

PDA: Software Development
Level 8
Student Evidence Checklist

Full name	Christopher Murphy
Cohort	G4

The evidence required can be taken from your assignments, homework that you have completed on your own or by creating a specific example for the PDA.

	Unit	Ref.	Evidence	Done
Week 2	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	
			<pre>const RecordCollector = function(name,cash){ this.name = name; this.collection = [] this.cash = cash; }</pre>	
			<pre>RecordCollector.prototype.buy = function (record) { if(this.cash > record.price){ this.cash -= record.price; this.collection.push(record) } };</pre>	

		<pre>it('record collector can buy a record', function(){ recordCollector.buy(record1); assert.strictEqual(recordCollector.cash, 61); assert.deepStrictEqual(recordCollector.collection, [record1]); })</pre>	<ul style="list-style-type: none"> ✓ record store is initially empty ✓ can add records ✓ can show store balance ✓ can print records details ✓ store can list its inventory ✓ store can sell a record ✓ store can show finances ✓ store can view records by genre ✓ record collector can buy a record 	
I & T	I.T 6			
I & T		Static and Dynamic testing task A		

	Unit	Ref.	Evidence	Done
	I & T	I.T 3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running	

```
public static <T> List<T> getAll(Class classType){  
    session = HibernateUtil.getSessionFactory().openSession();  
    List<T> results = null;  
    Criteria criteria = session.createCriteria(classType);  
    results = getList(criteria);  
    return results;  
}
```

Week 3

```
List<Team> foundTeams = DBHelper.getAll(Team.class);
```

```
foundTeams = {ArrayList@2167} size = 5  
► 0 = {Team@2175}  
► 1 = {Team@2176}  
► 2 = {Team@2177}  
► 3 = {Team@2178}  
► 4 = {Team@2179}
```

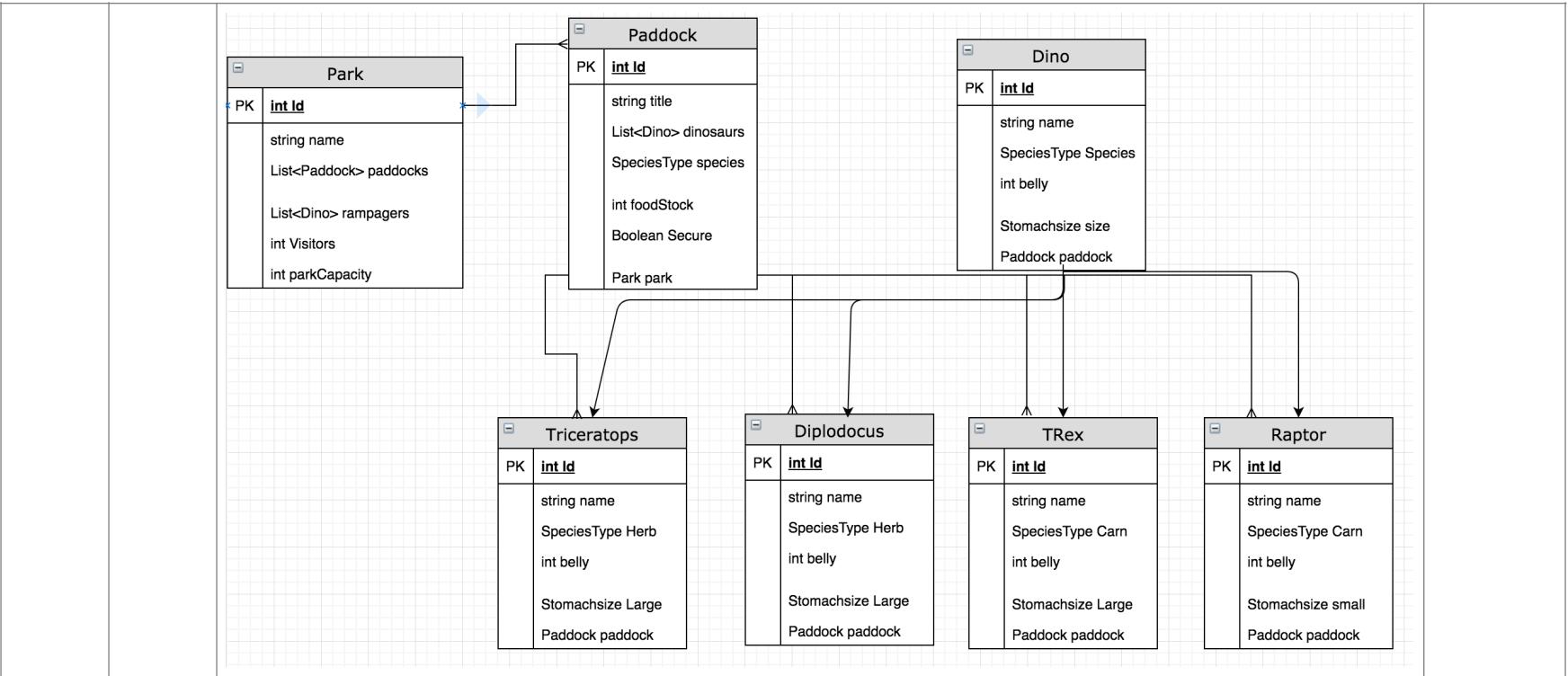
I & T I.T 4 Demonstrate sorting data in a program. Take screenshots of:
*Function that sorts data
*The result of the function running

```
public static List<Team> getTeamsInLeague(League league){  
    session = HibernateUtil.getSessionFactory().openSession();  
    List<Team> results = null;  
    Criteria criteria = session.createCriteria(Team.class);  
    criteria.add(Restrictions.eq("league", league));  
    criteria.addOrder(Order.desc("points"));  
    results = getList(criteria);  
    return results;  
}
```

```
List<Team> teamsfoundInPointsOrder = DBHelper.getTeamsInLeague(league);
```

```
▼ ┌─ teamsfoundInPointsOrder = {ArrayList@2170} size = 5
  ▼ ┌─ 0 = {Team@2183}
    ┌─ f id = 5
    ▶ f name = "Barcelona"
    ┌─ f points = 20
    ┌─ f manager = null
    ▶ f league = {League@2189}
  ▼ ┌─ 1 = {Team@2184}
    ┌─ f id = 4
    ▶ f name = "Newcastle"
    ┌─ f points = 19
    ┌─ f manager = null
    ▶ f league = {League@2189}
  ▼ ┌─ 2 = {Team@2185}
    ┌─ f id = 1
    ▶ f name = "soccer united"
    ┌─ f points = 12
    ┌─ f manager = null
    ▶ f league = {League@2189}
  ▼ ┌─ 3 = {Team@2186}
    ┌─ f id = 3
    ▶ f name = "Man blues"
    ┌─ f points = 7
    ┌─ f manager = null
    ▶ f league = {League@2189}
  ▼ ┌─ 4 = {Team@2187}
    ┌─ f id = 2
    ▶ f name = "Man reds"
    ┌─ f points = 3
    ┌─ f manager = null
    ▶ f league = {League@2189}
```

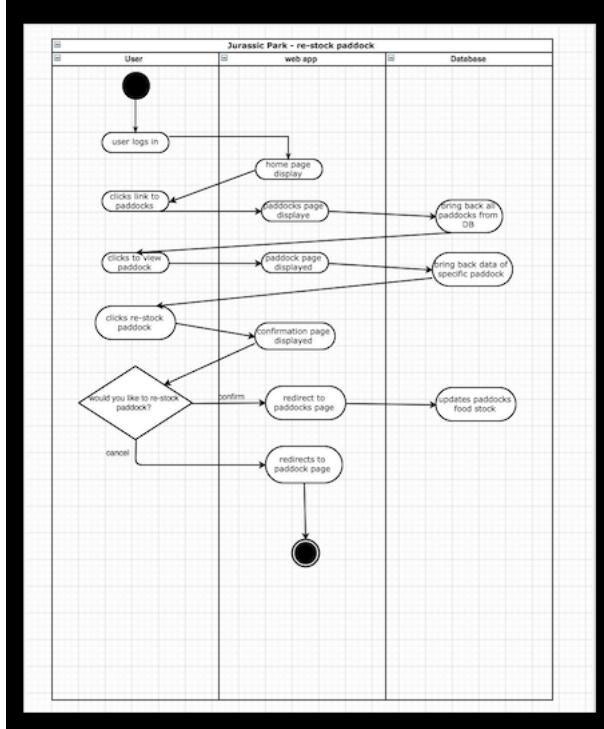
Unit	Ref.	Evidence	Done
A & D	A.D 1	A Use Case Diagram	
		<pre> graph LR Actor((Actor)) --> ViewInv(view inventory) Actor --> ViewStock(view stock levels) Actor --> UpdateStock(update stock levels) ViewInv --> ViewStock ViewStock --> UpdateStock ViewStock --> Database[Database] UpdateStock --> Database </pre>	
A & D	A.D 2	A Class diagram.	



