

FULL STACK DEVELOPMENT

Online Grocery Store

Summer Internship Report Submitted in partial fulfilment

of the requirement for under graduate degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

By

K. B. Indraneel

221710301030

<https://github.com/Indraneelkatta>

Under the Guidance of

Assistant Professor



Department Of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

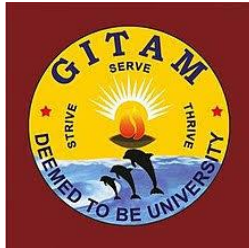
July 2020

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY**

GITAM

(Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS



DECLARATION

I submit this industrial training work entitled **“Online Grocery Store”** to GITAM (Deemed to Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of **“Bachelor of Technology”** in **“Computer Science and Engineering”**. I declare that it was carried out independently by me under the guidance of (), Asst. Professor, GITAM (Deemed to Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: Hyderabad

Date:

Name and Signature of Candidate

K. B. Indraneel

221710301030

CSE

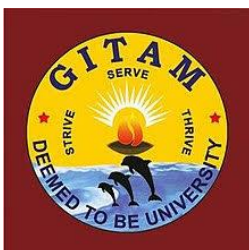
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS



CERTIFICATE

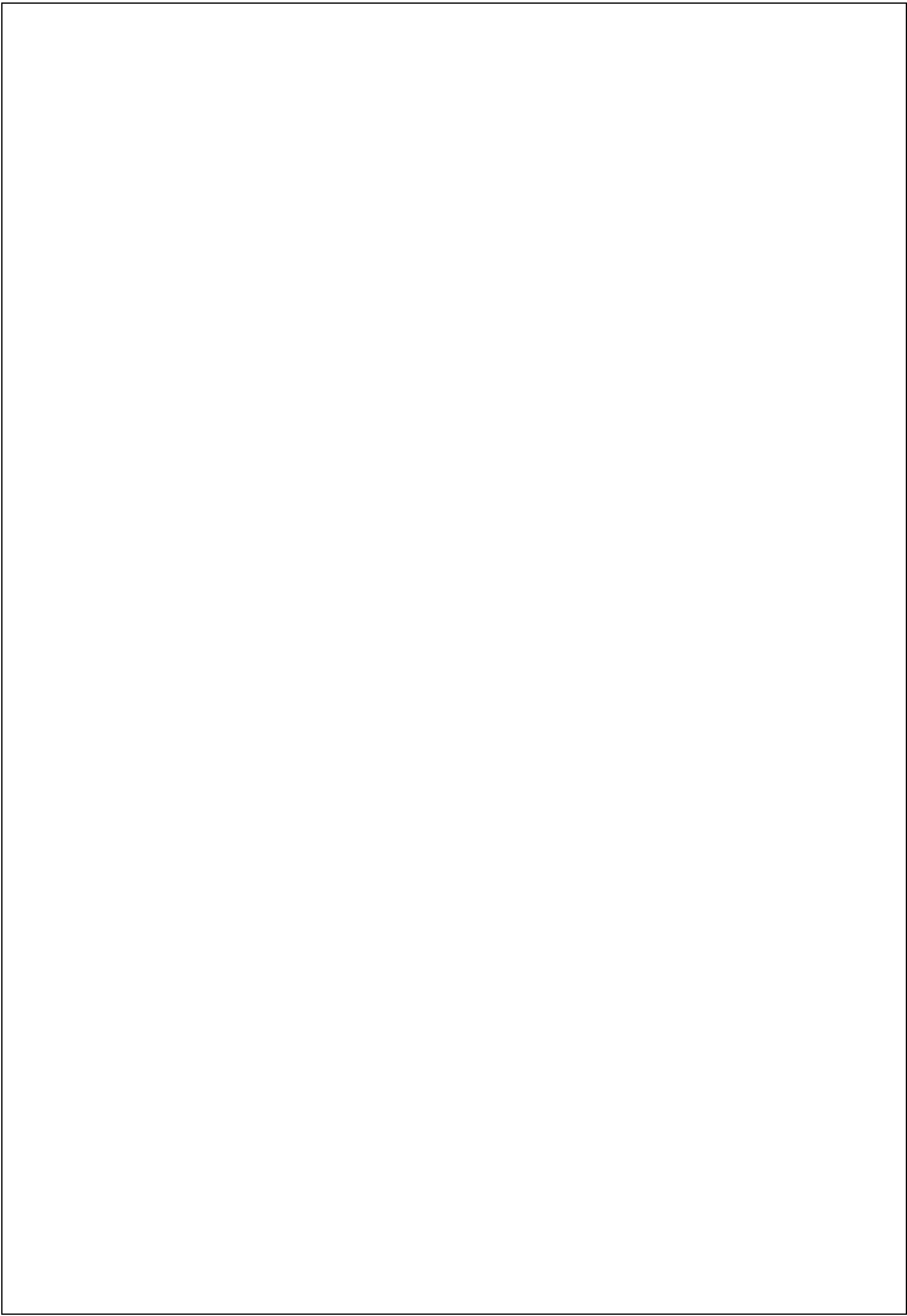
This is to certify that the Industrial Training Report entitled - "**Online Grocery Store**" is being submitted by **K. B. Indraneel (221710301030)**, submitted in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology in Computer Science and Engineering**.

Guided by

Head of the Department

Dr. Phani Kumar

Professor & HOD



ACKNOWLEDGEMENT

My project would not have been successful without the help of several people, I would like to thank the personalities who were part of my project in numerous ways, those who gave me outstanding support from the birth of the project.

I would like to thank our honorable Pro-Vice-Chancellor, **Prof. N. Siva Prasad** for providing necessary infrastructure and resources for the accomplishment of my project.

I would like to thank respected **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the project.

I would like to thank respected **Prof. S. Phani Kumar**, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in the completion of this project.

I would like to thank respected **Mr. ,** Assistant Professor in Computer Science Department for the esteemed guidance, moral support and invaluable advice provided by him for the success of the project.

I would like to thank my parents and friends who extended their help, encouragement and moral support either directly or indirectly in my project work.

Sincerely,
K. B. Indraneel
(221710301030)

ABSTRACT

As the Internet has become widely used by many segments of the population, the opportunity to shop online for groceries has been presented to consumers. Online grocery shopping is a new way of buying desired grocery products for household consumption. Due to the advancement of today technology, online grocery shopping is gaining their market share in the food retail industry.

An Online Grocery Store permits a customer to submit online orders for items and services from a store that serves for online customers. The online Store System presents an online display of all the items we want to sell. This web based application helps customers to choose their daily needs and add products to their shopping cart. Customers provide their complete details of address, contact and they get their chosen products in their home.

This Web Application saves a lot of time for customers.

Table of Content

1. Introduction.....	10
1.1. Front End.....	10
1.2. Back End.....	10
2. Web Development.....	12-14
2.1. Web-Site.....	12
2.2. Web-page.....	13
2.3. Web-Browser.....	13
2.4. Mobile Web Browser.....	14
3. The steps to create a Web Site.....	15-25
3.1. UI Development.....	16
3.1.1. HTML.....	16
3.1.2. CSS.....	17
3.1.3. Bootstrap.....	20
3.2. Scripting.....	22
3.2.1 Server Side Scripting.....	22
3.2.1.1 Server Side Scripting Languages.....	22
3.2.2 Client Side Scripting.....	23
3.3 Database.....	24
3.1.1 Advantages	24
3.1.2 Disadvantages	24
3.4 SQL.....	24
3.5 Queries.....	25
4. Scripting Languages.....	26-29
4.1. NodeJS.....	26
4.2. History.....	26
4.3. Javascript.....	27
4.4. JQuery.....	28
4.5. AJAX.....	28
4.6. JSON.....	28
4.7. XAMPP.....	29
4.8. Features.....	29
4.9. Usage.....	29
5. Software Requirement Specification.....	30
5.1. Hardware Specification.....	30
5.2. Software Specification.....	30
6. Data Flow Diagram.....	31-34
6.1. DFD-1.....	32
6.2. DFD-2.....	32
6.3. DFD-3.....	34

7. Project.....	35
7.1 Project	35
7.2 Technologies Used.....	35
7.3 Technology Details.....	36
7.4 Wireframes.....	37-40
7.5 E-R Diagrams.....	41
7.6 Screenshots.....	42-45
8. Assigned Modules	46-53
9. Maintenance.....	54
10. Future scope and enhancement.....	55
11. Conclusion.....	56
Bibliography	56

List of Figures

Figure 1.1	Classifications of Technologies	11
Figure 2.1	Architecture of Web Browsers.....	14
Figure 3.1	Steps to create a Web site	15
Figure 3.2	Versions of HTML	17
Figure3.3	Login form without CSS.....	19
Figure3.4	Login form with CSS.....	20
Figure 3.5	Login page using Bootstrap	21
Figure3.6	Client side Scripting	23
Figure4.1	Uses of Node JS.....	26
Figure 4.2	Frame works of Java script.....	27
Figure 6.1	Context level diagram	31
Figure6.2	First level DFD	32
Figure6.3	Second level DFD-1.....	32
Figure6.4	Second level DFD-2.....	33
Figure6.5	Use case diagram.....	34
Figure7.1	Wireframe for home page	37
Figure7.2	Wireframe for login page	37
Figure7.3	Wireframe for registration page	38
Figure7.4	Wireframe for vegetables page	38
Figure7.5	Wireframe for fruits page	39
Figure7.6	Wireframe for diary products page	39
Figure7.7	Wireframe for cart page	40
Figure7.8	Wireframe for update password page	40
Figure7.9	Class diagram for grocery store	41
Figure7.6.1	Cart icon.....	42
Figure7.6.2	Default Cart.....	42
Figure7.6.3	Add to cart.....	43
Figure7.6.4	Increase the quantity.....	43
Figure7.6.5	Decrease the quantity.....	44
Figure7.6.6	Remove the product.....	44
Figure7.6.7	Checkout cart.....	45
Figure7.6.8	Clear cart.....	45

CHAPTER-1

INTRODUCTION

The industry definition of a Full Stack Developer is an engineer who can work on different levels of an application stack. The term stack refers to the combination of components and tools that make up the application. The components could be in the front-end or the back-end of the system. The main objective of full stack engineer is to keep every part of the system running smoothly. A Full Stack Developer can perform tasks ranging from resizing an image or text in a webpage to patching the kernel.

Full stack development: It refers to the development of both front end (client side) and back end (server side) portions of web application.

Full stack web Developers: Full stack web developers have the ability to design complete web application and websites. They work on the front -end, back-end, database and debugging of web application or websites.

1.1. FRONT-END

Front-end web development, also known as client-side development is the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly. The challenge associated with front end development is that the tools and techniques used to create the front end of a website change constantly and so the developer needs to constantly be aware of how the field is developing.

The objective of designing a site is to ensure that when the users open up the site they see the information in a format that is easy to read and relevant. This is further complicated by the fact that users now use a large variety of devices with varying screen sizes and resolutions thus forcing the designer to take into consideration these aspects when designing the site. They need to ensure that their site comes up correctly in different browsers (cross-browser), different operating systems (cross-platform) and different devices (cross-device), which requires careful planning on the side of the developer.

Front end development manages everything that users visually see first in their browser or application. Front end developers are responsible for the look and feel of a site. It is the visible part of website or web application which is responsible for user experience. The user directly interacts with the front end portion of the web application or website.

1.2. BACK-END

Back end development refers to the server side of an application and everything that communicates between the database and the browser. It is responsible for managing the database through queries and APIs by client-side commands.

Back end development refers to the server side of development where you are primarily focused on how the site works. Making updates and changes in addition to monitoring functionality of the site will be your primary responsibility. This type of web development usually consists of three parts: a server, an application, and a database. Code written by back end developers is what communicates the database information to the browser. Anything you can't see easily with the eye such as databases and servers is the work of a back end developer. Back end developer positions are often called programmers or web developers.

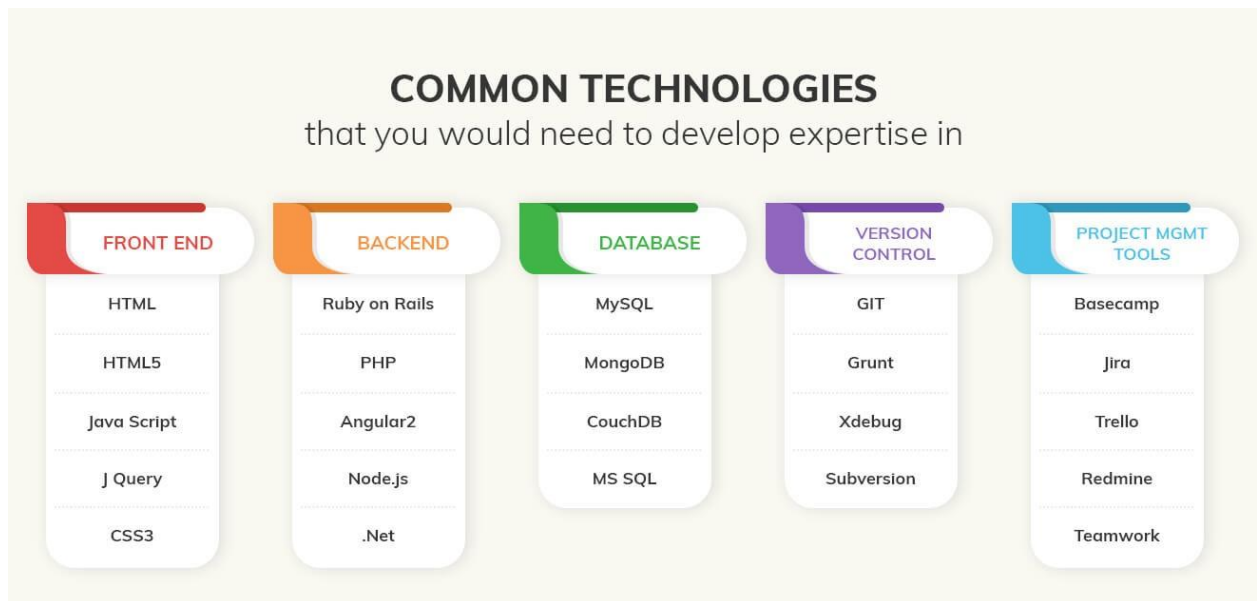


FIG 1.1 CLASSIFICATIONS OF TECHNOLOGIES

CHAPTER-2

WEB-DEVELOPMENT

Web development is a broad term for the work involved in developing a web site for the Internet(World Wide Web) or an intranet (a private network).Web development can range from developing the simplest static single page of plain text to the most complex web-based internet applications, electronic businesses, and social network services. A more comprehensive list of tasks to which web development commonly refers, may include web engineering, web design, web content development, client liaison, client-side/server-side scripting, web server and network security configuration, and e-commerce development. Among web professionals, "web development" usually refers to the main non-design aspects of building web sites: writing markup and coding. Most recently Web development has come to mean the creation of content management systems or CMS. These CMS can be made from scratch, proprietary or open source. In broad terms the CMS acts as middleware between the database and the user through the browser. A principle benefit of a CMS is that it allows non-technical people to make changes to their web site without having technical knowledge.

