Webservices to Support Departmental Dashboards

CSC 2310 – Spring 2022

Abstract

Deans, Associate Deans, Department chairs and others across the university are constantly in need of the data necessary to help drive decision making. In order to use that data, effective visualization is needed to facilitate interpretation of the vast amount of information available to these administrators. In this project we will create a number of webservices suitable for supporting the creation of dashboards that can be used to access valuable institutional data.

Data is critical for making both short-term and long-term decisions. Data scientists are often called upon to create tools for providing insight into the knowledge embedded in data by fusing information from various sources into a single cogent collection of information. For instance, the following table shows basic information related to courses offered by the CS Department:

	CRN	Subject	Course	Section	Credits	Time	Where	Instructor	Enrollment	Title
1	12038	CSC	1200	001	3	1500pm - 1615pm TR	BRUN 327		25/25	Principles of Computing
2	10625	CSC	1200	002	3	0800am - 0850am MWF	BRUN 228		28/40	Principles of Computing
3	13519	CSC	1200	600	3	1500pm - 1615pm TR	BRUN 327		3/3	Principles of Computing
4	11360	CSC	1300	002	4	1600pm - 1650pm MWF	BRUN 119		43/75	Intro/Prob Solving-Comp Prog
5	11050	CSC	1300	101	0	1300pm - 1350pm M	BRUN 209		31/26	Intro/Prob Solving-Comp Prog
6	12121	CSC	1300	102	0	1300pm - 1350pm W	BRUN 209		33/32	Intro/Prob Solving-Comp Prog
7	10566	CSC	1300	104	0	0800am - 0915am T	BRUN 209		32/32	Intro/Prob Solving-Comp Prog
8	11682	CSC	1310	001	4	0800am - 0915am TR	BRUN 119		70/60	Data Structures and Algorithms
9	10830	CSC	1310	002	4	1330pm - 1445pm TR	BRUN 119		71/60	Data Structures and Algorithms
10	11467	CSC	1310	101	0	1300pm - 1350pm M	BRUN 210		24/25	Data Structures and Algorithms
11	11785	CSC	1310	102	0	1300pm - 1350pm W	BRUN 210		23/25	Data Structures and Algorithms
12	10324	CSC	1310	103	0	0930am - 1045am T	BRUN 210		26/25	Data Structures and Algorithms
13	11034	CSC	1310	104	0	0930am - 1045am R	BRUN 210		24/25	Data Structures and Algorithms
14	11834	CSC	1310	105	0	1630pm - 1745pm T	BRUN 210		23/25	Data Structures and Algorithms
15	10454	CSC	1310	106	4	1630pm - 1745pm R	BRUN 210		21/25	Data Structures and Algorithms
16	11355	CSC	2400	002	3	1430pm - 1520pm MWF	BRUN 228		27/30	Design of Algorithms
17	10238	CSC	2400	003	3	0800am - 0850am MWF	BRUN 327		18/25	Design of Algorithms
18	10911	CSC	2500	002	1	0930am - 1045am R	BRUN 410		9/16	Unix Laboratory

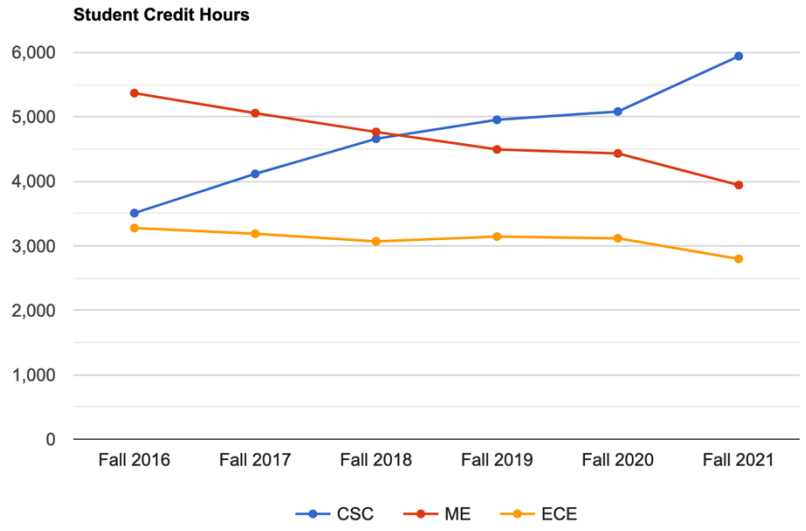
The knowledge embedded in this table includes both explicit and implicit information such as:

- (Explicit) Day, time, and room for a course
- (Explicit) Enrollment for a course
- (Explicit) List of courses taught by a specific faculty member
- (Implicit) Total student credit hours generated for the course (i.e., credits * enrollment)
- (Implicit) Total student credit hours generated by a department
- (Implicit) Total room utilization of a location of campus
- (Implicit) List of rooms available at any given moment

Information such as that found in the table, when supplemented with other data (such as the capacity of a given room on campus, list of colleges and the departments in each, etc.) can be valuable when looking to solve various problems, identifying which degree programs are growing/retracting, identifying which building spaces are being used efficiently, etc.

