Technological Feasibility

November 3rd, 2023

Sponsored by: Dr. Elise Donovan, Sneha Vissa, Adonna Rometo

Mentor: Italo Santos

Team members: Dayra Quinonez, Nicole Sylvester,
Rino De Guzman, Bailey Rosato

Table of Contents

1. Introduction

Anatomy and Physiology are the foundation of healthcare, and they have a profound role in the capability of healthcare professionals in diagnosing and treating patients. However, in textbooks from 1996 to 2018, a study conducted by The Journal of Surgical Research found that light skin color images made up 95.7% of the overall images[1]. The lack of diversity in educational material for future healthcare workers affects the quality of patient care and impedes diversity and inclusivity in the healthcare industry. Without a range of different models and diagrams, the educational aspects of healthcare are limited to generic, white, and skinny models.

Our clients, Dr. Elise Donovan, Adonna Rometo, and Sneha Vissa are professors in the Department of Biology Sciences and are all members of the Biological Sciences DEIJ Committee. As coordinators of the anatomy and physiology courses, designated as BIO201, our clients have evaluated their courses and recognized the lack of diversity in the models that they provide for students. Currently, the only non-white model that Northern Arizona University owns is a torso model as non-white models are very difficult to find and typically expensive. So, a poorly constructed partial model is the only option they have available aside from expensive virtual models. Our clients want students taking anatomy and physiology to feel represented and have a stronger connection to course material by fostering inclusivity with more diverse models. Our goal is to create a supplemental web application lab resource that allows students to generate models based on features such as skin tone, height, and weight.

The main goals of this application are to accurately display the adjustable traits and generate a 3D model that will mimic the drawings from textbooks and physical models. The web application will have other elements that are designed to reinforce topics discussed in lectures and lab sessions so once the model is generated, users will be able to study topics with a range of diverse models. For now, the web application will only need to display superficial elements of course content, but our clients would like the product to be expandable to all units of the anatomy and physiology course and cover a wider breadth of anatomical modeling.

At this stage, we are in the process of understanding how we are going to implement the desired features and are focused on analyzing technology options, finding alternatives, and identifying which of the alternatives would be the most applicable if necessary. In this Technological Feasibility Analysis document, we begin by analyzing the major technological challenges, a BIO201 specific interface, creating high-resolution 3D models, implementing these models with

adjustable features, and managing the web application with a database. Next, we will carefully analyze each of these main challenges and look for alternatives for each and explain the rationale for choosing a specific technology. We will also prove the feasibility of each technology and then discuss how each technology will be integrated in the further development stages.

2. Technological Challenges

This section will investigate the major technological challenges facing this project and the implementation decisions that will be made to create a diversified anatomy and philosophy web application specific to the BIO201 curriculum. We have identified four major technological challenges below.

2.1 A responsive, interactive, BIO201 specific user interface

Since this web application will be used as a learning tool for BIO201 students, our goal is to create an application that is responsive, interactive, scalable, and intuitive. We aim to provide students with the flexibility to access the application from both laptops and mobile devices, ensuring a seamless and intuitive learning experience. Our vision is to enable students to engage with course content beyond initial units, fostering ease of navigation and convenient learning throughout their educational journeys.

2.2 Creating high-resolution 3D models

The second challenge we have identified is creating high-resolution 3D models. High-resolution 3D models are essential for teaching anatomy and physiology effectively. They enable students to explore and understand complex concepts in detail. Since these models are used for educational purposes, these models need to be accurate, high-resolution, and realistic.

2.3 Integrating 3D models with adjustable features

The third challenge is integrating 3D anatomical models into our web application and allowing users to change the features using shaders, material, and adjustment features. Additionally, providing adjustable features such as skin color and body size, and allowing zooming and

rotation, enables students to customize their learning experience and better mirror the diversity of humans. This in turn supports a more inclusive learning environment for BIO201 students.

2.4 Managing the web application

Lastly, using an appropriate database to be able to manage the associated data of the models and user settings to have quicker retrieval times is key for user interaction with the web application. Our goal is to leverage a database for easier storage of modeling materials and ensure that scalability capabilities are appropriate for future iterations of the web application.
These are the main challenges our team has discovered, and they are all essential because they directly impact the student's educational experience. A responsive user interface and interactive 3D models make learning anatomy and physiology more engaging and inclusive, while the ability to support high traffic ensures that the application remains dependable.

3. Technology Analysis

3.1 Tech issue 1: A responsive, interactive, BIO201 specific user interface

3.1.1 Introduction to Issue

The first challenge we face is the development of a responsive, interactive user interface tailored for our anatomy and physiology web application. This challenge involves accommodating different screen sizes, and ensuring that the user interface is interactive, enabling students to actively engage with the web applications content and features.

3.1.2 Desired characteristics

For this project, an ideal solution to this challenge would exhibit a responsive design, user interaction support, scalability, and consistency.

- **3.1.2.1 Responsive Design**

A responsive design is critical since the user interface should seamlessly adapt to different screen sizes to ensure an optimal user experience on both laptops and mobile devices.

