HOTEL & RESTAURANT REVIEW MANAGEMENT SYSTEM

Project Structure & Architecture Report

Version 1.0 – November 2025

Table of Content

1. System Overview………………………………………………………………...………..page 2
2. Technology Stack ...………………………………………………………...…….…page 3
3. High-Level Architecture..……….…………………………………….………...………page 4
4. Data Flow Summary…………………………………………………………...……..page 5
5. Project Folder Structure..……...……………………………………………………...page 6
6. Summary……………...………………………………………………………………..page 7

Team LETHYS

IN2901 - Software Development Project

Faculty of Information Technology

University of Moratuwa

1. System Overview

The Hotel & Restaurant Review Management System is a hybrid cloud–desktop software solution developed to automate the collection, analysis, and visualization of customer feedback posted on major online platforms such as Booking.com, Agoda, and TripAdvisor.

The system provides:

- a centralized cloud infrastructure that manages scraping, synchronization, authentication, and analytics; and
- a local intelligent client application that operates independently within each hotel, capable of analyzing reviews offline using a lightweight AI model.

This architecture ensures high availability and scalability on the cloud while delivering responsive, AI-powered insight at the client end.
The design follows modern micro-service practices with strict separation of concerns between local and cloud components.

2. Technology Stack

Layer	Technology / Tool	Description
Frontend (Local & Admin)	React.js, TailwindCSS, Chart.js	Interactive dashboards and visualization interfaces
Desktop Shell	Electron	Bundles the local frontend + backend as a single installable client.
Local Backend	FastAPI (Python)	Provides APIs for data synchronization and local AI inference.
Local AI Model	MiniLM-L6-v2	Lightweight transformer for sentiment & topic detection (CPU-optimized).
Local Database	SQLite	Offline caching and persistence of review data.
Cloud Backend	FastAPI + Playwright + Celery + Redis	Central API hub, scraping automation, and scheduled job execution.
Cloud Database	Microsoft SQL Server	Structured, multi-tenant storage for reviews and analytics.
Authentication	Auth0 (OAuth 2.0 / JWT)	Secure identity management and role-based access control.
Hosting / Deployment	Docker + AWS	Containerized deployment for scalability and reliability.

3. High-Level Architecture

The platform consists of two integrated environments:

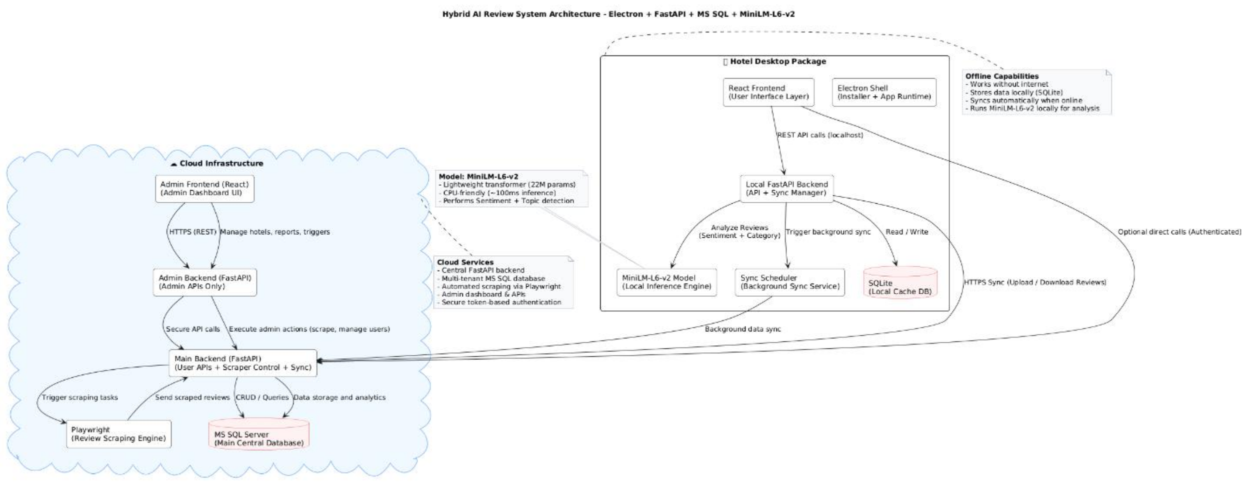
1. **Local Environment (Tenant Side)**

- Electron Application**:**
  - Delivers a seamless desktop experience combining the React interface with the embedded FastAPI backend.
- Local FastAPI Server:
  - Orchestrates communication between UI, local cache, and AI model.
- MiniLM-L6-v2 Model:
  - Performs on-device sentiment classification and topic categorization.
- SQLite Database:
  - Stores reviews and analysis results for offline access.
- Background Sync Service:
  - Automatically synchronizes data with the cloud whenever connectivity is available.

