CS631G, CRN 74069 – Team 3



SYSTEM TEST PLAN

Xero

(Cloud-based Accounting System)

Authors: Avinash Bhavancheekar, Rushab Ramesh Vaza, Ruthwik Mamidala, Jensi Jasoliya, Aasritha Bhimisetty, Arepally Shivaly, Darshna, Shivani Nallu, Sumedh Anumala

QA Manager: Avinash Bhavancheekar

Date: 12/13/2024

**Table of Contents**

[Introduction (Avinash Bhavancheekar) 3](#_Toc181477445)

[Test Items (Ruthwik) 3](#_Toc181477446)

[Features to be tested (Rushab) 3](#_Toc181477447)

[Features not to be tested (Sumedh) 4](#_Toc181477448)

[Approach (Darshna) 5](#_Toc181477449)

[Item Pass/Fail Criteria (Ruthwik Mamidala) 6](#_Toc181477450)

[Suspension Criteria and resumption requirements (Aasritha) 7](#_Toc181477451)

[Test deliverables (Shivani) 9](#_Toc181477452)

[Testing tasks (Shivani) 11](#_Toc181477453)

[Environmental needs (Jensi Jasoliya) 12](#_Toc181477454)

[Responsibilities (Sumedh) 12](#_Toc181477455)

[Staffing and training needs (Shivali) 12](#_Toc181477456)

[Schedule (Darshna) 16](#_Toc181477457)

[Risks and contingencies (Rushab) 17](#_Toc181477458)

# Introduction (Avinash Bhavancheekar)

This document provides the Test Plan for the Xero Cloud-Based Accounting Application, outlining a comprehensive strategy for testing its features and functionalities. The primary objective of this Test Plan is to ensure that stakeholders are well-informed about the testing process, which aims to guarantee the quality, reliability, and performance of Xero.

The Test Plan details the test items, features to be tested, the testing approach, entry and exit criteria, suspension and resumption criteria, test deliverables, testing activities, and resource requirements. It also includes information regarding the testing environment, team roles, scheduling, and identified risks with corresponding contingency plans.

By following the structured testing strategy outlined in this document, the aim is to provide stakeholders with confidence that the Xero application meets its intended requirements and is ready for deployment to end users.

# Test Items (Ruthwik)

This Test Plan Document will include the following modules in the Xero application:

1. Purchase Orders
2. Expense Claims
3. Accounts Payable

# Features to be tested (Rushab)

This section lists core features that will be tested, grouped by the relevant application modules.

**Purchase Orders**

* Purchase Order Creation
  + Verify that users can create a new purchase order with accurate details, including item descriptions, quantities, and pricing.
* Purchase Order Submission
  + Test the submission of purchase orders to suppliers and validate that notifications are generated appropriately.
* Purchase Order Tracking
  + Confirm that users can track the status of purchase orders from creation to completion.
* Purchase Order History
  + Ensure users can view their history of purchase orders and access specific order details as needed.

**Expense Claims**

* Expense Claim Submission
  + Verify that users can submit expense claims with itemized entries, descriptions, and attached receipts where required.
* Expense Claim Approval Workflow
  + Test the system’s ability to route expense claims through the appropriate approval or rejection workflow.
* Expense Claim Editing
  + Ensure that users can edit or update claims before submission and that they cannot modify approved or rejected claims.
* Expense Claim History
  + Confirm that users can view the status and history of past expense claims, with details on approvals or rejections.

**Accounts Payable**

* Bill Entry
  + Verify that users can enter bills from suppliers, including details such as due dates, amounts, and any relevant descriptions.
* Payment Scheduling
  + Test that users can schedule payments for bills, set reminders, and mark payments as complete.
* Accounts Payable Aging Report
  + Ensure that the system can generate an aging report for accounts payable, providing a breakdown of overdue and upcoming bills.
* Bill Tracking and History
  + Confirm that users can view the status and history of bills, track payment progress, and access records of past transactions.