- **3.1.2.2 User Interaction Support**

User interaction support is essential as the web application should provide support for various input methods, including keyboard, mouse, and touch interactions. This should also include

dropdown menus, navigation buttons, feedback dialog pop-ups, setting sliders, buttons, and checkboxes. These elements are vital for enhancing user engagement and ease of use.

- **3.1.2.3 Scalability**

Since this application's purpose is to supplement BIO201 content, scalability is a key characteristic to ensure that the user interface can handle growth in data without a drop in performance. As the application grows, more BIO201 curricula will be added and will need to remain responsive, saving future users from extensive changes as the content expands.

- **3.1.2.4 Consistency**

Last, consistency in a user interface is crucial because design elements should remain the same so user interaction patterns can form. This reduces confusion and application-specific emails to their professors.

3.1.3 Alternatives

With these factors in mind, we explored frontend framework options to use alongside HTML/CSS and JavaScript. These options include no framework, React, Angular, and Vue.

- **3.1.3.1 No Framework**

The first alternative is to use traditional web development technologies, including HTML/CSS and JavaScript with no framework. This approach will involve manually coding the structure and style of web pages in the application using HTML/CSS while JavaScript will be used to add interaction to the interface. We know about these technologies since we have been introduced to them in previous classes.

HTML is the standard markup language for creating web pages, created by Tim Berners-Lee in 1993. CSS is used to control the layout and design and the first version was created in 1996 by Håkon Wium Lie. JavaScript, on the other hand, was created in 1995 by Brendan Eich and is used for client-side interactivity. These technologies have been the foundation of web development for many years and are used in both simple static websites to complex applications.

- **3.1.3.2 React**

The second alternative is to use the React framework for developing a responsive and interactive web application. React is a JavaScript library designed to simplify the creation of dynamic user interfaces through component-based architecture. React applications are built using JSX and use DOM for performance optimization. Our team knows about React, as a few of us have used it before in internship work. React was developed in 2013 by Jordan Walke at Facebook. React's

component-based structure makes it a favorable choice for web application projects that require a high level of interactive and dynamic content.

- **3.1.3.3 Angular**

The third alternative to adopt is the Angular framework for building responsive, interactive user interfaces. Angular is a frontend framework that provides an environment for building web applications. Angular is a new framework for our team and was found as we researched frontend frameworks. Angular was released by Google in 2016 and is suitable for large-scale applications. Angular has many powerful tools but as a result, has a steep learning curve. Angular is used in various applications that require structured and well-organized architecture.

- **3.1.3.4 Vue**

The fourth alternative is the Vue framework for building user interfaces and web applications. Vue is an open-source Javascript framework developed by Evan You in 2014 and is known for its simplicity and flexibility. Vue has a more gradual learning curve and is used to create a variety of web applications from small single pages to more complex projects.

3.1.4 Analysis

Based on our desired characteristics, responsive design, user interaction support, scalability, and consistency, we will analyze each of the above alternatives.

Since most of our team was familiar with HTML/CSS and Javascript and had all programmed with them before, we evaluated the decision to use no framework on previous knowledge of capabilities and researched more features. However, React, Angular, and Vue required more research on their capabilities since they were less familiar to us.

- **3.1.4.1 No Framework**

Responsive Design: HTML and CSS are flexible and can be used to create responsive layouts for various screen sizes. However, achieving responsiveness often requires manual coding and testing. JavaScript can be utilized to enhance responsiveness but may involve more effort to adapt the interface to various devices.

User Interaction Support: JavaScript is a versatile language for adding user interaction and dynamic elements, and offers better control, however, better control means more effort to implement.

Scalability: HTML/CSS and JavaScript can be used for small to medium sized projects. For larger projects, it becomes more challenging to maintain scalability without a structured framework.

Consistency: Achieving consistency will depend more on our team's collaboration, because without a framework to enforce consistency, maintaining a uniform look across the web application will be more difficult.

- **3.1.4.2 React**

Responsive Design: React provides a foundation for building responsive web applications. Its virtual DOM and component-based structure make it easy to create responsive designs that adapt to different screen sizes.

User Interaction Support: React is designed for building interactive user interfaces. It offers tools and libraries, such as React Router and state management options, which help create interactive components.

Scalability: React is used in various sized projects. Its component-based architecture encourages code modularity and reusability, making it scalable for small and large applications.

Consistency: React enforces consistency through its component-based structure, making it easier to maintain a uniform look throughout the application.

- **3.1.4.3 Angular**

Responsive Design: Angular is capable of creating responsive designs but requires additional CSS frameworks to support high responsiveness. Angular focuses more on structuring and managing web applications, rather than their responsiveness across devices.

User Interaction Support: Angular provides tools for adding interaction to web applications. It supports two-way data binding, which allows components to share data and simplifies user interaction development.

Scalability: Angular is designed for building large, complex web applications. It offers features like a well-structured codebase, making it great for scalability.

Consistency: Angular enforces consistency through its architecture, but has a steeper learning curve due to its many built-in features.

- **3.1.4.4 Vue**

Responsive Design: Vue offers support for responsive design but still requires additional CSS libraries for more complex layouts.

