NEW HORIZON COLLEGE OF ENGINEERING

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

A MINI PROJECT REPORT

for

Mini Project in Web Frameworks or Operating System (20CSE68)

on

AMAZON CLONE

Submitted by

R. MANOJKUMAR

USN: 1NH19CS138, Sem-Sec: 6-C

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Academic Year: 2021-22(EVEN SEM)

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

CERTIFICATE

This is to certify that the mini project work titled

AMAZON CLONE

submitted in partial fulfillment of the degree of Bachelor of Engineering in Computer Science and Engineering by

R MANOJKUMAR
USN:1NH19CS138

DURING

EVEN SEMESTER 2021-2022

for

Course: Mini Project in Web Frameworks or Operating System-20CSE68

Signature of Reviewer

Signature of HOD

SEMESTER END EXAMINATION

Name of the Examiner	*Signature with date*
1. _____	_____
2. _____	_____

ABSTRACT

One of the characteristics that may be used to classify a country is its ability to provide an inexpensive and functional . Shopping is now one of India's fastest-growing industries in terms of both employment and income.

In any industry, there is a lot of rivalry. They're always on the hunt for a tried-and-true technique to boost their company's income. Restaurants, retail stores, vape shops - whatever a company wants to offer, if it doesn't have an e-commerce website, it's throwing money away!

The world has shifted online, and businesses must embrace this fact and create a website to handle it. Amazon is a great example of a website that has all of the essential characteristics of a strong e-commerce site.

Amazon's e-commerce website was built using basic HTML, CSS, and JavaScript at first. However, as time passed and new frameworks gained popularity, the website was updated.

We'll learn how to use React and Firebase to create a working clone of Amazon's e-commerce website in this project. The complete software was written in HTML and REACTJS and CSS

The mini-project is entirely built on the high-level language React and the HTML language, and it employs GUI programming to give users with a simple and easy-to-understand platform.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman, New Horizon Educational Institutions, for providing the necessary infrastructure and creating a good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha,** Principal, New Horizon College of Engineering, for his constant support and encouragement.

I would like to thank **Dr. Anandhi R J**, Professor, and Dean-Academics, NHCE, for her valuable guidance.

I would also like to thank **Dr. B. Rajalakshmi**, Professor, and HOD, Department of Computer Science and Engineering, for her constant support.

I also express my gratitude to **Ms. Pramilarani K**, Senior Assistant Professor, Department of Computer Science and Engineering, my mini-project reviewer, for constantly monitoring the development of the project and setting up precise deadlines. Her / His valuable suggestions were the motivating factors in completing the work.

R. MANOJKUMAR

USN: 1NH19CS138

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

There is always competition in any sector. They're constantly looking for a tried-and-true method to increase their company's revenue. Restaurants, retail stores, vape shops, whatever a business offers, if it does not have an e-commerce website, it is wasting money!

The world has migrated online, and businesses must recognise this and design a website to accommodate it. Amazon is an excellent example of a website that possesses all of the necessary components of a strong e-commerce site.

Initially, Amazon's e-commerce website was designed with basic HTML, CSS, and JavaScript. The website was modified as time went and new frameworks gained popularity.

We'll learn how to utilise React and Firebase to construct a functional clone in this project.

1.1.1 OBJECTIVES

Once developed, the application will provide services to

- Register users and maintain a database with their required information
- Allow users to create new account requests.
- Allows users to login and add products to a cart with happy shopping.

Expected OUTCOMES

- Create a header/navigation bar to navigate between pages.
- Create a home page to display products.
- Create a login page for user login.
- Add functionalities like a basket, payment, and authentication.

REQUIREMENTS AND DESIGN

Technologies used

- HTML
- CSS
- JAVASCRIPT
- MYSQL

HARDWARE & SOFTWARE COMPONENTS

- RAM- 1GB OR ABOVE
- HARDDISK- Minimum 20GB free Space
- Processor- Pentium 4(1.6GHZ) or Higher
- Operating system- Windows DXP/2000/Vista/0/7/8/10AS
- Front End- HTML, CSS
- Programming Languages: HTML, CSS, REACTJS, JAVASCRIPT

CHAPTER 2

FUNDAMENTALS OF HTML

2.1 INTRODUCTION TO HTML

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Figure 2.1: Basic example program

2.1.1 HTML ATTRIBUTES

- All HTML elements can have **attributes**

- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value".**

- **1. Absolute URL** - Links to an external image that is hosted on another website. Example: src="https://www.w3schools.com/images/img_girl.jpg".
- **Notes:** External images might be under copyright. If you do not get permission to use it, you may violate copyright laws. In addition, you cannot control external images; they can suddtheyly be removed or changed.
- **2. Relative URL** - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current page. Example: src="img_girl.jpg". If the URL begins with a slash, it will be relative to the domain. Example: src="/images/img_girl.jpg".

- All HTML elements can have **attributes.**
- The href attribute of <a> specifies the URL of the page the link goes to.
- The src attribute of <img> specifies the path to the image to be displayed.
- The width and height attributes of <img> provide size information for images.
- The alt attribute of <img> provides an alternate text for an image.
- The style attribute is used to add styles to an element, such as color, font, size, and more.
- The lang attribute of the <html> tag declares the language of the Web page.
- The title attribute defines some extra information about an element.

2.2 HTML STYLES

- Use the style attribute for styling HTML elements
- Use background-color for background color
- Use color for text colors
- Use font-family for text fonts
- Use font size for text sizes
- Use text-align for text alignment

2.3 HTML CLASS ATTRIBUTE

- The HTML class attribute specifies one or more class names for an element
- Classes are used by CSS and JavaScript to select and access specific elements
- The class attribute can be used on any HTML element
- The class name is case sensitive
- Different HTML elements can point to the same class name
- JavaScript can access elements with a specific class name with the getElementsBy-ClassName() method

```
<!DOCTYPE html>
<html>
<head>
<style>
.note {
  font-size: 120%;
  color: red;
}
</style>
</head>
<body>

<h1>My <span class="note">Important</span> Heading</h1>
<p>This is some <span class="note">important</span> text.</p>

</body>
</html>
```

Figure 2.3: HTML class attribute example

2.4 HTML Iframes

- The HTML <iframe> tag specifies an inline frame
- The src attribute defines the URL of the page to embed
- Always include a title attribute (for screen readers)
- The height and width attributes specify the size of the iframe
- Use border: none; to remove the border around the iframe

2.5 HTML Setting the Viewport

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

You should include the following <meta> element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

2.6 HTML XHTML

What is XHTML?

