**1. Non-Functional Requirements (NFRs) for FYP**

**1.1 Performance Requirements**

1. **Response Time**
   * APIs must respond within **500ms** for 90% of requests under normal load.
   * Under peak load, response times must not exceed **1 second**.
2. **Concurrent Users**
   * The system should support **10,000 concurrent users** without degradation.
3. **Throughput**
   * The application must handle up to **1,000 API requests per second**.
4. **Database Query Latency**
   * Database queries must execute within **100ms** for 95% of operations.
5. **Scalability**
   * The system must scale horizontally to accommodate increased load.
6. **Error Rate**
   * Error rates should remain below **0.1%** under normal load conditions.

**1.2 Security Requirements**

1. **Authentication and Authorization**
   * Use **OAuth 2.0** or **JWT (JSON Web Tokens)** for secure user authentication.
   * Ensure role-based access control (RBAC) to restrict unauthorized access.
2. **Data Encryption**
   * Enforce **HTTPS** using **TLS 1.2+** for all API communications.
   * Encrypt sensitive data stored in the database using AES-256.
3. **Input Validation and Sanitization**
   * Validate all inputs to prevent **SQL Injection**, **XSS**, and other attacks.
4. **Rate Limiting**
   * Limit API usage to prevent abuse (e.g., 100 requests per minute per user).
5. **Error Handling**
   * Ensure error responses do not expose sensitive system information.
6. **Logging and Monitoring**
   * Implement logging for all authentication, authorization, and data access events.
   * Use monitoring tools to detect and respond to suspicious activity.
7. **API Gateway**
   * Deploy an API gateway to centralize authentication, monitoring, and rate-limiting.

**2. Software Test Plan**

**2.1 Objectives**

The objective is to validate the system against functional and non-functional requirements, ensuring scalability, security, and usability.

**2.2 Test Scope**

* **Functional Testing**: Validate that all RESTful APIs function correctly.
* **Performance Testing**: Test API performance under load and stress conditions.
* **Security Testing**: Identify vulnerabilities and ensure data integrity.
* **Usability Testing**: Test the system's user interface for intuitiveness and responsiveness.

**2.3 Testing Strategies**

**Functional Testing**

* **Unit Tests**: Test individual API endpoints for correct responses.
* **Integration Tests**: Verify that different system modules communicate effectively.
* **End-to-End Tests**: Simulate real-world scenarios to validate workflows.

**Performance Testing**

* Use **K6** or **Apache JMeter** to simulate different loads.
* **Load Testing**: Test system behavior under expected load.
* **Stress Testing**: Determine the system's breaking point.

**Security Testing**

* Use **OWASP ZAP** or **Burp Suite** to perform:
  + **Penetration Testing**: Simulate real-world attacks.
  + **Input Validation Checks**: Verify inputs are sanitized and validated.

**Usability Testing**

* Conduct usability studies with real users.
* Use tools like **Hotjar** to gather heatmap data for frontend usability.

**2.4 Testing Tools**

1. **Functional Testing**:
   * **Jest + Supertest** (for Node.js-based APIs).
   * **Postman** (for manual API testing).
2. **Performance Testing**:
   * **K6**, **Apache JMeter**, **Locust**, or **Gatling**.
3. **Security Testing**:
   * **OWASP ZAP**, **Burp Suite**, **Nmap**.
4. **Usability Testing**:
   * **Hotjar**, **Google Lighthouse**.

**2.5 Testing Environment**

| **Component** | **Details** |
| --- | --- |
| **API Environment** | Node.js backend with PostgreSQL. |
| **Frontend Tools** | React/Next.js. |
| **Performance Tools** | K6 and JMeter. |
| **Security Tools** | OWASP ZAP, Burp Suite. |

**2.6 Sample Test Case**

| **Test ID** | **Scenario** | **Steps** | **Expected Outcome** |
| --- | --- | --- | --- |
| TC-001 | Validate GET /users endpoint. | 1. Send a GET request to /users. | Status 200; returns JSON with user details. |
| TC-002 | Test POST /register endpoint validation. | 1. Send a POST request with invalid email. | Status 400; returns validation error. |
| TC-003 | Verify API performance under load. | 1. Simulate 100 concurrent users sending requests to /users. | Response time < 500ms for all requests. |
| TC-004 | Test unauthorized access. | 1. Send a GET request to /users without an auth token. | Status 401; returns authentication error. |