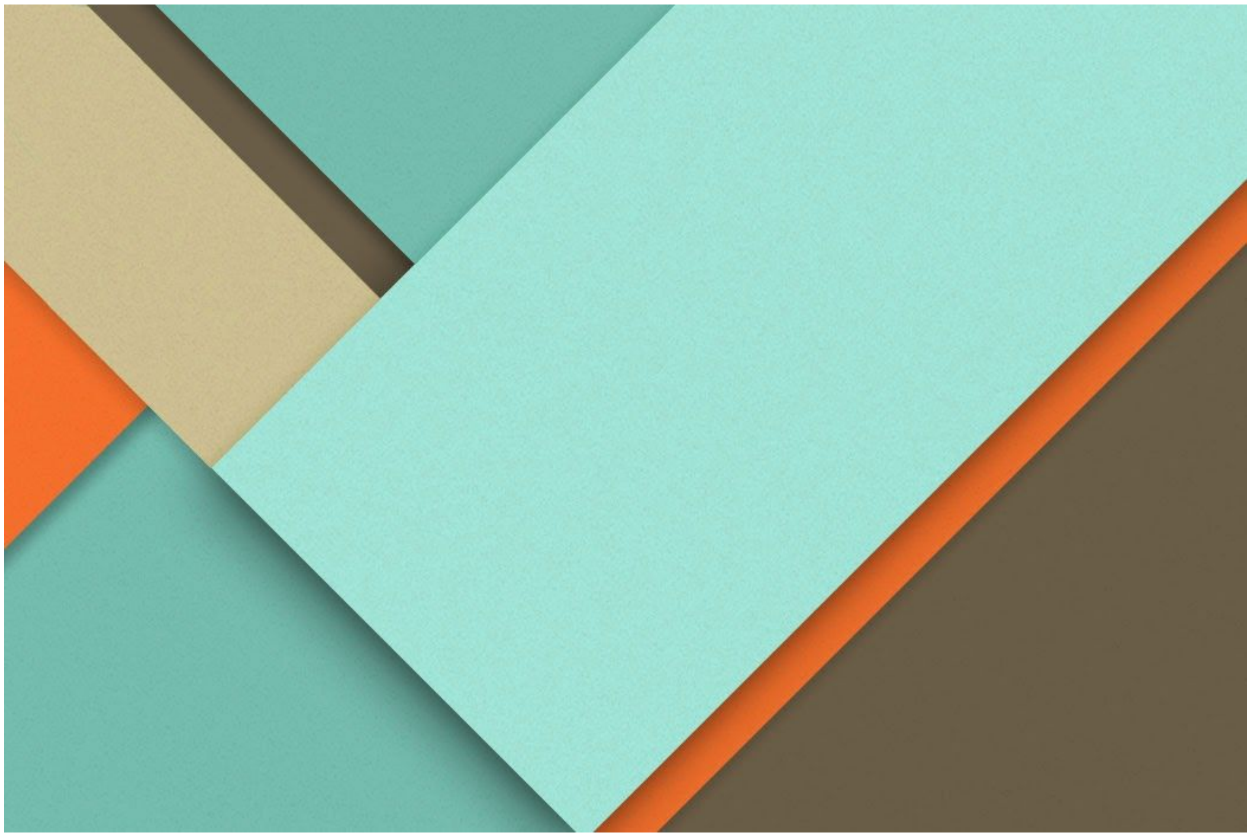


Amazon Product Review Sentiment Analysis

Devin Thomas

Contents

References

[1] Article: https://www.buzzfeednews.com/article/nicolenguyen/amazon-review-reuse-fraud#4ldqpgc

[2]AWS Amazon Comprehend: https://www.aws.amazon.com/comprehend/?nc2=type_a

[3] Stanford Research Paper: http://www.cs229.stanford.edu/proj2018/report/122.pdf

[4] Fakespot: https://www.fakespot.com

[5] Python: https://www.python.org

[6] Django: https://www.djangoproject.com

[7] ReactJS: https://www.reactjs.org

[8] React Bootstrap: https://www.react-bootstrap.github.io

[9] BeautifulSoup: https://www.pypi.org/project/beautifulsoup4/

[10] ScrapeHero: https://www.scrapehero.com/how-to-scrape-amazon-product-reviews/

[11] Free Proxy List: https://www.free-proxy-list.net

Analysis

The Problem

In the first quarter of 2018, e-commerce increased by 2% but rose to a 20% increase in the first quarter of 2019. Presently, a larger number of people are moving from physical shopping to online shopping. As a result e-commerce companies have benefited greatly. Amazon, the largest e-commerce company in the world, saw net sales rise by 31% to $232.9 Billion in 2018. However, the issue with online shopping is that we, as consumers, cannot guarantee the quality of the products that we are buying because we only view a picture of the item. "Customers normally don't read reviews. ... They usually base their decision on how many reviews the product has and what the star rating is." This is a direct quote from an anonymous Amazon seller[1]. They highlight how people do not normally read the reviews but look at the rating instead. We often rely on other buyers and their experience to form our own decisions and ultimately buy or ignore the product. Several factors come into account such as country of origin, seller, reviews, ratings and number of both reviews and ratings. All these factors allow us to make justified decisions to ascertain if the product is worth it.

My solution proposes to streamline the process of deciding whether a product is beneficial and if the user is likely to be satisfied with it. By allowing the user to select a product from Amazon's website, my solution will analyse the entirety of the information provided about the product in order to give a percentage confidence that the product is good or not. This complete solution is not currently available anywhere due to the niche nature of it. However, it is essential as it would provide a good tool in being able to trust products that appear on Amazon. By initially buying the best product, it will save the user from having to return items and buy another: a laborious and tedious process. By cutting down this process the user can therefore streamline the process of online shopping through Amazon. Additionally, businesses would benefit by looking at their buyers' opinions about their products and researchers would also be able to use the information to do in-depth analyses of market trends which may lead to better predictions of the stock market.

A computer is best suited to solving this problem as there is an inherent bias that lies within humans when making decisions. For example, we may decide that the product with

[1] https://www.buzzfeednews.com/article/nicolenguyen/amazon-review-reuse-fraud#4ldqpgc

the best rating is the best when in fact another cheaper but identical product would be ideal as it saves money. The system will be required to read through possibly thousands of reviews and determine the sentiment of each review in order to depict whether a product is of good quality. A computer can therefore read these reviews much faster and gain a less biased view of the product. Additionally, the system can use computational features such as objects and classes to represent each review displayed and sequentially work through each review.

Computational methods allow this problem to be solved as a physical method would not be efficient or simple to implement. This solution will involve large amounts of calculations and computations which would be impossible for humans to do alone. The repeated iterative process of reading reviews to determine their sentiment and report back to the user would be much more effectively done by a computer. Additionally, a computer follows a strict sequential nature and utilises fast processing speeds. This means that all of the calculations will be dealt with quickly and efficiently so that the user experiences little to no delay - something that would be prevalent if solved by physical or human methods.

Currently, there are no non-computational methods that solve this solution due to the fact that Amazon is an online company and all of their services are online. The only two ways to determine sentiment are for a human to read reviews or create a machine that will read them and create links to decide. The former method is too laborious and so the latter would be the best solution to read through products with an excess of 1000 reviews.

Target Market

This product would be aimed towards shoppers who use Amazon. Therefore, the demographic of this product would be the same as all users of Amazon's services. This includes the 310 Million active users of Amazon who are looking to gain an insight into whether the product that they intend to buy is good quality and worth the money. Narrowing this target market down we can argue that it would be more useful for all shoppers who are aged 18 and above considering the fact that in order to buy an item online, you must be of this age. Users are likely to be technologically advanced due to the nature of buying online. Therefore, the system must be easily navigable and simple to use. This allows elderly users who may not be as familiar with technology to easily use the system.

Additionally, other users may include businesses who are looking to find out about their product and the overall sentiment about it. This will provide them with a view of the public's opinion without having to go out and survey. Being a free solution, the businesses would save money as they do not have to spend money on sending out surveys and handling with the incoming data. However, businesses might be more inclined to user a more professional and bespoke solution.

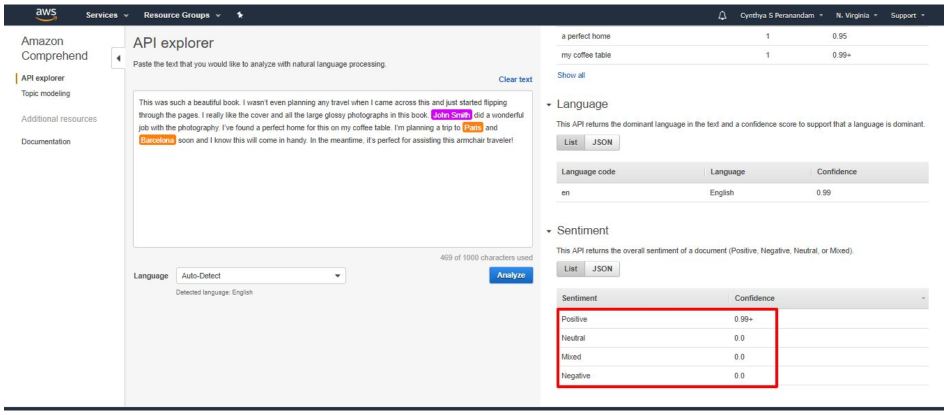
Research

Pre-existing Solutions

Currently, there are many researchers who have produced papers which carry out a very similar experiment in which they sample many Amazon reviews and attempt to evaluate the sentiment of the product whilst predicting whether the review is a highly rated or lowly rated product.

As such, I have aimed to research three different solutions that all encompass some form of my solution.

AWS Amazon Comprehend



API explorer

Paste the text that you would like to analyze with natural language processing.

This was such a beautiful book. I wasn't even planning any travel when I came across this and just started flipping through the pages. I really like the cover and all the large glossy photographs in this book. John Smith did a wonderful job with the photography. I've found a perfect home for this on my coffee table. I'm planning a trip to Paris and Barcelona soon and I know this will come in handy. In the meantime, it's perfect for assisting this armchair traveler!

469 of 1000 characters used

Language: Auto-Detect

Detected language: English

a perfect home	1	0.95
my coffee table	1	0.99+

Language

This API returns the dominant language in the text and a confidence score to support that a language is dominant.

Language code	Language	Confidence
en	English	0.99

Sentiment

This API returns the overall sentiment of a document (Positive, Negative, Neutral, or Mixed).

Sentiment	Confidence
Positive	0.99+
Neutral	0.0
Mixed	0.0
Negative	0.0

Figure 1. AWS Amazon Comprehend

Amazon Comprehend[2], shown in Figure 1., already is a very good solution to the problem at hand. It will provide a confidence for each of the different possible sentiments in a very clear and user friendly way (highlighted in red). The user inputs a review and the page will return different details such as the language, sentiment, entities such as places or people and keyphrases.

However, there are a few disadvantages to this service. Firstly, to use this service, you are required to pay in order to access Amazon Comprehend. Though it is free to create an AWS account, you must give your credit card details so that you can pay for the other services provided. For many, the added expenses of this will be enough to turn them away from using the service. As a result, I believe that my solution should be free and easily accessible to all. This is so that using it does not become a hassle and a greater expense than it is worth. Additionally, you can only analyse one review at a time in the example above and so

[2] https://www.aws.amazon.com/comprehend/?nc2=type_a

it does not give an overall depiction of the quality of a product. Due to this, it is very unlikely that people will individually input each review and assess whether each one is positive.

Good Features	Missing/Bad Features
Various analytics create a strong analysis of each individual review.	Users have to analyse one review at a time.
More analyses than just sentiment.	Costs money.
Complex solution indicates rigorous analysis in the backend.	Unless you are a large company, the analytics are not useful.

Stanford Research Paper

Another already existing solution to the problem is addressed in a paper published by researchers at Stanford[3]. Within this paper, the three researchers aimed to classify the positive and negative reviews of customers across a range of different products. They then built a supervised learning model to polarise the vast amount of reviews.

By testing a wide variety of different machine learning algorithms such as Naive Bayes analysis, K-Nearest Neighbours and a Recurrent Neural Network. In the end, they used 13 different models and gained various test accuracies between 52% to 71%.

To begin, the researchers took their dataset and preprocessed it by removing all the irrelevant fields such as reviews.dateAdded — representing the date the review was published to Amazon. An issue with their dataset was that it is heavily biased towards 5 star reviews. There were 23,775 4.8–5 star reviews compared to only 410 0–1.2 star reviews. Additionally, the researchers noticed that there were repeated samples in the dataset. This introduces more bias to the dataset and can therefore lead to model overfitting - an issue that they ended up experiencing.

Good Features	Missing/Bad Features
Various different models were tested for different accuracies and results.	Only a research paper — no tangible end product or solution produced.
Utilises a large dataset of 34,660 data points and so reliable models were more likely to be produced than from a small	Best model test accuracy: 71.5% Worst model test accuracy: 52.4% Overfitting was a large factor and so they

[3] http://www.cs229.stanford.edu/proj2018/report/122.pdf

dataset.	produced low test accuracy. They have stated that this was due to data imbalance and a dataset that wasn't large enough.
	Heavily biased dataset towards positive reviews.

Fakespot

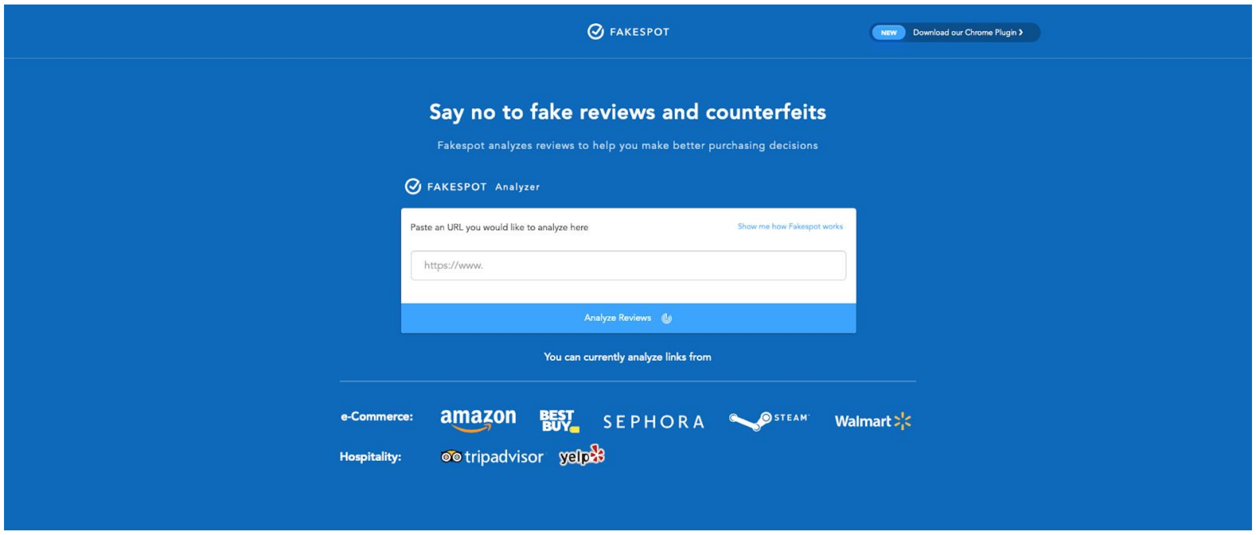

FAKESPOT
NEW Download our Chrome Plugin
Say no to fake reviews and counterfeits
Fakespot analyzes reviews to help you make better purchasing decisions
FAKESPOT Analyzer
Paste an URL you would like to analyze here
Show me how Fakespot works
https://www.
Analyze Reviews
You can currently analyze links from
e-Commerce:
amazon
BEST BUY
SEPHORA
STEAM
Walmart
Hospitality:
tripadvisor
yelp

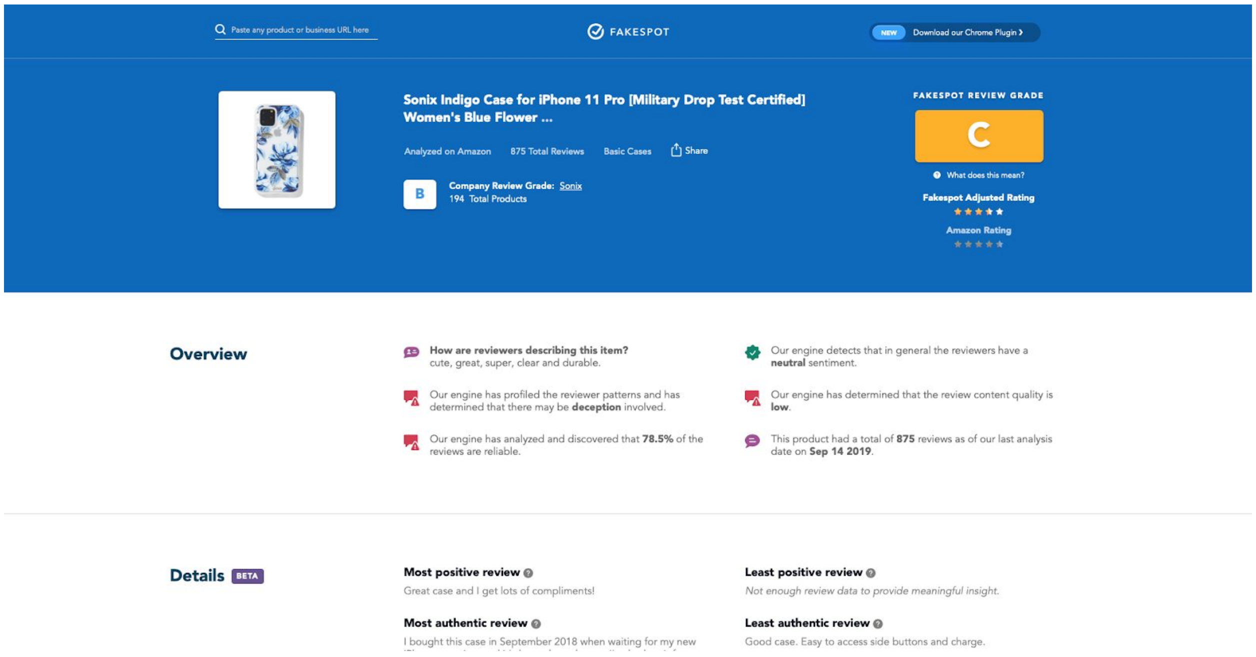
Figure 2. Fakespot

Fakespot[4], shown in Figure 2., is an online service that aims to provide consumers with a new way of filtering product reviews in order to find out what real users are saying about various products online. Their technology analyses millions of product reviews whilst looking for suspicious patterns and incentivised reviews. As a result, they then flag reviews and products that come up as unreliable.

The previous two solutions, though fit the end result better, lack in presentation and overall usability. This is what Fakespot makes up for; though it only looks at reviews for falsity and unreliability, it is presented in a much better format and analyses all the reviews of a product rather than individually.

The way in which the user inputs the URL of the amazon product that they are looking at is a very good and easy to use method and I believe this would be a good way to source the product from the user. As a result, validation will have to be carried out in order to ensure only Amazon products are used to begin with. Like Fakespot, it may be beneficial in the future to support the use of multiple different e-commerce sites.

