

Evidence Gathering
Level 8 Professional

Document for SQA
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 of what you are showing.

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	
		Description: The guest array holds guest objects, check_in_guest adds a guest to the Guest array. The final screenshot the result of the test running.	

```

class Room

  attr_reader :name, :capacity
  attr_accessor :guests, :songs

  def initialize(name, capacity)
    @name = name
    @capacity = capacity
    @guests = []
    @songs = []
  end

```

Capture of my function that utilises 'Guest_List' array.

```

def check_in_guest(guest)
  room = get_room_with_space()
  room.guests.push(guest)
  return room
end

def check_out_guest(guest, room)
  room.guests.delete(guest)
end

```

Print out result of check_inGuest screenshot result with print

```
Run options: --seed 28060
```

```
# Running:
```

```
.....
```

```
Finished in 0.001455s, 4123.7116 runs/s, 4123.7116 assertions/s.
```

```
6 runs, 6 assertions, 0 failures, 0 errors, 0 skips
```

```
→ Karaoke git:(master) █
```

Result of my function 'check_out_guest' running passing in the array 'Guest_List' as a parameter.

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	
		Description: wallets is a hash that stores peoples amount of money. get_first_key returns the first key in the hash. The final screenshot is the result of the test running.	

Hash within my program

```

62   def test_get_first_key
63     # arrange
64     wallets = {
65       "Alice" => 12,
66       "Bob"   => 10,
67       "Charlie" => 1356,
68       "Dave"  => 1
69     }
70     # act
71     result = get_first_key( wallets )
72     # assert
73     assert_equal( 'Alice', result )
74   end
75

```

Hash being used in a function

```

# get first key -
def get_first_key(hash_table)
  return hash_table.keys[0]
end

```

Result of using hash within my function

```

➔ start_point 2 ruby specs/my_functions_spec.rb
Run options: --seed 62703

# Running:

.....

Finished in 0.001184s, 4222.9740 runs/s, 4222.9740 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips
➔ start_point 2 █

```

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	
		Description: has_song uses .find to find a specific song name from within @songs. The final screenshot is the result of the test running, One for finding song and one for not finding song.	

```

24   def has_song(song_name)
25     song = @songs.find{|song| song.title == song_name }
26     # binding.pry
27     if(song == nil)
28       return false
29     end
30     return true
31   end
32

```

```

[→ Karaoke git:(master) ✖ ruby specs/room_spec.rb
Run options: --seed 5404

# Running:

.Song found...No Song.

Finished in 0.001114s, 4488.3299 runs/s, 4488.3299 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips
→ Karaoke git:(master) ✖

```

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	
		Description: most_popular_film orders the films by popularity then returns the most popular one.	

```
def self.most_popular_film
  sql = "SELECT film_id, COUNT(*) FROM tickets GROUP BY film_id ORDER BY count DESC LIMIT 1;"
  result = SqlRunner.run(sql)[0]

  sql2 = "SELECT * FROM films WHERE id = $1"
  values = [result["film_id"]]
  result2 = SqlRunner.run(sql2, values).first
  # binding.pry
  return Film.new(result2)
end
```

```
[→ weekend_homework ruby code_clan_cinema/console.rb
{"id"=>"3", "title"=>"Blade Runner", "price"=>"15", "tickets_left"=>"3"}
```

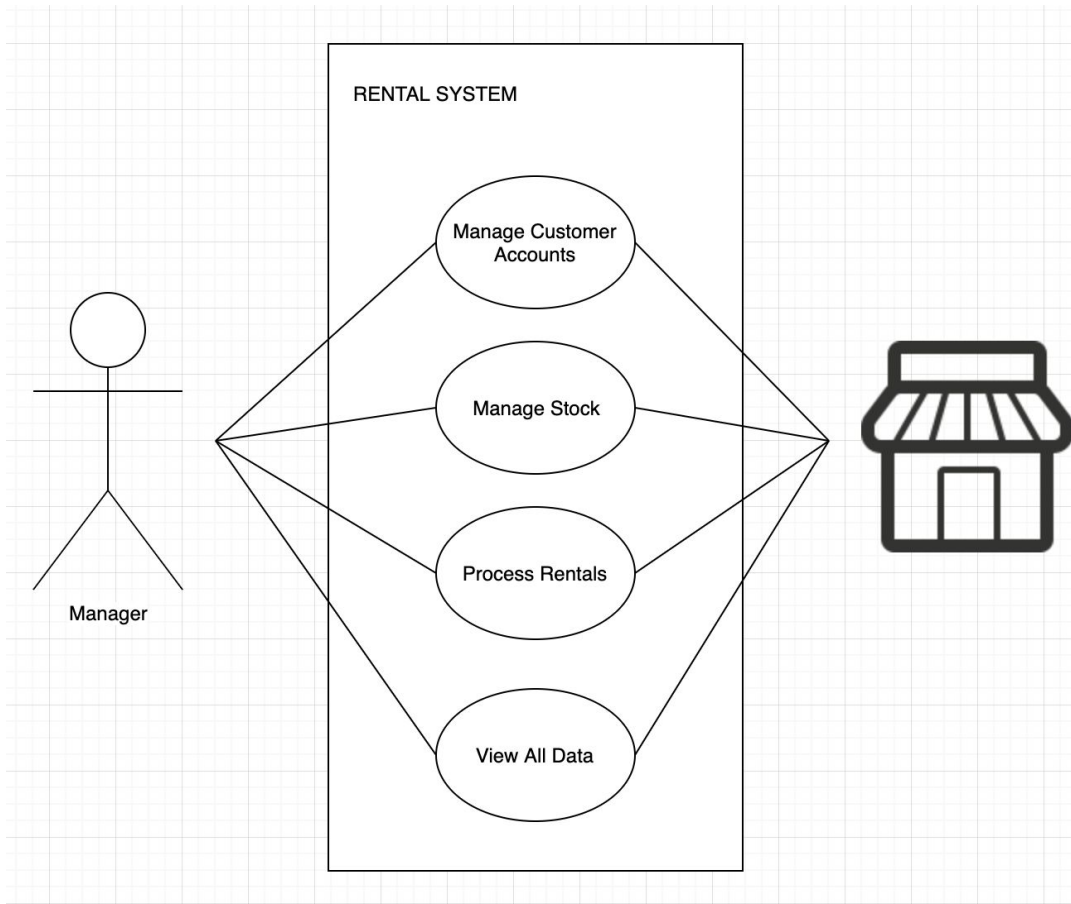
From: /Users/user/codeclan_work/week_03/weekend_homework/code_clan_cinema/console.rb @ line 69 :

```
64: # customer1.delete
65: # ticket1.delete
66:
67: binding.pry
68:
=> 69: nil
```

