Name: Adrian Raszkiewicz

Student ID: 7128671

Module: 340CT Software Quality and Process Management

PDF Reflective Report: Developing the Modern Web

04/12/2018

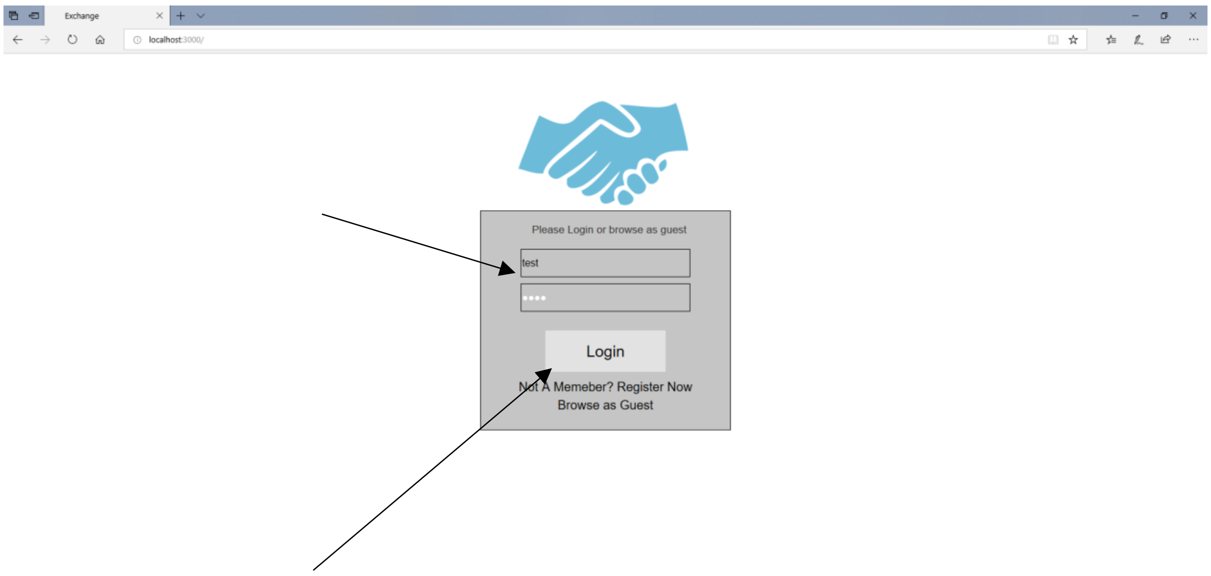
Functionality

Requirements for Exchange project:

- Users can login
- Users can post list of items they want to exchange
- Users can post Wishlist
- Visitors can browse for items
- Visitors suggest swap only after login
- Swap will send email to user

Functionality Implemented

1. Users can login



Please Login or browse as guest

test

••••

Login

Not A Memeber? Register Now

Browse as Guest

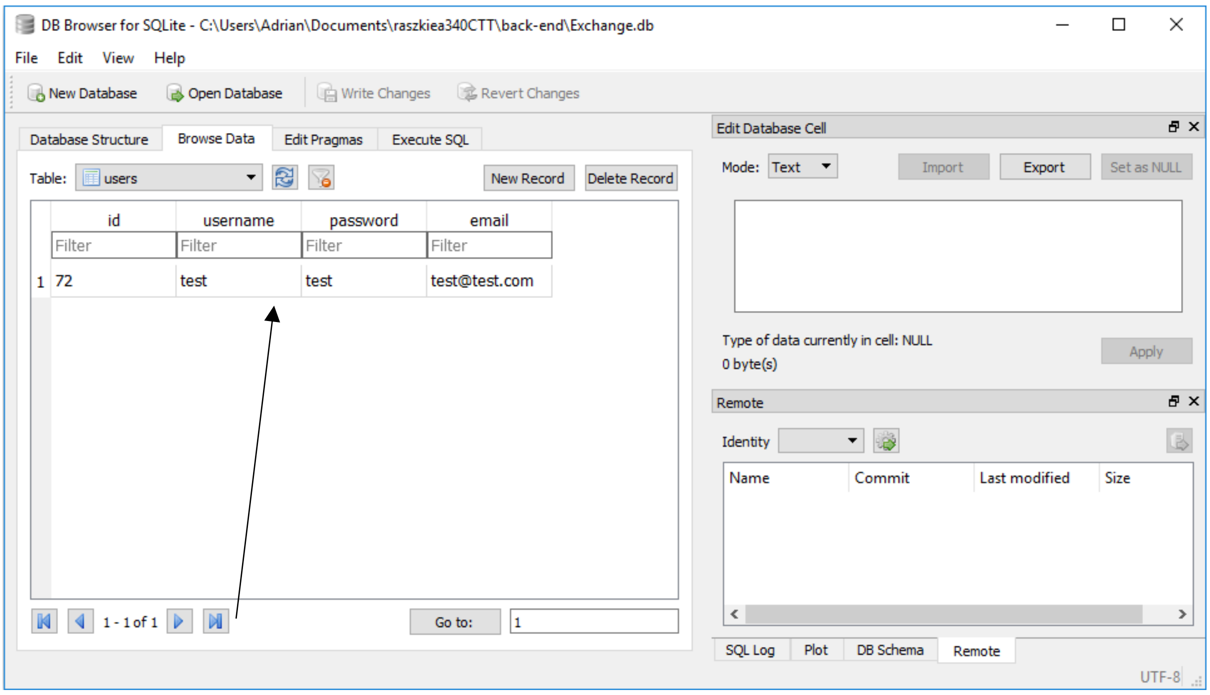


Table: users

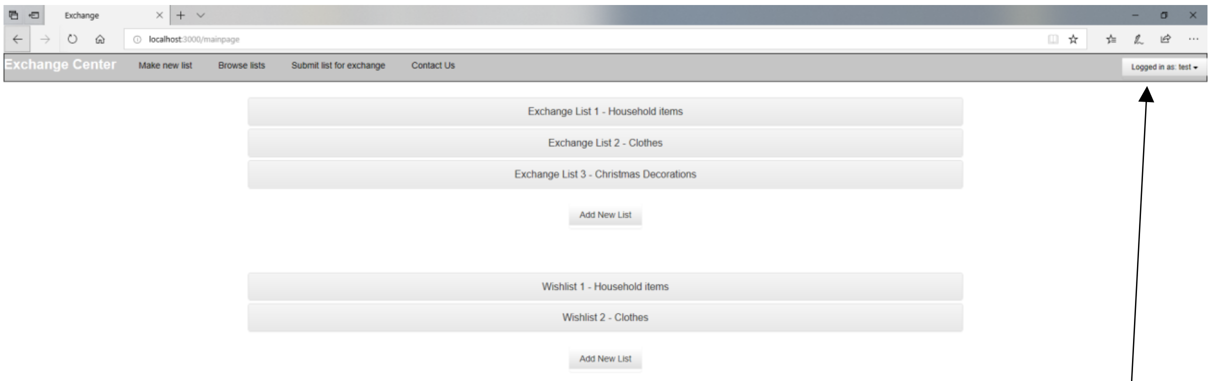
	id	username	password	email
	Filter	Filter	Filter	Filter
1	72	test	test	test@test.com

1 - 1 of 1 Go to: 1

Mode: Text

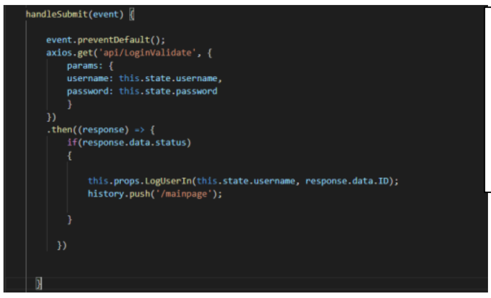
Type of data currently in cell: NULL
0 byte(s)

Name	Commit	Last modified	Size



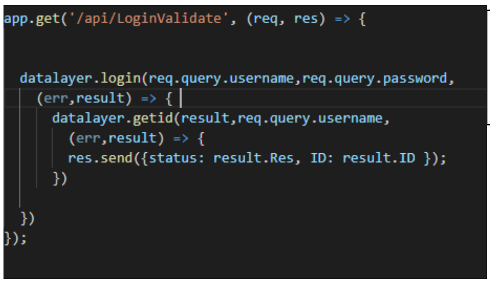
Successful login

<u>Code</u>



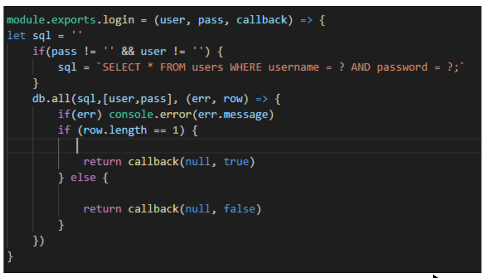
```
handleSubmit(event) {

    event.preventDefault();
    axios.get('api/LoginValidate', {
        params: {
        username: this.state.username,
        password: this.state.password
        }
    })
    .then((response) => {
        if(response.data.status)
        {

            this.props.LogUserIn(this.state.username, response.data.ID);
            history.push('/mainpage');

        }

     })

 }
```

Handle submits

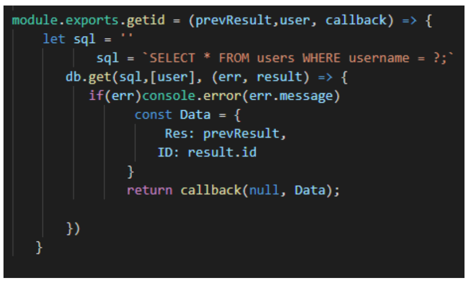


```
app.get('/api/LoginValidate', (req, res) => {


  datalayer.login(req.query.username,req.query.password,
    (err,result) => {
      datalayer.getid(result,req.query.username,
        (err,result) => {
        res.send({status: result.Res, ID: result.ID });
      })

  })
});
```

