Blog Lite V2

Problem definition

Modern Application Development - II

Last Updated: 22-12-12

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.

Blog Lite

- It is a multi-user app
- Used for uploading blogs with images
- User can post multiple times
- Each post will have
  - ID
  - Title
  - Caption/Description
  - ImageURL
  - Timestamp
- A user can follow other users using the app
- Each user will have
  - username
  - Password
  - No of followers
  - No of posts
- Every user will have its own feed
- System will automatically show the blogs from the users you follow in a particular sequence
- The recommended order of blogs in a user’s feed is based on the timestamp of blogs

Terminology

- Social Platform
- Profile - Basic stats, List of blogs
- Feed - Lists of blogs uploaded by other users you follow
- Archive (optional) - Blogs can also be made private / hidden from others

Similar Products in the Market:

1. Instagram
   - Web, IOS and Android
2. Facebook
   - Open Source
   - Web, IOS and Android
3. Twitter
   - Open Source
   - Web, IOS and Android

- 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 signup and login
    - User profile view with basic stats
    - Blog Post Management
    - Search and Follow / Unfollow Others
    - User’s Feed
- Backend Jobs
    - Export Jobs
    - Reporting Jobs
    - Alert Jobs
- Backend Performance

Core - User Signup and Login

- Form for username and password (both login and signup)
- Use Flask Security and Token Based Authentication
- Suitable model for user

Core - User’s Profile

- Basic profile view for a user
- Ability to view the number of blogs created
- Ability to view the number of followers and people you follow
- Ability to view the list of posts created

Core - Blog management

- Create a new blog
  - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
  - Content should handle the safe HTML tags
- Edit a blog
  - Change title/caption or image
- Remove a blog
  - With a confirmation from the user
- Export option is required

Core - Search and Follow / Unfollow Others

- Ability to search other users
- Ability to follow others
- Ability to unfollow others

User’s Feed

- Show the blogs/posts created by other users
- Navigate to the user’s profile on clicking the username on the blog or post

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 not visited/posted anything
  - If yes, then send the alert asking them to visit/post

Core - Scheduled Job - Monthly Engagement Report

- Scheduled Job - Monthly Engagement 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 blogs
  - 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 22nd December, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.