Cameron Jones

cjones3@tcd.ie | +353 83 327 9442 | LinkedIn: cameron-jones-545652350 | GitHub: CameronJ-06

EDUCATION

Trinity College Dublin **Dublin, Ireland**
B.A. (Mod.) in Integrated Computer Science *Expected Graduation, May 2029*

- **Current Grade:** 2:1
- **Related Coursework:** Data Structures & Algorithms, Object-Oriented Programming (Java), Introduction to Computing (ARM Assembly), Systems Programming (C), Concurrent Systems & OS, Data Modelling (UML, XML), Computer Architecture (VHDL), Microprocessors (Embedded C), Discrete Maths, Probability & Stats (R)

PROJECTS

Fitter – Running Fitness Analytics with Machine Learning | cameronjones.eu/fitter **Solo Project**
Solo Developer *Oct 2026 – Present*

- Built Fitter, a Java web app (Spring Boot, PostgreSQL) that tells runners if they are getting fitter, ingesting 1 GB Apple Health XML exports with a StAX streaming parser and async jobs, never loading whole files into memory
- Trained a gradient-boosted model in Java with the Smile ML library on 28M ten-second segments from ~97,000 real runs, predicting heart rate with 11% lower MAE than a standard linear baseline model
- Found and fixed a Simpson's paradox that made heart rate look 31% less sensitive to pace than it really was
- Built a per-user ridge-regression layer that cut heart rate MAE by 39% (12.5 to 7.7 bpm), predicting each run only from that runner's earlier runs, across 82 runners held out from all model training
- Trained a second Smile model on 9,828 hard efforts to predict race times with a median error of 3.2% (5 km) and 3.4% (10 km), then used the Daniels–Gilbert formula to build a pace planner for any distance
- Generated coaching reports with Llama 3.2 self-hosted on Ollama, validating 100% of its numbers in Java before display and cutting its prompt by 59% (2,260 to 930 tokens) to run fast on a CPU-only server
- Kept all health data on my own server, running Spring Boot, PostgreSQL (Flyway migrations) and Ollama in Docker Compose with 0 data sent to outside services, backed by 227 JUnit 5 tests covering every module

Europe 2100 – European Demographics Web App | europe2100.com **Group Project**
Frontend Lead, 2-Person Team *Apr 2026 – Present*

- Co-building Europe 2100, a full-stack platform (React, Python, FastAPI, pandas) that turns 18,000+ Eurostat and World Bank data points into an interactive picture of 65 years of European population change
- Lead the 7-page React frontend, building an interactive map of Europe with D3 and TopoJSON where users scrub through 65 years of data and compare up to 8 countries, with every view saved in a shareable URL
- Made the site feel instant by caching the full dataset in a shared store built on React's useSyncExternalStore and lazy-loading pages with Vite code splitting, so switching indicators needs zero API calls
- Building a Python scenario model that projects population and ageing 75+ years ahead to 2100, measuring its error and uncertainty and letting users test "what if?" assumptions such as fertility at 1.8

Flight Data Visualiser – Java Desktop App **University Group Project**
Team Leader & Developer, 4-Person Team *Feb 2026 – Apr 2026*

- Led a 4-person team to build a Java desktop app in Processing that loads 563,737 real U.S. flights (10 airlines, 369 airports) into interactive charts of delays, diversions, busy routes and hourly traffic
- Implemented all 4 chart types (histogram, bar, scatter and pie) from scratch with Processing's graphics API, organising flight data with Java HashMaps and writing the largest share of the team's code
- Designed a search screen with 11 combinable filters and a dual-handle time slider, filtering 560,000+ flights live with Java collections and showing a live match count before any chart is drawn on screen
- Applied object-oriented design to split the app into 20 classes (screens, charts, widgets) and ran 5 weekly milestones with Git branches and pull requests, so all 4 of us could build in parallel

COURSES & CERTIFICATES

Kaggle: Intro to Machine Learning – decision trees, random forests, model validation and overfitting *Nov 2026*
Claude Academy: Building with the Claude API – prompt evaluation, tool use, RAG and MCP *Nov 2026*

SKILLS

Languages: Java, C, Python, JavaScript, SQL, C++, HTML/CSS, VHDL, ARM Assembly, R, XML/XQuery
Frameworks & Libraries: Spring Boot, React, D3.js, JUnit 5, Smile (ML), Thymeleaf, Chart.js, Processing
Tools & Platforms: Git & GitHub, Linux, Docker Compose, Nginx, PostgreSQL, Ollama, Vite, Vivado, GCC, GDB