Test Plan Document

1.Introduction

Our product, Course Planner, is a website in which students can rate specific tasks in their various courses, organize their schedule and receive reminders based on the ratings to these tasks. The website will run on a LAMP stack and the front-end of the site will be written in Javascript, PHP, CSS and HTML.

2.Verification Strategy

In order to verify that our website is meeting users needs, it is important that we remain focused on the goals of our product, as stated in our requirements documents, as they are our best feelings on what our expected users will need and what functions our product will have to meet those needs. However, we must remain willing to adapt such requirements to the actual needs and wants of our users, should they differ. To obtain such feedback from users, we will begin with internal testing with ourselves. As we are all currently students, this product will be helpful to serve us as well as our main user base. We will test using CPEN 321 course elements and information, as it is a common course for all of us and if all of our testers are using the same course information and tasks, gathering data to use in testing will be much easier by focusing on a single common course. We will also demo iterative GUI designs to our group of users to ensure users find elements of the site easy to find and use.

3.Non-Functional Testing and Results

1. Configuration Testing:
   This testing is used to ensure all team member understand the project requirements: operating system platform used, type of network connection, server provide type and browser used.
   1. Action: we will have a group meeting to require each member to describe his own understanding of the project requirements, and he ask questions and get feedback from other team members.
   2. Expect result: all members understand the project requirements.
   3. Actual result: all members understand the project requirements.
   4. P/F: P
   5. Notes: the project requirement is discussed by all members, so each of us understand the project requirements.

2. Usability Testing:
   This testing is used to ensure end-user can accept the website standards and guidelines.

1. Actions: we will provide a standards and guidelines document for the website users, and will collect their usability surveys.
2. Expect result: 80% of users can follow the standards and guidelines document.
3. Actual result:
4. P/F:
5. Notes: this testing is not start yet, because the website is not finished.

3. Security Testing:

   This testing is used to check the website security regarding of regulate access to database or users' information, verify user identities, and encrypt confidential information is of paramount importance.

   1. Actions:
      a. Using an unauthorized account to login to the website and browser the log files.
      b. Checking URLs that ensure there is no data leaks while using a GET command.
      c. If users do not accept cookies, the website should not work.
      d. Checking the sensitive information in cookies is encoded or encrypted.
   2. Expected result:
      a. Unauthorized account cannot login
      b. No data leaks
      c. Cookies acceptation required
      d. Sensitive data encrypted
   3. Actual result:
   4. P/F:
   5. Notes: this testing is not start yet, because the website is not finished.

4. Recoverability Testing:

   This testing is used to ensure the data in database can be recovery.

   1. Actions: If suddenly closing browsers while data is inputting, database will be checked that the percentage of data has lost.
   2. Expected result: lost data percentage should be controlled within 5%.
   3. Actual result:
   4. P/F:
   5. Notes: this testing is not start yet, because the website is not finished.

Test #	Purpose	Action	Expected Result	Actual Result	P/F	Notes
1	Configuration testing	Group meeting	Members understand requirements	Members understand requirements	P	

2	Usability testing	Users surveys	80% users satisfied			Not started
3	Unauthorized account testing	Using unauthorized account	Unauthorized account Failed			Not started
4	URLs testing	Using a GET command	No data leaks			Not started
5	Cookies testing	Disable/Enable cookies acceptation	Cookies acceptation required			Not started
6	Data encrypted testing	Check data in cookies	Data encrypted			Not started
7	Recoverabilit y testing	Close browsers	Data lost within 5%			Not started

4.Functional Testing Strategy

1. Test login with facebook account
    a. Case 1 Login successfully
        i. Input: facebook accounts of our teammates
        ii. Expected Output: log into the Course Planner account
    b. Case 2 Login with wrong password
        i. Input: facebook accounts of our teammates with random password
        ii. Expected Output: facebook dialog indicating account or password is wrong
2. Test sidebar navigation (Unit test)
    a. Case 1 open sidebar:
        i. Input: Click on open button
        ii. Expected Output: length of sidebar was set from 0 to a certain width
    b. Case 2 close sidebar:
        i. Input: Click on close button
        ii. Expected Output: length of sidebar was set from a certain width to 0
    c. Case 3 navigate sidebar:
        i. Input: Click on each sidebar-option in order
        ii. Expected Output: web page changes correspondingly

