INFO 6068 Capstone
Test Strategy
Team Pixel

Team Members:
Rutvik Patel (Manger): 1220039
Sakshi Modi (Scribe): 1179590
Parth Patel (Tester): 1229403
Vijul Vyas (Developer): 1227571

Amendment History:

Version	Date	Amendment History
V.1.0	May 22, 2024	Initial Draft
V.1.1	May 26, 2024	Final Test Strategy Document

Reviewers:

Name	Signature	Title / Responsibility	Date	Version
Rutvik Patel	R.P.Patel	Project Manager	May 22, 2024	V.1.0

Approvals:

Name	Signature	Title / Responsibility	Date	Version
Shrijy Atodaria		Project Sponsor	May 22, 2024	V.1.0

Distribution:

Name	Title / Responsibility	Date
Shradha Gupta	Product Owner	May 26, 2024
Shrijy Atodaria	Project Sponsor	May 26, 2024
Rutvik Patel	Project Manager	May 26, 2024
Parth Patel	QA Team	May 26, 2024
Vijul Vyas	Developer Team	May 26, 2024
Sakshi Modi	Scribe	May 26, 2024

Related Documents:

Ref no	Doc Reference Number	Title	Version
R_01	**D_01**	Test Strategy Document	**V.1.0**
R_02	**D_02**	Meeting Agenda	**V.1.0**
R_03	**D_03**	Meeting Minutes	**V.1.0**
R_04	**D_04**	IAD logs	**V.1.0**
R_05	**D_05**	Weekly Status Report	**V.1.0**

Table of Contents

1. Introduction

This Test Strategy Document outlines the comprehensive approach Team Pixel will take to ensure the quality and reliability of our software products. It serves as a roadmap for our testing activities, detailing methodologies, tools, and processes to be employed throughout the development lifecycle. Our goal is to deliver robust, high-performing applications that meet both user expectations and business requirements. This strategy will guide our team in maintaining consistency, improving efficiency, and achieving excellence in all our testing endeavours.

1.1. Objectives

The primary objective of our test strategy is to ensure the application's functionality, reliability, and usability. By implementing a comprehensive testing approach, we aim to identify and resolve any potential issues or bugs, thereby enhancing the overall user experience. Our goal is to deliver a robust and dependable platform that meets the needs and expectations of our users.

- Verify the reliability of the application by conducting rigorous testing to detect and address any software bugs or errors.
- Assess the usability of given system to ensure a seamless and intuitive user experience.
- Collaborate closely with developers and stakeholders to prioritize testing efforts and address any issues identified during the testing process.

1.2. Scope

- The scope of our test strategy encompasses the following key aspects:
  - Development of a comprehensive test plan to monitor and track progress throughout the testing process.
  - Definition of an appropriate test approach tailored to the specific requirements and functionalities.
  - Conducting testing across multiple stages to ensure thorough validation of the system's performance and functionality.
  - Verification of the functionality of all five modules: Administrator, System Functions, Data Management, User Interfaces, and Reporting.
  - Reporting of project progress to the stakeholders at each stage, with testing proceeding only upon review and approval.

2. Roles and Responsibilities

Name	Role	Responsibilities
Rutvik Patel	Project Manager	Project Manager is tasked with coordinating with team members to develop a comprehensive test plan, ensuring adherence to the defined test approach, and monitoring progress across multiple testing stages. Holds the responsibility of overseeing the entire testing process.
Sakshi Modi	Scribe	The Scribe is responsible for documenting all testing activities. They keep track of test plans, results, and any issues encountered during testing. The Scribe ensures that all information is accurately recorded and shared with the team and stakeholders, contributing to the smooth progress of the testing process.
Vijul Vyas	Developer	The Developer is tasked with resolving any bugs or issues identified during testing. They work closely with the testing team to understand and address reported issues promptly. Additionally, the Developer ensures that the application's code is updated and optimized to improve performance and functionality based on feedback from testing.
Parth Patel	Tester	The Tester is responsible for thoroughly examining the application's functionality and usability. They meticulously execute test cases, identify potential issues or bugs, and report their findings to the development team.

