

Murdoch University

CourseLoop - Callista Parser Requirements and Analysis

ICT302 IT Professional Practice Project,
Team (10) Black Widow

Blake Williams, Enrique Getino Reboso, Jordan Chang-Yeat
Lau, Patrick NKhata, Jared Eldin and Sara
Hussain

25-10-2019

Version 2.0

Table of Contents

Executive Summary	2
Introduction	2
Project Purpose	2
Background	3
Problems and Opportunities	3-4
Objectives and Goals	4
Methodology	4-5
Solution Outline	5
Scope	5
Narrative Description	5-6
Functional Requirements	6-9
Primary	6-9
Secondary	6-9
Non- Functional Requirements	9-11
Diagrammatic representation of the requirements	12-14
Conclusion	15
Appendices	15

Executive Summary

Murdoch University has in place a manual process where the staff communicate information between two applications, CourseLoop to Callista. Due to the likelihood of human error, the University has assigned Black Widow to create a software solution that allows some degree of automation to reduce human error and to provide efficient and timely information to students.

This project purposes the requirements for the extraction of the course structural information from the CourseLoop curriculum management system via the use of a parser software solution, which will then be transformed to the course and unit set rule strings and that can be loaded to the Callista student management systems using the Callista standard rule strings.

This document outlines the problems with the current application communication processes as well as the benefits of implementing an automation tool to reduce human interaction. Also outlined are:

- A methodology
- A solution
- Primary and secondary Functional and Non-Functional requirements
- Diagrammatic requirements

Black Widow aims to develop the aforementioned web application for a user to parse the CourseLoop data into Callista strings manually, to ensure that students receive up to date course and unit information to increase successful and accurate enrolments.

Introduction

Project Purpose

Black Widow purposes the extraction of the course structural information from CourseLoop curriculum management system via web application and transform the payload into course and unit set rule strings for use in the Callista student management systems, utilising Callista standard rules syntax. The software will aim for parsing and translating CourseLoop data into strings that meet the defined business rules. The software application will also warn the user if by mistake, a course json file has been dropped into the major area or vice versa.

This project is needed in order to automate and facilitate the correct communication between CourseLoop and Callista. This will assist in providing a better understanding of the Handbook by students, University staff and users in general. It will also provide a better procedure in terms of managing the communication between both systems, as the software will be designed to try to avoid human errors in the processes executed. This software will be oriented towards the simplification of the communication processes between CourseLoop and Callista, avoiding human interaction as much as possible.

Background

The current process that is being used for communication between CourseLoop and Callista is not automated. It is necessary to take the data generated by the first application and handle it manually so the second application can read the payload extracted. This process is being carried out by university staff, which means that human resources are used in the communication management of both platforms. The design and development of a software solution that performs the parsing and translation functions between both systems would be very beneficial for better management of both, the information and the resources of the university. This software should ease the mentioned processes as well as make them faster, releasing workload from the university staff who is carrying out these tasks at the moment.

Problems and Opportunities

PROBLEMS

- The actual communication between CourseLoop and Callista is done manually by university staff, to create a new course a course structure is made, then an employee has to manually create a Callista string out of the new course structure to make the courses compatible with the system.
- It is time consuming and the fact that the process is done manually can always cause human errors which could cause malfunctions in the systems.
- A malfunction or error in the system may cause that the Handbook shows wrong or incomplete information about the course structure, leading the user to be misinformed about the requisites needed to obtain a degree.

OPPORTUNITIES

- The creation and implementation of the software could mean an enhancement of the whole communication process between systems.
- The human resources used to perform the process manually could save a considerable amount of time which could be used in other fields.
- Automation of this type of processes can also help to identify errors in the communication of the systems.
- The software could be adapted to perform other tasks if needed.

Objectives and Goals

The objectives of the software, as stated in the previous sections, is to automate the parsing and translation processes between both platforms, CourseLoop and Callista. Our goal is, as requested by the client, to ensure that at least 80% of communications between both platforms is accurate and without failures.

