**Test Plan**

**Instructor-Course Assignment Application**

**Project Manager: Kiana McDaniel**

**David Johnson, Phillip Tran**

**Texas State University**

**Ingram School of Engineering**

**Texas State University**

**601 University Dr**

**San Marcos, Texas, 78666**

**11/28/2018**



**Test Plan Revision History**:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Revision Date | Description | Author |
| .05 | 11/21/2018 | Started document based on template, finished first page and drafted Overview | David |
| .1 | 11/23/2018 | Add an initial list of tests to be conducted | David + Phillip |
| .5 | 11/27/2018 | Filled in some of the test case information | Kiana + Phillip |
| 1.0 | 11/28/2018 | Finalized Test Plan | ALL |

Contents

[1 Overview – David 3](#_Toc531206513)

[2 Features to be tested/not to be tested -David 3](#_Toc531206514)

[2.1 Features to be tested 3](#_Toc531206515)

[2.1.1 View Courses 4](#_Toc531206516)

[2.1.2 Add a Course 4](#_Toc531206517)

[2.1.3 Edit a Course 4](#_Toc531206518)

[2.1.4 View Instructors 4](#_Toc531206519)

[2.1.5 Add an Instructor 4](#_Toc531206520)

[2.1.6 Edit an Instructor 4](#_Toc531206521)

[2.1.7 Make an Instructor-Course Assignment 4](#_Toc531206522)

[2.1.8 Response Time 4](#_Toc531206523)

[2.1.9 Memory Usage 4](#_Toc531206524)

[2.1.10 Security 4](#_Toc531206525)

[2.1.11 System Stability 4](#_Toc531206526)

[2.1.12 Assignment Recommendation 4](#_Toc531206527)

[2.2 Features not to be tested 4](#_Toc531206528)

[2.2.1 Functional on different platforms 4](#_Toc531206529)

[2.2.2 Functional on Windows 10 4](#_Toc531206530)

[3 Testing Approach – Phillip+David 5](#_Toc531206531)

[4 Test Cases – David + Kiana 9](#_Toc531206532)

[4.1 Test Case #1: View Courses Test – Kiana 10](#_Toc531206533)

[4.2 Test Case #2: Add a Course Test - Kiana 11](#_Toc531206534)

[4.3 Test Case #3: Edit a Course Test – Kiana 12](#_Toc531206535)

[4.4 Test Case #4: View Instructors Test – Kiana 13](#_Toc531206536)

[4.5 Test Case #5: Add an Instructor Test – Kiana 14](#_Toc531206537)

[4.6 Test Case #6: Edit an Instructor Test – Kiana 15](#_Toc531206538)

[4.7 Test Case #7: Make an Instructor-Course Assignment Test – David 16](#_Toc531206539)

[5 Testing Schedule 17](#_Toc531206540)

# Overview – David

The Instructor-Course Assignment Application is a web-based application that encompasses the software components needed to allow users to remotely access a database and manage Instructor-Course assignments. Interaction with the system occurs through a web-page based front end that allows the user to view and edit Instructor and Course information. The primary goal of this application is to be convenient and ergonomic for the user, while remaining simple to implement and maintain by Administrators. The project is being designed with the intention of being implemented on a secure server on the Texas State University network system. Users of this application include Program Coordinators (who will eventually be Department Chairs), the Director (who will eventually be the Dean), Administrative Assistants of the Ingram School of Engineering, and System Administrators.

# Features to be tested/not to be tested -David

## Features to be tested

The following are the major functionalities of the application that need to be tested in the testing process:

### View Courses

### Add a Course

### Edit a Course

### View Instructors

### Add an Instructor

### Edit an Instructor

### Make an Instructor-Course Assignment

### Response Time

### Memory Usage

### Security

### System Stability

### Assignment Recommendation

## Features not to be tested -David

### Functional on different platforms

Although the program should be functional on most platforms, including most Linux distributions, Mac OS, and some mobile platforms, only functionality on Windows 10 will be tested since there is no sponsor requirement, or project precedent to explicitly test it on other platforms.

