



Dr. B. R. Ambedkar National Institute Technology, Jalandhar

**Project Report On
“Online Cake”
(Internship)**

Mentor: Hardeep kaur

**Submitted by:
Shivam Jaswal (20103133)
B.tech SEM-6TH
(NIT JALANDHAR)**

DECLARATION

I Beg **SHIVAM JASWAL** hereby declare that the Project Report entitled **“CAKEZONE WEBSITE”** submitted by me to **NIT JALANDHAR** is record of an original work done by me under the guidance of **HARDEEP KAUR**.

This Project report has been submitted in partial fulfillment of the requirement for the award of degree of **B.TECH**. This is my original work and the conclusions drawn therein are based on the material collected by myself. This project report has not been submitted to any other university or institute for the award of any degree or diploma.

ACKNOWLEDGEMENT

A Few typewritten words of thanks can-not really express the sincerity of my gratitude. But I am trying to put into words my gratefulness towards all who have helped & encourage me in carrying out the project.

I would like to thank my trainer **Mrs. HARDEEP KAUR** Who helped me throughout the project.

I would then like to thank my faculty guide, **Prof. NAME**, for all his valuable inputs and constant support towards me throughout my project and providing me an opportunity to learn outside the class room. It was a truly wonderful learning experience.

I would like to dedicate this project to my parents. Without their help and constant support this project would not have been possible. I would like to thank all my friends who did their valuable suggestions and support.

Last but not the least I would like to thank all the respondents who offered their opinions and suggestions and sometimes critical views throughout the survey which made me constantly update myself come out with a successful project.

Roll No: 20103133

Sudent Name: SHIVAM JASWAL

INDEX

<u>S.no</u>	<u>Topic</u>
1.	Introduction to Full Stack Development
2.	Introduction to MERN STACK
3.	Introduction about topics used in project
4.	Project Overview
5.	METHODOLOGY
6.	Project Overview
7.	Screenshots
8.	References

Introduction to Full Stack Development

Full-stack Development refers to the full study of a computer system application, and full-stack developers straddle two distinct areas of web development: front-end and back-end. The front end includes everything a customer, or web user, can see and interact with. In contrast, the back-end refers to all the servers, databases, and other internal architectures that drive the application; generally, the end user never interacts directly with this domain.

The easiest way to put the full stack into perspective is to imagine a restaurant. The front part includes the well-decorated and comfortable seating areas where visitors enjoy their food. The kitchen and pantry make up the “back-end” and are usually hidden from view from the customer.

Front end developers work to optimize the visible parts of an application for web browsers and mobile devices. Front end platforms are usually built with HTML, CSS, and JavaScript. Back end developers, in contrast, refine the software code that communicates with servers, databases, or other proprietary software that conveys information to front end interfaces.

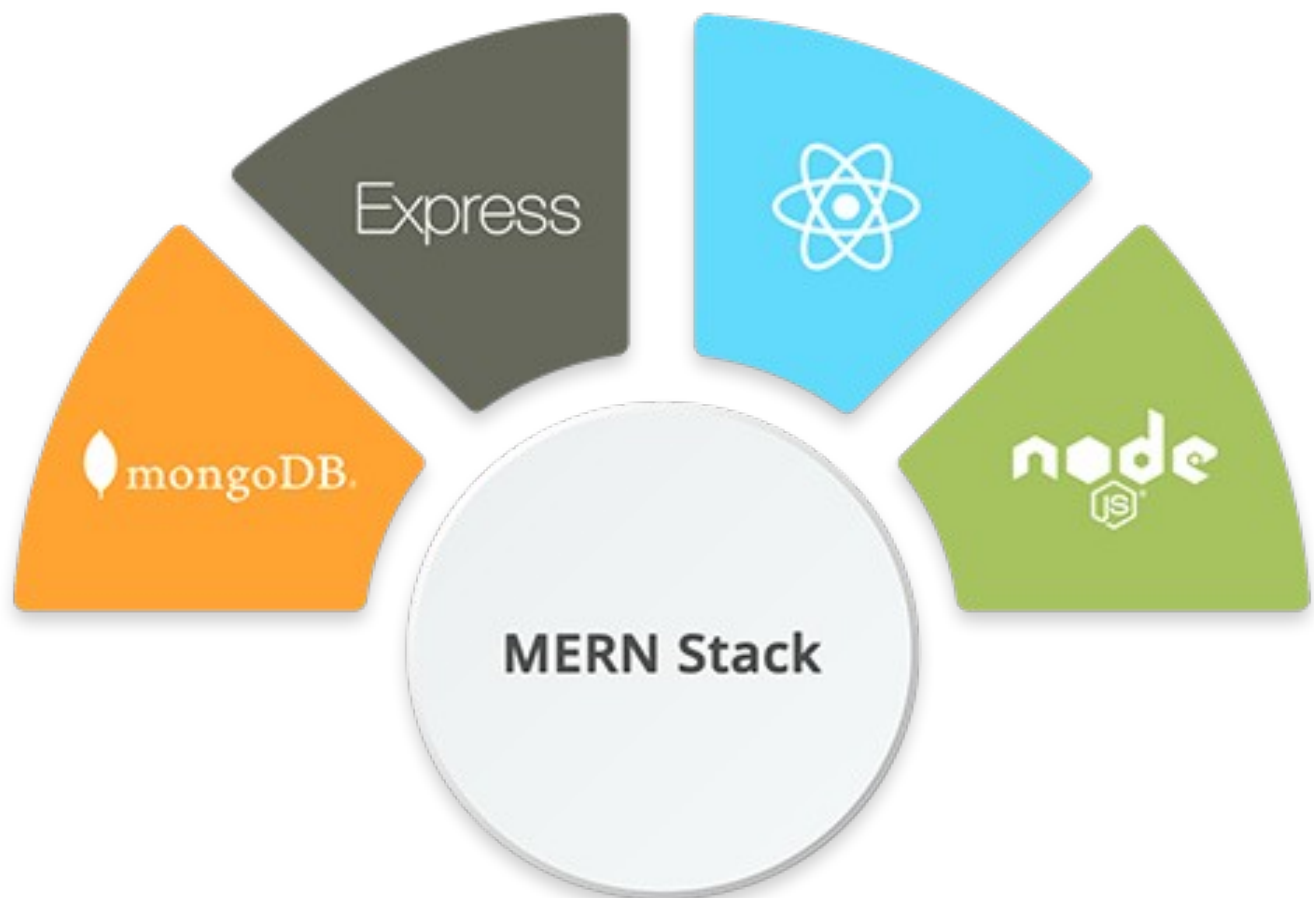
The modern full stack developer is an experienced generalist who can build a minimal viable product — i.e., an application with enough functionality to please early customers and spark feedback for continued development — on their own.

Companies rely on full stack professionals to spot errors between the front and back end and tackle tasks that straddle both disciplines. This versatility has become increasingly vital as newer apps begin to incorporate AI and other sophisticated technologies into their programming.

INTRODUCTION TO MERN Stack

MERN stack is a web development framework. It is a JavaScript Stack that is used for easier and faster deployment of full-stack web applications. It consists of Mongo DB, Express JS, React JS, and Node JS as its working components. It is designed to make the development process smoother and easier.

- **Mongo DB**
- **Express JS**
- **ReactJS:**
- **NodeJS:**



- **Mongo DB:**

Mongo DB is a No SQL database where each record is a document comprising of key-value pairs that are similar to JavaScript Object Notation (JSON) objects.

Mongo DB is flexible and allows its users to create schema, databases, tables, etc.

Documents that are identifiable by a primary key make up the basic unit of Mongo DB. Once Mongo DB is installed, users can make use of Mongo shell as well. Mongo shell provides a JavaScript interface through which the users can interact and carry out operations.

- Fast – Being a document-oriented database, easy to index documents.
Therefore a faster response.
- Scalability – Large data can be handled by dividing it into several machines.
- Use of JavaScript – MongoDB uses JavaScript which is the biggest advantage.
- Schema Less – Any type of data in a separate document.
- Data stored in the form of JSON –
 1. Objects, Object Members, Arrays, Values, and Strings
 2. JSON syntax is very easy to use.
 3. JSON has a wide range of browser compatibility.
 4. Sharing Data: Data of any size and type(video, audio) can be shared easily.
- Simple Environment Setup – Its really simple to set up MongoDB.

Features of MongoDB

Indexing

Fields in a MongoDB document can be indexed with primary and secondary indices or index.

Replication

MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica-set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally serve read operations, but that data is only eventually consistent by default.

