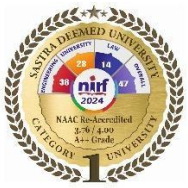
SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

Internship Projects at Vizh AI Solutions

AI Coding Platform

INDUSTRIAL PROJECT

Submitted By

Sanjai S (126018042)

B.Tech CSBS

AUGUST 2025

SCHOOL OF COMPUTING - SASTRA DEEMED TO BE UNIVERSITY

THANJAVUR - 613401

Submitted To

Dr. DEVI SRI NANDHINI M
Asst. Professor – III, SOC
SASTRA Deemed-to-be University
Thanjavur-613401

Dr. RAJILAL M V
Asst. Professor – II, SOC
SASTRA Deemed-to-be-University
Thanjavur-613401

VIZH AI SOLUTION

CERTIFICATE OF INTERNSHIP

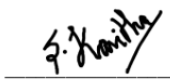
This is to certify that

Sanjai Sivakumar

has successfully completed a one-month internship as a Software Engineering Intern at Vizh AI Solution from June 5, 2025 to July 6, 2025.

During the internship period, he demonstrated excellent learning ability, proactive problem-solving skills, and contributed meaningfully to our development projects.

Jul 12 2025
Date


Signature

DECLARATION & NOVELTY STATEMENT

I, **Sanjai S**, bearing registration number **126018042**, student of **Computer Science and Business Systems**, **SASTRA Deemed To Be University**, hereby declare that the internship report titled:

"Internship Report at Vizh AI: AI Coding Platform"

is the result of my own work carried out during my **one-month internship** at **Vizh AI Solutions**. The work described in this report is original, has not been copied from any source, and has not been submitted for any other academic requirement. All sources of information have been duly acknowledged.

I also affirm that the projects described in this report exhibit significant novelty and innovation , including:

- Development of a **VS Code–style, AI-integrated coding platform** capable of generating coding challenges, test cases, and automated evaluations using **LLaMA 2**.

This report reflects the genuine technical work, challenges, and learnings I have undertaken during my internship, contributing to Vizh AI's mission of reimagining recruitment through AI-driven solutions.

Place : Thanjavur
Date : 13.08.2025

Name : Sanjai S
Registration Number: 126018042

Examiner 1 **Examiner 2**

INTRODUCTION

As a **Software Engineering Intern (SWEI)** at **Vizh AI Solutions**, I contributed to flagship project aligned with the company's motto **"Agentic Recruitment for Enterprise"** – using AI agents to modernize hiring. The project was an **AI-Powered Coding Interview Platform**, akin to LeetCode, designed to automatically generate coding challenges and evaluate solutions. In this report, I detail project architecture (with architecture diagrams), the technologies used, merits/demerits, and comparisons with existing solutions.

AI-Powered Coding Interview Platform

Architecture & Workflow

The coding platform provides candidates with a web-based **IDE (similar to VS Code)** where they solve algorithmic problems. The **React frontend** (monaco editor) communicates with a **FastAPI backend** (Python) that manages problem data, code execution, and AI-driven operations

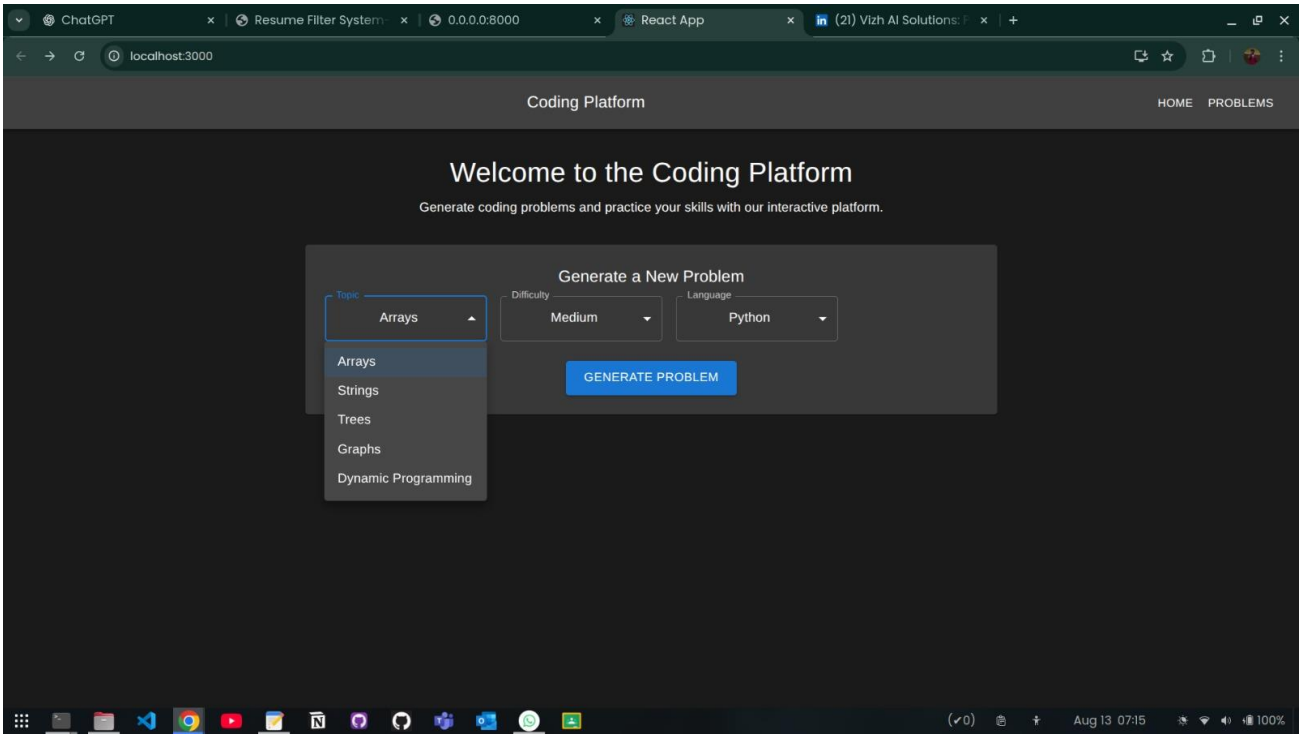
Interviewer/Recruiter logs into the platform and either creates a custom coding problem or uses the AI to generate one.

