Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026 . Vol. 5 . No. 3; e-ISSN: 2808-4519

Design and Implementation of the Vocaseek Platform as a Web-Based Internship Recruitment and Student Talent Management Information System Using React and Laravel

Muhammad Rafi Adiansyah [1], Laili Magfiroh Novia [2], Rendra Ardika [3], Dea Indah Lestari [4*], Ardhon Rakhmadi [5], Mohamad Irwan Afandi [6]

[1,2,3,4,5,6] *UPN Veteran Jawa Timur,Indonesia*

23081010086@student.upnjatim.ac.id [1], 23082010017@student.upnjatim.ac.id [2], 23081010074@student.upnjatim.ac.id [3], 23082010023@student.upnjatim.ac.id [4*], ardhon.rakhmadi.fasilkom@upnjatim.ac.id [5], mohamadafandi.si@upnjatim.ac.id [6]

Abstract

The development of information technology has driven digital transformation in various fields, including the internship recruitment process and student talent management. However, the internship search and candidate selection process is still often done manually through various unintegrated platforms, causing difficulties for both students and companies. This study aims to design and implement the Vocaseek platform as a web-based internship recruitment and student talent management information system. The system was developed using React and Vite on the frontend and Laravel as a REST API-based backend. The platform supports four types of users: applicants, partner companies, admin staff, and super admins. The main features provided include account registration and verification, applicant profile management, work character pre-tests, vacancy management, candidate management, company verification, and system administration management. The implementation results show that Vocaseek is able to integrate the entire internship recruitment process into one centralized platform. The system makes it easier for students to find internship opportunities and helps companies in selecting candidates more effectively and efficiently.

Keywords *: Internship recruitment, Laravel, React, Talent management, Vocaseek*

1. Introduction

Internship programs play a crucial role in higher education because they provide students with the opportunity to connect academic knowledge with real-world work situations. These experiences can strengthen job readiness, communication skills, problem-solving skills, discipline, and understanding of organizational culture. Research on internships also shows that students' involvement in structured work experiences contributes to the development of skills needed when entering the professional world [1]. Therefore, access to relevant internship information and a clear recruitment process are crucial in supporting students' transition from academia to the workforce.On the other hand, the internship search process is still often conducted through separate channels, such as social media, chat groups, company websites, and personal information sharing. This scattered information forces students to conduct repeated searches, compare inconsistent job posting formats, and submit documents across multiple channels. This can create uncertainty regarding the validity of companies, application deadlines, position requirements, and application status updates. For students entering the recruitment process for the first time, this unclear process can also lead to incomplete documents or applications being submitted before candidate profiles are ready for review.

Similar issues arise for companies. Managing candidates through email and separate documents complicates data retrieval, candidate comparison, recording selection results, and providing status updates. The digital transformation of human resources has shifted recruitment approaches toward the use of digital platforms, data analytics, and faster, more documented processes [2]. Recent recruitment trend studies also place the use of digital channels, candidate experience, data-driven decision-making, and the ability to reach a wider audience as key components of the evolving recruitment process [3].

Although numerous job vacancy platforms are readily available, an internship recruitment system is not sufficient simply to display a list of positions. The system must address candidate data availability, document completeness, partner credibility, access restriction, and a continuous selection process. In the context of Vocaseek, these requirements translate into a registration process with email verification, company legal document review, student profile management, work character *pre-testing* , job applications, candidate reviews, selection

status changes, and administrative user management. This approach positions the platform as an operational recruitment and *talent management system* , not simply a job board.

Previous research on recruitment digitization has largely discussed decision support, the use of artificial intelligence, or candidate management in specific domains. A REST API-based recruitment decision support system, for example, focuses on candidate ranking using the fuzzy TOPSIS method. Other research develops a REST API for candidate referral management in human resource information systems. Studies on artificial intelligence in recruitment highlight the opportunities for automation as well as the issues of bias, transparency, and data protection [4]. These studies demonstrate the importance of recruitment digitization, but the need for a multi- *role internship platform* that combines partner verification, applicant profile readiness, *pre-testing* , vacancy management, *talent management* , and user administration still requires integrated implementation.

Vocaseek is designed to address these needs through a React *frontend architecture* and a Laravel *backend* based on a REST API. React is used to build a component-based interface, while Laravel handles business logic, validation, authentication, file storage, email delivery, and database access. The system differentiates authentication sessions based on user portals to prevent *tokens* belonging to applicants, companies, *admin staff* , and *super admins* from being mixed up. Furthermore, the system provides Indonesian and English language support, API documentation using Swagger/OpenAPI, and a development environment that can be run through Docker Compose or local MySQL.

The purpose of this research is to design and implement vocaseek as a web-based internship recruitment and student talent management information system. The main contributions of the research include: (1) designing an end-to-end internship recruitment flow; (2) implementing a multi- *role access model* ; (3) integrating profile completion and *pre-tests* as application prerequisites; (4) implementing tiered company verification; and (5) providing a rest api architecture that supports *frontend* and *backend separation* . Evaluation focuses on the suitability of the main functions through *black box testing* , while large-scale usability evaluation and performance testing are placed as the next development agenda.

2. Related work

Digital recruitment is the application of technology to expand candidate reach, accelerate information processing, and improve the traceability of the selection process [5] explains that digital transformation changes the recruitment paradigm through digital platforms, analytics, and automation. Triastuti also shows that recruitment developments are leading to the use of technology, strengthening employer branding, and a more responsive candidate experience. These two studies form the basis that recruitment systems need to provide more than just job postings, namely consistent data management and process communication.

In the information system domain, developing a decision support system for outsourcing recruitment using fuzzy topsis integrated with rest api. The system shows that service separation through api can support recruitment decision processing. [6] developed rest api in human resource information system for employee referral management domain and reported valid *endpoint test results. Compared to the two systems, vocaseek does not focus on mathematical ranking or internal referral* , but rather on the student internship flow involving applicants, companies, *admin staff* , and *super admins* .

The use of rest API in Vocaseek is based on the need to separate interface and business logic. [7] found that adherence to rest design rules improves the understandability of Web API. Therefore, Vocaseek *endpoints* are grouped by public, auth, *internal* , *company* , and admin domains, and use consistent response patterns. *Endpoint documentation* is also provided through OpenAPI. The OpenAPI specification defines a standard, language-independent interface for describing HTTP APIs, so that service consumers can understand the capabilities, parameters, data schemas, and authentication mechanisms [8].

