

UIT2739 – FULL STACK DEVELOPMENT

A PROJECT REPORT

On

INTERN TRACKING SYSTEM

Submitted by

SNEHA SENTHIL (3122225002132)

SUBASRI G S (3122225002137)

TAFHY E (3122225002144)



Department of Information Technology

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Rajiv Gandhi Salai (OMR), Kalavakkam – 603 110

NOVEMBER 2025

**SRI SIVASUBRAMANIYA NADAR COLLEGE OF
ENGINEERING**



Department of Information Technology

CERTIFICATE

Certified that this project titled “**INTERN TRACKING SYSTEM**” is the bonafide work of “**Sneha Senthil (3122225002132), Subasri G S (3122225002137), Tafhy E (3122225002144)** and is submitted for project review on **27 November 2025**.”

Place : Kalavakkam

Date :

Internal Examiner

TABLE OF CONTENTS

S.NO.	TITLE	PAGE NO.
1.	PROJECT OVERVIEW	1
2.	PROJECT REQUIREMENTS	2
	2.1 Functional Requirements	2
	2.2 Non-Functional Requirements and Use cases	8
3.	TECHNICAL DETAILS	9
	3.1 Tech Stack	9
	3.2 Architecture Diagram	11
	3.3 Design Patterns	13
4.	CONCLUSION	15

1.PROJECT OVERVIEW

The project is a comprehensive full-stack Internship Management Web Application—InternTrack—developed using the MERN stack (React.js, Node.js, Express.js, and MongoDB), Google Drive API integration, and OCR-based document verification. The platform is designed to streamline the way academic institutions handle internship registration, verification, storage, and approval, thereby offering an organized digital solution for students, coordinators, and faculty members. It unifies authentication, internship submission, document management, coordinator review tools, and cloud-based storage into a single, scalable system.

The core functionality centers on enabling students to submit internship details and upload associated documents through an intuitive web interface. Using React.js for interactive form handling and API communication, the application allows users to enter details such as internship type, duration, company name, stipend, location, and dates. All entries are stored securely in MongoDB, while files are uploaded to Google Drive using dedicated backend API routes. The system also integrates Python-based Tesseract OCR to automatically verify document authenticity by matching names, registration numbers, and company keywords. Unverified files are categorized and stored as unknown documents for further review.

A central aspect of the platform is its role-based access system, which differentiates between students and coordinators. Students can create accounts, manage profiles, upload documents, and monitor the approval status of each internship entry. Coordinators, on the other hand, gain access to powerful management tools including full internship data review, approval/rejection workflows, filter-based searching, and the ability to add comments or request corrections. This ensures a continuous, structured verification pipeline while eliminating manual paperwork and fragmented communication.

The system also integrates several automated backend processes to enhance reliability and user experience. By leveraging Node.js with Express.js, the backend handles secure authentication, request validation, Drive API calls, and OCR verification flows. MongoDB acts as the central NoSQL database where all user credentials, internship metadata, and document paths are stored using a clean, scalable schema. Each update made by a coordinator immediately reflects in the student's dashboard, ensuring real-time visibility of status changes and reducing delays commonly seen in traditional administrative workflows.

The user interface is built using a component-based React architecture enhanced with React Hooks and controlled forms for smooth state management. Navigation is handled through dedicated routes and protected pages, keeping student and coordinator views strictly separated. The UI incorporates a clean, modern theme featuring dashboards, dynamic quotes, responsive tables, and visually structured forms, all optimized for ease of use across desktop and mobile environments. Features such as persistent login, live document status display, interactive internship creation, and structured multi-step workflows contribute to a polished and efficient user experience.

From a technical standpoint, the platform demonstrates strong implementation of full-stack concepts such as RESTful API design, cloud-storage integration, OCR-based data validation, role-based access control (RBAC), and modular frontend-backend communication. The integration with Google Drive ensures scalable and secure document storage, while MongoDB provides flexible schema handling for dynamic internship entries. The system adopts industry-standard practices such as access middleware, encrypted communication with cloud APIs, clean separation of concerns, and scalable component structures suitable for large-scale institutional deployment.

In terms of real-world application, the platform offers a robust solution for academic departments aiming to manage internships efficiently. It supports multiple internship entries per student, securely archives completed submissions, preserves document records, and provides coordinators with tools for evaluating, filtering, and exporting internship data. By digitizing approval cycles and integrating document intelligence through OCR, the platform minimizes human errors, prevents data duplication, and maintains a centralized repository for institutional use.

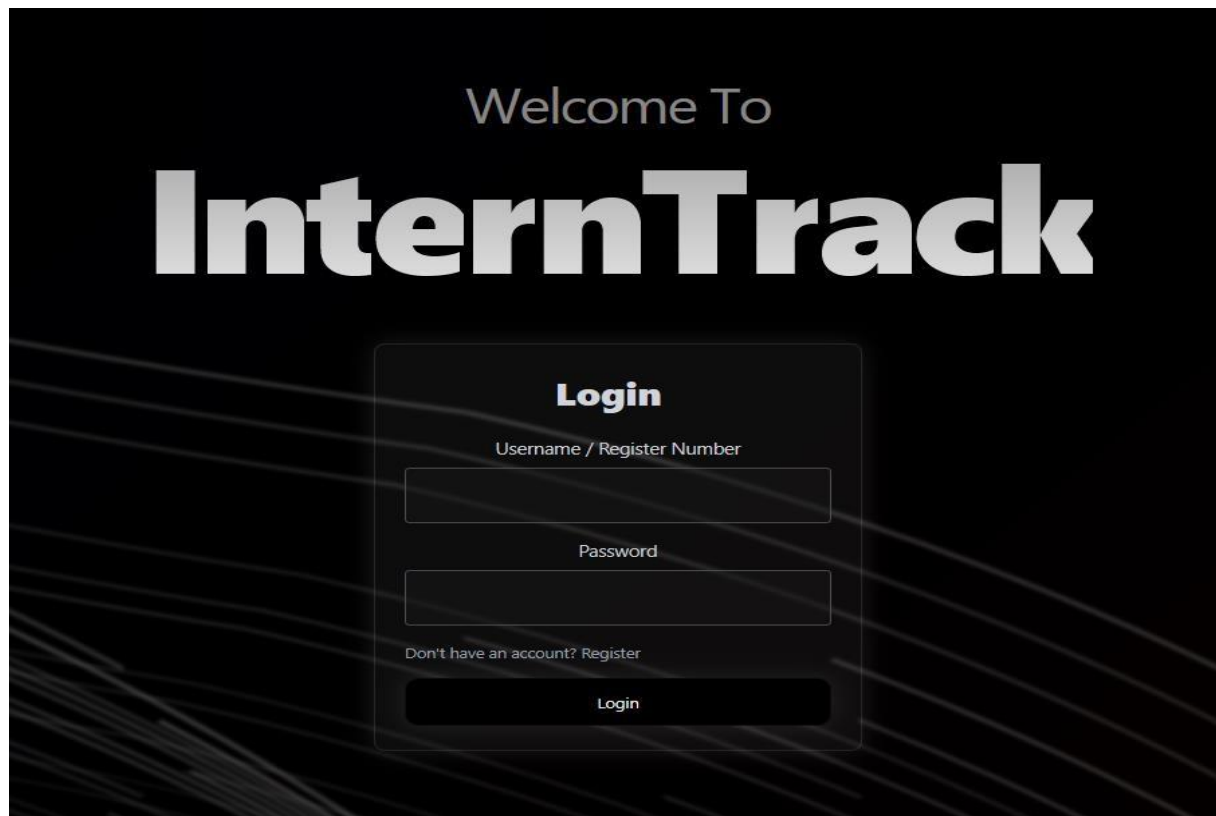
Overall, this project highlights strong proficiency in full-stack development, cloud integration, automated verification, and modern UI/UX practices. It showcases the ability to build scalable, secure, and production-ready web systems tailored for educational institutions. InternTrack demonstrates advanced technical understanding across backend architecture, frontend interactivity, API orchestration, and cloud-based storage, making it a complete and reliable solution for internship management and departmental workflow automation.

