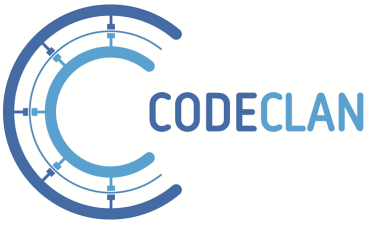
Evidence Gathering Document for SQA Level 8 Professional Developer Award.

This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

Each point that required details the Assessment Criteria (What you have to show) along with a brief description of the kind of things you should be showing.

Please fill in each point with screenshot or diagram and description.



Week 2

Unit	Ref	Evidence	
I&T	I.T.5	Demonstrate the use of an array in a program. Take screenshots of: *An array in a program *A function that uses the array *The result of the function running	

Paste Screenshot here

River.rb

```ruby
class River
  attr_reader(:name)

  def initialize(name)
    @name = name
    @fishes = []
  end

  def add_fish(fish)
    @fishes << fish
  end
```

```ruby
def test_add_fishes
  p @river.fish_count()
  @river.add_fish(@fish1)
  @river.add_fish(@fish2)
  @river.add_fish(@fish3)
  p @river.fish_count()
  assert_equal(3, @river.fish_count())
end
```

```
➜ homework-bears-river-fish git:(master) ruby specs/River_spec.rb
Run options: --seed 33464

# Running:

.0
3
...

Finished in 0.001317s, 3037.2058 runs/s, 3037.2058 assertions/s.

4 runs, 4 assertions, 0 failures, 0 errors, 0 skips
```

Description here

the fishes array is declared with @fishes=[]
3 fish are added to the array with @fishes.push()
the test shows that 3 fish are now in the array

Unit	Ref	Evidence	
I&T	I.T.6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running	

Paste Screenshot here

```ruby
@pet_shop = {
    pets: [
      {
        name: "Arthur",
        pet_type: :dog,
        breed: "Husky",
        price: 900,
      },
      {
        name: "Sir Percy",
        pet_type: :cat,
        breed: "British Shorthair",
        price: 500
      },
      {
        name: "King Bagdemagus",
        pet_type: :cat,
        breed: "British Shorthair",
        price: 500
      }
    ]
```

```ruby
def find_pet_by_name(shop, pet_name)
  for pet in shop[:pets]
    return pet if pet[:name] == pet_name
  end

  return nil
end
```

```ruby
def test_find_pet_by_name__returns_pet
  p @pet_shop
  pet = find_pet_by_name(@pet_shop, "Arthur")
  p pet
  assert_equal("Arthur", pet[:name])
end
```

```
➜ weekend_homework_start_point git:(master) ✗ ruby specs/pet_shop_spec.rb
Run options: --seed 53216

# Running:

{:pets=>[{:name=>"Arthur", :pet_type=>:dog, :breed=>"Husky", :price=>900}, {:name=>"Sir Percy", :pet_type=>:cat,
d=>"British Shorthair", :price=>500}, {:name=>"King Bagdemagus", :pet_type=>:cat, :breed=>"British Shorthair", :p
>500}], :admin=>{:total_cash=>1000, :pets_sold=>0}, :name=>"Camelot of Pets"}
{:name=>"Arthur", :pet_type=>:dog, :breed=>"Husky", :price=>900}
.

Finished in 0.009834s, 101.6880 runs/s, 101.6880 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

Description here

Week 3

Unit	Ref	Evidence	
I&T	I.T.3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running	

Paste Screenshot here

HTML

CSS (SCSS)

JS

```
const places = [
  {
    name: "Edinburgh",
    population: 1000
  },{
    name: "Falkirk",
    population: 100
  },{
    name: "Glasgow",
    population: 2000
  }
];

function findPlace(name, array) {
  for (const element of array) {
    if (element.name === name) return element;
  }

