**Course Information Forms Management System**

**Version 1.0**

Jesse Luo

**Table of Contents**

[Introduction 3](#_Toc532833012)

[Purpose 3](#_Toc532833013)

[Scope 3](#_Toc532833014)

[Definitions, Acronyms, and Abbreviations 4](#_Toc532833015)

[Context Diagram 5](#_Toc532833016)

[Software Requirements Specifications 6](#_Toc532833017)

[Functional Requirements 6](#_Toc532833018)

[Non-Functional Requirements 10](#_Toc532833019)

[Domain Model 12](#_Toc532833020)

[Use Case Diagram 13](#_Toc532833021)

[Use Case Descriptions 14](#_Toc532833022)

[Activity Diagrams 19](#_Toc532833023)

[Robustness Diagrams 26](#_Toc532833024)

[Sequence Diagrams 33](#_Toc532833025)

[Set-Up 40](#_Toc532833026)

[Tools 40](#_Toc532833027)

[Setting Up ColdFusion 40](#_Toc532833028)

[Setting Up a Database Instance on Amazon Relational Database Service 42](#_Toc532833029)

[Accessing the ColdFusion Administrator Panel 45](#_Toc532833030)

[Setting Up SQL Server Management Studio 45](#_Toc532833031)

[Setting Up CFEclipse 46](#_Toc532833032)

[Integration 47](#_Toc532833033)

[Conclusion 48](#_Toc532833034)

[Challenges 48](#_Toc532833035)

[Achievements 48](#_Toc532833036)

[Results 48](#_Toc532833037)

# Introduction

The Course Information Forms Management system is an application that exists as a satellite feature to the Virtual Student Advisor ecosystem. Course information will be captured, stored, and maintained with this system, which will then be used by Virtual Student Advisor and other users. The system features the ability to add or edit courses and their attributes and see an archive of previous versions for a course. Three levels of users are present so that course additions and/or changes must be approved when they are submitted by a lower level. This application is relevant because it provides a front-end for faculty and admin personnel while providing a back-end connection with the Virtual Student Advisor. The application is also extensible and potentially useful to other community colleges.

The purpose of this application is to standardize and centralize course information at the Everett Community College. Courses were originally stored on MS Word documents of varying versions and in different formats across different network directories. Thus, for this application, the solution would be to provide a user-friendly front-end to capture and manage course information, provide a back-end connection to a central database to maintain course information and different versions, and make the application extensible to internal Everett CC applications and external applications such as the Virtual Student Advisor.

## Purpose

The purpose of this document is to define the specifications and design of the Course Information Management application.

## Scope

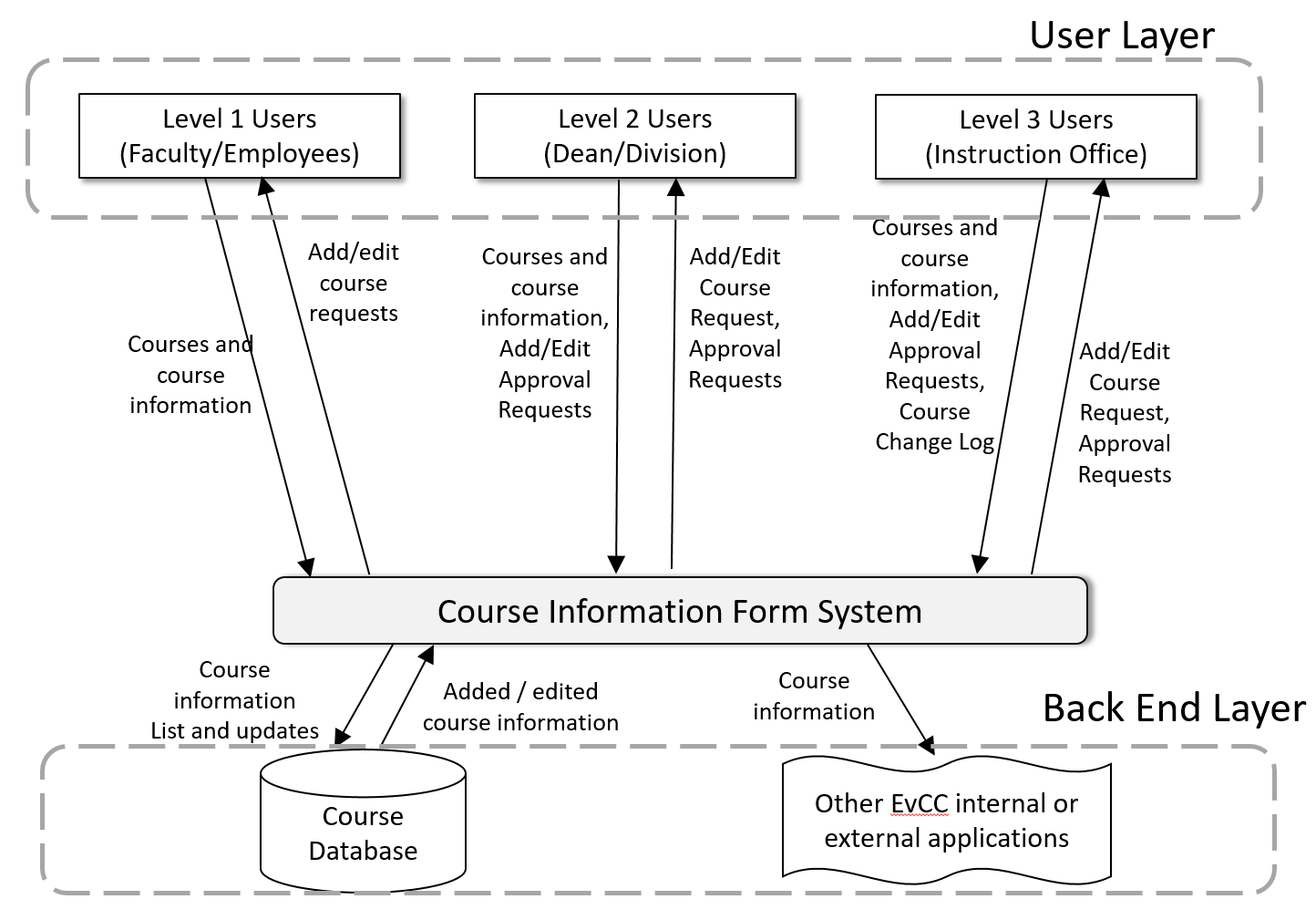
This document will focus on the features of the Course Information Management system, which are adding and editing course information, there different levels of users, parsing course information documents, viewing past versions of a course’s information, downloading a course’s information, and viewing course information that is present in the system.

Users of this application will log in with a username and password. Depending on their level of clearance, different options/tabs will be available to them on the front screen. These different levels of clearance are important to ensure the integrity of the course information stored in the database and ensure any adds or edits can be reviewed and approved by users of high level of clearance.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Web Page | The web page is the front page all users see when they first enter the application. At first, a login page will be shown. After logging in, different tabs will be shown depending on the user’s level of clearance. |
| User | A user is an employee of a college that uses this application. Users are split into three levels: Level 1 Users (Faculty/Employees), Level 2 Users (Dean/Division), and Level 3 Users (Instruction Office). |
| Tab | A tab is one of the options present on the web page that a user can click to perform a specific task. There are 4 possible tabs a user can access depending on level of clearance: “Add Course”, “Search for Course”, “Approve Add/Change Request”, and “View Change Log”. |
| Parser | The parser takes an uploaded course information document, parses the relevant data fields from it, and returns it to the web page for review. |
| Course | The course and the information associated with the course. |
| Database | The database stores all the course information and user information. |

# Context Diagram



The goal of the CIF management system is to allow the capture, storage and maintenance of course information. The main flow of data is course information from the CIF Management system to the users, database, and Virtual Student Advisor. Course information data flows from the users to the system when they add, edit or search for information, and when additions or edits are approved by a Level 2 or 3 user. Course information flows to the database when additions or edits are approved and when course data is searched for. Course information also flows to and from the Virtual Student Advisor.

# Software Requirements Specifications

## Functional Requirements

The following functional requirements are categorized into three different categories. These categories are the course information web page, course information database, and course information parser. The course information web page section concerns features of the web page that the user can interact with. The course information database section concerns what kind of data and relations the database holds. The course information parser section concerns the functionality of the parser when it comes to retrieving the relevant information from the course information documents.

**Web Page**

|  |  |
| --- | --- |
| **Requirement ID.** | **Description** |
| FR\_CIWP\_1.0 | The web page will have a login page and users will have different levels of access and permissions regarding course adds and changes. |
| FR\_CIWP\_1.1 | Level 1 Users (faculty and employees) will be able to submit an add request to add a new course, and make a change request for existing courses. |
| FR\_CIWP\_1.2 | Level 2 Users (Dean/Division) has the same permissions as Level 1 Users, and will be able to, via an admin tab, review and approve add and change requests, which are then passed to a level 3 user for final approval. |
| FR\_CIWP\_1.3 | A Level 3 user (Instruction Office) can add or change courses, and makes the final approval for all add or changes requests made on the system. |
| FR\_CIWP\_2.0 | The user should be able to add/submit an add request for new courses. |
| FR\_CIWP\_2.1 | The user should be able to add/submit an add request for new courses by filling out a web form. |
| FR\_CIWP\_2.2 | The user should be able add/submit an add request for new courses by uploading a document. |
| FR\_CIWP\_2.3 | The web page will display the parsed information for review and editing by populating the data fields in the webform. |
| FR\_CIWP\_3.0 | The user should be able to search for a course by using a web form and filling out one or multiple fields. |
| FR\_CIWP\_3.1 | A level 2 or 3 user should be able to see a history of edits and changes for a specific course. |
| FR\_CIWP\_3.12 | The history of edits and changes for a specific course will also display which user submitted a change request and which user(s) approved the request, or just which user edited the course information. |
| FR\_CIWP\_3.2 | The user should be able to see how many courses are in the system. |
| FR\_CIWP\_4.0 | The user should be able to edit via web form and/or submit a change request for another courses info. |
| FR\_CIWP\_4.1 | When editing course information, the user should be able to see the original on the left, the version being edited on the right, and changes between the two highlighted. |
| FR\_CIWP\_5.0 | A level 3 user should be able to see a change log with the most recent course information additions or changes. |
| FR\_CIWP\_5.1 | The list of recent changes will also display which user submitted the add/change request and which user(s) approved the change request, or just which user added/edited the course information. |
| FR\_CIWP\_6.0 | The user should be able to download a document containing the course information manipulated into an organized form. |
| FR\_CIWP\_7.0 | The data fields to be parsed from uploaded documents and that will be present on the web form for adding, editing, and searching will be as follows:   * Institution * Campus * Academic Career * Academic Groups * Admin Unit * Academic Organizations * Effective Quarter * Subject * Course Prefix * Catalog Number * Course Name * Course Description * Grading Scheme * Components * Component Hours * Faculty Workload * Units and FA Progress Units * Course Challenge * Repeatable for Credit * CIP * Attributes * Enrollment Requirements * Course Topic * Course Fee * Student Learning Objectives * Core Learning Objectives * Program Specific Outcomes |

