D3L

Project Development Manual

Original Plan Date: March First, Twenty Twenty-one

Revision Date: March First, Twenty Twenty-one

Revision: 1.1

About this Document

This document, the **Project Development Manual**, is a comprehensive collation of all relevant system specification documents developed in relation to the D3L application over the course of prototyping, implementation, and designing.

Revision History

Revision Number	Date	Comment
1.0	March 1, 2021	Original DoIT PMO Document
1.1	March 9, 2021	Added User Manual

3. Requirements Analysis

3.1 Introduction

1. Purpose of the system

The D3L system 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.

2. Scope of the system

The system is intended to be made available to all students enrolled in, as well as all faculty employed by, a singular academic institution. The system must manage an arbitrary number of users (both students and faculty members) as well as an arbitrary number of academic courses.

3. Objectives and success criteria of the project

3.1. Objectives

3.1.1. Implement team communication interface(s)
3.1.2. Implement essential user management functions
3.1.3. Implement an efficient database schema

3.2. Success criteria

3.2.1. Visualizations are easy to understand
3.2.2. Application uptime is continuous and reliable
3.2.3. Response time exhibits minimal latency
3.2.4. Application is scalable
3.2.5. User interface is intuitive and accessible

4. Definitions, acronyms, and abbreviations

4.1. Definitions

4.1.1. Refer to the **Glossary** for more information

4.2. Acronyms

4.2.1. **CSS** - Cascading Style Sheets
4.2.2. **DBMS** - Database Management System
4.2.3. **GUI** - Graphical User Interface
4.2.4. **JS** - JavaScript
4.2.5. **NPM** - Node Package Manager
4.2.6. **SQL** - Structured Query Language

5. Overview

5.1. Function of system

5.1.1. Admin/Faculty

5.1.1.1. Create course listings
5.1.1.2. Manage course material
5.1.1.3. Administer grades
5.1.1.4. Create teams
5.1.1.5. Add users to teams

5.1.2. Students

5.1.2.2. Submit assignments
5.1.2.3. Track academic progress

5.1.3. All users

5.1.3.1. Communicate with other users via discussion

5.2. Reason for development

Educational institutions worldwide have seen a large increase in the usage of digitally-administered academic management tools. We likewise aim to develop an online web application that implements common features of such digital academic aids, with the hopes of providing a clean user experience coupled with a sufficiently robust data management infrastructure.

3.2 Current system

Frontend and backend are both currently operational and have a wide variety of functionality implemented. The frontend is implemented with Vuetify using Vue 2, and the backend is implemented via PostgreSQL, Knex.JS, and Express.JS.

The following components are presently functional:

1. **General**
    a. View differentiation according to user roles
        - Administrative functionality
        - Faculty-specific functionality
        - Student-specific functionality
    b. Responsive web view
    c. Responsive mobile view
    d. Binary file uploading

2. **Accounts**
    a. User account creation
    b. User account deletion
    c. User login
    d. Backend authentication
    e. Secure session management

3. **Courses**
    a. Course creation
    b. Course registration
    c. Course viewing

4. **Content**
    a. Content creation
    b. Content viewing
    c. Grade assignment

5. **Teams**
    a. Team creation
    b. Team member assignment
    c. Team viewing

3.3 Proposed system

1. **Overview**

   The system under development is intended to be a tool for usage by educational institutions to facilitate project management for and collaboration between students and faculty members.

2. **Functional requirements**

   2.1. Student users

2.1.1. Easy addition to class roster(s)
2.1.2. Password modification and authentication
2.1.3. Restricted view according to privileges

2.2. Admin/Faculty users

2.2.1. Class section creation
2.2.2. Class content creation
2.2.3. Team creation and modification
2.2.4. Grade administration and access privileges

2.3. All users

2.3.1. Account creation
2.3.2. Account deletion
2.3.3. Login and logout functionality
2.3.4. Responsibility/agreement opt-in functionality
2.3.5. Password modification and authentication
2.3.6. Security
2.3.7. Communication