  return false;
}

console.log(findPlace("Falkirk", places));
```

Console

```
Object {
  name: "Falkirk",
  population: 100
}
```

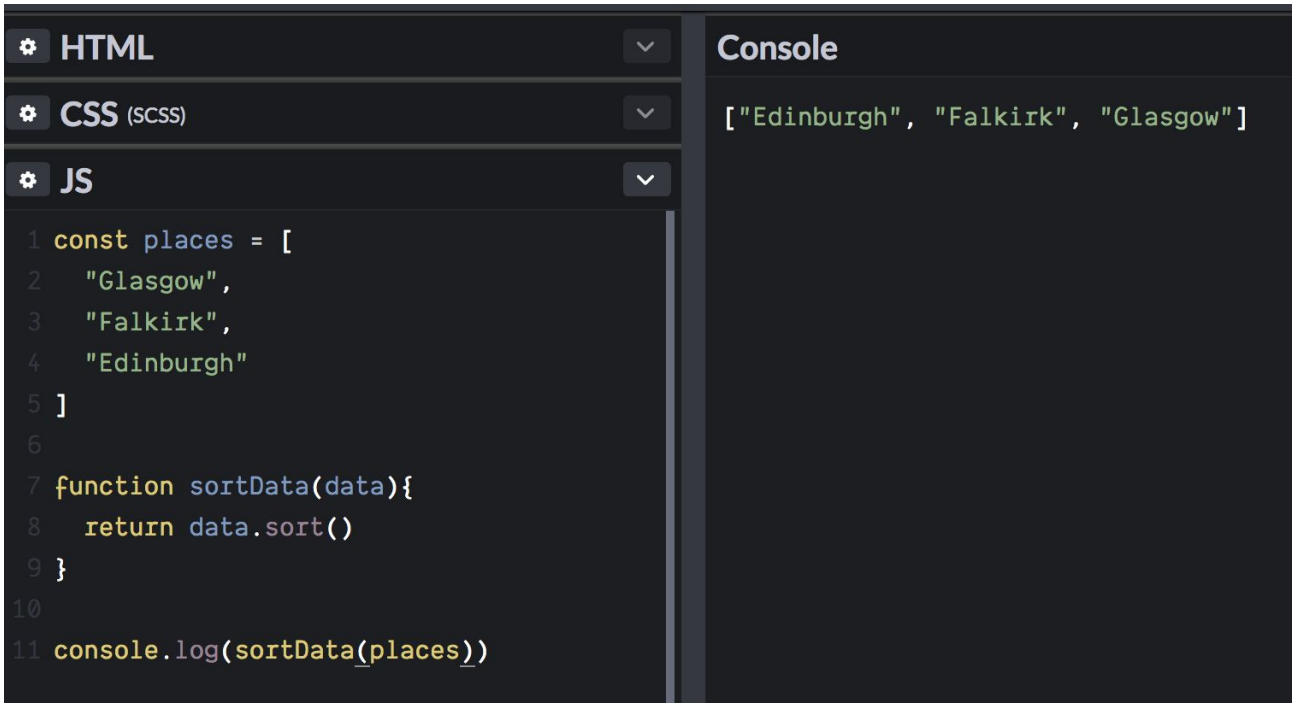
>

Description here

function findPlace receives a name to search for and an array and returns the element if found or false if not found

Unit	Ref	Evidence	
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running	

Paste Screenshot here

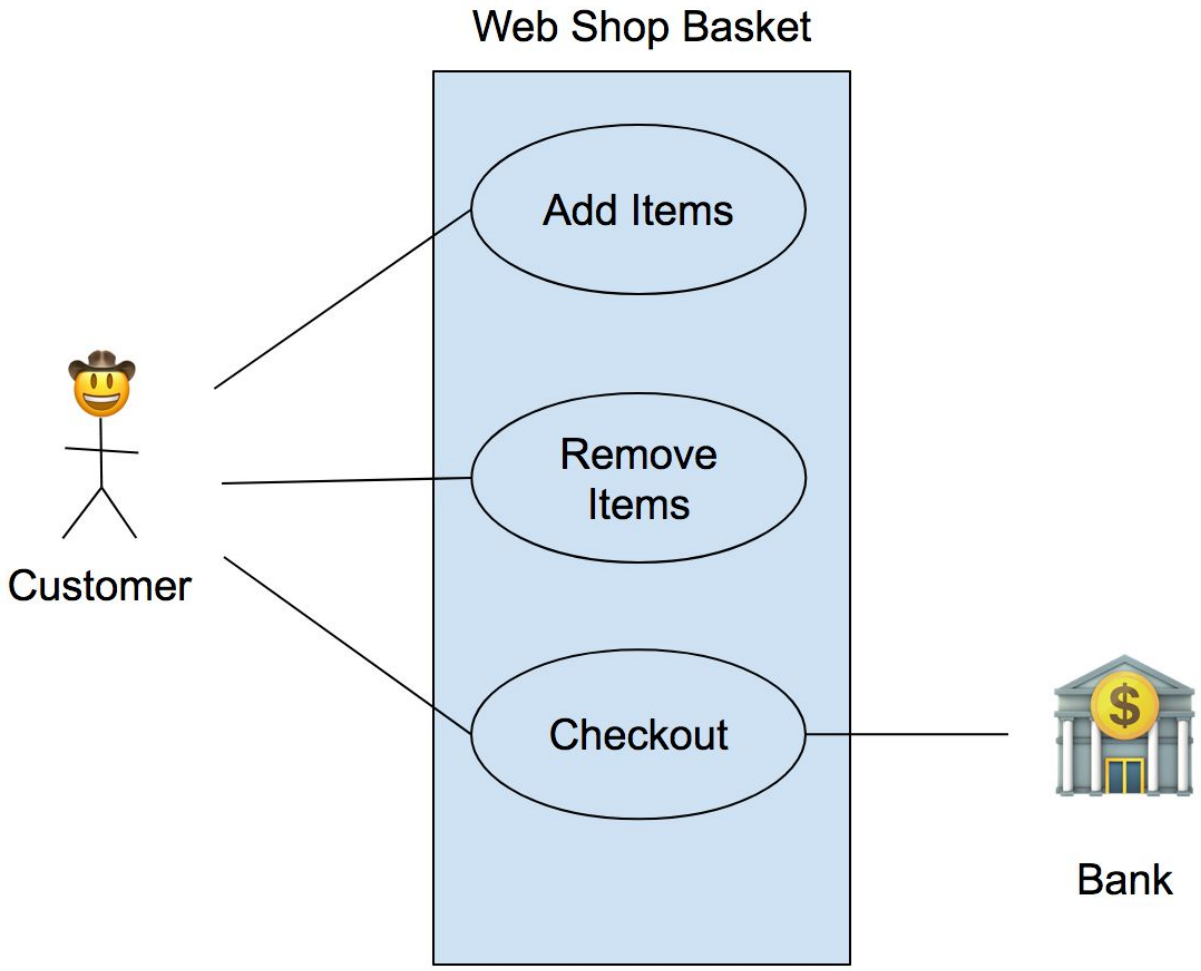


Description here

Week 5

Unit	Ref	Evidence	
A&D	A.D.1	A Use Case Diagram	

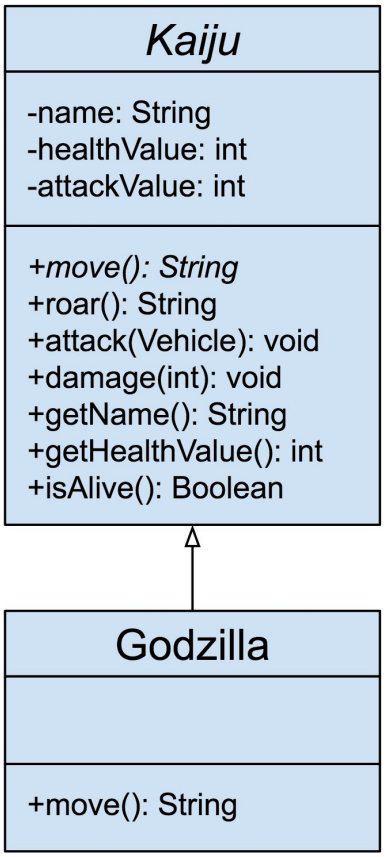
Paste Screenshot here



Description here

Unit	Ref	Evidence	
A&D	A.D.2	A Class Diagram	

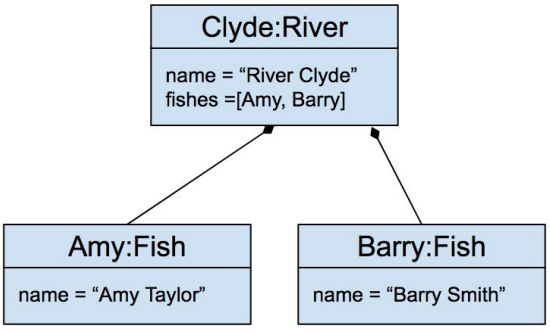
Paste Screenshot here



Description here

Unit	Ref	Evidence	
A&D	A.D.3	An Object Diagram	

Paste Screenshot here

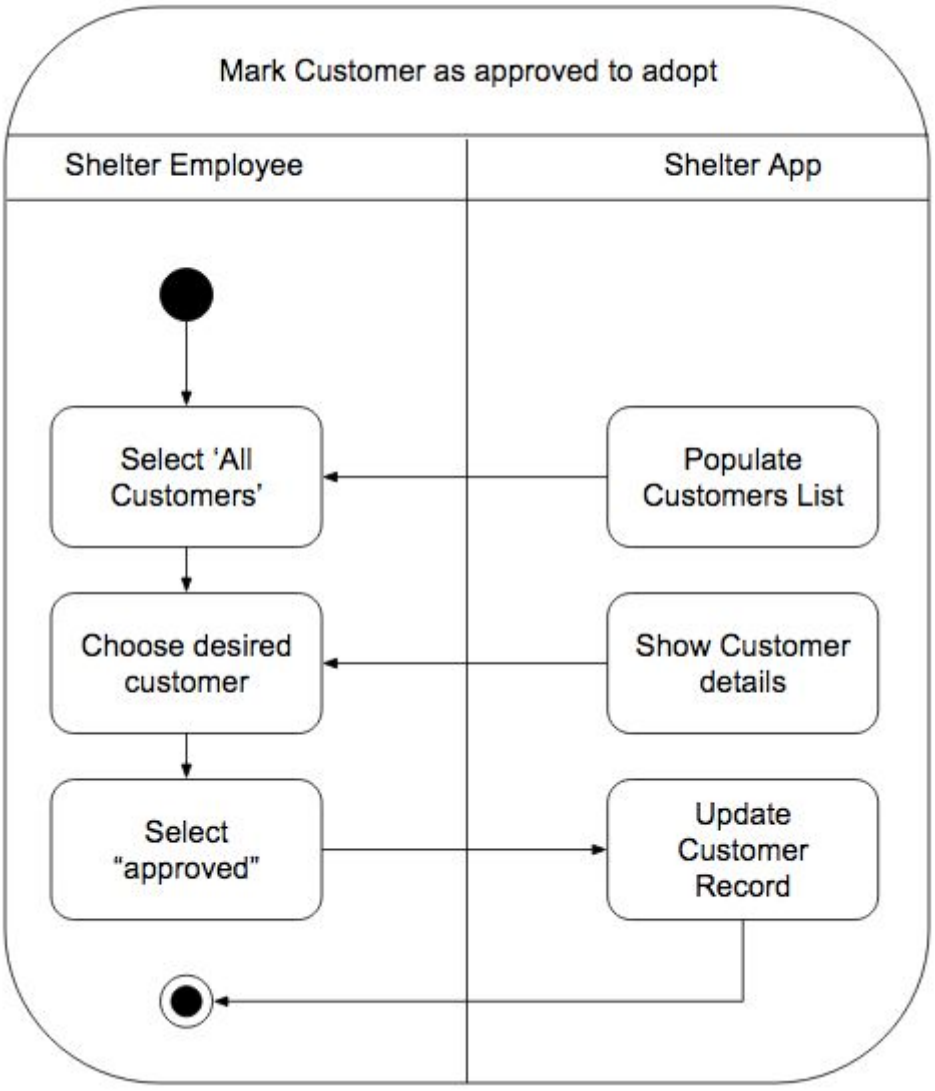


Description here

Unit	Ref	Evidence	

A&D	A.D.4	An Activity Diagram

Paste Screenshot here



Description here

Unit	Ref	Evidence	
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	

Paste Screenshot here

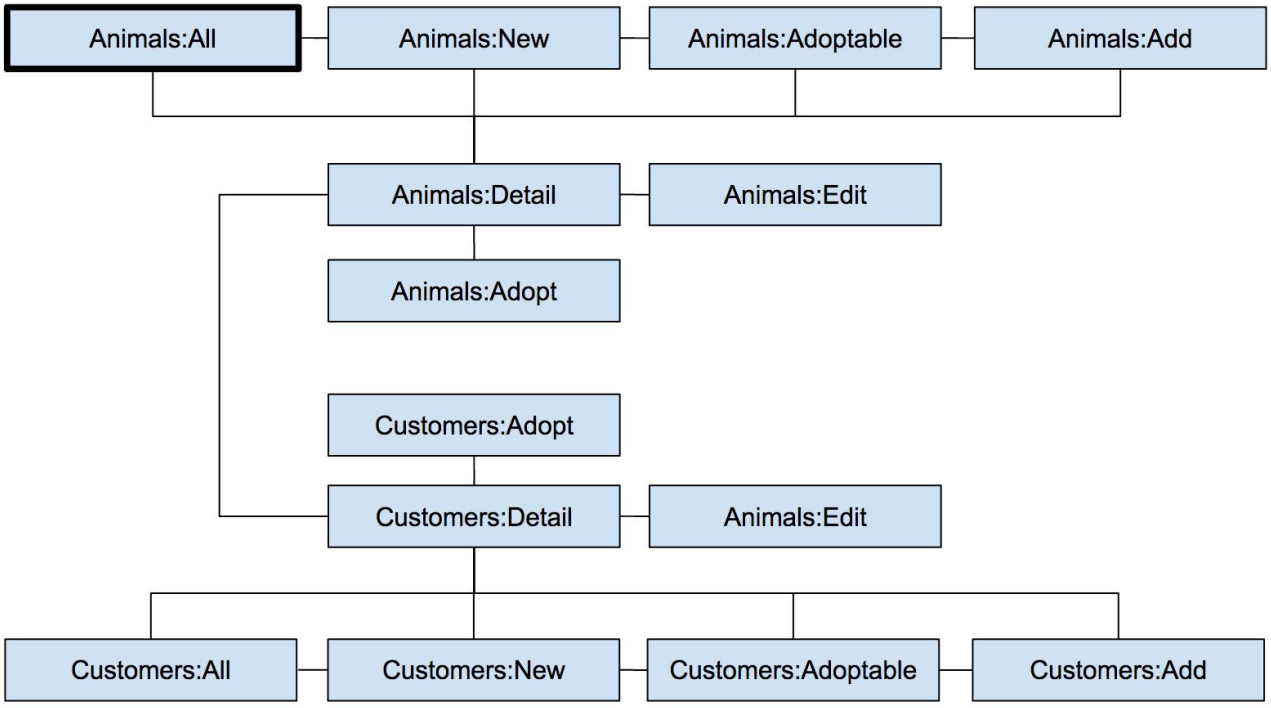
Description here

Constraint	What, How, Why	Solution
Hardware and software platforms	Users have old, small screens. The design of the web page, built on a larger, modern screen may not fit. This will frustrate users and slow down their use of the page.	Use responsive design, allowing wide pages to wrap and fit to any screen width.
Performance requirements	Users have slow internet connection. Pages with lots of data or image will be very slow to load and may cost the user extra to download.	Use image thumbnails on main pages for faster loading and only show full-size where required, or specifically selected by the user.