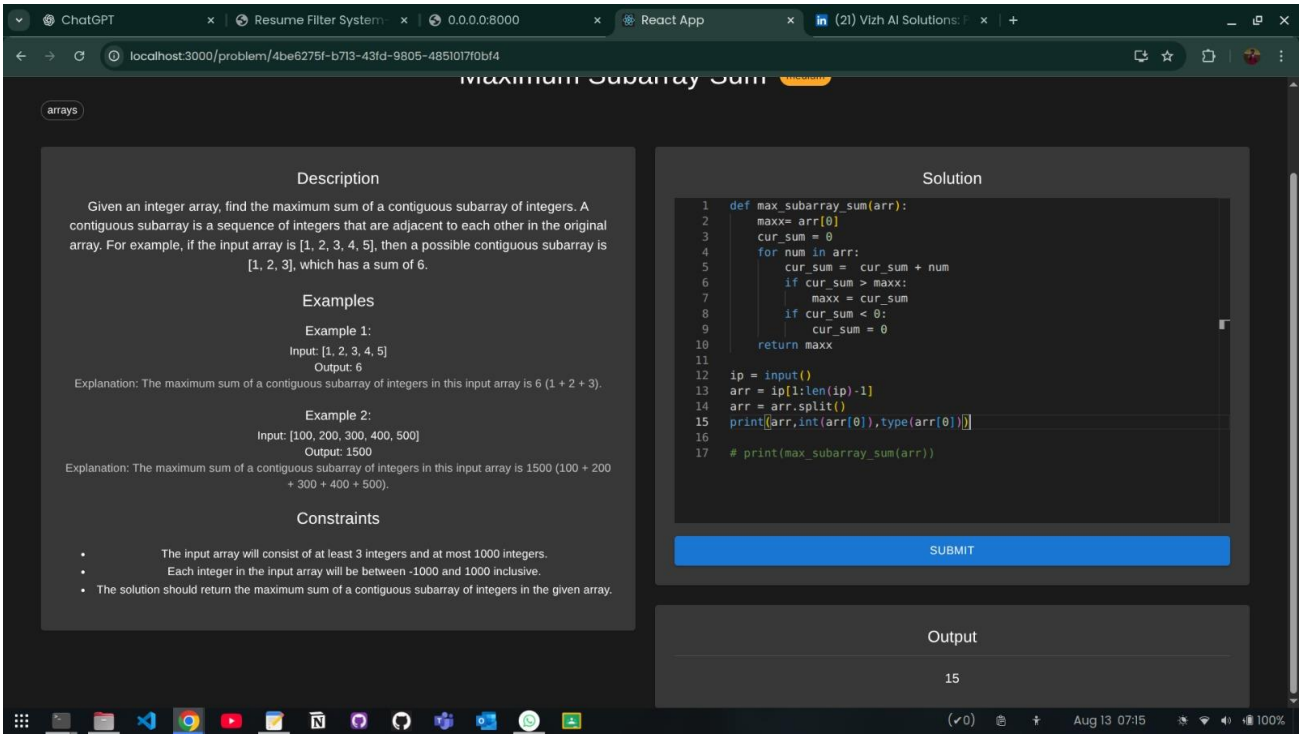
- The **backend** stores problems and test cases in a database (SQL via SQLAlchemy/Alembic).
- A **candidate** selects or is assigned a problem, writes code in the frontend, and submits it.
- The code is sent to the **backend**, which runs it in a sandboxed execution environment against pre-generated test cases.
- Meanwhile, our LLM (Meta's **Llama-2-7b** via llama-cpp-python) is used both for generating problem statements/testcases and for evaluating answers (by checking logic or comparing outputs).
- The backend returns the result (pass/fail, scores, or feedback) to the frontend, where the candidate sees their score and next steps.

