COMP-3340 Project: Chirp

Nouman Malik
University of Windsor
Windsor, Ontario, Canada
malik63@uwindsor.ca

Danny Jr. Duronio
University of Windsor
Windsor, Ontario, Canada
duroniod@uwindsor.ca

ABSTRACT

Chirp is a vibrant web application designed to mimic twitter and offer users a platform to communicate and share their thoughts. The two primary uses of the app include: posting messages and also managing the timeline. Users can very easily create messages and also have the power to delete their own messages if they want to remove them from the timeline. The functionality of the app and the way it was designed promotes effective and engaging content to be created and curated for the platform. Next, to ensure secure and personalized access from user to user Chirp has user incorporated authentication mechanisms. Upon use of the platform all users are required to sign up if they don't already have an account. The account can be created with an email and password to contribute to the user's privacy. Also, Chirp employs the creation of a unique username for every account that will be seen by other users on the timeline when a message is sent. This eliminates possible duplicate usernames so that individuality is ensured. In conclusion, Chirp allows users to engage in conservation during real-time in a secure and user-friendly environment.

ACM Reference Format:
Nouman Malik and Danny Jr. Duronio. 2023. COMP-3340 Project: Chirp. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Twitter is one of the most popular social media sites in the world. And it is also owned by one of the richest people in the world. This has a funny outcome, where people don't like him so they make their own Twitter clone. There are about 7 serious Twitter clones on the app store right now, such as BlueSky, Mastodon, and now Threads. These types of text based apps are very simple, and therefore each company looks for their own niche market to which they can serve basically the same app. So, we thought it would be a good idea to give it our own try. This is a viable idea, as long as it's tailored for a niche market. Imagine a twitter that is just for UWindsor students. Our drive comes from our belief that if we narrow down our audience to the amazing students of UWindsor we can create a platform that connects our students and enhances their academic journey. This platform has the potential to provide a space for student interaction, allow knowledge sharing and provide a space for students to discuss. We think that us capturing that potential and giving students a platform such as Chirp this project understands and address the needs of the UWindsor students.

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELEVANT LITERATURE REVIEW

A very important part of this project was to choose which technology stack we would use to build on. A research project that we used for the decision was the comparison between Node.js, Apache/PHP stack, and Nginx. The results from this research paper concluded that Node.js has significant benefits compared to both of these other tools. It is asynchronous and event-driven, which allows for non-blocking operations. So, while you are using a Node.js web app, the other programs on your computer won't be slowing down. It also has a vast ecosystem of libraries and packages that makes developing web apps faster. In the testing, it was shown that Node.js and JavaScript outperformed PHP in computational performance, specifically, being more memory efficient and being able to utilize all available processing power. Because of this we decided to use Node.js and a JavaScript based stack for the project.

3 PROJECT DETAILS AND METHODOLOGY

For the front-end framework, we use Next.js and JavaScript. This is because, Next.js has a feature that allows for server rendering sites, so instead of sending JavaScript to the client machine which takes time to render, the JavaScript renders on the server and only sends HTML and CSS to the client which make the site load significantly faster. We used this feature for the homepage, and the sign-up and sign-in pages. For the CSS and component library we were using TailwindCSS and DaisyUI. These 2 tools allowed us to significantly speed up development. Then for the backend, we used a Postgres SQL database that is hosted on Supabase, and Clerk dev, which hosted the user authentication. The finished site is hosted on Vercel.

4 DEFINITIONS

Front-end Framework: This is a tool that allows us to build a site using JavaScript completely, which then compiles and renders to HTML for the end-user. This tool is designed to make the creation process for our web application smoother. Once constructed using the framework, a complication process occurs that converts the code done in Javascript to HTML in our case.

Backend: This is a serverless database, that allows us to have a dedicated database without a dedicated server that we have to maintain. Cloud-based services bring many positives including managing backend processes. These backend processes include: security, data storage and computation. This approach allowed us to focus more on the core functionality and less on managing the physical hardware which helped increase our efficiency.

Serverless Database: This is a type of database that leverages AWS Lambda functions to have a database always running in the cloud on multiple machines all around the world. AWS Lambda is especially important because it ensures that all requests are responsive at all times in the database.

Component Library; Things like buttons and chat bubbles and search bar, are common things, so instead of remaking the wheel every time, there are component libraries with already styled buttons, that we can customize the colors without too much effort, making development easier. Efficiency is also maintained because these components are made with the focus of reusability and consistency. These components can be easily updated by updating parameters for each component such as color, size, etc.

5 SPECIFICATION

User Sign-up: Here is the order of actions that need to happen in the user signup. First, they are going to enter their email and create a password, after that data is saved to the database, and an authenticated session is created for the user, the user can then enter their name and their username that they would like. After they submit this information, the data is sent to save to the database, and in the current session that the user is in. They are then redirected to the timeline page, where they can see all messages, and send new messages.