[4] https://www.fakespot.com



Paste any product or business URL here

FAKESPOT

NEW Download our Chrome Plugin ›

Sonix Indigo Case for iPhone 11 Pro [Military Drop Test Certified] Women's Blue Flower ...

Analyzed on Amazon 875 Total Reviews Basic Cases Share

B **Company Review Grade:** Sonix
194 Total Products

FAKESPOT REVIEW GRADE

C

What does this mean?

Fakespot Adjusted Rating

Amazon Rating

Overview

How are reviewers describing this item?
cute, great, super, clear and durable.

Our engine detects that in general the reviewers have a **neutral** sentiment.

Our engine has profiled the reviewer patterns and has determined that there may be **deception** involved.

Our engine has determined that the review content quality is **low**.

Our engine has analyzed and discovered that **78.5%** of the reviews are reliable.

This product had a total of **875** reviews as of our last analysis date on **Sep 14 2019**.

Details BETA

Most positive review
Great case and I get lots of compliments!

Least positive review
Not enough review data to provide meaningful insight.

Most authentic review
I bought this case in September 2018 when waiting for my new

Least authentic review
Good case. Easy to access side buttons and charge.

Figure 3. Fakespot Analysis

Figure 3. is an example of the output of analysing the reviews for a specific product. As shown, the user is able to see a review grade for the product based on existing reviews and creates an adjusted rating to the one on Amazon. This is therefore likely to give the user a more holistic view on the product and allow them to make a better decision on whether they want to buy the product.

Good Features	Missing/Bad Features
Good, intuitive User Interface and User Experience.	Only looks at the reliability of reviews for products and not the quality or positivity of reviews.
Free to use with no additional payments for other features.	
Distinguishes between real and fake reviews and products.	

Evaluation

Looking at all of the pre-existing solutions, the clearest similarity is that they have different demographics but inherently use similar technologies. AWS Amazon Comprehend is aimed towards businesses looking to analyse their products and reviews on those products. The research paper from Stanford is intended as research so that it may be useful to others in the future. As such, there is no interaction between users and their solution. Finally, Fakespot is designed with individual buyers in mind. With it's simple User Interface and easy User Experience it outperforms the other two solutions though it lafalters cks in the overall intention of the solution. By this, I am suggesting that Fakespot does not analyse reviews for whether they are positive or negative but for the falsity of reviews. It does have a section which displays the most positive but this is unhelpful as it has no gauge of the overall sentiment on the product from all of the reviews. Therefore, my solution will be aimed at all three types of demographics: individual users, businesses and researchers. By creating a useful and easy to manipulate web page, different users will be able to use it to accommodate their different needs. The solution will deliver on reporting if the product is good to the buyers and the businesses

Additionally, out of the three solutions presented, only the research paper provided a form of solution and how it was achieved. Both AWS Amazon Comprehend and Fakespot are proprietary and most likely profit from the service that they are dealing; as such they are unlikely to share the technology behind how they work. Due to this, my solution will aim to look and report like Fakespot whilst utilising the inner technology behind the research paper.

Another feature which is evident in both Fakespot and AWS Amazon Comprehend is the variety of different analytics. AWS Amazon Comprehend provides a variety of features in addition to the sentiment analysis such as entity detection and language. Comparatively, Fakespot will provide the details, helpful insights and similar products in addition to the sentiment analysis of the reviews. In order to most effectively help the user, multiple different facets of the information should be provided. In my solution I will begin with the basics and as the project develops I will integrate new features such as "Best Review" and "Worst Review".

The Proposed Solution

Solution

Firstly, the solution will take inspiration from all three of the existing solutions so that it provides a service that is both complex computationally but simple for the user to interact with.

The user interface will take inspiration from the simplicity of the Fakespot website. This is because the intuitiveness of their solution is unparallelled and significantly better than the other two solutions. Though the website will be easy to use, I will implement a simple help button in order to aid those who are confused or in need of help. By making the results simple to manipulate and read, a reader range of people will be able to use the solution without difficulty. Within the web page, there will be a search box and submit button so that the user can search for various different Amazon products. Additionally, the product that they search for will be displayed on the screen so that they can both analyse the product themselves as well as read the systems analysis of the product and its reviews.

Next, the inner technology of the Stanford research paper will be utilised to analyse the sentiment of all the reviews for a specific product. This is something that the other two lack. A machine learning model will be responsible to analyse all incoming reviews and output a decision on the overall rating of the product. The model will be trained by a dataset that is created by scraping random reviews from random products on Amazon.

The development of the solution will be split up into various sequential sprints. By following the agile method, features can be segmented and developed individually so that they can eventually be integrated together into one seamless service.

The Sprints

Sprint 1

Sprint 1 will consist of creating the front end. I will use React and React Bootstrap in order to create an accessible and scalable web page that others can easily use with very little instruction. It should be intuitive and easy to use whilst being laid out in a readable format. This format and design will be explored further during the sprint with sketches of possible layouts.

Sprint 2

Sprint 2 will be the setting up of the web scraping in order to create a file for the machine learning model to train with. This will be done in Python and will either use an Amazon Reviews API, a pre-existing dataset or creating my own using a python package called BeautifulSoup.

Sprint 3

Sprint 3 will be the first iteration of the sentiment analysis. I will start with the simple implementation of natural language processing by creating a set of positive, negative, amplifier and negating words and assigning a score and multiplier to them. The overall score produced from a review will determine whether the review is positive or negative and an overall score will be averaged from all the reviews for a product. This will then create a rating for the product and return it to the user through the web page.

Sprint 4

Sprint 4 will aim to implement a form of machine learning into the sentiment analysis. This will include learning about the machine learning models such as the ones implemented in the Stanford research paper as well as developing my own to produce outputs that I require.

Sprint 5

Sprint 5 will be the implementation of additional features such as “Best Review” and “Worst Review”. Some potential features are listed below.

1	Best and Worst Review.

2	Include number of reviews as a factor in analysis.
3	Include seller location and seller as a factor in analysis (more trusted means better chance of a good product).
4	Similar products displayed.
5	Similar product ratings displayed alongside.
6	Training whilst deployed — users help to fine tune the system.
7	Break up reviews into phrases and establish sentiments for those phrases.
8	Account system to save products and reviews.
9	Save already computed products which can be stored in a database and returned upon another request.

Risks

Feasibility Study

Economic Aspect

The project will not cost anything to complete. This is because the hardware and software required are already available and open source. Some cost may incur from using an AWS account, however, I do not intend on opening a free AWS account and paying to test the sentiment analysis feature.

Technical Aspect

The project is possible and can be completed. Some training and learning will be required when web scraping as currently there are two possible routes: either finding an Amazon product review API or scraping them myself. I already have experience in Django, Python and React and so the main part of the training will be learning the models to use when carrying out the sentiment analysis.

Social Aspect

The solution will be very useful to buyers as they are the main stakeholders in this product. However, businesses may dislike the product as it might dissuade buyers from buying their products if they do not have an overall positive review. Businesses, however, can then use the product in order to alter the product and take into account the various buyer reviews to improve their score and sell more stock.

Time Constraints

Creating the front end should not be too time consuming as it will be a fairly straightforward webpage. Future features can be added at a later date after the main sentiment analysis feature has been implemented. The model will have to be trained as well which may take a long time once it has been developed. Having about 4-6 months to complete the project, I will focus on completing the first three sprints and then adding the additional features when the main priority has been successfully completed.

Limitations

The first limitation may be time management. Having around 4–6 months to develop a working solution, it is essential to focus on the most important features first before developing the additional, helper features. For example, it is necessary to implement the sentiment analysis but this can be done in a number of ways. The easiest method would be to create a database of positive, negative, amplifier and negating words and create an algorithm that can analyse the appearances of these words in reviews. Being a simple solution, this may be developed quickly and soon I would be able to produce a better machine learning model to analyse the reviews. However, having the existing solution already in place, there would be no issue of delays or producing a half-finished solution as we would be extending features once the base amount is complete. The isolation of each sprint makes development like this possible as we can alter the backend or frontend in isolation to improve the solution.

Another limitation caused by time would be the time allowed for training the machine learning model and the availability of test data. It might be beneficial to find a dataset online similar to the Stanford research paper. However, their dataset was heavily biased and so this caused overfitting in their model. Alternatively, I could scrape the reviews myself and fill quotas. For example, we would create a file of 10,000 1 star reviews, 10,000 2 star reviews and so on. This way, there is no inherent bias from the number of samples. The issue here would be the time taken to generate the dataset which would take a considerable amount of time.

One issue that may arise is the actual effectiveness of the solution. As the machine learning models may be hard to research and properly implement and train, the results that the solution produces may not be accurate or reliable. The machine learning model aims to solve the issue of the existing solutions not providing an answer to if the user should buy the product or not and returning a rating.

Requirements

Software

In order to produce the solution, I will aim to work in a variety of languages. These will be Python[5] with the Django[6] framework for the backend whilst, for the frontend, Javascript in React[7] will be used with React Bootstrap[8] to produce a clean user interface that is easy to interact with. A variety of different modules will be required. BeautifulSoup[9] will be required in order to carry out the web scraping for the amazon reviews and ratings. This will incorporate bootstrap in order to make the resulting webpage easy to resize and use on different sized devices. As both the frontend and backend are web-based and are hosted locally, the system will eventually be available on all devices and web browsers that have javascript enabled.

Hardware

The hardware required is very minimal. I will be using an Apple Mac in order to develop the solution. The Mac has a 2.9 GHz processor with 8GB RAM. Essentials such as a mouse and keyboard are required too. The documentation will be stored online on Google Drive whilst the solution will be stored locally on the Mac harddrive and can be transported using another medium such as a USB flash drive.

[5] https://www.python.org
[6] https://www.djangoproject.com
[7] https://www.reactjs.org
[8] https://www.react-bootstrap.github.io
[9] https://www.pypi.org/project/beautifulsoup4/

Success Criteria

Criteria Reference	Criteria	Reason	Test Strategy
1	**The User Interface**		
1.1	Display a URL search box and submit button.	Allow the user to search for a specific Amazon product.	There should be a text input box with a submit button.
1.2	Return the first image of the product and all relevant information.	Allow the user to read a summary of the product.	An image should appear as well as a card holding the product information.
1.3	Display a result.	The user must be able to see the results of the analysis of the reviews.	A calculated rating should appear as well as the confidence that the product is positive or negatively rated.
1.4	Display Amazon results	The user should be able to view the data taken from Amazon about the product.	The extra information about a product should be visible. For example, dimensions.
1.5	Restart search.	The user must be allowed to create multiple analyses for different products.	A restart button should be included or the search bar and submit button must always appear at the top of the page.
2	**Web Scraping**		
2.1	Take a URL from the user.	The scraper must be able to take all of the reviews from the product provided by the user.	The scraper will be able to find the page referenced by the user and eventually extract the relevant details.
2.2	Access the reviews.	The reviews must be taken from the Amazon site and then analysed by the software.	Sanitise the incoming reviews and store them locally. They can then be output to

			check existence.
2.3	Dataset creation.	In order to train and test the machine learning model, it will have to analyse thousands of reviews to gain an overall view on what is positive and negative.	One file will be created that includes a mixture of 50,000 reviews of different stars. These will then be fed into the model to train and test it.
3	**Sentiment Analysis**		
3.1	Analyse an individual review for the words in it.	The system must be able to distinguish between good and bad reviews.	A score is returned for the review based on the occurring words within it.
3.2	Incoming reviews must be sanitised and prepared before analysis.	Words may not match if there is capitalisation or if they are misspelt.	The sanitisation function will be tested to ensure it is effectively working before use.
3.3	The machine learning model is trained	The model must be trained in order to effectively analyse reviews to an acceptable level of accuracy and confidence	After training, the system will be tested and it will output an accuracy. Once the accuracy is acceptable, it will be ready for use.
3.4	The machine learning model is tested	The model must then be tested to ensure acceptable accuracy and error.	An accuracy of 80% and above will be ideal but this will vary depending on the dataset created.
4	**Additional Features**		
4.1	Best and Worst Reviews are displayed.	These are displayed so that the user can gain more information about the product they may buy.	Create a fake product with fake reviews and submit that product. The most positive and negative reviews will be output and then compared to see whether they are

			working.
4.2	Number of reviews included in analysis.	More reviews normally highlights that the product is popular and better value	Create a coefficient that incorporates the number of reviews into the overall outcome. Test this with very popular products and less popular ones.
4.3	Display similar products.	The user may look at an untrusted or unreliable product and so providing other ones may be beneficial to the user.	The system can search Amazon for the product tags and return the three most popular ones. Alternatively, the first page could all be analysed and the top three returned but this will add to computation time and increase delay.
4.4	Model is trained whilst in use.	Users can train the model whilst it is being used in order to improve accuracy.	Every request made by the user can then add to the training set and the next request can therefore test the model. As a result, the accuracy should steadily improve over time.

Sprint 1

Design

Sprint 1 is all about the design of the frontend; the webpage that the user can interact with in order to use the service. The normal flow of development would be creating the backend and then the frontend so that the API is created first and the webpage designed around the results returned from it. However, the issue that I face is that machine learning is a very complex topic and the nature of the end solution will take weeks of research and testing before a tangible service can be made. In order to accommodate for this, I will take a user-centric approach and design the API and the webpage simultaneously such that the frontend template is made initially and can be altered in the future if changes are required. By developing the frontend first, I am allowing myself time to fully research the methods to analyse the sentiments of the reviews. Additionally, the user-centric approach is justified as ultimately the service will be used by average people and their technological aptitude is not assumed. Therefore, it is necessary to effectively plan the layout so that it is easy for all to use.

In this sprint I will set up the front end development environment and create a webpage that users will interact with when using the service. This sprint can be broken down into developing and creating different items according to how React deals with states and props and passes them up through parents. All elements can be broken up into different sections which then allows bootstrap to be easily implemented in order to create a resizable web app.

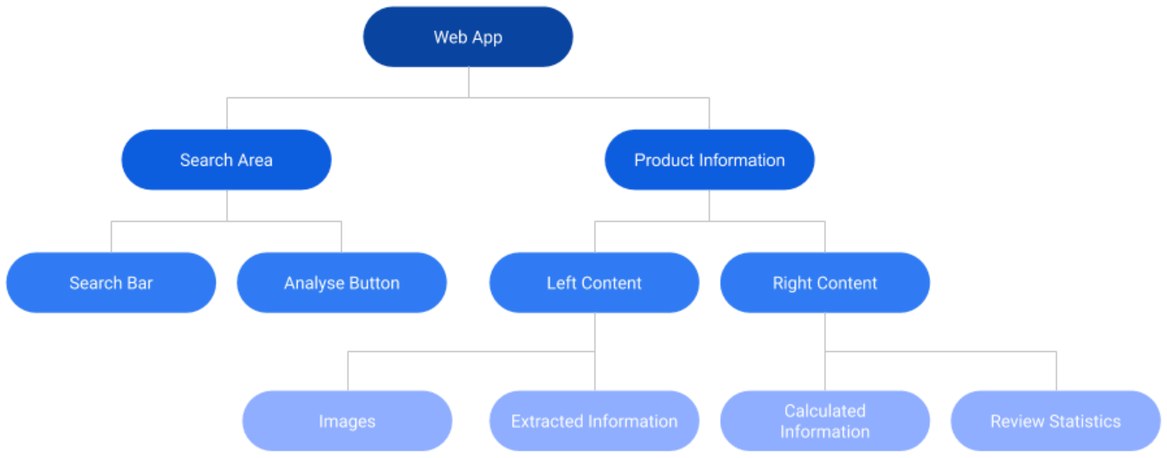


Figure 4. Class Hierarchy

The web app is the highest class and contains all child classes. For example, the URL taken from the search bar gets passed up through the hierarchy and used in the web scraper where the information taken is displayed in the Product Information classes.

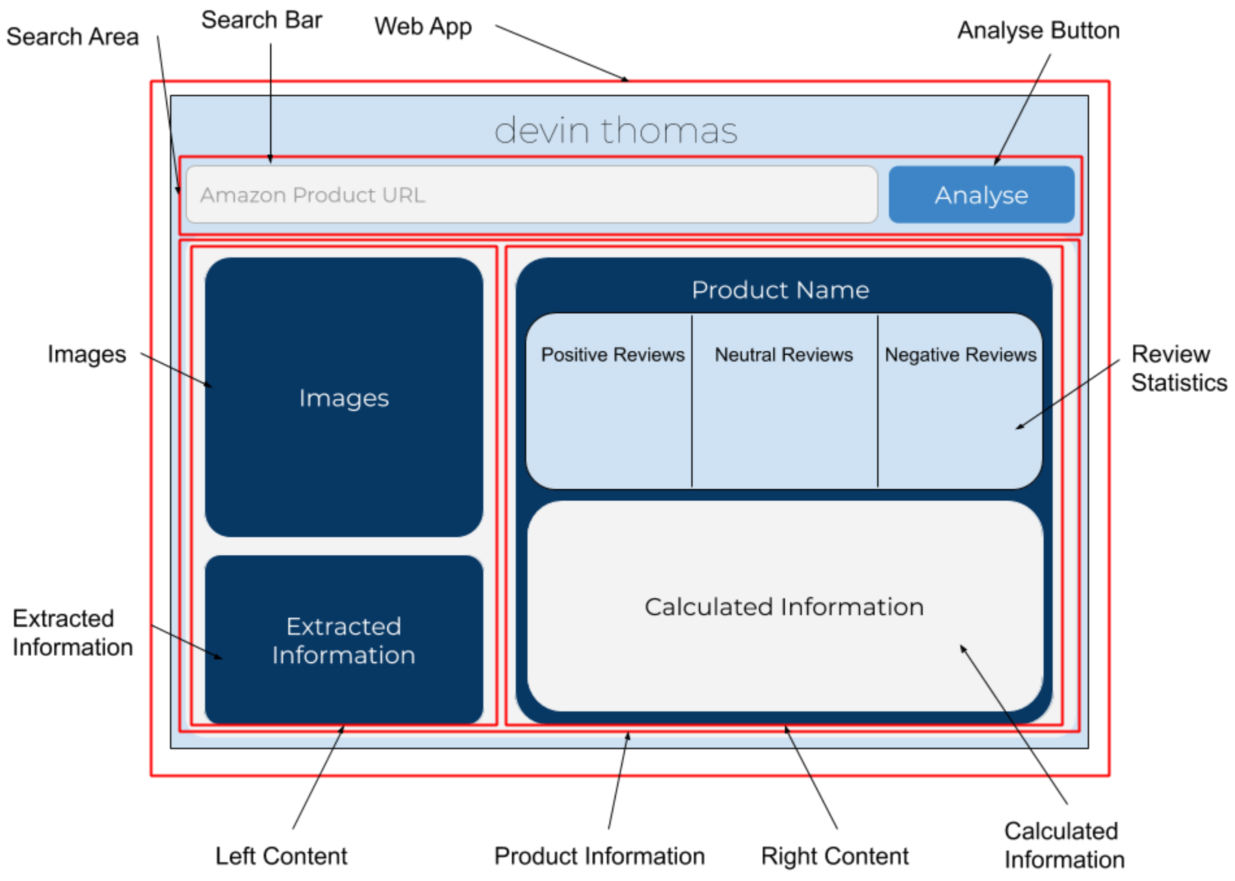


Figure 5. Web page design

Figure 5. represents a rough template of what the final service will look like. Inspiration has been taken from Fakespot, one of the previous solutions researched in order to make the site more user friendly and easy to interact with. Each section is broken down into its smaller constituent parts. These are then broken into smaller parts to create the hierarchy shown in Figure 4. By breaking down the components into individual sections we have cut down the amount of work to do later. Firstly, we can create individual parent and child classes to be used in React. This will allow states to be passed back through the hierarchy. Additionally, React Bootstrap will also benefit from this as by splitting it up into different parts, we can create a site that automatically resizes and reformats so that the readability remains high when used on different devices such as mobile phones, tablets and laptops.

