WK4: QUALITY ASSURANCE AND SOFTARE TEST PLAN

Part 6

Anila Naz

University of Phoenix

BSA-425: BSIT Capstone

Professor: Dr. Reid

23rd. July 2024

Project Name: Fintech LLC Internet Bank

Purpose of Project: Internet Bank to offer a secure, remote, financial platform to target a widespread diversity population in catering remote banking without any schedule or locality constraints.

Software Test Plan

Introduction

The software test plan outlines the testing strategy for the Internet Bank project, detailing the features to be tested, the testing methodology, and the scope of testing. This document ensures that all critical functionalities are thoroughly validated and provides a clear understanding of what will and will not be tested.

Features To Be Tested

The following features of the Internet Bank project will be tested:

1. **User Authentication**
    - **Login and Logout**: Verify that users can log in and log out securely using valid credentials.
    - **Multi-Factor Authentication (MFA)**: Test the MFA setup and authentication process.
2. **Account Management**
    - **View Account Details**: Ensure users can view account details, including balance and transaction history.
    - **Update Personal Information**: Test the functionality for users to update their personal information.
3. **Transactions**

- **Deposit and Withdrawal**: Validate the functionality for depositing and withdrawing funds, including error handling for insufficient funds.
- **Transaction History**: Ensure users can view their transaction history accurately.

4. **Security Features**
   - **Data Encryption**: Test the encryption of sensitive data both at rest and in transit.
   - **Session Management**: Verify session timeouts and security measures to prevent session hijacking.
5. **Customer Support**
   - **Contact Support**: Test the functionality for users to contact customer support through the application.
6. **Fraud Detection**
   - **Transaction Monitoring**: Test the system's ability to detect and flag suspicious transactions.
7. **CRM Integration**
   - **Customer Interaction Logging**: Verify that customer interactions are logged correctly in the CRM system.
8. **Data Warehouse Integration**
   - **Data Sync**: Ensure that data is correctly synchronized with the data warehouse for reporting purposes.

Features Not to Be Tested

The following features will not be tested in this phase, along with explanations:

1. **Mobile Application Functionality**

- **Reason**: The current phase of the project focuses on web application functionality. The mobile app will be tested in a subsequent phase.

2. **Third-Party API Integrations (e.g., Payment Gateways** or email integration**)**
    - **Reason**: These integrations are handled by third-party vendors, and it is assumed they have been tested by the respective vendors. Our focus will be on testing our interface with these APIs.
3. **Stress and Load Testing**
    - **Reason**: The initial phase focuses on functional testing. Stress and load testing will be conducted in a later phase to evaluate the system's performance under high demand.
4. **UI/UX Design Consistency**
    - **Reason**: Usability testing is planned for a later stage. The current phase emphasizes functionality and security testing.
5. **Non-critical Features**
    - **Reason**: Features deemed non-critical for the initial release, such as advanced reporting tools or customization options, will not be tested in this phase.
6. **Extremely Low-Frequency Features**: Features used by a negligible portion of customers, e.g., specific tax-related transactions for businesses.
7. **Backup and Disaster Recovery**
    - **Reason**: The backup and disaster recovery plan will be tested during the system's maintenance phase. The focus is currently on ensuring basic functionalities are working correctly.

Conclusion

This test plan provides a focused approach to testing the critical functionalities of the Internet Bank project. By specifying what will and will not be tested, the plan ensures that resources are efficiently utilized, and that the testing effort aligns with the project's priorities and timelines. Future phases will address the features not covered in this initial testing scope.

Testing Pass/Fail Criteria Framework

The pass/fail criteria for each test in the Internet Bank project are defined on the bases of functionality, accuracy, usability, performance, security, and compliance to ensure clear and measurable standards for evaluating the success of the testing process. These criteria specify the conditions under which a test is considered to have passed or failed.