Methodology

The methodology that Black Widow has decided to proceed with this project is the Agile methodology. We believe that Agile will be the most appropriate methodology for this project due to the application needing constant iterations for a perfected product. Breaking bigger tasks (stories) into smaller tasks (activities) allows the team to react to immediate feedback set by both the supervisor and the client, where as if Black Widow followed the waterfall method, we would have to retrace all of our steps in order to respond to feedback. SCRUM is also an excellent tool bundled with the Agile methodology (which we have deployed via a web application called Trello), in turn the team gets a clear indication of where activities are at in their stage, what the member is doing what task and an overall summary of how far the stories are from being completed. The Agile methodology is used in many other IT projects, and we believe it suits our needs perfectly.

Solution Outline

Scope

The project is to take course structural information from CourseLoop curriculum management system via web app and transform the payload in to course and unit set rule strings for use in the Callista student management systems utilising Callista standard rules syntax.

The software should create strings for all course and unit set changes meeting defined business rules. The software should include error warnings, for example where the user operating the software uploads the payload file to the wrong input box.

Narrative Description

The implementation of the system will be through the Javascript programming language, using the typescript superset to add language features that will aid in development. Most notably static type checking which should make code more robust and aid in debugging

data type errors which javascript development is prone to. The application will be developed on a node.js server, serving a web page to the user to process the data locally. This will make deploying the finished application intuitive and simple in most environments, as well as accessible anywhere on the network.

Receive input from CourseLoop

- Translation of the structural information from CourseLoop into strings readable by Callista.
- This task is critical and is the main priority. It requires being able to receiving an input from CourseLoop, making it its dependency. A major risk associated with this task is that if the input is incorrectly translated, we could be creating the wrong string to be used in the Callista system. This task will be initiated after the design document is complete.

Output translation to Callista system

Data must be in a string format following the syntax rules of Callista platform

Functional Requirements

REQ-ID	Description	Criticality (1-5)	Priority (LOW, MEDIUM, HIGH)	Requires
REQ-1	Extract required data from Courseloop payload	2	MEDIUM	Courseloop payload (Major or course)

REQ-2	Transform payload into Callista standard rules syntax query string	5	HIGH	REQ-1
REQ-3	Verify output string	4	HIGH	REQ-2
REQ-4	Output callista string to user	3	MEDIUM	REQ-4

Breakdown of Functional Requirements

REQ-1

1. Extract the required payload from the CourseLoop system and sort it into relevant data structures required to process the data into a string query as easily as possible (these data structures will be designed and detailed after we have example query strings to base them off)

REQ-2

1. Use loops and/or Regex on the data structures created in REQ-1 to create the Callista query string

REQ-3

1. Verify string meets requirements by comparing unique aspects of a standard Callista string to the string generated in REQ-2. This step will likely use Regex to determine string validity.

2. Depending on the outcome of step 1; mark string VALID if passed tests or INVALID if failed.

REQ-4

1. Return the query string to the user requesting the data.
2. If the system attempts to return back a second query string catch the data and handle as an exception (dealing with a query string that returns another query string is out of the scope of this project, it will be documented how to catch the data and ideas on how to work with it for future improvements to the application).

Schedule

Reference the project management plan and Gantt charts for further scheduling details of the overall project. After the detailed design phase further details will be uncovered and task scheduling will be broken down further for each functional requirement before implementation begins. As a general guide development will begin with Functional requirements which have no other requirements. Functional requirements with the most prerequisite requirements to ensure functionality will be completed last.

Limitations of Current Functional Requirements

Development – Limited to user input as there is no access to the systems

User can still make mistakes parsing the data into the Callista system.

Non- Functional Requirements

- **PERFORMANCE:** System must process CourseLoop data structures in a timely manner to ensure systems relying on the produced query strings aren't delayed when calling the end product software solution application (especially important for systems that produce and consume these strings in real time and are user facing). Anything over 10 seconds would be considered unacceptable even under peak load and generally in

lower load environments a significantly faster time would be expected and desired (Less than 1 seconds).

