Guiding Through OpenWebUI DB

Overview

These are the instructions on how to access the OpenWebUI SQLite3 DB and look through it.

Using this you will be able to see the different tables and schemas of those tables. If there is a problem that arises with a feature using the DB then this may help!

Prerequisites

Before starting, make sure you have the following:

- Connected to the UW Madison VPN (Global Protect)
- VS Code is running the GCP VM
- You have read access to /opt folder (you should be able to have this if you have access to the VM)

Steps for Prerequisites

Step 1: Running VS Code

- Follow all the steps using from docs/vm_setup/vscode_ssh_documentation.pdf

Step 2: Access to /opt folder

- You should by default have read access to the /opt folder if you have access to the GCP VM.
- If not however, have the professor use this command:

```
sudo apt install acl
sudo setfacl -R -m u:<user>:r /opt/openwebui
```

  - Note: Change <user> with username

VS Code Terminal to query DB

Open a terminal in VS Code

- Note: Bash shell is preferred
- Type in these commands to tap into the db

```
@qasap-vm01:~$ cd /opt/openwebui/data
@qasap-vm01:/opt/openwebui/data$ sqlite3 webui.db
```

  - Output after running this:

```
SQLite version 3.40.1 2022-12-28 14:03:47
Enter ".help" for usage hints.
sqlite>
```

Run Basic Queries

- Note: SQLite3 commands: https://sqlite.org/cli.html

- View the tables

```
sqlite> .tables
alembic_version   config           knowledge         oauth_session
api_key           document         knowledge_file    prompt
auth              feedback         memory            tag
channel           file             message           tool
channel_member    folder           message_reaction  user
channel_webhook   function         migratehistory
chat              group            model
chatidtag         group_member     note
```

  - Note: The grading dashboard uses tables chat and user
    - user table – Has all of the user info with id as a primary key for other tables

```
sqlite> .schema user
CREATE TABLE IF NOT EXISTS "user" (
        id VARCHAR(255) NOT NULL,
        name VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL,
        role VARCHAR(255) NOT NULL,
        profile_image_url TEXT NOT NULL,
        created_at INTEGER NOT NULL,
        updated_at INTEGER NOT NULL,
        last_active_at INTEGER NOT NULL,
        username VARCHAR(50),
        bio TEXT,
        gender TEXT,
        date_of_birth DATE,
        profile_banner_image_url TEXT,
        timezone VARCHAR,
        presence_state VARCHAR,
        status_emoji VARCHAR,
        status_message TEXT,
        status_expires_at BIGINT,
        oauth JSON,
        info JSON,
        settings JSON
);
CREATE UNIQUE INDEX user_id ON user (id);
```

    - chat table – Has all the info of each chat given a user_id

```
sqlite> .schema chat
CREATE TABLE IF NOT EXISTS "chat" ("id" VARCHAR(255) NOT NULL, "user_id" VARCHAR(255) NOT NULL, "title"
TEXT NOT NULL NOT NULL, "share_id" VARCHAR(255), "archived" INTEGER NOT NULL, "created_at" DATETIME NOT
NULL NOT NULL, "updated_at" DATETIME NOT NULL NOT NULL, chat JSON, pinned BOOLEAN, meta JSON DEFAULT '{}
' NOT NULL, folder_id TEXT);
CREATE UNIQUE INDEX "chat_id" ON "chat" ("id");
CREATE UNIQUE INDEX "chat_share_id" ON "chat" ("share_id");
CREATE INDEX folder_id_idx ON chat (folder_id);
CREATE INDEX user_id_pinned_idx ON chat (user_id, pinned);
CREATE INDEX user_id_archived_idx ON chat (user_id, archived);
CREATE INDEX updated_at_user_id_idx ON chat (updated_at, user_id);
CREATE INDEX folder_id_user_id_idx ON chat (folder_id, user_id);
```