FlashyPep
Project Report for Final Project
Mordern Application Development - I

Author

SURUPI NANDI
21f1006207
21f1006207@student.onlinedegree.iitm.ac.in
I am a college student. I enjoy learning about programming and data science. This project was a lot of fun to make. There were points where I got stuck but felt great joy after I overcame those challenges.

Description

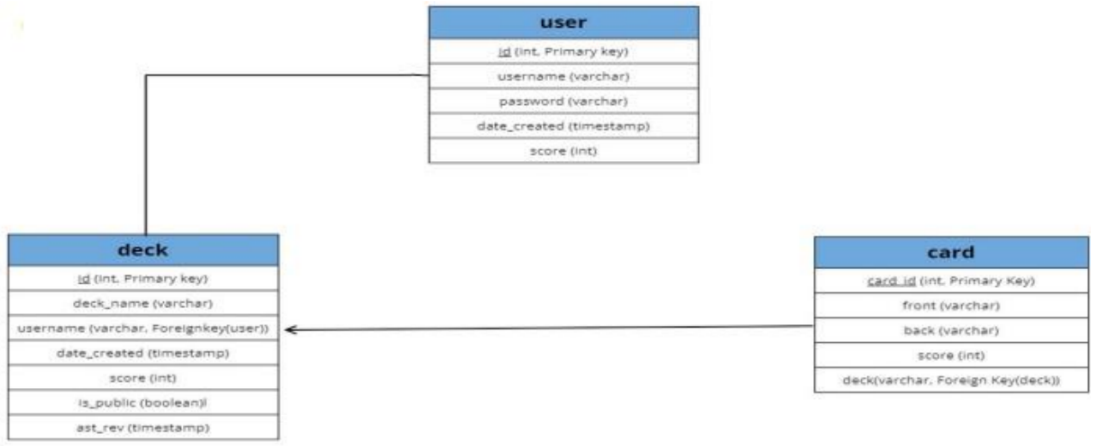
In this project, we were supposed to use HTML, CSS, bootstrap, flask, restful APIs, SQLAlchemy, and other necessary modules to build an app for flashcards. We had to build a login/signup page where we can store the usernames and passwords of people who have visited this flashcard app. Here, users can create multiple decks, and every deck can have multiple cards. Decks and cards can be created, deleted, and updated. User can track their progress of how well they remember the answers by looking at the score presented in the dashboard. The score gets based on the five levels of self-evaluation: super easy, easy, medium, hard, and rocketScience. Further, every time the user access the cards, time gets logged.

Technologies used

1. Flask: used for building the web application.
2. Flask-SQLAlchemy: extension of Flask, used to handle database connections across the app.
3. Flask-RESTful: an extension of Flask, used as the back-end which builds and handles APIs for the endpoints.
4. Flask-Login: an extension of Flask, used for the user session management. Logging in and out, remembering the user, storing active user data etc are handled with this extension.
5. Datetime: This python module is used for time stamping the class operations.
6. security: for username and password
7. Requests: This python module is used to handle HTTP requests and responses.

DB Schema Design

Relation: One user can have many decks, one deck can be used by many users, hence user and deck are in a many-to-many relation. One deck can have many cards, but one card cannot belong to many decks, hence the card to the deck is a many to one relation. User has 5 attributes with id being the primary key. Deck has 7 attributes with id being the primary key and username being the foreign key. Cards has 5 attributes with card_id being primary key and deck being foreign key.



API Design

- User_api: Implemented using get and post methods. Get method is used to fetch the user and related deck and the score from the database. The post method is used to enter new users in the database.
- Deck_api: Implemented using get and post methods. Get method is used to filter decks according to the username and a list is created to return the desk details such as deck name, score, and last reviewed card.
- Card_api: Implemented using get, post, and put methods. Get method is used to fetch users and related decks from the database. Cards are fetched from the deck for the related user, and card_id, deck, front and back information is retrieved.

Architecture and Features

The project code is organized based on its utility in different files. I named my project FlashyPep. Inside the FlashyPep folder: there is the python pages named __init__ , api, models, and views. There is also the sqlite page. Further, there is a static folder that contains all the images that were used for background, there is the templates folder that contains all the HTML files. The app.py folder is situated outside the FlashyPep folder. The HTML files include the landing page, signup page, login page, dashboard page, and review page. In the review page, hover over the flashcard to see the backside. The HTML pages are rendered through User, Deck, and Card APIs. The controllers that were used:

- @views.route('/dashboard', methods = ['GET', 'POST'])
- @views.route('/', methods=['GET'])
- @views.route('/review/<string:deck>', methods = ['GET', 'POST'])
- @views.route('/review/<string:deck>/<int:card_id>', methods = ['GET', 'POST'])
- @views.route('/login', methods = ['GET', 'POST'])
- @views.route('/register', methods = ['GET', 'POST'])
- @views.route('/<string:user>/deck/<string:deck>/delete', methods=['GET','POST'])
- @views.route('/logout')

Video

Link: https://drive.google.com/file/d/1pU_PeGYsIZfP2pwRx77x8XmZmd65LDhO/view?usp=sharing