Architecture Design and Process Report

Team 1

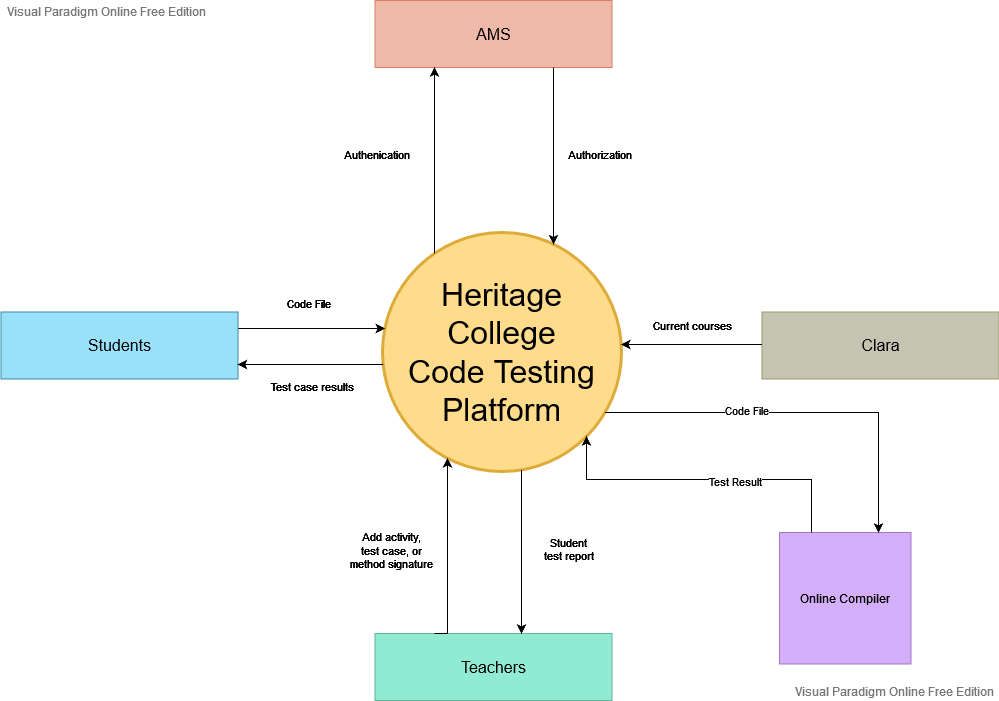
Heritage College | Development project  
Last updated: March 9, 2022

THE HERITAGE COLLEGE CODE TESTING PLATFORM

# Introduction

The purpose of this document is to give an in-depth explanation of the architecture, design and process of the Heritage College Code Testing Platform (HCCTP). It explains the black box view as a context diagram and clearly shows the system’s boundaries. The white box view at a high level is shown as well. The architecture decision and the considerations/alternatives that were thought of. The tiers of the architecture at a high level and the decision of what is done client-side vs server-side. The object model and database model, also details on how to add on to them. The decision on frameworks and tools. The decision on coding standards for all coding languages being used and the folder structure of source file. Documentation organization standards for project. Testing standards from a black box and a white box view. Lastly, the team has some proposed recommendations on changes wanting to be made with the Azure Dev/Ops workflows.

# Black Box View



# White Box View

A picture containing graphical user interface

Description automatically generated

Object ModelDiagram

Description automatically generated

# Data Model

Diagram

Description automatically generated

# Software Architecture

Considerations and Alternatives

The first decision made from our team was to use the .NET Core framework using MVC. We had to consider which version of .NET Core to go between 3.1 and 5. We decided to go with .NET 5 because it is the current version and an upgrade to .NET 6 from .NET 5 will be easier than an upgrade from a lower version. The .NET 5 is also the currently recommended version by Microsoft. Additionally, our team is most experienced with .NET Core using MVC. The MVC model is flexible and will make separating the different layers easy. Furthermore, we can add additional project such as libraries and the data layer to the solution as needed.

The second consideration was whether to use code-first versus data-first for the database integration. Since we need data from the Clara Database for the users and courses, we have decided to go with a data-first approach. Planning the initial design will be simplified by using data-first, due to the various data needed for this application to function. This includes the method signatures with its name, parameters and the test cases associated with the signature.

Architecture Tiers

The application will be split into 5 different layers. The views will only contain the data to be displayed and only use logic if needed to display data correctly. The controllers will serve only to relay the data from the view to the model. The models will contain all logic and communicate the data to the data layer and compiler models. Furthermore, the models will contain the validation for the server side. The data layer will contain all the entities related to our database. Finally, the utility layer will have 3 parts: the first will be the authentication and authorization from AMS, the second will be the compiler models for the calls to the compiler API, and thirdly, the entities for the calls to the Clara database.

# Frameworks and Tools

The frameworks that we will be using for the development of the Heritage College Code Testing Platform will be: MVC, .NET Core and Entity Framework with a data-first approach. We decided to go with these frameworks because we feel comfortable with them and they are flexible.