### Functional on Windows 10

Since the system is considered functional on Windows 10 if all of the other tests are able to pass successfully, and those tests will only be conducted on Windows 10, the system’s functionality on Windows 10 is tested implicitly by the other test cases.

# Testing Approach – Phillip+David

|  |  |
| --- | --- |
| *View Courses 2.1.1* | -Phillip |
| Approach | This test will be performed manually by a human user at various points of the system’s operation and during different states, to test the common and edge cases. The user will attempt to use the view courses action to have the system display the list of courses for the semesters in the system. |
| Pass/Fail Criteria | The system will pass this test if all of the courses currently in the database are displayed on the user front-end, and the front-end handles text wrapping and enough courses to require multiple screens without any columns/rows overlapping. |
| Verification Method | The courses displayed will be verified against the courses currently in the database, both visually and using a script to parse out the displayed data to ensure it is correct. |

|  |  |
| --- | --- |
| *Add a Course 2.1.2* | -Phillip |
| Approach | This test will be performed manually by a human user with a variety of different inputs to test the input verification, as well as performed automatically by a script to ensure the system can handle a large number of courses. |
| Pass/Fail Criteria | The system will pass this test if the database and front-end reflect the new course(s) that was added. |
| Verification Method | This test will be verified visually, as well as by checking the database for the added course. |

|  |  |
| --- | --- |
| *Edit a Course 2.1.3* | -Phillip |
| Approach | This test will be performed manually by a human user on various pre-existing courses in the system. |
| Pass/Fail Criteria | The system will pass this test if the database and front-end reflect the edits made to the course. |
| Verification Method | This test will be verified visually, as well as by checking the database for the edited course. |

|  |  |
| --- | --- |
| *View Instructors 2.1.4* | -Phillip |
| Approach | This test will be performed manually by a human user at various points of the system’s operation and during different states, to test the common and edge cases. The user will attempt to use the view instructors action to have the system display the list of instructors in the system. |
| Pass/Fail Criteria | The system will pass this test if all of the instructors currently in the database are displayed on the user front-end, and the front-end handles text wrapping and enough courses to require multiple screens without any columns/rows overlapping. |
| Verification Method | The instructors displayed will be verified against the instructors currently in the database, both visually and using a script to parse out the displayed data to ensure it is correct. |

|  |  |
| --- | --- |
| *Add an Instructor 2.1.5* | -Phillip |
| Approach | This test will be performed manually by a human user with a variety of different inputs to test the input verification, as well as performed automatically by a script to ensure the system can handle a large number of instructors. |
| Pass/Fail Criteria | The system will pass this test if the database and front-end reflect the new instructor(s) that was added. |
| Verification Method | This test will be verified visually, as well as by checking the database for the added instructor. |

|  |  |
| --- | --- |
| *Edit an Instructor 2.1.6* | -Phillip |
| Approach | This test will be performed manually by a human user on various pre-existing instructors in the system. |
| Pass/Fail Criteria | The system will pass this test if the database and front-end reflect the edits made to the instructor. |
| Verification Method | This test will be verified visually, as well as by checking the database for the edited instructor. |

|  |  |
| --- | --- |
| *Make an Instructor Course Assignment 2.1.7* | -David |
| Approach | This test will be performed manually by a human user with a wide variety of instructors and courses to test the assignment function, as well as the warnings for when an instructor is assigned to a course they cannot teach, when an instructor is assigned too many courses, when they are not assigned to enough courses, and when a course does not have enough sections being taught |
| Pass/Fail Criteria | The system will pass this test if the assignment is reflected in the database and front-end, and any applicable warnings are displayed to the user |
| Verification Method | This test will be verified by checking the database that the assignment has been made, and visually to confirm any applicable warnings. |

|  |  |
| --- | --- |
| *Response Time 2.1.8* | -Phillip |
| Approach | During the standard operation of the program (simulated by the first 7 tests), the response time to user inputs should be consistently below 50ms |
| Pass/Fail Criteria | The system will pass this test if the response time is below 50ms on average during simulated use |
| Verification Method | The browsers used to access the webpage provide dev tools which can report the average response time of the webpage and ensure it is below 50ms |