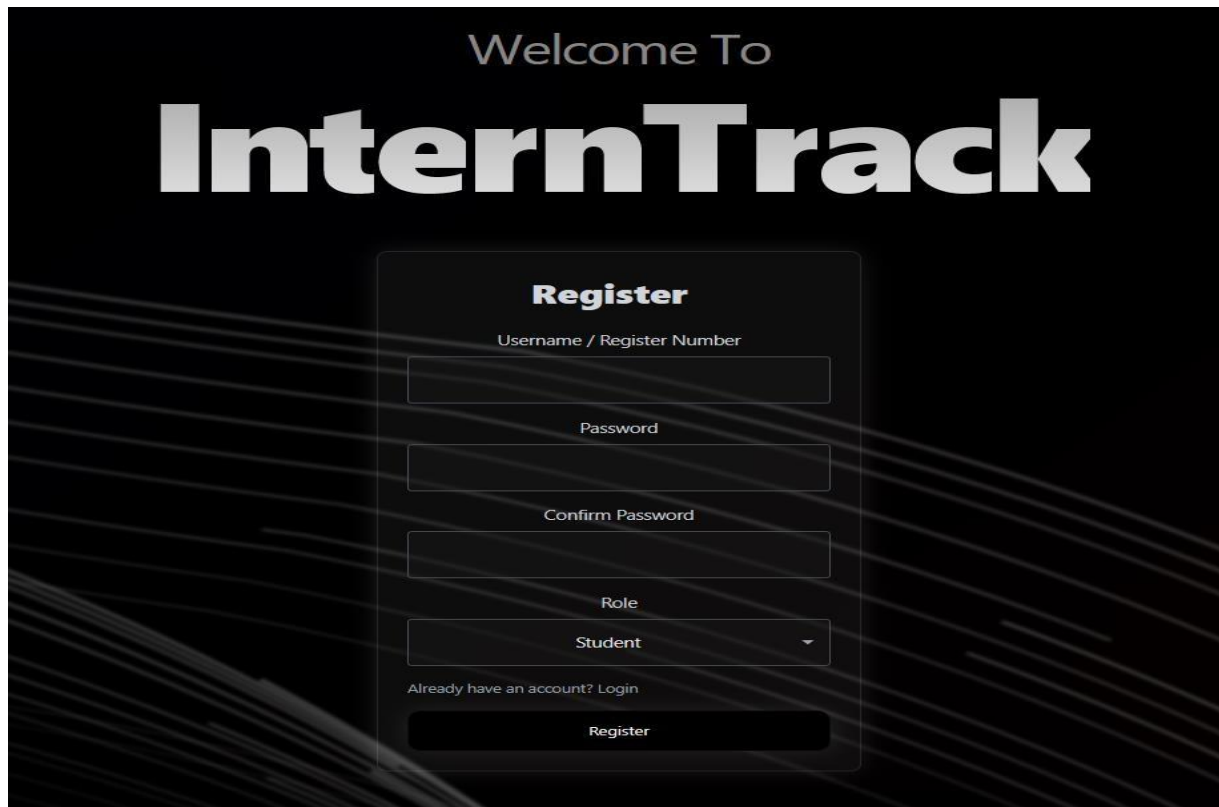
2. Project Requirements

2.1. Functional Requirements

User Authentication

The system must provide secure user registration and login for both students and coordinators. Each user should authenticate using unique credentials such as register number/username and password. Authentication must be mandatory before accessing any internship-related features. Role-based access control (RBAC) must ensure that students and coordinators have distinct privileges, preventing unauthorized access to restricted pages or operations.





Welcome To

InternTrack

Register

Username / Register Number

Password

Confirm Password

Role

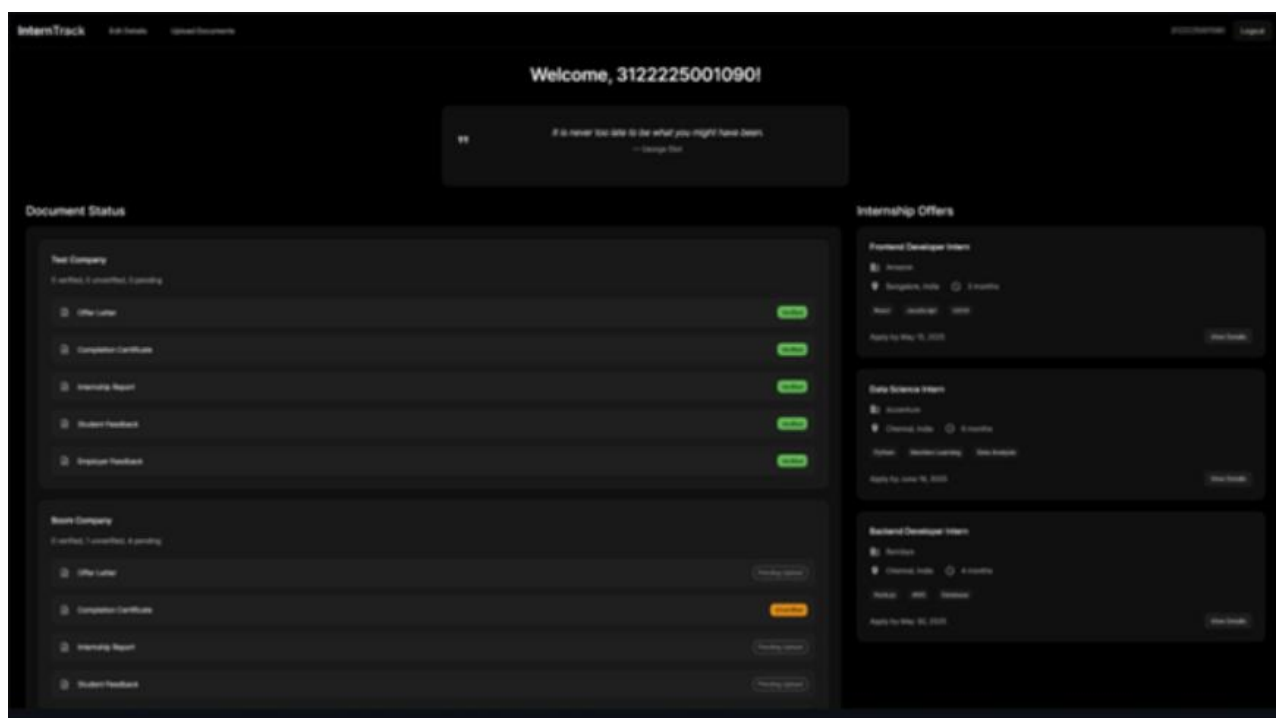
Student

Already have an account? [Login](#)

