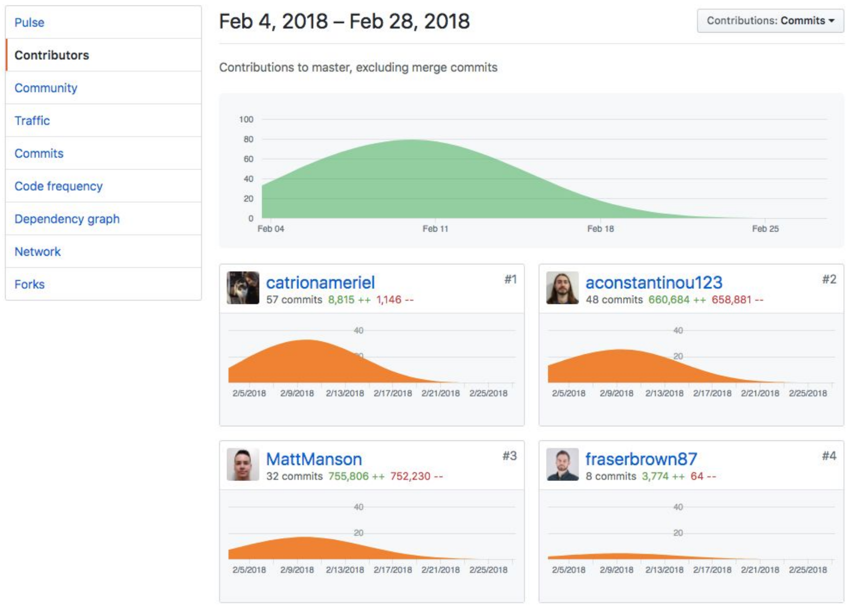
Evidence for Project Unit

Catriona Meriel

E17

P. 1 - Github Contributors page



Feb 4, 2018 – Feb 28, 2018

Contributions to master, excluding merge commits

catrionameriel #1
57 commits 8,815 ++ 1,146 --

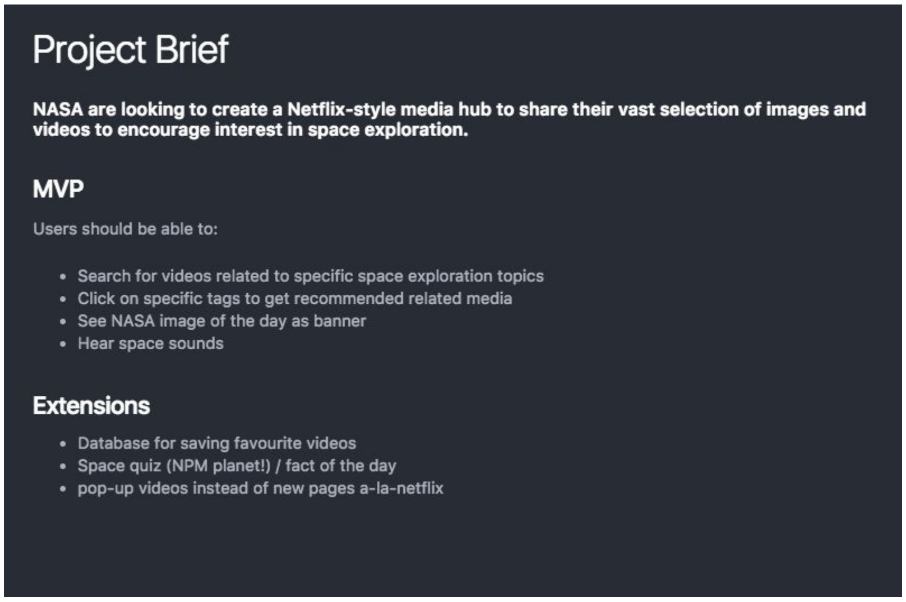
aconstantinou123 #2
48 commits 660,684 ++ 658,881 --

MattManson #3
32 commits 755,806 ++ 752,230 --

fraserbrown87 #4
8 commits 3,774 ++ 64 --

Contributors page for SpaceFlix group project

P. 2 - Project Brief



Project Brief

NASA are looking to create a Netflix-style media hub to share their vast selection of images and videos to encourage interest in space exploration.

MVP

Users should be able to:

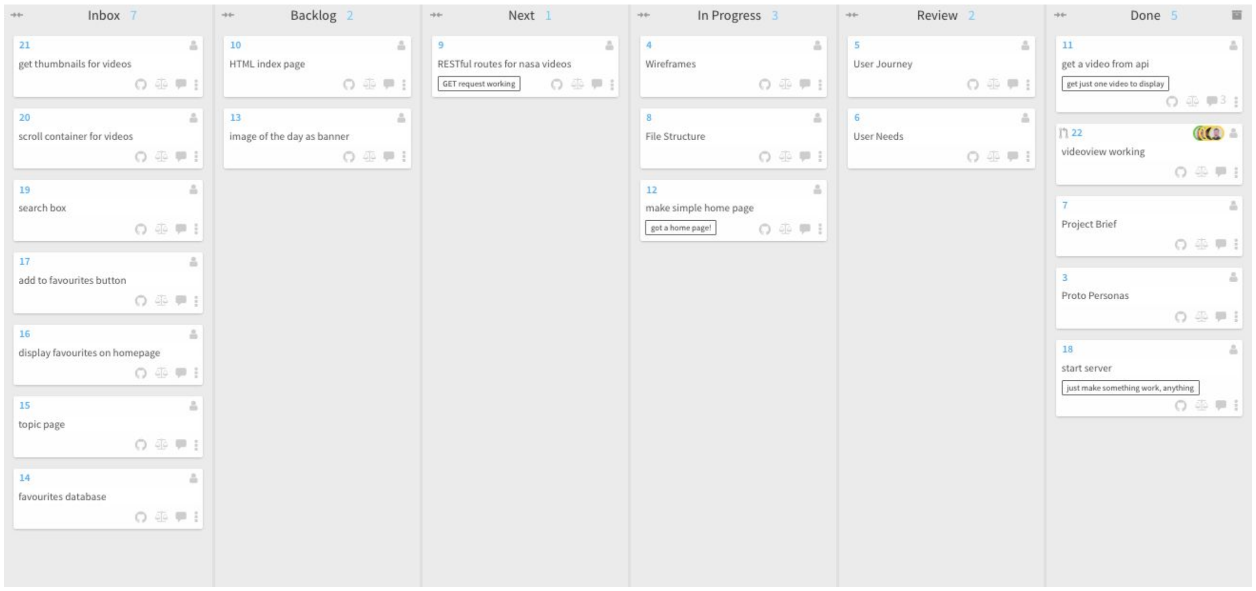
- Search for videos related to specific space exploration topics
- Click on specific tags to get recommended related media
- See NASA image of the day as banner
- Hear space sounds

Extensions

- Database for saving favourite videos
- Space quiz (NPM planet!) / fact of the day
- pop-up videos instead of new pages a-la-netflix

Project brief for SpaceFlix - our group project

P. 3 - Use of Waffle.io



Inbox 7
- 21 get thumbnails for videos
- 20 scroll container for videos
- 19 search box
- 17 add to favourites button
- 16 display favourites on homepage
- 15 topic page
- 14 favourites database

Backlog 2
- 10 HTML index page
- 13 image of the day as banner

Next 1
- 9 RESTful routes for nasa videos — GET request working

In Progress 3
- 4 Wireframes
- 8 File Structure
- 12 make simple home page — got a home page!

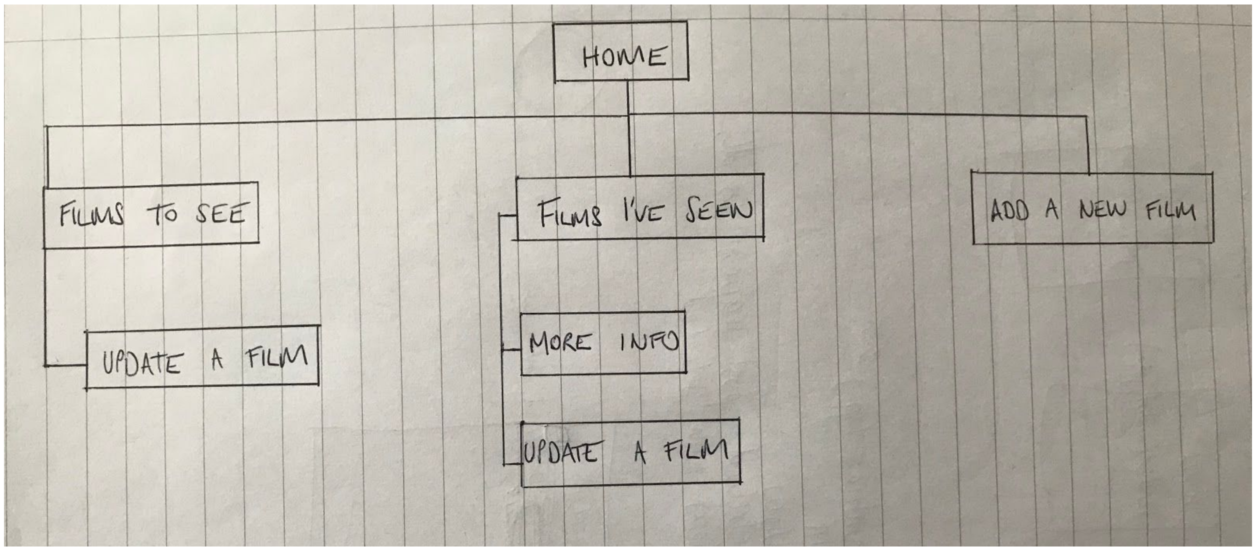
Review 2
- 5 User Journey
- 6 User Needs

