Curriculum Management System

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Keeran Ratnasabapathy 27658095	Keeran10	12
Irina Fakotakis 40029185	irinafakotakis	5 in progress
Georgik Barsemian 40032101	BarsemianGeorgik	8 in progress
Matthew Benchimol 27759614	MBenchimol	14 in progress
Scott Bouchard 26251625	sbouchard09	6
Antoine Betenjaneh 27161956	soenAnt	7

Project summary

CMS, Curriculum Management System, is a software-as-a-service which will allow University faculty members to easily modify their respective University Academic Calendar. This system will provide a clean and intuitive interface, as well as automation of the approval request workflow which accompanies changes to the University Academic Calendar.

Risk

The CMS is a project that has a lot of inter-dependencies going on all at once on the inside. Requirements need to be easily modified in future releases, but also very strict while they are valid and are in use. The biggest risk is making sure that the initial design which includes the database that the entire system will depend on is properly designed to handle all the different criteria the system will have. This risk was mitigated first by making sure that three people were assigned the task to design the database, review that design and make sure that it will hold up against future iterations and releases, remaining flexible enough for any future modifications by us or the stakeholder. The initial E/R diagram design of the database can be found in the **Overall Arch and Class Diagram** section.

Legal and Ethical issues

- Since the stakeholder is an associate dean at Concordia University, there are no legal obligation for the CMS team to sign Concordia's IP opt-out. Furthermore, we have decided to license the project under the MIT agreement.
- There is an ethical issue concerning the transfer of sensitive documents pertaining to the curriculum approval process. These are provided to the team for study purpose exclusively; therefore, these documents should not be leaked. Such action would lose stakeholder trust.

Velocity

Project Total: 2 stories, 13 points over 2 weeks

Iteration 1 (2 stories, 13 points)

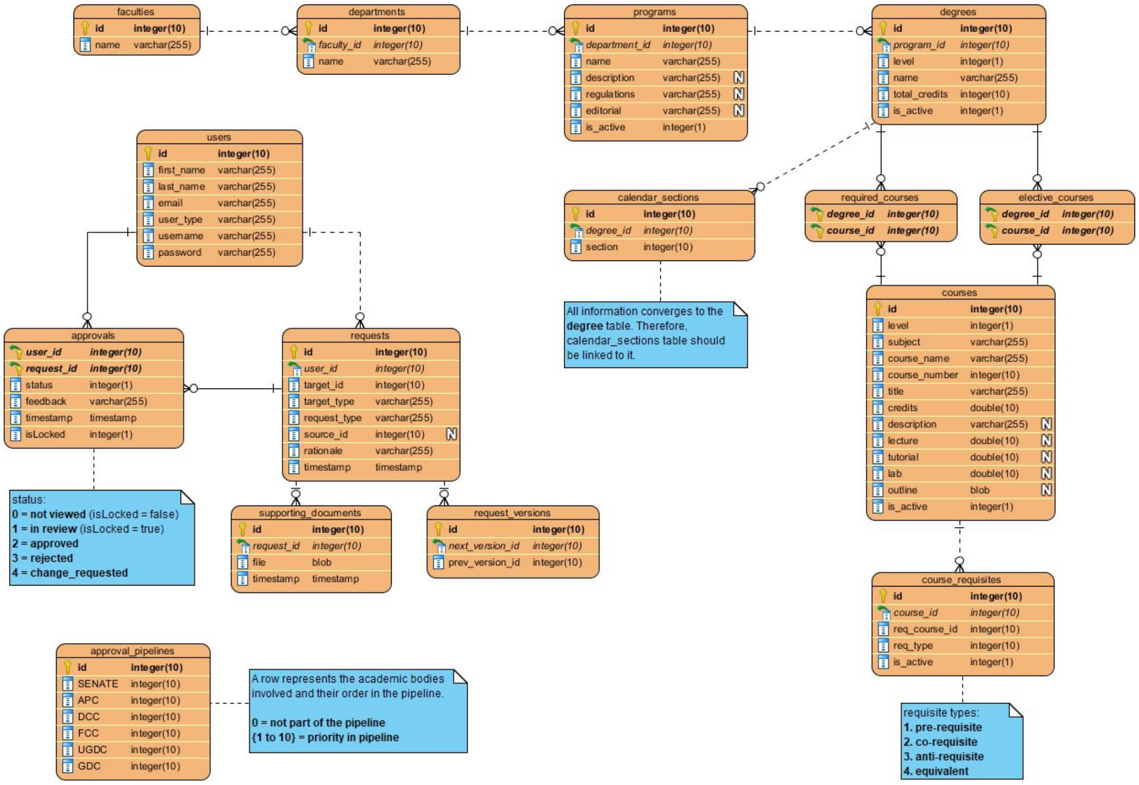
Iteration 1 was focused on project setup, mitigating project risks before beginning to work on features. The project was containerized via Docker, TravisCI was integrated, mockito and logging libraries were added as well as linting and the feature tag config file, all to prepare for future development.