A & D A.D 3 An Object diagram.



A & D A.D 4 An Activity Diagram



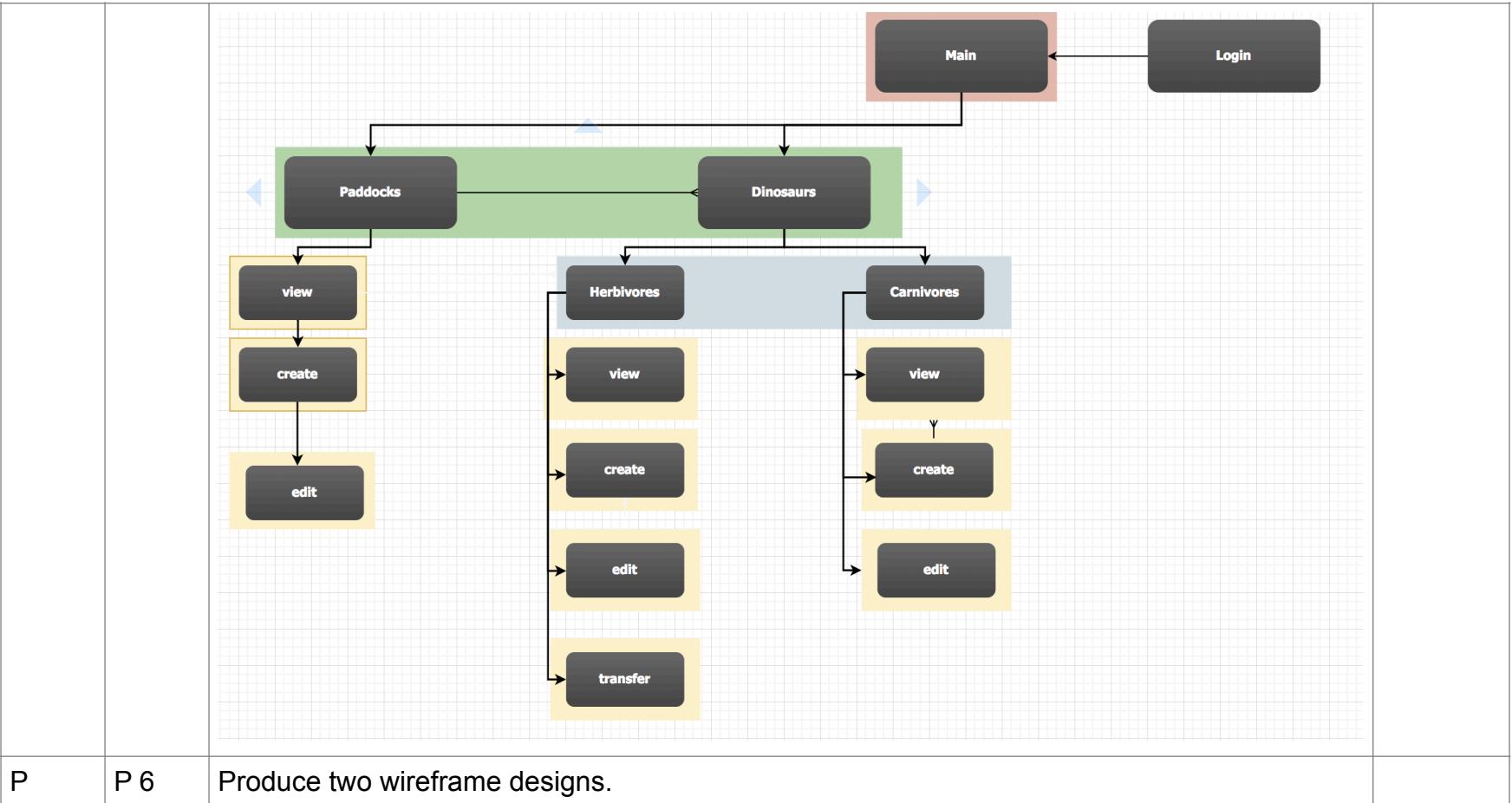
A & D

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 Constraints			
	Topic	Possible effect	Solution
	Hardware/Software Platforms	No Constraints on hardware or software platforms at this stage	n/a
	Performance Requirements	Application can only be accessed via localhost	Gateway can be set-up to be hosted on another port.
	Persistent Storage And transactions	No storage issues as application does not require large amounts of data to be stored	n/a
	Usability	Application is being developed for one user at a time	Could add a login/admin to allow for multiple users to access the app.
	Budgets	Budget could be set on applications design	More developers could be brought in to add further functionality
	Time Limitations	Project Submission Deadline	Post submission, further functionality can be added to the application
P	P 5	Create a user sitemap.	



P

P 6

Produce two wireframe designs.

Week 5

Paddocks

paddock	Type	Status	view	update
Paddock 1	Herbivore	Food Stock Low!	view	update
paddock 2	Carnivore	Safe	view	update
paddock 3	Carnivore	No Stock! Re-stock immediately	view	update

[Add Paddock](#)

		<p>Dyno Dino's</p> <p><u>Dinos</u></p>  <p><u>Paddocks</u></p>  <p>Days since last Rampage</p> <p>5</p> <p>Status: Park Safe to Open</p> <p>Open Park</p> <p>Visitors</p> <p>5</p> 	
P	P 10	Take a screenshot of an example of pseudocode for a function.	

```

let newArray = [];
// for each index position in arr.
for(let i in arr){
    // if the index number is less than the number passed in.
    if(i < index){
        // push those elements into the newArray.
        newArray.push(arr[i]);
    }else{
        // then push itemToAdd to the end of the newArray.
        newArray.push(itemToAdd);
        //then push the remaining elements to the end of newArray.
        newArray.push(arr[i]);
    }
}

```

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 clicks new raptor link

The screenshot shows a table titled "Raptors" with columns: Name, Species, Status, and Paddock. The rows contain data for Echo, Delta, Charlie, and Blue, all categorized as CARNIVORE. The status for all is "Raging: Check Raptors Paradise food store". The paddock for all is "Raptors Paradise". Action buttons (View, edit, Delete) are provided for each row. Below the table, a message says "Park Visitors : 0".

Name	Species	Status	Paddock	
Echo	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete
Delta	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete
Charlie	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete
Blue	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete

Park Visitors : 0

Inputs the name of the new raptor and paddock and clicks save

The screenshot shows a browser window with the URL "localhost:4567/raptors/new". The page title is "New Raptor". It has input fields for "Name" (containing "Greg") and "Paddock" (containing "Raptors Paradise"). There are "Save" and "Cancel" buttons. Below the form, a message says "Park Visitors : 0".

The input is saved to the database which is confirmed when viewed on raptors page.

The screenshot shows the same table as the first one, but now it includes a new row for "Greg" at the top of the list. The data for Greg is identical to the others: Species CARNIVORE, Status "Raging: Check Raptors Paradise food store", Paddock "Raptors Paradise", and action buttons. The other four rows (Echo, Delta, Charlie, Blue) remain the same. The message "Park Visitors : 0" is also present.