|  |  |
| --- | --- |
| *Memory Usage 2.1.9* | -Phillip |
| Approach | During the standard operation of the program (simulated by the first 7 tests), the memory usage on the user’s computer used to access the webpage should stay below 500MB |
| Pass/Fail Criteria | The system will pass this test if the memory usage does not exceed 500MB during simulated use |
| Verification Method | The operating systems of the computers used for testing include system tools which can report the memory used by the program, and ensure it is below 500MB |

|  |  |
| --- | --- |
| *Security 2.1.10* | -Phillip |
| Approach | During the standard operation of the program (simulated by the first 7 tests), the data streams between the webpage, server, and database should be encrypted to prevent security breaches |
| Pass/Fail Criteria | The system will pass this test if the data streams are encrypted and the data transmitted cannot be recovered by entities outside of the system |
| Verification Method | A middleman application will be set up between the server, webpage, and database and scan through all transmissions to see if any data can be recovered. |

|  |  |
| --- | --- |
| *Stability 2.1.11* | -David |
| Approach | During the standard operation of the program (simulated by the first 7 tests), the webpage should be stable and should not crash or have unexpected events from any user inputs. |
| Pass/Fail Criteria | The system will pass this test if the webpage cannot be “crashed” by a series of statistically random inputs |
| Verification Method | A “monkey test” will be used on the webpage, to generate random mouse clicks and keyboard inputs to the webpage in an attempt to crash the page. |

|  |  |
| --- | --- |
| *Assignment Recommendation 2.1.12* | -David |
| Approach | When making an instructor course assignment, the available instructors should be listed in order of what the system has calculated to be ideal to least ideal. |
| Pass/Fail Criteria | The system will pass this test if it can consistently list the instructors in a reasonable order |
| Verification Method | Multiple sets of instructors and courses will be generated to simulate all reasonable combinations that could appear in the real world. |

# Test Cases – David + Kiana

NOTE For test cases: Since listing all possible and relevant combinations of tests that will be run would require several dozen test cases, the tests for each feature have been condensed into a single test case. Additionally, in the process of starting all of these tests, the programs needed to record the response time, memory usage, and transmission security will be initialized to make better use of the time spent testing.

When the term system is used, it refers to the overall system, including the database, python code, and webpage interface.

When the term server is used, it refers to the python and JavaScript program running on a physical server which handles data processing.

## Test Case #1: View Courses Test – Kiana

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Kiana McDaniel | |
| **Test Case Number** | | 1 | |
| **Test Case Name** | | View Courses Test | |
| **Test Case Description** | | **This test case will ensure that the user can view all of the courses that are currently in the system.** | |
| **Item(s) to be tested** | | | |
| 1 | User Interface’s ability to display all required screens without text overlapping or going off screen | | |
| 2 | Database’s successful storage of course data | | |
| 3 | Server’s ability to retrieve course data from database and push to user interface | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * User mouse inputs to navigate web-page * Database’s stored course data | | | A list of all of the courses currently in the system for the current school year should be displayed on the frontend user interface. |
| **Resources Required** | | | |
| 1 | Laptop or PC with mouse and keyboard | | |
| 2 | Access to the server directly or via IP/URL | | |
| **Procedural Steps** | | | |
| 1 | Mouse hover over book icon | | |
| 2 | Mouse click on “View Courses” dropdown option | | |
| 3 | View Courses | | |
| 4 | Validate displayed courses for successful test | | |

## Test Case #2: Add a Course Test - Kiana

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Kiana McDaniel | |
| **Test Case Number** | | 2 | |
| **Test Case Name** | | Add a Course Test | |
| **Test Case Description** | | **This test case will ensure that the user can add a new course to the system.** | |
| **Item(s) to be tested** | | | |
| 1 | User Interface’s ability to display all required screens without text overlapping or going off screen | | |
| 2 | Database’s successful storage of course data | | |
| 3 | Server’s ability to retrieve user input and convert it into course data to be saved to database | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * User mouse inputs to navigate web-page * User keyboard inputs to fill out course information | | | A form will appear that shows data fields to be entered by the user. Addition of the course is reflected in “View Courses” page, and database |
| **Resources Required** | | | |
| 1 | Laptop or PC with mouse and keyboard | | |
| 2 | Access to the server directly or via IP/URL | | |
| **Procedural Steps** | | | |
| 1 | Mouse hover over book icon | | |
| 2 | Mouse click on “Add Course” dropdown option | | |
| 3 | Add Course | | |
| 4 | Mouse click on “Save & Submit” | | |
| 5 | Visually confirm the new course with “View Courses” | | |
| 6 | Probe the database to ensure the course has been added | | |

