Test Plan

for

Buyzu

Version <1.0>

Prepared by

Group Name: E7

|  |  |  |
| --- | --- | --- |
| GUO Menglong | 1155191409 | 1155191409@cuhk.edu.hk |
| MA Heyang | 1155191433 | 1155191433@cuhk.edu.hk |
| XUE Guangxuan | 1155194956 | 1155194956@cuhk.edu.hk |
| YAN Yitao | 1155191529 | 1155191529@cuhk.edu.hk |
| ZHU Keyu | 1155191834 | 1155191834@cuhk.edu.hk |

|  |  |
| --- | --- |
| Instructor: | *Dr.LAM Tak Kei* |
| Course: | CSCI3100 |
| Date: | May 9, 2025 |
|  |  |

**Table of Content**

[1  Test Plan for Buyzu 3](#_Toc5623)

[1.1 Scope and Objectives 3](#_Toc5189)

[1.1.1 Scope: 3](#_Toc9214)

[1.1.2 Out of Scope: 3](#_Toc11797)

[1.1.3 Objectives: 3](#_Toc6266)

[1.2 Test Cases and Scenarios 4](#_Toc25086)

[1.2.1 Example Test Cases for Functional Requirements: 4](#_Toc17219)

[1.2.2 Example Test Cases for Non-Functional Requirements 2](#_Toc7731)

[1.3 Resource Allocation 4](#_Toc10390)

[1.3.1 Team Roles and Responsibilities 4](#_Toc12678)

[1.3.2 Tools and Software 5](#_Toc29023)

[1.3.3 Testing Environments 5](#_Toc26891)

[1.3.4 Time Allocation 6](#_Toc18128)

[1.3.5 Budget Allocation 6](#_Toc24536)

[1.4 Testing Approach 7](#_Toc15275)

[1.4.1 Types of Testing used for Buyzu 7](#_Toc14751)

[1.4.2 Methodologies 8](#_Toc14573)

[1.5 Timeline and Schedule 8](#_Toc1844)

[1.5.1 Key Phases &Timeline for testing (12 days) 8](#_Toc20666)

[1.5.2 Key Milestones: 9](#_Toc19527)

[1.6 Risk Assessment and Mitigation 10](#_Toc12727)

[1.7 Success Criteria 10](#_Toc6096)

[1.8 Reporting Requirements 11](#_Toc1595)

[1.8.1 Documentation 11](#_Toc27907)

[1.8.2 Communication 11](#_Toc9725)

# 1  Test Plan for Buyzu

## 1.1 Scope and Objectives

### 1.1.1 Scope:

This test plan governs the verification of the Buyzu online shopping platform, ensuring it meets its functional and non-functional requirements and is release-ready.The following areas will be tested:

* Core supporting mechanics including the following aspects:
* User account workflows (signup, login, password reset, SSO)
* Product discovery (search, filter, recommendation algorithm)
* Shopping cart operations (add, update, remove, partial checkout)
* Checkout operations (price adjustment, shipping selection, third-party gateways)
* Order tracking (real-time logistics API)
* Reviews & ratings submission and display
* REST API performance and reliability
* User interface (UI) and user experience (UX)
* Compatibility across supported platforms (e.g., PC, console, mobile).
* Audio and visual effects.
* Localization and language support.

### 1.1.2 Out of Scope:

* Testing of administrative interfaces for manipulating users and products.
* Extensive data import tools for product catalogs.
* Modding or hacking scenarios, including malicious attacking from different layers.
* Third-party API services beyond user authentication, payment and logistics.
* Pressure tests for unexpected, excessive scenarios (high concurrent peaks, etc.).

### 1.1.3 Objectives:

* Ensure platform performance within designed load limit.
* Assure the platform is user-friendly, intuitive, and free of critical design flaws.
* Validate that all core user privacy data and journeys are encrypted and stored.
* Confirm UI/UX design complies accessibility and other quality standards.
* Verify that security controls perform as anticipated.

## 1.2 Test Cases and Scenarios

This section summarizes representative test cases that cover Buyzu's major requirements.

