SBAC Attendance System README

====================================================

1. Project Overview

This project automates extraction of attendance logs from ZKTeco biometric devices across 120+ SBAC Bank branches, inserts them into a central Microsoft Access database (att2000.mdb), and stores a deduplicated copy in a Flask/MySQL backend. Key features:

* **Reliable Data Capture**: Poll devices concurrently using IP ranges/subnets.
* **Dual-Persistence**: Logs saved both in Access and Flask DBs for auditing & integration.
* **Performance**: Bulk insertions for high-volume data, colored logging, elapsed-time metrics.
* **Scalability**: ThreadPoolExecutor for parallel device polling, configurable worker count.

1. Folder Structure

attendance-system/  
├── backend/  
│ ├── app/  
│ │ ├── \_\_init\_\_.py # App factory & extension init  
│ │ ├── config.py # Config: DB URIs, polling intervals  
│ │ ├── models.py # SQLAlchemy models: Branch, Device, AttendanceLog  
│ │ ├── views/  
│ │ │ ├── devices.py # CRUD endpoints for branches & devices  
│ │ │ └── logs.py # Polling endpoint and health check  
│ │ ├── tasks.py # Poll scheduler, fetch/forward logic, Access & bulk DB inserts  
│ │ └── seed\_data.py # CSV/Excel-driven seeder for Branch & Device tables  
│ ├── migrations/ # Flask-Migrate scripts  
│ ├── requirements.txt # Python dependencies  
│ └── wsgi.py # Production entry point (gunicorn/uWSGI)  
└── frontend/  
 ├── public/  
 ├── src/  
 │ ├── api/ # axios instance & API wrappers to Flask backend  
 │ ├── components/ # React components (BranchList, LogsTable, Dashboard)  
 │ ├── contexts/ # React Context (Auth, WebSocket)  
 │ ├── App.jsx # Top-level React component  
 │ └── index.jsx # ReactDOM bootstrap  
 └── package.json # Frontend dependencies

1. Python Dependencies

Listed in backend/requirements.txt:

Flask  
Flask-SQLAlchemy  
Flask-Migrate  
apscheduler  
pyzk  
pyodbc  
pandas  
openpyxl  
colorama

1. Setup & Run Backend
2. **Clone repo** and cd attendance-system/backend
3. **Create & activate venv**:

* python -m venv venv  
  source venv/bin/activate # Linux/macOS  
  source ./venv/Scripts/activate # Windows

1. **Install dependencies**:

* pip install -r requirements.txt

1. **Configure Access DB Path** in app/config.py or tasks.py (att2000.mdb).
2. **Initialize Flask DB** (MySQL or SQLite):

* flask db init  
  flask db migrate -m "Initial"  
  flask db upgrade

1. **Seed** Branches/Devices:

* flask shell  
  >>> exec(open('app/seed\_data.py').read())

1. **Run**:

* flask run
* Scheduler auto-starts and polls devices every POLL\_INTERVAL seconds.

1. Frontend Setup

frontend/

├── public/

│ └── index.html

├── src/

│ ├── api/

│ │ └── attendanceApi.js # Axios instance and calls to Flask

│ ├── components/

│ │ ├── Dashboard.jsx # Top-level dashboard layout

│ │ ├── StatCard.jsx # Clickable card for branch/device/log counts

│ │ ├── BranchList.jsx # Modal or page to list/add/update/delete branches

│ │ ├── DeviceList.jsx # Modal/page for devices

│ │ ├── LogsOverview.jsx # Total log count and controls

│ │ ├── DeviceStatusTable.jsx # Table of devices with online/offline, manual fetch buttons

│ │ └── SchedulerControl.jsx # Start/stop scheduler & display its logs

│ ├── contexts/

│ │ └── SchedulerContext.jsx # Provide scheduler status/logs via WebSocket or polling

│ ├── hooks/

│ │ └── useFetch.js # Reusable data-fetching hook

│ ├── App.jsx

│ └── index.jsx

└── package.json

cd ../frontend  
yarn install # or npm install  
yarn start # or npm start

* Connects to Flask at http://localhost:5000/api
* Dashboard shows branches, device status, logs in real time.

1. Key Design Decisions

* **ThreadPoolExecutor**: Parallel device polling to handle 120+ branches concurrently.
* **pyodbc**: Direct Access DB writes for legacy integration without HTTP overhead.
* **Bulk Save**: db.session.bulk\_save\_objects() for high-volume inserts to Flask DB.
* **Colorama logs**: Colored terminal output for faster issue diagnosis.

1. Future Enhancements

* **WebSocket**: Push live status & logs to frontend.
* **Retry logic**: Exponential backoff on device timeouts.
* **Monitoring**: Integrate Prometheus/Grafana for scheduler metrics.
* **Dockerization**: Containerize backend & frontend for easy deployment.

====================================================