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.

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

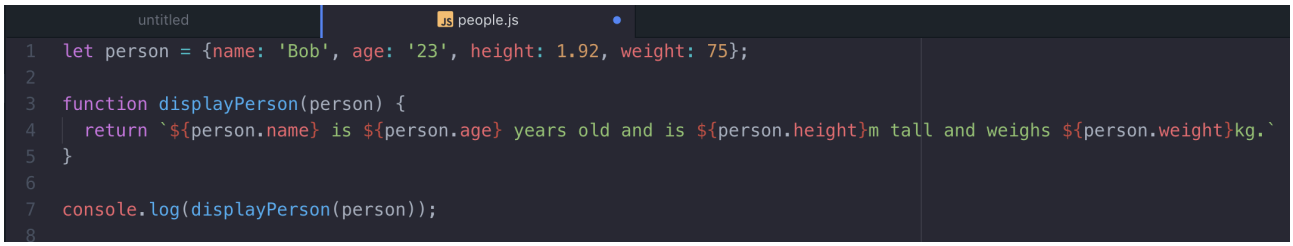
Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

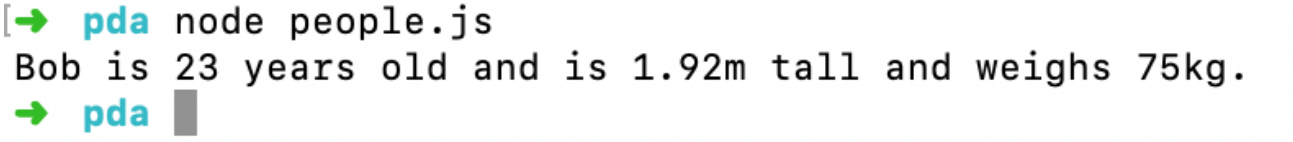
Evidence Key: A&D - Analysis and Design Unit
I&T - Implementation and Testing Unit
P - Project Unit

Week 1

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

Paste Screenshot here



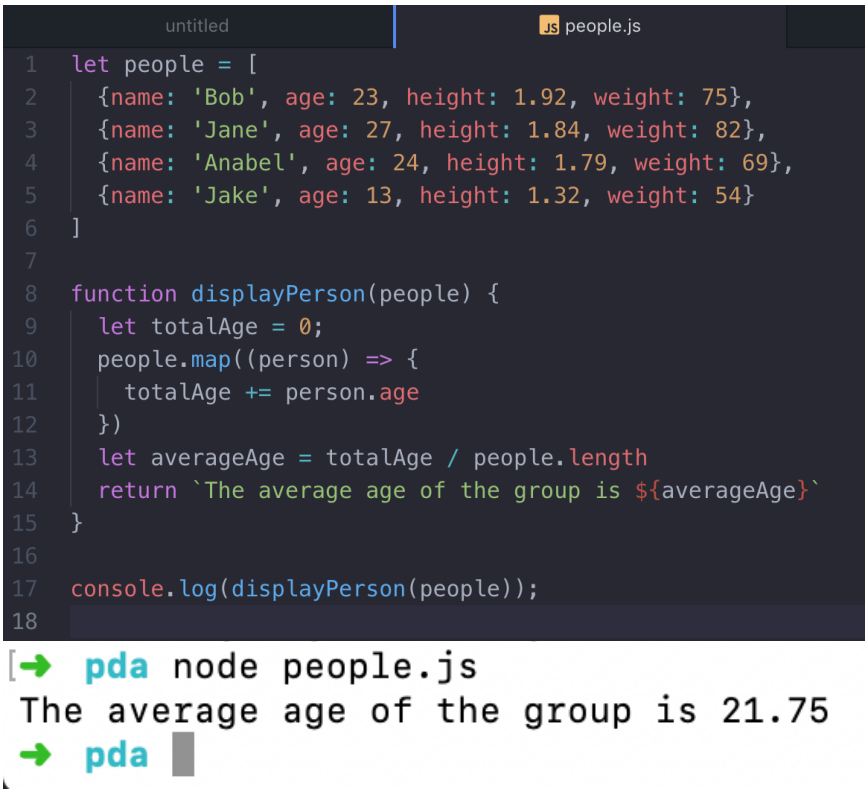


Description here

Here the object is for a person called Bob, the function “displayPerson” is used to log some information about the person. The second screenshot is of node displaying the result of running the program.

