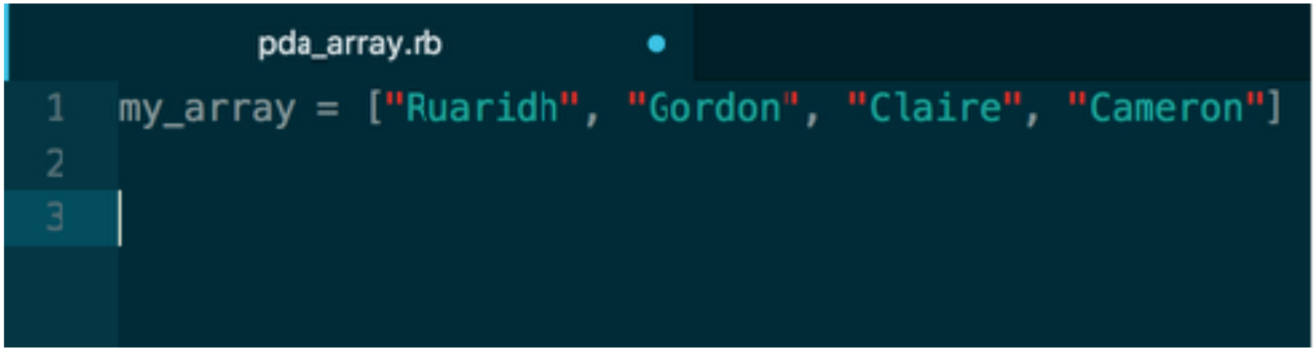
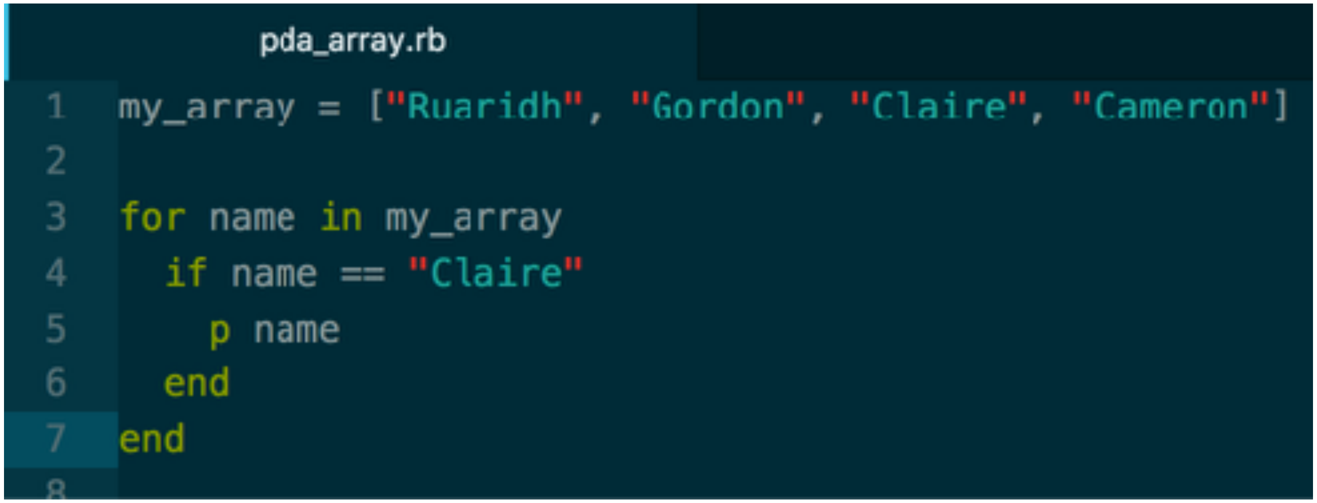
PDA: Software Development
Level 8
Student Evidence Checklist

Full name	**Ruaridh Dunbar**
Cohort	**G5**

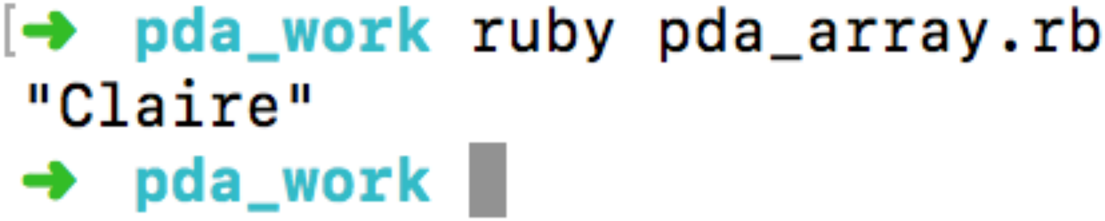
The evidence required can be taken from your assignments, homework that you have completed on your own or by creating a specific example for the PDA.

	Unit	**Ref.**	**Evidence**	**Done**
	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	
			 pda_array.rb `my_array = ["Ruaridh", "Gordon", "Claire", "Cameron"]`	

<table>
<tr><td></td><td></td><td></td><td>

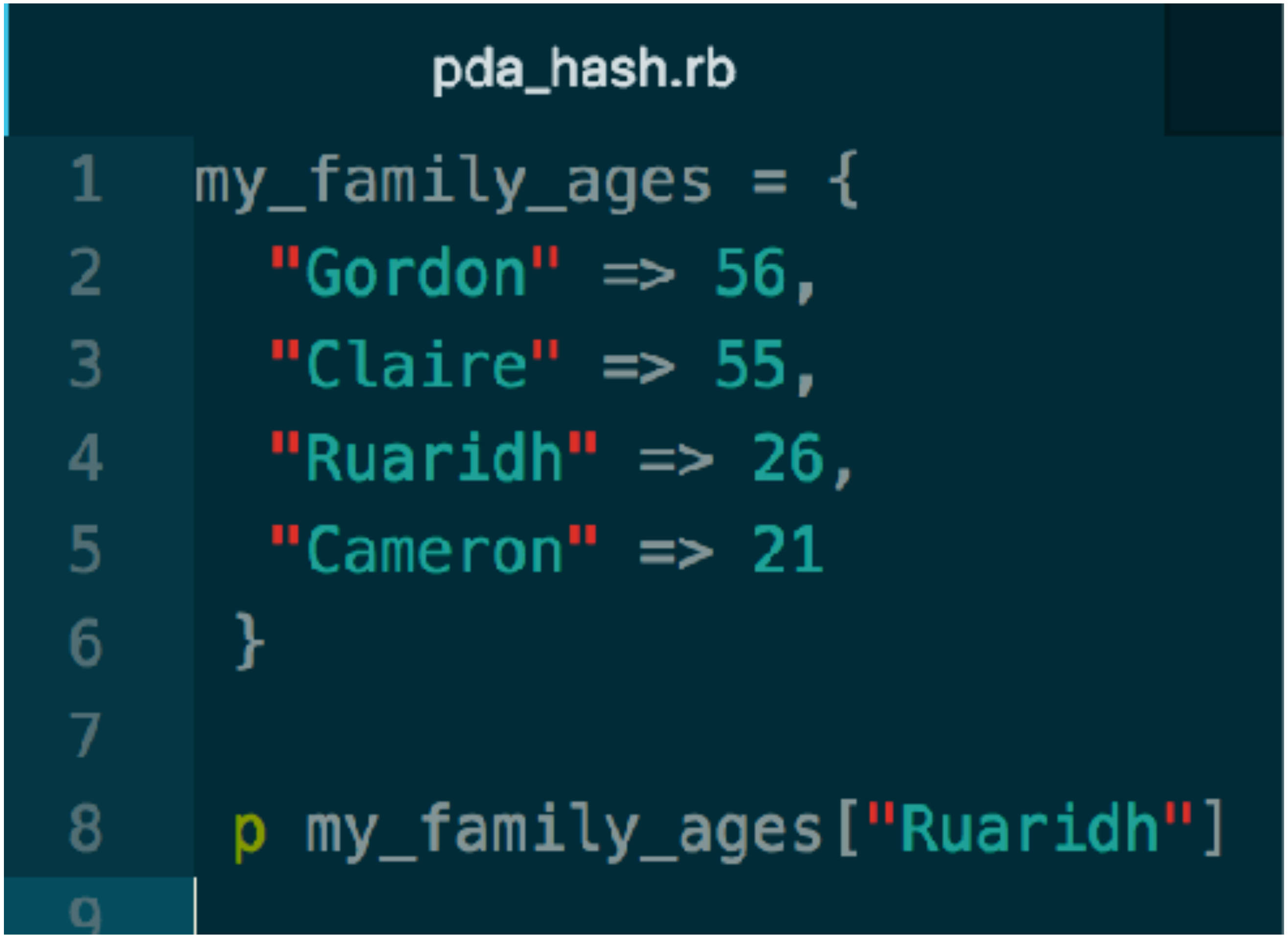


```
my_array = ["Ruaridh", "Gordon", "Claire", "Cameron"]

for name in my_array
  if name == "Claire"
    p name
  end
end
```



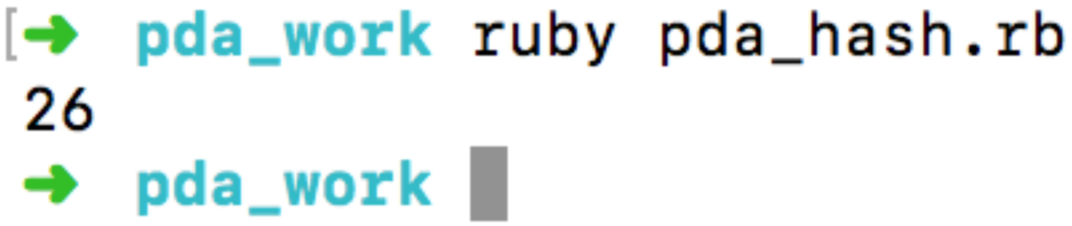
```
[➜  pda_work ruby pda_array.rb
 "Claire"
 ➜  pda_work 
```

</td><td></td></tr>
<tr><td><b>Week 2</b></td><td>I &amp; T</td><td>I.T 6</td><td>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</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>



pda_hash.rb

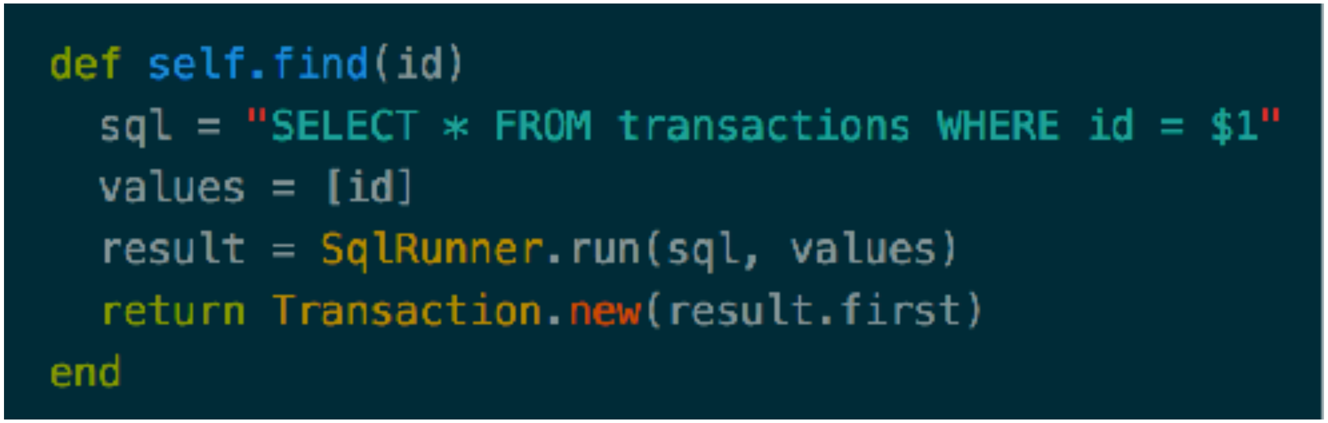
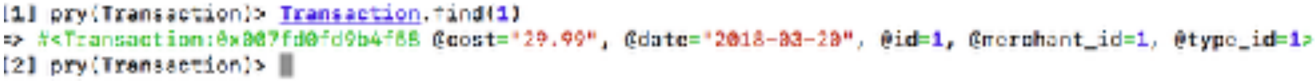
```ruby
my_family_ages = {
  "Gordon" => 56,
  "Claire" => 55,
  "Ruaridh" => 26,
  "Cameron" => 21
 }

 p my_family_ages["Ruaridh"]

```



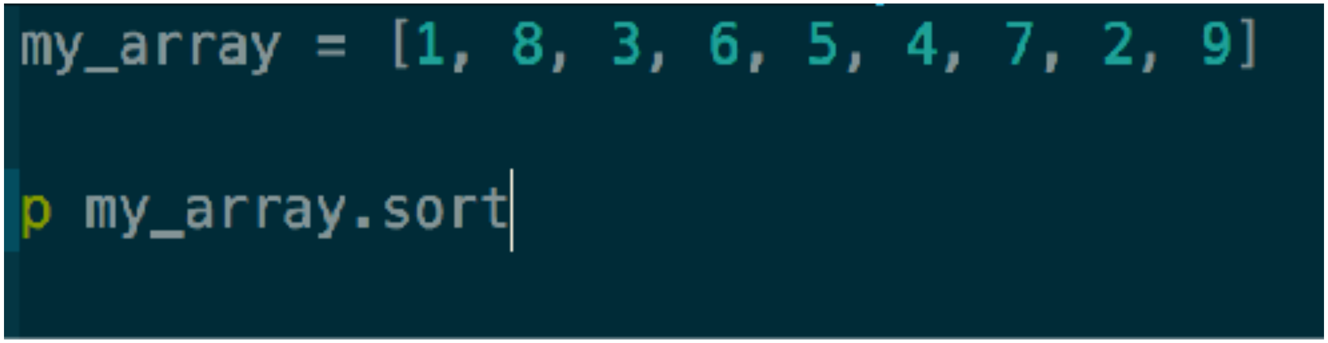
```
[➜  pda_work ruby pda_hash.rb
26
➜  pda_work ▊
```

	I & T		Static and Dynamic testing task A	

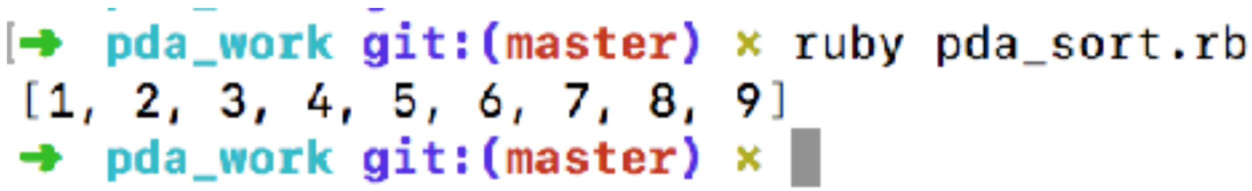
	Unit	Ref.	Evidence	Done
Week 3	I & T	I.T 3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running	

			
			`def self.find(id)`
			`  sql = "SELECT * FROM transactions WHERE id = $1"`
			`  values = [id]`
			`  result = SqlRunner.run(sql, values)`
			`  return Transaction.new(result.first)`
			`end`
			
			`[1] pry(Transaction)> Transaction.find(1)`
			`=> #<Transaction:0x007fd0fd9b4f88 @cost="29.99", @date="2018-03-20", @id=1, @merchant_id=1, @type_id=1>`
			`[2] pry(Transaction)>`	

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

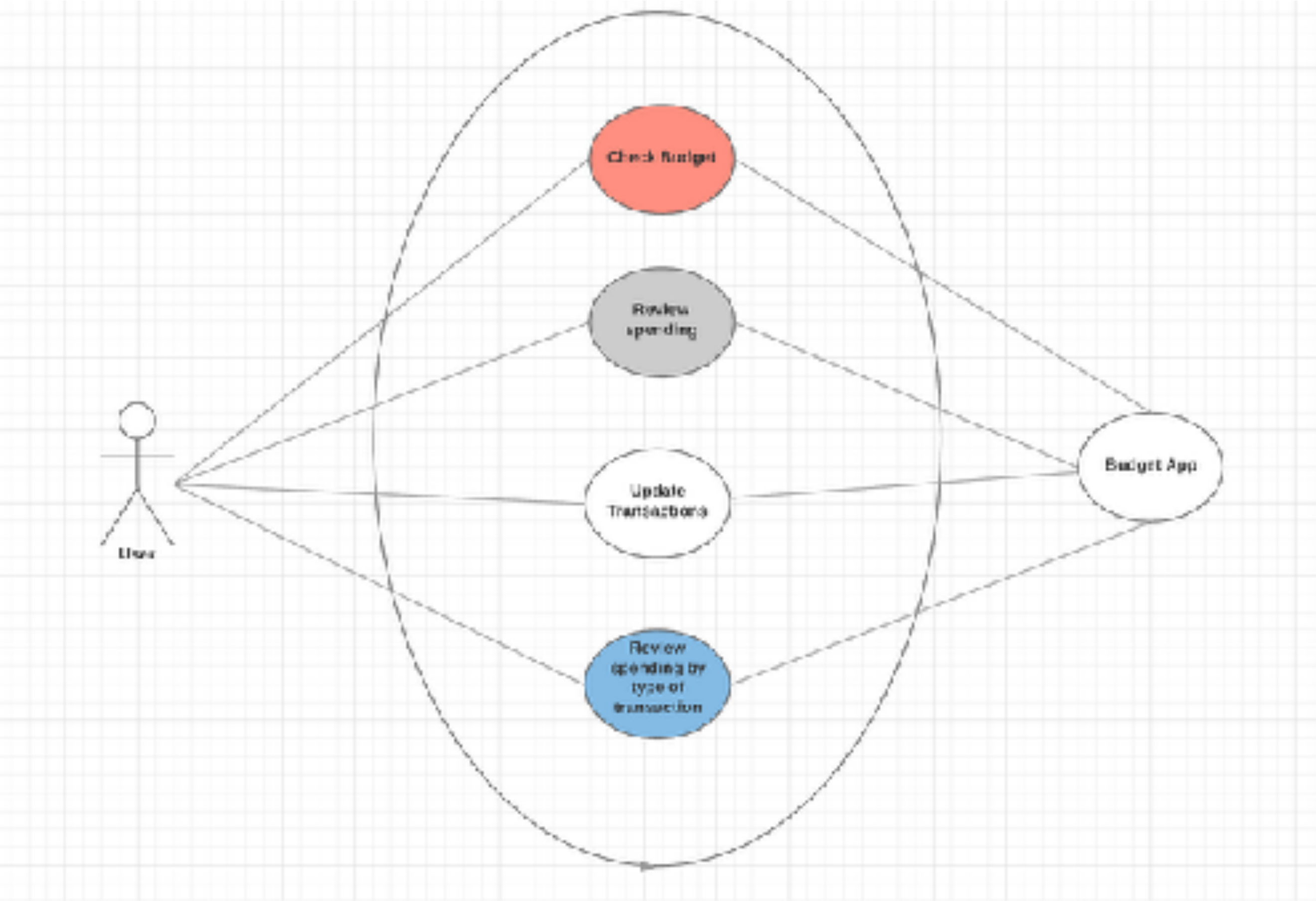
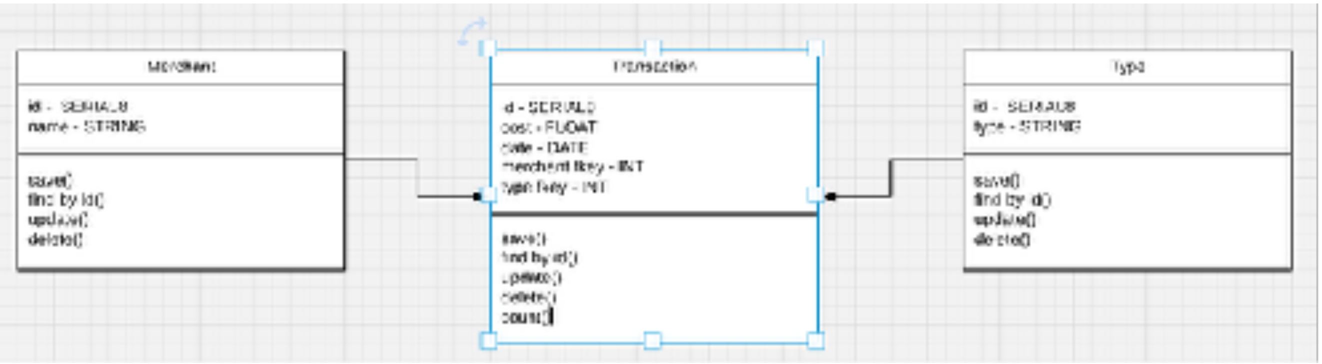


