Seneca
SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

BTI425

Web Programming for Apps and Services

Notes	Weekly
Resources	Graded work
Policies	Standards
Professors & addendum	Code examples

BTI425 Assignment 2

Notice
The course delivery and its dates and times have been affected by the worldwide novel coronavirus crisis. The content on this website may change frequently, so please refresh your viewer when consuming or reviewing content.

The purpose or objective of the assignment is to create a substantial Angular app that interacts with a web API.

Read/skim all of this document before you begin work.

While you are doing the work, if a *specific task* is not clear, or it seems to require an unreasonable amount of time to complete, or it seems to require knowledge way beyond the content we've covered in the course, please don't hesitate to contact your professor.

> *You should NOT have to search for or locate resources "out there" in an effort to complete this work.*
> *The resources provided in this course - notes, linked content, code examples - provide sufficient coverage. Review them frequently.*
> *If you think that you will find "the answer" to this assignment somewhere "out there", you're wrong. Use the course resources as your shortcut.*

Due Date

Tuesday, April 7, 2020, at 11:00pm ET

Grade value: 25% of your final course grade

If you wish to submit the assignment before the due date and time, you can do that.

Overview and purpose

As noted above, the purpose or objective of the assignment is to create a web API and an Angular app that has good coverage of the topics in the course.

The app's purpose is to define or translate the English-language terminology we use in the computer programming courses in the School of SDDS, for use by all students, whether or not English is a student's first language.

We have so many students who are learning English and computer programming at the same time, and that is a challenging combination. The technical terminology is often unclear, complicated, and sometimes isn't explained well. For example, when we say "asynchronous", what does it clearly mean in English, as well as in a student's first language?

The general functionality of the user app will likely include:

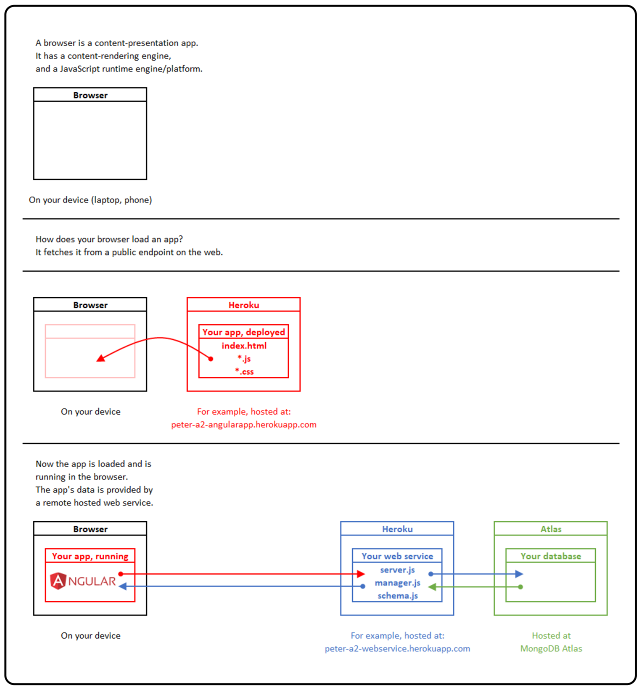
- View and browse a list of all terms
- Search for a term, in any language
- View details about a term, and its meaning in English and in other languages
- Add a new term, English, or in another language (including its first or initial definition)
- For an existing term (English or other), add another definition/data

In summary, the app would have some of the utility of a dictionary, Wikipedia, and other resources (including this course's notes and learning resources).

A web API will hold the data for the app. To enable scale and crowd-sourcing of data, the professors will also publish a web API that will be designed to aggregate (with some data entry *done by you*) the terms that each student enters, for the benefit of ALL students. More info about this is below.

The web API will be posted to Heroku and Atlas, and will provide the data for the Angular app. The Angular app will also be deployed to a public host (Heroku), so that you can deliver it to other devices (including, for example, your smartphone).

Here's a diagram that shows the relationships among your browser, the deployed Angular app, and the deployed web API. Right-click and open it in a new tab/window to view it full size.


A browser is a content-presentation app.
It has a content-rendering engine,
and a JavaScript runtime engine/platform.
Browser
On your device (laptop, phone)
How does your browser load an app?
It fetches it from a public endpoint on the web.
Browser
Heroku
Your app, deployed
index.html
*.js
*.css
On your device
For example, hosted at:
peter-a2-angularapp.herokuapp.com
Now the app is loaded and is
running in the browser.
The app's data is provided by
a remote hosted web service.
Browser
Your app, running
ANGULAR
Heroku
Your web service
server.js
manager.js
schema.js
Atlas
Your database
On your device
For example, hosted at:
peter-a2-webservice.herokuapp.com
Hosted at
MongoDB Atlas

Where this idea came from

Recently, in our School of SDDS, some of the web programming teachers have decided to use the course sequence (and particularly this course) as a way to create and deliver software that could be used by our School community (students and faculty) and improved over time (by students and faculty).

A number of app ideas were developed, and this idea was selected to be worked on in the Winter 2020 academic term.

Getting started, web API

Use `npm init` to initialize a new folder, probably named `a2-web-api`. Alternatively, use a code example from the repo as a base for your project work.

Data schemas

As implied above, data for the app will be generated by you (the student) as you work on the app. In addition, the professors will publish a web API that will aggregate and share the data generated by all students. More info about is found below.

For this Assignment 2, the database will include two collections:

- terms in English

    - includes an embedded subdocument collection, definitions in English

- terms in other languages
    - includes an embedded subdocument collection, definitions in other languages

As suggested by this list of collections, a *term* could have zero or more *definitions*. This idea supports alternative or evolving definitions over time. It is implemented as a one-to-many association or relationship that uses the MongoDB “embedded document” technique: In a English term document, it has a field that stores a collection (i.e. an array) of “definition” *subdocuments*.

> *Review the embedded subdocument coverage in the introduction note and in the how-to note.*

> *The embedded subdocument must be described by a Mongoose schema.*
> *Therefore, the web API will have three Mongoose schemas:*
>
> 1. *definition*
> 2. *terms in English (maybe named “termEnglish”)*
> 3. *terms in other languages (maybe named “termNonEnglish”)*

Each term collection will have the same data fields:

Assignment 2 “termEnglish” schema fields:

- wordEnglish
- wordNonEnglish
- wordExpanded
- languageCode
- image
- imageType
- audio
- audioType
- linkAuthoritative
- linkWikipedia
- linkYouTube
- authorName
- dateCreated (date)
- dateRevised (date)
- fieldOfStudy
- helpYes (integer)
- helpNo (integer)
- definitions (array of “definition” objects)

The “termNonEnglish” schema will be the same, and it will *also* have a field (probably named “termEnglishId”) that will be a document reference to the English term.

Assignment 2 “definition” schema fields:

- authorName
- dateCreated (date)
- definition
- quality (integer)
- likes (integer)

In addition, an item in the English collection will be associated or related, in a one-to-many manner, with zero or more items in the non-English collection. It is implemented using the MongoDB "document reference" technique: In a *non-English term document*, it has a field (probably named "termEnglishId") that stores an object identifier "reference" to the English term document.

> *Review the document reference coverage in the introduction note and in the how-to note.*

If you need a reminder about working with the MongoDB database engine, we suggest that you do these tasks:

- Create a database named `db-a2` (database for Assignment 2)
- In `db-a2`, create a collections likely named `TermsEnglish` and `TermsOther`

More explanations about some of the schema fields

The two "term" schemas are almost the same, by design. We wanted to make it easier to program by enabling copy-paste. Despite this, there are a few considerations which will affect your programming later on. Here's how some of the fields will be used.

