Development of a dashboard as part of PinDown’s new graphical user interface

Mai Linh Nguyen & Erik Samuelsson

Conclusion

Our conclusion is that the design met Verifyter’s needs and that the dashboard should improve the usability of PinDown. After we had complete the work, we were able to answer our questions.

What are the pros and cons with the current implementation in terms of usability?
The most obvious drawback with the current implementation is that it is not a dashboard. With a dashboard you collect all the necessary data in one area, thus removing the need to switch to different views to complete tasks. When evaluating the current implementation against five factors of usability it is important to remember that these factors are balanced off each other. That means that while you should consider all of them, sometimes you have to make compromises. We found that current implementation is easy to learn, easy to remember and easy to understand. This comes with the cost of being less task efficient and as consequence of this also less subjectively satisfying.

What web frameworks are of interest, and how are they different from each?
The frameworks that we researched in this thesis were Ruby on Rails, Django and Play framework. We found that either of them could be used in our case. Our decision to use Play as our web framework was based on our knowledge of the Java language, but also because the current implementation of PinDown is Java based, which would mean an easier transition. The main difference between the frameworks really is the language used. They all follow the same pattern of an MVC architecture, however it is slightly different in Django.

What needs does Verifyter have when it comes to design and functionality of the new GUI?
The previous two questions progressed into this one and as a result we concluded that Verifyter’s needs of the dashboard were:

- Backward compatibility.
- A design that is responsive, accessible and easy to learn and use.
- The functionality of the dashboard should improve task efficiency by removing some of the usability issues in the current design.

How should the project dashboard be designed?
To answer the question we worked with two different types of mockups, paper and digital, and then we conducted a set of user tests to evaluate the design.
The user tests aimed towards usability and we learnt a lot from them. However, we think that most of the usability problems found by our test subjects either had to do with bad code, or quality of life changes. What we mean with quality of life changes in this context is that the problems found didn’t force us to rework the design, but rather improve on it. There is difference there which we think is worth pointing out. A rework would e.g. mean to move components to new locations, while a quality of life change means to add or remove something from existing components. The latter often being easier to fix. An example of bad code was the error messages which only displayed when an error occurred and both of the text fields in “Create project” were filled in.

Problem

The thesis consisted of two problems. The first one was to design a new project dashboard and the second was to implement this design as part of PinDowns new graphical user interface. For the second problem we chose frameworks for web application development to work with.