Name	Species	Status	Paddock	
Greg	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete
Echo	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete
Delta	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete
Charlie	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete
Blue	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View edit Delete

Park Visitors : 0

P	P 14	<p>Show an interaction with data persistence. Take a screenshot of:</p> <ul style="list-style-type: none"> * Data being inputted into your program * Confirmation of the data being saved 	
		<p>This diagram shows a seeds file which on launch of the application stores this set of dinosaurs to the database</p>  <pre> 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 DBHelper dippy1 = new Diplodocus("LittleFoot", paddock1); DBHelper.saveOrUpdate(dippy1); Triceratops tri1 = new Triceratops("Cera", paddock1); DBHelper.saveOrUpdate(tri1); Triceratops tri2 = new Triceratops("Woog", paddock1); DBHelper.saveOrUpdate(tri2); TRex bigT1 = new TRex("Rex", paddock9); DBHelper.saveOrUpdate(bigT1); Raptor raptor1 = new Raptor("Blue", paddock3); DBHelper.saveOrUpdate(raptor1); Raptor raptor2 = new Raptor("Charlie", paddock3); DBHelper.saveOrUpdate(raptor2); Raptor raptor3 = new Raptor("Delta", paddock3); DBHelper.saveOrUpdate(raptor3); Raptor raptor4 = new Raptor("Echo", paddock3); DBHelper.saveOrUpdate(raptor4); </pre>	
P	P 15	<p>Show the correct output of results and feedback to user. Take a screenshot of:</p> <ul style="list-style-type: none"> * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program 	

User clicks view to see details for Herbs Green

localhost:4567/paddocks

Apps Free Online YouTube... Free Online YouTube... Musicroom.com - S... Registration - MyS... http://www.lenzie-p... Come As You Are ... Suggested Sites Outlook - thom.mur... Other Bookmarks

Home

Logged in as: Chrism_lugo

[Logout](#)

All Paddocks

New Paddock

Name	Species	Food Stock	Secure	View	edit	Delete
Herbs Green	HERBIVORE	0	false	View	edit	Delete
Rex Zone	CARNIVORE	0	false	View	edit	Delete
Raptors Paradise	CARNIVORE	0	false	View	edit	Delete

Park Visitors : 0

Link Navigates to paddock page for Herb's Green

localhost:4567/paddocks/5

Apps Free Online YouTube... Free Online YouTube... Musicroom.com - S... Registration - MyS... http://www.lenzie-p... Come As You Are ... Suggested Sites Outlook - thom.mur... Other Bookmarks

Home

Logged in as: Chrism_lugo

[Logout](#)

Paddock: Herbs Green

Re-Stock Paddock

Dinos

Name	Species	Status
LittleFoot	HERBIVORE	Unhappy: Check Herbs Green food store or check for paddock transfer
Cera	HERBIVORE	Unhappy: Check Herbs Green food store or check for paddock transfer
Woog	HERBIVORE	Unhappy: Check Herbs Green food store or check for paddock transfer

Park Visitors : 0

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	
---	------	---	--

```
Project — ~/codecademy/JavaScript/week_4/day_2/hogwarts.lib/hogwarts
Project — ~/codecademy/JavaScript/homework/record_store + ↳ record_store git:(master) ✘ npm test
> record_store@1.0.0 test /Users/chrism_lugo/codecademy/JavaScript/homework/record_store
  ✓ record store is initially empty
  ✓ can add records
  ✓ can show store balance
  ✓ can print records details
  ✓ store can list its inventory
  ✓ store can sell a record
  ✓ store can show finances
  ✓ store can view records by genre
  ✓ record store can view records by genre
    ✓ record store can view records by genre
      ✓ record store can view records by genre
        ✓ record store can view records by genre
          ✓ record store can view records by genre
            ✓ record store can view records by genre
              ✓ record store can view records by genre
                ✓ record store can view records by genre
                  ✓ record store can view records by genre
                    ✓ record store can view records by genre
                      ✓ record store can view records by genre
                        ✓ record store can view records by genre
                          ✓ record store can view records by genre
                            ✓ record store can view records by genre
                              ✓ record store can view records by genre
                                ✓ record store can view records by genre
                                  ✓ record store can view records by genre
                                    ✓ record store can view records by genre
                                      ✓ record store can view records by genre
                                        ✓ record store can view records by genre
                                          ✓ record store can view records by genre
                                            ✓ record store can view records by genre
                                              ✓ record store can view records by genre
                                                ✓ record store can view records by genre
                                                  ✓ record store can view records by genre
                                                    ✓ record store can view records by genre
                                                      ✓ record store can view records by genre
                                                        ✓ record store can view records by genre
                                                          ✓ record store can view records by genre
                                                            ✓ record store can view records by genre
                                                              ✓ record store can view records by genre
                                                                ✓ record store can view records by genre
                                                                  ✓ record store can view records by genre
                                                                    ✓ record store can view records by genre
                                                                      ✓ record store can view records by genre
                                                                        ✓ record store can view records by genre
                                                                          ✓ record store can view records by genre
                                                                            ✓ record store can view records by genre
                                                                              ✓ record store can view records by genre
                                                                                ✓ record store can view records by genre
                                                                                  ✓ record store can view records by genre
                                                                                    ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
                                                                                      ✓ record store can view records by genre
................................................................
7 passing (8ms)
1 failing

1) Record Store
   store can view records by genre:
     TypeErrot: recordStore.findRecordsByGenre is not a function
       at Context.<anonymous> (specs/Record_store_spec.js:87:40)

npm ERR! Test failed. See above for more details.
→ record_store git:(master) ✘
```

Project — ~/codeclean_work/JavaScript/week_4/day_2/hogwarts_lab/hogwarts

RecordStore.js — ~/codeclean_work/JavaScript/homework/record_store

```
+ ➔ record_store git:(master) ✘ npm test
> record_store@0.8.0 test /Users/chrismlugo/codeclean_work/JavaScript/homework/record_store
> mocha specs

Record Store
✓ record store is initially empty
✓ can add records
✓ can show store balance
✓ can print records details
✓ store can list its inventory
✓ store can sell a record
✓ store can show finances
✓ store can view records by genre

8 passing (8ms)
➔ record_store git:(master) ✘
```

Project Tree:

- record_store
- specs
- Record_store_spec.js
- .gitignore
- package-lock.json
- package.json
- Record.js
- RecordCollector.js
- RecordStore.js

File Content (RecordStore.js):

```
4  this.name = name;
5  this.city = city;
6  this.Inventory = {};
7  this.balance = 0
8  );
9
10
11 RecordStore.prototype.countRecords = function () {
12   return this.inventory.length;
13 }
14
15 RecordStore.prototype.addRecord = function (record) {
16   this.inventory.push(record);
17 };
18
19 RecordStore.prototype.recordDetails = function (record) {
20   return `artist: ${record.artist} title: ${record.title} genre: ${record.genre} price: ${record.price}`;
21 };
22
23 RecordStore.prototype.showInventory = function () {
24   return this.inventory;
25 };
26
27 RecordStore.prototype.sell = function (record) {
28   this.balance += record.price;
29   _.remove(this.inventory, record);
30 };
31
32 RecordStore.prototype.finances = function () {
33   let inventoryTotal = _.sumBy(this.inventory, 'price');
34   return `Store Balance: ${this.balance} inventory Total: ${inventoryTotal}`;
35 };
36
37 RecordStore.prototype.findRecordsByGenre = function (genre) {
38   return _.filter(this.inventory, {genre: genre});
39 };
```

	Unit	Ref.	Evidence	Done
	I & T	I.T 7	Demonstrate the use of Polymorphism in a program.	

mynewproject

src

java

Dino

```
package models;
import ...
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class Dino {
    private int id;
    private String name;
    private int belly;
    private SpeciesType species;
    private Paddock paddock;
    private StomachSize stomachSize;
    public Dino() {
    }
    public Dino(String name, SpeciesType species, Paddock paddock, StomachSize stomachSize) {
        this.name = name;
        this.species = species;
        this.paddock = paddock;
        this.belly = 0;
        this.stomachSize = stomachSize;
    }
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

at org.eclipse.jetty.util.component.AbstractLifeCycle.start(AbstractLifeCycle.java:69)
at spark.webserver.SparkServer\$Ignite.run(SparkServer.java:148)
at spark.webserver.SparkBase\$3.run(SparkBase.java:368)

Compilation completed successfully in 296ms (today 15:16)

mynewproject

src

java

Dino

