Acknowledgments

I would like to express my sincere gratitude to my supervisor, Mr. Adel Ayari, for his guidance and support throughout the internship. I am grateful for his valuable feedback and advice, which have been instrumental in the successful completion of this project.

I am particularly thankful to Mr. Zino Abidi for guiding me through Playwright integration and sharing valuable insights on CI/CD workflows, which were instrumental in completing my tasks.

I would like to also thank my colleagues at the company for their help and support during the internship. I am grateful for their assistance and collaboration, which have made this internship a rewarding and enriching experience.

List of Abbreviations

QA	Quality Assurance
CI/CD	Continuous Integration/Continuous Deployment
E2E	End-to-End
UX/UI	User Experience/User Interface
VAT	Value-Added Tax
BVA	Boundary Value Analysis
EP	Equivalence Partitioning

Contents

List of Figures

Introduction

This report presents the work carried out during the internship at SeekMake for a duration of two months.

The internship was focused on software testing and quality assurance of the SeekMake web application. It covers the testing journey from exploratory testing to end-to-end testing and automation.

The report is divided into four chapters:

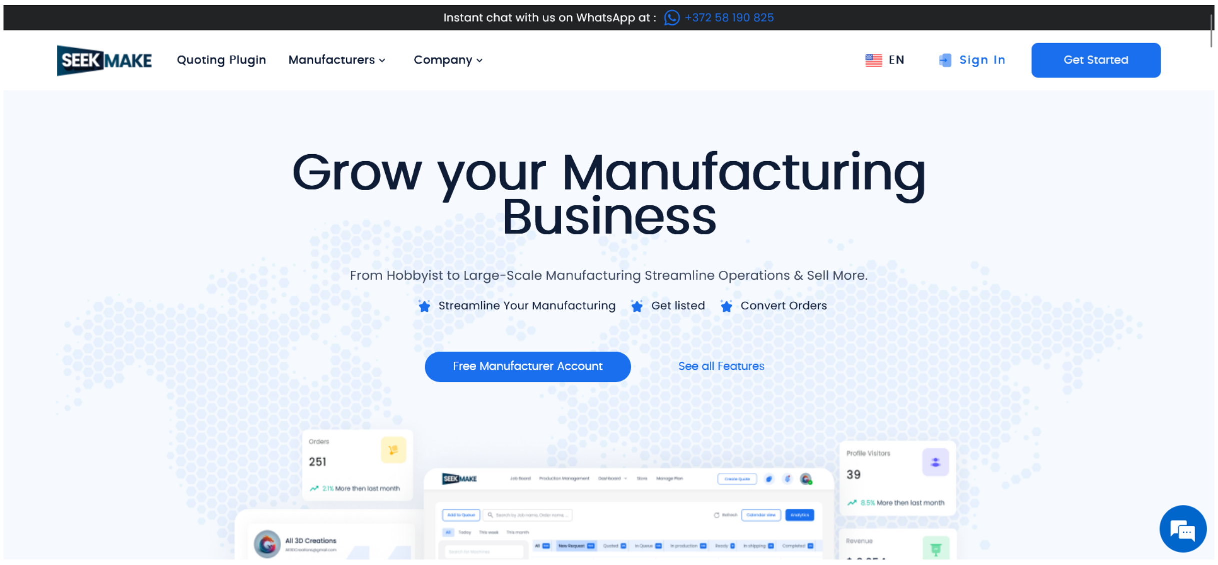
- **Chapter 1: Company Overview** provides an introduction to SeekMake, a leading provider of digital transformation services.
- **Chapter 2: Quality Assurance Framework** covers the basics of software testing and quality assurance, including the testing lifecycle and methodologies.
- **Chapter 3: Exploratory Testing Insights** discusses the exploratory testing carried out during the internship.
- **Chapter 4: End-to-End Testing and Automation** discusses the end-to-end testing and automation of the SeekMake web application.

Chapter 1: Company Overview

1.1. About SeekMake

SeekMake, a leading provider of digital transformation services, specializes in streamlining manufacturing processes through its comprehensive online platform. Key offerings include instant quoting, CAD tools, secure file storage, and analytics, all designed to reduce development cycles and enhance project delivery speed. These services empower manufacturers to manage production, customer interactions, and payments efficiently through an intuitive web interface. [13]

Figure 1.1: SeekMake Logo



Instant chat with us on WhatsApp at : +372 58 190 825
SEEKMAKE
Quoting Plugin
Manufacturers
Company
EN
Sign In
Get Started
Grow your Manufacturing Business
From Hobbyist to Large-Scale Manufacturing Streamline Operations & Sell More.
Streamline Your Manufacturing
Get listed
Convert Orders
Free Manufacturer Account
See all Features
Orders
251
Profile Visitors
39
All 3D Creations

Figure 1.2: A screenshot of SeekMake's homepage

Chapter 2: Quality Assurance Framework

2.1. Introduction

This chapter presents the quality assurance framework used during the software testing process at SeekMake.

It covers quality assurance vs testing, the Scrum methodology, the integration of DevOps, testing strategy, testing types, testing tools, and testing techniques used to ensure the quality of the software.

2.2. Quality Assurance vs Testing

While people often use the terms “testing” and “quality assurance” (QA) interchangeably, testing and QA are not the same [2]:

2.2.1. Quality Assurance

QA is a process-oriented, preventive approach that focuses on the implementation and improvement of processes. It works on the basis that if a good process is followed correctly, then it will generate a good product. [2]

QA at SeekMake guided the testing process by defining the testing strategy, planning the testing activities, and ensuring that the testing activities were aligned with the sprint goals.

2.2.2. Software Testing

Testing is a product-oriented, corrective approach that focuses on those activities supporting the achievement of appropriate levels of quality. [2]

Testing at SeekMake was carried out to identify and fix defects[1] in the web application. This corrective approach was complemented by QA activities that focused on process improvement and defect prevention.

2.2.3. Differences and Complementarity

Test results are used by QA and testing. In testing they are used to fix defects, while in QA they provide feedback on how well the development and test processes are performing. [2]

2.3. Scrum Methodology

At SeekMake, the testing process was aligned with the Scrum methodology.

2.3.1. Scrum Overview

Scrum helps people and teams deliver value incrementally in a collaborative way. As an agile framework, Scrum provides just enough structure for people and teams to integrate into how they work, while adding the right practices to optimize for their specific needs. [12]

At SeekMake, Scrum was used to manage the testing process, ensuring that the testing activities were aligned with the sprint goals and that the testing process was integrated with the development activities.

2.3.2. Scrum Team

The fundamental unit of Scrum is a small team of people, a Scrum Team. The Scrum Team consists of one Scrum Master, one Product Owner, and Developers. [12]

At SeekMake, the Scrum Team consisted of developers (including QA testers and UX/UI designers), a product owner, and a Scrum master.

[1]A defect is an error, flaw, failure, or fault in a computer program that causes it to produce an incorrect or unexpected result, or to behave in unintended ways

Scrum Master

the Scrum Master is accountable for establishing Scrum. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization while serving the Scrum Team as well as the larger organization. [12]

Product Owner

The Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. How this is done may vary widely across organizations, Scrum Teams and individuals. [12]

Developers

Developers are the people on the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint. [12]

2.3.3. Scrum Events

Each event in Scrum is a formal opportunity to inspect and adapt Scrum artifacts. These events are specifically designed to enable the transparency required. Failure to operate any events as prescribed results in lost opportunities to inspect and adapt. Events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. [12]

Sprints

Sprints are fixed length periods of work that last one month or less to create consistency and ensure short iterations for feedback in order to inspect and adapt both how work is done and what is being worked on. [12]

At SeekMake, sprints lasted two weeks and were used to define the scope of the testing activities, assign tasks to team members, and estimate the effort required to complete the testing activities.

Sprint Planning

Sprint Planning initiates the Sprint by laying out the work to be performed for the Sprint. This resulting plan is created by the collaborative work of the entire Scrum Team. [12]

Daily Scrum

The Daily Scrum (or daily Stand-Up) is a 15-minute event for the Developers of the Scrum Team. To reduce complexity, it is held at the same time and place every working day of the Sprint. [12]

At SeekMake, daily stand-ups were used to share updates, collaborate on testing activities, and resolve issues.