1. User Authentication

- **Login and Logout**
    - **Pass**: The user can log in with valid credentials and is directed to the correct landing page. The system successfully authenticates valid users, rejects invalid login attempts, sensitive data is encrypted during transmission and storage. System detects and prevents unauthorized access. The user can log out successfully, and the session is terminated.
    - **Fail**: The user cannot log in with valid credentials, is incorrectly directed, or cannot log out properly. The system fails to authenticate valid users, allows invalid login attempts, sensitive data is not encrypted or is exposed. The system fails to detect or prevent unauthorized access.
- **Multi-Factor Authentication (MFA)**
    - **Pass**: The user is prompted for MFA after entering correct login credentials and can access the account upon providing correct MFA details.

    - **Fail**: The user is not prompted for MFA, the MFA process fails, or the account is accessible without MFA.

2. Account Management

- **View Account Details**
    - **Pass**: The user can view accurate and complete account details, including balance and transaction history.
    - **Fail**: Account details are incomplete, inaccurate, or not displayed.
- **Update Personal Information**
    - **Pass**: The user can update personal information, and changes are correctly reflected in the system.
    - **Fail**: The user cannot update information, or updates are not saved or displayed correctly.

3. Transactions

- **Deposit and Withdrawal**
    - **Pass**: Transactions are processed correctly, balances are updated accurately, and transaction records are stored properly. Transactions are confirmed with correct transaction details e.g., date, time, amount.
    - **Fail**: Transactions fail to process, balances are incorrect, or records are not updated. Transaction detail is incorrect or missing.
- **Transaction History**
    - **Pass**: Users can view a complete and accurate history of their transactions.
    - **Fail**: Transaction history is incomplete, inaccurate, or inaccessible.

4. Security Features

- **Data Encryption**
    - **Pass**: Data is encrypted according to standards (e.g., AES-256), and encrypted data cannot be read without proper decryption keys.
    - **Fail**: Data is not encrypted or is easily compromised.
- **Session Management**
    - **Pass**: Sessions expire correctly after a period of inactivity, and session tokens are secure.
    - **Fail**: Sessions do not expire as expected or are vulnerable to hijacking.

5. Customer Support

- **Contact Support**
    - **Pass**: Users can successfully contact support, and support requests are logged and acknowledged.
    - **Fail**: Users cannot contact support, or requests are not logged or acknowledged.

6. Fraud Detection

- **Transaction Monitoring**
    - **Pass**: The system accurately detects, and flags suspicious transactions based on predefined criteria.
    - **Fail**: The system fails to detect suspicious transactions or generates excessive false positives.

7. CRM Integration

- **Customer Interaction Logging**
    - **Pass**: All customer interactions are accurately logged in to the CRM system.
    - **Fail**: Interactions are missing, incorrect, or not logged.

8. Data Warehouse Integration

- **Data Sync**
    - **Pass**: Data is correctly synchronized with the data warehouse, and reports reflect accurate and up-to-date information.
    - **Fail**: Data is not synchronized properly, leading to inaccuracies in reports.

General Pass/Fail Criteria for All Tests

- **Pass**
    - All functionalities work as specified in the requirements document.
    - The system handles expected edge cases without failure.
    - No critical or high-severity bugs are present.
- **Fail**
    - The system exhibits unexpected behavior that violates specifications.
    - Critical or high-severity bugs are present.
    - The system fails to handle expected edge cases or produces incorrect outputs.

The testing process ensures that only features meeting the pass criteria are approved for release, maintaining the quality and reliability of the Internet Bank project.

Testing Approach for Internet Bank Project

The testing approach for the Internet Bank project encompasses a comprehensive strategy that includes various testing processes, types, and methods. This approach ensures thorough validation of the system's functionality, performance, security, and usability.