For larger organizations and businesses, web development teams can consist of hundreds of people (web developers) and follow standard methods like agile methodologies while developing websites. Smaller organizations may only require a single permanent or contracting developer, or secondary assignment to related job positions such as a graphic designer or information systems technician. Web development may be a collaborative effort between departments rather than the domain of a designated department. There are three kind of web developer specialization: front-end developer, back-end developer, and full-stack developer.

2.1 WEB-SITE

A **website** is a collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server. A website may be accessible via a public Internet Protocol (IP) network, such as the Internet, or a private local area network (LAN), by referencing a uniform resource locator (URL) that identifies the site.

Websites have many functions and can be used in various fashions; a website can be a personal website, a commercial website for a company, a government website or a non-profit organization website. Web sites are typically dedicated to a particular topic or purpose, ranging from entertainment and social networking to providing news and education. All publicly accessible websites collectively constitute the World Wide Web, while private websites, such as a company's website for its employees, and are typically a part of an intranet.

Web pages, which are the building blocks of websites, are documents, typically composed in plain text interspersed with formatting instructions of Hypertext Markup Language (HTML, XHTML).They may incorporate elements from other websites with Suitable markup anchors. Web pages are accessed and transported with the Hypertext Transfer Protocol (HTTP), which may optionally employ encryption (HTTP Secure, HTTPS) to provide security and privacy for the user. The user's application, often a web browser, renders the page content according to its HTML markup instructions onto a display terminal.

Hyper linking between web pages conveys to the reader the site structure and guides the navigation of the site, which often starts with a home page containing a directory of the site web content. Some websites require user registration or subscription to access content. Examples of subscription websites include many business sites, news websites, academic journal websites, gaming websites, file-sharing websites, message boards, web-based email, social networking websites, websites providing real-time stock market data, as well as sites providing various other services. As of 2016 end users can access websites on a range of devices, including desktop and laptop computers, tablet computers, smart phones and smart TVs

A web site consists of web pages which are interconnected to each other and contain various data and functionalities.

2.2 WEB-PAGE

A **web page**, or **webpage**, is a document that is suitable for the World Wide Web and web browsers. A web browser displays a web page on a monitor or mobile device. The web page is what displays, but the term also refers to a computer file, usually written in HTML or comparable markup language. Web browsers coordinate the various web resource elements for the written web page, such as style sheets, scripts, and images, to present the webpage.

Typical web pages provide hypertext that includes a navigation bar or a sidebar menu to other web pages via hyperlinks, often referred to as links.

On a network, a web browser can retrieve a web page from a remote web server. On a higher level, the web server may restrict access to only a private network such as a corporate intranet or it provides access to the World Wide Web. On a lower level, the web browser uses the Hypertext Transfer Protocol (HTTP) to make such requests.

A static web page is delivered exactly as stored, as web content in the web server's file system, while a dynamic web page is generated by a web application that is driven by server-side software or client-side scripting. Dynamic website pages help the browser (the client) to enhance the web page through user input to the server.

2.3. WEB BROWSER

A web browser is a software application for accessing information on the World Wide Web. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on a screen. As a client/server model, the browser is the client run on a computer or mobile device that contacts the Web server and requests information. The web server sends the information back to the browser which displays the results on the Internet-enabled device that supports a browser.

Web browsers are used on a range of devices, including desktops, laptops, tablets, and smart phones. In 2019, an estimated 4.3 billion people used a browser. The most used browser is Google Chrome, with a 64% global market share on all devices, followed by Safari with 18%.

2.4. MOBILE WEB BROWSER

There are a number of browsers that are designed to access the Web using a mobile device. A mobile browser, also called a micro browser, is optimized to display Web content on smaller mobile device screens and to perform efficiently on these computing devices, which have far less computing power and memory capacity as desktop or laptop. Mobile browsers are typically "stripped down" versions of Web browsers and offer fewer features in order to run well on mobile devices.

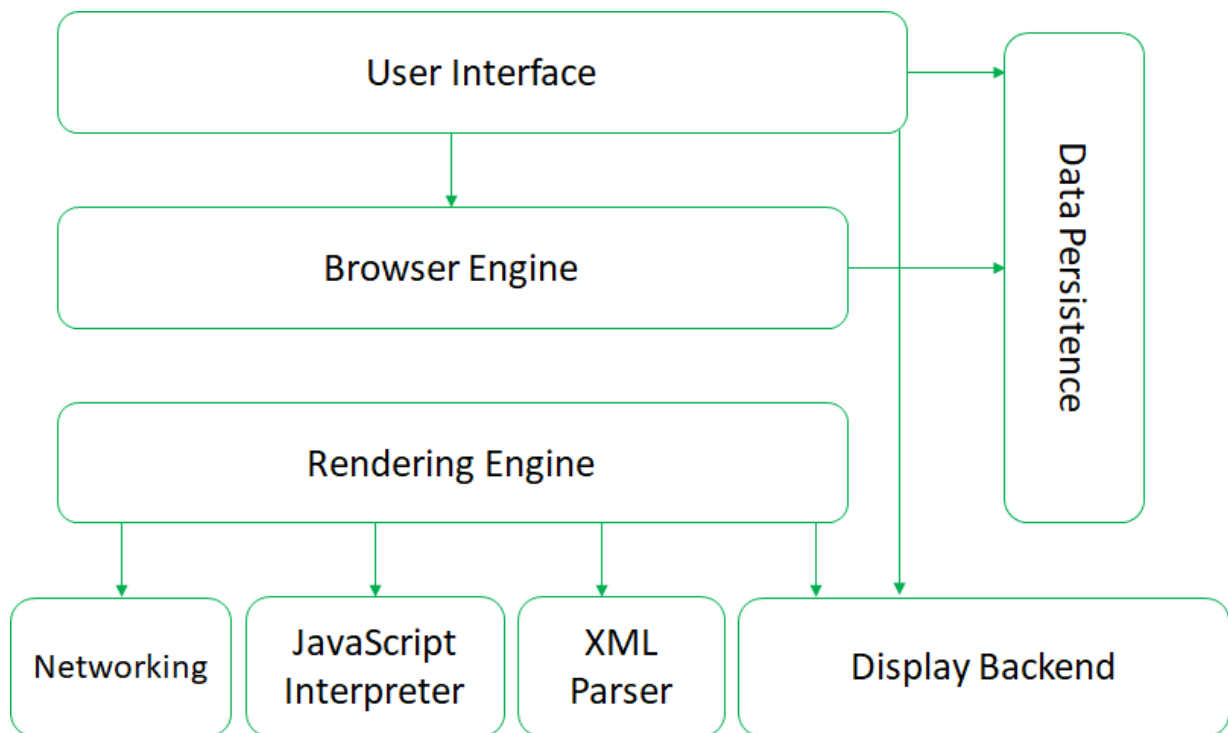


FIG 2.1 ARCHITECTURE OF WEB BROWSERS

CHAPTER-3

THE STEPS TO CREATE A WEB SITE

Creating a web site requires multiple steps which includes the following:

- Creating a UI(User interface)
- Scripting(Both at server end and client end)
- Creating a backend or the database

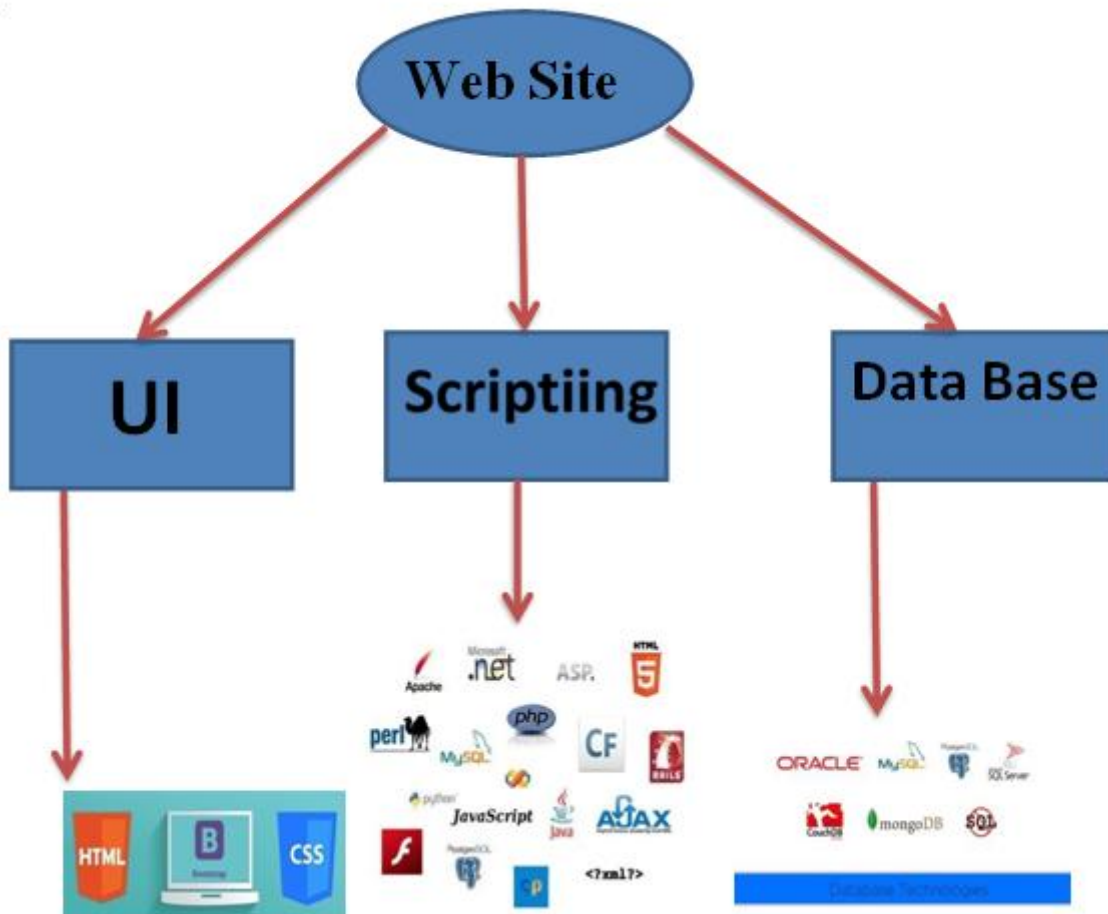


FIG 3.1 STEPS TO CREATE A WEB SITE

3.1 UI DEVELOPMENT

Technologies that are mostly used to develop a User Interface are:

- HTML
- CSS
- Bootstrap

3.1.1. HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the min to multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` introduce content into the page directly. Others such as `<p>...</p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references. HTML tags most commonly come in pairs like `<h1>` and `</h1>`, although some represent empty elements and so are unpaired, for example ``. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags).

Another important component is the HTML document type declaration, which triggers standards mode rendering.

The following is an example of the classic Hello world program, a common test employed for comparing programming languages, scripting languages and markup languages. This example is made using 9 lines of code:

General Syntax of HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

The text between `<html>` and `</html>` describes the web page, and the text between `<body>` and `</body>` is the visible page content. The mark up text "`<title>This is a title</title>`" defines the browser page title.

The Document Type Declaration `<!DOCTYPE html>` is for HTML5. If a declaration is not included, various browsers will revert to "quirks mode" for rendering.

HTML VERSION	YEAR
HTML 1.0	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML 5	2014

FIG 3.2 VERSIONS OF HTML

3.1.2. CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML 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 makes it possible to present the same mark up 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. It can also display the web page differently depending on the screen size or viewing device. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified.

Changes to the graphic design of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

Types of CSS:

- **Inline CSS:**

In this CSS is applied in between the tags

Eg: <tag style="styling">Hello World</tag>

- **Internal CSS:**

In this the css code is defined inside the style tag in the head section of the HTML page.

General Syntax:

```
<html>
  <head>
    <style>
      <!-- CSS STYLING -->
    </style>
  </head>
</html>
```

- **External CSS:**

In this the CSS code is written on another page and is linked to the HTML page. It is advantageous to use this type of styling as we can use the same file to style various HTML pages.

External CSS uses the extension .css and is applied using the following syntax

```
<html>
  <head>
    <link relation="style sheet" type="css" href="url to the page">
  </head>
</html>
```

All the CSS style types are important but can be used in different situations.

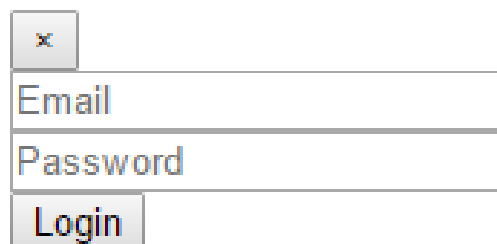
- Inline CSS is used when only small changes are to be done to the HTML tag and the changes are to be reflected only to that specific tag
- Internal CSS is used when the individual HTML page have to be designed differently. This also slows the page load system if the internal styling is long.
- External CSS files are maintained to design multiple pages and use common styles over various pages. It is useful as it helps in managing the resources in an easy manner.

Both HTML and CSS are used to create a UI but CSS behaves like a makeup on the face of an actress which makes her look even more beautiful than she is in reality.

And here is the difference:

Before using CSS in HTML page:

Enter your account details to login!



