C-LASS

CYBERSECURITY LEARNING WITH AI FOR STATIC SYSTEMS

Requirements Document

November 26th, 2025

C-LASS (Cybersecurity Learning with AI for Static Systems)

Sponsor: Dr. Lan Zhang

Assistant Professor, School of Informatics, Computing, and Cyber Systems, Northern Arizona University

Team Mentor: Scott LaRocca

Team Members: Sean Golez, William Barnett, Kayden Vicenti, Colton Leighton

Accepted as baseline requirements for the project:

Lan Zhang	Kayden Vicenti
For the Client	For the Team
11/26/2025	11/26/2025
Date	Date

Table of Contents

Introduction

The cybersecurity industry plays a critical role in protecting the world's digital infrastructure, with global spending exceeding $100 billion annually. Within this field, Capture the Flag (CTF) challenges have emerged as one of the most engaging and effective methods for teaching and reinforcing cybersecurity concepts such as vulnerability discovery, program analysis, and reverse engineering. CTFs promote active, inquiry-based learning by simulating real-world security problems that require creativity, persistence, and technical skill. However, despite their popularity, CTFs focused on static analysis, the process of analyzing code without executing it, remain especially difficult to master. Static analysis requires deep reasoning, and many students struggle to develop these skills with limited structured guidance or expert feedback. Artificial Intelligence (AI) has the potential to enhance cybersecurity education by providing scalable, adaptive, and personalized learning support. Yet, current large language models (LLMs) fall short in addressing the specific demands of static analysis instruction. They often produce inaccurate or inconsistent explanations, lack grounding in authoritative technical content, and do not represent complex software analysis concepts with sufficient precision. As a result, while AI tutoring holds promise, its current limitations prevent it from serving as a dependable teaching tool for this highly specialized domain.

Our project sponsor, Dr. Lan Zhang, an Assistant Professor at Northern Arizona University, teaches SE450: Software Engineering, a course that focuses on software testing and vulnerability discovery through CTF challenges. In this course, students use disassembly tools to convert binary files into human-readable code and then develop Python programs utilizing frameworks such as *Angr* and *SimProcedure* to analyze and solve these challenges. However, there are currently limited structured or accessible educational resources that guide students through these tasks. Students rely heavily on instructor intervention or online explanations, which can be inconsistent and time-consuming. To temporarily fill this gap, Dr. Zhang has used ChatGPT to support students with their CTF challenges, but the model's lack of accuracy and domain depth limits its usefulness as a reliable educational assistant.

To address this issue, our team, C-LASS, is developing a web-based AI tutoring system designed specifically for static analysis CTF challenges. The system integrates knowledge graphs and Retrieval-Augmented Generation (RAG) to provide accurate, conceptually grounded, and consistent explanations of complex cybersecurity topics. This design ensures that responses remain consistent across sessions and are aligned with authoritative content curated by instructors. The envisioned tutoring agent will improve how students engage with static analysis CTFs. By visualizing the relationships among concepts, tools, and techniques, students will gain a deeper understanding of how different problems connect within the broader field of cybersecurity. The system will also enable users to efficiently trace conceptual dependencies and explore knowledge pathways at their own pace. Ultimately, this platform aims to reduce instructor workload, improve student comprehension, and promote independent learning.

Problem Statement

Dr. Lan Zhang's current workflow for teaching static analysis Capture the Flag (CTF) challenges in SE450 follows a traditional, instructor-centered model that relies heavily on one-on-one support. Each semester, students are introduced to binary exploitation and software testing through CTF challenges that require the use of disassembly tools and the Python-based *Angr* framework to analyze and solve problems. The process begins when students receive a compiled binary challenge. They must then deconstruct it using a disassembler, identify functions and control structures, and apply *Angr* scripts and *SimProcedure* implementations to reason about the program's logic. Once students generate potential solutions, they validate their results and submit code for feedback from the instructor or teaching assistants.

This workflow functions effectively in small class settings but becomes increasingly inefficient as challenge complexity increases. The existing process depends on frequent instructor intervention to clarify technical errors, explain abstract static analysis concepts, and provide step-by-step debugging guidance. Because each CTF challenge is unique and demands deep reasoning about code behavior, students often reach cognitive bottlenecks where progress halts without personalized help. The result is a fragmented learning experience that limits both instructional scalability and student independence. Furthermore, as students increasingly rely on AI chatbots, further confusion may occur since these tools are not properly equipped to generate correct and reliable answers to complex, domain-specific questions.

