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

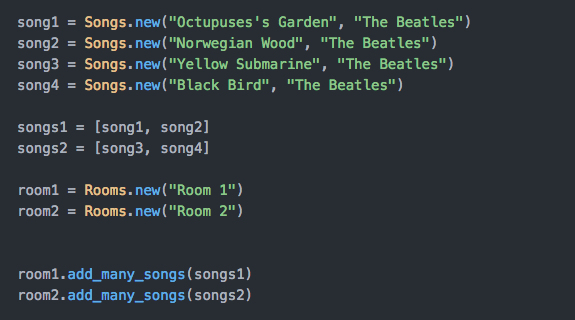
This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

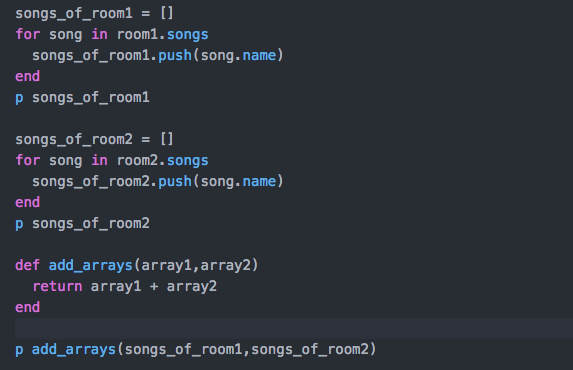
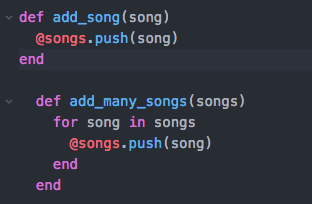
Each point that required details the Assessment Criteria (What you have to show) along with a brief description of the kind of things you should be showing.

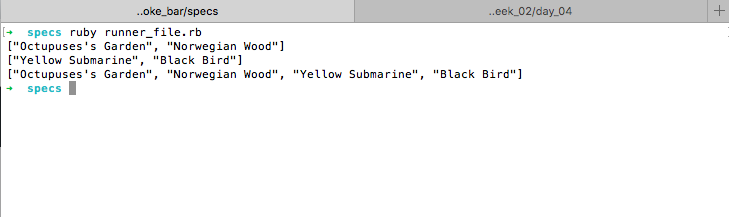
Please fill in each point with screenshot or diagram and description of what you are showing.

**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:** | |

**Paste Screenshot here**

****

****

**Description here**

Here I created a Ruby Class of Songs and Rooms that are part of a Karaoke. In the left superior corner I created four songs with a name and an artists, and then I created two arrays (songs1 , songs2) made out of the songs. In the Room class I created methods to add one individual song or an array of rooms through two methods. In the image at the right I looped through the songs found in each room to create a new array of just the names of the songs in the room. At the bottom I created a “add\_array” method that took in two arrays and combined them.

In the last image which displays the console, I have an array of the song’s names of the first room and then the second room and then an array of the names of the songs for bothrooms.

| 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:** | |

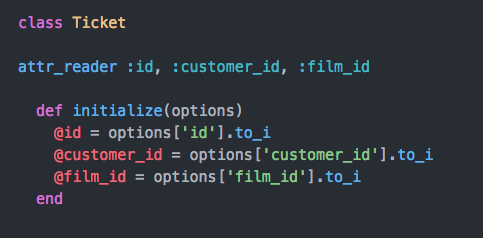
**Paste Screenshot here**

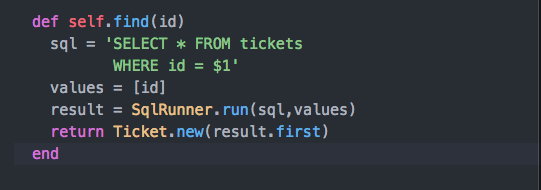
**Description here**

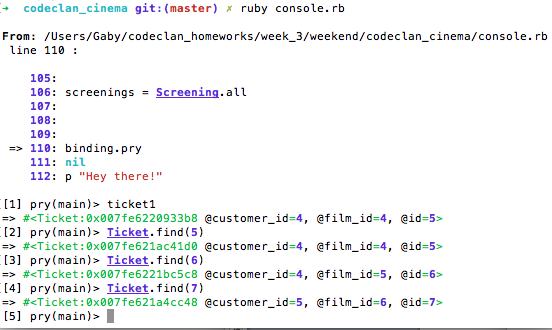
**Week 3**

| 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:** | |

**Paste Screenshot here**

****

****

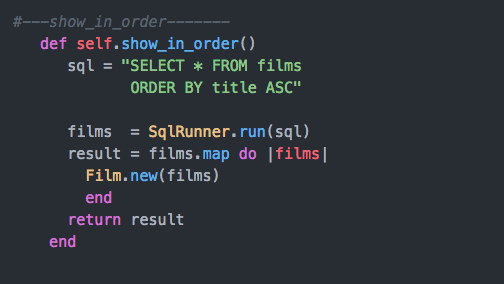
****

**Description here**

On week 3, we began to create SQL databases and write our code in Ruby. In the example above, there was a many to many relationship between the classes, where class Ticket stored the id of the customer and the film. I created a method in the Ticket class that aimed to find a specific ticket when it’s id was given as a parameter, which matched with my codeclan cinema database that I created for the project. The method takes the id given and returns an array of hashes as the ‘result’. Of this array of hashes I took the first and created a new Ticket object with it and then returned it. In the terminal, using pry, I can access the ticket1 that was created and then then through the find() methods I can display as with other tickets. As it has not real properties of it’s own but was dependent on the two other classes and the ids of the customers and the films.