- BOTTLENECKS: the software should be able to process as fast as the user can upload. Disk read/writes should be much faster, and the main bottleneck will be network latency and speed of transferring data over the network from CouseLoop and Callista too.
- QUALITY: The finished application should be able to automate translation from the CourseLoop software application to a Callista standard rules syntax query string for at least 80% of the most common Callista query strings. Uptime should target 98% and generally beat this goal. System restarts should target having the system back up and running within 5 minutes.
- PORTABILITY: The software should be able to run on common server hardware and common computers. The software application will be built in NodeJS, serving a HTML webpage with javascript, the application will be largely portable between operating systems - which will simply need to install NodeJS, git, npm and a web browser to run the software solution. Therefore, this software application will be suitable to run on a large variation of host machines, aiding in deployability options.
- DOCUMENTATION: Required for each method, and a description of its relationship to other methods and how an IT department could implement the overall system with current systems will need to be provided with the finished project to enable using and potentially upgrading the application in the future.
- ERRORS: The software should be able to handle errors such as invalid or corrupted input, Courseloop software application HTTP error codes and handle crashes by automatically restarting the application and recovering job tickets. Where The program crash is repeated and likely to a job ticket logic error the queue should be emptied after several failed restart attempts to allow an automatic restart.
- SECURITY: The application will reside on a secure server with no internet connection, any user that is connected to the network will be able to request the web page off the

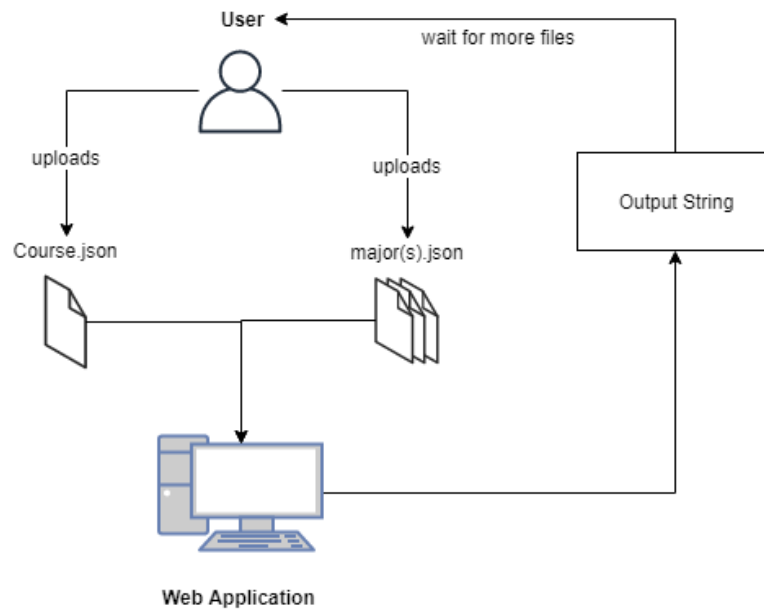
server to parse a course or major payload locally, a firewall between the server and the connection will assist the server from receiving rogue connections.

- **BACKUPS:** A copy of the software should be kept offline and in the cloud to ensure it can be recovered if the host system fails and data is lost. a version control system is used (GitHub) so old versions can be archived and recovered if issues with an upgraded software solution are discovered. Modifications should also use Version control techniques so old versions are available to rollback in case of issues.
- **MODIFICATIONS:** The software currently pertains to the existing applications' structure. In the future, should Murdoch developers make changes to the existing systems, the interactive software will also require modifications to suit the evolving needs. Documentation should be referenced first to ensure the functionality isn't already available. From there old software should be backed up and then new changes can be implemented. If the form of the string Callista consumes or variations of strings are required, upgrades would be expected.

