

# **Frontend to Backend: A Learning Tool for Full Stack Web Development**

Aliya Ismagilova

Advisor: Robert Fish

Submitted in Partial Fulfilment

of the Requirements for the

Degree of

Bachelor of Science in Engineering

Department of Computer Science

Princeton University

2022

**© Copyright by Aliya Ismagilova, 2022. All rights reserved.**

I hereby declare that I am the sole author of this independent work.

I authorise Princeton University to lend this independent work to other institutions or individuals for the purpose of scholarly research.

---

Aliya Ismagilova

I further authorise Princeton University to reproduce this independent work by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

---

Aliya Ismagilova

# Abstract

*This paper details the design and development of an online learning tool which teaches people the principles of client-server interaction. It takes the form of a website which contains short, animated and interactive guides, complete with a quiz at the end of each one to test the learner on the content. This learning tool aims to teach some of the most fundamental concepts behind full stack web development in an accessible and dedicated manner, such that programmers from all backgrounds can feel more confident and comfortable building web applications of their own.*

# Acknowledgements

This independent work, written and submitted in Fall 2021, would not have been possible without the love, support, and wisdom of my family. Мама, Папа, Timur, I love you so much. I am so fortunate to have you by my side.

To my dear, beloved friends who graduated from Princeton in 2021 - you assured me that I would have an incredible senior year and at first I didn't believe you. You were totally right.

To my dear friend, teammate and partner in crime, Anna - thanks for making my senior spring awesome.

This independent work is posthumously dedicated to Abraham Joshua '21 - my first true friend at Princeton, a lifelong mentor, teacher, artist and a total academic weapon. Thank you for everything.

# Contents

<b>Abstract</b>	3
<b>Acknowledgements</b>	4
<b>1. Introduction</b>	6
1.1 Motivation and Goal	6
1.2 Overview of challenge and previous work	8
1.3 Approach	9
1.4 Summary of Implementation	9
1.5 Summary of Results	10
<b>2. Problem Background and Related Work</b>	10
<b>3. Approach</b>	16
3.1 Overview	16
3.2 Learning Path	16
3.3 Quizzes	17
3.4 Animation Choices	18
<b>4. Implementation</b>	19
4.1 Project Architecture	19
4.2 Workflow of the Learning Path	20
4.3 Design Features	21
<b>5. Evaluation</b>	26
5.1 Approach	26
5.2 Results	27
<b>6. Conclusion</b>	28
6.1 Summary	28
6.2 Limitations	29
6.3 Future Work	29
<b>7. Ethics</b>	30
<b>Bibliography</b>	31

# 1. Introduction

## 1.1 Motivation and Goal

To build a website is to create a complicated and multi-layered online entity which many types of users will interact with for various purposes. With virtually limitless implementable functionality, there is growing interest in learning how to build complete and maintainable websites which are fit for any purpose: online stores for small businesses, virtual galleries for photographers, blogs, tourist information, online forums, and social media. With representations of physical space shifting online, there is a growing demand for, and interest in, the ability to construct and maintain modern and functional websites suited to their goal. To meet this demand there is a growing number of online courses which seek to teach web development, targeted at skill levels ranging from the “Hello World” novice, to the industry programmer with a Computer Science bachelor’s degree. However, most learning tools of this nature exhibit a range of limitations which causes them to fall short of their goal of providing quality and accessible teaching material. Namely, many exist behind a paywall, are often six or more weeks long, and take the form of a narrow project which often doesn’t allow for generalisations or coverage of fundamental concepts. Furthermore, many of these courses are built upon, or teach, an exclusive set of frameworks which are constantly updated, resulting in the instructions of the course becoming outdated themselves. A fundamental idea which is not covered in great depth and generality in these courses is a comprehensive overview of exactly how the clients-side and server-side technologies of a website interact with each other. I believe that understanding and mastering these

interactions is fundamental to understanding how a website truly works, and the lack of information on this topic greatly limits the confidence and aptitude of computer science students and software engineers in building great, modern websites.

At Princeton, the student who is interested in full stack web development, and has taken introductory computer science courses already is typically recommended to take COS 333: Advanced Programming Techniques. This course is well known amongst students to come the closest to teaching the sorts of programming techniques and skills used in industry. Indeed, the course description, as of Fall 2021, is the following:

*“The practice of programming. Emphasis is on the development of real programs, writing code but also assessing tradeoffs, choosing among design alternatives, debugging and testing, and improving performance. Issues include compatibility, robustness, and reliability, while meeting specifications. Students will have the opportunity to develop skills in these areas by working on their own code and in group projects.”[1]*

Having taken COS 333 myself, read its course evaluations, and spoken to fellow students who have taken the course, it is clear that although the course briefly covers the elements of under-the-hood web development for the sake of assignments, the nature and goal of the course doesn’t allow enough depth to be gained on this topic to make students feel truly comfortable with the mechanics of it, and leaves students to engage in independent learning on this topic to enrich their understanding.