Done 5
- 11 get a video from api — get just one video to display — 3
- 22 videoview working
- 7 Project Brief
- 3 Proto Personas
- 18 start server — just make something work, anything

P. 4 - Acceptance Criteria

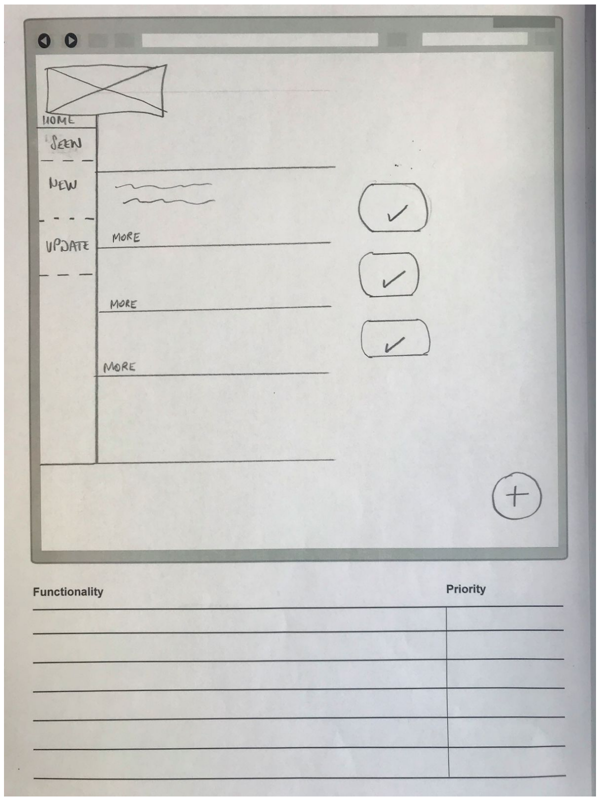
Acceptance criteria	Expected result/output	Pass/Fail
A user can search for media	A user can put a search term into the search box and the results will be displayed on the page	Pass
A user can save their favourite videos	A user can press add to favourites on a particular video page and it will be added to favourites.	Pass
A user can access their favourite videos	A user can press the button at the bottom of the home page and it takes them to their favourites page	Pass

