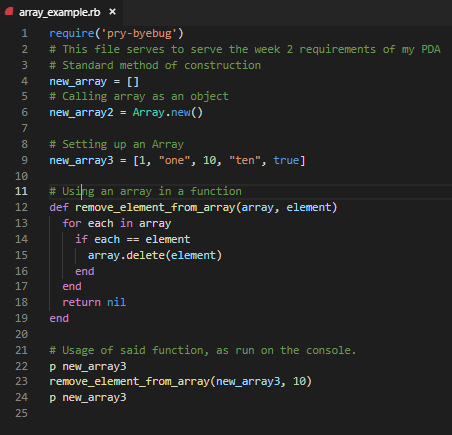
**Evidence Gathering Document for SQA Level 8 Professional Developer Award.**

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.

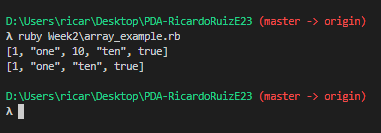
Each point that required details the Assessment Criteria (What you have to show) along with a brief description of the kind of things you should be showing.

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

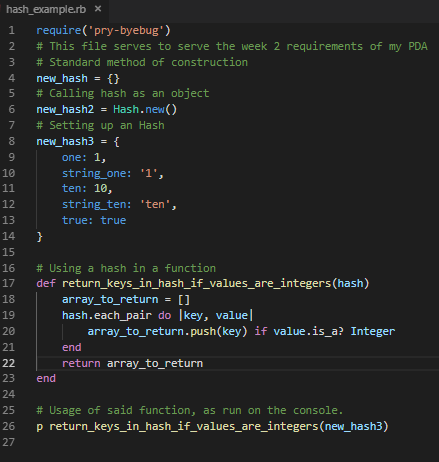
**Week 2**



| 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 | |



* Lines 4, 6 and 9 are arrays in a program
* Line 12 is a function that removes an element from the array based on the element itself
* Second screenshot shows the array before and after the function

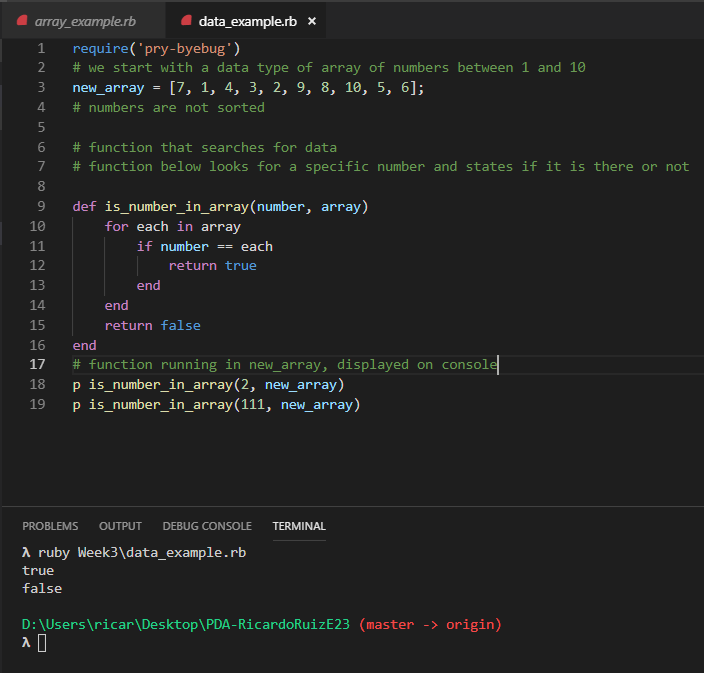


| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.6 | Demonstrate the use of a hash in a program. Take screenshots of:  \*A hash in a program  \*A function that uses the hash  \*The result of the function running | |

* Lines 4, 6 and 8 are hashes in a program
* Line 17 is a function builds a new array and returns it filled with keys from the hash if the elements are integers
* Second screenshot the returned array from the hash

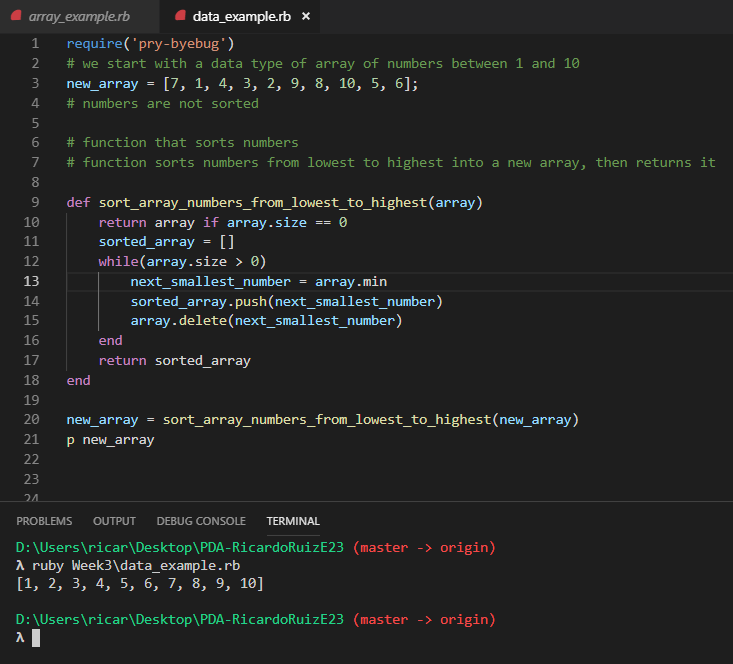
**Week 3**

| 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 | |



* An array is created on line 3
* Function on line 9 iterates through the array checking each number until it comes across any instance of the desired number, or until it iterates through the entire array
* Console prints out true or false to that function

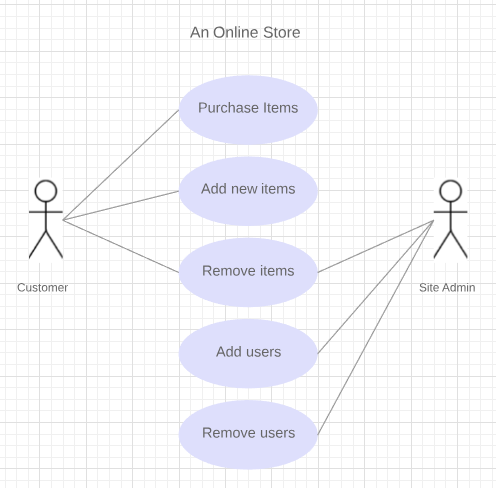
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.4 | Demonstrate sorting data in a program. Take screenshots of:  \*Function that sorts data  \*The result of the function running | |



* Array is created on line 3
* Function on line 9 creates a temporary array, and pushes the lowest number on the unsorted array to itself.
* New array was set to the returned value, then printed.

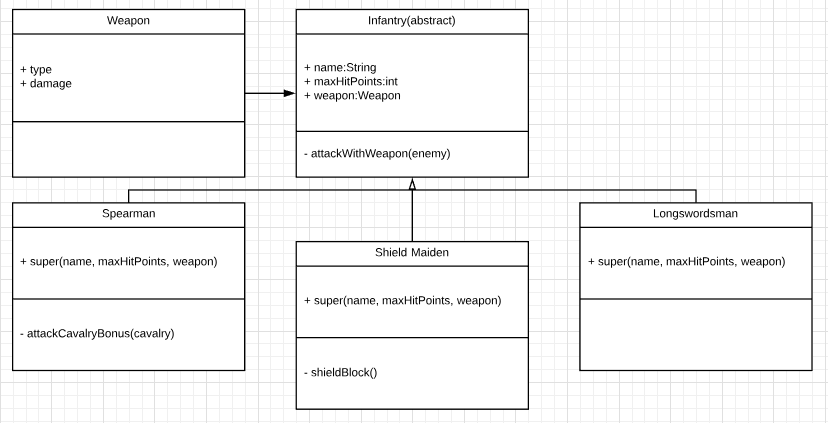
**Week 5**

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



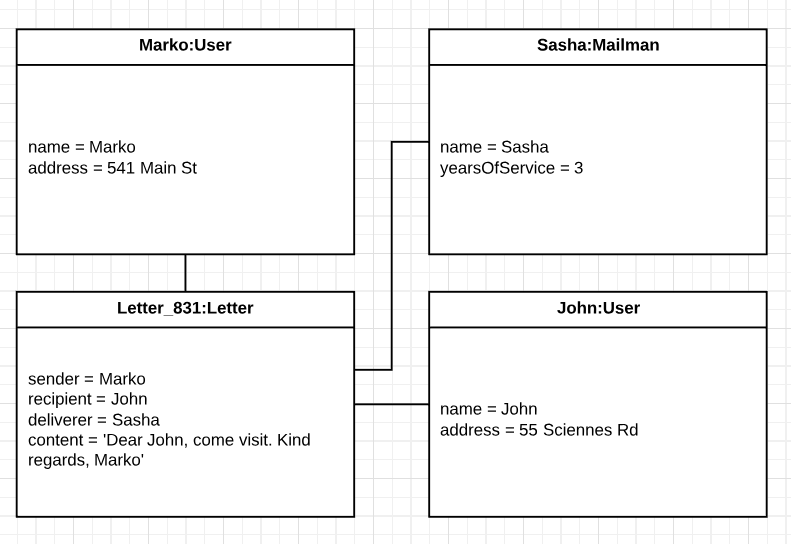
A use case diagram of an online store concept.

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



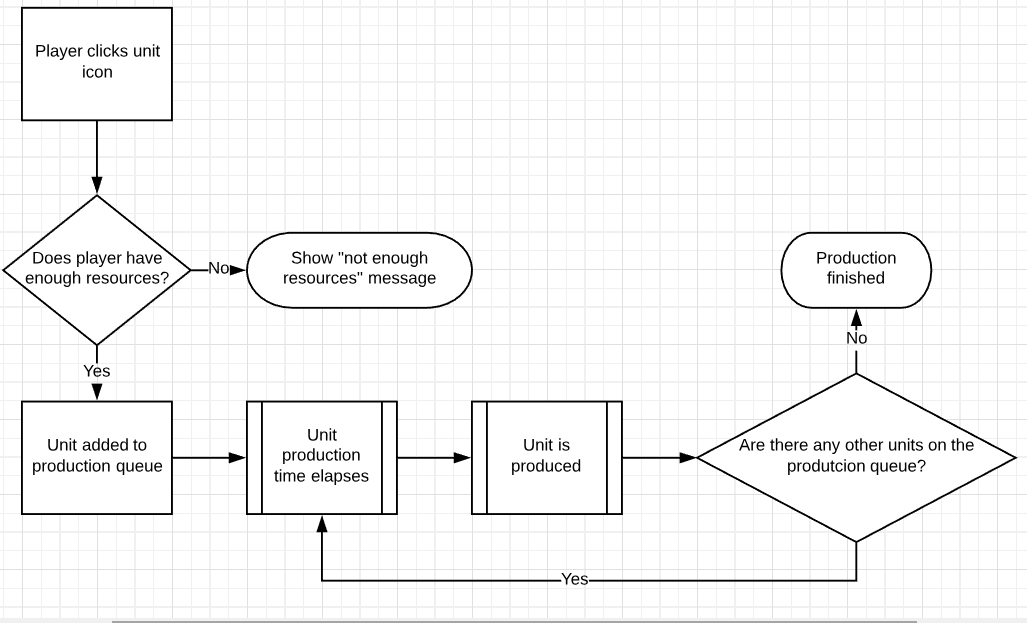
A class diagram concept on an infantry type unit on a strategy videogame.

