Animations In Stress Learning Content: Final Report

HalfGrads

Product Owner:

Charles Rothrock (cwrothrock_17@tamu.edu)

Scrum Master:

Benjamin D’Antonio (benjamin.dantonio@tamu.edu)

Members:

Shane Poldervaart (shane.poldervaart@tamu.edu)

Siddharth Sundar (ssundar@tamu.edu)

Kevin Tang (kdt0116@tamu.edu)

Yiqing Zhao (yiqingzhao@tamu.edu)

Course Information:

CSCE 606: Software Engineering

Instructor: Dr. Hank Walker

Spring 2020

May 5, 2020

Table of Contents

1. **Summary**

This project set out over two months ago to create a set of animations and interactive games that could be added to the StepStone learning environment, part of the Peer Program's One Health curriculum. More specifically, our team focused on the Stress Module, one of the many themed clusters of activities included in the application so far. We were given a set of slides covered by the module and a list of animations/games requested by Torri Whitaker, a content specialist for the Peer Program. Ultimately, HalfGrads became responsible for implementing as many of the desired animations for the benefit of the College of Veterinary Medicine at TAMU, the Peer Program in particular, and for Dr. Walker who was our primary customer and point of contact.

We chose to implement the animations using HTML5, standard CSS, and basic JavaScript. We were able to give each of our six team members a user story such that each user story corresponded to a specific slide's animation; some animations were reusable across multiple slides. As a result of this type of organization, we did not follow the traditional format of completing a set of user stories each iteration; instead, each iteration involved progress on the development and testing of each individual "mini-app", as they became called, with the last iteration involving the integration of the animations into production. After testing using Cucumber-js and selenium-webdriver, as well as hand-testing the apps in production across devices and browsers, we were able to declare all but one user story as finished. This project was heavily dependent on our team's ability to learn how to use StepStone, figure out JavaScript and the testing of JS-based applications on the fly, and the collaboration with other teams and StepStone developer Daniel Shuta, who helped us get our apps into production. The final video demo for our project can be found on YouTube.

2. **Development and Management**

a. **BDD/TDD**

We used Pivotal Tracker to codify our user stories and provide guidance for the behavior necessary in each slide and then implemented the behavior requirements using Cucumber-js and Selenium WebDriver. This proved useful in ensuring that requirements for a user story were being fulfilled, despite the disconnect between the production and testing environments of StepStone. Because the StepStone framework required an iframe resizer script to make the embedded apps fit inside of the frame, automating testing for apps inside a StepStone browser was not compatible. In turn, there were commonly caused issues when attempting to deploy for the apps which could not be caught in the standard behavior testing or locally automated testing.

b. Configuration Management

We configured our code management with a few releases and no branches. This was because each user story we implemented was completely modular and separate from all other stories, as they were each their own unique animation to be placed in StepStone. With no branches and four releases, one for each iteration after iteration 0, it was easy to manage our project. Each release contained all of the progress towards completing our user stories as of the end of the corresponding iteration.

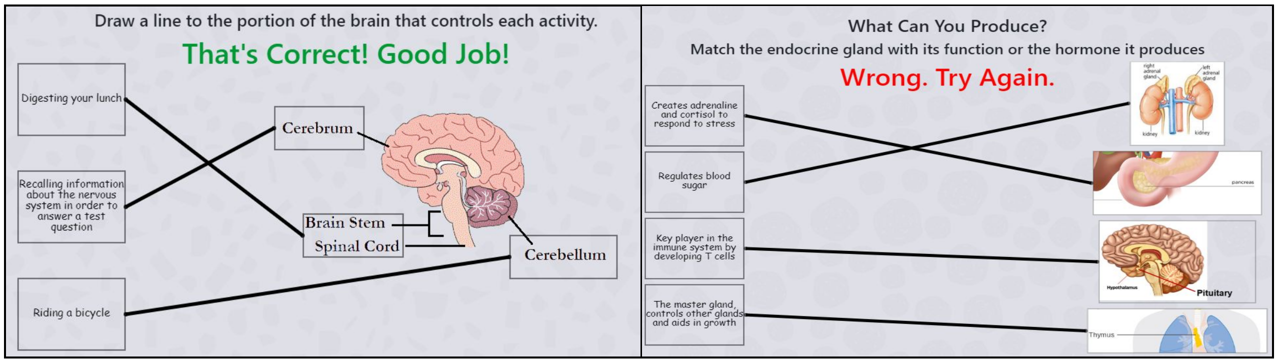
We only had one spike. We discovered that the *Slide 15: Multiselect Question* story contained a slide in the StepStone stress module with a relatively clean multi-select question. Since it included all of the functionality we believed necessary, we conferred with our customer and were able to close this extraneous story.

3. User Stories

a. User Story 1 - Slide 12 & 42: Matching / Connect

```
Points: 1, Status: Complete
Feature: Answer a matching question
As a student, so that I can learn portions of the brain that control
certain activities, I want to connect an activity to a brain portion
and receive feedback for being "correct" or "incorrect".
```

This story did not change throughout development. It was simple and straightforward on the expectations and possible solution: users press the left mouse button and drag between their answer and the portion of the brain or picture associated with the answer; the line is drawn once the user finishes dragging and releases the mouse left button. If users want to change their original answer and picture matching, they can redo the line creation process as described previously, and the old line will get replaced with the new one. If all answers are matched then the mini-app will notify the user if it was done correctly or incorrectly. Here are the finished mini-apps for slide 12 on the left showing a successful solve and slide 42 on the right showing an unsuccessful solve:


Draw a line to the portion of the brain that controls each activity.

That's Correct! Good Job!

What Can You Produce?

Match the endocrine gland with its function or the hormone it produces

Wrong. Try Again.
Digesting your lunch
Recalling information about the nervous system in order to answer a test question
Riding a bicycle
Cerebrum
Brain Stem
Spinal Cord
Cerebellum
Creates adrenaline and cortisol to respond to stress
Regulates blood sugar
Key player in the immune system by developing T cells
The master gland, controls other glands and aids in growth
right adrenal gland
left adrenal gland
kidney
kidney
pancreas
Hypothalamus
Pituitary
Thymus

b. **User Story 2 - Slide 15: Multiselect Question**

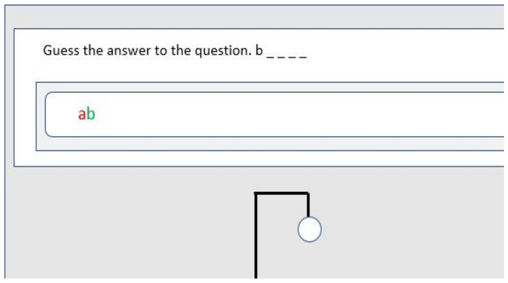
`Points: 0, Status: Complete`

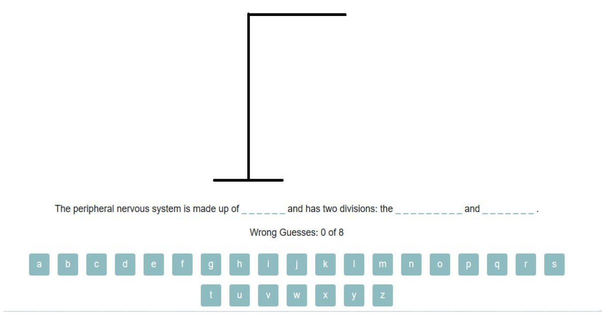
This story was the target of a spike very early in the project, as we discovered that this specific slide in the stress module already contained a relatively clean multi-select question that included all of the functionality we believed necessary. After conferring with Dr. Walker, we were able to close this story and not devote any development time to recreating the animation.

c. **User Story 3 - Slide 20: Hangman**

```
Points: 3, Status: Complete
Feature: Play hangman in real-time
As a student, so that I can track the correctness of each letter
guessed, I want the hangman figure and word(s) to update in real-time.
```

This story experienced very little change throughout the project. The look-and-feel of the UI, which started as the figure on top, is what changed. It was decided that the game would be much more user-friendly if each letter was given its own button. The flow during interaction with the app grew to involve changing images, disabling buttons, and updating text. The final animation that was pushed into production looks like the figure on the bottom. When one of the letter buttons is clicked, the button becomes disabled, and either the hangman


Guess the answer to the question. b _ _ _ _


The peripheral nervous system is made up of ______ and has two divisions: the __________ and ________.

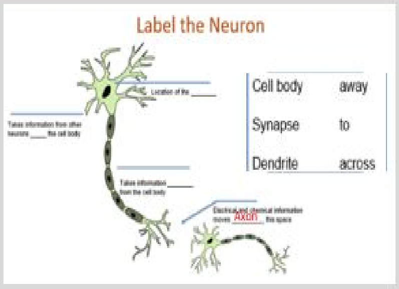
Wrong Guesses: 0 of 8

image updates due to an incorrect guess or the blanks shown in the sentence update with the letter filled into its corresponding places.

