

David Pears

**Evidence Gathering Document
SQA Level 8 Professional Developer Award.**

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

```

array.rb
1 def users
2   users = {
3     "jonathan" => {
4       :twitter => "jonnyt",
5       :lottery_numbers => [6, 12, 49, 33, 45, 20],
6       :home_town => "Stirling",
7       :pets => [
8         {
9           :name => "fluffy",
10          :species => "cat"
11        },
12      ]
13    },
14
15   "Avril" => {
16     :twitter => "bridpally",
17     :lottery_numbers => [12, 13, 33, 38, 9, 25],
18     :home_town => "Dunbar",
19     :pets => [
20       {
21         :name => "monty",
22         :species => "snake"
23       }
24     ]
25   }
26 }
27 end
28
29 p users["Avril"][:lottery_numbers]
30

```

Simple program that shows two users in an array, each user has various attributes (their twitter, lottery numbers, home town and pets)

The function returns Avril's lottery numbers.

```

pda_examples — davidpears@Davids-MacBook-Pro — ..../pda_examples — -zs...
[→ pda_examples ruby array.rb
[12, 13, 33, 38, 9, 25]
→ pda_examples

```

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

◆ PDAExample2.rb

```

1 person1 = {
2   age: 40,
3   name: "David",
4   quote: "I am studying at CodeClan",
5   can_talk: true}
6
7 def person_speaks person
8   if person[:can_talk] == true
9     return person[:quote]
10    else
11      return "I have nothing to say."
12    end
13  end
14
15 puts person_speaks(person1)
16
17 <

```

```
[→ Wednesday git:(master) ✘ ruby PDAExample2.rb
I am studying at CodeClan
→ Wednesday git:(master) ✘ ]
```

An example of a hash in a program that returns a short sentence if a person has an ability to 'talk'.

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

👉 PDAExample3.rb

```

1 ✓ holidays = [
2   {destination: "Cuba", year: 2013, airline: "Air France"}, 
3   {destination: "Sri Lanka", year: 2014, airline: "Qatar"}, 
4   {destination: "Myanmar", year: 2015, airline: "Qatar"}, 
5   {destination: "Madagascar", year: 2017, airline: "Turkish 
6   Airlines"}]
7 
8 ✓ def find_holiday_by_year(holidays, year)
9   return holidays.find{ |holidays|
10    holidays[:year] == year
11  }
12 
13 puts find_holiday_by_year(holidays, 2015)
14

```

```
[→ Wednesday git:(master) ✘ ruby PDAExample3.rb
{:destination=>"Myanmar", :year=>2015, :airline=>"Qatar"}
→ Wednesday git:(master) ✘ █
```

Search function that finds all details of a holiday using the year of the trip.

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

```

1  class_mates = ["David", "Shaun", "Jesus", "Can", "Ben", "James
• A", "James H", "Gary", "Lee", "Iona", "Helen", "Gregor",
• "Stuart", "Ed", "Chris", "Daniel", "Magda", "Stephen", "Wil"]
2
3  def sort_class(array)
4      array.sort|
5  end
6
7  puts sort_class(class_mates)
8

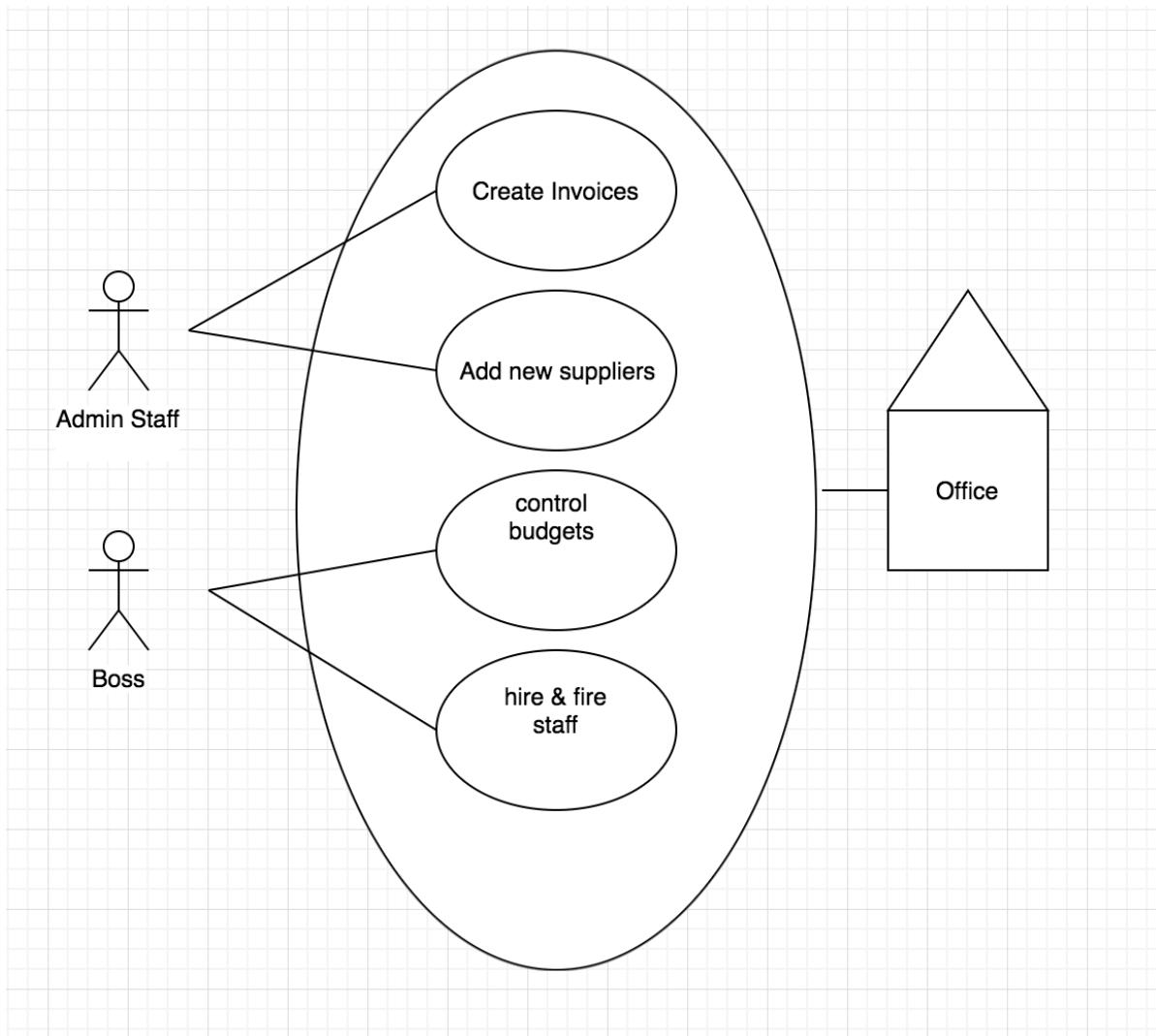
```

```
[→ Wednesday git:(master) ✘ ruby PDAExample4.rb
Ben
Can
Chris
Daniel
David
Ed
Gary
Gregor
Helen
Iona
James A
James H
Jesus
Lee
Magda
Shaun
Stephen
Stuart
Wil
→ Wednesday git:(master) ✘ ]
```

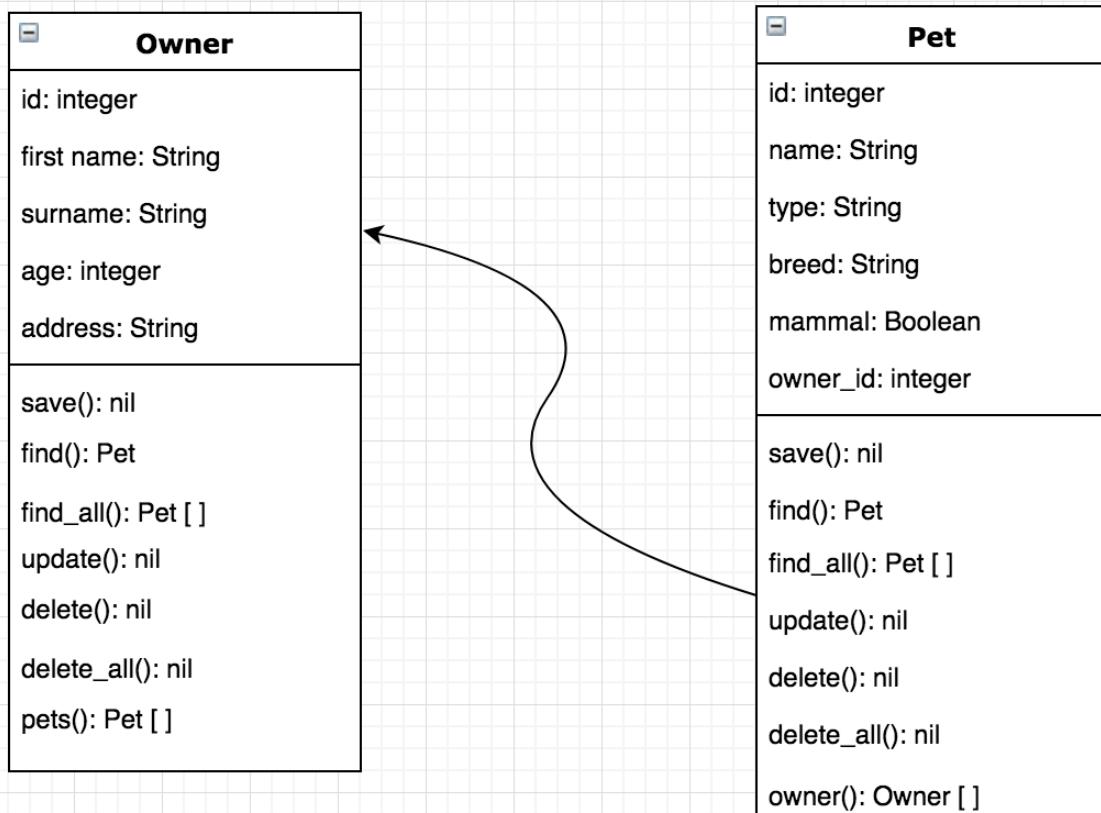
The above function sorts the people from the class in alphabetical order.

