IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

SmartPath AI: An Integrated Full-Stack Framework for AI-Driven Government Exam Discovery, Eligibility Matching, and Adaptive Preparation

Dr. Indumathi S K[1], Poorvisha Vasanth[2], Nikitha S Shetty[3]

Associate Professor, Department of MCA, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka, India[1]

Student, Department of MCA, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka, India[2]

Student, Department of MCA, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka, India[3]

Abstract: Aspirants preparing for government recruitment examinations in India face a uniquely fragmented information landscape: notifications for national examinations such as the Union Public Service Commission (UPSC) Civil Services Examination, the Staff Selection Commission (SSC) Combined Graduate Level examination, Institute of Banking Personnel Selection (IBPS) recruitment, and Indian Railways recruitment, alongside state-level examinations conducted by the Karnataka Public Service Commission (KPSC), are published independently across dozens of authority-specific websites, each with its own format, eligibility clause, and application timeline. This paper presents SmartPath AI, a full-stack web platform that unifies exam discovery, eligibility screening, and personalized preparation for both national and Karnataka state government examinations behind a single interface. The system combines a scheduled web-scraping layer that continuously harvests notifications from official examination-authority websites with a human-moderated approval queue, a rule-based eligibility engine that evaluates a candidate's age, state of domicile, and academic qualification against each examination's criteria, and a course-to-career "wizard" that maps a candidate's educational background onto a ranked list of matching examinations and government job profiles. Layered on top of this discovery-and-matching foundation is a generative-AI content engine, built on the Anthropic Claude API, that produces exam-specific mock tests, topic-wise quizzes, structured study notes, and an adaptively recalibrated study timetable, together with a document-intelligence pipeline that extracts structured fields from scanned government notification PDFs and academic marksheets. A conversational AI study coach and a gamification layer comprising streaks, badges, a leaderboard, and peer study groups are added to sustain daily engagement. The platform is implemented with a React and Vite single-page frontend, a Python FastAPI backend backed by SQLAlchemy and SQLite, a lightweight Node.js/Express static-and-proxy layer, JSON-Web-Token-based authentication, and Razorpay-based subscription billing, and is deployable as an installable, offline-capable Progressive Web Application. Functional evaluation of the deployed system indicates that combining scraped-and-moderated official data, transparent rule-based eligibility logic, and generative AI content substantially reduces the manual effort an aspirant would otherwise spend locating, verifying, and preparing for relevant examinations.

Keywords: Government Exam Preparation, Eligibility Screening, Rule-Based Systems, Web Scraping, Generative Artificial Intelligence, Large Language Models, Conversational AI, Gamification, FastAPI, React.

I. INTRODUCTION

Government recruitment examinations remain one of the most sought-after career pathways in India, with individual notifications from bodies such as the Staff Selection Commission, the Institute of Banking Personnel Selection, Indian Railways, the Union Public Service Commission, and state public service commissions routinely attracting applicant pools that number in the hundreds of thousands. Each conducting authority publishes its own notification PDF, maintains its own website, and defines its own age limits, category-based relaxations, educational-qualification clauses, and application deadlines, so a single aspirant tracking multiple examinations across national and state levels must manually monitor dozens of unrelated sources, interpret dense eligibility language, and separately source study material for each examination's syllabus. Karnataka-domiciled aspirants face an additional layer of complexity, since they must track both national examinations and Karnataka-specific recruitment conducted by the Karnataka Public Service Commission and the Karnataka State Police, alongside eligibility tests such as KARTET and KSET.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

This paper presents SmartPath AI, a full-stack government-exam-intelligence platform that unifies notification discovery, eligibility screening, and AI-generated exam preparation behind a single web application targeted at this dual national-and-Karnataka-state examination landscape. The system is designed as a layered application comprising a Python/FastAPI backend that exposes modular REST APIs, a React-based single-page frontend, a background scraping-and-scheduling layer that continuously ingests examination notifications, and a generative-AI layer that produces exam-specific practice content on demand. The central contribution of this work is the integration of a moderated web-scraping pipeline, a rule-based eligibility-matching engine, and a large-language-model-driven content-generation and document-intelligence layer within one cohesive platform, so that an aspirant can move from discovering a relevant notification to confirming their eligibility to practising exam-specific questions without leaving a single application.

II. LITERATURE REVIEW

Research relevant to this work spans several streams that are more commonly treated separately. Rule-based expert systems have long been used to automate eligibility and advising decisions that would otherwise require a human expert to manually cross-check a candidate's profile against a set of criteria; Engin et al. demonstrate that a rule-based expert system can reliably automate course-advising and scholarship-eligibility decisions for university students by encoding institutional policy as an explicit, auditable rule base rather than as opaque statistical weights [4]. Web scraping has become a well-established technique for aggregating information published across many independent, loosely structured sources; Bhatt et al. survey web-scraping techniques and their application across business-intelligence and information-aggregation use cases, noting that HTML-parsing libraries such as BeautifulSoup remain a practical choice for structured extraction from heterogeneous websites [2].

On the content-generation side, large language models have recently been shown to be an effective aid to manual item-writing in assessment design; Biancini et al. compare several large language models for automatic multiple-choice-question generation and report that carefully prompted models can produce informative, curriculum-aligned questions with substantially less educator effort than fully manual authoring [3]. Separately, a systematic literature review by Khaldi et al. of gamification in higher-education e-learning platforms finds that points, badges, and leaderboards remain the most consistently applied gamification elements and are associated with improved learner engagement when implemented alongside, rather than as a replacement for, substantive learning content [1]. Token-based authentication, specifically the JSON Web Token standard defined in IETF RFC 7519, provides a stateless mechanism for securing REST APIs consumed by single-page applications and is the authentication approach adopted throughout SmartPath AI's backend [5].