The image shows a basic, unstyled login form. At the top is a small square button with an 'x' icon. Below it are two text input fields, one labeled 'Email' and one labeled 'Password'. At the bottom is a 'Login' button. The form is composed of simple HTML elements without any CSS styling.

FIG 3.3 LOGIN FORM WITHOUT CSS

After using CSS in HTML Page:

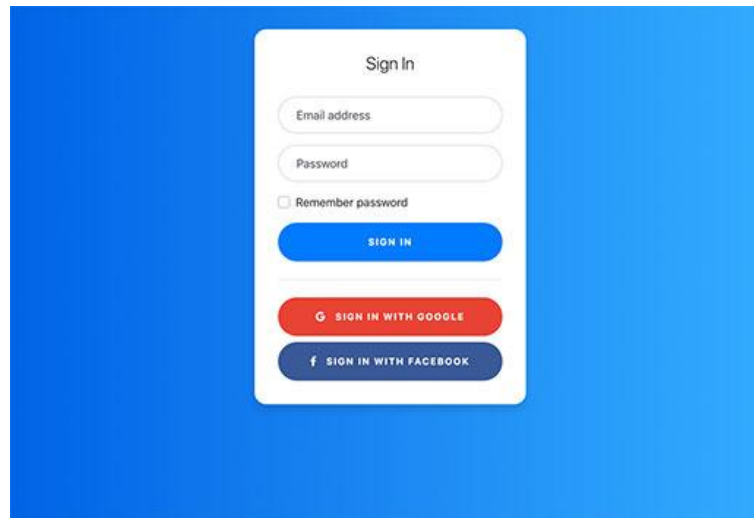


FIG 3.4 LOGIN FORM WITH CSS

3.1.3 BOOTSTRAP

Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Bootstrap is the second most-starred project on Git Hub, with more than 107,000 stars and 48,000 forks.

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. Before Bootstrap, various libraries were used for interface development, which led to inconsistencies and a high maintenance burden. According to twitter developer Mark Otto:

“A super small group of developers and I got together to design and build a new internal tool and saw an opportunity to do something more. Through that process, we saw ourselves build something much more substantial than another internal tool. Months later, we ended up with an early version of Bootstrap as a way to document and share common design patterns and assets within the company.”

After a few months of development by a small group, many developers at Twitter began to contribute to the project as a part of Hack Week, a hackathon-style week for the Twitter development team. It was renamed from Twitter Blue print to Bootstrap, and released a san.

Open source project on August19, 2011. It has continued to be maintained by Mark Otto, Jacob Thornton, and a small group of core developers, as well as a large community of contributor

On January 31, 2012, Bootstrap 2 was released, which added a twelve-column responsive grid layout system, in built support for Glyph icons, several new components, as well as changes to many of the existing components.

On August 19, 2013, Bootstrap 3 was released, which redesigned components to use flat design, and a mobile first approach.

On October 29, 2014, Mark Otto announced that Bootstrap 4 was in development. The first alpha version of Bootstrap 4 was released on August 19, 2015.

Bootstrap 3 supports the latest versions of the Google Chrome, Firefox, Internet Explorer, Opera, and Safari (except on Windows). It additionally supports back to IE8 and the latest Firefox Extended Support Release (ESR).

Since 2.0, Bootstrap supports responsive web design. This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone).

Starting with version 3.0, Bootstrap adopted a mobile-first design philosophy, emphasizing responsive design by default.

The version 4.0 alpha release added Sass and flex box support.

Installing and linking bootstrap to the HTML page

- Install bootstrap from <https://getbootstrap.com/>
- Copy the bootstrap.min.css file to your CSS folder and link it to the HTML page in the similar manner to how any other CSS file is linked.
- Link the bootstrap.min.js file which is present in the JS folder of the bootstrap. It can be linked using script tag.

Ex: `<script src="url to bootstrap.min.js"></script>`

- Now use bootstrap classes to reduce the work of designing which was earlier done through CSS.



FIG 3.5 LOGIN PAGE USING BOOTSTRAP

3.2 SCRIPTING

There are two scripting methodologies.

1. Server side scripting: This scripting is done at the server end
2. Client side scripting: This scripting is done at the client end or the browser.

3.2.1. SERVER SIDESCRIPTING

Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website. The alternative is for the web server itself to deliver a static web page. Scripts can be written in any of a number of server-side scripting languages that are available (see below). Server-side scripting is distinguished from client-side scripting where embedded scripts, such as JavaScript, are run client-side in a web browser, but both techniques are often used together.

Server-side scripting is often used to provide a customized interface for the user. These scripts may assemble client characteristics for use in customizing the response based on those characteristics, the user's requirements, access rights, etc. Server-side scripting also enables the website owner to hide the source code that generates the interface, where as with client-side scripting, the user has access to all the code received by the client. A down-side to the use of server-side scripting is that the client needs to make further requests over the network to the server in order to show new information to the user via the web browser. These requests can slow down the experience for the user, place more load on the server, and prevent use of the application when the user is disconnected from the server.

When the server serves data in a commonly used manner, for example according to the HTTP or FTP protocols, users may have their choice of a number of client programs (most modern web browsers can request and receive data using both of those protocols). In the case of more specialized applications, programmers may write their own server, client, and communications protocol that can only be used with one another.

Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be considered client-side operations.

3.2.1.1 Server Side scripting Languages

There are several languages that can be used for server-side programming:

- PHP
- ASP.NET (C# OR Visual Basic)
- C++
- Java and JSP
- Python
- Ruby on Rails

3.2.2 CLIENT SIDE SCRIPTING

Client-side scripting is changing interface behaviors within a specific web page in response to mouse or keyboard actions, or at specified timing events. In this case, the dynamic behavior occurs within the presentation. The client-side content is generated on the user's local computer system.

Such web pages use presentation technology called rich interfaced pages. Client-side scripting languages like JavaScript or Action Script, used for Dynamic HTML (DHTML) and Flash technologies respectively, are frequently used to orchestrate media types (sound, animations, changing text, etc.) of the presentation. Client-side scripting also allows the use of remote scripting, a technique by which the DHTML page requests additional information from a server, using a hidden frame, XML Http Requests, or a Web service.

The first widespread use of JavaScript was in 1997, when the language was standardized as ECMA Script and implemented in Netscape 3.

Example:

The client-side content is generated on the client's computer. The web browser retrieves a page from the server, then processes the code embedded in the page (typically written in JavaScript) and displays the retrieved page's content to the user. The most popularly used client side scripting languages is Java Script. Flow of request from browser to server:

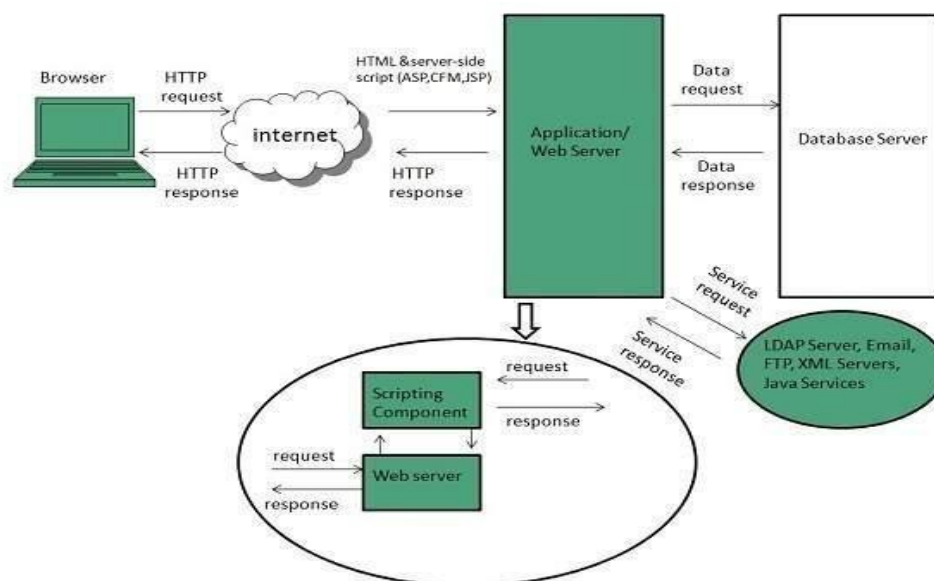


FIG 3.6 CLIENT SIDE SCRIPTING

3.3 DATABASE

A **database** is an organized collection of data. It is the collection of schemas, tables, queries, reports, views, and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

A **database management system (DBMS)** is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, MongoDB, Maria DB, Microsoft SQL Server, Oracle, Sybase, SAP HANA, MemSQL and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can inter operate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS. Database management systems are often classified according to the data base model that they support the most popular data base systems since the 1980s have all supported the relational model as represented by the SQL language. Sometimes a DBMS is loosely referred to as a "database".

3.3.1 ADVANTAGES OF DATA BASE

- Reduced data redundancy
- Reduced updating errors and increased consistency
- Greater data integrity and independence from applications programs
- Improved data access to users through use of host and query languages
- Improved data security
- Reduced data entry, storage, and retrieval costs
- Facilitated development of new applications program

3.3.2 DISADVANTAGES OF DATABASE

- Database systems are complex, difficult, and time-consuming to design
- Substantial hardware and software start-up costs
- Damage to database affects virtually all applications programs
- Extensive conversion costs in moving from a file-based system to a database system
- Initial training required for all programmers and users

3.4 SQL

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F.Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks." Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986 and of the International Organization for Standardization (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

3.5 QUERIES

The most common operation in SQL, the query, makes use of the declarative `SELECT` statement. `SELECT` retrieves data from one or more tables, or expressions. Standard `SELECT` statements have no persistent effects on the database. Some non-standard implementations of `SELECT` can have persistent effects, such as the `SELECT INTO` syntax provided in some databases.

Queries allow the user to describe desired data, leaving the database management system (DBMS) to carry out planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to include in the final result, normally immediately following the `SELECT` keyword. An asterisk ("`*`") can be used to specify that the query should return all columns of the queried tables. `SELECT` is the most complex statement in SQL, with optional keywords and clauses that include:

- The `FROM` clause, which indicates the table(s) to retrieve data from. The `FROM` clause can include optional `JOIN` sub clauses to specify the rules for joining tables.
- The `WHERE` clause includes a comparison predicate, which restricts the rows returned by the query. The `WHERE` clause eliminates all rows from the result set where the comparison predicate does not evaluate to `True`.
- The `GROUPBY` clause projects rows having common values into a smaller set of rows. `GROUP BY` is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The `WHERE` clause is applied before the `GROUP BY` clause.
- The `HAVING` clause includes a predicate used to filter rows resulting from the `GROUP BY` clause. Because it acts on the results of the `GROUP BY` clause, aggregation functions can be used in the `HAVING` clause predicate.
- The `ORDER BY` clause identifies which column[s] to use to sort the resulting data, and in which direction to sort them (ascending or descending). Without an `ORDERBY` clause, the order of rows returned by an SQL query is undefined.
- The `DISTINCT` keyword eliminates duplicate data.

CHAPTER-4

SCRIPTING LANGUAGES

4.1 Node JS

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

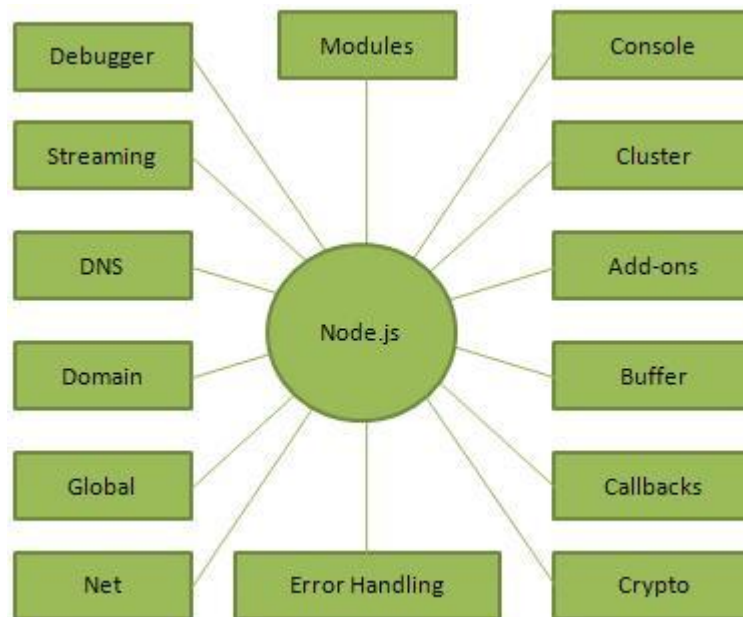


FIG 4.1 USES OF NODE JS

4.2. HISTORY

Node.js was written initially by Ryan Dahl in 2009, about thirteen years after the introduction of the first server-side JavaScript environment, Netscape's Livewire Pro Web. The initial release supported only Linux and Mac OS X. Its development and maintenance was led by Dahl and later sponsored by Joyent.

In June 2011, Microsoft and Joyent implemented a native Windows version of Node.js. The first Node.js build supporting Windows was released in July 2011.

4.3 JAVASCRIPT

JavaScript, often abbreviated as "JS", is a high-level, dynamic, untyped, and interpreted run-time language. It has been standardized in the ECMA Script language specification. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production; the majority of websites employ it, and all modern Web browsers support it without the need for plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two are distinct languages and differ greatly in their design. JavaScript was influenced by programming languages such as self and scheme.

JavaScript is also used in environments that are not Web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built-up on them have also increased the popularity of JavaScript for server-side Web applications. On the client side, developers traditionally implemented JavaScript as an interpreted language, but more recent browsers perform just-in-time compilation. Programmers also use JavaScript in video-game development, in crafting desktop and mobile applications, and in server-side network programming with run-time environments such as Node.js.



FIG 4.2 FRAME WORKS OF JAVA SCRIPT

4.4 JQUERY

Jquery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is free, open-source software using the permissive MIT license. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin.

Jquery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. Jquery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme able widgets. The modular approach to the jquery library allows the creation of powerful dynamic web pages and Web applications.

The set of jquery core features DOM element selections, traversal and manipulation enabled by its selector engine (named "Sizzle" from v1.3), created a new "programming style", fusing algorithms and DOM data structures. This style influenced the architecture of other JavaScript frameworks like YUI v3 and Dojo, later stimulating the creation of the standard Selectors API.

