

Jacob Tucker

Website: jacob-tucker.github.io | GitHub: github.com/jacob-tucker | Email: jacobtucker818@gmail.com | Phone: 914-217-8139

EDUCATION

Northwestern University

B.A Computer Science

GPA: 3.96/4.00

Expected Graduation: 2023

- Selected Courses: Web Development, Algorithms

EXPERIENCE

Undergraduate Teaching Assistant

July-August 2020

Northwestern CS211

- Helped instruct a Computer Science course (CS211) in C/C++ and oversaw the completion of multiple mini-game final projects.

PROJECTS

Search Algorithm Tool (searchalgorithmtool.netlify.app)

July 2020

Personal Project

- Implemented a visual tool for learning graph search algorithms using React.js hooks that handled the state of the walls, the nodes currently being searched, the discovered nodes, and the start & end points simultaneously.
- Wrote 5 graph traversal algorithms, including A*, Dijkstra (w and w/o a Binary Heap), Greedy Best-First Search, and Depth-First Search using priority queues and other data structures.
- Programmed a maze generator using recursive division along with a random maze generator.

EasyTalkv2 (easytalkv2.netlify.app)

May-June 2020

Personal Project

- Built an instant messaging chat application that allows users to send messages back and forth simultaneously through WebSockets (using Socket.io library), via a Node.js server using the Express library, hosted on Heroku.
- Enabled the ability to continue conversations at any time by retrieving user profiles and chat conversations stored on a NoSQL MongoDB Atlas cluster, using AWS as the cloud provider.
- Designed user personalization by giving users the ability to change the styling of their chat.
- Utilized the Axios library to make GET/POST requests from React front end to Node server.
- Implemented JSON authentication tokens stored through browser cookies that prevent 3rd party data interference, using a built Restful API.

MusicLife (musiclife.netlify.app)

June-July 2020

Personal Project

- Created a social media application centered around users' love for music. Users can add songs to a general database, add songs to their personal library, create playlists, join/create communities, chat in these communities with people who share similar interests, and add songs to them, which are all stored in each user's account using a Firestore database.
- Utilised Firebase authentication to allow users to login, signup, and logout whenever they want, while preventing outside interference with the database.
- Used React-Redux to handle client-database communications and the overall state of the application.

SKILLS

Front-end: React.js, JavaScript, HTML5, CSS3, Sass, Axios

Back-end: Node.js, MongoDB, REST APIs, JSON, Python, WebSockets

Tools: AWS EC2, Docker, Git, Heroku, Netlify