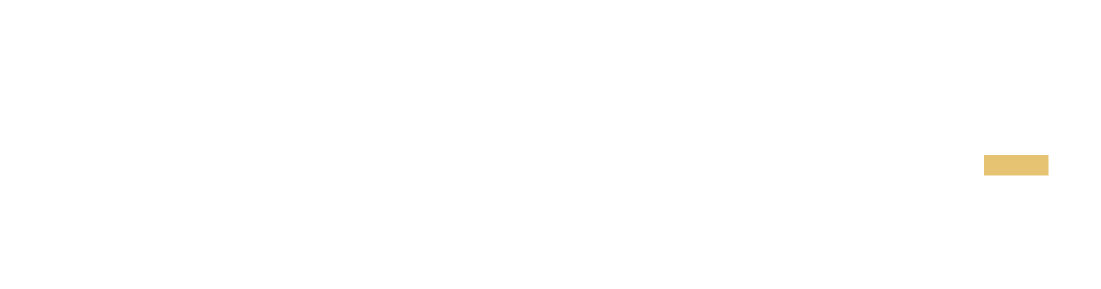
Evidence Gathering Document for SQA Level 8 Professional Developer Award.

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

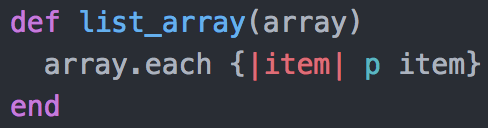
Please fill in each point with screenshot or diagram and description of what you are showing.

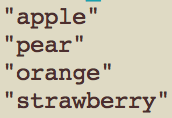


Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

**Week 2**

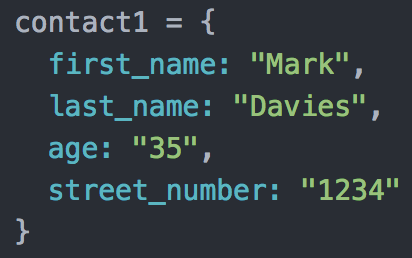
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.5 | Demonstrate the use of an array in a program. Take screenshots of:  \*An array in a program  \*A function that uses the array  \*The result of the function running | |
|  |  | **Description:** | |

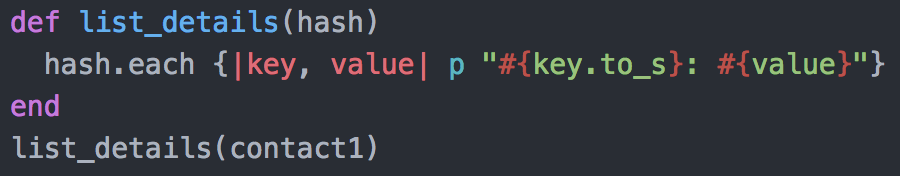
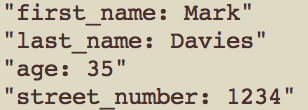
****

****

This is a simple function which takes an array, in this case fruits, and ‘puts’ out each item in turn.

| 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:** | |

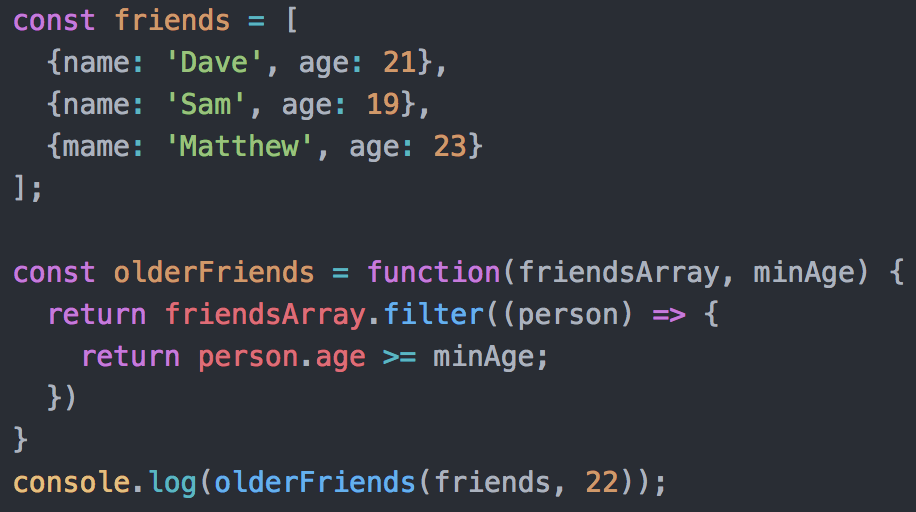
****

****

contact1 is a hash including someone’s personal details. The list\_details function prints each field of the hash in a structured sentence.

**Week 3**

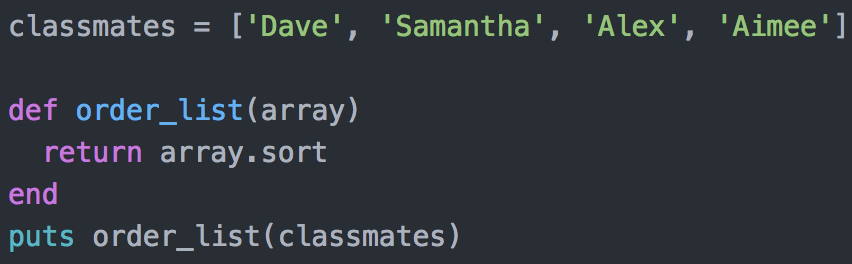
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.3 | Demonstrate searching data in a program. Take screenshots of:  \*Function that searches data  \*The result of the function running | |
|  |  | **Description:** | |

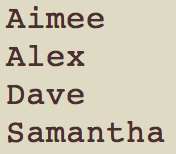


../../Desktop/Screen%20Shot%202018-12-26%20at%2016.59.17.

