

E-Commerce Web Application

Enrol. No. (s) - 16103333, 16103274

Name of Student (s) - Amrinder Singh, Shubham Pandey

Name of supervisor(s) - Dr. Prakash Kumar



Dec - 2019

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

in

Computer Science Engineering

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING &
INFORMATION TECHNOLOGY**

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

(I)

TABLE OF CONTENTS

Chapter No.	Topics	No.
	Student Declaration	3
	Certificate from supervisor	4
	Acknowledgement	5
	Summary	6
	List of Figures	8
	List of acronyms/ meanings	9
Chapter-1	Introduction	10 to 19
	1.1 General Introduction	
	1.2 Problem Statement	
	1.3 Significance/Novelty of the problem	
	1.4 Brief Description of the Solution Approach	
	1.5 Comparison of existing approaches to the problem framed	
Chapter-2	Literature Survey	20 to 22
	2.1 Summary of papers studied	
Chapter 3:	Requirement Analysis and Solution Approach	23 to 28
	3.1 Overall description of the project and Solution Approach	
	3.2 Requirement Analysis	
Chapter-4	Modelling and Implementation Details	28 to 41

Chapter-6	Conclusions, Future Scope & Results	42 to 48
	6.1 Results	
	6.2 Conclusion	
	6.3 Future Scope	
References	IEEE Format (Listed alphabetically)	49 to last

(II)

DECLARATION

We hereby declare that this submission is my/our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name: Amrinder Singh

Enrolment No: 16103333

Signature:

Name: Shubham Pandey

Enrolment No: 16103274

Place: Noida, Uttar Pradesh, India

Date:

(III)

CERTIFICATE

This is to certify that the work titled “**E-Commerce Web Application**” submitted by “**Shubham Pandey, Amrinder Singh**” in partial fulfilment for the award of degree of **B. Tech** of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

Name of Supervisor Dr. Prakash Kumar

Date

Signature of Co-Supervisor

Name of Supervisor

Date

(IV)

ACKNOWLEDGMENT

We would like to express our deepest gratitude to all those who provided us the support to complete this project within the stipulated time frame. A special gratitude to our final year project supervisor, co-supervisor whose contribution in stimulating suggestions and encouragement, helped us to accomplish our project. Furthermore, we would also like to acknowledge with much appreciation the crucial role of the staff for our mid-term evaluation, whose suggestions to improve project outlines and features gave direction to the project and helped in its improvement. A special thank goes to our batch mates who also gave us suggestions to improve further on our project. Last but not least, we would like to Thank all the professors who have invested their efforts in guiding us throughout the academic curriculum with the technical expertise required in achieving the goal. We Appreciate the guidance given by other supervisor as well as the panels for their comments and advices especially in our project presentation that has helped improved our presentation skills and project as well.

Signature:

Name: Shubham Pandey

Enrolment No: 16103274

Signature:

Name: Amrinder Singh

Enrolment No: 16103333

(V)

SUMMARY

Earlier the web development is done in original Javascript which is also called Vanilla Javascript combined with the frontend tools like HTML which stands for Hypertext Markup Language, which was used to create a structure of the websites and also the styling, now the styling part has become separated from the structure part and now the styling is done using a separate component which is called CSS which stands for Cascading Style Sheets.

Dividing the development part in three components makes it easier for us to develop and segment our websites. It allows the work to be divided easily among different team members also.

Also, a segmented website is easier to update and in future if there is a need to scale the website in future it will become a little easier to update.

Now days, A lot of various softwares are being used for developing different components of the websites viz. Frontend, Backend and the interface which allows the interaction between the frontend and backend called APIs, sometimes the APIs can be made using the Backend framework and sometimes the APIs framework can be embedded in the frontend framework, an example of that is React, React have some libraries like axios etc. which allows the creation of the APIs.

For creation of APIs a new design pattern is being used nowadays which is called RESTful APIs. In the project we used CRUD design pattern to create the APIs.

CRUD stands for Create, Read, Update and Delete, although we did not needed the Delete functionality here but we want this functionality also, so we tried adding the delete functionality also in the project.

The project is made using pure HTML, CSS with Bootstrap and pure Vanilla Javascript.

The website which we choose to create is an e-commerce website which can be used to sell various products of no certain types, where the seller decides the products and the price. It allows selling of old and new both types of products.

It sounds like a small scale Amazon, but amazon did not allows the selling of old products, in our case the selling of old products are also allowed, we wanted to add a lot of functionalities of Amazon in our site like the sort functionallity but while trying

to implement it we found that how complex it is to build these advanced functionalities for inexperienced developers like us. It never even occurred to us while using amazon or flipkart that simply how much complex an application becomes by even adding a simple new feature, it came to us only while trying to copy it.

The use of various frameworks simply the process somehow but we wanted to develop the website in pure core technologies that's why we have used any frameworks in our projects.

The report contains some small research papers descriptions about various frameworks we read or heard about. The most popular frontend framework React is explained in a lot of research papers and various magazines report.

Why we decided to build an e-commerce application

E-commerce applications are everywhere and it is always a good idea to know how they are made step by step and it can be learned only by trying to make something similar to it.

It was a good experience to make this application as we came to know about a lot of technologies and design patterns we never even heard about earlier.

Why we did not use any frameworks

While it seems widely important to use and know about different frameworks, it is also equally important to know about how you would develop an application with using only the core technologies as under the hood these frameworks simply use the same core technologies.

For ex:

- The React which is a frontend framework is just a sugarcoating on the plain javascript.
- The Node which is a backend framework is also in simple terms a sugarcoating of plain javascript.

So, that's why we decided to not use any frameworks for this project as by using core technologies we can get more control and also get to learn how the application works more properly.

Signature of Student

Name

Date

Signature of Supervisor

Name

Date

\

(VI)

LIST OF FIGURES

Figure No.	Figure Name	Page no.
1	Snippet Of Data	30
2	Snippet Of Data(2)	31
3	Header	32
4	Button	33
5	Default Error Page	33
6	Product Render Page	34
7	Description	35
8	Navbar	36
9	Product	37
10	ProductList.js	38
11	Title	38
12	Cart File Design	40

(VIII)

LIST OF SYMBOLS & ACRONYMS

- **JSX** – JavaScript extension
- **JS** – JavaScript
- **E-commerce** – Electronic commerce
- **B2C** – Business to Customer
- **C2C** – Customer to Customer
- **B2B** – Business to Business
- **Regex** – Regular Expression
- **Pretail** – engaging for pre-launch readiness

CHAPTER - 1

INTRODUCTION

1.1 Intro to e-commerce

Buying or selling of products electronically on the online services over the internet is known as **electronic commerce** or simply **e-commerce**. Various technologies such as supply chain management, mobile commerce, automated data collection systems, electronic-data interchange, transaction processes, funds transfer, marketing over the internet, etc. are completely based on e-commerce technology. E-commerce is in fact, the largest sector of the electronics industries.

