“D3L”

# ***Test Plan***

##### **Original Plan Date: January Twenty-fifth, Twenty Twenty-one**

##### **Revision Date: February Thirteenth, Twenty Twenty-one**

##### **Revision: 2.0**

## 

# **About this document**

This document constitutes a brief description of the test plan methodologies we plan to employ for ensuring that our project meets the client’s specifications. It contains a brief overview of the project’s objectives and a high-level overview of the system description, followed by an exhaustive listing of features we intend to test, features we have deemed to be out of scope, and how we plan to approach the various phases of software testing.

**Revision History**

|  |  |  |
| --- | --- | --- |
| Revision Number | Date | Comment |
| 1.0 | January 25, 2021 | Original DoIT PMO Document |
| 2.0 | February 13, 2021 | Cut down less prominent testing phases, added end-to-end |
|  |  |  |
|  |  |  |

# **Overview**

## **Project Objectives**

The system under development is meant to provide a communicative and collaborative interface in the form of a web application for usage by students and faculty in the context of an academic setting. The system must facilitate project management, project completion, monitoring of progress, and visualization of project timelines.

**Implementable:**

* Team communication interface(s)
* Data analytic visualizations
* Gamification elements
* Essential user account management functions
* Efficient database schema

## **System Description**

The system is an academically oriented project management tool that must facilitate collaboration and scheduling for both students and faculty. All users must be able to create accounts and communicate with one another, such as via chat rooms or discussion boards. Students must be able to enroll in courses, submit assignments, and track their overall academic progress. Faculty must be able to create course listings, manage course material, and administer grades.

## 

## **Plan Objectives**

**Responsibilities**

* Nick Bashor
* Patrick Grzegorczyk
* Wesam Jabali
* Joey Navarro
* Kevin Sass
* Luigi Yebra

**Test methodologies**

* Unit testing
* Integration testing
* Regression testing
* End-to-end testing

**Test environments**

* End-to-end
* Node.JS
* Vue Plugin
* Browser console

**Project life cycle considerations**

We anticipate that the testing environments and/or testing frameworks that we intend to use will change and evolve as our project’s needs become more well-defined.

## **References**

* [Test Plan/Strategy Template](https://www.doit.state.nm.us/project_templates.html)
* [Learn Vue 3 for Beginners - Full 2020 Tutorial Course](https://youtu.be/ZqgiuPt5QZo)
* [Software Testing Fundamentals - Unit Testing](https://softwaretestingfundamentals.com/unit-testing/)
* [Integration Testing: What is, Types, Top Down & Bottom Up Example](https://www.guru99.com/integration-testing.html)
* [What Is System Testing - A Ultimate Beginner's Guide](https://www.softwaretestinghelp.com/system-testing/)
* [Prototype testing - What, how and why?](https://www.testbirds.com/blog/prototype-testing-what-how-and-why/)

**Test Scope**

## **Features to be Tested**

**Generic Features:**

* Login
* Profile editing & deletion
* Using discussion boards
* Viewing content
* Ability to be added to courses
* Viewing relevant class pages
  + Any classes that the account is associated with/signed up for should be accessible, with no others being shown
* Posting/reading/replying to discussion board posts
  + Anyone should be able to post on a class’s discussion board page, so long as they’re teaching or registered for the class

**Faculty Specific Features:**

* Posting grades
  + Notifications for posted grades to inform students that their work has been appraised
* Posting announcements
  + Notifications are properly sent out to the people that are registered for the class so that they know there is an announcement
* Adding content to class pages
* Replying to student/staff created discussions.

# 

# **Test Methodologies**

## **Testing Approach**

* **All features**
  + **Test types**
    - Unit tests
    - Pilot tests
    - End-to-end
  + **Major activities/techniques/tools**
    - Qualitative interface navigation
    - Database analysis (PostgreSQL)
    - Query generation (Knex.JS)
  + **Methodology**
    - In-house staff (i.e. project members)
    - Ensure database is accessible and properly modifiable
    - Ensure query responses match expectations
    - Qualitatively ensure interface responds appropriately
* **Reliability**
  + **Test types**
    - Pilot tests
    - Stress tests
  + **Major activities/techniques/tools**
    - Database analysis (PostgreSQL)
  + **Methodology**
    - Simulate network downs and/or server restarts
    - Ensure data is preserved in the backend
    - Quantitatively ensure timely recovery
* **Performance**
  + **Test types**
    - Automation testing
    - Integration testing
    - Regression testing
  + **Major activities/techniques/tools**
    - Breeze automation
    - Query response timing
  + **Methodology**
    - Run automated scripts to query endpoints
    - Measure time taken for responses
    - Compare elapsed time to predefined thresholds

**Test Data**

* **Staff**
  + Will be arbitrary
    - Staff ID number
    - First/Last Name
    - Classes being taught.
    - Assignment grades given
* **Student**
  + Will be arbitrary
    - Student ID number
    - First/Last Name
    - Classes being taken
    - Assignment Grades received

## **Test Documents**

* Unit test results (e.g. percentage passed within a test suite)
* Qualitative evaluation forms
  + Integration testing
  + System testing
  + Prototype testing

**Requirements Validation & Control Procedures**

**Student user requirements**

* Record qualitative binary success or failure of essential components
* Cybersecurity concerns (e.g. no XSS, no remote code execution)
* Unit test suite results (automated) must exceed a certain pass percentage

