Curriculum Management System - Iteration 5

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Keeran Ratnasabapathy 27658095	Keeran10	3 SP
Irina Fakotakis 40029185	irinafakotakis	4 SP
Georgik Barsemian 40032101	BarsemianGeorgik	1 SP
Matthew Benchimol 27759614	MBenchimol	5 SP
Scott Bouchard 26251625	sbouchard09	4 SP in progress
Antoine Betenjaneh 27161956	soenAnt	7 SP

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 database design needs to be flexible to encompass the entire academic calendar. Many outliers exist in the calendar which can't be automated. Three team members are constantly working on refactoring and maintaining the

database through a one-to-one mapping with the model package found *Overall Arch and Class Diagram* section.

- The search feature might not handle every query accurately to retrieve programs and courses. To circumvent this need, users will be able to list all existing programs and courses in the off-chance the queries fail.
- The impact statement feature might not generate accurate reports all the time. This can happen during database expansion where new relations are formed between programs, degrees and courses. To solve this, we will provide the date and version of the database used to generate each report.
- The system needs to be compatible with the Student Information System and therefore some required database fields need to be added to the appropriate tables.

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: 11 stories, 174 points over 8 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.

Iteration 3 (4 stories, 50 points)

Iteration 3 focused on developing the course request functionality to handle change requests for a course by the user. The course impact statement was also developed such that course change requests which affect other courses in the database (such as prerequisites) will be known as to fix any potential conflicts. Database and model classes have been refactored to meet increasing demand such as the addition of the package class which aggregates requests from a user.

Iteration 4 - Release 1 (3 stories, 86 points)

Iteration 4, the end of Release 1, focused on completing all the components to the following use case: user logs into the system, user sees all the (course request) packages. From there, the user can now 1. Edit a request and see the impact, 2. Create a new request and see the impact, 3. Submit a package and choose the approval flow. 4. Generate the package pdf file. 5. Track their package in the approval pipeline to see its status and its progression.

Iteration 5 (0 story, 18 points)

Iteration 5 focused on bug fixes/refactoring needed after Release 1.

Overall Arch and Class diagram

Provided below is the architecture for the CMS system. We believe these associations will provide University faculty with a smart system which displays the impact of change to their calendar. The package view of the class diagram is provided along with an expanded version of the package diagram for each component. Note, these will be updated as more classes are added to the project.

