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.

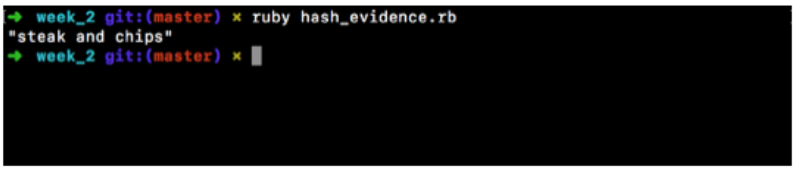
Please fill in each point with screenshot or diagram and description.

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

This screenshot shows an array of songs, with a function that adds a song to the array and the test for this, with a screenshot of the test passing in the terminal.

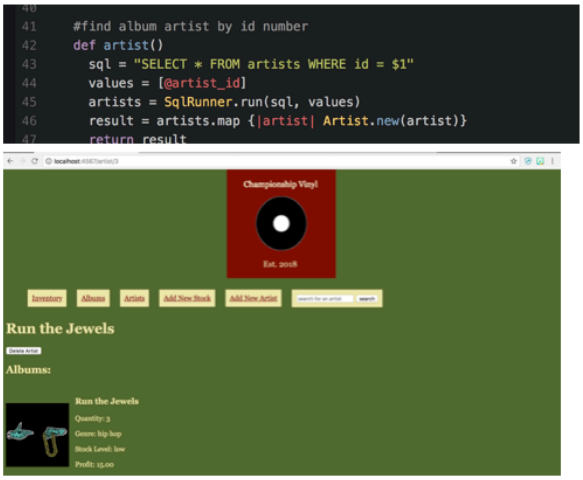
Unit	Ref	Evidence
I&T	I.T.6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running

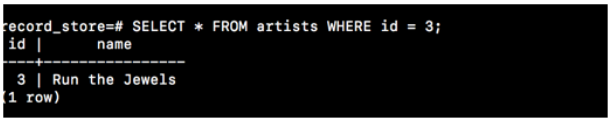


This screenshot shows the use of a hash, with a function accessing the keys of “Sarah” and “main” to print the value of this “Steak and Chips” to the terminal.

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





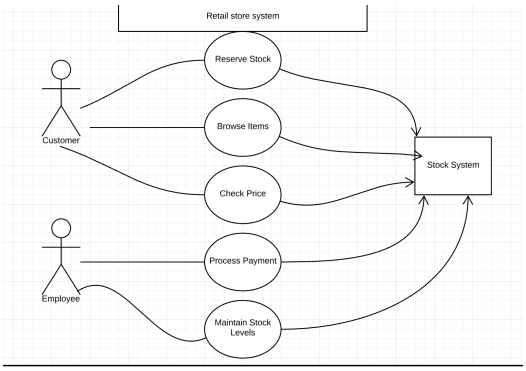
This screenshot shows a function that searches for all music artists in a database within a program by id number and the result of using the search in psql and a web app made using Sinatra.

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

This screenshot shows a function that searches for all musical artists in a database and sorts them in alphabetical order by name, with the results in psql and a web app.

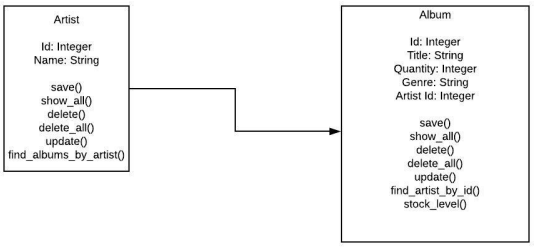
Week 5

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



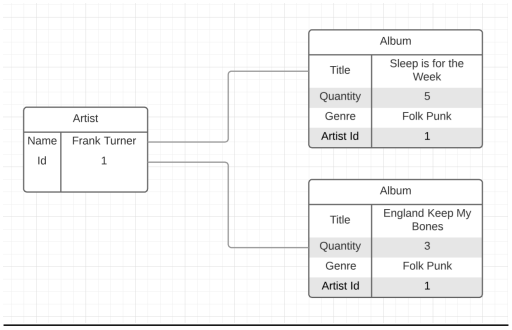
This is an example of a use case diagram for a retail store system that allows both staff and customers to interact with it.

Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram



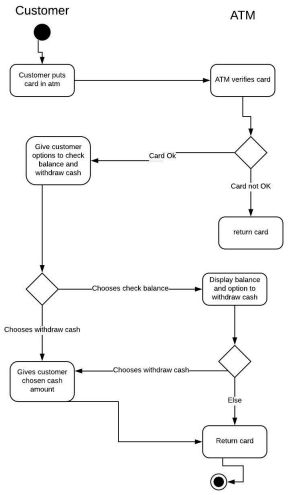
This is an example of a class diagram for a program which has two classes - one of Artists and one of Albums, with different properties and methods in each.

Unit	Ref	Evidence
A&D	A.D.3	An Object Diagram



This is an example of an object diagram in which one musical artist has many albums.

Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram



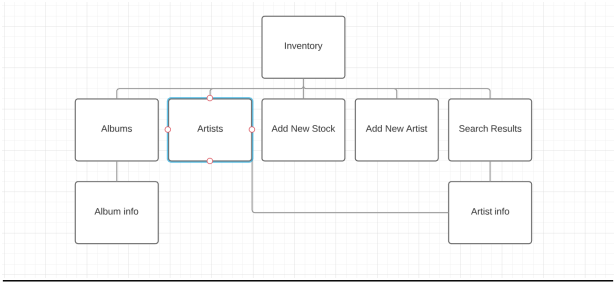
This is an activity diagram which shows the process of a customer using an ATM to check their balance and withdraw money.

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

Topic	Possible Effect of Constraint on Product	Solution
Hardware and software platforms	Product may not be compatible with all machines or browsers	Test on a variety of different machines and browsers to ensure compatibility.
Performance requirements	Product may require a certain type of processor or OS to run	Ensure product is compatible with all current OS and processors.
Persistent storage and transactions	Product may require a lot of memory or data to run.	Ensure all files are as small as they possibly can be.
Usability	Product may have usability issues for people with disabilities or may not be completely user friendly.	Use tools such as alt tags on images to make product as accessible as possible for all, carry out user tests to see where things need to change to make product more user friendly.
Budgets	Not enough money to complete product exactly how the client wants it to be completed.	More money, change the product slightly to fit the budget, find ways to save money and still deliver the best possible product
Time	Product may not reach its full potential in the given time frame as there is only time to implement the basics	More time

An implementations constraints plan for a program.

Unit	Ref	Evidence
P	P.5	User Site Map



A user sitemap for a record store inventory web app.

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

Wireframe diagrams showing an inventory page and an add new artist page for a record store web app.

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

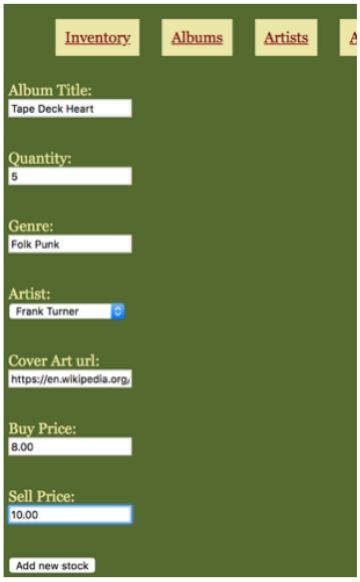
```
#if the quantity is greater than or equal to 10 then return high, else if the quantity is
between 5 and 9 return medium, else return low.

def stock_level()

  if @quantity >= 10
    return "high"
  elsif @quantity >= 5 && @quantity <= 9
    return "medium"
  elsif @quantity <= 4
    return "low"
  end
end
```

Pseudocode for a function that returns low, medium or high based on stock level.

