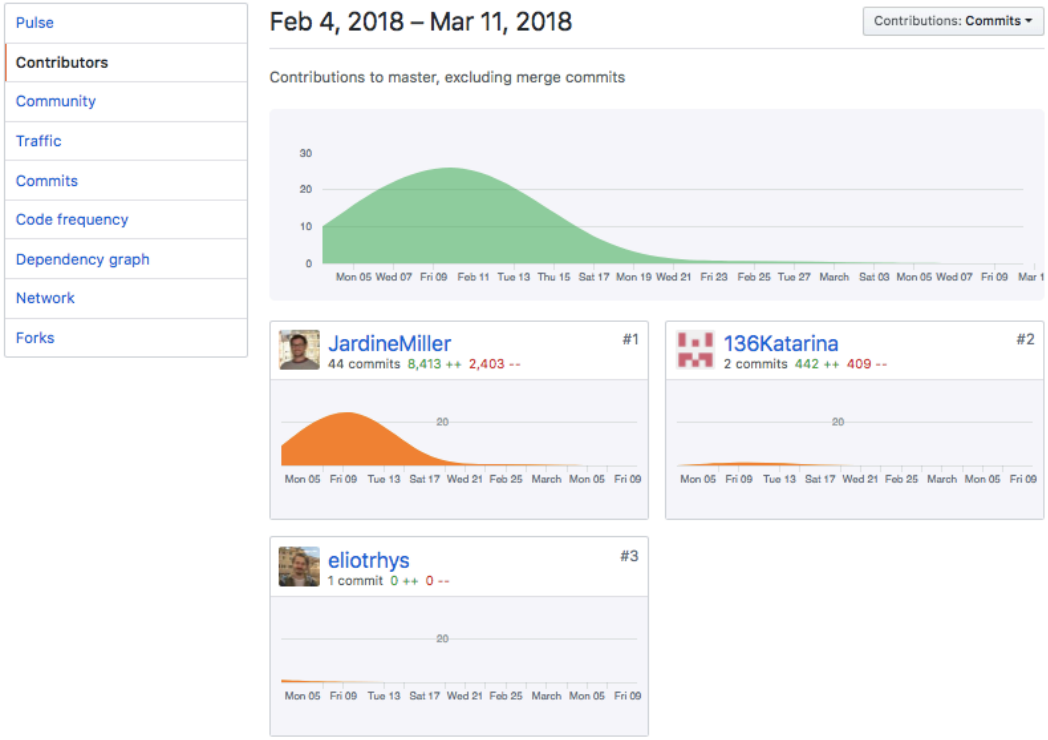
Evidence for Project Unit

Katarina Zemplenyiova
11/March/2018
Cohort 17

P- 1 Github Contributors page



P - 2 Project Brief

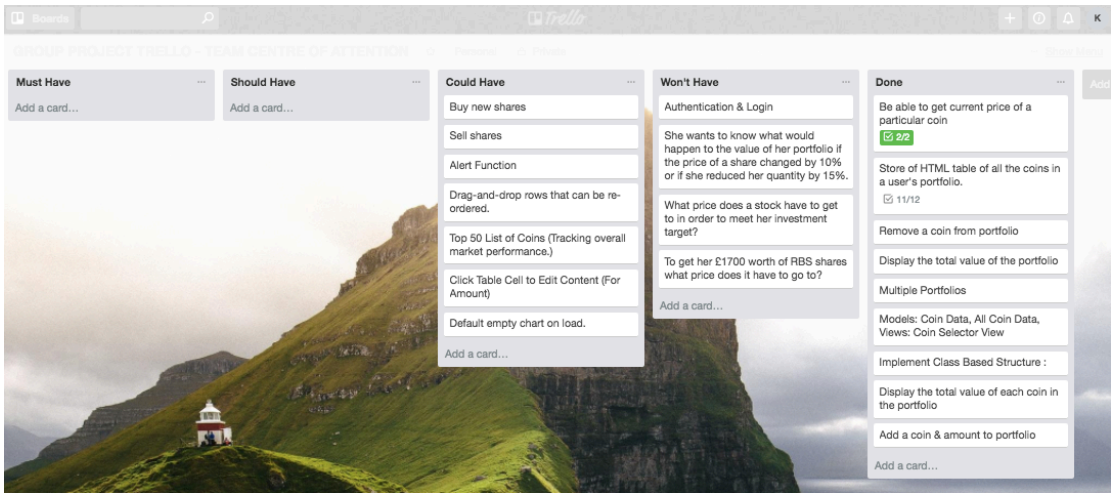
Shares App

A local trader has come to you with a portfolio of shares. She wants to be able to analyse it more effectively. She has a small sample data set to give you and would like you to build a minimal viable product (MVP) that uses the data to display her portfolio in useful ways so that she can make better decisions.

