

## Part A

Choosing a system to schedule periodic tasks, such as downloading a list of ISINs every 24 hours, is a critical decision. While I'm not an expert in this area, I can provide some insights into a commonly used tool: **cron**.

### Cron:

#### Why I Chose It:

- Cron is widely used and trusted for task scheduling on Unix-like systems.
- It's relatively easy to learn, configure, and set up.
- Scheduling tasks at regular intervals can be achieved with a simple cron expression.

#### Scalability:

- For smaller-scale applications, cron works well. However, it may not be suitable for large-scale, distributed, or more complex task scheduling requirements.

#### Potential Problems:

- Cron can become challenging to manage as the number of tasks and their complexity grows.
- It may not handle distributed scheduling or provide centralized monitoring.
- Error handling and reporting are often basic and manual.

#### Large-Scale Production:

- For larger-scale or distributed systems, consider job scheduling and orchestration tools like Apache Airflow, Celery, or Kubernetes CronJobs.
- These tools offer advanced scheduling capabilities, centralized monitoring, error handling, and are designed to scale.
- They provide more robust features, but they have a steeper learning curve.

In summary, while cron is a reliable choice for simpler periodic tasks, it may not be the best fit for large-scale, complex, or distributed systems.

## Part B

When deciding between Flask and Django for web development in Python, several factors come into play.

Flask is the preferred choice when simplicity and minimalism are key. It's a lightweight micro-framework, offering a basic structure, and grants us flexibility to select the components they need. Flask is an excellent choice for small-scale or straightforward projects.

On the other hand, Django is ideal for complex, large-scale web applications. It's a high-level, full-stack framework that includes built-in features like an ORM, authentication, and an admin panel.

Django emphasizes convention over configuration, making it a speedy choice for project development.

In essence, Flask prioritizes flexibility and minimalism, while Django streamlines development with built-in features. The choice between them hinges on the project's size and complexity.