Project Samarth - Complete Setup & Deployment Guide

Table of Contents

1. Prerequisites

Before starting, ensure you have the following installed:

Required Software

- **Python 3.9+** (Python 3.10 recommended)
- **Node.js 16+** and npm
- **Git**
- **VS Code** or any code editor

API Keys Required

- data.gov.in **API Key**: Register at https://data.gov.in and get API key from "My Account"
- **Groq API Key** (FREE): Register at https://console.groq.com and get API key
  - OR **OpenAI API Key**: From https://platform.openai.com (requires payment)

2. Project Setup

Step 1: Create Project Directory

Open your terminal and run:

```
# Create main project directory
mkdir project-samarth
cd project-samarth
```

```
# Create subdirectories
mkdir backend frontend notebooks deployment scripts
```

Step 2: Initialize Git Repository

```
git init
echo "*.pyc" &gt;&gt; .gitignore
echo "__pycache__/" &gt;&gt; .gitignore
echo "*.env" &gt;&gt; .gitignore
echo "node_modules/" &gt;&gt; .gitignore
echo "vector_store/" &gt;&gt; .gitignore
echo ".vscode/" &gt;&gt; .gitignore
```

3. Backend Setup

Step 1: Create Virtual Environment

```
# Navigate to backend directory
cd backend

# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate
```

Step 2: Create Backend Structure

```
# Create all backend directories
mkdir data_fetcher embeddings chatbot utils

# Create __init__.py files
touch data_fetcher/__init__.py
touch embeddings/__init__.py
touch chatbot/__init__.py
touch utils/__init__.py
```

Step 3: Create requirements.txt

Create `backend/requirements.txt`:

```
flask==3.0.0
flask-cors==4.0.0
```

```
python-dotenv==1.0.0
requests==2.31.0
pandas==2.1.4
numpy==1.26.2
langchain==0.1.0
langchain-community==0.0.10
groq==0.4.1
sentence-transformers==2.2.2
faiss-cpu==1.7.4
chromadb==0.4.22
cachetools==5.3.2
aiohttp==3.9.1
```

Step 4: Install Dependencies

```
# Make sure venv is activated
pip install --upgrade pip
pip install -r requirements.txt
```

Step 5: Create Environment Variables

Create `backend/.env`:

```
# API Keys
DATA_GOV_API_KEY=your_data_gov_api_key_here
GROQ_API_KEY=your_groq_api_key_here

# Application Settings
FLASK_ENV=development
FLASK_PORT=5000
DEBUG=True

# LLM Configuration
LLM_PROVIDER=groq
LLM_MODEL=mixtral-8x7b-32768
EMBEDDING_MODEL=sentence-transformers/all-MiniLM-L6-v2

# Cache Settings
CACHE_DURATION=3600
```

Important: Replace `your_data_gov_api_key_here` and `your_groq_api_key_here` with your actual API keys.

Step 6: Copy All Backend Code Files

Copy all the backend Python files from the provided code documentation into their respective directories:

- `config.py` → `backend/`
- `app.py` → `backend/`

- All files in `data_fetcher/` → `backend/data_fetcher/`
- All files in `embeddings/` → `backend/embeddings/`
- All files in `chatbot/` → `backend/chatbot/`
- All files in `utils/` → `backend/utils/`

4. Frontend Setup

Step 1: Initialize React App

```
# Go to frontend directory from project root
cd ../frontend

# Create React app
npx create-react-app .
```

Step 2: Install Additional Dependencies

```
npm install axios recharts react-markdown lucide-react
```

Step 3: Create Frontend Structure

```
# Create component directories
mkdir src/components src/services src/utils
```

Step 4: Create Environment Variables

Create `frontend/.env`:

```
REACT_APP_API_URL=http://localhost:5000
REACT_APP_API_TIMEOUT=30000
```

Step 5: Copy All Frontend Code Files

Copy all the frontend React files from the provided code documentation:

- `src/index.js` → Replace default
- `src/index.css` → Replace default
- `src/App.js` → Replace default
- `src/App.css` → Replace default
- `src/services/api.js` → Create new
- All component files → `src/components/`

5. Running Locally

Step 1: Start Backend Server

Open Terminal 1:

```
# Navigate to backend directory
cd project-samarth/backend

# Activate virtual environment
# Windows:
venv\Scripts\activate
# macOS/Linux:
source venv/bin/activate

# Run Flask server
python app.py
```

Expected Output:

```
Initializing RAG Pipeline...
Loading embedding model: sentence-transformers/all-MiniLM-L6-v2
Model loaded. Embedding dimension: 384
LLM Handler initialized: groq - mixtral-8x7b-32768
Vector store not found. Indexing data...
Starting data indexing...
Fetching crop production data...
...
============================================================
Starting Project Samarth Backend Server
============================================================
Port: 5000
Debug Mode: True
Vector Store Indexed: True
============================================================

 * Running on http://0.0.0.0:5000
```

Note: First run will take 5-10 minutes to download models and index data.