The current workflow suffers from several key deficiencies and missing capabilities:

- Limited structured learning support: There are limited guided materials or automated tutoring tools to assist students as they progress through static analysis challenges.
- High instructor dependency: Students rely on Dr. Zhang or teaching assistants for explanations of core concepts such as control-flow graphs, symbolic execution, and taint propagation.
- Inconsistent external resources: Students resort to online tools such as ChatGPT, which often produce inaccurate or inconsistent responses that can reinforce misconceptions.
- No centralized knowledge structure: There is no unified system for storing, retrieving, or updating CTF knowledge, examples, or solutions in an organized, reusable way.

Together, these deficiencies hinder student engagement and slow mastery of core static analysis principles. The absence of a scalable, reliable, and structured support system has created a critical gap between instructional capacity and learner demand. Addressing these issues requires a tool that can automate explanation delivery, reinforce accurate conceptual understanding, and reduce reliance on continuous human supervision.

Solution Vision

To address the challenges faced by Dr. Zhang and her students, our team proposes the development of a web-based AI Tutoring System for Static Analysis CTF Challenges. This system will combine knowledge graphs and Retrieval-Augmented Generation (RAG) to deliver accurate, consistent, and conceptually grounded explanations to students. The tutoring agent will function as an interactive, intelligent assistant capable of providing guided instruction. The goal is to create a platform that not only improves students' understanding of static analysis but also scales instruction efficiently, reducing instructor workload while enhancing learning outcomes.

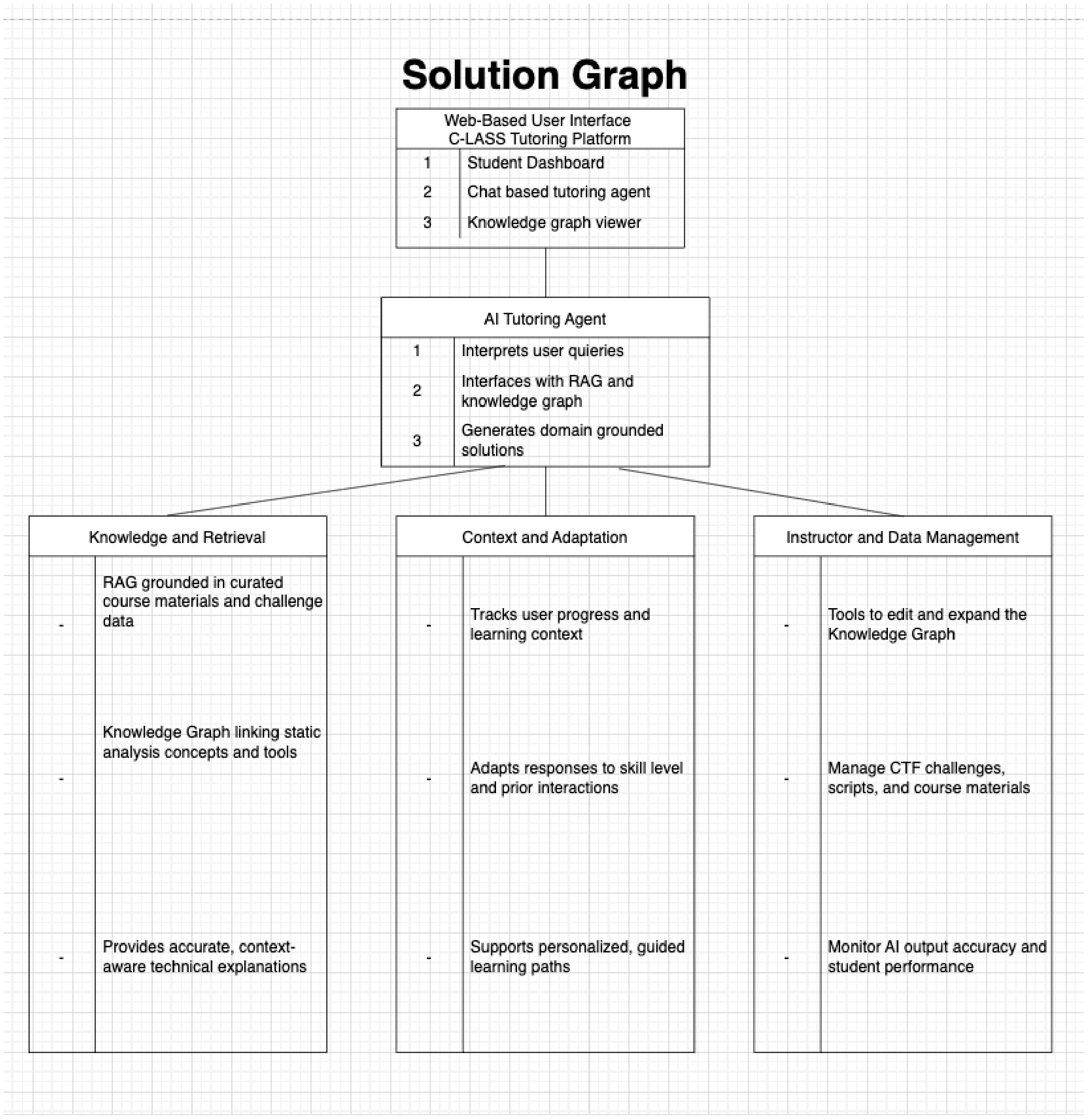
This system directly addresses the deficiencies identified in the current workflow by providing structured learning pathways. It ensures that students receive reliable, contextually relevant guidance while freeing instructors from repetitive, one-on-one explanations. The following features form the core of the proposed solution:

- AI-Driven Tutoring Interface: A text-based chat system where students can ask questions related to CTF challenges and receive explanations that reference accurate, authoritative sources.
- Knowledge Graph Integration: Visual networks that map concepts, relationships, and dependencies within static analysis topics (e.g., symbolic execution, control-flow graphs, and taint tracking).
- Retrieval-Augmented Generation (RAG): Enhances large language model responses by grounding them in curated instructional materials, code examples, and challenge documentation.
- Instructor Editing Tools: Enables instructors to expand, revise, or update the knowledge graph and training materials to align with evolving course content.
- Context-Aware Responses: The AI will recognize follow-up questions and adapt explanations to the student's progress or level of understanding.

The tutoring system will rely on a combination of curated course materials, instructor-provided documentation, and publicly available static analysis resources. Data inputs will include CTF challenge descriptions, *Angr* and *SimProcedure* scripts, class notes, and supplementary learning materials supplied by the instructor. This data will populate both the knowledge graph and the RAG database, forming the foundation for accurate, verifiable tutoring responses. When a student interacts with the AI tutor, the system retrieves relevant information from the RAG database, cross-references it with nodes in the knowledge graph, and generates a response that cites specific sources or concepts.

The core computational process involves three major steps:

1. Retrieval: The system searches the knowledge graph and document database for relevant content.
2. Augmentation: Retrieved data is provided to the LLM as contextual information.
3. Generation: The LLM produces a structured, referenced, and pedagogically aligned explanation for the user.


Solution Graph
Web-Based User Interface
C-LASS Tutoring Platform
1 Student Dashboard
2 Chat based tutoring agent
3 Knowledge graph viewer
AI Tutoring Agent
1 Interprets user quieries
2 Interfaces with RAG and knowledge graph
3 Generates domain grounded solutions
Knowledge and Retrieval
- RAG grounded in curated course materials and challenge data
- Knowledge Graph linking static analysis concepts and tools
- Provides accurate, context-aware technical explanations
Context and Adaptation
- Tracks user progress and learning context
- Adapts responses to skill level and prior interactions
- Supports personalized, guided learning paths
Instructor and Data Management
- Tools to edit and expand the Knowledge Graph
- Manage CTF challenges, scripts, and course materials
- Monitor AI output accuracy and student performance

By automating structured guidance, the system will transform the current instructional workflow from an instructor-centered model to a scalable, self-guided learning environment. The trade-off is that instructors will initially invest time in curating materials and validating responses, but this short-term effort results in a long-term, reusable system that supports future course iterations and continuous content improvement. The system's broader impact extends beyond SE450. Once developed, it could be adapted for use in other courses or domains that involve complex reasoning, such as program analysis, formal verification, or reverse engineering. It has the potential to reform how cybersecurity and computer science concepts are taught by combining adaptive AI technology with structured, instructor-driven content.

Alternative solutions were considered, such as developing a static library of tutorials, expanding office hours, or relying on existing AI tools like ChatGPT. However, these options fail to address the key limitations identified in the problem statement, specifically, the lack of structure, consistency, and conceptual grounding. The proposed system uniquely integrates knowledge graphs and RAG to provide explanations that are both dynamic and anchored in verified information. This approach balances scalability with accuracy, creating a long-term, sustainable solution tailored to the sponsor's needs. By addressing the existing workflow deficiencies and providing structured, adaptive, and accurate tutoring, the system will empower students to master static analysis challenges more independently and confidently. For Dr. Zhang and her students, this solution not only fills an immediate instructional gap but also lays the foundation for a future of more interactive, intelligent, and accessible learning in cybersecurity education.