API



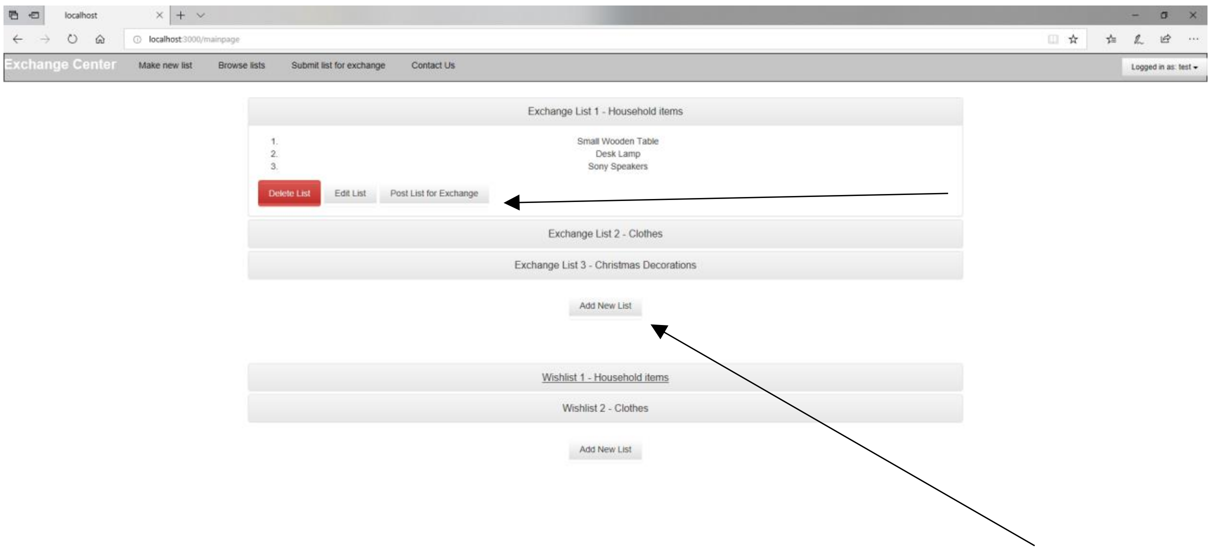
```
module.exports.login = (user, pass, callback) => {
let sql = ''
    if(pass != '' && user != '') {
        sql = `SELECT * FROM users WHERE username = ? AND password = ?;`
    }
    db.all(sql,[user,pass], (err, row) => {
        if(err) console.error(err.message)
        if (row.length == 1) {

            return callback(null, true)
        } else {

            return callback(null, false)
        }
    })
}
```



```
module.exports.getid = (prevResult,user, callback) => {
    let sql = ''
        sql = `SELECT * FROM users WHERE username = ?;`
    db.get(sql,[user], (err, result) => {
        if(err)console.error(err.message)
            const Data = {
                Res: prevResult,
                ID: result.id
            }
            return callback(null, Data);

    })
  }
```

DB functions

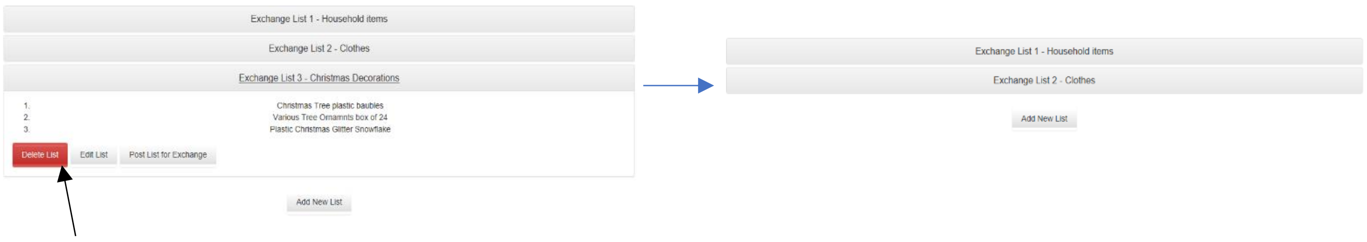
2. Users post list of items for exchange

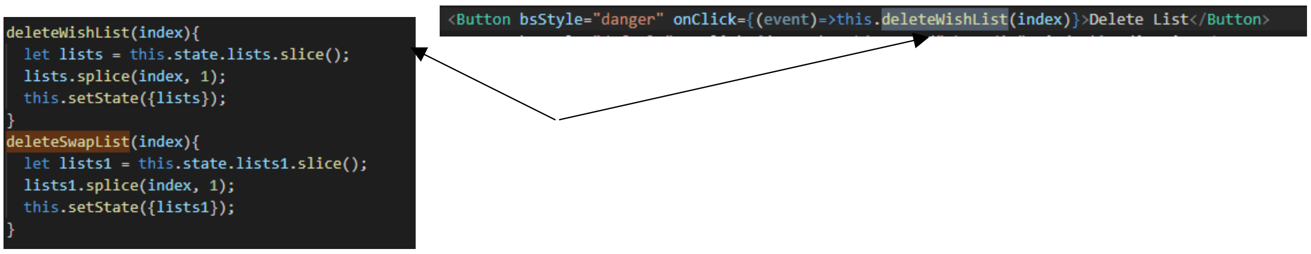


Logged in, client can post lists.

Delete

Successful delete

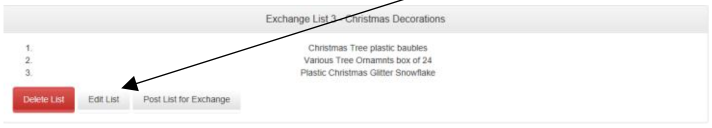

Exchange List 1 - Household items
Exchange List 2 - Clothes
Exchange List 3 - Christmas Decorations
1. Christmas Tree plastic baubles
2. Various Tree Ornamnts box of 24
3. Plastic Christmas Glitter Snowflake
Delete List
Edit List
Post List for Exchange
Add New List
Exchange List 1 - Household items
Exchange List 2 - Clothes
Add New List



```
deleteWishList(index){
  let lists = this.state.lists.slice();
  lists.splice(index, 1);
  this.setState({lists});
}
deleteSwapList(index){
  let lists1 = this.state.lists1.slice();
  lists1.splice(index, 1);
  this.setState({lists1});
}
```

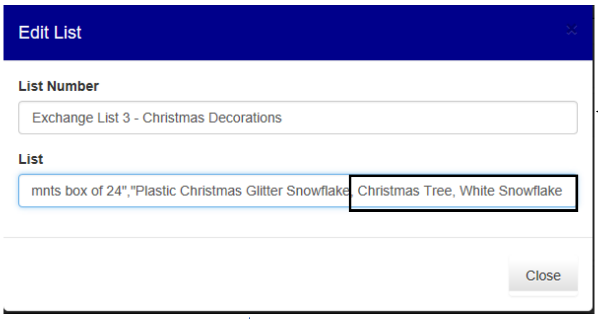
```
<Button bsStyle="danger" onClick={(event)=>this.deleteWishList(index)}>Delete List</Button>
```

Edit



Exchange List 3 - Christmas Decorations

1. Christmas Tree plastic baubles
2. Various Tree Ornamnts box of 24
3. Plastic Christmas Glitter Snowflake



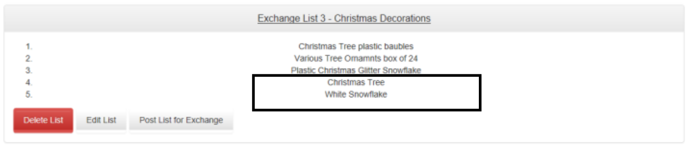
Edit List

List Number

Exchange List 3 - Christmas Decorations

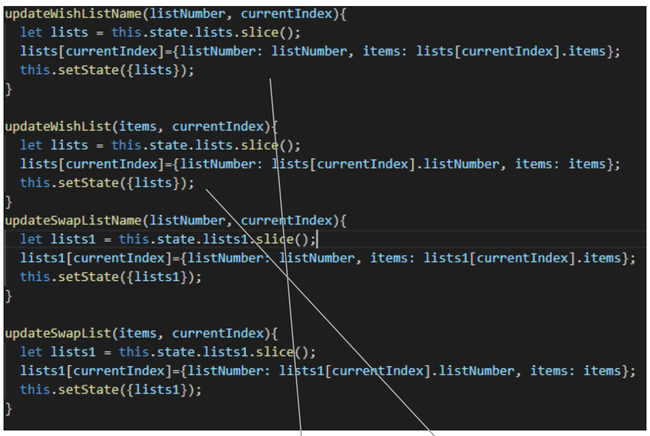
List

mnts box of 24","Plastic Christmas Glitter Snowflake, Christmas Tree, White Snowflake

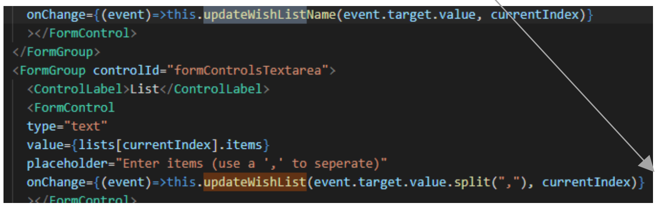


Exchange List 3 - Christmas Decorations

1. Christmas Tree plastic baubles
2. Various Tree Ornamnts box of 24
3. Plastic Christmas Glitter Snowflake
4. Christmas Tree
5. White Snowflake

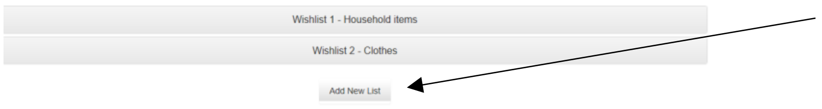


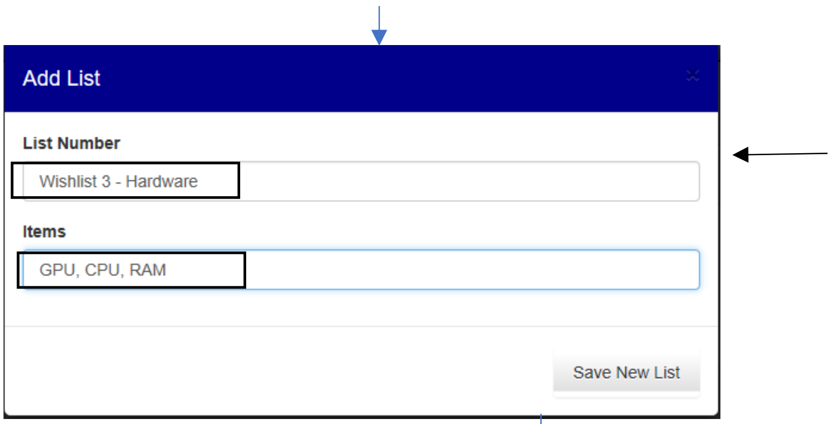
```
updateWishListName(listNumber, currentIndex){
  let lists = this.state.lists.slice();
  lists[currentIndex]={listNumber: listNumber, items: lists[currentIndex].items};
  this.setState({lists});
}

updateWishList(items, currentIndex){
  let lists = this.state.lists.slice();
  lists[currentIndex]={listNumber: lists[currentIndex].listNumber, items: items};
  this.setState({lists});
}
updateSwapListName(listNumber, currentIndex){
  let lists1 = this.state.lists1.slice();
  lists1[currentIndex]={listNumber: listNumber, items: lists1[currentIndex].items};
  this.setState({lists1});
}

updateSwapList(items, currentIndex){
  let lists1 = this.state.lists1.slice();
  lists1[currentIndex]={listNumber: lists1[currentIndex].listNumber, items: items};
  this.setState({lists1});
}
```



```
   onChange={(event)=>this.updateWishListName(event.target.value, currentIndex)}
   ></FormControl>
 </FormGroup>
 <FormGroup controlId="formControlsTextarea">
   <ControlLabel>List</ControlLabel>
   <FormControl
   type="text"
   value={lists[currentIndex].items}
   placeholder="Enter items (use a ',' to seperate)"
   onChange={(event)=>this.updateWishList(event.target.value.split(","), currentIndex)}
```

