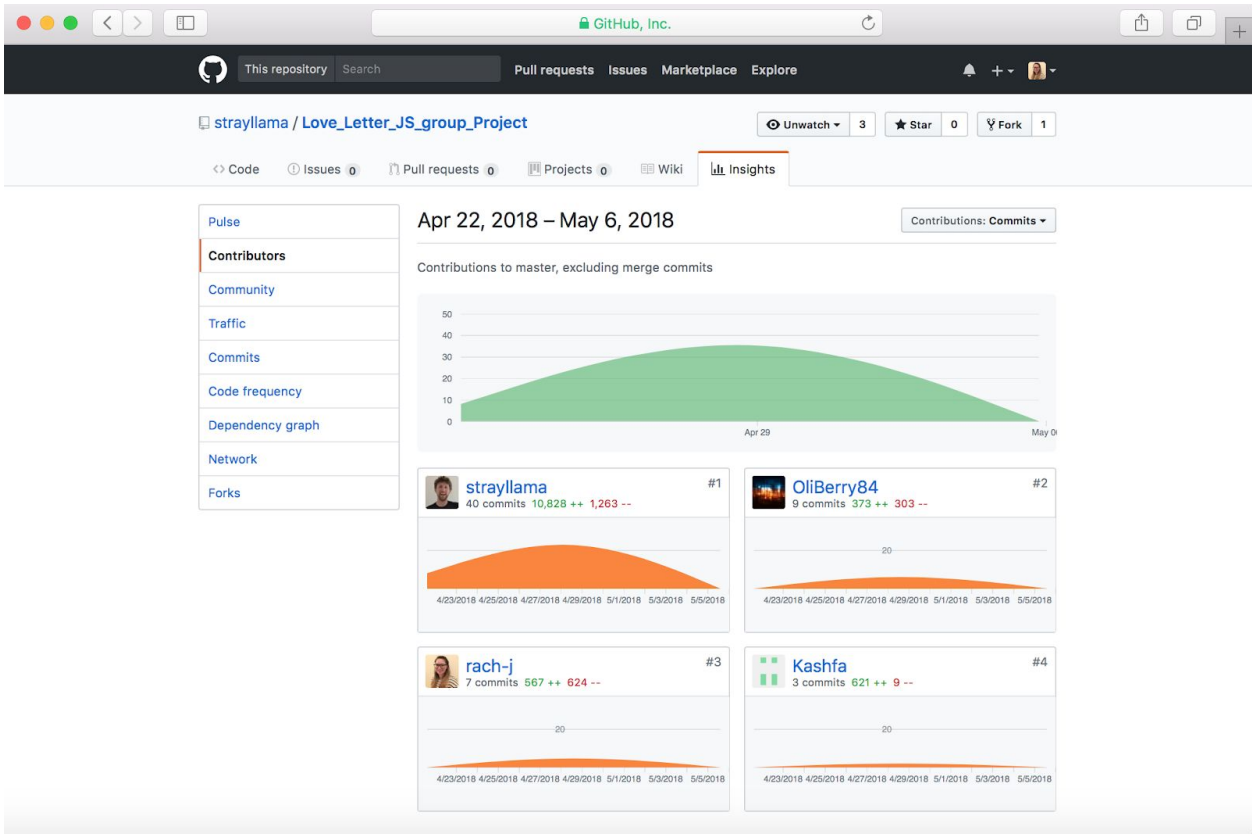
Evidence for Project Unit

Rachel Johnson
E19

P.1 Contributors page on Github



P.2 Project brief

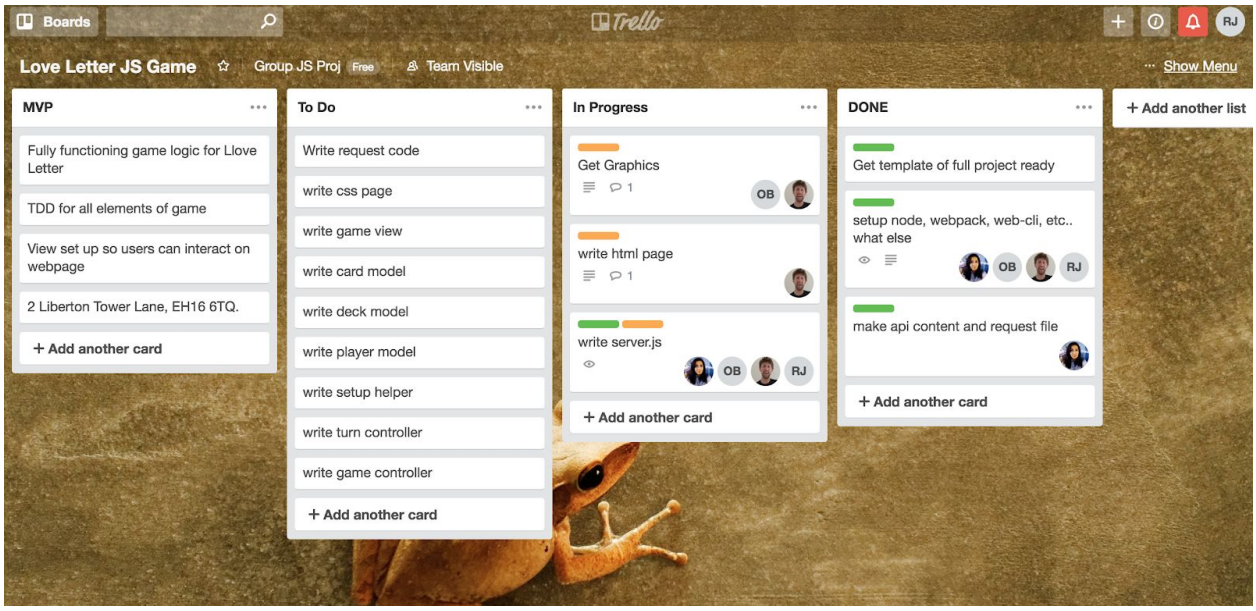
Browser Game

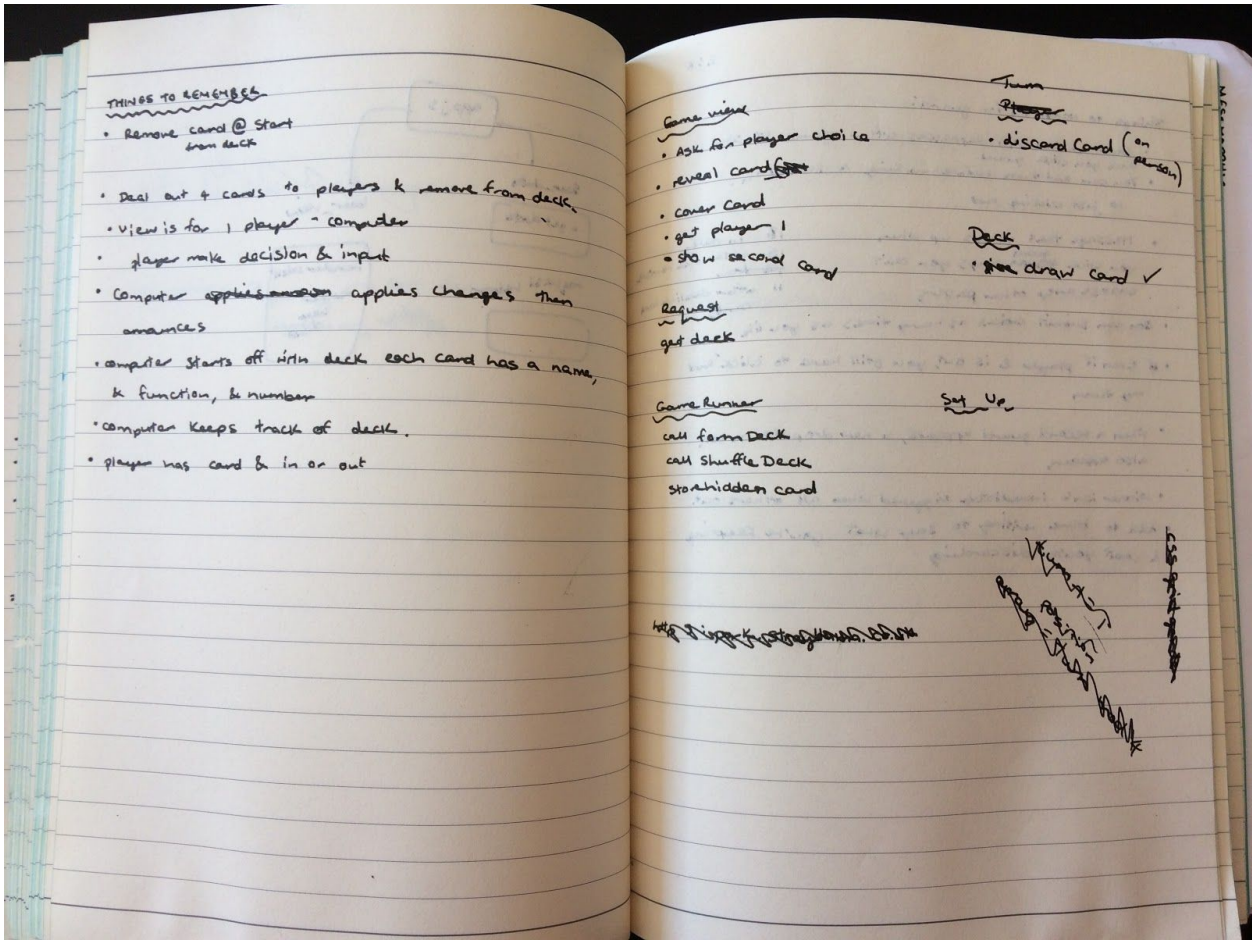
Create a browser game based on an existing card or dice game. Model the game logic and then display it in the browser for a user to interact with.

Make your own MVP with some specific goals to be achieved based on the game you choose to model.

You might use persistence to keep track of the state of the game or track scores/wins. Other extended features will depend on the game you choose.