3. Nonfunctional requirements

3.1. Usability

3.1.1. Effective visualizations
3.1.2. Simplicity and ease of use

3.2. Reliability

3.2.1. System will be available at least 95% of the time
3.2.2. System can recover from unexpected downtime

3.3. Performance

3.3.1. System is scalable
3.3.2. System is timely and responsive

3.4. Supportability

3.4.1. Microsoft Edge
3.4.2. Mozilla Firefox
3.4.3. Google Chrome

3.5. Implementation

3.6.1. Vue.JS (Vue 2)
3.6.2. Vuetify
3.6.3. Express.JS
3.6.4. PostgreSQL
3.6.5. Knex.JS

3.6. Interface

3.6.1. User interface is easily navigable
3.6.2. Basic features must not be concealed from users
3.6.3. Relevant information must be reachable and readable
3.6.4. Important prompts/reminders should be made explicit

4. System models

4.1. Scenarios

1. Future students register via home page.
2. Faculty registration in tandem with administrator
3. Class, schedule interfaces visible to students/faculty
4. Admin can register faculty and student accounts
5. Mitigate SQL injection and cross-site scripting.

3.4 Glossary

accessibility - a frontend design concern that addresses usability in the context of differently-abled users of the system

account - a data entity that encapsulates all identifying information associated with a single and unique user of the system

browser - a software application that allows the user to access the Internet; more specifically, in the context of this system, the means by which a user will interact with this project

backend - developmental entities related to the system's internal logic, such as maintaining, processing, and retrieving records; not meant to be accessible by the user

faculty - a user associated with an employee of the university (or similar academic institution) that administers this system; a non-student, usually bearing elevated privileges (see **student**)

frontend - developmental entities related to the system's outward-facing logic, such as chat, account management, and course management interfaces; the portion of the system with which the user interacts

GUI - acronym for graphical user interface; see **interface**

interface - the graphical and textual components of the system with which the user interacts, ultimately bridging external input with internal logic; see **frontend**

login - the functionality that enables a user to establish a unique online session via the system, during which they may access their account, view coursework-related materials, etc.

logout - the functionality that enables a user to safely and securely terminate an online session that was previously established via the login process

password - the cryptographic means by which a user may securely verify their identity in order to log into their account

performance - a functional metric that measures how efficiently user queries are transmitted and processed by the system, typically in terms of units of time

reliability - a functional metric that evaluates how consistently and continuously the system is available to provide its services, typically expressed as a percentage of uptime

roster - a list of all student and faculty users who bear access to a single given course

schema - the formal organizational structure behind the system's internal representation of data entities

student - a user associated with an individual enrolled in the university (or similar academic institution) who primarily takes courses but does not instruct them; bears limited privileges (see **faculty**)

supportability - a functional metric that measures the range of compatibility the system has across modern Internet browsers; see **browser**

system - the project currently under development that this document describes; a comprehensive, user-oriented project management tool, collaborative engine, and general-purpose academic aid

usability - a functional metric that measures how intuitive or unintuitive the user experience can be evaluated when said user interacts with the system

4. System Design

4.1 Scope

4.1.1 Software

- Vue.JS (Vue 2)
- Express.JS
- PostgreSQL
- Knex.JS

4.1.2 Audience

Users/readers must have a fair understanding of code, having at least a bit of experience with or some knowledge involving Vue.JS, Express.JS, PostgreSQL, & Knex.JS.

4.1.3 Related Documentation

Right now we have three other documents that we have already filled out. Provided below are clickable links to all of them:

- D3L Test Plan Document
- D3L Requirements Analysis Document
- D3L Use Case Document
- D3L System Deployment Plan Document
- Templates for Documentation

4.2 System Overview

4.2.1 Description

The purpose of this system is to manage the relationship between students, faculty, and administration in a university. It has the ability to:

- Manage users
    - Create user IDs and faculty IDs
- Manage courses
    - Courses are organized in alignment to DePaul's course number format.
    - Students are assigned to courses
    - Teams are assigned to courses
    - Content is assigned to courses
    - Manage discussion forums per course
- Manage content

    - Content can be any file upload or text submission by faculty
    - Content can be graded or ungraded
    - Manage discussion forums per piece of content
- Manage teams
    - Students can be assigned to teams
    - Teams assigned to courses
    - Teams can have group submissions, as used in D2L.

4.2.2 System Architecture

Software Architecture

This section outlines the software architecture established for the project. Provide references to the System Architecture Document and a brief summary of the software architectures.