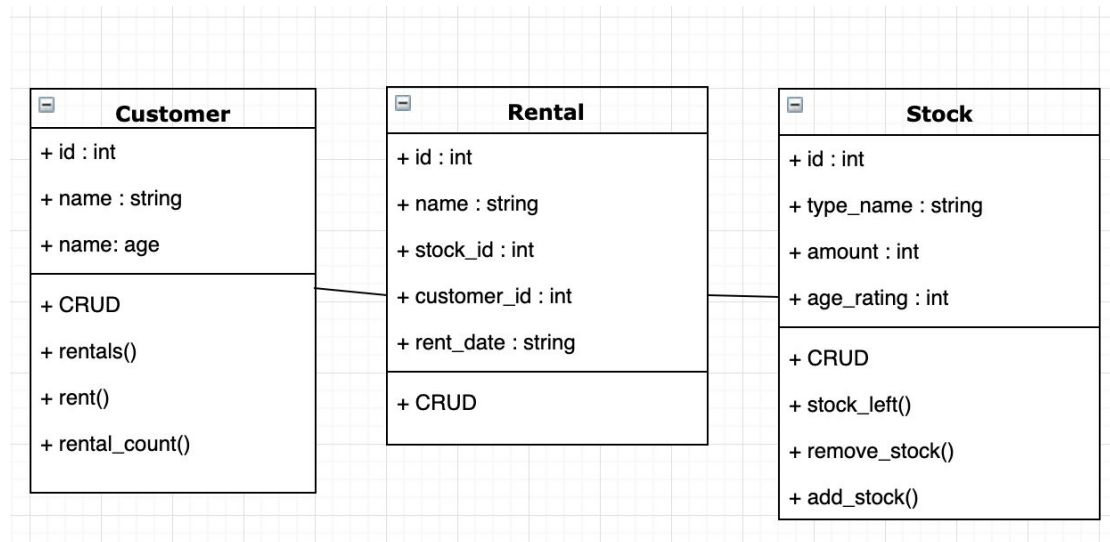
```
[1] pry(main)> █
```

Week 5 and 6

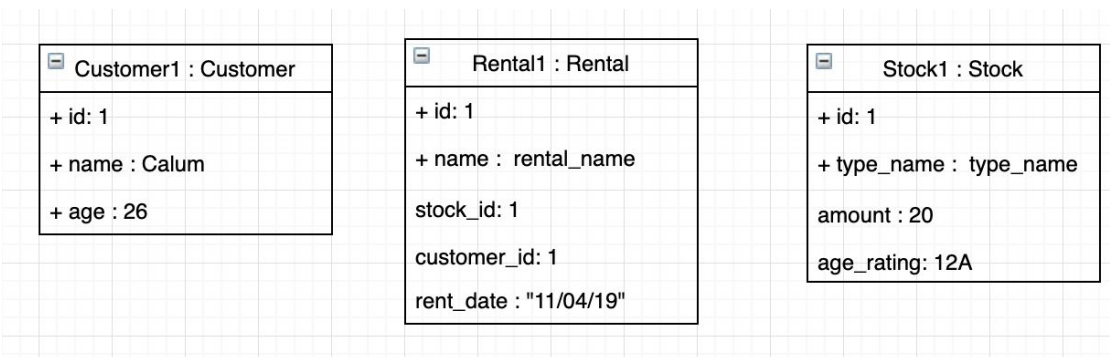
Unit	Ref	Evidence	
A&D	A.D.1	A Use Case Diagram	
		Description: This is a use case diagram for the rental stock control system.	