3. Testing Overview

3.1. Test Lifecycle

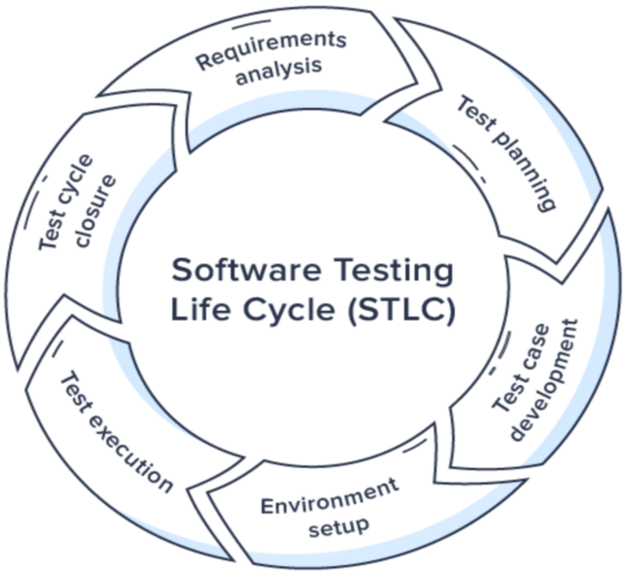

Figure 3.1.1: Software Testing Lifecycle (Bamboo Group, 2022)

- **Requirement Analysis:**
  - In this stage, the testing team thoroughly analyzes the software requirements document to gain a clear understanding of what needs to be tested.
  - Testers identify the functional and non-functional requirements, as well as any ambiguities or inconsistencies in the specifications.
  - This stage is crucial for defining the scope of testing and developing a comprehensive test strategy.

- **Test Planning:**
  - Test planning involves creating a detailed plan outlining the approach, objectives, scope, resources, and schedule for testing.
  - Testers identify the testing techniques, tools, and methodologies to be used based on project requirements.
  - Test planning ensures that testing activities are organized, efficient, and aligned with project goals and objectives.

- **Test Case Development:**
  - In this stage, test cases are developed based on the requirements and test scenarios identified during requirement analysis.
  - Testers create detailed test cases that cover various scenarios, inputs, and expected outcomes for each test scenario.
  - Test cases include preconditions, test steps, expected results, and postconditions to guide testers through the testing process.

- **Environment Setup:**
  - Environment setup involves configuring the testing environment to replicate the production environment as closely as possible.
  - Testers install and configure the necessary hardware, software, and tools required for testing.
  - The testing environment should be stable, secure, and properly configured to ensure accurate and reliable test results.

- **Test Execution:**
  - Test execution is the stage where test cases are executed, and actual testing takes place.
  - Testers execute the prepared test cases in the testing environment and record the results, including any defects or issues encountered.
  - Testing may involve manual testing, automated testing, or a combination of both, depending on the project requirements.

- **Test Cycle Closure:**
  - Test cycle closure marks the end of the testing phase for a particular cycle or iteration.
  - Test closure activities may include generating test reports, documenting lessons learned, and preparing for the next testing cycle.
  - This stage ensures that all testing activities are completed, and the testing artifacts are properly documented and archived.

3.2. Test Approach

Team Pixel has opted for Agile Testing over the Waterfall Method. While Waterfall is simpler, it's not very adaptable. Agile, on the other hand, breaks the work into smaller parts, allowing us to adjust to new requirements easily. This flexibility helps us respond better to changes in the project's needs. For more information on Agile Testing, refer to Figure 3.2.1.

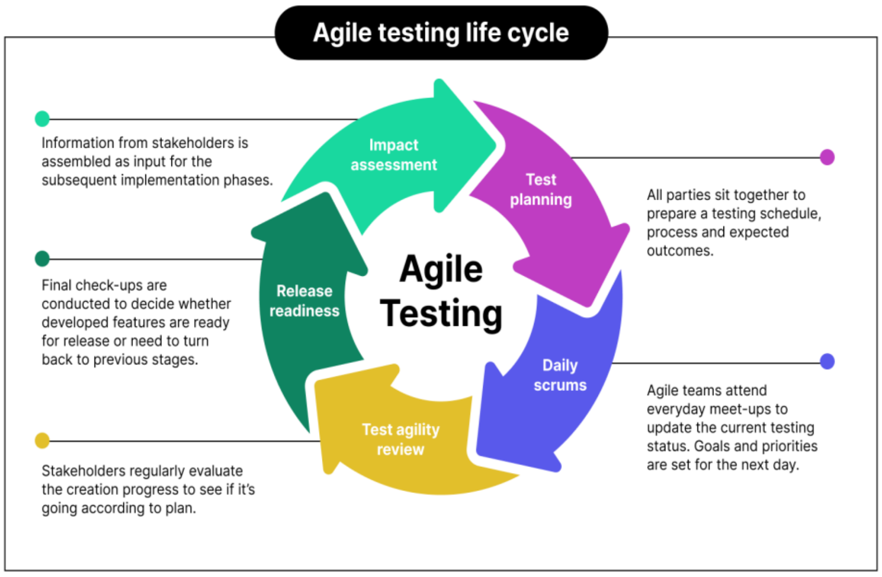
Agile testing life cycle
Agile Testing
Impact assessment
Information from stakeholders is assembled as input for the subsequent implementation phases.
Test planning
All parties sit together to prepare a testing schedule, process and expected outcomes.
Daily scrums
Agile teams attend everyday meet-ups to update the current testing status. Goals and priorities are set for the next day.
Test agility review
Stakeholders regularly evaluate the creation progress to see if it's going according to plan.
Release readiness
Final check-ups are conducted to decide whether developed features are ready for release or need to turn back to previous stages.