Research Stream	Representative Focus	Relevance to This Study
Rule-based expert systems	Eligibility rules; age, qualification and domicile checks	Forms the eligibility engine that screens users against each examination's criteria.
Web scraping and information aggregation	HTML parsing; scheduled crawling of heterogeneous sources	Forms the notification-ingestion pipeline that feeds the exam and job catalog.
Generative AI for assessment content	LLM-based multiple-choice-question generation	Underpins the mock-test, quiz and study-notes generation engine.
Gamification in e-learning	Points, badges, leaderboards, streaks	Motivates the streak, badge and leaderboard layer used to sustain engagement.
Token-based web authentication	JWT, stateless REST API security	Motivates the JWT-based authentication layer securing all API routes.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

III. RESEARCH GAP

Existing government-exam-preparation platforms typically specialize in a single examination category — a UPSC-focused current-affairs application, a banking-focused numerical-reasoning application, or a state-specific notification aggregator — and rarely combine notification aggregation, eligibility screening, and generative practice-content creation within a single system. Where notification-aggregation services do exist, they typically republish scraped listings without a moderation step, creating a risk of surfacing stale, duplicate, or incorrectly parsed notifications to the end user; and where eligibility calculators exist, they are usually presented as isolated, single-purpose tools rather than as an integrated part of a broader preparation workflow. Few open-source or academic systems combine a scheduled, multi-source scraping pipeline with a human-in-the-loop approval queue, a rule-based eligibility engine that accounts for category-based age relaxation, and a large-language-model content-generation layer that produces exam-specific mock tests, study notes, and an adaptively recalibrated study timetable, all behind a single authenticated web application.

The primary novelty of this work is therefore not any single algorithmic component but the integration of a moderated scraping-and-notification pipeline, a rule-based eligibility and career-matching engine, and a generative-AI practice-content and document-intelligence layer into one cohesive full-stack platform that simultaneously serves national-level and Karnataka state-level government examinations.

IV. RESEARCH OBJECTIVES

The primary objective of this work is to design and implement an integrated full-stack platform capable of aggregating government-examination notifications, screening a candidate's eligibility against each notification's criteria, and generating personalized, exam-specific preparation content through a combination of rule-based logic and generative AI. Secondary objectives include: implementing a scheduled, multi-source scraping pipeline covering national examination authorities (UPSC, SSC, IBPS, Indian Railways) and Karnataka state authorities (KPSC); introducing a human moderation queue that gates scraped notifications before they are surfaced to end users; developing a rule-based eligibility engine that evaluates age, category-based age relaxation, state of domicile, and academic qualification; building a course-to-career wizard that maps an aspirant's educational background onto a ranked list of matching examinations and government job roles; integrating a generative-AI engine for mock tests, topic-wise quizzes, structured study notes, and adaptive timetable recalibration; building a document-intelligence pipeline that extracts structured fields from government notification PDFs and academic marksheets; layering a conversational AI study coach and a gamification system on top of the preparation workflow; and deploying the complete system as an authenticated, installable, offline-capable web application.

The study addresses the following research questions: RQ1: Can a scheduled web-scraping pipeline, gated by a human moderation queue, reliably aggregate examination notifications from heterogeneous authority websites into a single structured catalog? RQ2: Can a rule-based eligibility engine that accounts for age, category-based relaxation, domicile, and qualification correctly and transparently screen a candidate against a heterogeneous set of examination criteria? RQ3: Can a large-language-model content-generation layer produce exam-specific mock tests, quizzes, and study notes appropriately tailored to a given examination's syllabus and difficulty level? RQ4: How does combining scraped notification data, rule-based eligibility screening, and generative practice content within one workflow affect the overall usability of exam preparation compared with using separate, single-purpose tools? RQ5: What architectural design choices allow such a system, spanning a scraping layer, a relational data model, a rule engine, and an LLM-backed generation layer, to be deployed as a scalable, RESTful, installable web application?

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

V. PROPOSED SYSTEM ARCHITECTURE

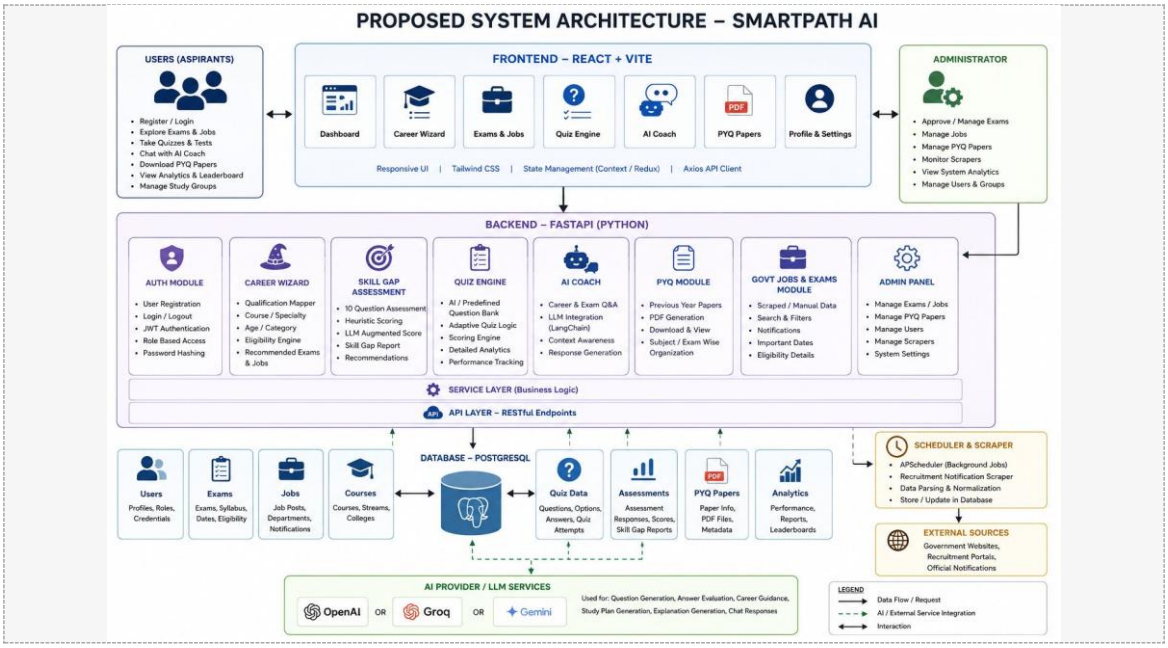

Fig. 1. System architecture diagram of the SmartPath AI platform, showing the React/Vite frontend, the Node/Express static-and-proxy layer, the FastAPI backend, the SQLite/PostgreSQL data tier, the APScheduler-driven scrapers, and the Anthropic Claude API integration.

