InternHub - Complete MERN Stack Project Structure

Technology Stack

Frontend: React.js with React Router, Axios, Material-UI/Tailwind CSS Backend: Node.js with Express.js Database: MongoDB Atlas (Cloud Database) Authentication: JWT-based with bcrypt ML Service: Python (Flask/FastAPI) for Resume Analysis File Storage: Cloudinary/AWS S3 for documents and images Email Service: SendGrid/Nodemailer for OTP and notifications Deployment: Frontend (Vercel/Netlify), Backend (Railway/Render), ML Service (Railway/Render)

Complete Page Structure

Authentication Pages

1. Landing Page (/)

Components: Hero section, features overview, login/register buttons Functionalities:

- Welcome message and project overview
- Navigation to login/register
- Feature highlights (company listings, resume analysis, etc.)
- Footer with contact info
- Responsive design for mobile/desktop

2. Login Page (/login)

Components: Login form, role selection (Student/Admin) Functionalities:

- Email/username and password fields
- Role-based login (Student/Admin toggle)
- JWT token generation and storage in localStorage
- Redirect to respective dashboards
- "Forgot Password" link
- Social login options (Google OAuth)

3. Register Page (/register)

Components: Registration form with OTP verification Functionalities:

- Student registration: Name, email, phone, college ID, branch, batch, CGPA
- Admin registration: Name, email, phone, college department, admin code
- OTP verification via email
- Profile picture upload to Cloudinary
- Terms and conditions acceptance
- Real-time validation

4. Forgot Password (/forgot-password)

Components: Email input, OTP verification, password reset Functionalities:

- Email input for password reset
- OTP verification via email
- New password setup with validation
- Password strength meter

Student Portal Pages

5. Student Dashboard (/student/dashboard)

Components: Stats cards, quick actions, recent activities Functionalities:

- Total companies available
- Applications submitted count
- Bookmarked companies
- Recent announcements
- Quick access to resume analyzer
- Upcoming deadlines widget
- Profile completion status
- Real-time notifications

6. Company Listings (/student/companies)

Components: Company cards, search bar, filters, pagination Functionalities:

- Display all available companies in card format
- Search by company name, role, domain
- Filter by: Date, CGPA requirement, branch, batch, domain
- Sort by: Latest, deadline, CGPA requirement
- Infinite scroll or pagination
- Quick view of eligibility status
- Apply button with status indication

7. Company Details (/student/companies/:id)

Components: Company info, JD display, application form Functionalities:

- Complete company information
- Job description with PDF preview
- Eligibility criteria display
- Selection rounds breakdown
- Skills required section
- Application deadline countdown
- Apply button (if eligible)
- Bookmark/Save company option
- Share company link

• Company FAQs section

8. My Applications (/student/applications)

Components: Application cards, status filters, timeline Functionalities:

- List of all applied companies
- Application status tracking (Applied, Under Review, Selected, Rejected)
- Timeline view of selection rounds
- Filter by status, date, company
- Download application receipt
- Withdraw application option
- Real-time status updates

9. Bookmarked Companies (/student/bookmarks)

Components: Saved company cards, remove bookmark option Functionalities:

- List of bookmarked companies
- Quick apply from bookmarks
- Remove bookmark option
- Filter bookmarked companies
- Sort by deadline, date added

10. Resume Analyzer (/student/resume-analyzer)

Components: Upload area, analysis results, suggestions Functionalities:

- Resume upload (PDF format) to cloud storage
- ML-based resume scoring via Python API
- Skill gap analysis
- Improvement suggestions
- ATS compatibility check
- Keyword optimization tips
- Download improved resume template
- Compare with job requirements

11. Student Profile (/student/profile)

Components: Profile form, academic details, skills section Functionalities:

- · Personal information editing
- Academic details (CGPA, branch, batch)
- Skills and certifications
- Resume upload and management
- Profile picture update
- · Contact information

Change password option

12. Notifications (/student/notifications)

Components: Notification list, mark as read, filters Functionalities:

- All notifications display
- Mark as read/unread
- Filter by type (Application, Deadline, Results)
- Delete notifications
- Notification preferences settings
- Real-time updates via WebSocket

🧸 Admin Portal Pages

13. Admin Dashboard (/admin/dashboard)

Components: Analytics cards, charts, quick stats Functionalities:

- Total companies added
- Total student applications
- Active vs closed positions
- Student application analytics
- Domain-wise interest charts
- Recent activity feed
- Quick actions (Add company, Post announcement)

14. Company Management (/admin/companies)

Components: Company table, search, filters, action buttons Functionalities:

- List all companies in table format
- Search companies by name, role
- Filter by status (Active, Closed, Draft)
- Edit, delete, duplicate company
- Bulk actions (activate/deactivate)
- Export company data to CSV/Excel
- View applicant count per company

15. Add New Company (/admin/companies/add)

Components: Multi-step form, JD upload, criteria setting Functionalities:

- Step 1: Basic info (name, role, description, visit date, apply link)
- Step 2: Eligibility criteria (CGPA, branch, batch, skills)
- Step 3: Selection rounds (Aptitude, Technical, GD, HR, etc.)
- Step 4: JD upload (PDF) with preview

