FLASH(K)ARD

FINAL PROJECT FOR APP DEV-1

ABHISHEK KUMAR

28.11.2021

EMAIL:21f1002429@student.onlinedegree.iitm.ac.in

Screencast Link

INTRODUCTION

As a part of Final Project, I have made Flashcard Web Application which is used for memory training. I have named the Application as Flash(k)ard as it is made with Flask Framework.

PACKAGE DEPENDENCY

Following Packages have been used for making the Application:

1. **Flask-** Flask is a web application framework written in Python.It lets you develop web application easily
2. **Flask-SQLAlchemy** - Flask-SQLAlchemy is an extension for Flask that aims to simplify using SQLAlchemy with Flask by providing defaults and helpers to accomplish common tasks. One of the most sought after helpers being the handling of a database connection across the app.
3. **flask-restful** - Extension of flask to create API Resource
4. **flask-bcrypt** - Extension of flask to create hashing and verifying the login-password
5. **Bootstrap**: It is used for styling

MODELS

The following models are defined with their attributes:

1. User Model with id(primary key), username, password and active(Boolean). Some functions are also defined in the User model for the authentication purpose to enable current_user functionality.
2. Deck Model with id(primary key), title(String), deck_score(Integer), user_id(Foreign Key to user.id), review_time(Datetime)
3. Card Model with id(primary key), front(String), back(String), deck_id(Foreign Key to deck.id)

SYSTEM VIEW

A base.html is defined with a basic view on which other views are extended. These view are extended using the jinja template extend function. Bootstrap Navbar is used in the top with functionality of clicking the **Flash(k)ard** on the left upper corner redirects the view to the index page only if the user is authenticated.

The application is divided into following views:

1. Login/Register View
2. Dashboard/Deck Management View
3. Add Deck View
4. Add Card View
5. Edit Deck/Card View
6. Test View

LOGIN/REGISTER

The Login/Register page has been made with the help of Bootstrap container and form. On the controller side, FlaskForm is used and defined in forms.py for LoginForm() and RegistrationForm(). It has been made to validate input, like password should be of minimum length 4 and fields should not be empty. **wtforms.validators** are used to validate input. These validations are directly applied in **login.html** and **register.html** and in case of error bootstrap class `alert alert-warning` has been used to show the error. **flash** module from the Flask has been used in the controller.py in the function login and register to show when user inputs username/password that doesn't match with the one stored in the database or when user registers successfully. The flash message is then integrated to the login page using the jinja template.

DASHBOARD/DECK MANAGEMENT VIEW

The deck management view is designed on index.html. In this view, all decks with No. of cards, deck_score, last review time(UTC time is changed to IST using pytz) have been populated. If the user comes back to its account after more than 1 hour(which can be changed in the code), a review button is enabled under the score column to inform the user that it is now the time to review the deck.

Four actions have been defined for each deck:

a) Add card- To add the card to the particular deck
b) Delete- To delete the particular deck
c) Test- To test/learn the card
d) Reset- If you try to test an already reviewed card, then the test dashboard will show- All the cards have been reviewed, Return to Dashboard. So if you want to review again, you can click Reset and review the cards again
e) Edit Deck/Cards: If you just click Deck name, you will be redirected to deck title updation and card updation in this particular deck

In the Last row, Add New Deck Button has been made to create a new deck.

ADD DECK VIEW

By clicking the Add deck at the bottom of the dashboard, you will be redirected to this view. In the view, deck Title can be entered to create a new deck.

ADD CARD VIEW

For each deck, cards can be added by clicking on Add Card. You will be presented with an add card view where you can enter the Front Side and Back Side of the Card. Textarea has been used as an input method.

EDIT/DECK CARD VIEW

By clicking on the title of Deck, you will be redirected to the Edit/Deck Card view designed on showcard.html. All the cards in the particular deck will be populated in this view. Bootstrap Card with fixed width and min-height is used and is arranged with the help of grid using container and class col-3 so that 4 cards can be arranged in a grid

TEST VIEW

The cards in the deck can be learned by clicking on the Test link which redirects to the Test page designed on **test.html**. Bootstrap card is used and placed in the middle of the screen where the front of the card is shown. Above the card Current deck name and Current Score in percent Terms is shown. The back content of the card and the buttons (I know & Don't Know) is styled with visibility hidden. Javascript onclick event listener is used on the card which will then execute the function defined in customvalidation.js. As soon as the user clicks the card, the front content is set to hidden and the back content with buttons is set to visible. Clicking on the button will change the score on scoring pattern. As soon as all cards are reviewed, a message will be displayed that all cards have been reviewed and a button will be enabled to return to Dashboard.

SCORING PATTERN

Individual deck score is decided on the basis of the number of cards and number of cards for which you clicked the **I know** button during the test session. For example if there are 10 cards in the deck and you clicked I know 6 times, then your score will be 60% which will be displayed and updated simultaneously and after the review it will be updated in the dashboard.

RESTFUL API

CRUD API is created as UserAPI, DeckAPI and CardAPI to read, update, delete or post the respective resource. Yaml file for API definition is included in the Folder.

WORKFLOW

1. If you are a new user, click the Register Link. Then the user can login with username and password.
2. Add the deck by clicking the Add deck button and the topic/title of the Deck
3. The Deck will be populated on the dashboard.

4. Add card in the deck by clicking add card.
5. Test the deck by clicking the test link.
6. If the deck is not reviewed for more than one hour, the **Review** button will be enabled on the score column. If you want to test the deck before one hour, you can click the reset button and then **Test** link.
7. By clicking the deck title on the dashboard, you can either delete or edit the card. You can also edit the deck title in this view
8. You can also delete the deck by clicking the 'Delete' link in the dashboard.

REFERENCES:

1. **Bootstrap5 Tutorial**
2. **Stackoverflow for unlimited errors**
3. **App Dev Live Session-All Solve with Instructor & ScreenCasts**
4. **Pretty Printed tutorial for Login and Authentication**
5. **GeeksforGeeks for conversion of UTC Datetime to IST**
6. **Python Documentation Page**
7. **Flask, flask-restful, Flask-login, Flask-SQLAlchemy, flask-bcrypt documentation page**
8. **Week-6 Lab Assignment for creation of API**