SmartPath AI follows a layered, client-server architecture spanning four logical tiers. The presentation tier is a React and Vite single-page application, built with Zustand for client-side state management and TanStack React Query for server-state caching, that renders the exam catalog, personalized dashboard, mock-test interface, career wizard, chatbot widget, and gamification views, and that registers a service worker to enable offline caching and installable Progressive-Web-App behaviour. A lightweight Node.js and Express layer serves the built frontend bundle, exposes a small set of static informational pages, and, where configured, reverse-proxies API traffic to the primary backend using http-proxy-middleware, allowing the static portal and the primary API to be started together with a single combined script during development. The application tier is a Python FastAPI service exposing eleven modular routers — authentication, examinations, dashboard, features, admin, community, payments, chat, courses, wizard, and quiz — each delegating business logic to a corresponding service or AI module rather than embedding it directly in the route handler.

The data tier persists twenty-four SQLAlchemy models, covering users, examinations, government jobs, scraped notifications, eligibility rules, mock-test and quiz attempts, study timetables, badges, leaderboard entries, study groups, and scraper run logs, over SQLite in local development with a PostgreSQL-compatible connection string for production deployment. A background scheduling tier, built on APScheduler, runs the notification scrapers for each examination authority on a fixed six-hour interval and records the outcome of every run to a dedicated audit log table. All inter-tier communication is authenticated using stateless JSON Web Tokens signed with a server-held secret key, and all AI-driven features route through a single Anthropic Claude client used across the mock-test generator, study-notes generator, resume/marksheet parser, timetable recalibrator, and chat coach.

VI. DATA MODEL AND EXAMINATION CATALOG

The SmartPath AI data model is organized around two parallel catalog entities: Exam, representing formally structured national and Karnataka-state examinations with syllabus and pattern metadata, and GovernmentJob, representing direct-entry government positions outside the examination-and-syllabus structure, together with an ExamNotification table that stores raw, scraper-sourced listings pending moderation. Each Exam record stores its conducting authority, minimum and maximum age, qualification requirement, application and examination dates, salary range, difficulty rating, selection process, and links to the notification PDF and official portal, while a companion EligibilityRule table stores a structured JSON rule — minimum and maximum age, permitted domicile state, and allowed or required qualification keywords — that the eligibility engine evaluates against a candidate's profile. The quiz-generation subsystem separately maintains a compact catalog of nine Karnataka-focused examination categories — KAS, FDA, SDA, PDO, PSI, PC, CTI, KSET,

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

and TET — each mapped to a set of realistic previous-year-question-style source-paper labels used to make generated questions feel grounded in an examination's actual paper structure.

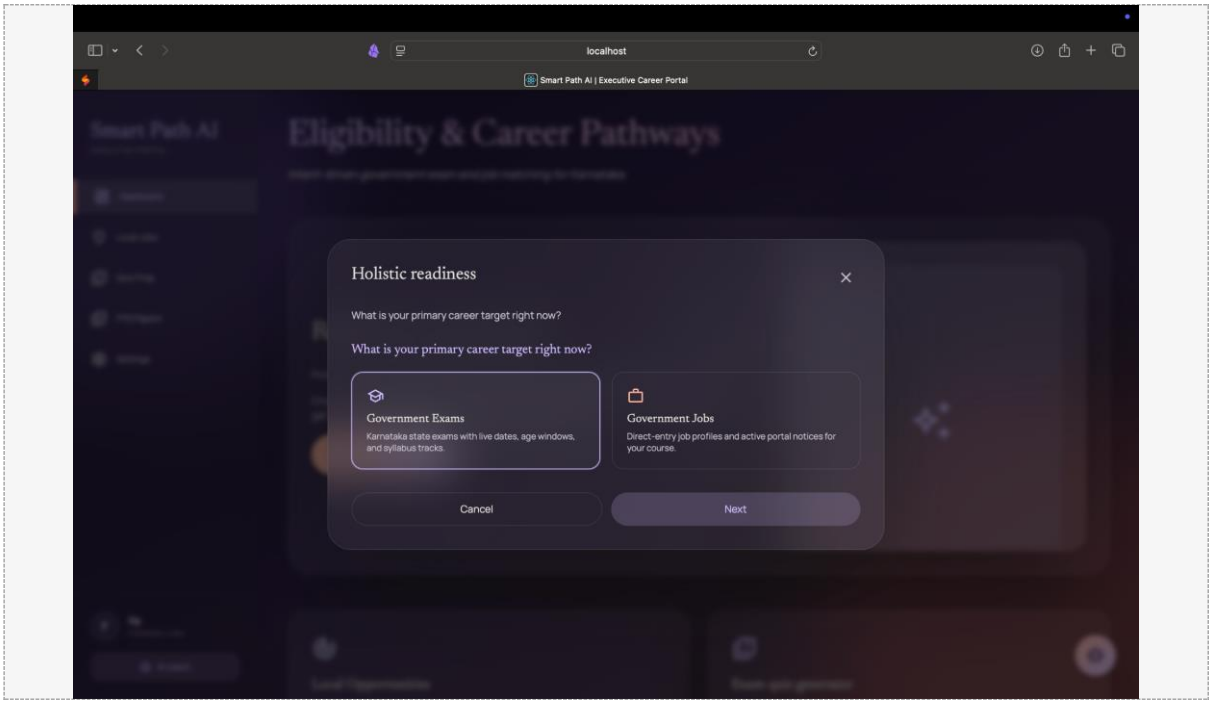

Fig. 2. Screenshot of the examination and government-job catalog / browse page.

