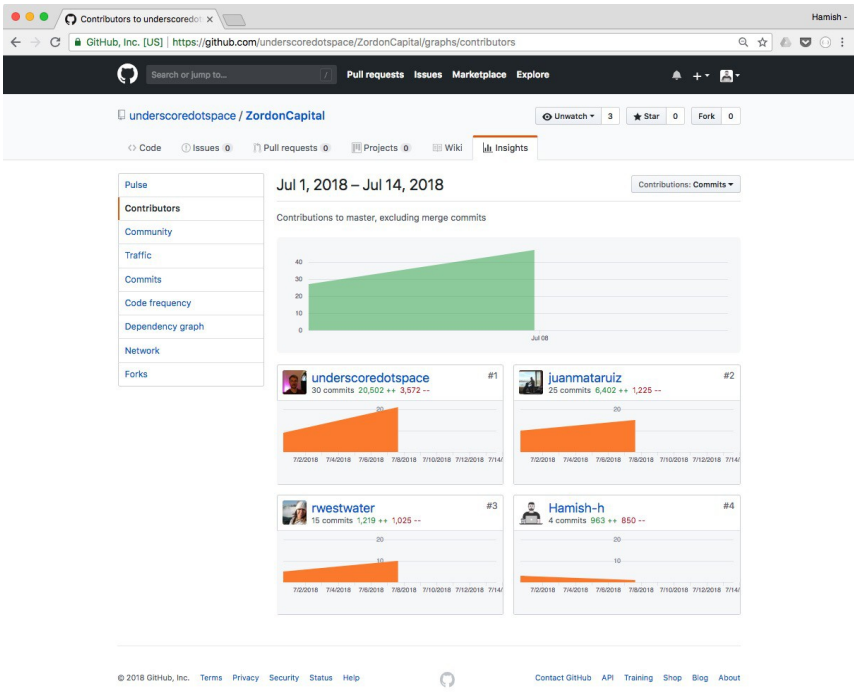
Evidence for Project Unit

Hamish Hoad

Cohort E-21

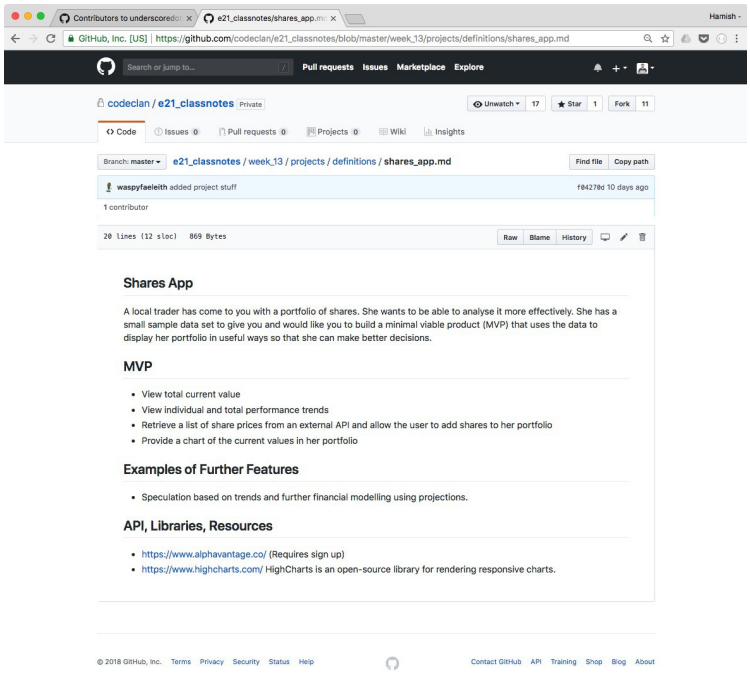
P. 1 Github Contributors page



Evidence for unit

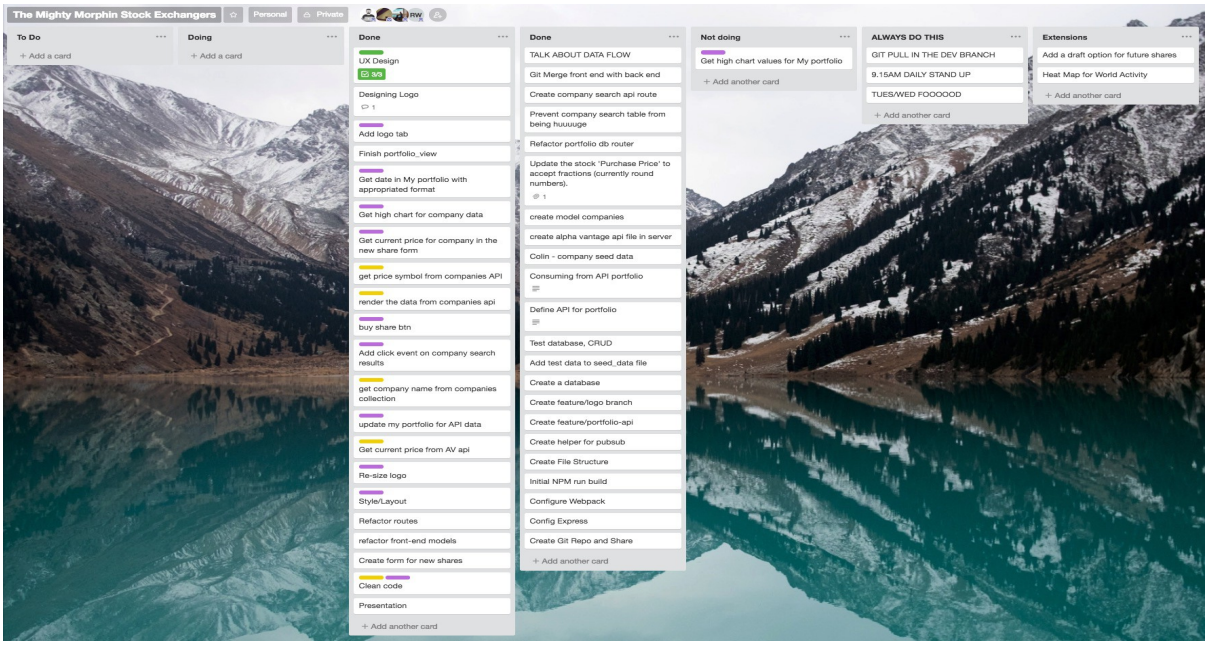
P. 2 Project Brief

Evidence for unit



P. 3 Use of Trello

Evidence for unit



P. 4 Acceptance Criteria

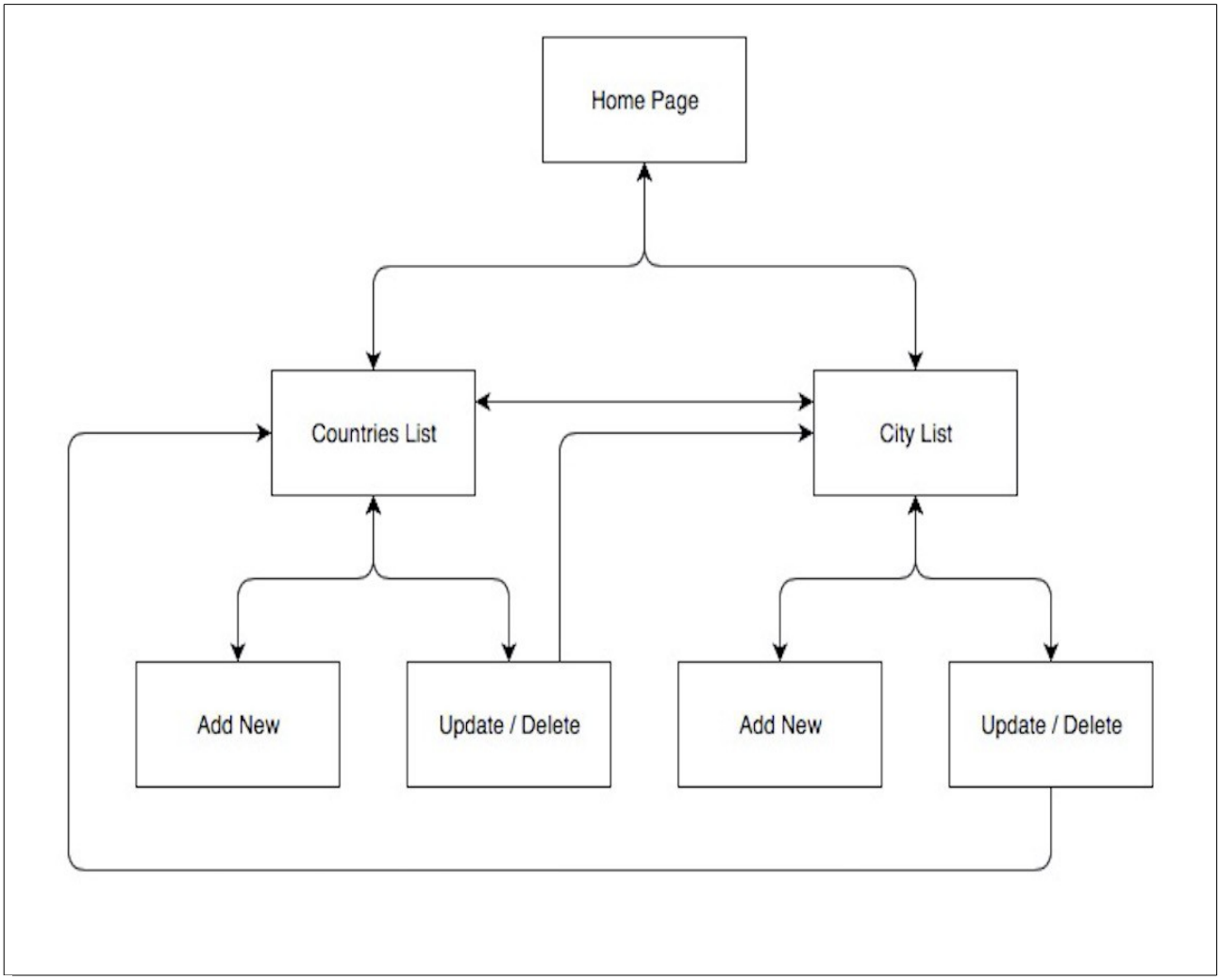
Evidence for unit

Acceptance Criteria and Test Plan

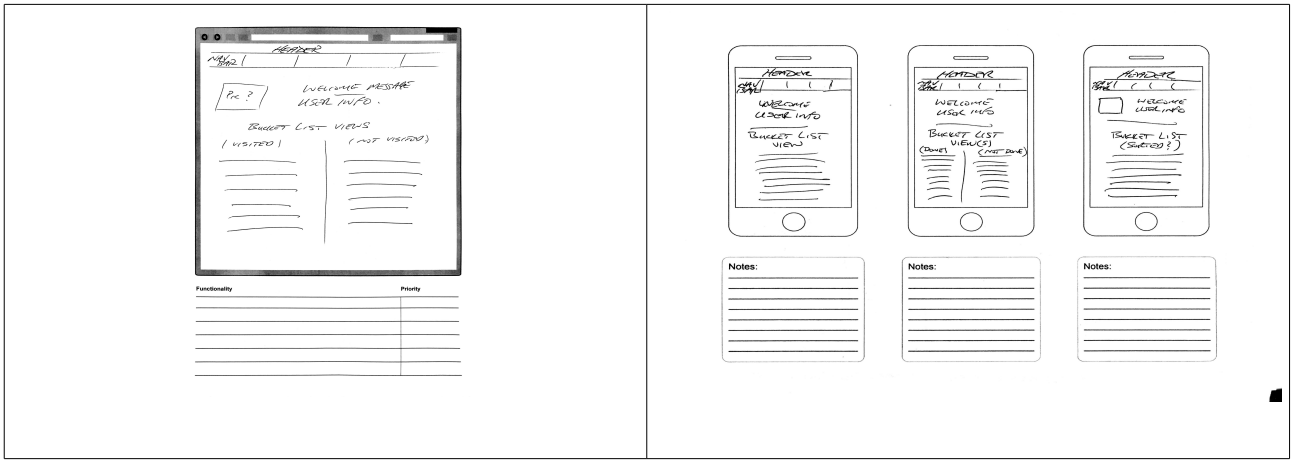
To display and analyse a portfolio of shares in useful ways to make better decisions.

Acceptance Criteria	Expected Result / Output	Pass / Fail
The user should be able to see a portfolio list of shares with the number purchased, the date of purchase, the price paid, the current valuation of each share and the profit and losses that have resulted.	List of purchased shares with the requisite information are displayed when the application opens.	Pass
The user should be able to search for new shares and see the historical share price with its profits and losses to assess the viability for a purchase.	A search box is displayed. The user can search for new shares and access the requisite information.	Pass
The user should be able to purchase shares which are added to the portfolio.	The user can purchase shares which add to the portfolio.	Pass
The user should not be able to purchase shares unless mandatory fields have been completed and a buy process initiated.	Information message appears when attempt to purchase is made without completion of mandatory fields.	Pass

