

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green. They are positioned diagonally, with the blue one partially covering the green one.

QA CINEMAS PROJECT

Anoush Lowton, George Ryalls,
Jhon Urrego, Kevin Tse, Raphael Fywell



Our Approach

- Split the project into different epics - Front-End, Back-End, Testing, Documentation
- Separately went through the project specification and wrote user stories for each requirement
- Gave each story acceptance criteria and split them up into manageable tasks
- Collectively discussed each user story to assign story points to each task and prioritised them based on MoSCoW methodology

Sprints

- Aimed to reach the MVP and have a functional website by the end of the first sprint
- Wanted to focus on styling and testing in the second sprint
- If there were any unresolved issues from the first sprint those would be prioritised

Designs

- Created wireframe designs for the front-end to visualise the general layout of each webpage
- Wrote data model designs for the database structure
- Assigned each team member specific components to work on for front-end



Technologies Learnt

MERN Stack

- **MongoDB:** A document store (non-relational database) where the documents are stored in BSON (Binary JavaScript Object Notation). We used MongoDB Atlas to host our database.
- **Express:** A framework used to handle functionality in Node.js applications. Works with Mongoose to allow you to interact with a MongoDB instance.
- **React:** A front-end JavaScript library for building web applications. It uses a modular structure with states and components to quickly change and adapt to data.
- **Node:** Node provides a JavaScript runtime which allows it to be separated from the browser and used as the back-end of an application.

Testing

- Mocha - Unit & integration testing back-end
- Chai - Assertions and HTTP
- Istanbul
- Jest



Version Control

- Main ← Dev ← Features
- Each team member works on their own feature branch and pushes commits to that branch
- Create pull requests to merge with dev branch, resolving any conflicts that might arise first
- To minimise the occurrence of merge conflicts, frequently pull from dev and merge dev into the feature branch that you're working on
- Merge dev with main when satisfied that dev functions correctly
- Testing was mostly done on a single branch
- Team members were writing their own tests in different files, which enabled everyone to work simultaneously on the same branch without conflicts





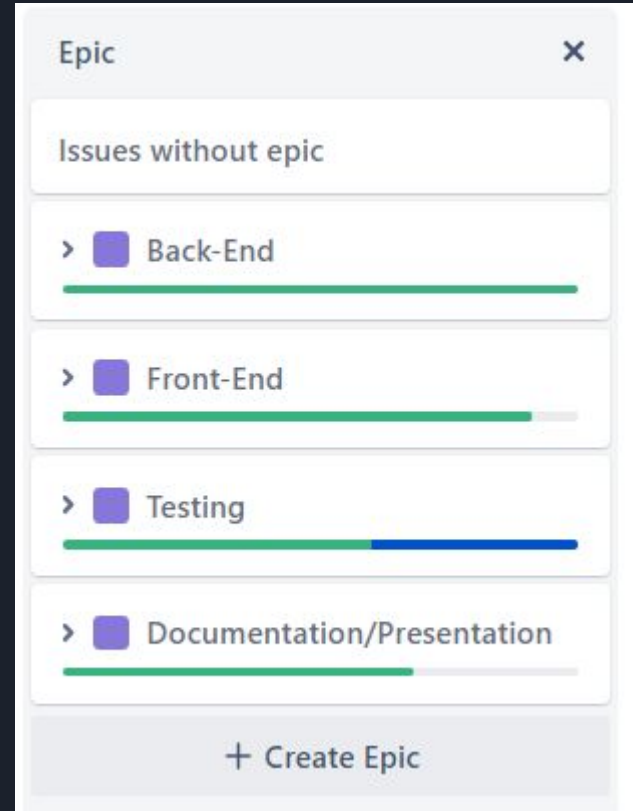
Approach Jira

This project would be completed over two sprints.

The first sprint would be focusing on the Front-End and Back-End epics.

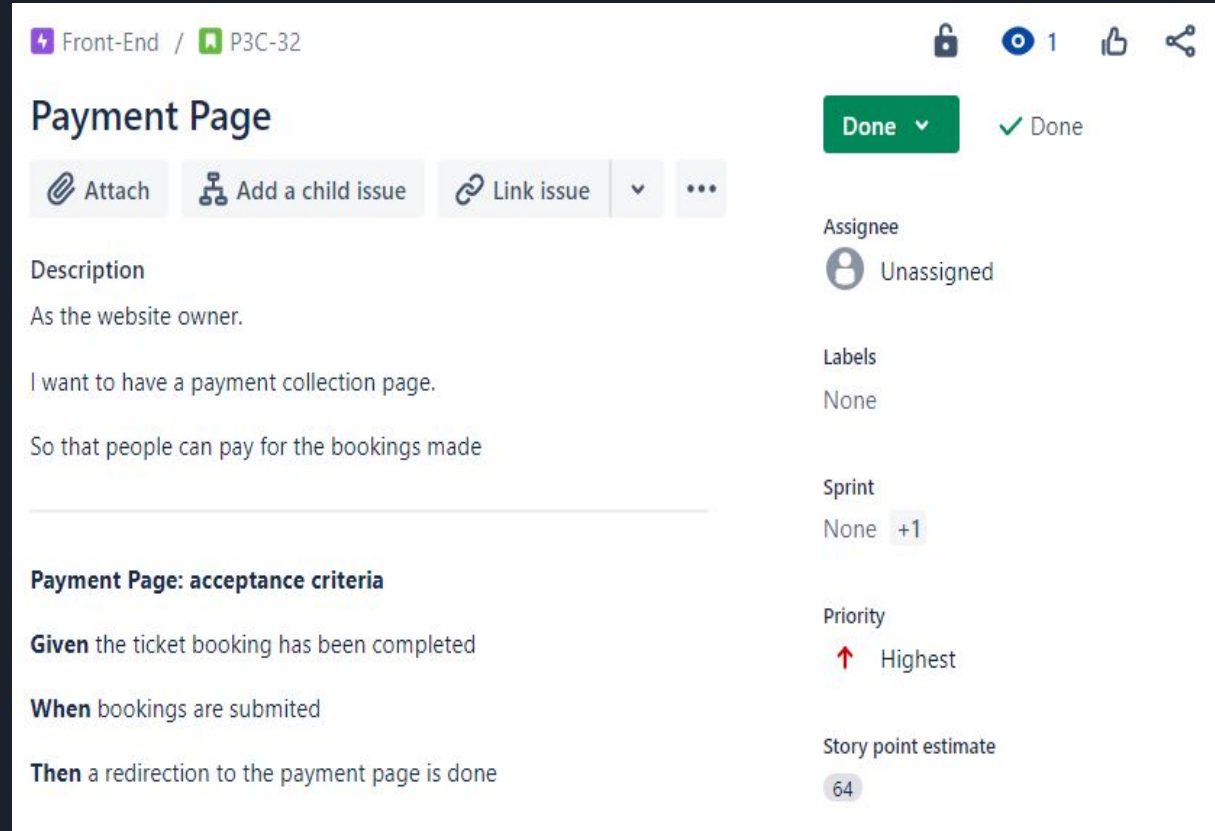
While the second sprint would be Testing.

