MetaModel:

Our Meta Model is constructed referring to the mechanics translational content in MSL (Modelica Standard Library). The *Model* domain is where we define our 1-dimensional translational mechanical systems and it contains four child nodes as listed below:

(a) *Flange* is a port type Meta. It act as the sources or destinations of inter-block connections.
(b) *Components* is an abstract Meta and it has two attributes: ModelicaURI and init, which are used to specify elements in MSL and set the initial condition of each elements respectively. Its child nodes are *Single_Flange_Component* and *Two_Flange_Compon* who also serves as abstract Meta and contain valid definitions of mechanical elements divided by the number of ports . Every mechanical components have different parameters and initial states. Their parameters and default settings are defined in individual attributes while their initial state can be set through the common attributes 'init'.
(c) *Sources* is an abstract Meta used to indicate the driven forces of the whole model. It also contains the attribute 'ModelicaURI' which do the same thing as it does in *Components.*
(d) *Sensors* is an abstract Meta used to indicate the built-in sensors of MSL.

When building 1-dimensional translational mechanical systems, Sources and Sensors is not required to be added.

Interpreters:

Our interpreters contain two parts: Code Generator and Simulator.

(a) Code Generator:
The code generator first loads the node map and then extracts the data to build connections and components. Then it begins to write modelica code including connections and components into the .mo file and upload the it into the blob storage for the further simulation.
(b) Modelica Simulator
OpenModelica supports scripting of model-editing and simulation via mos scripts. We will get a simulate.mos file after we run the modelica simulator. Inside the modelica simulator, the command "omc simulate.mos" will be executed to invoke modelica tool to do the simulation. The simulate.mos code will load to the and simulate the Translational mechanics model we create before and store the results in a .csv file.

We put an "invokeplugin" inside the simulation code, so to get the result of the simulation, only "Modelica Simulation" is enough.

Visualizer:

Visualizer is very import for our project Modelica. Our visualizer should be able to draw a graph based on our simulation which contains a csv document that contains many numerical data.

We would be able select the attribute they wanted to see and paired with time attribute. After the selection, we can click the button we created in our visualizer and the graph would appear on the web. We could visually see the result of the simulation. That would be very useful for them.

For the implementation of the visualizer, we need to create a button, a selector, and a div element to display the graph first. Then, we connect them all together. So, the selector would pass the selected option to the graph part after the clicking. The graph part would generated the graph based on the selected attributes by using the plotly library. Every data would be passed from the control side.