P. 5 User sitemap

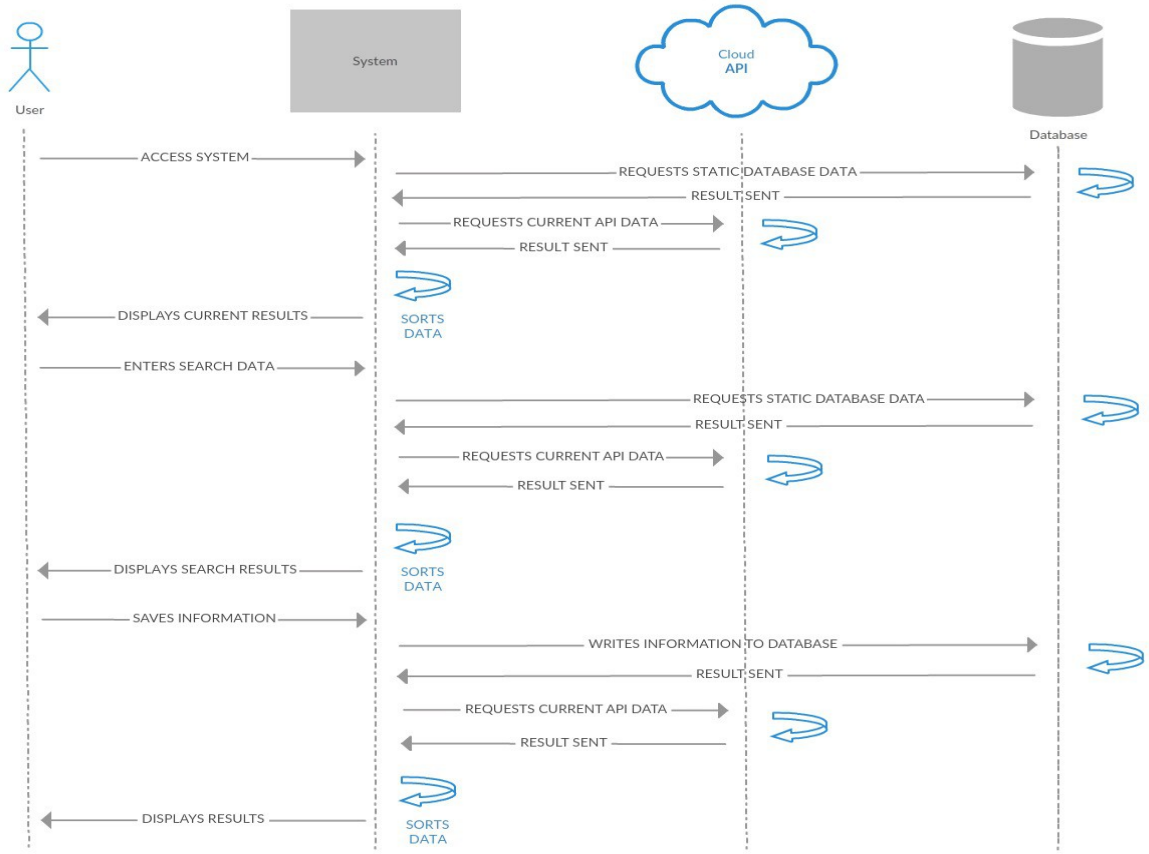


P. 6 Wireframes designs

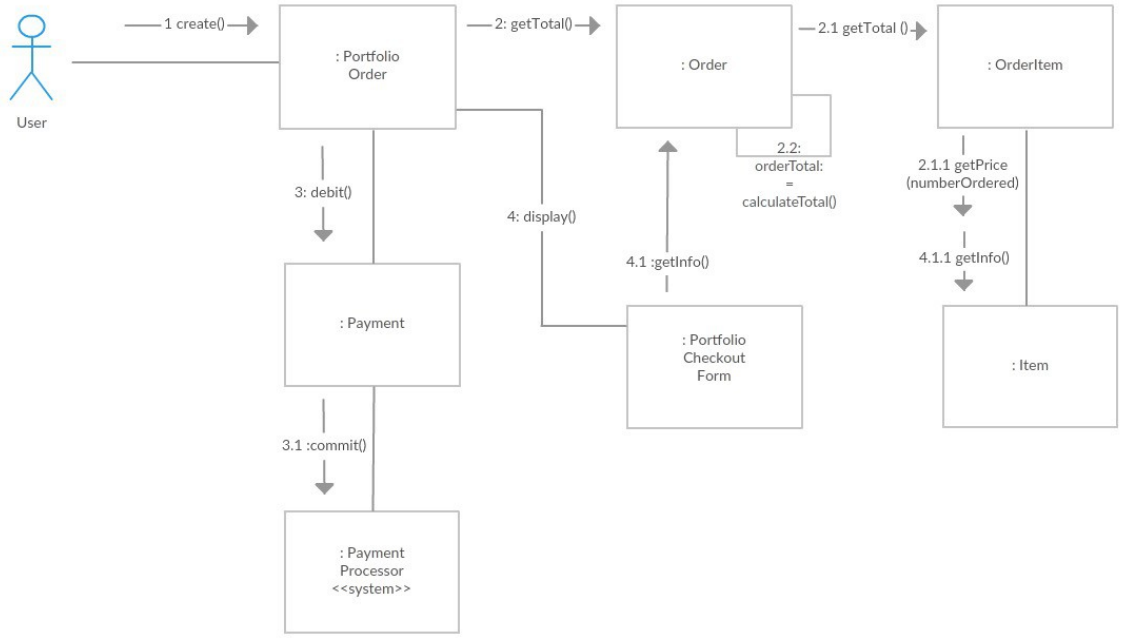


P. 7 System interactions diagrams

Sequence diagram

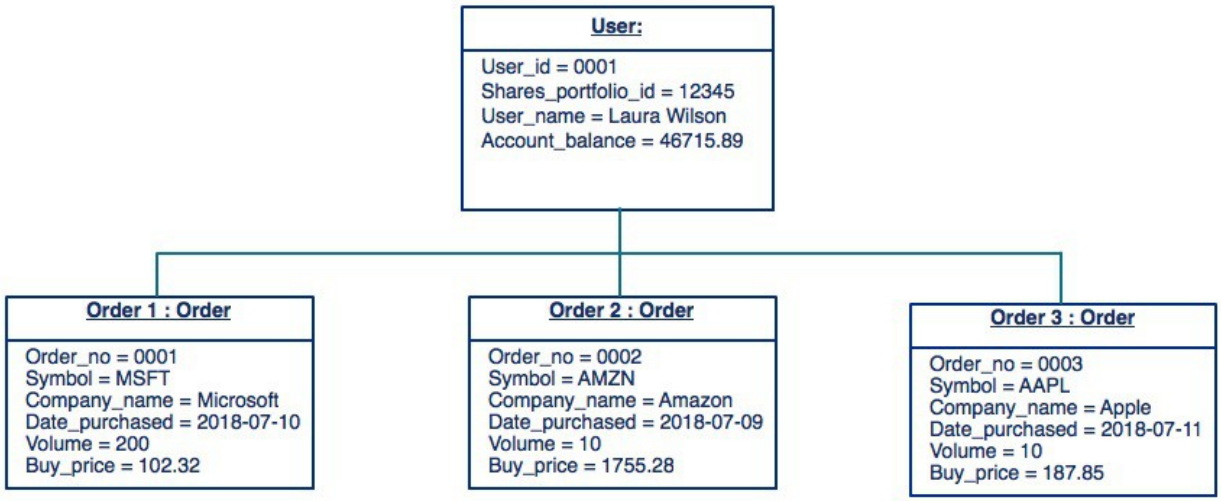


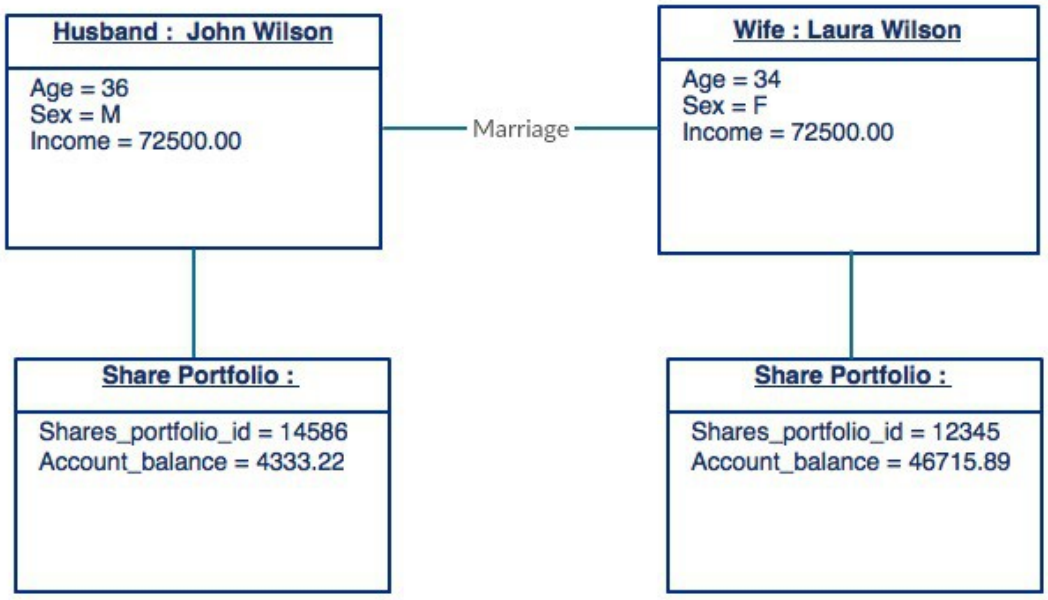
Collaboration diagram



P. 8 Two Object Diagrams

Evidence for unit Evidence for unit

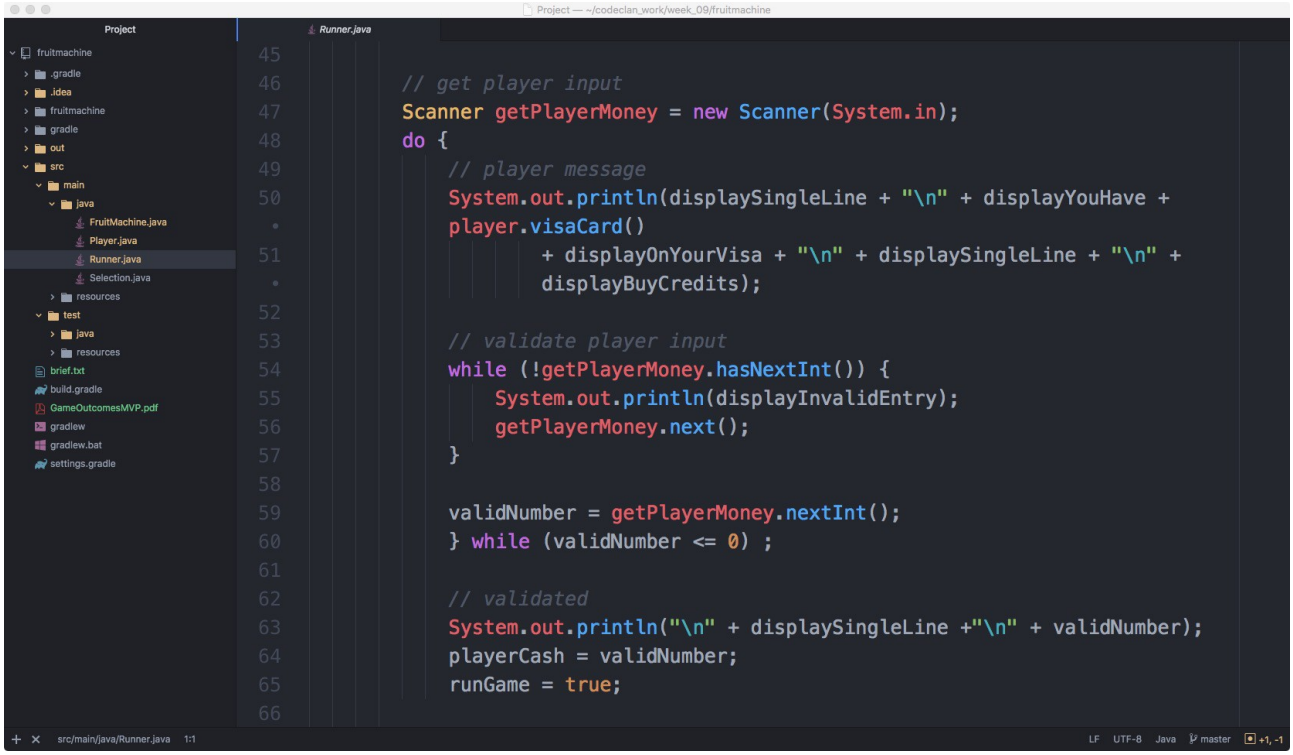




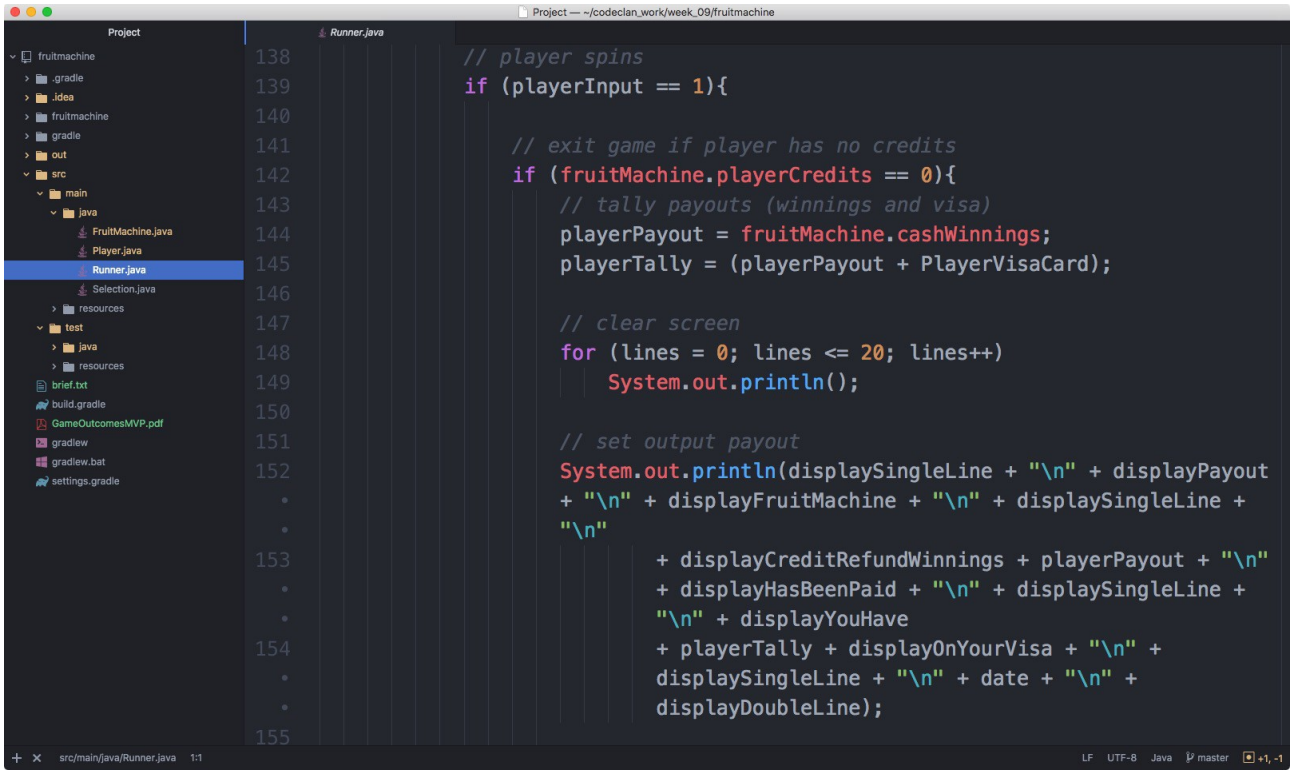
P. 9 Choice of two algorithms (find the algorithms on a program you might have written, show the code you have used.)

On this example please take a screenshot and write what it is doing and why you decided to use it.

