**Name:** Adrian Raszkiewicz

**Student ID:** 7128671

# **Module:** 340CT Software Quality and Process Management

**PDF Reflective Report:** Developing the Modern Web

*04/12/2018*

## Functionality

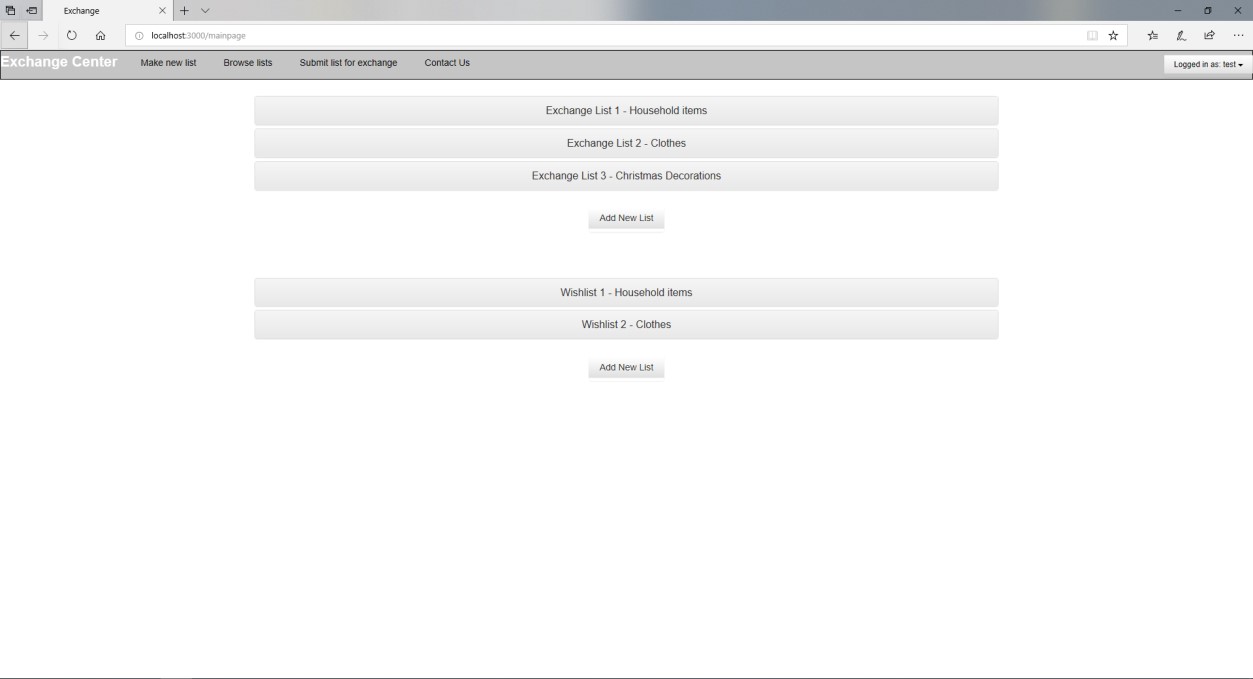
Requirements for Exchange project:

* Users can login
* Users can post list of items they want to exchange
* Users can post Wishlist
* Swap will send email to user

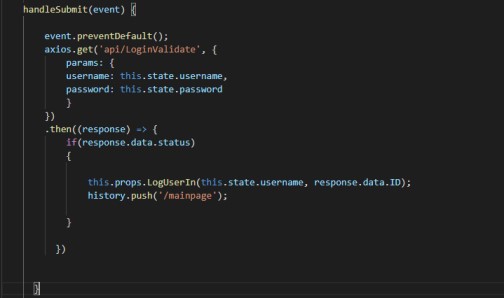
## Functionality Implemented

Successful

login



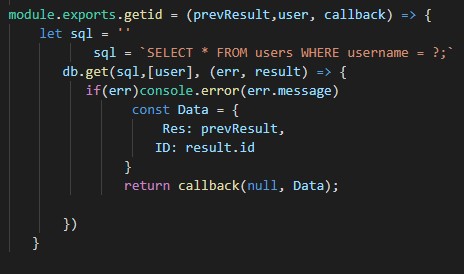
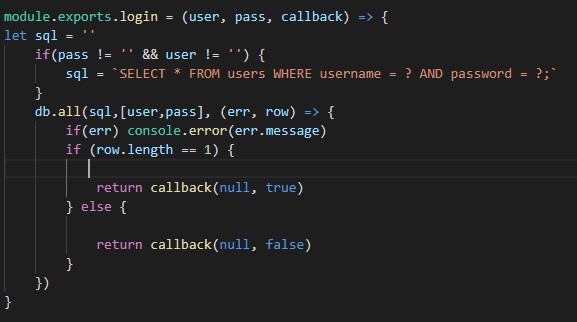
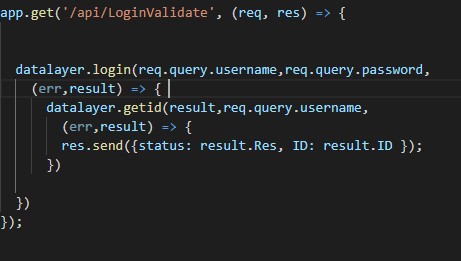
**Code**



