1. Introduction and overview

1.1 Purpose of this Document

This document describes how we will test our code for robustness. It includes test cases and other methods of testing.

1.2 Scope of the Development Project

The project entails creating an interface for The American Academy to sponsor tasks for employees to complete to help their company grow the teaching materials base. This will require heavy use of the Drupal Content Management System that TAA uses to create modules representing tasks, projects containing tasks, and the interfaces to store, create, and search tasks.

1.3 Definitions, Acronyms, and Abbreviations

Drupal - Content Management System for administering websites.
TAA - The Academic Academy
V&V - Verification and Validation Plan

1.4 References


SRS - http://www.eng.utah.edu/~blatnick/cs4500/SRS.pdf

1.5 Overview of Document

This document will describe ways in which we will debug, test, and verify our code as being accurate and free from defects. It will include ways to maintain bug-free code through frequent tests as well as considerations to keep in mind while developing and aspects to test that we may have misses in the production phases.

2. Reviews, walkthroughs, inspections, and audits

Every a week on Tuesday we meet with TAA to show the progress, review the work, walk TAA though what has been done, and have them inspect out code. On Thursdays we will assign member to perform quality assurance tests on the different components to ensure their correct functionality.

3. Component test plans and procedures

This section describes the test planning for individual software components. Note that this section and the next will describe expected behaviors. When testing is done, any time the actual output of a test case does not match the expected output, the team will create an incident report within the defect tracker.
3.1 Component Test Strategy Overview

There are two categories of modules we will need to worry about: standard Drupal modules and modules we build ourselves. For the standard Drupal modules, the primary source of bugs will be configuration. We'll test these by clicking through all of the different user scenarios. For modules we create, we will need to unit test more thoroughly. For these, we will use asserts and print statements as well as click through prints.

3.2 Drupal Core:

- Testing process: Most defects we encounter here will be configuration problems. Since this module is required for everything to work, testing will occur automatically as we test other modules. Primary testing has already been done by the Drupal user base.
- Requirements traceability: Drupal core is expected to provide the basic functionality of accessing the database, providing security, and session tracking.
- Items or components tested:
- Testing schedule and resources: Since if anything breaks here, nothing will work, testing will occur on an ongoing basis as we run the application.
- Test recording procedures: It won't be necessary to record anything here.
- Hardware and software requirements: We need a Drupal installation provided by TAA. The web server and database are already provided.
- Constraints: None

3.3 Search Module:

- Testing process: This will be tested every week. The process is to create content of different types and then search to make sure they are all displayed in the proper order.
- Requirements traceability: This module is expected to provide the ability for users to see lists of tasks sorted by project, credentials, and availability.
- Items or components tested:
  - Sort by project
  - Sort by credentials
  - Sort by availability.
- Testing schedule and resources: This will be tested every week before we demo for TAA
- Test recording procedures: Upon testing, the tester will make a log detailing the tests that were run and what happened.
- Hardware and software requirements: We need a Drupal installation provided by TAA. The web server and database are already provided.
- Constraints: we need basic functionality to create content in order to test this module.

3.4 Workflow Module:

- Testing process: This will be the most difficult part of the project because it requires a tremendous amount of customization. We will have to write most of it ourselves, so we will have to do more detailed white box testing than we have to do with most of the other modules. Testing will include a liberal use of asserts in order to make sure the right things are being done. Testing will also include black-box testing to make sure the requirements are met.
- Requirements traceability: This module is expected to provide the ability for an administrator at TAA to customize to create projects and tasks, and verify the work is completed. The administrator will have the ability to customize the workflow according to several categories, for instance, how many people will work on the project at a time, whether bids are accepted or whether there is a specific amount of payment, and whether users with certain credentials are allowed to work on projects without approval.
- Items or components tested:
  - The admin interface: including the ability to customize work flows.
  - The teacher/proctor interface: including the ability to bid on tasks and view whether the work has been approved.
- Testing schedule and resources: This will be tested every week before we demo for TAA
- Test recording procedures: Upon testing, the tester will make a log detailing the tests that were run and what happened.
- Hardware and software requirements: We need a Drupal installation provided by TAA. The web server and database are already provided.
- Constraints: we need the basic ability to create content in order to test this module.
3.4 User Module:

- Testing process: Most of the required functionality is provided by Drupal. Testing will include creating users and verifying security by making sure that administrator functions are only available to administrators. For instance, the ability to verify the credentials of a user should be available only to an administrator.
- Requirements traceability: It should be possible for users to create an account in order to log in. TAA should be able to limit access to certain groups of users by setting parameters in the system.
- Items or components tested:
  - The administrator functionality.
  - The teacher/proctor functionality.
- Testing schedule and resources: This will be tested every week before we demo for TAA
- Test recording procedures: Upon testing, the tester will make a log detailing the tests that were run and what happened.
- Hardware and software requirements: We need a Drupal installation provided by TAA. The web server and database are already provided.
- Constraints: we need the basic ability to create content in order to test this module.