The goal of this project is to service this curiosity and create an online learning tool dedicated to explaining the underlying structure of a website, how the different



components on the front end and back end of a website interact with each other to service the website and create the flow of information, and how frameworks can be used to build modern websites.

## **1.2 Overview of challenge and previous work**

As mentioned previously, there exists an industry providing virtual and in-person courses which guarantee to teach some level of full stack web development to their students.

Besides that, there also exists a lot of information online in the form of books, PDFs, and blog posts, which teach some aspects of website-building. These latter resources aid the independent learning undertaken by those who have some familiarity with writing code, who wish to learn how to take on their own website-building venture. The resources, however, are static in their nature, and success in their use relies, almost paradoxically, on certain prerequisite knowledge that the user already has. In other words, they do not cater to every skill level, and understanding them largely depends on the user's own understanding of their context.

The challenge this project aims to solve is to create a learning tool which is more guided than the independent learning route which is taken by those who seek out more information, but does not mimic the traditional online course or coding bootcamp. Rather, it dynamicises the existing information on this topic and makes it more visual, more interactive, and more accessible. It makes the information provided online in written form more digestible, and breaks it down so that it can be learned and implemented faster.

## **1.3 Approach**

The online learning tool takes the form of short, interactive guides. These guides have visual and animated elements which aid information retention and engagement. I used popular and innovative animation libraries developed for React such as react-spring and Framer Motion to create these animations. Alongside these guides, I also created accompanying quiz sections, so that the students taking the courses can reflect on what they have learned and solidify their understanding of it.

## **1.4 Summary of Implementation**

This project was implemented using a stack of front end and back end technologies. Successfully constructing the front end to be visually appealing and engaging was vital to the project, so I built the user-facing portion of my website with a framework I had worked with before and knew well: NextJS. The front end code was written in Javascript and React, both of which I had worked with in past projects (not least of which is my own COS 333 project). For user authentication, I used NextAuth.js - an authorisation framework built around NextJS, and for storing tokens and user data such as quiz scores, I used Heroku PostgreSQL, accessed with Prisma as the object-relational mapping service to put everything together. I deployed my website using Vercel, a hosting company highly integrated with NextJS.

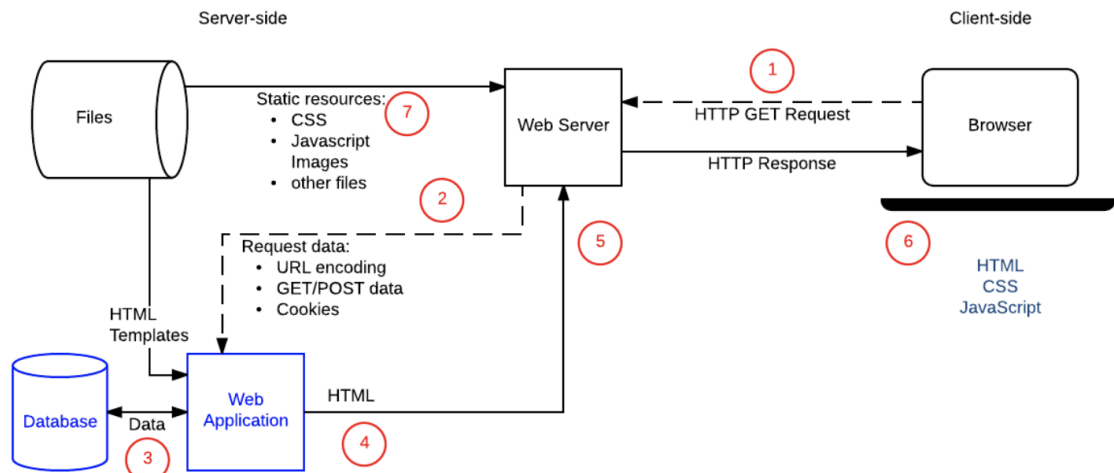
## **1.5 Summary of Results**

I asked past and present students who have taken COS 333 to assess and evaluate my learning tool by completing the first three courses and filling out a survey form filled with questions asking them to qualitatively and quantitatively assess the website. Overall, the results of the evaluation deemed the project as successful in fulfilling its goal of teaching programmers who are interested in web development the fundamental concepts of client-server interaction which they were not previously familiar with, using an engaging and interactive interface.