- `wordEnglish`
  Required
  The term, in English
  This will be present (non-empty) in both kinds of objects, termEnglish and termNonEnglish
  It will be unique too

- `wordNonEnglish`
  The term, in another language
  This will be empty in termEnglish, and present in termNonEnglish

- `wordExpanded`
  If the wordEnglish is an acronym or initialism, this will have the expansion
  For example, JSON and "JavaScript Object Notation"
  Otherwise, it will be empty

- `languageCode`
  Required
  The ISO and *de facto* standardized codes for a language
  For example "en" for English, or "en-ca" for English (Canada)

For termEnglish items, it will be one of the language codes that begin with “en”
For termNonEnglish items, it will be one of the others
Below, there is a link to a downloadable file of language codes that will be supported in the app

- `image`, `imageType`, `audio`, and `audioType`
  We will use these in a future class/lesson
  For now, they can be empty
  They will hold file names (image and audio) for non-text media items for a term (e.g. a diagram or its pronounciation)
  (imageType and audioType are internet media type strings, e.g. “image/png”)

- `linkAuthoritative`, `linkWikipedia`, and `linkYouTube`
  URLs to web resources for the term
  Can be present or empty
  The “linkAuthoritative” is a link to the term’s authoritative info source
  For example, for Node.js, it’s https://nodejs.org

- `authorName`
  Required
  Your name

- `dateCreated` and `dateRevised`
  Required
  These values are set in program code
  Do NOT get them from the app’s user

- `fieldOfStudy`
  Probably “computer programming” for our terms
  The course professors foresee that this app can be used by learners in any field of study, and not just ours, so that’s its purpose - to identify the term’s field of study

- `helpYes` and `helpNo`
  There will be a user interface (UI) item that will enable these values to be incremented
  Obviously, with enablement by the web API
  The idea is that we could ask the user whether the term they were looking at was hepful or not (to their learning)

- `definitions`
  A collection of one or more “definition” documents
  It is always possible for someone to have an alternate or better definition than an existing one
  It’s also possible that a term (e.g. “server”) could have multiple and somewhat different definitions

- For a termNonEnglish document… `termEnglishId`
  Required (for a termNonEnglish document
  It is the object identifier of the termEnglish it’s related to

In the “definition” schema…

- `authorName`
  Required
  Your name

- `dateCreated`
  Required
  This value is set in program code
  Do NOT get it from the app’s user

- `definition`
  Required
  The actual possibly-lengthy *definition* or explanation of the term
  The text format will be HTML
  And, the plan is that the content will use the language’s character set

- `quality`
  Ignore this for now; for future use
  (A human and/or AI-ML curator could assign a quality rating value to the definition)

- `likes`
  There will be a user interface (UI) item that will enable this value to be incremented
  Obviously, with enablement by the web API
  The idea is that the user could quickly and easily indicate whether they “like” the specific definition

Data service tasks

The data service tasks are similar for each entity (termEnglish and termNonEnglish).

We suggest that you write the method pairs (in `server.js` and `manager.js`) for the termEnglish entity first, and thoroughly test them. Then, you will be able to essentially copy-paste-and-edit them for use for the termNonEnglish entity. The “add new” task will be slightly different in the termNonEnglish method pair, because *we must use the object identifier for the related termEnglish document*.

During the code-writing process, test incrementally and frequently with Postman.

It is expected that the following, at a minimum, will be needed for the **termEnglish** entity:

1. get all (sorted)
2. get one, by object identifier
3. get one (or some), by “wordEnglish”

4. add new (termEnglish document, including one definition embedded subdocument)
5. edit existing (termEnglish document), to add a *new* definition
6. edit existing (termEnglish document), to increment the "helpYes" value
7. edit existing (termEnglish document), to increment the "helpNo" value
8. edit existing (definition document), to increment the "likes" value

> *For guidance about how to handle "get some" for number 3, read this:*
> *Web API CRUD - "get some" technique*

> *Reminder about the associated/related data coverage and how-to notes.*
> *The `webapi-data-assoc-embed-doc` code example shows how to approach the coding task for numbers 4 and 5.*

> *For guidance about how to handle the "increment" tasks for numbers 6, 7, and 8, read this:*
> *Web API - "command" technique*

It is expected that the following, at a minimum, will be needed for the **termNonEnglish** entity:

1. get all (sorted) (it's possible that the app won't need to use this)
2. get one, by object identifier
3. get one (or some), by "wordNonEnglish"
4. add new (termNonEnglish document, including one definition embedded subdocument)
5. edit existing (termNonEnglish document), to add a *new* definition
6. edit existing (termNonEnglish document), to increment the "helpYes" value
7. edit existing (termNonEnglish document), to increment the "helpNo" value
8. edit existing (definition document), to increment the "likes" value

> *Reminder about the associated/related data coverage and how-to notes.*
> *The `webapi-data-assoc-doc-ref` code example shows how to approach the document reference coding task for number 4.*

> *FYI, that code example also shows how to approach the coding task to fetch a termEnglish document which includes its associated/related termNonEnglish documents.*

> *For guidance about how to handle the "increment" tasks for numbers 6, 7, and 8, read this:*
> *Web API - "command" technique*

Get large amounts of data (eventually!)

As noted above, the professor's version of the web API has been published here:

Professor Assignment 2 web API

Here's the plain text URL for the professor's version:
`https://pam-2020-a2and3webapi.herokuapp.com/api`

Using Postman, you can add these segments to the URL, and interact with the web API. Replace “XXXXXX…” with a MongoDB object identifier:

- `/terms/english`
  All terms in English
- `/terms/english/XXXXXX...`
  One term, in English, for a specific identifier
- `/terms/other`
  All terms in other languages
- `/terms/other/XXXXXX...`
  One non-English term, for a specific identifier

In addition, you can add a new English term (with its initial definition), using this URL:

- `/terms/english`
  Obviously POST, application/json, and a JSON entity body

And, for an *existing* English term, you can add another definition, using this URL:

- `/terms/english/XXXXXX.../add-definition`
  PUT, application/json, and a JSON entity body

As you would expect, you can add a new non-English term (with its initial definition), using this URL:

- `/terms/other`
  POST, application/json, and a JSON entity body

For an *existing* non-English term, you can add another definition, using this URL:

- `/terms/other/XXXXXX.../add-definition`
  PUT, application/json, and a JSON entity body

To increment a term’s “helpYes” counter:

- `/terms/english/helpyes/XXXXXX...`
  PUT, application/json, and a JSON entity body that looks like…
  `{ "_id": "XXXXXX..." }`
- Or, begin the URL with `/terms/other...` for non-English

To increment a term’s “helpNo” counter:

- `/terms/english/helpno/XXXXXX...`
  PUT, application/json, and a JSON entity body that looks like…
  `{ "_id": "XXXXXX..." }`
- Or, begin the URL with `/terms/other...` for non-English

Finally, to increment a definition's "likes" counter:

- `/terms/english/definition-like/XXXXXX...`
  PUT, application/json, and a JSON entity body that looks like…
  `{ "_id": "XXXXXX..." }`
  (this is the object identifier for the *definition*, not the *term*)
- Or, begin the URL with `/terms/other...` for non-English

To help yourself, and your fellow students, plan on using Postman and adding at least one English term and one non-English term to the professor web API. We also suggest that you add another definition to a term (one English and one non-English).

> *Data entry tips:*
>
> - *Send a "get one" request to get the JSON for one term*
> - *That will give you a model that you can use to copy, paste, then edit*
> - *Do not send `_id` values - MongoDB will generate the identifier*
> - *As an example, here is the JSON that the professor used to add the "asynchronous" English term - notice how the definition is packaged*
> - *If you want to send a double-quote character in a string, it must be "escaped" with a backslash - for example, `\"hello`*
> - *When you are entering a non-English term, use one of the official language codes (e.g. "fr-ca" or "ru" etc.; you can fetch that collection any time)*
> - *As an example, here is the JSON that the professor used to add the "asynchronisme" French term - notice how the definition is packaged*
> - *If the response is "bad request", as text/html, with HTTP status 400, then the JSON in the request is malformed - make sure to use the Postman "Beautify" button/feature to check your JSON for correctness*

The intention is that each student will be able to get access to a substantial collection of data, because the efforts of all 100 students in the course, each creating at least two terms, should result in about 200 sharable terms. How?

The best feature of the professor web API is the ability for YOU to DOWNLOAD the data as import-ready JSON, and then use MongoDB Compass to "import" the data into YOUR database. Open a browser, and use these URLs to download each collection:

- `/terms/english/download`
  All English terms, downloaded to your computer

- `/terms/other/download`
  All non-English terms, downloaded to your computer

After you have these downloaded files, you can easily import them into your own database collections.

> *What if the import tasks does not work?*
> *Please CONTACT YOUR PROFESSOR, because it should work.*
>
> *What if some fields are missing when you query the data with YOUR web API code?*
> *It's likely a property name or type mismatch - carefully compare your schema with the schema for the published data.*

Getting started, Angular app

Getting started includes generating a new project, and configuring your development environment.

> *You can create a new project with `ng new...`*
> *Or, you can use a template from the code example repo folder "Templates and solutions"*

Make sure that your web API has been completed, has some data, and is deployed to Heroku and MongoDB Atlas. Make sure that you can interact with it correctly with Postman. This is important, because you must have confidence in the hosted app to make progress on the Angular app.

Set up the rest of your dev environment (terminal windows, editor, browsers and tools).

The professor's sample solution is posted online:
Sample solution

> *Please note that you do NOT have to replicate its appearance and task flow.*
> *We are expecting your app to meet the specs below.*
> *We are NOT expecting your app to look and work exactly the same.*

Suggestion - use your debug tools

As you make progress on the Angular app, feature by feature, make sure you watch the command line that started the app locally (i.e. by using the `ng server -o` command). If there is a compile error, as you incrementally code, it will show up there.