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

A user inputting album information for a new album into a program, then that album appearing in the inventory.

```
<div class="new">

  <form action="/album" method="post">
    <label for="title">Album Title: </label><br>
    <input type="text" id="title" name="title"><br>

    <label for ="quantity"> Quantity: </label><br>
    <input type="number" id="quantity" name="quantity"><br>

    <label for="genre">Genre: </label><br>
    <input type="text" id="genre" name="genre"><br>

    <label for="artist"> Artist: </label><br>
    <select name="artist_id" id="artist_id">
      <% @artists.each do |artist| %>
      <option value="<%=artist.id%>"><%=artist.name%></option>
      <% end %>
    </select><br>

    <label for="url"> Cover Art url: </label><br>
    <input type="text" id="url" name="url"><br>

    <label for="buy_price"> Buy Price: </label><br>
    <input type="text" id="buy_price" name="buy_price"><br>

    <label for="sell_price"> Sell Price: </label><br>
    <input type="text" id="sell_price" name="sell_price"><br>

    <input type="submit" value="Add new stock">

  </form>
```



Frank Turner	Love, Ire and Song	7	folk punk	medium	£4.50	£10.00	£5.50	£38.50
Frank Turner	Poetry of the Deed	9	folk punk	medium	£4.50	£9.00	£4.50	£40.50
Frank Turner	Tape Deck Heart	5	Folk Punk	medium	£8.00	£10.00	£2.00	£10.00
Frank Turner	Sleep Is For The Week	2	folk punk	low	£5.50	£10.00	£4.50	£9.00
Run the Jewels	Run the Jewels	3	hip hop	low	£5.00	£10.00	£5.00	£15.00

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

Seed data for a program with the save function and the result of that save function storing the data to the database in psql.

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

```
album1 = Album.new(
  'title' => 'Sleep Is For The Week',
  'quantity' => 5,
  'genre' => 'folk punk',
  'artist_id' => artist1.id,
  'url' => 'https://i.imgur.com/HURz1ul.jpg',
  'buy_price' => 5.50,
  'sell_price' => 10.00
)

album1.save

album2 = Album.new(
  'title' => 'Love, Ire and Song',
  'quantity' => 7,
  'genre' => 'folk punk',
  'artist_id' => artist1.id,
  'url' => 'https://i.imgur.com/QkClQ0L.jpg',
  'buy_price' => 4.50,
  'sell_price' => 10.00
)

album2.save

album3 = Album.new(
  'title' => 'Poetry of the Deed',
  'quantity' => 9,
  'genre' => 'folk punk',
  'artist_id' => artist1.id,
  'url' => 'https://i.imgur.com/SE8HKdB.gif',
```

```
#save albums
def save()
  sql = "INSERT INTO albums (title, quantity, genre, artist_id, url, buy_price, sell_price)
  VALUES ($1, $2, $3, $4, $5, $6, $7) RETURNING id;"
  values = [@title, @quantity, @genre, @artist_id, @url, @buy_price, @sell_price]
  album = SqlRunner.run(sql, values).first
  @id = album['id'].to_i
end
```

```
id |        title         | quantity |   genre   | artist_id |                                  url
          | buy_price | sell_price
----+----------------------+----------+-----------+-----------+-----------------------------------------------------------------------------------------------------
 2 | Love, Ire and Song   |        7 | folk punk |         1 | https://i.imgur.com/QkClQ0L.jpg
          |       4.5 |      10.0
 3 | Poetry of the Deed   |        9 | folk punk |         1 | https://i.imgur.com/SE8HKdB.gif
          |       4.5 |       9.0
 4 | With Love and Squalor |        5 | indie     |         2 | https://i.imgur.com/1oaLG21.jpg
          |       3.0 |       8.0
 5 | Brain Thrust Mastery |        4 | indie     |         2 | https://i.imgur.com/cZ8YhZB.jpg
          |       3.0 |       7.5
 6 | Run the Jewels       |        3 | hip hop   |         3 | https://i.imgur.com/sbdQQDi.jpg
          |       5.0 |      10.0
 7 | 1989                 |       11 | pop       |         4 | https://i.imgur.com/eHUgso7.png
          |       5.5 |      11.0
 8 | Blackened Sky        |       -1 | rock      |         5 |
          |       4.5 |       8.5
 9 | rupaul               |        6 | drag      |         6 |
          |       8.0 |      10.0
10 | Tape Deck Heart      |        5 | Folk Punk |         1 | https://en.wikipedia.org/wiki/Tape_Deck_Heart#/media/File:Tape_Deck_Heart_Alb
um_Cover.jpg |       8.0 |      10.0
 1 | Sleep Is For The Week |        2 | folk punk |         1 |
          |       5.5 |      10.0
(10 rows)

(END)
```

