Whiteboard
Software Design Document

Document Version 0.1
Last Updated: 10/19/2018

Josh Reichman
Josh Heri, Sam Jenkins, Yash Goswami

Change Log

Date	Version	Description	Author(s)
10/13/2018	0.0.1	Initial Draft	Josh Reichman Sam Jenkins Yash Goswami
10/13/2018	0.0.2a	Wrote Introduction section	Josh Heri
10/14/2018	0.0.2b	Added Architecture overview	Josh Heri
10/17/2018	0.0.3	Improved document formatting	Josh Reichman
10/18/2018	0.0.4	New Figures	Josh Reichman Josh Heri Sam Jenkins
10/19/2018	0.0.5a	Condensed wording and Improved figures	Josh Heri
10/19/2018	0.0.5b	Updated document formatting	Josh Reichman
10/19/2018	0.0.5c	Added Server Architecture	Josh Heri
10/19/2018	0.1	Revision Finalized	Josh Reichman Josh Heri Sam Jenkins Yash Goswami

Table of Contents

Introduction

Purpose

This design document serves to provide a comprehensive overview of the architecture behind Whiteboard, a end-user oriented Learning Management System (LMS). It has been written in order to document and explain the design decisions behind the software, as well as serve as the basis for the development process.

Scope

The scope of the Whiteboard project is to create a LMS capable of supporting basic curricular organization for a mid-sized educational institution.

Definitions

Term	Meaning
Login	In the context of this paper, the term “login” may be used as a noun interchangeably with that of “account”. For example “the user’s *login* is assigned student permissions”.

Abbreviations

Abbreviation	Meaning
LMS	Learning Management System
S	Identifier for “Student” role within the application
I	Identifier for “Instructor” role within the application
H	Identifier for “Teaching Assistant” role within the application
A	Identifier for “Administrator” role within the application

References

None

Core Architecture

Basic Overview

The application will be split into two components: server-facing and client-facing.

The client-side of this service will service students, instructions, and administrators alike. Depending on login permissions it will assist with operations ranging from assignment submission, to class grading, to the creation of users and courses. It will operate within a Java desktop application, largely due to the language's platform-independence.

The server-side of this service will handle authentication and translate requests from the clients, returning appropriate files from a hosted file server or data from the integrated database. It will feature a C# framework running on a Windows 2016 Server. This application will run alongside the service's SQL Server database and NTFS filesystem.

Graphical Representation

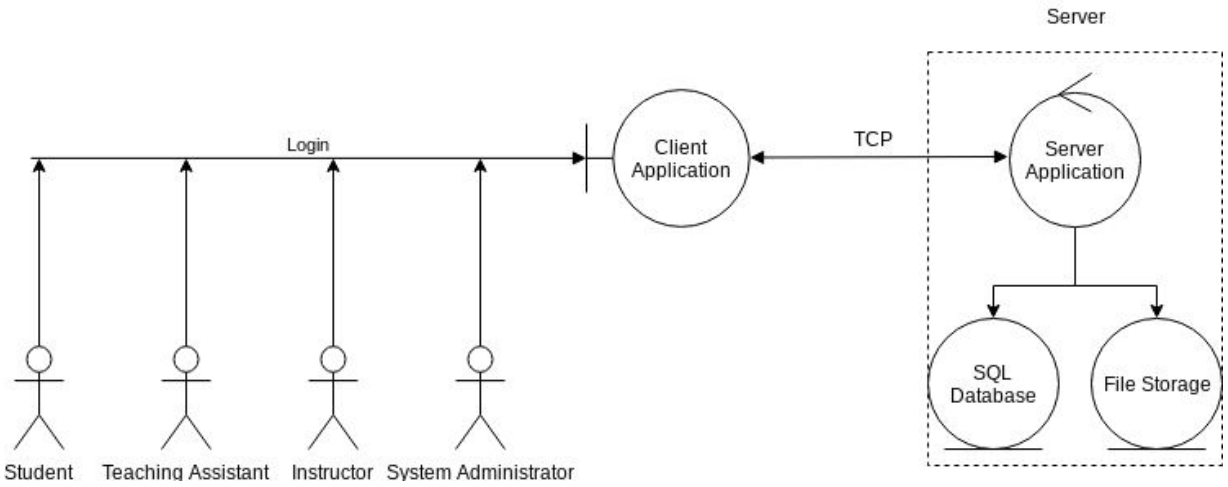

Figure 1

Client Architecture

Inputs and Outputs

Type	Method
Input	• System will accept input from keyboard and mouse on host terminal • Users will be able to upload files from client terminal
Output	• System will output search results to the screen • Users can download files/reports from the system