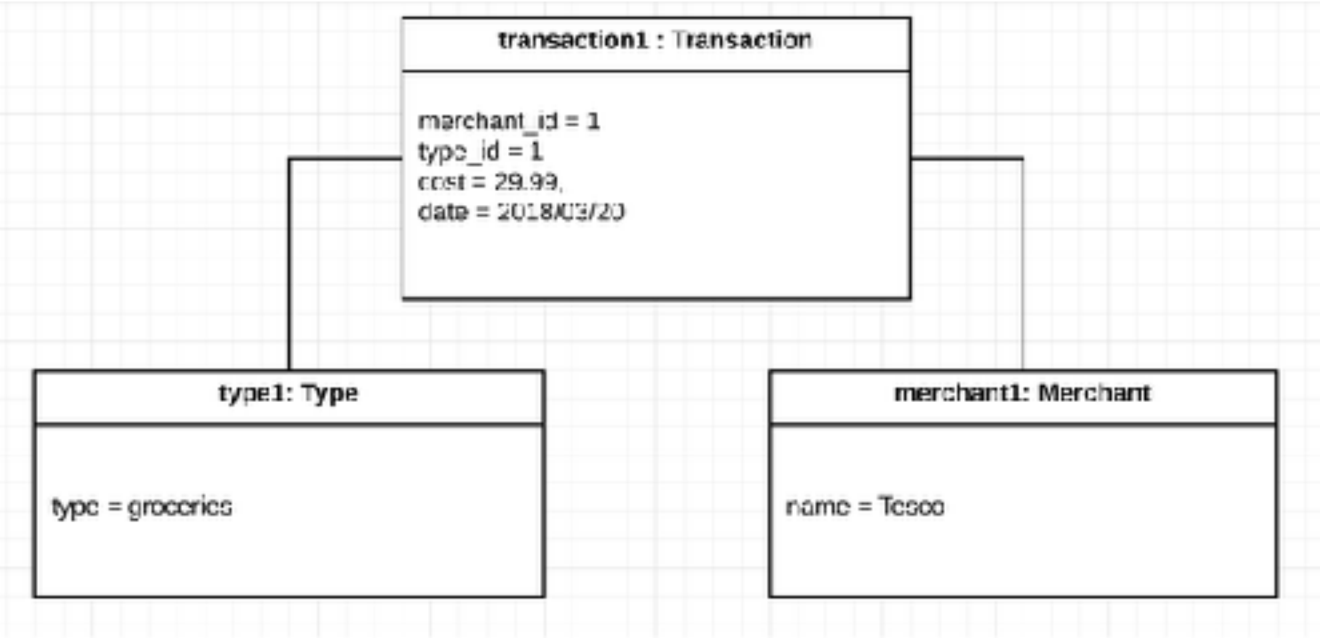
```
my_array = [1, 8, 3, 6, 5, 4, 7, 2, 9]

p my_array.sort
```

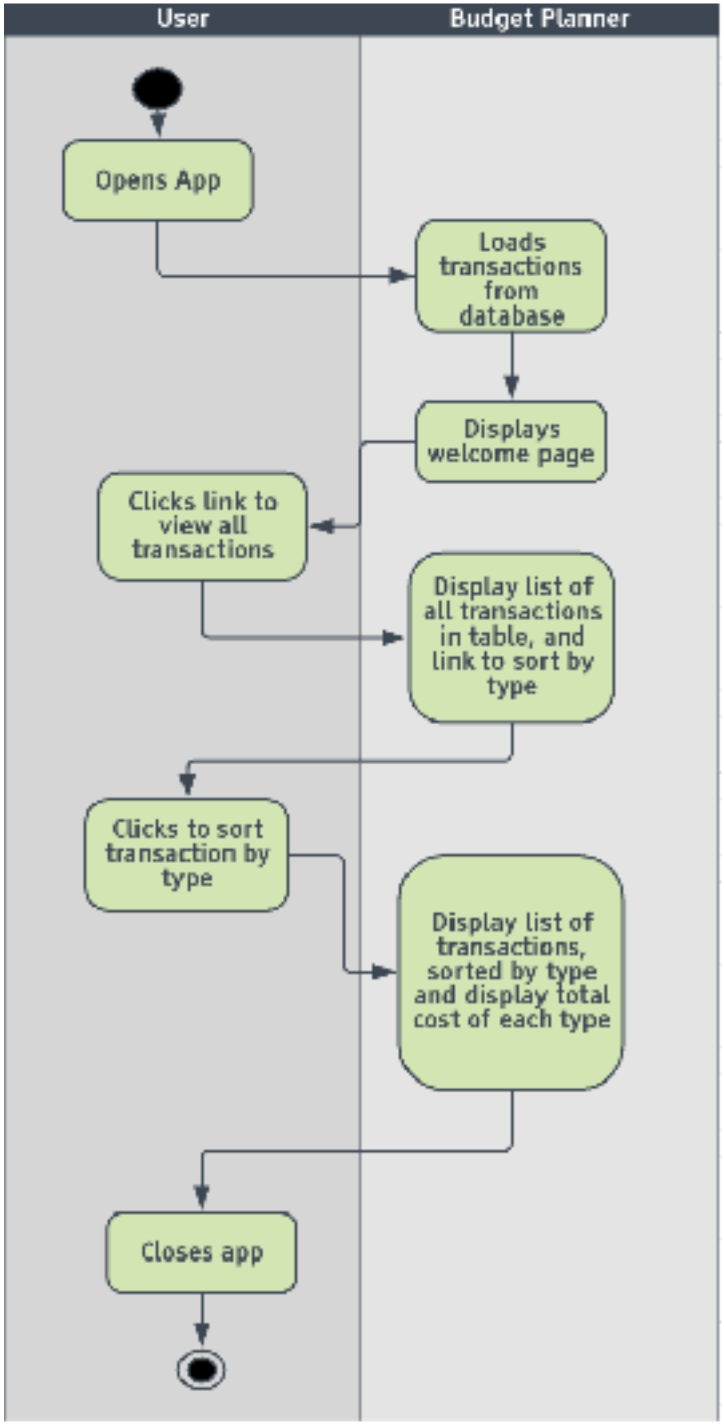


```
[➜  pda_work git:(master) ✗ ruby pda_sort.rb
[1, 2, 3, 4, 5, 6, 7, 8, 9]
 ➜  pda_work git:(master) ✗ 
```

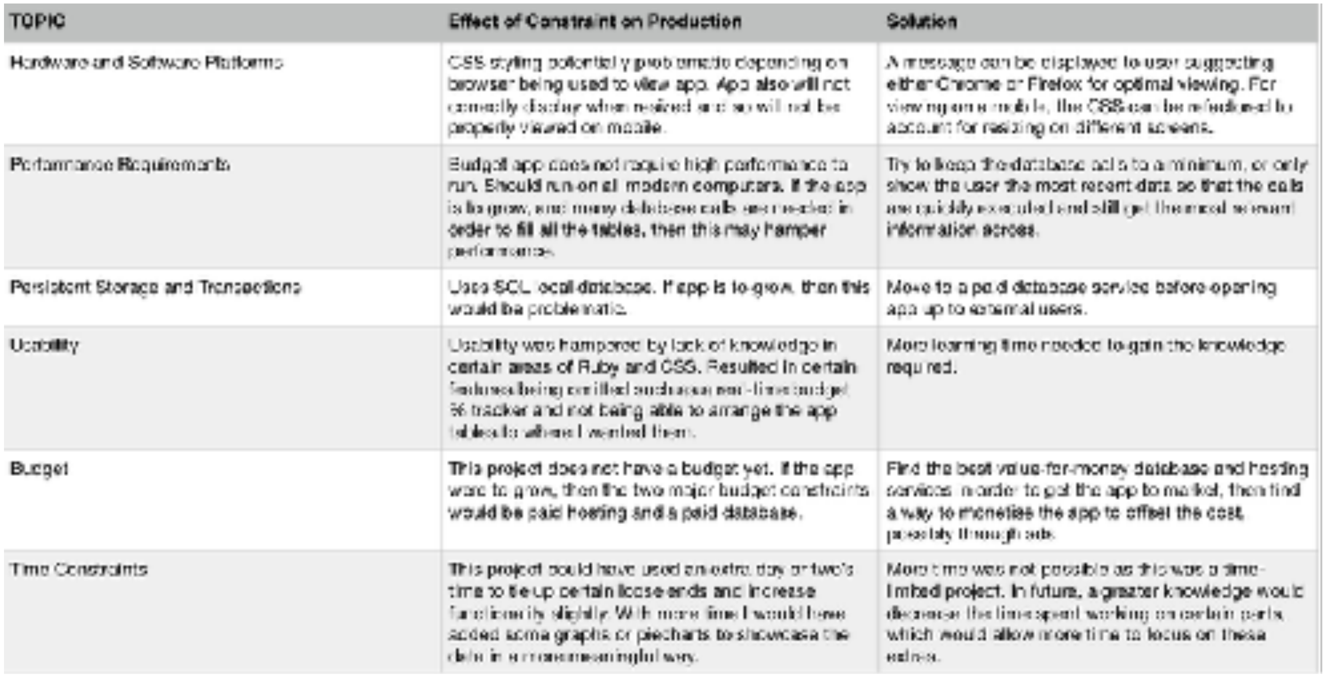
	Unit	**Ref.**	**Evidence**	**Done**
	A & D	A.D 1	A Use Case Diagram	

		 Check Budget Review spending User Update Transactions Budget App Review spending by type of transaction	
A & D	A.D 2	A Class diagram.	
		 Merchant id - SERIAL8 name - STRING save() find by id() update() delete() Transaction id - SERIAL8 cost - FLOAT date - DATE merchant fkey - INT type fkey - INT save() find by id() update() delete() count() Type id - SERIAL8 type - STRING save() find by id() update() delete()	

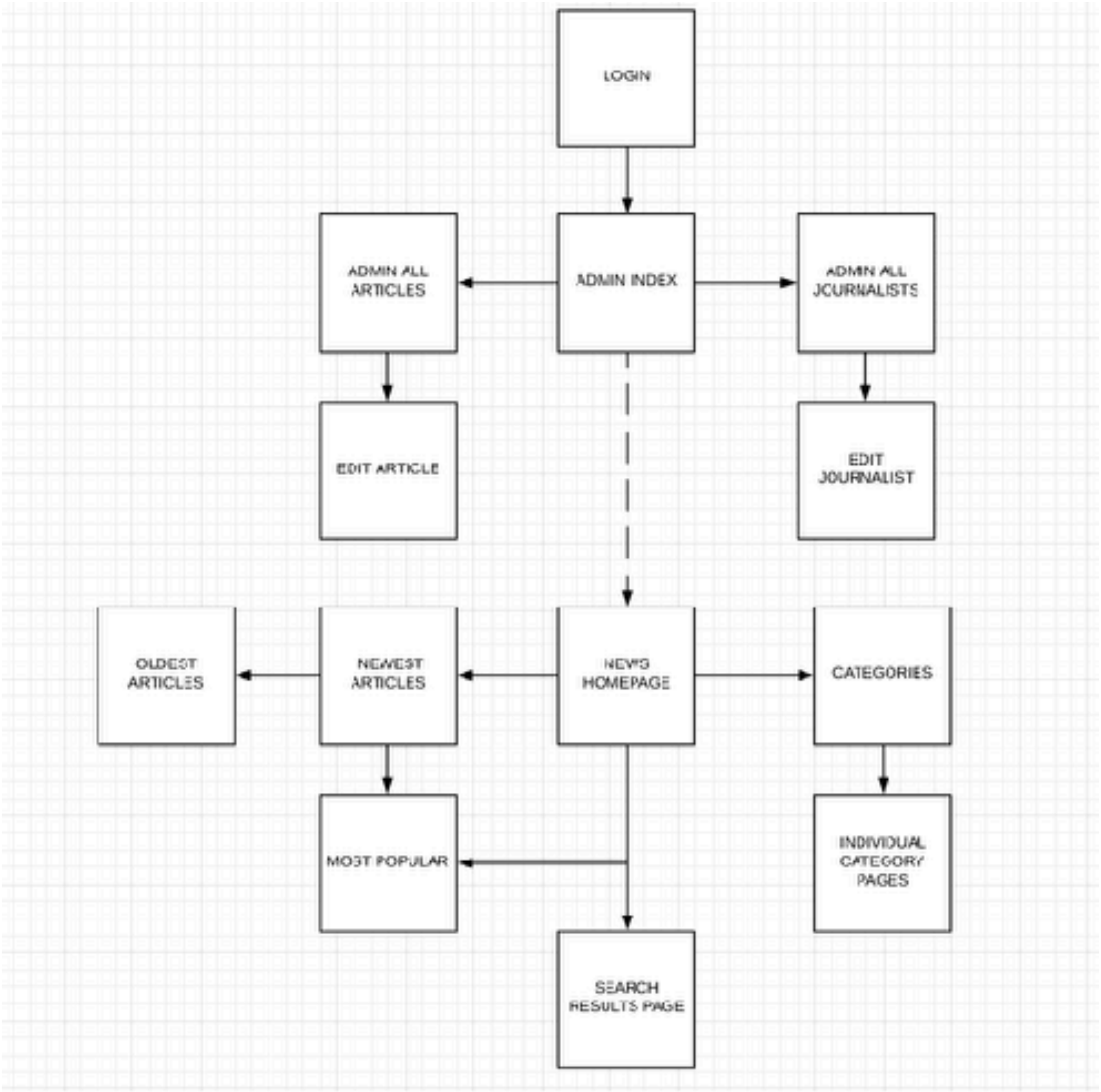
A & D	A.D 3	An Object diagram.	
			
A & D	A.D 4	An Activity Diagram	

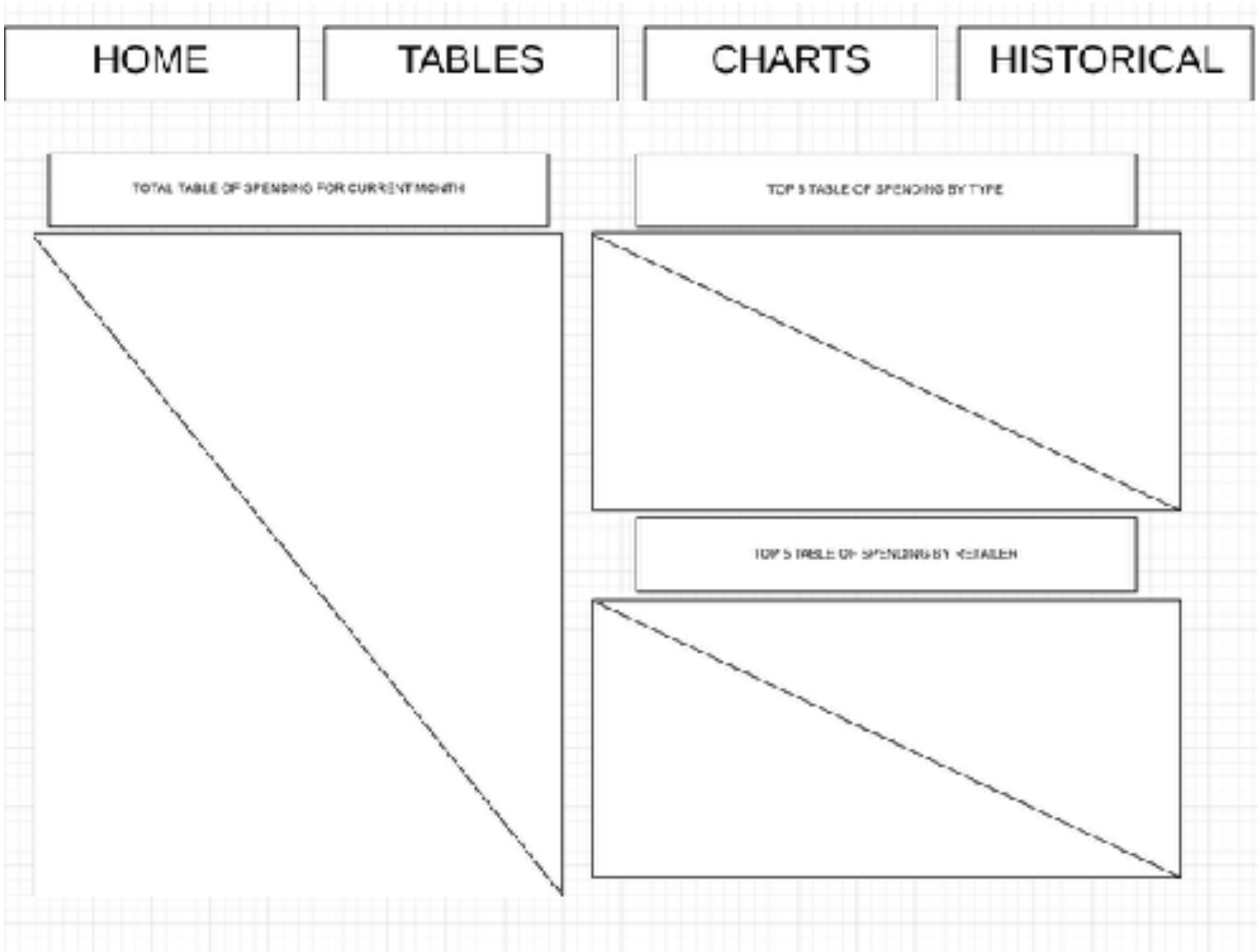
			

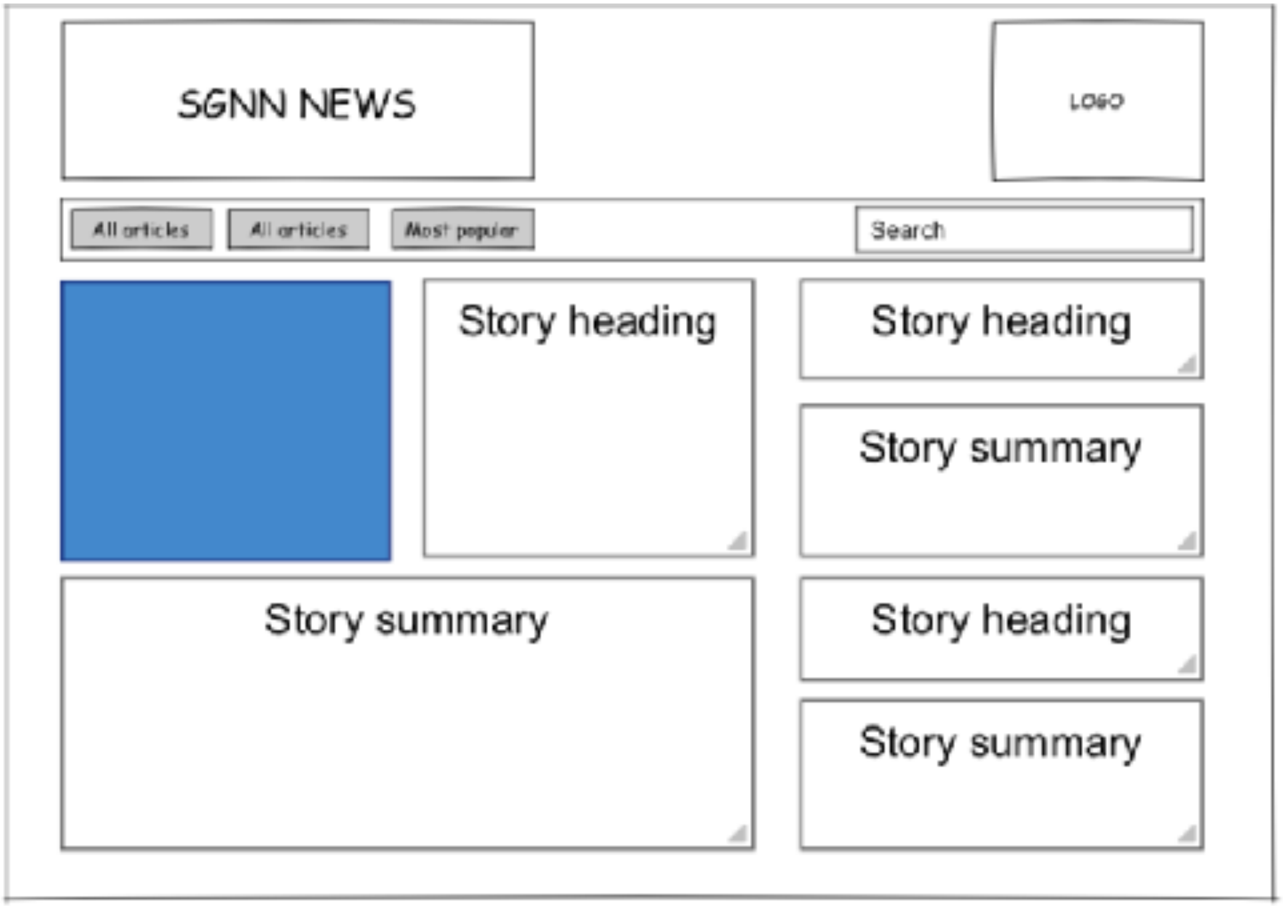
User
Budget Planner
Opens App
Loads transactions from database
Displays welcome page
Clicks link to view all transactions
Display list of all transactions in table, and link to sort by type
Clicks to sort transaction by type
Display list of transactions, sorted by type and display total cost of each type
Closes app