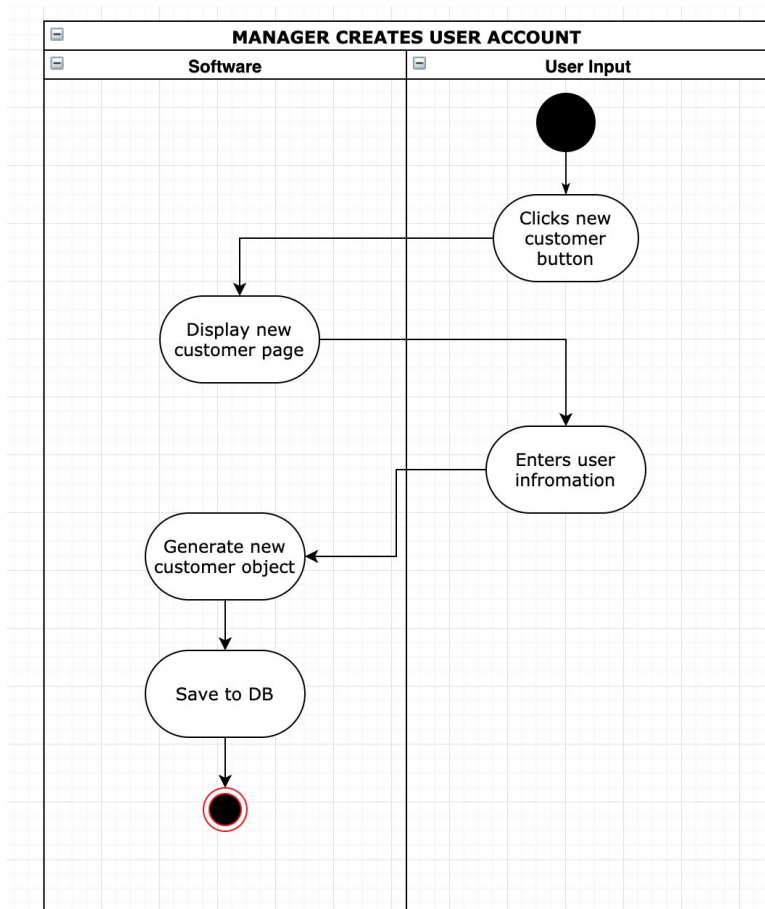
Unit	Ref	Evidence	
A&D	A.D.2	A Class Diagram	
		Description: Customer, Rental and Stock classes for the rental system	



Unit	Ref	Evidence	
A&D	A.D.3	An Object Diagram	
		Description: Object diagrams for Customer, Rental and Stock	



Unit	Ref	Evidence	
A&D	A.D.4	An Activity Diagram	
		Description: Activity diagram showing how the manager would create a new customer.	

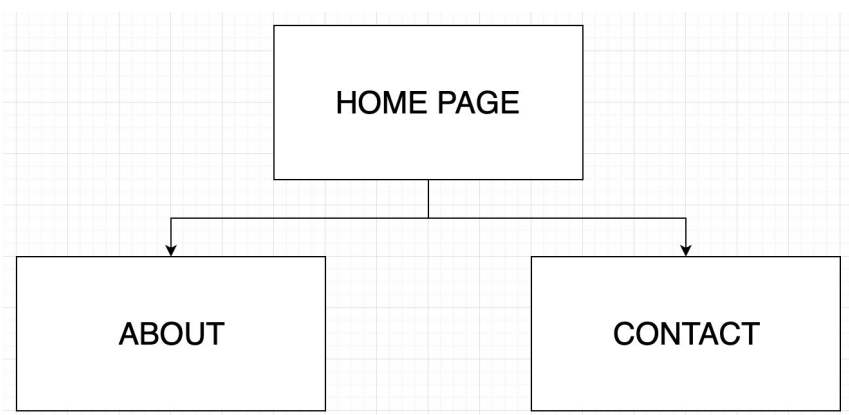


Unit	Ref	Evidence	
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	
		Description:	

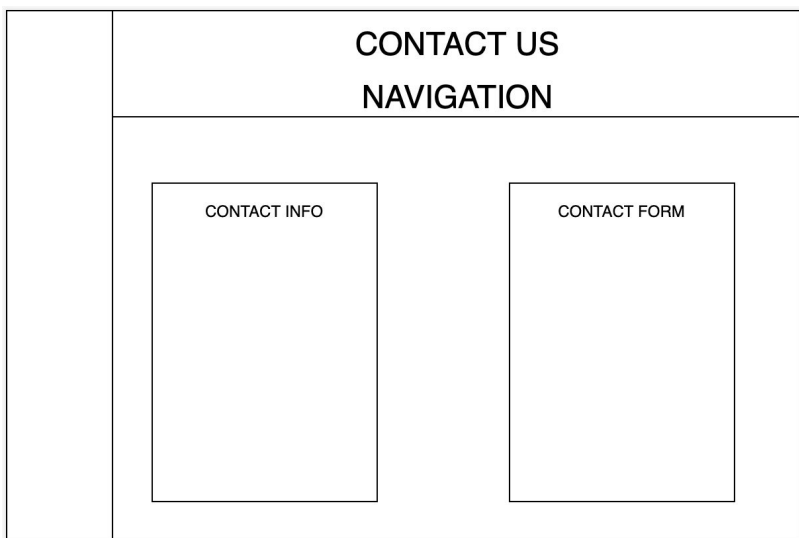
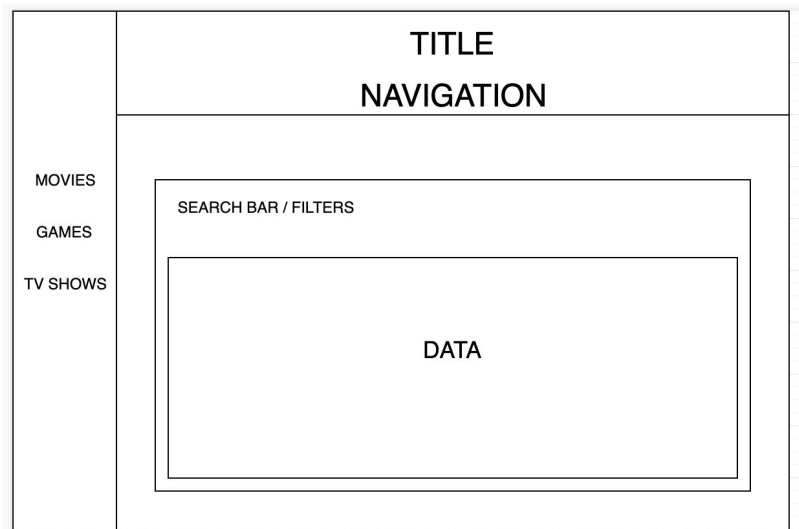
Type	Implementation Constraints	Solution
Hardware / Software	Has to run in a web browser on any computer	Use Ruby, Sinatran and PostgreSQL
Performance Requirements	App must run smoothly	Keep functionality simple

