# MyAIFactCheck Development & User Guide

## Table of Contents

1. Project Overview
2. Purpose & Objectives
3. Key Features
4. System Architecture
5. Prerequisites & Dependencies
6. Account Setup & Credentials
7. Development Environment Setup
8. Database Setup
9. Search-Augmented Fact-Checking Implementation
10. Application Structure
11. Deployment Process
12. How to Use MyAIFactCheck
13. Challenges & Solutions
14. Testing & Monitoring
15. Maintenance & Updates
16. Troubleshooting Guide

## 1. Project Overview

**MyAIFactChecker** is an AI-powered multilingual fact-checking platform developed by **Brain Builders Youth Development Initiative (BBYDI)** under the **FactCheck Africa** initiative.

The platform empowers users, journalists, and communities to verify claims, news articles, and online content, reducing misinformation and disinformation in Africa.

* **Repository:** <https://github.com/Brain-Builders/myaifactcheck>

## 2. Purpose & Objectives

* Provide **fast, AI-assisted verification** of claims.
* Support **citizen engagement** with reliable fact-based responses.
* Reduce **spread of misinformation** on social and digital platforms.
* Archive **fact-checking results** for research and accountability.

## 3. Key Features

* **Multilingual Support:** English, Hausa, Yoruba, Igbo, French, Arabic, Swahili.
* **Search-Augmented Generation:** Uses Tavily + Serper APIs with Groq/OpenAI LLMs.
* **Direct Verdicts:** True, False, Misleading, or Unverifiable.
* **Source Transparency:** Relevant URLs logged and exportable.
* **Voice Input:** Users can dictate claims for verification.
* **CSV Export:** Admins export fact-check history for analysis.

## 4. System Architecture

**User (Web browser)** → Django Views →

* Input validation
* Search APIs (Tavily, Serper)
* AI Summarization (Groq/OpenAI)
* Translations (Translate module)
* DB Persistence (Factcheck model)  
  → **Preview Page** → User

## 5. Prerequisites & Dependencies

* **System Requirements**
  + Python 3.10+
  + PostgreSQL 12+
  + Git & Virtualenv
  + VS Code or preferred IDE
* **Python Dependencies** (requirements.txt excerpt)
* django
* djangorestframework
* openai
* langchain
* langchain-groq
* tavily-python
* translate
* google-serper
* nltk
* requests
* gunicorn
* psycopg2-binary

## 6. Account Setup & Credentials

Required Services:

1. **GitHub**: Repo hosting (Brain-Builders/myaifactcheck)
2. **OpenAI**: API key for GPT models
3. **Groq**: API key for LLaMA models
4. **Tavily**: API key for web search
5. **Serper**: API key for Google results
6. **Railway**: Hosting and Postgres DB
7. **QServers**: Custom domain and SSL

## 7. Development Environment Setup

git clone https://github.com/Brain-Builders/myaifactcheck.git

cd myaifactcheck

python -m venv venv

source venv/bin/activate # Windows: venv\Scripts\activate

pip install -r requirements.txt

cp .env.example .env # add API keys

python manage.py migrate

python manage.py runserver

## 8. Database Setup

**Models:**

* **Factcheck:** stores claim, result, sources, sentiment.
* **UserReport:** stores community feedback.

**Export:** /export-factchecks/ generates a CSV for admins.

## 9. Search-Augmented Fact-Checking

1. User submits claim.
2. **Tavily + Serper** fetch web results.
3. Context compiled.
4. **Groq/OpenAI LLM** produces verdict.
5. Verdict + sources stored in DB.
6. If needed, text is **translated** into user’s language.

## 10. Application Structure

myaifactcheck/

├── app/

│ ├── views.py # Fact-check logic, translations, exports

│ ├── models.py # Factcheck, UserReport

│ ├── templates/ # HTML templates (index, preview, multilingual)

│ ├── serializers.py # DRF serializers

│ └── urls.py # Routes

├── myaifactcheck/

│ ├── settings.py

│ └── wsgi.py

├── manage.py

└── requirements.txt

## 11. Deployment Process

* **Railway**
  + Connect GitHub repo
  + Add environment variables (.env)
  + Deploy Django app
  + Use Railway Postgres
* **QServers (Domain)**
  + Point DNS → Railway URL
  + Enable SSL (Railway issues certificates)
* **Procfile**
* web: gunicorn projectname.wsgi --log-file -

## 12. How to Use MyAIFactCheck

* **End-Users:**
  1. Visit homepage.
  2. Enter a claim or use speech input.
  3. Receive verdict + explanation.
  4. Switch interface to Hausa, Yoruba, Igbo, French, Arabic, or Swahili.
* **Admins:**
  1. Access /all-factchecks/ to see records.
  2. Export CSV for analysis.
  3. Monitor Railway logs.

## 13. Challenges & Solutions

* **High API costs** → caching + limiting requests.
* **Translation errors** → chunking long texts.
* **Inconsistent verdicts** → fallback to secondary LLM.
* **Timeouts** → optimize search calls.

## 14. Testing & Monitoring

* **Unit Tests:** Django tests + Pytest.
* **Manual Testing:** verify multilingual pages.
* **Monitoring:** Railway logs, API dashboards, DB health.

## 15. Maintenance & Updates

* **Weekly:** Check logs, costs.
* **Monthly:** Update deps, system prompts.
* **Quarterly:** Rotate API keys, security audit.

## 16. Troubleshooting Guide

* **Empty result:** check .env API keys.
* **API 429 errors:** rate limit exceeded.
* **Translation mismatch:** retry or fallback English.
* **CSV issues:** open with UTF-8 encoding.
* **DB errors:** confirm DATABASE\_URL + run migrate.