MVP

- View total current value
- View individual and total performance trends
- Retrieve a list of share prices from an external API and allow the user to add shares to her portfolio
- Provide a chart of the current values in her portfolio

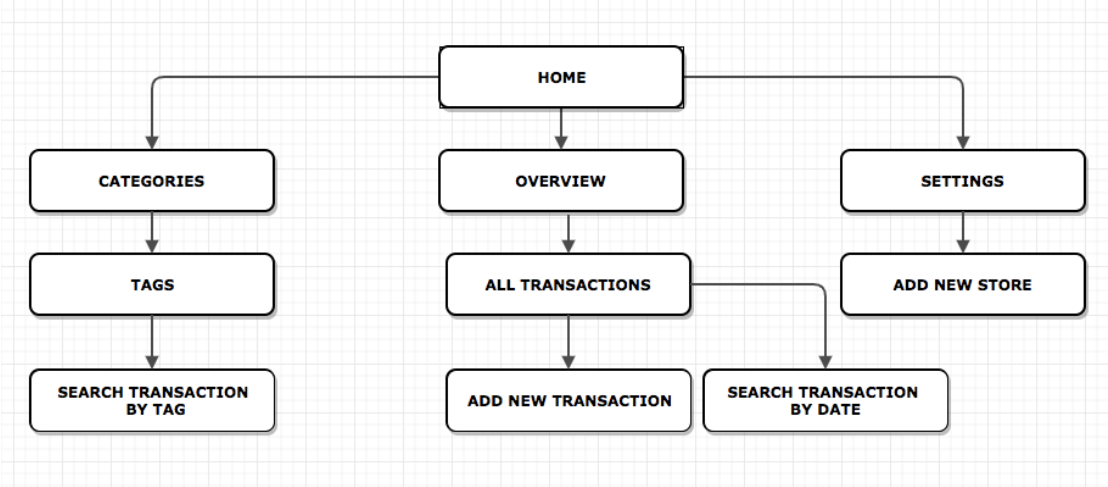
P -3 Use of Trello



P -4 Acceptance Criteria

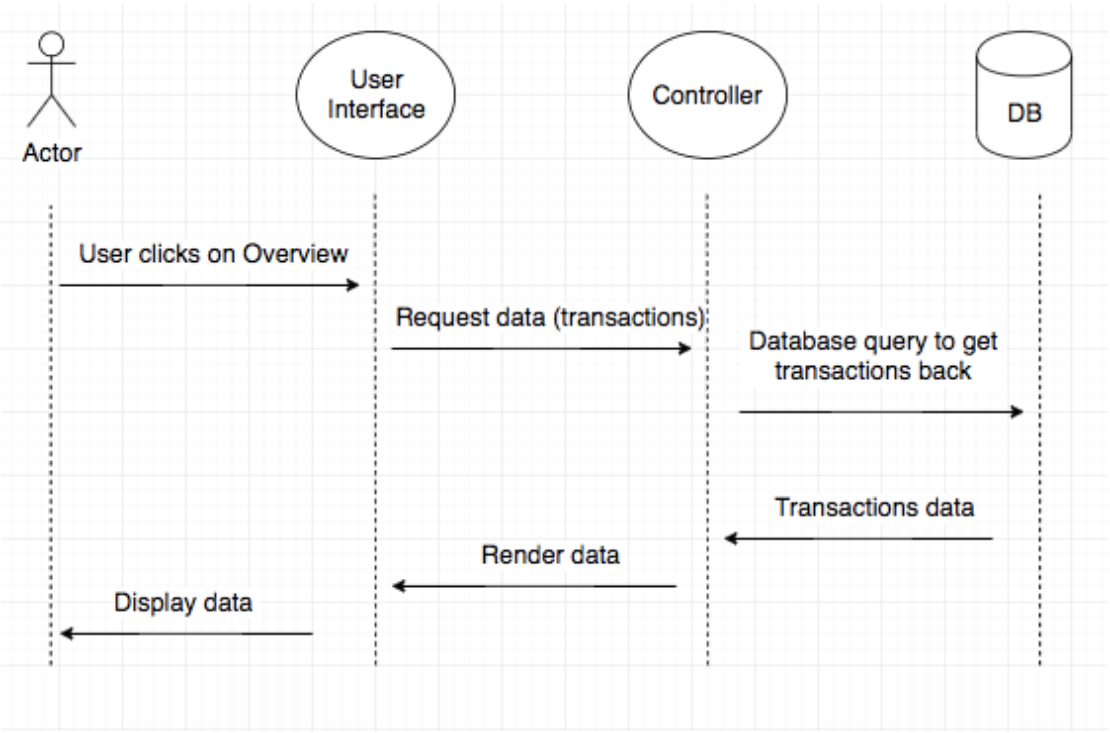
Acceptance Criteria	Expected Result	Pass/Fail
User is able to search for a transaction by tag or date	Transactions that match the tag or date are displayed in a new table	PASS
User can add and save a new transaction	Transaction is saved into database when “Add New” button is clicked and displayed within the transaction list on the main page	PASS
User can see a list of the stores and a new one	List of stores can be adjusted in Settings by adding a new store that will be displayed in dropdown list when adding new transaction	PASS
User can see list of transaction tags	All transaction tags are displayed when Category section is clicked on	PASS

