TEST STRATEGY FOR STARSCHINERS.RO

DOCUMENT NAME: PROIECT STARCHINERS TESTARE MANUALA (PSTM)

DOCUMENT CONTROL

VERSION: 1.0
DATE: 23.05.2023
CREATED BY: SABIN MARCU

DOCUMENT SIGN-OFF

VERSION	STATUS	DATE	APPROVED BY	JOB TITTLE
1.0	FINAL	23.05.2023	JOHN SMITH	PRODUCT OWNER

CONTENTS

1 INTRODUCTION

The project objective is to thoroughly test the functionality and usability of the Starschiners.ro website, ensuring that all the identified features, such as login, logout, password reset, social sign-in, search, add to cart, and checkout, are working as intended, meeting quality standards, and providing a seamless user experience. Additionally, the project aims to identify and report any issues, bugs, or security vulnerabilities for timely resolution and improvement of the website.

The project objectives for testing the various functionalities on the starschiners.ro website can be defined as follows:

1. Login Testing Objective:
    - Verify that users can successfully log in with valid credentials.
    - Validate that users are unable to log in with invalid credentials.
    - Ensure the system displays appropriate error messages for invalid login attempts.
    - Check if the login functionality is secure and protected against common security vulnerabilities, such as SQL injection or cross-site scripting.

2. Sign up process Testing Objective:

    - Test the registration process to ensure users can create new accounts.
    - Verify that all required fields are properly validated during sign up.
    - Check for any error messages or issues during the registration process.
    -

2. Reset Password Testing Objective:
    - Test the password reset functionality to ensure users can successfully reset their passwords.
    - Verify that users receive the password reset email or notification.
    - Validate that users can set a new password and log in using the updated credentials.
    - Check the security measures in place to protect against unauthorized password resets.
3. Sign in with Facebook Testing Objective:
    - Verify that users can sign in using their Facebook accounts.
    - Test the integration with Facebook's authentication system to ensure a smooth sign-in process.
    - Validate that the necessary user information is fetched correctly from Facebook and stored securely.

- Confirm that users can link their Facebook accounts to the website's user accounts.

4. Search Testing Objective:
    - Verify that the search functionality returns relevant results based on user queries.
    - Test different search scenarios, such as searching by product name, category, or specific attributes.
    - Validate that the search results are displayed in an organized and user-friendly manner.
    - Check if the search feature handles errors gracefully and provides informative messages when no results are found.
5. Add to Cart Testing Objective:
    - Test the functionality of adding items to the shopping cart.
    - Verify that users can add products to their cart successfully.
    - Validate that the correct quantity and details of the selected items are displayed in the cart.
    - Check if the cart updates dynamically when items are added or removed.
6. Checkout Testing Objective:
    - Test the checkout process to ensure a smooth and secure transaction.
    - Verify that users can proceed to the checkout page from the shopping cart.
    - Validate that users can provide their shipping and payment information accurately.
    - Confirm that the system calculates the total cost correctly, including taxes and shipping fees.
    - Test different payment methods (credit card, PayPal, etc.) to ensure they are processed correctly.

2 Purpose

In the context of ensuring that all the identified features are adequately tested, the test strategy for Starschiners.ro website would outline the following objectives:

Feature Coverage: The test strategy aims to ensure that all the identified features, such as login, sign up, password reset, social sign-in, search, add to cart, and checkout, are included in the testing scope. It ensures that each feature is thoroughly tested to validate its functionality, usability, and reliability.

Functional Testing: The test strategy includes objectives related to functional testing, which involves validating that each feature behaves as expected, performs the intended functions, and meets the specified requirements. It ensures that users can successfully perform actions like logging in, signing up, resetting passwords, searching, adding items to the cart, and completing the checkout process.

Security Testing: The test strategy may outline objectives for security testing to ensure that the identified features, especially login, sign up, and password reset, are secure and protect user data. It involves testing for vulnerabilities, such as authentication weaknesses, data breaches, and protection against common security threats.

SYSTEM OVERVIEW

4.Scope of testing

The scope of testing for Starschiners.ro website is to check the folowing:

- User Registration and Authentication: Testing the registration process, login functionality, and password recovery features to ensure users can create accounts and access their profiles securely.
- Product Browsing and Search: Verifying that users can browse through different categories, filter and sort products, and search for specific items using keywords.
- Shopping Cart and Checkout: Testing the functionality of adding products to the shopping cart, updating quantities, applying discounts or promotional codes, and successfully completing the checkout process.
- Payment Gateway Integration: Verifying the integration with payment gateways to ensure that users can securely make payments using various payment methods (credit cards, PayPal, etc.) without any issues.
- Order Management: Testing the order placement process, order tracking functionality, and ensuring that users receive order confirmation emails with the correct details.
- Account Management: Verifying that users can update their account information, manage their addresses, and view order history.

4.1 In Scope

- **Functional Testing**: This type of testing ensures that the individual functionalities of login, sign up, password reset, social sign-in, search, add to cart, and checkout are working as intended. It involves verifying that each functionality performs the expected operations and produces the correct results.
- **Security Testing**: Security testing is essential for login, sign up, password reset, and social sign-in functionalities to ensure that user credentials and personal information are handled securely. It involves checking for

vulnerabilities, verifying the strength of password encryption, and preventing common security risks.

- **Compatibility Testing**: Compatibility testing ensures that the mentioned functionalities work correctly across different browsers, operating systems, and devices. It involves testing on various platforms and configurations to ensure consistent behavior and functionality.

4.2 Out of scope

- **Usability Testing**: Usability testing focuses on evaluating the user-friendliness and ease of use of the mentioned functionalities. It involves assessing factors such as user interface design, navigation, and overall user experience. Usability testing helps identify any usability issues or improvements that can enhance the user's interaction with the system.
- **Security Testing**: Security testing is essential for login, sign up, password reset, and social sign-in functionalities to ensure that user credentials and personal information are handled securely. It involves checking for vulnerabilities, verifying the strength of password encryption, and preventing common security risks such as SQL injections, cross-site scripting (XSS), and session hijacking.
- **Performance Testing**: Performance testing focuses on evaluating the responsiveness and efficiency of the system during heavy usage scenarios. For functionalities like search, add to cart, and checkout, it involves measuring response times, server load handling, and identifying any performance bottlenecks that may affect the user experience.
- **Compatibility Testing**: Compatibility testing ensures that the mentioned functionalities work correctly across different browsers, operating systems, and devices. It involves testing on various platforms and configurations to ensure consistent behavior and functionality.
- **Regression Testing**: Regression testing is performed to verify that any changes or updates made to the system have not introduced new defects or broken existing functionalities. It involves retesting the login, sign up,

password reset, social sign-in, search, add to cart, and checkout functionalities to ensure their continued proper functioning.

