Biochemistry Virtual Lab Software

Team 12
Client: Professor Stone Chen
Faculty Advisor: Professor Simanta Mitra
Team Members:
Peter Bancks
Jacob Christopherson
Enzo Ciccarelli-Asta
Steven Dirth
Romain Ndoutoume
Brody Sunsten

Revised Project Design

Evolution from 491

Project Scope at End of 491

- Web-based software providing virtual Biochemistry lab modules for BBMB 102.
- 3D lab environment featuring interactive lab equipment that is used to complete the same lab procedures as the physical lab.
- Multiple lab modules allowing for different types of lab procedures to be executed.
- Connectivity with Okta SSO and Canvas assignment submissions to track and grade student progress.
- Lightweight resource focus to ensure better accessibility from a broad range of web browsers and computer specifications.

Project Scope Focusing

After reviewing our initial design for the project, we determined that a reduction in the scale of our scope was necessary to fit the time and resources we had available for this semester. During discussions with our client, we refocused our efforts on the features they identified as being most important.

- **Enhance Virtual Lab Experience:** Usability feature requests proposed by our client were added to our scope, such as alternative camera controls and refactoring the current controls to be smoother.
- **Lab Module 3 Functionality:** We reduced our scope from finishing multiple lab modules to finishing only the largest of the lab modules. This allowed us to focus more on providing a better user experience within the 3D lab.
- **No Canvas Assignment Functionality:** Connectivity with assignments in a Canvas class was removed from the scope. Testing this function would require access at the course administrator level to a fake canvas course. Between the setup requirements and the payoff of this feature, we determined that development time would be better utilized by focusing on other features.

Requirements & Standards

Functional Requirements

- 3D environment: User can see and interact with a 3d virtual lab, including objects, tables, PPE, and a notebook.
- Object Interactions: Objects should act according to the module/experiment script to demonstrate interactions. Objects that are not related to the current step will not be interactable.
- Notebook: User should be able to record data necessary for the experiment, graph the data they find, and read instructions and feedback about the experiment.
- Scripted Events: Upon the completion of specified actions, a specific event or reaction will occur, and the experiment will either progress or end.

- User Progress: The user should be able to sign in using Okta SSO. If a user completes a lab, their progress should be saved.

UI Requirements

- Must function on all major web browsers
- UI should direct the user to experiments as intended
- Scripts within modules should be error-free and linked to scripted events
- User can interact with environment smoothly

Standards

- SCRUM – our team roughly followed the SCRUM guidelines to complete the project. We meet with our client once a week to discuss our project. Additionally, every two weeks we hold a demonstration of the project and plan our goals for the next two weeks. We use GitLab's integrated Kanban board to track our progress.
- Git Feature Branch Workflow – our team follows the Git Feature Branch workflow, creating a new branch for each feature. These feature branches are then reviewed and merged into the master branch once completed.
- React Eslint – We use the default eslint configuration for react applications created by create-react-app. This enforces naming conventions and consistent indentation, among other things, making it much easier to comprehend the codebase.

Engineering Constraints

- User hardware – The simulation should be able to run smoothly on as many hardware configurations as possible. As such, our team needed to be aware of the performance of our software, as it is likely that many users would be running it on a laptop or lower-end PC. To work around this, we developed 3d models with performance in mind (using lower poly-counts), and attempted to write efficient, lightweight code.
- Network Capabilities – We were aware of some discrepancies in our network speeds while working on the project. Because of this, we needed to ensure that our simulation could run as fast and responsively as possible even on slow network speeds. This was a relatively simple issue, as the toolset we chose to use (ThreeJS) only requires an initial connection before it can run a locally cached version in the browser. Therefore, simulation performance is not bottlenecked by the network as the simulation itself doesn't require any data transfer to update.
- Toolset Constraints – Our team also ran into some issues while using the various toolsets we chose for our project. For example, ThreeJS seemed to separate any 3D models into separate models by the material that was applied to them. This meant that objects that were clicked and dragged could split apart. We also ran into some issues with how ThreeJS and react handles the scroll wheel, as well as various camera, texturing, and CPU usage capabilities. Some of these issues are still a work in progress.

Security

Our final design has very few security concerns as the website will be static (i.e., the software runs entirely client side). We do not store any user data, nor is there any application data the user could access maliciously to cause harm to the system. However, the user could potentially manipulate the webpage to display an invalid state.

While it is currently hosted on a virtual machine, it would be relatively simple to deploy to any static web hosting service, and any COTS tool for preventing DDoS (Distributed Denial of Service) attacks could be used, such as CloudFlare DDoS Protection.

Implementation Details

Yarn

Yarn is a package manager used for development. It allows us to integrate useful libraries into the project and automatically handle dependencies. It also lets us create optimized builds of the project for production and publication.

