A REPORT OF SIX WEEK TRAINING (TR-102)

at

Oops InfoSolutions Pvt. Ltd.

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer science and Engineering)

JUNE-JULY, 2025

SUBMITTED BY: -

Name: Roshan Kumar
UNIVERSITY ROLL NO.: 2302654

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

CERTIFICATE BY OOPS INFOSOLUTIONS PVT. LTD.

Oops InfoSolutions Pvt. Ltd.
Regd. Under Ministry of Corporate Affairs GOVT. OF INDIA

Ref No: CGFP-6705-2025 Date: 26/07/2025

Certificate of Completion

This is to certify that Mr**. Roshan Kumar** student of B.Tech (CSE) , Rollno:- 2302654, Guru Nanak Dev Engineering College, Ludhiana has successfully completed industrial training in Web Development from 23rd June 2025 to 24th July 2025. During the training he is trained under the guidance of Mr. Manjit Singh. His overall performance during the training is Excellent.

We recommend Roshan Kumar for his outstanding performance and the skills he has developed during this period. We believe these experiences will significantly contribute to his future endeavors. We extend our best wishes for Roshan Kumar continued success and professional growth.

Manjit Singh
(Director)

S.C.O. 80-81-82, 2nd Floor,
Sector 34-A, Chandigarh
Ph : 0172-5009244, Mobile : 9855222244

Website : www.oopsinfosolution.com
e-mail : info@oopsinfosolution.com

CANDIDATE'S DECLARATION

I, **Roshan Kumar**, hereby declare that I have undertaken four week training "**WEB DEVELOPMENT**" during the period from **23th June 2025** to **24th August 2025** in partial fulfillment of requirements for the award of the degree of **B.Tech. (Computer Science and Engineering)** at **Guru Nanak Dev Engineering College, Ludhiana.** The work presented in this training report, submitted to the Department of Computer Science and Engineering at Guru Nanak Dev Engineering College, Ludhiana, is an authentic record of the training work.

Signature of Student

The four week industrial training Viva–Voce Examination of_____ has been held on _____ and accepted.

Signature of Internal Examiner Signature of External Examiner

ABSTRACT

This report provides an overview of the four-week industrial training program undertaken on **Web Development**, which served as a crucial bridge between theoretical learning and practical implementation. The training aimed to enhance my understanding of the key concepts, tools, and technologies used in developing modern and interactive web applications. Through this program, I gained valuable hands-on experience in designing and building responsive user interfaces, improving usability, and implementing effective **UI/UX design principles**.

The training primarily focused on mastering core frontend technologies such as **HTML**, **CSS**, and **JavaScript**, which are fundamental for creating visually appealing and dynamic web pages. In addition to these, I also learned how to integrate interactivity, manage layouts effectively, and ensure that web applications are optimized for different screen sizes and devices. Emphasis was placed on creating a seamless user experience, where aesthetics and functionality work together to engage the user and improve accessibility.

Throughout the duration of the training, I worked on various practical tasks and mini projects that helped in strengthening my understanding of real-world frontend development challenges. These activities included creating structured webpage layouts, styling with advanced CSS features, adding interactivity through JavaScript functions, and applying responsive design techniques. The training also provided exposure to debugging and optimization, which are essential skills for developing efficient web solutions.

ACKNOWLEDGEMENT

I would like to sincerely thank my mentor, **Ms. Manjit Singh**, for his constant guidance, encouragement, and valuable insights throughout my **FRONTEND** training. His support not only deepened my understanding of the subject but also motivated me to apply my learning in practical ways. I am also grateful to **Guru Nanak Dev Engineering College** and the **Department of Computer Science and Engineering** for giving me the opportunity to take up this training and for their continuous support in my academic journey. This experience has helped me gain a solid understanding of **HTML, CSS and JavaScript**. The hands-on projects and assignments were especially helpful in boosting my confidence and preparing me for real-world challenges.

Finally, I would like to extend my heartfelt thanks to everyone who, directly or indirectly, contributed to the successful completion of this training. I am excited to carry forward the skills and knowledge I have gained and apply them in my future endeavors.

ABOUT OOPS INFOSOLUTIONS PVT. LTD.

OOPS Info Solutions Pvt. Ltd. was established in 2003 as an India-based IT service provider offering solutions in technology infrastructure, enterprise resource planning, software application engineering, and professional training. The company operates from Chandigarh and Mohali and has consistently expanded its client base since inception.

Founded by **Manjit Singh**, an ex-Infosys employee with over 15 years of experience in IT, Java, J2EE, PHP, Android, and Networking, the company has been featured in renowned publications such as *Computer Times*, *IT India*, and *Computer World*. In 2011, OOPS Info Solutions was nominated for the **"Spirit of Enterprise"** award for promoting entrepreneurial excellence.

OOPS Info Solutions is an **ISO Certified Company** and a registered member of **Professional Training and Software Development** organizations. The company specializes in **mobile app design and development**, as well as website integration services.

The primary goal of OOPS Info Solutions is to provide efficient and reliable software solutions that enhance productivity and performance. Through continuous innovation and a commitment to quality, the company ensures that its clients benefit from the latest advancements in technology.

List of Figures

List of Tables

Contents

CHAPTER 1

INTRODUCTION

1.1 Basic Introduction:

Front-end development focuses on the part of a website or web application that users directly interact with. It involves creating the layout, design, and interactive features that make a website functional and visually appealing. Front-end developers use foundational technologies such as **HTML, CSS, and JavaScript** to build user interfaces that are both responsive and user-friendly.

Through this training, I gained a practical understanding of how these core front-end technologies work together — how **HTML structures content**, **CSS styles the layout**, and **JavaScript adds interactivity**. This knowledge helped me understand the core principles behind developing modern, responsive web interfaces.

1.1.1 Key Learning

1. **HTML:** Learned to structure web pages using semantic elements and forms.
2. **CSS:** Gained knowledge in designing responsive layouts using Flexbox and Grid.
3. **JavaScript:** Understood core concepts such as DOM manipulation, ES6 features, and event handling.
4. **GitHub:** Learned how to use Git commands for version control and collaborate on projects.

1.1.2 Background

- **HTML (Hypertext Markup Language) :** It provided a simple way to structure text and add links to other documents, forming the very backbone of a webpage. Early websites were purely informational, like digital brochures with basic text and links.

- **CSS (Cascading Style Sheets):** As the web grew, so did the desire to make it look better. This was a game-changer, allowing for more creative layouts, colours, and fonts. The first version of CSS was officially released in **1996**.

- **JavaScript:** This scripting language brought interactivity to websites, enabling things like pop-up alerts, form validations, and dynamic content changes without needing to reload the entire page.

1.2 Objectives:

The main objective of using **HTML, CSS, and JavaScript** for front-end development is to create well-structured, visually engaging, and highly interactive user experiences directly in the web browser. These three languages are the essential pillars that allow a developer to create any functional website.

