

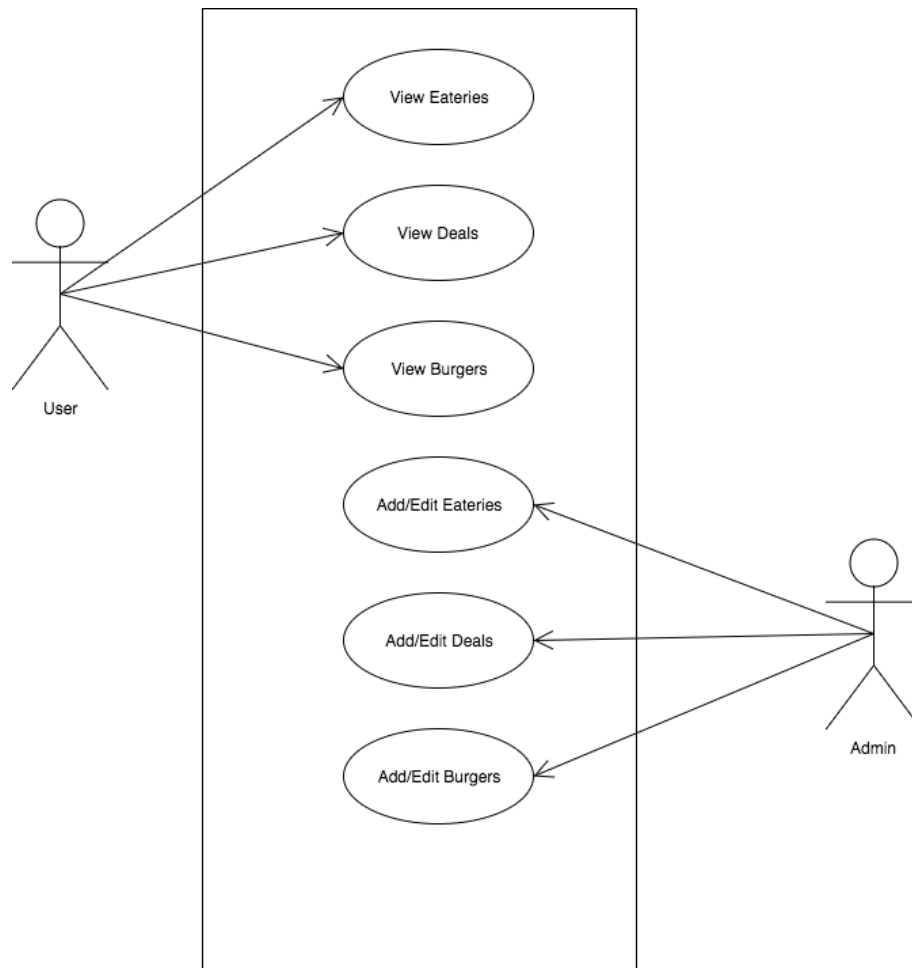


PDA

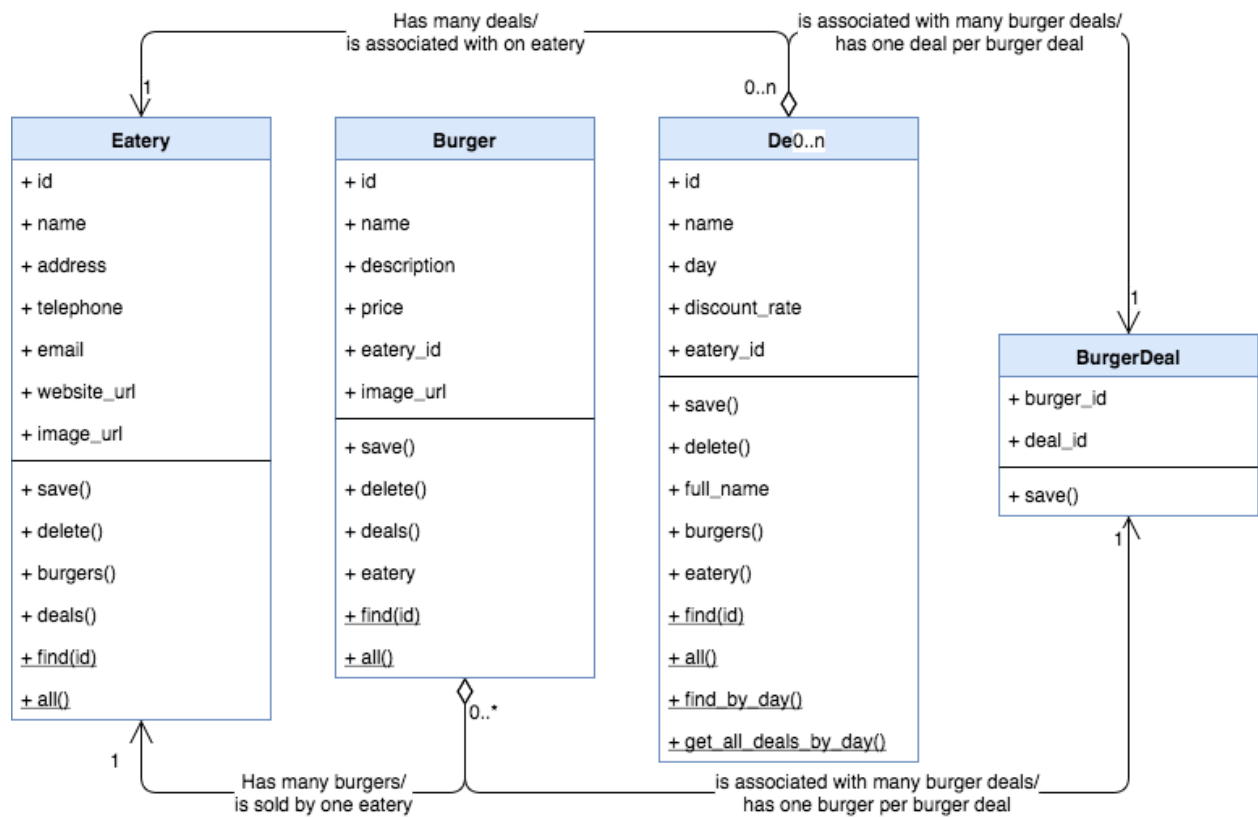
Chris Marshall

Analysis & Design

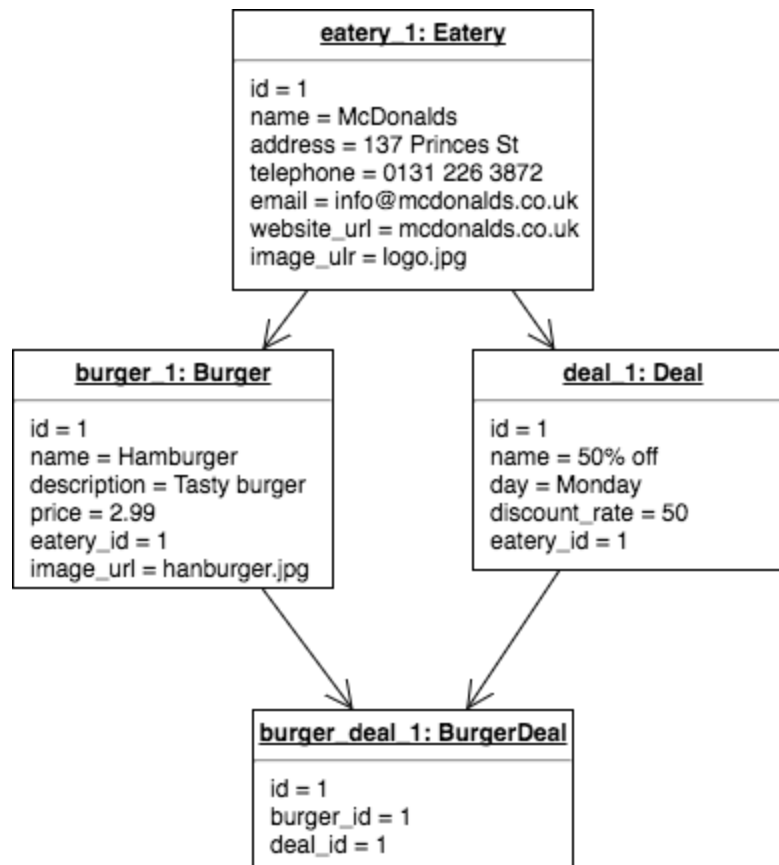
A.D 1 - Use Case Diagram



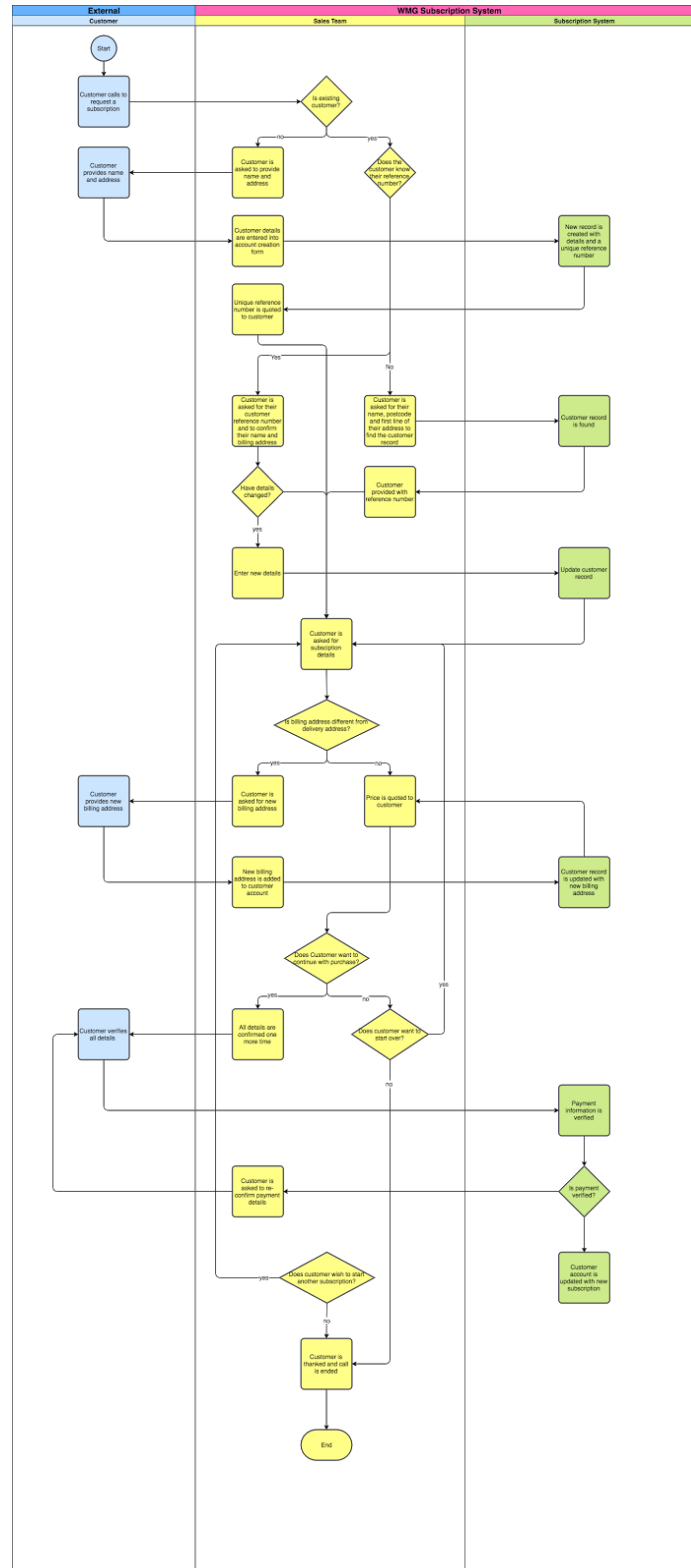
A.D 2 - Class Diagram



A.D 3 - Object Diagram



A.D 4 - Activity Diagram



A.D 5 - Inheritance Diagram

A.D 6 - Implementation Constraints

	Constraint & Possible Effect	Solution
Hardware and Software Platforms	The memory of a computer that a program runs on is a constraint as it limits how much the program can process	Either add more memory or decrease the memory used by the program
Performance Requirements	The program is limited by the number of cycles that the CPU can perform/second.	The performance will be increased if more threads are used
Persistent Storage and transactions	The read capacity of a database can be a limitation as it slows down the speed in which data is retrieved	By using read replicas the performance of a db is increased beyond that of a single instance
Budgets	The budget of a project can be a limitation as it can be a deciding factor on which technologies are used and the hardware that is used for deployment	Use hardware that is value for money based on performance/ £