Register

Student Profile and Internship Management

Students should be able to create and update their personal and academic profiles, including details such as register number, name, contact information, department, and year. The platform must allow students to submit internship information including company name, internship type, duration, location (India/Abroad), start and end dates, stipend, and placement source. Students must be able to add multiple internships, view previous submissions, and edit details before coordinator approval.



InternTrack Add Details Upload Documents 3122225001090! Logout

Welcome, 3122225001090!

It is never too late to do what you might have been thinking about

Document Status

Test Company
 1 verified, 1 unverified, 1 pending

- Offer Letter Upload
- Completion Certificate Upload
- Internship Report Upload
- Student Feedback Upload
- Employer Feedback Upload

Bank Company
 1 verified, 1 unverified, 1 pending

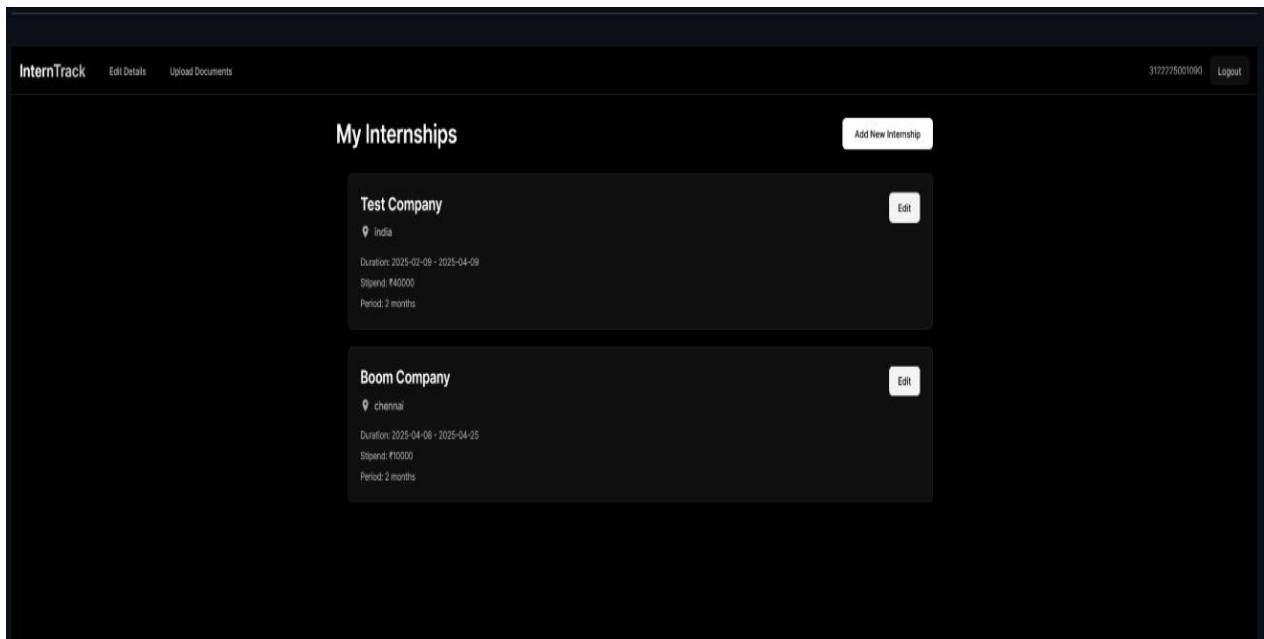
- Offer Letter Upload
- Completion Certificate Upload
- Internship Report Upload
- Student Feedback Upload

Internship Offers

Frontend Developer Intern
 1 verified, 1 unverified, 1 pending
 Bangalore, India 3 months
 Start: 2023-07-01 End: 2023-09-01
 Apply for May 15, 2023 View Details

Data Science Intern
 1 verified, 1 unverified, 1 pending
 Chennai, India 4 months
 Start: 2023-07-01 End: 2023-11-01
 Apply for June 15, 2023 View Details

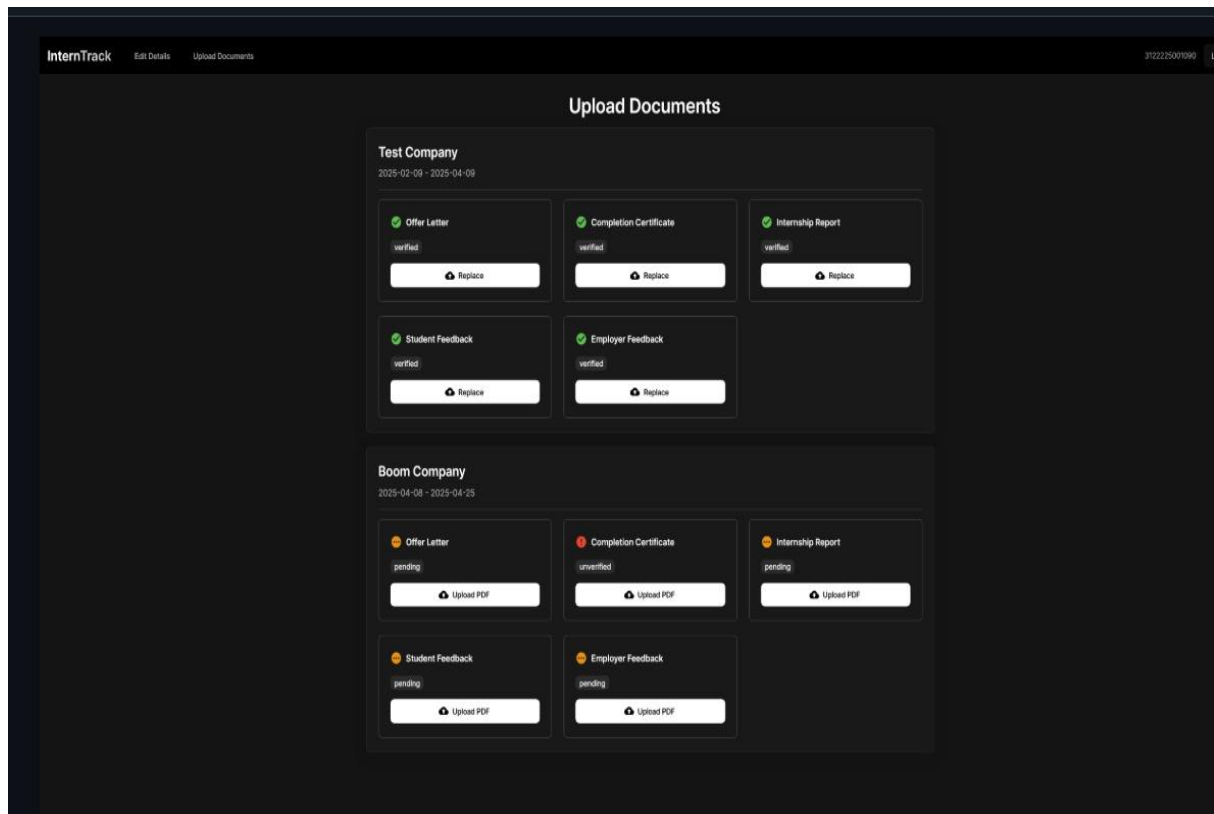
Backend Developer Intern
 1 verified, 1 unverified, 1 pending
 Chennai, India 4 months
 Start: 2023-07-01 End: 2023-11-01
 Apply for May 15, 2023 View Details