## **2. Problem Background and Related Work**

Whilst researching existing material which teaches client-server interactions, one of the main mediums which I came across was static, printed content. This information can be found in the form of PDFs and documentation online which is freely available online. An example of documentation on client-server interaction, found on MDN Web Docs, is titled “Client-Server Overview”[2]. It is a written, open-source document which details the eponymous relationship between the client and the server. It gives examples and explanations of HTTP requests and responses, explains the structures of static and dynamic websites with the help of diagrams, and introduces the concept of web frameworks and how they work. Overall, it is a very comprehensive guide, and a good reference point for those who are seeking to understand the fundamentals of client-server interaction. Figure 1 shows an example of a diagram found in the documentation, which

accompanies an explanation of the internal communication between the client and server sides of the website. This diagram is particularly useful because it is visual: all the components which make up the website are clearly represented and labelled, and the order of operations is numbered to accompany the explanatory text.



**Figure 1: From “Client-Server Overview” under “Dynamic Sites”**

Another example of similar documentation is titled “Client/Server Communication”[3], and it offers a similar solution to teaching the process. The material is very similar to what is found on MDN Web Docs, covering definitions of Client and Server-Side rendering, fetch, URL Paths and GET/POST requests. One major feature in this documentation is the use of inline code accompanied by screenshots of its implementation in a real website. The explanations of the various aspects of client-server communication are thus put into practise through these code snippets, and makes the explanations more accessible to the reader. Moreover, because the code is itself implementable by the reader with very little prior experience or environmental setup, this documentation also doubly serves as a self-paced course or lesson of its own. Figures 2

and 3 are from the documentation's explanation on Server-Side rendering, showing the code snippet and result on the web page which it generates for fetching and displaying Unix Time on a web page. Open-source documentation is not the only example of static information on client-server interactions, however. Existing frameworks which have features that facilitate interactions such as fetch, and request/response handling, also provide richly detailed documentation which a first-time website-builder can find helpful when starting out. Frameworks such as ExpressJS are simple enough that their guides, although specific to their own functionality, can apply generally to other frameworks or help the user understand basic or fundamental concepts that can help them move on to using other frameworks in the future. For example, ExpressJS utilises the *req* and *res* objects which represent the HTTP requests and responses respectively. The *req* object is written to have built-in properties "for the request query string, parameters, body, HTTP headers, and so on"[4]. Thus, this object closely implements the true properties of a HTTP request, and using this framework and studying its documentation is useful for understanding how HTTP requests actually work and what they entail. Although the API specification is not truly generalisable for all features of client-server interaction, learning frameworks such as ExpressJS is useful for understanding some of the concepts behind these interactions. Furthermore, frameworks are meant to be used, so the user can "learn-by-doing" rather than passively studying the concepts behind client-server interaction - they can actively build a website using the framework, and learn the concepts in the process.

```
import java.io.IOException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/time.html")
public class UnixTimeServlet extends HttpServlet {

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        long unixTimeSeconds = System.currentTimeMillis() / 1000;

        response.setContentType("text/html");
        response.getWriter().println("<h1>Unix Time</h1>");
        response.getWriter().println("<p>Current Unix time: " + unixTimeSeconds + "</p>");
        response.getWriter().println("<p><a href='\">Refresh</a></p>");
    }
}
```

**Figure 2: Screenshot of a code snippet from “Client / Server Communication” under “Server-Side Rendering”**



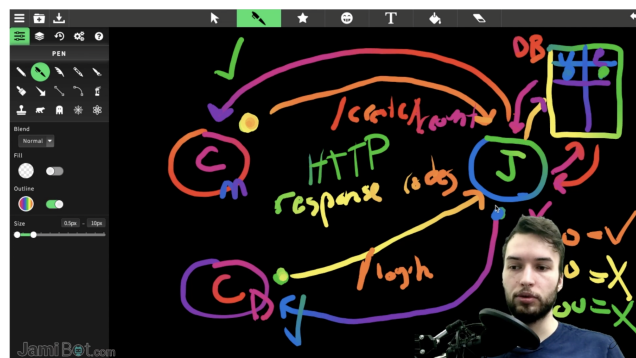
**Figure 3: Screenshot of the result of using Server-Side Rendering to show Unix Time in the browser, from ibid.**

The above examples of different types of static information are, by all means, complete, accurate, and helpful. However they have several limitations which this project seeks to improve upon. Firstly, this type of information, although available quite plainly and fundamental to understanding website architecture, is often excluded from the educational courses and dynamic learning content towards which keen programmers and students gravitate. Thereby, it is generally sought out independently, and self-taught by those who are seeking more comprehensive information on the topic. There are very few existing dynamic learning courses which dedicate at least a portion of their instruction to these web fundamentals. I believe that this is a topic which deserves coverage in a dynamic and interactive way. Secondly, this static information is quite hard to find unless

you know what you are looking for, as documentation is easy to miss, and may be inaccessible to more novice users. Because a lot of it is open-source, it is hard to tailor this information to students of different experience levels, and thus the learning journey of each student takes on a self-guided form. Put another way, this documentation exists in the form of information about functionality, and is not written from the perspective of teaching it. There are however, as mentioned before, several features of this static information which are educationally valuable, which I took guidance and inspiration from when choosing the educational content for this project, as discussed below in the next section.

