Author

Name: Karthik Babu Nambiar
Roll Number: 21f1004345
Email: 21f1004345@student.onlinedegree.iitm.ac.in
About Myself: I am currently a 5th-year student at the Indian Institute of Science Education and Research (IISER), Bhopal, Madhya Pradesh. I was born and raised in the Kannur district of Kerala. I am currently doing my MS Thesis on Visual Inertial Navigation Systems in the department of Data Science and Engineering at IISER Bhopal

Description

The project is to develop a web application that implements the idea of flashcards. The project focuses on implementing flashcards for various languages and structuring them neatly in decks and cards. A user must be able to access these cards and also manipulate them. Create/Read/Update/Delete of users, cards and deck should be implemented. REST API of the application should also be implemented.

Technologies used

- **Flask**: flask provides a web framework with tools libraries and technologies that allow us to build various web applications
- **Flask_SQLalchemy**: It is an extension for Flask to allow developers to use SQL inside a python program. Used to connect the database across the application.
- **Flask_restful**: Extension for Flask that is used to build REST APIs. Brings easy integration with the existing Flask application.
- **CSS Styling**: To develop the aesthetics of the web pages.
- **HTML**: Templates: To describe the structure of the web pages.
- **Javascript**: To build charts and client-side validation and constraints

DB Schema Design

There are four tables in the DataBase:

- **users**
  - **user_id**: (Integer) (Primary key) Stores the unique id of each user
  - name: (Text) (Not Null) Stores the name of the user. This attribute is also unique as it is used to determine a user for login purposes.
  - password: (Text) (Not Null) Stores the password of each user. This attribute has a constrain of a minimum of seven characters maintained by the server-side application. (application/controller.py)
- **category**
  - **category_id**: (Integer) (Primary key) Stores the unique id of each category/deck.
  - name: (Text) (Not Null) Stores the title/name of the category/deck
  - description: (Text) (Not Null) Stores the description of the category/deck.
- **cards**
  - **card_id**: (Integer) (Primary key) Stores the unique id of each card
  - front: (Text) (Not Null) Stores the front/question of the card.
  - answer: (Integer) (Not Null) Stores the correct option number among the options in the card. This attribute has a constraint to be any one of the values 1, 2, 3, 4 as there as for options for each card.
  - ***category_id***: (Integer) (Foreign key - category table) Each card belongs to a unique category that is described by this foreign key.
  - option_1: (Text) (Not Null) Stores the 1st option of the card.
  - option_2: (Text) (Not Null) Stores the 2nd option of the card.
  - option_3: (Text) (Not Null) Stores the 3rd option of the card.
  - option_4: (Text) (Not Null) Stores the 4th option of the card.
- **scores**
  - **score_id**: (Integer) (Primary key) Stores the id for each score made by a user.
  - ***user_id***: (Integer) (Foreign key - users table) Each score is uniquely made by a user which is described by this foreign key.
  - score: (Integer) (Not Null) Stores the total score obtained by the user at this attempt.

- datetime: (DateTime) (Not Null) Stores the date and time of this attempt.
- ***category_id***: (Integer) (Foreign key - category table) The category of this attempt can be uniquely determined by this foreign key.
- q1: (Integer) (Not Null) Encodes whether the 1st question/card was answered correct, wrong, not answer. Stores value 1 for a correct answer, -1 for a wrong answer, and 0 for not answered. This constraint is taken care of by the server-side application.
- q2: (Integer) (Not Null) Same as q1 for 2nd question/card.
- q3: (Integer) (Not Null) Same as q1 for 3rd question/card.
- q4: (Integer) (Not Null) Same as q1 for 4th question/card.
- q5: (Integer) (Not Null) Same as q1 for 5th question/card.
- q6: (Integer) (Not Null) Same as q1 for 6th question/card.
- q7: (Integer) (Not Null) Same as q1 for 7th question/card.
- q8: (Integer) (Not Null) Same as q1 for 8th question/card.
- q9: (Integer) (Not Null) Same as q1 for 9th question/card.
- q10: (Integer) (Not Null) Same as q1 for 10th question/card.

API Design

API was created with the help of Flask_restful. There are three resources for which API has been created:

- **User**:
  - GET: “/api/user/{username}”: to get the user detail using the username.
  - PUT: “/api/user/{username}”: to update the password of an already existing user using username.
  - DELETE: “/api/user”: to delete the user using the user the username and password of a user.
  - POST: “/api/user”: to create a new user by inputting a non-existing username and a password
- **Card**:
  - GET: ”/api/card/{card_id}”: to get the card details using the card_id.
  - PUT: “/api/card/{card_id}”: to update the card details using the card_id.
  - DELETE: “/api/card”: to delete a card using the card_id.
  - POST: “/api/card”: to create a new card by inputting the card details.
- **Deck**:
  - GET: ”/api/deck/{category_id}”: to get the deck details using the category_id.
  - PUT: “//api/deck/{category_id}”: to update the deck details using the category_id.
  - DELETE: “/api/deck”: to delete a deck using the category_id.
  - POST: “//api/deck”: to create a new deck by inputting the deck details.

Architecture:

The main directory contains main.py, api documentation and local setup and run, requirements.txt, and readme files. The project is organized into 4 sub-directories:

- application: Contains the controller, models, database, api, and the validation for api.
- db_directory: Database for the application
- static: Contains CSS, JavaScript, Bootstrap, and image files essential for the application.
- templates: Contains the HTML files for the web application.

Features:

There is a login page for existing users and a registration page for new users. Upon login, a dashboard appears with the profile of the user. It also contains a graph plot of the last 7 scores of the user with the date-time and deck. It shows all the decks all a link to attempt the deck with the average score and last attempt date-time of the deck by the user. A leaderboard is shown on the dashboard page which ranks all users in ascending order of score and attempt date-time for each deck By clicking the attempt link, the user is shown cards one by one with their answers upon submission of each card. Upon completion of the attempt on any deck, the user is shown the score they obtained in that attempt and with a chart to show the number of correct, wrong and skipped questions. Also, all cards attempted are shown with the right answer. The user can see all cards in the deck, add a card to an existing deck, delete a card from a deck and also update the content of a card. The user can also add/delete/update a deck. The option to sign out or delete the user is also available on the dashboard page.

Video: https://drive.google.com/file/d/16s012tzEZwalQh2klj8BhMw7YWqcm6If/view?usp=sharing