**Course Information Database**

|  |  |
| --- | --- |
| **Requirement ID.** | **Description** |
| FR\_CID\_1.0 | The database shall store course information that is passed to it from the parser or sent from the web form from the web page. |
| FR\_CID\_2.0 | The course information in the database can be updated by approved add or change requests to the database sent from the web page. |
| FR\_CID\_3.0 | The database shall have an archive of previous versions for each course. |

**Course Information Parser**

|  |  |
| --- | --- |
| **Requirement ID.** | **Description** |
| FR\_CIP\_1.0 | The parser shall parse course information documents that are uploaded to the web page. |
| FR\_CIP\_2.0 | The parser will pass relevant course information parsed from the document back to the web page for review. |

## Non-Functional Requirements

The non-functional aspects of the system concern security, usability, correctness and interoperability.

**Security**

|  |  |
| --- | --- |
| **Requirement ID.** | **Description** |
| NFR\_SEC\_1.0 | The application shall have different levels of users to ensure information integrity. |
| NFR\_SEC\_1.1 | Level 1 Users (employees, faculty) will be able to submit an add request to add a new course, and make change requests for existing courses. |
| NFR\_SEC\_1.2 | Level 2 Users (Dean/Division) will be able to review and approve add and change requests for final approval by a Level 3 User. |
| NFR\_SEC\_1.3 | Level 3 Users (Instruction Office) will make the final add and change approvals to have course information added to the database. Level 3 users can also directly add/change course information. |

**Usability**

|  |  |
| --- | --- |
| **Requirement ID.** | **Description** |
| NFR\_USAB\_1.0 | The application shall have labels around input fields. |
| NFR\_USAB\_2.0 | The application shall have images with alternate text. |
| NFR\_USAB\_3.0 | The application shall have the functions of buttons and other interactive icons clearly marked. |
| NFR\_USAB\_4.0 | The application shall produce clean output. |

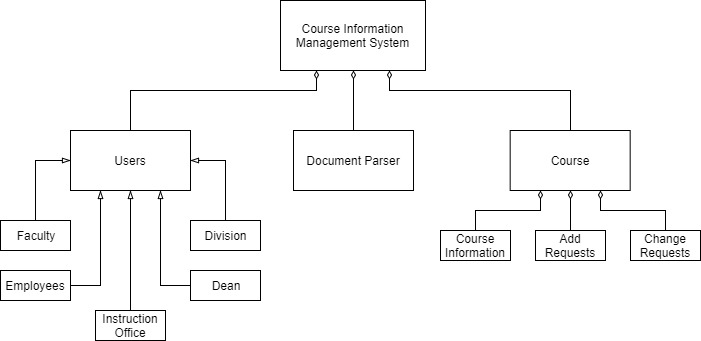
**Correctness**

|  |  |
| --- | --- |
| **Requirement ID.** | **Description** |
| NFR\_COR\_1.0 | The application shall store and maintain complete and correct course information. |
| NFR\_COR\_2.0 | The application shall ensure that information return from searching is complete and correct, and answers the user’s query. |

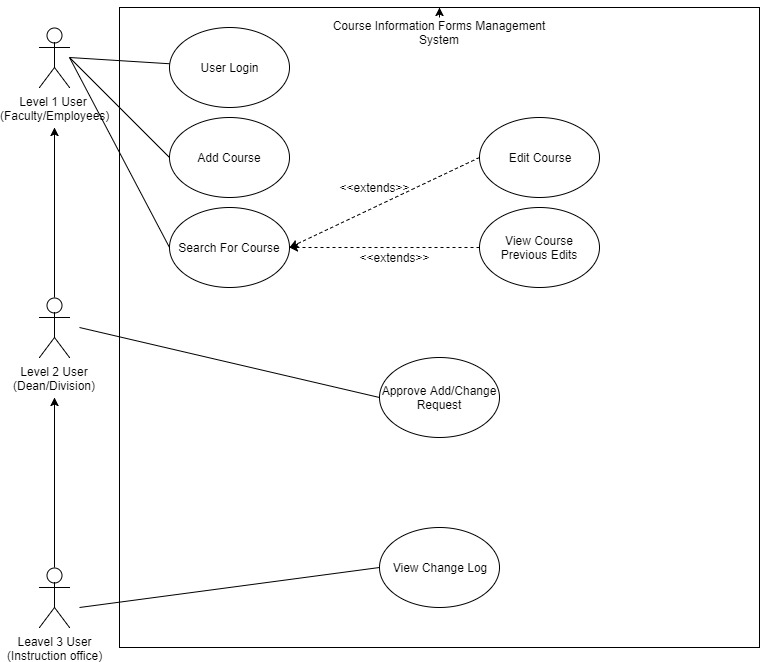
**Interoperability**

|  |  |
| --- | --- |
| **Requirement ID.** | **Description** |
| NFR\_COR\_1.0 | The application will extensible and integrate with the Virtual Student Advisor. |
| NFR\_COR\_2.0 | The application will be extensible and leave open the possibility of integrating with other systems used by other organizations. |

# Domain Model

****

# Use Case Diagram

****

# Use Case Descriptions

**Name: User Login**

Scenario:

1. The web page displays a web form for a user name and password.
2. The user types in their user name and password.
3. The user is logged in to the application. The application displays various tabs depending on the user’s level of access. From these tabs, the user can then access the Add Course, Search for Course, Approve/Reject Add/Change Request, and View Change Log user cases by clicking the associated tab.

Alternates:

1. If the user name or password is invalid after completing step 2, an error message will display, notifying the user that either the user name or password as invalid. The user will have to repeat step 2.
2. If the user is a Level 1 user, they will only be able to see the tabs for entering the Add Course, and Search for Course (which can leads to Edit Course) use cases.
3. Along with the tabs available for a Level 1 user, a Level 2 user can see tabs for entering the View Course Previous Edits (from Search for Course tab) and Approve Add/Change Request use cases
4. Along with the tabs available for a Level 2 user, a Level 3 user see tabs for entering the View Change Log use case.

**Name: Add Course**

Scenario:

1. The user clicks on the “Add Course” tab.
2. In the “Add Course” tab, the web page will display an uploader for a course information document and a web form.
3. The user fills out the data fields in the web form with relevant course information, OR
4. The user uploads a word document containing course information.
5. After the file is uploaded and parsed, the web form will be populated with the data parsed from the document.
6. After reviewing the information in the web form and ensuring that it is correct, the user clicks the “Add Course” button at the bottom of the screen.
7. A confirmation message will display, prompting the user to confirm the addition or cancel.
8. If the user confirms, a message will display showing the confirmation is successful, OR
9. If the user cancels, the user can continue reviewing and editing the course information until they decide to confirm the course information and complete step 8.

Alternates:

1. After clicking “Add Course”, an add request is sent to the system if the user is a Level 1 or 2 user. If the user is a Level 3 user, the course is added.
2. If the file is not a valid course information document, the page will display an error after step 4. The user will be returned to step 2 to try a new upload path.
3. If the user attempts to complete step 6 without filling out all the web form data fields, an error message will pop-up preventing step 6 completion, letting the user know that not all fields are filled out.
4. The user can click any of the other tabs to enter another use case at any time. If the user attempts to leave the “Add Course” tab after making filling out any number of data fields in the web form, a notification will pop-up notifying them that all changes will be lost. The user will either confirm or stay on the editing screen.
5. After clicking “Add Course” in step 6, the system will check if there is an existing course with the same Course ID/Name. If a course in the system exists with the same Course ID/name, the system will prompt the user to enter the “Edit Course” use case.

**Name: Search For Course**

Scenario:

1. The user clicks on the “Search” tab.
2. The web page will display a web form with relevant course information data fields.
3. The user fills out one or more of the data fields.
4. The user clicks the “Search” button.
5. The application returns a list of courses with values in the data fields that match what the user has filled out.
6. The user clicks on a course.
7. The course’s information is displayed.
8. The user clicks the “Edit” button to enter the Edit Course use case or clicks the “View Course Edits” to enter the View Course Previous Edits use case.

Alternates:

1. The user can click the “Back” button after step 7 to return to the search screen. The last searched results will still be displayed until the user enters new search parameters.
2. If there are no courses with data fields that match what the user has entered, an error message will display, notifying the user when they click the “Search” button in step 4 and preventing progress.
3. If the user inputs an invalid value for a data field (such as a string where only an integer can be), an error will pop up after either step 4, notifying the user and preventing progress.
4. The user can click any of the other tabs to enter another use case at any time.

**Name: Edit Course**

Scenario:

1. The user clicks the “Edit” button while on a course information page.
2. The web page displays the current original version of the course information on the left and a web form for editing on the right. Any differences between the two will be highlighted on the left-side original.
3. The user edits one or more of the data fields in the right-side editing version.
4. The user clicks the “Save Changes” button.
5. A confirmation message will display, prompting the user to confirm the changes or cancel.
6. If the user confirms, a message will display showing the confirmation is successful, and return the user to the course’s information screen, OR
7. If the user cancels, the user can continue reviewing and editing the course information until they decide to confirm the course information and complete step 5.