Persistent storage and transactions	Risk that over time storage would run out. this would prevent the app from accepting new data, rendering it unsuitable for users requirements.	Host on flexible cloud storage such as AWS. This would mean users only pay for what they use and would remove storage restriction.
Usability	Partially sighted users may use a screenreader to navigate web page. Screenreader must be able to find sections and links. Designer may subvert html/css standard to provide more aesthetically pleasing page.	Ensure semantic HTML is used and anything that can be clicked is either a link or button element. Add alt tags to images so users can hear a description if they use a screenreader. Use accessibility validator to test usability against standards.
Budgets	User is a charity and must keep cost to a minimum and be able to justify any costs. This can restrict features available such as persistent storage.	Assess potential costs before starting development to ensure minimum viable product can be achieved within budget requirements. Use free services where possible for hosting and data storage.
Time	Only having one week to deliver project may mean that key features aren't delivered in time and user has to delay deployment of application,	Plan for minimum viable product and ensure this is delivered within time limitation. Further features can be added after delivery,

	adding further costs associated with current manual system.	reducing cost to user.

Unit	Ref	Evidence	
P	P.5	User Site Map	

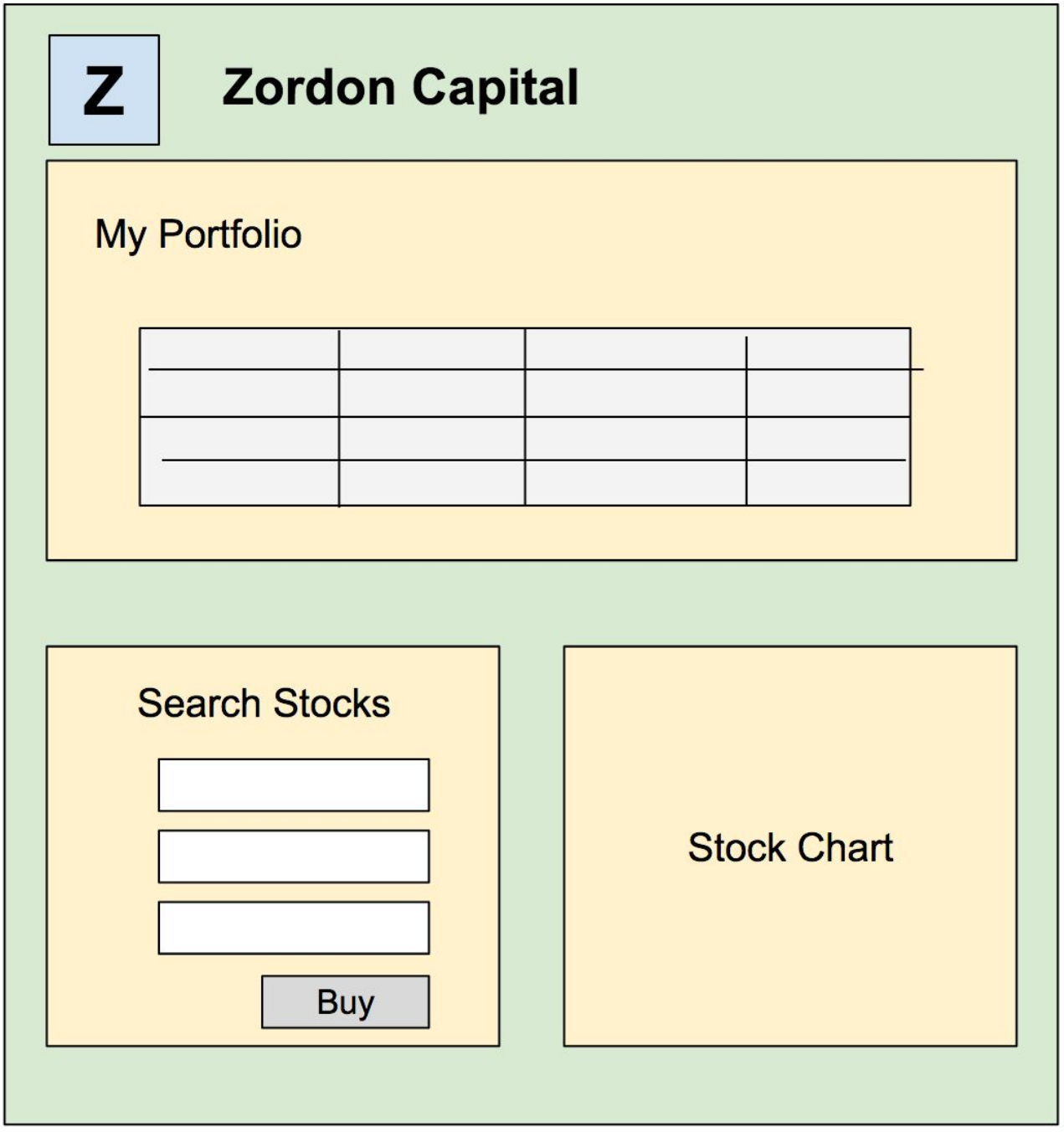
Paste Screenshot here

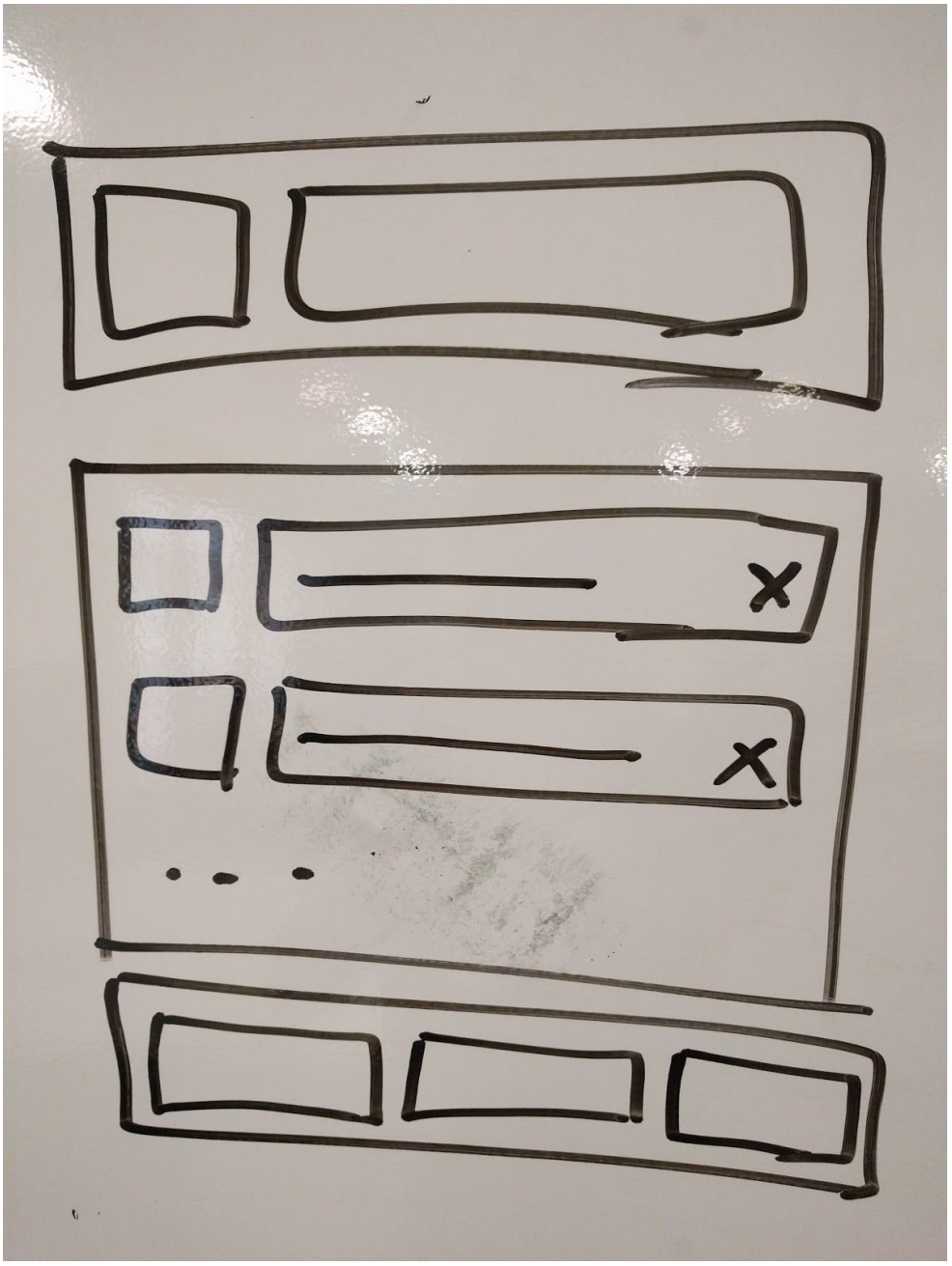


Description here

Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	

Paste Screenshot here


Z
Zordon Capital
My Portfolio
Search Stocks
Buy
Stock Chart



Description here

Outline of full page SPA from share portfolio group project
To-do list ReactJS component breakdown for final solo project

Unit	Ref	Evidence	
P	P.10	Example of Pseudocode used for a method	

Paste Screenshot here

connect to database
retrieve all records from todo table
iterate over each record and create a new todoItem object from record
insert each todo object into array

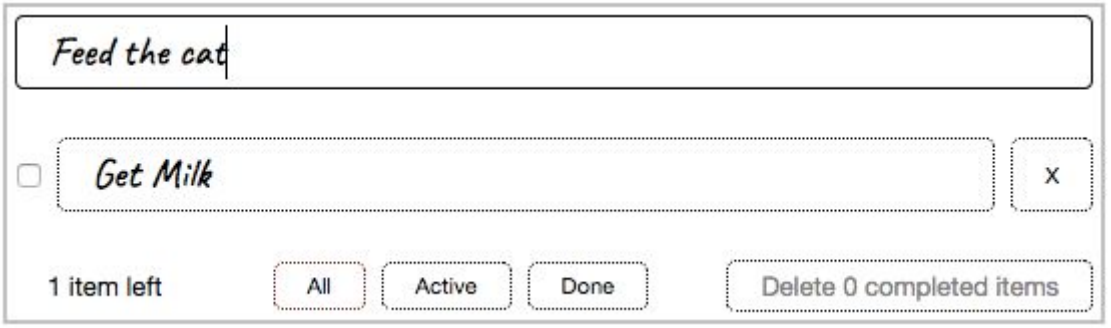
return array

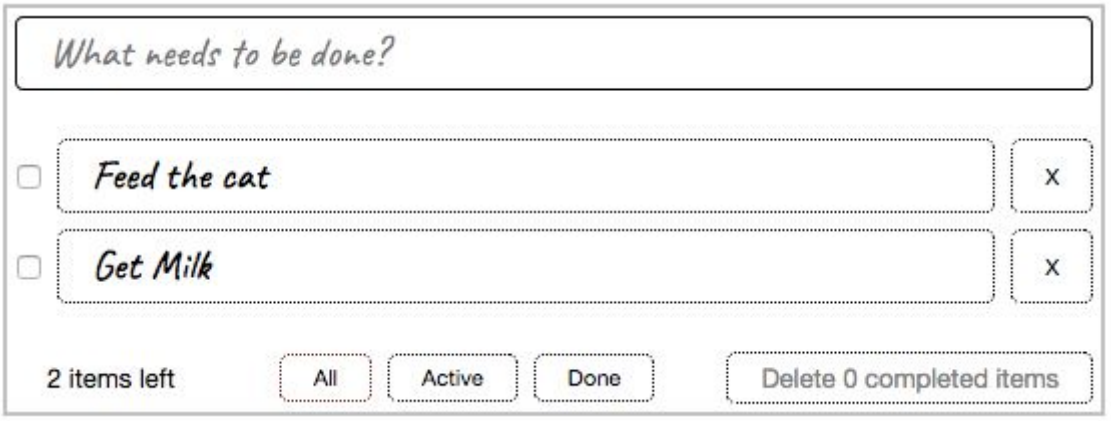
Description here

method returns array of todo items from database

Unit	Ref	Evidence	
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way	

Paste Screenshot here



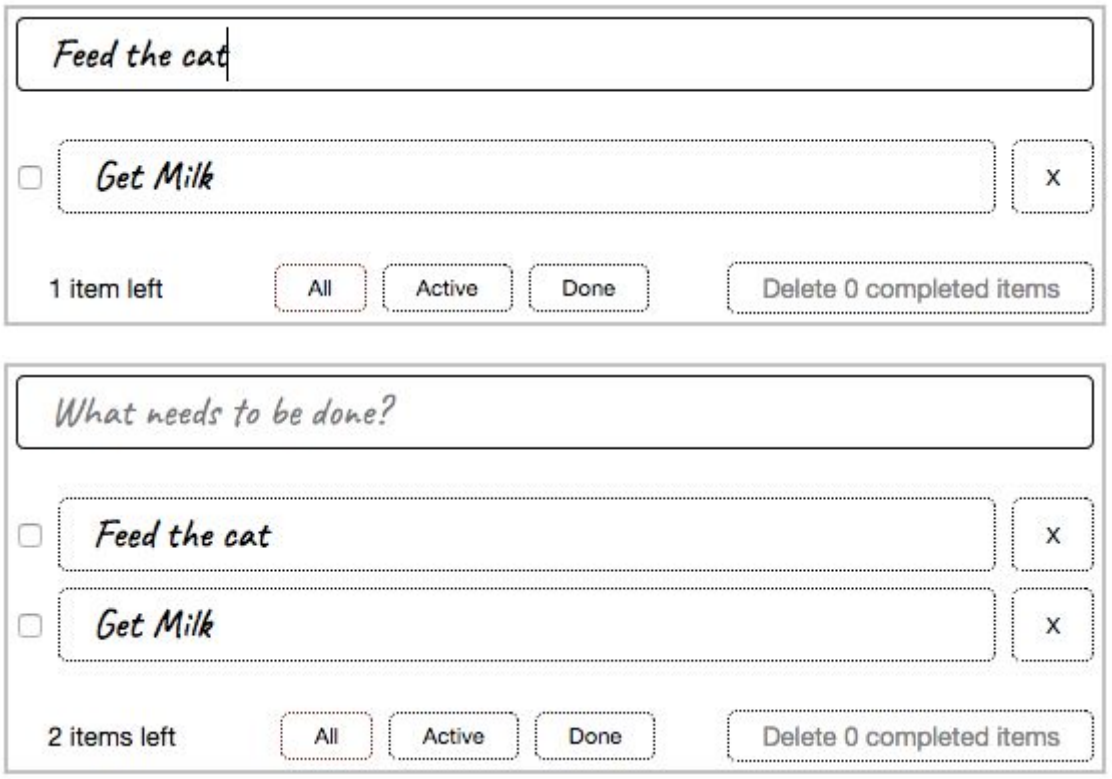


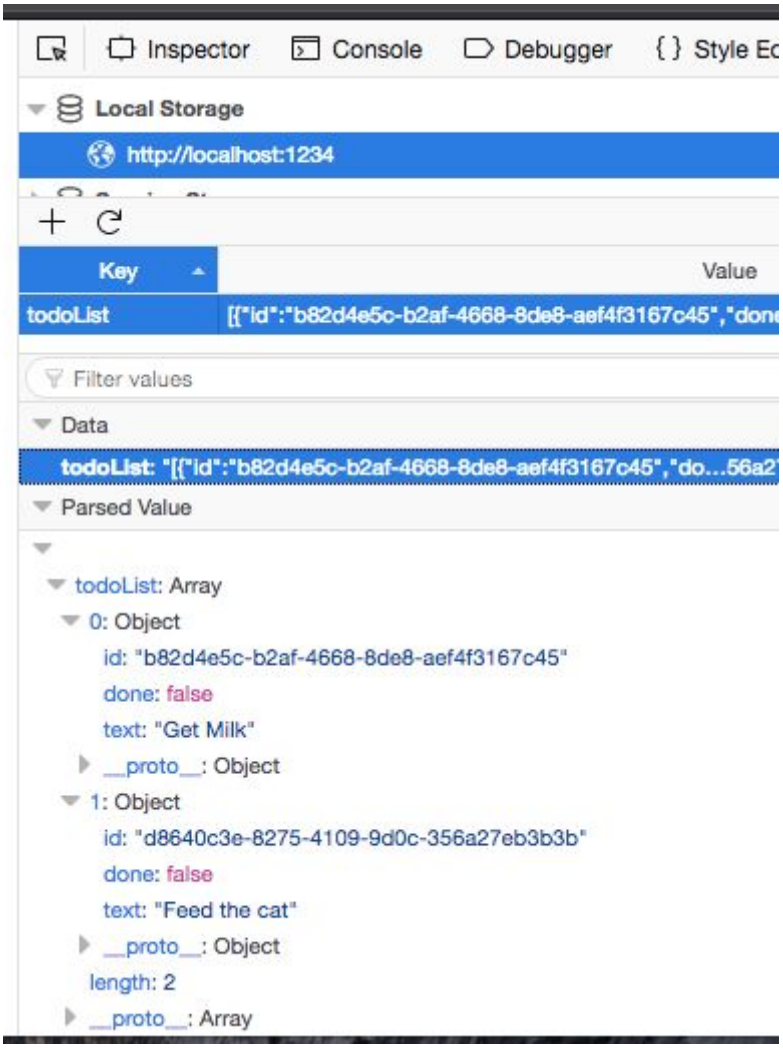
Description here

User can add a new todo item (“Feed the cat”) by typing in input box and pressing enter. New item appears in list shown in second screenshot

Unit	Ref	Evidence	
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved	

Paste Screenshot here



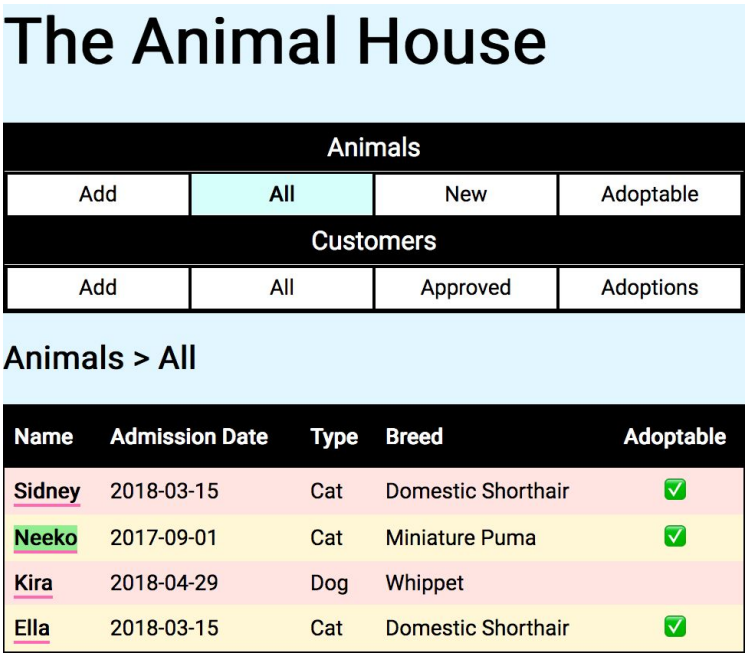


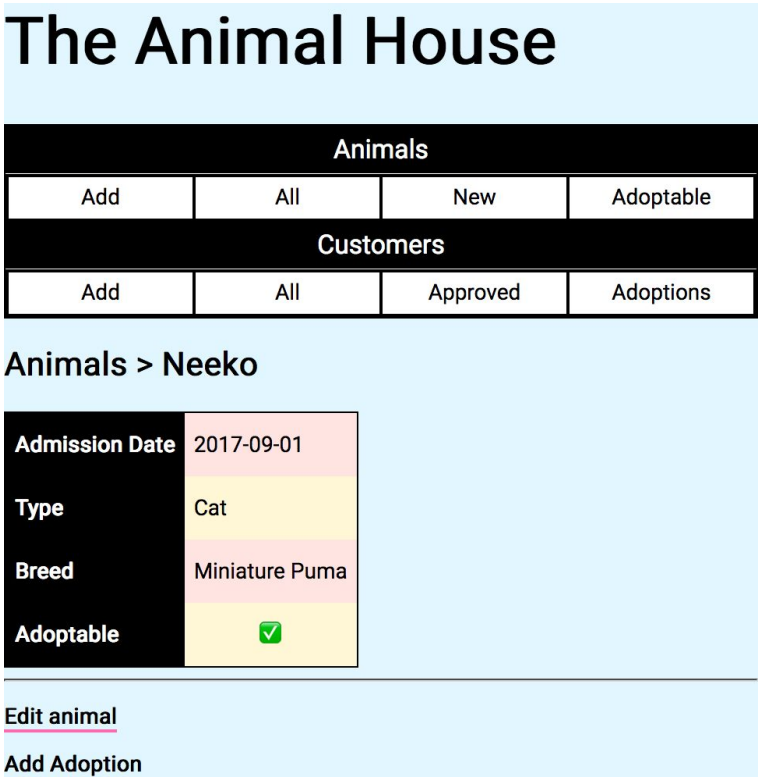
Description here

same user interaction as above item; last screenshot shows the data persisting in browser's local storage

Unit	Ref	Evidence	
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program	

Paste Screenshot here





Description here

User clicks on link “Neeko” (highlighted) and is presented with “Animals > Neeko” detail page

Unit	Ref	Evidence	
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing	

Paste Screenshot here

arrayFunctions.js