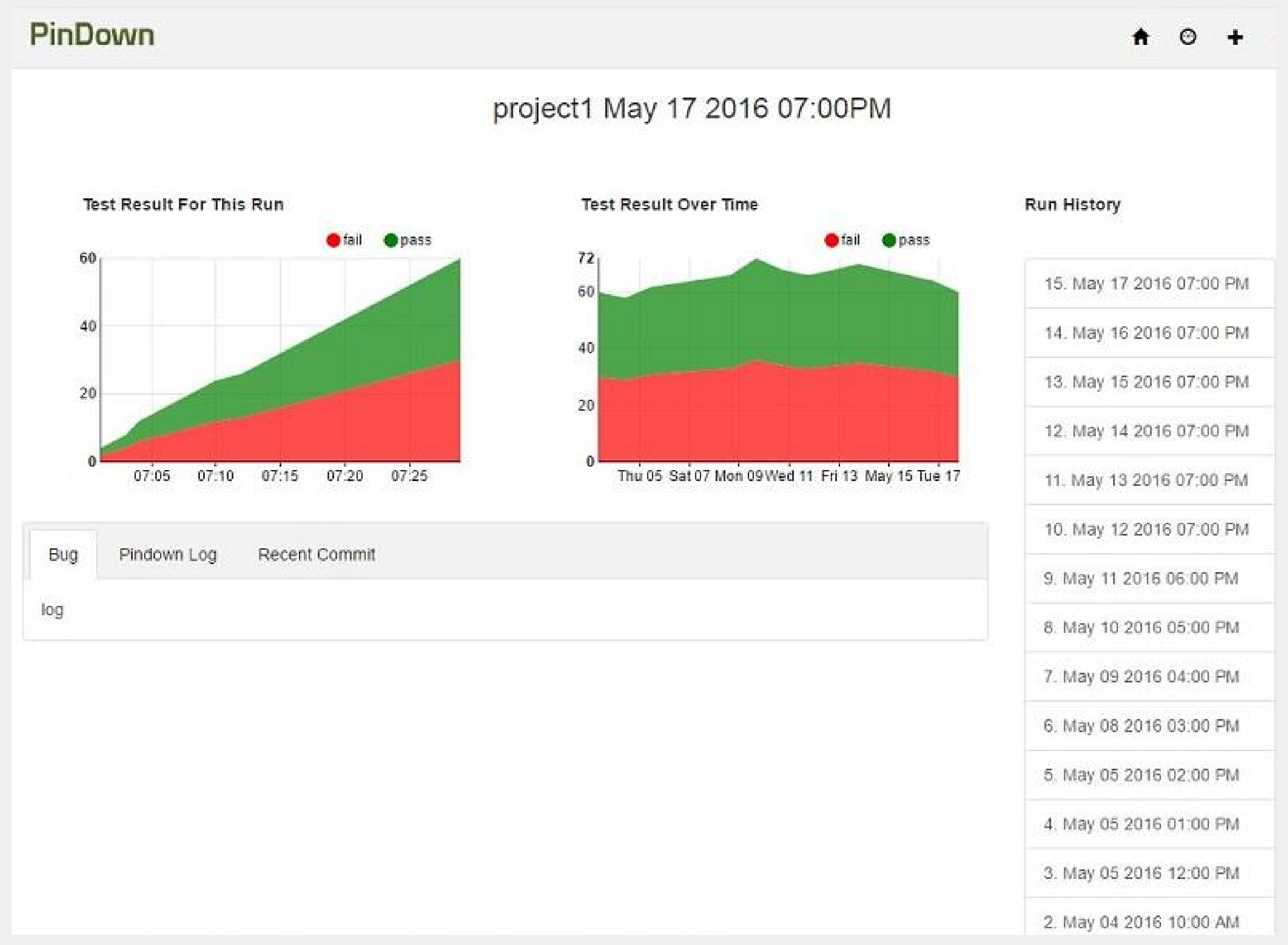

Figure 1: The dashboard.

Introduction

This thesis was commissioned by Verifyter AB, a software startup company which has developed a verification tool called PinDown that automatically debugs test failures. The goal of this thesis was to design a dashboard for one of the features of the graphical user interface. Based on a set of criteria we looked into which frameworks would be good options for implementing the design and for future development of the web application.

With a new GUI Verifyter is looking to address some issues as well as to update the overall look and feel.

Approach

For question 1 and question 2 we made analysis based on different sets of criteria. For question 1 we used Lauesen’s (2002) definition of usability as our criteria. Lauesen’s definition is made of five factors; ease of learning, task efficiency, ease of remembering, subjective satisfaction, understandability. For question 2 we had two sets of criteria, one set for front end libraries and the other for back end frameworks. These criteria were formed after discussion with Verifyter. The front end criteria were; backward compatibility, responsiveness, offline JavaScript libraries (hosted locally). And the back end criteria were; backward compatible with JVM, compile to WAR, able to obfuscate, server-side thread-safe.

For question 3 and question 4 we followed a process of based on *Dashboard development guide* (Staron 2015). Staron describe five steps in the process of developing a dashboard; requirement elicitation, dashboard type selection, dashboard design, impact evaluation, dashboard maintenance. For this thesis we found that the first three steps were of particular interest, as they are within our scope.

Result

Figure 1 shows the dashboard design. In order to evaluate how well our design worked and to test our theories and design principles against the five factors of usability we conducted a usability test which revealed some usability issues that we didn’t recognize before.

Evaluation

The evaluation was done through user tests, completed by three students in computer science. We led them through a total of six task-based scenarios and then asked them a few questions about the overall design and their thoughts of it. The tasks were written after establishing the following test goals based on Lauesen’s definition of usability:

- Ease of learning: can a user without training complete the various tasks?
- Task efficiency: how long does it take to complete a task, is it within a reasonable timeframe?
- Subjective satisfaction: was the experience satisfying or enjoyable?
- Understandability: is it hard for the users to grasp the context of the tasks they are preforming?

We excluded one of the factors, ease of remembering, from our goals. The reason for this was because PinDown is a dedicated tool, and it is no likely to have occasional users. Even if there are occasional users, we would argue that they would be few enough for it to not be worthwhile testing.

The evaluation led to some improvement of the functionalitiy, and fixes to a few errors that we had missed, but none of the feedback led to redesign.

References

Lauesen, S. (2002). *Software Requirements – Styles and Techniques.* Addison-Wesley, Harlow.

Staron, M. (2015). *Dashboard Development Guide – How to build sustainable and useful dashboards to support software development and maintenance.* Research Reports in Software Engineering and Management 2015:02 ISSN 1654-4870 Chalmers University of technology and University of Gothenburg, Department of Computer Science and Engineering.