```
import ...
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@Table(name = "raptors")
public class Raptor extends Dino {
    public Raptor() {
    }
    public Raptor(String name, Paddock paddock) {
        super(name, SpeciesType.CARNIVORE, paddock, StomachSize.SMALL);
    }
}
```

mynewproject

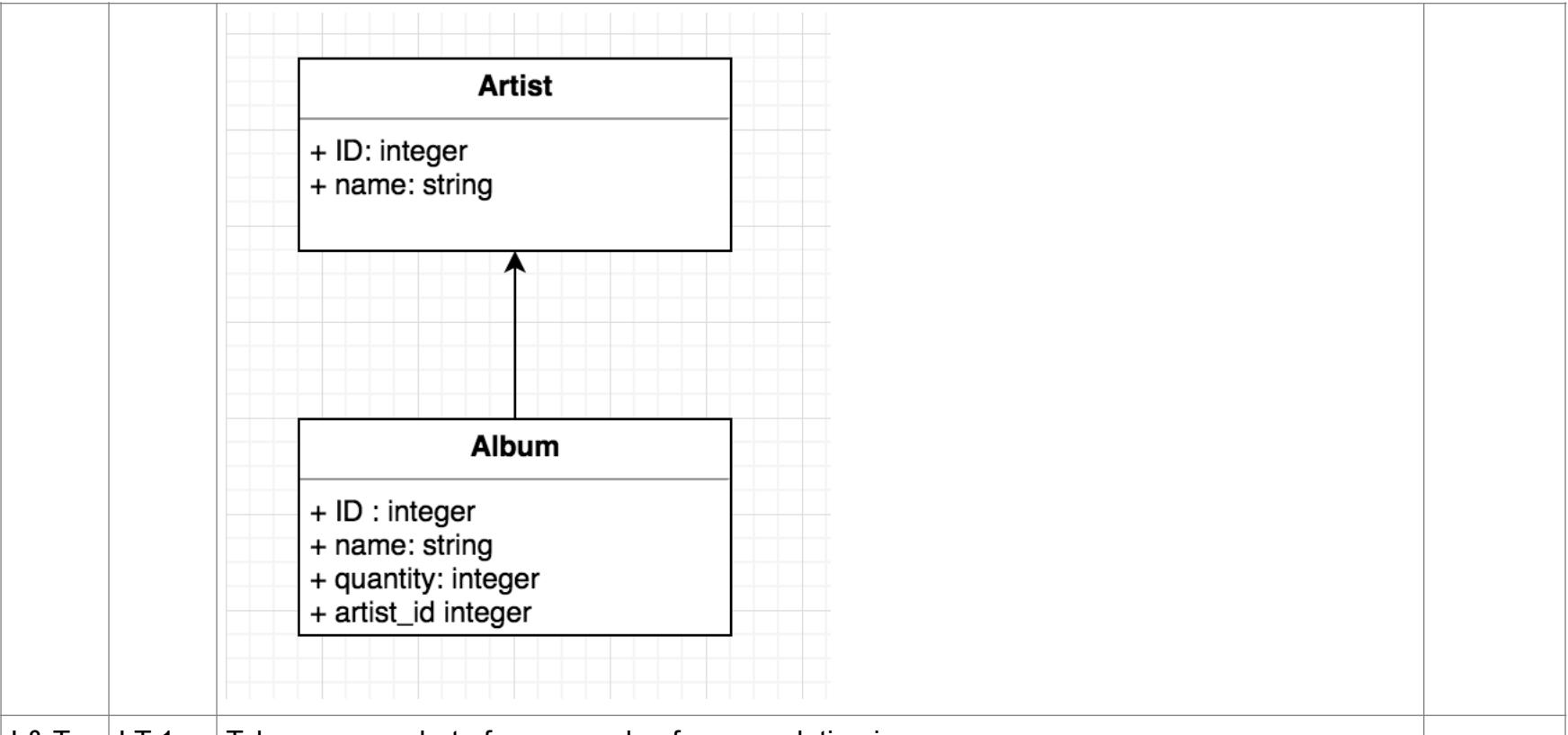
src

java

Paddock

```
import ...
@ManyToOne
@JoinColumn(name = "park_id", nullable = false)
public Park getPark() {
    return park;
}
public void setPark(Park park) {this.park = park;}
public void feedDinos(){
    for(Dino dino: this.dinosaurs){
        Random ran = new Random();
        int foodAmount = ran.nextInt((5) + 1);
        if(this.foodStock >= foodAmount) {
            dino.feed(foodAmount);
            this.foodStock -= foodAmount;
            DBHelper.saveOrUpdate(dino);
        }
    }
}
public void addDino(Dino dino){
    this.dinosaurs.add(dino);
}
```

A & D A.D 5 An Inheritance Diagram



I & T I.T 1 Take a screenshot of an example of encapsulation in a program.

The screenshot shows a Java code editor within an IDE. The code is for a **Paddock** class, which is annotated with `@Entity` and `@Table(name="paddocks")`. The class has private fields for `id`, `name`, a list of `Dino`s, `species`, `foodStock`, `paddockSecure`, and a `Park` object. The constructor takes `String name`, `SpeciesType species`, and a `Park park` parameter. The code editor shows the class definition and some associated methods like `toString()` and `getDinos()`.

```

package models.paddocks;
import ...
@Entity
@Table(name="paddocks")
public class Paddock {
    private int id;
    private String name;
    private List<Dino> dinosaurs;
    private SpeciesType species;
    private int foodStock;
    private boolean paddockSecure;
    private Park park;

    public Paddock() {}

    public Paddock(String name, SpeciesType species, Park park) {
        this.name = name;
        this.species = species;
        this.foodStock = 0;
        this.dinosaurs = new ArrayList<>();
        this.paddockSecure = true;
        this.park = park;
    }

    @Override
    public String toString() {
        return "Paddock{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", dinosaurs=" + dinosaurs +
                ", species=" + species +
                ", foodStock=" + foodStock +
                ", paddockSecure=" + paddockSecure +
                ", park=" + park +
                '}';
    }

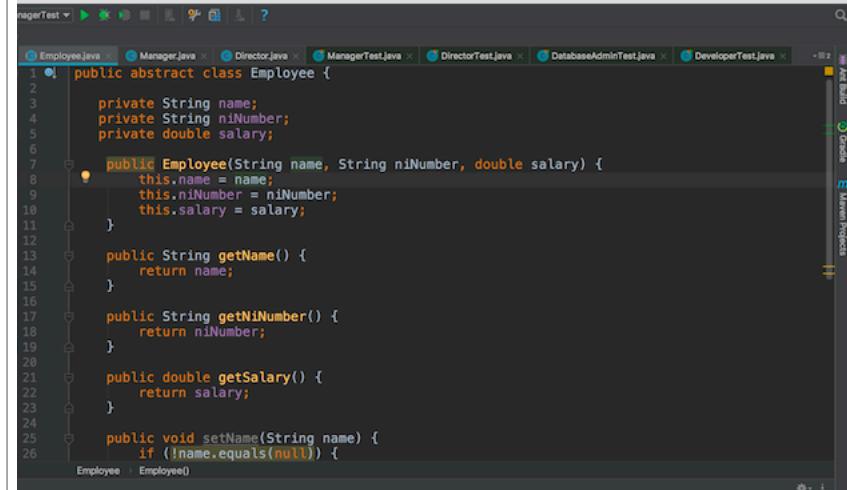
    public List<Dino> getDinos() {
        return dinosaurs;
    }
}

```

Week 7

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.	
-------	-------	--	--

This Employee Class contains all the functionality for all types of employee



```
public abstract class Employee {
    private String name;
    private String niNumber;
    private double salary;

    public Employee(String name, String niNumber, double salary) {
        this.name = name;
        this.niNumber = niNumber;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public String getNiNumber() {
        return niNumber;
    }

    public double getSalary() {
        return salary;
    }

    public void setName(String name) {
        if (!name.equals(null)) {
            this.name = name;
        }
    }

    public void setNiNumber(String niNumber) {
        if (!niNumber.equals(null)) {
            this.niNumber = niNumber;
        }
    }