Persistent Storage and Transactions		
Usability	Must be clear and easy to use	Use simple page layout and clear buttons/text
Budget	None	
Time	1 Week	Project scope will be small
Usability	Must be usable for Visually Impaired	Page design must be simple, use a readable font, Make buttons big
Hosting Website	No budget	Host on free service like heroku

Unit	Ref	Evidence	
P	P.5	User Site Map	
		Description: Site map of rental shop project	



Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	
		Description: First diagram is a wireframe of the main page, Second diagram is of the contact us page	



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

Connect to the database
 Prepare the sql statement
 Run the prepared statement
 close database
 Return results

Unit	Ref	Evidence	
P	P.13	<p>Show user input being processed according to design requirements. Take a screenshot of:</p> <ul style="list-style-type: none"> * The user inputting something into your program * The user input being saved or used in some way 	
		Description: The user inputs information for a new film being added to the stock system. The second screenshot shows the added stock item.	

Paste Screenshot here

	Predator	11	12	VIEW
---	----------	----	----	----------------------

Unit	Ref	Evidence	
P	P.14	<p>Show an interaction with data persistence. Take a screenshot of:</p> <ul style="list-style-type: none"> * Data being inputted into your program * Confirmation of the data being saved 	
		Description: First screenshot shows user inputting information for a new film to be added to the stock system. The second screenshot shows the information in the database.	

RENTAL SHOP

NEW STOCK

stocks

customers

rentals

Name: Predator

Stock Left: 12

Price: 10

Image Source: /images/predator.jpg

Select Type: Game

```
[rentals=# SELECT * FROM stocks;
```

id	name	amount	price	image	type
3	Die Hard	2	12	/images/diehard.jpeg	Film
4	Counter Strike	24	18	/images/cs.jpeg	Game
5	Game of Thrones	16	18	/images/GoT.jpg	TV-Show
7	Predator	11	12	/images/predator.jpg	Film
2	Life Aquatic	7	12	/images/LifeAquatic.jpg	Film

(5 rows)

Unit	Ref	Evidence	
P	P.15	<p>Show the correct output of results and feedback to user. Take a screenshot of:</p> <ul style="list-style-type: none"> * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program 	
		Description: User inputs a search value to find a film. Then the film is displayed below.	

STOCKS

NEW STOCK

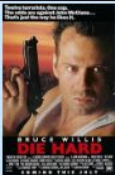

Die Hard

Q

Filter by category

Game

FILTER

	Name
	Die Hard
	

STOCKS

NEW STOCK


Search

Q

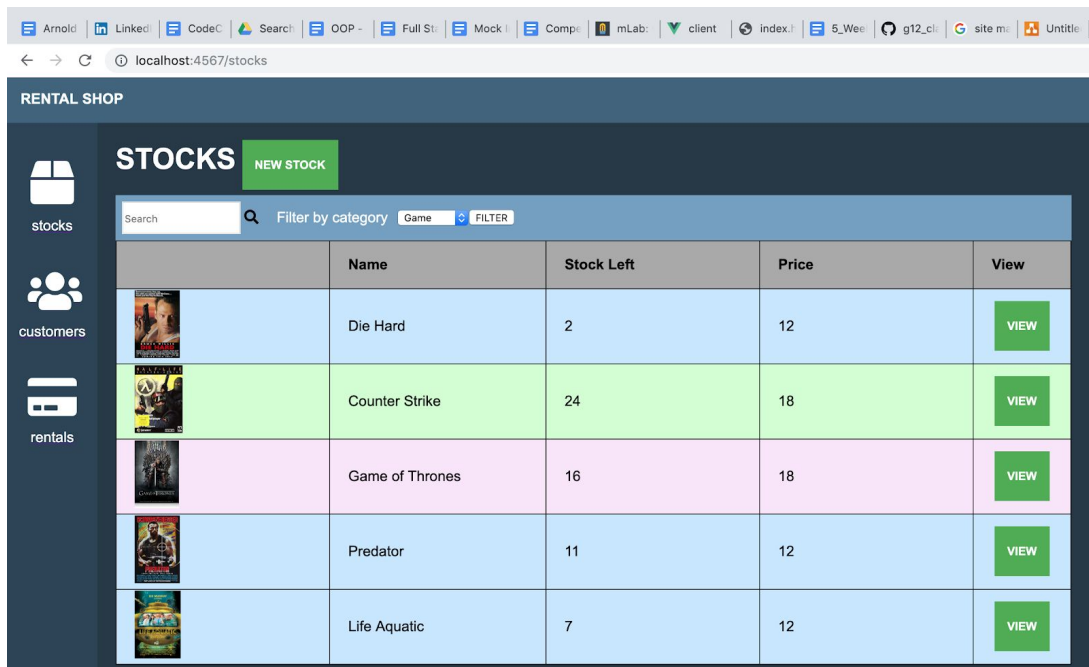
Filter by category

Game

FILTER

	Name	Sto
	Die Hard	2

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



<https://github.com/CalumCannon/MVC-rental-shop-project>

Week 7

Unit	Ref	Evidence	
P	P.16	Show an API being used within your program. Take a screenshot of: <ul style="list-style-type: none"> * The code that uses or implements the API * The API being used by the program whilst running 	
		Description: Using the punkipa API, displays a list of beers	

```

5  <script>
6  import BeerList from '@components/BeerList';
7
8
9  export default {
10     components: { BeerList },
11     data() {
12         return {
13             beers: []
14         };
15     },
16     mounted(){
17         fetch('https://api.punkapi.com/v2/beers')
18             .then(response => response.json())
19             .then(beers => this.beers = beers);
20     },
21