1. Testing Processes

The testing processes involve several iterative steps and will follow a structured sequence:

1. **Test Planning**: Involves creating a comprehensive test plan defining the scope, objectives, resources, deliverables, and schedule for testing activities.
2. **Test Design**: Based on test plan, develop a detailed test case, test scripts, and scenarios with clear inputs, expected outputs, and pass/fail criteria per requirements and design documents.
3. **Test Environment Setup**: The necessary hardware software, and data infrastructure will be established to simulate real-world conditions.
4. **Test Execution**: Perform testing according to the test cases and document the results. Defects will be logged and tracked.
5. **Defect Reporting and Tracking**: Log and track defects, retest fixes, and ensure resolution.
6. **Test Closure**: Evaluate the test results, prepare a test summary report, and archive test artifacts. Final testing reports and metrics will be generated.

2. Testing Types

The project will incorporate the following testing types to ensure comprehensive coverage:

1. **Unit Testing**
    - **Purpose**: Verify that individual components or modules of the application work as intended.
    - **Method**: Conducted by developers using automated testing tools or frameworks.
2. **Integration Testing**
    - **Purpose**: Test the interactions between different integrated units, components, or systems to identify interface defects.

- **Method**: Both automated and manual testing will be used to validate data flow and interaction logic.

3. **Functional Testing**
    - **Purpose**: Validate that the application functions according to the specified requirements.
    - **Method**: Manual testing based on predefined test cases, covering all critical functions.
4. **System Testing**
    - **Purpose**: Evaluate the entire system as a complete unit and integrated software to ensure it meets the specified requirements.
    - **Method**: A combination of automated and manual testing, covering end-to-end scenarios.
5. **Security Testing**
    - **Purpose**: Identify vulnerabilities and ensure the application is protected against threats.
    - **Method**: Conduct vulnerability assessments, penetration testing, and secure code reviews.
6. **Performance Testing**
    - **Purpose**: Determine the responsiveness, stability, and scalability of the application under rigorous load conditions.
    - **Method**: Load testing, stress testing, and endurance testing using specialized tools.
7. **User Acceptance Testing (UAT)**

- **Purpose**: Validate that the application meets business needs and is ready for production.
- **Method**: End-users perform testing using real-world scenarios and workflows.

8. **Regression Testing**
    - **Purpose**: Ensure that new changes do not adversely affect existing functionalities.
    - **Method**: Automated regression testing scripts run after code changes.
9. **Usability Testing**
    - **Purpose**: Evaluate the user interface and user experience to ensure ease of use and accessibility.
    - **Method**: Conducted with end-users, observing their interactions and gathering feedback.

3. Testing Methods

The testing methods employed include:

1. **Manual Testing**
    - Used for exploratory testing, usability testing, and tests that require human judgment.
    - Involves testers executing test cases without the use of automation tools.
2. **Black-Box Testing**: Testing the system without knowledge of its internal structure.
3. **White-Box Testing**: Testing the system with knowledge of its internal structures.
4. **Gray-Box Testing**: A combination of black-box and white-box testing.
5. **Exploratory Testing**: Testing without predefined test cases, relying on tester's knowledge and intuition.
6. **Automated Testing**

- Used for regression testing, performance testing, and repetitive tasks.
- Involves using automated tools and scripts to execute test cases, often integrated into the CI/CD pipeline.

4. Testing Strategy

The testing strategy is designed to ensure thorough and efficient testing, including:

1. **Risk-Based Testing**
   - Prioritize testing based on the potential impact and likelihood of defects or failures. Focus on critical features and high-risk areas.
2. **Early Testing**
   - Start testing activities early in the development lifecycle, including unit and integration testing during development.
3. **Test Automation**
   - Automating repetitive tests cases wherever feasible, especially for regression and performance testing, to improve efficiency and coverage.
4. **Continuous Testing**
   - Integrate testing into the CI/CD pipeline, ensuring that code changes are tested continuously and automatically for faster feedback.
5. **Cross-Functional Collaboration**
   - Engage developers, testers, business analysts, and end-users in the testing process to ensure comprehensive understanding and coverage.
6. **Defect Tracking and Management**: Implementing a robust defect tracking system.
7. **Test Metrics**: Collecting and analyzing test metrics to measure test effectiveness.
8. **Feedback Loop**

- Maintain a feedback loop with the development team, enabling quick resolution of defects and continuous improvement.