The purpose of this project is to identify the needs of administrators and to develop the webservices needed to support the creation of a dashboard of widgets that can be used to visualize information related to enrollment, course offerings, building utilization, etc. For instance, by using course

enrollment data, a graph showing historical trends of student credit hour generation can be displayed. Such a graph can be varied by displaying information for a specific college (such as the College of Engineering), by selecting specific departments, selecting different date ranges, and so forth.



In this iteration, you will be doing the following:

- Identify all of the potential user types for this system
- Brainstorm the potential features of the system – for instance, some users might want to add information, others may want to view specific course information (by clicking on a row?)
- Specify user stories for each of the user types based on your problem analysis using the user story format of "As a <user>, I want <feature>, so that <reason>".

As you begin the brainstorming process, recall that as a "developer" you have biases, and that you need to try adopt the mindset of different kinds of users. You may want to ask yourself questions such as:

- What other information should we display? What does each type of user need to see and do?
- Should a user be able to add or modify data? Which type of users can do this?
- What should a user be able to learn from the display of the data?

As you go through this activity, you should attempt, as much as possible, to avoid thinking about how any of this is going to be implemented. The most important aspect of this activity is identifying all of the *possible* features for this system (noting that identification of all of the possible features does not mean you will have to implement all of the features).

Project Phases

The project will be conducted in two major phases: *concept-initiate* and *iteration-construction*. In the *concept-initiate* phase you will identify project requirements, create models, and prep your development environment to support your implementation activities. In the *iteration-construction* phase of the project you will iteratively develop software for the project by adding support

according to the requirements developed during the *concept-initiate* phase. The schedule of the phases of the project and the individual iterations contained therein are defined below:

Date (Midnight)	Phase	Description
February 4	Concept Initiate 0	User stories
February 18	Concept Initiate 1	Use case diagrams
March 4	Concept Initiate 2	Initial Design
March 25	Iteration 0	Project environment setup and initial execution demonstration
April 8	Iteration 1	Tier 1 services
April 22	Iteration 2	Tier 2 services (collections)
April 29	Iteration 3	System Integration

Concept Initiate 0

User Stories

In the initial phase of the project, you will create user stories based on the description provided while also generating ideas of other potential features for such a system. The user stories that you create should be specified in the following form:

As a <user type>, I want <desired feature>, so that <outcome>

You must also include, with each user story, a description of how you would expect to be able to observe this behavior in the system. Your source of information for the user stories you will identify are given in the description provided above as well as any discussion sessions we conduct in class. You may also reach out to faculty or other students to generate ideas on potential stories. **However, the work you turn in must be your own. You may not copy other students' user stories, descriptions, or other similar work products.**

A template for your user stories is provided below.

User Story Template

Use the following user story template for *concept-initiate 0*. Replicate this as many times as is necessary.

User Story Name: <Short phrase for the user story>	
As a:	<user type>
I want:	<feature>
So that:	<outcome or reason>
Description	<description of how the user story / feature would be observed and tested in the system>

Submission

Submit your user stories as a PDF document only. *Microsoft Word, LibreOffice, or any other native word processing format will not be accepted.*

Rubric

Completeness: You will be graded on the completeness of your turn-in. In particular, you must analyze the description and attempt to identify as many features as you can that are relevant for

the project. Note, there is no upper bound on the number of user stories you can define, but there is a lower bound that *will not* be specified.

Correctness: You will be graded based on your adherence to the format of the user story template and on the accuracy of the user stories with respect to relevance to the described project, including your description of how you might expect to observe the user story / feature in the system.

Points: The assignment is worth 20 project points.

Concept Initiate 1

Use Case Diagrams

In our initial step on this project, we did some work to try to identify user stories for the *Dashboard* system. A baseline set of 15 user stories has been uploaded to the course iLearn site and should be used as a starting point for completing the next activity in the project. You must add additional user stories from your own set produced from the set you created for Concept Initiate 0. For Concept Initiate 1, you will be creating use case diagrams for the *Dashboard* system. The baseline user stories fall into a few categories:

- *Course Display – User stories that provide analytics on courses in the current university schedule*
- *Departmental / College Metrics – User stories that show metrics for credit hours generated by individuals, departments, and colleges*
- *Building Information – User stories that*

As you did in Laboratory 2, you will use GenMyModel to create your use case diagram(s) for the Dashboard system. The user stories that you will use for creating the use cases in the diagram are included as an attachment to the assignment in iLearn. Some things to note about these user stories:

- There are 15 base user stories specified in the reference set across 6 different user types in the baseline set. You must add your own additional user stories (at least 5-10)
- User stories may differ by users with the same feature. These should be represented by different users in the diagram linked to a common use case.
- The use case diagram should be partitioned according to the feature categories specified above. Additional stories may result in additional categories that you should name, if appropriate.

