Final Development Project Report

cegep heriage college | Richard Chan

Development Project 1

Philip Dumaresq

2017

Table of Contents

[Table of Figures 2](#_Toc501144839)

[Introduction 3](#_Toc501144840)

[Development Process 4](#_Toc501144841)

[Course Purpose 4](#_Toc501144842)

[Process and Tools 5](#_Toc501144843)

[Teamwork 7](#_Toc501144844)

[Charts 7](#_Toc501144845)

[Design and Build – Analysis 11](#_Toc501144846)

[Planned vs Actual 11](#_Toc501144847)

[Refactoring 15](#_Toc501144848)

[Testing and Validation 16](#_Toc501144849)

[Black-Box testing 16](#_Toc501144850)

[Functional Testing 16](#_Toc501144851)

[Usability Testing 18](#_Toc501144852)

[Exploratory Testing 20](#_Toc501144853)

[Defects 21](#_Toc501144854)

[White-Box Testing 27](#_Toc501144855)

[Conclusion 28](#_Toc501144856)

# Table of Figures

[Figure 1: Sprint 1 Burndown Chart 8](#_Toc501144857)

[Figure 2: Sprint 2 Burndown Chart 8](#_Toc501144858)

[Figure 3: Sprint 3 Burndown Chart 9](#_Toc501144859)

[Figure 4: Original Layered Architecture 11](#_Toc501144860)

[Figure 5: Current Layered Architecture 12](#_Toc501144861)

[Figure 6: Original Class Diagram 13](#_Toc501144862)

[Figure 7: Current Class Diagram 14](#_Toc501144863)

[Table 1: Functional Test Cases 18](#_Toc501144864)

[Table 2: Usability Test Cases 20](#_Toc501144865)

[Table 3: Exploratory Test Cases 21](#_Toc501144866)

[Table 4: Defects 26](#_Toc501144867)

# Introduction

This report is going to cover topics from the development project course, starting with the course itself. It will go over the purpose of the development project, and what the project entailed, going into some of the technical skills used and acquired in the course. It will then go over the use of the Agile development methodology, and how we used it in the development of our system, including things that could have been changed to have improved the development of our system. While on the topic of the sprints and the work that was actually done, the report will go into a description of the team dynamics that we had and how each member of the team contributed to the development of this system. Following that will be our own team’s progress through each sprint, demonstrating our sprint burndown charts. Following discussions about the development process, the report will cover some of our original designs, including class diagrams, data models, and system architecture and how all of that changed over the course of the development of the system to meet changing user requirements. Looking past what we have done, and what we can improve, there will be a portion going into how sections of code can be refactored to improve the efficiency, clarity, or complexity of the code. Finally, the report will go into the testing method applied to the system, including black box and white box testing strategies. Incorporated in the black box testing strategies will be a variety of different test cases for functional testing, usability testing and exploratory testing. Then for white box testing, our own unit tests are going to be analyzed in terms of code coverage, how well they performed, the passing of our unit tests, and what we need to improve in our white box testing for future sprints.

# Development Process

## Course Purpose

The purpose of the development project course is to develop our skills in working as a team and bringing about the creation of a software system from conception all the way through the deployment. To be able to bring about the creation of a new system through every phase of the SDLC, we needed to have be able to demonstrate and apply a wide variety of skills learned throughout the course and in the rest of the program. We needed to apply our knowledge gathering user requirements throughout the earlier phases, meeting with our client to discuss requirements we’ve gathered, and being able to go and create diagrams of our system using the basic requirements that have been gathered. From there, we needed to have the technical knowledge to begin the actual development of the system. This means knowing the language, the framework, and the tools used to create the system, while learning anything we didn’t know – primarily MVC and the Entity Framework.

The course prepares us well decently well for building this system. The course could’ve been improved by giving us our development teams earlier on in the semester and told us which project we’ll be working on earlier on. I think having had this information, we could have focused a lot more on our own system instead of dividing our attention to things we wouldn’t need to touch again. In terms of technical skills, we we’re well prepared in terms of using Visual Studio, developing in C# using SQL Server, but Entity Framework even today remains a thorn in our side, not understanding how it handles certain things and feeling the need to fight with it to get it to work for us, instead of us doing what EF wants us to do. While MVC wasn’t as straightforward as originally thought by the team, it was very easy to pick up and we’re all very glad that MVC is the framework we chose to use for this.

In the first section of the course, we defined the functions of the two applications for both of the clients. We had meetings with each of the users where they described their system to us, going over the details of the process, allowing us to ask questions clarifying portions of the system process. After this, we went off and built diagrams showing different parts of the system, mainly use case diagrams showing off all of the primary functions that the system had be able to carry out. We then talked about how the system would be developed, and the order in which different functions needed to be built into the system. This whole portion of the development covered the first competency, defining the functions of the application.

After defining all of the functions that the application had to carry out, we needed to learn the different frameworks that we had as options for developing the system, and so we learned the Microsoft Model View Controller and Entity Frameworks. Once we’d learned those, along with different approaches to using the Entity Framework, we needed to define the technological requirements for the system, recommending which frameworks to use, and which approaches to use within the framework.

Knowing the technological requirements for the system carried us into being able to lay the groundwork for the application. This meant that as a team we needed to be able to think about everything that needed to get done in terms of user functions, how that translates with the technological requirements, and start actually planning the development of the system. This meant using Team Foundation Server to input all of the product backlog items that went into our 3 sprints, and creating tasks for each product backlog item. Then, at the beginning of each sprint going back through and carrying tasks forward and sizing each task.

Before beginning the actual development of the system, we needed to model the architecture that the system would follow. This meant designing each layer of our application and figuring out what the layer would do, and how they all interacted together. As a part of modelling the application, we also needed to model the database that we we’re going to use. Both the original data model used and the layers that we came up with we’re not what we ended up implementing by the end of the third sprint, so we did a lot of application modelling.

Finally, we wrote all of the code to implement this functionality in the back end, and tied a user interface to this to be able to show off our prototypes in client meetings. This included being able to take in all the required user information to create profiles for the user, and output the data that the user needed to fill out back to them. This included writing validation for user input, and coming up with as few ways as possible for the user to be able to break the system.

## Process and Tools

The sprint process was the cornerstone of this whole development project. Being marked on a sprint-by-sprint basis meant that evaluations were marked on what we committed to for each of our sprints, and what we were able to output in terms of the deliverables for each sprint. The sprint reviews overall went well for our team, minus the very first meeting with Alain. I think that there wasn’t that much that we could have improved in terms of how the sprint review meetings were done, although our team could have improved our organization before going into the sprint review, creating some sort of Agenda, or something to follow.

The scrums were very good to have I feel. We’ve talked about the Agile methodology in class extensively, but this was for a lot of us, our first actual time getting to develop following the agile method. I know that personally, there was a few scrum meetings that I went in to somewhat unprepared, either not remembering what I’d worked on the last time we got together, or not having picked up any tasks for that day to discuss at all during the scrum meeting. Another issue we had with the scrums was that because class time was so short, especially on Thursdays, the scums really cut into our time, and there were days when we just needed to get work done. So while overall, the scum meetings were very valuable, there were also times where they seemed almost like an extra burden to the team.

The sprints overall went quite well for our team. After the first sprint where more of our changesets were missing comments and/or tasks that were associated with them, things really picked up for our team for doing that, including keeping the time left for a task up to date. Another thing that I feel went very well over the course of all of our sprints was the taking of the meeting minutes during each of our meetings as a team. Having the recordings of decisions made and items discussed from each meeting proved to be very helpful when going back and trying to remember what we decided on especially during our client meetings. In terms of the sprints though, the ownership of tasks was all over the place. Tasks were constantly getting traded between people and bouncing around for several days before someone claimed it and finished it.

