Project Final Report: Luxury Car Rental Reservation App

Data Management Systems

Vidurshan Sribalasuhabiramam - 100558257

Table of Contents

Abstract

This paper will detail the purpose of this project along with the planning, developing, management and deployment of the SQL database back-end with a web application front-end to create the Luxury Car Rental Reservation application and website interface. To accomplish this, we deployed a MySQL database alongside a Django web server which enabled queries to be written in a compact relational object-oriented manner. The development of the database was guided by requirements that were elicited during the ADD process along with decomposing Use-Cases into objectifiable components in the system, and furthermore by utilizing an entity-relationship (ER) diagram. Finally, tabular definitions were derived for the application database from the ER diagram.

Background

The foundational framework for this project is built off the freedom-loving, open-source Python, Django and SQL frameworks and software. In order to rapidly prototype and provide the end-user with a working demo, we chose to stick with this framework along with the fact that after the first iteration of Architecture Driven Design (ADD) Analysis was done, the best reference for web application architecture was the aforementioned above. The application works as such: User logins and can see all public reservations. If the user wants to add a reservation, they press the add reservation button and then can fill a form field with details along with a picture of their preferred vehicle. They are then able to mark the reservation as public or private and it will then be added to the database to be interfaced by the protocol that will manage the supply and demand of vehicle reservations.

The development stack that was chosen for this web application includes these main tools:

- ✓ Bootstrap: chosen for rapid layout implementation and responsive user interface/experience
- ✓ Python: chosen for rapid development and prototyping
- ✓ Django: chosen for MVC design pattern
- ✓ SQLite3: chosen for database reliability and security

Objectives

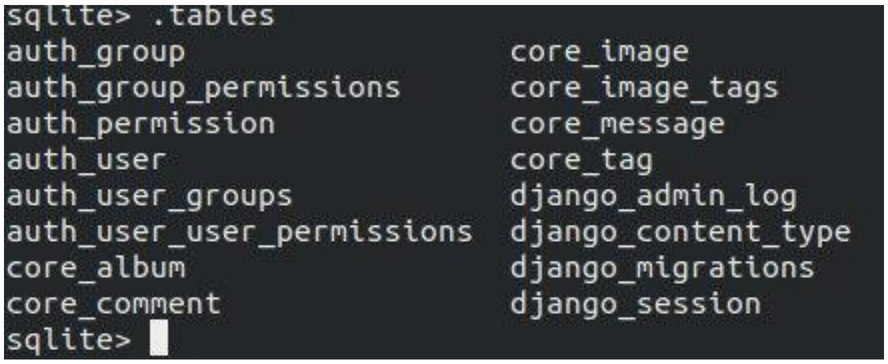
This Luxury Car Rental Reservation App was created with two primary use-cases in mind. UC-1 and UC-2, concerning user functionality and user view upon login were addressed in the development of this project. The first was to ensure that the app was usable by any average person. There are other (luxury) car reservation apps and protocols that can interface supply and demand for personal mobilization and transportation. The intent behind our app is to allow the user to seamlessly interface with the database and make reservations with minimal friction. To accomplish this, only bare essential information is asked of and displayed to the user in order to ensure the application is intuitive and easy to experience. The intent is to reduce user friction and ease experience when making luxury car rental reservations.

The second constraint in mind is that the site should be freedom and open-source and as such the application and protocol should be free to the end-user client but any fees necessary to conduct transactions will be covered by network and participant agents/actors of the network. Most car reservation apps such as Uber and Lyft charge excessive fees to the customer and driver increasing user friction. Our target audience will be customers dissatisfied with Uber, Lyft, etc. and willing to participate in a software protocol that connects supply and demand together by eliminating middlemen. In order to make transit accessible to everyone, the cost of entry and barrier to entry should be marginally minimal enough to not dissuade the customer upon first experience.