Iteration 2 (2 stories, 25 points)

Iteration 2 focused on developing the course search functionality. A local database was added to store the courses with sample data. Search functionality was then added to retrieve courses from the database. Finally, backend functionality was implemented for adding/retrieving supporting documents for a given course.

Overall Arch and Class diagram

Provided below is our initial design for the CMS database. We believe these associations will provide University faculty with a smart system which displays the impact of change to their calendar.



All information converges to the **degree** table. Therefore, calendar_sections table should be linked to it.

status:
0 = not viewed (isLocked = false)
1 = in review (isLocked = true)
2 = approved
3 = rejected
4 = change_requested

A row represents the academic bodies involved and their order in the pipeline.

0 = not part of the pipeline
{1 to 10} = priority in pipeline

requisite types:
1. pre-requisite
2. co-requisite
3. anti-requisite
4. equivalent

Figure 1 - E/R Diagram

Next, the package view of the class diagram is provided along with a package diagram for each component. Note, these will be updated as more classes are added to the project.

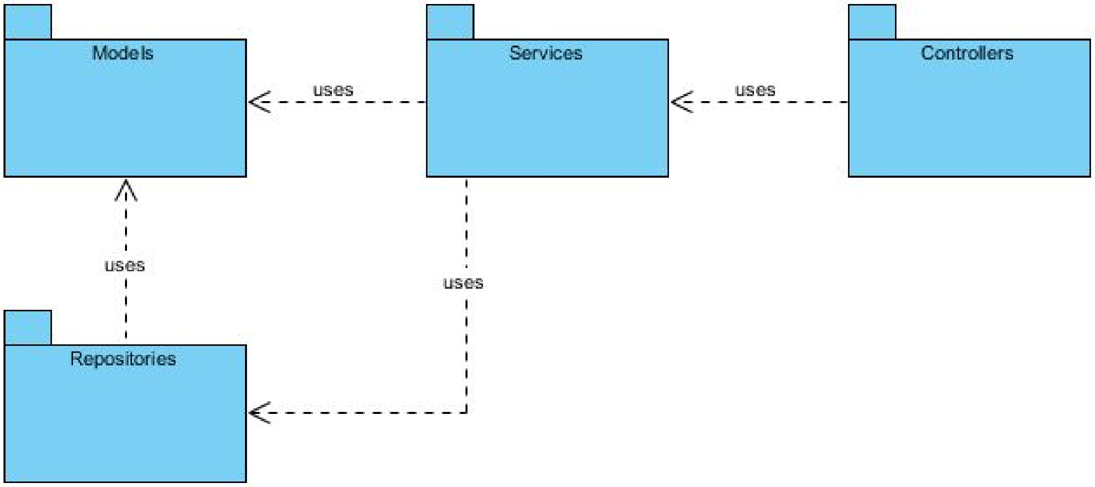


Figure 2 - Class Diagram: Package View

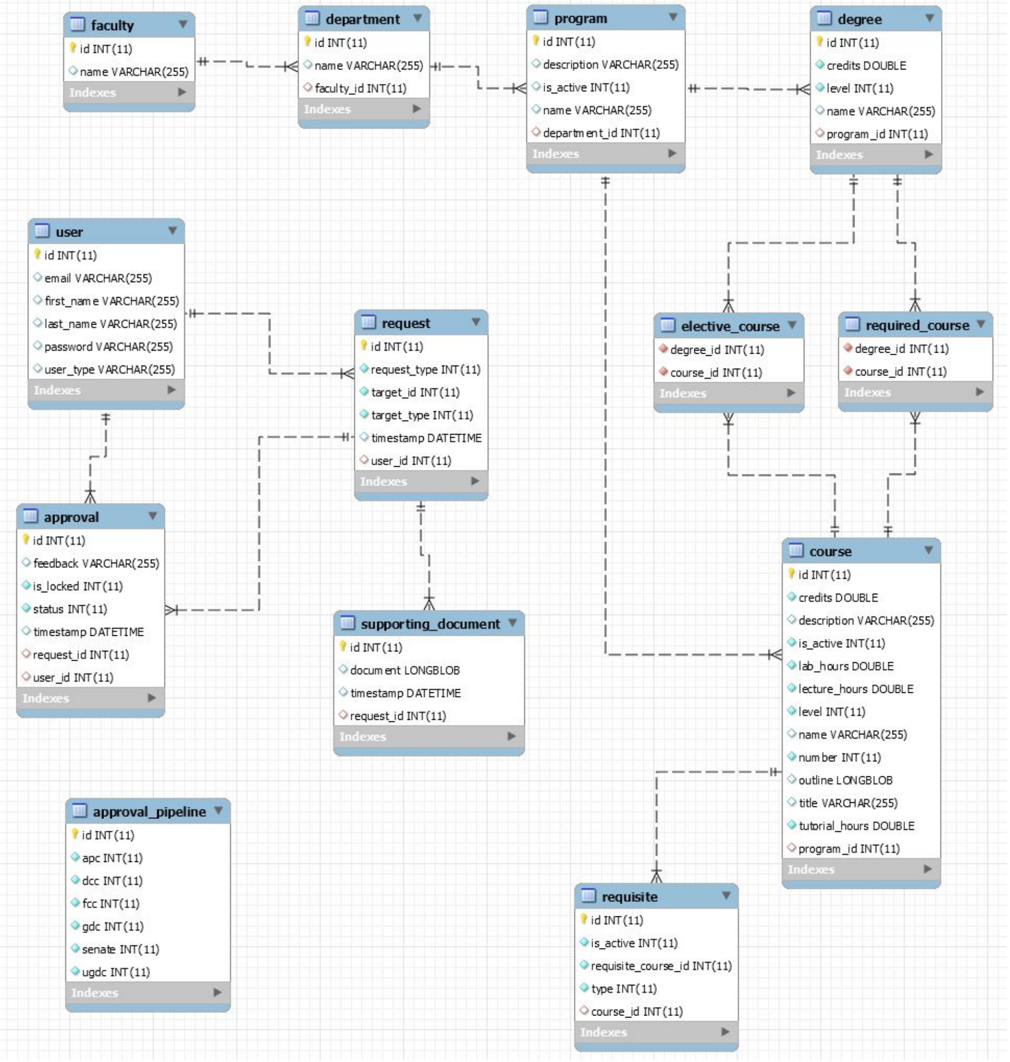


Figure 3 - Model/Database Package

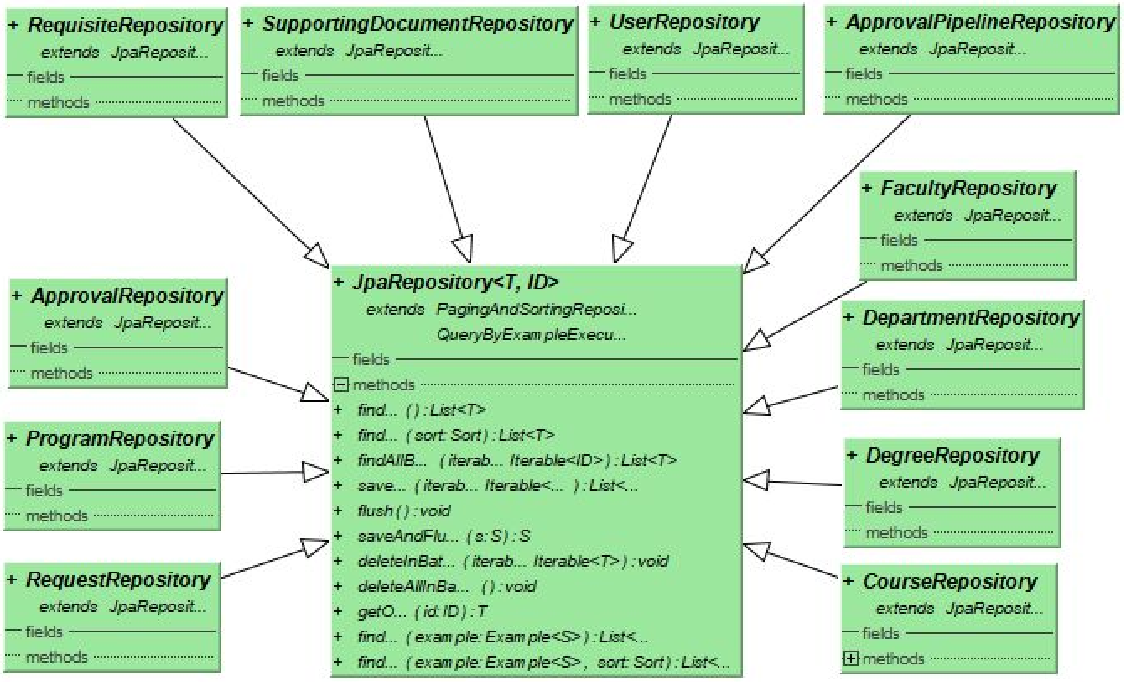


Figure 4 - Repository Package

Infrastructure

For the CMS project, we are using:

Docker

Docker helps to containerize the project to reduce runtime and the deployment process. By dockerizing the project it allows us to deploy easily on different platforms such as Windows, Linux and Mac. With no current dependencies with specific operating systems, the containers are also very lightweight and are easily shareable between different parties.

Travis Ci.

Travis allows to evaluate every commit that is happening to our GitHub repository. It is mainly a build machine that tests if the new commit is possibly breaking the compilation. It allows to notify all the development team whenever a build fails or succeeds. Most importantly Travis Ci comes with a free cloud based hosting environment and is easily integrate with the GitHub repository.

Apache Maven Checkstyle Plugin

Maven's checkstyle plugin allows the project's code to be verified. The code is checked for violation such as indentation and variable namings. The plugin produced a report of an HTML format that can be accessed to see the number of count of errors and warning. The report also directs to the specific class along with its line number to show the code standard violation. This enforces the project to adopt certain coding standards and rules.

Jacoco

Jacoco is a Code coverage tool. With the code coverage its possible to evaluate if all the classes and functions in the project are being tested inside the unit test directory. The code coverage tool produces a HTML report with the percentage of each class evaluating if the entire class was tested or not, This tool enforces that the development team tests all of their functions.

MySQL

MySQL will be used as our database management system for all the project's datas, It provides data security, since the product is going to be used by the university to update course calendar therefore its important that the data remains secure so that it cannot be compromised and modified externally. It also has high performance compared to the other types of databases, the connection and the transfer of the requests are always done time efficiently and at high speed. Lastly, MySQL is highly scalable whenever the demand of storing data rises with time. That feature could be beneficial since a program curriculum might require more database usage with time.

Angular

Angular is known to be two-way data binding either interacting from the server to the frontend or sending data from the frontend to the backend. It helps to improve the server performance as well.

Spring

Java Spring can quickly start up a project and provides HTTP endpoints to integrate api calls easily. It also provides an implementation of injection system to facilitate object creation.

Lombok

Lombock provides tools to easily log different process states within the project. It is easily integratable within the Spring Framework. And it can be called very easily and statically from anywhere in the project in order to make any type of log. Logging allows the team to collect user data and it helps detect any type of code bugs and any type of crashes. The logs can help the team to retrace the client's step to reproduce the bug,

In further iterations, the advantages of the different frameworks and libraries that we are planning to use will further be discussed and explained once the development of the project progresses.

Name Conventions

We will be following the following naming conventions for our project:

- Front-End: Angular Style Guide
- Back-End: Java and Spring Bean Naming Conventions
- Database: MySQL Naming Convention

Code

The code for this iteration implements the tasks as described in the Course Search issue that can be found here

File path with clickable GitHub link	Purpose (1 line description)
/src/cms-client/src/app/search-page/search-page.component.ts	Defines the front end logic for the search page
/src/cms-client/src/app/search-page/search-page.component.html	HTML representation of the search page

/src/cms-api/src/main/java/com/soen490/cms/Controllers/CourseController.java	Implements the CourseController class which handles the user's course search requests
/src/cms-api/src/main/java/com/soen490/cms/Services/CourseService.java	Implements the CourseService class which contains the course search functions
/src/cms-api/src/main/java/com/soen490/cms/Models/Course.java	Contains the Course model

Testing and Continuous Integration

Test File path with clickable GitHub link	What is it testing
Initial dummy test JUnit/Mockito	Testing the correct importation of JUnit/Mockito into the project.

We are using TravisCI as our continuous integration environment for testing and deployment. TravisCI is very useful as it builds and runs the project in a simulated environment and makes sure that no commit pushed breaks the build. Every commit pushed triggers TravisCI and re-builds and runs the entire project. A notification of a build failure would advise the member that caused the build failure to investigate the cause of this failure. TravisCI does provide a detailed log of the build operation with a message that can provide clues as to why the build failed. Another benefit to using TravisCI is the fact that it can separate the building/testing processes into two or more stages that run subsequent to each other, independently and a stage cannot proceed if the stage before it has failed. For more information, please click here.