# D2L (Desire 2 Learn) 3rd Party App

Class/Section - 4850-02

Fall Semester 2023

Date: 08/31/2023

 Sean Work

 Christopher Dymanus

 Kenneth Molinari

**A Software Design Specification Template**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Sean Work, Kenneth Molinari, Christopher Dymanus*

*09/16/2023*

Table of Contents

[D2L (Desire 2 Learn) 3rd Party App Cover Page 1](#_Toc146018094)

[1. Introduction 4](#_Toc146018095)

[1.1. Purpose 4](#_Toc146018096)

[1.2. Scope 4](#_Toc146018097)

[1.3. Intended Audience 4](#_Toc146018098)

[1.4. Associated Documentation 4](#_Toc146018099)

[1.5. Summary 4](#_Toc146018100)

[2. Design Considerations 4](#_Toc146018101)

[2.1. Assumptions and Dependencies 4](#_Toc146018102)

[2.2. General Constraints 5](#_Toc146018103)

[2.3. Goals and Guidelines 5](#_Toc146018104)

[2.4. Development Methods 5](#_Toc146018105)

[3. Architectural Strategies 6](#_Toc146018106)

[3.1. Programming Language 6](#_Toc146018107)

[3.2. Reuse of Components 6](#_Toc146018108)

[3.3. Future Enhancement 6](#_Toc146018109)

[3.4. User Interface 6](#_Toc146018110)

[3.5. Error Handling 6](#_Toc146018111)

[3.6. Data Storage 6](#_Toc146018112)

[4. System Architecture 6](#_Toc146018113)

[4.1. Overview 6](#_Toc146018114)

[4.2. Components/Subsystems 7](#_Toc146018115)

[4.3. Component Collaboration 7](#_Toc146018116)

[4.4. Rationale for Decomposition 8](#_Toc146018117)

[4.5. Subsystem Architecture 8](#_Toc146018118)

[5. Policies and Tactics 8](#_Toc146018119)

[6. Detailed System Design 9](#_Toc146018120)

[6.1. Classification 10](#_Toc146018121)

[6.2. Definition 10](#_Toc146018122)

[6.3. Responsibilities 10](#_Toc146018123)

[6.4. Constraints 10](#_Toc146018124)

[6.5. Composition 10](#_Toc146018125)

[6.6. Uses/Interactions 10](#_Toc146018126)

[6.7. Resources 10](#_Toc146018127)

[6.8. Processing 11](#_Toc146018128)

[6.9. Interface/Exports 11](#_Toc146018129)

[6.10. Detailed Subsystem Design 11](#_Toc146018130)

[7. Glossary 11](#_Toc146018131)

[8. Bibliography 12](#_Toc146018132)

# Introduction

## Purpose

This document serves as the Software Design Specification (SDS) for the 3rd party D2L Project Planning Application. The purpose of this document is to elaborate on the system architecture, design patterns, data models, and other technical aspects related to the implementation of the application. This document will aim to guide the software development team and serve as reference for future modifications and maintenance efforts

## Scope

The SDS covers the architectural design, data model design, detailed design, and security aspects of the D2L Project Planning Application. It focuses exclusively on the design and technical elements of the project.

## Intended Audience

This document is meant for software developers, testers, project managers, and stakeholders involved in the development of the D2L Project Planning Application.

## Associated Documentation

* + Software Requirements Specification (SRS) for D2L Project Planning Application
  + Project Plan for D2L Project Planning Application
  + TBD if there needs to be more.

## Summary

The remainder of this document is divided into several sections detailing the system architecture, design considerations, and data models. Each section serves to expand and illuminate the underlying systems that will allow the application to meet the requirements as outlined in the Software Requirements Specification (SRS) document.

# Design Considerations

## Assumptions and Dependencies

This (D2LPP) Application is primarily built utilizing Visual Studio 2022 IDE and coupled with .NET 7 Framework for both front and back-end development. Microsoft SQL Management Studio will serve as the primary database tool to store our data. The application will also rely on Azure web services and Git for hosting and repository version control. It is assumed that all these technologies will provide a tight cohesive environment that will be implemented with minimal issue.

Our main given assumption is that our web application will be OS agnostic and will function on any platform that has a compatible web application. The only dependency we have as of writing is the usability for both sets of users as that will be defined later in development depending on what is feasible for the prototype.

## General Constraints

The (D2LPP) Application is built to function in a web environment that has its server-side logic both maintained and constrained by the .NET 7 Framework which we assume might be limited in the future if new updates come about. The same can be said about the current data management provided by the Microsoft SQL Management Studio.

## Goals and Guidelines

* + *User-Friendly UI*

The primary goal here is to ensure that both the students and the faculty have a simple, intuitive and easy-to-use interface that will provide a massive improvement compared to the current system. Tasks will be completed efficiently with minimal clicks and data retrieval will be uncomplicated.

- *Security*

The security of our application is paramount as we do not want to

introduce any vulnerabilities to the D2L Platform or the educational institution our product will be hosted on. Since our product does not have access to the API as of this moment, we will instead create our own secure login with two-factor authentication along with encrypted data transmissions.