```

[View All Beers](#)

My List of Beers

[Buzz](#)

[Trashy Blonde](#)

[Berliner Weisse With Yuzu - B-Sides](#)

[Pilsen Lager](#)

[Avery Brown Dredge](#)

[Electric India](#)

[AB:12](#)

[Fake Lager](#)

[AB:07](#)

[Bramling X](#)

[Misspent Youth](#)

[Arcade Nation](#)

[Movember](#)

[Alpha Dog](#)

Unit	Ref	Evidence	
------	-----	----------	--

P	P.18	Demonstrate testing in your program. Take screenshots of: <ul style="list-style-type: none"> * Example of test code- * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing
		Description:

Paste Screenshot here

```
@Test
public void canDealOneCard(){
    deck.dealOne();
    assertEquals( expected: 60, deck.getCardCount());
}
```

```
java.lang.AssertionError:
Expected :60
Actual   :51
<Click to see difference>

<1 internal call>
at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal calls>
at DeckTest.canDealOneCard(DeckTest.java:46) <26 internal calls>
```

```
@Test
public void canDealOneCard(){
    deck.dealOne();
    assertEquals( expected: 51, deck.getCardCount());
}
```

```

✓ DeckTest 6 ms "E:\Program Files\Java\jdk1.8.0_25\bin\java.exe" ..
✓ canDealOneCard 6 ms
Process finished with exit code 0
|
```

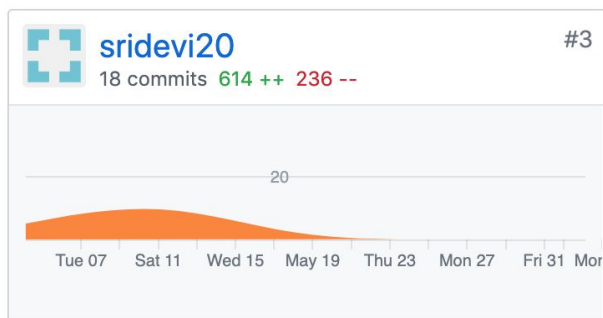
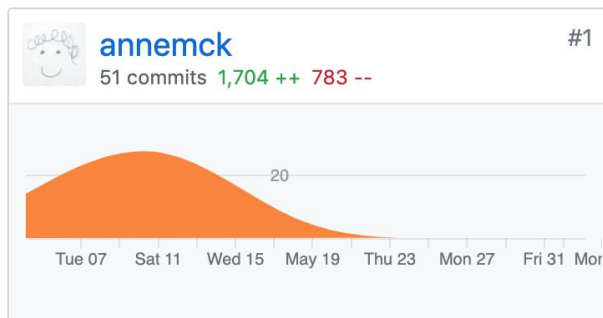
Week 9

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.	
		Description:	

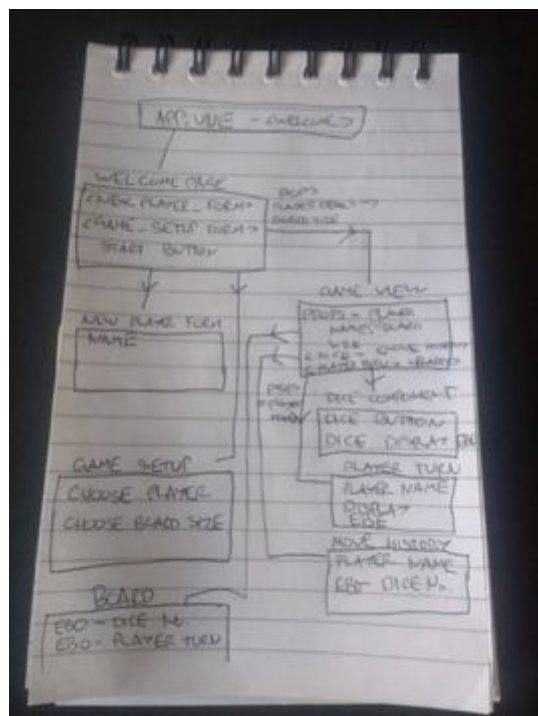
May 5, 2019 – Jun 3, 2019

Contributions: Commits ▾

Contributions to master, excluding merge commits



Unit	Ref	Evidence	
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.	
		Description: First image is a wireframe, second is an activity diagram.	





Unit	Ref	Evidence	
P	P.2	Take a screenshot of the project brief from your group project.	
		Description: Snakes And Ladders	

Snakes & Ladders Dice Game

You have been tasked to create a web-based app for people to play Snakes & Ladders against friends.

MVP

A player should be able to:

- Roll a dice
- Move on a game board
- Create a user profile
- Select their profile
- Win or loose the game

The game should:

- Be able to have more than one player
- Snakes should move a player down the board
- Ladders should move a player up the board

