System Design Document (SDD)

LinkedIn Clone (MERN Fullstack Project)

1. Introduction

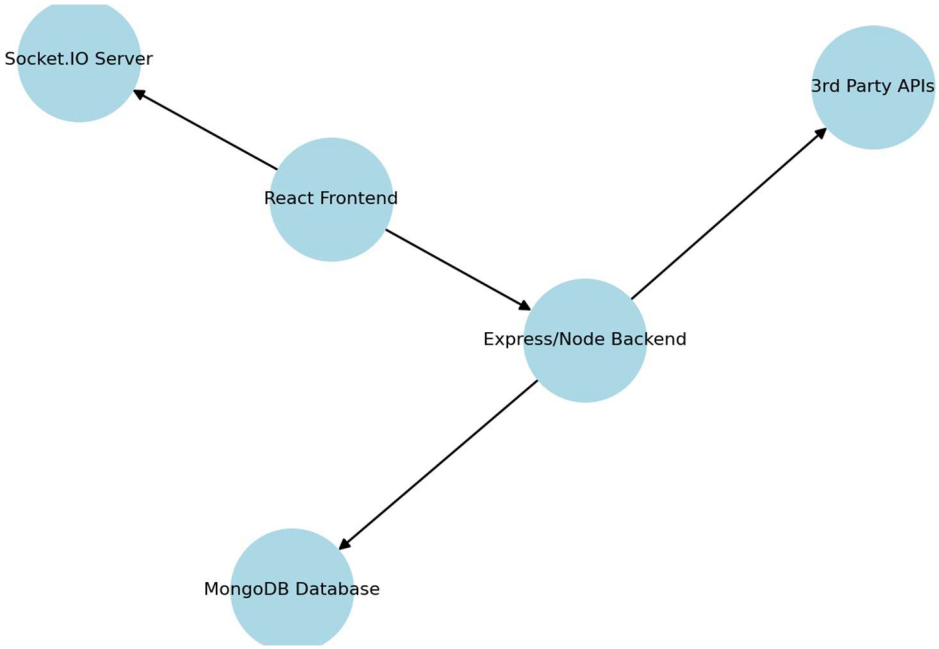
The LinkedIn Clone aims to replicate the professional networking features of LinkedIn, including user authentication,
posting content, connecting with other users, chatting, job listings, and profile management.

Scope:
- User Registration & Login (JWT-based authentication)
- User Profiles (education, skills, experience, endorsements)
- Posts, Likes, Comments, and Feeds
- Connection System (invite/accept connections)
- Messaging (real-time chat with Socket.IO)
- Job Listings (apply and post jobs)
- Notifications & Search functionality

2. System Architecture

Definition: System Architecture describes the **high-level structure** of the system.

- **Architecture Components:**
  - **Frontend:** React.js (UI, Redux for state management)
  - **Backend:** Node.js + Express (REST APIs, authentication, business logic)
  - **Database:** MongoDB (NoSQL schema, scalable document store)
  - **Authentication:** JWT with OAuth (Google/LinkedIn login possible)
  - **Real-Time Communication:** Socket.IO for chat & notifications
  - **Deployment:** Nginx reverse proxy + Docker containers (optional)
- **System Architecture Diagram:**

