**Question 3:**

1. Write and share a small note about your choice of system to schedule periodic tasks (such as downloading a list of ISINs every 24 hours). Why did you choose it? Is it reliable enough; Or will it scale? If not, what are the problems with it? And, what else would you recommend to fix this problem at scale in production?

*Answer:*

In managing periodic tasks, such as the daily download of ISIN lists, I would opt for Celery—a dependable and widely adopted system. Celery, an open-source distributed task queue for Python, stands out due to its reliability, scalability, and seamless integration with Python applications. Its fault-tolerance allows tasks to be distributed across multiple nodes, ensuring robustness against failures. With the ability to scale horizontally, Celery accommodates varying workloads by adding more worker nodes. The system offers flexible task scheduling, supporting straightforward syntax for setting up tasks to run at specific intervals, like every 24 hours. Additionally, Celery benefits from an active community, providing ample resources for troubleshooting and learning. While Celery is a strong choice for many scenarios, it's crucial to consider specific project requirements and constraints when selecting a task scheduling system. For exceptionally large-scale applications, alternatives such as Apache Airflow or Kubernetes CronJobs may be explored. In summary, Celery's reliability, scalability, and community support make it a suitable choice for efficiently managing periodic tasks in Python applications, including the regular download of ISIN lists.

1. In what circumstances would you use Flask instead of Django and vice versa?

*Answer:*

The decision between Flask and Django hinges largely on the unique demands of your project. Below is an overview of when each might be the preferable choice:

Opt for Flask when:

Your project is a straightforward, lightweight application: Flask operates as a microframework, offering a minimal core and relying on extensions for additional functionalities. It's well-suited for small-scale projects that demand swift development without an abundance of built-in features.

Fine-grained control over architecture is crucial: Flask provides complete autonomy over how you structure your application and which libraries/tools you employ. This flexibility proves advantageous for meeting specific requirements or steering clear of vendor lock-in.

You're already proficient in Python: Flask's code is uncomplicated and easily graspable for Python developers, even those lacking extensive web development experience.

You're constructing a microservice or API: Flask excels in crafting lightweight APIs and microservices that seamlessly integrate into larger systems.

Opt for Django when:

Your project requires a sophisticated, feature-rich application: Django stands as a comprehensive web framework with a "batteries-included" philosophy, furnishing an extensive set of features like authentication, authorization, and database support. This proves beneficial for expeditiously developing larger projects with numerous common functionalities.

You prioritize faster development and simplified maintenance: Django's integrated features and structured methodology streamline development processes and enhance long-term codebase maintenance.

Collaboration with a team of developers is essential: Django's conventions and standardized structure facilitate seamless collaboration among multiple developers on a project.

Robust security features are imperative: Django boasts built-in security measures that fortify your application against prevalent vulnerabilities.

Specific Use Cases:

Use Flask for:

Creating a simple landing page for your portfolio website.

Developing a small API to expose data from a sensor network.

Designing a basic internal tool for your team.

Use Django for:

Establishing a large e-commerce website.

Implementing a content management system.

Constructing a complex social networking platform.

Ultimately, the optimal approach involves scrutinizing your project's unique requirements and selecting the framework that aligns best with those needs and the strengths of your development team.