Setup Guide

This guide walks you through a clean, sequential setup from zero to running the app locally.

Prerequisites

- Docker Desktop (for PostgreSQL)
- Node.js 18+
- Python 3.10+
- XeLaTeX (MacTeX/TeX Live) for PDF generation

1) Start PostgreSQL

```
# Run a local Postgres container (first time only)
docker run --name fairtestai-db \
  -e POSTGRES_PASSWORD=postgres \
  -e POSTGRES_DB=ftai \
  -p 5432:5432 -d postgres:15

# Verify container is running
docker ps --filter name=fairtestai-db

# Verify DB responds (non-interactive)
docker exec fairtestai-db psql -U postgres -d ftai -c 'SELECT 1 AS ok;'
```

What you should see:

- The `docker ps` output should include a `postgres:15` container named `fairtestai-db` with `0.0.0.0:5432->5432/tcp`.
- The `psql` command should print a table with a single row `ok | 1`. By default the backend expects:

```
postgresql+psycopg://postgres:postgres@localhost:5432/ftai
```

2) Configure environment variables

Create `backend/.env` (backend reads this at startup):

```
DATABASE_URL=postgresql+psycopg://postgres:postgres@localhost:5432/ftai
OPENAI_API_KEY=sk-...                  # required if you want LLM evaluation
ENABLE_LLM=1                           # set 0 to disable LLM evaluation during testing
USE_OCR=0                              # set 1 to use OCR pipeline (requires OpenAI vision)

# Code Glyph (C+G) attack font configuration
CODE_GLYPH_FONT_MODE=prebuilt
CODE_GLYPH_PREBUILT_DIR=<absolute-path-to-fonts>
# Optional base font embedding (improves rendering)
# CODE_GLYPH_BASE_FONT=/absolute/path/DejaVuSans.ttf
```

Optionally, create a root `.env` for the frontend (Vite injects at build time):

```
GEMINI_API_KEY=...
```

3) Place the Code Glyph fonts (for C+G attack)

- Put your prebuilt fonts under: `backend/data/prebuilt_fonts/`
- Example (ships with repo): `backend/data/prebuilt_fonts/DejaVuSans/v4/`
- Set `CODE_GLYPH_PREBUILT_DIR` in `backend/.env` to the absolute path, e.g.:

```
CODE_GLYPH_PREBUILT_DIR=/Users/<you>/path/to/repo/backend/data/prebuilt_fonts/DejaVuSans/v4
```

4) Backend installation

```
cd backend
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt

# Run DB migrations
export FLASK_APP=run.py
flask db upgrade

# Verify Alembic head is applied
flask db current
```

What you should see:

- `flask db upgrade` ends without errors.
- `flask db current` prints a revision hash (e.g., `head (310bfa3111f6)`), not None.
- Optional deeper check:

```
flask shell -c "from app.models import Assessment; print('Assessments:', Assessment.query.coun
```

It should print `Assessments: 0` on a clean DB. Tip (macOS): Port 5000 is often used by AirPlay. This project uses 5001.

5) Run the backend (in its own terminal)

```
cd backend
source .venv/bin/activate
python -c "from app import create_app; app = create_app(); app.run(debug=True, host='0.0.0.0',
```

API base: `http://127.0.0.1:5001`

Verify the API is up:

```
# Status code only
curl -s -o /dev/null -w "%{http_code}\n" http://127.0.0.1:5001/api/assessments
# Response preview (first 200 chars)
curl -s http://127.0.0.1:5001/api/assessments | head -c 200; echo
```

What you should see:

- A `200` status code.
- JSON output, typically including keys like `total` and `items`.

6) Frontend installation

```
# From the repo root (next to package.json)
npm install
```

7) Run the frontend (in a separate terminal)

```
npm run dev
```

Frontend dev server: `http://localhost:5173`

8) End-to-end test

- Open the app at `http://localhost:5173`
- Upload a question paper PDF (answer key optional)
- Choose an attack and click Run

CLI sanity check for upload (should return 201):

```
curl -s -S -X POST \
  -F "original_pdf=@/absolute/path/to/your.pdf;type=application/pdf" \
  -F "attack_type=Hidden Malicious Instruction (Prepended)" \
  -w "\nHTTP %{http_code}\n" \
  http://127.0.0.1:5001/api/assessments/upload
```

What you should see:

- A `HTTP 201` at the end, and JSON like `{ "assessment_id": "<uuid>" }`.
- Then download outputs (replace `$AID` with the returned id):

```
AID=<returned-uuid>
# Attacked PDF
curl -s -o /dev/null -w "%{http_code}\n" http://127.0.0.1:5001/api/assessments/$AID/attacked
```

```
# Reference report
curl -s -o /dev/null -w "%{http_code}\n" http://127.0.0.1:5001/api/assessments/$AID/report
```

Both should return `200`.

Notes

- To disable evaluation during development, set `ENABLE_LLM=0` in `backend/.env` and restart the backend.
- Output artifacts are written under `backend/data/assessments/<uuid>/` and mirrored to `backend/output/` for convenience.

Export these docs to PDF

From the `docs/` folder, run:

```
npm install
npm run build-pdf
```

This will produce `SETUP.pdf` and `SYSTEM_OVERVIEW.pdf` in the same folder.