User Interaction Support: Vue provides an interactive user experience using two-way binding, making it easier to develop interactive features.
Scalability: Vue is useful for a range of project sizes. It has a component-based structure that encourages reusability and scalability.
Consistency: Vue also supports consistency through its component-based architecture that allows for maintainable code and a consistent feel throughout the application.

3.1.5 Chosen Approach

Through our investigation, we found that HTML/CSS and JavaScript are common web development languages but without a framework, lack advanced options for responsiveness, user interaction, scalability, and consistency. React is a JavaScript library that offers better responsive design and user interaction support as well as strong scalability and consistency. Angular offers similar levels of interactiveness, scalability, and consistency as React but has a steep learning curve and does not offer responsiveness as nice as React without other libraries. Vue is excellent at user interaction support, scalability, and consistency but requires additional libraries for responsive design.

Table 3.1.5 Rating Alternatives based on desired characteristics.

	Responsive Design	User Interaction Support	Scalability	Consistency	Average
No Framework	2	2	2	2	2
React	5	5	5	4	4.75
Angular	4	4	5	3	4
Vue	4	4	4	4	4

As Table 3.1.5 shows, various possible options for a framework to use with HTML/CSS and Javascript. Each option is rated in the categories of responsive design, user interaction support, scalability, and consistency on a scale of 1 to 5, with 5 being the highest rating.
Using no framework received a 2 in every desired characteristic since without a framework, achieving a responsive design, implementing interactive elements, scalability and consistency are all more challenging and less efficient.

React received a 5 in responsive design, user interaction support, and scalability since it excels in responsive design due to its virtual DOM, component architecture, and interactivity tools. It falls short and received a 4 in consistency since it maintains a uniform look but flexibility in design requires more effort.

Angular received a 4 in both responsive design and user interaction support since Angular is capable of creating responsive designs but requires additional CSS frameworks to achieve the same level of responsiveness as React and Angular provides tools for interactivity like two-way data binding but has a steeper learning curve than React and Vue. Angular received a 5 in scalability since angular is designed for large-scale applications but received a 3 in consistency since it requires more setup and learning compared to React or Vue.

Vue received 4 in every category because it supports responsive design but, similar to Angular, requires additional CSS frameworks for responsiveness. It also provides interactive support through two-way binding similar to Angular and is scalable and consistent because of its component structure.

Therefore, HTML/CSS and JavaScript combined with the React framework is the most promising solution to a responsive, interactive user interface for a BIO201 specific diversified web application.

3.1.6 Proving Feasibility

To validate our choice of using React as a framework for building the responsive, interactive, BIO201 specific user interface, we plan to further test and validate through specific demos. First, we will create sample web pages with various layouts and designs, in different screen sizes to ensure that our interface adjusts seamlessly to various devices. We will also develop interactive features and components such as drop-down menus, navigation buttons, sliders, and any interactive content that our clients want to showcase React's user interaction capabilities. These demos will serve as validation that our chosen approach aligns with the specific needs of our project.

3.2 Tech Issue 2: Creating high-resolution 3D models

3.2.1 Introduction to Issue

The second challenge will be providing images to users that are reliable and accurate. These images will be a supplement to the learning of the students. It should be allowed to be

personalized not only to the Biology 201 course, but to each student as well. The material should be realistic enough to learn from. It is essential to have a production of models that are interactive and engaging.

3.2.2 Desired characteristics

Accordingly, the solution to this matter would be to create three-dimensional models through software that caters to the Biology 201 course.

- **3.2.2.1 Realistic Rendering**

The rendering of these images must be up to standard to be academic. The images should be able to look realistic enough and result in high quality.

- **3.2.2.2 Real Time Rendering**

The renderings must be rendered in real-time. In the web application, the users will be allowed to modify and adjust features to which they prefer. However, for this specific issue, a generic model needs to be created and it is important as the model is being made, the changes are automatically visible.

- **3.2.2.3 Interoperability**

Interoperability is essential for the purpose of this web application. After the creation of the renderings, it is necessary to export these models to another software. The other software will be in charge of the interaction with these same models on the web application. It is important to make sure that this software is compatible with the other software or library.

- **3.2.2.4 3D Modeling Features**

The features that the software will offer are important. The software should have features that allow creators to create anatomical models. The foundation should be well established. The models should be well-built. Accuracy and realism are important characteristics of this issue specifically. The material that is being demonstrated must be an accurate representation of anatomy material.

3.2.3 Alternatives

After discussing the desired characteristics, we examined different software that creates three-dimensional models. The softwares examined are Blender, Unity, and Adobe Substance 3D.

- **3.2.3.1 Blender**

The first alternative is to utilize Blender Software. After searching for softwares to create three-dimensional models, Blender was one of the softwares that was recommended on the web. The software was created by a Software Developer, Ton Roosendaal, after he was interested in starting his own 3D animation studio. He released the software in January of 1994. Blender software is typically used by 3D animators, graphic designers, video editors, and game designers. The software supports modeling, animation, game creation, compositing, and motion tracking. It is used in game development and architecture visualization.