1.2.1 Establish Clear and Semantic Structure (HTML)

The primary goal of HTML is to provide the **content** and **structure** of a webpage.

- **Logical Organization:** Use semantic elements (like <header>, <main>, <footer>, <article>) to ensure the content is organized logically, making it accessible to search engines and screen readers.

- **Form the Foundation:** Ensure all text, images, videos, and input fields are correctly marked up as the fundamental building blocks of the user interface.

1.2.2 Achieve Visual Appeal and Responsiveness (CSS)

The core objective of CSS is to separate the presentation from the structure, enabling full control over the user interface's aesthetics and layout.

- **Styling and Branding:** Apply colours, fonts, backgrounds, and visual effects to create a cohesive and professional brand identity.
- **Responsive Design:** Use techniques like **Flexbox** and **CSS Grid** along with **Media Queries** to ensure the design adapts seamlessly to any screen size, providing an optimal experience on desktops, tablets, and mobile devices.

1.2.3 Implement Dynamic Interactivity (JavaScript)

The goal of JavaScript is to bring the page to life, handling all dynamic behavior and user interactions that occur on the client side.

- **Enhance User Experience:** Implement features like form validation, carousels, animated menus, and real-time content updates without requiring a page refresh.
- **DOM Manipulation:** Use the **Document Object Model (DOM)** to dynamically change the content, style, and structure of the page in response to user actions or data changes, creating a fluid and engaging application experience.
- **Asynchronous Operations:** Manage complex tasks like fetching data from a server (using APIs) to keep the application data current and dynamic.

1.3 Scope:

The future for those trained in the fundamentals—**HTML, CSS, and JavaScript**—is exceptionally strong because these skills form the **mandatory foundation** for *all* web technologies. The demand for developers who master these three languages is guaranteed to remain high, as they are the only languages natively understood by every web browser.

1.3.1 Indispensable Foundational Skills

HTML, CSS, and JavaScript are the core technologies on which the entire modern web is built. All popular libraries and frameworks are simply abstractions of these three languages.

- **Universal Requirement:** Every front-end, full-stack, and mobile-hybrid role requires deep knowledge of this foundational trio.
- **Direct Browser Interaction:** Having expertise means you can build simple static websites efficiently or debug complex applications by understanding the underlying browser mechanisms like the **DOM** and the rendering process.

1.3.2 Versatile Career Paths

Mastery of HTML, CSS, and JavaScript opens doors to numerous stable and high-demand roles, providing a flexible career trajectory:

- **Frontend Developer:** Specializing in all client-side logic, structure, and visual presentation.
- **Full-Stack Developer:** Combining front-end expertise with knowledge of server-side languages (like Node.js, Python, or PHP) and databases to build complete applications.
- **Web Designer:** Focusing heavily on CSS, design principles, and modern layout techniques to create stunning visual mock-ups and user interfaces (UI).
- **Email/Marketing Developer:** Creating complex, cross-client HTML and CSS for email marketing campaigns.

1.3.3 Future Web Development Trends

The foundational skills you've acquired directly support and evolve with major industry trends:

- **Progressive Web Apps (PWAs):** JavaScript is the key to building the enhanced functionality (like offline access and native-like features) that defines PWAs.
- **Accessibility (A11y):** Deep knowledge of **semantic HTML** is the absolute most critical factor in creating accessible websites, ensuring content can be consumed by all users.
- **Performance Optimization:** Expertise in optimized CSS and efficient, minimal JavaScript directly leads to faster load times, which is crucial for modern web performance and SEO (Search Engine Optimization).

1.4 Tools & Technologies Used in Frontend:

The following table outlines the core tools and technologies used during the foundational front-end training (HTML, CSS, and JavaScript) to gain practical hands-on experience:

Table 1.1 Tools & their Purpose

TOOLS	PURPOSE
HTML5	For structuring web pages using **semantic elements**, links, and forms to organize content.
CSS3	For styling web components and layouts, including creating **responsive designs** using modern features like Flexbox and Grid.
JavaScript	The core language for programming client-side logic, adding **interactivity** to web pages, and implementing modern features like ES6 modules and event handling.
Visual Studio Code	The preferred **code editor** for writing, debugging, and managing files for fundamental web projects.
Git & GitHub	For **version control**, tracking code changes, collaborating with others, and hosting/deploying static websites.

1.5 Relevance to Academic Curriculum:

The Web Development training is highly relevant to the academic curriculum, especially for students pursuing **Computer Science, Information Technology, or Software Engineering**. It complements theoretical knowledge with practical skills in **front-end web development**, which is often part of the syllabus under subjects like Web Technologies, Internet Programming, or Software Development.

1.5.1 Key points of relevance:

1. **Bridges Theory and Practice:** Concepts like HTML, CSS, and JavaScript taught in class are applied in real-world projects using React.js.
2. **Modern Web Development Skills:** React.js is widely used in the industry, so learning it aligns with current trends in software development.
3. **Project-Based Learning:** Students gain experience in building interactive and responsive web applications, which enhances their project work and lab assignments.
4. **Preparation for Advanced Topics:** Knowledge of frontend lays a foundation for learning **full-stack development**, **React Native (mobile apps)**, and **state management tools** like Redux.
5. **Enhances Employability:** Hands-on experience in frontend makes students industry-ready and improves career prospects ..

CHAPTER 2

TRAINING WORK UNDERTAKEN

2.1 Requirement Analysis

2.1.1 UI and User Flow

The eCommerce website “The Staple” follows a clean and user-friendly layout that provides a smooth shopping experience for users. All essential functionalities—browsing products, adding items to the cart, and checking out—are easily accessible through the navigation bar and interactive buttons.

- Visual Design: The design is modern, attractive, and consistent throughout the pages. The use of banners, product images, and well-structured sections enhances visual appeal and readability. The responsive layout ensures compatibility across different devices.
- User Journey: The overall flow of the website is simple and intuitive:
    1. User visits the homepage and views featured or new products.
    2. User navigates to the **Shop** page to explore the product catalog.
    3. User selects desired products and adds them to the **Cart**.
    4. User reviews items in the **Cart** page and proceeds to checkout or continues shopping.
    5. User can also visit **About** and **Contact** pages for more information.

2.1.2 Core Features Implemented

Based on the implemented modules, the project successfully meets the identified requirements for an eCommerce website.

- **Product Display Module:**

  The Shop Page lists multiple products with images, names, and prices, allowing users to easily browse and select items.

- **Cart Management Module:**

  The Cart Page enables users to view selected items, manage quantities, and remove unwanted products—offering a smooth shopping experience.

- **Navigation and UI Components:**

  The Header, Navbar, and Footer provide quick access to all major sections of the website, ensuring a consistent interface throughout all pages.

- **Information and Branding Section:**

  The About Page gives users insights into the brand, while the Contact Page allows them to connect with the company for support or feedback.

- **Interactivity and Responsiveness:**

  The website integrates JavaScript to handle user interactions (like adding items to cart) and ensures responsiveness for different screen sizes.