```
function addToStartOfArray(arr, newItem) {
  const newArray = arr.slice(); // take copy of array
  newArray.push(newItem);
  return newArray;
}

module.exports = addToStartOfArray;
```

script.spec.js

```
const addToStartOfArray = require("./arrayFunctions");

describe("addToStartOfArray function", () => {
  test("new item is insterted to top of given array", () => {
    const newItem = "cat";
    let animals = ["dog", "hamster", "birb"];

    animals = addToStartOfArray(animals, newItem);

    expect(animals[0]).toBe(newItem);
  });
});
```

```
● addToStartOfArray function › new item is insterted to top of given array

  expect(received).toBe(expected) // Object.is equality

  Expected: "cat"
  Received: "dog"

     8 |     animals = addToStartOfArray(animals, newItem);
     9 |
  > 10 |     expect(animals[0]).toBe(newItem);
       |                        ^
    11 |   });
    12 | });
    13 |

    at Object.toBe (script.spec.js:10:24)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        1.576s, estimated 3s
Ran all test suites.
npm ERR! Test failed.  See above for more details.
```

JS arrayFunctions.js ×

```
1  function addToStartOfArray(arr, newItem) {
2    const newArray = arr.slice(); // take copy of array
3    newArray.unshift(newItem);
4    return newArray;
5  }
6
7  module.exports = addToStartOfArray;
8
```

```
PASS ./script.spec.js
  addToStartOfArray function
    √ new item is insterted to top of given array (14ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.514s
Ran all test suites.
```

Description here

Week 7

Unit	Ref	Evidence	
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.	

Paste Screenshot here