React provides a component-based interface approach that can be combined into pages and applications [9]. These characteristics support the creation of repeatable components such as navbars, sidebars, job cards, forms, modals, and candidate tables. Vite is used as a development and *build tool* because it provides a development server and a *build process* to produce production bundles [10]. On the server side, Laravel provides application structure, *routing* , *middleware* , validation, ORM, storage, mail, and *queues* . Laravel Sanctum is used because it supports SPA authentication and *token* -based APIs. MySQL is chosen to store relational data for users, profiles, jobs, applications, test answers, experience, certifications, and admin invitations.

Vocaseek's difference from previous research lies in the integration of internship recruitment and talent management flows into a single multi- *role system* . Applicants cannot apply directly until their profile and *pre-test* are complete; companies cannot publish vacancies until their partner status is active; *admin staff* can assist *with reviews* but cannot make final decisions; and *super admins* manage company verification and administrator invitations. This mechanism places process control within the system's business rules, allowing for accountability and monitoring of operational flows.

3. Research Method

This research uses a software engineering approach with *agile development methods through the scrum* framework . System development is carried out in stages in several cycles that include needs analysis, design, implementation, testing, evaluation, and system improvement, the main stages consist of needs analysis, system design, module implementation, rest api integration, functional testing, and improvement. The iterative approach was chosen because the interface and *backend service requirements* evolve along with the evaluation of user flows. Each module is developed in smaller parts, tested, then integrated with other modules. The object being evaluated is the implementation of the vocaseek platform in a development environment with a react *frontend* , a laravel *backend* , and a mysql database.

3.1. Requirements Analysis

The needs analysis begins with mapping key actors and activities. Applicants require access to registration, email verification, login, profile management, document upload, experience, certification, *pre-testing* , job search, application, and status monitoring. Companies require registration with legal documents, email verification, partner approval, profile management, job creation, applicant review, document

review and assessment , and candidate status updates. *Administrative staff* assist with the *review process* , while *super admins* have final authority over the company and administrator management.

Non-functional requirements include access rights separation, authentication session consistency, file validation, interface responsiveness across screen sizes, API documentation, and the system's ability to be used in both Indonesian and English. Access security is implemented through *bearer tokens* , *middleware, and a secure interface* . *Role* Check and *endpoint restrictions* . Applicant and company files are stored in Laravel's public storage with file path information in the database. Verification emails, *password resets* , and administrator invitations are processed through *a queue* so that the main *request* doesn't have to wait for the email to complete.

Table 1: Main actors and functional requirements

Actor	Main requirements	Access limitation
Intern applicant	Profile, documents, experience, certification, *pre-test* , vacancy search, application and history	Cannot apply before profile completion and *pre-test* submission
Partner company	*Company* profile, job posting, applicant *review* , *talent management* and status update	Can create vacancies only when *partner* status is active
Admin staff	Talent and *partner review* , document inspection and own profile management	Cannot perform final *company approval* or full user management
Super admin	System overview, talent, *partners* , *company* verification, admin users and invitations	Has final *approval* and rejection authority

3.2. System Design and Implementation

The system design includes *client-server architecture, frontend* route mapping , *endpoint design* , data modeling, and interface design. The project structure is separated into *frontend* and *backend folders* . *The frontend* is built as a *single-page application* and uses React Router DOM for navigation without full page reloads. Axios handles HTTP communication to the API. *The backend* groups routes based on access rights and function domains, then the controller validates input, executes business rules, accesses models, and returns JSON *responses* .

The implementation is modular. On *the frontend* , pages are divided into public, *internal* , *company* , *staff admin* , and *super admin portals* . Repeated components are separated for reuse. On *the backend* , controllers, models, *middleware* , *request* validation, support services, mail, and *queues* are separated by responsibility. API documentation is published via Swagger in /docs or /docs/swagger, while the raw specifications are available in /docs/openapi.yaml. This separation helps synchronize data contracts between *the frontend* and *backend teams* .

3.3. Functional Testing

Testing was conducted using *black box testing* because the evaluation focused on the suitability of input, business processes, and outputs as seen from the user's perspective without assessing the *internal code structure* [10]. The literature states that *black box testing* is suitable for functional testing based on system behavior, while white box testing is more oriented towards the *internal structure* of the system. Test scenarios included authentication, email verification, *role restrictions* , profile completion, *pre-test* , application, job management, candidate status updates, and company verification. Testing was conducted in a development environment. Test results were used to ensure the core flow was usable, not to replace broader security, load, or usability testing.

4. System Design

4.1. System Architecture

Vocaseek implements *a client-server architecture* with a separation of presentation layers, application services, and data storage. The user's browser loads a React SPA from *the frontend* . User interactions, such as login, profile updates, job searches, and candidate status changes, generate *requests* through Axios to the Laravel API. *The backend* verifies authentication and *roles* , validates payloads, executes business rules, and then reads or updates data in MySQL. The results are returned in a JSON *response* and rendered by React components.

Separating *the frontend* and *backend* offers several benefits. Interfaces can be developed without mixing display logic with database management, while APIs can be used by more than one client if the platform is developed into a mobile application. On the other hand, this separation requires clear *endpoint contracts, consistent error management, and proper token control. The findings of Bogner et al. [7] regarding the understandability of REST design rules support the use of* consistent URL patterns, HTTP methods, status codes, and *response structures.*

The development environment provides Docker Compose for running *the frontend* , *backend* , Laravel *queue* , and MySQL Docker options. The system also supports MySQL Laragon on Windows. The local *backend* is accessed by default via http://localhost:8001/api, while *the frontend* is accessed via http://localhost:5173. This flexibility makes it easy for teams to use the appropriate environment, but environment configuration must ensure compliance with API URLs, database credentials, mail services, storage links, and allowed domains.

