**Experiment-No.2**

**Objective:** Prepare an SRS document in line with the IEEE recommended standards for the specified Case Study. (Non Functional Requirements)

**Non-functional requirements (NFRs) :-** are essential for shaping the overall quality and behavior of a system. Unlike functional requirements that define what the system does, NFRs focus on **how** the system performs and behaves. Let’s explore some key non-functional requirements relevant to an **Online Blogging System**:

1. **Reliability**:
   * The system should be highly available and minimize downtime.
   * Blogs and user data must be securely stored and backed up regularly.
   * In case of failure, the system should gracefully recover without data loss.
2. **Performance**:
   * **Response Time**: Web pages should load quickly (e.g., within 3 seconds) to enhance user experience.
   * **Scalability**: The system should handle increasing traffic (e.g., 20 million users) without performance degradation.
   * **Throughput**: The platform should support concurrent blog creation, editing, and commenting.
3. **Security**:
   * **Authentication and Authorization**: Ensure secure user authentication and authorization mechanisms.
   * Protect against common vulnerabilities (e.g., SQL injection, cross-site scripting).
   * Safeguard user data and prevent unauthorized access.
4. **Usability**:
   * The user interface should be intuitive, with clear navigation and minimal learning curve.
   * Accessibility features (e.g., screen reader compatibility) should be considered.
5. **Scalability**:
   * The system architecture must accommodate growth in users, blogs, and comments.
   * Horizontal scaling (adding more servers) should be feasible.
6. **Maintainability**:
   * Codebase should follow best practices for readability, modularity, and maintainability.
   * Documentation should be comprehensive for future developers.
7. **Data Integrity**:
   * Ensure consistency and accuracy of blog content.
   * Prevent data corruption or loss during updates or migrations.
8. **Compatibility**:
   * The system should work across different browsers (Chrome, Firefox, Safari) and devices (desktops, tablets, mobiles).
   * Consider backward compatibility with older browsers.
9. **Performance Testing**:
   * Regularly test performance under load (e.g., stress testing, load testing).
   * Identify bottlenecks and optimize critical paths.