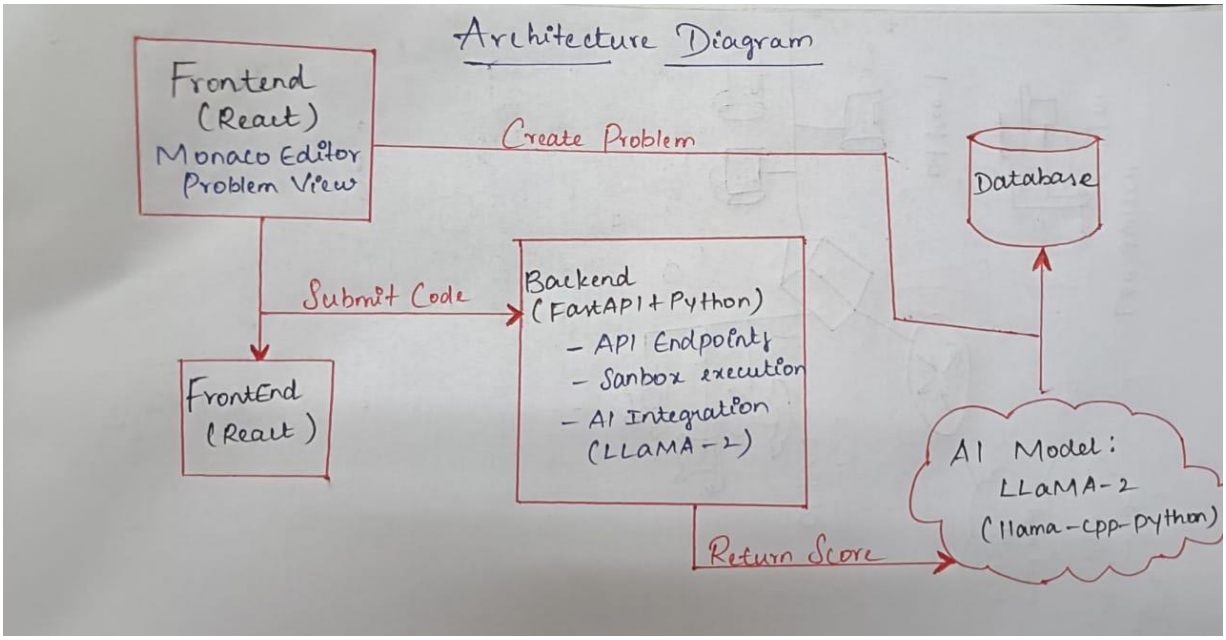
This architecture is similar to modern coding assessment systems, but with an added AI layer: the LLM autonomously generates and adapts content. FastAPI (with Uvicorn) handles HTTP requests, python-dotenv manages credentials, and SQLAlchemy/Alembic handle persistence.

Frontend (React) (IDE)	**⇄ REST APIs (FastAPI) ↔ Backend (Python)**	**Database**
- Code editor UI (Monaco)	1. Candidate submits code/test requests →	- Stores problems,
- Problem description display	2. Backend runs code against testcases (in **Exec Engine**)	test cases,
- Displays results/feedback	3. Backend uses LLM (Llama-2) to generate or validate Qs & tests	candidate data
	4. Results and scores returned to frontend	

IDE PICTURES







Technologies Used

- **Frontend**: React.js with Monaco editor for a rich coding interface. This provides a familiar development environment and allows custom UI/UX similar to VS Code.
- **Backend**: FastAPI (Python) for the RESTful API and orchestration, running under Uvicorn. Key Python libraries include llama-cpp-python (to run Llama-2 locally), sqlalchemy/alembic for database schema, and standard packages (requests, tqdm, etc.) for utility.
- **AI Model**: Meta's *Llama-2-7b* (via llama.cpp) is used on-premise to generate problem statements, sample test cases, and even to help evaluate code correctness. This avoids per-call API costs and keeps data local.
- **Execution Engine**: The backend compiles/runs submitted code (e.g. via subprocess or a container) against test cases to verify correctness. The outputs are compared to expected results.
- **Data**: A SQL database (managed through SQLAlchemy) stores problems, test cases, and candidate submissions. Environmental variables (via python-dotenv) hold secrets like DB credentials.