Submission

For your turn-in, you should export the model to an image and copy and paste the image to word or some other word processing suite and generate a PDF.

Rubric

Completeness: You will be graded on the completeness of your turn-in. In particular, your submission must at the very least include the user stories provided in the baseline set. However, it is also expected that you will include other stories in your diagram.

Correctness: You will be graded on the correctness of your use case diagrams. In particular, your submission must correctly represent use cases, actors (both interactive and external actors), and partitioning of the stories into appropriate subsystems by category.

Points: The assignment is worth 20 project points.

User Story Name: course list	
Category: course	
As a:	chair
I want:	a list of courses offered
So that:	I can review what the department is offering in a given semester
Description	Simple display of courses with typical information (credits, enrollment, time, room, etc.)

User Story Name: total credit hours	
Category: metrics	
As a:	chair
I want:	a summary of total credit hours generated for the department
So that:	I have a sense of the productivity of the unit
Description	Credit hours generated = credits * enrollment

User Story Name: CH per faculty	
Category: metrics	
As a:	chair
I want:	credit hours per faculty member
So that:	I know the relative productivity of faculty in the department
Description	subdivided credit hours by faculty

User Story Name: courses by faculty	
Category: metrics	
As a:	chair
I want:	a list of courses taught by faculty
So that:	so that I can review the teaching list for an individual
Description	Same information as the data shown at the department level

User Story Name: CH per department	
Category: metrics	
As a:	dean
I want:	a summary of the credit hours generated for the college
So that:	I know how productive my division is
Description	A display of the credit hours for a given college; can also be a summary of each department within the college as well

User Story Name: CH per college	
Category: metrics	
As a:	dean
I want:	a global display of credit hours by college
So that:	I can see how my college compares to other colleges
Description	Colleges include Business, Engineering, Education, Nursing, Interdisciplinary Studies, Agriculture and Human Ecology, Fine Arts, Arts and Science

User Story Name: CH by year	
Category: metrics	
As a:	chair
I want:	a summary of credit hours generated by year
So that:	I can see how the productivity of the department over a long period of time
Description	Related to the user story of credit hours generated

User Story Name: Room utilization	
Category: building	
As a:	(n) administrator
I want:	to know room usage statistics
So that:	I can see how efficiently rooms are being used
Description	This helps with identifying where to put resources for renovation

User Story Name: Room availability	
Category: building	
As a:	room scheduler
I want:	a list of rooms that are available during a given time
So that:	I can schedule courses or events during that time
Description	Room availability provides information about what is open across campus

User Story Name: student schedule	
Category: course	
As a:	student
I want:	to get a list of my courses
So that:	know my general availability
Description	Can use this to communicate my schedule to others

User Story Name: line chart for CH	
Category: metrics	
As a:	chair
I want:	a line chart of credit hours by year
So that:	I can visually view the performance of the department
Description	A visual analytical view of information from a previous story

User Story Name: line chart by dept	
Category: metrics	
As a:	dean
I want:	a line chart of credit hours for all department in my college
So that:	I can visually view the performance of all departments in my college
Description	visual version of the data from a different user story

User Story Name: faculty productivity

Category: metrics	
As a:	dean
I want:	to view a per faculty measure of credit hours generated for a department
So that:	I can compare relative performance of different departments in my college
Description	Scaled version of department metrics to understand total workload of faculty by department

User Story Name: faculty schedule	
Category: course	
As a:	faculty member
I want:	a list of courses I am teaching
So that:	I can use the data in my annual report
Description	Provides a list of all my courses along with credit hours generated

User Story Name: room statistics	
Category: course	
As a:	faculty member
I want:	classroom statistics for rooms
So that:	I can choose appropriately sized rooms for my courses
Description	Faculty request to change rooms from time to time

Concept Initiate 2

Class Diagrams

The *Dashboard* system is a web-based application that is meant to provide information about courses, faculty loads, building usage, student credit hours, and other similar data. The primary data source for the system is provided by an API endpoint that returns information similar to the following:


```
 {
  "CRN": "10803",
  "DEPARTMENT": "CSC",
  "COURSE": "2310",
  "SECTION": "001",
  "CREDITS": 4,
  "ISTIMEDETERMINED": "true",
  "STARTTIME": "1600",
  "STARTAM_PM": "pm",
  "ENDTIME": "1650",
  "ENDAM_PM": "pm",
  "CLASSDAYS": "MWF",
  "ISLOCDETERMINED": "true",
  "BUILDING": "BRUN",
  "ROOM": "228",
  "ISONLINE": "false",
  "PROF": "Gannod, Gerald C",
  "STUDENTCOUNT": 40,
  "MAXIMUMSTUDENTS": 42,
  "ISOPEN": "true",
  "TITLE": "Object-Oriented Prgrming/Dsgn"
 }
```