Documentation would be part of both sprints as some things would need to be updated throughout both sprints



Approach Jira User Story Example

Rough example of how user stories were laid out and each one mentioning what the function was and what the acceptance criteria is



The screenshot shows a Jira issue page for 'Payment Page' in the 'Front-End' project, issue ID 'P3C-32'. The issue is in the 'Done' status. The description includes the role 'As the website owner.', the goal 'I want to have a payment collection page.', and the acceptance criteria 'So that people can pay for the bookings made'. The acceptance criteria are listed as: 'Payment Page: acceptance criteria', 'Given the ticket booking has been completed', 'When bookings are submitted', and 'Then a redirection to the payment page is done'. The right sidebar shows the assignee as 'Unassigned', labels as 'None', sprint as 'None +1', priority as 'Highest', and a story point estimate of '64'.

Front-End / P3C-32

Payment Page

Done

Attach Add a child issue Link issue

Description

As the website owner.

I want to have a payment collection page.

So that people can pay for the bookings made

Payment Page: acceptance criteria

Given the ticket booking has been completed

When bookings are submitted

Then a redirection to the payment page is done

Assignee

Unassigned

Labels

None

Sprint

None +1

Priority

Highest

Story point estimate

64

Approach Jira Tasks Example

One of the tasks within a user story to be completed to help achieve the goal stated in the story

Dedicated page to bookings


Attach Add a child issue Link issue

Description

Create a single page that only deals with ticket bookings

Activity

Show: **Comments** History Newest first ↓



Pro tip: press **M** to comment

Done ▾

✓ Done

Assignee

Unassigned

Labels

None

Sprint

None **+1**


Priority

Highest

Story point estimate

4

Reporter

 Kevin Tse

Data Model Designs

We designed the structure of our collections and documents using data model designs, which we would implement into schemas in Mongoose.

BOOKINGS SCHEMA

```
{
  movieTitle: {
    type: String,
    required: [true, "Movie title is required"],
  },
  date: {
    type: Date,
    required: [true, "Date is required for booking"],
  },
  time: {
    type: String,
    required: [true, "Time is required for booking"],
  },
  name: {
    type: String,
    required: [true, "Name is required for booking"],
  },
  numberOfSeats: {
    type: Number,
    required: [true, "Number of seats are required for booking"],
    min: [1, "Must choose at least 1 seat"],
    max: [30, "You can't book more than 30 seats"],
  },
  adults: Number,
  child: Number,
  concession: Number
}
```

PAYMENTS SCHEMA

```
{
  name: {
    type: String,
    required: [true, "Cardholder's name must be included"],
  },
  cardNumber: {
    type: Number,
    min: [16, "Card number must be 16 digits"],
    max: [16, "Card number must be 16 digits"],
  },
  expiryDate: {
    type: string,
    required: [true, "Expiry date is required"],
  },
  cvc: {
    type: Number,
    min: [3, "cvc must be 3 digit number"],
    max: [3, "cvc must be 3 digit number"],
  }
}
```

DISCUSSION BOARD SCHEMA

```
{
  username: String
  comment: {
    type: String,
    max: [280, "Comment must be less than 280 chars"],
  },
  rating: {
    type: Number,
    min: [1, "Rating must be between 1-5"],
    max: [5, "Rating must be between 1-5"],
  },
  movieTitle: String
}
```

Data Model Designs

This included designing schemas for sub-documents.

MOVIES SCHEMA

```
{
  title: {
    type: String,
    required: [true, "You must include a title"]
  },
  actors: [String],
  director: [String],
  imageUrl: String,
  classification: {
    type: String,
    enum: ['U', 'PG', '12', '12A', '15', '18'],
    message: "{VALUE} is not supported."
  },
  releaseDate: {
    type: Date,
    required: [true, "Movie must have a release date"]
  },
  showingTimes: [SHOWING_TIMES_SCHEMA]
}
```

SHOWING TIMES SCHEMA

```
{
  day: {
    type: String,
    enum: ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"],
    message: "{VALUE} is not supported."
  },
  times: [String]
}
```

Data Model Designs

Some designs evolved over time.

```
BOOKINGS SCHEMA
{
  movieTitle: {
    type: String,
    required: [true, "Movie title is required"],
  },
  date: {
    type: Date,
    required: [true, "Date is required for booking"],
  },
  time: {
    type: String,
    required: [true, "Time is required for booking"],
  },
  name: {
    type: String,
    required: [true, "Name is required for booking"],
  },
  numberOfSeats: {
    type: Number,
    required: [true, "Number of seats are required for booking"],
    min: [1, "Must choose at least 1 seat"],
    max: [30, "You can't book more than 30 seats"],
  },
  adults: Number,
  child: Number,
  concession: Number,
  paymentInfo: PAYMENT_INFO_SCHEMA
}
```