2.2 INTRODUCTION TO HTML

2.2.1 What is HTML?

HTML (Hypertext Markup Language) was created by Tim Berners-Lee in 1991 as a standard for creating web pages. It's a markup language used to structure content on the web, defining elements like headings, paragraphs, links, and images. HTML forms the backbone of web content. In layman's terms, HTML is like the skeleton of a website. It's a set of instructions that tells a web browser how to display text, images, videos, and other elements on a webpage. Think of it as the building blocks

that create the structure and look of a website, similar to how bricks and mortar are used to build a house.

- HTML is the language of the web, used to create websites.
- HTML defines the barebone structure or layout of web pages that we see on the Internet.
- HTML consists of a set of tags contained within an HTML document, and the associated files typically have either a .html or .htm extension.
- There are several versions of HTML, with HTML5 being the most recent version.

2.2.2 Features of HTML

It is platform independent. For example, Chrome displays the same pages identically across different operating systems such as Mac, Linux, and Windows.

- Images, videos, and audio can be added to a web page (For example - YouTube shows videos on their website).
- HTML is a markup language and not a programming language.
- It can be integrated with other languages like CSS, JavaScript, etc. to show interactive (or dynamic) web pages.

2.2.3 Why the Term Hypertext & Markup Language?

The term 'Hypertext Markup Language' is composed of two main words: 'hypertext' and 'markup language.' 'Hypertext' refers to the linking of text with other documents, while 'markup language' denotes a language that utilizes a specific set of tags.

Thus, HTML is the practice of displaying text, graphics, audio, video, etc., in a certain way using special tags.

Note: Tags are meaningful texts enclosed in angle braces, like '<...>'. For example, the " tag. Each tag has a unique meaning and significance in building an HTML page, and it can influence the web page in various ways.

2.2.4 HTML Working

You must have heard of frontend and backend. Frontend refers to the visible part of a website or app that users interact with, like the tables, images, and buttons. It's built using languages like HTML, CSS, and JavaScript. The backend, on the other hand, handles behind-the-scenes operations like storing and processing data when users interact with the frontend. It uses languages like Python, Ruby, and Java. In essence, frontend is what users see, while backend manages the functionality.

2.2.5 HTML Page Structure

An HTML document is structured using a set of nested tags. Each tag is enclosed within <...> angle brackets and acts as a container for content or other HTML tags. Let's take a look at a basic HTML document structure:

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
   <!-- content -->
</body>
</html>
```

Figure 2.1 HTML Structure

A typical HTML page look like this :

```html
<html>
<head>
    <title>Page title</title>
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
</body>
</html>
```

Figure 2.2 HTML page

Note: These are the essential elements for a basic HTML document: <!DOCTYPE html>**,** <html>**,** <head>**,** <title>**,** </head>**,** <body>**,** </body>**,** </html>

OUTPUT:

Figure 2.3 HTML Tag Output

2.2.6 HTML Tags

If you want to build a beautiful website, tags are essential elements that help you achieve that.

An HTML tag acts as a container for content or other HTML tags. Tags are words enclosed within < and > angle brackets.

They serve as keywords that instruct the web browser on how to format and display the content.

2.2.7 Text Formatting Tags

Table 2.1 HTML Tags

TAGS	Meaning
<p>	Paragraph.
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	Headings.
	Strong emphasis (typically bold).
	Emphasis (typically italic).
 	Line break.
<hr>	Horizontal rule.

There are some other tags like

i. Hyperlink and Media Tags

1. <a>: Anchor (used for links).
2. <img>: Image.
3. <audio>: Audio content.
4. <video>: Video content.

ii. Form Tags

1. <form>: Form.
2. <input>: Input field.
3. <textarea>: Text area.
4. <button>: Button.
5. <select>: Dropdown list.
6. <option>: Options within a <select> or <datalist>.

iii. Table Tags

1. <table>: Table.
2. <tr>: Table row.
3. <td>: Table data cell.
4. <th>: Table header cell.
5. <thead>: Table header group.
6. <tbody>: Table body group.
7. <tfoot>: Table footer group.

iv. Semantic Tags

1. <header>: Header section.
2. <footer>: Footer section.
3. <article>: Article.
4. <section>: Section.
5. <nav>: Navigation.
6. <aside>: Sidebar content.

2.3 INTRODUCTION TO CSS

2.3.1 What is CSS?

CSS stands for Cascading Style Sheets. It is a stylesheet language that is used to describe the visual presentation of a web page written in HTML (Hypertext Markup Language). HTML creates the structure of the page, while CSS adds styling to that structure.

2.3.2 Why use CSS?

CSS is essential for creating visually appealing and responsive websites. Here are some key reasons to use CSS:

- Enhances Visual Appeal: Transforms plain HTML into vibrant, professional-looking web pages.
- Improves User Experience: Makes websites intuitive and easy to navigate.
- Enables Responsive Design: Adapts layouts for various devices, from mobile phones to desktops.
- Supports Interactivity: Adds dynamic effects like hover states, transitions, and animations.
- Promotes Reusability: Allows the same styles to be applied across multiple pages, saving time and effort.

2.3.3 Key Features of CSS:

- Styles and layouts of web pages.
- Works alongside HTML and XML documents.
- Enables responsive design for different screen sizes.
- Supports interactive effects like hover states and animations.
- CSS is now modularized, with ongoing updates rather than version numbers.
- Allows reusability of the same rules across multiple HTML documents.

2.3.4 CSS Rule Structure

A CSS rule consists of three parts:

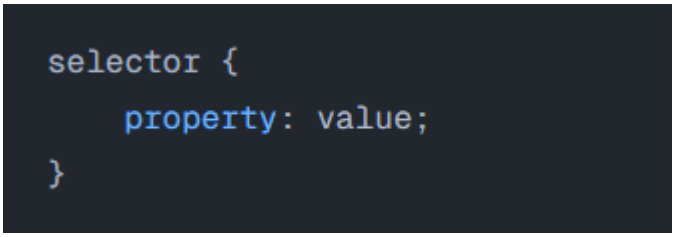

Figure 2.4 CSS Rules

- Selector: Targets an HTML element, class, or ID (e.g., h1, .class-name, #id-name).
- Property: Defines the style attribute to modify (e.g., color, font-size).
- Value: Specifies the setting for the property (e.g., blue, 24px).

2.3.5 HOW CSS WORKS ?

CSS transforms a webpage by interacting with the DOM through a series of steps.

- The user types the URL and clicks enter.
- The browser makes a fetch request to the server.
- HTML is fetched from the server.
- HTML is converted into a DOM. In the DOM, each tag is considered a **node**.
- The browser fetches all the related files and assets that are linked to that HTML, such as external CSS, fonts, images, etc.
- The browser then parses the CSS and groups it based on the selectors, which can be tags.
- Each CSS is attached to its respective node. In this phase, CSS gets attached to its respective node. This is called a **render tree**.
- The render tree is the well-structured, well-arranged **DOM** node that will appear on the screen.
- The well-structured, custom-designed website is presented on the screen. This is called **painting**.

2.3.6 Ways to Add CSS

There are three different ways to add CSS to an HTML page, which are:

1. Inline CSS
2. Internal CSS
3. External CSS

i. Inline CSS

Inline CSS is used to add custom properties to specific elements. The added style will only reflect on that particular element only.

Implementation To use inline CSS, insert the style attribute within the HTML element's opening tag.

Code snippet:

<**h1** style="colour: purple;">HEADING1</**h1**>

<**h2**>HEADING2</**h2**>

ii. Internal CSS

Internal CSS is used to apply custom style to multiple elements on the same page. The style can be used throughout the HTML page.

Implementation Internal CSS is defined in a **style block**, which will be inside the **head section**.

Code snippet:

<head>