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



Concept of a letter being sent, with recipient and sender

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



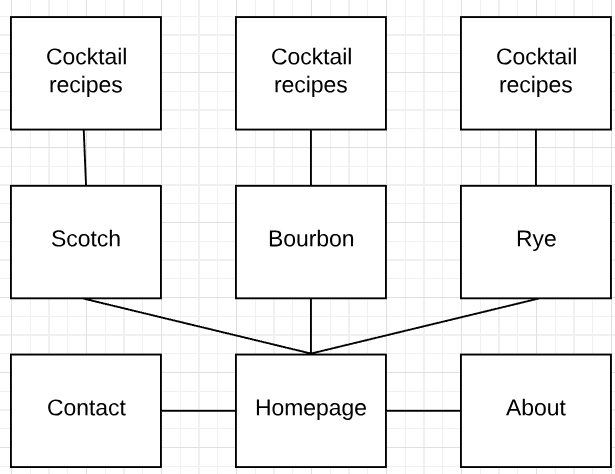
Activity diagram of how unit production would be on a strategy videogame

| 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 | |

|  |  |  |
| --- | --- | --- |
|  | **Implementation constraint** | **Solution** |
| **Hardware and software platforms** | Software not compatible with mobile devices, being a web application.  Software should be made native on mobile devices to allow younger users to use it without having to access a browser.  Users that are accustomed to using mobile applications are discouraged from using the software. | Using a library such as React Native to allow the current stack to be used as a mobile application. |
| **Performance requirements** | Application does not have enums for its iterations.  Older computers might have a problem fetching information from the database.  This is an issue because currently the code is inefficient | Refactor code to implement enums for better iteration. |
| **Persistent storage and transactions** | Current database has many to many relationships.  Software should have one to many and many to one relationships with joint tables.  This is an issue due to not being a best practice, harming security | Refactor code and refactor database to include joint tables. |
| **Usability** | Software currently loads assets progressively. Since colours clash, there is a small moment in which the window has strobing colours, which could affect people that suffer from epilepsy. | Code can be refactored to preload assets so that loading times are slower, but assets are loaded all at once |
| **Budgets** | Application is free for the user, and funded by the National Healthcare Service but made by an outsourced company. Considering that the app is not maintained by the NHS, an increase in the number of users could place a strain in the resources used.  This could make it harder for users to connect and use the app | A meeting with the NHS to better explain the requirements to maintain the app might be in order. Also, a possibility to place advertisements on the application should not be completely off the table. |
| **Time** | The application had a longer-than-expected planning phase. This is going to place a constraint in the later stages. This could mean that not enough word will be put on certain features | Two solutions are possible: a meeting to request stakeholders for longer time to implement the solution, or the reduction of the number of planned features. |

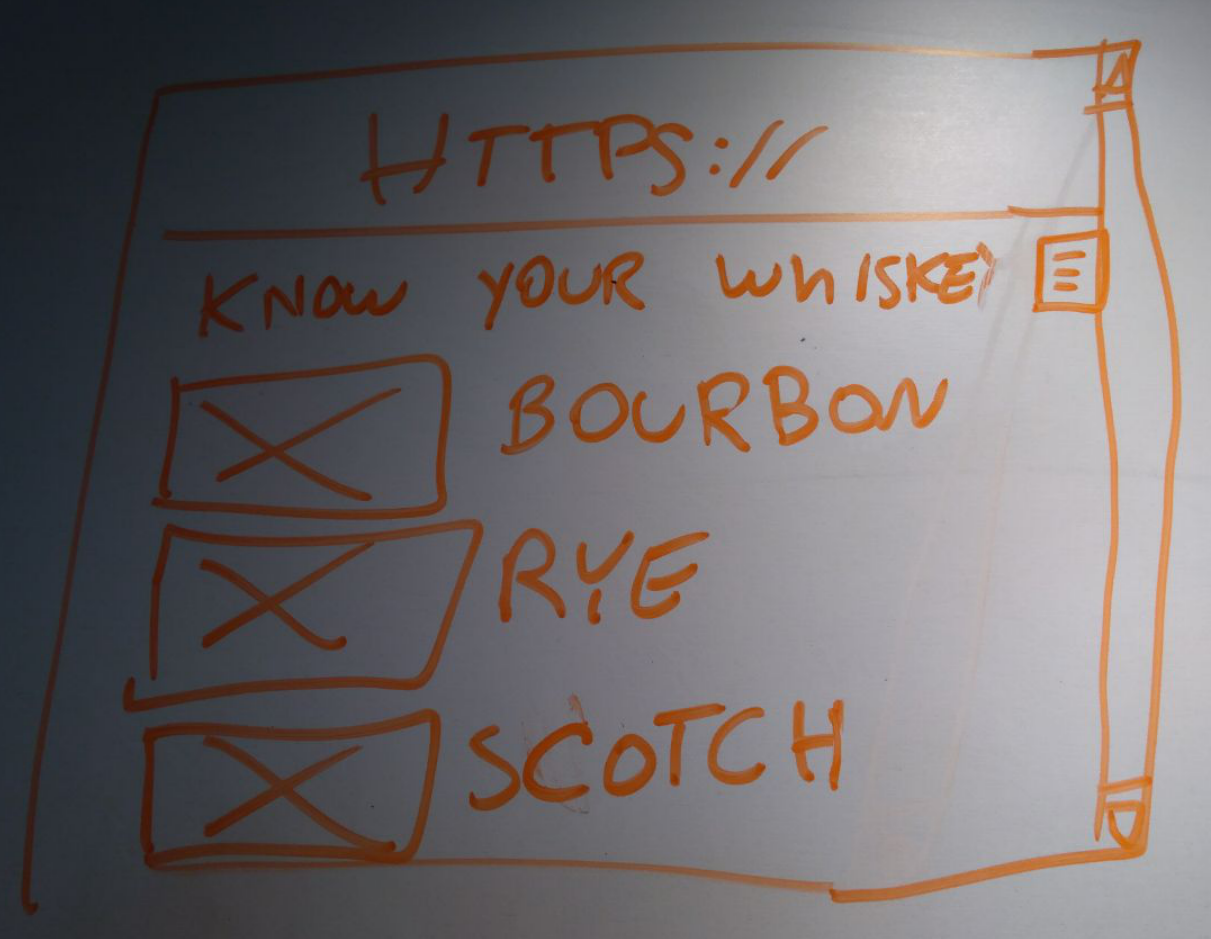
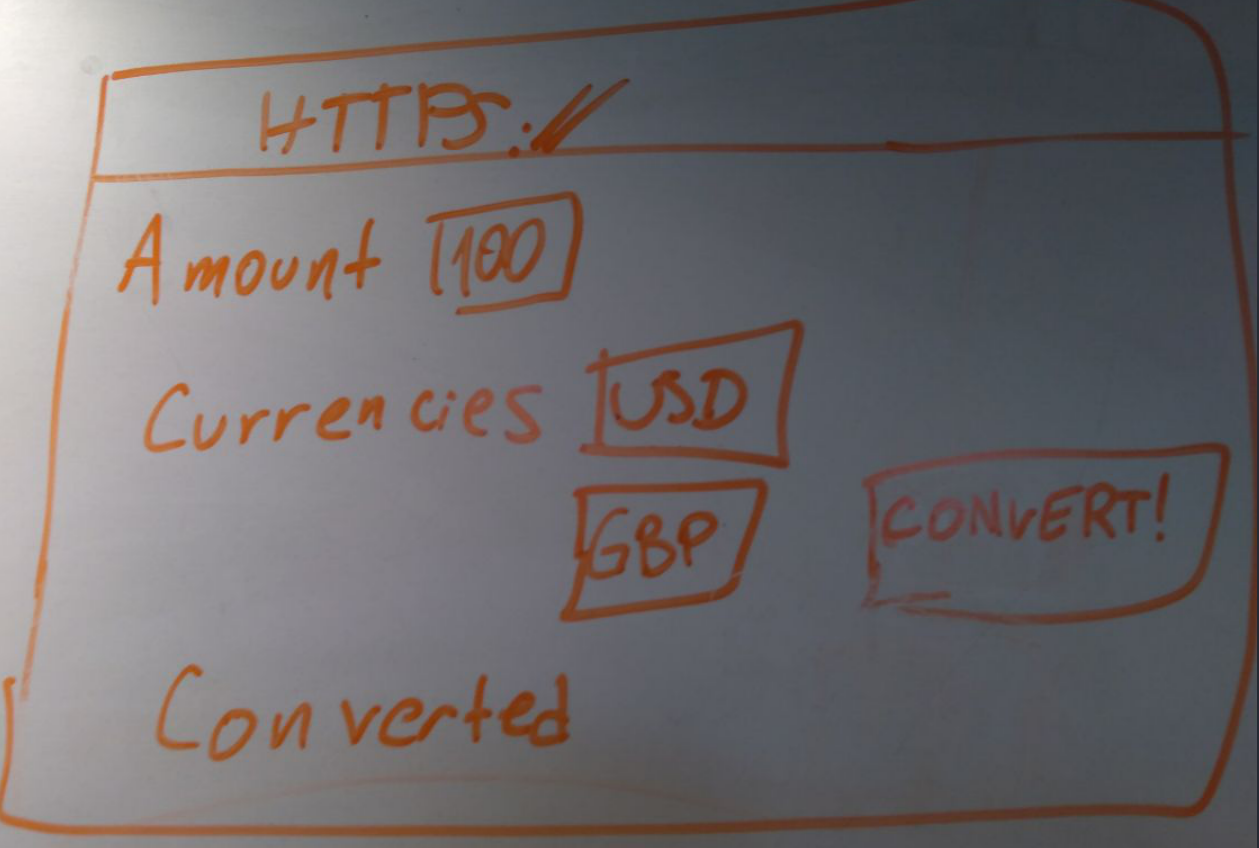
Implementation Constraints Plan based on a medical scheduling software

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.5 | User Site Map | |



Sitemap of a whisky trivia website

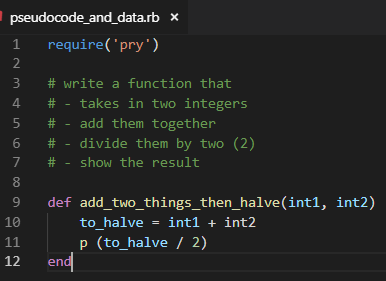
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.6 | 2 Wireframe Diagrams | |



First wireframe for a whisky trivia website. Has images and headers, expected scrollbar and menu on the side.