Creating the Base React App

Firstly, we need to create the base React app that will be built upon. We can create this by installing Node and npm and using the following commands in Terminal:

```
npx create-react-app reviews_ui
cd reviews_ui
npm start
```

The first command creates an empty react app and names it “reviews_ui”. The next command will then change the active directory to “reviews_ui” and finally the last command will run the server locally. Now, looking into the reviews_ui directory, we will see a set of files to be edited for our React frontend.

To include different React Bootstrap Components into the site, we will have to firstly import the component type and then reference them using tags as such:

```
import Component from ‘react-bootstrap/Component’

<Component>
</Component>
```

Usability

When discussing usability, we must take into account the 5E’s: Effective, Efficient, Engaging, Error Tolerant, and Easy to Learn. Therefore, as we are approaching the problem with a user-centric view, it is necessary to focus on these aspects.

Effective

In order to create an effective system, it must achieve the users goals with completeness and accuracy. In this case, we are looking at whether the user will be able to gain the results of the analysis on their product. Due to the very simple steps in finding these results, we can be sure that the system will be effective. Results will be displayed logically and coherently so that the user effectively receives the answers to their query.

Efficient

We can guarantee that the resulting web page will be efficient due to the very simple steps in order to get the results of a product. The user will first have to input the URL of a product that they would like to analyse and then press the ‘Analyse’ button. These two simple instructions ensure that the solution is efficient as the user does not have to carry out many steps to get to their desired result. They are efficiently abstracted from the inner workings of the system and as a result can use the product effectively and swiftly.

Engaging

I have taken inspiration from more modern and easy to use websites in order to engage the user. Information is clearly separated into different sections and allows the user to visualise which parts are extracted from Amazon and which are calculated by the system. Overall, the system should not be difficult to use and therefore remain engaging. The use of colour and a simple, modern font will ensure that the user is visually stimulated and therefore cause them to be satisfied with the layout of the website.

Error Tolerant

In order to produce an error tolerant system we have to analyse the ways that there could be errors. One way that an error could come is whether the amazon product being searched for exists or not. If the user inputs a URL that doesn’t exist, the system must be able to output an error message to tell them that there is an issue. We can do this using alert boxes in React. Another error that might occur is that the computation of the product takes too long and the user mistakes the delay for a fault with the system. This can be fixed with a loading bar.

Easy to Learn

As previously talked about within the efficiency section, the system will be very easy and intuitive to use. Users will only be required to complete two steps: input the URL and press analyse. From this point the system will later return the results of its analysis to the user. As there is very minimal user interaction, the resulting webpage should be simple to use.

Key Variables, Data Structures and Classes

React Bootstrap Components

These are the components that will be necessary when creating the webpage. They will be imported using the statement already highlighted earlier on.

Component	Use	React Use
Container	Used to contain different components.	`<Container>...</Container>`
Row	Used to separate different areas into rows. This identifies which parts stack when the window changes size.	`<Row>...</Row>`
Column	Used to split rows into columns. Rows are split into 12 sections and each column can be defined to take up a certain number of these horizontal sections: it dictates the ratio of the columns.	`<Col sm={4}>...</Col>`
Alert	An Alert will be required for when the user creates an error. The error will be displayed as an alert in order to effectively communicate back with the user.	`<Alert variant="danger">...</Alert>`
Buttons	A Button is required for the 'Analyse' button. Additionally, other buttons might be used for example to take the user to the Amazon page of the product they are looking at.	`<Button variant="Primary">...</Button>`

Cards	Cards will be required to display the information that is extracted and calculated.	`<Card>`
		`    <Card.Body>`
		`        <Card.Title>...</Card.Title>`
		`        <Card.Text>...</Card.Text>`
		`    </Card.Body>`
		`</Card>`

Carousel	A Carousel will be necessary to display the	`<Carousel>`
		`    <Carousel.Item>`

	different pictures of the product that the user is viewing	`        <img src=""/>`
		`    </Carousel.Item>`
		`    <Carousel.Caption>`
		`        <h3>...</h3>`
		`    </Carousel.Caption>`
		`</Carousel>`

Forms	A Form is required in order to allow the user to input the URL of the product they are analysing.	`<Form>`
		`    <Form.Group>`
		`        <Form.Label>...</Form.Label>`
		`        <Form.Control/>`
		`    </Form.Group>`
		`</Form>`

React Classes

In order to create the different sections in React, we will be creating classes for each component that will return and render React Bootstrap components. Classes will be created such that the props and states of child components can be passed through the hierarchy and used by other components.

We can create a new section such as the Search Area like this:

```
class SearchArea extends React.Component {
    constructor(props) {
        super(props);
    }

    render() {
    }
}
```

Now that the `SearchArea` class is created, we can then reference it and use it within a parent class as such:

```
class App extends React.Component {
    render() {
        <SearchArea />
    }
}
export default App;
```

Now, when the server is launched, we will be able to see the Search Area and whatever it contains because it has been referenced already by the parent class `App` and this is the class that is initialised upon the launch.

Key Variables

There are a number of key variables that will be used throughout the entire process but there are only a few concerning the front end. These are listed in the table below.

Variable	**Data Type**	**Use**
Product URL	String	The URL of the product being analysed must be taken from the user in order to extract the reviews later on.
Extracted Rating	Float	The extracted Amazon rating is part of the extracted information and is therefore necessary to save and display.
Calculated Rating	Float	The calculated rating is part of the calculated information and is therefore necessary to save and display.
Extracted Text	String	The various pieces of extracted data such as the name, dimensions and product description can all be displayed and therefore are required to be passed as variables.
Calculated Text	String	The best and worst reviews must also be saved and displayed to the user.

Validation

As the interaction between the user and the system is very minimal there is not very much to validate. The only input from the user is the product URL. Therefore, it is necessary that we validate that this product exists before proceeding with the analysing. This validation exists between the submission of the URL and the scraping of the products reviews.

Testing

To test this sprint, I will develop the front end in conjunction with observing the layout. As it is being built up from the base app, we can add code and test that features are added by observing the resulting web page that is produced. The whole of this sprint will be tested against section 1 (User Interface) of the testing table depicted in the Analysis Section of this document.

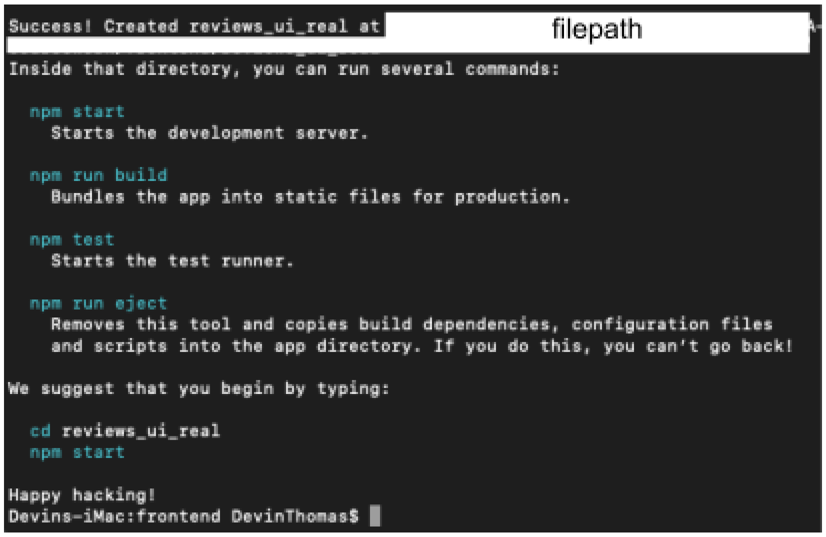
Post Development

As the frontend is standalone, there is not much data to be used in the post development phase. Everything completed within this sprint is isolated from the rest of the system. The only piece of data that must be considered is the how the URL will be transferred from the frontend to the backend. Additionally, it is necessary to plan out how the JSON API response from the backend will look so that during the development, we can plan to extract the data from this JSON response and display it to the user. As a result, we must also consider how data is transferred from the backend to the frontend.

Development

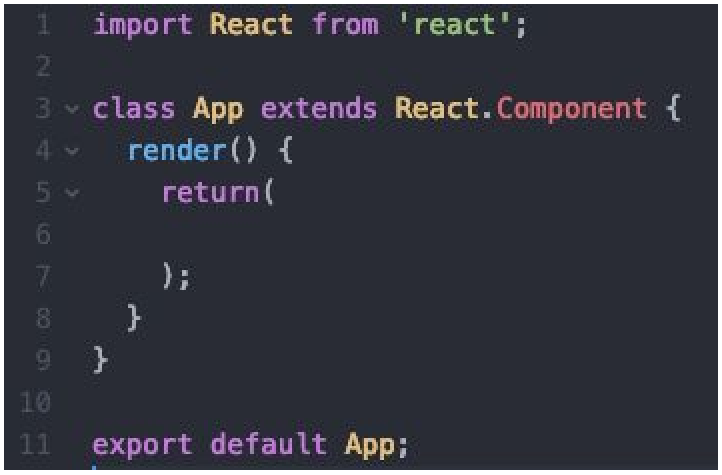
Setup

In order to create the frontend we need to set up the react app as shown below .



```
Success! Created reviews_ui_real at filepath
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd reviews_ui_real
  npm start

Happy hacking!
Devins-iMac:frontend DevinThomas$
```

npx has created a new React app for us called reviews_ui within the specified directory. This has created a range of files. We are currently concerned with two of them: App.js and App.cs. As we will be using React-Bootstrap, the entire contents of App.cs can be deleted and we can reduce the code setup.



```
import React from 'react';

class App extends React.Component {
  render() {
    return(

    );
  }
}

export default App;
```

This code is the basis of what I will build upon. The App class is the highest class in the hierarchy. This is what is initialised as seen at the bottom with export default App;. From this simple base, we can build up the next few classes.

In order to consume React-Bootstrap, we must first install react-bootstrap with:

```
npm install react-bootstrap bootstrap
```

After having done this, we can then create a link to it within ‘index.html’ by including the following link:

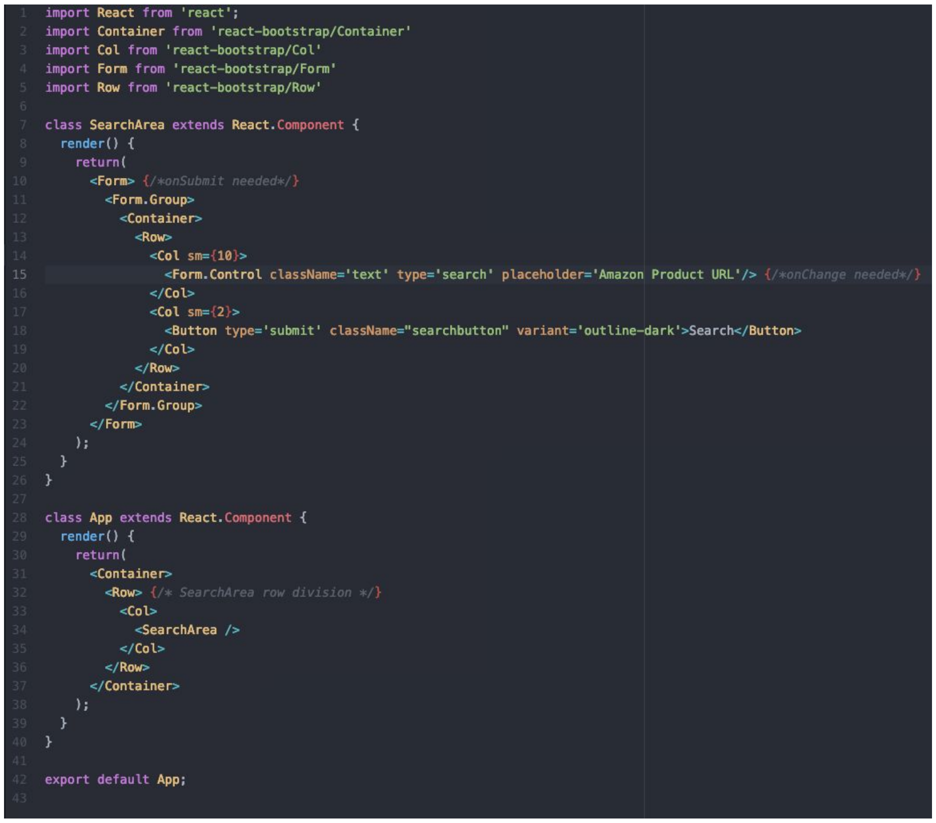
```
<link
  rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
  crossorigin="anonymous"
/>
```

Now, by creating a component within App.js, we can first import a react component and then use it.

Search Area

In order to create the search area, we will split it into two sections: the search bar and the search button.

Firstly, we will create the `SearchArea` class and then reference it within the `App` class.

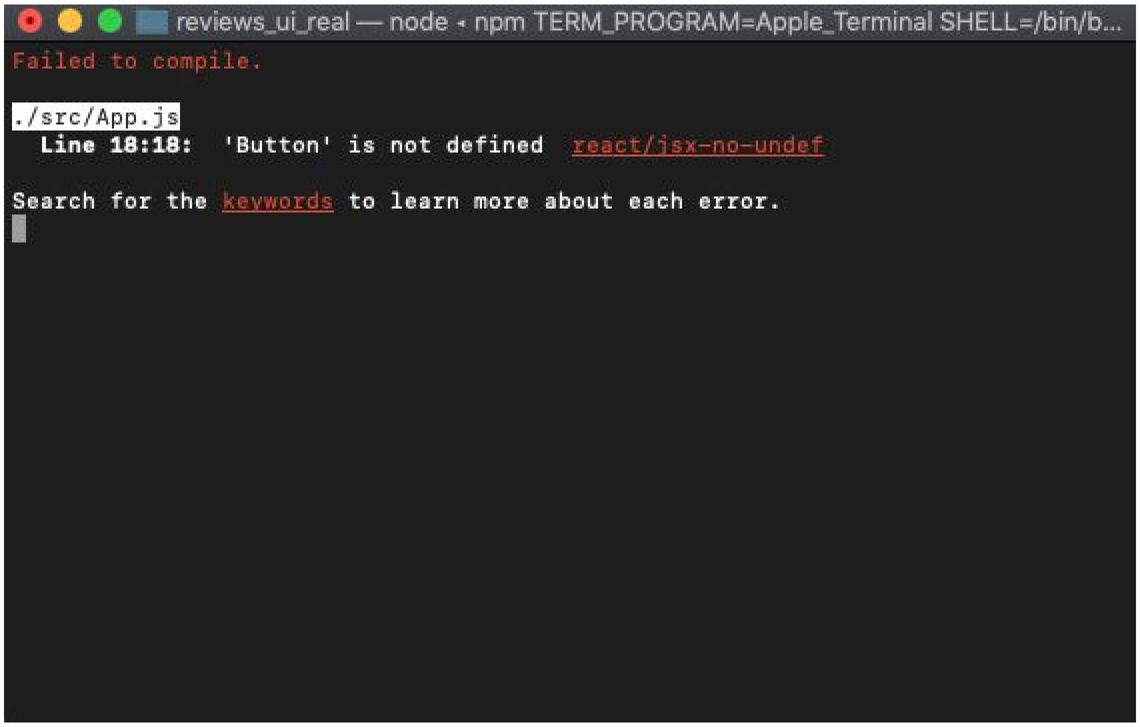


```
import React from 'react';
import Container from 'react-bootstrap/Container'
import Col from 'react-bootstrap/Col'
import Form from 'react-bootstrap/Form'
import Row from 'react-bootstrap/Row'

class SearchArea extends React.Component {
  render() {
    return(
      <Form> {/*onSubmit needed*/}
        <Form.Group>
          <Container>
            <Row>
              <Col sm={10}>
                <Form.Control className='text' type='search' placeholder='Amazon Product URL'/> {/*onChange needed*/}
              </Col>
              <Col sm={2}>
                <Button type='submit' className="searchbutton" variant='outline-dark'>Search</Button>
              </Col>
            </Row>
          </Container>
        </Form.Group>
      </Form>
    );
  }
}

class App extends React.Component {
  render() {
    return(
      <Container>
        <Row> {/* SearchArea row division */}
          <Col>
            <SearchArea />
          </Col>
        </Row>
      </Container>
    );
  }
}

export default App;
```

Within this update of App.js, we have imported new necessary react components: the container, column and row. As seen within the code, the `App` class now contains an instance of the `SearchArea` contents.

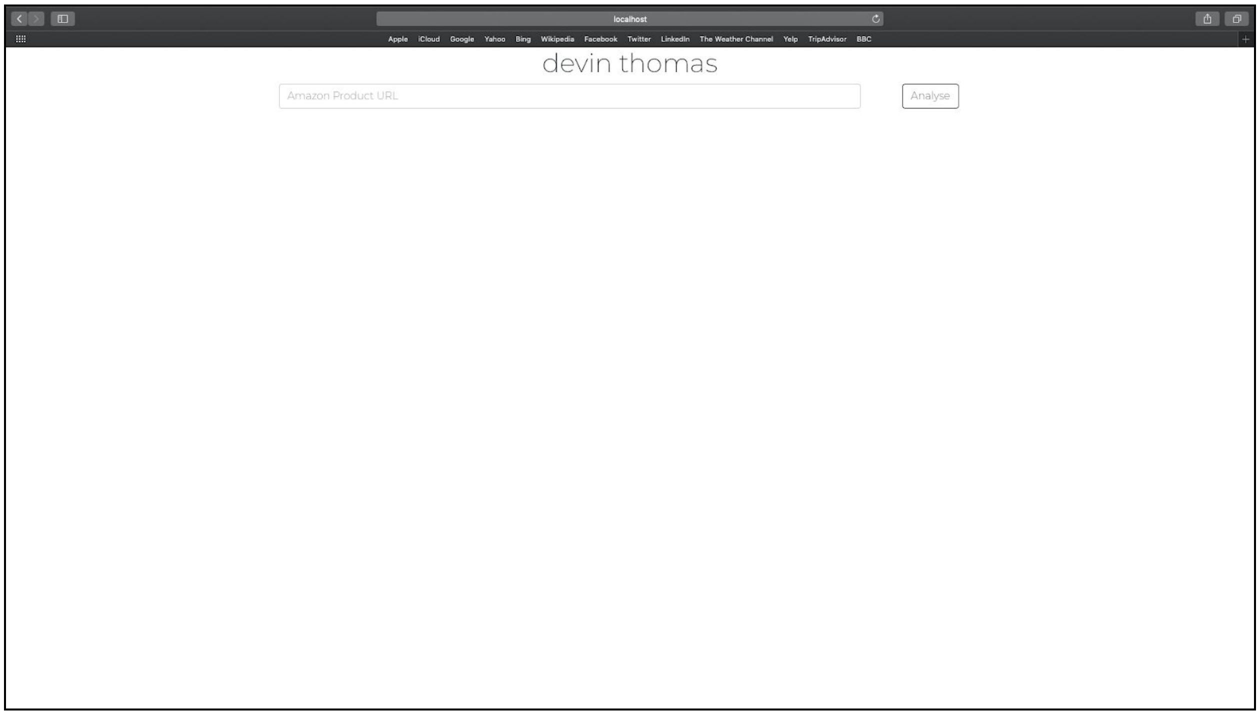
To create the Search Bar and Analyse Button, we need to create a Form. This will allow the user to input a URL and press the Analyse Button. Once the button has been pressed, the future `onSubmit` will be used to extract the URL and therefore find the product from Amazon.

