching:** Leveraging algorithms to recommend potential connections by analyzing user profiles and interests.
- **Career advancement hub:** Providing resources, job listings, and mentorship opportunities to support users in their professional growth.
- **Virtual events platform:** Arranging virtual conferences, webinars, and networking gatherings to foster global connections.
- **Community – Centric Approach:** We can emphasis on fostering vibrant and strong communities based on shared interests and passions. This is achieved through a variety of features such as interest-based groups, discussion forums, and collaborative projects, encouraging a deep sense of belonging and fostering active participation among our users.
- **User-centric Design:** The design will prioritize around user-centricity, ensuring that the interface and experience are tailored to user's need. We prioritize simplicity,

intuitiveness, and accessibility, enabling seamless navigation, connection with others, and access to key features according to their preferences.

Development Plan:

1. Research and Analysis:

Undertake surveys, and carry out market research to gain insight into user needs, preferences, and the gaps present in the social networking domain. Analyze competing platforms to identify strengths, weaknesses, and opportunities for setting oneself apart.

2. Design:

Getting ideas from UX/UI designers to craft intuitive and visually appealing interfaces that emphasize usability and accessibility. Generate prototypes to iterate and enhance design concepts, incorporating user feedback into the refinement process.

3. Development:

Front-end: Employ frameworks such as React.js to construct responsive and interactive user interfaces.

Back-end: Deploy a scalable architecture utilizing technologies like Node.js, Express.js, and Postgres to ensure robust data management and API development capabilities.

Implemented, personalized content recommendation can also be useful.

4. Testing:

Carry out thorough testing across various devices and browsers to guarantee compatibility and responsiveness. Additionally, conduct security audits and penetration testing to pinpoint and address potential vulnerabilities.

5. Deployment and Maintenance:

Implementing and prioritizing user acquisition and retention efforts. Establish procedures for consistent updates, bug resolution, and customer support to maintain seamless operations and user satisfaction.

System Overview

ConnectHub uses a client-server setup. Users interact with a web-based front end, while a backend server manages data processing, storage, and business logic.

▪ Front End

- The front end of the platform is created using contemporary web technologies like Sass (short for syntactically awesome style sheets) a dynamic style sheet language that extends CSS, along with JavaScript.

- It offers an intuitive user interface enabling seamless navigation across platform sections, content interaction, and profile management.
 - This interface adapts responsively, ensuring smooth functionality across a spectrum of devices, encompassing desktops, tablets, and smartphones.
- **Backend**
 - The backend is constructed using a microservices structure, enabling scalability, adaptability, and maintainability.
 - It is composed of multiple standalone services, each handling different tasks on the platform.
 - These backend services are created using Node.js and interact with one another through RESTful APIs.
 - We might implement JWT (JSON Web Tokens) for secure authentication and authorization.
- **Development Tools:**
 - **Version Control** – We will use GitHub for version control, collaboration and to track our progress.
 - **IDE** – We will use Visual Studio Code for coding.
 - **Package Manager:** npm for managing the project and install packages.
- **Core Services**
 - User Service: Manages user authentication, registration, profile management and access control
 - Content Service: Manages the creation, storage, fetching, and monitoring of content.
 - Feed Service: Creates customized news feeds for users by considering their interests, connections, and actions.
 - Analytic Service (Ambition): Gathers and studies user information to offer suggestions for ads and content that suit their interests. This feature seems big for us right now, but it is something we aim to add.
- **Database:**
 - The system utilizes a distributed database setup to effectively handle user data, content, and associated information.
 - It includes both relational databases such as MySQL, which is offered by AWS.
 - User Database: Holds user profiles, authentication details, and preferences.
 - Content Database: Contains diverse content types including posts, images, videos, and comments.
 - We keep track of user activity, like their interactions and engagement statistics, by recording them in MySQL databases hosted on AWS.