Screenshots of the code used for a user to delete an album from a web app and the result of that album being deleted from the inventory.

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

```ruby
def test_card_total()
cards = [@card1, @card2]
assert_equal("You have a total of 6", @cardgame.card_total(cards))
end
```

```ruby
def card_total(cards)
  total = 0
  for card in cards
    total += card.value

  end
    return "You have a total of" + total.to_s
end
```

```
testing_task_2_spec.rb
➜ specs git:(master) ✗ ruby testing_task_2_spec.rb
Run options: --seed 60268

# Running:

F..

Finished in 0.001132s, 2650.1767 runs/s, 3533.5689 assertions/s.

  1) Failure:
TestCardGame#test_card_total [testing_task_2_spec.rb:33]:
Expected: "You have a total of 6"
  Actual: "You have a total of6"

3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
➜ specs git:(master) ✗
```

```ruby
def card_total(cards)
  total = 0
  for card in cards
    total += card.value

  end
    return "You have a total of " + total.to_s
end
```

```
# Running:

...

Finished in 0.001153s, 2601.9082 runs/s, 3469.2110 assertions/s.

3 runs, 4 assertions, 0 failures, 0 errors, 0 skips
➜ specs git:(master) ✗
```

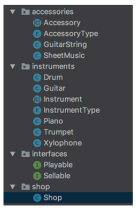
Artist	Album	Quantity	Genre	Stock Level	Buy Price	Sell Price	Margin	Total Profit
Biffy Clyro	Blackened Sky	-1	rock	low	£4.50	£8.50	£4.00	£-4.00
Frank Turner	Poetry of the Deed	9	folk punk	medium	£4.50	£9.00	£4.50	£40.50
Frank Turner	Tape Deck Heart	5	Folk Punk	medium	£8.00	£10.00	£2.00	£10.00
Frank Turner	Sleep Is For The Week	2	folk punk	low	£5.50	£10.00	£4.50	£9.00
Run the Jewels	Run the Jewels	3	hip hop	low	£5.00	£10.00	£5.00	£15.00

Examples of a test to get the total value when adding two cards together failing and then passing when the code has been corrected.

Week 7

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

```java
public class Shop {
    private ArrayList<Sellable> stock;

    public Shop() {
        this.stock = new ArrayList<>();
    }

    public int stockCount() {
        return stock.size();
    }

    public void addStock(Sellable item) {
        this.stock.add(item);
    }

    public void removeStock(Sellable item) {
        this.stock.remove(item);
    }
```

