Programming in Finance II: Big Projects 2025

Peter H. Gruber

April, 2025

General. The projects are deliberately formulated in an open way. A detailed formulation of the problem and a project plan is already part of the project. This includes the decision, which features to implement beyond the minimum requirements and how to document your project. In any case, focus on the implementation and the computing.

The main goal is to produce one working solution. Complexity is rewarded, but everything that you hand in has to work.

Document all steps of your project such that it is reproducible. This includes all code and all commands. If using several environments or programming languages, make clear which code is written in which language.

Presentation.

– Present your project plan in a 15x3 group presentation.
– Six minutes per group
– You decide who speaks

Delivery via Github

– Publish your project on Github. You decide on the specific format.
– Include all steps for installation and deployment in the documentation.
– User guide

Academic documentation.

– Deliver the project documentation in PDF on iCorsi (5-8 pages pure text)
– Use LaTeX
– Link to Github page
– Project plan
– Project diary (your steps, what you did, why, which tools)
– Economics/mathematics/data science behind your project
– Sample results or applications and reflection of (economic) results
– Lessons learned from the project

Criteria

1. Problem definition
   Creative? Useful? New?
2. Problem solution
   Complete? Works? Realistic results? Suitable algorithms/methods?
3. Scope
   Did you do the work? Difficulty/complexity/comprehensiveness
4. Coding: structure, style
5. Quality and depth of Documentation

Material from the web or from other courses

Use of foreign code (libraries, tutorials from stackexchange or others) is permitted. Foreign code (=anything that you found on the web and that you did not write yourself for this project) needs to be cited correctly. Foreign code is ignored when assessing your contribution (i.e. it counts neither positively nor negatively towards the contribution. This implies that if you used only foreign code, there is no contribution from your part).

It is absolutely forbidden to use one project in two courses. If you work on similar problems in two courses, make sure that your project is a safe distance away from anything you or your colleagues are doing any other course.

Use of generative AI

Use of generative AI is expected and should be acknowledged (which tools?) in the PDF documentation.

1 Generic project criterion

In 2025, the rule for projects is just this: Combine at least three of the following elements in your project.

- Your own linux server with cron and database
- LLMs via API
- Local LLMs
- Python package
- Web front end
- Monte Carlo simulation
- Web scraping
- Serous data visualization

2 The 2024 projects as inspiration

These projects serve as *inspiration* only. It may be the case that projects DO NOT fulfil the *Generic project criterion* above. Developing your topic is part of the problem.

2.1 Outsourcing

Minimum requirements: Find an interesting problem, involving (a) data science, (b) asset pricing or (c) a consumer-facing web product. Write a detailed specification book. Find a contractor. Communicate with the contract and supervise delivery. Depending on your specification, have the contractor deploy the product or deploy it yourself. Document all interactions with the contractor. Do not write a line of code, however, create the user documentation and project documentation (containing all communication with the contractor). **Nice to have:** design work by (the same or another) contractor, mobile solution

2.2 Set up a personal data server

Minimum requirements: Set up a Linux server with a VPS provider, set up a database using SQL commands, write a program that downloads some data and writes it into the database, set up a cron job to regularly execute this task. Provide one API (text or graphics). **Nice to have:** create a dashboard, register a domain name, set up https

2.3 Python Package and open source software

Minimum requirements: Find an interesting and relevant topic, create a Python package with at least three different functions and a sample dataset, create standard documentation for Github, deploy on GitHub (with an open source license), create additionally a Jupyter notebook with an example of installing and using your package. **Nice to have:** interaction of your package with an LLM, an API or web data.

2.4 R Package and open source software

Minimum requirements: Write an R package that serves as API to access the services of `https://groq.com`. Be inspired by the Python package that implements the API for OpenAI.

2.5 Nocode program

Minimum requirements: create an app that allows for user sign up/sign in, that allows the user to enter data in some form and that displays some data in some form on the mobile phone screen. Additionally, retrieve data from an API (yours or a public one) and embed the API results into the app. Deploy the app in such a way that it can be tested on an Android phone and/or iPhone. Take special care of UX design. **Nice to have:** enable local storage on the phone, allow for upload from the phone, enable additional Nocode elements

2.6 LLMs and RAG

Find an interesting question that can be answered with RAG (Retrieval Augmented Generation). Procure a larg(er) corpus of texts, e.g. from EDGAR and create a vector store. Set up your an LLM (local, hugging face, openAI, or groq.com). Analyse your text corpus in a *systematic* way and produce systematic output. **Nice to have:** run a regression or produce charts of your results.

2.7 Local LLMs

Set up a (larger) VPS server with at least 2GB RAM (better more). Install a (GGUF) quantised open source LLM on the server. Using FastAPI or any other web-based technology, provide an API or a web interface to interact with the LLM on the server. **Nice to have:** create a web service based on a custom system prompt, enable prompt security.

2.8 Trading

Minimum requirements: Implement a simple trading strategy in a *robust* way. The strategy should involve at least limit orders. Returns are not important, what is important is knowing what you do and verifying everything twice, e.g. verify execution of limit orders, verifying buying power before buying something, verifying whether an asset is short-able before shorting it etc. **Nice to have:** reporting (graphics?) and/or backtesting

2.9 Smart contracts on the Algorand blockchain

Minimum requirements: create one or multiple account(s), create an Algorand Standard Asset and write and deploy a Smart Signature on the Algorand Main Net. **Nice to have:** write a smart contract, implement security