ISSN: 2455 - 9172

EIJO Journal of Engineering, Technology And Innovative Research (EIJO–JETIR)

Einstein International Journal Organization (EIJO)

Available Online at: www.eijo.in

Volume – 11, Issue – 2 , March – April – 2026 , Page No. : 01 – 09

Job Pilot AI: A Full-Stack Intelligent Job Application Tracking System

[1]Digvijay Kumar Singh, Department of MCA, IIMT College of Engineering, Greater Noida

[2]Shivani Kumari Singh, Department of MCA, IIMT College of Engineering, Greater Noida

[3]Chiranjeev Singh, Department of MCA, IIMT College of Engineering, Greater Noida

[4]Dr. Naveen Kumar Sharma, Department of MCA, IIMT College of Engineering, Greater Noida

Abstract

The proliferation of digital recruitment platforms has fundamentally transformed job-seeking, compelling candidates to manage numerous concurrent applications across disparate portals and timelines. Existing approaches—including spreadsheets and generic project-management tools—offer no automation, no intelligent communication assistance, and no analytical feedback. This paper presents Job Pilot AI, a cloud-deployed full-stack web application that consolidates job application management into a single intelligent platform. The system integrates a Kanban-style drag-and-drop workflow engine, an SMTP-based automated reminder subsystem, an AI-powered follow-up message generator driven by a large language model (LLM) API, and a real-time analytics dashboard. The frontend employs Next.js 14 and React.js, deployed on Vercel's global CDN. The backend exposes a versioned RESTful API in Node.js and Express.js, with MongoDB Atlas as the data layer. Security is enforced through JWT-based HTTP-only cookie authentication and Google OAuth 2.0. Production benchmarks yielded a Lighthouse performance score of 87, accessibility score of 93, average API latency of 245 ms, and email delivery within 12 seconds. All 38 functional test cases passed, and user acceptance testing returned an overall satisfaction score of 4.5/5.0.

Keywords: Job Application Tracker, Kanban Workflow, Artificial Intelligence, Large Language Model, RESTful API, JWT Authentication, MongoDB Atlas, Next.js, Automated Reminder System, Analytics Dashboard, Full-Stack Web Application

Introduction

The global employment landscape has undergone a structural shift driven by widespread adoption of online recruitment channels. Platforms such as LinkedIn, Naukri, Indeed, and company-specific career portals have dramatically expanded the volume of accessible opportunities for candidates. Paradoxically, this expansion has introduced a secondary challenge: the cognitive and organizational burden of managing multiple, asynchronous, and interdependent application processes concurrently. A single candidate during an active placement season may simultaneously track between thirty and one hundred organizations, each operating on distinct assessment schedules and communication protocols.

Students and early-career professionals face this challenge most acutely. Without a dedicated organizational system, even technically qualified candidates routinely miss critical deadlines, neglect timely follow-up communications, or fail to identify patterns in their application outcomes. Conventional responses—spreadsheets and general-purpose tools such as Trello—are structurally inadequate, lacking domain-specific constructs, AI communication assistance, and automated notifications.

This paper presents JobPilot AI, a purpose-built, cloud-deployed full-stack web application with four principal contributions: (1) a Kanban-based visual pipeline engine supporting real-time drag-and-drop management across seven application stages; (2) an automated reminder subsystem delivering SMTP-based email notifications without manual intervention; (3) an AI-powered follow-up message generator constructing context-aware professional communications using a large language model API; and (4) a real-time analytics dashboard rendering actionable insights on pipeline distribution and conversion metrics.

The remainder of this paper is organized as follows: Section II reviews related literature. Section III defines the problem statement. Section IV describes system architecture. Section V details implementation. Section VI presents experimental results. Section VII concludes with future work.

Literature Review

Personal information management (PIM) has been a sustained area of inquiry in human-computer interaction research. Jones and Dumais[1] established that individuals frequently fail to re-find previously located information—a phenomenon directly applicable to job seekers relying on fragmented stores. Boardman and Sasse [2] identified that tool proliferation exacerbates organizational complexity rather than alleviating it.