Alternates:

1. After clicking “Save Changes”, a change request is sent to the system if the user is a Level 1 or 2 user. If the user is a Level 3 user, the course is changed.
2. The user can close the application or click the “Back” button at any time to return to the course’s information page. If the user attempts to close the application or go back after making some changes, a notification will pop-up notifying them that all changes will be lost. The user will either confirm or stay on the editing screen.
3. The user can click any of the other tabs to enter another use case at any time. If the user attempts to leave the “Edit Course” page after making some changes, a notification will pop-up notifying them that all changes will be lost. The user will either confirm or stay on the editing screen.
4. If the user leaves any field blank or inputs an invalid value for a data field (such as a string where only an integer can be), an error will pop up after step 3, letting the user know there are errors and preventing them from progressing past step 3 until the errors are fixed.

**Name: View Course Previous Edits**

Scenario:

1. The user clicks the “View Course Edits” button while on a course information page.
2. The web page displays a list of timestamps of previous course edits, and who made/approved them.
3. The user clicks on any of the previous course versions.
4. The information from the old course version will be displayed, along with a timestamp and who made/approved them.

Alternates:

1. The user can click the “Back” button at any time to return to the course’s information page.
2. The user can click any of the other tabs to enter another use case at any time.

**Name: Approve/Reject Add/Change Request**

Scenario:

1. The user clicks on the “Add/Change Requests” tab.
2. A list of Add and Change requests are shown, with Add requests in a column on the left, and Changes request in a column on the right.
3. The user clicks on a change request, and the current version is shown on the left, and the new information that is being requested to be changed into will be shown on the right, OR
4. The user clicks on an add request, and the course information to be added is displayed on the screen.
5. The user clicks the “Approve” or “Reject” button to approve or reject an add or change request.
6. If the user clicks the “Reject” button, a confirmation message will display, and if confirmed, the request is removed from the request list, OR
7. If the user clicks the “Approve” button, a confirmation message will display. The confirmation message will also have a date field the user can edit to choose the time for when the course information will go live.
8. When the date field is edited by a Level 1 or 2 user, the date field will appear as a suggestion for when the info should go live to the next level user. When the date field is edited by a Level 3 user, that is when the info will go live.
9. If the user confirms, the new course information will be added to the database if the user is a Level 3 user (and go live on the input date), OR the Add/Change request will be forwarded to a higher level user if the user is a Level 1 or 2 user, OR
10. If the user cancels, they can continue to review the Add/Change request list.

Alternates:

1. The user can click the “Back” button while reviewing an add or change request to return to the list of add and change requests.
2. The user can click any of the other tabs to enter another use case at any time.

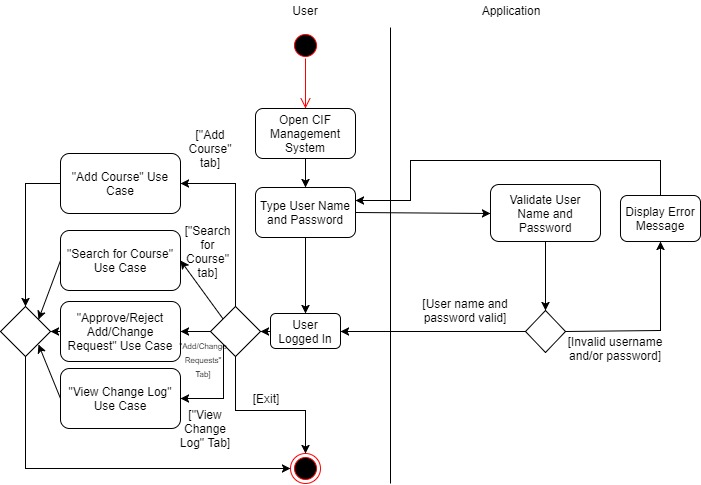
**Name: View Change Log**

Scenario:

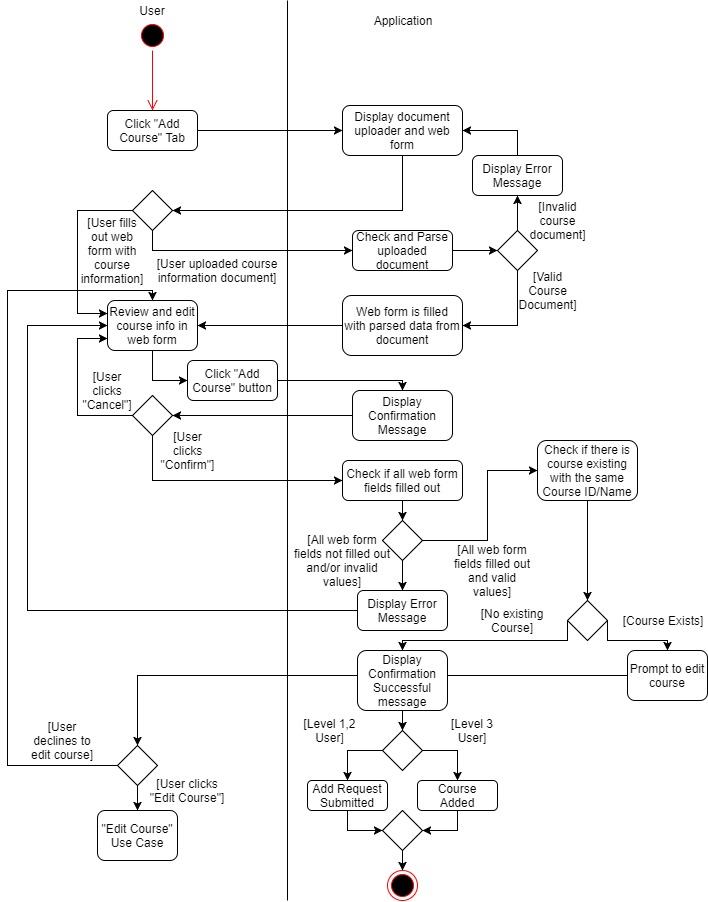
1. The user clicks on the “Change Log” tab.
2. A list of recent approvals or additions/edits, their timestamps, and the users involved are shown.
3. The user clicks on one of the recent additions/edits.
4. If the item clicked is an edit, the old version will be displayed on the left while the new version will be displayed on the right, OR
5. If the item is an addition, the new course information will be displayed.

