**Project Objectives**

**First Milestone– Eventual Consistency**

In this first milestone, I will look at implementing eventual consistency over an acid database. The back-end of my system will be accessed through a web service. I will also build a front-end that shows what pieces of data are currently inconsistent with the master copy of the database. A user will be able to configure the amount of consistency that the application uses. At one end of the scale, requests are just passed straight through to the database, at the other end of the scale, requests are stored in the JVM for a period of time before being processed.

Features I want to implement here include the ability to add data to the database, modify data already in the database and be able to retrieve data. I will also develop a web service as part of the front-end that shows the data that is currently inconsistent with the master, and also shows how much time the user has left for a refresh.

In terms of the features that I would expect this version of the software to be capable of the plan would be for the user to be able to perform all the standard database operations through interacting with my system. I would also expect eventual consistency to be simulated, as described above, and furthermore to be simulated whenever one of these operations is performed. I would then provide the means for the user of the application to have a window in on the eventual consistency operations being currently performed, such as through the web page to show consistency. I would also add parameters to control the amount of consistency used in the application.

**Second Milestone – Basic Availability**

The second stage of the project would be to develop the features for emulating basic availability within the application. This would allow me to build on top of some of the features implemented for the eventual consistency requirement, making development much easier. The main requirement of the feature would be that it should be possible to simulate certain parts of the system going down and not have the whole system go down.

In order to accomplish this, I would first have to look at and implement a strategy to ensure that when one of the nodes goes down, the system has an acceptable method of recovering any lost data and getting back on track. As a result of this, the first part of this milestone will be based around developing policies to facilitate this transfer.

Once this has been accomplished, I would need to find a way of visually representing what’s going on to the end user. This could be by modifying information about the actors of the system. As well as showing consistency levels, they also show the time since an actor was last disabled.

I would then allow the user to choose the system’s level of availability. This could be through allowing the user to send individual components of the system’s shutdown message, as well as allowing them to do it themselves if they become overloaded.

**Third Milestone – Evaluation**

Once these two properties have been implemented, if there is time, I would like to take the sample applications that I have developed for the two previous milestones and run them on a real cloud platform like Amazon’s Cloud in order to test how comparable the systems were.

A major part of the iteration would be adapting the sample programs that I’ve written in order to allow them to run on whatever cloud platform I choose. I would then spend time devising some meaningful tests to allow me to both analyze the system to ensure that it effectively emulates BASE properties, and will also evaluate its effectiveness compared to real systems.