System Design and Architecture:

- **Database Diagram:**

Here is the detailed database diagram, which includes following entities:

- User
- Message
- Post
- GroupMembership
- InterestGroup
- UserEventAttendance
- Event
- Job

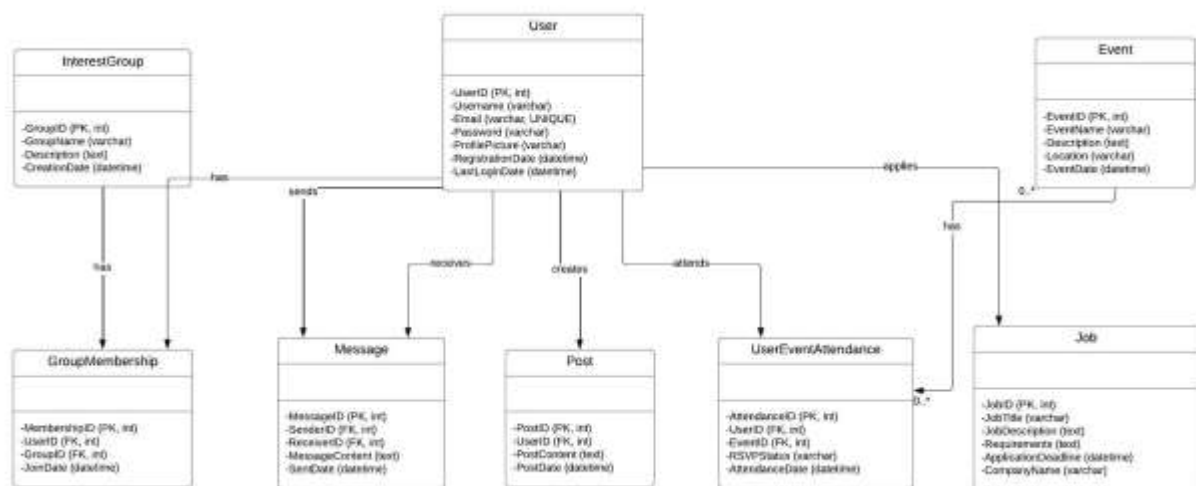


Fig 1: Database Diagram

Legend:

- **PK:** Primary Key
- **FK:** Foreign Key
- **int:** Integer
- **varchar:** Variable Character (String)
- **text:** Text
- **datetime:** Date and Time
- **UNIQUE:** Ensures that each value in the column is unique