- Step 5: Company FAQs and preparation tips
- Save as draft option
- Preview before publishing

16. Edit Company (/admin/companies/edit/:id)

Components: Pre-filled form, update options Functionalities:

- Edit all company details
- Update JD documents
- Modify eligibility criteria
- Change selection rounds
- Update application deadline
- View applicant list

17. Applicant Tracking (/admin/applicants)

Components: Applicant table, filters, export options Functionalities:

- View all applicants across companies
- Filter by company, status, branch, batch
- Export applicant data (Excel/CSV)
- Mark application status
- Send bulk emails to applicants
- View individual student profiles

18. Company Applicants (/admin/companies/:id/applicants)

Components: Applicant list for specific company Functionalities:

- List all applicants for a company
- View student profiles and resumes
- Update application status
- Send individual/bulk notifications
- · Export applicant list
- Filter by eligibility criteria

19. Analytics & Reports (/admin/analytics)

Components: Charts, graphs, data tables **Functionalities:**

- Student application trends
- Domain-wise interest analysis
- Company popularity metrics
- Branch-wise application stats
- Placement success rates
- Time-series analysis of applications

· Export analytical reports

20. Announcements (/admin/announcements)

Components: Announcement form, announcement list Functionalities:

- Create new announcements
- Target specific batches/branches
- Schedule announcements
- Edit/delete announcements
- View announcement reach statistics
- Urgent announcement option

21. Student Management (/admin/students)

Components: Student table, search, filters Functionalities:

- View all registered students
- Search by name, email, college ID
- Filter by branch, batch, CGPA
- View student application history
- Export student data
- Send individual notifications

22. Settings (/admin/settings)

Components: Configuration forms, system settings Functionalities:

- Platform settings (deadlines, notifications)
- Email templates configuration
- Admin user management
- System backup options
- Integration settings
- Platform customization

■ MongoDB Atlas Collections Structure

Users Collection

javascript

```
_id: ObjectId,
email: String, // unique
password: String, // hashed with bcrypt
role: String, // 'student' or 'admin'
profile: {
 name: String,
 phone: String,
 college_id: String, // for students
 branch: String, // for students
 batch: String, // for students
 cgpa: Number, // for students
 department: String, // for admins
 profile_picture: String, // Cloudinary URL
 skills: [String], // for students
 resume_url: String // for students
},
is_verified: Boolean,
is_active: Boolean,
last_login: Date,
created_at: Date,
updated_at: Date
```

Companies Collection

javascript

```
_id: ObjectId,
name: String,
role: String,
description: String,
domain: String,
visit_date: Date,
application_deadline: Date,
apply_link: String,
eligibility: {
 min_cgpa: Number,
 allowed_branches: [String],
 allowed_batches: [String],
 required_skills: [String]
},
selection_rounds: [String],
jd_document: String, // Cloudinary URL
company_logo: String, // Cloudinary URL
faqs: [{
 question: String,
 answer: String
preparation_tips: [String],
status: String, // 'active', 'closed', 'draft'
created_by: ObjectId, // Admin ID
created_at: Date,
updated_at: Date
```

Applications Collection

```
javascript
{
    _id: ObjectId,
    student_id: ObjectId,
    company_id: ObjectId,
    status: String, // 'applied', 'under_review', 'selected', 'rejected'
    applied_at: Date,
    resume_file: String, // Cloudinary URL
    additional_documents: [String], // Cloudinary URLs
    notes: String,
    status_history: [{
        status: String,
        changed_at: Date,
        changed_by: ObjectId
    }]
}
```

Bookmarks Collection

```
{
  _id: ObjectId,
  student_id: ObjectId,
  company_id: ObjectId,
  created_at: Date
}
```

Notifications Collection

```
javascript

{
    _id: ObjectId,
    recipient_id: ObjectId,
    type: String, // 'application', 'deadline', 'result', 'announcement'
    title: String,
    message: String,
    is_read: Boolean,
    created_at: Date
}
```

Resume Analysis Collection

```
javascript
 _id: ObjectId,
 student_id: ObjectId,
 resume_file: String, // Cloudinary URL
 analysis_score: Number,
 suggestions: [String],
 keywords_found: [String],
 keywords_missing: [String],
 ats_score: Number,
 sections_analysis: {
  contact: Object,
  summary: Object,
  experience: Object,
  education: Object,
  skills: Object
 analyzed_at: Date
```

Announcements Collection

javascript

```
{
    _id: ObjectId,
    title: String,
    message: String,
    target_audience: {
        branches: [String],
        batches: [String],
        all_students: Boolean
    },
    priority: String, // 'normal', 'high', 'urgent'
    created_by: ObjectId,
    created_at: Date,
    scheduled_at: Date
```

API Endpoints Structure

Authentication APIs

- POST /api/auth/register User registration
- (POST /api/auth/login) User login
- (POST /api/auth/verify-otp) OTP verification
- (POST /api/auth/forgot-password) Password reset request
- (POST /api/auth/reset-password) Password reset confirmation
- (POST /api/auth/refresh-token) Refresh JWT token
- (POST /api/auth/logout) User logout

