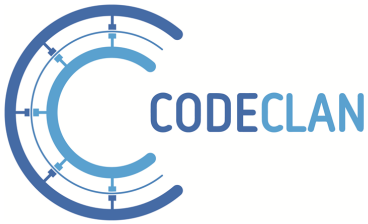


Evidence Gathering Document for SQA Level 8 Professional Developer Award.

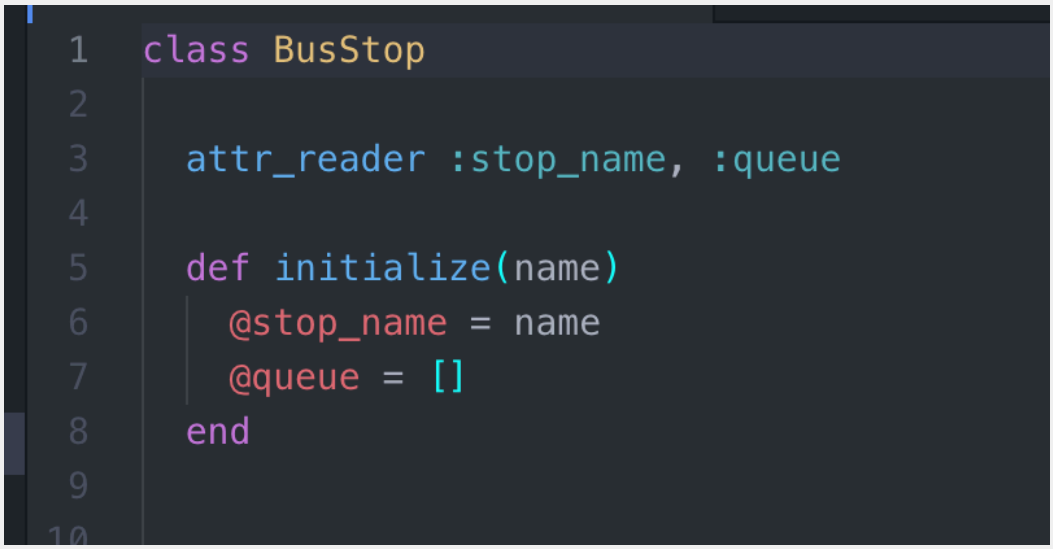
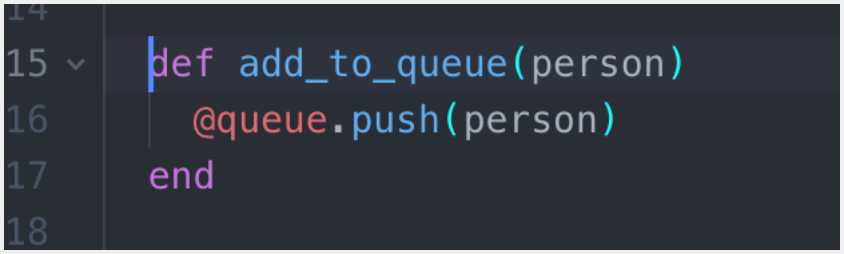
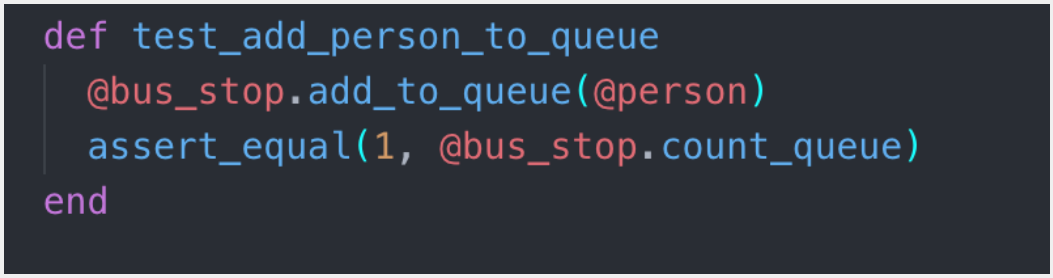
This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

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

Unit	Ref	Evidence
		Description: BusStop class created with empty queue array  **Function add_to_queue takes in a person object and adds it to the queue array**  **Test to see add_to_queue function works**  **Test passes and shows person object with name “Hugh” and age 43 inside bus stop’s queue array** 

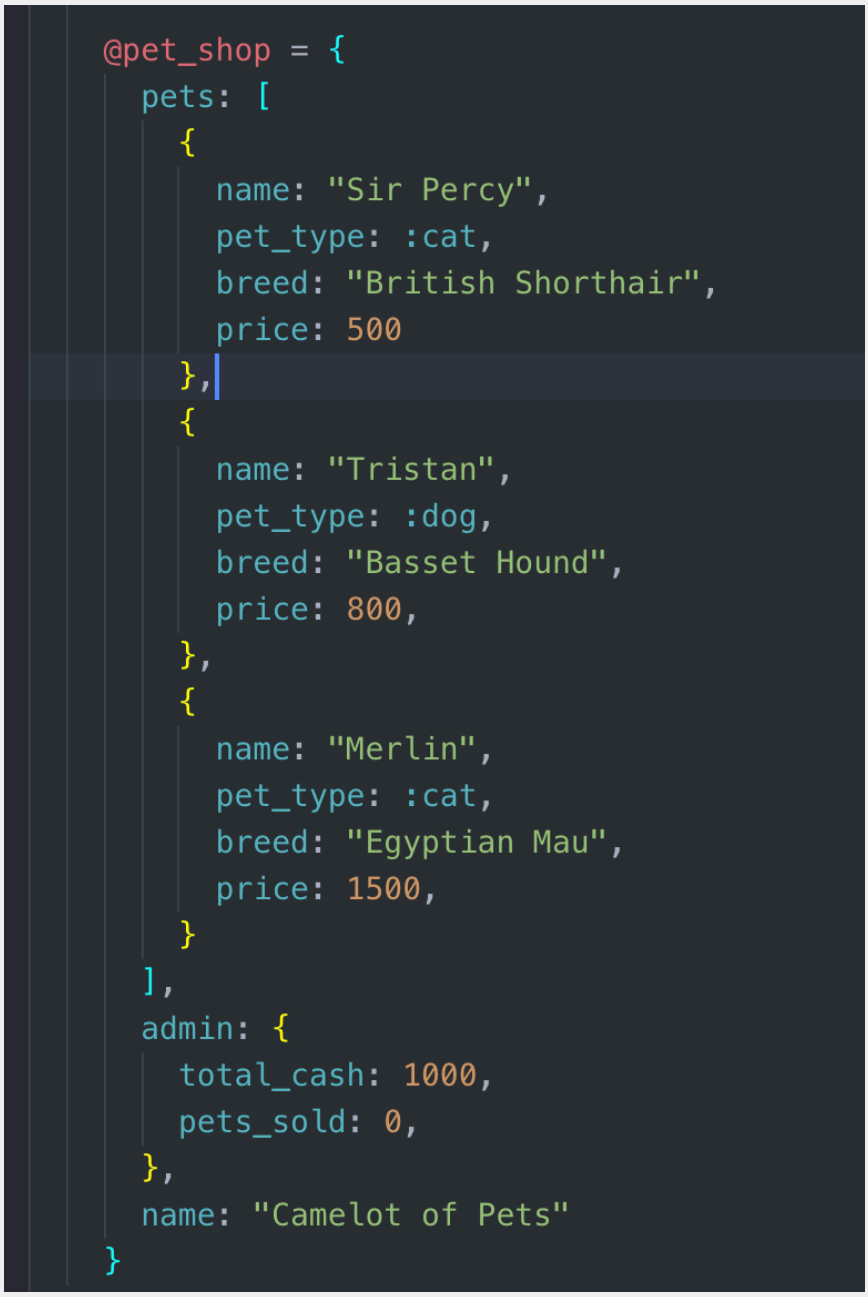
```
class BusStop

  attr_reader :stop_name, :queue

  def initialize(name)
    @stop_name = name
    @queue = []
  end
```

```
  def add_to_queue(person)
    @queue.push(person)
  end
```

```
def test_add_person_to_queue
  @bus_stop.add_to_queue(@person)
  assert_equal(1, @bus_stop.count_queue)
end
```

```
# Running:

[#<Person:0x007fefdc81e600 @name="Hugh", @age=43>]
..

Finished in 0.000779s, 2567.3941 runs/s, 2567.3941 assertions/s.

2 runs, 2 assertions, 0 failures, 0 errors, 0 skips
➜  specs
```

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

Unit	Ref	Evidence
		Description: Pet_shop has an array of pet hashes  **Function removes pet hash from an array** 

```ruby
@pet_shop = {
  pets: [
    {
      name: "Sir Percy",
      pet_type: :cat,
      breed: "British Shorthair",
      price: 500
    },
    {
      name: "Tristan",
      pet_type: :dog,
      breed: "Basset Hound",
      price: 800,
    },
    {
      name: "Merlin",
      pet_type: :cat,
      breed: "Egyptian Mau",
      price: 1500,
    }
  ],
  admin: {
    total_cash: 1000,
    pets_sold: 0,
  },
  name: "Camelot of Pets"
}
```

```ruby
def remove_pet_by_name(pet_shop, pet_name)
  pet_to_delete = find_pet_by_name(pet_shop, pet_name)
  pet_shop[:pets].delete(pet_to_delete)
end
```

Test checks that remove_pet_by_name works

```ruby
def test_remove_pet_by_name
  remove_pet_by_name(@pet_shop, "Sir Percy")
  pet = find_pet_by_name(@pet_shop,"Sir Percy")
  assert_nil(pet)
end
```

Test passes

```
# Running:

.

Finished in 0.000884s, 1131.2217 runs/s, 1131.2217 assertions/s.

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

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

Unit	Ref	Evidence

Description: Function searches database for vehicle of given id

```
def Vehicle.find(id)
  sql = "SELECT * FROM vehicles WHERE id = $1"
  values = [id]
  result = SqlRunner.run(sql, values)[0]
  return Vehicle.new(result)
end
```

Vehicles in sql database

```
;
 id |      name       |   category    | on_hire |        image
----+-----------------+---------------+---------+---------------------
  1 | KITT            | screen_stars  | false   | KITT.jpg
  3 | Crane           | heavy_plant   | false   | crane.jpg
  5 | Bulldozer       | heavy_plant   | true    | bulldozer.jpeg
  2 | Tank            | military      | true    | tank.jpeg
  4 | Penny Farthing  | circus        | true    | penny_farthing.jpg
  6 | Velocipede      | circus        | true    | velocipede.jpg
  7 | Airwolf         | screen_stars  | false   | airwolf.jpg
(7 rows)
```

Result of function running: Airwolf details displayed in form for editing

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: Function gets all vehicles on hire from database and sorts them alphabetically **Vehicles in database in order of id**