Week 5

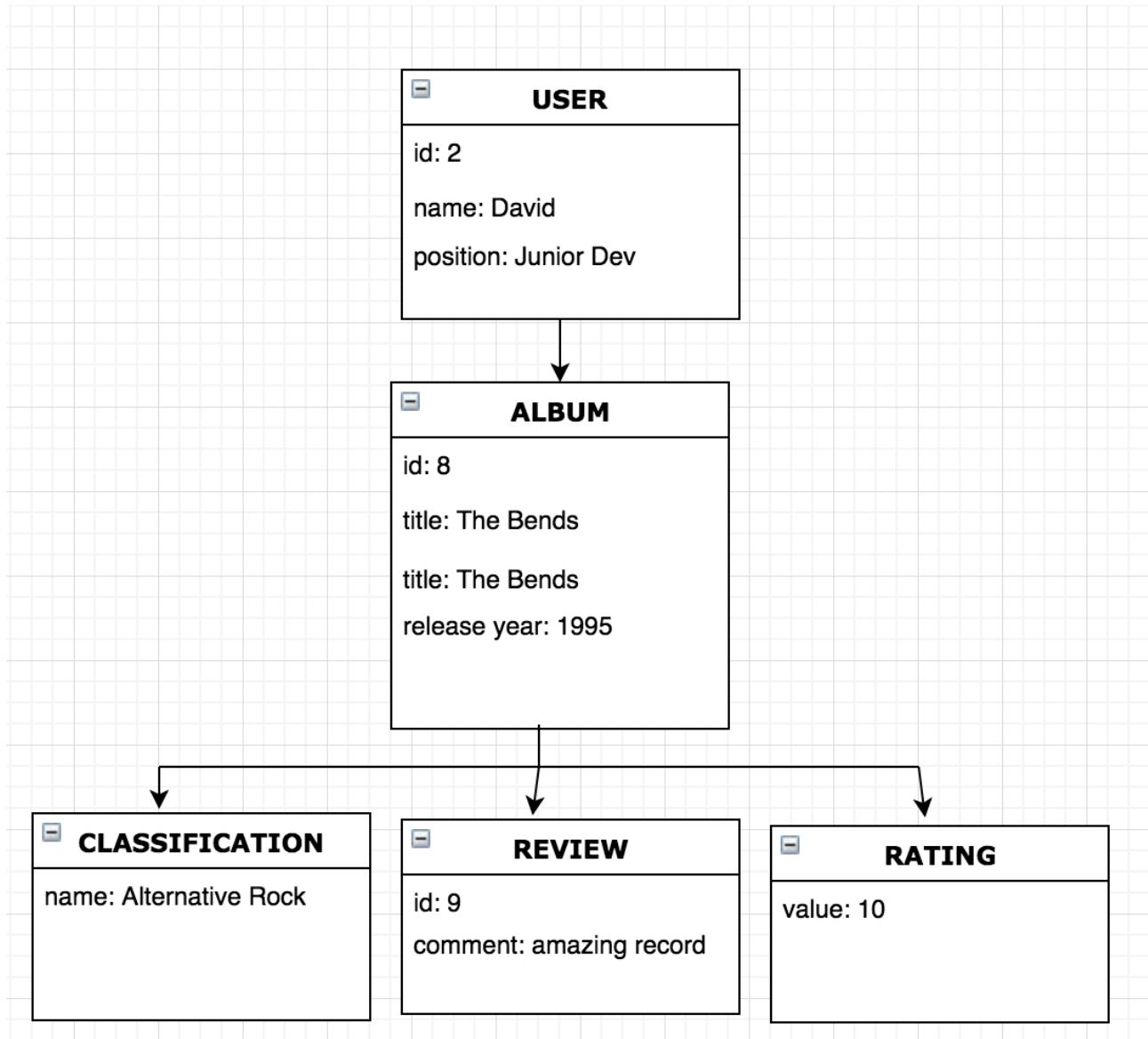
Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram



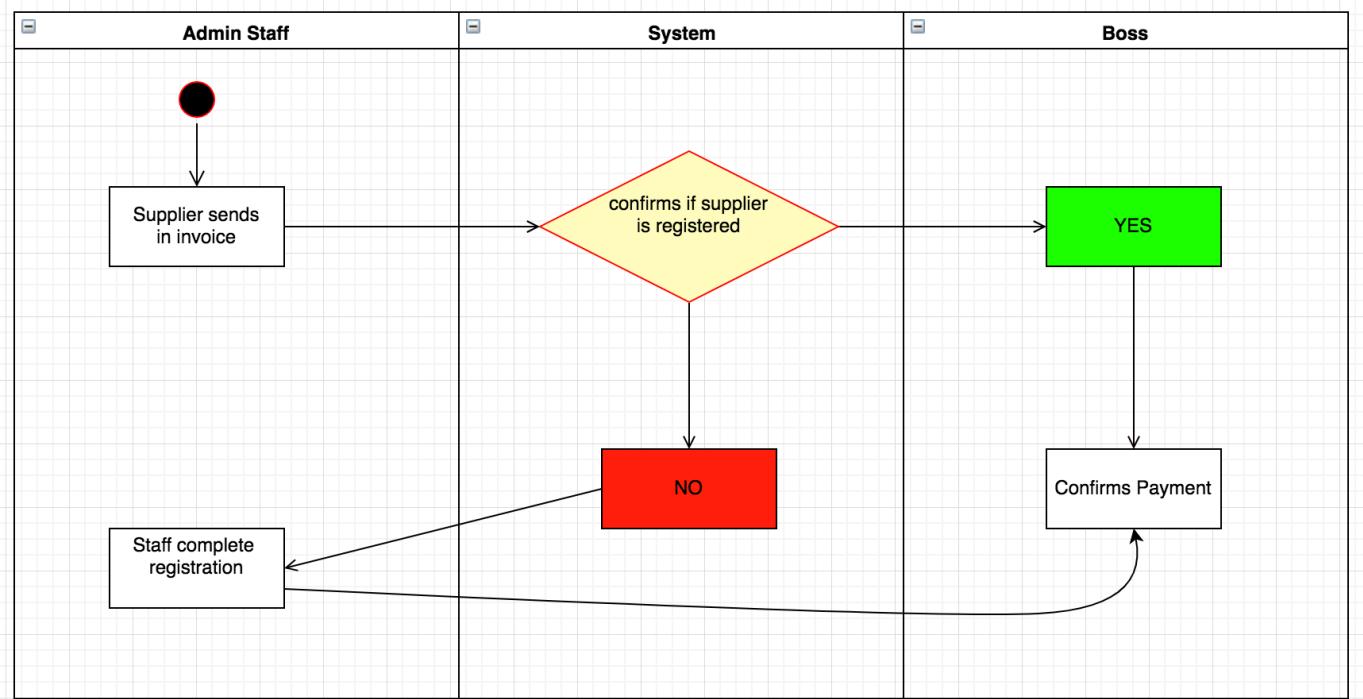
Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram



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



Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram

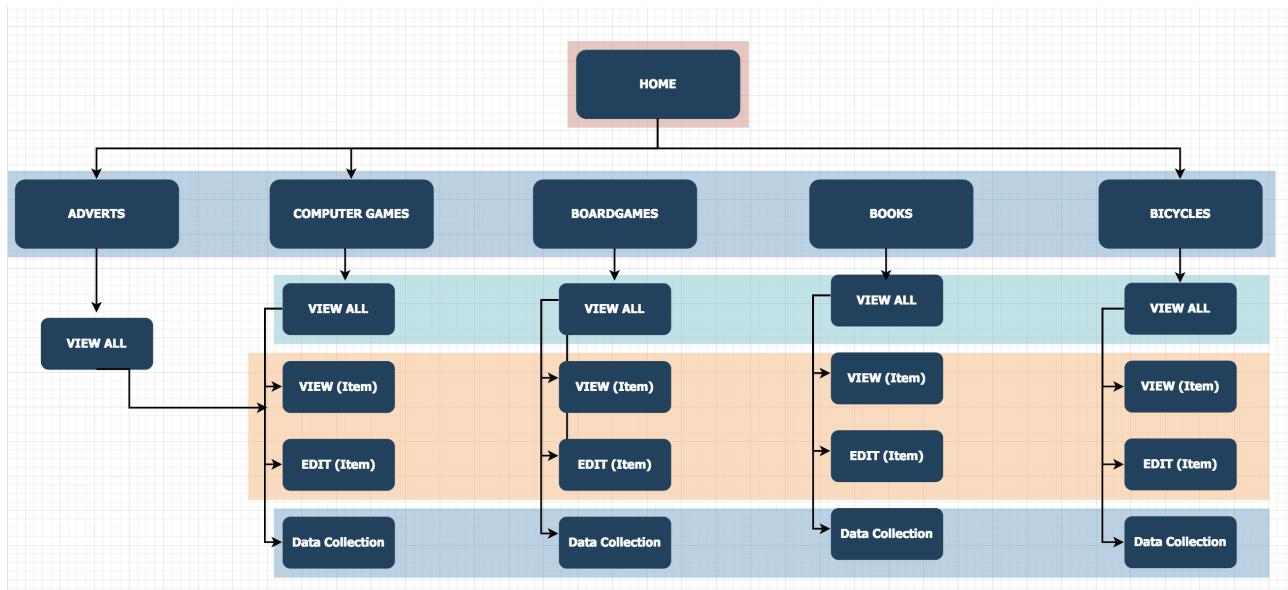


Unit	Ref	Evidence
A&D	A.D.6	<p>Produce an Implementations Constraints plan detailing the following factors:</p> <ul style="list-style-type: none"> *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time

	Constraints & possible effect	Solutiion
Hardware & software platforms	App/Project not responsive to mobile users and only tested in firefox. Other browsers may display differently	User a responsive design (flexbox or grid) to ensure mobile users can rcv the correct experience
Performance requirements	User device(s) may not be ideal. However optimisation for anything other than a desktop web-app not specified in the brief.	User-test to reveal any issues, then reformat for to work with a larger selection of devices
persistent storage and transactions	images added as url strings (i.e web-hosted images), which relies on those urls remaing active	Save file images to the database so they can be accessed permanently
Budgets	Limited budget designed to reach MVP	Once MVP reached, use proof of concept to aquire further budget to increase functionality and extend project
Time	timeframe (1 week) resulted in some extenstions/plans not being completed	Further extensions/functionality to be added at a later date