The length of the sprints for this project have been ideal. 2 weeks isn’t enough time to deviate off the path that we set on the last time we talked to our user, while still being long enough that we can get a solid list of deliverables out to our user. I feel like any longer of a sprint and we’d end up having worse time estimation since we have more PBIs to complete, and we’d end up making more mistakes in terms of making things that the user didn’t really want. So, I think that 2 weeks for the length of each sprint was ideal.

The sprint retrospective meetings I feel are very valuable to the agile process. Especially with a new team where we’re working to find our group dynamic, having that ability to openly talk about things that we feel are going well and what needs to be improved is hugely beneficial. For me at the very least, it’s a time to really think about everything and try to come up with something new to say. I know that a lot of people on our team often go into the retrospectives and just agree with what the first person said, but I feel having that opportunity to reflect and share is very valuable to me. The only issue is that once the retrospective is over, comments that were made during the meeting are never really addressed again.

I think that the biggest advantage to Agile is that it’s focused on being able to improvise, adapt, and overcome, changing user requirements. Users will very rarely have a perfect image in their head about what a system should be, and will often miss things when the team is trying to gather requirements for the system. The big drawback however is that with only enough time to get together our deliverables for the next sprint, we end up having some code whose quality is subpar, we end up with a lot of messy code that will solve a problem quickly, but not necessarily properly, and we end up with a lot of things not going the way we want them. Because of this, I think that we should modify our sprint process slightly by having the last 4 days of the sprint dedicated to bug fixing, and not implementing new functionality. This way, we end up with a system that’s much more robust in the end, adheres to our own standards much better and is overall of a much higher quality. We might get some deliverables out to the client a sprint later, but it saves a lot of time at the end of the process where we scramble to fix bugs before the system gets deployed.

The part of the project that’s been the most helpful to my learning for me, is having a much more formal development environment. Last semester when we worked on teams, we coded together and that was it. This semester, we’re really developing by the agile process, including scrum meetings in our classes, having sprint retrospective meetings, logging tasks in TFS, following our burndown chart, making off PBIs, etc. It’s much more representative of an actual environment than our development last semester in our teams.

## Teamwork

Overall, our team functioned well over the course of the development this semester and the work on our team was overall divided up relatively fairly. There were members of our team who took on more than expected of them, named Cody and Max, and members of the team who failed to meet my expectations, either on a personal level or my overall expectations of how a teammate should be. Brae and Andrew are both very strong individuals who I expected more out of them personally, but overall performed well as teammates. Louis I had little expectation of going in since I’ve never worked with him and he is overall a very quiet person, yet Louis I felt did not meet those expectations, taking work that was assigned to him and doing the minimum for that work, and then not really taking tasks once he’d finished up what he had been working on. So, I felt there was some people in the team who very definitely took a much larger burden in the workload than others on the team.

## Charts

A sprint burndown chart is a chart designed to show your team’s progress as you finishing work throughout the course of a sprint. What it shows is the amount of work remaining on the y-axis and the amount of time left in the sprint on the x-axis, and then shows the ideal trend that you’d like to be hitting over the course of the sprint. Burndown charts are very useful to all members of the development team in the agile process. It’s useful for everyone to know exactly how much needs to get done before the end of each sprint.

In our first sprint, we managed to stick on top of our ideal trend well all things considered. We didn’t get much any work done in the first couple of days, which resulted in us being above our ideal trend most of the sprint, but otherwise our decrease over time was very good. The graph never really flatlines out, which indicates that there was work getting done every day on our system by at least one member of our team, and you can see that even during the weekend the graph continues to go down.

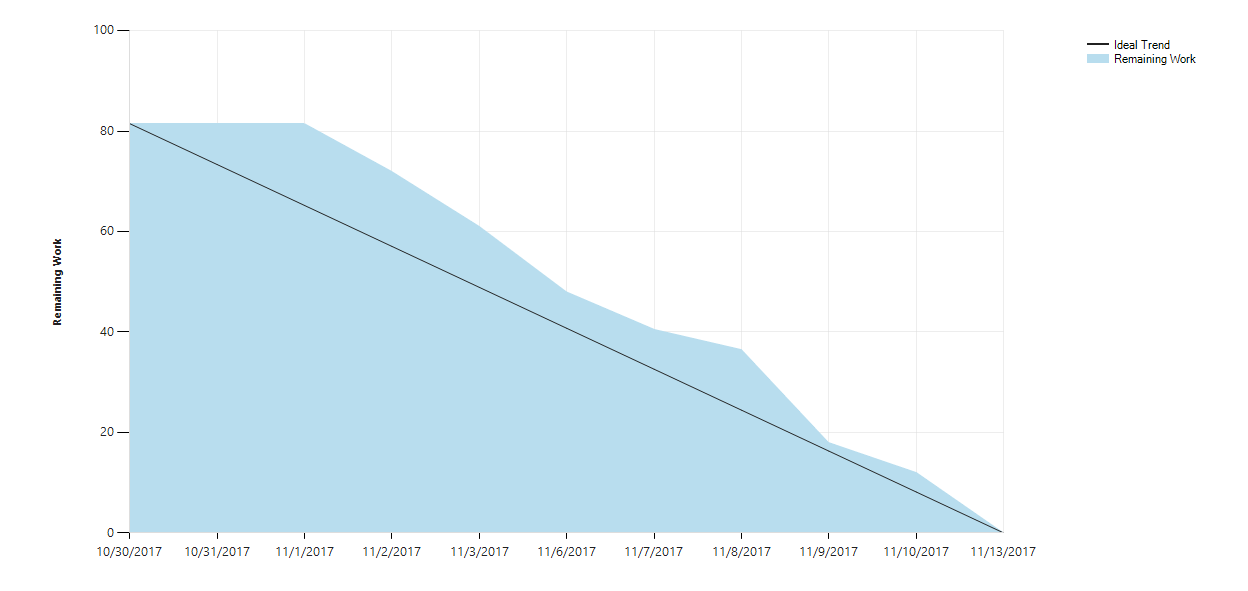


Figure : Sprint 1 Burndown Chart

Our second sprint’s burndown chart is not nearly as nice as the one that we had after our first sprint. We seemed to have gotten a decent amount of work done on the first Monday, but then no one really touched it until the weekend where we can see a massive drop. One of the things that you notice right away in this one compared to the others is that halfway through our sprint, the amount of work that we had actually went up. This was due to the team having overlooked some things in this sprint due to improper planning and we needed to go and create more tasks to account for changes that we hadn’t worked into our sprint planning. We then got a lot done the following class, and over the weekend to pull us towards our end goals for that sprint.