```
58
59  #Function to return all vehicles available for rent
60  def Vehicle.available()
61    sql = "SELECT * FROM vehicles WHERE on_hire = $1 ORDER BY name ASC"
62    values = ["false"]
63    results = SqlRunner.run(sql, values)
64    return results.map{ |vehicle| Vehicle.new(vehicle) }
65  end
66
```

```
;
 id |      name       |   category    | on_hire |       image
----+-----------------+---------------+---------+--------------------
  1 | KITT            | screen_stars  | false   | KITT.jpg
  3 | Crane           | heavy_plant   | false   | crane.jpg
  5 | Bulldozer       | heavy_plant   | true    | bulldozer.jpeg
  2 | Tank            | military      | true    | tank.jpeg
  4 | Penny Farthing  | circus        | true    | penny_farthing.jpg
  6 | Velocipede      | circus        | true    | velocipede.jpg
  7 | Airwolf         | screen_stars  | false   | airwolf.jpg
(7 rows)
```

Result of function running - Vehicles displayed in app in alphabetical order

- Home
- Vehicles
- Customers
- Rentals

VEHICLES

- available
- on-hire
- Add new vehicle

Categories

- screen_stars
- heavy_plant
- military
- circus

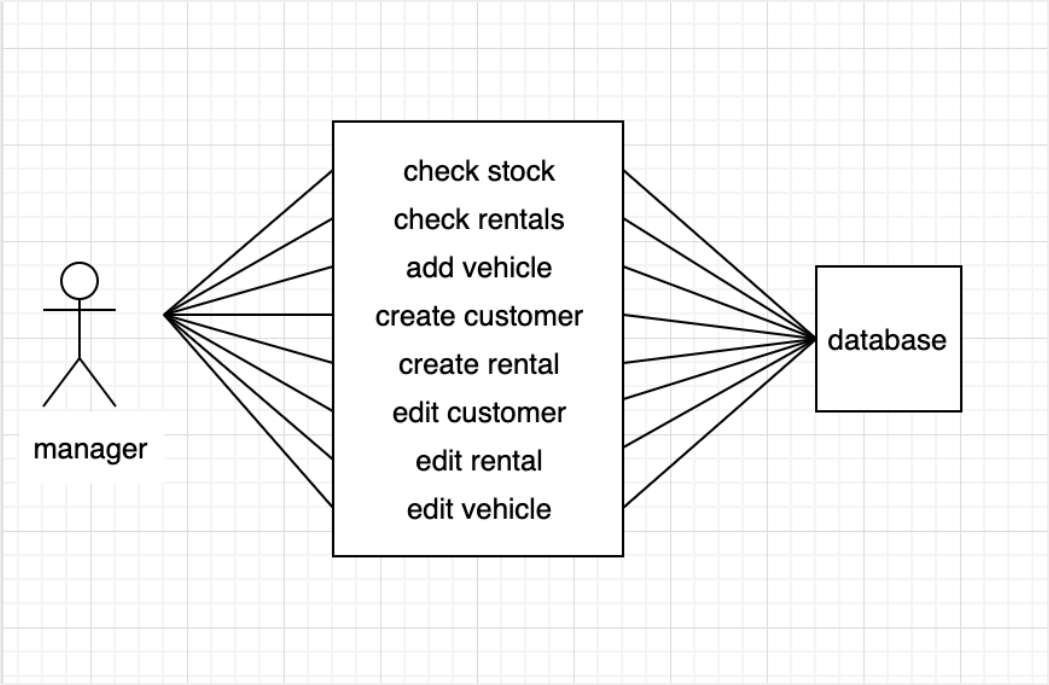
Airwolf

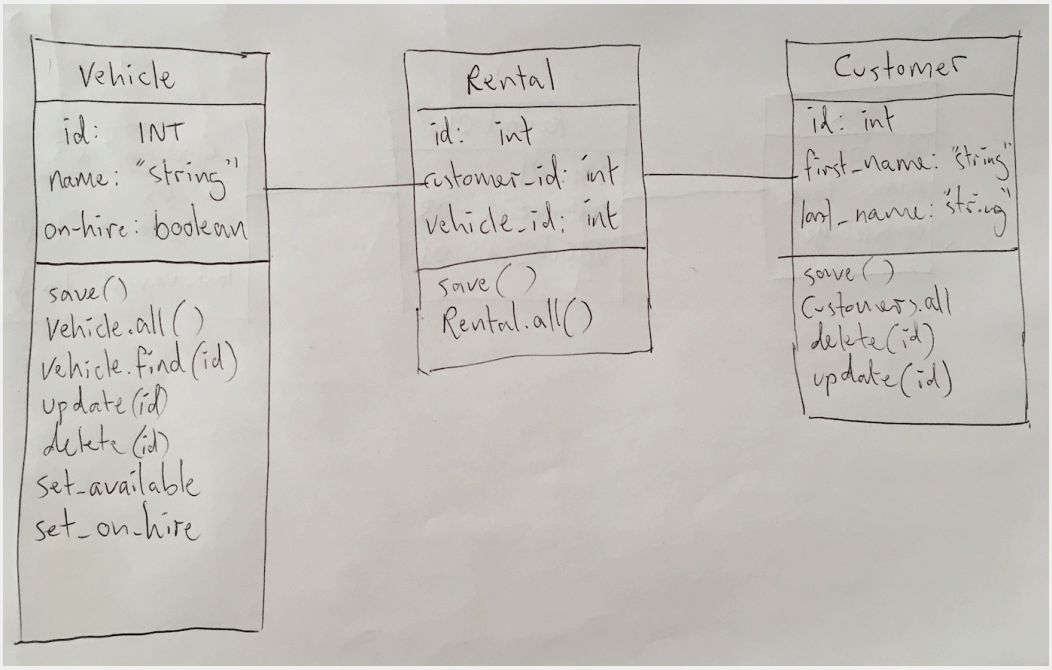
- Show Airwolf details
- Rent Airwolf

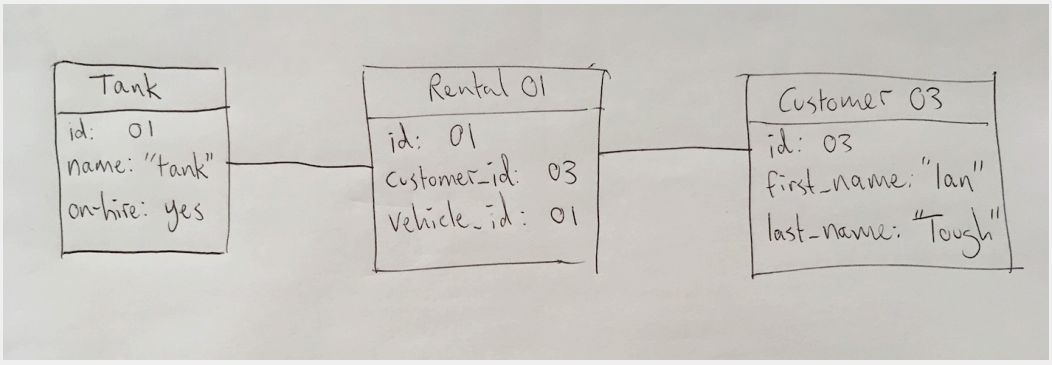
Bulldozer

- Show Bulldozer details

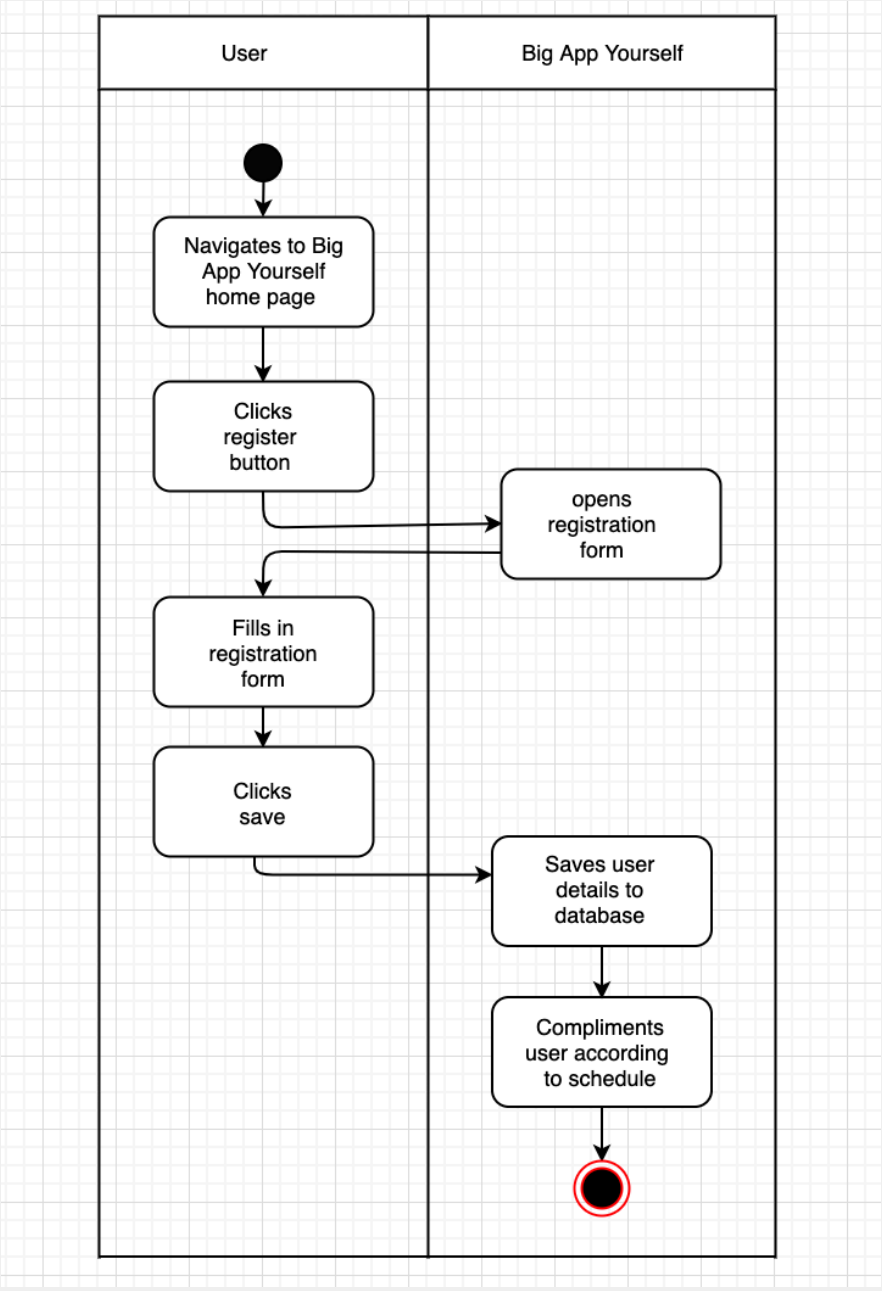
Check-in Bulldozer

Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram
		Description: Use case diagram from rental shop project 

Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram
		Description: Vehicle, Rental and Customer class diagrams from rental shop project 

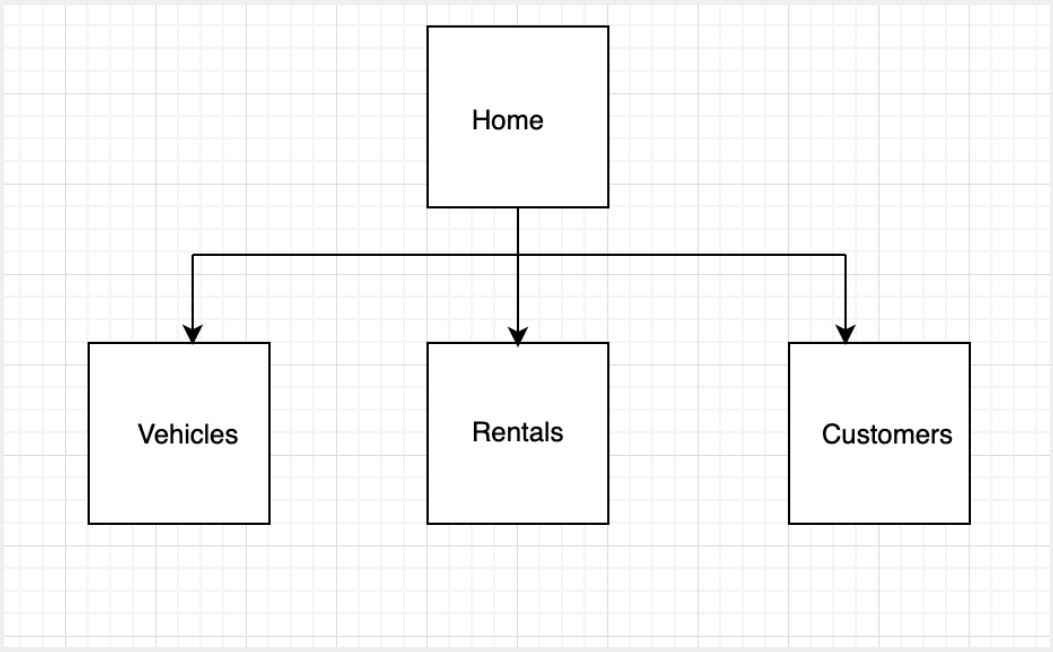