Also, please remember the other debug strategies that you have been exposed to.

Suggestion - deploy to Heroku often

As you make progress on the Angular app, feature by feature, deploy it to Heroku, and maybe use another device (your smartphone?) to connect to it and interact with it.

How-to instructions for Angular app to Heroku are here:
Deploy Angular app to Heroku

Prepare to use data

One of the important initial tasks is to prepare the app to use data:

1. Define/write data classes (as TypeScript classes)
2. Write methods in the data manager service that send requests to your web API

In the app's `data-classes.ts` source code file, define/write TypeScript classes that match the shape of the web API schema classes. A few other recommendations:

- Set a reasonable initial value for each property; do this inline, for example:
` firstName: string = 'Peter';`

- For date fields, we can use a `string` type, and use the JavaScript `Date` object's `toISOString()` method whenever we must convert a date into a string for use with JSON.

- Use a constructor and some code to set the initial value for the date fields to the current date-and-time (we do NOT want to gather that from the user).

- In both of the "term…" classes, define an optional `_id` property. This will enable us to omit the value when we're doing an "add new" task, but it will enable receiving the value in responses that should have it.

In the app's `data-manager.service.ts` source code file, we suggest that you define/write methods incrementally as you code each component.

Doing the work, initial

The first big task is to design and plan the components needed and how the user will interact with the app. Here's an overview of how your professors thought about when making a sample solution:

- We'll need a "home" component
- It should either show a list of English terms, or include a link to a dedicated component
- Maybe it's a good idea to use the "home" component to hold the "search" functionality
- From the list of English terms, include an "add new" link, and for each item on the list, include a link to a detail component
- The detail component will show all of the term's data

    - And it will include its definitions
    - And translations (full text or link - to be decided)

- From there, it should be possible to “add” a new definition for the displayed term
- And it should be possible to “add” a new non-English translation for the displayed term
- When displaying a term, the UI should have a control that would enable the viewer/user to easily tap “yes this helped me” or “no this didn’t help me”
- When displaying a definition for a term, the UI should have a control that would enable the viewer/user to easily indicate that they “like” the definition

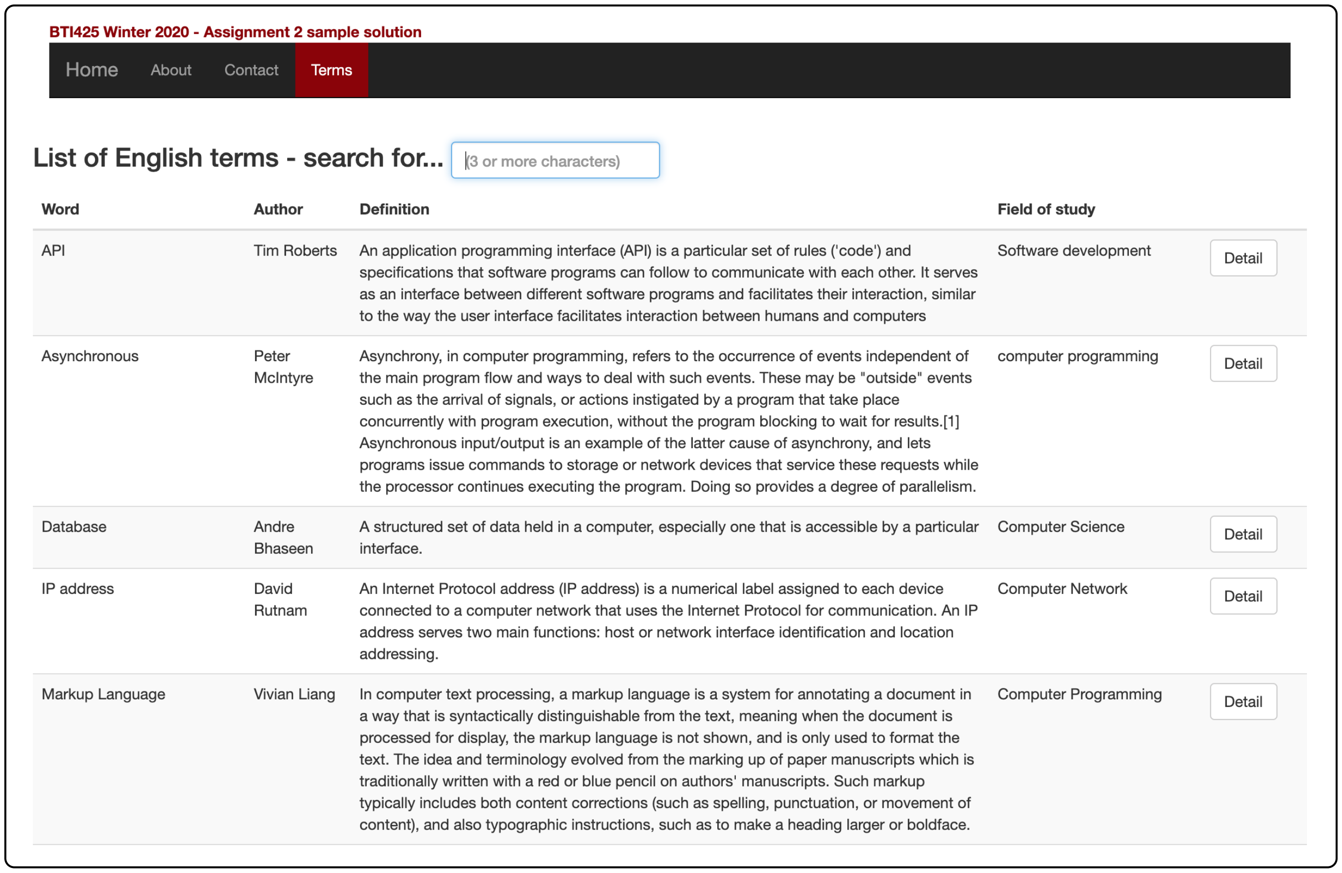
So, overall, maybe about six - plus or minus - components. Here is some commentary.

List of English terms

As noted above, the list of English terms can be on the home or landing component, or in a dedicated “list” component.

Decide which fields you want to display in the list.

In the data manager service, write a method that will send a request to the web API resource. Obviously, you must write classes (in the data classes source code file) that match the shape of the web API results (definition, English term, and non-English term).



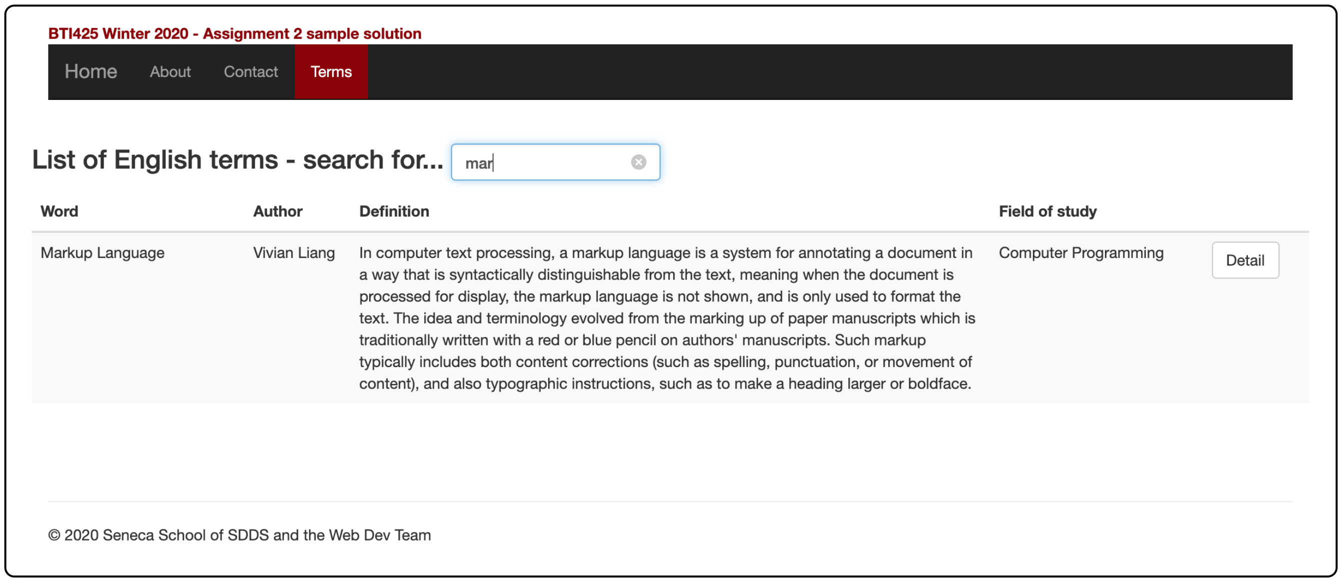
List of English terms - search for...