The above code, written in javascript, demonstrates a function searching through some data and filtering it dependant on the criteria given: *friends* is an array of objects which each have name and age properties. The function *olderFriends* takes an array of friends and returns an array of those from the original array which or at least a given age. The second image shows the result of calling the function with a minimum age of 22

| 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:** | |

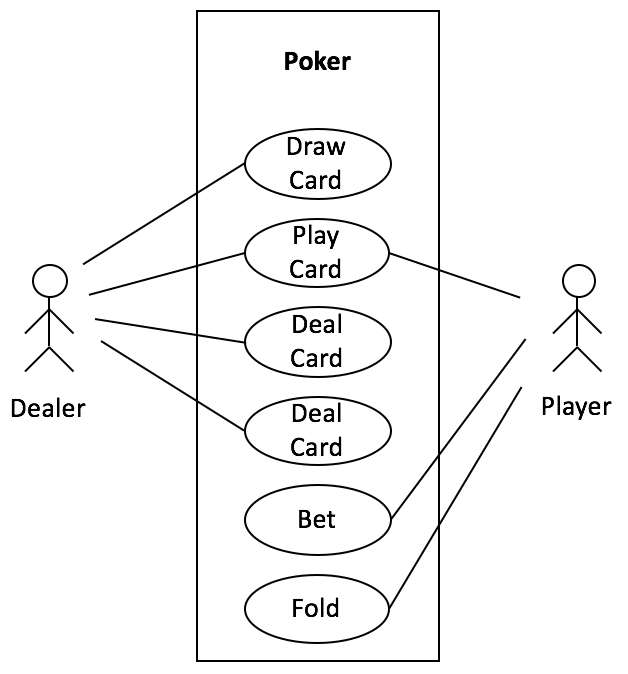




The above code is written in Ruby. The first image shows the creation of the array *classmates* which contains four in no particular order. The function *order\_list* takes an array and returns an array of the same length, sorted alphabetically. The result of calling *order\_list* on *classmates* is shown in the second image.

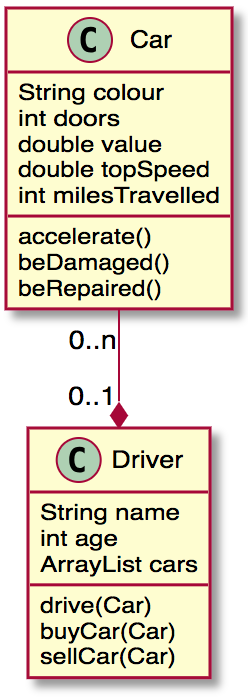
**Week 5 and 6**

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| A&D | A.D.1 | A Use Case Diagram | |
|  |  | **Description:** | |



This diagram shows the how a dealer and player interact with a poker system. Both users share the play card method, but have their own unique methods beyond that.

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| A&D | A.D.2 | A Class Diagram | |
|  |  | **Description:** | |



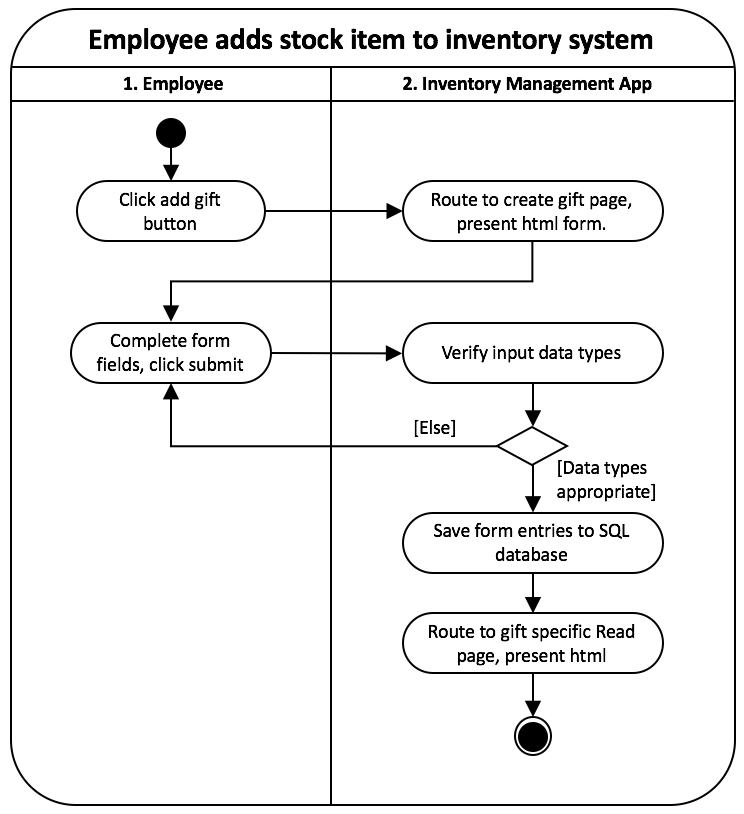
This class diagram shows two classes: A car class, composed of 5 instance variables and three methods. A Driver which is composed of three instance variables (including an array of car objects) and 3 methods.

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| A&D | A.D.3 | An Object Diagram | |
|  |  | **Description:** | |



This object diagram is the object version of the class diagram provided in the previous example. car1 is an object with 5 known instance variables, contained within an array in driver1, which has two other instance variables.

