Kevin R Kelly

(267) 644-9370 | kevink2019@gmail.com | kevink2022.github.io/**blog** | /in/**kevink2019** | /**kevink2022**

Software Engineer who makes complex systems maintainable, robust, and adaptable by thoroughly understanding software decomposition and how organizations communicate.

Experience

FAST Enterprises July 2023 – Present

Software Developer / Implementation Consultant *Sacramento, CA*

- Eliminated false negatives in a fraudulent account search, cutting it from 20% to 0%, by restructuring a complex 450-line **SQL** query into manageable CTEs, leading to a decrease in fraudulent claims.
- Aligned the client's expectations with their actual needs by building an interactive prototype demonstrating the false negatives, which clarified the original specifications and rectified the reporting discrepancy in the fraudulent account search.
- Addressed and solved critical bugs impacting multiple FAST sites by performing root cause analysis and uncovering a missed case. Communicated the bugs and the fix to affected parties via UML diagrams.
- Improved coding standards by integrating external resources with FAST internal practices, creating style guide and best practice documentation for coding and unit testing **Object-Oriented** programs in **VB/C#** and utilizing CTEs and Execution Plans in SQL Server, driving consistent and reliable coding efforts.

RTD Embedded Technologies, Inc. May 2022 – August 2022

Engineering Intern *University Park, PA*

- Diagnosed a build-time bug impacting 40% of our **Linux Kernel Driver** library due to a kernel update, crafting a rapid response plan to promptly deliver fixes to all impacted clients and finalize document-controlled releases within 2 months. Preempted future client impact with a **Python** script to automate and test build our entire driver suite on new kernel versions.
- Maintained and improved Legacy Linux Kernel Drivers by diagnosing and resolving bugs, including identifying a data structure change in a kernel library as the cause of a runtime error through detailed analysis of Linux source code. Implemented updates to ensure compatibility across all supported kernel versions.
- Tested and verified stability for 13 Linux drivers by testing on multiple distributions such as Ubuntu, Red Hat, and OpenSUSE, while recording results through ISO 9001 and AS9100 compliant document control processes.

Education

Pennsylvania State University August 2019 – December 2022

Bachelor of Science in Computer Engineering *University Park, PA*

- GPA: **3.85**/4. Committed to continuous learning through books, videos, articles, and projects.

Projects

Boomic Music | *Swift, SwiftUI, Combine, async/await* January 2024 – Present

- Optimized playlist curation by categorizing songs with specific tags and constructing 'taglists' based on inclusion/exclusion criteria, leading to a substantial reduction in effort required for assembling diverse playlists with overlapping tracks.
- Established a safe experimental workspace by designing a functional data persistence system with a complete change history, allowing rollbacks to any point in time, saving hours in fixing errors and bugged transactions.
- Built in **Swift** with a combination of functional and object-oriented programming principles. Utilized **SwiftUI**'s powerful observation libraries in conjunction with combine queues facilitate the use of immutable data types.
- Comprehensive **unit testing** suite, adhering to test-driven design principles by writing tests before code, with 142 automated tests so far, covering generic collection types, data persistence, change history, and queuing.

Channels | *C, GDB, Valgrind, pthread, semaphores* January 2022 – May 2022

- Implemented a multithread safe communication system with channel-based messaging, balancing non-blocking, droppable operations with guaranteed completion of blocking messages.
- Included a flexible 'select' function that allows threads to submit multiple send/receive operations, ensuring only one operation is executed.
- Leveraged reference counting garbage collection to allow select threads to return promptly upon completing an operation, despite having outstanding queued tasks, optimizing performance and minimizing resource usage.
- Developed in **C**, employing linked lists for queue management and utilizing mutex locks and semaphores for thread safety and resource control. Debugged with GDB and Valgrind.