Handle

submits

API



DB functions

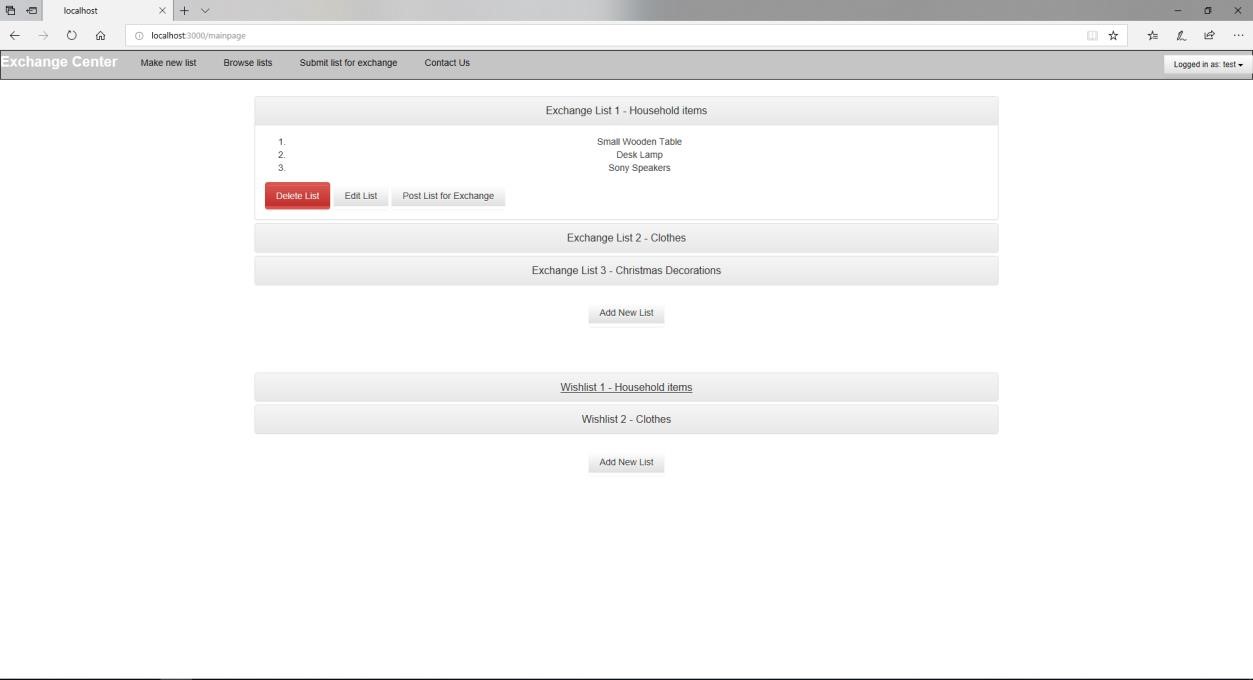
### 2. Users post list of items for exchange

Logged

in,

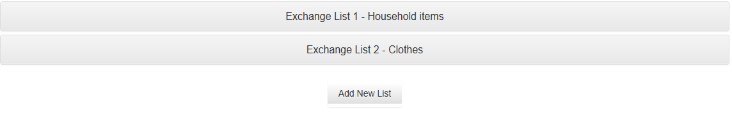
client can post

lists.

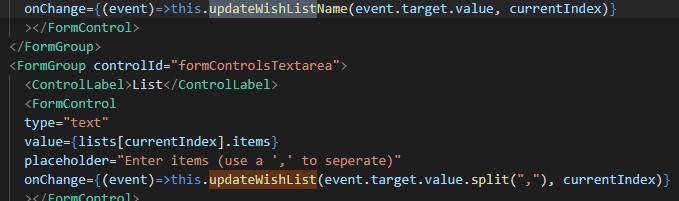
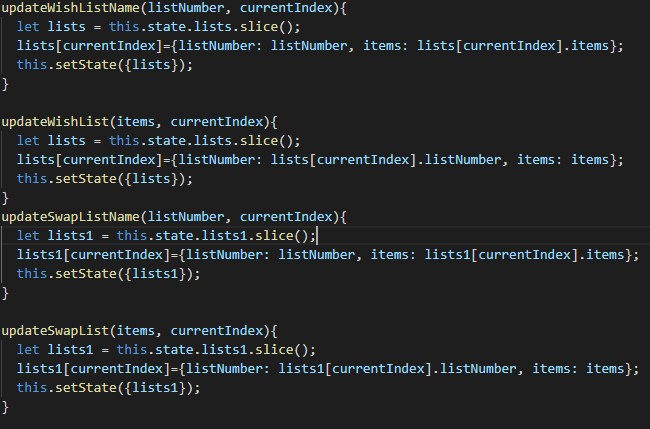
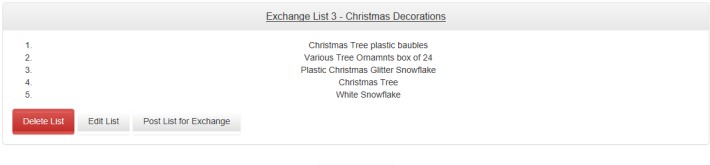
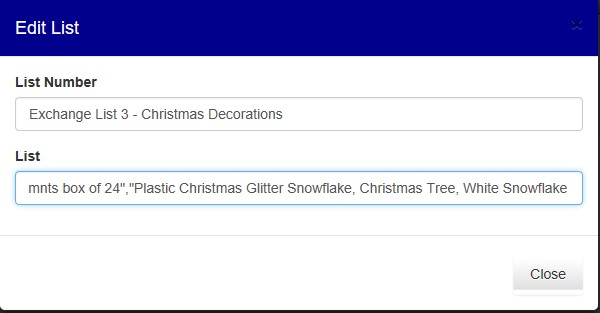


