CS510 Course Project Proposal

Track: Development Track – AI-Powered Interview Guide for CS Students

---

Team Members

Name	Email
Xinyao Qiu	xinyaoq3@illinois.edu
Richard Yang	yueqian8@illinois.edu
Fangyang Xu	xu94@illinois.edu
Hongbin Yang	hongbin3@illinois.edu

Functions and Users

We plan to develop a new Web-based application designed for computer science students preparing for technical job interviews. The main goal of this tool is to provide an organized and searchable collection of technical interview questions and real interview experiences.

The major functions of the tool include:

- A search feature that allows users to find relevant technical questions or interview experiences by keyword, company, or topic (e.g., data structures, operating systems, system design).
- A personalized recommendation system that suggests popular or relevant technical questions based on users' previous searches and interactions.
- A user submission system, where students can contribute their own interview experiences or technical questions.
- LLM Assistant: We will integrate a Large Language Model assistant to help users quickly understand, summarize, or extend interview questions. For this, we currently plan to use Gemini-2.0-Flash-Thinking or QWQ-32B due to their good cost-performance ratio.

- Bookmarking and saving features to help users keep track of important questions or articles.
- A homepage feed that highlights trending or recommended content for each user.

The primary users of this software tool are university students majoring in computer science or related fields who are preparing for software engineering internships or full-time job interviews.

Significance

In the competitive landscape of technical interviews, candidates often struggle to find high-quality, up-to-date, and well-structured resources to practice and refine their problem-solving skills. Existing platforms provide a vast collection of problems, but they often lack community-driven discussions, personalized recommendations, and the ability to compare different solution approaches dynamically. Our proposed system aims to bridge these gaps by creating a collaborative platform where users can contribute real interview questions, discuss solutions, and leverage AI-powered assistance to enhance their preparation.

One of the key challenges in technical interview preparation is the lack of real-world, company-specific questions that accurately reflect the latest industry trends. Many candidates rely on outdated question banks, which may not align with current hiring practices. By allowing users to upload and share real interview questions, our platform ensures that the content remains fresh and relevant. Furthermore, the discussion-driven model enables users to compare multiple solution strategies, learn from different perspectives, and refine their coding skills through peer interactions.

Another significant advantage of our system is the integration of an AI-powered assistant, which allows users to engage with LLM for instant explanations, alternative solutions, and clarifications. This feature helps candidates quickly grasp concepts they may find challenging, providing immediate feedback rather than waiting for responses from the community. Comparing AI-generated solutions with human-submitted ones encourages critical thinking and helps users evaluate different approaches in terms of efficiency, readability, and maintainability.

The recommendation system embedded in our platform further enhances user experience by suggesting questions based on an individual’s past interactions, skill level, and areas for improvement. Unlike static problem lists found on traditional platforms, our system personalizes the learning experience, making interview preparation more effective and targeted.

By addressing these pain points, our project not only enhances the way candidates prepare for technical interviews but also fosters a collaborative and knowledge-sharing community. In the long run, this system can benefit recruiters and hiring managers as well, as they gain insights into common solution patterns, popular problem topics, and candidate performance metrics. The impact of this tool extends beyond individual users, potentially shaping the broader ecosystem of technical hiring by promoting better learning resources and more informed problem-solving strategies.

Approach

We divide our system into 3 main parts: a community discussion, an LLM assistant, and a recommendation system. For the community discussion part, we are exploring two options:
(1) Using React, Express, and MongoDB for full control and flexibility
(2) Leveraging existing solutions like giscus or GitHub Discussions with webhooks to sync data with our backend.
We will continue to monitor the data quality and implementation difficulty of these two solutions.

For the LLM assistant part, we temporarily choose to use Gemini-2.0-Flash-Thinking or QWQ-32B to generate outputs for their good cost performance.

For the recommendation system, we aim to use Elasticsearch for content suggestions, given its compatibility with our node.js backend.

We anticipate that there may be some potential barriers to indexing the data for our recommendation system, as most of us have no experience with it. If we cannot complete the planned work on time, to mitigate the risk, we will downgrade our recommendation system to recommending by a single factor such as only users' recent replies, or even downgrade to a search-only system.

Evaluation

In order to evaluate the effectiveness and correctness of the platform, we will use the following methods:

- Basic functional testing: We will conduct unit and integration tests to ensure that all core functionalities (e.g., problem contribution, AI assistant interaction, solution discussion forums, and recommendation system) work as expected in different user scenarios.

- User testing and feedback: We plan to invite a small group of target users (e.g., software engineering students and job seekers) to try out the platform and provide feedback through surveys and interviews. This will help evaluate the usability and relevance of the AI assistant's recommended content.

- Comparative Evaluation: We will compare the performance and user experience of our platform to existing tools by collecting metrics such as the number of problems solved by users and satisfaction ratings with the content of the site. The comparison will be used to see if there are areas where our platform can be improved.

- AI Assistant Response Evaluation: For the AI Assistant feature, we will evaluate the quality of its suggestions using manually labeled benchmarks to determine if the responses generated are accurate, useful, and readable.

Timeline

- **Week 1**: Finalize requirements and user flow design; set up development environment and tech stack.

- **Week 2**: Implement core features and design basic UI interfaces.

- **Week 3**: Integrate AI assistant for question explanation and comparison; begin testing core functionality.

- **Week 4**: Build the recommendation engine and user profiling system; continue UI improvements and bug fixing.

- **Week 5**: Conduct user testing, collect feedback, and refine features based on insights.

- **Week 6**: Final round of testing; prepare final demo and documentation.

Task Division

- **Xinyao Qiu**: Implement backend service and interact with database.

- **Richard Yang**: Integrate the LLM service into our system and find a full-stack solution for the community forum.

- **Fangyang Xu**: Implement fronted service and interact with backend interfaces.

- **Hongbin Yang**: Implement backend service and integrate with front end interfaces.