# MVC

We concluded that MVC is better than WebForms for the implementation of this project. Since MVC is more structured, it will be easier to divide the tasks and the maintenance will be more efficient.

# .NET

Our team decided to go with .NET Core 5. As explained earlier, this version of .NET is the current version, and it is well supported. We have also been working in class with this version, so the team is familiar with it.

# Entity Framework – Data-First

The team chose to use Entity Framework with the data-first approach. As we are going to be using the Clara database to access the data for the users and courses, we decided that it would be more efficient to use that approach.

# Presentation Frameworks

For the project’s presentation, we will be using the Heritage College style sheets, Bootstrap version 5.1 to make the visual design look clean and professional, and jQuery to aid with the UI. Additionally, we will be using Razor pages to dynamically display data in the HTML.

## Judge0

The CTP team decided to use Judge0 as a compiler in our project. The source code will be sent to Judge0 as an API call. Judge0 will compile the source code and return the result as a HTTP response. In addition, the team will use HTTP Client from .Net Core library to make HTTP request and response calls to Judge0.

The project will use several supported programming languages from Judge0. To check the supported languages from Judge0, follow the link: <https://ce.judge0.com/languages/>. Only the following programming languages will be used for uploading source code:

* C#
* Java
* JavaScript
* PHP
* Python

The database will use the same IDs and language name as the Judge0 supported list of programming languages. To check the IDs and language name and version. Follow the link: <https://ce.judge0.com/languages/>.

**Source**: <https://ce.judge0.com/>

Coding Standards and Naming Conventions

All of our identifiers and names must have a meaning and be descriptive of what it is supposed to do.

Coding languages

Our team decided that the Heritage College Code Testing Platform will be using C# for our back-end code used with LINQ to get database data, JavaScript for our front-end code, and Transact-SQL for our database code in Microsoft SQL Server Management Studio.

Coding standards and conventions

The coding standards we will use for the project are the ones we learned in school. For C# and Transact-SQL, we learned it the Microsoft way and for JavaScript, we used the coding standards taught to us in school. As for the testing and LINQ code to get database information, we will follow the same conventions and standards as the C# ones. JavaScript standards and conventions will consist of using camelCase for identifier names (variables and functions), space around operators, always use two spaces for indentation of code blocks, etc.

.Net Core 5.0, C#

|  |  |  |
| --- | --- | --- |
| Identifier Type | Convention | Example |
| Class | PascalCase | public class Student {} |
| Property | PascalCase | public string Username { get; set; } |
| Methods | PascalCase | public void UserMethod() {} |
| Parameters | camelCase | public void UserMethod(int userId) { } |
| Local variables | camelCase | int userId; |
| Brackets | Egyptian Style | public void UserMethod(int userId) {  string test = "Big Test";  string newTest = test;  } |
| Readonly Property | CamelCase using an underscore before the property name. | private readonly \_context; |

**C# coding standard:** <https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>

LINQ

|  |  |  |
| --- | --- | --- |
| Identifier Type | Convention | Example |
| Linq queries | Both method and query syntax will be used. | Var users = from user in Users select user;  Var users = await \_context.Users.OrderBy(u => u.Name).ToListAsync(); |

HTML/CSS

For HTML and CSS formatting, we will be using Google’s HTML and CSS guide document.

Our CSS files will be stored in the wwwroot --> css folder.

**HTML/CSS:** <https://google.github.io/styleguide/htmlcssguide.html#CSS_Style_Rules>

JavaScript

Here is a reference to the JavaScript standards we will use.

Our JS files will be stored in the wwwroot --> js folder.

**JavaScript:** <https://www.w3schools.com/js/js_conventions.asp>

Override

* Use static methods when the code does not dépend on the object’s attributes
* Instantiate objects using <ObjectType> <ObjectName> = new() format
* Code indentation should be 4 spaces for all languages in this project

White Box

## For testing we will be using unit testing to test our functionalities. We will test in the purpose of single responsibility and name all our tests with meaningful names. We will also test using Dependency Injection to ease our testing. Our test environment will be XUnit on Visual Studio.

# Document Organization Standards

## Storage and Folder Structure

All project documentation will be kept in the appropriate HCCTP Azure DevOps project. All members of the Azure DevOps project will be able to access these files.

## Directory

All documents will be stored on [Azure DevOps](http://b218-t.cegep-heritage.qc.ca:8080/F2021-DevProject/_git/Code%20Testing) to ensure that the team and project manager have access to all up‑to‑date documents required for the project:

* Code Test Platform > Documentation
* Diagrams and Charts [Folder]
* Client Documents [Folder]
* Reports [Folder]
* Scripts [Folder]
  + Deployment [Folder]
  + System Test [Folder]
  + Tables\_and\_Views\_Scripts [Folder]
* Sprints [Folder]
  + Sprint# – [START\_DATE]-[END\_DATE] [Folder]
    - Emails [Folder]
      * CTP\_SprintCommit\_[Date].docx [File Format]
      * CTP\_Meeting\_Minutes\_[Date].docx [File Format]
      * CTP\_SprintReview\_[Date].docx [File Format]

CTP Files and solution

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

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 as the directory example above.