In this iteration, you are to create a class diagram that identifies all of the classes, their relationships (e.g., inheritance, composition, aggregation, association), the multiplicities of the relationships (e.g., 1-1, 1..*, etc.) and potential attributes and (non-constructor, non-getter, and non-setter) methods.

The data is available for any given semester starting from the current semester backwards into the past. From this data there is much that can be derived about courses, faculty that teach in a department, enrollment in a given course, courses taught in a building, the rooms contained in a building, and more. Just from the data provided above, there are a number of explicit relationships that can be derived between potential classes in the system, including:

- Courses (and course sections) in departments
- Faculty that teach in a given department
- Names and numbers of courses taught by a faculty member
- Buildings and classrooms contained in those buildings

There are also a number of derivable facts that are present in this data, including:

- The total number of student credit hours generated by a given course (computed by taking the product of STUDENTCOUNT and CREDITS)
- Total number of student credit hours generated by a department
- Total number of student credit hours generated by a faculty member
- Plus many more

Some information embedded in the data is either implicit or hidden, for instance:

- Section number and number of credits can be used to indicate whether a course is a lecture or a lab. Specifically, sections numbered 0xx, 50x, and 60x are standard lecture, online, or dual enrollment courses. Sections number 1xx are designated as laboratories.
- The MAXIMUMSTUDENTS measure, when compared against all of the courses that use a particular room, provides a hint regarding the maximum capacity of that room

Time series data can be derived by taking successive semesters of information and aggregating the information. As such, any information collected for a given year, may also yield interesting time series visualizations. For instance:

- How many total student credit hours were generated by a department over the past five years?
- How frequently were given courses taught in the past five years?
- What was the level of utilization of a given room over the past five years?

Finally, there is some data that is not contained explicitly or implicitly above but is relevant to the system, including:

- Which departments or subjects are contained in a given college?
- What are all the building codes?
- What are all the subject codes?
- What is the relative proximity of the buildings on campus?

The class diagram(s) that you will create are to focus on the modeling of the data rather than modeling the mechanics of the implementation for the system. There are several classes that have already been mentioned in this document (albeit implicitly) that should be modeled. (HINT: the data shown above alone identifies at least 5 classes).

To help simplify the modeling, it is suggested that you create modeling diagrams as follows:

- Relationships: Create a model that shows the classes and their relationships. Omit the use of attributes and methods on this diagram and focus only on the relationships
- Class information: Create one or more models that omit the relationships but show the attributes and methods of each class. This should include any parameters used for methods.

Data

Each of you has been provided an API key for accessing the API endpoint located at:
https://portapit.tntech.edu/express/api/unprotected/getCourseInfoByAPIKey.php
with the following parameters:

- Key: your specific API key. Do not share your key. It will expire at the end of the semester.
- Term: yyyySS, where yyyy is the year and SS is the semester. SS specified using the following codes:
    - 10 – Spring
    - 50 – Summer
    - 80 – Fall
- Subject: subject code. For instance, CSC. Using ALL will return

We will cover how to access the API endpoint programmatically in class. The data can, however, be accessed using a web browser, by using the following format:

https://*URL*?Key=*YOURAPIKEY*&Subject=*subjectcode*&Term=*termcode*

Note that in order to use this API endpoint you must either be on campus or use the campus VPN. Details on using the VPN can be found here:

https://its.tntech.edu/display/MON/Off-Campus+VPN+Installation

The data is delivered using the JavaScript Object Notation (JSON) which is a key-value pair representation of the data.

Submission

Use the https://app.genmymodel.com system to create your models. Export your model to a PNG and submit as a PDF to iLearn. Videos are available online on the different modeling notations for UML.

Rubric

Completeness: You will be graded on the completeness of your submission, most notably the inclusion of classes described above as well as on the identification of implicit and derived classes that arise from a thorough analysis of the problem.

Correctness: You will be graded on the correctness of the class specifications you have created including correct capture of the relationships between the classes and the specification of attributes and methods that arise from a thorough analysis of the problem.

Points: The assignment is worth 30 project points.