Running this code however, produces an error. The React app fails to compile as there is a missing component: the Button component.


reviews_ui_real — node ◂ npm TERM_PROGRAM=Apple_Terminal SHELL=/bin/b...

```
Failed to compile.

./src/App.js
  Line 18:18:  'Button' is not defined  react/jsx-no-undef

Search for the keywords to learn more about each error.
```

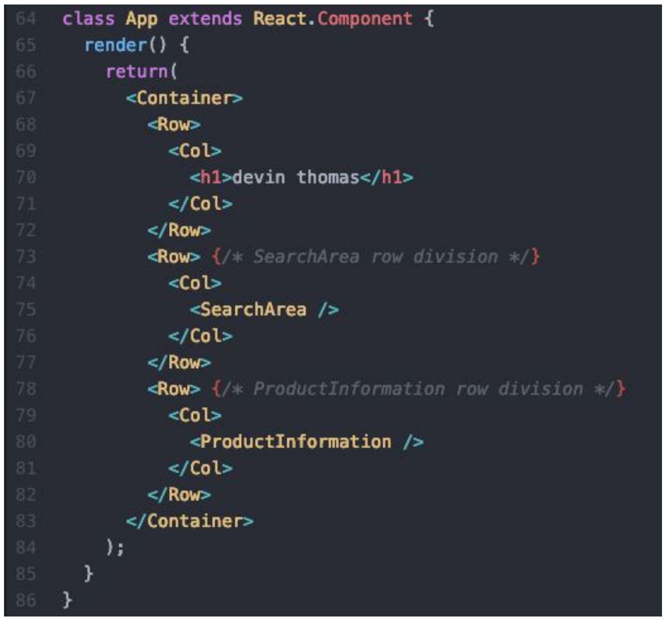
In order to fix this error, we have to import the Button component within App.js by adding the line `import Button from 'react-bootstrap/Button'` After doing so, we are greeted with this page below.


localhost
Apple iCloud Google Yahoo Bing Wikipedia Facebook Twitter LinkedIn The Weather Channel Yelp TripAdvisor BBC
devin thomas
Amazon Product URL
Analyse

Currently, there is no form of validation included within the Search bar input. This is because the validation will all be completed by the backend as only valid URLs will be accessible. As the frontend does not access the URL, it cannot know whether it is valid or invalid.

Product Information

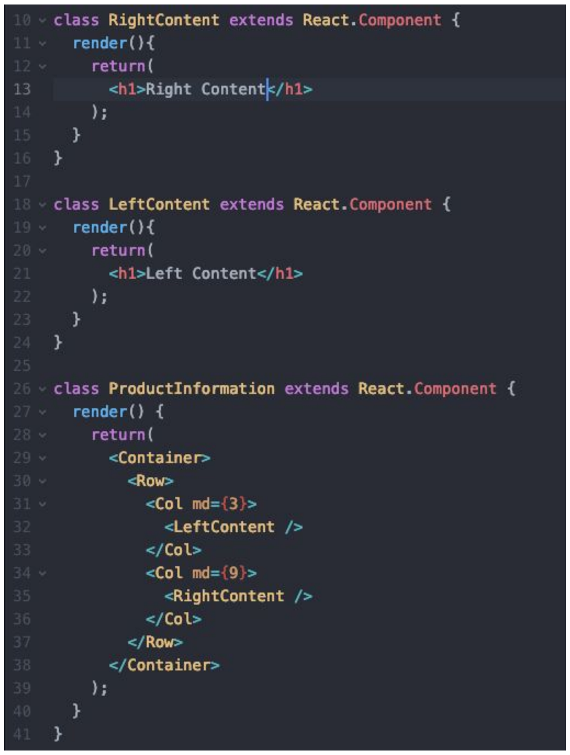
Below the SearchArea, we will introduce the product information.



```
class App extends React.Component {
  render() {
    return(
      <Container>
        <Row>
          <Col>
            <h1>devin thomas</h1>
          </Col>
        </Row>
        <Row> {/* SearchArea row division */}
          <Col>
            <SearchArea />
          </Col>
        </Row>
        <Row> {/* ProductInformation row division */}
          <Col>
            <ProductInformation />
          </Col>
        </Row>
      </Container>
    );
  }
}
```

Devin Thomas

The `App` class has now once again been updated in order to accommodate for the new class called ProductInformation. This class subsequently references another two child classes: `LeftContent` and `RightContent`.

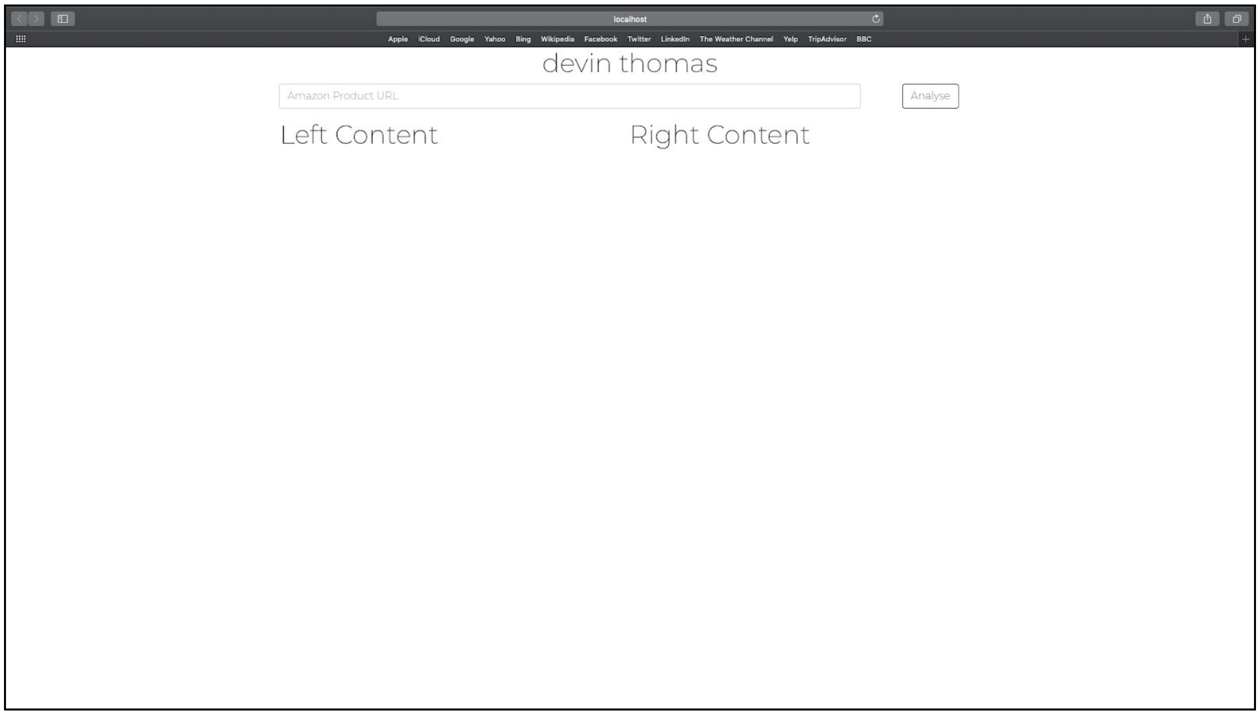


```
class RightContent extends React.Component {
  render(){
    return(
      <h1>Right Content</h1>
    );
  }
}

class LeftContent extends React.Component {
  render(){
    return(
      <h1>Left Content</h1>
    );
  }
}

class ProductInformation extends React.Component {
  render() {
    return(
      <Container>
        <Row>
          <Col md={3}>
            <LeftContent />
          </Col>
          <Col md={9}>
            <RightContent />
          </Col>
        </Row>
      </Container>
    );
  }
}
```

These are the new classes that have been added. As can be seen, there are two referenced child classes within the `ProductInformation` class. This is in order to split the content into two separate sections as shown in Figure 5.

The render of `ProductInformation` directly calls on `LeftContent` and `RightContent` and so these classes are also rendered.

After reloading the web page now, we can see the new separation of the left and right contents.

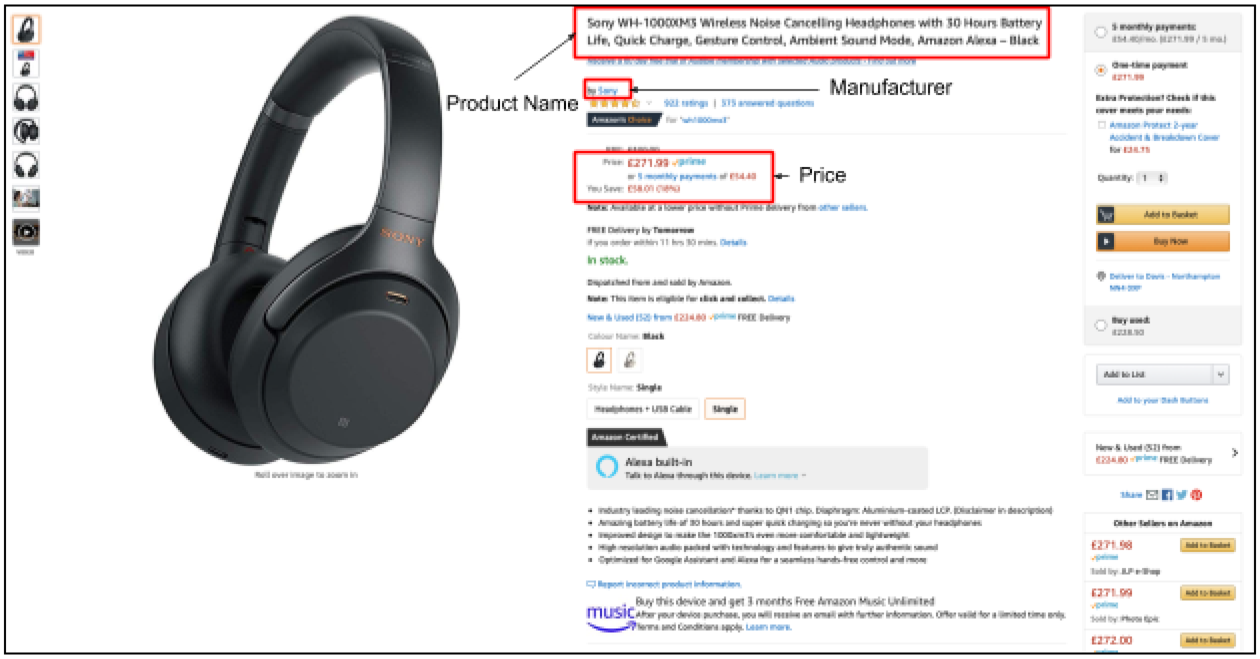

localhost
Apple iCloud Google Yahoo Bing Wikipedia Facebook Twitter LinkedIn The Weather Channel Yelp TripAdvisor BBC
devin thomas
Amazon Product URL
Analyse
Left Content
Right Content

Inside Left Content

Within the `LeftContent` Column, we are using two separate components, the Images and the Card for the Extracted Information.

Extracted Information

When designing the layout for this section, we must consider what segments of information are the most critical to extract and display on the page.



Sony WH-1000XM3 Wireless Noise Cancelling Headphones with 30 Hours Battery Life, Quick Charge, Gesture Control, Ambient Sound Mode, Amazon Alexa – Black

by Sony

922 ratings | 375 answered questions

Amazon's Choice for "wh1000xm3"

Price: £271.99 ✓prime
or 5 monthly payments of £54.40
You Save: £58.01 (18%)

Note: Available at a lower price without Prime delivery from other sellers.

FREE Delivery by Tomorrow
If you order within 11 hrs 30 mins. Details

In stock.

Dispatched from and sold by Amazon.

Note: This item is eligible for click and collect. Details

New & Used (52) from £224.80 ✓prime FREE Delivery

Colour Name: Black

Style Name: Single

Headphones + USB Cable | Single

Amazon Certified

Alexa built-in
Talk to Alexa through this device. Learn more

- Industry leading noise cancellation* thanks to QN1 chip. Diaphragm: Aluminium-coated LCP. (Disclaimer in description)
- Amazing battery life of 30 hours and super quick charging so you're never without your headphones
- Improved design to make the 1000xm3's even more comfortable and lightweight
- High resolution audio packed with technology and features to give truly authentic sound
- Optimized for Google Assistant and Alexa for a seamless hands-free control and more

Report incorrect product information.

Buy this device and get 3 months Free Amazon Music Unlimited
After your device purchase, you will receive an email with further information. Offer valid for a limited time only. Terms and Conditions apply. Learn more.

5 monthly payments: £54.40/mo. (£271.99 / 5 mo.)

One-time payment: £271.99

Extra Protection? Check if this cover meets your needs:
Amazon Protect 2-year Accident & Breakdown Cover for £26.75

Quantity: 1

Deliver to Davis - Northampton NN4 0DP

Buy used: £228.50

Add to your Dash Buttons

New & Used (52) from £224.80 ✓prime FREE Delivery

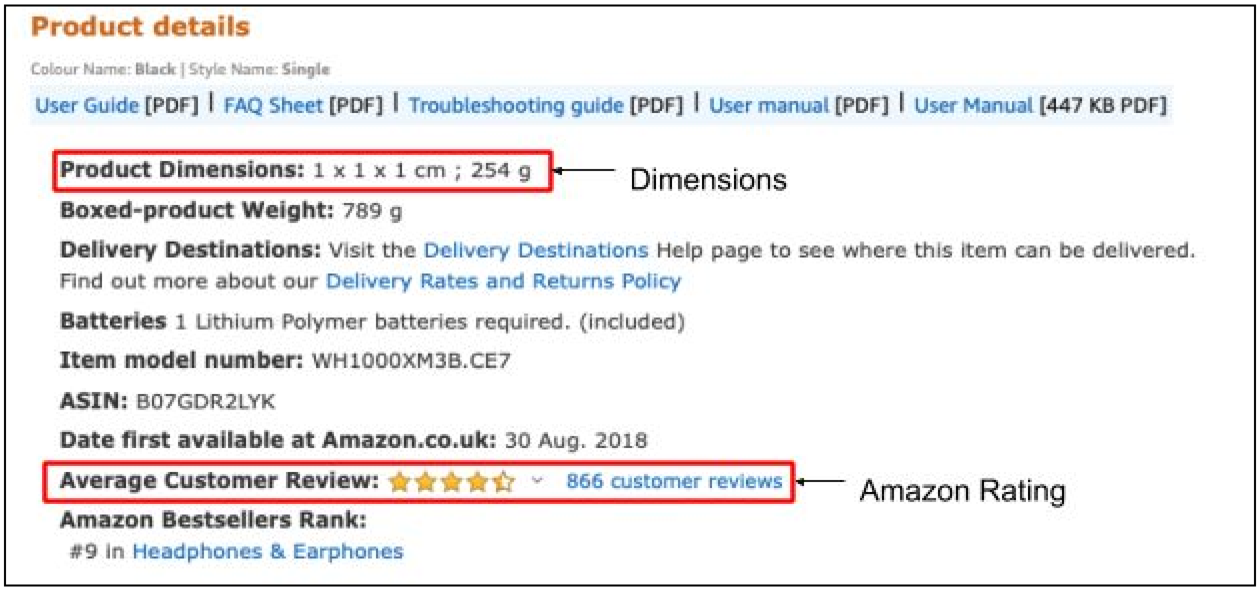
Other Sellers on Amazon

£271.98 ✓prime Sold by: JLP e-Shop

£271.99 ✓prime Sold by: Photo Epic

£272.00

Figure 6. An example Amazon page



Product details

Colour Name: Black | Style Name: Single

User Guide [PDF] | FAQ Sheet [PDF] | Troubleshooting guide [PDF] | User manual [PDF] | User Manual [447 KB PDF]

Product Dimensions: 1 x 1 x 1 cm ; 254 g

Boxed-product Weight: 789 g

Delivery Destinations: Visit the Delivery Destinations Help page to see where this item can be delivered. Find out more about our Delivery Rates and Returns Policy

Batteries 1 Lithium Polymer batteries required. (included)

Item model number: WH1000XM3B.CE7

ASIN: B07GDR2LYK

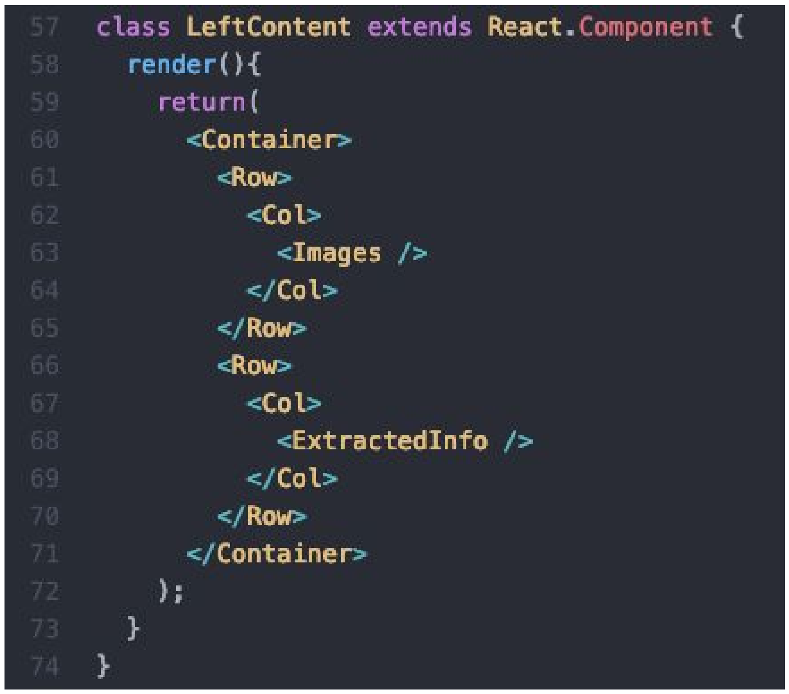
Date first available at Amazon.co.uk: 30 Aug. 2018