# Activity Diagrams

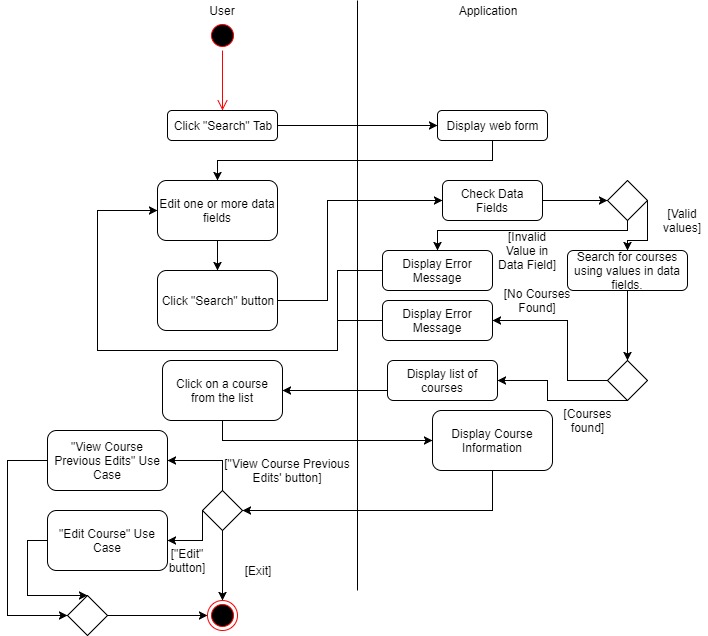
User Login



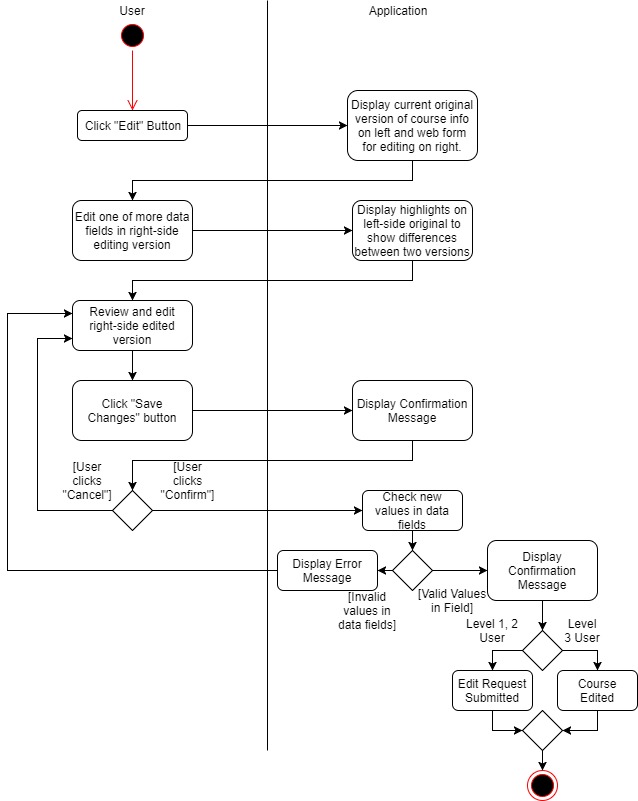
Add Course



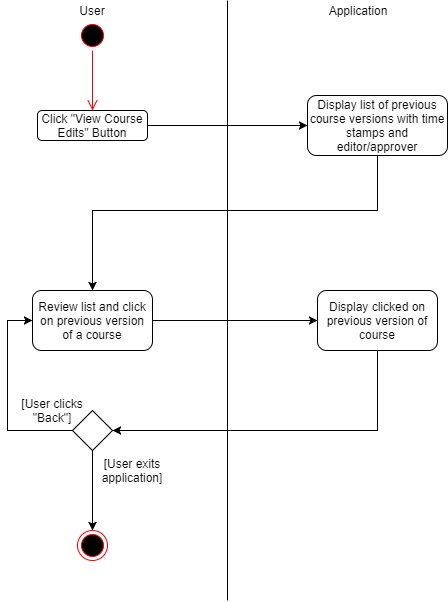
Search For Course



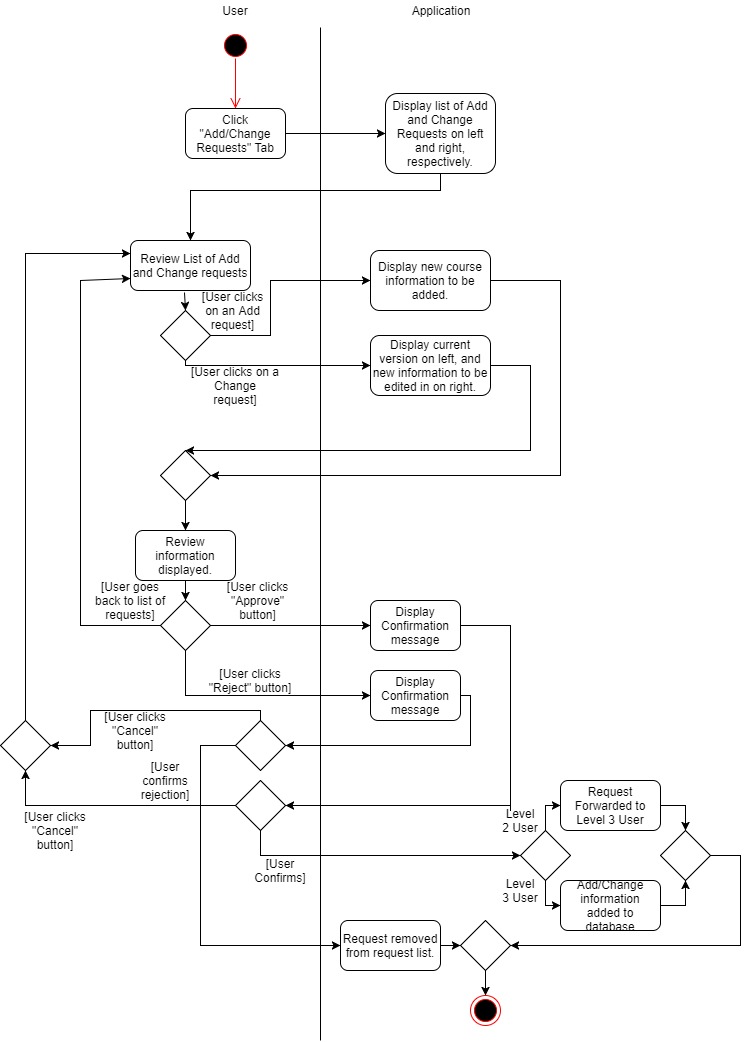
Edit Course



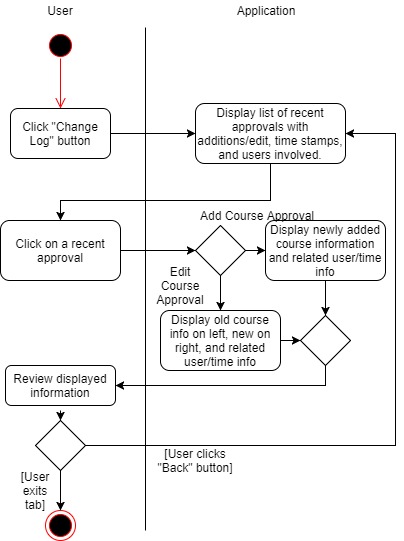
View Course Previous Edits



Approve/Reject Add/Change Request

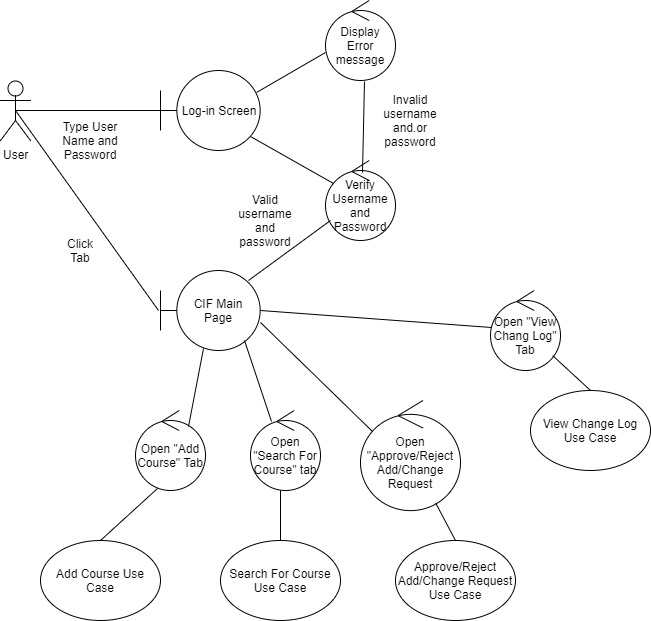


View Change Log

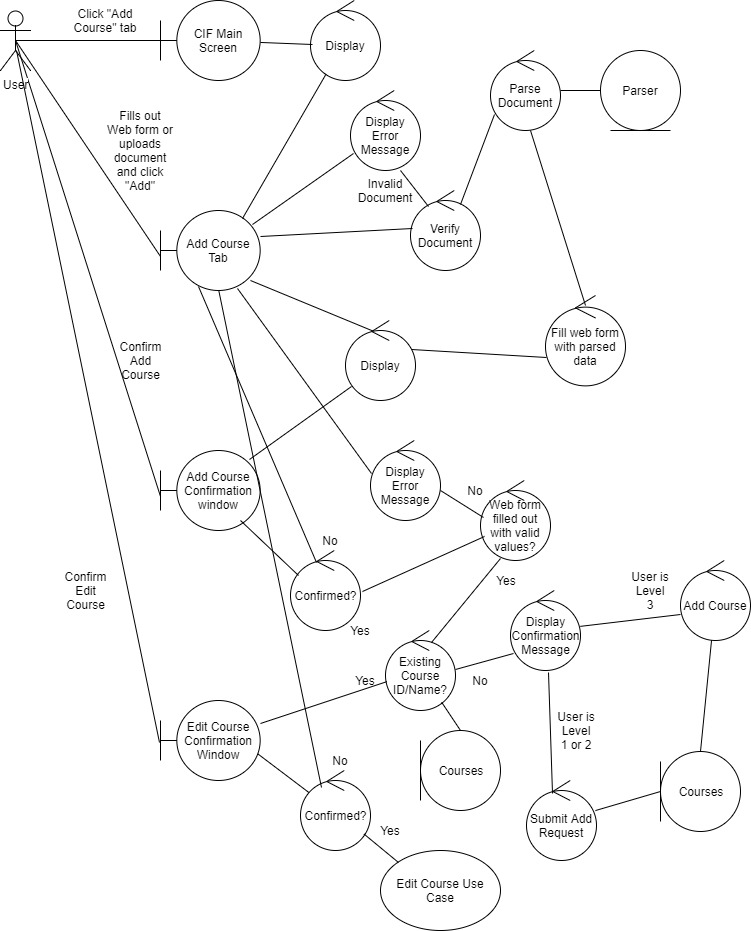


# Robustness Diagrams

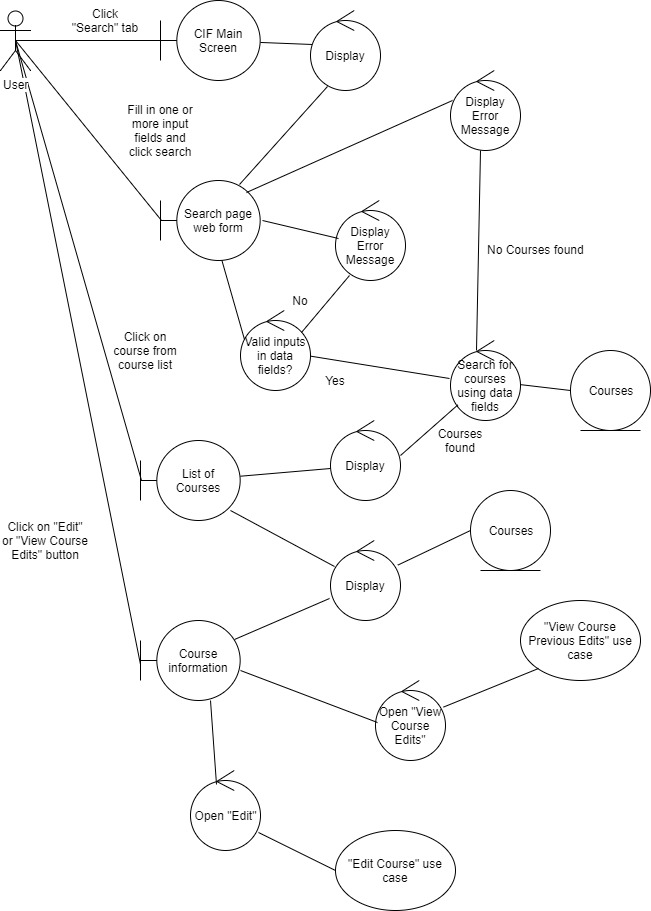
User Login



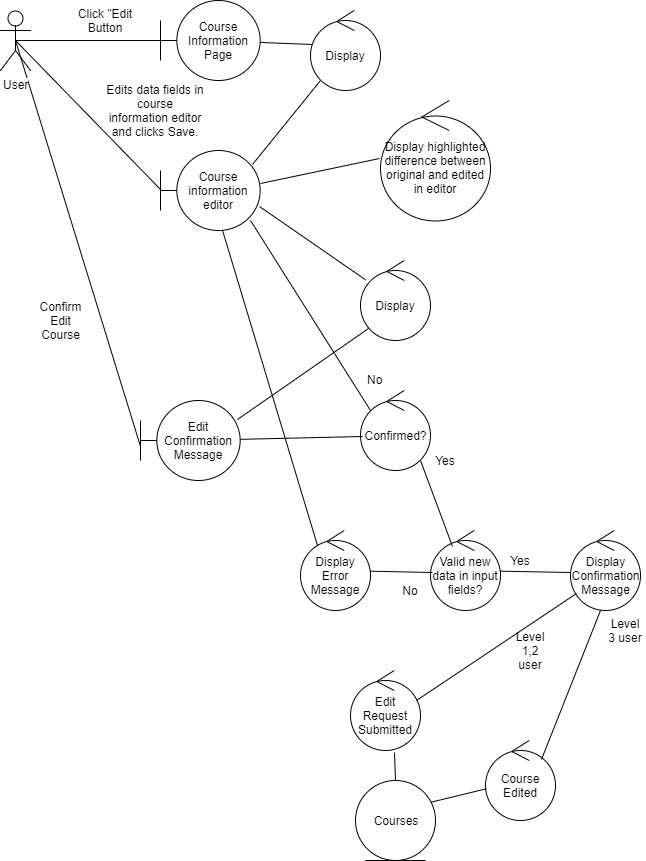
Add Course



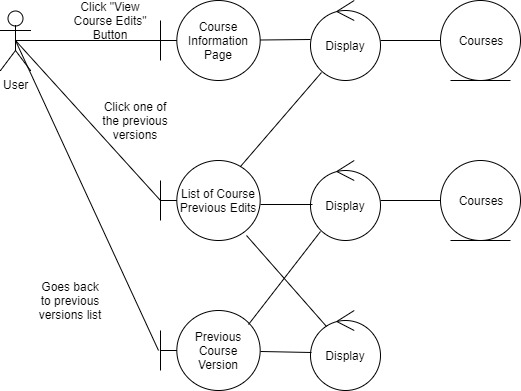
Search For Course