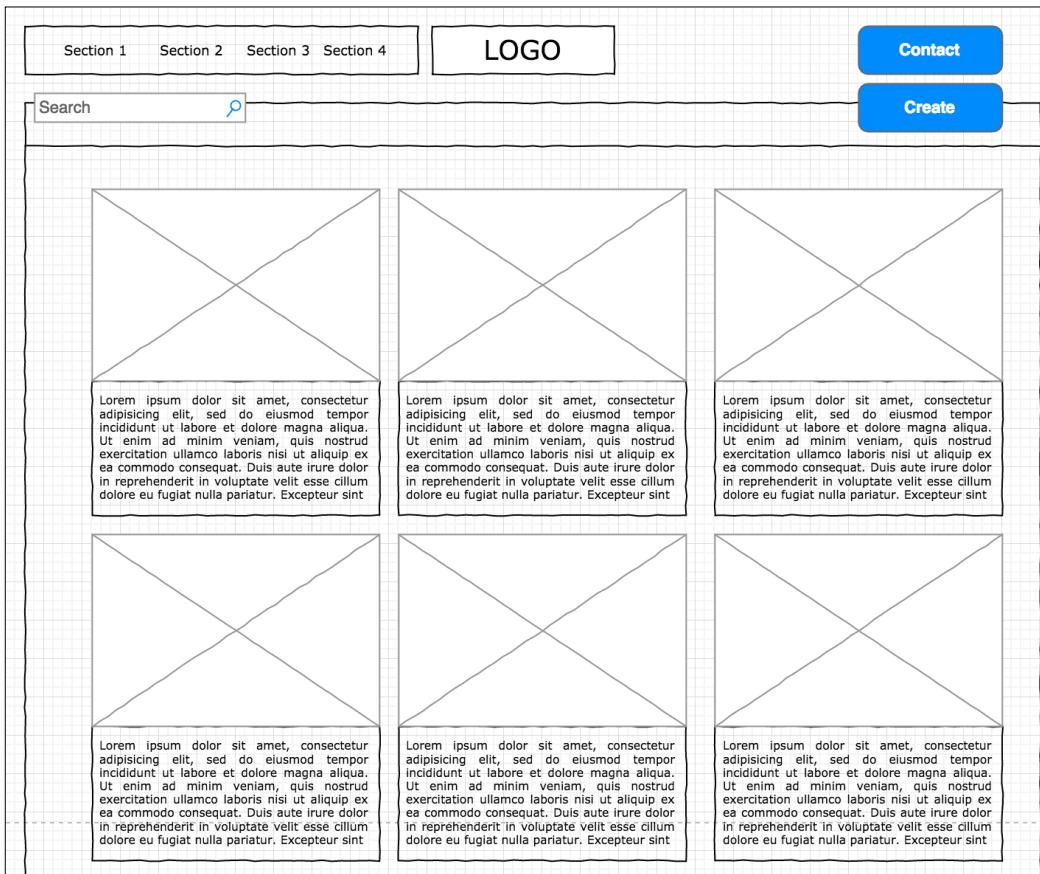
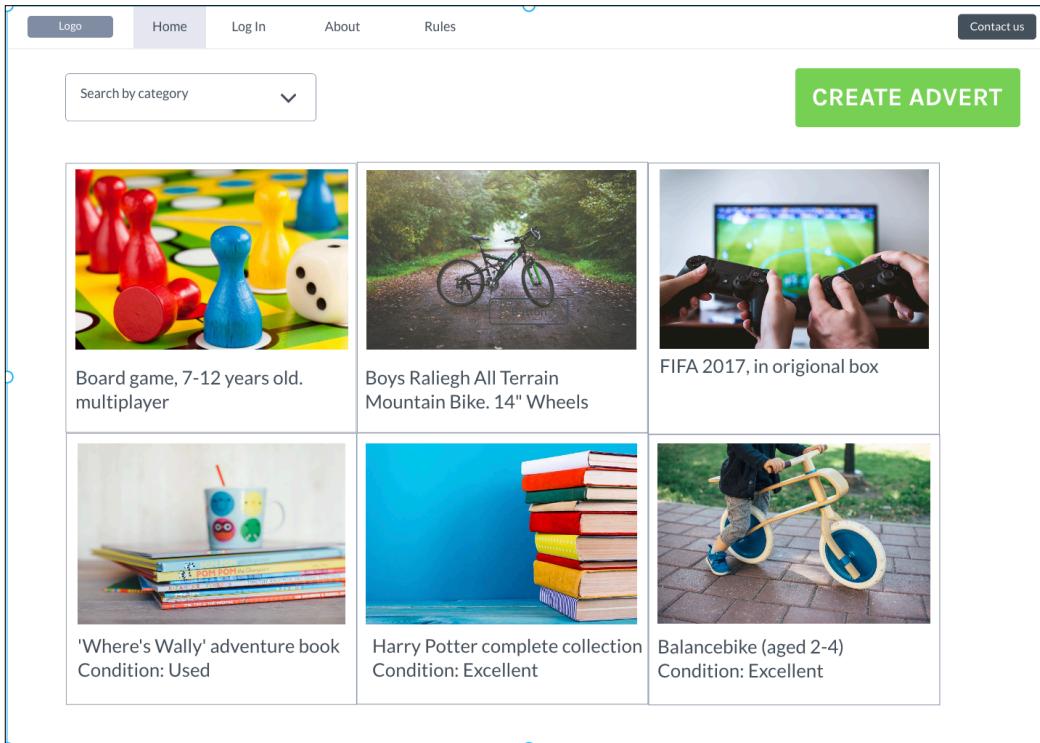
Unit	Ref	Evidence
P	P.5	User Site Map

Site map for 'Yodo' Java project.



Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

Taken from the 'Yodo' Java Project:



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

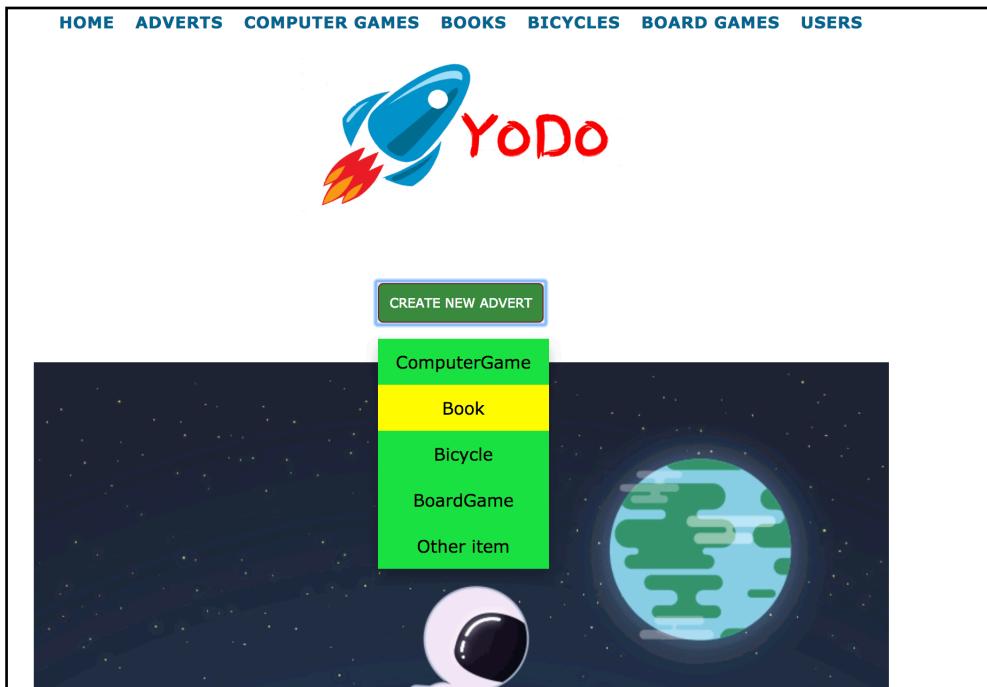
Pseudocode bookmarking a search bar:

```
1 it('should search for advert using keyword', function(){
2   # a search bar that will return adverts
3   # if the term entered appears in that advert
4   # possibly using the 'like' operator in java
5 })
```

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

User adding 'book' to our online shop ('YoDo' project, Java)

STEP ONE: User selects drop down category



STEP TWO: User adds the details:

 A screenshot of the 'YoDo' application showing a 'New book' creation form. The form includes fields for Title ('The Tiger Who Came To'), Description ('He had a good appetite'), Price ('3'), imageUrl ('https://img.com/images/'), User ('Grace | user id: 7'), Genre ('Animals'), and Format ('Paperback'). There is also a 'Save' button at the bottom of the form.

STEP THREE: New advert appears

HOME ADVERTS COMPUTER GAMES BOOKS BICYCLES BOARD GAMES USERS



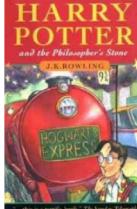
[CREATE NEW ADVERT](#)

All books (3)



The Railway Series
Box of 26 classic books by the Rev. W Awdry. The Fat Controller pre-dates Intellijj.
Price: 20Y
User: [Ferdinand](#)
Genre: Travel
Format: Hardback

[Edit](#) [Delete](#)



Harry Potter and the Philosopher's Stone
Book by J. K. Rowling
Price: 5Y
User: [Ferdinand](#)
Genre: Fantasy
Format: Hardback

[Edit](#) [Delete](#)



The Tiger Who Came to Tea
He had a good appetite actually. By Judith Kerr
Price: 3Y
User: [Grace](#)
Genre: Animals
Format: Paperback

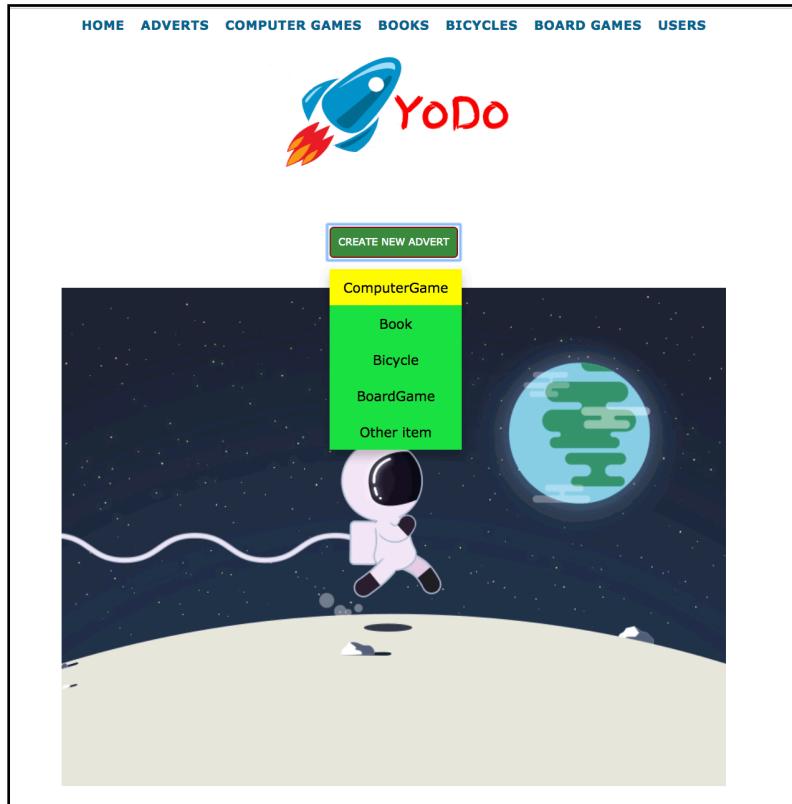
[Edit](#) [Delete](#)

[CREATE NEW ADVERT](#)

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

User adding 'computer game' to our online shop ('YoDo' project, Java)

STEP ONE: User selects drop down category



STEP TWO: User adds all relevant info:

 A screenshot of a web browser displaying a form for creating a new computer game on the 'YoDo' website. The page has a dark blue header with the word 'YoDo' in red and white. Below the header is a navigation bar with links: HOME, ADVERTS, COMPUTER GAMES, BOOKS, BICYCLES, BOARD GAMES, and USERS. The main content area features a large, bold heading 'New computer game'. Below the heading is a form with the following fields:

- Title: Mario Bros.
- Description: Mario and Luigi in the se
- Price: 6
- imageUrl: <https://iimg.com/images>
- User: Benedict | user id: 2
- Console: Super NES
- Age Classification: 3
- Game Type: Action

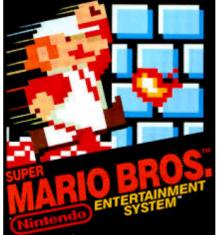
 At the bottom of the form is a 'Save' button.

