Architecture Design and Process Report

THE CODING CHALLENGE SYSTEM

Heritage College | Development Project

Last Updated: April 25, 2022

Contents

[Introduction 2](#_Toc470983942)

[Black Box View 3](#_Toc696235758)

[White Box View 4](#_Toc1939291909)

[Software Architecture 4](#_Toc801374229)

[Architecture Tiers 4](#_Toc17665138)

[Object Model 5](#_Toc1480309053)

[Data Model 5](#_Toc1539916172)

[Presentation Frameworks 6](#_Toc994296208)

[MVC vs Webforms 6](#_Toc1056585438)

[.NET vs .NET CORE 7](#_Toc1635377651)

[Code-first vs Data-first 7](#_Toc242090805)

[Coding Standards and Naming Conventions 7](#_Toc976550990)

[C# .NET Core 7](#_Toc1477700479)

[LINQ 8](#_Toc1671492126)

[Testing 8](#_Toc1907061754)

[Security 9](#_Toc425349436)

[Views 9](#_Toc679321036)

[Formatting 10](#_Toc1278592442)

[Folder Structure and File Layout 11](#_Toc1380385857)

[Document Organization Standards 13](#_Toc233418553)

[Storage and Folder Structure 14](#_Toc820403485)

[Directory 14](#_Toc317336839)

[CCS Files and Solution 14](#_Toc1746162932)

[Sprint 15](#_Toc1246160574)

[Test Standards 15](#_Toc1649785179)

[Black Box 15](#_Toc780517789)

[White Box 21](#_Toc1626611163)

[Process 22](#_Toc636420166)

[Appendix 22](#_Toc2030500909)

[Hosting Judge0 with Docker 23](#_Toc912212317)

Figures

[Figure 1 Black Box View 3](#_Toc101766783)

[*Figure 2 White Box View* 4](#_Toc101766784)

[*Figure 3 Object Model Diagram* 5](#_Toc101766785)

[Figure 4 Data Model Diagram 6](#_Toc101766786)

[Figure 5 Document Organization Standards 13](#_Toc101766787)

[Figure 6 CCS Files and Solution 14](#_Toc101766788)

Tables

[Table 1 Coding Standards and Naming Conventions 8](#_Toc101766798)

[Table 2 LINQ 8](#_Toc101766799)

[Table 3 Testing 9](#_Toc101766800)

[Table 4 Security 9](#_Toc101766801)

[Table 5 Views 10](#_Toc101766802)

[Table 6 Formatting 11](#_Toc101766803)

[Table 7 Folder Structure and File Layout 12](#_Toc101766804)

[Table 8 Create Challenge ATDD 15](#_Toc101766805)

[Table 9 Edit Challenge ATDD 16](#_Toc101766806)

[Table 10 Delete Challenge ATDD 17](#_Toc101766807)

[Table 11 Create/Edit Test Cases ATDD 17](#_Toc101766808)

[Table 12 Quick Compiler in Python ATDD 17](#_Toc101766809)

[Table 13 Teacher Main Dashboard to Navigate ATDD 18](#_Toc101766810)

[Table 14 Three Main Browsers ATDD 18](#_Toc101766811)

[Table 15 Heritage Themed Design and Layout ATDD 19](#_Toc101766812)

[Table 16 Log into the System ATDD 20](#_Toc101766813)

[Table 17 Student Start a Challenge ATDD 20](#_Toc101766814)

[Table 18 Process 22](#_Toc101766815)

# Introduction

The purpose of this document is to outline all the requirements set into place for the CCS project and to provide resources used and the setup process our team is carrying out before the implementation phase. This document consists of black/white box views, S/W architecture, framework and tool decisions, coding, and document organization, and testing standards, as well as the process in which we will be following for tasks off the backlog. In the Black Box section under Test Standards, we included acceptance criteria for user stories we have worked on, however there are some unfinished tests that have not been run yet. For that reason, there will be some rows in the tables that will not be applicable.

# Black Box View

Below in figure 1, black box view, you can see the interactions between CCS and the teacher, student, and AMS.