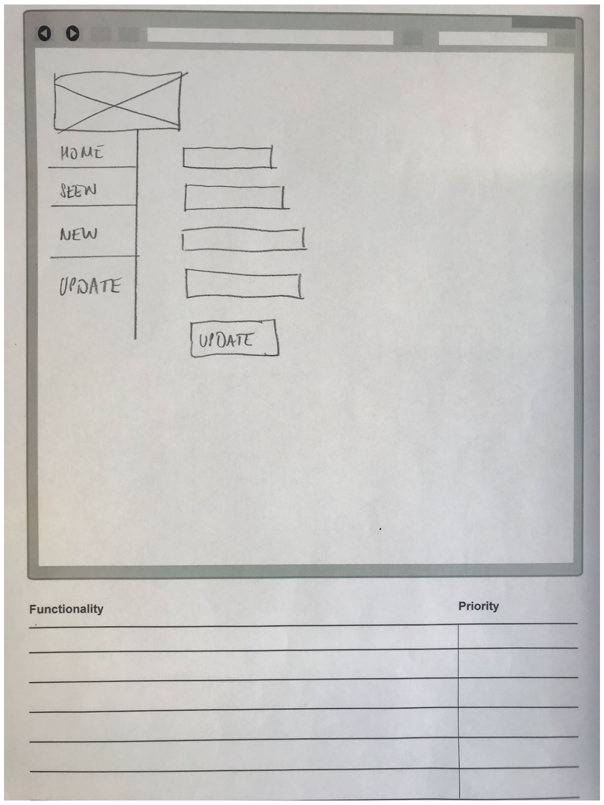
P. 5 - User sitemap



User sitemap for SeeSaw

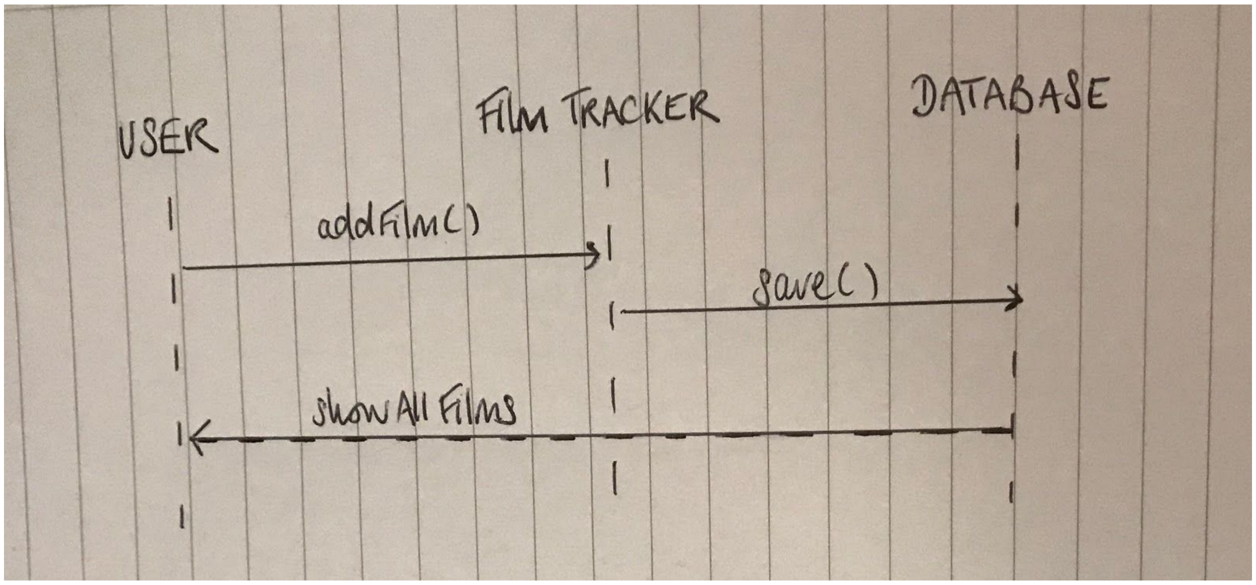
P. 6 - Wireframes designs



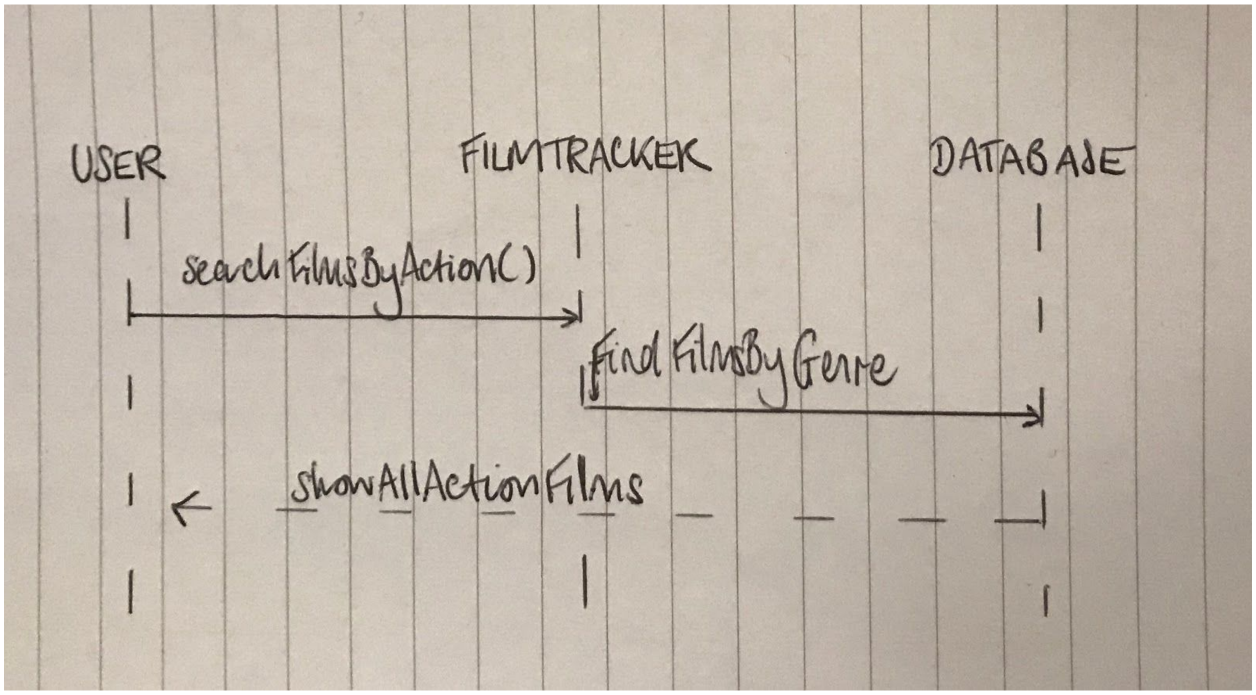


Home page and update form wireframes

P. 7 - System interactions diagrams

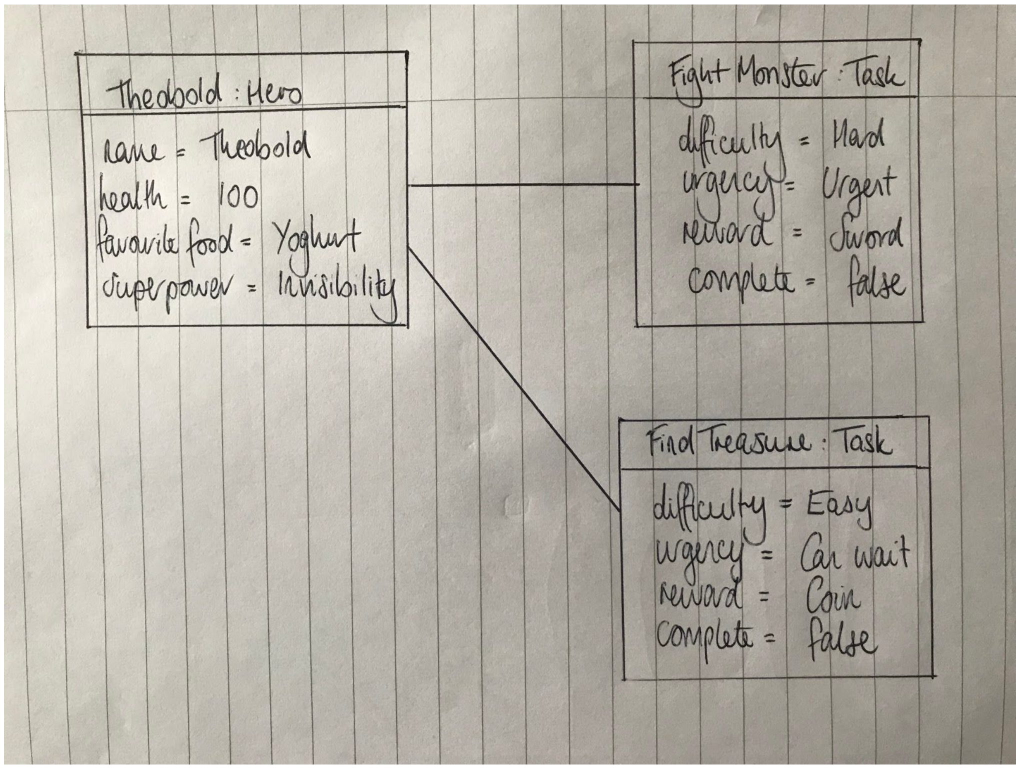


Adding a new film to the app

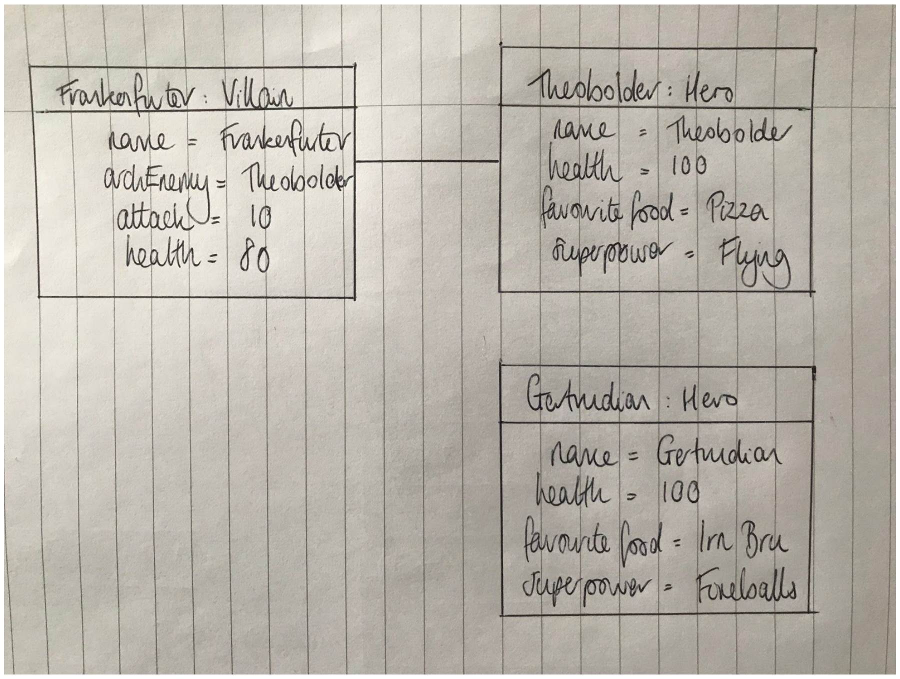


Searching for films by action genre

P. 8 - Two Object Diagrams

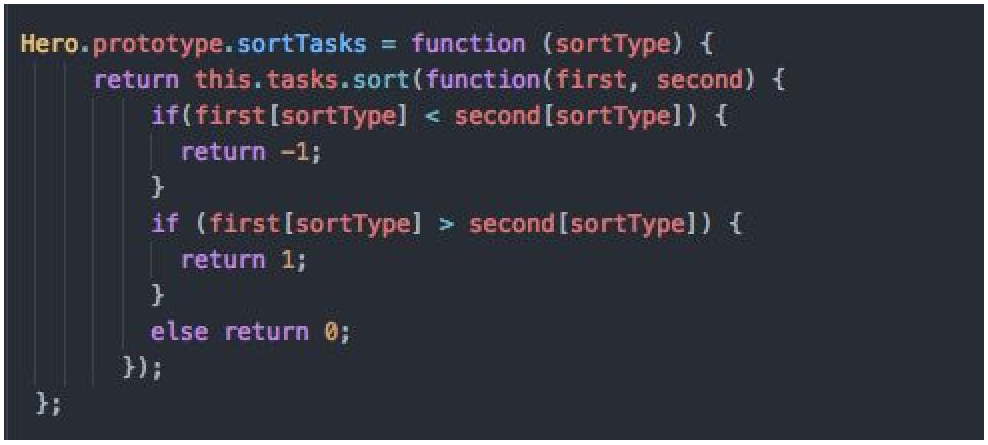


Object diagram showing instances of the hero and task classes and their relationships



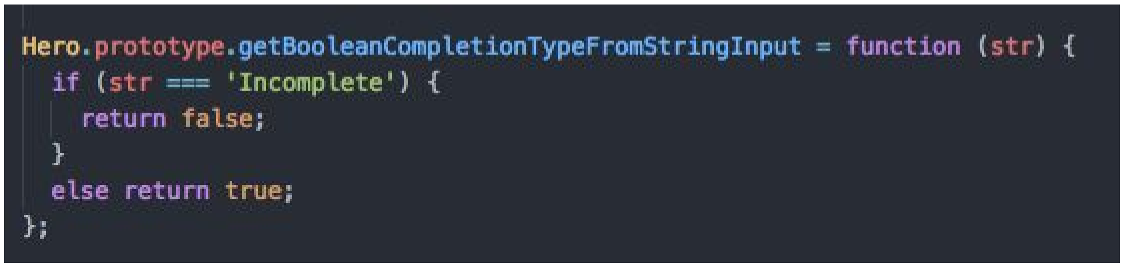
Object diagram showing instances of villain and hero

P. 9 - Choice of two algorithms

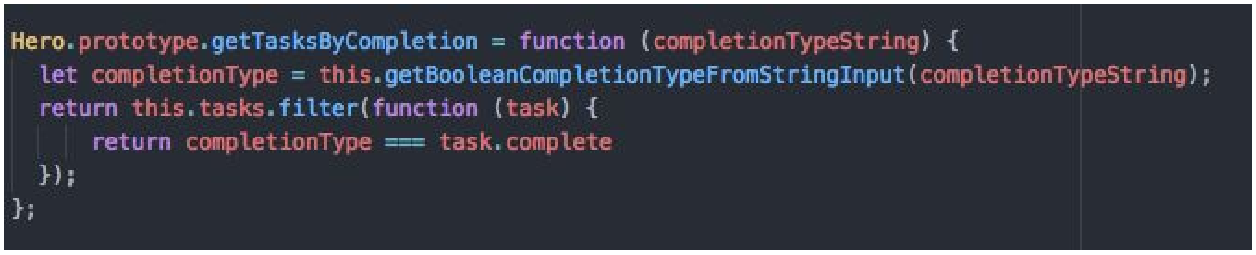


```
Hero.prototype.sortTasks = function (sortType) {
    return this.tasks.sort(function(first, second) {
        if(first[sortType] < second[sortType]) {
          return -1;
        }
        if (first[sortType] > second[sortType]) {
          return 1;
        }
        else return 0;
      });
 };
```

I decided to use a sort method on my array of tasks as it is the most efficient and shortest way to achieve a sorted array. You can also pass in what you want to sort by, which makes the algorithm polymorphic.