STEP THREE: Advert is placed/displayed

HOME ADVERTS COMPUTER GAMES BOOKS BICYCLES BOARD GAMES USERS



CREATE NEW ADVERT

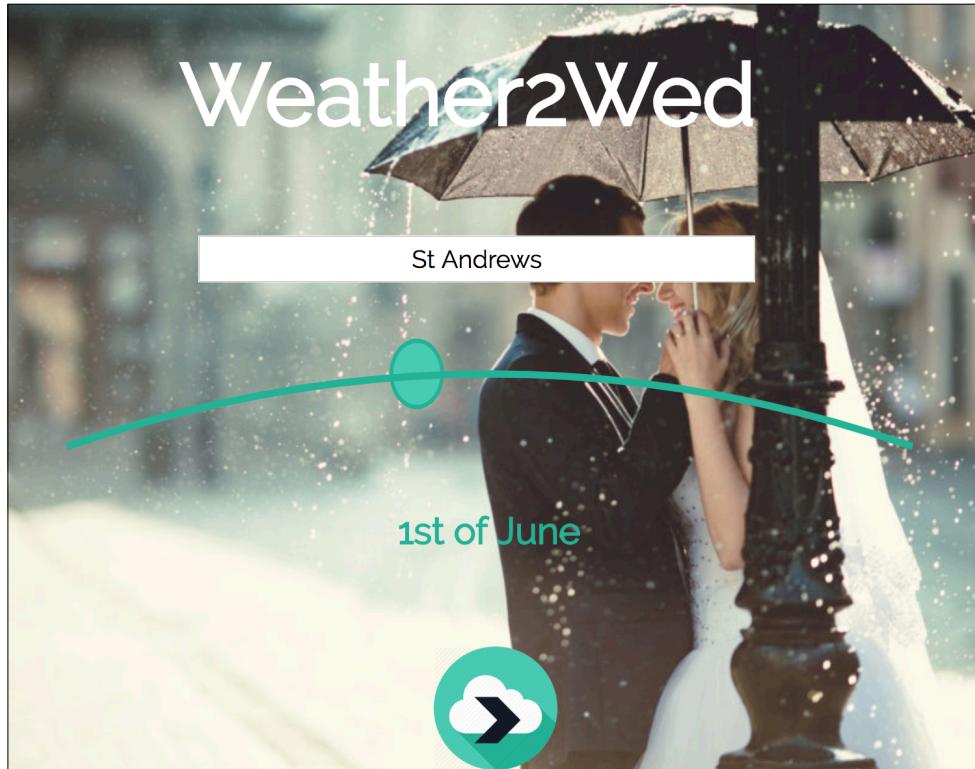


Mario Bros.
Mario and Luigi in the sewers!
Price: 6Y
Seller: Benedict
Console: Super NES
Age rating: 3
Type: Action

Edit Delete

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

User enters a request for weather data based on 'location' and 'date'



User is returned weather information for that date/location

Typical weather for the 1st of June: Mostly cloudy throughout the day.

Average temp:
18°C

Chance of rain:
0%

Sunrise:
4:31 am
Sunset:
9:49 pm

+
-

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

Testing for a Barbarians name in a game:

- test initially fails as his name is spelt incorrectly:

```

5  public class BarbarianTest {
6
7
8
9
10 Club club;
11 Barbarian barbarian;
12
13 @Before
14 public void before() {
15     club = new Club();
16     barbarian = new Barbarian( name: "Barry", healthPoints: 80, club);
17 }
18
19 @Test
20 public void getName() {
21     assertEquals( expected: "Barrie", barbarian.getName());
22 }
BarbarianTest > getName()

>> ⓘ Tests failed: 1 of 1 test - 21 ms
21ms /Library/Java/JavaVirtualMachines/jdk1.8.0_172.jdk/Contents/Home/bin/java ...
21ms
org.junit.ComparisonFailure:
Expected :Barrie
Actual   :Barry
<Click to see difference>

```

- test then passes as mistake is rectified:

```

5  public class BarbarianTest {
6
7
8
9
10 Club club;
11 Barbarian barbarian;
12
13 @Before
14 public void before() {
15     club = new Club();
16     barbarian = new Barbarian( name: "Barry", healthPoints: 80, club);
17 }
18
19 @Test
20 public void getName() {
21     assertEquals( expected: "Barry", barbarian.getName());
22 }
BarbarianTest > getName()

>> ✅ Tests passed: 1 of 1 test - 1 ms
1ms /Library/Java/JavaVirtualMachines/jdk1.8.0_172.jdk/Contents/Home/bin/java ...
1ms
Process finished with exit code 0

```

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

Example of an 'ISell' Interface in a 'Music Shop' project.

Here both drum sticks and drum skins can be 'sold' - i.e we can perform a single action in different ways.

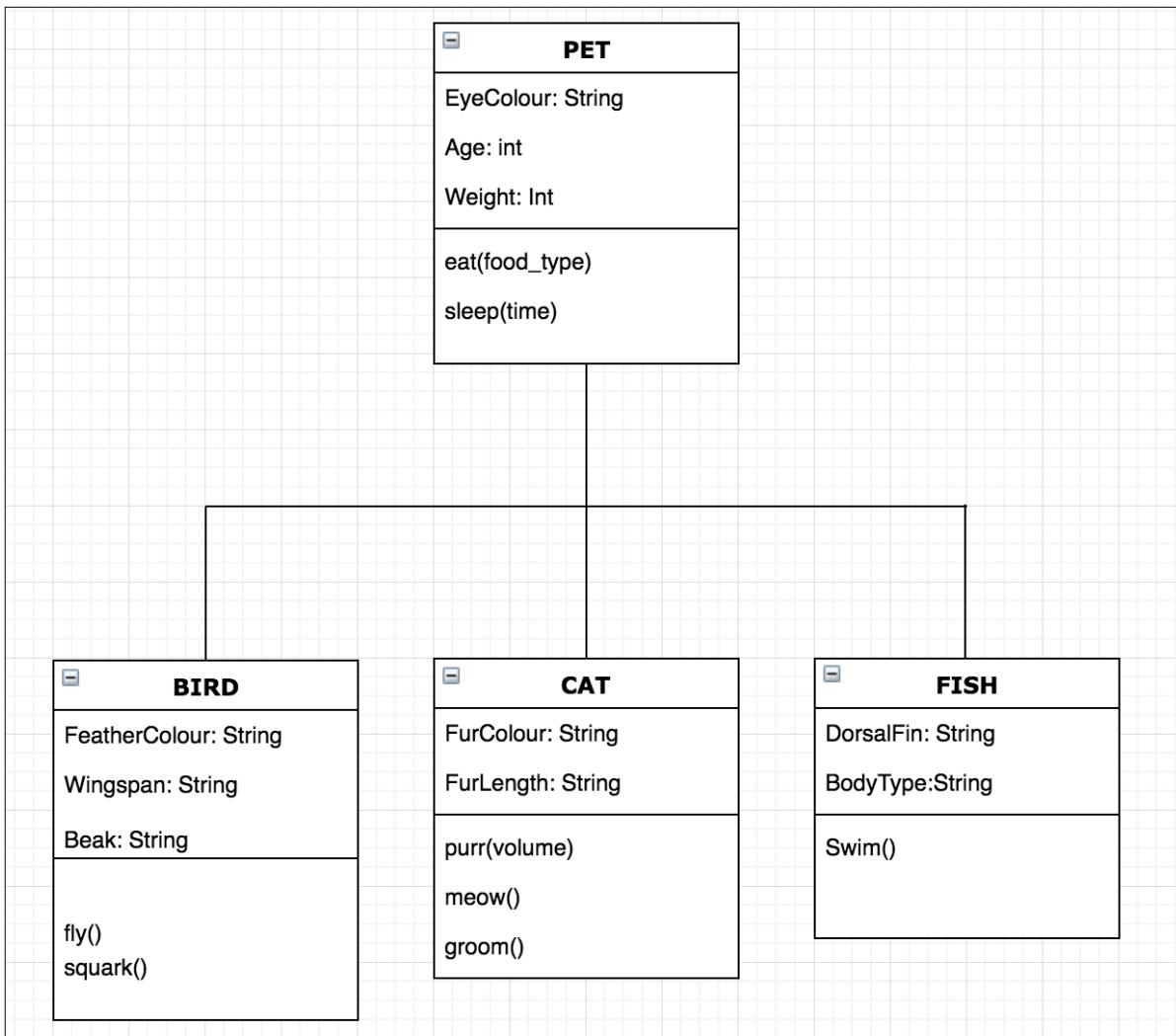
```

1 package Accessories;
2
3 public interface ISell {
4
5     double calculateMarkup();
6
7     double getBuyingPrice();
8
9     double getSellingPrice();
10
11 }
12
13
14 package Accessories;
15
16 public class Drumsticks implements ISell {
17
18     private String type;
19     private double buyingPrice;
20     private double sellingPrice;
21
22     public Drumsticks(String type, double buyingPrice, double sellingPrice) {
23         this.type = type;
24         this.buyingPrice = buyingPrice;
25         this.sellingPrice = sellingPrice;
26     }
27
28     public String getType() { return type; }
29
30     public double getBuyingPrice() { return buyingPrice; }
31
32     public double getSellingPrice() { return sellingPrice; }
33
34     @Override
35     public double calculateMarkup() { return sellingPrice - buyingPrice; }
36
37 }
38
39
40 
```

```

1 package Accessories;
2
3 public class Drumskins implements ISell {
4
5     private String type;
6     private double buyingPrice;
7     private double sellingPrice;
8
9     public Drumskins(String type, double buyingPrice, double sellingPrice) {
10         this.type = type;
11         this.buyingPrice = buyingPrice;
12         this.sellingPrice = sellingPrice;
13     }
14
15     public String getType() { return type; }
16
17     public double getBuyingPrice() { return buyingPrice; }
18
19     public double getSellingPrice() { return sellingPrice; }
20
21     @Override
22     public double calculateMarkup() { return sellingPrice - buyingPrice; }
23
24 }
25
26
27 
```

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



Inheritance. Inheritance means that objects can inherit the traits and functions of other objects. Think of a cat. It can inherit characteristics of a 'pet', but also has its own unique characteristics.