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is a stricter, more XML-based version of HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

Why XHTML?

XML is a markup language where all documents must be marked up correctly (be "well-formed").

XHTML was developed to make HTML more extensible and flexible to work with other data formats (such as XML). In addition, browsers ignore errors in HTML pages and try to display the website even if it has some errors in the markup. So XHTML comes with a much stricter error handling.

The Most Important Differences from HTML

- <!DOCTYPE> is **mandatory**
- The xmlns attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> are **mandatory**
- Elements must always be **properly nested**
- Elements must always be **closed**
- Elements must always be in **lowercase**
- Attribute names must always be in **lowercase**
- Attribute values must always be **quoted**
- Attribute minimization is **forbidden**

2.7 WORLD WIDE WEB

World Wide Web, which is also known as a Web, is a collection of websites or web pages stored on web servers and connected to local computers through the internet. These websites contain text pages, digital images, audio, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc. The WWW, along with the internet, enables the retrieval and display of text and media to your device. The building blocks of the Web are web paths thick formatted in HTML and connected by links called "hypertext" or hyperlinks and accessed by HTTP. These links are electronic connections that link related pieces of information so that users can access the desired information quickly. Hypertext offers the advantage to select a word or phrase from text and thicknessing's other pages that provide additional information related to that word or phrase. A web page is given an online address called a Uniform Resource Locator (URL). A particular collection of web pages that belong to a specific URL is called a website, e.g., www.facebook.com, www.google.com, etc. So, the World Wide Web is like a huge electronic book whose pages are stored on multiple servers across the world. Small websites store all of their WebPages on a single server, but big websites or organizations place their WebPages on different servers in different countries so that when users of a country search their site they could get the information quickly from the nearest server.

2.8 WEB BROWSERS :

Web Browser is application software that allows us to view and explore information on the web server can request or any a web page by just entering a URL in the to address bar.

Browsers are clients - always initiate, servers react (although sometimes servers require responses)

Mosaic - NCSA (Univ. of Illinois), in early 1993 – First to use a GUI, lead to an explosion of Web use – Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993

- Most requests are for existing documents, using Hypertext Transfer Protocol (HTTP) – But some requests are for program execution, with the output being returned as a document. Browsers can show text, audio, video, animation, and more. It is the responsibility of a web browser to interpret text and commands contained in the web page. Earlier the web browsers were text-based while now day's graphical-based or voice-based web browsers are also available. Following are the most common web browser available today:
- Internet Explorer Microsoft
- Google Chrome Google
- Mozilla Firefox Mozilla
- Netscape Navigator Netscape Communications Corp.
- Opera Software Webb server is a computer where the web content is storekeeper server is used to host the web sites but theorists' other web servers also such as gaming, storage, FTP, email, etc. The website is a collection of web pages while web server is software that responds to the request for web resources.

    - Responses to browser requests, either existing documents or dynamically built documents
    - Browser-server connection is now maintained through more than one request-response cycle

- All communications between browsers and servers use Hypertext Transfer Protocol (HTTP).

2.9 OPERATION OF WWW

WWW works on client-server approach. The following steps explains how the web works:

a. User enters the URL (say, http://www.newhorizonindia.edu) of the web page in the address bar of a web browser.
b. Then the browser requests the Domain Name Server for the IP address corresponding to www.newhorizonindia.edu.
c. After receiving IP address, the browser sends the request a for web page to the web server using HTTP protocol which specifies the way the browser and web server communicate.
d. Then web server receives a request using HTTP protocol and checks its search for the requested web page. If found it returns it to the web browser and closes the HTTP connection.
e. Now the web browser receives the web page, interprets it, and display the contents ofa web page in the web browser's window.

STATIC WEB PAGE: A static web page (sometimes called a flat page or a stationary page) is a web page that is delivered to the user's web browser exactly as stored, in contrast to dynamic web pages which are generated by a web application. Consequently, a static web page displays the same information for all users, from all contexts, subject to the modern capabilities of a web server to negotiate content-type or language of the document where such versions are available and the server is configured to do so. Static web pages are often HTML documents stored as files in the file system and made available by the web server over HTTP (nevertheless URLs ending with ".html" are not always static). However, loose interpretations of the term could include web pages stored in a database, and could even include pages formatted using a template and served through an application server, as long as the page served is unchanging and presented essentially as stored. Static web pages are suitable for content that never or rarely needs to be updated, though modern

web template systems are changing this. Maintaining large numbers of static pages as files can be impractical without automated tools, such as static site generators. Dynamic Web page: Dynamic web page shows different information at different points of time. It is possible to change a portion of a web page without loading the entire web page. It has been made possible using Ajax technology. Server-side dynamic web page: It is created by using server-side scripting. There are server-side scripting parameters that determine how to assemble a new web page which also includes setting up of more client- side processing.

Client-side dynamic web page: It is processed using client-side scripting such as JavaScript. And then passed in to Document Object Model (DOM).

2.10 WEB 2.0

Web 2.0 is the business revolution in the computer industry caused by the move to the internet as a platform, and any attempt to understand the rules for success on that new platform. When it comes to defining web 2.0 the term means such internet applications which allow sharing and collaboration opportunities to people and help them to express themselves online. It's a simply improved version of the first worldwide web, characterized specifically by the change from static to dynamic or user-generated content and also the growth of social media.

Advantages of Web 2.0:

- Available at any time, any place.
- Variety of media.
- Ease of usage.
- Learners can actively be involved in knowledge building.
- Can create dynamic learning communities.
- Everybody is the author and the editor, every edit that has been made can be tracked.
- User-friendly.

- Updates in the wiki are immediate and it offers more sources for researchers.
- It provides real-time discussion.

2.11 Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS was first proposed by HåkonWium Lie on October 10, 1994. At the time, Lie was working with Tim Berners-Lee at CERN. Several other style sheet languages for the web were proposed around the same time, and discussions on public mailing lists and inside World Wide Web Consortium resulted in the first W3C CSS Recommendation (CSS1) being released in 1996. In particular, Bert Bos' proposal was influential; he became co- author of CSS1 and is regarded as a co-creator of CSS.

Style sheets have existed in one form or another since the beginnings of Standard Generalized Markup Language (SGML) in the 1980s, and to provide stylesheets for the web. One requirement for a web style sheet language was for style sheets to come from different sources on the web. Therefore, existing style sheet languages like DSSSL and FOSI were not suitable.