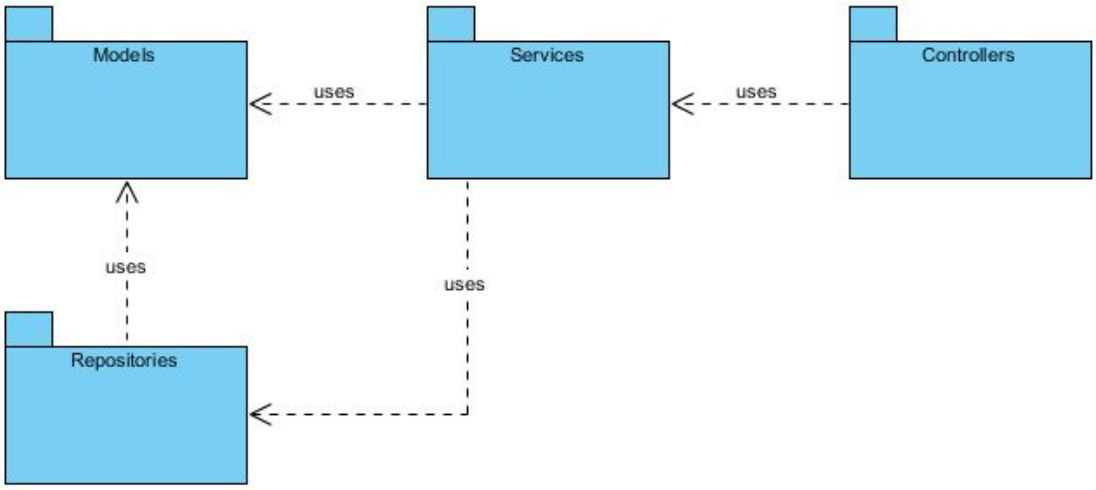

Figure 1 - Class Diagram: Package View

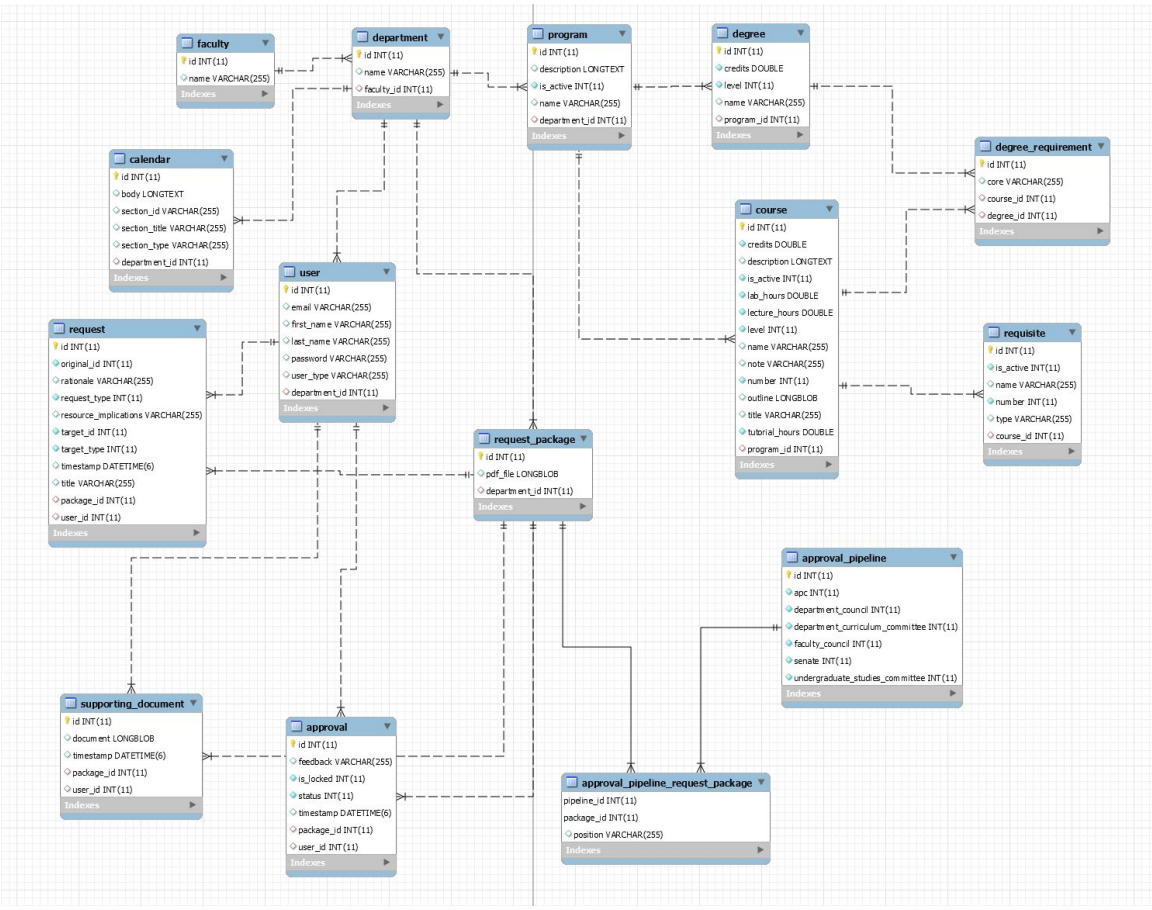

Figure 2 - Model/Database Package

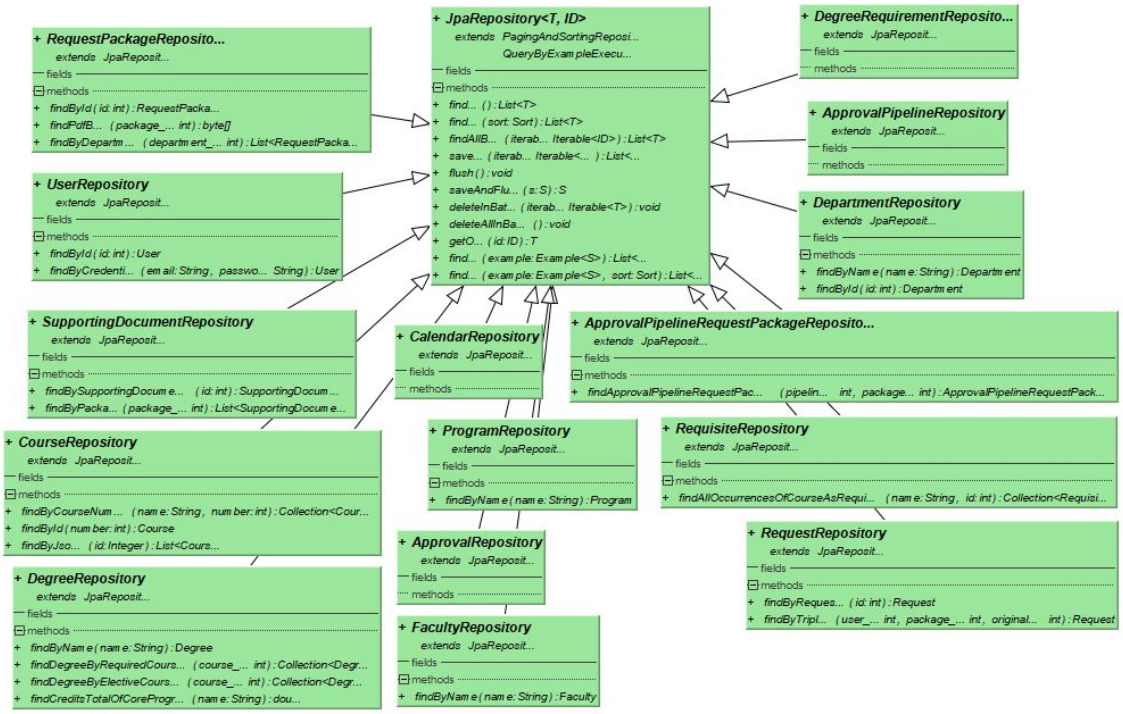

Figure 3 - Repository Package

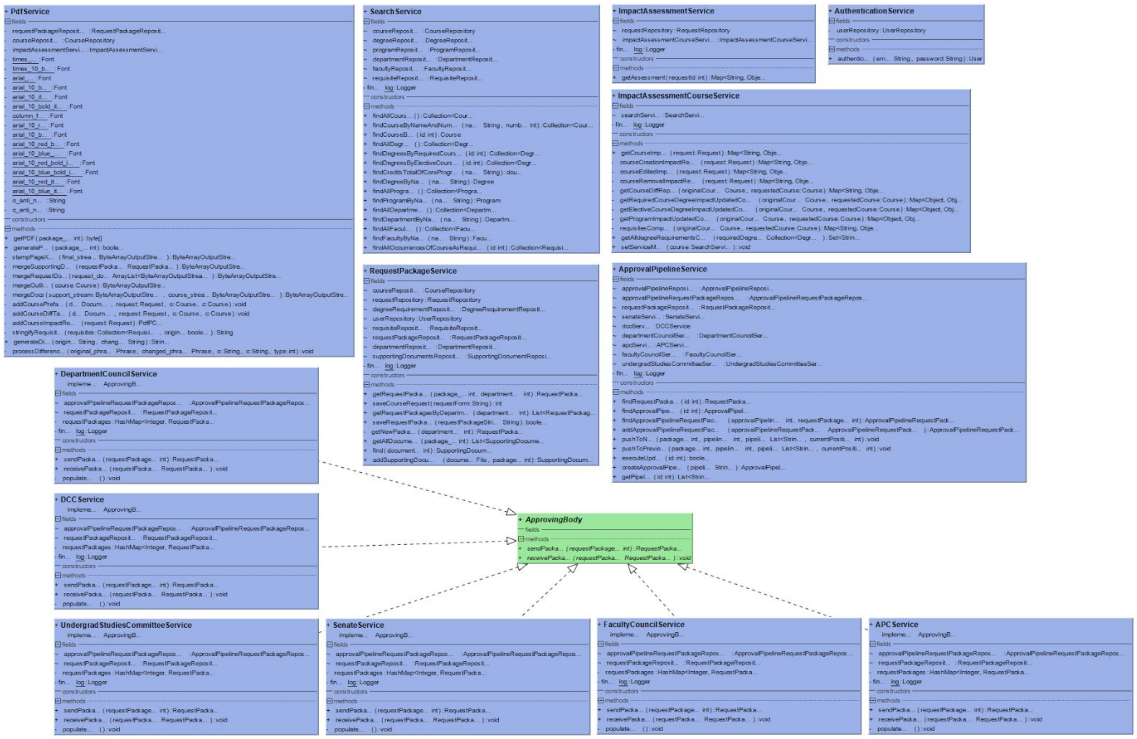

Figure 4 - Service Package

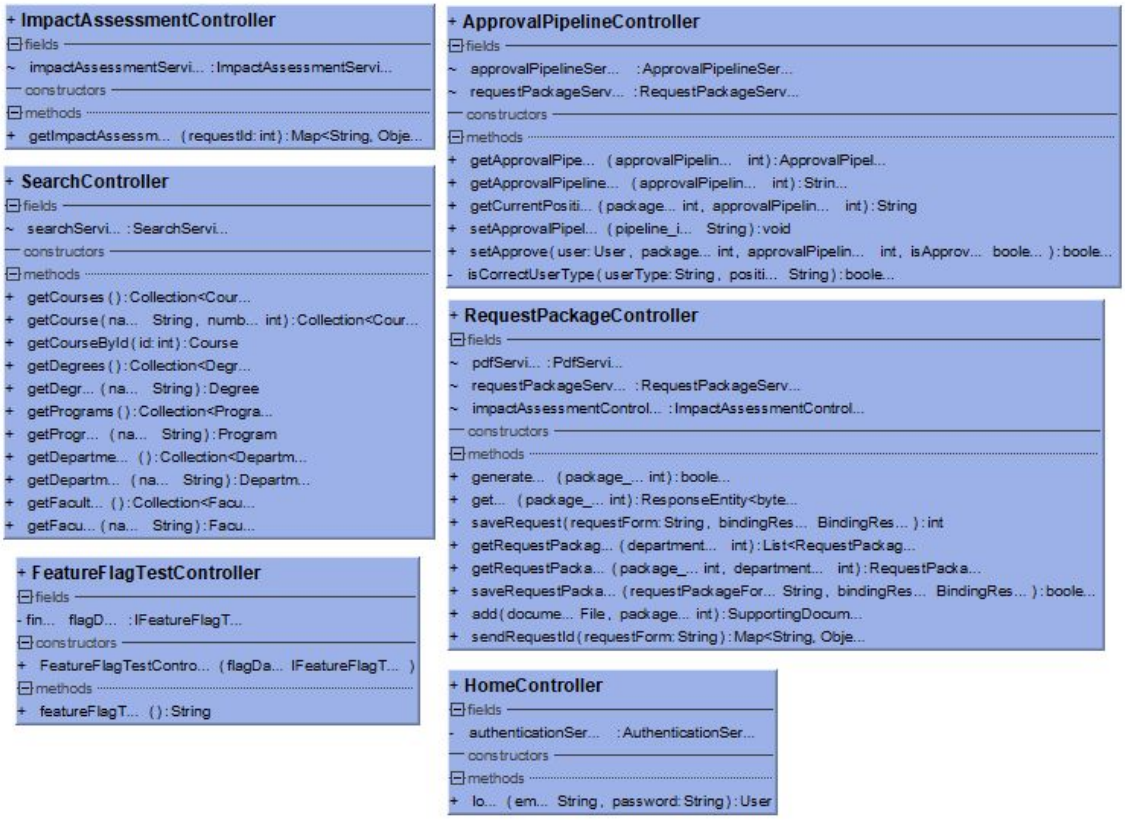

Figure 5 - Controller Package

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 data, 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.

Diff Util

Diff Utils library is an OpenSource library for performing the comparison. It takes in usually two arrays of characters, one representing the original text field and the other the updated text. And then the tool will draw a line over all the elements removed from the original text and will make it in bold everything changed and added on the second updated text. This is the ideal tool to have a side by side comparisons between texts.

IText

IText provides the core functionalities to generate a PDF type of document in the language of Java. It provides detailed functionalities to provide columns and rows. Along with a variety of font classes to choose from and text sizes. The library provides clear interfaces which facilitates for a Pdf document creation and to be downloaded by the end user.

