### **Functional Requirements**

**2.1 User Registration & Authentication**:

**2.1.1 Description**:

* The system shall allow users to register using their email or social media accounts (OAuth).
* The system shall authenticate users during login using OAuth tokens.
* The system shall provide password recovery options.

**2.1.2 User Stories**:

* *As a user, I want to register with my email or social media account so that I can create an account and use the app.*
* *As a user, I want to log in securely so that I can access my account and post content.*

**2.1.3 Acceptance Criteria**:

* The system should send a confirmation email upon registration.
* The user should be able to log in with valid credentials and receive an OAuth token.
* The user should be able to reset their password via email.

**2.2 Post Creation and Interaction**:

**2.2.1 Description**:

* The system shall allow users to create, edit, and delete posts.
* The system shall allow users to like and comment on posts.
* The system shall refresh the feed using API polling to display the latest posts.

**2.2.2 User Stories**:

* *As a user, I want to create a post so that I can share content with my friends.*
* *As a user, I want to like and comment on posts so that I can engage with content from other users.*

**2.2.3 Acceptance Criteria**:

* Users should be able to create posts with text, images, or videos.
* Users should be able to like and comment on posts, with the UI updating dynamically.
* The feed should refresh at regular intervals to display new posts.

**2.3 Real-Time Chat**:

**2.3.1 Description**:

* The system shall allow users to send and receive messages in real-time using WebSockets.
* The system shall allow users to chat only if they are friends.

**2.3.2 User Stories**:

* *As a user, I want to send and receive messages in real-time so that I can communicate with my friends instantly.*
* *As a user, I want to chat only with my friends so that my communication is secure and private.*

**2.3.3 Acceptance Criteria**:

* Users should be able to initiate a chat with friends and receive messages in real-time.
* The system should notify users of new messages via the UI.
* Users should not be able to chat with users who are not in their friend list.

**2.4 Friend Request Management**:

**2.4.1 Description**:

* The system shall allow users to send, receive, and accept friend requests.
* The system shall only allow users who are friends to interact via chat.

**2.4.2 User Stories**:

* *As a user, I want to send friend requests so that I can connect with other users.*
* *As a user, I want to accept or decline friend requests so that I can control my social connections.*

**2.4.3 Acceptance Criteria**:

* Users should be able to send friend requests to other users.
* Users should be notified of received friend requests and be able to accept or decline them.
* Only users who have accepted each other's friend requests should be able to chat.

**2.5 Media Streaming**:

**2.5.1 Description**:

* The system shall allow users to stream video and photos in posts using WebRTC.
* The system shall ensure that media streams are efficient and of high quality.

**2.5.2 User Stories**:

* *As a user, I want to upload and stream videos and photos in my posts so that I can share media content with my friends.*

**2.5.3 Acceptance Criteria**:

* Users should be able to upload and stream media in their posts.
* The media should load and play efficiently without significant buffering.

### **3. Non-Functional Requirements**

**3.1 Performance**:

* The system shall support at least 1,000 concurrent users with minimal latency.
* The feed refresh (API polling) shall not exceed a 5-second interval.

**3.2 Security**:

* All data transmissions shall be encrypted using SSL/TLS.
* OAuth tokens shall be securely stored and transmitted.
* The system shall implement role-based access control (RBAC) for admin and user roles.

**3.3 Usability**:

* The system shall have a responsive design, accessible on both desktop and mobile devices.
* The UI shall be intuitive and user-friendly, with clear navigation and feedback mechanisms.

**3.4 Reliability**:

* The system shall maintain an uptime of 99.9%.
* The system shall include automated testing to ensure the reliability of features before deployment.

**3.5 Scalability**:

* The system shall be designed to scale horizontally to accommodate increasing numbers of users.
* The system shall allow for database sharding and load balancing to manage high traffic.

### **4. System Architecture Requirements**

**4.1 Backend**:

* The backend shall be built using NestJS, with PostgreSQL as the database.
* The backend shall expose RESTful APIs for frontend interaction and use WebSockets for real-time communication.

**4.2 Frontend**:

* The frontend shall be built using React.js with Redux for state management.
* The frontend shall consume the backend APIs and implement WebRTC for media streaming.

**4.3 Data Management**:

* The system shall ensure data consistency and integrity using PostgreSQL.
* The system shall implement caching mechanisms where appropriate to optimise performance.

### **5. Acceptance Criteria and Testing**

**5.1 Acceptance Testing**:

* All features shall undergo thorough testing to ensure they meet the defined requirements.
* User acceptance testing (UAT) shall be conducted to ensure the application meets user needs.

**5.2 Performance Testing**:

* Load testing shall be performed to ensure the system can handle the expected number of concurrent users.

**5.3 Security Testing**:

* Security testing shall be performed to identify and mitigate vulnerabilities, ensuring data protection.