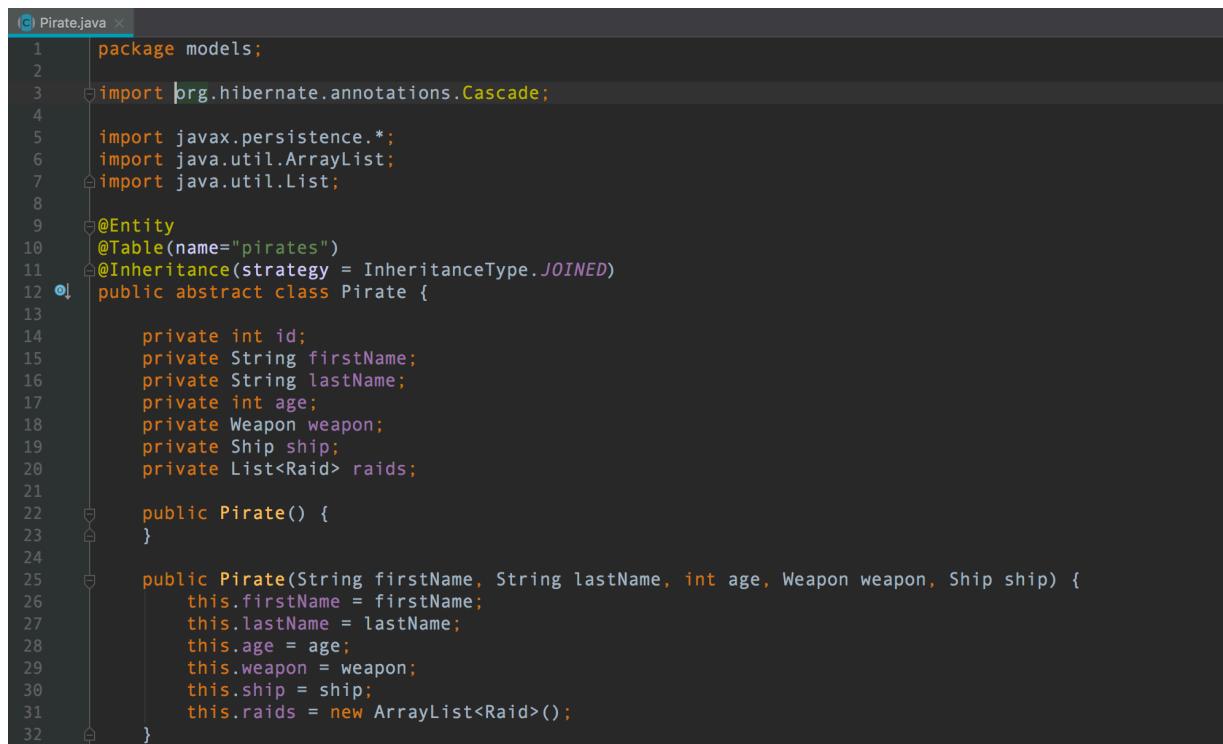
Unit	Ref	Evidence
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.

A screenshot of a Java code editor showing the `BankCustomer.java` file. The code defines a class `BankCustomer` with private fields `first_name`, `surname`, and `email`. It includes a constructor that initializes these fields and three getter methods: `getFirst_name`, `getSurname`, and `getEmail`. The code is color-coded, and the editor interface is visible at the top.

```
BankCustomer.java
1 public class BankCustomer {
2     private String first_name;
3     private String surname;
4     private String email;
5
6     public BankCustomer(String first_name, String surname, String email) {
7         this.first_name = first_name;
8         this.surname = surname;
9         this.email = email;
10    }
11
12    public String getFirst_name() {
13        return this.first_name;
14    }
15
16    public String getSurname() {
17        return this.surname;
18    }
19
20    public String getEmail() {
21        return this.email;
22    }
23
24 }
25 }
```

Unit	Ref	Evidence
I&T	I.T.2	<p>Take a screenshot of the use of Inheritance in a program. Take screenshots of:</p> <ul style="list-style-type: none"> *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.

Example: Class ‘Pirates’ (Abstract class)

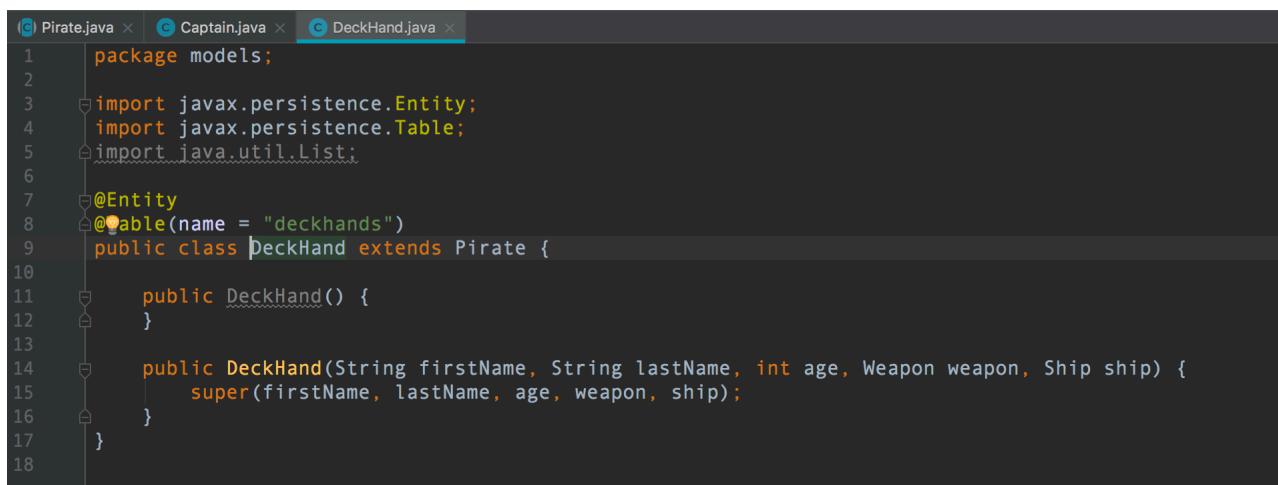


```

1 package models;
2
3 import org.hibernate.annotations.Cascade;
4
5 import javax.persistence.*;
6 import java.util.ArrayList;
7 import java.util.List;
8
9 @Entity
10 @Table(name="pirates")
11 @Inheritance(strategy = InheritanceType.JOINED)
12 public abstract class Pirate {
13
14     private int id;
15     private String firstName;
16     private String lastName;
17     private int age;
18     private Weapon weapon;
19     private Ship ship;
20     private List<Raid> raids;
21
22     public Pirate() {
23     }
24
25     public Pirate(String firstName, String lastName, int age, Weapon weapon, Ship ship) {
26         this.firstName = firstName;
27         this.lastName = lastName;
28         this.age = age;
29         this.weapon = weapon;
30         this.ship = ship;
31         this.raids = new ArrayList<Raid>();
32     }

```

The class ‘Deckhand’ **inherits** from ‘Pirate’, creates a ‘deckhand’ object. It has the ‘super’ characteristics from the abstract class, plus its own characteristics.

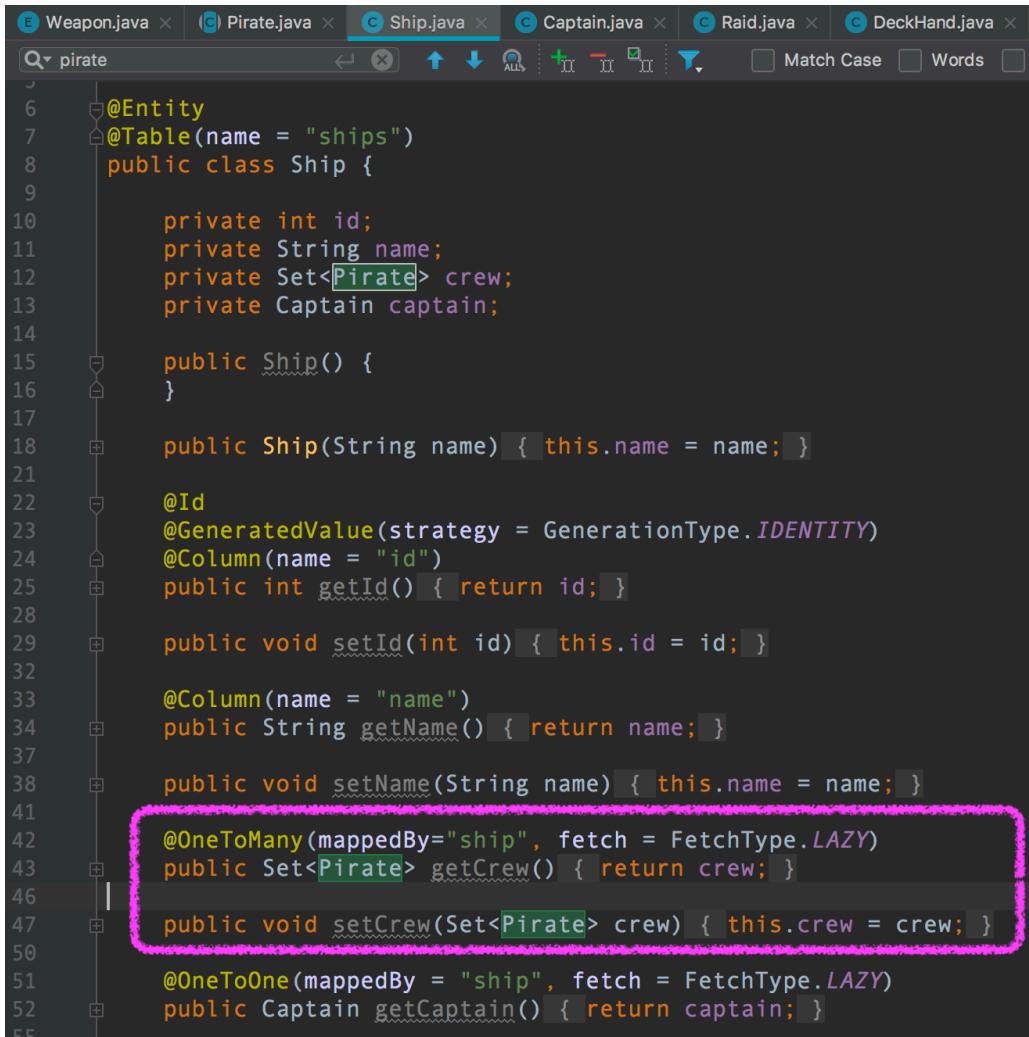


```

1 package models;
2
3 import javax.persistence.Entity;
4 import javax.persistence.Table;
5 import java.util.List;
6
7 @Entity
8 @Table(name = "deckhands")
9 public class DeckHand extends Pirate {
10
11     public DeckHand() {
12     }
13
14     public DeckHand(String firstName, String lastName, int age, Weapon weapon, Ship ship) {
15         super(firstName, lastName, age, weapon, ship);
16     }
17 }