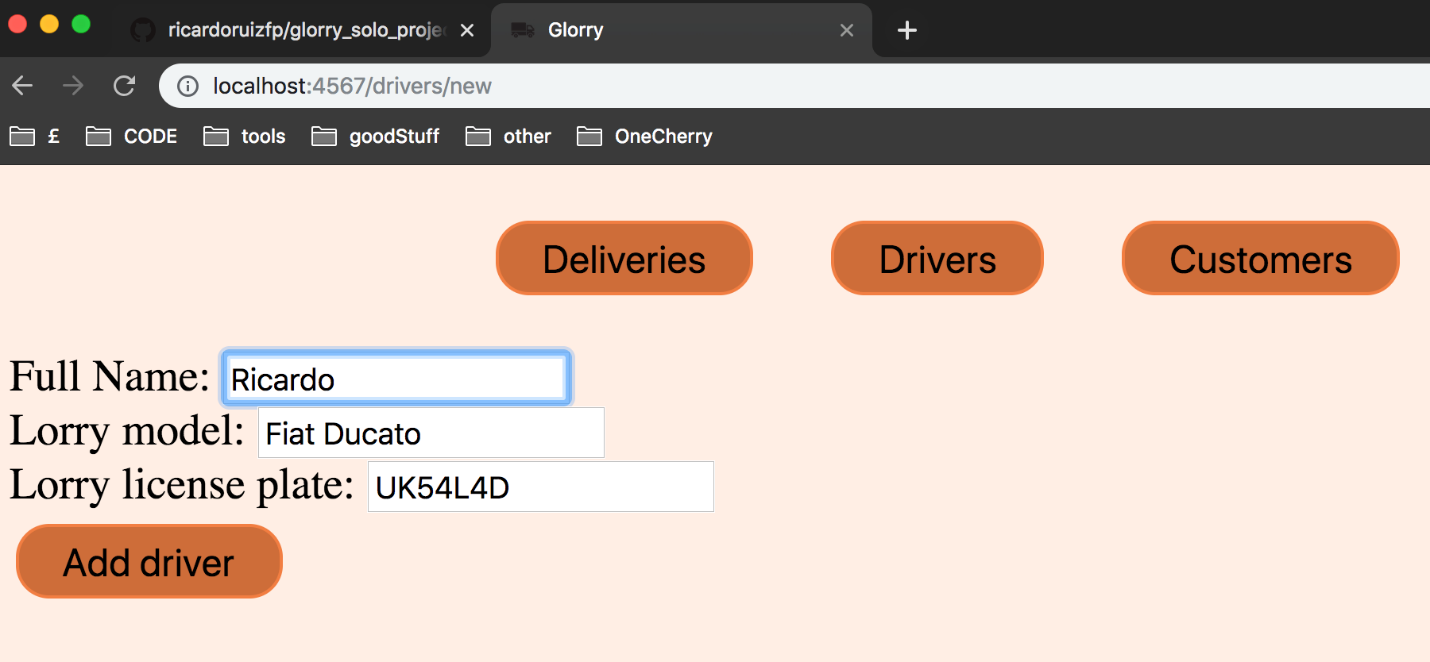
Second wireframe for a currency exchange website.

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

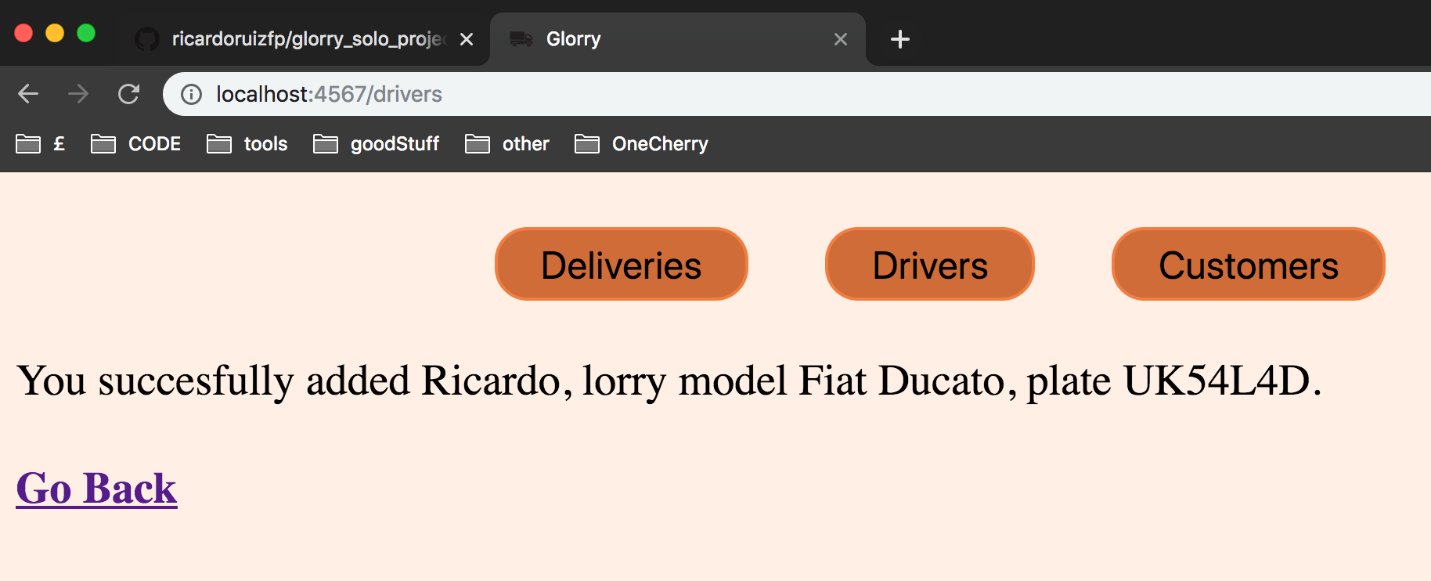


Pseudocode denotes the requirements of the method, then method is created

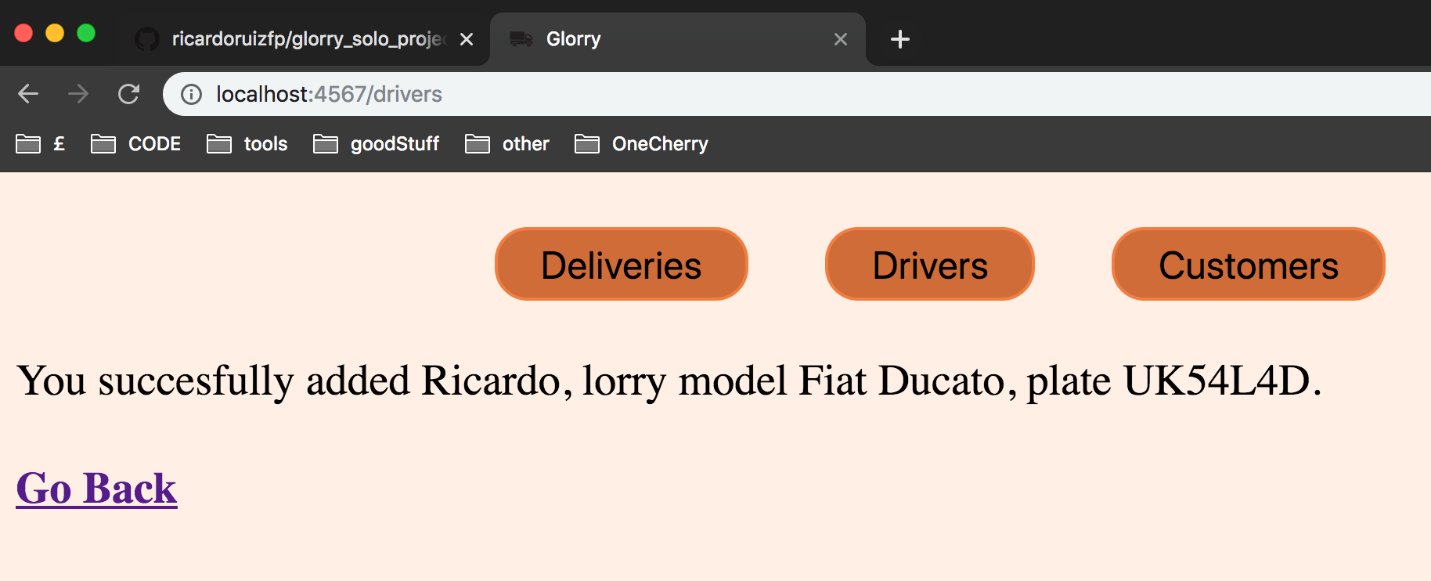
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.13 | Show user input being processed according to design requirements. Take a screenshot of:  \* The user inputting something into your program  \* The user input being saved or used in some way | |



