**Implementation of SQLite Database in RShiny Application**

**Introduction**

This document outlines the implementation of an SQLite database in an RShiny application. The objective was to compare different databases and determine the most suitable one for the Shiny Proxy homepage. After comparing PostgreSQL, SQL Server, and SQLite, it was determined that SQLite would be the ideal choice due to its serverless nature and minimal dependencies.

**Database Comparison**

A thorough comparison was conducted among different databases to evaluate their suitability for the Shiny Proxy homepage. The following table summarizes the key aspects of each database:

**Implementation Steps**

The implementation of SQLite in the RShiny application involved the following steps:

**Package Usage:** The R packages DBI, RSQLITE, and dplyr were utilized for efficient database interactions and connections within the RShiny application.

**Database Creation**: A SQLite database was created to store the Projects information. The data was initially sourced from an Excel file and subsequently transferred to the SQLite database. The data included textual information and project logos.

The textual data, containing project details, was copied to the database with the project\_title serving as the primary key.

The project logos, being in image format, were converted to Blob objects and saved as blobs in the database. Read and write binary operations were employed, along with parameterized SQL queries, to insert the project\_logo data.

To maintain image references, an additional column named "project\_logo\_name" was created to store the actual image file names (including extensions) for easy retrieval within the application.

**Application Integration:** The SQLite database was seamlessly integrated into the RShiny application. The existing data loading mechanism, which initially read from an Excel file, was replaced with data retrieval from the SQLite database.

After replacing the data source, the application was quality checked. An issue was discovered in the Project Overview section, where data frame assignmentsare done for plotly click events resulted in errors. Upon debugging, it was found that the loaded data from the database had a different class compared to the data loaded from the Excel file. To resolve this, the data loaded from the database was converted to the class "tibble," resolving the problem.

As per the Shiny Proxy mechanism, all project\_logo images are referred to from the application's file system (www/ folder). To ensure proper referencing, the project logos were written from the database back to the application's file system. This ensured that the application could retrieve the project logos directly from the database.

Another challenge encountered was with rendering the rhandson table. The project\_logo data, stored as Blob objects, occupied a significant amount of space in table rows. This caused issues with row visibility, as adjacent rows were not properly displayed. Attempts to address this issue, such as modifying row heights, column widths, and applying manual CSS stylings, proved unsuccessful due to limitations of the rhandson table package.