Microsoft and Nokia bundle jquery on their platforms. Microsoft includes it with Visual Studio for use within Microsoft's ASP.NET AJAX and ASP.NET MVC frameworks while Nokia has integrated it into the Web Run-Time widget development platform.

4.5 AJAX

Ajax (also **AJAX** short for "asynchronous JavaScript and XML") is a set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications. With Ajax, Web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows for Web pages, and by extension Web applications, to change content dynamically without the need to reload the entire page. In practice, modern implementations commonly substitute JSON for XML due to the advantages of being native to JavaScript.

Ajax is not a single technology, but rather a group of technologies. HTML and CSS can be used in combination to mark up and style information. The DOM is accessed with Java Script to dynamically display – and allow the user to interact with – the information presented. JavaScript and the XML Http Request object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

4.6 JSON

In computing, **JavaScript Object Notation** or **JSON**, is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser/server communication, including as a replacement for XML in some AJAX-style systems.

JSON is a language-independent data format. It was derived from JavaScript, but as of 2017 many programming languages include code to generate and parse JSON-format data. The official Internet media type for JSON is application/json. JSON filenames use the extension .json.

Douglas Crockford originally specified the JSON format in the early 2000s; two competing standards, RFC 7159 and ECMA-404, defined it in 2013. The ECMA standard describes only the allowed syntax, whereas the RFC covers some security and interoperability considerations.

I-JSON (short for "Internet JSON"), seeks to overcome some of the interoperability problems with JSON. It is defined in RFC7493.

4.7 XAMPP

Xampp is a free and open source cross platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, Maria DB database, and interpreters for scripts written in the PHP and Perl programming languages. XAMPP stands for Cross-Platform (X), Apache (A), Maria DB (M), PHP (P) and Perl (P). It is a simple, light weight Apache distribution that makes it extremely easy for developers to create a local web server for testing and deployment purposes. Everything needed to set up a web server – server application (Apache), database (Maria DB), and scripting language (PHP) – is included in an extractable file. XAMPP is also cross-platform, which means it works equally well on Linux, Mac and Windows. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server extremely easy as well

4.8 FEATURES

XAMPP is regularly updated to the latest releases of Apache, Maria DB, PHP and Perl. It also comes with a number of other modules including Open SSL, php My Admin, Media Wiki, Joomla, Word Press and more. Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another. XAMPP is offered in both a full and a standard version (Smaller version).

4.9 USAGE

Officially, XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the Internet. To make this as easy as possible, many important security features are disabled by default. XAMPP has the ability to serve web pages on the World Wide Web. A special tool is provided to password-protect the most important parts of the package.

XAMPP also provides support for creating and manipulating databases in MariaDB and SQLite among others. Once XAMPP is installed, it is possible to treat a local host like a remote host by connecting using an FTP client. Using a program like FileZilla has many advantages when installing a content management system (CMS) like Joomla or Word Press. It is also possible to connect to local host via FTP with an HTML editor.

CHAPTER-5

SOFTWARE REQUIREMENT SPECIFICATION

5.1 Hardware Requirements

The selection of hardware is very important in the existence and proper working of any software. When selecting hardware, the size and requirements are also important.

Processor	Intel CORE i5
RAM	4.0 GB
Hard Disk Drive	500 GB

5.2. Software Requirements

Number	Description
1	Windows 10
2	HTML/CSS/Ajax/JavaScript/ Bootstrap.
3	Apache server/ XAMPSERVER
4	PHP 5.5.38
4	MySQL
5	Compiler: MSVC11 (Visual C++ 2012)
6	Apache version: Apache/2.4.23 (Win32) Open SSL/1.0.2h PHP/5.5.38

CHAPTER-6

DATA FLOWDIAGRAM

Data Flow Diagrams show the flow of data from external entities into the system, and from one process to another within the system. There are four symbols for drawing a DFD:

- I. Rectangles representing external entities, which are sources or destinations of data.
- II. Ellipses representing processes, which take data as input, validate and process it and output it.
- III. Arrows representing the data flows, which can either, be electronic data or physical items.
- IV. Open-ended rectangles or a Disk symbol representing data stores, including electronic stores such as databases or XML files and physical stores such as filing cabinets or stacks of paper.

Figures below are the Data Flow Diagrams for the current system. Each process within the system is first shown as a Context Level DFD and later as a Detailed DFD. The Context Level DFD provides a conceptual view of the process and its surrounding input, output and data stores. The Detailed DFD provides a more detailed and comprehensive view of the interaction among the sub-processes within the system.

CONTEXT LEVEL D I A G R A M



FIG 6.1 CONTEXT LEVEL DIAGRAM

6.1 DFD-1

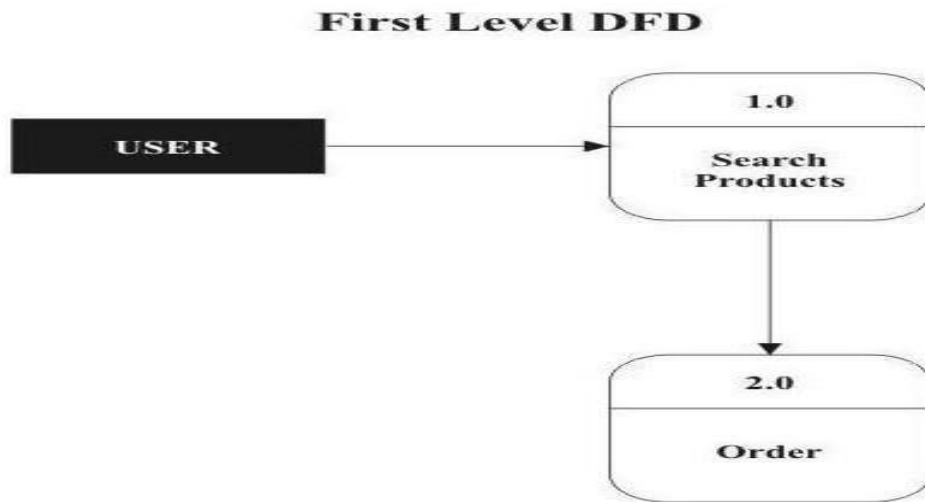


FIG 6.2 FIRST LEVEL DFD

6.2 DFD-2

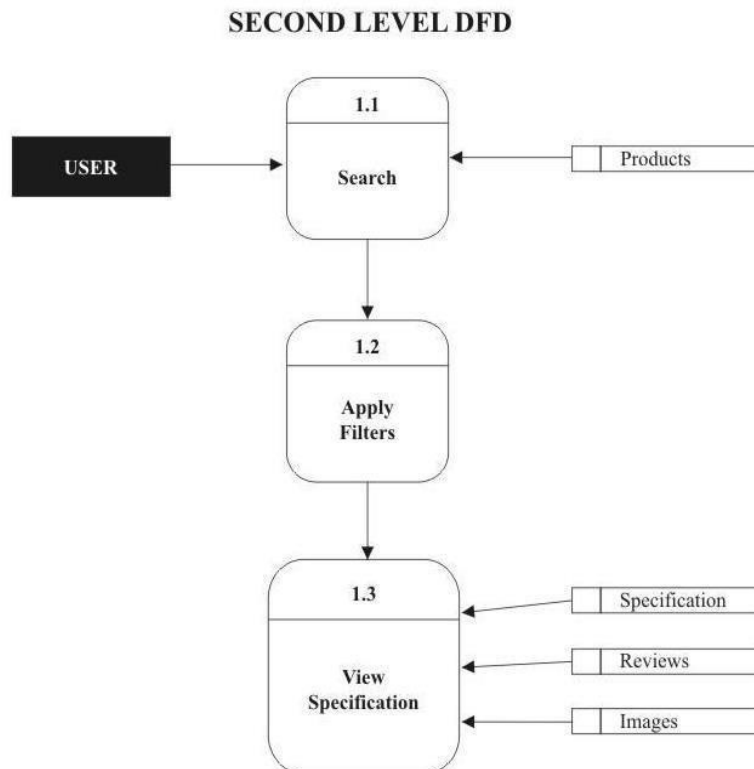


FIG 6.3 SECOND LEVEL DFD-1

SECOND LEVEL DFD

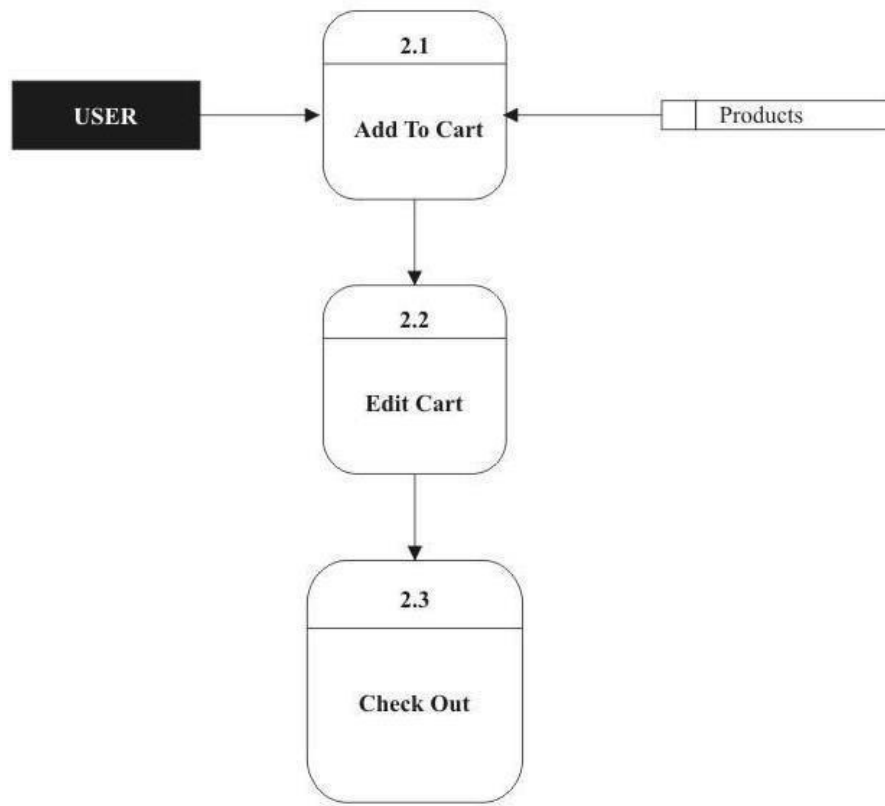


FIG 6.4 SECOND LEVEL DFD-2

6.3 DFD-3

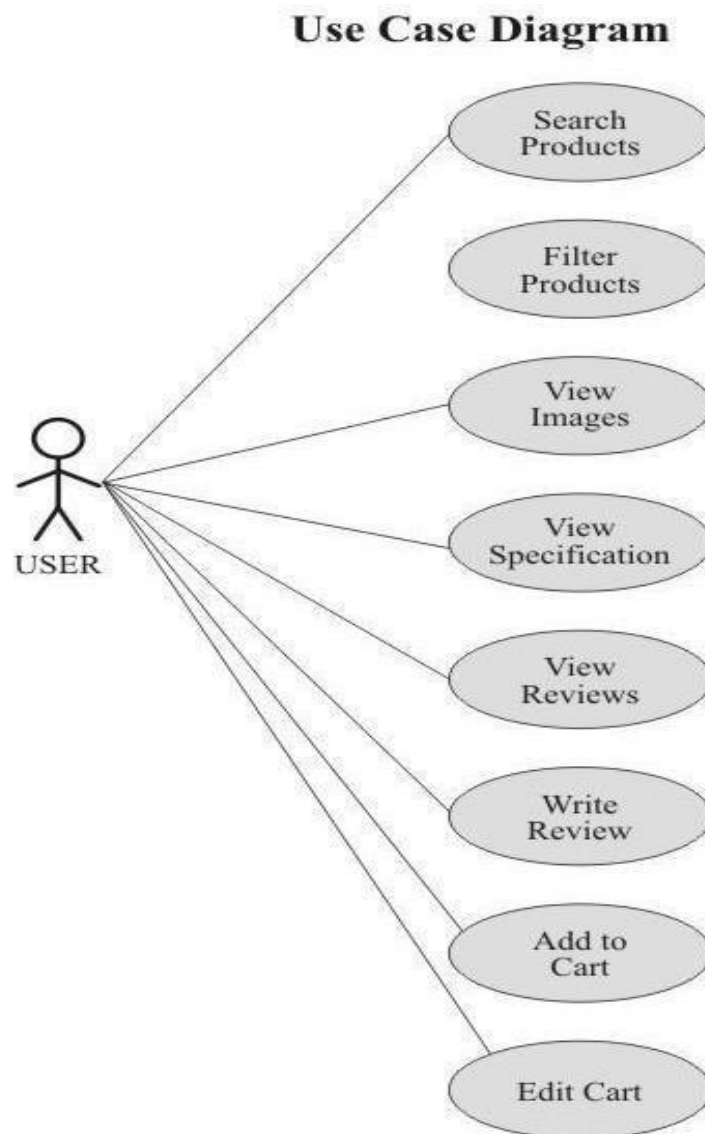


FIG 6.5 USE CASE DIAGRAM

CHAPTER-7

PROJECT

7.1 PROJECT (Advanced Technologies):

Name: Grocery Store

7.2 Technologies Used:

- HTML
- CSS
- Bootstrap
- NodeJS
- Java Script
- JQuery
- AJAX

Server: Apache (XAMPP)

Database: MySQL

Operating System: Windows 7/8/8.1/10

Wire framing tool: Balsamic

Team Size: 3

Name	Work Done
G Venu Prakash	Validations & Backend
J Satwik	Front End
K B Indraneel	Cart & Its Functionalities

7.3 TECHNICAL DETAILS:

- Front end is designed using HTML, CSS and Bootstrap. Ajax used to perform behind the screen requests and JavaScript used to perform client side scripting.
- Backend is based on NodeJS + MySQL based RDB (Relational Data Base) model.
- The SQL queries are run using the CI SQL library functions.
- Backend online host includes a centralized database resident on the server, the script uses SQL query to connect to the database on user's request for transaction of data.
- The forms are made using the HTML, Bootstrap for designing and NodeJS, SQL for back-end.
- JavaScript, AJAX and JQuery used for client side scripting and NodeJS for the server side development.