User-facing engagement is tracked through MockTestAttempt, QuizAttempt, UserActivity, UserBadge, and LeaderboardEntry tables, while StudyTimetable stores each user's AI-generated and subsequently recalibrated weekly study plan as a JSON payload. Coverage of the seeded catalog spans five scraper-fed national and state examination authorities, fourteen syllabus topic areas with associated study tips ranging from Polity and Kannada to Reasoning and Environment & Ecology, and two subscription tiers, free and pro, that gate the monthly quota available for mock tests, AI chat messages, study-note generation, and study timetables.

Attribute	Value
Examination authorities scraped	5 — UPSC, SSC, IBPS, Indian Railways, KPSC
Karnataka-focused quiz categories	9 — KAS, FDA, SDA, PDO, PSI, PC, CTI, KSET, TET
Syllabus topic areas with study tips	14 (Polity, History, Geography, Economy, Reasoning, etc.)
Core data entities	users, exams, government_jobs, exam_notifications, eligibility_rules, mock_test_attempts, quiz_attempts, study_timetables
Subscription tiers	2 — free, pro
Scraper run interval	Every 6 hours (APScheduler)

VII. DATA ACQUISITION, PREPROCESSING AND FEATURE ENGINEERING

Each scraper subclasses a common BaseScraper abstract class that wraps an asynchronous HTTP client configured with a descriptive User-Agent header and automatic redirect following, and implements a scrape() coroutine that parses a conducting authority's homepage with BeautifulSoup, selecting anchor elements that link to PDF documents or contain notice- and circular-related keywords, and normalizing each match into a common ExamNotification record comprising exam name, authority, PDF URL, and post name. The scheduler layer runs each of the five authority-specific scrapers

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

every six hours and upserts results into the exam_notifications table keyed on the combination of examination name and authority, so that repeated scraper runs update rather than duplicate an existing listing.

Where a scraped listing links to an actual notification PDF, a separate document-intelligence pipeline downloads the file, extracts text from its first five pages using PyMuPDF, and passes the extracted text to the Claude API with an instruction to return a structured JSON object containing the examination name, post name, vacancy count, age limits, qualification requirement, key dates, salary range, and selection process, effectively using the language model as a field-extraction layer over unstructured government-notification text rather than relying on brittle, authority-specific regular expressions. Parsed results are cached in Redis, keyed by an MD5 hash of the source PDF URL, to avoid re-parsing an unchanged document on subsequent requests. The same document-intelligence approach is reused for academic marksheets: a user-uploaded marksheet image is base64-encoded and sent to Claude's multimodal endpoint with an instruction to extract degree name, specialization, university, year of passing, and percentage or CGPA as structured JSON, populating the profile fields that the eligibility engine subsequently evaluates.

For AI-generated practice content, each exam-specific generator constructs a purpose-built prompt rather than a single generic instruction: the mock-test generator selects an exam-style description from a lookup table — for example, an SSC CGL-style prompt requests straightforward, fact-based questions, while a UPSC Prelims-style prompt requests nuanced, conceptual, India-focused questions — before requesting a fixed-schema JSON response from the model. The general quiz generator additionally supports a configurable AI provider, Groq or OpenAI, selected via environment variables, and falls back to a curated static question bank whenever no AI provider is configured or the model's response fails schema validation, ensuring that the quiz feature degrades gracefully rather than failing outright when an external AI dependency is unavailable.

VIII. SYSTEM IMPLEMENTATION AND FUNCTIONAL EVALUATION

The end-to-end platform was exercised through the deployed application, from user registration and authentication, through notification browsing and eligibility screening, to mock-test generation and dashboard review, as illustrated in the screenshots referenced below. Registration and login are handled through a dedicated authentication router that hashes passwords with pbkdf2_sha256 — selected in preference to bcrypt for its consistent performance across development environments — and issues a seven-day JSON Web Token on successful login, which the frontend attaches as a bearer token to every subsequent API request.

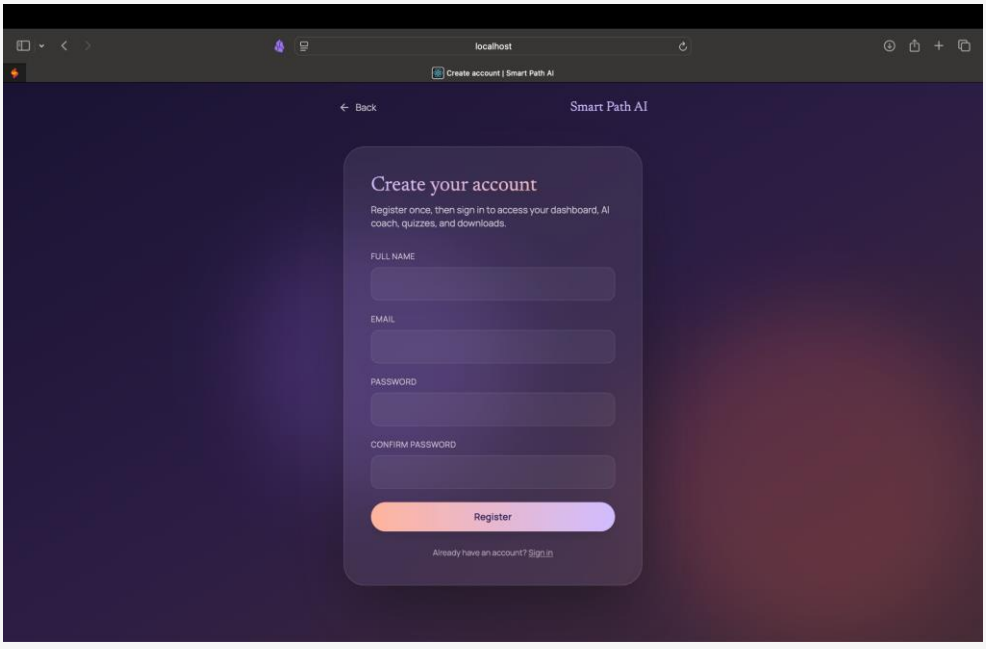

Fig. 3. Screenshot of the landing page and the login / registration screen.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

The personalized dashboard endpoint was exercised by authenticating as a seeded demo user, retrieving the /api/dashboard response, and confirming that the returned payload correctly combined the user's live eligibility evaluation across the full examination catalog, their saved-examination list, the count of currently approved notifications, and their existing study-timetable count into a single aggregated view, indicating that the dashboard's cross-cutting read from the eligibility service, the saved-exam table, and the notification and timetable tables functioned correctly end-to-end.

