Module 18 Challenge

Start Assignment

Due Tuesday by 11:59pm **Points** 100 **Submitting** a text entry box

NoSQL Challenge: Social Network API

MongoDB is a popular choice for many social networks due to its speed with large amounts of data and flexibility with unstructured data. Over the last part of this course, you'll use several of the technologies that social networking platforms use in their full-stack applications. Because the foundation of these applications is data, it's important that you understand how to build and structure the API first.

Your challenge is to build an API for a social network web application where users can share their thoughts, react to friends' thoughts, and create a friend list. You'll use Express.js for routing, a MongoDB database, and the Mongoose ODM. In addition to using the **Express.js (https://www.npmjs.com/package/express)** and **Mongoose (https://www.npmjs.com/package/mongoose)** packages, you may also optionally use a JavaScript date library of your choice or the native JavaScript `Date` object to format timestamps.

Because this application won't be deployed, you'll also need to create a walkthrough video that demonstrates its functionality and all of the following acceptance criteria being met. You'll need to submit a link to the video and add it to the README of your project.

User Story

```
AS A social media startup
I WANT an API for my social network that uses a NoSQL database
SO THAT my website can handle large amounts of unstructured data
```

Acceptance Criteria

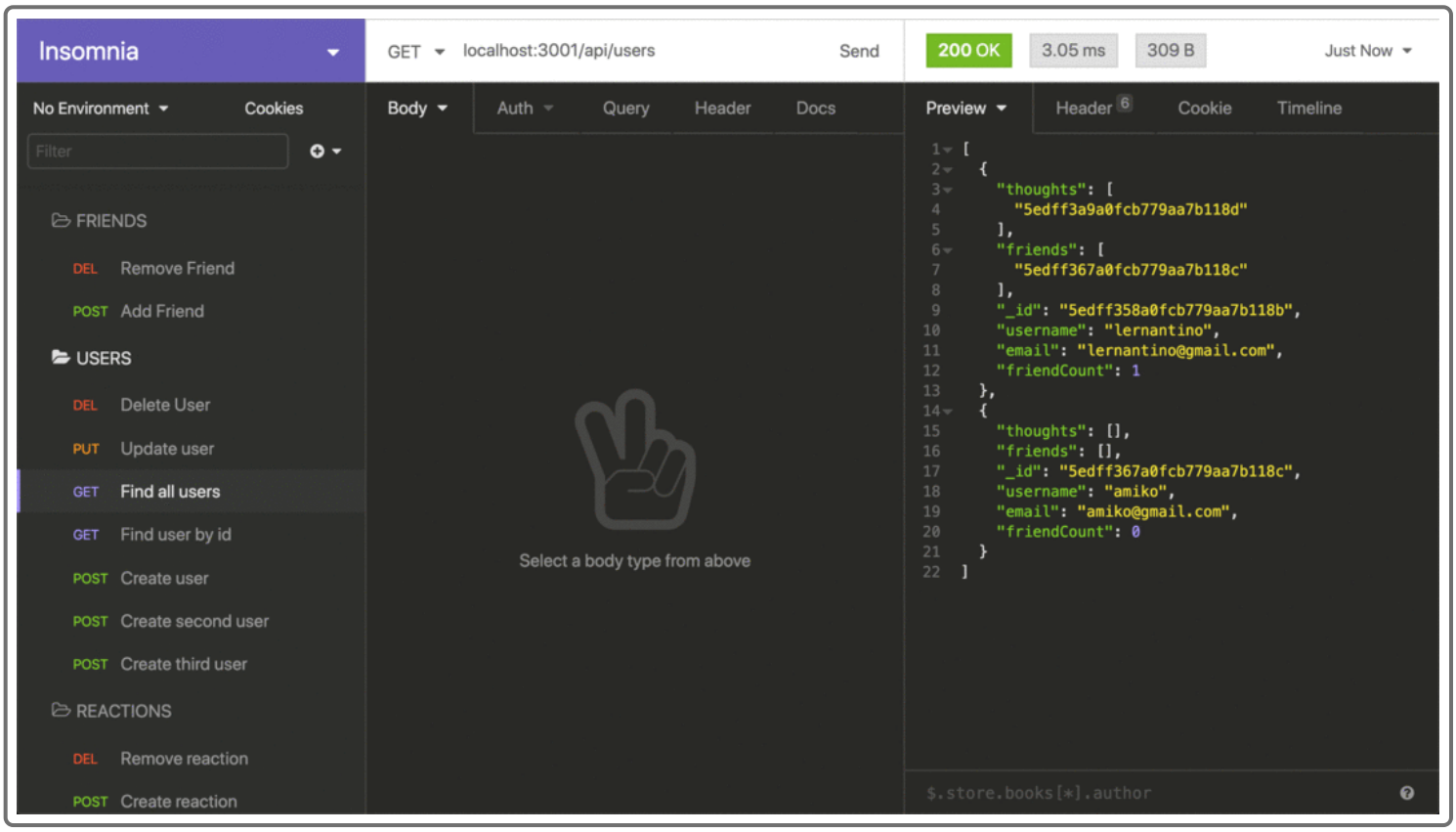
```
GIVEN a social network API
WHEN I enter the command to invoke the application
THEN my server is started and the Mongoose models are synced to the MongoDB database
WHEN I open API GET routes in Insomnia for users and thoughts
```

```
THEN the data for each of these routes is displayed in a formatted JSON
WHEN I test API POST, PUT, and DELETE routes in Insomnia
THEN I am able to successfully create, update, and delete users and thoughts in my database
WHEN I test API POST and DELETE routes in Insomnia
THEN I am able to successfully create and delete reactions to thoughts and add and remove friends to a
```