The app is written in Java to mimic the gaming actions of a fruit machine which accepts money and gambles, replicating spinning reels of random fruits with a payout for matching values. The code displays a message to the user, accepts user input and validates that input. **Valid data** proceeds to the game and **Invalid data** returns a request to enter valid data.

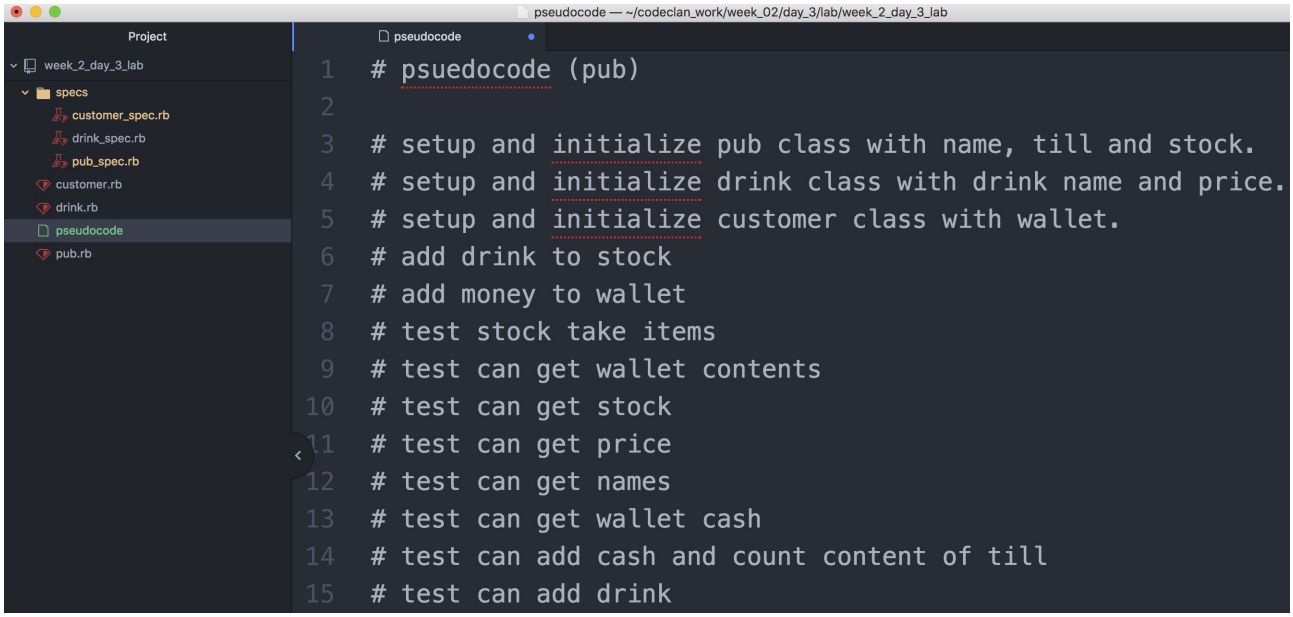


The code checks if the user / player has e-money in the fruit machine and if not, cashes out – tallying up any winnings which are displayed as a receipt credited back to his visa card.