### 1.2.1 Example Test Cases for Functional Requirements:

1. User Registration (via email):
   * Steps:
     + Navigate to ‘signup’ section
     + Enter the entire email address (must be valid and new to Buyzu, otherwise login/password reset)
     + Enter the high strength password
     + Confirm the password entered
     + Submit registration application
     + Verify email address through clicking the secure verification link sent

* + Expected Result: “Verification link sent” message; Active account after clicking link OR errors raise according to authentication mechanism of platform.
  + Pass/Fail Criteria: Account appears in the User table with verified tag if registered successfully.

1. User Registration (SSO):
   * Steps: Click “Login with SSO” -> Choose authentication service provider (e.g. Google) -> complete OAuth flow.
   * Expected Result: JWT returned and automatically redirected to dashboard.
   * Pass/Fail Criteria: JWT is valid, message code 200 received with correct userID.
2. User Login:
   * Steps:
     + Navigate to ‘login’ section
     + Enter the email address and password OR directly login through SSO
     + Confirm and submit
   * Expected Result: “Login successful” message; page redirected to homepage
   * Pass/Fail Criteria: The account exists in the database; The email address paired with password are valid; No incomplete/mismatched/forged information allowed.
3. Product Recommendation:
   * Steps: Autonomous without user operation.
   * Expected Result: Algorithms synthesize recommendations for products that may be of interest based on the user's past history.
   * Pass/Fail Criteria: Product thumbnails appear in the ‘Related to items you've viewed’ section of the dashboard.
4. Product Search & Filter:
   * Steps:
     + Enter the desired item name in the search box
     + Click on the associative entries in the drop-down box or simply press enter to jump to the matches
     + Apply the filter manually as wanted, e.g. price range, brand name, etc.
   * Expected Result: Results which associate with the keyword will be displayed.
   * Pass/Fail Criteria: No mismatches or data loss during searching and filtering.
5. Add/Update/Remove Shopping Cart Items:
   * Steps:
     + On a product’s detail page, click “Add to Cart” button.
     + Navigate to the “Shopping Cart” page and change the item quantity to designated number.
     + Refresh or revisit the “Cart” page and review the remaining items and the total price.
   * Expected Result: The cart correctly reflects the add, update, and remove actions.
   * Pass/Fail Criteria: The mechanism will handle and exclude some special cases (stock shortage, etc.).
6. Checkout & Payment:
   * Steps:
     + From the “Cart” page, click “Proceed to Checkout.”
     + On the checkout page, select a shipping address and delivery method, then confirm to pay.
     + Enter payment details on the payment page and submit.
     + Simulate the third-party payment success callback and return to the order confirmation page.
   * Expected Result: Order status shows “Processing”; payment status is “Success”; inventory for purchased items is decreased.
   * Pass/Fail Criteria: Database records (order, payment, inventory) and confirmation email are correct; otherwise fail
7. Order Tracking Update:
   * Steps:
     + Log in and go to “My Orders.”
     + Click into the newly placed order to view its details.
     + Wait for—or manually refresh—to see the shipping status update.
   * Expected Result: Order detail page updates to show status “Shipped” along with a tracking number.
   * Pass/Fail Criteria: Displayed status and tracking number match backend data; otherwise fail
8. Submit Review & Rating
   * Steps:
     + On a completed order, click “Write a Review.”
     + Select a 5-star rating, enter “Great!” in the comment field, and submit.
     + Return to the product’s detail page and switch to the “Reviews” tab.
   * Expected Result: The new review appears with five stars and the text “Great!” at the top of the list.
   * Pass/Fail Criteria: Review is persisted and visible on the product page; otherwise fail

### 1.2.2 Example Test Cases for Non-Functional Requirements

Non-functional requirements focus on the quality attributes of the Buyzu online shopping platform, such as performance, scalability, usability, reliability, and security.

**1. Performance Testing**

1.Homepage Load Time

Objective: Verify that the homepage loads within the acceptable time limit.

Steps:

Launch the software on a supported platform.

Measure the time taken to load the main menu from the splash screen.

Expected Result: The game loads within 2 seconds on high-end devices and 3 seconds on low-end devices.

