Jay Mistry

jay@jellyware.co.uk | www.jellyware.co.uk
www.github.com/jaym-01 | www.linkedin.com/in/jay-mistry1

EDUCATION

Imperial College London - Electronic and Information Engineering **Graduation Date: June 2026**

- Obtained **Dean's List** in my 1st, 2nd & 3rd year for academic performance (**top 10%** of the year).
- Built a pipelined RISC-V 32-I (CPU): System Verilog & C++.
- Built a C90 to RISC-V compiler with C++, Flex and Bison.
- Built a hardware accelerator for a softcore CPU on an FPGA, providing 200x performance improvement to a specific computation (from 16s to 8ms)

Notable Modules

- **Advanced Computer Architecture** – 89%
- **Machine Learning** – 86%
- **Digital System Design (FPGA)** - 86%
- **Compilers** – 81%
- **Deep Learning** – 77%

PROFESSIONAL EXPERIENCE

Meta - Software Engineering Intern (Facebook Messenger Web) **June 2025 – September 2025**

- Designed, built, tested and began shipping new features for Messenger to the **1 billion** Facebook users worldwide; experiments with these features demonstrated a **2% increase in message sends**.
- Took care in providing good user experiences and performance, considering issues such as flickers in the UI and JavaScript bundle size. Designed and built for both the frontend and backend, using **React** & **JavaScript**.
- Proactively identified and fixed production bugs: fixed race conditions and incorrect function calls for creating new chats, improving the reliability and future maintainability of the feature.
- Architected and built a new set of APIs for an existing client-side persistent database store, offering a **10% reduction in latency** for reads from and writes to the store.
- Led cross-functional collaboration between architecture, QA and privacy teams to navigate the complex compliance requirements and to begin feature roll-out to users.
- Created unit and end-to-end tests for critical Messenger features, such as chat previews, which were used by several engineers introducing changes that could break these features.

OncoFlow (AI Health tech startup) – Full-stack engineer **June 2024 – September 2024**

- Joined as the first **Full-Stack Engineer**, leading the development of the company's main product.
- Built a dashboard platform enabling NHS workers to log in, interface with and run AI models, connect to medical databases with cancer treatment information, and generate and edit AI-produced medical reports.
- Designed and built the web app and backend (REST API Server running AI models) from scratch independently, using Next.js (with **React, Typescript & Tailwind**) & **Python** FastAPI.
- Developed a demo environment to simulate the data environment used in the NHS, for testing before integration.
- Added optimisations such as caching to reduce the use of AI models, reducing cost and latency.
- Implemented unit, integration and end-to-end tests using pytest, Jest and Playwright.
- Created local testing environments to simulate production, using Docker and Bash scripts.
- Worked in a fast-paced environment, adding features and fixing bugs within a few hours or days.

SKILLS

Programming Languages/Frameworks: (Proficient): **Python**, **C++, TypeScript, JavaScript, Node.js, React.js**; (Intermediate): **C, SQL, Jest, Rust, Verilog.**

Proficient with **Git, GitHub & UI design with Figma.** Familiar with **Unix terminal, Bash Scripting, Docker & AWS**.

PROJECTS

(IC HACK Winner) VisuMath (Worked on Backend - **Python**) (https://devpost.com/software/intellilearn-kjxv19):

- A gallery for educational math videos, search for videos with a prompt and generate new videos with an AI Agent.
- Designed the overall architecture for a robust, reliable and responsive backend – to ensure the server could handle requests from multiple users as it was performing intensive operations (e.g. running the agentic pipeline).
- The backend was made of 3 layers: an API server (using Python Fast API), a Redis queue and a Celery Worker (a Python process handling long-running tasks, queued in the Redis queue).
- Videos are stored in AWS S3 buckets, their metadata is stored in AWS Dynamo DB and video titles were stored in a Vector Database, to enable searching for videos that closely matched the user's prompt.

Snipr (Tauri desktop app, frontend – **React** & **Typescript**, backend – **Rust**) (https://github.com/jaym-01/snipr):

- Built a production-ready desktop app that removes silences and provides transcripts for audio files, with 1 user.
- Implemented a CI-CD pipeline using GitHub Actions to build the app and provide automatic updates for users.
- Implemented decoding and encoding audio files to PCM samples with FFMPEG, an algorithm to remove silences, and provided a responsive UI by spawning new threads to process the audio file, all in Rust.
- Developed a Rust-Python integration using PyO3 to call OpenAI's Whisper model to provide audio transcriptions.