Non-Functional Requirements – Simplified and Realistic

1. Performance

1. **Response Time:**
    - The system should load the homepage within **3-5 seconds** under normal load conditions.
    - Notifications and messages should be delivered with a maximum delay of **2-3 seconds**.
2. **Scalability:**
    - The system should support up to **1,000 concurrent users** without significant performance degradation.
    - The system should be able to handle **5,000 daily active users**.
3. **Database Performance:**
    - Database queries should execute in less than **200 milliseconds** for most operations.
    - Basic caching (e.g., in-memory caching) should be used to improve performance for frequently accessed data.

---

2. Security

4. **User Authentication:**
    - Passwords should be hashed using **bcrypt** with a minimum of **10 rounds**.
    - User sessions should be managed using **JSON Web Tokens (JWT)** with a secure secret key.
5. **Data Encryption:**
    - All sensitive data (e.g., passwords, personal information) should be encrypted **in transit (using HTTPS)**.
    - Basic encryption (e.g., AES-128) can be used for sensitive data at rest.
6. **Access Control:**
    - Users should only be able to access their own data (e.g., messages, profile information).
    - Admins should have restricted access to user data based on their roles.
7. **Prevention of Common Attacks:**
    - The system should be protected against common web vulnerabilities such as **SQL Injection** and **Cross-Site Scripting (XSS)**.
    - Implement basic rate limiting to prevent brute-force attacks on login and password reset endpoints.

---

3. Usability

8. **User Interface:**
    - The application should have a **responsive design** that works on desktop and mobile devices.
    - The user interface should be simple and intuitive, following basic design principles.
9. **Accessibility:**
    - The application should support basic accessibility features, such as alt text for images and keyboard navigation.
10. **Error Handling:**
    - Error messages should be clear and guide users on how to resolve issues.
    - Basic tooltips and help text can be provided for complex features.

---

4. Reliability

11. **Uptime:**
    - The system should have an uptime of **99%** (less than 3.65 days of downtime per year).
12. **Backup and Recovery:**
    - The system should perform **weekly backups** of all critical data.
    - In case of failure, the system should be able to recover data within **24 hours**.
13. **Fault Tolerance:**
    - The system should be able to handle minor hardware or software failures without significant downtime.
    - Basic redundancy can be implemented for critical components (e.g., database).

---

5. Maintainability

14. **Code Quality:**
    - The codebase should follow basic **clean code principles** and be well-documented.
    - The system should have a **modular architecture** to allow for easy updates and maintenance.
15. **Version Control:**
    - The system should use **Git** for version control, with a simple branching strategy (e.g., main and feature branches).
    - Code changes should be reviewed through **pull requests**.

16. **Testing:**
    - The system should have **unit tests** and **integration tests** covering at least **70%** of the codebase.
    - Automated tests should run as part of the **CI/CD pipeline**.