Earlier, e-commerce transactions were not so much widely used and were limited to purchasing books online on amazon.com, music albums purchases on various different online music stores(Eg. Itunes by Apple), and was not in regular use by average customers. But now, conditions have changed and e-commerce transactions has flourished, amazon.com extended its product range from books only to everything required in day to day life(not literally, 1 crore products in India only as per the ad shown), various other platforms related to e-commerce has been introduced, and the products range has been widened to include almost everything on earth, including fresh groceries sold by Grofers and BigBasket.

Now, e-commerce is further divided into different areas such as: Online Retail, Online Auctions, Online markets, Used products store, and dozens of more.

E-commerce also includes participation in businesses such as:

- Currency exchanges.
- Business-to-business buying and selling, and data exchanges.
- Business-to-consumer buying and selling, data sharing, online forms, various other sales options, contact and support, feedback and rating systems, etc.
- Consumer-to-consumer data sharing, buying and selling with conversational sharing via live chat options, etc.

- Launching and promotion of latest products in the market.
- Direct retail sales to consumers using mobile apps, webapps, voice-assistants, etc.
- Collecting demographic data through social media platforms.
- Creating new consumers for itself.

Modern e-commerce application uses various third-party or inbuilt packages/software such as:

- Digital Wallets
- Conversional Features
- Automatic generation of product related documents
- e-tickets
- Virtual assistants
- Shopping cart software
- Auto-payment feature
- Third party banking options
- Teleconference
- Loan features
- Instant messaging features
- Various security features
- Secure payment options
- International payment support

Some statistics and key-facts about e-commerce are :-

- As per 2019 statistics, the global e-commerce is amounted to over 3.5 trillion dollars worldwide, that is approximately 14.1% of all of the global retail sales involved online transactions and orders, and this number is obviously expected to grow over the next few years. E-commerce may even capture more than 30% of retail sales over the next five-seven years, and upto 50% in the upcoming ten-twelve years.
- China tops the list of e-commerce markets with sales estimated to be \$700 billion in the year 2019, that is roughly 35% of all retail sales in China and the numbers in the country are rapidly growing faster than any other country. E-commerce in China is expected to cover 50% of all online retail sales globally. Chinese e-commerce giant Alibaba, founded by Jack Ma recorded sales of \$420 bn leaving amazon and ebay behind(which combined turnover is \$90 bn), in 2014. Also, Alibaba recorded more 300 million customers equalling the entire USA population.
- Other than China, the e-commerce market share is largely captured by USA, followed by United Kingdom, Japan, Germany, France, South Korea, Canada, and others.
- Brazil also holds astounding and rapid growth in e-commerce sales. It grew from 10 million people in 2014 to 32 people int 2022, despite large number of internet users base of more than 450 million people but is growing at a very rapid pace.
- E-commerce in India is much lower in India. Despite having a very large population of 135 crore as of 2019, and an internet second highest user base of more than 500 million people as of 2019(first being China), it has relatively very low number of online consumers. Indian e-commerce is dominated by international e-commerce giant Amazon and Flipkart with more than 80% of online retail sales from these sites only. As of 2018, India holded an e-commerce market of roughly \$40 billion and is expected to grow to \$200 billion by 2026.
- As per a report conducted in 2010, United Kingdom holded the highest per capita e-commerce spending in the world with an average investment of \$1000 per person online.
- Czech Republic is the country where almost one – fourth(24%) of total country's turnover is generated via the online e-commerce channel.

- Latest research conducted by large tech and stats gigs indicates that electronic commerce shapes people's manner for shopping.
- In Arab countries such as Egypt, Saudi Arabia and United Arab Emirates, e-commerce industry is evolving at a very high pace. Introduction of Arabic e-commerce websites and mobile apps, with Arabic-language support, e-commerce companies are attracting a lot of people there. The mobile market or so called m-market holds a very large portion of sales in Arab countries. However, despite having a very large internet user base, these countries have relatively low number of online consumers.
- Purchase of goods online rather than offline is soon to become a common trend in Arab, some European, and Western Countries. Consumer packaged goods manufacturers would be expected to use online channels for trade of there products.
- As mentioned above, m-markets or mobile markets had already become much more sold online than offline. UAE has already the highest market retail sale of mobile phones online with 75% of market penetration, with 92% of users available online. Europe has an smartphone penetration of about 65%, with 80% of users being available online. However, the coming future may see a great e-commerce rivalry in these two areas, and both will hol an active e-commerce consumers base.
- Not only the two countries mentioned above, but all of the major countries including India will be an active competitor for largest consumer base in coming 10-12 years. Government of every country will focus at bringing more online consumers to support fair means of payment and bargaining for their people. And the measures taken by the government will help enhance tech, support and society standards of the people.
- Mobile sales or m-commerce made up a total of 25% of all sales done in 2014. Therefore, it has a very high importance in enhancing e-commerce.
- Augmented reality technologies will enhance the e-commerce industry in near future, As of now, various 3D softwares have been made and put to tes by companies and large tech giants like Apple and Facebook. These technoogies will help in testing furnitures, goggles and wall paints without even purchasing them. Lenskart 3D Try on by Lens and glasses company lenskart.com provides augmented reality based try on to test its goggles. Even more companies are getting into augmented reality based features to promote their products.

- Now e-commerce has become a tool to connect two or more businesses or consumers, and is leading the way to more and more easier trade than ever before.
- E-commerce sales have already topped \$1 trillion sales in 2012, and is now rising at exponential rates.
- E-commerce base giants like Amazon, Walmart, Alibaba, etc. are already breaking the sales record every year.
- Even technology giants like Google and Apple are using e-commerce or their sales and this has already lead to Apple being the first ever country to top \$1 trillion mark.
- A recent research states that e-commerce and Infromation Technology(IT) is literally a very good opportunity for countries to develop at a much faster rate than ever before. Companies like Apple, Amazon, Walmart and other e-commerce giants have invested a large sum of money to develop and upgrade their mobile apps, websites, and to improve user experience, so in order to attract more consumers and get more benefits.
- Even non-ecommerce companies are hosting their products online to get enhanced amount of sales and to attreact large number of consumers.
- And finally the best thing about e-commerce is that that it cuts large amount unnecessary cost and bring the product or service to consumer from businesses at a very low price.

1.2 **Problem Statement**

Static websites which we were initially planning to use to build our project was a big hindrance to the project as by using static pages we would have needed to make a lot of more pages and we have to make pages for every action we have allowed on our websites.

Also, some pages like signup or login pages which render dynamic content would be near to impossible to replicate in static pages.

In older times when the websites used to be simple, using static pages has an advantage as inspite of having so many disadvantages, it is super lite which means faster loading of the website and less waiting time.

Today, only a few companies are using static webpages and only the websites who rarely have some updates or even they have some updates it was usually very minor updates use static webpages. And usually the companies that use them are considered very tech-old comapies.

Static means whose state remains same. And static webpages also have the same properties.

The biggest problem of static websites is that the code is highly reduntant in the codebase. So a lot of functionalities which would have been simply reused have to rewritten again and again each time the component is needed.

Also, there will be no scripting that means that it will nearly impossible to add any actions to the websites.

1.3 Significance/Novelty of the problem

First we have not used any static pages for the development of the application. All the webpages contains a lot of dynamic content.