### Delete

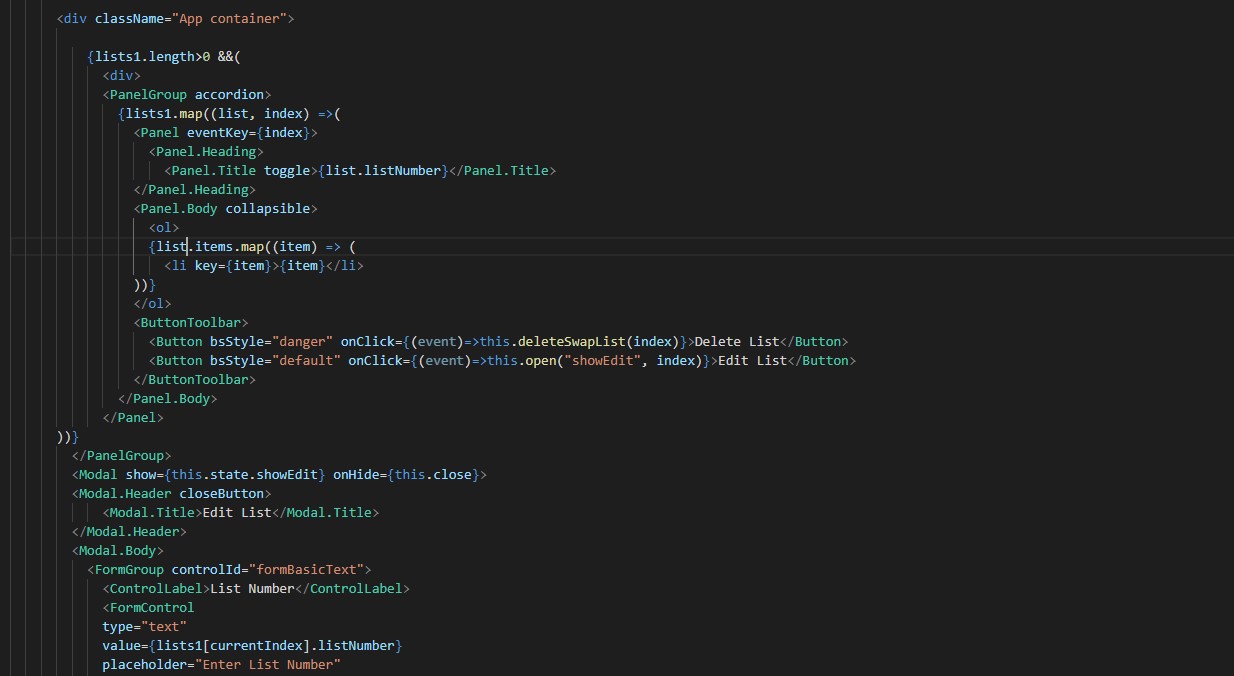
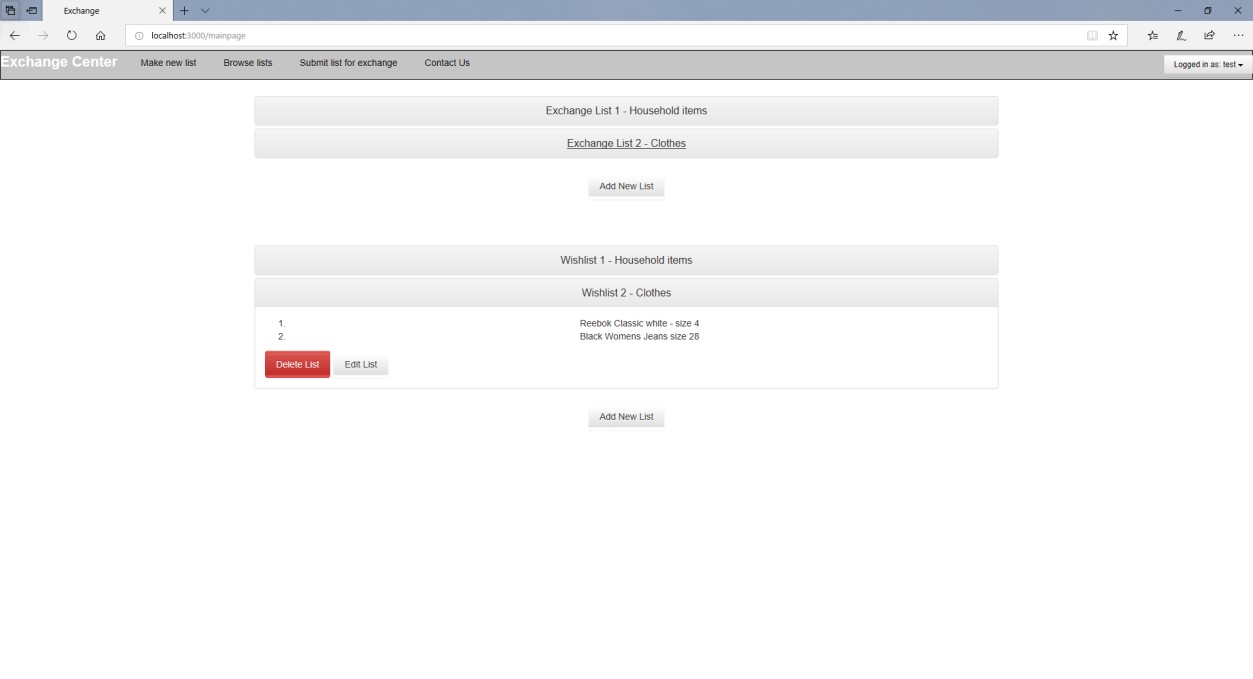
Successful delete