The second medium through which the concept of client-server interaction can be learned online is in educational video courses on streaming websites such as Youtube. Because these are generally made by independent instructors, and for the most part exist without a paywall, they are generally more accessible to a wider range of audiences. The videos can range from 2 minutes to half an hour, and are easily digestible at the user's own pace. Figure 4 contains a screenshot of one of these longer-form videos, entitled "The Mysterious Backend To Frontend Connection - Client Server Architecture"[5]. The instructor is clearly visible on screen in the lower-right corner, as he talks through the concepts of client-server interactions and sketches diagrams on the screen. The visualisations he uses to present the content are dynamic, and the video allows the learner to engage in the content he is explaining. At present, videos like this are the most visual form of instruction on the concept of client-server interaction which is available for self-teaching. It is easily digestible, however as with the written documentation model that we

have discussed above, this model of teaching the concepts also has its own limitations. Firstly, although the content is visually engaging, the learner cannot interact with it and thus learn actively, unless the video also references a follow-along guide or DIY tutorial website which the learner can make whilst they are following along with the video. Secondly, each of these videos is a standalone, one-piece project which cannot be modularised or compartmentalised in any practical way. As a result, some of the information in these videos may become outdated with time, and it will not be possible to amend the outdated content in sections of the video without replacing the entire video and creating a new one. In other words, it is not possible to maintain these video projects, and thus they may not always be useful or complete as newer technologies come into existence. Thirdly, due to the nature of the video format it is not possible to include everything which is relevant to the topic, and it is not possible to interact with the audience of the video to improve or amend the work until the video has been posted in its completed form.



**Figure 4: Video still at roughly 14:01 of “The Mysterious Backend To Frontend Connection - Client Server Architecture”.**



## **3. Approach**

### **3.1 Overview**

The learning tool is an online collection of short animated guides, each of which teaches an aspect of client-server interaction. Each guide is short and digestible, and contains animations which are made using two React animation libraries: react-spring and Framer Motion. The guides use educational information which already exists online in the form of open-source documentation, but it is enlivened and made dynamic through the use of visuals and animations to enhance the learning experience. At the end of each guide is a quiz consisting of a few questions which aim to test the knowledge gained by the learner. This encourages the learner to retain the information they just gained from the guide. I believe that, because this tool focuses on providing the best possible educational experience and therefore seeks to deliver the information in a way which prioritises this experience, it will benefit the learner more than if they were to just read the static information online.

### **3.2 Learning Path**

The educational information taught on the website is split up into five courses. Each one is short and complete, covering a topic which is outlined at the beginning of each course. Figure 5 shows a screenshot of the course directory on the homepage of the site. A user can click on any of the icons to be taken to the first page of that particular course. Note that the colours simply indicate whether a logged in user has not started, started but not

completed, or completed a course via a traffic light colour-way system. Also note that the courses do not have to be completed in order, and that no course is a prerequisite for another. This is a flexibility feature which allows the user to explore the courses and learn at their own pace or take their learning path in their own direction. I considered the idea of restricting course access only to those who had completed the relevant prior courses, however with the small number of courses on the site I don't think that such a feature is necessary at this stage.



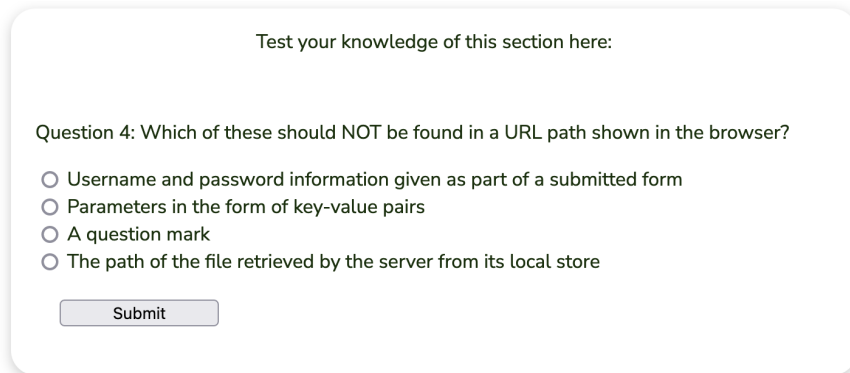
**Figure 5: Links to the courses on the website's landing page**

### 3.3 Quizzes

Testing the user on their understanding of the guides is a crucial part of the learning process, and allows the user to reflect on the knowledge they gained from the previous course. This part of the approach to teaching the concepts of client-server interaction makes this learning tool stand out from the current ways of learning this material.

