ERPNext Software Testing

test PLAN

-Aradhana Nandhikeswaran, 2861686

# Introduction

ERPNext is a comprehensive business management solution that enables small and medium-sized enterprises to manage all their business transactions and operations in one system. Its primary modules include Accounting, Human Resources, and Customer Relationship Management (CRM).

The Accounting module allows users to manage accounts, transactions, and taxes, while the Human Resources module offers features such as attendance tracking, expense claims, employee payroll management, and KPI tracking. The CRM module helps users manage presales processes, including lead capture, opportunity tracking, email and call management, quotation preparation, and pricing.

The software is built on the Frappe Framework, a Python and JavaScript-based full-stack web application development framework. The software is open-source and can be accessed on the public repository at https://github.com/frappe/erpnext. This report outlines the plan that will be used to test the software.

## **Objectives**

The testing plan will include the following objectives:

* Ensure ERPNext functions as intended and meets SMEs' business requirements.
* Uncover defects, bugs, and other issues in the software.
* Ensure compatibility with major platforms, operating systems, and browsers.
* Validate the security of ERPNext, including user authentication, data encryption, and protection against vulnerabilities.
* Test the system's performance, speed, scalability, and response time under different load conditions.
* Evaluate the software's usability, interface, navigation, and user experience to ensure it is user-friendly.
* Test compliance with relevant industry standards and regulations such as accounting regulations and data privacy laws.
* The objectives will guide the testing effort and ensure the software meets business requirements and quality standards.

Any anticipated limitations during the testing process for each objective will be described in the document.

# FEATURES TO BE TESTED

The testing process will cover all features within the core modules of ERPNext software, including:

* Accounting: sales and purchase management, inventory tracking, attendance tracking, expense claims, employee payroll management, and KPI tracking.
* CRM: lead capture, opportunity tracking, email and call management, and quotation preparation for effective selling prices.
* The Human Resource module includes features such as attendance tracking, expense claims, and employee payroll management.

ERPNext has also customized its features for different industries, including manufacturing, healthcare, and agriculture, with functionalities such as stock replenishment, managing sales orders, customers, suppliers, shipments, deliverables, order fulfillment, creating purchase invoices, and journal entries for accounting entries like payments and credits.

# TESTING STRATEGY

1. The testing approach will involve analysing the business requirements for each module and designing test scenarios and cases accordingly.
2. Prioritization of testing will be based on the importance of features and modules, determined via the gathered requirements.
3. Prioritization criteria will include criticality to businesses, customer impact, and complexity.
4. Criticality will evaluate how essential the feature is to SMEs' day-to-day operations.
5. Customer impact will determine which feature or module has a direct impact on the customer experience.
6. Complexity will prioritize features or modules that are more complex and have a higher risk of defects or issues.
7. The testing approach will include system testing to ensure that the software meets the performance, compatibility, compliance, and usability requirements. This will be done using black box testing.
8. Unit testing will be conducted on individual functions within the core modules using pytest and examining code files for bugs and issues.
9. Functional testing will be performed to verify that the system satisfies SMEs' business requirements and specifications.
10. Integration testing will be carried out to ensure that the core modules work well together and with third-party integrations.
11. Automated testing tools such as Selenium and Robot Framework will be utilized to reduce testing time and increase test coverage.
12. Jira will be used to manage test cases and track bugs and issues.

## **IV.1. Unit and Functional Testing**

Unit and functionality testing will be performed on the UI and code level for each core module of ERPNext, including security testing.The testing will ensure that all functions within the core modules are tested, cover all possible input combinations and edge cases, and execute all code paths at least once.

Code coverage analysis will be used to measure the comprehensiveness of the testing effort, with a minimum requirement of 60% code coverage.The requirement traceability matrix (RTM) technique will be used to trace requirements and ensure that all requirements are covered by the tests.

Functionality testing will be conducted using the BlackBox approach, while unit testing will use the WhiteBox approach.

* The testing process will start by reviewing the functional requirements and specifications for the feature or system being tested.
* Test scenarios will be created based on the functional requirements and specifications. Each team member will create scenarios that cover all possible inputs, workflows, and expected results.
* Test cases will be designed to outline the specific steps to execute each scenario.
* The test cases will be executed.
* Results of each test case, including pass/fail status, any defects or issues discovered, and additional notes or comments, will be recorded.

## **IV.2. System and Integration Testing**

System testing will be conducted in a BlackBox approach to validate that the software meets all specified requirements and functions as intended in a real-world environment, from a user's perspective.

• Integration testing will involve combining the three core modules in different ways to test compliance, performance, and compatibility.