P.3 Planning for group project



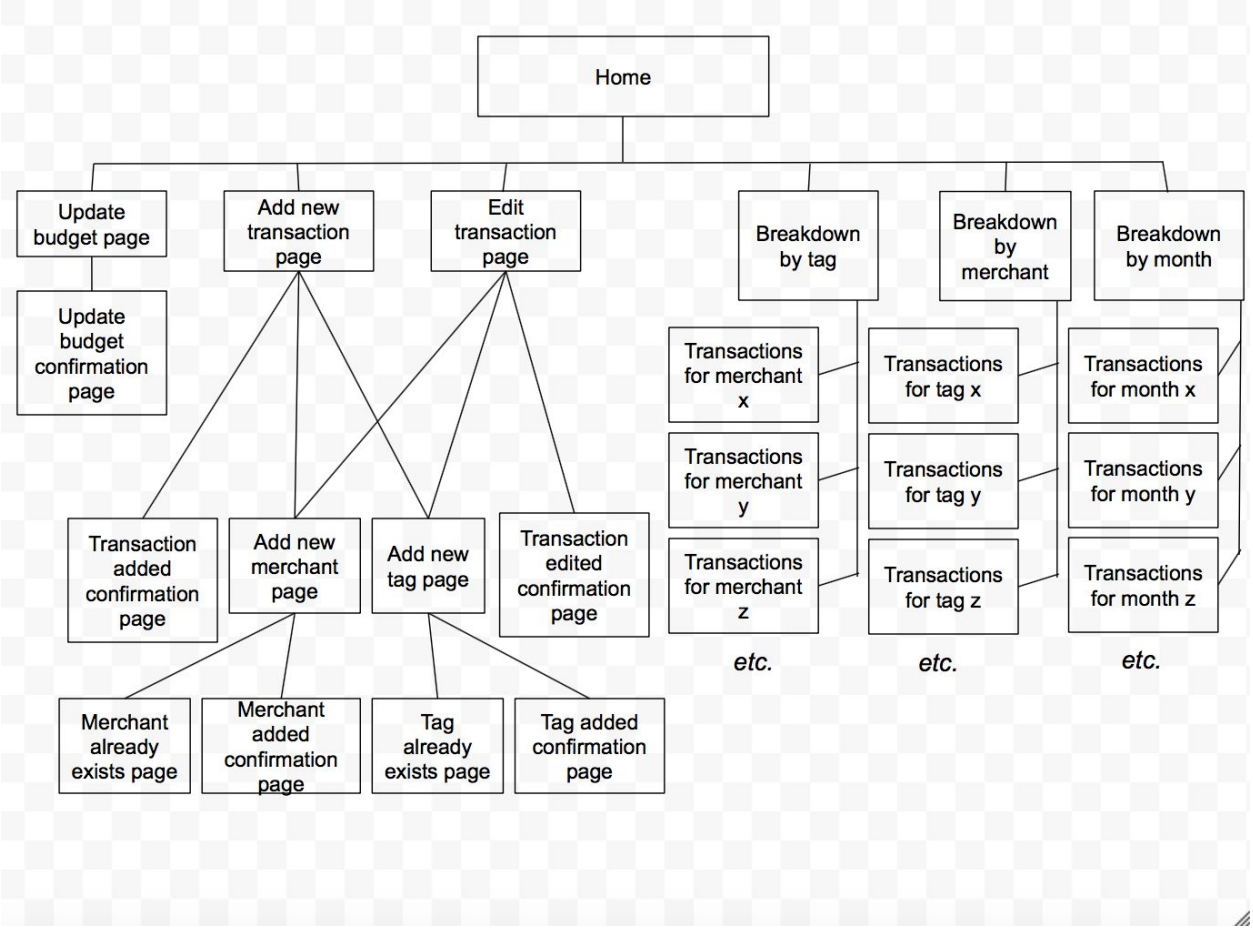


P.4 Acceptance criteria and test plan

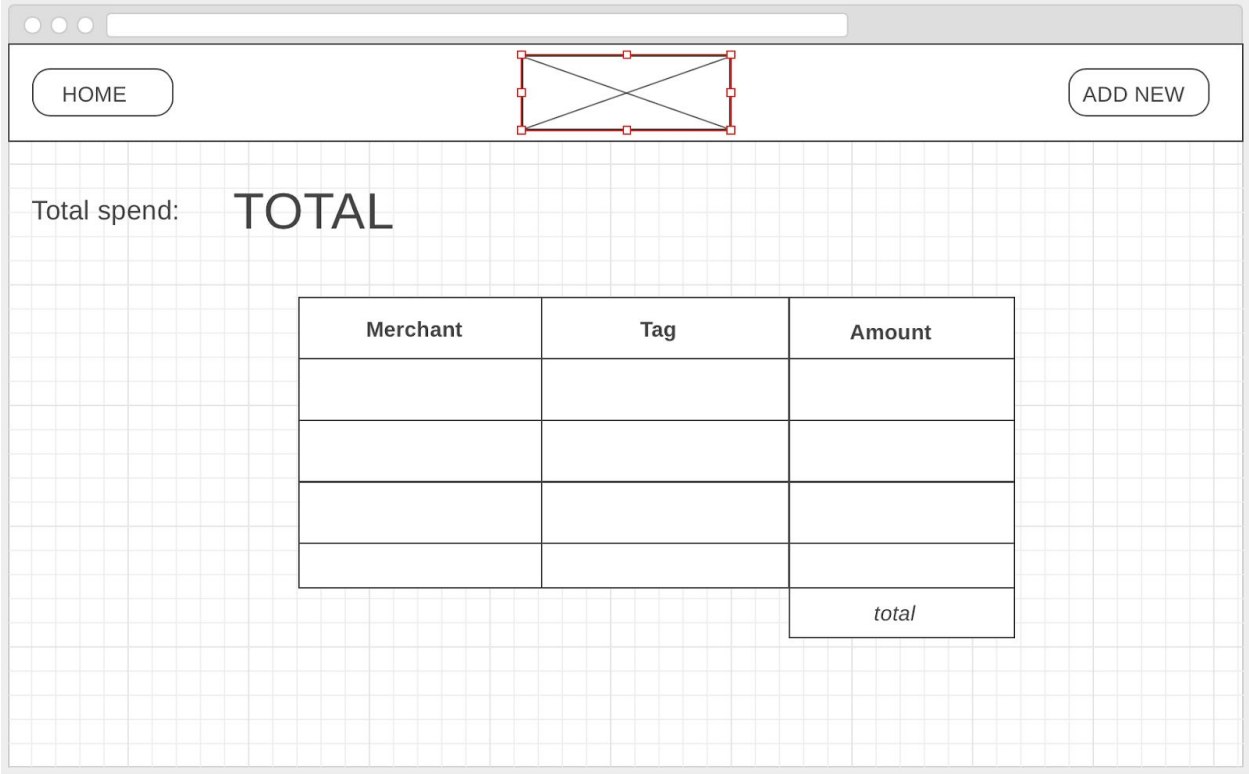
Acceptance criteria	Expected result / output	Pass / Fail
User is able to enter name	Free text box enables user to type in name & upon pressing start button, name is fixed and appears above each player's cards	Pass
User is able to start game	When start button is clicked, audio message instructs user to start playing and first cards are revealed.	Pass
User is able to select a card from his/her hand (where game rules permit), and this should then appear in the discard pile.	When selected card is clicked, providing game rules allow that card to be selected, card will appear in 'discard pile' in centre of screen, area where card was previously shown will then show a blank space, and options relating to the card will appear on screen.	Pass
User is only able to select one of his/her own cards.	On a users turn, only his / her own cards will be visible on screen. All other card spaces on screen will be blank and unclick-able.	Pass
User is unable to end turn until a card has been selected and necessary actions have been carried out.	'End turn button' is greyed out and unclick-able until player has selected a card and completed any required actions.	Pass
User is able to select another player where the rules of the card require this.	When relevant cards are selected, a dropdown box containing other players names, and a submit button will appear.	Pass
Where a player must select another player, he / she may only select players permitted under the game rules.	When the dropdown box containing names of players appears (upon selection of particular cards), only the names of players that can be	Pass

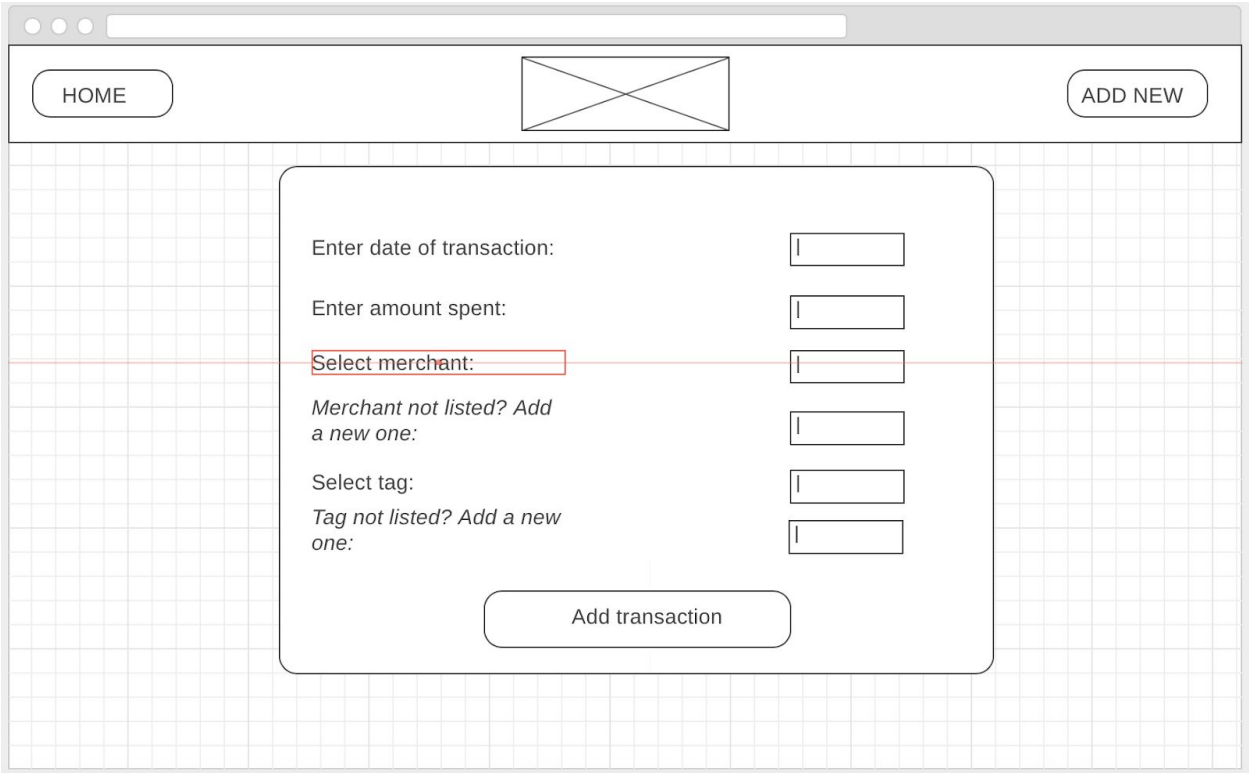
	selected under the rules of the game will appear.	
User can remove other players from the game through certain cards.	When a player has been removed from the game under the game rules, a skull and cross-bones appear under his / her name, and he / she is no longer given the option to select a card on what would be his / her turn (game play skips to the next active player). He / she does not appear in the drop down box for selection when another player is required to select a player.	Pass
User is able to win / lose the game by being the only remaining active player in the game, or by having the highest card in his / her hand when all the cards in the deck have been used up.	When the deck has run out, or there is only one person left in the game, an audio message plays, announcing there has been a winner and a message appears on screen announcing the winner.	Pass