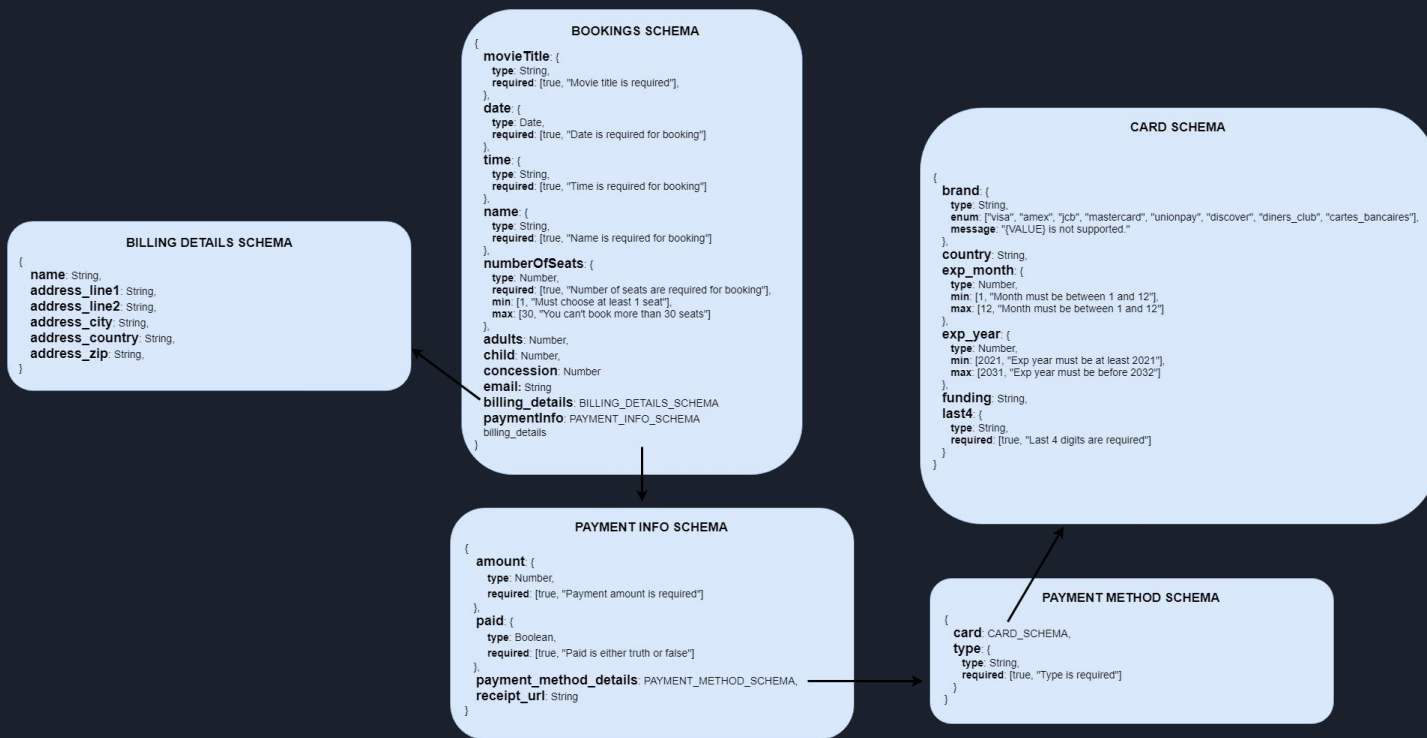
```
PAYMENT INFO SCHEMA
{
  amount: {
    type: Number,
    required: [true, "Payment amount is required"],
  },
  paid: {
    type: Boolean,
    required: [true, "Paid is either truth or false"],
  },
  payment_method_details: PAYMENT_METHOD_SCHEMA,
  receipt_url: String
}
```

```
CARD SCHEMA
{
  brand: {
    type: String,
    enum: ["visa", "amex", "jcb", "mastercard", "unionpay", "discover", "diners_club", "cartes_bancaires"],
    message: "[VALUE] is not supported.",
  },
  country: String,
  exp_month: {
    type: Number,
    min: [1, "Month must be between 1 and 12"],
    max: [12, "Month must be between 1 and 12"],
  },
  exp_year: {
    type: Number,
    min: [2021, "Exp year must be at least 2021"],
    max: [2031, "Exp year must be before 2032"],
  },
  funding: String,
  last4: {
    type: String,
    required: [true, "Last 4 digits are required"],
  }
}
```

```
PAYMENT METHOD SCHEMA
{
  card: CARD_SCHEMA,
  type: {
    type: String,
    required: [true, "Type is required"],
  }
}
```

Data Model Designs

Some designs evolved over time.



Risk Assessment

Description	Evaluation	Likelihood	Impact Level	Responsibility	Response	Control Measures
Method doesn't produce desired result	Back-end won't function correctly	High	Medium	Developer	Refactor method	Create and implement unit and integration tests
Mocha and Chai tests fail	Software may not be functioning correctly	Medium	Medium	Developer	See if failure is due to test code or a fault in the program and correct accordingly.	Write test code correctly and test functionality of new features as they're implemented, to ensure issues are caught quickly.
Jest can't reproduce desired behaviour	Test won't be able to replicate end-user interaction	Medium	Low	Developer	Try to research a solution, otherwise perform the test manually.	
More time spent than planned on tasks	May need to use time allocated for other tasks	Medium	Medium	Developer	Less time spent on other tasks if appropriate	Ensure the project plan is as accurate as possible and manage time properly so that problems with one task don't affect others.
Connection issues with GCP	The program would fail to connect to the DB.	Medium	Medium	Developer	Add support for local MySQL instance	Add support for local MySQL instance
Stuck finding a solution to a problem	Time would be lost	Medium	Low	Developer	Seek help from within the team and if still stuck ask a trainer	Keep on top of course content, complete exercises and practise regularly. Read into anything else that may be useful in the project.
Misunderstood requirement	Refactor would be needed	Low	High	Developer	Refactor to correct any mistakes	Thoroughly read the spec and check frequently to stay on task.
Missed deliverable	Marks would be lost	Low	High	Developer	Attempt to hand in deliverable ASAP	Thoroughly read the spec and check frequently to stay on task.
Can't achieve 80% test coverage	Marks would be lost	Low	Medium	Developer	Seek help from trainers to improve tests	Research Mocha, Chai and Istanbul
Failure to deliver on time	Potentially fail the project	Low	High	Developer		Ensure the project plan is as accurate as possible and accounts for appropriate risks.
Developer becomes unwell	Project would be delayed	Low	High	N/A	Ask for an extension	
Web page doesn't load correctly	User won't be able to interact with back-end	Low	High	Developer	Investigate and refactor code	
Local repository becoming corrupted	All changes made to the local repository will be lost	Low	High	Developer	Pull down the latest commit from GitHub	Regularly push changes to GitHub



Styling decisions

The general template of the site was decided in the beginning by creating a WireFrame diagram which determined the positioning of containers, footers and navbars.

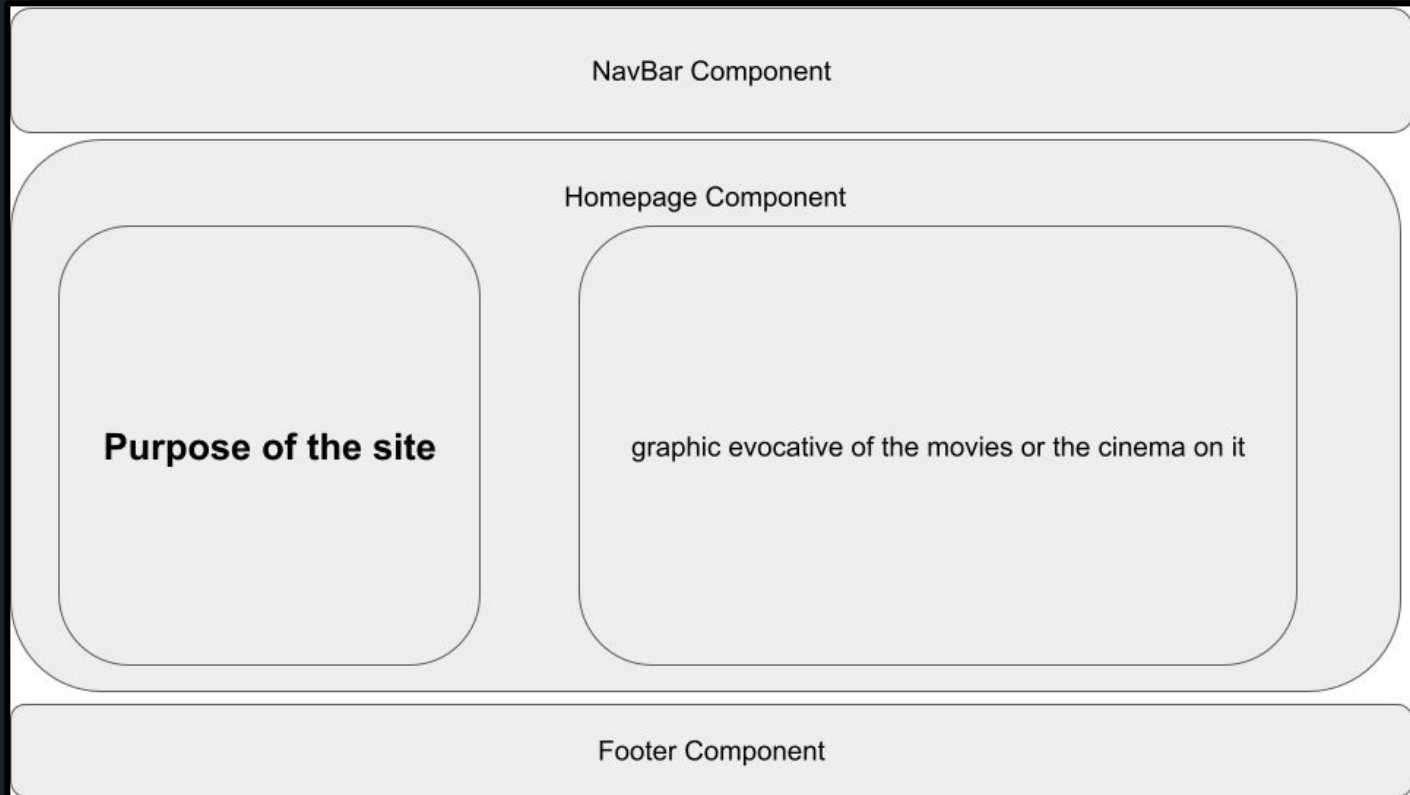
We had a general idea of how we wanted the website to look. We decided to do some styling as we were completing each page to give a general idea of how the finished project would look. Functionality of each page was prioritised first.