React

React is the framework used to create the site's UI. It has an expansive library of forms and buttons for handling input, as well as tools for page navigation.

Three

Three is the open-source library that handles 3D graphical calculations. It allows us to set up a scene that contains cameras, lighting, and objects. It also allows transformations of those objects, and interaction with the mouse and keyboard.

React-Three-Fiber

React-Three-Fiber is the framework used to link signals between React and Three. It allows us to have an interactive laboratory, where objects are moved in a 3D environment, and the UI responds accordingly.

React Testing Library

React Testing Library is a light-weight solution for testing React components. It provides light utility functions on top of react-dom and react-dom/test-utils. The utilities this library provides facilitate querying the DOM in the same way the user would.

Testing Process and Results

React Testing Library

We use the react testing library to test the logic of the web application. Because we have been working on module 3 only, all the tests we wrote are only designed for the Module 3 laboratory. In module 3, we test the lab's four steps that need to be completed by the students. The testing library helped us test that the next step is rendered correctly after successfully completing the previous step. With access to ReactDom, we were able to simulate input events to ensure that the code is working as intended.

Result: Most of the tests we wrote are passing, which means the code is behaving correctly. However, we have some failed tests that are all related to importing Three js. 3D web applications are very new, and many testing libraries don't support them now. At this time, we are testing our 3D environment manually on the browser, and we are using react testing library to test the 2D part of the application.

Manual Testing

Once each team member has completed the implementation of a feature, we manually test the feature locally on the web browser to ensure correct behavior. In this process, we play the role of an end-user and are looking for defects that may prevent the user from having a good user experience. If defects are found, we make sure to fix the issue, and then we test the application again. This process is repeated as needed until the application works as it is supposed to.

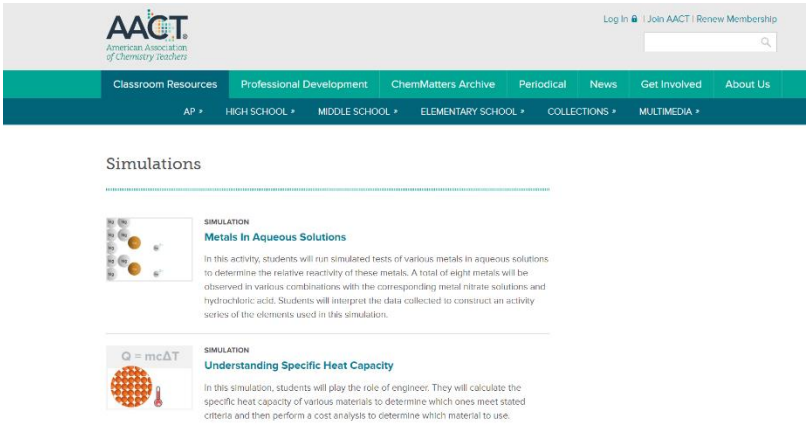
Acceptance Testing

In this testing phase of the software, we get client approval of the work we have done in the past sprint. A 'pass' in this stage ensures that the customer has accepted the current software and is ready to move onto the next iteration. For that, we hold demonstrations of our current progress every two weeks to ensure the project is advancing in a manner acceptable to the client. We also meet with our client weekly to ensure our specifications are as up-to-date as possible and any deviations from the original design are approved. Our team carefully reviews any feedback from the client and the faculty advisor, and we make sure by the next sprint, the issue is resolved.

Related Products and Literature

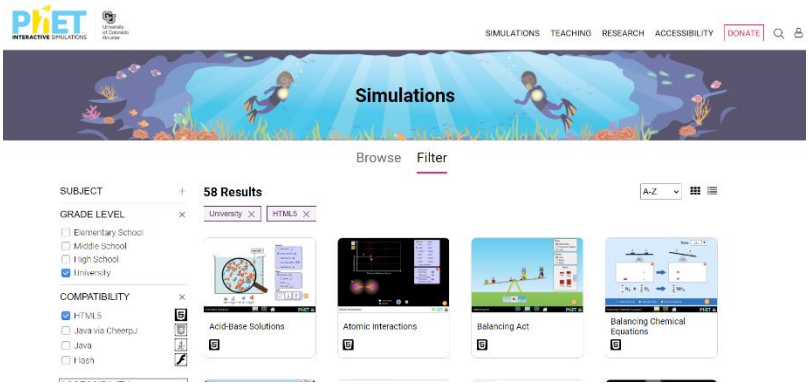
Related Products

AACT: https://teachchemistry.org/classroom-resources/simulations



Contains multiple 2D chemistry simulations and videos explaining and showing other labs created by the American Association of Chemistry Teachers. The units this website shows are for students in Elementary School, Middle School, High School, and a few for AP classes.