Application of Testing Strategy:

The testing strategy will be applied throughout the software development lifecycle, starting with unit testing during development and progressing through integration, system, and user acceptance testing. Security, performance, and usability testing will be conducted at specific intervals based on project risks and requirements.

By employing this structured, rigorous, and comprehensive testing approach, the Internet Bank project aims to deliver a reliable, secure, and user-friendly application leading to a high quality and reliable product that meets all business and technical requirements.

15 Testing Cases for the Internet Bank:

Test Cases

1. User Login with Valid Credentials

- **Objective**: Verify that a user can log in with valid credentials.
- **Preconditions**: The user account exists in the system.
- **Steps**:
    1. Navigate to the login page.
    2. Enter a valid username and password.
    3. Click the "Login" button.
- **Expected Result**: The user is logged in and redirected to the account overview page.

2. User Login with Invalid Credentials

- **Objective**: Verify that the system prevents login with invalid credentials.
- **Preconditions**: The user account exists, but the entered credentials are incorrect.

- **Steps**:
    1. Navigate to the login page.
    2. Enter an invalid username or password.
    3. Click the "Login" button.
- **Expected Result**: The user is not logged in, and an error message is displayed.

3. Multi-Factor Authentication (MFA) Verification

- **Objective**: Verify that MFA is required and functions correctly. Test login credentials. Test lockout mechanism after multiple failed attempts. Verify password recovery functionality
- **Preconditions**: MFA is enabled for the user account.
- **Steps**:
    1. Log in with valid credentials.
    2. Enter the MFA code sent to the user's device.
- **Expected Result**: The user is logged in after entering the correct MFA code.

4. View Account Balance

- **Objective**: Ensure users can view their account balance accurately.
- **Preconditions**: User is logged in.
- **Steps**:
    1. Navigate to the account overview page.
- **Expected Result**: The correct account balance is displayed.

5. Deposit Funds

- **Objective**: Verify that a user can deposit funds into their account.
- **Preconditions**: User is logged in and on the deposit page.

- **Steps**:
    1. Enter a valid deposit amount.
    2. Select a source account.
    3. Click "Deposit."
- **Expected Result**: The deposit is processed, and the account balance is updated accordingly.

6. Withdraw Funds

- **Objective**: Verify that a user can withdraw funds from their account.
- **Preconditions**: User is logged in and on the withdrawal page.
- **Steps**:
    1. Enter a valid withdrawal amount.
    2. Select a destination account.
    3. Click "Withdraw."
- **Expected Result**: The withdrawal is processed, and the account balance is updated accordingly.

7. Insufficient Funds for Withdrawal

- **Objective**: Ensure that the system prevents withdrawals that exceed the account balance.
- **Preconditions**: User is logged in and the withdrawal amount exceeds the available balance.
- **Steps**:
    1. Attempt to withdraw an amount greater than the available balance.
- **Expected Result**: The system prevents the transaction and displays an error message.

8. Transaction History Display

- **Objective**: Verify that users can view their transaction history.
- **Preconditions**: User is logged in and has completed transactions.
- **Steps**:
    1. Navigate to the transaction history page.
- **Expected Result**: All transactions are accurately listed with details such as date, amount, and type.

9. Update Personal Information

- **Objective**: Test the functionality for updating user personal information.
- **Preconditions**: User is logged in.
- **Steps**:
    1. Navigate to the personal information page.
    2. Update details (e.g., address, phone number).
    3. Save changes.
- **Expected Result**: The system saves the updates and displays a confirmation message.

10. Secure Session Timeout

- **Objective**: Ensure sessions expire after a period of inactivity for security.
- **Preconditions**: User is logged in.
- **Steps**:
    1. Leave the session inactive for a predefined period.
    2. Attempt to perform an action after the timeout period.
- **Expected Result**: The session expires, requiring re-authentication.