CSS, on the other hand, lets a document's style be influenced by multiple style sheets by way of "cascading" styles. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate.css file, and reduce complexity and repetition in the structural content. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices.CSS also has rules for alternate formatting if the content is accessed on a mobile device. The name cascading comes from the specified priority scheme to determine which

style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS

Inline CSS: Inline CSS contains the CSS property in the body section attached with the element is known as inline CSS. This kind of style is specified within an HTML tag using the style attributes

Internal or Embedded CSS: This can be used when a single HTML document must be styled uniquely The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

External CSS: External CSS contains a separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, etc). CSS property is written in a separate file with .css extension and should be linked to the HTML document using LINK tag. This means that for each element, style can be set only once and that will be applied across web pages.

PROPERTIES OF CSS:

Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. If for an HTML tag, styles are defined in multiple stylesheets then the below order will be followed. ¬ As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles. ¬ Internal or Embedded stands 2 nd in the priority list and overrides the styles in the external style sheet. External style sheets have the last priority. If there are no styles defined either in the line or internal style sheet then external style sheet rules are applied for the HTML tags.

CHAPTER 3

REACT JS

3.1 INTRODUCTION TO REACT

ReactJS is JavaScript library used for building reusable UI components. According to React official documentation, following is the definition −

React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

React Features

- **JSX** − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
- **Components** − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
- **Unidirectional data flow and Flux** − React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
- **License** − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

React Advantages

- Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.

- Can be used on client and server side as well as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.

React Limitations

- Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
- Uses inline templating and JSX, which might seem awkward to some developers.

React versions

The initial version, 0.3.0 of React is released on May, 2013 and the latest version, *17.0.1* is released on October, 2020. The major version introduces breaking changes and the minor version introduces new feature without breaking the existing functionality. Bug fixes are released as and when necessary. React follows the *Sematic Versioning (semver)* principle.

Features

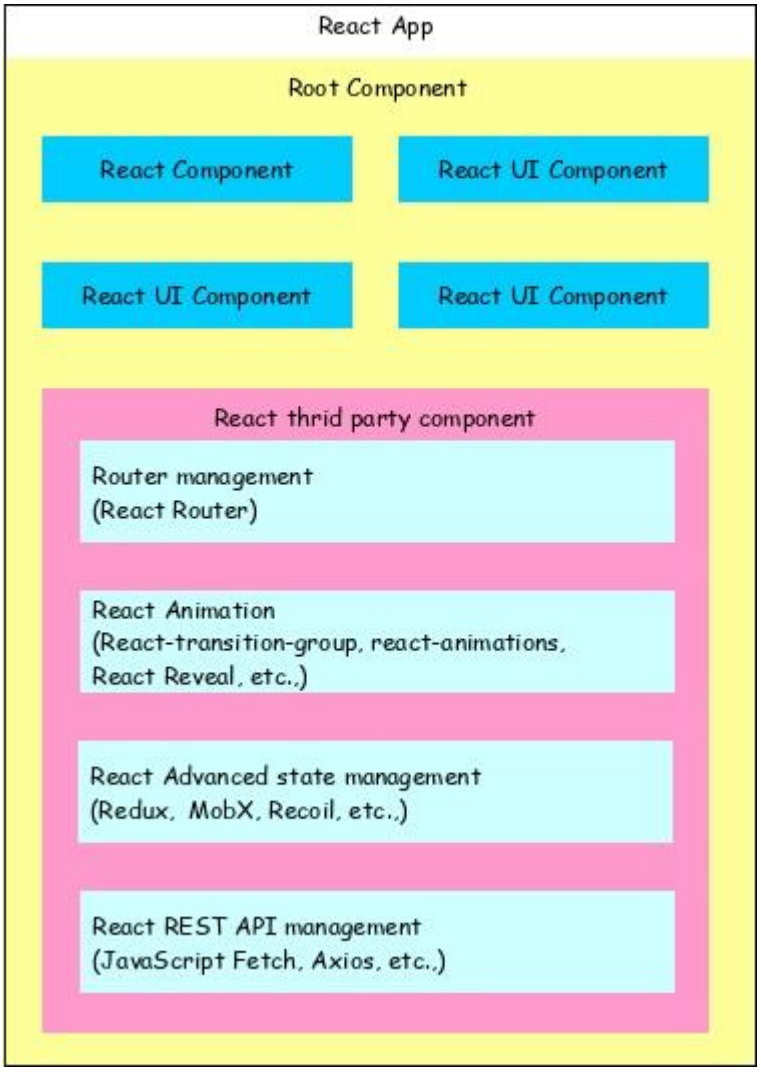
The salient features of *React library* are as follows −

- Solid base architecture
- Extensible architecture
- Component based library
- JSX based design architecture
- Declarative UI library

Benefits

Few benefits of using *React library* are as follows −

- Easy to learn
- Easy to adept in modern as well as legacy application
- Faster way to code a functionality
- Availability of large number of ready-made component
- Large and active community



Applications

Few popular websites powered by *React library* are listed below –

- *Facebook*, popular social media application
- *Instagram*, popular photo sharing application
- *Netflix*, popular media streaming application
- *Code Academy*, popular online training application
- *Reddit*, popular content sharing application

As you see, most popular application in every field is being developed by *React Library*.

3.2 Architecture of the React Application

React library is just UI library and it does not enforce any particular pattern to write a complex application. Developers are free to choose the design pattern of their choice. React community advocates certain design pattern. One of the patterns is Flux pattern. React library also provides lot of concepts like Higher Order component, Context, Render props, Refs etc., to write better code. React Hooks is evolving concept to do state management in big projects. Let us try to understand the high level architecture of a React application.

- React app starts with a single root component.
- Root component is build using one or more component.
- Each component can be nested with other component to any level.
- Composition is one of the core concepts of React library. So, each component is build by composing smaller components instead of inheriting one component from another component.
- Most of the components are user interface components.
- React app can include third party component for specific purpose such as routing, animation, state management, etc.

3.3 REACTJS-ROUTING

In web application, Routing is a process of binding a web URL to a specific resource in the web application. In React, it is binding an URL to a component. React does not support routing natively as it is basically an user interface library. React community provides many third party component to handle routing in the React application. Let us learn React Router, a top choice routing library for React application.

Install React Router

Let us learn how to install *React Router* component in our Expense Manager application.