Unit	Ref	Evidence
A&D	A.D.3	An Object Diagram
		Description: Object diagram showing customer Ian Tough renting a tank 

Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram
		Description: Activity diagram for Big App Yourself project

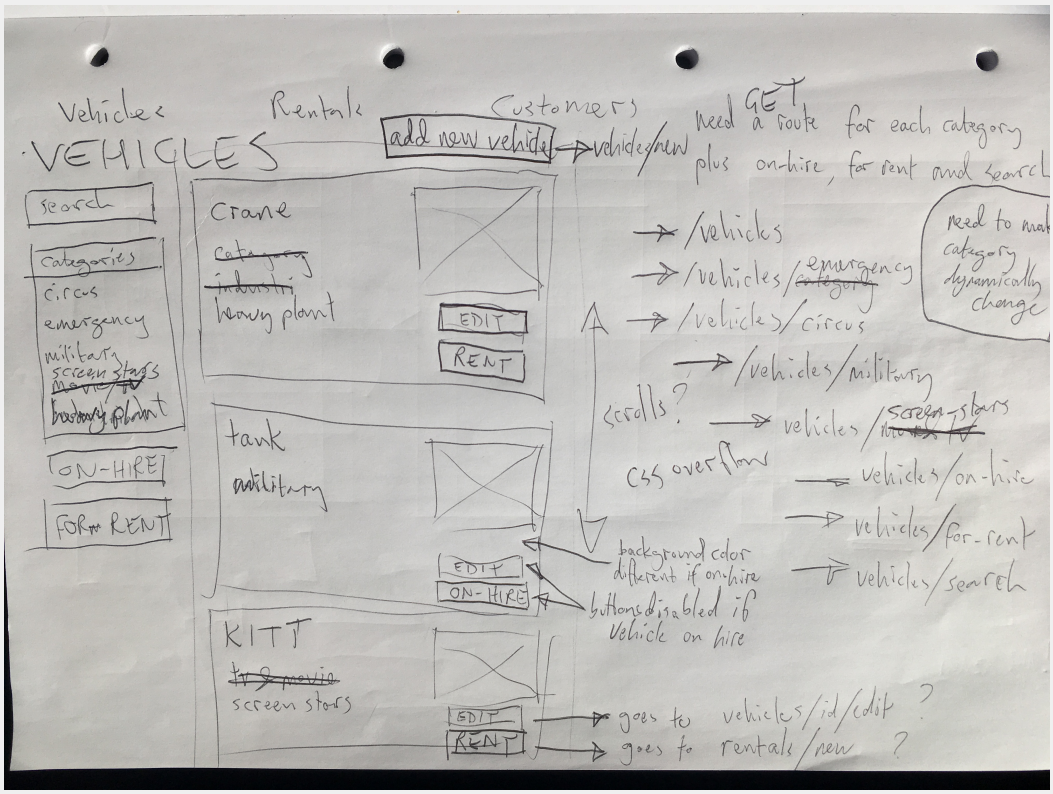
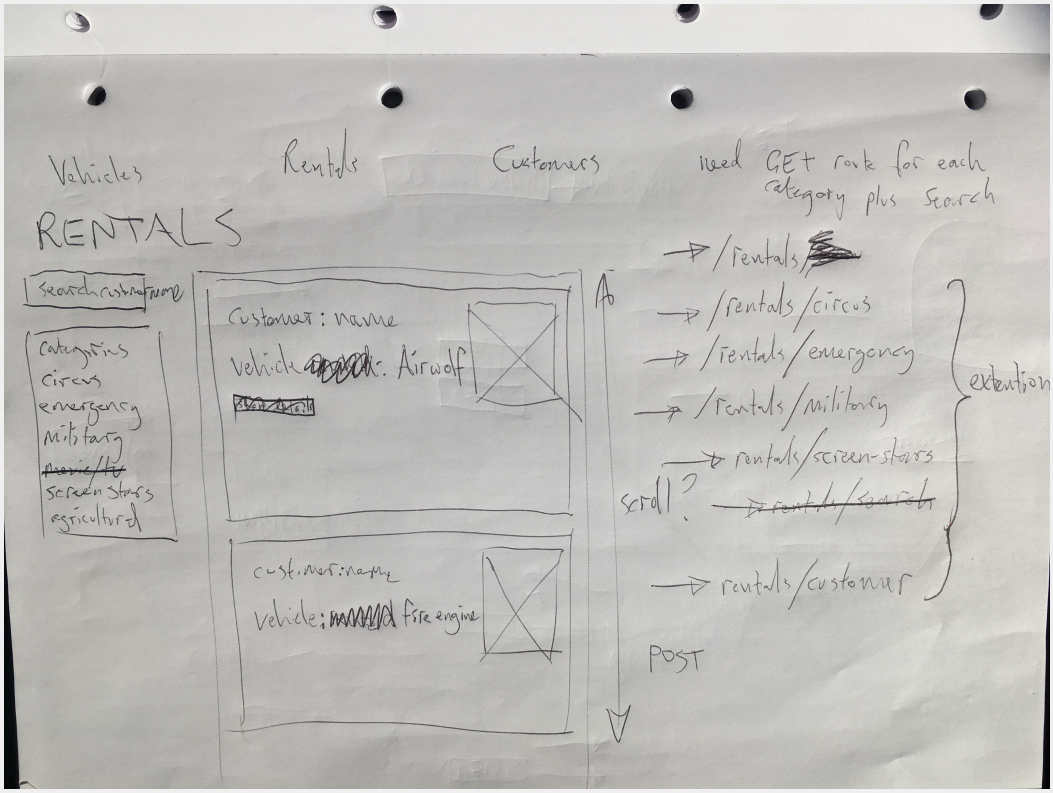


Unit	Ref	Evidence
A&D		Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time
		Description: Implementation and constraints plan for Big App Yourself

Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	Some limitations in date/time functionality supported by web browsers.	Use more standard date/time functionality supported universally.
Performance Requirements	Web server is local and not capable of supporting larger user base	Scaleable web server would be a solution
Persistent Storage and Transactions	Database is local and not capable of supporting larger user base	Scaleable database would be a solution
Usability	Some limitations on mobile device usage, smaller screens	Implement better mobile functionality with CSS
Budgets	Scaleable web server and database costs money	Investment would support scaleability.
Time Limitations	App functionality is limited due to time constraints	More time spent on developing functionality

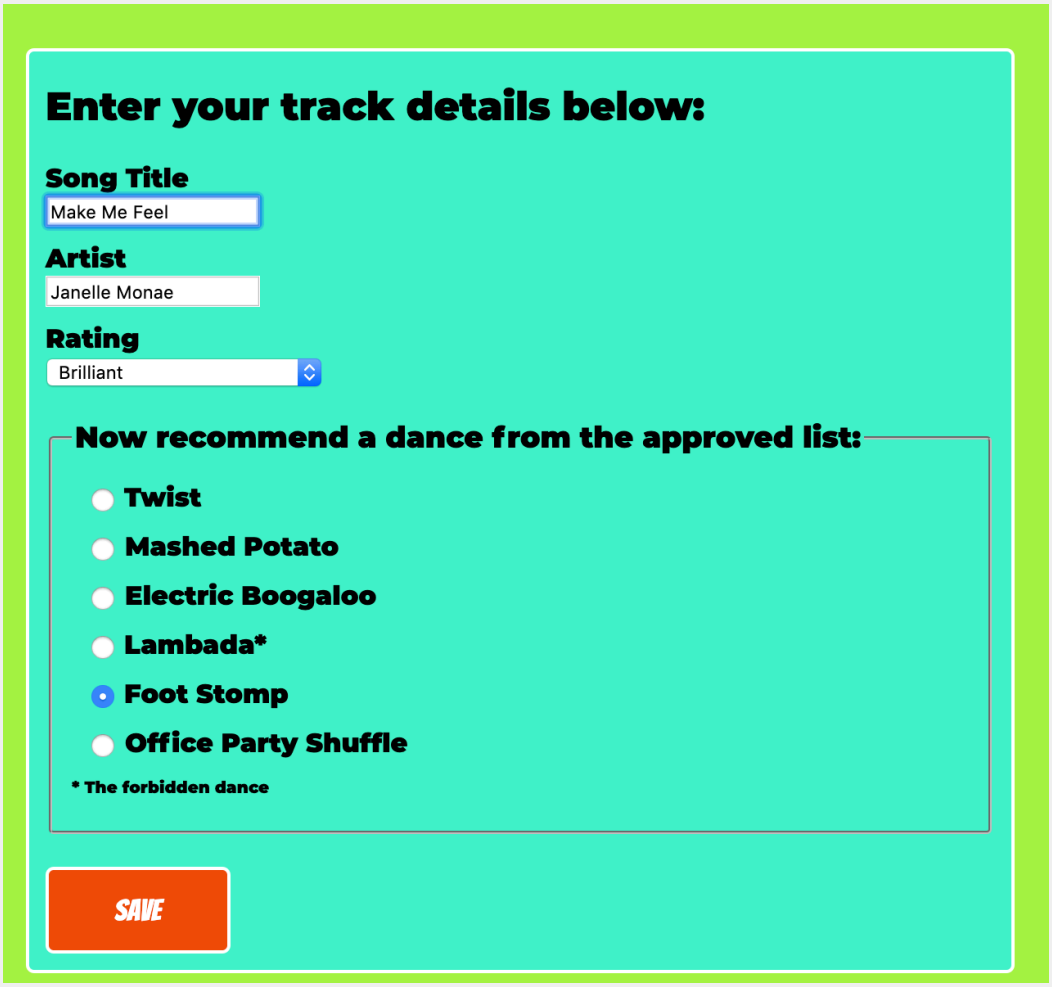
Unit	Ref	Evidence
P	P.5	User Site Map
		Description: Site map for rental shop app 

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

Unit	Ref	Evidence
		Description: Wireframe for Vehicles view in rental shop project  **Wireframe for Rentals view in rental** 

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

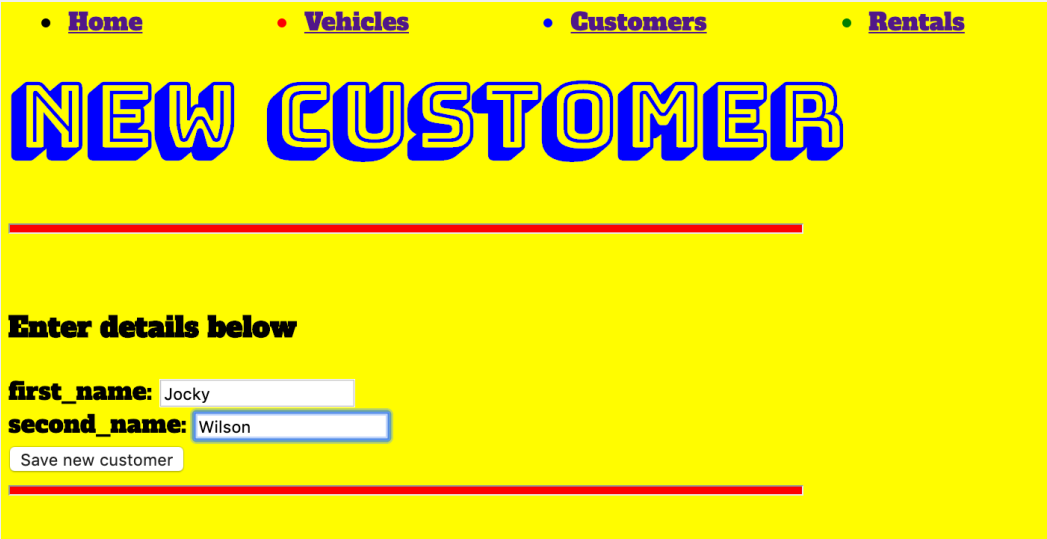