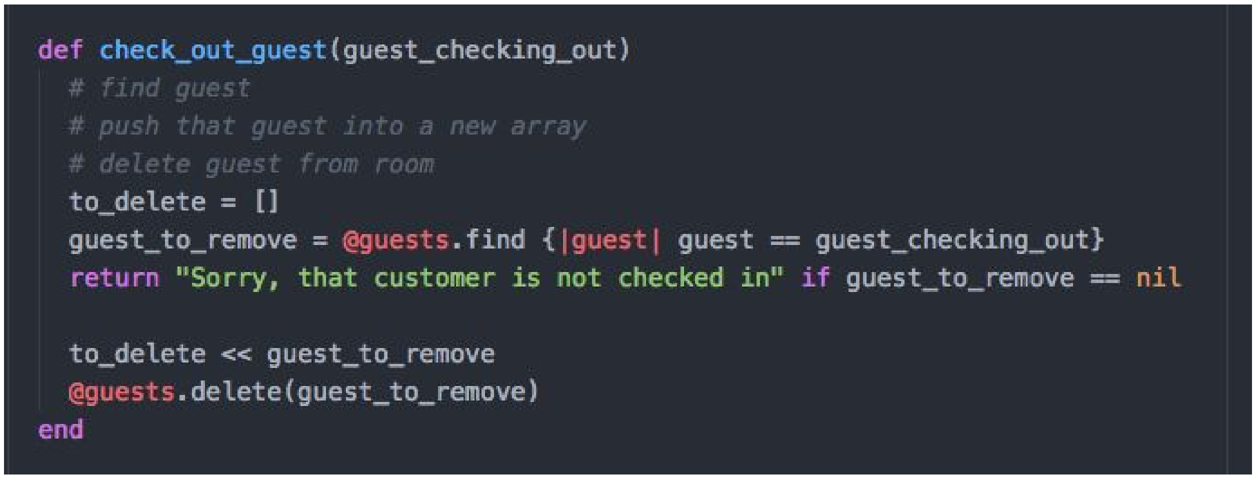
```
Hero.prototype.getBooleanCompletionTypeFromStringInput = function (str) {
  if (str === 'Incomplete') {
    return false;
  }
  else return true;
};
```



```
Hero.prototype.getTasksByCompletion = function (completionTypeString) {
  let completionType = this.getBooleanCompletionTypeFromStringInput(completionTypeString);
  return this.tasks.filter(function (task) {
      return completionType === task.complete
  });
};
```

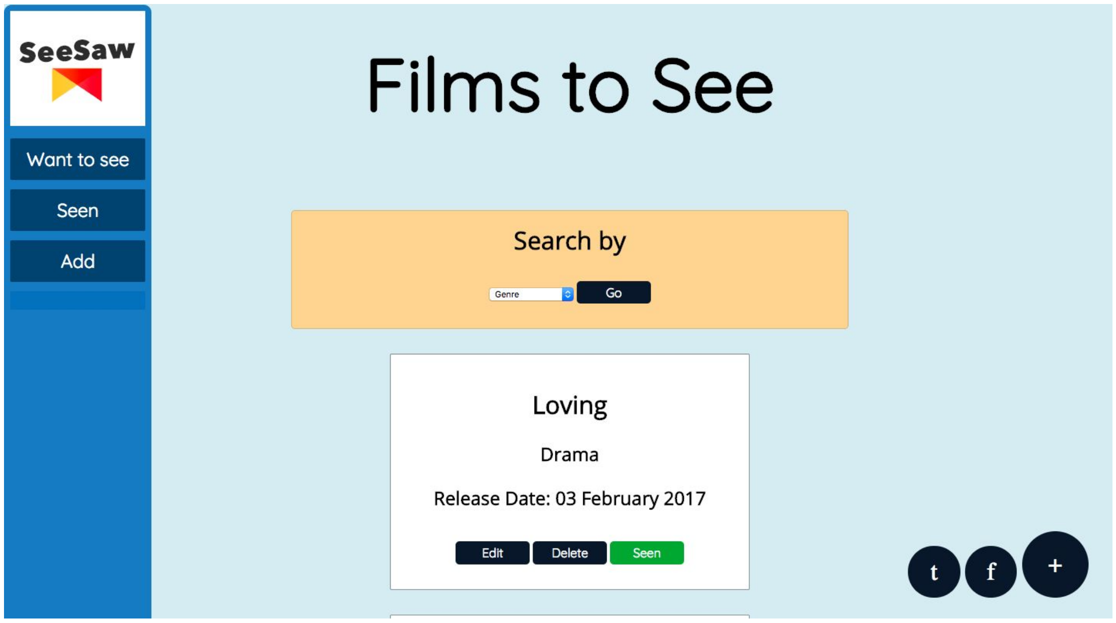
I split this algorithm into two smaller and neater functions. I wanted to be able to return the tasks by completion level so I first needed to check whether the user wanted completed tasks or incomplete tasks. This function returned a boolean which was then passed into a filter method. This method looped through each task in the array and returned a new array of the task which matched what you were searching for. I used this algorithm as it gave me the result I wanted in only a few lines of code.

P. 10 Example of Pseudocode



```
def check_out_guest(guest_checking_out)
  # find guest
  # push that guest into a new array
  # delete guest from room
  to_delete = []
  guest_to_remove = @guests.find {|guest| guest == guest_checking_out}
  return "Sorry, that customer is not checked in" if guest_to_remove == nil

  to_delete << guest_to_remove
  @guests.delete(guest_to_remove)
end
```

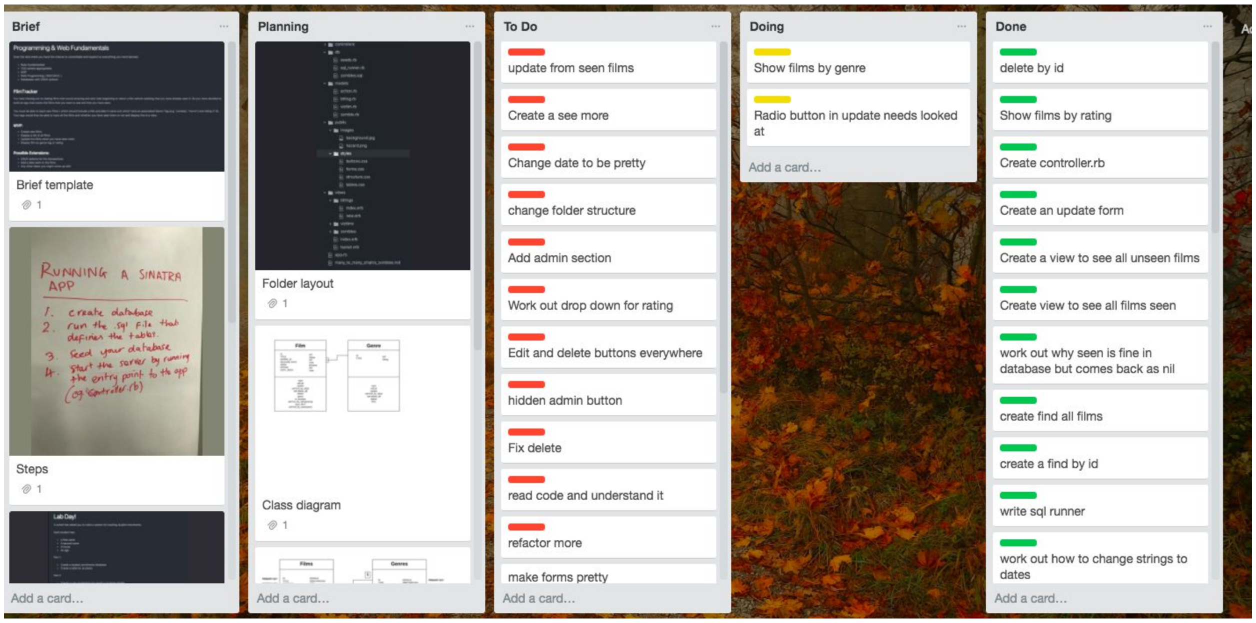
P. 11 Github link to one of your projects



Front page of Ruby project

Github link: https://github.com/catrionameriel/Film_Tracker

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



Brief
- Brief template
- RUNNING A SINATRA APP 1. create database 2. run the sql file that defines the tables. 3. Seed your database 4. Start the server by running the entry point to the app (eg 'controller.rb') — Steps
- Lab Day!
- Add a card...