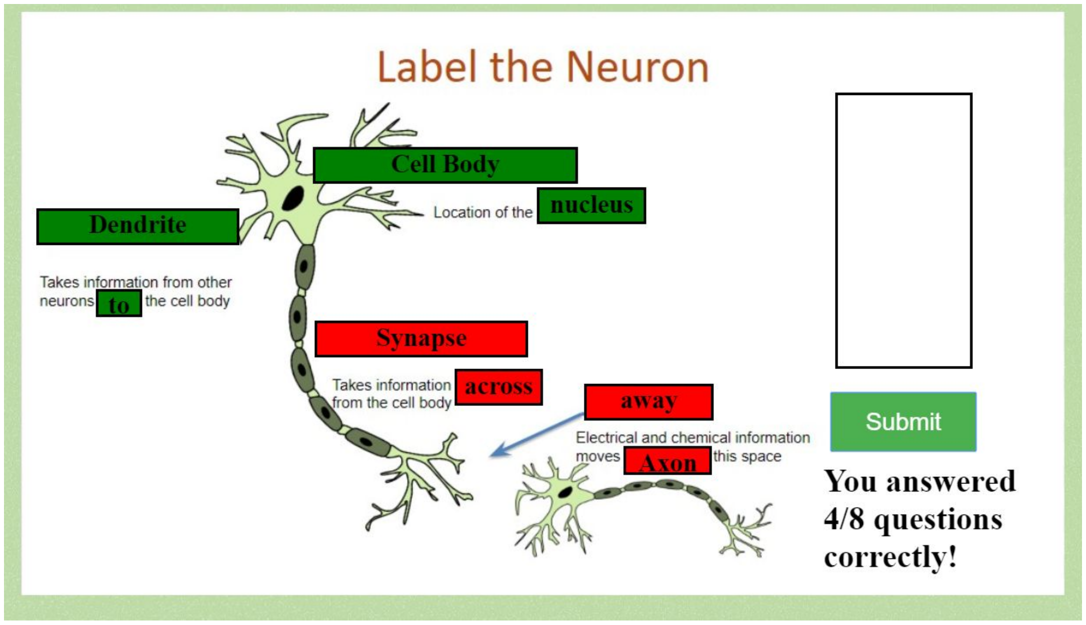
d. User Story 4 - Slide 24,66: Drag and Drop

```
Points: 2, Status: Complete
Feature: Label a diagram with given words/images (drag&drop)
As a student, so that I can learn the different parts of a neuron or
process steps,  I want to drag labels to designated positions and be
told if I am correct or incorrect; I want my correct labels to remain
in place.
```

The animation for Slide 24 involved labeling parts of a Neuron by dropping words from the word bank into corresponding drop boxes. The initial idea was to drop the words into the blanks directly and you would get feedback on whether the answer was correct based on the color of the word. The image below shows an example of incorrect feedback when a user dropped the word Axon (indicated by the red color).


Label the Neuron
Location of the ____
Takes information from other neurons ____ the cell body
Takes information ____ from the cell body
Electrical and chemical information moves Axon the space
Cell body
away
Synapse
to
Dendrite
across

However, instead of dropping the words into the preexisting blank, we added a colored box that covered the above lines to allow the user a specific box to drop the words in; this made interaction much smoother. The feedback would then be based on the color of the box rather than the text: correct answers are indicated by green boxes and incorrect answers are indicated by red boxes. The user also receives a textual description of how many questions you answered correctly upon submission. The answers in the green-colored boxes cannot be modified.



Label the Neuron

Cell Body

Location of the nucleus

Dendrite

Takes information from other neurons to the cell body

Synapse

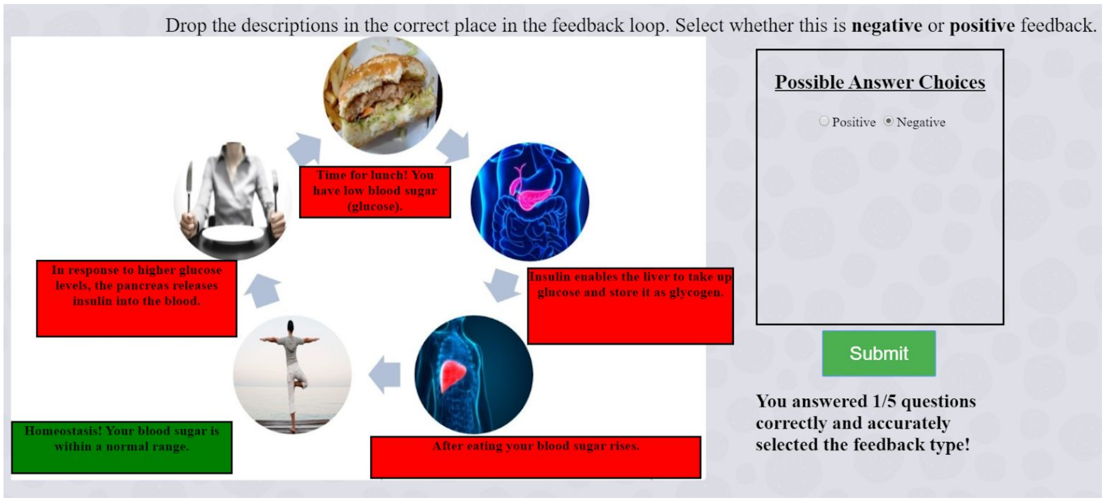
Takes information across from the cell body

away

Electrical and chemical information moves Axon this space

You answered 4/8 questions correctly!

The drag and drop animation for slide 66 used the same mechanics as above, except the user is dragging entire sentences/descriptions. The only other difference was that users are also required to answer an additional question prior to submission, which was achieved through radio buttons. Below is an image of slide 66 indicating the concepts I have mentioned here.



Drop the descriptions in the correct place in the feedback loop. Select whether this is **negative** or **positive** feedback.

Time for lunch! You have low blood sugar (glucose).

Insulin enables the liver to take up glucose and store it as glycogen.

After eating your blood sugar rises.

Homeostasis! Your blood sugar is within a normal range.

In response to higher glucose levels, the pancreas releases insulin into the blood.

Possible Answer Choices

Positive [✓ Negative]

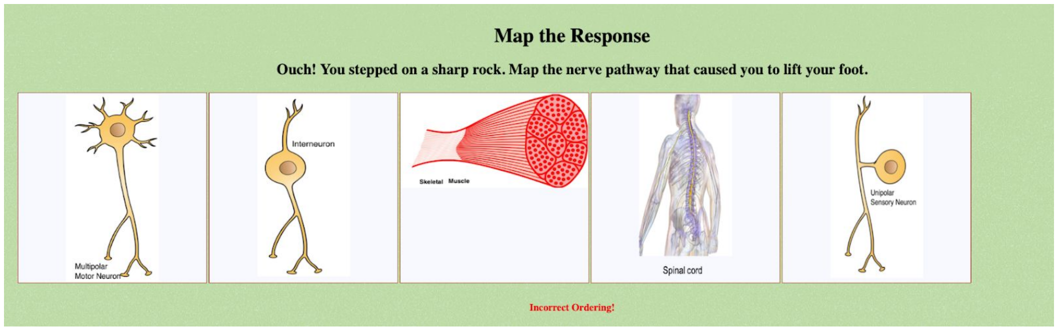
You answered 1/5 questions correctly and accurately selected the feedback type!

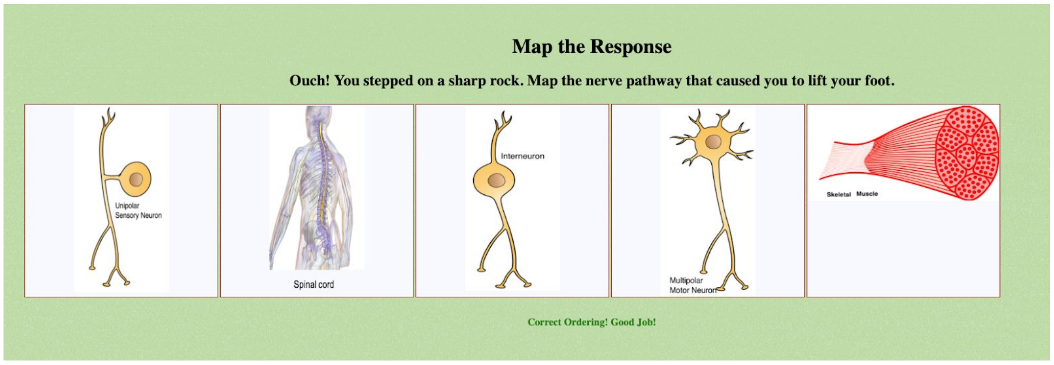
e. User Story 5 - Slide 27: Ordering Images

```
 Points: 2, Status: Complete
Feature: Rearrange images until a specific ordering is achieved.
As a student, so that I can place steps in a process in order, I want
```

`to swap the placement of images and submit my guess to check if I have the correct ordering.`

With the completed app, images are originally in random order, the user simply needs to click on an image of interest to snap the cursor and drag said image to the intended location in between two other images; the to the right or left of the newly placed image will shift to compensate. One deviation from the original plan was that, instead of having the user click a "submit" button upon completion of image reordering, a line of text is constantly presented and updated according to the ordering of the images, as shown in the following images.