		Description: Pseudocode in BusStopTest method
		
		`class BusStopTest < MiniTest::Test`
		
		`def setup`
		`  @bus_stop = BusStop.new("Duke Street")`
		`  @person = Person.new("Hugh", 43)`
		`end`
		
		`  def test_get_stop_name`
		`  assert_equal("Duke Street", @bus_stop.stop_name)`
		`  end`
		
		`# write test for adding erson to queue`
		`  def test_add_person_to_queue`
		`  # add person defined in setup to queue`
		`  @bus_stop.add_to_queue(@person)`
		`  #print out the queue to see if person is in it`
		`  p (@bus_stop.queue)`
		`  #assert there is now one person in the queue`
		`  assert_equal(1, @bus_stop.count_queue)`
		`  end`
		
		`end`

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: User enters song details and chooses accompanying dance  **Track and dance added to playlist** 

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

Unit	Ref	Evidence

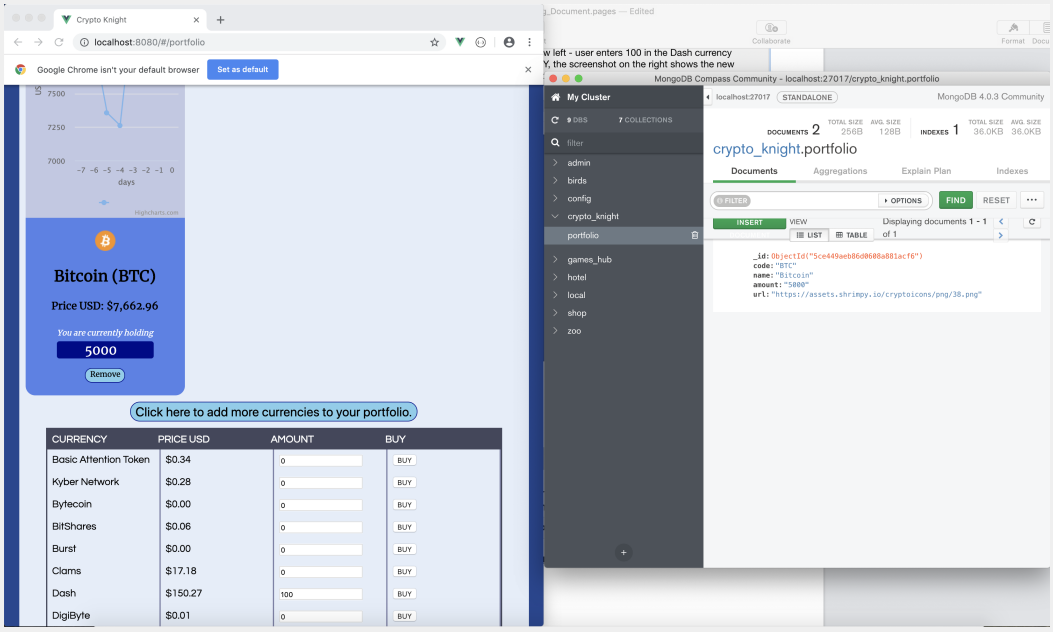
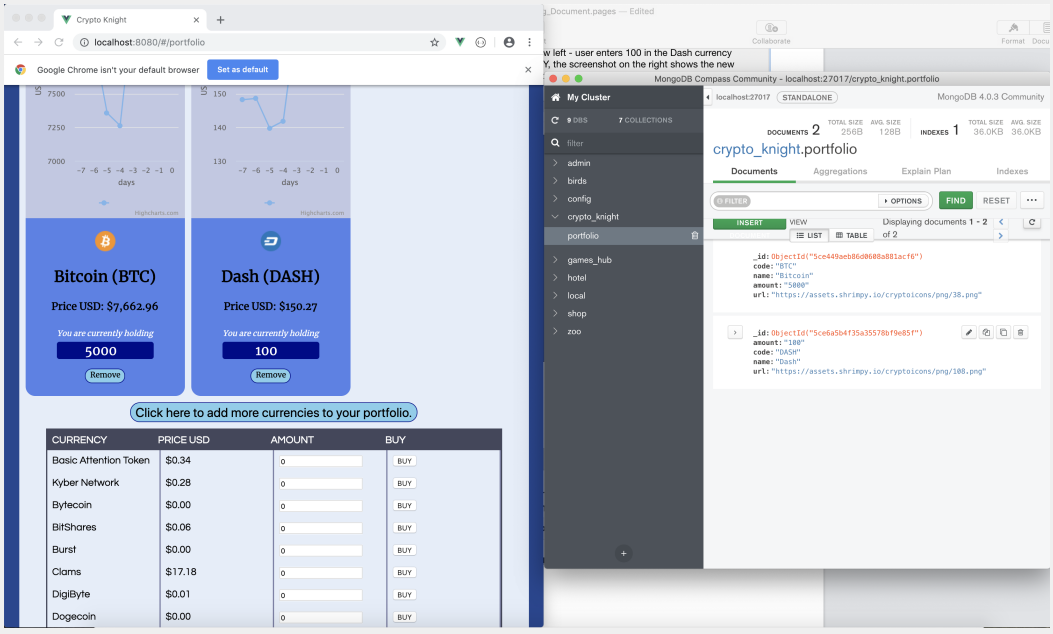
Description: Customer details entered into rental shop app

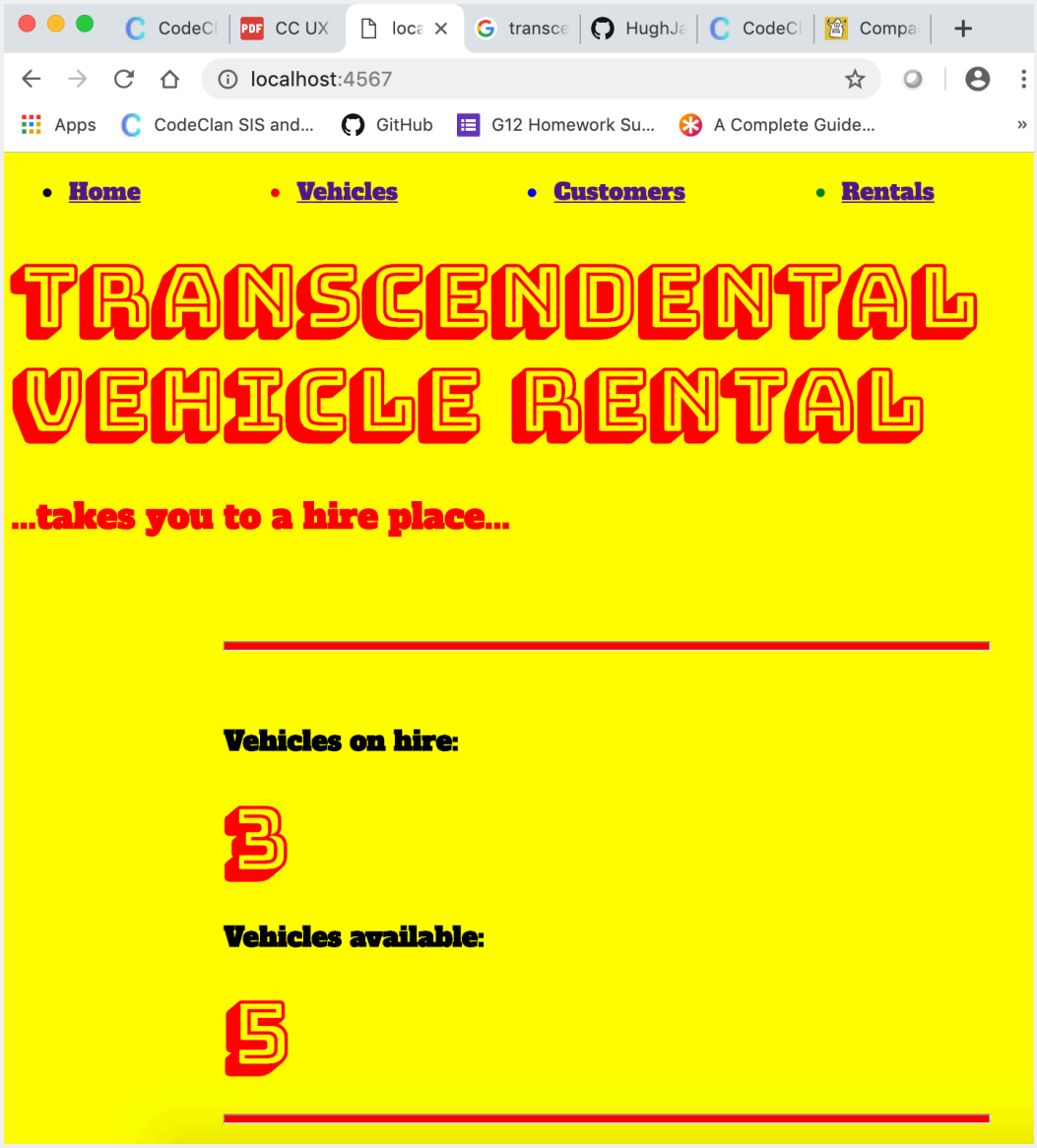


Customer now appears in customer list

Customer's details showing in sql database (id 15)

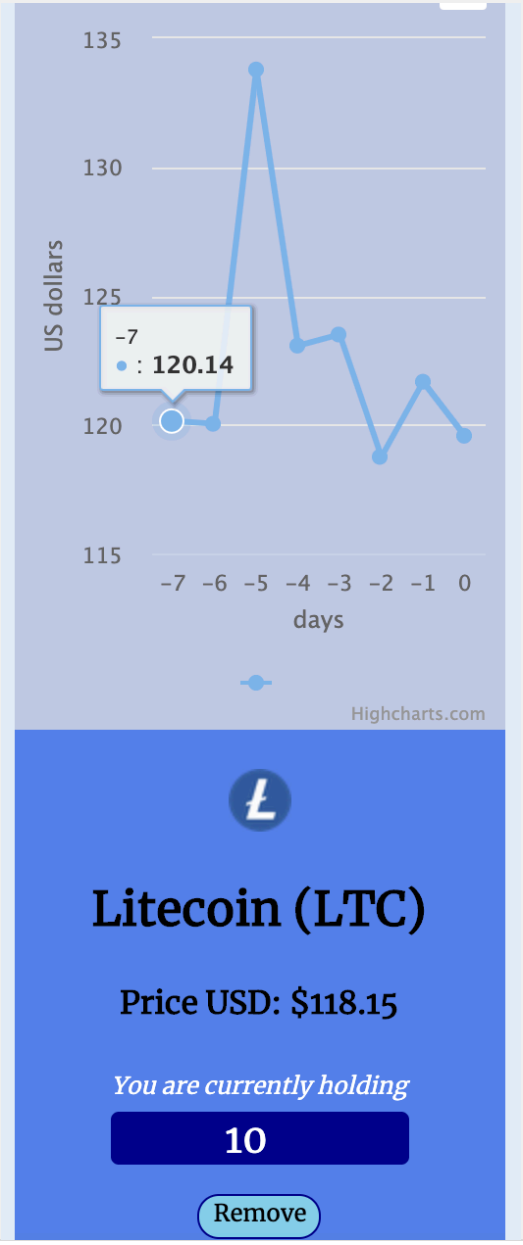
```
rental_shop=# SELECT * FROM customers;
 id | first_name | second_name
----+------------+-------------
  1 | Lorraine   | Kelly
  2 | Ian        | Tough
  3 | Andy       | Murray
  4 | Francis    | McKee
  5 | Eugene     | Kelly
  6 | Charlie    | Reid
  7 | Mary       | Marquis
  8 | Lorraine   | Kelly
  9 | Ian        | Tough
 10 | Andy       | Murray
 11 | Francis    | McKee
 12 | Eugene     | Kelly
 13 | Charlie    | Reid
 14 | Mary       | Marquis
 15 | Jocky      | Wilson
(15 rows)
```

