Duane Ryan Bailey
33 S Letitia St, Apt 304
Philadelphia, Pennsylvania 19106
(413) 884-2314 - Cell
bailey.d.r@gmail.com
duane@curalate.com

Education

University of Massachusetts (2012-2013)
Amherst, Massachusetts
B.S. in Computer Science, not completed

Massachusetts College of Liberal Arts (2011-2012)
North Adams, Massachusetts

The College of William & Mary (2009-2010)
Williamsburg, Virginia

Williams College (Spring 2009, during high school)
Williamstown, Massachusetts

Mount Greylock Regional High School (Graduated 2009)
Williamstown, Massachusetts

Employment

Software Engineer at Curalate (Jan 2014-Present)
Half backend engineer, half full-stack startup engineer.

Software Engineer at Rap Genius (Summer 2013)
Worked on site backend and mobile application.

Software Engineer at Google (Summer 2012)
Worked on F1 team
Chad Whipkey (cwhipkey@google.com)

Contract Programmer, working on the Williams College C3D project (Summer 2011)
Stephen Sheppard (stephen.c.sheppard@williams.edu)

College of William & Mary IT, Network & Systems Assistant (Fall 2009, Fall 2010)

Academics

PLASMA Research Lab at UMass (Fall 2012)
`checkedheap`, a heap which can check for overflows with high likelihood:
`https://github.com/duane/checkedheap`.
`stopgap`, a system for dynamically detecting heap overflows in multithreaded programs and rolling back program state (with Tongping Liu, tonyliu@cs.umass.edu).
Advisor: Emery Berger (emery@cs.umass.edu)
Site: `http://plasma.cs.umass.edu/emery/plasma`

Skills

- Functionally able to use any programming language, but with a background in C, C++, Python, Bash, and Scala.
- Strong interest in algorithms, systems, distributed services, databases, and compilers.
- Proficient with reasoning through SQL- and NoSQL- database behavior, performance, configuration, and data layout.
- Proficient with graphic design, UX, CSS, HTML, and JavaScript.
- Relentless when diagnosing bugs, performance issues, inconsistent data, and eradicating unhealthy patterns.

Curalate: After I decided to take time off from school at the end of 2013, I found a job as a Software Engineer in Philadelphia at a promising startup named "Curalate" (`http://www.curalate.com`). Since starting in January of 2014, I have covered a wide range of responsibilities:

- Implemented and maintained Curalate iOS application when it launched in February of 2014 (sole application developer)
- Writing design and technical specifications, and the implementation of code from these specifications.
- Implemented, debugged, deployed, and maintained distributed services with my colleagues, including:
  - A distributed, fail-safe work queue.
  - A service for managing credentials used on a client's behalf and auditing their use (sole developer).
  - Distributed clusters mining client web sites and image-based social networks.
  - A custom permissions service authorizing users for actions of virtually any granularity.
- In addition to writing new code I maintain our massive codebase every day, fixing arcane bugs, improving test coverage, patching new failures, updating API support, and debugging client problems.
- On top of backend endeavors, I have significantly contributed to virtually every product offered by Curalate, including writing user-facing UX code and copy-editing press releases.
- Finally, on an engineering team with PhDs and ex-Microsoft engineers, I served as an encyclopedia for systems knowledge and ability to rapidly diagnose a problem.

F1: Worked on F1, a RDBMS, at Google over the summer of 2012, where I partially implemented indexing on (restricted) SQL expressions rather than values directly; for example, this allows case-insensitive string indexing, which is quite useful. I worked in a team environment and gained great experience with source code control, unit and integration testing, code reviews, and working with extremely large code bases. See `http://research.google.com/pubs/pub38125.html` for a presentation on F1, and the more recent paper on Spanner (which forms the backend of F1) at `http://research.google.com/archive/spanner.html`.

C3D: In Summer 2011 I worked as a contract programmer for the Center for Creative Community Development (`http://www.c-3-d.org/`) at Williams College. In particular, I built a site to view the economic impact of Arts institutions. I parsed Excel spreadsheets using Apache's excellent `POI` framework and evaluated the macros manually to build a system of expressions used to calculate the desired numbers, storing it in XML. Then a PHP fronted loaded the XML to display it on the website, caching results between requests and recalculating the numbers as necessary.

An modern implementation of TEX: For Spring 2012, I worked on an implementation of TEX in modern C++ as an independent study. This was intended to be a representation of my mastery of proper and efficient memory management, optimization, documentation, testing, and other software engineering skills. However, I have not worked on the project since finishing the independent study, but I want to resurrect this project with techniques I learned at Google and, more recently, research with the PLASMA lab at UMass; the code is still visible at `https://github.com/duane/ctex` but does not reflect what I feel was an immense amount of expertise gained at Google during my internship.

Reading: More than anything, however, I spend my time these days reading code and papers. I've studied the source of `LLVM`, `python`, the `PyPy` project, the `rust` language, `PostgreSQL`, and the `FreeBSD` kernel quite vigorously. In terms of papers, I've been following along with the recent "developments" in deep convoluted neural nets with a mixture of curiousity and scepticism; in my spare time, I've been reading back through the history of the consensus problem, both in terms of distributed systems (Paxos, raft, vector clocks, and the nature of causality) and in terms of shared-memory parallelism (transactions, software and hardware transactional memory). Recently, I've also been following along with efforts to reduce power consumption of processors in an effort to boost overall performance per watt (or, performance and maintenance overhead per cent).

2015: Recently, I've had my hands full with work, so occasionally I've attempted to explore new technologies. Recently I've been enjoying running plan 9 on my raspberry pi, while thinking about provisioning, deployment, distributed data stores, and fossil/venti/9p; I think Bell Labs really wrote the precursor to today's micro service environment. Plan 9 is about 10,000 kLOC of tooling and modern language runtime support away from being an ideal development environment for distributed systems and large data sets. My only recent public code is at `https://github.com/duane/skip-list`, and was an exploration into the performance and safety of a non-trivial datastructure that is at the core to many LSM-based storage systems. I was very satisfied with the result, though it probably took a month to work out the rough edges of the langauge and understand the idiomatic systems style. I enjoy rust *very much* and thoroughly exhausted the code base in an effort to understand the language's and runtime's internals.

Miscellaneous Projects: I have written numerous internal patches to open source projects, from `python` to `MongoDB`; I have proven my ability to immerse myself in foreign code and make sense of the flow of data quickly across a wide variety of languages and programming styles. In addition, I have written countless unix scripts, python modules, and one-off web servers. Finally, I have experience constructing Linux distributions meant for PXE booting and Live-CDs, where the systems are meant to boot from read-only hardware and configure themselves automatically to their environment as best as possible, and in automated provisioning and deployment (think Chef).