Pass/Fail Criteria: Pass if the load time is within the acceptable range in 95% of total attempts; fail otherwise.

2.API Response Under Load

Objective: Ensure each request latency <200 ms at 1 000 requests/min.

Steps:

Configure and start a test plan using JMeter to launch 1,000 requests per minute to the Buyzu target page.

Collecting and analyze response latency data.

Expected Result: 99% of p95 measurements <200 ms, error rate <1%.

Pass/Fail Criteria: Metrics within threshold; fail otherwise.

**2. Usability Testing**

1.Navigation Intuitiveness

Objective: Confirm a new user can complete “add to cart → checkout” in under 3 minutes unaided.

Steps:

Recruit 10 participants unfamiliar with Buyzu.

Instruct each: “Find any product, add it to your cart, and complete checkout.”.

Observe and time each session.

Expected Result: ≥ 9 of 10 users succeed within 3 min without help.

Pass/Fail Criteria: ≥ 90% success; fail otherwise

2.Browser Compatibility

Objective: Verify UI renders and functions correctly on different mainstream web browsers (e.g. Chrome, Firebox, Safari, Edge, etc.).

Steps:

Using BrowserStack, load Buyzu Homepage, Product, Cart, and Checkout.

Perform basic interactions (search, add to cart, checkout).

Note any JS console errors or visual defects.

Expected Result: No console errors; no layout or functional regressions.

Pass/Fail Criteria: No critical issues; fail otherwise

**3.Security Testing**

1.Authentication & Authorization

Objective: Verify protected endpoints reject requests without valid JWT.

Steps:

Call Buyzu webpage without Authorization header.

Call again with an expired or malformed token.

Expected Result: status code: 401 Unauthorized.

Pass/Fail Criteria: Both responses are handled correctly; fail otherwise

## 1.3 Resource Allocation

This section outlines the personnel, tools, environments, and other resources required to execute the test plan effectively. Proper resource allocation ensures that the testing process is eﬀicient, well-organised, and capable of meeting the project’s objectives within the given time- line. In larger organisations, there is typically a specialised team responsible for testing.

### 1.3.1 Team Roles and Responsibilities

Based on the system architecture and component design, the testing team roles and responsibilities are defined as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Name | Responsibilities | Time Commitment |
| Test Lead | GUO Menglong | Develop test strategies, coordinate the team, and report progress. Oversee test coverage for core functionalities (e.g., payment, cart operations). | Full-time |
| QA Tester | MA Heyang | Execute manual test cases (e.g., user authentication, cart updates, checkout flow). Log defects and validate fixes. | Full-time |
| Automation Engineer | XUE Guangxuan | Execute manual test cases (e.g., user authentication, cart updates, checkout flow). Log defects and validate fixes. | Full-time |
| Developer | YAN Yitao | Assist in integration testing (e.g., payment gateway and order service interactions). Fix backend defects. | Full-time |
| UI/UX Designer | ZHU Keyu | Validate responsive layouts, accessibility (WCAG compliance), and cross-device compatibility (PC/mobile). | Full-time |

### 1.3.2 Tools and Software

Testing tools and frameworks are selected based on the technical stack and design requirements:

**1.Testing Tools:**

1. Test Management:

Tool: GitHub Issues

Track test cases and defects. Link to code repositories (GitHub).

2. Automation:

Tool: Postman (API testing), Cypress (E2E UI testing), JUnit (unit testing).

Validate RESTful APIs, Vue.js components, and Spring Boot services.

3. Performance:

Tool: JMeter

Simulate high concurrency (e.g., flash sales) and stress test system load.

4. Security:

Tool: OWASP ZAP

Check security of payment APIs and encryption mechanisms.

5. Database:

Tool: MySQL Workbench

Verify ACID transactions (Section 5.2.5) and data consistency (e.g., inventory deduction).

6. Compatibility:

Tool: BrowserStack

BrowserStack Test cross-browser (Chrome/Firefox/Safari) and mobile (iOS/Android) compatibility

### 1.3.3 Testing Environments

This subsection defines the types of testing environments, their configurations, and example allocations to ensure compatibility with the system architecture and design specifications outlined in the Design and Implementation Documentation.