Word	Author	Definition	Field of study	
API	Tim Roberts	An application programming interface (API) is a particular set of rules ('code') and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers	Software development	Detail
Asynchronous	Peter McIntyre	Asynchrony, in computer programming, refers to the occurrence of events independent of the main program flow and ways to deal with such events. These may be "outside" events such as the arrival of signals, or actions instigated by a program that take place concurrently with program execution, without the program blocking to wait for results.[1] Asynchronous input/output is an example of the latter cause of asynchrony, and lets programs issue commands to storage or network devices that service these requests while the processor continues executing the program. Doing so provides a degree of parallelism.	computer programming	Detail
Database	Andre Bhaseen	A structured set of data held in a computer, especially one that is accessible by a particular interface.	Computer Science	Detail
IP address	David Rutnam	An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two main functions: host or network interface identification and location addressing.	Computer Network	Detail
Markup Language	Vivian Liang	In computer text processing, a markup language is a system for annotating a document in a way that is syntactically distinguishable from the text, meaning when the document is processed for display, the markup language is not shown, and is only used to format the text. The idea and terminology evolved from the marking up of paper manuscripts which is traditionally written with a red or blue pencil on authors' manuscripts. Such markup typically includes both content corrections (such as spelling, punctuation, or movement of content), and also typographic instructions, such as to make a heading larger or boldface.	Computer Programming	Detail

Search for an English term

Somewhere in the user interface (in an existing component or in a dedicated component), search (for an English term) should be supported. To learn one approach, read the Angular app "search" UI and UX document.

In the professor's sample solution, it was implemented as part of the list of English terms.


BTI425 Winter 2020 - Assignment 2 sample solution
Home
About
Contact
Terms
mar
Detail
© 2020 Seneca School of SDDS and the Web Dev Team

List of English terms - search for...

Word	Author	Definition	Field of study	
Markup Language	Vivian Liang	In computer text processing, a markup language is a system for annotating a document in a way that is syntactically distinguishable from the text, meaning when the document is processed for display, the markup language is not shown, and is only used to format the text. The idea and terminology evolved from the marking up of paper manuscripts which is traditionally written with a red or blue pencil on authors' manuscripts. Such markup typically includes both content corrections (such as spelling, punctuation, or movement of content), and also typographic instructions, such as to make a heading larger or boldface.	Computer Programming	Detail

Add new English term

This is a classic implementation of an "add new" use case. As you have learned, it needs:

- A component
- In the component template, controls are needed to gather data from the user
- In the component code, properties for the form data and other needs, and a method to handle form submission
- Also in the component code, a data class that matches the shape of the form data
- In the data manager service, it needs a method that sends the data to the web API

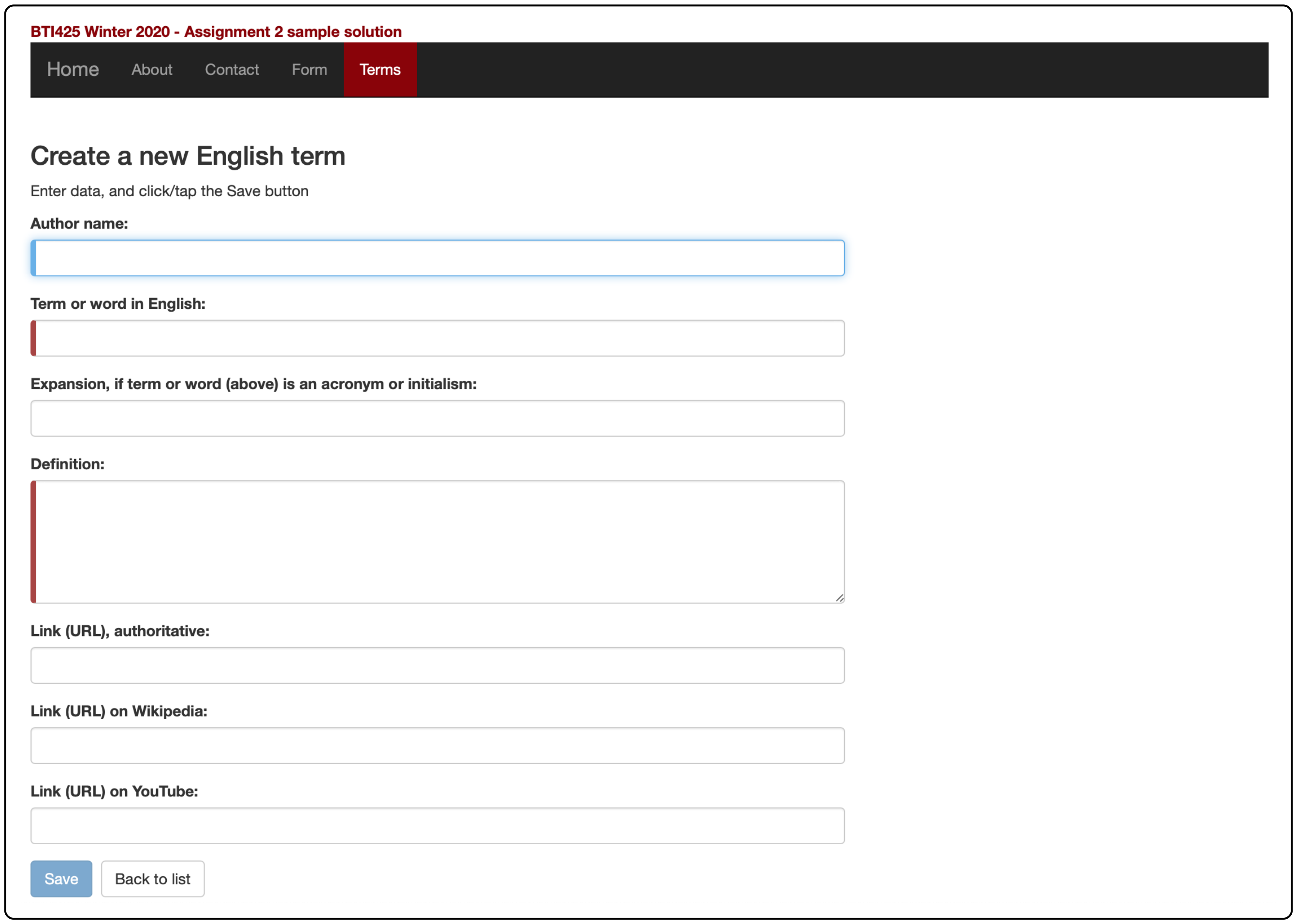
Some of the data items can be set or calculated in code. In other words, it is obviously *not necessary* to gather the "create date and time" from the user. Do that programmatically. For new English terms, we also will just set its language code value because we know it's English ("en").

Remember that you MUST gather a definition's text when you are creating a new English term. (As you know, additional definitions can be added later.)

Also remember the best practice that says you must create a *data model class* to define the data that will be entered on the form. Then, as you know, the shape of the data package that is sent to the web API "add new" request handler must match the "termEnglish" schema. This doesn't happen automatically, as you must write code in the form submit button handler method to do the data preparation. The links below are both a reminder about the process and provide additional how-to info that will help.

- Angular Forms Data Models
- JavaScript Spread and Object Mapping

After the “add new” completes successfully, remember to follow the PRG pattern, and redirect (probably to the “detail” component).


BTI425 Winter 2020 - Assignment 2 sample solution

Create a new English term

Enter data, and click/tap the Save button

Author name:

Term or word in English:

Expansion, if term or word (above) is an acronym or initialism:

Definition:

Link (URL), authoritative:

Link (URL) on Wikipedia:

Link (URL) on YouTube:

English term detail

This is a classic implementation of a “get one” use case.

Its visual UI design is somewhat important. For example, an English term could have one - or more - definitions. It would be nice to conveniently display, or have access to, the multiple definitions. Also, an English term could have zero - or more - translations. As above, it would be nice to display or link to them (if present).