7.4 WIRE FRAMES

- Home

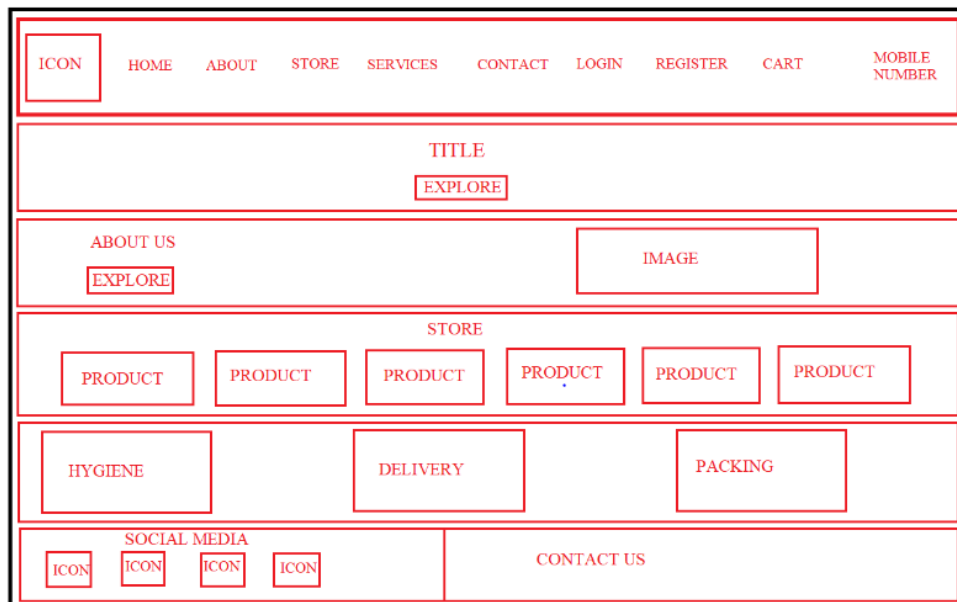


FIG 7.1 WIREFRAME FOR HOME PAGE

- Login Page

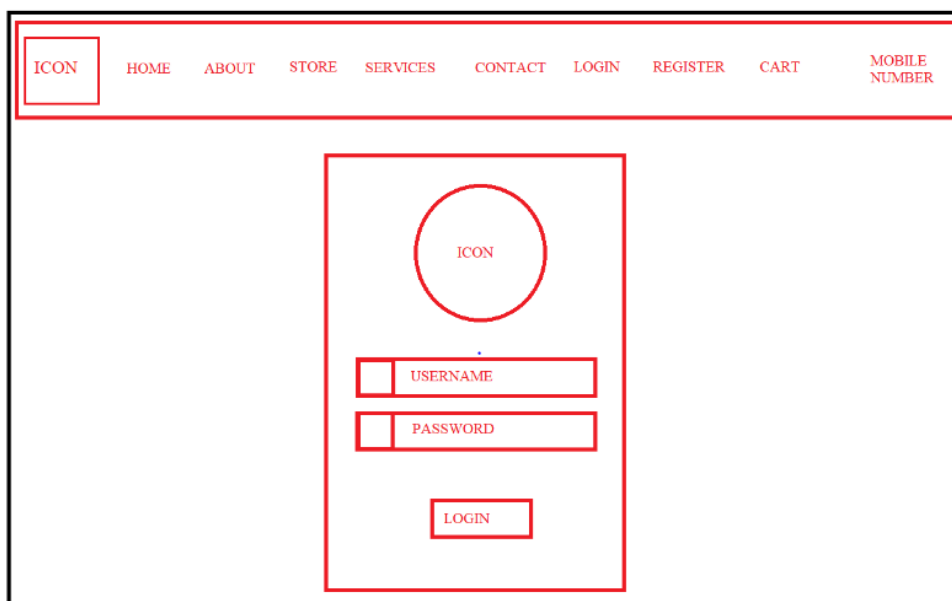
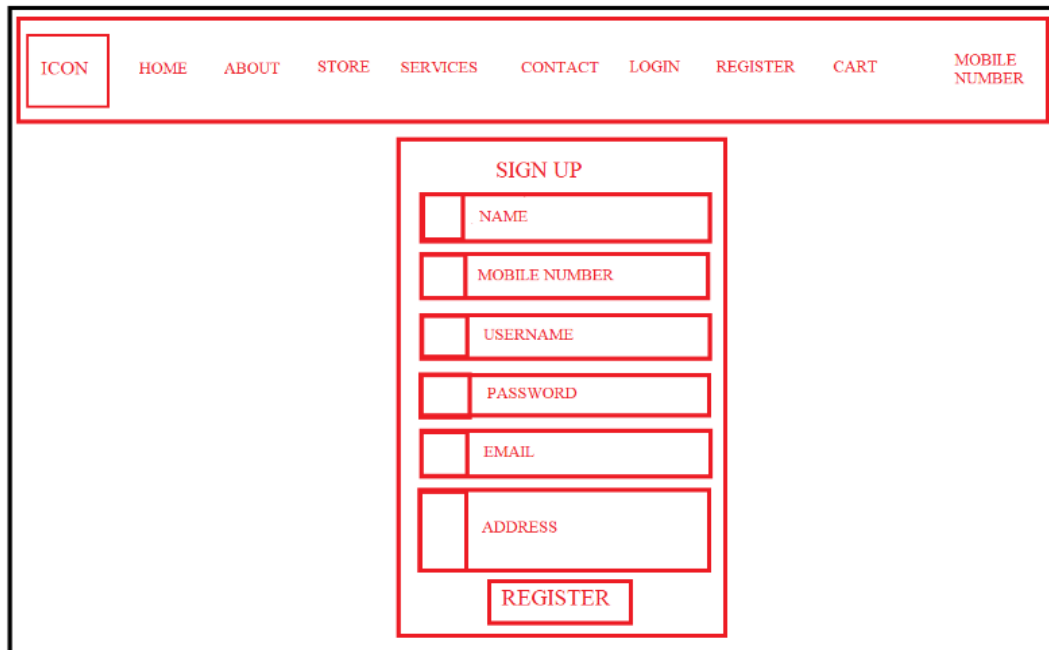


FIG 7.2 WIREFRAME FOR LOGIN PAGE

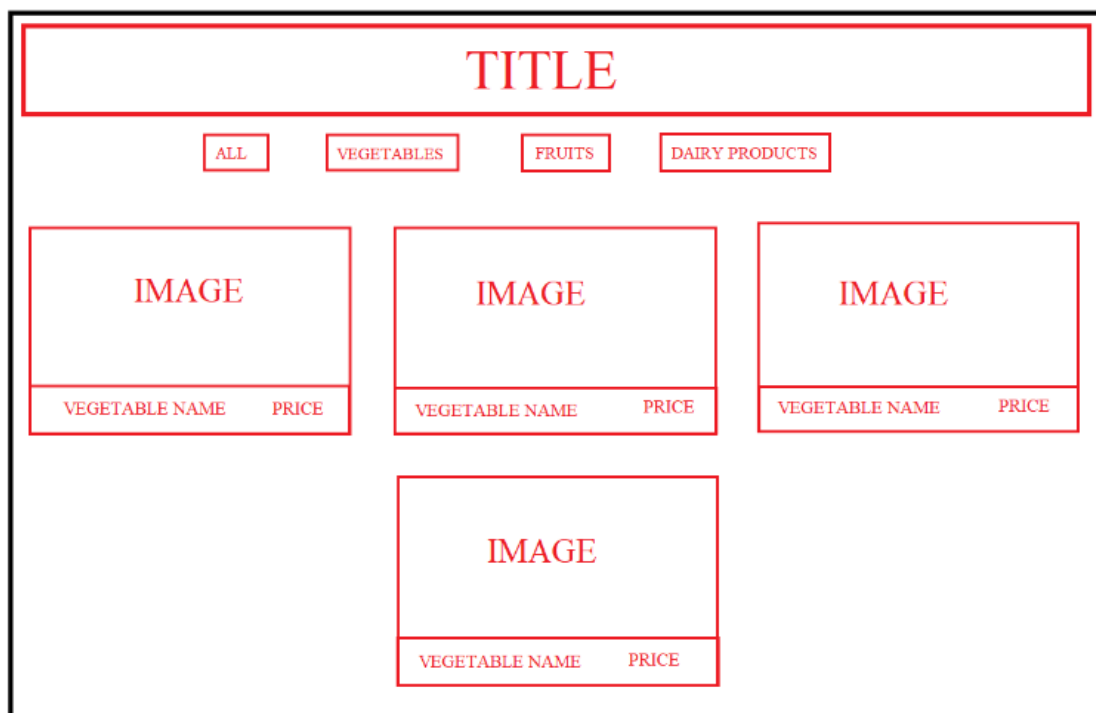
- **Registration Page**



The wireframe for the registration page features a top navigation bar with a red border. On the left is an 'ICON' placeholder, followed by links: HOME, ABOUT, STORE, SERVICES, CONTACT, LOGIN, REGISTER, and CART. On the right is a 'MOBILE NUMBER' placeholder. The main content area contains a 'SIGN UP' form with a red border. The form includes six input fields: NAME, MOBILE NUMBER, USERNAME, PASSWORD, EMAIL, and ADDRESS, each preceded by a small square icon. A 'REGISTER' button is positioned at the bottom of the form.

FIG 7.3 WIREFRAME FOR REGISTRATION PAGE

- **Vegetables**



The wireframe for the vegetables page has a red-bordered header with a 'TITLE' placeholder. Below the header is a category filter row with buttons: ALL, VEGETABLES, FRUITS, and DAIRY PRODUCTS. The main content area displays four product cards. Each card consists of an 'IMAGE' placeholder at the top and a table below. The table has two columns: 'VEGETABLE NAME' and 'PRICE'.