4.3 Hardware Design

4.3.1 Computer Systems

- Netlify
- Heroku

4.4 Software Design

4.4.1 Software Packages

- **Frontend**
    - **components/**
        - **Admin/**
            - **AddCourse.vue**
            - **AddRole.vue**
            - **AdminBar.vue**
            - **NewCourse.vue**
        - **Faculty/**
            - **AddTeam.vue**
            - **NewContent.vue**
            - **NewTeam.vue**
        - **User/**
            - **DeleteAccount.vue**
            - **NewDiscussion.vue**

                - **ViewContent.vue**
                - **ViewDiscussion.vue**
            - **HeaderComponent.vue**
            - **Snackbar.vue**
        - **plugins/**
            - **Axios.js**
            - **Vuetify.js**
            - **Snack.js**
        - **router/**
            - **Index.js**
        - **store/**
            - **Index.js**
        - **views/**
            - **Course.vue**
            - **Home.vue**
            - **Login.vue**
            - **Playground.vue**
            - **Profile.vue**
            - **Register.vue**
        - **App.vue**
        - **Main.js**
- **Backend**
    - **database/**
        - **migrations/**
            - **Starting_schema.js**
        - **seeds/**
            - **01_d3l_user.js**
            - **02_d3l_user_role.js**
            - **03_d3l_course.js**
            - **04_d3l_user_course.js**
            - **05_d3l_team.js**
            - **06_d3l_user_team.js**
            - **07_d3l_content.js**
            - **08_d3l_user_content.js**
            - **09_d3l_discussion_post.js**
        - **sqlScripts/**
            - **Starting_schema_down.pgsql**
            - **Starting_schema_up.pgsql**
    - **knex.js**
    - **middleware/**

        - **authMiddleware.js**
        - **cors.js**
        - **rolesMiddleware.js**
    - **Node_modules**
        - **Too many...**
    - **public/**
    - **routes/**
        - **app/**
            - **admin/**
                - **Course.js**
                - **Index.js**
                - **user.js**
            - **faculty/**
                - **Content.js**
                - **Course.js**
                - **Discussion.js**
                - **Index.js**
                - **Team.js**
                - **user.js**
            - **user/**
                - **Content.js**
                - **Course.js**
                - **Discussion.js**
                - **Index.js**
                - **Team.js**
        - **Index.js**
        - **non_auth.js**
    - **Index.js**
    - **.env**
    - **App.js**
    - **Knexfile.js**
    - **Package-lock.json**
    - **Package.json**
    - **Profile**