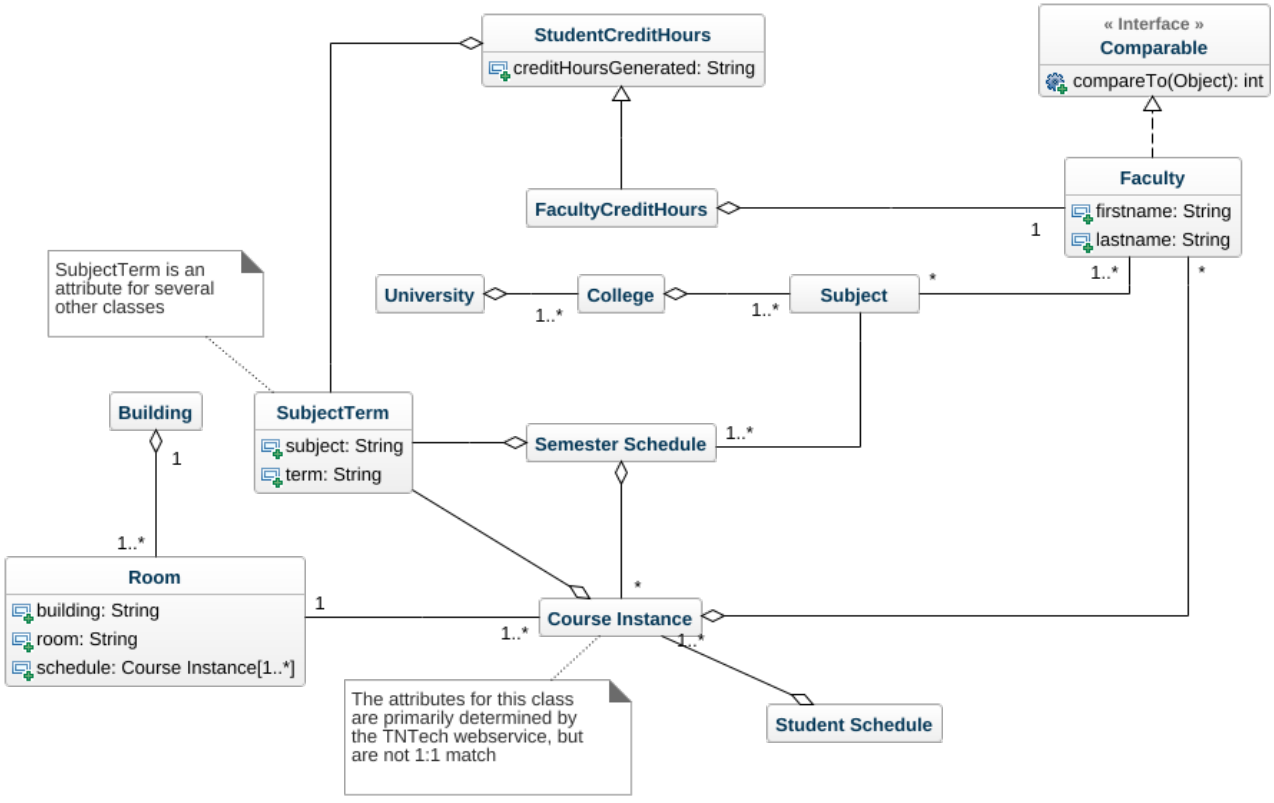
Iteration 0

Overview

Iteration 0 of a project is often devoted to setting up a development environment and performing "spikes" on use of a new technology. For Iteration 0 of the dashboard project you will be using the approach demonstrated in class (and captured in video) on how to create a RestController, and how to consume a web service using the Google GSON library.

Data Model

The reference data model for the project is shown below. The lecture from Monday, March 21st provides some details on the rationale for the model. In particular, note the use of the **SubjectTerm** class to provide context to data, and the **Faculty** object, which decomposes faculty names into firstname and lastname. For this iteration, your primary concern is the **CourseInstance** class without the **SubjectTerm** and **Faculty** composites, which you will add in during Iteration 1.



Data Source Web Service

Each of you has been provided an API key for accessing the API endpoint located at:
https://portapit.tntech.edu/express/api/unprotected/getCourseInfoByAPIKey.php
with the following parameters:

- Key: your specific API key. Do not share your key. It will expire at the end of the semester.

- Term: yyyySS, where yyyy is the year and SS is the semester. SS specified using the following codes:
  - 10 – Spring
  - 50 – Summer
  - 80 – Fall
- Subject: subject code (for instance, CSC). Omitting the "Subject=*xxx*" will return all courses.