If the replicated MongoDB deployment only has a single secondary member, a separate daemon called an *arbiter* must be added to the set. It has a single responsibility, which is to resolve the election of the new primary. As a consequence, an idealized distributed MongoDB deployment requires at least three separate servers, even in the case of just one primary and one secondary.

Load balancing

MongoDB scales horizontally using sharding. The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more replicas.) Alternatively, the shard key can be hashed to map to a shard – enabling an even data distribution.

MongoDB can run over multiple servers, balancing the load or duplicating data to keep the system up and running in case of hardware failure.

File storage

MongoDB can be used as a file system, called GridFS, with load balancing and data replication features over multiple machines for storing files.

This function, called grid file system, is included with MongoDB drivers.

MongoDB exposes functions for file manipulation and content to developers.

GridFS can be accessed using mongofiles utility or plugins

for Nginx and lighttpd. GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.

Aggregation

MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single-purpose aggregation methods.

Map-reduce can be used for batch processing of data and aggregation operations.

But according to MongoDB's documentation, the Aggregation Pipeline provides better performance for most aggregation operations.

The aggregation framework enables users to obtain the kind of results for which the SQL GROUP BY clause is used. Aggregation operators can be strung together to form a pipeline – analogous to Unix pipes. The aggregation framework includes the \$lookup operator which can join documents from multiple collections, as well as statistical operators such as standard deviation.

Server-side JavaScript execution

JavaScript can be used in queries, aggregation functions (such as MapReduce), and sent directly to the database to be executed.

Capped collections

MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a circular queue.

Transactions

MongoDB claims to support multi-document ACID transactions since the 4.0 release in June 2018. This claim was found to not be true as MongoDB violates snapshot isolation.

- **Express JS:**

Express is a Node.js framework. Rather than writing the code using Node.js and creating loads of Node modules, Express makes it simpler and easier to write the back-end code. Express helps in designing great web applications and APIs.

Express supports many middleware which makes the code shorter and easier to write.

- Asynchronous and Single-threaded.
- Efficient, fast & scalable
- Has the biggest community for Node.js
- Express promotes code reusability with its built-in router.
- Robust API

Frameworks built on Express

- [Feathers](#): Build prototypes in minutes and production ready real-time apps in days.
- [ItemsAPI](#): Search backend for web and mobile applications built on Express and Elasticsearch.

- [KeystoneJS](#): Website and API Application Framework / CMS with an auto-generated React.js Admin UI.
- [Poet](#): Lightweight Markdown Blog Engine with instant pagination, tag and category views.
- [Kraken](#): Secure and scalable layer that extends Express by providing structure and convention.
- [LoopBack](#): Highly-extensible, open-source Node.js framework for quickly creating dynamic end-to-end REST APIs.
- [Sails](#): MVC framework for Node.js for building practical, production-ready apps.
- [Hydra-Express](#): Hydra-Express is a light-weight library which facilitates building Node.js Microservices using ExpressJS.
- [Blueprint](#): a SOLID framework for building APIs and backend services
- [Loomotive](#): Powerful MVC web framework for Node.js from the maker of Passport.js
- [graphql-yoga](#): Fully-featured, yet simple and lightweight GraphQL server
- [Express Gateway](#): Fully-featured and extensible API Gateway using Express as foundation
- [Dinoloop](#): Rest API Application Framework powered by typescript with dependency injection
- [Kites](#): Template-based Web Application Framework
- [FoalTS](#): Elegant and all-inclusive Node.js web framework based on TypeScript.
- [NestJs](#): A progressive Node.js framework for building efficient, scalable, and enterprise-grade server-side applications on top of TypeScript & JavaScript (ES6, ES7, ES8)

- [Expressive Tea](#): A Small framework for building modifiable, clean, fast and descriptive server-side applications with Typescript and Express out of the box.

- **ReactJS:**

React is a JavaScript library that is used for building user interfaces. React is used for the development of single-page applications and mobile applications because of its ability to handle rapidly changing data. React allows users to code in JavaScript and create UI components.

React is a good fit for projects with multiple state changes that are intertwined and dependent on each other. Changes are tracked on the virtual DOM and then applied to the real DOM, ensuring that React uses the virtual DOM to keep track of changes in the application, then updates the real DOM with those changes.

Developers prefer it

While React comes with its drawbacks, developers around the world acknowledge its benefits and use it as often as possible. With the flexibility it provides, its low learning curve, its performance improvements, and its widespread popularity, it is clear why it is a favorite among many. React is a

solid, battle-tested library developers enjoy using because it provides the best tools for building effectively.

- **Virtual DOM** – A virtual DOM object is a representation of a DOM object. Virtual DOM is actually a copy of the original DOM. Any modification in the web application causes the entire UI to re-render the virtual DOM. Then the difference between the original DOM and this virtual DOM is compared and the changes are made accordingly to the original DOM.
- **JSX** – Stands for JavaScript XML. It is an HTML/XML JavaScript Extension which is used in React. Makes it easier and simpler to write React components.
- **Components** – ReactJS supports Components. Components are the building blocks of UI wherein each component has a logic and contributes to the overall UI. These components also promote code reusability and make the overall web application easier to understand.
- **High Performance** – Features like Virtual DOM, JSX and Components makes it much faster than the rest of the frameworks out there.

Examples of companies that use React.js

Some of the biggest companies in the world use React, including Facebook, Netflix, and Codecademy.

Facebook and its subsidiaries

Facebook is the earliest adopter of React and also where it was built, so it would be strange if React's parent company didn't use it. Besides that, WhatsApp and Instagram, subsidiaries of Facebook, made their web client applications with React and their mobile applications with React Native.

Netflix

Netflix wanted to provide their users with the best performance and rapidly integrate extensive testing into their components, so they decided to use React. Their engineering team shared how React helped them go fast and boosted their runtime performance and modularity, among other advantages. Read about it [here](#).

Codecademy

Codecademy overhauled its platform in 2014 and decided to use React. React's approach to handling DOM manipulation, its one-directional flow of data, and the rich developer experience it provides with JSX, were some reasons they chose React.

• NodeJS:

Node.js provides a JavaScript Environment which allows user to run their code on the server (outside the browser). Node pack manager i.e. npm allows user to choose from thousands of free packages to download.

- Open-source JavaScript Runtime Environment
- Single threading – Follows a single-threaded model.

- Data Streaming
- Fast – Built on Google Chrome's JavaScript Engine, Node.js has a fast code execution.
- Highly Scalable Initialize a Node.js application by typing running the below command in the command window. Accept the standard settings.

Why NodeJS

A common task for a web server can be to open a file on the server and return the content to the client.

Here is how PHP or ASP handles a file request:

1. Sends the task to the computer's file system.
2. Waits while the file system opens and reads the file.
3. Returns the content to the client.
4. Ready to handle the next request.

Here is how Node.js handles a file request:

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

Node.js eliminates the waiting, and simply continues with the next request.

Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

What Can Node.js Do?

- Node.js can generate dynamic page content

- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

Role of JavaScript in Web Designing

In November 1996, Netscape submitted JavaScript to Ecma International, as the starting point for a standard specification that all browser vendors could conform to. This led to the official release of the first ECMAScript language specification in June 1997.

The standards process continued for a few years, with the release of ECMAScript 2 in June 1998 and ECMAScript 3 in December 1999. Work on ECMAScript 4 began in 2000.

Meanwhile, Microsoft gained an increasingly dominant position in the browser market. By the early 2000s, Internet Explorer's market share reached 95%. This meant that JScript became the de facto standard for client-side scripting on the Web.

JavaScript is the dominant client-side scripting language of the Web, with 97% of websites using it for this purpose. Scripts are embedded in or included from HTML documents and interact with the DOM. All major web browsers have a built-in JavaScript engine that executes the code on the user's device.

Role Of Java Script in MERN STACK

JavaScript is the universal language that can be used across all software layers, so a person who applies it for both front end and back end programming is called a JavaScript full stack developer.

The fact that companies like Groupon, Airbnb, Netflix, Medium, and PayPal adopted the full stack JavaScript approach to build some of their products speaks for itself. However, small startups seem to enjoy using it as well. This is mostly due to the number of benefits full stack programming offers.

Common language, better team efficiency with fewer resources

Having all parts of your web application written in JavaScript allows for better understanding of the source code within the team. Therefore, there is no such thing as a gap between front and back end engineering that occurs when two teams are working separately using different technologies.

Extensive code reuse

With full stack JavaScript, you save time through code reuse and sharing.

Following the “don’t repeat yourself” ([DRY](#)) principle, you might be able to

reduce the effort by reusing the parts of the code (or sharing libraries, templates, and models) on both back and front end that are very close in terms of logic and implementation.