Add



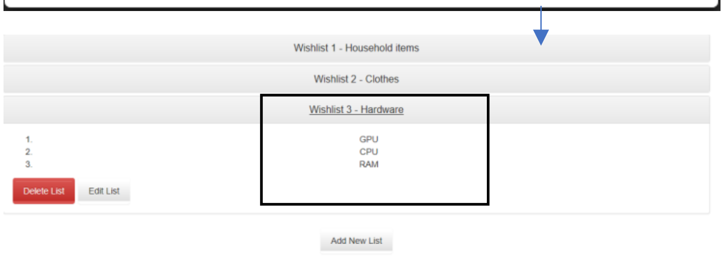


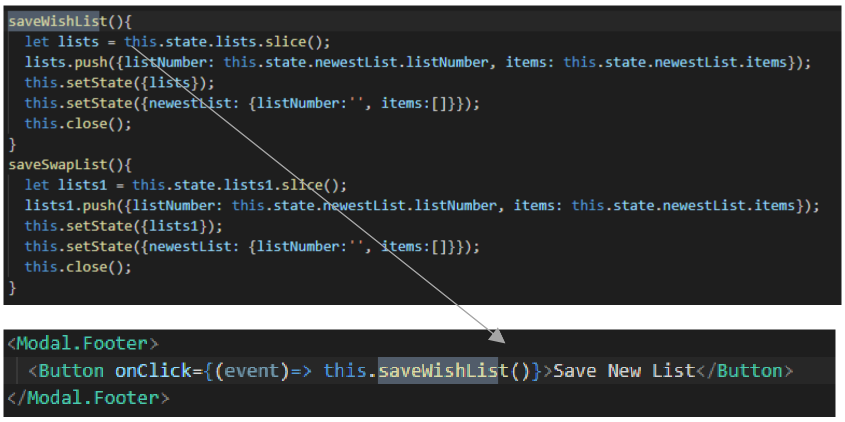
List Number

Wishlist 3 - Hardware

Items

GPU, CPU, RAM

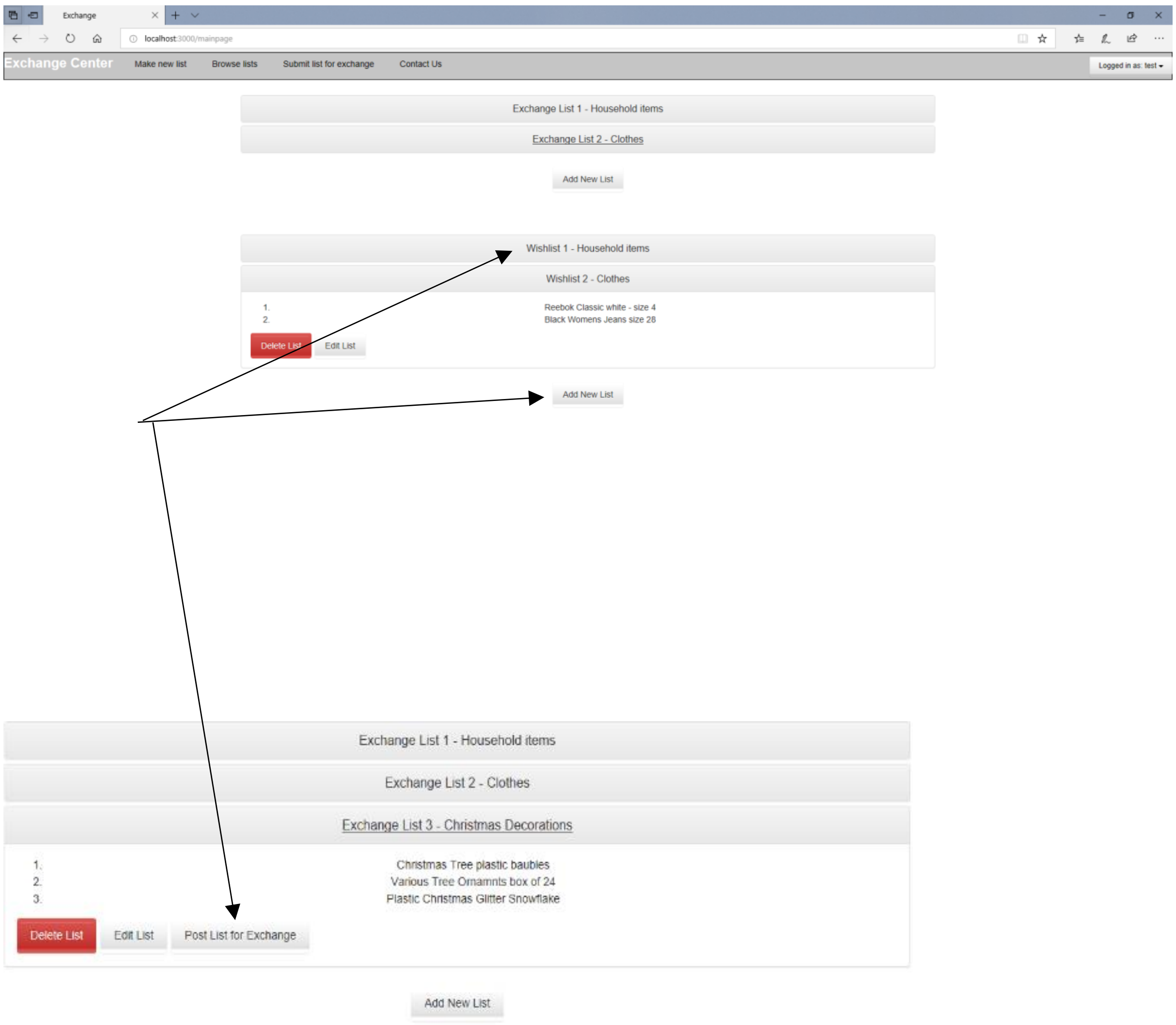




```
saveWishList(){
  let lists = this.state.lists.slice();
  lists.push({listNumber: this.state.newestList.listNumber, items: this.state.newestList.items});
  this.setState({lists});
  this.setState({newestList: {listNumber:'', items:[]}});
  this.close();
}
saveSwapList(){
  let lists1 = this.state.lists1.slice();
  lists1.push({listNumber: this.state.newestList.listNumber, items: this.state.newestList.items});
  this.setState({lists1});
  this.setState({newestList: {listNumber:'', items:[]}});
  this.close();
}
```

```
<Modal.Footer>
  <Button onClick={(event)=> this.saveWishList()}>Save New List</Button>
</Modal.Footer>
```

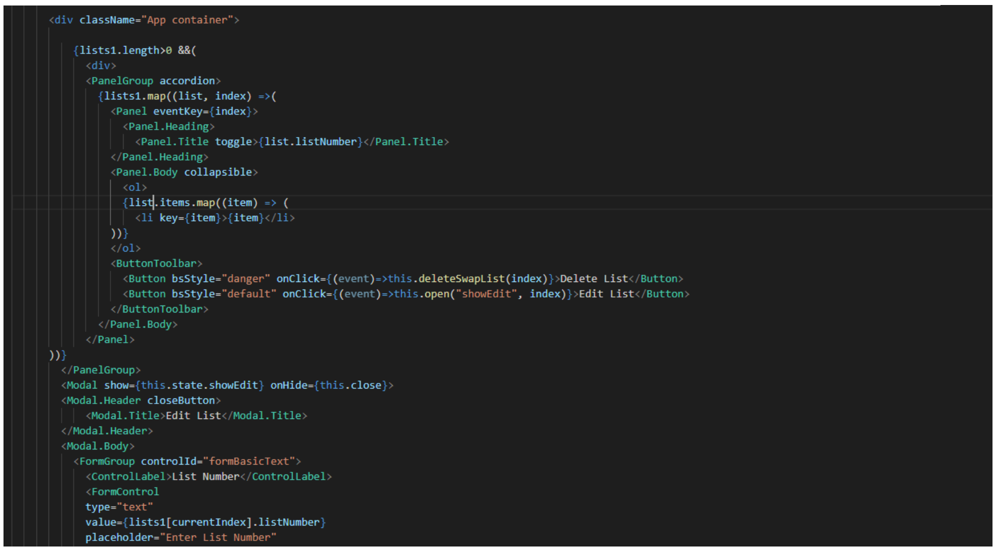
3. Users can post their SwapList



Exchange List 1 - Household items
Exchange List 2 - Clothes
Add New List

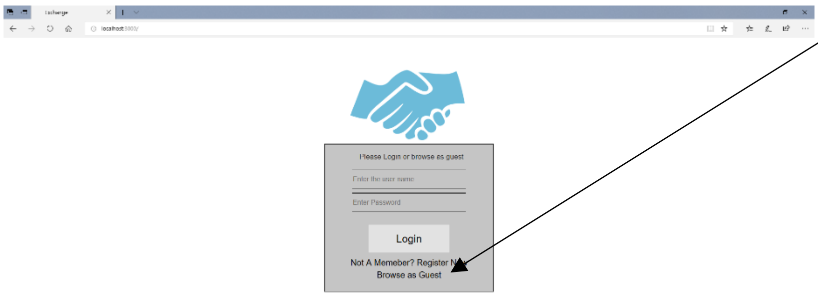
Wishlist 1 - Household items
Wishlist 2 - Clothes
1. Reebok Classic white - size 4
2. Black Womens Jeans size 28
Delete List   Edit List
Add New List

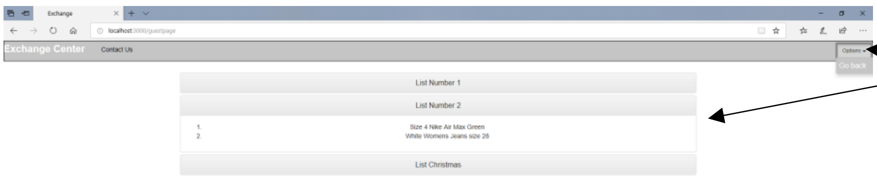
Exchange List 1 - Household items
Exchange List 2 - Clothes
Exchange List 3 - Christmas Decorations
1. Christmas Tree plastic baubles
2. Various Tree Ornamnts box of 24
3. Plastic Christmas Glitter Snowflake
Delete List   Edit List   Post List for Exchange
Add New List



