Project Samarth - Step-by-Step Execution Guide

Quick Start Guide for Running Locally

This guide will take you from zero to a fully functional Project Samarth system in under 30 minutes.

Prerequisites Checklist

Before you begin, make sure you have:

- [ ] Python 3.9+ installed (`python --version`)
- [ ] Node.js 16+ installed (`node --version`)
- [ ] Git installed (`git --version`)
- [ ] Code editor (VS Code recommended)
- [ ] Terminal/Command Prompt access
- [ ] Internet connection (for downloading dependencies and API access)

Phase 1: Get API Keys (5 minutes)

Step 1: Get data.gov.in API Key

1. Go to: https://data.gov.in
2. Click "Register" (top right)
3. Fill in details:
    - Name
    - Email
    - Create password
4. Verify email
5. Login and go to "My Account"
6. Copy your API Key (40 characters)

Example: `579b464db66ec23bdd000001cdd3946e44ce4aad7209ff7b23ac571b`

Step 2: Get Groq API Key (FREE - No Credit Card Required)

1. Go to: https://console.groq.com
2. Click "Sign Up" (free tier)
3. Login with Google/Email

4. Go to API Keys section
5. Click "Create API Key"
6. Copy the key immediately (shown only once)

Example: `gsk_abc123xyz456def789`

Note: Groq is FREE and provides fast inference with models like Mixtral. No credit card needed!

Phase 2: Backend Setup (10 minutes)

Step 1: Create Project Structure

Open your terminal and run:

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

# Create backend directory
mkdir backend
cd backend
```

Step 2: Set Up Virtual Environment

```
# Create virtual environment
python -m venv venv

# Activate it
# On Windows PowerShell:
venv\Scripts\Activate.ps1

# On Windows CMD:
venv\Scripts\activate.bat

# On macOS/Linux:
source venv/bin/activate

# You should see (venv) at the start of your terminal prompt
```

Step 3: Create Backend Files

Create these directories:

```
mkdir data_fetcher embeddings chatbot utils
```

Create `__init__.py` files:

```
# Windows:
type nul &gt; data_fetcher\__init__.py
type nul &gt; embeddings\__init__.py
type nul &gt; chatbot\__init__.py
type nul &gt; utils\__init__.py

# macOS/Linux:
touch data_fetcher/__init__.py
touch embeddings/__init__.py
touch chatbot/__init__.py
touch utils/__init__.py
```

Step 4: Create requirements.txt

Create file `backend/requirements.txt` and paste:

```
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 5: Install Dependencies

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

This will take 3-5 minutes. You'll see packages being downloaded and installed.

Step 6: Create .env File

Create file `backend/.env` and paste (replace with YOUR keys):

```
DATA_GOV_API_KEY=your_actual_data_gov_key_here
GROQ_API_KEY=your_actual_groq_key_here

FLASK_ENV=development
FLASK_PORT=5000
DEBUG=True

LLM_PROVIDER=groq
```

```
LLM_MODEL=mixtral-8x7b-32768
EMBEDDING_MODEL=sentence-transformers/all-MiniLM-L6-v2

CACHE_DURATION=3600
```

IMPORTANT: Replace the placeholder text with your actual API keys from Phase 1!

Step 7: Copy All Backend Code Files

Now copy all the Python code files from the documentation into your backend directory:

Files to create:

1. `backend/config.py`
2. `backend/app.py`
3. `backend/data_fetcher/data_gov_client.py`
4. `backend/data_fetcher/data_processor.py`
5. `backend/data_fetcher/cache_manager.py`
6. `backend/data_fetcher/__init__.py`
7. `backend/embeddings/embedding_generator.py`
8. `backend/embeddings/vector_store.py`
9. `backend/embeddings/__init__.py`
10. `backend/chatbot/llm_handler.py`
11. `backend/chatbot/rag_pipeline.py`
12. `backend/chatbot/query_processor.py`
13. `backend/chatbot/__init__.py`
14. `backend/utils/helpers.py`
15. `backend/utils/__init__.py`

Tip: Use the code files provided in the markdown documents. Copy-paste each file's content into the correct location.

Step 8: Test Backend Installation

```
# Make sure you're in backend/ directory with venv activated
python -c "import flask, pandas, sentence_transformers; print('All imports successful!')'
```

If you see "All imports successful!", you're ready!