Average Customer Review: ★★★★☆ 866 customer reviews

Amazon Bestsellers Rank:
#9 in Headphones & Earphones

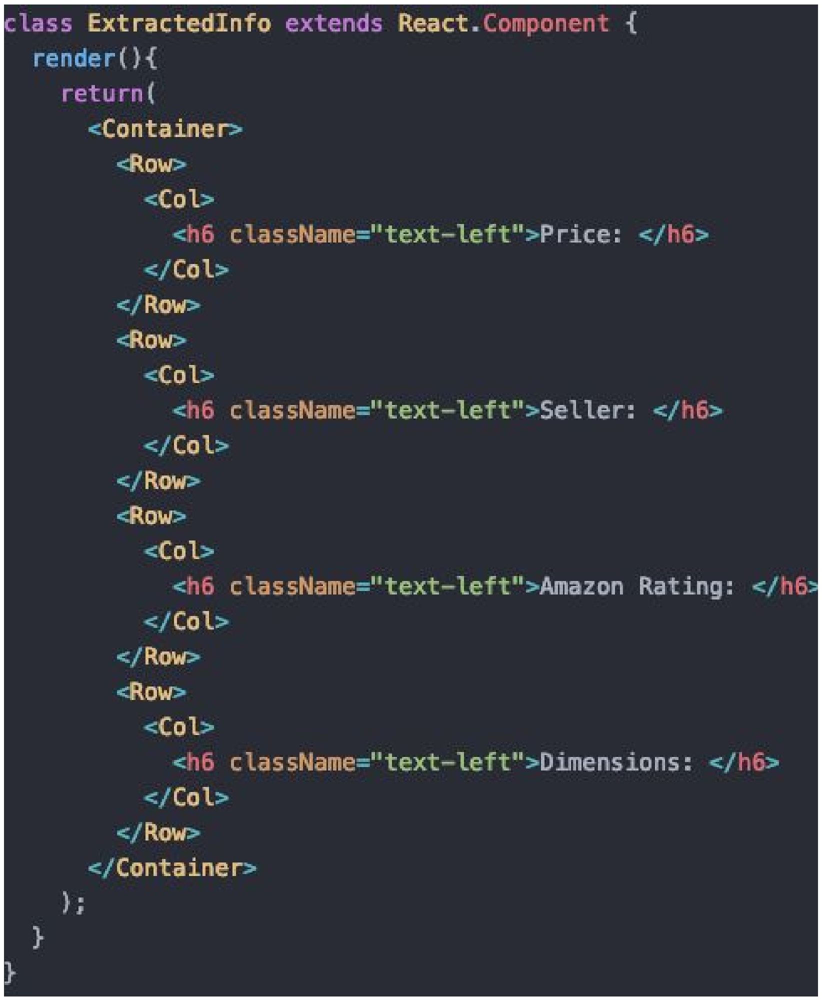
Figure 7. Product details given by Amazon

Looking at an example of a pair of headphones, we can identify the most important information about the product. It is necessary to extract the name of the product, however this will not be displayed in the extracted information section in order to maintain the design; instead, it will be displayed as the title of the Right Content. The name of the seller is also quite important. This is so that the user can view who is selling the item and determine whether it is authentic. Next, we must display the price of the product so that the user can find out the price quickly and efficiently without having to refer back to the Amazon page. Additionally, it might be useful to include the product dimensions. This is so that the user can see the size of the product from this webpage. By doing so, they can compare different products where size is a necessary variable in deciding which product to buy. Finally, we must look at the rating provided by Amazon. This is so that the user can see what other people think of the product in a raw and concise form. This is to provide the user with information about what other people think of the product without delivering all of the written reviews.



```
class LeftContent extends React.Component {
  render(){
    return(
      <Container>
        <Row>
          <Col>
            <Images />
          </Col>
        </Row>
        <Row>
          <Col>
            <ExtractedInfo />
          </Col>
        </Row>
      </Container>
    );
  }
}
```

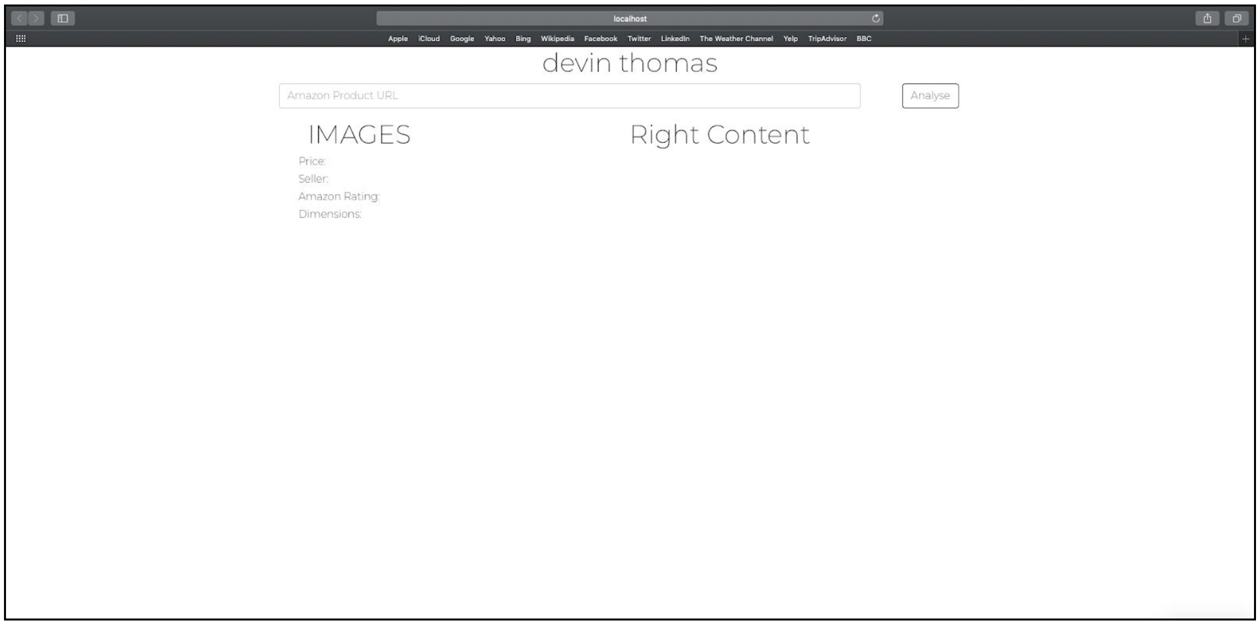
By editing the `LeftContent` class, we can include two new child classes called `Images` and `ExtractedInfo`. We then create the `ExtractedInfo` class as shown below



```
class ExtractedInfo extends React.Component {
  render(){
    return(
      <Container>
        <Row>
          <Col>
            <h6 className="text-left">Price: </h6>
          </Col>
        </Row>
        <Row>
          <Col>
            <h6 className="text-left">Seller: </h6>
          </Col>
        </Row>
        <Row>
          <Col>
            <h6 className="text-left">Amazon Rating: </h6>
          </Col>
        </Row>
        <Row>
          <Col>
            <h6 className="text-left">Dimensions: </h6>
          </Col>
        </Row>
      </Container>
    );
  }
}
```

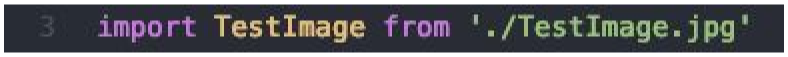
This is the `ExtractedInfo` class. Within it, we can see the different rows that are created for the four pieces of information deemed important. Once they are extracted, more code will be added in order to interpret the JSON response and display it on the screen.

When reloading the page now, we can see this page. The design is beginning to look more like Figure 5.


localhost
Apple iCloud Google Yahoo Bing Wikipedia Facebook Twitter LinkedIn The Weather Channel Yelp TripAdvisor BBC
devin thomas
Amazon Product URL
Analyse
IMAGES
Right Content
Price:
Seller:
Amazon Rating:
Dimensions:

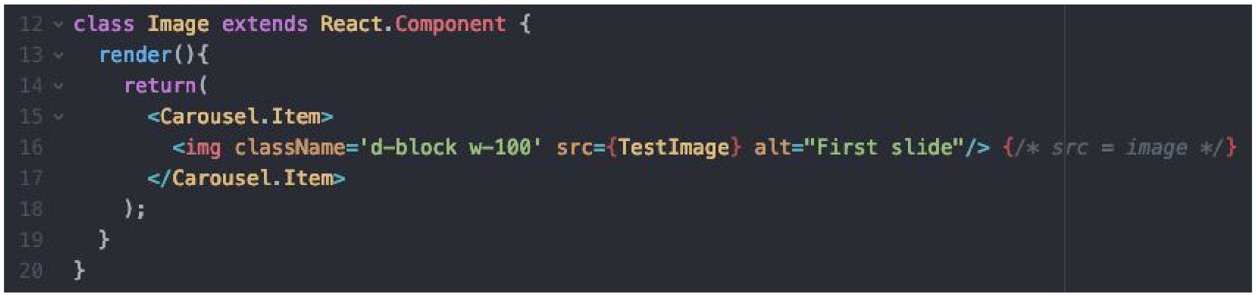
Images

In order to display the images of the product, we need to create a carousel. This carousel will be used to cycle through all the images of the product that Amazon has. In Amazon, they are displayed along the left side of the screen, however, we will create an auto-cycling carousel that will periodically change the image as well as allow the user to change it themselves.

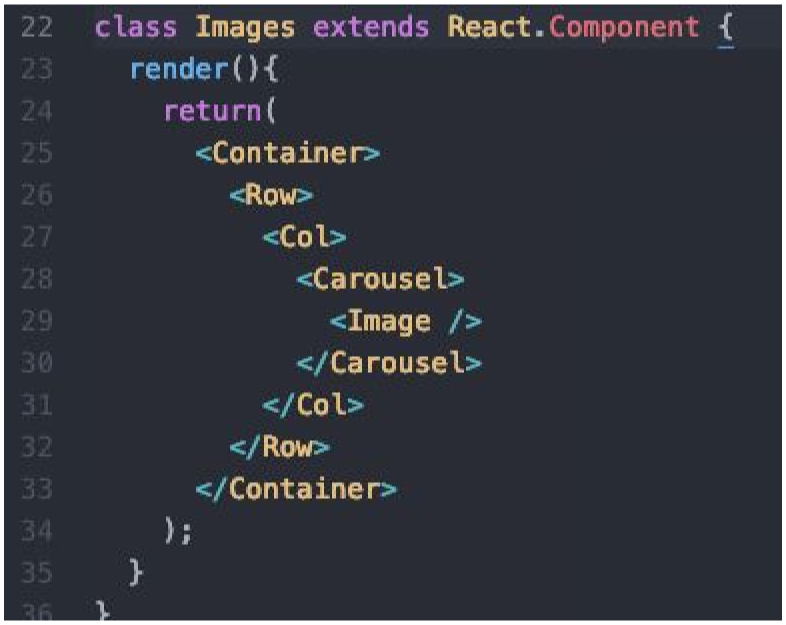


```
import TestImage from './TestImage.jpg'
```

This line has been added to import the image being used to test the Carousel



```
class Image extends React.Component {
  render(){
    return(
      <Carousel.Item>
        <img className='d-block w-100' src={TestImage} alt="First slide"/> {/* src = image */}
      </Carousel.Item>
    );
  }
}
```

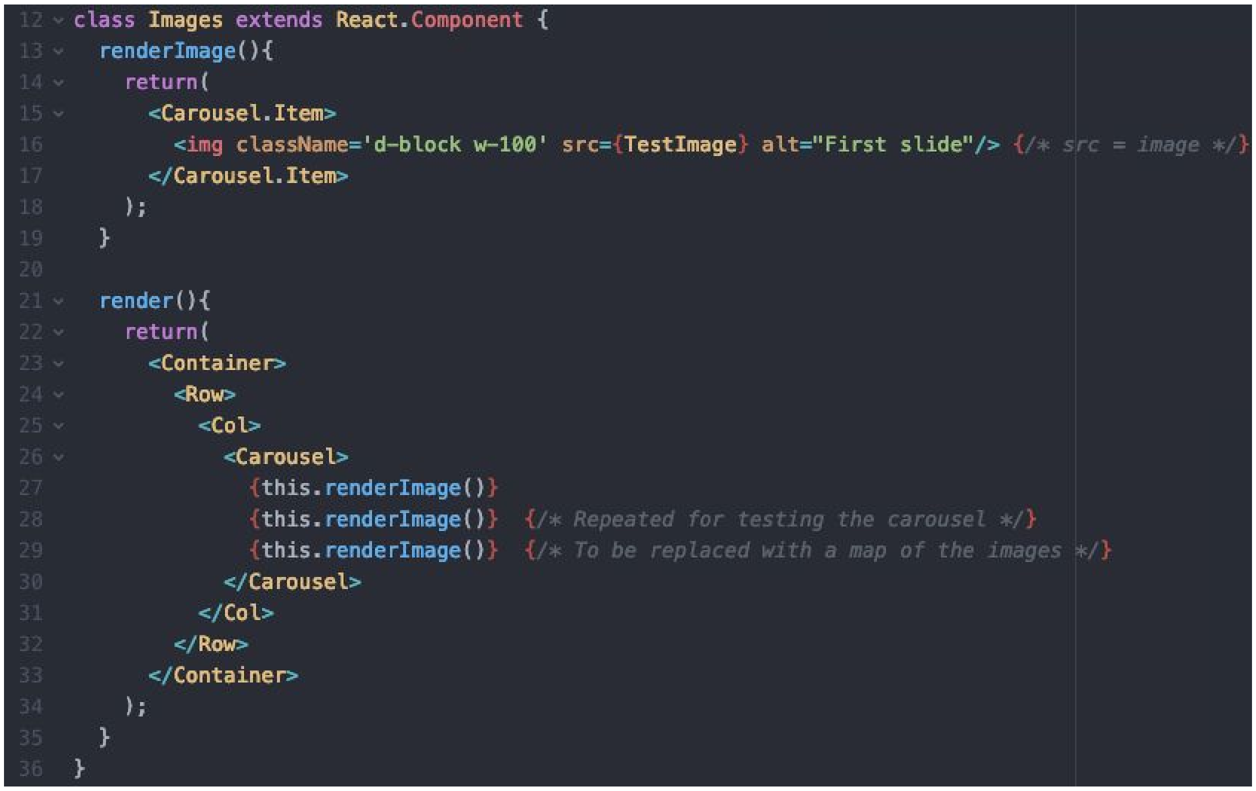


```
class Images extends React.Component {
  render(){
    return(
      <Container>
        <Row>
          <Col>
            <Carousel>
              <Image />
            </Carousel>
          </Col>
        </Row>
      </Container>
    );
  }
}
```

We have now created another class called `Image`. We have done this so that each image can be individually rendered within the Carousel.

However, when reloading the page, we do not see an image within the Carousel. The error here stems from the fact that React cannot input a Carousel Item from another render. Additionally, we will encounter issues when multiple images are presented as the code will not be able to pass the props forward. It will be much more effective to use another method.

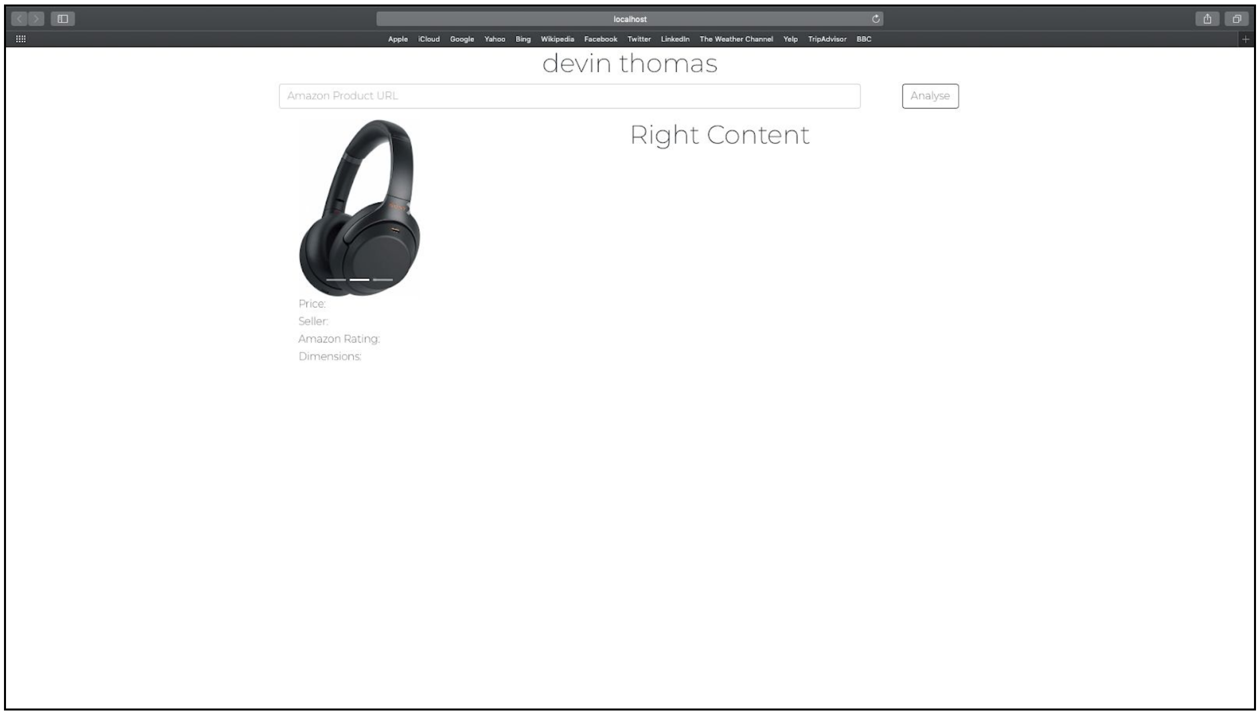
The number of images per product changes from product to product but we still want to display every image available. As a result, we are forced to create a function called `Images` that will be called by a map function. By doing this, we can render every image and add it to the Carousel.



```
class Images extends React.Component {
  renderImage(){
    return(
      <Carousel.Item>
        <img className='d-block w-100' src={TestImage} alt="First slide"/> {/* src = image */}
      </Carousel.Item>
    );
  }

  render(){
    return(
      <Container>
        <Row>
          <Col>
            <Carousel>
              {this.renderImage()}
              {this.renderImage()}  {/* Repeated for testing the carousel */}
              {this.renderImage()}  {/* To be replaced with a map of the images */}
            </Carousel>
          </Col>
        </Row>
      </Container>
    );
  }
}
```

The code for rendering the Images looks as above. We have created a Carousel with three components inside to test the functionality of the renderImage function. In the future sprints, we shall edit this so that there is a map function that takes all of the images and renders them.