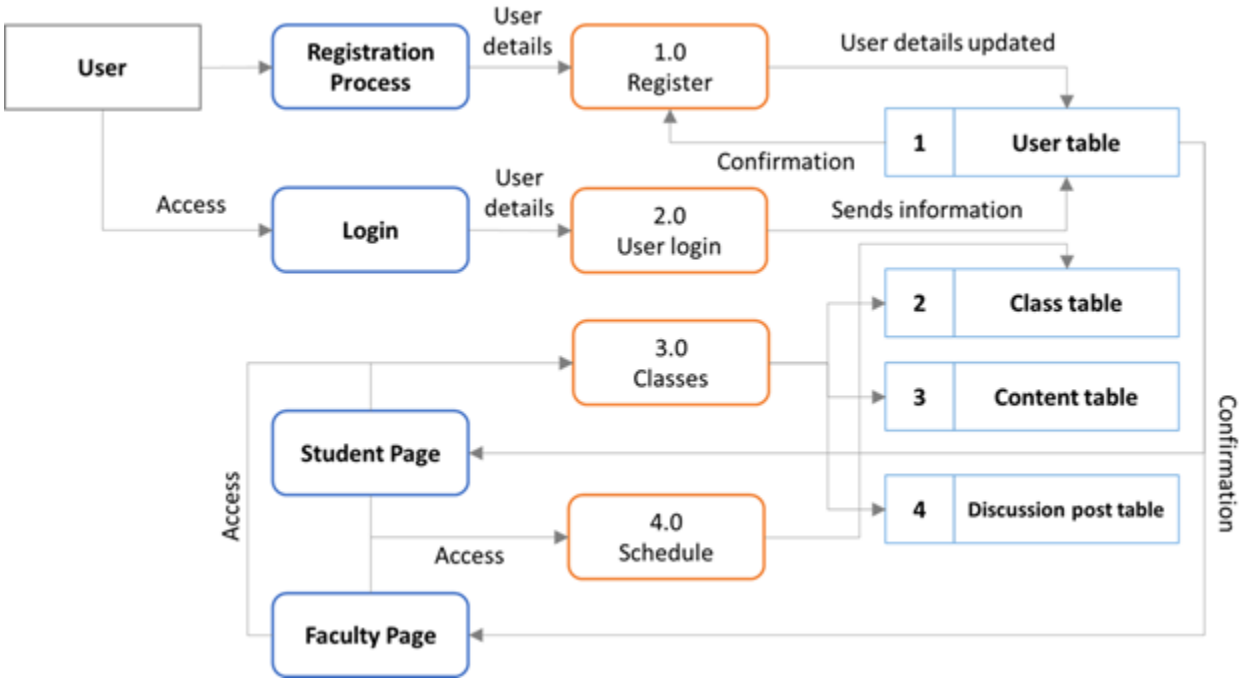
4.4.2 Software Integration

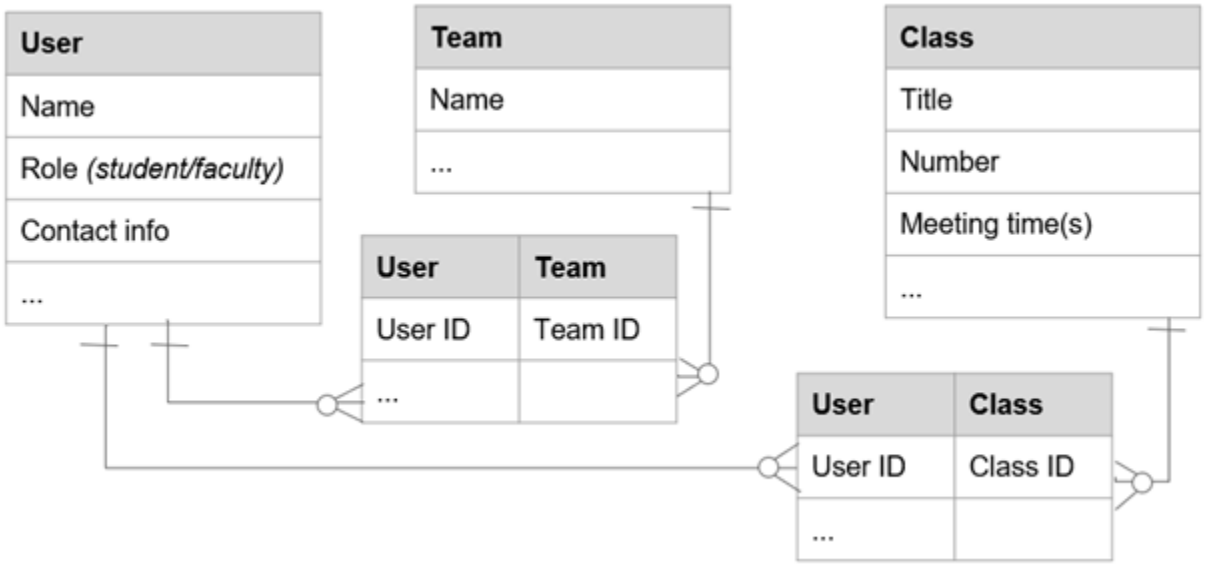
- **Vue.JS** - Frontend framework
- **Node.JS** - Backend framework
    - **Express.JS** - Web application framework

- **Knex.JS** - Querying middleware

4.5 Data / Database / Files

4.5.1 Data Flow Diagrams





4.5.1 Data Flow Diagrams (cont'd)

Content
Class ID
Title
Description
File URL *(optional)*
Is graded? *(True/False)*
Grade *(optional)*
...

Discussion Post
Content ID *(optional)*
Class ID *(optional)*
Parent ID *(optional)*
User ID
Title
Post content
...

- If **Content ID** is non-null, the discussion post belongs to a content item
- If **Class ID** is non-null, the post belongs to a general class discussion
- If **Parent ID** is non-null, the post is a response to another parent (root) post

4.5.2 Database Design

List and describe tables, fields, and entity relationships (also known as data dictionary and logical/physical database design), schema, query language, key and indices, data management functions.

