🛠️ Technical Backend Specification Report

📌 Project Title

Smart Industrial Practical Training Report Generator (With AI support, PDF/Word export, and secure user management)

🧩 Backend Technology Stack

Component	Technology
Language	Python 3.x
Framework	Django + Django REST Framework
Auth System	JWT Authentication (SimpleJWT)
Database	PostgreSQL
AI Integration	OpenAI GPT (via REST API)
File Export	python-docx, xhtml2pdf
Deployment	Render (Backend), Railway (DB)

🗃️ Backend Architecture & Structure

🏗️ Django App Modules:

App Name	Description
users	Handles registration, login, and user profile details
companies	Stores companies where students are placed
reports	Manages daily, weekly, and general reports
ai_assist	Handles AI enhancement for content generation
exporter	Exports reports to PDF and Word
core (optional)	Shared utilities, constants, mixins

👤 User Authentication

🔐 JWT-Based Authentication:

- Login/Logout/Token Refresh using SimpleJWT

- Frontend will store access + refresh tokens securely (e.g., localStorage + Axios interceptor)

✅ **API Endpoints:**

Method	Endpoint	Function
POST	/api/auth/register/	Register new user
POST	/api/auth/login/	Obtain JWT tokens
POST	/api/auth/logout/	Revoke token (blacklist)
POST	/api/auth/token/refresh/	Refresh token
GET	/api/profile/	Get user profile
PUT	/api/profile/update/	Update profile

👩‍🎓 **User Profile Extension (Model: UserProfile)**

Holds additional student-specific data.

📎 **Fields:**

- user (OneToOne → Django User)
- program: BSc. Mechanical Engineering, etc.
- year_of_study: Integer
- pt_phase: Choice (PT1, PT2, PT3)
- department: String
- supervisor_name: String
- company (FK → Company)

🧭 **Usage:**

- Must be filled during onboarding
- Used to link reports to internship metadata

🏢 **Company Model**

Holds internship company data.

📎 **Fields:**

- name

- address
- contact_person
- phone (optional)
- email (optional)
- industry_type (optional)

🚀 **API Endpoints:**

Method	Endpoint	Description
GET	/api/companies/	List all companies
POST	/api/companies/	Create new company (admin or self-entry)

🗓️ **Daily Report Model**

Captures daily activity per student during IPT.

🧷 **Fields:**

- student (FK → User)
- date (e.g., 2025-07-17)
- description: What was done

hours_spent: Decimal (e.g., 6.5)

- week_number: Integer
- weekly_report (optional FK → WeeklyReport)

🚀 **API Endpoints:**

Method	Endpoint	Description
POST	/api/reports/daily/	Submit daily report
GET	/api/reports/daily/	Get all daily reports
PUT	/api/reports/daily/<id>/	Update daily report
DELETE	/api/reports/daily/<id>/	Delete daily report

📘 **Weekly Report Model**

A weekly summary built on top of daily entries.

🔗 **Fields:**

- student (FK → User)
- week_number
- start_date (Monday)
- end_date (Saturday)
- summary: Manual or AI-generated
- main_job_title: Main job name for the week
- main_job_detail: Description of that job

When submitting, student selects daily reports within the date range + inputs main job manually or with AI.

🚀 **API Endpoints:**

Method	Endpoint	Description
POST	/api/reports/weekly/	Create weekly summary
GET	/api/reports/weekly/	List user weekly reports
GET	/api/reports/weekly/<id>/	View one weekly report
PUT	/api/reports/weekly/<id>/	Edit a weekly report

🔁 **Main Job Operation Table**

Each weekly report has a detailed breakdown of the selected main task.

🔗 **Fields:**

- weekly_report (FK)
- step_number: Integer (step order)
- operation_name: e.g., Assembling
- tools_used: Tools/equipment used

🚀 **API Endpoints:**

Method	Endpoint	Description
POST	/api/reports/weekly/<id>/operations/	Add operation step
GET	/api/reports/weekly/<id>/operations/	List steps
DELETE	/api/reports/operations/<id>/	Delete a step

🧠 AI Assistance Integration (OpenAI GPT)

📎 Usage:

- Assist user by enhancing or rewording text
- Suggest general summaries
- Auto-complete general report fields if data exists

🚀 API Endpoints:

Method	Endpoint	Description
POST	/api/ai/enhance/weekly/	Improve weekly content
POST	/api/ai/enhance/general/	Improve general report
POST	/api/ai/autocomplete/general/	Fill missing fields based on data

---

📄 General Report Model

Summarizes the entire IPT experience.

📎 Fields:

- user (FK → User)
- introduction
- company_overview
- objectives
- achievements
- challenges
- recommendations
- compiled_from (ManyToMany → WeeklyReport)

🚀 API Endpoints:

Method	Endpoint	Description
POST	/api/reports/general/	Create general report
GET	/api/reports/general/	View general report
PUT	/api/reports/general/	Edit general report

📄 File Export API (PDF/Word)

Exports reports using academic template.

🚀 Endpoints:

Method	Endpoint	Description
GET	/api/export/weekly/<id>/?format=pdf	Export weekly as PDF
GET	/api/export/weekly/<id>/?format=docx	Export weekly as DOCX
GET	/api/export/general/?format=pdf	Export general report
GET	/api/export/general/?format=docx	Export general report

🔐 Security Configuration

- JWT tokens stored securely and refreshed periodically
- All API endpoints protected by IsAuthenticated
- CSRF protection enabled for unsafe methods
- Input validation via DRF serializers
- Passwords hashed using Django's PBKDF2
- XSS & SQL Injection mitigated via Django ORM

⚙️ Configuration & Deployment

- .env file for secrets (OpenAI keys, DB creds)
- PostgreSQL DB hosted on Railway
- Django + Gunicorn hosted on Render
- CORS configured for React frontend
- Static files served via WhiteNoise (or CDN for scale)

📅 Development Timeline (Backend Only)

Phase	Duration	Description
Setup & Auth	2 days	Django project + JWT auth
Models & DB Schema	2 days	UserProfile, Company, Reports, etc.
Daily/Weekly APIs	2–3 days	CRUD endpoints

Phase	Duration	Description
AI Integration	2 days	OpenAI-enhanced fields
Export Engine	2 days	DOCX + PDF format logic
Testing & Docs	2 days	Postman tests, schema review

📍 Final Notes for Frontend Devs

Key Things to Know:

- All endpoints are versioned under /api/
- All date fields are in YYYY-MM-DD format
- All content-enhancement fields are optional, can work with or without AI
- Upload of files (e.g., logs, images) is not in scope for now, can be added later
- Export APIs return binary files,

so handle blob download logic in frontend