Sign-in: This is a little simpler, after the user enters their email and password and that is verified by our User Authentication system from Clerk Dev, an authenticated session is created and the user is redirected to the timeline page.

Timeline page: This is a little more complicated than the previous two. Firstly, this page is protected by middleware, which does not allow anyone who is not authenticated on it. After this middleware is passed, then a SELECT * FROM MESSAGES, SQL call is made to the database to get all the messages, for the initial load. This will then load and show all the tweets to the user. We keep track of actions that the user does, such as reload to send a new message, or delete a message. If any of these actions happen, the database is updated accordingly, and new messages are fetched. But, when new messages are fetched, it doesn't redisplay or re-render messages that are already on the user's screen. We only render the new messages. This allows for a smooth and fast interaction that the user gets when using the web app.

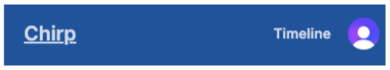
A general feature that we have implemented on the site is caching. So, let's say you load up the main homepage and then go to the timeline. The main homepage is cached on your computer, so when you go back to it, it doesn't need to make a fetch to the hosting site and render again. It will quickly open back up because the rendered page is already saved in the browser. This makes the user experience feel much better since they don't need to wait for the same page to load over and over again.

6 PLATFORM

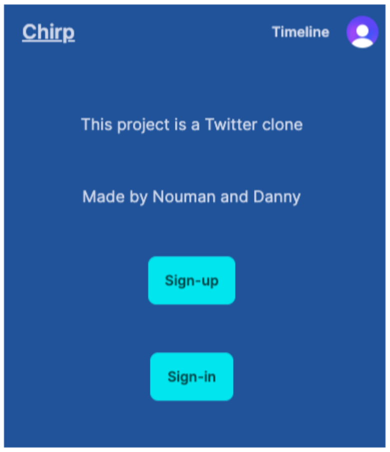
The base of the platform is built on the Next.js framework and Node.js. Next.js has a ton of great features. They support routing patterns based on the filesystem, which means the file ./app/timeline.js will render when the user visits https://chirp-xi-nine.vercel.app/timeline. Also, it supports both client-side and server-side rendering, which makes our lives easier. Node.js allows us to include all the different free developer tools that we used to make the app. Things like Supabase, Clerk, TailwindCSS, and even Next.js were all imported using npm. This makes setting up the development environment super easy and simple.

7 DESIGN

To focus more on the backend, we aimed for a simple design for the website. First we have our Navbar.



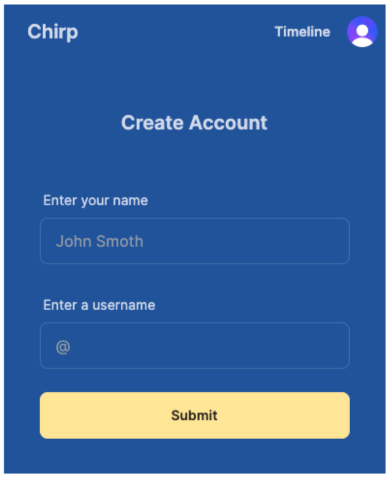
Both words are buttons. Which will route you to where you would like to go. And there is an underline to let the user know where they are in the site.



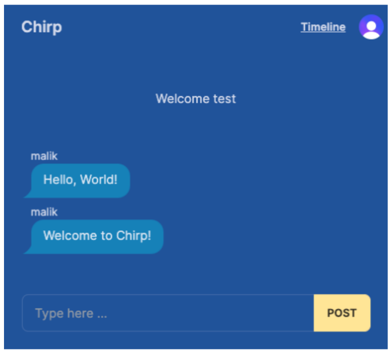
This project is a Twitter clone

Made by Nouman and Danny

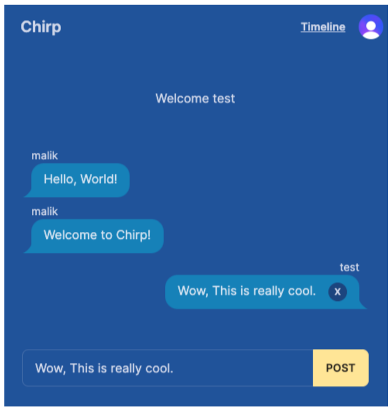
The main page just has our names are 2 action buttons. Sign-up or Sign-in. After you sign up, we have the create account page



This page is quite simple which is good. There is only a single form on this page. We make the button the same length as the input because that then makes the symmetry of the page very good. Inside the text inputs, we have placeholders, to make the user experience better so then can quickly understand what to enter into the form Then here is the timeline page