IMAGE	
VEGETABLE NAME	PRICE

IMAGE	
VEGETABLE NAME	PRICE

IMAGE	
VEGETABLE NAME	PRICE

IMAGE	
VEGETABLE NAME	PRICE

FIG 7.4 WIREFRAME FOR VEGETABLES PAGE

- **Fruits**

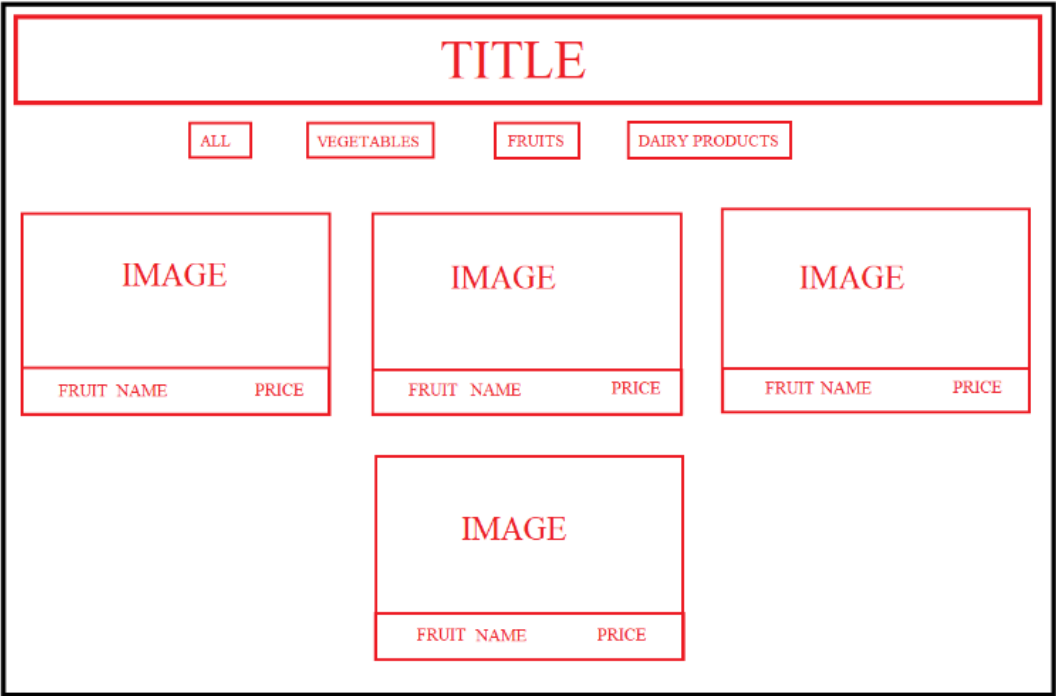


FIG 7.5 WIREFRAME FOR FRUITS PAGE

- **Dairy Products**

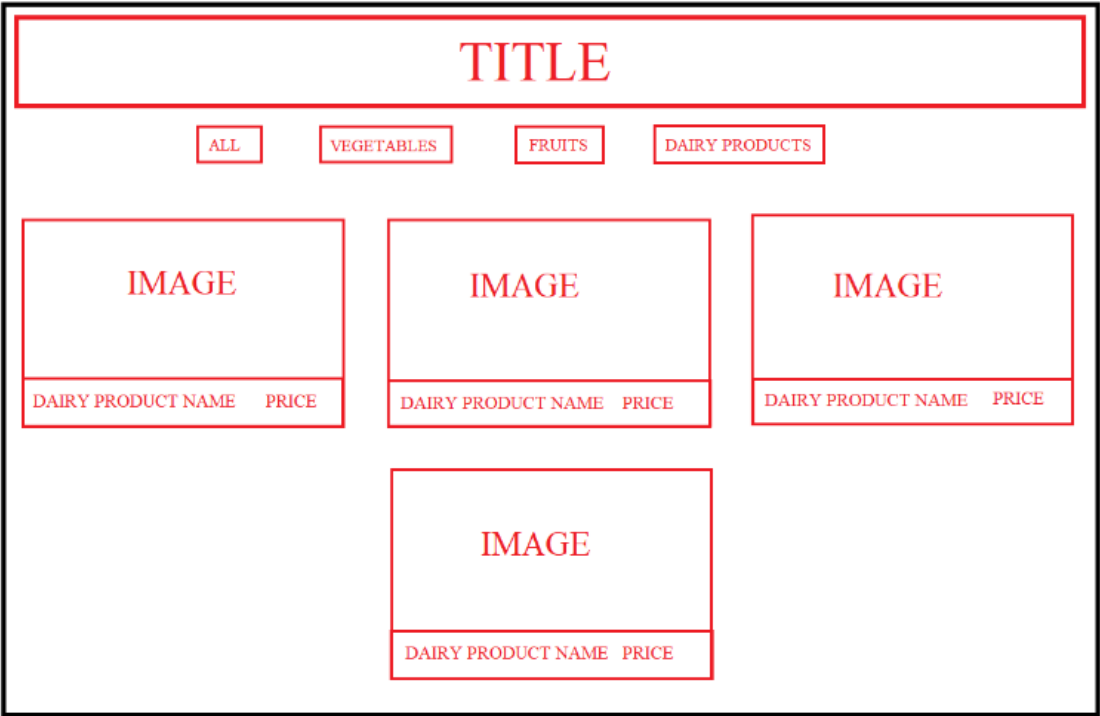


FIG 7.6 WIREFRAME FOR DAIRY PRODUCTS PAGE

- **Cart**

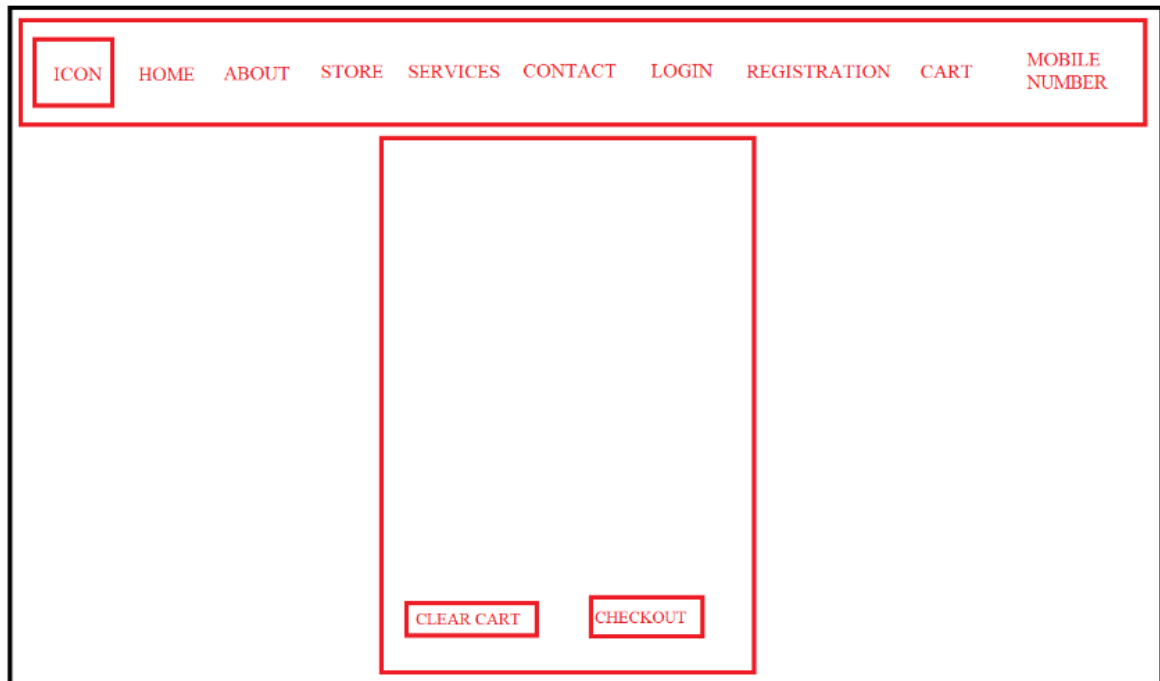


FIG 7.7 WIREFRAME FOR CART PAGE

- **Update password**

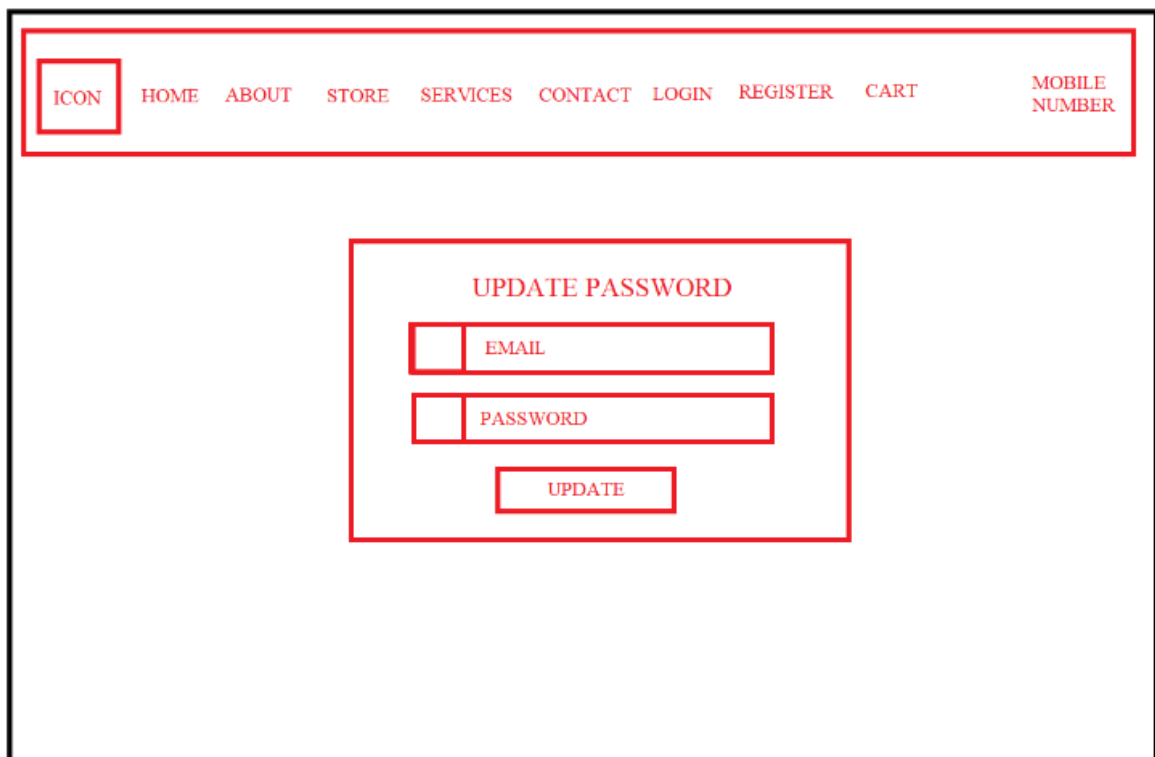


FIG 7.8 WIREFRAME FOR UPDATE PASSWORD PAGE

7.5 E-R Diagrams

- Class Diagram

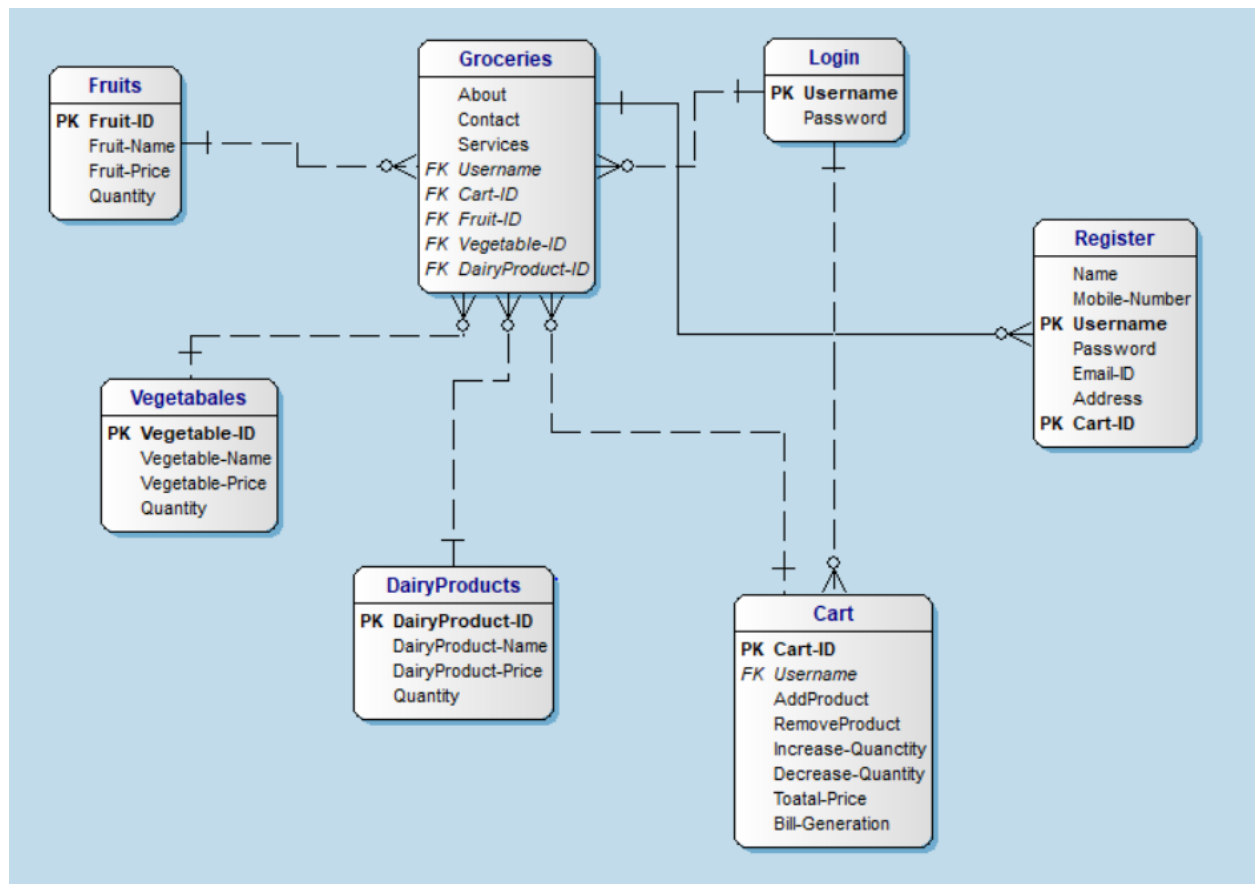


FIG 7.9 CLASS DIAGRAM FOR GROCERY STORE

7.6 SCREENSHOTS

- **CART ICON:** When the user clicks on the cart symbol which is blue in color as shown in the below picture the item gets added to the cart.

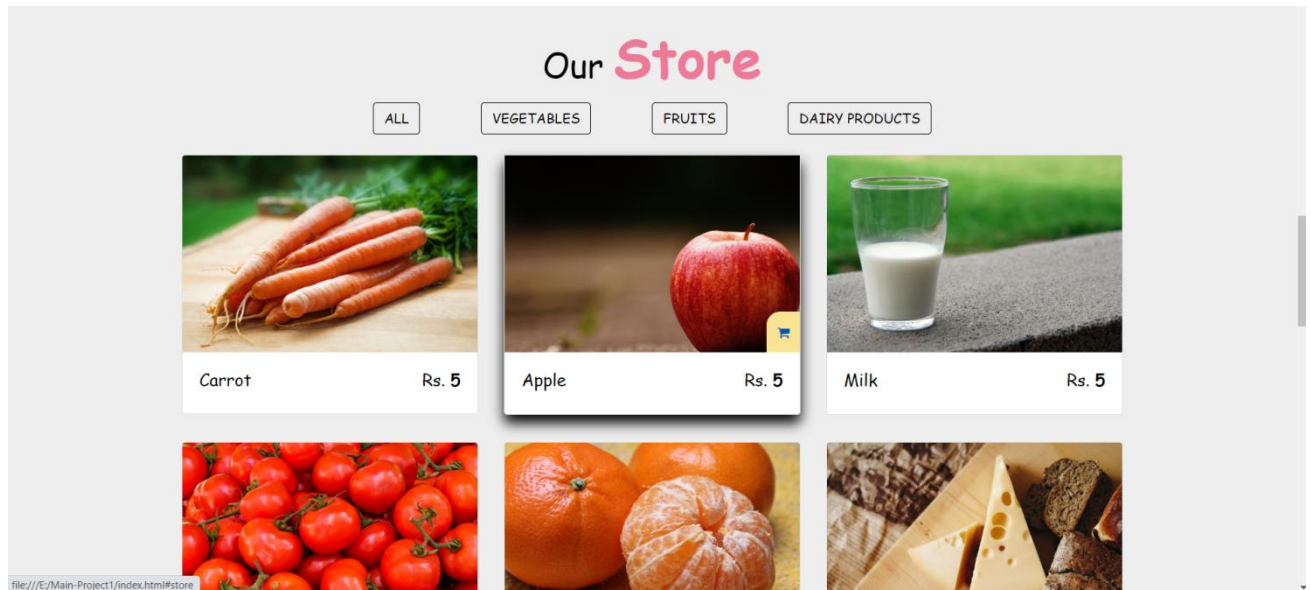


FIG 7.6.1 CART ICON

- Initially the cart is empty as shown in the picture.

