Detailed Brief: Final mini-project

for Advanced Algorithms and Programming

In this brief you will be given the details of the final mini-project to develop during the rest of this unit. We will work during class and at home.

The general idea is that you propose and develop your own tool that uses a series of algorithms combined for solving a particular problem you define.

1. **Framework:**

You will use *Python* for developing your algorithm. You are not forced to use *Marimo*, you can use *Jupyter* notebooks, or you can create your own standalone *Python* app, or any other *Python* implementation. Just stick to *Python*.

It could happen, however, that you need to use a different framework for a peripheral function (i.e. for something that is not the core of your tool). For example, your whole code is in *Python* but you need *Javascript* for visualising the output in the browser. In these cases, you can contact me and we can discuss on adding this extra framework, as long as most of you code is implemented in *Python*.

2. **Expected output:**

The final output of your mini-project includes:
I. A **report** of the design and development process. The report should at least include:

a. Introduction: What did you do and why? What problem is it tackling? To what other previous work it relates?
b. Design Diagram: Include a decomposition of the problem (as we did in first class) AND a flowchart of your proposed solution.
c. Tool: Description of how each part of your tool works (inputs required, technical specifications, details or parameters used for different algorithms).
d. Time Complexity Analysis: You will select one low-level algorithm that your tool is using (for example: sorting, searching, genetic algorithms, graph path search, etc) and you will optimise it for your use case. You must justify why is that particular option chosen. For example, if you created a taxi searching app, you would implement some sorting algorithms as part of this. Then, you would have to justify why did you use merge sort, or quicksort, etcetera, based on the data structures you are using. You will use the O( ) notation to indicate how is time complexity increasing with input size n.
e. Critical aspects: You will critically review your tool and algorithms. You will discuss what potential biases could it have, which populations could be

negatively affected by it, what biases could it have, and what have you done (or what could be done) to counteract these effects.

II. A semi-functional or fully functional **prototype of your tool.**

You will implement in *Python* the proposed tool. The user should be able to run your tool only with the files and script you provide in the submission. No extra sources or files should be needed. You should include a README file to guide the user on how to run the app.

The app should be functional in terms that it allows the user to navigate and understand how it works. The tool won't be judged on "look and feel" aspects, but you are free to add graphical and design aspects to the interface if you want. Also, you can simulate external datasets if needed (for example, if you app is supposed to have some user-related data collected from somewhere, you can simulate it through a pre-defined *.csv* file, or other).

III. (Optional): If you have a non-fully functional prototype and you want to show how would the final product look (for example, you would like to show the end user interaction with the tool), you can also create a mockup design of the final product (for example, in *Figma*, *Canva*, *Photoshop,* among others). You can use this for displaying the "look and feel" aspects of the tool, which are not necessarily accounted in the *Python* version.

3. **Submission:**

Deadline for submission is **Friday 12th of December, 2:00pm.**

You will submit in Moodle a **zip file** containing:

a. Every weekly task solved available as independent Marimo notebooks.
b. The final mini-project folder, containing the report, the code, and any other supporting material (for example, files or links to mockups, or README file for running the code successfully)

The submitted zip file structure should look like something like this:

```
Final_Submission/
├── week1/
│   └── week1.py
├── week2/
│   └── week2.py
├── week3/
│   └── week3.py
├── week4/
│   └── week4.py
├── week5/
│   └── week5.py
├── ...
├── mini-project/
│   ├── report.pdf
│   ├── repo/
│   ├── README
│   └── Extra Material/
```

You should also have all of your work available to see in your online portfolio (*git.arts.ac.uk*), where the final project should have its own subpage, where you can show your tool as you like (you can deploy your *Python* code, if possible in the web, so the tool can be used directly, or you can add mockups and a demo video, etcetera). Your whole portfolio should look professional and tidy, with proper explanations of what are you presenting on each part. Feel free to add *javascript*, *css* or other web frameworks to your main page (NOT to the weekly tasks, those are always in *Marimo*) to make it more appealing.

4. Marking criteria

Final mark will be given by weekly tasks (50%) and mini-project (50%)

Marks for mini-project will be given according to the following criteria:

Criteria	**Description**	**Weight**
Problem formulation	Clarity, concreteness	20%
Algorithm design	Correctness and creativity	20%
Time Complexity Analysis	Formal reasoning and empirical support	20%
Critical aspects	Biases detected, measurements proposed	20%
Report and code presentation	Organisation, readability, comments and acknowledgement	20%

5. Acknowledgements

You are required to acknowledge, both in your code and report, any use of AI, following the AI tool guidance given in class (available in Moodle).