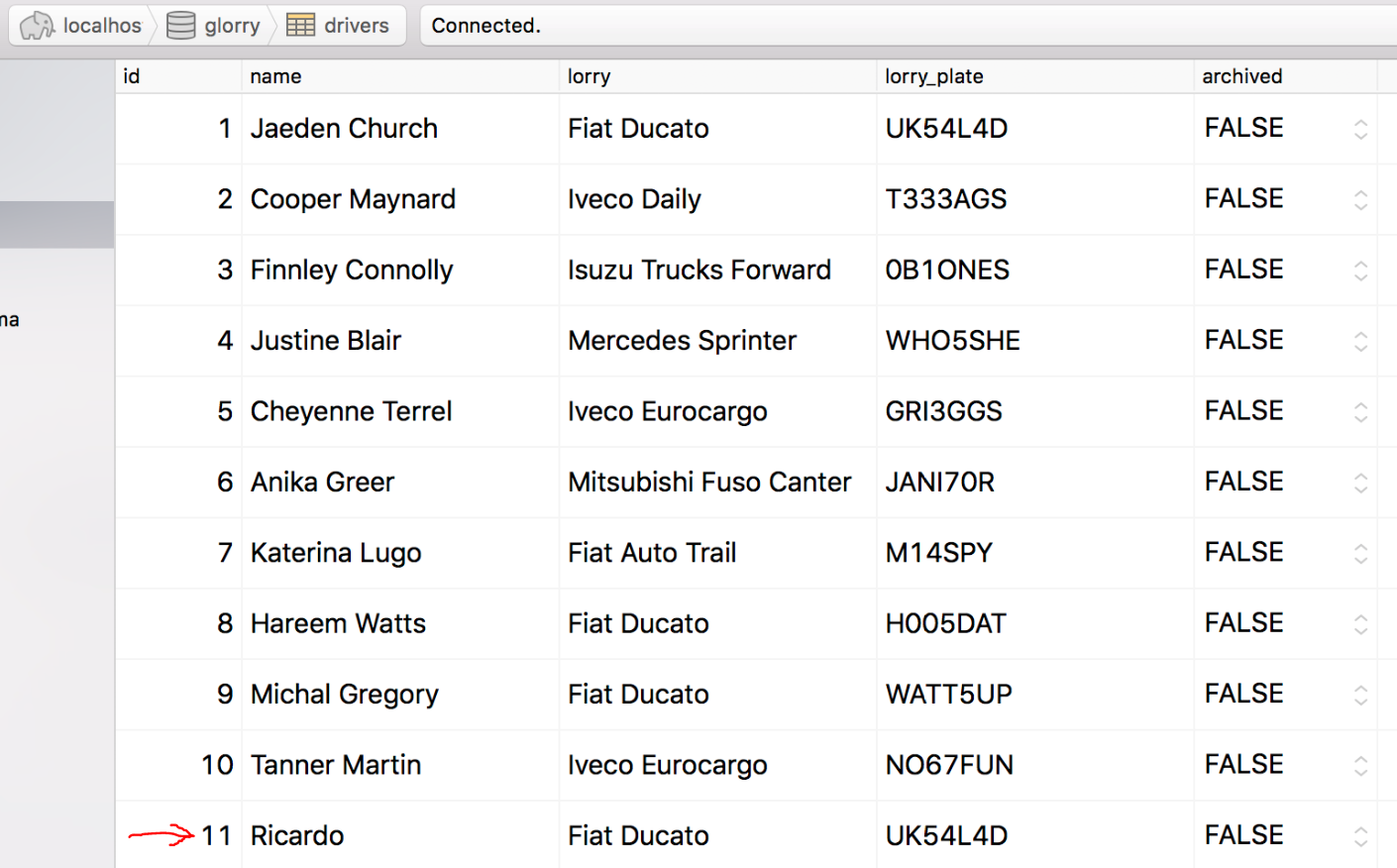
User inputs name of driver, model of the lorry of that driver and its license plate