Project Requirements

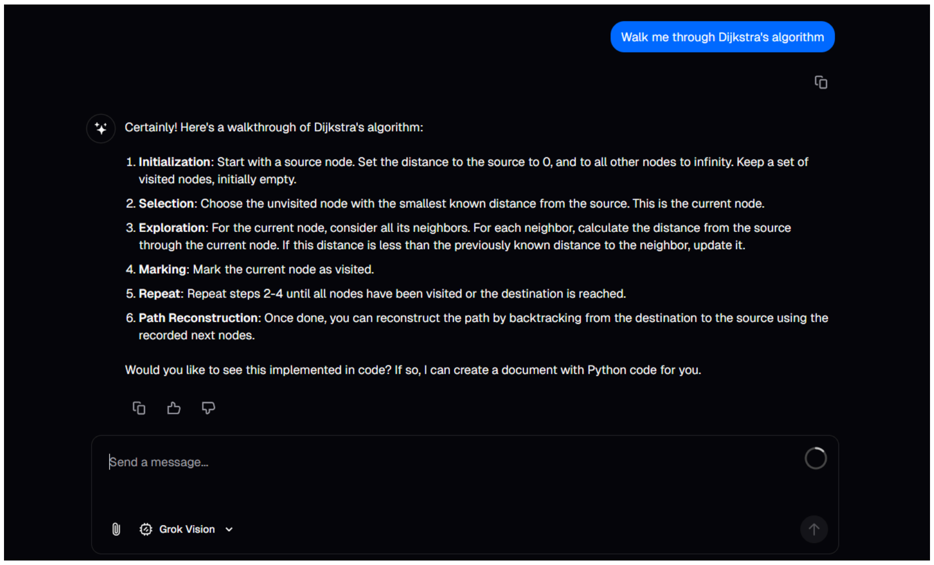
In order for our project to be successful, we outline the following requirements that are essential for delivering the chatbot tutoring agent, the domain-specific knowledge graph, and the user management components. Before detailing the system's functionality, it is important to outline the domain-level needs that define what our project is trying to accomplish. Students need a timely and conceptually grounded response; in contrast, instructors require a structured way to organize course knowledge and ensure consistent explanations without the need for a one-on-one explanation. These domain-level requirements motivate the specific functional, non-functional, and environmental requirements described in the following sections.

DR 1 Chatbot Tutoring Agent

The system must provide a knowledgeable and responsive chatbot interface that will allow users to interact with the system through natural language queries. The chatbot agent must be able to retrieve contextual information, analyse the natural language query, and guide the user to the correct solution.

Functional Requirements

- Must-Have
  - As a user, I want to be able to send messages to the chatbot so it feels like I am having a conversation with the chatbot tutor
  - As a user, I would like the chatbot to respond clearly and in a conversational language so that I can easily understand the solutions to my questions
  - As a user, I would like the chatbot to be able to understand what I want out of my questions and receive the corresponding responses so that I can be appropriately helped.
  - As a user, I would like to be able to see my prior conversational history and responses so that I can have a better understanding later on.
  - As a user, I would like the chatbot to relay to me any issues with my requests and inquiries about information so that I can get the help that I need out of the application.



Walk me through Dijkstra's algorithm

Certainly! Here's a walkthrough of Dijkstra's algorithm:

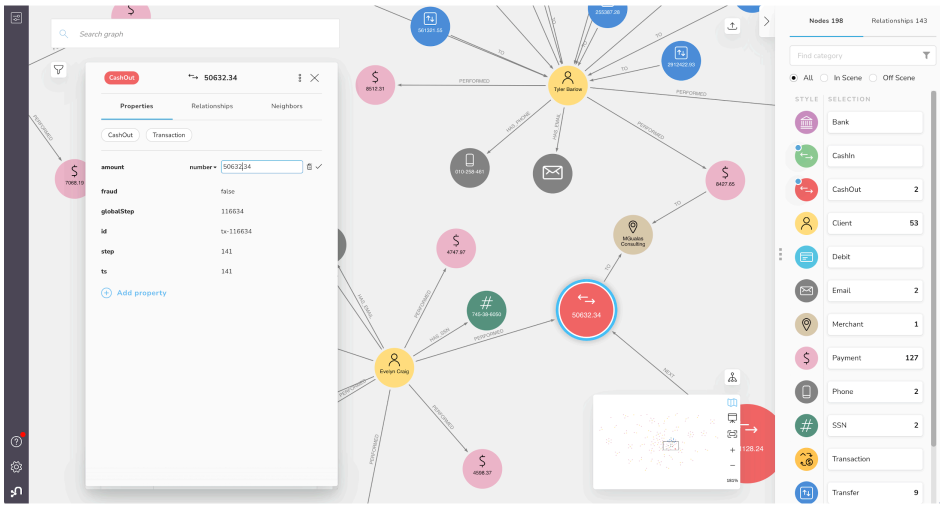
1. **Initialization**: Start with a source node. Set the distance to the source to 0, and to all other nodes to infinity. Keep a set of visited nodes, initially empty.
2. **Selection**: Choose the unvisited node with the smallest known distance from the source. This is the current node.
3. **Exploration**: For the current node, consider all its neighbors. For each neighbor, calculate the distance from the source through the current node. If this distance is less than the previously known distance to the neighbor, update it.
4. **Marking**: Mark the current node as visited.
5. **Repeat**: Repeat steps 2-4 until all nodes have been visited or the destination is reached.
6. **Path Reconstruction**: Once done, you can reconstruct the path by backtracking from the destination to the source using the recorded next nodes.

Would you like to see this implemented in code? If so, I can create a document with Python code for you.

- Should-Have
  - As a user, it would be nice if the chatbot could retain provided documents and conversations and use them to produce more efficient responses.
  - As a user, I think the chatbot should be able to display a thinking message so that I am not confused about whether the chatbot is working or not.
  - As a user, I would like to be able to have question templates of good prompts for the chatbot.
- Could-Have
  - As a user, I think that it would be helpful to provide speech-to-text and text-to-speech features so that I can better understand presented to me through another sense
  - As a user, I would like to be able to select predefined responses from the chatbot so that it feels more interactive and personable.
  - As a user, I would like to be able to give the chatbot feedback on answers that are generated well or incorrectly, so that the chatbot can improve.
- Wont-Have
  - The system will not support voice-call interactions in this version

Non-Functional Requirements

- The system must be able to start providing a knowledgeable response within an average of 2 seconds from user input
- The system must be able to correctly interpret the user's intent from the input 90% of the time
- The system must be able to maintain usability of the system across desktop and mobile web browsers
- The chatbot service must be able to handle at least 50 users simultaneously without significant user degradation
- The chatbot must not store any personally identifiable information



amount	number 50632.34
fraud	false
globalStep	116634
id	tx-116634
step	141
ts	141

Bank	
CashIn	
CashOut	2
Client	53
Debit	
Email	2
Merchant	1
Payment	127
Phone	2
SSN	2
Transaction	
Transfer	9

Environmental Requirements

- An internet connection is required for natural language processing services done through API services
- The chatbot's data processing must comply with NAU's privacy and data security standards

DR 2 Domain-Specific Knowledge Graph

The system must have a curatable knowledge graph for teachers provided with domain-specific information. It is required that the chatbot agent use the information provided in this graph to inform its responses.

Functional Requirements

- Must-Have
  - As a teacher, I want to be able to import documents directly into nodes.
  - As a user, the knowledge graph should be stored in a persistent database, so that it does not need to be frequently reconstructed.
  - As a teacher, I want to be able to create and edit in an interface that allows me to add, delete, and label nodes and edges.
  - As a teacher, I want knowledge graphs to contain interconnected sub-graphs so that I can add more detail into specific topics, while keeping everything connected.
- Should-Have
  - As a teacher, I would like to be able to import bulk, unorganized documentation, and for the system to automatically process the provided data into a knowledge-graph format.
  - As a teacher, I want to be able to easily edit graphs in an intuitive editor that does not require me to write any code.
  - As a teacher, I want the knowledge graph requirements to be very flexible, so that I can design it however I need.

    - As a user, I would like the system to be able to store multiple knowledge graph databases, so that AI tutoring agents can have knowledge in multiple specific domains.
- Could-Have
    - As a teacher, I want to be able to expand and compress portions of the graph structure to make navigation and visualization easier.
    - As a teacher, I only want students in my class to be able to utilize my knowledge graph with the AI tutoring agent.
    - As a teacher, I would like to be able to add custom annotations to the knowledge graph that students can see if particular nodes are retrieved.
- Wont-Have
    - As a student, I want to be able to view the graph so that I can directly see the exact solution to a given problem.
    - As a teacher, I want course-specific information in the knowledge graph to be shareable with other teachers.