- **Localization and Internationalization Testing**: If the system is intended to be used in multiple languages or regions, localization and internationalization testing become important. It ensures that the mentioned functionalities can handle different languages, cultural conventions, date/time formats, and currency symbols without any issues.

5.Principles and aplications

7. Exhaustive Testing is Impossible: It is practically impossible to test every possible scenario and combination in an e-commerce website like Starschiners.ro. However, you can prioritize the critical functionalities such as login, sign up, password reset, social sign-in, search, add to cart, and checkout, user flows, to achieve maximum coverage.
8. Early Testing: Begin testing activities as early as possible in the software development life cycle. This ensures that issues are identified and addressed at an early stage, reducing the cost and effort of fixing them later. Early testing the Starschiners website can involve verifying basic functionalities, such as ogin, sign up, password reset, social sign-in, search, add to cart, and checkout.
9. Defect Clustering: Defect clustering refers to the observation that a small number of modules or functionalities tend to have a significant number of defects. It is important to identify and focus on the critical areas that are prone to defects, such as the checkout process, payment gateway integration,
10. Pesticide Paradox: The pesticide paradox principle states that if the same set of tests is repeated over and over, eventually, they become ineffective in finding new defects. In the context of an e-commerce websitelike Starschiners.ro, it is crucial to regularly update and enhance the test suite to incorporate new features, changes, and known issues.
11. Testing is Context Dependent: Testing strategies and techniques should be tailored to the specific context of the Starschiners.ro website. Consider factors such as target audience, supported devices and browsers, integration with

third-party services, and security requirements. This ensures that testing efforts align with the unique characteristics of the e-commerce platform.

12. Absence-of-Errors Fallacy: The absence of errors during testing does not guarantee the absence of defects. On Starschiners.ro website, ensure thorough testing of critical functionalities, user interactions, and edge cases, even if they appear to be error-free. Employ techniques such as boundary value analysis, equivalence partitioning, and stress testing to uncover potential defects.
13. Testing is a Risk-Based Endeavor: Testing efforts should be focused on areas of the e-commerce website that present higher risks. Identify the potential risks associated with the website, such as security vulnerabilities, performance bottlenecks, data integrity issues, and prioritize testing accordingly. This approach helps allocate testing resources effectively and ensures critical risks are mitigated.

6. PROJECT TEAM

The TEAM methodology defines six dimensions that are essential for successful software development, with specific focus on testing. These dimensions encompass various aspects that contribute to effective testing practices. Here's a breakdown of each dimension and its relevance to the testing discipline within the methodology:

- Process: The process dimension refers to the structured approach followed for software development and testing activities. Within testing, this dimension encompasses the test planning, test design, test execution, and test closure processes. It involves defining testing objectives, identifying test requirements, designing test cases, executing tests, and analyzing test results. An effective testing process ensures systematic and thorough testing of software.

- Team: The team dimension recognizes the importance of collaboration and teamwork in software development, including testing. Within the testing discipline, this dimension emphasizes the composition and roles of the testing team. It involves defining the responsibilities and skill sets of testers, test leads, and other stakeholders involved in testing. Effective teamwork and coordination among team members help in efficient test execution, bug reporting, and defect resolution.
- Tools: The tools dimension focuses on the selection and utilization of appropriate testing tools and technologies. In the testing discipline, this dimension includes the identification and adoption of tools for test management, test automation, defect tracking, and performance testing, among others. Using the right testing tools enhances efficiency, accuracy, and repeatability in testing activities.
- Communication: The communication dimension emphasizes the importance of effective communication within the testing team and with other stakeholders. Clear and timely communication helps in understanding requirements, discussing test plans, reporting defects, and sharing test results. It also facilitates collaboration with developers, business analysts, and project managers, ensuring a shared understanding of testing goals and progress.
- Culture: The culture dimension refers to the organizational culture and values that influence testing practices. It includes the mindset towards quality, the emphasis on continuous improvement, and the establishment of a supportive and learning-oriented environment. Within testing, this dimension promotes a culture of quality assurance, where testing is seen as a critical and integral part of software development.
- People: The people dimension recognizes that skilled individuals are the driving force behind successful testing. It involves recruiting and retaining competent testers, providing training and professional development opportunities, and fostering a culture of learning and knowledge sharing. Investing in people ensures that the testing team has the necessary expertise and skills to perform their roles effectively.

There are two distinct software development approaches described by TEAM:

- Agile
- Planned Iterative

This project will be delivered using the Planned Iterative methodology:

Planned iterative methodologies, also known as iterative development or incremental development, are software development approaches that involve breaking down a project into smaller iterations or increments. Each iteration focuses on delivering a specific set of features or functionalities, and feedback from users and stakeholders is used to refine and enhance the subsequent iterations. Here are the key aspects of a planned iterative methodology:

- Planning: The project is divided into multiple iterations, with each iteration having its own set of goals, deliverables, and timeline. The overall project plan is created based on the requirements and priorities, and the iterations are planned accordingly.
- Requirements Gathering: The initial requirements are gathered at the beginning of the project, but detailed requirements for each iteration are gathered and refined before the start of that iteration. This allows for flexibility and adaptability as the project progresses.
- Iterative Development: Each iteration follows a similar development cycle, including requirements analysis, design, implementation, testing, and deployment. However, the scope of each iteration is limited to specific features or functionalities, making it more manageable and focused.
- Incremental Delivery: At the end of each iteration, a working increment of the software is delivered. This allows stakeholders to review and provide feedback on the delivered functionality. The feedback is then used to refine and improve subsequent iterations.
- Feedback and Iteration Review: The feedback received from stakeholders is carefully considered, and any necessary changes or enhancements are incorporated into the project plan and subsequent iterations. Regular meetings and reviews are conducted to evaluate the progress and adjust the plan as needed.
- Risk Management: Risk management is an integral part of planned iterative methodologies. Risks are identified and assessed throughout the project, and

mitigation strategies are implemented. Regular feedback and iteration reviews help identify and address potential risks early on.

- Continuous Improvement: The iterative nature of the methodology allows for continuous improvement. Lessons learned from each iteration are carried forward to subsequent iterations, ensuring that the development process becomes more efficient and effective over time.

Planned iterative methodologies provide several advantages, such as flexibility, adaptability to changing requirements, early delivery of working software, and improved stakeholder engagement. However, they also require effective communication, collaboration, and project management to ensure successful implementation.

