Flask Web App News Aggregator:

Overview: The Flask web app news aggregator is a personalized tech and breaking news feed. Users can create an account and receive a curated list of news articles relevant to their 2 preferred domains which will be selected from a list of 6 domains. The app will display 6 articles on the user's dashboard, 3 from a tech domain and 3 from another news domain, pulled in order of popularity each day from each site. The app will also include a feature for saving articles, allowing users to easily access and view articles they have saved for later.

Features:

1. User authentication: Users can create an account by providing their email, username, and password. Once they have an account, they can log in and access their personalized news feed and saved articles.
2. User profile: The user profile will store the user's preferred domains and any saved articles. This information will be used to generate the personalized news feed for each user.
3. News article feed: The news article feed will be generated by retrieving articles from the News API based on the user's preferred domains. The articles will be displayed on the user's dashboard in a clean and organized manner, with the option to read the full article or save it for later. The dashboard will show 6 articles
4. Saved articles: The saved articles section will allow users to easily access and view articles they have saved for later. This section will be accessible from the user's dashboard.
5. API integration: The API integration with the News API will allow the app to retrieve real-time news articles based on the user's preferred domains.

Technologies:

1. Python 3
2. Flask web framework
3. MySQL database
4. SQLAlchemy ORM for database management
5. HTML, CSS, and JavaScript + Frameworks for front-end design and interactivity
6. Bootstrap front-end framework for responsive design
7. News API for retrieving news articles based on user domain selection

Feedback: Can you please send me a link to the API you plan to use? If it's the news api [https://newsapi.org](https://newsapi.org/) then I can imagine there would be an issue with the saving of news articles unless you are saving the URL, but honestly you would have to save the whole object. My suggestion, because user logins would be a little beyond scope maybe due to handling tokens and keeping track of the state of several users at a time, would be to use localstorage in the browser. You can store some information there like their last three favorite or saved articles and their chosen news items.  
  
If it is that API that you are planning to use that I outlined above, for free you get 100 calls a day. So take care when working as you can easily hit 100 api calls in 20 mins if you setup the API call to happen every time you refresh the screen and that would render that work day moot.  
  
When developing as in setting up HTML and CSS I would make sure the API call is turned off and I would use dummy data to drive what I am working on. You can do this by taking the response you get from the API and creating a variable with it for example:

// commented out API call that would normally set response

response = { ...the object for the response with the key value pairs with maybe 5 to 10 articles }

You also don't need to limit the number of articles. That can easily grow using a bootstrap card in a for loop while you iterate through the list of news articles. You could display everything the API returns to you

Flask Web App News Aggregator (Updated Proposal):

Overview: The Flask web app news aggregator is a personalized tech and breaking news feed. Users can set their preferences for two domains from a list of six domains, and receive a curated list of news articles relevant to their selections. The app will display news articles pulled from each domain based on popularity. The app will also include a feature for saving articles, allowing users to easily access and view articles they have saved for later.

Features:

User preferences: Users can set their preferred domains and save them in their browser's local storage. This information will be used to generate the personalized news feed for each user.

News article feed: The news article feed will be generated by retrieving articles from the News API based on the user's preferred domains. The articles will be displayed in a clean and organized manner, with the option to read the full article or save it for later. The dashboard will show all articles returned by the API.

Saved articles: The saved articles feature will allow users to easily access and view articles they have saved for later. This section will be accessible from the user's dashboard, and the saved articles data will be stored in the browser's local storage.

API integration: The API integration with the News API will allow the app to retrieve real-time news articles based on the user's preferred domains.

Development Considerations:

We will use the News API (https://newsapi.org) for retrieving news articles. To address the concern of saving news articles, we will save the entire article object, including the URL, in the browser's local storage.

To handle user preferences and saved articles without implementing user authentication, we will use the browser's local storage.