    public void setSalary(double salary) {
        if (salary > 0) {
            this.salary = salary;
        }
    }
}
```

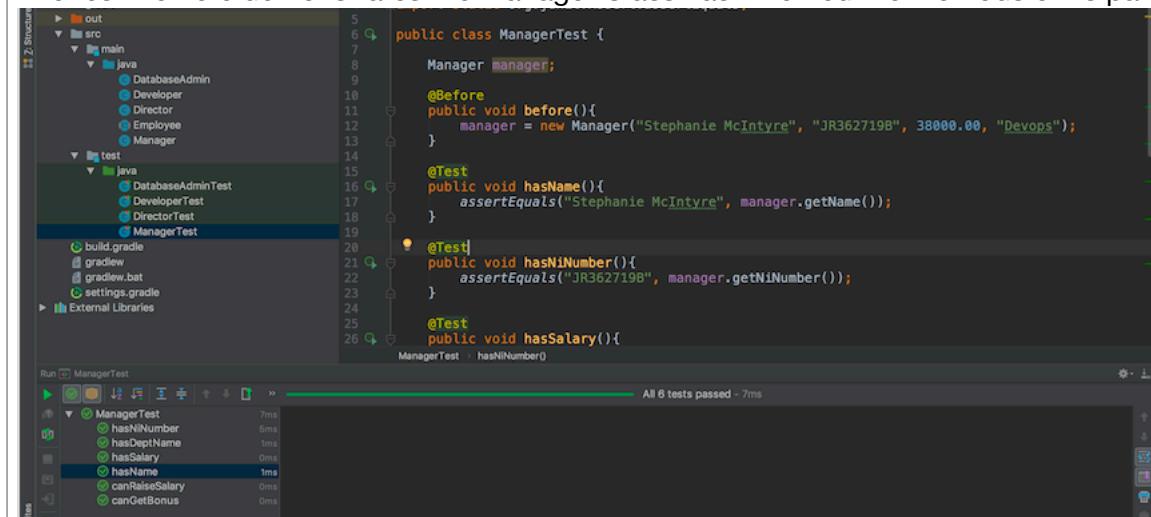
The Manager class shown here inherits from the Employee class.

```
public class Manager extends Employee {
    private String deptName;

    public Manager(String name, String niNumber, double salary, String deptName) {
        super(name, niNumber, salary);
        this.deptName = deptName;
    }

    public String getDeptName() {
        return deptName;
    }
}
```

The test file here demonstrates the manager Class has inherited the methods of its parent class.



```
public class ManagerTest {
    Manager manager;

    @Before
    public void before(){
        manager = new Manager("Stephanie McIntyre", "JR362719B", 38000.00, "Devops");
    }

    @Test
    public void hasName(){
        assertEquals("Stephanie McIntyre", manager.getName());
    }

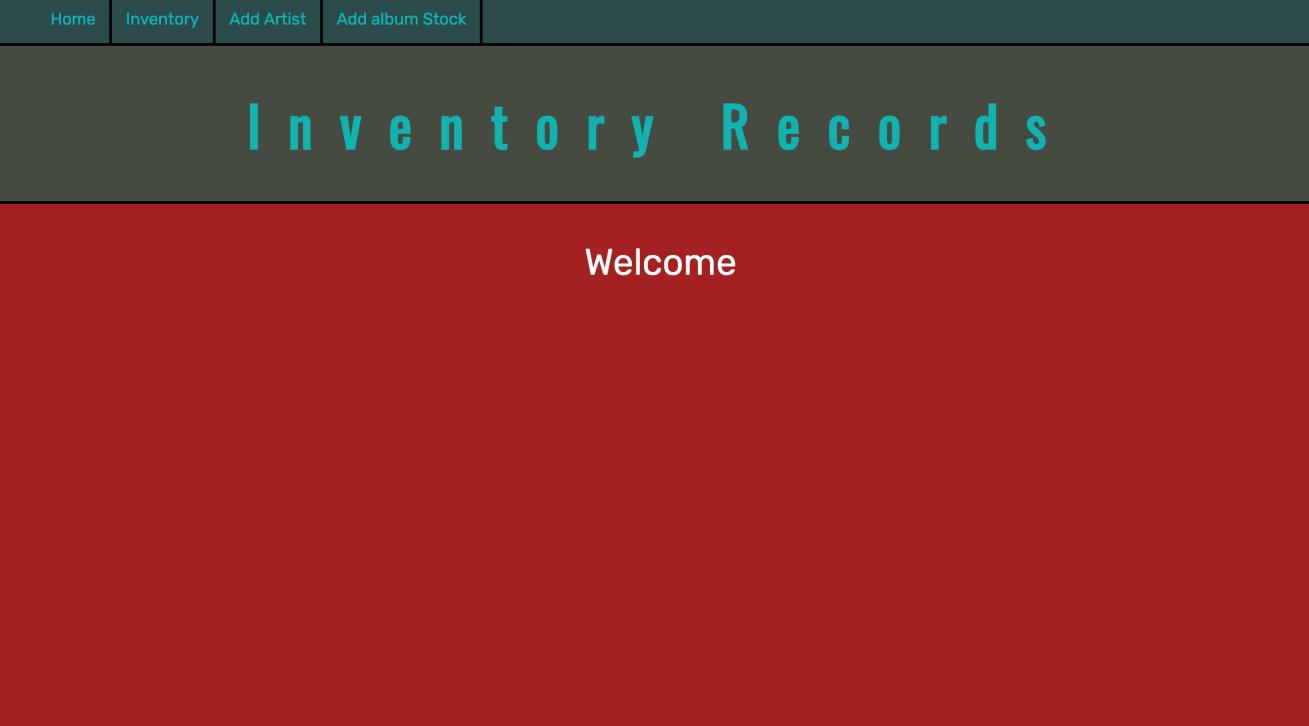
    @Test
    public void hasNiNumber(){
        assertEquals("JR362719B", manager.getNiNumber());
    }

    @Test
    public void hasSalary(){
        assertEquals(38000.0, manager.getSalary());
    }

