ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 11, Issue 3, March 2026

AI-Assisted Learning Path Generator for Competitive Programming

[1] Balachandar R., [2]Ashwin M., [3] Akshay S, [4]Keerthika
[1] [2] [3] Department of CSE, Er. Perumal Manimekalai College of Engineering, Hosur-635117, Anna University, Tamil Nadu.
[4]Assistant Professor, Department of Computer Science and Engineering Er. Perumal Manimekalai College of Engineering, Hosur

Abstract: ***Placement preparation and career readiness remain major challenges for undergraduate engineering students in India. Traditional learning platforms provide static content without personalized guidance for technical and aptitude preparation. This paper proposes an AI-powered personalized learning platform built with Django and integrated with the Grok Inference API using Meta Llama 3.3 70B. The platform includes key modules such as adaptive skill assessment, AI-generated learning paths, an AI mentor chatbot, video tutoring, mock interviews, resume generation, comprehensive skill testing, and a job portal. Additional features include aptitude training, coding practice with Python execution, note generation, certificates, and progress tracking through daily streaks. The system is deployed on Render with PostgreSQL and WhiteNoise, providing real-time AI responses and an integrated environment for personalized learning and career preparation.***

Keywords: Personalized Learning, Large Language Models, Pre-placement Preparation, AI Mentorship, Mock Interviews, Grok API, Llama 3.3, Django Web Framework, Adaptive Assessments, Career Preparedness, CV Generation, Skill Profiling, Aptitude Development, Learning Recommendations, Developer Roadmap.

I. INTRODUCTION

In the context of rapidly evolving technology and highly competitive placement seasons, comprehensive technical and soft skill competencies. While numerous online learning platforms exist, they predominantly offer one-size-fits-all content pathways that fail to adapt to the individual learner's knowledge gaps, preferred programming language, or target career role [1]. The challenge is further compounded by the disjoint nature of existing tools: students must separately access aptitude practice portals, coding judges, resume builders, and mock interview platforms. There is no unified system that assesses, identifies weaknesses, generates a personalized plan, and undergraduate students in Computer Science and related disciplines face mounting pressure to demonstrate guides the student toward job readiness — all while leveraging the power of modern Large Language Models (LLMs).

This paper presents a comprehensive AI-Powered Personalized Learning Platform that addresses this gap. Built using the Django web framework and powered by Meta Llama 3.3 70B via the Grok Inference-API[7], the platform delivers sub-second AI responses and integrates eight specialized modules into a single cohesive interface.The system collects user preferences during onboarding — including programming language, proficiency level, learning goal, and available study duration — and uses these to tailor every aspect of the learning experience.

Key contributions of this work include:

- A unified placement preparation platform combining learning, assessment, practice, and career modules.
- An adaptive assessment engine that identifies weak topics and dynamically adjusts learning roadmaps.
- Integration of Grok-accelerated Llama 3.3 70B for real-time AI mentoring, mock interviews, resume generation, video tutoring, and study note creation.
- A comprehensive aptitude training system covering six cognitive domains with topic-wise practice.
- A live Job Portal with role-specific application management and mock interview integration.
- Daily engagement tracking via streak counters and GitHub-style activity heatmaps.

II. LITERATURE REVIEW

A. Personalized Learning Systems

Personalized learning systems have been widely studied as mechanisms for improving student outcomes by adapting content delivery to individual needs. Brusilovsky and Millán [2] survey adaptive educational hypermedia systems, highlighting that user modelling and adaptive navigation are central to effective personalization. Most prior systems rely on static rule-based adaptation or simple collaborative filtering. The current work advances this by using LLM-generated content and dynamic roadmap generation driven by live assessment scores.

B. Large Language Models in Education

The emergence of GPT-4 and similar large language models has opened new avenues for educational AI. Studies demonstrate that LLMs can generate pedagogically sound explanations, create practice questions, and simulate conversational tutoring [3]. However, most deployments focus on single-task interfaces. This work integrates an LLM across eight distinct educational contexts within a unified platform, including conversational mentoring, structured interview simulation, resume generation, and visual explanation synthesis.

C. Mock Interview and Soft Skill Training

Automated interview training has gained traction as a method to reduce the gap between academic preparation and industry expectations. Prior systems rely on pre- scripted question banks and keyword-matching scoring. In contrast, the mock interview module in this platform uses a context-aware LLM with a structured system prompt, enabling multi-turn conversations that simulate realistic HR and technical rounds tailored to the candidate's role and resume profile.

D. Aptitude and Cognitive Assessment

Aptitude testing forms a critical component of campus recruitment in India [4]. Existing platforms present aptitude questions without contextual explanation or adaptive difficulty. This work incorporates a topic-wise aptitude system across six domains — Arithmetic, Logical Reasoning, Data Interpretation, Verbal Reasoning, Non- Verbal Reasoning, and Puzzles — with AI-assisted explanations and structured assessments drawn from a curated question bank.