Because static information found online is not written from the perspective of successfully teaching the material, the user must truly self-learn the content, create methods of ensuring their understanding of it, and thus devising their own methods of reflection on it. The guides therefore benefit from not only being short and devoted to teaching the precise information on client-server interaction, but they also acknowledge the audience - a learner of this information - and create a solution for them to be able to

test their knowledge. Figure 6 shows the interface for a quiz question from Course Two. All the quizzes are multiple choice, allowing the learner to reflect and reason why a certain answer is correct. All of a quiz's content is derived from the content of its course. The implementation of the quiz section, as well as the workflow through the quizzes, and the cases for wrong or incomplete answers, is also covered below.

The image shows a quiz question interface within a light gray rounded rectangle. At the top, it says "Test your knowledge of this section here:". Below that is the question text: "Question 4: Which of these should NOT be found in a URL path shown in the browser?". There are four radio button options: "Username and password information given as part of a submitted form", "Parameters in the form of key-value pairs", "A question mark", and "The path of the file retrieved by the server from its local store". At the bottom of the options is a "Submit" button.

**Figure 6: Multiple Choice quiz question from Course Two**

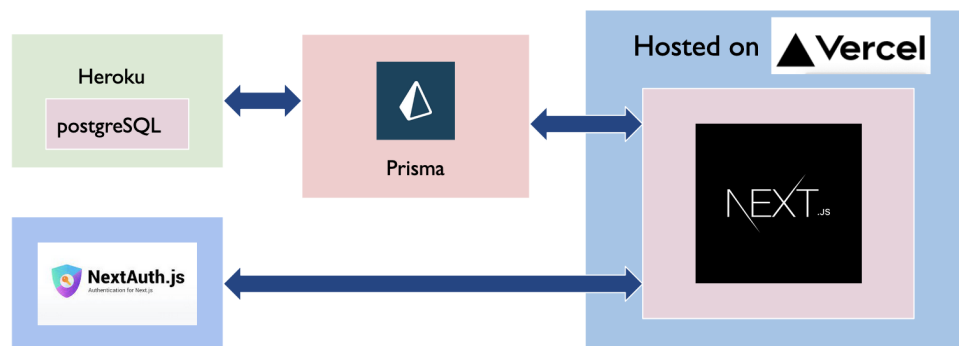
### 3.4 Animation Choices

Animations in the course are mostly triggered by the user interacting with the site through various gestures (like hovering, clicking, and panning). I implemented a selection of different animations, described in detail in the implementation section below, based on gestures I thought made sense to the course content. For example, miming the act of sending a request from the client to the server, I created an animation which would be triggered by the user dragging their mouse across a portion of the screen. This tight coupling of course content and physical gestures made by the user would make the information delivered more memorable to the user.

## 4. Implementation

### 4.1 Project Architecture

An overview of the website architecture can be found in Figure 7. The website for the learning tool is built using NextJS, a Javascript framework which allows you to build web applications using React. As the learning tool was being developed, it became apparent that it would not need to have a traditional ‘back end’ due to the NextJS server-side rendering capabilities and existing API routing solutions. There were two requirements that needed to be met in order to give the learning tool the required functionality: a user authentication and sign-in system, and a database to store quiz results and scores. It was possible to, with the help of the API routes architecture provided by NextJS, directly connect the database and authentication technologies to the front end, using Prisma as the object-relational mapping framework.



**Figure 7: Website Architecture Diagram**

NextAuth.js is the authentication framework, specifically built for NextJS, which would be used to authenticate the user and store a session cookie. It interacts directly with the

NextJS framework. The database to store user data, such as authentication tokens and quiz results and scores is Heroku PostgreSQL, and the NextJS frontend uses the API endpoints and Prisma to make POST and GET requests to retrieve the user data that is displayed on screen.

## **4.2 Workflow of the Learning Path**

When a user clicks on a course, they are taken to the start page of that course. From there, most of the course follows a linear pattern, and there are almost no clickable links to diverge the user from the flow of the course. When the user has gotten sufficiently far in a course, clickable links for the quiz portion of the course appear, and the user can also click the title of the website, “Frontend to Backend”, found in the navigation bar at the top of each page, in order to get back to the main landing page for the website. This workflow is achieved in the following way: each course has its own ‘page’ - in terms of the NextJS framework this is a React Component which has an associated route and URL path based on its file name[6]. The course is composed using React components which are rendered on the page, depending on where the user is in the course. As the user moves through the course, they will be guided to interact with relevant parts of the components, i.e by toggling switches or moving sliders, and new components will be rendered as other components leave the DOM. Each course has one, singular workflow which ends with a quiz, complete with microflows or additional branches which contain more information.