High performance and speed

Node.js uses an event-driven, non-blocking IO model that makes it lightweight and fast as compared to other commonly used back end technologies. To prove this, PayPal published a comprehensive [report](#) on the results they have seen in the process of migrating from Java to full stack JavaScript. The company was able to make the development almost 2 times faster while reducing the engineering personnel involved.

Large talent pool

According to the [Stack Overflow annual survey](#), JavaScript is the most popular programming language, used by 67.7 percent of the respondents. SlashData estimates that, as of October 2020, there are around [12.4 million](#) active JavaScript developers in the world.

Extensive knowledge base

Backed by giants like Facebook and Google, JavaScript has a powerful and fast-growing community. There are around 522,000 repositories on [Github](#) and over [2.2 million questions](#) tagged JavaScript on Stack Overflow, indicating the high activity of the developer community and the huge amount of valuable information that can be found out there.

Free, open source toolset

Most of the full stack [JavaScript development tools](#) are free or open source projects. This means you don't need to bear additional expenses for costly licenses or subscriptions. The tools that are open sourced are updated regularly and evolving fast due to the active community contributions. Instead of relying on a fixed set of technologies, you may use any of more than 1.3 million packages hosted by [npm](#), the largest JavaScript modules registry in the world, recently [acquired by Microsoft](#) and [integrated with GitHub](#).

COCEPTS OF JAVASCRIPT

Some of the basic Concepts of JavaScript which are important to learn

SCOPE - The scope is simply a box with a boundary for variables, functions, and objects. These boundaries put restrictions on variables and determine whether you have access to the variable or not. It limits the visibility or availability of a variable to the other parts of the code. You must have a clear understanding of this concept because it helps you to separate logic in your code and also improves readability. A scope can be defined in two ways –

- Local Scope allows access to everything within the boundaries (inside the box)

- Global Scope is everything outside the boundaries (outside the box). A global scope can not access a variable defined in the local scope because it is enclosed from the outer world, except if you return it.
- Block Scope is everything inside the boundaries but it works only for var and const keywords. It does not work with the var keyword.

IIFE (Immediately Invoked Function Expression)

As the name suggests IIFE is a function in Javascript which immediately invoked and executed as soon as it is defined. Variables declared within the IIFE cannot be accessed by the outside world and this way you can avoid the global scope from getting polluted. So the primary reason to use IIFE is to immediately execute the code and obtain data privacy.

Hoisting - A lot of developers get unexpected results when they are not clear about the concept of hoisting in Javascript. In Javascript, you can call a function before it is defined and you won't get an error 'Uncaught ReferenceError'. The reason behind this is hoisting where the Javascript interpreter always moves the variables and function declaration to the top of the current scope (function scope or global scope) before the code execution. Let's understand this with an example.

Closures - A closure is simply a function inside another function that has access to the outer function variable. Now, this definition sounds pretty much straightforward but the real magic is created with the scope. The inner function (closure) can access the variable defined in its scope (variables defined between its curly brackets), in the scope of its parent function, and in the global variables. Now

here you need to remember that the outer function can not have access to the inner function variable (we have already discussed this in the scope concept). Let's take an example and understand it in a better way.

Callbacks - In javascript, a callback is simply a function that is passed to another function as a parameter and is invoked or executed inside the other function. Here a function needs to wait for another function to execute or return a value and this makes the chain of the functionalities (when X is completed, then Y is executed, and it goes on.). This is the reason callback is generally used in the asynchronous operation of javascript to provide the synchronous capability.

Promises - We understand the concept of callback but what will happen if your code will have callbacks within callbacks within callbacks and it goes on? Well, this recursive structure of callback is called 'callback hell' and promises to help to solve this kind of issue. Promises are useful in asynchronous javascript operations when we need to execute two or more back-to-back operations (or chaining callback), where each subsequent function starts when the previous one is completed. A promise is an object that may produce a single value some time in the future, either a resolved value or a reason that it's not resolved (rejected). According to the developer.Mozilla "A Promise is an object representing the eventual completion or failure of an asynchronous operation. Essentially, a promise is a returned object to which you attach callbacks, instead of passing callbacks into a function.". Promises resolve the issue of 'callback hell' which is

nothing but a recursive structure of callbacks (callbacks within callbacks within callbacks and so forth).

A promise may be in three possible states...

Fulfilled: When the operation is completed successfully.

Rejected: When the operation is failed.

Pending: initial state, neither fulfilled nor rejected.

Async & Await - Stop and wait until something is resolved. Async & await is just syntactic sugar on top of Promises and like promises it also provides a way to maintain asynchronous operation more synchronously. So in javascript asynchronous operations can be handled in various versions...

ES5 -> Callback

ES6 -> Promise

ES7 -> async & await

You can use Async/Await to perform the Rest API request where you want the data to fully load before pushing it to the view. For Nodejs and browser programmers async/await is a great syntactic improvement. It helps the developer to implement functional programming in javascript and it also increases the code readability.

JavaScript Frameworks

The JavaScript testing framework is a dynamic framework based on JS, which is well known for its ease of use in front-end and back-end development. These transitions over time also result in the need for excellent testing tools.

- 1) Mocha JS
- 2) JEST
- 3) Jasmine
- 4) Karma
- 5) Puppeteer
- 6) Nightwatch JS
- 7) Cypress

FRONT END DEVELOPMENT

Role of HTML in Web Designing

Hypertext Mark-up Language, or HTML, is a programming language used to describe the structure of information on a web page. The HyperText Markup Language or HTML is the standard mark-up language for documents designed to be displayed in a web browser.

Together, HTML, CSS, and JavaScript make up the essential building blocks of websites, with CSS controlling a page's appearance, and JavaScript programming its functionality. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. You can think of HTML as providing the bones of a web page, while CSS provides the skin, and JavaScript provides the brains.

A web page can contain headings, paragraphs, images, videos, and many other types of data. Front-end developers use HTML elements to specify what kind of information each item on a web page contains — for instance, the “p” element indicates a paragraph. Developers also write HTML code to specify how different items relate to one another in the overall structure of the page.

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. HTML 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 />` directly introduce content into the page.

Every website you open in your browser, from social networks to music services, uses HTML. A look under the hood of any website would reveal HTML code providing structure for all the page's components.

Role of CSS in Web Designing

Cascading Style Sheets, commonly known as CSS, is an integral part of the modern web development process. It is a highly effective HTML tool that provides easy control over layout and presentation of website pages by separating content from design.

The standout advantage of CSS is the added design flexibility and interactivity it brings to web development. Developers have greater control over the layout allowing them to make precise section-wise changes.

As customization through CSS is much easier than plain HTML, web developers are able to create different looks for each page. Complex websites with uniquely presented pages are feasible thanks to CSS.

CSS works by creating rules. These rules are simultaneously applied to multiple elements within the site. Eliminating the repetitive coding style of HTML makes development work faster and less monotonous. Errors are also reduced considerably. Since the content is completely separated from the design, changes across the website can be implemented all at once. This reduces delivery times and costs of future edits.

Improved website loading is an underrated yet important benefit of CSS. Browsers download the CSS rules once and cache them for loading all the pages of a website. It makes browsing the website faster and enhances the overall user experience.

Role of JavaScript in Web Designing

In November 1996, Netscape submitted JavaScript to Ecma International, as the starting point for a standard specification that all browser vendors could conform to. This led to the official release of the first ECMAScript language specification in June 1997.

The standards process continued for a few years, with the release of ECMAScript 2 in June 1998 and ECMAScript 3 in December 1999. Work on ECMAScript 4 began in 2000.

Meanwhile, Microsoft gained an increasingly dominant position in the browser market. By the early 2000s, Internet Explorer's market share reached 95%. This meant that JScript became the de facto standard for client-side scripting on the Web.

JavaScript is the dominant client-side scripting language of the Web, with 97% of websites using it for this purpose. Scripts are embedded in or included from HTML documents and interact with the DOM. All major web browsers have a built-in JavaScript engine that executes the code on the user's device.

Role Of Java Script in MERN STACK

JavaScript is the universal language that can be used across all software layers, so a person who applies it for both front end and back end programming is called a JavaScript full stack developer.

The fact that companies like Groupon, Airbnb, Netflix, Medium, and PayPal adopted the full stack JavaScript approach to build some of their products speaks

for itself. However, small startups seem to enjoy using it as well. This is mostly due to the number of benefits full stack programming offers.

Common language, better team efficiency with fewer resources

Having all parts of your web application written in JavaScript allows for better understanding of the source code within the team. Therefore, there is no such thing as a gap between front and back end engineering that occurs when two teams are working separately using different technologies.

Extensive code reuse

With full stack JavaScript, you save time through code reuse and sharing.

