GroupE4 Test Plan

BuzzNet

Group E4

Group Members:

LI Chun Leung 155193164
PENG Minqi 1155191548
WONG Kwok Kam 1155192018
ZENG Bai Chuan 1155193167
ZHANG Ka Sing 1155194769

Department of Computer Science and Engineering

Version 1.2

11 May 2025

Document Revision History

Version	Revised by	Revision Date	Comments
0.1	LI Chun Leung	8 May 2025	Initial draft and testing outline.
1.0	LI Chun Leung	10 May 2025	Major updates to all sections. Illustrate test cases and scenarios and emphasize on testing approach.
1.1	LI Chun Leung	11 May 2025	Revise document.
1.2	Wong Kwok Kam	11 May 2025	Review and finalize the document.

1. Test Plan for BuzzNet

This document serves as the Test Plan for *BuzzNet*, a web-based social media platform designed to facilitate user interactions through posts, comments, and follow functionalities. The purpose of this document is to outline the structured approach for validating the system against its functional and non-functional requirements.

The key sections of this document include:

1. **Scope and Objectives**: Defines the components to be tested (e.g., user authentication, post management) and explicitly excludes areas outside current priorities (e.g., GDPR compliance).
2. **Test Cases and Scenarios**: Provides concise, actionable test cases to evaluate core functionalities such as user registration, post creation, and role-based access control.
3. **Team Roles and Responsibilities**: Assigns tasks to team members, ensuring accountability for testing activities.
4. **Testing Approach & Timeline**: Details methodologies (unit, integration, system testing) and schedules to align with project deadlines.

This document prioritizes brevity and clarity, focusing on critical testing aspects to ensure *BuzzNet* meets its requirements while adhering to time and resource constraints.

1.1 1. Scope and Objective

This section outlines what will and will not be tested, helping to avoid scope creep and maintain focus throughout the testing process. It also establishes clear boundaries for the testing efforts.

1.1.1 Objective:

The primary objective of this Testing Document is to define the testing strategy, processes, and criteria for validating the *BuzzNet* social media platform. This document ensures:

- Comprehensive validation of functional and non-functional requirements.
- Identification of testing methodologies and tools.

- Clarity on tested components and gaps for future iterations.
- Alignment with project timelines, assumptions, and constraints.

1.1.2 Scope:

This section outlines the components and functionalities explicitly included in the testing process:

In-Scope Components:

- Frontend Functionality:
    - User registration, login, and profile management.
    - Post creation, deletion, and comment submission workflows.
- Backend Operations:
    - API endpoint validation (e.g., authentication, post/comment CRUD operations).
    - Database integrity checks (user data, posts, comments).
    - Role-Based Access Control (RBAC) enforcement for users and administrators.
- Security Features:
    - Input validation (e.g., empty posts/comments, SQL injection attempts).
- Performance Metrics:
    - Response times for UI actions (≤ 3 seconds) and complex queries (≤ 6 seconds).
    - Concurrent user handling (0 – 20 users).
- User Interface Compatibility:
    - Browser compatibility (Chrome, Firefox).
    - Screen resolution optimization (1920x1080).

1.1.3 Out of Scope

The following components are excluded from the current testing phase due to resource constraints, assumptions, or future roadmap priorities:

- Compliance Audits:
    - GDPR/HIPAA compliance verification (requires external legal/technical audits).
- Scalability Beyond SRS Requirements:
    - Stress testing for >100 concurrent users.
    - Cross-platform compatibility beyond specified OS/browsers.
- Third-Party Integrations:
    - Social media sharing or external APIs (not specified in SRS).
- Edge Case Scenarios:

- Recovery testing for server hardware failures.
- Long-term data retention and archival processes.

1.2 2. Test Cases and Scenarios

1.2.1 Test Cases for Functional Requirements

This section provides a concise summary of the types of test cases and scenarios to be executed, their purpose, and their alignment with project goals. This ensures that stakeholders, including non-technical team members, have a clear understanding of what is being tested without needing to delve into the code repository.

1. **Valid Registration (FR1.1)**
    - **Objective:** Verify successful account creation with valid email and password.
    - **Steps:**
        1. Navigate to the registration page.
        2. Enter a unique username and strong password.
        3. Submit the form.
    - **Expected Result:**
        - Account is created.
    - **Pass Criteria:** Account appears in the database.