11. Password Change Functionality

- **Objective**: Verify that users can change their passwords securely. Enforce password policy, password update, and password history policy,
- **Preconditions**: User is logged in.
- **Steps**:
    1. Navigate to the password change page.
    2. Enter the current and new passwords.
    3. Enforce the user to select long, complex, unique, alphanumeric, and non-alphanumeric string, that is not repeated before.
    4. Save changes.
- **Expected Result**: The password is changed, and the user is notified.

12. Contact Customer Support

- **Objective**: Test the functionality for contacting customer support.
- **Preconditions**: User is logged in.
- **Steps**:
    1. Navigate to the support page.
    2. Submit a query or request.
- **Expected Result**: The query is logged, and a confirmation is sent to the user.

13. Detect and Flag Suspicious Transactions

- **Objective**: Ensure the system flags transactions that appear suspicious based on predefined criteria.
- **Preconditions**: User is logged in.
- **Steps**:

1. Perform a transaction that meets the criteria for being flagged (e.g., large amount, unusual pattern).

- **Expected Result**: The transaction is flagged, and an alert is generated.

14. Data Encryption Verification

- **Objective**: Verify that sensitive data is encrypted both at rest and in transit.
- **Preconditions**: Access to the database and network traffic monitoring tools.
- **Steps**:
    1. Monitor data storage and transmission.
- **Expected Result**: Sensitive data is encrypted, and unauthorized access to plaintext data is not possible.

15. Cross-Browser Compatibility

- **Objective**: Ensure the application functions correctly across different web browsers.
- **Preconditions**: Access to multiple web browsers (Chrome, Firefox, Safari, Edge).
- **Steps**:
    1. Perform key functionalities on each browser.
- **Expected Result**: The application behaves consistently and as expected across all tested browsers.

Testing Materials – Hardware/Software Requirements

For the Internet Bank project, a comprehensive set of resources is required to ensure effective and thorough testing. These resources include physical facilities, hardware, software, tools, and other essential elements needed for various testing activities.

1. Physical Facilities

- **Testing Environment**: A dedicated space equipped with testing workstations, network access, and necessary infrastructure.
- **Security Measures**: Access controls and monitoring systems to protect sensitive data and ensure a secure testing environment.

2. Hardware Requirements

- **Testing Workstations**:
    - Multiple workstations with varying configurations to simulate different user environments. These should include different operating systems (e.g., Windows, macOS, Linux) to ensure compatibility and coverage.
    - Specifications: At least 8 GB RAM, Intel i5 or equivalent processor, SSD storage, high-resolution monitors.
- **Servers**:
    - Servers for hosting the test environment, including application servers, database servers, and web servers.
    - Specifications: High availability and performance, with sufficient processing power, memory, and storage capacity.
- **Mobile Devices**:
    - A range of smartphones and tablets with different operating systems (e.g., iOS, Android) and screen sizes for testing mobile compatibility (if applicable).
- **Network Equipment**:
    - Routers, switches, and firewalls for simulating different network conditions and ensuring secure testing.

3. Software Requirements

- **Operating Systems**:
    - Various operating systems to test cross-platform compatibility (e.g., Windows 10, macOS, Linux distributions).
- **Browsers**:
    - Multiple web browsers for cross-browser testing, including Chrome, Firefox, Safari, Edge, and others.
- **Database Systems**:
    - Database management systems used in the project (e.g., MySQL, PostgreSQL, MariaDB) for testing data interactions and storage.
- **Automation Tools**:
    - Tools for automating testing processes, LambdaTest – a cloud-based AI-powered test orchestration and execution platform accelerate speed and efficiency with end-to-end test orchestration platform with high-end test automation on the cloud. Supports various frameworks. Selenium (open-source suite of tools and libraries) for web scraping, programmatically extract data from websites, web applications across browsers like Safari, Edge, Firefox, Chrome and operating systems like Windows, macOS, and Linux with CI/CD for broader compatibility and a smooth user experience. Playwright – enables parallel browser testing. Other choices are Cypress – frontend test automation in JavaScript, no extra dependencies, real-time support. Appium used to test native, hybrid, and mobile webapps. XCUITest modify app's accessibility behavior for disables. JUnit/TestNG (for unit testing), and JMeter (for performance testing).
- **Security Testing Tools**:

    - Tools for vulnerability scanning and penetration testing, such as OWASP ZAP, Burp Suite, and Nessus, Wireshark, Nmap – checking for open ports. VAPT tools automatically detect system vulnerabilities and generate pen-testing reports as an infrastructure IT administrator.
- **Continuous Integration/Continuous Deployment (CI/CD) Tools**:
    - Jenkins, GitLab CI/CD, or similar tools for automating builds and integrating testing into the development pipeline.
- **Development Tools**: IDEs and compilers for developers to make necessary code changes.
- **Version Control System**:
    - Git or other version control systems for managing code and test scripts.
- **Bug Tracking and Project Management Tools**:
    - Jira, Bugzilla, or similar tools for tracking defects, managing test cases, and coordinating testing activities.

4. Additional Resources

- **Test Data**:
    - Simulated data sets, including both normal and edge cases, to test various functionalities and scenarios.
    - Realistic and representative data for testing various scenarios
- **Documentation**:
    - Access to project documentation, including requirements specifications, design documents, user manuals, and test case documentation.
- **Testers and Personnel**:

    - A skilled team of subject matter expert testers for domain knowledge and expertise, including specialists in manual testing, automation testing, security testing, and performance testing.
- **Training and Support**:
    - Training materials and support resources for testers to familiarize themselves with the tools, processes, and specific aspects of the system being tested.
- **Backup and Recovery Systems**:
    - Systems for data backup and recovery to protect against data loss during testing.

By ensuring that these resources are available and well-coordinated, the testing team can effectively validate the Internet Bank project, ensuring it meets the required quality standards and functions as expected in various environments.

Testing Schedule:

Testing Activity	Duration	Resource	Comments
Test Planning	5 days	Test Manager QA Lead	Define scope, objectives, & resources
Test Specification Creation	2 weeks	Test Leads	Develop detailed test specifications
Test Specification Team Review	5 days	Project Team, Developers, BA	Review and refine test specifications.
Test Case Development	10 days	QA analysts, Test engineers	Create detailed test cases and scripts.
Test Environment Setup	5 days	IT Support, DevOps Team	Prepare hardware, software, and networks

Unit Testing	Ongoing	Developers	Automated, part of development lifecycle.
Component Testing	2 weeks	Developers, QA Team	Test individual components in isolation.
Integration Testing	2 weeks	Test Engineers, Developers	Focus on data flow and module interfaces.
Functional Testing	1 week	Test Engineers	Validate application against requirements.
System Testing	2 weeks	QA Team	End-to-end testing of the entire system
Security Testing	1 week	Security Analysts	Pen-testing, vulnerability scanning.
Performance Testing	1 week	Performance testers, IT support	Load stress, endurance testing
Alpha Testing	2 weeks	Internal Users, QA Team	Early testing by internal staff.
Beta Testing/Pilot program	2weeks	Selected End Users, QA Team	Testing in a controlled real-world environment
User Acceptance Testing UAT	1 week	End Users, UAT Coordinators	Verify system meets business needs.
Regression Testing	1 week	QA Team, Automation, Engineers	Ensure new changes don’t break existing functionality. Critical after any code changes to prevent new bugs.
Usability Testing	1 week	UX Designer Testers	Evaluate user interface and experience.

Defect Reporting and Fixing	Ongoing	Testers, Developers	Continuous throughout testing phases.
Test Closure & Reporting	1 week	QA Lead, Test Manager	Summarize results, review test coverage.