Edit



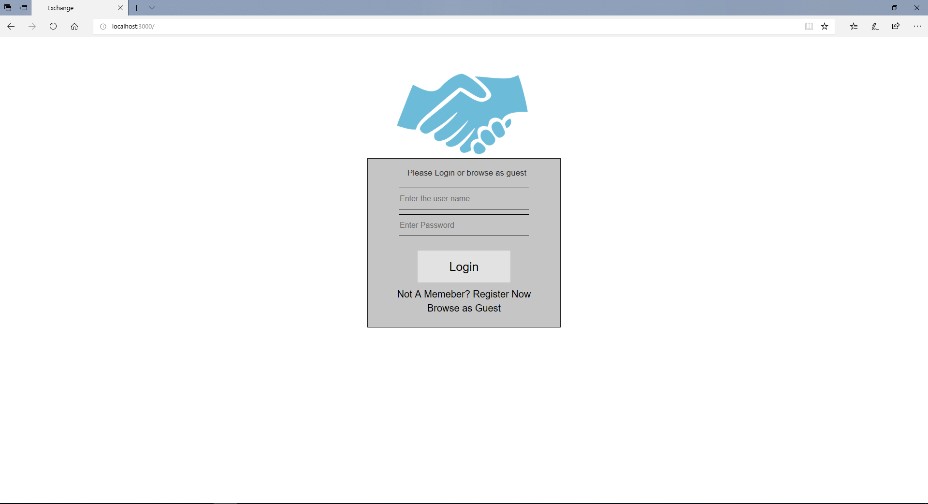
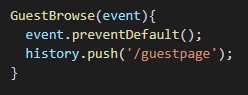
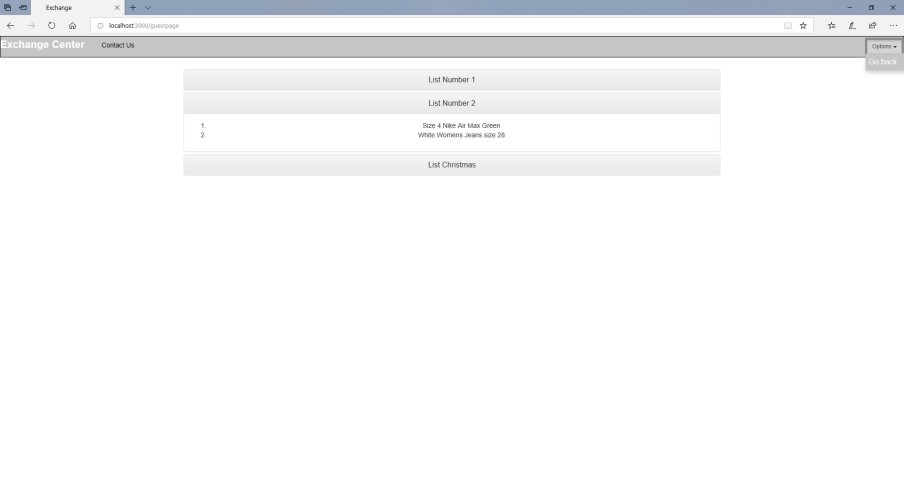
### Add



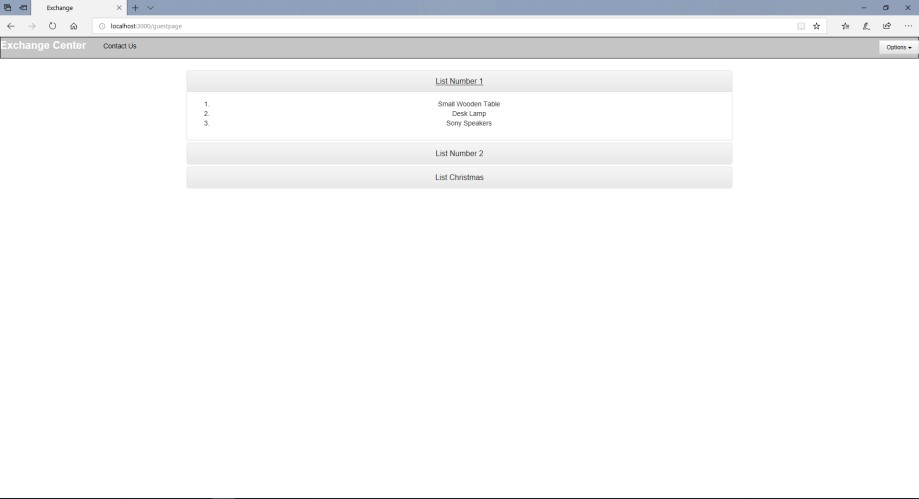
**4.**

**Visitors can**

**browse/search for items**



**5. Visitors can only suggest swaps after logging in**



GUEST

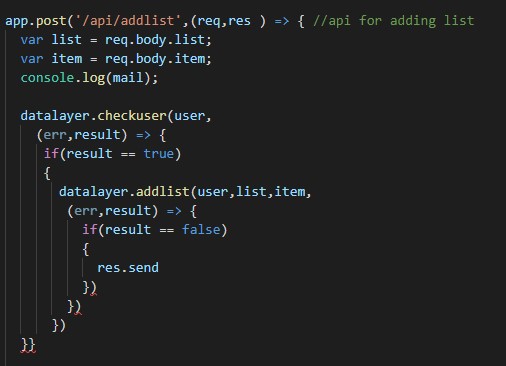
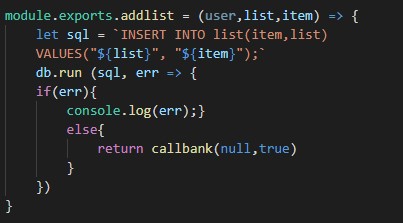
|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | User | |

## Functionality to be implemented

### 1. Improve: Posts Lists

I would add a clean connection to the database, everything was stored locally. Decided to create a feature branch for this and kept working to improving this functionality from there.