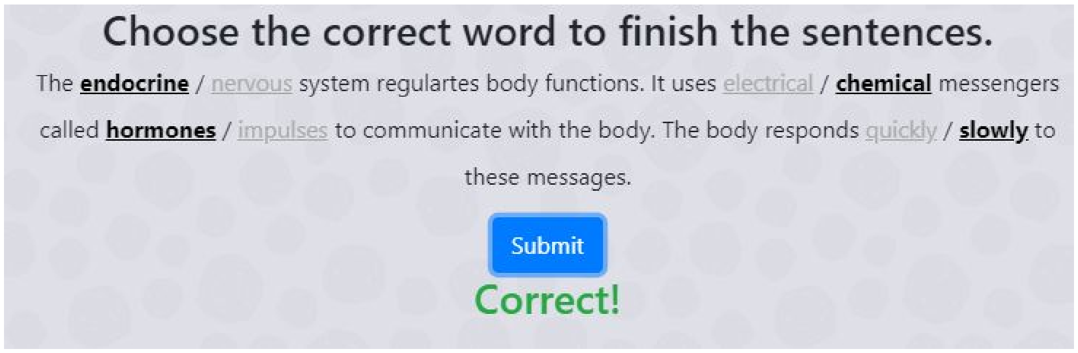



Map the Response
Ouch! You stepped on a sharp rock. Map the nerve pathway that caused you to lift your foot.
Unipolar Sensory Neuron
Spinal cord
Interneuron
Multipolar Motor Neuron
Skeletal Muscle
Correct Ordering! Good Job!

f. User Story 6 - Slide 33: Fill-in-the-blank

```
Points: 1, Status: Complete
Feature:  Select given words in a sentence to fill the blank(s).
As a student, so that I can answer a fill-in-the-blank question,
I want to click one word from a pair of choices, one or more times,
and know if I am correct or incorrect upon submission.
```

The options to select are plainly labeled inside of the question and upon selection have a visual change to indicate the selected term. The final state of the slide was pushed onto stepstone production and looks as below when successfully completed.



Choose the correct word to finish the sentences.

The **endocrine** / nervous system regulartes body functions. It uses electrical / **chemical** messengers called **hormones** / impulses to communicate with the body. The body responds quickly / **slowly** to these messages.

g. **User Story 7 - Slide 42: Scale Balancing**

```
Points: 2, Status: Incomplete
Feature: Drag and drop items on a scale to balance both sides
As a student, so that I can answer two questions simultaneously,
I want to place answers on the left and right parts of a scale and for
the scale to balance when my answers to the questions are correct.
```

A baseline implementation for the story was completed. However, the animation required drag-and-drop functionality, which first needed to be implemented in another user story. Due to all the roadblocks surrounding rolling out and testing animations, the story could not be completed in time.

4. Iterations

As mentioned previously, our non-traditional organization of stories means that we did not complete stories/points each iteration. Each team member built their animation/app throughout the project's duration and ultimately completed it and its corresponding points in the final iteration(s).

a. **Iteration 0**

```
Points Completed: 0
```

We spent this iteration learning StepStone, creating our user stories, contacting our customer, and looking into similar legacy projects. While no previous teams had worked specifically on the Stress Module, the Cell Biology Module contained work from prior teams. We contacted their Scrum Master Unnati Eramangalath so that she could forward us the correspondence between her team, Dr. Walker, and Daniel Shuta during their project.

b. **Iteration 1**

```
Points Completed: 1
```

After revising our premature user stories created in Iteration 0, we cleaned up the stories in Pivotal Tracker, made sure each team member had an assignment, and finally began working on our individual mini-apps. We spent a considerable amount of time discussing our approach to ensuring we met the iframe compatibility required by StepStone, looking over some of the legacy code for similar animations, and learning the interaction between vanilla HTML5/CSS/JavaScript for these types of apps. The one completed point was awarded based on the spike into User Story 2 which was resolved through its exclusion.

c. **Iteration 2**

`Points Completed: 0`

This iteration saw each of our team members make considerable progress on the development of their mini-apps. User Story 1, Story 4, Story 5, and Story 6 were the main apps that saw progress. For an overall demo of this iteration's progress, see the video linked here. Iteration 2 also saw efforts focused on figuring out what testing environments we should use and resulted in us looking at ones like Cucumber-js and Jest. It was also during Iterations 1 and 2 that we learned how to put our apps into a testing environment that mimics StepStone, thanks to the incredible efforts of Daniel Shuta.

d. **Iteration 3**

`Points Completed: 0`

These two weeks involved a great amount of progress on every user story as well as the inclusion of automated testing for the majority of the mini-apps. Again, we chose to use Selenium WebDriver as the browser simulator, Cucumber-js as a BDD framework, and Chai as the assertion library to achieve automated testing; after talking with our customer, it was decided that this automated local testing, as well as heavy manual testing with the apps in production, would be sufficient.

e. **Iteration 4**

`Points Completed: 7`

Along with the conclusion of the last iteration came the integration of most of our apps into production. User Story 3, User Story 4 (slide 24), User Story 5, and User Story 6 were all successfully added to the StepStone Stress Module path by Daniel.

f. **Post Iteration 4**

`Points Completed: 3`

A few more apps needed some fixing after Iteration 4 and they were marked as incomplete in the Iteration 4 documentation. User Story 4 (slide 66) was successfully added to the StepStone Stress Module path, as well as User story 1 (slides 12 and 24).

5. **Customer Meetings**

   a. **February 24th, 2020, 6:00 P.M.:**

      This was our Iteration 0 meeting with Dr. Walker, our customer, which served to introduce our team to the goal of the project in his eyes, get clarification on the expectations of the animations we needed to produce, and ensure we had the necessary materials and contacts we would need to be successful throughout the project.

   b. **March 4th, 2020, 5:30 P.M.:**

      We used this meeting to gain extra clarification on how to get started using StepStone, who we should contact, and make sure everything was in order for us to make progress on our user stories.

   c. **April 8th, 2020, 2:30 P.M.:**

      This meeting occurred during Iteration 3 and served to discuss the progress we were making on the animations. We demoed the User Story 3, but the focus of the meeting was to clarify what our approach to testing our animations should be since we are still under the impression StepStone isn't responsive to things like Capybara. The meeting concluded with us headed towards the use of Cucumber-js and Selenium WebDriver to test our apps locally as well as the goal to get at least one app deployed to production in order to get a feel for what issues it might introduce.

   d. **April 29th, 2020, 4:00 P.M.:**

      With this meeting marking the last meeting and end of the, we demonstrated User Stories 1 and 6, showed more information on our testing setup, and showed the apps that were live in StepStone at the time. Lastly, we asked Dr. Walker a lot of questions about the expectations and requirements for the final report/poster/demo. Since our project wasn't traditional in the sense that we did not build a standalone app with lots of interactions and instead had a lot of standalone small animations completely independent from each other, we wanted to get everything in order.

6. **Issues**

   a. **Pivotal Tracker**

      Organizing user stories originally proved problematic, as each customer item was underspecified. Because of this, we had to reorganize and iterate on our stories multiple times over the course of the project as we better understood what was desired of us. Each story was atomically very large, with each animation being individual but intensive work. Because of this, our Agile process was a bit different from usual, so we had to adjust to this change. We had one story that was already complete, but it took a bit of investigation to uncover that no further work was needed.

b. Integration with StepStone

Because mini apps are placed inside the StepStone environment in a customized “iframe” element, the apps had to be resized using an iframe resizer. If this isn’t triggered properly, the app is incompatible with StepStone; you can refer to Tutorial 5 for more information. Another obstacle we overcame was touch screen compatibility: some mini-apps used mouse events in the canvas element. Since mouse events are not the same as touch screen events, we added event triggers for “touchBegin”, “touchDrag”, and “touchEnd” to certain mini-app scripts.

The browser compatibility issue was complicated because some apps worked flawlessly in all browsers except that *one* where loading or interaction just doesn’t work properly. Some mini-apps used canvas and the “2D context”, which had some issues on Firefox. Wrapping the render function in a “try, catch” statement solved the issue, as the render function would error on load up for firefox. Making sure to test apps on different browsers is important because there might be inconsistencies or discrepancies that would cause your app to fail for users. We also experienced numerous loading issues that would not happen locally. Daniel Shuta added an interval function to the beginning of our *document.ready* function in the JS file and told us to clear the cache when testing, as the iframe resizer works oddly in the StepStone environment.

7. Legacy

a. Prior Code

All legacy material relates to previous semesters’ teams working in the Biology Module. The learning games repository is also useful. The primary functions of the animations repository are the documentation it provides for setting up a jQuery application with Stepstone as well as the slides related to biology made with HTML, CSS, JavaScript, and jQuery. The use of “iframeResizer.contentWindow.min.js” for enabling JS application resizing is crucial for compatibility with StepStone.

b. Our Code and Reconfiguration

If you would like to reuse any of our mini-apps for your own slides:

1. Drag / Connect:

   To reuse the drag and connect animation for other similar slides, you will need to configure the JSON object called “config” in “/src/main.js” defined at the top of the “init” function. This JSON object has a structure that determines the answer boxes (left side) size, position, and text content, and the image boxes (right side). To configure the left boxes, just add box objects in the “boxesLeft” array pertaining to the answer you want to represent. For example, this represents a box on the left side that has text content “Digesting your lunch” and answer at the index 1 in the “boxesRight” array:

   ```
   boxesLeft: [{
           content: “Digesting your lunch”,
           side: “left”,
           answer: 1,
   }]
   ```

   The “boxesRight” array is similar but will require extra configuration, and is best explained in the code. Slide 12 is an example where there is only one image and multiple boxes per image, while slide 42 has multiple images each with only one box.