5.2 (Test Phase 1)- First Sprint

5.2.1 Objectives

The objective of testing sign-up, log-in, sign-in with Facebook, and reset password functionalities in a software application or website is to ensure their functionality, security, and usability. Here are the key objectives for testing each of these functionalities:

Sign-Up Testing:

- Verify that users can successfully create an account by providing the required information.
- Validate the validation rules for different input fields, such as username, email, password, etc.
- Ensure that appropriate error messages are displayed when invalid or incomplete information is provided.
- Test the integration with any third-party services used for email verification or account activation.
- Verify that the newly created account is properly stored and associated with the user.

Log-In Testing:

- Validate that users can log in to their existing accounts using their registered credentials.

- Test for different scenarios, such as valid username/password, incorrect credentials, and account lockouts after multiple failed attempts.
- Verify that appropriate error messages are displayed for different login failure scenarios.
- Ensure that the logged-in user's session remains active and their authentication persists across different pages or sessions.

Sign-In with Facebook Testing:

- Test the integration with Facebook's authentication APIs to ensure a smooth and secure sign-in process.
- Verify that users can log in using their Facebook credentials and grant the necessary permissions.
- Test the handling of various scenarios, such as successful sign-in, invalid or expired access tokens, and user denial of permissions.
- Validate that the user's Facebook profile information is properly retrieved and associated with the application's user account.
- Ensure that appropriate error messages are displayed in case of any integration or authentication failures.

Reset Password Testing:

- Test the "Forgot Password" functionality to ensure that users can request a password reset.
- Verify that users receive the password reset email with the necessary instructions and a secure reset link.
- Test the reset link to ensure that it properly verifies the user's identity and allows them to set a new password.
- Validate the password reset process, including the strength requirements, confirmation prompts, and successful password update.
- Ensure that appropriate error messages are displayed in case of any failures or invalid reset requests.

Completing each stage of testing for sign up, log in, sign in with Facebook, and reset password is important to ensure the quality, functionality, and security of these critical features in a software application. Each stage of testing is necessary for these specific functionalities:

Sign Up Testing: Sign up testing focuses on validating the registration process for new users. It involves testing the user interface, data validation, and error handling during the sign-up flow. This stage ensures that users can successfully create an account, input and save their information correctly, and receive appropriate feedback or error messages if any issues arise.

Log In Testing: Log in testing verifies the functionality and security of the login process. It includes testing various scenarios, such as entering correct and incorrect credentials, handling different user roles and permissions, and testing the session management or authentication mechanisms. This stage ensures that users can securely access their accounts and that unauthorized access is prevented.

Sign In with Facebook Testing: Sign in with Facebook functionality allows users to log in to an application using their Facebook credentials. This stage of testing focuses on the integration with the Facebook API or SDK, ensuring that the authentication process works smoothly, user data is properly retrieved, and any required permissions are handled correctly. It also involves testing the fallback mechanisms in case the Facebook integration fails.

R**eset Password Testing**: Reset password testing ensures that users can securely reset their passwords in case they forget them or need to change them. It involves testing the password recovery flow, verifying that the user receives the password reset email or notification, and ensuring that the password can be reset successfully. Additionally, this stage includes testing the security measures in place, such as verifying that the reset link has an expiration time and that proper validation is performed during the password update process.

Completing each stage of testing for these functionalities is crucial because they directly impact the user experience, security, and overall functionality of the application. These features involve sensitive user data, authentication mechanisms, and interactions with external services (such as Facebook), making them potential points of failure or vulnerability if not thoroughly tested. By rigorously testing each stage, you can uncover and address any issues, bugs, or usability concerns to provide a reliable and user-friendly experience for your application's users.

5.2.2 Scope

The scope of testing for the functional and nonfunctional aspects of the sign-up, log-in, sign-in with Facebook, and reset password functionalities can be defined as follows:

Functional Testing Scope:

Sign-Up Functionality:

- Verify that users can successfully create an account by providing required information such as name, email, password, etc.
- Validate the registration process by ensuring that all mandatory fields are properly validated and error messages are displayed for invalid inputs.
- Test the functionality to check for email uniqueness to prevent duplicate accounts.
- Confirm that a confirmation email is sent or an account activation process is in place.

Log-In Functionality:

- Verify that registered users can log in using their credentials (email/username and password).
- Test the authentication process to ensure that only valid users with correct credentials can log in.
- Validate the handling of incorrect login attempts, including displaying appropriate error messages or implementing account lockouts for multiple failed attempts.
- Test the "Remember Me" functionality to confirm that user sessions persist across browser sessions.

Sign-In with Facebook Functionality:

- Test the integration with Facebook's authentication API to ensure users can log in using their Facebook credentials.
- Verify that user information obtained from Facebook (name, email, profile picture, etc.) is correctly captured and stored in the user profile.
- Validate the handling of scenarios where users revoke Facebook access or encounter authentication errors.

Reset Password Functionality:

- Test the process for resetting passwords, including verifying the availability of a "Forgot Password" option.
- Validate the functionality for sending password reset emails to the registered email address.
- Confirm that users can successfully reset their passwords by following the instructions provided in the reset email.
- Test the security measures in place, such as preventing unauthorized access to password reset functionality.

Nonfunctional Testing Scope:

Security:

- Validate the security measures in place for protecting user information during the sign-up, log-in, sign-in with Facebook, and reset password processes.
- Test for vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
- Validate the implementation of secure password storage mechanisms (e.g., hashing and salting).

Error Handling and Logging:

- Validate that appropriate error messages are displayed for invalid inputs, missing fields, or other error conditions during the sign-up, log-in, sign-in with Facebook, and reset password processes.

- Test the logging of relevant events and errors for troubleshooting and auditing purposes.

5.2.3 Test Preparation

Preparing for the testing phase of the sign-up, log-in, sign-in with Facebook, and reset password functionalities involves a systematic approach to ensure comprehensive and effective testing. Here's a step-by-step guide to preparing for testing:

- Requirement Analysis: Thoroughly review the functional and non-functional requirements related to the sign-up, log-in, sign-in with Facebook, and reset password functionalities. Understand the expected behavior, user flows, input validations, security requirements, and integration specifications with external systems (such as Facebook API for sign-in with Facebook).
- Test Planning: Develop a detailed test plan specifically targeting these functionalities. Identify the scope, objectives, and success criteria of the testing phase. Define the test scenarios and test cases to cover different scenarios, user actions, and potential edge cases. Also, consider negative testing scenarios, such as invalid inputs or error conditions.
- Test Environment Setup: Set up the required test environments for testing these functionalities. This includes creating test user accounts, configuring the necessary authentication mechanisms, and ensuring the availability of test data for different scenarios (e.g., valid and invalid passwords, expired or locked accounts for reset password testing).
- Test Data Preparation: Prepare relevant test data to cover various scenarios. For sign-up and log-in, include both valid and invalid user credentials, email addresses, and passwords. For sign-in with Facebook, you may need to create test Facebook accounts or obtain access to pre-existing test accounts. For reset password, ensure a mix of valid and invalid user accounts, including accounts with and without associated email addresses.
- Test Case Design: Create test cases that cover all possible combinations of inputs, user actions, and expected outcomes. Ensure that test cases address positive scenarios (valid data and expected success), negative scenarios (invalid data and expected error messages), and boundary cases (minimum and maximum input values).