**1.3.3.1 Environment Types：**

1. Development：

- Local development setups (IDEs, Docker containers).

- Incomplete features or mock services.

- Direct access to developers for rapid iteration.

2. Staging/Testing:

- Mirrors production infrastructure (server clusters, databases).

- Sandboxed third-party APIs (e.g., Alipay/WeChat Pay test endpoints).

- Automated test execution pipelines.

3. Production (UAT):

- Identical to live production environment.

- Limited access to core team only.

- Real-world data sanitized for testing.

**1.3.3.2 Environment Setup:**

Hardware/Software Configurations:

1. Development:

- IDEs: IntelliJ, VS Code.

- Docker for containerized MySQL and Redis.

- Postman, JUnit, and Vue DevTools.

1. Staging/Testing:

- Jenkins/GitHub Actions for CI/CD.

- JMeter for performance testing.

- BrowserStack for cross-platform validation.

1. Production (UAT):

- Monitoring: Prometheus/Grafana.

- Security: OWASP ZAP, TLS 1.3 encryption.

- Alipay/WeChat Pay live APIs (test mode).

### 1.3.4 Time Allocation

|  |  |  |
| --- | --- | --- |
| Activity | Effort (%) | Responsible Team Member(s) |
| Test Planning | 10% | Test Lead |
| Test Case Creation | 20% | QA Testers |
| Test Execution | 40% | QA Testers, Automation Engineers QA |
| Bug Fixing & Retesting | 20% | Developers |
| Regression Testing | 10% | QA Testers, Automation Engineers |

### 1.3.5 Budget Allocation

This subsection details the budget considerations for personnel, tools, hardware, and training.

1. Personnel Costs

As a student project, there are no direct salary expenses. However, the effort distribution across roles is tracked to ensure accountability. The Test Lead oversees test planning and coordination on a full-time basis, while QA Testers and Automation Engineers contribute part-time efforts to manual and automated testing, respectively. Developers assist as needed, focusing on resolving integration-level defects. All roles operate on a volunteer basis, with time logs maintained to monitor progress and align responsibilities with the testing phases defined in Section 1.3.4.

1. Tool Licences

Testing tools are selected based on free tiers, open-source alternatives, or educational discounts. BrowserStack is used for cross-browser and device compatibility testing, accessed via a free educational license secured through the university. Postman, employed for API testing and automation, utilizes its free tier to validate endpoints such as /api/payment and /api/cart. Performance testing relies on the open-source tool JMeter to simulate high concurrency scenarios. Security testing is conducted with OWASP ZAP, integrated into CI/CD pipelines for vulnerability scanning. GitHub Actions provides free CI/CD automation, supporting up to 2,000 build minutes per month for test execution.

1. Tool Licences:

Hardware costs are minimized through shared university resources and virtualization. The staging environment deploys AWS EC2 at an estimated cost of $20/month to test backend services under load. Local development environments use university-provided machines (8GB RAM, quad-core CPUs) at no additional cost.

1. Tool Licences:

Training resources focus on free or low-cost options to upskill the team. Cypress documentation and interactive tutorials are utilized for frontend end-to-end testing, while Postman’s Student Program provides guided learning paths for API testing best practices. JMeter’s official documentation and community guides are referenced to configure performance tests. Security testing fundamentals are covered through OWASP ZAP’s GitHub wiki, which includes workshops and use-case examples. These resources ensure the team can efficiently use the selected tools without incurring additional expenses.

## 1.4 Testing Approach

The testing approach for Buyzu will employ a comprehensive, multi-layered strategy to ensure all aspects of the software meet quality standards and requirements. This approach combines various testing types and agile principles, suitable for Buyzu's microservices architecture and iterative development.

### 1.4.1 Types of Testing used for Buyzu

1. **Unit Testing**

* Goal:Validate individual components (classes,method,modules).
* Targets: Core functionalities (e.g., password hashing, product search)
* Method: Pytest;Black-box and white-box testing

2. **Integration Testing**

* Goal: Verify interactions between different modules, services, and external systems.
* Targets: Frontend-backend API integration, inter-service communication (e.g., Order to Payment Service), and third-party integrations (Payment Gateways, Logistics APIs).
* Method: Black-box and white-box testing using tools like Postman, REST Assured. Mocks/Stubs for isolating dependencies.