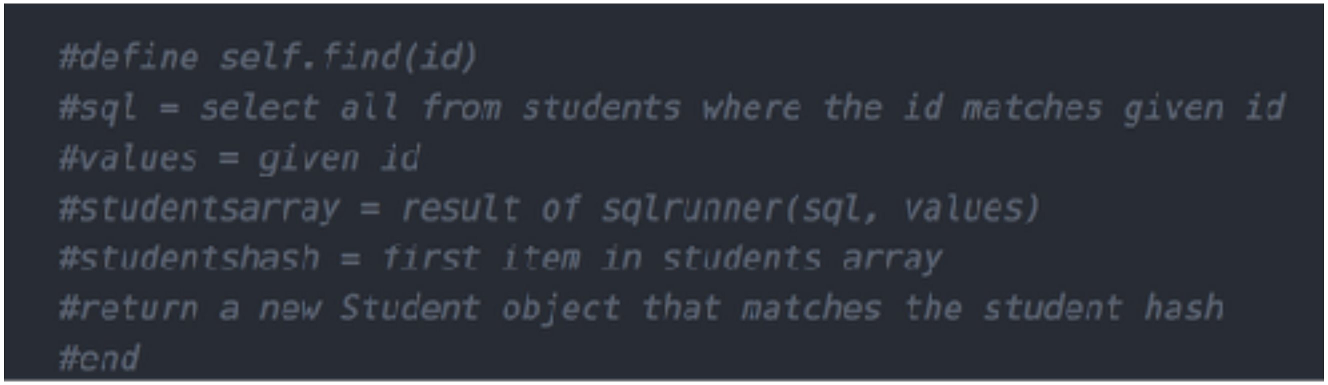
A & D	A.D 6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	
			
P	P 5	Create a user sitemap.	

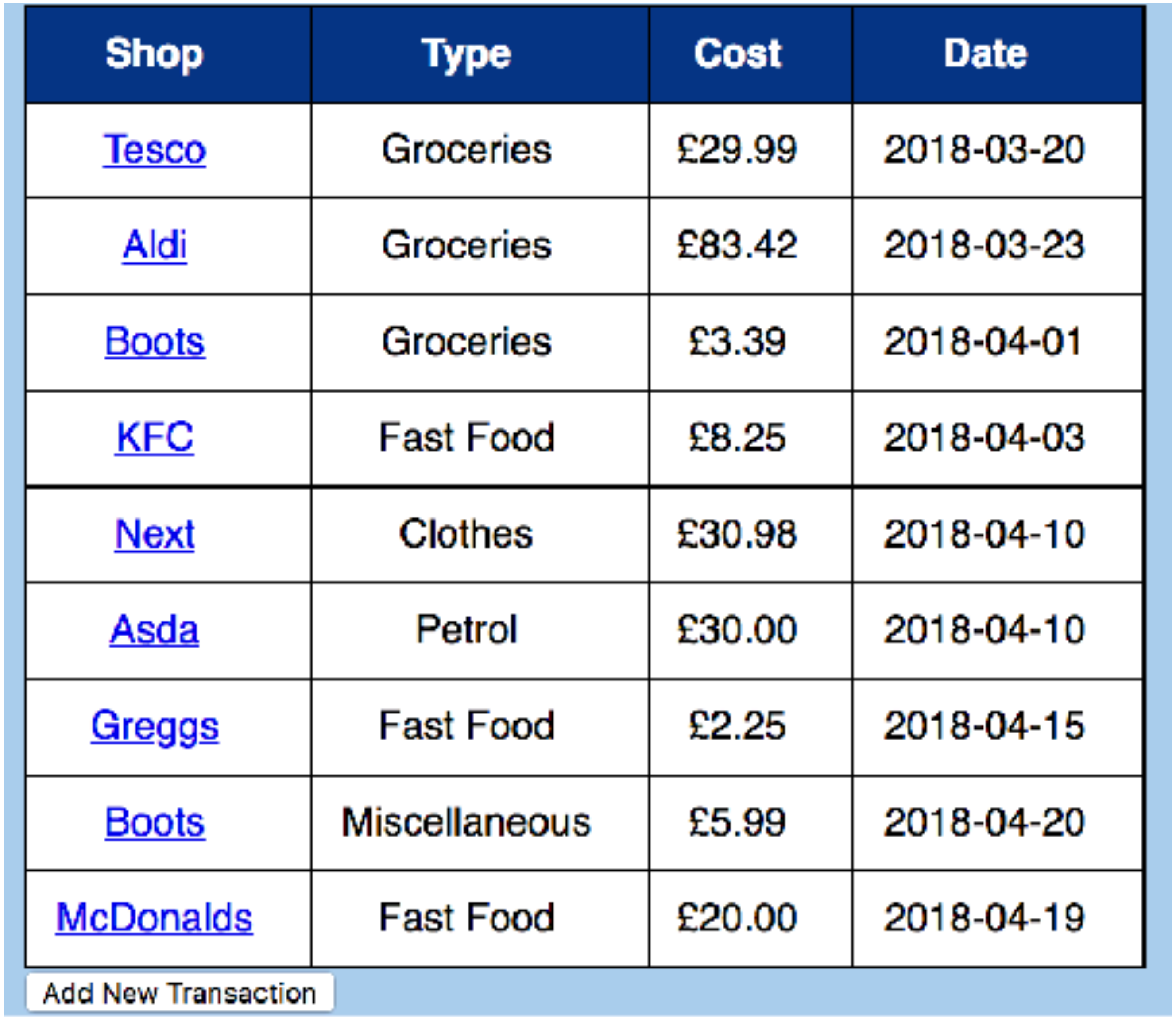
TOPIC	Effect of Constraint on Production	Solution
Hardware and Software Platforms	CSS styling potentially problematic depending on browser being used to view app. App also will not correctly display when resized and so will not be properly viewed on mobile.	A message can be displayed to user suggesting either Chrome or Firefox for optimal viewing. For viewing on a mobile, the CSS can be refactored to account for resizing on different screens.
Performance Requirements	Budget app does not require high performance to run. Should run on all modern computers. If the app is to grow, and many database calls are needed in order to fill all the tables, then this may hamper performance.	Try to keep the database calls to a minimum, or only show the user the most recent data so that the calls are quickly executed and still get the most relevant information across.
Persistent Storage and Transactions	Uses SQL local database. If app is to grow, then this would be problematic.	Move to a paid database service before opening app up to external users.
Usability	Usability was hampered by lack of knowledge in certain areas of Ruby and CSS. Resulted in certain features being omitted such as a real-time budget % tracker and not being able to arrange the app tables to where I wanted them.	More learning time needed to gain the knowledge required.
Budget	This project does not have a budget yet. If the app were to grow, then the two major budget constraints would be paid hosting and a paid database.	Find the best value-for-money database and hosting services in order to get the app to market, then find a way to monetise the app to offset the cost, possibly through ads.
Time Constraints	This project could have used an extra day or two's time to tie up certain loose ends and increase functionality slightly. With more time I would have added some graphs or piecharts to showcase the data in a more meaningful way.	More time was not possible as this was a time-limited project. In future, a greater knowledge would decrease the time spent working on certain parts, which would allow more time to focus on these extras.

		 LOGIN ADMIN ALL ARTICLES ADMIN INDEX ADMIN ALL JOURNALISTS EDIT ARTICLE EDIT JOURNALIST OLDEST ARTICLES NEWEST ARTICLES NEWS HOMEPAGE CATEGORIES MOST POPULAR INDIVIDUAL CATEGORY PAGES SEARCH RESULTS PAGE	
P	P 6	Produce two wireframe designs.	

Week 5			 HOME TABLES CHARTS HISTORICAL TOTAL TABLE OF SPENDING FOR CURRENT MONTH TOP 5 TABLE OF SPENDING BY TYPE TOP 5 TABLE OF SPENDING BY RETAILER An early representation of how the ‘tables’ page would look on my budget tracker app	

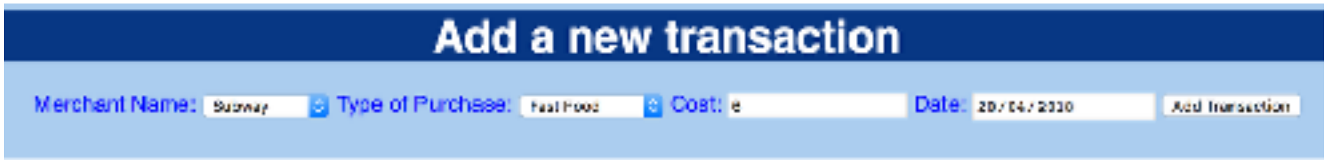
		 SGNN NEWS LOGO All articles All articles Most popular Search Story heading Story heading Story summary Story summary Story heading Story summary A wireframe of the main story page on the Java pair project news website app	
P	P 10	Take a screenshot of an example of pseudocode for a function.	

		 #define self.find(id) #sql = select all from students where the id matches given id #values = given id #studentsarray = result of sqlrunner(sql, values) #studentshash = first item in students array #return a new Student object that matches the student hash #end	
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	



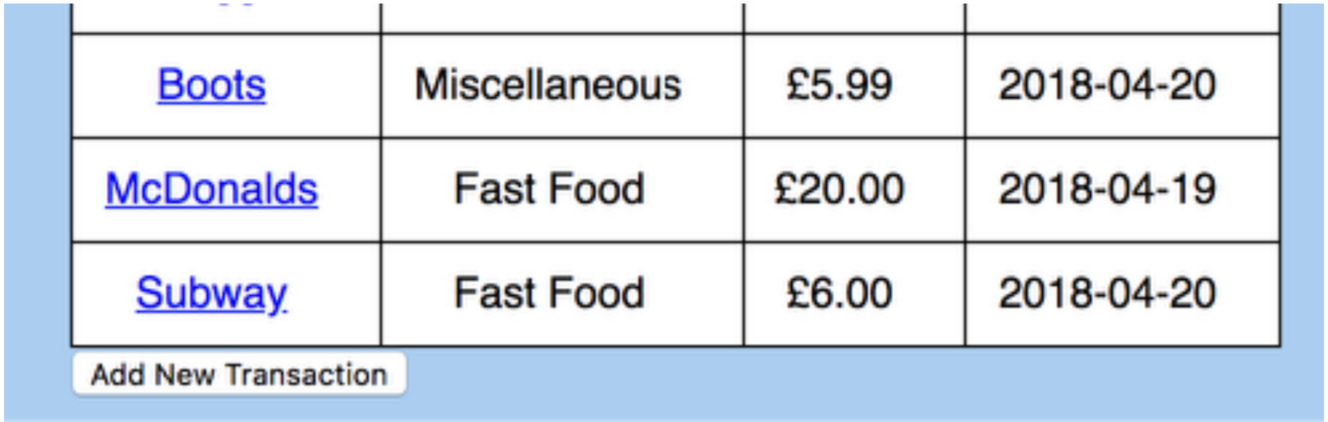
Shop	Type	Cost	Date
Tesco	Groceries	£29.99	2018-03-20
Aldi	Groceries	£83.42	2018-03-23
Boots	Groceries	£3.39	2018-04-01
KFC	Fast Food	£8.25	2018-04-03
Next	Clothes	£30.98	2018-04-10
Asda	Petrol	£30.00	2018-04-10
Greggs	Fast Food	£2.25	2018-04-15
Boots	Miscellaneous	£5.99	2018-04-20
McDonalds	Fast Food	£20.00	2018-04-19

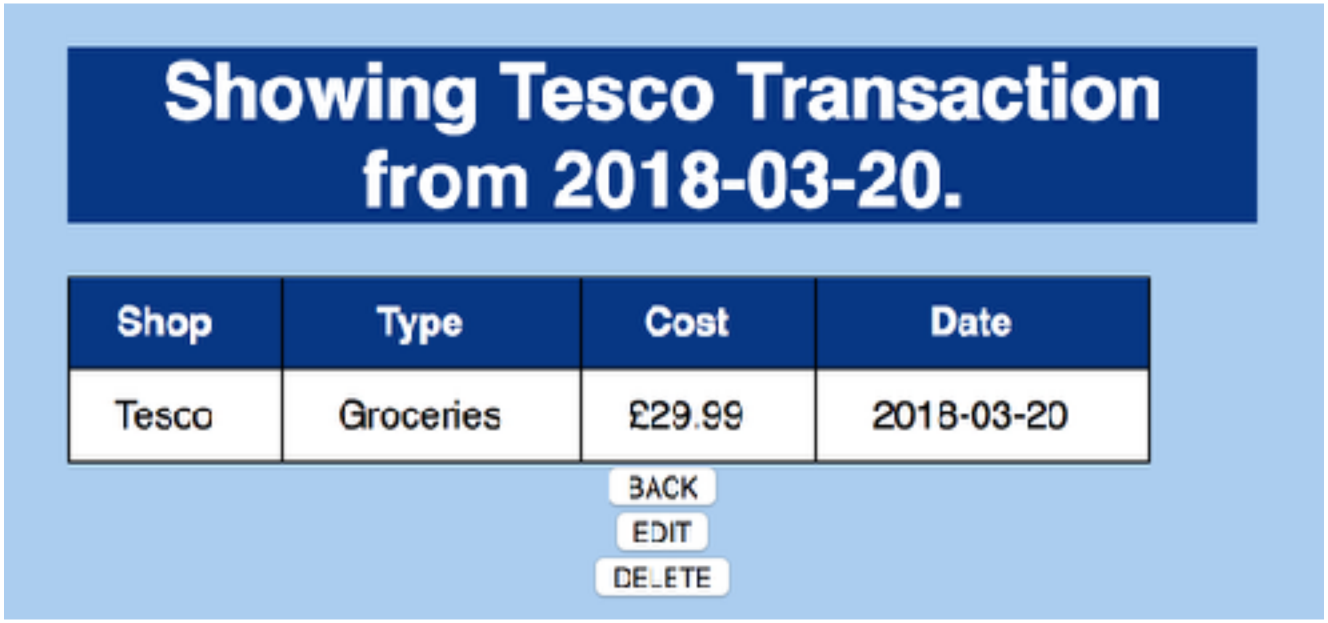
Step 1 - Shows prepopulated table



Add a new transaction

Merchant Name: Subway Type of Purchase: Fast Food Cost: 6 Date: 20/04/2030

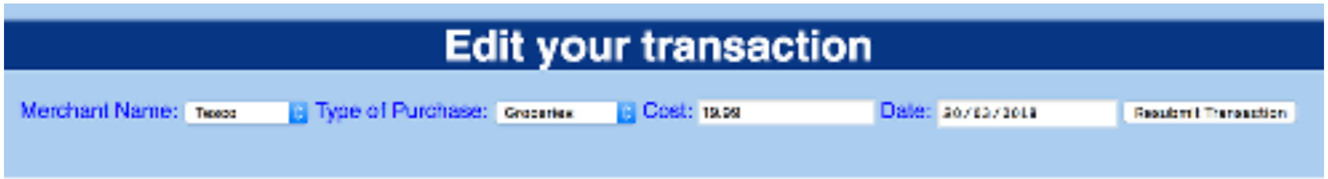
<table>
<tr><td></td><td></td><td>Step 2 - Shows the user creating a new transaction
<table><tr><td>Boots</td><td>Miscellaneous</td><td>£5.99</td><td>2018-04-20</td></tr><tr><td>McDonalds</td><td>Fast Food</td><td>£20.00</td><td>2018-04-19</td></tr><tr><td>Subway</td><td>Fast Food</td><td>£6.00</td><td>2018-04-20</td></tr></table>
Step 3 - Shows the updated table, with the user’s created transaction at the bottom.</td><td></td></tr>
<tr><td>P</td><td>P 14</td><td>Show an interaction with data persistence. Take a screenshot of:
* Data being inputted into your program
* Confirmation of the data being saved</td><td></td></tr>
</table>



Showing Tesco Transaction from 2018-03-20.