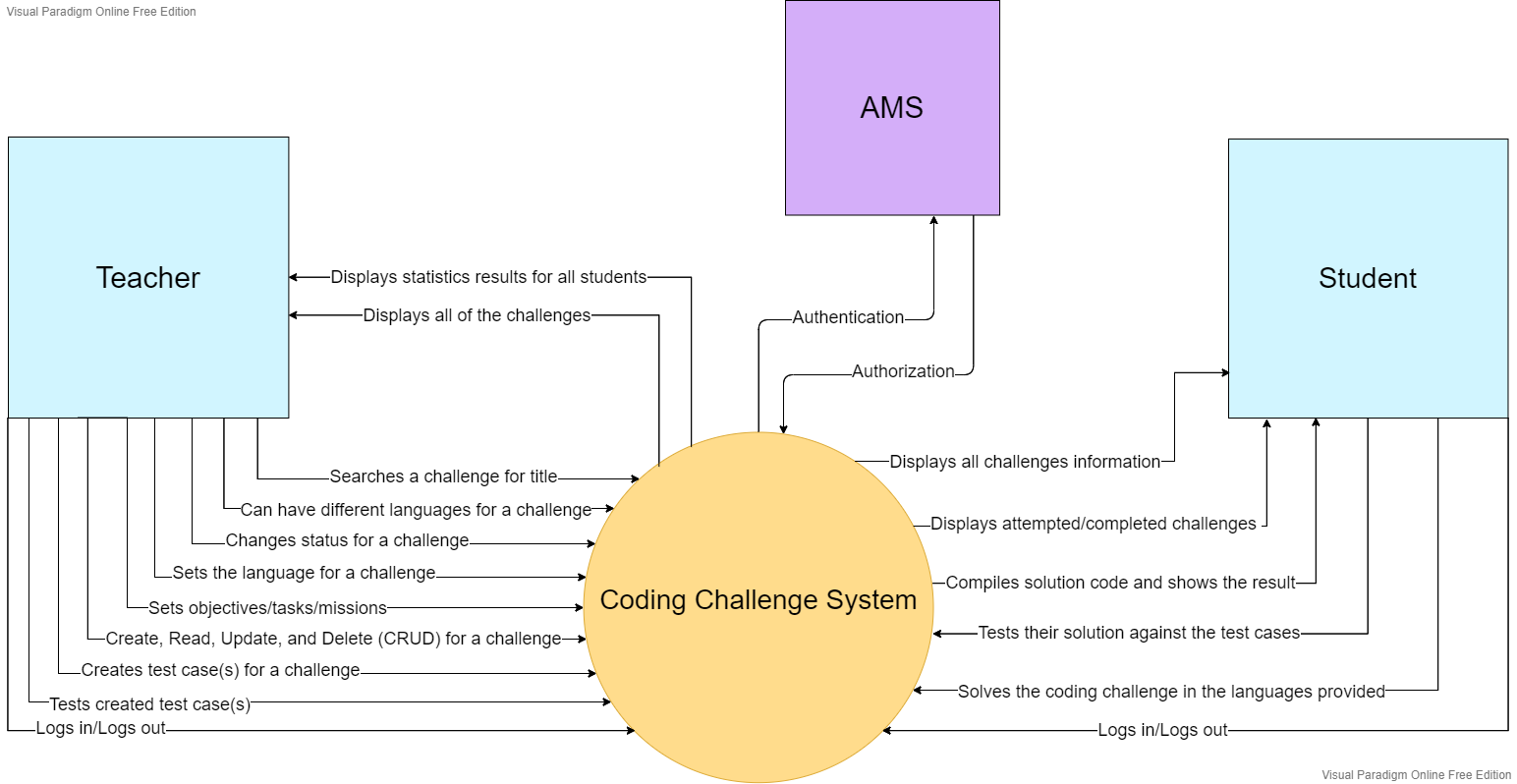
