Project Report: Kanban App

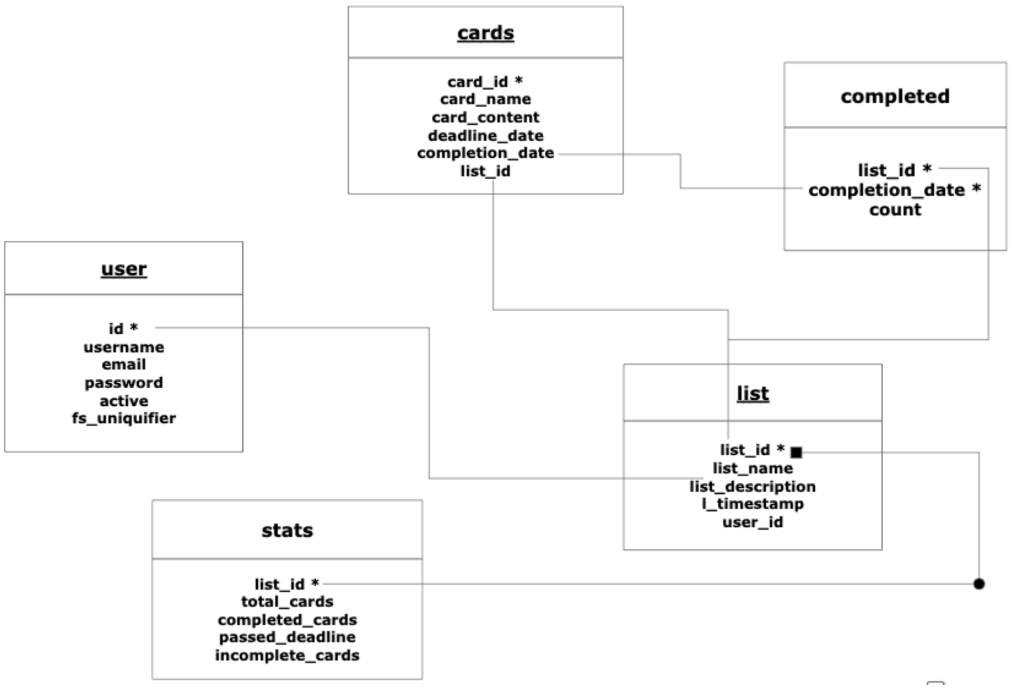
Name: Anushka Krishna
Roll: 21f1003017
Email: 21f1003017@student.onlinedegree.iitm.ac.in

I am Anushka, a final-year electronics and communication engineering student, from Bihar.

Description

Having used the Flask framework to create the Kanban app, with authentication fulfilled using flask-security and different requirements of the app accomplished using API built using flask-restful. It presents the performance graph of the user in completing the tasks before the deadline, generated using the matplotlib library of python. The front end is based on VueJS CLI with inline CSS for aesthetics. The reminders and exports are aimed using smtplib.

DB Schema Design



It is an app-based project for public use so multiple users can login to create their own lists and tasks with token-based authentication. The project requirement requires secure login for the registered users which is aimed using flask-security, also it requires that users can have a number of lists, with multiple users being able to log in so keep the email unique with the id from the user table as a foreign key for list table. Further, then the addition of cards in every list is required, list_id is set as a foreign key for the cards table. list_id gets updated further by moving cards to a different list. If the mark as complete is checked in the form, it gives the current date as the completion date. Those cards which have a deadline after the completion date are considered complete, one whose deadline is before the present date is considered a passed deadline case and if no completed date refers to an incomplete card. Further, the cards completed before the deadline are collected with their list_id and completion_date as primary keys and the count of completed cards completed in a day for a particular list of a user. Further, the graph is plotted using the completed table. Also, we get statistics of completion before the deadline, incomplete and passed deadline cards using the stats table.

Technologies used

- **os**: For giving the path of the database and folders containing images
- **flask**: **Flask**: For creating the application
- **flask_sqlalchemy**- **SQLAlchemy**: For creating models in the database for the app
- **flask_security:** for ensuring authentication of the application
- **flask_restful:** for creating APIs for different tasks
- **flask_caching:** for caching purpose to ensure performance
- **VueJS CLI:** for frontend and UI
- **CSS**: (Inline) To add styling effects
- **matplotlib**: For creating graphs
- **datetime**: To use dates as data
- **celery:** for task and jobs management
- **redis:** for caching
- **smtplib:** for sending emails of reminders and exports

Architecture and Features

The code for the Kanban app is organized based on the usage and update required. The code for the app can be viewed inside the project folder with the name app.py python file. The app named Kanban, on starting gives us a login page where we have a link to the registration page in order to register first which redirects us to the login page which requires the correct combination of email and password to log in, which further takes us to the home page where there are no lists for new users and an option to add them, and for existing users, there are existing lists and cards and also an option to add new lists and cards. For new users, adding card options gets available once lists are added. We could visually identify the completed and incomplete cards as they have labels for that. On the home page, there are links for exporting the lists, the summary page and a link to log out. Every list's elements can also be exported using the button provided for the purpose. The user has the ability to update and delete the lists and cards using the buttons available at the right corner of lists and cards which onclick opens up the update page in case the update button is clicked and in case the delete button is clicked, confirmation with the user is made for the delete action. Also, the user gets daily and monthly reminders via mail and exported files in the mail The UI for the app is based on VueJS CLI, the code for which is inside the frontend folder. The different views are based on different frontend routes, the components for which are inside the components folder of the src folder and routes are mentioned in the routes folder of the same as components. The backend is organized such that we have app.py for running the app in which we have defined different URL's for different purposes and the purpose is defined inside the api.py of the application folder. The exports and reminders are defined inside tasks.py which are taken where required. Reminders and Exporting jobs are ensured via celery code which is mentioned in clry.py and used as required in the app.py(for reminders) and tasks.py(for exporting). We also use flask-caching in api.py for caching. The database has 3 primary tables and 2 secondary tables and a table for security. Sending reminders and exports over mail is aimed using the code in emailgen.py in which we use smtplib of python to send emails and required files are attached using MIMEMultipart.The app is served at `http://127.0.0.1:5000` and the front UI of the app can be viewed at `http://localhost:8080/`.

Video

Project Video Link