Shop	Type	Cost	Date
Tesco	Groceries	£29.99	2018-03-20

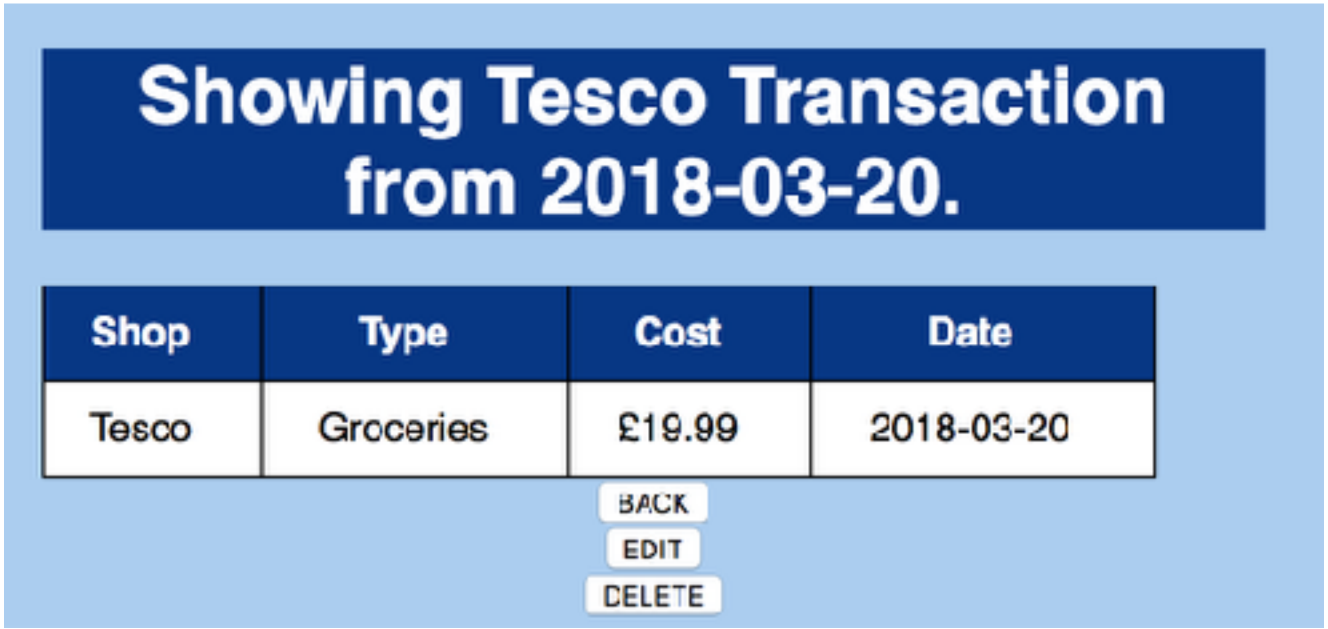
Step 1 - Shows single transaction



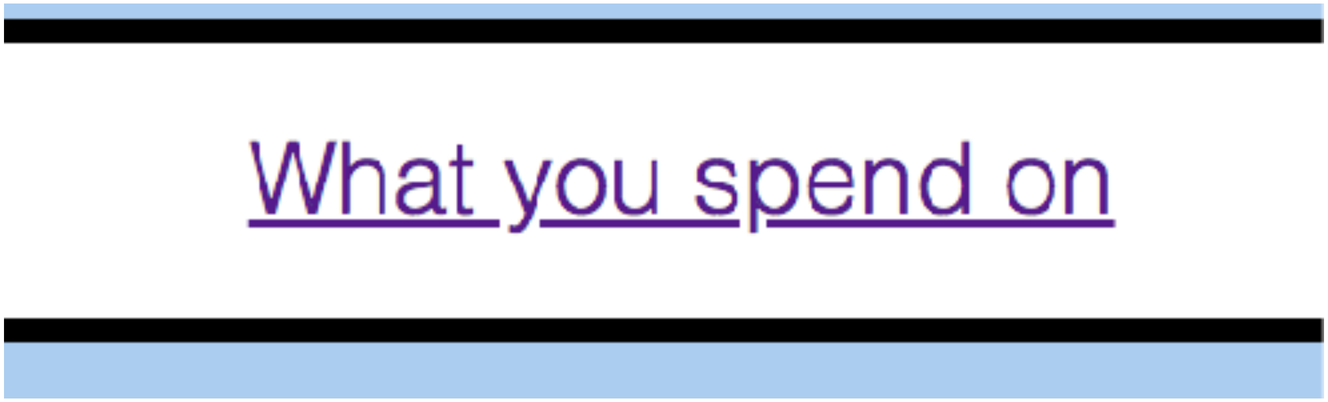
Edit your transaction

Merchant Name: Tesco Type of Purchase: Groceries Cost: 19.99 Date: 20/03/2018

Step 2 - Shows user editing the cost of the transaction

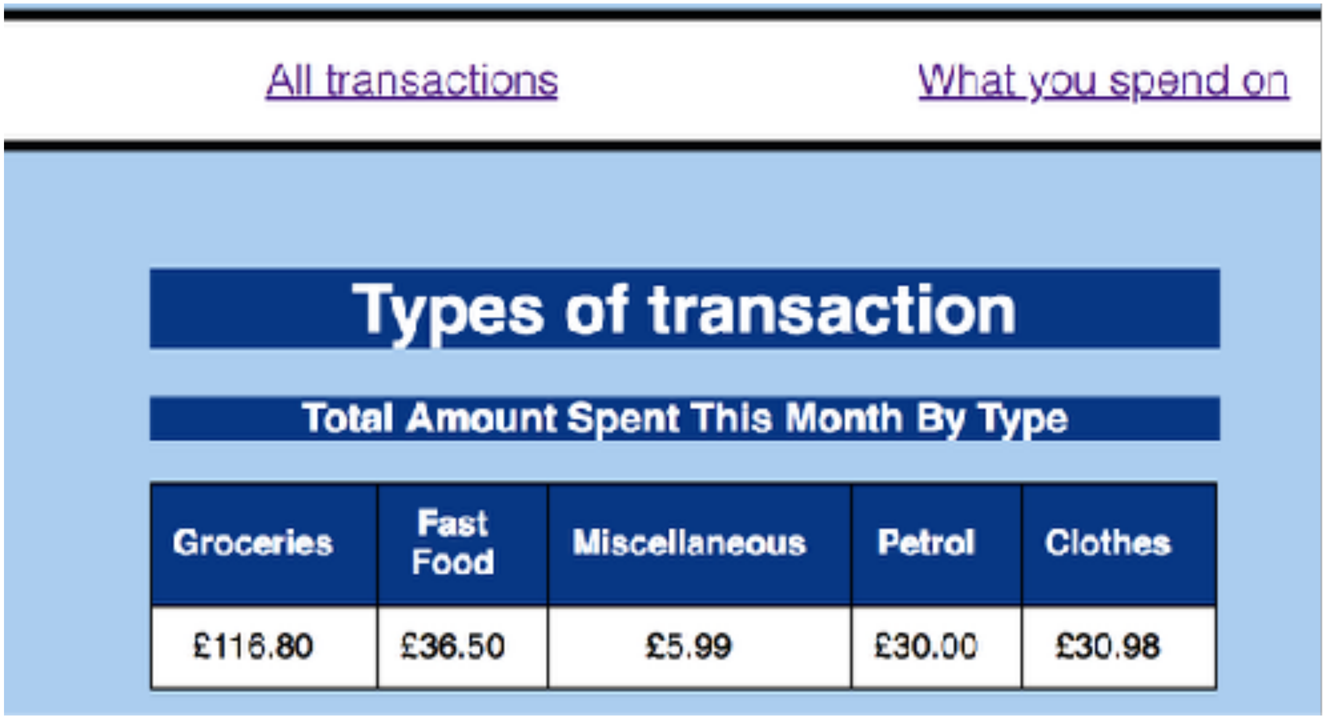
<table>
<tr><td></td><td></td><td>
<p><b>Showing Tesco Transaction from 2018-03-20.</b></p>
<table>
<tr><th>Shop</th><th>Type</th><th>Cost</th><th>Date</th></tr>
<tr><td>Tesco</td><td>Groceries</td><td>£19.99</td><td>2018-03-20</td></tr>
</table>

<p>Step 3 - Shows the single transaction with the updated cost of £19.99 down from £29.99</p></td><td></td></tr>
<tr><td>P</td><td>P 15</td><td>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</td><td></td></tr>
</table>



What you spend on

Step 1 - Hyperlink on main page header to be clicked by user



All transcriptions

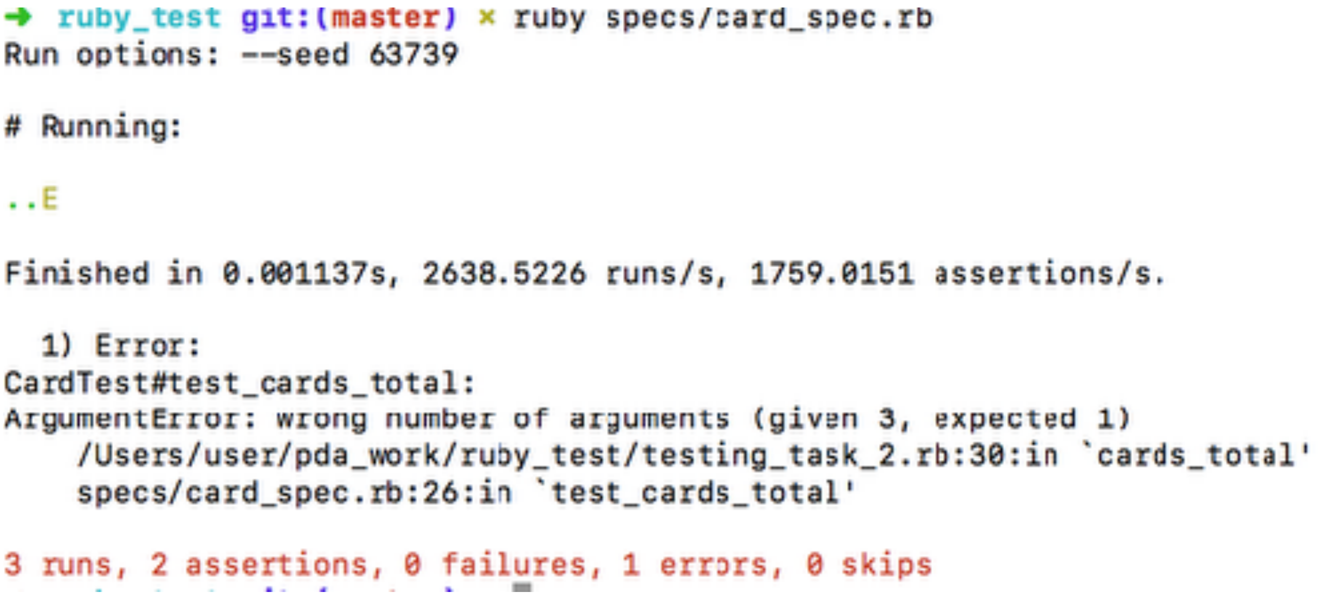
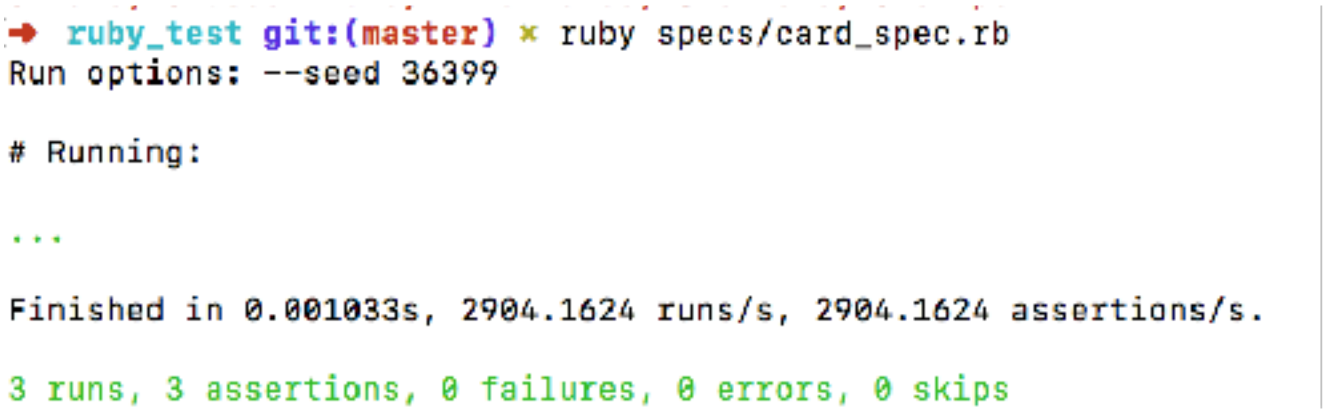
What you spend on

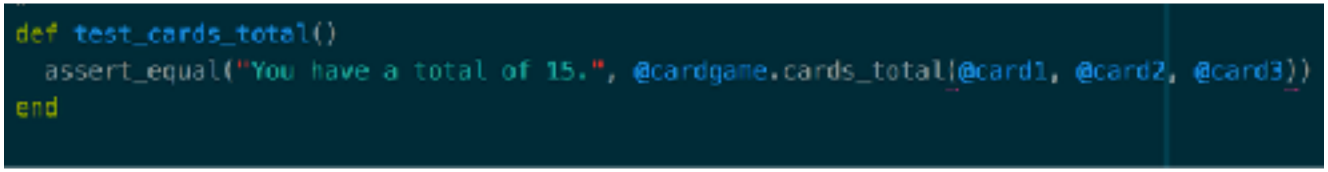
Types of transaction

Total Amount Spent This Month By Type

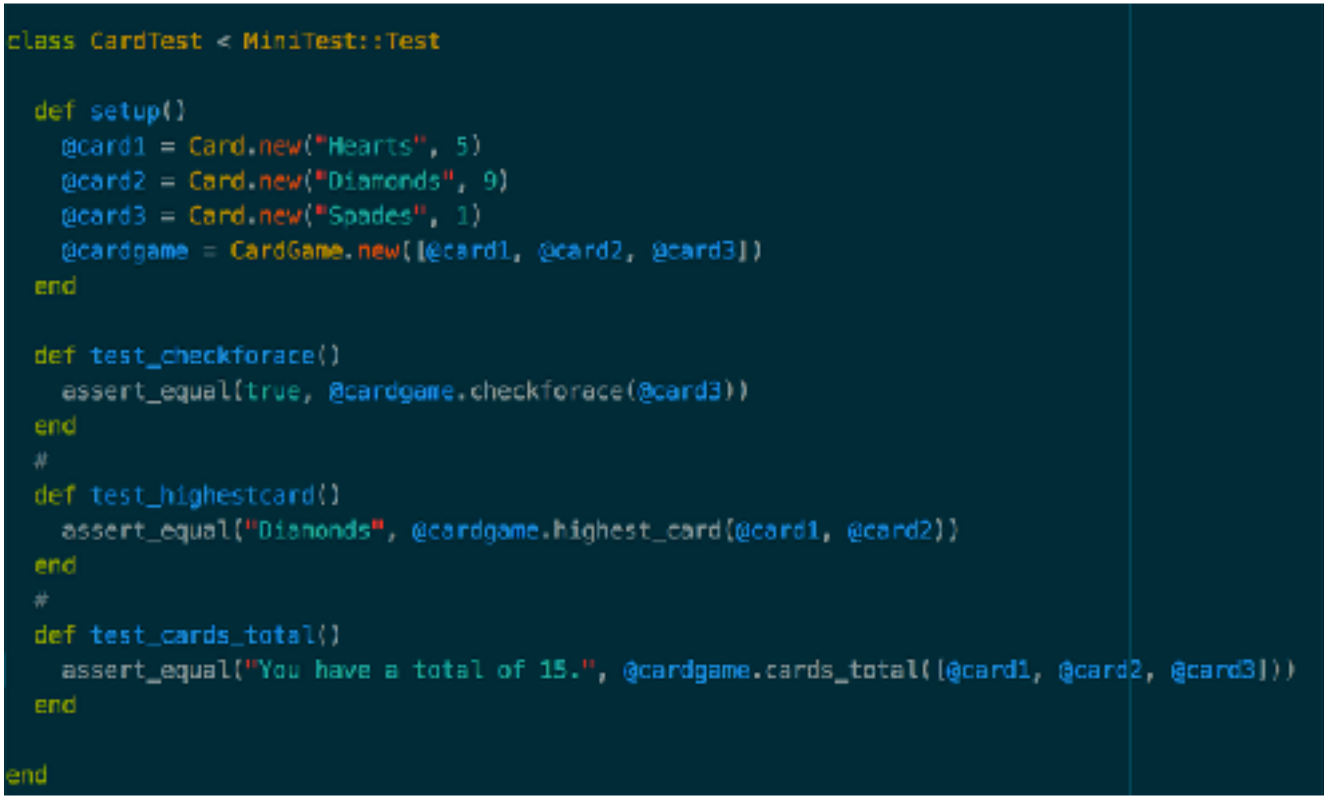
Groceries	Fast Food	Miscellaneous	Petrol	Clothes
£116.80	£36.50	£5.99	£30.00	£30.98

Step 2 - Resulting page displayed to user

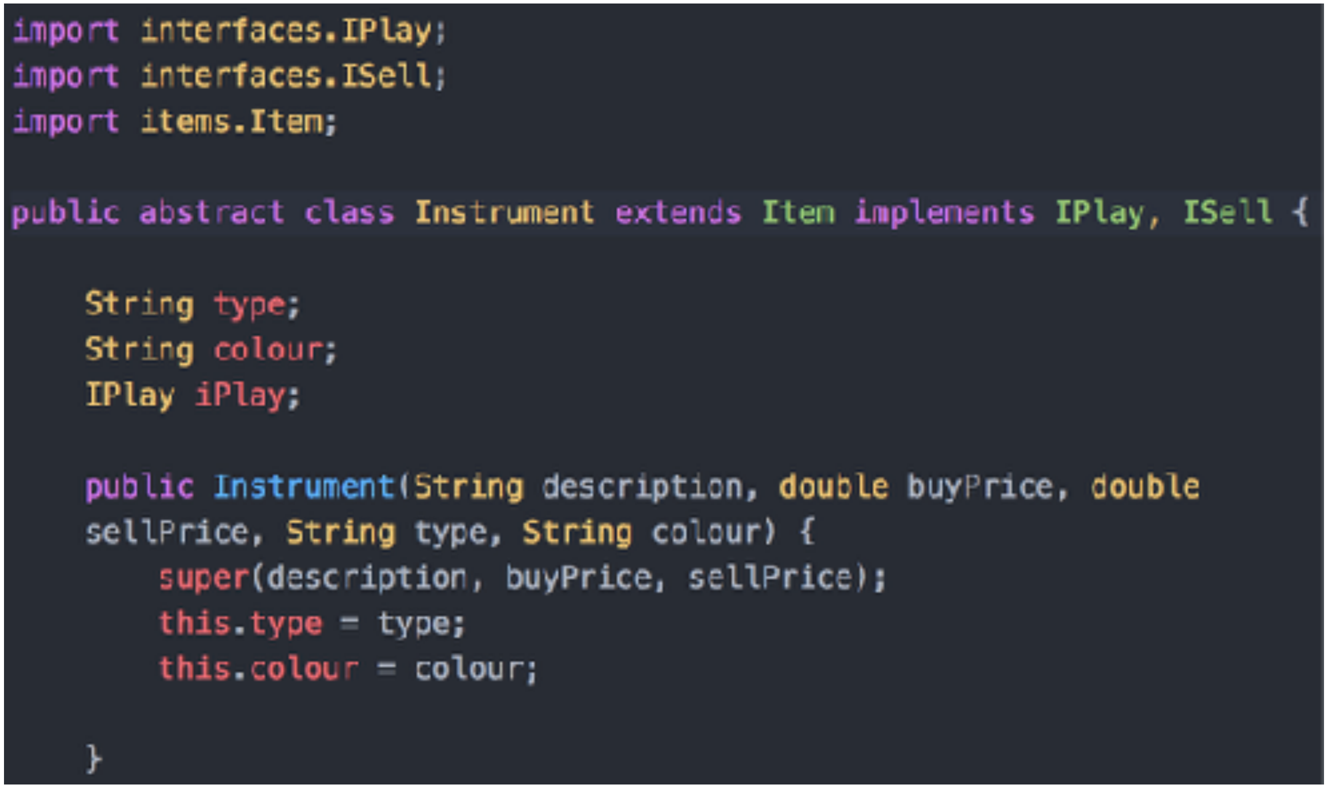
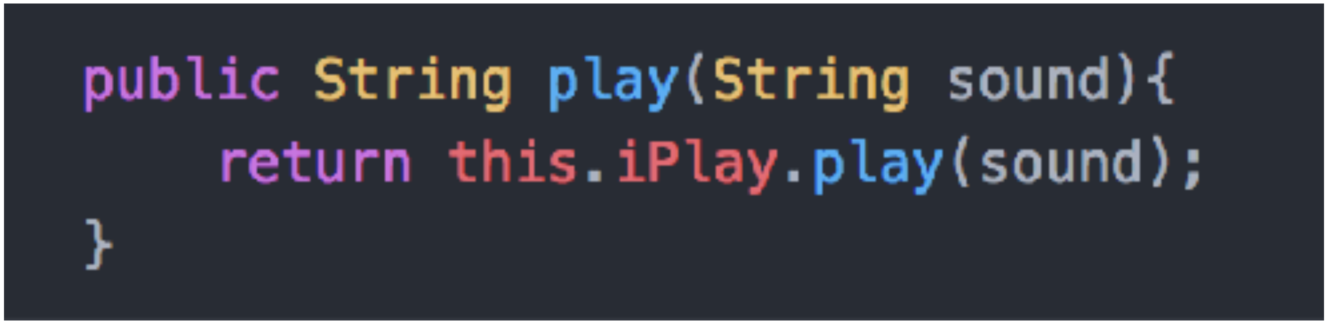
	P	P 18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing	
			 ➜ ruby_test git:(master) ✗ ruby specs/card_spec.rb Run options: --seed 63739 # Running: ..E Finished in 0.001137s, 2638.5226 runs/s, 1759.0151 assertions/s. 1) Error: CardTest#test_cards_total: ArgumentError: wrong number of arguments (given 3, expected 1) /Users/user/pda_work/ruby_test/testing_task_2.rb:30:in `cards_total'     specs/card_spec.rb:26:in `test_cards_total' 3 runs, 2 assertions, 0 failures, 1 errors, 0 skips	
			 ➜ ruby_test git:(master) ✗ ruby specs/card_spec.rb Run options: --seed 36399 # Running: ... Finished in 0.001033s, 2904.1624 runs/s, 2904.1624 assertions/s. 3 runs, 3 assertions, 0 failures, 0 errors, 0 skips	



```ruby
def test_cards_total()
  assert_equal("You have a total of 15.", @cardgame.cards_total[@card1, @card2, @card3])
end
```