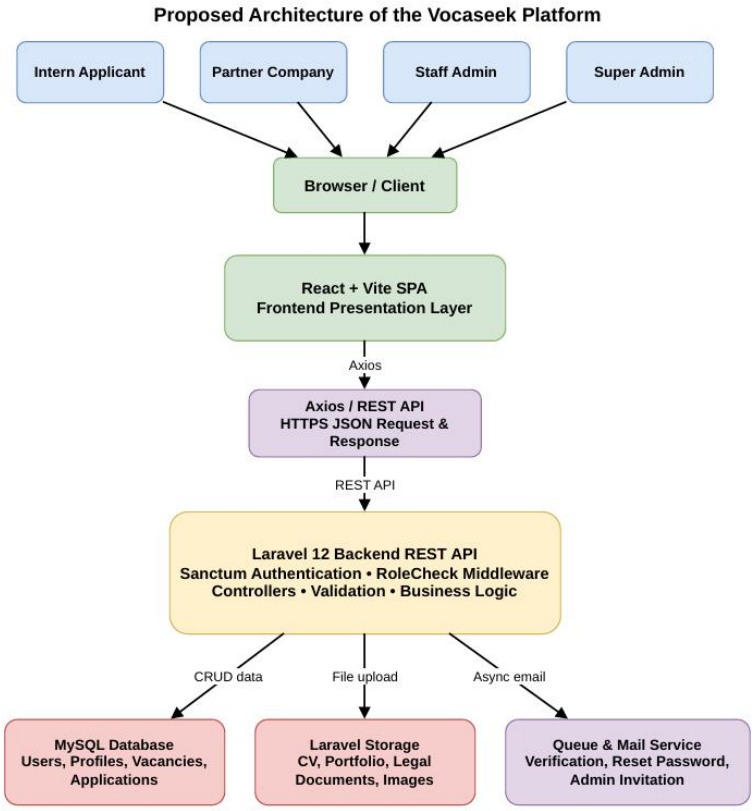

Fig. 1: Proposed architecture of the Vocaseek platform

4.2. Role and Business Workflow Design

Access rights are designed based on four *roles* . Applicants use the *internal portal* , companies use the *company portal* , and system administrators are divided into *staff admins* and *super admins* . After logging in, the authentication *response* contains identity, *role* , status, and *token* . *The frontend* maintains sessions *scoped* by portal. This strategy prevents *tokens* from different portals from replacing each other when multiple accounts are used in the same browser or tab. *The backend* remains the primary decision-making source through *middleware. Role* Check so that route manipulation on the client side does not provide access to data that is not authorized to be seen.

workflow begins with registration, email verification, login, profile completion, document upload, and *pre-test* . A profile is considered complete once essential attributes, such as a photo, CV, and university, are available. *The pre-test* can only begin if the profile meets the requirements, is completed once, contains 20 questions, and has a 20-minute time limit. Once both prerequisites are met, applicants can submit their applications. The system checks the uniqueness of the applicant-vacancy combination to prevent duplicate applications. The company's *internal status* is then normalized as Pending, Accepted, or Rejected on the applicant's screen for easier understanding.

workflow begins with registration with NIB, LOA, and deed in PDF format. After email verification, the company account cannot be used until it goes through an administrative *review* . *Admin staff* can check documents and provide initial *reviews* , while active or rejected decisions can only be given by *the super admin* . Active companies can update profiles, create vacancies, view applicants per vacancy, review academic data, experience, certifications, documents, and *pre-test answers* , then change the candidate status from pending to the *review* , interview, *shortlisted* , accepted, or rejected stage.

On the administrative side, *super admins* manage the system overview, talent data, *partner data* , company verification, user management, and administrator invitations. Admin invitations use *tokens* with an expiration date, used, cancelled, or expired status, and resend and cancel functions. Invitees must open an activation link and create a password before the account becomes active. *Admin staff* have their own workspace to assist with *reviews* without having *final* approval rights . This separation applies the principle of least privilege to business processes.

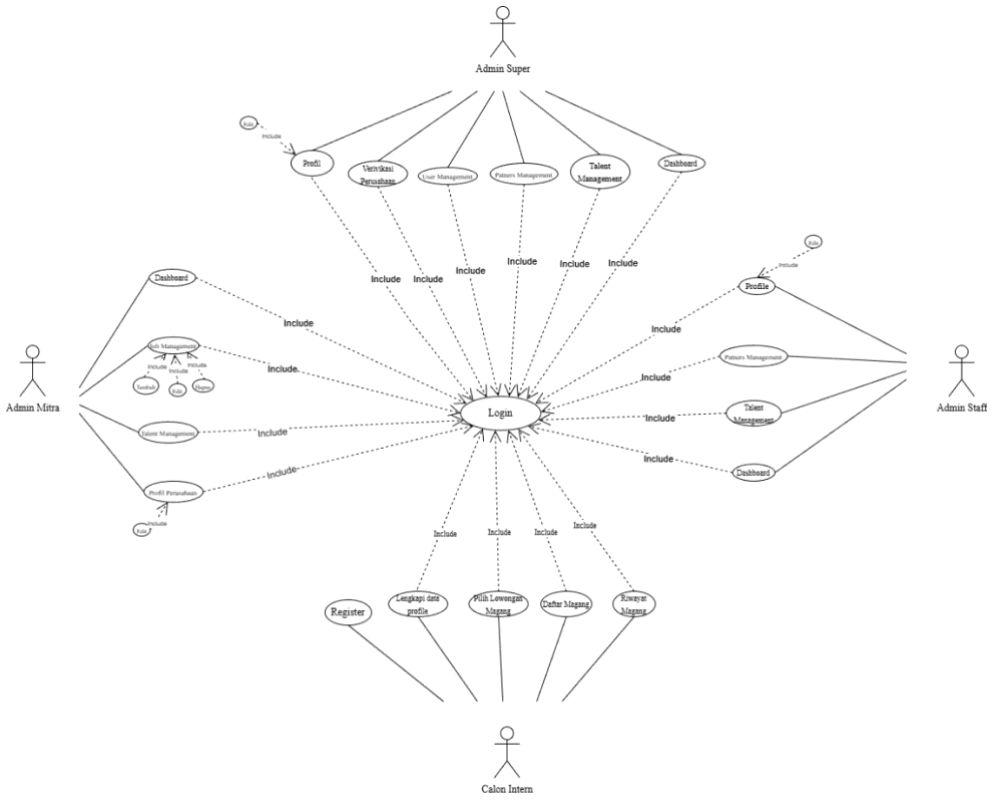

Fig. 2: Use case diagram of Vocaseek

4.3. Database Design

The database is designed with users as the central account table for all *roles* . This table stores name, email, password, *role* , status, phone number, photo, preferred locale, google_id, and email_verified_at. Centralizing accounts simplifies authentication and identity management, while *role- specific information* is placed in the profiles table. *intern_profiles stores personal data, academic data, documents, completion status, and pre-test* implementation information . *company_profiles* stores company identity, industry, size, address, contact information, legal documents, logo, banner, vision, mission, social media, and partner status.