2. **Duplicate User name Registration (FR1.1 Exception)**
    - **Objective:** Prevent registration with an existing username.
    - **Steps:**
        1. Attempt to register using a username already in the database.
        2. Submit the form.
    - **Expected Result:**
        - System displays an error: "User already exists."
    - **Pass Criteria:** Registration is blocked; error message is clear.

3. **Create Post (FR2.1)**
    - **Objective:** Test text post creation by a logged-in user.
    - **Steps:**
        1. Log in as a standard user.
        2. Navigate to the post creation page.
        3. Enter text and submit.
    - **Expected Result:**
        - Post appears on the feed view.

- **Pass Criteria:** Post is stored in the database and displayed.
- **Exceptional Case:**
  - **Steps:** Submit an empty post.
  - **Expected Result:** Error: "Post content cannot be empty."

4. **Delete Own Post (FR2.2)**
   - **Objective:** Verify users can delete their own posts.
   - **Steps:**
     1. Log in as a standard user.
     2. Navigate to the post.
     3. Click "Delete."
   - **Expected Result:**
     - Post and associated comments are removed from the database.
   - **Pass Criteria:** Post no longer visible; database record deleted.
   - **Exceptional Case:**
     - **Steps:** Attempt to delete another user's post.
     - **Expected Result:** Error: "You do not have permission to delete this post."

5. **Post Reaction (FR3.1)**
   - **Objective:** Verify users can "like" posts.
   - **Steps:**
     1. Log in as a standard user.
     2. Click the "Like" button with each post.
   - **Expected Result:**
     - Like count increments by 1.
   - **Pass Criteria:**
     - Like count updates in real-time.
     - Database records the like action.
   - **Exceptional Case:**
     - **Steps:** Attempt to like the same post again.
     - **Expected Result:** toggles to "unlike".
     - **Pass Criteria:** System prevents duplicate likes or handles toggling appropriately.

6. **Display Comment (FR6.2)**
   - **Objective:** Verify comments are displayed in chronological order with like counts.
   - **Steps:**
     1. Log in as a standard user.
     2. Navigate to a post with multiple comments.

- **Expected Result:**
    1. Comments appear in chronological order (oldest to newest).
    2. Each comment displays the correct number of likes.
- **Pass Criteria:** Order and like counts are accurate.
- **Exceptional Case:**
    - **Steps:** Submit a new comment and refresh the post.
    - **Expected Result:** New comment appears at the bottom of the list.

7. **User Like Comment (FR6.3)**
    - **Objective:** Verify users can like comments.
    - **Steps:**
        1. Log in as a standard user.
        2. Navigate to a comment.
        3. Click "Like".
    - **Expected Result:**
        - The number of likes is incremented by 1.
    - **Pass Criteria:** Comment like count is accurate.
    - **Exceptional Case:**
        - **Steps:** Attempt to like the same comment again.
        - **Expected Result:** toggles to "unlike".
        - **Pass Criteria:** System prevents duplicate likes or handles toggling appropriately.

1.2.2 Test Cases for Non-Functional Requirements

1.1 System Requirements

- **SYS01: Web-Based Support**
    - **Objective:** Verify compatibility with Chrome and Firefox.
    - **Steps:**
        1. Test core features (login, post creation) on Chrome and Firefox.
    - **Expected Result:** All features function identically across browsers.
    - **Pass Criteria:** No browser-specific errors.
- **SYS02: Language Support**
    - **Objective:** Ensure only English is supported.
    - **Steps:**
        1. Input non-English text in posts/comments.
    - **Expected Result:** System accepts input but provides no localization.
    - **Pass Criteria:** No multi-language features enabled.

1.2 Performance Requirements

- **PER01: Response Time**
  - **Objective:** Ensure UI requests ≤ 3s, complex queries ≤ 6s.
  - **Steps:** Simulate 50 users loading feeds.
  - **Expected Result:** Average response times meet thresholds.
  - **Pass Criteria:** 95% of requests comply.

- **PER02: Concurrent Users**
  - **Objective:** Support 0–30 concurrent users.
  - **Steps:** Simulate 30 users using JMeter.
  - **Expected Result:** No crashes or slowdowns.
  - **Pass Criteria:** System handles load without errors.