The screenshot shows the 'Add New Internship' form. It has a title 'Add New Internship' and is divided into two main sections: 'Personal Information' and 'Internship Information'.
Personal Information:
 - Register Number*: 312225001090
 - Full Name*:
 - Mobile Number:
 - Section:
Internship Information:
 - Obtained Internship: Yes (dropdown)
 - Internship Period (in months):
 - Start Date: dd/mm/yyyy (calendar icon)
 - End Date: dd/mm/yyyy (calendar icon)
 - Company Name*:
 - Placement Source:
 - Stipend (₹.):
 - Internship Type:
 - Location:
 At the bottom right, there are 'Cancel' and 'Add Internship' buttons.

Document Upload and Verification

The system must enable students to upload internship-related documents such as offer letters, completion certificates, and supporting proofs through Google Drive integration. Uploaded documents should be automatically verified using OCR (Tesseract) to match student name, register number, and company keywords. Unverified files should be flagged and stored as "unknown documents" until further review. All documents must be stored in uniquely generated Google Drive folders organized per student.

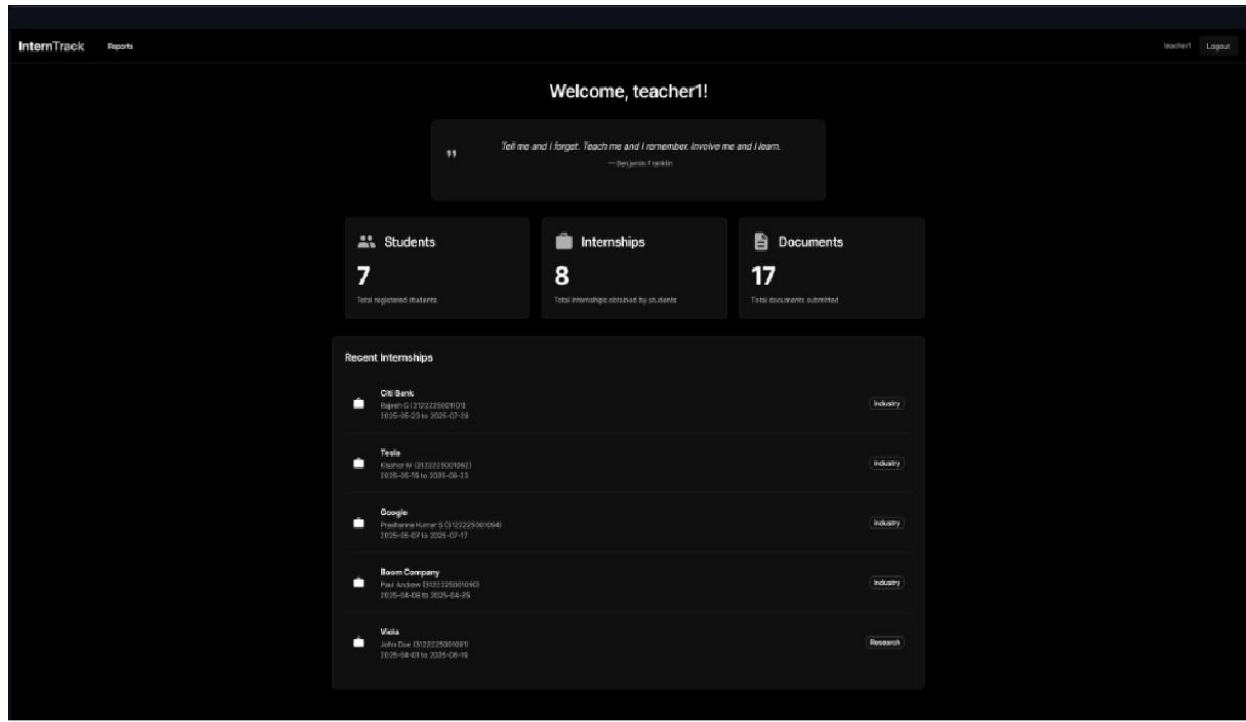


Internship Status Tracking

Students must be able to view the status of their internship submissions in real time. The system should display statuses such as **Pending**, **Approved**, or **Rejected**, ensuring transparency and clarity. Any change made by the coordinator must instantly reflect in the student's dashboard, enabling quick follow-up actions or corrections.