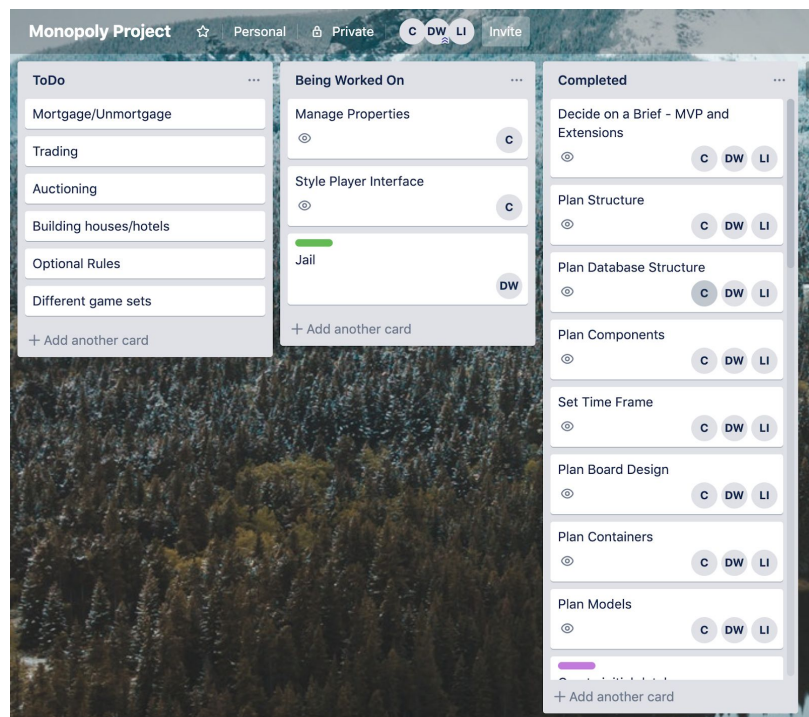
Extensions

- A player can view the leaderboard/their wins & loses
- Players can select the number of players in the game, and those players profiles