Student APIs

- (GET /api/student/dashboard) Dashboard data
- (GET /api/student/companies) Company listings with pagination
- GET /api/student/companies/:id) Company details
- (POST /api/student/applications) Apply to company
- (GET /api/student/applications) My applications
- (PUT /api/student/applications/:id) Update application
- (DELETE /api/student/applications/:id) Withdraw application
- [GET /api/student/bookmarks] Bookmarked companies
- (POST /api/student/bookmarks) Add bookmark
- (DELETE /api/student/bookmarks/:id) Remove bookmark
- (GET /api/student/profile) Get profile
- (PUT /api/student/profile) Update profile
- (POST /api/student/upload-resume) Upload resume
- (GET /api/student/notifications) Get notifications
- (PUT /api/student/notifications/:id) Mark as read

Admin APIs

- (GET /api/admin/dashboard) Admin dashboard data
- (GET /api/admin/companies) Company management
- (POST /api/admin/companies) Add new company
- (PUT /api/admin/companies/:id) Update company
- (DELETE /api/admin/companies/:id) Delete company
- (GET /api/admin/applicants) Applicant tracking
- (GET /api/admin/companies/:id/applicants) Company applicants
- PUT /api/admin/applications/:id/status Update application status
- (GET /api/admin/analytics) Analytics data
- (POST /api/admin/announcements) Create announcement
- (GET /api/admin/announcements) Get announcements
- PUT /api/admin/announcements/:id) Update announcement
- DELETE /api/admin/announcements/:id) Delete announcement
- (GET /api/admin/students) Student management
- (POST /api/admin/send-notification) Send notification

File Upload APIs

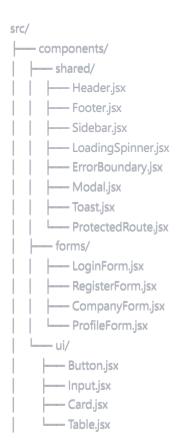
- (POST /api/upload/resume) Upload resume to Cloudinary
- (POST /api/upload/jd) Upload job description
- (POST /api/upload/profile-picture) Upload profile picture

ML Service APIs (Python Flask/FastAPI)

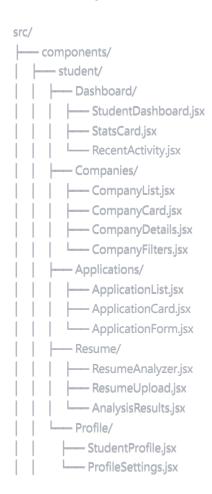
- POST /api/ml/analyze-resume Analyze resume
- (POST /api/ml/match-resume) Match resume with job requirements
- [POST /api/ml/suggest-improvements] Get improvement suggestions

Frontend Component Structure

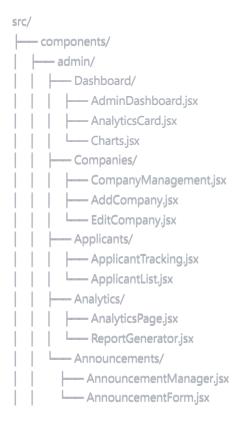
Shared Components



Student Components



Admin Components



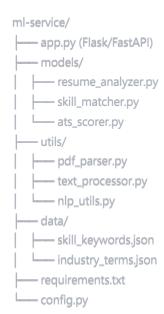
■ Backend Structure (Node.js/Express)

Main Server Structure



Q Python ML Service Structure

ML Service Structure



ML Features

- Resume Parsing: Extract text from PDF using PyPDF2/pdfplumber
- Skill Extraction: NLP-based skill identification using spaCy/NLTK
- ATS Scoring: Keyword matching and formatting analysis
- Job Matching: Similarity scoring between resume and job requirements
- Improvement Suggestions: Al-powered recommendations

Deployment Strategy

Frontend (React.js)

• Platform: Vercel or Netlify

• **Build Command:** (npm run build)

• Environment Variables: API endpoints, authentication keys

Backend (Node.js/Express)

• Platform: Railway, Render, or Heroku

• Environment Variables: MongoDB Atlas connection, JWT secret, Cloudinary config

• Port: Dynamic port assignment

ML Service (Python)

• Platform: Railway or Render

• Requirements: Python 3.8+, Flask/FastAPI

• Environment Variables: ML model paths, API keys

Database

• MongoDB Atlas: Cloud-hosted MongoDB

• Backup: Automated backups

• Scaling: Auto-scaling based on usage

II Performance Optimization

Frontend

- Code splitting with React.lazy()
- Image optimization with Cloudinary
- Caching strategies
- Progressive Web App (PWA) features

Backend

- API response caching with Redis
- Database indexing for frequent queries
- Compression middleware
- Rate limiting

Database

- Proper indexing on frequently queried fields
- Connection pooling
- Query optimization

This comprehensive MERN stack structure provides a scalable, production-ready foundation for your InternHub project with integrated ML capabilities and cloud deployment strategy!