MODERN APPLICATION DEVELOPMENT PROJECT REPORT

Author

Allan Nestor Pais
1001663
21f1001663@student.onlinedegree.iitm.ac.in
Hello my name is Allan. I thoroughly enjoyed the capstone project and it was a great learning experience for me!

Description

My project is a flashcard application in flask using python and is designed and developed end- to end with flask at the front-end and sqlite database at the backend connected by SqlAlchemy. It implements all CRUD operations and is based on the MVC (Model-View-Controller) architecture. It also implements REST-APIs for additional functionality.

Technologies used

The technologies that I have used are the Flask python framework for the front end. SQLAlchemy for linking the database to the application and Sqlite database for the backend data storage. I have also used a REST API for information retrieval which allows the admin to retrieve data from the database about the application by using a secret key.

DB Schema Design

I have 3 tables in the database in Sqlite. They are the Flashcard store, Data Store and Login information respectively. The Email id along with the deck names are used for identifying the unique tuples of information.

Schemas

```
CREATE TABLE "Data_store" (
	"Email_id"	TEXT,
	"Deck"	TEXT NOT NULL,
	"Score"	INTEGER NOT NULL,
	"Last_visited"	REAL NOT NULL,
	"Num_cards"	INTEGER NOT NULL,
	FOREIGN KEY("Email_id") REFERENCES "Login_information"("Email_id"),
	PRIMARY KEY("Deck","Email_id")
);

CREATE TABLE "Flashcard_store" (
	"Email_id"	TEXT,
	"Deck_name"	TEXT NOT NULL,
	"Question"	TEXT NOT NULL,
	"Answer"	TEXT NOT NULL,
	FOREIGN KEY("Email_id") REFERENCES "Login_information"("Email_id"),
	PRIMARY KEY("Email_id","Deck_name","Question")
);
```

```
CREATE TABLE "Login_information" (
    "Email_id"        TEXT NOT NULL,
    "Password"        TEXT NOT NULL,
    "f_name" TEXT NOT NULL,
    "last_name"       TEXT NOT NULL,
    PRIMARY KEY("Email_id")
);
```

API Design

I have created a REST API called the admin API. This is a get function API that is invoked when the Admin login button is clicked on the user screen. The main function of the API is to authenticate and provide the admin with necessary information about the users of the database and their usage habits. 3 sets of information namely usernames, deck names and total number of cards can be retrieved by using this functionality.

Architecture and Features

The main stand out feature of this application is that it allows multiple users to login and store their data without mixing it with other users. The application implements all CRUD Functionalities and allows users to add cards, delete cards, add decks and delete decks. The practice functionality can be used by the users to practice each deck as many times that they want.The score then gets updated once the review is done. The application only increments the score when the user marks the card as easy since, a response marked as medium implies recognition and one marked as hard implies new knowledge.The count, last seen and score is likewise updated after the practice exam is taken.

The biggest functionality of this application is the selective intermixing of user data when required. The examination mode combines cards of a similar deck from all users so that each user could practice another user’s cards without a privacy breach. This application could be used for a local community of students with a common database so that every person’s knowledge can help another person. The Admin login allows an admin sitting remotely to access the database and see which users use it along with the decks associated with the database. Users and decks have been decoupled in order to ensure privacy. The admin has to enter a predefined secret key into the url after entering Admin mode and the required data will be then displayed on the screen.

Thus a highly secure ,flexible and multipurpose application has been developed. The code has been uploaded in a single file called app.py. The start of the code consists of all the imports to be made. Then the database is initialized and paired with SQLAlchemy. Now the controllers are written to control the entire view generation process. The html pages are developed using html,Bootstrap and CSS and reside in the templates folder. The external CSS files reside in the static folder. Set-up shell scripts along with the Requirements.txt file are also provided

Video - https://drive.google.com/drive/folders/1TpbQL3axoDQwBORlCGBSd1WwHWv6HB-L?usp=sharing