```

The method that uses the information (pirates) inherited from another class



```

6  @Entity
7  @Table(name = "ships")
8  public class Ship {
9
10     private int id;
11     private String name;
12     private Set<Pirate> crew;
13     private Captain captain;
14
15     public Ship() {
16     }
17
18     public Ship(String name) { this.name = name; }
19
20     @Id
21     @GeneratedValue(strategy = GenerationType.IDENTITY)
22     @Column(name = "id")
23     public int getId() { return id; }
24
25     public void setId(int id) { this.id = id; }
26
27     @Column(name = "name")
28     public String getName() { return name; }
29
30     public void setName(String name) { this.name = name; }
31
32     @OneToMany(mappedBy="ship", fetch = FetchType.LAZY)
33     public Set<Pirate> getCrew() { return crew; }
34
35     public void setCrew(Set<Pirate> crew) { this.crew = crew; }
36
37     @OneToOne(mappedBy = "ship", fetch = FetchType.LAZY)
38     public Captain getCaptain() { return captain; }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

```

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

The library Project:

Github Link: <https://github.com/DavidAPears/The-Library-Ruby--Solo-Project->

Loans							
Surname	Book	Loan Start Date	Length Of Loan (days)	Has Book Been Returned?	Edit Existing loan	Book Returned	
Smith	Harry Potter	2018-06-22	14	No			
Person	A Brief History of Time	2018-06-22	14	Yes			
Sanchez	The Great Gatsby	2018-09-10	14	No			
Ants	The Old Man and the Sea	2018-09-10	7	Yes			

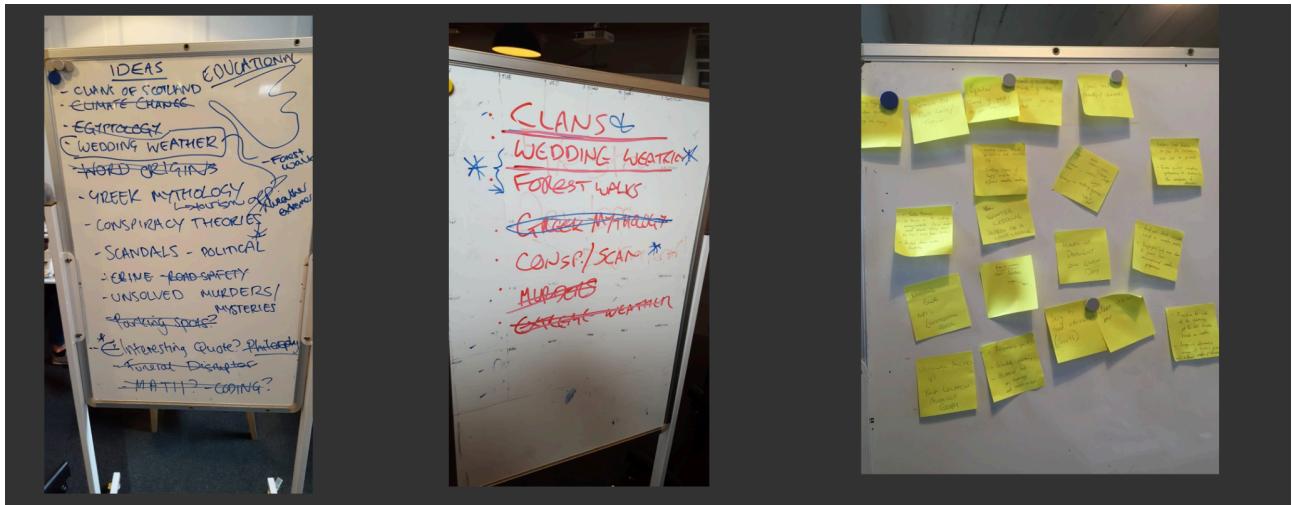
Members							
First Name	Last Name	Post Code	E-mail	Active Membership	Edit Member	Deactivate Membership	
Bird	Person	EH2 3UT	birdperson@birdworld.tv	Yes			
Million	Ants	EH3 6NS	ant@colony.com	Yes			
David	Pears	YO23 1NR	davidaepears@gmail.com	No			
Rick	Sanchez	EH8 8EY	wubalubadubdub@gmail.com	No			
Morty	Smith	EH7 1SL	ohjeez@ohjeez.com	Yes			

Books							
Title	Author	DavidAPears / The-Library-Ruby--Solo-Project-					
The Great Gatsby	F Scott Fitzgerald		Issues 0	Pull requests 0	Projects 0	Wiki	Insights
Harry Potter	J K Rowling		Issues 0	Pull requests 0	Projects 0	Wiki	Settings
A Brief History of Time	Stephen Hawking		Issues 0	Pull requests 0	Projects 0	Wiki	Insights
The Old Man and the Sea	Ernest Hemingway		Issues 0	Pull requests 0	Projects 0	Wiki	Settings

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

Planning for 'Weather2Wed' app
A JavaScript project

The below shows the concept being narrowed down from a huge list of initial ideas to a single idea, and then brain storming on features that were to appear on the project.



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.

```

sort.java x
1
2
3
4
5   for (int i = 0; i < n-1; i++)
6     for (int j = 0; j < n-i-1; j++)
7       if (arr[j] > arr[j+1])
8       {
9         //swap temp and arr[i]
10        int temp = arr[j];
11        arr[j] = arr[j + 1];
12        arr[j + 1] = temp;
13      }
14    }
15
16  // Prints the array
17  void printArray(int arr[]){
18    int n = arr.length;
19    for (int i=0; i<n; ++i)
20      System.out.println(arr[i] + " ");
21    System.out.println();
22  }
23
24 // Driver method to test above
25 public static void main(String args[]){
26   BubbleSort ob = new BubbleSort();
27   int arr[] = {64, 34, 25, 12, 22, 11, 90};
28   ob.bubbleSort(arr);
29   System.out.println("sorted array");
30   ob.printArray(arr);
31 }
32 }
33

```

The examples below are of a ‘bubble sort’ and an ‘insert sort’ both are used to sort arrays. These are both fairly easy to implement and work well. Neither of these were part of projects per se, just

Sorted array
11 12 22 25 34 64 90

something I studied to understand the concept.

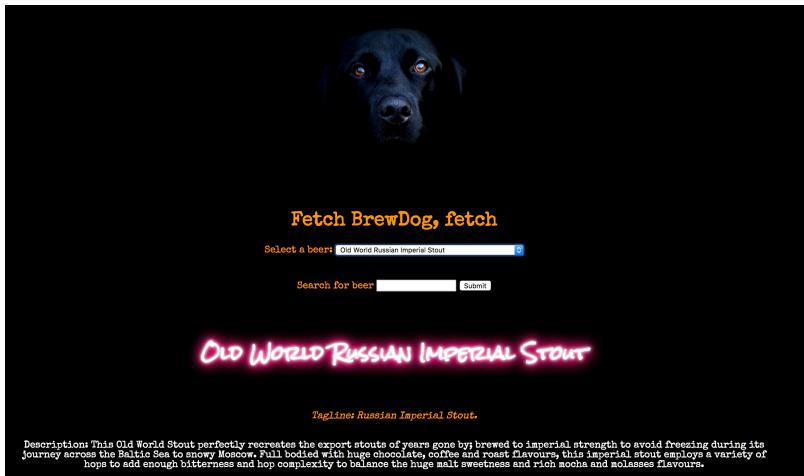
```

sort.java x InsertionSort.java x
1
2
3  public class InsertionSort {
4
5    void sort(int arr[]) {
6      int n = arr.length;
7      for (int i = 1; i < n; ++i) {
8        int key = arr[i];
9        int j = i - 1;
10       while (j >= 0 && arr[j] > key) {
11         arr[j + 1] = arr[j];
12         j = j - 1;
13       }
14       arr[j + 1] = key;
15     }
16   }
17
18  static void printArray(int arr[]) {
19    int n = arr.length;
20    for (int i = 0; i < n; ++i)
21      System.out.print(arr[i] + " ");
22
23    System.out.println();
24  }
25
26  public static void main(String args[]) {
27    int arr[] = {12, 11, 13, 5, 6};
28
29    InsertionSort ob = new InsertionSort();
30    ob.sort(arr);
31
32    printArray(arr);
33  }
34

```

Output: 5, 6, 11, 12, 13

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



The ‘Brewdog’ API being called to return info on their beer selection (limited to 80 pages per page).

```

JS beer_detail_view.js x JS beer_form_view.js x JS beer_list_view.js x JS select_beer_view.js x JS beers.js x
1 const Request = require('../helpers/request_helper.js');
2 const PubSub = require('../helpers/pub_sub.js');

3
4 const Beers = function () {
5   this.data = null;
6 }

7
8 // OPTION A 'GETTER': Get request, returns all data from API (all beers)
9 Beers.prototype.getData = function () {
10   const url = `https://api.punkapi.com/v2/beers?page=2&per_page=80`;
11   const request = new Request(url);
12   request.get()
13     .then((data) => {
14       this.data = data
15       PubSub.publish('Beers:beers-ready', this.data)
16     })
17     .catch((message) => {
18       console.error(message);
19     });
20 }

```

The code shows the API being used to populate the dropdown search box and returning info on that particular beer once one has been selected (see screen shot).