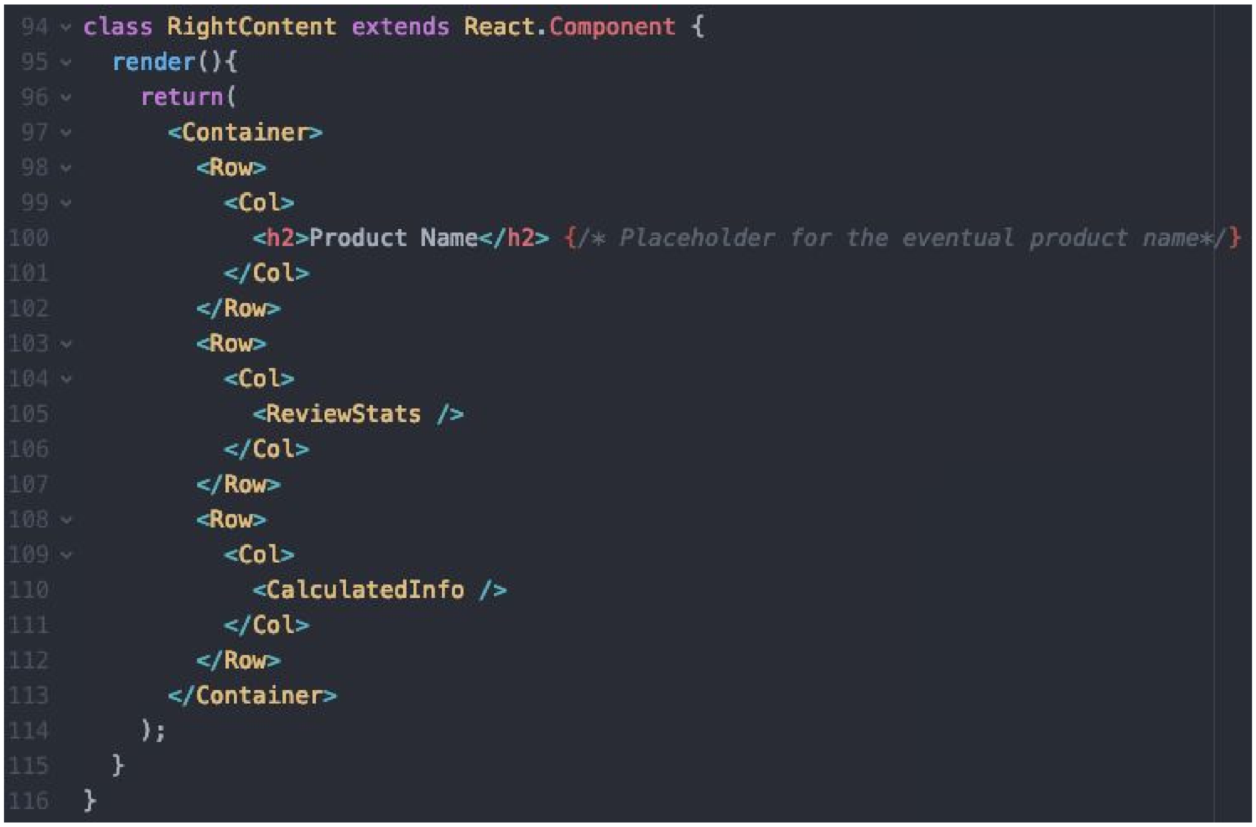
For now, we can see the webpage looks like this:


localhost
Apple iCloud Google Yahoo Bing Wikipedia Facebook Twitter LinkedIn The Weather Channel Yelp TripAdvisor BBC
devin thomas
Amazon Product URL
Analyse
Right Content
Price:
Seller:
Amazon Rating:
Dimensions:

The test image has been rendered and within the image, three small bars can be seen indicating the images provided. This is sufficient evidence to ensure that the Carousel is now working.

Inside Right Content

We must first update the `RightContent` class so that it contains the appropriate styling for the contents within it. This can be seen in the code below. Currently, we do not have the product names that are being scraped. Therefore, we need to include a placeholder. Eventually, in a later sprint, we will update this to include the variable of the product name that has been scraped from the Amazon site.

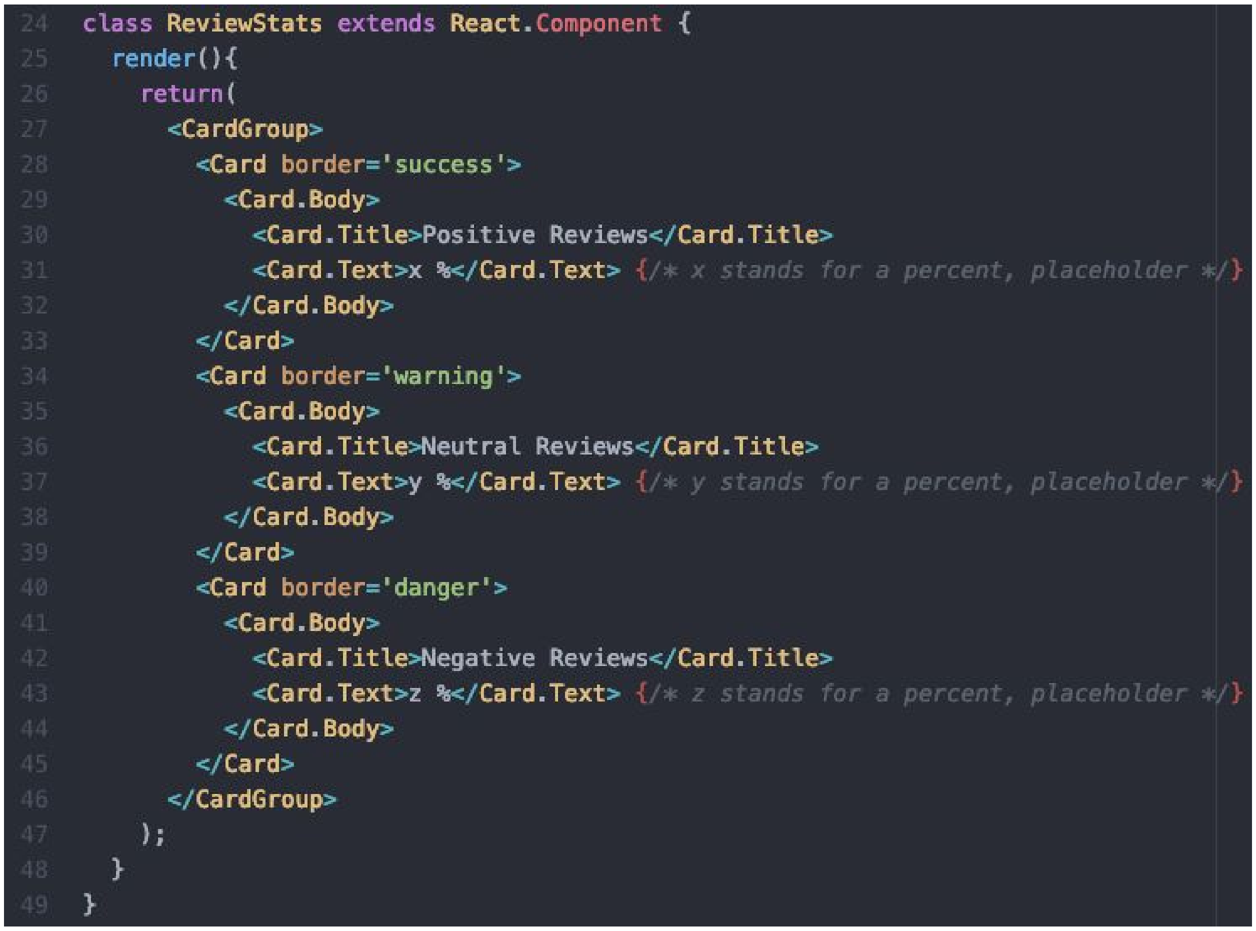


```
class RightContent extends React.Component {
  render(){
    return(
      <Container>
        <Row>
          <Col>
            <h2>Product Name</h2> {/* Placeholder for the eventual product name*/}
          </Col>
        </Row>
        <Row>
          <Col>
            <ReviewStats />
          </Col>
        </Row>
        <Row>
          <Col>
            <CalculatedInfo />
          </Col>
        </Row>
      </Container>
    );
  }
}
```

As seen above, there are three separate rows for the individual parts within the right content. The product name placeholder is at the top followed by the `ReviewStats` class and the `CalculatedInfo` class. By separating out the individual components, we are making the code easier to read and maintain. Variables can be easily passed through props and states and so it is beneficial to improve the readability of the code. Additionally, it also makes sure that when a component needs to be changed, the others remain untouched and so changes occur singly.

Review Statistics

Within the review statistics there are three sections: the positive reviews, negative reviews and the neutral reviews. After having analysed all the reviews, we must then display the percent of positive reviews, negative reviews and neutral reviews. By being passed a float, we can include a converter that multiplies by 100 to produce a percent for the reviews. For example 0.8 positive would be 80% positive reviews.

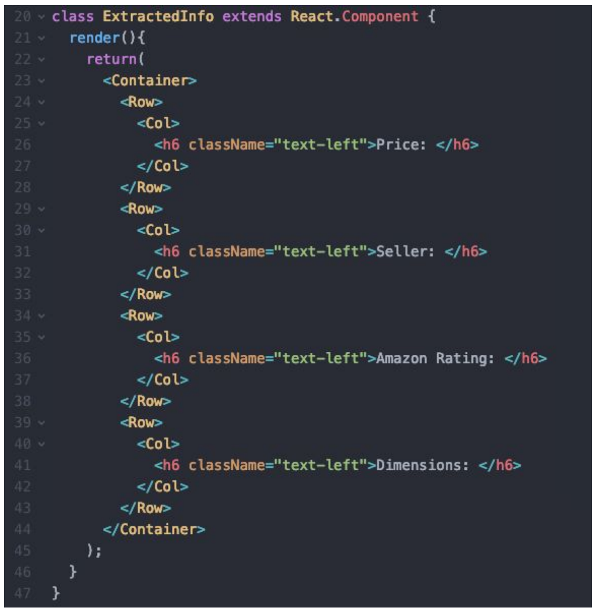


```
class ReviewStats extends React.Component {
  render(){
    return(
      <CardGroup>
        <Card border='success'>
          <Card.Body>
            <Card.Title>Positive Reviews</Card.Title>
            <Card.Text>x %</Card.Text> {/* x stands for a percent, placeholder */}
          </Card.Body>
        </Card>
        <Card border='warning'>
          <Card.Body>
            <Card.Title>Neutral Reviews</Card.Title>
            <Card.Text>y %</Card.Text> {/* y stands for a percent, placeholder */}
          </Card.Body>
        </Card>
        <Card border='danger'>
          <Card.Body>
            <Card.Title>Negative Reviews</Card.Title>
            <Card.Text>z %</Card.Text> {/* z stands for a percent, placeholder */}
          </Card.Body>
        </Card>
      </CardGroup>
    );
  }
}
```

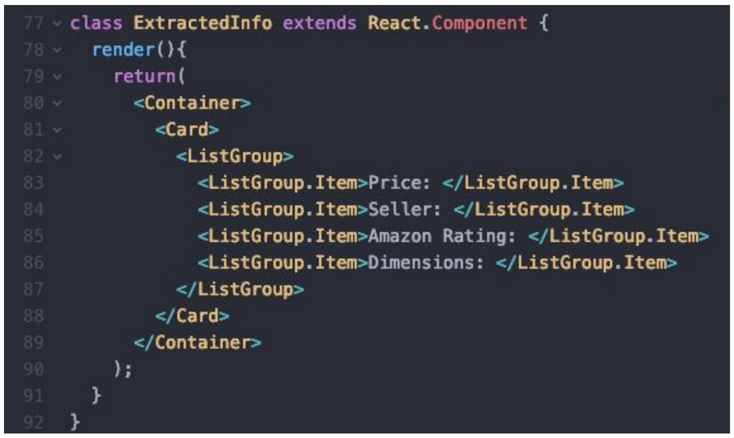
Within the code above, we now have a new class called `ReviewStats` as mentioned before. Within this class we will render a new components called Cards and Card Groups. These components allow us to create three separate sections for the different sentiments of reviews. As bootstrap allows for cascading websites, these three sections will sit above next to each other on a desktop screen but stacked on a mobile screen.

Having looked at Card Groups for the sentiments of reviews, I decided to revisit the structure of the `ExtractedInfo` class. Originally, I had stacked all the different pieces of information on top of each other so that they could be displayed. However, the most effective way of presenting this data would be to use another component called a List Group.

As a result, the left side of code has been transformed into the right side seen below:

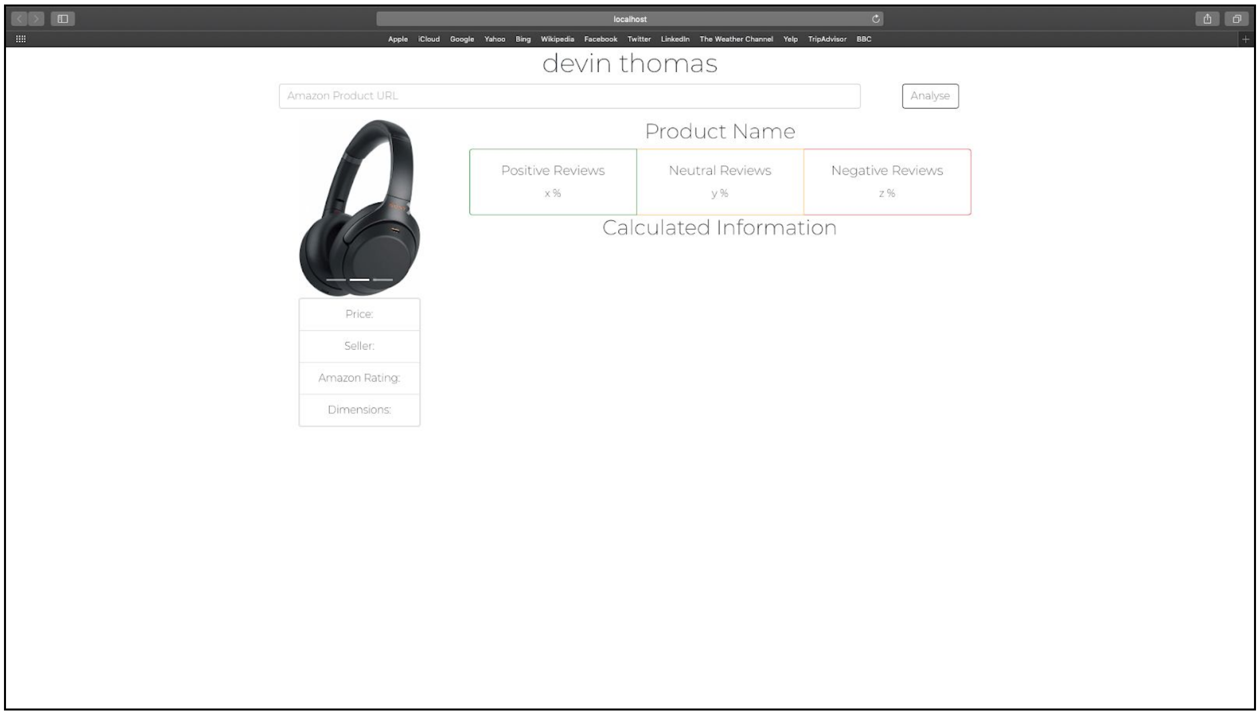


```
class ExtractedInfo extends React.Component {
  render(){
    return(
      <Container>
        <Row>
          <Col>
            <h6 className="text-left">Price: </h6>
          </Col>
        </Row>
        <Row>
          <Col>
            <h6 className="text-left">Seller: </h6>
          </Col>
        </Row>
        <Row>
          <Col>
            <h6 className="text-left">Amazon Rating: </h6>
          </Col>
        </Row>
        <Row>
          <Col>
            <h6 className="text-left">Dimensions: </h6>
          </Col>
        </Row>
      </Container>
    );
  }
}
```



```
class ExtractedInfo extends React.Component {
  render(){
    return(
      <Container>
        <Card>
          <ListGroup>
            <ListGroup.Item>Price: </ListGroup.Item>
            <ListGroup.Item>Seller: </ListGroup.Item>
            <ListGroup.Item>Amazon Rating: </ListGroup.Item>
            <ListGroup.Item>Dimensions: </ListGroup.Item>
          </ListGroup>
        </Card>
      </Container>
    );
  }
}
```

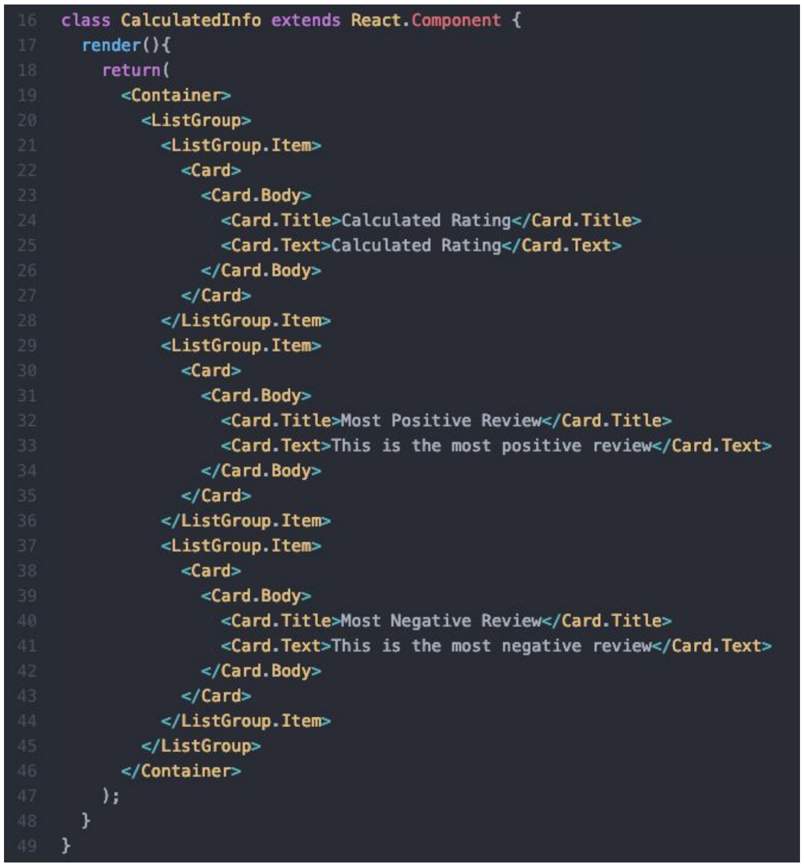
Now, when relaunching the local server and reloading the webpage, we see this screen:



Now, we can see three sections within the `ReviewStats` class being shown. In order to improve the aesthetic of the webpage and improve the usability, I have made the outlines of the different sentiments of review representative of the actual sentiment. The positive reviews are green, the neutral are yellow and negative are red. This is in order to allow the user to read the different values quickly and easily by using colours to associate the right sentiment to the information provided within the Card.