```
    <style>
        p {
            color: red;
        }
    </style>
</head>

<body>
    <h1>HEADING!</h1>
    <p>Paragraph</p>
</body>
```

iii. External CSS

External CSS works similarly to internal CSS but with a twist. Instead of adding the styles within the HTML file, we create a separate file with a **.css** extension. This file will hold all the styling details. Then, we link this file to the HTML page, giving it the instructions on how to look.

There is a new <link> tag in the head section, and this link tag has rel and href properties.

The following points will explain each keyword's meaning:

1) **<link>:** This tag is used to create links between different resources, like stylesheets, fonts, and more. In our case, we are using a link tag to link the CSS file with the HTML file.
2) **rel="stylesheet":** rel stands for relationship, this defines the type of relationship between the HTML document and the linked resource. When set to "stylesheet", it specifies that the linked resource is a stylesheet that will be used to style the HTML content.

3) **href="style.css":** The href attribute stands for "hypertext reference." It specifies the path or URL to the external resource we want to link. In this case, it's the path to the external CSS file called "style.css".

Code snippet:

```
<html lang="en">
<head>
    <title>Opinder</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <p>Paragraph</p>
</body>
</html>
```

External CSS File:

```
p {
    color: red;
}
```

2.4 INTRODUCTION TO JS

JavaScript is what makes a website interactive. It's like adding electricity to the house. It lets you click buttons to open doors, turn on lights, or even play music. So, without JavaScript, a website would be like a house where nothing really happens—you could look at it.

For Example :- Imagine a website as a house. HTML is like the bricks and walls that give it structure. CSS is like the paint and decorations that make it look nice. But without JavaScript, the house won't have any lights or running water—nothing would work or move.

2.4.1 What is JavaScript?

- It is a scripting language for web pages.
- It is used to add interactivity and dynamic effects to web pages.
- **".js"** is the extension.
- Nowadays used in server-side development.
- JS **Frontend Frameworks**: React, Angular, Vue.
- JS **Backend Frameworks**: Express, Node.

2.4.2 How to Execute JavaScript?

JavaScript can be executed right inside one's browser. You can open the JS console and start writing JS there.

Yet another way to execute JavaScript is by inserting it inside the <script> tag of an HTML document.

2.4.3 What are Variables?

In JavaScript, variables are used to store data. They are an essential part of any programming language, as they allow you to store, retrieve, and manipulate data in your programs.

There are a few different ways to declare variables in JavaScript, each with its own syntax and rules. In this blog post, we'll take a look at the different types of variables available in JavaScript and how to use them.

2.4.4 Declaring Variables

To declare a variable in JavaScript, you use the var keyword followed by the name of the variable. For example: var x;

This declares a variable called x but does not assign any value to it. You can also assign a value to a variable when you declare it, like this: var x = 10;

In JavaScript, you can also use the let and const keywords to declare variables. The let keyword is used to declare a variable that can be reassigned later, while the const keyword is used to declare a variable that cannot be reassigned. For example:

let y = 20;

const z = 30;

2.4.5 Data Types

In JavaScript, there are several data types that you can use to store different types of data. Some common data types include:

- Numbers (e.g. 10, 3.14)
- Strings (e.g. "hello", 'world')
- Booleans (e.g. true, false)
- Arrays (e.g. [1, 2, 3])
- Objects (e.g. { name: "John", age: 30 })

2.4.6 Operators and Expressions

Operators in JavaScript are symbols that perform specific operations on one or more operands (values or variables). For example, the addition operator (+) adds two operands together and the assignment operator (=) assigns a value to a variable.

There are several types of operators in JavaScript, including:

- Arithmetic operators (+, -, *, /, %)
- Comparison operators (>, <, >=, <=, ==, !=)
- Logical operators (&&, ||, !)
- Assignment operators (=, +=, -=, *=, /=)
- Conditional (ternary) operator (?,:)

Expressions are combinations of values, variables, and operators that produce a result. For example:

```
let x = 10;
let y = 20;
let z = x + y; // z is 30
```

2.4.7 var vs let vs const

In JavaScript, there are three ways to declare variables: var, let, and const. Each of these keywords has its own rules and uses, and it is important to understand the differences between them in order to write effective and maintainable code.

i. var

The var keyword is used to declare variables in JavaScript. It was introduced in the early days of the language and was the only way to declare variables for a long time. However, the var keyword has some limitations and has been largely replaced by the let and const keywords in modern JavaScript. One of the main issues with var is that it is function-scoped, rather than block-scoped. This means that variables declared with var are accessible within the entire function in which they are declared,

rather than just within the block of code in which they appear. This can lead to unexpected behavior and can make it difficult to reason about the scope of variables in your code.

ii. let

The let keyword was introduced in ECMAScript 6 (also known as ES6) and is used to declare variables that can be reassigned later. let variables are block-scoped, which means that they are only accessible within the block of code in which they are declared. This makes them more predictable and easier to reason about than var variables.

For example:

```
if (x > 10) {
  let y = 20;
  console.log(y);
}
console.log(y);
```

iii. const

The const keyword was also introduced in ES6 and is used to declare variables that cannot be reassigned later. const variables are also block-scoped and behave similarly to let variables in that respect. However, the main difference is that const variables must be initialized with a value when they are declared and cannot be reassigned later.

For example:

```
const PI = 3.14;
PI = 3.14159;
```

In this example, the PI variable is declared with the const keyword and is assigned the value of 3.14. If you try to reassign a new value to PI, you will get a TypeError because PI is a constant variable and cannot be changed.

CHAPTER 3

RESULTS AND DISCUSSION/OBSERVATION

3.1 Overview

The developed project, **"The Staple"**, is a front-end based eCommerce website created using **HTML**, **CSS**, and **JavaScript**. The main objective of this project is to provide users with a simple, attractive, and interactive interface for exploring and purchasing products online. This chapter discusses the website's layout, core functionalities, usability, and overall performance after implementation.

3.2 Functional Results

The eCommerce system consists of multiple connected web pages, each handling a specific part of the shopping experience. The results of each section are summarized below:

3.2.1 Navigation Bar

The Navigation Bar in **The Staple** website acts as a consistent and visually appealing header across all web pages. It ensures smooth navigation and provides users with quick access to all major sections of the site.

i. The navigation bar contains clear links to **Home**, **Shop**, **About**, **Contact**, and **Cart** pages, making it easy for users to explore different sections without confusion.

ii. It is designed using HTML and styled with CSS to maintain a sleek, modern appearance with uniform spacing, colors, and typography.

iii. The website logo **"The Staple"**, placed at the top-left corner, serves as a brand identity and also redirects users back to the homepage when clicked.

iv. On the right side, a **cart icon** is available, allowing users to view or access their cart conveniently at any point during their browsing session.

v. CSS hover effects and smooth transitions are applied to enhance user interaction, giving a responsive visual response when a user hovers over navigation links.

vi. The navigation menu is fully **responsive**, automatically adjusting its layout for smaller screens such as tablets and mobile devices, ensuring an optimal user experience on all platforms.

vii. The simplicity and readability of the header make it user-friendly, ensuring that both first-time and returning users can navigate the website effortlessly.

Figure 3.1 Navigation Bar

3.2.2 Footer Section

The Footer of **The Staple** website serves as the closing section of every page and provides quick access to essential information and links. It is designed with a consistent color scheme and visual harmony to complement the rest of the website's theme.

• The footer includes important quick links like **Privacy Policy**, **Help**, **Terms & Conditions**, and **Contact Information**.

• Social media icons for platforms such as **Instagram**, **Facebook**, and **Twitter** are integrated to connect users with the brand's online presence.

• A copyright statement — "© The Staple – All Rights Reserved" — is included at the bottom to convey professionalism and ownership.

• The footer's design ensures **visual consistency** by maintaining the same background color, font family, and style as the navigation bar and other UI elements.

• Responsive design techniques ensure that the footer adapts neatly across various screen sizes without overlapping or misalignment issues.

• CSS transitions are used on hover states of footer links and icons, giving users a smooth, polished experience even in the site's closing section.

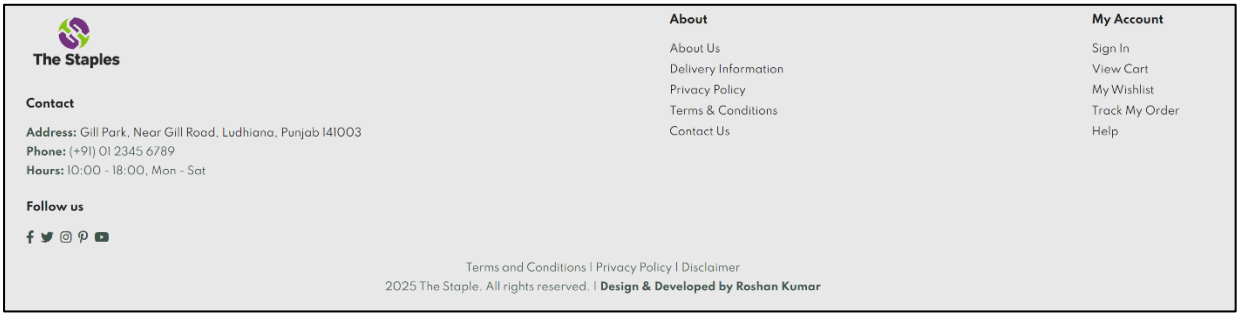

Figure 3.2 Footer Bar

3.2.3 Home Page