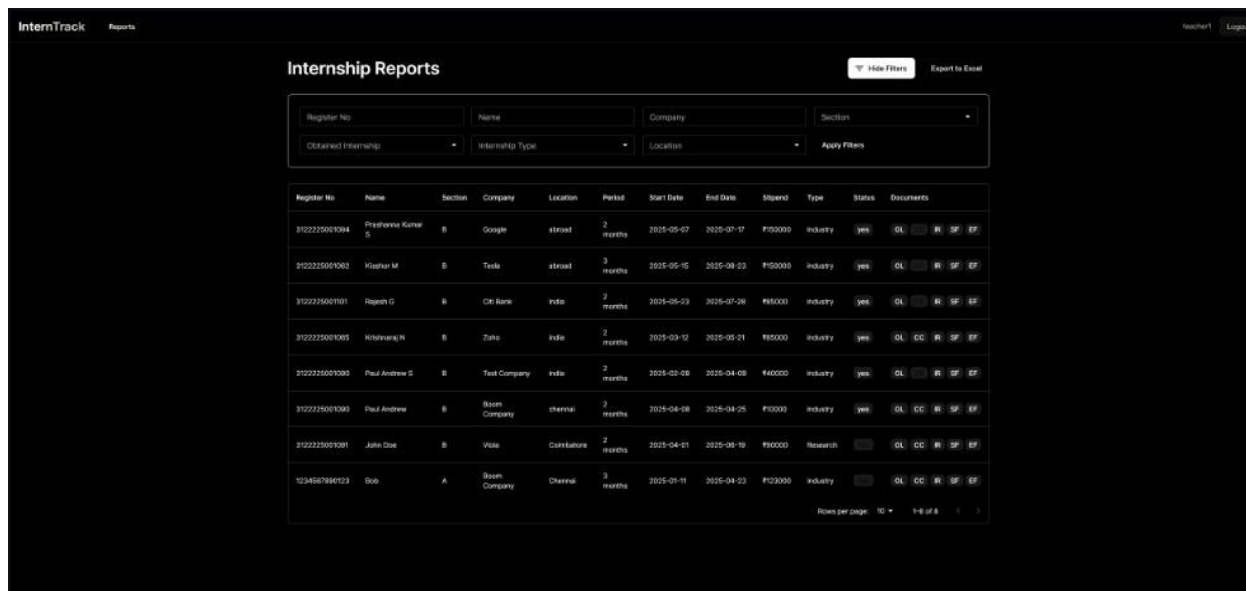
Coordinator Review and Approval System

Coordinators must have a dedicated dashboard where they can review all internship submissions. They should be able to view details submitted by students, assess uploaded documents, approve or reject records, add comments, and request corrections. The system must allow coordinators to edit student profiles if necessary. This ensures a structured, centralized workflow for managing all internship evaluations within a department.



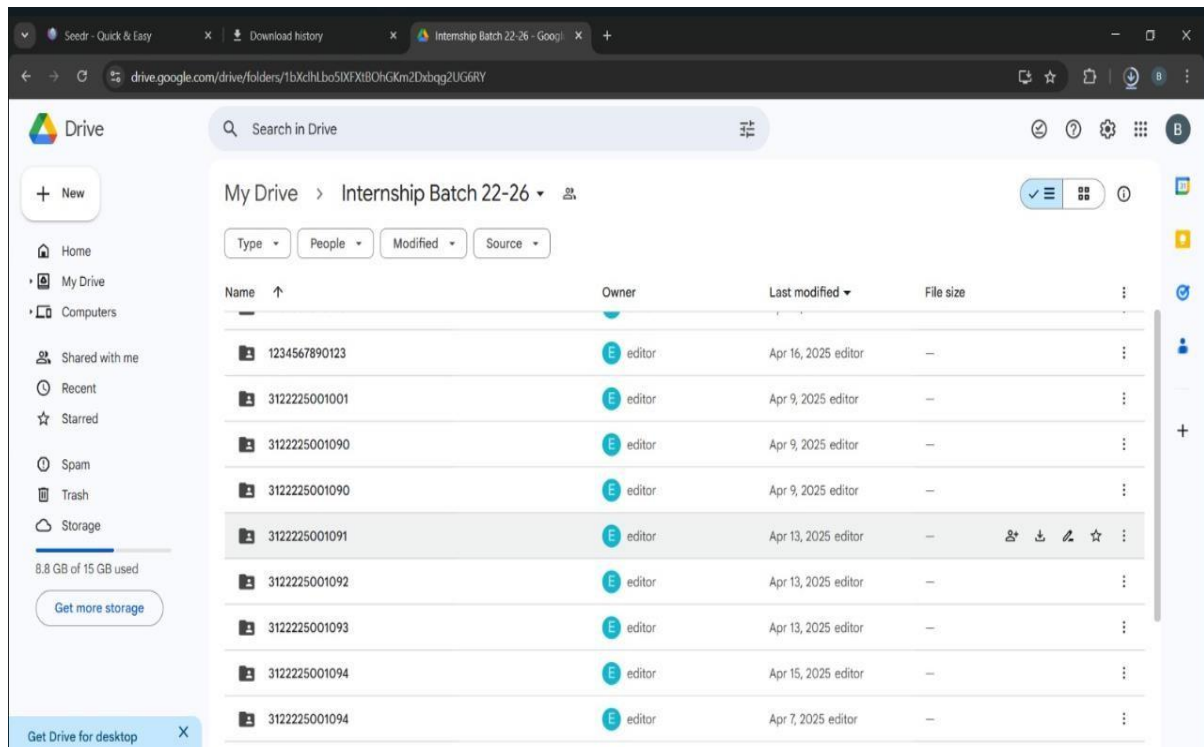
Filter, Search, and Reporting Tools

The system must allow coordinators to filter student internship records based on parameters such as company name, internship type, status, register number, and student name. A dedicated reports page should display internship data in a structured table format, with options to export filtered results to CSV for administrative use. This supports efficient data handling and decision-making.



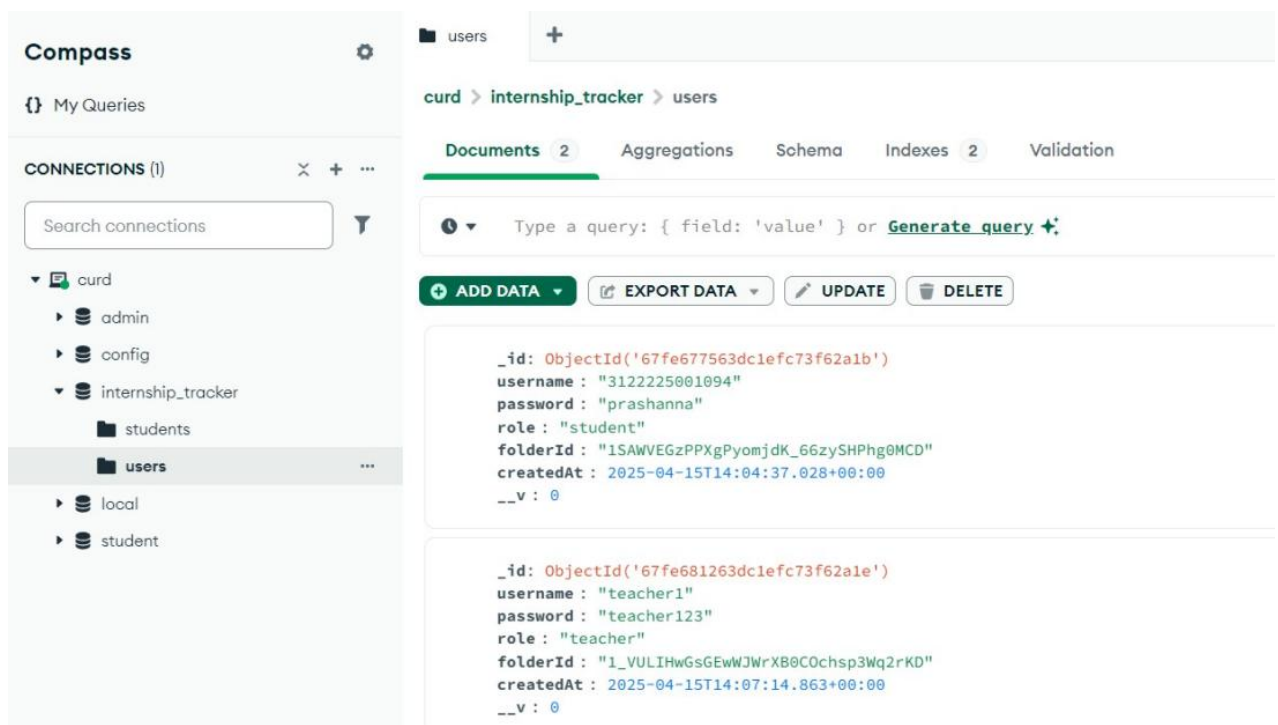
Cloud Storage Integration (Google Drive API)

All internship documents must be stored securely using Google Drive API integration. The backend should automatically create separate folders for each student and manage upload paths for all submitted documents. Coordinators must be able to access these documents directly through Drive links. The system must ensure that file organization is consistent, secure, and easily accessible.