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

Paste Screenshot here



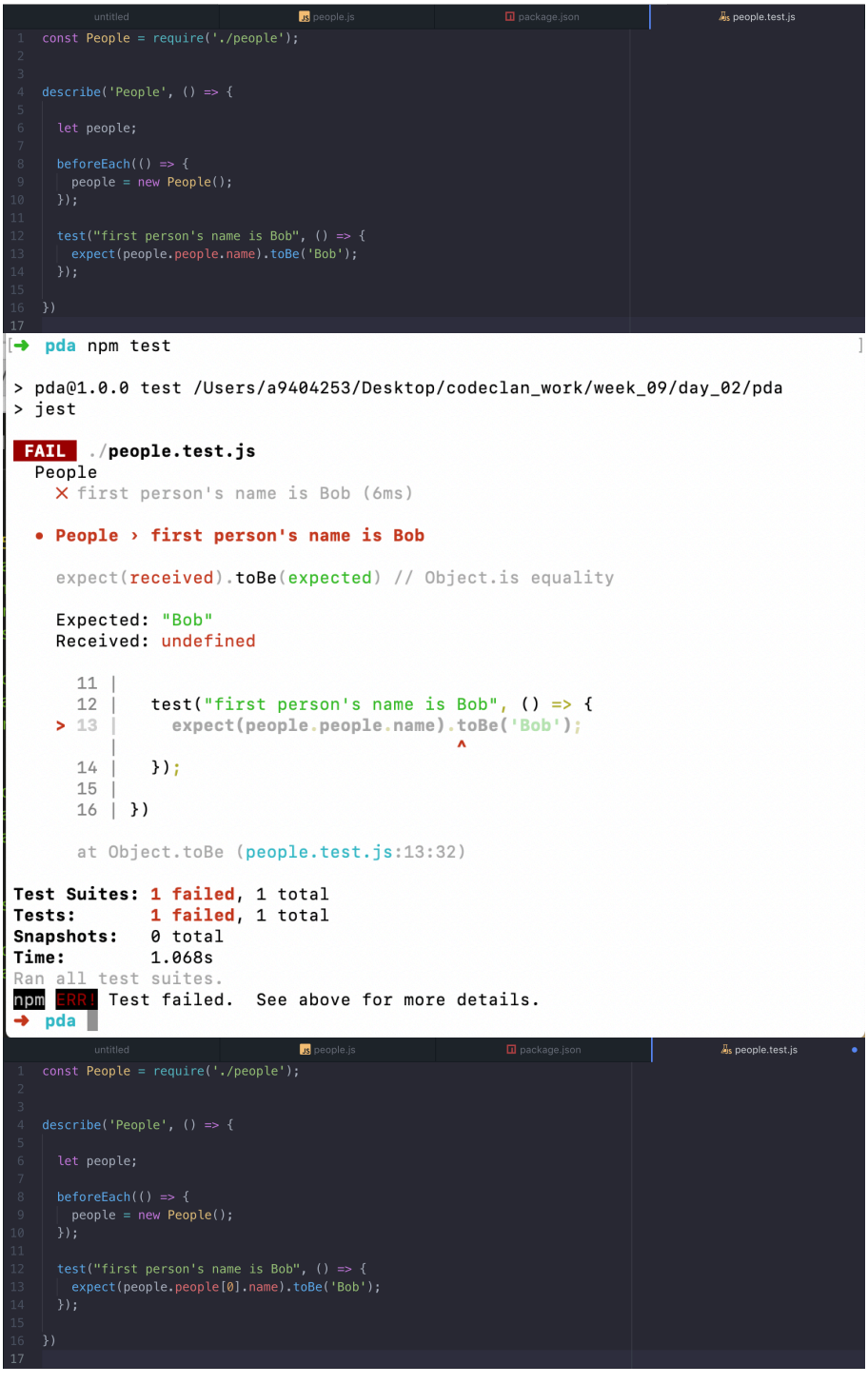
Description here

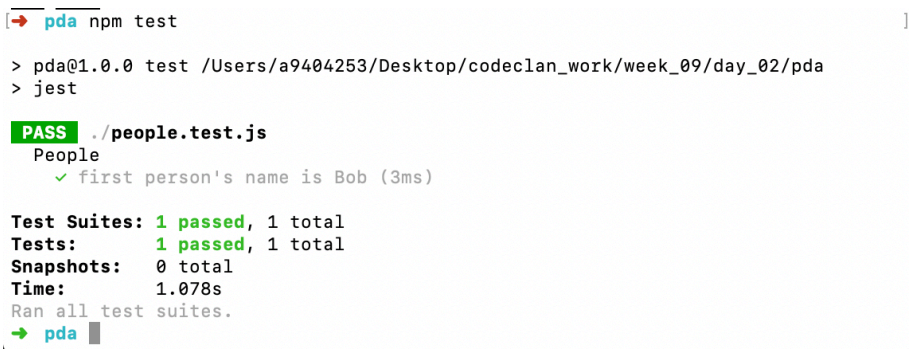
Here the people array is an array of person objects. The function “displayPerson” is used to calculate the average age of the people in the array. The second screenshot shows the program working using node.

Week 2

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

Paste Screenshot here





Description here

Here a class “People” is imported containing an array of people. The first test doesn’t specify which person it expects to be “Bob” and thus fails. The second test expects the first person in the array to be Bob and so passes.

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

Paste Screenshot here

people.js | package.json | people.test.js | encapsulation.js

```
1  class Skyscraper {
2    constructor() {
3      let height = 100; // private instance data
4      this.getHeight = function() {
5        return height;
6      }; // methods with access to private variable
7      this.setHeight = function(newHeight) {
8        height = newHeight;
9      };
10   }
11 }
12
13 var skyscraper = new Skyscraper();
14 console.log(skyscraper.getHeight()); // returns 100
15 skyscraper.setHeight(120);
16 console.log(skyscraper.getHeight()); // returns 120
17 console.log(skyscraper.height); // undefined
```

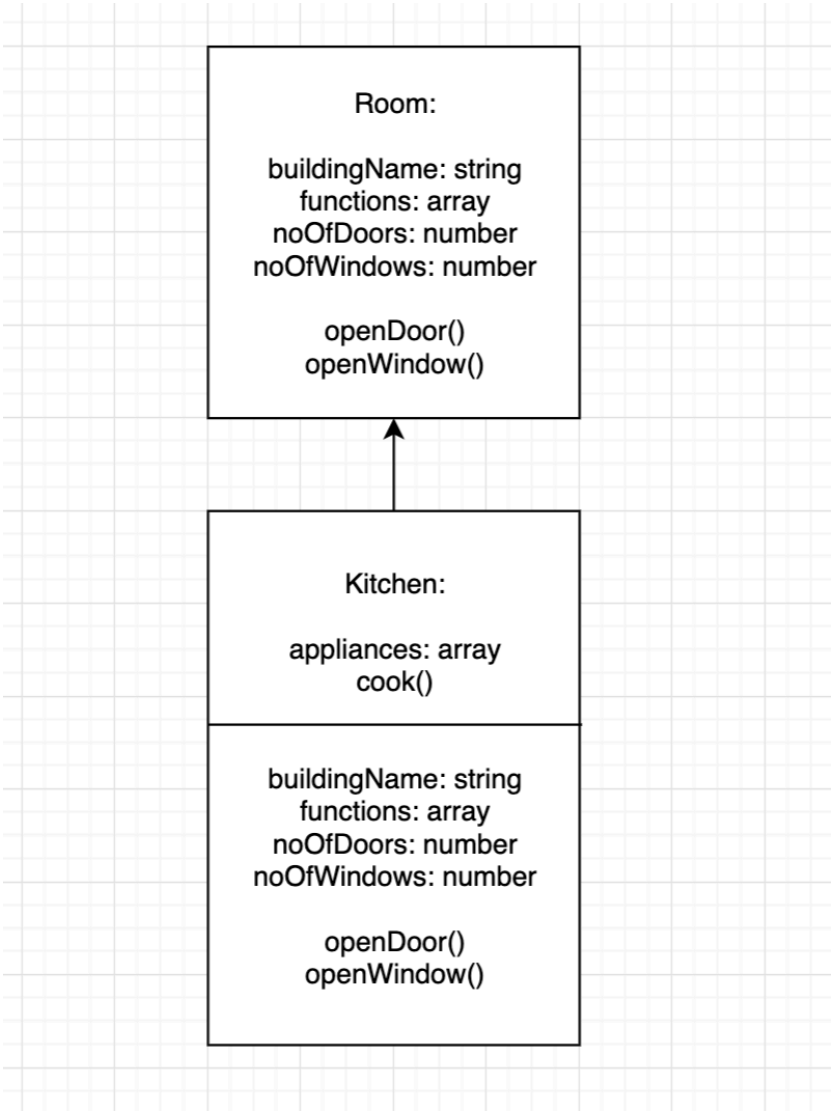
```
[➜  pda node encapsulation.js
100
120
undefined
```

Description here

The height of the skyscraper here cannot be called or set outside the Skyscraper class, instead you need to call the functions to make changes or log the height.

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

Paste Screenshot here



Description here

In the above inheritance diagram, the kitchen inherits all the properties of the room but also has it's own variable and function.

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.

Paste Screenshot here

```
class Room {
  constructor(name, array, doors, windows) {
    this.room = {
      buildingName: name,
      funtions: array,
      noOfDoors: doors,
      noOfWindows: windows
    }
  }
  numberOfDoors() {
    return this.room.noOfDoors;
  }

  openDoor() {
    console.log("Door has been opened");
  }

  openWindow() {
    console.log("Window has been opened");
  }
}

module.exports = Room;
```

```
const Room = require('./room.js');

class Kitchen extends Room {
  constructor(name, array, doors, windows, appliances) {
    super(name, array, doors, windows)
    this.appliances = appliances
  }

  cook() {
    if (this.room.noOfWindows > 0) {
      console.log("cooking food");
    }
  }

}
let kitchen = new Kitchen("PH", ["cooking"], 2, 3, ["oven", "fridge", "microwave"])
kitchen.cook()
```

```
➜  day_05 node kitchen.js
cooking food
```

Description here

The Kitchen class inherits from the Room class and the method cook() in the Kitchen class uses information from the object in the Room class to log “cooking food”.

Week 3

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

```
constructor(props){
  super(props)
  this.state = {
    answer: 0,
    max: 0
  };
  this.diceRoll = this.diceRoll.bind(this)
  this.setMax = this.setMax.bind(this)
}

setMax(maxNum) {
  this.setState({max: maxNum})
}

diceRoll(){
  this.setState(() => {
    return {answer: (Math.floor(Math.random()* Math.floor(this.state.max))+1)}
  })
}
```

```
const newRandom = () => {
  for (var i = 0; i < 1;) {
    let randomPosRow = Math.floor(Math.random()*4)
    let randomPosCol = Math.floor(Math.random()*4)
    let value = Math.floor(Math.random()*2)
    if (props.rows[randomPosRow][randomPosCol] === 0) {
      props.changeNumber(((randomPosRow*4) + randomPosCol), ((value+1)*2))
      i++
    }
  }
}
```

Description here

1

The first screenshot is of a dice rolling app, it lets the user enter a number of sides on the app (max) and then returns a random number between 1 and the max.