In case of using static only webpages, the problem become a lot bigger when we try to change some properties of the components, also due to redundancy we have to write the same components code again and again. This takes a lot of time which we could have been used to make our website a lot more prettier or faster by thinking about the backend portinn in that time. This also indirectly implies that being a static oly developer takes a lot of time and the experience curve is very flat. Nowadays we can use only bootstrap or similar css libraries to make a good looking website without even knowing about the css part, but in ealier times that would have required deep knowledge of the CSS and HTML part.

Also there would be no server side in a static web application, so no login/signup or form inputs are needed. We can say that the static applications are client side only application.

There are some advantages also but that fails badly against the disadvantages of the static application. So for us a complete static application is not an option. Inspite of all the disadvantages the major one is that the by building an static application we are losing on the advantages of the scripting which is by far the most important of any web application today.

Now all of the above are the disadvantages of the static web-application but there are some advantages also viz:

- Static application are easier to build and hence cheaper.
- Static application since do not have a server it does not need to host a database which takes a lot of server space and result in extra charges.
- Static applications are super light so taht will assist it in a smaller loading time.
- SEOs which is search engines optimizations are needed for heavy websites to keep it at the top of a search result, static websites on the other hand does need a lot of optimization as they are faster to load which is considered a big point in SEO.
- As there are no databases the server side programming which would have been needed is not needed in case of static applications.

Some disadvantages discussed above :

- To update something a lot of redundant code is needed to be written.
- It takes a lot of time to learn to make static only web application so a time taking learning process.
- The content will not look upto date.

Now, we also want to implement some analytics figures like who visited our application, at which time, for what duration etc. All big e-commerce websites to drive their research deeply and to make themselves ready for the future.

The reason why Amazon is so ahead of Flipkart is because of the power of the analytics, they are able to use the data more properly than their rivals.

So that's why we want to use some analytics also in our application.

Also, I don't think anyone has made any e-commerce projects which are super heavy sites using only core-technologies.

1.4 **Brief Description of the Solution Approach**

If we come to the the dynamic pages now than we get a lot of functionalities which earlier we have no access to, the first major advantage is that we have successfully obtained a control on redundancy.

The second major advantage is that by using dynamic application we have opened a new door for the server side programming which earlier we have no access to in case of static application.

Compared to the static websites, the dynamic websites have server-side programming. This means that some of the programming parts are handled by the application server processed by server-side scripting. So the amount of coding a developer has to do will be minimized. Let us go back to the parts of a website to know what this means.

As we have seen, we have four main design components in the layout of a website. Out of which, three (Header, sidebar, and footer) appears on almost all pages of the website. Only the content varies from page to page. So the server stores the information regarding the header, footer, and the sidebar separately in order to access when somebody opens a page. The coding for each page is not saved separately. So when a page is opened, the server side populates all this information to present us with the total layout.

This enables the developer to code once for the common elements and then forget about it. Only the changing part, the content is to be focused on. This saves a lot of time as the need for writing code for separate pages. So the design of the page is saved as a template which is then run for all the pages. In fact, for any page, the server side will go through all the information stored in it to produce the output.

Making a change in the layout

As opposed to the cumbersome process of making a change by coding all pages, dynamic websites make it a bit easier. If a change has to be made in a repeating part, let's say the Header, all the developer has to do is simply edit the part corresponding to the header once. This single edit will make the change appear in all the pages simultaneously. There is no need to edit all pages in which the header appears. This will save a ton of time.

Producing an interactive website

As we told earlier, there is a lot that scripting can do. Server-side scripting enables developers to add features to the website which are not accessible to static websites. This will enable the readers to do more than just read. Web apps also make use of dynamic capabilities to dish out impressive features that enable the users to do more with the websites.

- Changes are easy to make.
- Server-side scripting makes it possible to add interactive features.
- The website will be a lot more functional than a static website.
- Updating the website is an easy affair, even for a tech-noob.
- Takes more time to develop initially.
- More expensive to develop and host.

Many websites still use static websites for static pages like announcement pages, landing pages, coming soon pages. The convenience and stability of static websites have worked well for these applications as they do not require constant editing. Another place where static pages find refuge in these times is simple readable websites that have less number of pages. These websites require faster responses than convenient editing. As static websites are faster than dynamic websites, websites like these make use of that.

On the other page, the dynamic websites provide more room for advanced features. The server-side scripting is updated according to the latest trends. This lets developers make use of the latest trends out in the market to spruce up their website. Live videos, interactive forums, 360-degree media, you name it. Tons of features like this can be added to the website in a jiffy. And for a website that requires constant updating, dynamic websites will provide convenient editing. As the need to edit every single page is eliminated, it is easier to update content. Even for a user with minimal or no coding knowledge, updating the content on a website will be a piece of cake.

The role of Content Management Systems in the Static Vs Dynamic website debate. The popularity of CMS has to lead to most dynamic websites to make use of these systems. Almost all of the dynamic websites out there is managed using a CMS. Exceptions do exist but that is the general case. On the other hand, static websites have no provision to add a CMS into the equation as there is no server-side coding. This will drastically reduce its usability. The massive degree of convenience offered by a CMS is enjoyed by most dynamic users. Usage of a CMS can help in updating content or managing the website.

Using a CMS can also save you a lot of money and time as you don't have to pay your web developer a large sum to produce CMS based websites. On the other hand, creating a static website requires more coding which can lead to inflated web development bills. Though static websites are cheaper to host, many hosting plans available now are affordable, even for dynamic websites.

1.5 Comparison of existing approaches to the problem framed

Angular.js provides a lot of features to make to make dynamic web pages.

Chapter 2: Literature Survey

2.1 Summary of papers studied

Nearly all the papers we studied are about comparisons between React based development viz. React Native which makes a single source file to be viewable on all the web platforms whether on mobile, android, IOS, Linux or windows.

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It's based on React, Facebook's JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly "native," all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.

Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque mark-up, known as JSX. Then, under the hood, the React Native "bridge" invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, *not* web views, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user's location.

React Native currently supports both iOS and Android, and has the potential to expand to future platforms as well. In this book, we'll cover both iOS and Android. The vast majority of the code we write will be cross-platform. And yes: you can

really use React Native to build production-ready mobile applications! Some anecdota: Facebook, Palantir, and TaskRabbit are already using it in production for user-facing applications.

Advantages of React Native

The fact that React Native actually renders using its host platform's standard rendering APIs enables it to stand out from most existing methods of cross-platform application development, like Cordova or Ionic. Existing methods of writing mobile applications using combinations of JavaScript, HTML, and CSS typically render using web views. While this approach can work, it also comes with drawbacks, especially around performance. Additionally, they do not usually have access to the host platform's set of native UI elements. When these frameworks do try to mimic native UI elements, the results usually "feel" just a little off; reverse-engineering all the fine details of things like animations takes an enormous amount of effort, and they can quickly become out of date.

In contrast, React Native actually translates your mark-up to real, native UI elements, leveraging existing means of rendering views on whatever platform you are working with. Additionally, react works separately from the main UI thread, so your application can maintain high performance without sacrificing capability. The update cycle in React Native is the same as in React: when props or state change, React Native re-renders the views. The major difference between React Native and React in the browser is that React Native does this by leveraging the UI libraries of its host platform, rather than using HTML and CSS mark-up.