2.4. Integration of DevOps

At SeekMake, the testing process was integrated with DevOps practices to ensure that the testing activities were aligned with the development activities.

2.4.1. DevOps Overview

DevOps is an organizational approach aiming to create synergy by getting development (including testing) and operations to work together to achieve a set of common goals. [2]

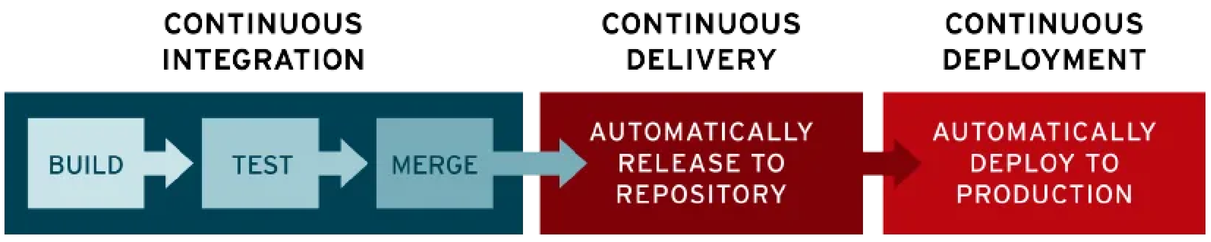


Figure 2.1: DevOps practices.

Continuous Integration

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It's a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run. Automated tools are used to assert the new code's correctness before integration. [1]

Continuous Delivery

Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage. [1]

Continuous Deployment

Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production. [1]

2.5. Testing Strategy

The testing strategy outlines the approach that was adopted during the software testing process at SeekMake. It includes the testing objectives, scope, and resources required to carry out the testing activities.

2.5.1. Testing Objectives

The primary testing objectives were to ensure seamless user experiences by validating the core functionalities of SeekMake's web application. Specific goals included verifying the checkout workflow, ensuring secure payment processing, and assessing the application's compatibility across major browsers and devices.

2.5.2. Testing Scope

The testing scope included the following areas of the web application:

- **Functionality:** Verify that the web application functions as expected.

- **Performance:** Evaluate the performance of the web application under different conditions.
- **Usability:** Assess the usability of the web application from an end-user perspective.
- **Compatibility:** Test the web application on different devices and browsers to ensure compatibility.
- **Security:** Identify and address security vulnerabilities in the web application.

2.5.3. Testing Resources

The testing resources required to carry out the testing activities included the following:

- **QA Testers:** QA testers responsible for designing, executing, and reporting test cases.
- **Developers:** Developers responsible for fixing defects identified during testing.
- **Product Owner:** Product owner responsible for defining the requirements and acceptance criteria.
- **Scrum Master:** Scrum master responsible for facilitating the testing process and ensuring that the testing activities are aligned with the sprint goals.
- **Test Environment:** Test environments including production, staging, and development environments to test the web application in different scenarios.
- **Testing Tools:** Testing tools including GitHub, Playwright, Microsoft Teams, BrowserStack Live, and Lighthouse to manage the testing process, automate testing activities, and track defects.
- **Documentation:** Test plans, test cases, test reports, and defect reports to document the testing activities and results.
- **Communication:** Regular communication between team members to share updates, collaborate on testing activities, and resolve issues.
- **Training:** Training sessions to familiarize team members with the testing tools and techniques used during the testing process.

- **Feedback:** Feedback sessions to provide continuous feedback on the testing process and identify areas for improvement.

2.6. Testing Types

A lot of test types exist and can be applied in projects [2]. During this internship, the following three test types are addressed:

2.6.1. Functional Testing

Functional testing evaluates the functions that a component or system should perform. The functions are "what" the test object should do. The main objective of functional testing is checking the functional completeness, functional correctness and functional appropriateness. [2]

At SeekMake, functional testing was carried out to evaluate the functionality of the web application.

2.6.2. Non-Functional Testing

Non-functional testing evaluates attributes other than functional characteristics of a component or system. Non-functional testing is the testing of "how well the system behaves". The main objective of non- functional testing is checking the non-functional quality characteristics. [2]

At SeekMake, non-functional testing was carried out to evaluate the performance, usability, compatibility, and security of the web application.

2.6.3. Black-box Testing

Black-box testing is specification-based and derives tests from documentation not related to the internal structure of the test object. The main objective of black-box testing is checking the system's behavior against its specifications. [2]

At SeekMake, black-box testing was carried out to evaluate the functionality of the web application from an end-user perspective.

2.7. Testing Tools

Various testing tools were used during the software testing process at SeekMake.

2.7.1. GitHub, GitHub Actions and GitHub Projects

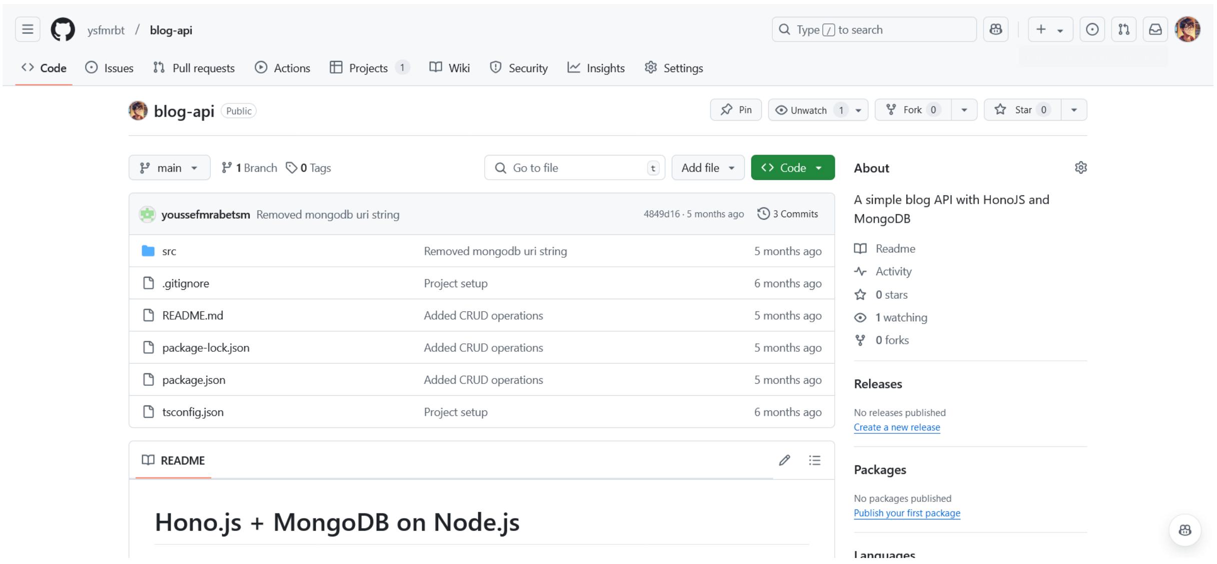
GitHub

GitHub is a cloud-based platform where you can store, share, and work together with others to write code. [5]

Collaborative working, one of GitHub's fundamental features, is made possible by the open-source software, Git, upon which GitHub is built. [5]

Git is a version control system that intelligently tracks changes in files. Git is particularly useful when you and a group of people are all making changes to the same files at the same time. [5]

At SeekMake, GitHub was used to store the source code in repositories, manage branches, and collaborate on code changes.