Database Management

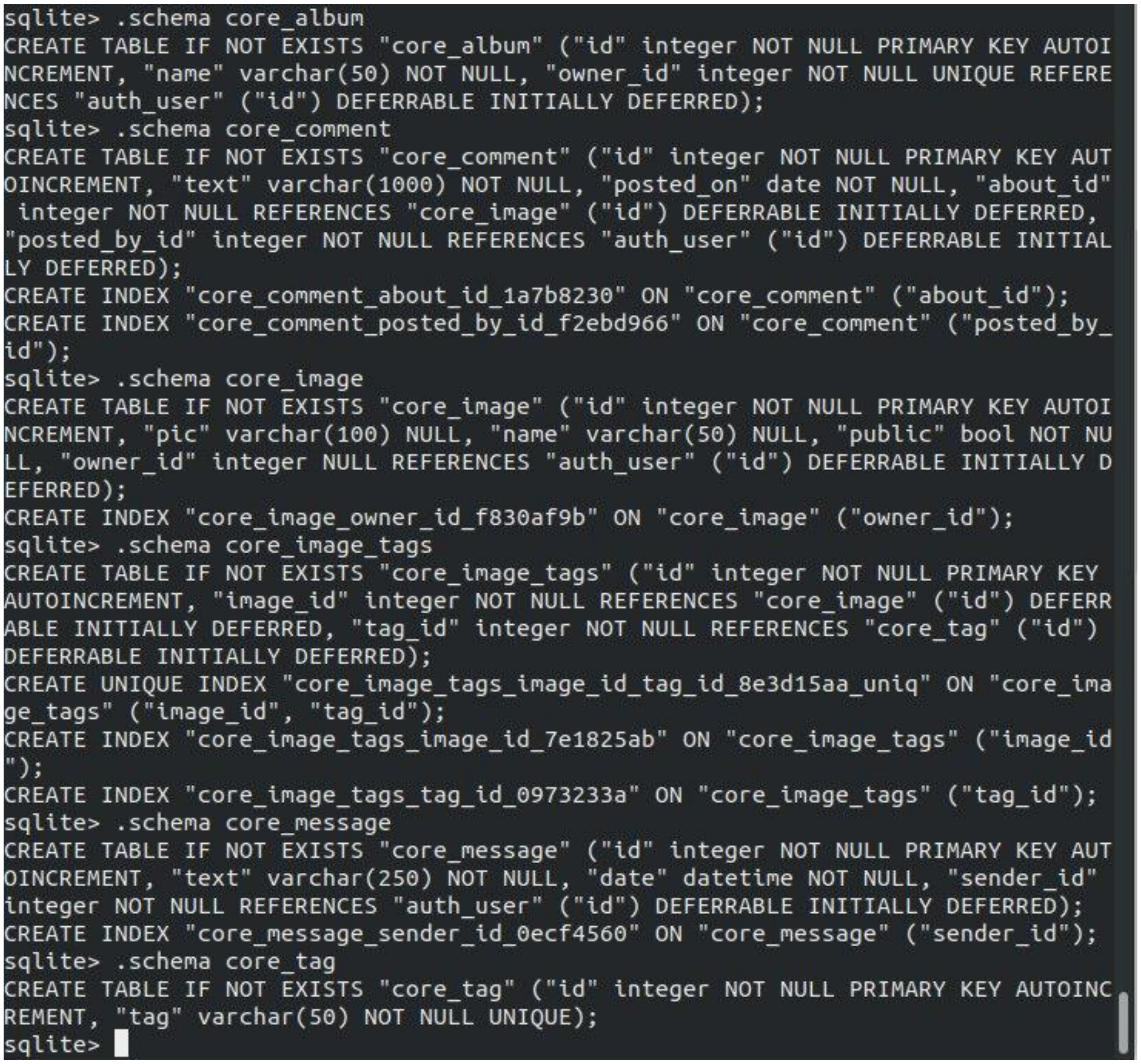
Database Structure

The database used for this project consists of 17 SQL3 tables. The core tables are the ones utilized by the core of the application which would be our reservation field form. The auth tables are all regarding group/user privileges/permissions. The Django tables are for any changes made to the tables in between instantiation, runtime and to manage admin sessions.