```java
package player;

import player.Interfaces.IAttack;
import player.Interfaces.IDefend;

public class Magician extends Player implements IAttack, IDefend {
    private Ability weapon;
    private Ability defense;
    private int stamina;

    Magician(String name, Ability weapon, Ability defense) {
        super(name);
        this.weapon = weapon;
        this.defense = defense;
        this.stamina = 100;
    }

    @Override
    public void damage(int amount) {
        this.health -= (amount - this.defense.getStrength());
    }

    @Override
    public void attack(Player player) {
        int attackValue = Math.floorDiv(this.weapon.getStrength() * this.stamina, 100);
        player.damage(attackValue);
        this.stamina -= 10;
    }

    @Override
    public Ability getWeapon() {
        return this.weapon;
    }

    @Override
    public Ability getDefense() {
        return defense;
    }

    @Override
    public void switchWeapon(Ability weapon) {
        this.weapon = weapon;
    }

    public void switchDefense(Ability defense) {
        this.defense = defense;
    }

    public int getStamina() {
        return this.stamina;
    }
}
```

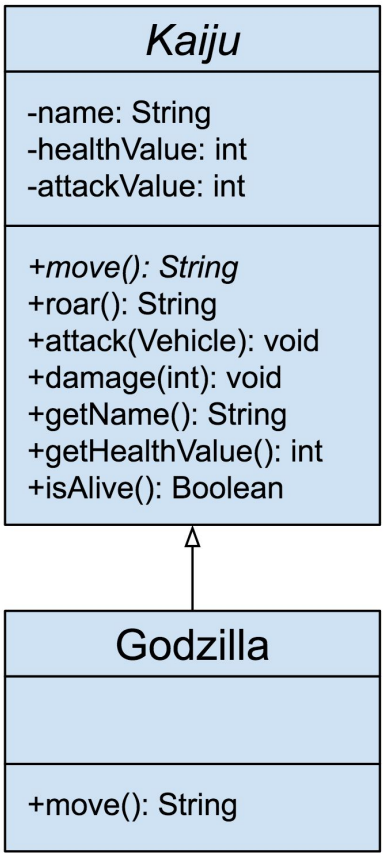
```java
package player.Interfaces;

import player.Ability;
import player.Player;

public interface IAttack {
    void attack(Player player);
    void switchWeapon(Ability weapon);
    Ability getWeapon();
}
```

Description here

Class Magician implements interface IAttack by overriding methods attack, switchWeapon and getWeapon

Unit	Ref	Evidence	
A&D	A.D.5	An Inheritance Diagram	

Paste Screenshot here



Description here

Unit	Ref	Evidence	
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.	

Paste Screenshot here

```java
package player;

public abstract class Player {
    private String name;
    int health;
    int treasure;

    Player(String name) {
        this.name = name;
        this.health = 50;
        this.treasure = 0;
    }

    public String getName() {
        return this.name;
    }

    public int getHealth() {
        return this.health;
    }

    public int getTreasure() {
        return this.treasure;
    }

    public abstract void damage(int amount);

    public void heal(int amount) {
        this.health += amount;

        if (this.health > 50) {
            this.health = 50;
        }
    }

    public void collectTreasure(int amount) {
        this.treasure += amount;
    }
}
```

Description here

private String name is encapsulated and can only be set by the constructor

Unit	Ref	Evidence	

I&T	I.T.2	Take a screenshot of the use of Inheritance in a program. Take screenshots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.

Paste Screenshot here

```
1   public abstract class Kaiju {
2       private String name;
3       private int healthValue;
4       private int attackValue;
5
6       Kaiju(String name, int healthValue, int attackValue) {
7           this.name = name;
8           this.healthValue = healthValue;
9           this.attackValue = attackValue;
10      }
11
12      public abstract String move();
13
14      public String roar() {
15          return "*clears throat* ...RoaR";
16      }
17
18      public void attack(Vehicle vehicle) {
19          vehicle.damage(this.attackValue);
20      }
21
22      public void damage(int attackValue) {
23          this.healthValue -= attackValue;
24      }
25
26      public String getName() {
27          return this.name;
28      }
29
30      public boolean isAlive() {
31          return this.healthValue > 0;
32      }
33
34      public int getHealthValue() {
35          return this.healthValue;
36      }
37  }
```

```java
public class Godzilla extends Kaiju {
    Godzilla() {
        super("ゴジラ", 200, 50);
    }

    @Override
    public String move() {
        return "*Stomp stomp*";
    }
}
```

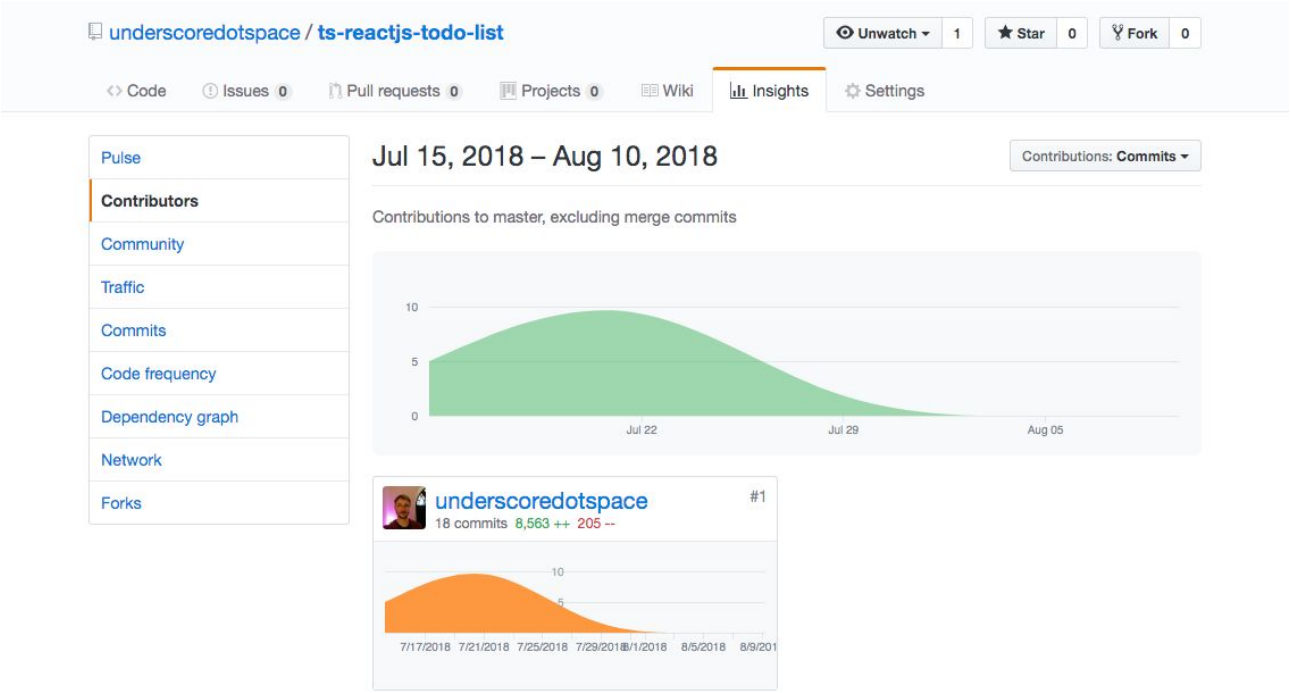
```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class GodzillaTest {
    Godzilla gojira;

    @Before
    public void before() {
        this.gojira = new Godzilla();
    }

    @Test
    public void hasName() {
        assertEquals("ゴジラ", gojira.getName());
    }

    @Test
    public void isAlive() {
        assertEquals(true, gojira.isAlive());
    }

    @Test
    public void isNotAlive() {
        gojira.damage(200);
        assertEquals(false, gojira.isAlive());
    }

    @Test
    public void canRoar() {
        assertEquals("*clears throat* ...RoaR", gojira.roar());
    }

    @Test
    public void move() {
        assertEquals("*Stomp stomp*", gojira.move());
    }

