Test Plan

Academic Behaviour Recommendation system

Shreyas Ugemuge sugemuge2014@my.fit.edu

Yaqeen AlKathiri yalkathiri2013@my.fit.edu

Mohammed AlHabsi malhabsi2013@my.fit.edu

Shiru Hou shou2015@my.fit.edu

Faculty Sponsor: Dr. Phillip Chan pkc@cs.fit.edu

February 23, 2017
v1.0

Contents

1 Introduction

ABRS (Academic Behavior Recommendation System) is a system used to correlate behaviors, which are extracted from syllabus, grades, and log files, to strong performances. This system is used by teachers and students. Teachers are the providers of the documents mentioned above and the students are the receivers of the recommendations.

2 Testing Objectives

This document contains the list of functions and features expected to be incorporated in our system. Each function has a list of tests and the expected outcome of each test.

2.1 Identify behaviors from syllabus

- This objective is non-functional. Although without it, the system will not be functional.
- Pinpoint at least 10 behaviors that leads to strong performance using 'common sense'.

2.2 Extract behaviors from log files

- Input from files: read the information from a specific type of files (log files) and extract the behaviors.

2.3 Extract performances from grade files

Input from files: read the information from a specific type of files that contain the grades and extract performances accordingly.

2.4 Accepting input from the syllabus

Input: the ability of the software to read and sort the information entered from the syllabus, for example:

- Assignment due dates
- Quizzes due dates

2.5 Display the recommended behaviors for strong performance

Output: the ability of the software to display all the recommendations after extracting the behaviors from different documents.

2.6 Main page

The software behavior in the main page:

- Choose course button
- help button

2.7 Recommendation Page

The software behavior in the recommendation page:

- Reccomendation List
- Performance Report

3 Scope and Limitation of Test Plan

This document is a specification of features to be tested in the ABRS. Features from (2.2 - 2.7) will be tested in how they function and their integration with each other. Since testing take a great amount of time, some features may not have further testing (e.g., 2.5)

4 Sources of Test Data

The sources of test data are the syllabus, log files, grade files, and the codes of the software. The test data, syllabus, log files, and grade files, are given by Dr. Chan. The other test data, codes, will come from the codes that our team is going to write. The amount of effort that will be necessary to acquire all the data, given that we are working in a team, will be basic. Since we are using GITHUB for all sort of documentation. Also, we will be using copies of the software and database, since it is most cost effective from testing perspective and easiest from a data preparation perspective to use copies of the software and database.

5 Testing Strategy

The software will be tested in several different ways which will ensure that the features groups are adequately tested.

5.1 Static Testing

This will test the conducted part of the software. The main objectives to be static tested are from (2.4-2.7). Which will show the performance of the software.

5.2 Unit Testing

- The objectives will be tested using White Box testing techniques since we have all the source code and the executable code. The objectives 2.2-2.6 will need to be unit tested formally by going through each function in the written codes.The testing will also include testing documentation including the requirements, data design, interface design, database structure design, and platform design of the project. This test will be done with maximum degree of comprehensiveness. Some additional criteria to be tested are the errors and warnings frequencies. As this will be the most important part of testing the codes and their executions, we will be using several techniques to trace the requirements for example: inspection review, walkthrough, and final checking.

- Since each team member will be writing different part of the codes from this software, each team member will be testing the other member's work, as another member will be providing the test script for the unit testing.

5.3 Performance Testing

- The performance testing will test how the software will perform. As this matter, we will test objectives 2.6 and 2.7

- For objective 2.7, we will test the main page when the user is the teacher and when the user is the student.

  - Teacher
    - input : entering information from syllabus.
    - input from : uploading log files
    - input from file : uploading grade files
  - Student: Input: choose course
  - For objective 2.7, we will test the output for the user who in this case will be the student.
    - output: reccomendation report
    - output: Performance report

5.4 User Acceptance Testing

- This test will be conducted by asking a group of student to use the software and see how it is working and then collect reviews

- This test will help developing the software performance and correct some objectives (2.6 and 2.7)

5.5 Automated Regression Testing

We will be retesting objectives (2.2-2.5) after each process to check for any error in executable code or running time output error

6 Control Procedures

6.1 Problem Reporting

We will be documenting the procedures to follow when an incident is encountered during the testing process.

- Opening Application
- Login / sign up
- Choosing course
- Displaying reports
- input data from given fies

6.2 Changes in project

We will be documenting the process of modifications to the software. Dr. Chan will be signing off on the changes. The changes will be specified under the criteria that includes them. If the changes will affect existing objectives, they will be identified.

7 Features to be tested

7.1 Login

- Usual test case: Enters valid credentials.
  Expected output: Logs in successfully.
- Unusual test case: Enters information for an old account that has been deleted.
  Expected output: failure to login.

7.2 Change Password

- Usual test case: Enters valid new password.
  Expected output: password is changed.
- Unusual test case: Enters a password that is too weak.

7.3 Sign Up

- Usual test case: Enters valid information.
  Expected output: Account created.
- Unusual test case: Enters information about an account that already exists.
  Expected output: failure to create account.

7.4 Information from syllabus

- Usual test case: teacher inputs all syllabus information correctly.
  Expected output: Information processed correctly.
- Unusual test case: input is missing some information.
  Expected output: cannot proceed to generate report.

7.5 Log File and Grade file upload

- Usual test case: the files uploaded are complete and in the right format.
  Expected output: accepts files to be processed.
- Unusual test case: upload files in an unsupported format.
  Expected output: display an error message and attempt to take a different file

7.6 Extracting performance from grade file

- Usual test case: click the correlate behaviors and performance button after correctly uploading all files.
- Expected output: displays report.

7.7 Extracting behaviors from Grade File

- Usual test case: click the button after correctly uploading all files.
  Expected output: displays behavioral report.

7.8 Select course

- Usual test case: User enters a correct course name.
  Expected output: proceed to view recommendations.
- Unusual test case: Enters a course name that does not exist.
  Expected output: cannot proceed to recommendations page

7.9 View Recommendations Report

- Usual test case: User clicks on view recommendations after choosing a course.
  Expected output: correct report is generated.

8 Features Not Being Tested

Some features in our system will not be tested as we are not likely to benefit from testing these feature:

- Testing whether or not the syllabus information inputted is consistent with the syllabus.
- Testing if the grade file and log file are records of the same students.

9 Risks/Assumptions

- Identify the high-risk assumptions of the test plan. We will specify contingency plan for any risks. For example:

  Delay in completions of a test item might require increase night scheduling to meet the delivery date.

10 Tools

- Python/Java
- Github
- one of more for:
  - Advanced Miner
  - Ghost Miner
  - GNOME
  - ESTARD
  - TANAGRA
  - SPAD
  - WEKA