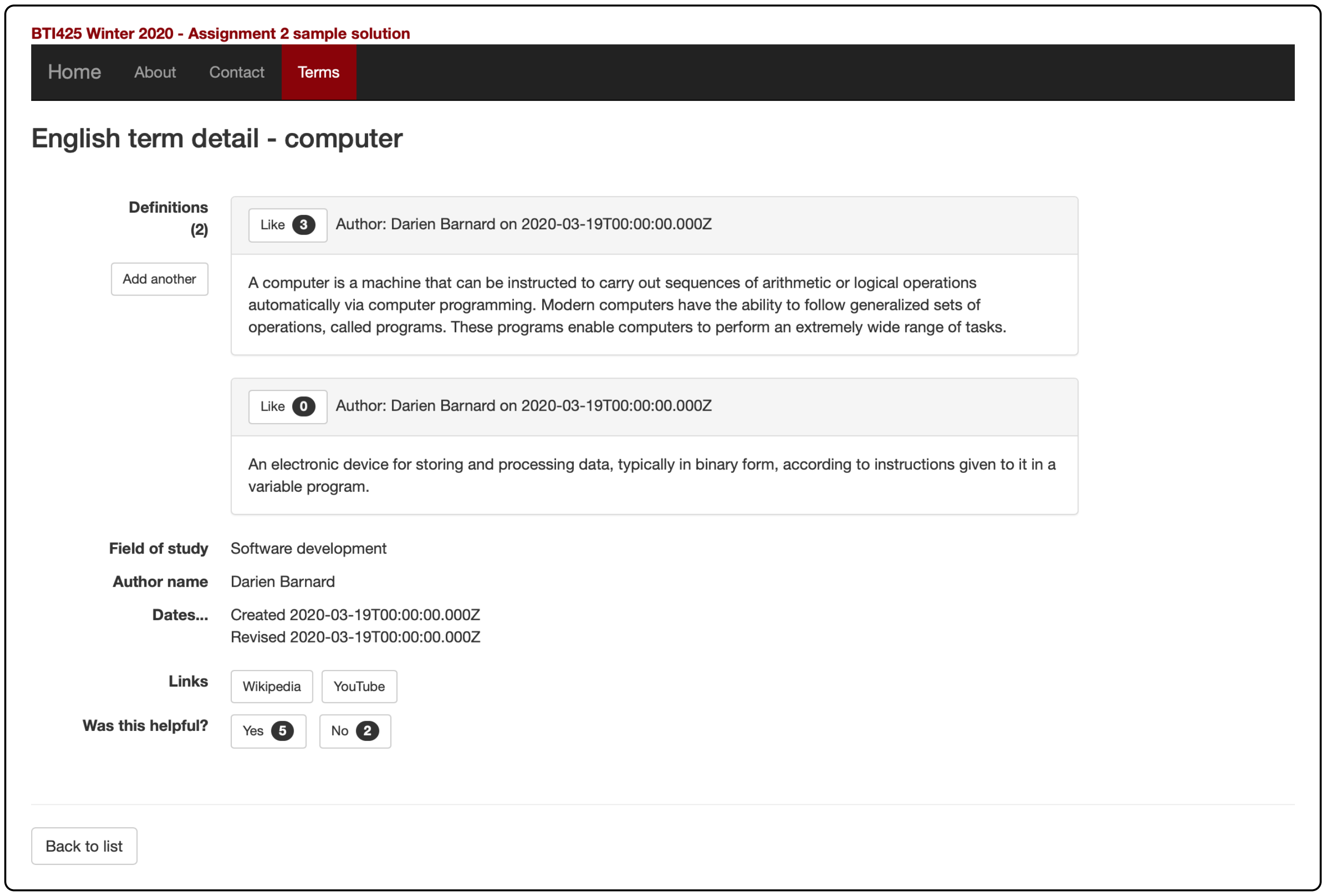
This means that we can tolerate different visual UI designs. One choice is to render everything in one component. Another choice is to render each translation separately in its own component.

Whatever is done, one of the tasks a user may want to do after viewing an English term and its definition(s) is to *add another definition*. Enable that capability in the UI.

Another task a user may want to do after viewing an English term and its definition(s) is to *add a translation* in another language. Enable that capability in the UI.

Another task a user may want to do is to “vote” on whether the term and its definitions were helpful. (That’s what the “help yes” and “help no” fields are for in the web API and database schema.) Gathering that info from the user can be done in many ways, but they all require an element that causes a method (in the component code) to run, which in turn calls a data manager service method, which sends a request to the appropriate web API resource.

Similarly, each definition should offer the ability to “like” it, again via an element > method > data manager service > web API request process.



English term detail - computer

Definitions (2) Add another	Like 3 Author: Darien Barnard on 2020-03-19T00:00:00.000Z A computer is a machine that can be instructed to carry out sequences of arithmetic or logical operations automatically via computer programming. Modern computers have the ability to follow generalized sets of operations, called programs. These programs enable computers to perform an extremely wide range of tasks.
	Like 0 Author: Darien Barnard on 2020-03-19T00:00:00.000Z An electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program.
Field of study	Software development
Author name	Darien Barnard
Dates...	Created 2020-03-19T00:00:00.000Z Revised 2020-03-19T00:00:00.000Z
Links	Wikipedia YouTube
Was this helpful?	Yes 5 No 2

Back to list

Add a new definition to an English term

Above, we identified a possible user task is to add another definition to an English term. This will be a classic “edit existing” (English term) use case, which will end up sending a request to the web API method that you wrote for that purpose.

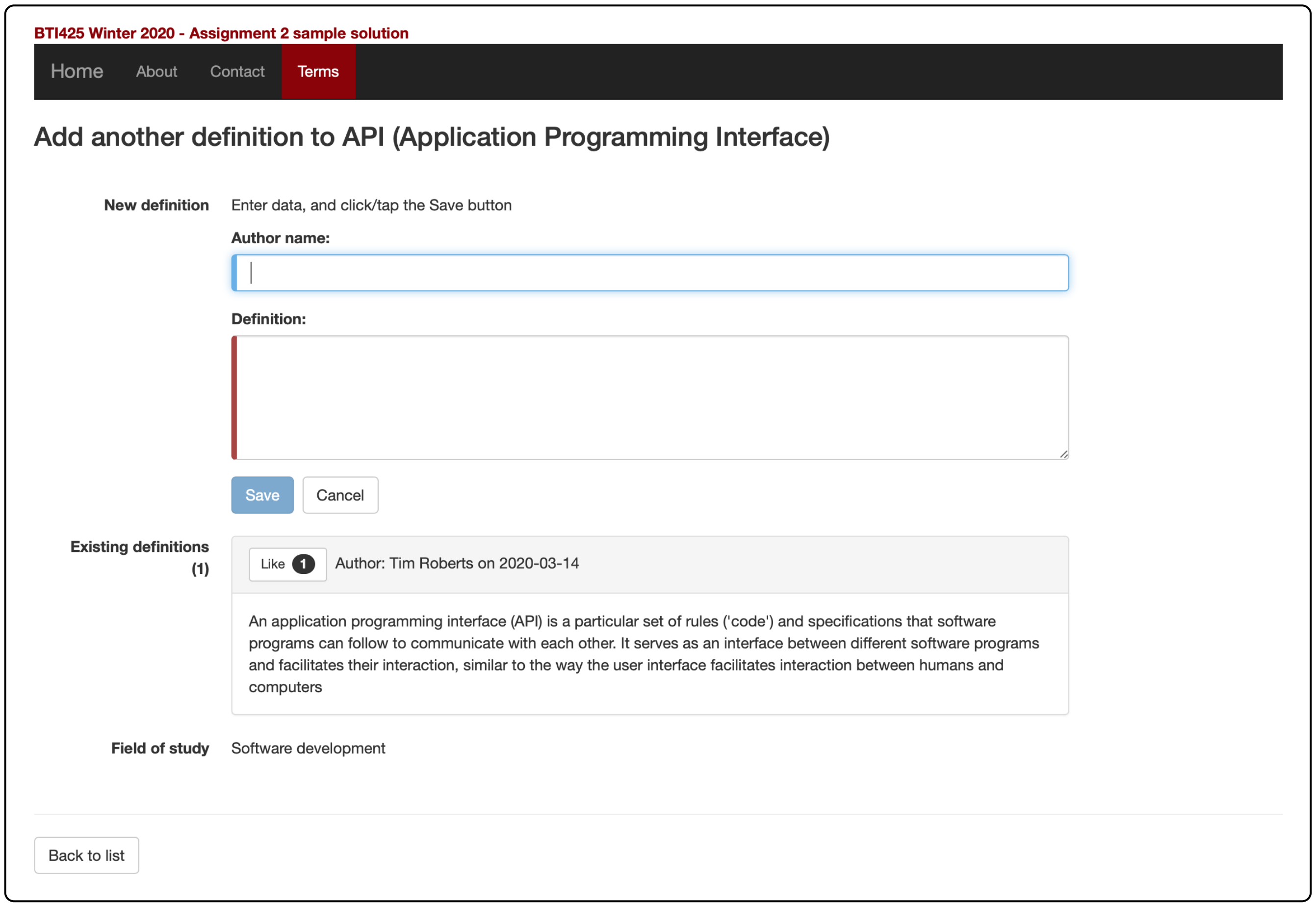
When you are composing the form, remember that the web API request will need the English term identifier, so make sure that you pass that along from the term detail component to the “add new” definition component.