    @Test
    public void canAttack() {
        Tank tank = new Tank();
        gojira.attack(tank);
        assertEquals(200, tank.getHealthValue());
    }
}
```

Description here

Week 10

Unit	Ref	Evidence	
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	

Paste Screenshot here

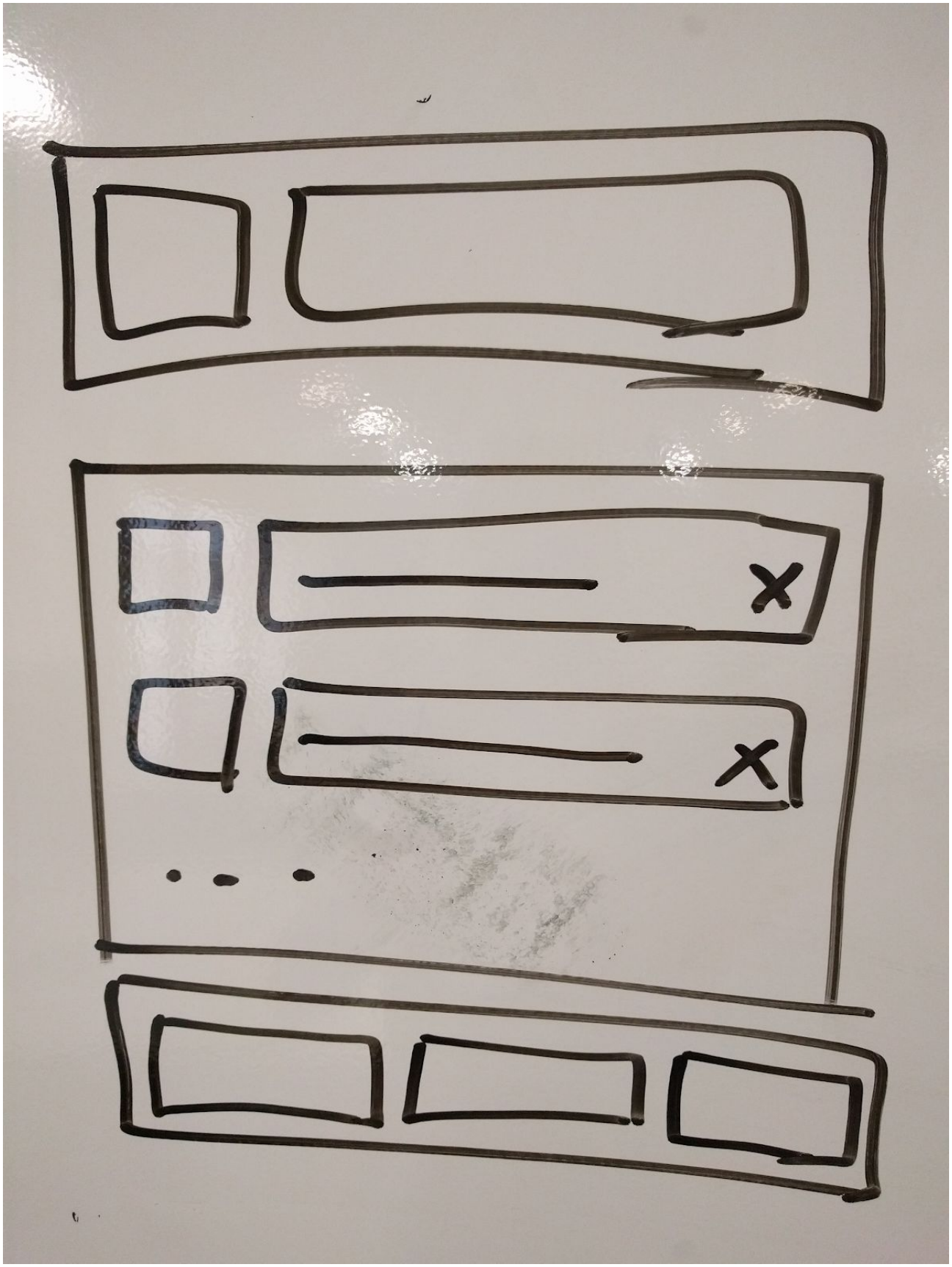


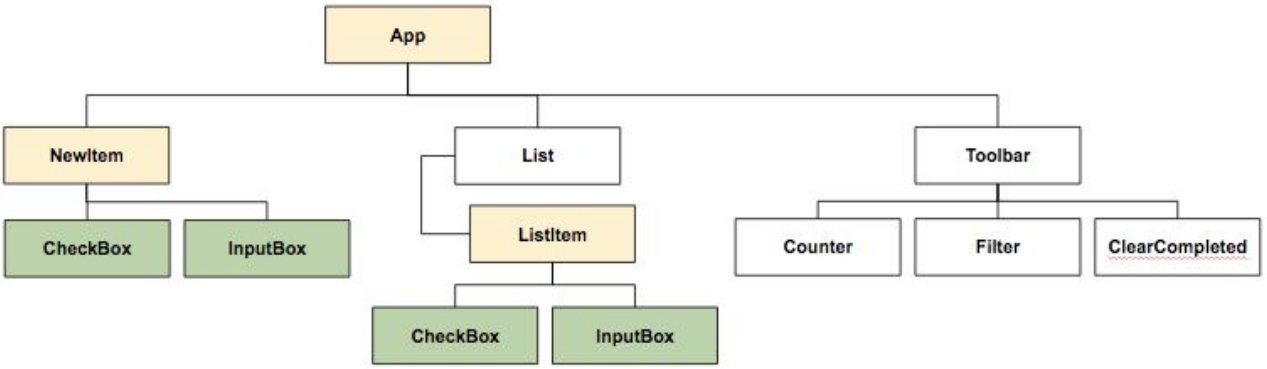
Description here

https://github.com/underscoredotspace/ts-reactjs-todo-list

Unit	Ref	Evidence	
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.	

Paste Screenshot here





Description here

Initial component planning, showing high level layout with no naming
Component diagram, showing reusable components (green) and those that have state (yellow)

Unit	Ref	Evidence	
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.	

Paste Screenshot here

```
handleTextChange = (text: string, id: string) => {
  const { todoList } = this.state
  const listItem = todoList.find(item => item.id === id)
  if (!listItem) return
  listItem.text = text

  this.setState({ todoList })
}
```

```ruby
class Game

  def initialize(player1, player2)
    @player1 = player1.downcase
    @player2 = player2.downcase
  end

  def isValid?(move)
    case move
    when 'rock', 'paper', 'scissors'
      return true
    end

    return false
  end

  def play()
    return "Player 1's move is invalid" if !self.isValid?(@player1)
    return "Player 2's move is invalid" if !self.isValid?(@player2)
    return "It's a draw. You both lose. " if @player1 == @player2

    win_test = {
      "rock" => "scissors",
      "paper" =>  "rock",
      "scissors" => "paper"
    }

    if win_test[@player1][@player2]
      return "Player 1 wins"
    end

    return "Player 2 wins"
  end
end
```

Description here

First example uses higher order function ‘find’ to locate single element of todoList array, and updates the text property of that element. I chose it because it takes full advantage of the optimisations in React (with setState) and the JavaScript language engine, making it as efficient as some of the world’s best engineers can make it.

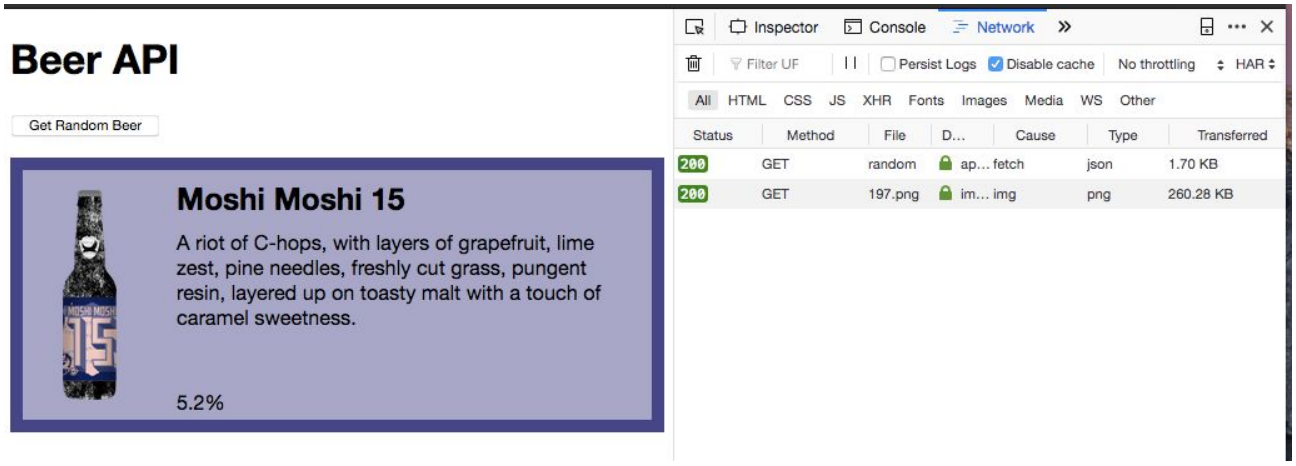
The second example shows the game of Rock, Paper, Scissors. I chose this because it demonstrates an algorithm that is broken down in a way that easy to understand. It is also efficient as it aims to do the least amount of work possible in each possible case, i.e., not even creating the win_test hash if it knows there was a draw, or invalid input.

Week 12

Unit	Ref	Evidence	
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	

Paste Screenshot here