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| A&D | A.D.4 | An Activity Diagram | |
|  |  | **Description:** | |



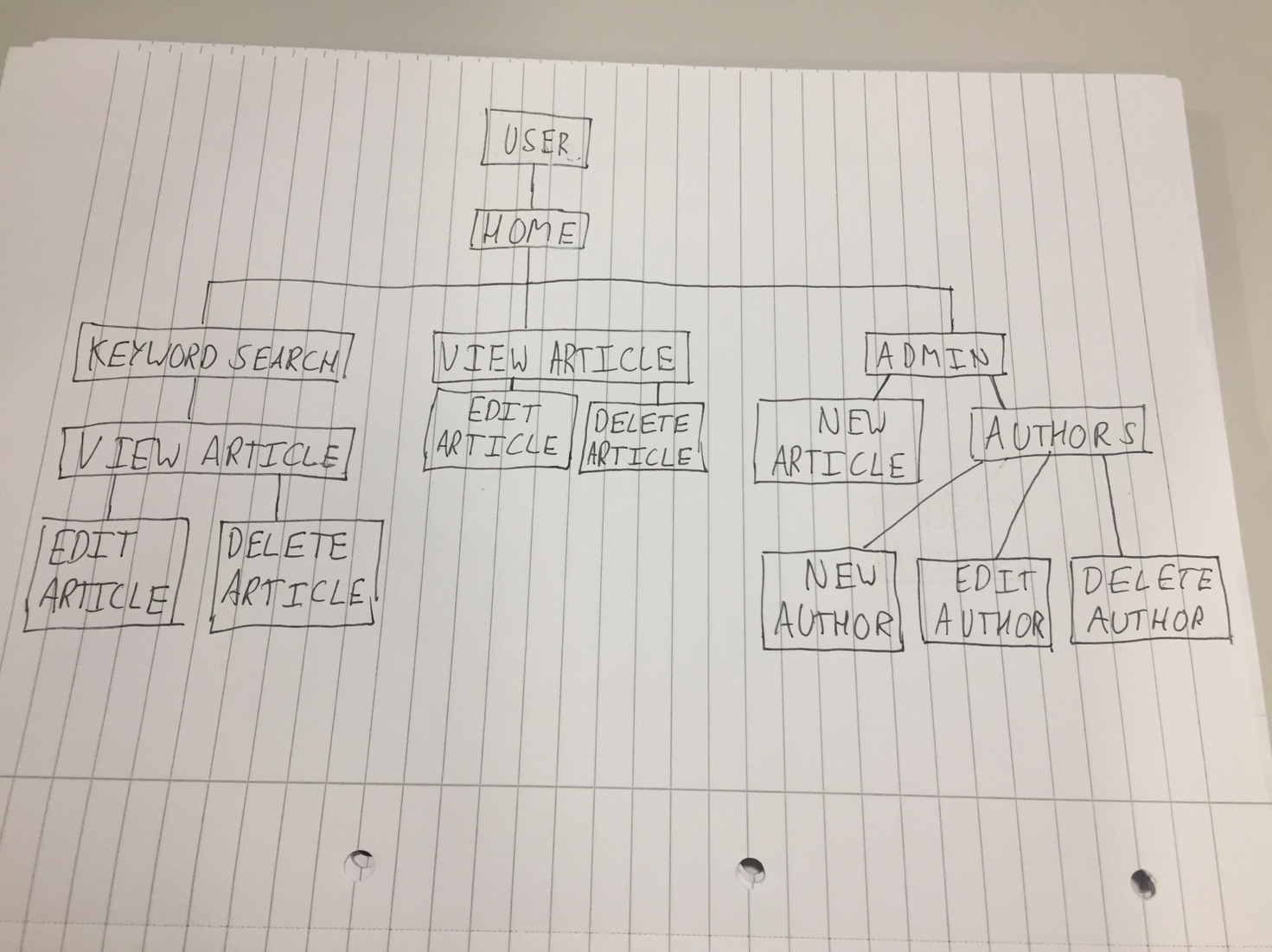
This is an activity diagram from a project where I made an inventory management system for a gift store. The activity being described is an employee (user) adding a new item to the stock list, which was previously not stocked by the store. From the home page, the user clicks the add item button. The app then directs them to the form where they can create a new item. Once the user has completed all the fields, they hit the submit button. The app then checks that each piece of information is of the appropriate data type. If they are, it saves the item to the shop database and redirects the user to the item specific page. If one or more field contains inappropriate data, the form does not submit, and the user is directed to fill the form out properly.

| 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:** | |

|  |  |  |
| --- | --- | --- |
| Constraint Category | Implementation Constraint | Solution |
| Hardware and Software Platforms | The site is to be built using mac OS, meaning any developer which wishes to contribute to the project will need to use a mac, potentially limiting the skill pool. | Throughout the project, all viable developers will have a mac made available to them. |
| Performance Requirements | An API is to be built as a component of the project. That will then need to be hosted on local machines by developers as they make the front-end of the application. If their machines are not capable of hosting the server reliably, mistakes in the construction of the front-end should be expected. | Developers should terminate other services being hosted on their local machines so as to maximise the performance of the purpose built API and minimise build errors. |
| Persistent Storage and Transactions | Local data could be lost or corrupted during development, impacting delivery time. | Using GitHub at regular intervals will mitigate the loss of progress by serving as a backup. |
| Usability | Staff who are unfamiliar with the layout of the site and its features may be unable to make effective use of it. | By conducting interviews and user testing with staff members who would be expected to use it, we can make it as ergonomic as possible, so existing and future staff can use it with minimal training. |
| Budgets | The budget for this project is very small, limiting the amount of specialist tools and expertise that can be employed. | As some of this project will take place over the weekend, premises for collaborative work will need to be rented. To limit expenditure in other areas only open-source technologies will be used, and a free hosting service, such as Heroku. |
| Time Limitations | Only 6 days are available for the completion of this project. This includes not only the development of the web-app, but the preparation and rehearsal of its presentation. | Agile practices will be used to make the most of this 6 day sprint period. A clear separation of features in to MVP and extensions at the start of the 6 days should make sure the most vital elements of the site are developed first. |

The above implementation constraints were considered for a group project during which we made a news web-app intended to be used by CodeClan staff.