P – 5 User sitemap

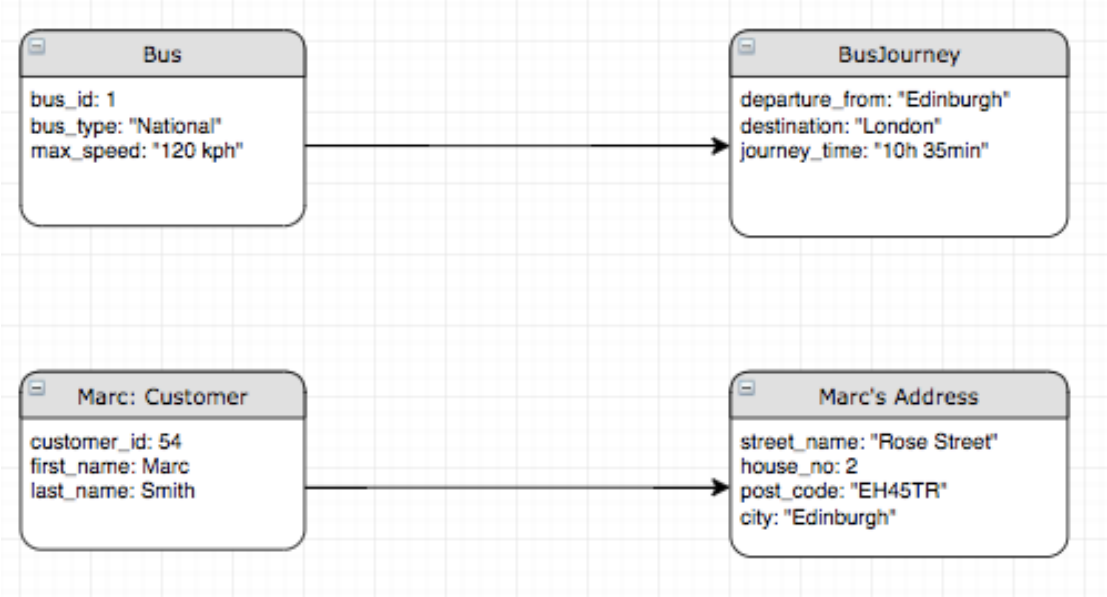


P – 6 Wireframes designs

P -7 System interactions diagrams



P -8 Two Object Diagrams



P- 9 Algorithms

```ruby
def self.tag_type(id)
    sql = "SELECT *
    FROM transactions
    WHERE tag_id = $1"
    values = [id]
    results = SqlRunner.run(sql, values)
    return results.map { |transaction| Transaction.new(transaction) }
end
```

This algorithm had the purpose of finding transaction by selected tag. The result of iteration is returned in a new array. All found transactions of the same tag are displayed in a new list.

```
Hero.prototype.orderByUrgency = function(){
  return this.tasks.sort(function(a,b){
    return a.urgencyLevel - b.urgencyLevel;
  })
}
```

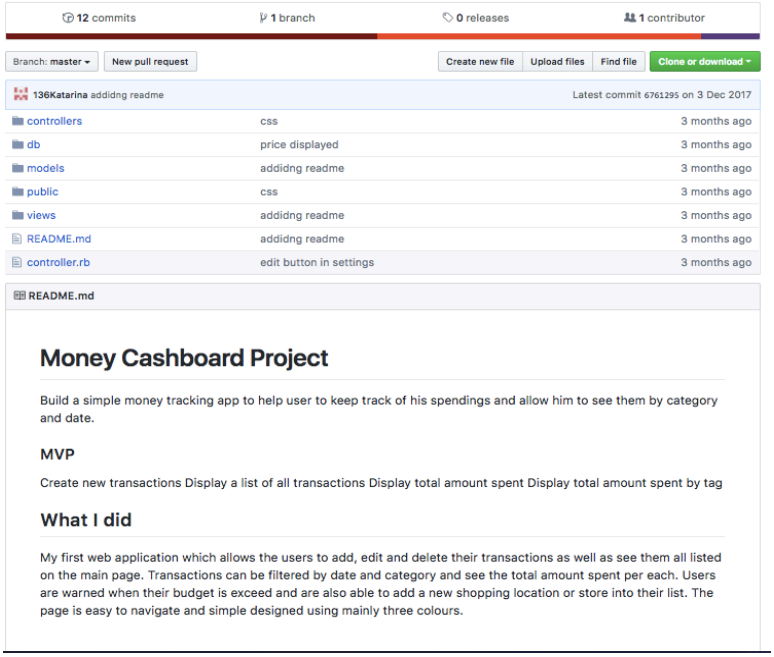
This algorithm had the purpose to order Hero's tasks by their urgency level. The sort function compares whether a (task urgency level) is smaller than, equal to or larger than value of b (another task urgency level) and returns a new result array with correctly position tasks in descending order.

