Written interview

September 13, 2025

1 Web Engineering Experience

1.1 Skills Acquired in the Past Year

What skill or knowledge have you acquired in the past year that has been particularly helpful? What motivated you to learn it? What has the impact been for you and your team?

Given the rise of AI, The past year has been a roller coaster for everyone in the software domain. I studied LLM's and trained a small LLM model on stock charts (image data). I understood how to train effectively and what all limitations arise by having incorrect or unbalanced training data. I also understood that it is no magic bullet and basically was giving me an output of an statistical average of all the training data which matches the input pattern.

The effect of this learning was that I got a head start into AI in the organisation and was able to spearhead Bankbazaar's AI journey. I implemented several small AI tools which boosted both developer productivity and business ops productivity.

1.2 Web Programming Experience

Describe your experience of web programming - JavaScript, Typescript, React, CSS and Python in particular.

I have been part of the evolution of web programming from the early days of REST APIs, then continuing my journey through both ReactJS and AngularJS. On the CSS front, I have worked with plain CSS, SASS till tailwindcss. The below section has some of the notes and recollections on the same in chronological order.

- **JSF @ IBM (2008-10)** Starting as a fresh graduate, I learnt core java and JSF at IBM. The product was an internal tool hence the styling was given in the form of pre defined JSF components, I was able to do was design the pages using those components and implement the backend.

  I would not comment on the experience of using JSF as that was the technology available at the time and I had no real baseline to compare against.

- **Early react and grommet (javascript framework) @ HPE (2010-16)**

  This is where I had real experience with JavaScript. I lead a team of 4 developers (two for backend and two for frontend) to move the entire application from Flash to HTML/JS along with writing new REST api's.

  The CSS was again given in SASS defined company wide (grommet framework) and we had the liberty to modify the it slightly to fix issues found during testing.

  Having also learnt Adobe Flash for migration, I was glad that the application moved to JavaScript. The migration was almost seamless because I was able to make decisions on both the REST api's as per the needs of the screens and also on how the layout of the application would be knowing the restrictions from backend. Learning JavaScript, ReactJS and SASS was a good challenge and also I had to make hiring decisions on the team and train them on SASS and ReactJS. The project shipped on time with 0 critical bugs and about 30 medium ones (which needed a framework fix).

- **AngularJS @ Digilend (2016-17)** It was a team of 3 individuals writing a fully paperless loan origination system. The work involved picking up AngularJS skills, We had complete control on all aspects of the website.

  Since this was a early stage startup the traffic on the site was relatively low but we had to extensively optimise for Indian internet conditions. We encountered performance bottlenecks because the bundle size was large and the customers were on low data connectivity. I did make few optimisations which brought

page loads on 3G to less than 30s. Other issues were with consistent styling which I took as a challenge and moved to SCSS. The overall time to deliver features was less compared to plain JavaScript frameworks.

- **ReactJS, TS, NodeJs, Java and Python @ Bankbazaar.com** This being a large website with 1 Million DAU, Most parts of the customer facing website uses ReactJS with TS. There are 100+ services with many internal application written using combination of ReactJS, Java, Python and NodeJs.

  My experience is with designing and implementing few internal applications for the SRE and InfoSec teams using Python for REST and ReactJS for UI. I prefer tailwind for CSS (as it is internal application) and Typescript for its type checking. I have gotten consistent results with the choices along with Selenium for testing. I also have experience of debugging the core website from SRE perspective caching issues, chrome bugs, performance drops, mobile device issues with audio / video etc.

1.3 Building Large Systems

Describe your experience building large systems with many services - web front ends, REST APIs, data stores, event processing and other kinds of integration between components. What are the key things to think about in regard to architecture, maintainability, and reliability in these large systems?

My thoughts based on experience at Bankbazaar:

Bankbazaar is a complex system running on Kubernetes (150 nodes) with 100+ microservices written in Java, JavaScript, ReactJS, Python and more. The platform integrates multiple data stores and event systems including Redis, MongoDB, MariaDB, Kafka, and Cassandra, along with a separate data lake warehouse built on Redshift and Power BI. I am responsible for the platform side of all of these in terms of configurations, architectural design decisions, and ensuring scalability, security, and developer productivity.