# Features not to be tested (Sumedh)

**1. Purchase Order Module:**

* Customization Options: Xero allows users to personalize some settings, which may vary significantly based on individual preferences. Testing these customizations is out of scope, as they are highly specific and do not impact core functionalities.
* Rare Edge Cases: Certain edge cases, like exceptionally high or low purchase quantities that are unlikely to occur under typical usage, will be excluded. These scenarios are deemed low-risk for most users and will be prioritized for later testing if needed.
* Third-party Integrations: The Purchase Order module might integrate with various third-party tools. Since these integrations can vary widely and are often managed by external vendors, they will not be included in this testing cycle. Testing will focus on Xero's in-built capabilities, while integration testing will be delegated to relevant vendors.

**2. Expense Claim Module:**

* Deprecated Features: Certain older features that are set to be removed in future updates are excluded from testing. Testing these would provide little long-term benefit, as they will soon be unavailable to users.
* Advanced Reporting Functions: The Expense Claim module includes some advanced reporting functions primarily used by a minority of users. These reports will not be included in the core testing scope, as they have minimal impact on the application’s primary functions.
* Mobile-Only Features: Any functionality that is unique to the mobile application version will not be tested here, as this testing phase focuses on the desktop and web application features. Mobile testing may be scheduled separately if needed.

**3. Accounts Payable Module:**

* Regional Settings Not in Scope: The Accounts Payable module includes region-specific settings for taxes and compliance, which may not be required for the primary user regions of this deployment. These will not be included in the testing scope.
* Optional Invoice Layouts: Xero offers customizable invoice layouts; however, these custom templates do not impact system performance or functionality. Testing will focus on standard templates, as custom designs are user-specific.
* Automated Notifications: Customizable automated notifications and email alerts are not included, as they are optional features. Unless directly impacting core functionality, they will be tested at a later stage, focusing now on primary operations.

# Approach (Darshna)

**4.1 Approach to Functional Testing**

The functional testing of Xero software will be carried out using black-box testing techniques, focusing on user-oriented scenarios for validating the software's features. This method leverages external functional specifications, such as accounting workflows, invoicing, reconciliation, and financial reporting, to design test cases that mimic real-world usage.

Functional components of Xero can be categorized into several business logic types, including:

* User Interface (UI) and Navigation: Testing navigational elements, responsiveness, and layout consistency.
* Field Validations and Edits: Verifying the input fields' behavior with boundary values, acceptable ranges, and valid/invalid data.
* Field Dependencies and Interactions: Ensuring that field changes dynamically impact other fields, as expected.
* Business Rules and Calculations: Testing financial calculations, tax applications, and other core accounting functions according to business logic.

Test conditions will be designed using techniques such as boundary value analysis, equivalence partitioning, and decision tables to ensure comprehensive coverage. Detailed logic for each category will be captured in the test design specification.

**4.2 Approach to Non-Functional Testing**

Non-functional testing for Xero will focus on validating the software’s performance, usability, and compatibility under various scenarios. Using a black-box approach from the user perspective, the following will be addressed:

* Performance Testing: Simulating volume scenarios with heavy data loads, such as multiple transactions and large data imports, to ensure that Xero can handle typical accounting workloads efficiently.
* Compatibility Testing: Verifying Xero’s operation across different browsers, devices, and operating systems to confirm cross-platform compatibility.
* Usability Testing: Assessing user experience with key tasks to ensure intuitive design and ease of navigation, especially for financial professionals.
* Security and Data Privacy: Testing access controls, data encryption, and session management to ensure that sensitive accounting data is protected.

Each type of non-functional testing will be documented with specific test objectives and metrics, ensuring that the Xero software meets both functional and non-functional requirements effectively.

# Item Pass/Fail Criteria (Ruthwik Mamidala)

To ensure the Xero application meets quality standards, the following Exit Criteria must be achieved before completing the testing phase:

* **Test Design Completeness**: All functional requirements have matching test cases, covering every essential feature and workflow.
* **Test Execution Completeness**: All test cases must be executed, ensuring each planned scenario has been verified.
* **Product Stability**: No critical or high-severity defects remain open; only minor issues that do not affect primary functionality are permissible.

Core Feature Pass/Fail Criteria:

1. **Purchase Orders**: All test cases for creating, editing, deleting, searching, filtering, and approving purchase orders must execute without critical issues. Minor issues can be documented but should not hinder functionality.
2. **Expense Claims**: Every step, from creating and editing to approval and payment of claims, should pass without high-severity defects. All interactions should follow expected behaviors to support accurate expense tracking.
3. **Vendor Bills and Contacts**: Features like vendor bill recording, management, and contact functionality must perform as expected, with successful completion of all test cases and no severe issues affecting usability.

