Kanban App V2

Problem definition

Modern Application Development - II

Last Updated: 22-09-07

Frameworks to be used

- Flask for API
- VueJS for UI
- VueJS Advanced with CLI (only if required, not necessary)
- Jinja2 templates if required
    - Not to be used for UI
- Bootstrap, if required
- SQLite for database
- Redis for caching
- Redis and Celery for batch jobs
- It should be possible to run all the demos on the student's computer, which should either be a Linux based system or should be able to simulate the same. You can use WSL for Windows OS.

Kanban

- Used for tracking tasks
- User can have multiple lists
- Each list will have
  - ID
  - Name
- User can add one or more cards to a list. Each card will have
  - Title
  - Content
  - Deadline
  - Completed flag
  - List it belongs to
- System will automatically capture the card created datetime and also card last updated datetime
- System will also automatically capture task completed datetime
- System will track progress over time and shows graphs trend lines etc. as Summary

Terminology

- Board - Kanban board
- List - List of tasks
- Card - Each card is represented as a card
- Movement - Card can be moved from one list to another list
- Summary - Shows how the user is performing across lists based on the completed flag, time when it completed, it also shows graphs

Reference Material

- Kanban on Wikipedia
- Examples - Board Examples for Your Team

Similar Products in the Market

1. Kanboard
   - Open Source
   - Web
2. WeKan
   - Open Source
   - Web
3. Trello
   - Commercial
   - Web and Android App

- These are meant for exploring the idea and inspiration
- Don’t copy, get inspired

Example Wireframe

- Click this link to check the wireframes
- It is just given to gain a basic understanding, and not meant to be followed exactly

Core Functionality

- This will be graded
- Base requirements:
  - User login
  - Main Board with Lists
  - List management
  - Card management
  - Summary page
- Backend Jobs
  - Export Jobs
  - Reporting Jobs
  - Alert Jobs
- Backend Performance

Core - User Login

- Form for username and password
- Use Flask Security and Token Based Authentication
- Suitable model for user

Core - Board

- Board view with lists
- Ability to add or manage a list
- Ability to add or manage a card
- Ability to move cards between tasks/lists
- Ability to identify the completed tasks visually

Core - List management

- Create a new list
  - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
- Edit a list
  - Change title
- View List on Board
- Remove a list
  - Move the cards to a different list or delete the cards too with a confirmation from the user
- Export option is required

Core - Card management

- Create/Add a card to a list
- Edit: Change title or content
- Edit: Mark as complete
- Edit: Move the card from one list to another
- Remove/Delete a card
- Export option is required

Core - Summary Page

- Show the summary of cards per list basis
- No of completed tasks
- No of tasks crossed deadline
- Graph or trend lines as to when tasks were completed

Core - Daily Reminder Jobs

- Scheduled Job - Daily reminders on Google Chat using webhook or SMS or Email
  - In the evening, every day (you can choose time of your choice)
  - Check if the user has any pending tasks
  - If yes, then send the alert asking them to update the status

Core - Scheduled Job - Monthly Progress Report

- Scheduled Job - Monthly Progress Report
  - Come Up with a monthly progress report in HTML (email)
  - On the first day of the month
    - Start a job
    - Create a report
    - Send it as email

Core - User Triggered Async Job - Export as CSV

- User Triggered Async Job - Export as CSV
  - Come up with an export CSV format for list and cards
  - Have a dashboard where the user can export
  - Trigger a batch job, send an alert once done

Core - Performance and Caching

- Add caching where required to increase the performance
- Add cache expiry
- API Performance

Recommended (graded)

- Backend Jobs
  - Import Jobs
- Well designed PDF reports (User can choose between HTML and PDF reports)
- Single Responsive UI for both Mobile and Desktop
  - Unified UI that works across devices
  - Add to desktop feature

Optional

- Styling and Aesthetics

Evaluation

- Report (not more than 2 pages) describing models and overall system design
  - Include as PDF inside submission folder
- All code to be submitted on portal
- A brief (2-3 minute) video explaining how you approached the problem, what you have implemented, and any extra features
  - This will be viewed during or before the viva, so should be a clear explanation of your work
- Viva: after the video explanation, you are required to give a demo of your work, and answer any questions
  - This includes making changes as requested and running the code for a live demo
  - Other questions that may be unrelated to the project itself but are relevant for the course

Instructions

- This is a live document and will be updated with more details and FAQs (possibly including suggested wireframes, but not specific implementation details) as we proceed.
- We will freeze the problem statement on or before 30th September, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.