**Testing Document and Specification**

Test Plan

Pharmacy Team B

CS 451, WS 2006

| Seth Emery | Blake Lord | Brandon Piotrowski |
| --- | --- | --- |
|  | Ryan Borland |  |

# **Introduction**

This document outlines the test plan for the University of Missouri-Kansas City Graduate Teaching Assistant application portal. As outlined in the project Requirements Document, this system needs to provide a medium for UMKC students, staff and faculty to apply for job positions within the GTA environment. This system will also provide a mechanism for reviewing and administrating jobs and their applications. The testing activities discussed in this document will verify that the software for the GTA system meets the needs of the customer by verifying that the requirements for this system, as outlined in the Requirements Document, are met.

The testing routine will test the GTA system’s response to creating and managing user accounts, information input, job searching and filtering, application handling and job administration.The results of this testing procedure will enable the creators of this system to gauge project success as outlined in the Project Plan.

# **Terminology**

Throughout this document the terms user, student, provider, system, site and username/password will be used frequently therefore, formal definitions will be given.

**System**- the database and bulletin board that supports the backend of the website

**User**- people affiliated with UMKC (students, faculty, and staff) who will exclusively use only the front-end of the website

**Student**- students of UMKC who will help oversee and are involved in this website

**Provider**- faculty and staff affiliated with UMKC who will help oversee and are involved in this website

**Site**- the front-end website (main page)

**Username/Password**- unique identifiers that authenticate and validates a user

# **Items Tested**

Items that will be tested during the testing phase as laid out by the Project Plan will be but are not limited to:

* Ability for a non-logged in user to only access select pages.  
  Test Case: 1.1, 1.2, 1.3, 1.4, 1.5, 1.6
* Ability to create a user account and to login  
  Test Case: 2.1, 2.2
* Ability for a logged in user to only access select pages.  
  Test Case: 3.1, 3.2, 3.3, 3.4, 3.5, 3.6
* Ability for an admin user to access all pages.  
  Test Case: 4.1, 4.2, 4.3, 4.4, 4.5, 4.6
* Ability for any user to filter job listings based on criteria  
  Test Case: 5.1, 5.2, 5.3, 5.4. 5.5
* Ability for users to apply to jobs and to update their information  
  Test Case: 6.1, 6.2, 6.3, 6.4
* Ability for admin users to modify and view jobs and applications

Test Case: 7.1, 7.2, 7.3, 7.4, 7.5, 7.6

# **Items Not Tested**

There are features that will not be included in the current testing procedure. This does not mean that these features will not be implemented, but that they have not been implemented and are not available for testing. Those features include but are not limited to:

* Incorrect Responses
* Querying the Database

# **Approach**

The overall method to this testing procedure is manual system testing. Each test case created will have a direct link to the requirements as laid out in the Requirements Document. Test cases that include similar Feature methods will be tested together. Examples of these features include logging in to view applications, logging in to administrate jobs and applications, and/or logging in to the back-end website. Test cases such as these test the security features of the system along with the ability to submit information to the database. Each test case will test the security feature with valid data (usernames and passwords) to ensure that user requirement of valid users for these features is met.

The features that specify user access and features that specify user actions will be tested together but in separate test cases. Will verify that filtering and the search function is working and displaying the correct jobs. Will also verify the users ability to create and login to user accounts.

Manual system testing will continue throughout the project. Adding new features or functionality can sometimes interfere with the functionality of old features and to ensure product/project success, all features implemented should function as intended throughout the life of the software.

# **Item Pass/Fail Criteria**

The minimum requirements for this software system were laid out in the Requirements Document and the Project Plan outlined what the creators of the software considered project success.

Implemented features that meet the requirements as determined by the customer, meaning the feature does what the user wants it to do with very little difficulty, passes the testing procedure. Difficulty, as used here, is determined by user comprehension and user ability to use the feature with little to no training.

Features that contain major defects will fail the testing procedure and will be documented via an incident report and turned over to the developer for investigation and revision.

# **Test Deliverables**

In addition to the Test Plan, other test deliverables include the Test Specification which outlines the specific test cases and expected results of each test, and Test reports which is comprised of Incidents, Defects and Changes.

# **Testing Tasks**

The following list the testing deliverables and the activities required to produce the deliverable.

| **Deliverables** | **Activities** |
| --- | --- |
| **Test Plan** | * Analyze Requirements for System Features * Determine Testable/Non-Testable Features * Develop Approach/Method for testing * Determine Task and Estimate Efforts * Develop Schedule for Testing |
| **Test Specifications** | * Analyze Requirements * Define Test Cases for Testable Features as Outlined by the Test Plan |
| **Test reports** | * Implement Test Cases as Outlined by the Test Specifications * Document Incidents and Defects * Determine Severity of Incidents and Defects * Determine Changes that Need to be Made to System * Document and Submit Change Request to Developer |