AI Career System: Design and Development of An Intelligent Career Guidance Platform Using Machine Learning and Generative AI

Prof. Umesh A. Patil[1], Faizan Naikwadi[2], Adinath Patil[3], Diksha Bansode[4], Tushar Keskar[5]
[1]*Asst. Prof, Department of Computer Science and Engineering, D.Y. Patil Technical Campus Faculty College of Engineering, Talsande, India*
[2,3,4,5]*Department of Computer Science and Engineering, D.Y. Patil Technical Campus Faculty College of Engineering, Talsande, India*

***Abstract*—Students and professionals today face significant challenges when navigating their career trajectories due to the extreme fragmentation of guidance tools. Existing platforms typically offer either scattered college searches, rigid keywordbased job portals, or static personality quizzes, leaving users without a cohesive, actionable pathway. To resolve this, we developed the AI Career System—a centralized, intelligent web application utilizing a 3-Tier MVC architecture (HTML5/Bootstrap 5, Python/Flask, and MySQL). The system integrates five core modules: an AI Resume Builder using dynamic PDF rendering, a localized College Finder, a Real-Time Job Portal, a Machine Learning-powered Career Recommendation engine, and a Generative AI Chatbot. By implementing Scikit-learn for TF-IDF vectorization and Cosine Similarity, the system shifts from basic keyword querying to mathematical intent-matching. Furthermore, integrating the Google Gemini 3 Flash API provides users with low-latency, context-aware career counselling. System testing demonstrated high accuracy in career matching and sub-second rendering speeds for dynamic PDF generation, proving that combining local Machine Learning with external Generative APIs creates a highly effective, unified career guidance ecosystem.**

***Index Terms*—Artificial Intelligence, Machine Learning, Flask, Scikit-learn, Generative AI, Career Guidance, Resume Builder, Natural Language Processing.**

I. INTRODUCTION

The methodology by which students research and prepare for their professional careers has fundamentally shifted. However, the digital tools provided to assist them have largely remained fragmented and static. When students attempt to map their futures, they are forced to use multiple disconnected platforms: one website to format a resume, another to search for colleges, and yet another to hunt for jobs.

Furthermore, traditional career guidance systems rely on rigid, dropdown-based keyword searches that fail to capture a student's true intent or nuanced skills. Realizing this gap, we designed the AI Career System as our final-year capstone project.

The core philosophy of this platform is centralization and intelligent personalization. Instead of forcing the user to adapt to the system, the system uses Natural Language Processing (NLP) to adapt to the user.

We built this as our final-year capstone at D.Y. Patil Technical Campus, Talsande. The core of the system is three separate role-based dashboards—one for admins, one for teachers, one for learners—each built specifically for what that user type needs to do, not just as cosmetic variations of the same interface. The key contributions of this work are:

The key contributions of this work are:

- Centralized Dashboard:

A unified, responsive "Glassmorphism" interface housing resume generation, college searches, and job portals in a single environment.

- Mathematical Intent-Matching:

Utilizing TF-IDF vectorization and Cosine Similarity to recommend careers based on the contextual meaning of a user's input, rather than exact keyword matches.

- Real-Time Generative Counseling:

Integration of the Google Gemini API to provide dynamic, 24/7 conversational advice regarding interview prep and academic routing.

- Automated Deliverables:

An ATS-compliant resume builder that captures user data and dynamically renders downloadable PDF documents using the pdfkit library.

II. LITERATURE REVIEW

To architect a robust solution, we evaluated recent advancements in AI-driven educational platforms and identified specific deficiencies in existing systems.

2.1. Empirical Evidence on Career Counselling Efficacy

Lokam et al. (2025) highlighted the foundational transition from manual counseling to automated platforms, noting the inability of human counselors to scale personalized advice. Dahanke et al. (2022) demonstrated that Machine Learning classifiers could accurately predict career fields based on historical data; however, their proposed systems lacked actionable end-user tools like resume generation. Kulugh et al. (2023) emphasized the need for hyper-personalization, proving that generic career tests fail without Natural Language Processing to decode nuanced user intent. Finally, Westman et al. (2021) identified "tool fatigue," confirming that the high fragmentation of the modern career preparation lifecycle is a primary barrier to student success.