2

The second screenshot is some code out of an app that was planned to recreate the game “2048” where numbers need to be randomly placed on a 4x4 grid, this code generates 2 random numbers between 0 and 4 and rounds then down, this is used to select the position on the grid. The “value” variable is used to randomly select the number that is added to the game as either 0 or 1, which is later turned to 2 or 4. The changeNumber function is then called to change the number in the selected box to the new value.

Week 4

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

Paste Screenshot here

```
1
2     let people = [
3       {name: 'Bob', age: 23, height: 1.92, weight: 75},
4       {name: 'Jane', age: 27, height: 1.84, weight: 82},
5       {name: 'Anabel', age: 24, height: 1.79, weight: 69},
6       {name: 'Jake', age: 13, height: 1.32, weight: 54}
7     ]
8
9
10   function findPerson(someone) {
11     return people.find((person) => {
12       return (person.name == someone)
13     })
14   }
15
16 console.log(findPerson('Bob'));
17
```

```
➜  pda node people.js
{ name: 'Bob', age: 23, height: 1.92, weight: 75 }
➜  pda
```

Description here

The function “findPerson” Is used to search through the people array using the inputted name from the user. The second screenshot shows the result of the console log using node.

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

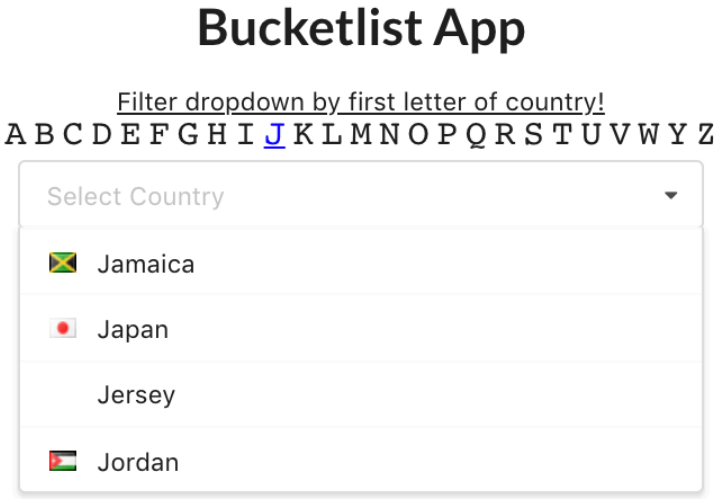
Paste Screenshot here

```
createLetters(evt) {
  let letters=("ABCDEFGHIJKLMNOPQRSTUVWYZ").split("")
  return letters.map((letter, index) => {
    return <option value={letter} key={index} onClick={() => {this.handleSelectLetter(letter)}} className='selectedLetter'>{letter}</option>
  });
}

handleSelectLetter(letter) {
  this.props.selectLetter(letter)
}

options() {
  console.log(this.handleSubmit);
  if (this.props.letter === null) {
    return this.props.countries.map((element, index) => {
      return <DDOption info={element.name} flag={element.alpha2Code} key={index} handleSubmit={this.handleSubmit}/>;
    });
  } else {
    let countriesArray = []
    this.props.countries.forEach((country, index) => {
      let name = country.name.slice(0,1)
      if (name === this.props.letter) {
        let newCountry = {name: country.name, flag: country.alpha2Code, index: index}
        countriesArray.push(newCountry)
      }
    })
    console.log(countriesArray);
    return countriesArray.map((element, index) => {
      console.log(element);
      return <DDOption info={element.name} flag={element.flag} key={index} handleSubmit={this.handleSubmit}/>;
    });

  }
};
```



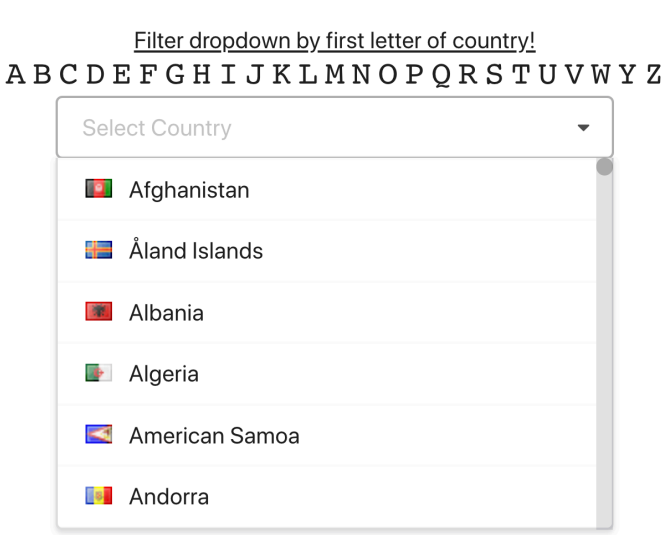
Description here

This program has a dropdown populated by an external API, as you can see in the first screenshot, I have written logic so that if the user clicks on one of the letters above the dropdown it filters the dropdown to only countries that start with the selected letter.

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

```
getCountry(country) {
  dispatch(() => {
    fetch(`https://restcountries.eu/rest/v2/name/${country}`)
    .then(res => res.json())
    .then(bucketlist => {
      bucketlist[0].visited = false
      dispatch({
        type: 'ADD_BUCKET',
        bucketlist: bucketlist[0]
      })
    })
  })
},
```

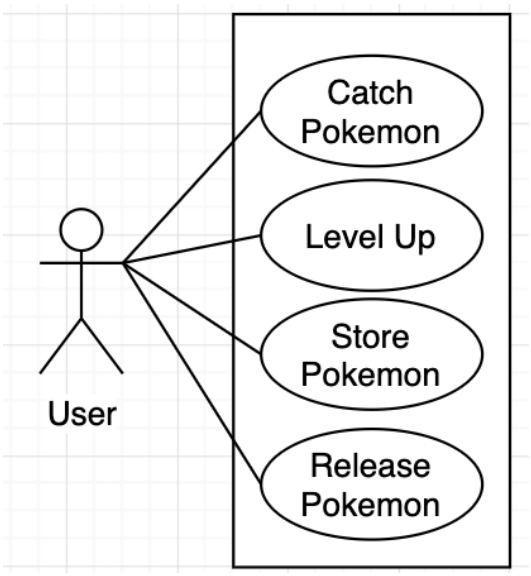


Description here

The first screenshot above shows the API call to get countries from the REST Countries API, this is used in the program to populate the dropdown box shown in the second screenshot.