* + *Flexibility/Scalability*

The architecture for our application will be modular and scalable with the confines of the tech stack we are using to implement it. The application will meet current requirements but also be flexible enough to add future updates, modify existing features and add new ones if desired.

## Development Methods

The development methodology we will use at least tentatively will be the AGILE method as this will allow us to make ongoing changes while being flexible enough to adapt to new or changing requirements. This was chosen as the best option for a small development group that will more than likely need to work at varying speeds to accomplish the final product.

# Architectural Strategies

## Programming Language

* + - *.NET 7 and C#*

Ease of development, scalability and Azure integration.

## Reuse of Components

* + - *Use of open-source libraries (TBD)*

Speeds up development and ensures reliability.

## Future Enhancement

* + - *Modular Architecture*

Simplifies future updates

## User Interface

* + - *Single-Page Application(SPA) TENTATIVE*

Enhanced User experience with snap response to the necessary section.

## Error Handling

* + - *Complete Logging and Exception-Handling*

Immediate detection and quick resolution of issues.

## Data Storage

* + - *Microsoft SQL Management Studio*

Best option for our data management and query needs.

# System Architecture

## Overview

The system (D2LPP) is a 3rd Party Web Application that will be made available on the D2L Marketplace and/or a freeware prototype that will have no connection due to lack of access to the D2L API. The product will aim to serve both students and faculty members to facilitate project management, team creation, surveys, statistics and other academic needs.

(D2LPP) will take on a few significant responsibilities highlighted below:

* + User Authentication: To securely authenticate Students, Faculty and Admins.
  + Project and Team Management: To allow faculty to create and manage projects and teams.
  + Survey Management: To create surveys that can be distributed.
  + Notification System: To be able to shoot out notifications in real time to all students on a project or all students on a team.
  + Data Analysis: To display data related to ongoing projects, teams, surveys and deadlines.

## Components/Subsystems

The system architecture has been broken down into the following top-level components.

*Frontend*:

Responsibilities – User Interface (UI), Client-side validation, and User experience.

Utilized Technologies – Microsoft Visual Studio 2022 with .NET 7 Framework

*Backend*:

Responsibilities – Business logic, processing of data, API management.

Utilized Technologies – Microsoft Visual Studio 2022 with .NET 7 Framework

*Database:*

Responsibilities – Data storage, Retrieval of data, Management of data

Utilized Technologies – Microsoft SQL Management Studio (Latest)

*Cloud Services:*

Responsibilities – Hosting the web application, CI/CD, Repository Management.

Utilized Technologies – Azure Web Services

*Security and Communication:*

Responsibilities – Encryption, Security, Data integrity for email, database, and web.

Utilized Technologies – SMTP Protocol, SSL/TLS encryption, IP Address

## Component Collaboration

Details of how our processes will communicate with one another.

*Frontend and Backend*: The frontend will communicate with the backend through API calls. The backend will process data and return output to the frontend.

*Backend and Database*: The backend will make queries to the database to fetch/store data.

*Cloud Services and Backend:* Azure will host the backend as well as the entire project repository to enable CI/CD.

*Security and Communication:* Will ensure that all components are secure and capable of encrypted communication.

## Rationale for Decomposition

The system is broken down into these parts namely because they fit with modern frameworks and align with current software development practices. This will favor compartmentalization, scalability and maintainability which in contrast is harder but ultimately a better long-term choice to a more monolithic structure which might produce problems down the road.

## Subsystem Architecture

The complexity and security demands of the authentication system warrant a more detailed discussion which will be provided in 6 - Detailed System Design section of this SDS document. Although it mainly breaks down into:

*Login Manager:* Will handle the verification of the username and password.

*Role-Based Access Control:* Will manage the different functionalities/privileges accessible to students/faculty and admins.

By following this architecture, the system aims to provide a comprehensive, scalable, and secure solution for project planning/management in online educational settings.

# Policies and Tactics

This section will describe the various design policies and tactics utilized in the interface of our systems and overarching components that make up the D2LPP application. These policies do not have sweeping architectural implications but do significantly affect the details of said implementations.

## Compiler and Development Environment

*Decision:*

We have chosen to use Microsoft Visual Studio 2022 in tandem with the .NET 7 Framework for both front and backend development.

*Alternatives Considered:*

We considered VS Code, JetBrains and other applicable tech stacks.

*Rationale:*

Visual Studio provides an all-encompassing solution that will seamlessly communicate with Azure Web Services, has built in functionality for the .NET 7 Framework and contains all the extensive debugging tools we would need to accomplish the D2LPP Application. Other IDE’s fell short in terms of their required setup to integrate the other technologies and ultimately this design choice was made based upon the experience of the developers.

## Database Management

*Decision:*

Microsoft SQL Management Studio (Latest) was chosen as the database management tool.

*Alternatives Considered:*

MySQL and PostgreSQL were considered.

*Rationale:*

SQL Management Studio seamlessly integrates with both Azure Web Services and Visual Studio while offering the necessary toolset for our project. There might have been better options but this one seems to be the simplest to implement based on our limited experience.