Non-Functional Requirements

- Saving and loading in knowledge graphs into the editing/visualization interface occurs at a rate of at least 0.25 seconds per node.
- Saving an edited knowledge graph into the database should efficiently append/move the data, rather than completely overwriting it each time.
- Editing knowledge graphs should be responsive, with very little input delay.

Environmental Requirements

- In production, the knowledge graph database will be hosted by a 3rd party service.
- The knowledge-graph editing/visualization interface will be compatible with any modern web browser.

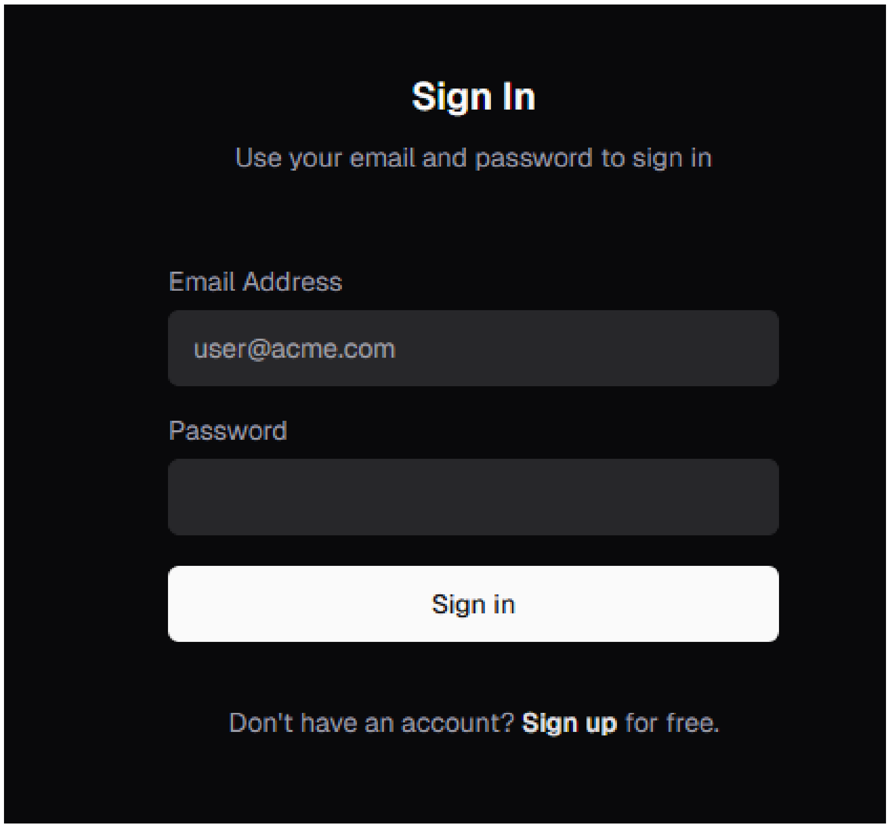
DR 3 User Management

An important specification for this system is that the experience is different depending on whether the user is a student, who will be using the chatbot as a tutor to help them learn course materials, or a teacher, who will be curating the domain-specific knowledge graph that the chatbot will be using to inform responses. So, it is important that the system can manage user information.

Functional Requirements

- Must-Have
    - As a user of the tutoring platform, I want the system to dynamically adjust available features and permissions based on my authenticated role (teacher or student), so that teachers can create and interact with tutoring agents connected to the knowledge graph, while students can only access curated learning experiences without modifying or interacting directly with the knowledge graph.

    - As a user, I would like to be able to easily create and log in to my account.


Sign In
Use your email and password to sign in
Email Address
user@acme.com
Password
Sign in
Don't have an account? Sign up for free.

- Should-Have
    - As a teacher, I would like to be able to view a list of all of the students who are able to use a tutoring agent that has access to the knowledge graph.
    - As a student, I would like my information to be collected and utilized by the chatbot for a more personalized experience.
- Could-Have
    - As an instructor, I want the system to provide insights into commonly asked topics across students, so that I can identify trends and adjust instructional content towards certain areas of interest.
    - As a user, I want my account to be linked to my school email to securely verify my identity.
- Wont-Have
    - As a student, I cannot query or view other students' data to enforce access control policies.

Non-Functional Requirements

- All information must be protected by encryption both while being transmitted and while stored.
- User verification must be complete within 5 seconds.
- The system must support at least 50 concurrent user queries to the database.