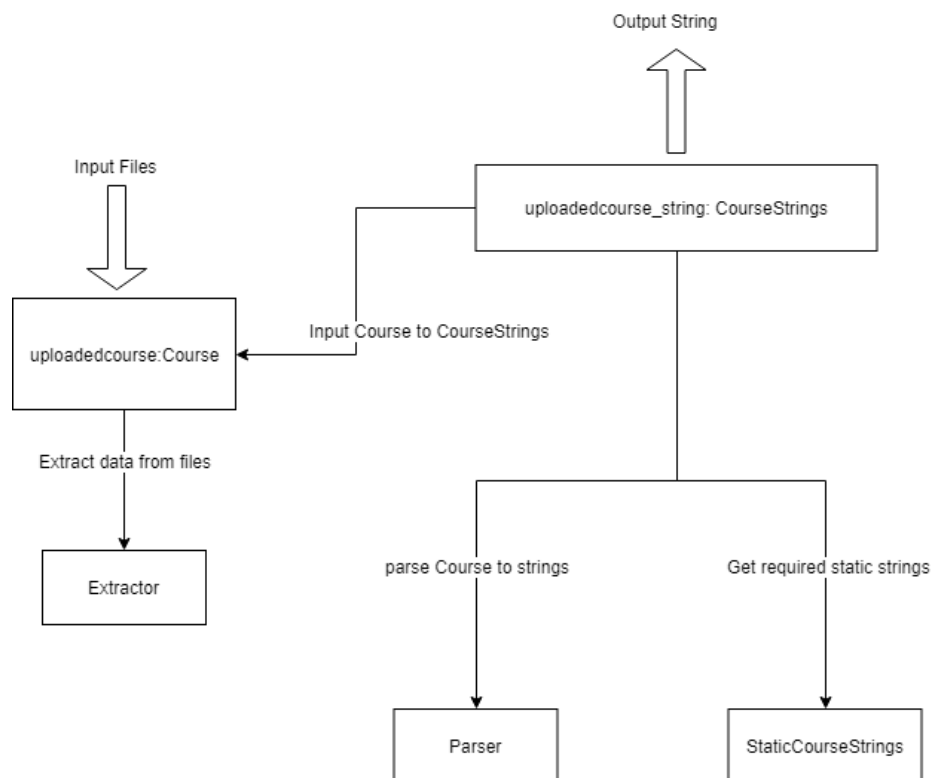
Diagrammatic representation of the requirements

CourseLoop-Callista Parser Process Flow

1. User opens web page
2. User uploads CourseLoop payload
3. Application parses CourseLoop payload
4. Application outputs Callista string



Data Flow Model



Conclusion

Murdoch's current method of manually sharing information between two systems, CourseLoop and Callista, via human input leaves room for unintentional errors. Therefore, Black Widow has been assigned the task of automating the CourseLoop to Callista interaction to reduce human interaction to provide efficient and accurate information to all users. The team has purposed the software application as a middle-man to cancel out all human interaction amongst the two systems. This document outlines problems with the current systems communications as well as SCRUM as an agile methodology that will be used to complete the end-product. Also included is an outline of the solution and the functional and non-functional requirements of application in addition to a diagrammatic representation of these requirements.

Appendices

Appendix A – Deliverable Task Breakdown Statement (signed, scanned and inserted into document)

Unfortunately, due to having external students in the group, the document could not be signed and scanned. Nevertheless, all team members participated in the creation of the R&A document.

Requirements and Analysis

Action to be taken	Person/s responsible	Action to be completed by
Executive Summary	Enrique Sara	23/10/19
Introduction	Enrique	23/10/19
Solution outline	Enrique Jordan	23/10/19
Functional requirements	Blake Jared	23/10/19
Non-functional requirements	Jared	23/10/19
Diagrammatic representation of the requirements	Patrick	23/10/19

Appendix B: - Copies of client documents on which you based the analysis.

Unit Set Rules Specification Document

When you start to build a unit set completion rule, below are the Rule options from which you can select:

Rule Options	
▲	(
▼	For
	Must complete all units in
	Must complete all units in training package set
	Must complete special requirement of type
	Must complete stage of type
	Must have completed
▼	Must have enrolled and completed
▲	Must not exceed
	Must pass
	Must pass SVH minimum contact hours for course
	Must pass all units in
	Must pass all units in training package set
	Must pass credit points for course with no more than
	Must pass minimum contact hours for course
▼	Must pass minimum contact hours for course with no more than
▲	Must pass
	Must pass SVH minimum contact hours for course
	Must pass all units in
	Must pass all units in training package set
	Must pass credit points for course with no more than
	Must pass minimum contact hours for course
	Must pass minimum contact hours for course with no more than
▼	true

I select 'Must pass all units in'. When a rule option is selected, the list of available Rule Options may reduce or change.

Rule Text	
Must pass all units in	
Options/Validate	Unprocessed:

Rule Options	
▲	Unit Set
▼	unit set
	{

Rule Text	
Must pass all units in {	
Options/Validate	Unprocessed:

Rule Options	
▲	*** Valid Unit Code ***
▼	
▼	
▼	

In the screenshot below I selected the bracket '{'. Now I need to type in a valid unit code

I type in 'COM107' and click Options/Validate button. Rule Options available are:



- Comma – if I want to add another unit
- Fullstop – if I want the rule to apply to only specific versions of a unit (eg. COM107.1)
- End Bracket – if I only want to specify one unit and no more.