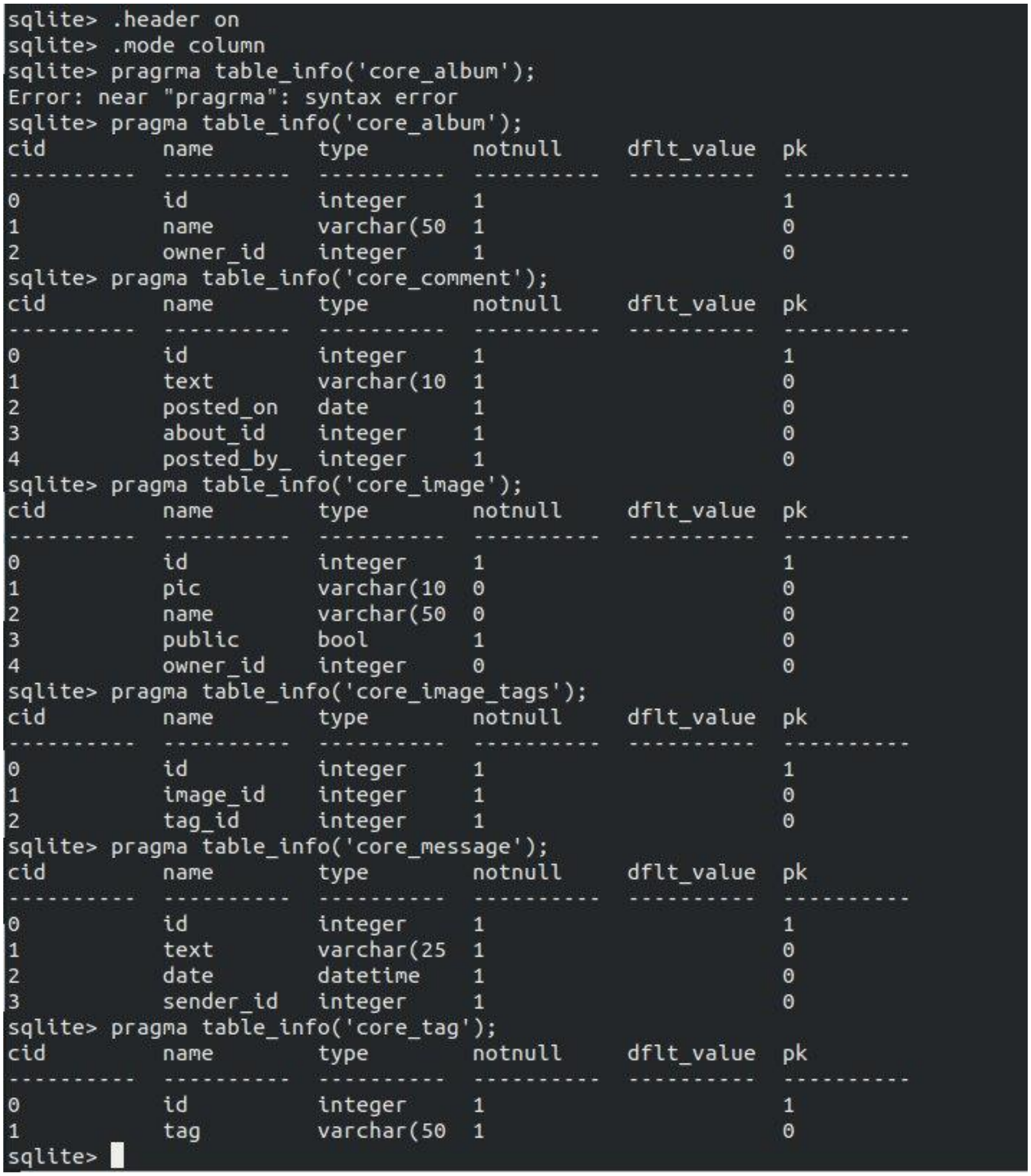
```
sqlite> .tables
auth_group                    core_image
auth_group_permissions        core_image_tags
auth_permission               core_message
auth_user                     core_tag
auth_user_groups              django_admin_log
auth_user_user_permissions    django_content_type
core_album                    django_migrations
core_comment                  django_session
sqlite>
```