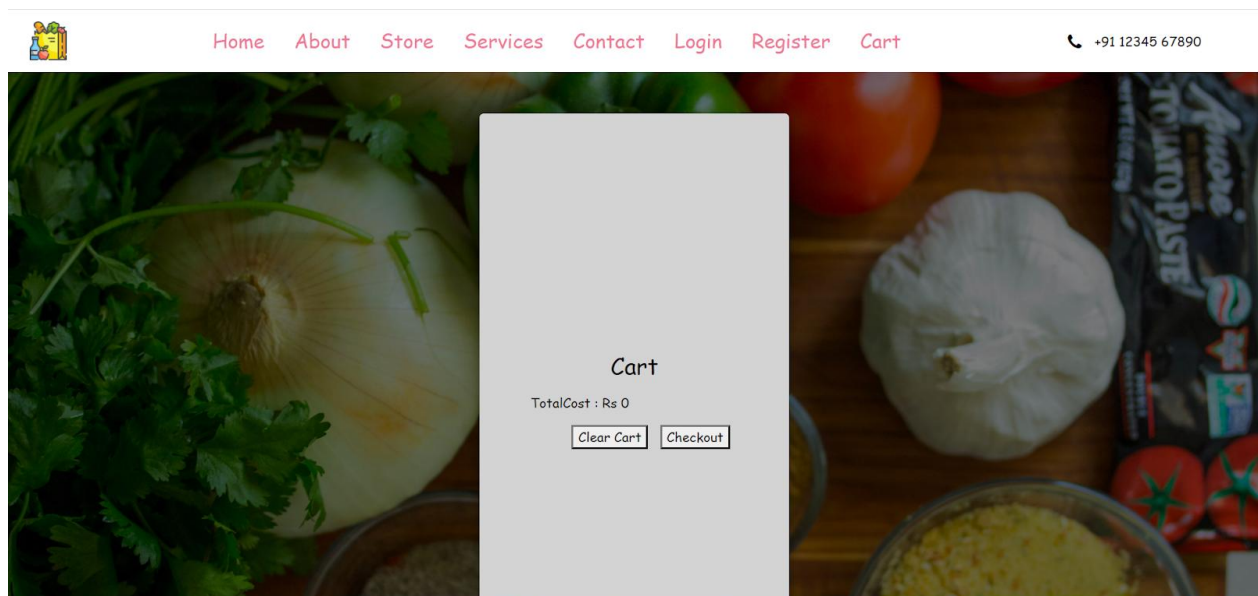


FIG 7.6.2 DEFAULT CART

- **ADD TO CART:** The user has added several items to the cart such as carrot, apple, orange, cheese, milk, tomato, grapes and banana. The cost of each product is shown separately in the same row while the total cost is shown at the bottom of the page.

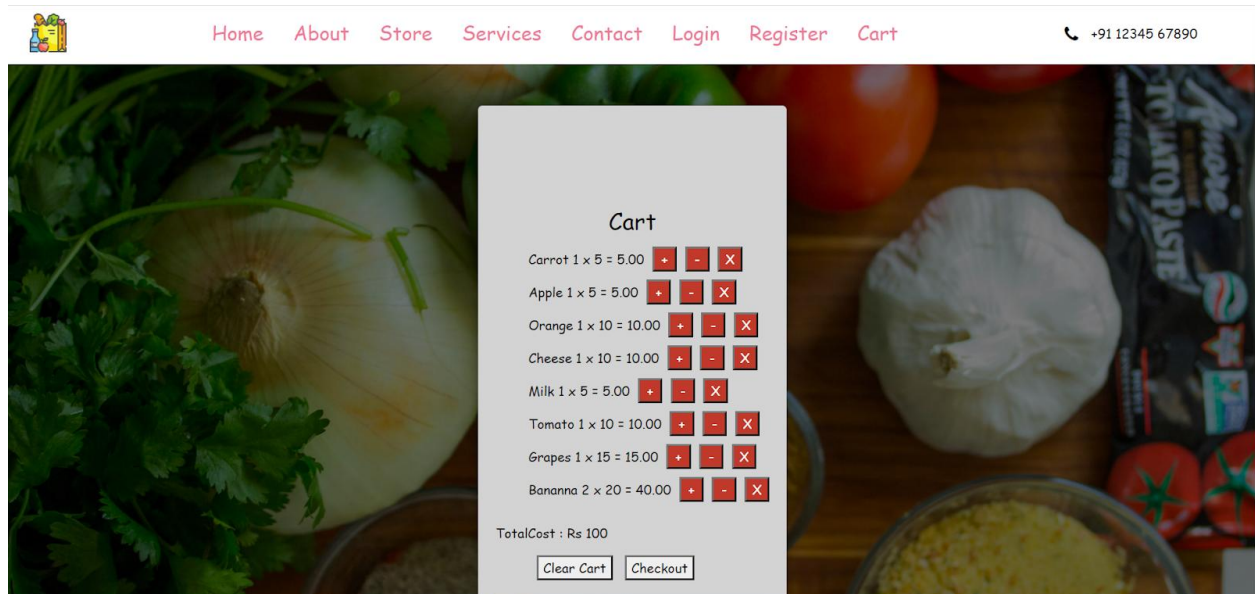


FIG 7.6.3 ADD TO CART

- “+”: Clicking on the plus button user can increase the number of items to be added to the cart. Here the user adds few carrots, apples, oranges by using the plus button.

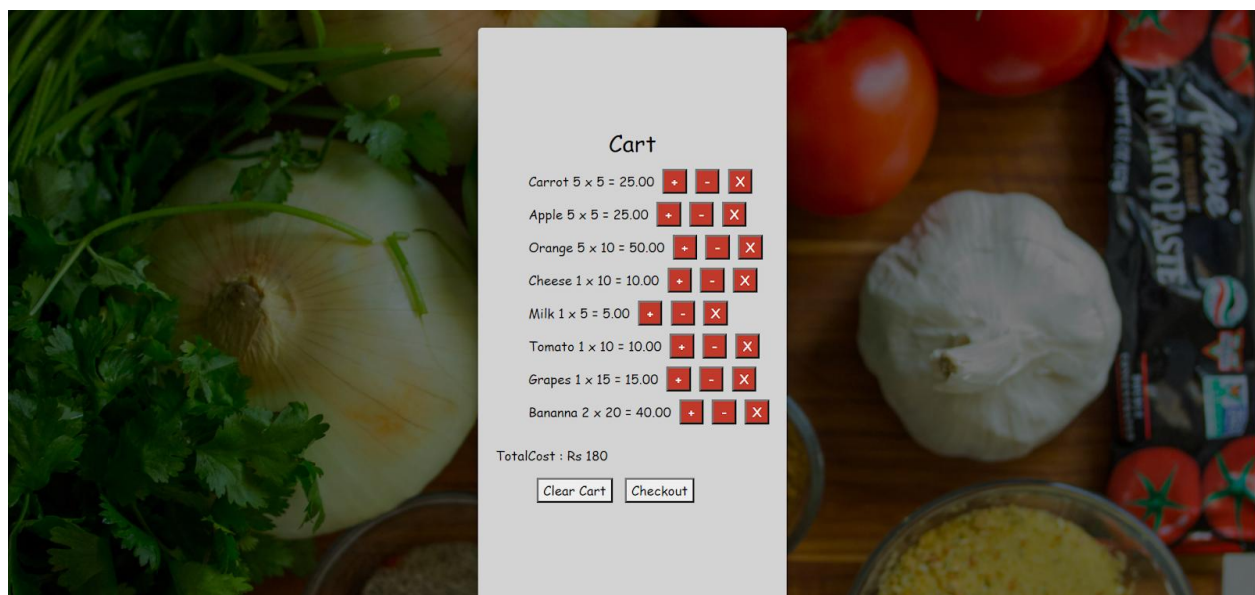


FIG 7.6.4 INCREASE THE QUANTITY

- “-”: Clicking on the minus button the user can reduce the number of items added to the cart. Here the user has reduced few items such as carrots, apples, orange using the minus button.

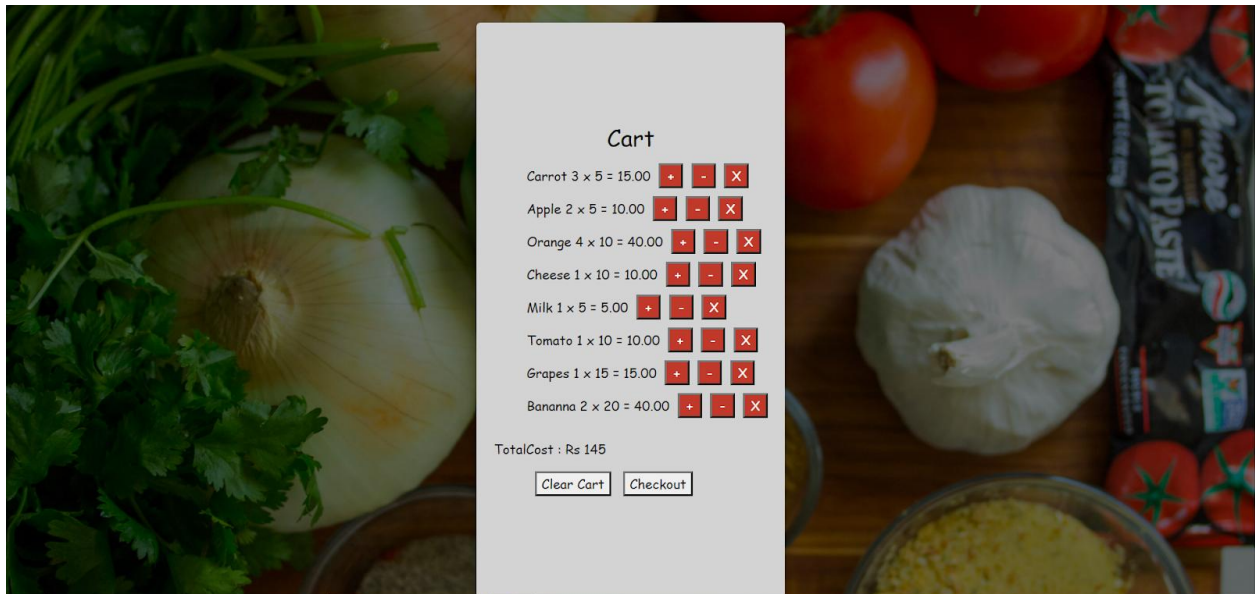
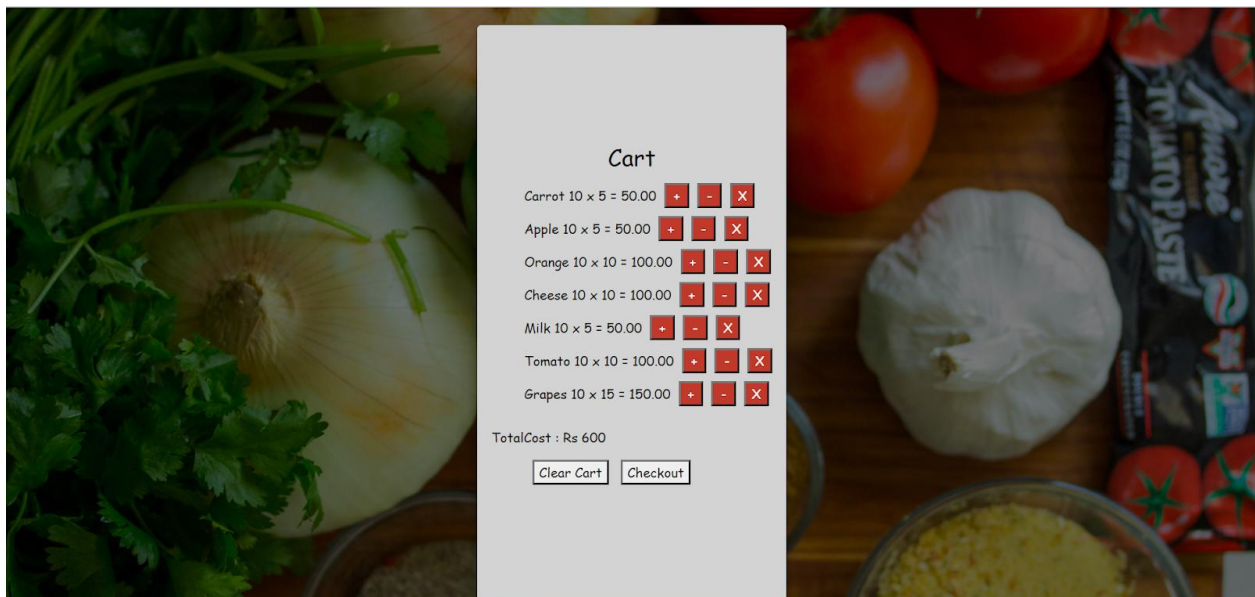


FIG 7.6.5 DECREASE THE QUANTITY

- “X”: Clicking on the cross button the user can completely remove the item from the cart irrespective of their number. Here the user removes bananas from the cart using the cross button.



FIGS 7.6.6 REMOVE THE PRODUCT

- **Total Cost:** It displays the cost of all the items placed in the cart by the user. The value is responsive and changes when the new items are added or removed and when the number of items are increased or decreased by the user.
- **Check Out:** When the user clicks on the check out button a message pops up saying that “Order has been placed” which confirms that the order has been placed by the user.

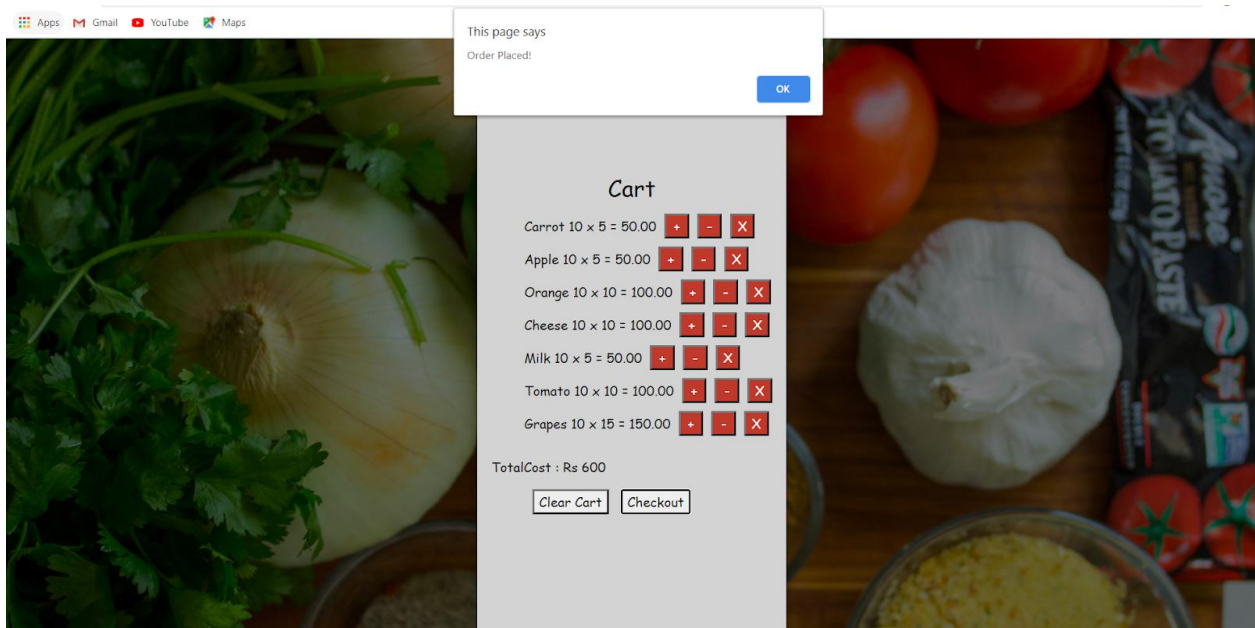


FIG 7.6.7 CHECKOUT CART

- **Clear cart:** When the user clicks on the clear cart button all the items present in the cart are removed and the cart becomes empty.