Recall that the data can be accessed for testing purposes using a web browser by using the following format:

https://*URL*?Key=*YOURAPIKEY*&Subject=*subjectcode*&Term=*termcode*

Tutorials on Publishing and Consuming Web Services

- Creating a service: https://elearn.tntech.edu/d2l/le/content/8948309/viewContent/81235623/View
- Creating a client: https://elearn.tntech.edu/d2l/le/content/8948309/viewContent/81235640/View
- Building and Running with Maven: https://elearn.tntech.edu/d2l/le/content/8948309/viewContent/81239943/View

Required Elements
For this iteration, you are required to do the following:

- If you have not watched the videos above, do so first.
- Create a "Maven" project with dependencies to the **com.google.code.gson** and **junit** 4.13.2
- Use package **edu.tntech.csc2310.dashboard** for all of your classes
- Create a client class named **Spike** that acts as the driver for the application
  - This class should implement a **main** method and support execution using the following command-line command

    mvn exec:java -Dexec.mainClass=*classname* -Dexec.args="*args*"

    where *classname* is **edu.tntech.csc2310.dashboard**.**Spike**, and *args* is an argument list of subject, term, and course (i.e., CSC 202210 2310).
  - The output should be a list of courses that match the course number (for example, exec.args="CSC 202210 2310" will produce the following):

    CSC 2310-103 (Gannod, Gerald C) T 1630pm - 1745pm
    CSC 2310-102 (Gannod, Gerald C) R 1500pm - 1615pm
    CSC 2310-101 (Gannod, Gerald C) T 1500pm - 1615pm
    CSC 2310-001 (Gannod, Gerald C) MWF 1600pm - 1650pm
- Create a class named **ServiceBridge** as the primary class for accessing the course web service. This class should use the GSON library to call the service and populate an array of **CourseInstance** objects (see next bullet) by using the service url and your assigned key.

- The constructor for this class should be the following:

  public ServiceBridge(String term, String subject)

- The ServiceBridge class should provide a method that returns an array of CourseInstance objects with signature as follows:

  public CourseInstance[] getCourses()

- Note: Failure to implement the constructor and getCourses method as specified will impact the JUnit tests created to grade your solution.

- Create a class named **ServiceBridgeTest** and write the tests needed to verify that your ServiceBridge retrieves the data correctly.
- Create a class named **CourseInstance** to store the retrieved data: The attributes of the class must match the names (and case!) of the keys of the key-value pairs used by the service. Also, you may find it helpful to implement a toString method with the following signature:

  public String toString()

Submission

A git repository has been created for your project on gitlab.csc.tntech.edu upon which you may create your code. The Maven "pom.xml" file has not been configured for any dependencies other than Spring Boot, which will not be used in Iteration 0, but will be used in later iterations.

You can access your repo at:

https://gitlab.csc.tntech.edu/csc2310-sp22-students/*yourid*/*yourid*-dashboard_0.git

where *yourid* should be replaced with your specific userid.

Note that the repository also contains a data directory with example json files that will be used as reference in later iterations.

When you have completed your solution, commit with -m "Completed solution" and push to gitlab.

Rubric

Completeness: You will be graded based on completion of all of the required elements provided above.

- Project created as a Maven project
- Program is executable on the command-line using the stated maven command
- Package and class name adherence
- Classes implemented
- Tests implemented

Correctness:

- Program produces correct output

- JUnit tests created by teaching team pass
- Student supplied tests pass

Submission:

- Project correctly submitted via git to gitlab

Points: This assignment is worth 40 points.

Schedule Adjustment

The schedule has been adjusted to reflect a delay in starting Iteration 0. See the table below for details.

Date (Midnight)	Phase	Description
February 4	Concept Initiate 0	User stories
February 18	Concept Initiate 1	Use case diagrams
March 4	Concept Initiate 2	Initial Design
~~**March 25**~~ **April 1**	Iteration 0	Project environment setup and initial execution demonstration
April ~~8~~ 15	Iteration 1	Tier 1 services
April ~~22~~ 29	Iteration 2	Tier 2 services (collections)
~~**April 29**~~	~~Iteration 3~~	~~System Integration~~

Iteration 1

Overview

In the last iteration you performed a spike to understand how to consume data from a web service. In this iteration you will create several methods/services according to Table 1. You will create a class called *ServiceBridge* that implements each of the methods in the table similar to the implementation of Iteration 0. The details of what to implement can be found in the user stories from the Concept Initiate 1 phase.