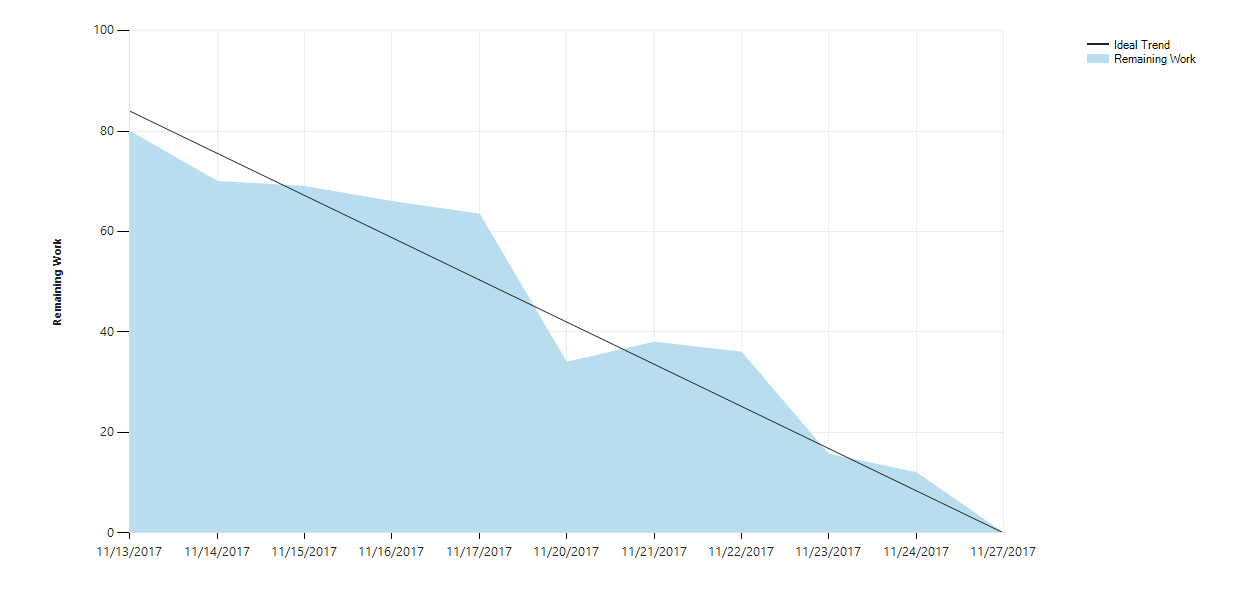


Figure : Sprint 2 Burndown Chart

The final sprint that we did we started off well, and then again had a massive drop over the weekend where we all worked on things. After that big push over the weekend though, we flatlined a bit for a couple of days, then made another push to get things done on the last Thursday of the sprint. After that we ended up getting a little bit done over the weekend, which isn’t reflected in the burndown chart due to a lot of unforeseen issues with the code that we ended up going back and marking tasks as uncompleted again.

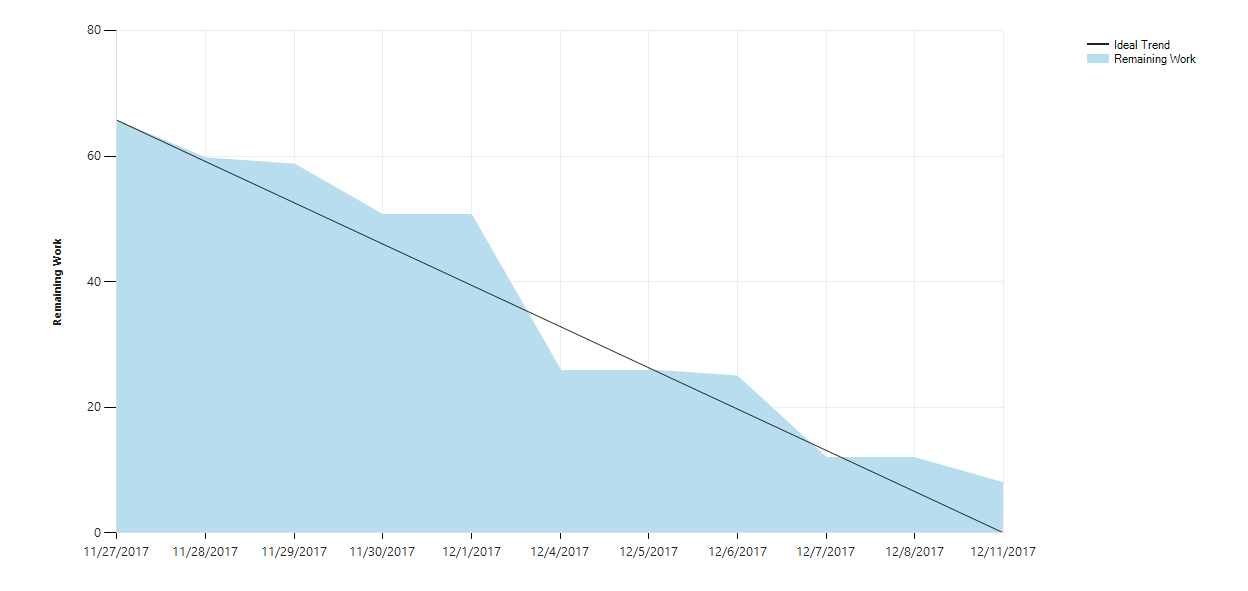


Figure : Sprint 3 Burndown Chart

One trend among all our burndown charts is that we tended to get a lot of work done during the weekend and never made as big of pushes during the week, which results in us having these big dips over what the graph shows as a 1 block period, and it’s only looking at the dates that you see that it’s a weekend there. This is because over the weekends we were able to sit down and focus on developing the system without needing to interact as much with other members of the team. While we’re all together we often spend a lot of time with a couple of us together brainstorming solutions to things, and the weekends are a dedicated time where we can just sit down and not need to worry about the other members of our team as much.

Over the course of the 3 sprints there was no improvement in our team’s ability to accurately estimate the amount of time required for each sprint. If anything, the trend that the graphs show is quite the opposite, in that as the sprints went on, our estimation tended to get worse and worse. This is likely because as the semester went on, we ended up with less and less time to work on our system because we had final assignments, tests, and studying to do. There was nothing that really could’ve been done to improve this I think, because at the beginning of every sprint we told ourselves that we over committed a bit in the last sprint and tried to adjust accordingly to make our sprints a little smaller, so this drop in our ability to estimate the sprints isn’t something that I think could have very easily have been fixed.

One things that the graphs reveal that should be addressed is that if you look at the changesets associated with those big pushes on the weekend, a lot of the major drops in amount of work is because of things that we’re done by only a couple members of our team. The amount of work done by everyone on our team outside of class time wasn’t very evenly distributed and we had a couple members making big pushes over the weekends to get things done. I know personally I was not one of those who put in quite as much effort on the weekends to get things done, but we need to try and distribute the work a little more evenly among the members of our team, because this is something that the burndown charts represent, although something that you only really notice as much when paired with the changesets from the team.

# Design and Build – Analysis

## Planned vs Actual

Our original architecture was relatively simple, abstracting the presentation from the presentation logic, the business logic, and then the rest of the application from the database through the data access layer. This was our original design because it’s not that dissimilar from how we’d modelled things before in past semester with our Presentation/Business Logic/Data Access. However, once development started and we began to both understand Entity framework a little better and realize some of the issue in our model coming out of it, we needed to make some minor adjustments to the architecture model.