Attempt:



### 2. Search function

Unfortunately, I didn’t have the time to implement this and decided to work on more important features instead. I would eventually add a search box on the main page that would search through all the lists of different users Attempt:

**3. Swap - send email**

Not enough time

## Process

### Process I followed

The Incremental Model: The practice of repeatedly delivering the system into a production ready state, in series of expanding functionality. Instead of solving the whole problem at once, I broken down my task into a series of small objectives.

**Step1:** Information Gathering

At the very beginning of this project I tried to gather the most information about the system I will be making. So I had a clear understanding what features I need to include on the application, the **goals** I’m trying to accomplish and understand the **target audience**.

**Step2:** Planning: Site sketch & UI planning

Here I have planned the site sketch as well as the UI. This is to help me understand the different components my website will have and their connectivity.

**Exchange Lists**

**Wish Lists**

Logo and buttons

Log

info

**Step3:** Architecture Style

My architecture style is the Mediator Event Driven Architecture, everything is event driven from the clients point of view.

#### **Step4:** Coding

Firstly I’ve set-up my working environment, and installed all the necessary packages

Created frontend and backend directories, and all components inside (src, images etc.)

First coding was done on frontend, made the App.js file which is used for routing, after this coded all my components for the different pages.

Then I moved onto the backend part as I know I would struggle with this more, firstly i created a few tests for my functionality, after this I created my database and the DataLayer file which handles insertion and selection. Once this was done I made an index.js files to contain all my API’s.

At the end I made all my css files to style my work, decided to leave till end as functionality is important not the design.

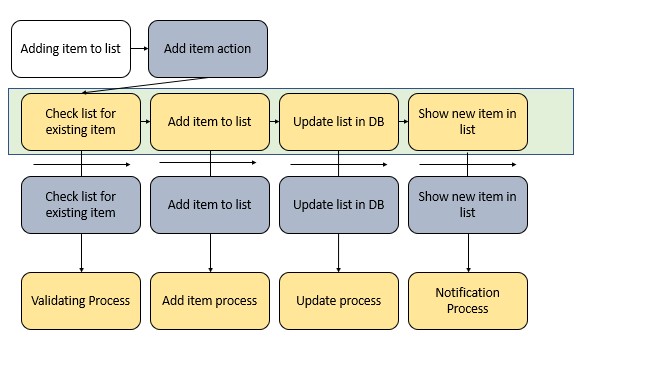
**Step5:** Testing and Reviewing

Here I test the website to see if each component is working and connecting as intended.

I do manual testing by loading the website up and I also am writing unit tests to check individual parts of my code.

Once I encounter an error or an element behaving not as it should be, I repair this and retest.

I have also used the **Event-Driven Design** approach for my work, this is an example of the though process behind this:



This is the mediator diagram for the process of adding a new item to an Exchange List.

1. The mediator executes the validation process which checks if the item already exists, if it does it will notify the user.
2. If first process passes, second process will be executed to add the item, afterwards the third will be executed which will update database.
3. Finally the user will be displayed with the added item.

### Issues faced

#### Database connection

I have had quite some trouble when linking my backend and frontend together. To be more precise the different API’s included in my system, they were not working properly alongside my data layer files which had all the queries related to the database. To solve this I had to do a lot of reasearch online on how to create a working link between them, I have then learned that all of my functions for selecting information from the database have to be called inside my API functions, this will create a link so my API can then be called inside the realted components.

#### React related errors

I have a lot of trouble setting up the React library to set up the User Interface.

Firstly I was getting ‘ERROR ELIFECYCLE’ which was preventing my from starting my application, I looked through the internet for suggestions, some of them were: reinstall node modules, delete package json lock, clean the cache by force.

I tried all of these but non worked, then I realized this is because the package.json file from the front end was conflicting with the package.json from the backend, so i fixed this problem by creating two separate directories for the front and back end.

### Areas for improvement

If I was to start this project all over again I would:

#### Unit Testing

I have wrote **three out of my seven** test only after I had the functionality they were testing working. I realize this is bad practice for code as it has proven very difficult to write tests for functions that already work and are implemented in the system. I would firstly create all my unit tests for the different components I have planned for, and than only after the tests are running smoothly I would create the actual function required for the passing of my test.

#### More detailed planning

I would have put more detail into planning my website with component diagrams and sitemap.