How should this be seen in the user interface, and supported in code? Well, there are several ways. One way is to recognize that the “English term detail” view (above) already has the existing definitions. Maybe we should just configure an “add another definition” button or a link to that view. That’s what you see above in the screen capture. Obviously, we need another route. Then, for either a button or a link, it will navigate to this new component.

How must we design this new component? Well, from a component code perspective, it needs to load the current English term, so that it has all the data it needs (including identifier). And, it obviously must have a “save” handler method to process the user input.

What should its UI be? This can vary widely. One approach that your professors took in the sample solution was to recognize that context is important, so we wanted to display the current or existing definition(s) during the “add new definition” process. So essentially, the “detail” component template markup was copied to this new component’s template markup, and a form was added. We only need to gather the author name and the definition text, as the other “Definition” object properties are generated in code.

After the “add new” completes successfully, remember to follow the PRG pattern, and redirect (probably to the “detail” component).



Add another definition to API (Application Programming Interface)

New definition: Enter data, and click/tap the Save button

Author name:

Definition:

Existing definitions (1)

Author: Tim Roberts on 2020-03-14

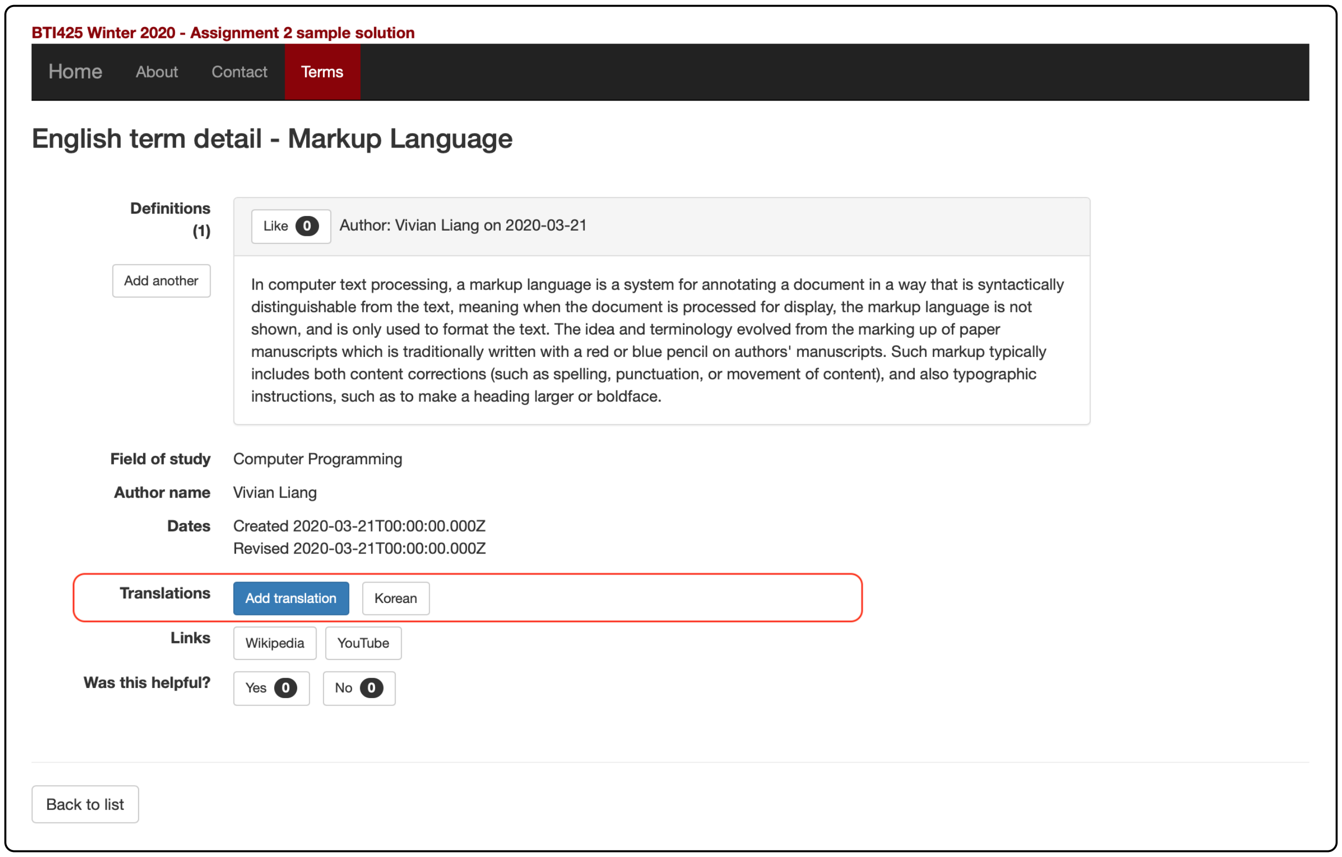
An application programming interface (API) is a particular set of rules ('code') and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers

Field of study: Software development

Updates to the “get one” (detail) for an English term

There are several different ways to think about the task flow and UI for displaying translations (if they exist). Customize your app to meet your needs, design, and preferences. Below, we describe what was done in the professor's sample solution.

In the professor's sample solution, a new strip of links was added to the "get one" (detail) for English term:


BTI425 Winter 2020 - Assignment 2 sample solution

Home About Contact Terms

English term detail - Markup Language

Definitions (1)	Like 0 Author: Vivian Liang on 2020-03-21
Add another	In computer text processing, a markup language is a system for annotating a document in a way that is syntactically distinguishable from the text, meaning when the document is processed for display, the markup language is not shown, and is only used to format the text. The idea and terminology evolved from the marking up of paper manuscripts which is traditionally written with a red or blue pencil on authors' manuscripts. Such markup typically includes both content corrections (such as spelling, punctuation, or movement of content), and also typographic instructions, such as to make a heading larger or boldface.
Field of study	Computer Programming
Author name	Vivian Liang
Dates	Created 2020-03-21T00:00:00.000Z Revised 2020-03-21T00:00:00.000Z
Translations	Add translation Korean
Links	Wikipedia YouTube
Was this helpful?	Yes 0 No 0

Back to list

Code was added to fetch, from the web API, the collection (of zero or more) translations for the specific English term. If one or more translations exist, a button-styled link was generated.

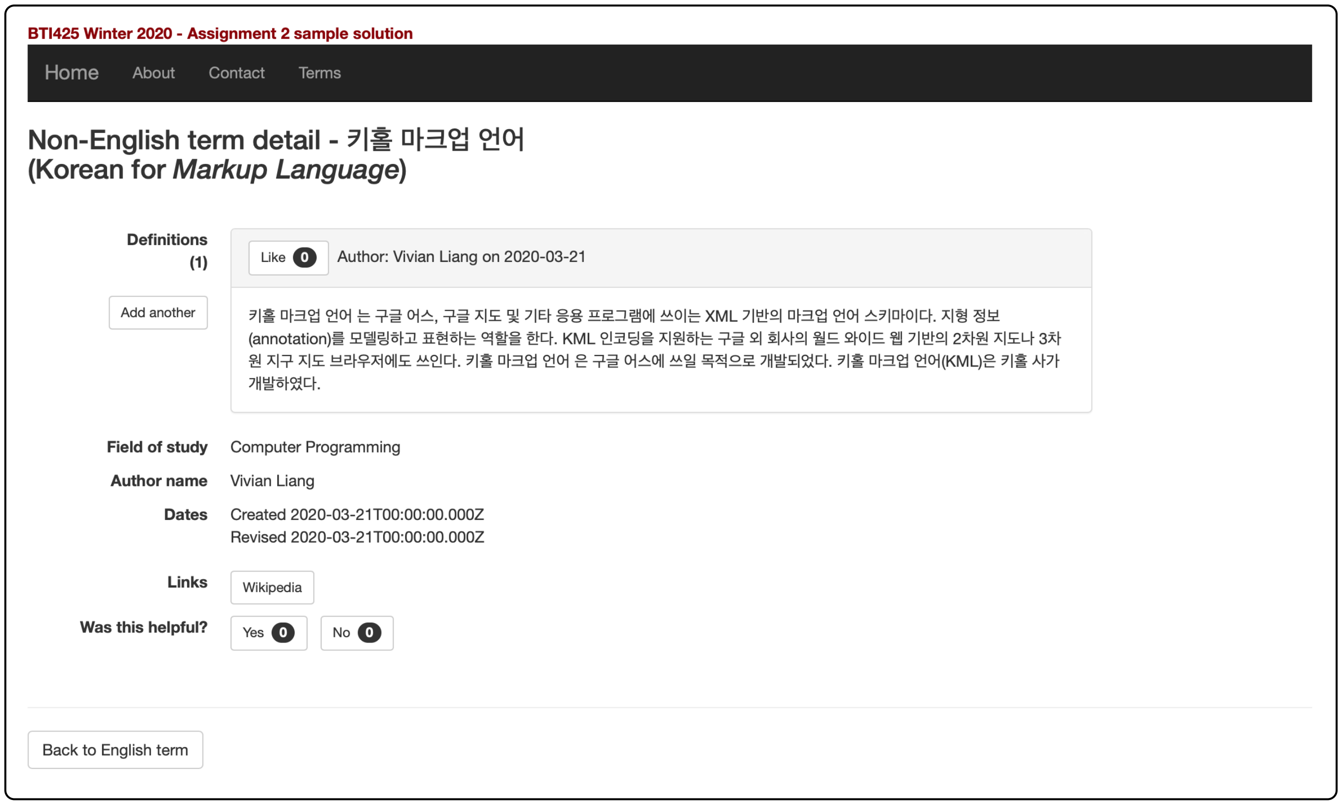
The text label on the button-styled link is generated by a lookup of the languages collection. So yes, in the "get one" (detail) for the English term, code was added to fetch the languages, and then do the lookup locally (in the JavaScript array of results).

