# Project modules

For our Financial Analyst Assistant Portal, where we're working with multiple modules, it’s crucial to organize the project in a professional, maintainable way. Below is a suggested project structure that aligns with Flask, Jupyter Notebooks for prototyping, and scalability for future modules.



## Explanation of Key Components:

1. **app/ Directory**: Contains all application-related code.
   * **\_\_init\_\_.py**: Initializes the Flask app and configures the app settings.
   * **routes.py**: Defines the endpoints (like file upload, Q&A, and news retrieval).
   * **models.py**: Database models if you're using something like SQLAlchemy for handling users, companies, etc.
   * **utils/**: Modularize your utility functions:
     + **nlp.py**: Functions for your NLP and deep learning tasks.
     + **news\_scraper.py**: Code for scraping news related to specific companies.
     + **file\_processing.py**: Handles document uploads and any parsing you might need for analysis.
   * **templates/**: HTML templates for rendering your front-end.
2. **static/ Directory**: Static assets such as CSS, JavaScript, and images.
   * **css/**: Contains style files for the user interface.
   * **js/**: JavaScript for any front-end functionality (e.g., dynamic content).
3. **notebooks/ Directory**: This is where you'll keep your Jupyter notebooks for experimenting with code before finalizing it into Python scripts.
4. **tests/ Directory**: Unit and integration tests. Writing tests will help ensure that your modules work correctly when integrated.
   * **test\_nlp.py**: Tests for the Q&A and NLP functionalities.
   * **test\_scraper.py**: Tests for the web scraping functionality.
   * **test\_routes.py**: Tests to ensure the Flask routes work as expected.
5. **Configuration (config.py)**: Centralized configurations like API keys, database settings, and environment variables.
6. **requirements.txt**: All the Python dependencies. As your project grows, you can freeze the libraries to ensure consistent environments.
7. **run.py**: The entry point for the Flask app. This file runs the app when executed.
8. **README.md**: Detailed project documentation for setup, running the app, and the features included.

## Workflow:

1. **Prototyping**:
   * Use the notebooks/ folder to experiment with your models (e.g., document analysis using NLP or scraping news) before moving the final code into the corresponding utility files.
2. **Developing Core Modules**:
   * Break down each module into its own script in the utils/ folder (e.g., nlp.py, news\_scraper.py).
   * Handle Flask routes in routes.py, calling these utility functions as needed.
3. **Deployment**:
   * The Flask app (app/) structure is scalable and can easily be deployed using services like Heroku or Dockerized for cloud deployments.