2.2. Review of Existing Platform

Based on this review, Traditional portals have vast amounts of data, but the moment you try to use them as an integrated tool—to say 'help me find a career, build my resume for it, and find a college for it'—you hit a wall. Westman et al. (2021) identified "tool fatigue," a phenomenon confirming that the high fragmentation of the modern career preparation lifecycle is a primary barrier to student success and causes high platform abandonment rates.

2.3. Adaptive and Intelligent System

Kulugh et al. (2023) emphasized the critical need for "hyper-personalization" in career systems. Their studies proved that generic, multiple-choice career tests fail because they do not account for nuanced user intent or unique combinations of skills. This research pushed our architecture toward using Natural Language Processing (Scikit-learn) for career matching, allowing users to speak naturally to the system.

2.4. Flask and MySQL in Educational Applications

We selected Python and Flask for reasons that were partly technical and partly practical. Python is the industry standard for Machine Learning, allowing us to natively integrate Scikit-learn without building complex microservices. Flask gave us a lightweight, non-blocking backend capable of handling API requests and PDF generation simultaneously. MySQL made sense for the data model because our structured content (colleges, users, jobs) requires strict relational integrity.

2.5. Summary of Research Gaps

After this review, we identified the following gaps not addressed by existing platforms:

- No platform provides proper learning-path sequencing for institutional use that is also easy to set up and maintain.
- Progress data exists on most platforms but is buried or useless. A teacher should see in one glance which students are behind and on which module.
- Teacher-student interaction is treated as a communication feature, not a learning feature. It should be tied to content—which module the student is stuck on, not just 'any questions?'
- Admin approval workflows are essentially absent from open platforms, leading to low-quality content polluting course catalogues.
- Content is spread across Drive, WhatsApp, email, and YouTube links. There is no single place for a teacher to organise everything accessibly.

Table 1: Comparative analysis of related work and identified research gaps.

Platform / Study	Key Contribution	Limitation	Gap Addressed by EduPlatform
Lokam et al.	AI-based career guidance framework	Broad categorization; lacks granular counseling	Real-time generative counseling
Dahanke et al.	ML for career prediction	No actionable tools for students	Integrated automated PDF Resume Builder
Kulugh et al.	Hyper-personalized guidance models	Difficult to scale dynamically	NLP intent-matching using TFIDF
Westman et al.	Identified "tool fatigue" in students s	No centralized platform proposed	Unified 5-tool Glassmorphism Dashboard

III. SYSTEM ARCHITECTURE AND DESIGN

About midway through development, we realized our system needed to handle both highly structured relational data (like college names and job listings) and highly unstructured data (like conversational AI responses and NLP vectors). The three-layer MVC split (Bootstrap frontend, Flask backend, MySQL database) made this seamless.

3.1. Overall Architecture

One decision we debated was whether to build separate pages for every tool or a single-page application (SPA) style dashboard. We opted for a centralized dashboard utilizing HTML5, CSS3, and Bootstrap 5 to deliver a modern, responsive Dark Theme Glassmorphism interface. The Application Logic Layer was developed with Python to handle server routing, secure user authentication, and connect frontend user inputs to backend processing services.

3.2. User Authentication and Security Controls

Every protected API endpoint checks the user's active session independently. The UI is not a security boundary—it is a convenience layer. We utilized @login_required decorators in Flask. Anyone can reach the public login page, but access to the AI Chatbot, Resume Builder, and Machine Learning engines is strictly blocked without a cryptographically verified session. This ensures API bandwidth is conserved and user data is protected.

3.3. Data Model

Everything structured is stored in MySQL using highly normalized tables. The users table handles authentication, while colleges and job_listings act as persistent data stores for the research modules.

Table 2: MySQL database schema overview.

Collection	Key Fields	Description
Users	id, full_name, email, password_hash	Securely stores authenticated user credentials. Passwords are cryptographically hashed using pbkdf2: sha256.
Colleges	id, name, location, stream, url	Acts as the persistent storage for the College Finder, housing localized academic data (e.g., Pune, Kolhapur).
Jobs	id, title, company, skills, apply_link	Feeds the Job Portal with active industry roles, skills required, and external ATS routing links.