Risks and Drawbacks

As with anything, using React Native is not without its downsides, and whether or not React Native is a good fit for your team really depends on your individual situation.

The largest risk is probably React Native's maturity, as the project is still relatively young. iOS support was released in March 2015, and Android support was released in September 2015. The documentation certainly has room for improvement, and continues to evolve. Some features on iOS and Android still aren't supported, and the community is still discovering best practices. The good news is that in the vast majority of cases, you can implement support for missing APIs yourself.

Because React Native introduces another layer to your project, it can also make debugging hairier, especially at the intersection of React and the host platform.

React Native is still young, and the usual caveats that go along with working with new technologies apply here. Still, on the whole, I think you'll see that the benefits outweigh the risks.

Summary

React Native is an exciting framework that enables web developers to create robust mobile applications using their existing JavaScript knowledge. It offers faster mobile development, and more efficient code sharing across iOS, Android, and the Web, without sacrificing the end user's experience or application quality. The tradeoff is that it's new, and still a work in progress. If your team can handle the uncertainty that comes with working with a new technology, and wants to develop mobile applications for more than just one platform, you should be looking at React Native.

In the next chapter, we'll go over some of the main ways in which React Native differs from React for the Web, and cover some key concepts.

Chapter 3: Requirement Analysis and Solution Approach

3.1 Overall description of the project and Solution Approach

We made our project using core technologies of web application development only. But we would have used React also in our project.

React can be used as a base in the development of single-page or mobile applications, as it is optimal for fetching rapidly changing data that needs to be recorded. However, fetching data is only the beginning of what happens on a web page, which is why complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API; Redux, React Router and axios are respective examples of such libraries.

3.1.1 Components

React code is made of entities called components. Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass in values that are known as "props"

```
ReactDOM.render(<Greeter greeting="Hello World!" />,  
document.getElementById('myReactApp'));
```

3.1.2 Functional Components

Functional components are declared with a function that then returns some JSX.

```
const Greeting = (props) => <div>Hello, {props.name}!</div>;
```

3.1.3 Class-Based Components

Class-based components are declared using ES6 classes. They are also known as "stateful" components, because their state can hold values throughout the component and can be passed to child components through props:

```
class ParentComponent extends React.Component {  
  state = { color: 'green' };  
  render() {  
    return (  
      <ChildComponent color={this.state.color} />  
    );  
  }  
}
```

3.1.4 Virtual DOM

Another notable feature is the use of a virtual Document Object Model, or virtual DOM. React creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change.

3.1.5 Lifecycle Methods

Lifecycle methods are hooks that allow execution of code at set points during a component's lifetime.

- `shouldComponentUpdate` allows the developer to prevent unnecessary re-rendering of a component by returning false if a render is not required.
- `componentDidMount` is called once the component has "mounted" (the component has been created in the user interface, often by associating it with a DOM node). This is commonly used to trigger data loading from a remote source via an API.
- `componentWillUnmount` is called immediately before the component is torn down or "unmounted". This is commonly used to clear resource demanding dependencies to the component that will not simply be removed with the unmounting of the component (e.g., removing any `setInterval()` instances that

are related to the component, or an "eventListener" set on the "document" because of the presence of the component)

- render is the most important lifecycle method and the only required one in any component. It is usually called every time the component's state is updated, which should be reflected in the user interface.

3.1.6 JSX

JSX, or JavaScript XML, is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Google for PHP called XHP.

An example of JSX code:

```
class App extends React.Component {  
  render() {  
    return (  
      <div>  
        <p>Header</p>  
        <p>Content</p>  
        <p>Footer</p>  
      </div>  
    );  
  }  
}
```

3.1.7 Nested elements

Multiple elements on the same level need to be wrapped in a single container element such as the <div> element shown above, or returned as an array.

3.1.8 Attributes

JSX provides a range of element attributes designed to mirror those provided by HTML. Custom attributes can also be passed to the component. All attributes will be received by the component as props.

3.1.9 JavaScript expressions

JavaScript expressions (but not statements) can be used inside JSX with curly brackets {}:

```
<h1>{10+1}</h1>
```

The example above will render

```
<h1>11</h1>
```

3.1.10 Conditional statements

If-else statements cannot be used inside JSX but conditional expressions can be used instead. The example below will render {i === 1 ? 'true' : 'false'} as the string 'true' because i is equal to 1.

```
class App extends React.Component {  
  render() {  
    const i = 1;  
    return (  
      <div>  
        <h1>{ i === 1 ? 'true' : 'false' }</h1>  
      </div>  
    );  
  }  
}
```

The above will render:

```
<div>  
  <h1>true</h1>  
</div>
```

Functions and JSX can be used in conditionals:

```

class App extends React.Component {
  render() {
    const sections = [1, 2, 3];
    return (
      <div>
        {sections.length > 0 && sections.map(n => (
          /* 'key' is used by react to keep track of list items and their changes */
          /* Each 'key' must be unique */
          <div key={"section-" + n}>Section {n}</div>
        ))}
      </div>
    );
  }
}

```

The above will render:

```

<div>
  <div>Section 1</div>
  <div>Section 2</div>
  <div>Section 3</div>
</div>

```

Code written in JSX requires conversion with a tool such as Babel before it can be understood by web browsers. This processing is generally performed during a software build process before the application is deployed.

Architecture beyond HTML

The basic architecture of React applies beyond rendering HTML in the browser. For example, Facebook has dynamic charts that render to `<canvas>` tags, and Netflix and PayPal use universal loading to render identical HTML on both the server and client.

React Hooks

Hooks are functions that let developers "hook into" React state and lifecycle features from function components. Hooks don't work inside classes — they let you use React without classes.

React provides a few built-in Hooks like `useState`. You can also create your own Hooks to reuse stateful behavior between different components.

React provides a few built-in Hooks like `useEffect`. Which provides lifecycle methods in functional components same as class based components.

React does not attempt to provide a complete "application library". It is designed specifically for building user interfaces and therefore does not include many of the tools some developers might consider necessary to build an application. This allows the choice of whichever libraries the developer prefers to accomplish tasks such as performing network access or local data storage. Common patterns of usage have emerged as the library matures.

4.2 Implementation Details

node start

Runs the app in the development mode.
Open <http://localhost:3000> to view it in the browser.

The page will reload if you make edits.
You will also see any lint errors in the console.

npm test

Launches the test runner in the interactive watch mode.
See the section about [running tests](#) for more information.

npm run build

Builds the app for production to the build folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.
Your app is ready to be deployed!

See the section about deployment for more information.

npm run eject

Note: this is a one-way operation. Once you **eject**, you can't go back!

If you aren't satisfied with the build tool and configuration choices, you can eject at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (Webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except eject will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own.

You don't have to ever use eject. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However, we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.

Data collection

The data we are using is stored in a mongo database which is a noSQL database, we did not use MySQL or relational database as the database structure can be changed in the future which is not easily allowed in case of Relational database.