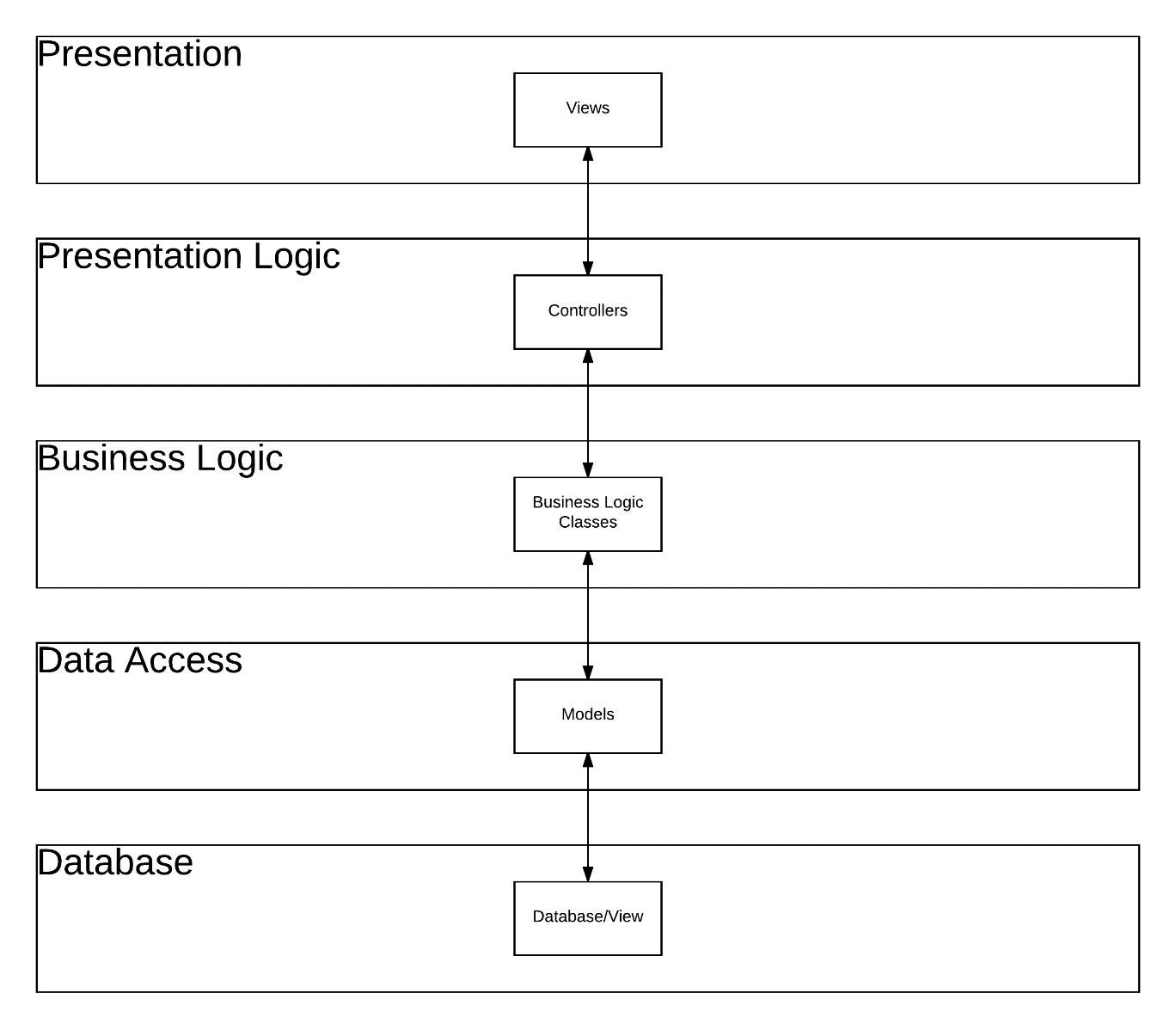


Figure : Original Layered Architecture

Coming out of our 3rd sprint, this is what our architecture model resembles now. Our database is abstracted away from us almost entirely through Entity Framework, and then to account for our model’s getting completely reset when we change the Data model, we added a validation model in the middle to ensure that everything going from our presentation to our business logic and data access layers were data that we would exact. So this layer contains classes that perform validation and sanitization of the data going through it.

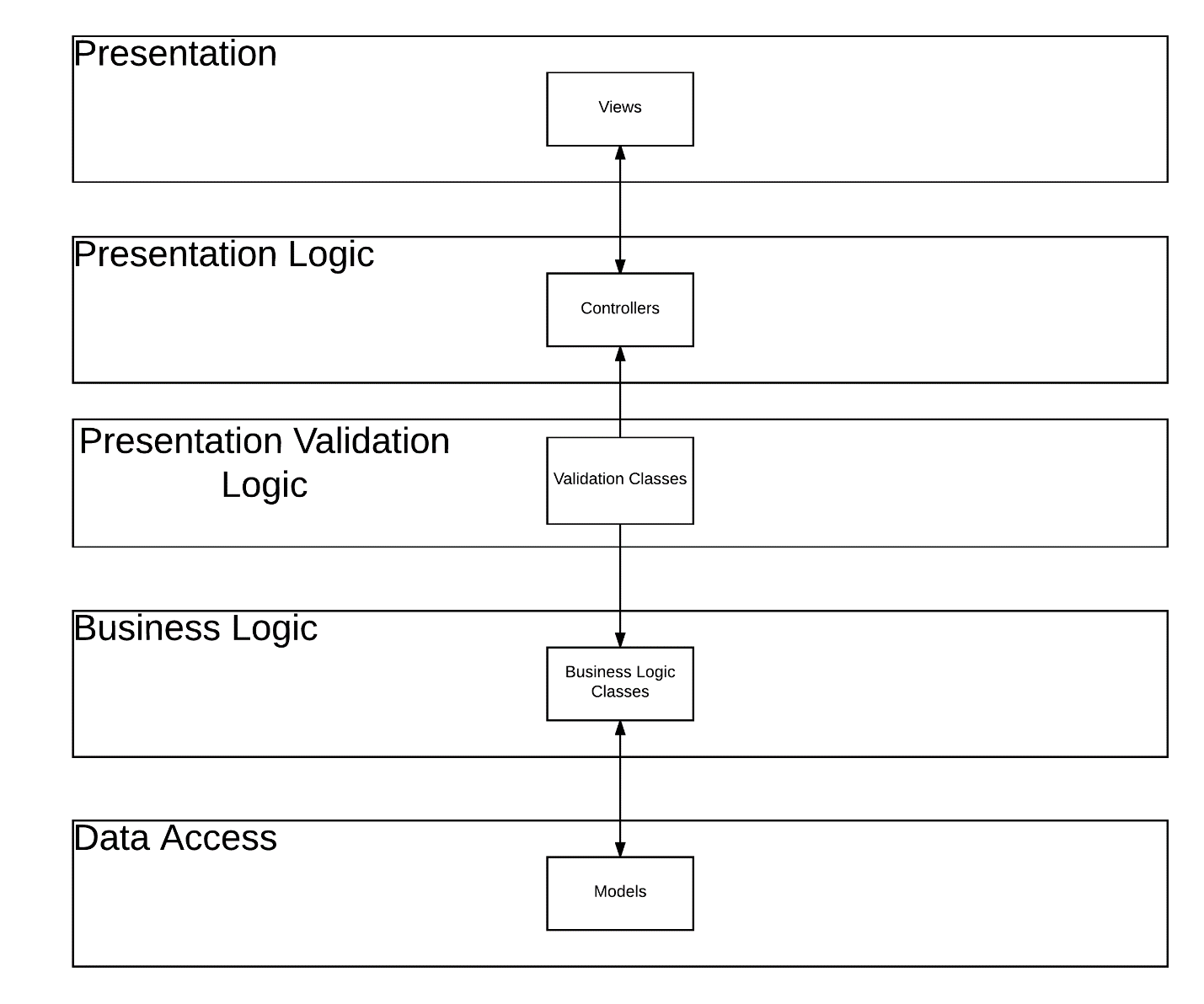


Figure : Current Layered Architecture

The original model we had was almost right, but it didn’t account for the way Entity Framework hits the reset button every time we want to make a slight change to our data model. Going back, I think that the only real thing that we could have done to have improved our original modelling would be have had a better understanding of Entity Framework and the model first method. We knew the concept behind it, but we didn’t know how to use it at all and we didn’t know how to model with it in mind, so the only thing that really could have changed our mistakes would’ve been learning Entity Framework earlier on.

Our original class diagram, much like our architecture model, ended up quite similar to our current version of diagram. The diagram here was supposed to have the 3 different types of users in our system, with all of the data tied to the Candidate through the RAC request. This diagram went through many changes throughout the development of our system, with the Notifications getting removed since it’s all handled through email and we didn’t need to track that in our own system. Once we figured out the Clara contained the Programs and Competencies, we also removed that whole set of table from our diagram, only to go an re-add it again later on once we realized that the college only offers RAC for certain programs and that the Competency Elements don’t exist in the database. The original class diagram mostly actually works for our system with some minor tweaks to it, but we had to step through a lot hoops to wind up with the class diagram that we are currently using.