Job openings are stored in lowongans and are associated with companies. Job attributes include position title, category, job type, description, requirements, location, *remote - onsite - hybrid work arrangements* , salary, closing date, start date, and status active, open, closed, or draft. Applications are stored in job_applications, which connects applicants to openings and records the status of the selection process. Uniqueness constraints are applied logically to prevent applicants from submitting the same application more than once.

test_answers stores Yes/No answers to each *pre-test question* . Experience and certifications are separated into *intern_experiences* and *intern_certifications* to allow applicants to have more than one record. admin_invitations stores the administrator's invitation *token* and status, while pending_registrations holds registrations that have not yet completed email verification. categories and jobs remain available to support *landing page data* or legacy module compatibility.

The backend provides field aliases between the old and new schemas, for example, position_title with job_title and internship_type with work_arrangement. This compatibility reduces disruption when old and new modules use different attribute names. However, aliases need to be managed carefully to avoid creating technical debt. In future development, schema contracts need to be standardized through API versioning and OpenAPI documentation.

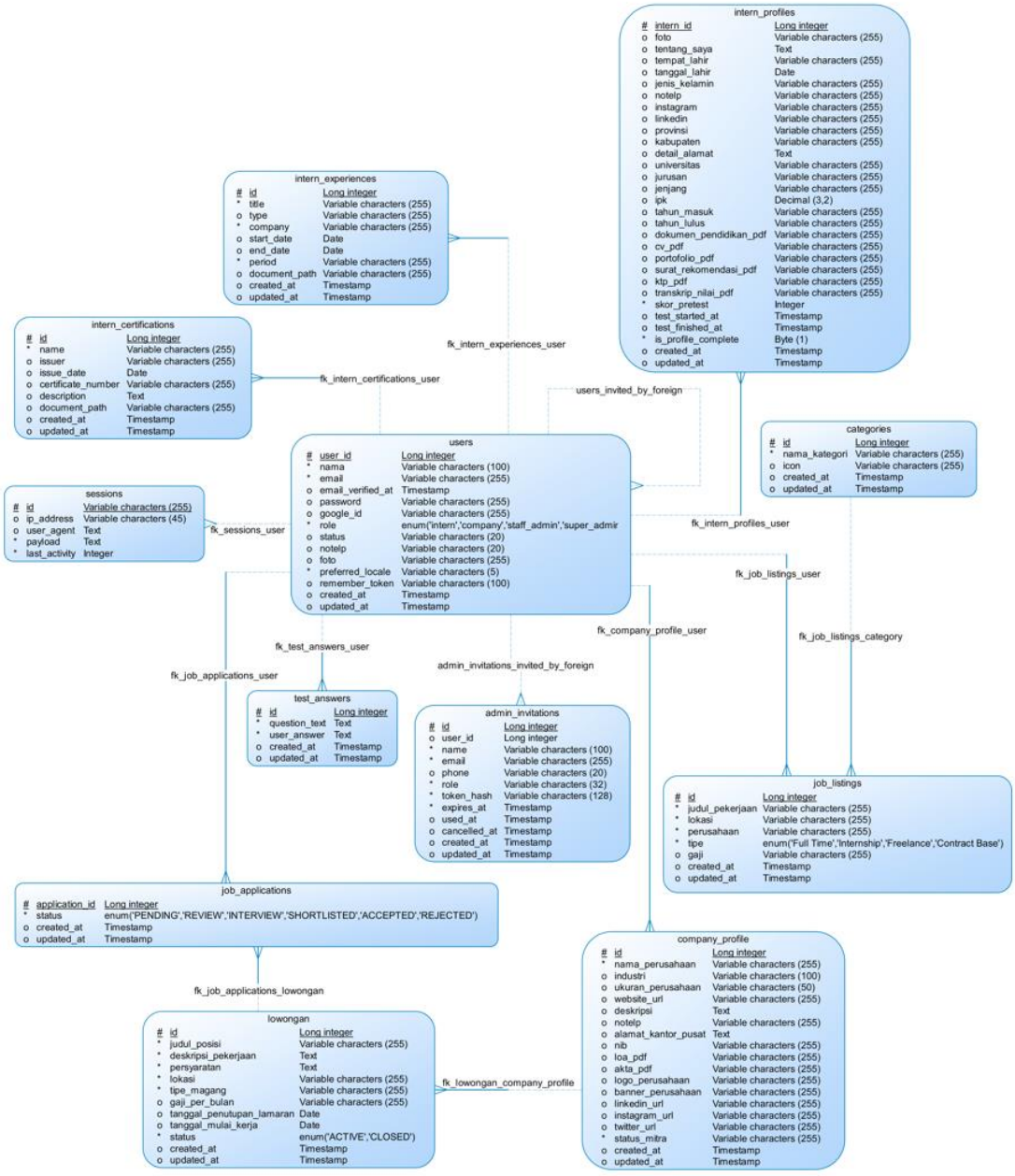

Fig. 3: Entity relationship diagram of the main Vocaseek database

Table 2: Main data entities in Vocaseek

Entity	Purpose	Main relation
users	Stores account, *role* , status, locale and verification data	One-to-one with *internal* / *company* profile; referenced by applications and invitations
intern _profiles	Stores personal, academic and applicant document data	Belongs to user; related to experience, certification and test answers
company profiles	Stores *company* identity, legal documents and *partner* status	Belongs to user; has many vacancies
vacancies	Stores vacancy information and publication status	Belongs to *company* ; has many job applications
job_applications	Stores applicant-vacancy relationship and selection status	Belongs to *intern* and vacancy
test_answers	Stores each *pre-test response*	Belongs to *internal* and question/test context
admin_invitations	Stores administrator invitation *token* and lifecycle	Created by administrator and consumed by invited user

5. Implementation and Results

5.1. Public Interface and Landing Information

The public page serves as an entry point for unauthenticated users. Navigation provides access to the homepage, job listings, partner listings, contact information, login, and registration. *The landing page* displays the service's identity, an invitation to find an internship,

information on the user process, public statistics, a list of popular jobs, and active partners. Statistics are displayed *via* the /landing-stats *endpoint* , while jobs and partners are displayed via /popular-vacancies and / *partners* . Separating data from visual components allows page content to be updated without changing the interface structure.

The display in Fig. 4 uses visual hierarchy to place the main message, action buttons, and illustrations in the first area users see. The next section explains the steps of use, so potential applicants understand that the process doesn't stop at finding a job, but includes profile creation and application. From an information system perspective, the public page serves both an educational and conversion function: users see *an overview of the service before selecting an internal* or *company* registration portal .

