Author

Name : Yogesh Bhutkar

Roll No : 21f1004862

Email : 21f1004862@student.onlinedegree.iitm.ac.in

I am currently studying B.Tech. in Hyderabad, India as well as my B.S. from IITM

Description

This project is aimed at logging and registering a user into an app where the user is capable of creating multiple lists and each list can have multiple list Items(cards) contained inside the list. These List Items are basically To-Do's. Summary page shows trendline by analysing the data. CRUD API is used to manage data flow.

Technologies used

To build this project, I used HTML, CSS, Bootstrap, basic JavaScript and majorly Python-flask, jinja2 templates, flask_sql alchemy.

HTML, CSS is being used to provide my project with a proper structure (HTML) as well as styling (CSS). HTML is implemented in the form of Jinja2 templates since code repetition is greatly eliminated by using the inheritance feature of jinja2 thus making programming task lot easier than sticking with the traditional HTML approach for everything.

Flask-SqlAlchemy is used to store the data generated by the users while using the flask app. This data is stored in an SQLite database which is fairly easy to set-up and gives an edge over other methods of setting up a database for the storage of data generated by the user. It is also used to provide various users with a sign-in as well as a register option to personalize their experience of using the flask app.

Apart from these technologies, I have also used ChartJS (A javascript library) for making html-based charts. Basically using chartjs we can directly embed the data into a canvas element where the data would be depicted in the form of a trendline specified by the user. In my case, it happens to be a line chart.

Bootstrap is used to provide basic styling to the web-application to make the application aesthetically pleasing in the frontend aspect.

DB Schema Design

My DB consists of three tables, namely Users, KanBanList, ListItems.

Attributes present in Users are : sno (primary key), username(nullable = false and unique = True), password(nullable = false and unique = True).

Here, sno stores a unique number to uniquely identify each tuple inside of the databse, username and password stores the login information of the user.

The table users stores the information about letting the user log-in as well as log-out of the app. That is, a user will be asked for username and password in order to log-in to the app. The user can register with a unique username as well as a password.

Attributes present in KanBanList are: sno (primary key), listName (nullable = False), username (nullable = False)

Here, sno stores the unique number associated to all the KanBan Lists present in the app, listName stores the name of the list which stores string values and username is associated with a particular user as it would help to display user specific elements inside the web-app.

Attributes present in ListItems are : sno(primary key), title, reference, deadline, content, completedFlag, timeCreated, timeCompleted, username

Sno being a primary key identifies each row uniquely, title stores the title of the card, reference stores which list a particular card is associated to (this helps while displaying the elements inside the web-app), deadline stores the deadline before which the task is supposed to be completed, content describes the title of the web-app in greater detail, completedFlag stores whether a particular task is finished performing by marking it as complete inside the completedFlag, time created, time completed are both system generated times and convey the obvious meaning whereas username is used to refer a card to a particular user. (would help In displaying the cards associated with a user in the web-app).

Reason to design my database in this way is to provide simplicity to my database design. Only primary keys and nullable fields are used to maintain the integrity of the database and no foreign key is used to reduce the complexity of the database so that I can focus on the core functionality more than that of the data base. All the information required is able to be captured by these attributes with cent percent success rate.

API DESIGN

The usual CRUD operations were implemented with the help of Sql-alchemy package in python which uses Sqlite as a backend database. This data is pushed and retrieved as required.

Apart from this, another API called as ChartJS is being used to make html-based charts. This is basically a JavaScript based framework which helps in making trendlines corresponding to the web-app.

The graph is plotted inside a canvas element inside the html which is referenced by a script tag which provides labels and values using which the graph is supposed to be drawn / plotted.

Architecture and Features

The project folder contains env, static, templates folder along with app.py, Kanban.db, username.txt, count.txt

env stores the packages installed, static stores css files and image file. Template folder contains various html jinja2 templates to implement various tasks like implementing login, register, index, and statistics pages. App.py contains the python code / business logic to implement all the user functionalities of the web-app. Kanban.db contains the SQLite database structure so that the user generated data can be safely stored and accessed as required.Username.txt stores the username of the currently logged in user and count.txt stores the next valid unique id of the list item which increments by a factor of 1 every time a list is appended to the database.

Features: This app lets the user to sign in and have a personalized experience where the user can only see lists that he made and not the lists made by others which provides a great level of data abstraction. The app also enables a particular user to create multiple lists as much as he wants and can name however he wants and can update when felt necessary, the lists can contain any number of cards describing the tasks supposed to done (which would be described in the content area), it would contain all the information mentioned in the DB schema of the ListItems table. The user can add details regarding to the card while creating it and can update by clicking it from the list view. The items can be transferred from one list to other with ease which is implemented with the help of a simple drop down button. The user can also see various statistics related to the completion of the project like, a trendline which shows tasks completed with respect to the date of completion, the details of tasks completed, the details of tasks which have passed the deadline and still are not completed.

Video

https://drive.google.com/file/d/1snoMZ1k3H5cWjho9yYyOz9YWTag0a7mN/view?usp=sharing