Rule Text	
Must pass all units in {COM107	
Options/Validate	Unprocessed:

Rule Options	
▲	,
▼	.
▼	}
▼	

I have 6 core major units I need to specify in this rule, so I select the comma and the Rule Options advise me to type in another valid unit code.







Rule Text	
Must pass all units in {COM107,	
Options/Validate	Unprocessed:

Rule Options	
	*** Valid Unit Code ***
	

I repeat the above steps for my six units then I select the End bracket.

Parse Successful means that I have built a complete rule and can save and exit if I want.

Rule Text	
Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 }	
Options/Validate	Unprocessed:

Rule Options	
	&
	*** Parse Successful ***
	+
	And
	Or
	

This major also has specified electives so I need to append another rule. So I select '&' and get the beginning list of Rule Options once again.

Rule Text

Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 } &

Options/Validate Unprocessed:

Rule Options

- (
- For
- Must complete all units in
- Must complete all units in training package set
- Must complete special requirement of type
- Must complete stage of type
- Must have completed
- Must have enrolled and completed

This time I select the 'Must pass' option. Now the Rule Options advise me that I need to select a number.

Rule Text

Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 } & Must pass

Options/Validate Unprocessed:

Rule Options

- *** A Number ***

I type in '1' and click the Options/Validate button. A new set of Rule Options are now available.

Rule Text

Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 } & Must pass 1

Options/Validate

Unprocessed:

Rule Options

▲

🔍

contact hours

🔍

contact hours at levels

🔍

contact hours in

🔍

contact hours in discipline

🔍

contact hours in training package set

🔍

contact hours not in

🔍

contact hours not in training package set

▼

🔍

contact hours with no more than

In the screenshot below, you will see there are two similar options – ‘unit(s) in’ and ‘units in’. In the Rule Option Description, it states ‘units in’ is for specifying completion of units with a specific grade while ‘unit(s) in’ is for specifying a list of units that have been passed or awarded as advanced standing. I select ‘unit(s) in’.

Rule Options

▲

🔍

credit points not in

🔍

credit points with no more than

🔍

unit(s) in

🔍

unit(s) in unit class

🔍

units in

🔍

units in training package set

🔍

units not in

▼

🔍

units with no more than

Rule Options Description

Has passed or been granted advanced standing for the given set of units.

Rule Options

▲

🔍

credit points not in

🔍

credit points with no more than

🔍

unit(s) in

🔍

unit(s) in unit class

🔍

units in

🔍

units in training package set

🔍

units not in

▼

🔍

units with no more than

Rule Options Description

Must have at least the given grade.

Now I select the bracket in preparation for listing the units.

Rule Text

Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 } & Must pass 1 unit(s) in {

Options/Validate Unprocessed:

Rule Options

- unit set
- {

The Rule Options advise me to type in a valid unit code.

Rule Text

Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 } & Must pass 1 unit(s) in {

Options/Validate Unprocessed:

Rule Options

- *** Valid Unit Code ***

Like before, I type in my unit code, click Options/Validate button, select the comma from the list of Rule Options, type in another unit code and select the end bracket from the rule options. Again, one of the Rule options is *****Parse Successfully***** which means I could save and exit with a complete rule if I want.

Rule Text

Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 } & Must pass 1 unit(s) in {COM109, GRD118}

Options/Validate Unprocessed:

Plain Rule Text

Rule Options

- &
- *** Parse Successful ***
- +
- And
- Or

For this major, there is another set of specified electives so I select '&', then I select 'Must pass', then I type in '1', then I select 'unit(s) in', then I select bracket, then I type in 'COM302', then I select the comma, then I type in GRD263, then I select end bracket.

Rule Text

Must pass all units in {COM107, COM202, COM214, COM215, COM307, COM345 } & Must pass 1 unit(s) in {COM109, GRD118 } & Must pass 1 unit(s) in {COM302, GRD263}

Options/Validate Unprocessed:

Plain Rule Text (-) Comments (-)

Rule Options

- &
- *** Parse Successful ***
- +
- And
- Or

My unit set completion rule is complete so now I save and exit. Note: the Rule Options can be selected from the Rule Options list or manually typed into the Rule text field.