```
const pubSub = require('../helpers/pubSub')
const request = require('../helpers/request')

class BeerAPI {
  constructor() {
    const baseURL = 'https://api.punkapi.com/v2/beers'

    this.url = {
      random: `${baseURL}/random`
    }
  }

  bindEvents() {
    pubSub.subscribe('RandomBeer:get-random', () => {
      this.getRandom()
    })
  }

  get(url, callback) {
    request(url, (error, response) => {
      if (error) {
        alert('Error!')
        console.error(error)
        return
      }

      callback(response)
    })
  }

  getRandom() {
    this.get(this.url.random, response => {
      const randomBeer = response[0]
      pubSub.publish('BeerAPI:random-beer', randomBeer)
    })
  }

}

module.exports = BeerAPI
```

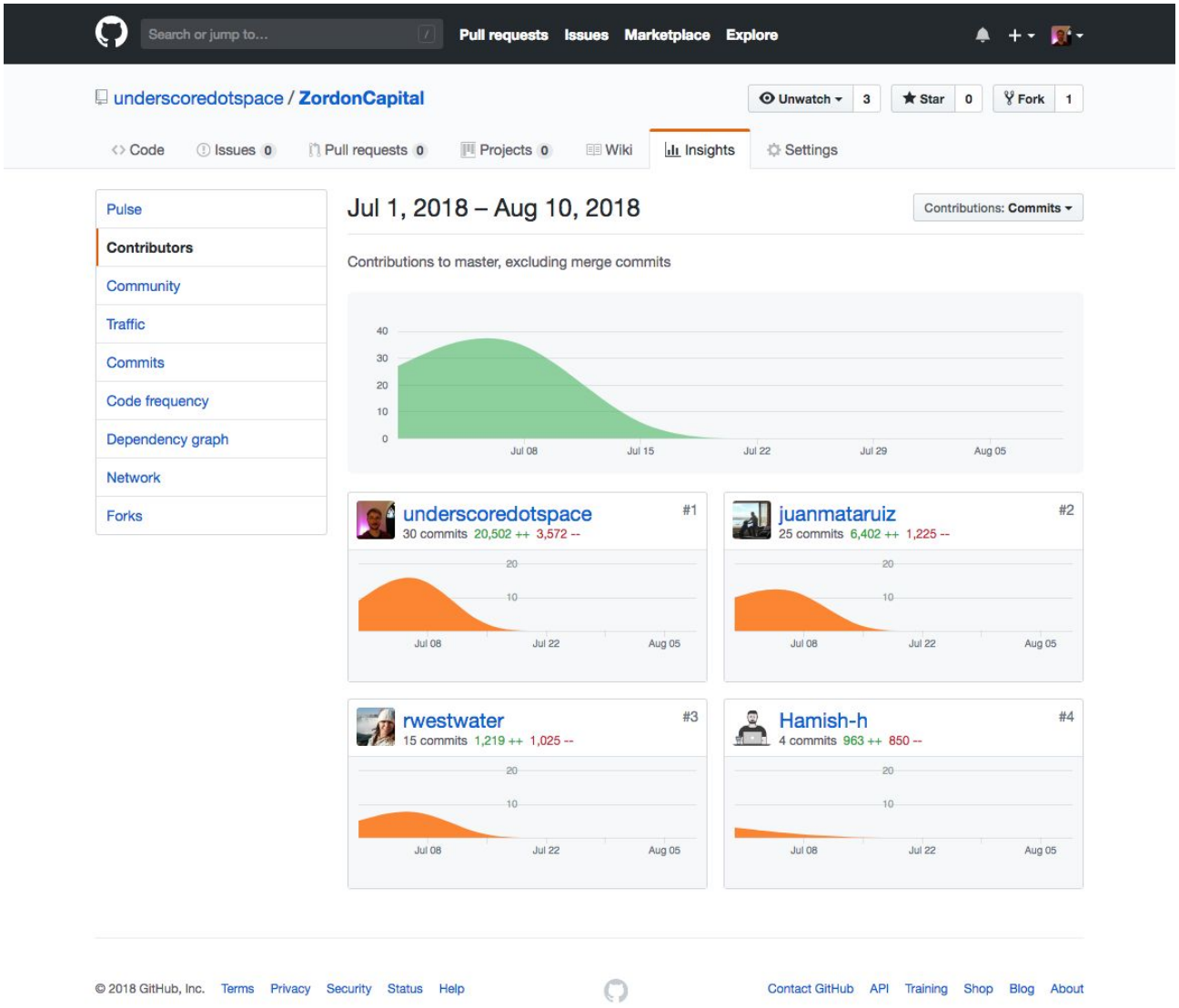


Description here

Week 15

Unit	Ref	Evidence	
P	P.1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.	

Paste Screenshot here



Description here

Unit	Ref	Evidence	
P	P.2	Take a screenshot of the project brief from your group project.	

Paste Screenshot here

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

Examples of Further Features

- Speculation based on trends and further financial modelling using projections.

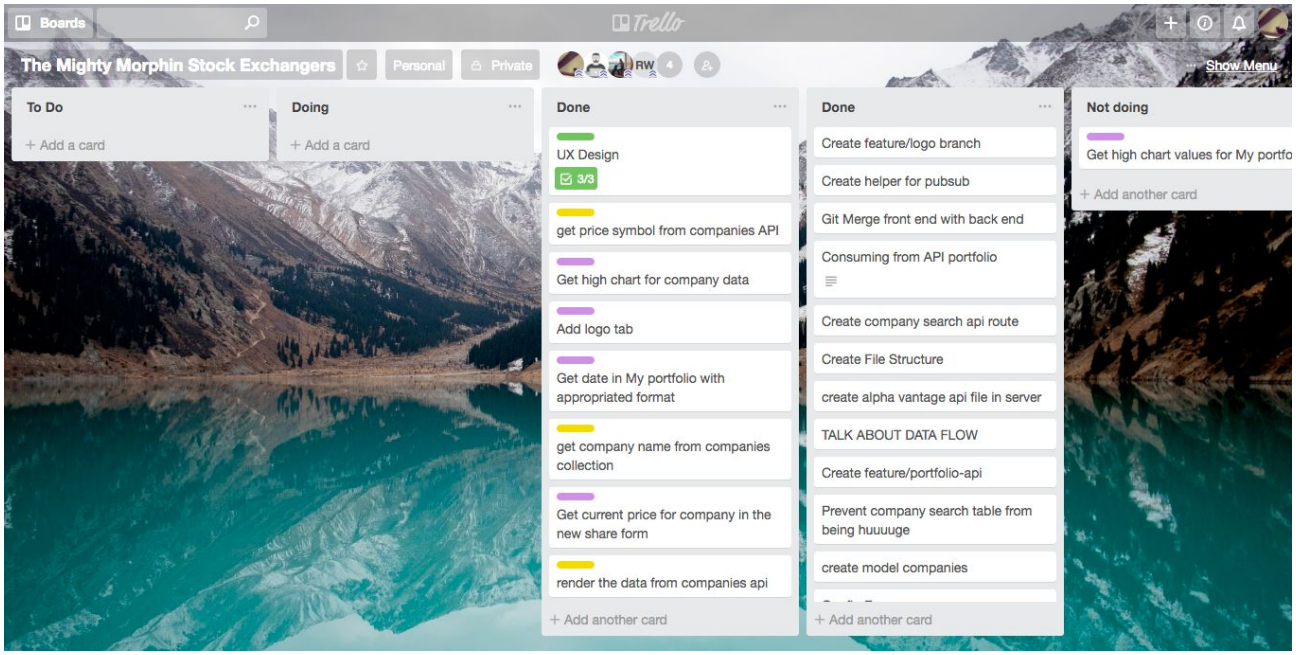
API, Libraries, Resources

- https://www.alphavantage.co/ (Requires sign up)
- https://www.highcharts.com/ HighCharts is an open-source library for rendering responsive charts.

Description here

Unit	Ref	Evidence	
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	

Paste Screenshot here



Description here

Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	

Paste Screenshot here

Acceptance Criteria	Expected Result/Output	Pass/Fail
User should be able to add a new item to their todo list	Item appears in list and is saved in local storage	pass
User should able to filter for All/Active/Done todo items	List displays only items that match filter	pass

Description here

Unit	Ref	Evidence	
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.8	Produce two object diagrams.	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.17	Produce a bug tracking report	

Paste Screenshot here

Description here