AURA

Requirements Document

1. Overview

This system processes academic transcripts uploaded via a frontend web interface and generates performance alerts (e.g., low grade or attendance) using a Python backend. It integrates with Firebase Realtime Database for storing settings and pushing alerts, and supports download of processed transcript files.

2. Components

2.1 Frontend (HTML/JavaScript)

File: index.html

Description: Provides a file upload interface for .xlsx transcript files.

Key Features:
- Accepts .xlsx files
- Sends files to process.php via fetch API
- Displays processing result and a download link to the processed file

2.2 Backend PHP Interface

File: process.php

Description: Acts as a bridge between the HTML frontend and the Python backend.

Key Features:
- Receives uploaded file from frontend
- Executes app.py using shell_exec()
- Extracts generated ZIP filename from output
- Responds with JSON indicating success and filename

2.3 Python Application

a) app.py
Purpose: CLI script to drive the processing.
Responsibilities:
- Takes .xlsx file path as input

- Calls process_transcripts() and push_alerts_to_realtime_db()
- Zips all generated transcript files
- Prints a PROCESSED_FILE::<filename> line for PHP to capture

b) process_transcripts.py
Purpose: Processes transcript .xlsx file.
Responsibilities:
- Extracts student data from "Raw" sheet
- Applies template-based formatting
- Validates attendance and grade thresholds
- Generates alert JSON entries

c) firebase_utils.py
Purpose: Firebase integration logic.
Responsibilities:
- Fetches dynamic thresholds (min_grade, min_attendance) from Firebase
- Pushes alert data to Firebase under Alerts_test/{file_name}

3. Functional Requirements

ID	Requirement	Description
FR1	Upload Form	Users must be able to upload .xlsx files via the browser.
FR2	PHP Bridge	PHP must call app.py and return JSON response with status and filename.
FR3	Alert Generation	Python must analyze transcripts and generate alerts based on thresholds.
FR4	Firebase Integration	Thresholds must be fetched dynamically from Firebase. Alerts must be pushed.
FR5	Download Processed Files	User must be able to download a ZIP of processed files.

4. Non-Functional Requirements

- Compatibility: System must work in major browsers and on local PHP+Python server.

- Modularity: Backend logic must be modular: thresholds, alerts, and transcript logic separated.

- Security: Only .xlsx files are allowed. User-uploaded filenames must be sanitized.

5. Dependencies

- Python 3.x
- PHP 7+
- Firebase Realtime Database
- Python Libraries:
  - openpyxl
  - requests

6. Installation Requirements

Python requirements.txt content:

openpyxl
requests

7. Firebase Configuration

Database node `settings` must include:

```
{
  "min_grade": 50,
  "min_attendance": 75
}
```

Alerts will be stored under: Alerts_test/{sanitized_file_name}/alert_1, alert_2, ...