Statement of Purpose - Computer Science (PhD) - Fall 2026

iamVL.github.io Vaishnavi Lokhande slokhand@buffalo.edu

My interest in research began while working on a healthcare project supported by the Experiential Learning Grant. Through this work, I started thinking about how post-surgery recovery is understood once patients leave the hospital. Doctors rely on diagnostic reports written during clinical visits, but these reports often miss what recovery looks like in daily life, such as ongoing fatigue, headaches, changes in routine, or discomfort that appears gradually over time. Patient-provided logs do not solve this problem either. Daily checklists and short journal entries are subjective, inconsistently filled, and often missing when patients feel the worst. As a result, neither clinical reports nor patient logs alone give enough information to fully understand recovery. Even when both are available, important context is still missing.

This experience helped me realize that the real challenge is not collecting more data, but **understanding context from incomplete and imperfect information**. I became interested in how **AI systems** can reason when different sources describe the same situation in different, and sometimes conflicting ways. This question has guided the systems I built and the research direction in two distinct yet very related domains: Healthcare systems and Software systems.

RecovR: Post-Surgical Recovery Monitoring In Real-World Healthcare Systems

In this direction, I developed RecovR (code) which is a post-surgery recovery platform that allows doctors to monitor patients remotely after discharge. The system combines patient-written recovery journals with physiological data from wearable devices, presenting trends over time to both patients and clinicians.Rather than solving the challenges of recovery monitoring, RecovR made them easier to observe. Patient logs and physiological signals are each incomplete on their own, but when combined they provide a more complete, yet still uncertain, picture of patient recovery. RecovR's advantage was that it helped notify doctors based on patient journals, particularly in situations where clinical diagnostic reports were incomplete or unavailable. This highlighted a key issue: recovery systems must reason using information that is fragmented, noisy, and often missing.

While building and testing RecovR, I observed a mismatch between what clinicians expect recovery data to show and what the system actually captures. Clinicians often look for clear signs of improvement or concern, yet the inputs are inconsistent. Patient entries may be brief or missing, sensor readings fluctuate, and identical statements can have different meanings depending on recovery history. For example, a patient writing "I feel fine" may indicate improvement or mask worsening symptoms if earlier trends suggest decline. Even when multiple signals are combined, the system frequently lacks enough information to determine which interpretation is correct. This revealed a broader pattern: a system's intended meaning, the data it records, and the conclusions users draw from it often do not align.

One limitation of RecovR was its reliance on typed patient logs. During recovery, many patients are unable to write detailed entries. To explore an alternative, I built a separate voice-based transcription system using Whisper to capture spoken recovery notes. While voice input reduced effort, it introduced new uncertainty related to timing and context, reinforcing that the core challenge lies in interpreting incomplete and ambiguous information rather than collecting more data.

Reconstructing Intent In Evolving Software Systems

During my undergraduate studies, I chose to specialize in software engineering. Through this focus, I worked on several full-stack software systems using technologies such as React, PHP, and SQL with projects such as BookTrack Library (code), The Melting Pot (code) and PetPatrol (code). Although these projects differed in application, they shared a common structure: evolving user requirements, growing codebases, and collaboration among multiple contributors. Over time, I found that understanding what a system was intended to do became increasingly difficult, even when the full code repository was available.

In each project, the main sources of information were typically not aligned. Requirements and feature descriptions outlined what the system was expected to do, but were often incomplete or outdated. The code reflected what the system actually did, embedding assumptions and design decisions that were never explicitly

documented. Tests expressed expected behavior, yet frequently captured only partial interpretations of system intent. As the projects evolved, these sources drifted apart, creating ambiguity about which version of the system's behavior was correct.

This mismatch closely resembled my earlier experiences in healthcare systems. When requirements, code, and tests each provide partial or conflicting views of intent, understanding system behavior becomes a matter of reconstructing missing context rather than reading documentation. These experiences led me to a central research question: **how can intelligent systems detect conflicts and recover intended behavior when evidence across software artifacts is incomplete or inconsistent?**

I became especially interested in AI methods that can reason across these sources to identify where intent has drifted, infer likely system behavior, and suggest meaningful fixes. This view treats software repositories not as static code collections, but as evolving records of human decisions, making them a natural setting for studying how context can be reconstructed from incomplete information.

Research Interests

My interests are to build AI systems that can reason about intent and meaning when information is incomplete, inconsistent, or noisy. In both healthcare and software systems, I have seen that understanding behavior often requires inferring context that is not explicitly recorded. In software engineering, this problem is especially clear. Large code repositories store intent across many evolving artifacts, documentation, code, and tests that rarely stay aligned over time. Instead of treating repositories as fixed sources of truth, I am interested in viewing them as imperfect records of human decision-making. This raises questions about how conflicts across artifacts can be detected, how intended behavior can be inferred when information is missing, and how uncertainty should be handled when different sources disagree.

Methodologically, I focus on AI systems that reason under uncertainty by integrating information from multiple imperfect sources. In healthcare, this includes clinical narratives, patient-reported data, and sensor measurements; in software systems, it involves documentation, code, and tests. I aim to design methods that detect inconsistencies and recover intent when these representations conflict, allowing systems to operate reliably despite incomplete or noisy inputs.

Teaching

Teaching has complemented my research work throughout my academic journey. I served as a teaching assistant for FSE 100 (Introduction to Engineering) and received an offer for CSE 412 (Database Management). I conduct weekly lab sessions for CSE 115 (Introduction to Computer Science), helping students build programming fundamentals and problem-solving confidence. These experiences have strengthened my ability to explain complex concepts clearly. I find that academic careers are multifaceted, combining research, teaching, and building impactful systems. This motivates me to pursue an academic research and teaching career after graduation.