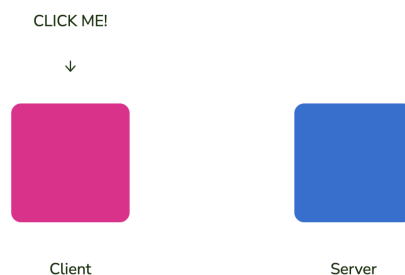
Each quiz is also rendered on its own NextJS page. When the user clicks on the ‘go to quiz’ button at the end of each course, they are routed to the page of the first quiz

question. The result that the user gets from the quiz (for example, an increase in their overall score if they get one of the questions right), is sent in a POST request via an API endpoint to the database to be stored. As the file architecture currently stands, each quiz has its own separate quiz page. However, given that NextJS has effective dynamic routing functionality, which would allow the URL path parameter to be generated dynamically depending on which quiz the client-side is requesting to route to, it would be necessary to refactor the code which generates the quizzes in future iterations of the site, especially if the site is to expand and become more complex (i.e if more quizzes or more courses are to be published).

### 4.3 Design Features

Visual and animated features were important to the delivery of the material in the course. Each course had several such features within it in order to make the course more interactive and make the material more interesting and engaging. I used two React animation libraries, Framer Motion and react-spring, to implement my animations. React-spring uses the principles of spring-physics, as opposed to the classical approach using time and duration, to power animations. It is built on the principle that spring motion more closely resembles the ‘continuous, fluid interactivity’ which is witnessed in animated objects in real life[7]. I liked the idea of using spring motions, rather than durations, to synchronise collections of animations. An example of one of the animations I implemented using react-spring was a hover-shake, inspired by the hover transition known as *Boop!* which I found when researching react-spring animations on the web[8]. Figure 8 shows a screenshot of the icon representing the Client in one of the courses in

the learning tool titled *Course One: Client and Server - what are they?*, which itself is wrapped in a hover-shake component. When the user hovers their mouse over the Client, it will rotate clockwise and counterclockwise around its centre and come to a stop, overall moving in a way that resembles a shake. The hover-shake component is made using the useSpring hook from react-spring. This hook has configurable properties, or props, which can be changed to fine tune the behaviour of the spring which governs the motion of the component it creates. For hover-shake, I experimented with changing the tension and friction of the spring, until I got the shake velocity and duration that I wanted. Other changeable spring properties include mass, clamp (when set to true, it stops the spring from “overshooting its boundaries”), and damping [9] . I created hover-shake as a separate component which would be able to wrap around any React component or icon, and thus be reused to create the same shake-on-hover movement. In future renditions of the website I would, as what was done with the original *Boop!*, refactor the hover-shake and make it into a React custom hook to allow for different spring configurations for different uses [8].



**Figure 8: The Client icon in Course One**

I used Framer Motion for a wide range of different animations on the site. I experimented with different gestures that the user could do to interact with the components on screen. Framer Motion provides animation support for gestures such as drag, pan, hover, click

and focus, and allows temporal adjustments and settings, such as delaying animations, or activating them based on certain combinations of gestures. Figure 9 shows an example of a click-and-drag component in Course Three accompanied by a written instruction telling the user to “drag [the slider] to open a connection with the server and send a request”. Upon interaction, a new component would be revealed after an animated delay which gives information on how the server listens to HTTP requests, and what it does when it receives them.

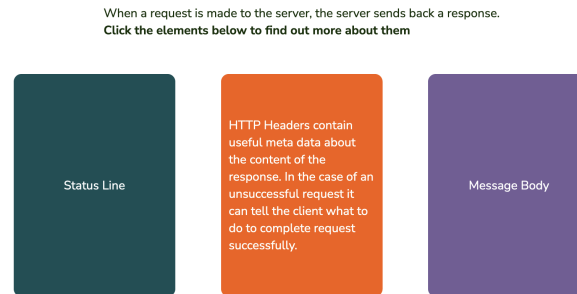
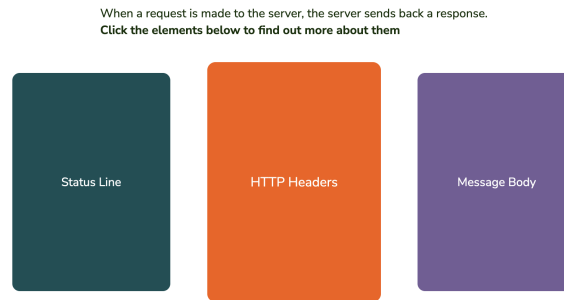
Drag to open a connection with the server and send a request