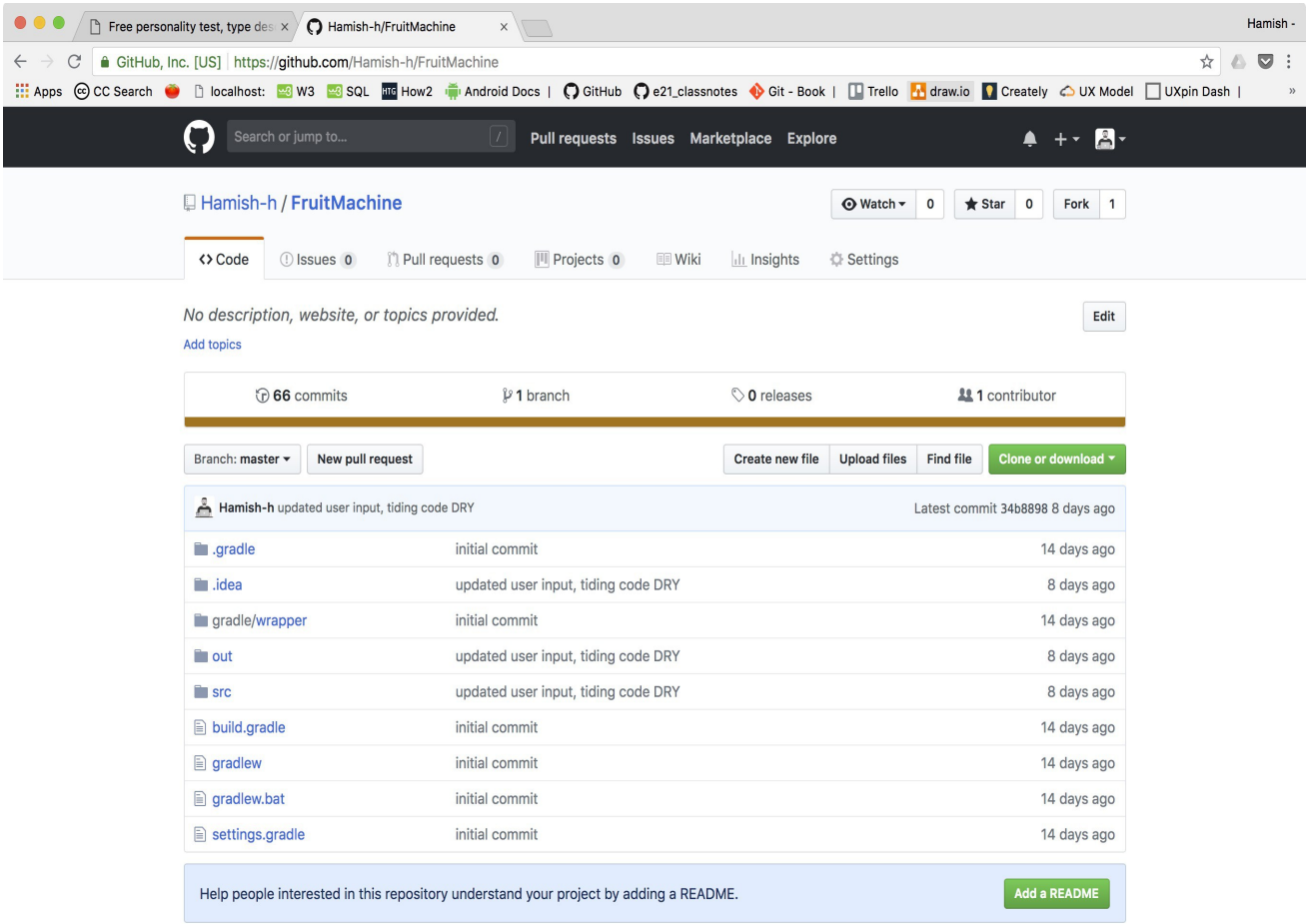
P. 10 Example of Pseudocode

Evidence for unit



P. 11 Github link to one of your projects

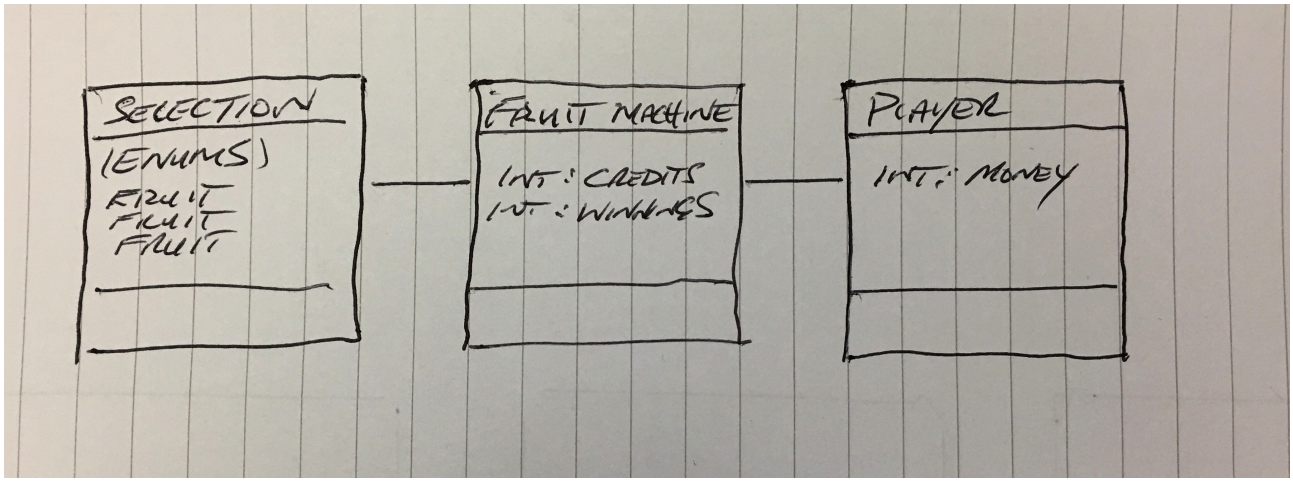
https://github.com/Hamish-h/FruitMachine



P. 12 Screenshot of your planning and the different stages of development to show changes.

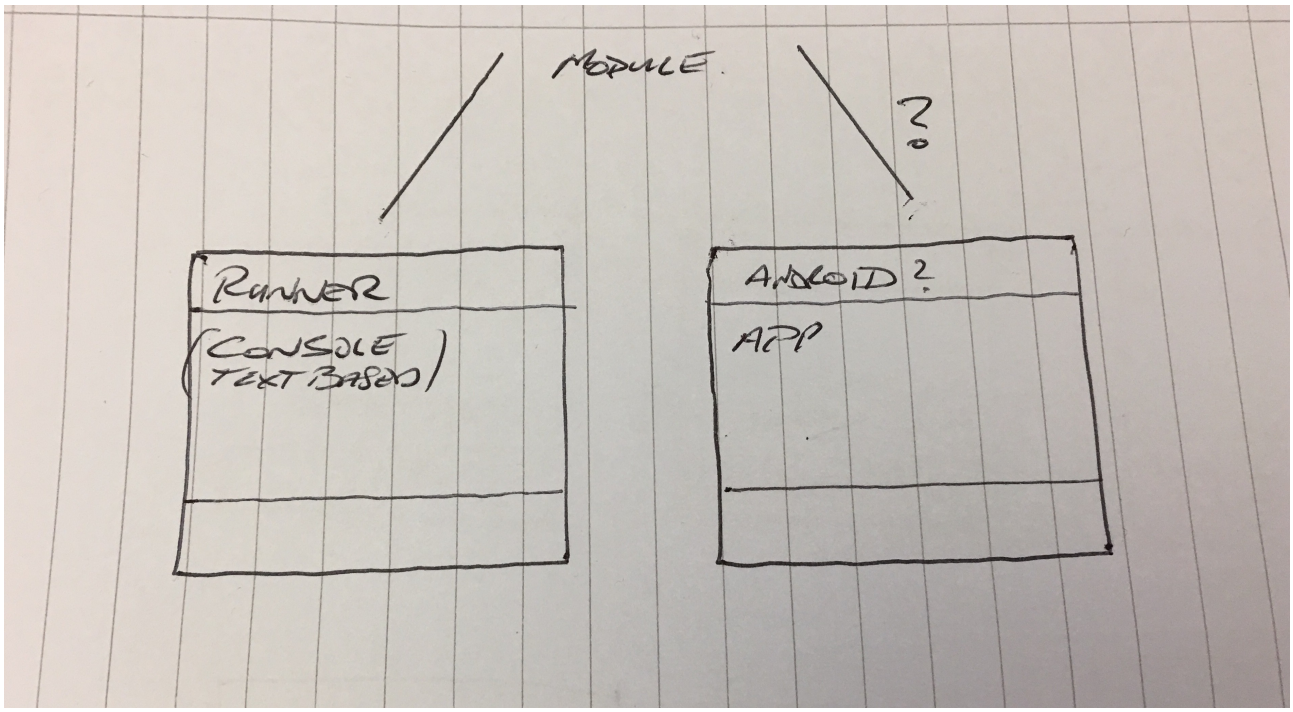
Planning and development – fruit machine