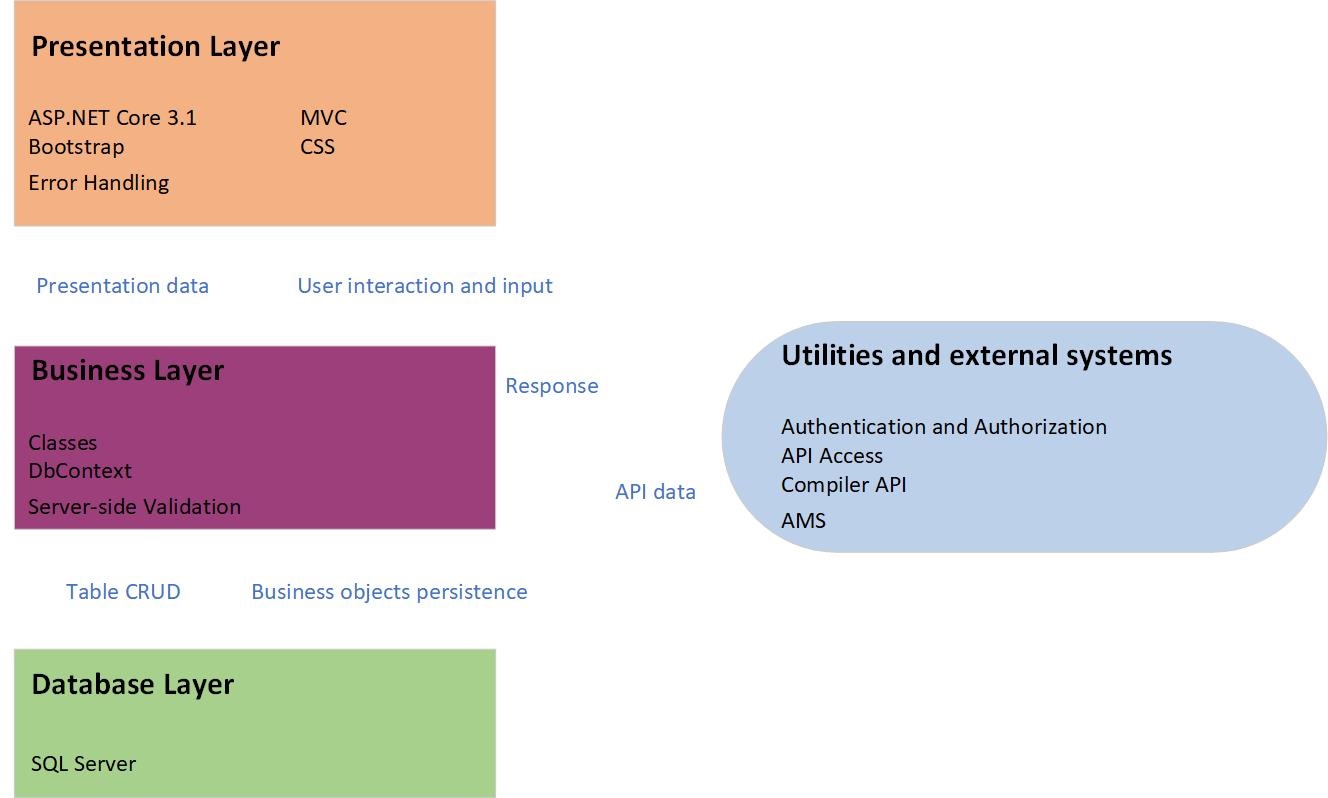