First we are parsing the data in a variable and then we are traversing or modifying or editing the data according to the need and requirement.

The data will be stored in a object type variable which is supported by the Javascript, which is similar to the JSON file format.

JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.

We can also convert any JSON received from the server into JavaScript objects.

This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

Data pre-processing

We do not need to do any special formatting or pre processing of data as the data used is already in normalized and ready to use form.

Here is a snippet of the data. The data we used is stored in a mongo database. Whose structure will be same. The below snippets depict the data form.

```

export const storeProducts = [
  {
    id: 1,
    title: "Google Pixel - Black",
    //Shubham : { /*Looking for images in public folder*/}
    img: "img/product-1.png",
    price: 10,
    company: "GOOGLE",
    info:
      "Lorem ipsum dolor amet offal butcher quinoa sustainable gastropub, echo park actually greer
    inCart: false,
    count: 0,
    total: 0
  },
  {
    id: 2,
    title: "Samsung S7",
    img: "img/product-2.png",
    price: 16,
    company: "SAMSUNG",
    info:
      "Lorem ipsum dolor amet offal butcher quinoa sustainable gastropub, echo park actually greer
    inCart: false,
    count: 0,
    total: 0
  }
]

```

Figure 1


```

{
  id: 4,
  title: "HTC 10 - White",
  img: "img/product-4.png",
  price: 18,
  company: "htc",
  info:
    "Lorem ipsum dolor amet offal butcher quinoa sustainable gastropub, echo park actu
  inCart: false,
  count: 0,
  total: 0
},
{
  id: 5,
  title: "HTC Desire 626s",
  img: "img/product-5.png",
  price: 24,
  company: "htc",
  info:
    "Lorem ipsum dolor amet offal butcher quinoa sustainable gastropub, echo park actu
  inCart: false,
  count: 0,
  total: 0
},
{

```

Figure 2

We made various components and are attaching the snippets of them here -

header.ejs

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    >

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bo
    integrity="sha384-Vkoo8x4CGs03+Hh xv8T/Q5PaXtkKtu6v
    >

    <!-- FontAwesome CDN -->
    <script src="https://kit.fontawesome.com/364469015
    <title>E-Commerce</title>

    <script type="text/javascript" src="/js/main.js">
  </head>
  <body>

    <!-- NAVIGATION BARR -->

    <nav class="navbar fixed-top navbar-expand-lg
      <a class="navbar-brand" href="/"><i class=
      <button class="navbar-toggler" type="butto
      #navbarSupportedContent" aria-controls="na
      Toggle navigation">
        <span class="navbar-toggler-icon"></sp
      </button>

      <div class="collapse navbar-collapse" id
        <ul class="navbar-nav mr-auto">
          <li class="nav-item active">
            <a class="nav-link" href="/">Home
```

Figure 3

Header.ejs file main motive is to make the header file reusable which is among the main motives of the project. By making the header file separately we can reuse the same functionalities also called generic functionalities in different different files without wirting them again ans again.

Buttons

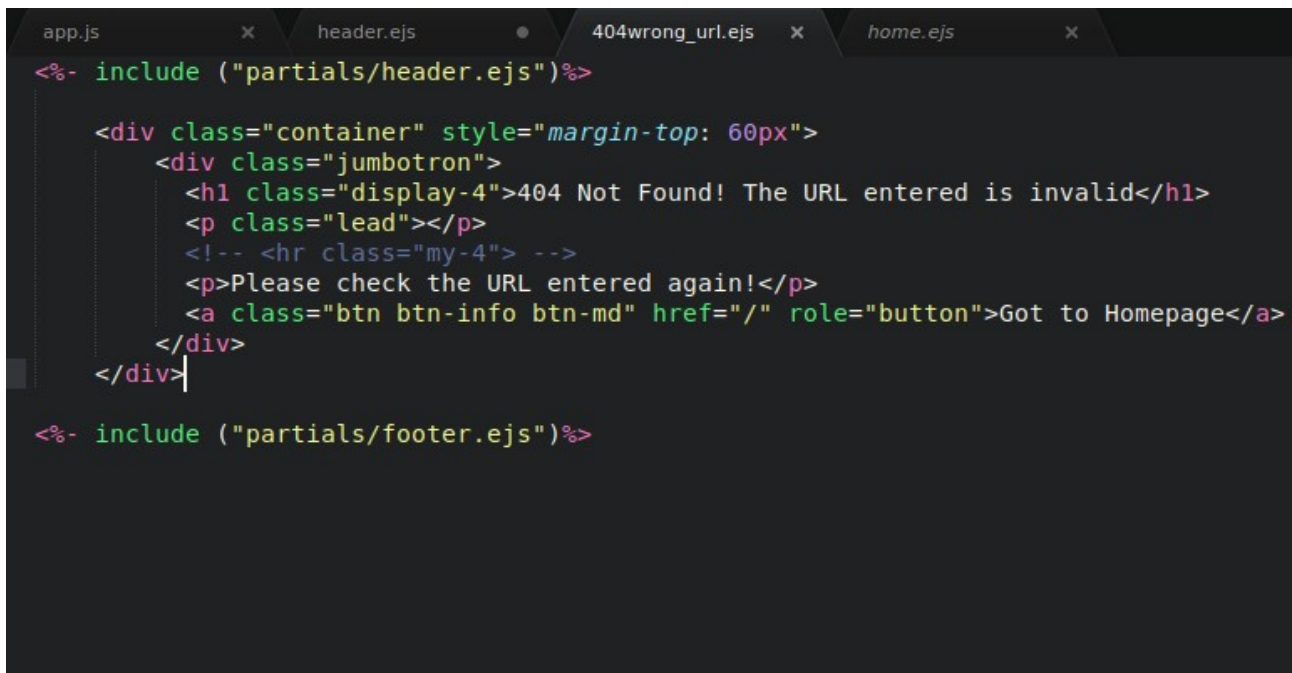
Button main work is to save the redundancy of styling and adding the details from minute to big again and again each time we make a button link. We used bootstrap buttons for now. But the code behind the bootstrap button is somewhat similar to this.

```
export const ButtonContainer = styled.button`
  text-transform: capitalize;
  font-size: 1.4rem;
  background: transparent;
  border: 0.05rem solid var(--lightBlue);
  border-color: ${props =>
    props.cart ? "var(--mainYellow)" : "var(--lightBlue)"};
  color: var(--lightBlue);
  color: ${props => (props.cart ? "var(--mainYellow)" : "var(--lightBlue)"};
  border-radius: 0.5rem;
  padding: 0.2rem 0.5rem;
  outline-color: red;
  cursor: pointer;
  display: inline-block;
  margin: 0.2rem 0.5rem 0.2rem 0;
  transition: all 0.5s ease-in-out;
  /*shubham: & -->>> applies on ButtonContainer only while hovering*/
```

Figure 4

Default 404 redirect file

There should always be a default file for handling wrong URL APIs. The 404.ejs provide the same functionality here. Also a small feature to go back to the main page is also provided.