- Test Execution: Execute the test cases systematically, following the defined test plan. Test the sign-up process by registering new users, validating the input fields, and checking if the user accounts are created correctly. Test the log-in process by verifying the authentication, session management, and access control mechanisms. For sign-in with Facebook, test the integration by verifying successful authentication and data synchronization. Test the reset password functionality by initiating the reset process, following the steps, and validating if the password is successfully reset.
- Defect Reporting: Document any observed defects, issues, or discrepancies during the testing phase. Clearly describe the problem, steps to reproduce, and expected versus actual behavior. Log the defects in a defect tracking system and assign them to the respective development team or stakeholders for resolution.
- Retesting and Regression Testing: After the reported defects are fixed, conduct retesting to verify the resolution and ensure that no new issues were introduced. Additionally, perform regression testing to ensure that the fixes or changes to one functionality did not impact the other related functionalities.
- Test Completion and Reporting: Once testing is completed and all critical defects are addressed, summarize the testing activities, including the test coverage, test execution results, and any metrics or observations. Provide a comprehensive test report that highlights the status, overall quality, and any recommendations for improvement.

5.2.3.1 Entry Criteria/DOR

Entry criteria, also known as Definition of Ready (DoR), specify the necessary conditions or prerequisites that must be met before starting the testing or development of a particular functionality. Here are the entry criteria or DoR for the sign up, log in, sign in with Facebook, and reset password functionalities for Starschiners.ro website:

Sign Up Functionality:

- UI/UX design for the sign-up form is finalized.
- User registration requirements and validation rules are defined.
- Database and backend infrastructure for storing and managing user account information are set up.
- Error handling and validation messages for the sign-up process are defined.
- User acceptance criteria for the sign-up functionality are clearly defined.
- Test data for different scenarios, such as valid and invalid inputs, is prepared.

Log In Functionality:

- UI/UX design for the login form is finalized.
- User account creation functionality is in place.
- User credentials (username or email and password) are stored securely in the database.
- Account verification process (if any) is implemented and tested separately.
- Error handling and validation messages for the login process are defined.
- User acceptance criteria for the login functionality are clearly defined.
- Test data for different scenarios, such as valid and invalid credentials, is prepared.

Sign In with Facebook Functionality:

- Facebook authentication API and integration are implemented and tested.
- UI/UX design for the sign-in with Facebook feature is finalized.
- App registration and configuration with the Facebook developer platform are completed.
- Required permissions for accessing user information through Facebook are defined and requested.
- User acceptance criteria for the sign-in with Facebook functionality are clearly defined.
- Test data for different scenarios, such as successful and failed authentication, is prepared.

Reset Password Functionality:

- UI/UX design for the password reset form is finalized.
- User account information and password reset options are securely stored in the database.
- Password reset email functionality is implemented and tested.
- User acceptance criteria for the password reset functionality are clearly defined.
- Test data for different scenarios, such as valid and invalid email addresses, is prepared.

5.2.3.2 Exit Criteria/DOD

Exit criteria, also known as Definition of Done (DoD), are specific conditions or requirements that must be met before a particular task or user story can be considered complete. Here are some examples of exit criteria/DoD for the functionalities of sign up, log in, sign in with Facebook, and reset password on Starschiners.ro website:

Sign Up Functionality:

- User should be able to successfully create an account.
- User information (such as name, email, password) should be validated and stored securely.
- User should receive a confirmation email (if applicable) for account verification.
- Error handling should be in place for invalid or duplicate entries.
- User should be redirected to the appropriate page after successful sign up.

Log In Functionality:

- User should be able to enter valid credentials and log in successfully.
- User authentication should be performed securely (e.g., using encryption, hashing).
- Invalid login attempts should be handled properly, such as displaying error messages and enforcing account lockouts.
- User should be redirected to the appropriate page after successful login.

- User session and authentication state should be maintained throughout the user's session.

Sign In with Facebook Functionality:

- User should be able to sign in using their Facebook account credentials.
- User authentication should be securely handled through Facebook's APIs.
- User data retrieved from Facebook (such as name, email) should be integrated and stored securely.
- User should be redirected to the appropriate page after successful sign in.

Reset Password Functionality:

- User should be able to initiate a password reset process.
- User should receive a password reset email with a secure and unique reset link.
- User should be able to follow the reset link and securely set a new password.
- Password reset should invalidate any previously stored passwords and enforce password complexity rules.
- User should be redirected to the appropriate page after successful password reset.

5.3 (Test Phase 2)- Second sprint

5.3.1 Objective

The objective for search, add to cart, and checkout functionalities on Starschiners.ro system is to provide a seamless and user-friendly shopping experience for customers. These functionalities aim to enable users to find desired products, add them to their shopping carts, and complete the purchase process efficiently. Here's a breakdown of the objectives for each functionality:

Search Functionality:

- Objective: Enable users to easily search and find products based on their specific requirements.
- Key Goals:
    - Provide a fast and accurate search feature that returns relevant results.
    - Support various search options, such as keyword-based search, filters, sorting, and category-based browsing.
    - Display search results with essential product information, including images, prices, and ratings.
    - Offer advanced search features, such as autocomplete suggestions and predictive search, to enhance usability.

Add to Cart Functionality:

- Objective: Allow users to collect desired items and prepare for the checkout process.
- Key Goals:
    - Enable users to add products to their shopping carts with a simple and intuitive interface.
    - Provide clear feedback when an item is successfully added to the cart.
    - Support quantity selection, product variations (e.g., size, color), and options for customizing products (if applicable).
    - Display a summary of the cart contents, including the total cost, and allow users to modify the cart (e.g., update quantities, remove items).

Checkout Functionality:

- Objective: Facilitate a smooth and secure process for users to complete their purchases.
- Key Goals:
    - Guide users through the checkout process with clear steps and instructions.
    - Collect necessary customer information, such as shipping address, payment details, and contact information.
    - Provide multiple payment options and ensure a secure payment gateway.
    - Display an order summary with itemized details, pricing, and any applicable discounts or promotions.
    - Offer order review and confirmation before finalizing the purchase.
    - Provide order tracking information and confirmation to the user after successful checkout.