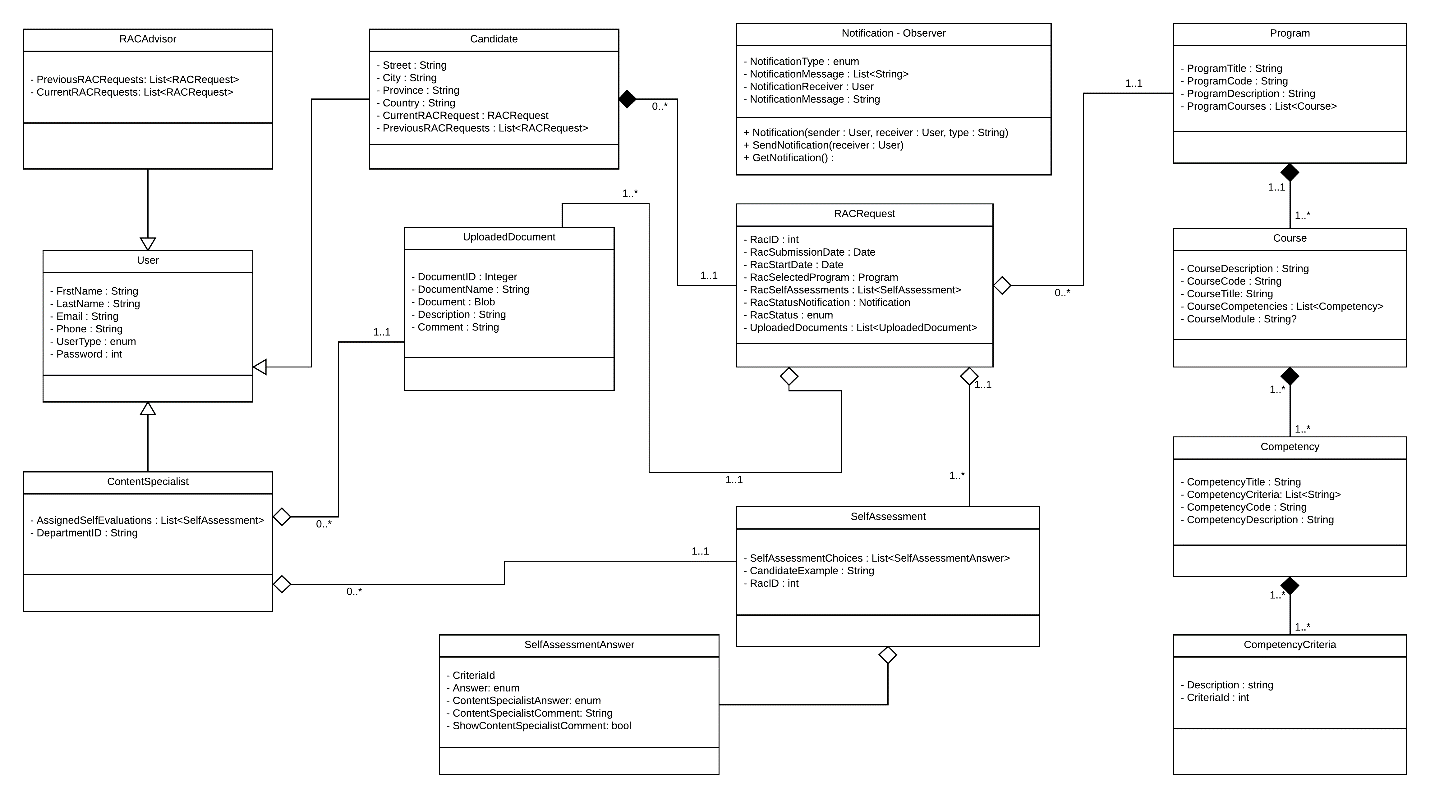


Figure : Original Class Diagram

Our current class diagram represents all of the changes in user requirements and understanding of the process that we’ve acquired over the course of the development. The way that we’re handling the self assessments for a RAC request in our new model makes it so that there is a straight mapping of the Program to the RAC request, and then the competency to their comments for it, and the competency elements to their answer for that element of the competency. This is all tied to the RAC request, which is tied only to the Candidate in this iteration, it’s not longer tied to the ContentSpecialist as well, which I’m honestly unsure of why it was in the first place. The new class diagram represents only the model for our layers in the architecture, and this is mapped directly to the data model in the DB as well, so this is representative of our data model.

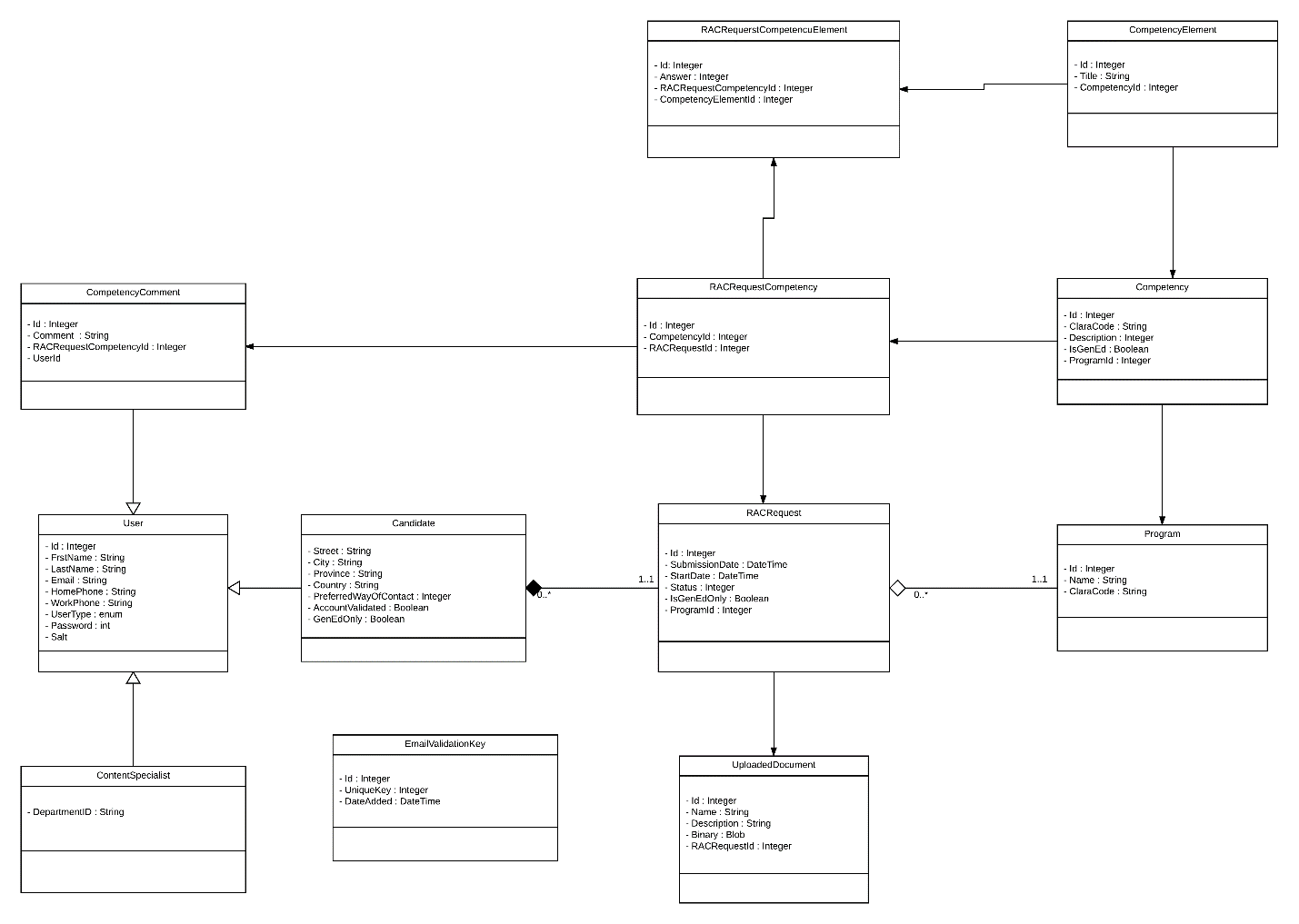


Figure : Current Class Diagram

We could have gotten our class diagram done right from the start I think if we’d had the time to do it properly. Because of the speed that we went through each step in the process before development, there was a lot of mistakes that as a team we knew existed in our class diagram, but considering all of the changes away from our original diagram we ended up seeing through development, only to come back to something very similar, we ended up doing a good job of modelling this the first time around.