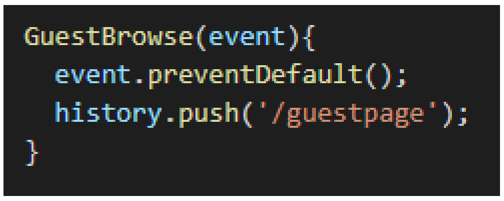
```
<div className="App container">

    {lists1.length>0 &&(
      <div>
      <PanelGroup accordion>
        {lists1.map((list, index) =>(
          <Panel eventKey={index}>
            <Panel.Heading>
              <Panel.Title toggle>{list.listNumber}</Panel.Title>
          </Panel.Heading>
          <Panel.Body collapsible>
            <ol>
            {list.items.map((item) => (
              <li key={item}>{item}</li>
          ))}
          </ol>
          <ButtonToolbar>
            <Button bsStyle="danger" onClick={(event)=>this.deleteSwapList(index)}>Delete List</Button>
            <Button bsStyle="default" onClick={(event)=>this.open("showEdit", index)}>Edit List</Button>
          </ButtonToolbar>
        </Panel.Body>
      </Panel>
))}
  </PanelGroup>
  <Modal show={this.state.showEdit} onHide={this.close}>
  <Modal.Header closeButton>
      <Modal.Title>Edit List</Modal.Title>
  </Modal.Header>
  <Modal.Body>
    <FormGroup controlId="formBasicText">
      <ControlLabel>List Number</ControlLabel>
      <FormControl
      type="text"
      value={lists1[currentIndex].listNumber}
      placeholder="Enter List Number"
```

4. Visitors can browse/search for items

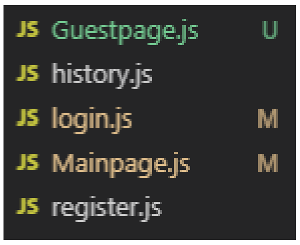


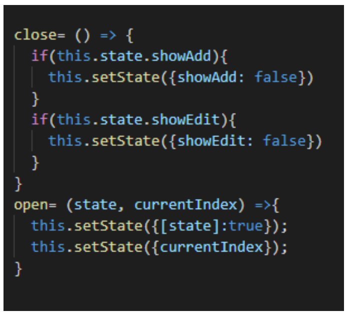


1. Size 4 Nike Air Max Green
2. White Womens Jeans size 28



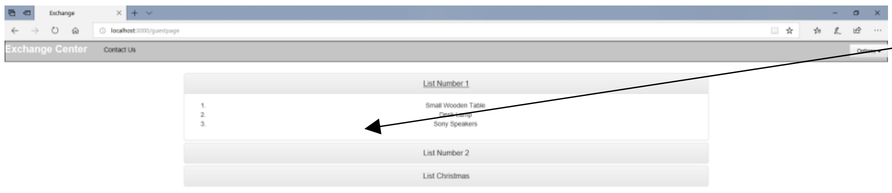
```
GuestBrowse(event){
  event.preventDefault();
  history.push('/guestpage');
}
```





```
close= () => {
  if(this.state.showAdd){
    this.setState({showAdd: false})
  }
  if(this.state.showEdit){
    this.setState({showEdit: false})
  }
}
open= (state, currentIndex) =>{
  this.setState({[state]:true});
  this.setState({currentIndex});
}
```

5. Visitors can only suggest swaps after logging in



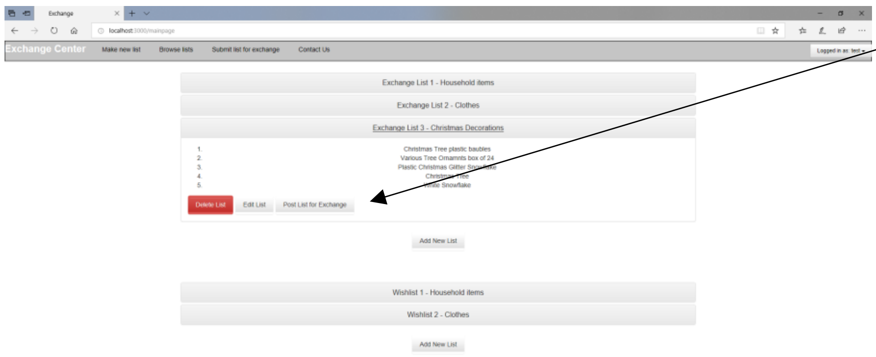
List Number 1

1. Small Wooden Table
2. Desk Lamp
3. Sony Speakers

List Number 2

List Christmas

GUEST



Exchange List 1 - Household items

Exchange List 2 - Clothes

Exchange List 3 - Christmas Decorations

1. Christmas Tree plastic baubles
2. Various Tree Ornamnts box of 24
3. Plastic Christmas Glitter Snowflake
4. Christmas Tree
5. White Snowflake

Wishlist 1 - Household items

Wishlist 2 - Clothes

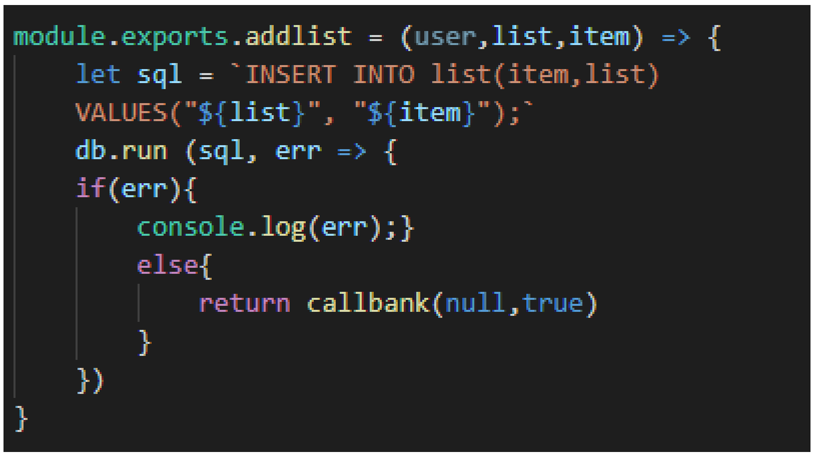
User

<u>Functionality to be implemented</u>

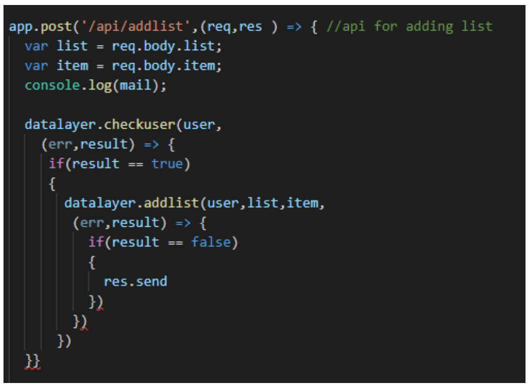
1. Improve: Posts Lists

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:



```
module.exports.addlist = (user,list,item) => {
    let sql = `INSERT INTO list(item,list)
    VALUES("${list}", "${item}");`
    db.run (sql, err => {
    if(err){
        console.log(err);}
        else{
            return callbank(null,true)
        }
    })
}
```

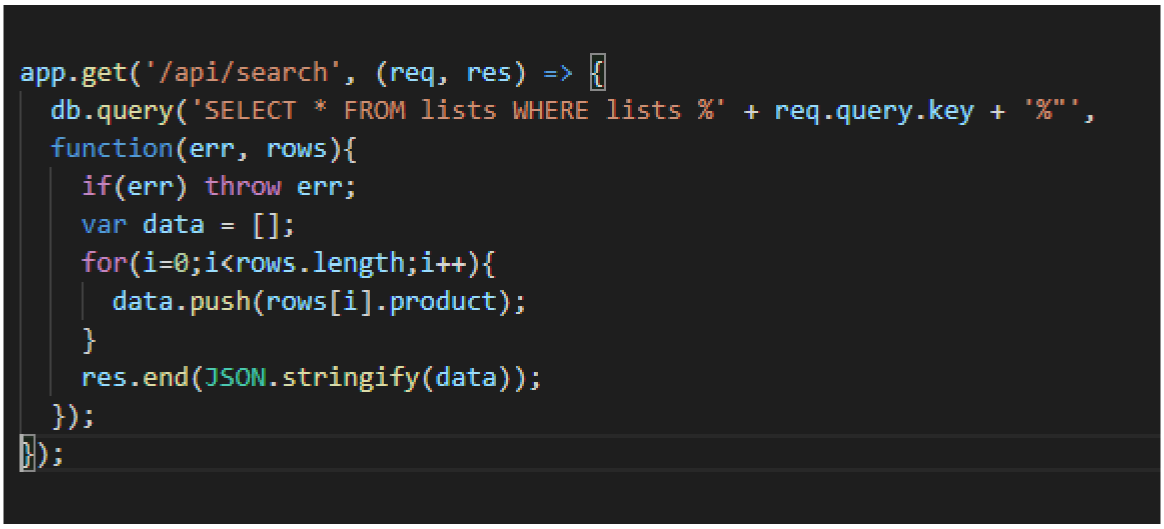


```
app.post('/api/addlist',(req,res ) => { //api for adding list
  var list = req.body.list;
  var item = req.body.item;
  console.log(mail);

  datalayer.checkuser(user,
    (err,result) => {
     if(result == true)
     {
       datalayer.addlist(user,list,item,
        (err,result) => {
         if(result == false)
         {
           res.send
         })
       })
     })
  }}
```

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:



```
app.get('/api/search', (req, res) => {
  db.query('SELECT * FROM lists WHERE lists %' + req.query.key + '%"',
  function(err, rows){
    if(err) throw err;
    var data = [];
    for(i=0;i<rows.length;i++){
      data.push(rows[i].product);
    }
    res.end(JSON.stringify(data));
  });
});
```

3. Swap - send email

Not enough time

Process

Process I followed

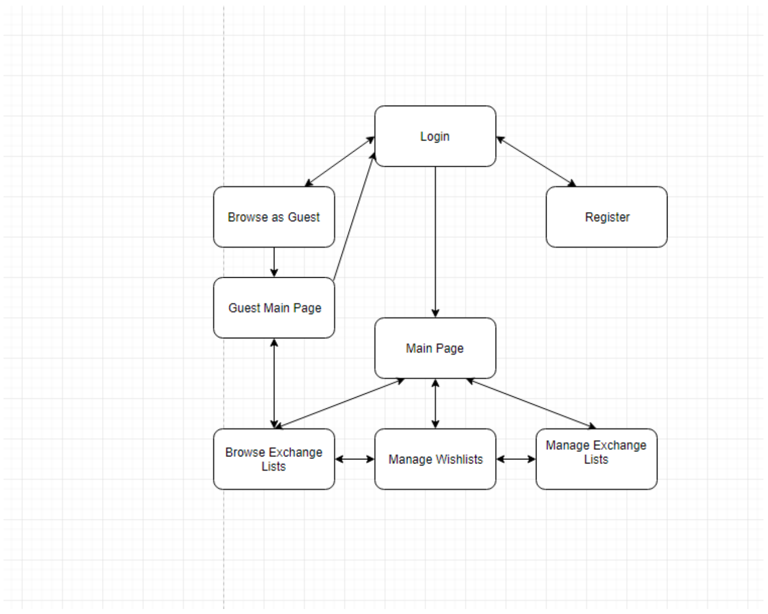
The Incremental Model: The practice of repeatedly delivering the system into a production ready state, in series of expanding functionality. Instead of solving the whole problem at once, I broken down my task into a series of small objectives.