The application of Kanban methodology to personal productivity was formalized by Anderson [3], who demonstrated that visualizing work-in-progress through stage-based columns reduces cognitive load and accelerates task throughput. Studies in software engineering confirmed that Kanban boards reduce context-switching penalties and improve prioritization accuracy [4].

The role of timely communication in employment outcomes has been empirically established. Turban and Cable[5] demonstrated that candidate responsiveness during recruitment correlates positively with employer perception of professionalism. Automated reminders improve response timeliness in comparable domains including clinical appointment management [6] and project deadline adherence [7].

Brown et al. [8] introduced GPT-3 and demonstrated its capacity for few-shot generation of contextually appropriate text. Wei et al.[9] further refined instruction-tuned model capabilities for structured prompts. LLM integration has been demonstrated in legal drafting [10], customer service automation [11], and academic writing assistance [12], establishing precedent for the application explored here.

Despite the breadth of research across individual contributing domains, no prior work has synthesized Kanban workflow management, AI-driven communication generation, automated SMTP scheduling, and real-time analytics into a unified career management platform. Job Pilot AI occupies this identified gap.

Problem Statement and Motivation

The problem domain is characterized by four primary failure modes in current job-tracking practices:

A. Information Fragmentation and Disorganization

Candidates managing simultaneous applications across multiple platforms lack a centralized repository for application state. Critical data including submitted resume versions, application URLs, contact person names, and stage-specific notes are distributed across email threads, bookmark folders, and informal notes. This fragmentation results in duplicate applications, missed confirmations, and inability to reconstruct application history.

B. Absence of Proactive Notification Infrastructure

Professional recruitment etiquette requires candidates to follow up with recruiters at defined intervals. Without an automated reminder system, candidates either neglect follow-ups or execute them at inappropriate intervals, reducing their probability of advancing through the selection process. No existing general-purpose tool provides domain-specific, application-aware scheduling.

C. Deficit of Analytical Feedback

Manual tracking methods provide no mechanism for extracting patterns from application data. A candidate cannot quantify application-to-interview conversion rates, identify sectors generating the highest response rates, or determine average pipeline stage duration—preventing evidence-based strategic adjustments.

D. Communication Quality Variability

The professional quality of follow-up emails varies substantially across candidates and correlates with writing proficiency rather than technical competence. A system capable of generating polished, contextually appropriate communications from structured application data would substantially equalize this dimension of the hiring process.

System Architecture and Design

Job Pilot AI is architected as a classical three-tier web application comprising a presentation layer, an application logic layer, and a data persistence layer. The system additionally integrates three external service components: a Google OAuth 2.0 authorization server, an SMTP email relay, and an AI language model API endpoint.

A. Presentation Layer

The frontend is constructed using Next.js 14, leveraging the App Router architecture stabilized in version 14. Server-Side Rendering (SSR) minimizes time-to-first-contentful-paint. React.js 18 manages component-level state through useState and useReducer hooks, while Axios handles API communication. Tailwind CSS governs responsive styling supporting viewport widths from 320px to 1920px. The presentation layer is deployed on Vercel's global CDN.

B. Application Logic Layer

The backend is implemented as a RESTful API server using Node.js 18 LTS and Express.js 4. All routes are versioned under /api/v1/. The application layer is organized into discrete router modules: authentication, job management, reminders, analytics, AI message generation, and user settings. Cross-cutting concerns including JWT authentication verification, CORS policy enforcement, and structured error handling are implemented as composa2ble Express middleware. The backend is containerized and deployed on Render's cloud infrastructure.

C. Data Persistence Layer

MongoDB Atlas provides cloud-hosted document storage. The document-oriented model accommodates the variable and semi-structured nature of job application records. Mongoose 7 serves as the Object Data Mapper, enforcing schema validation, enabling relationship modeling through ObjectId references, and providing query optimization through indexed fields.

D. Database Schema Design

Three primary collections constitute the data model as shown in Table I.

Table 1: Mongodb Collection Schema Overview

Collection	Key Fields	Description
Users	_id, email, username, password, auth Provider, google Id, theme	User identity, authentication provider, hashed credentials, and UI preferences.
Jobs	_id, user Id, company, role, status, salary, link, notes, applied Date	Full application records. Status field drives Kanban column placement.
Reminders	_id, user Id, job Id, message, remind At, sent	Scheduled notifications. The 'sent' boolean prevents duplicate dispatch.