Information that user inputted is saved on the database

| 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 | |

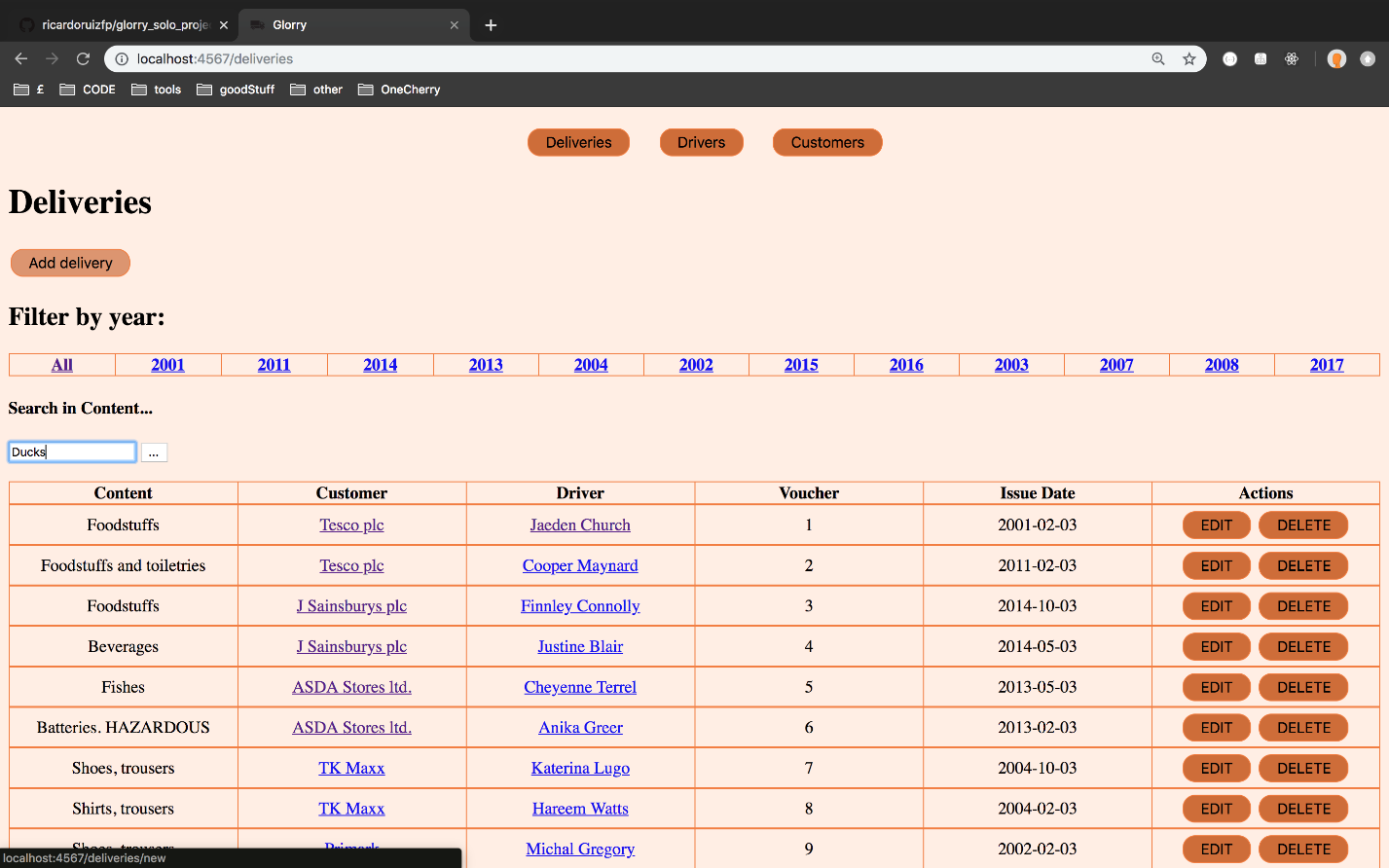


From the previous evidence, user input is saved on database

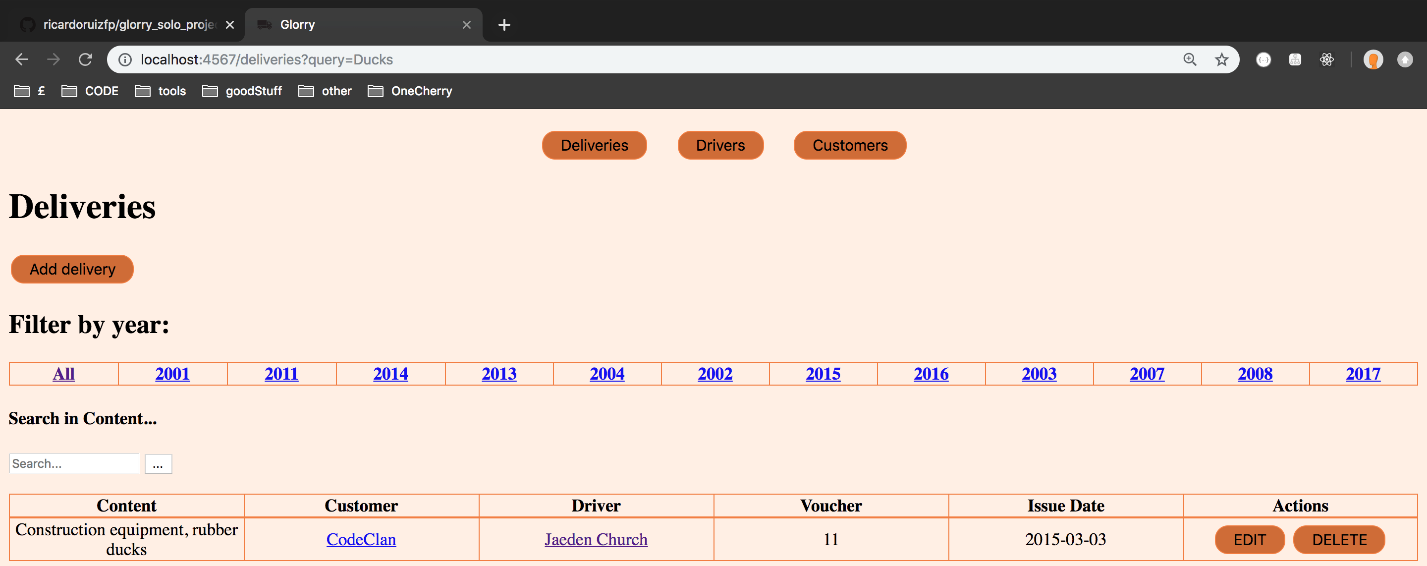


Driver entry saved on the database

| 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 | |

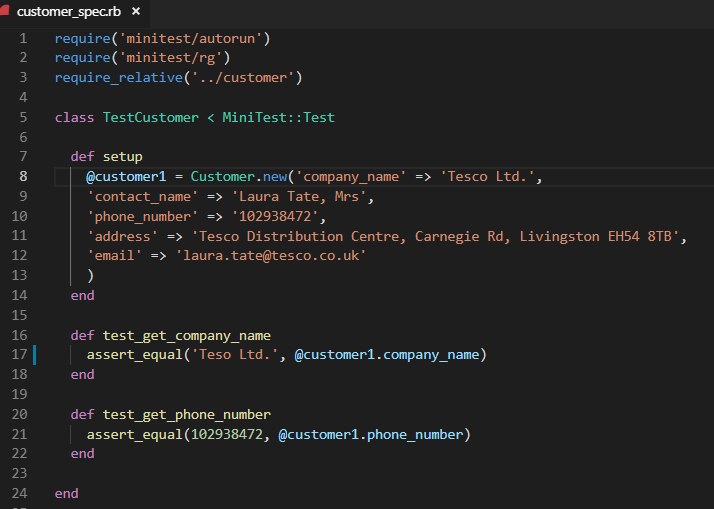


User searches for ducks on the delivery list

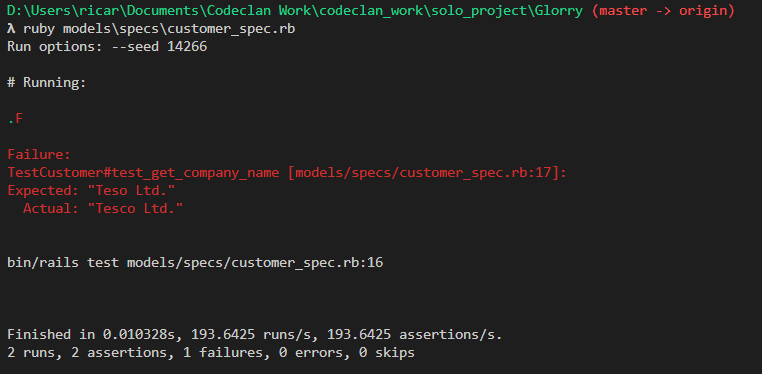


User gets filtered list with only ducks

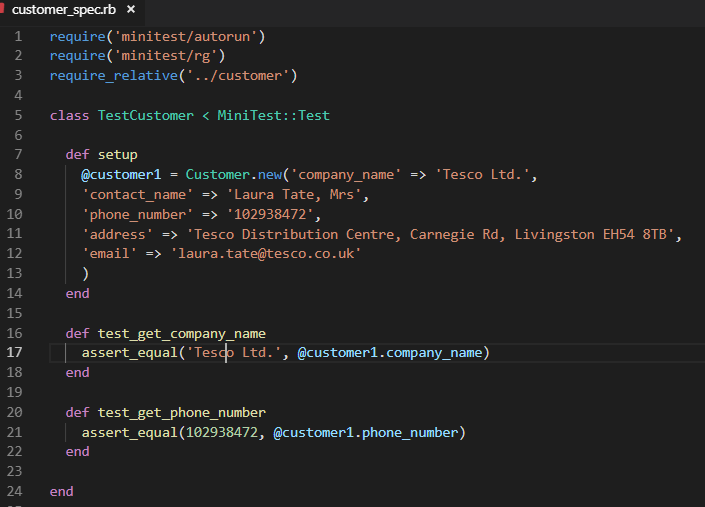
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.18 | Demonstrate testing in your program. Take screenshots of:  \* Example of test code  \* The test code failing to pass  \* Example of the test code once errors have been corrected  \* The test code passing | |



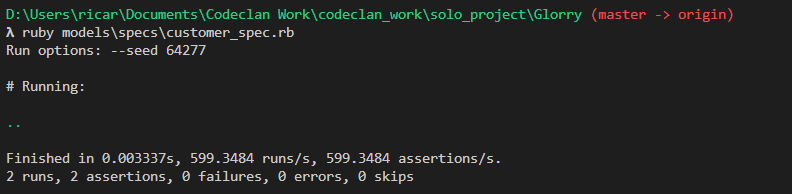
Example of test code



Test code failing



Fixed test code, on line 17 the name Tesco was written wrong

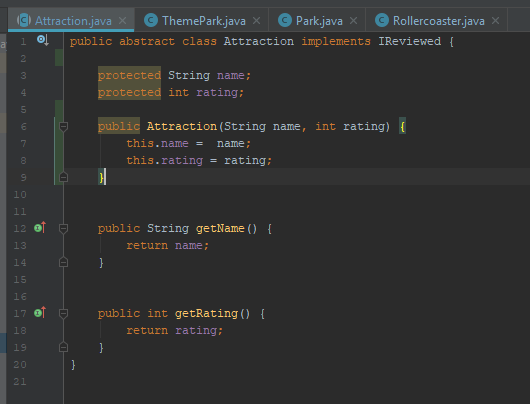
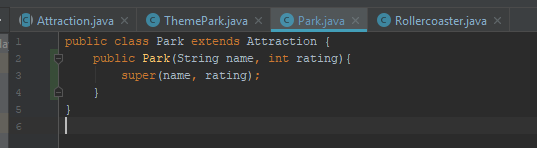
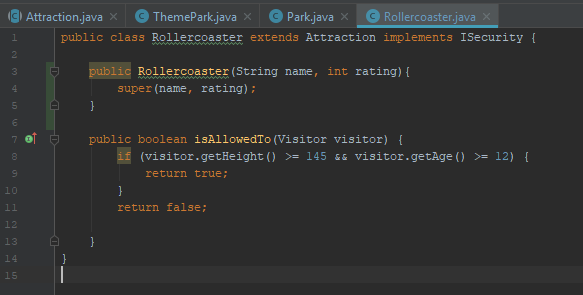


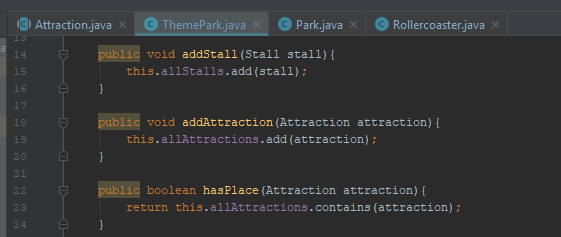
Now all tests are passing

**Description here**

**Week 7**

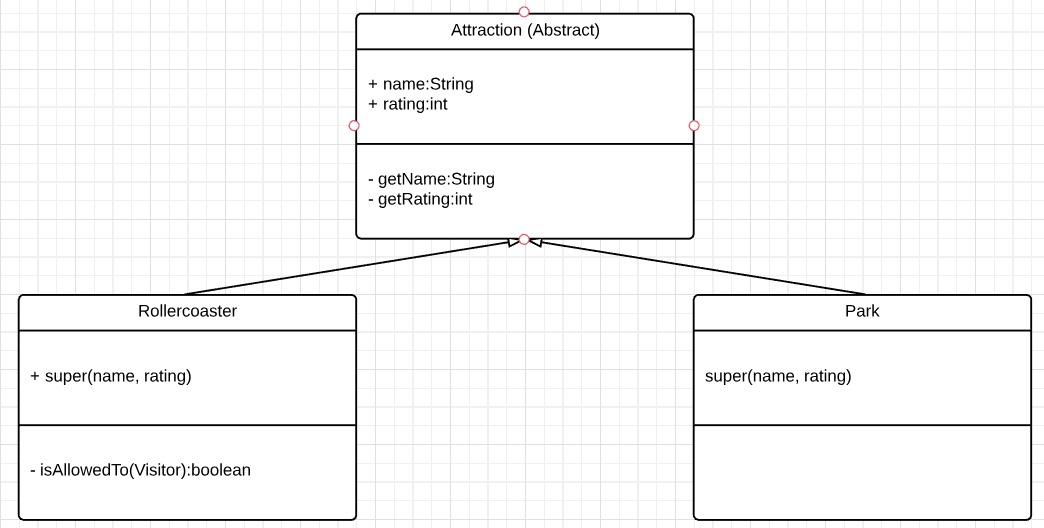
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.7 | The use of Polymorphism in a program and what it is doing. | |





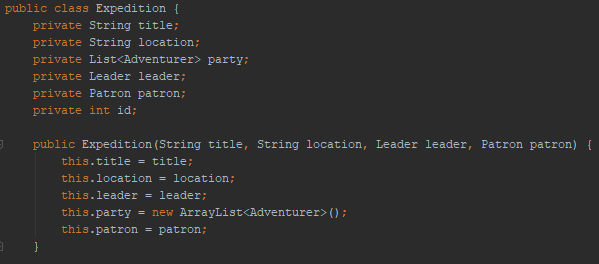
Attraction is an abstract class. Park and Rollercoaster are Attractions, therefore they extend (inherit) over Attractions. ThemePark has the methods addAttraction and hasPlace. Both can take either a Park instance or a Rollercoaster instance, seeing as they are both Attractions, but cannot take an Attraction instance, because Attracion, being abstract, can never be instantiated.

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



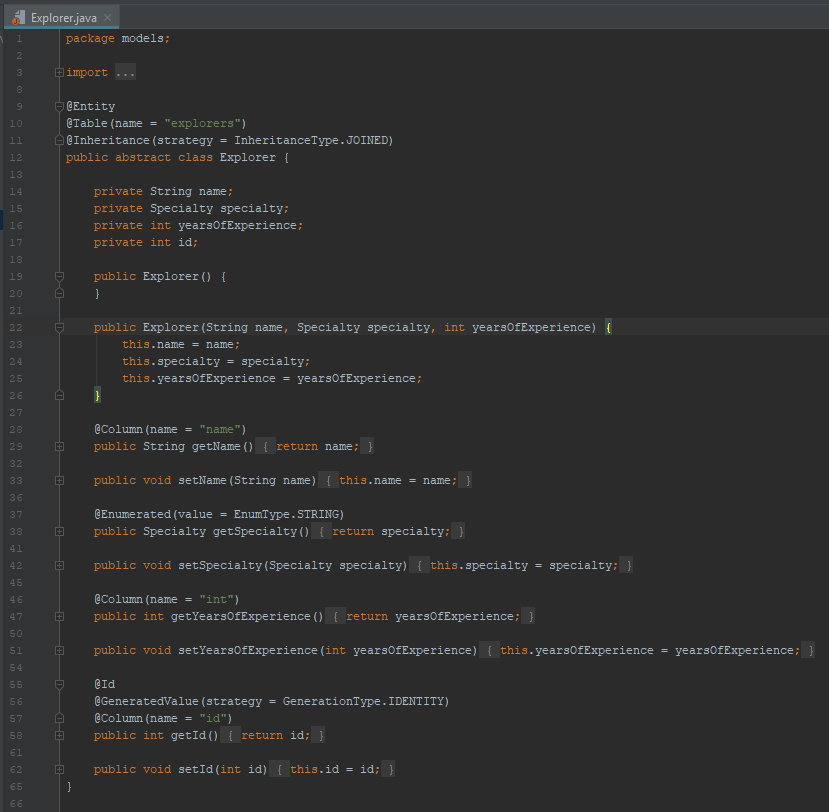
Rollercoaster and Park are classes that inherit from the Attraction superclass

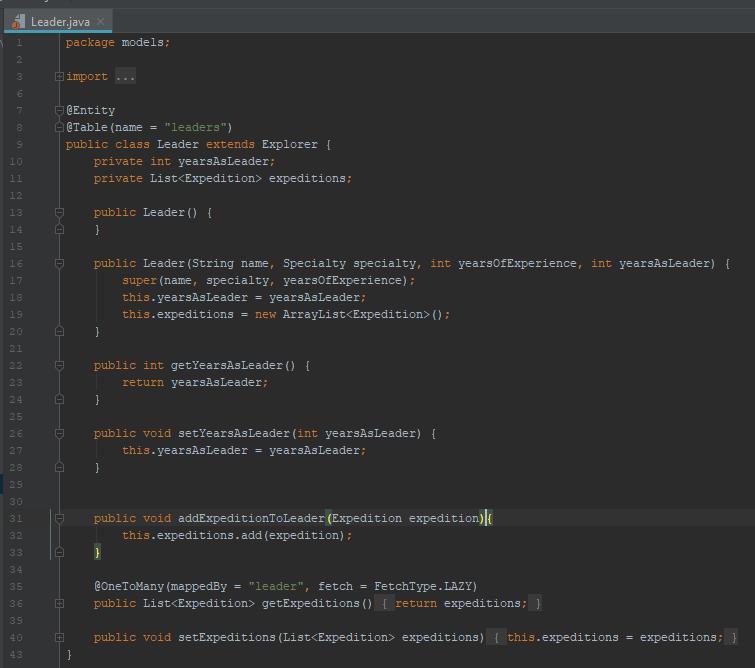
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.1 | The use of Encapsulation in a program and what it is doing. | |



