GA Hiresync – Full Environment Setup Guide

This document provides **complete setup instructions** for running the **Backend (Django)** and **Frontend (React)** of GA Hiresync.
It includes **Celery**, **Docker**, **Environment Variables**, and commands to get everything running.

🛠️ Backend Setup (Django)

1️⃣ Clone the Repository

```
git clone https://github.com/<your-repo>/ga-hiresync.git
cd ga-hiresync/backend
```

2️⃣ Create and Activate Virtual Environment

```
python3 -m venv venv
source venv/bin/activate      # For Linux/Mac
venv\Scripts\activate         # For Windows
```

3️⃣ Install Dependencies

```
pip install -r .requirements.txt
```

4 Setup Environment Variables

```
SECRET_KEY='3knfsd@!v=9p%8s@_$qttk'
DB_NAME='rtmas'
DB_USER='postgres'
DB_PASSWORD='12312'
DB_HOST='localhost'
DB_PORT='5433'
EMAIL_ID='kal23k2323iswerwr2r23r32mail.com'
EMAIL_PASSWORD='dbfopassworx'
apiurl='http://recruitment.gaorgsync.com'
environment='localhost'
LINKEDIN_CLIENT_ID='86p6yunklw1f9h'
LINKEDIN_CLIENT_SECRET='WPL_AP1.tOaYgyJh6HD262Zk.1knlLA=='
SIGNING_KEY='sI3k6dN6eQh6wYz9CwFzG2jM5pR7tUwXzAeD7gSjWnZr4u7z'
GEMINI_API_KEY='AIzaSyDdx7YY6AcBYTt4AaJb_0PjTN8_5CUIDwk'
frontendurl='http://localhost:3000'
backendurl='http://localhost:8000'
FRONTENDURL='http://localhost:3000'
```

5 Run Database Migrations

```
python manage.py makemigrations
python manage.py migrate
```

6 Create Superuser

```
python manage.py createsuperuser
```

7 Start Django Server

```
python manage.py runserver
```

8 Run Celery and Celery Beat

```
celery -A backend_app worker -l info
celery -A backend_app beat -l info
```

9️⃣ Docker Setup (Backend)

```
docker-compose up --build
```

💻 Frontend Setup (React)

1️⃣ Navigate to Frontend

```
cd ../frontend
```

2️⃣ Install Dependencies

```
npm install
```

3️⃣ Setup Environment Variables

```
REACT_APP_BACKEND_URL=http://localhost:8000
REACT_APP_GOOGLE_AI_API_KEY=aefijjafaweef w;efjfw
```

4️⃣ Start React App

```
npm start
```

📁 Version Control & Branching

Our project follows Git-based version control using **GitHub**, with two separate repositories:

- **Frontend Repository** – React-based UI
- **Backend Repository** – Django (or Go-based) API server

Currently, both repositories use the default branch (`main`). We recommend the following branching strategy for better collaboration and future scalability:

- `main` : Stable production-ready code
- `dev` : Ongoing development and integration
- `feature/<feature-name>` : New feature branches off `dev`
- `bugfix/<issue-name>` : Hotfixes or patches

- `release/<version>` : Pre-release staging before merging to `main`

> 💡 Tip: Always create a pull request from your feature/bugfix branch into `dev` and ensure CI checks pass before merging.

GitHub Repositories

- Frontend: github.com/GA-DIGITAL-SOLUTIONS/RMS_BACKEND
- Backend: github.com/GA-DIGITAL-SOLUTIONS/RMS_FRONTEND

🔌 Third Party Dependencies & Integrations

We use a number of external services to enhance functionality and automation in our application:

✅ LinkedIn API Integration

We've integrated the **LinkedIn API** to allow features such as:

- Social profile fetching (with user consent)
- Job-related data retrieval
- Posting or sharing updates programmatically

Documentation:

- LinkedIn API Overview
- LinkedIn Developer Portal
- Authentication Guide

Authentication:

- OAuth 2.0 (Authorization Code Flow)
- Requires `client_id` , `client_secret` , and a valid `redirect_uri`

🔐 LinkedIn Credential Management

We have implemented dedicated models to manage LinkedIn account credentials separately for security and scalability:

- `AgencyLinkedInAccount` – Stores LinkedIn credentials specific to each **agency**.
- `GAHireSyncLinkedInAccount` – Stores LinkedIn credentials for the **GA HireSync system account**.

> 🔄 **All user accounts logging into the system are required to have a valid LinkedIn page.**
> This is essential, as all job posts created via the system are automatically published to the

> respective LinkedIn page of the agency or the GA HireSync account.

🚀 Future Enhancement

To improve security, we plan to implement **encryption** for storing LinkedIn credentials in the database. This includes:

- Encrypting access tokens and refresh tokens using AES or RSA
- Managing encryption keys via environment variables or a secure key vault (e.g., AWS KMS, Google Secret Manager)
- Adding token expiry tracking and auto-refresh support

> 🛡️ This enhancement will ensure credentials are not stored in plain text and remain secure even in the event of a data breach.

✅ Google Gemini API Integration

We also leverage **Google's Gemini API** for AI-powered features like summarization, semantic understanding, and intelligent suggestions.

Documentation:

- Gemini API Quickstart (REST)
- Gemini API with Python SDK
- Authentication via Google Cloud

Features Used:

- Natural language processing
- AI-based reasoning
- Interactive summarization

> ⚠️ Note: Google Gemini requires authentication via **Google Cloud IAM** and access to a valid **Gemini API key**.

Keep all tokens, secrets, and API keys securely stored in environment variables and never commit them to version control.