I think the below information will also be helpful:

All About Rule Syntax

Rules syntax is the available components of language, and their structure that are available for any particular rule. The components and structure are determined dynamically within Callista and are dependent upon either the rule subgroup or the rule description.

Rule Operators

Operators	The allowable range of symbols or text, representing functionality, available. Example. +, -, AND etc
------------------	---

Using -AND- and -OR- Rule Operators

If complex rules are created using AND/OR logic, it is recommended that the -AND- and -OR- (High Level Logical) options are used rather than the standard AND and OR operators.

e.g. Must be enrolled in at least 4 units in course teaching period(s) { SEM-1 } -AND- (Must be enrolled in at least 4 units in course teaching period(s) { SEM-2 } -OR- Must be enrolled in at least 4 units in course teaching period(s) { SEM-2 })

The use of the -AND- and -OR- options eliminates 'out of context messages' that may appear when using AND and OR.

Callista (Legacy Example)

CALLISTA B1317 BACHELOR OF SCIENCE RULES

Part 1 - Course Stage Completion rules

Must pass 24 credit points &

Must pass 18 credit points at levels {1} &

Must pass 1 unit(s) in {BSC150, MSP100} &

For SCA_LOCATION IN {DUBAI-ISC, KAPLAN-SGP} Do

Must pass 1 unit(s) in {BBS100, BSC100}

Otherwise

Must pass all units in {BSC100}

Part 2 - Course Stage Completion rules

Must pass credit points for course with no more than 30 credit points at levels {1} &

((Must have enrolled and completed 1 unit sets in {MJ-COGPSY} And Must pass 1 unit(s) in {BSC200, BSC201} And Must pass 1 unit(s) in {BSC300, BSC302}) Or (Must have enrolled and completed 1 unit sets in {MJ-MOVE, MJ-SPOR} And Must pass 1 unit(s) in {BSC200, BSC206} And Must pass 1 unit(s) in {BSC300, BSC306}) Or (Must have enrolled and completed 1 unit sets in {MJ-BIS, MJ-CMPT, MJ-CYBR, MJ-GAMS, MJ-GAMT, MJ-INW, MJ-ISC, MJ-MWAD} And Must pass 1 unit(s) in {BSC200, BSC203} And Must pass 1 unit(s) in {BSC300, BSC301}) Or (Must have enrolled and completed 1 unit sets in {MJ-ANMH, MJ-ANMS, MJ-BIOL, MJ-BIOM, MJ-CHEM, MJ-CLIN, MJ-CONS, MJ-CROP, MJ-ENGT, MJ-ENV, MJ-ENVM, MJ-FBT, MJ-MATH, MJ-MIN, MJ-MOLB, MJ-MRN, MJ-MRNB, MJ-PHYS} And Must pass 2 unit(s) in {BEN200, BEN300, BIO246, BIO247, BIO257, BIO282, BIO359, BIO367, BIO377, BIO388, BIO393, BIO394, BMS211, BMS218, BMS316, BMS317, BMS323, BMS327, BSC200, BSC300, BSC304, CHE207, CHE309, CMS100, COD302, COM103, ENG207, ENG255, ENG297, ENG299, ENG336, ENG341, ENV241, ENV303, ENV328, ENV332, ENV334, ICT283, ICT289, ICT319, ICT373, ICT374, MAS182, MAS220, MAS221, MAS222, MAS223, MAS224, MAS351, MAS352, MAS353, MAS354, SUS305}) Or (Must have enrolled and completed 1 unit sets in {MJ-BIOT} And Must pass all units in {BSC200, BSC300}))

