TDT Project Documentation

Stanciu Alin, Pricop Laurentiu

Academic Year 2023-2024, Spring Semester

Contents

1 Application Details. Investigated features

The application we are testing is called Physical Mail Manager. The application is available for fiddling around with publicly [2], and its code is available on GitHub [3]. We will refer to the app as "PMail" from now on.

PMail intends to provide a means of keeping track of physical mail (alternatively known as snail mail) that Laurentiu personally was sending and receiving. It allows configuring destinations and can assign to each letter a unique trackable code.

PMail provides the following features:

1. Authentication based on username and password, along with registration.
2. Maintain a list of receivers, along with information required on the envelope (address, full name, postal code) and other optional user-defined information.
3. Store a list of letters. Each letter is attached to a receivers (which keeps the same name even if they are actually a sender), can store metadata such as letter unique code, price, and other optional user-defined information.

In particular, we tested the following aspects of PMail:

- Authentication (login/logout with password);
- Adding receivers;
- Adding letters;
- Grouping of letters based on receiver and type (sent/received).

2 AC. IOs

We are employees at a software agency, specializing in web software, in particular business-to-business solutions customized for each customer. The company lead has a middle-school child which came up with the idea of an application to manage snail mail they were constantly sending to their friends. Having some leftover allocation in the company, and being in the middle of a transition from Angular to React company-wide, the company lead figured

that this idea is worth investigating further as a learning project for his engineers.

However, rather surprisingly, a specific market demand was identified for the application after development was complete. It seems the company's business partners can apply the software as an internal solution to manage mail going in and out of their headquarters.

The company lead wants to capitulate on this newfound demand, but the application is currently in an unfit state. As such, our company decided to expand the development team on the so-called PMail project and bring in some new roles. In particular, we, Laurentiu and Alin, were hired as QA engineers, which now have the task of identifying an initial round of bugs and issues found with the product after its initial, internal, launch.

Being an in-house and a training project, bugs are expected and the company lead insists on listing as many of them as possible. Additionally, the user interface and experience was an after-thought during development of PMail, so the company lead also wants to identify specific elements and flows of the application that might be a bit harder to grasp for a person unfamiliar with it, in the form of user experience hiccups.

3 Testing Mission

The initiative's goal is judging the app's fitness for releasing as a product of our company, and creating an improved version that can level up from the status of "learning project" to that of "marketable product". As QA engineers, we can only help with the first part.

We need to perform an initial assessment of the application, drawing up test cases around usual user behavior and also potential edge situations, while also investigating common user experience anti-patterns.

4 Testing Strategy

The strategy is fairly straightforward, and is broken up in two parts.

1. API Black-Box Testing

   We are going to write integration tests for two functionalities of the application, that call APIs and test for proper responses and error messages.

We limit ourselves to the API initially since we consider it to be the backbone of any web application, thus being of significant importance testing-wise.

These tests try to uncover an assortment of types of bugs that relate to, but are not limited to, the following aspects: database connection, syntactically invalid data, semantically invalid data, incorrect references.

2. User Experience: Exploratory Testing

   We are going to fiddle with the application's frontend, manually and in an exploratory manner, to try to uncover as many design anti-patterns as we can.

   We're going to make a list of behaviors that we identify as potential issues and hand it over to the company lead, which will decide what, if any, corrective actions are going to be taken for each of them.

5 Selected Test Design Techniques

Part	Test Strategy	Test Design Technique	Dimension Covered	Students and Features
Part I	Process-compliant	Beta Testing	Coverage	Laurentiu Pricop (Authentication)
Part I	Reactive	Boundary Testing	Risk	Alin Stanciu (Adding letters)
Part II	Reactive	User Interface Testing	Risk	Laurentiu Pricop (Authentication)
Part II	Reactive	User Interface Testing	Coverage	Alin Stanciu (Adding destinations)

6 Test Design. Test implementation. Test execution. Test Report

6.1 Test design

Student	Feature	Test Design Technique	Details	Input, Expected Output
Laurentiu Pricop	Authentication	Beta Testing	Can a normal user tinkering with the app with no intention to break it, break it?	**In:** create account **ExpOut:** account created **In:** login with valid credentials **ExpOut:** logged in **In:** login with invalid credentials **ExpOut:** error
Alin Stanciu	Adding letters	Boundary Testing	Is there a limit on the size of the address? Can letters be added in numerical fields? Is Unicode supported? Are SQL injections possible?	**In:** empty address **ExpOut:** error **In:** address very long **ExpOut:** error **In:** unicode text **ExpOut:** properly displayed **In:** SQL-like text **ExpOut:** properly displayed **In:** HTML-like text **ExpOut:** properly displayed