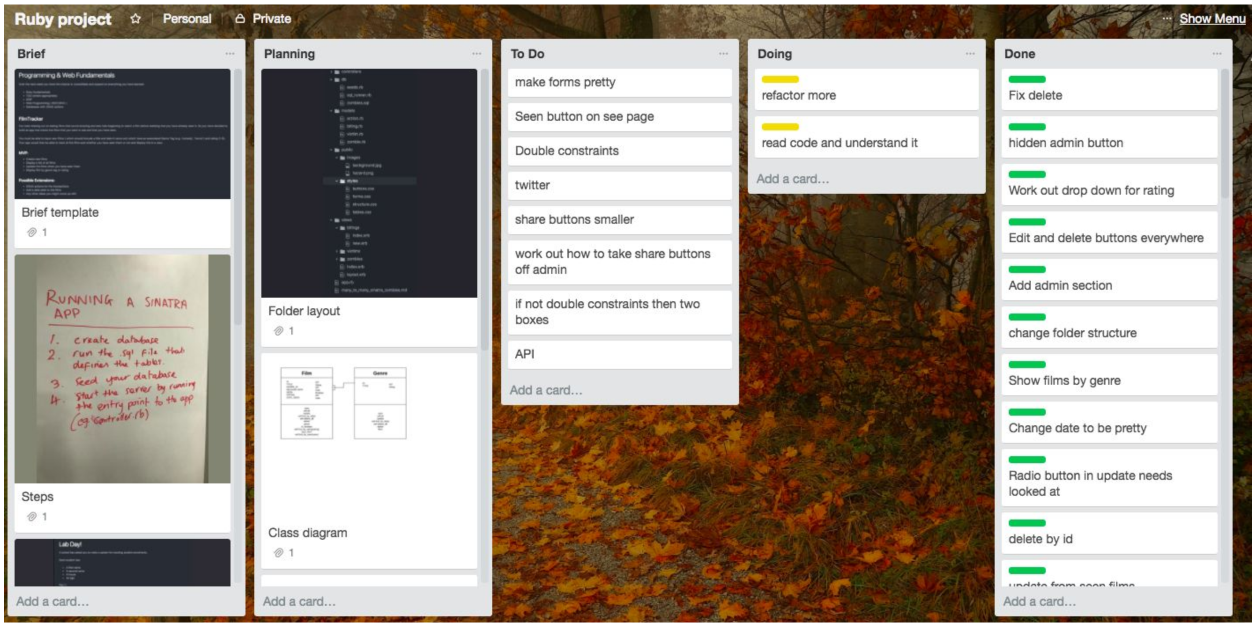
Planning
- Folder layout
- Class diagram
- Add a card...

To Do
- update from seen films
- Create a see more
- Change date to be pretty
- change folder structure
- Add admin section
- Work out drop down for rating
- Edit and delete buttons everywhere
- hidden admin button
- Fix delete
- read code and understand it
- refactor more
- make forms pretty
- Add a card...

Doing
- Show films by genre
- Radio button in update needs looked at
- Add a card...

Done
- delete by id
- Show films by rating
- Create controller.rb
- Create an update form
- Create a view to see all unseen films
- Create view to see all films seen
- work out why seen is fine in database but comes back as nil
- create find all films
- create a find by id
- write sql runner
- work out how to change strings to dates
- Add a card...

Trello board mid-way project



Ruby project Personal Private Show Menu

Brief
- Brief template
- RUNNING A SINATRA APP 1. create database 2. run the sql file that defines the tables. 3. Seed your database 4. Start the server by running the entry point to the app (eg 'controller.rb') — Steps
- Lab Day!
- Add a card...

Planning
- Folder layout
- Class diagram
- Add a card...

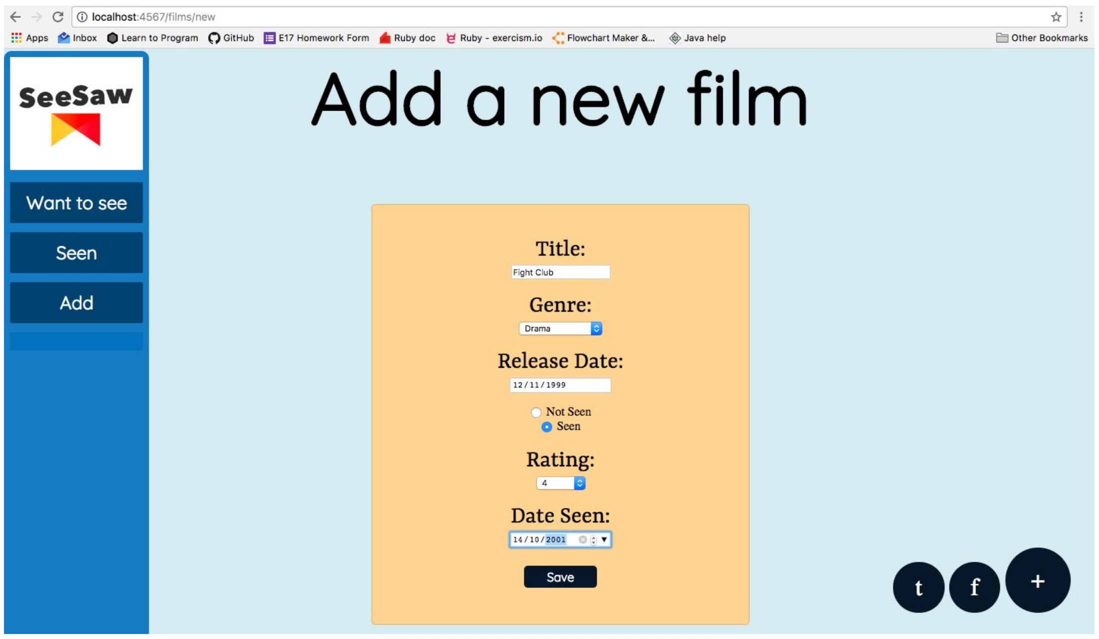
To Do
- make forms pretty
- Seen button on see page
- Double constraints
- twitter
- share buttons smaller
- work out how to take share buttons off admin
- if not double constraints then two boxes
- API
- Add a card...

Doing
- refactor more
- read code and understand it
- Add a card...

Done
- Fix delete
- hidden admin button
- Work out drop down for rating
- Edit and delete buttons everywhere
- Add admin section
- change folder structure
- Show films by genre
- Change date to be pretty
- Radio button in update needs looked at
- delete by id
- update from seen films
- Add a card...

End Trello board for SeeSaw project

P. 13 User input & P. 14 Interaction with data persistence



Add a new film

Title: Fight Club

Genre: Drama

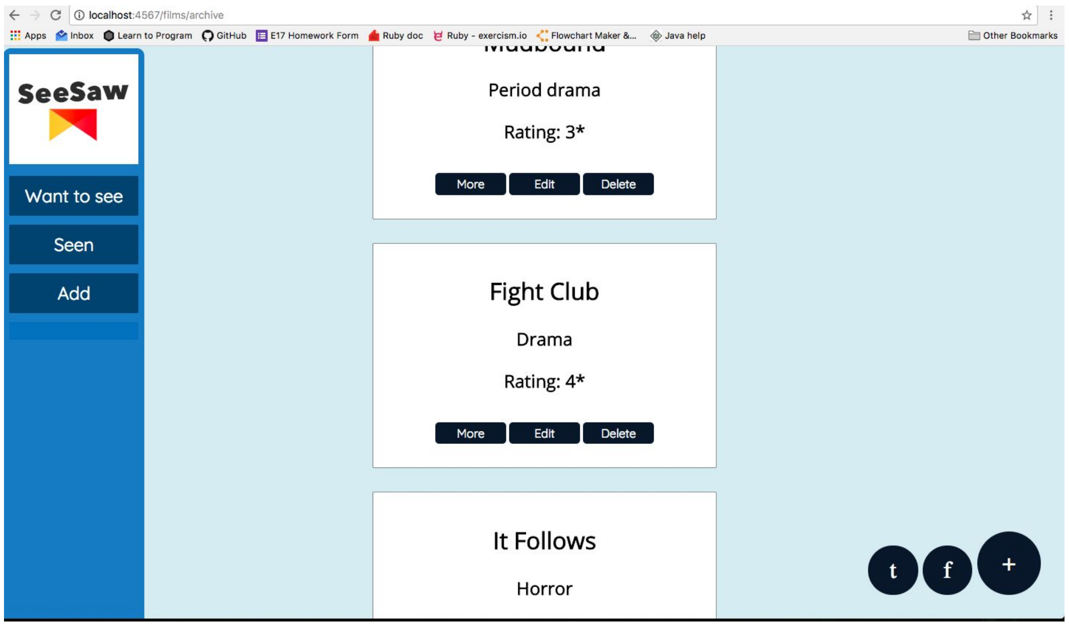
Release Date: 12/11/1999

Not Seen [✓ Seen]