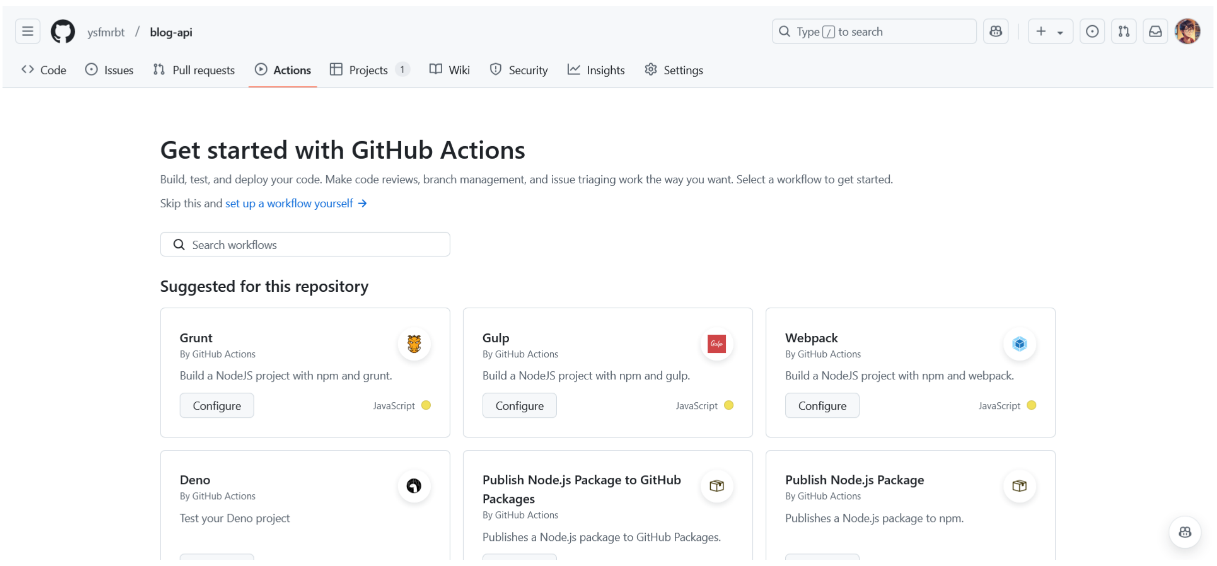
src	Removed mongodb uri string	5 months ago
.gitignore	Project setup	6 months ago
README.md	Added CRUD operations	5 months ago
package-lock.json	Added CRUD operations	5 months ago
package.json	Added CRUD operations	5 months ago
tsconfig.json	Project setup	6 months ago

Figure 2.2: An example of a GitHub repository.

GitHub Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. [6]

GitHub Actions automated the execution of Playwright scripts, providing immediate feedback on code changes during sprints.


ysfmrbt / blog-api

Type / to search

Code | Issues | Pull requests | Actions | Projects 1 | Wiki | Security | Insights | Settings

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and set up a workflow yourself →

Search workflows

Suggested for this repository

Grunt
By GitHub Actions
Build a NodeJS project with npm and grunt.
Configure | JavaScript

Gulp
By GitHub Actions
Build a NodeJS project with npm and gulp.
Configure | JavaScript

Webpack
By GitHub Actions
Build a NodeJS project with npm and webpack.
Configure | JavaScript

Deno
By GitHub Actions
Test your Deno project

Publish Node.js Package to GitHub Packages
By GitHub Actions
Publishes a Node.js package to GitHub Packages.

Publish Node.js Package
By GitHub Actions
Publishes a Node.js package to npm.

Figure 2.3: GitHub Actions.

GitHub Projects

Projects is an adaptable, flexible tool for planning and tracking work on GitHub. [7]

At SeekMake, GitHub Projects was used to manage the testing process, track defects, and collaborate on testing activities.

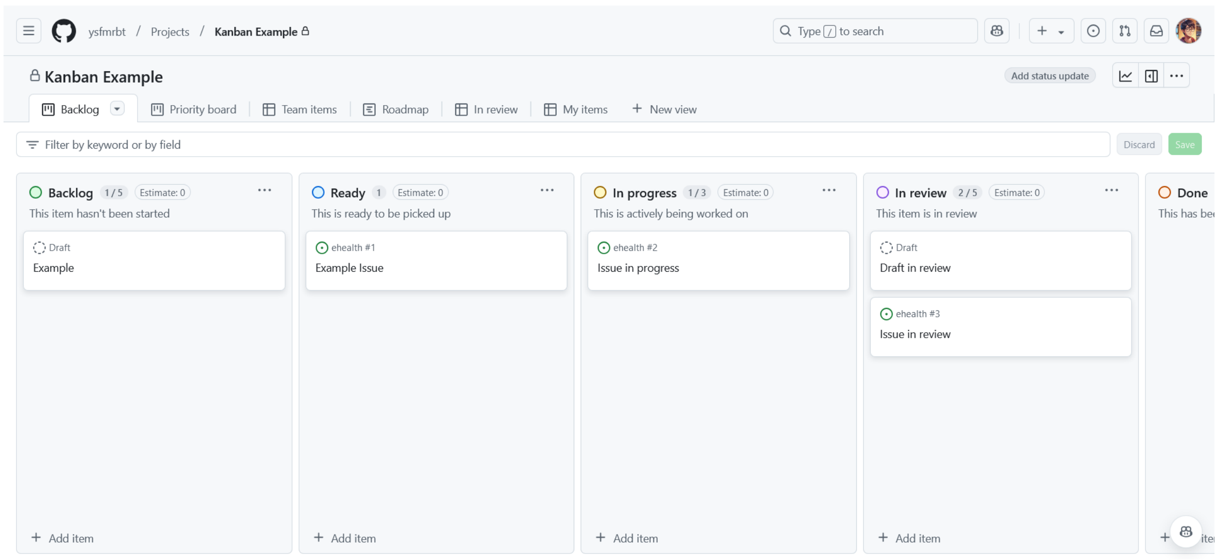


Figure 2.4: A GitHub project board resembling the Kanban board used at SeekMake.

2.7.2. Playwright

Playwright is a testing framework that enables reliable end-to-end testing for modern web apps [11]. It was created specifically to accommodate the needs of end-to-end testing. Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox. Test on Windows, Linux, and macOS, locally or on CI, headless or headed with native mobile emulation. [10]

Playwright was used for end-to-end testing, enabling automation across Chromium, WebKit, and Firefox browsers. It streamlined test execution for critical paths, such as the login process and account management workflows.

Figure 2.5: Playwright logo.

2.7.3. Microsoft Teams

Microsoft Teams is a Meet, chat, call, and collaborate with communications software that makes it easy to work, plan, and innovate together. [9]

At SeekMake, Microsoft Teams was used to communicate with team members, share updates, and collaborate on testing activities.

Figure 2.6: Microsoft Teams logo.

2.7.4. BrowserStack Live

BrowserStack Live is a cross browser on desktop and mobile testing platform. [3]

At SeekMake, BrowserStack Live was used to test the web application on Apple devices.

Figure 2.7: Browser Stack logo.

2.7.5. Lighthouse

Lighthouse is an open-source, automated tool to help you improve the quality of web pages. You can run it on any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO, and more.

At SeekMake, Lighthouse was used to assess web application performance and highlight areas for improvement, such as load times and accessibility.

Figure 2.8: Lighthouse logo.

2.8. Testing Techniques

Test techniques support the tester in test analysis (what to test) and in test design (how to test). Test techniques help to develop a relatively small, but sufficient, set of test cases in a systematic way. Test techniques also help the tester to define test conditions, identify coverage items, and identify test data during the test analysis and design. [2]

At SeekMake, the following testing techniques were used to design test cases:

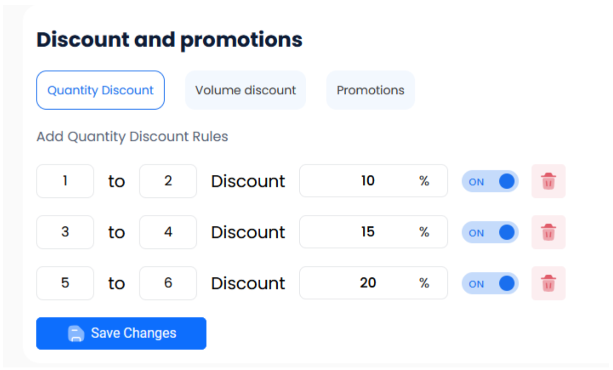
2.8.1. Black-box Testing Techniques

Black-box test techniques (also known as specification-based techniques) are based on an analysis of the specified behavior of the test object without reference to its internal structure. Therefore, the test cases are independent of how the software is implemented. Consequently, if the implementation changes, but the required behavior stays the same, then the test cases are still useful. [2]

Equivalence Partitioning

Equivalence Partitioning (EP) divides data into partitions (known as equivalence partitions) based on the expectation that all the elements of a given partition are to be processed in the same way by the test object. The theory behind this technique is that if a test case, that tests one value from an equivalence partition, detects a defect, this defect should also be detected by test cases that test any other value from the same partition. Therefore, one test for each partition is sufficient. [2]

At SeekMake, we used equivalence partitioning to design and execute test cases that verify discounts based on intervals of quantities.