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

Paste Screenshot here

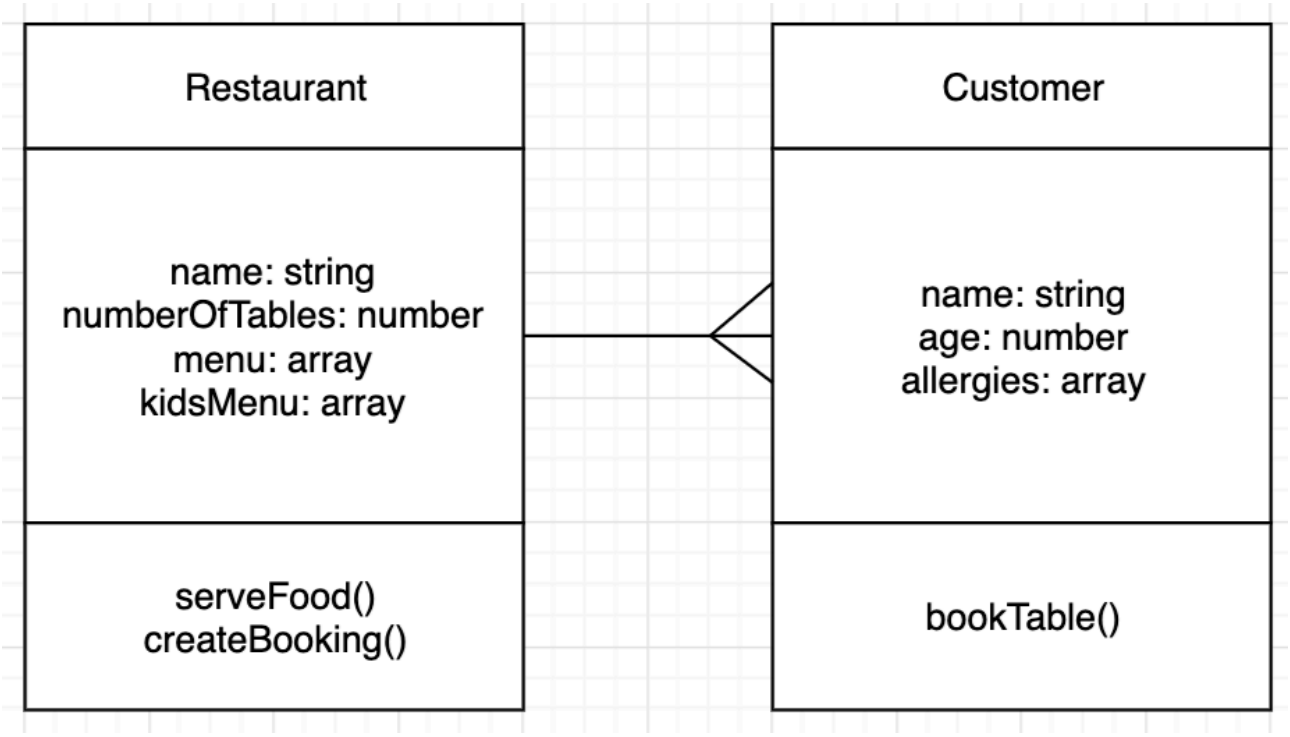


Description here

This example is from my individual project which allows the user to catch pokemon and then update information on them by levelling them up, moving them to their library using the “store” button, and finally release the pokemon back into the wild (delete them).

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

Paste Screenshot here

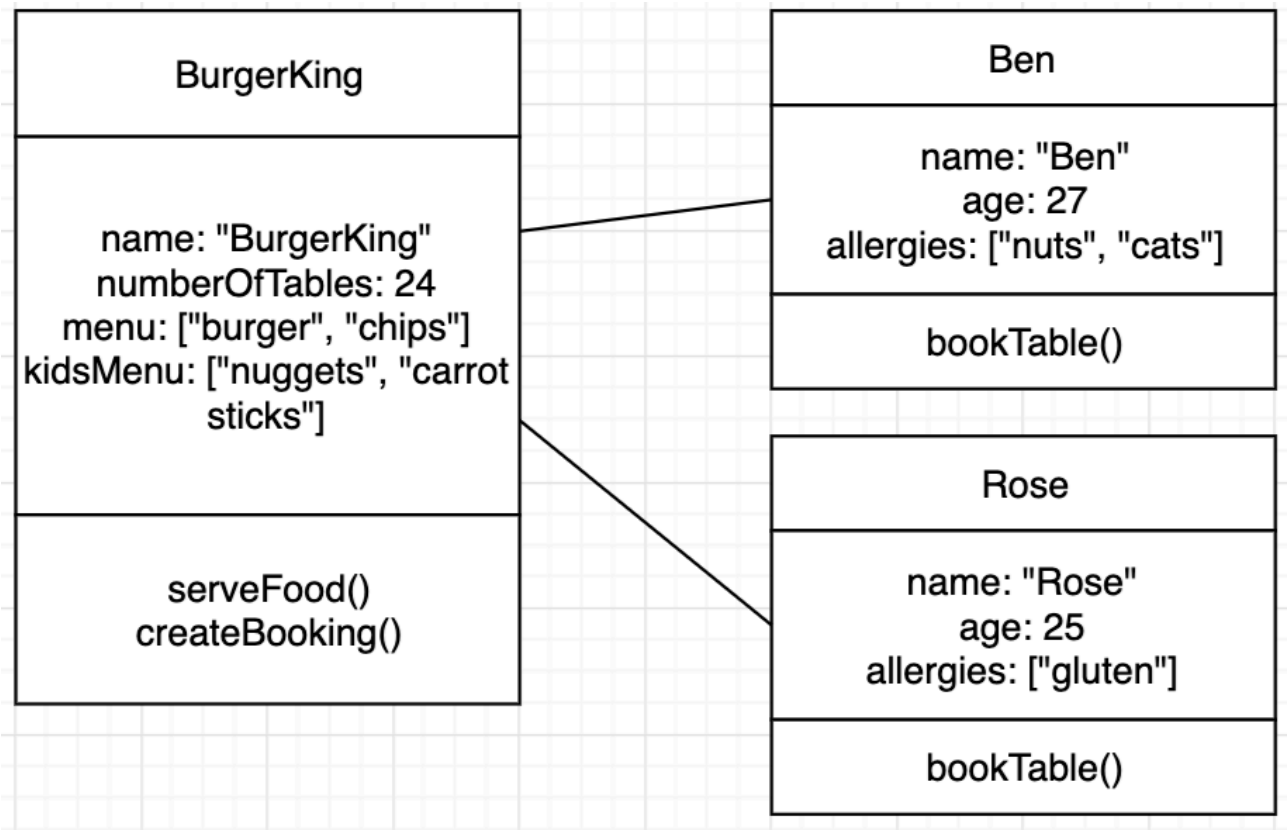


Description here

This class diagram shows the relationship between a restaurant and its customers and shows that a restaurant can have many customers.

Unit	Ref	Evidence
A&D P	A.D.3 P.8	Produce three Object Diagrams

Paste Screenshot here

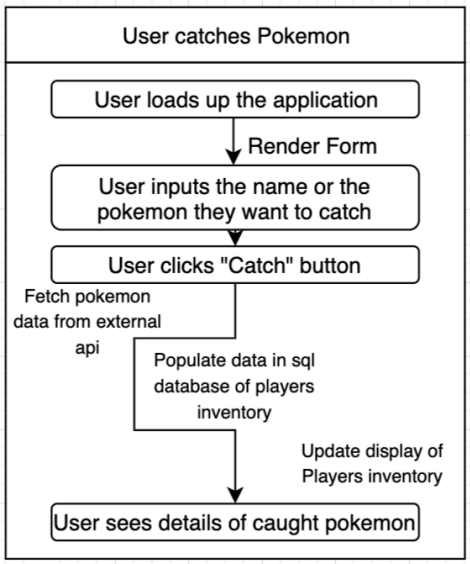


Description here

The diagram above shows 3 object diagrams using the model of the restaurant used in the previous evidence.

Unit	Ref	Evidence
A&D	A.D.4	Produce an Activity Diagram

Paste Screenshot here



Description here

The Screenshot here details the process for a user to add a pokemon to their inventory. This involves inputting the pokemon name and submitting the form. From then its all handled by the software which fetches data from an external api, uses this information to populate a local database which is then used to render information on the page.