- **3.2.3.2 Unity**

The second alternative to apply is Unity. The software was created by Nicholas Francis, Joachim Ante, and David Helgason. They created a game that failed but found value in their foundation, so they instead created a platform for developers to develop 2D and 3D content that is interactable. The software was released in 2004. This software is mainly utilized by game developers to create game applications, but it can create 3D renderings. Our team is knowledgeable about Unity as some are utilizing this software this semester.

- **3.2.3.3 Adobe Substance 3D**

The third alternative to adopt would be Adobe Substance 3D. Adobe was founded back in December of 1982 by John Warnock and Charles Geschke. After searching for softwares that created three-dimensional models, Adobe Substance 3D software was one of the softwares that was recommended. Adobe Substance 3D was released to the public in August 2019. Illustrators and animators tend to use this software.

3.2.4 Analysis

Referring to the desired characteristics: of realistic rendering, real-time rendering, Interoperability, and 3D modeling features, we will examine the three alternatives.

- **3.2.4.1 Blender**

Realistic Rendering: Blender is a powerful tool for rendering, and it offers amazing quality models. The models that can be created can be as realistic as you please.

Real-time rendering: They have incorporated an engine, Eevee, that does the rendering in real time.

Interoperability: This software offers the opportunity to be able to import and export to and from other softwares.

3D modeling features: Blender has a lot of foundational features that would allow us to create the anatomical models that we need. It is regarded highly in the industry for its versatility in features.

- **3.2.4.2 Unity**

Realistic Rendering: Unity offers the ability to create models that are realistic and look accurate.
Real-time rendering: Unity is a software application that allows you to render the models as one is working and making updates.
Interoperability: Unity allows you to be able to import and export models from and to other softwares through specific file extensions.
3D modeling features: Unity has well established extensive libraries of pre-built assets and plugins

- **3.2.4.3 Adobe Substance 3D**

Realistic Rendering: Adobe does not support the expectations of realism needed for this project.
Real-time rendering: Adobe does support real time rendering.
Interoperability: Adobe does not have a high interoperability. It does offer a few file extensions, but not the ones needed to be able to apply the models to the next step of the project.
3D modeling features: Adobe has established their 3D modeling features, but it seems to have limitations that may cause a problem.

3.2.5 Chosen Approach

After researching all of the alternatives, there were some interesting findings. Both Blender and Unity are softwares that can develop high resolution and realistic models, however, a key difference is the cost, while Unity charges for a license, Blender is an open-source software. Adobe Substance appears to be lacking in what we are looking for.

Table 3.2.5 Rating Alternatives based on desired characteristics.

	Realistic Rendering	Real Time Rendering	Interoperability	3D modeling features	Average
Blender	5	4	5	5	4.75
Unity	4	5	5	4	4.5
Adobe	2	3	1	3	2.25

As demonstrated on Table 3.2.5, Blender received a score of 5 for realistic rendering, interoperability, and 3D modeling features. Blender has excellent support for modeling features and rendering realistic models. The software also supports the ability to be able to import and export models with a certain extension that many other softwares and libraries support as well. As for real time rendering, Blender received a 4 because when comparing it to Unity's ability, it needed to incorporate an engine, Eevee, for real time rendering.

Unity received a score of 5 for real time rendering and interoperability because it has a good support for both of these characteristics. Unity is a game engine that relies on real time rendering and since it is an excellent software, it allows the ability to work with other softwares and libraries like Blender. This software received a 4 for realistic rendering and 3D feature models because although it does a good job, compared to Blender it is not as equipped. However, it is more equipped than Adobe Substance 3D.

Adobe Substance 3D received a score of 3 for real time rendering and 3D modeling features. The software is set, and it can do three-dimensional modeling however it is not as advanced as Unity or Blender. As for realistic rendering, it received a 2 because the models that were demonstrated did not meet the standards that we are looking for to create these anatomical models. Whereas interoperability received a score of 1 because it only supported two extensions that would not be as useful as the extensions that the other two alternatives allow.

Therefore, utilizing Blender Software would be our best choice to create these three-dimensional models based on the Biology 201 course material.

3.2.6 Proving Feasibility

To support our choice of going with Blender as our software to build three dimensional models, we plan to further test through demonstrations. We plan to create a prototype of an anatomical model utilizing the software to prove that it will meet the requirements. Testing if the model is realistic enough, renders in real time, can be transferred easily, and that it offers the modeling features necessary.

3.3 Tech Issue 3: Integrating 3D models with adjustable features

3.3.1 Introduction to Issue

The third challenge that we will face is to choose a Javascript 3D library to upload 3D anatomical models and implement features that allow the user to customize the imported models in the web application. In order for the product to be genuinely valuable, the student must be able to interact with the model in multiple different ways to enhance their learning experience in BIO 201.

3.3.2 Desired Characteristics

The key characteristics of an optimal solution for this technical challenge would display efficient performance, ease of use, expandability, and functionality.