3. **System Testing**

* Goal: Validate the integrated Buyzu system against specified requirements from an end-user perspective.
* Targets: End-to-end user scenarios.(e.g., registration->choose items->checkout)
* Method: Primarily black-box testing(simulating real user scenarios).Presure testing.

4. **Regrassion Testing**

* Goal: Ensure throughout the development process, that new changes or bug fixes do not negatively impact existing functionality.
* Targets: Any changes made->potentially every components.
* Method:Automated test suites (unit, integration)& manual checks of critical paths after changes made.

5. **Validation Testing**

* Goal: Ensure that Buyzu satisfys the specified business requirements
* Targets: Following requirements document.
* Method:Black-box testing&Manually simulating.

6. **User Acceptance Testing (UAT)**

* Goal: Ensure Buyzu meets users’ needs&expectations, and ready for deplotments.
* Method:Gather feedback from users to make specifical modification.

### 1.4.2 Methodologies

* **Agile Testing:** Testing is integrated iteratively throughout the developing process, with early involvement of testers in requirements and design. Continuous feedback and collaboration are emphasized.
* **Manual Testing:** Manually simulate user case to ensure smooth user experience
* **Automated Testing:** Pytest; usefor repetitive tasks (e.g., regression testing)..
* **Black-box&White-box testing**
* **Exploratory testing** : To uncover edge cases and unexpected issues.

## 1.5 Timeline and Schedule

This part involves a testing timeline that includes key milestones & time arrangement for differnt testing phases with our agile developing process.

The testing timeline for Buyzu will align with the Agile methodology, with testing activities distributed into each Sprint(development iteration), and occurs iteratively throughout the project lifecycle.

### 1.5.1 Key Phases &Timeline for testing (12 days)

| **Day** | **Activity** |
| --- | --- |
| Day 1 | Sprint planning, define acceptance criteria for user stories, write initial test cases for User Registration (email and SSO) and Login features based on requirements. |
| Day 2 | Set up test environment with sample user and product data, begin testing User Registration flows including email verification process and SSO authentication paths. |
| Day 3 | Execute User Login test cases, verify JWT token validation and security, start testing Product Recommendation algorithm with various user history scenarios. |
| Day 4 | Complete Product Recommendation testing, begin Product Search & Filter test cases including keyword matching and filter application, log any search-related bugs. |
| Day 5 | Perform mid-sprint integration testing between frontend and backend components, conduct Authentication & Authorization security testing on protected endpoints, test API Response Under Load. |
| Day 6 | Execute Shopping Cart test cases (add/update/remove items), verify correct handling of edge cases like stock limitations, test Homepage Load Time across devices. |
| Day 7 | Begin Checkout & Payment flow testing including shipping selection and third-party payment gateway integration, verify inventory updates after purchase. |
| Day 8 | Complete Checkout & Payment testing, verify order status updates and confirmation emails, begin Order Tracking Update testing with logistics API integration. |
| Day 9 | Test Submit Review & Rating functionality, conduct Navigation Intuitiveness usability testing with participants, perform Browser Compatibility testing across Chrome, Firefox, Safari, and Edge. |
| Day 10 | Sprint review/demo of working features, prepare test summary report highlighting test coverage and discovered issues across functional and non-functional requirements. |
| Day 11 | Post-sprint regression testing focusing on core user journeys from registration through checkout, update automated test suite with new scenarios for tested features. |
| Day 12 | Backlog refinement for upcoming sprint, document lessons learned from current sprint testing, make corresponding modification&updates. |