Responsive design is implemented so that menus, cards, images, and spacing between elements can adapt to screen size. Navigation components and cards are reused on other public pages to maintain consistency. Company information is only displayed when a partner is active, so the publicly visible *partner list is for entities that have gone through an administrative verification process.*

Fig. 4: Vocaseek public landing page

5.2. Internal Dashboard and Application Readiness

After successful authentication, applicants are directed to *the dashboard. internal* . This page presents the main steps: completing a profile, taking *a pre-test* , searching for vacancies, and monitoring status. Presenting the flow in stages helps users understand the dependencies between activities. The system not only provides menus but also guides users to ensure the company's data is sufficient.

The dashboard utilizes profile information and test status to determine possible actions. When a profile is incomplete, buttons and messages direct users to personal data, academic information, and document pages. Once the profile is complete, users can begin *the pre-test* . Once the test is complete, the application menu can be accessed. Prerequisite validation remains in place on *the backend* to prevent users from bypassing the flow through URL manipulation or manual *requests* .

From a user experience perspective, *the dashboard* reduces cognitive load because candidate readiness status is displayed on a single page. Users don't need to navigate through menus to determine the next step. This model also benefits companies because candidates who reach the application stage already have minimal data to review.

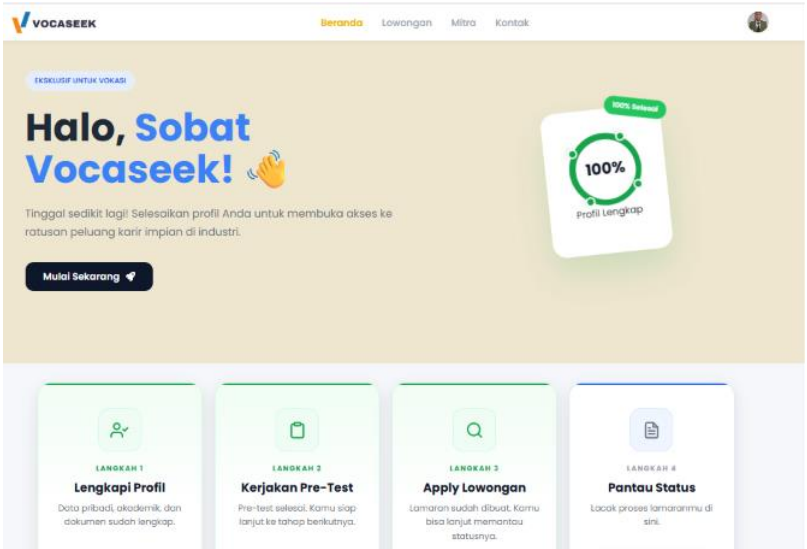

Fig. 5: Internal dashboard and main recruitment stages

5.3. Applicant Profile and Document Management

The applicant profile module includes personal data, academic data, documents, experience, and certifications. Personal data includes name, email, photo, about me, place and date of birth, gender, phone number, Instagram, LinkedIn, province, district, and detailed address. Academic data includes university, major, level, GPA, year of entry, and year of graduation. Separating pages by data group prevents the form from becoming too long and makes it easier for users to correct specific sections.

Uploadable documents include CVs, portfolios, educational documents, letters of recommendation, ID cards, and transcripts. Validation is performed on file types and mandatory attributes before *the backend* saves files to public storage. Experience and certifications use one-to-many relationships, allowing users to add multiple notes and supporting documents. The interface displays previously saved data so users can make updates without repeating all input.

Fig. 6 shows a profile view that summarizes a candidate's identity and information. The profile view page is important because it provides a preview of the data that will be seen by the company. This allows applicants to evaluate the completeness and readability of the information before submitting an application. A complete profile's status is calculated based on key data, not just account existence, so the system can maintain a minimum quality of candidates entering the *talent pool* .

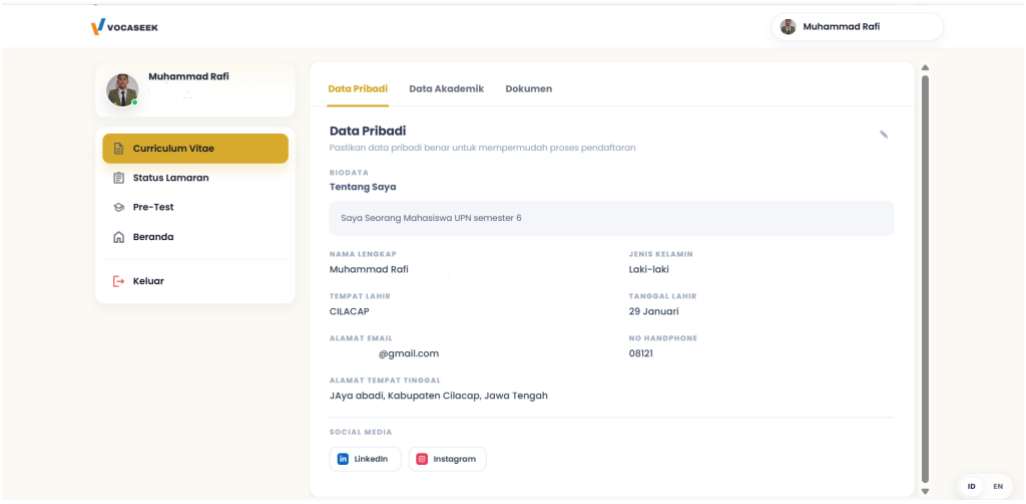

Fig. 6: Applicant profile summary in the internal portal

5.4. Pre-Test and Job Application Workflow

The pre-test is one of Vocaseek's distinguishing features. The test contains 20 Yes/No questions related to work character and *soft skills* , is 30 minutes long, and can only be taken once. The / *intern /start-test endpoint* is used to initiate the session and record the time, while / *intern* /submit-test receives answers and saves the results to test_answers. The single-test limit is intended to maintain the consistency of the *assessment data* used in the initial *review* .

The system ensures that profiles are complete before the test begins. Once the test is complete, applicants can select a job, review details, and submit an application via / *intern* /apply. *The backend* checks authentication, *role* , profile completeness, test status, job validity, closing date, and the existence of previous applications. If all requirements are met, job_applications is created with an initial status. Application history and status are then accessed via / *intern* /applications.

The pre-test in this implementation serves as an *internal screening instrument* , not a clinical psychometric tool or the sole basis for candidate acceptance. Answers should be read in conjunction with academic data, experience, documentation, and the interview process. This limitation is important to note so that the system's use does not result in oversimplified decisions about candidate character. Construct validation, reliability, and question fairness are further work if test results are to be used for automated ranking.