```
<%- include ("partials/header.ejs")%>

<div class="container" style="margin-top: 60px">
  <div class="jumbotron">
    <h1 class="display-4">404 Not Found! The URL entered is invalid</h1>
    <p class="lead"></p>
    <!-- <hr class="my-4"> -->
    <p>Please check the URL entered again!</p>
    <a class="btn btn-info btn-md" href="/" role="button">Got to Homepage</a>
  </div>
</div>

<%- include ("partials/footer.ejs")%>
```

Figure 5

The product render ejs file

The product render will handle how the products would be displayed on the application handles the detail page information viz. how to render that etc.

This part is handled completely by using dynamic content, as all the data have to be fetched from the database.

As more and more products would be added to the database by the users it would have become impossible to add these products one by one so thats why we used nealry complete dynamic page to handle it.

```

<div class="container">
  <div class="row d-flex text-center">
    <% data.product.forEach(function(product){ %>
      <div class="col-md-4 col-sm-6 my-2 mx-auto card" style="width: 18re
        
          <h5 class="card-title"><%=product.title%></h5>
          <p class="card-text"><%=product.info%></p>
          <a href="#" class="btn btn-primary">Buy Now!</a>
        </div>
      </div>
    <% }) %>
  </div>
</div>

```

Figure 6

Description of the product

On clicking any product a small description about the product provided by the poster of the product will be displayed, also the price and the condition, category etc will also be displayed on the description page. The description page have to be made completely dynamic also like the product display page.

So, we have choosen the ejs format which is embedded Javascript file for creation of the description page also.

It provide some options also like Add to Cart or Go back to main page, etc.

```
app.js x header.ejs 404wrong_url.ejs x home.ejs
<%- include ("partials/header.ejs")%>

<div class="container">
  <div class="row d-flex text-center">

    <div class="col-md-4 col-sm-6 my-2 mx-auto card" styl
      
        <h5 class="card-title"><%-product.title%></h5>
        <p class="card-text"><%-product.info%></p>
        <p class="card-text"><%-product.price%></p>
        <a href="#" class="btn btn-primary">Buy Now!</a>
        <a href="#" class="btn btn-primary">Go Back!</a>
      </div>
    </div>
  </div>
</div>

<%- include ("partials/footer.ejs")%>
```

Figure 7

Navigation Bar

Navbar will provide a always visible navigation panel at the top of the website and contains links to all the possible pages like cart, home, login, signup etc. The navbar is created using bootstrap with CSS and some custom styling.

```

<nav class="navbar fixed-top navbar-expand-lg navbar-dark bg
  <a class="navbar-brand" href="/"><i class="fas fa-hat-co
  <button class="navbar-toggler" type="button" data-toggle
  #navbarSupportedContent" aria-controls="navbarSupportedC
  Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSuppor
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="s
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Products</a>
      </li>
    </ul>
    <ul class="navbar-nav mr-0">

    <!-- NOT LOGED IN -->

    <%if(data.loggedin == 0){%>
      <li class="nav-item active">
        <a class="nav-link" href="/login">Login <spa
      </li>

```

Figure 8

Product.js

Product file will handle all the details of the product into a small component wrapped in a `<div>`. This is the most used component in the website we made.

The product file looks the same as the product render file.

```

<div class="container">
  <div class="row d-flex text-center">
    <% data.product.forEach(function(product){ %>
      <div class="col-md-4 col-sm-6 my-2 mx-auto card" style="width: 18re
        
          <h5 class="card-title"><%-product.title%></h5>
          <p class="card-text"><%-product.info%></p>
          <a href="#" class="btn btn-primary">Buy Now!</a>
        </div>
      </div>
    <% }) %>
  </div>
</div>

```

Figure 9

ProductList.js

Product List file will combine all the products into one page and then render it to the front end.

```

return (
  <React.Fragment>
    <div className="py-5">
      <div className="container">
        <Title name="our" title="products"/>

        <div className="row">
          <ProductConsumer>
            {
              /*Shubham:
              Since we are n consumer we can access all the props of
              producer from here it will be automatically passed as an argument
              */
            }
            {(value)=>{
              console.log(value);
              return value.products.map(
                (product) => {
                  return(
                    <Product key={product.id} product={product}/>
                  );
                }
              );
            }}
          </ProductConsumer>
        </div>
      </div>
    </div>
  </React.Fragment>
)

```

Figure 10

Title part of the website

Title will handle the title and heading of the web page. It is embedded in the header file and will also contain the CDN of various libraries we are using in our project like bootstrap for styling and FontAwesome for displaying the icons.

```
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
  integrity="sha384-Vkoo8x4CGs03+Hhvxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9I"
  >
  <!-- FontAwesome CDN -->
  <script src="https://kit.fontawesome.com/3644690152" class="fa-kit"></script>
  <script>
  <title>E-Commerce</title>

  <script type="text/javascript" src="/js/main.js"></script>
</head>
<body>
```

Figure 11

Cart

Cart is the most important component of an e-commerce website and also the most complex to design as it requires various more components in itself to handle various cases. The cart code structure looks like this. Not the same though as the below code is written in React while we are rebuilding it in Pure JS.









 <code>Cart.js</code>	paypal and cart updated
 <code>CartColumns.js</code>	paypal and cart updated
 <code>CartItem.js</code>	paypal and cart updated
 <code>CartList.js</code>	paypal and cart updated
 <code>CartTotals.js</code>	paypal and cart updated
 <code>EmptyCart.js</code>	paypal and cart updated
 <code>PayPalButton.js</code>	paypal ready
 <code>package.json</code>	paypal and cart updated

Figure 12

It is clear from the snippet above that the cart itself contains and requires many components.

Cart.js

Cart file is handling two states of cart viz. if there are no items in the cart show the empty card page else render the products added in the cart.

All the below cart templates are written in React JS while we are trying to add the same functionalities using Vanilla Javascript.

```

<section>
  <ProductConsumer>
    {value => {
      const { cart } = value;
      if (cart.length > 0) {
        return (
          <React.Fragment>
            <Title name="your" title="cart" />
            <CartColumns />
            <CartList value={value} />
            <CartTotals value={value} history={this.props.history} />
          </React.Fragment>
        );
      } else {
        return <EmptyCart />;
      }
    }}

```

Figure 13

Cart-Columns.js

Cart Columns will show various information in column wise manner in case of large screens. Viz. price, quantity, etc.

```

<div className="row ">
  <div className="col-10 mx-auto col-lg-2">
    <p className="text-uppercase">products</p>
  </div>
  <div className="col-10 mx-auto col-lg-2">
    <p className="text-uppercase">name of product</p>
  </div>
  <div className="col-10 mx-auto col-lg-2">
    <p className="text-uppercase">price</p>
  </div>
  <div className="col-10 mx-auto col-lg-2">
    <p className="text-uppercase">quantity</p>
  </div>
  <div className="col-10 mx-auto col-lg-2">
    <p className="text-uppercase">remove</p>
  </div>

```

Figure 14

Paymet Button

The most rewarding thing for any e-commerce website is sthe payment :) .

We are adding a good payment button in terms of style and functionality for the same.