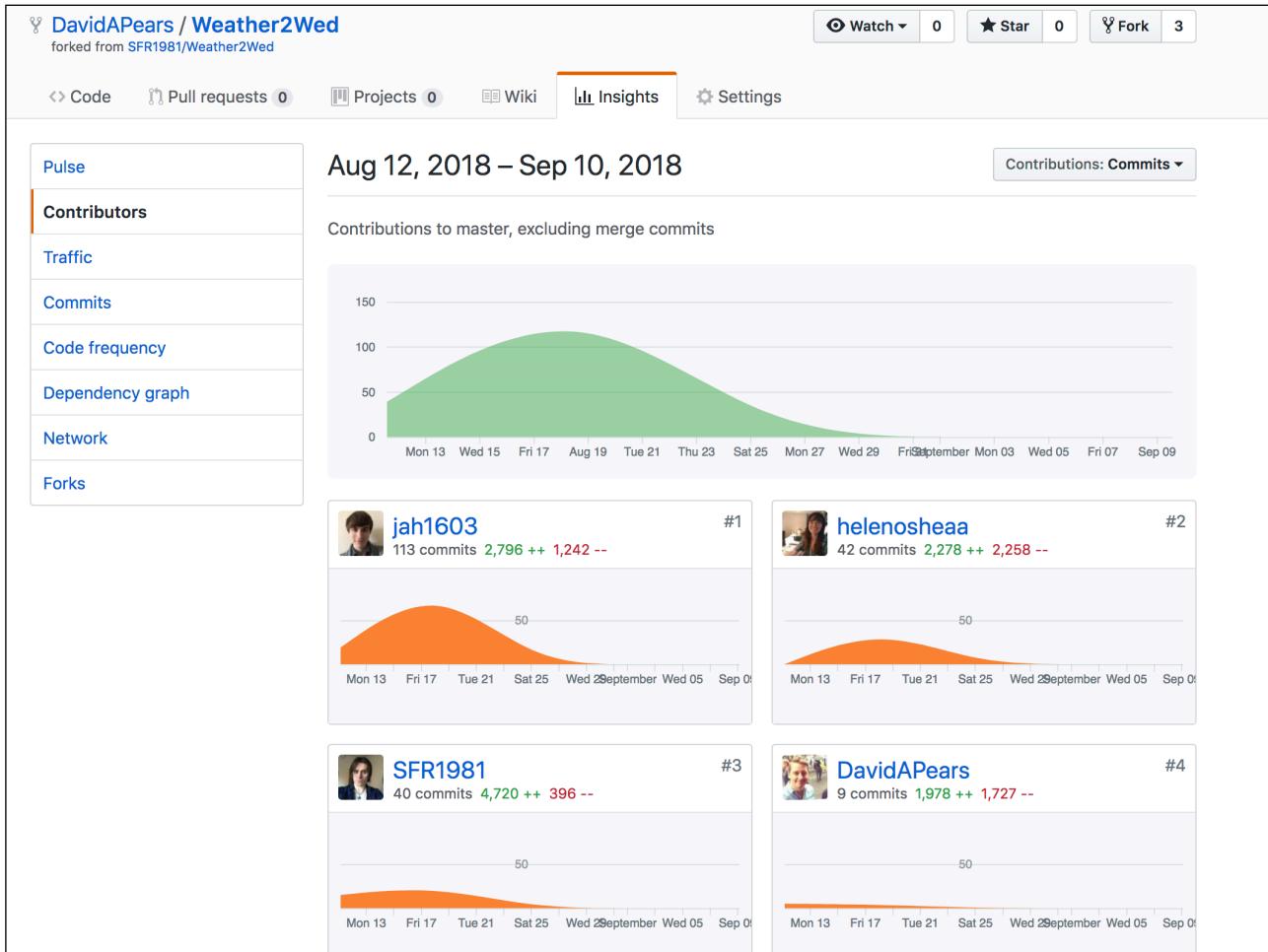
```

JS app.js JS beers.js JS index.html JS request_helper.js JS beer_detail_view.js JS beers.js
1 // SELECT_BEER_VIEW = BEER DROPDOWN BOX
2
3 const PubSub = require('../helpers/pub_sub.js');
4
5 const SelectBeerView = function (selectElement) {
6   this.element = selectElement;
7 };
8
9 SelectBeerView.prototype.bindEvents = function () {
10   PubSub.subscribe('Beers:beers-ready', (evt) => {
11     this.populate(evt.detail)
12   });
13
14   this.element.addEventListener('change', (evt) => {
15     const selectedIndex = evt.target.value;
16     PubSub.publish('SelectView:change', selectedIndex);
17   });
18 };
19
20 SelectBeerView.prototype.populate = function (beers) {
21   beers.forEach((beer, index) => {
22     const beerOption = this.createOption(beer.name, index);
23     this.element.appendChild(beerOption);
24   });
25 };
26
27 SelectBeerView.prototype.createOption = function (name, index) {
28   const option = document.createElement('option');
29   option.textContent = name;
30   option.value = name;
31   return option;
32 };
33
34 module.exports = SelectBeerView;

```

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.



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

The brief

You have been approached by a wedding magazine looking to improve their online offering of wedding content by developing some interactive apps that display information in a fun and interesting way.

MVP

Display some information about a particular topic in an interesting way.

Have some interactivity that enables a user to move through different sections of content.

Examples of further features
Bring in data using an API or create your own.

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

Planning

Weather | Weather2Wed | Team Visible | S JH 4 | +

Must have	Should have	Could have	Won't have (this time)
Clean search bar	Accommodation recommendations by location you search	Instagram feed of weddings	Address book feature
Search by postcode/town	Wedding venues from a location searched	Comparison of a chosen location with a colder location	Wedding guest list
Easy to read information	Daylight hours	Filter by weather preference	Astronomy constellations of wedding date
Responsive design	Full Moon (astronomy api?)	Popular times of year to get married	Saving searches / login
Map location you've typed in	+ Add another card	Wedding inspirational quote	Wedding content tailored to weather result
Moon phases		Speech to text input	+ Add another card
+ Add another card		+ Add another card	

Kanban | Weather2Wed | Team Visible | S 3 | +

To Do	Doing	Done
+ Add a card	Presentation	4 square
	+ Add another card	Tagging for CSS
		Results View

Planning(wireframes)

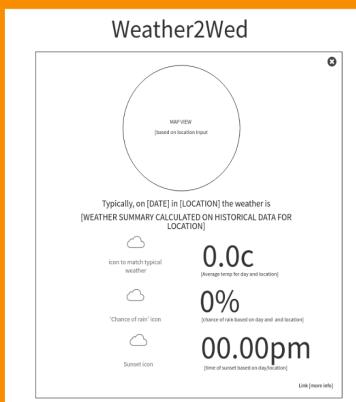
Tip

Visualise the product

Weather2Wed

Search (location)

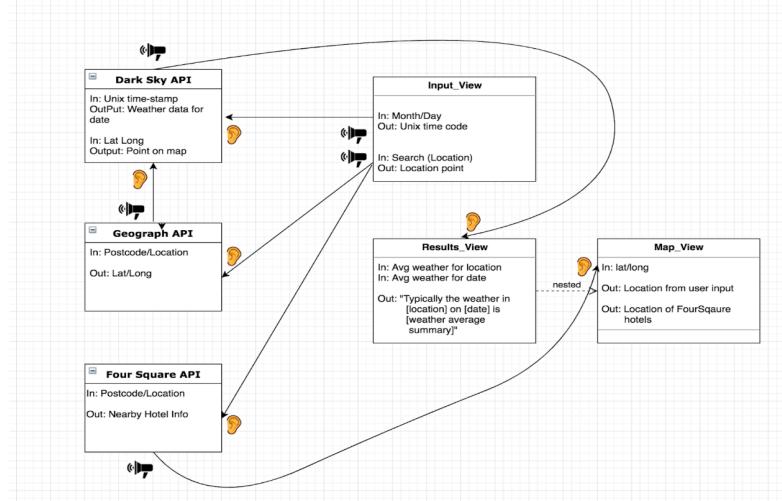
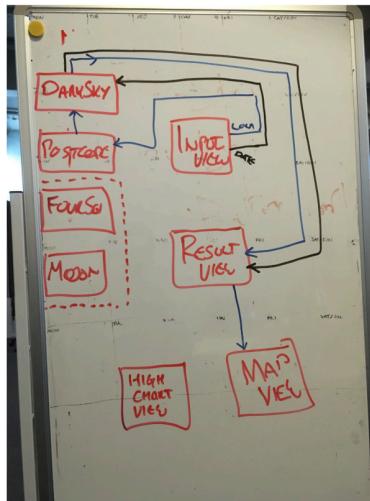
12 May 2016



Planning(models)

Tip

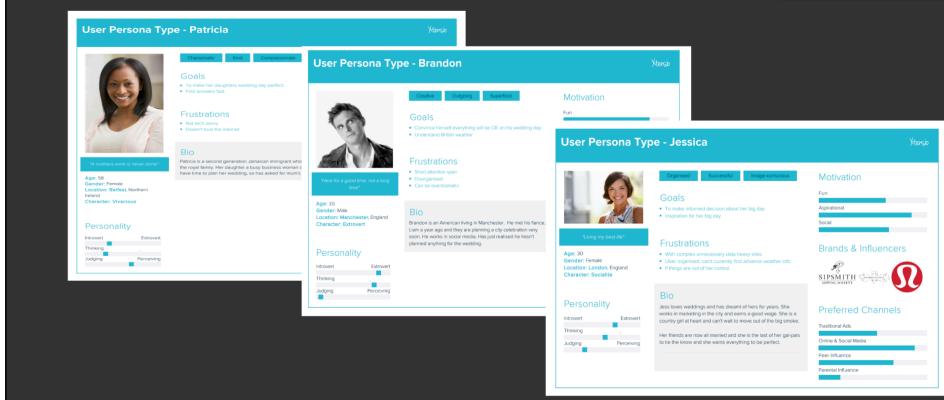
Understand your data flow



Planning(UX)

Tip

User tests on friends & family



A sample of the planning phase of the group project (taken from our group presentation)