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

Paste Screenshot here

Category	Constraint	Solution
Hardware and Software Platforms	SQL database structure was clunky and unsuited to the data structure	Update database using MongoDB to have a more suitable structure.
Performance Requirements	Fetching large amounts of data from an external API and rendering it all on the starting page of the app caused a slow load time	Implement a home page before the page with api information to allow it time to load correctly before the user can see the info.
Persistent storage and transactions	User needs to create a database using sql structure in server side of code	Host the database on a web server to reduce storage used on users pc and reduce setup time of app as a whole
Usability	Parts of the site that have onClick events aren't clearly defined and may not be clear that they can be clicked on	Use css hover to add clear indication that an element is clickable
Budget	Requires purchasing an API key to access the API	Create own api with only the information needed to run the app.
Time	Only 5 days to create the app	Planned the application modularly so that once the MVP was complete, new functionality can be added without affecting what is already there.

Description here

The above table details constraints that were encountered when creating my individual pokemon project and the solutions found to get around these issues.

Week 5

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

Paste Screenshot here

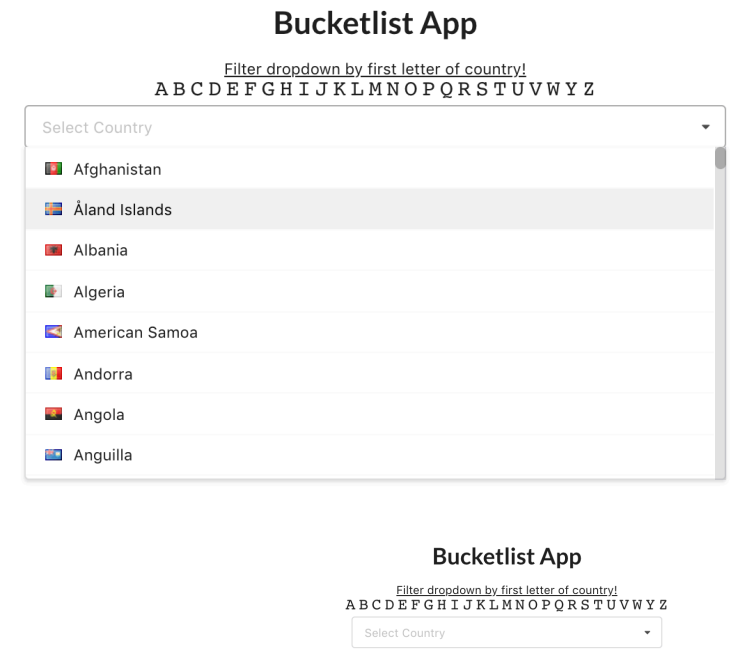
```
displayPoke(pokemon) {
  //
  // create thumbnail container
  //
  // add image to thumbnail container
  //
  // add pokemon name to container
  //
  // add pokemon's current lvl to the container
  //
  // add the pokemons types to the container
  //
  // add a button to withdraw a pokemon to my inventory
  //
  // limit number of pokemon in inventory to 6
  //
  // add pokemon to inventory
  //
  // remove pokemon from library
  //
  // remove highlight from pokemon thats withdrawn
  //
  })
```

Description here

This is an example of pseudocode used in my individual project to create the elements to render information about a Pokemon on the page.

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

Paste Screenshot here



Description here

Here you can see that a user can select a country from the dropdown at the start of the page and it will add that country to the users “bucketlist”.

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

Paste Screenshot here

```
router.post('/', function(req, res) {
  SqlRunner.run("INSERT INTO pokemon (natno, name, types, height, weight, abilities) VALUES ($1, $2, $3, $4, $5, $6)", [req.body.natno,
  req.body.name, req.body.types, req.body.height, req.body.weight, req.body.abilities]).then(result => {
    SqlRunner.run("SELECT * FROM pokemon ORDER BY id ASC")
      .then((result) => {
        res.status(201).json(result.rows);
      });
  });
});
```

localhost › projectPokemon › pokemon Connected. PostgreSQL 11.2 LOCAL

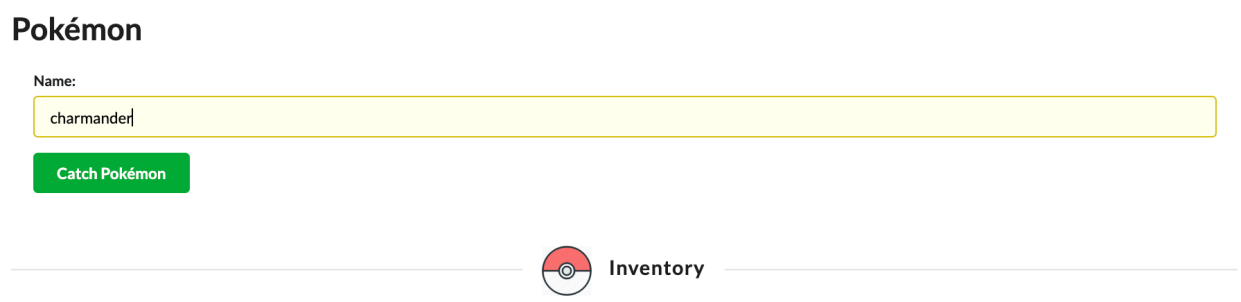
id	natno	name	types	height	weight	abilities
17	#004	Charmander	fire	0.6	8.5	+blaze+solar-power-h

Description here

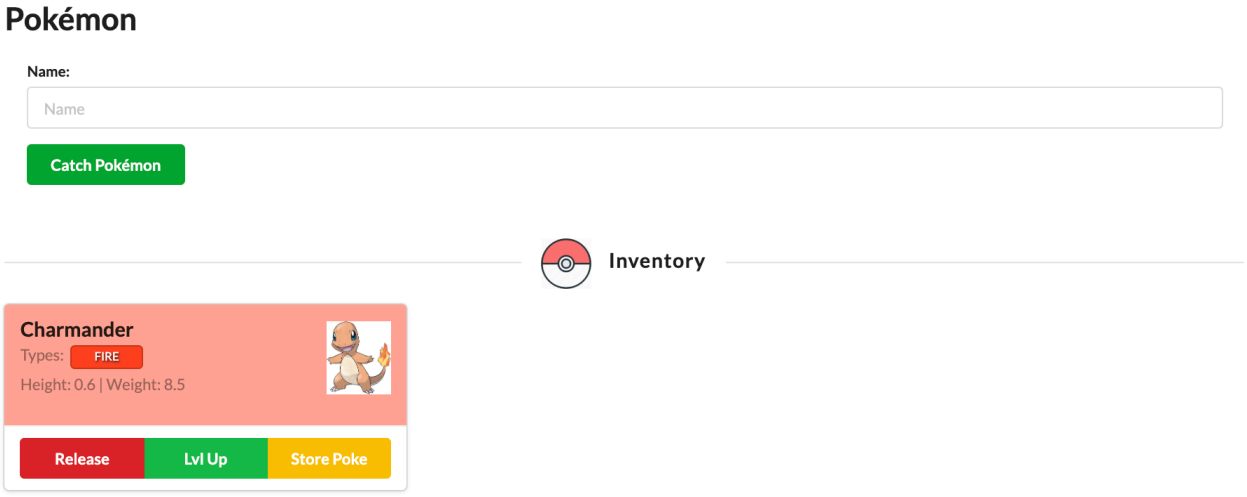
Here you can see sql code that adds a pokemon to a database using stats provided by a user or in this case an external API. The second screenshot is of the database once a pokemon was added.

