Attention:
HOA DAM
CSCI311
7 April 2014

Software Plan (Detail)

Table of Contents

Version History

Date	Version	Comment	Author
6th April	0.1	First draft	James Wilson
7th April	0.2		
8th April			

Introduction

This report details the implementation plan of Team Awesome's new software product called PyTag. This software will allow for the developers and contributors to search the Python code base, issue database and mailing lists from a central Dashboard.

Abbreviations

TODO: Insert document-specific abbreviations.

References

TODO: Insert documents that this report references.

1. Scope

1.1. System Overview

This system will be comprised of two major components - client side and server side - with several modules making up the server-side component.

4. General Software Activities

4.1. Software Development Process

The chosen development method for this product will be the ‘feature-driven’ method, a derivative of the Agile development method. This method is an iterative-based approach to the development of the client software, whereby each feature is designed, planned, tested and then integrated into the main program. Each cycle is an ‘iteration’ and is detailed in the Gantt chart.

4.2. General Plans for Software Development

4.2.1. Software Development Methods

Feature-driven development requires extensive testing of all implementations of the various features when being imported into the main code base. The tools that will be used in this process include:

- **Git** - the use of a Git repository to share code amongst the team, so that all members can modify files individually and then commit them so others may test their changes
- **Travis-CI -** a continuous building and testing system Travis-CI will allow for each commit of a feature (during an iteration) to be tested against a set of pre-defined units. This will mean commits that fail to meet a build status of ‘pass’ will not be allowed in the main branch.
- …

4.2.2. Standards for Software Products

The final deliverable - the ‘PyTag’ software - is comprised of two parts: client and server.

The client (or end-user) component will be made to conform to HTML5, JavaScript and CSS3 standards (commonly known as ‘web’ standards). This will ensure maximum cross-platform compatibility without a loss in function.

As the client has not specified a desired language for the development and running of the software, Ruby on Rails has been chosen by Team Awesome as the platform that will power the server-side component of the software.

To ensure that final software product is delivered on time, and with the features requested from the client, a standard will be applied to all code submitted. All source files will be documented so that future development may occur with minimal cost and training / down time.

4.2.3. Reusable Software Products

The application shall take full advantage of existing solutions (known as ‘Ruby Gems’) publicly available for commercial use if such a solution is found to solve a minor or complex part of the software.

A minimum of one (1) day will be spent on testing and attempting to implement Ruby Gems within the existing codebase.

4.2.4. Handling of Critical Requirements

Critical requirements of the project, as determined in the “Software Requirements Sheet”, will be guaranteed as deliverable under the following:

4.2.4.1. Security Assurances

Team Awesome shall put in place a user-account system analysing and transforming data to ensure that only developers have access to the code and that actions which may result in data deletion or modification require appropriate access levels and authority before proceeding.

4.2.4.2. Privacy Assurance

Team Awesome shall undertake utmost care and diligence when information contained with given Archive is not accidentally disseminated to the public from PyTag without prior approval of the author or owner of the given piece of information

4.2.4.3. Other Assurances

Team Awesome shall ensure that any code taken from a public domain be correctly referenced and licences shall remain intact, to negate (to the best of their ability) any legal actions that may be taken against the Client.

4.2.5. Hardware Resources

The final deployment of the Software will require a server running nginx, Ruby, MySQL all running under a Linux OS.

4.2.6. Recording Rationale

Team Awesome shall record meeting minutes, work diaries, IM conversations, emails and other information during the development of PyTag. Key Decisions made in the project shall be voted on by the Team and the results recorded in the meeting minutes. All arguments for and against will be noted for reference if a dispute occurs in the future.

4.2.7. Client Review

The Client will have the ability to, at any stage of PyTag's development, arrange a meeting to discuss progress and review functionality of the Software. The Client will be the authorised party to make changes to the SRS, provided Team Awesome inform them of their ability to deliver the proposed changes after a vote is performed and noted.

5. Software Development Activities

5.1. Project Planning and Oversight

5.1.1. Software Development Planning

The

5.1.2. Computer Software Configuration Item Test Planning

The

5.1.3. System Test Planning

The

5.1.4. Software Installation Planning

The

5.1.5. Software Transition Planning

The

5.1.6. Management Review

The

5.2. Software Development Environment

5.2.1. Software Engineering Environment

The

5.2.2. Software Test Environment

The

5.2.3. Software Development Library

The

5.2.4. Software Development Files

The

5.2.5. Non-deliverable Software

The

5.3. System Requirements Analysis

5.3.1. Analysis of User Input

The

5.3.2. Operational Concept

The

5.3.3. System Requirements

The

5.4. System Design

5.4.1. System-wide design decisions

The

5.4.2. System architectural diagram

The

5.5. Software Requirements Analysis

The

5.6. Software Design

5.6.1. CSCI-wide design decisions

The

5.6.2. CSCI architectural design

The

5.6.3. CSCI detail design

The

5.7. Software implementation and unit testing

5.7.1. Software Implementation

The

5.7.2. Preparing for Unit Testing

The

5.7.3. Performing Unit Testing

The

5.7.4. Revision and Re-testing

The

5.7.5. Analysing and Recording Results

The

5.8. Unit Integration and testing

5.8.1. Preparing for unit integration and testing

The

5.8.2. Performing unit integration and testing

The

5.8.3. Revision and Re-testing

The

5.8.4. Analysing and Recording Results

The

5.12. Preparing for software use

5.12.1. Preparing the executable software

The

5.12.2. Preparing user manuals

The

5.12.3. Installation and client site

The