Figure 1 Black Box View

# White Box View

Below in figure 2, white box view, you can see the implemented components used in CCS.



*Figure 2 White Box View*

# Software Architecture

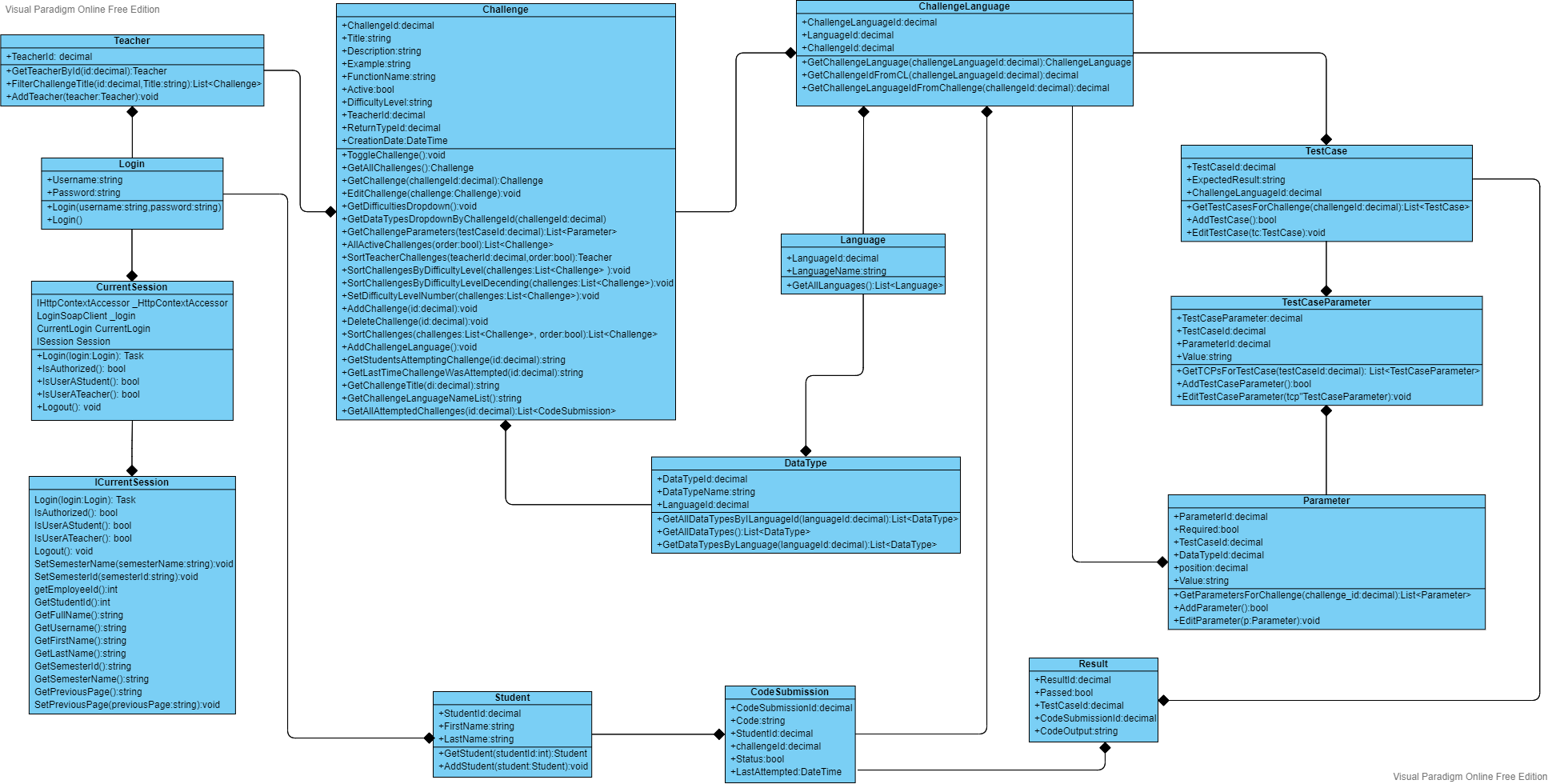
## Architecture Tiers

For this project, we will be using the Model View Controller (MVC) design pattern. As a best practice, we will be doing this in the fat model and skinny controller method, which entails keeping both the database access and main logic to live within the Models, and having the Controller simply be a nice interface between the View and Model. As such, access to AMS will be done through the model. Validation will be done in the Models and Views, to ensure maximum robustness. Most of the code will be server side, except for partial client-side validation in the Views for any forms that may require it.

This system will interact with two outside systems. AMS will be used for authentication and authorization, and a self-hosted Application Programming Interface (API) to compile code. No extensive configuration is needed for AMS, as it is already hosted on CSDEV; a Heritage College application & development server running on the local network on campus. This server is not available externally to this network. Currently, the Judge0 API (v1.13.0) is being hosted on UXDEV, due to issues with the school network not trusting HTTP and the API must be HTTP. Heritage IT is planning to make it accessible on CSDEV. However, to use this in Production, the Judge0 API will need to be hosted on CSPROD, a windows server. Reference the Appendix section of this document for more information about Judge0.

## Object Model

Below in figure 3, object model diagram, represents the class diagram of CCS.



*Figure 3 Object Model Diagram*

## Data Model

Below in figure 4, data model diagram, represents the class diagram of CCS.

