

Decibel: Instant Text to Speech Web Application

CSE 518 Final Project Name: Atharva Kadam SBU ID:111593743

I. Abstract

Over time the text to speech (TTS) has improved and that has resulted in a lot of fun TTS applications to improve user experience. While there has been advancement in speech synthesis, there haven't been a lot of applications that provide a plug-and-play interface with high accessibility value that can be utilized by a wide range of people; from students studying for tests all the way to people with visual impairments. Decibel is a web application that implements a clear, concise, user-friendly, user interface that instantly converts any input text to speech in 67 different accents from around the world.

II. Introduction & Background

Speech Synthesis is being used as a substitute for different forms of auditory entertainment such as music, podcasts, entertainment, and others[3]. TTS has been mainly used to replace audiobook reader applications by providing different voices from different countries. But there aren't any general applications for using this audiobook-like feature and the power of TTS libraries/APIs that can not only be used to read a given pdf but can read anything with the help of a plug and play-oriented interface. The idea is very simple, and it is to make a simple and user-friendly user interface of a web-based application that can simply take in any text as an input and immediately perform speech synthesis on it. Now, the question obviously would be why do we need this application, if an audiobook reader application does something very similar. So the answer to this question is that we are trying to generalize an

interface/application that can not only be used by audiobook enthusiasts but also by anyone who needs a quick TTS solution. With the clean, concise, minimalistic user interface along with visual cues and high accessibility scores, my project attempts at providing an instant text-to-speech solution to a wide range of people. The accessibility part is added to it to make it more accessible and easy to use for people with visual impairments. Technology has developed a lot over the last few years and that has helped make the world more accessible to all demographics of people. With advancements in technology, it is our responsibility to make technology easier for people with special needs. So keeping this in mind, my project also provides a lot of visual cues to the users with visual impairments. We also know that learning with TTS results in increased attention and retention for students[2]. So everyone including audiobook readers, students studying for their test, people with visual impairments, and people learning a new accent, can use this web application to perform TTS. In order to evaluate the model, I performed qualitative as well as quantitative studies. As a part of the qualitative study, I used the ten principles of heuristic evaluation as my design principles. I did not get an opportunity for experts/evaluators to test my interface but using the ten heuristics, I verified if my interface was passing most of these cases. As a part of the quantitative study, I collected data from five of my friends from different backgrounds to check what they think of the interface and rate the interface for each of the 5 metrics: Ease of use, accessibility, elegance, performance, speed. My contributions are as follows:

- 1) Built a minimalistic user interface to run the text to speech API using JavaScript, React JS, React Bootstrap, HTML, and CSS
- 2) Add additional options to control the rate and pitch of TTS to increase user control and add more functionality
- 3) Performed a qualitative and quantitative analysis on the interface to validate the initial idea/hypothesis

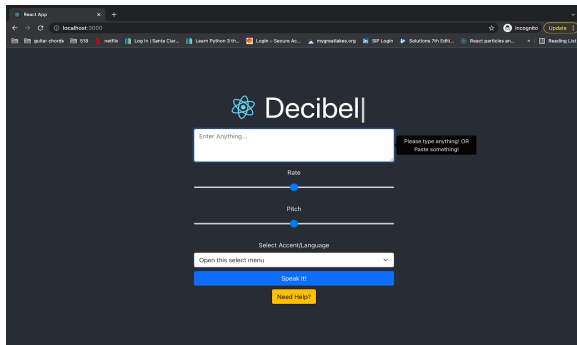


Fig 1: The home/landing page of the UI. The tooltip is visible for the input text box. The tooltips are triggered when you hover over certain parts of the UI.

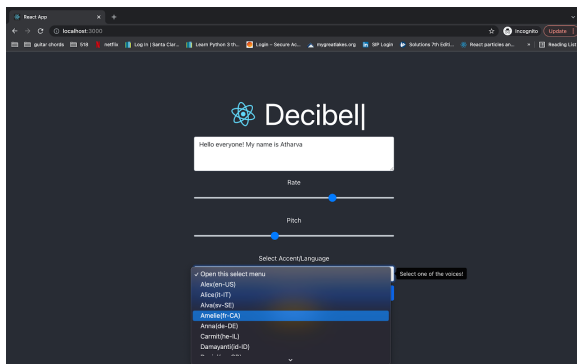


Fig 2: Dropdown for voices/accents showing up with some values set for input text & pitch