We decided on the final design of the website as a result of the development of the web pages over the last two weeks. Key examples that stand out that we decided on were:

- We felt that a light background on a page with a light blue font didn't look visually pleasing. We also studied other commercial Cinema webpages and found that these pages generally had darker themes. As a result of this we experimented with different tones and ultimately decided on a darker background with a blue contrasting text which felt smoother.
- The Home page felt empty and as a result of this we added a carousel of cinema related imagery to improve the feel of the page.
- The Classifications page was changed so that each card had a colour that reflected its classification. This contrast on the page was much preferred by the team.
- Positioning of elements
- We will go through the different iterations of style design in the next slides.



WireFrame diagram: Homepage



Initial design: Classifications page

Welcome!

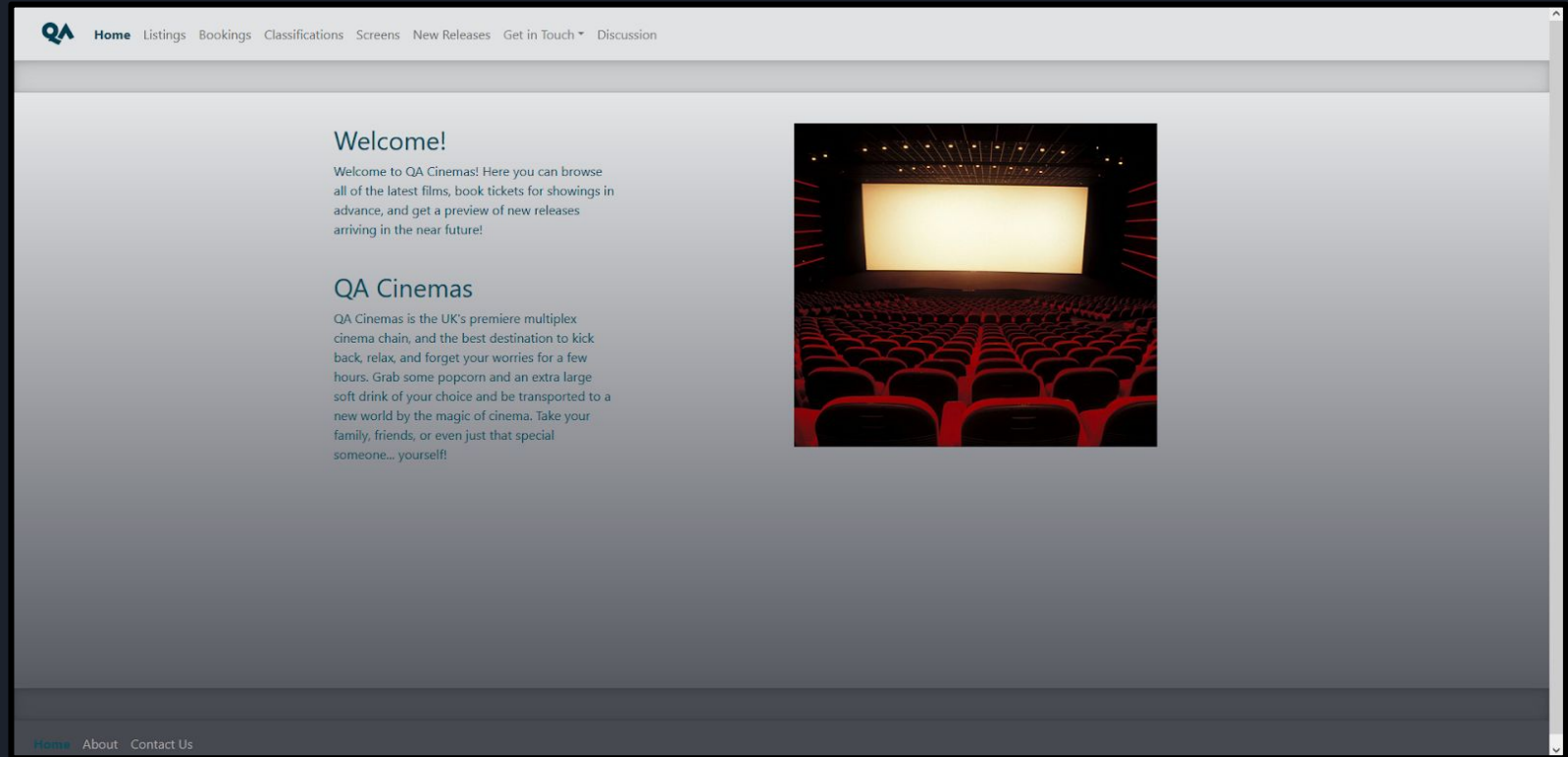
Welcome to QA Cinemas! Here you can browse all of the latest films, book tickets for showings in advance, and get a preview of new releases arriving in the near future!