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

Paste Screenshot here



Returns:



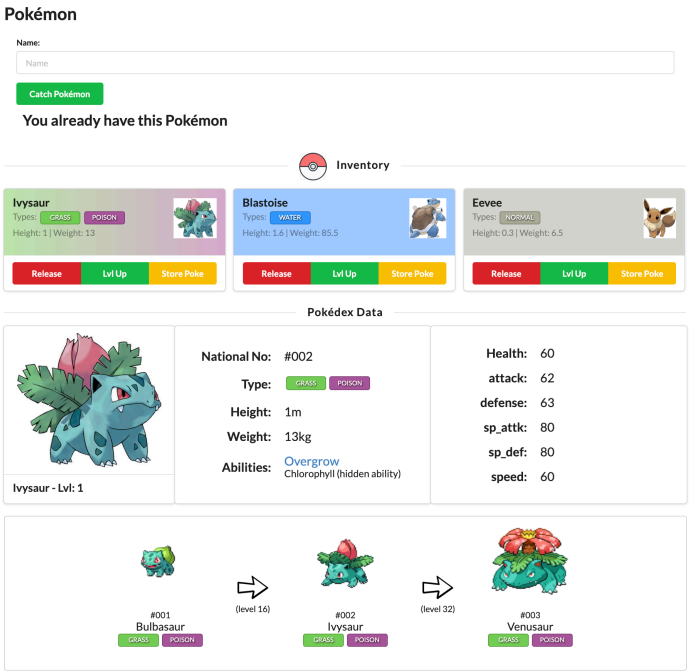
Description here

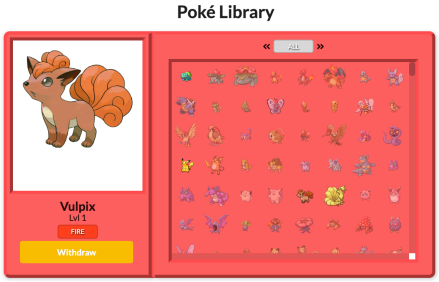
User requests a Pokemon from an external api, the app returns the Pokemon along with specified stats

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

https://github.com/ergilly/projectPokemon



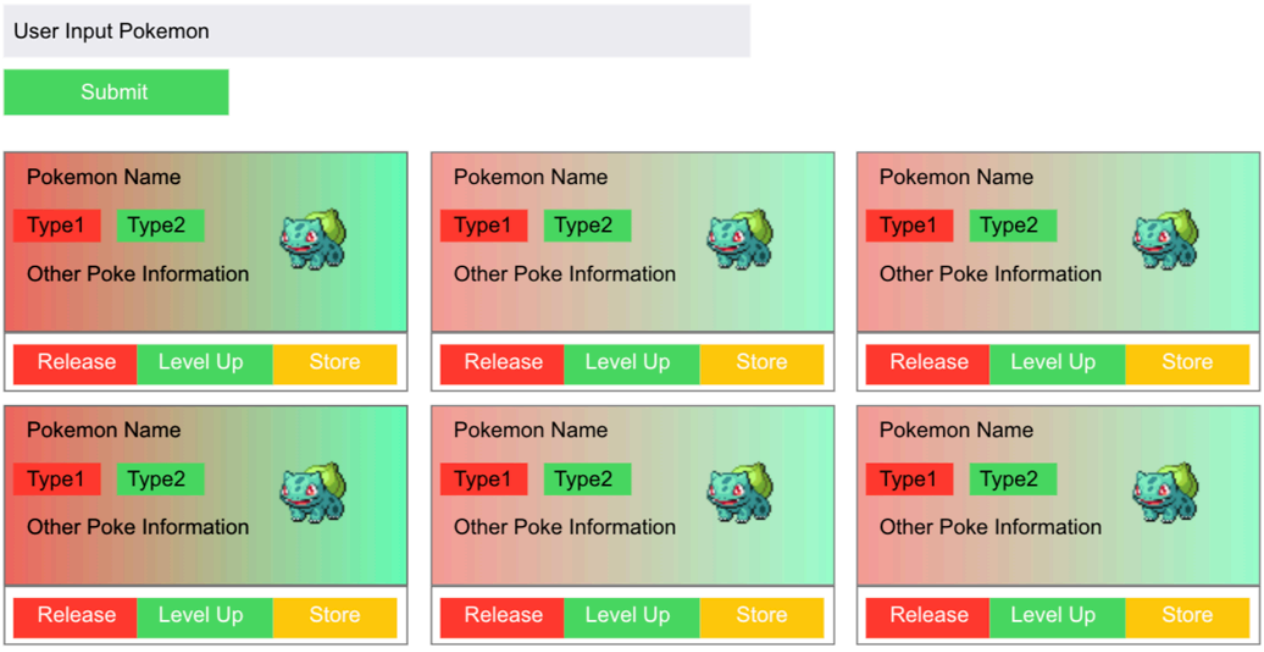


Description here

This project was created as a way for someone to keep track of the pokemon they catch whilst playing the game Pokemon. It has multiple functions including adding a pokemon to the players Inventory or library and also includes features from the game, i.e. a player can only have a maximum inventory of 6 pokemon, after which, pokemon are added to the library instead.

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
Concept Plan:
MVP:



Ext. 1

Ext. 2

Ext. 3

Description here

The above 4 steps show the modular design stages I created when approaching my individual project. The first screenshot is of the MVP of a players inventory with pokemon being added as its functionality. Once this was complete I added a further information view as the first extention, providing information of the selected Pokemon's stats. Next was designing the user's Library, allowing for more pokemon to be stored without cluttering the page. And finally an Evolution tree was added as the last extension to allow players to see the evolutions of pokemon they caught and also the requirements of the evolution.

Week 7

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

Paste Screenshot here

Bug Description	Date Discovered	Severity	Status	Date Resolved
Only pokemon stats are removed when delete button pressed	29/03/2019	Database corruption	Resolved - Fixed order of which database entries are deleted.	29/03/2019
Catching pokemon would sometimes catch multiple	30/03/2019	Caused previous bug to reappear	Resolved - added limiter IF statement to prevent multiple API calls	30/03/2019
When running program on another computer, sprites would not render	01/04/2019	Minor	Unresolved due to time restraints, caused by images being stored locally and not uploaded to GitHub due to avoid oversized repo.	N/A
A specific pokemon with multiple evolutions would not show any in evolution tree.	02/04/2019	Minor - only 1 of 151 pokemon affected	Resolved - Had spare time so added functionality for multiple evolutions with a loop through the API's array.	02/04/2019

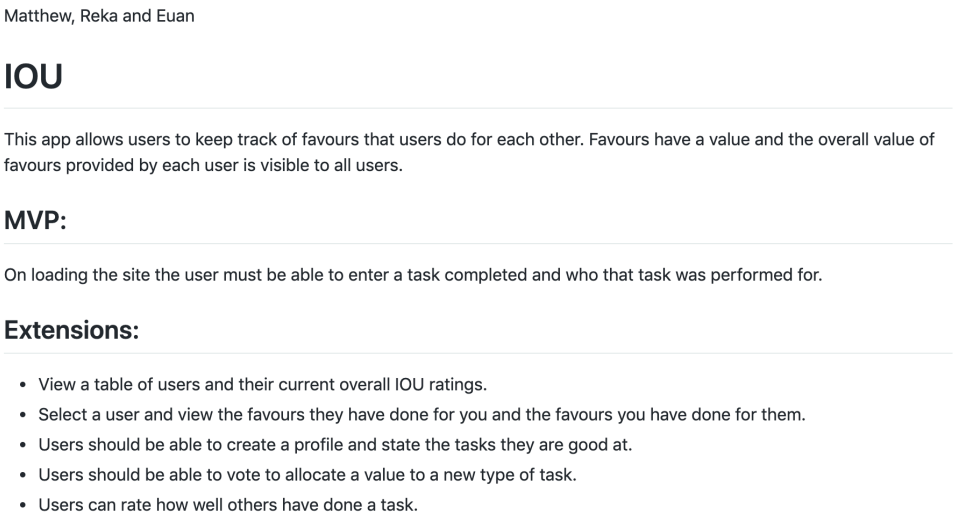
Description here

The above table details bugs encountered during my individual project and how they were resolved.