P – 10 Pseudo code for function

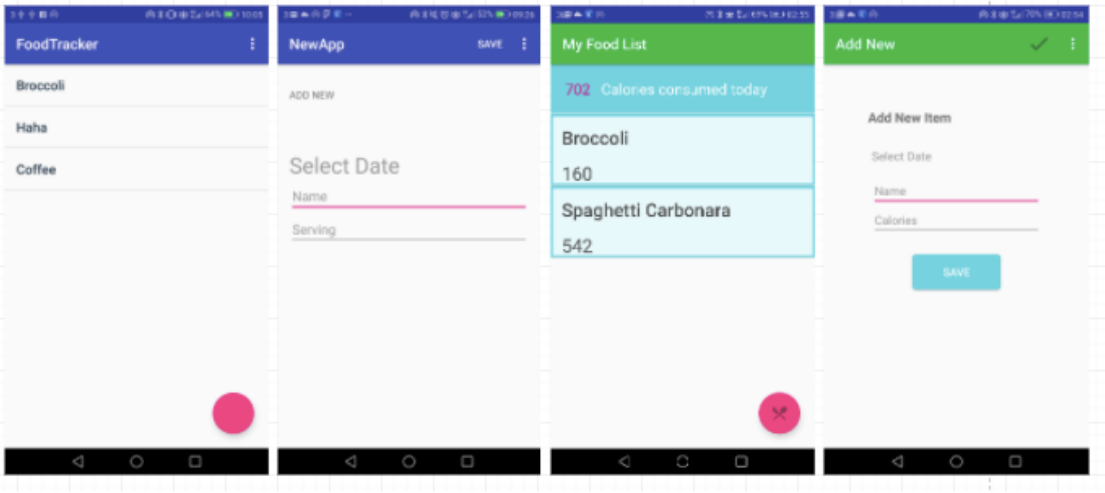
```
//create function
// if damage is less than zero, decrease health bar by damage
//multiply damage by value of armour to calculate damage result
// decrease health bar by result of calculated damage
//end function
```

P – 11 Github link to one of your projects

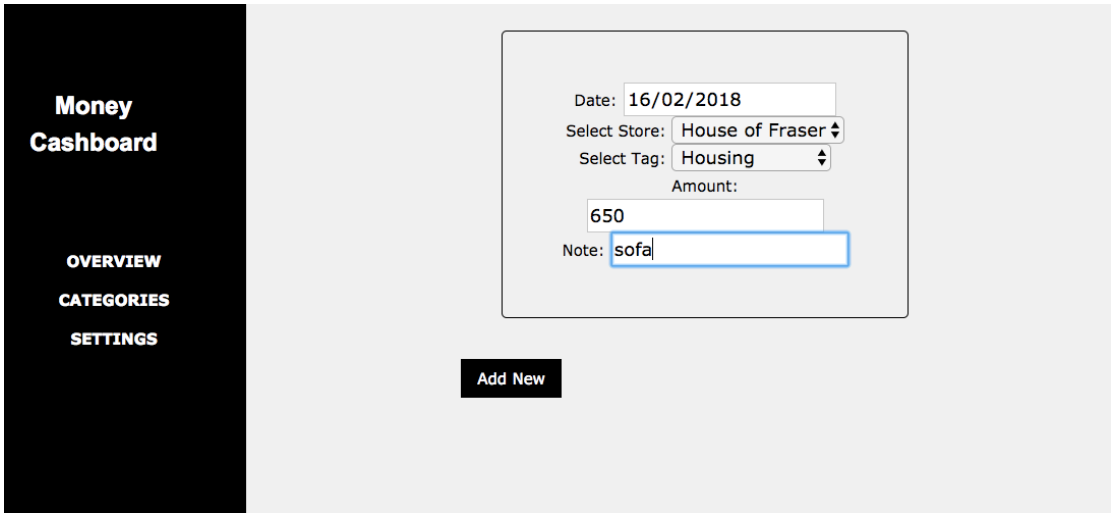
https://github.com/136Katarina/MoneyCashboard-