- **3.3.2.1 Performance**

It is important to choose a Javascript 3D library that provides efficient performance. It is critical for the user to be able to immediately see their chosen modifications in real time. The user also needs to be able to interact with the model smoothly without too much lag or delay.

- **3.3.2.2 Ease of Use**

On the developer side, the library needs to be relatively easy to work with, as individuals in our team are not deeply experienced with 3D programming. It is crucial that the library makes it easy for the team to focus more on developing the features we want to implement for the project than navigating through the nuances of the Javascript library.

- **3.3.2.3 Expandability**

The clients see this product as an ever-growing project that will continue to build over time. The clients intend to add new features and improvements every year. Though the required features of the minimum viable product of this project are relatively simple. The long-term commitment that the clients have with the project makes it very important that the web application is supported by a Javascript library that is expandable. It is important that the 3D Javascript Library is capable and future-proof for any future endeavors that the clients may have for the project.

- **3.3.2.4 Functionality**

It is important that the Javascript Library can accommodate the features and requirements of the project's minimum viable product. The library is of no use to the team if it is not able to fulfill the requirements and needs of the features.

3.3.3 Alternatives

With these desired characteristics in mind, there are 3 alternatives that can be used to implement the anatomical models into our web application: Three.js, Babylon.js, and PlayCanvas.

- **3.3.3.1 Three.js**

The first alternative that the team found is Three.js. Three.js was the first and by far the most popular library that popped up in our search for technologies with the ability to implement interactive 3D models. Three.js was created by Ricardo Cabello and was first released on Github in April of 2010. It is a 3D library under MIT licenses that aims to make it as easy as possible to get 3D content on a webpage. Some individuals get confused between Three.js and WebGL, to clarify, Three.js uses WebGl to draw 3D in most cases. Three.js sits on a level above WebGL and handles scenes, lights, shadows, materials, textures and 3d math which would all have to be coded manually by the programmer if WebGL was solely used. Three.js is seen as a great choice for developers who want to quickly and easily create simple 3D applications.

- **3.3.3.2 Babylon.js**

The second alternative is the library Babylon.js. When researching Three.js, Babylon.js appeared alongside it. We often found forums and articles that pitted the two libraries together. The library was developed by two Microsoft employees and was initially released in 2013 under the Microsoft Public License. The two developers were David Catuhe and David Rousset. David Catuhe created the 3D game engine while David focused on VR, Gamepad, and IndexedDB support. This was mostly developed in their spare time as a side-project! It was only later in which it became his full-time job and his team's primary focus. Babylon.js has grown largely since its release and is used in a variety of fields such as: virtual worlds, crime data visualization, medicine education, and product design. Babylon.js is seen as a full-fledged 3D game and rendering engine built for the web. It is designed specifically for game development and offers many advanced features such as physics/animation engines.

- **3.3.3.3 PlayCanvas**

The third and final alternative is PlayCanvas. PlayCanvas is the least popular out of the 3 and it appeared through our search for web 3D engines. PlayCanvas is an open-source 3D game engine that also has a proprietary cloud-hosted creation platform to allow for collaborative real time editing from multiple computers via a browser-based interface. PlayCanvas was open-sourced on June 4, 2014 and was created by a multitude of developers. Some names of the developers

include Will Eastcott, Dave Evans, Vaios Ilias, Kevin Rooney and Maksims Mlihejevs. Like Babylon.js, it is a full-fledged 3D game engine with capabilities such as rigid-body physics simulation, three-dimensional audio and 3D animations. Although it may seem like an unpopular program it has still been used by the likes of Disney, Miniclip and King (Candy Crush).

3.3.4 Analysis

Based on our desired characteristics of Performance, Ease of Use, Expandability, and Functionality we will analyze each of the above alternatives.

- **3.3.4.1 Three.js**

Performance: Three.js is a lightweight, cross-browser general purpose 3D library that is meant for simple 3D visualizations with some levels of interactivity. Three.js specializes in rendering 3D scenes inside web pages in a browser.

Ease of Use: The main benefit of Three.js is its supposed Ease of Use. Three.js boasts a large community with examples and plenty of third-party plugins. In addition, Three.js is known for its informational and organized documentation. These aspects in combination with its ability to create complex 3D scenes with ease makes it a very compelling library.

Expandability: Depending on the goal of the application and the requirements of the project, Three.js is capable of more than just 3D visualizations. There are many users in the community that still opt to use three.js for their game development endeavors.

Functionality: Three.js includes many features such as Anaglyph, cross-eyed, parallax barrier effects, multiple camera techniques, animations, light techniques ,materials/shaders, objects, geometry, import/export, debugging and AR/VR capabilities. On the other hand, it is not a game engine, you will not find Physics, Particles, Input, Asset Management, Scripting, etc as part of its functionality.

- **3.3.4.2 Babylon.js**

Performance: Babylon.js is a heavy-hitting 3D game engine that offers great performance for complex applications. Babylon.js renders graphics in real-time through the combinational use of WebGL and WebGPU, this results in the capability of fast frame rates even in complex scenes.

Ease of Use: The drawbacks of Babylon.js is its learning curve. Babylon.js provides a decent number of examples but it is difficult to adapt to your own project. The documentation, although well written, is said to take some time to become familiar with. It is a small community; this may make it harder to search through forums or find information.

