**Jack Davey**

**Project Plan for Testing And Tailoring Cloud Storages**

**Milestone One**

This milestone concerns the implementation of eventual consistency over a standard ACID Database.

**Iteration 1**

**Start Date: 8th June 2015**

**End Date: 14th June 2015**

**User Story 1**

**Allow the user to perform basic creation and deletion of data in the database.**

**User Story 2**

**Allow the user to insert, update and delete data from existing tables**

This should work by the end user being able to pass in an SQL query as JSON and have this executed by the database. The results should then be sent back as another piece of JSON. This feature will not implement eventual consistency at this point, and is partly here to allow me to ensure that I am comfortable with the tools that I am using to complete the project. Later iterations will involve the parsing of user queries so that they can be persisted in the database for certain periods of time. I will also develop a small HTML page that allows the user to type in SQL queries into a text box and then have them executed by the backend Database to ensure that the system is working as planned.

**Iteration 2**

**Start Date: 15th June 2015**

**End Date: 21st June 2015**

**User Story 1**

**Allow the user to insert, update and delete data from existing tables. Eventual consistency must be active across all of these operations.**

This story will involve the user passing in an SQL query as JSON before. I will most likely modify the format of the JSON to make it easier to parse. Rather than being sent straight to the database as before, the query will be stored in the application for a period of time. Once this time has expired, the application will then store the query in the database, following the procedure for achieving eventual consistency as outlined in my design.

**User Story 2**

**Repeat the process of adding eventual consistency for inserting and deleting data**

This story will enable insertions and deletions on the data to be eventually consistent. This is second on the list for this iteration because for deletions I will need to think very carefully about how to apply the eventual consistency algorithm with regards to stale data floating around the system. An example of this is that once a record is deleted, I will need to think about how to erase this from all the other parts of the system so it doesn’t end up getting added back in.

**User Story 3**

|  |
| --- |
| **Allow the user to synchronise the system (make everything fully consistent)** |

This should be fairly straightforward to implement, as all it involves is providing the user with a mechanism by which to make the system fully consistent without waiting for the system to do that by itself.

**Iteration 2**

**Start Date: 22nd June 2015**

**End Date: 28th June 2015**

**User Story 1**

**Add a web page that shows the current consistency status of the system and also add customization options**

This user story will involve writing a piece of functionality that allows the user to look at the pieces of data that are not currently within the system. For each piece of data, the web page should show what the change is, (such as updates, deletes, inserts) etc., as well as the time at which it is due to become consistent with the rest of the data.

**User Story 2**

**Write some sample programs that show that the system is working**

This story will involve writing some small test applications to ensure the application is working as it should be. I plan to use code from some of the other functionalities to achieve this goal.

**Milestone 2**

This milestone will be devoted to adding the second BASE property of eventual consistency to the system.

**Iteration 1**

**Start Date: 29th June 2015**

**End Date : 5th July 2015**

**User story One**

**Research common algorithms for basic availability**

Some time in this first iteration will be spent going over the possible algorithms for basic availability and trying to find something that both is well suited to my application and is reasonable to implement in the time that I have available to me.

**User Story Two**

**Implement a mechanism inside a component of the system so it can be shutdown gracefully.**

In this iteration, I will develop a piece of code inside each system component so that it can shut itself down gracefully and any data it is holding can be transferred to another part of the system .

**Iteration 2**

**Start Date: 6th July 2015**

**End Data: 13th July 2015**

**User Story One**

**Based on the Basic availability algorithms researched during the last iteration build in a complex strategy so that the whole system adapt to failures.**

This story is mainly about taking the algorithm implemented in the last iteration and implementing it in code form, It needs to cater for a wide range of situations and challenges

**User Story Two**

**Develop a mechanism for displaying Basic Availability to end users.**

This story is about finding a way to show how available the system is to end users, it could involve providing a detailed service to report on failures, or maybe some statistics that show how much of the time the service has been down

over a certain period.

**Iteration 3**

**Start Date: 14th July 2015**

**End Date: 20th July 2015**

**User Story One**

**Allow the user to customize the amount of baic availability provided to end user.**

**User Story Two**

**Produce some sample applications to show the system working.**

As I haven’t decided hwo I’m going to implement basic availibiilty yet, I cannot say for certain what I am gong to do here. It will have similar aims to the code implemented for eventual consistency however.

**Milestone Three Evaluation**

**Iteration One**

**Start Date: 21st july 2015**

**End Date: 26th July 2015**

**User Story One**

**Modify the sample applications produced for milestones One and two so that they run on a cloud infrastructure rather than on my system**

**Iteration Tow**

**Start Date: 26th July 2015**

**End Date: 26th July 2015**

**User Story One**

**Compare the sample applications to my system and try to gauge how effective the system is at simulating eventual consistency.**