E. Security Architecture

Authentication employs a dual-mechanism design. Primary authentication uses JWT tokens signed with HS256, transmitted as HTTP-only, Same Site=Strict cookies. This configuration prevents JavaScript-level access, mitigating XSS vectors, while SameSite reduces CSRF exposure. Token expiry is set to 24 hours. Secondary authentication supports Google OAuth 2.0 via Passport.js. User passwords are hashed using bcrypt with a salt factor of 12.

Implementation Methodology

A. Technology Stack

The complete technology stack is summarized in Table II.

Table 2: Technology Stack Summary

Layer	Technology	Purpose
Frontend	Next.js 14 / React.js 18	SSR, routing, component architecture
Frontend	Tailwind CSS 3.x	Responsive utility-first styling
Frontend	Axios	HTTP client for API communication
Backend	Node.js 18 LTS / Express.js 4	RESTful API server and middleware
Database	MongoDB Atlas / Mongoose 7	Cloud NoSQL document storage and ODM
Auth	JWT / bcrypt / Google OAuth 2.0	Stateless auth, password hashing, social login
Email	Nodemailer 6.x (SMTP)	Automated email notification delivery
AI	OpenAI / LLM API	Context-aware professional message generation
Deploy FE	Vercel (Global CDN)	CI/CD, global edge delivery, zero-config
Deploy BE	Render (Container)	Backend containerized cloud hosting

B. Kanban Workflow Engine

The Kanban board constitutes the primary user interface for application status management. Seven pipeline stages are defined: Applied, Assessment, Interview, Offer, Rejected, Withdrawn, and Joined. Drag-and-drop interaction is implemented using pointer event listeners. Upon card release onto a destination column, a PATCH request is dispatched

to /api/v1/jobs/:id/status, updating the status field in the database. React state is updated optimistically, re-rendering the board without a full page reload. Pessimistic rollback handles network failure scenarios.

C. Automated Reminder and Notification Subsystem

The reminder subsystem operates as a scheduled background process on the Node.js backend. A cron-style job running at one-minute intervals queries the Reminders collection for documents where the remindAt field has elapsed and the sent field remains false. For each qualifying record, Nodemailer dispatches an SMTP email to the user's registered address. Following successful dispatch, the sent field is atomically set to true via findByIdAndUpdate, ensuring idempotent behavior.

D. AI Follow-Up Message Generator

The AI message generation feature accepts four structured inputs: company name, applied role title, current application stage, and optional personal notes. The backend constructs a structured prompt specifying formal business English tone, 150–250-word length, and content constraints. This prompt is transmitted to an LLM API via authenticated HTTPS. The model's response is parsed, sanitized, and returned to an editable text field. Five distinct message categories are supported: status enquiry, post-interview thank-you, offer acceptance, offer declination, and reconnection after rejection.

E. Analytics Dashboard

The analytics module employs MongoDB's aggregation pipeline to compute summary statistics from the authenticated user's job records. The primary aggregation groups documents by status field and counts occurrences, producing the pipeline distribution breakdown. A secondary aggregation groups records by creation date truncated to weekly intervals, computing application volume per week over the preceding four weeks. Chart.js renders doughnut and bar chart visualizations within React components.

F. Restful API Design

The API exposes the primary endpoint groups listed in Table III.

Table 3: Primary Rest API Endpoints

Method	Endpoint	Description
POST	/api/v1/auth/register	Register new user with bcrypt-hashed credentials
POST	/api/v1/auth/login	Authenticate user, issue JWT HTTP-only cookie
GET	/api/v1/auth/google	Initiate Google OAuth 2.0 authorization flow
GET	/api/v1/jobs	Retrieve all job records for authenticated user
POST	/api/v1/jobs	Create new job application record
PATCH	/api/v1/jobs/:id	Update job record fields (status, notes, etc.)
DELETE	/api/v1/jobs/:id	Remove job application record
POST	/api/v1/reminders	Schedule a new email notification
POST	/api/v1/ai/generate-message	Generate AI follow-up message from job context
GET	/api/v1/analytics/summary	Retrieve aggregated pipeline metrics

Results and Discussion

A. Performance Benchmarks

Performance evaluation was conducted on the production deployment using Google Lighthouse automated auditing and manual API benchmarking via Postman over five independent runs with median values reported. API latency measurements were collected over 200 requests per endpoint with the 95th-percentile reported alongside the mean. Results are summarized in Table IV.