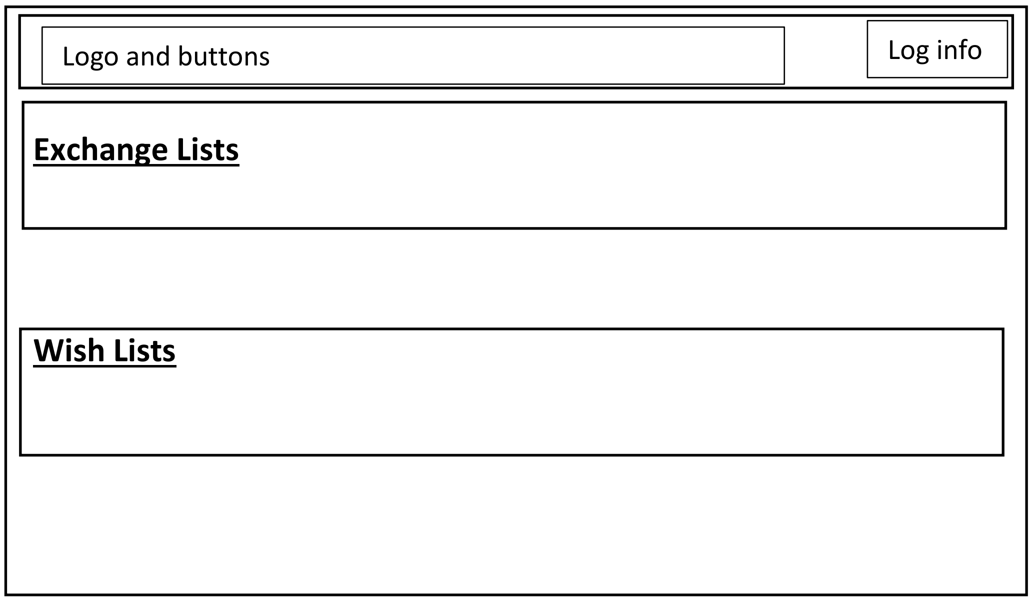
Step1: Information Gathering

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.





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.

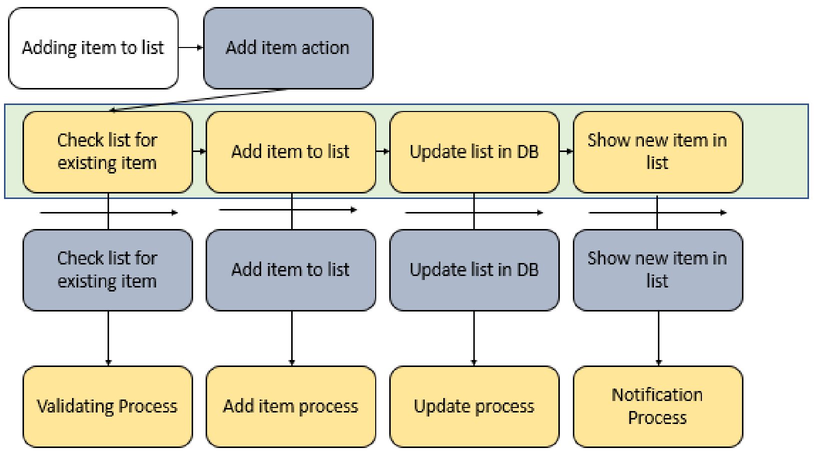
Step5: Testing and Reviewing

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.

<u>Issues faced</u>

<u>Database connection</u>

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.

<u>React related errors</u>

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.

<u>Areas for improvement</u>

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

<u>Unit Testing</u>

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. This method will help me get more code coverage and just generally make my life easier.

Complete code encapsulation

I have tried to make my work as tidy and readable as possible, however i realize that a lot of the different components I have put on one page and ended up with a few files that are over 200 lines long. Next time I would try to encapsulate them by having separate files for all my components, after that I would import them onto the required page. This would make everything far neater and more readable.

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.

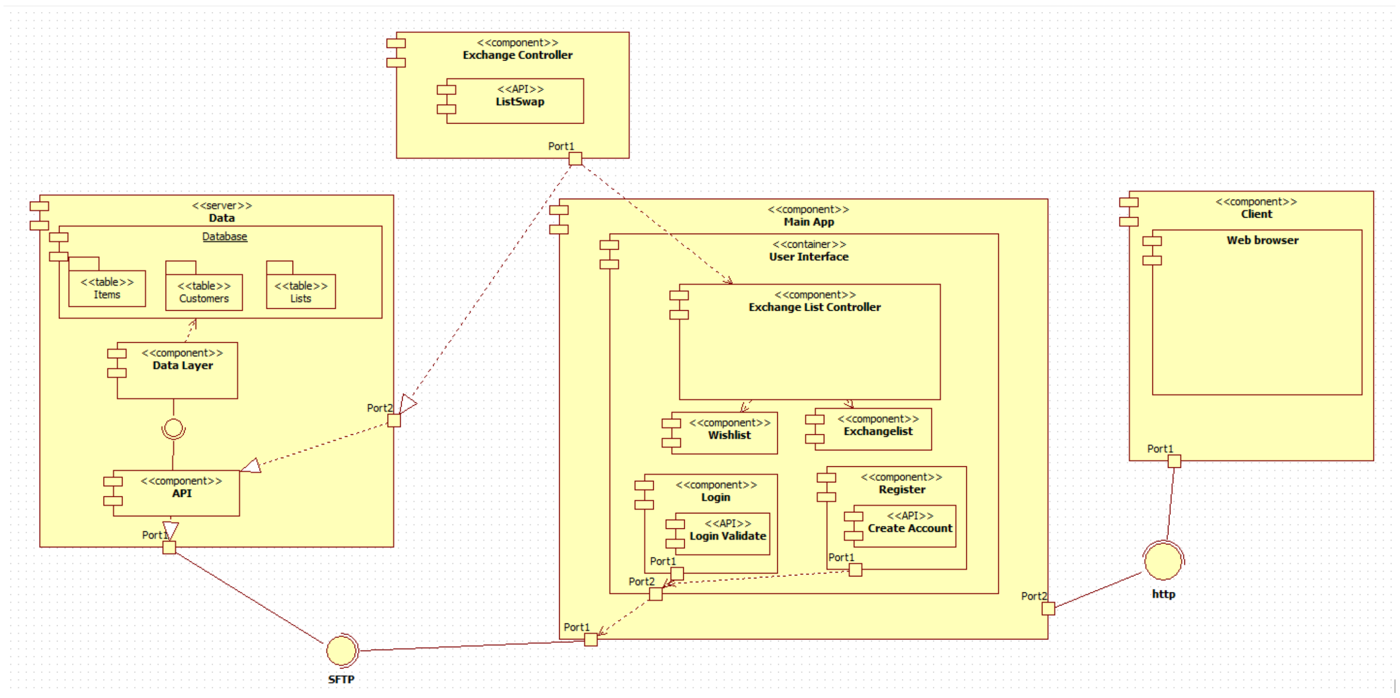
I would also approach the styling of my website with my css differently, I have different css files for my pages, I have tried to make them identical however i can still pinpoint some slight changes in layout from one page to another.

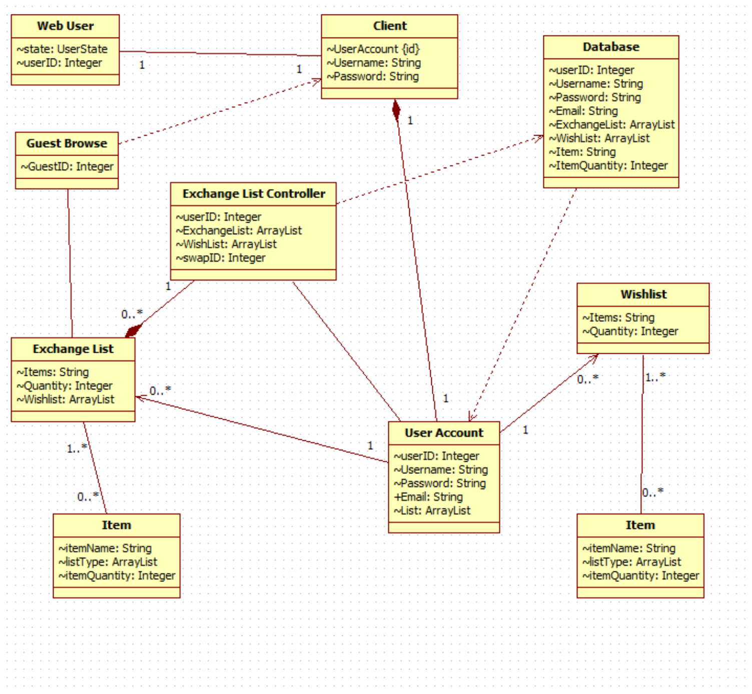
In the future i would has a main.css file that is responsible for all my container on the website so that the layout will be consistent.

Architecture

Component Diagrams

Two diagrams that reflect the components in my system.