## Coding Guidelines and Conventions

*Decision:*

We will follow Microsoft’s official C# coding conventions and best practices when coding. We will also attempt to be flexible since our developers are not as well versed in the C# language.

*Alternatives Considered:*

None considered

*Rationale:*

We want to make sure that the coding style of each of our developers is easily readable and translatable to one another. Ultimately the goal for the code we produce is to be maintainable long after we are no longer working on the project.

## Algorithm Choice

*Decision:*

(Tentative) For authentication, we will employ JSON Web Tokens for token-based authentication.

*Alternatives Considered:*

OAuth2 was considered.

*Rationale:*

JSON Web Tokens is stateless and can store information the token itself. This makes it suitable for our use case especially our prototype if we don’t gain access to the API.

## Communication Protocols

*Decision:*

SMTP protocol with SSL/TLS encryption has been the method chosen for email communication.

*Alternatives Considered:*

None considered.

*Rationale:*

SMTP is widely used and well-supported. Since we will be dealing with potential sensitive information it will be necessary to provide encrypted communication and this will suffice.

## Requirements Traceability

*Decision:*

Each component and feature will be documented and linked back to the original SRS document.

*Alternatives Considered:*

None considered

*Rationale:*

This will aid in future debugging efforts and will ensure that we have documentation in place to compare the product with the original requirements.

## Testing Plan

*Decision:*

(Tentative) Unit Tests will be created using xUnit for .NET.

*Alternatives Considered:*

None considered

*Rationale:*

This option is natively supported in Visual Studio which will allow us to have quick and efficient testing cycles.

## Maintenance Plan

*Decision:*

We will utilize Git/GitHub to implement semantic version control along with basic and regular code reviews.

*Alternatives Considered:*

None considered

*Rationale:*

This will over a simplified way to implement a structured maintenance plan that will allow for immediate roll out updates or rollbacks if necessary.

## User Interface

*Decision:*

We have chosen to use Bootstrap CSS Framework to style our UI.

*Alternatives Considered:*

We considered vanilla CSS and Tailwind

*Rationale:*

This choice was done simply on knowledge base between the developers and the fact that any of the options could provide a modern look that will integrate into our .NET product.

## Source Code Organization

*Decision:*

We will use a traditional hierarchy of folders based on components and features.

*Alternatives Considered:*

None considered

*Rationale:*

This will enhance code maintainability and ultimately will not create confusion between developers by applying unfamiliar practices with alternatives.

## Build and Generation

*Decision:*

We have chosen to use Azure DevOps for CI/CD.

*Alternatives Considered:*

None considered

*Rationale:*

Since this tool is already made available with Azure Web services and integrates with Visual Studio it makes sense to implement this versus other alternatives.

# Detailed System Design

For the time being this will be relatively scarce since we have not coded anything as of yet.

## Classification

The components for this project will mostly be modules within a .NET7 Framework and classes for handling various functions such as survey creation, team creation, deadline scheduling, and numerous others.

## Definition

The modules and classes are designed to fulfill the software requirements as specified in the SRS. The aim is to facilitate students and teachers with project planning and management.

## Responsibilities

Each model/class will be responsible for a certain subset of features.

*Survey Creation:* Generating, editing, and managing surveys.

*Team/Project Creation:* Handles the creation, editing, modifying and management of teams/projects within a defined class session.

*Deadline Scheduler:* Generating, editing, and managing deadlines on a synced calendar.

*Archive:* Manage the archived projects in a repo.

*Notification Handler:* Generating, editing, and managing notifications sent out to student users.

*Data Display:* Generating and managing reports from projects/classes/teams/surveys.

## Constraints

We only have two major concerns/constraints regarding our database and overall security. The system implemented should ensure that all communications are secure and encrypted following the specifications outlined in section 5.3 of the SRS. Our database should also not only ensure data security, but data integrity that will avoid race conditions that could cause issues since we are dealing with a massive student base.

## Composition

The modules will interact with their associating sub components in order to accomplish their relevant I/O towards the overarching project plan.

## Uses/Interactions

-Some Brief Examples-

Survey Creation will interact with the database to store surveys and will use the notification system to alert students. Team/Project Creation will interact with the database and the Survey Creation module for automatic team generation. Meanwhile the Deadline Scheduler will interact with both Team Creation and UI for setting and displaying deadlines. The Archive will communicate with the database to retrieve files that are no longer active.

## Resources

* + Database will be managed with Microsoft SQL Management Studio
  + Notifications will be handled through SMTP Protocol
  + UI Hosted on Azure Web Services

## Processing

Survey Creation – Will utilize algorithms specialized in grabbing aggregated data to utilize with the Data Display module.

Team Creation – Will use sorting and matching algorithms for automatic team creation based on surveys.

Deadline Scheduler - Will use date-time algorithms to manage and display deadlines on a calendar.

## Interface/Exports

Services Included:

* + CRUD operations for survey, teams, deadlines, archive and notifications.
  + Notification services for email alerts.

Each service will include proper documentation in the source code, following the guidelines in the SRS.

## Detailed Subsystem Design

I will include a detailed diagram showcasing this later.

# Glossary

NA – Will fill this out later when I have more information.

# Bibliography

Nothing referenced so far.