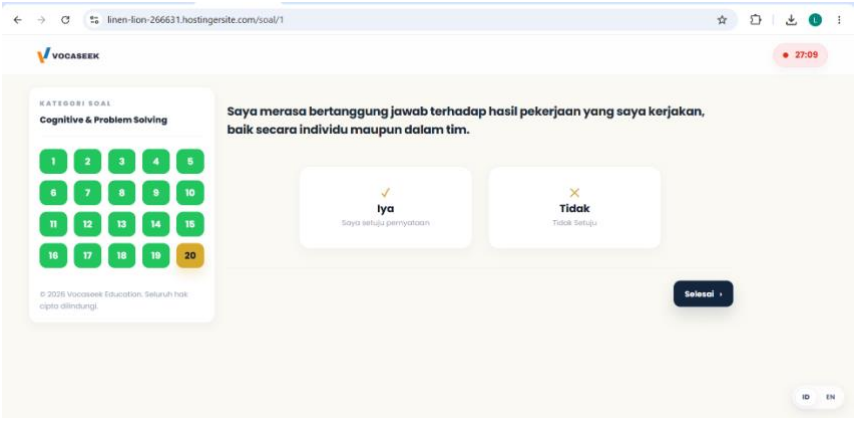

Fig. 7: Pre-test interface for internal applicants

5.5. Company Portal and Talent Management

company portal provides a statistics *dashboard* , company profile management, job postings, applicant lists, and *talent management* . The company profile includes the organization's identity, industry, size, website, description, vision, mission, contact information, address, legal documents, logo, banner, partner status, and social media. Companies can only create vacancies when their partner status is active. This rule ensures that the entity posting the vacancy has passed an administrative *review* .

Job openings can be created, viewed, edited, and deleted through the / *company /jobs endpoint* . The job opening form contains the position title, category, job type, description, requirements, location, work arrangements, salary, closing date, start date, and status. Using the draft status allows companies to prepare information without immediately displaying it, while "open" or "active" indicates the job is open to applicants, and "closed" indicates the recruitment process is over.

On the applicant page, companies can filter candidates based on vacancies and open candidate details. Information displayed includes personal data, academic background, experience, certifications, documents, and *pre-test answers* . Companies can change the status to pending, *review* , interview, *shortlisted* , accepted, or rejected. Each status change can be forwarded as a notification to applicants. *Talent management* also allows companies to view all candidates, shortlisted candidates, and create manual candidate entries for specific administrative needs.

The more detailed status structure on the company portal is normalized into simpler statuses on the applicant portal. *Review* , interview, and *shortlisted* can still be displayed as pending, while accepted can be changed to accepted and rejected to rejected. Normalization prevents *internal* company details from confusing users, but the original data remains for process monitoring purposes.

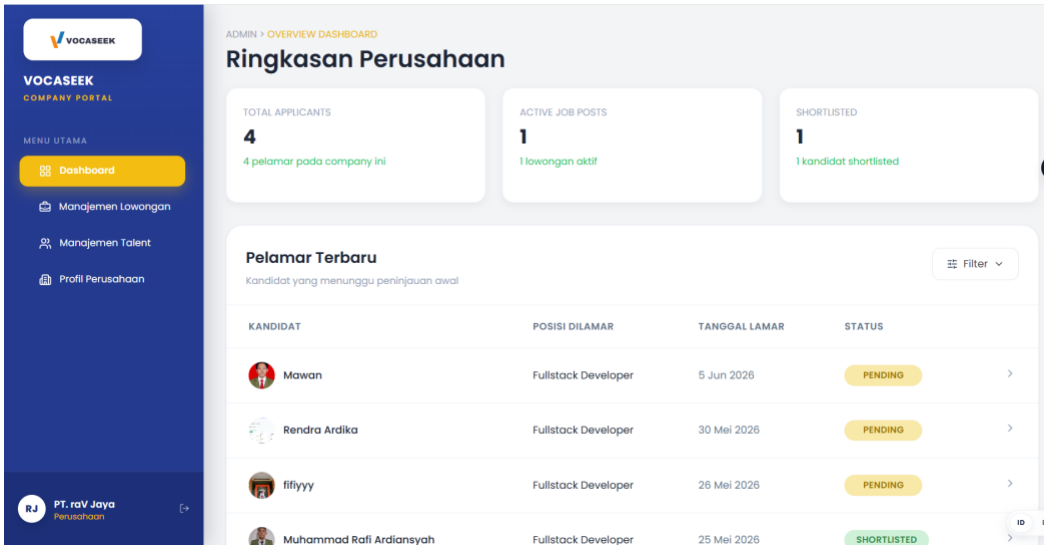

Fig. 8: Company dashboard and talent management module

5.6. Administration, Company Verification, and Invitation

Dashboard The super admin in Fig. 9 serves as the system's monitoring center. A summary of the number of talents, partners, vacancies, and activity helps administrators understand the platform's health before opening detailed modules. The *Talent Management menu* is used to review candidate data and documents. *Partner* Management is used to view companies, profile details, and legal documents. *Company Verification displays companies with pending or review* statuses , which can then be decided as active or rejected.

Separation of authority between *staff admins* and *super admins* is implemented on routes and controllers. *Staff admins* can perform initial checks and assist with *reviews , but final approval* or rejection actions are restricted to *super admins* . This restriction prevents sensitive decisions from being made by accounts with limited authority. Additionally, *super admins* manage administrators through user management and invitation mechanisms. Invitation *tokens* have an expiration date and a status that can be verified, used, resend, or revoked.

Administrator activation via invitation avoids creating an active account without the email address owner's consent. The invitee accesses the Admin Activate Account page, verifies *the token* , and then creates a password. This process is similar to a controlled registration flow and provides a better administrative footprint than manual credential sharing. Email delivery is done through *a queue* , so the invitation creation process doesn't depend on the duration of communication with the email server.

