Managing and Presenting Planning Application Decisions: A Flask and SQLite Web Application

This Flask application is designed to manage and present data for decision making on planning applications for large and small developments in counties across England. By integrating ID tables and categorised data, the application provides a user-friendly interface for browsing, querying and analysing data. As the volume of data continues to grow, the application has been designed to enable decision makers, researchers and the public to easily access and understand this information. The application uses a SQLite database to manage the data and provides a web interface using the routing, templates and forms capabilities of the Flask framework. Data is imported into the SQLite database via CSV files, retrieved using SQL queries and presented in tabular form on the web page.

Key features include: a home page view of all planning application records; a detail view, where users can click on a specific feature ID to view more categorised information; and a linked data view, which provides a statistical view of the data by linking the ID table to the categorised table. The requirements for installing and using the application include Python 3.x, Flask and SQLite3. It is necessary to prepare the database using SQLite3 and the provided CSV file, and access the specified port in the browser to view the application interface after starting the server by running the Flask application.

To ensure the long-term stability of the application and a good user experience, maintenance work includes code updates, database maintenance, performance tuning, security updates, gathering user feedback, regular backups and documentation updates. The application renders its interface using the following URLs: home (/), details (/show/<Feature_ID>) and associations (/associations/<Feature_ID>).

Example 1 shows how to connect to a SQLite database and retrieve data from a database table in a Flask application. The get_connection function is defined to connect to the Planning_Applications_Decisions.db database, and the get_data_from_db function allows an SQL query to be executed and all records from the specified table to be retrieved.

Example 2 @app.route shows how dynamic routing can be used to respond to a user request to view the details of a particular Feature ID. The database function is used to retrieve the data associated with this Feature ID and the render_template function is used to render an HTML template, passing the retrieved data as parameters to the template to be displayed on a web page.

With these implementation and maintenance details, this project not only demonstrates how to develop a web application using Flask and SQLite, but also illustrates the challenges and solutions of handling and presenting large amounts of data.