Phase 3: Frontend Setup (10 minutes)

Step 1: Create React App

Open a NEW terminal (keep backend terminal open):

```
# Go to project root
cd project-samarth

# Create frontend
mkdir frontend
cd frontend

# Initialize React app (this takes 2-3 minutes)
npx create-react-app .
```

Step 2: Install Additional Dependencies

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

Step 3: Create Component Directories

```
mkdir src/components
mkdir src/services
mkdir src/utils
```

Step 4: Create .env File

Create `frontend/.env`:

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

Step 5: Copy All Frontend Files

Replace these default files:

1. `frontend/src/index.js`
2. `frontend/src/index.css`
3. `frontend/src/App.js`
4. `frontend/src/App.css`

Create these new files:

1. `frontend/src/services/api.js`
2. `frontend/src/components/ChatInterface.js`

3. `frontend/src/components/ChatInterface.css`
4. `frontend/src/components/MessageList.js`
5. `frontend/src/components/MessageList.css`
6. `frontend/src/components/InputBox.js`
7. `frontend/src/components/InputBox.css`
8. `frontend/src/components/SourceCitation.js`
9. `frontend/src/components/SourceCitation.css`
10. `frontend/src/components/Statistics.js`
11. `frontend/src/components/Statistics.css`
12. `frontend/src/components/LoadingIndicator.js`
13. `frontend/src/components/LoadingIndicator.css`

Tip: Copy content from the React Components markdown document.

Phase 4: Running the Application (5 minutes)

Terminal 1: Start Backend

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

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

# Run backend
python app.py
```

What to expect:

```
Initializing RAG Pipeline...
Loading embedding model: sentence-transformers/all-MiniLM-L6-v2
Downloading model... (first time only)
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...
Generating embeddings for XXX crop documents...
Fetching rainfall data...
Generating embeddings for XXX rainfall documents...
Data indexing completed!
Total documents indexed: XXXX

============================================================
Starting Project Samarth Backend Server
```

```
============================================================
Port: 5000
Debug Mode: True
Vector Store Indexed: True
============================================================

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

IMPORTANT: First run will take 5-10 minutes because it:

1. Downloads embedding models (~100MB)
2. Fetches data from data.gov.in API
3. Generates embeddings for all documents
4. Saves vector store to disk

Subsequent runs will be much faster (5-10 seconds) as it loads from saved vector store.

Terminal 2: Start Frontend

Open a NEW terminal:

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

# Start React app
npm start
```

What to expect:

```
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

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Browser will automatically open at http://localhost:3000

Phase 5: Testing the System (5 minutes)

Test 1: Check Backend Health

Open browser and go to: http://localhost:5000

You should see JSON response:

```
{
  "status": "online",
  "service": "Project Samarth - Agricultural Data Q&amp;A System",
  "version": "1.0.0",
  "indexed": true,
  "vector_store_stats": {
    "total_documents": 1500,
    "embedding_dimension": 384,
    "index_size": 1500
  }
}
```

Test 2: Check Frontend

Go to: http://localhost:3000

You should see:

- ✅ Project Samarth header
- ✅ "Connected to data.gov.in" green indicator
- ✅ Chat interface with welcome message
- ✅ Statistics panel showing system stats
- ✅ Example queries on the right

Test 3: Send a Query

Click on an example query or type:

Query 1 (Simple):

```
What are the top 5 crops produced in India?
```

Expected: System should respond in 5-10 seconds with answer and sources.

Query 2 (Complex):

```
Compare the average annual rainfall in Punjab and Haryana for the last 5 years
```

Expected: More detailed answer with comparisons and data sources cited.

Query 3 (Correlation):

```
Analyze the relationship between rainfall and rice production in West Bengal
```

Expected: Analysis with correlations and data citations.

Common Issues and Solutions

Issue 1: "ModuleNotFoundError"

Problem: Missing Python package

Solution:

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

Issue 2: "Port 5000 already in use"

Problem: Another app using port 5000

Solution:

Windows:

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

macOS/Linux:

```
lsof -ti:5000 | xargs kill -9
```

Or change port in `backend/.env`:

```
FLASK_PORT=5001
```

And in `frontend/.env`:

```
REACT_APP_API_URL=http://localhost:5001
```

Issue 3: "API Key Invalid"

Problem: Wrong API key or not set

Solution:

1. Check `backend/.env` file
2. Verify API keys are correct (no quotes, no spaces)
3. For data.gov.in: Make sure you're logged in and copied from "My Account"
4. For Groq: Generate a new key if needed

Issue 4: "CORS Error" in Browser Console

Problem: Backend not running or wrong URL

Solution:

1. Make sure backend is running (Terminal 1 should show "Running on...")
2. Check `frontend/.env` has correct `REACT_APP_API_URL`
3. Restart both backend and frontend

Issue 5: "No Data Indexed"

Problem: Data fetching failed

Solution:

1. Check internet connection
2. Verify DATA_GOV_API_KEY in `.env`
3. Delete `vector_store/` folder and restart backend
4. Check data.gov.in API limits (1000 requests/day)

Issue 6: Models Not Downloading

Problem: Network issues or disk space

Solution:

```
# Manually download model
cd backend
source venv/bin/activate
python -c "from sentence_transformers import SentenceTransformer; SentenceTransformer('se
```

Performance Benchmarks

First Run:

- Backend startup: 5-10 minutes (downloading + indexing)
- Frontend startup: 30 seconds
- Total: ~10-15 minutes

Subsequent Runs:

- Backend startup: 5-10 seconds (loads from disk)
- Frontend startup: 30 seconds
- Total: ~1 minute

Query Response Time:

- Simple queries: 2-5 seconds
- Complex queries: 5-10 seconds
- Very complex queries: 10-15 seconds

Verifying Everything Works

Checklist

Backend (Terminal 1):

- [ ] Virtual environment activated
- [ ] No error messages
- [ ] Shows "Vector Store Indexed: True"
- [ ] Shows "Running on http://0.0.0.0:5000"

Frontend (Terminal 2):

- [ ] npm start successful
- [ ] No compilation errors
- [ ] Shows "Compiled successfully!"

Browser (http://localhost:3000):

- [ ] Page loads completely
- [ ] Green "Connected to data.gov.in" indicator
- [ ] Statistics panel shows numbers
- [ ] Can type in chat input
- [ ] Example queries are clickable
- [ ] Sending a query gets a response with sources

Next Steps After Successful Setup

1. **Test All Sample Queries**: Try each example query
2. **Check Sources**: Expand source citations to verify data
3. **Explore Statistics**: Monitor system performance
4. **Test Edge Cases**: Try unusual or complex questions
5. **Take Screenshots**: Document your working system
6. **Record Loom Video**: Show functionality for submission

File Checklist

Backend Files (15 files)

- [ ] backend/config.py
- [ ] backend/app.py
- [ ] backend/requirements.txt
- [ ] backend/.env
- [ ] backend/data_fetcher/**init**.py
- [ ] backend/data_fetcher/data_gov_client.py
- [ ] backend/data_fetcher/data_processor.py
- [ ] backend/data_fetcher/cache_manager.py
- [ ] backend/embeddings/**init**.py
- [ ] backend/embeddings/embedding_generator.py
- [ ] backend/embeddings/vector_store.py
- [ ] backend/chatbot/**init**.py
- [ ] backend/chatbot/llm_handler.py
- [ ] backend/chatbot/rag_pipeline.py
- [ ] backend/chatbot/query_processor.py
- [ ] backend/utils/**init**.py
- [ ] backend/utils/helpers.py

Frontend Files (14 files)

- [ ] frontend/package.json (modified)
- [ ] frontend/.env
- [ ] frontend/src/index.js
- [ ] frontend/src/index.css

- [ ] frontend/src/App.js
- [ ] frontend/src/App.css
- [ ] frontend/src/services/api.js
- [ ] frontend/src/components/ChatInterface.js
- [ ] frontend/src/components/ChatInterface.css
- [ ] frontend/src/components/MessageList.js
- [ ] frontend/src/components/MessageList.css
- [ ] frontend/src/components/InputBox.js
- [ ] frontend/src/components/InputBox.css
- [ ] frontend/src/components/SourceCitation.js
- [ ] frontend/src/components/SourceCitation.css
- [ ] frontend/src/components/Statistics.js
- [ ] frontend/src/components/Statistics.css
- [ ] frontend/src/components/LoadingIndicator.js
- [ ] frontend/src/components/LoadingIndicator.css

System Requirements

Minimum:

- CPU: Dual-core processor
- RAM: 4 GB
- Disk: 2 GB free space
- Internet: Stable connection

Recommended:

- CPU: Quad-core processor
- RAM: 8 GB+
- Disk: 5 GB free space
- Internet: 10+ Mbps

Time Estimates

Total Setup Time: 30-45 minutes

- Getting API keys: 5 minutes
- Backend setup: 10 minutes
- Frontend setup: 10 minutes

- First run & testing: 10-15 minutes

Subsequent Startups: 1-2 minutes

Support Resources

If you get stuck:

1. **Check Error Messages**: Read the full error in terminal
2. **Verify API Keys**: Most issues are due to incorrect keys
3. **Check File Locations**: Make sure files are in correct directories
4. **Review Logs**: Backend terminal shows detailed logs
5. **Browser Console**: Press F12 to see frontend errors

You're Ready! Start Building!

Follow this guide step-by-step and you'll have a fully functional Project Samarth system running locally.