2. **Cloud Environment**

- Main Backend (FastAPI):
  - Provides secure REST APIs, manages scheduling, scraping, and synchronization.
- Playwright Scraper Module:
  - Automates review retrieval from supported platforms.
- Celery Scheduler + Redis:
  - Executes periodic scraping and update tasks, including user-defined schedules.
- Admin Backend & Dashboard:
  - Enables system administrators to oversee tenants, scraping jobs, and aggregated analytics.
- MS SQL Database:
  - Stores all normalized review data and analytics results.

A unified Auth0 identity layer authenticates both local and cloud requests, ensuring secure communication across environments.


Hybrid AI Review System Architecture - Electron + FastAPI + MS SQL + MiniLM-L6-v2
Cloud Infrastructure
Admin Frontend (React) (Admin Dashboard UI)
HTTPS (REST)
Manage hotels, reports, triggers
Admin Backend (FastAPI) (Admin APIs Only)
Secure API calls
Execute admin actions (scrape, manage users)
Main Backend (FastAPI) (User APIs + Scraper Control + Sync)
Trigger scraping tasks
Send scraped reviews
CRUD / Queries
Data storage and analytics
Playwright (Review Scraping Engine)
MS SQL Server (Main Central Database)
Model: MiniLM-L6-v2
- Lightweight transformer (22M params)
- CPU-friendly (~100ms inference)
- Performs Sentiment + Topic detection
Cloud Services
- Central FastAPI backend
- Multi-tenant MS SQL database
- Automated scraping via Playwright
- Admin dashboard & APIs
- Secure token-based authentication
Hotel Desktop Package
React Frontend (User Interface Layer)
Electron Shell (Installer + App Runtime)
REST API calls (localhost)
Local FastAPI Backend (API + Sync Manager)
Analyze Reviews (Sentiment + Category)
Trigger background sync
Read / Write
MiniLM-L6-v2 Model (Local Inference Engine)
Sync Scheduler (Background Sync Service)
SQLite (Local Cache DB)
Offline Capabilities
- Works without internet
- Stores data locally (SQLite)
- Syncs automatically when online
- Runs MiniLM-L6-v2 locally for analysis
HTTPS Sync (Upload / Download Reviews)
Optional direct calls (Authenticated)
Background data sync

4. Data Flow Summary

1. Scraping Phase (Cloud):
   - Playwright retrieves raw reviews from external platforms under the control of Celery scheduler tasks.
   - Reviews are normalized and stored in the central MS SQL database.
2. Synchronization Phase (Local ↔ Cloud):
   - The hotel's local backend periodically requests new reviews through secure REST APIs. Data is cached in SQLite and made available to the local UI.
3. Analysis Phase (Local AI):
   - The MiniLM model processes each review, classifying sentiment and identifying service categories (e.g., staff, room, food).
   - Results are stored locally and then synced back to the cloud.
4. Visualization Phase:
   - Both local and admin dashboards display metrics and trends derived from the analyzed data.

5. Project Folder Structure

```
hotel-review-system/
│
├── client_app/                 # Local hotel application
│   ├── electron/               # Electron shell & build configs
│   ├── frontend/               # React UI components
│   ├── backend/                # FastAPI services (AI + Sync)
│   │   ├── api/
│   │   ├── core/
│   │   ├── services/
│   │   ├── ml/
│   │   └── db/
│   ├── sqlite_cache/           # Local SQLite database
│   ├── installer/              # Packaging scripts
│   └── README.md
│
├── cloud_backend/              # Cloud infrastructure
│   ├── main_backend/           # APIs, scraping, scheduling
│   ├── admin_backend/          # Admin API endpoints
│   ├── admin_frontend/         # React admin dashboard
│   ├── database/               # Schema & migrations
│   └── README.md
│
├── common/                     # Shared documentation & scripts
│   ├── docs/
│   ├── scripts/
│   ├── .env.example
│   └── LICENSE
│
└── README.md
```

Each top-level module functions independently and communicates through defined APIs, enabling parallel development and simplified maintenance.

6. Summary

This phase of the project defines a robust, scalable structure combining local AI capabilities with cloud automation and central management.
The design ensures secure, authenticated communication, continuous data collection, and offline usability—laying the foundation for subsequent phases that will incorporate CI/CD pipelines, monitoring, and analytics services.