Overall, the objectives for search, add to cart, and checkout functionalities are centered around enhancing the user experience, simplifying the buying process, and ensuring customer satisfaction throughout their journey on Starschiners.ro platform.

5.3.2 Scope

The scope of testing for search, add to cart, and checkout functionalities are:

Search Functionality: Functional Requirements:

Allow users to enter search queries.

Retrieve relevant products based on the search query.

Display search results in a clear and organized manner.

Support filtering and sorting options for search results.

Handle misspelled or alternative search terms.

Provide autocomplete suggestions while typing the search query.

Allow users to refine their search criteria.

Nonfunctional Requirements:

Performance: Provide fast search results, ideally within seconds.

Scalability: Support a large number of concurrent search requests.

Accuracy: Return accurate and relevant search results.

Availability: Ensure the search functionality is always available to users.

Usability: Design an intuitive and user-friendly search interface.

Security: Protect user search queries and results from unauthorized access.

Add to Cart Functionality: Functional Requirements:

Allow users to add products to their shopping cart.

Display the contents of the cart, including product details and quantities.

Provide the option to adjust the quantity or remove items from the cart.

Calculate the total price of the items in the cart.

Handle out-of-stock items or quantity limitations.

Allow users to continue shopping after adding items to the cart.

Nonfunctional Requirements:

Performance: Provide a responsive and seamless add to cart process.

Scalability: Support a large number of concurrent users adding items to their carts.

Reliability: Ensure that items added to the cart are not lost or cleared accidentally.

Usability: Design a clear and intuitive cart interface.

Persistence: Retain the cart contents across user sessions.

Checkout Functionality: Functional Requirements:

Allow users to initiate the checkout process.

Collect shipping and billing information from the user.

Provide multiple payment options (e.g., credit card, PayPal).

Calculate and display the final order total, including taxes and shipping fees.

Validate and verify the user's information and payment details.

Generate an order confirmation or receipt after successful checkout.

Nonfunctional Requirements:

Security: Ensure the confidentiality and integrity of user's payment and personal information.

Performance: Process the checkout quickly to reduce user wait times.

Reliability: Prevent errors or data loss during the checkout process.

Usability: Design a clear and intuitive checkout flow.

Compatibility: Support different devices and browsers for checkout.

Compliance: Follow relevant regulations and standards for handling payment information.

To prepare for testing search, add to cart, and checkout functionalities on Starschiners.ro, youwill follow these steps:

Understand the requirements: Read and understand the requirements or user stories related to the search, add to cart, and checkout functionalities. Make sure you have a clear understanding of what is expected from these features.

Identify test scenarios: Identify the different scenarios you need to test for each functionality. For example, for the search functionality, you may need to test searching with different keywords, searching with filters, searching for specific items, etc. Similarly, for the add to cart and checkout functionalities, you may need to test adding single and multiple items, updating quantities, applying discounts or coupons, checking shipping options, payment processing, etc.

Create test data: Prepare the necessary test data for your scenarios. This can include creating sample products, user accounts, coupons, and any other data required to execute your test cases.

Test environment setup: Set up a test environment that closely resembles the production environment. This includes having the required hardware, software, databases, and configurations in place.

Write test cases: Document your test cases based on the identified scenarios. Each test case should have a clear objective, steps to execute, and expected results. Include both positive and negative test cases to ensure thorough coverage.

Execute test cases: Execute your test cases in the test environment. Follow the steps outlined in each test case, perform the required actions, and compare the actual results with the expected results.

Report issues: If you encounter any issues or discrepancies during the testing process, report them in a structured manner. Provide detailed steps to reproduce the issue, along with any relevant logs or screenshots.

Cross-browser and device testing: Test the functionalities across different browsers (Chrome, Firefox, Safari, etc.) and devices (desktop, mobile, tablets) to ensure compatibility and consistent behavior.

5.3.3.1 Entry Criteria/DOR

Entry criteria, also known as Definition of Ready (DoR), refers to a set of conditions or prerequisites that must be met before work on a specific task or user story can begin. In the context of search, add to cart, and checkout functionalities, the entry criteria you will consider:

Search Functionality:

- The user interface (UI) design for the search feature is finalized.
- The search functionality is integrated with the backend or data source.
- Any necessary search filters or options are defined and implemented.
- The search results display correctly and are aligned with the specified requirements.
- Any relevant error handling and validation mechanisms are in place.

Add to Cart Functionality:

- The UI design for the add to cart feature is finalized.
- The product catalog or inventory system is integrated with the shopping cart.
- The selected products are added to the cart with the correct quantity and options.
- The cart total and individual item prices are calculated accurately.
- Any relevant discounts, promotions, or coupons are applied correctly.
- The cart items are displayed correctly in the UI, including product details and quantities.
- The cart supports operations like updating quantities, removing items, or saving for later.

Checkout Functionality:

- The UI design for the checkout process is finalized.
- The customer's shipping and billing information can be entered and saved.
- The shipping methods and costs are calculated accurately based on the user's location.
- The available payment options, such as credit card, PayPal, or other gateways, are integrated.
- The selected payment method is validated, and any necessary payment processing is handled.
- Any additional order-related details, such as gift messages or special instructions, are supported.
- The order summary and final total are displayed correctly before confirming the purchase.
- The order confirmation and receipt generation, including email notifications, are implemented.

5.3.3.2 Exit Criteria/DOD

Exit Criteria (also known as Definition of Done or DoD) for search, add to cart, and checkout functionalities iStarschiners.ro website will include the following:

Search Functionality:

- The search functionality should accurately return relevant results based on user queries.
- The search results should be properly ranked based on relevance.
- The search should handle common user inputs, such as misspellings or synonyms.
- The search results page should display the correct product information, including title, description, price, and availability.
- Pagination or infinite scrolling should be implemented to handle large result sets.
- The search functionality should be tested and verified against different scenarios and edge cases.

Add to Cart Functionality:

- Users should be able to add products to the cart from product detail pages or search results.
- The cart should display the correct product information, including title, price, and quantity.
- The total price of all items in the cart should be accurately calculated.
- Users should be able to update the quantity or remove items from the cart.
- The cart should be properly synchronized across sessions and devices.
- The add to cart functionality should be tested and verified against different scenarios, such as adding multiple products or adding products with various attributes.

Checkout Functionality:

- Users should be able to proceed to checkout from the cart.
- The checkout process should guide the user through the necessary steps, such as providing shipping and billing information.
- Users should receive proper validation and error messages if any required information is missing or incorrect.
- The checkout process should securely handle sensitive information, such as credit card details.
- Users should receive a confirmation or order summary after successfully completing the checkout process.
- The checkout functionality should be tested and verified against different scenarios, including different payment methods, international shipping, and guest checkout options.