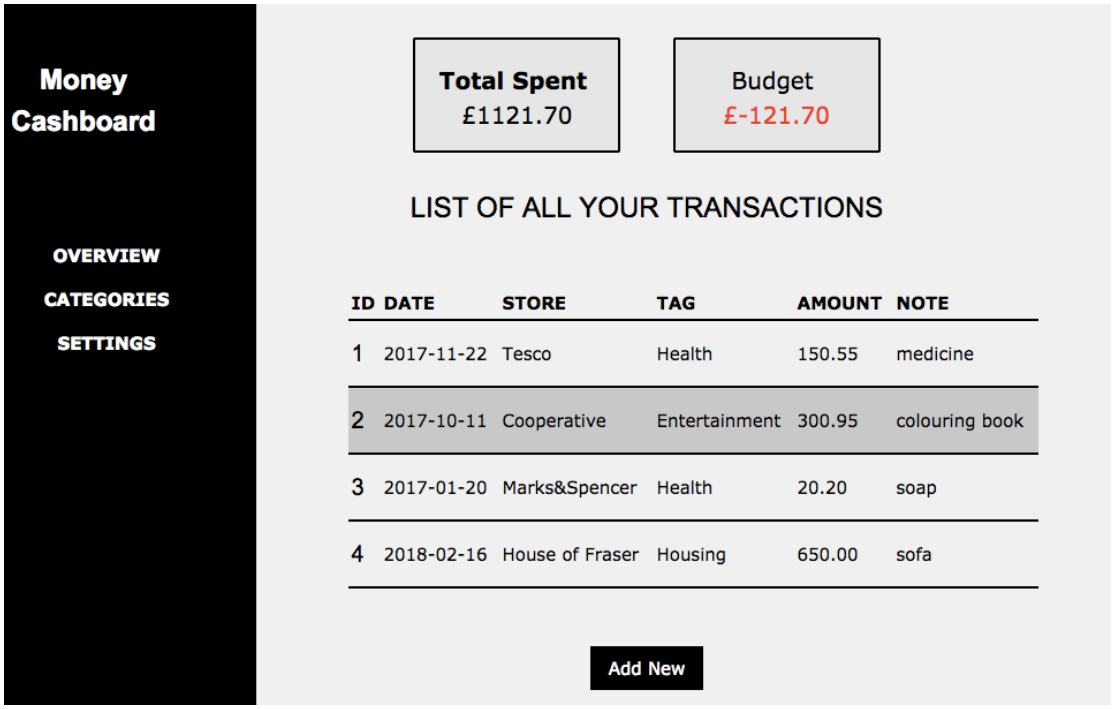


P – 12 Screenshot of your planning and the different stages of development to show changes.

