github.com/gannonprudhomme
linkedin.com/in/gannonprudhomme

Gannon Prudhomme

gannonprudhomme@gmail.com

Education

- **Texas A&M University** College Station, TX
  BS Computer Science; GPA: 3.76/4.0 *Expected: May 2021*
- **Relevant Coursework**: Data Structures & Algorithms, Programming Languages, Discrete Math, Computer Architecture, Parallel Computing, Computer Graphics

Programming Skills

- **Languages**: Python, Swift, Javascript, Java, C++
- **Technologies**: Git, Node.js, iOS, OpenGL, Visual Studio, Eclipse, Docker

Experience

- **Amazon (eero)**
  Software Development Engineer Intern *May 2020 - Aug 2020*
  - Implemented features and bug fixes using Swift in an iOS App with 100,000+ monthly users
  - Migrated build pipeline from Jenkins to GitLab, increasing build reliability, scalability, and future-proofing the CI/CD process.
  - Implemented invalid login alert, saving $100k/year in customer support costs
- **ACE Lab TAMU**
  Lead Developer *Jan - May 2019; Aug 2019 - May 2020*
  - Lead iOS Developer overseeing a team of 4 undergraduate developers and QA testers
  - Implemented continuous heart-rate monitoring and long-term trend graphs to the Apple Watch and iPhone app using Swift and HealthKit
- **Preventice Solutions**
  Software Engineering Intern *May - Aug 2019*
  - Lead creation of unit-testing and code design standards utilized in onboarding of new developers
  - Using C# and ASP.NET, implemented unit and integration tests alongside development tasks

Personal Projects

- **A&M Class Scheduler** *Sept 2019 - Present*
  - Project manager for team of 7 student developers working on an auto course scheduler written in Python, Django, and React
- **Desktop Control Tablet** *Oct 2018 - Feb 2019*
  - Created touchscreen tablet used to control my PC using Node.JS/Javascript with ExpressJS, Electron, and HTML/CSS
  - Decreased latency to under 5ms by replacing REST API calls with web-socket communication
- **Terrain Generator** *Jan 2018 - Mar 2018*
  - Created a random-terrain generator using C++ and OpenGL following the Perlin noise algorithm
  - Optimized to allow the rendering of 12 million vertices while maintaining above 60 fps using modern OpenGL techniques