QA Cinemas

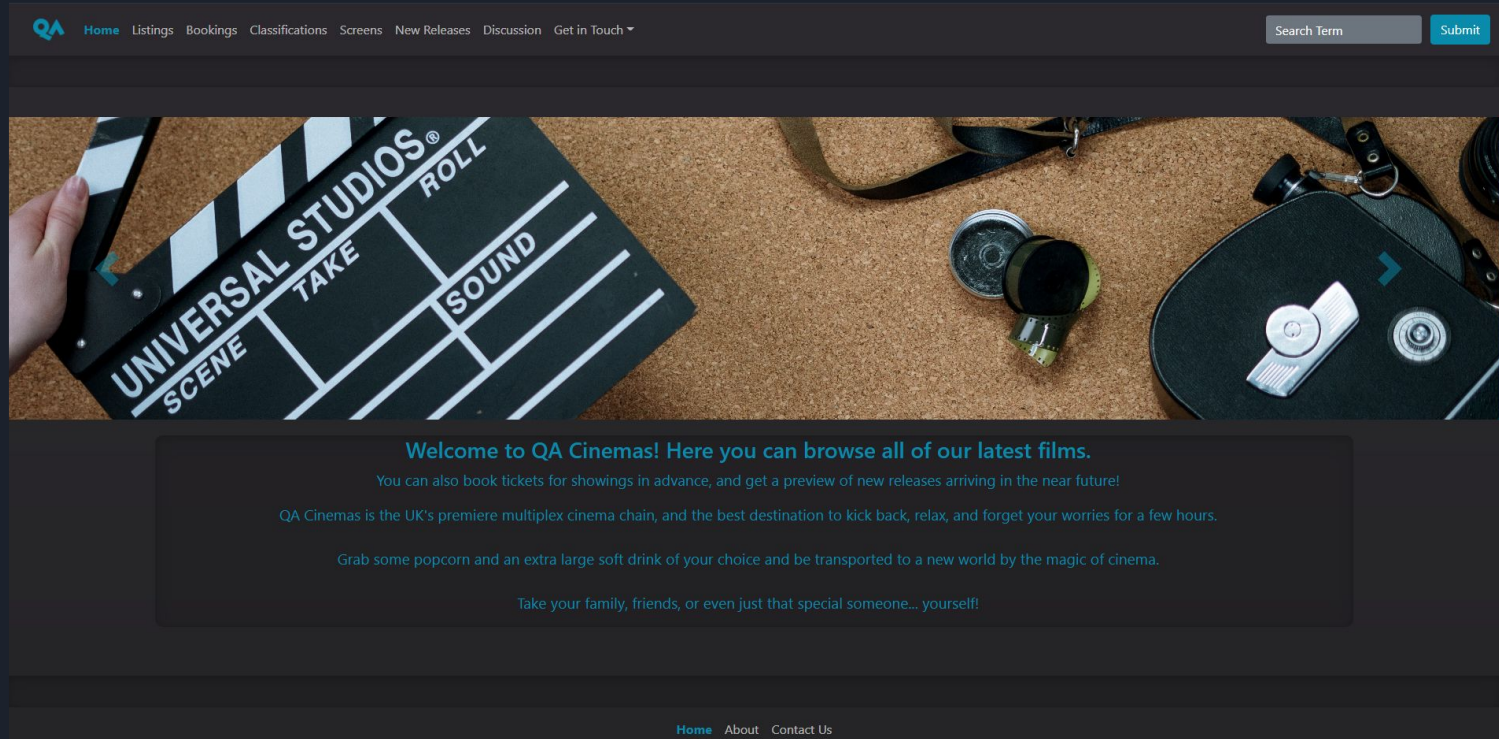
QA Cinemas is the UK's premiere multiplex cinema chain, and the best destination to kick back, relax, and forget your worries for a few hours. Grab some popcorn and an extra large soft drink of your choice and be transported to a new world by the magic of cinema. Take your family, friends, or even just that special someone... yourself!