**Figure 9: A custom-built slider, animated using Framer Motion’s gesture tracking properties**

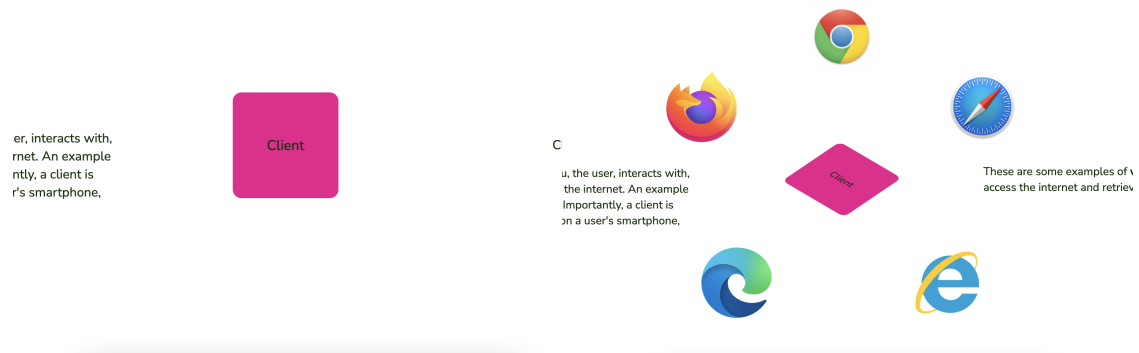
I combined Framer Motion’s gesture recognition system with React’s `onClick` and `useState` hooks to create complex transitions and animations. Figures 10.1 and 10.2 show the before and after of clicking a box, with the box flipping 180 degrees about the x-axis and displaying text explaining what HTTP Headers are on the reverse side. The animations occur after the `onClick` hook has been triggered, with the state of the box (whether it has been clicked on or not) stored in a `useState` hook. The animation variants listen for this hook to receive the cue that they must activate.





**Figures 10.1 and 10.2: Demonstration of a ‘card-flip’ animation being activated by the user clicking on the card.**

Another interesting challenge which I worked on was combining CSS transformations with Framer Motion animations. Figures 11.1 and 11.2 demonstrate before and after clicking on the Client icon in Course One. When the user clicks on the icon, hidden icons of browsers appear and position around it, creating a geometric formation on the screen, whilst the icon itself rotates about the Z and X axes, creating a 3D illusion of the shape in space. If the user clicks on the icon again, the browser icons fluidly retreat to behind the main icon once again. I had to experiment with CSS positions and the z-index, along with transformations, in order to animate this transition.



**Figures 11.1 and 11.2: Click gesture activating a series of parallel transitions**

The whileHover hook supplied by Framer Motion also proved useful as I strategized how to nudge the user to proceed with the workflow of a course. Whenever a user needed to click or interact with a certain component in order to proceed with the course, along with a written cue such as “click here”, that component would be accompanied by an animation, such as a scale change or the hover-shake discussed above, which would confirm to them that they need to interact in some way with that component in order to proceed.

In summary, the design choices made to do with the animations implemented through the website and its courses sought to fulfil two goals: enhance the user’s learning experience of the content by making it interesting to engage with, and supporting the workflow of the website and courses, making the website user-friendly, easy to navigate, and interactive.

## 5. Evaluation

### 5.1 Approach

This learning tool is aimed at people who are generally interested in making a website for the first time, and have some prior experience with either web development, or coding in general. I decided that current and past COS 333 students were an ideal target user group for evaluating this project. As discussed above, this is because the course content of COS 333 provides a limited introduction to web development, but to gain broader insight one needs to self-teach some of the concepts.

In order to evaluate this course, I acquired 5 participants who had taken COS 333 in the past, and expressed interest in learning more about client-server communication. I informed them that this website was a prototype for a potential future learning tool for full stack web development. I asked them to create an account on the website, and complete the first three courses.<sup>1</sup> I then asked them to fill out a survey form, which consisted of a combination of questions about the quality of the website and their experience using it, and questions asking them to assess the quality and information given in the courses. I asked them to report their quiz results, and their opinion of the difficulty of the quiz questions. I also included an open-answer question at the end, leaving feedback on how the site could be improved in the future, and other features which could be added.

---

<sup>1</sup> At time of assessment, Course Four and Course Five were still in development and not ready for assessment.

## 5.2 Results

All 5 test users signed in to the website successfully and completed the first three courses. All reported a total score of 6, which is the maximum total score which can be obtained from using the website. A piece of feedback received from one user was that, when he got one question wrong, he was simply able to work through the course again and complete the same question with the correct answer. They suggested that it would be beneficial for future users of the site to make it more difficult to obtain the correct answer and thus gain the maximum possible score in this way.

In the survey form, I included questions asking the test users to assess the quality of the animations from each course, by ranking each courses' overall animation quality and value on a 1-5 scale. Course Two received the highest average score of 4.2. In an open-answer question asking about animation feedback, one user said they liked the variety of animations, and that they were clear and simple to interact with. Another user provided feedback that it would be great to see even more animations in each course.