Discount and promotions

Add Quantity Discount Rules

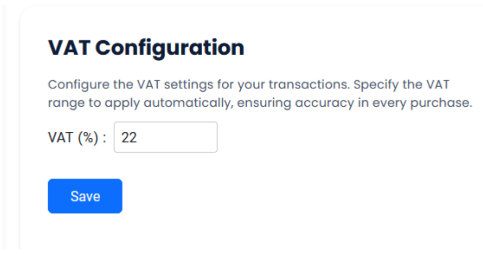
1	to	2	Discount	10	%	ON
3	to	4	Discount	15	%	ON
5	to	6	Discount	20	%	ON

Figure 2.9: Screenshot taken from SeekMake's web application showing discounts based on intervals of quantities.

Boundary Value Analysis

Boundary Value Analysis (BVA) is a test technique based on exercising the boundaries of equivalence partitions. Therefore, BVA can only be used for ordered partitions. The minimum and maximum values of a partition are its boundary values. In the case of BVA, if two elements belong to the same partition, all elements between them must also belong to that partition. [2]

At SeekMake, we used boundary value analysis to design and execute test cases that verify the behavior of the web application when the percentage of VAT is at the minimum and maximum values.


VAT Configuration

Configure the VAT settings for your transactions. Specify the VAT range to apply automatically, ensuring accuracy in every purchase.

VAT (%) : 22

Figure 2.10: Screenshot taken from SeekMake's web application showing the percentage of VAT.

2.8.2. Experience-based Testing Techniques

Experience-based test techniques effectively use the knowledge and experience of testers for the design and implementation of test cases. The effectiveness of these test techniques depends heavily on the tester's skills. Experience-based test techniques can detect defects that may be missed using the black-box test techniques and white-box test techniques. Hence, experience-based test techniques are complementary to the black-box test techniques and white-box test techniques. [2]

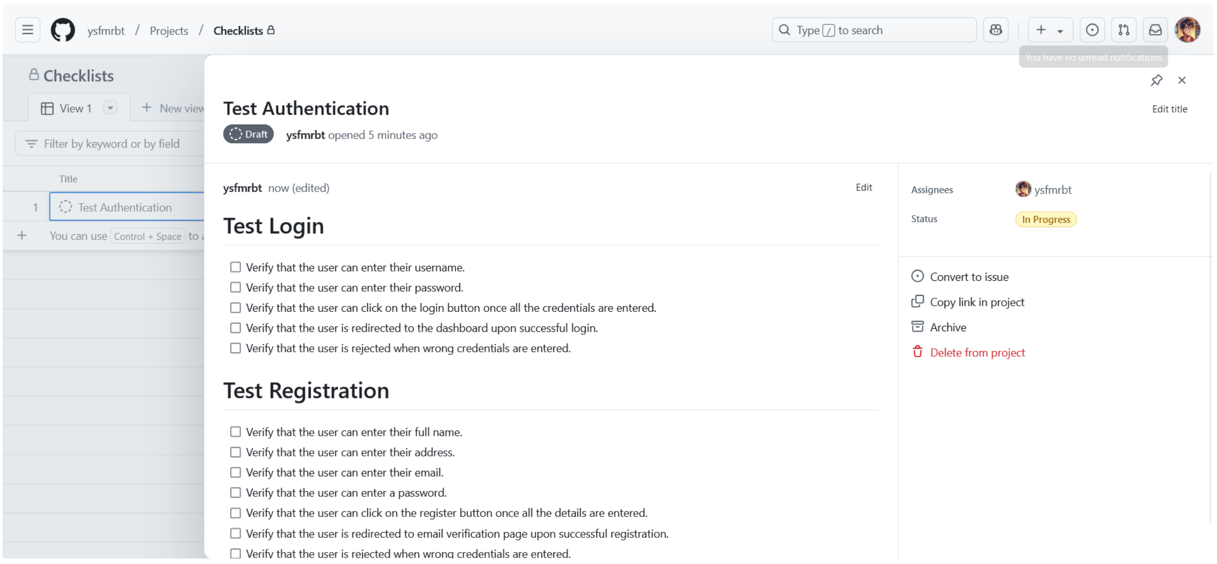
Exploratory Testing

In exploratory testing, tests are simultaneously designed, executed, and evaluated while the tester learns about the test object. The testing is used to learn more about the test object, to explore it more deeply with focused tests, and to create tests for untested areas. [2]

At SeekMake, exploratory testing was carried out while we were learning about the web application and exploring its functionalities.

Checklist-based Testing

In checklist-based testing, a tester designs, implements, and executes tests to cover test conditions from a checklist. Checklists can be built based on experience, knowledge about what is important for the user, or an understanding of why and how software fails. Checklists should not contain items that can be checked automatically, items better suited as entry criteria, exit criteria, or items that are too general (Brykczynski 1999). [2]



Test Authentication

ysfmrbt opened 5 minutes ago

ysfmrbt now (edited)

Test Login

- ☐ Verify that the user can enter their username.
- ☐ Verify that the user can enter their password.
- ☐ Verify that the user can click on the login button once all the credentials are entered.
- ☐ Verify that the user is redirected to the dashboard upon successful login.
- ☐ Verify that the user is rejected when wrong credentials are entered.

Test Registration

- ☐ Verify that the user can enter their full name.
- ☐ Verify that the user can enter their address.
- ☐ Verify that the user can enter their email.
- ☐ Verify that the user can enter a password.
- ☐ Verify that the user can click on the register button once all the details are entered.
- ☐ Verify that the user is redirected to email verification page upon successful registration.
- ☐ Verify that the user is rejected when wrong credentials are entered.

Figure 2.11: An example checklist used for Checklist-based testing at SeekMake.

At SeekMake, checklist-based testing was carried out to ensure that all test conditions were covered during the testing process. We used the insights gained from exploratory testing to create checklists for future testing activities.

2.9. Summary

This chapter presented the methodology used during the software testing process at SeekMake. It included the testing strategy, testing types, testing tools, and testing techniques used to ensure the quality of the software.

The testing process was aligned with the Scrum methodology, and various testing tools were used to manage the testing process, automate testing activities, and track defects.

The testing techniques used to design test cases included black-box testing techniques and experience-based testing techniques.

The testing activities were carried out in parallel with the development activities to ensure that the software was thoroughly tested before each release.

In the next chapters, we will present how the sprints were carried out during the internship, the testing activities performed, and the results obtained.

Chapter 3: Exploratory Testing Insights

3.1. Introduction

This chapter presents the first two sprints carried out during the internship at SeekMake. The sprints were focused on exploratory testing of the web application.

3.2. Background

At the beginning of the first sprint, the team was introduced to the SeekMake web application.

The team was given a brief overview of the application and its functionalities. The team was also provided with access to the application and the necessary documentation.

3.3. Testing Approach

The QA team used exploratory testing to test the SeekMake web application. The QA team explored the application without any pre-defined test cases or scripts. The team used their knowledge and experience to test the application and identify any issues or bugs.

3.3.1. Micro Frontends Architecture

The SeekMake web application is built using the Micro Frontends architecture.