Second design: Homepage

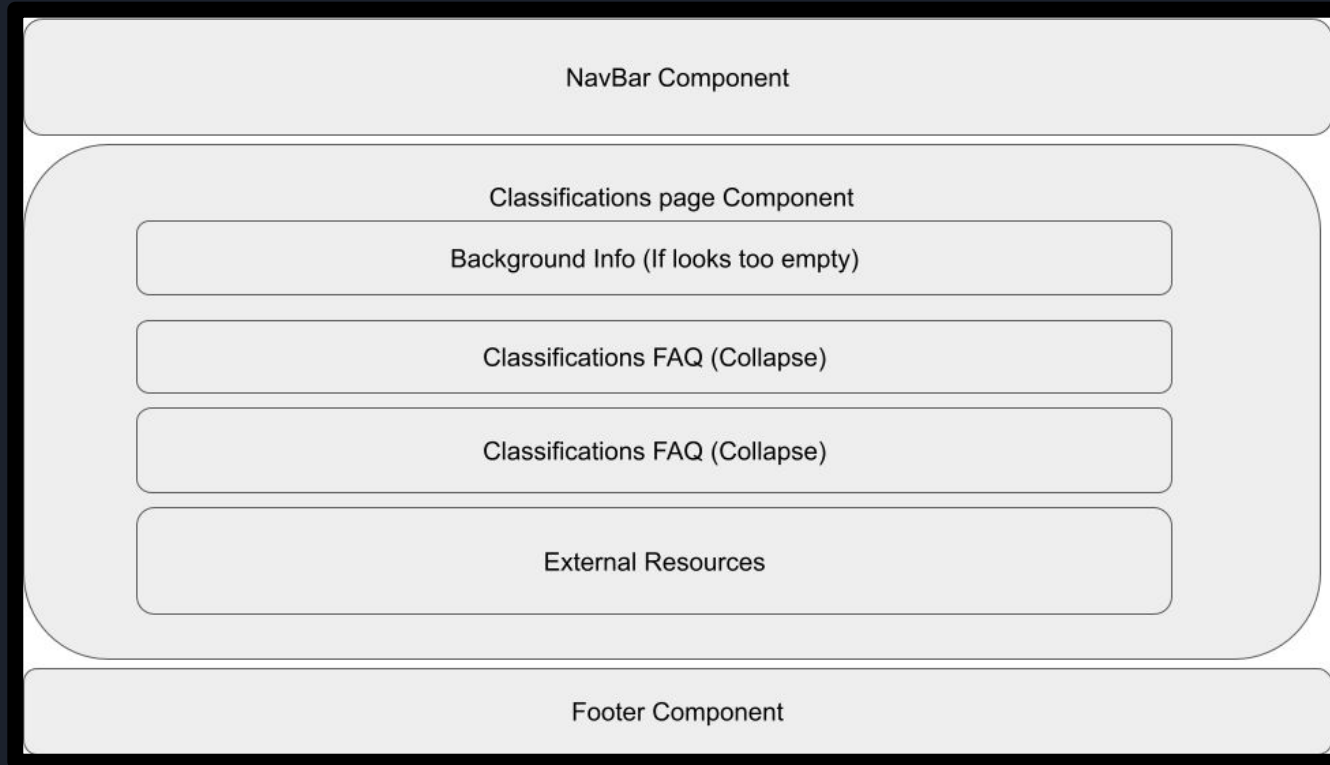


Final design: Classifications page

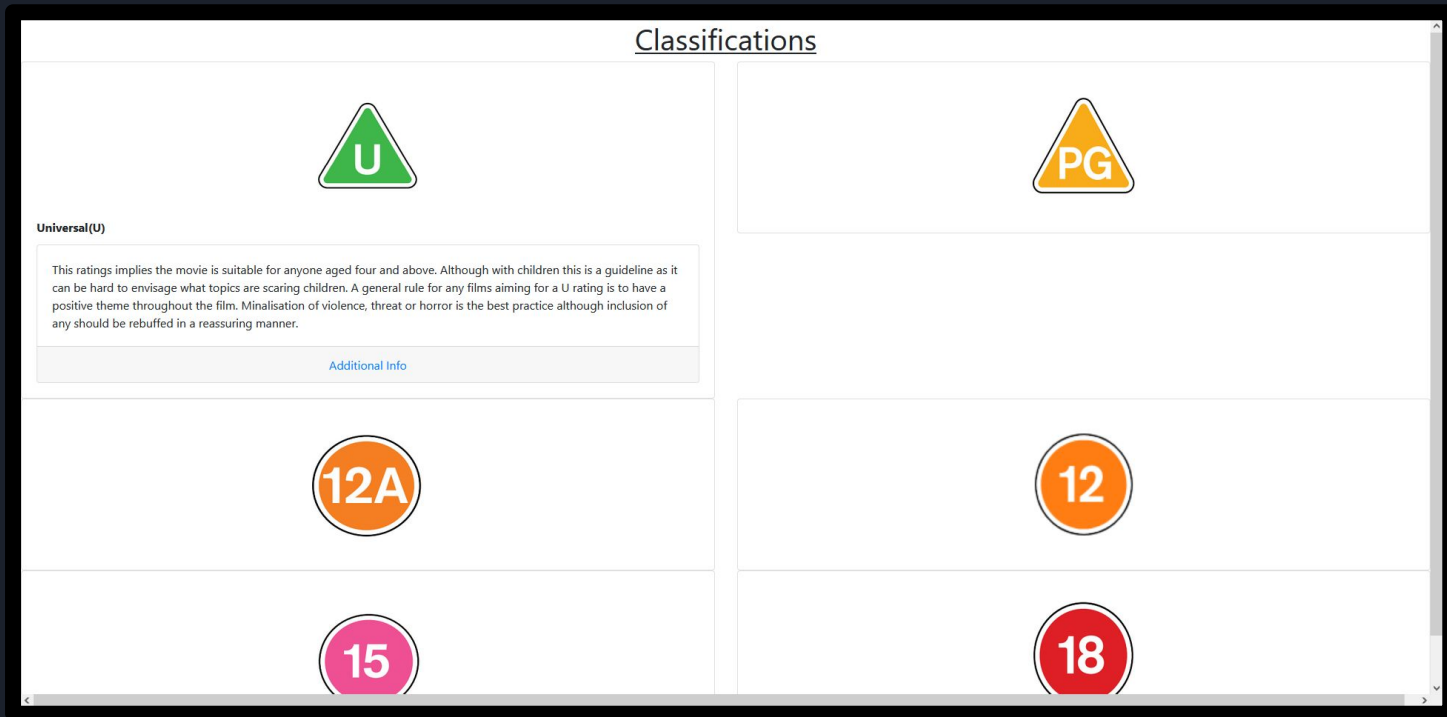




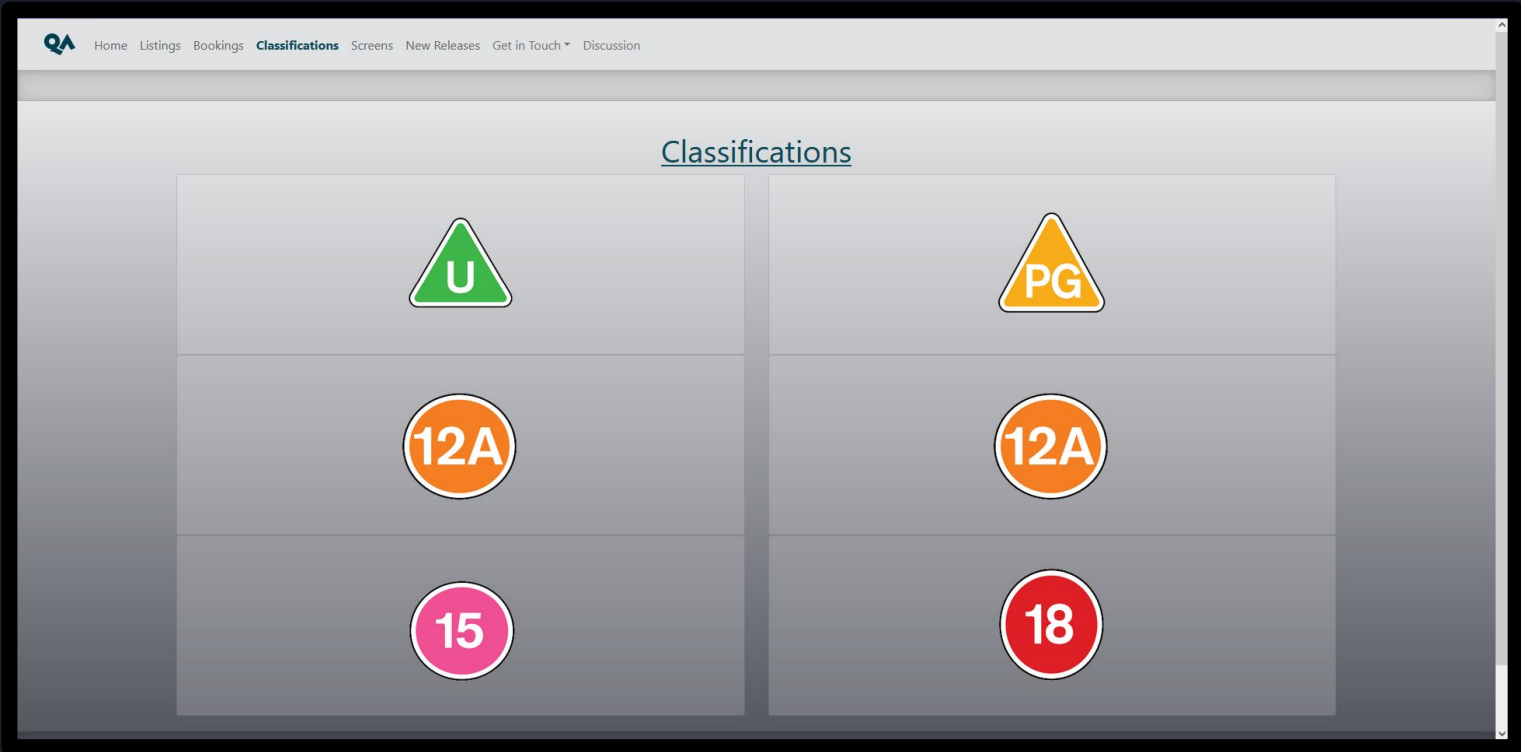
WireFrame diagram: Classifications page



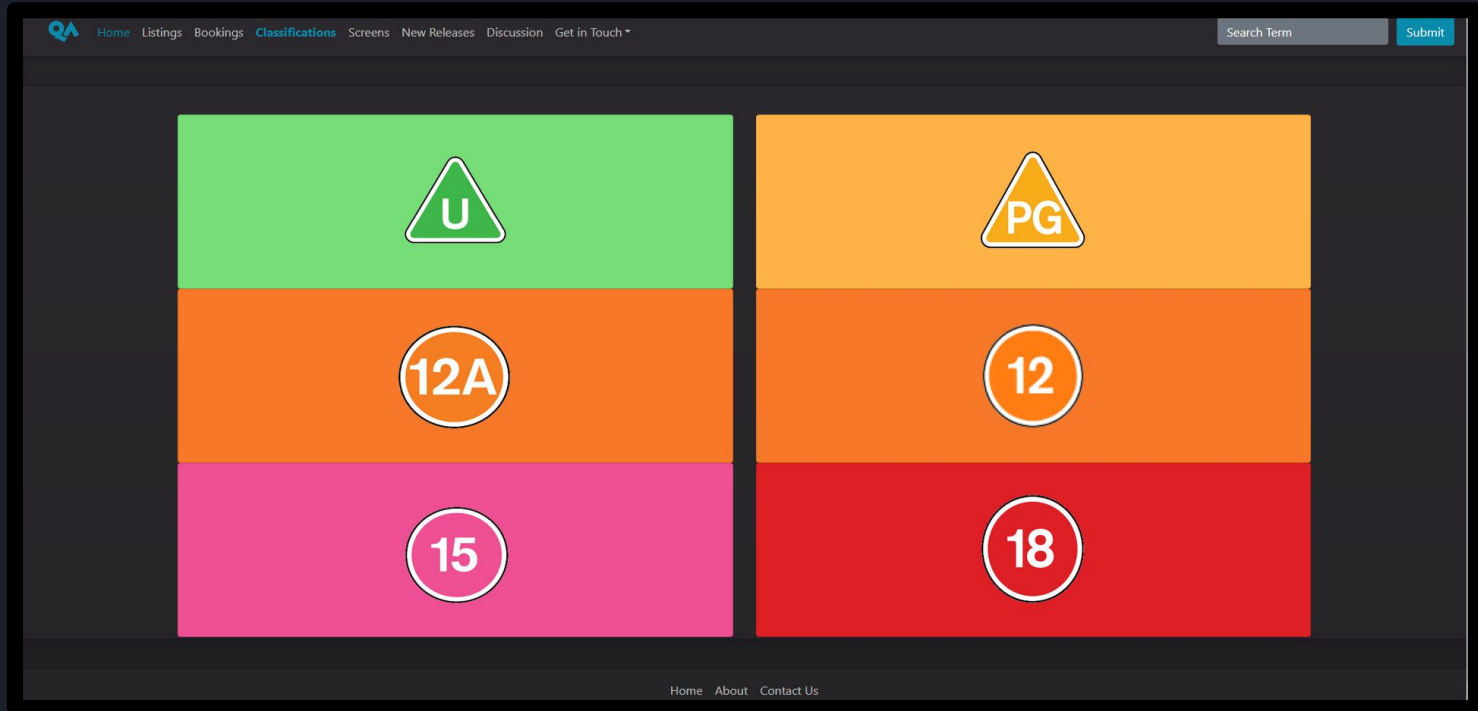
Initial design: Classifications page



Second design: Classifications page



Final design: Classifications page





Back-End Testing

- Mocha was used for testing the back-end.
- Target was 80% line coverage.
- Chai was used for assertions.
- Chai-http was used for performing API requests and integration testing the back-end.

Back-End Testing

- Here's an example of a get request test, a post request test, and a failed request due to violating a constraint from the associated schema.

```
describe('Testing the discussion board routes', function() {  
  
  it('Should complete getAll request with status 200', function(done) {  
    CHAI.request(APP)  
      .get('/discussions/getAll')  
      .end(function(err, response) {  
        EXPECT(err).to.be.null;  
        EXPECT(response).to.have.status(200);  
        done();  
      });  
  });  
  
  it('Should complete post request with status 201', function(done) {  
    CHAI.request(APP)  
      .post('/discussions/post')  
      .send({username: 'test', 'comment': 'test comment', 'rating': '2', 'movieTitle': 'Inception'})  
      .end(function(err, response) {  
        EXPECT(err).to.be.null;  
        EXPECT(response).to.have.status(201);  
        done();  
      });  
  });  
  
  it('Should fail post request due to rating validation', function(done) {  
    CHAI.request(APP)  