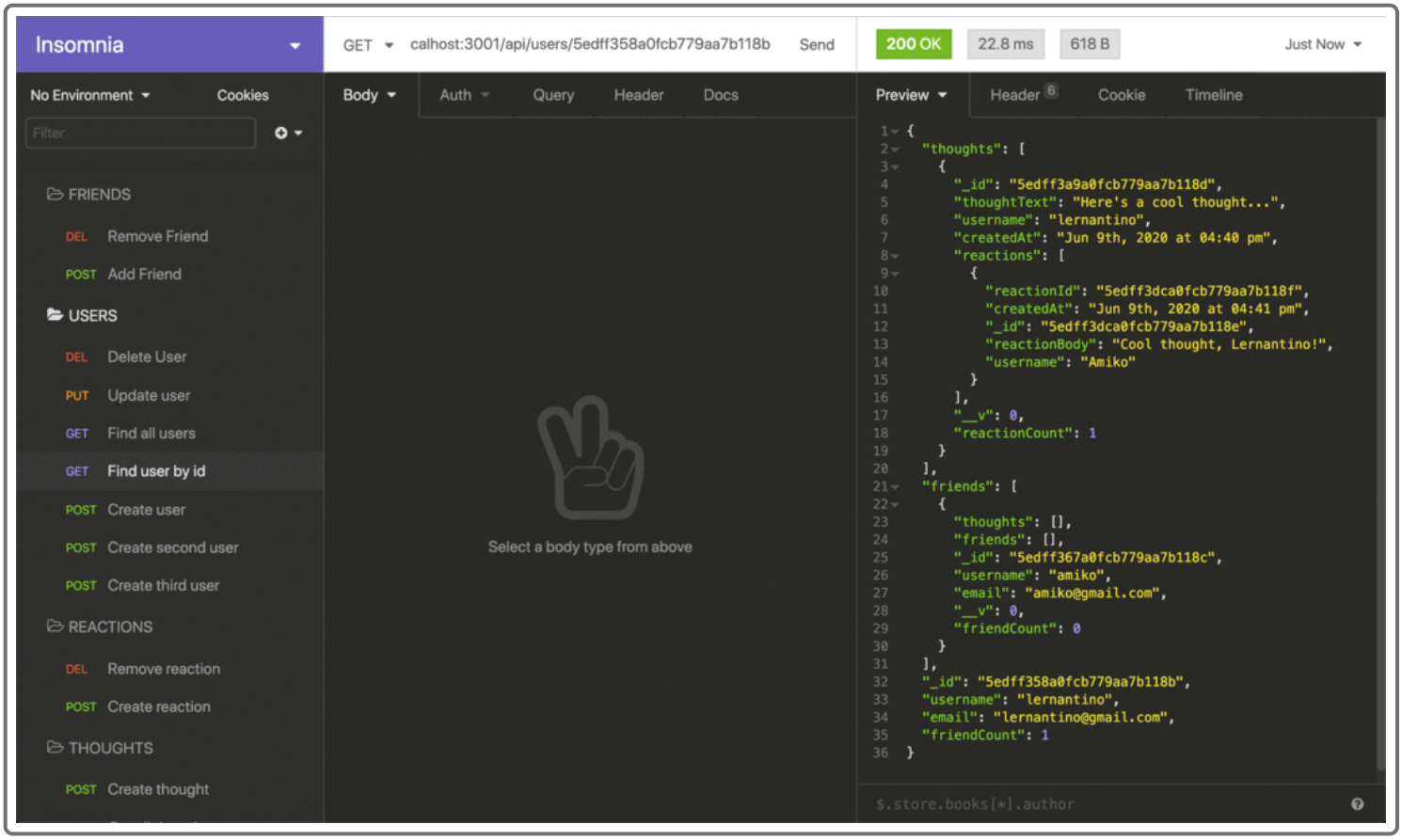
Mock-Up

The following animations show examples of the application's API routes being tested in Insomnia.