Week 9

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

Paste Screenshot here



Matthew, Reka and Euan

IOU

This app allows users to keep track of favours that users do for each other. Favours have a value and the overall value of favours provided by each user is visible to all users.

MVP:

On loading the site the user must be able to enter a task completed and who that task was performed for.

Extensions:

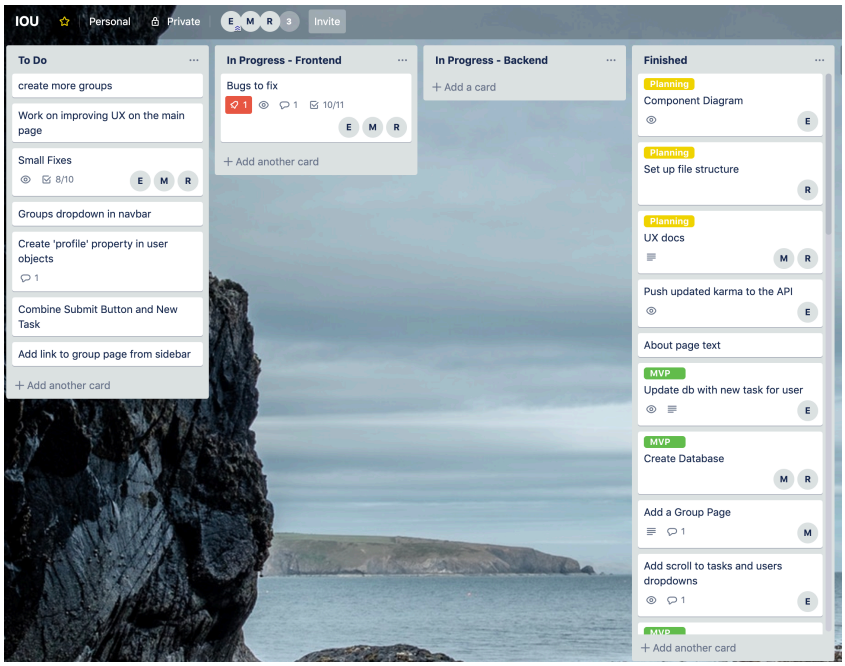
- View a table of users and their current overall IOU ratings.
- Select a user and view the favours they have done for you and the favours you have done for them.
- Users should be able to create a profile and state the tasks they are good at.
- Users should be able to vote to allocate a value to a new type of task.
- Users can rate how well others have done a task.

Description here

The above screenshot is of the brief for our group project, this breed was created on the first day of the project and was followed carefully when deciding which additional features should be prioritised for implementation

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

Here is a screenshot of the Trello board from the end of our Group Project. As you can see we used coloured tags to show when card was specifically MVP, Planning, Extension or Testing. We also assigned different cards to specific people to divide the workload evenly and also give each member of the team the ability to work on what they individually wanted to improve on. By the end of the project, due to time constraints there were still small bugs to be fixed and also features that we were not able to implement.

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

Paste Screenshot here

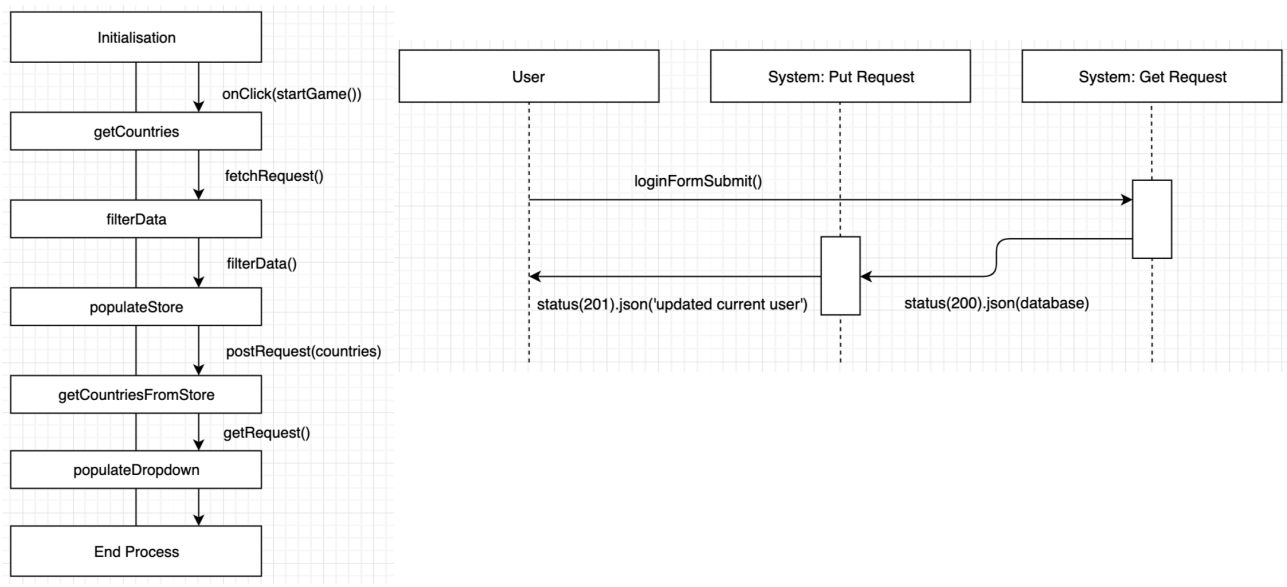
Acceptance Criteria	Expected Result	Pass/Fail
User should be able to add a new pokemon	User types in pokemon name and clicks “catch” pokemon shows up in users inventory	Pass
User should not be able to have more than 6 pokemon in their inventory	User catches a pokemon when they have 6 pokemon already. Pokemon should not be added to inventory	Pass
When Inventory is full pokemon should be added to the Library	User catches a pokemon when they have 6 pokemon already. Pokemon should be added to library	Pass
User should be able to move a pokemon from their inventory to the library	User presses Store button, Pokemon is no longer in inventory. Pokemon Is shown In library	Pass
User should be able to move a pokemon from their library to the inventory	User presses Withdraw button. Pokemon is no longer in Library. Pokemon is shown in inventory	Pass

Description here

Above is a table of the test planned for adding and moving pokemon within my individual project application

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

Paste Screenshot here



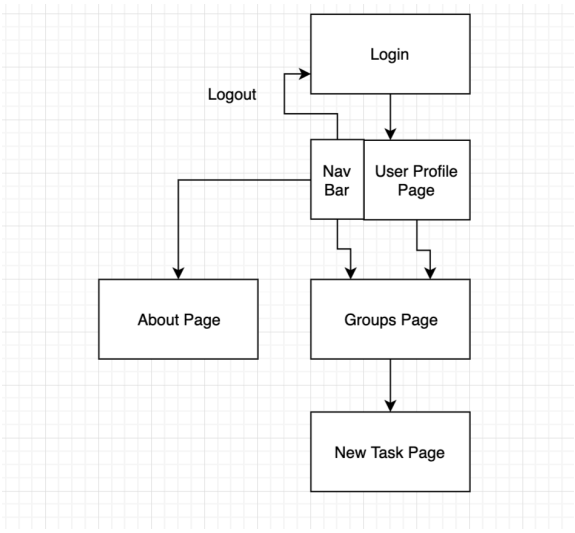
Description here

The screenshot on the left details how my bucketList program populated the dropdown. First it gets the countries from an external api, then filters this data into what is needed in the app, this then gets saved onto the local database so it can be sorted. This data is then fetched using a get Request to populate the dropdown.
The Screenshot on the right details a login function where a user types in their username, this is then used to get the user's information from the database, then the put request is run to change the 'isCurrent' element to 'true' for that user in the database.