- **PER03: Resource Usage**
  - **Objective:** Ensure CPU <70%, memory <80%.
  - **Steps:** Monitor metrics during peak load.
  - **Expected Result:** CPU ≤70%, memory ≤80%.
  - **Pass Criteria:** Metrics stay within limits.

1.3 Security Requirements

- **SEC01: RBAC Enforcement**
  - **Objective:** Ensure role-based access control.
  - **Steps:** Standard user tries to access admin panel.
  - **Expected Result:** Access denied.
  - **Pass Criteria:** Unauthorized actions restricted.

- **SEC02: Compliance (GDPR/HIPAA)**
  - **Objective:** Verify data protection alignment.
  - **Steps:** Audit encryption and logs.
  - **Expected Result:** Data encrypted; logs capture critical events.
  - **Pass Criteria:** Processes align with standards (*Note: Requires external audit*).

- **SEC03: Logging & Monitoring**
    - **Objective:** Validate real-time logging.
    - **Steps:** Perform security actions (e.g., login attempts).
    - **Expected Result:** Logs generated with timestamps.
    - **Pass Criteria:** Logs are accessible and detailed.

1.3 3. Resource Allocation

1.3.1 Team Roles and Responsibilities

- **Project Lead:**
  - Oversee testing timelines and deliverables.
  - Coordinate team communication and SRS compliance.
- **Test Manager:**
  - Design test strategies and ensure alignment with SRS.
  - Monitor test execution and report outcomes.
- **Test Engineers and developers:**
  - Develop and execute test cases (functional/non-functional).
  - Document results and track defects.
- **Security & Compliance Lead:**
  - Validate security requirements (RBAC, GDPR/HIPAA alignment).
  - Audit logs and encryption standards.

1.3.2 Tools and Software

- **Test Management Tools:**
  - JIRA: For test case organization and tracking.
  - Google Sheets: Collaborative test planning.
- **Bug Tracking Tools:**
  - JIRA: Log and prioritize defects.
- **Performance Testing Tools:**
  - JMeter: Simulate concurrent users and measure response times.
  - cProfile: Test *Buzznet's* performance under load.
- **Compatibility Testing Tools:**
  - BrowserStack: Test cross-browser compatibility (Chrome, Firefox).
  - Responsive Design Checker: Validate 1920x1080 resolution.
- **Security Testing Tools:**
  - OWASP ZAP: Scan for vulnerabilities (SQL injection, XSS).
  - Wireshark: Monitor network traffic for compliance logging.
- **Additional Tools:**
  - Selenium: Automate UI tests.
  - Postman: API endpoint validation.
  - Git: Version control for test scripts.

1.3.3 Testing Environment

1.3.3.1 Environment Types:

- **Development Environment:**
    - **Purpose:** Used by developers for unit testing, initial debugging, and incomplete feature implementation.
    - Testing machine: Local machines (Windows 10/Ubuntu 22.04).
    - Partial feature access: Post creation without moderation tools).

- **Testing/Staging Environment:**
    - **Purpose:** Replica of production for system, integration, and regression testing.
    - **Setup:**
        - Staging server with identical configurations to production.
        - All features enabled (e.g., RBAC, post moderation).
        - Chrome and Firefox browsers for cross-compatibility checks.

- **Production Environment (for UAT):**
    - **Purpose:** Final validation during User Acceptance Testing (UAT) to mimic live deployment.
    - **Setup:**
        - Cloud-hosted (AWS/Azure, school-provided resources).
        - Full security protocols (encryption, RBAC, logging).
        - Optimized for 1920x1080 resolution and 0 – 100 concurrent users.

1.3.3.2 Environment Setup:

- **Hardware Requirements:**
    - **Compatibility Testing:**
        - PCs running Windows 10, Ubuntu 22.04, and Android 12 (per Assumption 3).
        - Devices with 1920x1080 resolution (per SRS Assumption 6).
    - **Performance/Stress Testing:**
        - Windows 10 (8-core CPU, 16GB RAM).
        - Cloud instances (AWS/Azure) for load testing (100+ concurrent users).