Rating: 4

Date Seen: 14/10/2001

Showing where the user adds a new film (in this case Fight Club) and its details to the database/ app



Period drama

Rating: 3*

Fight Club

Drama

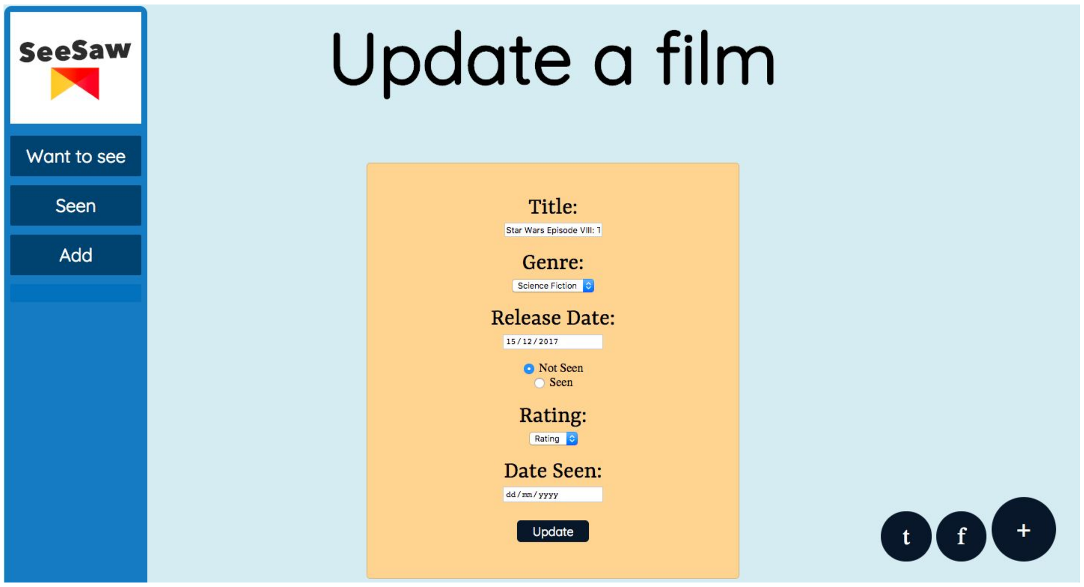
Rating: 4*

It Follows

Horror

Showing that film has now been saved in the database/ app

P. 15 - User output result



Update a film

Title:
Star Wars Episode VIII: T

Genre:
Science Fiction

Release Date:
15/12/2017

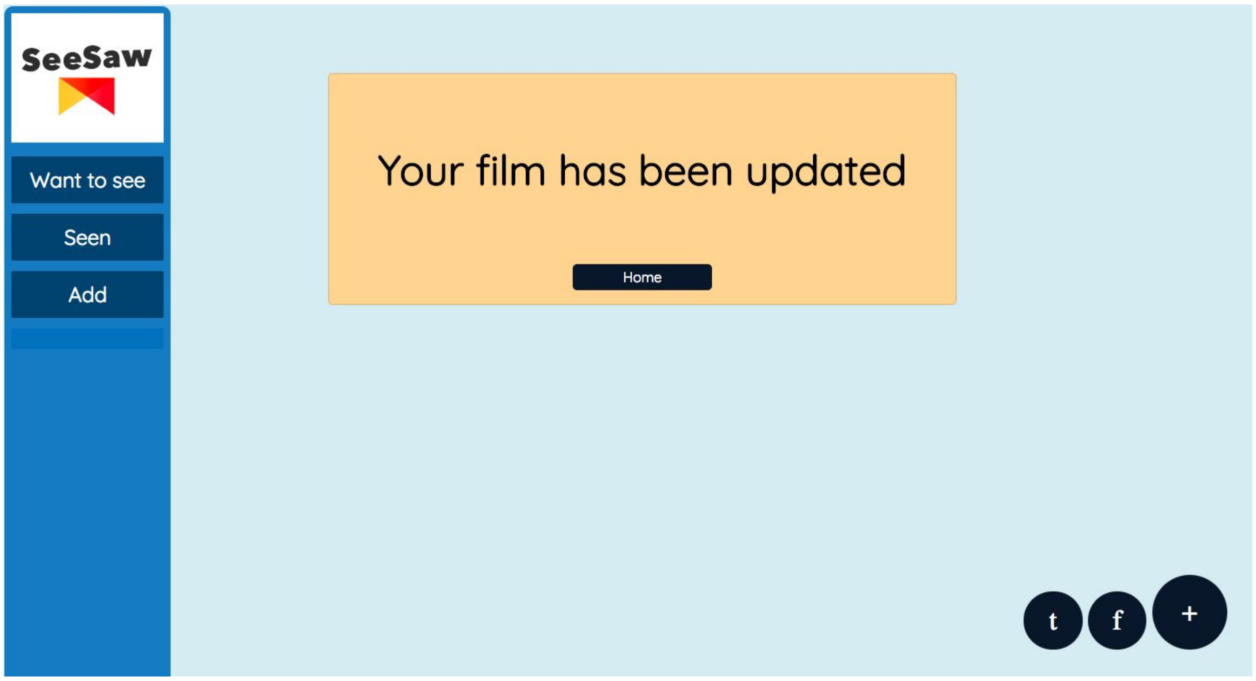
● Not Seen
○ Seen

Rating:
Rating

Date Seen:
dd/mm/yyyy

Update

The update film page where you can change the details of the film

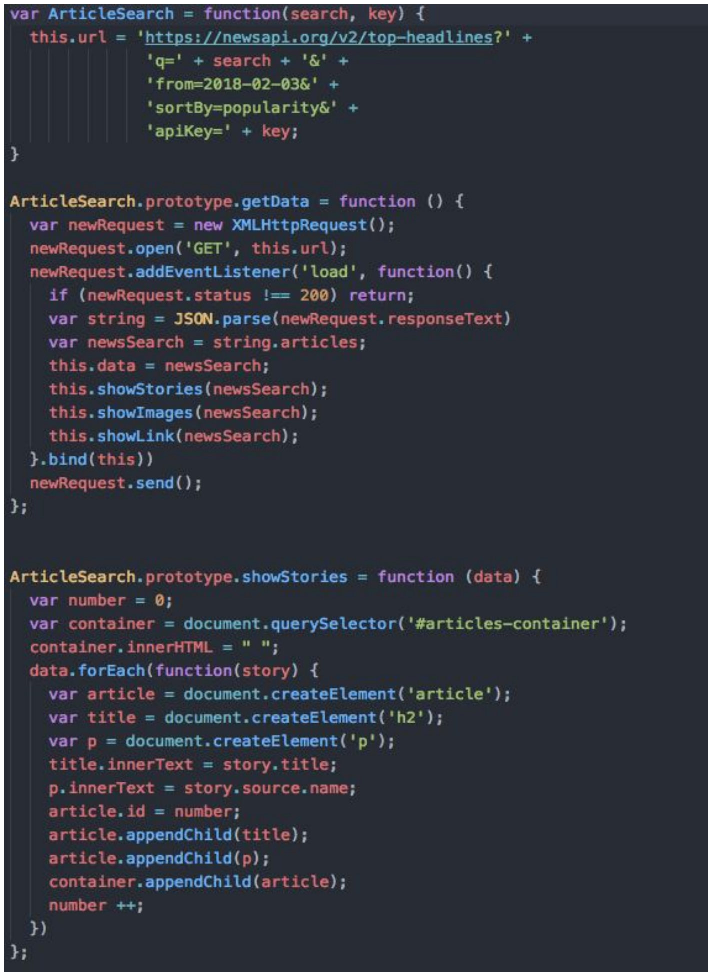


Your film has been updated

Home

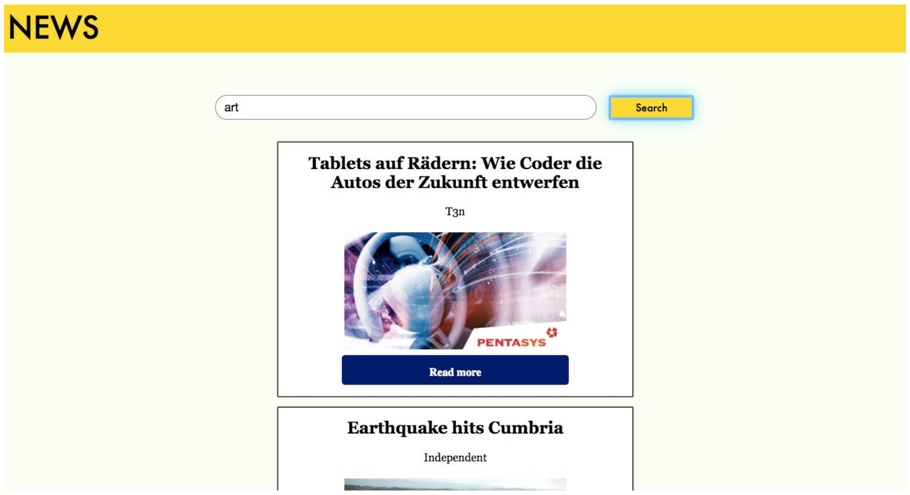
The page shown when film is updated

P. 16 Implementing APIs



```javascript
var ArticleSearch = function(search, key) {
  this.url = 'https://newsapi.org/v2/top-headlines?' +
            'q=' + search + '&' +
            'from=2018-02-03&' +
            'sortBy=popularity&' +
            'apiKey=' + key;
}

ArticleSearch.prototype.getData = function () {
  var newRequest = new XMLHttpRequest();
  newRequest.open('GET', this.url);
  newRequest.addEventListener('load', function() {
    if (newRequest.status !== 200) return;
    var string = JSON.parse(newRequest.responseText)
    var newsSearch = string.articles;
    this.data = newsSearch;
    this.showStories(newsSearch);
    this.showImages(newsSearch);
    this.showLink(newsSearch);
  }.bind(this))
  newRequest.send();
};


ArticleSearch.prototype.showStories = function (data) {
