Task 6 – Realtime Digital Filter Design

Implement a desktop application that helps users to design a custom digital filter via zeros-poles placement on the z-plane. Your application should have the following features:

- A plot for the z-plane with the unit circle, where the user can place different zeros and poles. The user can also make the following modifications:
  - Modify the placed zeros/poles by dragging them,
  - Click on a zero or pole and delete it,
  - Clear all zeros or clear all poles or clear all,
  - Has the option to add conjugates or not for the complex elements,
  - Zero-pole swapping: The user can swap all the zeros into poles and vice versa,
  - Undo/redo all the above operations,
  - Save/Load functionality (into notepad or csv files) for the designed filters,
  - Filter realization (direct form II & Cascade) with exporting option,
  - Automatically generate C code for the designed filter.
- A plot that shows the corresponding frequency response for the placed elements: One graph for the magnitude response and another graph for the phase response.
- A built-in library of at least 10 famous digital filters of all types (LPF, HPF and BPF) such that Butterworth, Chebyshev, inv Chebyshev, Bessel, and Elliptic filters.

Upon finishing the filter design and visualizing the filter response, the user should be able to:

- Apply the filter on a lengthy signal (minimum of 10,000 points) as if it is a real-time filtering process. A graph should show the time progress of the signal (i.e. do not show all the signal on the graph at once), and another graph to show the time progress of the filtered signal (i.e. upon applying the difference equation on the points one by one) as well as the phase-corrected (as explained below) filtered signal. The user should be able to control the speed/temporal-resolution of the filtering process. For example, the filter can process 1 point per second or 100 points per second or any number in between via a slider.
  - The user can also input an arbitrary real-time signal via moving the mouse on a small padding area. The input signal would be the x- or y- dimension of the mouse coordinate. The faster the user moves the mouse, the higher frequencies the generated signal will have and vice versa (i.e. slower motion → low frequencies).
- Correct for the phase by adding some All-Pass filters. The user can pick the suitable all-pass through a library available in your application. This needs the website to have the following features:
  - A library of all-pass filter that the user can visualize (its zero-pole combination as well as its phase response), then pick one or more to add to the original design filter.
  - A custom-built all-pass: if the user cannot find a good all-pass in the provided library, then s/he builds his own. i.e. provide an arbitrary "a" and the website would calculate its phase response and integrate it with its library.
  - The user can enable/disable the added all-pass elements via a drop-menu or checkboxes group.

Examples:
https://www.earlevel.com/main/2013/10/28/pole-zero-placement-v2/
https://www.earlevel.com/main/2016/12/08/filter-frequency-response-grapher/
https://www.micromodeler.com/dsp
Notes: Examples are provided to give an idea about the parts of the requirements. They do not represent the whole requirements and you are not obligated to replicate it. Please, use your own design and think about how the user can easily use your tool.