Merits (Advantages)

- **Automated Pre-Screening** – Automating the coding round acts as an efficient filter. As Toggl Hire points out, online coding tests "automate the initial filter, saving you valuable time" in recruiting. Our platform immediately ranks candidates by code score, focusing recruiter attention on qualified developers.
- **Skill-Based Assessment** – Unlike resumes or interviews that may only hint at ability, a live coding test shows "how candidates actually perform". The platform tests actual coding skill, providing an objective measure beyond credentials.
- **Customizable Content** – By using an LLM, questions and testcases can be generated or tailored on demand. Whereas standard platforms like LeetCode have fixed problem libraries, our system can create problems on the fly, targeting specific data structures or difficulty levels. This flexibility lets recruiters adjust topics (arrays, strings, etc.) and complexity in real time.
- **Immediate Feedback** – Candidates get instant results upon submission. The UI can display scores, passed testcases, or hints dynamically. Such rapid feedback (gamification) keeps candidates engaged and informed.
- **Scalability** – Once deployed, the system can handle many simultaneous candidates without needing extra interviewers. This is especially useful for large hiring drives or hackathons.

Demerits (Challenges)

- **Unrealistic Interview Environment** – Critics note that algorithmic coding tests can feel "unnatural" and time-consuming for candidates. In practice, developers work on projects, not puzzles, so this screening may not reflect day-to-day work. There's a risk of **candidate frustration** or anxiety (the "freeze" on a coding riddle).
- **Limited Scope** – Coding platforms evaluate only one dimension of a developer's ability (problem-solving and coding speed). Soft skills, design sense, and teamwork are ignored. As one perspective argues, focusing on a single technical metric "feels wrong" and overlooks other important developer traits.
- **Dependence on AI Accuracy** – LLM-generated questions and testcases may have errors or ambiguities. While Llama-2 can draft problems, its outputs may require human review. Similarly, automated evaluation could misjudge correct solutions if test cases are incomplete. There is a risk of **false negatives/positives**, so manual validation may still be needed.
- **Cheating Risk** – With time and internet resources, candidates could use AI tools to generate solutions. To mitigate this (and as noted by Toggl Hire), such platforms must update question banks and possibly include anti-cheating features like code similarity checks.
- **Resource Overhead** – Running a large LLM like Llama-2 (even the smaller 7B version) requires significant CPU/GPU resources. Ensuring low-latency code evaluation and generation at scale may demand powerful servers.

Comparison with Existing Platforms

Popular coding assessment tools (LeetCode, Codility, HackerRank) offer extensive libraries of challenges across languages. These are battle-tested and have vast user bases. However, they rely on **static content** – the set of questions is pre-made by experts. In contrast, our platform's use of AI means **dynamic content creation**. For example, if a recruiter wants a new problem on balanced trees with specific constraints, our LLM can generate it instead of digging through a fixed repository. This on-demand generation is a key improvement over existing work.

On the other hand, mainstream platforms have mature anti-cheating measures and community support. Our platform, being custom-built, lacks these refinements initially. We aim to differentiate by seamless integration (IDE in-browser), and by tailoring questions precisely to company needs – an area where one-size-fits-all solutions fall short. Notably, while critics argue coding tests waste candidates' time, the benefit of our AI-driven approach is to make each test **as relevant and efficient as possible**, focusing on necessary skills for the role.

In summary, our coding platform leverages AI to extend traditional assessment methods. It automates the initial screening to save recruiter hours and provides a more **accurate evaluation than resumes**, though it must be balanced against the potential stress and narrow focus inherent to algorithmic coding tests.

Conclusion – AI Coding Platform

During my internship, I contributed to building an AI-driven coding assessment platform designed to streamline the technical screening stage of recruitment. The platform's VS Code–style IDE, coupled with LLaMA-2's ability to generate custom coding questions, test cases, and automated evaluations, provides a flexible and efficient alternative to static coding sites. Recruiters can tailor questions by topic, difficulty, and constraints, ensuring assessments align closely with the job role.

Compared to existing platforms like LeetCode or HackerRank, our solution offers **on-demand, context-specific problem generation**, removing dependency on pre-built question banks. This enables more targeted evaluation and helps identify candidates with the precise skill set required. The system's automation reduces manual effort for interviewers, accelerates decision-making, and supports scalable hiring processes without sacrificing technical rigor.

However, the platform is not without challenges. Automated scoring may miss nuances in code quality or design, and candidates may feel pressured by time limits and strict constraints. Additionally, maintaining the accuracy of generated questions and test cases requires continuous fine-tuning of AI prompts and evaluation logic.

Overall, the AI coding platform demonstrates how generative AI can elevate technical assessments from static quizzes to **dynamic, adaptive, and recruiter-specific evaluations**, making it a valuable step forward in intelligent recruitment technology.