6.Environment requirements/Production environment

Environment Requirements:

Production Environment:

- The production environment should be stable, reliable, and highly available.
- It should have sufficient resources to handle the expected user load.
- The environment should be securely isolated from unauthorized access and protected against cyber threats.
- Regular backups should be performed to ensure data integrity and disaster recovery.

- Monitoring and alerting mechanisms should be in place to detect and respond to any issues promptly.

Development and Testing Environments:

- Development and testing environments should mirror the production environment as closely as possible.
- They should provide a safe and isolated space for development and testing activities without affecting the production environment.
- It should be possible to simulate realistic user scenarios and test different configurations.
- Proper version control and release management processes should be in place to ensure code and configuration changes are properly tracked and deployed.

Staging Environment:

- A staging environment is often used to mimic the production environment and test the application before deployment.
- It should closely resemble the production environment in terms of infrastructure and configurations.
- Testing in the staging environment helps identify any potential issues or incompatibilities before releasing changes to the production environment.

Overview of System Infrastructure Components:

Servers and Virtual Machines:

- Physical or virtual servers are the core components of the system infrastructure.
- They host the application, database, and other necessary software components.
- Load balancers may be used to distribute incoming requests across multiple servers for scalability and redundancy.

Networking:

- Networking components, such as routers, switches, and firewalls, ensure connectivity and security within the infrastructure.
- Network configuration should allow proper traffic routing and firewall rules to protect against unauthorized access.

Databases:

- Databases store and manage the application's data.
- Relational databases (e.g., MySQL, PostgreSQL) or NoSQL databases (e.g., MongoDB, Redis) may be used based on the application's requirements.

Storage:

- Storage systems, such as Network Attached Storage (NAS) or Storage Area Networks (SAN), provide persistent storage for application data and files.
- They ensure data durability, availability, and scalability.

Monitoring and Logging:

- Monitoring tools collect metrics and log data to monitor the health and performance of the infrastructure and applications.
- Monitoring helps detect issues, track system behavior, and ensure SLA compliance.

The management of environments typically involves collaboration of:

- Infrastructure Team: Responsible for managing servers, networking, storage, and related components. They handle tasks such as provisioning, configuring, monitoring, and maintaining the infrastructure.
- DevOps or Release Management Team: Responsible for deployment, configuration management, and release processes. They ensure smooth deployments, handle environment configurations, and manage version control systems.
- Development Team: Responsible for developing and testing the application code, ensuring compatibility with the target environments, and working closely with the infrastructure and DevOps teams.
- QA Team: Responsible for testing the application in various environments and ensuring quality and stability before deployment to production.

- Security Team: Responsible for implementing security measures, such as access control, encryption, and vulnerability management, to protect the environments and applications.

7. Test data requirements

For test data requirements like sign up, login, sign up with Facebook, reset password, search, add to cart, and checkout functionalities you will apply this:

Sign Up Functionality:

- Test Input Data: You need to provide valid and invalid data for fields such as name, email, password, and additional user profile information.
- Test Reference Data: You can have a set of expected outcomes for different scenarios, such as successful sign up, validation errors for invalid inputs, or duplicate email detection.
- Different Phases: During different phases of testing, you can use different datasets to cover various scenarios, such as testing with unique email addresses, testing with common passwords, or testing with special characters in the name field.

Login Functionality:

- Test Input Data: You need to provide valid and invalid email/username and password combinations.
- Test Reference Data: You can have a set of expected outcomes for different scenarios, such as successful login, incorrect credentials, or account lockout after multiple failed attempts.
- Different Phases: In different phases, you can use datasets with different combinations of valid and invalid credentials, including edge cases like long passwords, special characters, or different email formats.

Sign Up with Facebook Functionality:

- Test Input Data: You need to simulate the Facebook authentication process, which involves obtaining access tokens or test users from Facebook's developer platform.
- Test Reference Data: You can have a set of expected outcomes for different scenarios, such as successful authentication, error handling for invalid access tokens, or linking an existing account with Facebook.
- Different Phases: In different phases, you can use different sets of access tokens or test users to cover various scenarios, such as testing with different Facebook account types or testing with expired tokens.

Reset Password Functionality:

- Test Input Data: You need to provide valid and invalid email addresses or usernames to trigger the password reset process.
- Test Reference Data: You can have a set of expected outcomes for different scenarios, such as successful password reset, error handling for non-existent or blocked accounts, or expiration of reset links.
- Different Phases: Different phases may involve testing with different email addresses, including valid ones, ones that do not exist in the system, or ones associated with blocked accounts.

Search Functionality:

- Test Input Data: You need to provide different search queries, including valid and invalid ones, to test the search functionality.
- Test Reference Data: You can have a set of expected search results for different queries, including both successful matches and empty results.
- Different Phases: Different phases may involve testing with different search queries, including short queries, long queries, queries with special characters, or queries with common misspellings.

Add to Cart Functionality:

- Test Input Data: You need to provide different product IDs or SKUs to simulate adding items to the cart.
- Test Reference Data: You can have a set of expected outcomes for different scenarios, such as successful addition to the cart, handling out-of-stock items, or validation errors for invalid product identifiers.
- Different Phases: Different phases may involve testing with different product IDs, including valid ones, ones that do not exist, or ones associated with unavailable products.

Checkout Functionality:

- Test Input Data: You need to provide data for shipping addresses, billing information, payment methods, and order details.
- Test Reference Data: You can have a set of expected outcomes for different scenarios, such as successful checkout, validation errors for missing or incorrect information, or handling of different payment scenarios.
- Different Phases: Different phases may involve testing with different combinations of shipping addresses, billing information, or payment methods, including valid and invalid ones, international addresses, or different payment gateways.

The test data requirements for these functionalities involve providing various inputs and expected outcomes for different scenarios, covering both valid and invalid cases.

8 TESTING TOOLS & TECHNIQUES

Testing tools and techniques are essential for ensuring the quality and reliability of software applications. There are various tools and techniques available to assist in different phases and types of testing.

Test Management Tools:

- JIRA: Popular issue tracking and project management tool that can be customized for test management.
- Xray: Test management tool integrated with JIRA.

Testing techniques:

Functionality Testing

Compatibility Testing

Content Testing

Cross-Browser Testing

8.1 REQUIREMENTS & USE CASE MANAGEMENT

To manage the requirements and use cases for the following functionalities: sign up, login, sign up with Facebook, reset password, search, add to cart, and checkout for a website like Starschiners.ro, you need to follow this steps:

Identify the stakeholders: Determine the key stakeholders involved in the process, such as users, administrators, developers, and any other relevant parties.

Gather requirements: Engage with stakeholders to gather their requirements and expectations for each functionality. This can be done through interviews, surveys, or meetings.