III. Related Work

In order to come up with the idea for the project, I had to perform some level of research on articles, libraries, and APIs for

text to speech. One of the first things that I found was the Google Text to Speech API. Initially, the idea was to use this Google Text to Speech API since it had support for over 220+ voices in 40+ languages[1]. But on further research, I found google had a limit on the total characters sent to synthesize each month, and that discouraged me from using it. The base idea for my project was to use free open-source libraries and APIs to perform a decent level of TTS. So to keep consistent with this idea, I found another free JavaScript TTS library called Web-Speech API by the Web MDN Docs. This API has two parts - Text To Speech with SpeechSynthesis and Asynchronous Speech Recognition with SpeechRecognition.[6] Since I am currently only doing text to speech, all I needed was the SpeechSynthesis part of the API. In order to make my application more accessible for people with visual impairments, I followed certain contrasting color schemes as well as tooltips and text as well as voice cues wherever possible. On reading multiple articles, I figured out how to implement a UI that can handle different types of impairments such as loss of acuity, peripheral vision, ghosting vision, and large spots. [5]. Using TTS can be beneficial for learning amongst kids (specifically students) because of increased focus and retention since more than one sense is focused on studying[2]. An essential part of my implementation was implementing the UI in a clean, concise way by making the UI responsive. To do this I used React Bootstrap along with React JS, JavaScript, HTML, and CSS.[8][12]. Finally, in order to deploy it, I used netlify which seamlessly performs CI/CD with your GitHub repository whenever you push to your main branch. [4]

IV. Methodology

To build this application the most necessary thing was the JavaScript API, so to make this web speech API run, I decided to build this web application project using JavaScript, HTML and CSS. I have a lot of experience designing applications and building front end in React JS with the help of UI frameworks such as Material UI, React Bootstrap, PrimeReact, etc. So in order to keep a clean minimalistic UI, I decided to go with React and React Bootstrap. To begin, I ran the following command:

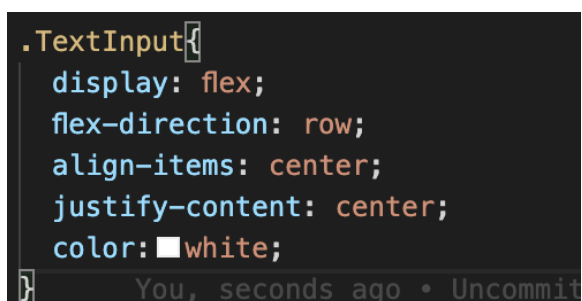
- `npx create-react-app`

The above command creates a new react app with node modules and installs all dependencies and sets up the package.json file. It also has a default code in the App.js file which is called index.js. Before I started working on App.js, I first installed all the required libraries which included:

- `react-bootstrap`
- `React-typing-effect`
- `react-bootstrap-range-slider`

After installing the above libraries, I started working on App.js and App.css. React Bootstrap will make most of the UI responsive because of its responsive UI components, but there are a few changes that still need to be handled in terms of CSS. So this was done in the App.css file by creating custom class names which were later used to align items correctly. The first thing that was needed was a basic layout that would encapsulate how the application would look as a whole. So to do this, I first figured out that I needed only an input and output box because of the intended plug-and-play

nature of this application. So the easiest way was to follow Occam's Razor and make the layout intuitive by making it top to bottom navigation. Another important thing to keep in mind was to center objects on the webpage. This was to accommodate for the visual impairments (the ones listed before) which mainly made the outer parts of the web page out of focus. So the best way to deal with it was to center align all the elements of the page. Now the CSS for center aligning looks as follows:



```
.TextInput {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: center;
  color: white;
}
```

Fig 3: CSS Class to align items to the center in order to accommodate visually impaired users