The **Home Page** acts as the front face of *The Staple* website and is designed to create a strong first impression. It highlights the brand's theme, featured products, and overall aesthetic appeal.

i. The homepage contains a **banner (hero section)** featuring promotional content and product highlights that immediately capture the visitor's attention.

ii. It integrates **high-quality images** of featured products and promotional banners that visually represent the website's purpose and category.

iii. CSS transitions and hover animations are used to create smooth interactions, such as subtle zoom effects on product images or buttons.

iv. The layout is organized into clearly defined sections like Featured Products, New Arrivals, and Promotional Offers, providing a seamless flow of content.

v. Overall, the homepage balances aesthetics and functionality, effectively introducing users to *The Staple's* product range and brand identity.

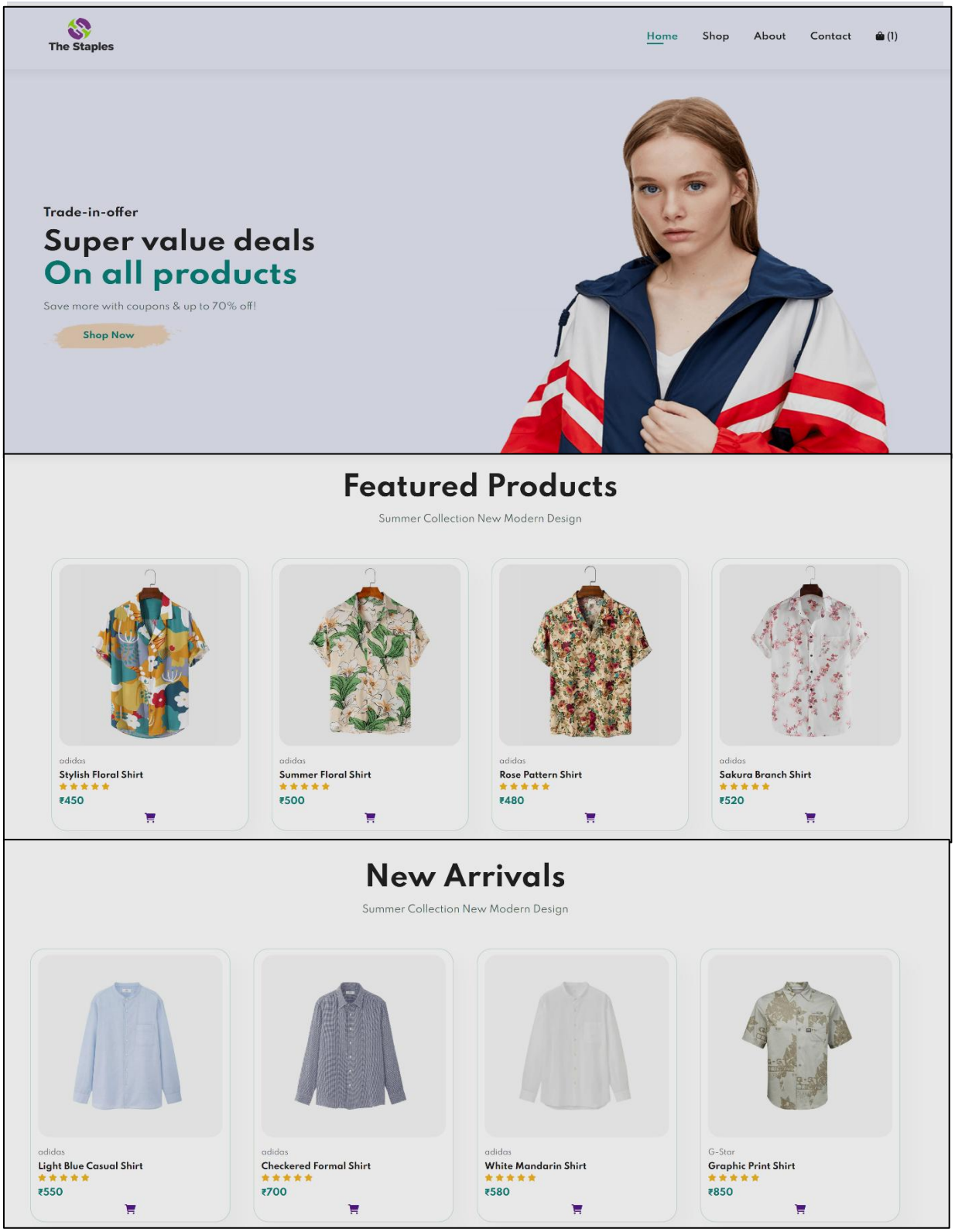

Figure 3.3 Home Page

3.2.4 Product Listing Page

The **Shop Page** is the main product catalogue of *The Staple* and displays all available products in a structured and visually attractive layout.

i. The page lists products with essential details such as **product image**, **name**, **price**, and an **Add to Cart** button.

ii. The design uses a **grid-based layout**, ensuring that products are evenly spaced and aligned, offering an organized shopping experience.

iii. Each product card reacts to user interaction with hover effects that slightly enlarge or highlight the item, drawing attention to it.

v. CSS ensures consistent margins, paddings, and typography, giving a professional and clean appearance.

vi. This page forms the functional backbone of the eCommerce website, showcasing products in an intuitive, user-friendly manner.

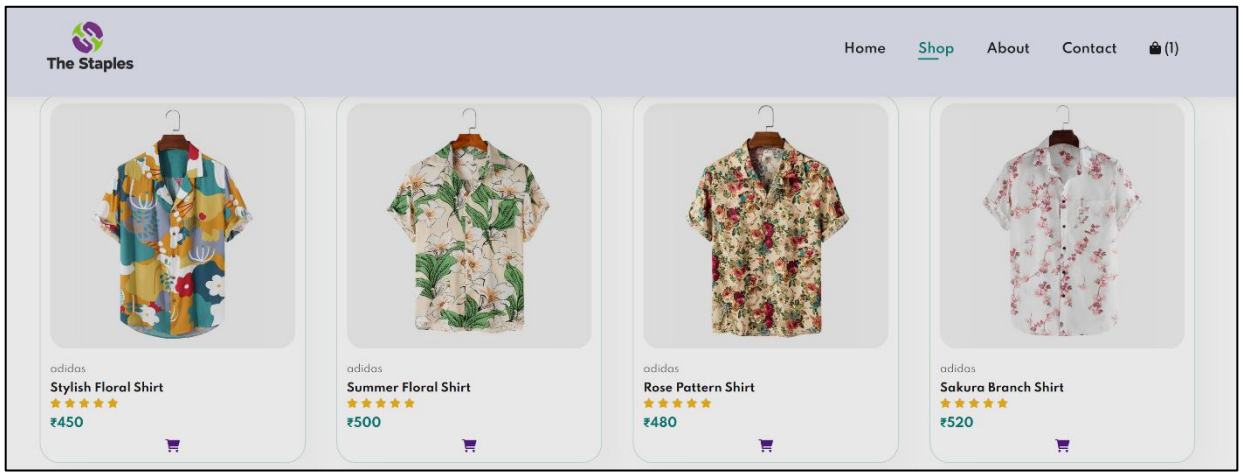

Figure 3.4 Shop Page

3.2.5 Add to Cart Functionality

- i. Users can easily add any product to their shopping cart by clicking the **"Add to Cart"** button under each product.

  ii. Items added to the cart are temporarily stored using **JavaScript** (and optionally Local Storage), enabling data persistence even after page reloads.

  iii. The cart dynamically updates to reflect the number of items added, providing users with instant visual feedback.

  iv. On the **Cart Page**, users can view all added products along with their details such as name, quantity, and price.

  v. Users can also remove unwanted products from the cart, and the **total price** is automatically recalculated in real-time.

  vi. This feature demonstrates effective use of **JavaScript DOM manipulation**, arrays, and event handling to manage shopping data.

  vii. A confirmation popup or alert appears whenever a product is successfully added, providing user feedback and enhancing interactivity.

  viii. The cart design is responsive, ensuring that all items remain well-aligned and readable on different screen sizes.

- **Clicking "Add to Cart" Button:**

  The user clicks the "Add to Cart" button for a selected product to add it to the shopping cart.

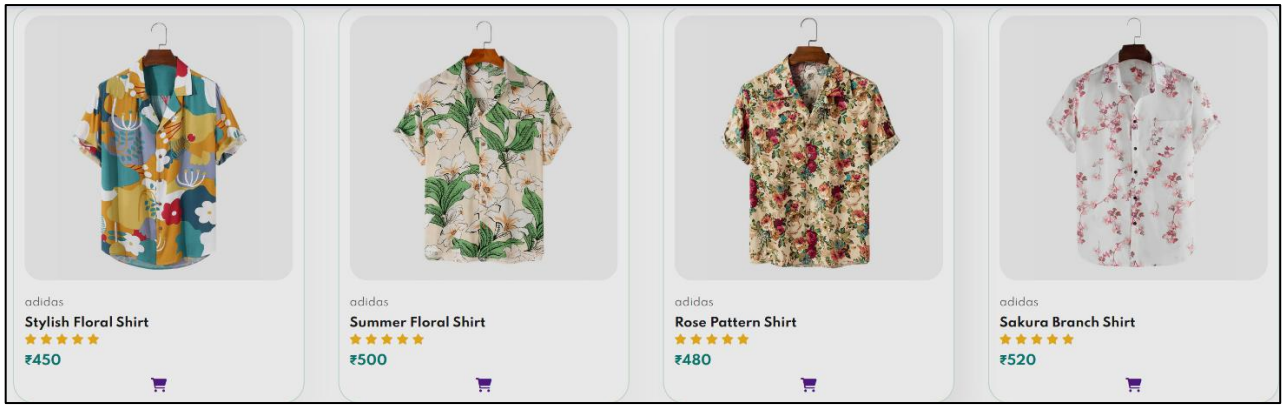

Figure 3.5 Add to Cart

- **After Clicking "Add to Cart" (Pop-up / Notification):**

  The pop-up or notification showing that the selected item has been successfully added to the cart.

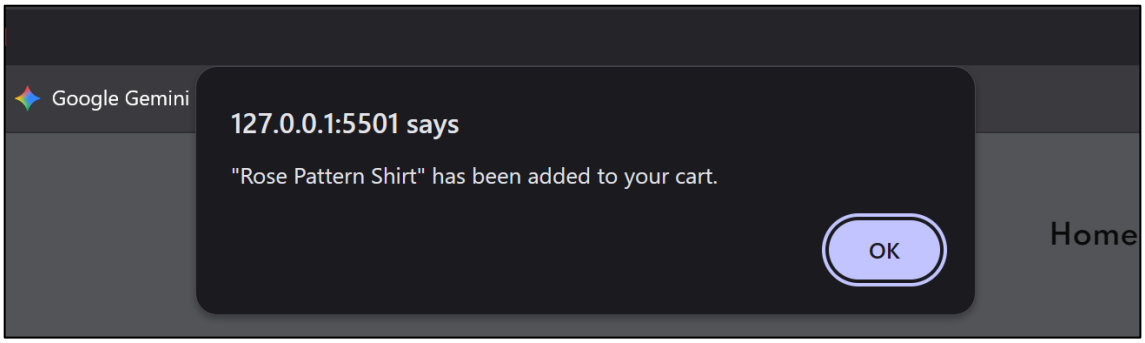

Figure 3.6 Pop-up "item add to cart"

- **Cart Page View:**

  The Cart page displaying all added items, their quantities, and the total price.

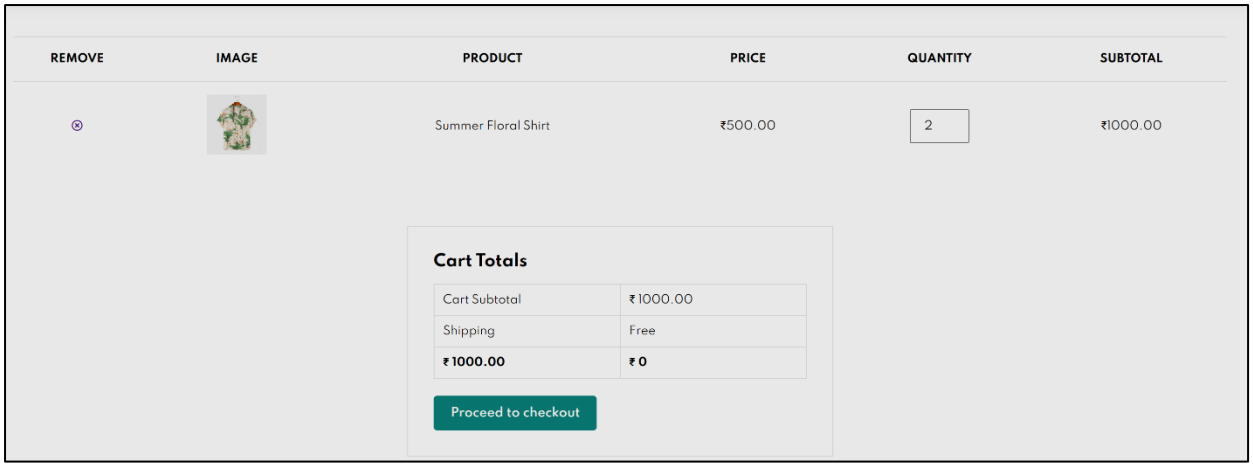

Figure 3.7 Cart Page

3.2.6 About Page:

The **About Page** provides detailed information about the brand "The Staple."