Advanced Extensions

A player can choose the size of the game board, small, medium or large

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

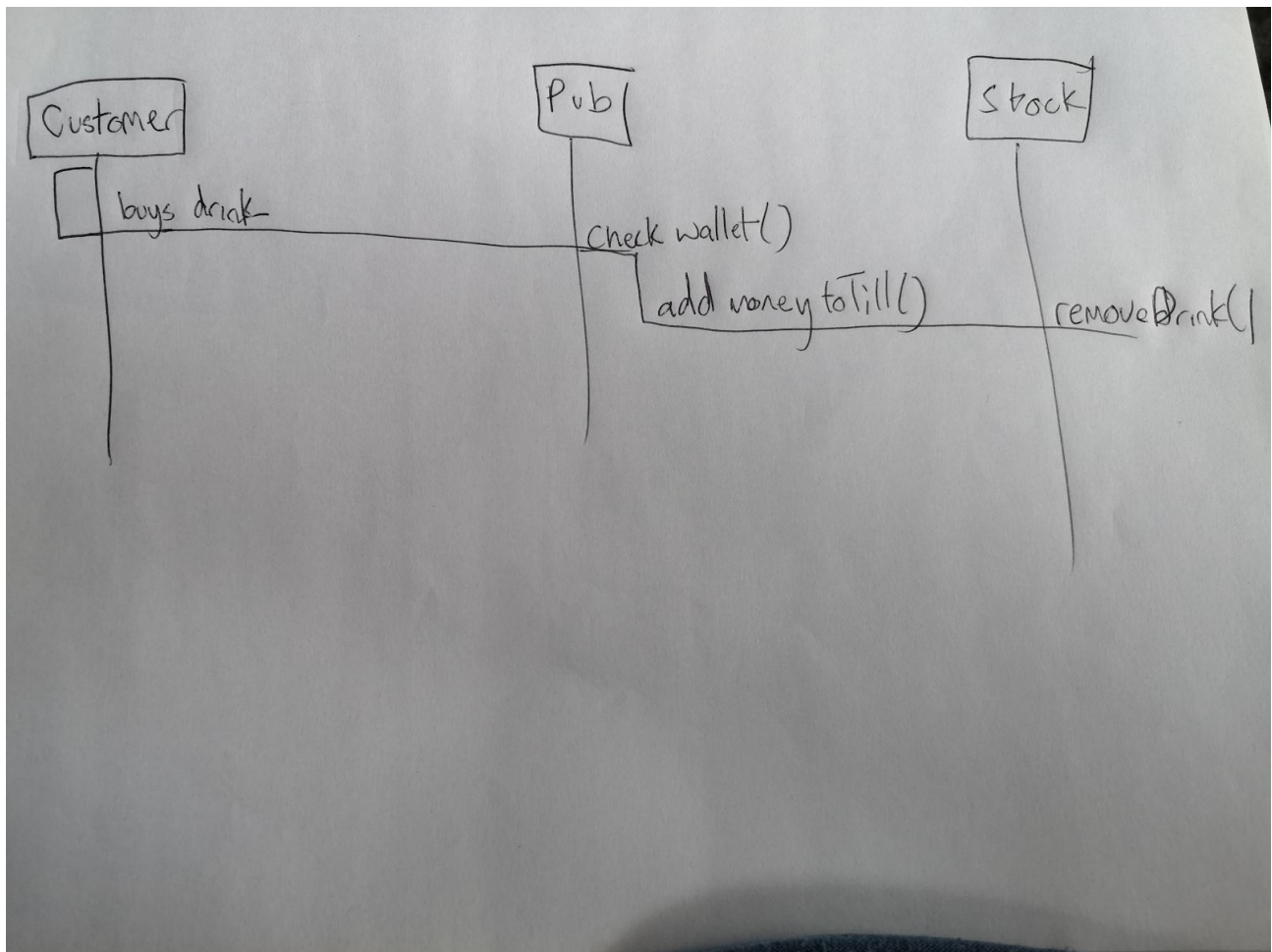


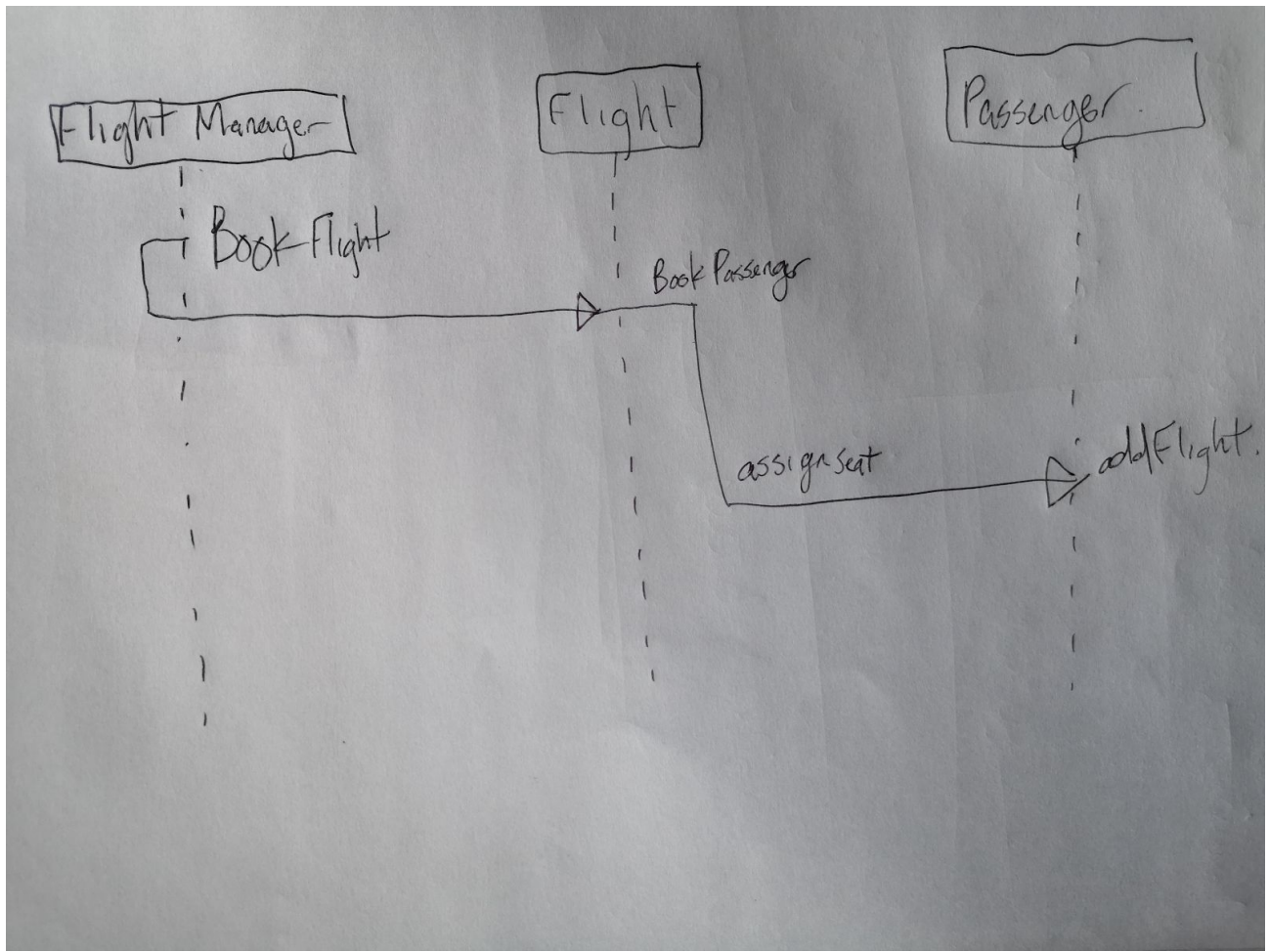
Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	

<u>Criteria</u>	<u>Solution</u>	<u>Pass/Fail</u>
Must be able to roll dice	Implement dice component with random number generator	PASS
Players must be able to move around the board	Use canvas to render players position and animate movement	PASS
Players must be able to win or lose	Once only one player is left end game	FAIL

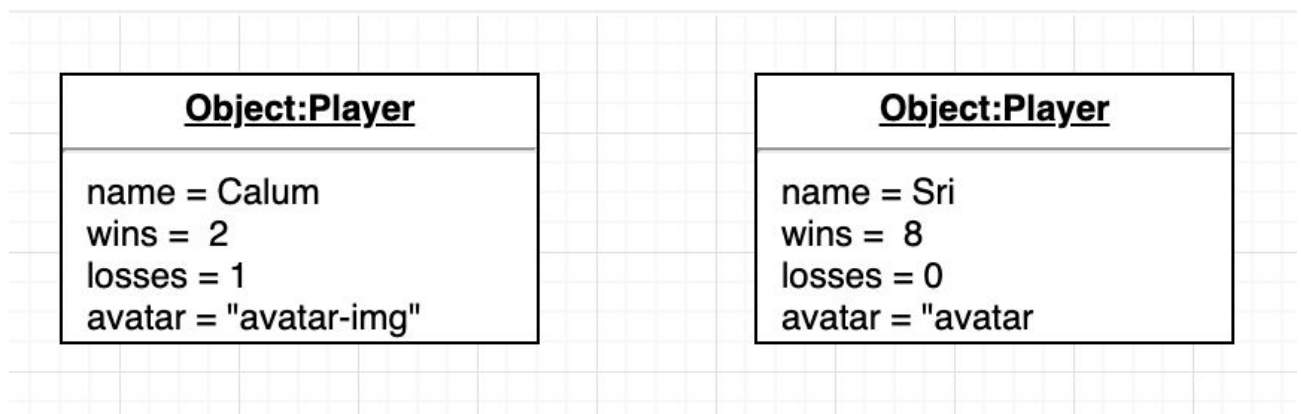
Unit	Ref	Evidence	
------	-----	----------	--