Calculated Information

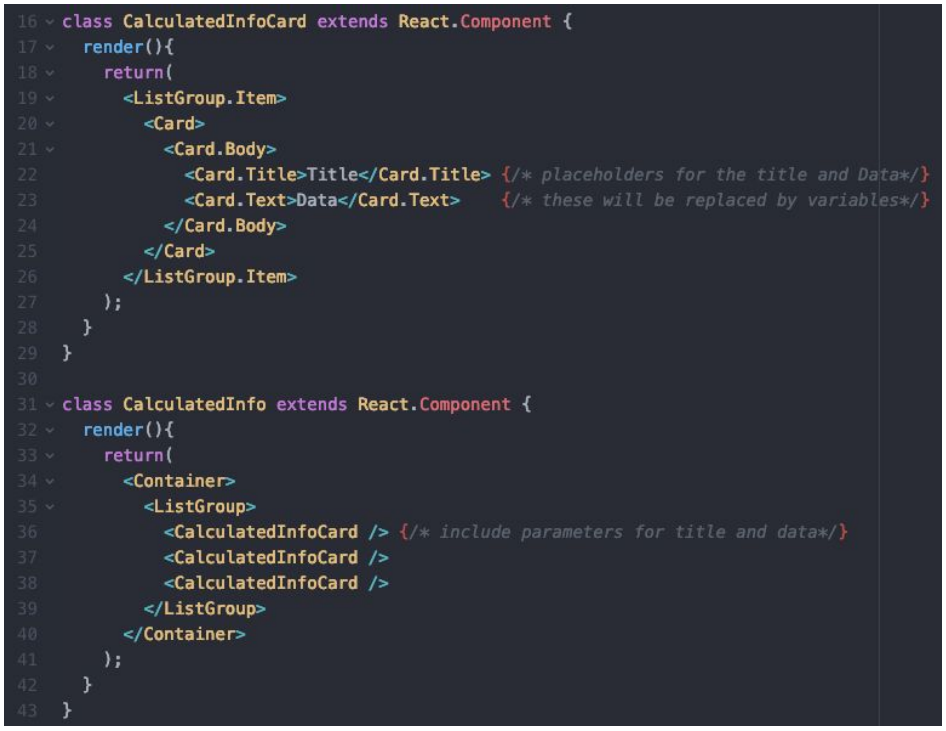
When displaying the calculated information, it is necessary to discuss what types of information are being displayed. It will be necessary to provide the most positive and most negative reviews. These could therefore be displayed in Cards. Additionally we will be displaying the calculated rating. Therefore we will display these three categories within Cards and a List Group.



```
class CalculatedInfo extends React.Component {
  render(){
    return(
      <Container>
        <ListGroup>
          <ListGroup.Item>
            <Card>
              <Card.Body>
                <Card.Title>Calculated Rating</Card.Title>
                <Card.Text>Calculated Rating</Card.Text>
              </Card.Body>
            </Card>
          </ListGroup.Item>
          <ListGroup.Item>
            <Card>
              <Card.Body>
                <Card.Title>Most Positive Review</Card.Title>
                <Card.Text>This is the most positive review</Card.Text>
              </Card.Body>
            </Card>
          </ListGroup.Item>
          <ListGroup.Item>
            <Card>
              <Card.Body>
                <Card.Title>Most Negative Review</Card.Title>
                <Card.Text>This is the most negative review</Card.Text>
              </Card.Body>
            </Card>
          </ListGroup.Item>
        </ListGroup>
      </Container>
    );
  }
}
```

The `CalculatedInfo` class has been created within this piece of code. Here I have created a List Group that contains three different Cards within it. This is so that the user can now see three separate pieces of information that have been calculated by the system.

However, there is a lot of repetitive code within this. As a result, we could turn this into a single class called `CalculatedInfoCard` and reference it within the `CalculatedInfo` class. We can provide the variables within this parent class so that they can be accessed by the children.

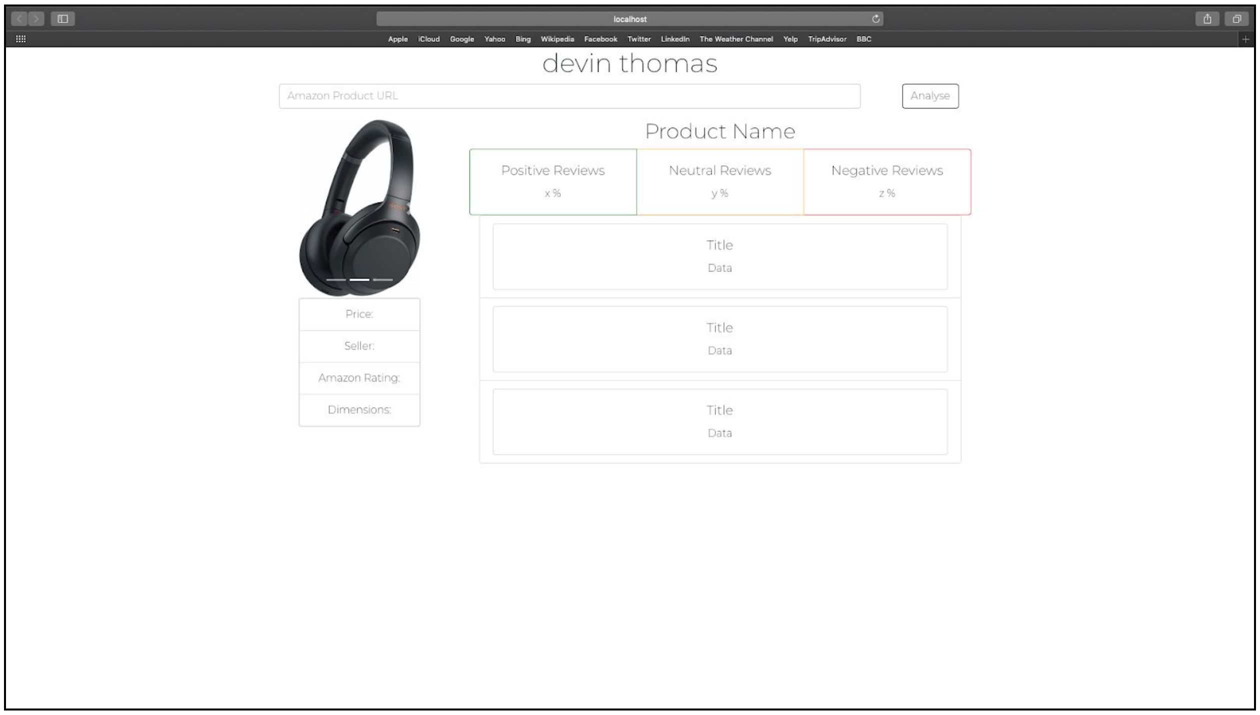


```
class CalculatedInfoCard extends React.Component {
  render(){
    return(
      <ListGroup.Item>
        <Card>
          <Card.Body>
            <Card.Title>Title</Card.Title> {/* placeholders for the title and Data*/}
            <Card.Text>Data</Card.Text>    {/* these will be replaced by variables*/}
          </Card.Body>
        </Card>
      </ListGroup.Item>
    );
  }
}

class CalculatedInfo extends React.Component {
  render(){
    return(
      <Container>
        <ListGroup>
          <CalculatedInfoCard /> {/* include parameters for title and data*/}
          <CalculatedInfoCard />
          <CalculatedInfoCard />
        </ListGroup>
      </Container>
    );
  }
}
```

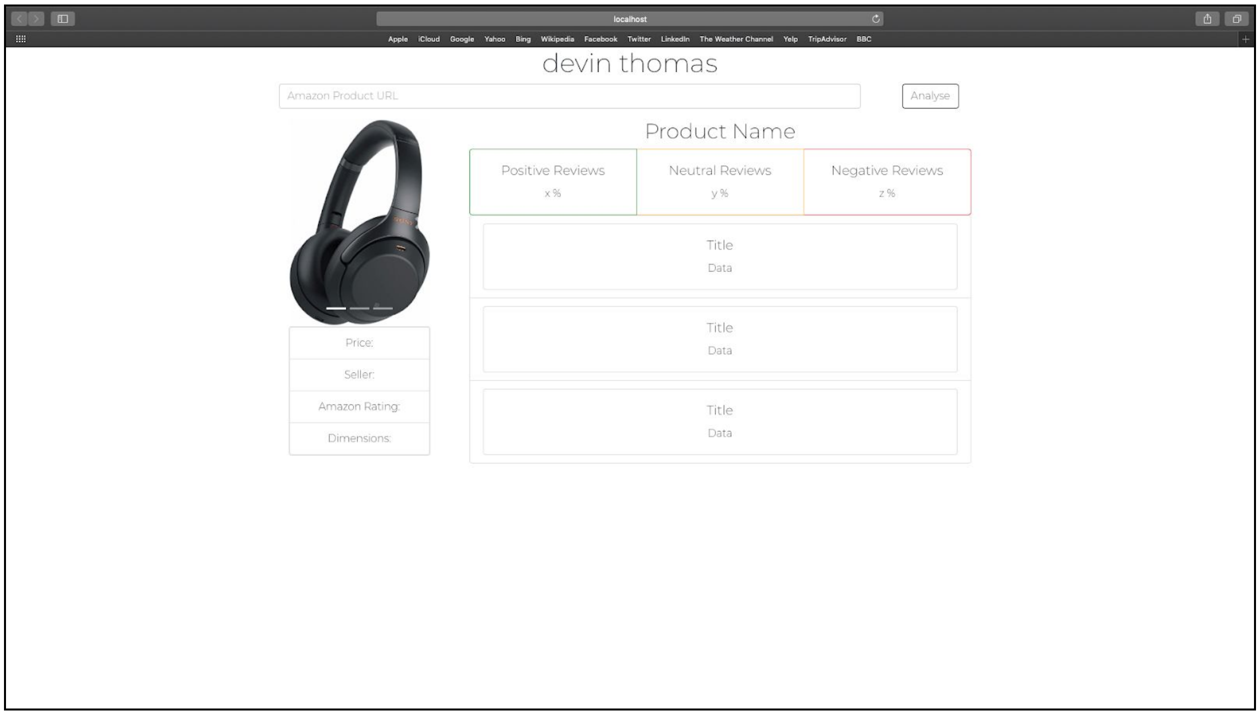
By introducing this new class, we are now able to cut down the amount of code required. We have, however, added to the complexity of the solution as more variables will have to be passed through the hierarchy. However, we have made the code much more readable and maintainable.

Additionally, this code makes it very easy to add new features later on. This may be very useful within Sprint 5 where we will look at incorporating new features.

When reloading the page, we observe this result:



After careful observation of the code having completed the development for this sprint, I noticed that I had included a few redundant Container components. This was resulting in an odd shape and arrangement of the sections I have included. By removing them, the resulting web page looks much more elegant and consistent.



Looking at the final webpage compared to the previous one, we can see that the image has grown as well as the extracted information section. In addition, the right content section is more aligned and the calculated information section is clearly aligned with the review statistics.

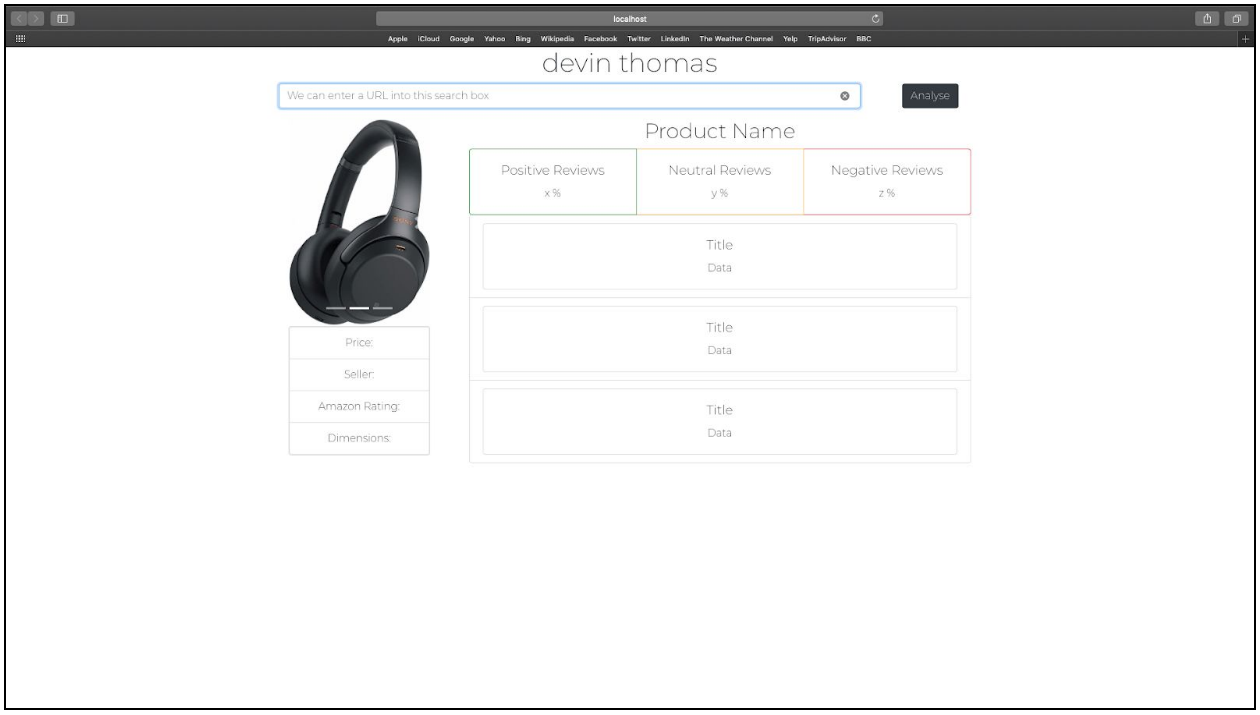
Testing

To Inform Development

Throughout the development process, I was testing the code consistently. This was done every time that the webpage was reloaded. Currently as it is just a structural layout, there is no validation required. The only piece of testing is checking that the page loads without crashing. As in all cases (apart from errors) the code compiled effectively and the page was displayed, each test was being passed.

To Inform Evaluation

In order to inform evaluation, we need to first check the functionality of the resulting webpage. As it is currently a simple structural design, there is not much that it can do. The only interactable parts are the images, the search bar and the Analyse Button. These parts are all clickable and accessible as shown in the figures below.



We can test the usability of the system by giving it to a range of different people who belong to the target market. As this includes a wide variety of people, it would be very easy to find a focus group and distribute images or show them the current locally hosted server with the webpage on it. This will allow us to look at how receptive users are to the design of the system and find out what improvements could be made. For example, some colours may be changed in order to keep the user's attention. Additionally, the layout could be changed in order to make the system more intuitive to use. The easy creation of classes using React and React-Bootstrap makes all the components very easy to move around and manipulate — therefore decreasing the time taken to meet the clients needs after feedback.

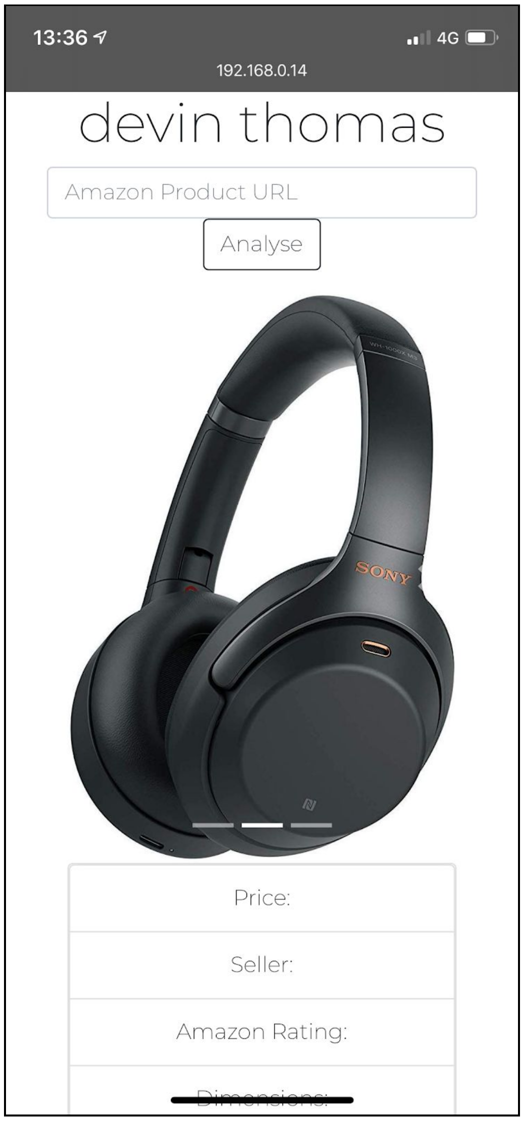
Once again, as this is simply a design prototype, there is no way to test valid, invalid or borderline inputs for robustness. Either the page loads or it doesn't. Later on, we will have to use validation to make sure that the JSON response is valid but currently, this is not necessary.

Evaluation

Success Criteria

When going through the success criteria in the Analysis section, we can now look at whether they have been met.

1	The User Interface		
Criteria Reference	**Criteria**	**Test Strategy**	**Completed?**
1.1	Display a URL search box and submit button.	There should be a text input box with a submit button.	🟩
1.2	Return the first image of the product and all relevant information.	An image should appear as well as a Card holding the product information.	🟩
1.3	Display a result.	A calculated rating should appear as well as the confidence that the product is positive or negatively rated.	🟩
1.4	Display Amazon results	The extra information about a product should be visible. For example, dimensions.	🟩
1.5	Restart search.	A restart button should be included or the search bar and submit button must always appear at the top of the page.	🟥



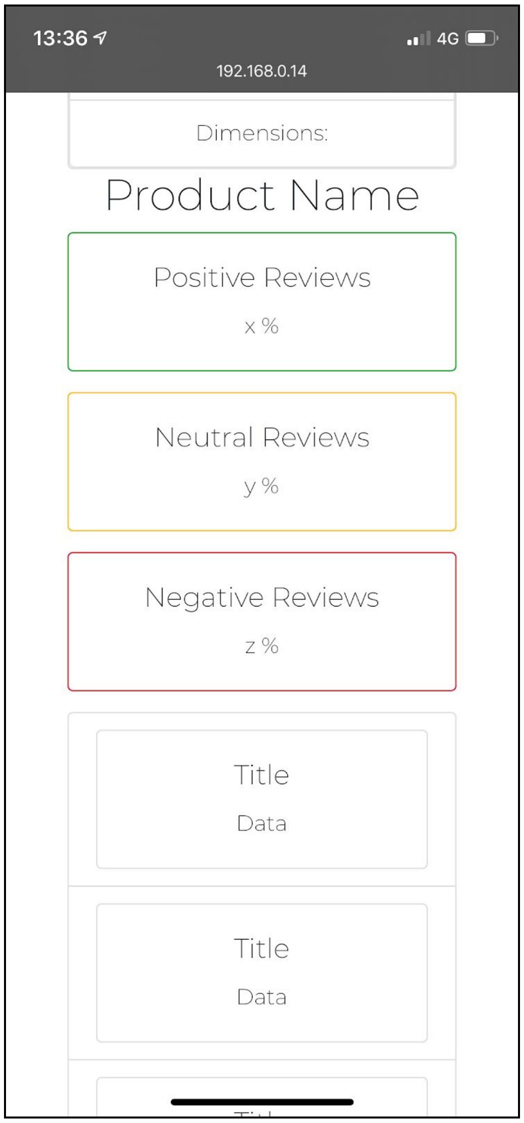


Figure 8. Top half of the webpage on mobile

Figure 9. Bottom half of the webpage on mobile

<u>1.1 — URL Search Box and Analyse Button</u>

1.1 has been a success as when loading the page, a search box and the Analyse Button appear. Additionally, we can test the resizing aspect of the webpage as seen in Figure 8 and 9. The search box resize to the width of the screen however the button remains small. In order to improve on this, I should allow the button to take the screen's width once it has stacked underneath the search box. By doing so, the usability of the system is improved because users may struggle pressing a small button on a mobile screen.

1.2 — Image Display

1.2 has also been a success as we can display multiple images on a carousel that automatically rotates. This test was important and the criteria was essential as the user will likely want to view the product that they are looking at. Therefore, it is necessary to provide some form of image of said product. We can extract the image from Amazon but for now we have included a simple test case that can be easily manipulated by providing the link for that image. The images are also nicely sized and visible; they do not distract the user from the predominant information being displayed on the screen and provide as visual stimulation to improve usability and interest within the system.