Following the “don’t repeat yourself” (DRY) principle, you might be able to reduce the effort by reusing the parts of the code (or sharing libraries, templates, and models) on both back and front end that are very close in terms of logic and implementation.

High performance and speed

Node.js uses an event-driven, non-blocking IO model that makes it lightweight and fast as compared to other commonly used back end technologies. To prove this, PayPal published a comprehensive report on the results they have seen in the process of migrating from Java to full stack JavaScript. The company was able to make the development almost 2 times faster while reducing the engineering personnel involved.

Large talent pool

According to the Stack Overflow annual survey, JavaScript is the most popular programming language, used by 67.7 percent of the respondents. SlashData estimates that, as of October 2020, there are around 12.4 million active JavaScript developers in the world.

Extensive knowledge base

Backed by giants like Facebook and Google, JavaScript has a powerful and fast-growing community. There are around 522,000 repositories on Github and over 2.2 million questions tagged JavaScript on Stack Overflow, indicating the high activity of the developer community and the huge amount of valuable information that can be found out there.

Free, open source toolset

Most of the full stack JavaScript development tools are free or open source projects. This means you don't need to bear additional expenses for costly licenses or subscriptions. The tools that are open sourced are updated regularly and evolving fast due to the active community contributions. Instead of relying on a fixed set of technologies, you may use any of more than 1.3 million packages hosted by npm, the largest JavaScript modules registry in the world, recently acquired by Microsoft and integrated with GitHub.

Why choose the MERN stack?

Let's start with Mongo DB, the document database at the root of the MERN stack. Mongo DB was designed to store JSON data natively (it technically uses a binary version of JSON called **BSON**), and everything from its command line interface to its query language is built on JSON and JavaScript.

Mongo DB works extremely well with Node.js, and makes storing, manipulating, and representing JSON data at every tier of your application incredibly easy. For cloud-native applications, **Mongo DB Atlas** makes it even easier, by giving you an auto-scaling Mongo DB cluster on the cloud provider of your choice.

Express.js and React.js make the JavaScript/JSON application MERN full stack, well, full. Express.js is a server-side application framework that wraps HTTP requests and responses, and makes it easy to map URLs to server-side functions. React.js is a front-end JavaScript framework for building interactive user interfaces in HTML, and communicating with a remote server.

The combination means that JSON data flows naturally from front to back, making it fast to build on and reasonably simple to debug. Plus, you only have to know one programming language, and the JSON document structure, to understand the whole system.

MERN is the stack of choice for today's web developers looking to move quickly, particularly for those with React.js experience.

Top MERN Stack Development Companies in INDIA-

- 1. Deligence Technologies PVT.LTD.**
- 2. Apps Maven**
- 3. Value Coders**
- 4. Techno Score**
- 5. Tech Scooper**
- 6. Phontinent Technologies**

Business Benefits of Using the MERN Technology Stack

1. Open-Source Technology

Startups prefer MERN because it's an open-source code that's constantly being improved upon by tech experts from around the world. All the components of the MERN stack are open source, and thus you can use it to create robust web apps.

For example, AngularJS is an open-source frontend framework explicitly created to address some of the issues inherent in earlier versions of HTML and JavaScript. Frameworks like these improve coding efficiency and provide countless tools to help you build more advanced apps faster.

2. Free Templates are Available Online

Free templates are available online, which will save you tons of time. It would have taken me at least three times as long to customize a theme compared to downloading an off-the-shelf solution. You'll also get help from experts if you run into any issues along the way.

Many platforms have online communities where you can post questions and get feedback on your code directly from other developers working with that platform. It's always easier to learn when experts are there to hold your hand!

3. Allows You to Build Fast

The MERN stacks rely on open-source technologies that developers can use freely, which means you don't have to build everything from scratch. For example, if you want a blogging platform, chances are someone has already developed a popular framework like WordPress.

All you need to do is pay detailed attention to any setup instructions and customize as needed. This kind of plug-and-play approach gives you an edge against more prominent companies that might not understand (or care about) such emerging

platforms. The bigger they are, after all, the more likely they stick with proven solutions.

4. Easy to Use

The underlying technology gets so well documented that you can deploy it with ease. It's simple to understand and use, making it an ideal choice for beginners learning web development. With so many tools available to developers, it can be challenging to decide which ones are worth learning about. The minimalist nature of these tools makes them easier to learn since there are fewer moving parts you need to understand to use them effectively.

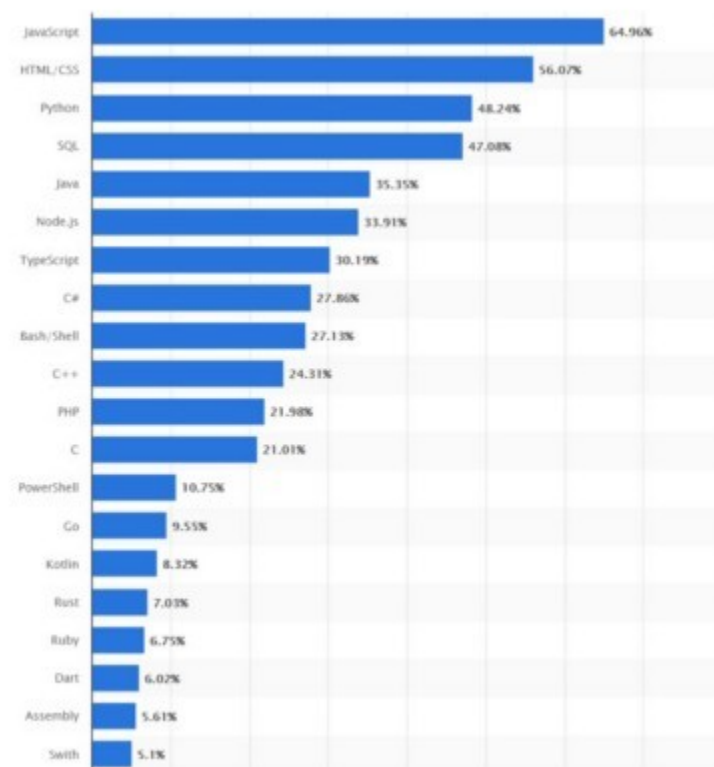
5. Full-Stack Development

MERN's full-stack development approach means you're responsible for building an application's frontend and backend components. This approach draws on principles from both UI design and software engineering.

It is increasingly popular with startups since it can save them money by not having to bring in outside developers separately for the frontend and backend of the project.

6. Great Community Support

This set of technologies is backed by a solid community, making it easier to find answers to all sorts of technical questions that may come up. Thus when you're unsure how something works or why there's usually someone else who has already encountered (and fixed) similar problems.



7. Offers Native Experience to the Users

Native applications are more robust, secure and can provide a more compelling experience for users. So when startups develop mobile apps, they usually build them on top of hybrid frameworks like Ionic or React Native.

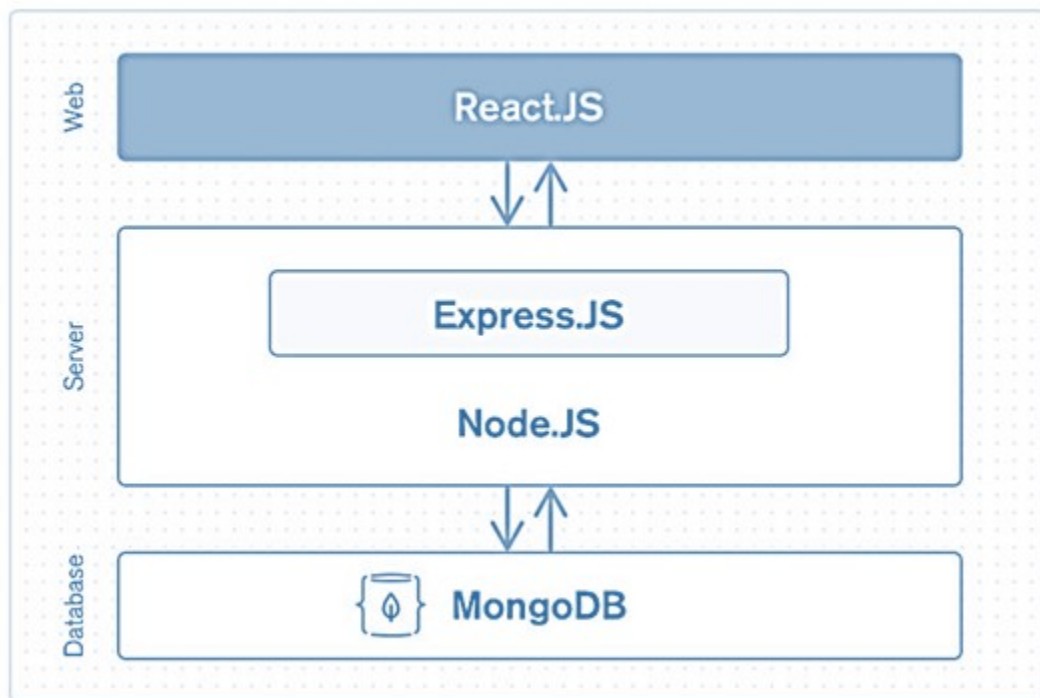
Although these frameworks are remarkable, they run on web technologies that offer little performance or deep integration with devices. A mobile application built on MERN leverages native features like camera access and will be able to seamlessly sync data between offline and online states within an app.