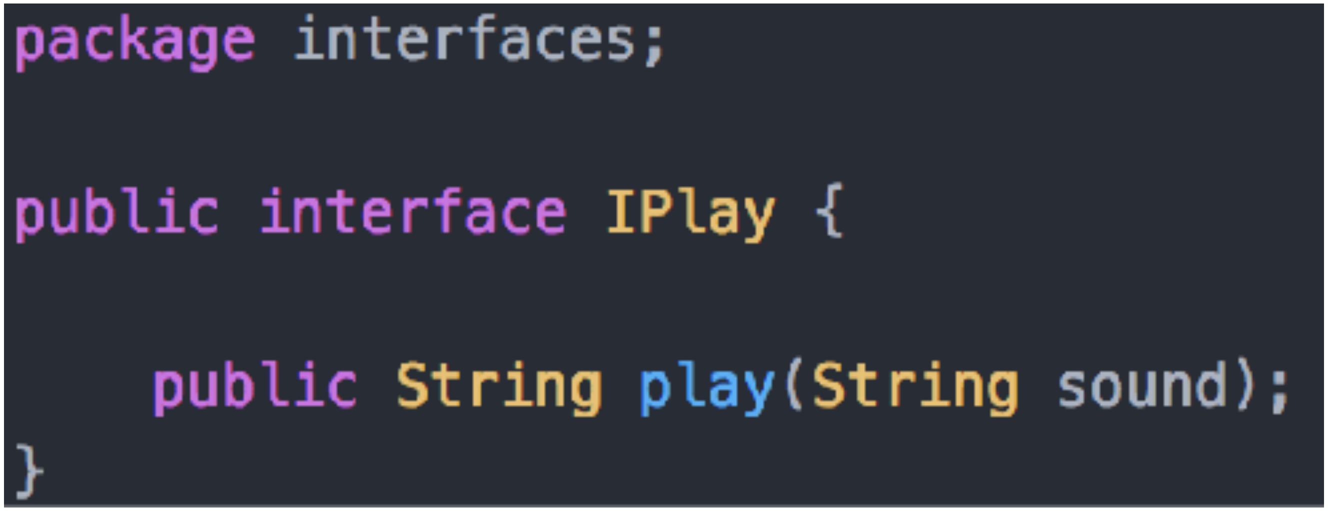


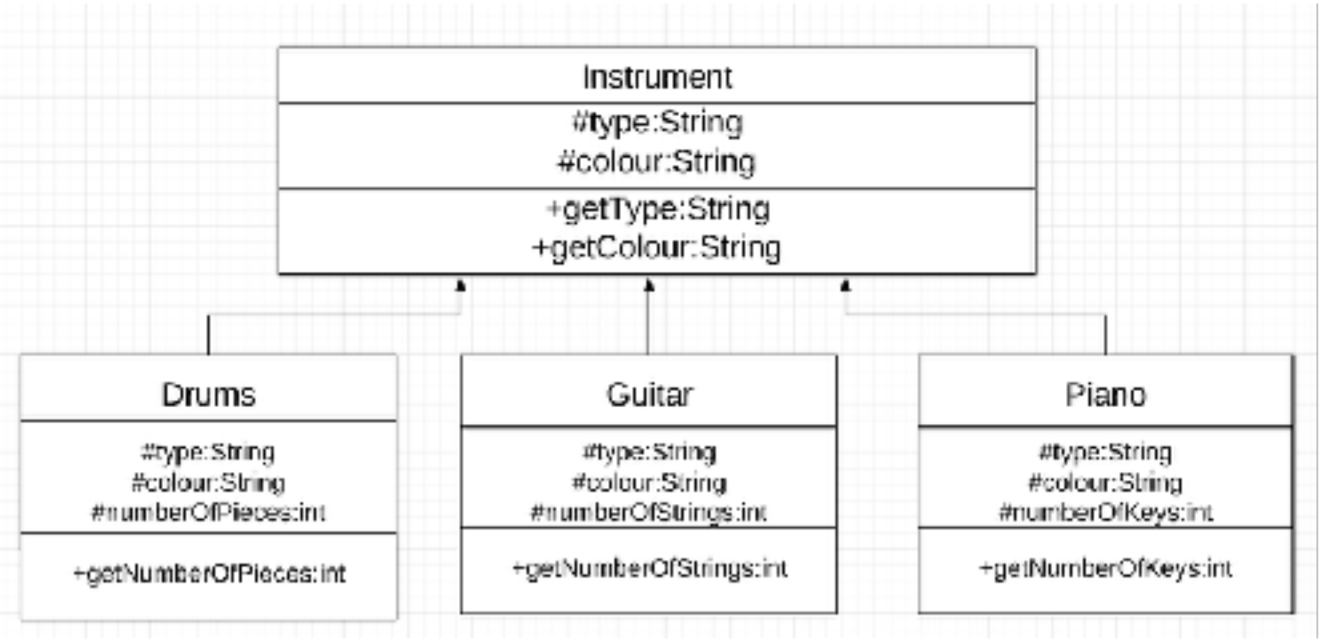
```ruby
class CardTest < MiniTest::Test

  def setup()
    @card1 = Card.new("Hearts", 5)
    @card2 = Card.new("Diamonds", 9)
    @card3 = Card.new("Spades", 1)
    @cardgame = CardGame.new([@card1, @card2, @card3])
  end

  def test_checkforace()
    assert_equal(true, @cardgame.checkforace(@card3))
  end
  #
  def test_highestcard()
    assert_equal("Diamonds", @cardgame.highest_card(@card1, @card2))
  end
  #
  def test_cards_total()
    assert_equal("You have a total of 15.", @cardgame.cards_total([@card1, @card2, @card3]))
  end

end
```

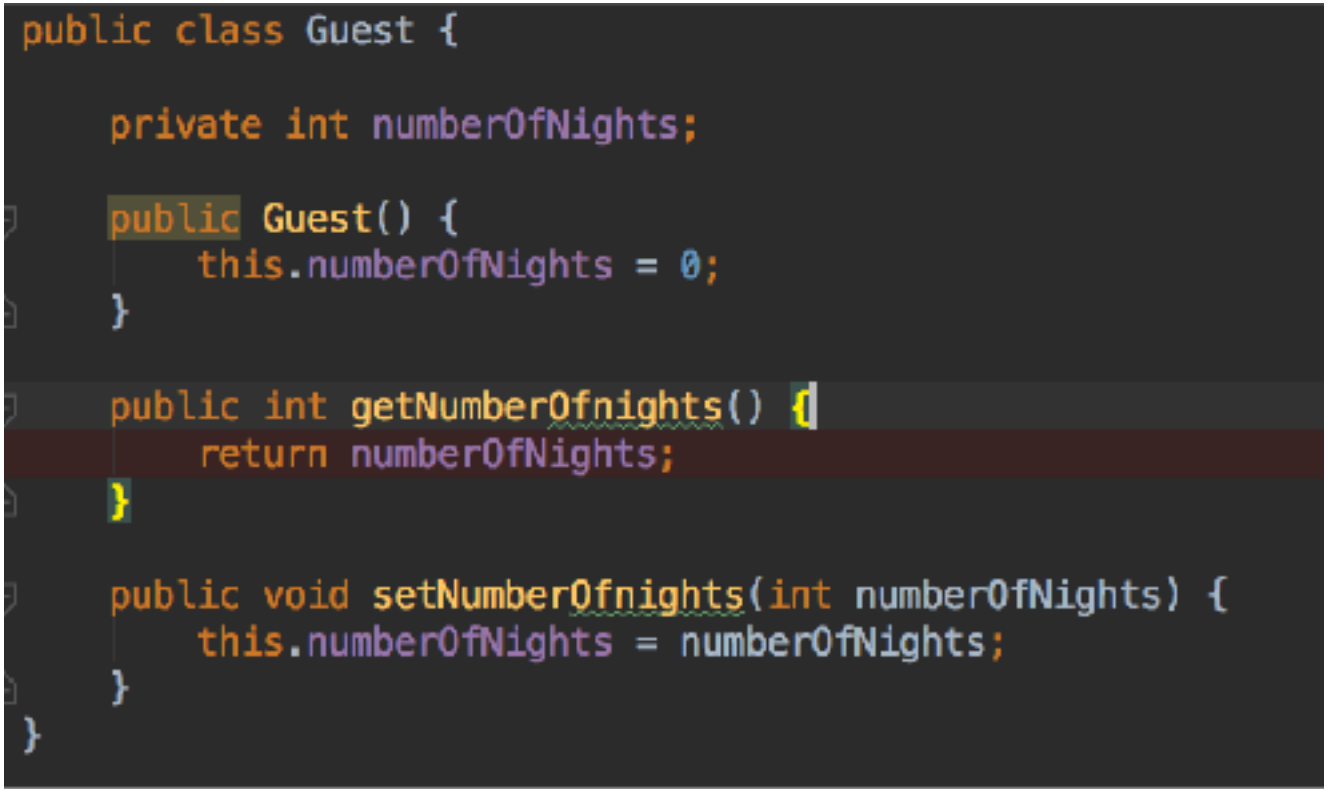
	Unit	Ref.	Evidence	Done
	I & T	I.T 7	Demonstrate the use of Polymorphism in a program.	
			 	

```java
import interfaces.IPlay;
import interfaces.ISell;
import items.Item;

public abstract class Instrument extends Item implements IPlay, ISell {

    String type;
    String colour;
    IPlay iPlay;

    public Instrument(String description, double buyPrice, double
    sellPrice, String type, String colour) {
        super(description, buyPrice, sellPrice);
        this.type = type;
        this.colour = colour;

    }
```

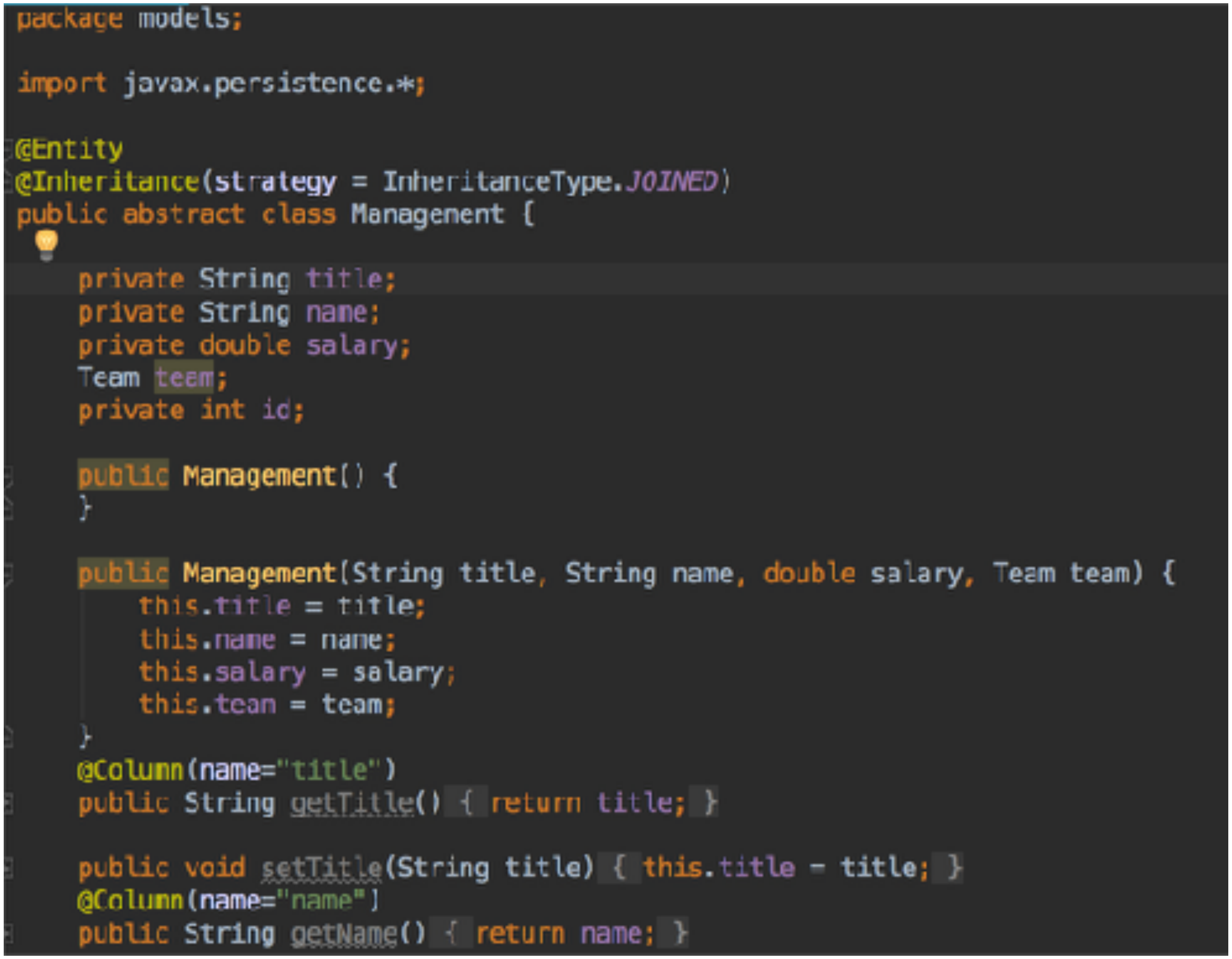
```java
public String play(String sound){
    return this.iPlay.play(sound);
}
```

		package interfaces;
		
		public interface IPlay {
		
		    public String play(String sound);
		}	

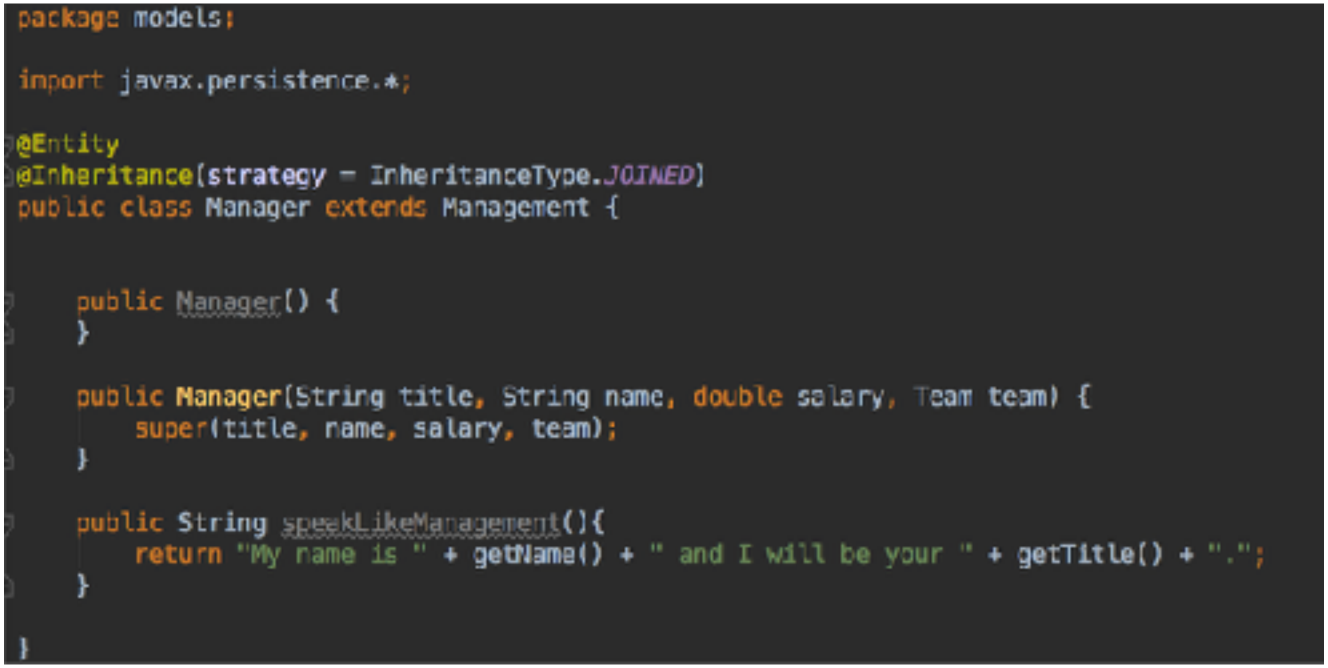
A & D	A.D 5	An Inheritance Diagram	
		 Instrument #type:String #colour:String +getType:String +getColour:String Drums #type:String #colour:String #numberOfPieces:int +getNumberOfPieces:int Guitar #type:String #colour:String #numberOfStrings:int +getNumberOfStrings:int Piano #type:String #colour:String #numberOfKeys:int +getNumberOfKeys:int	
I & T	I.T 1	Take a screenshot of an example of encapsulation in a program.	

Week 10				
	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.	