**Faculty user requirements**

* Record qualitative binary success or failure of essential components
* Cybersecurity concerns (e.g. no XSS, no remote code execution)
* Unit test suite results (automated) must exceed a certain pass percentage

**All user requirements**

* Record qualitative binary success or failure of essential components
* Unit test suite results (automated) must exceed a certain pass percentage

**Usability**

* Testing user will fill out an evaluation form while piloting the application
* Usability criteria (e.g. ease of access, readability) ranked 1 to 5
* Total points on evaluation form must exceed a predetermined threshold

**Reliability**

* System uptime will be quantitatively monitored periodically
* System downtime will be calculated as a percentage of system uptime (per period of unit time)
* Ratio between downtime and uptime must exceed a predetermined threshold

**Performance**

* Unit tests that hit application endpoints will be timed upon execution
* Response latency will be recorded and measured in milliseconds
* Latency time must be below a certain predetermined threshold to pass

**Supportability**

* Testing user will fill out an evaluation form while piloting the application
* Basic functional features will be enumerated, each next to a binary checkbox
* User will mark off features that work for each major browser
  + Microsoft Edge
  + Google Chrome
  + Mozilla Firefox

# **Test Phases**

**Unit**

**Definition**

Automated tests written to ascertain proper functionality of discrete and singular application endpoints.

**Participants**

Every member of the team will be involved in the testing process.

**Sources of Data**

Test data not applicable; see entrance and exit criteria.

**Entrance and Exit Criteria**

Unit tests combine to form a particular test suite. Each unit test targets a single application endpoint of unique and relevant functionality. Unit tests output a binary result of either pass or failure; the percentage of tests that pass within a given test suite will be used as evaluation criteria for whether a certain module can be deemed acceptable.

**Requirements**

Unit tests will target basic account functionality, such as addition to class rosters, password creation, and various means of internal data modification. In addition, various non-functional requirements will also be tested, such as performance metrics.

**Work Products**

Test documents and reports produced in this section include a recorded output of the percentage of tests that pass among each individual test suite.

**Test Completion Acceptance**

Unit tests must be done every week, up through production and deployment. They will pass for the given week if at least 85% of the unit tests among each test suite have passed.

**Integration**

**Definition**

Testing of multiple application endpoints and modules to ensure proper functionality when executed in tandem with one another.

**Participants**

Every member of the team will be involved in the testing process.

**Sources of Data**

Dummy account data will be generated randomly.

**Entrance and Exit Criteria**

Integration testing will begin once at least two or more distinct modules have passed the above mentioned unit testing success threshold, at which point those modules will be subjected to manual pilot testing using randomly-generated source data. Requirements will be evaluated both quantitatively and qualitatively.

**Requirements**

Several non-functional, qualitative requirements such as usability, reliability, and supportability will begin testing at this stage. In addition, complex functionality such as account creation and modification, content creation, and security will be tested here as well.

**Work Products**

Test documents and reports produced in this section will include written evaluations of the user experience for all modules under test. Backend concerns such as data persistence and database entity relationships will also be analyzed and noted.

**Test Completion Acceptance**

Integration testing is complete once all basic functionality of all major components of the system (e.g. student, faculty, etc.) have been qualitatively shown to operate on a functional level.

**System**

**Definition**

Following integration testing, the detection of any potential inconsistencies that may have arisen between individual modules and components. Incorporates regression testing.

**Participants**

Every member of the team will be involved in the testing process.

**Sources of Data**

Dummy account data will be randomly generated. Additionally, test data may be manually input by testing participants via the interface in its current state.

**Entrance and Exit Criteria**

System testing will begin once integration testing has completed. Interaction of system components will be qualitatively assessed for proper functionality. The system will be deemed to be in an acceptable state only when all units are consistent in their expected behavior and when all requirements are known to have been met.

**Requirements**

Usability, reliability, and supportability will continue to be tested, as in the integration phase. Functional aspects, ranging from general to student-specific to faculty-specific, will likewise need to pass.

**Work Products**

Test documents and reports produced in this section will include written evaluations of the user experience for all modules under test. Backend concerns such as data persistence and database entity relationships will also be analyzed and noted. In other words, this is similar to integration testing, only larger in scope.

**Test Completion Acceptance**

System testing is complete once basic and intermediate functionality of all major components of the system (e.g. student, faculty, etc.) have been qualitatively shown to operate reliably and robustly.

**End-to-End**

**Definition**

Effective end-to-end testing of a fully functional model for this system. Vertical flow from account creation to course interaction for both student and faculty users will be addressed here.

**Participants**

Someone not related to the project will be using the application making sure that it works overall.

**Sources of Data**

Account and course data will be provided by the testing user(s), as input via the web application’s own interface (must be functional at this point).

**Entrance and Exit Criteria**

Someone from the testing team will act as a professor while the rest are students, and they’ll interact by creating courses, teams and discussions.

**Requirements**

As in previous testing phases, usability, reliability, supportability, and performance must all be tested. All general, student-specific, and faculty-specific functions must likewise operate as expected.

# **Test Environment**

## **Software**

Describe the software requirements for the test environment. Identify automated testing tools, operating systems, compilers, etc.

* Node and NPM
* Breeze testing
  + Java
  + XML
  + YAML
* Primary operating systems
  + Windows
  + Ubuntu Linux
  + Mac OS X
* Modern browsers
  + Microsoft Edge
  + Google Chrome
  + Mozilla Firefox

# **Approvals and Distribution**

Professor Muscarello will be the one approving our test plan. This test plan will be distributed to the professor, as well as every member of our team.