2. Hangman

   Reusing Hangman is incredibly easy. “hangman.js” is solely contained in a *window.onload* function. At the top of this function, replace the *sentence* variable with whatever sentence you would like filled such that each word guess is replaced by ‘1?’, then ‘2?’, and so on for however many words must be guessed. Then, place each word as a string into the *words* array immediately below *sentence*. If you’ve followed these instructions, as well as the more detailed instructions in the comments within “hangman.js”, you’ll be able to adapt it however you like.

3. Fill-in-the-blank

   Reusing the fill-in-the-blank is straightforward and modular. It requires a few small changes such as the answers array in the wordSelect.js and studentAnswers array to be the correct length. Next, all that is required is the clickable options (labels were used in this scenario) in the HTML are configured to call the question function with the question number it is in reference to on click. Finally, a button connected to the submit function and an element with the id of “result” will be edited to provide feedback.

4. Drag and Drop

First, swap the background image accordingly to the one required for the slide; this can be done within “style.css” file.

```
background-image: url("img/neuron.jpg");
```

Second, change the location and size of the drop boxes by changing the absolute pixel percentages, found in “index.html” under the div with class “question-container”. In the code below, I have bolded the aspects that need to be modified.

```
<div id = "#1" class = "dropBox" style = "top:16.8%;
 left:28%; width:25%; height:4%">#1</div>
```

One thing to note is that this user story only works on mouse events. It will not work on mobile touch events, so a few lines of code will need to be added to the “js/drag_and_drop.js” file in order to account for this.

5. Image Ordering

Simply reuse the “sorting.js” file and refer to “index.html” to coordinate element IDs, as each image element will have a corresponding ID indicating the correct order. The code for the image ordering slide expects five images to be ordered, and each image will have an ID such as “first”, or “second”, indicating their correct order in the entire lineup of images. The code for animating how an image snaps to the cursor, gets relocated to a new place in line, and bumps all images to the right one space over, can all be replicated exactly to simulate the animation effects.

8. Logistics

a. Team Roles

i. Product Owner: Charles Rothrock

ii. Scrum Master: Ben D’Antonio

b. Our Project’s Important Links

i. Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2435185

ii. GitHub: https://github.com/cwrothrock/Animations-in-Stress-Learning-Content

iii. Project Video Demo: https://www.youtube.com/watch?v=Ous7CgI0jhw

c. Other Important Links

i. One Health Modules: https://vetmed.tamu.edu/peer/one-health/

ii. StepStone Editor: https://stepstonelearning.net/

d. Contacts

i. Daniel Shuta, StepStone developer: dshuta@cvm.tamu.edu

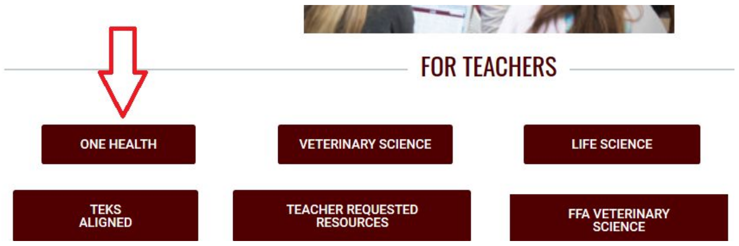
Tutorials

1. **Tutorial 1: What is StepStone?**

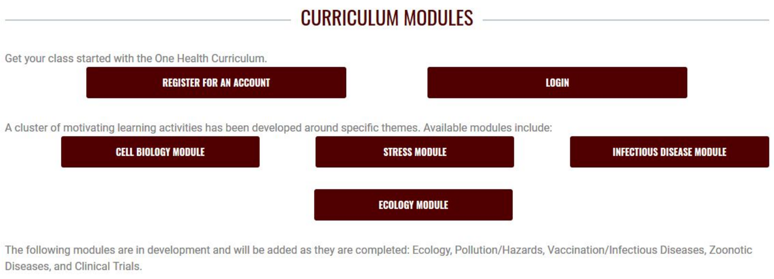
   StepStone is the learning environment system created by the Peer Program. The relevant web pages and steps for reaching the StepStone modules are as follows:

   1. Visit the Peer Program home page: https://vetmed.tamu.edu/peer/
   2. Near the middle, click the button that says *One Health.*

      The direct link is: https://vetmed.tamu.edu/peer/one-health/



   3. On the page, there is a section titled *Curriculum Modules*. This is the production environment where, depending on the module you work on, your mini-apps will be placed. No development is going to occur on this site, but you will go here to view your apps in action or determine where they are going to be placed.

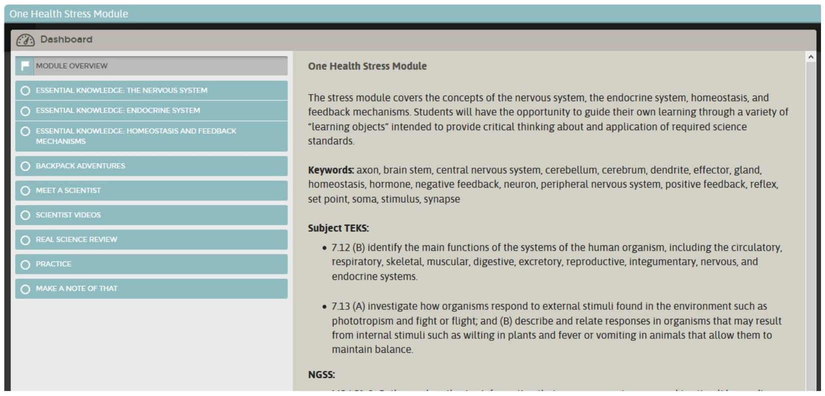

CURRICULUM MODULES

Get your class started with the One Health Curriculum.

A cluster of motivating learning activities has been developed around specific themes. Available modules include:

The following modules are in development and will be added as they are completed: Ecology, Pollution/Hazards, Vaccination/Infectious Diseases, Zoonotic Diseases, and Clinical Trials.

   4. When you click on a module, you will be redirected to the StepStone learning environment and can then navigate it using the Dashboard and internal paths.



One Health Stress Module

The stress module covers the concepts of the nervous system, the endocrine system, homeostasis, and feedback mechanisms. Students will have the opportunity to guide their own learning through a variety of "learning objects" intended to provide critical thinking about and application of required science standards.

Keywords: axon, brain stem, central nervous system, cerebellum, cerebrum, dendrite, effector, gland, homeostasis, hormone, negative feedback, neuron, peripheral nervous system, positive feedback, reflex, set point, soma, stimulus, synapse

Subject TEKS:

- 7.12 (B) identify the main functions of the systems of the human organism, including the circulatory, respiratory, skeletal, muscular, digestive, excretory, reproductive, integumentary, nervous, and endocrine systems.
- 7.13 (A) investigate how organisms respond to external stimuli found in the environment such as phototropism and fight or flight; and (B) describe and relate responses in organisms that may result from internal stimuli such as wilting in plants and fever or vomiting in animals that allow them to maintain balance.

NGSS:

2. Tutorial 2: How to Test Your Apps in StepStone

As of 5/2/2020, there is no way for you to stick our apps into the production environment; Daniel Shuta must do this for you. We are aware that Daniel is hard at work creating a method so that you CAN upload your apps yourself. Regardless, this tutorial explains how you can put your apps into a testing environment that Daniel provides.

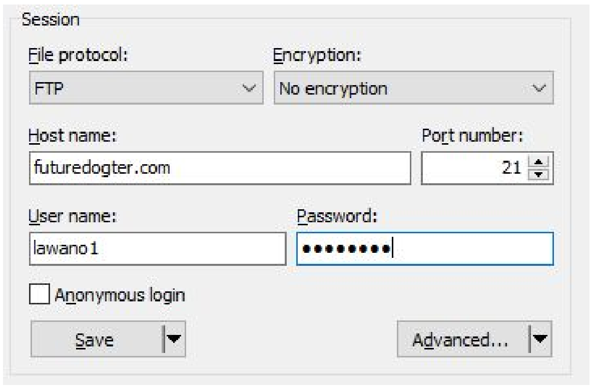
1. Download an app capable of establishing an FTP connection such as WinSCP or FileZilla. I used WinSCP, so this tutorial may have steps that won't match exactly with other clients.
2. Use the following credentials:

   `Server/Host: futuredogter.com`

   `Username: lawano1`

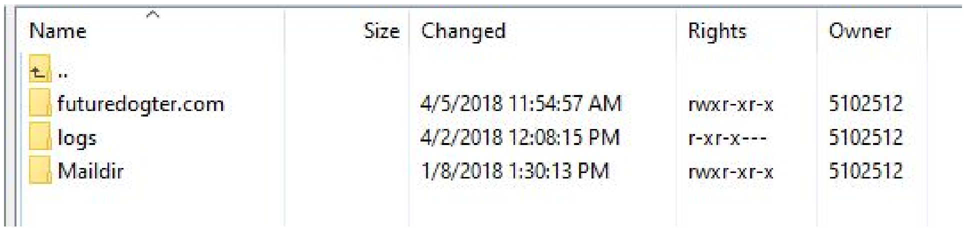
   `Password: H-8Uqp5Z`

   In WinSCP, establishing a new session should look something like this:

   Session
   File protocol: FTP
   Encryption: No encryption
   Host name: futuredogter.com
   Port number: 21
   User name: lawano1
   Password: ●●●●●●●●
   Anonymous login: ☐

3. Once you are connected, you should see a directory that looks something similar to this but it should always contain a `futuredogter.com` directory.