Expandability: Due to the application being a 3D game engine, it is more than capable of any future development that the clients may have with the project.
Functionality: Babylon.js is a very powerful tool capable of creating rich and complex applications. Babylon.js' mountainous features include Animations engine, Particles and Solid particles systems, Complete audio engine, Hardware accelerated GUI, Shaders/Rendering techniques, and many Special FX effects. In addition, Babylon.js allows for more realistic lighting and shading due to its support for physically based rendering and wide range of materials/textures.

- **3.3.4.3 PlayCanvas**

Performance: PlayCanvas provides overall great performance for 3D applications. PlayCanvas has Optimization Guideline documents that allow the developers to keep performance in mind during development.
Ease of Use: Unfortunately, there seems to not be much information available regarding PlayCanvas. The community is small which may make it harder to search the forums for issues that may arise during development. The editor is known to be easy to use and is able to rapidly build WebGL applications.
Expandability: Since PlayCanvas is a 3D game engine, it has capabilities for complex features and applications. This allows the application to be more future proof as it is able to cater to any future project needs.
Functionality: One of the features that make PlayCanvas stand out from the rest is its ability for live collaboration from multiple computers. This is incredibly useful for teams that want to work live together on development and design. The editor is known to be extremely powerful.

3.3.5 Chosen Approach

After thorough research of each alternative, both Three.js and Babylon.js seem to be the most compelling choices to manage the 3D renderings of the anatomical models.

Table 3.3.5 Rating Alternatives based on desired characteristics.

	Performance	Ease of Use	Expandability	Functionality	Average
Three.js	5	5	3	4	4.5
Babylon.js	5	3	4	5	4.5
PlayCanvas	3	3	3	4	3.25

As can be seen on Table 3.3.5, Three.js received a score of 5 both in Performance and Ease of Use. Three.js has excellent performance in regard to rendering 3D models and interactivity. Its large community and wealth of resources make it the most beginner friendly library out of the 3. It also is very functional but may not be the most expandable option.

Babylon.js received a score of 5 both in Performance and Functionality, Babylon's top tier hardware acceleration and jam-packed features make it the application with the most capability. Although, with the requirements of the current minimum viable product, it seems that Babylon's power is simply not needed. In terms of Ease of Use it scored a 3 due to its steeper learning curve. From searching forums, people who have had experience with Three.js and Babylon.js express that Three.js is quicker to pick up and easier to get started with.

Finally, PlayCanvas scored average across the board. It is a capable library in its own right but it does not excel and shine in its own ways like Three.js and Babylon.js. Its functionality scored a 4 since it is often compared with Babylon.js in terms of functionality. The only feature that sets it apart from the other 2 technologies is its ability for live collaboration within the editor.

Therefore, due to its Ease of Use and more than enough functionality, Three.js seems to be the most promising solution to manage our realistic and functional 3D anatomical models.

3.3.6 Proving Feasibility

To validate our choice of using Three.js as the 3D library API for our web application, we plan to further test and validate through prototype demos. First, we will create bare prototype models through Blender and export the 3D model. We will then use the Three.js library to import and display the 3D model into our web application. On the imported prototype models, we can test the functionality of the shader and material adjustments which change the skin tone and adjust

the material properties of the 3D model. The second feature to test on the prototype models is body size adjustment, this will allow us to see how we can modify the scale or geometry of the 3D model using Three.js functions. Lastly, the third feature to test is user interaction. It is important to test the ability to implement functionality that captures user inputs via sliders or buttons to apply the changes to the 3D model in real time.

3.4 Tech Issue 4: Managing the web application

3.4.1 Introduction to Issue

The final challenge of creating this product is being able to functionally store all the necessary data for the models and user settings. Furthermore, choosing the right database for this application is significant for scalability and performance optimization for data retrieval. To ensure the future iterations of the application are smooth, finding a database with robust functionality is key.

3.4.2 Desired characteristics

The desired characteristics of an optimal database for this technological challenge are data model, performance, scalability, and interoperability.

- **3.4.2.1 Data Model**

To ensure that the database adequately addresses the data organization needs of our web application, it is necessary to compare the data models of the databases. The two types of databases are relational and non-relational. The data model of the database can affect future limitations of data storage and functionality.

- **3.4.2.2 Performance**

It is important to choose a database library that has optimized performance.To reduce delays when generating models, choosing a database with quick retrieval times is desired. Higher retrieval times will also provide ease of use for students when altering traits on the model.

- **3.4.2.3 Scalability**

Due to the sheer volume of course content, scalability is a key characteristic to ensure that the database can handle growth in the size of data or servers without sacrificing performance. In future iterations, more BIO201 content will be created in the form of more models and associated data, so the database chosen must be able to handle significant future additions.

- **3.4.2.4 Interoperability**

Interoperability is another key factor in choosing a database because the front-end of the application must be able to functionally communicate with the database to generate models. An optimal database will be compatible with technologies and ecosystems that front-end features use.