Backend Processing and Data Management

The backend should handle all API requests using Node.js and Express.js. It must validate user inputs, enforce access control, perform document verification, and communicate with MongoDB for data storage. CRUD operations must be supported for users, internship entries, and document metadata. The backend must also manage Drive API actions such as creating folders, uploading files, and retrieving file links.



2.2 Non-Functional Requirements

Performance

The platform must deliver fast response times for data retrieval, profile updates, and internship submissions. Google Drive uploads and OCR verification should complete efficiently without causing delays. The React frontend must remain smooth and responsive even during frequent updates.

Security

The system must ensure secure handling of user data through authenticated routes, role-based access, and controlled Drive API operations. All sensitive actions such as approvals, data editing, and uploads should be restricted to authorized users. The backend must validate inputs to safeguard against vulnerabilities such as injection attacks.

Usability and Accessibility

The interface must be simple, intuitive, and suitable for both students and coordinators. Navigation across login, dashboard, submissions, and reports must be seamless, with clear instructions and consistent UI components. Students should be able to submit internships easily, and coordinators must be able to filter and review records without difficulty.

Responsiveness

The platform must adapt to desktops, laptops, and mobile devices. Each page—including dashboards, forms, tables, and upload screens—should automatically adjust layout and size for optimal user experience across all screen dimensions.

Use case

The InternTrack platform enables students and coordinators to efficiently manage internship submissions, document verification, and approval workflows through a unified full-stack web application. The main use case begins when a user logs into the system using secure authentication via the dedicated login interface. Once authenticated, students are taken to a personalized dashboard where they can view their internship status, browse previously submitted entries, and access options to submit new internship details. If a student wishes to register a new internship, they can navigate to the submission form to enter personal information, company details, internship duration, stipend, and other required fields before uploading necessary documents through Google Drive integration.

Students can continue managing their profile by editing previously submitted entries, updating incorrect details, or re-uploading documents before coordinator approval. Each internship submission is tagged with a status—Pending, Approved, or Rejected—allowing students to track their progress in real time. Document uploads undergo OCR verification using Tesseract to ensure that filenames, student identifiers, and company information match the submitted details. Unverified documents are automatically placed under an “unknown” category for further review, ensuring data accuracy and maintaining integrity throughout the verification process.

On the coordinator side, the use case expands with tools designed for streamlined evaluation. After logging in, coordinators can access their dashboard displaying key statistics such as total students, number of internships, and total uploaded documents. They can then browse the internship reports page, where all submissions are presented in a structured table format. Coordinators can filter records using parameters such as student name, register number, company name, internship type, and approval status. From here, they can open individual submissions, check uploaded Google Drive documents, approve or reject entries, request corrections, or update student information if required. All actions are handled through the backend API built with Node.js and Express.js, ensuring consistent validation and secure data processing.

Every update—status changes, document corrections, or profile edits—is stored in MongoDB and reflected instantly on the student’s dashboard. The platform ensures smooth synchronization between frontend components and backend APIs, creating a responsive user experience without manual page refreshes. By combining a structured approval workflow, cloud document management, real-time status tracking, and a clean React-based interface, InternTrack supports academic departments in maintaining organized internship records while simplifying the submission and verification process for students.

3. TECHNICAL DETAILS

3.1 Techstack

The technology stack selected for this project represents a modern, industry-standard approach to full-stack development, combining reliable and widely adopted technologies that seamlessly work together to deliver a secure, scalable, and efficient internship management system. This stack enables end-to-end development using JavaScript, reduces context switching, supports modular integration, and ensures high performance for real-world academic workflows that involve frequent data submissions, document uploads, and validation checks. The architecture supports cloud-based document handling, dynamic frontend rendering, and a flexible backend API design suitable for long-term expansion.

Backend Technologies

On the backend, Node.js serves as the primary runtime environment, offering an asynchronous, event-driven architecture that efficiently handles concurrent operations such as authentication, internship form submissions, document uploads, and approval workflows. The application uses Express.js as the backend framework, providing a clean and minimalist structure for defining RESTful routes, validating incoming data, managing middleware, and implementing secure access control for both student and coordinator roles. Express simplifies the organization of business logic through its routing and modular architecture, allowing the backend to remain lightweight while supporting complex interactions across multiple endpoints.

For persistent data storage, MongoDB functions as the primary NoSQL database. It stores user credentials, student profiles, internship submissions, document metadata, approval statuses, and coordinator updates in flexible JSON-like documents. MongoDB’s schema-less nature allows internship entries of varying formats to coexist, enabling students to submit multiple internships without structural constraints.

Its indexing capabilities ensure fast filtering and searching based on company name, internship type, approval status, register number, and other attributes. This design supports smooth performance even as the number of records grows across academic cycles.

Cloud document storage is implemented through integration with Google Drive API. Whenever a student uploads offer letters or certificates, the backend automatically creates a dedicated Drive folder for the student and stores all associated files within it. The system uploads documents programmatically, generates secure links, organizes them under relevant categories, and maintains consistency between database metadata and Drive storage. This cloud-native workflow eliminates the need for local file storage and ensures long-term accessibility of all documents.

Document authenticity is reinforced using the Tesseract OCR engine. OCR is employed to extract text from uploaded files and compare the extracted content with submitted internship data such as student name, register number, and company identifiers. Documents that fail to match expected patterns are flagged as unknown, allowing coordinators to review them manually. This automated verification layer reduces errors, prevents incorrect submissions, and significantly improves the reliability of internship validation.

Security and access control rely on role-based authorization implemented through backend middleware. Every protected route undergoes validation, ensuring that only authenticated users can view dashboards, submit internships, upload files, or perform approval actions. Coordinators have elevated privileges that allow them to view, filter, edit, or validate student submissions, while students are restricted to their own records. This structured separation ensures data integrity and prevents unauthorized operations.

Frontend Technologies

The frontend of the application is built using React.js, a powerful JavaScript library designed for creating dynamic, component-based user interfaces. React's declarative structure allows each part of the system—such as login pages, dashboards, forms, tables, and document upload modules—to be encapsulated into reusable components that simplify development and maintenance. Hooks are used to manage state transitions, store user information, track internship status updates, handle form interactions, and respond to backend validation results. React's rendering model ensures smooth UI updates and efficient performance even when data changes frequently.