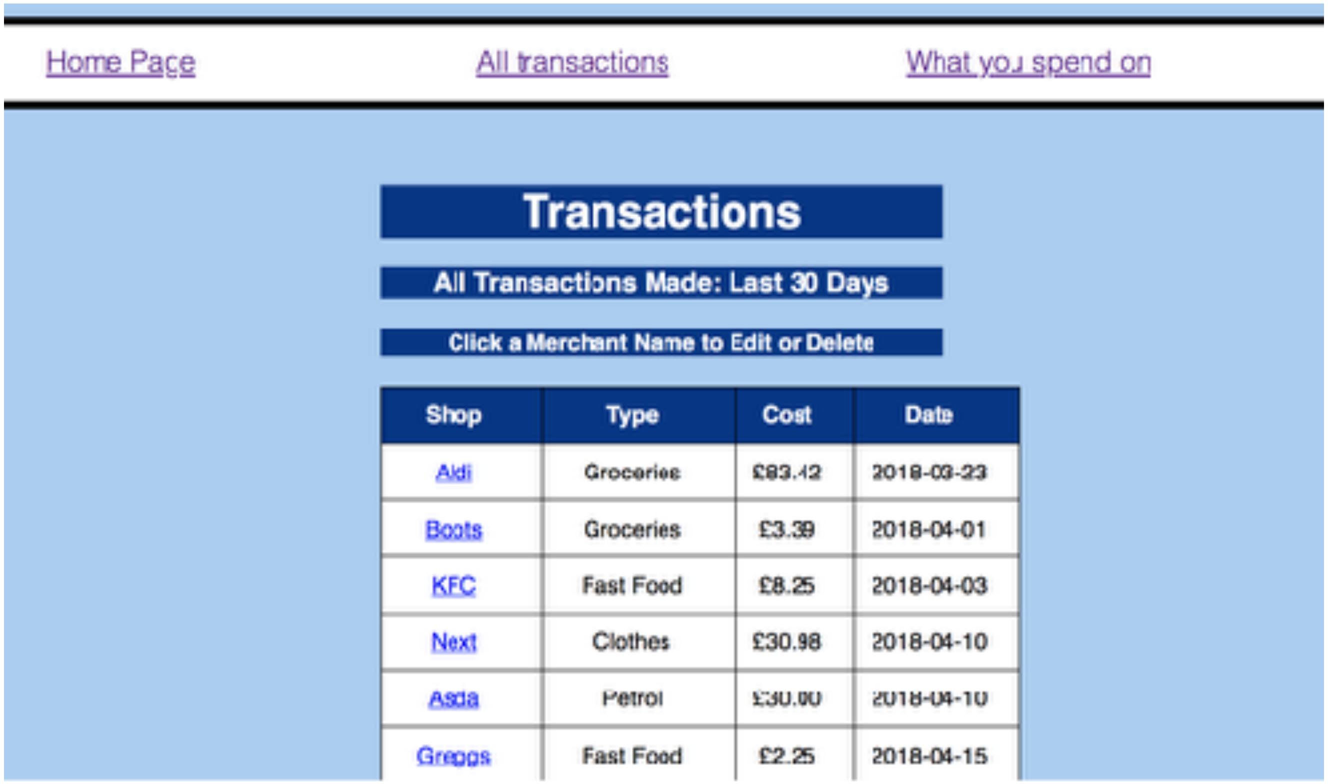
```
public class Guest {

    private int numberOfNights;

    public Guest() {
        this.numberOfNights = 0;
    }

    public int getNumberOfnights() {
        return numberOfNights;
    }

    public void setNumberOfnights(int numberOfNights) {
        this.numberOfNights = numberOfNights;
    }
}
```



```
package models;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Management {

    private String title;
    private String name;
    private double salary;
    Team team;
    private int id;

    public Management() {
    }

    public Management(String title, String name, double salary, Team team) {
        this.title = title;
        this.name = name;
        this.salary = salary;
        this.team = team;
    }
    @Column(name="title")
    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }
    @Column(name="name")
    public String getName() { return name; }
```

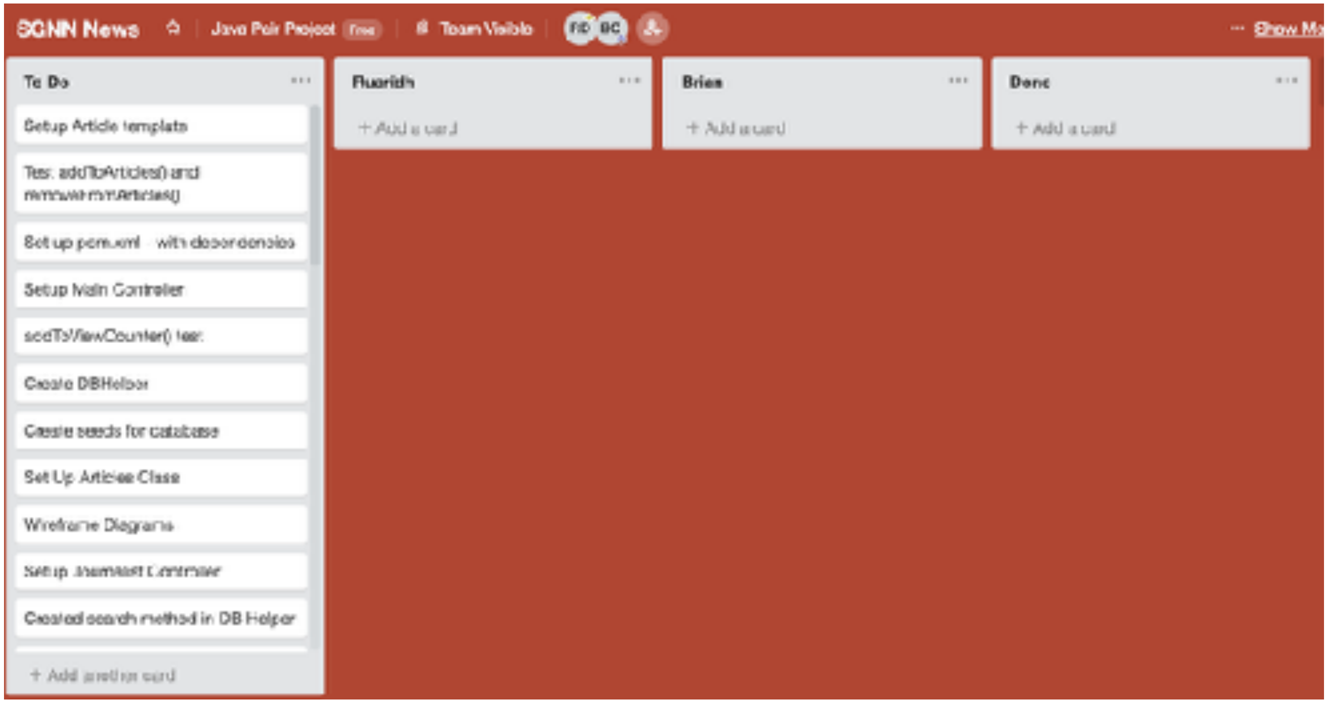
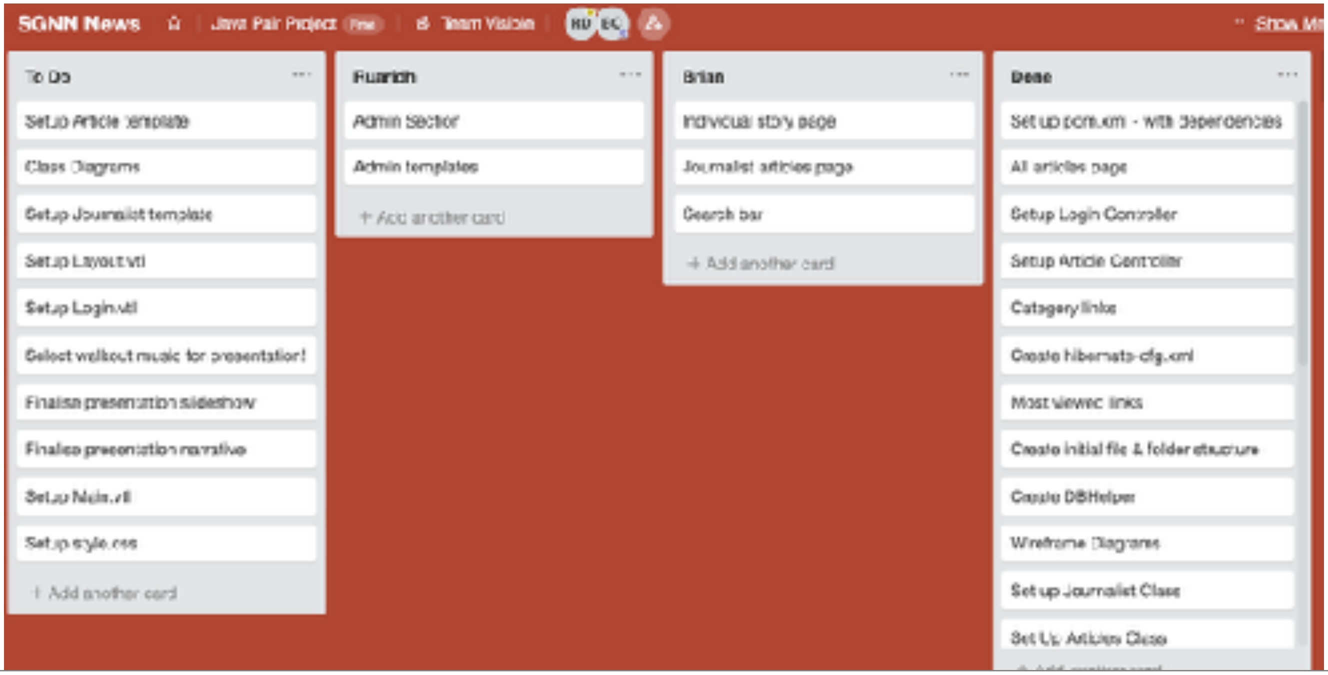
				
	P	P 11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	

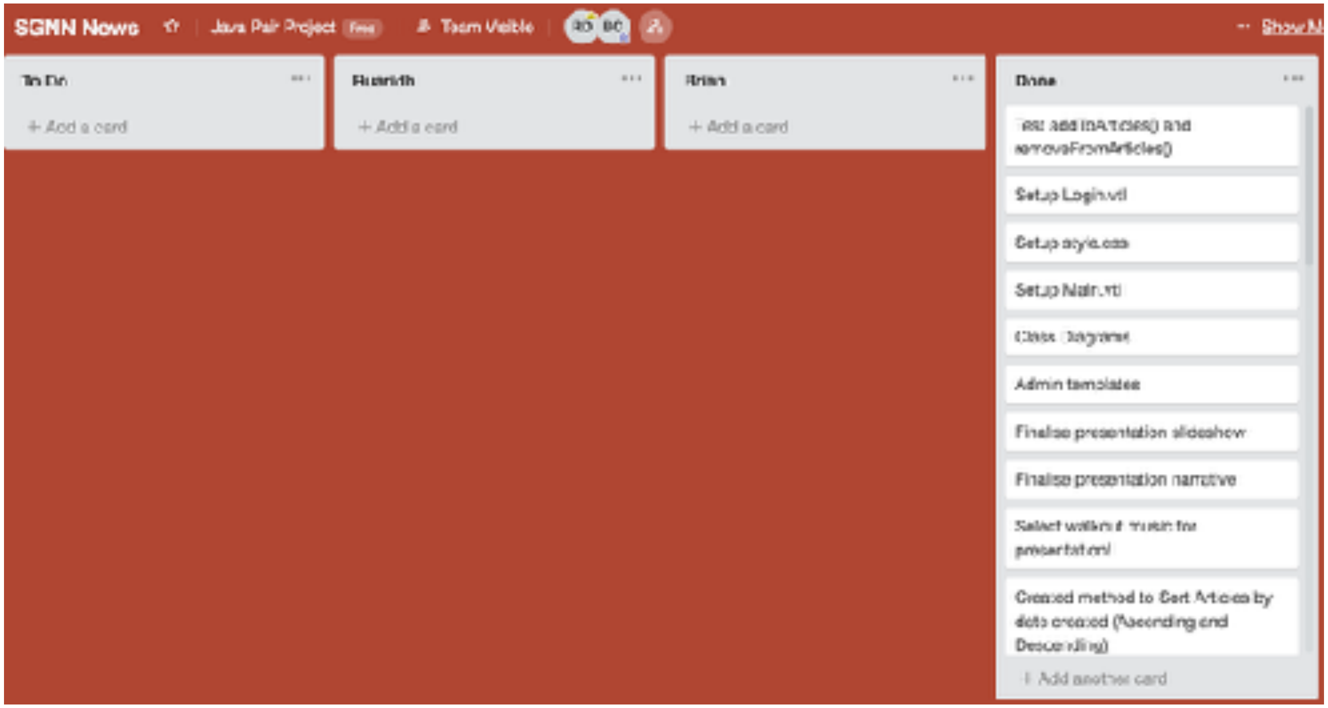
```java
package models;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class Manager extends Management {


    public Manager() {
    }

    public Manager(String title, String name, double salary, Team team) {
        super(title, name, salary, team);
    }

    public String speakLikeManagement(){
        return "My name is " + getName() + " and I will be your " + getTitle() + ".";
    }

}
```

<table>
<tr><td></td><td></td><td></td><td>
Home Page All transactions What you spend on

<b>Transactions</b>
All Transactions Made: Last 30 Days
Click a Merchant Name to Edit or Delete
<table>
<tr><th>Shop</th><th>Type</th><th>Cost</th><th>Date</th></tr>
<tr><td>Aldi</td><td>Groceries</td><td>£83.42</td><td>2018-03-23</td></tr>
<tr><td>Boots</td><td>Groceries</td><td>£3.39</td><td>2018-04-01</td></tr>
<tr><td>KFC</td><td>Fast Food</td><td>£8.25</td><td>2018-04-03</td></tr>
<tr><td>Next</td><td>Clothes</td><td>£30.98</td><td>2018-04-10</td></tr>
<tr><td>Asda</td><td>Petrol</td><td>£30.00</td><td>2018-04-10</td></tr>
<tr><td>Greggs</td><td>Fast Food</td><td>£2.25</td><td>2018-04-15</td></tr>
</table>

https://github.com/RuaridhD/moneycashboard_project</td><td></td></tr>
<tr><td></td><td>P</td><td>P 12</td><td>Take screenshots or photos of your planning and the different stages of development to show changes.</td><td></td></tr>
</table>

			 SGNN News Java Pair Project Free Team Visible Show M To Do Setup Article template Test addToArticles() and removeFromArticles() Set up pom.xml - with dependencies Setup Main Controller addToViewCounter() test Create DBHelper Create seeds for database Set Up Articles Class Wireframe Diagrams Setup Journalist Controller Created search method in DB Helper + Add another card Ruaridh + Add a card Brian + Add a card Done + Add a card  SGNN News Java Pair Project Free Team Visible Show M To Do Setup Article template Class Diagrams Setup Journalist template Setup Layout.vtl Setup Login.vtl Select walkout music for presentation! Finalise presentation slideshow Finalise presentation narrative Setup Main.vtl Setup style.css + Add another card Ruaridh Admin Section Admin templates + Add another card Brian Individual story page Journalist articles page Search bar + Add another card Done Set up pom.xml - with dependencies All articles page Setup Login Controller Setup Article Controller Category links Create hibernate-cfg.xml Most viewed links Create initial file & folder structure Create DBHelper Wireframe Diagrams Set up Journalist Class Set Up Articles Class	

Test addToArticles() and removeFromArticles()

Setup Login.vtl

Setup style.css

Setup Main.vtl

Class Diagrams

Admin templates

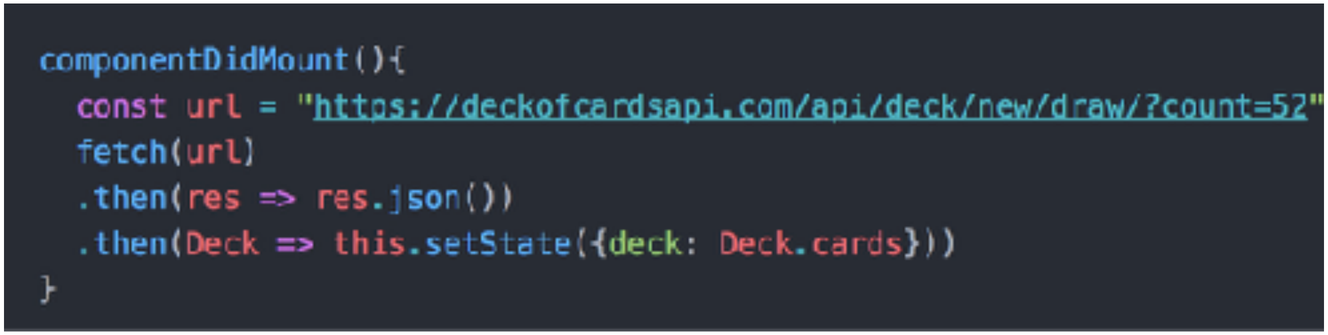
Finalise presentation slideshow

Finalise presentation narrative

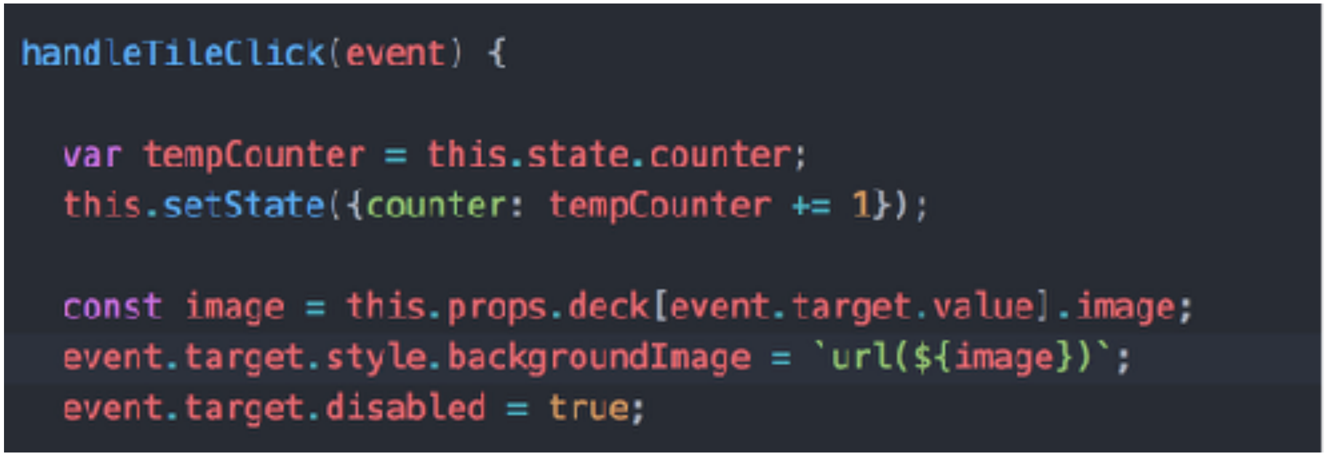
Select walkout music for presentation!