Step 2: Start Frontend Server

Open Terminal 2:

```
# Navigate to frontend directory
cd project-samarth/frontend

# Start React dev server
npm start
```

Expected Output:

```
Compiled successfully!

You can now view project-samarth-frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.x.x:3000
```

Step 3: Access the Application

Open your browser and go to: http://localhost:3000

You should see the Project Samarth interface with:

- Chat interface
- Statistics panel
- Example queries

6. Deployment

Option A: Deploy Backend to Render

Step 1: Create Render Account

- Go to https://render.com and sign up
- Connect your GitHub account

Step 2: Push Code to GitHub

```
cd project-samarth
git add .
git commit -m "Initial commit"
git remote add origin YOUR_GITHUB_REPO_URL
git push -u origin main
```

Step 3: Create Web Service on Render

1. Click **"New +"** → **"Web Service"**
2. Connect your GitHub repository
3. Configure:
    - **Name**: project-samarth-backend
    - **Region**: Choose nearest
    - **Branch**: main

- **Root Directory**: backend
- **Runtime**: Python 3
- **Build Command**: `pip install -r requirements.txt`
- **Start Command**: `gunicorn app:app`

4. Add Environment Variables:
   - `DATA_GOV_API_KEY`
   - `GROQ_API_KEY`
   - `FLASK_ENV=production`
   - `DEBUG=False`
5. Click **"Create Web Service"**

Step 4: Note Your Backend URL

After deployment, you'll get a URL like:
`https://project-samarth-backend.onrender.com`

Option B: Deploy Frontend to Netlify

Step 1: Build Production Files

```
cd frontend

# Update .env for production
echo "REACT_APP_API_URL=https://project-samarth-backend.onrender.com" &gt; .env.productio

# Build
npm run build
```

Step 2: Deploy to Netlify

Option 1: Drag & Drop

1. Go to https://app.netlify.com
2. Sign up/Login
3. Drag the `build/` folder to the deploy area

Option 2: Netlify CLI

```
npm install -g netlify-cli
netlify login
netlify deploy --prod --dir=build
```

Step 3: Configure Environment Variables

In Netlify dashboard:

1. Go to Site Settings → Environment Variables
2. Add: `REACT_APP_API_URL` with your Render backend URL

7. Testing the Application

Test Sample Queries

Try these queries in the chat interface:

1. **Simple Query**:
   ```
   What are the top 5 crops produced in India?
   ```
2. **Comparison Query**:
   ```
   Compare the average annual rainfall in Punjab and Haryana for the last 5 years
   ```
3. **Trend Analysis**:
   ```
   Analyze the production trend of wheat in Uttar Pradesh over the last decade
   ```
4. **Correlation Query**:
   ```
   What is the relationship between rainfall and rice production in West Bengal?
   ```

8. Troubleshooting

Issue 1: "Module not found" errors

Solution:

```
# Backend
cd backend
source venv/bin/activate  # or venv\Scripts\activate on Windows
pip install -r requirements.txt

# Frontend
cd frontend
rm -rf node_modules
npm install
```

Issue 2: Backend won't start - Port already in use

Solution:

```
# Windows
netstat -ano | findstr :5000
taskkill /PID &lt;PID&gt; /F

# macOS/Linux
lsof -ti:5000 | xargs kill -9
```

Issue 3: CORS errors in browser

Solution: Check that flask-cors is installed and backend is running on correct port.

Issue 4: No data indexed

Solution:

```
# Make sure DATA_GOV_API_KEY is set
# Delete vector_store folder and restart
cd backend
rm -rf vector_store
python app.py
```

Issue 5: Model download fails

Solution:

```
# Run this separately to download models
python -c "from sentence_transformers import SentenceTransformer; SentenceTransformer('se
```

9. Directory Structure Summary

```
project-samarth/
├── backend/
│   ├── venv/
│   ├── vector_store/
│   ├── data_fetcher/
│   ├── embeddings/
│   ├── chatbot/
│   ├── utils/
│   ├── app.py
│   ├── config.py
│   ├── requirements.txt
│   └── .env
│
├── frontend/
│   ├── node_modules/
```

```
│   ├── public/
│   ├── src/
│   │   ├── components/
│   │   ├── services/
│   │   ├── App.js
│   │   ├── App.css
│   │   ├── index.js
│   │   └── index.css
│   ├── package.json
│   └── .env
│
└── .gitignore
```

10. Performance Tips

Backend Optimization

1. **Use caching**: The cache manager is already implemented
2. **Limit vector store size**: Adjust limit parameter in data fetching
3. **Use smaller models**: Switch to smaller embedding models for faster response

Frontend Optimization

1. **Enable production build**: Always use `npm run build` for deployment
2. **Lazy loading**: Implement code splitting for components
3. **Optimize images**: Compress any images used

11. Free Resource Limits

data.gov.in

- **Rate Limit**: 1000 requests/day
- **Cost**: FREE

Groq

- **Rate Limit**: 30 requests/minute
- **Cost**: FREE

Render (Free Tier)

- **Instance**: 512 MB RAM
- **Sleep**: After 15 mins inactivity
- **Build Minutes**: 500/month

Netlify (Free Tier)

- **Bandwidth**: 100 GB/month
- **Build Minutes**: 300/month
- **Sites**: Unlimited

12. Next Steps

After successful deployment:

1. **Test thoroughly** with various queries
2. **Document** any issues encountered
3. **Create Loom video** demonstrating:
    - Dataset exploration
    - Code walkthrough
    - System design decisions
    - Live demo of Q&A functionality
4. **Submit** your Loom video link

Support & Resources

- data.gov.in **API Docs**: https://data.gov.in/help
- **Groq Documentation**: https://console.groq.com/docs
- **Flask Docs**: https://flask.palletsprojects.com/
- **React Docs**: https://react.dev/

Good Luck with Your Project!