1.3 — Calculated Results Display

1.3 has been successfully completed. Within Figures 8 and 9 shown above, we can see that the webpage will display the calculated information provided by the backend. Additionally, by making the code more modular, I have been able to make sprint 5 an easier process. I can add new features and information seamlessly by including new Cards within the `CalculatedInfo` class. Each section also efficiently stacks on top of one another and maintains its high level of readability and usability within the mobile resizing.

1.4 — Extracted Results Display

1.4 is very similar to 1.3 and as such has also been completed. Seen in Figures 8 and 9 the extracted information is ready to be implemented once the data has been scraped from the Amazon page. The sections also stack well so that readability and usability on a mobile format is retained.

1.5 — Restart Search

However, 1.5 has not been met. This is because we do not require a restart search button. As the URL search bar will remain at the top, if the user wants to search for another item, they can simply input a new URL and press the Analyse Button or Enter. This will reload the page with the appropriate information about the product they now want to view.

Maintenance Issues

By creating modular code, it remains very maintainable. Each section is created as a separate class within a hierarchy. Therefore, when new sections and features are to be added, I can easily incorporate them without having to write extensive amounts of code.

Additionally, by adding comments as to where I will eventually have to edit the code proves useful and maintainable as I will easily be able to find out where the new sections of code need to be added.

Limitations

As there is currently not much functionality within the structure of the webpage, there is not much to comment on the limitations of it. Currently the system performs as required and there is nothing to exploit the system. The interaction between the user and the system is very limited and all validation and error handling will be dealt with the backend rather than the frontend. This allows me to deal with all the possible errors in a single place rather than spreading them out over a variety of places. Additionally, this puts all the error checking server-side. This is less ideal as it means that the server may become inundated as many requests are sent to it. As a result, it would be beneficial to move validation to the frontend later on such as verifying that a URL is a valid URL and follows the standard rules for a URL.

Future Development

Currently the webpage is entirely structural but eventually variables will have to be passed between the classes when products are being reviewed and analysed. Variables such as the product name, price, dimensions as well as all the calculated information. I will have to look at including props and states into the classes in order to access the provided variables.

Additionally, new functions will have to be added to the classes in order to forward the props and states between class components and their parent components.

Submission and editing of the search bar will also have to be dealt with by creating new functions to handle the process that occur behind them.

Sprint 2

Legal Considerations

One of the issues that we may have is the idea of scraping Amazon to be against their terms of Service. However, after extensive research, I have discovered that my intended approach is not against their terms of service and therefore we are allowed to proceed.

Originally, Amazon used to provide access to product reviews through their Product Advertising API to developers and sellers. However, they discontinued this service on November 8th 2010, preventing customers from displaying Amazon reviews about their products, embedded in their own websites. Now, Amazon will only return a link to the review.

ScrapeHero[10], a data company, has published a comprehensive tutorial on how to successfully scrape the Amazon products and have their own cloud scraping service which can be used through an API. This will allow the backend here all to be done externally and with results being sent back across. This is a paid service that I may potentially use in the future to speed up the review scraping.

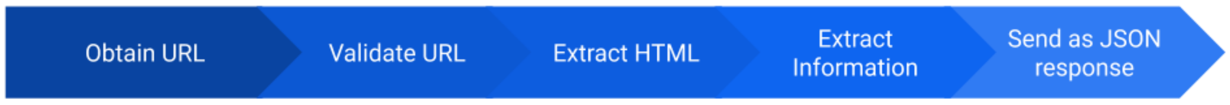
Furthermore, I have discovered that it is indeed illegal to scrape the HTML files of another website and portray it as your own. However, as I am scraping only some of the information from Amazon, this is not in breach. Moreover, I will be referencing Amazon in the website in order to make it absolutely clear that I am using their data and have obtained it from them directly.

Design

In order to scrape the webpage of the product, I will use a python library called Beautiful Soup. Beautiful Soup is a library that is used to pull data from HTML and XML files. Therefore, we can pass the URL of the Amazon product to the web scraping service which can then extract the HTML to locate the required variables such as the product name, the price, the seller and the rating.

Here is the sequential process that the web scraper must perform.

[10] https://www.scrapehero.com/how-to-scrape-amazon-product-reviews/



Firstly, we must obtain the URL from the user. This will be done by extracting the Amazon Standard Identification Number (ASIN)

The URL then must be validated to ensure that it there is a real product with that ASIN. This can be done by requesting the HTML file and seeing if it returns anything.

After receiving the URL, the scraper must extract the HTML from the webpage.

By searching through all the HTML classes, we can find all the relevant information about the product such as the product name, price and the current Amazon star rating.

Once the response has been formed, it needs to be sent when requested by the frontend.

We can install Beautiful Soup using the command:

```
pip install bs4
```

Beautiful Soup will allow us to request the HTML file of a given URL. It will then allow us to search through the returned file for specific components such as divs or classes in order to find specific HTML elements with the required information inside.

Generic Functions Used

In order to scrape the correct pieces of information from Amazon, we are going to first extract the whole HTML file of the webpage. Then, we will traverse the HTML and look for components with the required IDs or classes. We can complete this as such:

```
soup = HTML content
for divs in soup.Find('div')
    try:
        price = divs[data-asin-price].ToString
        product_json["price"] = '£' + price
        break
    except:
        pass
endfor
```

We are also going to have to update the React frontend so that the JSON response variables can be passed back and displayed when the user has submitted a URL to the system.

For the Analyse button, we will incorporate a procedure within the class that looks like this:

```
constructor(props){
      super(props)
      this.handleClick = this.handleClick.bind(this)
}
```

Then, within the class, we create a procedure that is referenced in the constructor procedure called handleClick. This procedure will deal with the events that occur once the Analyse button has been clicked. It will fetch the data from the backend web scraper and the reviews analysis system and begin the process of updating the ReactJS components.

```
handleClick() {
        fetch(send request to backend for information)
        save result as JSON
        update results // calls a function of a parent class
}
```

Then, within the same Component class, the button that calls the procedure `handleClick` will be written like this. The onClick function is called as seen below. This is necessary in order to trigger the fetch of the product information.

```
<Button type='button'onClick={this.handleClick}>Aggregate Variables
</Button>
```

This is the process of writing new functions within each React class in order to forward the variables through the containers and their child components. To do this, we need to write the required function in the parent class and pass it down to the child as such:

```
<Component updateResults={this.updateResults} product_name
={this.state.product_name}/>
```

The procedure `updateResults` in this case is written in the parent class and the state of `product_name` is declared in the parent class. Then, when calling an instance of the Component child class in the render procedure, the procedures and states are referenced in this manner. This can be passed through children to where it is required.

Usability

Sprint 2 deals with the backend, due to this, there is very limited usability to discuss. The user will not directly interact with the web scraper other than inputting a URL. The code remains very modular as functions are written to find certain HTML components and as the backend works in isolation to the frontend. In terms of usability, we could argue that the reusability of the code improves usability. This is because the code is repetitive when searching for HTML components. Additionally, when having to search for new components, the process will be the same.

Key Variables and Data Structures

As this sprint will focus on the development of the web scraping service, there are many variables that have to be accounted for that focus on the data about the product itself. This data must be collected in a JSON response and unpacked by the frontend to be displayed.

Variable	Data Type	Use
productName	String	The user must be able to see the name of the product that they are researching.
price	Float	The user must be able to see the price of the product.
seller	String	Another piece of information is the seller. An authorised and trusted seller is more likely to have a better rating.
dimensions	String	Dimensions aren't a critical piece of data but provides contextual information about the product.
rating	Float	The original Amazon rating would be an important piece of information for the user to have.
asin	String	The Amazon Standard Identification Number (ASIN) is required in order to identify the product that is being searched.
reviews	Array	The reviews that are scraped must be stored together in order to be sequentially analysed and reviewed.
review	String	When a single review is being analysed, it will be stored as a string.
response	Dictionary	All of these fetched pieces of data must be sent to the frontend in a JSON package. In order to do this, the backend must store all these values in a dictionary and

		each item is appended to it by using the variable name as the key and the extracted information for the data.

Validation

The first part of validation will be necessary here. If the user inputs a URL of a product that doesn't exist, the system must be able to deal with this case and handle the outcome successfully.

Additionally, we must consider the case where products are valid but have no reviews or no ratings. Consequently, the system must be able to respond by returning a message to the user stating the lack of reviews and ratings. As such, the system must be able to stop after this.

In order to account for both of these cases, we can look to see whether the reviews are empty. If they are, we can discount this product as unviable for analysis. A simple algorithm can be used to solve this.

```
if reviews:
      EMPTY = false
else
      EMPTY  = true
```

Essentially, if the list is empty, the if statement will return false and it will be caught. If the list is not empty, the if statement will return true and the analysis can continue.

We must carry out this validation in order to catch whenever a URL leads to an invalid product or whether a product has no reviews to analyse. In both of these cases, the reviews list will be returned as empty and so we can catch these cases together. By doing so, we eliminate any risk of the system trying to analyse a product with no reviews.

Testing

Testing this sprint will be more extensive than the testing in Sprint 1. This is because during this sprint we are taking an input from the user and thus introduce an opening for abuse and error. For example, the user may supply the URL of a product that doesn't exist and therefore the system will not be able to extract any information from a webpage.

This sprint will be tested against the second part of the testing table defined in the Success Criteria in the Analysis section.

Type	Reason	Example	Output

Valid	When a valid URL is provided, the webpage should be scraped and the information returned.	amazon.co.uk/… (VALID)	The webpage should update to store the new values of all the key variables.
Invalid	When an invalid URL is provided, the system should recognise that no data is returned upon trying to scrape it	apple.com/… (INVALID), amazon.co.uk/… (INVALID)	Returns nothing, an appropriate message returned to the user
Valid URL with no reviews	When a product with no reviews is input, the system will not be able to analyse the sentiment of them and as such, must be able to handle these exceptions	amazon.co.uk/…(VALID with no reviews)	The webpage should display an error message.

Post Development

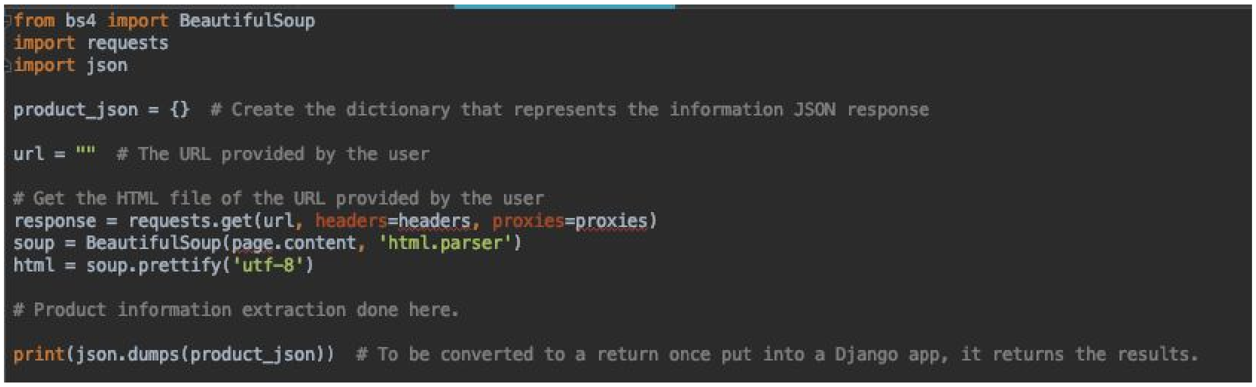
Regarding this sprint, we will need to return to the backend whilst creating the reviews analysis. In doing so, we can append to the JSON response after the analysis has been composited to return to the frontend. The frontend will then have to unpack this new data and display it to the user. Currently, we are only looking at the web scraping element so we are not concerned with this part of the development. The modularity of the code allows us to add new elements to the JSON response and display them when required. Additionally, the modularity of React allows us to add new components when required.

Development

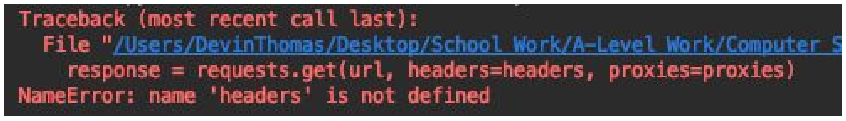
Setup

To begin, we need to create a new Python file that will be copied into a Django app when the scraper is up and running.

The first pieces of code that must be written are the import statements to include libraries that are required to effectively scrape Amazon products. Additionally, we can also set up the initial statements to return and obtain the HTML before encasing them in functions later. By encasing them inside functions, we create code that is more reusable and more modular. Furthermore, it will be easier to follow through.

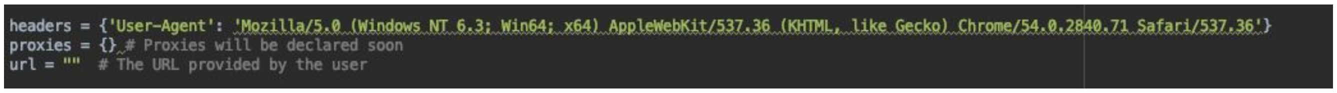


```python
from bs4 import BeautifulSoup
import requests
import json

product_json = {}  # Create the dictionary that represents the information JSON response

url = ""  # The URL provided by the user

# Get the HTML file of the URL provided by the user
response = requests.get(url, headers=headers, proxies=proxies)
soup = BeautifulSoup(page.content, 'html.parser')
html = soup.prettify('utf-8')

# Product information extraction done here.

print(json.dumps(product_json))  # To be converted to a return once put into a Django app, it returns the results.
```



```
Traceback (most recent call last):
  File "/Users/DevinThomas/Desktop/School Work/A-Level Work/Computer S
    response = requests.get(url, headers=headers, proxies=proxies)
NameError: name 'headers' is not defined
```

The code above shows two syntax errors. We have declared two parameters `headers` and `proxies`. We can fix the errors by declaring what these two variables should be. The headers are used in order to state where the requests come from and the proxies are required to act as an intermediary between the requests from the client and the server.



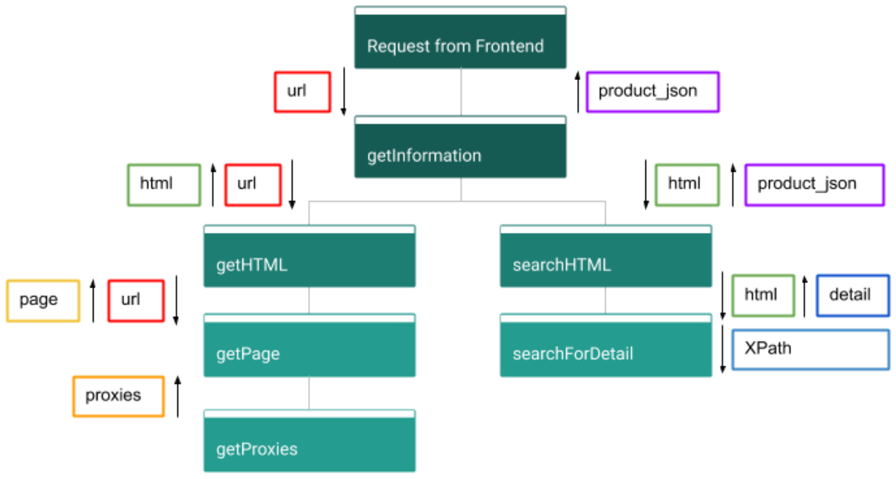
```python
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36'}
proxies = {} # Proxies will be declared soon
url = ""  # The URL provided by the user
```

Here, I have added the two undeclared variables in order to fix the errors.

Modular Functions

In order to write more efficient and reusable code, we can convert the program from its sequential order to reusable functions. Though this will make the code more difficult to follow through, it will enable us to make the program more efficient by writing less code to complete multiple instructions multiple times. For example, we can create a function that will search through the HTML for the specific product information. Rather than write a condition controlled loop for every variable, we can use a single function and pass the XPath as an argument in order to locate that piece of information.

The process will work in this hierarchy:



As shown in this hierarchy, a request comes from the frontend. A function called `GetInformation` is called. This function will call two functions: `getHTML` and `SearchHTML`. The `GetHTML` function will get the page and requests possible proxies to use from the `getProxies` function. The page is formatted and passed back to the `GetInformation` function. The `SearchHTML` function will then call the `SearchForDetail` function to individually find each of the details required. Within each transition, the data passed through has been tracked in order to easily identify what parameters are required to be declared whilst creating the functions. They have been colour coordinated in order of a rainbow to track the progression of the variables. It starts with the url which is input by the user. It travels down the functions to `getPage` where it receives the proxies from `getProxies`. It then will return the page to `getHTML` which extracts the HTML and returns the html. The variable html is passed down to `searchForDetail` along with the XPath of each detail being searched for. The detail is passed back and stored in product_json which is returned at the end.

getInformation

This function is the base function. It is called to collect the information. It has one parameter: url. It will return the product json collected. Within its sequence, it will have to call the function `getPage`. `getInformation` will have this structure:

```
function getInformation(string url)
      html = getHTML(url)
      product_json = searchHTML(html)
      return product_json
endfunction
```

getHTML

This function regards the collection of the html from the page of the Amazon product. It takes the url of the product from the user and will return the correctly formatted html. It calls the function `getPage` to get the page. This function will parse the results of that function and format it so that it becomes searchable.

```
function getHTML(string url)
      proxies = getProxies()
      fetched = false
      while not fetched
            proxy = next(proxies)
            try page = getPage(url, proxy)
            if not fetched, try next proxy
      soup = BeautifulSoup(page.content, 'html.parser')
      html = soup.prettify('utf-8')
      return html
endfunction
```

getPage

The function `getPage` will be used to actually fetch the page of the Amazon product before any Beautiful Soup manipulation. It will take the parameter `url` and `proxy` and return the variable `page`.

```
function getPage(string url, string sproxy)
      page = get url
      return page
endfunction
```

getProxies

Proxies are required in order to act as an intermediary between the client and the server. This function will be used to rotate a pool of proxies and return one. We can get the proxies from a site called free-proxy-list.net[11]. The structure of this function will be as such:

```
function getProxies()
      fetch HTML from 'https://free-proxy-list.net'
      proxies = find proxies
      return proxies
endfunction
```

searchHTML

This function concerns the extraction of the details by the web scraper. It will call the function `searchForDetail` to individually search for the required details. By doing this, we can pass the XPath of the element to search for to the `searchForDetail` which will return the detail. By using this function, we can easily add new details to add for when expanding the solution. `searchHTML` will look like this:

```
function searchHTML(string html, string xpath)
      product_json = {}
      xpaths = {"name": "" // Product name xpath,
                "price": "" // Price xpath
                 ... // All other necessary xpaths
               }
      for key in xpaths
            detail = searchForDetail(html, xpaths[key])
            product_json[key] = detail
      next key
      return product_json
endfunction
```

searchForDetail

Function `searchForDetail` will be used to individually search for the required pieces of data within the supplied HTML to extract the relevant data to display to the user. The xpath of the detail and the HTML itself will be passed as parameters to this function. It will return the detail itself. The function will be constructed as such:

[11] https://www.free-proxy-list.net

```
function searchForDetail(string html, string xpath)
      for element in html
            if element.xpath = xpath then
                  tag_details = element.detail
                  break
      next element
      return tag_details
endfunction
```

By using these modular functions, we can rewrite the new code like this:

Evaluation

Sprint 3

Design