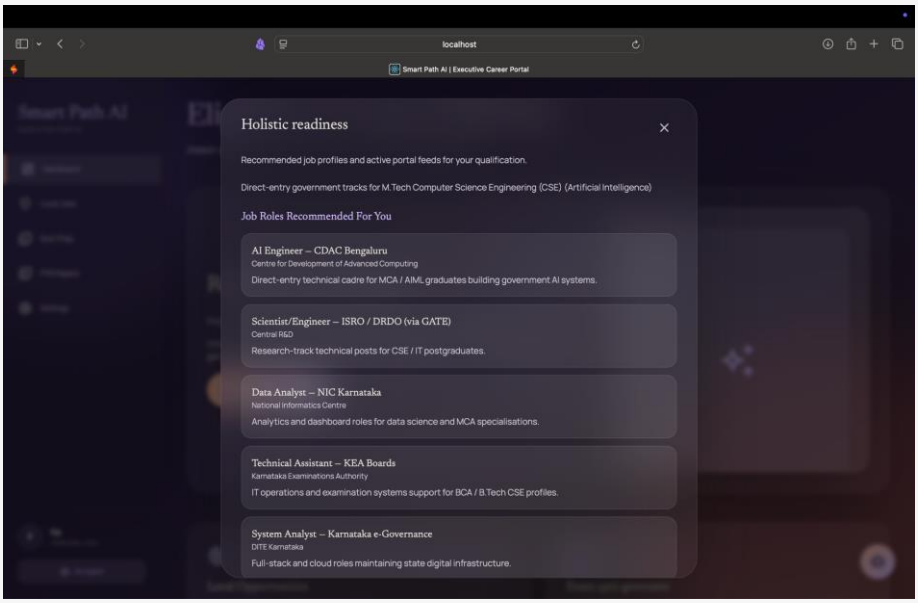

Fig. 4. Screenshot of the personalized dashboard showing the eligibility summary, saved examinations, and notification count.

IX. ELIGIBILITY MATCHING AND CAREER-PATH RECOMMENDATION METHODOLOGY

SmartPath AI's eligibility engine evaluates a candidate against an examination using three independent rule checks — age, state of domicile, and academic qualification — and reports an examination as eligible for a given user only when none of the three checks returns a definite failure:

$$eligible = age\_ok \ AND \ state\_ok \ AND \ degree\_ok$$

The age_ok check compares the user's declared age against the examination's minimum and maximum age, returning a definite failure only when the user's age falls outside that window, and treating a missing age as an indeterminate, non-blocking result. The state_ok check treats an examination whose declared eligible state is "India", "Any", or "All" as open to every domicile, and otherwise requires an exact or Karnataka-aware match between the user's declared state and the examination's declared state, again treating a missing user-declared state as indeterminate rather than a hard failure. The degree_ok check compares the user's qualification against either an explicit list of allowed degrees or a keyword-tokenized version of the examination's degree requirement, treating an unqualified "any graduate" requirement as automatically satisfied. Where a structured EligibilityRule record does not exist for a given examination, the engine falls back to the examination's own age and qualification fields, so that every seeded examination in the catalog can be evaluated even before a dedicated eligibility rule has been curated for it.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

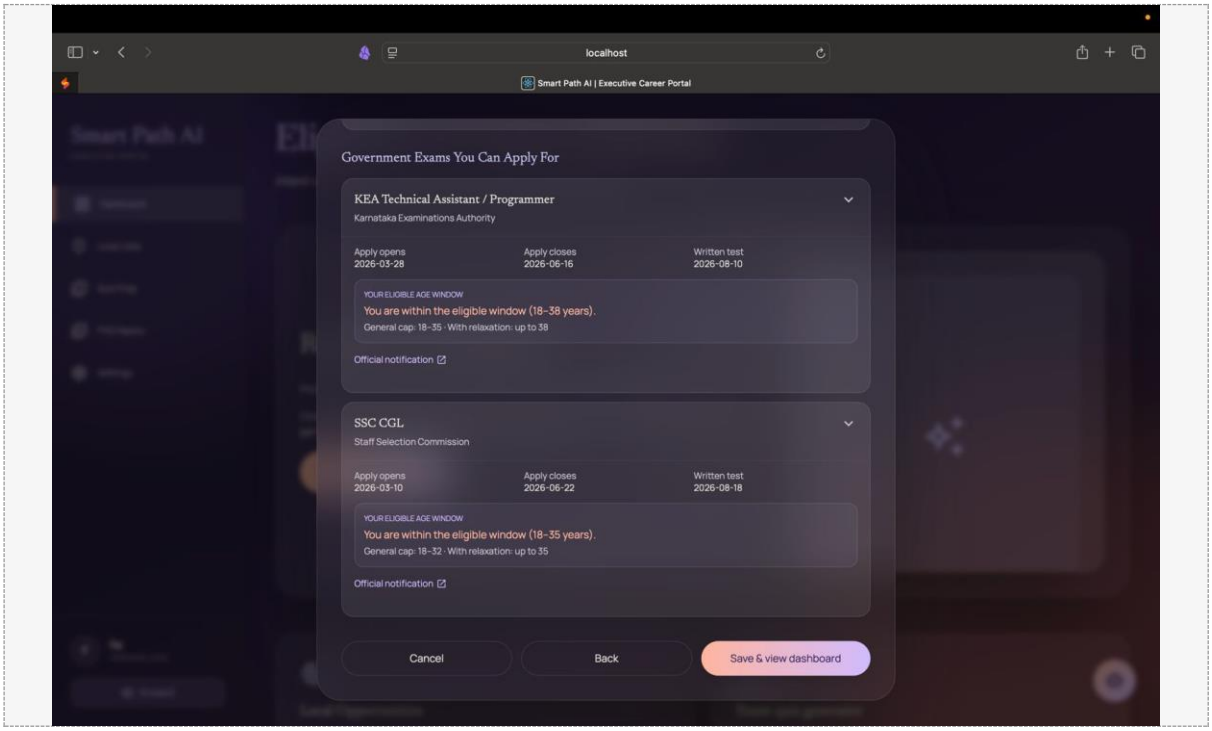

Fig. 5. Screenshot of the career / eligibility wizard showing course-to-examination matching and age-eligibility windows.

