**SAS: Master Test Plan**

**1 Test Plan Identifier**

1019MTP01.0

**2 References**

None

**3 Introduction**

This is the master test plan for the student advisement assisting tool project. It covers the user interface, Schedule checking system, schedule making system, and the form update process,The main focus of this plan is to make sure the information output from the system is reliable and meets the expectations of the user.

**4 Test Items**

The following is a list, by version and release, of the items to be tested:

1. System UI v 1.0
2. Schedule Checker v 1.0
3. Error Report System v 1.0
4. Schedule Maker v 1.0

**5 Software Risk Issues**

A. Correctly Parsing the unofficial transcript pdf

B. Backup recovery for lost files is required

C. Multiple people accessing system at once

**6 FEATURES TO BE TESTED**

A. Graphical interface in check sheet form, for faculty advisor to view, edit and update advisee’s academic information.

B. Recommend course schedule for upcoming semester.

C. Provide alerts if advisee is at risk of deviating from advisee’s planned graduation date.

D. Keep track of advisee’s academic progress and store student academic records.

E. Generate information to help advisor’s fill out advisee forms.

F. Allow advisor to store notes about student.

G. Retrieve and update course information, semester offerings, university requirements, and department check sheets.

**7 FEATURES NOT TO BE TESTED**

A. Network security and internet access.

B. Minors are not integrated into the system, and therefore will not be tested

**8 APPROACH**

**8.1 Testing Levels**

The testing for the Student Advisement System will consist of Unit and Integration test levels. There are three modules and three developers. Each developer will be assigned a module to test that he/she did not implement themselves. There is no major difference between Unit and Integration testing. Each module individually interacts with the locally stored data, and NOT with other modules in the system, and therefore the major concern is making sure each module is accurately capturing the data, and storing it locally on the server.

**UNIT Testing** will be performed by a tester who did not write the module and tests for that module. Unit test cases must be well documented (test case list, sample output, data printouts, defect information) before being accepted and passed on to the test person. All unit test information will also be provided to the test person. The test person will record their results and send them back to be analyzed and evaluated by the developer of the module. White box and black box testing will be used at this level of testing. For a black box approach, we will have a known set of outputs for a given set a of inputs. We will enter the inputs, and only examine what comes out the other end. We will also use a white box approach where we follow the data from input, throughout the execution of the program, to the output results.

**INTEGRATION Testing** will be performed by all three of the project developers. Modules should have no major defects before being accepted for integration testing. A top down testing approach will be used to ensure the modules work together. The one difference is that test data must be followed throughout each of the three modules. We will be using a big bang approach because there are only three modules to be tested.

**8.2 Configuration Management/Change Control**

All system files reside in the project’s Github repository [https://github.com/kusoftwarefun/Student-Advisement-System]. Github has an extensive configuration, version, and change management system. All changes, enhancements and other modification requests to the system will be handled through the Github interface, and typed out in a standard format; purpose of issue, high level description of the issue, low level description of the issue, priority level of issue (minor, major, critical ), unique ID for issue.

**8.3 Test Tools**

* CSIT Course Dependencies Graph
* CSIT Checksheets
* Browser

**8.4 Meetings**

The team will meet once every two weeks to evaluate progress to date and to identify

error trends and problems as early as possible. Additional meetings can be scheduled as required, and online communication is always available.

**8.5 Measures and Metrics**

The following information will be collected by the Development team during the testing

process. This information will be provided to the test team at program turnover as well as be

provided to the project team on a weekly basis.

1. Defects by module and severity.

2. Defect Origin (Requirement, Design, Code)

3. Time spent on defect resolution by defect, for Critical & Major only. All Minor defects

can be totaled together.

**9 ITEM PASS/FAIL CRITERIA**

At any phase testing is complete when the only remaining defects in the system are categorized as minor.

**10 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS**

Generally not applicable. The team will use common sense, such as not testing data sets that are virtually equivalent (i.e. recording duplicate test data). As such, tests should always be yielding valuable information, pass or fail.

**11 Test Deliverables**

As part of the test deliverables, our team will convey our findings in the form of a test plan that includes the following components:

* Test plan document
* Test design specifications
* Test cases
* Test procedure steps
* Test Data
  + Test logs (Bugs/Errors & Execution)
  + Test summary report
  + Test revision history for each bug or error

\*Software components (Modules & Algorithms) are not included in test deliverables\*