Next, I used React hooks such as `useState` and `useEffect`, to manage and handle the state of the objects in the application. So I used these react hooks in the App.js file. After designing the barebones UI with the structure and using components from React Bootstrap, I then moved on to actually calling the JavaScript Web Speech API. So to do this, I had to write a function called `speak`, which is triggered every time the user presses the 'Speak It' button. So this function is the most crucial part of the application as it makes the API call. The `speak` function is as follows:

```
const speak = () => {
  if (synth.speaking) {
    setShowSpeakAlert(true)
    return;
  }
  if (text === ''){
    setShowAlert(true)
  }
  if (text !== ''){
    const speechText = new SpeechSynthesisUtterance(text);
    // end of speech
    speechText.onend = e => {
      console.log('Done speaking!')
    }
    // speech error
    speechText.onerror = e => {
      console.error('Something went wrong!')
    }
    voicesList.forEach((eachvoice) => {
      if (eachvoice.name === voice){
        speechText.voice = eachvoice;
      }
    });

    speechText.rate = rateSlider;
    speechText.pitch = pitchSlider;

    synth.speak(speechText)
  }
}
```

Fig 4: *Speak function which makes all the calls to SpeechSynthesis part of Web-Speech API*

Voices are obtained asynchronously from `window.speechSynthesis.getVoices()`. So to handle that case, I had to use a promise and a `useEffect` to resolve it using the `await` and `async` keywords, and then after the promise gets fulfilled, I assign all the voices to the state variable I have for it. The code for handling the asynchronous nature of `.getVoices()` is as follows:

```
const allVoicesObtained = new Promise(function(resolve, reject) {
  let voices = window.speechSynthesis.getVoices();
  if (voices.length !== 0) {
    resolve(voices);
  } else {
    window.speechSynthesis.addEventListener("voiceschanged", function() {
      voices = window.speechSynthesis.getVoices();
      resolve(voices);
    });
  }
});

console.log(allVoicesObtained)

useEffect(() => {
  const fetchAllVoices = async () => {
    const voices = await allVoicesObtained;
    setVoicesList(voices)
  }
  fetchAllVoices();
}, [])
```

Fig 5: *Promise resolving using useEffect*

The rest of the technical approach is straightforward as callbacks and events are set up in the UI for every action that the user

takes. A few last additions (everything after the demo video presentation submission) included adding a help modal, which shows all the instructions and even says the instructions out loud. Similarly, there are voice cues added for all text on the UI including the title. On clicking Decibel, the 'Welcome to Decibel' message is heard. There are useful voice cues for all other parts of the UI. Tooltips are also displayed for better visual accessibility. Instead of the default `window.alert` command, I now use nicely formatted and color-coded alerts by React Bootstrap to alert the user when an error is made to guide them in the right way. There are voice buttons for the alerts too. In order to implement the saying it out loud functionality for voice buttons, I simply wrote a function called `sayitoutloud` which basically takes in an input text and then speaks out the input text. This function was called with the `onClick` event handler for any of the text on the UI. The `sayitoutloud` function looked as follows:

```
let sayitoutloud = function(msg){
  var speech = new SpeechSynthesisUtterance();
  speech.text = msg;
  window.speechSynthesis.speak(speech);
}
```

Fig 6: *sayitoutloud function which takes in any input text and reads it out loud.*

V. Evaluation

Now in order to do an evaluation for this part, I decided to do it in two parts - a Qualitative evaluation Study with the help of Heuristic Evaluation (Without evaluators/experts) and a Quantitative Evaluation Study where due to lack of people to survey due to covid, I performed a short informal survey between 5 of my friends to get their opinions on few decided evaluation metrics.

Qualitative Evaluation Study:

So for this study, I decided to use the Heuristic Evaluation process which involves checking or verifying your UI against the top ten heuristics as stated by the heuristic evaluation process. Following are the 10 heuristics and explanations whether that heuristic was followed or not:

H2-1: Visibility of System Status:

It is important to know the visibility of the system status at all points. An example of this type of heuristic would be a progress bar. But in my case, the action or the way the system responds is instantaneous, so no progress bar or any kind of hourglass is required while the output is processed. But my system does throw an alert when a user tries to run the system again while it's already reading out some input text. So in that case, my system notifies the user that the system is currently talking, so please wait for it to finish. So in this case you can say that the system lets the user know about the visibility of the system status. So yes, my system follows this heuristic.

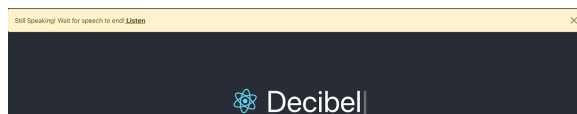


Fig 7: The alert message shows up at the top of the screen when the user tries to click on the 'Speak it' button while the system is already speaking.

