Booksterdam Log

**11 November 2016 (5hrs)**

* Was able to manipulate the database hosted in azure using SQL Server Management Studio.
* Wrote queries to create database schema, and some insertions and selections
* **ISSUES**:
* INT data type can’t hold more than 10ish characters. So, that’s a problem for storing the 13 digit ISBN. Use BIGINT. **(DONE)**
* Also, there are two ISBNs for books. So, need to add an additional ISBN -13 field.
* Make ISBN Not Null in ShelvesRecords as well, since there is no way to display book details if the user doesn’t provide ISBN. All we will be able to see is the BookID, seller info and price but no details of the book. **(DONE)**
* **TASK**: prepare further queries. Assignable to anyone else if interested. Just give them the description of queries that will be needed and they can write it for you.

**12 November 2016 (6hrs)**

* Added script to setup the connection to remote database, handler to basic query requests, and displaying of retrieved results in the web page.
* Changed datatype of ISBN in BookRecords and ShelvesRecords to BIGINT.
* Also made ISBN in ShelvesRecords NOT NULL.
* Added state NOT NULL column to CustomerRecords
* **TASK**: Need to develop a primitive front end layout and handle basic requests to the database to see how well it goes. (done partially with retrieval of customerRecords)
* **ISSUE**: The users page right now doesn’t retrieve updated results from the database until the server is restarted.

**13 November 2016 (2hrs)**

* Tried using module.exports to export the db connection, but didn’t work due to unpredictable asynchronous function executions. Had to revert to previous commit.

**17 November 2016 (4hrs)**

* Got help from sujil in setting up a global database connection. It can make requests now.
* Need to learn how to properly implement callback functions to wait updating displays until data is received.
* Implement those callbacks in index.js and/or dbRequester.js

**19 November 2016 (30mins)**

* Implemented call back function successfully. Basically this is how it works:
* DB function

Function RequestMaker (callback) {

Make requests to database and put it into ‘data’ object

Callback(data); //This is like a function that has to be later overloaded by the caller

}

* Caller

RequestMaker(function (data) {

Console.log(data);

});