Author: TS
Date: 11-Aug-25

HAILIE Insights Engine

The Application:

*"A modern, scalable, web application built using a **three-tier architecture** consisting of a frontend (client-side), a backend (server-side), and a database. For this application, a **cloud-native approach** is used; leveraging managed services to enhance scalability, reliability, and security while reducing operational overheads."*

Functional Requirements

Functional requirements are the core features and capabilities of an application—*what* it *does*.

For the described web application, these would include:

- **User Authentication and Authorization:**
  - A secure registration and login system for housing providers and professionals.
  - Role-based access control (RBAC) to ensure users can only access and query data they are authorized to see. For example, a user from one housing association should not be able to see data from another unless explicitly permitted.
- **Data Upload:**
  - A user-friendly interface for uploading Excel (.xlsx) and CSV (.csv) files.
  - Robust validation to ensure the uploaded data structure conforms to the expected format. This includes checking for correct column headers, data types, and required fields.
  - Clear error messaging to guide users if their upload fails validation.
- **Data Import and Processing:**
  - A backend process to parse the uploaded files and map the data to your custom database schema.
  - The ability to handle large datasets without timing out. This might involve asynchronous processing where the user is notified once the import is complete.
- **Database:**
  - A custom database schema designed to efficiently store and query the

Co-created with Google Gemini 2.5 Pro and Claude Sonnet 4

housing data. This schema should be flexible enough to accommodate potential future changes to the data structure.

- **Data Analysis and Querying:**
  - A user interface that allows users to build and execute queries. This could range from a simple form-based query builder for common questions to a more advanced interface for complex data analysis.
  - The ability to filter, sort, and aggregate data.
  - A results page that displays the query output in a clear and understandable format, such as tables, charts, and graphs.
- **Data Visualization and Insights:**
  - The platform should be able to generate visualizations (e.g., bar charts, pie charts, line graphs) to help users understand the data.
  - The ability to identify and present key insights from the data, such as trends, anomalies, and correlations.
- **Export and Reporting:**
  - Functionality for users to export query results and reports in various formats (e.g., CSV, PDF).

Non-Functional Requirements ("-ilities")

These are the quality attributes of your system—how well it performs its functions. They are critical for user satisfaction and the long-term success of your application.

- **Scalability:**
  - The application should be able to handle a growing number of users, larger datasets, and more complex queries without a degradation in performance. This will influence your choice of technologies and cloud hosting provider.
- **Accessibility:**
  - The application should be usable by people with disabilities. This means adhering to Web Content Accessibility Guidelines (WCAG), ensuring the application is navigable by keyboard, providing text alternatives for non-text content, and ensuring good colour contrast.
- **Security:**
  - As you are handling potentially sensitive data, security is paramount. This includes:
    - **Data Encryption:** Data should be encrypted both in transit (using HTTPS) and at rest (in the database).
    - **Secure Authentication:** Implementing strong password policies and

considering multi-factor authentication.
    - **Data Privacy:** Complying with UK data protection regulations, such as the GDPR and the Data Protection Act 2018.
- **Usability:**
    - The application should be intuitive and easy to use. A clean, well-designed user interface (UI) and a positive user experience (UX) are crucial for adoption.
- **Reliability and Availability:**
    - The application should be available and functioning correctly when users need it. This involves minimizing downtime through robust infrastructure and having a plan for disaster recovery.
- **Performance:**
    - The application should be fast and responsive. This includes quick page load times, fast data uploads, and efficient query execution.
- **Maintainability:**
    - The application should be designed and built in a way that makes it easy to fix bugs, add new features, and upgrade components in the future. This involves writing clean, well-documented code and following good software architecture principles.
- **Interoperability:**
    - Consider if the application needs to integrate with other systems in the future. For example, you might want to connect to other housing management software via APIs.

By considering these functional and non-functional requirements from the outset, HAILIE will be well-positioned to design a robust, secure, and user-friendly web application that meets the needs of the target audience.