Created method to Sort Articles by date created (Ascending and Descending)

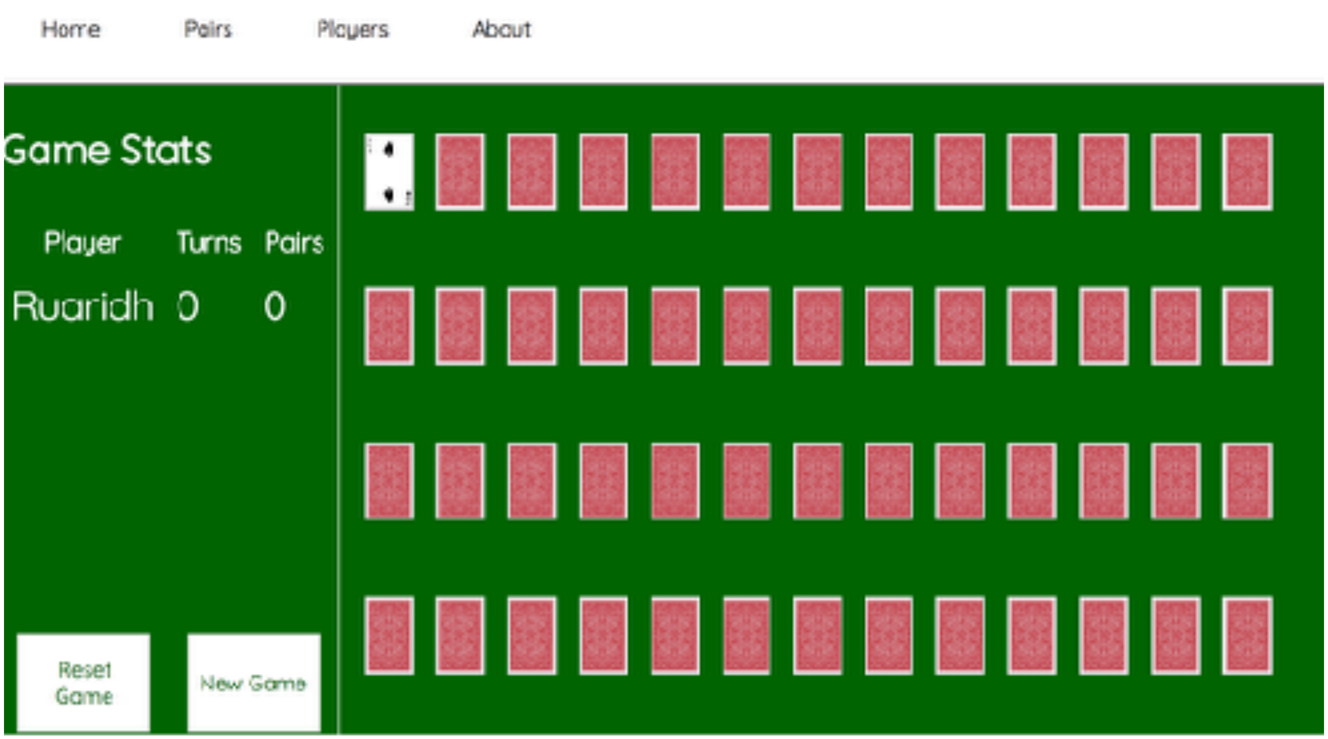
	Unit	Ref.	Evidence	Done
Week 12	I & T		Unit, integration and acceptance testing task B	
	P	P 16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	

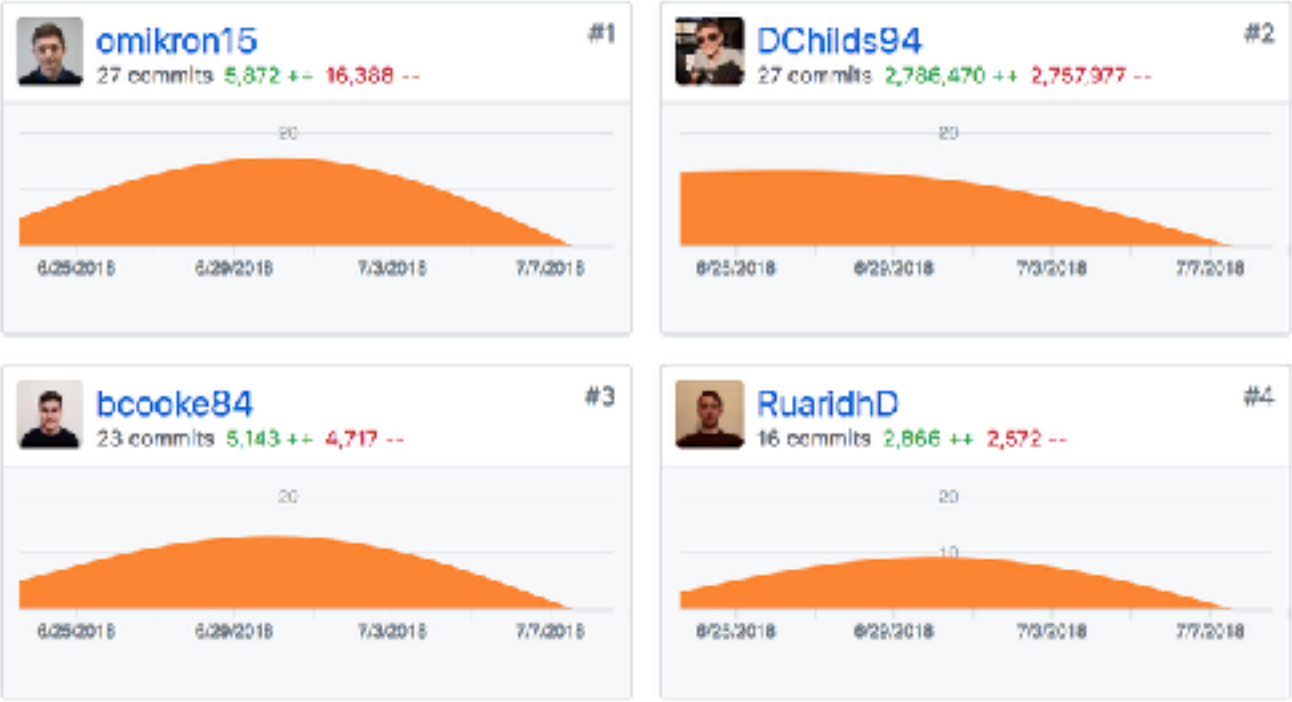


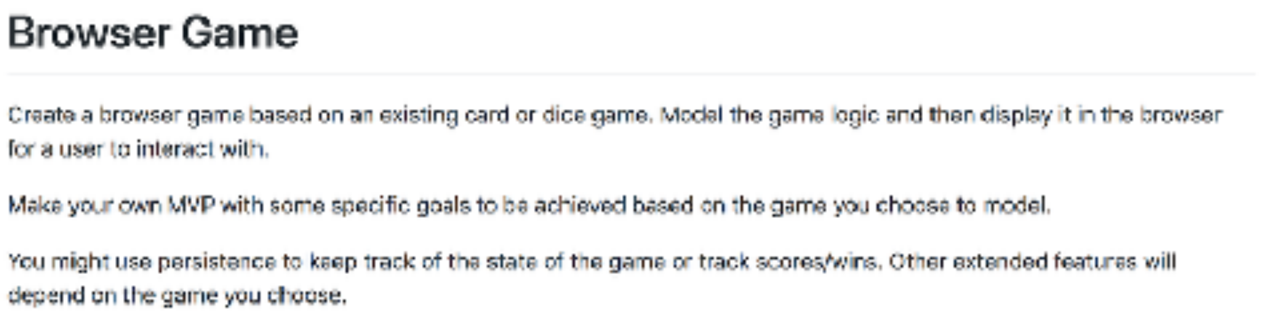
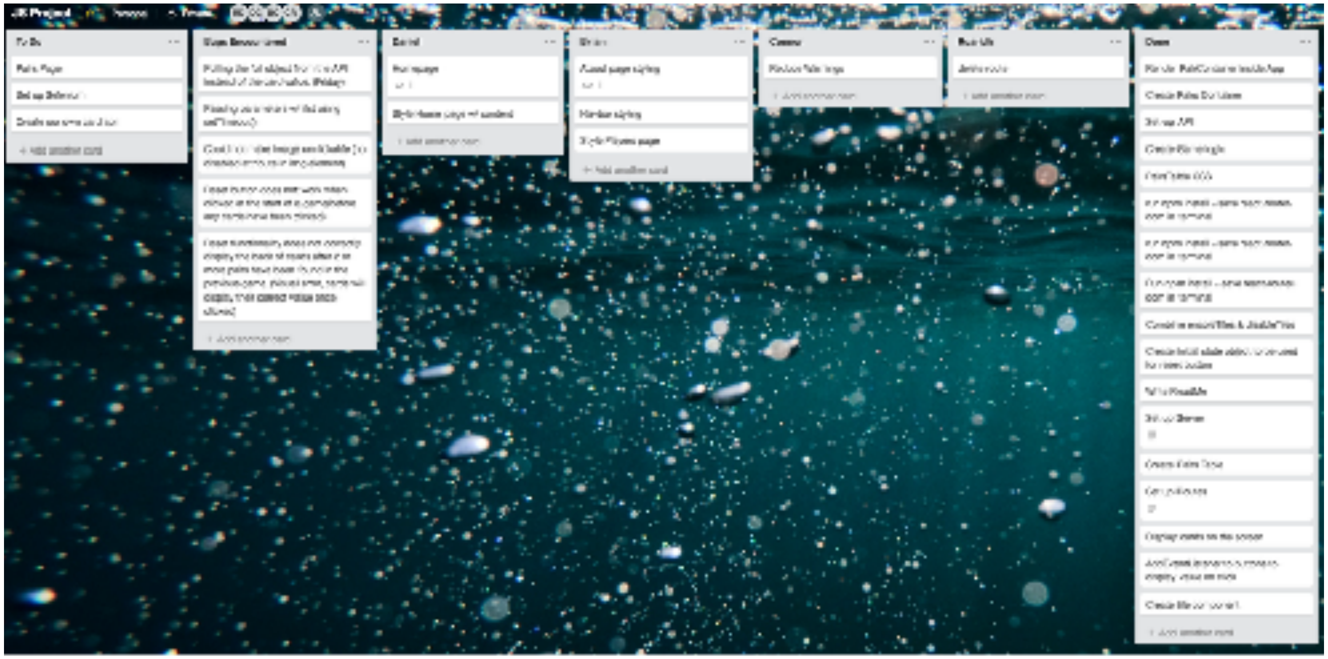
```
componentDidMount(){
  const url = "https://deckofcardsapi.com/api/deck/new/draw/?count=52"
  fetch(url)
  .then(res => res.json())
  .then(Deck => this.setState({deck: Deck.cards}))
}
```

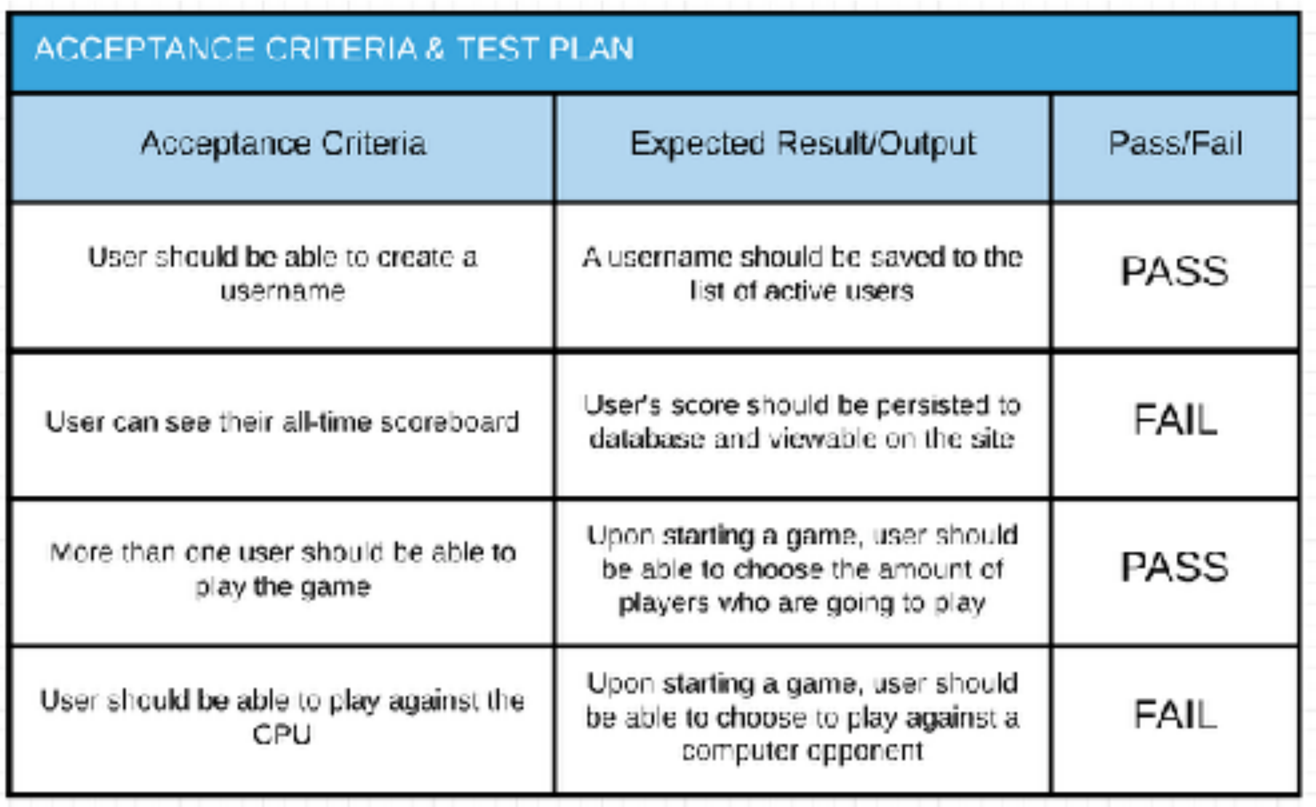


```
handleTileClick(event) {

  var tempCounter = this.state.counter;
  this.setState({counter: tempCounter += 1});

  const image = this.props.deck[event.target.value].image;
  event.target.style.backgroundImage = `url(${image})`;
  event.target.disabled = true;
```

			 Home Pairs Players About Game Stats Reset Game New Game Player Turns Pairs Ruaridh 0 0	

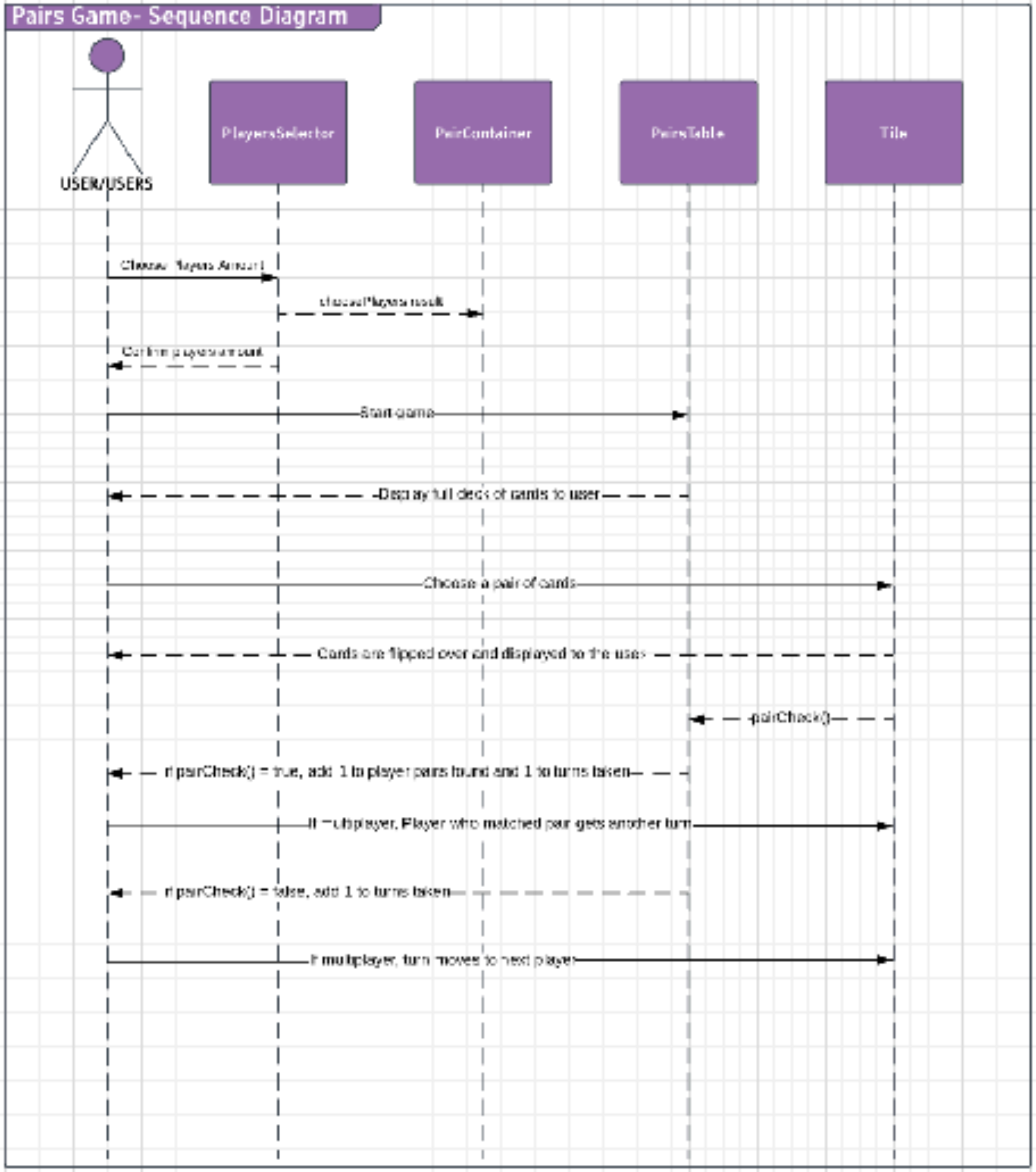
	Unit	Ref.	Evidence	Done
	P	P 1	Take a screenshot of the contributor’s page on Github from your group project to show the team you worked with.	
			 omikron15 #1 27 commits 5,872 ++ 16,388 -- DChilds94 #2 27 commits 2,786,470 ++ 2,757,977 -- bcooke84 #3 23 commits 5,143 ++ 4,717 -- RuaridhD #4 16 commits 2,866 ++ 2,572 -- 	
	P	P 2	Take a screenshot of the project brief from your group project.	

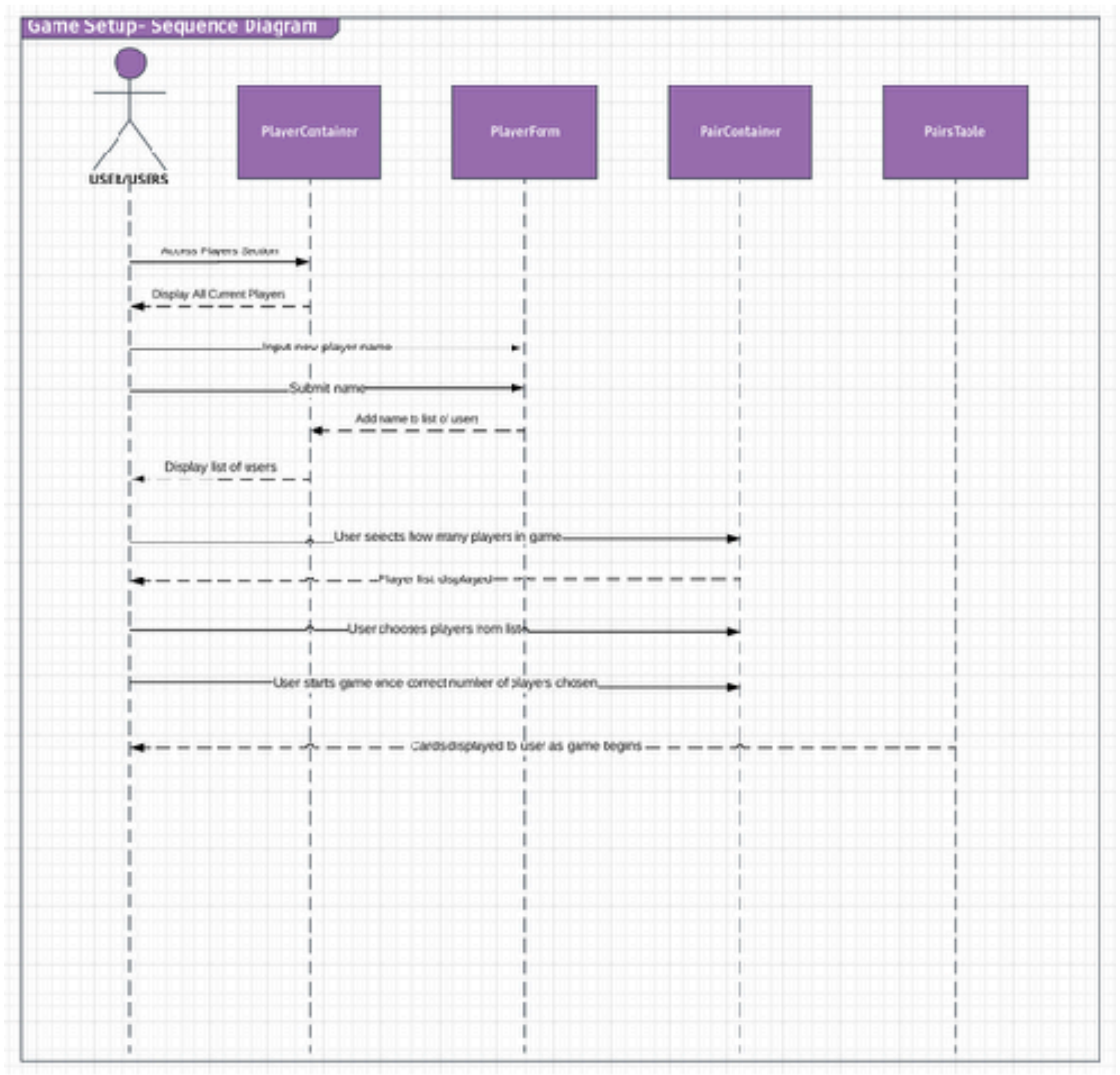
		 **Browser Game** Create a browser game based on an existing card or dice game. Model the game logic and then display it in the browser for a user to interact with. Make your own MVP with some specific goals to be achieved based on the game you choose to model. You might use persistence to keep track of the state of the game or track scores/wins. Other extended features will depend on the game you choose.	
P	P 3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	
		 JS Project To Do Done Set up API + Add another card	
P	P 4	Write an acceptance criteria and test plan.	