Rough plan

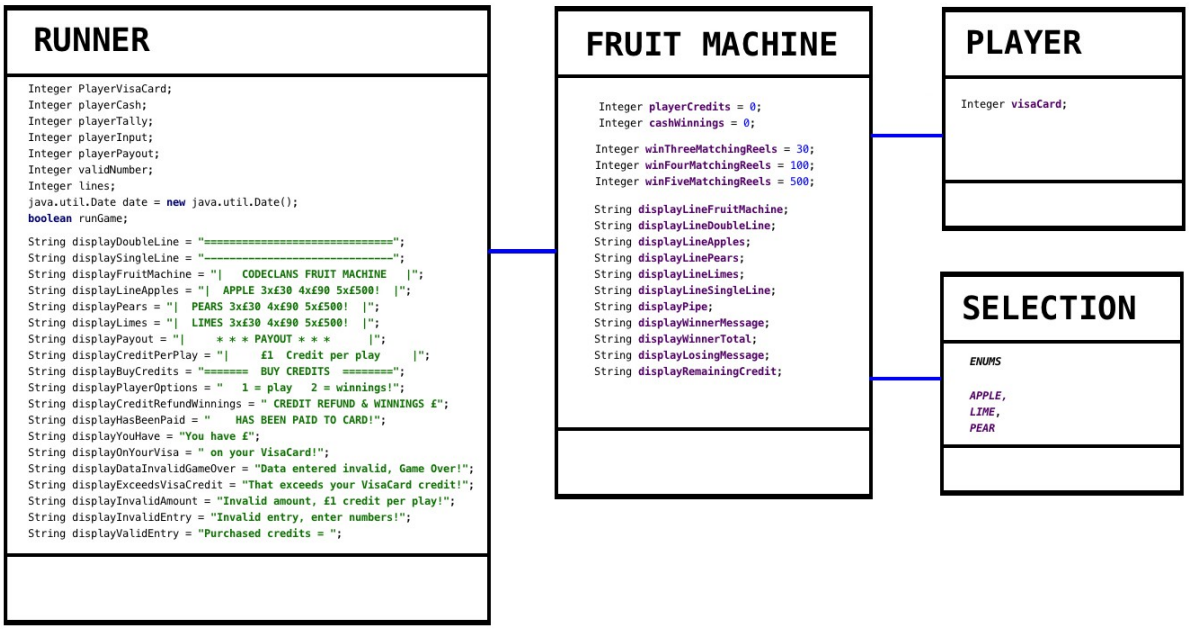


Rough plan addition for the control function

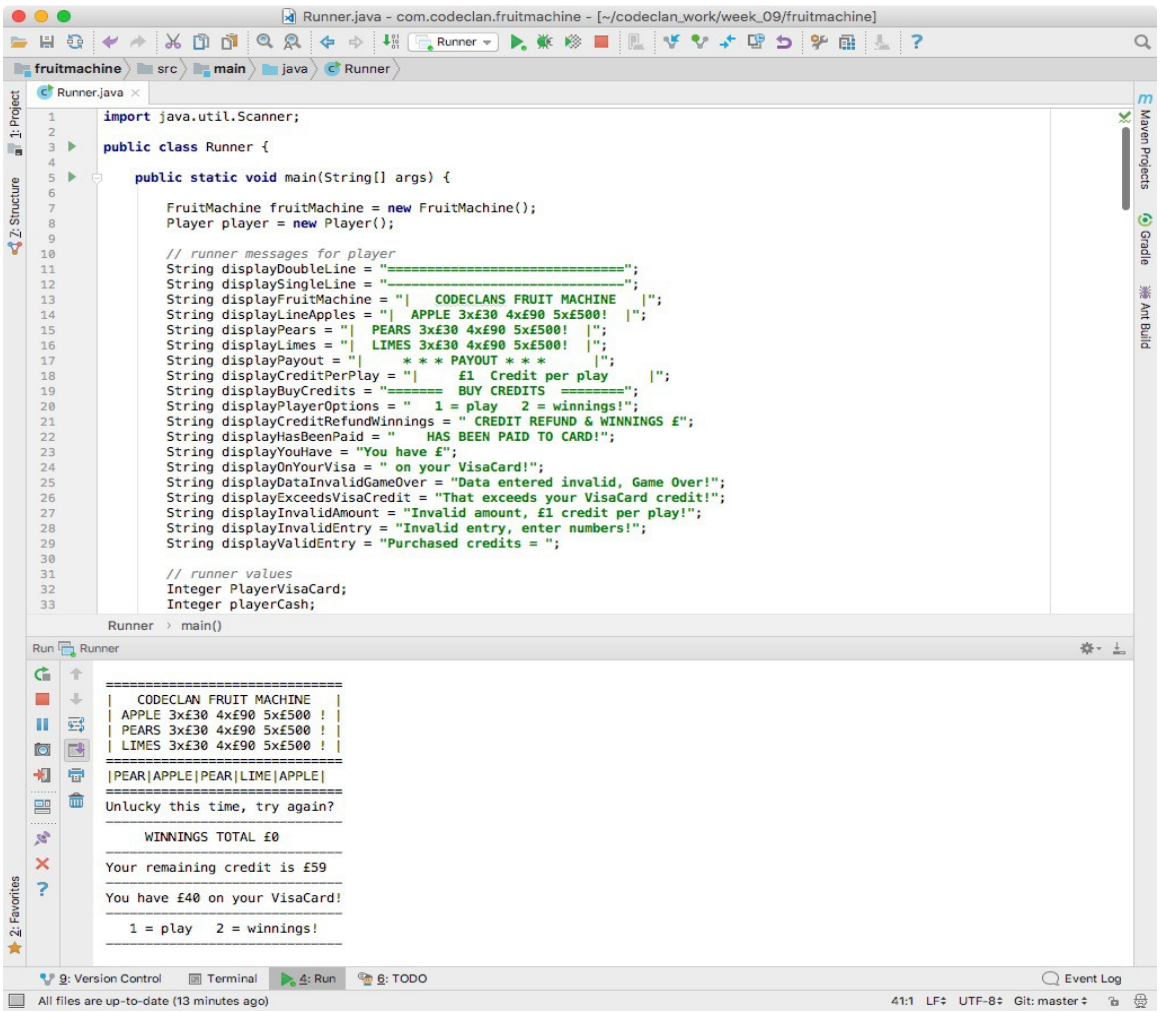
Java runner and Andriod app.



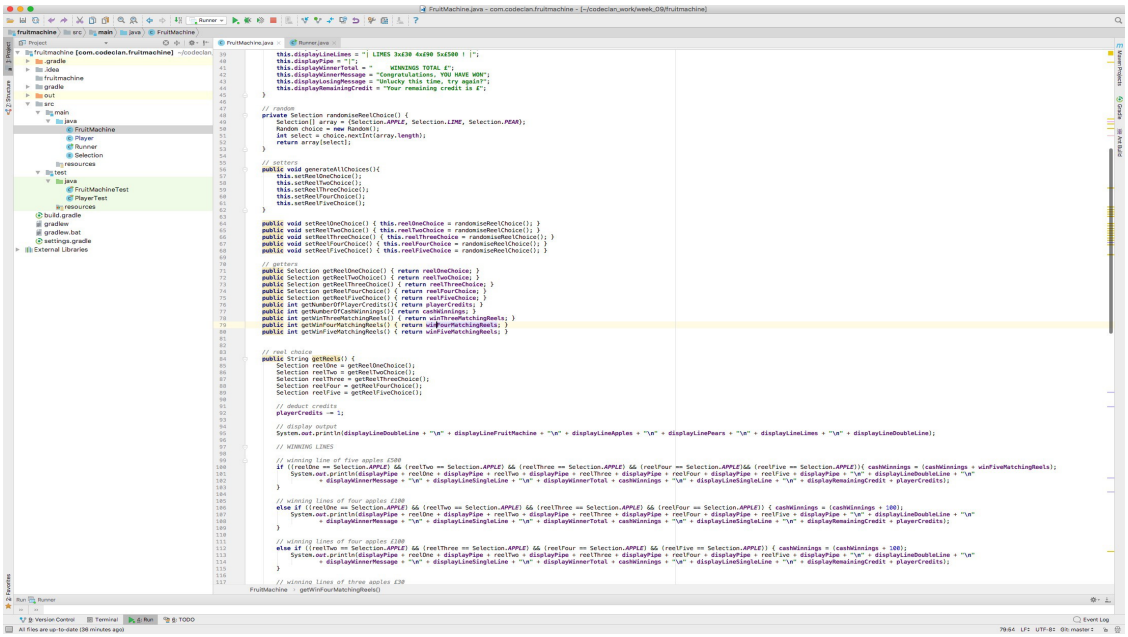
Detailed plan



Coding steps



Coding Steps



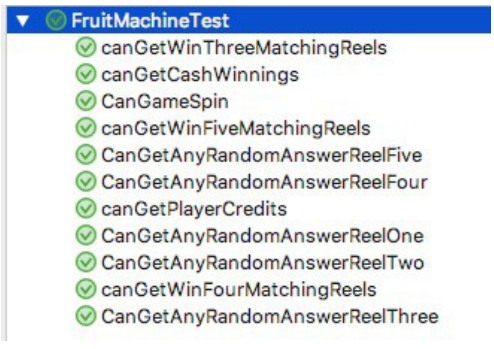
Coding Steps

```
// WINNING LINES

// winning line of five apples £500
if ((reelOne == Selection.APPLE) && (reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE)&& (reelFive == Selection.APPLE)){ cashWinnings = (cashWinnings + winFiveMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
            + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of four apples £100
else if ((reelOne == Selection.APPLE) && (reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE)) { cashWinnings = (cashWinnings + 100);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
            + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of four apples £100
else if ((reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE) && (reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + 100);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
            + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of three apples £30
else if ((reelOne == Selection.APPLE) && (reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE)) { cashWinnings = (cashWinnings + winThreeMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
            + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of three apples £30
else if ((reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE)){ cashWinnings = (cashWinnings + winThreeMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
            + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of three apples £30
else if ((reelThree == Selection.APPLE) && (reelFour == Selection.APPLE) && (reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + winThreeMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
            + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}
```

