FitchFork-2 Code Assessment System (version 0.1)

Fritz Solms, Vreda Pieterse, Linda Marshall,
Neels van Rooyen, Danny Pretorius & Hannes Jansen van Vuuren
Dept Computer Science, University of Pretoria

May 7, 2015

Contents

Chapter 1

Project context

Fitchfork is a central pedagogical and assessment system of the Department of Computer Science which has, over the years, significantly reduced the manual labour around code assessment and has provided real-time feedback on developed code to students.

1.1 What is FitchFork?

FitchFork is a code analysis and assessment tool which is meant to be used to automate aspects of

- analyzing code for functional correctness and a range of qualities,
- assigning marks based on the above analysis, and
- providing pedagogically valuable (potentially real-time) feedback to students.

1.2 History of FitchFork

[Fritz: Somebody who knows this, please wite this . . .]

1.3 Short-comings of current system

[Fritz: Please complete listing

- usability issues,
- missing or incorrect functionality,
- quality issues (e.g. reliability, scalability, performance, security, . . .),
- technical concerns,
- . . .

]

1.4 Alternatives

[Fritz: In this section let us look at the alternatives to Fitchfork, highlighting

- their strengths,
- their weaknesses,
- why they are not suitable for us, and
- what ideas we should take from them.

]

1.5 Project sponsor

The project is sponsored by the *Department of Computer Science* at the *University of Pretoria*.

1.6 Team

[Fritz: Here I thought to include the contact details of all team members as well as a description of their responsibilities within this project.]

Chapter 2

Architecture requirements

[Fritz: This section contains the quality requirements as well as any technical requirements around deployment environments, integration and access cahnnels.]

Chapter 3

Architecture design

[Fritz: This section specifies the software architecture addressing the non-functional (technical) requirements within which the application functionality addressing functional requirements is to be developed, deployed and executed. This is a very technical document.]

Chapter 4

Application Requirements and Design

[Fritz: This section contains the functional requirements and the application design which addresses the functional requirements. This is a document which will be incrementally refined as functionality (use-cases) is added to the system, facilitating an agile development process once the architecture has been put in place. The application design is preferably architecture and technology-neutral.]

Chapter 5

Notes on implementation mappings

[Fritz: The code should be a direct mapping of the application design onto the architecture and technologies as specified by the architecture design. As such the code should be largely self-documenting. However, if there are any tricky aspects around the implementation mapping which you would like to document outside the code, it can be done in this section.]

Chapter 6

Installation and configuration manual

[Fritz: Write section]

Chapter 7

User's manual

[Fritz: Write section]