```
const onError = err => {
  // The main Paypal's script cannot be loaded or somethings block the loading of that script!
  console.log("Error!", err);
  // Because the Paypal's main script is loaded asynchronously from "https://www.paypalobjects.com/api/checkout.js"
  // => sometimes it may take about 0.5 second for everything to get set, or for the button to appear
};

let env = "sandbox"; // you can set here to 'production' for production
let currency = "USD"; // or you can set this value from your props or state
//let total = 1;
// same as above, this is the total amount (based on currency) to be paid by using Paypal express checkout
// Document on Paypal's currency code: https://developer.paypal.com/docs/classic/api/currency_codes/

const client = {
  sandbox: "AQvn3zcBIQF6Nxqp4Y8NPpKxkazrPf6Em1rZynpYuMOVyRe1YTRZm9M5pY6qb0quBvJWrRSVGN2K12d3",
  production: "YOUR-PRODUCTION-APP-ID"
};
```

Figure 15

EmptyCart.js

Handles the state when the cart contains no products.

```
import React from "react";

export default function EmptyCart() {
  return (
    <div className="container mt-5">
      <div className="row">
        <div className="col-10 mx-auto text-center text-title text-capitalize">
          <h1>your cart is currently empty</h1>
        </div>
      </div>
    </div>
  );
}
```

Figure 16

Chapter 6: Conclusions, Future Scope & Results

6.1 Results:

We were able to successfully create the web application.

The Website contains some demo products and their representations also this is how the real products will be displayed.

After the linking from the database the application will work like a real crowdsourced product selling and buying platform.

Home Page:

Welcome to our E - commer

Check out some old products entered by the crowd.

[Learn more](#)

Checkout the products posted by some of th

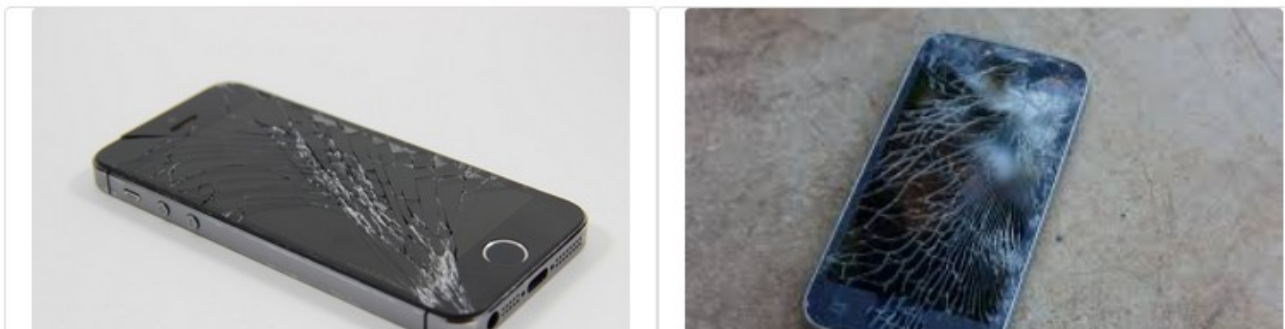


Figure 17

Detail Page:

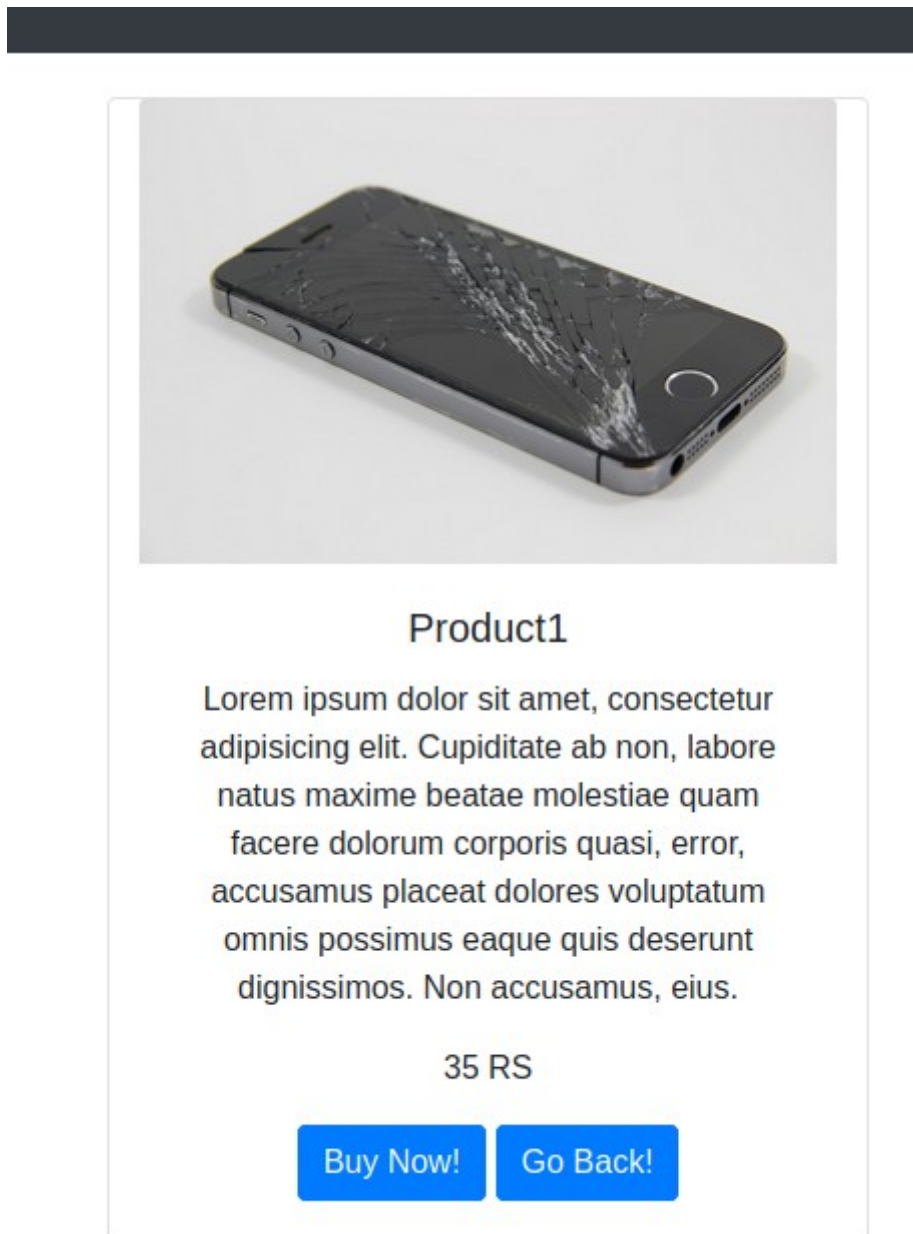


Figure 18

Product Component with the Cart Symbol visible after hovering:

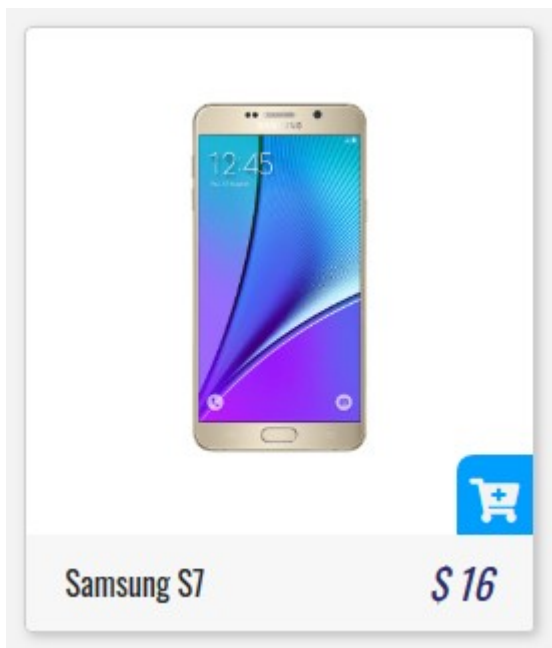


Figure 19

Normal Product Component(with no cart symbol):

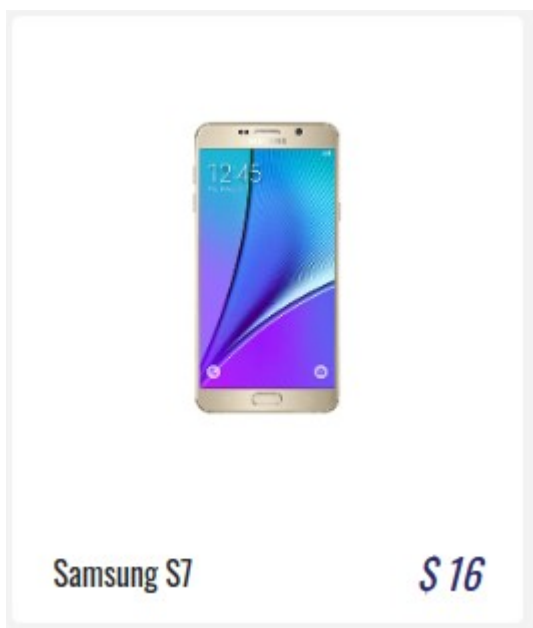


Figure 20

Typical View of the Cart:

Want to add some adjustment on how the cart is looking so the final art may not look similar to this but the structure remains the same.







YOUR CART			
PRODUCTS	NAME OF PRODUCT	PRICE	QUANTITY
	Samsung S7	\$16	- 1 +
	Google Pixel - Black	\$10	- 1 +
	HTC 10 - White	\$18	- 1 +

Figure 21

CART

QUANTITY	REMOVE	TOTAL
- 1 +		Item Total : \$16
- 1 +		Item Total : \$10
- 1 +		Item Total : \$18

CLEAR CART

SUBTOTAL : \$44

TAX : \$4.4

TOTAL : \$48.4

Figure 22

6.2 Conclusions:

The task of making a complete e-commerce application is tiring but equally rewarding, we learned about a lot of things which we did know earlier, also our views regarding the development has also changed, while coding in a compiler based environment it was super easy to find any bugs or errors in our code but in development the compilers are not that good and it sometimes took a lot of time to find a too much trivial bug.

The Dynamic web pages are really powerful as without them we would have no Amazon, no facebook and many other billions of dollars of companies.

We would continue to add new functionalities to the project even after the evaluation ends. As there are lot of improvements which can be done in the application, the use of analytics in our project is also very less, we want to explore more things from the data we collect also.

6.3 Future Scope:

This work provides several interesting future directions. First, we host it on AWS which will result in allowing us to handle a lot more data than currently allowed by using a DB plus various optimization techniques can be explored if we use AWS. Thirdly, if time permits we would like to add a recommendation system with the websites as now-days each websites specially e-commerce are laden with recommendation options so as to maximize their sales and increased profits.

References

- [1] Vipul Kaushik, Kamali Gupta, Deepali Gupta. (2018). React Native Application Development. International Conference on Cyber Security.
- [2] Aymen Beshir: Cross-platform development with React Native . Institutional för information's technology Department of Information Technology No.(2016)
- [3] William Danielsson (2016) React Native Application Development. In: Linköpings universitet
- [4] Faisal A. Amdani, Dr. S. A. Bhura (2019) Cross-Platform Educational Mobile Application Using React Native. In: IOSRJEN
- [5] Alexander Bakker (2012) Comparing Energy Profilers for Android, University of Twente
- [6] Andy Cockburn, Saul Greenberg, Bruce McKenzie, Michael Jasonsmith and Shaun Kaasten (2013) WebView: A Graphical Aid for Revisiting Web Pages In: University of Canterbury
- [7] Erik Johansson, Tobias Anderson (2012) A closer look and comparison of cross-platform development environment for smartphones, Mälardalens University
- [8] Anmol Khandeparkar, Rashmi Gupta (2015) An Introduction to Hybrid Platform Mobile Application Development, Department of Computer Engineering, Mukesh Patel School of Technology

- [9] Angel Torres Moreira (2015) Design and implementation of an Android application to anonymously analyse locations of the citizens in Barcelona, University of Catalonia
- [10] Virpi Roto, Marianna Obrist, Kaisa Väänänen-Vainio-Mattila (2015) User Experience Evaluation Methods in Academic and Industrial Contexts , Nokia Research Center
- [11] Jozef Goetz, Yan Li (2018) Evaluation of Cross-Platform Frameworks for Mobile Applications, IJEIT
- [12] Arnold P.O.S. Vermeeren, Effie Lai-Chong Law, Virpi Roto, Marianna Obrist, Jettie Hoonhout, Kaisa Väänänen-Vainio-Mattila (2015) User Experience Evaluation Methods: Current State and Development Needs, Philips Research Laboratories
- [13] Lide Zhang, Birjodh Tiwana (2015) Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones, University of Michigan
- [14] Andreas Sommer, Stephan Krusche (2015) Evaluation of cross-platform frameworks for mobile applications, in.tum.de
- [15] Maximilian Schirmer, Hagen Höpfner (2012) Software-based Energy Requirement Measurement for Smartphones , Bauhaus-Universität Weimar
- [16] John Elliot, Ah-Lian Kor, and Oluwafemi Ashola Omotosho (2014) Energy Consumption in Smartphones: An Investigation of Battery and Energy Consumption of Media Related Applications on Android Smartphones, Department of Computer Engineering, Mukesh Patel School of Technology

[17] David Abrams, Ron Baecker, Mark Chignell(2014): Information Archiving with Bookmarks: Personal Web Space Construction and Organization, Knowledge Media Design Institute.

[18] Andy Cockburn and Saul Greenberg. Issues of Page Representation and Organization in Web Browser's Re-Visitation Tools, Department of Computer Science, University of Canterbury, Christchurch, New Zealand

[19] Y. Xiao, D. Niyato, P. Wang, and Z. Han, "Dynamic energy trading for wireless powered communication networks," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 158–164, Nov. 2016.

[20] Ron R. Hightower, Laura T. Ring, Jonathan I. Helfman, Benjamin B. Bederson, James D. Hollan : Graphical Multiscale Web Histories: A Study of PadPrints, Computer Science Department