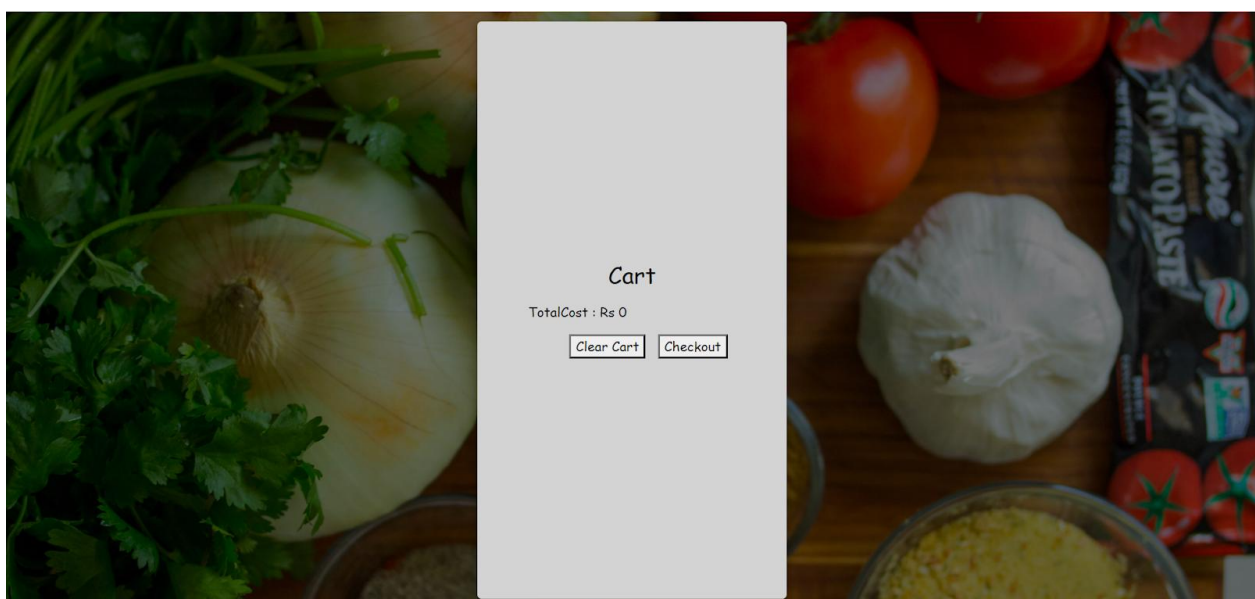


FIG 7.6.8 CLEAR CART

CHAPTER 8

ASSIGNED MODULES

Cart.html

```
<!DOCTYPE html>
<html>

  <head>
    <title>Grocery Store</title>
    <link rel="stylesheet" href="bootstrap-4.5.0-dist/css/bootstrap.min.css"/>
    <link rel="stylesheet" href="styles.css"/>
    <link rel="stylesheet" href="cart.css">
    <link rel="stylesheet" href="font-awesome-4.7.0/css/font-awesome.min.css"/>
    <script src="https://unpkg.com/ionicons@5.1.2/dist/ionicons.js"></script>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
  </head>
```

```
<body>
  <header id="header">
    <nav class="navbar navbar-expand-lg px-4">
      <a href="#header" class="navbar-brand">
        
      </a>
      <button type="button" class="navbar-toggler" data-toggle="collapse" data-target="#myNavbar">
        <span class="toggler-icon">
          <i class="fa fa-bars"></i>
        </span>
      </button>
      <div class="collapse navbar-collapse" id="myNavbar">
        <ul class="navbar-nav text-capitalize mx-auto">
          <li class="nav-item active">
            <a href="index.html" class="nav-link">Home</a>
          </li>
          <li class="nav-item active">
            <a href="index.html#about" class="nav-link">About</a>
          </li>
          <li class="nav-item active">
            <a href="index.html#store" class="nav-link">Store</a>
          </li>
          <li class="nav-item active">
            <a href="fruits.html#store" class="nav-link">Services</a>
          </li>
          <li class="nav-item active">
            <a href="index.html#contact" class="nav-link">Contact</a>
          </li>
          <li class="nav-item active">
            <a href="login.html" class="nav-link">Login</a>
          </li>
          <li class="nav-item active">
            <a href="register.html" class="nav-link">Register</a>
          </li>
          <li class="nav-item active">
            <a href="cart.html" class="nav-link">Cart</a>
          </li>
        </ul>
      </div>
    </nav>
  </header>
```

```

        <div class="nav-info-items d-none d-lg-flex">
            <div class="nav-info align-items-center d-flex justify-content-between mx-lg-5">
                <span class="info-icon mx-lg-3">
                    <i class="fa fa-phone fa-lg"></i>
                </span>
                <p class="mb-0">+91 12345 67890 </p>
            </div>
        </div>
    </nav>
</header>

<section>
    <div class="container h-100">
        <div class="d-flex justify-content-center h-100">
            <div class="user_card">
                <div class="d-flex justify-content-center form_container">
                    <form>
                        <h3>Cart</h3>
                        <ul id="show-cart">

                        </ul>
                        <div>TotalCost : Rs <span id="total-cart"></span></div>
                        <button class="buttons" id="clear-cart">Clear Cart</button>
                        <button class="buttons" id="submit-cart">Checkout</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</section>

```

All the below functions are enclosed with in a script tag in cart.html file.

Declaring the cart array.

```

var cart = [];

var Item = function(name, price, count){
    this.name = name
    this.price = price
    this.count = count
};

```

Add Item To Cart: This function helps us to add item to the cart.

```
function addItemToCart(name, price, count) {  
    for( var i in cart) {  
        if(cart[i].name == name) {  
            cart[i].count += count;  
            saveCart();  
            return;  
        }  
    }  
    var item = new Item(name, price, count);  
    cart.push(item);  
    saveCart();  
}
```

Remove Item From Cart: This function helps us to reduce the number of the items in the cart.

```
function removeItemFromCart(name) {  
    for ( var i in cart ) {  
        if(cart[i].name == name) {  
            cart[i].count--;  
            if(cart[i].count == 0) {  
                cart.splice(i, 1);  
            }  
            break;  
        }  
    }  
    saveCart();  
}
```

Remove Item From Cart All: This function helps us to remove the item from the cart.

```
function removeItemFromCartAll(name) {  
    for(var i in cart) {  
        if(cart[i].name == name) {  
            cart.splice(i, 1);  
            break;  
        }  
    }  
    saveCart();  
}
```


Clear Cart: This function helps to clear the cart.

```
function clearCart() {  
    cart = [];  
    saveCart();  
}
```

Count Cart: This function returns the count of each item in the cart.

```
function countCart() {  
    var totalCount = 0;  
    for (var i in cart) {  
        totalCount += cart[i].count;  
    }  
    return totalCount.toFixed(2); //precision  
}
```

Total Cart: This function returns the total cost of all the items in the cart.

```
function totalCart() {  
    var totalCost = 0;  
    for (var i in cart) {  
        totalCost += cart[i].price * cart[i].count;  
    }  
    return totalCost;  
}
```

Save Cart: This function helps to store the data into the local storage of the browser so that even when the browser is refreshed the content of the cart does not get changed.

```
function saveCart() {  
    localStorage.setItem("shoppingCart", JSON.stringify(cart));  
}
```

Load Cart: This function helps to load the data from the local storage of the browser so that even when the browser is refreshed the content of the cart gets does not get changed.

```
function loadCart() {  
    cart = JSON.parse(localStorage.getItem("shoppingCart"));  
}
```

List Cart: This function returns the list of all the items that are added to the cart.

```
function listCart() {  
    var cartCopy = [];  
    for(var i in cart) {  
        var item = cart[i];  
        var itemCopy = {};  
        for(var p in item) {  
            itemCopy[p] = item[p];  
        }  
        itemCopy.total = (item.price * item.count).toFixed(2);  
        cartCopy.push(itemCopy);  
    }  
    return cartCopy;  
}
```

On clicking the cart icon of the product image the function fetches data like name of the product and price of the product.

```
$(".add-to-cart").click(function(event){  
    event.preventDefault();  
    var name = $(this).attr("data-name");  
    var price = Number($(this).attr("data-price"));  
  
    addItemToCart(name, price, 1);  
    displayCart();  
});
```

On clicking + button the event increases the count of the product by one.

```
$("#show-cart").on("click", ".plus-item", function(event){  
    var name = $(this).attr("data-name");  
    addItemToCart(name, 0 ,1);  
    displayCart();  
});
```

On clicking - subtract button the event decreases the count of the product by one.

```
$("#show-cart").on("click", ".subtract-item", function(event){  
    var name = $(this).attr("data-name");  
    removeItemFromCart(name);  
    displayCart();  
});
```

On clicking X cross button the event removes the particular item from the cart.

```
$("#show-cart").on("click", ".delete-item", function(event){  
    var name = $(this).attr("data-name");  
    removeItemFromCartAll(name);  
    displayCart();  
});
```

On clicking **CLEAR CART** button all the items from the cart gets removed and it becomes empty.

```
$("#clear-cart").click(function(event) {  
    clearCart();  
    displayCart();  
});
```

On Clicking **CHECK OUT** button a pop message gets displayed saying “**Order Placed**”.

```
$("#submit-cart").click(function(){  
    window.alert("Order Placed!");  
    window.open("index.html");  
});
```

Display function is used to display the data like name, count, price, total cost of the products that are added to cart by the user.

```
function displayCart() {  
    var cartArray = listCart();  
    var output = "";  
    for(var i in cartArray) {  
        output += "<li class='display items'>"  
            +cartArray[i].name  
            + " " +cartArray[i].count  
            + " x " +cartArray[i].price  
            + " = " +cartArray[i].total  
            + " <button class='plus-item login_btn' data-name='" +cartArray[i].name+"'>+</button>"  
            + " <button class='subtract-item login_btn' data-name='" +cartArray[i].name+"'>-</button>"  
            + " <button class='delete-item login_btn' data-name='" +cartArray[i].name+"'>X</button>"  
            + "</li>"  
    }  
    $("#show-cart").html(output);  
    $("#total-cart").html(totalCart());  
}
```

CSS for Cart

```
.user_card {  
    height: 700px;  
    width: 500px;  
    margin-top: 50px;  
    margin-bottom: 50px;  
    background: lightgrey;  
    position: relative;  
    display: flex;  
    justify-content: center;  
    flex-direction: column;  
    padding: 10px;  
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);  
    border-radius: 5px;  
}  
  
.items{  
    padding-bottom: 10px;  
    word-spacing: 0.5rem;  
}
```

```

.login_btn {
  margin-left: 5px;
  width: 30px;
  background: #c0392b !important;
  color: white !important;
}

.login_btn:focus {
  box-shadow: none !important;
  outline: 0px !important;
}

.buttons#clear-cart{
  margin-top: 15px;
  margin-left: 50px;
}

.buttons{
  margin-top: 15px;
  margin-right: 10px;
}

section{
  min-height: calc(100vh - 76px);
  background: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('../Main Project/img/background1.jpg');
  background-position: center;
  background-size: cover;
  background-attachment: fixed;
  background-repeat: no-repeat;
}

h3{
  text-align: center;
  margin-bottom: 15px;
}

```

CHAPTER-9

MAINTENANCE

The maintenance phase involves making changes to hardware, software, and documentation to support its operational effectiveness. It includes making changes to improve a system's performance, correct problems, enhance security, or address user requirements. To ensure modifications do not disrupt operations or degrade a system's performance or security, organizations should establish appropriate change management standards and procedures.

Routine changes are not as complex as major modifications and can usually be implemented in the normal course of business. Routine change controls should include procedures for requesting, evaluating, approving, testing, installing, and documenting website modifications. Maintaining accurate, up-to-date hardware and software inventories is a critical part of all change management processes. Management should carefully document all modifications to ensure accurate system inventories. Management should coordinate all technology related changes through an oversight committee and assign an appropriate party responsibility for administering software patch management programs. Quality assurance, security, audit, regulatory compliance, network, and end-user personnel should be appropriately included in change management processes. Risk and security review should be done whenever a system modification is implemented to ensure controls remain in place.

For maintenance of the website:

1. The database has to be updated regularly according to new available information.
2. Redundant and false information must be removed from the database.
3. Newer version of PHP and MYSQL can be used for up gradation of website and to improve the overall performance of the system.

CHAPTER-10

FUTURE SCOPE & FUTURE ENHANCEMENT

PROJECT NAME: Grocery Store

1. Grocery Store would help each and every person to find any grocery item via our website and get it at home and this will save their time.
2. It would provide huge collection of grocery items of all fields.
3. We provide quality and fresh items.

CHAPTER-11

CONCLUSION

We have successfully implemented the site '**Grocer y Store**'. With the help of various links and tools, we have been able to provide a site which will be live soon and running on the web. We have been successful in our attempt to take care of the needs of both the user as well as the administrator. Finally we hope that this will go a long way in popularizing.

BIBLIOGRAPHY

1. www.javatutpoint.com
2. www.w3schools.com
3. www.getbootstrap.com
4. www.codeigniter.com
5. www.stackoverflow.com
6. www.fontawesome.io
7. www.php.net
8. Learn HTML and CSS faster(Mark Myers)
9. Wikipedia

GIT-HUB LINK: <https://github.com/Indraneelkatta>