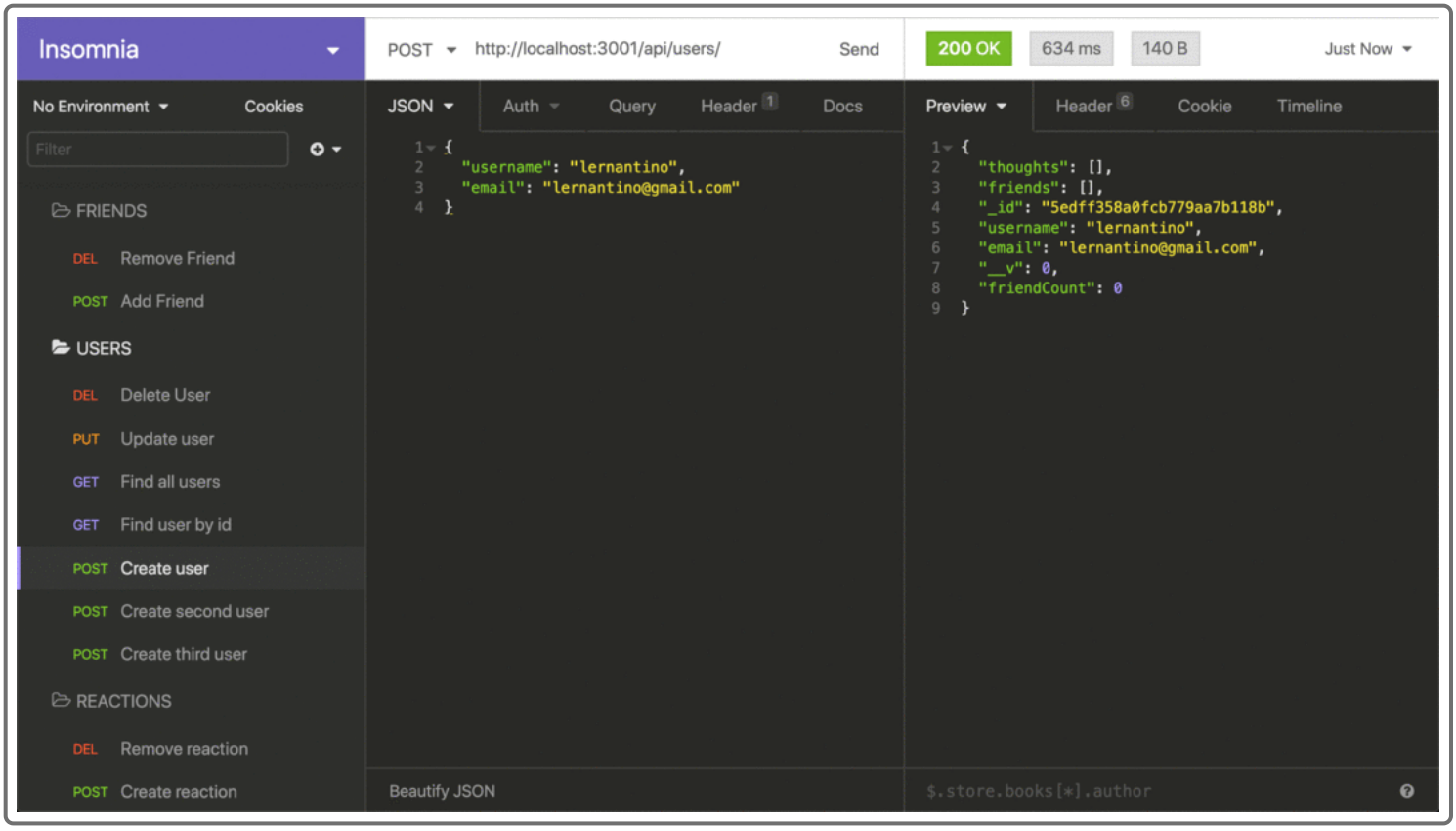
The following animation shows GET routes to return all users and all thoughts being tested in Insomnia:



The following animation shows GET routes to return a single user and a single thought being tested in Insomnia:

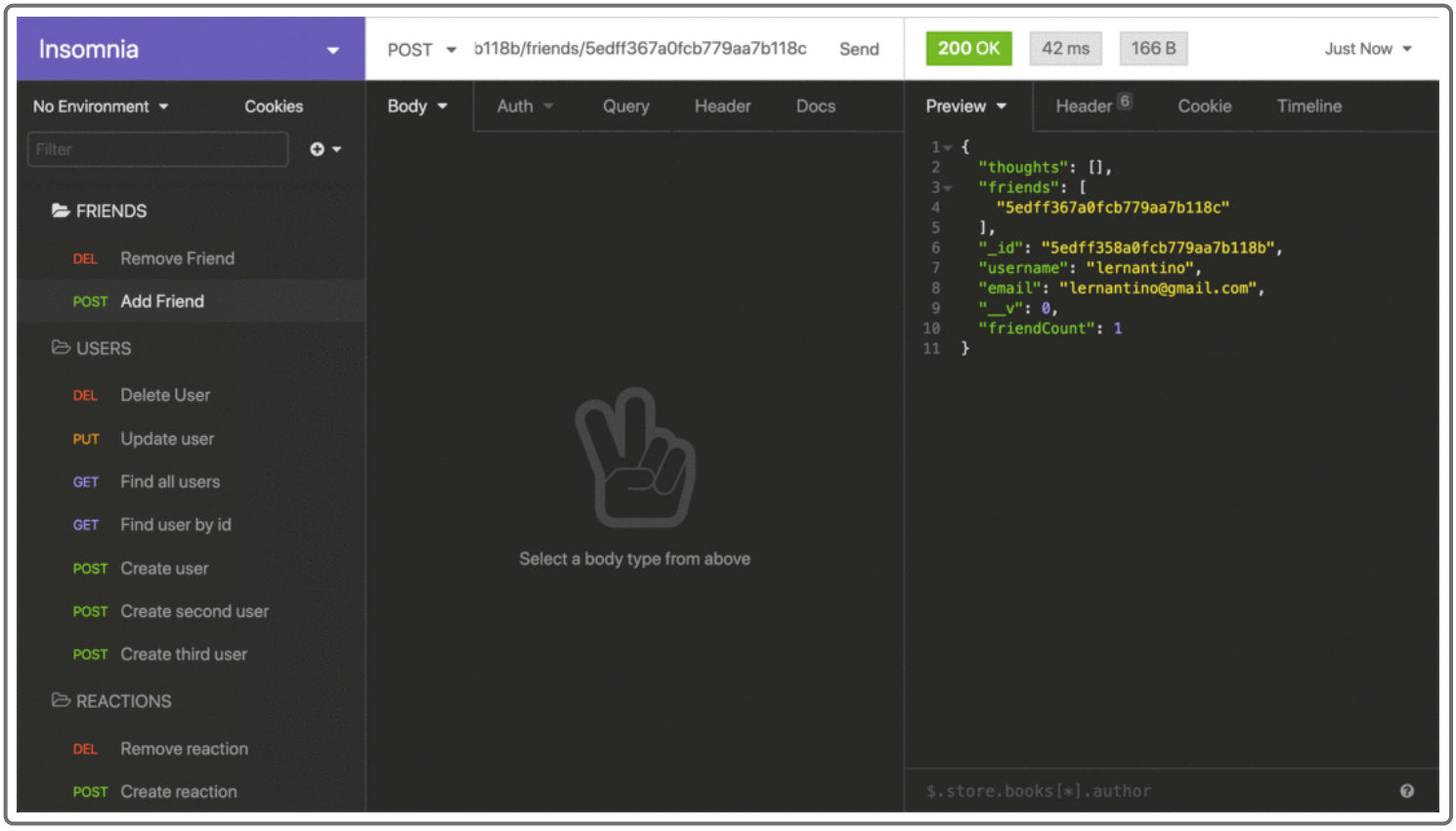


The following animation shows the POST, PUT, and DELETE routes for users being tested in Insomnia:



In addition to this, your walkthrough video should show the POST, PUT, and DELETE routes for thoughts being tested in Insomnia.

The following animation shows the POST and DELETE routes for a user's friend list being tested in Insomnia:



In addition to this, your walkthrough video should show the POST and DELETE routes for reactions to thoughts being tested in Insomnia.

Getting Started

Use the following guidelines to set up your models and API routes:

Models

User

- `username`
  - String
  - Unique
  - Required
  - Trimmed
- `email`
  - String
  - Required

  - Unique
  - Must match a valid email address (look into Mongoose's matching validation)
- `thoughts`
  - Array of `_id` values referencing the `Thought` model
- `friends`
  - Array of `_id` values referencing the `User` model (self-reference)

Schema Settings

Create a virtual called `friendCount` that retrieves the length of the user's `friends` array field on query.

Thought

- `thoughtText`
  - String
  - Required
  - Must be between 1 and 280 characters
- `createdAt`
  - Date
  - Set default value to the current timestamp
  - Use a getter method to format the timestamp on query
- `username` (The user that created this thought)
  - String
  - Required
- `reactions` (These are like replies)
  - Array of nested documents created with the `reactionSchema`

Schema Settings

Create a virtual called `reactionCount` that retrieves the length of the thought's `reactions` array field on query.

Reaction (SCHEMA ONLY)

- `reactionId`
  - Use Mongoose's ObjectId data type

  - Default value is set to a new ObjectId

- `reactionBody`
  - String
  - Required
  - 280 character maximum

- `username`
  - String
  - Required

- `createdAt`
  - Date
  - Set default value to the current timestamp
  - Use a getter method to format the timestamp on query

Schema Settings

This will not be a model, but rather will be used as the `reaction` field's subdocument schema in the `Thought` model.

API Routes

`/api/users`

- `GET` all users
- `GET` a single user by its `_id` and populated thought and friend data
- `POST` a new user:

```
// example data
{
  "username": "lernantino",
  "email": "lernantino@gmail.com"
}
```

- `PUT` to update a user by its `_id`
- `DELETE` to remove user by its `_id`

BONUS: Remove a user's associated thoughts when deleted.