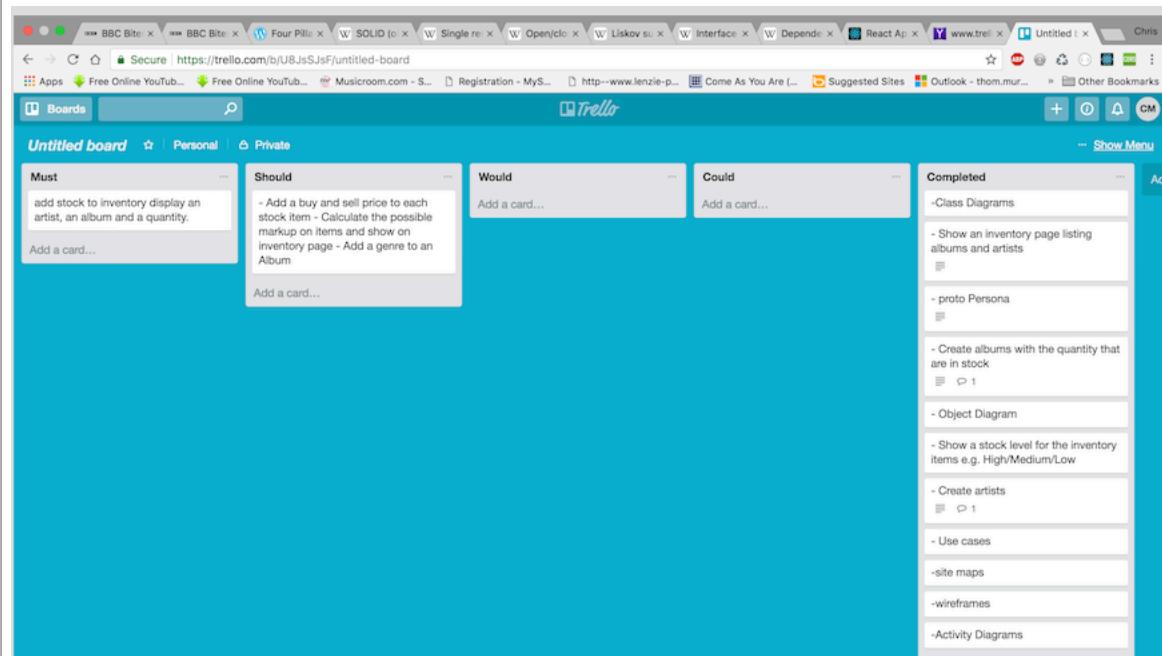
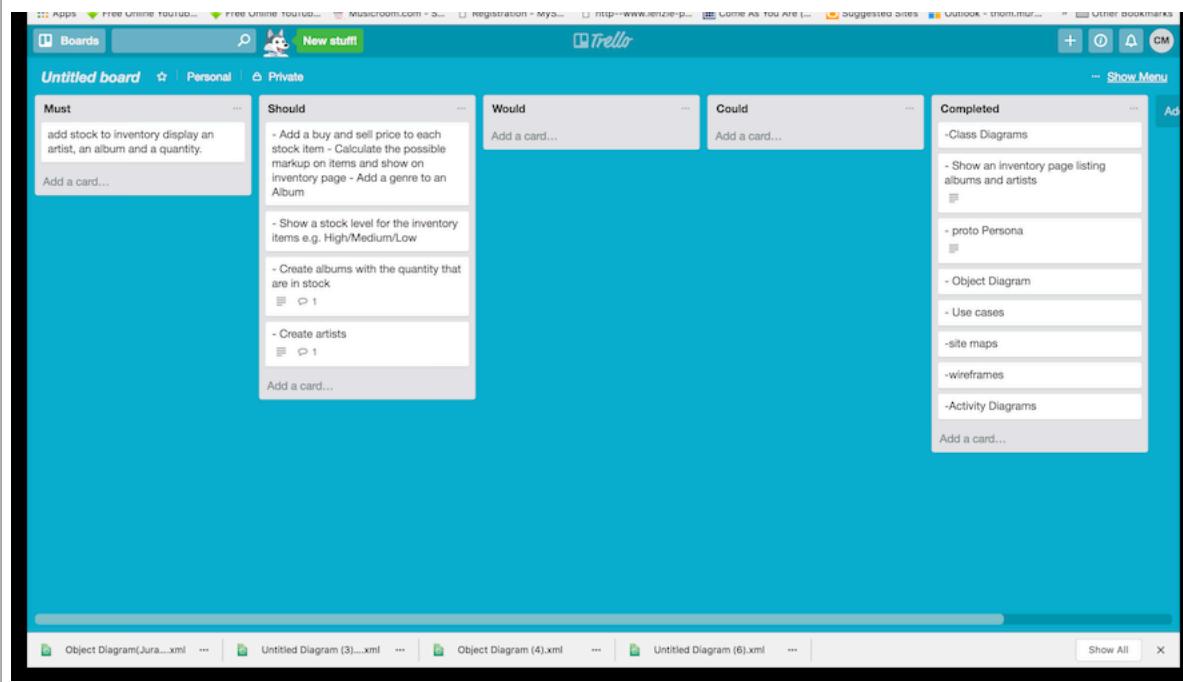
    @Test
    public void hasDeptName(){
        assertEquals("Devops", manager.getDeptName());
    }
}
```

Run ManagerTest
ManagerTest
hasName
hasDeptName
hasSalary
hasNiNumber
hasName
canRaiseSalary
canGetBonus

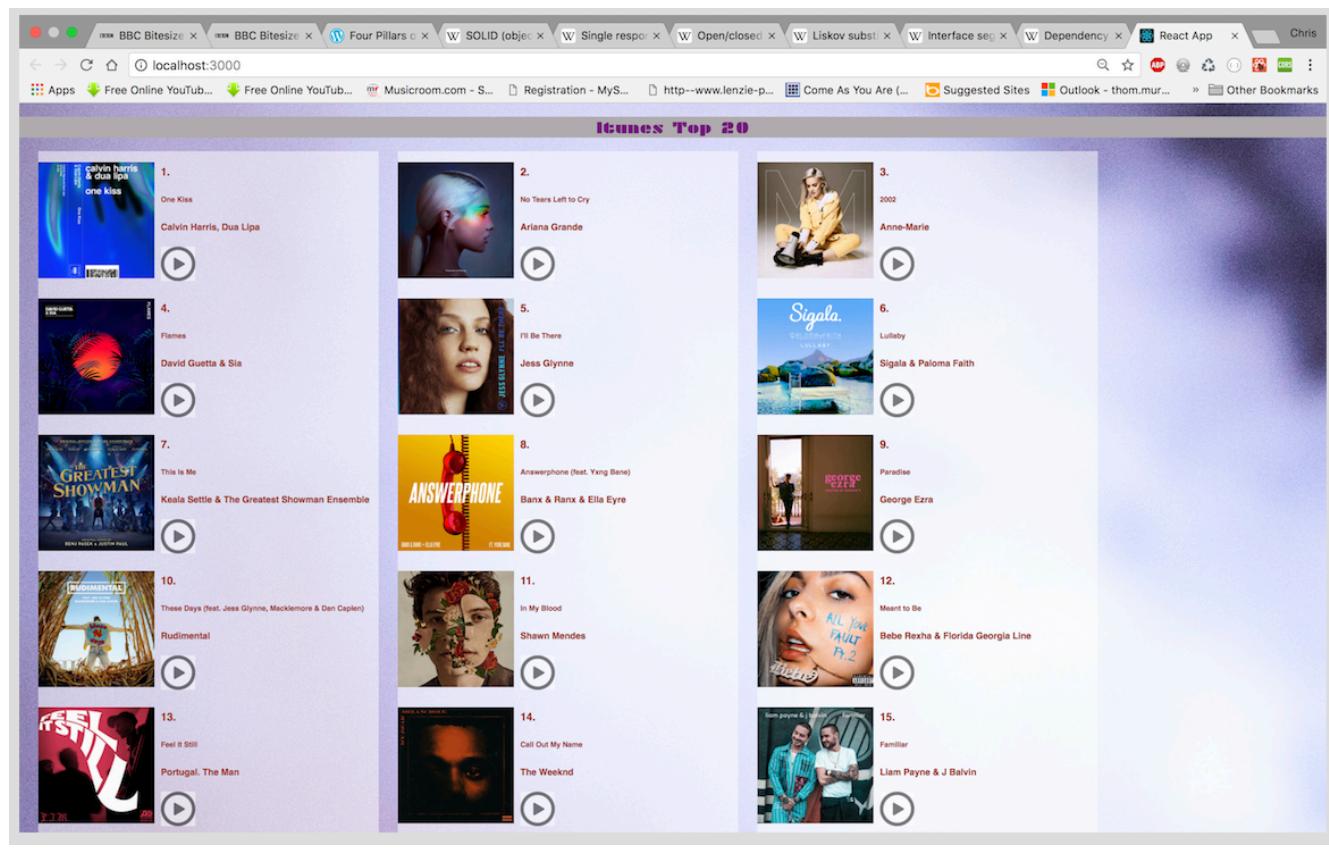
All 6 tests passed - 7ms

P	P 11	<p>Take a screenshot of one of your projects where you have worked alone and attach the Github link.</p>  <p>https://github.com/Chrismlugo/Project-1</p>	
P	P 12	<p>Take screenshots or photos of your planning and the different stages of development to show changes.</p>	

		User Needs																									
		<table border="1"> <thead> <tr> <th>As a</th> <th>I want to</th> <th>So that</th> </tr> </thead> <tbody> <tr> <td>Business owner</td> <td>Have a stock management app</td> <td>I can view, update and add stock.</td> </tr> <tr> <td>Person who isn't Too familiar with the latest tech</td> <td>Have a straightforward Layout that's familiar.</td> <td>Be able to easily understand and use the app.</td> </tr> <tr> <td>Manager of a team</td> <td>Have an app which is easy to use</td> <td>My staff can easily use it.</td> </tr> </tbody> </table>	As a	I want to	So that	Business owner	Have a stock management app	I can view, update and add stock.	Person who isn't Too familiar with the latest tech	Have a straightforward Layout that's familiar.	Be able to easily understand and use the app.	Manager of a team	Have an app which is easy to use	My staff can easily use it.													
As a	I want to	So that																									
Business owner	Have a stock management app	I can view, update and add stock.																									
Person who isn't Too familiar with the latest tech	Have a straightforward Layout that's familiar.	Be able to easily understand and use the app.																									
Manager of a team	Have an app which is easy to use	My staff can easily use it.																									
		User Journey																									
		<table border="1"> <thead> <tr> <th>Action 1</th> <th>Action 2</th> <th>Action 3</th> <th>Action 4</th> <th>Action 5</th> <th>Action 6</th> </tr> </thead> <tbody> <tr> <td>George views the home screen and would like to check the stores current inventory</td> <td>He clicks link to inventory and views current stock</td> <td>George wants to know the availability of a particular album</td> <td>The album is showing low stock availability. He would like to update the inventory with new stock</td> <td>George adds the new stock to the system and would like to confirm the update.</td> <td>George submits the form and would like to view the inventory to confirm stock has been updated</td> </tr> <tr> <th>System 1</th> <th>System 2</th> <th>System 3</th> <th>System 4</th> <th>System 5</th> <th>System 6</th> </tr> <tr> <td>Home page should have a link to navigate to the inventory page</td> <td>Searches artists database and albums database and displays all details on screen.</td> <td>Albums have a quantity. Can call a method with logic to determine high/med/low/out of stock.</td> <td>link next to each entry will navigate to an edit screen with a pre-populated form.</td> <td>Form will include a quantity bar which will be a number input.</td> <td>The submit button will update the album quantity and redirect to the inventory page.</td> </tr> </tbody> </table>	Action 1	Action 2	Action 3	Action 4	Action 5	Action 6	George views the home screen and would like to check the stores current inventory	He clicks link to inventory and views current stock	George wants to know the availability of a particular album	The album is showing low stock availability. He would like to update the inventory with new stock	George adds the new stock to the system and would like to confirm the update.	George submits the form and would like to view the inventory to confirm stock has been updated	System 1	System 2	System 3	System 4	System 5	System 6	Home page should have a link to navigate to the inventory page	Searches artists database and albums database and displays all details on screen.	Albums have a quantity. Can call a method with logic to determine high/med/low/out of stock.	link next to each entry will navigate to an edit screen with a pre-populated form.	Form will include a quantity bar which will be a number input.	The submit button will update the album quantity and redirect to the inventory page.	
Action 1	Action 2	Action 3	Action 4	Action 5	Action 6																						
George views the home screen and would like to check the stores current inventory	He clicks link to inventory and views current stock	George wants to know the availability of a particular album	The album is showing low stock availability. He would like to update the inventory with new stock	George adds the new stock to the system and would like to confirm the update.	George submits the form and would like to view the inventory to confirm stock has been updated																						