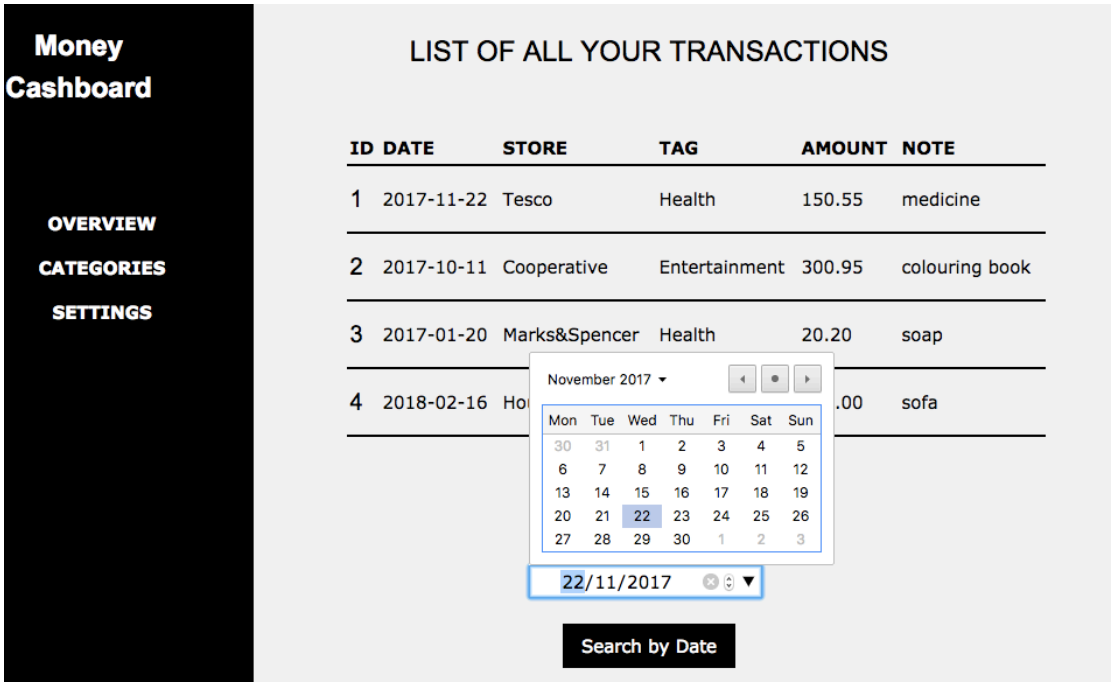


P – 13 User Input

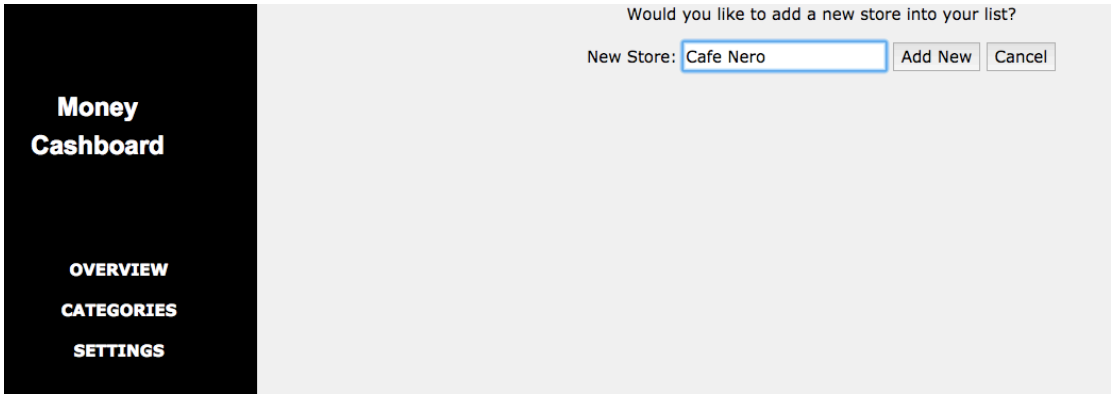


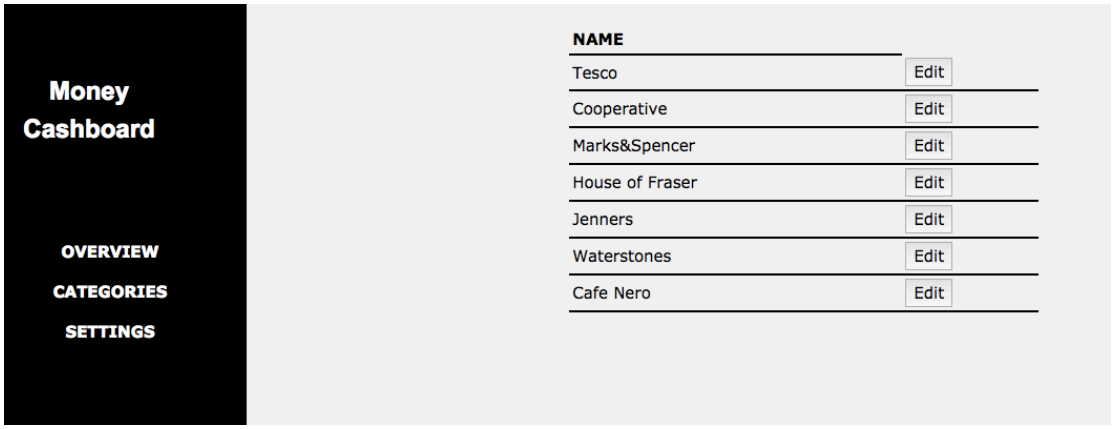


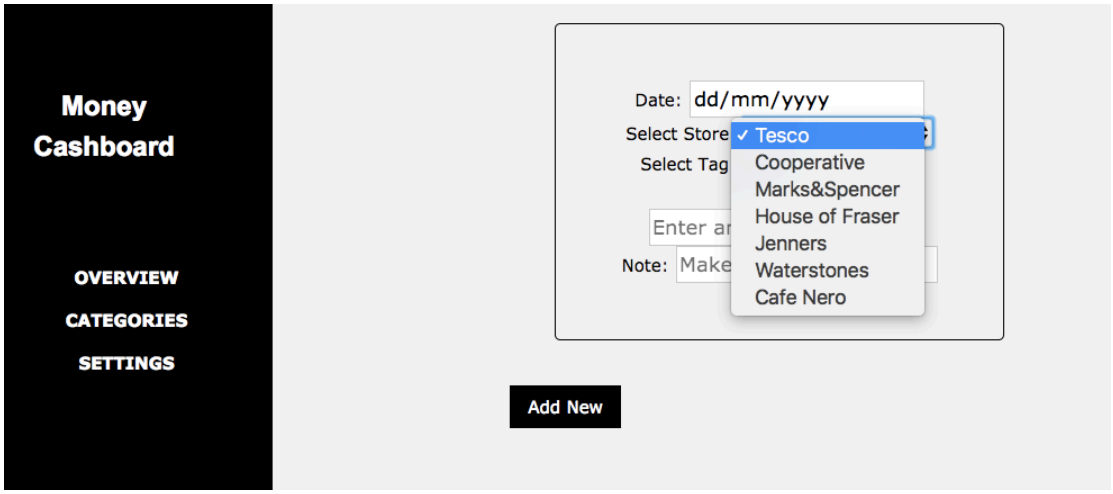
P – 14 Interaction with data persistence