> *The language code must be from the professor's sample solution for the web API. The URL is:*
> *`http://pam-2020-a2and3webapi.herokuapp.com/api/languages`*
> *It is OK to use that resource directly from your data manager service. You must write a data class that matches the shape of the collection's items. It is not necessary to add this to your own web API and database (but if you do want to, go ahead).*

Get one translation for an English term

We need a classic "get one" detail component, that will render a non-English translation. We obviously have the identifier for the non-English translation, and it will call the web API method that will look at the non-English collection (at the database) and return the match (or nothing).

At a minimum, it is a repeat of the "get one" detail component for an English term, with maybe a few UI edits. Make sure you bind the value of the language code to the definition container's `lang` attribute.



Non-English term detail - 키홀 마크업 언어
(Korean for *Markup Language*)

Definitions (1)	Like 0 Author: Vivian Liang on 2020-03-21
Add another	키홀 마크업 언어 는 구글 어스, 구글 지도 및 기타 응용 프로그램에 쓰이는 XML 기반의 마크업 언어 스키마이다. 지형 정보 (annotation)를 모델링하고 표현하는 역할을 한다. KML 인코딩을 지원하는 구글 외 회사의 월드 와이드 웹 기반의 2차원 지도나 3차원 지구 지도 브라우저에도 쓰인다. 키홀 마크업 언어 은 구글 어스에 쓰일 목적으로 개발되었다. 키홀 마크업 언어(KML)은 키홀 사가 개발하였다.
Field of study	Computer Programming
Author name	Vivian Liang
Dates	Created 2020-03-21T00:00:00.000Z Revised 2020-03-21T00:00:00.000Z
Links	Wikipedia
Was this helpful?	Yes 0 No 0

Back to English term

Add a new translation for an English term

Like the "add new" English term above, we must enable the ability to add a *new translation* for an existing English term.

This is a classic implementation of an "add new" use case. Very similar to the one above (for an English term), but different because it targets a different web API resource and database collection (terms non-English). Also, a new non-English term MUST include the English term identifier (you will remember from above about this additional field in the schema / class).

We have learned in the past that we typically define a route for an "add new" task that has the collection name and the word "create" in it. In your app, maybe the "add new" English term route looks something like this: `/termsenglish/create`

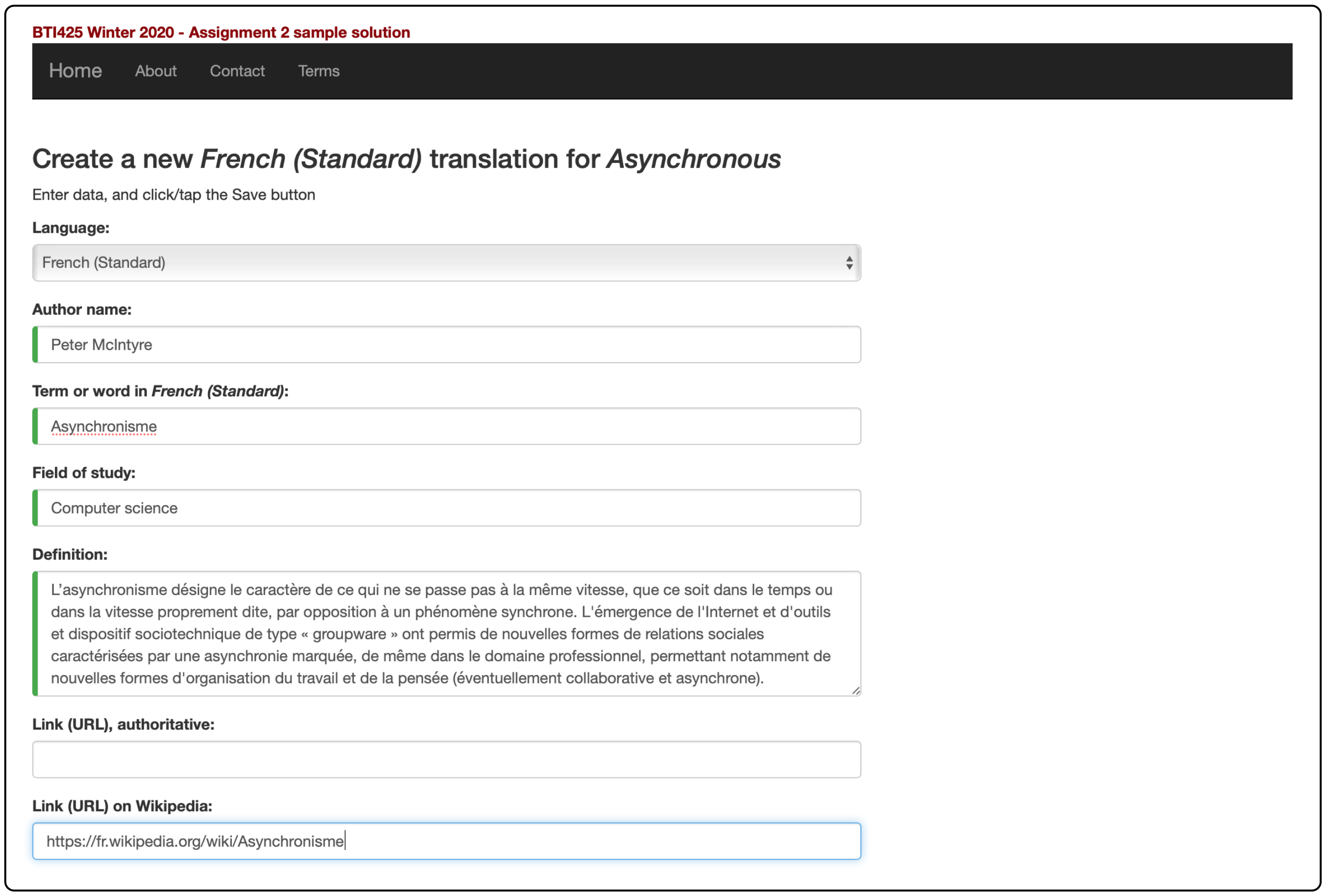
However, for this "add new" non-English term task, we suggest that you design a route that accepts the English term's identifier, because you'll need that. Maybe the "add new" non-English term route will look something like this: `/termsother/create/:id`

Then, when the component loads, the code will have the English term identifier, and can do a lookup, and include it as the value of the `termEnglishId` property as the data package is built up.

Also, when the component loads, the code must fetch the languages collection so that they can be rendered in an item-selection control (e.g. a drop-down list or whatever). The user must be able to select the language of this specific translation.

> *The language code must be from the "/api/languages" resource in the professor's web API sample solution.*

In the `select option` element group, the visible text on the item-selection control must be the language name. The non-displayed value attribute must be the language code (which ends up getting stored in the database).