H2-2: Match System and Real World:

So my system does not have anything explicitly matching a real-world metaphor. Something close to that is the way my instructions are given in the help section. There is a button that reads out the instructions to the user like how someone

would explain how to use a new interface in person. So in some real-world situation, if I had to explain to someone how to use my web application, I would do it this way. So in that way, my system matches the real world and thus I can say that in some way, it still matches the system with the real world.

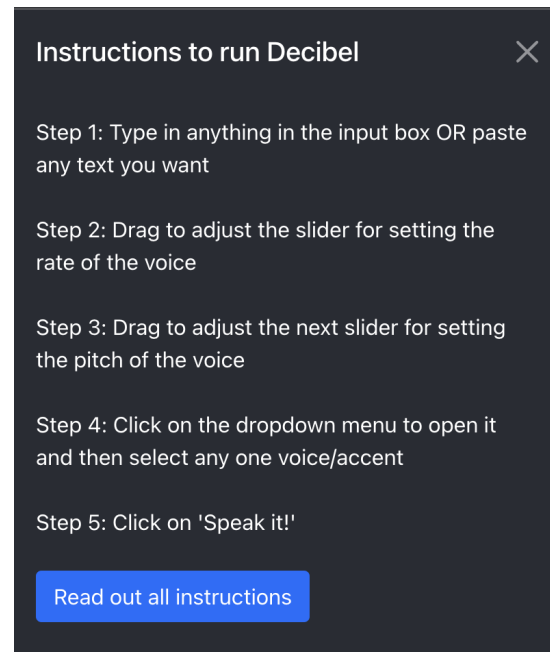


Fig 8: On clicking the 'Read out all instructions' button, the UI reads out the instructions like an actual instructor would do in real life.

H2-3: User Control and Freedom:

So for this part, the users have full control over how they write in the input text box, so they can perform all sorts of undo, redo, and other text operations that let them have more control and freedom. Similarly, users are not restricted to any given text or specific formats, they can use any text they want or can even have the option to type. The users are not bound by anything and can simply reload or revisit at a later time and all they have to do to get started again is simply to paste some text or type it in. So that is why

there is a lot of freedom and control for the user and so my UI follows this heuristic too.

H2-4: Consistency and Standards:

So for this heuristic, it is important that consistency is maintained throughout the application. In my web application, I have maintained consistency by using the same design classes for all the buttons without changing the shadows or other CSS effects for the buttons and the other parts of the baseline UI. Similarly, I also strive to maintain consistency throughout the other parts of the application. One such example would be the way I listed out all the steps to carry out the task or instructions on how to run the program. So I kept the same consistency in listing out all the accents too in the dropdown menu. The accents in the dropdown menu are sorted. This kind of consistency throughout the application really captures the essence of the web app as well as truly validates this heuristic

✓ Open this select menu

Alex(en-US)
Alice(it-IT)
Alva(sv-SE)
Amelie(fr-CA)
Anna(de-DE)
Carmit(he-IL)
Damayanti(id-ID)
Daniel(en-GB)
Diego(es-AR)
Ellen(nl-BE)
Fiona(en)
Fred(en-US)
Ioana(ro-RO)
Joana(pt-PT)
Jorge(es-ES)
Juan(es-MX)
Kanya(th-TH)
Karen(en-AU)
Kyoko(ja-JP)
Laura(sk-SK)
Lekha(hi-IN)

***Fig 9:** The accents in the dropdown menu are in sorted order thus maintaining the consistency and standard of the application.*

H2-5: Error Prevention:

Normally error prevention would be present in case of incorrect input by doing some sort of input validation, the kind that is usually seen in login/authentication systems. Now in this project, there is no need for a login system because of the design choice made in order to make it more plug and play. But nevertheless, error prevention is necessary, and I handle errors in my application such as the error where one cannot listen to something new when the previous thing is still playing. So in that case, the user gets an alert telling the user to wait. Similarly, the user is not permitted to paste anything but text into the input text box thus performing input validation. In the case of the drop-down menu and the rate and pitch slider controls, it was important to implement it in that way instead of giving an input text option because this way, the user cannot make an invalid input since he/she has to pick from one of the given valid options. That is how error prevention is implemented in this application.