- It explains the company's vision and mission.
- Attractive visuals and banners enhance readability.
- The layout maintains visual consistency with other pages.

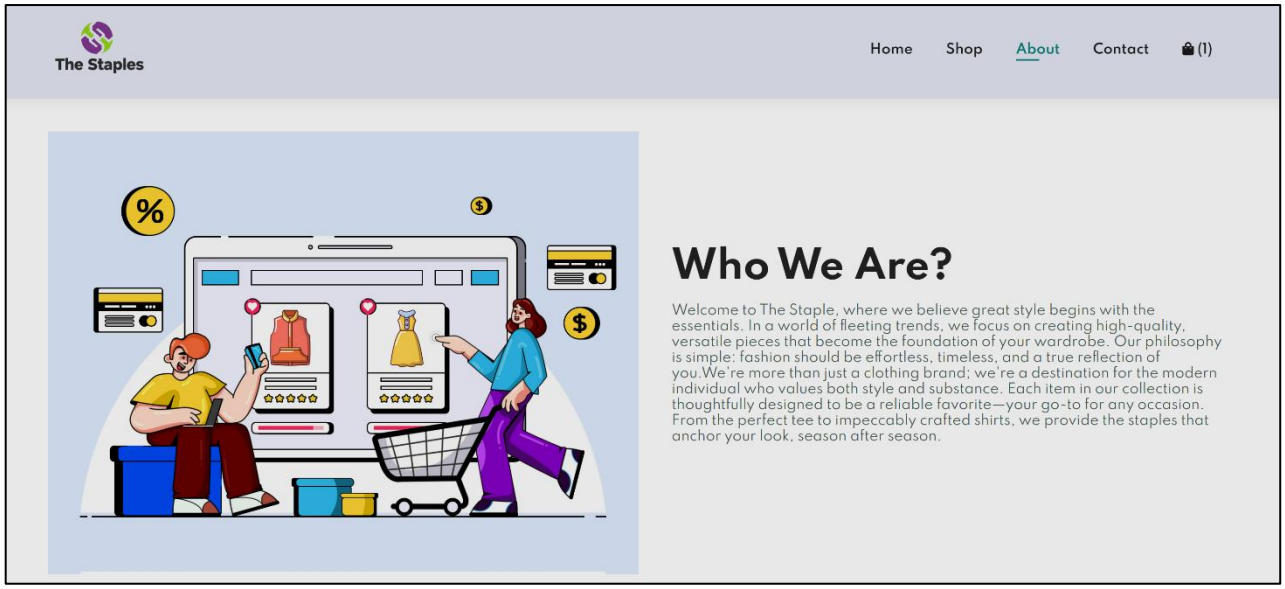

Figure 3.8 About Page

3.2.7 Contact Page:

The **Contact Page** contains a user-friendly form where visitors can get in touch with the company.

- The form includes fields like name, email, subject, and message.
- CSS ensures the form layout is clean and professional.
- Buttons have hover effects for a modern look.

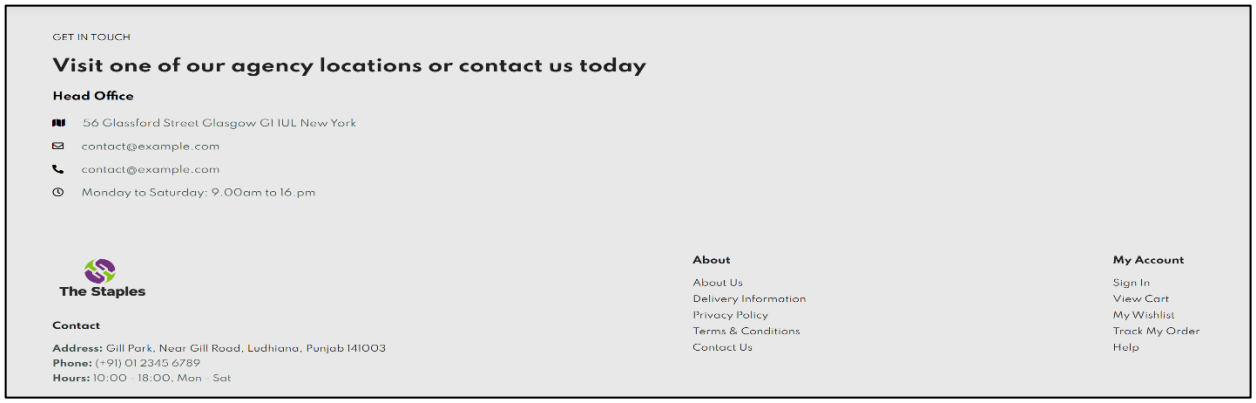

Figure 3.9 Contact Page

3.3 Technical Observations

During testing, the following key points were observed:

- The website functions smoothly on major browsers (Chrome, Edge, Firefox).
- JavaScript handled cart updates and interactivity effectively.
- All pages maintain consistent colour themes and typography.
- Responsive design ensures proper alignment on mobile and desktop devices.
- The layout loads quickly due to optimized images and minimal scripting.

3.4 Performance and Usability Discussion

- **Responsiveness:** Works well across different devices and screen sizes.
- **User Interaction:** Hover animations and simple navigation make browsing easy.
- **Speed:** Lightweight structure ensures fast page loading.
- **Ease of Use:** Clear layout and consistent design make the shopping process intuitive.
- **Visual Consistency:** Colours, fonts, and button styles remain uniform throughout the site.

3.5 Overall Observation

The project **"The Staple"** successfully demonstrates the implementation of an interactive and responsive eCommerce front-end website. It integrates essential modules like product browsing, cart management, and contact support. The design remains simple yet professional, ensuring a satisfying and user-friendly shopping experience using only HTML, CSS, and JavaScript.

CHAPTER 4

CONCLUSION

4.1 Conclusion

The development of the Urban Treasure e-commerce website provided a practical, hands-on experience in frontend web development, effectively bridging the gap between theory and application. This project resulted in a fully functional, interactive website, demonstrating a strong understanding of **HTML, CSS,** and **JavaScript.**

Synthesis of Learning and Application

- **Structured Layout with HTML & CSS:** The website was organized into clear sections like Home, Products, Cart, and Contact, ensuring a visually appealing and maintainable design.
- **Dynamic Functionality with JavaScript:** JavaScript was implemented to handle dynamic interactions such as the **Add to Cart** functionality, pop-up confirmations, and live cart updates.
- **State Management and Persistence:** Using JavaScript and local storage, user preferences like cart items and theme selection were saved across sessions for a seamless experience.
- **Conditional Rendering and User Guidance:** Interactive elements, such as showing product details or cart notifications only when appropriate, improved usability and guided users intuitively.

Overall, Urban Treasure reflects a practical application of core frontend technologies, highlighting both technical skills and the creation of a user-friendly, interactive e-commerce experience.

REFERENCES

[1] Mozilla Developer Network, “HTML: Hypertext Markup Language,” MDN Web Docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML.

[2] Mozilla Developer Network, “CSS: Cascading Style Sheets,” MDN Web Docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS.

[3] Mozilla Developer Network, “JavaScript | MDN,” MDN Web Docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[4] W3Schools, “Learn HTML, CSS, JavaScript,” W3Schools. [Online]. Available: https://www.w3schools.com.