- Defined microservice architecture standards and deployment patterns to simplify onboarding of new services (cut from 2 days to under 1 hour). This required thinking ahead about interoperability, API contracts, and developer self-service.
- Standardized observability across all services — enforcing logging, alerting, and metrics contracts. This eliminated blind spots and made reliability measurable, not just assumed.
- Built incident response automation and runbooks for AWS and Kubernetes. The key here was designing for predictable failure modes and reducing P1 resolution times to under 1 hour, with automation cutting frequent issue responses to 5 minutes.
- Migrated core services from data center to AWS. The focus was on phased cutover, disaster recovery readiness (cluster rebuild <1hr), and latency reduction (>50%). Each decision balanced reliability, performance, and cost.
- Spearheaded FinOps practices across 20+ AWS accounts. Reliability is not only technical — cost visibility, tagging, and dashboards ensured the platform stayed sustainable while scaling.
- Drove developer-first security automation: traffic management, vulnerability scans, and automatic CVE resolution. This reinforced the idea that quality and security cannot be bolted on — they must be built into workflows.

2 Software Engineering Experience

2.1 Previous Software Projects

What kinds of software projects have you worked on before? Which operating systems, development environments, languages, databases and frameworks?

I have written about my previous software projects under section 1.2. Please refer to the table below for further details.

Company	Kind	OS	Dev environment	languages	database	framework
IBM	Web tool that runs on IBM servers	Linux	Eclipse, Local webspehere	Java, JSF	Custom (In memory)	Websphere, OSGi
HPE	Installable application, opens in a web browser	Windows	Eclipse	Java, JavaScript	Mysql	ReactJS, Hibernate, Spring
Digilend.in	Website	Linux	Eclipse, Nginx	Java, JavaScript	Mysql	AngularJS, Hibernate, Spring boot, Drools
Bankbazaar	Website	Linux	IntelliJ, Docker, VSCode	Java, JavaScript, Python	MariaDB, Cassandra, MongoDB, Elastic-Search, Redis	ReactJS, Spring boot, Spring JPA and many more.

2.2 Open Source Contributions

Outline your thoughts on open-source software development. Have you been an open-source maintainer, and can you describe the scope of your contributions to those projects?

I admit that there is less to write in this section. My contributions to opensource was a very small enhancement to Eclipse JGIT project. I implemented AES encryption support for it, which was required by an internal tool in my company. Apart from that, I have used opensource software for almost all application on our stack at Bankbazaar. Ocassionally, these products misbehave and we either open bug reports or follow up on existing ones for workarounds and fixes. Especially around kubernetes, istio.

2.3 Proudest Success

What is your proudest success as an engineer, or leader?

In my career, I consider all the small and big achievements to be proud of equally. I have enjoyed, learnt and grown with all of them. There are a few which have left a mark on others, one of them is as follows:

- **Situation:** The kubernetes cluster in Datacenter crashed, taking down critical services for running the business.
- **Task:** I had to recover the cluster and restore services quickly for minimal business loss.
- **Action:** I initially tried to recover the cluster and failed to do so. After one hour of trying, I proposed to the management team that it is a better idea to move the services to cloud. It was in the plan but had to be expedited. The decision was taken to go ahead with the plan and I brought in all the teams needed together and we finished migration to cloud within the next two hours and the business continued to function from cloud.
- **Result:** Minimal loss to business despite of doing an sudden migration. The management team still talk to this day about how the company leaped into cloud on that fateful day.

2.4 Software Quality

Outline your thoughts on quality in software development. What practices are most effective in software teams to drive improvements in quality?

My thoughts on software development in the order of importance:

- The architect or team leader plays a critical role by keeping a larger frame of reference, anticipating integration issues, and guiding the team towards sustainable design choices.
- Balancing speed and quality is essential. Short-term delivery pressures should not compromise long-term maintainability, and this balance requires clear decision-making frameworks.

- Consistent practices such as code reviews, automated testing, and CI/CD pipelines help to catch issues early and make quality a shared responsibility.
- Encouraging open communication and feedback within the team ensures that problems surface quickly and improvements can be acted on without delay.