- **ER- Diagram**

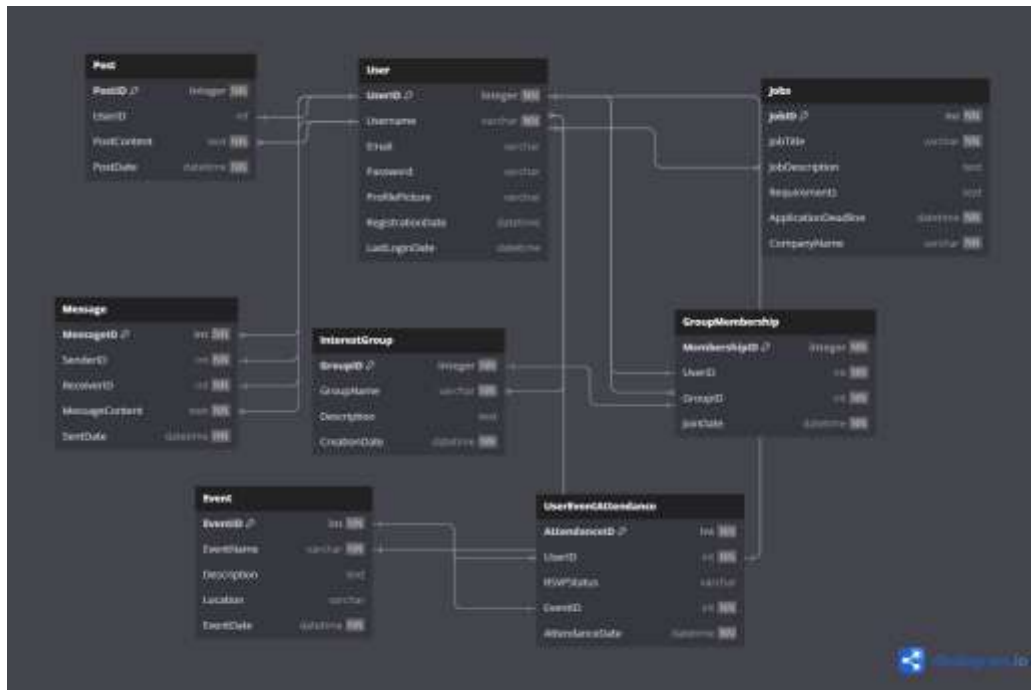


Fig 2: ER-Diagram

Legend:

- **NN** – Not Null
- **Primary Key** (With the symbol of key)

Normalization:

- The database design follows normalization principles up to the third normal form (3NF) to prevent duplication and enhance data integrity.
- **1NF:**
 - As we see in the table, each table have a primary key, ensuring that every entry within the table can be uniquely identified by one or more attributes termed as the primary key.
 - All tables have atomic values, ensuring there are no repeating groups. For instance, the User table does not store multiple email addresses within a single field.

- **2NF:**
 - Every attribute in the tables relies entirely on the primary key for its functionality. For example, in the GroupMembership table, no attribute depends partially on only a segment of primary key.
 - All requirements of 1NF are met
- **3NF:**
 - The tables do not have situations where one attribute is determined by another attribute, instead of directly by the primary key.
 - For instance, in the Job table, attributes like JobDescription are not influenced by attributes other than the primary key, such as JobTitle.
 - All requirements of 1NF and 2NF are met.

System Architecture:

- **Components Required:**
 - Frontend: This serves as the interface through which users engage, accessible via web or mobile platforms.
 - API Gateway: Functions as the access point for directing all API requests to the backend services of the platform.
 - Authentication Service: Manages user registration, login, and session control, ensuring that only authorized users can utilize designated features.
 - User Database: It stores user information such as profiles, preferences, and social connections (friends, followers), typically using MySQL
(*Note: MySQL might not be the optimal choice for storing user data in a social media application because of scalability and data complexities*)
 - Content Management Service: Manages user-generated content such as posts, comments, messages, and media files, potentially interfacing with a distinct content storage solution.
 - Content Storage: It stores user-generated content like images, videos, and documents, which could utilize services such as Amazon S3 for object storage.
 - Search Service: Enables user to search for individuals, content (posts, hashtags), and groups using keywords and filters.
 - Notification Service: Sends notifications to users promptly regarding mentions, messages, friend requests, and various activities, potentially leveraging a service such as Amazon SNS.
 - Analytics Service: Monitors user engagement, analyzes behavior patterns, and offers insights to enhance both the platform and user experience.

- Other Services: Additional services may be required based on the specific functionalities of the social media platform.
 - Live Streaming Service
 - Chat Service
 - Group Management Service
 - Administration and moderation tools

