TEST PLAN

PalletPals – Phase 1

Lecturers	Lukas Frey Prof. Bradley Richards
Authors	Tibor Haller Marco Kaufmann Daniel Locher
Place and date	Brugg-Windisch, 02.06.2022

Introduction

This test plan describes how the software solution PalletPals is tested. Specifically, what, how, when, and by whom the product is tested. Figure 1 summarizes the testing plan for different events of the development lifecycle.

what		when	who	where
tier[1]	automated	development of a feature	tier[1]	working device
all	automated	finished developing a feature	tier	test environment
all	automated manual	deployment	all	test environment
all	automated manual	final deployment	all + outside[2]	test environment

[1]Only tests for the tier that was worked on are performed and only by the developer(s) that work on this tier.
[2]Testers from outside the team are only performing manual testing.

Figure 1: Overview of test plan

Test Strategy

The test strategy defines what has to be tested (scope), how it is tested (type), and by whom and when it is tested (logistics).

Scope of Testing

Features to be tested

All use cases defined in the requirements are needed to be tested. This has to be ensured in both the backend and the frontend.

API: To ensure that the API fulfils the requirements, all API endpoints are tested by using Postman (https://postman.com). Each endpoint has to accept the defined methods and payload and return the according status codes and content.

Frontend: To ensure that the frontend fulfils the requirements, all use cases are tested with unit and/or integrated tests using a testing framework.

Features not to be tested

- Database Integrity
- Performance
- Hardware
- Dependencies

Testing Type

API

- Unit testing of backend functionalities for all relevant units
- Testing of all API requests using Postman to verify correct behaviours
- Integrated testing to verify correct webapp behaviour throughout the interaction of different components

Frontend

- Unit testing. For all relevant units, unit tests are written that verify the output with given input.
- Integrated testing. For selected parts of the frontend, integration tests are written to test the output of a combination of units with given input.

Both

- System testing: Before each deployment, system testing is performed. All use cases are manually tested to ensure expected behaviour. This is done on at least two different device types.

Test Logistics

Who is testing?

The backend tests are performed by the backend developer(s).

The frontend tests are performed by the frontend developer(s).

Before the final deployment, all tests are performed by all team members and additional third-party testers.

When is testing performed?

For every feature, the according test shall be created before the development of this particular feature has started. This ensures that the target that has to be developed is clear. During the development, the test can be used to verify the behaviour of the feature. Once a feature is finished, all tests for this tier (backend/frontend) are performed to ensure other tests were not affected.

Additionally, before each deployment, all tests are performed as well.

Test Objectives

The test objectives are…

- …**verifying** the **functionality** of the final product.
- …making sure **no broken code** is deployed.
- …finding and **fixing bugs** efficiently.

Test Criteria

A test phase is successful when…

- …**100%** (of the tier) of the tests are successful.
- …for every feature exists **at least 1** test that was performed.

During the development of a single feature, performing only the corresponding testing is acceptable. However, once a feature is finished, all tests are needed to be performed and fulfil the testing criteria.

Docker Test Environment in the Frontend

The test environment ensures a **consistent** and **reliable** environment for testing to eliminate side-effects that could cause unexpected behaviour.

This environment is defined as a Docker container in a Dockerfile. Docker (https://www.docker.com/) is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. It can simply be started from any computer where Docker is installed while staying consistent. The API tests are defined in Postman and organised in a collection and folders. This collection can be run at once and provides an overview over all tests. In Postman different environments can be defined – one is defined for testing and will be used to test the API endpoints. The frontend tests are defined using a testing framework and a command is setup to run all tests at once.

Automated Testing using GitHub Actions

During the frontend development, GitHub Actions (https://github.com/features/actions) was used to automate the testing for submitted pull requests and push actions to the main branch. As seen with the palletpals-client front development (https://github.com/mahgoh/palletpals-client/actions), the automated workflow makes it simple to verify code validity when new features are added.