### 1.5.2 Key Milestones:

* Testing is continuous and iterative throughout each sprint& the whole process, with each sprint delivers a potentially shippable product increment with tested features and no separate testing phase defined.
* Key testing milestones align with sprint boundaries and include:
* **Sprint Planning & Preparation Milestone**s: Complete test case development and acceptance criteria for all sprint user stories.
* **Initial Testing Milestone**: Successfuly validate User Registration, Login, Product Recommendation, and Search functionality.
* **Mid-Sprint Testing Milestone**:Finish the verification of integration points, security controls, and collect performance metrics.
* **Further Ket Features Testing Milestone:**Complete some further testing for Shopping Cart, Checkout, Payment, and Order Tracking.
* **End-of-Sprint Testing Milestone :**Finish usability testing, browser compatibility, and derive a test coverage report.
* **Post-Sprint Milestone:**Successfully execute regression testing, make some corresponding updates base on feedback.

## 1.6 Risk Assessment and Mitigation

This section identifies potential risks during the development and operation of the Buyzo platform and uses Fault Tree Analysis (FTA) to model and analyze the causes of system failure. Mitigation strategies are also provided for each identified risk.

**Description of the Fault Tree**

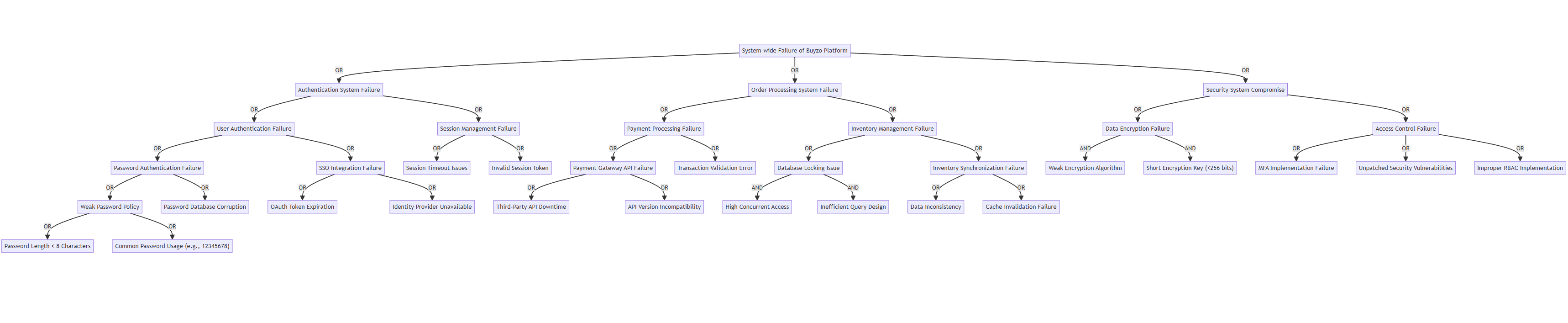
1. **Top Event:**The system failure represents a complete disruption of the Buyzo platform, affecting core functionalities such as user authentication, order processing, and data security.
2. **Sub-Events:**
   * Authentication Failure: Users cannot log in or access their accounts.
     + Weak Password Policy: Users are allowed to set weak passwords, making accounts vulnerable to brute-force attacks.
     + SSO Integration Failure: Issues with third-party identity providers or expired OAuth tokens.
   * Order Processing Failure: Orders cannot be processed or completed.
     + Payment Gateway Failure: Issues with third-party payment APIs or incorrect payment data.
     + Inventory Sync Failure: Problems with database locking or concurrency handling during high traffic.
   * Data Breach: Unauthorized access to sensitive user or system data.
     + Insufficient Encryption: Weak encryption algorithms or short encryption keys.
     + Unauthorized Access: Lack of MFA or unpatched vulnerabilities exploited by attackers.
3. **Basic Events:**These are the root causes of failures, such as weak passwords, expired tokens, API downtime, and lack of proper encryption.