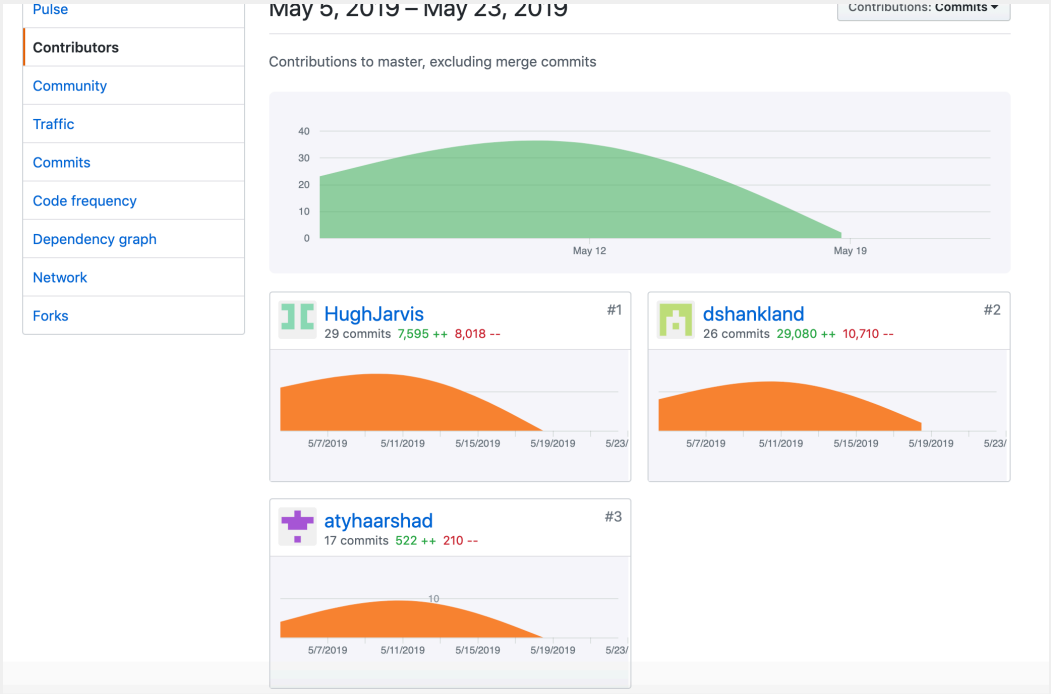
Unit	Ref	Evidence
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program
		Description: User has entered 100 Dash into form but has not clicked buy. Crypto tracking app and MongoDB show only user's Bitcoin holding.  **User has now clicked buy, so 100 Dash holding now shows in app and in MongoDB** 

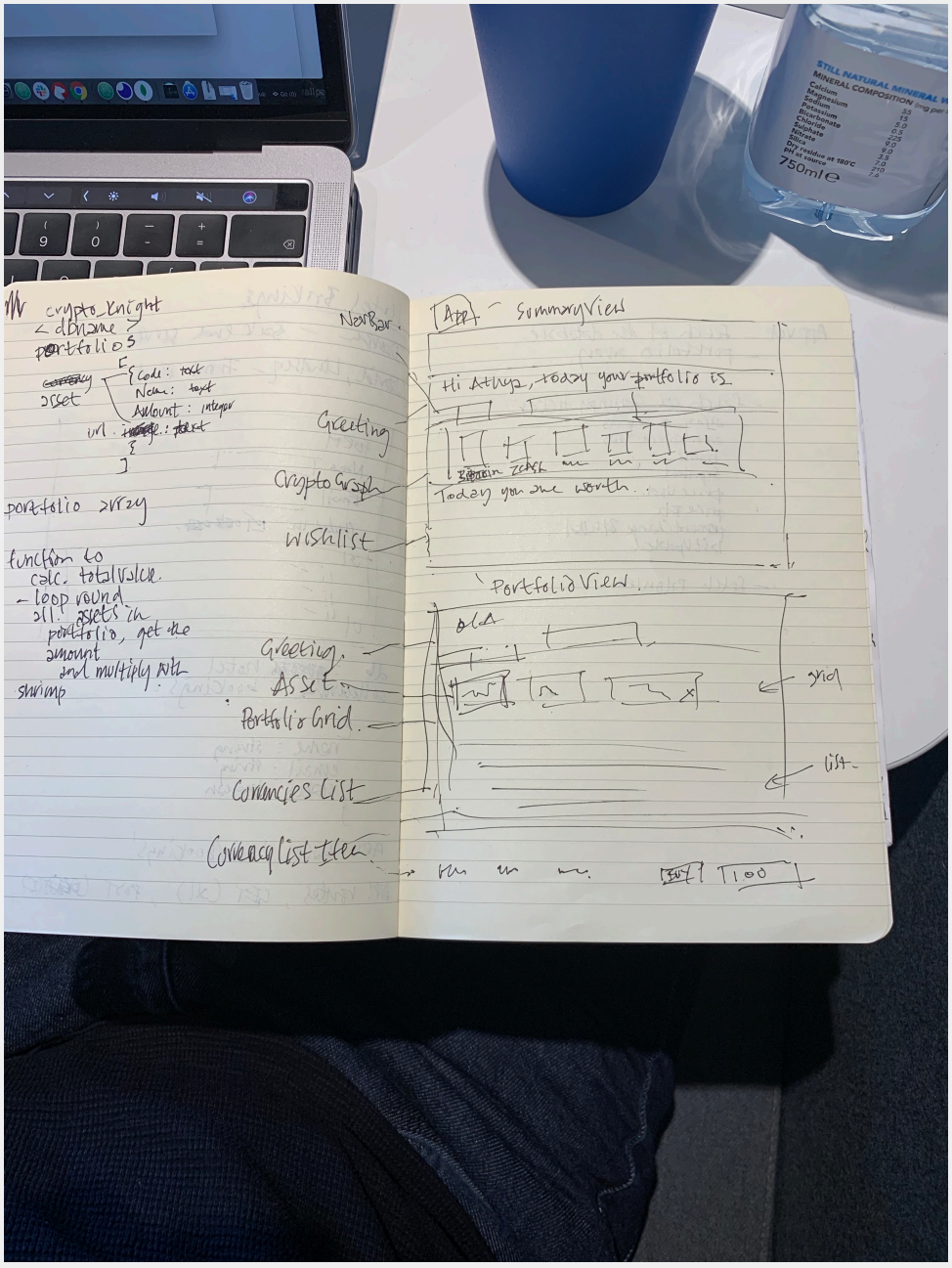
Unit	Ref	Evidence
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.
		Description: Screenshot of home page from my rental shop project  **Link to GitHub repo** https://github.com/HughJarvis/Rental_shop

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

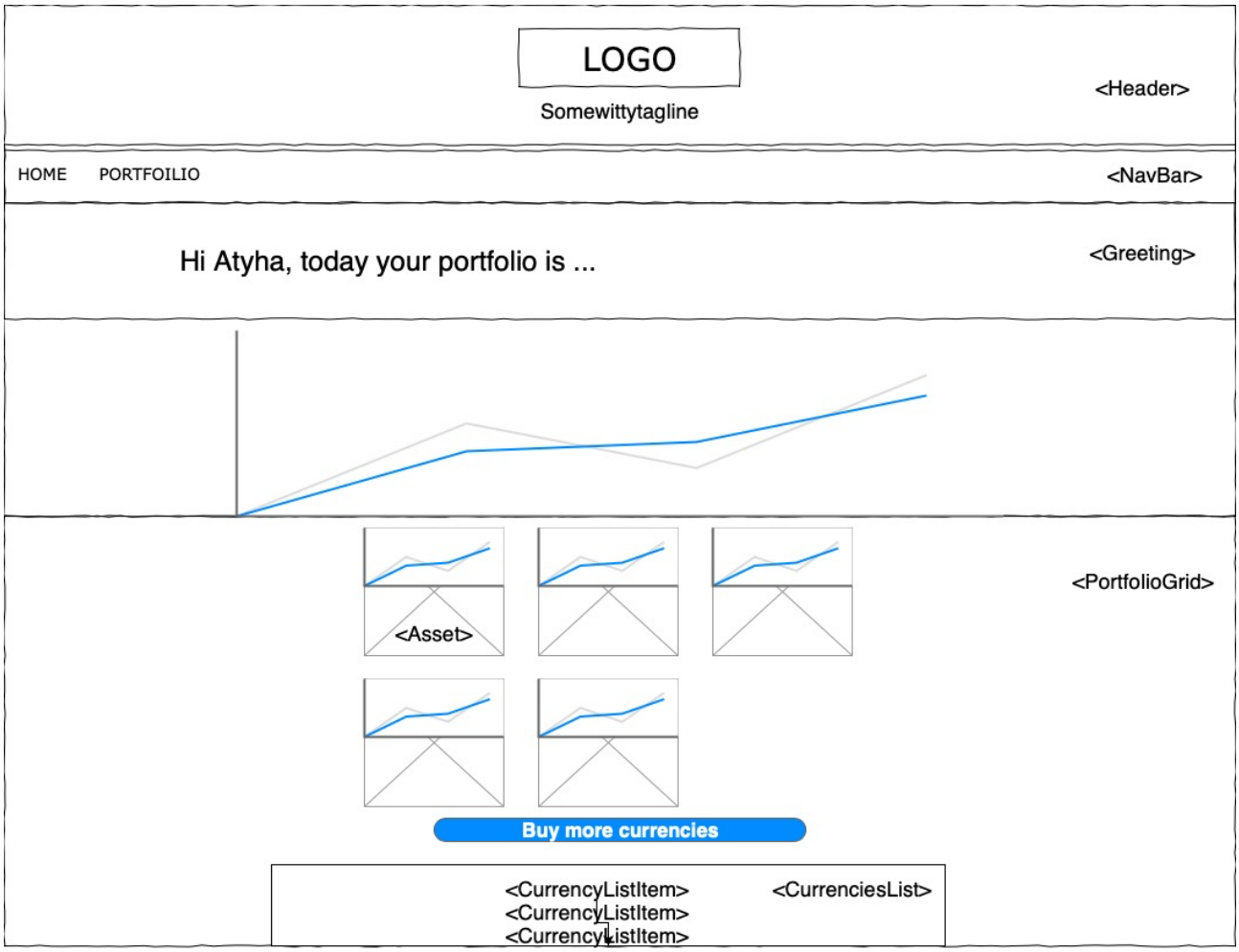
Unit	Ref	Evidence

		Fetch method to get info back from Shrimpy currencies API in Crypto Knight portfolio tracker
		
		`fetchShrimpyTicker: function (){`
		`  fetch('https://cors-anywhere.herokuapp.com/https://dev-api.shrimpy.io/v1/exchanges/poloniex/ticker')`
		`  .then(res => res.json())`
		`  .then(data => {`
		`  console.log(data);`
		`  this.shrimpy = data;`
		`  })`
		
		User adds 10 Litecoin to their portfolio
		
		CURRENCY / PRICE USD / AMOUNT / BUY
		Litecoin — $118.12 — 10 — BUY
		Ardor — $0.11 — 0 — BUY
		
		Price data from Poloniex API is rendered as a chart
		
		 

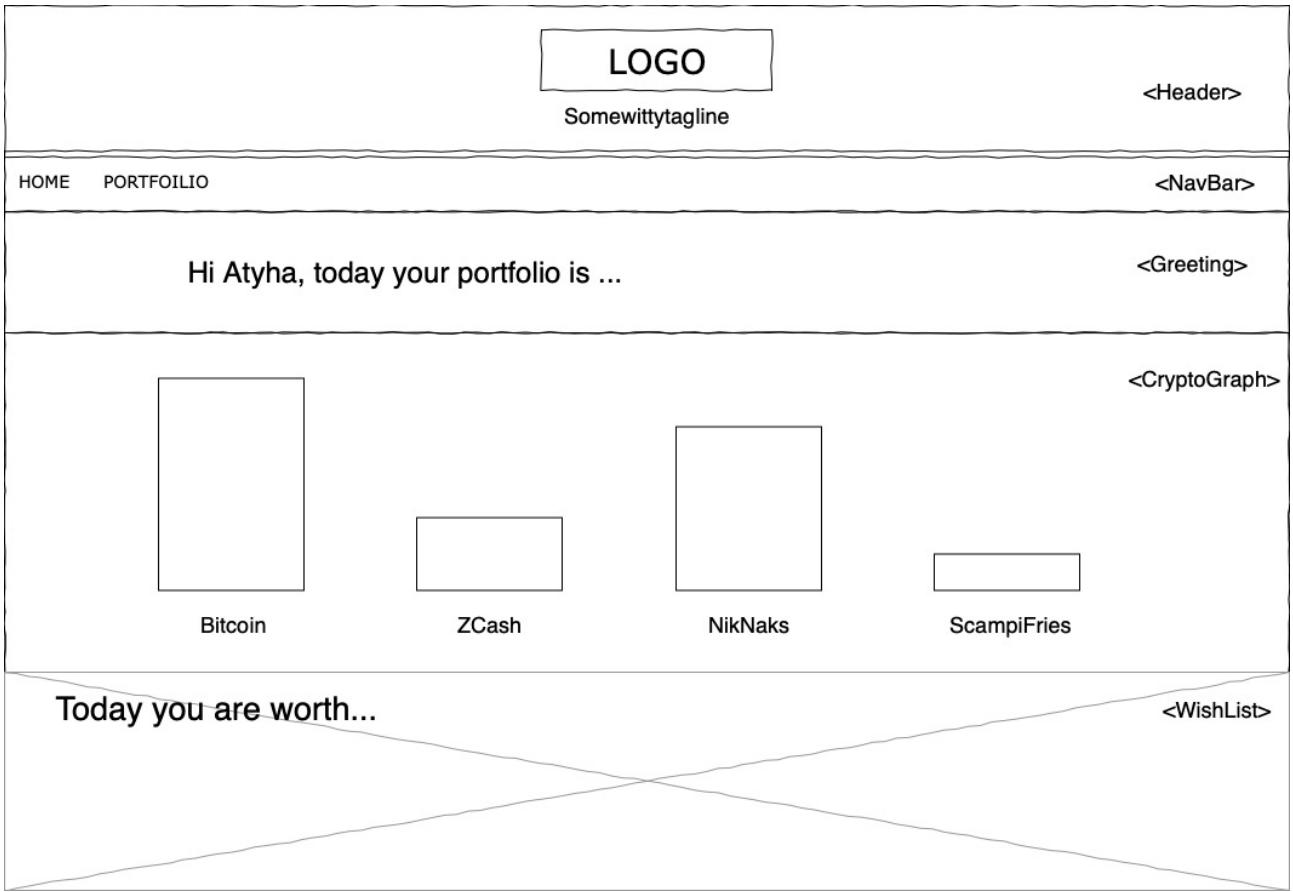
Unit	Ref	Evidence
P	P.1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.
		Description: Contributors page from crypto currency tracker group project GitHub repo 

Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.
		Description: Notes showing rough wireframes and Vue components. 

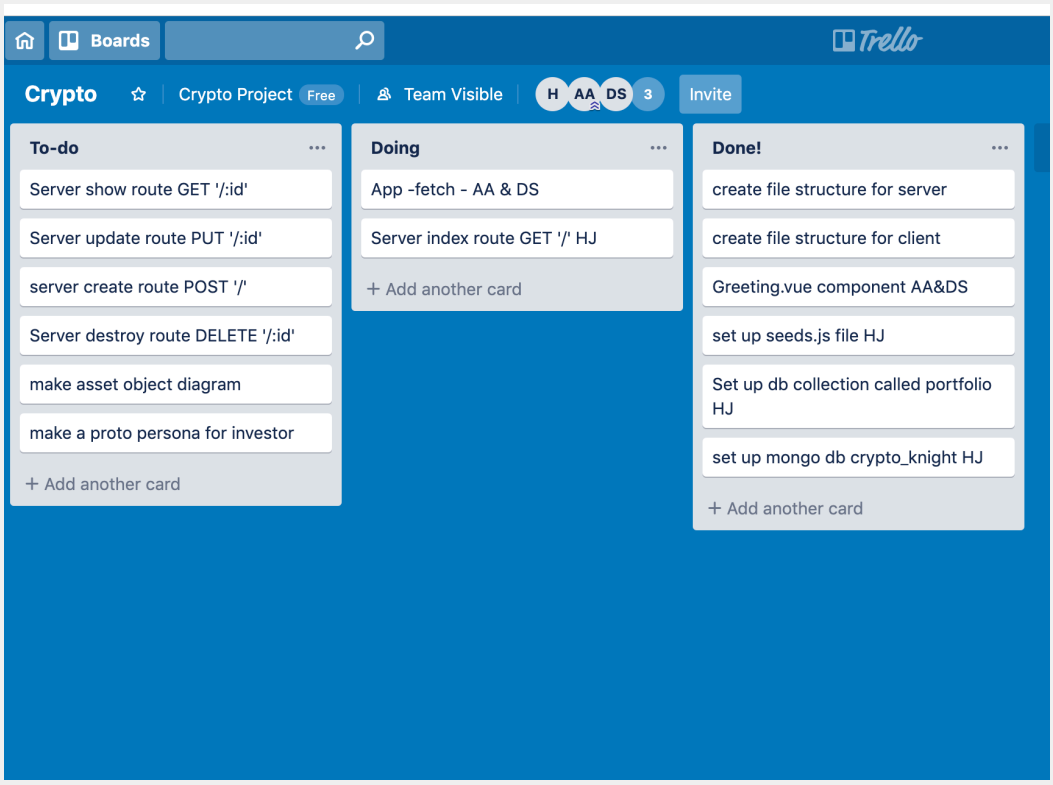
Wireframe for portfolio performance view