Architectural Structure of MERN Stack

MERN has a 3-tier Architecture system mainly consisting of 3 layers -

These layers are as follows:

1. Web as front-end tier
2. Server as the middle tier
3. Database as backend tier



METHODOLOGY

Software Development Life Cycle:

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.



Different stages of SDLC:

1. Requirement Analysis

Software Development Life Cycle begins with Requirement Analysis phase, where the stakeholders discuss the requirements of the software that needs to be developed to achieve a goal. The aim of the requirement analysis phase is to capture the detail of each requirement and to make sure everyone understands the scope of the work and how each requirement is going to be fulfilled.

2. Design

The next stage of Software Development Life Cycle is the Design phase. During the design phase, developers and technical architects start the high-level design of the software and system to be able to deliver each requirement.

The technical details of the design is discussed with the stakeholders and various parameters such as risks, technologies to be used, capability of the team, project constraints, time and budget are reviewed and then the best design approach is selected for the product.

The selected architectural design defines all the components that needs to be developed, communications with third party services, user flows and database communications as well as front-end representations and behavior of each components. The design is usually kept in the Design Specification Document (DSD).

3. Implementation

After the requirements and design activity is completed, the next phase of the Software Development Life Cycle is the implementation or development of the software. In this phase, developers start coding according to the requirements and the design discussed in previous phases.

Database admins create the necessary data in the database, front-end developers create the necessary interfaces and GUI to interact with the back end all based on guidelines and procedures defined by the company.

Developers also write unit tests for each component to test the new code that they have written, review each other's code, create builds and deploy software to an environment. This cycle of development is repeated until the requirements are met.

4. Testing

Testing is the last phase of the Software Development Life Cycle before the software is delivered to customers. During testing, experienced testers start to test the system against the requirements.

The testers aim to find defects within the system as well as verifying whether the application behaves as expected and according to what was documented in the requirements analysis phase.

Testers can either use a test script to execute each test and verify the results, or use exploratory testing which is more of an experience based approach.

It is possible that defects are identified in the testing phase. Once a defect is found, testers inform the developers about the details of the issue and if it is a valid defect, developers will fix and create a new version of the software which needs to be verified again.

This cycle is repeated until all requirements have been tested and all the defects have been fixed and the software is ready to be shipped.

5. Deployment and Maintenance

Once the software has been fully tested and no high priority issues remain in the software, it is time to deploy to production where customers can use the system.

Once a version of the software is released to production, there is usually a maintenance team that looks after any post-production issues.

Flowchart

Flow chart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing. The process of drawing a flowchart for an algorithm is known as “flowcharting”.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Basic Symbols used in Flowchart Designs

1. **Terminal:** The oval symbol indicates Start, Stop and Halt in a program’s logic flow. A pause/halt is generally used in program logic under some error conditions. Terminal is the first and last symbols in the flowchart.



2. **Input/Output:** A parallelogram denotes any function of input/output type.

Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



3. **Processing:** A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



4. **Decision:** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



5. Connectors & Flow lines: Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle. Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.



The graphics above represent different part of a flowchart. The process in a flowchart can be expressed through boxes and arrows with different sizes and colors. In a flowchart, we can easily highlight a certain element and the relationships between each part

PROJECT OVERVIEW:

Cake shop is an establishment that sells flour and cream based food baked in oven such as Bread, cake, patties etc and provide services for special occasion such as wedding , birthday pasties, anniversaries or even business events cake shop can provide a wide range of assorted cake in different design.

When baked goods like cakes are ready to sell it is essential they earn your customers attention and the product need to look good enough test presenting an attractive dietry is one way to garner that virual appeal from customers and passerby.

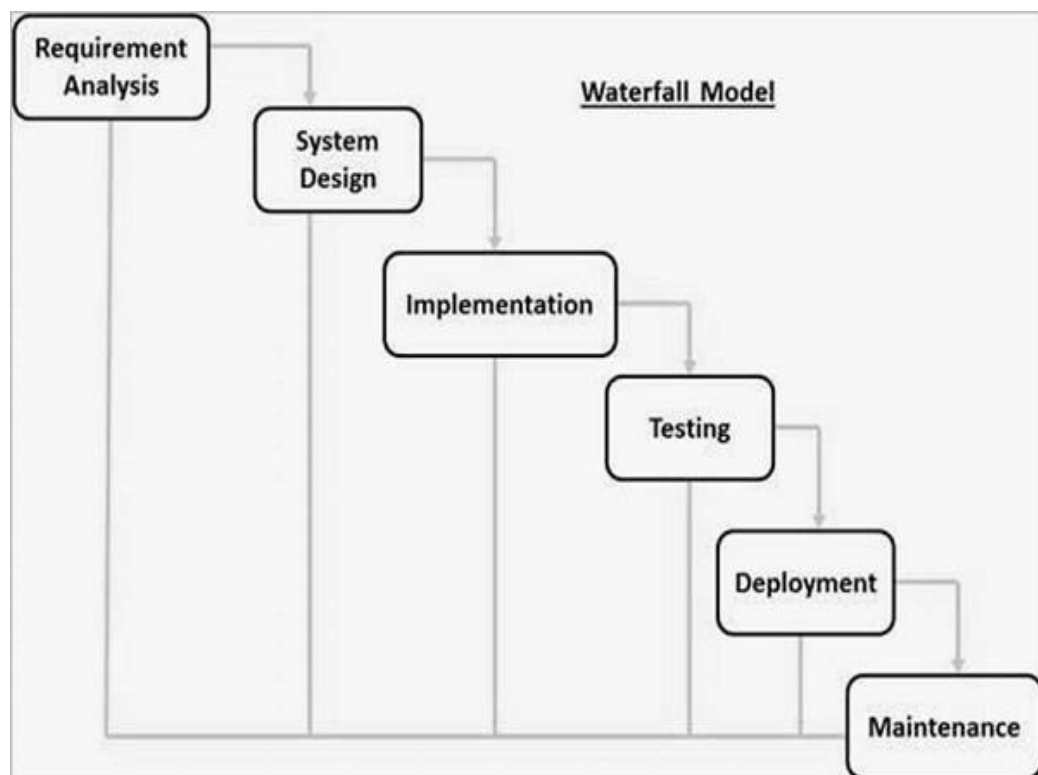
Pastries, Rusks, Bread, Cakes, Toffees etc. comes under the bakery product. Now the advancement of age food habits of the ready-made food. Most of all the bakery items are under the category of ready-made food. Now school going children are like to have the items like parties cakes, breads etc. for

their tiffin use office going peoples like to take breads and parties products, in our daily break fast table also take. The share of bakery products even daily wages laborers also habiterated to spend their lunch by taking bread or like that baking product.

SDLCMODEL TO BEUSED

- **Waterfall Model -Design**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. The following illustration is a representation of the different phases of the Waterfall Model.



These sequential phases in Waterfall model are—

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirements specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Waterfall Model– Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are–

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.

- Ampleresourceswithrequiredexpertiseareavailabletosupporttheproduct.
- Theprojectisshort.