The idea behind Micro Frontends is to think about a website or web app as a composition of features which are owned by independent teams. Each team has a distinct area of business or mission it cares about and specialises in. A team is cross functional and develops its features end-to-end, from database to user interface. [8]

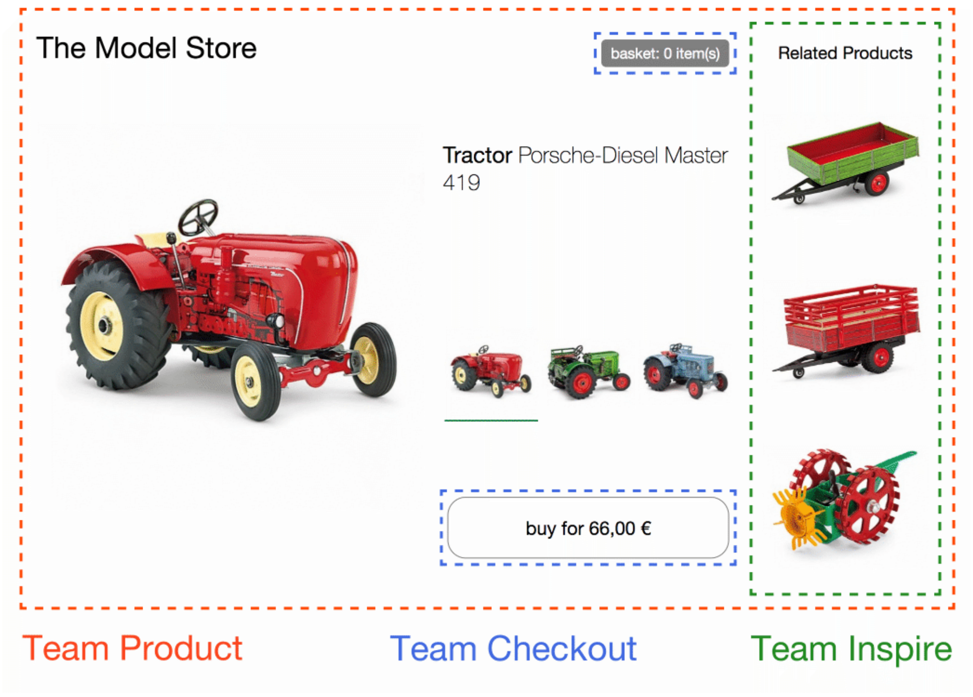


Figure 3.1: A figure demonstrating an example of Micro Frontends Architecture

This architecture allows the development team to work on different parts of the application independently, without affecting other parts of the application. It also allows the team to scale the application easily by adding new Micro Frontends.

The QA team was tasked with exploring the Micro Frontends of the application and identifying any issues or bugs. The team was also asked to provide feedback on the usability and user experience of the application.

3.3.2. Checklists

The QA team created checklists to guide the exploratory testing of the Micro Frontends, where each checklist focused on a specific Micro Frontend.

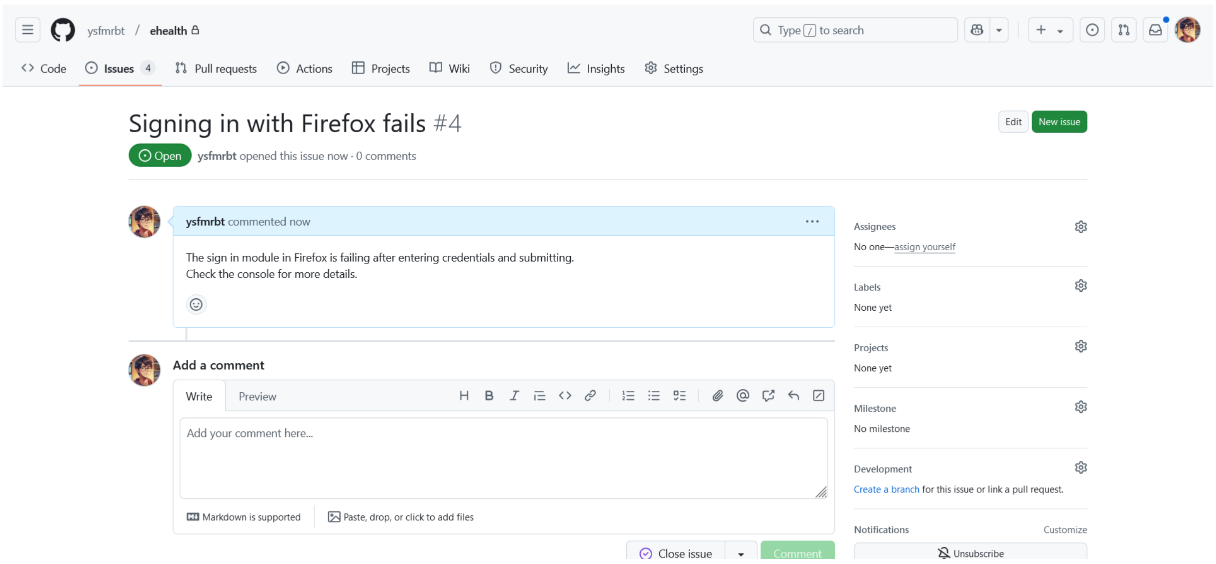
The checklists included a list of features and functionalities that needed to be tested, as well as a list of common issues and bugs that could be encountered.

GitHub Issues

The QA team used GitHub Issues to track bugs and issues identified during the exploratory testing.

The team created a new issue for each bug or issue identified, and assigned it to the relevant team member.

The team then worked together using collaboration tools like Microsoft Teams to resolve the issue and verify the fix.



Signing in with Firefox fails #4

Open ysfmrbt opened this issue now · 0 comments

ysfmrbt commented now

The sign in module in Firefox is failing after entering credentials and submitting.
Check the console for more details.

Add a comment

Add your comment here...

Markdown is supported

Paste, drop, or click to add files

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Create a branch for this issue or link a pull request.

Notifications Customize

Figure 3.2: An example of a GitHub Issue.

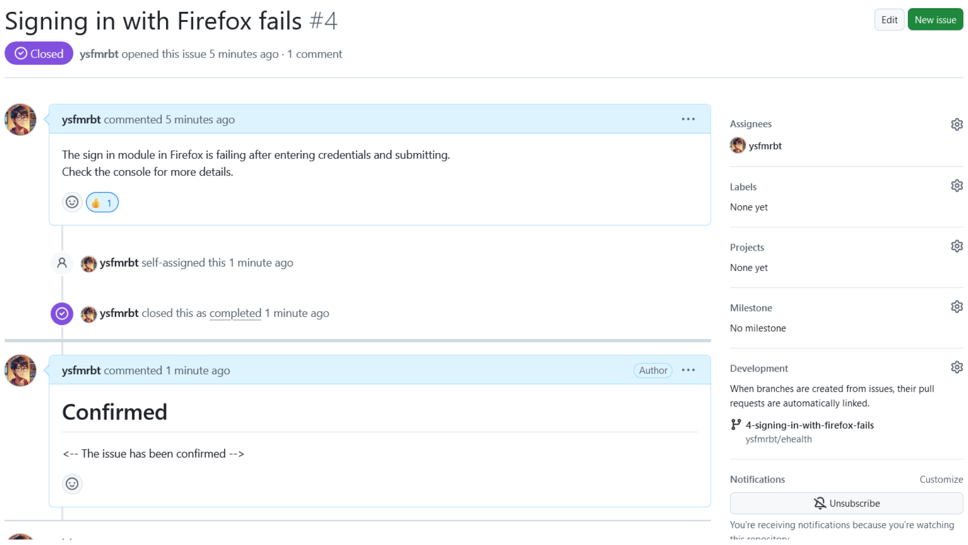
The use of GitHub Issues allowed the team to track the progress of bug fixes and ensure that all issues were resolved before the end of the sprints.

Confirmation Testing

Confirmation testing confirms that an original defect has been successfully fixed. [2]

Depending on the risk, one can test the fixed version of the software in several ways, including executing all tests that previously have failed due to the defect, or, also by adding new tests to cover any changes that were needed to fix the defect. [2]

The QA team performed confirmation testing to verify that all bugs and issues identified during the exploratory testing had been resolved.