Review

To prepare for completing the implementation, see the following video:

- Creating a service: https://elearn.tntech.edu/d2l/le/content/8948309/viewContent/81235623/View

Services and Methods

Table 1 is a summary of the services (and corresponding methods) that you will be creating. Each row shows the name of the method and the expected parameters in the columns. The final column of the table shows the expected output of the method. A row corresponds to a service url as follows (for the first row):

http://localhost:8080/allcourses?term=202210

Similarly, the third row for coursesbyfaculty would be called as follows:

http://localhost:8080/coursesbyfaculty?term=202210&subject=CSC&lastname=Gannod&firstname=Gerald C

When implementing the service methods, you must add the following tag before the definition of each method:

@GetMapping("/*servicename*")

where *servicename* is the name of the service from the table. For instance, the allcourses method will have the following definition

```
@GetMapping("/allcourses")
public SemesterSchedule allcourses(@RequestParam(value = "term", defaultValue = "na") String term)
{
  // implemented method
}
```

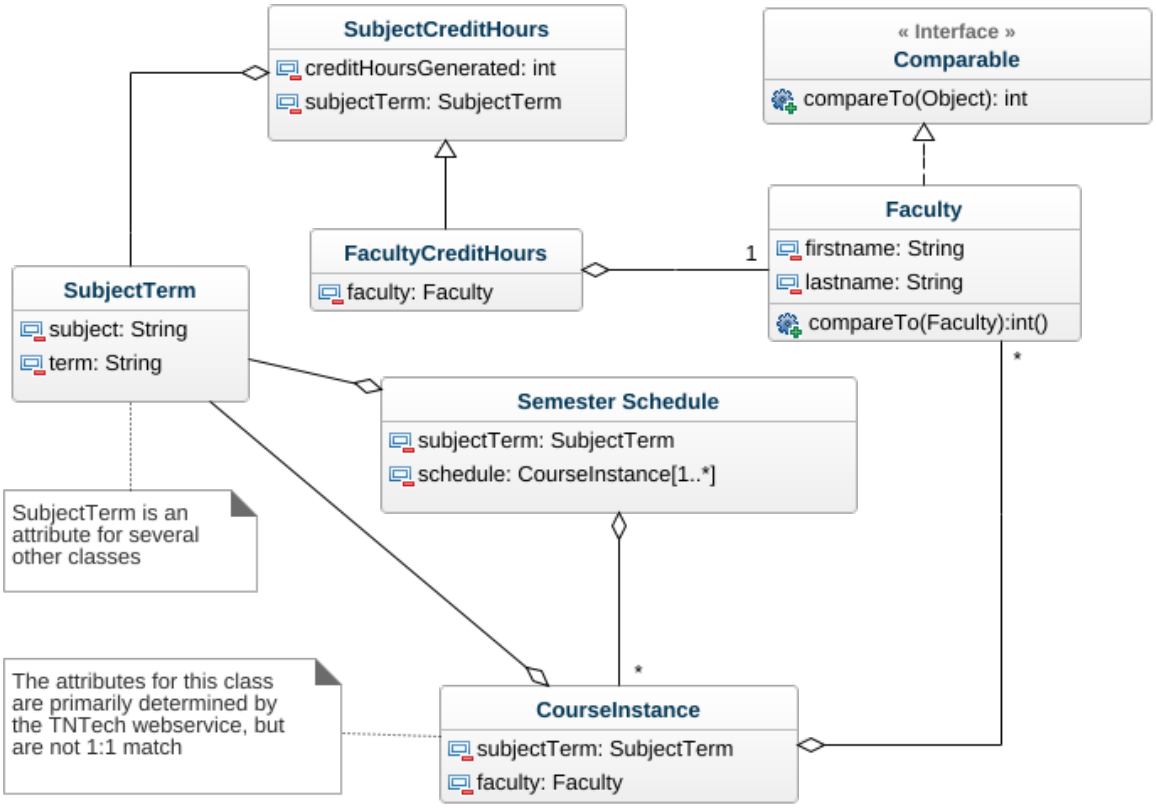
Note that the parameter term corresponds to the parameters for the service. Also, the default value for all parameters should be set to “na” to facilitate future error checking.

Table 1 Services

Service Name	Parameters												Return
	term	subject	course	section	lastname	firstname	beginterm	endterm	termlist	crnlist	building	room	
Iteration 1 Services													
allcourses	x												SemesterSchedule
coursesbysubject	x	x											SemesterSchedule
coursesbyfaculty	x	x			x	x							CourseInstance[]
coursebysection	x	x	x	x									CourseInstance
schbydepartment	x	x											SubjectCreditHours
schbyfaculty	x	x			x	x							FacultyCreditHours
Iteration 2 Services													
schbydeptandterms	x	x					x	x					SubjectCreditHours[]
schbydeptandtermlist	x	x							x				SubjectCreditHours[]
schbyfacultyandterms	x	x			x	x	x	x					FacultyCreditHours[]
schbyfacultyandtermlist	x	x			x	x			x				FacultyCreditHours[]
coursesbycrnlist	x									x			CourseInstance[]
facultybysubject	x	x											Faculty[]
getallsubjects	x												String[]

Output Classes

The model shown below contains all of the (6) classes that you will need to implement as outputs for the services. The classes should include getter methods for each of the attributes represented in the model. Note that the CourseInstance class also has several attributes based that are received from consuming the data from the TNTech course webservice.