| 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:** | |

**Paste Screenshot here**

****

****

**Description here**

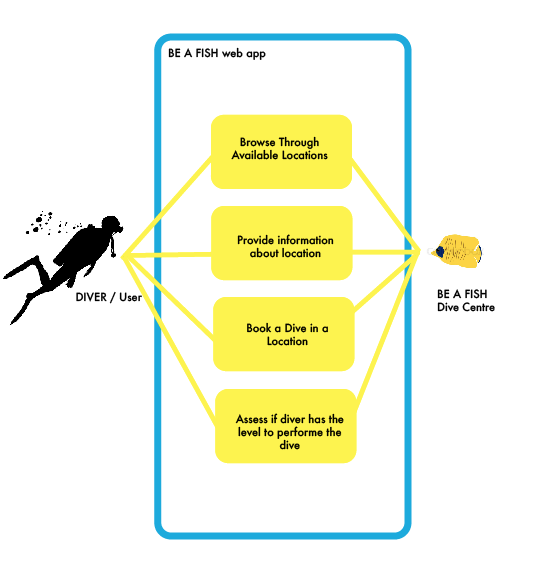
Following the project of the Codeclan Cinema, I created a method in the Film class that would take return an array of film objects arranged in alphabetical order. The method show\_in\_order()

Contrasts with self.all() because it uses SQL code to specify that the result should be in ORDER of the column that has the ‘Title’ values. This then again, returns an array of hashes, which I then map to create new Film objects. In the Terminal image I contrast both methods to see the effect that “ORDER BY title ASC” has on the order of the films.

**Week 5 and 6**

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

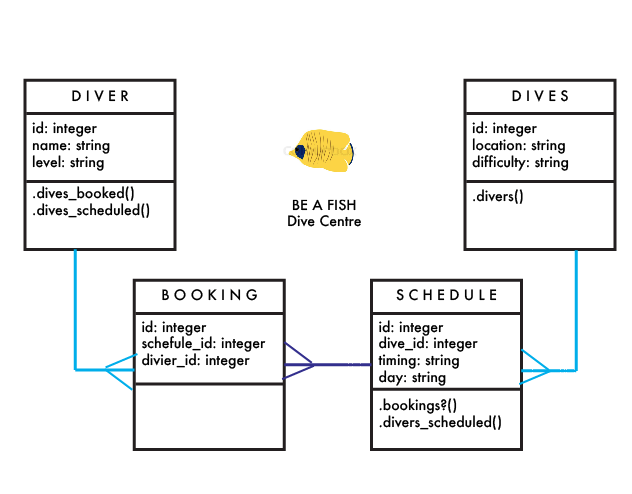
**Paste Screenshot here**

****

**Description here**

In the above image I depicted a use case diagram, which illustrates the most basic interactions among the elements of system, in this case the “Be A Fish” web application, and provides an overview of the usage requirements of the application. In this application there will give divers/users the possibility to view all the available diving locations that the dive center offers. Each location will also offer the difficulty of the dive and tell the schedule of the dive. The diver can book then a dive at a particular location for a particular time. The Dive shop can also see how many users are signed up for each scheduled dive in certain location. Update the list, delete divers, and update their information.

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

****

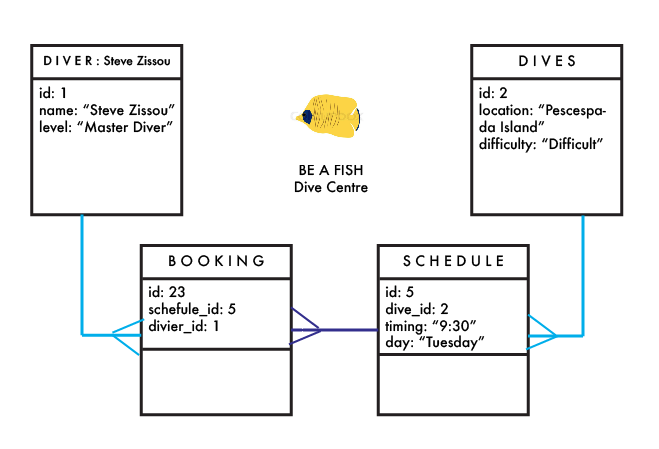
**Paste Screenshot here**

**Description here**

The diagram depicts a Class Diagram, where the structure of the web application Be a Fish is structured in different classes and the relationships between the classes as well as some methods included in each. In the web application there are four classes, *diver, divers*, *bookings* and *schedule.* Diver and booking is a one to many relationship, since 1 diver can have many bookings but a booking only has one diver, hence the @diver\_id. A scheduled *dive* can have many bookings so it’s a one to much relationship again. Finally, dives has a many to one relationship with schedule, since one dive/location can have many scheduled dives. A new diver, booking,schedulre,dives all can be creates, updated, deleted and views. Specifically to each classes are certain methods, a diver can checked this bookings, and scheduled dives. A scheduled dive can view which divers go in that dive and how many bookings it has. And a Dive can check how many divers are going and check if they are prepared for the difficulty of the dive.

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

**Paste Screenshot here**

****

**Description here**

Using and object diagram here, we can see that a diver (“Steve Zissou”) who is a “Master Diver” (the highest level of diver) has made a booking for the scheduled dive to the Dive site “Pescespada Island” which is a difficult dive site and is scheduled for Tuesday at 9:30.

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

**Paste Screenshot here**

**Description here**

| 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:** | |

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

| 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:** | |

**Paste Screenshot here**

**Description here**

| 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:** | |

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

**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:** | |

**Paste Screenshot here**

**Description here**

| 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:** | |

**Paste Screenshot here**

**Description here**

**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:** | |

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

| 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**