  var number = 0;
  var container = document.querySelector('#articles-container');
  container.innerHTML = " ";
  data.forEach(function(story) {
    var article = document.createElement('article');
    var title = document.createElement('h2');
    var p = document.createElement('p');
    title.innerText = story.title;
    p.innerText = story.source.name;
    article.id = number;
    article.appendChild(title);
    article.appendChild(p);
    container.appendChild(article);
    number ++;
  })
};
```

Code that requests data from API and displays it on page



NEWS

art

Tablets auf Rädern: Wie Coder die Autos der Zukunft entwerfen

T3n

Earthquake hits Cumbria

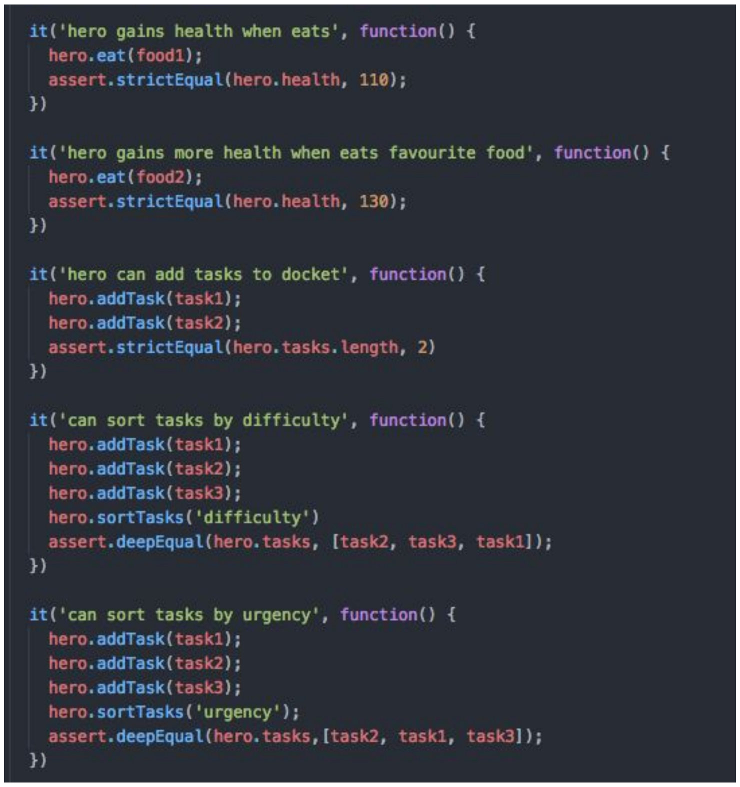
Independent

The result of the news API being used whilst running

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

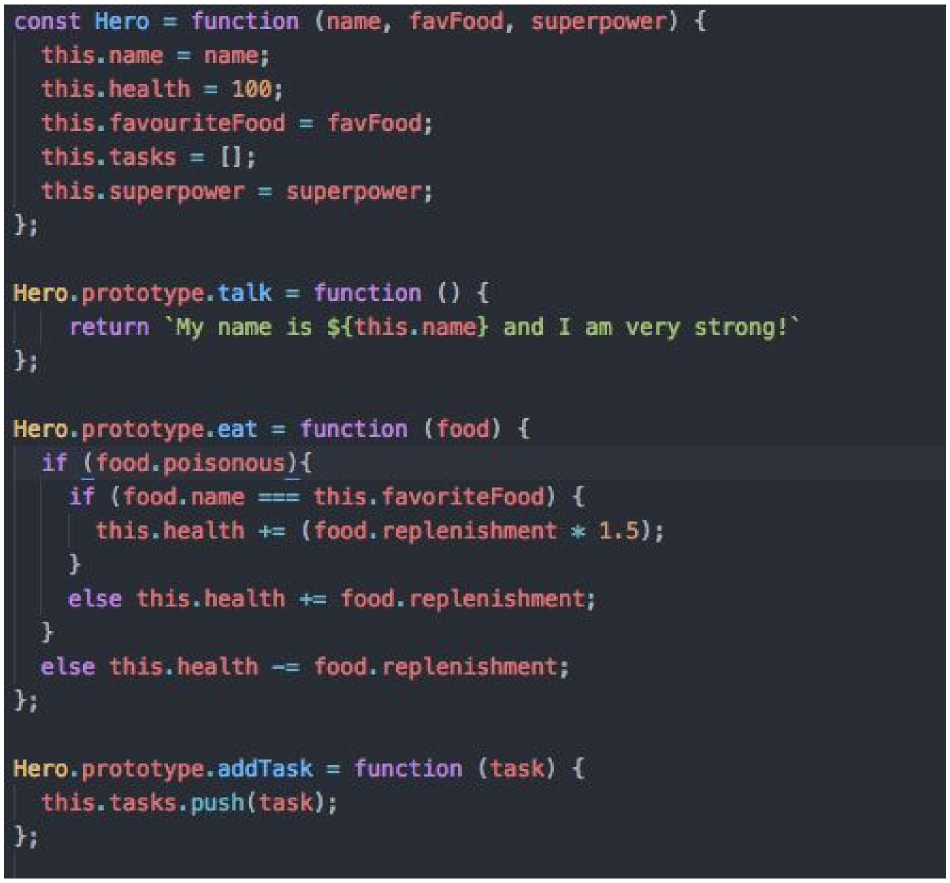
Only a certain number of videos and images are displayed on search	Failed	Added a limit to request so that only up to a certain number are returned	Passed
A user must be able to press the enter key on the search box to search	Failed	Found the enter key code from the event so that could trigger a search as well	Passed
A user must be able to hear a space sound	Failed	API was limited so had to create a random number generator that picks a random sound from the API	Passed
The design must be responsive	Failed	Used flexbox and rem to make design be able to be used on different screens	Passed
User must be able to store video in favourites	Failed	Created a database to be able to have data persistence	Passed