Layered on top of the per-examination eligibility engine is a course-to-career wizard that first maps a candidate's stated course of study onto one of several broad academic sectors defined in a course-catalog lookup table, filters the examination and government-job pool to that sector, computes a per-examination age-eligibility window that additionally accounts for category-based age relaxation (General, OBC, SC/ST, and other reservation categories), flags examinations whose application window is closing within ten days, and ranks candidate government-job roles higher when they match a specialty the user has indicated interest in. The wizard's output, a ranked, eligibility-annotated list of examinations and job roles for the candidate's specific course and specialty, was exercised functionally with representative course inputs from both the arts/humanities and STEM sectors, and correctly returned a narrower, appropriately reordered candidate list in each tested case.

X. GENERATIVE AI PRACTICE-CONTENT ENGINE

The mock-test generator, quiz generator, study-notes generator, and timetable recalibrator together form SmartPath AI's generative-AI practice-content engine, all routed through the Claude API and constrained to return a fixed-schema JSON object rather than free-form prose. The mock-test generator accepts an examination identifier, subject, difficulty level, and question count, selects an exam-style prompt fragment appropriate to that examination — for instance, distinguishing IBPS PO's banking-and-numerical-reasoning focus from KPSC KAS's Karnataka-state-governance focus — and requests a fixed number of four-option multiple-choice questions, each carrying a worked explanation for the correct option.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

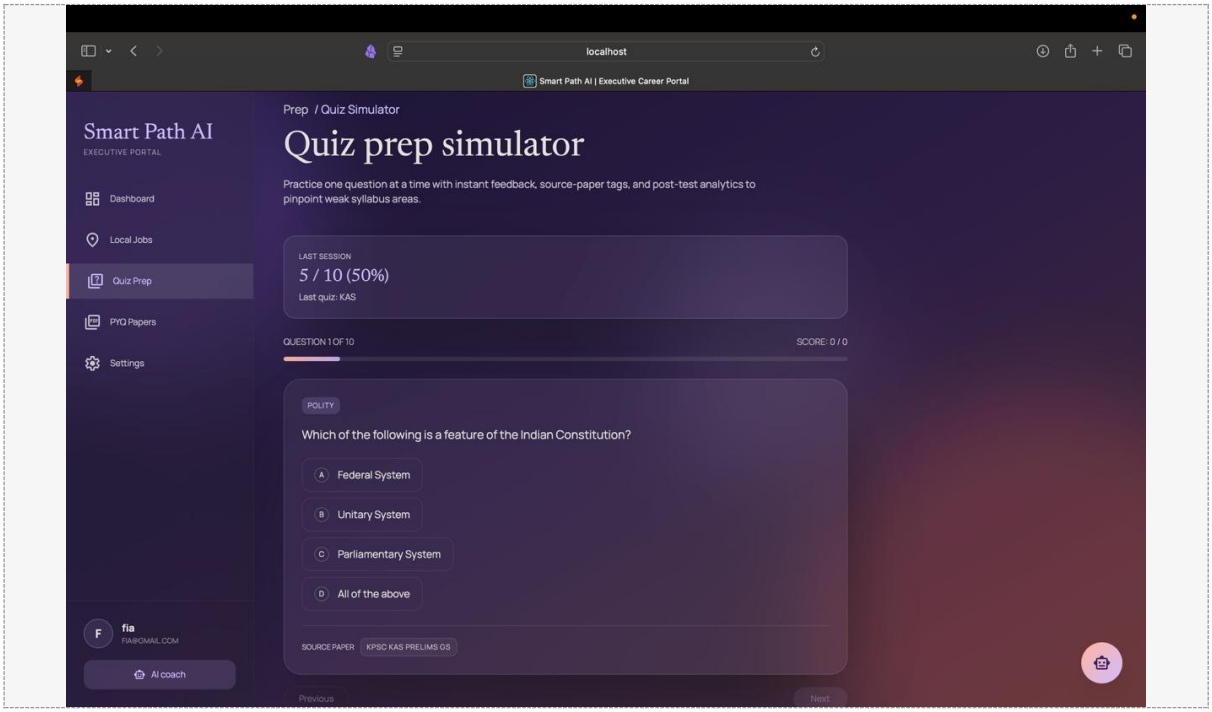

Fig. 6. Screenshot of the AI-generated mock test / quiz interface, showing a generated question set with worked explanations.

The study-notes generator similarly accepts an examination, topic, and self-reported proficiency level, and returns a structured note comprising a short summary, a set of key concepts with worked examples, mnemonics, common "exam traps," practice questions with hints, and a revision checklist, so that the output is directly consumable as a revision artifact rather than as an undifferentiated block of text. The timetable recalibrator inspects a user's most recent mock-test results, classifies each subject as weak (below fifty percent) or strong (seventy-five percent or above), and instructs the model to reallocate roughly thirty percent additional time to weak subjects and ten percent less time to strong subjects while preserving the user's existing total daily study hours and adding weekend revision blocks for weak areas, returning the recalibrated timetable in the same JSON shape as the input so that the frontend can render it without a schema change. Generation frequency is gated by the user's subscription tier: the free tier permits five mock tests, three study-note generations, and two saved timetables per calendar month, while the pro tier removes these caps, enforced through a shared limit-checking utility invoked before each generator runs.