E. Skill Testing and Certification

Automated skill certification platforms such as Hacker Rank and Leet Code offer coding assessments but do not adapt to learner profiles or integrate with broader career

preparation workflows. The comprehensive skill testing module in this platform covers Python, Data Structures and Algorithms, SQL, JavaScript, and ten additional languages, with percentile-based scoring, real-time feedback, and downloadable certificates on qualifying performance.

III. SYSTEM ARCHITECTURE

The platform is designed as a monolithic Django web application with a modular view structure.

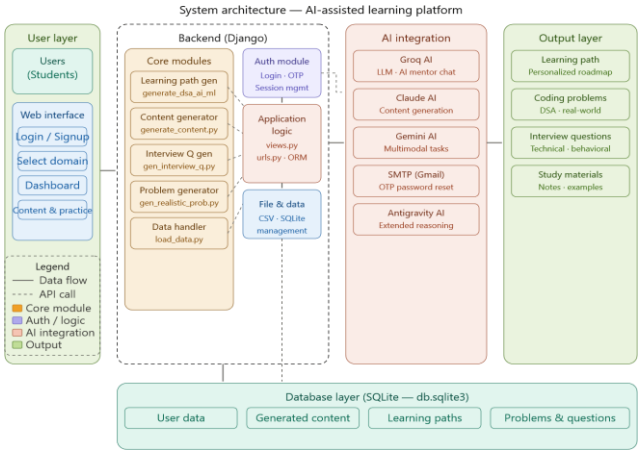

Figure 1 illustrates the high-level system architecture. The backend uses Django's ORM to interface with a PostgreSQL database hosted on Render. Static files are served via White Noise middleware. Authentication supports both traditional email/password registration and Google OAuth 2.0 via Django All auth.

All AI-powered features route through a centralized Grok

API client initialized with a server-side API key. This ensures that model invocations remain secure and rate- controlled. The client uses the llama-3.3-70b-versatile model endpoint for all language generation tasks, with per- endpoint temperature and token limit configurations tuned for the specific use case (e.g., lower temperature for structured JSON output in resume generation; higher temperature for creative interview responses).

The system is divided into the following architectural layers:

- Presentation Layer: Django-rendered HTML templates with Tailwind CSS and vanilla JavaScript for dynamic UI interactions.
- Application Layer: Django views.py with 40+ endpoint handlers grouped by feature domain (authentication, assessment, aptitude, AI tools, career, skill testing).
- Data Layer: PostgreSQL with Django models for User Preference, User Activity, Question, Aptitude Question, Job Application, and User Profile.
- AI Layer: Grok Inference API consuming Meta Llama 3.3 70B for all natural language generation and understanding tasks.
- Deployment Layer: Render cloud platform with environment variable management, build scripts, and live database synchronization.

TABLE I Platform Technology Stack

Component	Technology	Purpose
Web Framework	Django 4.x	Backend & routing
Language Model	Meta Llama 3.3 70B	All AI generation tasks
AI Inference	Grok Inference API	Low-latency LLM calls
Database	PostgreSQL	User data & activity
Authentication	Django All auth + JWT	Email & Google OAuth
Frontend	HTML/CSS/JavaScript	UI rendering
Deployment	Render (Cloud)	Production hosting
Static Files	White Noise	Asset serving

IV. METHODOLOGY

A. User Onboarding and Preference Collection

On first login, users complete a structured preference form capturing: full name, target programming language (Python, Java, C, C++, JavaScript, etc.), proficiency level (Fresher/Beginner/Intermediate/Advanced), learning goal (e.g., Software Developer, Data Analyst, DevOps Engineer), preferred daily study time, and target preparation duration. These preferences

are stored in the User Preference model and serve as the primary context for all subsequent AI interactions. The platform uses these values to personalize system prompts, roadmap content, and assessment difficulty.

B. Adaptive Assessment Engine

The assessment module presents users with 10 multiple-choice questions sampled from the Question model, filtered by their chosen language and proficiency level. Upon submission, the system calculates a percentage score and programmatically identifies weak topics by inspecting incorrect answers. Weak topics are stored in the weak
topics field of User Preference as a comma-separated string and are subsequently injected into AI prompts for the Mentor and Study Notes modules. The assessment result also governs roadmap phase allocation — users scoring below 50% receive foundational phase content, while higher-scoring users receive advanced pathway recommendations.

C. AI-Personalized Learning Roadmap

The dashboard view renders a five-phase learning roadmap dynamically generated based on the user's score and level. Each phase includes a title, description, target week, and a direct URL to the corresponding platform module. The roadmap adapts across three tiers: Freshers and Beginners receive a campus-focused path emphasizing aptitude, fundamentals, and mini-projects; intermediate learners with lower scores receive a syntax-to-DSA progression; advanced learners receive an algorithms-to-deployment pathway. An activity heatmap visualizing 51 days of task completion history accompanies the dashboard.