      .post('/discussions/post')  
      .send({name: 'test', 'comment': 'test comment', 'rating': '0', 'movieTitle': 'Inception'})  
      .end(function(err, response) {  
        EXPECT(response).to.have.status(500);  
        done();  
      });  
  });  
});
```


Back-End Testing

- Total line coverage of the back-end folder was 90.48%

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	84.38	60	83.33	84.38	
back-end	90.48	50	66.67	90.48	
index.js	90.48	50	66.67	90.48	27,33
back-end/Models	100	100	100	100	
booking.js	100	100	100	100	
discussionboard.js	100	100	100	100	
movie.js	100	100	100	100	
back-end/Routes	77.59	62.5	88.89	77.59	
bookings.js	56.25	100	50	56.25	10,24-38
discussionboard.js	92.86	100	100	92.86	13
movies.js	82.14	62.5	100	82.14	13,22,31,40,55



Back-End Testing - Challenges

- One challenge was that a lot of the back-end code was made of up of schemas, which serve no purpose until integrated with other parts of the application. So it was hard to devise unit tests to test these files.



Front-End Testing

Testing the front-end was done using JEST. Jest is a JavaScript testing framework focused on simplicity. Its aims to create minimal fuss for users in creating tests trying to make it simple and fast to write.

Types of testing used in the front-end:

- ***Snapshot*** tests were done to ensure the rendered UI does not change unexpectedly by taking a current snippet of what is currently on the page and comparing to a previous version.
- ***Unit*** tests were done to ensure components rendered as expected.
- ***User Acceptance*** tests were done to simulate user journeys through the website.