Routing and navigation within the frontend provide structured transitions between student and coordinator pages. The routing system manages private pages, ensures role-based redirection, and prevents unauthorized users from accessing restricted components. This structure supports intuitive navigation across login, profile editing, internship submissions, document upload screens, and coordinator dashboards.

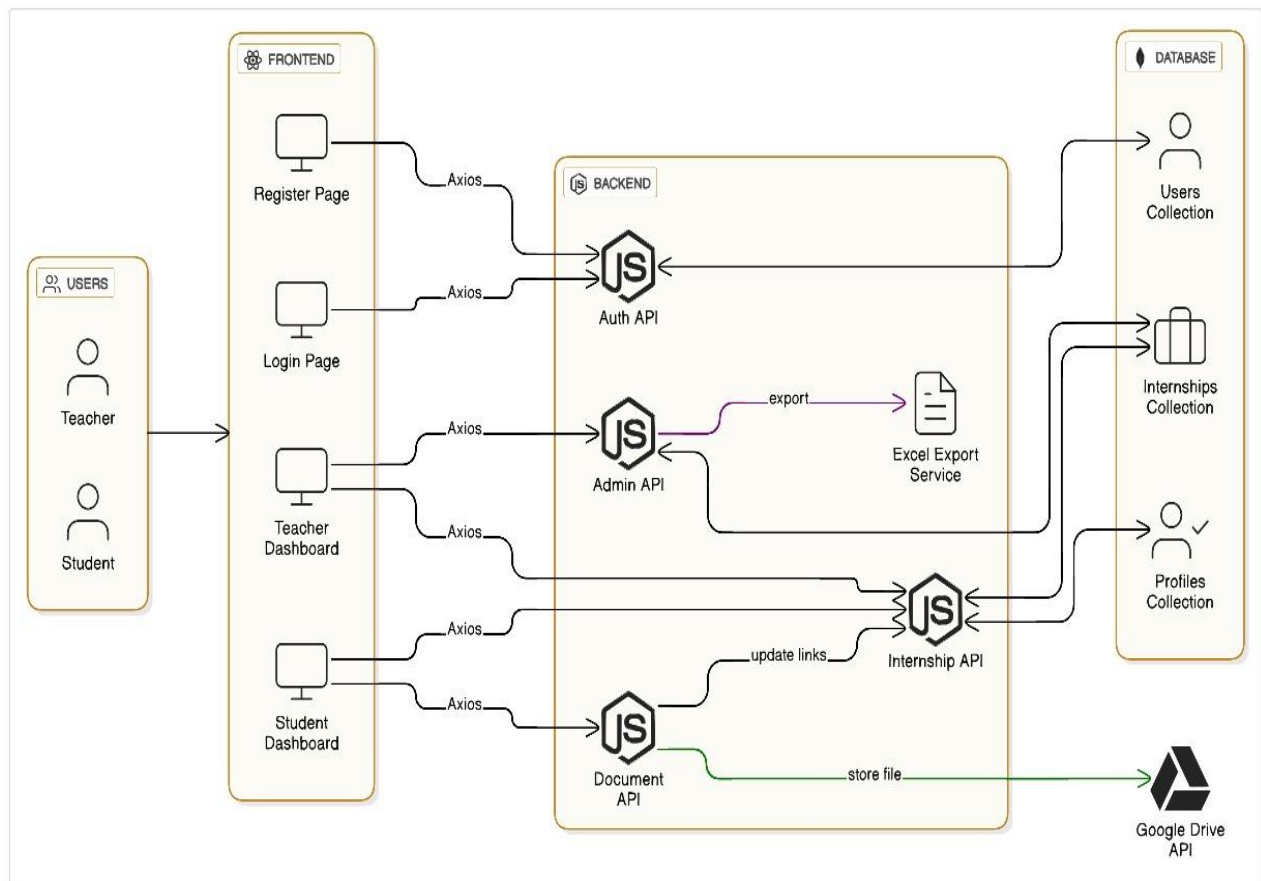
For API communication, Axios is used as the HTTP client, providing a consistent interface for sending requests to the backend. Axios handles JSON conversion automatically, supports interceptors for attaching authentication details, manages error responses, and simplifies asynchronous workflows involved in submitting forms and retrieving internship records. This enables reliable communication between React components and backend endpoints.

The user interface is designed to be responsive across devices, ensuring usability on desktops, laptops, and mobile screens. Components render dynamically based on the user's role and application state, displaying features such as approval status, edit options, filters, and document upload buttons only where appropriate. This ensures a clean and organized interface tailored to both students and coordinators.

The combination of these frontend technologies results in a smooth, interactive, and accessible user experience that supports academic workflows with clarity and efficiency. React's flexibility, combined with structured routing and robust API handling, enables the platform to deliver real-time updates, responsive layouts, and intuitive navigation for all users.

3.2 Architecture Diagram

The architecture of this internship management application follows a structured multi-tier architecture consisting of the presentation layer, application layer, and data layer, ensuring clear separation of concerns and well-defined communication flows between each part of the system. This architectural model enhances maintainability, scalability, and modular development while allowing each subsystem—frontend, backend, database, and external services—to evolve independently without affecting the others. The integration of cloud-based storage and OCR verification creates a hybrid workflow that combines traditional request–response patterns with external service interactions for document validation and file management.



This diagram illustrates the high-level architecture, showing the client layer running the React application, the server layer powered by Node.js and Express, the MongoDB cloud database that stores user and internship information, and the external services including Google Drive API for document storage and an OCR module for text extraction and verification. Data moves between layers using standard HTTPS REST API calls, supported by Axios on the frontend, while internal backend modules communicate with Google Drive and the OCR engine to complete document processing workflows.

The client layer represents the presentation tier, consisting of the React application running inside the user's browser. This layer is responsible for rendering the user interface through modular React components, capturing user input via event handlers, managing state with React hooks, and sending HTTP requests to the backend for authentication, internship submission, document uploads, status retrieval, and coordinator approvals. The React application is organized into several routed pages such as Register Page, Login Page, Student Dashboard, and Teacher Dashboard, each connected to the backend through Axios. UI components handle form validation, data binding, and dynamic rendering, ensuring a responsive and intuitive experience for both students and coordinators.

The diagram shows how user actions initiate from the React UI, trigger Axios requests, and are routed to the appropriate backend API within the Node.js/Express server. The backend layer exposes multiple RESTful APIs including the Auth API for login and registration, the Admin API used by coordinators for record review and export, the Internship API for creating and managing internship entries, and the Document API for uploading and verifying supporting documents. These APIs act as the central business logic engines, performing validations, applying rules, updating records, and orchestrating document-related workflows. When required, the backend communicates with the Google Drive API to store uploaded files and generate unique storage links, and with the OCR module to process documents and extract text for verification. Each response flows back to the client, updating UI state and reflecting changes such as approval status, verification results, or updated document links.