Student	Feature	Test Design Technique	Details	Input, Expected Output
Laurentiu Pricop	Authentication	User Interface Testing	Are buttons readable and clear? Are CTA texts easily understandable? Are buttons responsive?	No explicit test cases can be drawn out.
Alin Stanciu	Adding receivers	User Interface Testing	Are buttons readable and clear? Are CTA texts easily understandable? Are buttons responsive?	No explicit test cases can be drawn out.

6.2 Test implementation. Test execution

For our Beta and Boundary Testing, we devised a series of Postman integration tests which test for a variety of cases similar to the ones detailed in the section above. You can find the Postman collection on the project repository [4].

For our User Interface testing, we devised a series of test cases using the platform Testiny, which we then manually executed against the tested features. You can find the collection of test cases in an Excel sheet in the project repository [7].

6.3 Test reports

After running our devised tests, our tools (Postman and Testiny) can generate a series of test run reports. Screenshots of these reports are presented in Figures 1, 2, 3, 4.

Full testing reports for Testiny are available in the TDT Project repository [6] [5], while the Postman collection with the written tests is available in the same place [4].

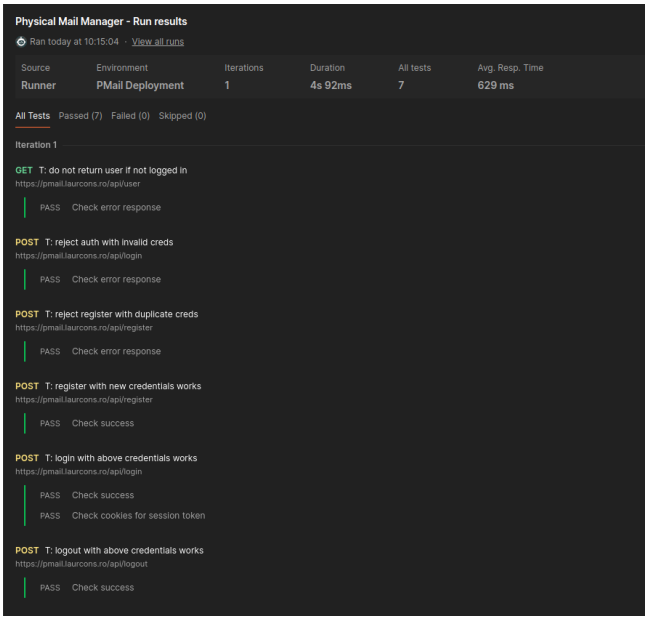

Figure 1: Screenshot of run results for the Authentication feature, after running Beta Tests in Postman.

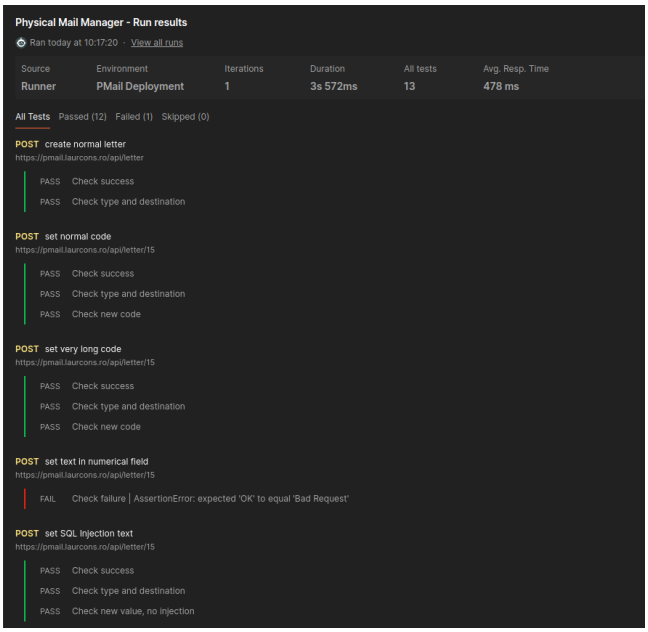

Figure 2: Screenshot of run results for the Add Letter feature, featuring Boundary Testing with certain edge cases tested.