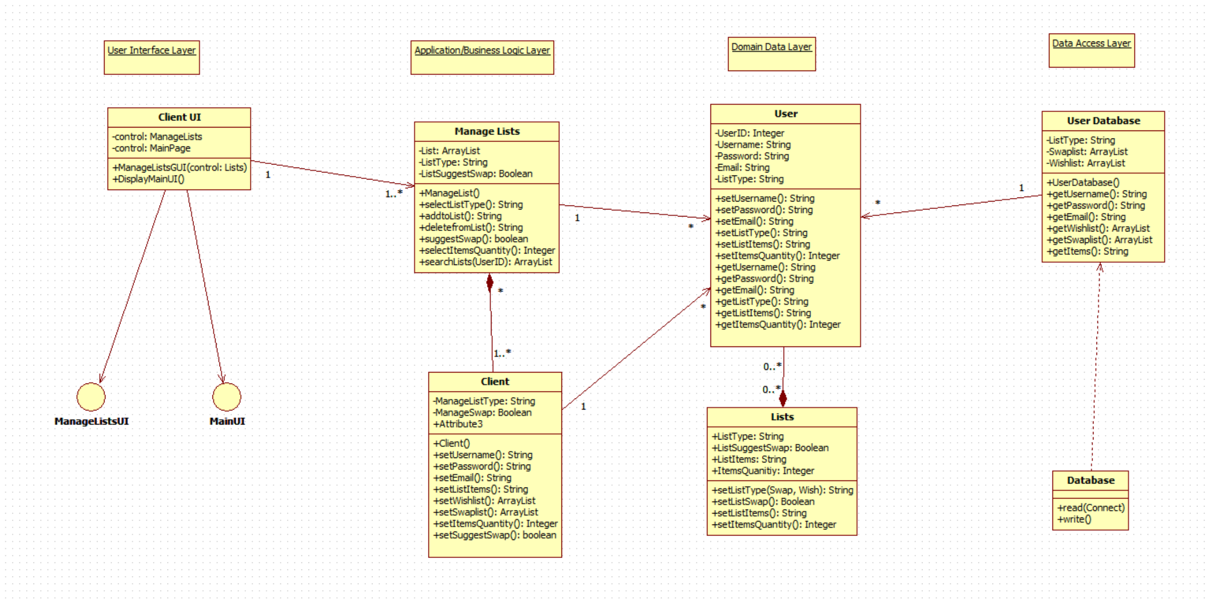
This reflects my system design. On my data component I have API for connectivity, a Data Layer file for database queries and an actual database.

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 Diagram

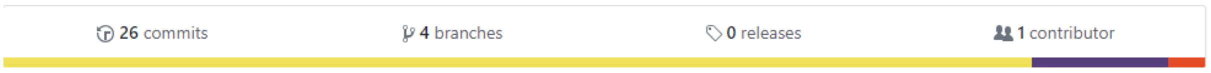
Displays the attributes & operations of system, as well as the constraints imposed on the system.



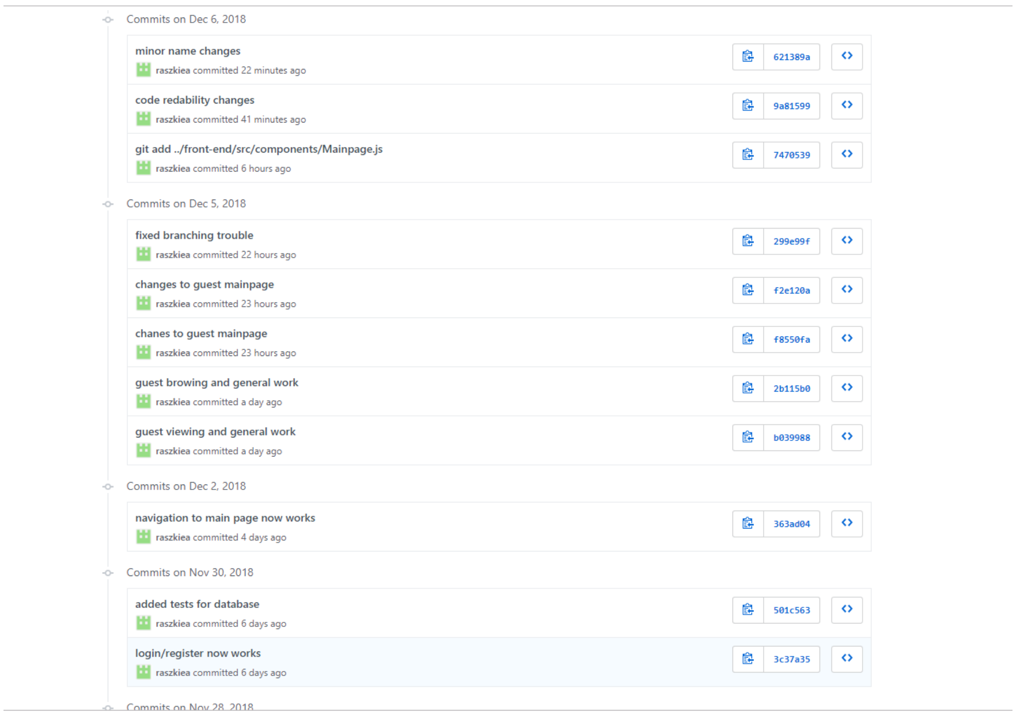
Version Control (GitHub)

Github link: https://github.coventry.ac.uk/340CT-1819SEPJAN/raszkiea340CTT

Evidence of version control



26 commits | 4 branches | 0 releases | 1 contributor



Commits on Dec 6, 2018

minor name changes
raszkiea committed 22 minutes ago — 621389a

code redability changes
raszkiea committed 41 minutes ago — 9a81599

git add ../front-end/src/components/Mainpage.js
raszkiea committed 6 hours ago — 7470539

Commits on Dec 5, 2018

fixed branching trouble
raszkiea committed 22 hours ago — 299e99f

changes to guest mainpage
raszkiea committed 23 hours ago — f2e120a

chanes to guest mainpage
raszkiea committed 23 hours ago — f8550fa

guest browing and general work
raszkiea committed a day ago — 2b115b0

guest viewing and general work
raszkiea committed a day ago — b039988

Commits on Dec 2, 2018

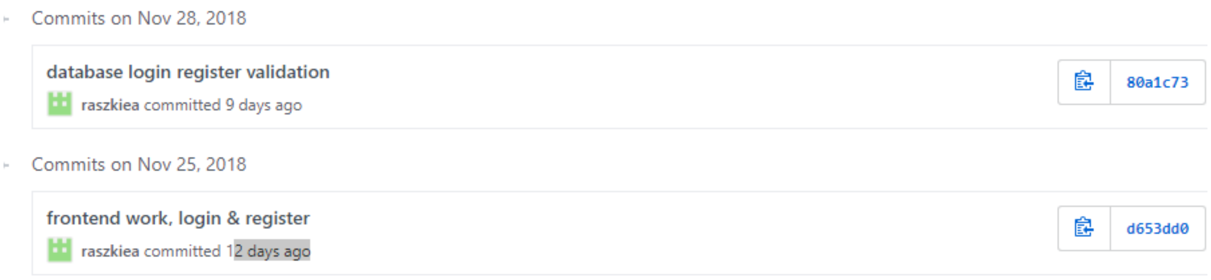
navigation to main page now works
raszkiea committed 4 days ago — 363ad04

Commits on Nov 30, 2018

added tests for database
raszkiea committed 6 days ago — 501c563

login/register now works
raszkiea committed 6 days ago — 3c37a35

Commits on Nov 28, 2018

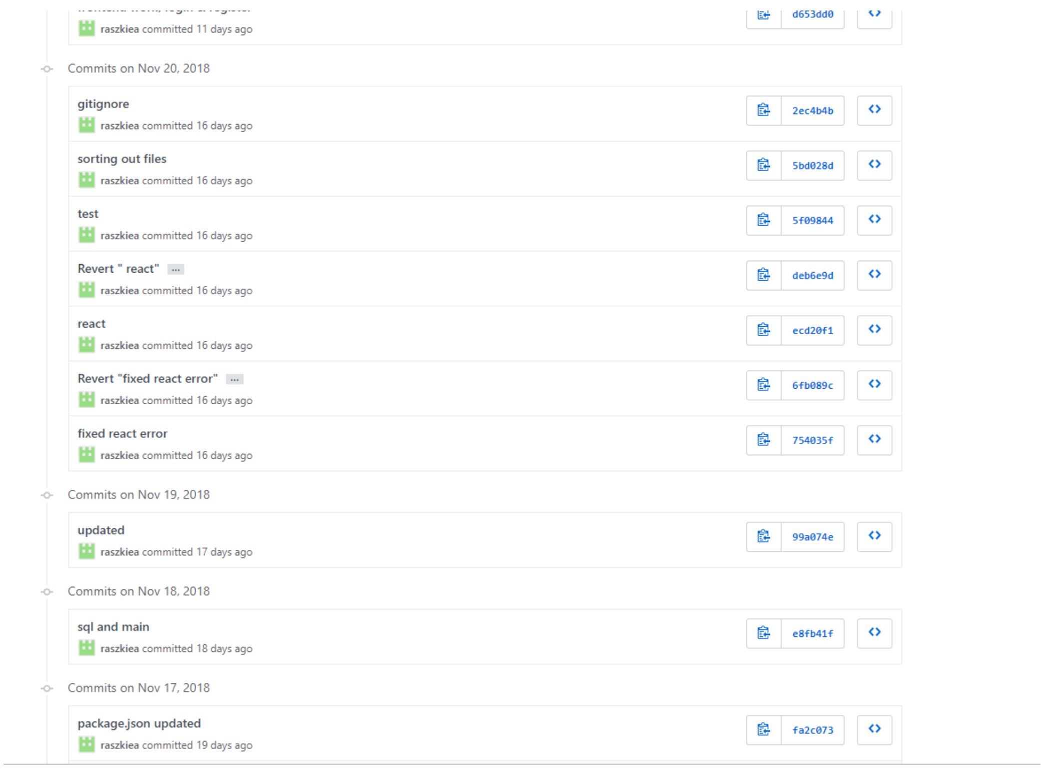


Commits on Nov 28, 2018

database login register validation
raszkiea committed 9 days ago — 80a1c73

Commits on Nov 25, 2018

frontend work, login & register
raszkiea committed 12 days ago — d653dd0



raszkiea committed 11 days ago d653dd0

Commits on Nov 20, 2018

gitignore
raszkiea committed 16 days ago 2ec4b4b

sorting out files
raszkiea committed 16 days ago 5bd028d

test
raszkiea committed 16 days ago 5f09844

Revert " react" ...
raszkiea committed 16 days ago deb6e9d

react
raszkiea committed 16 days ago ecd20f1

Revert "fixed react error" ...
raszkiea committed 16 days ago 6fb089c

fixed react error
raszkiea committed 16 days ago 754035f

Commits on Nov 19, 2018

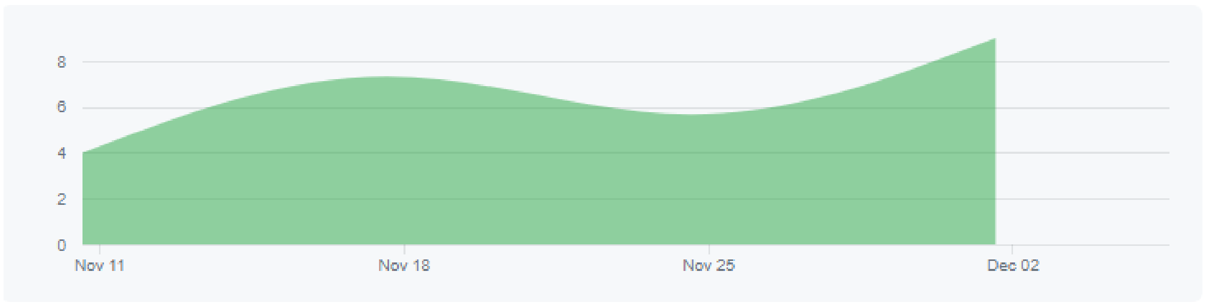
updated
raszkiea committed 17 days ago 99a074e