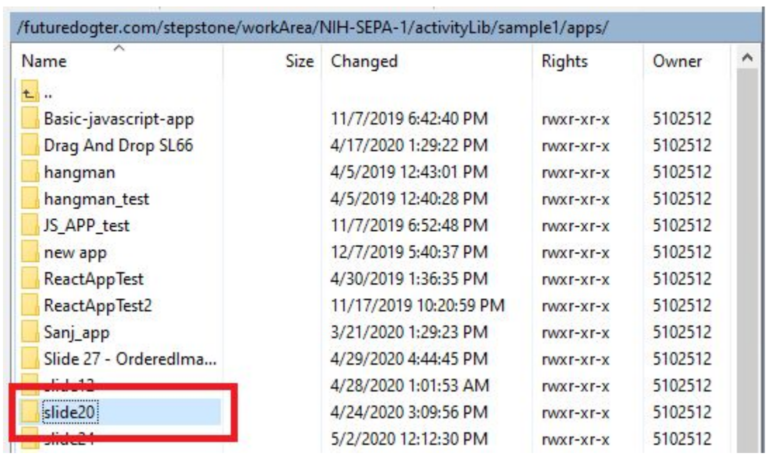
   

Name	Size	Changed	Rights	Owner
..				
futuredogter.com		4/5/2018 11:54:57 AM	rwxr-xr-x	5102512
logs		4/2/2018 12:08:15 PM	r-xr-x---	5102512
Maildir		1/8/2018 1:30:13 PM	rwxr-xr-x	5102512

   Now, navigate to the following path:

   `futuredogter.com/stepstone/workArea/NIH-SEPA-1/activityLib/sample1`

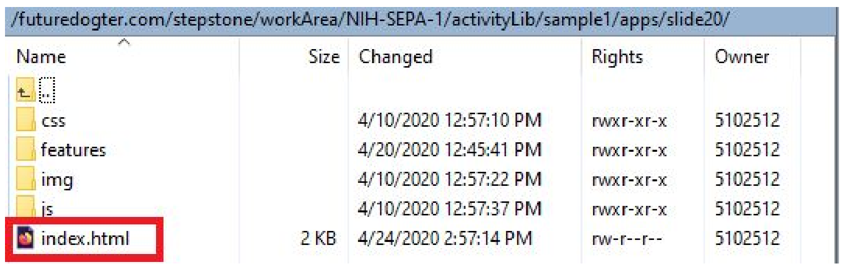
4. This directory should contain three subdirectories titled "apps", "json", and "media". Whenever you want to test out your standalone mini-app, upload the entire app contents into the `/apps` directory as a subdirectory.



/futuredogter.com/stepstone/workArea/NIH-SEPA-1/activityLib/sample1/apps/

Name	Size	Changed	Rights	Owner
..				
Basic-javascript-app		11/7/2019 6:42:40 PM	rwxr-xr-x	5102512
Drag And Drop SL66		4/17/2020 1:29:22 PM	rwxr-xr-x	5102512
hangman		4/5/2019 12:43:01 PM	rwxr-xr-x	5102512
hangman_test		4/5/2019 12:40:28 PM	rwxr-xr-x	5102512
JS_APP_test		11/7/2019 6:52:48 PM	rwxr-xr-x	5102512
new app		12/7/2019 5:40:37 PM	rwxr-xr-x	5102512
ReactAppTest		4/30/2019 1:36:35 PM	rwxr-xr-x	5102512
ReactAppTest2		11/17/2019 10:20:59 PM	rwxr-xr-x	5102512
Sanj_app		3/21/2020 1:29:23 PM	rwxr-xr-x	5102512
Slide 27 - OrderedIma...		4/29/2020 4:44:45 PM	rwxr-xr-x	5102512
slide12		4/28/2020 1:01:53 AM	rwxr-xr-x	5102512
slide20		4/24/2020 3:09:56 PM	rwxr-xr-x	5102512
slide21		5/2/2020 12:12:30 PM	rwxr-xr-x	5102512

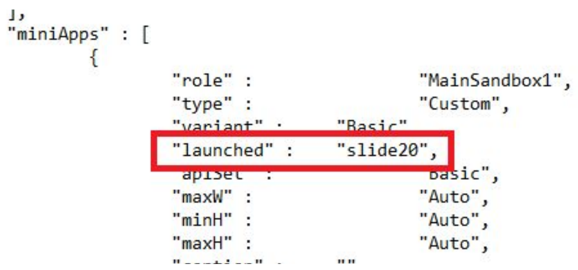
Be sure your specific app folder contains an `index.html` file at the root as well as any necessary JavaScript, CSS, and even other HTML files. It has been advised by Daniel that you shouldn't try to pull scripts externally, meaning any scripts you need to include for your app should be downloaded and included in your app's folder.



/futuredogter.com/stepstone/workArea/NIH-SEPA-1/activityLib/sample1/apps/slide20/

Name	Size	Changed	Rights	Owner
..				
css		4/10/2020 12:57:10 PM	rwxr-xr-x	5102512
features		4/20/2020 12:45:41 PM	rwxr-xr-x	5102512
img		4/10/2020 12:57:22 PM	rwxr-xr-x	5102512
js		4/10/2020 12:57:37 PM	rwxr-xr-x	5102512
index.html	2 KB	4/24/2020 2:57:14 PM	rw-r--r--	5102512

5. Once you've uploaded your app's folder into the `/apps` directory, go into the `/json` directory. You should see multiple JSON files, one of which is titled `draggybox40.json`. To view your app in a StepStone testing environment, open `draggybox40.json`, and edit the launched app to be the title of your app's folder. Remember, this folder must have an `index.html` file at the root.

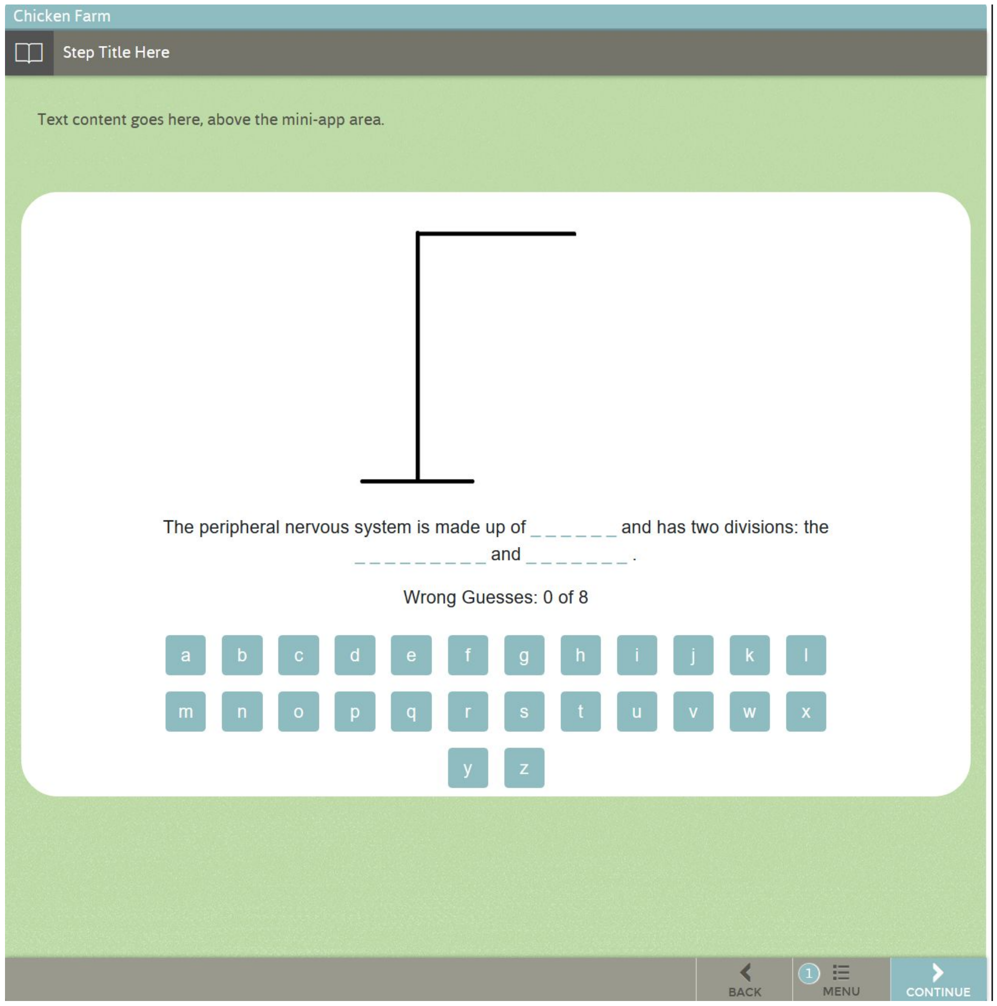


```
],
"miniApps" : [
        {
                "role" :                "MainSandbox1",
                "type" :                "Custom",
                "variant" :     "Basic",
                "launched" :    "slide20",
                "apiSet" :              "Basic",
                "maxW" :                "Auto",
                "minH" :                "Auto",
                "maxH" :                "Auto",
```

6. Now you're ready to view your app in the testing environment. Visit the following link: http://www.futuredogter.com/stepstone/playerShell.php?org=CET&sys=public.Late

st&pool=TAMU-CET-1&resourceloc=www.futuredogter.com&resourceavatar=NIH-SEPA-1&resource=sample1&ppj=1_1_40

7. At that link, you should see your app embedded into the frame. You can disregard the titles and whatnot, as your app has no control over the page and is only responsible for itself. You can use this testing environment to repeatedly launch your app and make small adjustments so that Daniel doesn't have to reapply your app to production numerous times. Note that we had some loading and responsiveness issues in this environment when switching launched apps, so be sure to clear your browser's cache if you experience these types of issues.