Time	The delivery time of a project can be a limitation as it determines how quickly the project must be completed. This can result in features being excluded.	Features must be prioritised to ensure that they are not excluded if there are any delays to the project

Implementation & Testing

I.T 1 - Encapsulation

```
public class Item {  
  
    private String name;  
    private double price;  
  
    public Item(String name, double price) {  
        ⚡ this.name = name;  
        this.price = price;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
}
```

I.T 2 - Use of Inheritance

```
public class Item implements Sellable {

    private String name;
    private String description;
    private String category;
    private float price;
    private float buyPrice;

    public Item(String name, String description, String category, float price, float buyPrice) {
        this.name = name;
        this.description = description;
        this.category = category;
        this.price = price;
        this.buyPrice = buyPrice;
    }

    public Item(String name, String category, float price, float buyPrice) {
        this.name = name;
        this.category = category;
        this.price = price;
        this.buyPrice = buyPrice;
    }

    @Override
    public void sell() {
        System.out.println("I have been sold");
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    public float getBuyPrice() {
        return buyPrice;
    }

    public void setBuyPrice(float buyPrice) {
        this.buyPrice = buyPrice;
    }
}
```



```
import java.util.Date;

public class PerishableItem extends Item {

    Date purchaseDate;
    Date sellByDate;

    public PerishableItem(
        String name,
        String description,
        String category,
        float price,
        float buyPrice,
        Date purchaseDate,
        Date sellByDate
    ) {
        super(name, description, category, price, buyPrice);
        this.purchaseDate = purchaseDate;
        this.sellByDate = sellByDate;
    }

    public PerishableItem(
        String name,
        String category,
        float price,
        float buyPrice,
        Date purchaseDate,
        Date sellByDate
    ) {
        super(name, category, price, buyPrice);
        this.purchaseDate = purchaseDate;
        this.sellByDate = sellByDate;
    }

    public float calculateProfit() {
        return this.getPrice() - this.getBuyPrice();
    }
}
```

I.T 3 - Searching

```
people = [  
  {name: 'Chris', age: 24},  