Open a command prompt and go to the root folder of our application.

```
cd /go/to/expense/manager
```

Install the react router using below command.

```
npm install react-router-dom --save
```

Concept

React router provides four components to manage navigation in React application.

Router − Router is th top level component. It encloses the entire application.

Link − Similar to anchor tag in html. It sets the target url along with reference text.

```
<Link to="/">Home</Link>
```

Here, **to** attribute is used to set the target url.

Switch & Route – Both are used together. Maps the target url to the component. **Switch** is the parent component and **Route** is the child component. **Switch** component can have multiple **Route** component and each **Route** component mapping a particular url to a component.

```
<Switch>
  <Route exact path="/">
    <Home />
  </Route>
  <Route path="/home">
    <Home />
  </Route>
  <Route path="/list">
    <ExpenseEntryItemList />
  </Route>
</Switch>
```

Here, **path** attribute is used to match the url. Basically, **Switch** works similar to traditional switch statement in a programming language. It matches the target url with each child route (**path** attribute) one by one in sequence and invoke the first matched route.

Along with router component, React router provides option to get set and get dynamic information from the url. For example, in an article website, the url may have article type attached to it and the article type needs to be dynamically extracted and has to be used to fetch the specific type of articles.

```
<Link to="/article/c">C Programming</Link>
<Link to="/article/java">Java Programming</Link>

...
```

```
...

<Switch>
  <Route path="article/:tag" children={<ArticleList />} />
</Switch>
```

Then, in the child component (class component),

```
import { withRouter } from "react-router"

class ArticleList extends React.Component {
  ...
  ...
  static getDerivedStateFromProps(props, state) {
    let newState = {
      tag: props.match.params.tag
    }
    return newState;
  }
  ...
  ...
}
export default WithRouter(ArticleList)
```

Here, **WithRouter** enables **ArticleList** component to access the tag information through **props**.

The same can be done differently in functional components −

```
function ArticleList() {
  let { tag } = useParams();
  return (
    <div>
      <h3>ID: {id}</h3>
```

```
    </div>
  );
}
```

Here, **useParams** is a custom React Hooks provided by React Router component.

3.4 Nested routing

React router supports nested routing as well. React router provides another React Hooks, **useRouteMatch()** to extract parent route information in nested routes.

```
function ArticleList() {
  // get the parent url and the matched path
  let { path, url } = useRouteMatch();

  return (
    <div>
      <h2>Articles</h2>
      <ul>
        <li>
          <Link to={`${url}/pointer`}>C with pointer</Link>
        </li>
        <li>
          <Link to={`${url}/basics`}>C basics</Link>
        </li>
      </ul>

      <Switch>
        <Route exact path={path}>
          <h3>Please select an article.</h3>
        </Route>
        <Route path={`${path}/:article`}>
```

```
          <Article />
        </Route>
      </Switch>
    </div>
  );
}
function Article() {
  let { article } = useParams();
  return (
    <div>
      <h3>The select article is {article}</h3>
    </div>
  );
}
```

Here, **useRouteMatch** returns the matched path and the target **url**. url can be used to create next level of links and **path** can be used to map next level of components / screens.

Creating navigation

Let us learn how to do routing by creating the possible routing in our expense manager application. The minimum screens of the application are given below −

- **Home screen** − Landing or initial screen of the application
- **Expense list screen** − Shows the expense items in a tabular format
- **Expense add screen** − Add interface to add an expense item

First, create a new react application, *react-router-app* using *Create React App* or *Rollup* bundler by following instruction in *Creating a React application* chapter.

Next, open the application in your favorite editor.

Next, create *src* folder under the root directory of the application.

Next, create *components* folder under src folder.

Next, create a file, *Home.js* under *src/components* folder and start editing.

Next, import *React library*.

```
import React from 'react';
```

Next, import **Link** from React router library.

```
import { Link } from 'react-router-dom'
```

Next, create a class, Home and call constructor with **props**.

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


Let us learn how to do production build and deployment of React application in this chapter.

3.5 Building

Once a React application development is done, application needs to be bundled and deployed to a production server. Let us learn the command available to build and deploy the application in this chapter.

A single command is enough to create a production build of the application.

```
npm run build
> expense-manager@0.1.0 build path\to\expense-manager
> react-scripts build

Creating an optimized production build...
Compiled with warnings.

File sizes after gzip:
```

```
41.69 KB build\static\js\2.a164da11.chunk.js
2.24 KB build\static\js\main.de70a883.chunk.js
1.4 KB build\static\js\3.d8a9fc85.chunk.js
1.17 KB build\static\js\runtime-main.560bee6e.js
493 B build\static\css\main.e75e7bbe.chunk.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

npm install -g serve
serve -s build

Find out more about deployment here:

https://cra.link/deployment
```

Once the application is build, the application is available under *build/static* folder.

By default, *profiling* option is disable and can be enabled through *–profile* command line option. *–profile* will include profiling information in the code. The profiling information can be used along with React DevTools to analyse the application.

```
npm run build -- --profile
```

3.6 Deployment

Once the application is build, it can be deployed to any web server. Let us learn how to deploy a React application in this chapter.

Local deployment

Local deployment can be done using serve package. Let us first install serve package using below command −

```
npm install -g server
```

To start the application using serve, use the below command −

```
cd /go/to/app/root/folder
serve -s build
```

By default, serve serve the application using port 5000. The application can be viewed *@ http://localhost:5000*.

Production deployment

Production deployment can be easily done by copying the files under build/static folder to the production application's root directory. It will work in all web server including Apache, IIS, Nginx, etc.

Relative path

By default, the production build is created assuming that the application will be hosted in the root folder of a web application. If the application needs to be hosted in a subfolder, then use below configuration in the package.json and then build the application.

```
{ ... "homepage": "http://domainname.com/path/to/subfolder", ... }
```

CHAPTER 4

ALGORITHM

1. Create HTML and CSS files, using google forms and add logos.
2. Import necessary library modules and library package installations.
3. As node modules are necessary install them first, later react libraries.
4. Create HOME.JS file & insert and import the necessary images and css files.
5. Style the major causes section and check the model file for the project.
6. Create HEADER.JS, CART.JS, ORDERS.JS, INDEX.JS, PAYMENT.JS, etc., JS files and import necessary .CSS files.
7. Start executing the files that are created as giving the command in terminal “npm start”.
8. As it directly redirects to the web browser and displays the content what you have given in the coding section.
9. As it displays the information about our website and helps us to raise orders to the user.