Risks and Contingencies Matrix:

Risk	Probability	Risk Type	Owner	Contingencies/Mitigation approach
Data Breach	30 %	Security	Security Team	Implement strong encryption, MFA, regular security audits, & employee training on data handling.
Project Delays	50%	Schedule	Project Manager	Monitor project progress closely, maintain a flexible schedule and allocate buffer time for unexpected issues.
Budget Overrun	25%	Financial	Finance Manager	Regular budget reviews, cost monitoring, and identifying cost-saving measures where possible.

System Down	20%	Operational	IT Operations	Set up robust infrastructure, redundancy, regular maintenance schedules, & a disaster recovery plan
Compliance & Regulatory Changes	35%	Legal Regulatory	Compliance Officer	Stay updated on regulations, consult with legal experts, and adjust project plans to stay compliance.
Technical Debt	60%	Technical	Development Team	Regular code reviews, refactoring, and maintaining clear documentation to mange & reduce technical debt.
Inadequate User Training	40%	User Experience	Training Coordinator	Develop comprehensive training programs, provide user manuals, and conduct workshops, tabletop exercises or webinars.

Stakeholder Misalignment	35%	Organizational	Project Sponsor	Regular stakeholder meetings, transparent communication, & involve key stakeholders in decision-making processes.
Scope Creep	60%	Scope	Project Manager	Clearly define project scope, implement process, & prioritize requests based on impact & feasibility.
Vendor Reliability Issues	20%	Vendor/Supplier	Procurement Officer	Thorough vendor vetting, regular performance review, & have backup vendors & solutions.
Poor Performance Under Load	40%	Performance	QA Lead	Conduct thorough performance testing, optimize code, infrastructure, & plan for scalability

Inadequate Testing Coverage	30%	Quality	QA Team	Implement use of automated testing tools, critical functionalities, & comprehensive test plan.
Loss of Key Personnel	15%	Resource	HR Manager	Develop a knowledge transfer plan, maintain thorough documentation & a succession plan in place.
Customer Dissatisfaction	30%	Customer Satisfaction	Customer Support	Implement a robust feedback system, regular surveys, & a clear escalation proves for resolving issues.
Integration Issues with Legacy System	35%	Technical	Integration Team	Early assessment of integration requirements, thorough testing, having fallback plans for data synchronization issues.
Not Enough Skilled workers for testing	30%	Resource	HR Manager, QA Lead	Cross-Train team members, hire temporary skilled contractors, and

				develop a resource allocation plan.
Testing Team Member Turnover	20%	Resource	HR Manager, QA Lead	Maintain a comprehensive onboarding process, retain detailed documentation and have a backup plan for critical roles.

REFERENCES:

CISA (CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY). (n.d.). *Cybersecurity Best Practices*. https://www.cisa.gov/topics/cybersecurity-best-practices

CIS Center for Internet Security. (n.d.). *CIS Critical Security Controls*. https://www.cisecurity.org/controls

FINRA. (n.d.). *Cybersecurity*. https://www.finra.org/rules-guidance/key-topics/cybersecurity

Interesting Engineers. (2020, July 25). *What Is Software Testing and What Types of It Are There?* https://interestingengineering.com/lists/what-is-software-testing-and-what-types-of-it-are-there

LAMBDATEST. (n.d.). *58 Top Automation Testing Tools for 2024*. https://www.lambdatest.com/blog/automation-testing-tools/

MSRC. (n.d.). *Microsoft Security Response Center*. https://msrc.microsoft.com/

NIST. (n.d.). *CYBERSECURITY FRAMEWORK*. https://www.nist.gov/cyberframework

OWASP. (n.d.). *Explore the world of cyber security*. https://owasp.org/

SANS.org. (n.d.). *Empowering Cyber Security Practitioners & Teams*. https://www.sans.org/

Singer, P. W., & Friedman, A. (2014). *Cybersecurity and Cyberwar: What Everyone Needs to Know*. Oxford University Press. https://whateveryoneneedstoknow.com/display/10.1093/wentk/9780199918096.001.0001/isbn-9780199918096-book-part-1