  {name: 'Darren', age: 21}  
]  
  
def find_by_name(people, name)  
  return people.find{ |person|  
    person[:name] == name  
  }  
end  
  
puts find_by_name(people, name 'Chris')
```

```
/Users/chris/.rbenv/versions/2.4.1/bin/ruby  
{:name=>"Chris", :age=>24}
```

```
Process finished with exit code 0
```

I.T 4 - Sorting

```
my_array = [23, 14, 65, 13, 95]

def reverse_sort(array)
  return array.sort { |num_1, num_2|
    num_2 <=> num_1
  }
end

puts reverse_sort(my_array)
```

```
/Users/chris/.rbenv/versions/2.4.1/bin/ruby
```

```
95
```

```
65
```

```
23
```

```
14
```

```
13
```

I.T 5 - Array

```
num_array = [1, 2, 3, 4, 5]

def sum(array)
  sum = 0
  array.each{|number| sum += number}

  return sum
end

puts sum(array, num_array)
```

```
/Users/chris/.rbenv/versions/2.4.1/bin/ruby
```

```
15
```

```
Process finished with exit code 0
```

I.T 6 - Hash

```
my_hash = {  
  name: 'Chris',  
  age: 24  
}  
  
def update(hash, params)  
  hash[:name] = params[:name]  
  hash[:age] = params[:age]  
end  
  
update(my_hash, {  
  name: 'Darren',  
  age: 21  
})  
  
puts my_hash
```

```
/Users/chris/.rbenv/versions/2.4.1/bin/ruby  
{:name=>"Darren", :age=>21}
```

```
Process finished with exit code 0
```