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.5 | User Site Map | |
|  |  | **Description:** | |



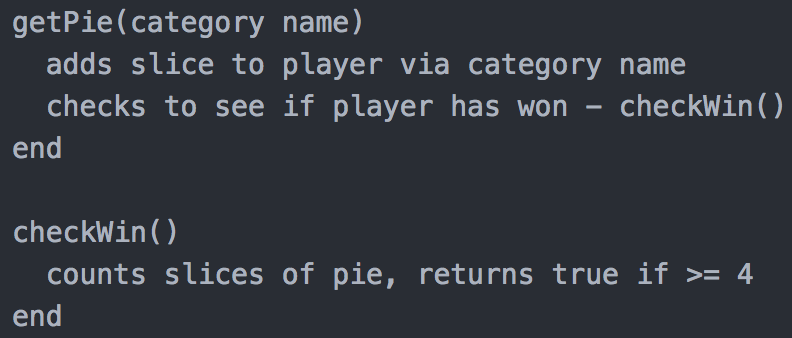
The above site map is for the news web-app developed during a group project. The site hosts news articles, each written by authors. From the home page users can open whole articles, search for articles using a keyword text bar or navigate to the admin screen. From within an article, admins can edit or delete the article. From the admin screen, admins can create new articles and create, edit or delete authors.

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.6 | 2 Wireframe Diagrams | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**

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



These two pseudocode functions are from a trivial pursuit emulator, and are expected to be part of a class called ‘Player’. The getPie() method will be expected to take in a category name and add the slice of pie to the players collection based on that category name. Then it should check to see if the player has won the game using the checkWin() function. The checkWin() function should count the slices of pie in the players collection and return true or false, depending on if they have four or more slices of pie.

| 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 | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**

| 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 | |
|  |  | **Description:** | |

**Paste Screenshot here**

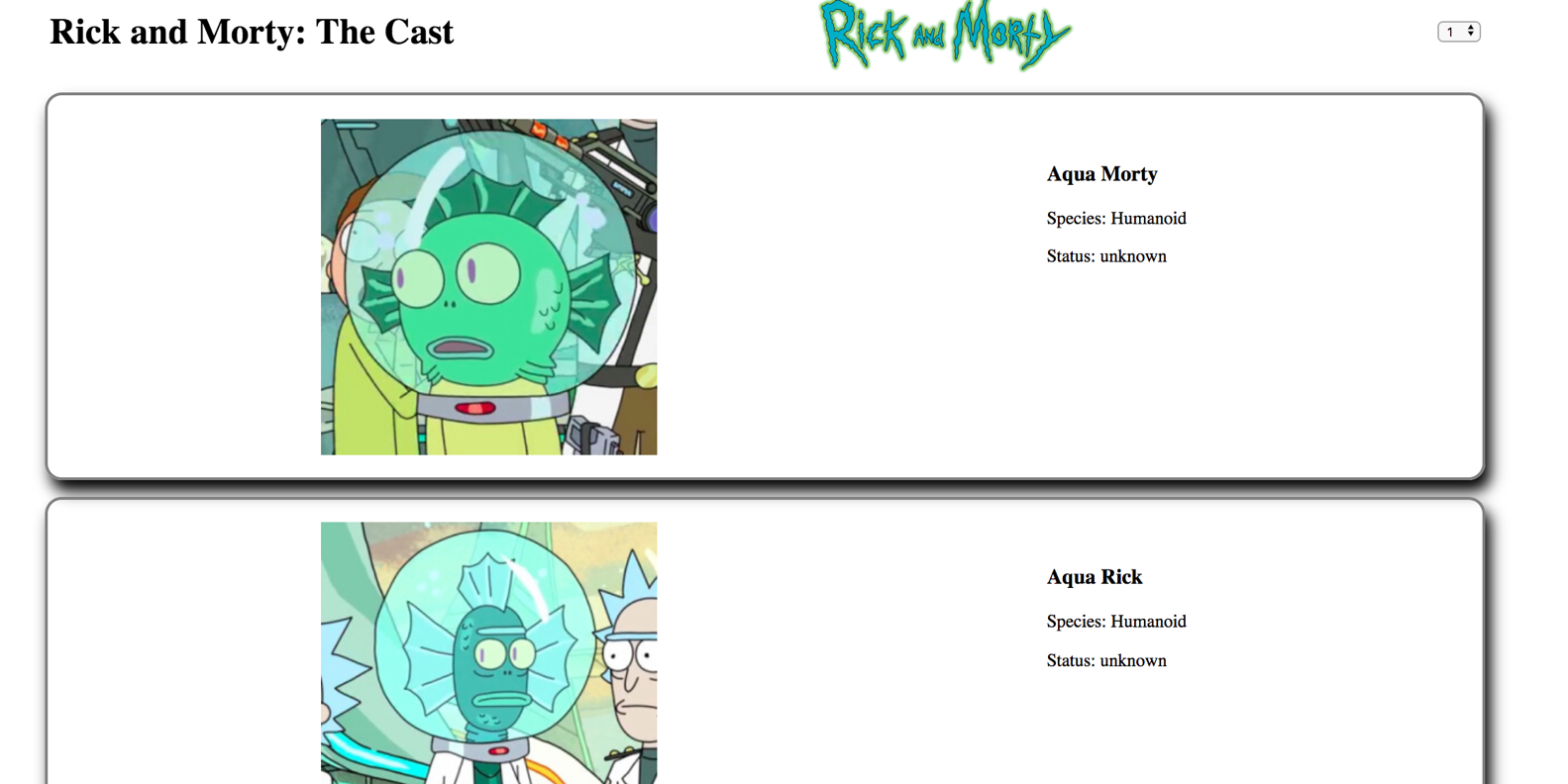
**Description here**

| 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 | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**