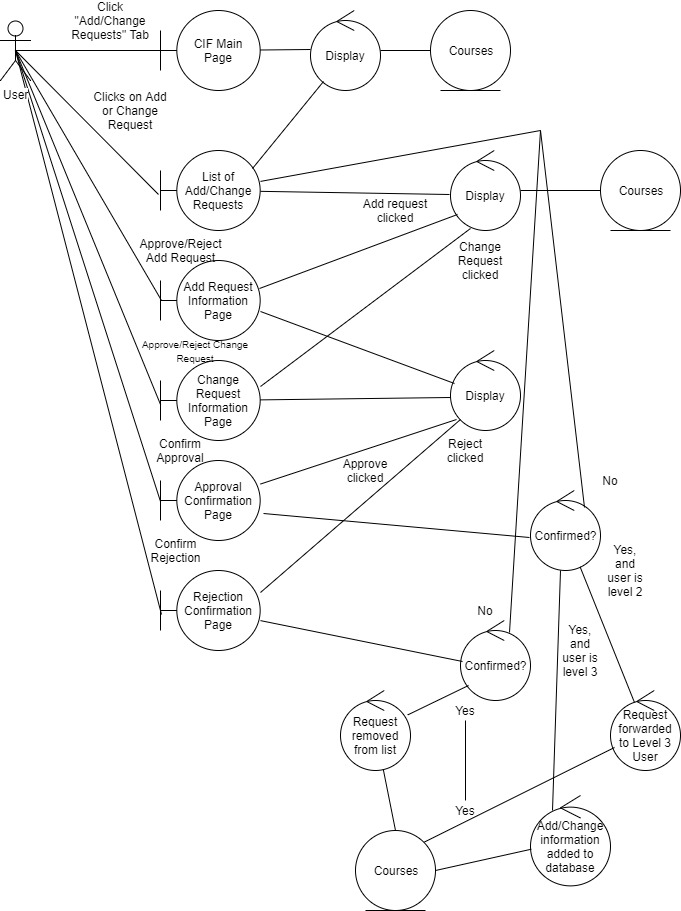
Edit Course



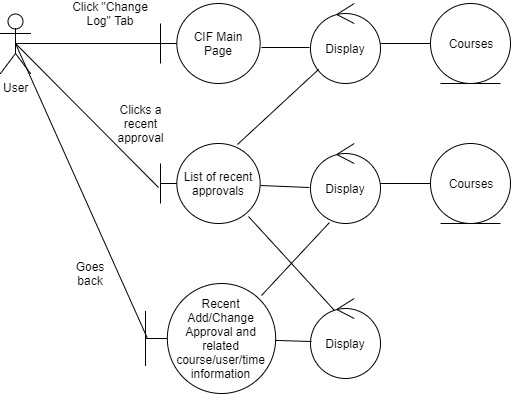
View Course Previous Edits



Approve/Reject Add/Change Request

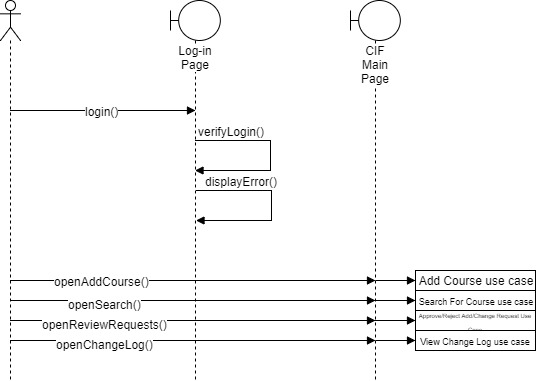


View Change Log

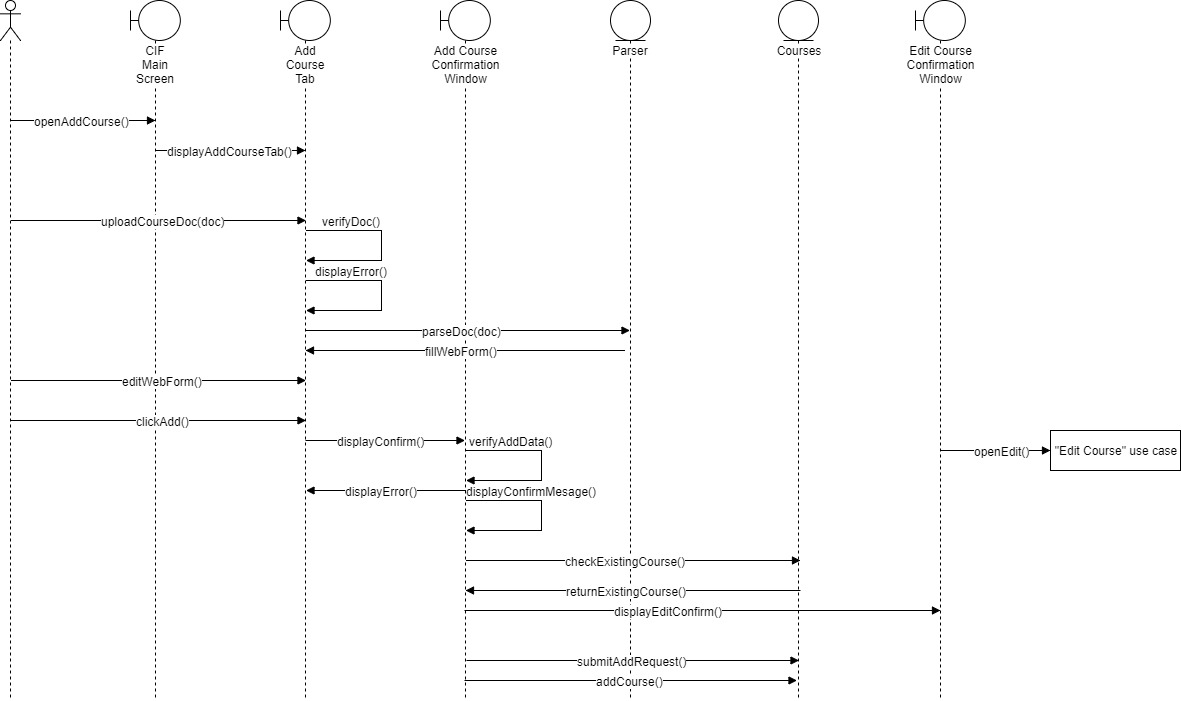


# Sequence Diagrams

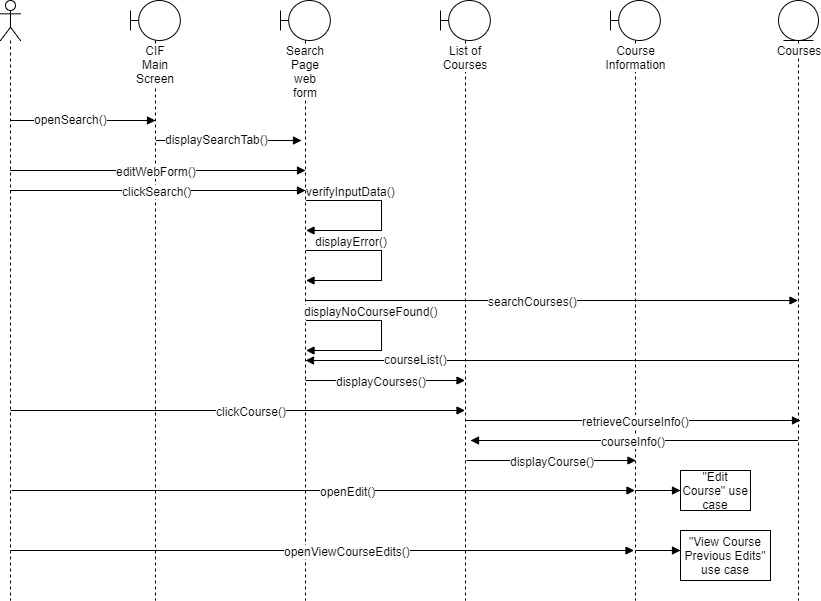
User Login



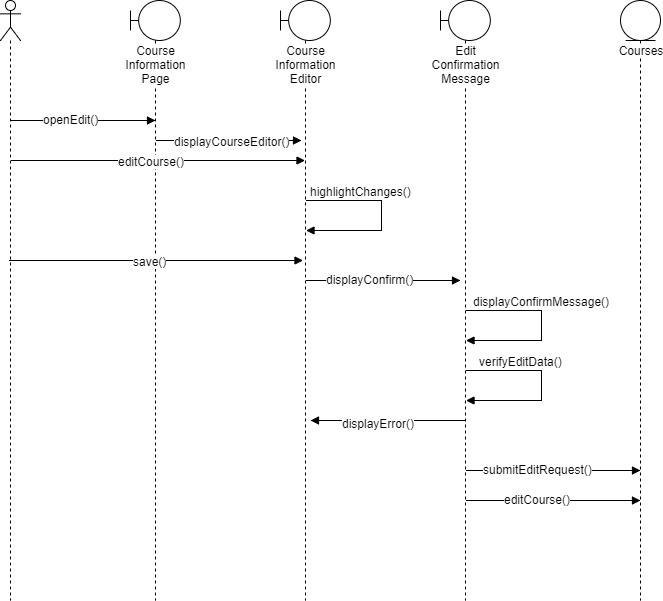
Add Course



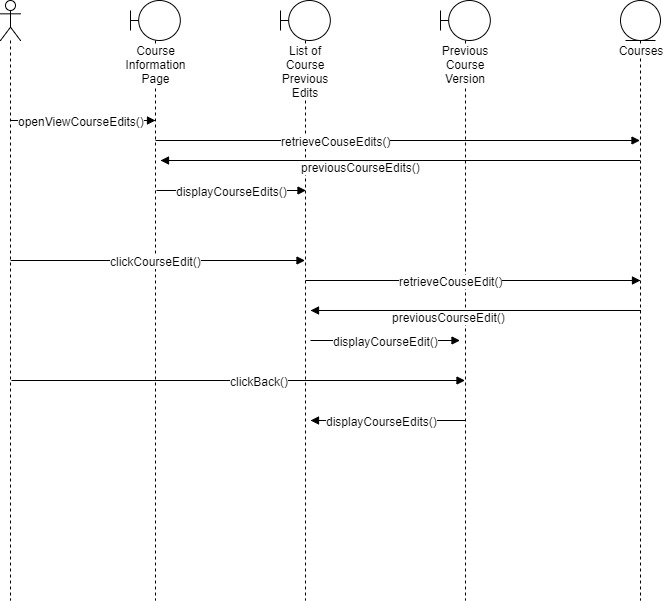
Search For Course



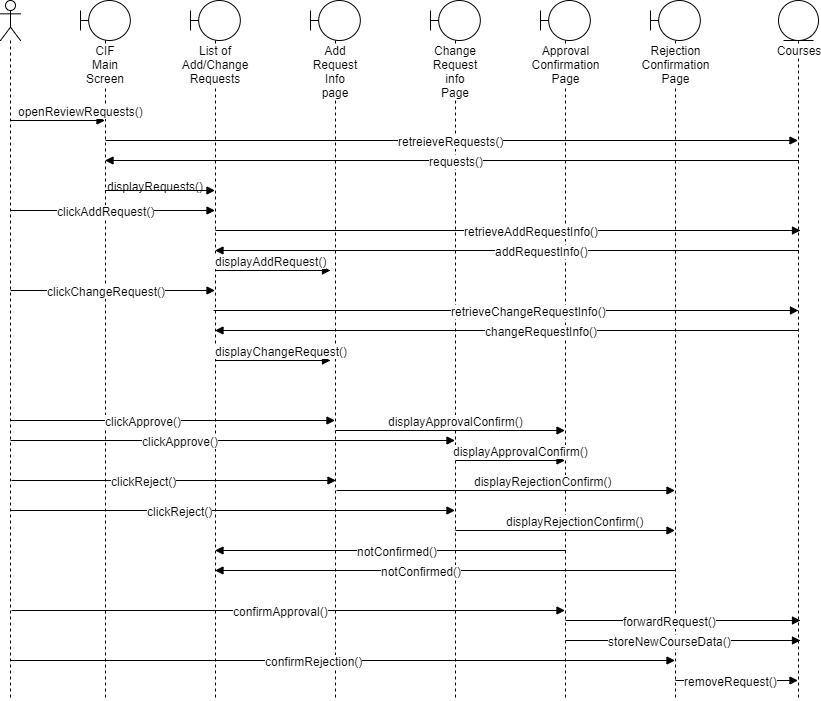
Edit Course



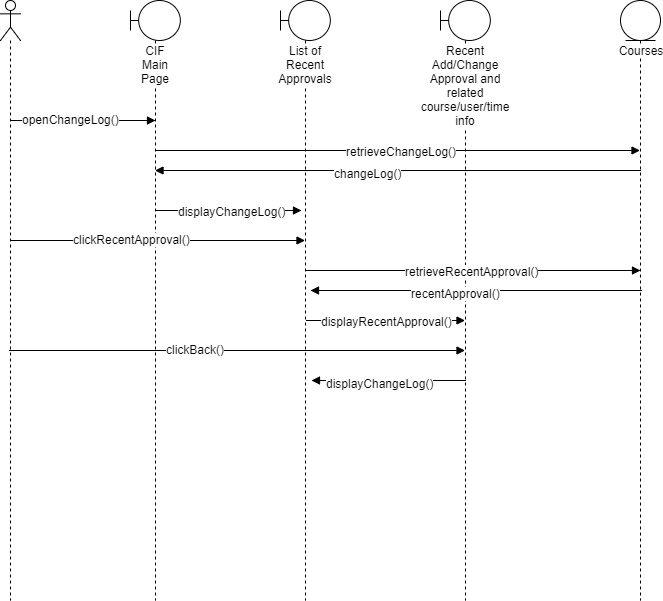
View Previous Course Edits



Approve/Reject Add/Change Request



View Change Log