Because we would like to keep the app like a conversation, each message is like a chat bubble that everyone is used to from their phones. We welcome the user at the top and display each message with the person who sent it. Then when you look at the bottom form, we have a Type here … placeholder that tells the user what to do without needing a label on the input that would look out of place. And we combined the search input with the button to make it look like 2 pieces that are combined into a single action. When a user sends a post like this,



Any message that the user sends, will be on the right, which is just like all other chat apps that people are familiar with. The user gets an x button to delete the message that they sent. We put the x inside the chat bubble itself, so then the user can delete their message with the most ease. In other apps, you usually have to click to open a menu on the message then choose to delete, which adds extra complication that we don't need in the design. This design makes it easy for users to find what they need to do and how to do it. With a simple design we keep the screen free of unnecessary clutter and the functionality of the site can be figured out by the user intuitively.

8 EXPERIMENTAL SETUP

For this project, there was no experimental setup. But when writing code, and developing the site, we had to implement constant user tests. Every little bit when part of a feature is complete, we would put a print statement, to watch if the event is being triggered and if the right information is being printed out at the right time. Then when full features were complete. Such as being able to send messages and delete. We tried as many different test cases as we could to see if the feature would work properly the entire time. And this way of testing made our iterative development process go like this. Code -> Test -> Some more code -> Test -> Complete Feature -> Test -> Save to GitHub.

9 IMPLEMENTATION DETAILS

To authenticate users. Clerk dev has an API and their own premade components. We first saved the proper environment values, then set up the middleware for the site. To set up the middleware, we were just protecting the /timeline page from user that were not yet authenticated. Then, we just imported the React Components from Clerk, and put them inside a button for SIgn-up and Sign-in.

After this, Clerk has a React Hook that allows us to get the current session information. So, from the secured session information, we create a Supabase Client that allows us to access the database. The reason we need the session information is because then, anyone who is not authenticated, cannot see what we have in the database.

After the Supabase client has been created, we create an action that runs only once on load of the timeline page, which loads all the chats. This is through a function call in the Supabase JavaScript Client.

10 DISCUSSION

As well as the project, we did encounter many struggles in creating Chirp. This was our first time creating anything like this so everything was so new to us and having no familiarity was a struggle. The struggles first started when figuring out what backend and framework to use to create this project. We did try multiple different ones but we had issues with user authentication. We then found no issues with that when using Supabase and Next.js so we decided to keep moving forward with those as our backend and framework for Chirp. We also chose Tailwind CSS to style it. We then chose Daisy UI as the component library. Daisy UI has a very user-friendly interface with alot of implementations we wanted to incorporate. We did go through component libraries and css styling options but ended up with these ones. We did alot of research but didn't find what we wanted until we got to Daisy UI and Tailwind CSS. We also did have some issues incorporating some components to our web application. Some issues include deleting messages from the timeline and having messages remain on the timeline but we ended up fixing those over time and ended up getting it all working smoothly. Our strengths in development were definitely user authentication and styling. User authentication when we decided on all components of the web app went pretty smooth and we feel we employed that very well. Also the styling of the web app we feel looks nice and the site looks like all components complement each other. Even though we like the styling it could be improved as always with everything everything can. We feel that everything can be improved greatly over time. We feel like Chirp has a very bright future and that everything will be improved the more we work on it. With all our future advancements, we have plans for this that will take Chirp to the next level and above that.

11 CONCLUSION AND FUTURE WORK

Chirp has many plans for the future and here is how we will improve the overall user experience. First, to improve the overall feel of the app. Possibly incorporating a logo in future advancements that will be attached to our app in the memories of our users. As well as improving the timeline's look and also creating multiple timelines for different categories. Incorporating direct messages and group chats is very important and we look to incorporate that in the future to add another dynamic to our web application. This would incorporate a way to network essentially. This would be a great way to bring more user's to the platform as it could offer that benefit. Also, intertwined with the ability to message privately we want to incorporate a user search. This user search could be based on the username and first and last name that is on the account. Lastly, a subscription service that can be purchased by users to subscribe to certain users for content they put out or to be on their thread. Content could include live streaming, posted videos and special offers the user decides to provide for subscribers only. In conclusion to Chirp and the developments made so far we think that the potential is limitless. With so many great ideas for the future and a great team behind it we don't believe there is anything in the way. We also believe that if we continue to build on what we started this app can really threaten its competitors one day.

A APPENDIX

A.1 GitHub Repository

GitHub: https://github.com/NoumanAMalik/Chirp

A.2 Presentation

GitHub: https://github.com/NoumanAMalik/Chirp-Presentation

A.3 Chirp Live Site

Chirp: https://chirp-xi-nine.vercel.app/