APP DEVELOPMENT PROJECT- FINAL REPORT

Author

Name: Ruchira Saha
Roll no: 21f1006521
Student email: 21f1006521@student.onlinedegree.iitm.ac.in

Besides being a student of IIT Madras, I"m a second year student pursuing Bachelor of Technology(BTech) in Computer Science and Engineering from Kalinga Institute of Industrial Technology(KIIT). I love learning and exploring new domains in the field of Computer Science and Machine Learning.

Description

The user needs to create an account/sign in if an account is already present. After logging in, the user is redirected to the dashboard. There is an option to add a deck in the dashboard. For existing decks, the user can add/review cards on another page(requested on clicking the add/review cards button). Database was to be created for storage & retrieval of data(deck name, front and back side of cards added to the decks by the user. Score and the time of last review is also available on the dashboard for every deck.

Technologies used

- Python - flask used is a Python based web framework. Certain python modules like os, re and datetime have been used in this web application.
- flask SQLAlchemy - flask extension that allows SQLAlchemy to be used in the application.
- flask -python based web framework to get the url for a certain .html page, to redirect to that page, to abort a request with an HTTP error code. Besides, current_app has been imported as it can be used to access data about the running application, including the configuration.
- flask.helpers - flash has been imported from flask.helpers to display error messages(flashed messages) on the screen when username/password doesn't satisfy a given set of criteria.
- flask_restful - to create a restful web api using python and flask,resource routing(sending get requests) and to parse arguments(username, password on the login and the register page).
- flask.templating - to respond to the user with an html page. Clicking login redirects the user to the 'login' page and the sign up button takes the user to the 'register' page.
- Werkzeug- generate_password_hash has been imported from werkzeug.security for authorisation, i.e., to check the entered password and security purposes.
- flask_login - to store the active user's ID in the session, and let you log them in and out easily.
- re - module in python which allows regular expressions to be checked for equality. Even special characters or characters written in other languages can be checked for equality.
- os- module in python to interact with the operating system.
- Werkzeug utils - to redirect clients to target locations.
- SQLAlchemy.sql- python SQL toolkit that allows the user's data, information about the decks and cards in databases.

- datetime - module in Python to know when the cards were last reviewed. Returns date and time of the user’s system.

DB Schema Design

User

```
id =(Integer, primary_key = True)
username = (String(30), unique=True, nullable = False)
password =(String(300), nullable=False)
date_created =(DateTime(timezone=True), default=func.now())
score =(Integer, default = 0)
user_deck = db.relationship('Deck', cascade='all, delete-orphan', backref='deck')
```

For users, id is the primary key. It cannot be null and is used to uniquely identify a user. username should be unique and cannot be null. Also user_deck establishes a one to many relationship of an user with decks.

Deck

```
id = db.Column(db.Integer, primary_key=True)
deck_name = db.Column(db.String(30))
user = db.Column(db.String, db.ForeignKey('user.username'), nullable=False)
date_created = db.Column(db.DateTime(timezone=True), default=func.now())
score = db.Column(db.Integer, default=0)
last_rev = db.Column(db.DateTime(timezone=True), default=func.now())
dcard = db.relationship('Card', cascade='all, delete-orphan', backref='card')
```

For Deck, id is the primary key. It cannot be null and it distinguishes between the users. Deck_name should be unique and cannot be null. ‘user’ is a foreign key from the user table. Also ‘score’ is ‘ integer’ type and at the time of creation of the deck, the default score should be zero. The ‘dcard’ establishes a one to many relationship of a deck with the cards.

Card

```
card_id = db.Column(db.Integer, primary_key=True)
front = db.Column(db.String(512), nullable = False)
back = db.Column(db.String(512), nullable = False)
score = db.Column(db.Integer, default = 0)
deck = db.Column(db.String, db.ForeignKey('deck.deck_name'), nullable = False)
```

For ‘Card’, card_id is the primary key. It cannot be null and it uniquely identifies a user. Front and back cannot be null. Also ‘score’ is an ‘integer' and when a deck is created the default score is 0. The foreign key from the ‘Deck’ table is deck.

API Design

The elements that I have created an API for are ‘Card API’ , ‘User API’ and ‘Deck API’.

The inputs like password, username, score will be received from the user.
Then, it will check the username and the password.
If the user is new, it will ask the user to enter the details and commit the same to the database.
The deck class will filter the deck according to the user's name and return the list of the
decks
created by the user else return that the deck already exists. If the method is 'get' , it displays the deck created by the user, its score and the last reviewed time.
For card class, the post method will add a new card to the database and then the
'get method will show the cards to the user.

Architecture and Features

All the html files are in the 'template' folder. Css,js files and images and in the static folder. Models.py has the database structure. Api.py has the api. Views.py contains the view part of the MVC. It has the route to every other html file. The controllers have been mentioned in the 'views.py' which are used for redirecting.

Jinja2 templates, flask and werkzeug have been used. There is user login/register, dashboard,review and deck management. A proper login system has been used and there is password validation. Scoring has been done according to the user's rating for a particular card. The final score is displayed as sum(score on a card/total number of cards). Score on a card is decided as follows:
Very Easy:10
Easy:8
Normal:6
Hard:4
Couldn't even guess:2
APIs have been created for login, interaction with decks and cards. There is styling and aesthetics.

Video

https://drive.google.com/file/d/1HaGlAVuncmroNNUaWoiF1URlIgbDO8SQ/view?usp=sharing