In terms of the course content, I asked each user to specify on a 1-5 scale, where 1 meant 'Strongly Disagree' and 5 was 'Strongly Agree', whether they found the material given in each course useful or meaningful in terms of helping them understand full stack web development and client-server interaction. Another question asked the users to rate on the same scale whether they learned something new from the course. The average scores were 4.4 and 4.2 respectively, meaning that all users either agree or strongly agree with the statement that the course material was useful, and that all users felt that they had learned something new, with one user feeling 'neutral' (i.e rating with a 3) on this statement. In an open-answer feedback question which asked users to expand on their

choice of ratings for the above questions, a user expressed that they had learned something new from the course, and that they hoped that in future iterations of the learning tool there would be more code examples alongside the materials given in the courses. Another user said that they found it helpful that there were quiz questions directly related to the material in each course. They remarked that it helped them to think more about the information given, rather than just ‘passively reading [it]’.

Overall, I believe that the evaluations given by the test users indicate that the learning tool was beneficial to them, helped them learn something new regarding client-server communication, and did so using tools such as gesture-triggered animations which enhanced the learning process.

## **6. Conclusion**

### **6.1 Summary**

It is clear that a project like this would benefit many interested programmers who would like a more dynamic approach to their learning path through the fundamental concepts of full stack web development. Currently, people who wish to learn about the fundamental principles of client-server interaction must seek out static guides and documentation in order to find out the most comprehensive information on the topic. This project provides a practical prototype of a solution to the challenges associated with learning from static content in this way, by taking that content, synthesizing it, and creating interactive and engaging guides to help users learn it. This project uses visuals and animations triggered

by the user's gestures, as well as quizzing functionality, thus making the learning experience more interactive for the user.

## **6.2 Limitations**

This project takes the form of a prototype for a potential future learning tool which would contain much more content, curated and created carefully with user input and feedback. During implementation, I found it difficult to fit in every module and component I thought would benefit the courses, and I was constrained by both experience and time. Nonetheless, in its current form, the website takes on the shape of a promising model of a learning tool which can exist in the future in order to educate motivated learners on the fundamentals of client-server interaction. I would like to acknowledge that there are limitations, therefore, on the content, and also the structure of the backend (how the data is stored), which would need to be refactored to make future iterations of the site as successful as possible.

## **6.3 Future Work**

Along with the improvements detailed throughout this paper on various ways that the current functionality can be improved upon, this project would benefit from an expansion of material beyond its current focus on client-server communication, focusing on other fundamental aspects of full stack web development, such as organising databases, improving the security of websites, and implementing effective and clear user interfaces. It would also benefit from some gamification, such as implementing a leaderboard or other incentives to encourage users to aim for high scores in the quizzes. The current

aesthetic design of the website is also somewhat limited, so it would benefit greatly from an overhaul and creative design process of its user interface to encourage users to return to it, and see it as a reputable and engaging learning tool.

## **7. Ethics**

This Senior Independent Work project and report represents my own work in accordance with University policies and regulations.

## Bibliography

- [1] R. Dondero, “Course Description” in *Computer Science 333: Advanced Programming Techniques*. Last Updated: Fall 2021. Retrieved from: <https://www.cs.princeton.edu/courses/archive/fall21/cos333/>.
- [2] H. Willee and wbamberg [GitHub username], “Client-Server Overview” in *Server-side website programming*. Last Modified November 24th 2021. Retrieved from: [https://developer.mozilla.org/en-US/docs/Learn/Server-side/First\\_steps/Client-Server\\_overview](https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview)
- [3] K. Workman, “Client / Server Communication” in *Server Tutorials*. Last Modified: 12th August 2021. Retrieved from: <https://happycoding.io/tutorials/java-server/client-server>
- [4] Various Authors [Open Source], “Request” in *API Reference*. Last Modified: Unknown. Retrieved from: <https://expressjs.com/en/4x/api.html#req>
- [5] J. Saunders [Jameson Saunders], “The Mysterious Backend To Frontend Connection – Client Server Architecture”. Uploaded on 8th April 2019. Retrieved from: [https://www.youtube.com/watch?v=7qHJSXLn\\_2c](https://www.youtube.com/watch?v=7qHJSXLn_2c)
- [6] Various Authors [Open Source], “Pages” from *Documentation*. Last Modified: 13th September 2021. Retrieved from: <https://nextjs.org/docs/basic-features/pages>
- [7] Various Authors [Open Source], “Why springs and not durations” from *Introduction*. Last Modified: Unknown. Retrieved from: <https://react-spring.io/#why-springs-and-not-durations>
- [8] J. W. Comeau, “Boop! A whimsical twist on hover transitions”. Last Updated: 10th June 2021. Retrieved from: <https://www.joshwcomeau.com/react/boop/>
- [9] Various Authors [Open Source], “Configs” from *API*. Last Modified: Unknown. Retrieved from: <https://react-spring.io/common/configs>