## Test Case #3: Edit a Course Test – Kiana

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Kiana McDaniel | |
| **Test Case Number** | | 3 | |
| **Test Case Name** | | Edit a Course Test | |
| **Test Case Description** | | **This test case will ensure that the user edit a course in the system.** | |
| **Item(s) to be tested** | | | |
| 1 | User Interface’s ability to display all required screens without text overlapping or going off screen | | |
| 2 | Database’s successful storage of non-redundant course data | | |
| 3 | Server’s ability to retrieve user input and convert it into course data to be saved to database, over-writing pre-existing data | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * User mouse inputs to navigate web-page * User keyboard inputs to fill out course information | | | A form will appear that shows data fields filled with the current course data. Edits to the course are reflected in “View Courses” page and database. |
| **Resources Required** | | | |
| 1 | Laptop or PC with mouse and keyboard | | |
| 2 | Access to the server directly or via IP/URL | | |
| **Procedural Steps** | | | |
| 1 | Mouse hover over book icon | | |
| 2 | Mouse click on “View Courses” dropdown option | | |
| 3 | Mouse click on edit icon button | | |
| 4 | Edit Course | | |
| 5 | Mouse click on “Save & Submit” | | |
| 6 | Visually confirm the edited course with “View Courses” | | |
| 7 | Probe the database to ensure the course has been edited | | |

## Test Case #4: View Instructors Test – Kiana

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Kiana McDaniel | |
| **Test Case Number** | | 4 | |
| **Test Case Name** | | View Instructors Test | |
| **Test Case Description** | | **This test case will ensure that the user can view all of the instructors that are currently in the system.** | |
| **Item(s) to be tested** | | | |
| 1 | User Interface’s ability to display all required screens without text overlapping or going off screen | | |
| 2 | Database’s successful storage of instructor data | | |
| 3 | Server’s ability to retrieve instructor data from database and push to user interface | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * User mouse inputs to navigate web-page * Database’s stored instructor data | | | A list of all of the instructors currently in the system should be shown on the frontend user interface, this list should match the database. |
| **Resources Required** | | | |
| 1 | Laptop or PC with mouse and keyboard | | |
| 2 | Access to the server directly or via IP/URL | | |
| **Procedural Steps** | | | |
| 1 | Mouse hover over profile icon | | |
| 2 | Mouse click on “View Instructors” dropdown option | | |
| 3 | View Instructors | | |

## Test Case #5: Add an Instructor Test – Kiana

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Kiana McDaniel | |
| **Test Case Number** | | 5 | |
| **Test Case Name** | | Add an Instructor Test | |
| **Test Case Description** | | **This test case will ensure that the user can add a new instructor to the system.** | |
| **Item(s) to be tested** | | | |
| 1 | User Interface’s ability to display all required screens without text overlapping or going off screen | | |
| 2 | Database’s successful storage of instructor data | | |
| 3 | Server’s ability to retrieve user input and convert it into instructor data to be saved to database | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * User mouse inputs to navigate web-page * User keyboard inputs to fill out instructor information | | | A form will appear that shows the data fields to be entered by the user. Instructor added shown in “View Instructors” page. |
| **Resources Required** | | | |
| 1 | Laptop or PC with mouse and keyboard | | |
| 2 | Access to the server directly or via IP/URL | | |
| **Procedural Steps** | | | |
| 1 | Mouse hover over profile icon | | |
| 2 | Mouse click on “Add Instructors” dropdown option | | |
| 3 | Add Instructor | | |
| 4 | Mouse click on “Save & Submit” | | |
| 5 | Visually confirm the new instructor with “View Instructors” | | |
| 6 | Probe the database to ensure the instructor has been added | | |

