

# JACKSON NEWMAN

Redwood City, CA, 94065 | 650-649-8204 | [jnewman167@gmail.com](mailto:jnewman167@gmail.com) | [linkedin.com/in/jacknewman](https://www.linkedin.com/in/jacknewman) |  
<https://github.com/JNewman-cell/> | [jacksonnewman.netlify.app/](https://jacksonnewman.netlify.app/)

## WORK EXPERIENCE

### AMD

San Jose, CA

Software Engineer Intern, Internal Development Tooling

June 2023 - September 2023

- Accelerated Vivado constraint processing by 50% with a new C++ pattern matching function.
- Developed unit tests achieving 100% coverage to ensure performance and accuracy for Wildcard Matching.
- Reduced Vivado memory usage by 2% by refactoring code to utilize Tessil C++ Hash map package.
- Automated memory, encryption tests, and key upgrades using Python, cutting testing time by 50%.

### Shellie.us

San Francisco, CA

Full Stack Software Engineer Intern, MERN Stack

June 2022 - September 2022

- Developed a React-based hierarchical UI with dynamic modals and edit functionality for exhibits.
- Created REST API integrations for editing and saving exhibit data in the NoSQL backend database.
- Created reusable React components, improving maintainability and reducing development time by 30%.
- Used React and Redux for state management, enhancing data consistency and reducing errors by 20%.

## PROJECTS

### AI Model Demonstration Website | *Flask, AWS, GCP, LLM, REST API*

- Configured and deployed 3 AWS EC2 Linux VMs for AI model performance testing and benchmarking.
- Benchmarked Llama.cpp and vLLM server with LLMPerf to determine the fastest LLM inference on CPU.
- Improved Stable Diffusion performance with NNCF quantization, cutting image generation time by 80%.
- Deployed an end-to-end Flask web app with Nginx to showcase OpenAI-like LLM inference via HTTP.
- Developed a Flask API server capable of handling hundreds of image generation requests.

### Historical Stock Information Visualization Website | *Flask, Javascript, SQLite, Git, CI/CD, GitHub Actions, REST API*

- Developed a custom Trie autocomplete, indexing over 6000 stocks by market cap, allowing for ticker search.
- Automated unit testing for the Trie autocomplete, achieving 100% coverage of all possible tickers.
- Improved stock info retrieval time by 90% through optimized API design and database storage.
- Created custom GitHub Actions to automate 100% of database management for 3 SQLite databases.
- Developed a custom caching system that cut GitHub Actions workflow time by up to 60%.

### REST API implementation and Database Management | *HTTP, PostgreSQL, Java, Springboot, Swagger, React*

- Developed CRUD operations in a Java backend for PostgreSQL database, deploying 6 key functions.
- Created a comprehensive testing suite to cover 100% of query functionality, including error handling.
- Integrated frontend and backend testing into the CI/CD workflow, ensuring 100% up time in production.

### Website for Daily Commute Information | *React, Javascript, CSS, Bootstrap, User Auth, GCP, Firebase, Git*

- Designed and developed a user-friendly commute reminder website, providing 100% current traffic info.
- Configured Firebase for secure user authentication and data storage, optimized for over 1000 users.
- Utilized Google Cloud Functions for serverless computing, sending emails with optimal routes daily.

## EDUCATION

### University of California Santa Barbara

Santa Barbara, CA

Bachelor of Science in Computer Engineering

September 2020 - June 2024

### 3.7 GPA, Dean's Honors, Relevant Coursework:

Data Structures, Algorithms, Operating Systems, Machine Learning, AI, Embedded Systems, Web Development

## TECHNICAL SKILLS

**Programming Languages:** C/C++, Java, Python, PostgreSQL, SQL, SQLite, JavaScript, HTML, CSS, JSON, YAML

**Frameworks:** React, Node.js, Flask, Bootstrap, Material-UI, Storybook, Spring Boot

**Developer Tools:** Git, Docker, Jira, Google Cloud Platform, Azure, AWS, GitHub, Confluence, Perforce, NPM, Firebase, GitHub Actions, MongoDB, Jenkins, Kubernetes

**Libraries:** pandas, NumPy, TensorFlow, PyTorch, Keras, Seaborn, matplotlib