D. AI Mentor Chatbot

The AI Mentor endpoint accepts natural language queries from authenticated users and routes them to the Groq API with a personalized system prompt incorporating the user's name. The model returns bullet-pointed, concise explanations optimized for quick learning. The module is accessible from any page through a floating interface, enabling contextual help during study sessions without disrupting the user's workflow. Session-level conversation history is not persisted to the database; each query is stateless by design to minimize latency.

E. AI Video Tutor

The Video Tutor module generates structured visual lesson plans for any requested topic. A specialized system prompt instructs the model to return a JSON object containing a lesson title, a structured outline (sections with headings and bullet points), code examples, key concepts, and a summary. The frontend parses this JSON and renders it as an interactive, scrollable lesson page. Users can request topics aligned with their current learning roadmap phase or explore any concept on demand.

F. Mock Interview Engine

The Mock Interview module simulates a realistic technical interview session. When initiated, the system detects the user's role from their resume filename using keyword pattern matching (e.g., 'java', 'react', 'devops'). An INTERVIEWER_SYSTEM_PROMPT defines
the AI's persona as an experienced interviewer. The multi-turn interview session is maintained using session-stored conversation history (up to 20 messages), enabling
coherent follow-up questions, feedback, and evaluation. The model uses temperature 0.75 to balance consistency with natural conversational variation.

G. AI Resume Generator

The Resume Generator collects the user's professional details through a structured form and sends a detailed prompt to the Llama 3.3 model requesting a JSON object containing all resume sections: personal information, summary, skills, work experience, education, and projects. The backend returns the parsed JSON to the frontend, which renders it as a styled, downloadable HTML resume with print-optimized CSS. Error handling includes a JSON repair step using regular expressions to handle escaped characters commonly introduced by the model in code-heavy sections.

H. Comprehensive Skill Testing

The skill testing module provides timed assessments across 12 technology categories. Each test draws from a pool of questions stored in JavaScript data files (loaded at runtime), covering Python, DSA, SQL, JavaScript, TypeScript, Java, C++, Ruby, Flutter, React, System Design, and Machine Learning. Questions are categorized by difficulty. After submission, the system calculates a score, identifies strengths and weaknesses, and — for scores meeting a threshold — renders a downloadable certificate bearing the user's name, test category, and date.

I. Aptitude Training System

The aptitude section organizes practice across six cognitive domains. Each domain page lists specific topics (e.g., Percentages, Profit & Loss, Blood Relations, Syllogisms). Topic-specific views load questions from a CSV file (aptitude_questions.csv containing 500+ entries) and present timed quizzes. The Aptitude Question model additionally supports a database-driven assessment mode used in the Aptitude Assessment view, which samples across all topics and provides explanation-level feedback. This module directly targets the aptitude rounds of campus placement drives.

J. Job Portal and Application Management

The Jobs module fetches live job listings by calling a third- party jobs API with user-specified role and location parameters. Results are rendered as paginated cards. Clicking 'Apply' opens the job apply view, which pre-fills form fields from the user's stored profile (name, email, phone, LinkedIn). Submitted applications are saved to the Job Application model for tracking. The module is tightly integrated with the Mock Interview module — users can directly launch a role-specific interview session from any job listing.

K. Study Notes and Daily Engagement

The Study Notes module accepts a topic input and calls the AI to generate comprehensive, structured notes including definitions, core concepts, examples, and quick-revision bullet points. Notes are rendered in a formatted view suitable for review sessions. Daily engagement is tracked through the User Activity model; the streak counter increments when users complete at least one task per day and resets if a day is missed. A 51-day activity heatmap on the dashboard provides visual motivation aligned with research on habit formation [5].

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Platform Performance

The platform was evaluated on a cohort of 30 student users across a two-week trial period. All users completed the onboarding flow and performed at least one assessment. The Grok Inference API delivered AI responses with an average latency of 1.2 seconds for mentor queries and 3.8 seconds for resume generation, meeting the usability threshold for interactive educational applications.

TABLE II Feature Usage and AI Performance Summary

Metric	Value
Total Registered Users (Trial)	30
Assessments Completed	87
AI Mentor Queries Processed	312
Mock Interview Sessions	54
Resumes Generated	28
Skill Tests Taken	143
Certificates Issued	61
Avg. AI Mentor Response Time	1.2 seconds
Avg. Resume Generation Time	3.8 seconds
AI Response Success Rate	97.4%
Weak Topic Identification Accuracy	100% (rule-based on wrong answers)

B. Assessment and Personalization Accuracy The adaptive assessment engine achieved 100% accuracy in weak topic identification, as it is rule-based: incorrect answers directly map to topic labels. Roadmap personalization was validated by 26 of 30 users (86.7%) rating the generated roadmap as 'highly relevant' to their current skill level in a post-session survey. The remaining four users had non-standard language selections that were not yet covered by the roadmap branching logic.