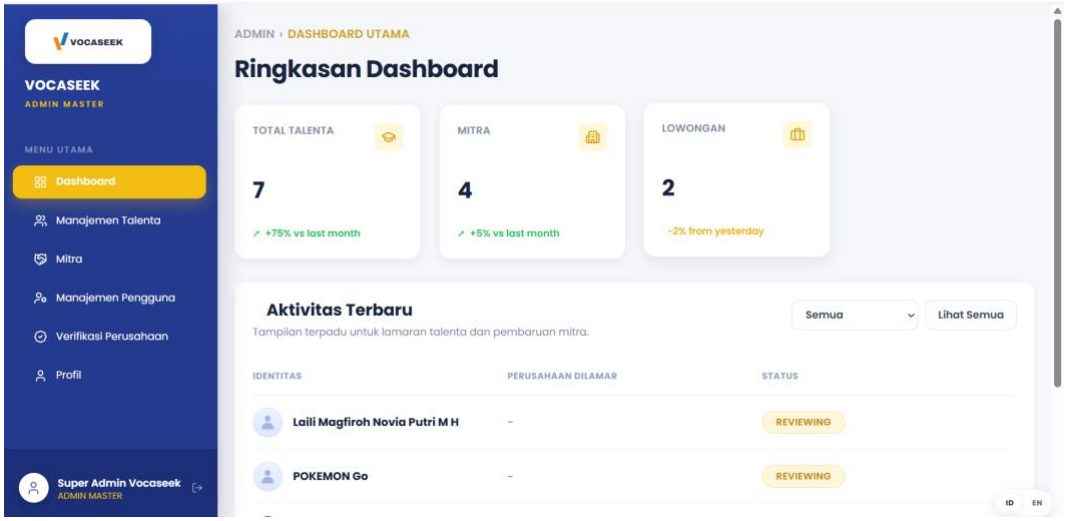

Fig. 9: Super administrator dashboard for system monitoring

5.7. Authentication, Localization, and API Documentation

The authentication module provides *internal registration* , *company registration* , login, logout, *email verification* , *forgot password* , *token* validation, *password reset* , and Google login through Laravel Socialite. New accounts are not immediately active until the email verification link is used. For *companies* , email verification only confirms ownership of the email address; portal access remains pending *approval. super admin* . This two-stage separation distinguishes the validity of the communication identity from the company's eligibility as a partner.

Laravel Sanctum is used to generate *bearer tokens* for SPAs. The official documentation explains the use of Sanctum for *single-page applications* and *token* -based APIs. Vocaseek prioritizes *bearer tokens* in *the middleware* to prevent browser sessions from causing identity confusion between portals. *The frontend* stores session data using different keys for *internal* , *company* , *staff admin* , and *super admin* . On logout, the server *token* is revoked and the client storage is cleared.

Language support is built on two layers. *The frontend* uses phrases.js as a global translator for static and partially dynamic text. *The backend* uses the SetLocale *middleware* , which reads values from the session, X-Locale header, query locale, or user's preferred locale. Preferences can be stored via /language or /preferences/language. This strategy allows specific *responses* or content to adapt to the language without duplicating the entire route.

API documentation uses L5 Swagger and OpenAPI. *Endpoints* are grouped into public, auth, *intern* , *company* , *company* talent, and admin. Documentation helps developers understand parameters, bearer authentication, *response structures* , and potential errors. OpenAPI provides a human- and tool-readable HTTP API description format, allowing documentation to be used for service contract exploration, testing, and maintenance.

6. Functional Testing and Discussion

6.1. Black Box Testing Results

Functional testing was performed on flows that directly impact the recruitment process. Each scenario used initial conditions, inputs, and expected outputs. *Internal testing results* showed that the main scenarios responded *and* changed data according to business rules. Table 3 summarizes the core testing. State transition testing of the web application was also used to verify changes in system behavior based on initial states and user actions.

Table 3: Functional test results for the main Vocaseek workflows

No.	Module	Test scenario	Expected result	Result
1	*Internal* registration	Submit valid account data	Pending registration is created and verification email is *queue* d	Pass
2	*Email verification*	Open a valid verification link	Email is marked verified and account can proceed according to *role*	Pass
3	*Company* registration	Submit *company* data and legal PDF documents	*Company* account is recorded with pending *partner* status	Pass
4	*Company* login	Login before final *approval*	Access is denied with information that *approval* is pending	Pass
5	Authentication	Login using valid email and password	*Role - scoped token* and user information are returned	Pass
6	*Role* protection	*Internal token* accesses an admin *endpoint*	*Request* is rejected by *Role* Check *middleware*	Pass

7	Profile update	Submit valid personal and academic data	Profile fields are updated and completion status recalculated	Pass
8	Document upload	Upload a valid CV PDF	File is saved and profile document path is updated	Pass
9	*Pre-test* prerequisite	Start test with incomplete profile	*Request* is rejected and user is directed to complete profile	Pass
10	*Pre-test* submission	Submit 20 answers within the active session	Answers and completion timestamp are saved	Pass
11	Duplicate test	Attempt to start test after completion	Second attempt is rejected	Pass
12	Job application	Apply with complete profile and finished test	A job application with initial status is created	Pass
13	Duplicate application	Apply to the same vacancy twice	Second application is rejected	Pass
14	Job posting	Active *company* submits valid vacancy data	Vacancy is created with the selected publication status	Pass
15	Candidate status	*Company* changes applicant status to INTERVIEW	Status is updated and notification process is triggered	Pass
16	*Company* verification	*Super admin* approves *review* ed *company*	*Partner* status becomes active and *company* can access its portal	Pass
17	Staff limitation	*Admin staff* attempts final *company approval*	Action is rejected because final authority belongs to *super admin*	Pass
18	Admin invitation	Invite new staff using valid email	Invitation *token* is created and activation email is *queue* d	Pass

The results in Table 3 show that key dependencies can be maintained on the server side. Applicants cannot skip profile completion and *pre-tests* , companies cannot publish jobs before they are active, and *admin staff* cannot make final decisions. Testing also verifies negative flows, such as incorrect *role access* , second tests, and duplicate applications. Negative testing is important because the success of a system is determined not only by its ability to accept valid input but also by its ability to reject conditions that violate business rules.

While all core scenarios produce appropriate output, these results are limited to functional verification in a development environment. Testing does not provide information on response time under high user load, resilience to email service failures, in-depth file upload security, accessibility, or user satisfaction levels. Therefore, the results should not be interpreted as a guarantee that the system is ready for all production conditions.

6.2. Discussion

The implementation of Vocaseek demonstrates how the internship recruitment process can be transformed from a collection of separate activities into a centralized *workflow* . In the manual process, students had to repeatedly submit IDs and documents, while companies compiled candidate lists separately. With Vocaseek, profile data is stored once and used for every application. This approach reduces duplication and provides a consistent source of candidate data. Companies can review the same data across multiple openings without requesting documents through multiple channels.

The second contribution lies in candidate readiness control. Applicants are not immediately allowed to apply after creating an account. The system checks profile completeness and *pre-test completion* . These rules provide structure for users and increase the likelihood of companies receiving reviewable data. However, overly strict rules can also be a barrier if the definition of completeness doesn't meet the needs of all student categories. Therefore, mandatory attributes should be configurable based on platform policy or job type.