Wireframe for main portfolio view



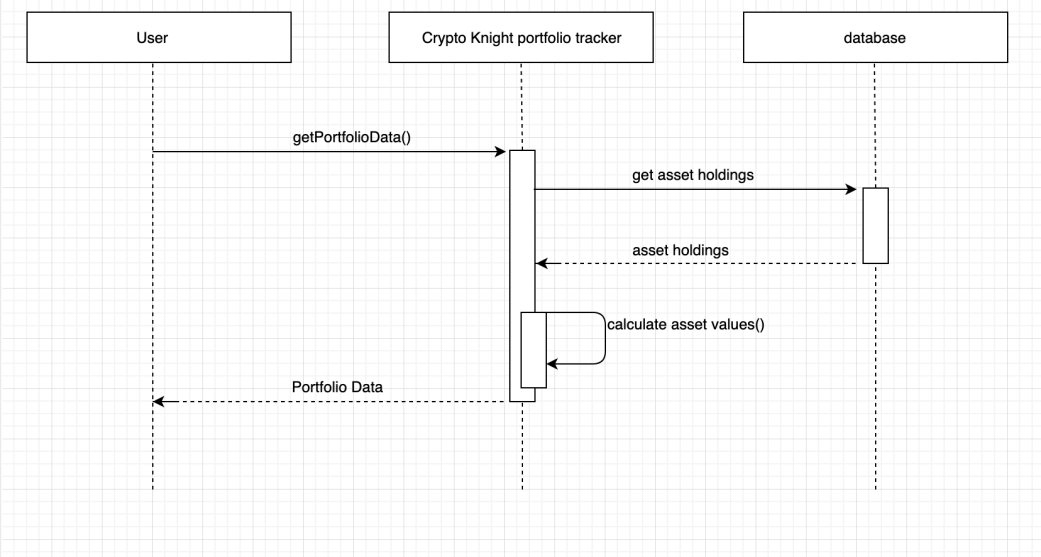
Unit	Ref	Evidence
P	P.2	Take a screenshot of the project brief from your group project.
		Description: Project brief from group project ## Shares Portfolio Application A local trader has come to you with a portfolio of shares. She wants to be able to analyse it more effectively. She has a small sample data set to give you and would like you to build a Minimum Viable Product that uses the data to display her portfolio so that she can make better decisions. ### MVP A user should be able to: - view total current value. - view individual and total performance trends. - retrieve a list of share prices from an external API and allow the user to add shares to her portfolio. - View a chart of the current values in her portfolio. ### Example Extensions - Speculation based on trends and further financial modelling using projections. ### API, Libraries, Resources - https://www.alphavantage.co/ (Requires sign up) - https://www.highcharts.com/ HighCharts is an open-source library for rendering responsive charts.

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

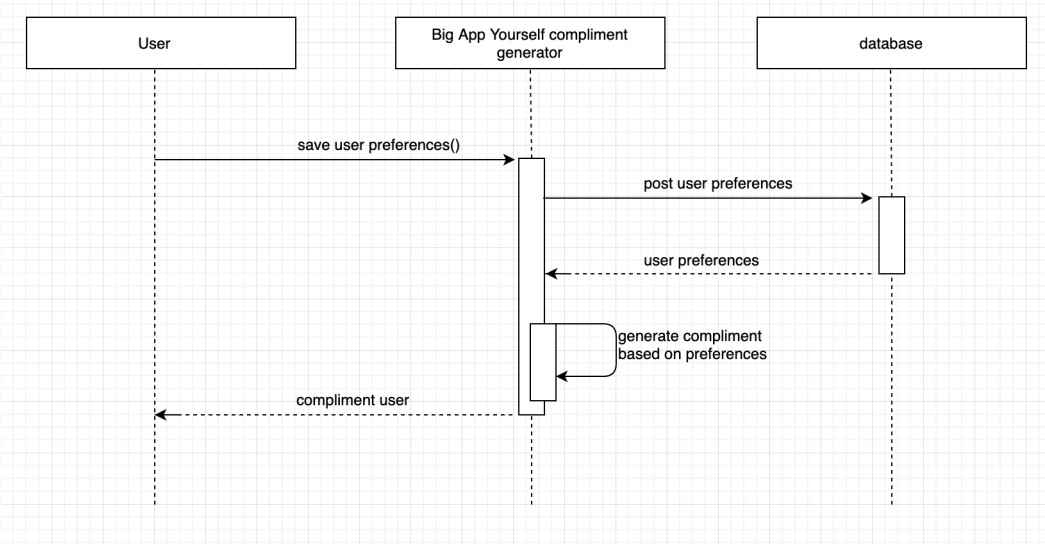
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.
		Acceptance criteria for Crypto Knight portfolio tracker

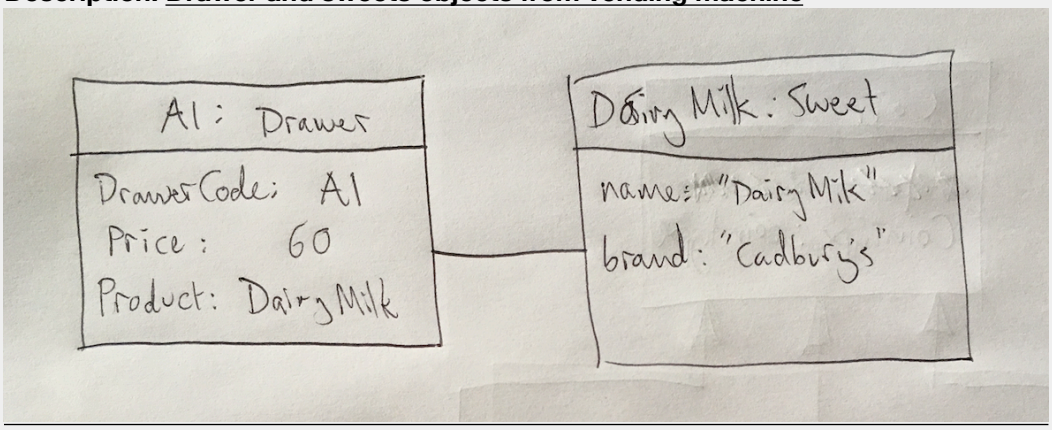
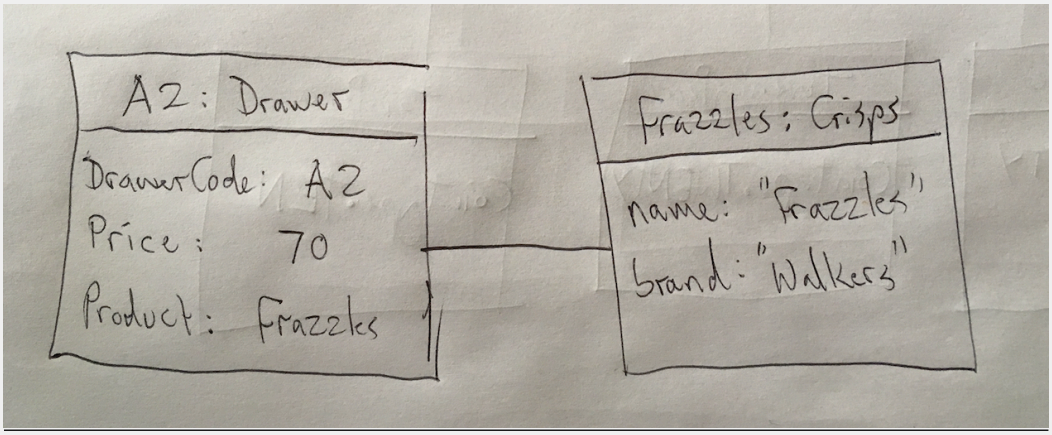
Acceptance criteria	Test plan	Pass/fail
User can add crypto currency to their portfolio	But function will add currency holding to user's portfolio	Pass
Portfolio updates when user adds new currency	App fetches current price info from APIs	Pass
User can sell currency from their portfolio	Sell function deducts currency from portfolio	Pass