3. Test scheduling panel (Unit test)
    a. Case 1 add a single task without reminder
        i. Input: trigger the addTask function by clicking on the empty space of the calendar, and fill in information with single task checkbox selected to the addTask dialog
        ii. Expected Output: updated information stored in expected database and shown on the web page
    b. Case 2 add a regular task without reminder
        i. Input: trigger the addTask function by clicking on the empty space of the calendar, and fill in information with regular task checkbox selected to the addTask dialog
        ii. Expected Output: updated information stored in expected database and shown on the web page
    c. Case 3 add a single task with customized reminder
        i. Input: trigger the addTask function by clicking on the empty space of the calendar, and fill in information with single task and remind me checkboxes selected to the addTask dialog
        ii. Expected Output: updated information stored in expected database and shown on the web page
    d. Case 4 add a regular task with customized reminder
        i. Input: trigger the addTask function by clicking on the empty space of the calendar, and fill in information with regular task and remind me checkboxes selected to the addTask dialog
        ii. Expected Output: updated information stored in expected database and shown on the web page
    e. Case 2 remove a task/class
        i. Input: trigger the removeTask function by clicking on an task/class time session on the calendar, and click on delete button on the dialog
        ii. Expected Output: information deleted from the expected database table and the web page
4. Test add/modify profile
    a. Case 1 add courses inside our database
        i. Input: navigate to the courseRegister.php page, and fill in courses information which are inside our database course table
        ii. Output: all the courses are added to the database table “Unique Calendar Entry” with the user ID.
    b. Case 2 add courses not in our database
    c. Case 3 fill in info
        i. Input: navigate to the courseRegister.php page, and fill in courses information which are not in our database course table
        ii. Output: nothing

4. Test the main page reminder
    a. Case 1 do not set up reminder date when add task:
        i. Input : NULL;
        ii. Expected Output: show the task at the exact task date, no reminder before that.
    b. Case 2 choose to remind user itself two days beforehand.
        i . Input : 2;
        ii. Expected Output: two days before the task happen, when user click the check your task button, should remind he/she this task.
5. Test the email reminder panel
    d. Case 1 email set:
        i. Input: related information, reminding time
        ii. Expected Output: the information sent to user's email
    e. Case 2 no email:
        i. Input: related information, reminding time, email address
        ii. Expected Output: the information sent to users

5.Adequacy Criterion

a)Make sure that at least one test exists for each method written.
b)Make sure that sufficient tests exist that each line of program code is executed by at least one test. It is the simplest way of testing.
c)Make sure that at least one test exists for each requirement and for each use case.
d)If a test suite fails to satisfy some criterion, the obligation that has not been satisfied may provide some useful information about improving the test suite.
d)If a test suite satisfies all the obligations by all the criteria, we do not know definitively that it is an effective test suite, but we have some evidence of its thoroughness.
e)if the specification describes different treatment in two cases, but the test suite does not check that the two cases are in fact treated differently, we may conclude that the test suite is inadequate to guard against faults in the program logic.
f)If no test in the test suite executes a particular program statement, the test suite is inadequate to guard against faults in that statement.

6.Test Cases and Results

Database Tests

Test #	Requirement	Action	Expected Result	Actual Result	P/F	Notes

	Purpose					
1	Be able to add entries to the database	Use mySQL commands to add new entries to the database	Entry information that was not previously in the database can now be retrieved from the database			Not started
2	Retrieve entry information from database	Use mySQL commands to display information held in the database	All appropriate information from the database is displayed			Not started
3	Be able to modify information fields in the database	Use mySQL commands to modify a given field	The targeted information in the database has been replaced with the new information			Not started

UI

Test #	Requirement Purpose	Action	Expected Result	Actual Result	P/F	Notes
1	Be able to sign into Course Planner	Use Facebook login extension to login	Be able to access your personal information and schedule			Not started
2	Pass information from database to the UI	Using the UI request information from the database and the information is then displayed by the UI	The UI displays course information retrieved from the database			Not started
3	Add user profile to database from the UI	Using the UI and the facebook login generate unique user profiles in the database	New user profile is added to the database by the UI			Not started
4	Modify	Using the UI add	Database entries			Not started

	database entries from the UI	or modify entry to database	are modified or created in the database as defined by the user			