Output

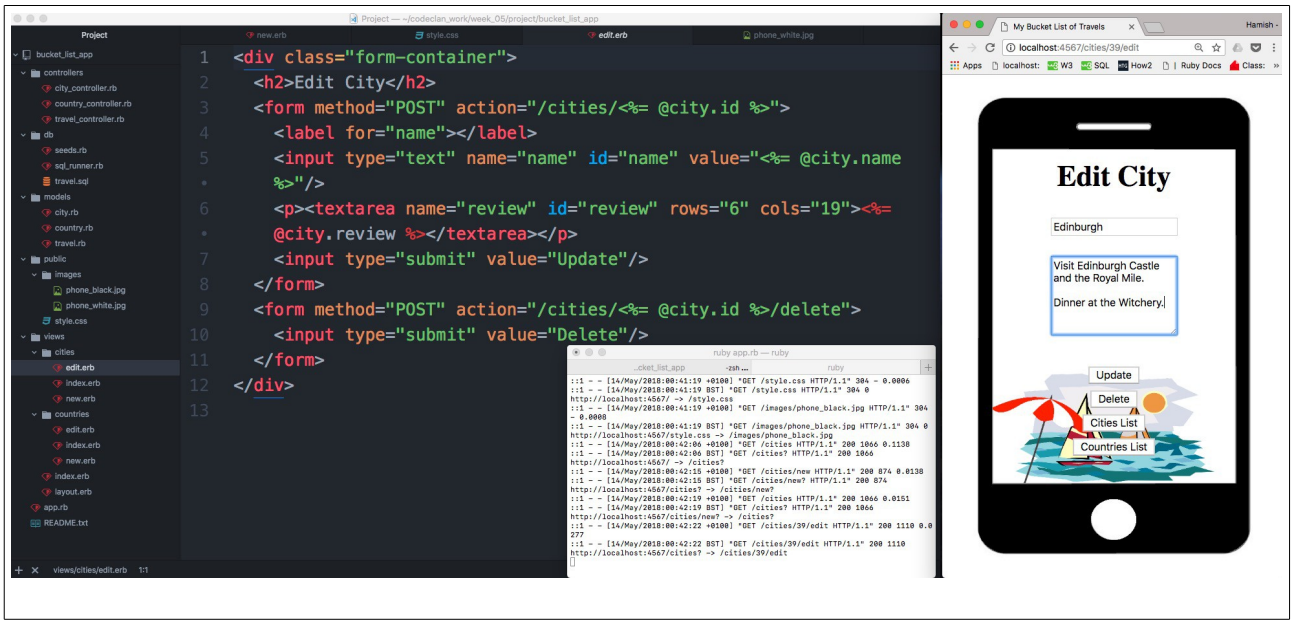
```
================================
|   CODECLAN FRUIT MACHINE     |
| APPLE 3x£30 4x£90 5x£500 ! |
| PEARS 3x£30 4x£90 5x£500 ! |
| LIMES 3x£30 4x£90 5x£500 ! |
================================
|PEAR|APPLE|PEAR|LIME|APPLE|
================================
Unlucky this time, try again?
--------------------------------
      WINNINGS TOTAL £0
--------------------------------
Your remaining credit is £59
--------------------------------
You have £40 on your VisaCard!
--------------------------------
    1 = play    2 = winnings!
--------------------------------
```

Tests



P. 13 User input

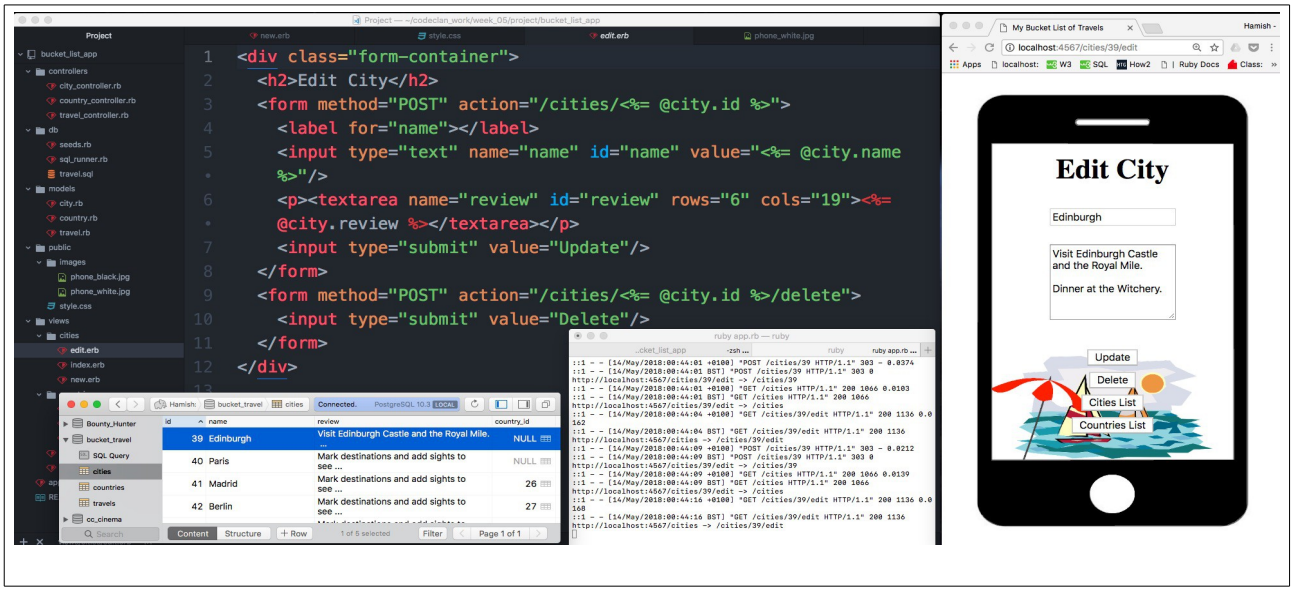
User inputs and edits a City, adding places to visit.



P. 14 Interaction with data persistence

Make sure you show the input being added.

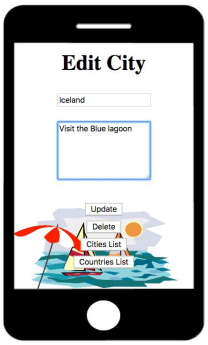
User inputs a City, adds places to visit and saves the information which is then added to the database as shown.



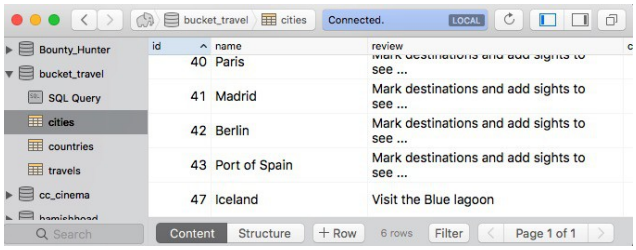
P. 15 User output result

User inputs information and clicks add city which adds information to the database.

The system then returns a list of cities showing the new city has been added to the database.



The user can then update (or delete) city information, such as a place to visit.