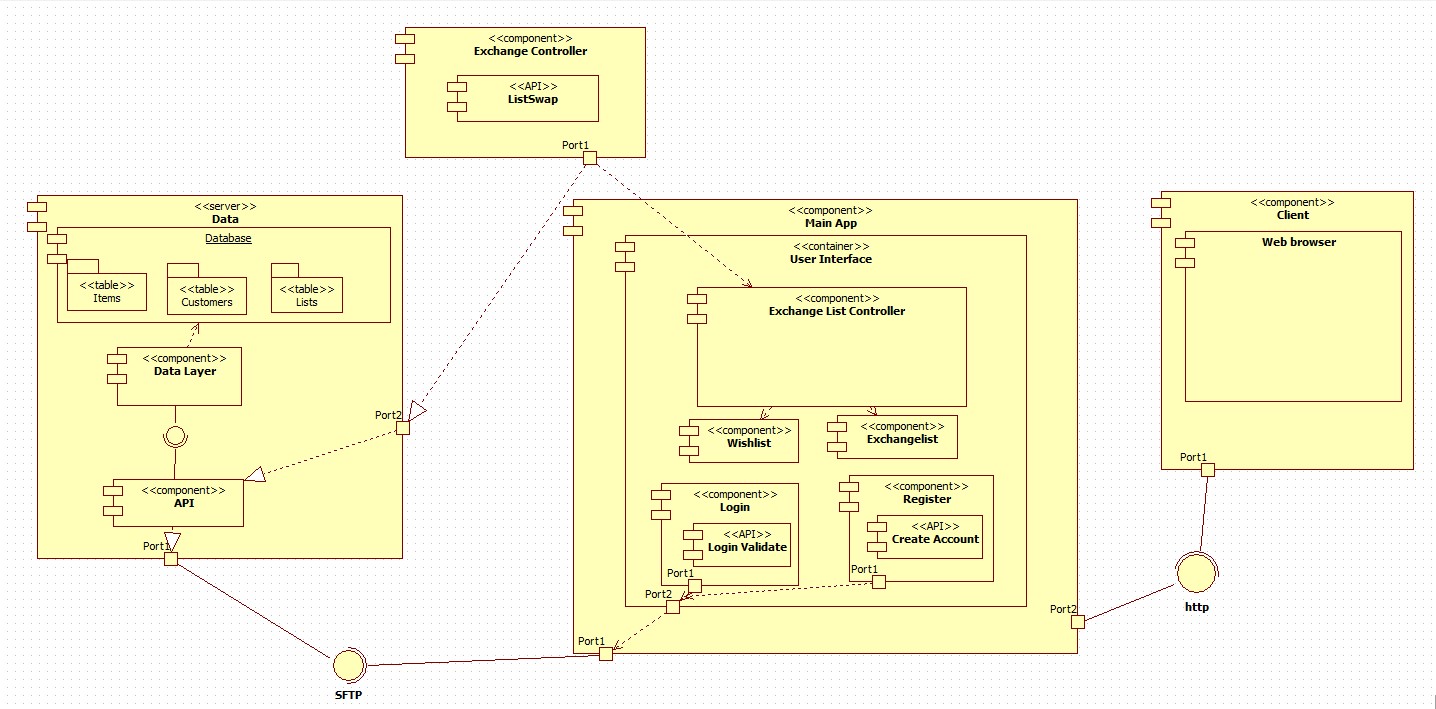
I believe that my planning has aided me a lot in the creation process as I had a understanding of how the website could be structured, however I still found myself kind of going along at some point once I have implemented the features I planned and realized that I will have to add a lot more different features.

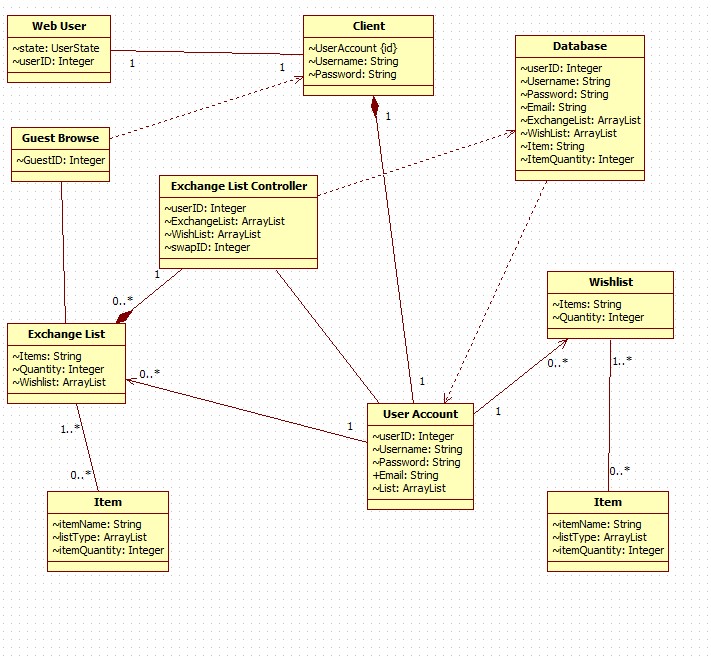
In the future i would has a main.css file that is responsible for all my container on the website so that the layout will be consistent.

## Architecture

### Component Diagrams

Two diagrams that reflect the components in my system.