# Suspension Criteria and resumption requirements (Aasritha)

**Suspension Criteria:**

1. **Critical Bugs:** 
   * **Xero-Specific Example:** A critical bug is discovered that causes Xero to crash when attempting to reconcile bank transactions, preventing further testing of the bank reconciliation feature.
   * **Impact:** This bug significantly impacts the core functionality of Xero and could lead to data loss or incorrect financial reporting for users.
2. **Environment Downtime:** 
   * **Xero-Specific Example**: The Xero testing environment is undergoing scheduled maintenance, making it unavailable for testing activities.
   * **Impact:** This downtime prevents testers from executing test cases and verifying the functionality of new features or bug fixes.
3. **Resource Unavailability:** 
   * **Xero-Specific Example:** Key testing team members are unavailable due to illness or vacation, limiting the available testing resources.
   * **Impact:** This lack of resources can delay the testing process and potentially impact the release timeline.

**Resumption Criteria:**

1. **Critical Bugs Resolved:** 
   * **Xero-Specific Example:** The critical bug causing the bank reconciliation crashes has been fixed and verified through regression testing.
   * **Impact:** This ensures that the core functionality of Xero is restored, and testing can resume without fear of data corruption or system instability.
2. **Environment Restored:** 
   * **Xero-Specific Example:** The Xero testing environment is back online and fully operational, with all necessary systems and data restored.
   * **Impact:** This allows testers to access the environment and execute test cases, ensuring that the application functions as expected.
3. **Resource Availability:** 
   * **Xero-Specific Example:** The key members of the testing team are back from leave and are available to resume their testing activities.
   * **Impact:** This ensures that the testing process can proceed efficiently and effectively, without any delays or bottlenecks.

**Xero-Specific Considerations:**

* **Data Integrity:** Any suspension of testing activities must prioritize the protection of sensitive financial data.
* **User Impact**: The impact of any testing suspension on Xero users, especially during critical accounting periods, must be carefully considered.
* **Risk Assessment:** A thorough risk assessment should be conducted to evaluate the potential impact of each suspension scenario and develop appropriate mitigation strategies.
* **Communication:** Effective communication between the testing team, development team, and other stakeholders is crucial to ensure a smooth resumption of testing activities.

By carefully considering these factors, Xero can effectively manage testing activities and minimize disruptions to the development and release process.

**Example: Testing Activity Suspension and Resumption in Xero**

**Scenario: Critical Bug in Bank Reconciliation**

* **Suspension:**
  + A critical bug is discovered in the bank reconciliation feature, causing the system to crash frequently and preventing accurate reconciliation of bank transactions.
  + **Impact:** This significantly impacts the ability of businesses to reconcile their accounts, potentially leading to financial errors and delays.
  + **Action:** The testing team suspends further testing of the bank reconciliation feature to allow developers to address the critical bug.
* **Resumption:**
  + The development team successfully fixes the critical bug and deploys a hotfix to the testing environment.
  + **Verification:** The testing team conducts regression testing to ensure that the bug has been fixed and that the bank reconciliation feature is functioning correctly.
  + **Resumption of Testing:** Once the regression testing is completed and the bug is confirmed to be fixed, the testing team resumes testing of the bank reconciliation feature.

# Test deliverables (Shivani)

1.Test Plan

Purpose and Scope: Defines the roadmap for testing, specifying the goals, scope, and objectives. This includes covering the Purchase Order, Expense Claim, and Accounts Payable modules, and areas like workflow approvals and error handling.

2.Test Design Specifications

Overview of Coverage: Lays out the specific objectives, scenarios, and conditions that need to be tested across different modules, such as input validations, error messages, and integrations.

3.Test Case Specifications

Detailed Structure: Lists test cases, including step-by-step instructions, input and output expectations, and success criteria. This specification includes test cases for module features like approval workflows and vendor management.

4.Test Logs

Logging Details: Records every test execution with timestamps, tester names, and results. Logs should also note any environmental or conditional changes observed during testing.

5.Test Incident Reports

Issue Documentation: Documents any unexpected results or issues encountered, along with reproducibility steps and severity ratings.