| 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:** | |



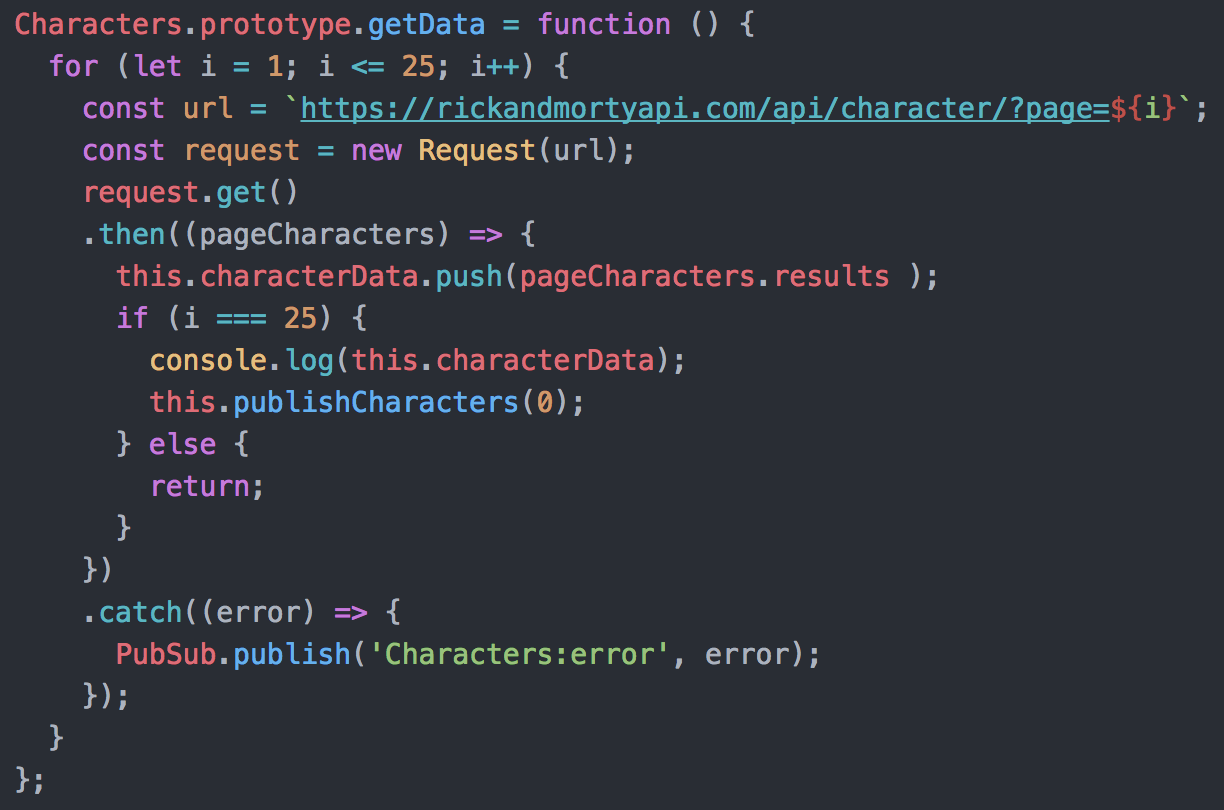
<https://github.com/robwilson195/Rick_n_Morty_The_Cast>

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

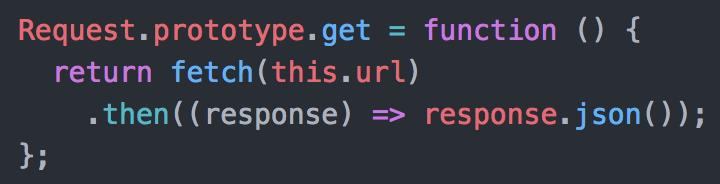
Does this have to be for the same project as P11?

**Week 7**

| 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 | |
|  |  | **Description:** | |