Create a new *French (Standard)* translation for *Asynchronous*

Enter data, and click/tap the Save button

Language: French (Standard)

Author name: Peter McIntyre

Term or word in *French (Standard)*: Asynchronisme

Field of study: Computer science

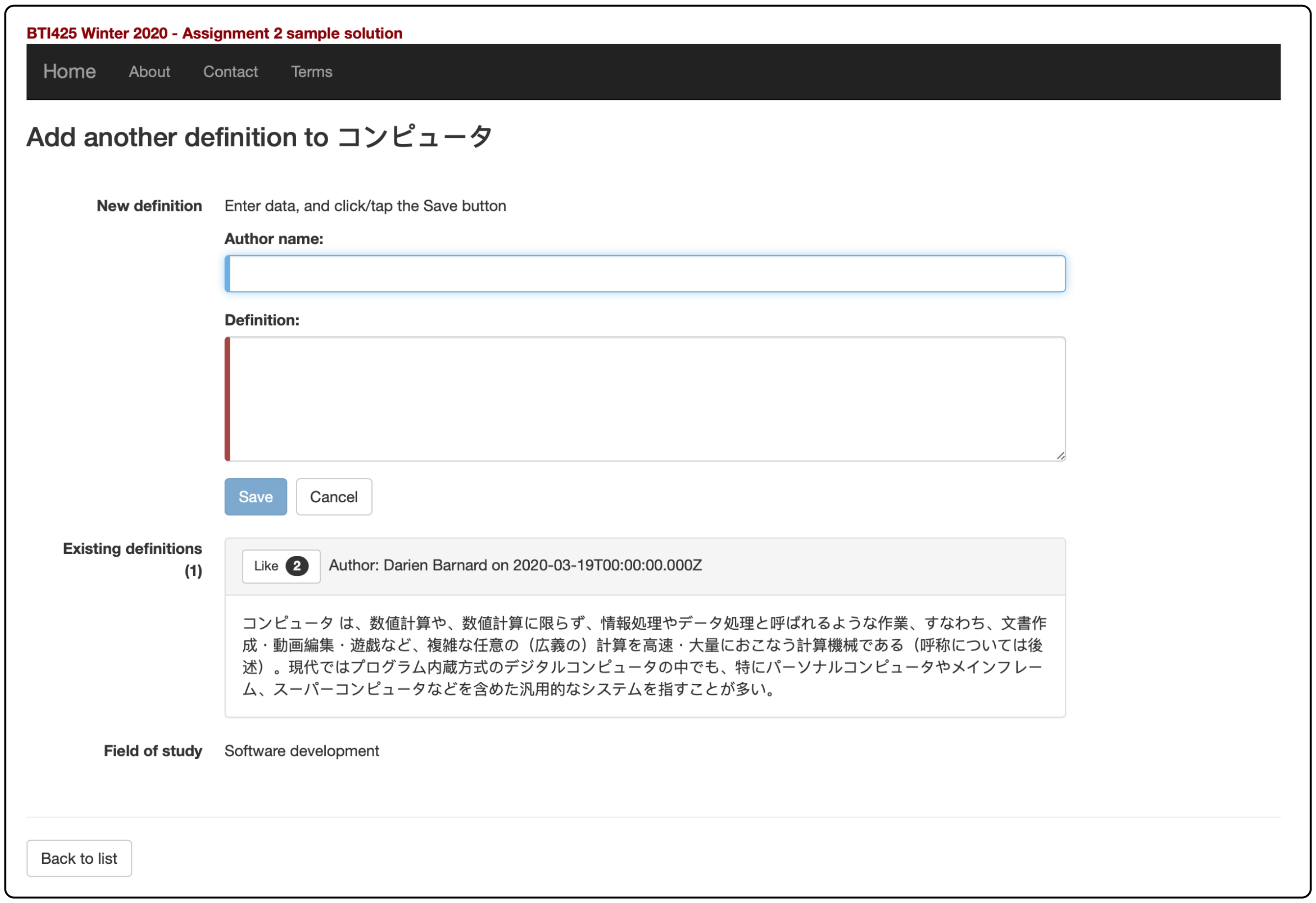
Definition: L'asynchronisme désigne le caractère de ce qui ne se passe pas à la même vitesse, que ce soit dans le temps ou dans la vitesse proprement dite, par opposition à un phénomène synchrone. L'émergence de l'Internet et d'outils et dispositif sociotechnique de type « groupware » ont permis de nouvelles formes de relations sociales caractérisées par une asynchronie marquée, de même dans le domaine professionnel, permettant notamment de nouvelles formes d'organisation du travail et de la pensée (éventuellement collaborative et asynchrone).

Link (URL), authoritative:

Link (URL) on Wikipedia: https://fr.wikipedia.org/wiki/Asynchronisme

Add a new definition to a translation

This will be almost the same as the one described above, in the "Add a new definition to an English term" section.


BTI425 Winter 2020 - Assignment 2 sample solution

Add another definition to コンピュータ

New definition: Enter data, and click/tap the Save button

Author name:

Definition:

Existing definitions (1)

Like 2 Author: Darien Barnard on 2020-03-19T00:00:00.000Z

コンピュータ は、数値計算や、数値計算に限らず、情報処理やデータ処理と呼ばれるような作業、すなわち、文書作成・動画編集・遊戯など、複雑な任意の（広義の）計算を高速・大量におこなう計算機械である（呼称については後述）。現代ではプログラム内蔵方式のデジタルコンピュータの中でも、特にパーソナルコンピュータやメインフレーム、スーパーコンピュータなどを含めた汎用的なシステムを指すことが多い。

Field of study: Software development

Testing your work

For this assignment, there is no required external testing capability. Therefore, rely on your browser tools for this step.

Deploy the Angular app to Heroku

Follow the guidance in the course notes, and deploy the Angular app to a new Heroku app.

> *Remember to do as noted above…*
> *Update your home page component to include two standard HTML hyperlinks:*
>
> 1. *One is the URL to your Heroku-hosted (Angular) app*
> 2. *The other is the URL to your Heroku-hosted (DEN) web API*

Grading procedure

Your professor will use a checklist during the grading process. The checklist will include items based on the assignment specifications. No, we will not distribute the checklist before the due date.

Here's some more comments on the grading procedure:

- Part marks can be earned (it's not an all-or-nothing scheme)
- Some marks will be earned for the deployed/hosted web API
- Some marks will be earned for a deployed/hosted Angular app
- Each of the interaction tasks will earn marks
    - Some tasks could be "worth" more than others

Please review (again) the information about grades. To repeat one of its points, you will not earn an "A" simply for meeting a set of specifications. High grades are earned with work that is clearly better than expected (by meeting the specs). Better work includes a range of "qualitative" measures, including code quality, app and/or UI appearance, organization, content formatting, building upon foundations, and so on.

Reminder about academic honesty

You must comply with the College's academic honesty policy. Although you may interact and collaborate with others, you must submit your own work.

When you are ready to submit your work, you will copy some of the code in your project to plain text files, so that the My.Seneca/Blackboard "SafeAssign" tool can do its job. The next section will tell you which files to copy.

SafeAssign compares your work with that of other current and past students, and with existing works on the web. It uses techniques that are difficult to defeat. The overall goal is to identify copied work.

Submitting your work

We need both the Node+Express web API and the Angular web app.

Here's how to submit your work, before the due date and time:

Node+Express web API

1. Locate the folder that holds your project files.

2. Make a copy of the folder. (You will be uploading a zipped version of the copy.)

3. Inside that folder, remove (delete) the `node_modules` folder. Your professor does NOT need that folder. Also, if it has a `.git` folder, remove that too.

4. Still in that folder, add a new folder named "MyCode". Copy these source code files to the "MyCode" folder:
   The JavaScript (JS) file that holds the "server" code

The JS file that holds the "manager" code
The JS file(s) that holds the "schema" code
For each of these files in the MyCode folder, change the file name extension to "txt".

5. Compress/zip the copied folder. Maybe the name should be something like "webapi.zip". The zip file SHOULD be about 1MB in size. If it isn't, you haven't followed the instructions properly.

Angular web app

1. Locate the folder that holds your project files.

2. Make a copy of the folder. (You will be uploading a zipped version of the copy.)

3. Inside that folder, remove (delete) the `node_modules` folder. Your professor does NOT need that folder. Also, if it has a `.git` folder, remove that too.

4. Still in that folder, add a new folder named "MyCode". Copy these source code files to the "MyCode" folder:
   your data classes .ts file
   your data manager service .ts file
   your app-routing.module.ts file
   each new component .ts file, and .html file (the components YOU created)
   For each of these files in the MyCode folder, add "txt" to the file name extension. For example, "term-english-create.component.ts" will get ".txt" added to the end, to become "term-english-create.component.ts.txt".

5. Compress/zip the copied folder. Maybe the name should be something like "angularapp.zip". The zip file SHOULD be about 1MB in size. If it isn't, you haven't followed the instructions properly.

Bundle both of them together

Ideally, bundle both of the zip files from above into a single zip file, maybe named something like "assignment2.zip". Then…

Login to My.Seneca.
Open the course area.
Click the "Assignments" link on the left-side navigator.
Follow the link for this assignment.
Submit/upload your zip file. The page will accept three submissions, so if you upload, then decide to fix something and upload again, you can do so.