Figure 3.2.1: Agile Testing Life Cycle (Katalon, 2023)

1. **Test Planning:**
   - In this phase, the team prepares for the upcoming testing activities. This involves identifying the scope of testing, determining the objectives, and planning the resources and schedules required. Test plans are often created for each iteration or sprint, ensuring they align with the overall project goals.

2. **Daily Scrums:**
   - These are daily stand-up meetings where team members discuss their progress, challenges, and plans for the day. The goal is to ensure continuous communication and quick resolution of any issues. This helps in maintaining transparency and keeping everyone aligned on the project's progress.

3. **Test Agility Review:**
   - After completing a testing cycle, the team conducts a review to evaluate the effectiveness and efficiency of their testing processes. This retrospective analysis helps identify areas of improvement, ensuring that testing practices can be optimized for future cycles.

4. Release Readiness:

- This step involves ensuring that the product is ready for release. It includes finalizing all testing activities, validating that all critical issues have been resolved, and confirming that the product meets the required quality standards. This phase often involves a final round of testing, such as regression or user acceptance testing.

5. Impact Assessment:

- After a release, the team assesses the impact of the changes made in the latest iteration. This involves gathering feedback from users, monitoring for any new issues, and analyzing the overall effect of the release on the product's functionality and performance. This assessment helps inform future planning and continuous improvement efforts.

3.3. Standards

The following severity levels are proposed for issues arising throughout the Test Lifecycle:

- **Critical (S3):**
  - Defects that render the system unusable or violate regulatory requirements, requiring immediate resolution.
- **High (S2):**
  - Issues that significantly impact system functionality, affecting a large portion of users or critical business processes.
- **Medium (S1):**
  - Defects with a moderate impact on functionality or usability, hindering specific features or causing inconvenience.
- **Low (S0):**
  - Minor issues that have minimal impact on functionality or usability, requiring attention but not urgent resolution.

3.4. Test Stages

Each test stage is a discrete form of testing with its own objectives, methods and requirements coverage and therefore a set of its own test scripts.
A coverage matrix of all the Test Stages / Test Areas to be covered in each Test Release is appended below

Test Areas/ Test Type	Component Testing	Component Integration Testing	System Testing	System Integration Testing	Acceptance testing
Functional	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Non-Functional	**-**	**-**	**Yes**	**Yes**	**Yes**
Business Processes	**-**	**-**	**Yes**	**Yes**	**Yes**
Volume	**-**	**-**	**Yes**	**Yes**	**-**
Performance	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Security (including Penetration Testing)	**Yes**	**Yes**	**Yes**	**Yes**	**-**
Data Protection	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Usability	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Interface Tests	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Installation & Configuration	**-**	**-**	**-**	**Yes**	**Yes**
Systems & Service Management & Service Level Reporting	**-**	**-**	**-**	**Yes**	**-**
Network Worthiness	**-**	**-**	**Yes**	**Yes**	**Yes**
Disaster Recovery	**-**	**-**	**-**	**Yes**	**-**
Helpdesk Tools & Processes	**-**	**-**	**-**	**Yes**	**Yes**
Management Information Reporting	**-**	**-**	**-**	**-**	**Yes**
Audit	**-**	**-**	**-**	**-**	**Yes**
Back-up, recovery, journaling	**-**	**-**	**-**	**Yes**	**Yes**

3.5. Reviews and Inspections

3.5.1. Reviews

A review is like a close inspection of a document by individuals or a team to find and fix problems early in the software testing process. Documents that go through reviews include requirements, design, code, test plans, and test cases. The goal is to catch issues early to save time and money on testing. Common problems found during reviews are related to requirements, design, and specifications. The review process has several stages.

