Client Requirements Specification (CRS)

LinkedIn Clone Project (MERN Stack)

---

1. Introduction

This document outlines the requirements for building a **LinkedIn Clone** using the **MERN stack** (MongoDB, Express, React, Node.js). The clone will provide features similar to LinkedIn, including user authentication, profile management, posts, connections, messaging, notifications, etc. It will allow users to interact and network online, mimicking the functionality of LinkedIn.

Purpose

The purpose of this document is to define the functional and non-functional requirements that the client expects from the LinkedIn Clone application. This will guide the development, testing, and deployment phases.

Scope

The system will provide a social networking platform for professionals, allowing them to:

- Create and manage profiles.
- Share posts and articles.
- Interact with other users through messages, comments, and likes.
- Send and accept connection requests.
- Search for other professionals.
- Receive notifications about network activities.

Stakeholders

- **Client**: The business or individual who has requested the development of this platform.
- **Users**: The end users who will use the platform for professional networking.
- **Development Team**: Developers responsible for building the platform.
- **Testing Team**: QA team responsible for testing the platform's functionality and performance.

---

2. Functional Requirements

The following describes the features and functionalities expected from the application.

2.1 User Authentication

- **FR1.1**: Users should be able to register by providing their name, email, password, and other basic details (e.g., profile picture).
- **FR1.2**: Users should be able to log in with their registered credentials.

- **FR1.3**: Users can reset their password via email.
- **FR1.4**: JWT (JSON Web Token) will be used for authentication and maintaining the session.
- **FR1.5**: Users should be able to log out of the system.

2.2 Profile Management

- **FR2.1**: Users should be able to view and edit their profile (name, headline, profile picture, experience, education, etc.).
- **FR2.2**: Users should be able to change their profile picture.
- **FR2.3**: Users should be able to update their professional information (e.g., job title, company name, location).
- **FR2.4**: Users should be able to add their skills, endorsements, and certifications.

2.3 Post and Feed

- **FR3.1**: Users should be able to create posts (text, images, links).
- **FR3.2**: Users should be able to edit and delete their posts.
- **FR3.3**: Users should be able to like and comment on posts.
- **FR3.4**: The home feed should show posts from the user's connections.
- **FR3.5**: Posts should be displayed with the author's profile picture and name.
- **FR3.6**: Users should be able to share posts with their connections.

2.4 Connections and Networking

- **FR4.1**: Users should be able to send connection requests to other users.
- **FR4.2**: Users should be able to accept or reject connection requests.
- **FR4.3**: Users should be able to remove connections.
- **FR4.4**: The system should display the number of connections on the user's profile.
- **FR4.5**: Users should be able to view their connections' profiles.

2.5 Messaging

- **FR5.1**: Users should be able to send private messages to their connections.
- **FR5.2**: Users should be able to view their message inbox and individual message threads.
- **FR5.3**: Users should be able to delete messages.
- **FR5.4**: The system should allow real-time messaging using **Socket.IO**.

2.6 Search Functionality

- **FR6.1**: Users should be able to search for other users by name, job title, company, or skills.
- **FR6.2**: The search results should display user profiles and related information.

- **FR6.3**: Users should be able to filter search results based on location, industry, or other relevant criteria.

2.7 Notifications

- **FR7.1**: Users should receive notifications when they receive a new message or connection request.
- **FR7.2**: Users should be notified when someone likes or comments on their post.
- **FR7.3**: The system should support real-time notifications.

---

3. Non-Functional Requirements

These requirements define the overall system qualities that are expected from the project.

3.1 Performance Requirements

- **NFR1.1**: The system should handle up to 5000 concurrent users.
- **NFR1.2**: Page load time should be under 2 seconds for a smooth user experience.
- **NFR1.3**: The system should support fast data retrieval, especially for user profiles and posts.

3.2 Scalability

- **NFR2.1**: The application should be scalable to accommodate future growth in user numbers, connections, and posts.
- **NFR2.2**: The system should support horizontal scaling for the backend servers (Node.js/Express).

3.3 Security Requirements

- **NFR3.1**: User passwords should be hashed and stored securely.
- **NFR3.2**: All sensitive data should be transmitted over HTTPS.
- **NFR3.3**: The system should implement role-based access control to protect user data.
- **NFR3.4**: The system should use JWT tokens to prevent unauthorized access to protected routes.

3.4 Usability

- **NFR4.1**: The user interface should be intuitive and user-friendly.
- **NFR4.2**: The platform should be responsive, working seamlessly on desktop and mobile devices.

3.5 Reliability

- **NFR5.1**: The system should be available 99.9% of the time.
- **NFR5.2**: Regular backups should be performed for user data.
- **NFR5.3**: Error logs should be maintained for troubleshooting and debugging.

3.6 Maintainability

- **NFR6.1**: The system should be modular and easy to maintain.
- **NFR6.2**: Code should be well-documented for future development and troubleshooting.
- **NFR6.3**: Automated tests should be implemented for critical functionality.

---

4. Assumptions and Constraints

4.1 Assumptions

- The application will use the **MERN stack** (MongoDB, Express.js, React.js, Node.js).
- All users will have access to a modern web browser.
- The application will support both desktop and mobile views.
- Data storage will be handled through **MongoDB** (using **MongoDB Atlas** if hosted).
- **Socket.IO** will be used for real-time messaging and notifications.

4.2 Constraints

- The application will initially be deployed on **Heroku** or similar cloud platforms.
- The database should be optimized to handle a large number of user interactions (i.e., posts, messages).
- The project needs to be completed within 3 months from the start date.

---

5. Appendix

5.1 Glossary

- **JWT (JSON Web Token)**: A compact, URL-safe means of representing claims to be transferred between two parties.
- **Socket.IO**: A library for real-time web applications, enabling bidirectional communication between clients and servers.
- **MERN**: MongoDB, Express, React, Node.js (a full-stack JavaScript framework).