- **Query language:** PostgreSQL
- **Data management**
  - Querying via Knex.JS
- **Tables**
  - User
    - Timestamp
    - ID
    - First name
    - Last name
    - Email address
    - Password (encrypted)
    - Phone number
    - Home address
  - Course
    - Timestamp
    - ID
    - Title
    - Course prefix
    - Course number
    - Section number
  - Team

        - Timestamp
        - ID
        - Team name
        - Course ID (foreign key to Course)
    - Content
        - Timestamp
        - ID
        - Course ID (foreign key to Course)
        - Title
        - Body
        - File URL
        - Graded
        - Points earned
        - Total points
    - Discussion Post
        - Timestamp
        - ID
        - User ID (foreign key to User)
        - Parent ID (foreign key to Discussion Post)
        - Content ID (foreign key to Content)
        - Title
        - Body
    - User-to-Team
        - Timestamp
        - ID
        - User ID (foreign key to User)
        - Team ID (foreign key to Team)
    - User-to-Course
        - Timestamp
        - ID
        - User ID (foreign key to User)
        - Course ID (foreign key to Course)
    - User-to-Role
        - Timestamp
        - ID
        - User ID (foreign key to User)
        - Role

4.5.3 Files

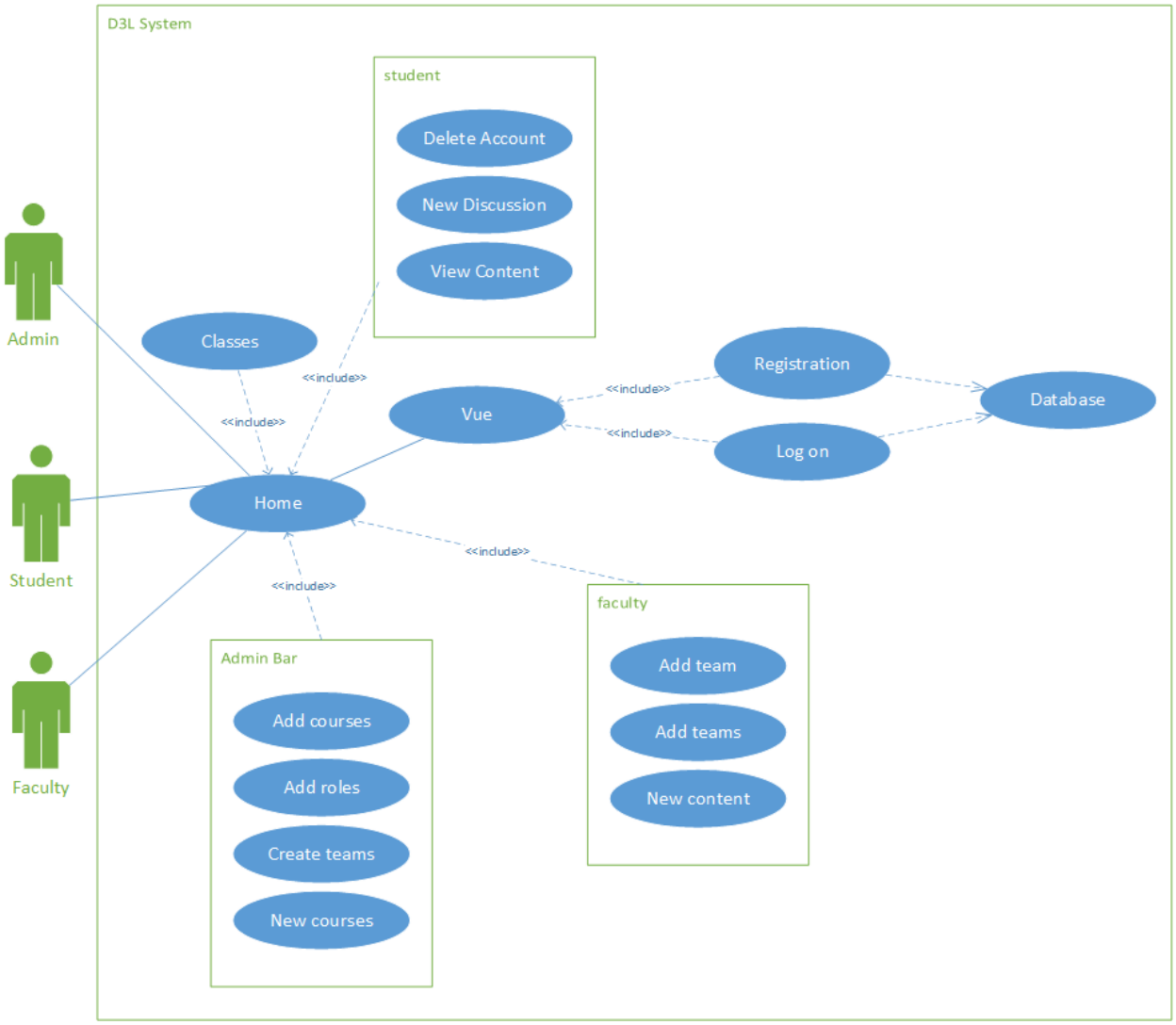
- `root/`
    - `database/`
        - `migrations/`
            - `starting_schema.js`
                - Identifies startup PostgreSQL script
                - Identifies shutdown PostgreSQL script