System 1	System 2	System 3	System 4	System 5	System 6																						
Home page should have a link to navigate to the inventory page	Searches artists database and albums database and displays all details on screen.	Albums have a quantity. Can call a method with logic to determine high/med/low/out of stock.	link next to each entry will navigate to an edit screen with a pre-populated form.	Form will include a quantity bar which will be a number input.	The submit button will update the album quantity and redirect to the inventory page.																						



	Unit	Ref.	Evidence	Done
Week 11	I & T		Unit, integration and acceptance testing task B	
			https://github.com/Chrismlugo/calculator_unit_and_integration_testing_mocha_selenium	
	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	
			<pre> componentDidMount(){ fetch("https://itunes.apple.com/gb/rss/topsongs/limit=20/json") .then(response => response.json()) .then(json => this.setState({songs: json.feed.entry})); } handlePlayPause(audio){ audio.paused ? audio.play() : audio.pause(); audio.classList.toggle('playing'); } render(){ return (<div> <Header/> <SongList songs={this.state.songs} handlePlayPause={this.handlePlayPause} /> </div>); } </pre>	



Unit	Ref.	Evidence	Done
P	P 1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.	

		<p>The screenshot shows the GitHub Pulse contributors page for the project 'ScotRail'. The main header indicates the period from April 22, 2018, to May 8, 2018. On the left, a sidebar menu includes options like Pulse, Contributors (which is selected), Community, Traffic, Commits, Code frequency, Dependency graph, Network, and Forks. The main content area displays a large green semi-circular chart representing total contributions to master, excluding merge commits. Below this are three detailed commit activity charts for individual contributors:</p> <ul style="list-style-type: none"> #1 AMalloch: 25 commits, 17,307 ++ 129 --. The chart shows activity peaking around April 29. #2 riakoronidi: 22 commits, 1,387 ++ 601 --. The chart shows activity peaking around May 1. #3 Chrismlugo: 14 commits, 263 ++ 71 --. The chart shows activity peaking around April 27.
P	P 2	Take a screenshot of the project brief from your group project.

Shares App

More and more people are playing the stock market. 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 minimal viable product (MVP) that uses the data to display her portfolio in useful ways so that she can make better decisions.

MVP

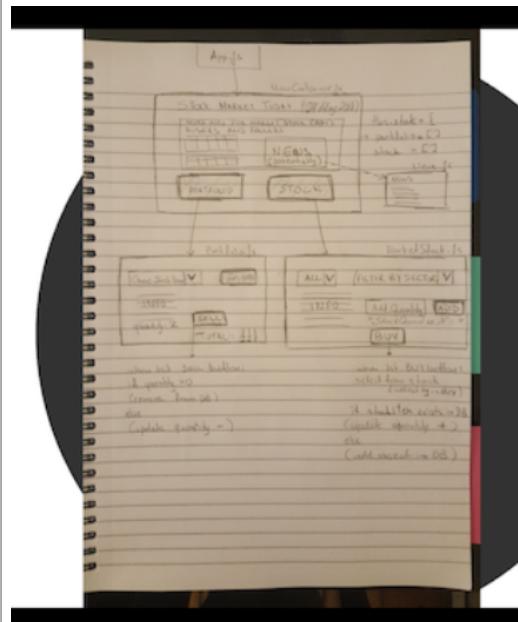
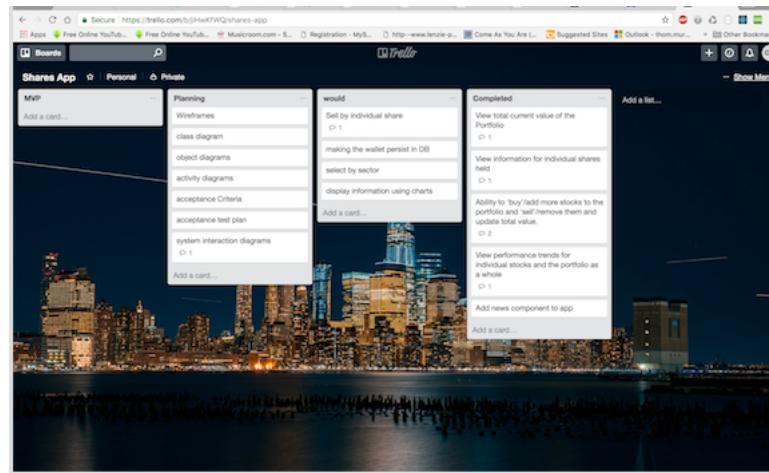
- View total current value of the portfolio
- View information for individual shares held
- Ability to 'buy'/add more stocks to the portfolio and 'sell'/remove them and update total value.
- View performance trends for individual stocks and the portfolio as a whole

Stock api's can be found here - <https://github.com/toddmotto/public-apis#finance>

If you want to make your own api an object may look something like this:

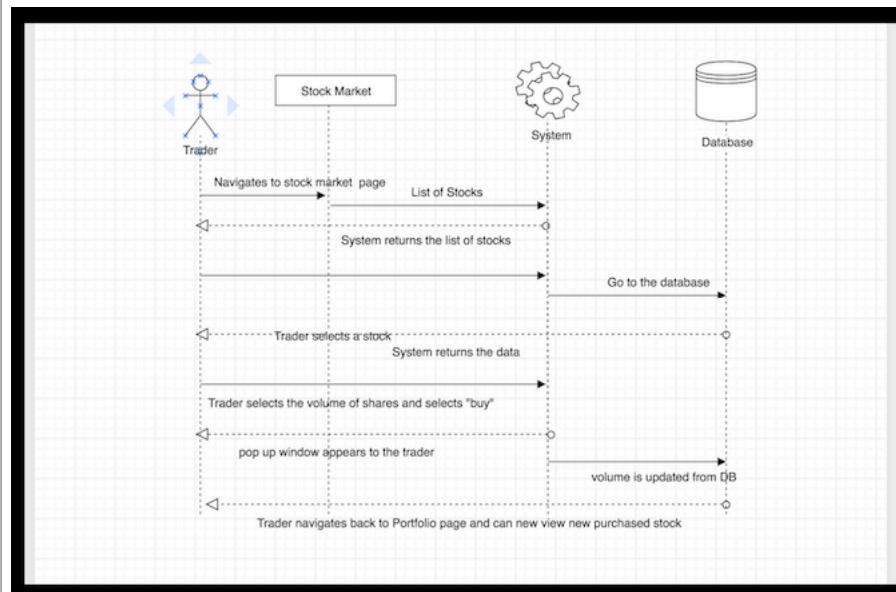
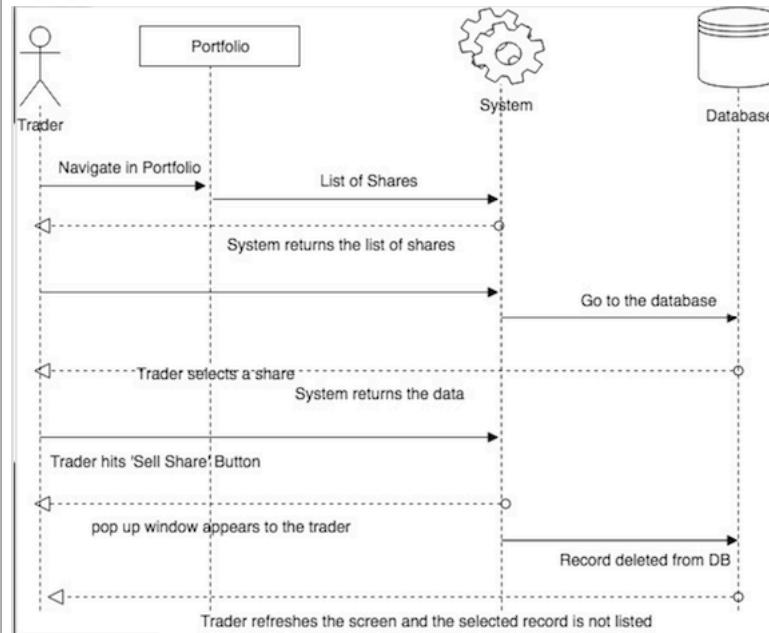
```
{  
    "name": "Fusionex",  
    "epic": "FXI",  
    "price": 120.00,  
    "quantity": 2000,  
    "buyPrice": 80.00,  
    "pastCloseOfDayPrices": [92.00, 89.00, 103.00, 125.00, 108.00, 98.00, 110.00],  
    "buyDate": "2014-11-15"  
}
```

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

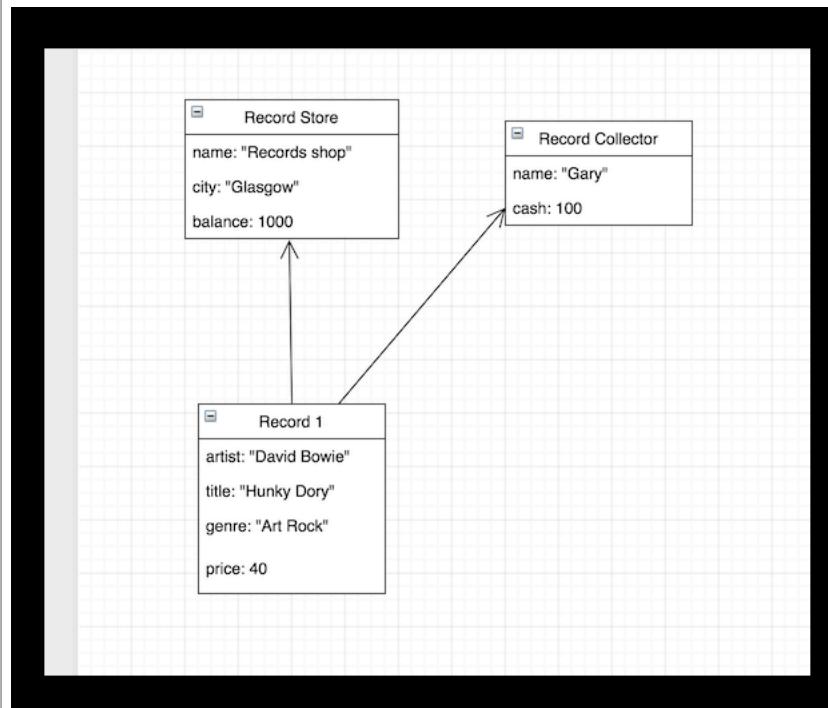
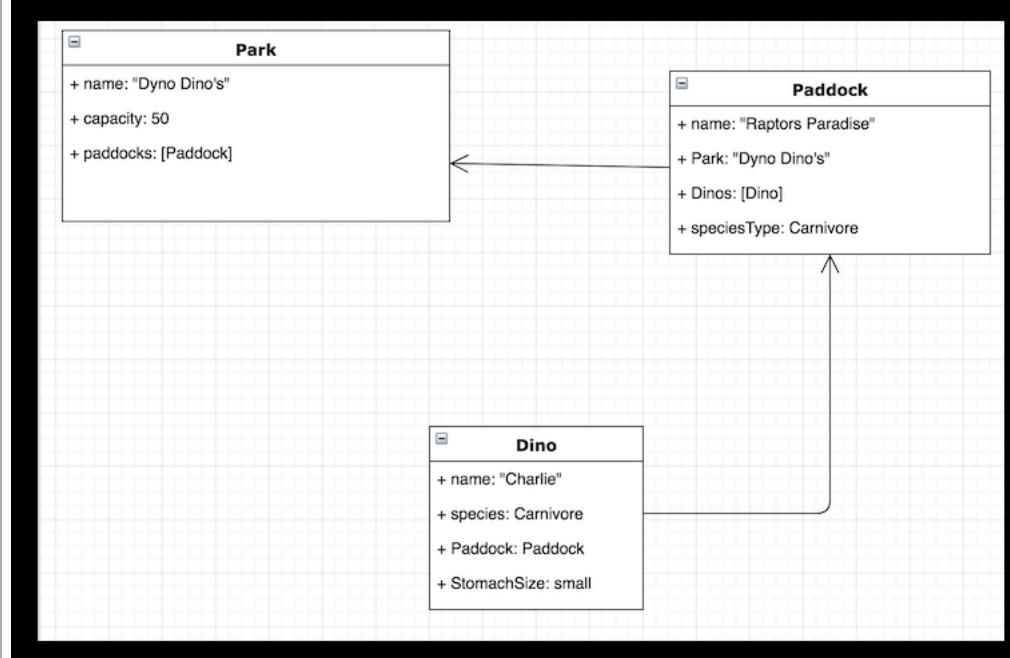


P	P 4	Write an acceptance criteria and test plan.	
Acceptance Criteria			
Acceptance Criteria	Expected Result/Output	Pass/Fail	
User is able to view the total value their portfolio	When link to portfolio is clicked, user is navigated to new page which displays total value of the portfolio on the page	Pass	
User is able to view an individual share that they own	On the Portfolio page, the user will select from a dropdown the desired share they want to view details of that share on screen	Pass	
User can view the performance of their portfolio against the current market	When an individual share is selected, the page will display a comparison between the bought share and the current market price, showing a profit/loss value	Pass	
User can buy an amount of shares from the stock market	On the stock market page, the user can select the chosen company, Select a quantity they'd like to buy and hit submit which adds to user's portfolio	Pass	
P	P 7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	

**Week
13**



P P 8 Produce two object diagrams.



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.
---	-----	---