MJ-CMPT (COMPUTER SCIENCE MAJOR) rules

Must pass all units in {ICT159, ICT167, ICT169, ICT170, MAS162} &

Must pass all units in {ICT283, ICT284, ICT285} &

Must pass all units in {ICT319, ICT373, ICT374} &

Must pass 1 unit(s) in unit set BREADTHUNI

MJ-ENGT (ENGINEERING TECHNOLOGY MAJOR) RULES

For COURSE_CODE IN {B1324, H1250} Do

((Must pass all units in {ENG125, ENG192, MAS182} Or Must pass all units in {CHE144, ENG125, MAS161, MAS182, PEN152}) And Must pass all units in {ENG207} And Must pass 1 unit(s) in {ENG297, ENG298} And Must pass 1 unit(s) in {MAS220, MAS221} And Must pass 1 unit(s) in {ENG294, ENG299} And Must pass 1 unit(s) in {ENG308, ENG309, ENG311, ENG317, ENG318, ENG337} And Must pass 1 unit(s) in {ENG319, ENG321, ENG322, ENG323, ENG338, ENG339} And Must pass 1 unit(s) in unit set BREADTHUNI)

Otherwise

((Must pass all units in {ENG125, ENG192, MAS182} Or Must pass all units in {CHE144, ENG125, MAS161, MAS182, PEN152}) And Must pass all units in {ENG207} And Must pass 1 unit(s) in {ENG297, ENG298} And Must pass 1 unit(s) in {MAS220, MAS221} And Must pass 1 unit(s) in {ENG294, ENG299} And Must pass 1 unit(s) in {ENG308, ENG309, ENG311, ENG317, ENG318, ENG337} And Must pass 1 unit(s) in {ENG319, ENG321, ENG322, ENG323, ENG338, ENG339} And Must pass 2 unit(s) in unit set BREADTHUNI)

MJ-ACC (ACCOUNTING MAJOR) RULES

For COURSE_CODE IN {B1325, H1249} Do

Must pass all units in {BSL165, BUS130, BUS170, BUS171, BUS176} &

Must pass all units in {BUS285, BUS286, BUS287} &

Must pass all units in {BSL305, BUS304, BUS356} &

Must pass 1 unit(s) in unit set BREADTHUNI

Otherwise

Must pass all units in {BSL165, BUS130, BUS170, BUS171, BUS176} &

Must pass all units in {BUS285, BUS286, BUS287} &

Must pass all units in {BSL305, BUS304, BUS356} &

Must pass 2 unit(s) in unit set BREADTHUNI

MJ-BLW (BUSINESS LAW MAJOR) RULES

For COURSE_CODE IN {B1325, H1249} Do

Must pass all units in {BSL165, BUS130, BUS170, BUS176} &

Must pass all units in {BSL201, BSL202, BSL203} &

Must pass all units in {BSL305, BUS303} &

Must pass 1 unit(s) in {BSL312, BSL391, LEG291} &

Must pass 1 unit(s) in unit set BREADTHUNI

Otherwise

Must pass all units in {BSL165, BUS130, BUS170, BUS176} &

Must pass all units in {BSL201, BSL202, BSL203} &

Must pass all units in {BSL305, BUS303} &

Must pass 1 unit(s) in {BSL312, BSL391, LEG291} &

Must pass 2 unit(s) in unit set BREADTHUNI

BREADTHUNI (LIST OF UNIVERSITY WIDE BREADTH UNITS)

{AIS204, AIS308, ANS337, ART102, ART325, AST288, BRD201, BRD202, BRD203, BRD204, BRD205, BRD206, BRD207, BRD208, BRD209, BRD210, BRD211, BRD250, BRD251, BRD306, BRD311, BRD350, BUS358, BUS3581, BUS3582, CMS306, COD303, CRM100, CRM310, ENV245, ICR101, ICR201, IND101, IND102, JPN101, JPN102, LLB301, LLB311, LLB312, LLB315, LLB325, LLB327, LLB337, LLB342, LLB376, LLB386, LLB387, LLB390, LLB393, MSP100, MSP200, MSP201, OAIS204, OASN288, OBRD202, OBRD209, OBRD210, OBRD251, OBRD350, OCOD303, OENV245, OPHL209, OPOL226, OREL101, PHL209, PHO312, PHO330, POL226, POL340, PRO107, PRO325, PSY380, REL101, REL102, SCR304, SOU101, SOU214, SOU268, SOU325, SOU378, SUS309, SUS331, TOU306, TOU323, WEB301, WEB303, WEB325}

Appendix C: Glossary of Terms

Glossary	
String	A sequence of characters
CourseLoop	Curriculum management system
Callista	Student management system
Node.js	Open-source JavaScript runtime environment that executes JavaScript code outside of a browser
Parse	Turning a set of data to another set of data