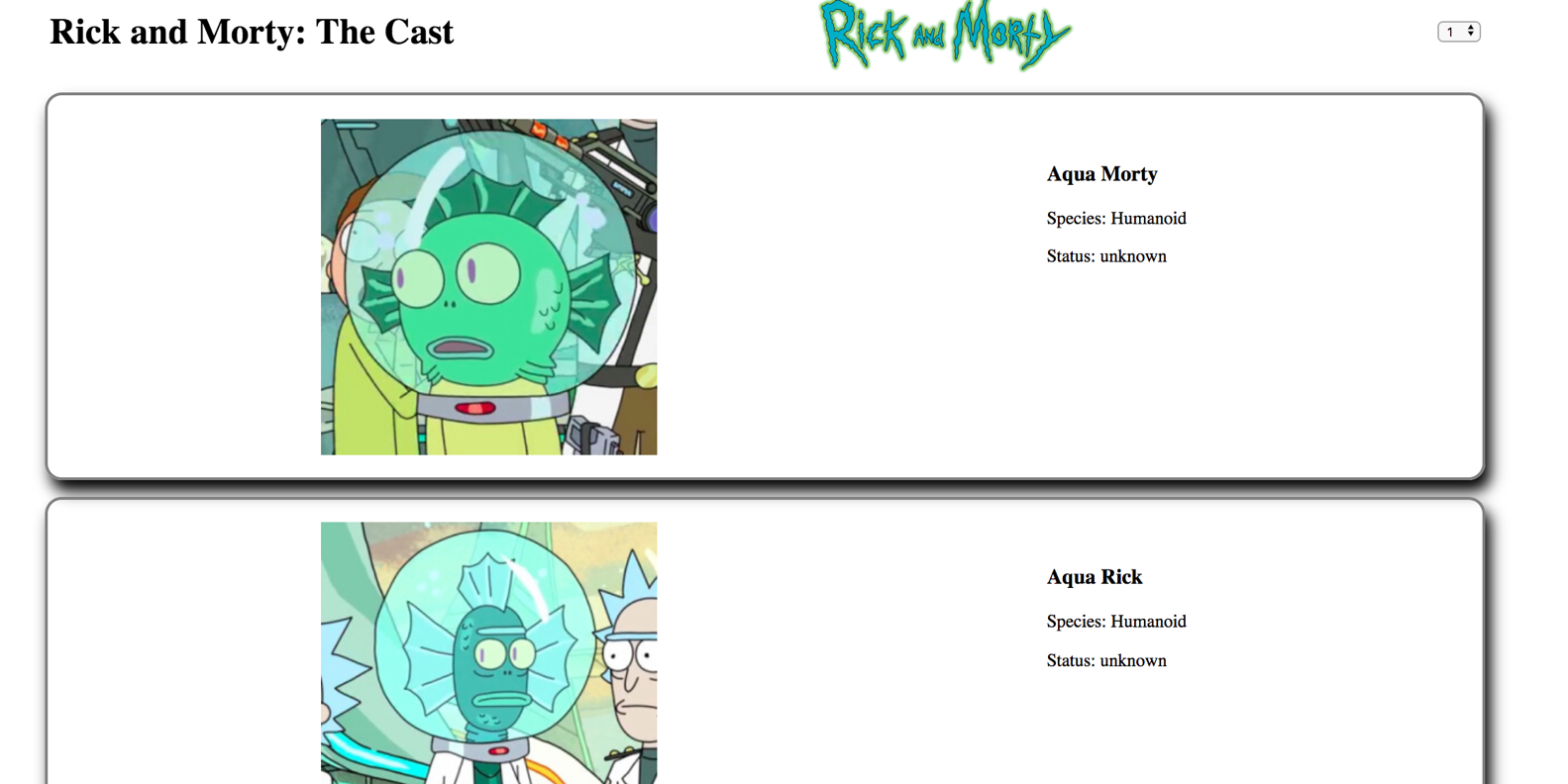
****

This first image shows a function (getData). It is responsible for making 25 requests to the Rick and Morty api, with the intention of returning 25 pages of character information. It creates a new request for each of the 25 urls and uses the get method, shown next:



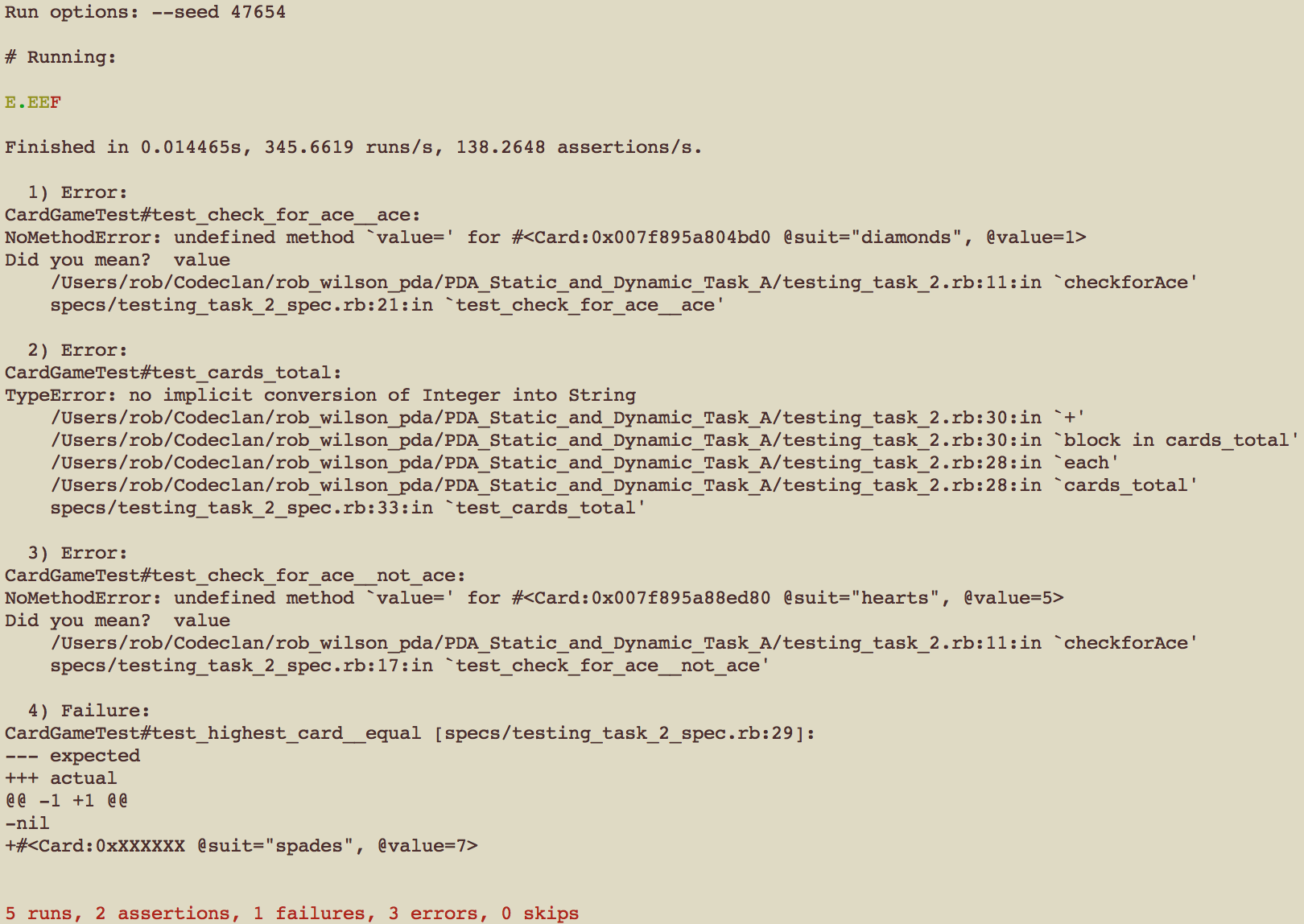
The get function uses fetch, targeting the url that was used to create it. The fetch request type defaults to ‘GET’. The response of the fetch is then sent back to the origin of the function.