H2-6: Recognition rather than recall:

So for someone who uses this application every day, the application is very straightforward and can easily be mastered by anyone. But because of its accessibility options, it is very easy to learn. This is because of all the tooltips provided with voice cues. All these options together truly bring make the UI handle this case where it is more about recognition rather than recall.

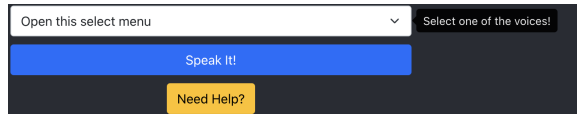


Fig 10: *The tooltips help users recognize rather than recall.*

H2-7: Flexibility and Efficiency of Use:

The application provides help and support in all ways possible. But at the same time, it also provides flexibility to expert users in order to maximize speed and improve efficiency.

H2-8: Aesthetics and Minimalist Design:

This application is all about Minimalistic design and aesthetics. From the get-go, I have explained how making this design minimalistic and easy to use while making it look appealing and accessible was the main goal of the assignment. To use principles learned in HCI to make the UI more accessible and reachable to more people. So yes, this heuristic too, is passed by my UI.

H2-9: Helps user recognize, diagnose and recover from errors:

We saw an example of this earlier where my application throws alerts in two situations - if the user clicks on 'speak it' while the system is already speaking and if the user clicks on 'speak it' before the user puts anything in the input field. In both cases, an alert is thrown with a clear message of what the user is doing wrong and how to fix it. The user also can play the voice button to listen if necessary.

H2-10: Help Documentation:

As shown earlier, when you click on the

'Need Help' button, a side modal opens up which explains how to use the UI in step-by-step instructions. In case you want to listen to these instructions, you can simply click on the 'Read out all instructions' button to listen to all instructions as you try and follow them.

So this way, I can conclude the qualitative studies or heuristic evaluation and show that all ten heuristics were followed as design principles in the application and even though there were no evaluators for this study, a great deal of comparison could be drawn between the way the UI is implemented and the heuristics which shows that the qualitative evaluation study of my project is positive and that is why the results of how good the UI is can be validated.

Next, we perform a very short quantitative study (as not a lot of data is available) to quickly show what users thought of the UI.

Quantitative Evaluation Study:

I asked 5 of my friends to do a simple informal survey where I gave each of them a few excerpts from my favorite book. Some excerpts were really long, some were short, I told them they could either use the excerpts I provided or they can use any text they want and use my application to perform TTS a couple of times with different accents (Maybe even in accents they are familiar with based on their family background). After playing around with it, I asked them to rate the UI based on 5 individual factors which are - Ease of use, accessibility, elegance, performance, speed from 1 to 10 where 1 is the worst and 10 is the best. So the results I received for this study from them is shown in the following table:

User Number	Ease of Use	Accessibility	Elegance	Performance	Speed
1	9	10	10	8	10
2	9	10	8	10	10
3	10	10	8	9	10
4	10	10	9	10	10
5	9	9	8	9	10
Total Average Rating	9.4	9.8	8.6	9.2	10

Now of course this data is by no means enough to validate the UI comprehensively. But it is the best given the current covid situation as well as the general expensiveness of a quantitative study. So based on this data, the goals that I had set out to achieve at the beginning of the projects were achieved and validated. A speed of 10 validates the idea of really fast and instant TTS. Similarly, a very high score of 9.8 for accessibility shows that a lot of effort into making it more accessible paid off. At the same time, performance and ease of use have high scores too. The only one that is lacking is Elegance which makes sense as the reviews for these users mentioned that they tried running it on a mobile phone browser. It makes sense that elegance went down with a mobile phone browser, but in the long run, this still makes sense as it was able to work everywhere and was responsive to some extent.

So overall I would call it a very successful evaluation phase - both qualitatively as well as quantitatively, as all goals seemed to have been achieved and validated.