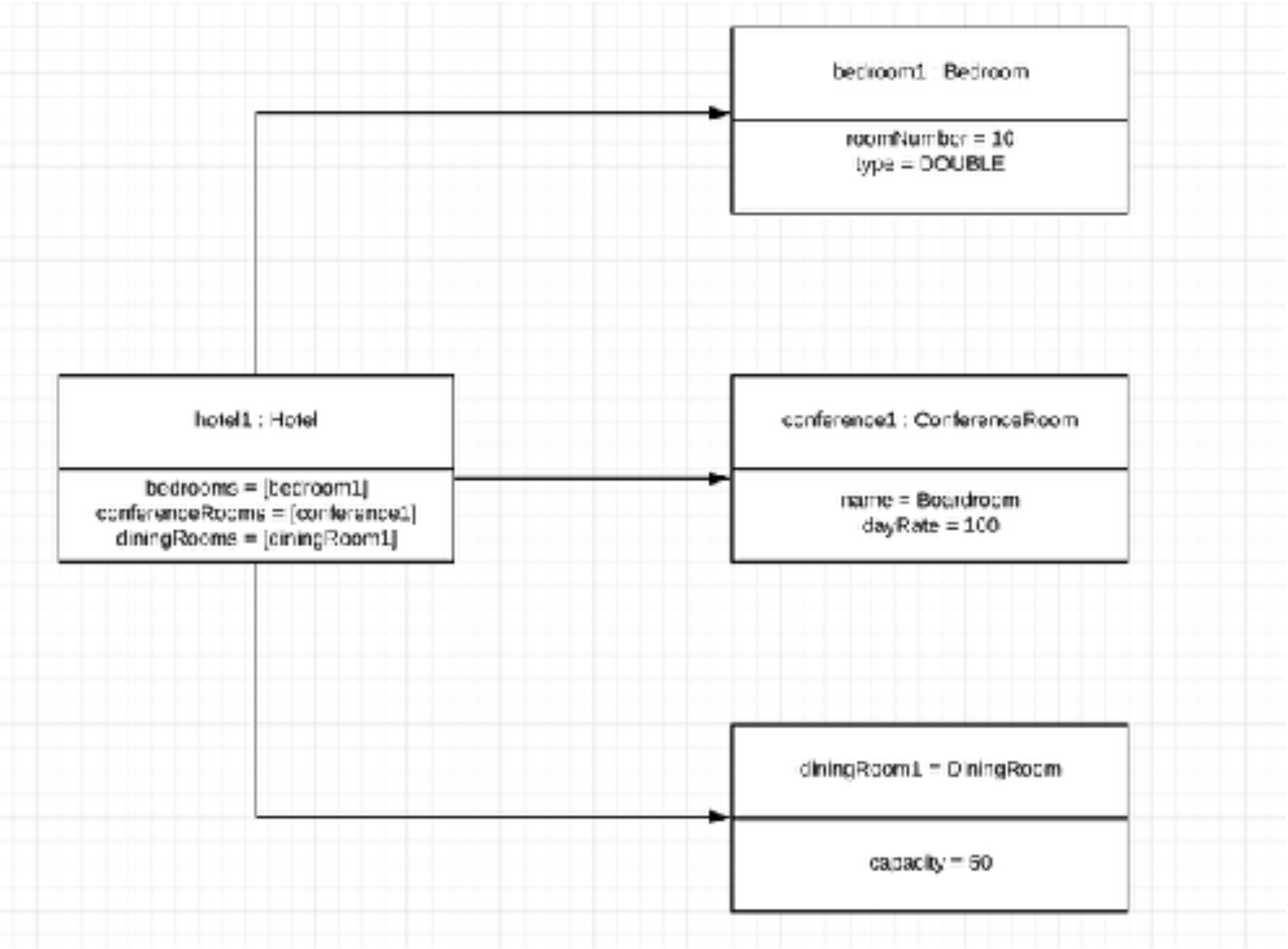


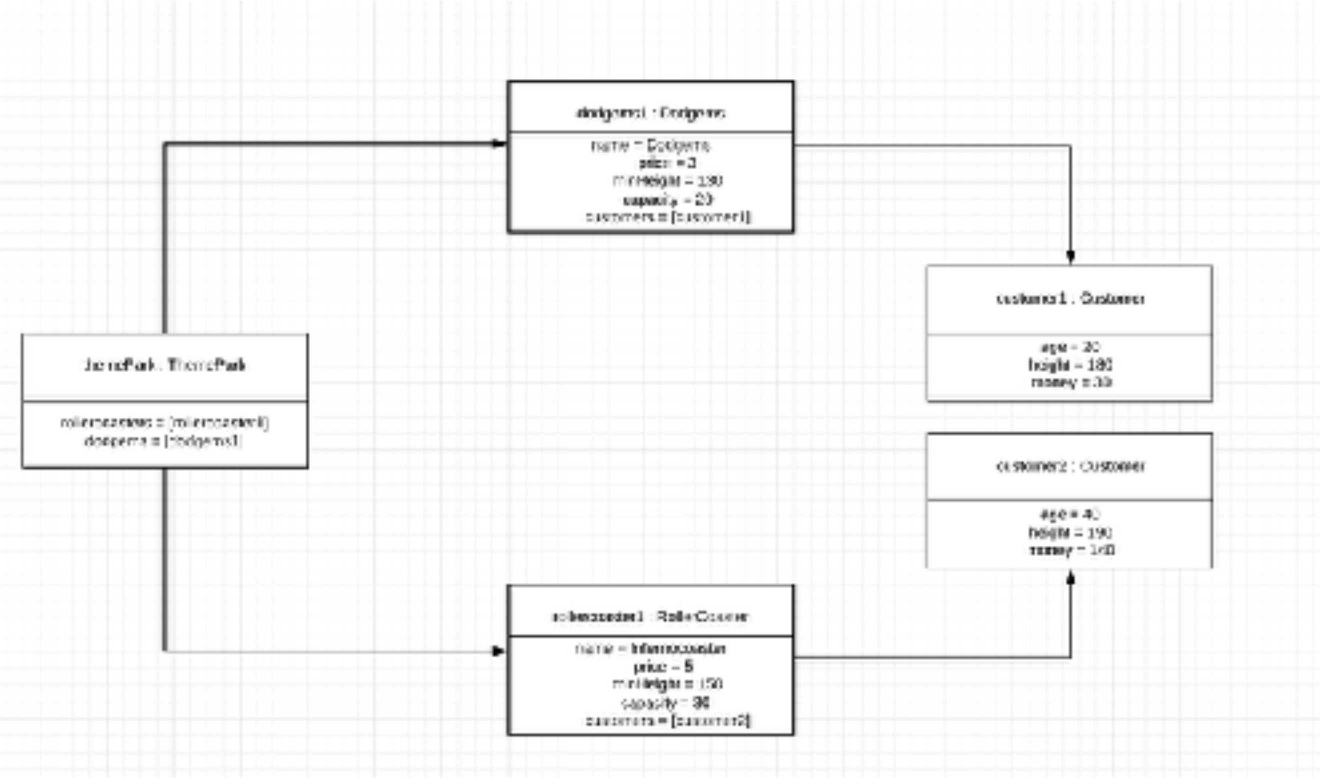
ACCEPTANCE CRITERIA & TEST PLAN		
Acceptance Criteria	Expected Result/Output	Pass/Fail
User should be able to create a username	A username should be saved to the list of active users	PASS
User can see their all-time scoreboard	User's score should be persisted to database and viewable on the site	FAIL
More than one user should be able to play the game	Upon starting a game, user should be able to choose the amount of players who are going to play	PASS
User should be able to play against the CPU	Upon starting a game, user should be able to choose to play against a computer opponent	FAIL

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

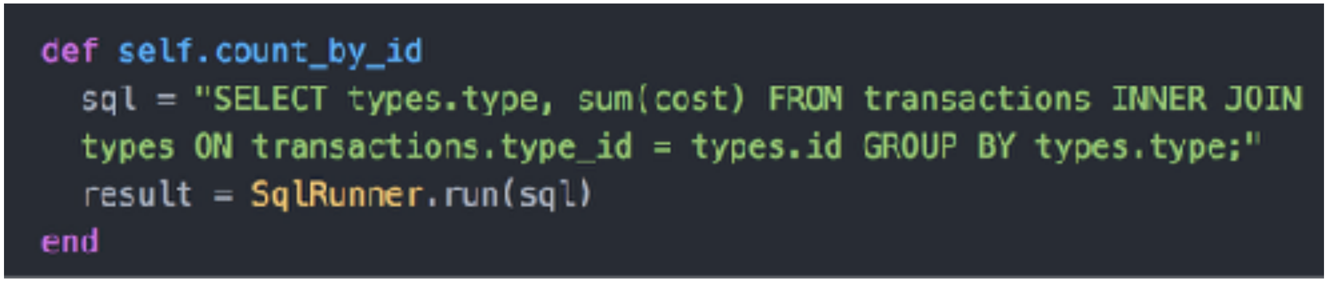

Pairs Game- Sequence Diagram
USER/USERS
PlayersSelector
PairContainer
PairsTable
Tile
Choose Players Amount
choosePlayers result
Confirm players amount
Start game
Display full deck of cards to user
Choose a pair of cards
Cards are flipped over and displayed to the user
pairCheck()
If pairCheck() = true, add 1 to player pairs found and 1 to turns taken
If multiplayer, Player who matched pair gets another turn
If pairCheck() = false, add 1 to turns taken
If multiplayer, turn moves to next player

Week 14			 Game Setup- Sequence Diagram USER/USERS PlayerContainer PlayerForm PairContainer PairsTable Access Players Section Display All Current Players Input new player name Submit name Add name to list of users Display list of users User selects how many players in game Player list displayed User chooses players from list User starts game once correct number of players chosen Cards displayed to user as game begins	
	P	P 8	Produce two object diagrams.	

		 bedroom1 : Bedroom
		roomNumber = 10
		type = DOUBLE
		hotel1 : Hotel
		bedrooms = [bedroom1]
		conferenceRooms = [conference1]
		diningRooms = [diningRoom1]
		conference1 : ConferenceRoom
		name = Boardroom
		dayRate = 100
		diningRoom1 : DiningRoom
		capacity = 50
		This is an object diagram representing a hotel containing various rooms and their corresponding properties.	

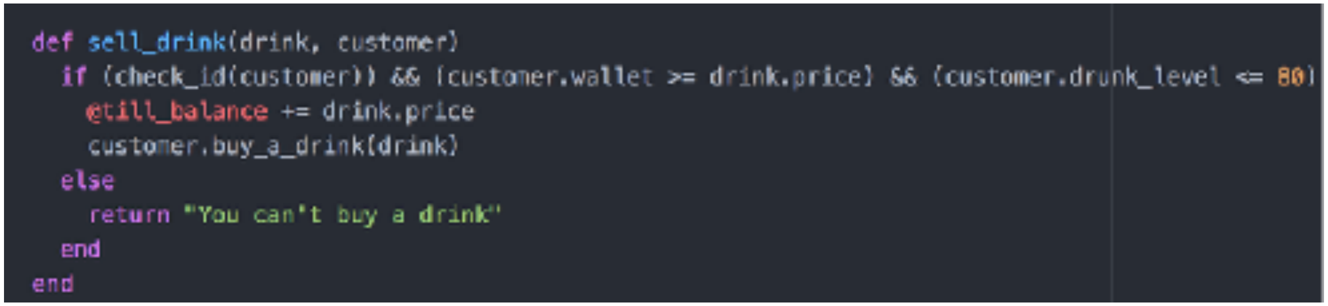
<table>
<tr>
<td></td>
<td></td>
<td>


This is an object diagram representing a Theme Park and the relationships between the rides and the customers of the park.
</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>P 9</td>
<td>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.</td>
<td></td>
</tr>
</table>



```
def self.count_by_id
  sql = "SELECT types.type, sum(cost) FROM transactions INNER JOIN
  types ON transactions.type_id = types.id GROUP BY types.type;"
  result = SqlRunner.run(sql)
end
```

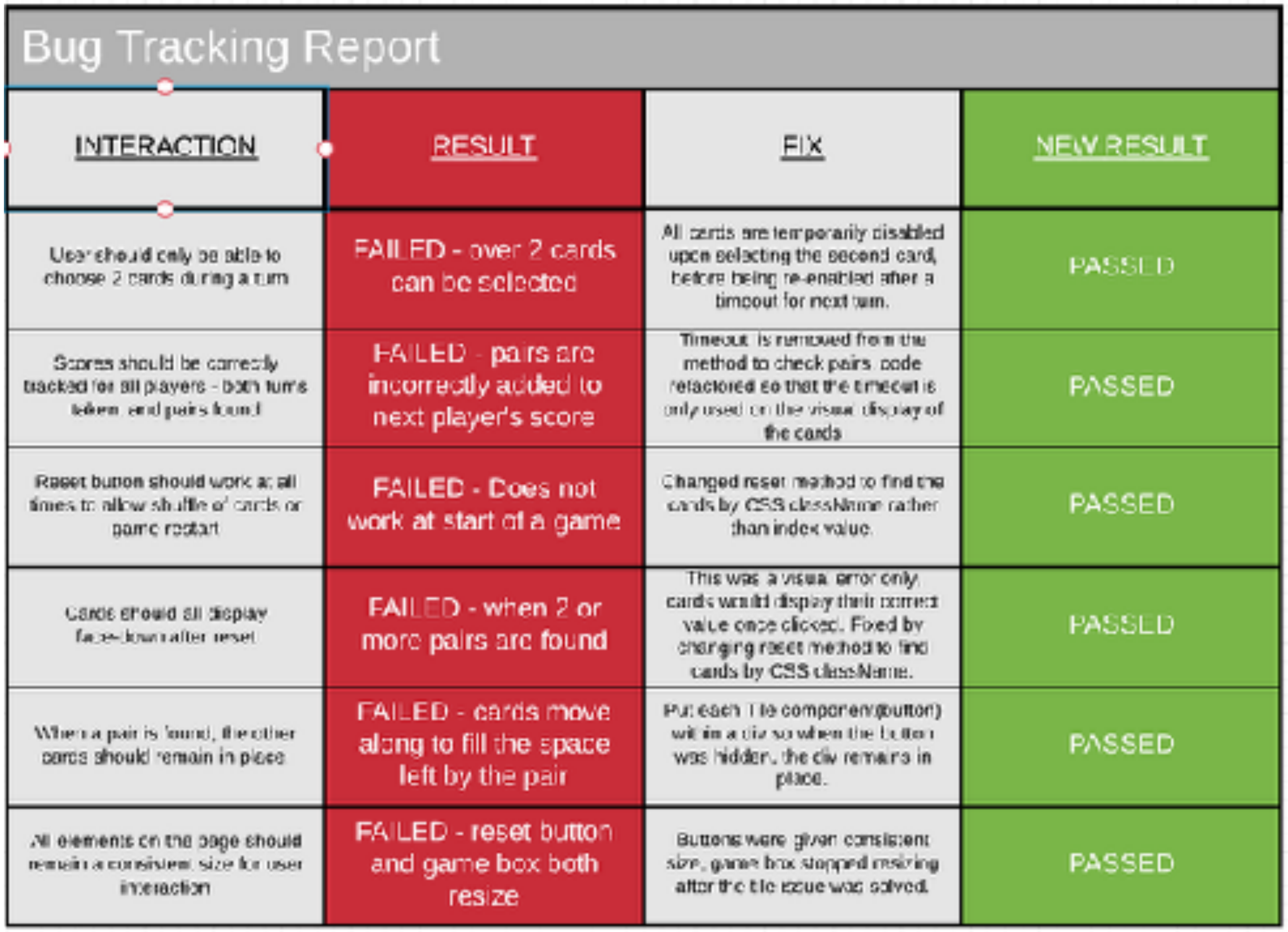
This method is from my Ruby budget tracker project, and accesses the database to gather the cost of each transaction in the database, which is then totalled and grouped by type of transaction. I then use this method later to display this information to a user in a table using another method so they can view what type of spending costs the most. I have chosen to use this algorithm as it was the most complex SQL query I used in my project, and the method I was most proud of.



```
def sell_drink(drink, customer)
  if (check_id(customer)) && (customer.wallet >= drink.price) && (customer.drunk_level <= 80)
    @till_balance += drink.price
    customer.buy_a_drink(drink)
  else
    return "You can't buy a drink"
  end
end
```

This method is from a project designing a pub. This does various checks as part of an if statement. First of all calls the ‘check_id’ method to ensure the customer is 18 or over is true. Then there is a check to make sure the customer has enough money for the requested drink. Lastly it does a check to make sure the customer is sober enough to buy the drink, set against a ‘drunk_level’ indicator. All three of these must be true in order for the drink to be sold. Otherwise a string is returned to say the drink can’t be purchased. I’ve chosen to use this algorithm because it was one of the first complex methods we had to create by ourselves near the start of the course.

	P	P 17	Produce a bug tracking report	

Bug Tracking Report

INTERACTION	RESULT	FIX	NEW RESULT
User should only be able to choose 2 cards during a turn	FAILED - over 2 cards can be selected	All cards are temporarily disabled upon selecting the second card, before being re-enabled after a timeout for next turn.	PASSED
Scores should be correctly tracked for all players - both turns taken and pairs found	FAILED - pairs are incorrectly added to next player's score	Timeout is removed from the method to check pairs, code refactored so that the timeout is only used on the visual display of the cards	PASSED
Reset button should work at all times to allow shuffle of cards or game restart	FAILED - Does not work at start of a game	Changed reset method to find the cards by CSS className rather than index value.	PASSED
Cards should all display face-down after reset	FAILED - when 2 or more pairs are found	This was a visual error only, cards would display their correct value once clicked. Fixed by changing reset method to find cards by CSS className.	PASSED
When a pair is found, the other cards should remain in place	FAILED - cards move along to fill the space left by the pair	Put each Tile component(button) within a div so when the button was hidden, the div remains in place.	PASSED
All elements on the page should remain a consistent size for user interaction	FAILED - reset button and game box both resize	Buttons were given consistent size, game box stopped resizing after the tile issue was solved.	PASSED