While developing, we will be mindful of the 100 free API calls per day limit. To avoid reaching this limit, we will use dummy data (e.g., a sample API response) during development for setting up HTML, CSS, and JavaScript.

We will display all articles returned by the API without limiting the number. Bootstrap cards in a loop will be used to iterate through the list of news articles and display them dynamically.

Technologies:

Python 3

Flask web framework

HTML, CSS, and JavaScript + Frameworks for front-end design and interactivity

Bootstrap front-end framework for responsive design

News API for retrieving news articles based on user domain selection

Step-by-Step Plan for Flask Web App News Aggregator:

Set up the project environment:

a. Install Python 3, Flask, and other necessary libraries.

b. Create a virtual environment for the project.

c. Set up a new Flask project.

Design the user interface:

a. Sketch out the main layout of the web app, including the dashboard and the saved articles section.

b. Design the user preferences input form.

c. Create an HTML template for displaying the news articles using Bootstrap cards.

Implement the user preferences feature:

a. Create a form for users to set their preferred domains.

b. Store the user preferences in the browser's local storage using JavaScript.

c. Retrieve the user preferences when the page is loaded, and use them to generate the personalized news feed.

Integrate the News API:

a. Obtain an API key for the News API.

b. Write a Python function to make requests to the News API using the user's preferred domains.

c. Parse the API response and extract the relevant data (title, description, URL, etc.) for each article.

Display the news article feed:

a. Create an HTML template for the news article feed, using Bootstrap cards to display each article.

b. Pass the article data obtained from the News API to the template.

c. Implement a loop in the template to dynamically generate the article cards based on the data provided.

Implement the saved articles feature:

a. Add a "save for later" button to each article card.

b. Create a JavaScript function to save articles in the browser's local storage when the button is clicked.

c. Create a separate section or page to display saved articles.

d. Retrieve the saved articles data from the local storage and display them in the saved articles section using the same article card template.

Optimize the app for development and production:

a. Use dummy data during development to avoid reaching the API call limit.

b. Replace dummy data with live API calls for the production version of the app.

Test the app:

a. Test the app on different devices and browsers to ensure responsive design and functionality.

b. Verify that user preferences are saved and used to generate the news feed.

c. Test the saved articles feature to ensure proper functionality.

Deploy the app:

a. Deploy the Flask web app to a suitable hosting platform.

b. Test the live app to ensure proper functionality and performance.

Maintain and update the app:

a. Monitor the app for any issues and address them as needed.

b. Keep the app up-to-date with any changes to the News API or other dependencies.

Example Response from the news API:

{

"status": "ok",

"totalResults": 10,

"articles": [

{

"source": {

"id": "techcrunch",

"name": "TechCrunch"

},

"author": "Author Name",

"title": "Article Title",

"description": "Short description of the article",

"url": "https://example.com/article-url",

"urlToImage": "https://example.com/image-url.jpg",

"publishedAt": "2023-03-15T10:00:00Z",

"content": "Full article content or a summary"

},

...

]

}

To use the News API in your Flask web app, you'll need to make HTTP requests to the appropriate endpoint with the required query parameters, parse the JSON response, and extract the relevant article data to display on your website. You can use Python libraries such as requests to make the API calls and work with the JSON data.

News API is a JSON RESTful API that allows developers to search and retrieve live news articles from various news sources and blogs. To get articles using News API, you need to make an HTTP request to one of its endpoints with appropriate query parameters, and the API will return a JSON object containing the results.

There are two main endpoints that return articles:

/v2/top-headlines: This endpoint returns the top headlines from selected sources or based on general queries.

/v2/everything: This endpoint allows you to search for articles based on a keyword or a phrase, and you can also filter the results by various parameters such as sources, domains, date ranges, and more.