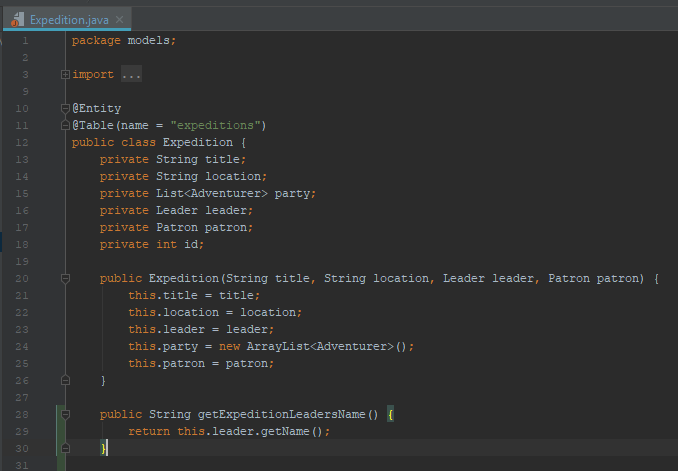
Expedition takes in a Leader and a Patron, two concrete classes that are used in the Expedition instance as private properties, for further use in methods.

| 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. | |





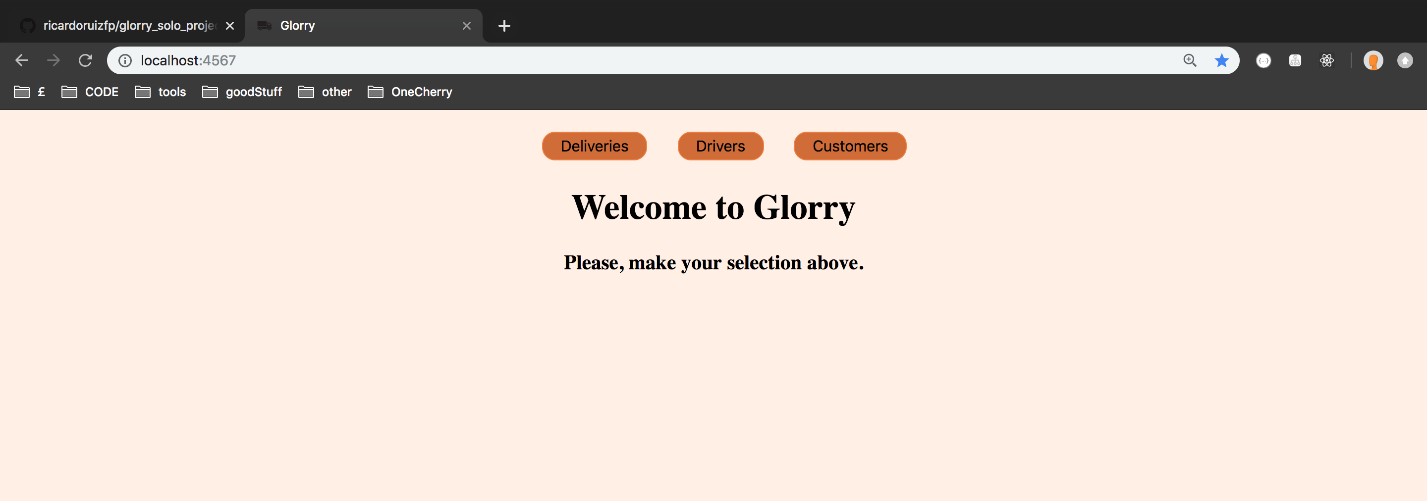




The first screenshot is the abstract class Explorer. The second screenshot is the Leader class, that inherits from explorer (another nondisclosed class that inherits from Explorer is Adventurer). The third screenshot is an instance of the class Leader. The fourth screenshot is the a method in the class Expedition that returns the name of that expedition’s leader, name being a property of the Explorer class, inherited to Leader.

**Week 10**

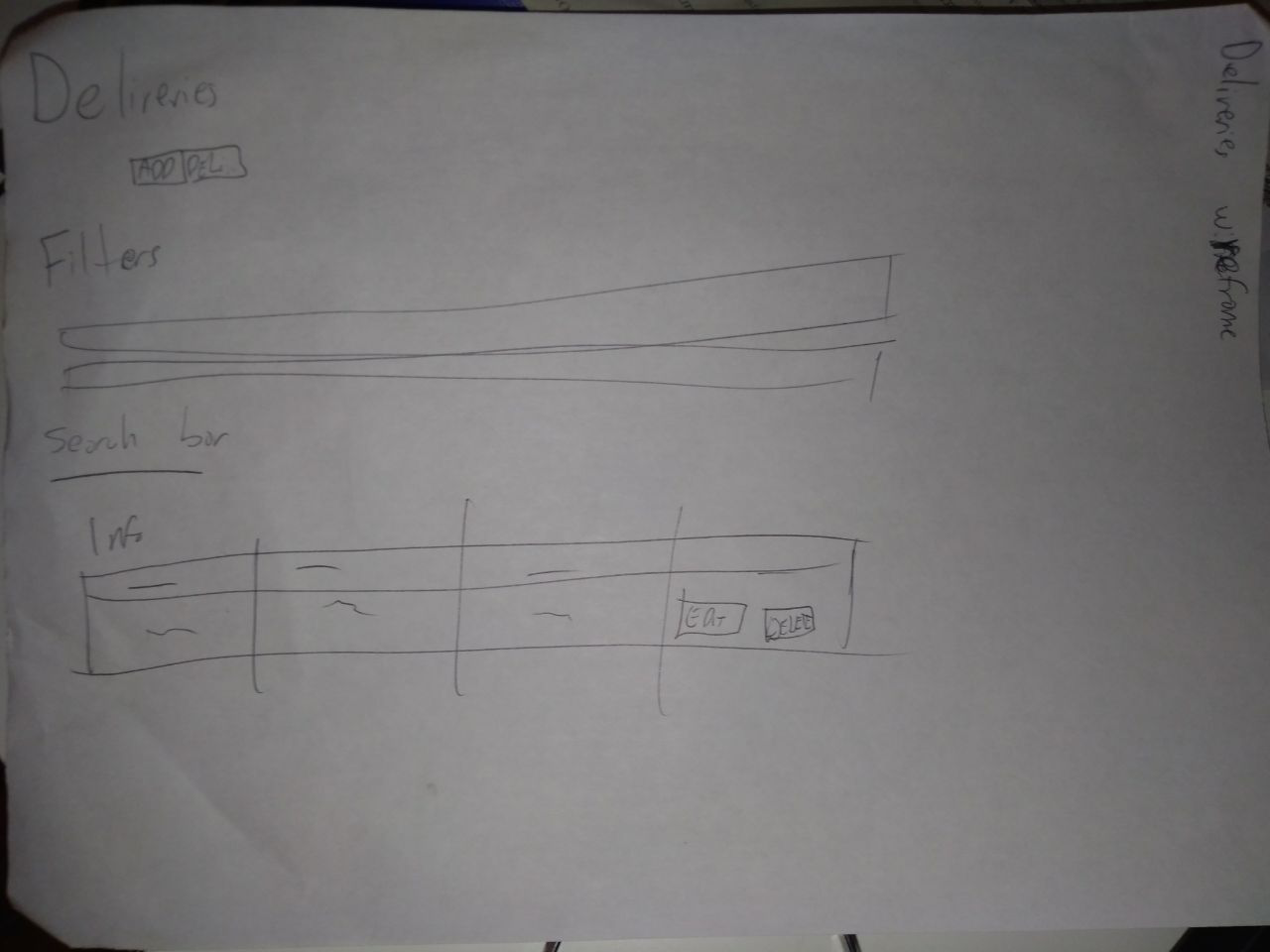
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.11 | Take a screenshot of one of your projects where you have worked alone and attach the Github link. | |

