Sharing Synergy

Today we are going to work together to design a distributed RSS feed aggregator system where each student has their own RSS feed, and the system pulls from all the feeds in the class. We'll call this project "Sharing Synergy" so that everyone knows it is going to be a fantastic project that is going to launch us to the moon 🚀🚀🚀.

Sharing Synergy is a practical application of the knowledge portfolio concept discussed in "The Pragmatic Programmer." Sharing Synergy! Will allow students to regularly invest in their knowledge and stay current with technology trends. That's not satire.

The collaborative nature of the project, where feeds are shared and aggregated, mirrors the book's advice on participating in user groups and learning from others. Furthermore, the technical challenges involved in building this system provide opportunities for students to experiment with different programming languages, tools, and environments – all key strategies mentioned in the reading for building a robust knowledge portfolio. Through this project, the hope is that you will not only learn new technical skills but also develop the habit of continuous learning and information sharing, which are crucial for long-term success in the dynamic field of programming.

The goal is for this to actually be useful for us as a class.

In-class Activity

You'll be divided into three groups, each focusing on a different aspect of the system:

1. UX Team: Responsible for user interface and experience.
2. Infrastructure Team: Handles the technical implementation and system architecture.
3. Security and Moderation: How do we keep this from going off the rails?

The project will proceed in four rounds:

Round 1: Group Discussion
Each team will discuss their priorities and ideas for the system within their group.

Round 2: Cross-Team Representatives
Each group will send representatives to the other groups to share ideas and gather feedback.

Round 3: Group Reflection
Teams reconvene to discuss the insights gained from other groups and refine their proposals.

Round 4: Class-Wide Discussion
The entire class comes together to finalize the system design and implementation plan.

Keep in mind that we are going to develop an MVP (Minimum Viable Product) within two weeks.
“We” .. as in you.

Group Focuses:

UX Team

- Design an intuitive process for students to submit RSS material and
- Email submission? Slack?
  - Re-use as much as possible
- UX is not just UI

- Ensure ease of use in submitting content and aggregating feeds
- Must be command-line only. We need rapid development.
- Most important part of the project

Infrastructure Team

- Develop the architecture for the distributed system
- Plan how individual RSS feeds will be created and maintained
- Design the aggregation mechanism to pull content from all feeds
- Most important part of the project

Content Team

- Mechanisms for client-side filtering, aggregation, blocking
    - The core of the product
- Includes some security
- Develop guidelines for appropriate content and sharing
- Most important part of the project

While each team has its specific focus, the goal is to collaborate and create a system that balances user experience, technical efficiency, and utility.

Throughout the process, consider these questions:

1. How can we make feed creation and content submission simple for students?
2. What's the best way to aggregate content from multiple feeds?
3. How do we ensure the system is secure while still being user-friendly?
4. What technical challenges might arise in a distributed system, and how can we address them?

By Next Monday

Use this GitHub classroom assignment to develop your prototype:
https://classroom.github.com/a/G7vaixKE

Every person in your group should design and implement a prototype of your team's part of the system in Python. For the sake of speed, this is going to be a pure command-line app. Remember, UX is not just UI, even in a command-line interface.

You are free to collaborate within your team. Everyone needs their own copy of the code.

UX Team

- Develop a mock command-line interface that demonstrates the full user workflow.
- Include all commands, their syntax, and expected outputs.
- Focus on creating an intuitive command structure and clear user prompts.
- No need for actual functionality - just show how users will interact with the system.

Infrastructure Team

- Create a Python script that can generate and publish a simple RSS file to a local directory (simulating a website).
- Implement a mechanism to maintain and update a list of RSS feed URLs.
- Functionality is key.

Content Team

- Develop a Python script that can read RSS feeds and demonstrate basic content filtering. Aggregation, and summarization.
- Create a basic set of content guidelines and sharing rules.
- Include a mechanism for users to block or filter specific content or feeds. Each user should be in control of what they see.

For all teams

- Your prototype should be runnable from the command line.
- Include comments in your code explaining your design decisions.
- Be prepared to present your prototype for the next class.
- Be ready to discuss how your part integrates with the other components of the system.

Remember, the goal is to have a working proof-of-concept for each component. Focus on core functionality rather than perfection. We'll integrate these components in the following week.