- Planning
- Preparation
- Review Meeting
- Rework
- Follow-Up

3.5.2. Inspections and Walkthroughs

Inspections and walkthroughs are formal methods used to review documents and code. These processes involve a thorough examination by a team to identify defects and improve quality.

- **Inspections:**

  Inspections are formal and rigorous reviews conducted by a designated team. They focus on identifying defects in documents, such as requirements specifications, design documents, and test plans. Inspections typically involve multiple reviewers and follow a structured process.

  - **Preparation:** Designate a review team and schedule a formal inspection session.
  - **Review Session:** Team members meticulously examine documents or code, focusing on identifying defects.
  - **Defect Identification:** Use a structured approach to identify and document any issues found during the inspection.
  - **Resolution:** Discuss and prioritize identified defects for resolution.
  - **Follow-Up:** Ensure that all documented issues are addressed and resolved promptly.

- **Walkthroughs:**

  Walkthroughs are less formal reviews where the document or code is presented to a group of stakeholders for feedback. The presenter guides the audience through the document or code, explaining its content and soliciting input. Walkthroughs are more interactive than inspections and are often used to gather feedback early in the development process.

  - **Presentation:** A presenter guides stakeholders through documents or code, explaining content and functionality.
  - **Feedback Gathering:** Encourage active participation and feedback from stakeholders during the walkthrough.

- **Discussion:** Facilitate discussions to address questions, concerns, and suggestions raised during the walkthrough.

- **Documentation:** Document feedback received during the walkthrough for future reference and action.

3.6. Test Documentation

Document	Phase and cycle
Test Strategy	Test Initiation (Planning Phase) (Milestone 1)
Test Plans	Test Initiation (Planning Phase) (Milestone 2)
Test Specifications	Design Phase (Milestone 3)
Test Cases	Case Development Phase (Milestone 4)
Test Scripts	Test Execution (Execution Phase) (Milestone 5)
Test Report	Reporting Phase (Milestone 6)

.

3.7. Test Execution

❖ The primary objective of the Test Execution phase is to validate the quality, functionality, performance, and reliability of the software product. This involves executing test cases, logging defects, and verifying fixes to ensure the product meets the specified requirements and is ready for release.

❖ Test Execution covers all planned test cases based on the requirements and design documents. This includes functional, non-functional, regression, integration, system, user acceptance testing and many more.

❖ **Pre-requisites:**
- Test Environment Setup
- Test Data Preparation
- Test Cases Review
- Access and Permissions

❖ **Tools and Resources:**
- Test Management Tools: (JIRA, TestRail)
- Defect Tracking Tools: (Bugzilla, JIRA)
- Automation Tools: (Selenium, QTP)
- Performance Testing Tools: (JMeter, LoadRunner)
- Team Resources: Rutvik Patel, Saskhi Modi, Parth Patel, Vijul Vyas

3.7.1. Recording Actual Results versus Expected Results

- ❖ During the Test Execution phase, actual results will be meticulously compared to expected results for each test case. Test Reports will be updated regularly to reflect the outcomes, with each test step marked as "Passed" or "Failed" based on whether it meets the expected criteria. This ensures transparency and effective defect tracking.

- ❖ Supporting evidence, such as screenshots and logs, will be collected and attached to provide a comprehensive record of the test activities. All information will be compiled in an Excel spreadsheet for easy analysis and review. This organized approach ensures clear communication and contributes to the overall quality assurance process.

- ❖ The following will be the format of test case spreadsheet:

Test Case ID	Test Description	Test Step	Pre-Condition	Expected Result	Actual Result	Post-Condition	Status (Pass/ Fail)	Comments	Evidence	Severity

3.7.2. Escalation of Issues for resolution

- ❖ During the Test Execution phase, any issues or defects identified will be promptly logged into JIRA, the defect management tool, with detailed descriptions and severity levels. Critical and high-severity issues will be escalated immediately to the project manager and relevant development teams. A structured escalation process will be followed, ensuring that unresolved issues are discussed in daily stand-up meetings or defect triage sessions to prioritize and assign them for resolution.

- ❖ The testing team will collaborate closely with developers to provide additional information and retest the fixes. Regular communication and status updates will be maintained to ensure timely resolution of issues. This approach helps minimize project delays and ensures the quality of the software product, allowing for any critical issues to be addressed swiftly and efficiently.

3.7.3. Test Execution Roles

- ❖ **Project Manager (Rutvik Patel):**

  - **Responsibilities:**
    - Monitor test progress.
    - Facilitate communication between teams.
    - Manage project timelines.
    - Address high-level escalations.