Signing in with Firefox fails #4

Closed ysfmrbt opened this issue 5 minutes ago · 1 comment

ysfmrbt commented 5 minutes ago

The sign in module in Firefox is failing after entering credentials and submitting.
Check the console for more details.

ysfmrbt self-assigned this 1 minute ago

ysfmrbt closed this as completed 1 minute ago

ysfmrbt commented 1 minute ago

Confirmed

<-- The issue has been confirmed -->

Assignees: ysfmrbt
Labels: None yet
Projects: None yet
Milestone: No milestone
Development: When branches are created from issues, their pull requests are automatically linked.
4-signing-in-with-firefox-fails ysfmrbt/ehealth
Notifications: You're receiving notifications because you're watching

Figure 3.3: Issue confirmation.

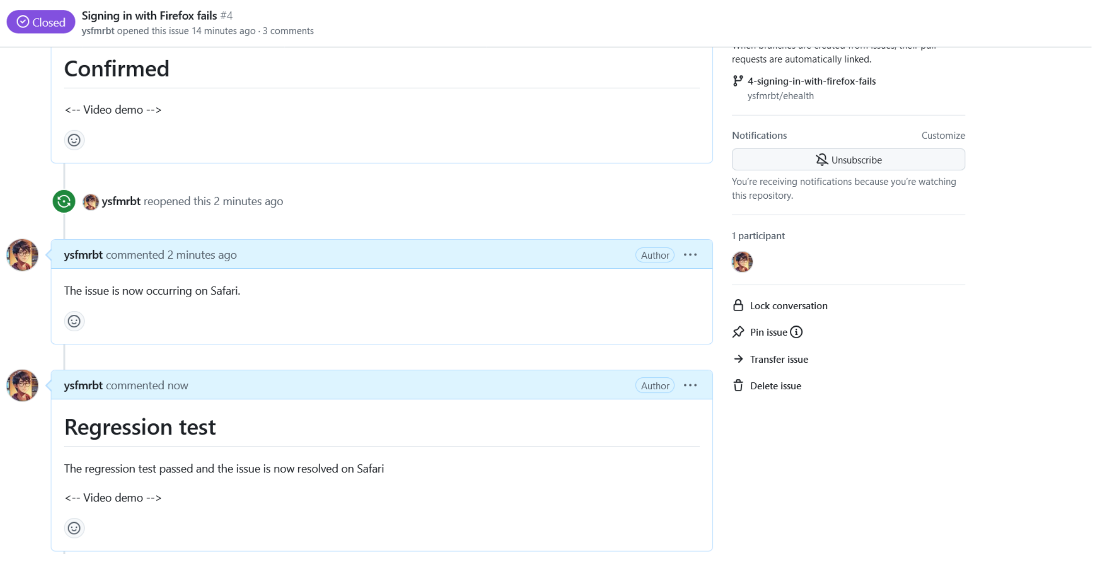
To submit a confirmation test, the QA team followed these steps:

1. Comment on the GitHub issue where the problem was reported.
2. Attach a media (image or video) demonstrating the resolution of the problem.
3. Close the issue to mark it as completed.

Manual Regression Testing

Regression testing confirms that no adverse consequences have been caused by a change, including a fix that has already been confirmation tested. These adverse consequences could affect the same component where the change was made, other components in the same system, or even other connected systems. [2]

The QA team performed manual regression testing to ensure that the bug fixes did not introduce any new issues or bugs.


Closed Signing in with Firefox fails #4
ysfmrbt opened this issue 14 minutes ago · 3 comments

Confirmed

<-- Video demo -->

ysfmrbt reopened this 2 minutes ago

ysfmrbt commented 2 minutes ago Author

The issue is now occurring on Safari.

ysfmrbt commented now Author

Regression test

The regression test passed and the issue is now resolved on Safari

<-- Video demo -->

requests are automatically linked.

4-signing-in-with-firefox-fails
ysfmrbt/ehealth

Notifications Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

1 participant

Lock conversation

Pin issue

Transfer issue

Delete issue

Figure 3.4: Regression test.

In the same way, to submit a manual regression test, the QA team followed these steps:

1. Re-comment on and open the the GitHub issue where the problem was originally confirmed as resolved.
2. Attach a media (image or video) demonstrating the resolution of the problem.
3. Close the issue to mark it as completed.

3.4. Web Application Environment

The SeekMake web application was deployed in different environments, including:

3.4.1. Production

The production environment is the live environment where the application is accessed by users. The production environment is deployed for using the application in a real-world scenario, and is monitored for performance and stability.

At the end of the sprints, the QA team deployed the application to the production environment for user acceptance testing.

The QA team used the production environment to test the application in a real-world scenario, and to identify any issues or bugs that were not present in the staging environment.

3.4.2. Staging

The staging environment is used for testing the application before it is deployed to production. The staging environment is an exact replica of the production environment, with the same configuration and data.

The QA team used the staging environment to test the application and identify any issues or bugs before it was deployed to production.

3.4.3. Development

The development environment is used by the development team to work on new features and bug fixes. The development environment is separate from the staging and production environments, and is used for testing changes before they are merged into the staging environment.

The QA team used the development environment to test new features and bug fixes before they were deployed to staging.

3.5. Testing Results

The QA team identified several bugs and issues during the exploratory testing of the SeekMake web application.

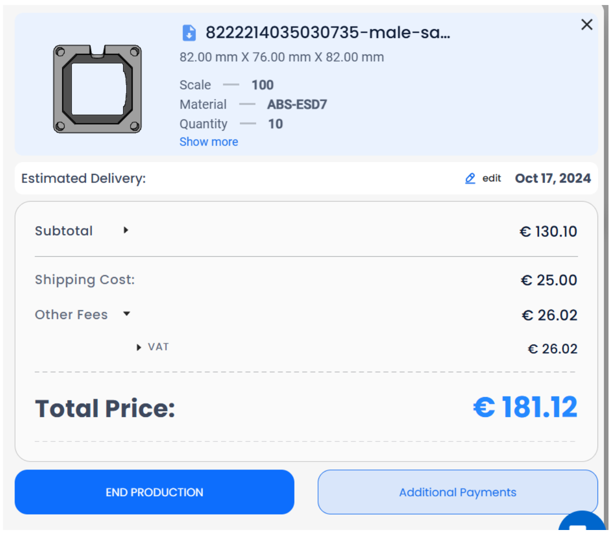
The bugs and issues were prioritised based on their severity and impact on the application and users. The team worked together to resolve the bugs and issues before the end of the sprints.

3.5.1. Bugs and Issues

The bugs and issues identified during the exploratory testing included:

- Broken links (e.g. links that do not work or lead to the wrong page) Two examples of broken links are:
  - The "Contact Us" link on the homepage redirected to the "About Us" page.
  - The "Terms and Conditions" link on the checkout page redirected to a 404 error page.