User can check their buying power	Buying power tracker shows how many Ferraris portfolio is worth	Pass

Unit	Ref	Evidence
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).
		Description: Sequence diagram for Crypto Knight portfolio tracker



Sequence diagram for Big App Yourself compliment generator



Unit	Ref	Evidence
P	P.8	Produce two object diagrams.
		Description: <u>Drawer and sweets objects from vending machine</u>  **<u>Drawer and crisps objects from vending machine</u>** 

Unit	Ref	Evidence
P	P.17	Produce a bug tracking report
		Description: Bug tracking report for Crypto currency project

Bug/error	Solution	Date
Bitcoin cash trading symbol doesn't match symbol, so can't add as asset. 10/05/19	Added Map functionality to fetch to allow mapping to portfolio asset.	10/05/19
router.put doesn't update amount in portfolioGrid (increaseAsset)	Added eventBus handler to increaseAsset to refresh the page.	11/05/19
Charts not rendering in time.	Tried vue mounted but this didn't work. Used vue watcher, and refactoring of forEach method.	13/05/19
Unable to match icon data using the currency symbol.	Changed this to trading symbol which meant it could match the icons id.	13/05/19
calculateHistoricalData method in FetchMethods.js not returning accurate data.	Changed the method to loop around the values returned by the polonium API first, then to loop around the assets in the asset list as this works better.	14/05/19

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

Description: Abstract class Baddie imposes an attack(player) method on its child classes

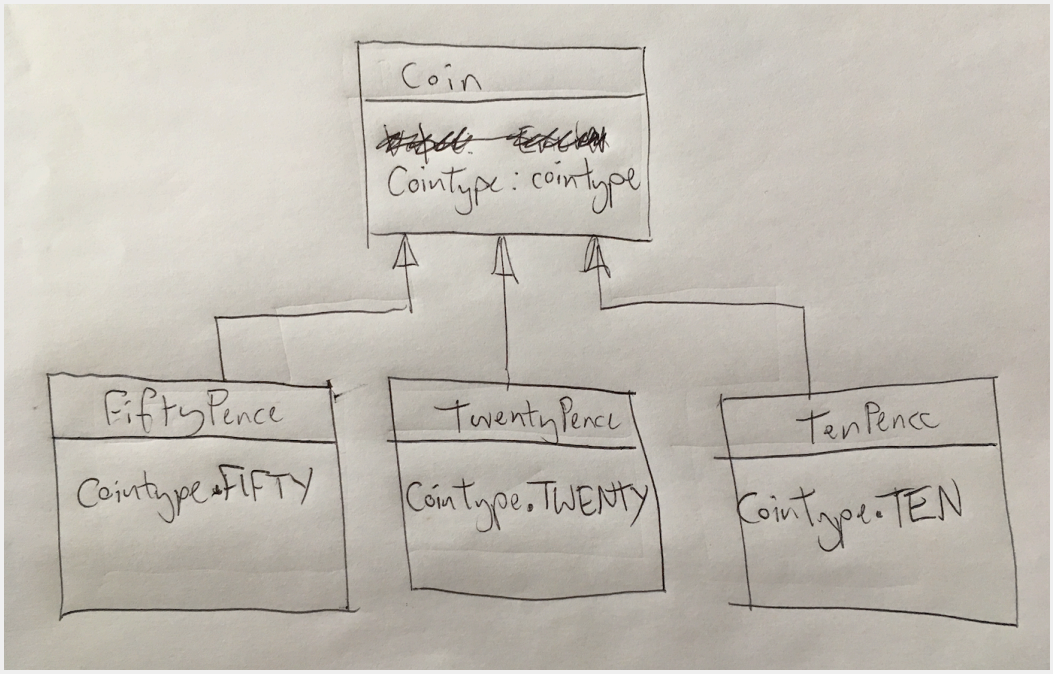
```
public abstract class Baddie {