The server layer implements the core application tier using Node.js and Express. This layer handles authentication logic, maps endpoints to controller functions, applies middleware for access control and input validation, manages coordinator operations such as filtering and exporting internship data, and integrates with external modules for Google Drive and OCR processing. Dedicated APIs ensure that user operations remain isolated by role, with students allowed to submit and edit their own internships, and coordinators granted extended privileges such as approval, rejection, and data export. An Excel export service is also connected to this layer to generate structured reports for departmental analysis.

The data layer consists of MongoDB, a cloud-hosted NoSQL database used for persistent storage of all application data. The database includes collections such as the Users Collection for storing login credentials and roles, the Internships Collection for recording details such as company, duration, type, location, dates, and status, and the Profiles Collection for student profile data. Each API interacts with these collections to store or retrieve relevant information, ensuring consistent data management across the system. MongoDB's flexible document structure supports dynamic, multi-entry internship submissions and enables efficient filtering by coordinators through indexed fields.

Together, these layers form a cohesive architecture capable of handling large-scale internship management operations with reliability, security, and clarity. The combination of a modern React frontend, a modular Node.js backend, a scalable NoSQL database, and cloud-based document handling makes the application robust and adaptable for future enhancements.

4.3 Design Patterns

The implementation of the InternTrack application incorporates several well-established design patterns that improve code organization, reuse, testability, and scalability. These patterns provide proven solutions to recurring engineering problems across the frontend, backend, and integration layers, and they help maintain a clean separation of concerns between document handling, verification logic, user flows, and real-time communication.

MVC Pattern (API-oriented)

The Model–View–Controller pattern is adapted to an API-first architecture: the Model is embodied by MongoDB collections and the data access layer that maps internship documents, user profiles, and document metadata; the Controller consists of Express.js route handlers and service functions that validate requests, apply business rules, invoke OCR and Drive operations, and format JSON responses; the View is the React frontend which renders UI components based on API responses. This separation allows the backend to expose stable resource-oriented endpoints (`/api/users`, `/api/internships`, `/api/documents`) while the frontend remains free to evolve its presentation independently, enabling multiple client types to consume the same API.

Repository / Data Access Pattern

A Repository layer abstracts direct database operations and provides a consistent interface for CRUD and query operations against MongoDB. Repositories encapsulate queries for users, internships, and document records, centralize index usage and pagination logic, and hide raw driver code from controllers. This abstraction simplifies unit testing (repositories can be mocked), makes migration to alternate storage simpler, and keeps data access concerns separate from request handling and business logic.

Adapter Pattern

Third-party integrations (Google Drive, Tesseract OCR) are wrapped with Adapter modules that present a consistent internal interface while translating to/from vendor APIs. The Drive Adapter hides upload, folder-creation, and link generation semantics; the OCR Adapter provides a uniform `extractText(file)` method that returns parsed fields regardless of file type. Adapters decouple the core logic from provider details, making it easier to swap implementations, mock integrations in tests, or add alternate storage providers later.

Singleton Pattern

Shared services that should exist as a single instance—such as the Drive client, the OCR engine wrapper, and the Socket.IO server—are implemented as singletons. These singletons initialize once at application startup and are reused across modules, ensuring consistent configuration, minimizing resource usage, and avoiding duplicate connections or repeated initialization overhead.

Observer / Event-Driven Pattern

The Observer pattern underpins real-time updates: server-side events (status changes, approvals, OCR results) are emitted and subscribed to by connected clients via Socket.IO. On the frontend, React components observe state changes (either from WebSocket events or from updated API responses) and re-render automatically. This decoupled event flow enables optimistic UI updates, live notifications of coordinator decisions, and immediate propagation of document verification results without polling.

Pub/Sub (Publish-Subscribe) Pattern

A publish–subscribe model implemented via Socket.IO rooms and server broadcasts decouples event producers from consumers. When a coordinator approves an internship or when OCR flags a document, the backend publishes an event to the relevant room (student, coordinator dashboard, or specific internship room), and all subscribed clients receive updates. This pattern supports scalable multi-client synchronization and allows selective targeting of updates to interested parties.

Factory Pattern

Factory functions produce configured instances of commonly used clients—e.g., Axios instances with interceptors that attach auth tokens, database connection factories that provide ready-to-use MongoDB clients per environment, or Drive client factories that load credentials per tenant. Factories centralize configuration, allow creation of test doubles, and standardize how external dependencies are instantiated across the codebase.

Strategy Pattern

Pluggable algorithms are selected at runtime using the Strategy pattern. Examples include multiple document verification strategies (full OCR match, relaxed keyword match, manual verification fallback), different membership approval flows (auto-approve for specific departments, coordinator-approval for others), and alternate notification strategies (in-app toast vs email). Implementing these as interchangeable strategies makes the system flexible and easier to extend without changing core workflows.

Command Pattern (for approval/workflow actions)

Approval-related operations (approve, reject, request-changes, attach-comment) are encapsulated as Command objects that contain the action, required parameters, and undo/log metadata. Using commands enables consistent logging, easier rollback or replay of actions, and a clear audit trail for coordinator decisions—valuable for administrative review and debugging.

Circuit Breaker / Retry Pattern (integration resilience)

Integration points with external services (Drive uploads, OCR processing) are wrapped with retry and circuit-breaker logic to handle transient failures and prevent cascading errors. Retrying idempotent operations and opening a circuit after repeated failures protects system stability and allows graceful degradation (e.g., flagging a document for manual review when OCR is unavailable).

These patterns together create a robust, testable, and extensible foundation for InternTrack. They help isolate third-party concerns, centralize cross-cutting logic, enable real-time collaboration, and provide clear extension points for future features such as analytics, notification channels, or multi-department support.

4. CONCLUSION

The development of this internship management platform demonstrates strong full-stack web development capability, combining secure authentication, cloud-integrated document handling, automated OCR verification, and role-based workflow management into a single cohesive system. By integrating technologies such as React.js, Node.js, Express.js, MongoDB, Google Drive API, and Tesseract OCR, the application successfully delivers a streamlined, scalable, and user-friendly environment for students and coordinators to manage internship submissions and approvals. The system enables students to create and edit internship entries, upload supporting documents, track approval status, and maintain accurate records, while coordinators can efficiently review submissions, verify documents, filter records, and manage departmental internship workflows through a centralized dashboard.

The project reflects solid understanding of modern software engineering practices including RESTful API design, cloud-based file storage, automated verification through OCR, role-based access control, form-driven data management, and responsive UI development. The modular architecture supports scalability through NoSQL database structures, component-based frontend design, structured backend routing, and reliable API communication. Overall, the application provides a practical and efficient solution for academic departments seeking to reduce manual paperwork, enhance verification accuracy, improve record-keeping, and maintain seamless communication between students and coordinators.

This work establishes a strong foundation for future enhancements such as automated email notifications, advanced analytics dashboards, multi-department integration, improved document classification, certificate validation workflows, and a dedicated mobile application. With its combination of cloud storage, automated verification, and structured internship workflows, the platform effectively addresses real-world academic requirements and showcases readiness for professional-level full-stack development and deployment.