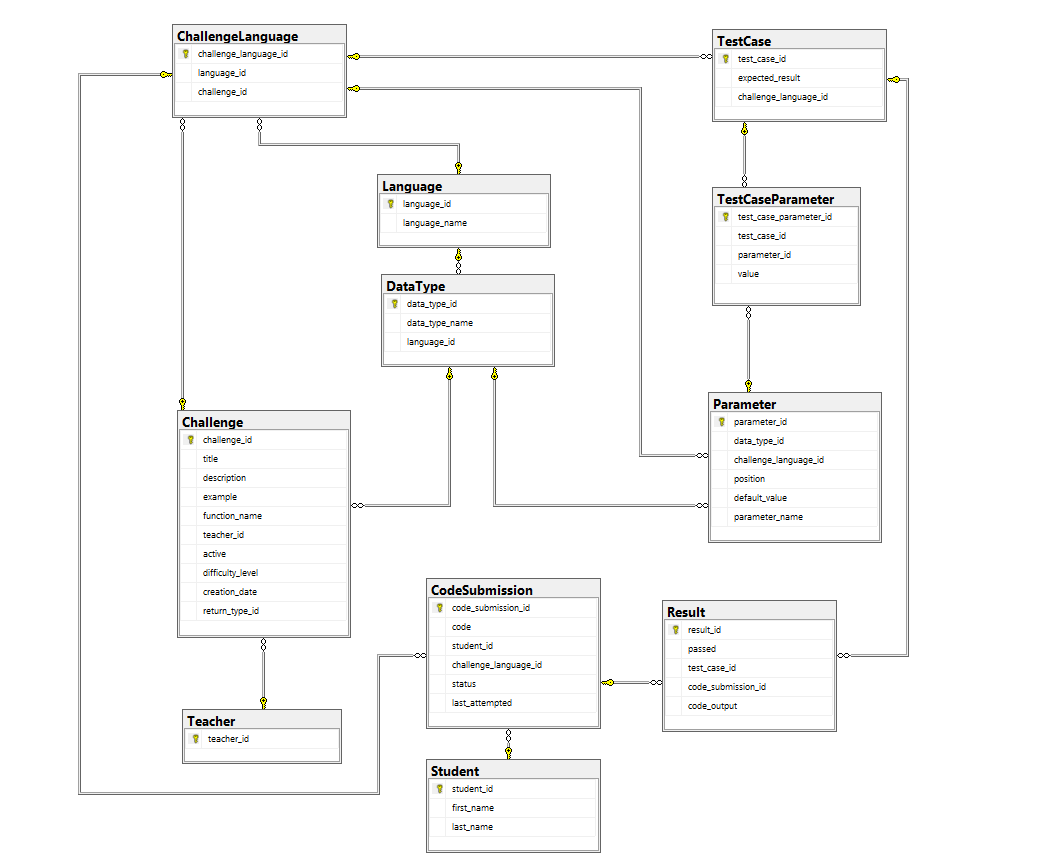


Figure 4 Data Model Diagram

# Presentation Frameworks

The Coding Challenge System will use the [Heritage College stylesheets](https://csazure.cegep-heritage.qc.ca:8080/F2021-DevProject/_git/Code%20Challenge?path=%2FDocumentation%2FResources%2FHeritageStyle&version=GBmaster) and Bootstrap for its CSS. We will use [Bootstrap v4.3.1](https://getbootstrap.com/docs/4.3/getting-started/download/).

## MVC vs Webforms

The first consideration our team had was whether to use Model–view–controller (MVC) or Webforms. MVC is a new standard design Pattern in Software development. Compared with Webforms, MVC provides more control over HTML, jQuery, JavaScript, and CSS. MVC also provides better testability of the Web Application and good support for Test Driven Development (TDD). Our team determined that MVC is better than webforms for our project. We decided that an MVC approach using .NET Core will be our best decision. The team made this decision because MVC is more logically organized, and the separation of layers will be more consistent as a result, as well as it currently being the Microsoft supported framework going forward since Webforms is at its end-of-life. This level of support will allow for a longer lifespan of the system before needing to be reworked for newer technologies.

## .NET vs .NET CORE

The second consideration was whether to use .NET Core or .NET Framework. We decided to use .NET Core. The team reached this decision because we hope our application will be used for a long time after we are gone, and the .NET Framework will no longer be supported after April 26, 2022. We decided to use .NET Core v3.1 since it is the newest version. This version will no longer be supported in December 2022.

## Code-first vs Data-first

Our third consideration was whether to use code-first or data-first. The code first approach allows you to define your project MVC model using C# as opposed to depending on the database design. On the other hand, the data-first approach is when using a database to create entity data models. We chose to do data-first because seeing what data and what type of data we are going to use before having to code it will make the process easier. Making the database first will also reduce the possible risks because it gives a strong sense of the system overall structure.

# Coding Standards and Naming Conventions

## C# .NET Core

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Convention** | **Example** |
| Namespace | PascalCase | namespace **CCS.Models** |
| Class | PascalCase,  Noun or noun phrases; | public class **User** {} |
| Property | PascalCase | public int **UserId** {get; set;} |
| Methods | PascalCase | public void **JoinQueue**() {} |
| Parameter | camelCase | public void JoinQueue(int **userId**) {} |
| Local Variables | camelCase | int **userId**; |
| Brackets | Egyptian Style | public void JoinQueue() **{**  **// code**  **}**  if (condition) {  **// code**  } else {  **// code**  } |
| Readonly property | camelCase with underscore | private readonly **\_db** |

Table 1 Coding Standards and Naming Conventions

## LINQ

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Syntax** | **Example** |
| LINQ Queries | Both Query and Lambda are acceptable | **Query:**  var teachers = from teacher in teachers select teacher;    **Lambda:**  var teachers = await \_db.Teachers.OrderBy(g => g.Name).ToListAsync(); |

Table 2 LINQ

## Testing

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Convention** | **Example** |
| Test Project Name | Should be name of the project that is being tested + “Test” all in PascalCase | **MlC50StoreTest** |
| Namespace | PascalCase | namespace **MlC50StoreTest** |
| Class | PascalCase | public class **UpdateBuyPriceTests**{} |
| Methods | Pascal\_Snake\_Case | public void **Test\_Join\_Queue**() {} |

Table 3 Testing

## Security

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Convention** | **Example** |
| Model Validation | - Data Annotation,  Custom Validations | [Required(ErrorMessage = "Title is required")]  public string Title { get; set; } |
| SQL Injection | - Antiforgery Token, on class or method | [ValidateAntiForgeryToken]  [HttpPost, Route("Semester/Add")]  public IActionResult Add(Semester semester) {  ...  } |
| Cross Site Scripting | - Encode all unvalidated input, using built in Razor, JS Encoder, or Html Encoder.  - More info: [Prevent Cross-Site Scripting (XSS) in ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-5.0#:~:text=HTML%20Encoding%20using%20Razor%20The%20Razor%20engine%20used,encoding%20rules%20whenever%20you%20use%20the%20%40%20directive.) | HtmlEncoder \_htmlEncoder;  JavaScriptEncoder \_javaScriptEncoder;  UrlEncoder \_urlEncoder; |