Week 10

Unit	Ref	Evidence
P	P.5	User Site Map

Paste Screenshot here

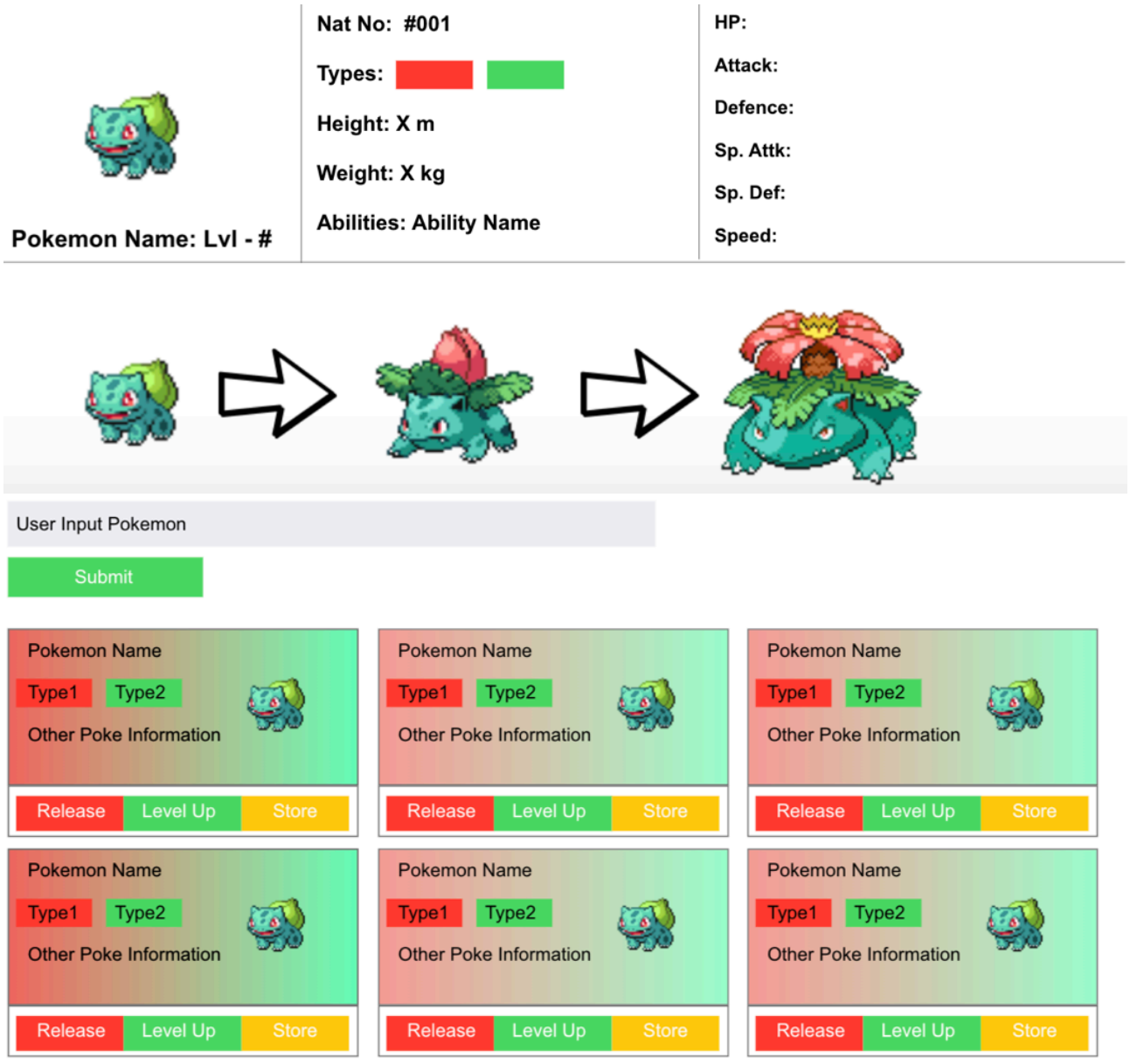


Description here

The user Site map above shows the route the user goes through when using the Application. I included a nav bar attached to the User Profile page to show the connections that the nav bar has, this nav bar is visible in the Groups, About and New Task page to help the user navigate the site.

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

Paste Screenshot here

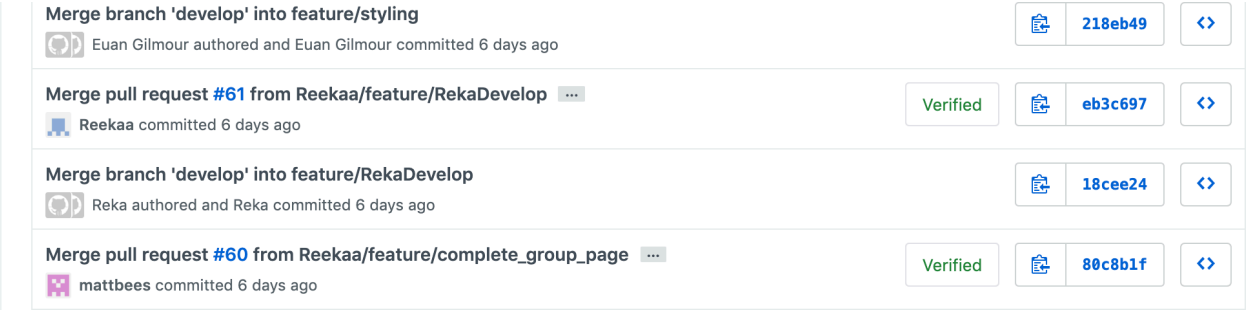


Description here

The two wireframes above were the initial design of my individual project, in the original design these were meant to be two separate pages but since this was before we were using react and react router it was decided to eventually combine them.

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

Above is a screenshot of a small section of the GitHub commits page for our group project.

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

Paste Screenshot here

```
import java.util.*;

public class MusicShop {
  private ArrayList<IPlay> stock;
  private String name;

  public MusicShop(String name){
    this.name = name;
    this.stock = new ArrayList<IPlay>();
  }

  public void addInstrument(IPlay instrument) {
    this.stock.add(instrument);
  }

  public String getName(){
    return this.name;
  }

  public ArrayList<IPlay> getInstruments(){
    return this.stock;
  }

  public static void main(String[] args) {
    MusicShop redDog = new MusicShop("Red Dog Music");
    Drums d = new Drums("Timpani", 25);
    Guitar g = new Guitar("Takamine", 12);
    Trumpet t = new Trumpet("Yamaha", "C");

    redDog.addInstrument(d);
    redDog.addInstrument(g);
    redDog.addInstrument(t);

    System.out.println(redDog.getName());
    for (int i=0; i<redDog.getInstruments().size() ; i++ ) {
      System.out.println(redDog.getInstruments().get(i).playSound());
    }
  }
}
```

```
public class Trumpet implements IPlay {

  private String name;
  private String tuning;

  // constructor
  public Trumpet(String name, String tuning){
    this.name = name;
    this.tuning = tuning;
  }

  public String playSound(){
    return "Toot!";
  }

}
```

```
public class Guitar implements IPlay {

  private String name;
  private int strings;

  // constructor
  public Guitar(String name, int strings){
    this.name = name;
    this.strings = strings;
  }

  public String playSound(){
    return "Twang";
  }

}
```

```
public class Drums implements IPlay {

  private String name;
  private int size;

  // constructor
  public Drums(String name, int size){
    this.name = name;
    this.size = size;
  }

  public String playSound(){
    return "Boom";
  }

}
```

```
public interface IPlay{
  public String playSound();
}
```

Description here

The IPlay interface is used to allow polymorphism in the MusicShop class. Allowing other variables (Trumpet, Guitar and Drums) to be used to build an array through the IPlay interface even though their variables are different.