3.4.3 Alternatives

With these desired characteristics being significant in deciding an optimal solution, the three products that will be investigated are PostgreSQL, MongoDB, and Firebase Realtime.

- **3.4.3.1 PostgreSQL**

The first option to be evaluated is PostgreSQL, which is a free, open-source relational database modeling system. Developed at the University of California Berkeley, Postgres is named after the INGRES database that it was based off of. The project was headed by Professor Michael Stonebreaker in 1986 and was first released in 1996. PostgreSQL is best known for its strict ACID(Atomicity, Consistency, Isolation, and Durability) compliance, use of advanced data types, and strong user and developer community. Postgres flexibility and extensibility make it applicable to a wide range of use cases.

- **3.4.3.2 MongoDB**

Another option that our team evaluated in MongoDB, which is an open source, non-relational database that is cross platform. MongoDB requires no schema for its database management system. It was developed by Kevin P. Ryan, Dwight Merriman, and Eliot Horowitz in 2007 to address scalability issues with traditional relational databases. First released in 2009, MongoDB was unique because it offered JSON querying and provided a more hierarchical organization of data which was more dynamic for developers. Compared to more traditional database management systems, MongoDB is slightly more complicated to initialize and manage if you are unfamiliar with it. Furthermore, MongoDB has a possibility of high memory requirements depending on the type of data the application requires.

- **3.4.3.3 Firebase**

Lastly, the final option found for a database management system is Firebase Realtime. Like MongoDB, Firebase Realtime is also a non-relational database management system but differs in that it is a cloud-hosted system. First released in 2012 by the Firebase company, Firebase Realtime was designed to synchronize cross platform data and store it on the Firebase cloud. Firebase Realtime is able to store and synchronize data to the cloud in real time. Firebase has a

free basic plan and has a quick and easy setup. Users have found that there are limitations in the complexity of querying for the database as a drawback.

3.4.4 Analysis

Considering the desired characteristics of data mode, performance, scalability, and interoperability we will now analyze each option's effectiveness.

- **3.4.4.1 PostgreSQL**

Data Model: As a relational database with extensive features, PostgreSQL has capabilities that could be useful for this project. However, because of the ACID compliance it has a high overhead that can lead to performance issues.

Performance: PostgreSQL is known for its robust querying capabilities, but it does rely on tables and relational designs for querying. The sizing of modeling documents may be large and in turn affect the retrieval times of querying.

Scalability: PostgreSQL does have the capability of vertical and horizontal scalability, which will be useful for using resources and considering the addition of more servers in future iterations.

Interoperability: PostgreSQL has several libraries and APIs available that are compatible with several languages and frameworks, such as Node.js. Furthermore, there is a large developer community that is helpful for integrating Postgres with other technologies.

- **3.4.4.2 MongoDB**

Data Model: MongoDB is a non-relational database, meaning that no schema is required, and no predefined structure has to be in place before use. This would be helpful for future iterations of the project because it would have more flexibility for future development if better modeling strategies and storage was found.

Performance: As a document-oriented model, MongoDB is optimal for quick retrieval times because retrieving a single document from an index is faster than the complex queries made from a table.

Scalability: MongoDB is made for horizontal scalability, which allows it to be suitable for high volumes of data and adding more servers. More servers can optimize performance in the application for the future.

Interoperability: MongoDB can be used in conjunction with React, Angular, and Node.js, which would be optimal for the front-end frameworks and technologies chosen. MongoDB also has

capabilities for importing and exporting files which would be helpful for storing models in compatible file extensions.

- **3.4.4.3 Firebase Realtime**

Data Model: Firebase Realtime is also a non-relational database, but it differs from MongoDB in that it is serverless, which can reduce operational overhead.

Performance: Firebase Realtime is best known for its real time data synchronization, which is valuable for creating real time collaborative applications, like our modeling system which will require responsiveness to user generated traits.

Scalability: Since it is hosted on Google Cloud, Firebase Realtime has extensive scalability possibilities. This will be helpful for future iterations when we do not currently know the amount of storage will be necessary for future BIO201 content.

Interoperability: Firebase Realtime is a full-stack solution so it will not require a separate backend server and is fully compatible with the Firebase ecosystem which includes third-party extensions that can help with its capabilities.

3.4.5 Chosen Approach

After thoroughly evaluating the capabilities of each alternative, it is clear that Firebase Realtime has the most desired characteristics for our web application. While PostgreSQL and MongoDB have their own unique capabilities that are useful, the combination of serverless management, cloud-based storage, and real time synchronization have the most potential for being useful for the current application requirements and for possible new modeling requirements for future implementations. As a non-relational database Firebase Realtime also provides flexibility in the structure of our document storage which will be helpful once we begin development.

Table 3.4.5 Rating Alternatives based on desired characteristics.

	Data Model	Performance	Scalability	Interoperability	Average
PostgreSQL	3	3	5	4	3.75
MongoDB	5	4	3	5	4.25
Firebase	4	5	5	4	4.5

As seen in Table 3.4.5, the rankings for the desired characteristics are displayed for PostgreSQL, MongoDB, and Firebase. Each option is rated in the categories of responsive design, user interaction support, scalability, and consistency on a scale of 1 to 5, with 5 being the highest rating.