- Incorrect data (e.g. data that is missing or incorrect)


8222214035030735-male-sa...
82.00 mm X 76.00 mm X 82.00 mm
Scale — 100
Material — ABS-ESD7
Quantity — 10
Show more

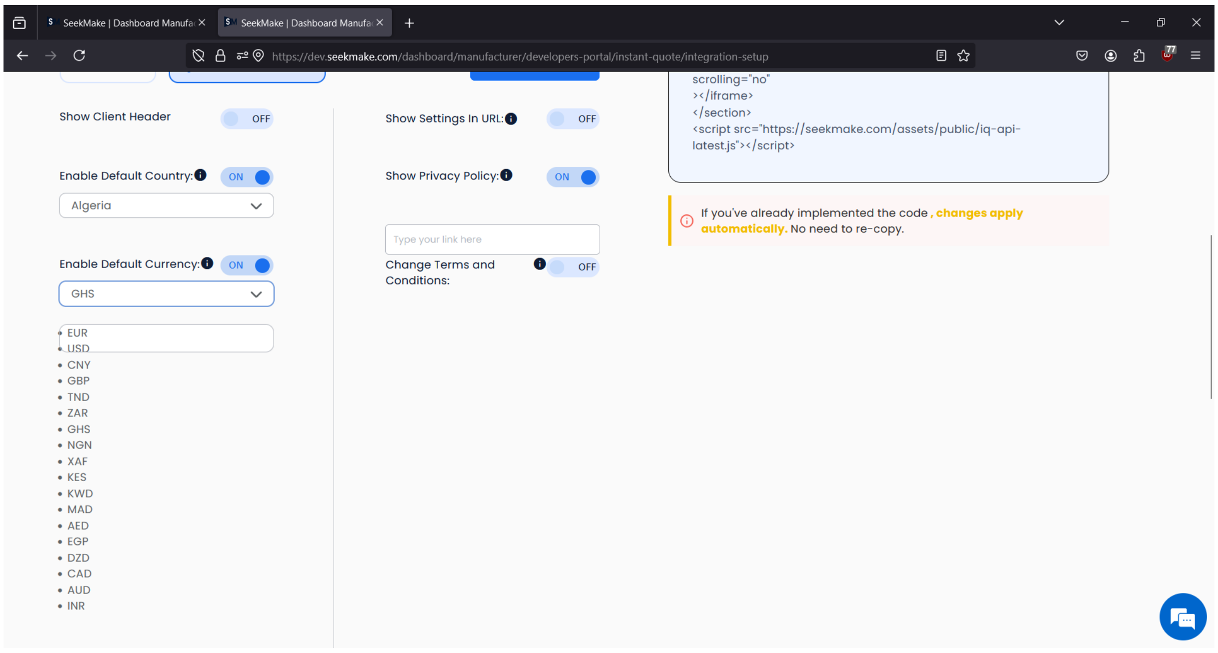
Estimated Delivery:	edit Oct 17, 2024
Subtotal	€ 130.10
Shipping Cost:	€ 25.00
Other Fees	€ 26.02
VAT	€ 26.02
Total Price:	**€ 181.12**

Figure 3.5: An example of incorrect data (Total price should be 181.02 instead of 181.12).

- Missing features (e.g. features that are not implemented or are not working as expected). An example of a missing feature that was encountered in SeekMake, is that users from certain countries were not able to complete the payment process.

- Performance issues (e.g. slow loading times or unresponsive pages). At the start of the sprint, the application took 10 seconds on average to load the homepage. After the sprint, the application took 5 seconds on average to load the homepage.

- Usability issues (e.g. poor navigation, confusing layout)


SeekMake | Dashboard Manufa
SeekMake | Dashboard Manufa
https://dev.seekmake.com/dashboard/manufacturer/developers-portal/instant-quote/integration-setup

Show Client Header OFF
Enable Default Country: ON
Algeria
Enable Default Currency: ON
GHS
- EUR
- USD
- CNY
- GBP
- TND
- ZAR
- GHS
- NGN
- XAF
- KES
- KWD
- MAD
- AED
- EGP
- DZD
- CAD
- AUD
- INR

Show Settings In URL: OFF
Show Privacy Policy: ON
Type your link here
Change Terms and Conditions: OFF

```
scrolling="no"
></iframe>
</section>
<script src="https://seekmake.com/assets/public/iq-api-latest.js"></script>
```

If you've already implemented the code , changes apply automatically. No need to re-copy.

Figure 3.6: An example of a dropdown not fully implemented.

- Security vulnerabilities (e.g. unprotected data). At the start of the sprint, a user could access the manufacturer dashboard by changing the URL. After the sprint, the manufacturer dashboard required the user to switch to a manufacturer account.
- Compatibility issues (e.g. the application does not work on certain devices or browsers). At the start of the sprint, the application did not work properly on Safari browser. After the sprint, the application worked on Safari.

The following are examples of how the bugs and issues were prioritised:

- Critical: Payment gateway failed to process certain transactions.
- Major: Navigation links redirected to incorrect pages.
- Minor: Text alignment issues on smaller screen resolutions.

3.6. Summary

The first two sprints focused on exploratory testing of the SeekMake web application. The QA team used exploratory testing to test the Micro Frontends of the application and identify bugs and issues.

The team created checklists to guide the exploratory testing, and used GitHub Issues to track bugs and issues. The team performed confirmation testing and manual regression testing to verify that all bugs and issues were resolved.

The QA team identified several bugs and issues during the exploratory testing, and worked together to resolve them before the end of the sprints.

In the next sprint, the team will focus on end-to-end testing of the SeekMake web application.

Chapter 4: End-to-End Testing and Automation

4.1. Introduction

This chapter presents the final sprint carried out during the internship at SeekMake. The sprint was focused on end-to-end testing of the web application.

4.2. E2E Testing Fundamentals

End-to-end testing, also known as E2E testing, is an approach to testing that that simulates real user experiences to validate the complete system. [4]

4.2.1. Goals

The primary goal of E2E testing is to ensure that the application behaves as expected from the user's perspective. E2E testing is used to validate the functionality, usability, and performance of the application.

4.2.2. Benefits

E2E testing offers several benefits, including:

- Validates the complete system
- Identifies issues that cannot be found with unit or integration testing
- Provides confidence in the application's quality
- Reduces the risk of bugs and issues in production

4.3. E2E Testing Strategy at SeekMake

The E2E testing strategy at SeekMake was designed to validate the functionality, usability, and performance of the web application. The strategy included the following components:

4.3.1. Test Environment Setup

The QA team set up test environments to test the application in different scenarios, including production, staging, and development environments.

4.3.2. Test Planning Approach

The QA team used a risk-based approach to test planning, focusing on high-risk areas of the application. The team created test plans and test cases to guide the testing process.

4.3.3. Test Case Design Methodology

The QA team used a combination of manual and automated test cases to validate the application. The team created test cases based on user stories and acceptance criteria.

4.4. Implementation with Playwright

The QA team used Playwright, an open-source testing framework, to automate the E2E testing of the web application. Playwright provides a simple and powerful API for automating web browsers, and supports multiple programming languages, including JavaScript, Python, and C#.

4.4.1. Framework Setup

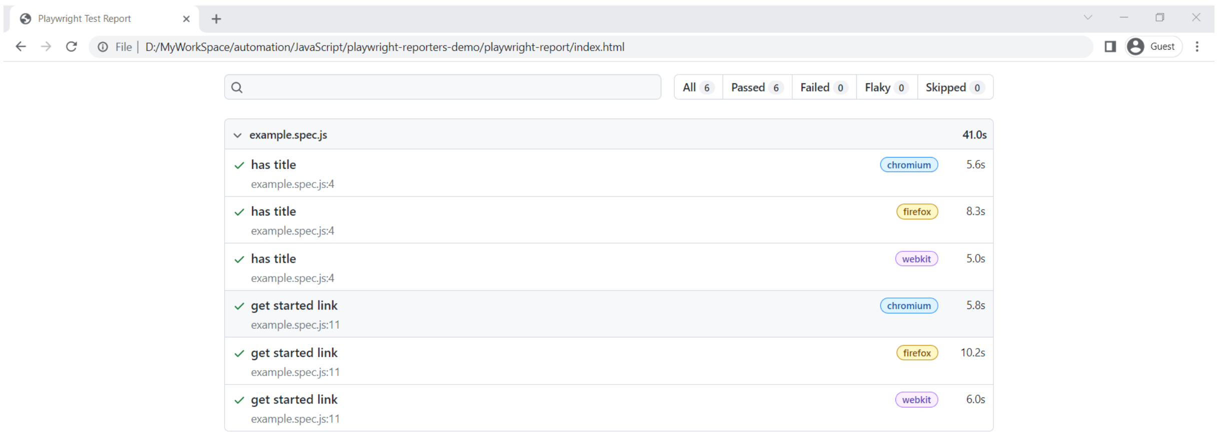
The QA team set up the Playwright framework with Node.js to automate the E2E testing of the web application. The framework included test scripts, test data, and test reports.

