Interwise: Integrating LLMs for Interview Practice

Group 30

Xinyao Qiu, Richard Yang, Fangyang Xu, Hongbin Yang

Background & Motivation

Background

- Tech interviews emphasize standardized CS questions
- Candidates often rely on large, unstructured question banks
- Lack of personalized, efficient preparation tools

Motivation

- Interview prep is time-sensitive and goal-driven
- LLMs can generate adaptive question paths
- Help users learn efficiently and stay focused

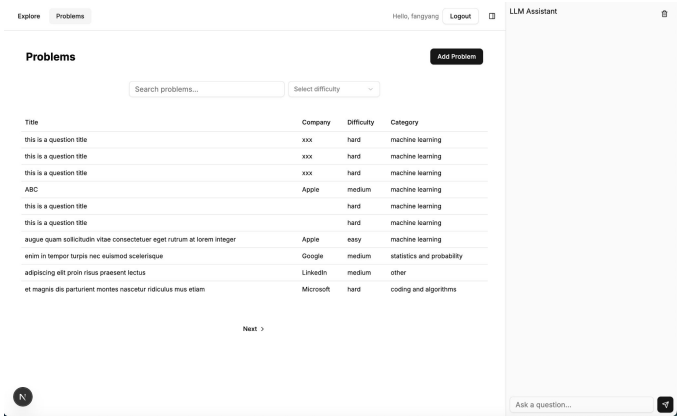
System Overview & Key features

Overview:

- Problem database with structured tags (company, difficulty, category)
- LLM-powered Q&A assistant on each problem page
- Editable Q&A and knowledge tagging
- Personalized via MongoDB text search

Key features:

- Real-time chatbot support for explanations
- Filter by company, topic, and difficulty
- Community-powered editable answers



Frontend Major Functions

Authentication and Session Management: Enables users to sign up, log in, and maintain a persistent session so they can access personalized question recommendations.

Question and Answer CRUD: Allows authenticated users to create, edit, and delete their own questions and answers, and to browse all content submitted by others.

Search and Filtering: Supports keyword-based search and difficulty-level filters to help users quickly find relevant questions.

LLM Assistant: Provides an always-available sidebar powered by an LLM, enabling users to ask for guidance or clarification on any page.

Frontend Implementation Details

Tech Stack: Next.js, Gemini 2.5 Flash
Routes & Features:

- `/explore` – a table of recommended questions
- `/problems` – a searchable, filterable question table and an add problem button
- `/problems/:id` – a question with answers and an add answer button

LLM Assistant:

- Collapsible panel
- Powered by Gemini 2.5 Flash via Vercel's AI SDK

Backend Major Functions

User Management: This module handles authentication, authorization, and user profile management.

Question Management: This component enables users to create, retrieve, and manage questions.

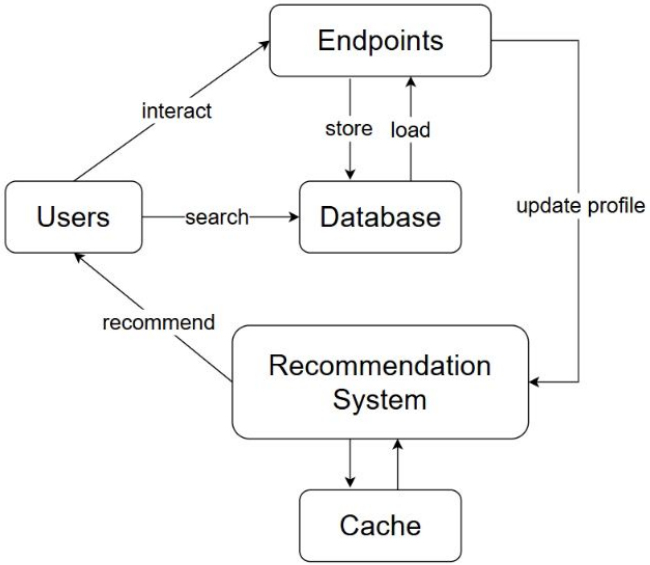
Answer Management: This module allows users to respond to questions and interact with the answers of others.

Voting and Interaction: Users can like/dislike content and bookmark relevant posts.

Search and Recommendation: This optional module supports full-text search and personalized question suggestions using MongoDB search engine.

Backend Implementation Details

Tech Stack: Node.js (Express), MongoDB, Mongoose, Redis



Architecture: RESTful API with modular separation

- `routes/` – define endpoints
- `controllers/` – handle HTTP logic
- `utils/` – some business logic functions
- `models/` – DB access using Mongoose

Backend Implementation Details (Cont.)

Authentication & Security:

- JWT-based authentication
- Role-based access control
- Password hashing with bcrypt
- Rate limiting with Redis

Caching:

- Redis for caching hot endpoints

Deployment:

- Deployed on Render

Method	Endpoint	Description
POST	/api/v1/auth/register	Register a new user account
POST	/api/v1/auth/login	Authenticate user and return JWT token
GET	/api/v1/current-user	Retrieve user profile
PATCH	/api/v1/update-user	Update user profile fields
GET	/api/v1/questions	List all questions with filters
POST	/api/v1/questions	Create a new question
GET	/api/v1/questions/:id	Retrieve a specific question's detail
PATCH	/api/v1/questions/:id	Edit an existing question
DELETE	/api/v1/questions/:id	Delete a question (owner only)
GET	/api/v1/answers	List all answers with filters
POST	/api/v1/answers	Submit an answer
GET	/api/v1/answers/:id	Retrieve a specific answer's detail
PATCH	/api/v1/answers/:id	Edit an existing answer
DELETE	/api/v1/answers/:id	Remove an answer
POST	/api/v1/vote	Like/dislike a question/answer
POST	/api/v1/bookmark	Bookmark a question/answer
GET	/api/v1/questions?search=...	Perform full-text or semantic search
GET	/api/v1/users/recommendations	Get personalized question suggestions

Evaluation

We evaluated our system on functionality, performance, and usability.

- Functionality Validation
  - End-to-end tested with Postman
  - Unit tested with Jest
  - Verified core APIs: correctness, auth, errors
- Performance Benchmarking
  - Load-tested with JMeter (100 users, 60s)
  - Most endpoints < 100 ms
  - Redis caching reduced latency by ~60%
- Usability Evaluation
  - 10-user pilot study on live system
  - Tasks: ask, vote, search
  - Positive feedback on ease of use
  - Suggestions: better tag autocomplete, answer sorting

Endpoint	Avg. Latency
GET /api/v1/questions?limit=10	72 ms
POST /api/v1/questions	104 ms
GET /api/v1/answers?limit=10	88 ms
POST /api/v1/answers	95 ms
GET /api/v1/users/recommendations	143 ms

Average Response Time per Endpoint

Future Work

Deepen LLM Assistant Integration

- Enable multi-turn conversation and contextual memory
- Add support for question clarification and solution explanations
- Evaluate response quality with user feedback or benchmarks

Build Recommendation and Personalization

- Implement question suggestions based on user history
- Explore content ranking via keyword embeddings or behavior signals
- Introduce homepage feed with trending and relevant questions

Expand Community and Interaction

- Add voting, commenting, and answer contributions
- Support user profiles and activity history
- Introduce moderation tools and content reporting

Thanks for watching!