Table 4 Security

## Views

|  |  |  |
| --- | --- | --- |
| **Category** | **Rule** | **Example** |
| Razor Views | - ASP HTML tags will be used as opposed to HTML helpers |  |
| HTML/CSS | - Google’s HTML and CSS standards will be used. | Link: <https://google.github.io/styleguide/htmlcssguide.html> |
| - All identifiers will have meaningful names  - Kebab-case  - Should have one newline space between each CSS code block |  |
| - CSS headers that are too long should be listed line-by-line or completely changed |  |

Table 5 Views

## Formatting

|  |  |  |
| --- | --- | --- |
| **Category** | **Rule** | **Example** |
| Indentation | - Four spaces will be used in indentation: | Refer to the above examples. |
| Parenthesis | - Always use parenthesis for clauses in expressions  - All braces will be on their own lines | if **(\_**cs.IsAuthorized()**)** {  return View();  } |

Table 6 Formatting

## Folder Structure and File Layout

|  |  |  |
| --- | --- | --- |
| **Category** | **Rule** | **Example** |
| Grouping CSS in Folder | - View-specific CSS should be in its own file  - All common CSS will be in main.css |  |
| Grouping CSS in File | - Group CSS by the flow of the page  - Label each group with a comment |  |
| Libraries | - Libraries should be put in their own folders |  |
| Images | - Images should be put in their own folder |  |

Table 7 Folder Structure and File Layout

# Document Organization Standards

The document organization standard which the CCS team is following is to have all essential documentation stored on school servers under the directory “CCS Team Documentation,” where sub-folders are organized to have specific files assigned to each of them. Our naming standard for the files are for each file to start with “CCS\_” followed by the file topic, which gets placed in the respective folder.

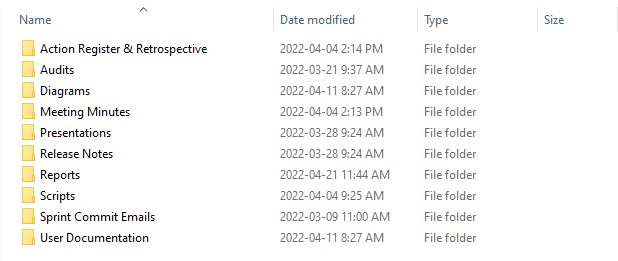


Figure 5 Document Organization Standards

## Storage and Folder Structure

All project documentation will be kept in the appropriate CCS project on CS AZURE. All members of the CS AZURE project will be able to access these files.

## Directory

All documents will be stored on CS AZURE to ensure that the team and project manager have access to all up‑to‑date documents required for the project.

## CCS Files and Solution

The files and solution for the system will be kept in a separate folder called CCS inside the first CCS folder. (CCS-> CCS).

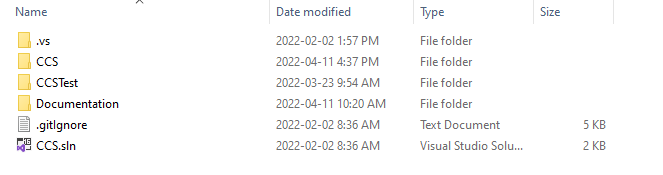


Figure 6 CCS Files and Solution

## Sprint

All sprint documents will be put in the appropriate sprint’s folder such as meeting minutes, sprint review, etc. These documents will follow the same naming conventions as the directory example above.

# Test Standards

## Black Box

Acceptance Test-Driven Development (ATDD) is a synergetic process where a development team, testing team, and business representatives all come together to task out requirements, identify potential drawbacks, and to reduce the likelihood of errors before coding is started.