`/api/users/:userId/friends/:friendId`

- `POST` to add a new friend to a user's friend list
- `DELETE` to remove a friend from a user's friend list

`/api/thoughts`

- `GET` to get all thoughts
- `GET` to get a single thought by its `_id`
- `POST` to create a new thought (don't forget to push the created thought's `_id` to the associated user's `thoughts` array field)

```
// example data
{
  "thoughtText": "Here's a cool thought...",
  "username": "lernantino",
  "userId": "5edff358a0fcb779aa7b118b"
}
```

- `PUT` to update a thought by its `_id`
- `DELETE` to remove a thought by its `_id`

`/api/thoughts/:thoughtId/reactions`

- `POST` to create a reaction stored in a single thought's `reactions` array field
- `DELETE` to pull and remove a reaction by the reaction's `reactionId` value

---

Grading Requirements

NOTE

If a Challenge assignment submission is marked as “0”, it is considered incomplete and will not count towards your graduation requirements. Examples of incomplete submissions include the following:

- A repository that has no code
- A repository that includes a unique name but nothing else

- A repository that includes only a README file but nothing else
- A repository that only includes starter code

This Challenge is graded based on the following criteria:

Deliverables: 10%

- Your GitHub repository containing your application code.

Walkthrough Video: 37%

- A walkthrough video that demonstrates the functionality of the social media API must be submitted, and a link to the video should be included in your README file.
  - The walkthrough video must show all of the technical acceptance criteria being met.
  - The walkthrough video must demonstrate how to start the application's server.
  - The walkthrough video must demonstrate GET routes for all users and all thoughts being tested in Insomnia.
  - The walkthrough video must demonstrate GET routes for a single user and a single thought being tested in Insomnia.
  - The walkthrough video must demonstrate POST, PUT, and DELETE routes for users and thoughts being tested in Insomnia.
  - Walkthrough video must demonstrate POST and DELETE routes for a user's friend list being tested in Insomnia.
  - Walkthrough video must demonstrate POST and DELETE routes for reactions to thoughts being tested in Insomnia.

Technical Acceptance Criteria: 40%

- Satisfies all of the preceding acceptance criteria plus the following:
  - Uses the **Mongoose package (https://www.npmjs.com/package/mongoose)** to connect to a MongoDB database.
  - Includes User and Thought models outlined in the Challenge instructions.
  - Includes schema settings for User and Thought models as outlined in the Challenge instructions.
  - Includes Reactions as the `reaction` field's subdocument schema in the Thought model.
  - Uses functionality to format queried timestamps properly.

Repository Quality: 13%

- Repository has a unique name.
- Repository follows best practices for file structure and naming conventions.

- Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.
- Repository contains multiple descriptive commit messages.
- Repository contains a high-quality README with description and a link to a walkthrough video.

Bonus

Fulfilling the following can add 10 points to your grade. Note that the highest grade you can achieve is still 100:

- Application deletes a user's associated thoughts when the user is deleted.

How to Submit the Challenge

You are required to submit BOTH of the following for review:

- A walkthrough video demonstrating the functionality of the application and all of the acceptance criteria being met.
- The URL of the GitHub repository. Give the repository a unique name and include a README describing the project.

NOTE

You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next Module.

Comments are disabled for graded submissions in BootCamp Spot. If you have questions about your feedback, please notify your instructional staff or the Student Success Advisor. If you would like to resubmit your work for an improved grade, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

IMPORTANT

It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo. This applies if you have worked with a peer on an assignment, used code in which you did not author or create sourced from a forum such as Stack Overflow, or you received code outside curriculum content from support staff such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a Challenge or any aspect of the curriculum, please remember that there are student support services available for you:

1. Office hours facilitated by your TA(s)

2. **Tutoring Guidelines** (https://docs.google.com/document/d/1hTldEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing) (schedule a session in the "Tutor Sessions" section of Bootcampspot)
3. AskBCS Learning Assistants