## Black Box

ATDD Approach

ATDD stands for Acceptance Test Driven Development. ATTD is a process that defines acceptance tests before implementing code. There is a collaboration between developers and stakeholders to create acceptance tests. The purpose of ATTD is to understand the behaviours of the system and what the expected outcome is.

User Story ID: 4 “As a teacher, I want all my courses from Clara for the current semester.”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Teachers can clone/share courses from other teachers | Teacher is able to choose another teacher to share a course. |  |  |  |
| Teacher gets a list of all courses for current semester automatically. | Get all courses for a teacher for only current semester. |  |  |  |
| Teacher get a course based off course name. | Get a course from a course name for a teacher. |  |  |  |
| Students enrolled in course have access | Student must be able to show only for the course/s they are enrolled in. |  |  |  |
|  | Student must not show other courses they are not enrolled in. |  |  |  |

User Story ID: 6 “As a teacher, I want to add activities (labs, assignments, tests, exercises) to my courses.”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result (Pass/Fail) |
| User is able to add an activity when all fields are valid. | User must enter a title. |  |  |  |
|  | User must enter a course. |  |  |  |
|  | User must enter a language. |  |  |  |
|  | User must enter a start date. |  |  |  |
| User receives a confirmation message when activity is successfully added. | Show a success message when all fields are valid. |  |  |  |
| User receives an error message when any required fields are not entered. | Show error message when one or more required fields is not entered. |  |  |  |
| Teacher is able to add an activity to a course. | Teacher adds an activity to selected course. |  |  |  |

User Story ID: 10 “As a teacher, I want to be able to add a method signature via form to assign to activities.”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Teacher can add method signature. | Teacher is able to enter the name of a method signature. |  |  |  |
|  | Teacher is not able to enter an empty field for the method signature. |  |  |  |
| CRUD functionality for method signature via a form. | Teacher is able to delete a method signature. |  |  |  |
|  | Teacher is able to edit a method signature. |  |  |  |
|  | Teacher receives a warning message when editing a method signature about test case parameters and expected results may be affected. |  |  |  |
| Test cases can be added to catalog. |  |  |  |  |
| Can add these test cases to an activity so students can test their code against it. |  |  |  |  |

User Story ID: 16 “As a teacher, I want to add test cases for the method signatures.”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Teacher receives a success confirmation message if the values are valid. | Teacher can successfully add a test case for a method signature |  |  |  |
| Teacher receives an error message when values are invalid | Error/ Warning message when adding test case with a missing required parameter |  |  |  |

User Story ID: 85 “As a Teacher I want to be able to edit test cases for an activity”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Teacher is able to remove non required parameters | Parameter is successfully removed |  |  |  |
|  | Error message when parameter is required (MethodSignature parameter) |  |  |  |
| Notification / warning message when signature changes | Add a parameter to a method signature |  |  |  |
|  | Remove a parameter from a method signature |  |  |  |
|  | Change a method signature parameter to be required |  |  |  |
|  | Change datatype of a method signature parameter |  |  |  |

User Story ID: 15 “As a student, I want to only have access to the courses I am enrolled in.”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acceptance Criteria | ATDD Test | Date Run | Tester | Result |
| Upon sign in, only get/display courses that student is registered to in Clara | Login as a student. Home page should display only student’s enrolled courses |  |  |  |
| Student should only see the activities from courses they are registered too | Make activities in a course that current student not registered in. Then go to activity summary for that student. The activities that were for a course that student not registered in should not be in the student’s activity summary. |  |  |  |
| If student drops a course, the course should no longer be available for student | Simulate a dropped course, then sign in with student. Course drop should no longer appear in the home screen |  |  |  |
| If student no longer a student, then should no longer have access to courses | Simulate a student leaving, sign in should fail, but access should still be removed from courses |  |  |  |
| Student from previous semester should no longer have access to courses | Change semester, login in as student, verify that previous semester courses are no longer displayed and cannot be accessed. |  |  |  |
| If there is a course search feature, course that student not registered too should not appear in search result | Search for a course that student not registered too, result should return empty. |  |  |  |

# Process

We recommend changing the Azure Dev/Ops workflow so that instead of an item being labeled as “Resolved” it would be “Pending review” so that it is easier for the team to know which item needs to be reviewed.

The process for the task would be: “New”, when the task is new and has not been worked on, “Active”, when it is being worked on, and when they think that they are done, they would then put it as “Pending Review” so that someone else from the team can look at it. When the task has been peer reviewed and it is approved, it would then go to “Closed”. If there are changes to be made, it would go back to “Active” and then back to “Pending Review” for another peer review to check the changes made and then to “Closed” if it is successful.

It is a similar process for a user story. It would have a state of “New” when it is a new user story and has not been worked on, “Active” when it is being worked on, “Pending review” when all its tasks are done and the acceptance criteria are getting looked at and then “Closed” when all the acceptance criteria work.