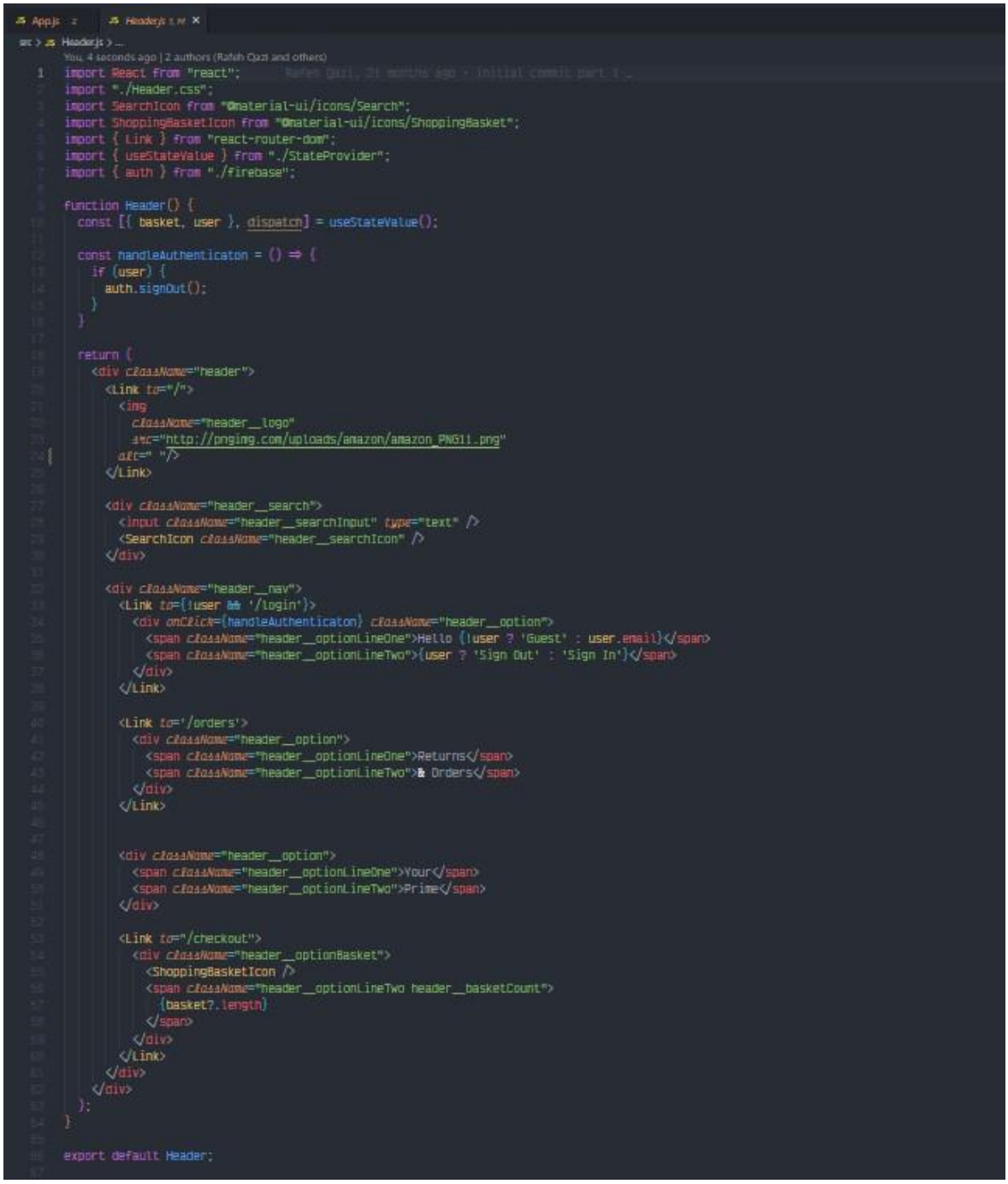
CHAPTER 5

CODING SECTION

5.1 APP.JS, HOME.JS AND HEADER.JS

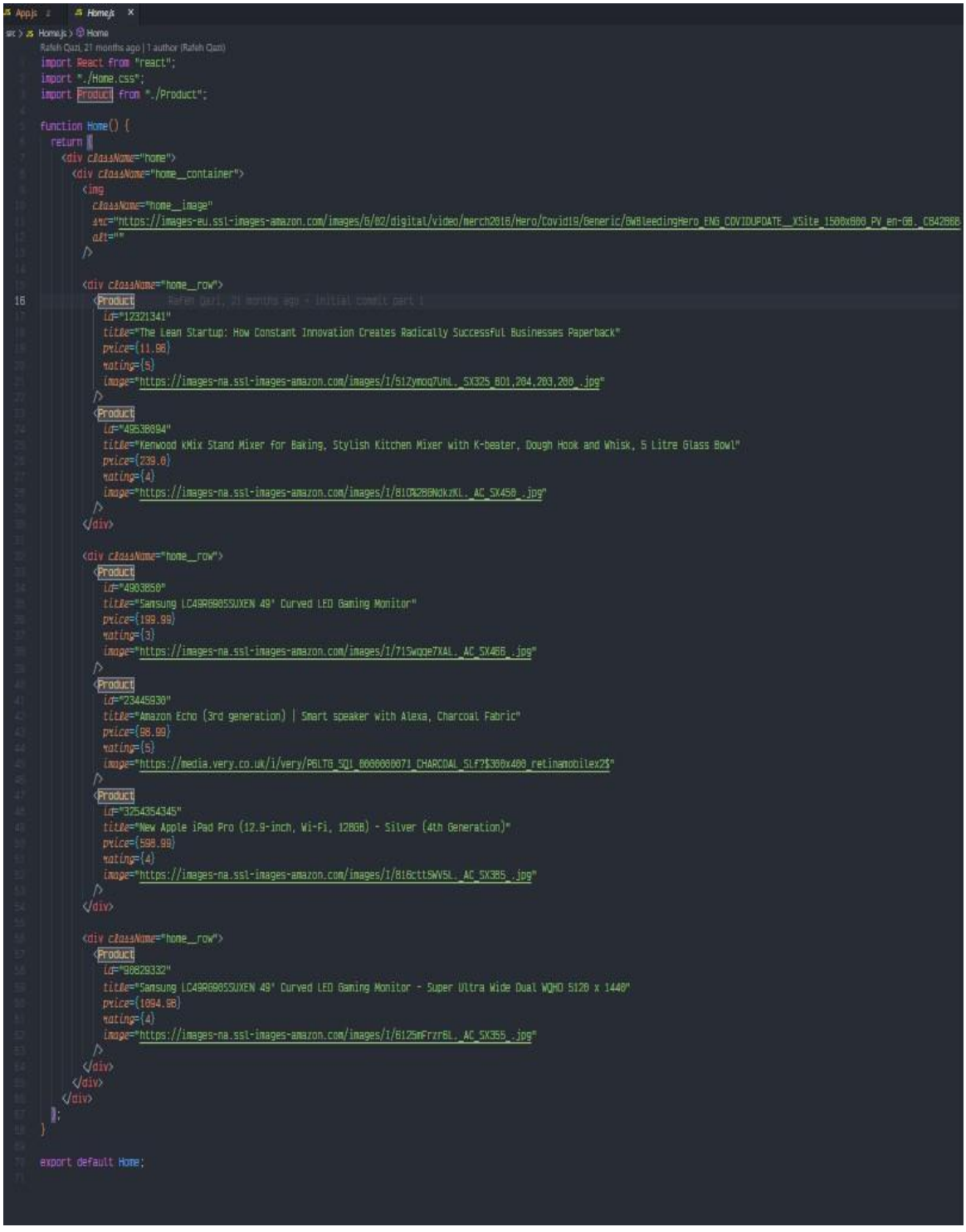
```
src > App.js > App
Rafeh Qazi, 21 months ago | 1 author (Rafeh Qazi)
import React, { useEffect } from "react";
import "./App.css";
import Header from "./Header";
import Home from "./Home";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import Checkout from "./Checkout";
import Login from "./Login";
import Payment from "./Payment";
import Orders from "./Orders";
import { auth } from "./firebase";
import { useStateValue } from "./StateProvider";
import { loadStripe } from "@stripe/stripe-js";
import { Elements } from "@stripe/react-stripe-js";

const promise = loadStripe(
  "pk_test_51HPvU8DFg5koCdL6JJbNo60QAU98BejacsvnKvTBxnCu1wFLCuQP3WBArscK3RvSQm5IB3NBPbsc7TtbQiJ1vaDi00X9sIbazL"
);

function App() {
  const [{}, dispatch] = useStateValue();

  useEffect(() => {
    // will only run once when the app component loads...

    auth.onAuthStateChanged((authUser) => {
      console.log("THE USER IS >>> ", authUser);

      if (authUser) {
        // the user just logged in / the user was logged in

        dispatch({
          type: "SET_USER",
          user: authUser,
        });
      } else {
        // the user is logged out
        dispatch({
          type: "SET_USER",
          user: null,
        });
      }
    });
  }, []);

  return (
    <Router>
      <div className="app">
        <Switch>
          <Route path="/orders">
            <Header />
            <Orders />
          </Route>
          <Route path="/login">
            <Login />        Rafeh Qazi, 21 months ago • day 2 build
          </Route>
          <Route path="/checkout">
            <Header />
            <Checkout />
          </Route>
          <Route path="/payment">
            <Header />
            <Elements stripe={promise}>
              <Payment />
            </Elements>
          </Route>
          <Route path="/">
            <Header />
            <Home />
          </Route>
        </Switch>
      </div>
    </Router>
  );
}