Environmental Requirements

- In production, the database must be hosted by a third-party service.

Potential Risks

With the system's requirements clearly defined, it is necessary to evaluate potential risks that the system poses that may hinder the success of our application. Anticipating these risks will allow us to effectively plan solutions and guardrails to ensure the system will behave as expected and mitigate any potential risks that arise. Below are key potential risks that the systems pose, along with our proposed solutions and mitigation strategies for each.

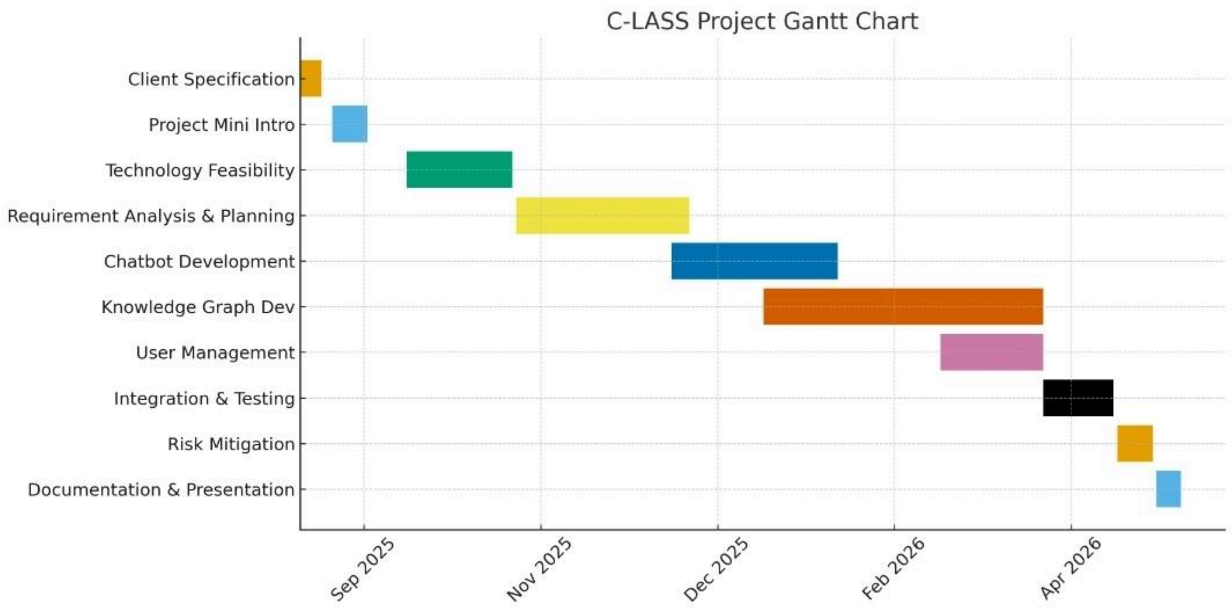
Accuracy and Misinformation Risk

- Description: With a system designed to guide students in their course materials with the agent acting as the agent, we run the risk of providing misinformation. The chatbot may generate incorrect or misleading explanations that lead the user astray rather than guide them to the correct solution.
- Severity: High - Can result in the User's scores dropping, affecting their grade
- Likelihood: Small - Early modeling would be most vulnerable; however, with further integrations and testing, errors are much less likely to occur
- Mitigation: Narrow context fields accessible to the agent to the specified content provided from the instructor, and double verification from the agent

Academic Integrity Risk

- Description: Students could potentially exploit the system and attempt to retrieve question answers rather than using the application as intended.
- Severity: High - Undermines the work that the agent attempts to achieve and impacts the students' learning experience
- Likelihood: Medium - The system would likely start with several ways to exploit the agent; however, after significant testing and review, patches can be pushed to resolve the issues
- Mitigation: Design the chatbot to respond educationally, mimicking the behaviors typical of a tutor, encouraging reasoning and guided questioning.

Project Plan



The development of the AI tutoring platform will proceed through a series of structured milestones that align with the three core design requirements (DR1-DR3). Each milestone builds upon the last, ensuring modular integration and continuous testing as the system matures.

