Author

Name: DEENA GAUTAM

Roll No: 21F1001012

student email: 21f1001012@student.onlinedegree.iitm.ac.in

I started this degree as a part-time degree which eventually got converted to a dedicated full-time course. I am focused, dedicated and an empath, which is why the end user experience is of utmost importance to me.

Description

The main motive of Kanban app is to make it a task-management and a personalized app, using Flask. By applying Create, Read, Update and Delete properties on the three major components of the database models that are – Users, List and Card. Enable the user to move cards across the lists and to be able to identify the completed tasks visually. And summarizing the task status using graphs and scatter plot, to enable the user to export the data from dashboard, sending them daily reminders about pending tasks and monthly report to enhance user experience.

Technologies used

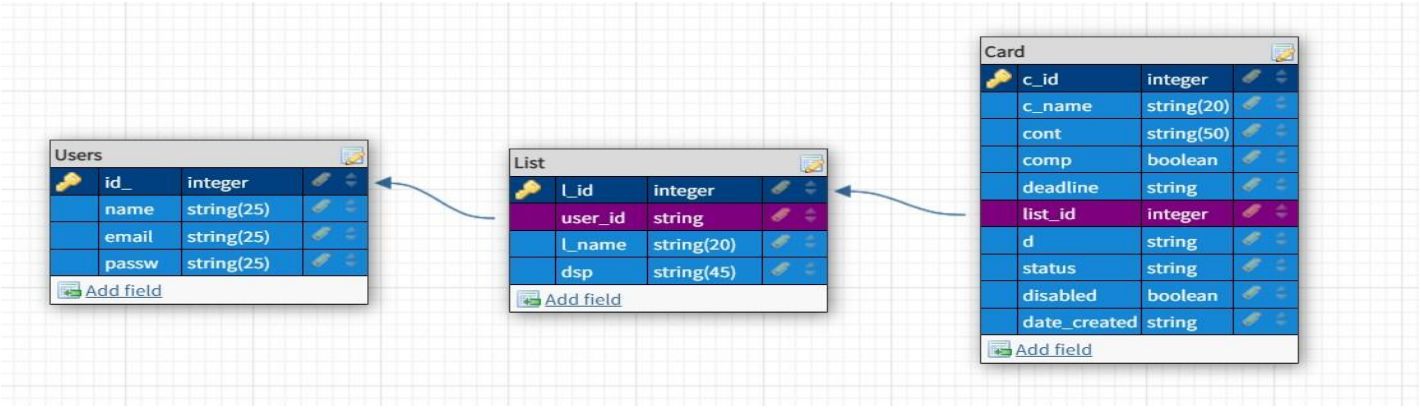
FRONTEND – Vue framework, JavaScript, Vue-router, Vue Components, HTML, CSS, Bootstrap

BACKEND – Flask, FlaskRESTful, FlaskCORS, SQLAlchemy, Flask caching, Matplotlib, Celery, Redis

DATA STORAGE – SQLite

IDE – Visual Studio Code, RedisInsight, Insomnia, Swagger(alternatively), Ubuntu(WSL)

DB Schema Design



The tables users and list share ***one to many*** relationship which places a foreign key user_id in the list table referencing the table users and to establish ***bidirectional relationship*** the users table uses relationship.backref parameters, referencing a collection of lists represented by the table list. The relationship also contains an additional parameter of ***cascade*** with

settings all and delete orphan where the all symbol is a synonym for save-update, merge, refresh-expire, expunge, delete, and using it in conjunction with delete-orphan indicates that the child object (list) should follow along with its parent (user) in all cases, and be deleted once it is no longer associated with that user.

The tables list and card also share the same relationship as mentioned above where the table list is the parent object and card is the child object.

The attributes id_, l_id, c_id are primary keys for the tables users, list and card respectively.

The attributes name, email, passw (from users table) , l_name (from list table) and c_name , deadline (from card table) are non-nullable attributes.

The attribute deadline stores the datetime as a string and the attribute d is updated whenever the user completes the task before deadline, status stores the status of card and date_created stores datetime of card creation in string format, disabled has a default beelean value false and is updated whenever the deadline expires.

API Design

API comprises of -

- **GET**, **POST** methods for **users** table to login and signup respectively.
- **GET** (to get list details), **POST** (to add a list), **PUT** (to update list details), **DELETE** (to delete list along with all the cards in the list) are the methods created for **list** table.
- **GET** (to get card details along with the status of the card), **POST** (to add card to a list), **PUT** (to update card details and move card to another list of the same user if required), **DELETE** (to delete card) are the methods created for the table **card**.
- **GET** (to fetch listwise **summary**/statistics about the tasks for the user).
- All the endpoints except for login, signup and logout are protected with Flask JWT and are cached.

Architecture and Features

The project folder comprises of the folders and files -

1. *application* – where resides python files for API (api.py), **database creation** (models.py) and configuration (config.py), workers.py to initiate celery, and task.py to run periodic jobs
2. *db_directory* – has the **SQLite database** (database.sqlite3)
3. *Report&docs* – has **Project repor**t (Project_Report.pdf)
4. *static* – folder to accommodate **images** in png format and **css** files, the **component folder** (has all the **js** files for all the components) and a **csv file** which is used for exporting.
5. *templates* – has all the **html** files, which are used to **send mails**.
6. *main.py* – this is where the **initialization** and **compilation** happens. It imports API from application folder and all the required packages then defines a function where database, API and the app are initialized and then calls the function and **runs** the app.
7. *README.md* – contains commands for **local setup** and the file structure
8. *requirements.txt* – contains all the **required packages** to be installed to be able to run the app

Video

Presentation link-
https://drive.google.com/file/d/1ARa0CCXJTgn6SYmDzg9lyKwGW_igJPj5/view?usp=share_link