The third contribution is tiered partner verification. *Company registration* requires legal documents and *super admin approval* . This process increases control over which companies can post vacancies. *Admin staff* assist with vetting without having *final* authority. This model clarifies responsibilities and reduces the risk of status changes by unauthorized accounts. However, document validity still depends on the thoroughness of *the reviewer* ; integration with official legal verification sources could be a further improvement.

From a technical perspective, separating React and Laravel facilitates parallel development of interfaces and APIs. React components are reusable, while Laravel provides *the middleware* , validation, storage, mail, and *queues* required by business processes. The REST API also opens up opportunities for developing other clients. However, the separate architecture increases the need for CORS management, environment configuration, *endpoint versioning* , schema synchronization, and error handling. Swagger/OpenAPI documentation is essential for reducing interpretation differences between *the frontend* and *backend* .

role - scoped authentication strategy provides a solution to the possibility of *token* exchanges between portals. This is relevant because Vocaseek has multiple login pages and *dashboards* . While key separation on *the frontend* is helpful, security shouldn't rely on client storage. *Role checks* on *the backend remain mandatory. Future development should evaluate token* expiration , *token* rotation , activity logging, rate limiting, file MIME validation, malware scanning, and personal data protection.

internalization feature expands user access and indicates that language is treated as a system preference, not just a text replacement on a single page. *Middleware The backend* can read locales from multiple sources, while phrases.js handles the *frontend text* . The challenge is translation consistency across dynamic text, validation messages, email notifications, and user-inputted data. Centralized dictionary management and testing for each locale are necessary to prevent language mix-ups.

Compared to recruitment system research that focuses on candidate ranking or employee referrals, Vocaseek emphasizes comprehensive *workflows* and collaboration between actors. The system does not yet use recommendation algorithms or artificial intelligence. This decision ensures that the initial implementation remains explainable and controllable. Studies of AI in recruitment warn of challenges with bias and transparency; therefore, if automated recommendations are added, the model must be accompanied by explanations, fairness evaluations, and human decision-making mechanisms.

Limitations of this study include the lack of usability testing with a representative number of respondents, the lack of comparison of processing times before and after system use, and the lack of comprehensive performance and security testing. Furthermore, *the pre-test* has not been psychometrically tested. This article therefore presents the results of design, implementation, and functional verification, not long-term organizational effectiveness. Future research could utilize the System Usability Scale, *User Acceptance Testing* , load testing, security audits, and analysis of the impact on recruitment duration and candidate suitability.

7. Conclusion

Vocaseek has been designed and implemented as a web-based internship recruitment and student talent management information system using React, Vite, Laravel REST API, Laravel Sanctum, and MySQL. The system integrates registration, email verification, profile and document management, *pre-testing* , job search, application, candidate review, company verification, administrator management, and language preferences in a single platform.

The implementation of four *roles* results in a clear division of responsibilities. Applicants manage data and application readiness; companies manage vacancies and candidates; *admin staff* assist with *review* ; and *super admins* exercise final authority and system administration. *Backend business rules* ensure that profiles and *pre-tests* are prerequisites for applications, companies must be active before creating vacancies, duplicate applications are rejected, and final *approval* is restricted to *super admins* .

Black-box testing results for 18 key scenarios indicate that the core flow produces appropriate output in the development environment. These findings demonstrate that the developed architecture and modules can support a centralized and traceable recruitment *workflow* . However, the results do not yet include usability measurements, high-load performance, comprehensive security, or *pre-test psychometric validity* .

Further development will focus on testing with real users, strengthening security and activity audits, improving accessibility, integrating document verification, and measuring the system's impact on the time and quality of the recruitment process. Job or candidate recommendation features can be added once sufficient data and a fairness and transparency evaluation mechanism are available.

Acknowledgement

The authors would like to thank all parties who have provided support, guidance, and assistance during the research, development, and preparation of this scientific article. Special thanks are extended to Ardhon Rakhmadi, S.Tr.T., M.Kom. and Mohamad Irwan Afandi, ST, M.Sc. for their direction, input, and guidance during the preparation of this article. The authors would also like to thank Mr. Ageng Permadi as CEO of PT Empat Beruang Perkasa and Deputy Director of PT. Lima Benua Koneksindo for their support, opportunities, and valuable experience during the development of the Vocaseek platform. Appreciation is also extended to all parties who have contributed to the development of the system and the completion of this research. Hopefully, this article can provide benefits for the development of science and become a reference for further research.

References

[1] K. Mwelwa and AS Mawela, "Effectiveness of internships as pedagogical practices in promoting employability skills among students graduating in selected socialscience degree programs in Zambia," International Journal of Educational Methodology, vol. 7, no. 4, pp. 649–668, 2021, doi: 10.12973/ijem.7.4.649.

[2] J. S. Black and P. van Esch, "AI-enabled recruiting: What is it and how should a manager use it?," Business Horizons, vol. 64, no. 4, pp. 513–524, 2021, doi: 10.1016/j.bushor.2021.02.001.

[3] NN Thang, HT Nguyen, and TN Tran, "Employer branding, organization's image and reputation, and job seekers' intention to apply," Humanities and Social Sciences Communications, vol. 11, 2024, doi: 10.1057/s41599-024-02998-3.

[4] AL Hunkenschroer and A. Kriebitz, "Is AI recruiting (un)ethical? A human rights perspective on the use of AI for hiring," AI and Ethics, vol. 3, no. 1, pp. 199–213, 2023, doi: 10.1007/s43681-022-00166-4.

[5] Z. Chen, "Ethics and discrimination in artificial intelligence-enabled recruitment practices," Humanities and Social Sciences Communications, vol. 10, 2023, doi: 10.1057/s41599-023-02079-x.

[6] MM Alshurideh, B. Al Kurdi, H. Alzoubi, et al., "The Effect of Talent Management on Organizational Performance," Information Sciences Letters, vol. 11, no. 2, pp. 543–551, 2022, doi: 10.18576/isl/110228.

[7] M. Bogner, M. Zimmermann, and F. Wagner, "An Empirical Study of the Implementation of REST APIs and their Design Rules," Information and Software Technology, vol. 134, 2021, doi: 10.1016/j.infsof.2021.106567.

[8] OpenAPI Initiative, "OpenAPI Specification Version 3.1.0," 2021.

[9] React Team, "React Documentation," Meta Platforms Inc., 2024.

[10] SR Jan, STU Shah, Z. Johar, Y. Shah, and F. Khan, "An Innovative Approach to Investigate Various Software Testing Techniques and Strategies," International Journal of Scientific Research in Science and Technology, vol. 8, no. 2, pp. 682–689, 2021.