# Progress This Iteration

The following is a list of progress which we have made this iteration.

1. **Refactored Database:** This included reverting to the old version of mongoDB which was required, and rewriting the database connection from the ground up in order to handle new schemas which we used in order to unify the representation of objects on both the database and server.
2. **Refactored Client-Side Code**: We entirely refactored the client side logic in order to handle our new concept of an event queue. This allowed us to streamline our previously messy code, and modularize our functionality. This can be seen within the "public" folder for our code.
3. Event Queue: Previously, all logic was carried out on both the client and the server. In this iteration, we implemented an "Event Queue". The client fills the event queue with actions which happened on the client side, and sends this event queue to the server to be processed. The server then returns a new state of the simulation to update the client. This allows our application to be a lot faster, and more error-free, while maintaining the offline functionality of our client-side code.
4. **New Underlying Data-structure:** In order to handle the amount of data which is sent within our server, we created a new data-structure which we call our "partition-list" in order to handle all of this information in a clean and efficient manner. This partition-list is built from the information stored in the database and is then propagated to the client to be handled.
5. **Socket IO**: We implemented socket IO in order to allow a better connection with the server. Rather than constantly communicating with the server, the client now only communicates with the server when necessary. This allows for much less overhaul for the server.
6. **Integration of Network Topology View**: In the previous iteration, we had created the network topology, but manipulating it did not change the internal representation of the topology on the server. In this iteration, we fully integrated the network topology GUI with the actual topology of the simulation. This topology is uniform throughout the application, and can be restored from the database.
7. **Realization of Topology View History:** In the previous iteration, the simulation history has not been fully integrated with the application. This iteration, we ensured that the simulation history is correctly stored in the database, and can be viewed in our simulation history view on our web page. Pressing on one of the dates on the side of our page will display the topology at that point. As well, this displays all of the events which have occurred on the simulation, including all of the manipulation to the topology up to this date. Pressing on one of the devices in that topology will then allow you to view the history for the device (all interactions which that device has had with the applications and replicated data-types.
8. **Topology View Population Algorithm**: In order to handle the topology being loaded from the database, we created a new algorithm to place the topology within our canvas. This algorithm is dynamic, and functions differently based on the number of devices and number of networks being populated on the page. We have calculated it in order to be what we believe to be the most view-able pattern of devices and networks.
9. **Partition Splitting Algorithm**: For this iteration, in order to break apart partitions of networks in our topology, we needed to develop an algorithm to determine which networks will be in the two split partitions. This is complicated by the fact 'transitive closure' of partitions. Thus, we developed a modified version of breadth-first-search in order to determine what networks are in the split apart partitions.
10. **Replicated Data Types:** This iteration, after discussing with Dr. Fiech, we implemented the ability to import and interact with replicated data  types in our simulation. This includes

propagating RDT's to different devices in the simulation.

11. **Applications:** This iteration, after discussing with Dr. Fiech, we realized the application feature, which allows users to import applications, according to a specific schema which is detailed in our documentation and on the 'applications' page on our website. This allows for applications to be uploaded, propagated to devices, and paired with replicated data-types. These applications may then be viewed by the devices.
12. **Automatic Test Scripts:** This iteration, we began planning for the structure and handling of automatic test scripts. This has been outlined in our prototype page 'test scripts' on our web page, as well as in our documentation. Automatic test scripts will allow a simulation to be automated and replicated certain operations at certain speeds in order to test out the effectiveness of applications and replicated data-types.