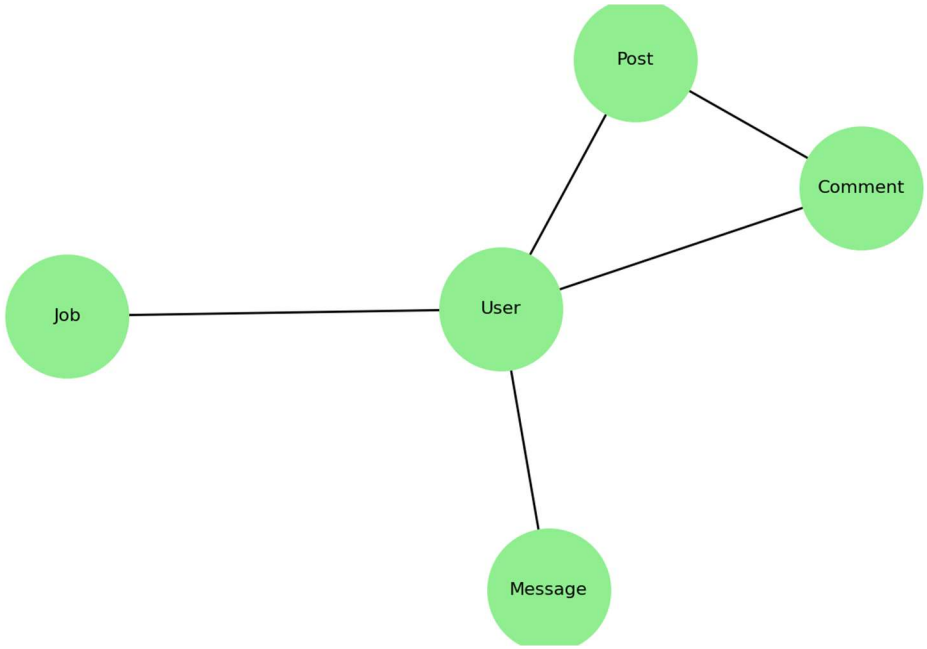


3. UML Diagrams

Use Case Diagram: User interacts with Profile, Posts, Jobs, Messaging, Admin moderates content.

Class Diagram

- **User**: id, name, email, password, skills, experience, connections
- **Post**: id, author, content, likes, comments
- **Message**: id, sender, receiver, content, timestamp
- **Job**: id, title, description, postedBy, applicants



4. Entity-Relationship (ER) Diagram

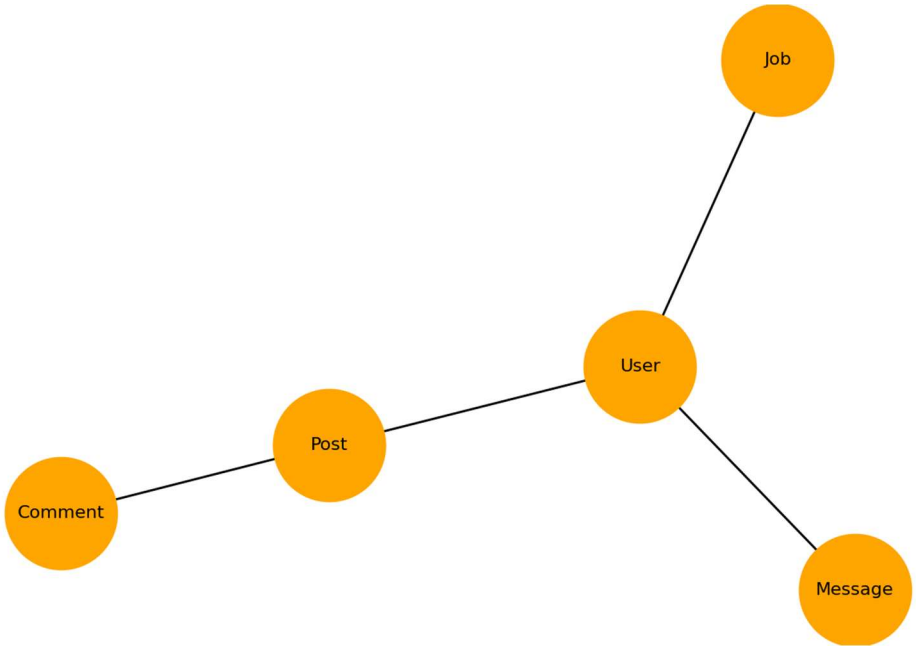
ER Diagram:

Entities:

- User (UserID, Name, Email, Skills, Experience)
- Post (PostID, Content, AuthorID, Likes)
- Comment (CommentID, Content, PostID, AuthorID)
- Message (MessageID, SenderID, ReceiverID, Content)
- Job (JobID, Title, Description, RecruiterID, Applicants)

Relationships:

- User ↔ Post (1-to-many)
- User ↔ Connection (many-to-many)
- User ↔ Message (many-to-many)
- Recruiter ↔ Job (1-to-many)



5. Data Models

Example Mongoose Schema (User, Post, Job).