- 11 testing suites
- 36 tests
- 12 snapshots
- 84% line coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	76.9	47.5	63.64	84.38	
src	66.67	100	66.67	66.67	
App.js	66.67	100	66.67	66.67	72
src/Components	43.48	0	28.57	52.63	
Homepage.jsx	43.48	0	28.57	52.63	19-21,25-27,31-32,40
src/Components/About	100	100	100	100	
About.jsx	100	100	100	100	
src/Components/Classifications	77.36	100	40	100	
Class12.jsx	75	100	33.33	100	
Class12A.jsx	75	100	33.33	100	
Class15.jsx	75	100	33.33	100	
Class18.jsx	75	100	33.33	100	
ClassPG.jsx	75	100	33.33	100	
ClassU.jsx	87.5	100	66.67	100	
Classifications.jsx	80	100	50	100	
src/Components/ContactPage	100	100	100	100	
ContactPage.jsx	100	100	100	100	
src/Components/DiscussionBoard	85.19	75	71.43	84.91	
CommentsForm.jsx	81.08	75	61.54	81.08	17-18,37-40,59,86,100
DiscussionBoard.jsx	94.12	100	87.5	93.75	33
src/Components/Getting_There	100	100	100	100	
GettingThere.jsx	100	100	100	100	
src/Components/Header_Footer	80	100	50	100	
FooterBar.jsx	80	100	50	100	
NavBar.jsx	80	100	50	100	
src/Components/Listings	71.05	33.33	63.64	77.14	
ListingGallery.jsx	71.05	33.33	63.64	77.14	26-28,32-33,44-45,53
NewReleases.jsx	71.05	33.33	63.64	77.14	26-28,32-33,44-45,53
src/Components/OpeningTimes	100	100	100	100	
OpeningTimes.jsx	100	100	100	100	
src/Components/PlacesToGo	100	100	100	100	
PlacesToGo.jsx	100	100	100	100	
src/Components/Screens	60.87	20	57.14	70	
Screens.jsx	60.87	20	57.14	70	49-51,55-56,63
src/Components/TicketBooking	88.31	78.57	78.13	92.31	
FormBooking.jsx	88	78.57	77.42	92.06	33,48-49,116,129
TicketBooking.jsx	100	100	100	100	

Test Suites: 11 passed, 11 total

Tests: 36 passed, 36 total

Snapshots: 12 passed, 12 total

Time: 13.623 s

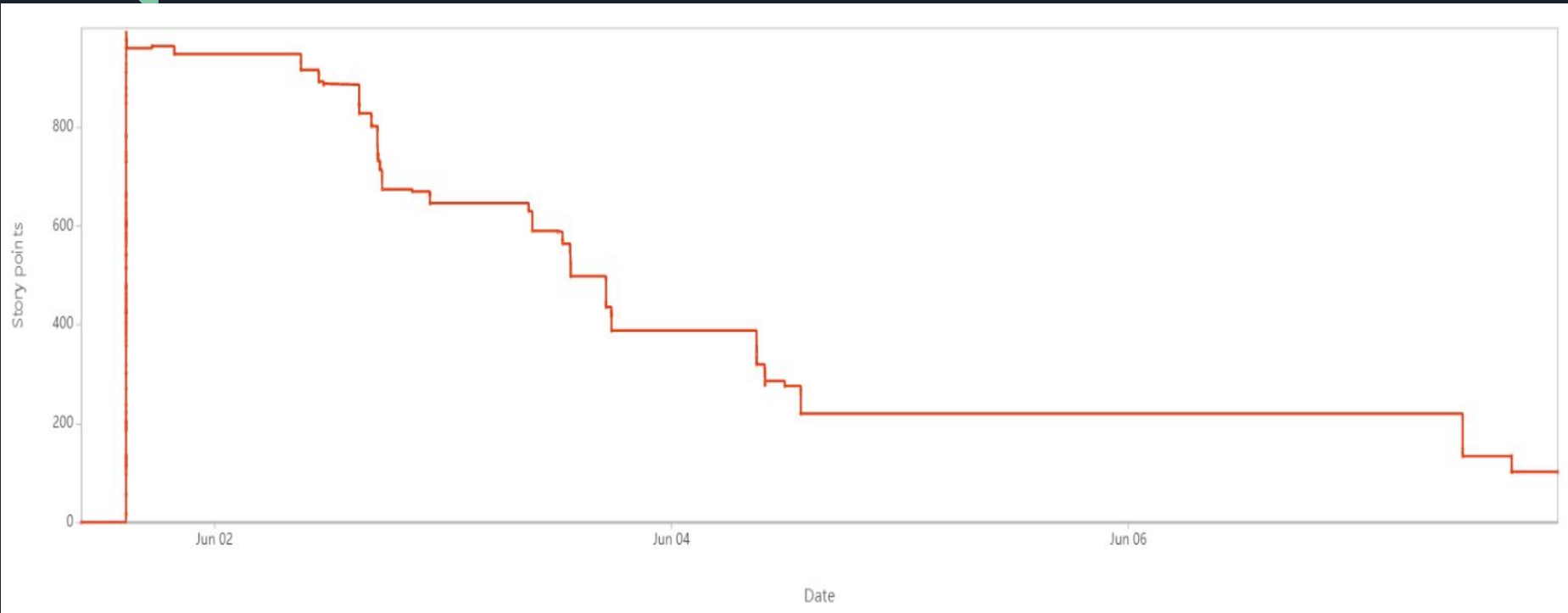
Ran all test suites related to changed files.



Demonstration

Sprint 1 Review

894/996 Story points





Sprint 1 Review

At the end of the first sprint there was a few things that were to be carried over into the second sprint

- Styling of the website was the biggest section to be carried
- Some final touches on some component pages were also left to fulfil.

Key	Summary	Issue type	Epic
P3C-93	Add explanation of how the team used Scrum.	✓ Task	FRONT-END
P3C-95	Create custom logo that fits with the overall design aesthetic of the site.	✓ Task	FRONT-END
P3C-94	Make scrum content a visibly distinct and separate section of the page.	✓ Task	FRONT-END
P3C-96	Add logo to front page.	✓ Task	FRONT-END
P3C-87	Wishlist	📖 Story	FRONT-END
P3C-98	Global Styling for Pages	📖 Story	FRONT-END

Sprint 2 Review

256/334 Story points



Sprint 2 Review

Second sprint was mainly testing and creating the presentations.

- All incomplete items were from the wish list.

Key :	Summary :	Issue type :	Epic :	Status :
P3C-93	Add explanation of how the team used Scrum.	✓ Task	FRONT-END	TO DO
P3C-95	Create custom logo that fits with the overall design aesthetic of the site.	✓ Task	FRONT-END	TO DO
P3C-96	Add logo to front page.	✓ Task	FRONT-END	TO DO
P3C-87	Wishlist	📖 Story	FRONT-END	TO DO
P3C-13	As a QA Cinemas user I want to present the completed project so that the work done i...	📖 Story	DOCUMENTATION/...	IN PROGRESS



Sprint Retrospective

Positive Outcomes:

- All website requirements completed.
- Some website wish list included.
- Estimations were mostly accurate.
- Daily stand ups performed well to bring everyone up to speed.

Future Improvements:

- Some of the tasks were added to testing during the sprint.
- Time management of problem solving could be managed better



Conclusion