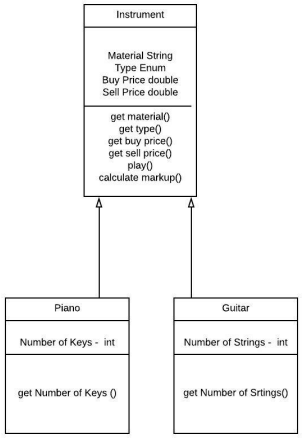



```java
public abstract class Accessory implements Sellable {

    private double buyPrice;
    private double sellPrice;
    private AccessoryType accessoryType;

    public Accessory(double buyPrice, double sellPrice, AccessoryType accessoryType) {
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
        this.accessoryType = accessoryType;
    }
```

```java
public abstract class Instrument implements Playable, Sellable {

    private String material;
    private String colour;
    private InstrumentType instrumentType;
    private String make;
    private double buyPrice;
    private double sellPrice;

    public Instrument(String material, String colour, InstrumentType instrumentType, String make, double buyPrice, doub
        this.material = material;
        this.colour = colour;
        this.instrumentType = instrumentType;
        this.make = make;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }
```

All instruments and accessories in the music shop implement the Sellable interface and therefore can be used in functions. All instruments and accessories can be added to the array list of items in the shop by using the object type Sellable in functions.

Unit	Ref	Evidence
A&D	A.D.5	An Inheritance Diagram



An Inheritance Diagram showing a Piano and a Guitar inheriting from an Instrument class.

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

```java
public class Book {

    private String title;
    private String genre;

    public Book(String title, String genre){
        this.title = title;
        this.genre = genre;
    }

    public String getTitle() {
        return title;
    }

    public String getGenre() {
        return genre;
    }
}
```

An example of a book class with private properties, preventing direct access to those properties outwit the class, with getter methods that can be used elsewhere in the program in order to access these values.

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.

```java
public abstract class Instrument implements Playable, Sellable {

    private String material;
    private String colour;
    private InstrumentType instrumentType;
    private String make;
    private double buyPrice;
    private double sellPrice;

    public Instrument(String material, String colour, InstrumentType instrumentType, String make, double buyPrice, double sellPrice) {
        this.material = material;
        this.colour = colour;
        this.instrumentType = instrumentType;
        this.make = make;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getMaterial() {
        return material;
    }

    public String getColour() {
        return colour;
    }
```

```java
public class Piano extends Instrument {

    private int numberOfKeys;

    public Piano(String material, String colour, InstrumentType instrumentType, String make, double buyPrice, double sellPrice, int numberOfKeys) {
        super(material, colour, instrumentType, make, buyPrice, sellPrice);
        this.numberOfKeys = numberOfKeys;
    }

    public int getNumberOfKeys() {
        return numberOfKeys;
    }

    public String play() {
        return "plinky plonky";
    }
}
```

```java
public class InstrumentTest {

    Instrument guitar;
    Instrument piano;

    @Before
    public void setUp() throws Exception {
        guitar = new Guitar("wood", "blue", InstrumentType.STRING, "Fender", 200.00, 250.00, 6);
        piano = new Piano("wood", "brown", InstrumentType.KEYBOARD, "Yamaha", 900.00, 1200.00, 88);
    }

    @Test
    public void hasMaterial() { assertEquals("wood", piano.getMaterial()); }

    @Test
    public void hasColour() { assertEquals("blue", guitar.getColour()); }

    @Test
    public void hasInstrumentType() { assertEquals(InstrumentType.STRING, guitar.getInstrumentType()); }

    @Test
    public void hasMake() { assertEquals("Yamaha", piano.getMake()); }
```

Screenshots of Instrument class and Piano class which inherits from Instrument, with examples of a new Piano object being used to test the get material and get make methods.

Week 10

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

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.

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.

Paste Screenshot here

Description here

Week 12

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

Paste Screenshot here

Description here

Week 15

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.

Paste Screenshot here

Description here

Unit	Ref	Evidence
P	P.2	Take a screenshot of the project brief from your group project.

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.

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).

Paste Screenshot here

Description here

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

Paste Screenshot here

Description here

Unit	Ref	Evidence
P	P.17	Produce a bug tracking report

Paste Screenshot here

Description here