6.Test Summary Reports

Test Outcome Overview: Summarizes all testing activities, including counts of passed, failed, and unresolved tests. Highlights specific observations or high-risk areas per module.

7.Test Input and Output Data

Data Sets for Validation: Identifies standardized input and output data, like sample orders and expense claims, to ensure consistent validation and tracking.

8.Test Tools

Tool Requirements: Lists any necessary tools, drivers, or stubs used to simulate complete environments and module interactions.

# Testing tasks (Shivani)

1. Develop Test Cases for Functional and Integration Testing

Supports comprehensive test case creation for each module, ensuring each feature—like purchase order creation, vendor management, expense claim submission, etc.—is thoroughly covered. This aligns directly with the features to be tested by creating structured cases for every required functionality.

1. Prepare Test Data

Necessary for verifying each module feature under realistic data conditions, such as different vendor profiles, expense types, and purchase orders. Test data preparation ensures scenarios are well-represented and aligns with data-dependent features like Expense Claim Submission and Vendor Management.

1. Set Up Test Environment

Creates an environment conducive to testing integrations across modules (e.g., purchase orders linking with accounts payable). This setup is vital for testing how features interact, as mentioned in Integration with Other Modules.

1. Execute Functional Testing

Directly supports feature validation, from Purchase Order Creation and Expense Claim Submission to Invoice Processing. Functional testing ensures that each feature operates correctly in isolation.

1. Conduct Integration Testing

Focuses on verifying interactions between modules, such as integration with inventory management, accounts payable, and general ledger modules. This task ensures all cross-module integrations are functioning as intended.

1. Test User Roles and Entitlements

Verifies that user permissions work as expected, supporting workflows like purchase order approval and expense claim approval. This aligns with role-based access to ensure only authorized users can perform certain actions.

1. Perform Error Handling Tests

Essential for features like Error Handling specified in each module (Purchase Order, Expense Claim, and Accounts Payable), ensuring the application handles invalid inputs and displays appropriate error messages.

1. Log Test Results and Generate Reports

Logs each test outcome, creating a record of all tested features and any failed tests. This documentation supports accountability for each feature tested and helps trace back issues in the features to be tested list.

1. Audit Logging Verification

Supports features like purchase order approval workflows and vendor management by ensuring that critical actions are logged, which is particularly important for compliance and tracking in financial applications.

# Environmental needs (Jensi Jasoliya)

# The test environment for the Xero application should closely match the production setup and include the following:

# **Hardware**: Sufficient servers, workstations, and network equipment to support the application and testing tools effectively.

# **Software**: All necessary software, including the Xero application, operating systems, databases, and any third-party integrations required for full functionality.

# **Testing Tools**: Tools for managing and tracking tests, such as JIRA, along with any specialized software needed for performance, load, or security testing.

# Security Requirements

# **Security Controls**: High-level security must be implemented to protect access to the test facilities, system software, and proprietary components (software, data, and hardware). Only authorized personnel should have access to sensitive areas.

# Additional Testing Needs

# **Special Tools**: Any additional tools specific to testing requirements, such as automated testing scripts or security scanning tools, should be available.

# **Other Resources**: Publications, office space, or specific documentation needed for testing should be accessible.

# **Sourcing Unavailable Items**: Identify and document the source of any items that are currently unavailable to ensure timely access.

# This setup aims to provide a complete and secure testing environment that mirrors production, allowing for thorough testing of the Xero application.

# Responsibilities (Sumedh)

# 

**1. QA Manager:**

* **Primary Oversight and Quality Assurance:** The QA Manager will oversee the entire testing process, managing the test team and ensuring that testing adheres to project timelines and quality standards. This includes setting priorities, identifying test requirements, and ensuring the team has the resources needed to conduct testing effectively.
* **Resource Allocation and Task Assignment:** Responsible for assigning tasks to each team member, the QA Manager will ensure all modules (Purchase Orders, Expense Claims, Accounts Payable) receive adequate test coverage. This also includes coordinating work schedules to maximize team productivity.
* **Stakeholder Communication:** The QA Manager will maintain open communication with stakeholders, providing regular updates on test progress, any issues encountered, and adjustments to timelines as necessary. This role involves managing expectations and delivering insights from test findings to the development and project management teams.