As a Teacher, I want to create a challenge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Ability to create and save new challenges | Successful creation of a challenge with proper input | 2021-12-02 | Abraham | Challenge is created and saved to database  Pass |
| Ability to display error message when required field is not filled | Unsuccessful creation of a challenge due to incorrect input | 2021-12-02 | Abraham | Challenge is not created, returns to Create view. Error message not displayed.    Fail (out of sprint scope) |
| The languages field is a dropdown list. | N/A | N/A | N/A | N/A |
| The edit challenge form has consistent design for the labels and fields. | N/A | N/A | N/A | N/A |
| Bootstrap icons are used in the create challenge page | N/A | N/A | N/A | N/A |

Table 8 Create Challenge ATDD

As a Teacher, I want to edit a challenge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| The teacher is successfully able to edit a challenge | Successful update of a challenge with proper input | 12-02-2021 | Emmanuelle | This test was partially successful. Most fields can be changed, and the changes will be in the database, except the return type in function that never changes. |
| Unsuccessful update of a challenge due to incorrect input | 12-02-2021 | Emmanuelle | This test was unsuccessful. We can remove the title and description which cannot be null in the database. It will make the system crash. |
| The languages use a dropdown list on the edit challenge page. | N/A | N/A | N/A | N/A |
| The languages are ordered in the edit challenge page. | N/A | N/A | N/A | N/A |
| The edit challenge form has consistent design for the labels and fields. | N/A | N/A | N/A | N/A |
| Bootstrap icons are used on the edit challenge page. | N/A | N/A | N/A | N/A |
| There is an Update button instead of Save button on the edit challenge page. | N/A | N/A | N/A | N/A |

Table 9 Edit Challenge ATDD

As a Teacher, I want to delete a challenge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Teacher is successfully able to delete a challenge | Cascading works successfully in the database after the deletion of a challenge. | 12-02-2021 | Noah | Deleting a challenge deletes all entries in other tables that are associated with that challenge (cascade happens) |
| Student who submits a deleted challenge will be prompted with an appropriate error. | Submit a challenge as a student after it is deleted, an appropriate error should be displayed. | N/A | N/A | N/A (Student submissions not a feature for sprint 1) |
| Delete warning message is more detailed | When trying to delete a challenge you can see information about how many students are attempting the challenge and when it was last attempted | 14-03-2022 | Emmanuelle | We can see how many students are attempting the challenge and the date that the challenge was last attempted |
| The challenge title is in the delete successful message | You can see the appropriate title in the delete message | 14-03-2022 | Emmanuelle | The right title will show up in the message when the challenge is deleted |

Table 10 Delete Challenge ATDD

 As a Teacher, I want to create/edit test cases for a challenge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Teacher can create a test case and add it to a challenge | A test case can successfully be created | N/A | N/A | N/A |
| An invalid test case will prompt an error | N/A | N/A | N/A |
| A test case can be successfully attached to a challenge | N/A | N/A | N/A |

Table 11 Create/Edit Test Cases ATDD

As a User, I want a quick compiler in Python

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| The compiler must interpret code in under 5 seconds | Code is interpreted in under 5 seconds | 12-02-2021 | Noah | The compiler takes approx. 2-4 seconds to return a result. |
| The compiler is running on CSDEV | N/A | N/A | N/A | N/A |
| The compiler is hosted on test | N/A | N/A | N/A | N/A |

Table 12 Quick Compiler in Python ATDD

As a Teacher, I want a main dashboard to navigate around.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Challenges should be sorted | Challenges should be sorted by titles in alphabetical order | 12-02-2021 | Emmanuelle | Shows all the challenges in alphabetical order and they can be sorted by difficulty level from easy to hard. |
| The list of challenges should be all there | All challenges in the database for the teacher logged in should be visible | 12-02-2021 | Emmanuelle | Shows all the challenges for the teacher that is logged in |
| Challenges can be searched by title | Challenges should be allowed to be searched by title for easier access | 12-02-2021 | Emmanuelle | When you search for a title, the closest matches will be moved to the top of the visible challenges |
| Title column in the teacher challenges page is left aligned | N/A | 04-06-2022 | Emmanuelle | The title is left aligned |

Table 13 Teacher Main Dashboard to Navigate ATDD

As a User, I want to be able to use the system on one of the three main browsers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Entire system should work appropriately and not fail for different browsers. | Chrome functionality is successful. | ​11-23-2021 | Zhaojiang Zhong | Successful |
| Firefox functionality is successful. | ​11-23-2021 | Zhaojiang Zhong | Successful |
| Microsoft Edge functionality is successful. | ​11-23-2021 | Zhaojiang Zhong | Successful |

Table 14 Three Main Browsers ATDD

As a User, I want a Heritage themed design and layout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Styling should work on different browsers | Chrome | 12-02-2021 | Nahom | the layout is the same |
| Firefox | 12-02-2021 | Nahom | the layout is the same |
| Microsoft Edge | 12-02-2021 | Nahom | the layout is the same |