## Test Case #6: Edit an Instructor Test – Kiana

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Kiana McDaniel | |
| **Test Case Number** | | 6 | |
| **Test Case Name** | | Edit an Instructor Test | |
| **Test Case Description** | | **This test case will ensure that the user can edit a instructor in the system.** | |
| **Item(s) to be tested** | | | |
| 1 | User Interface’s ability to display all required screens without text overlapping or going off screen | | |
| 2 | Database’s successful storage of non-redundant instructor data | | |
| 3 | Server’s ability to retrieve user input and convert it into instructor data to be saved to database, over-writing pre-existing data | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * User mouse inputs to navigate web-page * User keyboard inputs to fill out instructor information | | | A form will appear that shows data fields filled with the current instructor data. Edits made reflected in the “View Instructors” page. |
| **Resources Required** | | | |
| 1 | Laptop or PC with mouse and keyboard | | |
| 2 | Access to the server directly or via IP/URL | | |
| **Procedural Steps** | | | |
| 1 | Mouse hover over profile icon | | |
| 2 | Mouse click on “View Instructors” dropdown option | | |
| 3 | Mouse click on instructor’s name | | |
| 4 | Edit Instructor | | |
| 5 | Mouse click on “Save & Submit” | | |
| 6 | Visually confirm the edited instructor with “View Instructors” | | |
| 7 | Probe the database to ensure the instructor has been edited | | |

## Test Case #7: Make an Instructor-Course Assignment Test – David

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | David Johnson | |
| **Test Case Number** | | 7 | |
| **Test Case Name** | | Make an Instructor-Course Assignment | |
| **Test Case Description** | | **This test case will ensure that the system can assign an instructor to teach a course, and ensure the system will recommend instructors effectively** | |
| **Item(s) to be tested** | | | |
| 1 | User Interface’s ability to display all required screens without text overlapping or going off screen, and its ability to display any generated warnings | | |
| 2 | Database’s ability to successfully store instructor and course data | | |
| 3 | Server’s ability to retrieve user inputs regarding assignments, provide useful assignment recommendations, and generate appropriate warnings to the user interface | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * User mouse input to navigate webpage and make assignments * Database’s stored course and instructor data | | | * A dropdown list of instructors will be generated in descending order of ideal assignment * Any appropriate warnings will be generated and displayed to the user * The assignment will be reflected on the user interface |
| **Resources Required** | | | |
| 1 | Laptop or PC with mouse and keyboard | | |
| 2 | Access to the server directly or via IP/URL | | |
| **Procedural Steps** | | | |
| 1 | Mouse hover over book icon | | |
| 2 | Mouse click on “View Courses” dropdown option | | |
| 3 | Mouse click on specific course in list | | |
| 4 | Mouse click on specific section of course’s instructor assignment | | |
| 5 | Mouse click on specific instructor from dropdown list | | |

# Testing Schedule – Phillip+Kiana

By the nature of this project and the tests, any of the tests can happen concurrently and are not dependent on each other (since all of them are effectively binary, and the system is programmed in a very uniform method, they will all either work, or they will all fail, and the system will have to be debugged before any are functional)

|  |  |  |  |
| --- | --- | --- | --- |
| Test Dates | Test Case Number | Test Name | Responsible Engineers |
| 2/18/19  -3/30/19 | #1 | View Courses | Kiana + Phillip |
| 2/18/19  -3/30/19 | #2 | Add a Course | Kiana + Phillip |
| 2/18/19  -3/30/19 | #3 | Edit a Course | Kiana + Phillip |
| 2/18/19  -3/30/19 | #4 | View Instructors | Kiana + Phillip |
| 2/18/19  -3/30/19 | #5 | Add an Instructor | Kiana + Phillip |
| 2/18/19  -3/30/19 | #6 | Edit an Instructor | Kiana + Phillip |
| 3/1/19  -4/12/19 | #7 | Make an Instructor-Course Assignment | David |