Table 4: Production Performance Benchmark Results

Metric	Target	Achieved	Status
Dashboard Load	< 3.0 s	1.8 s	PASS
Lighthouse Perf.	> 80	87	PASS
Lighthouse Acc.	> 90	93	PASS
Lighthouse SEO	> 80	91	PASS
GET /jobs (mean)	< 500 ms	210 ms	PASS
POST /jobs (mean)	< 500 ms	280 ms	PASS
GET /jobs (p95)	< 800 ms	412 ms	PASS
JWT Auth Overhead	< 50 ms	18 ms	PASS
Email Notification	< 60 s	12 s avg	PASS
AI Message Latency	< 8 s	3.2 s avg	PASS
Mobile Responsive	Functional	All features	PASS

B. Functional Testing Results

Systematic testing was conducted across four levels: unit testing of individual API endpoints using Postman, integration testing of cross-module workflows, system-level testing on the production deployment, and user acceptance testing with eight MCA students. All 38 defined test cases were executed as shown in Table V.

Table 5: Functional Test Case Results By Module

Module	Total	Passed	Failed	Rate
Authentication	7	7	0	100%
Job Management	6	6	0	100%
Reminder System	4	4	0	100%
Analytics	3	3	0	100%
AI Message Gen.	5	5	0	100%
Kanban Workflow	5	5	0	100%

Module	Total	Passed	Failed	Rate
Settings & Theme	4	4	0	100%
Security (JWT)	4	4	0	100%
Total	38	38	0	100%

C. User Acceptance Testing

Eight MCA students participated in structured user acceptance testing over a seven-day period using the platform for their actual ongoing job applications. Post-usage structured questionnaires assessed usability on a five-point Likert scale. Results are presented in Table VI.

Table 6: User Acceptance Testing – Satisfaction Scores

Usability Criterion	Mean (1-5)	Std Dev	Rating
Ease of onboarding	4.6	0.52	Excellent
Kanban intuitiveness	4.5	0.53	Excellent
AI message quality	4.3	0.71	Very Good
Analytics clarity	4.4	0.74	Very Good
Reminder reliability	4.8	0.46	Excellent
Overall satisfaction	4.5	0.53	Excellent

D. Comparative Analysis with Existing Tools

Table VII presents a feature comparison of JobPilot AI against the primary competing approaches employed by job seekers in the target demographic.

Table 7: Feature Comparison with Competing Solutions

Feature	Job Pilot AI	Spreadsheet	Trello	Huntr	Teal
Kanban Pipeline	Yes	No	Partial	Yes	Yes
Auto Email Reminders	Yes	No	No	Partial	Partial
AI Message Generator	Yes	No	No	No	No
Analytics Dashboard	Yes	Manual	Manual	Basic	Basic
Google OAuth	Yes	N/A	Yes	Yes	Yes
Mobile Responsive	Yes	Poor	Partial	Yes	Yes
Zero Cost (Free)	Yes	Yes	Freemium	Freemium	Freemium
Open-Source	Yes	N/A	No	No	No

Discussion

The performance benchmarks confirm that Job Pilot AI meets or exceeds all defined non-functional requirements. The Lighthouse performance score of 87 reflects the optimization benefits of Next.js SSR combined with Vercel's edge CDN. API response times well below the 500 ms threshold demonstrate efficiency of MongoDB Atlas indexed document retrieval. The 12-second average email delivery latency is acceptable for reminder notifications typically set hours or days in advance.