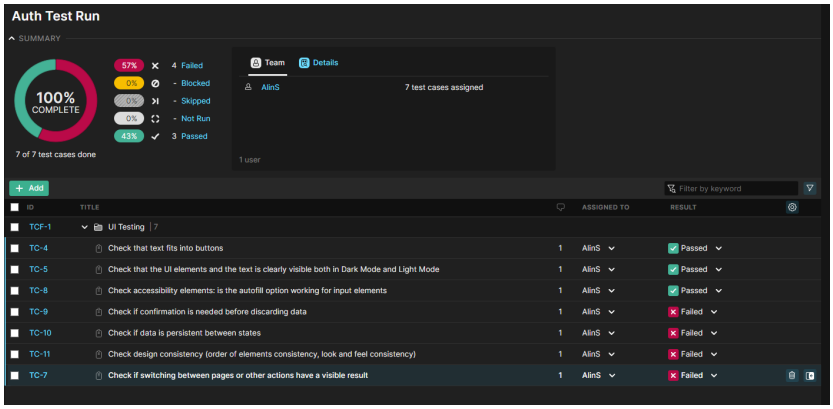

Figure 3: Screenshot of test run results for the Authentication feature, after manually running User Interface tests using Testiny.

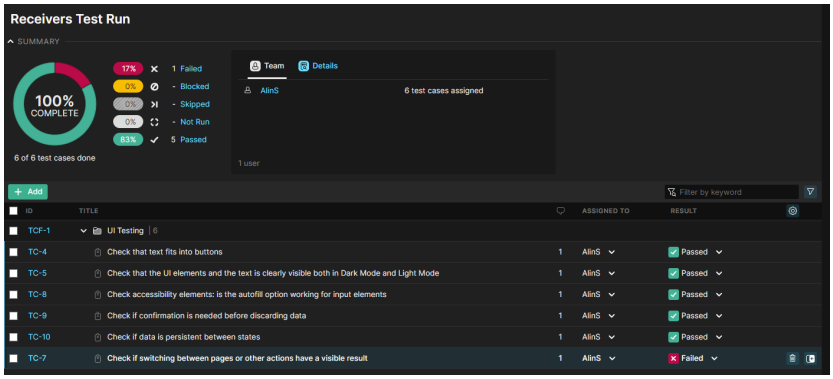

Figure 4: Screenshot of test run results for the Add Receivers feature, after manually running User Interface tests using Testiny.

7 Issue Reporting

We compiled a RIMGEA breakdown of an User Interface issue, which is available in the project repository [1].

8 Conclusions. Lessons Learned

Having run thorough tests of the application, using a variety of methods, techniques and scopes, we can characterize the application in the following way:

1. The API is stable and no noticeable functionality bugs have been noticed. The only thing that can be improved is the acceptance of data types, since fields that should numeric accept any kind of strings.

2. The UI is stable and functional, although little attention was paid to the user experience. A number of issues were identified across the application, which range from confusing movement of elements, to loss of data when navigating from one screen to another.

The QA team therefore concludes that the product is NOT yet ready for launch to business-to-business clients, and needs further polishing before a release is to take place.

References

[1] Alin Stanciu, Pricop Laurentiu. RIMGEA Issue Report. https://github.com/Laurcons/cs-tdt3-proj-docs/blob/master/documents/IssueReport.docx.

[2] Pricop Laurentiu. Physical Mail Manager application. https://pmail.laurcons.ro.

[3] Pricop Laurentiu. Physical Mail Manager repository. https://github.com/Laurcons/PhysicalMailManager.

[4] Pricop Laurentiu, Alin Stanciu. Postman Collection with our automated testing. https://github.com/Laurcons/cs-tdt3-proj-docs/blob/master/documents/PostmanCollection.json.

[5] Pricop Laurentiu, Alin Stanciu. Testiny manual testing report for Adding Receivers. https://github.com/Laurcons/cs-tdt3-proj-docs/blob/master/documents/Testiny%20%E2%80%93%20Test%20runs%20-%20Receivers.pdf.

[6] Pricop Laurentiu, Alin Stanciu. Testiny manual testing report for Authentication. https://github.com/Laurcons/cs-tdt3-proj-docs/blob/master/documents/Testiny%20%E2%80%93%20Test%20runs%20-%20Auth.pdf.

[7] Pricop Laurentiu, Alin Stanciu. The collection of test cases that were manually tested using Testiny. https://github.com/Laurcons/cs-tdt3-proj-docs/blob/master/documents/Testiny-export-testcases-TDP.xlsx.