export default App;
```

APP.JS FILE



```
App.js 2   Header.js 1, M ×
src > Header.js > ...
You, 4 seconds ago | 2 authors (Rafah Qazi and others)
import React from "react";
import "./Header.css";
import SearchIcon from "@material-ui/icons/Search";
import ShoppingBasketIcon from "@material-ui/icons/ShoppingBasket";
import { Link } from "react-router-dom";
import { useStateValue } from "./StateProvider";
import { auth } from "./firebase";

function Header() {
  const [{ basket, user }, dispatch] = useStateValue();

  const handleAuthenticaton = () => {
    if (user) {
      auth.signOut();
    }
  }

  return (
    <div className="header">
      <Link to="/">
        <img
          className="header__logo"
          src="http://pngimg.com/uploads/amazon/amazon_PNG11.png"
        alt="" />
      </Link>

      <div className="header__search">
        <input className="header__searchInput" type="text" />
        <SearchIcon className="header__searchIcon" />
      </div>

      <div className="header__nav">
        <Link to={!user && '/login'}>
          <div onClick={handleAuthenticaton} className="header__option">
            <span className="header__optionLineOne">Hello {!user ? 'Guest' : user.email}</span>
            <span className="header__optionLineTwo">{user ? 'Sign Out' : 'Sign In'}</span>
          </div>
        </Link>

        <Link to='/orders'>
          <div className="header__option">
            <span className="header__optionLineOne">Returns</span>
            <span className="header__optionLineTwo">& Orders</span>
          </div>
        </Link>


        <div className="header__option">
          <span className="header__optionLineOne">Your</span>
          <span className="header__optionLineTwo">Prime</span>
        </div>

        <Link to="/checkout">
          <div className="header__optionBasket">
            <ShoppingBasketIcon />
            <span className="header__optionLineTwo header__basketCount">
              {basket?.length}
            </span>
          </div>
        </Link>
      </div>
    </div>
  );
}

export default Header;
```