P – 15 User output result


Money Cashboard
OVERVIEW
CATEGORIES
SETTINGS
Would you like to add a new store into your list?
New Store: Cafe Nero
Add New
Cancel


Money Cashboard
OVERVIEW
CATEGORIES
SETTINGS
NAME
Tesco Edit
Cooperative Edit
Marks&Spencer Edit
House of Fraser Edit
Jenners Edit
Waterstones Edit
Cafe Nero Edit

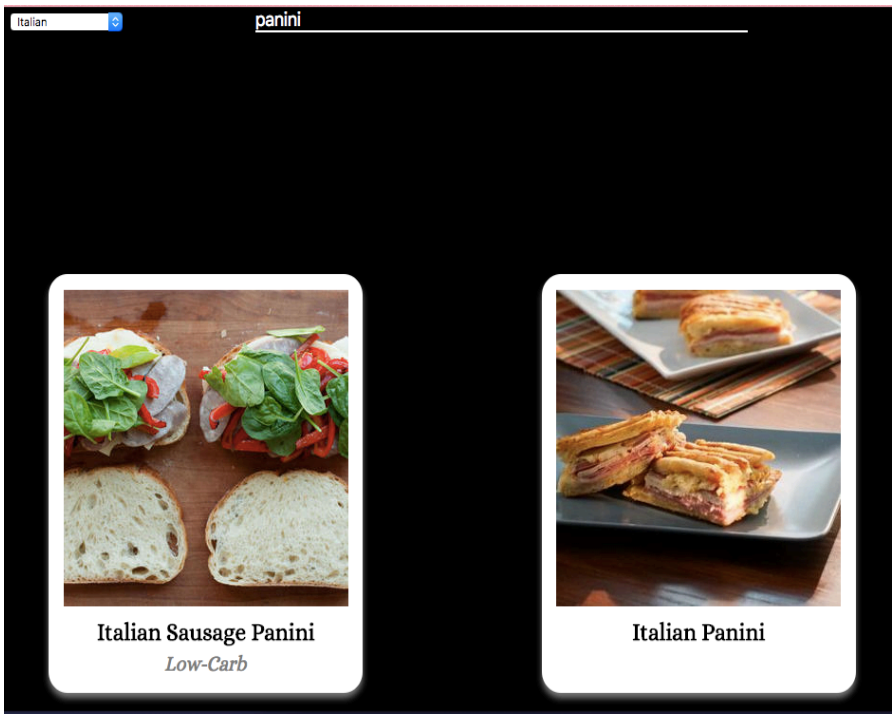

Money Cashboard
OVERVIEW
CATEGORIES
SETTINGS
Date: dd/mm/yyyy
Select Store
Select Tag
Note:
Tesco
Cooperative
Marks&Spencer
House of Fraser
Jenners
Waterstones
Cafe Nero
Add New

P – 16 Use of API

```
  componentDidMount(){
    const url = "https://api.edamam.com/search?q=" + this.state.recipeType +
    "&from=0&to=99&app_id=1492933c&app_key=f8f8215e26162f1c73a195557a055aa6";
    const request = new XMLHttpRequest();
  request.open('GET', url);
  request.addEventListener('load', () => {
    if (request.status !== 200) return;
    const jsonString = request.responseText;
    const data = JSON.parse(jsonString);
    console.log(data.hits);
    this.setState({
      recipes: data.hits
    })
  })
  request.send();
}


handleSelectChange(recipeType){
  const url = "https://api.edamam.com/search?q=" + recipeType +
  "&from=0&to=99&app_id=1492933c&app_key=f8f8215e26162f1c73a195557a055aa6";
  const request = new XMLHttpRequest();
request.open('GET', url);
request.addEventListener('load', () => {
  if (request.status !== 200) return;
  const jsonString = request.responseText;
  const data = JSON.parse(jsonString);
  console.log(data.hits);
  this.setState({
    recipes: data.hits
  })
})
request.send();

  this.setState({recipeType: recipeType});
}
```



P – 17 Bug tracking report showing the errors diagnosed and corrected.