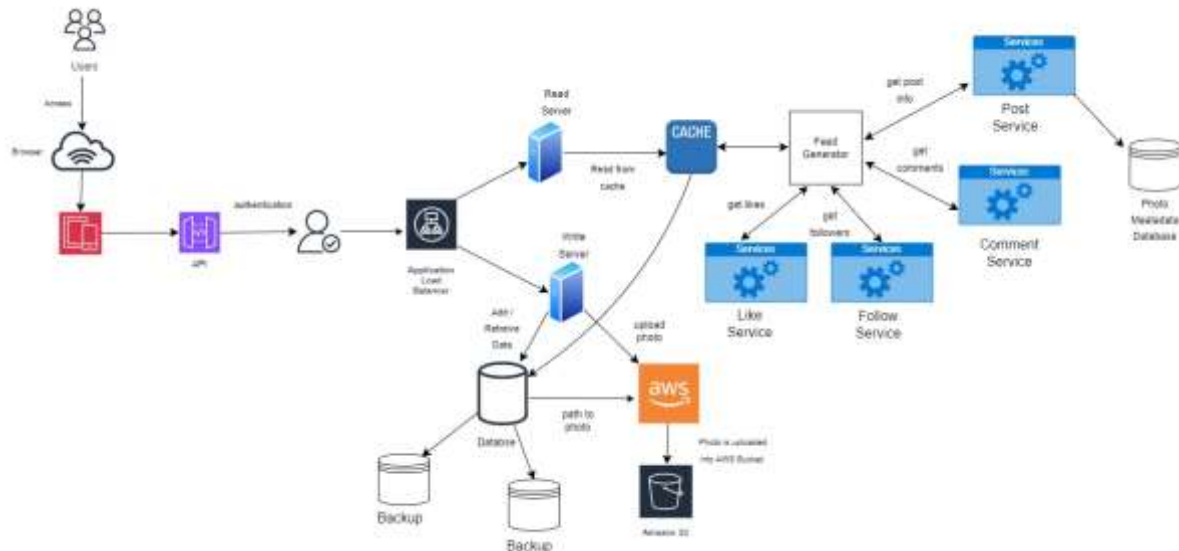


Fig: System Architecture

• Data Flow:

- **User Interaction:** Users engage with the frontend application (web app) to execute actions such as establishing profiles, publishing content, and interacting with fellow users.
- **API Requests:** The frontend communicates by sending API requests to the API Gateway.
- **API Gateway:** Directs the requests to the relevant backend services according to their functionalities (e.g., Authentication Service for login, Content Management Service for posting).
- **Backend Services:** Handles incoming requests, interacts with databases and storage services, and generates corresponding responses.
- **Database Interactions:** Services such as User Database and Content Management Service communicate with MySQL or other databases to store and retrieve user and content information.
- **Content Storage:** The Content Management Service interacts with Content Storage to manage the storage and retrieval of user-generated media files.
- **Feed Generation:** The Feed Generation Service customizes user feeds using data specific to each user and their interactions with content.
- **Response:** The API Gateway returns the response from backend services to the frontend application.
- **Frontend Updates:** The frontend application updates the user interface according to the response received.