I.T 7 - Polymorphism

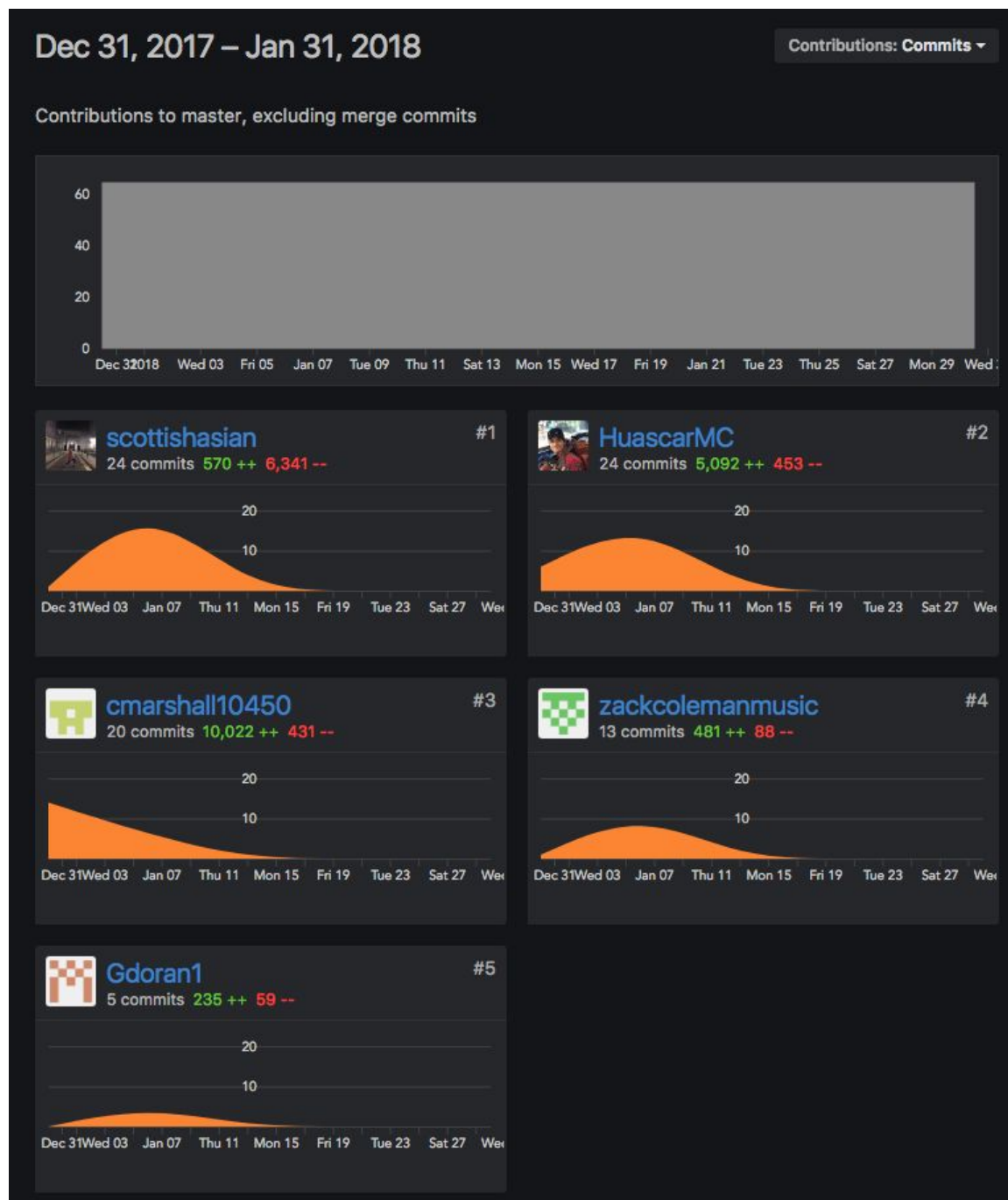
```
public class Shop {  
    private String name;  
    private ArrayList<Item> items;  
  
    public Shop(String name) {  
        this.name = name;  
        this.items = new ArrayList<Item>();  
    }  
  
    public void addItem(Item item) {  
        this.items.add(item);  
    }  
  
    public String getName() {  
        return name;  
    }  
}
```

```
public interface Sellable {  
  
    void sell();  
  
}
```

```
public class Item implements Sellable {  
    private String name;  
    private String description;  
    private String category;  
    private float price;  
    private float buyPrice;  
  
    public Item(String name, String description, String category, float price, float buyPrice) {  
        this.name = name;  
        this.description = description;  
        this.category = category;  
        this.price = price;  
        this.buyPrice = buyPrice;  
    }  
  
    public Item(String name, String category, float price, float buyPrice) {  
        this.name = name;  
        this.category = category;  
        this.price = price;  
        this.buyPrice = buyPrice;  
    }  
  
    @Override  
    public void sell() {  
        System.out.println("I have been sold");  
    }  
}
```