4. System test plans and procedures

This section describes the testing for the software product as a whole.

4.1 System test strategy overview

Because we will be implementing an iterative process for development, our code should be very clean. We will be developing it locally on our computers and then uploading it to the TAA web server and our own web server. We will use SVN to keep track of versions.

4.2 User Module Test

The User Module testing will be simple to do because a user will need to log into the website in order to view its services. Anytime a new function or data structure is added to the user module.

4.3 Work flow / Task

We will test the Work flow task by logging in as a user and selecting the Work flow / Task from the opening block. Then we should be able to manipulate the task and observe its behavior.

4.4 Search Module

The Search Module testing will be simple to do because a user will need to log into the website in order to view its services. Anytime a new function or data structure is added to the user module.

5. Defect tracking plans

Since there will be a degree of separation in the code between team members, defects will be logged in text files for personal use to keep track of known issues, and then forwarded as necessary to other group members. Due to the separation of work, delegation of defects is based on when bugs belong to the current team member's work. Then, if a member is
having difficulty resolving an issue, the bug will be e-mailed to the team member of his choice. The e-mails will be used as a group tracking system, which should prove sufficient due to the size of our group. If the group had been larger, we may have considered a tracking wiki as many companies do, but e-mails should prove effective as a direct communication for prompt results. Bugs are e-mailed to individuals and cc'ed to everyone in the group. Until the e-mail is replied, it is assumed that the bug is not solved.

To review code, we will assign, through e-mail or team meetings, an individual to evaluate someone else's work by clicking through the GUI, trying to break things through any typical or unusual use.

Severity codes used in e-mails will be given in e-mail titles, ranging from 10 (critical) to 1 (for convenience only).

6. Traceability from SRS to SDS

<table>
<thead>
<tr>
<th>Requirements from SRS</th>
<th>References of SDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2 Registration/Login/Logout UI</td>
<td>3.2 Project &amp; Task Management</td>
</tr>
<tr>
<td>3.2.3 Admin UI</td>
<td>3.2 Drupal Core</td>
</tr>
<tr>
<td>3.2.4 Teacher/Proctor UI</td>
<td>3.2 Search Modules</td>
</tr>
<tr>
<td>3.2.5 User Modules</td>
<td>3.2 Work flow Module</td>
</tr>
<tr>
<td>3.2.6 User Modules</td>
<td>3.2 User Modules</td>
</tr>
</tbody>
</table>

7. Test-requirements cross-reference matrix

<table>
<thead>
<tr>
<th>Requirement from SRS</th>
<th>Detailed Description from SDS</th>
<th>Description</th>
<th>Testing Method</th>
<th>Associated System Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1 Drupal Core</td>
<td>3.2</td>
<td>The Drupal back end</td>
<td>Inspection</td>
<td>This will be working if all the other modules are working as well.</td>
</tr>
<tr>
<td>3.2.2 Project / Task</td>
<td>3.3, 3.2, 3.4</td>
<td>Tasks chosen by user and admins</td>
<td>Inspection, Demonstration</td>
<td>Can test this by logging in with the users module and creating tasks.</td>
</tr>
<tr>
<td>3.2.3 Work flow</td>
<td>3.2, 3.3, 3.4</td>
<td>Task applications that are generated when tasks are accepted</td>
<td>Inspection, Demonstration</td>
<td>Can test this by creating a task and inspecting the task application</td>
</tr>
<tr>
<td>3.2.4 Search Modules</td>
<td>3.2 3.5</td>
<td>“View” modules created to help users sort tasks</td>
<td>Inspection, Demonstration</td>
<td>Can test this with the user module and specifying “views”.</td>
</tr>
<tr>
<td>3.2.5 User Modules</td>
<td>3.2, 3.4, 3.5</td>
<td>The module that allows users to login to the site</td>
<td>Analysis, Demonstration</td>
<td>Can test this by logging into the website.</td>
</tr>
</tbody>
</table>
8. Acceptance test and preparation for delivery

8.1 Procedure by which the software product will be acceptance tested

The team will target milestones for acceptance testing. Each milestone will provide a new feature or functionality that the game is capable of supporting. If these milestones are not met to the customers satisfaction, the negotiated changes will be made.

Current Milestones include:

- setting up local and network servers
- setting up users and roles
- setting up "Views"
- creating "task" content
- user menu

8.2 Specific acceptance criteria

The following checklist items will be part of the customer acceptance of the software.

- user (teacher) can register with the site and obtain a login name and password
- user can view tasks
- user can accept tasks
- user can complete tasks
- user can be compensated for tasks
- user can only see tasks for is clearance level
- admin can create tasks
  - variety of prices
- variety of titles
- variety of multilevel task trees
- admin can cancel tasks
- admin can delete tasks
- admin can compensate user for completed task

8.3 Scenario by which the software product will be installed

This product will be installed on any machine that has PHP, Apache and MySQL running. We will import the database with the proper tables when needed.