HEADER.JS FILE



```
App.js 2   Home.js ×
src > Home.js > Home
Rafeh Qazi, 21 months ago | 1 author (Rafeh Qazi)
import React from "react";
import "./Home.css";
import Product from "./Product";

function Home() {
  return (
    <div className="home">
      <div className="home__container">
        <img
          classNAme="home__image"
          src="https://images-eu.ssl-images-amazon.com/images/G/02/digital/video/merch2016/Hero/Covid19/Generic/GWBleedingHero_ENG_COVIDUPDATE__XSite_1500x600_PV_en-GB._CB428686
          alt=""
        />

        <div classNAme="home__row">
          <Product
            id="12321341"
            title="The Lean Startup: How Constant Innovation Creates Radically Successful Businesses Paperback"
            price={11.96}
            rating={5}
            image="https://images-na.ssl-images-amazon.com/images/I/51Zymoq7UnL._SX325_BO1,204,203,200_.jpg"
          />
          <Product
            id="49538094"
            title="Kenwood kMix Stand Mixer for Baking, Stylish Kitchen Mixer with K-beater, Dough Hook and Whisk, 5 Litre Glass Bowl"
            price={239.0}
            rating={4}
            image="https://images-na.ssl-images-amazon.com/images/I/81O%2BGNdkzKL._AC_SX450_.jpg"
          />
        </div>

        <div classNAme="home__row">
          <Product
            id="4903850"
            title="Samsung LC49RG80SSUXEN 49" Curved LED Gaming Monitor"
            price={199.99}
            rating={3}
            image="https://images-na.ssl-images-amazon.com/images/I/71Swqqe7XAL._AC_SX466_.jpg"
          />
          <Product
            id="23445930"
            title="Amazon Echo (3rd generation) | Smart speaker with Alexa, Charcoal Fabric"
            price={98.99}
            rating={5}
            image="https://media.very.co.uk/i/very/P6LTG_SQ1_0000000071_CHARCOAL_SLF?$300x400_retinamobilex2$"
          />
          <Product
            id="3254354345"
            title="New Apple iPad Pro (12.9-inch, Wi-Fi, 128GB) - Silver (4th Generation)"
            price={598.99}
            rating={4}
            image="https://images-na.ssl-images-amazon.com/images/I/816ctt5WV5L._AC_SX385_.jpg"
          />
        </div>

        <div classNAme="home__row">
          <Product
            id="90829332"
            title="Samsung LC49RG80SSUXEN 49" Curved LED Gaming Monitor - Super Ultra Wide Dual WQHD 5120 x 1440"
            price={1094.98}
            rating={4}
            image="https://images-na.ssl-images-amazon.com/images/I/6125mFrzr6L._AC_SX355_.jpg"
          />
        </div>
      </div>
    </div>
  );
}

export default Home;
```