Planning

P 1 - Github Contributors Page



P 2 - Group Project Brief

Educational App

The BBC are looking to improve their online offering of educational content by developing some interactive apps that display information in a fun and interesting way.

Your task is to make an MVP to put forward to them - this may only be for a small set of information, and may only showcase some of the features to be included in the final app. You might use an API to bring in content or a database to store facts. The topic of the app is your choice, but here are some suggestions you could look into:

- Interactive map of a historical event - e.g. World War 1, the travels of Christopher Columbus

MVP

- Display some information about a particular topic in an interesting way
- Have some user interactivity using event listeners, e.g to move through different sections of content

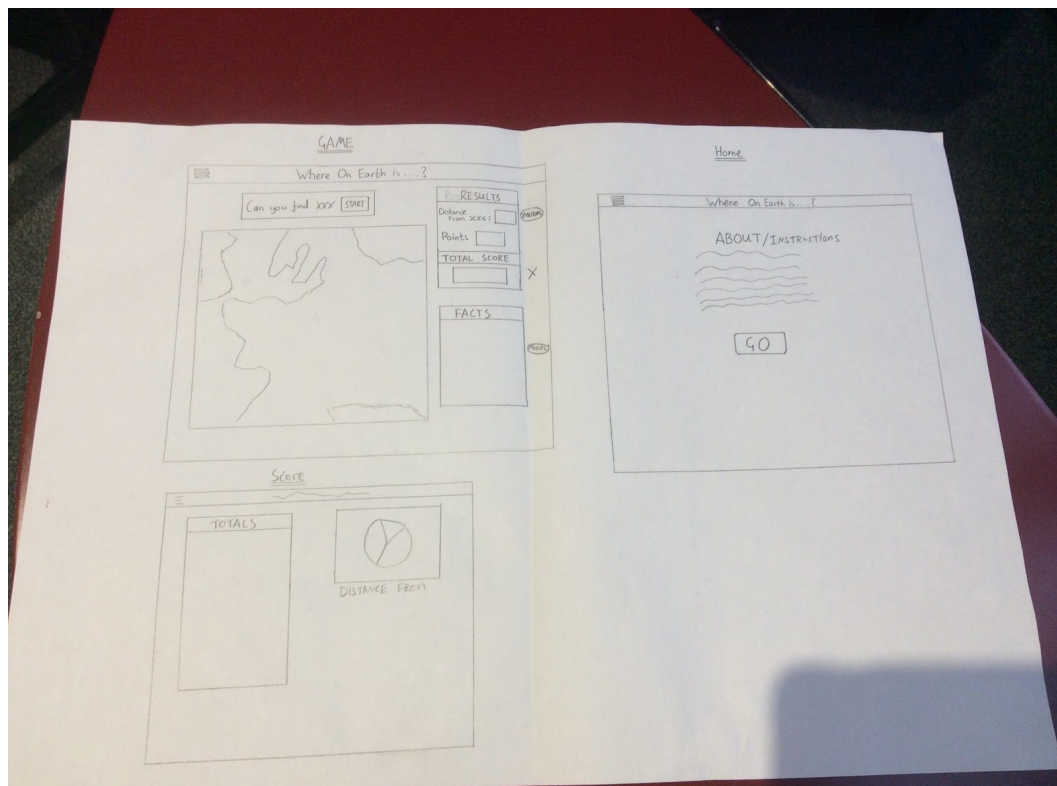
P 3 - Group Project Planning

P 4 - Acceptance Criteria

P 7 - System Interaction Diagrams

P 8 - Object Diagrams

P 9 - Algorithms



```

export default (grid, token) => {
  const horizontal = [0, 3, 6].map(i => [i, i + 1, i + 2]);
  const vertical = [0, 1, 2].map(i => [i, i + 3, i + 6]);
  const diagonals = [[0, 4, 8], [2, 4, 6]];

  const allWins = [...horizontal, ...vertical, ...diagonals];
  console.log(allWins);

  const win = allWins.some(
    winningLine =>
      grid[winningLine[0]] === token &&
      grid[winningLine[1]] === token &&
      grid[winningLine[2]] === token
  );

  return win;
};

```

```

const reverseString = str =>
  str.split('').reduce((string, char) => char + string, '')

console.log(reverseString('hello'))

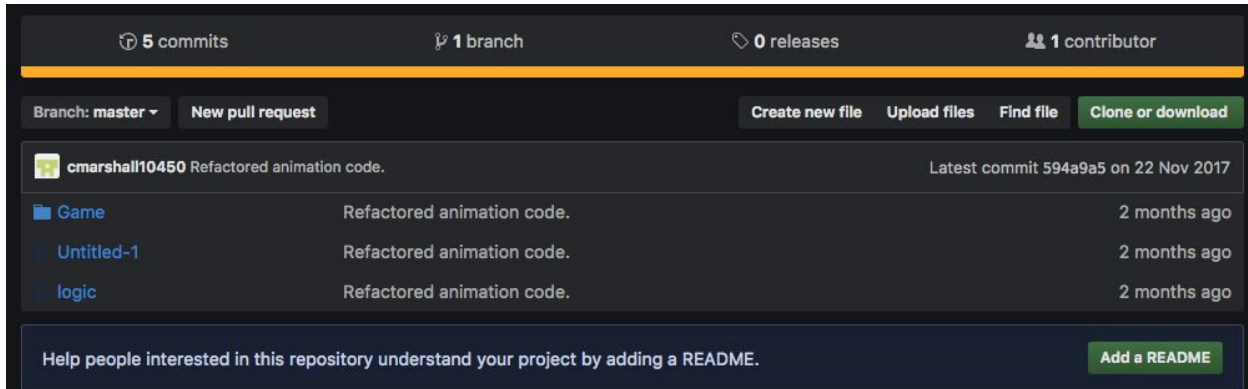
```

