


James Cantamantu-Koomson Jnr

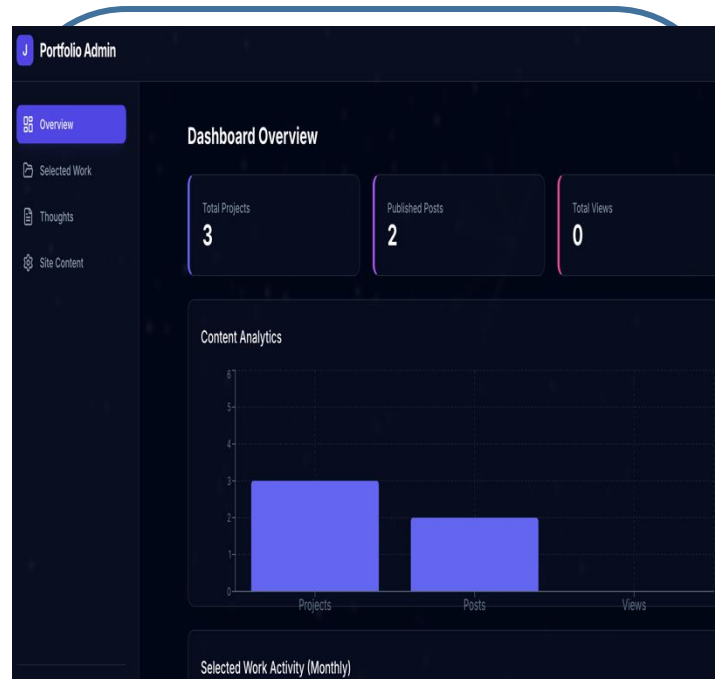
Web Tech Summer 2025 Final Exams





James is developing a data-powered website to help himself manage his portfolio, publish insights, and experiment with new skills as he grows professionally.

Project Title



PERSONA(s): 1. James Uses the site as a personal workspace to track work, publish thoughts, and experiment with real-world systems
2. Peers, recruiters, and collaborators browsing the portfolio to understand James's work, interests, and thinking. They consume content but do not manage it

PROBLEM: Before this website, projects, ideas, and learnings were scattered across files, notes, and code, making it hard to track progress or present growth clearly. The website solves this by creating one living, data-powered space where work, thoughts, and progress can evolve naturally over time

PAYOFF: *What is a simple drawn screen that is illustrative of something the website shows that helps solve the problem. It is likely a report or some results of data entered by the participants. **This is the most important square***

FINISH: . James can update, reflect, and grow without friction, while visitors get a clearer, more authentic view of his work and thinking.

User / Customer

- This website is a personal portfolio and learning workspace used personally by James Cantamantu-Koomson Jnr. Its purpose is to document projects, publish articles, track growth over time, and experiment with real-world product and technical skills in one living, data-driven space.
- **User roles**
 - Admin (Owner):** Manages projects, writes articles, uploads images, and maintains all site content.
 - Public Visitors:** Browse the portfolio, read articles, and engage with content through comments whilst having the opportunity to reach out to James and also download his CV.

Main Functions of the Website

- **Admin content management:** Allows me to add, edit, and update projects, articles, and personal information without touching the code.
- **Article publishing and reading:** Lets me write full articles that can be read on dedicated pages, making it easier to document what I'm learning.
- **Image upload and media handling:** Enables direct image uploads for projects instead of relying on external URLs.
- **View tracking and analytics:** Tracks portfolio views and monthly activity so I can see engagement and growth over time.

Architecture

1. Presentation Tier (Frontend)

- **What it does:** Displays the portfolio and admin dashboard, handles user interaction.
- **Programming languages:** TypeScript + React
- **Why:** TypeScript provides type safety and easier navigation, React allows component-based, dynamic UI updates.
- **Other technologies:** Tailwind CSS for styling, custom UI components for forms, buttons, and cards.
- **Hosting:** Vercel, for easy deployment and fast delivery.

2. Application Tier (Backend / Logic)

- **What it does:** Handles authentication, content management logic, data fetching, and validation.
- **Programming languages:** TypeScript (server-side via Supabase client)
- **Why:** Keeping the same language across frontend and backend simplifies reasoning and reduces context switching.
- **Other technologies:** Supabase functions and server-side actions handle writes and RLS enforcement.

3. Data Tier (Database / Storage)

- **What it does:** Stores projects, articles, comments, site content, and images.
- **Programming languages / tech:** PostgreSQL (via Supabase), Supabase Storage for images
- **Why:** PostgreSQL is reliable, structured, and well-supported; Supabase adds managed auth, storage, and RLS.
- **Other technologies:** Supabase handles authentication and storage APIs, Row-Level Security for safe access control.

How They Connect

- Frontend sends requests via Supabase client to the database and storage.
- Application logic validates inputs, enforces RLS, and manages content.
- Database responds with structured data, which the frontend renders dynamically.
- All tiers work together to allow secure content updates, real-time views, and analytics without a separate backend server.