XI. CONVERSATIONAL AI STUDY COACH

The chat module exposes a single authenticated endpoint that forwards the trailing twenty messages of a conversation, truncated to eight thousand characters per message, to the Claude API alongside a fixed system instruction that frames the assistant as a concise coach for Karnataka and national government examinations, instructs it to prefer bullet-point answers, and reminds it to direct the user to verify time-sensitive details against the corresponding official notification rather than treating the model's own output as authoritative for dates or vacancy counts.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026

DOI: 10.17148/IJARCCE.2026.15748

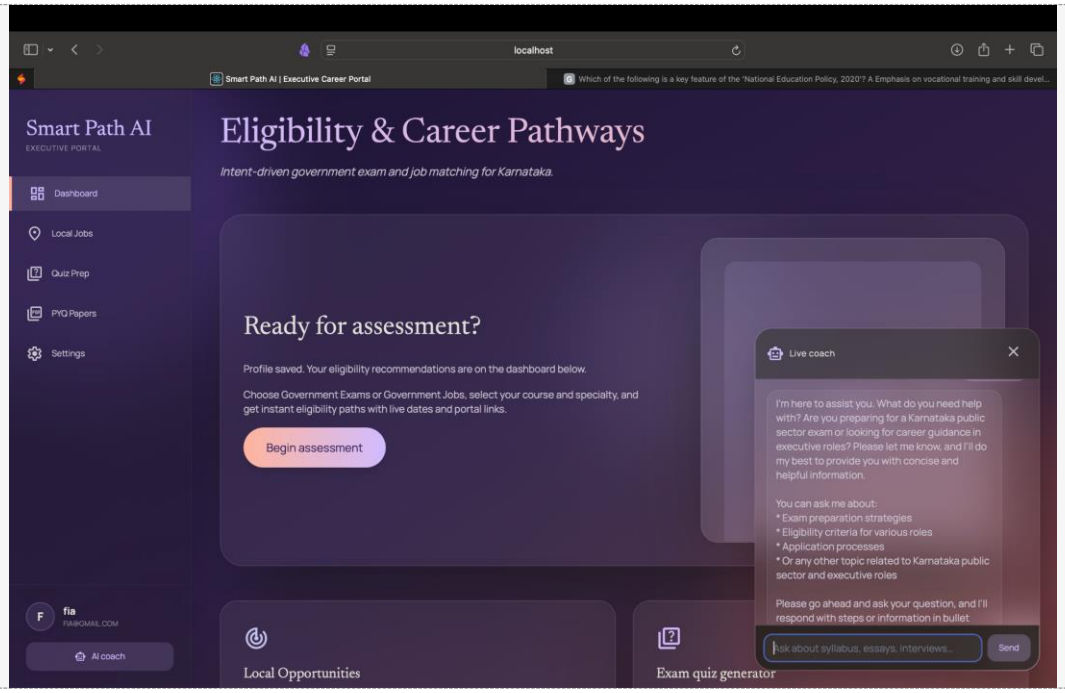

Fig. 7. Screenshot of a sample conversation with the AI study coach chatbot.

Because the chat endpoint is a thin, stateless wrapper around the underlying model rather than a bespoke dialogue-management system, conversational context is preserved simply by having the frontend resend the accumulated message history on every turn, which was found in functional testing to be sufficient for the assistant to maintain continuity across several turns of a typical exam-preparation conversation, such as a user first asking which examinations they are eligible for and then asking a follow-up question about one specific examination named in the assistant's previous reply.

XII. GAMIFICATION AND COMMUNITY ENGAGEMENT LAYER

To sustain daily engagement with the platform, SmartPath AI layers a gamification and community system on top of the core preparation workflow. A streak service logs each day's study minutes and questions attempted, increments a per-user streak counter when the previous day's activity record exists, and automatically awards badges — a three-day streak, a seven-day streak, a thirty-day streak, a first-mock-test badge, a ten-mock-test badge, and a fifty-study-hour badge — the first time each corresponding threshold is crossed, with badge state persisted so that an already-earned badge is never re-awarded. A leaderboard, cached for five minutes per examination to limit database load, ranks the top fifty users by average mock-test score for a given examination and separately reports the requesting user's own rank without requiring the full leaderboard to be recomputed on every request.

A study-groups feature allows a user to create or join a group scoped to a specific examination, with group membership and message history persisted so that peer discussion around a shared examination goal is possible directly within the platform, alongside a lightweight Pomodoro-style study timer that logs a twenty-five-minute focus session as completed activity once it finishes. A companion administrative surface, restricted to users with the admin role, exposes the queue of scraper-sourced notifications awaiting approval, allows an administrator to approve or reject each entry before it becomes visible to end users, surfaces a scraper-health log recording the status and record count of every scheduled scraper run, and allows an administrator to manually re-trigger any individual scraper outside its regular six-hour schedule.

XIII. RESULTS AND DISCUSSION

Functional evaluation of the deployed system indicates that the moderated scraping pipeline, the rule-based eligibility engine, and the generative-AI content engine each behave as intended when exercised individually, and that their combination materially reduces the number of separate tools an aspirant would otherwise need to consult. The scraping-and-moderation pipeline correctly upserted rather than duplicated notifications across repeated scraper runs, and the administrative approval queue provided a straightforward mechanism to prevent an incorrectly parsed or low-confidence

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026
DOI: 10.17148/IJARCCE.2026.15748

scraped listing from reaching end users, addressing a reliability concern inherent to any scraping-based aggregation system that operates against websites the platform does not control.

The rule-based eligibility engine's treatment of missing profile fields as indeterminate rather than as a hard failure was found, on review, to be an important design choice: because many users leave age, state, or qualification fields blank immediately after registration, treating a missing field as a failure would have caused the dashboard to report most examinations as ineligible by default, which would misrepresent the candidate's true eligibility and could discourage a genuinely eligible user from applying. The generative-AI content engine's fallback behaviour, returning a curated static question bank whenever no AI provider is configured or the model's JSON response fails validation, was similarly found to be an important resilience property, since it ensures that the quiz feature remains usable even when an external API dependency is temporarily unavailable, at the cost of reduced content variety during a fallback period.

