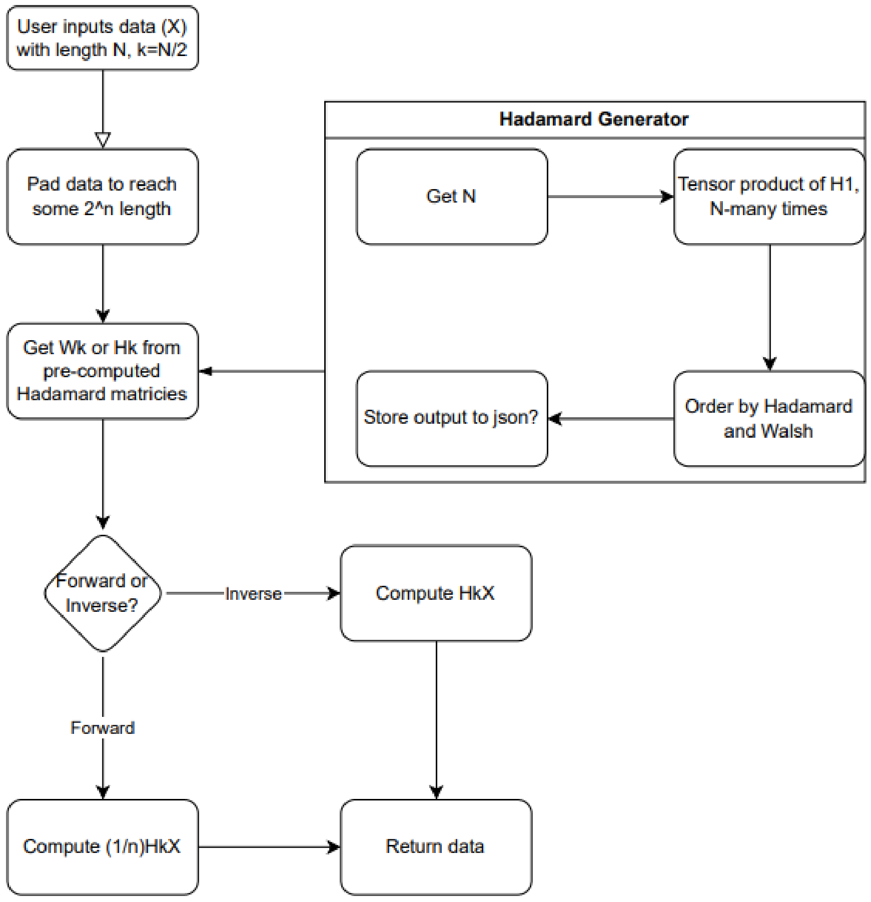
Progress Update 2

1. Minimum Viable Product:
   i. A Maui Blazor application for Windows that takes data from the user (.csv or json) and implements the Walsh-Hadamard transform to compress the data and return it. This application will run both the forward and inverse transform. It should be user friendly and professionally designed.
2. Components:
   i. Walsh-Hadamard Transform algorithm:



User inputs data (X) with length N, k=N/2
Pad data to reach some 2^n length
Get Wk or Hk from pre-computed Hadamard matricies
Hadamard Generator
Get N
Tensor product of H1, N-many times
Order by Hadamard and Walsh
Store output to json?
Forward or Inverse?
Inverse
Compute HkX
Forward
Compute (1/n)HkX
Return data

ii. Hadamard Generator:

We need to create a secondary program that generates the Hadamard matrices and returns them in both sequency and natural ordering. This can be in any language as long as it returns the data as a filetype that we can consume in the application. This should save computation time since we expect to be using very large matrices with the WHT.

iii. Maui Blazor application:

We will create a Blazor Maui application. The base application will allow the user to upload a file which will send it to the backend through the API. It will then wait to receive the transformed data back and return it to the user as well as graphing the data. We hope to expand this side of the project to also include widgets so the user can play with their data and possibly different types of transforms.

iv. Multi-Platform Application

We chose to use Maui Blazor because of its ability to deploy to multiple platforms. Our base product will be a native windows app but we would like to extend that to other platforms such as Andriod, IOS and the web.

v. Graphing API

We need to define a pipeline from user input to backing algorithms. We will be using Razor (.cshtm) pages to develop the UI so we will also need to find a graphing library that we can use to graph data. We plan on creating an API that will expose both this graphing library and our algorithms in a way that will be easier for the front-end to code. This would allow for the front end developer to just call a simple function that takes input data and returns graph data without needing to know the details of the interaction between the graphing and transform libraries. (i.e. ForewardWHT(userData).drawGraph() should return a graph object for display).

vi. Documentation/Planning:

We will need to plan and document our work if we want to meet the deadline and our advanced goals. We would like to use Pull Requests to maintain a clean main branch.

3. Tasks:
    i. Algorithms:
        1. Derik will plan out the algorithms to be implemented.
    ii. Hadamard Generator, WHT Code:
        1. Cain and Calvin will work on these together.
    iii. Maui Blazor Application:
        1. Finn will work on initializing the application architecture and define appropriate areas for code. He will help integrate the

other’s work as it gets to a state where it can start being tested and define the API interface.

iv. Multi-Platform Application

   1. Finn will work on implementing the application to several more platforms. At the very least we want to also have an Android implementation. After that we will see what other platforms are doable.

v. API:

   1. Chet will work on the API that will take user data and return transformed data to be graphed.

vi. Documentation/Planning:

   1. Derik will maintain documentation and plan out work for the other developers. He will hop between tasks and help get past bottlenecks.