**2. QA Analyst:**

* **Requirement Analysis and Test Planning:** The QA Analyst will review the project requirements to ensure the testing team fully understands the expected outcomes. This includes aligning test cases with user expectations and defining clear acceptance criteria.
* **Test Case Review and Validation:** Working closely with the testing team, the QA Analyst will review test cases to ensure they are accurate, comprehensive, and relevant to the specified requirements. This process will involve feedback sessions to refine test scenarios and improve test coverage.
* **Defect Management and Prioritization:** The QA Analyst will oversee the process of logging defects, categorizing issues by severity, and coordinating with developers for timely resolution. This role includes tracking defects from identification through to resolution and keeping records for accountability.

**3. Business Analyst:**

* **Requirement Clarification and Documentation:** The Business Analyst will clarify any ambiguous requirements by liaising with project stakeholders and translating their feedback into actionable details for the testing team.
* **Support for Test Documentation:** Assisting the QA team, the Business Analyst will help ensure that documentation (such as test cases and scenarios) remains aligned with business requirements. This support is crucial for maintaining accurate and usable documentation throughout the testing cycle.
* **User Acceptance Testing (UAT) Coordination:** In addition to requirement clarification, the Business Analyst will help coordinate UAT efforts, collecting feedback from end-users to verify that the application meets their needs. This will support final approval of tested modules before deployment.

**4. Software Tester:**

* **Test Execution Across Modules:** Software Testers will carry out manual and automated testing of each module, following the test plan and ensuring that each function operates as intended. This includes identifying any issues with the application’s functionality, usability, and reliability.
* **Detailed Reporting and Incident Logging:** After each test execution, testers will log outcomes and document any issues in detail. This documentation provides a record of anomalies encountered during testing, making it easier for the QA Analyst and development team to troubleshoot and resolve issues.
* **Regression Testing for Fixes:** Upon identifying and resolving defects, Software Testers will conduct regression tests to ensure that recent fixes do not introduce new issues. This is a crucial step in maintaining application stability after code changes.

# Staffing and training needs (Shivaly)

1. QA Manager (Senior-Level Role)

* Staffing Needs:
  + Leadership & Project Management: The QA Manager will coordinate and oversee all testing activities across the team, ensuring testing is executed on time, efficiently, and with high-quality standards.
  + Risk Management & Quality Assurance Expertise: Must have a strong background in quality assurance best practices and risk management to proactively identify and mitigate potential issues.
  + Cross-Functional Collaboration: This role will require close collaboration with other departments, such as development and product management, to ensure alignment between testing processes and project goals.
* Training Needs:
  + Advanced Project Management & Risk Management: Training on advanced project management and risk management strategies for effective handling of large-scale, complex testing projects.
  + Continuous Improvement & Test Process Optimization: Focused training to continuously enhance testing workflows and implement new methodologies for optimized test execution and reporting.

2. QA Analysts (Mid- to Senior-Level Roles)

* Staffing Needs:
  + Test Case Development & Execution: Responsible for developing comprehensive test cases, executing functional, integration, and regression tests, and documenting defects.
  + Domain Knowledge of Accounting Workflows: Familiarity with financial processes such as Purchase Orders, Expense Claims, and Accounts Payable, essential for validating business requirements within Xero.
  + Role-Based Access & Security Testing: Proficiency in understanding role-based access within Xero, verifying permissions, and ensuring security protocols are effective.
* Training Needs:
  + Xero Application Features: Detailed training on Xero’s core features to support accurate testing of financial workflows and accounting processes.
  + Advanced Black-Box Testing Techniques: Training in techniques like boundary value analysis, equivalence partitioning, and decision tables, tailored to the accounting domain.
  + Non-Functional Testing: Specialized training in non-functional testing areas, including performance, usability, and security testing, to ensure the application meets scalability, performance, and data privacy standards.

3. Business Analysts (Mid-Level Role)

* Staffing Needs:
  + Financial Systems Expertise: In-depth understanding of financial systems and accounting workflows to ensure accurate translation of business requirements into testable specifications.
  + Business-Technical Liaison: Act as a bridge between business stakeholders and the QA team, ensuring alignment of business needs with the technical testing approach.
* Training Needs:
  + Accounting Knowledge: Training on accounting principles and financial workflows to support effective validation of business requirements.
  + Compliance & Data Security: Training on compliance standards (e.g., SOX, GDPR) and data security protocols to ensure the responsible management of sensitive financial information.