3.4. System Workflow

The flow through the system in normal operation:

The flow through the system in normal operation is highly linear. A user registers, and their password is cryptographically hashed. Upon logging in, they access the dashboard. They can type an unstructured interest (e.g., "I like coding and fixing hardware") into the Recommendation engine. The Flask backend converts this text using TF-IDF, compares it against the MySQL database using Cosine Similarity, and returns the top 3 matches. The user then opens the Resume Builder, inputs their details, and the system

dynamically generates an ATS-friendly PDF. Finally, they use the Chatbot to ask for interview tips for their newly recommended career.

IV. IMPLEMENTATION AND TECHNOLOGY STACK

4.1. Frontend: HTML5, CSS3, and Bootstrap 5

We utilized Bootstrap 5 for functional, responsive layouts. Tailwind CSS was considered, but Bootstrap's pre-built grid systems allowed for much faster dashboard prototyping. The premium "Glassmorphism" aesthetic was achieved using custom CSS (backdrop-filter: blur(10px)) applied over semi-transparent backgrounds, giving the UI a modern, floatingglass appearance that highly engages student users.

4.2. Backend: Node.js and Express.js *(Adapted to Python and Flask)*

We organized our Python routes by feature—/auth, /dashboard, /api/chat, /generate-resume—rather than building a monolithic file. Each route module has its own middleware stack. For example, the PDF generation pipeline was built using the pdfkit library. Instead of saving files to the physical disk (which causes storage bloat), we utilized io.BytesIO to render the PDF in the server's RAM and stream it directly to the user's browser.

4.3. Database: MySQL

MySQL was chosen for its strict relational integrity. We defined required fields and constraints on every table to catch accidental empty writes. To optimize performance, the colleges table is heavily indexed by stream and location, ensuring that search queries execute in milliseconds even as the database grows.

4.4. Authentication and Security

We used the werkzeug.security library for data protection. Passwords are never stored in plaintext. Session cookies are utilized to track logged-in users. Furthermore, the Gemini API key lives only in a hidden .env file on the backend. This prevents the API key from being exposed to the frontend browser or accidentally uploaded to GitHub.

4.5. Module-Wise Feature Breakdown

During development, we divided responsibility across five core modules:

- Authentication Module: Registration, login, password hashing, and secure session creation.
- Resume Builder Module: Frontend form data capture, Jinja2 template injection, and dynamic PDF rendering.
- Career Recommendation Module: Scikit-learn NLP processing, Cosine Similarity mathematical calculations, and fallback error dictionaries.
- AI Chatbot Module: JSON payload formatting, System Prompts, and Google Gemini 3 Flash API communication.
- Search Modules: Parameterized MySQL querying for the College Finder and Real-Time Job Portal.

4.6. Third-Party Integrations

Table 3: Third-party services and integrations.

Service	Technology	Purpose
Generative AI	Google Gemini 3 Flash API	Called from the backend to provide real-time, context-aware career counselling.
Machine Learning	Scikit-learn (Python)	Performs TF-IDF vectorization and Cosine Similarity for intent-matching.
PDF Rendering	wkhtmltopdf & pdfkit	Converts dynamic HTML resume data into downloadable PDF formats instantly in server memory.

V. EXPERIMENTAL RESULTS AND EVALUATION

The system was evaluated through rigorous functional and integration testing to validate performance, security, and algorithmic accuracy.

5.1. Experimental Setup

We tested the system across multiple devices (desktop monitors, tablets, and mobile smartphones) to ensure Bootstrap 5 responsiveness remained intact. To evaluate the Machine Learning engine, we simulated both highly specific and highly ambiguous edge-case inputs to test the system's fault tolerance.

5.2. Usability Evaluation

Navigation flows, the Glassmorphism aesthetic, and session security were heavily verified. The @login_required decorators were tested by attempting to manually type backend URLs into the browser; the system successfully blocked 100% of unauthenticated API access attempts, redirecting the traffic to the secure login portal.

5.3. Algorithmic Accuracy and Reliability

The ML engine was tested with highly ambiguous inputs. A strict mathematical threshold was implemented within the Python logic. If the Cosine Similarity score dropped below 0.1 (10%), the system recognized the input as invalid and successfully triggered a fallback dictionary. This prevented the application from crashing or returning nonsensical career recommendations.