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





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



Unit	Ref	Evidence	
------	-----	----------	--

P	P.17	Produce a bug tracking report
		Description: Bugs from snakes and ladders group project

<u>Bug</u>	<u>Solution</u>	<u>Date</u>
Dice does not display number 1	Typo in image url	19/05/19
Player turn prompt displaying at wrong time	Move the call playerTurn prompt to appropriate place	13/05/19
Player1 is undefined until after first roll	Send player details to player turn component as props when game starts	11/05/19

Week 12

Unit	Ref	Evidence	
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.	
		Description: Using polymorphism to add multiple different types of objects into vehicles/soldiers arrays.	

```

public void addUnit(IVehicle vehicle){
    this.units.add((Unit)vehicle);
    this.vehicles.add(vehicle);
}

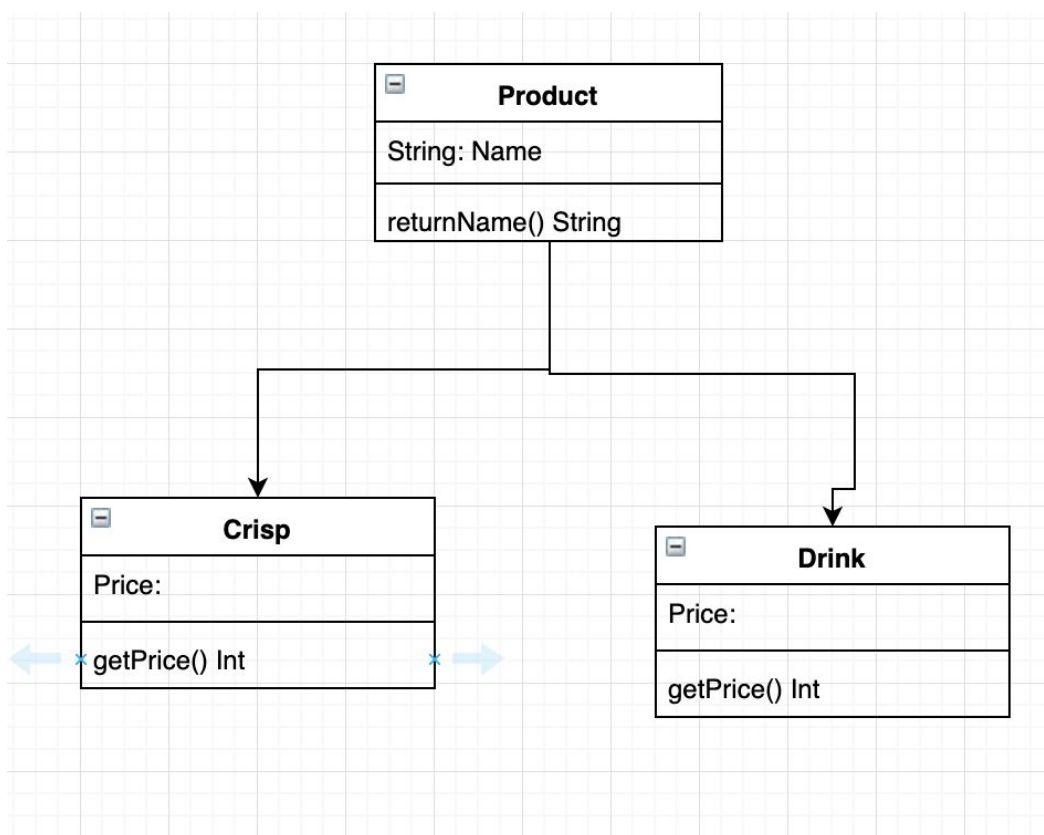
```

```

public void addUnit(ISoldier unit) {
    this.units.add((Unit)unit);
    this.soldiers.add(unit);
}

```

Unit	Ref	Evidence	
A&D	A.D.5	An Inheritance Diagram	
		Description: Drink and Crisps inherits from product.	



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

```
import java.util.ArrayList;

public class Player {

    private ArrayList<Card> hand;

    public Player(){
        hand = new ArrayList<Card>();
    }

    public ArrayList<Card> returnHand() { return hand; }

    public void addCardToHand(Card card) { this.hand.add(card); }

    public int handSize() { return hand.size(); }
```

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.	
		Description:	

```

public class Person {

    private String name;
    private String cohort;

    public Person(String name, String cohort){
        this.name = name;
        this.cohort = cohort;
    }

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

    public String getCohort() { return this.cohort; }

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

    public String talk(String language) { return "I love " + language; }

    public void setCohort(String cohort) { this.cohort = cohort; }

}

```

```

public class Student extends Person{

    public Student(String name, String cohort) { super(name, cohort); }

    @Override
    public String talk(String language) { return "I love learning " + language; }

}

```

```

@Test
public void canChangeName() {
    person.setName("Jim");
    assertEquals( expected: "Jim", person.getName());
}

```

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.	

		Description: calculate function checks to see if list array contains rock, scissors or paper then returns the appropriate answer. distanceCheck function returns the distance between two points
--	--	--

```
class Game

  def initialize(first, second)
    @first = first
    @second = second
  end

  def calculate
    list = [@first, @second]
    return "Paper Wins!" if(list.include?("rock") && list.include?("paper"))
    return "Rock Wins!" if(list.include?("rock") && list.include?("scissors"))
    return "Scissors Wins!" if(list.include?("scissors") && list.include?("paper"))
    return "ERROR"
  end
end
```

```
distanceCheck(player, tx, ty){
  var dist = Math.sqrt( Math.pow((player.xpos-tx), 2) + Math.pow((player.ypos-ty), 2) );
  if(dist < 50){
    return true;
  }
  console.log(dist);
  return false;
},
```