```
RecordCollector.prototype.genreValue = function (genre) {
  let filteredRecords = _.filter(this.collection, {genre: genre});
  return _.sumBy(filteredRecords, 'price');
};
```

I'd chosen to use this algorithm to calculate the total value of a record collectors collection of a particular genre.

Using Lodashes filter method the algorithm filters out the collection array and creates a new filtered array of records of the same genre.

The algorithm then returns the sum of all the filtered records prices using Lodashes sumBy method.

```
public void feedDinos(){
    for(Dino dino: this.dinosaurs){
        Random ran = new Random();
        int foodAmount = ran.nextInt((5) + 1);
        if(this.foodStock >= foodAmount) {
            dino.feed(foodAmount);
            this.foodStock -= foodAmount;
            DBHelper.saveOrUpdate(dino);
        }
    }
}
```

This algorithm was chosen to feed dinosaurs contained in a particular paddocks dinosaurs array.

The algorithm loops through the paddocks array of dinosaurs and then generates a random integer value between 1 and 5 which is set to the variable foodAmount.

It then checks if the foodAmount value is greater than the paddocks foodStock value, which is set at 0 initially, and if true the dinosaur will feed from the paddocks foodStock. The foodStock value will then be subtracted from the variable foodAmount, and then update the dinosaurs stomach value in the database.

P	P 17	Produce a bug tracking report	
Bug Tracking			
		Deleting all portfolio shares should return an empty drop down list but instead the screen would go blank	Added an if statement to check if array is empty Passed
		Values returned from buying/selling should be return upto 2 decimal places	Adds code to round results to 2 decimals Passed
		Bootstrap table data should be accessed from props.	Data passed through render in route Passed
		Balance should persist throughout application after buy or sell	Failed
		Must re-render page after share has been sold	Failed