# Set-Up

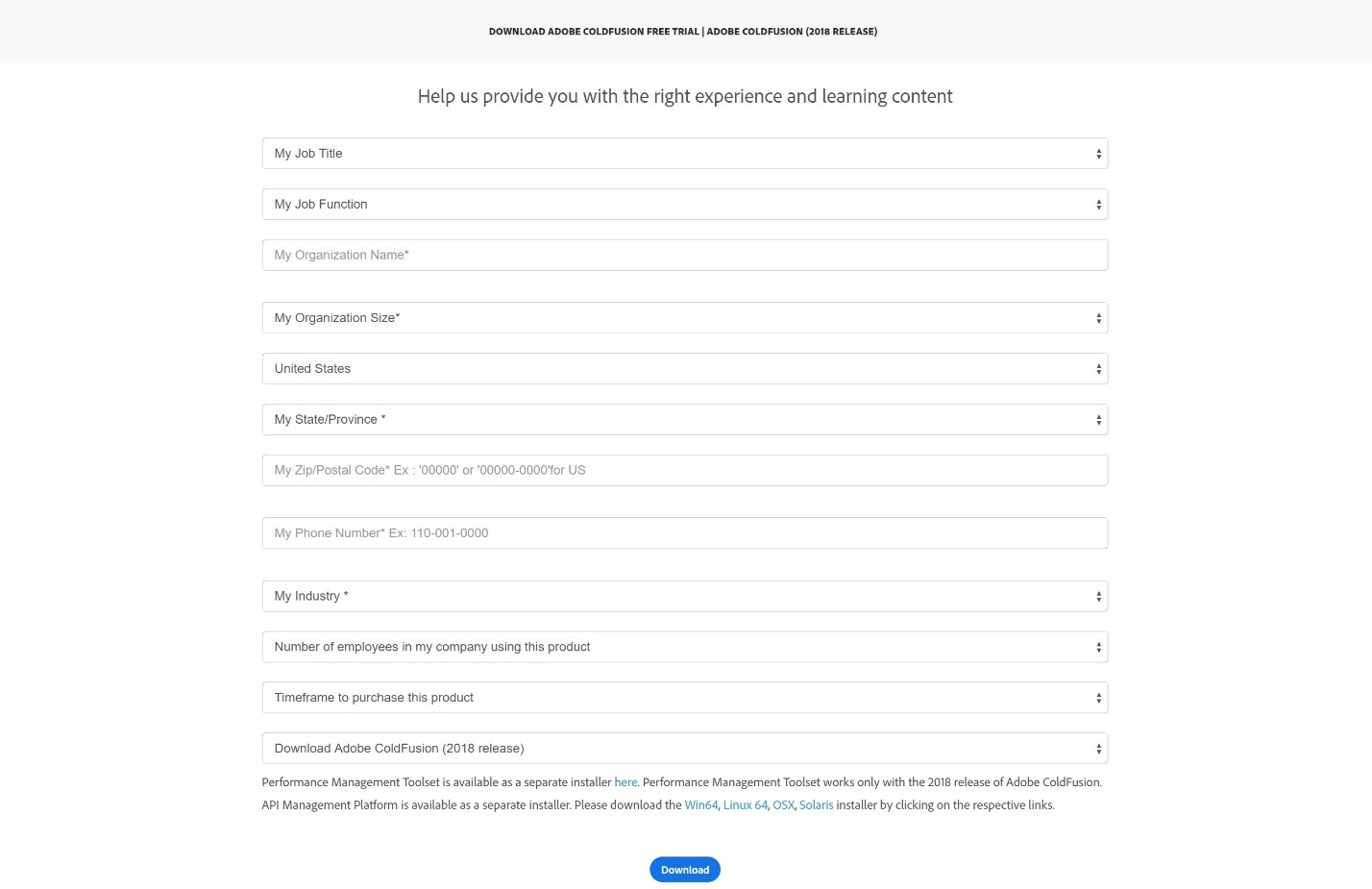
## Tools

Course Information Forms Management System requires ColdFusion to be used while developing the website portion because many of the systems at Everett CC run using ColdFusion. SQL Server will also be used to develop the database. For developing the website, Eclipse for Web Development will be used with the ColdFusion plugin. ColdFusion will also be installed so the Administrator Page can be accessed to set up data sources. The database will be hosted on Amazon Relational Database Servers, and SQL Server Management Studio will be used to connect to the database instance and manage it.