Waterfall Model- Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

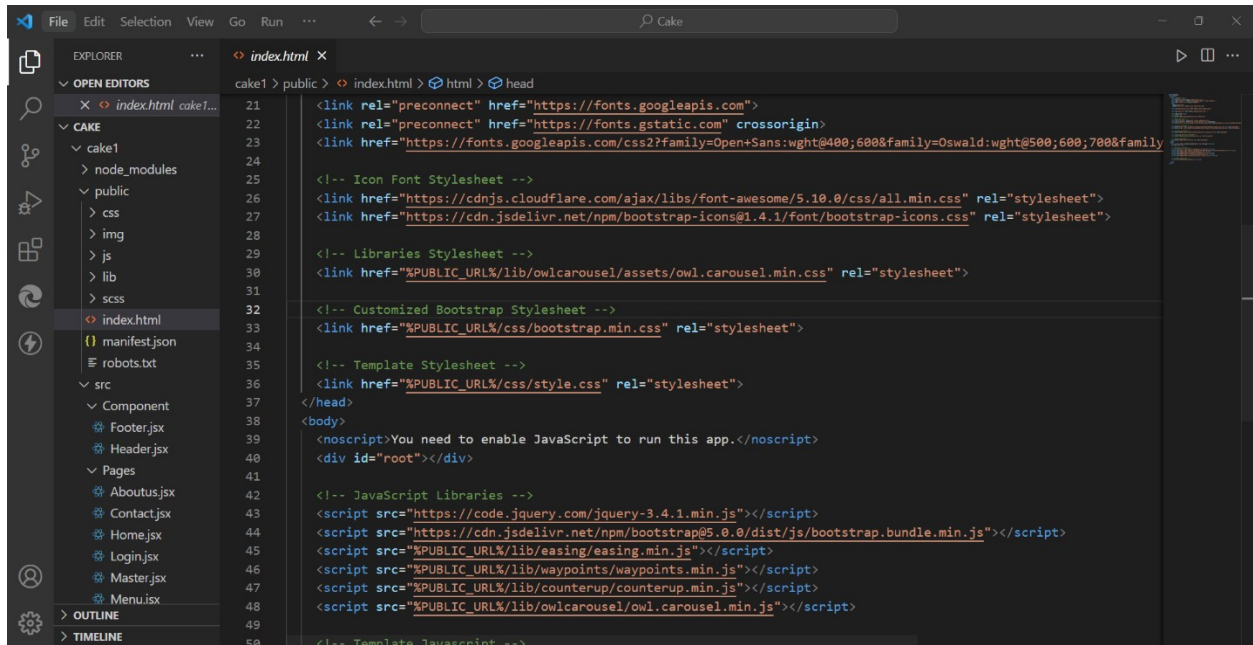
Some of the major advantages of the Waterfall Model are as follows—

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
 - Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.

- Wellunderstoodmilestones.
- Easytoarrangetasks.
- Processandresultsarewelldocumented.

CODING SCREENSHOTS

1.



The screenshot shows the Visual Studio Code interface with the 'index.html' file open in the editor. The Explorer sidebar on the left shows the project structure, including 'cake1' and its subdirectories 'node_modules', 'public', and 'src'. The 'index.html' file is selected in the Explorer. The editor displays the HTML code for 'index.html', which includes links to Google Fonts, Font Awesome, Bootstrap Icons, and various CSS and JavaScript files. The code is as follows:

```
21 <link rel="preconnect" href="https://fonts.googleapis.com">
22 <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
23 <link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600&family=Oswald:wght@500;600;700&family=
24
25 <!-- Icon Font Stylesheet -->
26 <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">
27 <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.1/font/bootstrap-icons.css" rel="stylesheet">
28
29 <!-- Libraries Stylesheet -->
30 <link href="%PUBLIC_URL%/lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">
31
32 <!-- Customized Bootstrap Stylesheet -->
33 <link href="%PUBLIC_URL%/css/bootstrap.min.css" rel="stylesheet">
34
35 <!-- Template Stylesheet -->
36 <link href="%PUBLIC_URL%/css/style.css" rel="stylesheet">
37 </head>
38 <body>
39 <noscript>You need to enable JavaScript to run this app.</noscript>
40 <div id="root"></div>
41
42 <!-- JavaScript Libraries -->
43 <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
44 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.bundle.min.js"></script>
45 <script src="%PUBLIC_URL%/lib/easing/easing.min.js"></script>
46 <script src="%PUBLIC_URL%/lib/waypoints/waypoints.min.js"></script>
47 <script src="%PUBLIC_URL%/lib/counterup/counterup.min.js"></script>
48 <script src="%PUBLIC_URL%/lib/owlcarousel/owl.carousel.min.js"></script>
49
50 <!-- Template Javascript -->
```

2.

```
41 <a href="index.html" class="navbar-brand d-block d-lg-none">
42 <h1 class="m-0 text-uppercase text-white"><i class="fa fa-birthday-cake fs-1 text-primary me-3"></i>CakeZ
43 </a>
44 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse">
45 <span class="navbar-toggler-icon"></span>
46 </button>
47 <div class="collapse navbar-collapse" id="navbarCollapse">
48 <div class="navbar-nav ms-auto mx-lg-auto py-0">
49 <Link to="/" class="nav-item nav-link active">Home</Link>
50 <Link to="/Aboutus/" class="nav-item nav-link">About Us</Link>
51 <Link to="/Menu/" class="nav-item nav-link">Menu & Pricing</Link>
52 <Link to="/Master/" class="nav-item nav-link">Master Chefs</Link>
53 <div class="nav-item dropdown">
54 <a href="#" class="nav-link dropdown-toggle" data-bs-toggle="dropdown">Pages</a>
55 <div class="dropdown-menu m-0">
56 <Link to="/Service/" class="dropdown-item">Our Service</Link>
57 <Link to="/Testimonial/" class="dropdown-item">Testimonial</Link>
58 </div>
59 </div>
60 <Link to="/Contact/" class="nav-item nav-link">Contact Us</Link>
61 <Link to="/Login/" class="nav-item nav-link">Login</Link>
62 <Link to="/Register/" class="nav-item nav-link">Contact Us</Link>
63 </div>
64 </div>
65 </nav>
66 <!-- Navbar End -->
67 </div>
68 }
69 }
70
```

3.

```
52 <h4 class="text-primary text-uppercase mb-4">Newsletter</h4>
53 <p>Amet justo diam dolor rebum lorem sit stet sea justo kasd</p>
54 <form action="">
55 <div class="input-group">
56 <input type="text" class="form-control border-white p-3" placeholder="Your Email"/>
57 <button class="btn btn-primary">Sign Up</button>
58 </div>
59 </form>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 <div class="container-fluid text-secondary py-4" style={{background: "#111111"}}>
67 <div class="container text-center">
68 <p class="mb-0">&copy; <a class="text-white border-bottom" href="#">Your Site Name</a>. All Rights Reserve
69 <a class="text-white border-bottom" href="https://htmlcodex.com">HTML Codex</a></p>
70 <br/>Distributed By: <a class="border-bottom" href="https://themewagon.com" target="_blank">ThemeWagon</a>
71 </div>
72 </div>
73 </div>
74 <!-- Footer End -->
75 </div>
76 }
77 }
78
```

4.

```
1 import React from 'react'
2
3 export default function Aboutus() {
4   return (
5     <div>
6       { /* <!-- Page Header Start --> */ }
7       <div class="container-fluid bg-dark bg-img p-5 mb-5">
8         <div class="row">
9           <div class="col-12 text-center">
10             <h1 class="display-4 text-uppercase text-white">About Us</h1>
11             <a href="#">Home</a>
12             <i class="far fa-square text-primary px-2"></i>
13             <a href="#">About</a>
14           </div>
15         </div>
16       </div>
17       { /* <!-- Page Header End --> */ }
18
19       { /* <!-- About Start --> */ }
20       <div class="container-fluid pt-5">
21         <div class="container">
22           <div class="section-title position-relative text-center mx-auto mb-5 pb-3" style={{maxWidth: "600px"}}>
23             <h2 class="text-primary font-secondary">About Us</h2>
24             <h1 class="display-4 text-uppercase">Welcome To CakeZone</h1>
25           </div>
26           <div class="row gx-5">
27             <div class="col-lg-5 mb-5 mb-lg-0" style={{minHeight: "400px"}}>
28               <div class="position-relative h-100">
29                 
6       { /* <!-- Page Header Start --> */ }
7       <div class="container-fluid bg-dark bg-img p-5 mb-5">
8         <div class="row">
9           <div class="col-12 text-center">
10             <h1 class="display-4 text-uppercase text-white">Contact Us</h1>
11             <a href="#">Home</a>
12             <i class="far fa-square text-primary px-2"></i>
13             <a href="#">Contact</a>
14           </div>
15         </div>
16       </div>
17       { /* <!-- Page Header End --> */ }
18
19       { /* <!-- Contact Start --> */ }
20       <div class="container-fluid contact position-relative px-5" style={{marginTop: "90px"}}>
21         <div class="container">
22           <div class="row g-5 mb-5">
23             <div class="col-lg-4 col-md-6">
24               <div class="bg-primary border-inner text-center text-white p-5">
25                 <i class="bi bi-geo-alt fs-1 text-white"></i>
26                 <h6 class="text-uppercase my-2">Our Office</h6>
27                 <span>123 Street, New York, USA</span>
28               </div>
29             </div>
30           </div>
31         </div>
32       </div>
33     </div>
34   )
35 }
```