3 Education

3.1 High School Performance

How did you fare in high school mathematics, physical sciences and computing? Which were your strengths and which were most enjoyable? How did you rank, competitively, in these subjects?

- **Mathematics:** Top 80% in school. I cannot do quick math like 12 x 14, but I can solve engineering problems (integrate and differentiate etc).
- **Sciences:** Top 95% in school. I had a fascination towards electronics.
- **Computing:** Top 95% in school. The fascination from electronics carried over to computing where I could finally learn and write code myself. I started with C and got hooked into programming. Which eventually led to embedded systems.

3.2 Leadership in High School

In high school, what leadership roles did you take on?

I usually took on the class representative roles. To maintain order in class etc. I do not remember much apart from this.

3.3 Degree and University

Which degree and university did you choose, and why?

Electronics and Communications, Because I was fascinated by the field. Having seen its growth in popularity from B/W television to telephones and modems. Plus, The moment I had my own computer I started tinkering with hardware and software to control. It was a no brainer choice.

3.4 University Achievements

What did you achieve at university that you consider exceptional?

I continued my journey of Electronics and computing towards Learning Robotics, embedded systems. We built a Mission to Mars robot and competed against the nations best engineering colleges. That was quite a exceptional moment, a culmination of all learnings over the years.

4 Context

4.1 Thoughts on Canonical's Mission

Outline your thoughts on the mission of Canonical. What is it about the company's purpose and goals which is most appealing to you? What is risky or unappealing? Are there any elements of the company goals that you are unsure about?

The following are my thoughts using the framework *Any origanisation needs to have a sustainable source of revenue, which it can then choose to use for common good* and my current understanding / experience with Canonical products.

- Mission and appeal: Canonical's purpose of making open source more accessible, enterprise-ready and cost-effective is very relevant. The appeal for me is in how this creates a sustainable revenue stream while lowering barriers for organisations—especially those in regulated industries or cost-conscious markets—to adopt open source at scale. It ensures the company can keep investing in the ecosystem while at the same time providing real value through products like Ubuntu Pro and datacenter solutions.
- Risks and challenges: The challenge I see is in mindset and affordability. Many organisations, particularly in emerging markets, look at offerings such as Ubuntu Pro as an "addon" rather than a necessity. Unless there is a push from internal InfoSec teams or a more suitable business model, adoption will remain limited.

On the datacenter side, there are already many players with similar offerings, all promising to "cloudify" infrastructure. Differentiation and visible traction become key, and without clear proof points it is not always easy to judge the impact.

- Open source support: Commercial support for open source is an important service and critical in regulated industries. However, in cost-sensitive markets companies hesitate to pay for support around software they perceive as "free." This becomes more complex with the rise of AI, which is already reducing the need for traditional support in many cases. The risk here is that only a limited set of organisations will continue to see the value in paying.
- Areas of uncertainty / personal view: From my own experience, I know the value Canonical brings, but I have also actively chosen not to adopt certain solutions because of affordability concerns. For example, we continue to use Ubuntu (not Pro) for most workloads, while relying on Amazon Linux for EKS, simply because it was the more cost-effective option for us. This is where I see the gap: the mission is strong, but for organisations like ours the business model has not yet aligned with the realities on the ground.

4.2 Changes You Would Suggest

What would you most want to change about Canonical?

In the past: Companies to managing their own infrastructure is the most cost effecient approach. However, doing so requires skilled employees who can dive deep and debug issues which becomes either too costly or too rare resulting in an unreliable infrastructure. Hence cloud migration seemed like a good choice for many organisations.

Proposal: AI's emergence brings a fresh oppurtunity to automate and fill the skills gap needed for each organisation to maintain their own infrastructure. I believe, in the near future many would be looking to decouple from cloud due to cost reasons and will be interested to deploy and run their own infrastructure with the help of AI to maintain uptime. I would like Canonical to develop products or features for the same.

4.3 Excitement About the Role

What gets you most excited about this role?

- The broad scope of Canonical's work, which creates space for me to learn and grow.
- The flexibility of remote work, allowing me to be productive from anywhere.
- The opportunity to have Canonical's brand and recognition as part of my career journey.