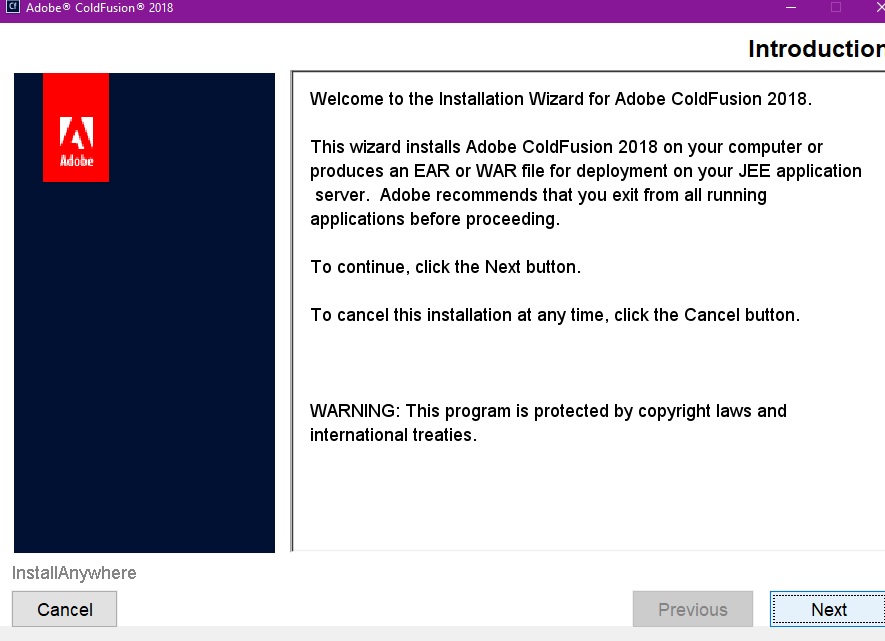
## Setting Up ColdFusion

First, ColdFusion needs to be installed on the computer:

1. Visit <https://www.adobe.com/devnet/coldfusion/downloads.html>. For the purpose of this project, click on the link “Try” to get a free version of ColdFusion.
2. Fill out the form on the next page and click download to get your free version of ColdFusion.



1. Once downloaded, open the downloaded file. The file name should look like “ColdFusion\_2018\_WWEJ\_win64” (for 64 bit Windows, may vary based on operating system). You should see the following window:

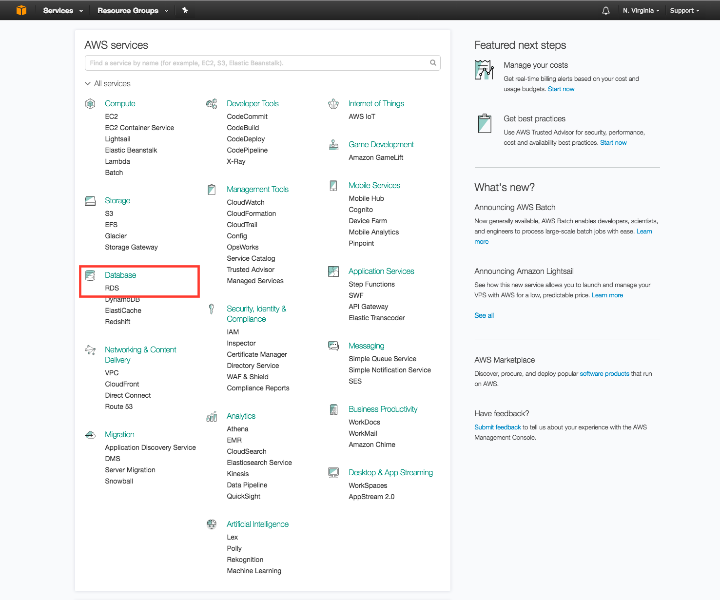


1. Click Next.
2. Confirm that you’ve read and agree to the License Agreement.
3. On the next page, the window will ask for a Serial Number. Instead, check the box marked “Developer Edition” and click Next.
4. On the next page, for the purposes of this project, pick the Server Configuration and click Next.
5. On the next page, for the purposes of this project, pick Development Profile, and click Next.
6. On the next page, check all the sub-components to install.
7. On the next page, you will be asked to create a username and password for remote start/stop. Create a username and password and save them somewhere in case you forget. Click Next.
8. On the next page, you will be asked to Access Add-on Services Remotely. For the purposes of this project, you will not need to check the box. Click Next.
9. On the next page, pick a directory to install ColdFusion. It will be important to remember where you install ColdFusion because the website code files need to be placed in a folder in that directory. Click Next.
10. On the next page, pick a port number for ColdFusion. Click Next.
11. On the next page, you can pick a hostname/DNS for the ColdFusion Server. Click Next after picking one if you chose to.
12. On the next page, you will be asked to create a password for the CF Administrator Panel. Create a password and save it somewhere in case you forget. This password will be used to access the administrator panel to later set up the data source.
13. On the next page, you will be asked to set-up RDS. Enable RDS and set-up a password. Click Next.
14. On the next page, you will be asked to automatically check for server updates. Check the box. Click Next.
15. On this final page, you will be prompted to Install. Click the Install button.

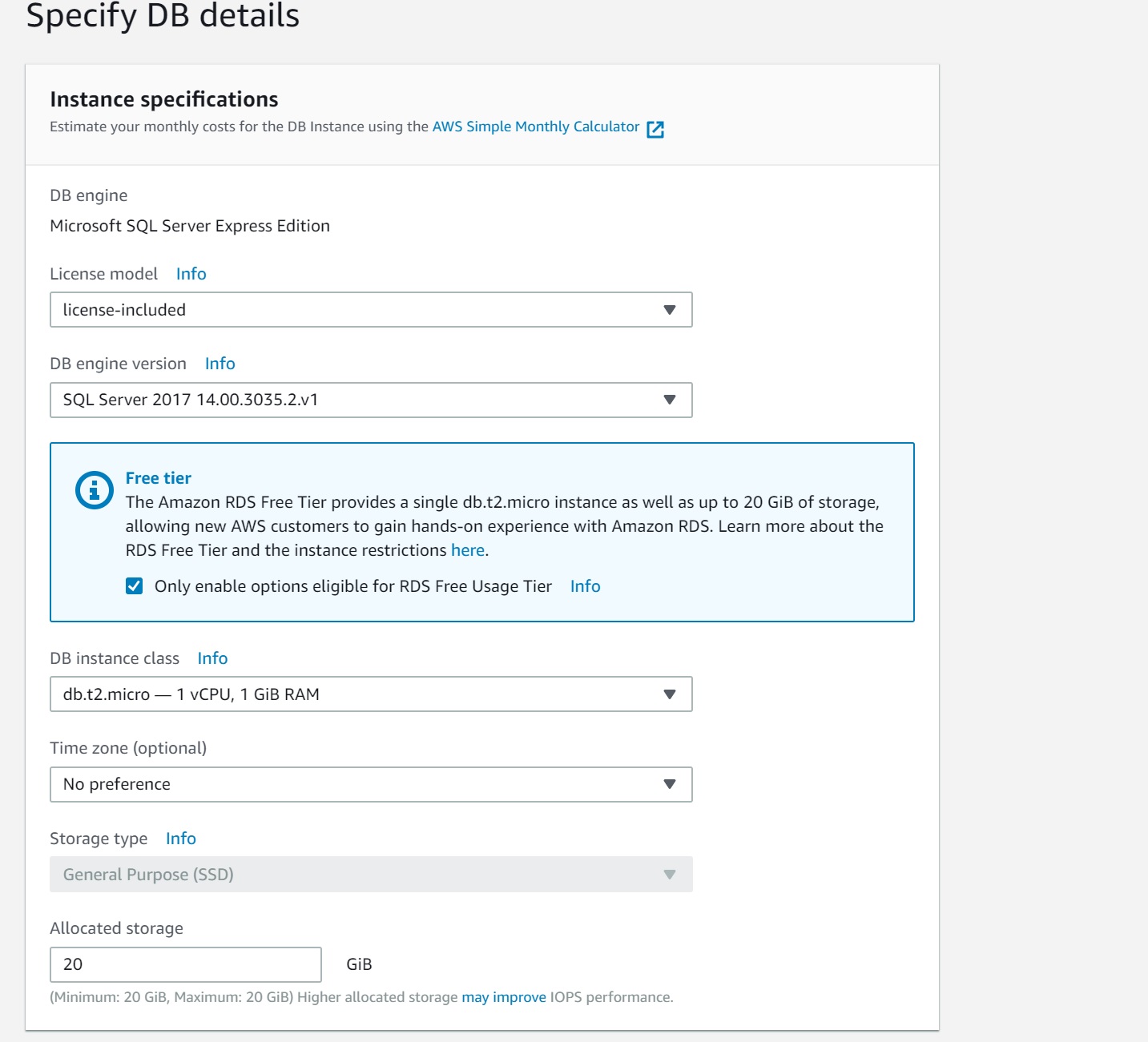
## Setting Up a Database Instance on Amazon Relational Database Service

The database instance for this project will be hosted on Amazon RDS. This will allow you to easily access the database from multiple computers and networks after setting up the correct permissions.