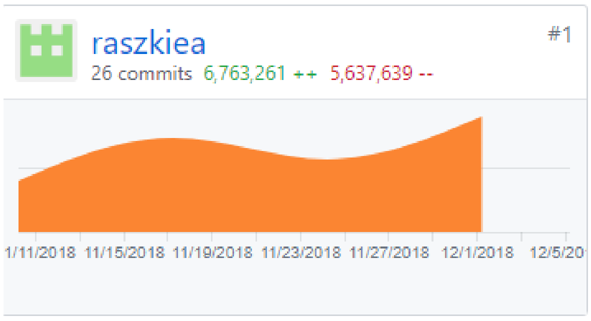
Commits on Nov 18, 2018

sql and main
raszkiea committed 18 days ago e8fb41f

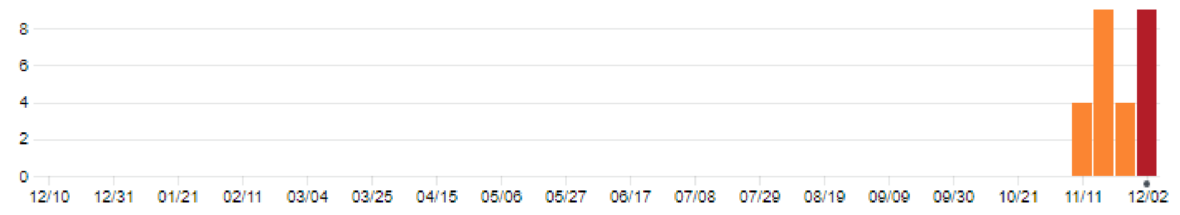
Commits on Nov 17, 2018

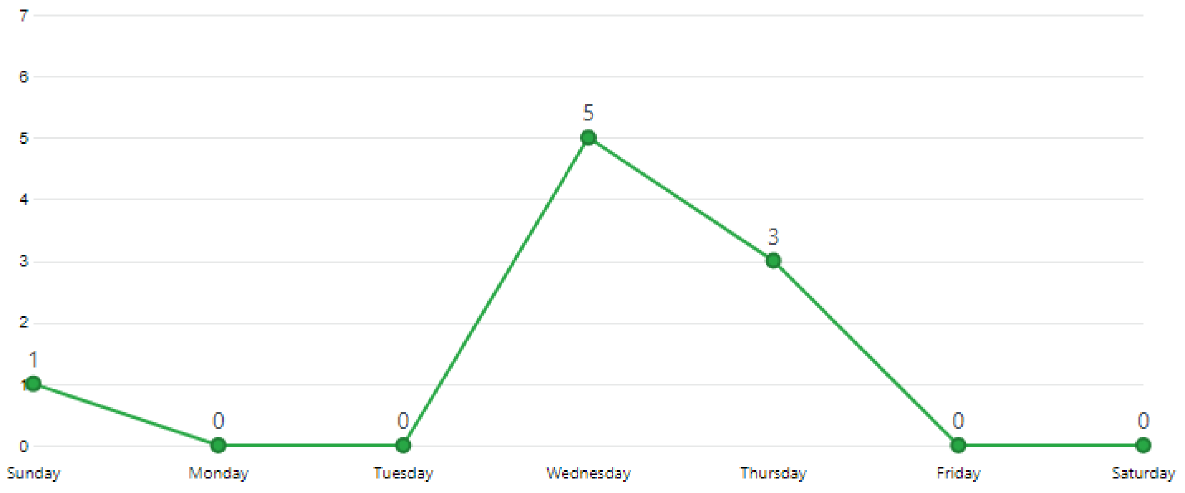
package.json updated
raszkiea committed 19 days ago fa2c073

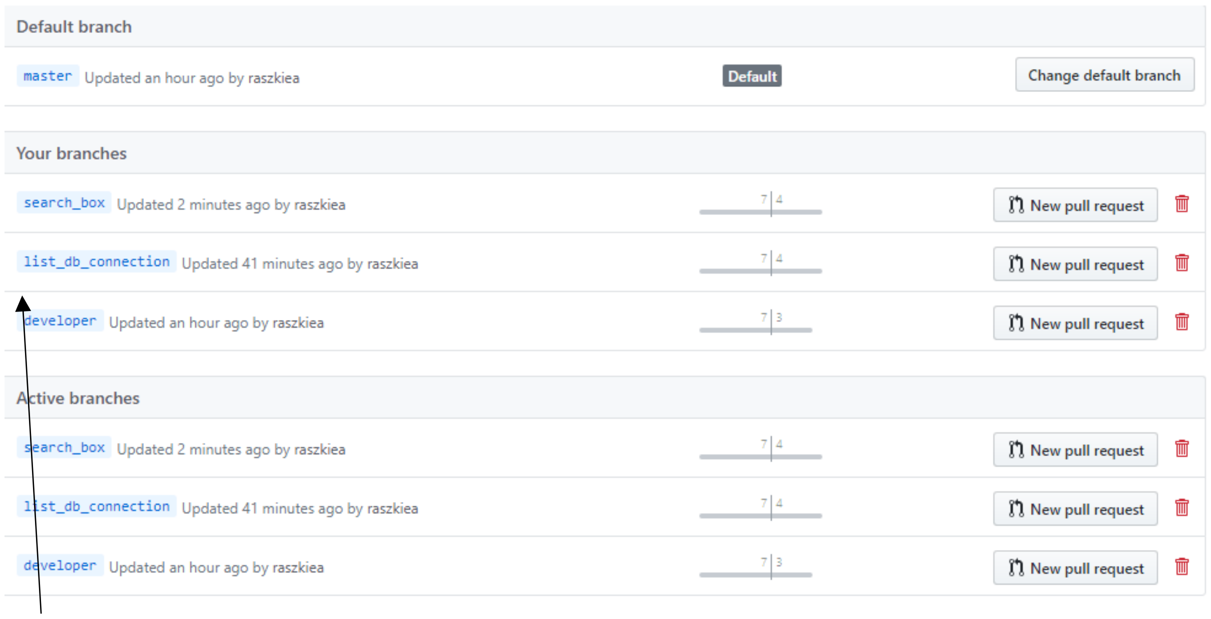




raszkiea #1
26 commits 6,763,261 ++ 5,637,639 --






Default branch

`master` Updated an hour ago by raszkiea Default Change default branch

Your branches

`search_box` Updated 2 minutes ago by raszkiea	7 | 4	New pull request	
`list_db_connection` Updated 41 minutes ago by raszkiea	7 | 4	New pull request	
`developer` Updated an hour ago by raszkiea	7 | 3	New pull request	

Active branches

`search_box` Updated 2 minutes ago by raszkiea	7 | 4	New pull request	
`list_db_connection` Updated 41 minutes ago by raszkiea	7 | 4	New pull request	
`developer` Updated an hour ago by raszkiea	7 | 3	New pull request	

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.

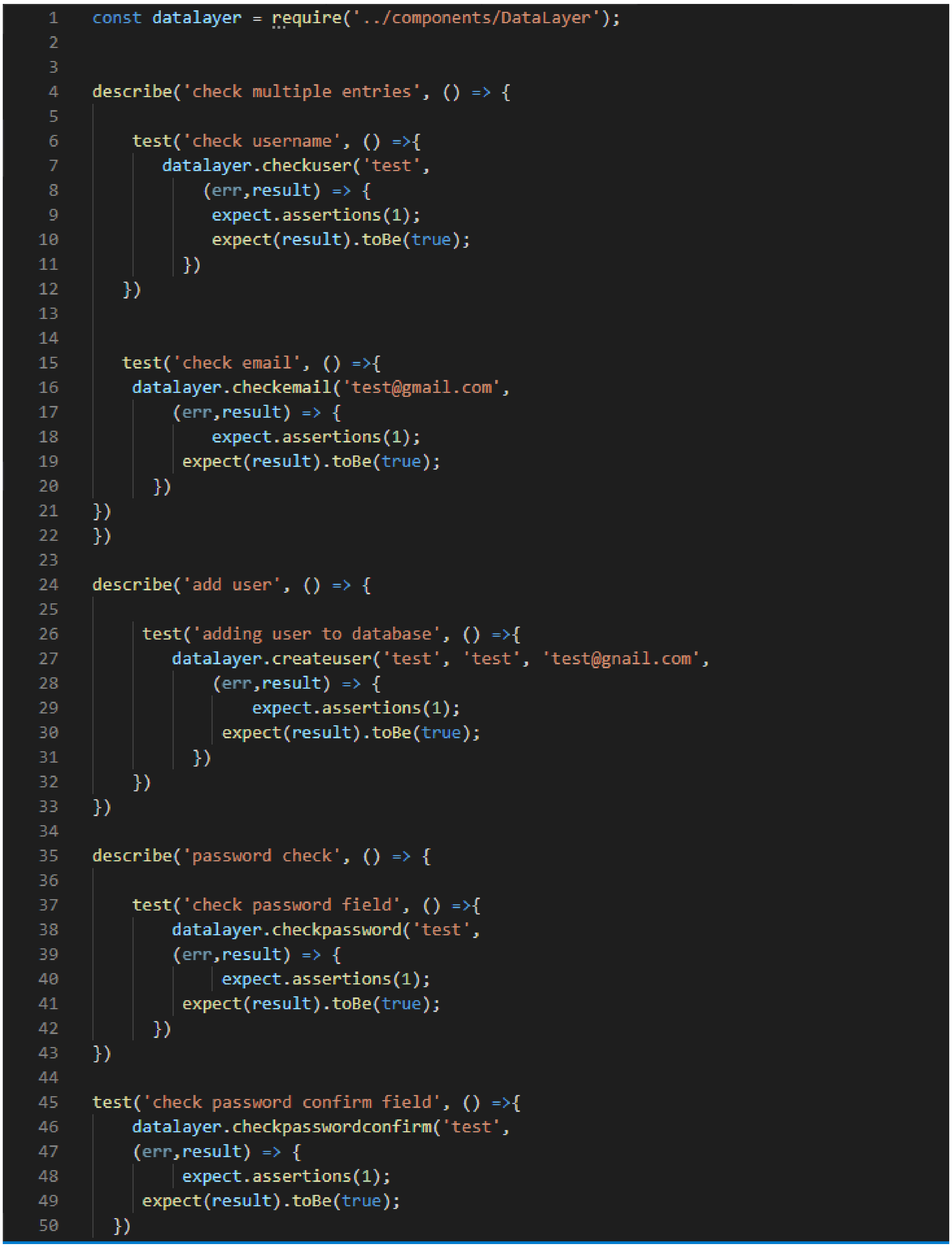
Feature branches (e.g.list_db_connection) used to develop new features for my application, exist as long as the feature is being developed, once the feature is finished, branch pushed back into ‘developer’.