User acceptance testing indicates strong adoption metrics across all evaluated dimensions. The highest satisfaction scores were recorded for reminder reliability (4.8/5.0) and onboarding ease (4.6/5.0). AI-generated message quality received 4.3/5.0; qualitative feedback indicated that users valued contextual relevance but occasionally found the formal tone somewhat rigid for informal recruiter relationships—informing planned tone-selection controls in future iterations.

The comparative analysis in Table VII demonstrates a clear differentiation advantage over all surveyed alternatives, with JobPilot AI being the only identified system providing all four core capabilities in a single, zero-cost, open-source platform. The primary limitation relative to commercial offerings is the absence of direct job portal integration, which represents the most frequently cited improvement request from UAT participants.

Conclusion and Future Work

This paper presented Job Pilot AI, a full-stack, cloud-deployed, AI-integrated web application addressing the organizational, communicative, and analytical deficiencies of contemporary job application management. The system successfully synthesizes Kanban workflow methodology, automated SMTP notification scheduling, large language model-driven professional communication generation, and real-time aggregation analytics into a unified, responsive platform accessible at zero cost.

Production performance benchmarking validated all defined non-functional requirements: a Lighthouse performance score of 87, API response latencies averaging 245 ms, email notifications delivered within 12 seconds, and 100% pass rates across 38 functional test cases. User acceptance testing yielded an overall satisfaction mean of 4.5/5.0, with feature comparison confirming superiority over all identified zero-cost alternatives.

Future development will prioritize four high-impact enhancements: (1) an AI-powered resume analyzer scoring uploaded files against ATS compatibility criteria; (2) a browser extension enabling one-click job application logging directly from recruitment portal pages; (3) bidirectional Google Calendar synchronization for interview scheduling; and (4) native iOS and Android applications providing push notification support and offline data access. Longer-term research includes collaborative filtering for personalized company and role recommendations based on aggregated, anonymized application outcome data.

References

1. W. Jones and S. Dumais, "The once and future Web: Accessing and managing information across locations and devices," in Proc. ASIST Annual Meeting, 2004, pp. 20–28.
2. R. Boardman and M. A. Sasse, "Stuff goes into the computer and doesn't come out: A cross-tool study of personal information management," in Proc. ACM CHI, 2004, pp. 583–590.
3. D. J. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

4. C. Ikonen et al., "On the impact of Kanban on software project management: An empirical study," in Proc. EUROMICRO SEAA, 2011, pp. 305–313.
5. D. B. Turban and D. M. Cable, "Firm reputation and applicant pool characteristics," J. Organ. Behav., vol. 24, no. 6, pp. 733–751, 2003.
6. S. Gurol, S. Teke, and F. Akbulut, "Effect of SMS reminders on appointment attendance in an outpatient clinic," Health Informatics J., vol. 26, no. 2, pp. 1–11, 2020.
7. J. Luthria and F. Rabhi, "Service-oriented computing in practice," J. Theor. Appl. Electron. Commer. Res., vol. 4, no. 1, pp. 39–56, 2009.
8. T. Brown et al., "Language models are few-shot learners," in Proc. NeurIPS, 2020, pp. 1877–1901.
9. J. Wei et al., "Finetuned language models are zero-shot learners," in Proc. ICLR, 2022.
10. N. Bommarito and D. M. Katz, "Legal knowledge engineering in the era of large language models," arXiv:2212.09101, 2022.
11. R. Shum, D. Mou, and M. Zhou, "From ELIZA to XiaoIce: Challenges and opportunities with social chatbots," Front. Inf. Technol. Electron. Eng., vol. 19, no. 1, pp. 10–26, 2018.
12. I. Lee and Y. J. Choi, "ChatGPT and its implications for education and research," J. Comput. Inf. Syst., vol. 63, no. 4, pp. 1–10, 2023.
13. R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. of California, Irvine, 2000.
14. Vercel Inc., Next.js 14 Documentation. [Online]. Available: https://nextjs.org/docs (accessed Apr. 2025).
15. Open JS Foundation, Node.js 18 LTS Documentation. [Online]. Available: https://nodejs.org/en/docs (accessed Apr. 2025).
16. MongoDB Inc., MongoDB Atlas Documentation. [Online]. Available: https://www.mongodb.com/docs/atlas/ (accessed Apr. 2025).