Author

Afnan Ahmad
21F1003730
21F1003730@student.onlinedegree.iitm.ac.in
I am currently in the 2nd year of the IIT Madras BS Degree program. I have submitted this project as part of the Modern Application Development I course for the May 2022 term.

Description

In this project, we need to develop a web app for a Kanban board. For this, we would need essentially 3 main components:

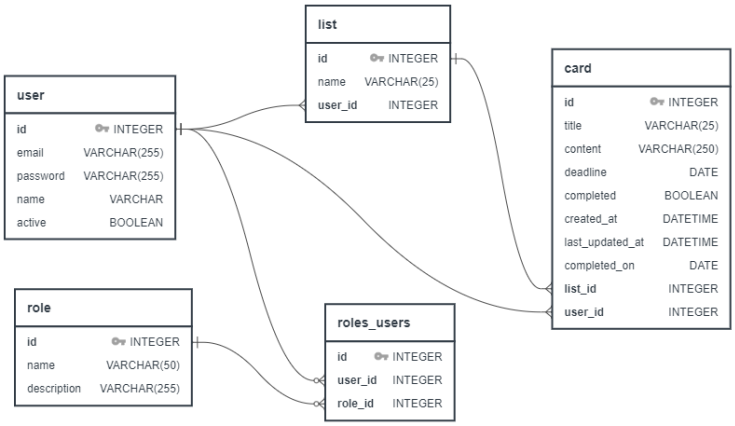
1. Views for Board, List and Cards.
2. Data Models for each of them.
3. Controllers that manage operations such adding a list or card, updating them, etc.

Technologies used

1. Flask, as the web framework.
2. Flask-SQLAlchemy, as an ORM for the database (SQLite).
3. Flask-Security, for user registration and login functionality.
4. bcrypt, email-validator, used by Flask-Security.
5. Flask-WTF, for creating forms with backend validation. Also used by Flask-Security for login/register forms.
6. Flask-RESTful, for the REST API controllers.
7. Flask-JWT-Extended, for securing API routes with token authentication.
8. humanize, for formatting dates in human-readable format.
9. Matplotlib, for creating the plots in summary page.
10. Bootstrap, for frontend styling and responsive design.
11. Inter (Google Font), as a typeface for the app and Material Icons, for icons.

Note: No JavaScript is used in this project, except for the script that comes with Bootstrap.

DB Schema Design



All fields are NOT NULL except:

1. card.content (it is optional)
2. card.completed_on (it might be null if card is not completed yet)

Note: “role” and “roles_users” tables are *not used* by the app currently. It is created as it was required by Flask-Security. It can be used potentially for implementing permissions.

card.created_at, card.last_updated_at and card.completed_on are automatically updated on creation of a card, updating a card and marking a card as complete respectively.

API Design

There are 3 endpoints (resources) for the API:

1. /api/list, for CRUD operations on lists.
2. /api/card, for CRUD operations on cards.
3. /api/summary, for getting basic stats for a list.

/api/list endpoint accepts GET, POST, PUT and DELETE requests for performing the appropriate action, that is, getting a list, creating a new list, updating a list and deleting a list respectively. Similarly, /api/card endpoint also accepts GET, POST, PUT and DELETE requests.

/api/summary endpoint accepts GET requests only, it returns the total list count for the board (if no list id is given), or total, completed and overdue tasks count for the given list id.

Architecture and Features

There are separate directories for templates, models, controllers, API controllers (api), forms and static files. Naming conventions are used in such a way that it is easy to navigate and understand the code. The main.py file is the entry point for the app. README.md contains instructions on running the app.

Most features described in the problem statement are implemented except Import/Export list functionality.

It is ensured that the user input is validated at the frontend and the backend for both web and API controllers.

As some additional features:

1. All views are mobile-friendly.
2. There is support for multiple user accounts (each having their own board).
3. CSS based styling is done to ensure that the user experience is good.

Video

https://drive.google.com/file/d/1vDEuQZZWPWNfE9D31BOw-tQXYp6egA0-/view