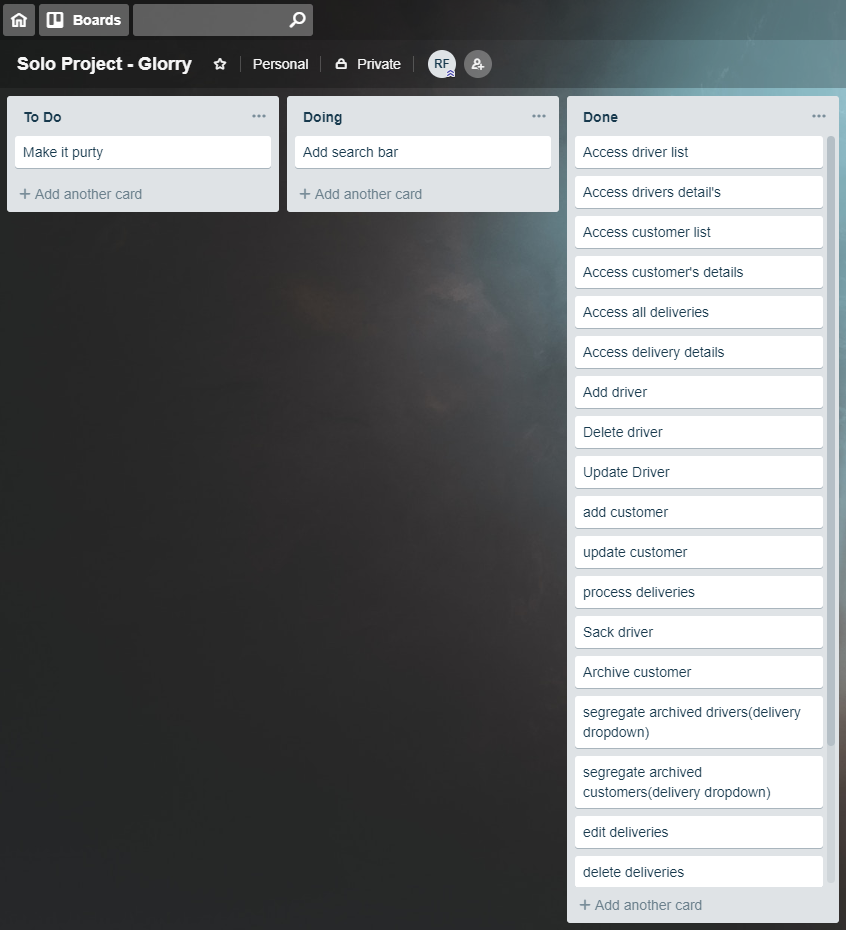


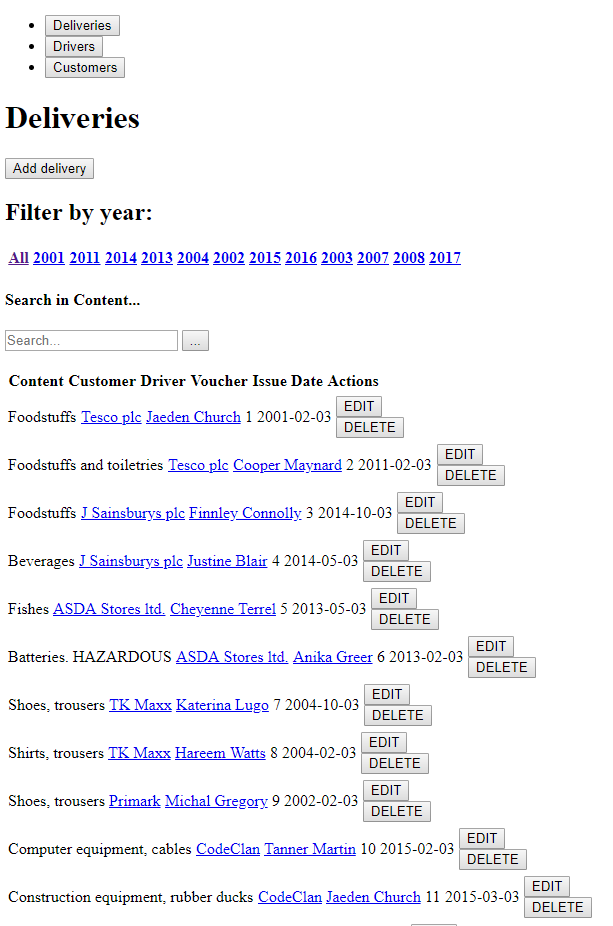
[**https://github.com/ricardoruizfp/glorry\_solo\_project**](https://github.com/ricardoruizfp/glorry_solo_project)

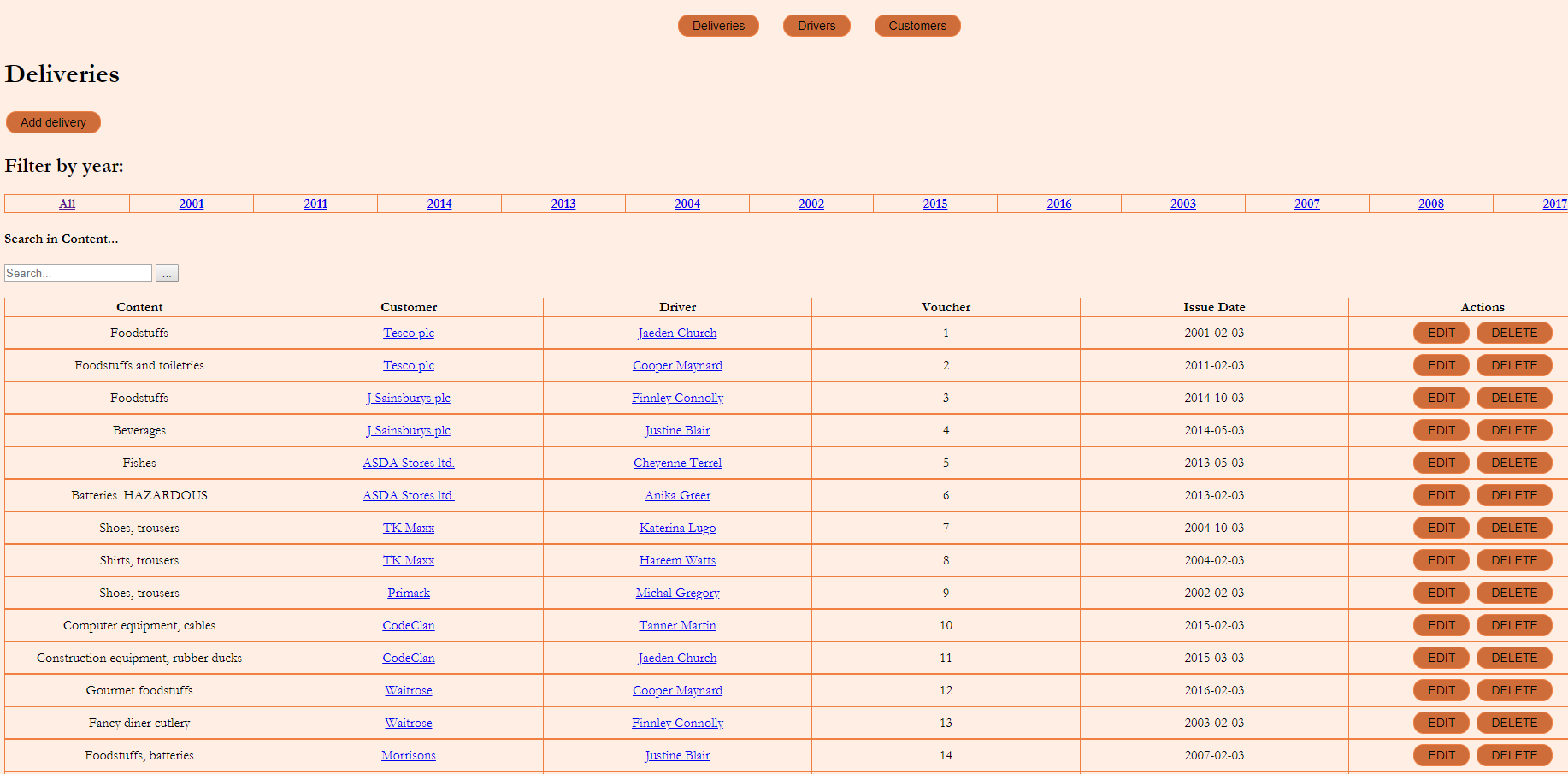
This is a proof of concept of an application designed for logistics managers to manage a drivers, deliveries and customers.

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

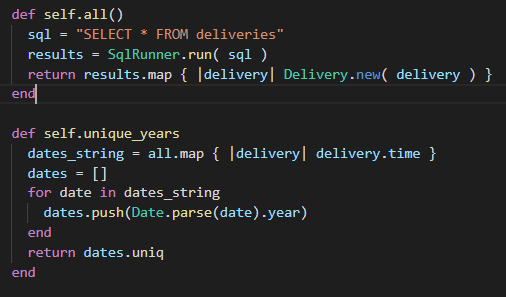
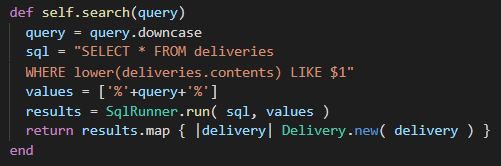








First screenshot is of my wireframe for the deliveries page of my application. The second is of my Trello board, that I used to manage my to-dos in a one-person Kanban way. The third and fourth are my deliveries page before and after I implemented the CSS, including the proper wrapping and formatting of tabular data.



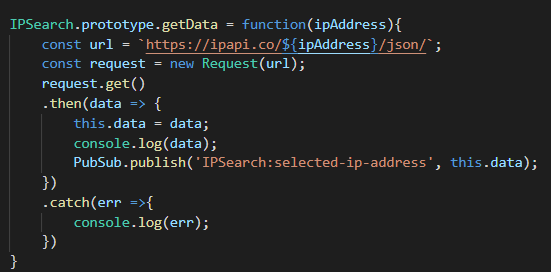
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.9 | Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms. | |

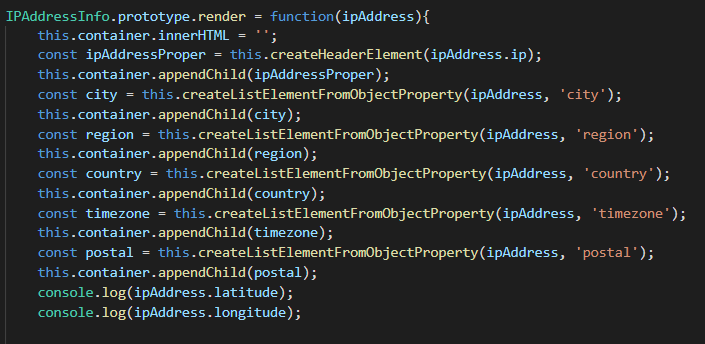
The first pair of algorithms are a query to get all deliveries in the database, and then, since deliveries have an issue date, all unique years for those dates. The second algorithm is a search function, that takes in a string and fetches all deliveries in the database with that string in its name.

I used those algorithms to provide the user with a search and filter functionality, so they can have an easier time trying to filter the deliveries inserted on the database

**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 | |



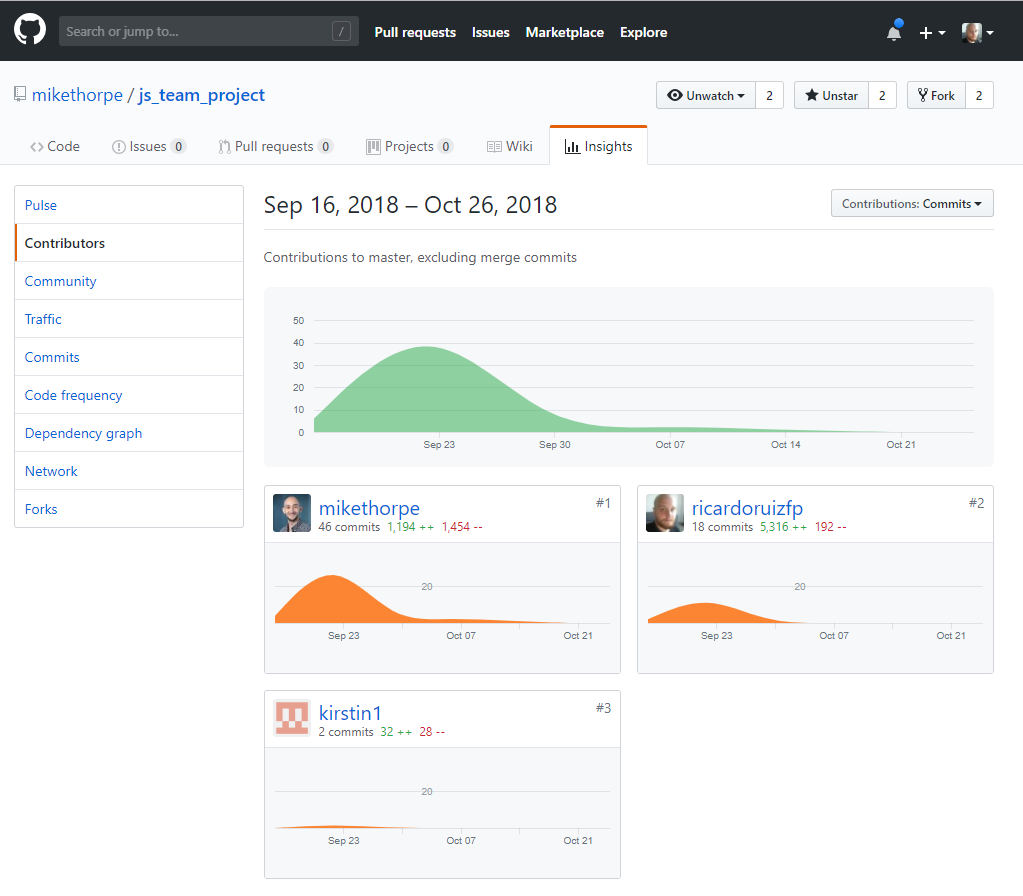




The program fetches information from the API ipapi, which returns a json file with information regarding the ip address provided, Then it renders information of that API, as listed.

