Kanban Application V1 Documentation

Author

KRUTHIVENTI M R S SAI CHARAN
21f1004450
21f1004450@student.onlinedegree.iitm.ac.in
I am studying my B.Tech currently and have programming experience over various technical domains. I am very much interested in this IITM BS degree and looking forward to learning more and applying more.

Description

Over emerging technologies, the need for fast and reliable tracking of tasks is a great challenge. KanbanApp is a stable one stop solution for all the user's needs and cravings for using a fast, responsive application for tasking and tracking.

Technologies used

flask - for app routing, templates rendering and redirecting, and flash messages display.
flask-SQLAlchemy - for connecting the SQLite database as a server with the flask application.
flask_restful - to implement CRUD operations using flask API.
json - for API communication and dumping & loading data from server to front end javascript

DB Schema Design

Users Table Schema			
Column Name	**Column Type**	**Constraints**	**Reason**
user_id	Integer	Primary Key, Auto Increment	To uniquely identify the user with id
username	String	Not Null, Unique	
password	String	Not Null	For Security
email	String	Not Null	To communication purposes

Lists Table Schema			
Column Name	**Column Type**	**Constraints**	**Reason**
list_id	Integer	Primary Key, Auto Increment	To uniquely identify the list with id
listname	String	Not Null	
isactive	String	Not Null	Status of deletion
uid	Integer	Foreign Key (user.id), Not Null	To identify the lists of particular user

Cards Table Schema			
Column Name	**Column Type**	**Constraints**	**Reason**
card_id	Integer	Primary key, Auto Increment	To uniquely identify the card with id
card_title	String	Not Null	
card_content	String		
deadline_dt	DateTime	Not Null	Deadline for every task
isactive	String	Not Null	Activity status of Card
created_dt	String	Not Null	To record the card create date
completed_dt	String	Not Null	Completed date time tracking
list_id	Integer	Foreign Key (list.id), Not Null	To identify the cards of a particular list.

API Design

API was designed for interaction with lists and cards using CRUD operations.
List has GET (fetches all lists for a particular user), POST (creates a new list), DELETE (deletes the existing list), PUT (updating the existing list)
Card has GET (fetches all cards for a particular list), POST (creates a new card), DELETE (deletes the existing card), PUT (updating the existing card).
A YAML file for the API documentation is attached in the root directory.

Architecture and Features

- Modals for Generic Implementation
- Extensive usage of Bootstrap for technicalities
- Cached JavaScript files for faster loading
- Efficient in terms of usage
- Capable of being built over various applications with the help of API

Video

https://drive.google.com/drive/folders/1a9o6dB4-5oFOMHmLYuk9m9iFWfLOja--?usp=sharing