Modules

Database: The database module contains all of the code required for interacting with the mongoDB database. This includes functions for saving, removing, and manipulating the database objects which represent device and simulation states.

Server: The server module is responsible for initially serving the web application to the client, as well as handling various requests for information provided by the client. The server facilitates communication between the database and client modules.

Client: The client module has various responsibilities. As such, we've broken it into a variety of sub-modules:

Client: Views: The Views module is responsible for managing the appearance of our web app, as well as how it behaves when information changes. This handles correctly displaying the pages which occur in our simulation and handling moving in between pages

Client: HTML Templates: HTML templates module is responsible for the actual html of all of our pages, this is fetched by the views modules and then sent to the client to view.

Client: Simulation Logic: This module manages the logic of the simulation, such as how networks are added.

Client: Local Storage: The local storage module manages how simulation and user data is stored on a device.

Client: GUI Rendering: All files related rendering the GUI on the client side

RDTs: All files pertaining to the replicated data types, their upload, and handling

Test Scripts: All files pertaining to handling test scripts for our simulation. This includes, their upload, verification and handling.

Applications: All files pertaining to handling applications for our simulation. This includes the upload of these applications, verifying that the replicated data types they use are in the simulation, and running these applications.

Module Design Decisions

We chose these modules as we believe that they cover the core aspects of our system. As well, these chosen modules are designed to be decoupled. Decoupling the modules allows for people to work separately on the different modules without worrying about dependencies with other modules. We believe that this will cause our software to be malleable and reusable for future iterations, and therefore provides good architecture.