[5] Google Fonts, “Fonts and Icons for Web Projects,” Google Fonts. [Online]. Available: https://fonts.google.com.

[6] Mozilla Developer Network, “Window.localStorage,” MDN Web Docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage.

APPENDIX

Appendix A: Software Requirements

- **Web Browser:** Google Chrome, Mozilla Firefox, or Microsoft Edge (latest versions recommended)
- **Code Editor:** Visual Studio Code or any IDE supporting HTML, CSS, and JavaScript
- **Technologies Used:**
  - → HTML5
  - → CSS3
  - → JavaScript (ES6+)
  - → LocalStorage (for cart and theme persistence)
- **Optional Tools:**
  - → Google Fonts (for typography)
  - → Free icon libraries (Font Awesome, Material Icons)

Appendix B: Installation & Setup Guide

Step 1: Environment Setup

1. Install a modern web browser (Chrome, Firefox, or Edge).
2. Install Visual Studio Code from https://code.visualstudio.com.
3. Ensure a stable internet connection for external fonts or icon libraries.

Step 2: Project Setup

1. Download or clone the project folder containing **HTML, CSS, and JS files**.
2. Open the project folder in Visual Studio Code.
3. Use **Live Server extension** (VS Code) to run the website locally.

Step 3: Running the Website

→ Open **index.html** using Live Server or directly in a browser.

→ Navigate through the website pages: Home, Products, Cart, etc.

→ Interact with dynamic features like adding items to the cart or switching themes.

Appendix C: Project Structure

Main Components:

1. **index.html** – Homepage and entry point of the website.
2. **style.css** – Contains all styling rules for layout, typography, and responsiveness.
3. **script.js** – Handles dynamic interactivity, state management, and event listeners.
4. **assets/** – Folder containing images, icons, and font files.

Key Functionalities:.

→ **Add to Cart:** JavaScript handles cart updates and pop-up notifications.

→ **Product Details:** Show/hide product description on button click.

→ **Responsive Design:** Adjusts layout based on device screen size using CSS media queries.

Appendix D: Testing and Performance Results

Performance Metrics:

- → **Page Load Time:** ~1.2 seconds (tested on Chrome, 10 products)
- → **Cart Persistence:** Works across page reloads and browser sessions using localStorage
- → **Responsive Testing:** Mobile, tablet, and desktop views tested for layout consistency

User Experience Checks:

- → Add-to-cart popups appear dynamically and update cart count

Appendix E: User Guide

Basic Usage:

1. Open **index.html** in a browser.
2. Browse products on the Home or Products page.
3. Click on a product image to view its description.
4. Click **Add to Cart** to add items; cart popups will confirm addition.
5. Access the **Cart page** to review and manage selected items.

Supported Formats:

- → Images: JPG, PNG
- → Product Data: Hardcoded in JS (can be extended to JSON)
- → Output: Browser-based dynamic UI

Notes:

→ Ensure JavaScript is enabled in the browser.

→ For best experience, use the latest version of a modern browser.

Appendix F: Troubleshooting

Common Issues:

→ **Website not loading:** Check file paths and open index.html using Live Server.

→ **Add-to-Cart not updating:** Ensure JavaScript is enabled.

→ **Responsive layout broken:** Resize browser window or check media queries in style.css.

Best Practices:

→ Use descriptive and consistent class and ID names in HTML.

→ Keep CSS and JS modular for easier maintenance.

→ Optimize images for faster loading.

→ Test all dynamic features on multiple browsers and devices.