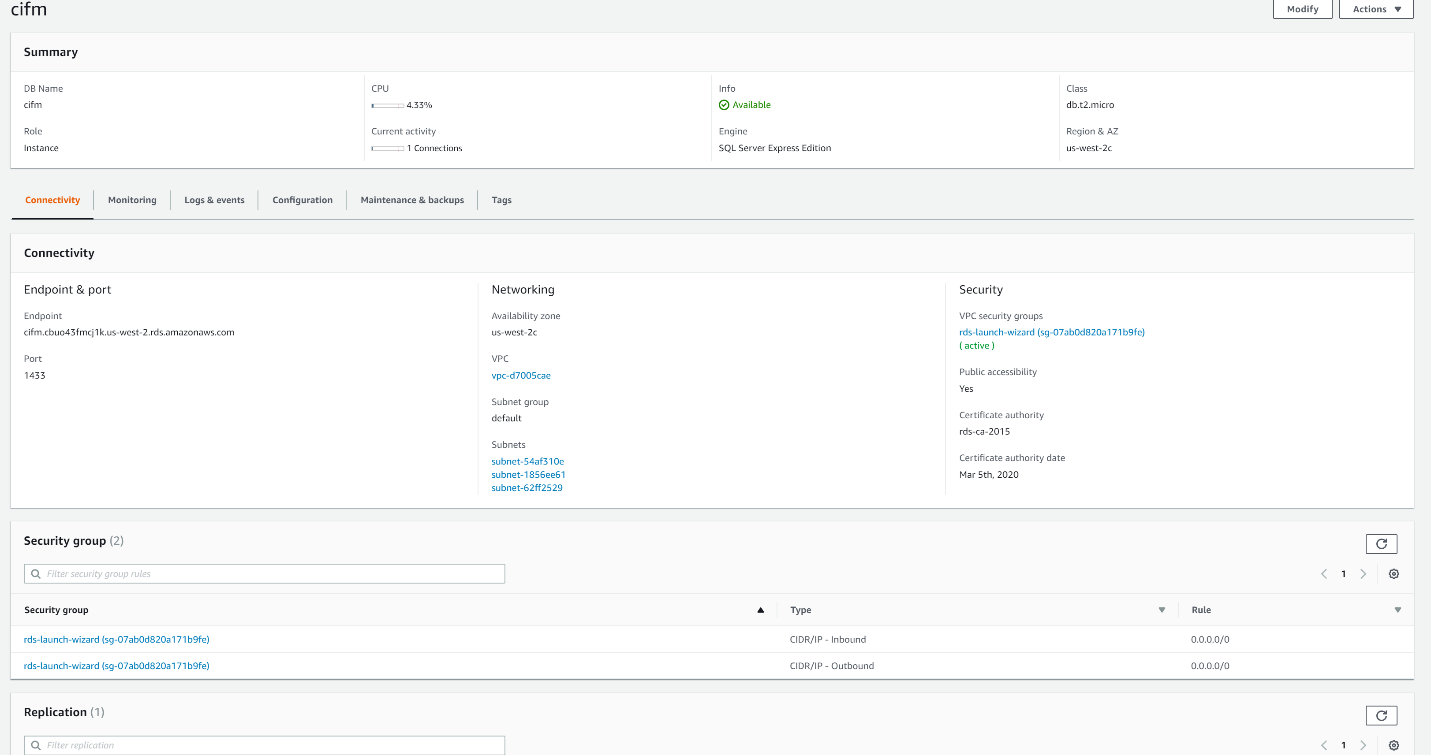
1. Navigate to <https://aws.amazon.com/rds/>
2. Click on My Account on the top right and click AWS Management Console.
3. Sign-in, or create an account.
4. On the management console, click RDS under the Database section.



1. Under Resources, click DB Instances, or click Create Database if the prompt appears on the screen.
2. You will be asked to select an engine for the database. For this project, SQL Server will be used. Click SQL Server and pick the Express Edition.
3. On the instance specifications, in a blue box, you will be asked if you want to enable options eligible for RDS Free Usage Tier. For this project, check this box first. All the settings in Instance Specifications should be set to the proper defaults:



1. Under settings at the bottom of the page, pick a DB Instance identifier and a username plus password. Make sure to save the username and password somewhere, as they will be used to access your database instance from the ColdFusion administrator panel and SQL Server Management Studio.
2. For Network and Security, leave the default settings for VPC and subnet group. It will be important to access the settings for these later. Allow Public Accessibility and Create new VPC security group.
3. Leave the Port number as 1433, and the default DB parameter group and option group.
4. The settings for Encryption, Backup, Monitoring, Maintenance, and Deletion Protection can be tailored to specific preferences.
5. Click Create Database.
6. The database will take a while to set-up. After it is set-up, view your database instances and click on the database you just created. You should see a screen like this. Make not of the Endpoint and Port because they will be used to set up database connectivity later:



1. Click on the VPC security group that you have under Security on the right.
2. A list of your security groups will be displayed. Right-click on the one related to your DB Instance (if you have only one DB instance created, there should just be one group here).
3. Click on Edit Inbound Rules. Modify the type to All Traffic. Do the same thing for Edit Outbound Rules.

## Accessing the ColdFusion Administrator Panel

1. Navigate to <http://localhost:8500/CFIDE/administrator/index.cfm>. The number 8500 will correspond to the port number you entered during ColdFusion installation. The default is 8500.
2. On the next page, click Data and Services.
3. A page will show up showing your data sources. Here, you will add your new DB Instance as a data source.
4. Type in a new data source name under Add New Data Source.
5. Pick a drive corresponding to your DB Instance; In this case, it should be Microsoft SQL Server. Click Add.
6. In the next page, input the details of your database. For Server, copy and paste your DB Instance Endpoint. Port, User name and password should also correspond to your instance.
7. Add the data source. If successful, you should return to the data sources page. Your DB Instance should be listed at the bottom of the list with OK listed as it status.

## Setting Up SQL Server Management Studio

1. Navigate to this site: <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017>. Download SQL Server and Install from the downloaded file. The installation itself is automated.
2. When finished, open SSMS.
3. SSMS will open and a window will open asking you to connect to server. If this doesn’t appear, you can also click File, and click Connect Object Explorer.
4. In the window, for Server Type pick Database Engine. Enter the DB instance endpoint as the server. For Authentication, click SQL Server Authentication. Enter the username and password for the Db instance. Click connect, and you will be connected to the database.
5. On the left side of the screen in the Object Explorer, expand the Databases folder. Right-click the Databases folder, and click New Database.
6. Enter the database name. For this project, the name will be CourseInfoForms. No other settings need to be modified. Click Ok.
7. Right-click your new Database, and click New Query.
8. In the query window, the queries for creating the database and inserting the test data can be inserted here. Copy and paste the queries from createcourses and testcourseinsert, one at a time, and click Execute individually for each one.
9. If the queries do not work, make sure to right click the database and Refresh. Also, with the query window open, go the Edit, Intellisense, and refresh local cache. Finally, make sure you are not in the master branch and instead in the CourseInfoForms branch (the drop down list under View).

## Setting Up CFEclipse

Before setting up CFEclipse, Eclipse must be installed. You can install Eclipse from <https://www.eclipse.org/downloads/packages/>. Download and Install Eclipse IDE for JavaScript and Web Developers.

1. After installing, open Eclipse. You will need to set up which directory to use as the workspace. For this project, it is imperative to set-up the workspace in your ColdFusion installation folder.
2. Navigate to your ColdFusion folder, open cfusion, then open wwwroot. Here, create the folder you will use to store the code for the website, and put the .cfm files required for the project into this folder.
3. Go back to the directory setup for Eclipse. The workspace path should look like: ColdFusion2018\cfusion\wwwroot. The drive letter of the file path will be dependent on your installation location. You can also browse to wwwroot, but make sure the file path ends with wwwroot and not the folder name that will store your code.
4. Next, you need to install MXUnit and CFEclipse.
5. Select Help > Install new software... menu option.
6. Type http://cfeclipse.org/mxunit/update-dev into the Work with box and press Add....
7. Type MXUnit into the Name field and press OK.
8. Wait while Eclipse says Pending in the main part of the dialogue.
9. Tick MXUnit and press Next.
10. Press Finish
11. Wait while Eclipse downloads and installs MXUnit. You should see a progress indicator in the bottom of the window.
12. After installing Eclipse will prompt you to restart Eclipse, which you should do.
13. After restarting Eclipse, select Help > Install new software... menu option again.
14. Type http://www.cfeclipse.org/update into the Work with box and press Add....
15. Type CFEclipse into the Name field and press OK.
16. Wait while Eclipse says Pending in the main part of the dialogue.
17. Tick CFEclipse (including CFUnit and Frameworks). **Ensure CFEclipse DocShare is NOT selected** and press Next.
18. Press Finish
19. Wait while Eclipse downloads and installs CFEclipse. You should see a progress indicator in the bottom of the window.
20. After installing Eclipse will prompt you to restart Eclipse, which you should do.

## Integration

After setting up all the above, the necessary files will need to be put in the correct places to ensure the project can be built correctly. A couple of the steps were covered in previous steps, but they will be covered again in general.

1. Navigate to ColdFusion2018\cfusion\wwwroot (drive letter will be what you set it to).
2. Open Eclipse and set the workspace to the file path in the previous step.
3. Right-click in the empty area under the Navigator and click New -> Project.
4. In the Wizard, click CFEclipse, and click CFML Project underneath it. Click Next.
5. Name the project CIFM\_Website. Click Finish.
6. Now, navigate to ColdFusion2018\cfusion\wwwroot and find the project folder, which should be named CIFM\_Website.
7. Copy and paste all the .cfm, .cfc, and .css files from luoj\_cifmfinal into this folder.
8. Go back to Eclipse and refresh to ensure all the files are shown in the project.
9. If the previous steps for setting up the other technologies has been performed correctly, navigate to <http://localhost:8500/CIFM_WEBSITE/>. The application’s front page should be displayed.

# Conclusion

## Challenges

Developing the Course Information Forms Management System was challenging because it was my first time working with most of the technologies required. ColdFusion was new territory, and it was especially daunting because there are not many ColdFusion tutorials and knowledge of it is limited. I learned most of my ColdFusion through trial and error. SQL Server and database management was also new to me and proved to be challenging to learn because certain settings, such as permissions, must be set-up correctly or constant errors will be encountered in database connectivity. Finally, it was also challenging to manage my schedule to ensure I remain in communication with the sponsor to continuously verify that requirements are met. Due to the challenge of learning these new technologies, it was important to not underestimate the workload required in learning and applying this new knowledge correctly

## Achievements

I’ve managed to build a prototype for CIFM that fulfills most of the basic requirements for this application. Courses can be added, searched for, and edited. I standardized the add and edit pages to make the process of adding and editing courses more familiar. I created a search page that allows for searching through a few important key terms of a course, such as the effective quarter or subject. Displaying the course information will also remain standard across different courses. Finally, I’ve allowed for some level of version control. Searching for a course will show only the recent version, but the older versions of a course are stored and can be viewed as well.

## Results

During the development of this application, I learned valuable concepts and lessons. Besides achieving a basic prototype, I also learned a lot about front-end development. Front-end development was unexplored as most of UW Bothell’s courses are focused on back-end development. I also learned how to design a system to fit with pre-existing software and tight requirements. I managed to expand my knowledge base and reinforce some existing knowledge, such as knowledge of the Software Development Life Cycle because my task was full-stack. As a result of broadening my knowledge base, I have a better idea of what areas of software development and what kinds of technologies I’d like to focus on in the future.