- **Software Requirements:**
    - **Application Builds:**
        - BuzzNet frontend (React.js) and backend (Node.js) codebase.
        - Database schema (MySQL) for data storage (SRS Section 3.2.4).
    - **Test Tools & Utilities:**
        - JMeter (performance testing), Selenium (UI automation), Postman (API validation).
        - OWASP ZAP (security scans), Git (version control).
    - **Access to Backend Systems:**
        - RESTful APIs for user authentication, post creation, and moderation.
        - Admin dashboard for RBAC enforcement (SRS FR4.1).
        - Real-time logging system (SRS SEC04).

- **Network Requirements:**
    - LAN/Wi-Fi with 100Mbps bandwidth (per SRS Assumption 5).

1.4 4. Testing Approach

1.4.1 Types of Testing

1. **Unit Testing**
    - **Objective:** Validate individual components.
    - **Examples:**
        - API endpoints (e.g., user registration, post creation).
        - Database queries (e.g., storing comments, deleting posts).
        - UI elements (e.g., buttons, forms).
2. **Integration Testing**
    - **Objective:** Test interactions between modules.
    - **Examples:**
        - Frontend-backend integration (e.g., UI triggers API requests).
        - Role-Based Access Control (RBAC) interactions (e.g., admin vs. user permissions).
3. **System Testing**
    - **Objective:** Verify end-to-end functionality from a user perspective.
    - **Examples:**
        - User journey: Registration → Post creation → Commenting.
        - Performance under load (0 – 100 concurrent users).
        - Security workflows (e.g. admin moderation).
4. **Regression Testing**

    - **Objective:** Ensure updates do not break existing features.
    - **Examples:**
        - Retest post deletion after a bug fix.
        - Validate login functionality after UI redesign.
5. **User Acceptance Testing (UAT)**
    - **Objective:** Confirm the system meets user expectations.
    - **Examples:**
        - Real users validate usability (e.g., intuitive navigation).
        - Stakeholders verify compliance with SRS requirements.

1.4.2 Methodologies

1. **Manual Testing**
    - **Purpose:** Validate UI/UX, edge cases, and ad-hoc scenarios.
    - **Examples:**
        - Exploratory testing for usability (e.g., error message clarity).
        - Security checks (e.g., RBAC enforcement, SQL injection attempts).
2. **Automated Testing**
    - **Purpose:** Efficiently execute repetitive or complex tests.
    - **Tools & Examples:**
        - **Selenium:** Automate UI workflows (e.g., post creation, comment submission).
        - **JMeter:** Simulate concurrent users for performance testing.
        - **Postman:** Validate API responses and error handling.
3. **Exploratory Testing**
    - **Purpose:** Uncover unexpected issues.
    - **Examples:**
        - Stress testing with invalid inputs (e.g., empty posts, duplicate emails).
        - Cross-browser testing (Chrome vs. Firefox behavior).

1.5 5. Timeline and Schedule

5.1 Waterfall Model

Testing follows a **sequential Waterfall Model**, with phases executed linearly after development milestones.

5.1.1 Dependencies:

- Completion of SRS sign-off .
- Finalization of high-level architecture.
- Availability of Windows servers and cloud resources (Assumption 5, 9).
- Timely resolution of defects logged during testing.

5.2 Key Phases and Timeline

1. **Test Planning & Preparation**
    - Review SRS v1.5 and finalize test strategy.
    - Develop test cases for functional (FR1.1–FR6.3) and non-functional (SYS, PER, SEC) requirements.
    - Set up a staging environment (Linux server, Chrome/Firefox browsers).
    - Alignment: SRS Sections 4–5.
2. **Unit Testing**
    - Validate individual components:
        - API endpoints (FR1.1: Registration, FR2.1: Post Creation).
        - Database operations (FR6.1: Comment Storage).
    - Alignment: FR1.1, FR2.1, FR6.1.
3. **Integration Testing**
    - Test frontend-backend interactions (e.g., post creation → API → database).
    - Validate RBAC workflows (FR4.2: Admin vs. user permissions).
    - Alignment: FR3.1, FR4.2, FR6.2.
4. **System Testing**
    - End-to-end user workflows (registration → post → comment → follow).
    - Performance testing (PER01: ≤3s response time, PER02: 100 concurrent users).
    - Security validation (SEC01: 2FA, SEC02: RBAC).
    - Alignment: SRS Sections 4.3, 5.2–5.3.
5. **User Acceptance Testing (UAT)**
    - Stakeholders validate usability and compliance with SRS.
    - Test real-world scenarios (e.g., content moderation, password reset).