- `seeds/`
  - `01_d3l_user.js`
    - Defines admin user
- `sqlScripts/`
  - `starting_schema_down.pgsql`
    - PostgreSQL script to execute when shutting down
  - `starting_schema_up.pgsql`
    - PostgreSQL script to execute when starting up
- `knex.js/`
  - Querying framework

5. Use Cases

5.1 Actors, Goals, and Use Case Briefs

Actors	Task-Level Goal	Priority	Brief
Students	• Setup page	• Medium	• See the classes • New Discussion • View Content
Faculty	• Setup page	• Medium	• Add Team • Create new content • New Team
Admin	• Setup page	• Medium	• Add roles • Create course • Add courses



5.2 Use Case Specifications

Title Log On

Main Goal	• To log in to a specific account

Actors Involved	• Students • Faculty • Admin
Props	• Database • Log on Page
Action	• Front end will communicate with the backend then the backend will verify if the username and password are within the database.
End State	• Display either Student or faculty or Admin page

Title Registration

Main Goal	• To create an account • Add it to database
Actors Involved	• Students • Faculty • Admin

Props	• Database • Registration Page
Action	• After the Registration page acquires the necessary information, It will pass the info to the backend and add to the database
End State	• Return to the Log on Page

Title Admin Bar

Main Goal	• Create New Course • Add Role • Add Courses • Create Team
Actors Involved	• Admin
Props	• Database • Home page • course card

	• team card • roles card
Action	• Front end will communicate with Back end, and back end will verify if user and password is within the database
End State	• Display admin page

Title Student

Main Goal	• provide student features
Actors Involved	• Students
Props	• Database • Student buttons
Action	• Display student's buttons
End State	• Display student page

Title	Faculty
Main Goal	• Show faculty functionalities
Actors Involved	• Faculty
Props	• Database • Home page • Add team card • Create team card
Action	• Gives the faculty member to view his or her classes • Faculty can create team based on student roster • Faculty can create a course
End State	• Display faculty page format

6. Test Plan

6.1 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

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
- Learn Vue 3 for Beginners - Full 2020 Tutorial Course
- Software Testing Fundamentals - Unit Testing
- Integration Testing: What is, Types, Top Down & Bottom Up Example
- What Is System Testing - A Ultimate Beginner's Guide
- Prototype testing - What, how and why?

6.2 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.

6.3 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

6.4 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

6.5 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.

6.6 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

6.7 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.

7. User Manual

7.1 Login Page

The login page is the landing page for D3L, and is a simple page. It contains the fields for the user's email and password, both of which are required to login.

There is a "Login" button that can be clicked to login, or the user can hit the "Enter" key for the same effect.

The other button on this page is the "Register" button. By clicking the register button, the user is able to access a page to register a user account which can then be used to successfully login upon completion.

7.2 Register Page

The register page consists of eleven fields asking the user for their relevant information. These fields include:

- E-Mail
- Phone Number
- Password
- Password Confirmation
- First Name
- Middle Name
- Last Name
- Street Address
- City
- State (with a drop-down list of all the states)
- Zip Code

All of these fields are required, with some having limits on the amount of characters that can be entered in each field. The "Confirm Password" field also correctly checks to make sure that the password is inputted the same in both fields.

Once all fields have been filled out, the user will successfully be able to register their account for use back on the login page.

7.3 Home Page

After the user has logged in, they will be taken to the home page.

This page (and every page beyond the login page) will have a navigation bar at the top. The “D3L” button on the left side of the bar can be used to take the user back to this home page.

On the right side of this bar, there is a button displaying the user’s first and last name. When clicked, it opens up a drop-down menu that allows the user to either visit their profile page or log out of D3L.

7.3.1 Faculty/Student Home Page

The home page will display any classes that the user has signed up for and added to by an administrator, (or classes that the user is teaching in the case of faculty). These classes will be displayed in a box with a large random picture (pulled from the internet), along with the course name under it. Below that, the course prefix, number, and section are also displayed.