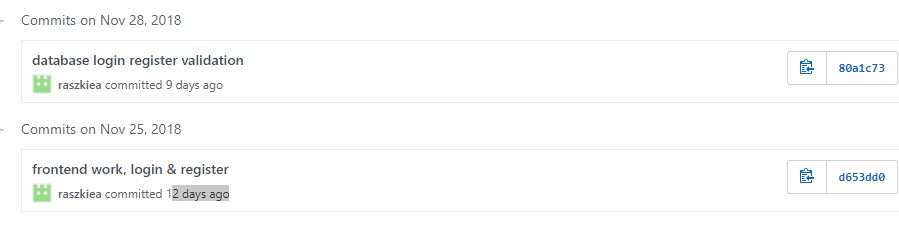
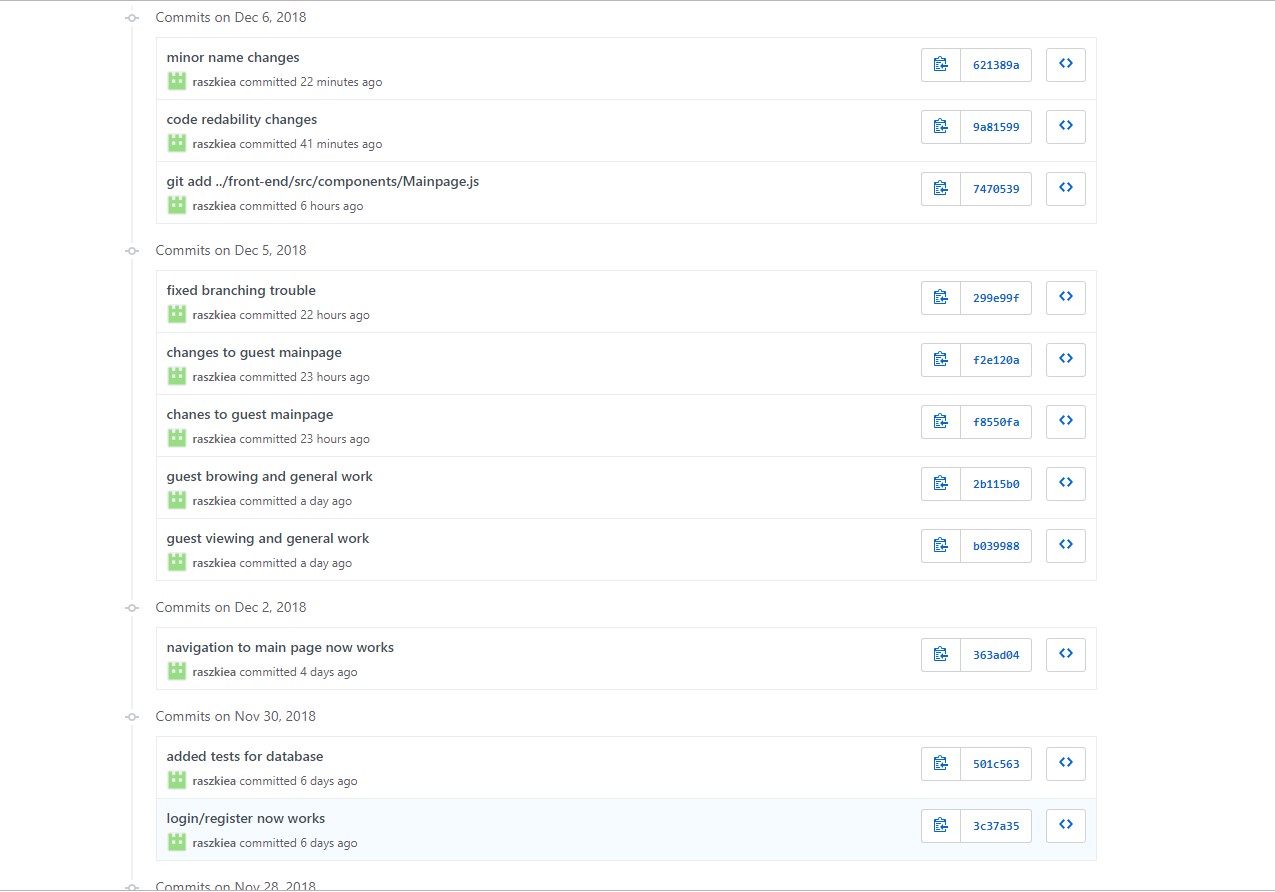
This reflects my system design. On my data component I have API for connectivity, a Data Layer file for database queries and an actual database.

I have the main app with all the corresponding components and API’s for usability, this is connected to the Exchange Controller which handles exchange, and stores result in database.

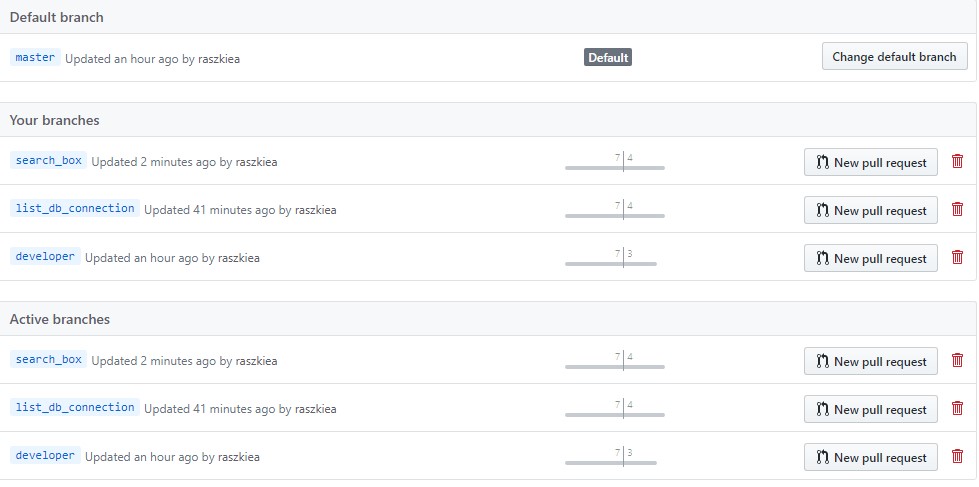
The final component is the Client Web which connect my client to the application. **Class**

## Version Control (GitHub)

Github link: <https://github.coventry.ac.uk/340CT-1819SEPJAN/raszkiea340CTT>Evidence of version control







Different branches used for adding additional features

I have branched from the ‘**master**’ and created a ‘**developer**’ branch.