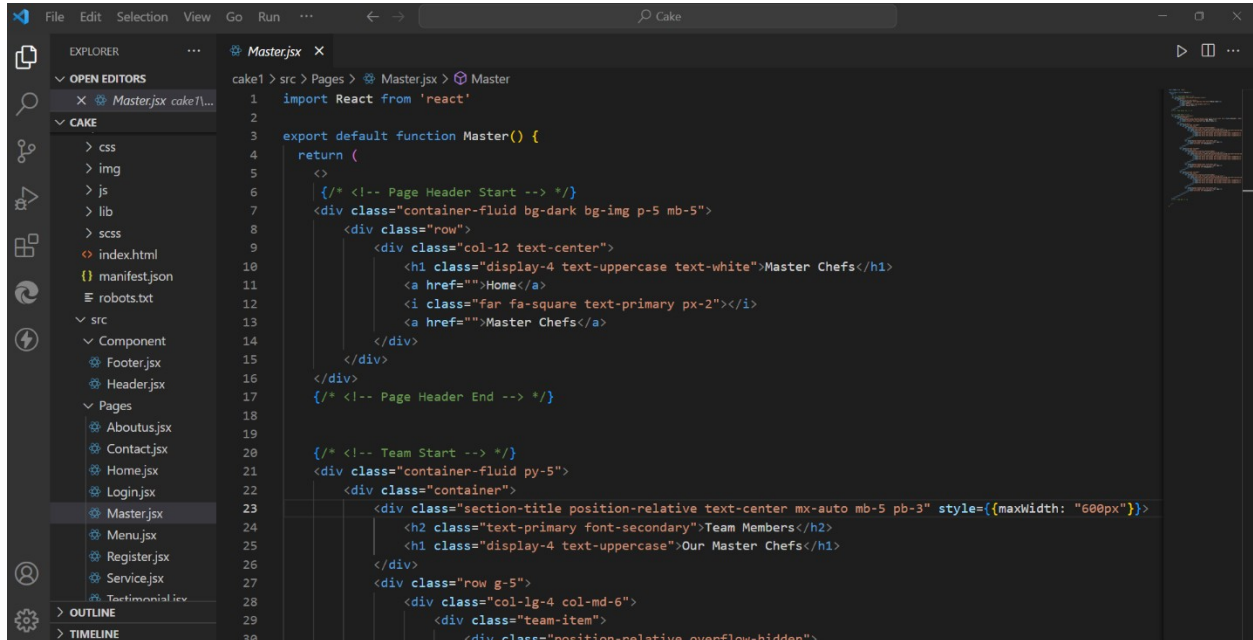
6.


```
543 <div class="ps-3">
544   <h4 class="text-primary text-uppercase mb-1">Client Name</h4>
545   <span>Profession</span>
546 </div>
547 </div>
548 <p class="mb-0">Dolor et eos labore, stet justo sed est sed. Diam sed sed dolor stet amet eirmod eos labore diam
549 </p>
550 <div>
551   <div class="testimonial-item bg-dark text-white border-inner p-4">
552     <div class="d-flex align-items-center mb-3">
553       
554       <div class="ps-3">
555         <h4 class="text-primary text-uppercase mb-1">Client Name</h4>
556         <span>Profession</span>
557       </div>
558     </div>
559     <p class="mb-0">Dolor et eos labore, stet justo sed est sed. Diam sed sed dolor stet amet eirmod eos labore diam
560     </p>
561   </div>
562   <div class="testimonial-item bg-dark text-white border-inner p-4">
563     <div class="d-flex align-items-center mb-3">
564       
565       <div class="ps-3">
566         <h4 class="text-primary text-uppercase mb-1">Client Name</h4>
567         <span>Profession</span>
568       </div>
569     </div>
570     <p class="mb-0">Dolor et eos labore, stet justo sed est sed. Diam sed sed dolor stet amet eirmod eos labore diam
571     </p>
572   </div>
```

7.

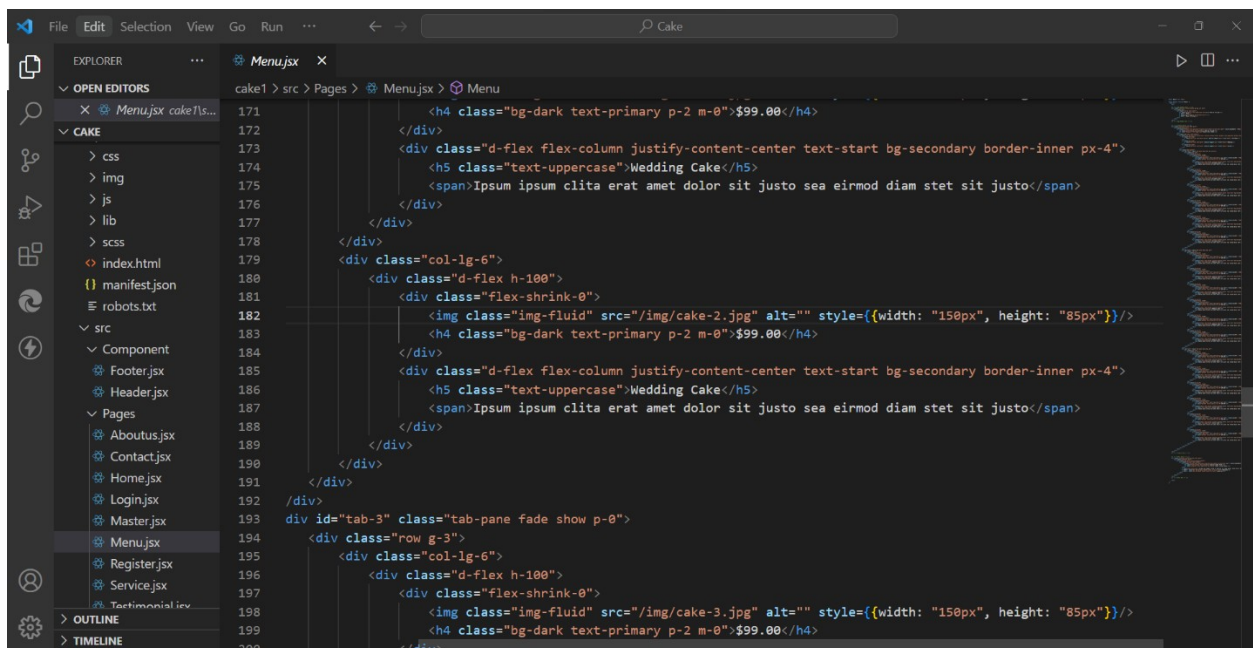
```
47 <div className="container-xxl py-5">
48   <div className="container">
49     <div className="text-center mx-auto wow fadeInUp" data-wow-delay="0.1s" style={{maxWidth: "500px;}}>
50       <p className="fs-5 fw-medium text-primary">Contact Us</p>
51       <h1 className="display-5 mb-5">If You Have Any Query, Please Contact Us</h1>
52     </div>
53     <div className="row g-5">
54       <div className="col-lg-6 wow fadeInUp" data-wow-delay="0.1s">
55         <h3 className="mb-4">Need a functional contact form?</h3>
56         <p className="mb-4">The contact form is currently inactive. Get a functional and working contact
57         <form onSubmit={onHandleForm} method="post">
58           <div className="row g-3">
59             <div className="col-md-6">
60               <div className="form-floating">
61                 <input type="text" className="form-control" id="name" placeholder="Your Name"
62                 name="username"
63                 value={formData.username}
64                 onChange={onChangeData}/>
65                 <label for="name">Your Name</label>
66               </div>
67             </div>
68             <div className="col-md-6">
69               <div className="form-floating">
70                 <input type="email" className="form-control" id="email" placeholder="Your Email"
71                 name="email"
72                 value={formData.email}
73                 onChange={onChangeData}/>
74                 <label for="email">Your Email</label>
75               </div>
76             </div>
```

8.