Text content goes here, above the mini-app area.

The peripheral nervous system is made up of _ _ _ _ _ _ and has two divisions: the _ _ _ _ _ _ _ _ _ and _ _ _ _ _ _ _ .

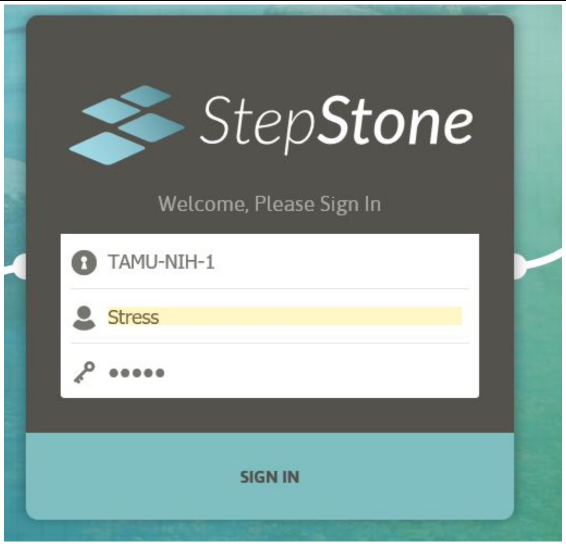
Wrong Guesses: 0 of 8

3. **Tutorial 3: Editing The StepStone Production Path**

   Before you email Daniel to add your apps into the production environment, you first need to edit the live StepStone paths that dictacte where your app is going to go on the slide.

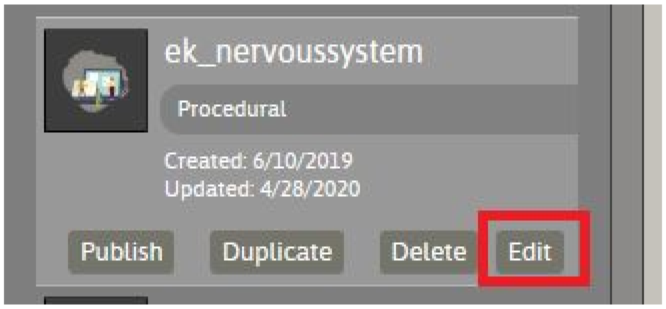
   1. Go to the StepStone learning path editing website: https://stepstonelearning.net/
   2. Log in with whatever credentials match your project. The account portal for all modules is `TAMU-NIH-1`

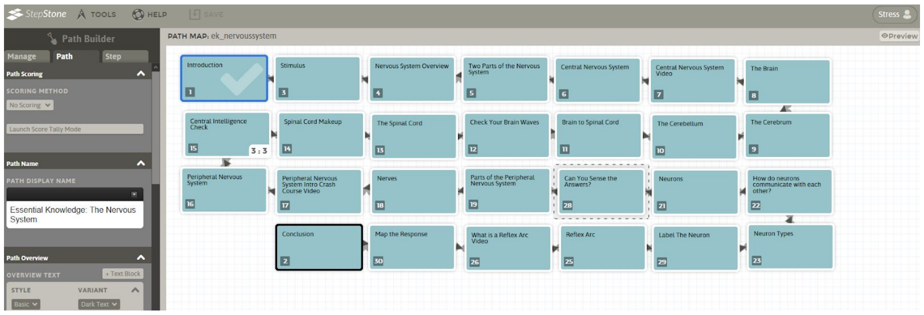
Module Name	Username	Password
Cell Biology	Cells	PEER1
Stress	Stress	PEER2
Ecology	Ecology	PEER3
Clinical Trials	ClinicalTrials	PEER4
Infectious Diseases	InfectiousDiseases	PEER5



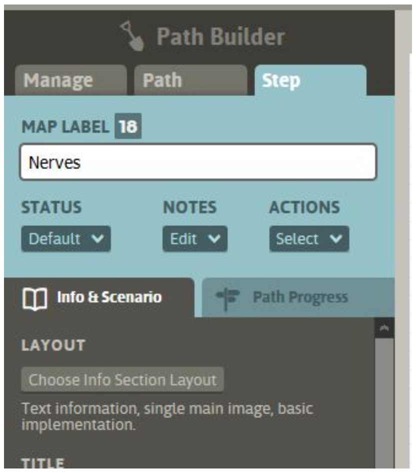
TAMU-NIH-1
Stress
●●●●●

   3. This is where things can get a bit confusing. Once logged in, you should by default be at the Path Builder Tool. For your module, this will contain all of the relevant paths that are used by the module in production. If you choose a path and click ‘Edit’, you will be shown all of the internal steps that the path contains.

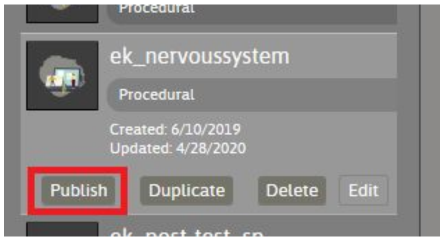




4. Each step, displayed as the blue boxes, is linked to its previous and next steps. To edit a specific step, click the step tab on the left of the page. This will display all of the relevant information (title, text contents, media, etc.) for that specific step. In StepStone, this will be one single slide in a path of the module. Use this feature of the tool to change anything you want about the step your app will go on, from text contents to background. The most likely occurrence will be that you want the slide to be completely empty because your mini-app SHOULD be completely independent and contain all information and interactions necessary on the slide. In that case, you might find it easier to simply delete the step from the path altogether and insert a new, completely empty step in its place.

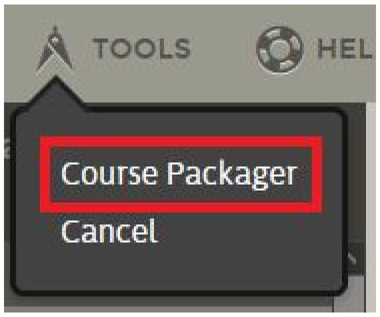


5. Once you have all of your steps edited for the current path and your changes saved (which should be required by the system whenever you switch steps), you need to publish the new path. Go back to the ‘Manage’ tab and click ‘Publish’ for the path

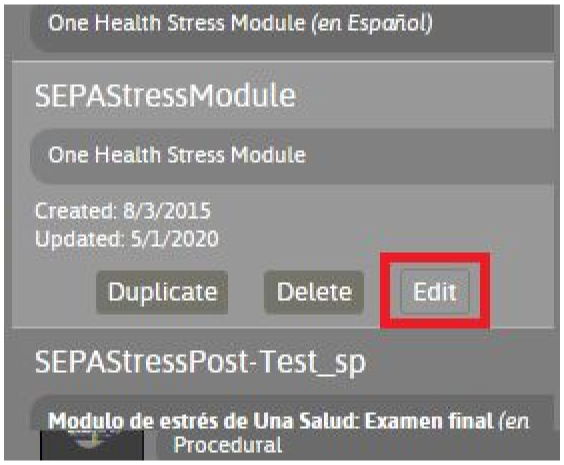


you've just edited. You will be asked for an ID for your new publication. Give it a unique ID, different from any of the IDs published for this path previously. If you try to name it a non-unique ID, the section will flash orange.

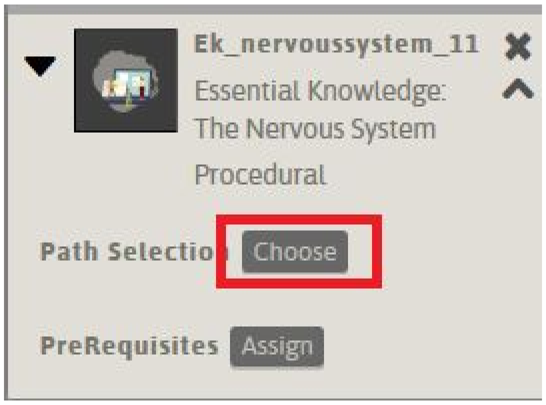
6. With your path published, go to the Course Packager tool.