VI. Discussion and Future Work

So after performing experiments to evaluate the interface, the initially intended results seem to have been achieved to some extent. In the first instance, the idea of just doing simple. instantaneous, plug-and-play text to speech would seem trivial, but because of this project, it can be seen how so much more can be done on it to make it work for a plethora of cases and a diversity of people. This application can be used by a student to improve their retention while learning. It can be used as a replacement for any kind of auditory entertainment like music, audiobooks, and podcasts. I am an international student, so even though I was accustomed to learning in English, I was used to learning in a different accent. Sometimes that can play a factor in how fast one is able to grasp knowledge. That is also a case when they can use Decibel, because of how it lets you listen to any text in any accent you prefer. This way you can understand and remember things faster. Actors or theatre students can use this to learn different accents. People studying abroad can understand things better by getting a feel of the accent. People can use rate and pitch controls to do any number of innovative things with accents and voices. Visually impaired people can use this the most by listening to text rather than having to read it, and the interface is made more accessible for them so they can benefit from this. So one can clearly see the potential of Decibel and why it is a successful project. I plan on working on this a little more by improving its interface and adding more functionality in the near future. I believe adding options for voice recognition would also make the application a little more interesting. But overall I am happy with the way this project turned out to be.

VII. Conclusion

So finally, I believe the goals of this project were achieved and validated quantitatively as well as qualitatively. The idea was simple. Make Text to speech(TTS) easier to work with and design an application that would create opportunities for people from all backgrounds to work more with TTS and use it a little more in their daily routines. The project had a few side goals which included making the project much more user-friendly and accessible for visually impaired users, as well as making it a plug-and-play kind of application. Overall I believe the application delivered on all fronts and completed almost everything that was promised at the beginning, and I am happy with the progress and the results of the project

VIII. References

- [1] Google. (n.d.). *Text-to-speech: Lifelike speech synthesis | google cloud*. Google. Retrieved December 16, 2021, from <https://cloud.google.com/text-to-speech>
- [2] *Text-to-speech technology: What it is and how it works*. Reading Rockets. (2019, October 16). Retrieved December 16, 2021, from <https://www.readingrockets.org/article/text-speech-technology-what-it-and-how-it-works>
- [3] Wolber, A. (2017, July 5). *4 text-to-speech apps that will read online articles to you*. TechRepublic. Retrieved December 16, 2021, from <https://www.techrepublic.com/article/4-text-to-speech-apps-that-will-read-online-articles-to-you/>
- [4] Written by David Wells Published in Guides & Tutorials on May 28. (n.d.). *Deploy in seconds with Netlify CLI*. Netlify. Retrieved December 16, 2021, from

- <https://www.netlify.com/blog/2019/05/28/deploy-in-seconds-with-netlify-cli/>
- [5] Chawla, P. (2021, August 5). *How to design accessibility app for visually impaired?* Appinventiv. Retrieved December 16, 2021, from <https://appinventiv.com/blog/design-accessibility-app-for-visually-impaired/>
- [6] *Web speech API - web apis: MDN*. Web APIs | MDN. (n.d.). Retrieved December 16, 2021, from https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
- [7] *React-typing-effect*. npm. (n.d.). Retrieved December 16, 2021, from <https://www.npmjs.com/package/react-typing-effect>
- [8] *Bootstrap*. react. (n.d.). Retrieved December 16, 2021, from <https://react-bootstrap.github.io/components/modal/>
- [9] Woodson, M. (2020, February 23). *Marques Woodson*. Pluralsight. Retrieved December 16, 2021, from <https://www.pluralsight.com/guides/executing-promises-in-a-react-component>
- [10] *React-bootstrap-range-slider*. npm. (n.d.). Retrieved December 16, 2021, from <https://www.npmjs.com/package/react-bootstrap-range-slider>
- [11] Bradtraversy. (n.d.). *Type-N-speak/main.js at master · bradtraversy/type-N-speak*. GitHub. Retrieved December 16, 2021, from <https://github.com/bradtraversy/type-n-speak/blob/master/dist/js/main.js>
- [12] *React – a JavaScript library for building user interfaces*. – A JavaScript library for building user interfaces. (n.d.). Retrieved December 16, 2021, from <https://reactjs.org/>

Github Repo Link => <https://github.com/atharvakadam/cse518FinalProj>

Deployed Project Link => <https://nostalgic-pasteur-2f1e32.netlify.app/>

Instructions to run the Project =>

(Note => No need to run it as it is already deployed on the above-mentioned netlify link. But in case you want to then follow the following instructions (You can follow instructions on GitHub readme too.))

Step 1 => Unzip the Project.zip file to get the finalproj directory

Step 2 => In your terminal, navigate to finalproj directory

Step 3 => Run npm install (Make sure you have npm installed)

Step 4 => Run npm start