4. Software Testers (Entry- to Mid-Level Roles)

* Staffing Needs:
  + Basic Testing Execution: Execute functional, regression, and integration tests, documenting results and defects in an organized manner.
  + Defect Tracking & Test Documentation: Ensure all test results, defects, and steps are clearly documented and communicated to facilitate team understanding.
  + Role-Based Access Testing: Verify role-based access controls within the system to confirm users have the appropriate permissions for their roles.
* Training Needs:
  + Test Execution & Reporting Skills: Hands-on training in functional, regression, and integration testing, along with best practices in defect logging and test documentation.
  + Foundational Testing Principles: Ongoing training in software testing fundamentals, such as test case design, exploratory testing, and effective communication of test results.

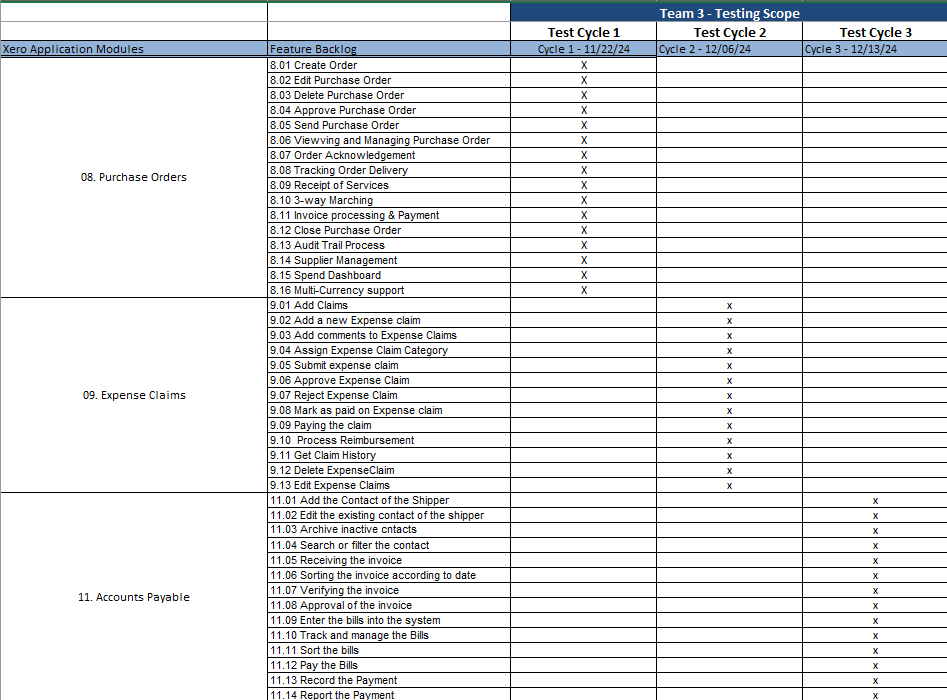
5. Additional Training for the Entire QA Team

* New Tools & Technologies:
  + Automation & Test Management Tools: Training in tools like Selenium (for test automation), Jira (for defect tracking), and TestRail (for test case management) to ensure the team can handle modern testing processes efficiently.
* Xero Application Features:
  + Comprehensive Xero Training: Training on Xero application features, especially financial modules such as Purchase Orders, Expense Claims, and Accounts Payable, to ensure a thorough understanding of the system’s core functionalities.
* Testing Best Practices:
  + Continuous Best Practices Training: Regular sessions on software testing best practices, including defect management, exploratory testing, and effective communication strategies.
  + Exploratory Testing & Test Design: Training in exploratory testing methods and advanced test case design to foster innovation and cover diverse real-world scenarios during testing.
* Performance Testing:
  + Load & Stress Testing: Training on load testing, stress testing, and scalability validation to ensure that the system can manage high transaction volumes effectively.

