**API Development and Integration Report**

|  |  |
| --- | --- |
| Date | 27-06-2025 |
| Team ID | LTVIP2025TMID58557 |
| Project Name | Project - FreelanceFinder |

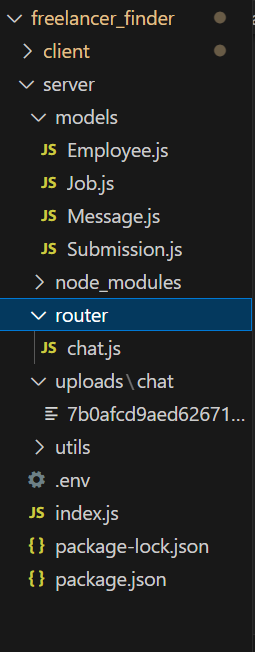
**Project Title:** FreelanceFinder  
**Date:** 27-06-25

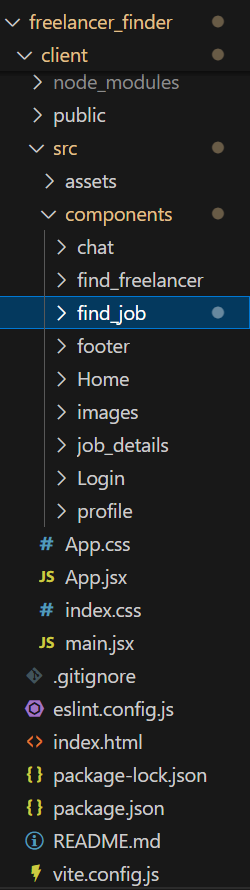
**Objective**  
The objective of this report is to document the API development and integration of backend services for the **Freelance Finder** platform. This includes implementation of user authentication, job posting, media uploads, and real-time chat features, all designed to streamline freelance hiring.

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB with Mongoose
* **Authentication:** JSON Web Token (JWT)
* **File Storage:** Cloudinary
* **Validation**: Express Validator
* **CORS**: Enabled for frontend-backend communication
* **Multer:** For handling multipart/form-data uploads (resumes, logos)

**Project Structure**





**Key API Endpoints**

**User Authentication**

* POST /signup – Registers a new freelancer or client
* POST /login – Authenticates user credentials and returns a JWT

**User Profile**

* PUT /api/employees/update – Updates user profile (skills, phone, resume, image)
* GET /api/employees/:id – Fetches user details by ID

**Job Management**

* POST /api/jobs – Client creates a job post
* GET /api/jobs – Fetch all jobs (with filters for category/location)
* GET /api/jobs/:id – Fetch single job details
* GET /api/jobs/applicants/:jobId – Get all applicants for a job
* POST /api/jobs/apply/:jobId – Freelancer applies to a job

**Chat System**

* POST /api/messages/send – Send a new message
* GET /api/messages/:chatId – Fetch messages of a chat
* GET /api/chats/:userId – Get all chats for a user

**Media Uploads (Cloudinary)**

* Resume, profile picture, and company logo uploads are handled via Multer and stored using Cloudinary’s API.

.

**Integration with Frontend**

The backend integrates with the React frontend using RESTful API architecture. Axios is used to make HTTP requests, and state is managed using React hooks.

* Auth Integration: JWT tokens are stored in localStorage and attached to headers for protected routes.
* Job Search & Listing: Jobs are fetched from the backend with query parameters for filters.
* Chat Integration: Chat messages are sent/received through API and displayed in real-time UI.
* Resume/Logo Upload: Frontend sends files via form-data to backend, which uploads them to Cloudinary and stores the URL.

**Error Handling & Validation**

* **Error Handling**: A centralized middleware captures exceptions and sends consistent error responses.
* **Validation**: Input validation is done using express-validator, especially for signup, job posts, and updates.

**Security Considerations**

* **Authentication:** All protected routes require a valid JWT token**.**
* **Data Encryption:** Passwords are hashed using bcrypt before storage.
* **CORS Configuration:** Proper origin control is enabled for secure frontend-backend communication.
* **File Upload Security:** Only specific file types and size limits are accepted via Multer**.**