Define use cases: Based on the requirements gathered, create detailed use cases for each functionality. A use case describes a specific interaction between an actor (user) and the system. For example:

a. Sign up:

- Use case: User Registration
- Actors: New User, System
- Preconditions: None
- Main Flow:
    - i. User navigates to the registration page.
    - ii. User fills in the required information (name, email, password, etc.).
    - iii. User submits the registration form.
    - iv. System validates the information and creates a new user account.
    - v. System sends a confirmation email to the user.
    - vi. Use case ends.
- Alternate Flows: Error handling for invalid inputs, existing email address, etc.

- b. Login:

    - Use case: User Login
    - Actors: Registered User, System
    - Preconditions: User account exists
    - Main Flow:
        - vii. User navigates to the login page.
        - viii. User enters their email and password.
        - ix. User submits the login form.
        - x. System validates the credentials and authenticates the user.
        - xi. User gains access to their account.
        - xii. Use case ends.
    - Alternate Flows: Error handling for invalid credentials, forgotten password, etc.

- c. Sign up with Facebook:
    - Use case: User Registration with Facebook
    - Actors: New User, Facebook API, System
    - Preconditions: User has a Facebook account
    - Main Flow:
        xiii. User clicks on the "Sign up with Facebook" button.
        xiv. User is redirected to Facebook for authentication.
        xv. User grants permission to share basic information.
        xvi. Facebook API returns the user's information to the system.
        xvii. System creates a new user account using the received information.
        xviii. User gains access to their account.
        xix. Use case ends.
    - Alternate Flows: Error handling for failed authentication, user denying permission, etc.
- d. Reset password:
    - Use case: Password Reset
    - Actors: User, System
    - Preconditions: User forgot their password
    - Main Flow:
        xx. User clicks on the "Forgot password" link.
        xxi. User enters their email address.
        xxii. User submits the password reset request.
        xxiii. System verifies the email address and sends a reset link.
        xxiv. User receives the reset link via email.
        xxv. User clicks on the link and is redirected to the password reset page.
        xxvi. User enters a new password and confirms it.
        xxvii. User submits the password reset form.
        xxviii. System updates the password and notifies the user.
        xxix. Use case ends.
    - Alternate Flows: Error handling for invalid email address, expired reset link, etc.
- e. Search:
    - Use case: Product Search

    - Actors: User, System
    - Preconditions: User is on the website
    - Main Flow:
        xxx. User enters a search query in the search bar.
        xxxi. User submits the search query.
        xxxii. System processes the query and retrieves matching products.
        xxxiii. System displays the search results to the user.
        xxxiv. User selects a product or refines the search query.
        xxxv. Use case ends.
    - Alternate Flows: Error handling for no matching results, search suggestions, etc.

- f. Add to cart:

    - Use case: Add Product to Cart
    - Actors: User, System
    - Preconditions: User is logged in and product is available
    - Main Flow:
        xxxvi. User navigates to a product page.
        xxxvii. User selects the desired product options (size, color, quantity, etc.).
        xxxviii. User clicks on the "Add to Cart" button.
        xxxix. System adds the selected product to the user's shopping cart.
        xl. System updates the cart total and displays a confirmation message.
        xli. User can continue shopping or proceed to checkout.
        xlii. Use case ends.
    - Alternate Flows: Error handling for out-of-stock products, invalid options, etc.

- g. Checkout:

    - Use case: Checkout Process
    - Actors: User, System, Payment Gateway
    - Preconditions: User has items in the shopping cart
    - Main Flow:
        xliii. User clicks on the "Checkout" button.
        xliv. User is redirected to the checkout page.
        xlv. User enters the shipping and billing information.

xlvi. User selects a payment method (credit card, PayPal, etc.).
xlvii. User submits the checkout form.
xlviii. System validates the information and calculates the total cost.
xlix. System redirects the user to the selected payment gateway.
l. User completes the payment process on the payment gateway's site.
li. System receives the payment confirmation.
lii. System updates the order status and sends an order confirmation to the user.
liii. Use case ends.

- Alternate Flows: Error handling for payment failures, address validation, etc.

14. Review and refine: Share the use cases with stakeholders for review and feedback. Make necessary revisions to ensure clarity and completeness.
15. Implement and test: Developers can use the defined use cases as a guideline for implementing the functionalities. Testers can use them to create test cases and ensure the functionalities work as expected.

By following these steps, you can effectively manage the requirements and use cases for the sign up, login, sign up with Facebook, reset password, search, add to cart, and checkout functionalities for a website like Starschiners.ro.

8.2 TEST MANAGEMENT & DEFECT TRACKING

When it comes to test management and defect tracking for various functionalities on Starschiners.ro, like sign up, login, sign up with Facebook, reset password, search, add to cart, and checkout, you can follow the following steps:

Test Management:

a. Test Plan: Create a comprehensive test plan that outlines the testing approach, objectives, scope, test environments, test data, and test cases for each functionality.

b. Test Cases: Design test cases for each functionality, covering positive and negative scenarios, boundary conditions, and edge cases.

c. Test Execution: Execute the test cases, record the test results, and capture any defects encountered during testing.

d. Test Coverage: Monitor the test coverage to ensure that all functionalities and scenarios are adequately tested.

Defect Tracking:

a. Issue Logging: Use a defect tracking system (e.g., JIRA, Bugzilla) to log and track issues/defects encountered during testing.

b. Issue Description: Provide detailed information about each defect, including steps to reproduce, expected behavior, actual behavior, screenshots, and logs.

c. Issue Prioritization: Prioritize the defects based on their severity and impact on the application's functionality.

d. Issue Assignment: Assign the defects to the respective development team or individual for resolution.

e. Issue Resolution: Monitor the defect resolution process, communicate with the development team, and verify fixes once they are implemented.

f. Issue Closure: Verify that the resolved issues are indeed fixed and close them in the defect tracking system.