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 focuses on bug fixes and refactoring needed after Release

1. All of the issues relevant to Iteration 5 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	Contains the search page component which allows search results to be displayed when the user performs a search
/src/cms-client/src/app/pipeline-tracking/pipeline-tracking.component.ts	Contains the tracking component which tracks which approving body is currently evaluating the request package in the approval pipeline
/src/cms-client/src/app/backend-api.service.ts	Connects the front-end to the back-end
/src/cms-api/src/main/java/com/soen490/cms/Services/RequestPackageService.java	Service class used to create and manipulate request packages
/src/cms-api/src/main/java/com/soen490/cms/Services/PdfService.java	Service used to generate a PDF for a request package, merge supporting documents into one PDF and return a PDF for a request package

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.

https://github.com/Keeran10/CurriculumManagementSystem/blob/search_feature/src/cms-api/src/test/java/com/soen490/cms/SearchTests.java	Tests to verify functionality of Search feature endpoints on the server-side.
https://github.com/Keeran10/CurriculumManagementSystem/blob/dev/src/cms-api/src/test/java/com/soen490/cms/ImpactStatementControllerTest.java	Tests to verify functionality of impact controller.
https://github.com/Keeran10/CurriculumManagementSystem/blob/dev/src/cms-api/src/test/java/com/soen490/cms/ImpactStatementUnitTest.java	Tests to verify functionality of impact service.
https://github.com/Keeran10/CurriculumManagementSystem/blob/dev/src/cms-client/src/app/backend-api.service.spec.ts	Tests to verify client connection to server endpoints.
https://github.com/Keeran10/CurriculumManagementSystem/blob/dev/src/cms-client/src/app/edit-form/edit-form.component.spec.ts	Tests to verify edit-forms for course modifications.
https://github.com/Keeran10/CurriculumManagementSystem/commit/42c969309f7610811a63af03360d2f296c0e8bbf	Assert functionality of third-party diff tool for String comparison.
https://github.com/Keeran10/CurriculumManagementSystem/commit/7bfb688f97306dbd2a18bc809d2422928ba346bf	Assert json data manipulation for saving course requests
https://github.com/Keeran10/CurriculumManagementSystem/blob/logs_tests/src/cms-api/src/test/java/com/soen490/cms/PdfServiceTests.java	confirmDiffPattern tests and proves that the diff tool has a particular pattern of storing differences in odd-numbered indexes in array.
https://github.com/Keeran10/CurriculumManagementSystem/blob/dev/src/cms-api/src/test/java/com/soen490/cms/ApprovalPipelineTest.java	tests that the current position isn't null and that the approval pipeline isn't null
https://github.com/Keeran10/CurriculumManagementSystem/pull/123	Front-end tests that cover packages, login, impact statement, course-list and approval pipeline.

For Iteration 5, the focus was to cover tests from the front-end side. The goal was to cover as much of the front-end to ensure that any future additions do not break backward compatibility with the previous features that are set in place. The way the front-end testing works was to setup the necessary conditions for the test by

creating the data injected objects and to ‘spy’ on the functions within them. When necessary, we are calling the functions being tested are called to check if the changes expected to happen do happen after the function gets called. The tests written were covering the most important functions being called and are crucial for the front-end to work.

For Iteration 4, we focused on creating more tests to cover some varied important functionalities that are critical to the app. For example, the backend sends data to the frontend packaged as a json object and as such, assuring that the data stored and sent back is valid is of utmost importance. One of the critical tests that were created for this functionality is a test to assert the json data manipulation for one of the features where json files are used, saving the course requests. Another big feature that was introduced to this iteration is the approval pipeline, a feature that is extremely critical to the entire project and as such, the test created for it makes sure that the current position of the approval pipeline as well as the approval pipeline itself are not null. As always, all of those tests are successfully linked to TravisCI to ensure continuous integration and every update done gets verified to ensure backwards compatibility.

For Iteration 3, TravisCI can now successfully distinguish between the backend and frontend aspects of the application. Tests that are performed with the Spring backend and Angular front-end run successfully. We also restricted the runs of TravisCI to runs on the Dev and Master branches to increase efficiency and productivity of the building and testing process. Furthermore, we also created an

in-memory database for testing purposes which enables TravisCI to run those tests without the need to connect to the MySQL server, further increasing the efficiency of TravisCI.

For Iteration 1 and 2, we implemented 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.