Gitlab Repository

A gitlab repository has been created for you. This repository contains the solution to Iteration 0. If you wish to use your own solution you must replace the ServiceBridge.java class with your own version. The repository can be found here:

https://gitlab.csc.tntech.edu/csc2310-sp22-students/*yourid*/*yourid*-dashboard_1.git

Testing

You must create your own JUnit tests for the project. These tests should verify differences in lowercase/uppercase usage of the service, test for failed searches, and other anomalous behaviors (in addition the Happy Path cases).

A solution for Iteration 1 has been deployed at the following url. You must append the appropriate service name and parameters to the url as specified in the Services and Methods section of the writeup.

http://csclnx01.tntech.edu:8080

Your solution must comply with the outputs provided by the test service.

Rubric

Completeness: You must complete all of the following

- Implement all of the services specified in Table 1, adhering to the parameters and return values shown in the table.
- Tests for each of the services; these tests must cover both successful and unsuccessful cases
- Build: your project must be able to be built using the following sequence of commands on the command-line:
    - mvn clean
    - mvn compile
    - mvn package

Correctness: Your solution must correctly handle the following

- Correctly produce an output for each of the services specified in the table
- Adhere to the outputs of the services based on the structure provide in the model (as demonstrated by the service example provided above)
- Tests pass for each of the services
    - Instructor tests
    - Self implemented tests

Submission: Your solution must be correctly submitted to gitlab

Points: This assignment is worth 50 points

Due Date: This submission is due on Sunday, April 17, 2022 at 11:59pm.

Iteration 2

In this iteration you will be completing development of the web services for the dashboard system. This includes the following

Complete the services from Table 1

- Clone service OR use your own by changing the remote destination and writing over existing

To begin the iteration you can either clone the provided solution found at the following url:

https://gitlab.csc.tntech.edu/csc2310-sp22-students/yourid/yourid-dashboard_2.git

Alternatively, you can use your solution from Iteration 1 for Iteration 2 by doing the following (replacing `url` with the above url):

```
> git remote rm origin
> git remote add origin url
> git push --force -u origin master
```

A video has been created that demonstrates how to change the remote destination for your git repository.

Test Site

As with Iteration 1, a test site has been deployed containing an implementation of the services found in Table 1.

Comments and Documentation

In addition to writing the code for the dashboard services, you must document your code as well as the project. In particular, you are required to add javadoc style comments for each of the methods found in the ServiceBridge class. For example, you will find a sample of the comments needed below for the listed method:

```
/**
 *
 * @param subject write your comment for what this parameter is
 * @param term write your comment for what this parameter is
 * @param lastname write your comment for what this parameter is
 * @param firstname write your comment for what this parameter is
 * @return
 */
@GetMapping("/schbyfaculty")
public FacultyCreditHours creditHoursByFaculty(
        @RequestParam(value = "subject", defaultValue = "CSC") String subject,
        @RequestParam(value = "term", defaultValue = "202210") String term,
        @RequestParam(value = "lastname", defaultValue = "Gannod") String lastname,
        @RequestParam(value = "firstname", defaultValue = "Gerald C") String firstname
)
```

After commenting the code, use IntelliJ to generate the javadoc html files into a directory named "docs" in the home directory of the project.

You will also add a **README.md** file to your project, with the following contents:

- Project Description with link to the full assignment assignment writeup
- Directory descriptions
    - src/main
    - src/test
- Compilation instructions
    - Describe how to compile the program using Maven within IntelliJ
- Development instructions
    - Describe how to clone the project from the repo
    - Describe how to load the project into IntelliJ including how to configure the sdk

You should approach the writing of the README.md file from the perspective of communicating to a new developer how to continue your work on the project. The format of markdown language is described in the following cheatsheet:

https://stationinthemetro.com/wp-content/uploads/2013/04/Markdown_Cheat_Sheet_v1-1.pdf

Rubric

Completeness: You must complete all of the following

- Implement all of the remaining services specified in Table 1, adhering to the parameters and return values shown in the table.
- Tests for each of the services; these tests must cover both successful and unsuccessful cases
- Build: your project must be able to be built using the following sequence of commands on the command-line
- Comment the code: your ServiceBridge class must be documented using javadoc and files generated into the docs directory
- Create a README.md file: document the project in a README file

Correctness: Your solution must correctly handle the following

- Correctly produce an output for each of the services specified in the table
- Adhere to the outputs of the services based on the structure provide in the model (as demonstrated by the service example provided above)
- Tests pass for each of the services
    - Instructor tests
    - Self implemented tests
- Docs correctly generated

Submission: Your solution must be correctly submitted to gitlab

Points: This assignment is worth 60 points

Due Date: This submission is due on Saturday, April 30, 2022 at 11:59pm.