P -18 Testing your program

```
Hero
  ✓ has a favourite food
  ✓ has a name
  ✓ has a health
  ✓ has ability to talk
  ✓ can eat food
  1) increase health by favourite
  ✓ it can add tasks to do
  ✓ it can sort tasks by descending value of reward
  ✓ it can sort tasks by urgency level
  ✓ can check is the task is completed
  ✓ eats poisonous food

Rat
  2) rat can touch food

Task
  ✓ has a difficulty level
  ✓ has an urgency level
  ✓ has a reward
  ✓ marked as completed or not


16 passing (28ms)
2 failing

1) Hero
     increase health by favourite:

    AssertionError [ERR_ASSERTION]: 1 === 115
    + expected - actual

    -1
    +115

    at Context.<anonymous> (specs/hero_spec.js:50:12)

2) Rat
     rat can touch food:

    AssertionError [ERR_ASSERTION]: true === false
    + expected - actual

    -true
    +false

    at Context.<anonymous> (specs/rat_spec.js:17:12)
```

```
const assert = require('assert');
const Hero = require('../hero.js');
const Task = require('../task.js');
const Food = require('../food.js');
const Rat = require('../rat.js');

describe('Hero', function(){
  var hero;
  var task1;
  var task2;
  var task3;
  var food1;
  var food2;
  var rat1;

  beforeEach(function(){
    food1 = new Food("melon", 10);
    food2 = new Food("tomato soup", 5);
    hero = new Hero("Katarina", 100,food1);
    task1 = new Task("difficult",1,30);
    task2 = new Task("average",2, 20);
    task3 = new Task("easy", 3, 10);
    rat1 = new Rat("Daisy");
  })

  it('has a favourite food', function(){
    assert.strictEqual(food1, hero.favouriteFood);
  })

  it('has a name', function(){
    assert.strictEqual( hero.canTalk(), "My name is Katarina");
  })

  it('has a health', function(){
    assert.strictEqual(100, hero.health);
  })

  it('has ability to talk', function(){
    assert.strictEqual('My name is Katarina', hero.canTalk());
  })


  it('can eat food', function(){
    hero.canEat(food2);
    assert.strictEqual(105, hero.health);
  })

  it('increase health by favourite', function(){
    hero.canEat(food1);
    assert.strictEqual(1, hero.health);
  })
```

```
const assert = require('assert');
const Rat = require('../rat.js');
const Food = require('../food.js');

describe('Rat', function(){
  var rat;
  var food1;


  beforeEach(function(){
    rat = new Rat("Daisy");
    food1= new Food("melon", 10)
  })

  it('rat can touch food', function(){
    rat.canTouchFood(food1);
    assert.strictEqual(food1.poisoned, false);
  })

})
```

Errors corrected:

```
➜  Hero git:(master) ✗ npm test

> hero@1.0.0 test /Users/katarinazemplenyiova/codeclan_work/working/week_11/day_5/Hero
> mocha specs


  Food
    ✓ has a name
    ✓ has a value

  Hero
    ✓ has a favourite food
    ✓ has a name
    ✓ has a health
    ✓ has ability to talk
    ✓ can eat food
    ✓ increase health by favourite
    ✓ it can add tasks to do
    ✓ it can sort tasks by descending value of reward
    ✓ it can sort tasks by urgency level
    ✓ can check is the task is completed
    ✓ eats poisonous food

  Rat
    ✓ rat can touch food

  Task
    ✓ has a difficulty level
    ✓ has an urgency level
    ✓ has a reward
    ✓ marked as completed or not


  18 passing (27ms)
```

```
const assert = require('assert');
const Rat = require('../rat.js');
const Food = require('../food.js');

describe('Rat', function(){
  var rat;
  var food1;


  beforeEach(function(){
    rat = new Rat("Daisy");
    food1= new Food("melon", 10)
  })

  it('rat can touch food', function(){
    rat.canTouchFood(food1);
    assert.strictEqual(food1.poisoned, true);
  })

})
```

```
const assert = require('assert');
const Hero = require('../hero.js');
const Task = require('../task.js');
const Food = require('../food.js');
const Rat = require('../rat.js');

describe('Hero', function(){
  var hero;
  var task1;
  var task2;
  var task3;
  var food1;
  var food2;
  var rat1;

  beforeEach(function(){
    food1 = new Food("melon", 10);
    food2 = new Food("tomato soup", 5);
    hero = new Hero("Katarina", 100,food1);
    task1 = new Task("difficult",1,30);
    task2 = new Task("average",2, 20);
    task3 = new Task("easy", 3, 10);
    rat1 = new Rat("Daisy");
  })

  it('has a favourite food', function(){
    assert.strictEqual(food1, hero.favouriteFood);
  })

  it('has a name', function(){
    assert.strictEqual( hero.canTalk(), "My name is Katarina");
  })

  it('has a health', function(){
    assert.strictEqual(100, hero.health);
  })

  it('has ability to talk', function(){
    assert.strictEqual('My name is Katarina', hero.canTalk());
  })


  it('can eat food', function(){
    hero.canEat(food2);
    assert.strictEqual(105, hero.health);
  })

  it('increase health by favourite', function(){
    hero.canEat(food1);
    assert.strictEqual(115, hero.health);
  })
```