Table 15 Heritage Themed Design and Layout ATDD

As a User, I want to be able to log into the system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Valid account should be accepted. | Teacher Account username:userte password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | Login successful, go to Teacher Home Page |
| Continuing Education User username:userce password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | Login successful, go to Teacher Home Page |
| Coordinator User username:userco password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | Login successful, go to Teacher Home Page |
| Central Co-op Coordinator User username:usercc password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | Login successful, go to Teacher Home Page |
| Student Account username:1111111 password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | Login successful, go to Student Home Page |
| Student Account username:2222222 password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | Login successful, go to Student Home Page |
| Student Account username:3333333 password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | Login successful, go to Student Home Page |
| Invalid account should be denied. | Administrator User username:userad password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | error page. No employeeId |
| Teacher Account username:userte password:123test! | 2021-12-02 | Zhaojiang Zhong | cannot log in, wrong username or password |
| Student Account username:1111111 password:cs@123 | 2021-12-02 | Zhaojiang Zhong | cannot log in, wrong username or password |
| Generic User username:cstest password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | cannot log in, wrong username or password |
| Co-op Coordinator User username:usercp password:cs@123test! | 2021-12-02 | Zhaojiang Zhong | cannot log in, wrong username or password |

Table 16 Log into the System ATDD

As a Student, I want to start a challenge

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| The choice of language is right aligned on the start challenge page | N/A | N/A | N/A | N/A |

Table 17 Student Start a Challenge ATDD

[Link to the Board on CS AZURE](https://csazure.cegep-heritage.qc.ca:8080/F2021-DevProject/Code%20Challenge/_boards/board/t/Code%20Challenge%20Team/Stories)

## White Box

Compared to NUnit (just for C#) or other frameworks, XUnit testing has a more extensible testing framework, although it might require more effort for migrating existing implementations to XUnit framework. Our team will use XUnit for unit testing.

TDD mocking is creating an object that simulates another object (normally by constructing simple dependent classes) and setting expectations for its behavior. Mocking is a testing isolation method that simulates the interaction between components and systems. The goal is to test the interface functionality and the correct behavior. No mocking means it puts all the code into the same class and achieves a pass on all tests. Then, refactor the code by decomposing the large class into constituent classes and methods. Our team will use no mocking to test the code at model layer.

We will use class name add “Test” for class for naming standards for tests. If we will test “Challenge” class, we name it for “ChallengeModelTest.” And we will use “MethodName\_WhatsBeingTested\_ExpectedResult\_OtherInfo” format as the naming standard for our test cases. If we test “GetTitleLength” method, we will name test method for “GetTitleLength\_Input\_NoMoreThan30\_Test”.

# Process

|  |  |
| --- | --- |
| **New** | Column is used when an item has been created but not started. |
| **Analysis/Software Design** | Column is used when an item is being scoped out regarding what needs to be done, and how to carry it out. This is optional and intended for larger items. |
| **Active/In-Development** | Column is used when an item has been started and is being worked on. |
| **Pending Review** | This column would be used when an item needs to be reviewed by another person. The item would stay assigned to the person that was initially working on it. The reviewer will write “Reviewer: their name” in the task discussion so that everyone knows the task will be reviewed by said person. The reviewer will then edit their comment in the task discussion with notes based off their review and either set the task back to active (needs more work) or resolved (it looks perfect). |
| **Resolved** | Column is used when the review is done, and the reviewer does not believe it needs more work. |
| **Closed** | This column is used once the branch has been merged back unto the master branch. |

Table 18 Process

# Appendix

## Hosting Judge0 with Docker

1. Install WSL 2 with Ubuntu if not already installed

* Notes
  + Windows server:
    - Follow these steps (**Not tested**): <https://docs.microsoft.com/en-us/windows/wsl/install-on-server>
  + Windows home
    - Follow these steps (**tested**) <https://docs.microsoft.com/en-us/windows/wsl/install>
    - If using windows 10 19041 or higher / windows 11
    - You only need to enter “wsl --install” in PowerShell
    - Else, follow manual installation <https://docs.microsoft.com/en-us/windows/wsl/install-manual>

1. Install Docker Desktop

* Windows Server 2018 and Windows 10:
  + Follow these steps: <https://docs.docker.com/desktop/windows/install/>

1. Host Judge0 with Docker

* Follow these steps: [https://github.com/judge0/judge0/blob/master/CHANGELOG.md#deployment-procedure](https://github.com/judge0/judge0/blob/master/CHANGELOG.md%23deployment-procedure)

1. After this:
   1. Open your web browser with this link

http://<IP ADDRESS OF YOUR SERVER>: 2358/dummy-client.html

* 1. Make sure there is an HTTP response

This document has been read and approved by the following people, responsible for its implementation:

|  |  |
| --- | --- |
| Name | Approval |
| Emmanuelle Fontaine | Approved |
| Nahom Haile | Approved |
| Abraham Getachew | Approved |
| Zhaojiang Zhong | Approved |
| Alexander Dionne | Approved |
| Noah Loomis | Approved |
| Arsen Hakobyan | Approved |