    private String name;
    private String weapon;
    private int speed;
    private int brutality;
    private int weaponEffectiveness;

    public Baddie(String name, String weapon, int speed, int brutality, int weaponEffectiveness) {
        this.name = name;
        this.weapon = weapon;
        this.speed = speed;
        this.brutality = brutality;
        this.weaponEffectiveness = weaponEffectiveness;
    }

    public String getWeapon() { return this.weapon; }

    public String getName() { return name; }

    public int getSpeed() { return speed; }

    public int getWeaponEffectiveness() { return this.weaponEffectiveness; }

    public int getBrutality() {
        return this.brutality;
    }

    public abstract void attack(Player player);
}
```

Child class RazorGang has attack(player) method inherited from Baddie class. Also has maim(player), bump(player) and stealBottles(player) imposed by interfaces IMaim, IStealBottles and Bump.

```
public class RazorGang extends Baddie implements IMaim, IStealBottles, IBump {

    public RazorGang(String name, String weapon, int speed, int brutality, int weaponEffectiveness) {
        super(name, weapon, speed, brutality, weaponEffectiveness);
    }

    public void maim(Player player) {
        int damage = this.getBrutality() * this.getWeaponEffectiveness() * 10;
        player.decreaseHealth(damage);
    }

    public void bump(Player player) { player.loseGirdLife(); }

    public void stealBottles(Player player) { player.loseAllBottles(); }

    public void attack(Player player){
        this.bump(player);
        this.maim(player);
        this.stealBottles(player);
    }
}
```

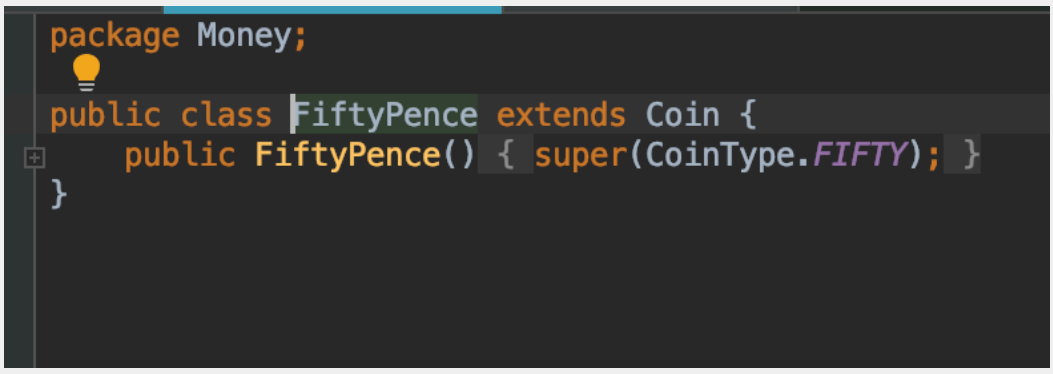
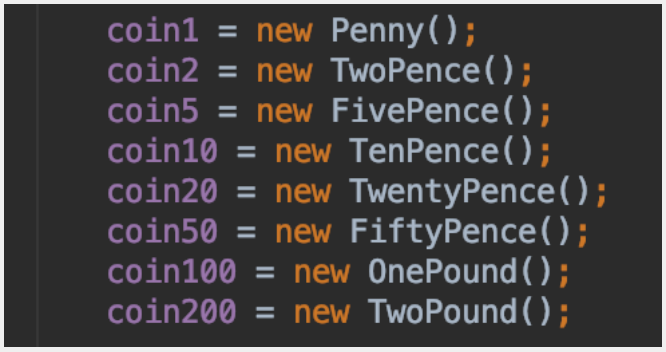
Child class RazorGang has attack(player) method inherited from Baddie class

```
public class Dentist extends Baddie {

    public Dentist(String name, String weapon, int speed, int brutality, int weaponEffectiveness) {
        super(name, weapon, speed, brutality, weaponEffectiveness);
    }

    public void attack(Player player){
        int damage = this.getBrutality() * this.getWeaponEffectiveness() * 10;
        player.decreaseHealth(damage);
    }
}
```

Unit	Ref	Evidence
A&D	A.D.5	An Inheritance Diagram
		Description: FiftyPence, TwentyPence and TenPence classes inherit from Coin class 

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

		Description: Game class in black jack app has private properties deck, players and dealer, which means they can only be accessed with getters and setters.
		
		`public class Game {`
		
		`private Deck deck;`
		`private ArrayList<Player> players;`
		`private Dealer dealer;`
		
		`public Game(Dealer dealer, Deck deck){`
		`this.players = new ArrayList<Player>();`
		`this.dealer = dealer;`
		`this.deck = deck;`
		`}`

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: Coin class  **FiftyPence class inherits from Coin class**  **Coin50 is an object of the FiftyPence class** 

```
package Money;
public abstract class Coin {

    private CoinType coinType;

    public Coin(CoinType coinType) { this.coinType = coinType; }

    public CoinType getCoinType() { return coinType; }

    public int getCoinValue() { return this.coinType.getValue(); }
}
```

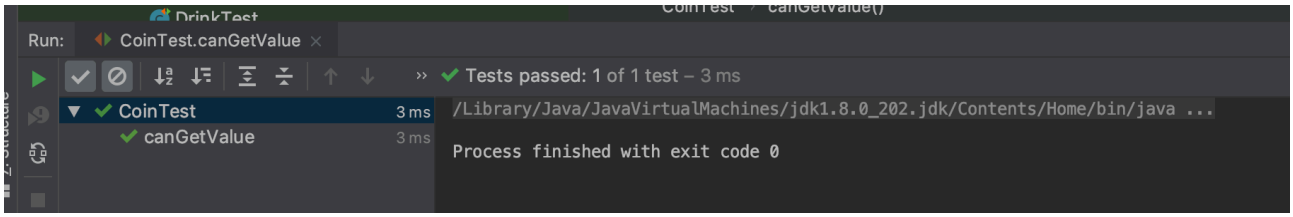
```
package Money;
public class FiftyPence extends Coin {
    public FiftyPence() { super(CoinType.FIFTY); }
}
```

```
    coin1 = new Penny();
    coin2 = new TwoPence();
    coin5 = new FivePence();
    coin10 = new TenPence();
    coin20 = new TwentyPence();
    coin50 = new FiftyPence();
    coin100 = new OnePound();
    coin200 = new TwoPound();
```

Tests getCoinValue() method inherited from Coin class works for FiftyPence coin

```java
@Before
public void setUp() {
    coin = new FiftyPence();
}

@Test
public void canGetValue() {
    assertEquals(50, coin.getCoinValue());
}
}
```

Test passes



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: Algorithm in method for vending machine to return the correct change. % 100 gives the remainder after dividing the change amount by 100 to determine the number of pound coins in the change. I chose it because it is simple but effective.
		`int changeLeft = change % 100;`
		Algorithm for calculating the damage done when a baddy with a given brutality attacks with a weapon of given effectiveness. I chose it because of its elegant efficiency.
		`int damage = this.getBrutality() * this.getWeaponEffectiveness() * 10;`