For additional features I would again branch off from ‘developer’ and create a branch based on what I’m doing, which are ‘**feature branches’**.

**‘Master’**: Main branch reflects ‘production ready’ state.

**‘Developer’**: Branch before master, once code reaches a ‘production ready’ state, it is pushed to the master.

**Feature branches** (e.g.list\_db\_connection) used to develop new features for my application, exist as long as the feature is being developed, once the feature is finished, branch pushed back into ‘developer’.

**Release branches** are also used for preparation of product releases, used for last minute changes or bug fixes. I would use this feature given more time.

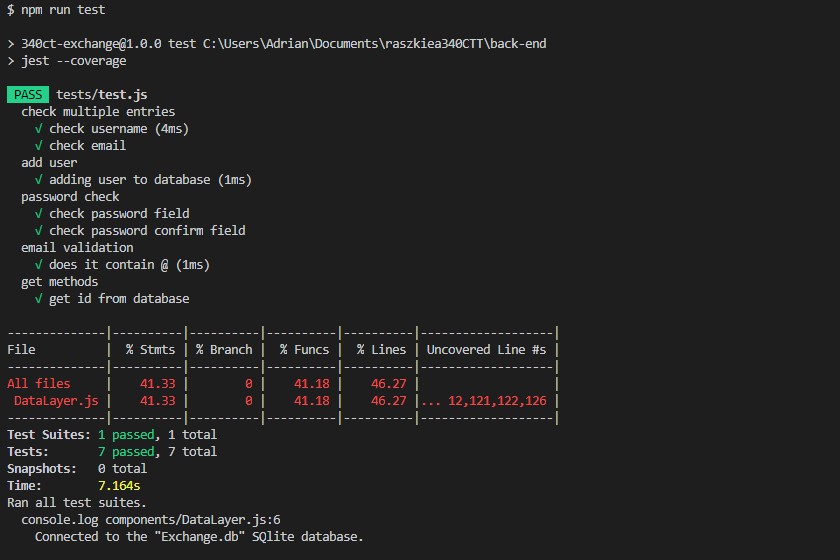
* **‘git commit -am’** and ‘*escape* **:wq**’ = all commits at once, separate comments
* **‘git branch -d branch\_name’** = Delete a branch
* **‘git status’** = check for modifications
* **‘git flow innit’** = Initializing git-flow by defining master/develop/feature branches• **‘git merge –no—off branch\_name’** = merging child brand into parent

•

## Test Driven Development







## Appendix References

Vincent Driessen. nvie.com (05/01/2010) A Successful Git branching model.[online] available at

[<**https://nvie.com/posts/a-successful-git-branching-model/>**](https://nvie.com/posts/a-successful-git-branching-model/)[04/12/2018]

Svetlana Gordiyenko. xbsoftware.com. (14/12/2015) Website Development Process: Full Guide in 7 setps. [online] available at < [**https://xbsoftware.com/blog/website-development-processfull-guide/>**](https://xbsoftware.com/blog/website-development-process-full-guide/)[04/12/2018]

Mark Tyers. cumoodle.coventry.ac.uk. (no date) Domain-Driven Design. [online] available at

[<**https://docs.google.com/presentation/d/15roChBN5xnttLZYg7Wdk7b26W5Q23nCYNycLj8G**](https://docs.google.com/presentation/d/15roChBN5xnttLZYg7Wdk7b26W5Q23nCYNycLj8GNy5w/edit#slide=id.g26933d8799_0_6)

[**Ny5w/edit#slide=id.g26933d8799\_0\_6**>](https://docs.google.com/presentation/d/15roChBN5xnttLZYg7Wdk7b26W5Q23nCYNycLj8GNy5w/edit#slide=id.g26933d8799_0_6) [04/12/2018]

Mark Tyers and Yih-Ling Hedley. cumoodle.coventry.ac.uk (no date) Software Architecture.

[online] available at <[**https://docs.google.com/presentation/d/1Ux83hzw-**](https://docs.google.com/presentation/d/1Ux83hzw-DdcBWn6PUXz_xF3b-ytE4PsWLpSKS_dqBfw/edit#slide=id.g26933d8799_0_6)

[**DdcBWn6PUXz\_xF3b-ytE4PsWLpSKS\_dqBfw/edit#slide=id.g26933d8799\_0\_6**>](https://docs.google.com/presentation/d/1Ux83hzw-DdcBWn6PUXz_xF3b-ytE4PsWLpSKS_dqBfw/edit#slide=id.g26933d8799_0_6) [04/12/2018]

Mark Tyers. cumoodle.coventry.ac.uk (no date) Software Quality 2. [online] available at [**https://docs.google.com/presentation/d/1takMoPNZTk4IyfqrEssDm9yFfkQQxuA16ZBoohUso80/e dit#slide=id.g26933d8799\_0\_6**>](https://docs.google.com/presentation/d/1takMoPNZTk4IyfqrEssDm9yFfkQQxuA16ZBoohUso80/edit#slide=id.g26933d8799_0_6) [04/12/2018]

## <https://nvie.com/posts/a-successful-git-branching-model/>https://xbsoftware.com/blog/website-development-process-full-guide/