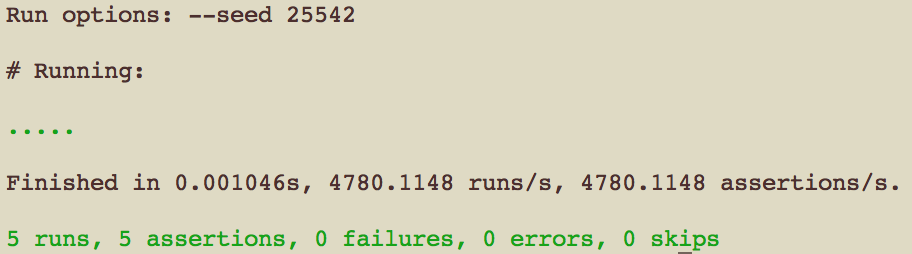
When the response is returned to the getData function in the first image, the data is pushed to the characters model under the characterData field, along with the other pages.

Once all the data has been gathered, the first page of characters is published to the browser as shown below:

| 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 | |
|  |  | **Description:** | |





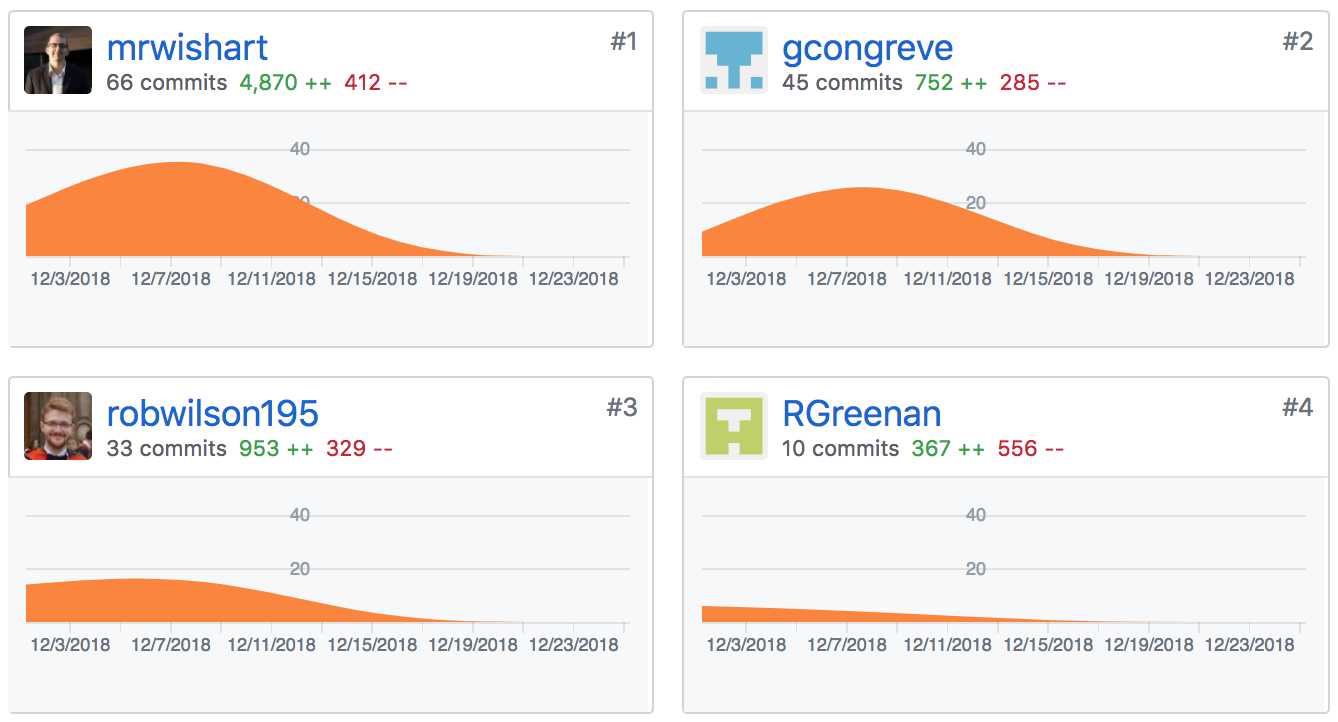


The first image is the test file, showing the MiniTest module in Ruby being used to test another file. The second image shows the 5 tests running before the tested code has been fully corrected. The third, once it has been altered and refactored to pass the 5 comprehensive tests.

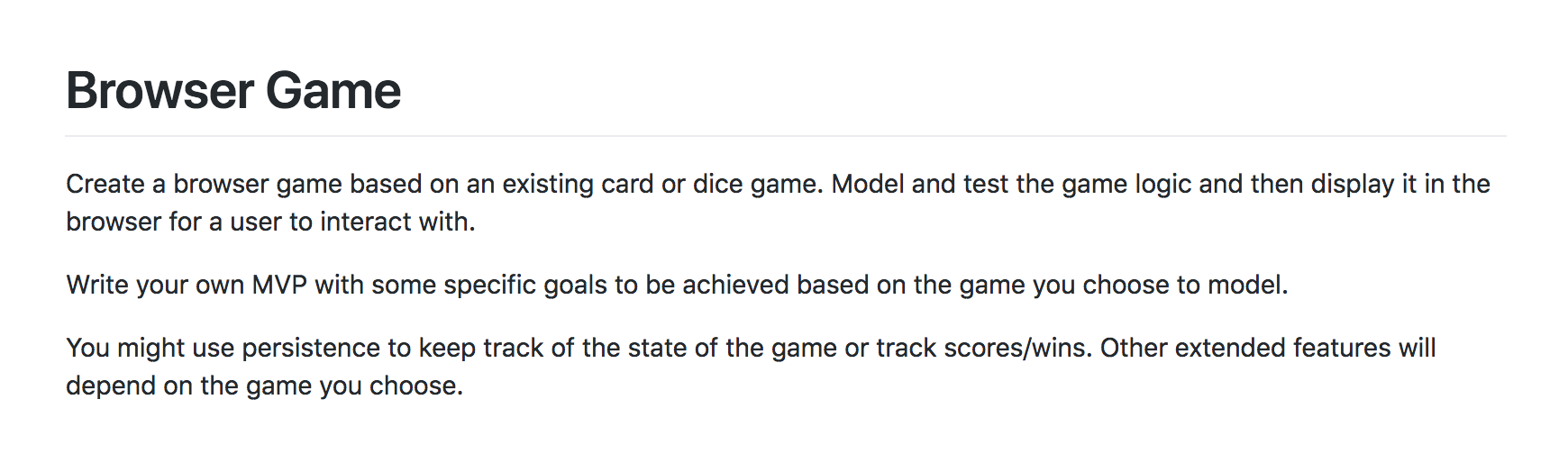
See the folder ‘PDA\_Static\_and\_Dynamic\_Task\_A’ within this repository for both the test code and the tested code.