These algorithms have been chosen as they both display functional programming techniques. The algorithms perform calculations that could otherwise be performed with other programming paradigms but would not be as efficient or as clean.

P 10 - Pseudocode

```
def reverse_string(str)
  # split sting into an array of characters
  # create an empty array
  # for each character
  #   add it to the start of the array
  #
  # join array into string
  # return new string
end
```

P 11 - Solo Project



The screenshot shows a GitHub repository page for the user 'cmarshall10450'. At the top, it displays repository statistics: 5 commits, 1 branch, 0 releases, and 1 contributor. Below this is a navigation bar with 'Branch: master' and a 'New pull request' button. To the right are buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The main content area shows the repository name 'cmarshall10450' with the description 'Refactored animation code.' and the latest commit '594a9a5 on 22 Nov 2017'. A file list shows three items: 'Game', 'Untitled-1', and 'logic', all with the description 'Refactored animation code.' and a timestamp of '2 months ago'. At the bottom, there is a prompt to 'Add a README' to help people understand the project.

5 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

cmarshall10450 Refactored animation code. Latest commit 594a9a5 on 22 Nov 2017

Game	Refactored animation code.	2 months ago
Untitled-1	Refactored animation code.	2 months ago
logic	Refactored animation code.	2 months ago

Help people interested in this repository understand your project by adding a README. Add a README

<https://github.com/cmarshall10450/Android-Project-Blackjack-Game>



P 12 - Planning & Stages of Development

P 13 - User Interaction Being Processed

P 14 - Interaction with Data Persistence

P 15 - User Feedback

P 16 - API Use

P 17 - Bug Tracking Report

P 18 - Program Testing