- **Tasks:**
  - Oversee the entire test execution process.
  - Coordinate with stakeholders.
  - Handle high-level escalations.
  - Provide regular status updates to stakeholders.

❖ **Tester (Parth Patel):**

- **Responsibilities:**
  - Execute test cases.
  - Log defects in JIRA.
  - Retest fixed issues.
  - Update test documentation.

- **Tasks:**
  - Execute test cases as per the test plan.
  - Log defects promptly in JIRA with detailed descriptions.
  - Retest fixed issues and update the status accordingly.
  - Maintain and update test documentation as needed.

❖ **Developer (Vijul Vyas):**

- **Responsibilities:**
  - Fix defects reported by the testing team.
  - Provide technical support for issues.
  - Participate in defect triage meetings.

- **Tasks:**
  - Fix defects promptly, prioritizing critical and high-severity issues.
  - Provide technical support to testers for issue resolution.
  - Participate actively in defect triage meetings to discuss and prioritize issues.

❖ **Scribe (Sakshi Modi):**

- **Responsibilities:**
  - Document meeting minutes.
  - Maintain test documentation.
  - Update issue resolution documentation.

- **Tasks:**
  - Document minutes of daily stand-up meetings and defect triage sessions.
  - Maintain and update test documentation, including test plans and test cases.
  - Update issue resolution documentation with details of fixes and retests

3.8. Entry & Exit Criteria

❖ **Entry Criteria:**

- **Test Environment Setup:** Ensure the test environment is configured and validated.
- **Test Data Preparation:** Verify completeness and correctness of test data.
- **Test Cases Review:** Review and approve test cases for accuracy and completeness.
- **Test Execution Plan Approval**: Review and approve the Test Execution Plan.
- **Defect Management Tool Setup**: Configure and set up the defect management tool.

❖ **Exit Criteria:**

- **Test Case Execution**: Execute all planned test cases with documented results.
- **Defect Resolution:** Resolve critical and high-severity defects identified during testing.
- **Test Coverage Achieved:** Achieve defined test coverage metrics.
- **Acceptance Criteria Met:** Meet acceptance criteria for each testing phase.
- **Test Artifacts Updated:** Ensure all test artifacts are updated to reflect the latest testing status.

3.8.1. Table of Entry and Exit Criteria

Criteria	Entry Criteria	Exit Criteria
Test Environment Setup	Test environment configured and validated.	-
Test Data Preparation	Test data completeness and correctness verified.	-
Test Cases Review	Test cases reviewed and approved for accuracy.	-
Test Execution Plan Approval	Test Execution Plan reviewed and approved.	-
Defect Management Tool Setup	Defect management tool configured and set up.	-
Test Case Execution	-	All planned test cases executed with documented results.
Defect Resolution	-	Critical and high-severity defects resolved.
Test Coverage Achieved	-	Defined test coverage metrics achieved.
Acceptance Criteria Met	-	Acceptance criteria for each testing phase met and signed off by stakeholders.
Test Artifacts Updated	-	All test artifacts updated to reflect the latest testing status and results.

3.9. Test Results Capture

- ❖ During each testing stage, thorough documentation of test outcomes is crucial. Test cases are executed according to the test plan, meticulously comparing observed outcomes with expected results. Results are meticulously documented, indicating whether each step passed or failed. If any discrepancy arises between expected and actual results, a test issue is promptly logged in the defect management tool, such as JIRA. These issues are described in detail, including steps to reproduce, severity levels, and supporting evidence like screenshots or logs.

- ❖ The testing team investigates the root cause collaboratively, involving developers and stakeholders as needed. Once resolved, fixes are retested to ensure effectiveness. Verification of resolution is conducted by re-executing associated test cases. Test issues are only closed after resolution has been verified through retesting. This rigorous process ensures any deviations from expected results are promptly identified, investigated, and resolved, ultimately enhancing the overall quality and reliability of the software product.

3.10. Progress Reporting

3.10.1. Test Report

❖ Results will be reported through concise test reports generated at regular intervals. The reports will include:

- Test execution status summary.
- Test coverage achieved.
- Defect metrics.
- Risk assessment with mitigation strategies.
- Recommendations for improvements.
- Appendices with detailed results and artifacts.

❖ These reports will be shared with key stakeholders to ensure transparency and facilitate informed decision-making. Regular meetings will be scheduled to discuss progress and address any concerns.