PhET: https://phet.colorado.edu/en/simulations/

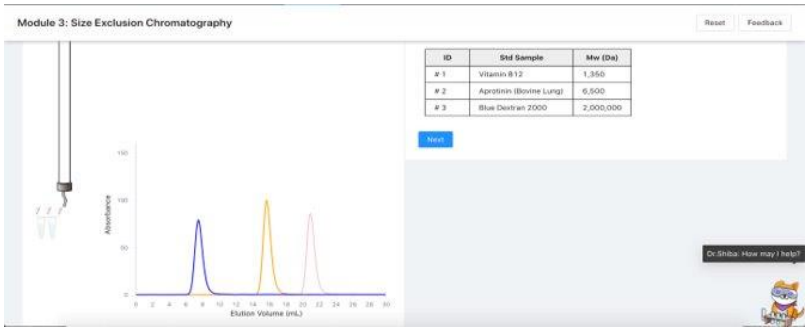


This website created by PhET (Physics Education Technology) allows the user to download or run imbedded code to complete short physics and chemistry examples and labs. Unlike the AACT website this emphasizes University and high school level resources, and can be filtered for compatibility, grade level (elementary-university), accessibility, and language.

An important thing to note is that there are few to no free interactive 3D simulations. Most of the online resources are either interactive 2D short simulations or are step by step labs with videos that are not interactive.

2D Version

BioChem 102: https://mitraresearch03.cs.iastate.edu/



There is a sister project where the lab is done within a two-dimensional website environment. Some of the calculations within our version of this project take inspiration from here at the offering of our client and the developers of this website.

Literature

"Using online simulations to teach biochemistry laboratory content during COVID-19"
Biochemistry and Molecular Biology Education, Volume 48 Issue 5

The University of South Australia moved to online learning over March of 2020 and ceased face-to-face teaching before the students had finished their in-person lab practicals. Luckily for them they had been developing and using online simulations for the course in question, Biochemistry (a 2nd year undergraduate course), since 2013. This article goes over some of the simulations, student feedback, and the practical scores resulting from the online learning. While short, it does reference the papers from 2012 from when the simulations were being developed and 2020 from a re-look at the findings in more recent years.

Appendix I – Operation Manual

Install Prerequisites

1. Install Node LTS: https://nodejs.org/en/download/
2. Install the yarn package manager: https://yarnpkg.com/getting-started/install
3. Install git LFS: https://git-lfs.github.com/

Clone the Repository

Note: Please ensure you have installed git LFS prior to cloning the repository! Some files will not download properly without it.

```
git clone https://git.ece.iastate.edu/sdmay-21-12/biochemistry-virtual-lab.git
```

OR

```
git clone git@git.ece.iastate.edu:sdmay-21-12/biochemistry-virtual-lab.git
```

Install Dependencies

In the project directory, run

```
yarn
```

Note: this will take quite a long time to resolve, download, and install all dependencies for our project

Run the Application in Development Mode

In the project directory, run

```
yarn start
```

to start the development server. You can then visit http://localhost in a browser to view the running application (this should open automatically).

Note: if you would like the development server to run on a different port (e.g. 8080), you can specify the port number by creating a file named ".env" in the project directory with the contents "PORT=8080". More advanced configuration settings are listed here: https://create-react-app.dev/docs/advanced-configuration/.

Compile the application for deployment to a static web server

In the project directory, run

```
yarn build
```

to create a production-ready build of the project. The final website will be in the build directory, which you can then serve with any static webserver. We provide a quick and easy example utilizing serve (https://github.com/vercel/serve#readme):

```
yarn global add serve
```

```
serve build
```

Appendix II: Alternative Designs

We briefly considered using a different UI Library, React Material UI – used by the 2d lab - but decided to use PrimeReact due to members experience with PrimeReact.

We considered a desktop application but decided against it due to the difficulty of supporting multiple platforms.

We attempted to host the website on Microsoft Azure, but due to billing requirements we opted to utilize a VM on the Iowa State Servers instead. This added an extra step to accessing the virtual lab, requiring users to use a VPN to connect to Iowa State's network if they were working remotely.

Citations

Costabile, M. Using online simulations to teach biochemistry laboratory content during COVID-19. *Biochem Mol Biol Educ*. 2020; 48: 509– 510. https://doi.org/10.1002/bmb.21427

"Create React App." Create React App · *Set up a Modern Web App by Running One Command*., create-react-app.dev/.

"Git Feature Branch Workflow: Atlassian Git Tutorial." *Atlassian*, www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow.

"SCRUM." *Scrum*, www.scrum.org/.