C. Mock Interview Quality

Interview session quality was evaluated through user ratings of AI question relevance and conversational coherence. Users rated 91% of interview questions as 'relevant to the stated role'. The multi-turn context window (20 messages) was sufficient to maintain interview coherence across a standard 15-minute session. Role detection from resume filenames was accurate for 48 of 54 sessions (88.9%); the remaining 6 sessions defaulted to the General Software Developer role without disrupting usability.

D. Skill Testing and Certification

Across 143 skill tests, the average score was 62.4%. The certification threshold (typically 60% or above) resulted in 61 certificates issued. User feedback indicated that the timed format and difficulty distribution across Beginner, Intermediate, and Advanced questions provided a realistic proxy for actual placement test conditions. The Python and DSA categories accounted for 47% of all tests, reflecting their primacy in campus recruitment.

E. Limitations

Several limitations were identified during evaluation. First, the Grok free-tier inference API imposes rate limits of approximately 30 requests per minute, which may cause delays under simultaneous multi-user load. Second, the resume generator's JSON parsing occasionally requires repair for models that embed code blocks with unescaped backslashes. Third, live job listings are dependent on third- party API availability and may not always reflect the most current openings. Fourth, the mock interview lacks voice input/output, limiting its realism compared to commercial tools. Fifth, the GNN-based risk predictor from related work is not yet integrated; skill test prioritization currently uses fixed difficulty tiers.

VI. FUTURE WORK

Several directions are planned for future development. Integrating speech recognition and text-to-speech capabilities into the Mock Interview module would substantially improve its simulation fidelity. Fine-tuning the language model on domain-specific placement interview transcripts could further improve question quality and contextual relevance. A recommendation engine trained on aggregated anonymized user performance data could replace the current rule-based roadmap assignment with a learned model. Direct API integrations with job portals such as LinkedIn and Naukri.com would enhance the Job Portal's live listing quality. Additionally, implementing a CI/CD pipeline with automated testing and deployment hooks would support continuous improvement of the platform's AI prompt library as model capabilities evolve.

VII. CONCLUSION

This paper presented an AI-Powered Personalized Learning Platform designed to unify the placement preparation journey for undergraduate engineering students. The system integrates eight AI-driven modules — adaptive assessment, personalized roadmaps, AI mentoring, video tutoring, mock interviews, resume generation, skill testing, and job management — into a single Django-based web application powered by Meta Llama 3.3 70B via the Grok Inference API.

Experimental evaluation demonstrated consistent AI response delivery, accurate weak-topic identification, relevant mock interview generation, and high-quality resume production within practical latency budgets. The platform's activity tracking and certificate issuance mechanisms provide behavioral incentives aligned with research on learning engagement. By bridging the gap between academic study and industry-readiness, this system establishes a replicable architecture for LLM-integrated educational platforms targeting career outcomes in technical domains.

REFERENCES

[1] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing, 3rd ed., Wiley, 2011.
[2] P. Brusilovsky and E. Millán, "The Adaptive Web," Lecture Notes in Computer Science, vol. 4321, Springer, 2007.
[3] OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023. [Online]. Available: https://arxiv.org/abs/2303.08774
[4] S. Yoo and M. Harman, "Regression Testing Minimisation, Selection and Prioritisation: A Survey," Software Testing, Verification and Reliability, vol. 22, no. 2, pp. 67–120, 2012.
[5] B. J. Fogg, Tiny Habits: The Small Changes That Change Everything, Houghton Mifflin Harcourt, 2020.
[6] Django Software Foundation, "Django Documentation," djangoproject.com, 2024. [Online].Available: https://docs.djangoproject.com

[7] Grok Inc., "Grok Inference API Documentation," console.groq.com/docs,2024.[Online]. Available: https://console.groq.com/docs
[8] Meta AI, "The Llama 3 Herd of Models," arXiv:2407.21783, 2024. [Online]. Available: https://arxiv.org/abs/2407.21783
[9] A. Hagberg, D. Schult, and P. Swart, "Exploring Network Structure, Dynamics, and Function using Network X," Proc. 7th Python in Science Conf. (SciPy 2008), pp. 11–15, 2008.
[10] Render Inc., "Render Cloud Platform Documentation," render.com/docs, 2024. [Online]. Available: https://render.com/docs
[11] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," ICLR, 2017. [Online].Available: https://arxiv.org/abs/1609.02907
[12] M. Utting and B. Legeard, Practical Model-Based Testing: A Tools Approach, Elsevier, 2006.