Here's how you can make API calls using the requests library in Python:

First, make sure you have the requests library installed:

pip install requests

import requests

Then, you can use the following Python code to make requests to the News API:

api\_key = "YOUR\_API\_KEY"

# For top-headlines endpoint

url\_top\_headlines = "https://newsapi.org/v2/top-headlines"

params\_top\_headlines = {

"country": "us",

"category": "technology",

"apiKey": api\_key

}

response\_top\_headlines = requests.get(url\_top\_headlines, params=params\_top\_headlines)

data\_top\_headlines = response\_top\_headlines.json()

# For everything endpoint

url\_everything = "https://newsapi.org/v2/everything"

params\_everything = {

"q": "bitcoin",

"from": "2023-02-15",

"sortBy": "publishedAt",

"apiKey": api\_key

}

response\_everything = requests.get(url\_everything, params=params\_everything)

data\_everything = response\_everything.json()

# Now you can access the JSON data (e.g., data\_top\_headlines, data\_everything) in your code

Remember to replace "YOUR\_API\_KEY" with your actual News API key. The params dictionary contains the query parameters for each endpoint. The requests.get() function sends an HTTP GET request to the specified URL with the provided query parameters, and the .json() method converts the response to a Python dictionary that you can work with in your code.

Milestones and division of work (tentative):

Milestone 1: Project setup and API integration

Tasks:

1.1 Set up the Flask web app environment (Scott)

1.2 Design the front-end layout using HTML, CSS, and Bootstrap (Karim)

1.3 Integrate News API and retrieve news articles based on user domain selection (Segundo)

1.4 Implement dummy data for development purposes (Scott)

Milestone 2: User preferences and news article feed

Tasks:

2.1 Implement user preferences feature, allowing users to select their preferred domains (Karim)

2.2 Save user preferences in the browser's local storage (Scott)

2.3 Create a dynamic news article feed based on user preferences (Segundo)

2.4 Display articles using Bootstrap cards in a loop (Karim)

Milestone 3: Saved articles feature and final touches

Tasks:

3.1 Implement the saved articles feature, allowing users to save articles for later (Scott)

3.2 Save and manage saved articles in the browser's local storage (Segundo)

3.3 Create a dedicated section for saved articles on the user's dashboard (Karim)

3.4 Perform testing and debugging to ensure a smooth user experience (All team members)

3.5 Optimize the web app for responsive design (Scott)

Task Distribution:

Karim:

Design the front-end layout using HTML, CSS, and Bootstrap (1.2)

Implement user preferences feature (2.1)

Display articles using Bootstrap cards in a loop (2.4)

Create a dedicated section for saved articles on the user's dashboard (3.3)

Scott:

Set up the Flask web app environment (1.1)

Implement dummy data for development purposes (1.4)

Save user preferences in the browser's local storage (2.2)

Implement the saved articles feature (3.1)

Optimize the web app for responsive design (3.5)

Segundo:

Integrate News API and retrieve news articles based on user domain selection (1.3)

Create a dynamic news article feed based on user preferences (2.3)

Save and manage saved articles in the browser's local storage (3.2)

Work that can be done concurrently

Milestone 1: Project setup and API integration

Set up the Flask web app environment (Scott) and design the front-end layout using HTML, CSS, and Bootstrap (Karim) can be done concurrently.

Once the environment is set up, integrate the News API (Segundo) and implement dummy data (Scott) concurrently.

Milestone 2: User preferences and news article feed

Implement user preferences feature (Karim) and save user preferences in the browser's local storage (Scott) can be done concurrently.

Once user preferences are implemented, create a dynamic news article feed (Segundo) and display articles using Bootstrap cards in a loop (Karim) concurrently.

Milestone 3: Saved articles feature and final touches

Implement the saved articles feature (Scott) and save and manage saved articles in the browser's local storage (Segundo) can be done concurrently.

Create a dedicated section for saved articles on the user's dashboard (Karim) can be done in parallel with the previous tasks.

Once the saved articles feature is fully implemented, perform testing and debugging (All team members) and optimize the web app for responsive design (Scott) concurrently.