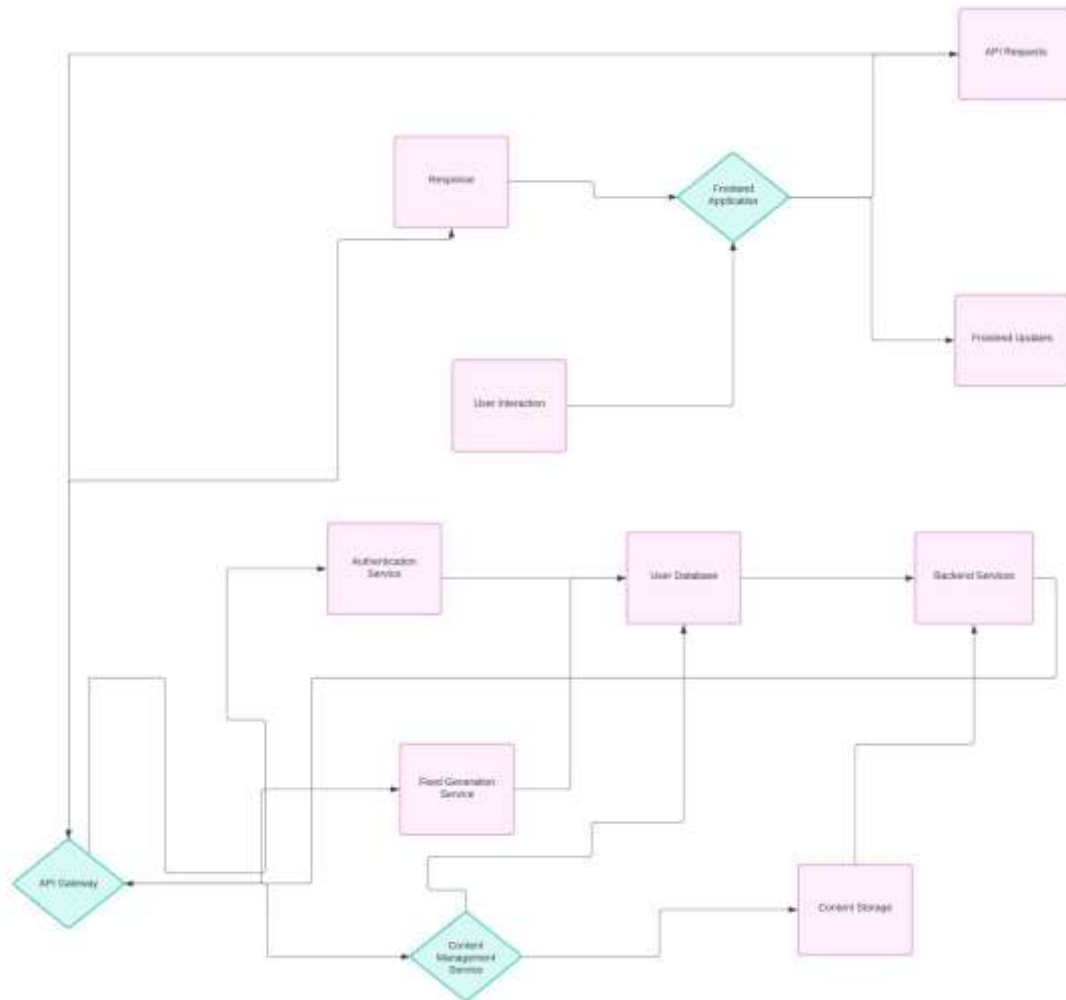


Fig: Dataflow Diagram

- **Additional Consideration:**

- **Security:** Security measures should be a priority in the architecture to protect user data and ensure platform integrity. This includes implementing secure authentication, encrypting data, and deploying intrusion detection/prevention systems.
- **Scalability:** The platform should be designed with scalability in mind to support an expanding user base and increasing data volume. Leveraging cloud-based services with auto-scaling capabilities can be advantageous.
- **Performance:** The architecture must be optimized for peak performance, ensuring rapid response times and smooth user experiences. This may involve incorporating caching mechanisms, Content Delivery Networks (CDNs), and efficient database querying techniques.

- **Testing:**

We will utilize POSTMAN to test our system, a robust tool for API testing that enables fast and efficient testing. Through POSTMAN, we can send different types of requests

like GET, POST, PUT, and DELETE to our API endpoints, and analyze the resulting responses.

Procedure to use Postman:

- ✓ **API Endpoint Testing:**
 - We will use POSTMAN to dispatch requests to our API endpoints to verify their proper functioning. This entails examining GET requests for data retrieval, POST requests for new data creation, PUT requests for existing data updates, and DELETE requests for data removal.
- ✓ **Parameter Request:**
 - POSTMAN allows us to define request parameters like headers, query parameters, and request body. We will utilize these functionalities to dispatch requests with different parameters and confirm that our API manages them accurately.
- ✓ **Validating Response:**
 - Once requests are sent, POSTMAN furnishes comprehensive responses comprising status codes, headers, and response bodies. We will verify these responses to confirm they align with our expectations, ensuring correct status codes and expected data within the response bodies.
- ✓ **Automation Testing:**
 - POSTMAN has features for automating tests, like running sets of requests in order and checking expected results. We can also use this automation to make test packages that we can use repeatedly to check if our API still works when we make changes.
- ✓ **Performance Testing:**
 - Moreover, POSTMAN lets us do performance testing by sending many requests at once and checking how long it takes to get responses. We can also use this to see how well our API works when many people are using it at the same time.

Conclusion and Future Work:

- ❖ In summary, "ConnectHub" offers a platform for users to connect, collaborate, and engage around shared interests. Thorough testing has ensured its reliability. Moving forward, we will focus on enhancing features, growing our user base, and ensuring platform stability and security.
- ❖ Next steps for "ConnectHub" involve improving current features, growing our user community, and ensuring the platform remains stable and secure. We can do this by listening to user feedback and regularly checking and updating the platform to keep it running smoothly.

