Code Explanation

Model-View-Template Pattern

In this project by using Django Framework, we implemented the Model-View-Template architectural pattern.
Our model represents the database structure, the view contains the logic for processing user requests and returns them the appropriate responses, *views.py* processes HTTP requests, interacts with our database in this case a csv file and passes the data to the templates which are stored in templates folder.
The *templates* folder stores user interface files, and those html files represent how the data will be rendered to the user.

Factory Pattern

Factory Method design pattern is used to dynamically create and execute HTTP requests for retrieving stock data over specified intervals for a range of users. In *scraper.py* the function *fetch_company_data* acts as creator that centralizes the request generation logic, and the HTTP requests are different issuers and date intervals.
The request creation logic is centralized in one function, which makes the code easier to maintain and extend. Since it is designed asynchronously it supports high scalability.

Pipeline Pattern

This pattern is used to streamline the flow of data through various stages in our system.
The process of fetching data, adding technical indicators and saving them to a csv file represents pipeline pattern. In first step we collect raw data for each company on each date from the web page, then apply some filters to the data and save the data to the file. On next step we transform the data and do calculations to compute technical indicators, and for last step we save the processed and non-processed data to different files.

Scheduled-task Pattern

This pattern allows us to execute a task at specific intervals. By using the *BlockingScheduler* from *apscheduler* library we are schedule and manage tasks. Standalone functions are defined to have decoupling which we will ensure that the logic of the task is not dependent on scheduler’s operation, which later will enable us reusability.