• System testing will be performed before integration testing, both based on the BlackBox approach, to ensure compliance, compatibility, and usability requirements are met.

• The testing efforts for both will ensure that all requirements and business processes are tested to meet the necessary performance, compatibility, usability, and compliance requirements.

• Test coverage analysis and traceability matrices will be used to ensure comprehensive testing, with error frequency used as a completion criterion.

The following step are followed to achieve system and integration testing:

1. Review the requirements for the feature or system being tested.
2. Create test scenarios that cover all possible combinations of inputs, workflows, and expected results.
3. Design test cases that outline the specific steps to execute each scenario.
4. Execute the test cases.
5. Record the results of each test case, including pass/fail status, any defects or issues discovered, and additional notes or comments.

Additionally, create their test scripts for their specific type of testing.

## **IV.3. Performance Testing**

Performance testing of ERPNext will include testing its stability, speed, and responsiveness under different workloads and stress conditions. The testing will aim to ensure that all functions and features meet the necessary performance requirements and goals. The comprehensiveness of the testing effort will be judged based on the following criteria:

1. Response time, which is the time taken for the system to respond to user actions, queries, and commands.
2. Throughput, which is the number of transactions or operations the system can handle within a given period.
3. Load capacity, which is the maximum load or number of concurrent users the system can handle without crashing or slowing down.
4. Resource utilization, which is the usage of system resources such as CPU, memory, and disk space during performance testing.
5. Error rate, which is the frequency and severity of errors and exceptions that occur during performance testing.
6. Each criterion will be measured and compared against the desired levels specified in the requirements.

The following steps will be done for performance testing:

* Review the performance requirements specified in the requirements documentation.
* Create test scenarios that cover different workloads and stress conditions.
* Design test cases that outline the specific steps to execute each scenario.
* Execute the test cases.
* Record the results of each test case, including the measured response time, throughput, load capacity, resource utilization, and error rate.

# Requirements and tools.

* The testing effort will require several tools including Jira for test management, Selenium and Robot Framework for test automation, Apache JMeter for load testing, OWASP ZAP for security testing, and SonarQube and Pytest for code analysis.
* Jenkins and GitLab CI will be used as continuous integration and delivery tools.
* The installation of these tools will require specific hardware capabilities including a RAM capacity of at least 4GB, 250GB or more of disk capacity, a processor speed of 2.0GHz or higher, and either Windows or Linux operating systems.

The testing efforts will require the following automation testing tools.

• A test management tool, Jira.

• Test automation tools. Selenium (Selenium 2023) and Robot Framework (Robot Framework 2023)

• Load testing tool, Apache JMeter.

• Security testing tool, OWASP ZAP

• Code analysis tools SonarQube and Pytest

• Continuous integration and delivery tools, Jenkins and GitLab CI (Jenkins 2023).

# TEST SCHEDULE

Each testing process will include the following phases:

* Test Preparation Phase:
  + - Define the test objectives and scope.
    - Determine the necessary testing environment and tools.
    - Create test plans and test cases.
* Test Execution Phase:
* Perform a smoke test to verify that basic functionality is operational.
* Execute the test cases as per the test plan.
* Record and report any defects discovered during testing.

| **Milestones** | **Estimated Completion Timeframe** |
| --- | --- |
| Test Preparation Phase | 04/10/2023 |
| Test Execution Phase | 04/17/2023 |

**VI.1. Test Record Keeping**

* Test record keeping for ERPNext involves documenting and organizing all testing-related information for future reference.
* Test cases and scenarios should be documented with details of the inputs, expected results, and actual results.
* Any defects or issues discovered during testing should be recorded in a defect tracking tool such as Jira, along with details such as severity, steps to reproduce, and other relevant information.
* Test results and progress should be documented and reported regularly to stakeholders, including any issues or risks that may impact the testing effort.
* Documentation should be organized and accessible, with clear and consistent naming conventions for files and folders.
* A version control system such as Git should be used to track changes to test cases and other testing-related documentation.
* Test summary reports should be generated at the end of each testing phase, highlighting the test results, defects found, and overall test coverage.

**Reference :**

* ERPNext manual: Table of contents. Home. (n.d.). Retrieved March 6, 2023, from https://docs.erpnext.com/docs/v13/user/manual/en
* Jenkins. GitLab. (n.d.). Retrieved March 6, 2023, from https://docs.gitlab.com/ee/integration/jenkins.html
* Selenium. (n.d.). Retrieved March 6, 2023, from https://www.selenium.dev/
* Robot framework. Robot Framework. (n.d.). Retrieved March 6, 2023, from https://robotframework.org/
* Docs ERPNext. (n.d.). Retrieved March 6, 2023, from https://docs.erpnext.com/