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.

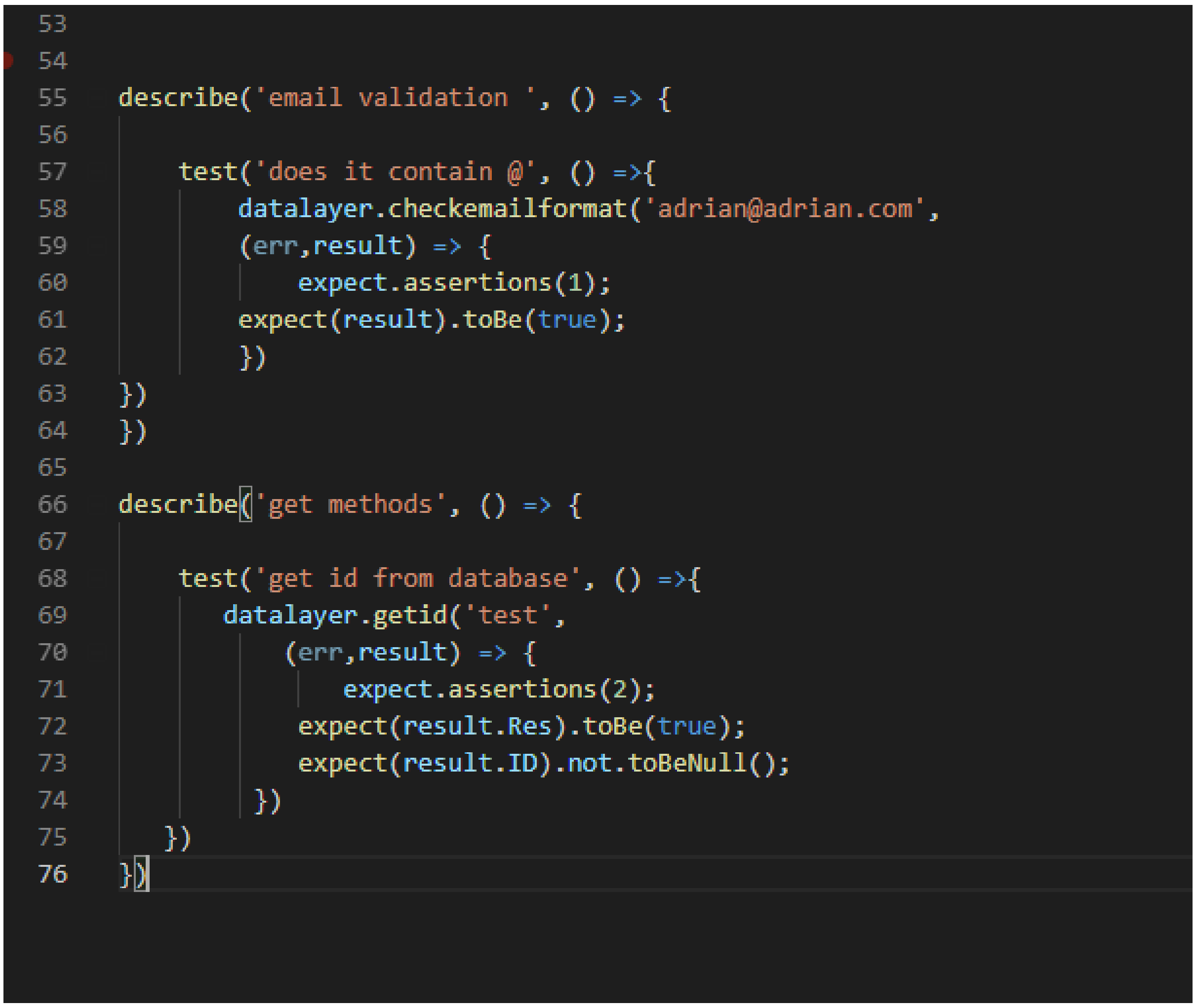
Commands used

- ‘**git reset –hard HEAD~1’** = Delete my previous commit
- **‘git checkout (-b) branch_name**’ = Making new branches and switching
- ‘**git rebase –skip’** and **‘git rebase –continue’** = skip last changes
- ‘**git branch -m name’** and ‘**git push :oldname newname’** = branch names
- ‘git flow innit’
- **‘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
-

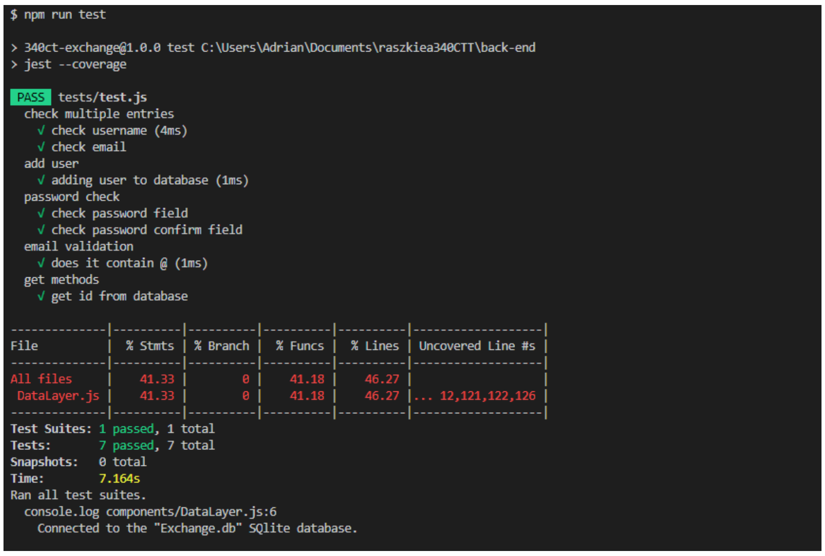
Test Driven Development



```
const datalayer = require('../components/DataLayer');


describe('check multiple entries', () => {

    test('check username', () =>{
       datalayer.checkuser('test',
           (err,result) => {
            expect.assertions(1);
            expect(result).toBe(true);
           })
   })


   test('check email', () =>{
    datalayer.checkemail('test@gmail.com',
        (err,result) => {
            expect.assertions(1);
         expect(result).toBe(true);
      })
})
})

describe('add user', () => {

     test('adding user to database', () =>{
        datalayer.createuser('test', 'test', 'test@gnail.com',
            (err,result) => {
                expect.assertions(1);
             expect(result).toBe(true);
            })
    })
})

describe('password check', () => {

    test('check password field', () =>{
        datalayer.checkpassword('test',
        (err,result) => {
             expect.assertions(1);
         expect(result).toBe(true);
      })
})

test('check password confirm field', () =>{
    datalayer.checkpasswordconfirm('test',
    (err,result) => {
        expect.assertions(1);
     expect(result).toBe(true);
  })
```

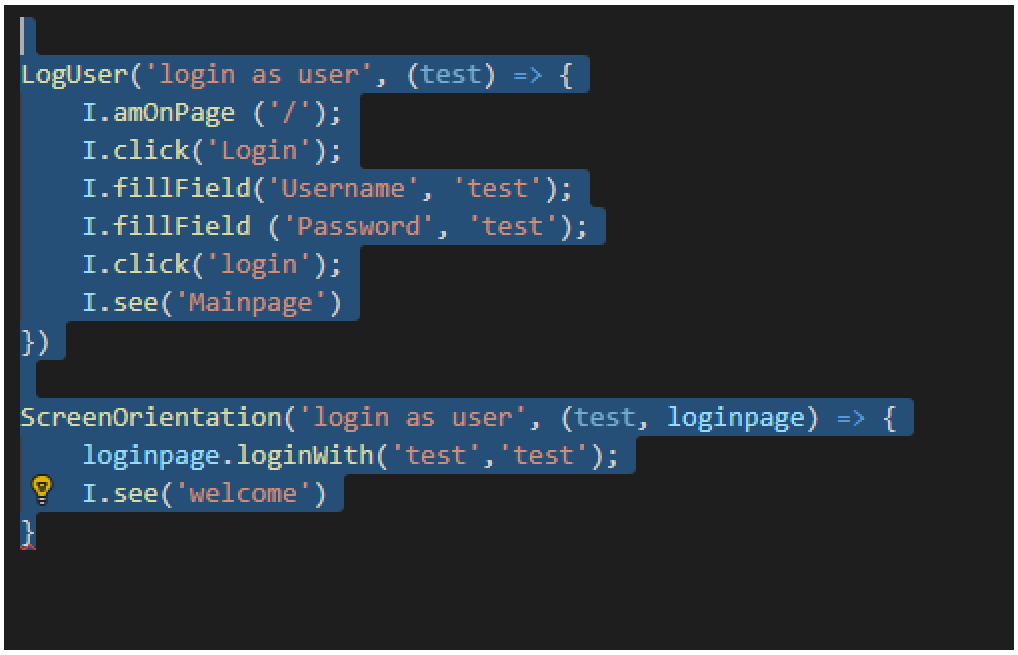


```javascript


describe('email validation ', () => {

    test('does it contain @', () =>{
        datalayer.checkemailformat('adrian@adrian.com',
        (err,result) => {
            expect.assertions(1);
        expect(result).toBe(true);
        })
})
})

describe('get methods', () => {

    test('get id from database', () =>{
       datalayer.getid('test',
           (err,result) => {
              expect.assertions(2);
           expect(result.Res).toBe(true);
           expect(result.ID).not.toBeNull();
           })
      })
})
```



```
$ npm run test

> 340ct-exchange@1.0.0 test C:\Users\Adrian\Documents\raszkiea340CTT\back-end
> jest --coverage

 PASS  tests/test.js
  check multiple entries
    √ check username (4ms)
    √ check email
  add user
    √ adding user to database (1ms)
  password check
    √ check password field
    √ check password confirm field
  email validation
    √ does it contain @ (1ms)
  get methods
    √ get id from database

--------------|----------|----------|----------|----------|-------------------|
File          |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
--------------|----------|----------|----------|----------|-------------------|
All files     |    41.33 |        0 |    41.18 |    46.27 |                   |
 DataLayer.js |    41.33 |        0 |    41.18 |    46.27 |... 12,121,122,126 |
--------------|----------|----------|----------|----------|-------------------|
Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        7.164s
Ran all test suites.
  console.log components/DataLayer.js:6
    Connected to the "Exchange.db" SQlite database.
```

Acceptance testing attempt



```
LogUser('login as user', (test) => {
    I.amOnPage ('/');
    I.click('Login');
    I.fillField('Username', 'test');
    I.fillField ('Password', 'test');
    I.click('login');
    I.see('Mainpage')
})

ScreenOrientation('login as user', (test, loginpage) => {
    loginpage.loginWith('test','test');
    I.see('welcome')
}
```

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/>** [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-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/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-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/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/