P. 18 Testing your program



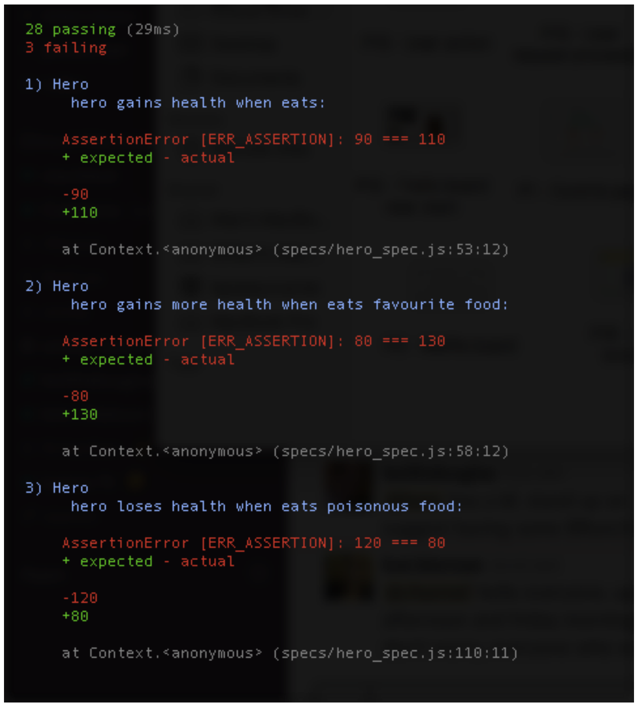
```
it('hero gains health when eats', function() {
  hero.eat(food1);
  assert.strictEqual(hero.health, 110);
})

it('hero gains more health when eats favourite food', function() {
  hero.eat(food2);
  assert.strictEqual(hero.health, 130);
})

it('hero can add tasks to docket', function() {
  hero.addTask(task1);
  hero.addTask(task2);
  assert.strictEqual(hero.tasks.length, 2)
})

it('can sort tasks by difficulty', function() {
  hero.addTask(task1);
  hero.addTask(task2);
  hero.addTask(task3);
  hero.sortTasks('difficulty')
  assert.deepEqual(hero.tasks, [task2, task3, task1]);
})

it('can sort tasks by urgency', function() {
  hero.addTask(task1);
  hero.addTask(task2);
  hero.addTask(task3);
  hero.sortTasks('urgency');
  assert.deepEqual(hero.tasks,[task2, task1, task3]);
})
```

Tests for hero model



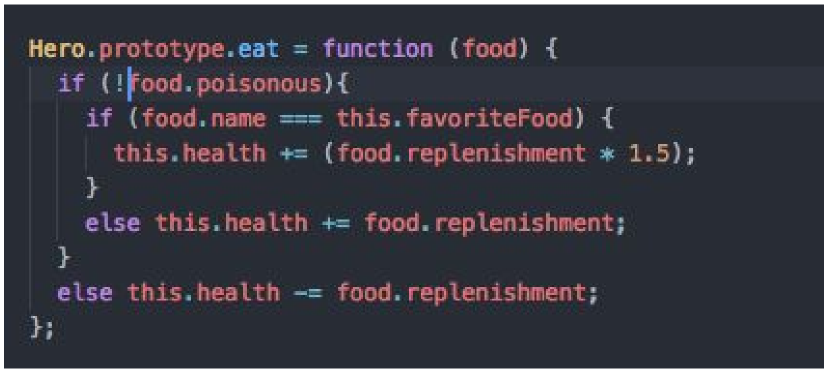
```javascript
const Hero = function (name, favFood, superpower) {
  this.name = name;
  this.health = 100;
  this.favouriteFood = favFood;
  this.tasks = [];
  this.superpower = superpower;
};

Hero.prototype.talk = function () {
    return `My name is ${this.name} and I am very strong!`
};

Hero.prototype.eat = function (food) {
  if (food.poisonous){
    if (food.name === this.favoriteFood) {
      this.health += (food.replenishment * 1.5);
    }
    else this.health += food.replenishment;
  }
  else this.health -= food.replenishment;
};

Hero.prototype.addTask = function (task) {
  this.tasks.push(task);
};
```

The eat function not working



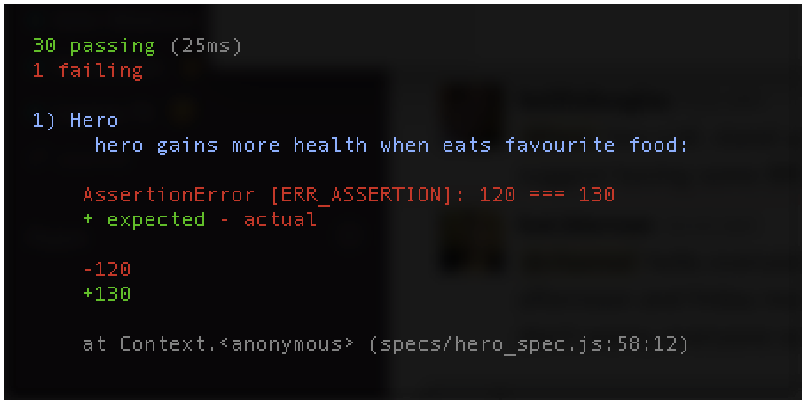
```
28 passing (29ms)
3 failing

1) Hero
     hero gains health when eats:

    AssertionError [ERR_ASSERTION]: 90 === 110
    + expected - actual

    -90
    +110

    at Context.<anonymous> (specs/hero_spec.js:53:12)

2) Hero
     hero gains more health when eats favourite food:

    AssertionError [ERR_ASSERTION]: 80 === 130
    + expected - actual

    -80
    +130

    at Context.<anonymous> (specs/hero_spec.js:58:12)

3) Hero
     hero loses health when eats poisonous food:

    AssertionError [ERR_ASSERTION]: 120 === 80
    + expected - actual

    -120
    +80

    at Context.<anonymous> (specs/hero_spec.js:110:11)
```

Three tests failing

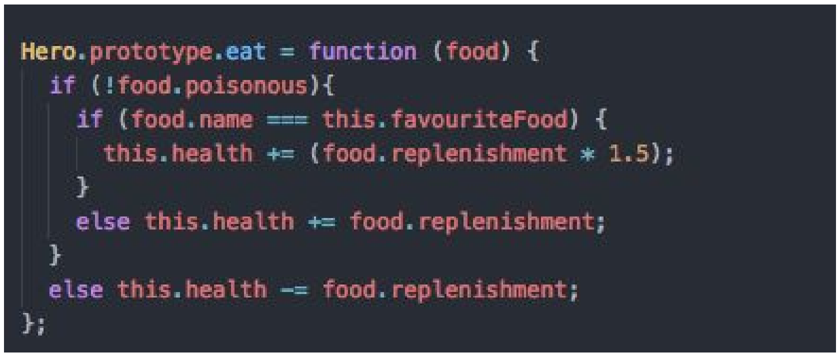


```
Hero.prototype.eat = function (food) {
  if (!food.poisonous){
    if (food.name === this.favoriteFood) {
      this.health += (food.replenishment * 1.5);
    }
    else this.health += food.replenishment;
  }
  else this.health -= food.replenishment;
};
```

Almost fixed function and now only one test is failing

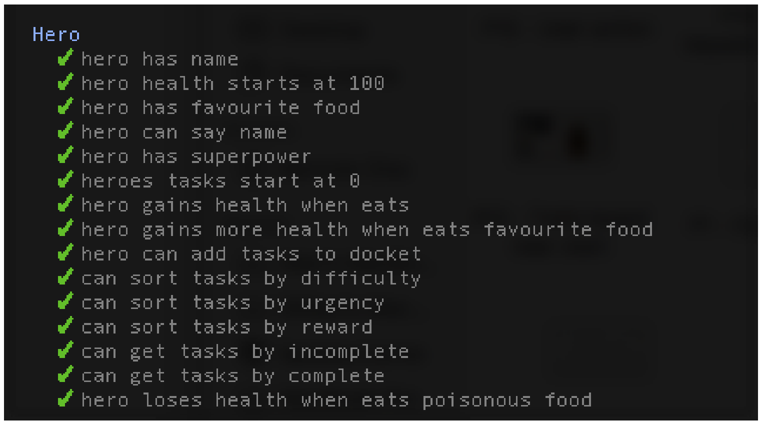


```
30 passing (25ms)
1 failing

1) Hero
     hero gains more health when eats favourite food:

    AssertionError [ERR_ASSERTION]: 120 === 130
    + expected - actual

    -120
    +130

    at Context.<anonymous> (specs/hero_spec.js:58:12)
```



```
Hero.prototype.eat = function (food) {
  if (!food.poisonous){
    if (food.name === this.favouriteFood) {
      this.health += (food.replenishment * 1.5);
    }
    else this.health += food.replenishment;
  }
  else this.health -= food.replenishment;
};
```

Found typo in function



```
Hero
  ✓ hero has name
  ✓ hero health starts at 100
  ✓ hero has favourite food
  ✓ hero can say name
  ✓ hero has superpower
  ✓ heroes tasks start at 0
  ✓ hero gains health when eats
  ✓ hero gains more health when eats favourite food
  ✓ hero can add tasks to docket
  ✓ can sort tasks by difficulty
  ✓ can sort tasks by urgency
  ✓ can sort tasks by reward
  ✓ can get tasks by incomplete
  ✓ can get tasks by complete
  ✓ hero loses health when eats poisonous food
```

Now all tests are passing