REPORT ON FINAL PROJECT

Date: 28th November 2021

Author

Chekuri Viroopaksh
21f1000374
21f1000374@student.onlinedegree.iitm.ac.in
I am a 3rd-year undergraduate student in metallurgy from IIT Madras.

Description

A complete web application that handles user registration, login. Every user can create any number of decks and can have any number of cards in them. A user can always update the decks and cards belonging to the user. A user can always communicate with the API of the project by reading the OpenAPI documentation.

Technologies used

HTML,CSS: For frontend
JSON python module: Used for responding to API calls with JSON objects.
flask python module:

- Flask submodule for initializing a web application
- Render_template for rendering a HTML template
- Redirect for redirecting to a particular endpoint
- Make_response for making a response request, error request

flask_restful module: for handling API requests.
flask_sqlalchemy python module: For interacting with the databases
Werkzeug python module: exception submodule for creating custom exceptions for APIs

DB Schema Design

User Table:
id [Integer, Primary Key, Auto Increment], email[String, Unique, Not null], username [String, Unique, Not null], password[String]
Deck Table:
Id [Integer, Primary Key, Auto Increment], name[String, Not Null], description[String]
Card Table:
Id [Integer, Primary Key, AUTOINCREMENT], front [String, Not Null], back[String, Not Null], DeckId [Integer]
Usercard Table:
User_id [Integer, Primary Key, References User.id], card_id [Integer, Primary Key, References Card.id], score [Integer, Not Null], difficulty [Integer], last_reviewed [Datetime, Not Null]
Userdeck Table:
user_id[Integer, Primary Key, References User.id], deck_id [Integer, Primary Key, References Deck.id], score [Decimal, Not Null], last_reviewed [Datetime]

Individual User, Card, Deck schemas for the individual objects. The relationship tables manage the user-card, user-deck relations such that the card or deck created is local to the user who created it.

API Design

POST /{username}/ - API for user session login.
GET, PUT, DELETE - /user/{user_id}/{deck_id}/ - APIs for getting details of a deck given the id, updating a deck's details and deleting a particular deck of an user.
POST - /user/{user_id}/ - API for creating a new deck for a user.
GET, PUT, DELETE - /user/{user_id}/{deck_id}/{card_id}/ - APIs for getting details of a card given the id of the user and deck to which it belongs, updating and deleting as well.
POST - /user/{user_id}/{deck_id}/card/ - API for creating a card for a user in a deck.

Architecture and Features

The complete project along with the API handlers and the controllers or the view functions are in the main.py file of the project. The templates folder consists of all the HTML files for the frontend. The static folder consists of the only JPG file for the background of the login page. The database.sqlite3 file is for the database implementation. README file consists of details about how to run the application.

User Registration: Form collects username, email, password (can be optional). Inserts values into the user table.
User login: username or email is entered along with password and the validation is done.
Deck creation, deletion and updating: A form is shown where the user can enter the details of the deck for creating and updating. Deletion is done by removing the entry from Userdeck and deck tables for the user and all the cards belonging to the deck are removed from usercard, card tables.
Card creation, deletion and updating: A form for creation and updating is shown where the user can enter the front and back(question and answer) of the card. Deletion is done by removing the entry related to the user in usercard and the card tables.

Updating the deck and card details are the additional features implemented.

Video

https://drive.google.com/file/d/1o4xN4TM9OMId87ohoi49IabSZvnX98Wb/view?usp=sharing