# Schedule (Darshna)

System testing for Modules are scheduled in three cycles as mentioned below:

* Cycle 1 – Focuses on testing all core features of the Purchase Order Module
* Cycle 2 – Focuses on testing all core features of the Expense Claims Module
* Cycle 3 – Focuses on testing all core features of the Accounts Payable Module



# Risks and contingencies (Rushab)

**Purchase Orders Module**

Risk 1: Limited Testing Resources

* Risk: Limited testing resources may result in delays in testing the purchase orders module. Insufficient staffing or inadequate testing tools can hinder the progress of testing activities, potentially leading to project delays.
* Contingency: To mitigate this risk, a proactive approach to resource management will be adopted. This may include reallocating resources from non-critical tasks, outsourcing specific testing activities related to purchase orders, or leveraging automation tools to improve testing efficiency.

Risk 2: Changes in Scope Objectives

* Risk: Any changes to the scope objectives related to purchase orders can cause delays or additional work in the testing process. Scope changes, whether due to evolving business requirements or stakeholder feedback, may necessitate adjustments to testing plans and timelines.
* Contingency: To address this risk, change management procedures will be implemented specifically for the purchase orders module. Clear communication channels will be established with stakeholders to manage expectations and prioritize testing efforts accordingly.

Risk 3: High Volume of Defects

* Risk: A high volume of defects in the purchase orders module may require longer time to fix and complete testing. The discovery of numerous defects can overwhelm the testing team, leading to extended defect resolution cycles and potential delays in project delivery.
* Contingency: To mitigate this risk, a robust defect management process will be implemented, focusing specifically on the purchase orders module. This includes prioritizing defects based on severity and impact, establishing efficient communication channels between testing and development teams, and implementing proactive measures to prevent the recurrence of similar defects.

**Expense Claims Module**

Risk 1: Limited Testing Resources

* Risk: Limited testing resources may result in delays in testing the expense claims module. Insufficient staffing or inadequate testing tools can hinder the progress of testing activities, potentially leading to project delays.
* Contingency: To mitigate this risk, a proactive approach to resource management will be adopted specifically for testing the expense claims module. This may include reallocating resources from non-critical tasks, outsourcing specific testing activities related to expense claims, or leveraging automation tools to improve testing efficiency.

Risk 2: Changes in Scope Objectives

* Risk: Any changes to the scope objectives related to expense claims can cause delays or additional work in the testing process. Scope changes, whether due to evolving business requirements or stakeholder feedback, may necessitate adjustments to testing plans and timelines.
* Contingency: To address this risk, change management procedures will be implemented specifically for the expense claims module. Clear communication channels will be established with stakeholders to manage expectations and prioritize testing efforts accordingly.

Risk 3: High Volume of Defects

* Risk: A high volume of defects in the expense claims module may require longer time to fix and complete testing. The discovery of numerous defects can overwhelm the testing team, leading to extended defect resolution cycles and potential delays in project delivery.
* Contingency: To mitigate this risk, a robust defect management process will be implemented, focusing specifically on the expense claims module. This includes prioritizing defects based on severity and impact, establishing efficient communication channels between testing and development teams, and implementing proactive measures to prevent the recurrence of similar defects.

**Accounts Payable Module**

Risk 1: Limited Testing Resources

* Risk: Limited testing resources may result in delays in testing the accounts payable module. Insufficient staffing or inadequate testing tools can hinder the progress of testing activities, potentially leading to project delays.
* Contingency: To mitigate this risk, a proactive approach to resource management will be adopted specifically for testing the accounts payable module. This may include reallocating resources from non-critical tasks, outsourcing specific testing activities related to accounts payable, or leveraging automation tools to improve testing efficiency.

Risk 2: Changes in Scope Objectives

* Risk: Any changes to the scope of objectives related to accounts payable can cause delays or additional work in the testing process. Scope changes, whether due to evolving business requirements or stakeholder feedback, may necessitate adjustments to testing plans and timelines.
* Contingency: To address this risk, change management procedures will be implemented specifically for the accounts payable module. Clear communication channels will be established with stakeholders to manage expectations and prioritize testing efforts accordingly.

Risk 3: High Volume of Defects

* Risk: A high volume of defects in the accounts payable module may require longer time to fix and complete testing. The discovery of numerous defects during testing can overwhelm the testing team, leading to extended defect resolution cycles and potential delays in project delivery.
* Contingency: To mitigate this risk, a robust defect management process will be implemented, focusing specifically on the accounts payable module. This includes prioritizing defects based on severity and impact, establishing efficient communication channels between testing and development teams, and implementing proactive measures to prevent the recurrence of similar defects.