This screenshot shows the schema of the tables titled: core_*



```
sqlite> .schema core_album
CREATE TABLE IF NOT EXISTS "core_album" ("id" integer NOT NULL PRIMARY KEY AUTOI
NCREMENT, "name" varchar(50) NOT NULL, "owner_id" integer NOT NULL UNIQUE REFERE
NCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED);
sqlite> .schema core_comment
CREATE TABLE IF NOT EXISTS "core_comment" ("id" integer NOT NULL PRIMARY KEY AUT
OINCREMENT, "text" varchar(1000) NOT NULL, "posted_on" date NOT NULL, "about_id"
 integer NOT NULL REFERENCES "core_image" ("id") DEFERRABLE INITIALLY DEFERRED,
"posted_by_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIAL
LY DEFERRED);
CREATE INDEX "core_comment_about_id_1a7b8230" ON "core_comment" ("about_id");
CREATE INDEX "core_comment_posted_by_id_f2ebd966" ON "core_comment" ("posted_by_
id");
sqlite> .schema core_image
CREATE TABLE IF NOT EXISTS "core_image" ("id" integer NOT NULL PRIMARY KEY AUTOI
NCREMENT, "pic" varchar(100) NULL, "name" varchar(50) NULL, "public" bool NOT NU
LL, "owner_id" integer NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY D
EFERRED);
CREATE INDEX "core_image_owner_id_f830af9b" ON "core_image" ("owner_id");
sqlite> .schema core_image_tags
CREATE TABLE IF NOT EXISTS "core_image_tags" ("id" integer NOT NULL PRIMARY KEY
AUTOINCREMENT, "image_id" integer NOT NULL REFERENCES "core_image" ("id") DEFERR
ABLE INITIALLY DEFERRED, "tag_id" integer NOT NULL REFERENCES "core_tag" ("id")
DEFERRABLE INITIALLY DEFERRED);
CREATE UNIQUE INDEX "core_image_tags_image_id_tag_id_8e3d15aa_uniq" ON "core_ima
ge_tags" ("image_id", "tag_id");
CREATE INDEX "core_image_tags_image_id_7e1825ab" ON "core_image_tags" ("image_id
");
CREATE INDEX "core_image_tags_tag_id_0973233a" ON "core_image_tags" ("tag_id");
sqlite> .schema core_message
CREATE TABLE IF NOT EXISTS "core_message" ("id" integer NOT NULL PRIMARY KEY AUT
OINCREMENT, "text" varchar(250) NOT NULL, "date" datetime NOT NULL, "sender_id"
integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE INDEX "core_message_sender_id_0ecf4560" ON "core_message" ("sender_id");
sqlite> .schema core_tag
CREATE TABLE IF NOT EXISTS "core_tag" ("id" integer NOT NULL PRIMARY KEY AUTOINC
REMENT, "tag" varchar(50) NOT NULL UNIQUE);
sqlite> 
```

Listed below are the records in each table



```
sqlite> .header on
sqlite> .mode column
sqlite> pragrma table_info('core_album');
Error: near "pragrma": syntax error
sqlite> pragma table_info('core_album');
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           id          integer     1                       1
1           name        varchar(50  1                       0
2           owner_id    integer     1                       0
sqlite> pragma table_info('core_comment');
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           id          integer     1                       1
1           text        varchar(10  1                       0
2           posted_on   date        1                       0
3           about_id    integer     1                       0
4           posted_by_  integer     1                       0
sqlite> pragma table_info('core_image');
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           id          integer     1                       1
1           pic         varchar(10  0                       0
2           name        varchar(50  0                       0
3           public      bool        1                       0
4           owner_id    integer     0                       0
sqlite> pragma table_info('core_image_tags');
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           id          integer     1                       1
1           image_id    integer     1                       0
2           tag_id      integer     1                       0
sqlite> pragma table_info('core_message');
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           id          integer     1                       1
1           text        varchar(25  1                       0
2           date        datetime    1                       0
3           sender_id   integer     1                       0
sqlite> pragma table_info('core_tag');
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           id          integer     1                       1
1           tag         varchar(50  1                       0
sqlite>
```

The *core_album* table contains all public/private reservations that are made on the site by any of the admins or users. The *core_comment* contains comments left by the users/admins on the reservation. The *core_image* contains the images per reservation. *Core_image_tags* contains the relations of tags that can be associated with the images in the reservations. *Core_message* is the messaging between the protocol admins, users and vehicle providers. *Core_tag* holds the tags that can be modified and attached to images.

Conclusions

The aim of this project is to provide a seamless frictionless environment where supply and demand can interact while providing the client with a low cost barrier to entry and access our service infrastructure. This system and design pattern could be utilized for a variety of private/public entities but would best shine under a universal transportation system protocol. Cloud-enabled AI agents would be able to access necessary information to provide services-as-a-service. This is very optimistic thinking and entirely possible under the right circumstances, but it will not be an overnight victory. There are several aspects that could be improved upon such as the UI being simply a few drag and drop clicks. Ideally a one touch or less than 5 touch point system would further provide the user with a simple seamless interface.