HOME.JS FILE

CHAPTER 6

RESULTS

6.1 Launch Screen

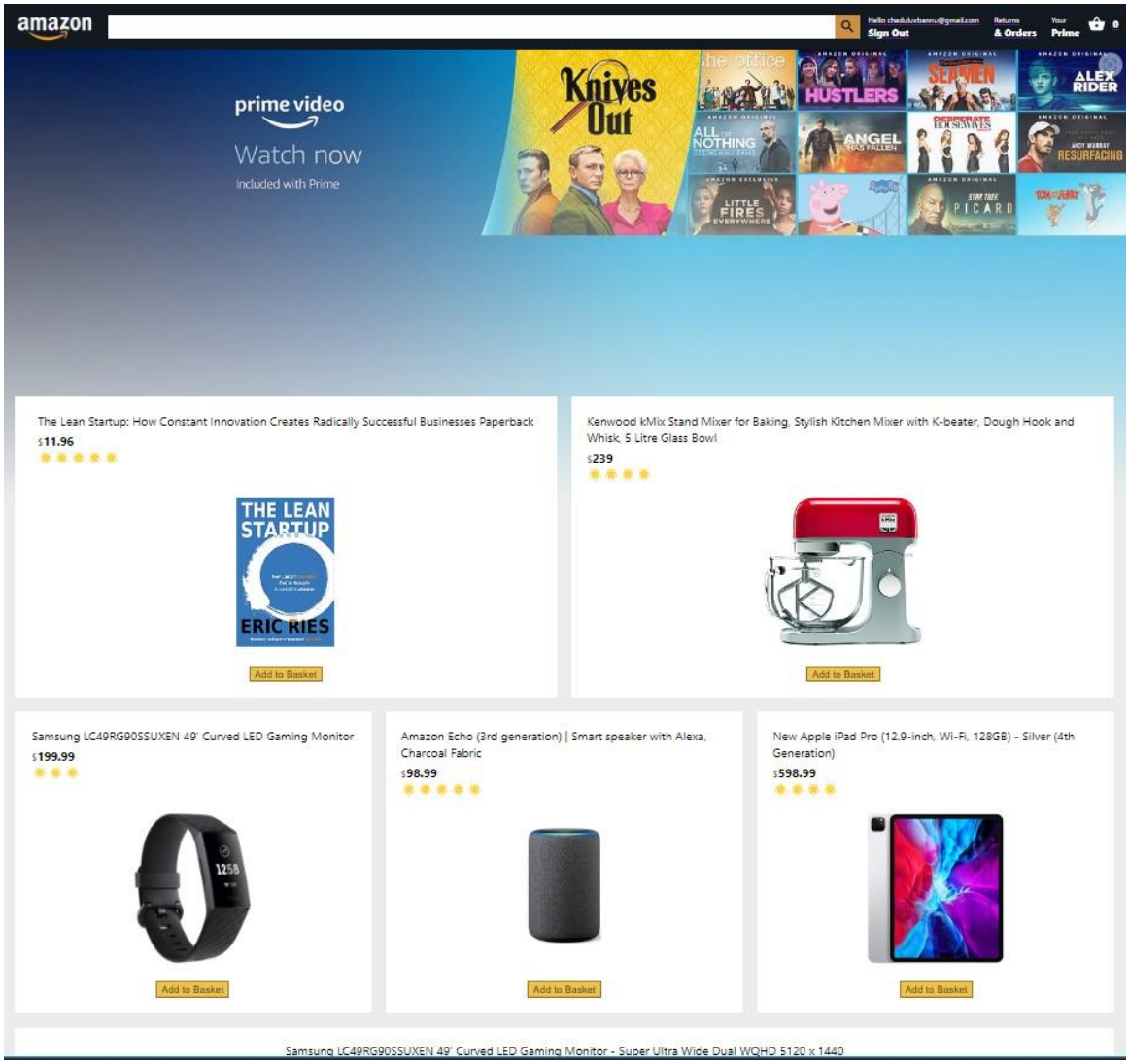


FIG 6.1 LAUNCH SCREEN

6.2 CART PAGE

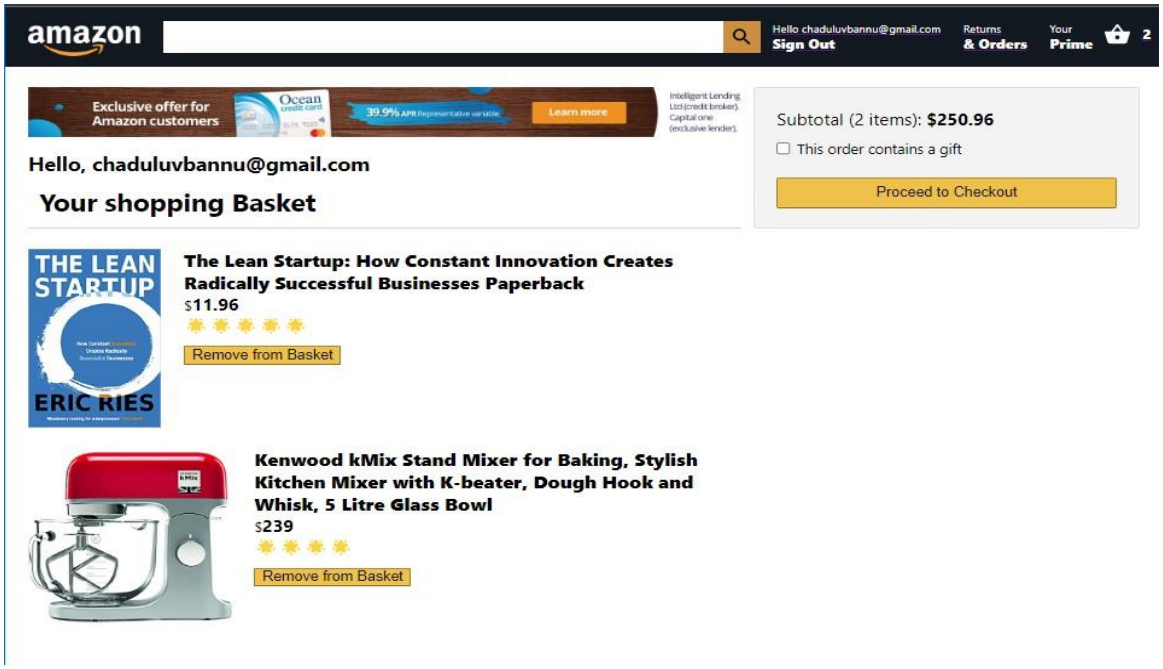


FIG 6.2 UPDATED CART PAGE

6.3 CHECKOUT PAGE

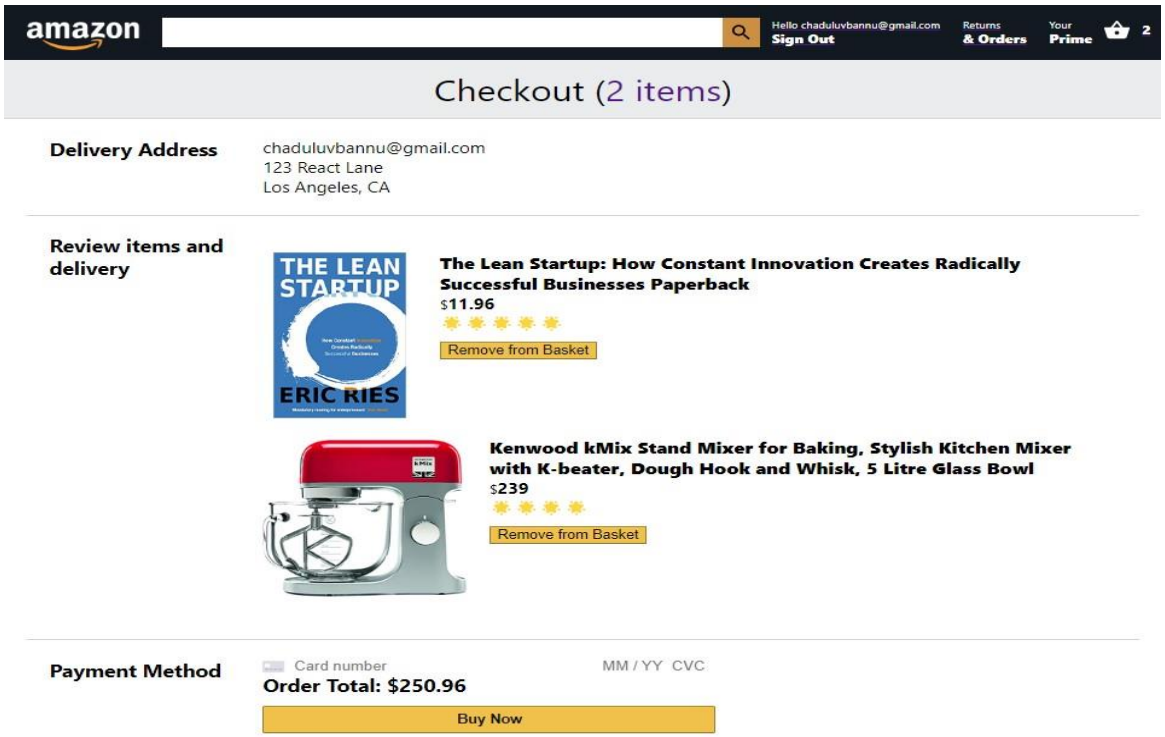


FIG 6.3 CHECKOUT PAGE

CHAPTER 7

CONCLUSION

The world has migrated online, and businesses must recognise this and design a website to accommodate it. Amazon is an excellent example of a website that possesses all of the necessary features of a successful e-commerce business.

Initially, Amazon's e-commerce website was constructed with simple HTML, CSS, and JavaScript. The website was modified as time went and new frameworks gained popularity.

In this project, we'll learn how to utilise React and Firebase to develop a functioning clone of Amazon's e-commerce website. The entire software was developed in HTML, REACTJS, and CSS.

The mini-project is totally based on the high-level language React and the HTML language, with GUI programming used to provide users with a basic and easy-to-understand platform.

The world has shifted online, and businesses must acknowledge this and create a website to suit it. Amazon is a great example of a website that has all of the key components of a successful e-commerce site.

Amazon's e-commerce website was first built using simple HTML, CSS, and JavaScript. The website evolved over time as new frameworks gained popularity.

In this project, we'll learn how to use React and Firebase to build a working clone.

REFERENCES

[1] HTML Tutorial (w3schools.com)

[2] https://www.crio.do/projects/

[3] A.Conci, J. E. R. de Carvalho, T. W. Rauber, A Complete System for Vehicle Plate Localization, Segmentation and Recognition in Real Life Scene, IEEE LATIN AMERICA TRANSACTIONS, VOL. 7, NO. 5, September 2009. (Example for paper referred)

[4] Joseph Yiu, The Definitive Guide to ARM Cortex-M3 and Cortex M4 Processor, 3rd Edition, Newness Publication (example for book referred)