Login Page

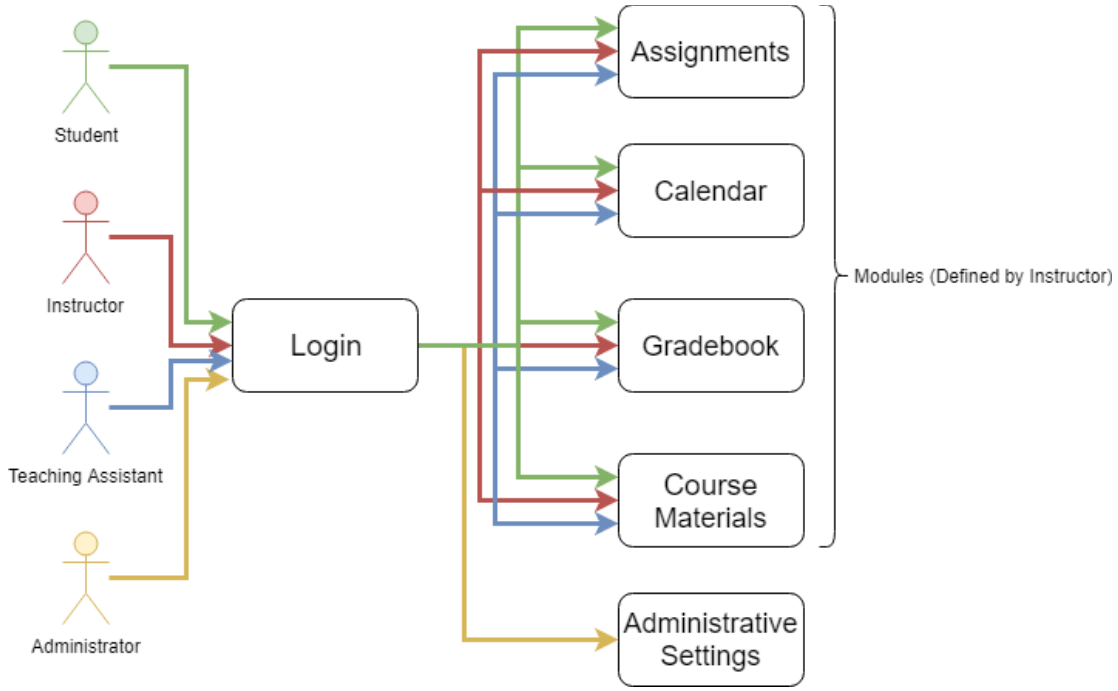

Figure 2

The crux of the client application is a central login screen. All users of the system (students, teaching assistants, instructors, and administrators) can enter their credentials in the same location. After authenticating with the server, the client will redirect them either to their classes (w/ various permissions depending on role), or to an administrative UI for adjusting classes / users.

Client Architecture - Use Case Representation

Once a login is authenticated against the server, and the core UI is built according to the following use-case realizations:

Use Case ID	Title	Description	For Roles
UCR.1	Load Courses	Loads appropriate courses	S, I, H
UCR.2	Edit User Settings	Edit User Preferences	S, I, H, A
UCR.3	Load Grades - Global	Loads global grade summary	S
UCR.4	Load Calendar - Global	Loads global calendar	S
UCR.5	Expand Course[1]	Select and load indv. course	S, I, H
UCR.6	View Course Assignments	Load assignments for selected course	S, I, H
UCR.7	Edit Assignments	Create or cancel assignments, add descriptions, set and change due dates, lock and unlock assignment visibility	I
UCR.8	Submit Assignments	Turn in assignments	S
UCR.9	Grade Assignments	Grade submissions	H, I
UCR.10	View Calendar	View course calendar	S, I, H
UCR.11	View Course Materials	View files / pages created by instructor	S, I, H
UCR.12	Edit Course Materials	Edit / add files & pages	I
UCR.13	Create / Assign Course	Create courses and add logins to them	A
UCR.14	Create / Assign User	Edit account profiles	A

[1] Note: The available modules visible on an expanded course to students / TA's depends on the instructor's initial choice. All courses include a gradebook and assignment section. See the server architecture section for a full course structure diagram.