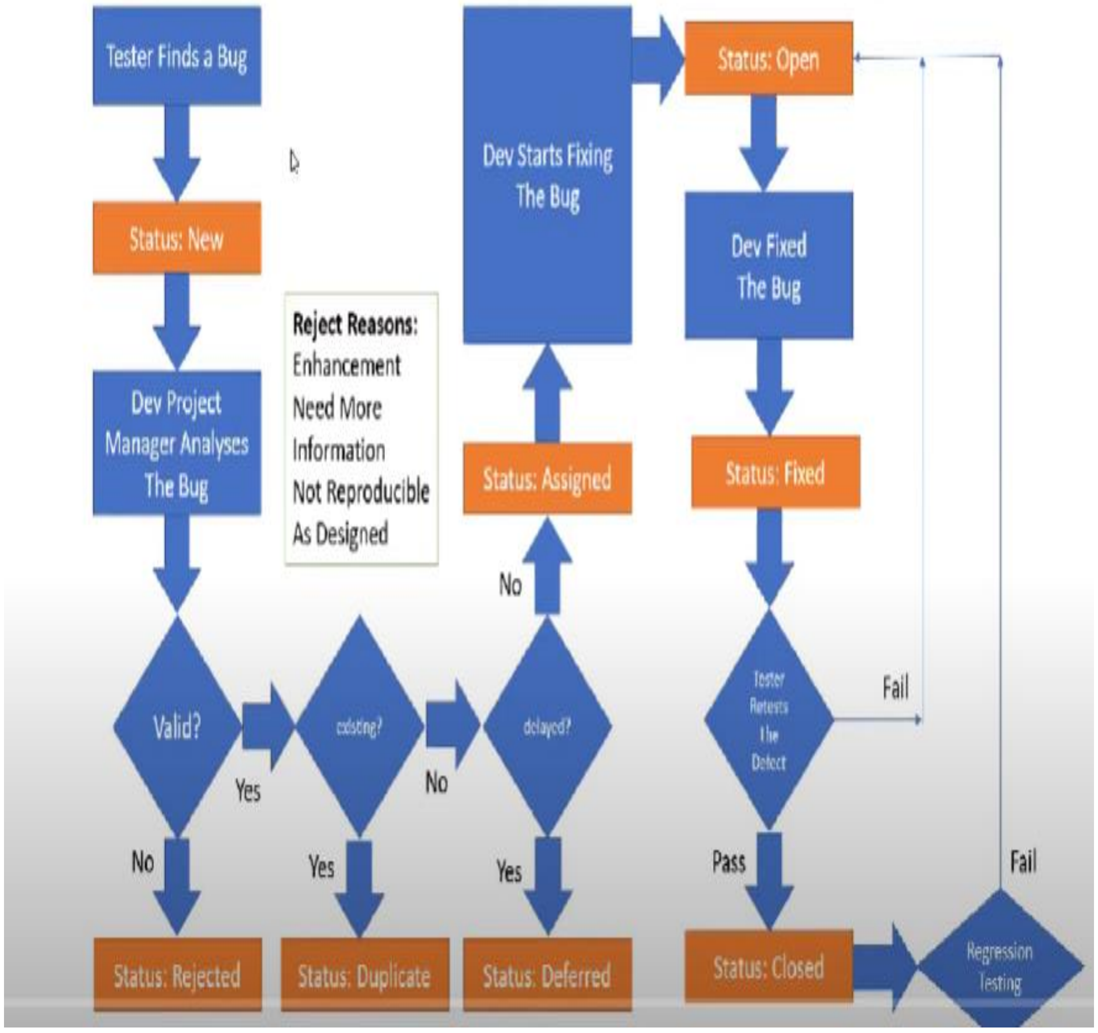
9 TEAM ROLES & RESPONSIBILITIES

Activity	Responsibility/Ownership	Name

Test Plan Creation	Test Manager	Ioana Marcu
Test Phase Plan Creation	Test Lead	Ioana Barb
Test Management	Development Company Test Manager & System Test Lead	Raul Popovici
Test Analysis and Design	Development Company Test Engineers	Ivan Marin
Test Preparation, Execution & Results	Development Company Test Engineers	George Apostolescu
Test Defect Submission	Development Company Test Engineers	Roxana Ivanescu
Test Summary Reporting	Development Company Test Engineers	Cosmin Nedelcu
Test Completion Reporting	Development Company Test Manager & Test Lead	Raul Popovici
Test Environment Deployment	Development Company Test Manager & Test Lead	Raul Popovici

10 Defect Management

Bug Life Cycle



10.1 Defect Management Process/Status

Defect management process typically involves the following steps:

Bug Reporting: Defects or bugs are identified by individuals within the organization, such as developers, testers, or end users. The person who identifies a bug opens a bug report by providing relevant information about the defect, such as a description, steps to reproduce, and any associated files or screenshots.

Bug Assignment: Once a bug report is opened, it needs to be assigned to the appropriate team or individual responsible for fixing it. This assignment ensures that the bug is properly tracked and addressed within the development process.

Bug Prioritization: Bugs are prioritized based on their severity and impact on the software/application. The priority level helps determine the order in which bugs will be addressed. High-priority bugs that severely impact functionality or usability are typically given more immediate attention.

Bug Fixing: The assigned developer or team starts working on fixing the bug. They analyze the reported issue, debug the code, and make the necessary changes to eliminate the defect.

Bug Verification: After the bug is fixed, it undergoes verification to ensure that the resolution is successful and the defect is resolved. Testers or quality assurance personnel test the fixed bug by following the steps provided in the bug report and confirm whether the issue is resolved.

Bug Closure: If the verification step confirms that the bug is resolved, the bug report is marked as closed. The person responsible for the fix typically closes the bug. At this point, the bug is considered resolved and no further action is required.

Throughout this process, the status of each bug is updated to reflect its current state, such as "Open," "Assigned," "In Progress," "Fixed," "Verified," or "Closed." These statuses help track the progress and provide visibility into the defect management process.

When providing defect comments, you need to include the following information for understanding and resolving the issue:

16. **Reproducibility**: Clearly describe the steps or actions necessary to reproduce the defect. Provide a detailed sequence of operations, including

any specific configurations, inputs, or data required. The goal is to enable others to replicate the problem consistently.

17. **Expected Results**: State the behavior or outcome that was expected when performing the actions or operations described above. This helps establish the intended functionality or behavior that the system or software should exhibit.
18. **Actual Results**: Describe the observed behavior or outcome that differed from what was expected. Be specific and provide relevant details, such as error messages, system responses, or any other relevant information. This information is crucial for understanding the deviation from expected behavior.

TEST SCHEDULE

Ref no	Stage	Profect milestone	Due Date
BRN 1	Test Preparation Stage	Phase Test Plan document completed.	16.04.2023

BRN 2	Test Preparation Stage	Test analysis completed on the detailed requirements and technical documentation	29.04.2023
BRN 3	Test Execution Stage	Test Conditions/Cases/ completed and signed off.	23.05.2023
BRN 4	Test Execution Stage	Execution of Test cases completed.	31.05.2023

REFERENCED DOCUMENTS

DOCUMENT	AUTHOR	DESCRIPTION
Functional Requirements Document (FRD)	Team document	Outlines the specific features, functions, and capabilities that a software system or application must have in

		order to meet the needs of its users
Functional Requirements Specification Document(FRS)	Team document	outlines the functional requirements of a software system or application.