- Alignment: SRS Section 1.1 (Overview).

6. **Bug Fixing & Regression Testing**
    - Resolve critical defects (e.g., login failures, post deletion errors).
    - Retest impacted features (FR2.2, FR4.1, FR6.3).
    - Alignment: FR2.2, FR4.1, FR6.3.
7. **Final Validation & Release**
    - Exhaustive testing of core features (registration, post creation, admin controls).
    - Prepare deployment package and user manual (DOC04).
    - Alignment: SRS Sections 3.2 (System Flow), 5.4 (Documentation).

1.6 6. Risk Assessment and Mitigation

1.6.1 Potential Risks

- **Resource Constraints**
    - **Impact:** Delays in testing due to limited server/cloud access.
    - **Mitigation:** Use backup local servers; prioritize critical tests.
- **Scope Creep**
    - **Impact:** Missed deadlines due to unapproved changes.
    - **Mitigation:** Enforce formal Change Requests.
- **Compliance Risks (GDPR/HIPAA)**
    - **Impact:** Legal penalties for non-compliance.
    - **Mitigation:** Schedule third-party audits; encrypt data and logs.
- **Performance Bottlenecks**
    - **Impact:** System failure under load.
    - **Mitigation:** Incremental load testing with JMeter; optimize queries.
- **Security Vulnerabilities**
    - **Impact:** Breaches due to weak RBAC.
    - **Mitigation:** OWASP ZAP scans; manual penetration testing.
- **Testing Environment Issues**
    - **Impact:** Flawed test results.
    - **Mitigation:** Validate environment setup against supported browsers and OS.

- **Team Communication Gaps**
    - **Impact:** Missed defects.
    - **Mitigation:** Daily stand-ups; JIRA task tracking.
- **Insufficient Test Coverage**
    - **Impact:** Undetected critical bugs.
    - **Mitigation:** Use traceability matrix; peer reviews.

1.7 7. Success Criteria

Clear criteria to evaluate testing progress and determine successful completion of objectives:

1. **Resolution of Critical/High-Priority Bugs**
    - All critical and high-severity defects (e.g., login failures, security vulnerabilities, data loss) are resolved and retested.
2. **Performance Benchmarks Met**
    - System meets SRS-defined thresholds:
        - Average UI response time ≤ 3 seconds (PER01).
        - Supports 50–100 concurrent users without degradation (PER02).
        - Resource usage (CPU < 70%, memory < 80%) under peak load (PER03).
3. **Positive User Acceptance Testing (UAT) Feedback**
    - End-users/stakeholders confirm:
        - Intuitive navigation (e.g., post creation, following users).
        - Compliance with functional requirements (FR1.1–FR7.1).
4. **All Test Cases Passed**
    - 100% of functional and non-functional test cases pass with no unresolved major issues.
    - Traceability matrix confirms full coverage of SRS requirements.
5. **Stakeholder Approval for Release**
    - Client and development team sign off on:
        - Compliance with SRS v1.5.
        - Readiness for deployment.

1.8 8. Reporting Requirements

Aligned with SRS Documentation Requirements (DOC01–DOC04).

1.8.1 Documentation

- **Test Case Repository:**
  - Test cases (FR1.1–FR7.1, SYS01–SEC04) stored in Git repository with clear descriptions.
- **Test Execution Reports:**
  - Daily logs of passed/failed tests, including performance metrics (PER01–PER03).
- **Bug Reports:**
  - Detailed defect descriptions, steps to reproduce, screenshots, severity (critical/high/medium), and status (open/resolved).
- **Weekly Progress Updates:**
  - Summary of testing progress, blockers, and risks shared with stakeholders.

1.8.2 Communication

- **Team Sync-Ups:**
  - Biweekly meetings with developers to address defects (e.g., RBAC issues).
- **Final Test Summary Report:**
  - Includes:
    - Test coverage matrix (SRS requirements vs. test cases).
    - Performance benchmarks (response times, concurrent users).
    - UAT feedback summary.
    - Stakeholder sign-off for release (per Success Criteria Section 7).