4.4.2. Test Script Development

The QA team developed test scripts to validate the functionality, usability, and performance of the web application. The team used Playwright's API to interact with the web application and verify the expected results.

4.4.3. Test Execution

The QA team executed the test scripts in different environments, including production, staging, and development environments. The team monitored the test results and reported any issues or bugs in a weekly report.


Playwright Test Report
File | D:/MyWorkSpace/automation/JavaScript/playwright-reporters-demo/playwright-report/index.html
Guest
All 6
Passed 6
Failed 0
Flaky 0
Skipped 0

example.spec.js		41.0s
✓ has title example.spec.js:4	chromium	5.6s
✓ has title example.spec.js:4	firefox	8.3s
✓ has title example.spec.js:4	webkit	5.0s
✓ get started link example.spec.js:11	chromium	5.8s
✓ get started link example.spec.js:11	firefox	10.2s
✓ get started link example.spec.js:11	webkit	6.0s

Figure 4.1: An example of a Playwright report

4.5. Results and Analysis

End-to-end testing focused on validating critical user workflows, including account creation, login, and payment processing. Cross-browser tests conducted via BrowserStack revealed rendering inconsistencies in Safari, which were resolved through CSS optimizations. Lighthouse metrics showed significant improvements in performance, with page load times reduced by 30% on average.

4.5.1. Test Coverage

The QA team tested all critical user journeys, such as account creation, login, and payment processing, ensuring these workflows functioned without issues across various devices and browsers.

4.5.2. Cross-Browser Testing

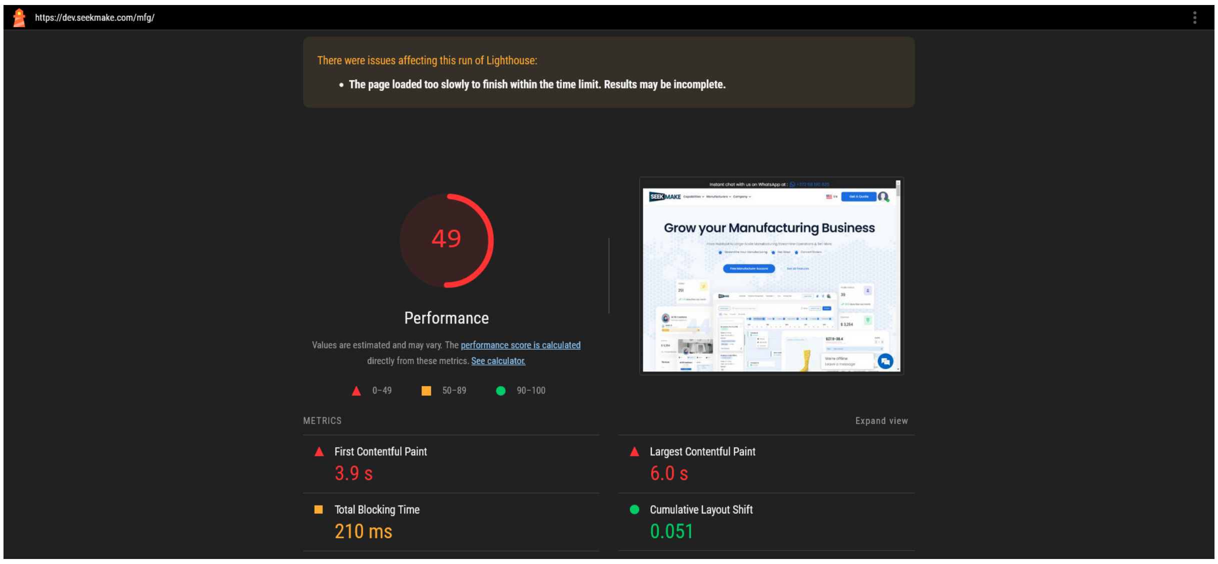
The QA team performed cross-browser testing to validate the application on different browsers, including Chrome and Firefox. The team identified browser-specific issues and reported them for resolution.

The team used BrowserStack Live to run the tests on Apple's Safari browser on different devices.

4.5.3. Performance Considerations

The QA team monitored the performance of the web application during the E2E testing. The team identified performance issues, such as slow loading times and unresponsive pages, and reported them for resolution.

The team used Lighthouse to measure the performance of the web application and identify opportunities for improvement.


https://dev.seekmake.com/mfg/

There were issues affecting this run of Lighthouse:

- **The page loaded too slowly to finish within the time limit. Results may be incomplete.**

49

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS

Expand view

▲ First Contentful Paint
3.9 s

▲ Largest Contentful Paint
6.0 s

■ Total Blocking Time
210 ms

● Cumulative Layout Shift
0.051

Figure 4.2: An example of a Lighthouse report

4.6. CI/CD Integration

The QA team integrated the E2E testing framework with the CI/CD pipeline to automate the testing process. The team used GitHub Actions to trigger the test execution on every code change.

4.6.1. GitHub Actions Workflow

The QA team created a GitHub Actions workflow to run the E2E tests on every pull request.

The workflow included steps to install dependencies, build the application, and run the test scripts.

4.6.2. Test Automation Pipeline

The QA team set up a test automation pipeline to automate the E2E testing of the web application.

The pipeline included stages for test execution, test reporting, and test result analysis.

4.7. Summary

The final sprint focused on end-to-end testing of the SeekMake web application. The QA team used Playwright to automate the testing process and validate the functionality, usability, and performance of the application.

The team identified several issues and bugs during the E2E testing, and reported them for resolution.

The team integrated the E2E testing framework with the CI/CD pipeline to automate the testing process. The team used GitHub Actions to trigger the test execution on every code change.

Summary

To summarize, this report has presented the results of an internship at SeekMake, a software development company that provides digital transformation services to businesses and customers. The internship focused on software testing, with an emphasis on end-to-end testing of the web application developed by SeekMake.

The initial sprints prioritized exploratory testing to uncover functional and usability issues in the web application's Micro Frontends architecture. The insights gained were instrumental in refining test cases for end-to-end testing during the final sprint. This iterative approach ensured that all critical workflows were thoroughly validated before deployment.

This internship was transformative, providing me with hands-on experience in test automation using Playwright and continuous integration through GitHub Actions. Collaborating with a cross-functional Agile team enhanced my communication and problem-solving skills, preparing me for future roles in software quality assurance. Additionally, the challenges of debugging cross-browser issues deepened my technical expertise in compatibility testing.

Bibliography

[1] Atlassian. https://www.atlassian.com/continuous-delivery/continuous-integration, 2024. [Online; accessed 1 December 2024].

[2] International Software Testing Qualifications Board. *ISTQB Certified Tester Foundation Level Syllabus v4.0*. International Software Testing Qualifications Board, 2024.

[3] BrowserStack. https://www.browserstack.com/, 2024. [Online; accessed 27 November 2024].

[4] CircleCI. https://circleci.com/blog/what-is-end-to-end-testing/, 2024. [Online; accessed 1 December 2024].

[5] GitHub. https://docs.github.com/en/get-started/start-your-journey/about-github-and-git, 2024. [Online; accessed 27 November 2024].

[6] GitHub Actions. https://docs.github.com/en/actions/about-github-actions/understanding-github-actions, 2024. [Online; accessed 27 November 2024].

[7] GitHub Projects. https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects, 2024. [Online; accessed 27 November 2024].

[8] Micro Frontends - extending the microservice idea to frontend development. https://micro-frontends.org/, 2024. [Online; accessed 1 December 2024].

[9] Microsoft Teams. https://www.microsoft.com/en-us/microsoft-teams/small-medium-business, 2024. [Online; accessed 27 November 2024].

[10] Installation | Playwright. https://playwright.dev/docs/intro, 2024. [Online; accessed 27 November 2024].

[11] Playwright | Homepage. https://playwright.dev/, 2024. [Online; accessed 27 November 2024].

[12] Scrum. https://www.scrum.org/learning-series/what-is-scrum/, 2024. [Online; accessed 1 December 2024].

[13] SeekMake | Features. https://seekmake.com/features/, 2024. [Online; accessed 27 November 2024].