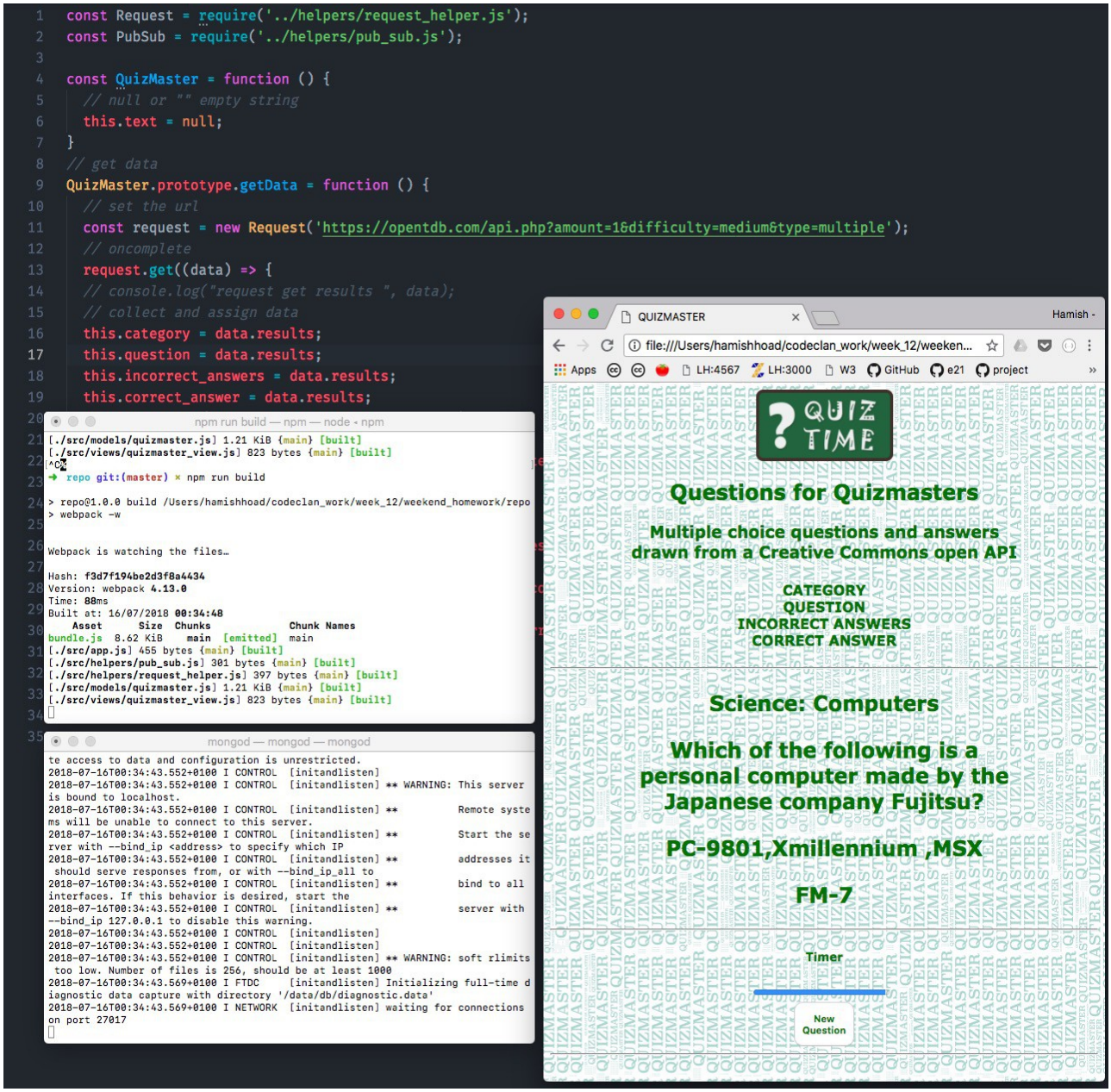
The information is then added to the database as shown in the above tables.

P. 16 Show an API being used within your program.

The code that uses / implements the API.

```
const Request = require('../helpers/request_helper.js');
const PubSub = require('../helpers/pub_sub.js');

const QuizMaster = function () {
  // null or "" empty string
  this.text = null;
}
// get data
QuizMaster.prototype.getData = function () {
  // set the url
  const request = new Request('https://opentdb.com/api.php?amount=1&difficulty=medium&type=multiple');
  // oncomplete
  request.get((data) => {
  // console.log("request get results ", data);
  // collect and assign data
  this.category = data.results;
  this.question = data.results;
  this.incorrect_answers = data.results;
  this.correct_answer = data.results;
  // console.log("this.category", this.category);
  // publish category
  PubSub.publish('QuizMaster:quizMaster-loaded', this.category[0].category);
  console.log(this.category[0].category);
  console.log(this.question[0].question);
  // publish
  PubSub.publish('QuizMaster:quizMaster-loaded', this.question[0].question);
  // publish incorrect answers
  PubSub.publish('QuizMaster:quizMaster-loaded', this.incorrect_answers[0].incorrect_answers);
  // publish correct answer
  PubSub.publish('QuizMaster:quizMaster-loaded', this.correct_answer[0].correct_answer);
  });
}
// export
module.exports = QuizMaster;

```

The API being used by the program while running.



P. 17 Bug tracking report showing the errors diagnosed and corrected.

Evidence for unit

Bug Tracking Report

Record of bugs in a Javascript Software Product.

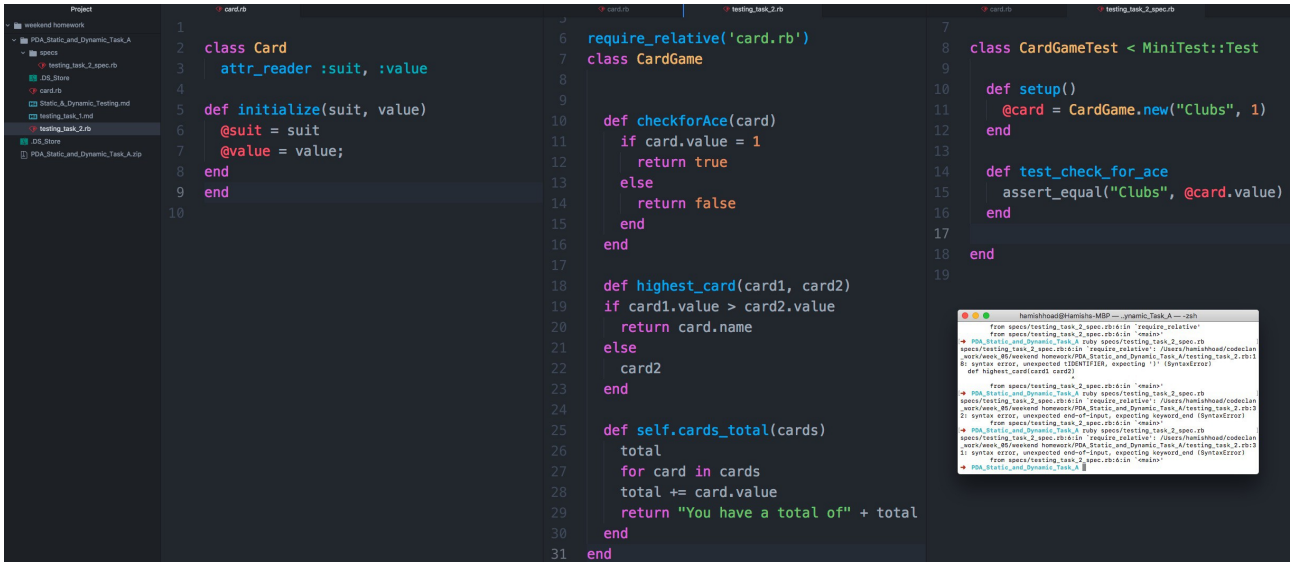
The back-end server models connect to the front end public src form views.	Failed	Corrected spelling mistake in the server/models module.exports name.	Passed
The database data populates views.	Failed	Updated bracket formatting of the seed_data JSON file and repopulated the database.	Passed
The API data populates the information on user screen.	Failed	Updated constructor require URL path to the API file.	Passed
The app calculates a price for an item.	Failed	Updated the item name from lower case to camel case.	Passed
The user input value box should prevent negative numbers.	Failed	Restricted user input to a number equal to or greater than 1.	Passed
The html form populated with API data should not be overwritten.	Failed	Amended the html input type value to equal disabled.	Passed

P. 18 Testing your program

Show the test code, the test not passing.....and then the test fixed.

Example of test code

with the test failing to pass **NOT Passing**



Example of test code once errors have been corrected

with the test then passing, fixed **Passing**