Xyan Bhatnagar

Software Engineer

xyan.pro
me@xyan.pro
linkedin.com/in/xyanbhatnagar
github.com/xbhatnag

About Me

- L5 Software Engineer with 4 years of full-time experience at Google.
- Graduated with a Bachelor's degree in Computer Science from the University of Waterloo.
- Deep expertise in C++, Rust, operating system design and developer tooling.
- Previously worked with Android, Java, Python, C++, Rust, Go.
- 5 internships at Google in various teams (Android, Core, Cloud, Fuchsia).

FTE Timeline

Cast - Linux Receivers

Location: San Francisco, California
Role: L5 Senior Software Engineer
Duration: May 2023 - Present
Accomplishments:

- Designed and developed Cast Settings app
  - A self-contained web app that can be deployed to TVs and allows users to accept Terms of Service, opt-in to usage reporting and view the terms and conditions and privacy policy of Cast, all using their remote.
  - Worked with UX and Localization teams to build an intuitive and localized experience for TVs.
- Designed and built a proof-of-concept for Sender-side Terms of Service
  - Android and iOS "sender" devices can be used to accept Cast Terms of Service on first cast attempt to device
  - This design accounts for different form factors (TVs vs speakers) and has graceful fallback mechanisms to Cast Settings app, if needed.
- Designed and developed Cast Extremely Lightweight Launcher (CELL)
  - CastCore used a lot of memory on TVs while idle (RSS > 20MiB). This was a big roadblock to expanding to more OEMs.
  - CELL is a lightweight daemon that advertises Cast capabilities and launches CastCore only when a sender starts a Cast session.
  - We were able to achieve our goal for CELL RSS to be under 10MiB while idle (~5MiB RSS with sample libraries).
- Checked in ~136 CLs to Cast codebase
  - Most of this code was written in C++
  - This includes bug fixes, internal cleanups, new features, documentation and tests.

Fuchsia - Component Framework

Location: Remote, Canada
Role: L5 Senior Software Engineer (promoted in Oct 2022)
Duration: Oct 2021 - May 2023 (1 year, 7 months)
Accomplishments:

- Primary point-of-contact for all component-related tooling.
  - Components are akin to "apps". They are a fundamental building block in Fuchsia.
- Designed and implemented debug APIs in Component Manager to expose information about components and control them.
  - Component Manager is the user-space administrator of components.
  - Replaced and turned down a pseudo file system called the `hub` with similar functionality.
  - The new APIs are more efficient, have stronger type safety and a good compatibility story.
- Designed and implemented structured configuration for components
  - This feature allows components to define a configuration schema in their manifest.
  - Values are set on a per-product basis as part of Fuchsia system image assembly.
  - Component Manager is responsible for parsing the configuration schema and resolving the values.
- Designed and implemented tools to query and manage components on Fuchsia.
  - These tools are the primary way to interact with components on Fuchsia.
  - Fuchsia developers at Google invoke these tools ~500 times a day.
  - Worked with DevRel to publish several guides for using tools.
- Collected user feedback from tools and used it to guide future development.
  - Most features/bug fixes came from Monorail bugs.
  - I also participated in developer walkthroughs and getting started workflows to find friction points with tools.
- Migrated 100+ components from appmgr to Component Manager as part of an org-wide deprecation effort.
  - Appmgr was the first administrator of components on Fuchsia.
  - This effort required jumping into unknown code bases and working with the owners to land migrations.
- Helped improve the lifecycle state management of components
  - Components can transition through lifecycle states in many ways, making state management complex.
- Checked in ~350 CLs to the Fuchsia codebase
  - Most of this code was written in Rust.
  - This includes bug fixes, internal cleanups, migrations, new features, documentation and tests.

Fuchsia - Resilience

Location: Waterloo, Canada
Role: L4 Software Engineer (promoted in Oct 2021)
Duration: Jun 2020 - Oct 2021 (1 year, 4 months)
Accomplishments:

- Implemented a stress test framework for putting load on critical system components like storage, logging, graphics and virtual memory.
  - Client-side stress test framework was written in Rust.
  - Wrote stress tests for each system component and had them reviewed by the respective teams.
  - Built a notification system in Google3 for stress test failures using PLX, F1 and Go that published Monorail bugs to respective teams.
  - Stress tests run for 6 hours, producing large random workloads on specific components.
  - Workload generation was tied to a unique seed, so there was some reproducibility (assuming no race conditions).
  - Stress tests revealed 2-3 critical bugs in each component (crashes, hangs, race conditions, etc.).
- Built a size tracker dashboard for monitoring Fuchsia build sizes.
  - Tracked the size trend data for each component in a Fuchsia system image.
  - This data was collected from size analysis done in each CI run.
  - The dashboard helped ensure that Fuchsia images fit the storage constraints of Nest hardware.
- Checked in ~200 CLs to the Fuchsia codebase.
  - Most of this code was written in Rust.
  - This includes bug fixes, internal cleanups, new features, documentation and tests.

Internship Timeline

Fuchsia - Resilience

Location: Waterloo, Canada
Duration: Sep 2019 - Dec 2019
Accomplishments:

- Added new features to a shell tool `cs` which parsed a pseudo-filesystem called the hub.
- Added new state information to the hub, including access to all the capabilities used or exposed by each component.
- Built the component events system which sends notifications about component lifecycle state to any interested listeners.

Cloud - Remote Build Execution

Location: Waterloo, Canada
Duration: Jan 2019 - Apr 2019

- Remote Build Execution provides distributed execution and caching of cross-platform build and test workloads.
- Improved stability of Windows workers. Worker software was written in Golang.

Android - Jetpack Libraries

Location: Mountain View, US
Duration: May 2018 - Aug 2018

- Helped prototype a Kotlin DSL for Android App UI layouts

Android - Jetpack Libraries

Location: Mountain View, US
Duration: Sep 2017 - Dec 2017

- Worked on the first release of the WorkManager Jetpack Library.

Core - Intern Tools

Location: Mountain View, US
Duration: Jan 2017 - Apr 2017

- Built the Intern Events tool for coordinating events among interns @ Google.

Education

University of Waterloo

Degree: Bachelor of Computer Science
Location: Waterloo, Ontario, Canada
Duration: Sep 2015 - Apr 2020

- Graduated with distinction (Dean's Honor List)
- Took 2 of the "Big 3" courses (Compilers, Real Time Operating Systems) in final term