Taken together, these results suggest that no single subsystem is sufficient on its own: the scraping-and-moderation pipeline provides breadth of coverage across many otherwise-siloed notification sources, the rule-based eligibility engine provides a transparent and auditable eligibility judgment that a purely statistical model would not directly provide, and the generative-AI layer provides content variety and personalization that a static question bank alone could not sustain at scale. A more rigorous, quantitatively powered evaluation of question quality, eligibility-rule accuracy against verified official criteria, and user engagement outcomes is identified as important future work once a larger base of real user interaction data has been collected.

XIV. PRACTICAL APPLICATIONS

The proposed framework has direct applications for competitive-exam coaching institutes, state-government digital-governance initiatives seeking to improve public awareness of recruitment opportunities, and portfolio-grade full-stack projects that seek to demonstrate the integration of web scraping, rule-based decision logic, and generative AI in a production-style architecture. Coaching institutes can adapt the moderated scraping pipeline and eligibility engine to keep their own notification boards current with minimal manual data entry, while continuing to layer their own subject-matter content on top of the platform's generative practice-content engine. State recruitment boards and allied digital-governance teams can draw on the eligibility-rule schema and the age-relaxation logic in the career wizard as a reference implementation for public-facing eligibility calculators. Developers and students can use the modular router, service, and AI-module structure demonstrated here as a reference architecture for combining a scheduled scraping layer, a rule engine, and a large-language-model content-generation layer on top of a modern FastAPI and React stack.

XV. LIMITATIONS

The most significant limitation is that the current evaluation is functional and qualitative rather than a statistically powered offline or online study: no formal precision or recall figures have been computed for the scraper's notification-extraction accuracy, nor has the eligibility engine's rule output been validated against a verified, independently sourced ground truth for every seeded examination. The scraper implementations rely on the HTML structure and keyword patterns of each authority website's current markup, so a redesign of any of the five scraped websites could silently degrade extraction quality until the corresponding scraper's selectors are updated; the administrative moderation queue mitigates but does not eliminate this risk, since it depends on a human reviewer noticing degraded output. The eligibility engine's degree-matching logic relies on simple substring and token matching against a qualification string rather than a curated taxonomy of recognized Indian degree names, so unusual phrasings of a qualification may be misclassified. The generative-AI content engine's output quality has not been benchmarked against subject-matter-expert-authored questions for factual accuracy, and, being backed by a third-party language-model API, its practice-content generation is also subject to that provider's availability and rate limits. Finally, the community features, leaderboards and study groups, depend on a sufficient base of active users to be meaningful, and their behaviour with a very small user base has not been separately stress-tested.

XVI. FUTURE WORK

- Conduct a formal accuracy evaluation of scraper-extracted notification fields against manually verified official notifications.
- Benchmark AI-generated mock-test and quiz questions against subject-matter-expert-authored questions for factual accuracy and difficulty calibration.
- Replace substring-based qualification matching in the eligibility engine with a curated taxonomy of recognized Indian academic qualifications.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 7, July 2026
DOI: 10.17148/IJARCCE.2026.15748

- Extend the moderated scraping pipeline to additional national and state examination authorities beyond the five currently covered.
- Activate push-notification delivery, already scaffolded through the platform's VAPID configuration, for approved notifications matching a user's saved examinations.
- Develop a native mobile client to complement the existing Progressive Web Application.
- Incorporate voice-based interaction with the AI study coach for accessibility.

XVII. CONCLUSION

This paper presented SmartPath AI, a full-stack government-exam-intelligence platform that unifies notification discovery, rule-based eligibility screening, and generative-AI-driven exam preparation for both national and Karnataka state-level government examinations. By combining a moderated, scheduled web-scraping pipeline, a transparent rule-based eligibility and career-matching engine, and a Claude-API-backed content-generation and document-intelligence layer within a single FastAPI and React architecture, SmartPath AI demonstrates that classical software-engineering techniques, scheduled scraping, relational data modelling, and rule-based decision logic, can be combined with modern generative AI to produce a practical, extensible platform for a genuinely high-stakes domain. The modular router and service structure of the system allows individual components, such as an individual examination scraper or the underlying language model powering the content-generation engine, to be upgraded independently as requirements evolve.

Collectively, these results suggest that no single technique is sufficient on its own: web scraping provides breadth of coverage, rule-based logic provides transparent and auditable eligibility decisions, and generative AI provides personalized, scalable content generation. Beyond its technical contribution, SmartPath AI offers practical value as a reference architecture for developers and institutions seeking to build eligibility-aware, AI-assisted platforms in domains, such as public-sector recruitment, where transparency, currency of information, and personalized preparation all matter simultaneously.

REFERENCES

[1]. A. Khaldi, R. Bouzidi, and F. Nader, "Gamification of e-learning in higher education: a systematic literature review," Smart Learning Environments, vol. 10, no. 1, p. 10, 2023.
[2]. C. Bhatt, A. Bisht, R. Chauhan, A. Vishvakarma, M. Kumar, and S. Sharma, "Web scraping techniques and its applications: A review," in Proc. 2023 3rd Int. Conf. Innovative Sustainable Computational Technologies (CISCT), 2023, pp. 1–8.
[3]. G. Biancini, A. Ferrato, and C. Limongelli, "Multiple-choice question generation using large language models: Methodology and educator insights," in Adjunct Proc. 32nd ACM Conf. User Modeling, Adaptation and Personalization (UMAP '24 Adjunct), 2024.
[4]. G. Engin, B. Aksoyer, M. Avdagic, D. Bozanlı, U. Hanay, D. Maden, and G. Ertek, "Rule-based expert systems for supporting university students," Procedia Computer Science, vol. 31, pp. 22–31, 2014.
[5]. M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF RFC 7519, May 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7519
[6]. FastAPI Documentation, "FastAPI framework, high performance, easy to learn, fast to code," 2025. [Online]. Available: https://fastapi.tiangolo.com/
[7]. React Documentation, "React – A JavaScript library for building user interfaces," 2025. [Online]. Available: https://react.dev/
[8]. Anthropic, "Claude API documentation," 2025. [Online]. Available: https://docs.claude.com/
[9]. PyMuPDF Documentation, "PyMuPDF: Python bindings for MuPDF," 2025. [Online]. Available: https://pymupdf.readthedocs.io/