User Schema: name, email, password, skills, experience, connections
Post Schema: author, content, likes, comments
Job Schema: title, description, postedBy, applicants

- **Conceptual Model**

High-level entities: User, Post, Job, Message

- **Logical Model (MongoDB Collections)**
  - Users Collection
  - Posts Collection
  - Jobs Collection
  - Messages Collection

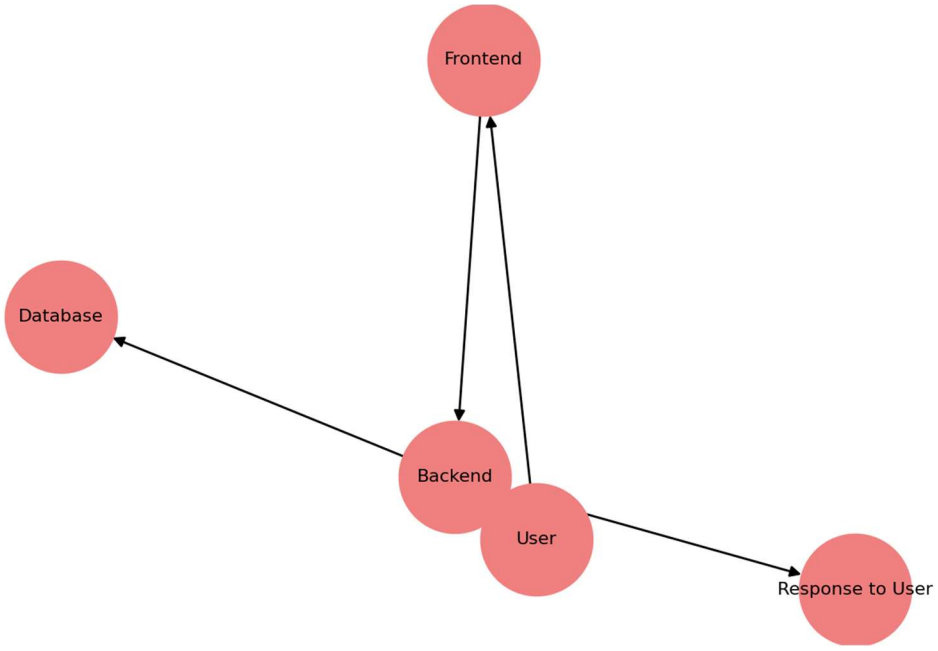
6. Data Flow Diagram (DFD):

Level 0:

- User → API Gateway → Backend Services → MongoDB

Level 1 (Example - Posting a Job):

- User inputs job → Backend validates → Database stores → Confirmation sent → Feed updated



7. Non-Functional Requirements

- Performance: Must support 10k+ concurrent users
- Security: JWT + bcrypt password hashing + HTTPS
- Scalability: Horizontal scaling using microservices & MongoDB sharding
- Availability: 99.9% uptime using load balancing
- Compliance: GDPR data protection

8. Results / Current Achievements

- Built a MERN-based LinkedIn clone with authentication, profile, posts, connections, messaging, and jobs
- Real-time chat and notifications using Socket.IO
- Deployed prototype with cloud support (AWS/Heroku)
- Verified CRUD operations across modules

9. Future Goals

- AI-Powered job and connection recommendations
- Endorsements and skill validation
- Groups and communities
- Premium subscription features
- Video/Voice calls
- Microservices & Kafka for large-scale systems
- Mobile application (React Native)
- Advanced search with Elasticsearch