## Refactoring

Code refactoring refers to different places within a system that you can go back and modify the existing code to improve it, without changing its functionality at all. This can mean going in and making code more readable by reducing its complexity, breaking things down into multi-line statements, changing code to meet the proper conventions in place, and overall cleaning up the code. Refactoring can also be re-writing certain pieces of functionality to improve their efficiency if they’re running particularly slow. This can all be observed in a profiling tool, so refactoring can be a big part of code optimization.

There are probably dozens of major re-factorings that can be done on the code base for our system, since in some places we really did have an agile mindset in terms of “make it work now, go back and fix it later”.

One place that the code needs a serious cleanup is in the \_CandidateHome.cshtml file. This file contains both embedded CSS and JavaScript. In both cases, the code should be pulled out of the view and into their own CSS or JavaScript files respectively. This would improve the code in serval ways. First of all, the view itself would be cleaned up a lot by taking out everything that isn’t directly related to it. Putting the JavaScript and CSS into other files however would also mean that you could then re-use that code for all the views that need to contain these competency tables, meaning you’d be avoiding writing all of this duplicate code for your views. While the \_CandidateHome.cshtml page is on View that could have some of this code pulled out of it, it’s certainly not the only view that has embedded CSS and JavaScript. This refactoring applies any page that contains embedded code, besides the Razor built into the page.

Another place that refactoring needs to be done is in the uploaded\_document.js file. JavaScript has the wonderful ability to nest functions one inside the other to improve scoping, however this has wound us up having a single function called initializeDropZone() in that file that contains almost 100 lines of code. This monstrous function needs to be broken down into each individual section of the function performing the one operation. This would make the code far more modular, reusable and readable. Now, the function is way too large and needs to be broken down.

A final place that the code needs some refactoring is in the CandidateBLL class. Again, this class contains the one massive Register() function containing just shy of 100 lines of code. One of the first things that this function does is it creates a new Candidate object, using the Candidate object that was passed into the function. Minus the UserType, Salt, and Password getting set, the two objects are identical. This is very confusing seeing as there’s absolutely no reason for this second redundant Candidate object. You can just set the properties of the one passed in, instead of making a whole new one. This dates to the very first changeset associated to this file, so I’m not sure why it was done like that, but it needs to get changed.

# Testing and Validation

## Black-Box testing

The black-box testing of the system that was done during the development of the system was relatively minimal. We tested each functionality of the system to ensure that it worked following a happy path, but we never really did testing to ensure that the system could hold up in a variety of different circumstances or much outside of the linear cases. We also made sure that each function of the system worked on its own, but not the impact that it had on everything around it, so we ended up with a lot of functionality butting heads to work. This differs greatly from system testing because the goal of system testing is to make sure that the flow through a whole system works in it’s entirety, making sure each step in the system works the way that it should and that it all meshes together to form something functional. We tested our basic flows through the system, but never tested edge cases, exploring what would happen if the user started messing around with the system or how it would hold up in some more unusual scenarios, so our testing of the system was not very complete in terms of system test cases.

### Functional Testing

#### Test Case Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Purpose | Expected Result | Actual Result |
| FU\_01 | Make sure that a new user is able to create an account in the system correctly. | The user fills out the registration form, choses which program they’d like to enroll in, fills out the captcha and registers. | The user cannot register, because the captcha is not registered under the CSDEV domain, and therefor will not work. |
| FU\_02 | Make sure that a user can log into the system properly | When the user goes to log in with a user that exists in the system, if they enter a valid email address and password then the system will log them in. | The user enters their email and password and when they click log in they are brought to the self-assessment page if their log in information was valid |
| FU\_03 | Make sure that a user can only log in to an existing account with the proper credentials | If a user enters an existing email into the system, but an invalid password, the user will not be authenticated. | The user cannot log in if they do not enter the proper password to an account. |
| FU\_04 | Ensure that the candidate can only see one of the types of competencies at a time to make sure that the vertical size of the page never gets too large. | When the user goes to fill out the self-assessment form, the user will only be able to see one of the types of competencies at once: General education OR program specific. | When the user is viewing the self-assessment form, the user can see just the program specific when on that tab, but the program specific competencies will also appear underneath the general education ones when on that tab. |
| FU\_05 | Make sure that the ‘jump to top’ and ‘jump to bottom’ buttons work correctly. | When you click on the ‘jump to top’ button, the page will scroll smoothly up to the top, and when you click the ‘jump to bottom’ button, the page will scroll smoothly down to the bottom. | The buttons function as expected. |
| FU\_06 | The user can attach new files to their RAC request | When a user drags a file into the upload box and click the upload button, the file(s) are successfully added to the list. | The user is able to upload files to the system properly and all uploaded files are added to the list of files attached to that user’s profile. |
| FU\_07 | The user can delete files from their RAC request. | When a user clicks the ‘Remove File’ button, the file will be removed from their list of files and it will be deleted from the RAC request in the back end. | When a user clicks to remove a file, nothing happens |
| FU\_08 | The user can only see the data that they should be seeing. | On the view completed RAC request page, the rows for their own comments and the RAC advisor’s comments will only be visible to the Candidate if they have been filled out. The Candidate also should not have access to seeing the Content Specialist’s comments. | On the page to view the completed RAC requests, the comments row is displayed whether a comment was entered by the Candidate or the RAC advisor. The Candidate should also not see the Content Specialist’s comments at this point. |
| FU\_09 | Make sure that a Candidate can log off the system properly. | When a user logs out of the system, they are brought back to the log in page and cannot navigate as the previously signed in user. | When a user logs off the system automatically navigates them back to the log in page, but once at the log in page that user can click on the home button to navigate back onto the self-assessment page again. |
| FU\_10 | The view submitted RAC request page should display all of the proper data for the RAC request, as entered by the user. | The data submitted, answers to all the questions, comments, and documents should appear in the page as they were entered in the candidate home page. | The data submitted shows up as having not yet been submitted, and the section for the uploaded documents doesn’t currently exist, so the data for the RAC request isn’t all accurate. |