7.3.2 Admin Home Page

The admin homepage looks exactly the same as the homepage for faculty and students outside of one big difference. Admin accounts have access to the admin toolbar, which is a black toolbar below the navigation bar, which has three buttons on it.

The first is the “Add New Course” button, which allows an admin to create a brand new course in the database. When clicked, it pops up a window which contains required fields for the new course’s title, prefix, number, and section. When all this is done, the admin can click the submit button to successfully create a new course.

Next is the “Add Course” button, which opens a window that is used to add users to certain courses. Both the User and Course fields have drop-down menus containing a list of users and a list of the current existing courses. When selecting courses, multiple may be selected and added for a user at one time. Once again, the submit button will add the user to whatever courses have been selected.

Finally, we have the “Add Role” button that opens a window which simply allows for the admin to add roles to the selected user. Both the User and Roles fields here are also drop-down menus, which contain a list of users and a list of roles

respectively. One or both of the roles can be selected and added at the same time. The roles will be added upon clicking the submit button or pressing the “Enter” key.

All of these windows can be cancelled out of by clicking “Cancel” or by clicking out of the window. In addition, when successfully or unsuccessfully submitting with one of the buttons, an appropriate message will pop-up indicating whether or not the admin was successful with their action(s).

7.4 Profile Page

The profile page simply displays all of the information the user submitted when filling out the fields for registration. In addition to this, there is a “Edit” button that changes the text fields to be editable for the user information. Upon clicking the edit button, a different set of buttons appears in its place.

The first, “Delete Account” opens up a prompt when clicked asking the user if they’re sure they want to delete their account, and will successfully delete the user’s account if the user clicks “Delete Account” in this prompt’s window. This of course, logs the user out as their account no longer exists.

Next we have the “Save Changes” button, which basically acts as a button to be clicked to update the user’s information in their profile after changing any of the now editable text fields.

Finally, there is the “Cancel” button, which takes the user out of editing mode and back to the viewing mode of the user’s profile page.

7.5 Course Page

Upon clicking a course on the home page, the user is taken to the respective course’s page. There will be a black toolbar under the navigation bar, similar to the one that admins have on the home page.

Below the course title and information, (which is under the toolbar), are three sections. One for course content, one for course discussions, and one for students in the class. Content, discussion posts, and students will be listed under these three headers with the appropriate information in each of their boxes.

7.5.1 Student Course Page

For students, the black toolbar only has a “New Discussion” button, which pulls up a window with fields for a title and body for a new discussion post to be posted under the Discussions section at the bottom of the column.

The Content section shows graded and ungraded content for the user, and when content is clicked it will show a description of the content as well as the point total (if the content is graded or graded content). It will also allow for the user to download the content to their device if the content is available.

For the Discussions section, when a discussion post is clicked, a window is opened that contains every response to the discussion post with clear indentation to show which replies are to which. If the user clicks on a post/reply, a window appears that allows for the user to reply with a title and message of their own. This reply is then added on to that respective discussion post. To leave the discussion post window, the user can either click the “Done” button at the bottom, or click off of the window.

Finally, the last column of the three is for Team Members. If the user is part of a team for the course, any teammates they have will be listed here, with their email and phone number displayed for teammates to use to contact them.

7.5.2 Faculty/Admin Course Page

The faculty and admin variant of the course page has a ton of similarity to the student version, but there are a couple differences and also some additions.

For the Content section, when clicking on graded content items, there will be a drop-down menu with a list of users in the course and ability to type in a number of points for submission as that user’s grade. If the grade has already previously been submitted, it will display the user’s percentage and points for the content.

The rightmost column for faculty and admins is a list of students for the course instead of a list of teammates, as the faculty/admins aren’t going to be on a team normally.

As far as new things go, faculty and admins have new buttons to access on the black toolbar at the top of the page. There is a “New Content” button and a “Manage Teams” button that are available for use.

The “New Content” button, when clicked, brings up a window that contains fields for a title, body, file attachment, and a number of points. If the content is intended

to be ungraded, the field can be left blank when submitting. Upon successful submission, this adds a content item to the column below on the course page.

As for the “Manage Teams” button, it reveals a drop-down menu with the choices “Create New Team” and “Add Team Member” when clicked. “Create New Team” opens up a window that just asks for a team name to create, with no other information needed. The “Add Team Member” option pops up a window that has a user and team field that have drop-down menus containing a list of the current users in the course and a list of the created teams for the course. One user can be added to a team at a time using the submit button.