Table 4: Performance metrics of the AI Career System vs. Traditional Methods.

Metric	AI Career System	Traditional Platforms	Change / Status
Resume Generation Time	~1.2 seconds	Manual formatting (Hours)	Instant Automation
Platform Centralization	5 Tools in 1 Dashboard	Highly Fragmented (3+ Sites)	Eliminated Tool Fatigue
API Response Time	~800 ms | N/A (Static FAQs)	3.1	Real-time interactive
Authentication Security	100% bypass blocked	Varies	Enterprise Standard

5.4. Performance Testing

Testing confirmed that utilizing io.BytesIO for PDF generation kept server memory usage remarkably low. Generating a heavily formatted ATS resume took an average of 1.2 seconds to stream to the client, preventing the disk bloat and slow input/output operations associated with traditional file-saving architectures.

5.5. Qualitative Feedback

Initial review from student testers focused almost entirely on the interface and centralization. Testers noted that housing all tools in one unified dashboard dramatically reduced their cognitive load, directly solving the "tool fatigue" identified during our literature review. The ability to instantly generate a resume immediately after the AI recommended a career was cited as the platform's strongest feature.

VI. DISCUSSION

The successful sub-second generation of PDFs and the high accuracy of the TF-IDF recommendation engine prove that effectively addressing "tool fatigue" requires a centralized, hybridized software approach. The data does not suggest that our platform is the only way to find a job; rather, it proves that providing structured, AI-assisted tools in a single environment produces a much higher user engagement rate than forcing students to perform unstructured web searches. The Google Gemini API worked exceptionally well for dynamic counseling. However, we noted during development that injecting highly specific "System Prompts" in the backend was strictly required to constrain the model, preventing the AI from answering non-career-related questions and keeping the computational focus strictly on professional guidance.

VII. CONCLUSION AND FUTURE WORK

We built the AI Career System because the platforms that currently exist were not executing the functions we thought mattered most: giving students a unified, mathematically accurate path from career discovery to execution. By successfully integrating Flask, MySQL, Scikit-learn, and the Gemini API, the platform delivers a secure, highly responsive ecosystem.

While the current iteration operates flawlessly as a capstone deployment, several avenues exist for commercial scaling. Planned future work includes:

- Advanced Predictive Analytics: Expanding the ML model to scan a user's generated resume against live industry requirements, automatically producing personalized "Skill Gap" reports.
- AI Mock Interview Simulator: Integrating Natural Language Processing voice capabilities to allow the Gemini AI to conduct real-time, audio-based mock interviews tailored to the user's specific resume.
- Automated ATS Integration: Upgrading the Job Portal to include direct API connections with platforms like LinkedIn and Indeed for seamless, one-click application submissions.
- Expanded Global Database: Scaling the localized MySQL database to encompass international college and training center data, moving beyond regional constraints.

ACKNOWLEDGEMENTS

The authors sincerely thank the Head of Department and the faculty of the Department of Computer Science and Engineering at D.Y. Patil Technical Campus, Talsande, for giving their time, providing vital infrastructure support, and continuously pushing us to be rigorous about our software engineering methodologies and project evaluations.

REFERENCES

[1] G. V. Lokam *et al.*, "AI-based career guidance system," *International Journal of Creative Research Thoughts (IJCRT)*, 2025.

[2] A. Dahanke et al., "An intelligent career guidance system using machine learning," International Research Journal of Engineering and Technology (IRJET), vol. 9, no. 3, 2022.

[3] V. E. Kulugh et al., "Artificial intelligence-powered personalised career guidance system," Dutse Journal of Pure and Applied Sciences, 2023.

[4] S. Westman et al., "Artificial intelligence for career guidance – Current requirements and prospects for the future," Journal of Educational Research and Innovation, 2021.

[5] "AI-powered career guidance system," IEEE Conference Paper, IEEE Xplore, 2024.

[6] Google for Developers, "Gemini API Documentation," 2026. [Online]. Available: https://ai.google.dev/docs

[7] Scikit-learn developers, "scikit-learn: Machine Learning in Python," 2026. [Online]. Available: https://scikit-learn.org/stable/

[8] Pallets Projects, "Flask Web Development Framework," 2026. [Online]. Available: https://flask.palletsprojects.com/