Artur Kashperskiy

Software Engineer

925 285 4348
arturk@uw.edu

https://sm5art.github.io
https://github.com/sm5art
https://www.linkedin.com/in/artur-
kashperskiy-9171ab11a/

Seattle/SF Bay Area

UNIVERSITY *of* **WASHINGTON '2021**

EXPERIENCE

Full Stack Engineer Intern @ Heali.ai
June 2019 - October 2019 // Santa Monica, CA
Stack: React Native, AWS, serverless, redis, DynamoDB
• Configured continuous integration and deployment for automating React Native TestFlight app builds/deployment and backend serverless deployment w/ CircleCI saving our team hours deploying new features.
• Collaborated with designers to build reusable, compact UI components in React Native app such as camera barcode scanning using native Swift camera OCR plugins.
• Built APIs with Express.js and serverless to support product features such as bug reporting and food product data endpoints.
• Integrated Phabricator bug triaging service to help simplify developer workflow and maintained documentation on developer operations and style.

Quantitative Finance Research Intern @ Nipun Capital
June 2017 - Sept 2017 // Foster City, CA
Stack: Anaconda Python 3.6, MySQL, GCloud
• Refactored alpha generation codebase from python 2.7 to 3.6 which improved runtime and library coverage issues and created deployment scripts that automated python environment management.
• Web-scraped endpoints with useful time-series financial data using BeautifulSoup. Used Airflow to schedule data fetching and processing pipelines and ran backtests on the data with internal tooling.
• NLP research on sentiment analysis of conference calls for signal researching and implementing word2vec sentiment embedding models and sentiment dictionary count analysis.

Software Engineer Intern @ Minted
June 2016 - August 2016 // San Francisco, CA
Fulfillment Team
Stack: Flask/Python, MySQL, React/Redux, Backbone
• Wrote unit/integration tests improving coverage of marketing insert logic in Minted orders and fixed bugs.
• Built an API endpoint and frontend for an analytics dashboard used as a productivity tracking tool for our stationary design auditing team.

SKILLS

LANGUAGES
Python, Javascript, Java, C/C++, C#, MATLAB

TECHNOLOGIES
React, Redux, Tensorflow, Keras, MySQL, Cassandra, MariaDB, MongoDB, Redis, Memcached, Nginx, Apache, Gatsby (CMS), Spark, Unity, Unreal Engine, React Native, GraphQL, Airflow

TOOLS
Git, Vagrant, Heroku, Netlify, Docker, CircleCI, Trello, Excel, AWS Lambda/DynamoDB/S3

EDUCATION
BS Applied Physics at University of Washington
Seattle, WA December '2021
GPA: 3.3
Activities and Societies: Washington Esports, Husky Snow Club, Game Dev Club
Achievements: Quarterly Dean's List (3 quarters)
Notable Coursework: Algorithms and Data Structures, Vector Calculus, Intro to Complex Analysis, Probability I, Artificial Intelligence, MATLAB for Numerical Analysis

PROJECTS

demix | https://github.com/sm5art/demix https://github.com/sm5art/demix-frontend
A web tool which allowed music producers to upload an audio file and receive vocals and instrumental individually separated from the file. Included building a python flask micro-service using a pretrained tensorflow model from Deezer (spleeter) and a front-end written with React/Gatsby using MongoDB as database. Unit tested authentication and API calls with unittest. This project saved music producers time and effort by giving them a ml-based approach to remix songs instead of manual equalisation. Launched to production using AWS ec2 and docker for deployment for 2 months gaining ~2000 WAU before I had to shutdown due to high AWS costs. Used Google Analytics to track user behaviour and statistics of usage.

genetic pong | https://github.com/sm5art/genetic-pong
Forked a friend's pong game written in the python package pygame from his GitHub and wrote a genetic algorithm from scratch on top in python that used an unsupervised fitness heuristic to optimise for the fittest pong "brain". The training sequence took roughly ~30 minutes or 30 generations to converge and resulted in an unbeatable AI player. Even reducing the AI paddle size it was adaptive and impossible (for me) to beat after convergence.

kernel | https://github.com/sm5art/kernel
I was curious about operating systems and decided to try to write my own. Using resources from operating system forums online, I was able to put together a simple kernel with graphics capabilities (printing text), a cpu clock, global descriptor table, and interrupt handling for x86 processors. Setup linking and compiling for the kernel with a linux gcc environment Makefile. Utilised qemu-i386 emulator to test the kernel.