4. Test Data

Test Type	Source of Test Data
Component Testing	Test data for component testing is typically generated by developers or testers. This data often includes a wide range of inputs to cover various scenarios, including boundary cases, edge cases, and typical usage patterns. It may involve manually creating or generating synthetic data to validate individual components in isolation. This data aims to thoroughly exercise the functionality of each component, ensuring that it performs as expected.
Component Integration	For component integration testing, test data is sourced from mock data, stubs, or simulated data for the integrated components. Mock objects or stubs are used to simulate the behavior of dependent components that are not yet developed or available. This allows for early integration testing without relying on the complete functionality of all components. The test data is designed to represent different integration scenarios and interactions between components, verifying their compatibility and communication.
System Testing	Test data for system testing is derived from realistic data sets that closely resemble the production environment. This data may include actual customer data (anonymized for privacy), sample transactions, or synthetic data generated to simulate real-world usage. The goal is to evaluate the system's behavior and performance under conditions that

	mimic the actual operational environment, ensuring that it meets functional and non-functional requirements.
System Integration Testing	In system integration testing, the test environment is populated with data from interconnected systems to assess the integration points and interactions between various subsystems or external interfaces. This data may consist of inputs and outputs exchanged between integrated systems, including messages, API requests/responses, or database transactions. The focus is on validating the seamless flow of data and functionality across the integrated components, detecting any compatibility issues or data inconsistencies.
Acceptance Testing	Test data for acceptance testing is typically sourced from user-generated data or production-like data provided by stakeholders. This data closely reflects the actual usage scenarios and business processes that end-users will encounter. It may include sample user profiles, typical transactions, or representative datasets relevant to the application's domain. The objective is to validate that the system meets user requirements and expectations, ensuring its usability, functionality, and overall suitability for deployment.

5. Testing Environments

5.1. Specification

5.1.1. Identification of the physical components, the communications, the system and middleware necessary

❖ **Physical Components:**

- Laptop: Asus ROG Strix G15
- Processor: Intel Core i5-10210U
- Memory: 8 GB DDR4 RAM
- Storage: 512 GB SSD
- Display: 14-inch Full HD
- Graphics: RTX 3050
- Connectivity: Wi-Fi 6
- Operating System: Windows 10 Pro 64-bit

❖ **Communications:**

- Networking: Ethernet, Wi-Fi, Bluetooth
- Interfaces: USB, HDMI, SD Card Reader

❖ **System and Middleware:**

- Operating System: Windows 10 Pro 64-bit
- Middleware: Microsoft Office Suite (Word, Excel, PowerPoint).

5.1.2. Other software or supplies needed to support testing

- **Test Automation Tools:** Selenium WebDriver for web application testing
- **Defect Tracking System:** JIRA for logging and tracking defects
- **Test Management Tool:** TestRail for test case management and execution tracking
- **Performance Testing Tool:** Apache JMeter for load and performance testing

5.1.3. Security and access requirements to the test area and equipment

- **User Authentication:** Login credentials with appropriate access levels to the test environment and tools.
- **Network Security:** Firewall and antivirus software installed and updated regularly.
- **Data Encryption:** Encrypted storage for sensitive test data.

5.1.4. Test tools and utilities required

- **Development Tools:** Integrated Development Environment (IDE) such as Visual Studio Code for scripting and coding test automation.
- **Browser:** Latest versions of Google Chrome, Mozilla Firefox, and Microsoft Edge for web application testing.
- **Performance Monitoring Tools:** Task Manager for monitoring system resources during performance testing.

.

6. Testing Tools

6.1. Test Management Tools

- TestRail
- Zephyr
- qTest

6.2. Test Automation Tools

- Selenium WebDriver
- Cypress
- Katalon Studio
- Postman
- SoapUI
- Jenkins

7. References

- ❖ SketchBubble. (n.d.). Presentation Entry and Exit Criteria. Retrieved from https://www.sketchbubble.com/en/presentation-entry-and-exit-criteria.html
- ❖ Wrike. (n.d.). What Is Agile Methodology in Project Management? Retrieved from https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/
- ❖ Bamboo Agile. (n.d.). What is STLC? Retrieved from https://bambooagile.eu/insights/what-is-stlc/
- ❖ Katalon. (n.d.). Agile Testing Methodology. Retrieved from https://katalon.com/resources-center/blog/agile-testing-methodology
- ❖ Akdogan, H. (n.d.). Software Testing Life Cycle (STLC). Retrieved from https://www.linkedin.com/pulse/software-testing-life-cycle-stlc-hasan-akdogan-3mkgf