**12 Test Tasks**

|  |  |
| --- | --- |
| Skill Level | Description |
| 0 | Consumer Level (No software experience, may need help navigating UI) |
| 1 | Intermediate (User knows how to navigate and input data) |
| 2 | Advanced (User is aware of what is going on behind the scenes, but is not familiar with the implemented modules) |
| 3 | Developer / Tester (User knows how to implement, design and write code pertaining to specific system) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Deliverable | Tasks | Inter-Task Dependencies | Skill Level Required |
| 01 | Test Plan Document | 1. Identify specifications for test components 2. Develop test case for components 3. Test cases 4. Develop test logs 5. Generate test summary reports 6. Record test revisions | A ->  B -> A  C -> B  D -> A, B, C  E -> A, B, C, D  F -> A, B, C, D, E | 2-3 |
| 02 | Test Design Specifications (What needs to be tested) | 1. Features to be tested 2. Analyze modules specifications 3. Create specifications list for each module tested (Include unique identifier, version date, number, author & contact information) 4. Create test identifications (Description of test or any test relationships) 5. Describe pass/fail criteria | A ->  B -> A  C -> A, B  D -> A, B, C  E -> A, B, C, D | 3 |
| 03 | Test Cases (What is tested) | 1. Create unique identifier for individual test cases 2. Test modules and features (Derived from TDS) 3. Record input 4. Record output 5. Describe environmental needs (Anything missing from TDS) 6. Record special procedures 7. List inter-case dependencies | A -> ID#02  B -> ID#02, A  C -> B  D -> B, C  E -> ID#2, B, C, D  F -> B, C, D  G -> ID#02, A | 3 |
| 04 | Test Procedure Steps | 1. Create a unique identifier for test procedures 2. Describe objective of test procedure 3. Describe special requirements for test 4. List procedure steps taken to test components | A ->  B ->  C -> B  D -> ID#03 | 3 |
| 05 | Test Data | 1. Create unique identifier for each test 2. Test modules based on test case procedures 3. Create summary based on acquired test data 4. Generate revision history for each failed test performed | A ->  B -> ID#03  C -> ID#03, B  D -> ID#03, B, C | 2-3 |

**13 Environmental Needs**

Special requirements for this specific test plan include:

* Communications: Network access for web based application
* Test data (Ex: Student information)
* Test cases for testing modules

Test Data will be provided in the form of a test case word document template.

Special requirements are the developers of this project will be carrying out the testing procedures and test cases.

Multi-parts are divided up between the developer team and will be swapping roles to test each other’s modules. Cross testing will also be fair game to each of the team members. Please refer to the WA1.1 in the engineer’s repository for specifics on team member module testing.

**14 Staffing & Training Needs**

Training for faculty will be provided with a simple README.txt file or if instructions are unclear, team members will assist faculty in operating the system.

No training is required for the developers who will be testing individual modules.

**15 Responsibilities**

|  |  |  |
| --- | --- | --- |
| Team Member | Task | Description |
| Patrick | Test Plan Components | Test Plan Components (11-16) |
| Patrick | Test Cases | UI Test Cases   * Features to be tested / not tested * Ensuring all elements are present for testing * Setting risks * Training for UI |
| Patrick | Testing | Will test scheduling module |
| Geoffrey | Test Plan Components | Test Plan Components (6-10) |
| Geoffrey | Test Cases | Advisement Module Cases   * Features to be tested / not tested * Ensuring all elements are present for testing * Setting risks * Training for advisement modules * Module breakdown / explanation |
| Geoffrey | Testing | Will test UI modules |
| Alan | Test Plan Components | Test Plan Components (1-5 & 17-18) |
| Alan | Test Cases | Scheduling Module Cases   * Features to be tested / not tested * Ensuring all elements are present for testing * Setting risks * Module breakdown / explanation |
| Alan | Testing | Will test advisement module |
| Patrick, Geoffrey, Alan | Decision Making | Critical go/no go situations will be discussed as a team and action will be implemented as a team |
| Patrick, Geoffrey, Alan | Plan Strategy | This level of plan will be discussed as a team and action will be implemented based off discussions |
| Patrick, Geoffrey, Alan | Scheduling | Conflicts in scheduling will be reviewed as a team. If one team member is absent, remaining team members will discuss matters |
| Patrick, Geoffrey, Alan | Dependencies | Team will discuss dependencies of modules (Ex: UI, Advisement & scheduling modules) Dependencies may effect scheduling |
| Patrick, Geoffrey, Alan | Environmental Needs | Team will determine if any environmental needs are necessary or required for testing |

**16 Schedule**

\*Please refer to SCH2.1 Microsoft Project Plan for estimates\*

**16.1 Handling Scheduling Slippage**

If a single developer notices scheduling slippage or has concerns about slippage, the entire developing team will be notified ASAP through any means of communication. (Phone, Hangouts, Texts, etc.) Team will refer to SCH2.1 Project Plan for any change in scheduling or any slippage. If major slippage occurs, team will decide in what features will be present or removed from the application in order to avoid major slippage. Decisions will be discussed as a whole. Test dates will be recorded along with related development activity dates from the SCH2.1 Project Plan.

**17 PLANNING RISKS AND CONTINGENCIES**

1. The Parsing Document will be studied and any problems will be sent to the advisor if they come up.
2. Proper backup system will be looked into to meet the needs of the users. Extra storage and storage outside of system will be looked into.
3. The System will be stress tested to make sure this is not an issue.

**18**

|  |  |
| --- | --- |
| Alan Duffy-Guy - Systems Analyst, Systems Developer |  |
| Geoffrey Pitman - Project Manager, Systems Designer |  |
| Patrick Gagliano,- Team Lead, Systems Developer |  |