Datastore design decisions

We have made several design choices in our application to interact with the database, specifically PostgreSQL, for storing persistent data such as notes and user accounts. These design choices, including the use of the jackc/pgx database driver, reflect a balance between data integrity, security, and performance, all of which are critical considerations for a production-ready application. Here are the key design choices we made:

Database Choice: We have chosen PostgreSQL as the relational database management system to store our data. PostgreSQL is known for its reliability, robustness, and support for advanced data types. This choice is suitable for handling structured data like user accounts and notes.

Data Modelling: We designed our database schema to reflect the structure of the application's data. This involves creating tables for notes, users, and user shares. By creating foreign key constraints, we have ensured data integrity between related tables.

Database Initialization: We have implemented a database initialization process that includes the creation of tables and the insertion of initial data, such as user accounts. Our initialization process checks to see if the initial import has already been performed, and if it has, it skips this initial step. By creating this automated setup for a fresh database, we are ensuring consistency in the database.

Prepared Statements and SQL Queries with jackc/pgx: We are using prepared statements and raw SQL queries to interact with the database, facilitated by the jackc/pgx database driver. Prepared statements help prevent SQL injection attacks and improve query performance.

User Password Hashing: To enhance security, we are hashing user passwords using the bcrypt algorithm before storing them in the database. This protects user credentials against data breaches.

Full-Text Search (FTS) Implementation: In our application, we have implemented FTS to enable search functionality. FTS allows efficient searching of notes and tasks. It's particularly useful for searching through large volumes of text-based data, such as notes and descriptions. To implement FTS in our application, we've added a fts_text column to the notes table. This column is populated with a text representation of the data to be searched.