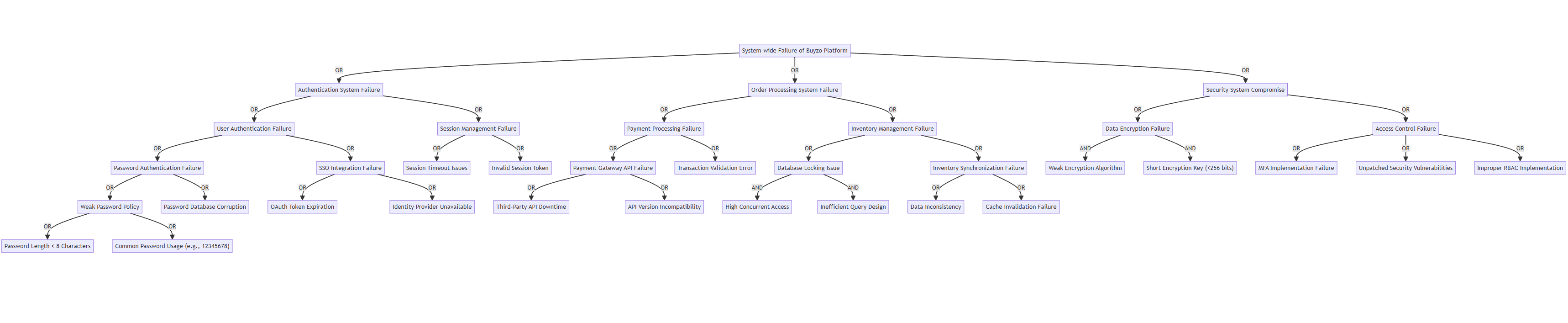
**Minimal Cut Sets**

From the fault tree, the minimal cut sets represent the smallest combinations of basic events that can lead to the top event (system failure). For example:

1. G,H*G*,*H*: Weak password policy (short passwords or common passwords).
2. I,J*I*,*J*: SSO integration issues (expired tokens or unavailable identity providers).
3. M,N*M*,*N*: Payment gateway failures (API downtime or incorrect data).
4. S,T*S*,*T*: Insufficient encryption (weak algorithm or short key).
5. U,V*U*,*V*: Unauthorized access (no MFA or unpatched vulnerabilities).

**Figure**





## 1.7 Success Criteria

The success of the Buyzo platform will be determined based on the following criteria:

1. **Bug Resolution**
   * All critical and high-priority bugs are resolved before the release.
   * Medium and low-priority bugs are documented and scheduled for future updates.
2. **Performance Benchmarks**
   * The platform meets performance requirements, including:
     + Response time under 2 seconds for critical functions under normal load.
     + Handling 10,000 transactions per minute with a success rate of 99%.
3. **User Acceptance Testing (UAT) Feedback**
   * Positive feedback is received from UAT participants, confirming the platform is intuitive and meets user expectations.
   * No major usability or functionality issues are reported during UAT.
4. **Compliance with Non-Functional Requirements**
   * The platform adheres to security, scalability, and reliability standards defined in the SRS document.
   * Accessibility features are implemented and verified.

## 1.8 Reporting Requirements

This section outlines how testing results and progress will be documented and communicated to stakeholders.

### 1.8.1 Documentation

1. **Test Case Repository**
   * All test cases will be stored in a GitHub repository with detailed descriptions, expected outcomes, and pass/fail criteria.
2. **Test Execution Reports**
   * Reports summarizing the status of executed test cases, including:
     + Total test cases executed.
     + Number of passed/failed test cases.
     + Defects identified and their severity.
3. **Bug Reports**
   * Detailed bug reports will be created for each defect, including:
     + Steps to reproduce.
     + Expected vs. actual results.
     + Severity and priority levels.
     + Screenshots or videos for visual reference.
4. **Final Test Summary Report**
   * A comprehensive report will be prepared before the release, covering:
     + Test coverage and execution statistics.
     + Summary of resolved and unresolved defects.
     + Recommendations for the release decision.

### 1.8.2 Communication

1. **Regular Meetings**
   * Weekly meetings with the development and testing teams to discuss progress, blockers, and priorities.
2. **Stakeholder Updates**
   * Weekly progress updates shared with stakeholders, including:
     + Completed tasks.
     + Identified risks and mitigation plans.
     + Upcoming milestones.
3. **Post-Release Reporting**
   * A post-release report will be prepared to evaluate the platform's performance in the production environment, including:
     + User feedback.
     + Any post-release issues or incidents.
     + Recommendations for future improvements.