Server Architecture

The core functionality of the server-side of Whiteboard occurs in a C# abstraction layer between the SQL database and file system. It is built around the following use-case realizations:

Use Case ID	Title	Description
UCR.1	Authenticate Login	Check user credentials submitted by client
UCR.2	Query Database	Query database and respond to client information request
UCR.3	Pass Files	Transfer files from file storage

The SQL database has 3 core tables (may be adjusted later in development) according to the following schema:

```
CREATE TABLE Logins
(
	ID int NOT NULL,
	DisplayName varchar(100) NOT NULL,
	UserName varchar(254) NOT NULL,
	PasswordHash varchar(70) NOT NULL,
	ApplicationRole char(1) NOT NULL,
	Aux_ApplicationRole char(1),
	LoggedIn bit NOT NULL,
	Token varchar(70),
	INDEX ix_graphid UNIQUE ($node_id)
)

CREATE TABLE Courses
(
	ID int NOT NULL,
	DisplayName varchar(100) NOT NULL,
	Students varchar(max) NOT NULL,
	Instructor varchar(1100) NOT NULL,
	TeachingAssistants varchar(1100),
	Modules char(10) NOT NULL,
)
```

```
CREATE TABLE Assignments
(
        ID int NOT NULL,
        CourseID int NOT NULL,
        StartDate
        DisplayName varchar(100) NOT NULL,
        Students varchar(max) NOT NULL,
        Instructor varchar(1100) NOT NULL,
        TeachingAssistants varchar(1100),
        Modules char(10) NOT NULL,
)
```

The login table stores basic information on each user's login, and most importantly, their primary (and secondary if needed) roles. When they authenticate against the client application, depending on the roles entered into this table, they will have access to different "views" (Student, TA, Instructor, Admin).

The courses associated with each login are actually determined by the course table. Each row is created when an administrator creates a course. Upon creation the administrator inputs the Students, TA(s), and Instructor(s) for a course from a list of available logins for each role. The Ids of the logins selected here are saved as a attribute in the course's row.

In addition to basic properties the course table also stores the "Modules" available for each course. This is set by the instructor upon first login.

The database follows the basic permission structure:

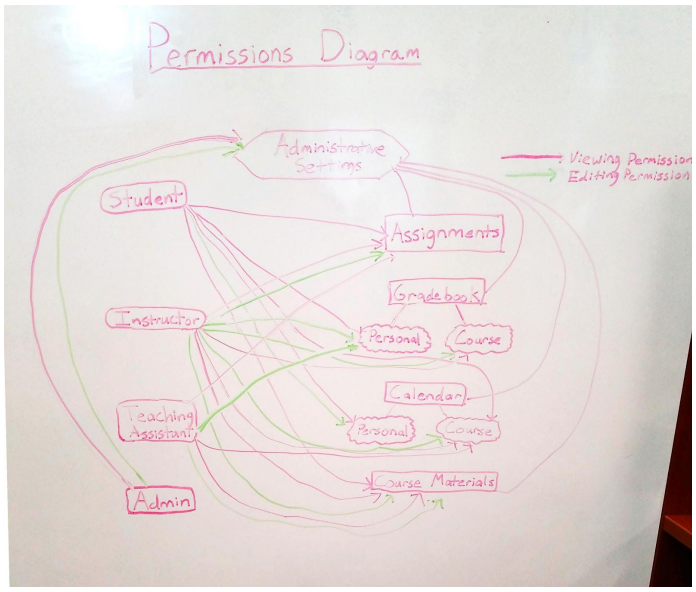

Figure 3

Business Requirements

Background

Currently, many students use the Canvas website in order to keep up with assignments and see their grades. However, Canvas has several problems that keep it from being an entirely user-friendly experience. These problems include a lack of a decent search utility, as well as difficulty in contacting teachers or seeking help for their assignments. Students need to be able to find their assignments quickly and easily, as well as access instructor and teaching assistant resources as need be. One of the most frustrating experiences with Canvas is when the student doesn't understand the assignment, and the only way to contact the professor is through an email they may or may not read before the due date. Additionally, on the instructor side, many professors have complained about the system for creating assignments, discussions, and adding resources to be unintuitive. For some less tech savvy professors, it can be very frustrating to have to contact the help desk over something as simple as uploading a PDF under the documents heading.

Objectives and Success Criteria

Rather than focusing on monetary success, the goal of this software is provide both students and instructors with an intuitive, user friendly experience with a planning and assignment organizing service.

Risks

RI1 - System is very similar to older software, may not provide enough of an advantage to justify switching and learning a new system
RI2 - Desktop application may cause unforeseen security risks or permissions issues