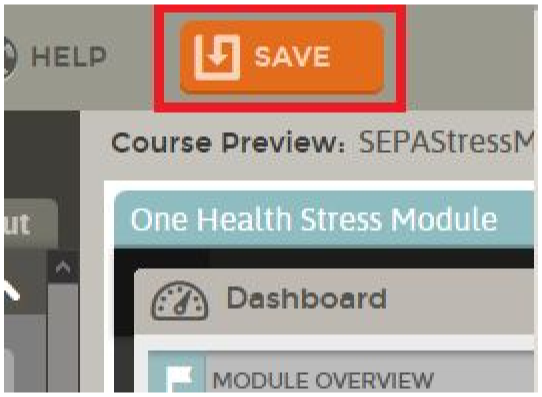
7. In the list of courses, click on the module you've edited paths for, and click edit.



8. In this course's 'Paths' tab, find the path you made changes to, select its dropdown, and click 'Choose' to select a new path for the course.



9. After selecting the ID of the path you published in Step 5, click the save button at the top of the page to apply this selection to production.



10. Now, you should see whatever changes you've made appear in the live corresponding StepStone module. At this point, you can email Daniel Shuta and tell him that your app is ready to be applied. The general procedure for doing that is:
    a. Have your most recent app version uploaded to the apps folder on the FTP server.
    b. Tell Daniel where the app should go; we followed the format:
    (app directory name) : (path name in StepStone) : (step/slide title)

4. Tutorial 4: Local Automated Testing for JS Apps

Since it is purely low-level HTML, CSS, and JavaScript-based apps, we used development-dependent node modules. If you'd like to set up your own automated testing environment like ours, you can follow the simple steps:

1. Install NodeJS
2. In the root directory for your app, run
   ```
   npm init
   ```
3. Install the necessary development modules:
   ```
   npm install --save-dev cucumber selenium-webdriver chai
   ```
   Depending on the browsers you want to test with, you can install the following modules:

Google Chrome:	`chromedriver`
Microsoft Edge:	`edgedriver`
Firefox:	`geckodriver`
Internet Explorer:	`iedriver`

4. It's important to include the `/node_modules` in the gitignore file so that these dependencies are not tracked by GitHub.
5. It is also important to note that the "production" version of your mini app should not include any of the testing files and dependencies
6. Be sure you have the latest version of the given browsers or else Selenium will throw errors.
7. Write your Cucumber features in `/features/my_feature.feature`
8. Write your step definitions in `/features/support/steps.js`
9. Follow the selenium tutorial: Selenium WebDriver JS API
10. Remember to use `async` and `await` in your step definitions/world functions so that your code can finish execution during steps/document manipulation. Here is more documentation for JavaScript Promises and Async/Await.
11. In `package.json`, you can write a test script to run with `npm`:
    ```
    "scripts": {
           "test": "./node_modules/.bin/cucumber-js"
    }
    ```
    Then, on your command line at root directory:
    ```
    npm run test
    ```
12. Valuable References
    1. Cucumber-js: https://github.com/cucumber/cucumber-js
    2. Selenium WebDriver: https://www.npmjs.com/package/selenium-webdriver, https://www.selenium.dev/selenium/docs/api/javascript/index.html
    3. Chai: https://github.com/chaijs/chai
    4. Our GitHub (referencing our file structures and code will likely be the most helpful): https://github.com/cwrothrock/Animations-in-Stress-Learning-Content

5. Tutorial 5: What is the iframe Resizer?

The iframe resizer is a JavaScript library that is used by StepStone to automatically maximize the height of iframe shells within each slide. The parent StepStone structure already implements one of the crucial library files. Your job when creating your mini-app is to include the second resizer responsible for resizing iframe contents. The JS file you must include is titled "iframeResizer.contentWindow.min.js". This file can be found in all of our apps, generally under the "js" directory. You can also find it at the library's GitHub:

https://github.com/davidjbradshaw/iframe-resizer

As per Daniel Shuta:

> "You can place it anywhere in your mini-app's structure, as long as it is invoked at run time to communicate with the paired parent file (I've been calling it right after the jQuery inclusion in the index.html head code of my mini-apps)."

Most importantly, the way you choose to design your app and implement CSS can affect the interaction of the app with the resizer. Be sure to thoroughly test your app using the FTP server and ensure this script is included properly within your app. Again, you can refer to our GitHub for examples, but it's pretty straightforward. Daniel was also incredibly helpful and responsive throughout the semester we worked on our apps.

Correspondence

This section represents a compilation of all significant correspondence between our team, Daniel Shuta, Dr. Walker, and even previous teams that worked on relevant projects. In the event you still have unanswered questions, hopefully some of this information will prove useful.

From Dr. Walker
Unnati Eramangalath eunnati@tamu.edu worked on the cell biology interactions last semester and can be a helpful resource.

From Daniel Shuta, January 24th 2019
suggestions/rules for their mini-app development:

- In StepStone Player, mini-app is contained within a sandboxed html5 iframe (with "allow-scripts" enabled.) Only one mini-app is allowed per-step (but the scope of that mini-app can be as complex as needed, so a single mini-app could of course behave like a mini-portal to multiple mini-app experiences.) There is also an optional media component (static display, launcher, grid, etc. — sits above the mini-app iframe) and optional step assessment (sits below the mini-app iframe), if desired. Mini-app currently does not communicate with StepStone Player outside of its own sandbox shell, except for a couple of ADA-focus actions via sandbox message events (these are still being reviewed/adjusted, so I can share them later.) Thus, it is currently just an activity on the page, not linked to the step's assessment or scoring or progress tracking, etc. (similar to media on the page — learner is not forced to interact, nor does the player system "know" what they've done with it.)

- Each mini-app should be set up in a uniquely-named "master directory" for the app, within that directory should be an index.html file and all other folder trees / files. When a visited step invokes a mini-app, StepStone Player will look for the intended master folder name and this "root-level" index.html as the default target file. The index file can redirect or whatever is needed, but that is what the systems seeks to initialize the mini-app when the step is loaded.

- All files in a mini-app directory should be client-executable materials (html, css, js, media files.) No server-side executable stuff like php.
- All js libraries (such as jQuery) should be (at this time) hard-included with the application's other files, preferably minified for distribution. I haven't done much testing reaching out to CDNs or GitHub, etc. for external code libraries — that might work or might not based on configuration(s), and our security measures are still in flux. The sandbox encapsulation method that I'm using is kinda strict at the moment so I would expect best results from all necessary files being included somewhere within the app's directory structure.

- Responsive design should be considered. Should build/test application with a narrow device width in mind (but also other sizes!). StepStone sets the mini-app within a full-width iframe container (centered), and adds a bit of padding on the sides (matching other content on the step) so it won't be full-viewport. We can discuss optimizing/forcing dynamic sizing if you have trouble getting a satisfactory fit in the iframe. If the mini-app "innards" do not horizontally fill the allotted real-estate, it will "float" nicely in the center of the page.

- For mini-apps that i've built thus far, I've been including this handy (MIT-licenced) library within each mini-app for automatic height maximization of the mini-app iframe shell : https://github.com/davidjbradshaw/iframe-resizer. This library consists of 2 files, one for outside of the iframe, and one for the iframe's contents. The result is a css height style is applied dynamically to the iframe shell, determined by the iframe's contents. The "parent" file of the library pairing is already included with the current StepStone Player backbone code. I am attaching the minified contentWindow js file for your convenience. You can place it anywhere in your mini-app's structure, as long as it is invoked at run time to communicate with the paired parent file (I've been calling it right after the jQuery inclusion in the index.html head code of my mini-apps). This is of course optional, you are welcome to "roll your own" height controls.

- The mini app is still sort of a "proof of concept". We don't really have a "smooth" method for authors to get their application into our server environment at this time. Currently I have to get their folder of files/subfolders and manually place it in the StepStone path on our server. I can probably devise some (hopefully safe) way to allow a nicer incremental development process, to see the live mini-app within StepStone Player.

- Multiple, individually-sandboxed instances of the same mini-app can be applied in a single path, if desired (say, if you wanted different narrative branches that a learner might choose to all expose the same mini-app.) There is currently no limit to the amount of mini-apps that can be included with a path.

From Daniel Shuta, March 4th 2019
OK, here's the info for ftp access:
Server: futuredogter.com
User: lawano1
Pass: H-8Uqp5Z

Once you are on the server, you should see a "stepstone" folder (might be inside the futuredogter.com parent directory, depending on your ftp client).

You want to get to stepstone/workArea/NIH-SEPA-1/activityLib/sample1/ . Inside of this directory, there is an "apps" folder. this is where you will place your mini-app stuff for testing/preview. The app currently in there, emrsFormSet, is an example of a mini-app that is loaded into the s.s. Player. You can directly preview this mini-app, in context, by visiting the url:
http://www.futuredogter.com/stepstone/playerShell.php?org=CET&sys=public.Latest&pool=TAMU-CET-1&resourceloc=www.futuredogter.com&resourceavatar=NIH-SEPA-1&resource=sample1&ppj=1_1_40

You can change the contents of the emrsFormSet folder and the updated contents should load up in that player url upon refresh.

You should always have an index.html file in the root of this mini-app folder. The other folders in there (css, img, etc..) are specific to this mini-app and are not globally essential. The s.s. player just just needs to latch on to that initial index file when it looks for the folder name. That said, I'd recommend using that iframe resizing helper that I previously referenced, this should help keep your mini-app(s) making the most of the available screen real estate.