Milestones

Previous Milestones:

1. Client Specification & Initial Requirements Meeting (September 2025)
   The team met with the client at the start of the semester to understand their instructional workflow, identify pain points in teaching static analysis CTFs, and gather the initial high-level goals and expectations for the project. This milestone established the foundation for early requirements exploration.
2. Project Mini Intro & Preliminary Analysis (September 2025)
   The team developed an introductory presentation summarizing the client's problem space, the importance of static analysis education, and the envisioned broad solution. This involved mapping the client's process, identifying early deficiencies, and outlining the system before formal requirements gathering.
3. Technological Feasibility Investigation (October 2025)
   A structured analysis was conducted to evaluate the feasibility of key system components—including LLM integration, retrieval pipelines, knowledge graph technologies, databases, and web frameworks. Alternative tools were compared, early prototypes were tested, and a preliminary technology stack was selected to ensure the project could be implemented reliably and within constraints.

Current Tasks:

4. Requirement Analysis & Planning (Nov 2025 – Dec 2025)
   The team will gather, refine, and validate functional and non-functional requirements. This includes understanding user interactions, data handling, and technical constraints across chatbot, graph, and management.
5. Chatbot Tutoring Agent Development (Dec 2025 – Jan 2026)
   Focused on implementing the chatbot interface and backend logic. Key functionalities include message handling, conversational flow, context storage, and real-time feedback messages.

Upcoming Tasks:

6. Knowledge Graph Database and Interface (Jan 2026 – Mar 2026)
   Teachers' tools will be developed for adding, deleting, and labeling nodes and edges in a knowledge graph. This phase includes document imports, persistent graph storage, and data visualization.
7. User Management Module (Mar 2026)
   Development of account creation, deletion, and authentication systems. Role-based access will differentiate between student and teacher permissions, ensuring secure and scalable access control.
8. Integration & System Testing (Mar 30 – Apr 19, 2026)
   The chatbot, knowledge graph, and user management components will be integrated. Testing will ensure efficient communication among system components and verify performance under concurrent usage.
9. Risk Mitigation & Final Review (Apr 20 – Apr 30, 2026)
   System adjustments will be made to reduce misinformation and maintain academic integrity. The chatbot will be fine-tuned to generate educational, guided responses.
10. Documentation & Presentation (May 1 – May 8, 2026)
    The final milestone will produce deliverables, including the final system documentation, user guide, and project presentation.

Conclusion

As the rise of Artificial Intelligence continues to expand in our technologically advancing world, we must consider how to effectively refine and improve current agents in the world of education. Presently, as discussed earlier, a persistent challenge remains: AI systems alone are not yet capable of providing accurate and reliable information within specialized academic domains.

Standard Large Language Models are not refined enough to explain the depth and contextual understanding of specialized fields within academic disciplines. This limitation emphasizes the importance of providing tools that leverage the strengths of AI alongside domain-specific academic materials to enhance educational outcomes. Our solution of implementing a tutoring chat system addresses this problem by introducing the capabilities of Artificial Intelligence supplemented with a knowledge graph utilizing domain-specific class materials provided by the instructors.

Within this document, the project's core structure has been defined through a series of domain-level, functional, performance, and environmental requirements that serve as steps to mark our substantial progress towards transforming our project’s plan into a conceptual foundation. Moreover, the risk assessment section highlights significant areas of concern within our conceptual framework that will aid in ensuring the reliability and trustworthiness of our proposed solution.

Moving forward, the key insights gathered from the creation of this document, alongside our prototyping and testing, will serve as a promising step towards the creation of AI tutoring tools that complement human teaching. As we continue to revise and refine our project in collaboration with our educator, this chatbot system has the potential to be a reliable educational assistant capable of supporting students in specialized fields.

Glossaries and Appendices

Capture the Flag (CTF): a cybersecurity challenge designed to teach and assess skills in areas such as vulnerability discovery, reverse engineering, and program analysis. Participants solve structured problems to locate “flags,” promoting hands-on learning and critical reasoning similar to real-world security analysis.

Knowledge Graph: a structured data model that represents entities and their relationships through interconnected nodes and edges. It enables context-aware reasoning, allowing systems to organize and retrieve information more precisely for domain-specific learning and analysis.

Retrieval-Augmented Generation (RAG): an AI method that combines language model generation with document or graph retrieval. It allows an LLM to reference curated data sources to produce accurate, grounded, and contextually consistent responses.

Angr (Python Framework): An open-source Python platform for binary analysis that combines static and dynamic symbolic execution. It enables AI tutoring agents to explore program paths, recover control-flow graphs, and teach vulnerability analysis in Static Analysis Challenges.

SimPro (simulation framework): A simulation-based analysis technique used with Angr to model binary program behavior through custom SimProcedures. It enables the AI tutoring agent to replicate instructor-guided reasoning, letting students explore program logic, test solutions, and receive interactive feedback in Static Analysis CTF challenges.