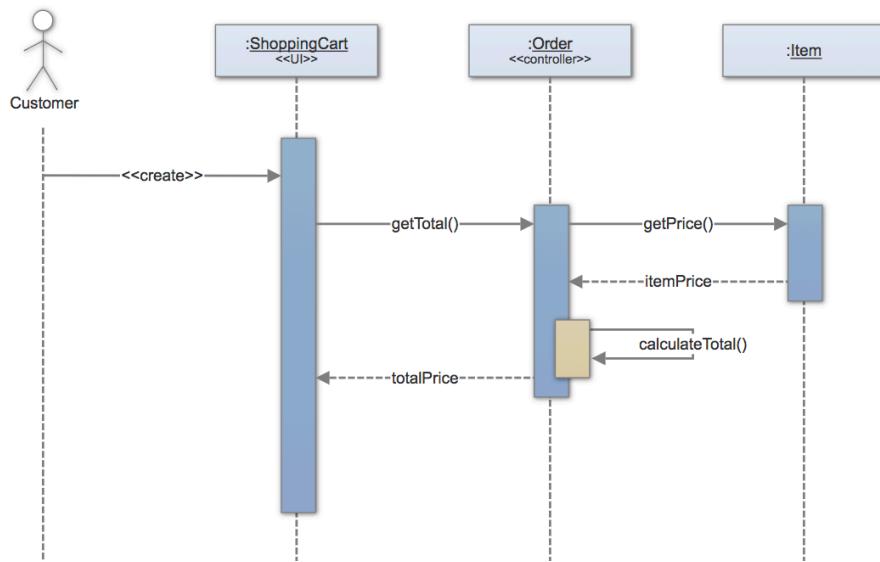
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

Criteria for 'Weather2Wed' app

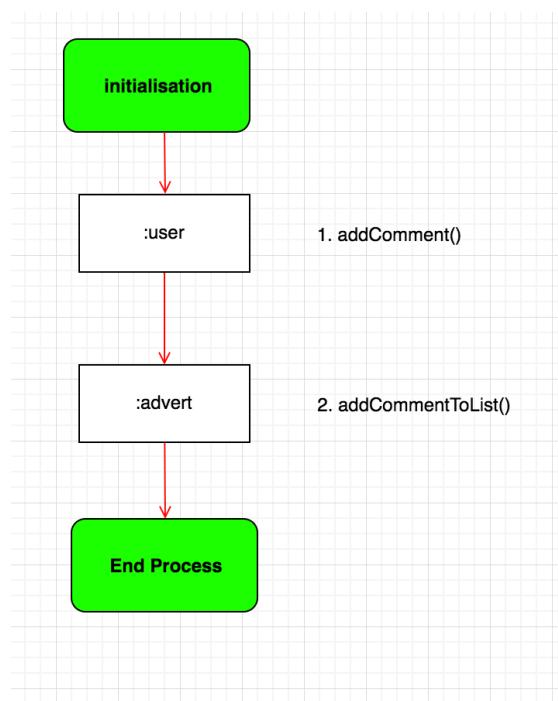
Acceptance Criteria	Expected Result	Pass / Fail ?
Can user enter date and location of future wedding	User should be able to enter the day (not year) and location of wedding	Pass
Can the user search by location	Weather data is returned using the location entered	Pass
Can the user search by specified date	The 'typical' weather (taken from 40 years of historical data) for given date is returned	Pass
Can user understand weather data	Weather data is displayed in familiar icon format (i.e icons used by other common weather apps)	Pass

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

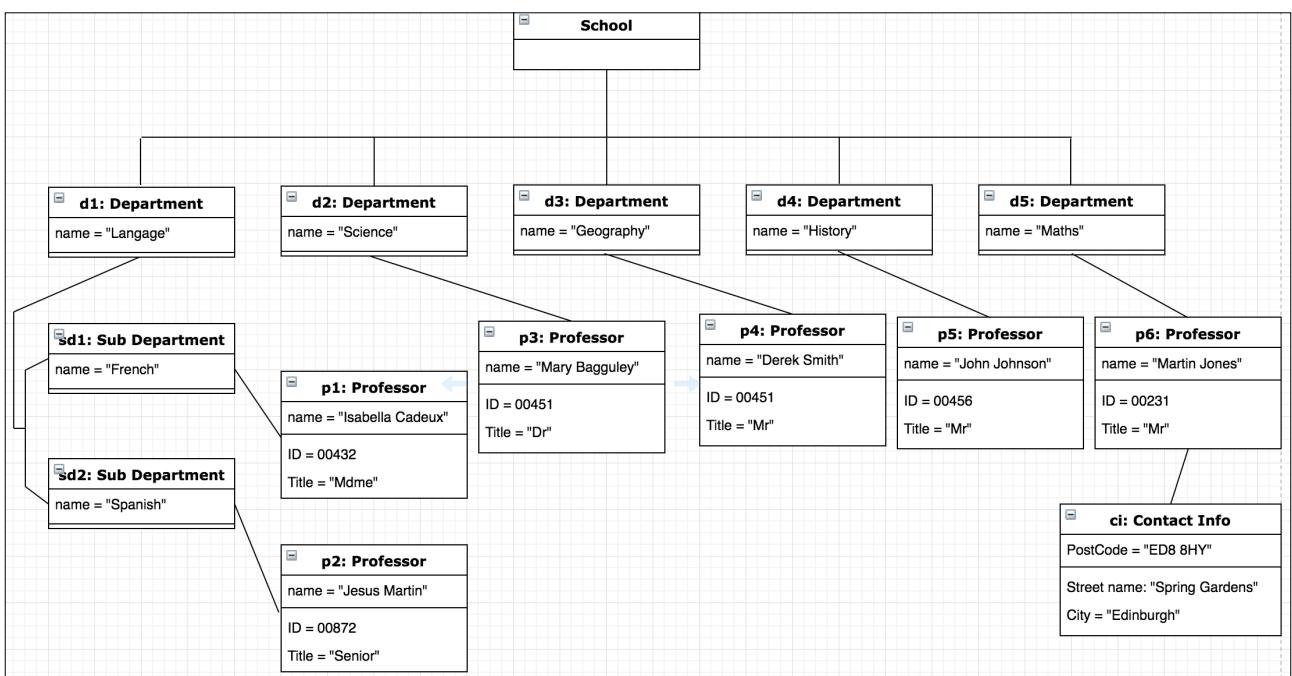
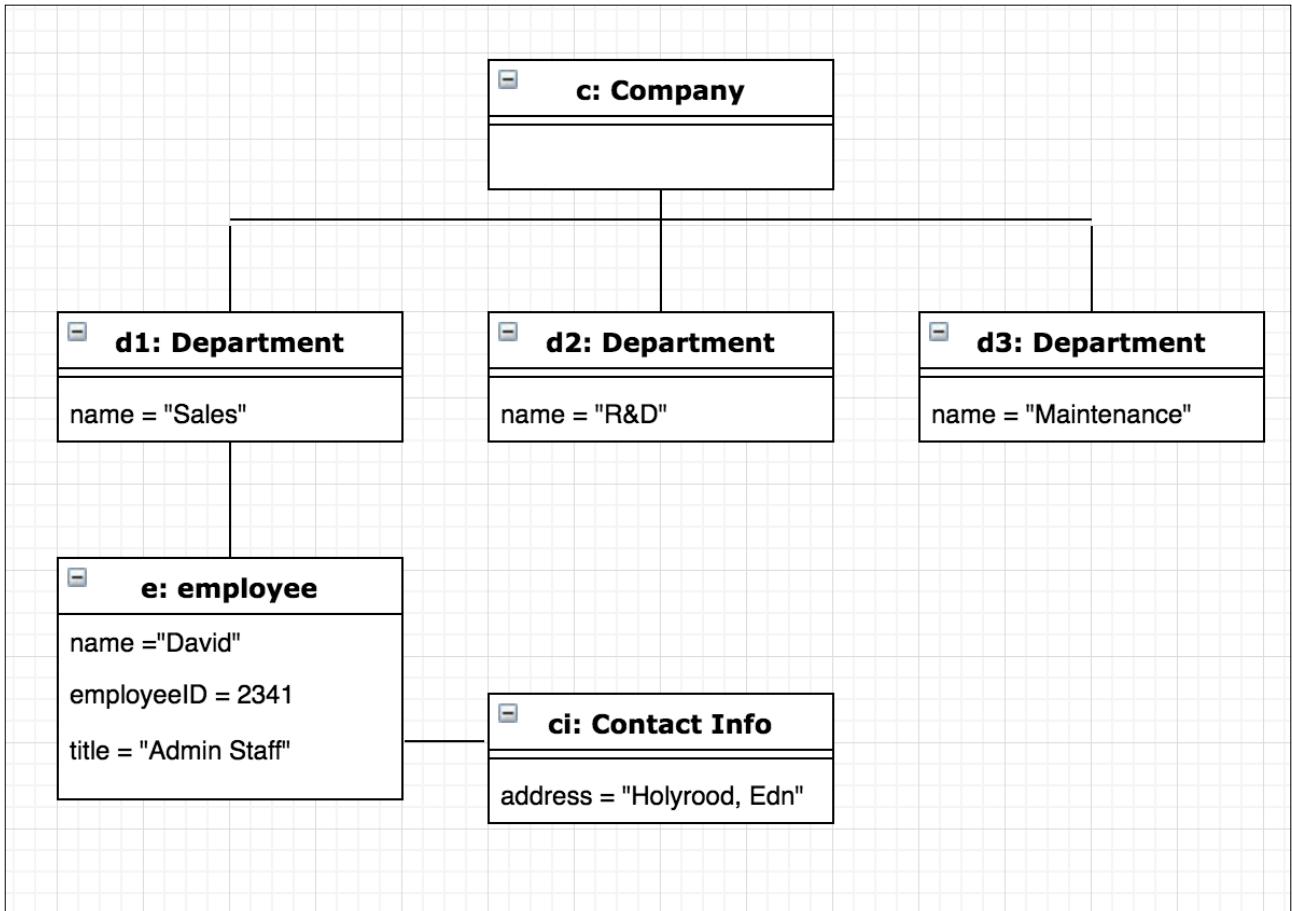
Sequence Diagram for a shopping cart:



Collaboration Diagram showing a user interacting with an advert (i.e leaving a comment on it)



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



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

Sample bug tracking report (as pertaining to the 'Weather2Wed' js app)

BUG		RESOLOUTION	
Weather prediction cannot be given (unless postcode has been entered). I.e search to restrictive	Failed	Fuzzy search added which allows user to enter postcode, town name, landmark or even partial placename	Passed
Day-of-year selector on slide bar/arc does not reset after each submit	Failed	A 'reset' added to ensure that on refresh the day-selector does not show historic selection	Passed
Map will not initialise / open	Failed	Debugged code, it had been looping so opening multiple maps at once.	Passed
Respone method in results (view) runs several times, but only called once	Failed	n/a - ongoing	n/a
API max's out at 100 calls per day	Failed	Changed API from 'geocode.xyz' to 'geograph'	passed