P. 5 User sitemap

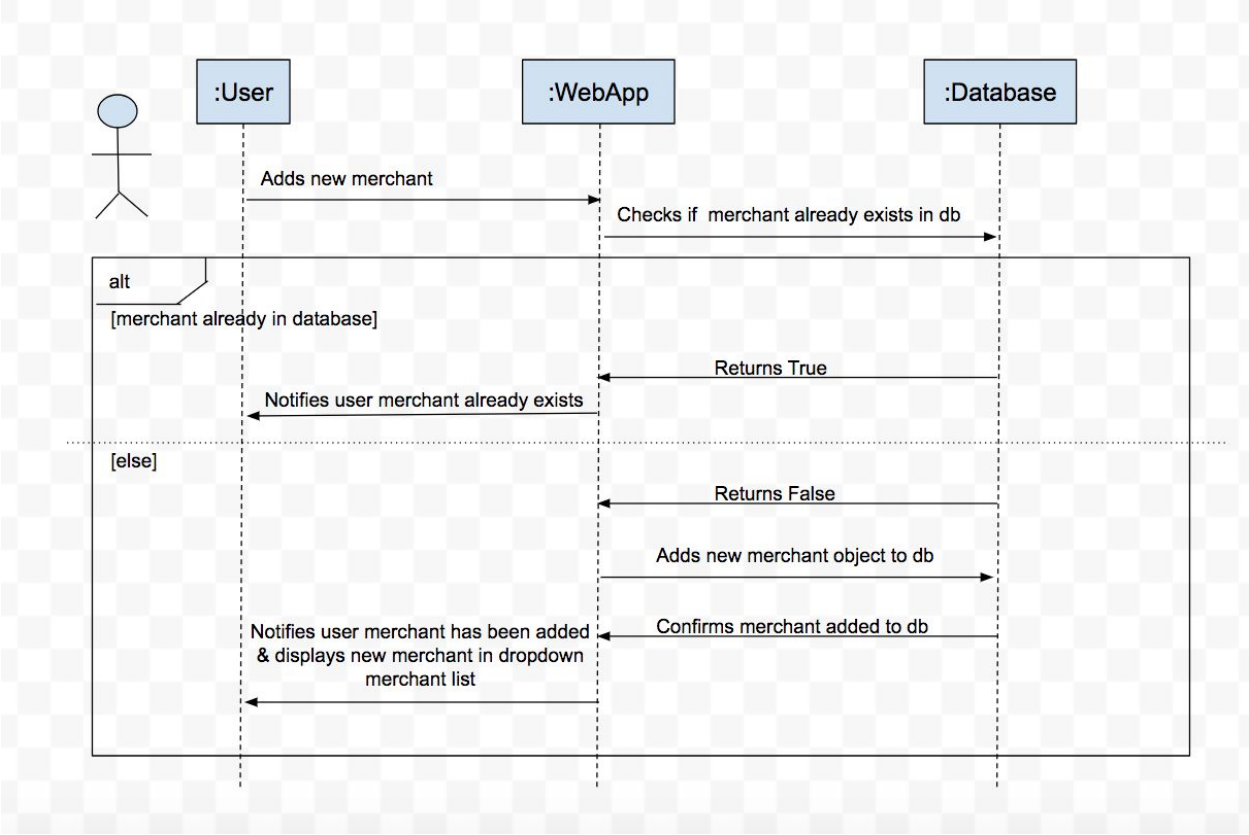


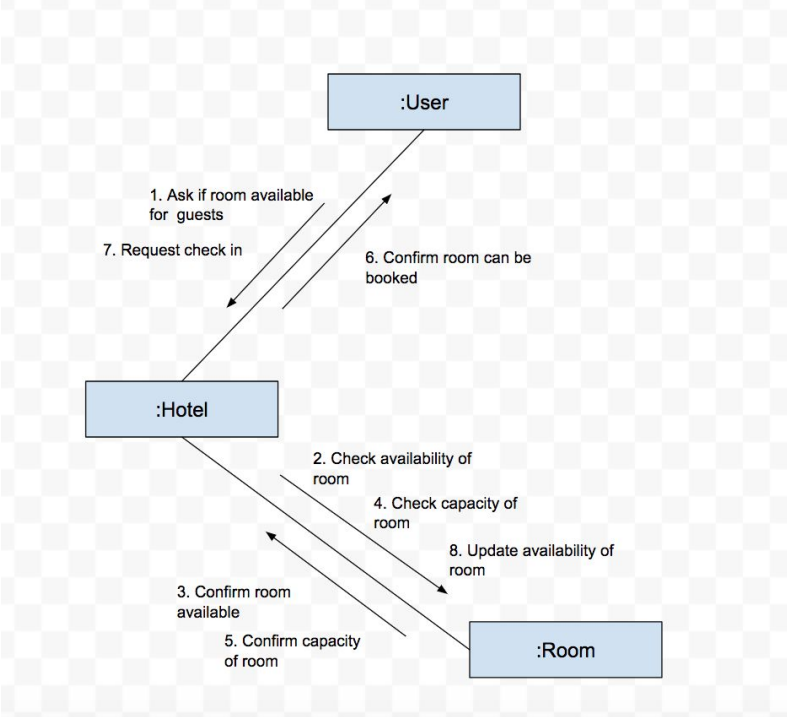
P. 6 Wireframes



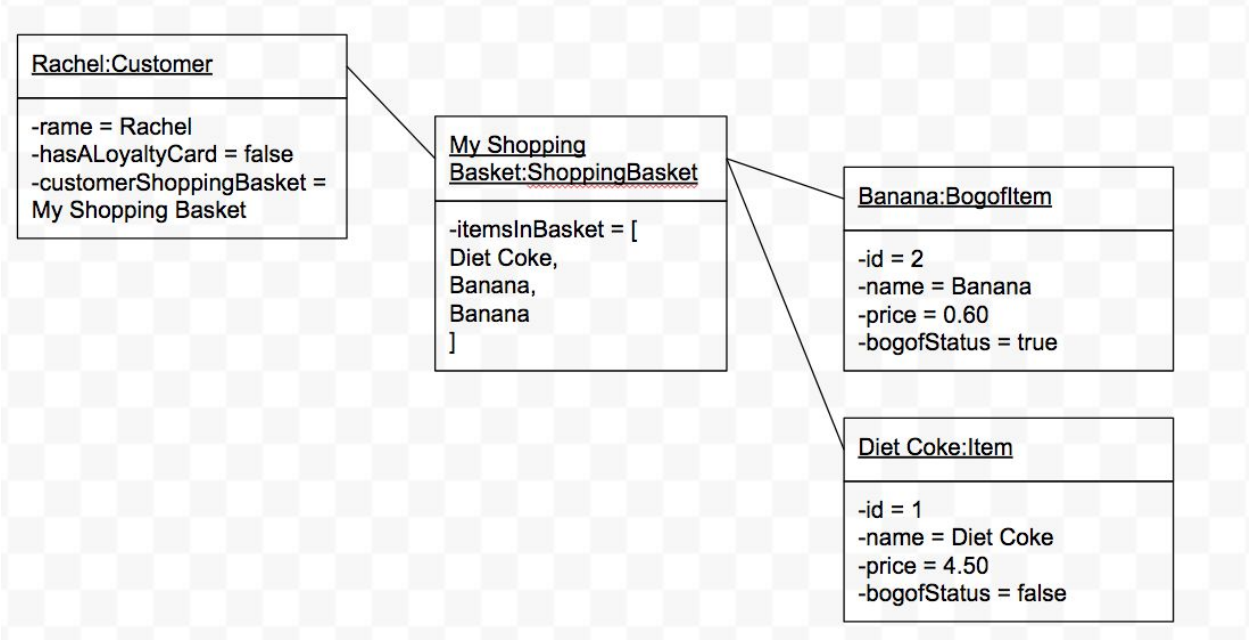


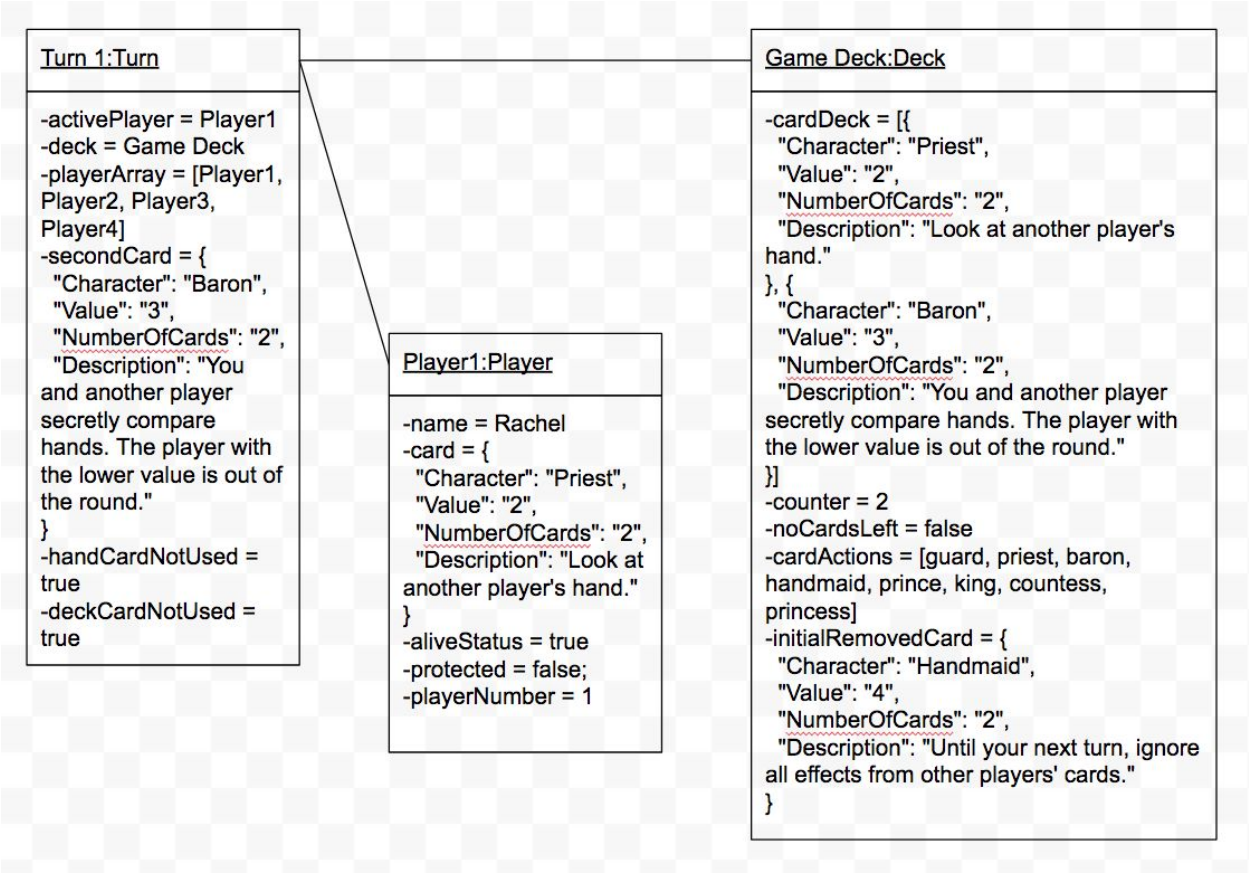
P.7 System interaction diagrams





P.8 Object diagrams





P.9 Algorithms

```
public double getValueOfNonBogofItems(ShoppingBasket shoppingBasket) {
    double total = 0;
    for (Item item: shoppingBasket.getItemsInBasket()) {
        if(!item.getBogofStatus()) {
            total += item.getPrice();
        }
    }
    return total;
}
```

The above algorithm runs through all items in a shopping basket, checks whether they are buy one get one free (bogof), and if they aren't, adds the cost of the item to the running total. This algorithm therefore enables you to calculate the total cost of non-bogof items in a shopping basket (which can then be added to the value of the buy one get one free items to get the total value of items in the shopping basket).

```
Park.prototype.numberOfDinosaursAfterYear = function (year) {
  if(year < 0) {
    return;
  }
  let total = 0;

  if (year === 0) {
    return this.enclosure.length;
  } else {
    for (const dinosaur of this.enclosure) {
      let totalForDino =  1;
      for (let i = 1; i <= year; i++) {
        totalForDino = totalForDino + totalForDino * dinosaur.numberOfOffspring;
      }
      total += totalForDino;
    }
  }
  return total;
};
```

This algorithm calculates the number of dinosaurs in a park after a specified number of years, where we know how many dinosaurs are in the park initially, and we know how many offspring each dinosaur produces each year (it is assumed that dinosaurs that produce e.g. 3 offspring per year will produce offspring that also produce 3 offspring per year).

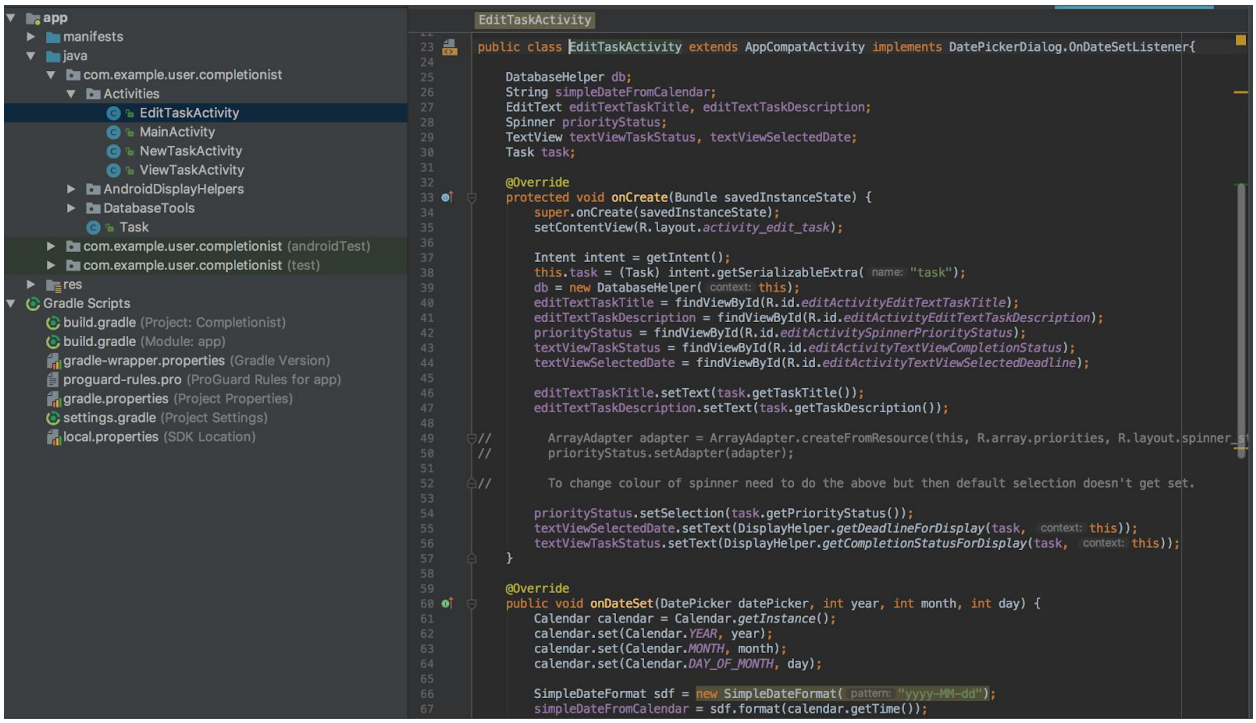
It does this by running through each dinosaur in the park, and for each dinosaur, totalling up the cumulative number of dinosaurs spawned from that dinosaur each year, and then adding all of these numbers together to get the total number of dinosaurs spawned all together over the number of years specified.

P.10 Pseudocode for a function

```
function to find all transactions with a given merchant
  find all transactions from the transactions table where the merchant_id in the table is the same as the id of the merchant in question
  return this selection as an array of transactions where the transactions are ordered by transaction date
end
```

P.11 Project I have worked alone on

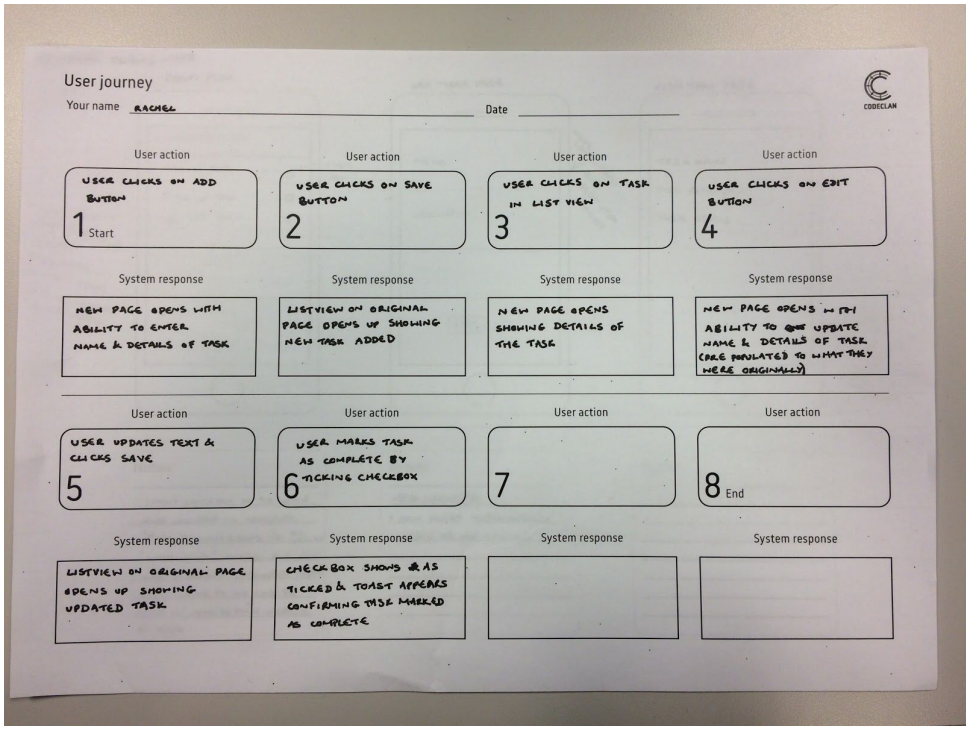
Screenshot:



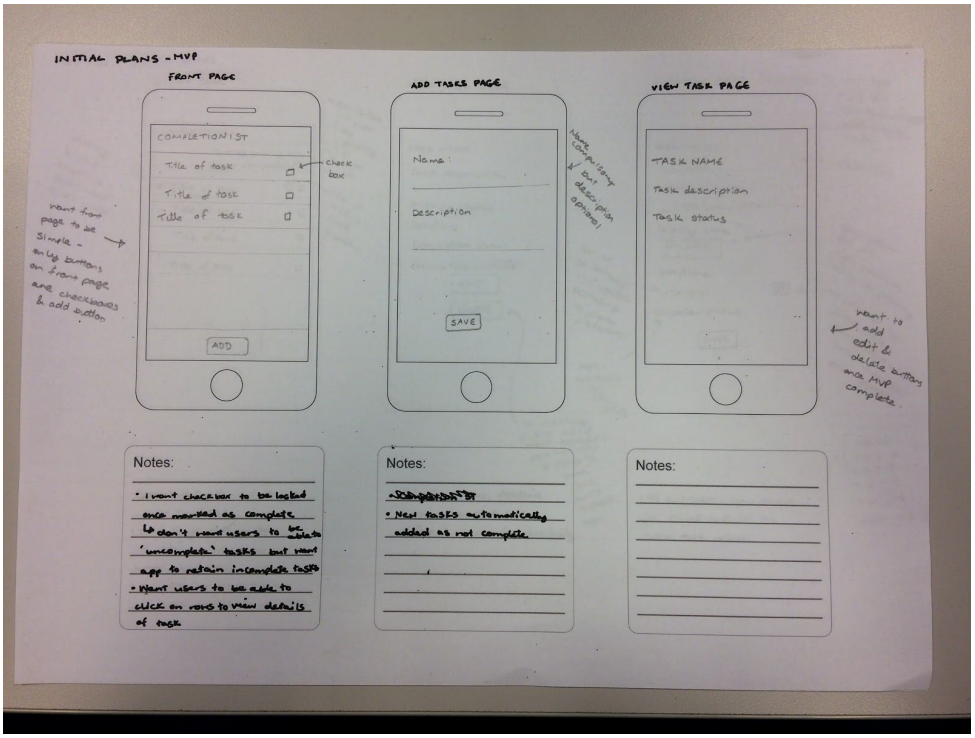
Github link: https://github.com/rach-j/project_2_to_do_list.git

P.12 Planning and development of project

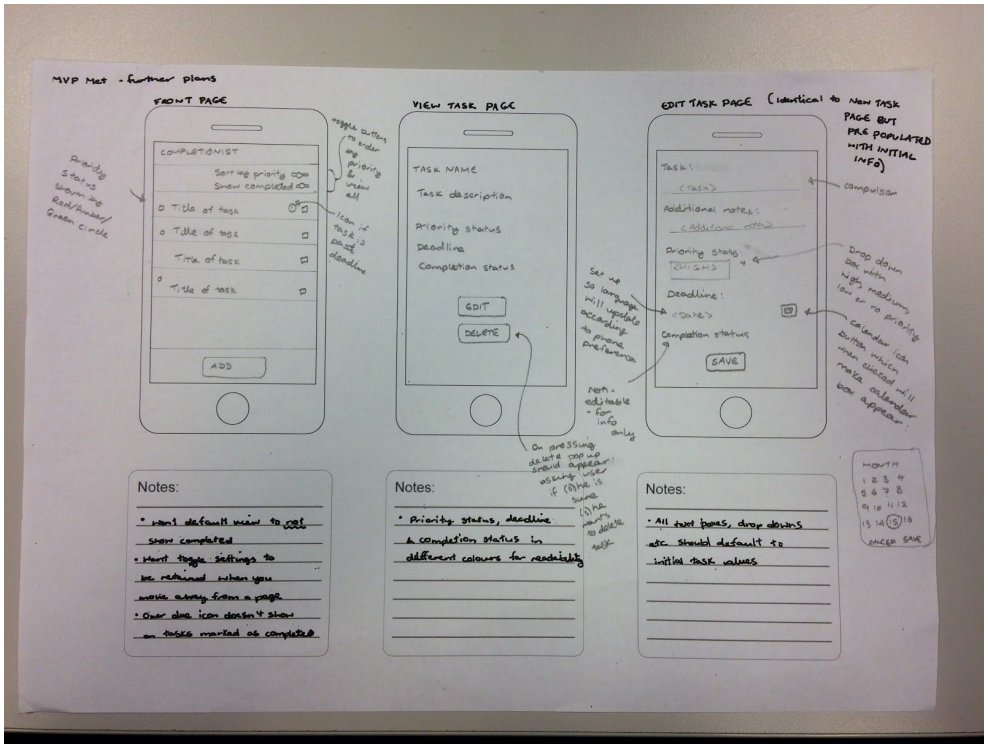
Planned user journey:



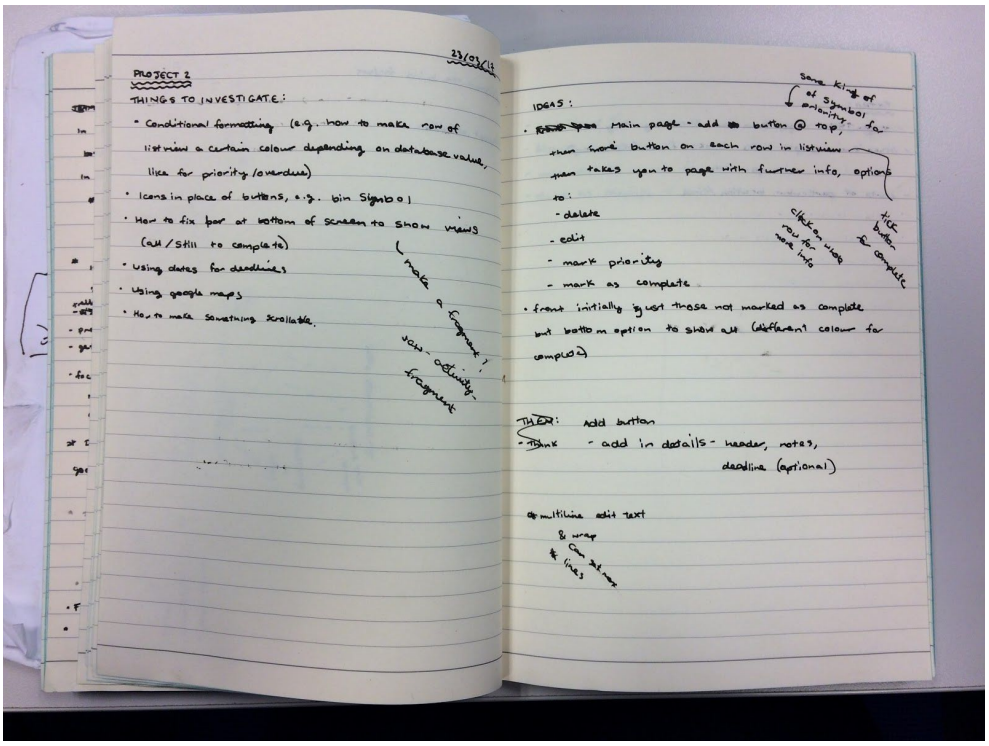
Initial app layout plans:

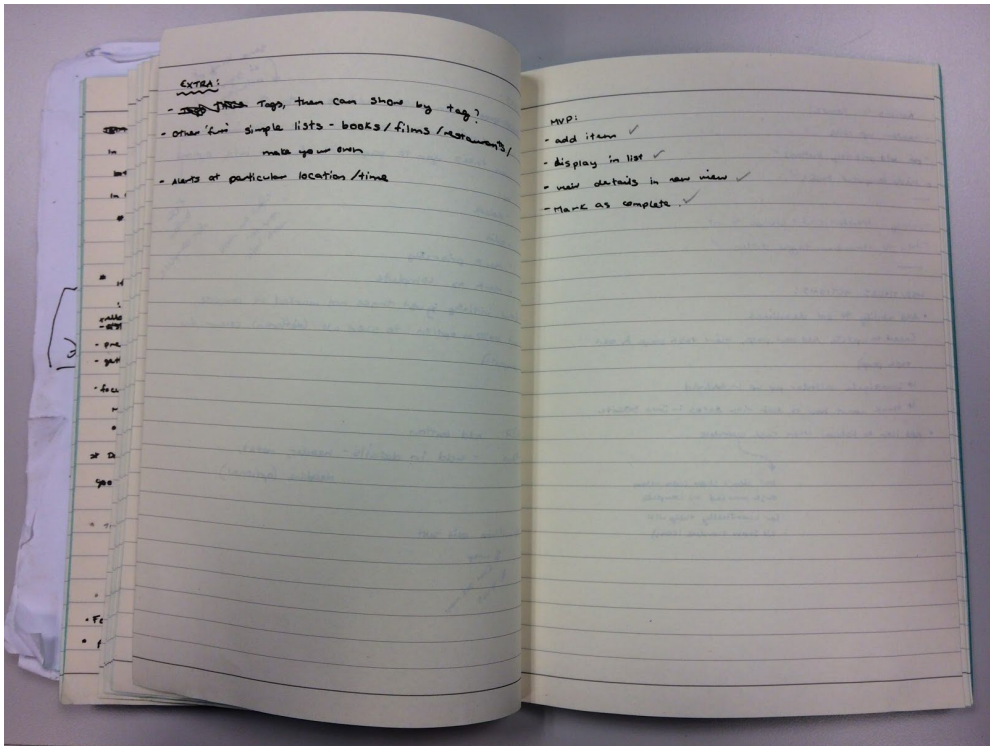


Next step layout plans once MVP hit:

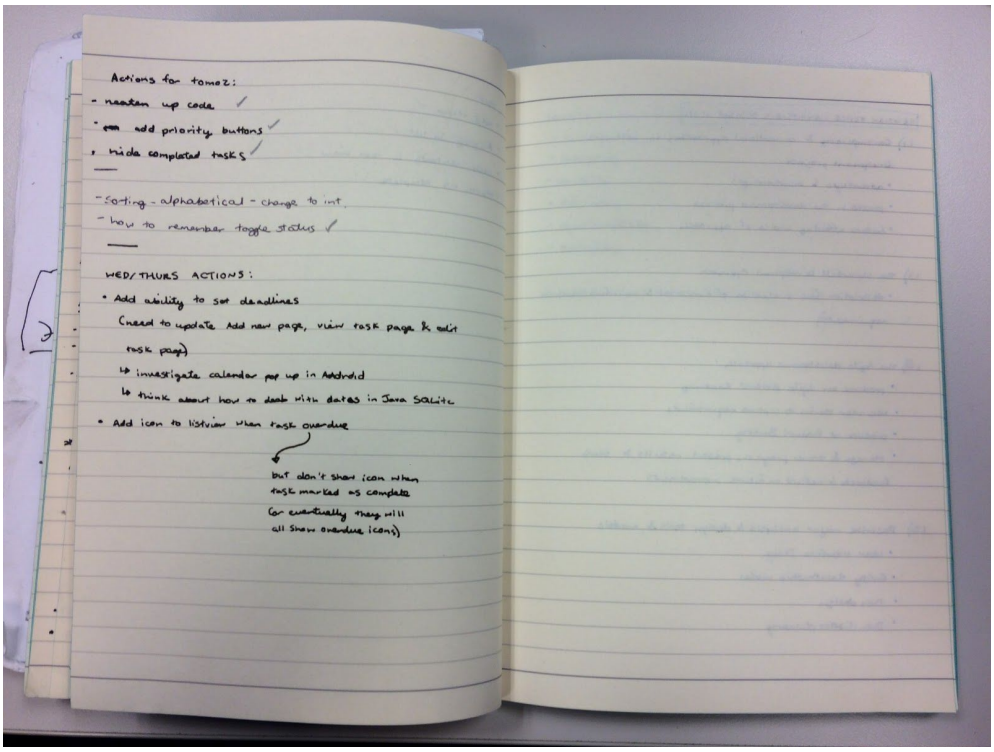


Initial plans / things to investigate for project:

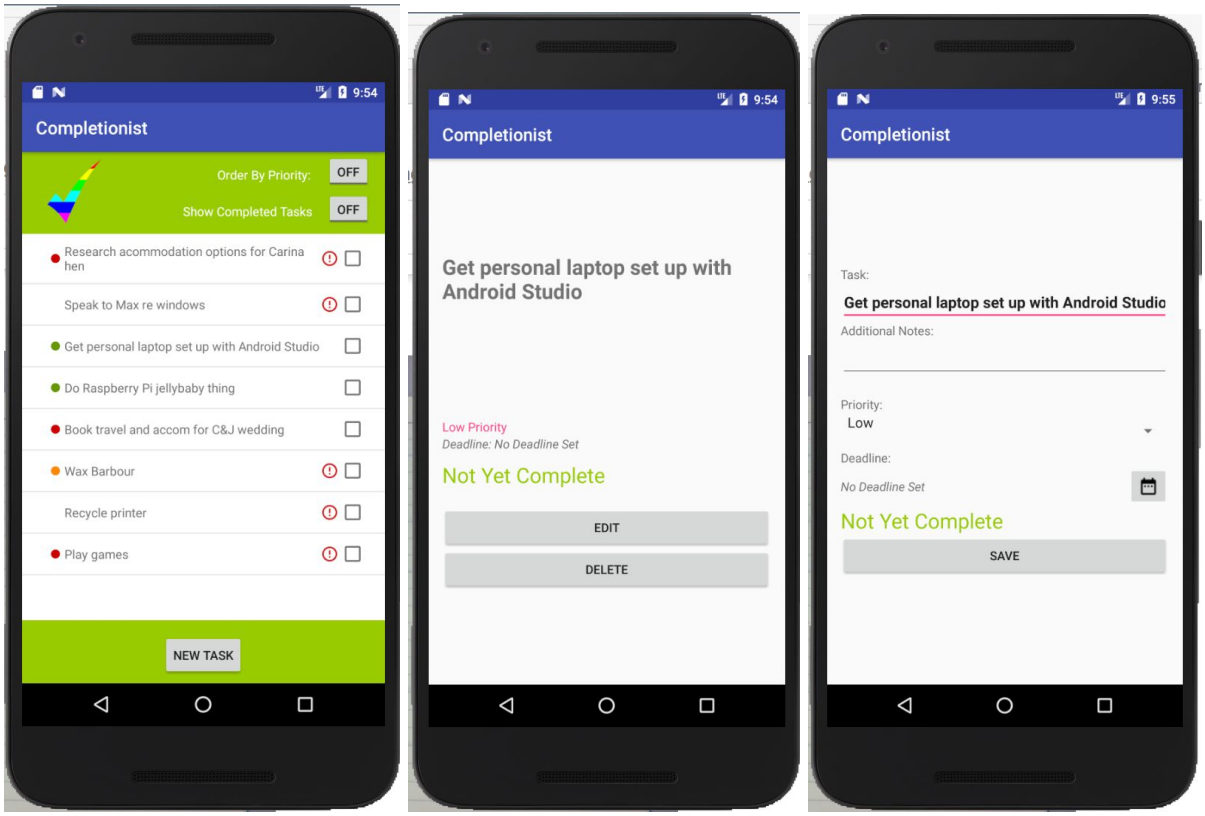




Plans once MVP hit:

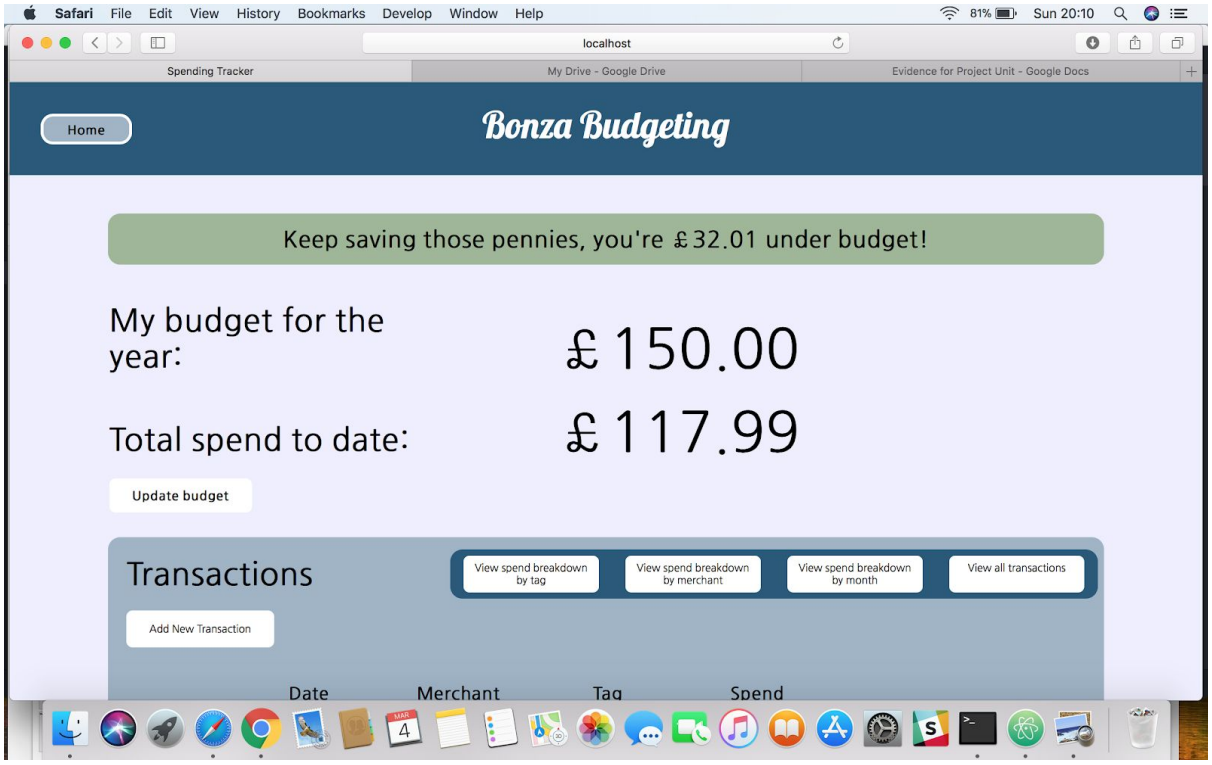


Screenshots of final app:

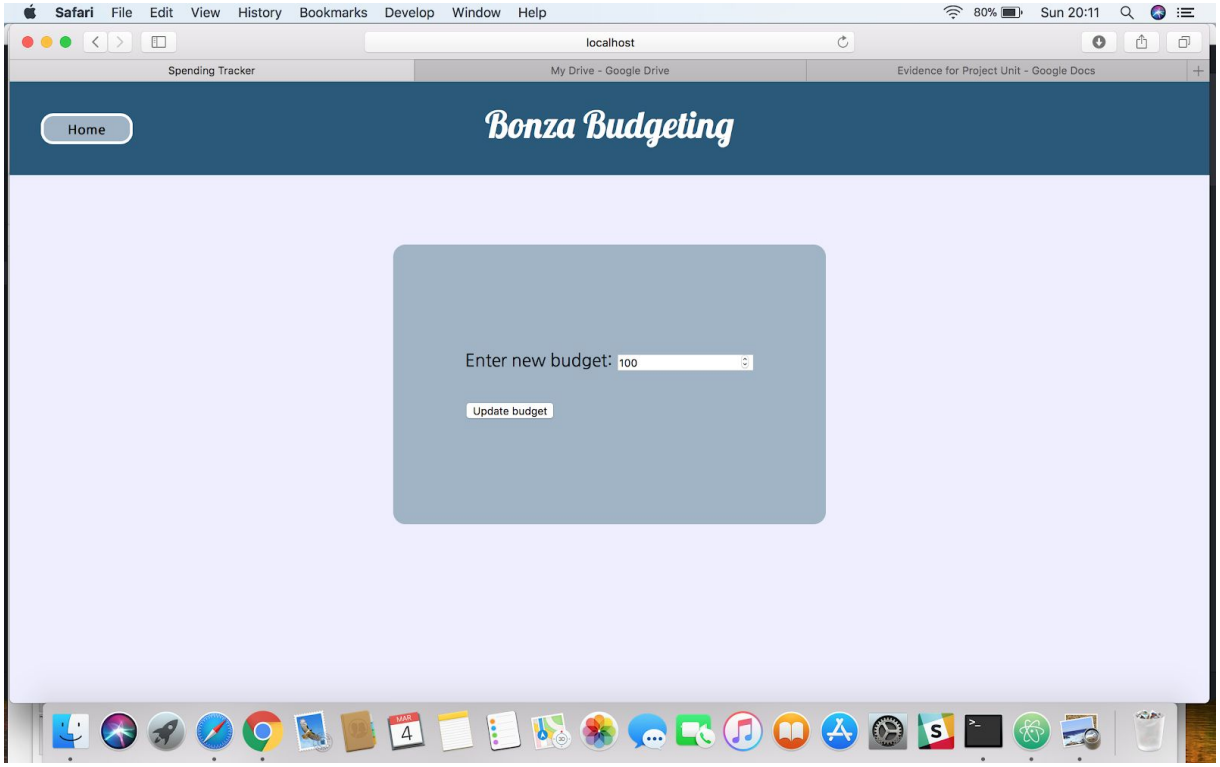


P.13 User input

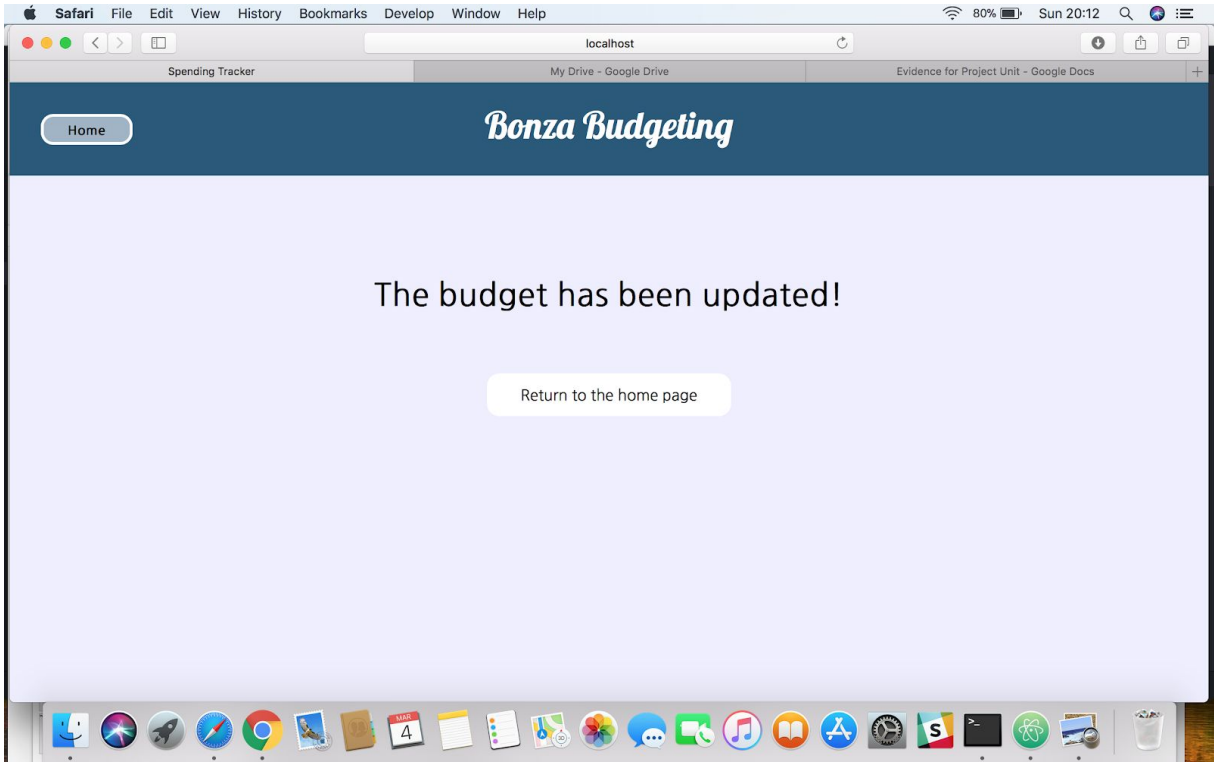
Original page. User clicks to update budget:



User updates budget:



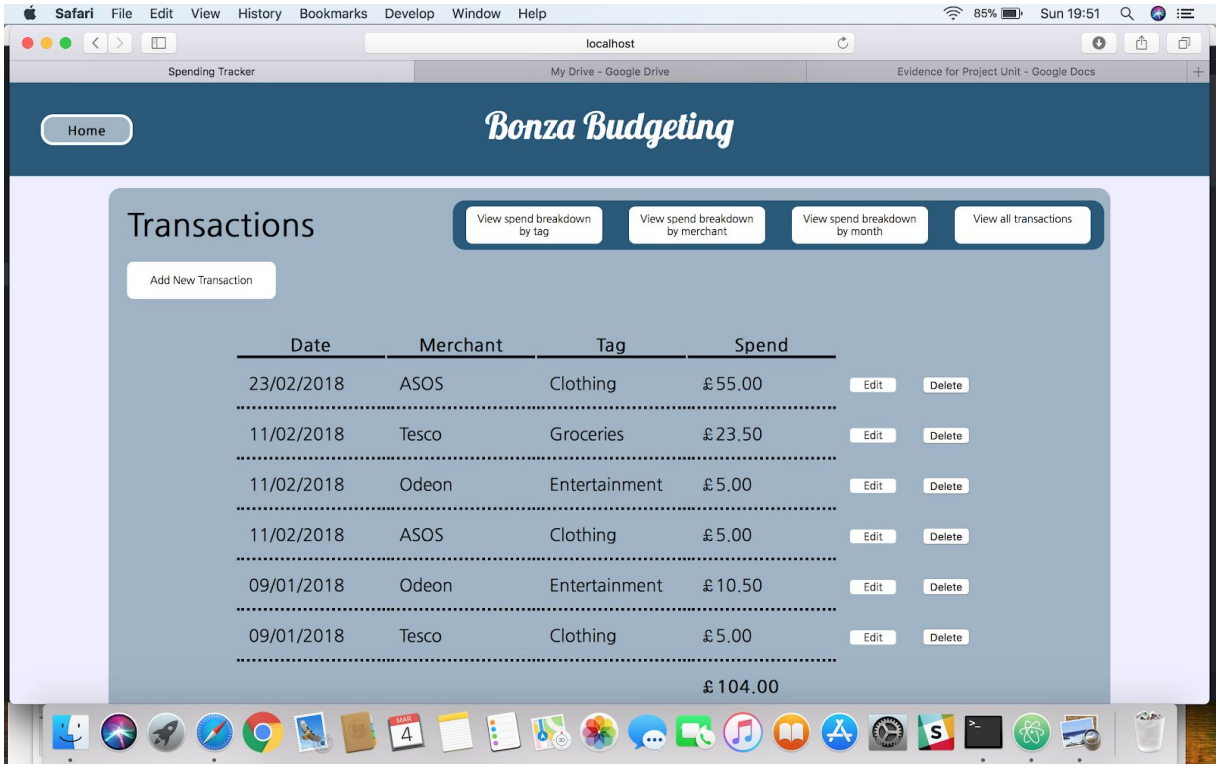
Confirmation page is generated:



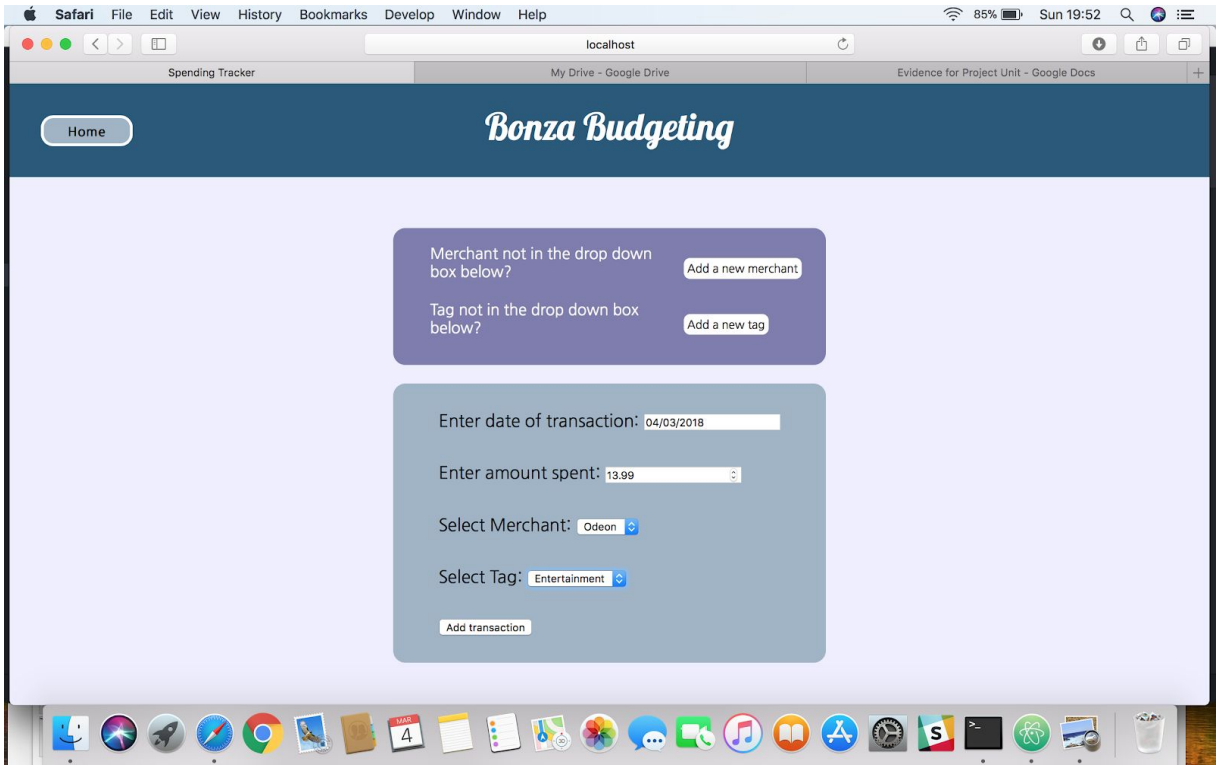
Home page now shows updated budget and budget message:

P.14 Interaction with data persistence

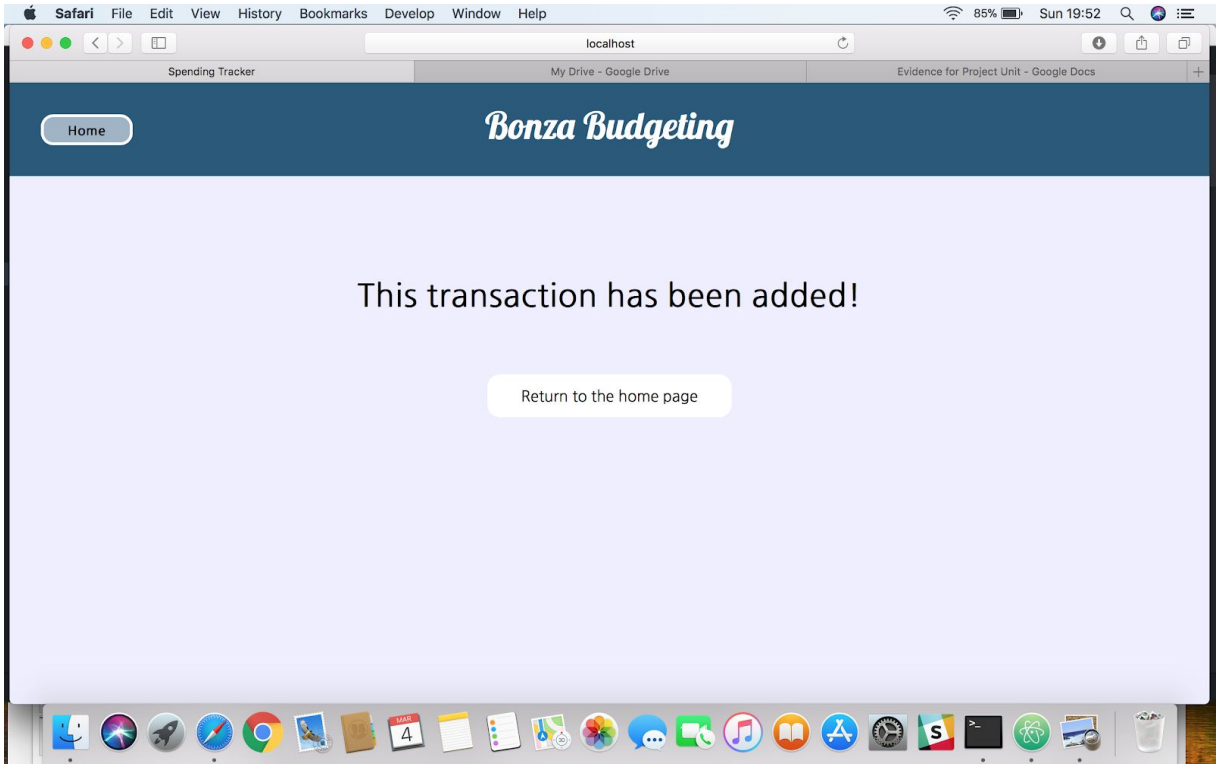
Original transactions:



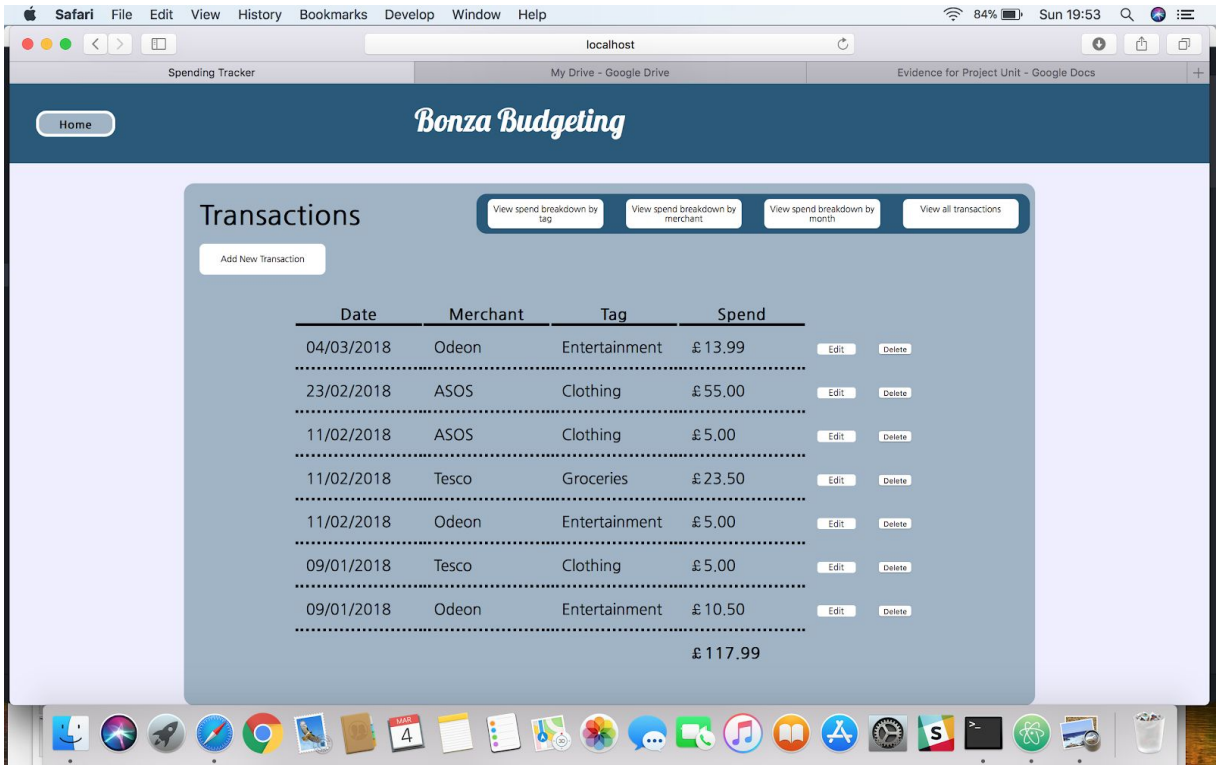
User adds new transaction:



Page is generated confirming transaction is saved:

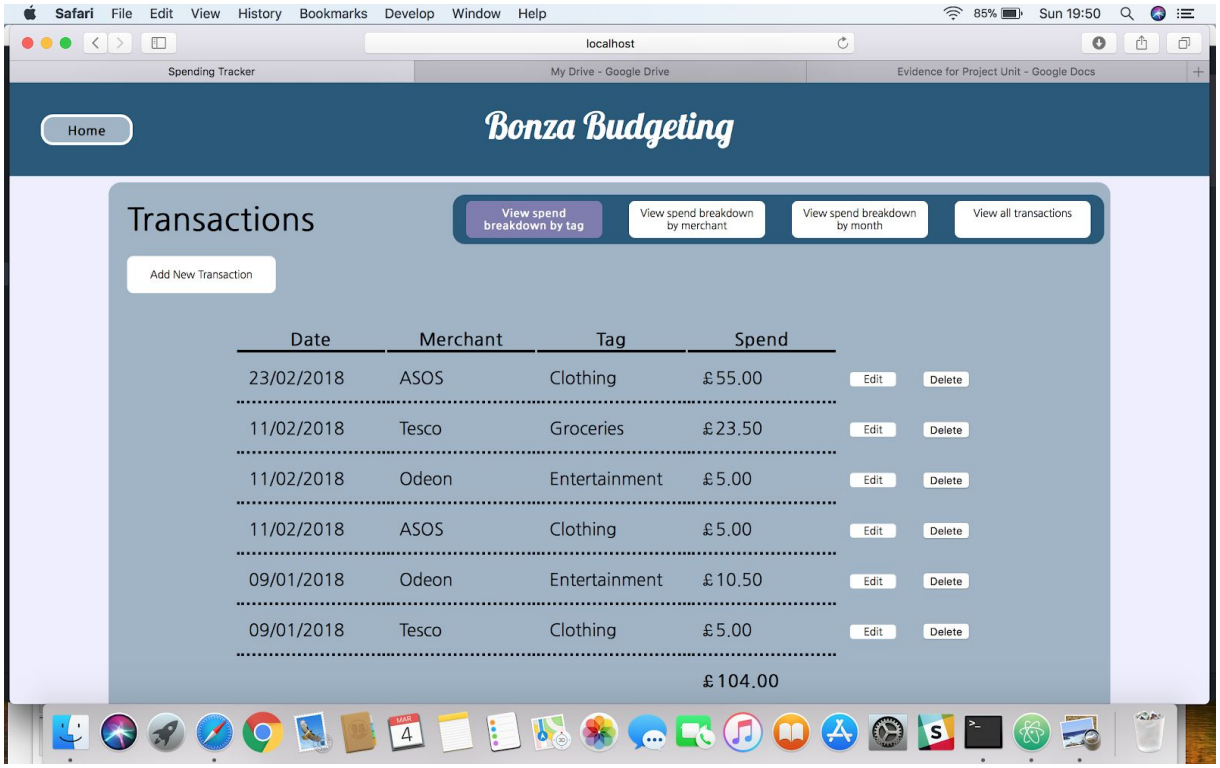


New transaction now shows in list of transactions recorded:

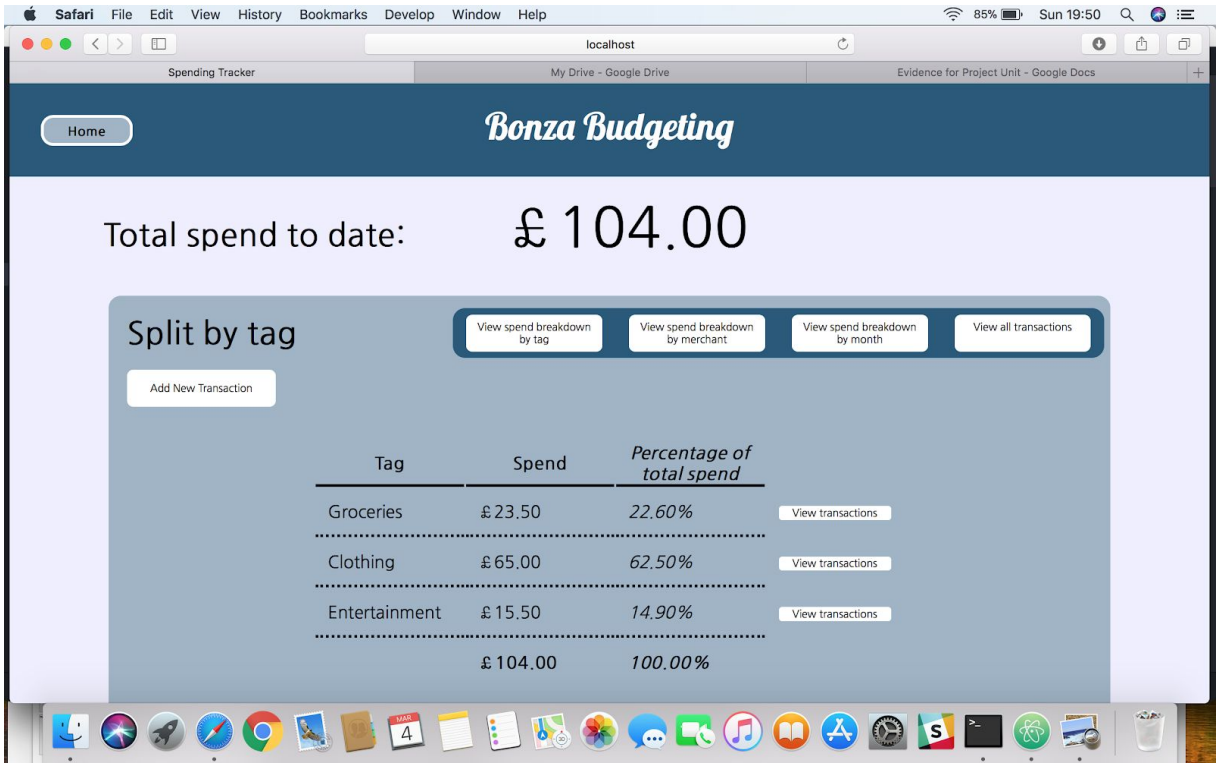


P.15 User output result

User clicks on ‘view spend breakdown by tag’ button:

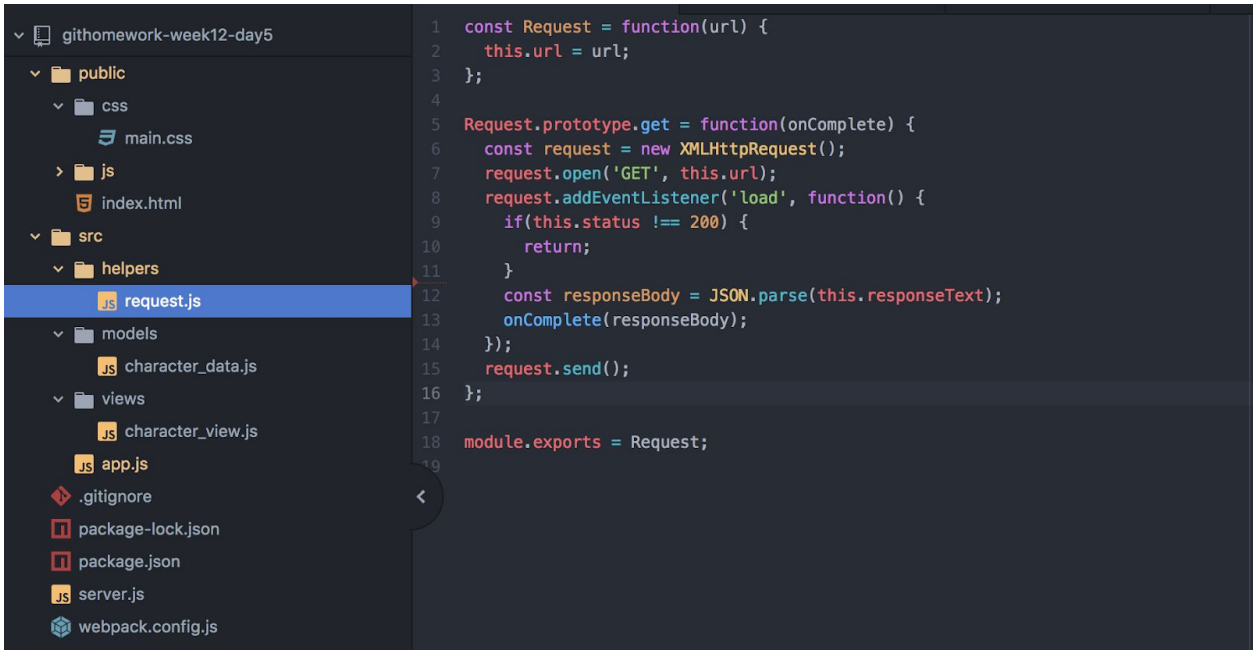


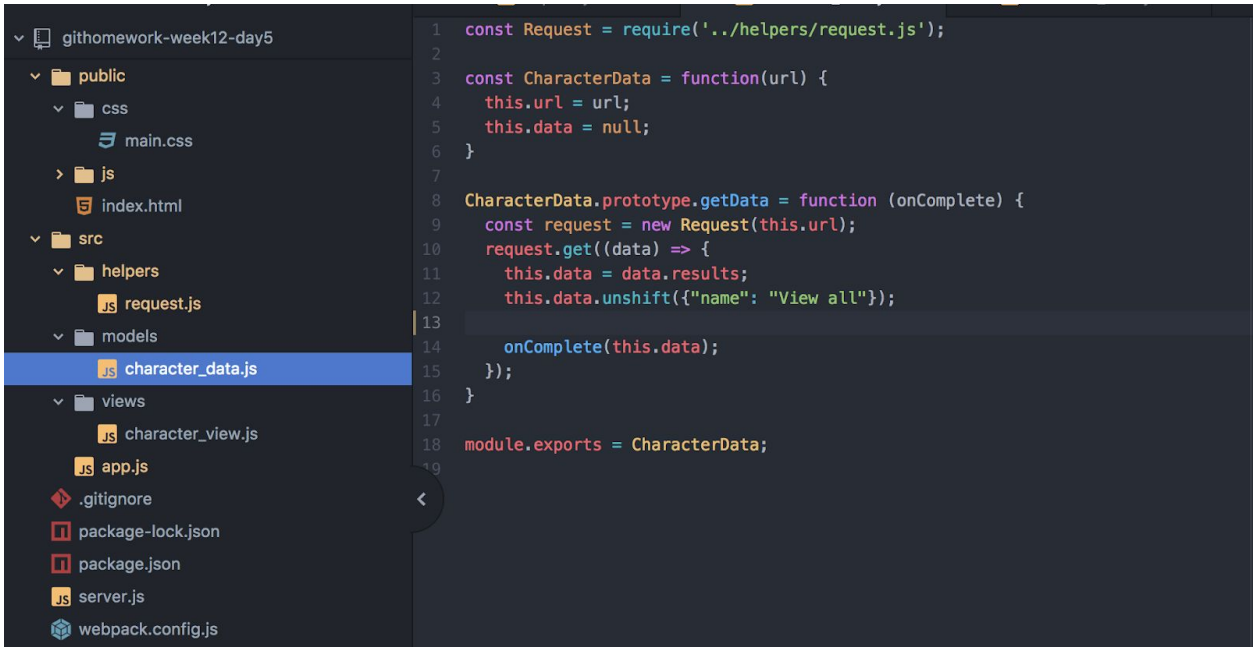
A page detailing the total spend split by tag is generated:

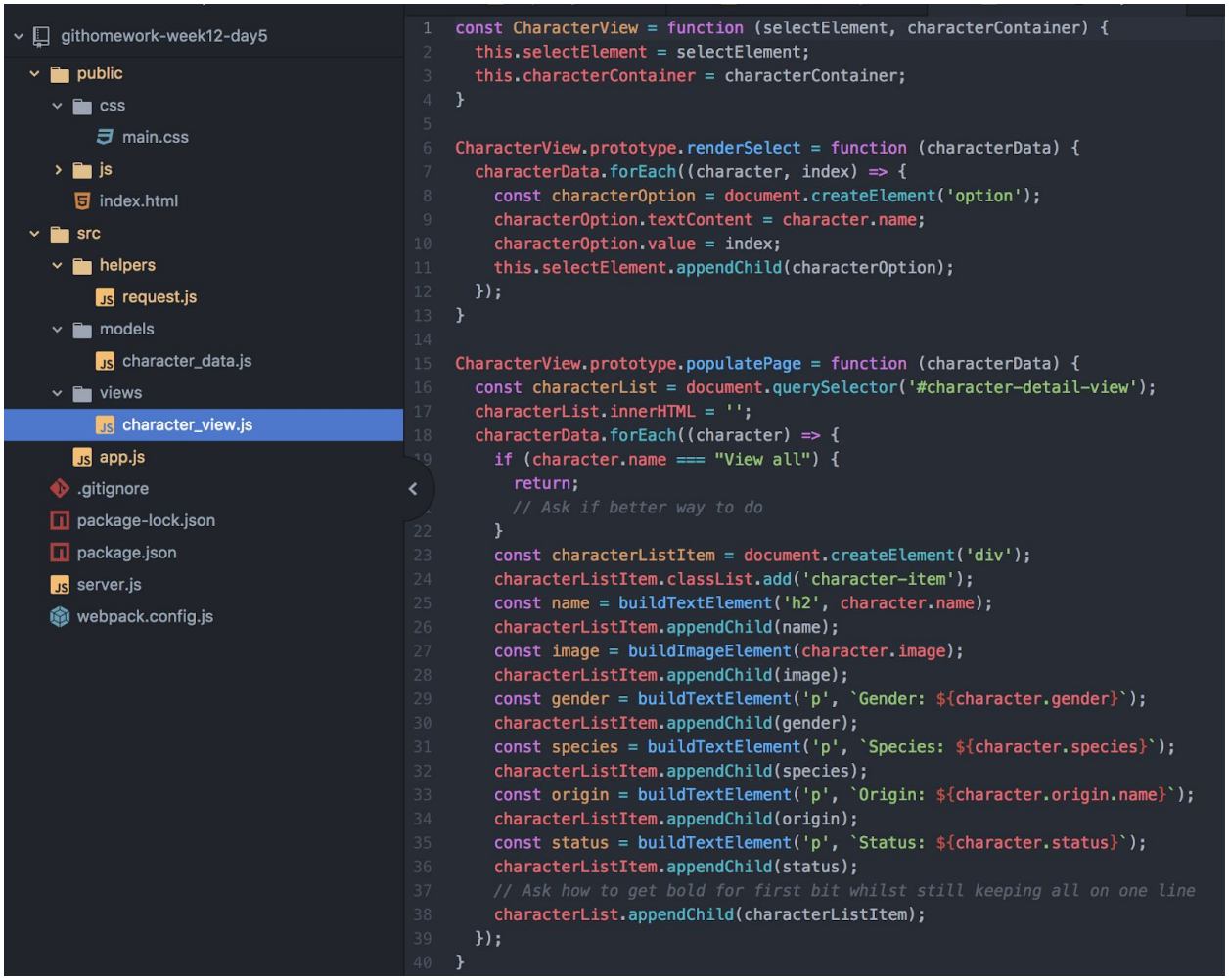


P.16 Use of API

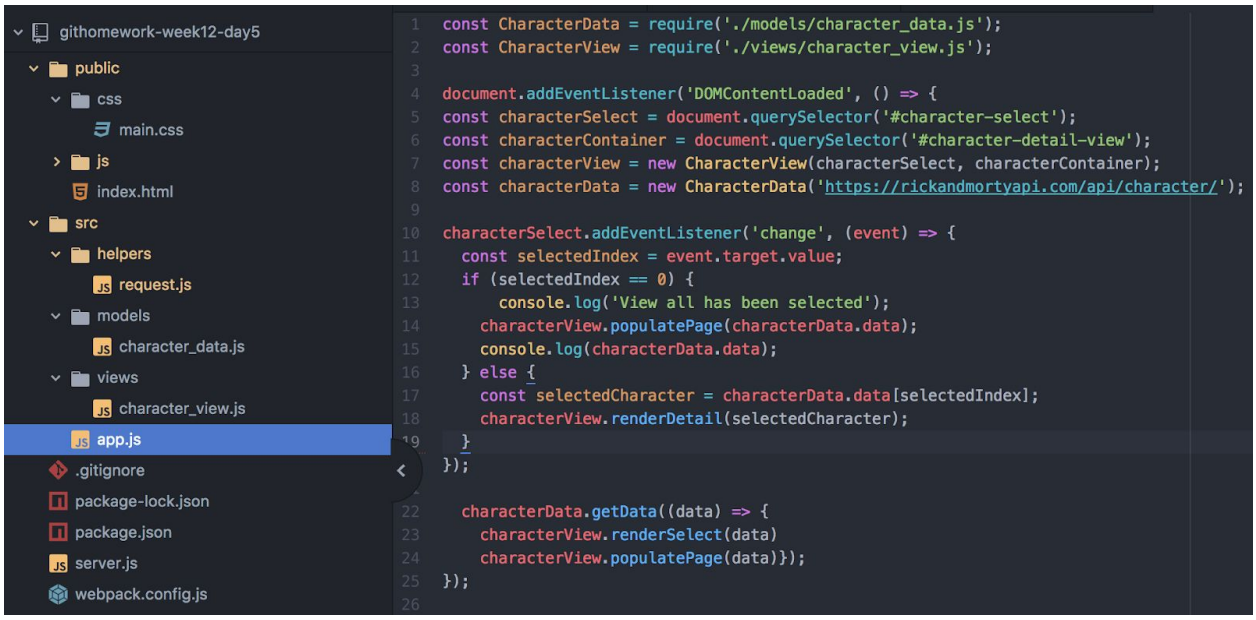
Screenshots of code:







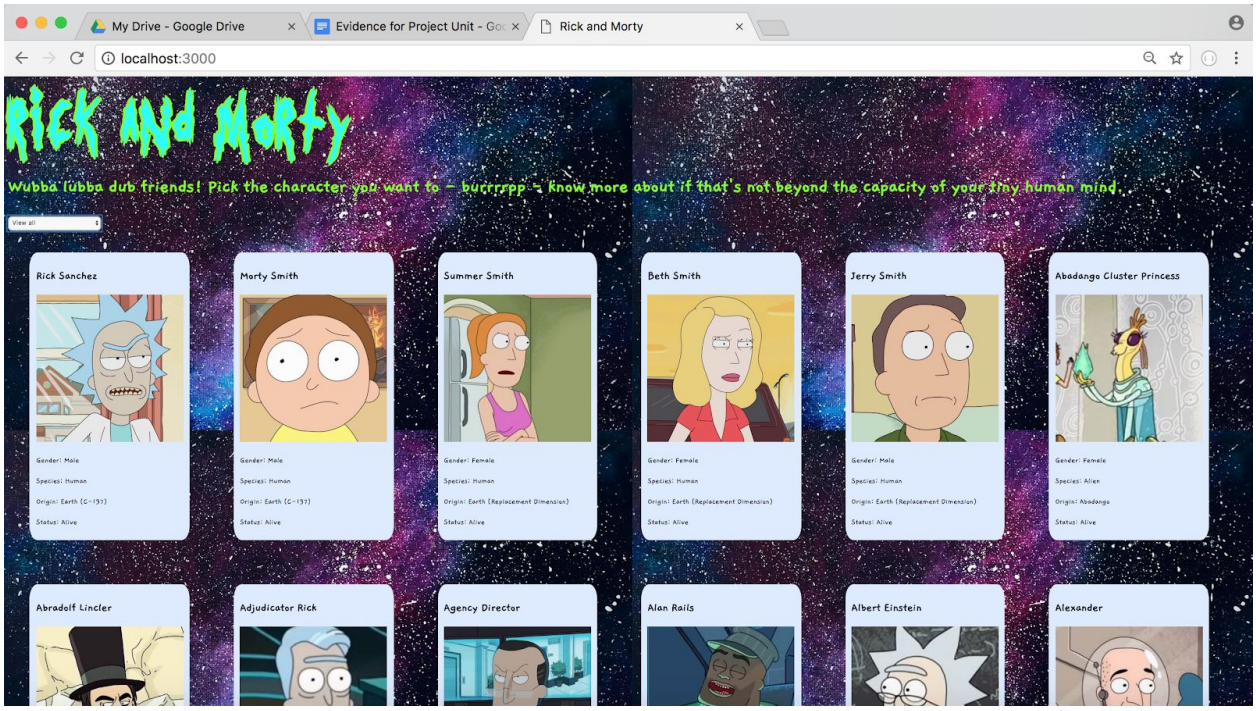

```
const CharacterView = function (selectElement, characterContainer) {
  this.selectElement = selectElement;
  this.characterContainer = characterContainer;
}

CharacterView.prototype.renderSelect = function (characterData) {
  characterData.forEach((character, index) => {
    const characterOption = document.createElement('option');
    characterOption.textContent = character.name;
    characterOption.value = index;
    this.selectElement.appendChild(characterOption);
  });
}

CharacterView.prototype.populatePage = function (characterData) {
  const characterList = document.querySelector('#character-detail-view');
  characterList.innerHTML = '';
  characterData.forEach((character) => {
    if (character.name === "View all") {
      return;
      // Ask if better way to do
    }
    const characterListItem = document.createElement('div');
    characterListItem.classList.add('character-item');
    const name = buildTextElement('h2', character.name);
    characterListItem.appendChild(name);
    const image = buildImageElement(character.image);
    characterListItem.appendChild(image);
    const gender = buildTextElement('p', `Gender: ${character.gender}`);
    characterListItem.appendChild(gender);
    const species = buildTextElement('p', `Species: ${character.species}`);
    characterListItem.appendChild(species);
    const origin = buildTextElement('p', `Origin: ${character.origin.name}`);
    characterListItem.appendChild(origin);
    const status = buildTextElement('p', `Status: ${character.status}`);
    characterListItem.appendChild(status);
    // Ask how to get bold for first bit whilst still keeping all on one line
    characterList.appendChild(characterListItem);
  });
}
```




```
const CharacterData = require('./models/character_data.js');
const CharacterView = require('./views/character_view.js');

document.addEventListener('DOMContentLoaded', () => {
const characterSelect = document.querySelector('#character-select');
const characterContainer = document.querySelector('#character-detail-view');
const characterView = new CharacterView(characterSelect, characterContainer);
const characterData = new CharacterData('https://rickandmortyapi.com/api/character/');

characterSelect.addEventListener('change', (event) => {
  const selectedIndex = event.target.value;
  if (selectedIndex == 0) {
      console.log('View all has been selected');
    characterView.populatePage(characterData.data);
    console.log(characterData.data);
  } else {
    const selectedCharacter = characterData.data[selectedIndex];
    characterView.renderDetail(selectedCharacter);
  }
});

  characterData.getData((data) => {
    characterView.renderSelect(data)
    characterView.populatePage(data)});
});
```

Screenshot of API being used whilst running:

```
➜  src git:(master) ✗ npm start

> githomework-week12-day5@1.0.0 start /Users/user/codeclan_work/week_12/day_05/githomework-week12-day5
> node server.js

Example app running on port 3000
```



P.17 Bug tracking report

Criteria	Pass / Fail	Resolution	Pass / Fail
DOM loads in browser	Fail	Corrected typo in path name for index.html file in callback function of get request for loading the page.	Pass
Audio clip plays when users start game	Fail	Called .play() on audio clip in function that is called when start button is pressed	Pass
End Go button is disabled until player has completed his / her go	Fail	'Disabled' property of End Go button is set to true initially, and is set to true as soon as it is clicked. 'Disabled' property is then changed to false once user has clicked either a card or a submit button (depending on the card) via a callback function.	Pass
Discarded cards are displayed in centre of page as soon as they are discarded.	Fail	.addToDiscard() function called whenever card is clicked on by user as part of their turn. This then triggers function to display image of card in a discard pile in the centre of the page. .addToDiscard() function is also called when user selects another player to discard their card.	Pass
End of game	Fail	Functionality added	Pass

message is displayed when cards have run out or only one player remains		to endgohandler function (which is called when end of turn button clicked), which checks the requirements for game to end and if either of them apply, displays a message to the screen announcing the winner.	

P.18 Testing

Screenshot of test code. Error is in number of dinosaurs after year function. Accidental + rather than * in line 39.

```
const Park = function() {
  this.enclosure = [];
};

Park.prototype.addDinosaur = function (dinosaur) {
  this.enclosure.push(dinosaur);
};

Park.prototype.removeDinosaursOfType = function (dinosaurType) {...};

Park.prototype.getDinosaursWithMoreThan2OffspringCount = function () {...};

Park.prototype.numberOfDinosaursAfterYear = function (year) {
  if(year < 0) {
    return;
  }
  let total = 0;

  if (year === 0) {
    return this.enclosure.length;
  } else {
    for (const dinosaur of this.enclosure) {
      let totalForDino =  1;
      for (let i = 1; i <= year; i++) {
        totalForDino = totalForDino + totalForDino + dinosaur.numberOfOffspring;
      }
      total += totalForDino;
    }
  }
  return total;
};

module.exports = Park;
```

Screenshot of tests:

```
const assert = require('assert');
const Park = require('../park.js');
const Dinosaur = require('../dinosaur.js')

describe('Park', function() {

  let park;
  let dinosaur1;
  let dinosaur2;
  let dinosaur3;
  let dinosaur4;

  beforeEach(function() {
    park = new Park();
    dinosaur1 = new Dinosaur("Tyrannosaurus", 3);
    dinosaur2 = new Dinosaur("Tyrannosaurus", 2);
    dinosaur3 = new Dinosaur("Tyrannosaurus", 3);
    dinosaur4 = new Dinosaur("Velociraptor", 3);
  });

  it('should start empty', function() {...});

  it('should add dinosaur', function() {...});

  it('should remove all dinosaurs of a particular type where type is in enclosure', function() {...});

  it('should not remove anything if type is not in enclosure', function() {...});

  it('should return all dinosaurs with offspring count of more than 2', function() {...});
```

```

  describe('Future projections', function() {

    it('should calculate number of dinosaurs after a particular number of years', function() {
      park.addDinosaur(dinosaur1)
      const result1 = park.numberOfDinosaursAfterYear(1);
      assert.strictEqual(result1, 4);
      const result2 = park.numberOfDinosaursAfterYear(2);
      assert.strictEqual(result2, 16);
      park.addDinosaur(dinosaur2)
      const result3 = park.numberOfDinosaursAfterYear(2);
      assert.strictEqual(result3, 25);
      const result4 = park.numberOfDinosaursAfterYear(0);
      assert.strictEqual(result4, 2);
      const result5 = park.numberOfDinosaursAfterYear(-5);
      assert.strictEqual(result5, undefined);
    });

  });
});
```

Screenshot of test failing:

```
➜  githomework-week11-day2 git:(master) ✗ npm run test

> githomework-week11-day2@1.0.0 test /Users/user/codeclan_work/week_11/day_02/githomework-week11-day2
> mocha specs



  Dinosaur
    ✓ should have a type
    ✓ should have a number of offspring per year

  Park
    ✓ should start empty
    ✓ should add dinosaur
    ✓ should remove all dinosaurs of a particular type where type is in enclosure
    ✓ should not remove anything if type is not in enclosure
    ✓ should return all dinosaurs with offspring count of more than 2
    Future projections
      1) should calculate number of dinosaurs after a particular number of years


  7 passing (12ms)
  1 failing

  1) Park
       Future projections
         should calculate number of dinosaurs after a particular number of years:

      AssertionError [ERR_ASSERTION]: 5 === 4
      + expected - actual

      -5
      +4
```

Screenshot of corrected code:

```
27    Park.prototype.numberOfDinosaursAfterYear = function (year) {
28      if(year < 0) {
29        return;
30      }
31      let total = 0;
32
33      if (year === 0) {
34        return this.enclosure.length;
35      } else {
36        for (const dinosaur of this.enclosure) {
37          let totalForDino =  1;
38          for (let i = 1; i <= year; i++) {
39            totalForDino = totalForDino + totalForDino * dinosaur.numberOfOffspring;
40          }
41          total += totalForDino;
42        }
43      }
44      return total;
45    };
46
47    module.exports = Park;
48
```

Screenshot of tests passing:

```
➜  githomework-week11-day2 git:(master) ✗ npm run test

> githomework-week11-day2@1.0.0 test /Users/user/codeclan_work/week_11/day_02/githomework-week11-day2
> mocha specs



  Dinosaur
    ✓ should have a type
    ✓ should have a number of offspring per year

  Park
    ✓ should start empty
    ✓ should add dinosaur
    ✓ should remove all dinosaurs of a particular type where type is in enclosure
    ✓ should not remove anything if type is not in enclosure
    ✓ should return all dinosaurs with offspring count of more than 2
    Future projections
      ✓ should calculate number of dinosaurs after a particular number of years


  8 passing (10ms)
```