PostgreSQL received the lowest scores for data model and performance because of the overhead required for deciding the relationships between all data in the relational model and being able to retrieve data quicker than the other options. Its scalability options for both vertical and horizontal options were more impressive than MongoDB but did not compare to Firebase Realtime.

MongoDB has the strongest performance for its non-relational model and its interoperability. MongoDB is compatible with a plethora of front-end frameworks and languages, which made it a strong contender for this application, but ultimately its lack of vertical scalability was a major drawback because, without vertical scalability, it has limitations for managing resources which could be useful in the future when the modeling becomes more complex and requires more generating power.

Therefore, Firebase Realtime is the optimal solution for this technological challenge because it has the strongest capabilities out of those desired. It will also simplify the deployment of this application because it does not require a server.

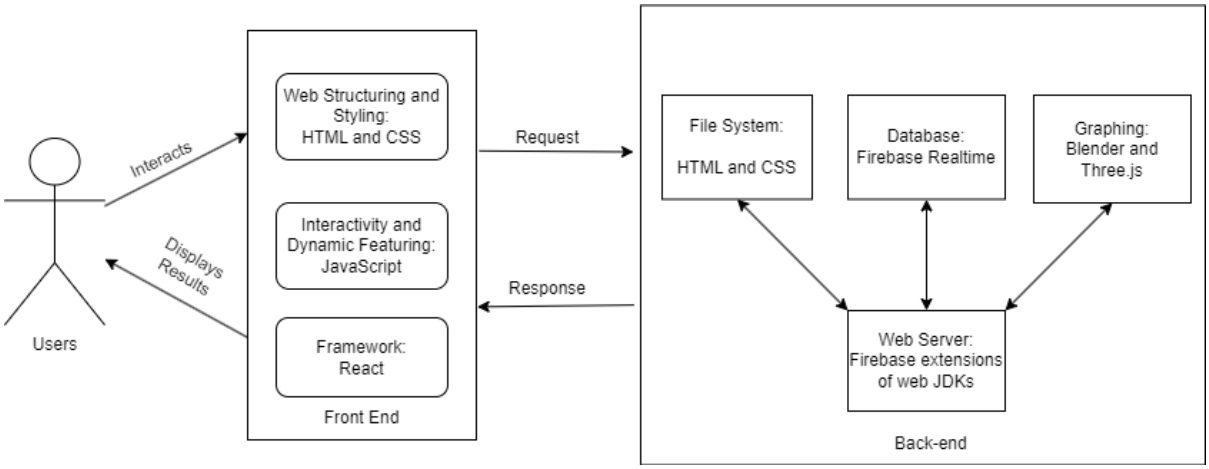
3.4.6 Proving Feasibility

To further support our use of Firebase Realtime as our management system, we will research further into the free tier of the product and begin familiarizing ourselves with the mechanics of a non-relational, serverless database by using test files and practicing querying with these test documents. We will also test its functionality with the front end by creating user settings that can be saved in a session and testing if they are maintained properly on the user interface.

4. Technology Integration

This section will cover how the technologies will integrate to tackle the technical challenges we will be facing.

Figure 4: A diagram showing how the technologies will integrate together.



As shown in Figure 4, to have a functioning web application we will have to integrate all of the technologies we have discussed in the previous section. The User will be interacting with the website application and will only be able to see the visualization of the page and the models. The website's structuring and styling will be coded through HTML and CSS. JavaScript will be enhancing its performance by adding dynamic features, increasing the user's interactivity level. React will be responsible for the framework. As the user is interacting and modifying models, the front-end will be requesting from the back end. The database will be handled through Firebase. Our three-dimensional models will be created using Blender Software and implemented and displayed through Three.js.

5. Conclusion

Despite the plethora of resources available to anatomy and physiology students, diverse models are still not widely available, and most students will only learn from a white and skinny model in their formal education. The lack of access to the available resources is not only a detriment to students and their education but also to patients who can face misdiagnosis. Creating a more accessible tool for diverse modeling will allow future healthcare professionals to be exposed to more realistic patient body types.

Using a combination of React, Blender, Three.js, and Firebase, our goal is to create an interactive web application that allows BIO201 students to analyze a 3D model and change features of that model to engage with more diverse anatomical models. After a thorough analysis of our technologies, we can move to the next steps of development. Upon client review and unforeseen limitations with technology, our chosen approaches and technologies will be subject

6. References

[1] Trabilsy M;Roberts A;Ahmed T;Silver M;Manasseh DME;Andaz C;Borgen PI;Feinberg JA; "Lack of Racial Diversity in Surgery and Pathology Textbooks Depicting Diseases of the Breast." *The Journal of Surgical Research*, U.S. National Library of Medicine, pubmed.ncbi.nlm.nih.gov/37562229/#:~:text=Among%2012%20textbooks%20reviewed%2C%20seven,%2 ,%20with%20dark%20skin%20color. Accessed 6 Oct. 2023.