Table : Functional Test Cases

### Usability Testing

#### Test Case Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Purpose | Expected Result | Actual Result |
| US\_01 | Ensure that the ‘jump to top’ and ‘jump to bottom’ buttons work appropriately. | When the whole page is in view, neither of the two buttons appear. When the user is not at the top of a page, the ‘jump to top’ button appears in the bottom right of the page. When the user can scroll down, the ‘jump to bottom’ button will appear underneath the ‘jump to top’ if it’s visible. | When the whole page is in view, the buttons work as expected. However, on all pages besides the CandidateHome, but buttons appear awkwardly at the bottom of content and not always sitting off to the side for the ease of use of the user of the page. |
| US\_02 | Check that all the fields on the registration form are nicely aligned and everything looks appealing to the user. | The registration form appears nicely and all fields on the form are aligned to the left with the label associated to them | The checkbox to apply for RAC under only the general education competencies is very large, and it’s centered under all the textboxes that appear above it. This is not at all appealing for a user to look at. |
| US\_03 | When a Candidate goes to the self-evaluation form, the greeting message generated for them should be grammatically correct. | The sentence and proper names are uppercased, proper user of grammar, etc. | The name appears as entered by the user when registering, so if they entered their name as lowercase, then their name will appear lowercased in the greeting. Names should be capitalized before going to the database. |
| US\_04 | Make sure that the system appears, and functions as should in Opera | The page looks the same in Opera as it does in Chrome. | On the registration page, the ‘general education only’ checkbox has a big thick ugly border all around it that makes it stand out in the worst way. |
| US\_05 | Make sure that the system appears the same on Edge | The page looks the same in Edge as it does in Chrome. | The ‘Jump to top’ and ‘jump to bottom’ arrow button appear side by side instead of one on top of the other as they do in every other browser. |
| US\_06 | Make sure that the system appears the same in Internet Explorer | The page looks the same in Internet Explorer as it does in Chrome | In Internet Explorer, the ‘Jump to top’ and ‘Jump to bottom’ buttons will appear all the time on every page, not matter how large the page is. The buttons also will not work, when you click to jump nothing will happen. |
| US\_07 | Make sure that the system appears the same in Firefox | The page looks the same in Firefox as it does in Chrome. | The ‘Jump to top’ and ‘jump to bottom’ arrow button appear side by side instead of one on top of the other as they do in every other browser. The ‘General Education only’ checkbox also has that big thick border around it like it does in Opera. |
| US\_08 | Make sure that on all pages, any field that is an enum will be displayed with it’s proper display name. | On all pages, anywhere where there is an enum as the source of the data, that field will appear properly with spaces and not using the enum’s name with the camel case. | On the page to view a submitted RAC request, the right-most column contains the user’s answers to each of the self assessment questions. All fields appear in Upper camel case in their enum names and not separated by spaces as they should be. |
| US\_09 | Each page is templated properly with the same header, footer, layout, ‘jump to top’, ‘jump to bottom’, color scheme, etc. | Each page is uniformly made with no glaring differences in design from one page to the next. | On the page to view a completed RAC request, the page appears with a footer and jump buttons at the bottom of the form, and then underneath the footer is the templated footer and jump buttons, so they appear twice for some reason. |
| US\_10 | Ensuring that validation works on various fields across the system. | When a user submits a file with an invalid extension, the system gives a proper error stating they can’t upload files of that type. | When a user submits a .zip file (as an example), the system displays an alert simply stating that there was an internal server ever with no further help. |

Table : Usability Test Cases

### Exploratory Testing

#### Test Case Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Purpose | Expected Result | Actual Result |
| EX\_01 | Entering the URL from to go to the site. | The page will load the same with or without the slash, the routing will take care of it. | Entering the URL for self-assessment without the slash at the end will yield an error that brings up an alert box telling the user error Not Found, with no further details. There’s also 2 404 errors in the JavaScript console. |
| EX\_02 | Once you’ve click submit a RAC request, you can’t go back to editing it again. | If you click the ‘submit RAC request’ button, log off, and come back you’ll be brought to the submitted RAC request page. | The system will direct you right back to the self-assessment form, which is an outcome unexpected considering earlier the system was forcing the submitted, and in this scenario it’s denying the submitted RAC request page. |
| EX\_03 | When you make a change in one tab and save it, then refresh another tab the change should appear. | When you make a change to a RAC request in one tab, then save it, the RAC request in both tabs will be update with the new self-assessment data. | When you make a change to a RAC request, not even the one in the tab you’re currently in will get properly submitted. |
| EX\_04 | Making sure uploading files works with all methods and valid files. | You can drag and drop multiple files into the drop zone at once before uploading, you can select multiple files at once from the file explorer and select them, and you can do any valid files in your upload. | When you click to upload a new file instead of dragging and dropping one in, it does nothing. |
| EX\_05 | Make sure a user can cancel an upload before the file gets fully uploaded. | The user can click “cancel upload”, and the file will disappear, and they can continue uploading new files afterwards. | When a user cancels an upload, the canceled will file will stick around inside the dropzone and won’t leave until the user leaves the page. As long as that cancelled file is in there, the user can’t upload new documents. |

Table : Exploratory Test Cases

### Defects

#### Defect Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Severity | Priority | Steps to reproduce |
| DEF\_01  FU\_01 | The captcha isn’t registered with google under the CSDEV domain, so it’s not going to work off localhost until we get it registered with google. | 1 | 3 | * Navigate to the registration page. * Attempt to fill out the captcha – there’s an obvious error in it. |
| DEF\_02 FU\_04 | The user can see both the program specific and the general education competencies when they have only the general education ones selected. | 3 | 2 | * Create a user without general education only if one has not already been created. * When viewing the self assessment form, switch tabs to the general education competencies and scroll down a little bit, and you can see the program specific ones right underneath. |
| DEF\_03 FU\_07 | When a user clicks to remove a file, nothing happens | 3 | 3 | * Log in with an existing user who is in the process of filling out a RAC request. * Upload a new document to the user’s RAC request * Once the file has been added to the list, click the delete button. Nothing happens. |
| DEF\_04 FU\_08 | On the page to view the completed RAC requests, the comments row is displayed whether a comment was entered by the Candidate or the RAC advisor. The Candidate should also not see the Content Specialist’s comments at this point. | 4 | 4 | * Log in with an existing user. * Submit the RAC request if it has not been submitted. * Scroll to the bottom of the first competency for that user, and notice the comments that appear there. |
| DEF\_05 FU\_09 | When a user logs off the system automatically navigates them back to the log in page, but once at the log in page that user can click on the home button to navigate back onto the self-assessment page again. | 3 | 1 | * Log in with an existing user. * Log out of their account. * Click the home button, and you will be navigated back to the previously logged in user’s page, and not the RAC system home page. |
| DEF\_06 FU\_10 | When the user is on the view completed RAC request page The data submitted shows up as having not yet been submitted, and the section for the uploaded documents doesn’t currently exist, so the data for the RAC request isn’t all accurate. | 3 | 3 | * Log in with an existing user. * Submit the RAC request if it has not already been submitted. * Go to view the completed RAC request * Note the data submitted at the top of the page. This page should never appear if the date isn’t submitted. * Note that the Upload documents panel isn’t at the bottom of the page either right now. |
| DEF\_07 US\_01 | The ‘jump to top’ and ‘jump to bottom’ buttons appear awkwardly underneath content on the left-hand side of the page | 4 | 2 | * Go to the registration page * Scroll to the bottom and note the buttons with the arrows point up and down inside them. |
| DEF\_08 US\_02 | The checkbox for the ‘General Education Only’ is far too large and sits aligned to the center on the registration page | 4 | 3 | * Go to the registration page * Scroll down to where the checkbox appears and note it’s size and the checkbox’s alignment compared to the rest of the fields in the form. |
| DEF\_09 US\_03 | The user’s name appears as entered in the registration form when being displayed, so if they entered is lowercase, it will appear lowercased. | 4 | 2 | * Go to the registration page. * Create a new user (if you could), and give their name improper casing * Login as the user and visit the self-assessment page * Their name in the welcome message will be as entered in the form. |
| DEF\_10 US\_04 | The registration page displays poorly in Opera. The ‘general education only’ checkbox has a big thick ugly border all around it that makes it stand out in the worst way. | 4 | 4 | * Open the system in the Opera browser * Navigate to the registration page to create a new user. * Note the big border around the checkbox. * This error is different from the previous comment about the checkbox, since this is about the border which would not change with just a re-positioning of the checkbox. |
| DEF\_11 US\_05 | In Edge, the ‘Jump to top’ and ‘jump to bottom’ arrow button appear side by side instead of one on top of the other as they do in every other browser. | 4 | 4 | * Open the system in Microsoft Edge browser. * Navigate to any page large enough for the scroll to appear. * Not that the ‘Jump to top’ and ‘Jump to bottom’ buttons are side by side instead of the up on top of the down. |
| DEF\_12 US\_06 | In Internet Explorer, the ‘Jump to top’ and ‘Jump to bottom’ buttons will appear all the time on every page, not matter how large the page is. The buttons also will not work, when you click to jump nothing will happen. | 3 | 4 | * Open the system in Internet Explorer * Immediately you will notice the ‘Jump to top’ and ‘jump to bottom’ buttons on every page. * Navigate to the registration page (or another page long enough to use them) * Try clicking on the buttons to jump up or down. Nothing happens. |
| DEF\_13 US\_07 | The ‘Jump to top’ and ‘jump to bottom’ arrow button appear side by side instead of one on top of the other as they do in every other browser. The ‘General Education only’ checkbox also has that big thick border around it like it does in Opera. | 4 | 4 | * Open the system in Mozilla Firefox * Navigate to the registration page. * Notice the same error in Opera appears on the checkbox here. * Notice the same error with the jump buttons as in Edge appear here too. |
| DEF\_14 US\_08 | On the page to view a submitted RAC request, the right-most column contains the user’s answers to each of the self assessment questions. All fields appear in Upper camel case in their enum names and not separated by spaces as they should be. | 4 | 2 | * Log in with an existing user. * If not already done, submit that user’s RAC request. * Note the answer’s column contains upper camel cased strings instead of separating them by spaces. |
| DEF\_15 US\_09 | On the page to view a completed RAC request, the page appears with a footer and jump buttons at the bottom of the form, and then underneath the footer is the templated footer and jump buttons, so they appear twice for some reason. | 4 | 2 | * Log in with an existing user. * If not already submitted, submit their RAC request. * Navigate to the view completed RAC request page. * Scroll down to the bottom of the page. * Not the double footer and the two sets of jump buttons. The page appears to have the templated footer and a manually added one. |
| DEF\_16 US\_10 | When a user submits a .zip file (as an example), the system displays an alert simply stating that there was an internal server ever with no further help. | 4 | 2 | * Log in with an existing user. * Navigate to the Self assessment form. * Scroll down to the upload documents section. * Try adding a document without a valid file extension. * Note the simple alert box with the non-descript error. |
| DEF\_17 EX\_01 | Entering the URL for self-assessment without the slash at the end will yield an error that brings up an alert box telling the user error Not Found, with no further details. There’s also 2 404 errors in the JavaScript console. | 1 | 3 | * Log in with an existing user. * In the URL, remove the slash from the path. * Hit enter to refresh the page. * There will be a pop-up with a non-descript error in it. * The JavaScript console contains errors and the page doesn’t work. |
| DEF\_18 EX\_02 | The system will direct you right back to the self-assessment form, which is an outcome unexpected considering earlier the system was forcing the submitted, and in this scenario it’s denying the submitted RAC request page. |  |  |  |
| DEF\_19 EX\_03 | When you make a change to a RAC request, not even the one in the tab you’re currently in will get properly submitted. | 1 | 2 | * Log in with an existing user who’s RAC request has not yet been submitted. * Scroll to the bottom of the page and click the submit RAC request button. * After the page redirects you, click home and you’ll be brought back to the self-assessment page. You cannot submit this way. |
| DEF\_21 EX\_05 | When a user cancels an upload, the canceled will file will stick around inside the dropzone and won’t leave until the user leaves the page. As long as that cancelled file is in there, the user can’t upload new documents. | 3 | 4 | * Log in with an existing user who’s RAC request has not yet been submitted. * Scroll to the bottom of the page * Open up the file explorer and drag a new file in upload. * Before the upload can finish, click the cancel button. * Upload a second document. * Try clicking the upload documents button with the valid and the cancelled document, and you cannot submit. |