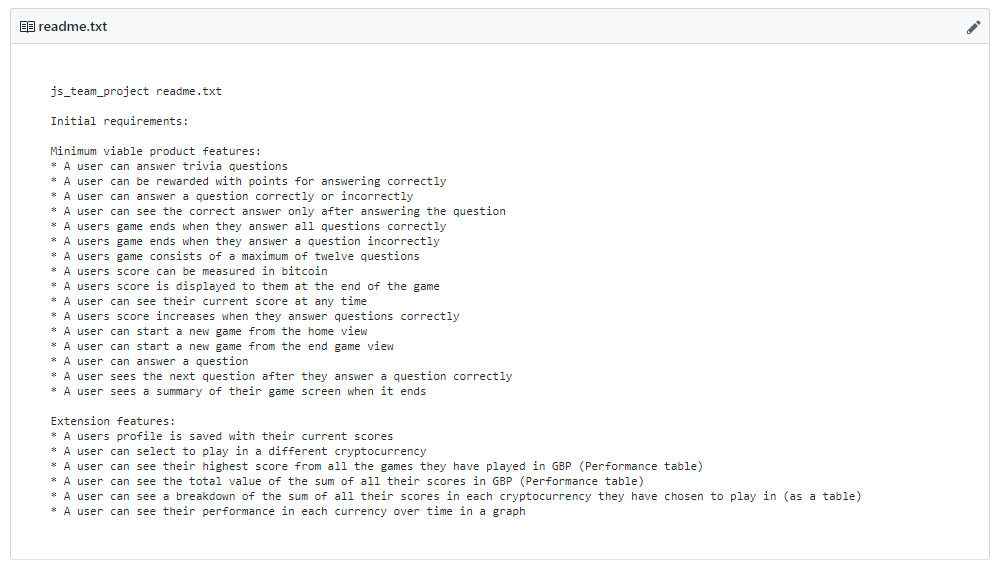
**Week 15**

| 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. | |



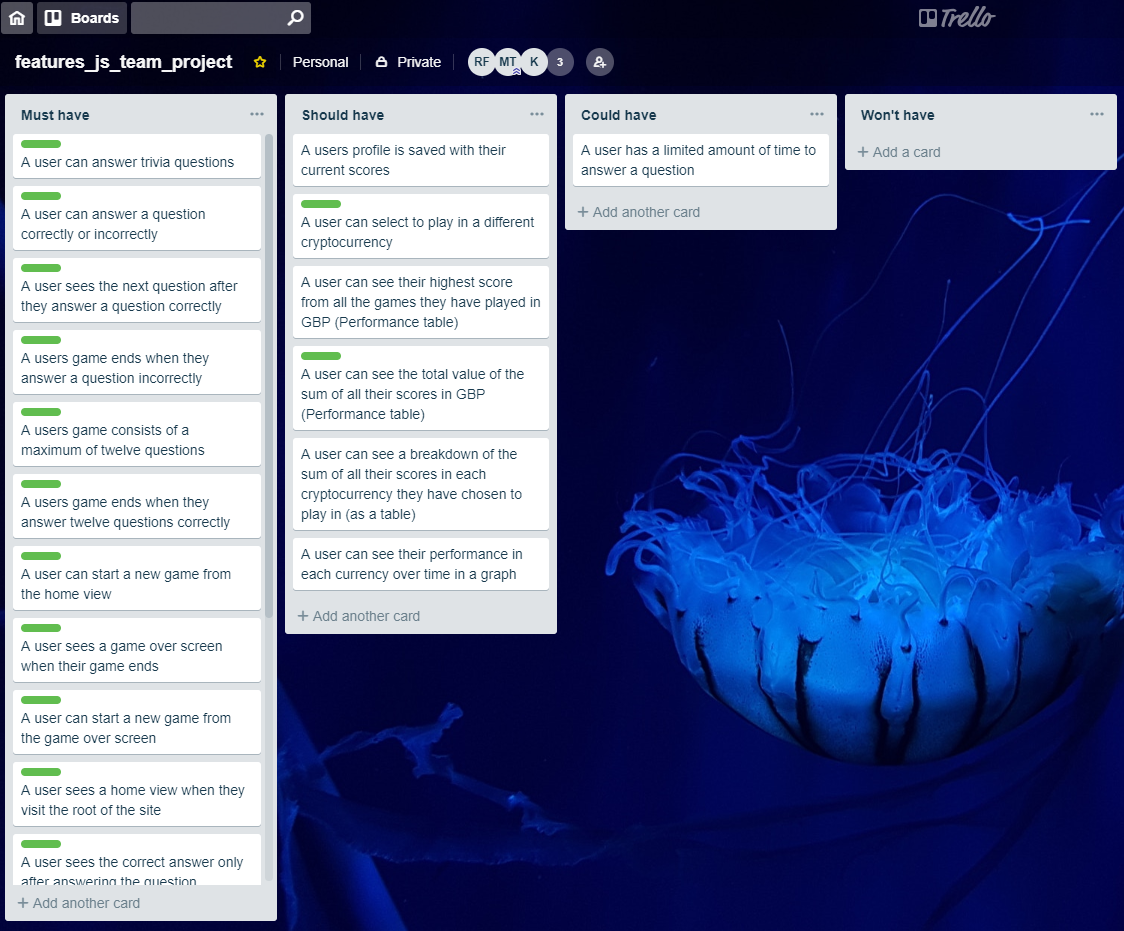
The contributor page of our javascript group project.

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



Our project brief was slightly adapted to be a browser game, but taking information from two different apis.

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



This was our MoSCoW board for our group project.

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

|  |  |  |
| --- | --- | --- |
| **Acceptance Criteria** | **Expected result / output** | **Pass/Fail** |
| A user is able to add a new driver to the database | Another driver is added to the driver’s list view, another driver row is added to the database. | Pass |
| A user is able to add a new customer to the database | Another driver is added to the customer list view, another customer row is added to the database. | Pass |
| A user is able to add a new delivery to the database | Another delivery is added to the delivery view, another row is added to the database. Database entry joins driver and customer. Driver/customer information is retrievable from the individual delivery view. | Pass |

Acceptance criteria for a logistics backoffice software.

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

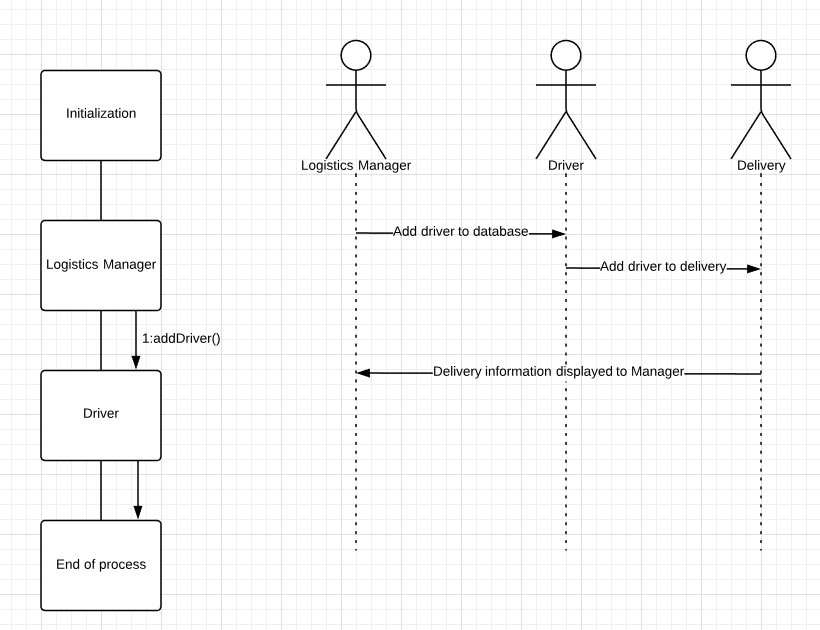
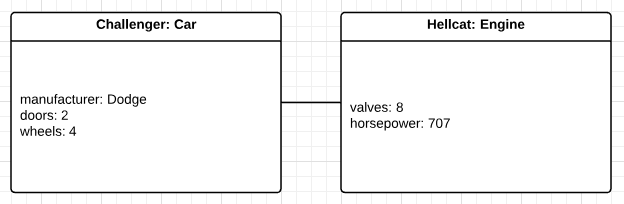


Diagram on the left is a sequence diagram of a manager using a logistics backoffice program to add a new driver to the database. Diagram on the right is a collaborative diagram of a manager using a logistics backoffice program to assign a driver to a delivery.

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





First diagram shows a car object in relation with an engine object. Second shows the relations between a delivery, driver and customer objects.

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

|  |  |  |
| --- | --- | --- |
| **Bug / Error** | **Solution** | **Date** |
| User is able to add new deliveries to customers they do not have contact with, even after being archived. | Archive flag was not applied to customers | 08/08/08 |
| User is not able to add new deliveries past the first one. | Connection with the database is not being closed. Add a closing method to the database. | 09/09/09 |
| User is not able to search for deliveries after filtering to a year or month | The search functionality was not implemented in the individual filtered views. They should be added respecting the filters. | 10/10/10 |
| User cannot remove drivers after their deliveries have been removed. | Driver archive button is not checking if the driver has any deliveries. A query to the database would solve this. | 11/11/11 |
| User cannot edit deliveries. There is seeming no option to do, even as it has been implemented | The edit button is hidden behind the deliveries table. Wrapping it on the same table cell as the archive button solves it. | 12/12/12 |

The table above shows a few bugs I had when building my delivery app.