**Week 9**

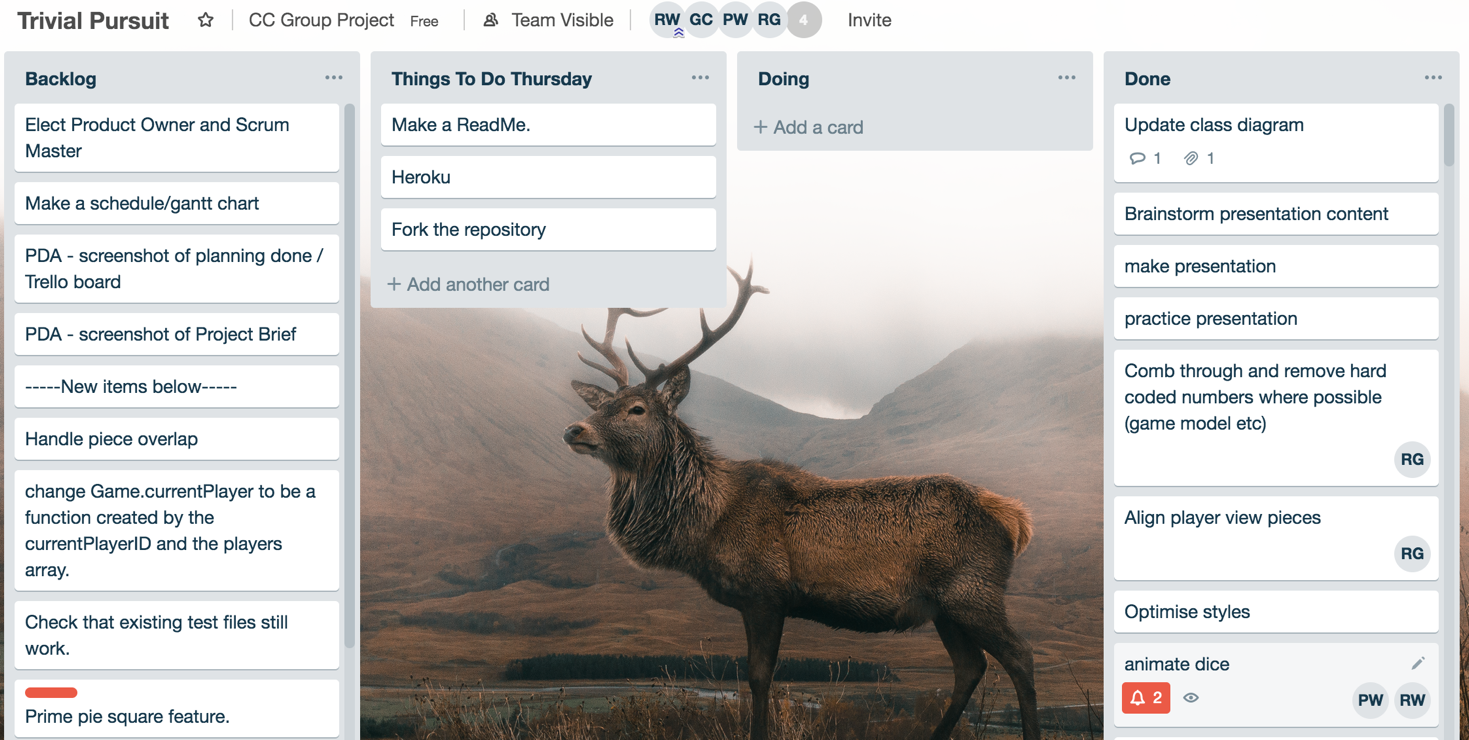
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.1 | Take a screenshot of the contributor’s page on Github from your group project to show the team you worked with. | |
|  |  | **Description:** | |



| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.2 | Take a screenshot of the project brief from your group project. | |
|  |  | **Description:** | |

****

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



This is a Kanban board from a group project I was part of when we made a trivial pursuit game online using javascript. We moved tasks to be done from the ‘backlog’ in to the ‘things to do’ column for a specific sprint. Then, when someone was actively working on a feature, they would add their name to it and move it to the ‘doing’ column. Finally it would be moved to ‘done’ when it had been completed.

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

**Paste Screenshot here**

**Description here**

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.7 | Produce two system interaction diagrams (sequence and/or collaboration diagrams). | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.17 | Produce a bug tracking report | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**

**Week 12**

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

**Paste Screenshot here**

**Description here**

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| A&D | A.D.5 | An Inheritance Diagram | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

| 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:** | |

**Paste Screenshot here**

**Description here**

| 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:** | |

**Paste Screenshot here**

**Description here**