You can add as many other mini-apps as you like to that apps folder, but only one single mini-app can be loaded on a given step (in addition to text, media, step assessment.) To target an app on a step, you can change the name of the folder in the json file that points to the mini-app. In this case, in sample1/json/draggybox40.json, you will see the "emrsFormSet" folder name within the json file. Change this to target a different mini-app (within the same apps parent folder). If you need to create multiple modules for testing different mini-apps simultaneously, I can further describe the system setup for you to do that.

There is another directory in that workArea parent folder, "LAwano". This is another work stream folder that is currently in use by another team member, so please steer clear of this folder to avoid crossing the streams!

From Nicola Ritter, November 4th 2019
Hi Hank,
Changing from a procedural path to a case study path is not available at this time in the builder tool. However, our programmer can do this on the back end. To do this, we would need to know the account the path is in and the name of the path that you want converted from procedural to case study.

Please note, that when a case study path is used the table of contents at the beginning of the path is not available. Without the table of contents, students cannot skip around within a path. Torri, can you verify that this is how you'd like to proceed or if you'd like to change the design of what Hank's team is building.

From Daniel Shuta, May 8th 2019, regarding the use of React.js
I don't use ReactJS, but if ReactJS is like jQuery, you should be able to place the required minified js library files within your mini app folder. Does this thread help? :
https://stackoverflow.com/questions/53899444/i-need-my-reactjs-app-working-offline-without-server

I also don't currently use npm, but if there's a way to obtain the deployment files (scripts?) and place them in the mini app's content folder (and those aren't server-side executable), that should be ok. Is this thread helpful? :
https://stackoverflow.com/questions/43064107/how-to-install-npm-package-while-offline

---

From Daniel Shuta, March 18th 2020

I'm still developing a method to upload and apply miniApps within the full StepStone authoring system (as well as a step assessment miniApp option). There isn't yet a "clean" way to attach miniApps to Steps with the authorware.

The easiest way to go about developing and testing for now would be to upload your miniApps to the "apps" folder within the "sample1" directory in the workArea's activityLib. You can then swap out the miniApp that you want to test/review by changing the target miniApp "launched" value in the "draggybox40.json" file in that Path. Once changed, that miniApp should be viewable via the direct preview link noted in the March 4 2019 email (let me know if you don't have this.)

Will this work for you? Once you have your miniApps completed, I can assist with relocating them into the actual Paths/Steps where you intend for them to reside. Please remember that miniApps should only include client-side technology, so no server-side-executing script like .php, etc..

If you are comfortable navigating the ftp environment and editing json data, I can provide further details regarding placing the miniApps where you need them on your own (within the workArea environment.) It's a pretty straightforward process.

From Daniel Shuta, March 31st 2020
Hi, hope everyone is staying healthy and productive!

Currently, any "Mini-Apps" that you build/embed in StepStone modules will be "info-only", meaning that they will exist in a sandboxed state and will not significantly communicate with the StepStone system outside of themselves. The Step Assessment section of each step is the part that actually determines the learner's navigation path and narrative options, and this section is independent of the "info-only" components of each step.

Think of each step as a stacked sandwich :

==================

Step Info area (title + optional text, multimedia, sub-narratives)

===================

(optional) Step Mini App

===================

Step Assessment (either some sort of interactive decision-making piece, or a basic "pass-through" to the next target step)

===================

So, currently the Mini App can be used as an interactive or customized information "experience" that enhances the Step Info (and better informs a learner to make decisions on the respective Step Assessment), or you can *just* have a Mini App as the sole information aspect of a Step. It will not, however, affect the Step Assessment in any functional way.

All that said, I am currently developing a Mini App environment for the Step Assessment section that will allow custom interactive pieces exactly like the info-only Mini-App. It will be possible to have an info-only Mini App AND an assessment Mini App, per-step (or minimal info-only content and JUST an assessment Mini-App on the Step, presented to the learner as the entire Step content and decision stack.) The assessment Mini App will allow a convention-defined javascript post-message json-formatted assessment result to be communicated to the StepStone player shell, and you will thus be able to make customized assessments/quizzes that the built-in assessment variations do not provide. No ETA on completion of this new piece, but it is in active development now.

Regarding placing your finished Mini App code and components, I'm also developing a method of uploading a zipped package containing your Mini App(s) to a Path in the S.S. authoring tools, but for now I'll need to manually place your Mini Apps in the desired locations on the actual S.S. content server (you'll just need to tell me which Mini App and where it should be placed.) Once migrated, you'll be able to use the S.S. course/module packaging tools to wrap your Mini-App-bearing modules in SCORM packages and distribute as needed.

From Daniel Shuta, April 6th 2020
Hi Benjamin, please refer to my previous email from the 31st. Currently the Mini-App is considered a sandboxed, non-mandatory, non-enforceable, "info-only" experience for the learner. StepStone Player will not "track" or respond to interactions in the Mini-App. All learner interactions in the Mini-App must be tested against conditions that exist solely within the respective Mini-App.

I am developing an assessment variant for the Mini-App that WILL pass conditional/score data to the Player shell just like a "regular" Step assessment (and you will even be able to have an info-only Mini-App AND an assessment Mini-App if desired), but no ETA on this addition yet, unfortunately. Thus, currently you'd need to devise your Mini-App to display the animation sequence approval to the learner as you described, and then have a Step assessment that would independently require the learner to "confirm" a correct sequence (or otherwise separately test the learner's comprehension of the Mini-App "lesson".) For example, if your animation sequence has numeric or lettered indicators for each sequential component, you could have a series of selections in the Step's assessment like :

1.) A > B > C ,
2.) B > C > A ,
3.) A > C > B

… and the learner would, being informed of the proper sequence via interacting with the Mini-App, be able to select the proper sequence. The learner would not be mandated to interact with the Mini-App in order to perform the Step Assessment, regardless of whether or not the Step Assessment can be legitimately "solved" without engaging with the Mini-App (vs. guessing the answer.) It is not possible at this time to "require" that the Mini-App be engaged by the learner in order to operate the Step Assessment on the same step.

From Daniel Shuta, April 27th 2020
The common process is to use the Path Builder tool to create the Path(s) (using the “public space" link that you noted), “publish” the finished Path(s) to your author-pool shared area, then jump over to the Course (Module) Packager tool to assemble one or more published Paths into a complete Course/Module.

The direct preview links that I sent to you are for the already-packaged versions of the modules, in which I’ve manually placed the MiniApps carried over from your external workspace. As MiniApps are not part of the Path-building system, newly published Paths won’t contain the MiniApp embeds. We don’t have a method just yet of placing MiniApps using the authoring tools, so you’ll need to :

- make your desired changes in the Path Builder.
- using the Manage tab in the Path Builder, re-publish your updated Path(s) (must repub. with a different ID/name than the original published version, the publish controls will flash orange if you are attempting to publish with an identical ID).
- re-package the newly published Paths in the Course Packager (either making new Course(s) for the updated, published Paths, or just point to the new Path iterations in the existing Courses.)
- once you have the updated Course packaged/saved, send me a notice and i’ll re-apply your Mini-Apps in the new/modified Course. Just let me know the target Course/Paths/Steps after your updates.

I’m developing a method of uploading .zip files that are automatically unpacked via the authoring tools for the purposes of uploading/managing these MiniApps without my intervention, but for now i’ll need to assist.

Let me know if you need further clarification!

From Daniel Shuta, April 28th 2020
I’m still seeing some placement and loading issues. some suggestions :

- don’t reach out for external components/libraries, any javascript libraries etc. that you require should be placed inside of the respective app’s folder in your workspace. reaching out to a cdn, etc. should theoretically work (we regularly use vimeo and youtube in these modules, of course!) but keeping as much as possible local will help isolate problems out of the equation.

- within your mini-app code, perform an initial sizing routine on document ready or some other stage of loading the mini app to maximize your needed vertical space for the given viewport size. this will help trigger the suggested/provided helper iframeresizer lib. to vertically fit the mini app iframe contents to the iframe container once the iframe contents are loaded. the exact methods and components that need sizing applied will depend on the mini-apps design, of course — i always employ some combination of css and javascript (absolute positioning, forcing total height after summing pertinent DOM element heights, etc.), depending on what needs sizing.

- be sure to test the mini apps on the workspace server within the sample stepstone path/player, not just the mini-app directly, itself — when embedded in the sample path on the workspace, the mini-app should function exactly the same after it is carried over to the main server, so ensure it is behaving/sizing/loading as expected in workspace first.

From Dr. Walker, April 20th 2020, regarding testing with this type of project
What I told the other teams is that since this environment is funky, you use whatever test process you can manage. In theory you should be able to run Jasmine, and there is a Cucumber-js. I suggest trying, but I wouldn't be surprised if it doesn't work. You should be able to use Capybara, since it is essentially a scripted web browser, automating your manual testing of the code.

Teams last semester didn't really do anything other than manually testing.

So in summary, do the best you can, but I recognize that you will not be able to have a fully automated continuous integration environment. **And please, please, please carefully document what you do, so that teams in future semesters can learn from it.**