Table : Defects

Test Results:

|  |  |  |
| --- | --- | --- |
| Type of test | Number of test run | Number of tests passed |
| Functional | 10 | 6 |
| Usability | 10 | 0 |
| Exploratory | 5 | 0 |

Overall, while we managed to get a lot of features built into the system, all the features were of relatively low quality. There are many more bugs in the system beyond what I have documented here, and a lot of time will need to get spent fixing these bugs and making a much more robust system before this ever goes out to production. When we come back from the winter break, we should have a sprint that we dedicate to fixing bugs, because the system as is will not be functional if we continue developing as we are without fixing all the bug’s we’ve created.

## White-Box Testing

The purpose of unit testing for systems like this is to make sure that all the business logic works in all cases. While you can test certain functions by using the system, writing unit tests you can isolate different function in the code to make sure that every function works properly. This means that even if the whole system doesn’t work, you can ensure that each individual component does, before tying it all together. This is why integrating both black box and white box tests is so important. One white box testing technique is called test driven development. This is a methodology where you write your test cases prior to the code. One advantage to this is method is that you have no preconceptions about how well your code really works, so you’re more likely to write more thorough test cases. Some of the cons to white box testing are that in Agile development, the developers are also the testers, so developers who have attitudes that their code works might not be as thorough in their testing, which leads to flaws in the system where edge cases aren’t handled or there’s some subtle unexpected behaviors.

When writing unit tests, you want to aim to get as much of your code tested as possible, to ensure that all your code works. This metric is defined in terms of code coverage, which is the percentage of your total code base that is covered by your unit tests. You want to have as high a code coverage as possible to ensure that everything in your system works, however when you don’t have the time to do that, you need to make trade-offs somewhere. Some of the first code to not be tested will be anything facing the user in the front end – in the case of MVC, pretty much anything existing in the View or the Controller will not be tested as much because we can determine if this code works much more easily in our black-box testing techniques.

The code coverage for our system is pretty weak. We currently have 26 unit test methods overall, which is low as it is, and of those 26 unit test methods, only 20 of them are passing. This means that we have only 77% of our code working, which if these 26 unit tests are representative of the whole system, we could determine that only around 77% of the code that we’ve written actually works.

Testing for a new functionality should be done as early as possible while you’re writing that piece of code. Making sure that a function works as intended is much easier to do as you’re developing it rather than doing it right at the end once it’s done. Testing a function only after all the code has been written will leave you more likely to have bugs in that function, and make the bugs hared to find since the code will be more complex. Our tests we’re run when they we’re written and once the code they were testing was finished. Our unit testing for this system so far has been weak, and we have not been following testing standards appropriately. We did run old tests, since we usually ran all tests at once, but we often had a number of tests failing.

In conclusion, we did a poor job of testing this system. We currently have a low code coverage percentage, with a lot of tests that are failing. We are not testing appropriately seeing as we’ve been almost only running our tests once the functionality is finished and not periodically as we should be. As things change, testing should continue to happen, which is not the way we’ve been handling things right now.

# Conclusion

Overall this course has so far been incredibly beneficial to my learning of how to function in a real-world environment. The project has been teaching me how to properly work within a team, both sometimes acting to help lead the team, and sometimes working as a member of it to develop this system. The project has taught me a lot about working in an agile environment that I never really grasped from any of the learning that we did in systems class. While what we’ve done so far isn’t 100% representative of what a real environment would look like, the amount of trade-offs that we made in getting more functionality packed into the system for the sake of having demonstrable code was way higher than I ever would’ve imagined. We often came up with poorly designed solutions in the mindset that it works now, we’ll make it better later, which is not a mindset that I’m used to developing in when working on assignments individually. I’m much more meticulous, making sure to meet all acceptance criteria and in some cases even following a TDD method.

I feel like this new knowledge of the Agile process is really the biggest thing that I’ve taken out of this course, and understanding how to work in it so that we can improve how we adapt to agile in our next semester where hopefully we’ll do much better in it.

I think that there’s not that much that could’ve been a whole lot better in this course. We spend a lot of time at the beginning of web reviewing JavaScript and AJAX which resulted in us learning MVC much later and not fully understanding the paradigm when beginning the development project, and I think having learnt MVC sooner would’ve been incredibly beneficial. However, that was something that was unintended in the web programming course that ended up impacting us in development, but that shouldn’t happen in the future. I would absolutely recommend this class to future students. You learn a lot of lessons that aren’t taught by the teacher, and that you need to learn from actually experiencing using agile. At the beginning of the course I didn’t see it’s benefit nearly as much, but nearing the end of the 2nd sprint, I realized just how much I had learnt from the course and how beneficial it’s been to me.