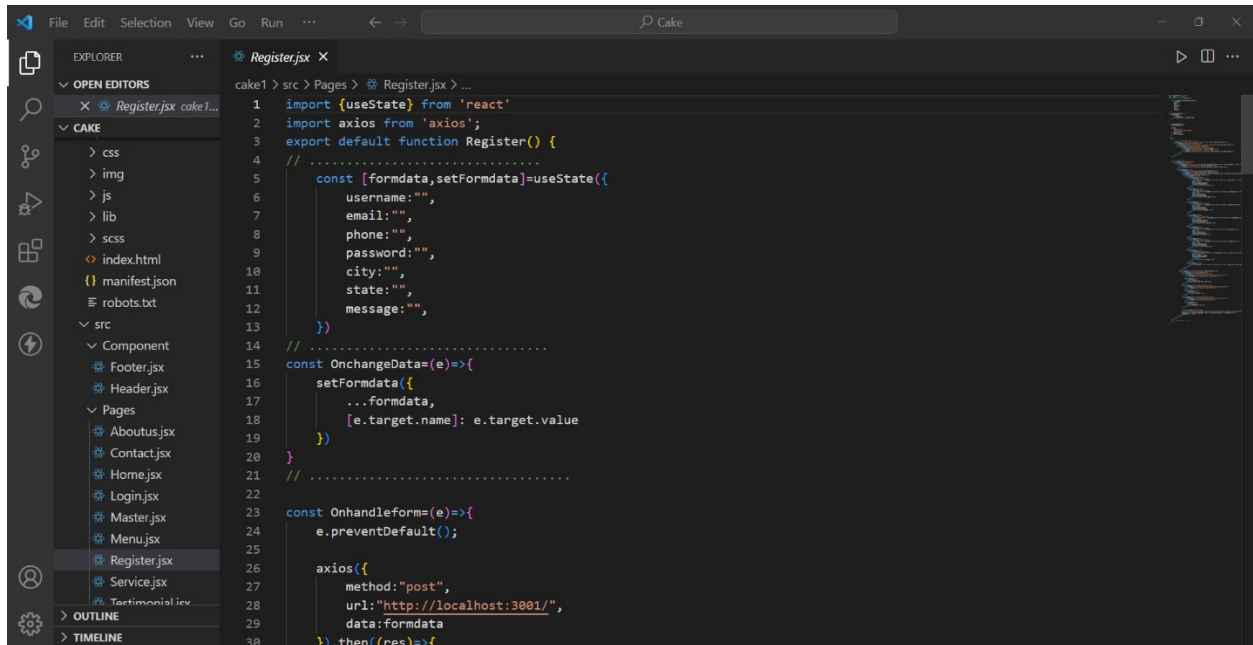
```
1 import React from 'react'
2
3 export default function Master() {
4   return (
5     <>
6     <!-- Page Header Start --> </>
7     <div class="container-fluid bg-dark bg-img p-5 mb-5">
8       <div class="row">
9         <div class="col-12 text-center">
10           <h1 class="display-4 text-uppercase text-white">Master Chefs</h1>
11           <a href="#">Home</a>
12           <i class="far fa-square text-primary px-2"></i>
13           <a href="#">Master Chefs</a>
14         </div>
15       </div>
16     </div>
17     <!-- Page Header End --> </>
18
19     <!-- Team Start --> </>
20     <div class="container-fluid py-5">
21       <div class="container">
22         <div class="section-title position-relative text-center mx-auto mb-5 pb-3" style={{maxWidth: "600px"}}>
23           <h2 class="text-primary font-secondary">Team Members</h2>
24           <h1 class="display-4 text-uppercase">Our Master Chefs</h1>
25         </div>
26         <div class="row g-5">
27           <div class="col-lg-4 col-md-6">
28             <div class="team-item">
29               <div class="position-relative overflow-hidden">
```

9.



```
171 <h4 class="bg-dark text-primary p-2 m-0">$99.00</h4>
172 </div>
173 <div class="d-flex flex-column justify-content-center text-start bg-secondary border-inner px-4">
174   <h5 class="text-uppercase">Wedding Cake</h5>
175   <span>Ipsum ipsum clita erat amet dolor sit justo sea eirmod diam stet sit justo</span>
176 </div>
177 </div>
178 </div>
179 <div class="col-lg-6">
180   <div class="d-flex h-100">
181     <div class="flex-shrink-0">
182       
183       <h4 class="bg-dark text-primary p-2 m-0">$99.00</h4>
184     </div>
185     <div class="d-flex flex-column justify-content-center text-start bg-secondary border-inner px-4">
186       <h5 class="text-uppercase">Wedding Cake</h5>
187       <span>Ipsum ipsum clita erat amet dolor sit justo sea eirmod diam stet sit justo</span>
188     </div>
189   </div>
190 </div>
191 </div>
192 </div>
193 <div id="tab-3" class="tab-pane fade show p-0">
194   <div class="row g-3">
195     <div class="col-lg-6">
196       <div class="d-flex h-100">
197         <div class="flex-shrink-0">
198           
199           <h4 class="bg-dark text-primary p-2 m-0">$99.00</h4>
200         </div>
```

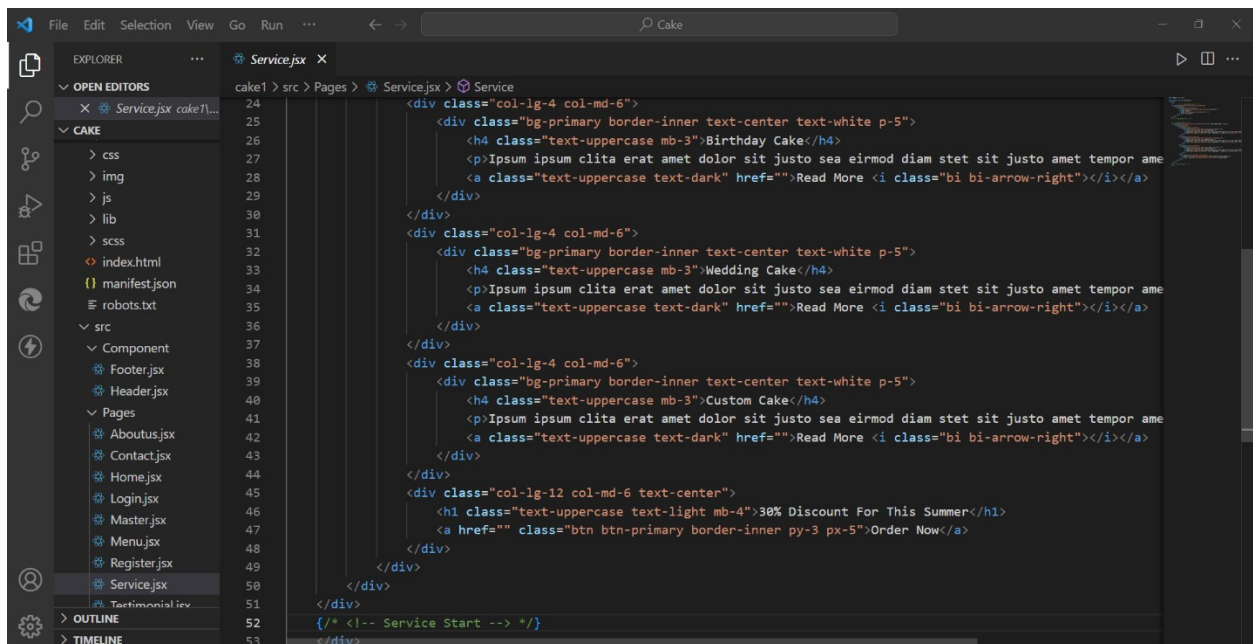
10.



The screenshot shows the Visual Studio Code editor with the 'Register.jsx' file open. The Explorer sidebar on the left shows the project structure for 'cake1', including folders for 'css', 'img', 'js', 'lib', 'scss', and 'src'. The 'src' folder is expanded, showing subfolders 'Component' and 'Pages'. The 'Pages' folder contains several files, with 'Register.jsx' selected. The main editor area displays the code for 'Register.jsx', which includes imports for 'useState' and 'axios', a default export function 'Register()', and logic for handling form data and sending a POST request.

```
1 import {useState} from 'react'
2 import axios from 'axios';
3 export default function Register() {
4   // .....
5   const [formdata,setFormdata]=useState({
6     username:"",
7     email:"",
8     phone:"",
9     password:"",
10    city:"",
11    state:"",
12    message:"",
13  })
14  // .....
15  const OnchangeData=(e)=>{
16    setFormdata({
17      ...formdata,
18      [e.target.name]: e.target.value
19    })
20  }
21  // .....
22
23  const Onhandleform=(e)=>{
24    e.preventDefault();
25
26    axios({
27      method:"post",
28      url:"http://localhost:3001/",
29      data:formdata
30    }).then((res)=>{
```

11.



The screenshot shows the Visual Studio Code editor with the 'Service.jsx' file open. The Explorer sidebar on the left shows the project structure for 'cake1', with the 'src' folder expanded and 'Pages' subfolder selected. The 'Service.jsx' file is highlighted. The main editor area displays the code for 'Service.jsx', which is a React component using Bootstrap classes to render a list of cake items (Birthday Cake, Wedding Cake, Custom Cake) and a discount announcement.

```
24 <div class="col-lg-4 col-md-6">
25   <div class="bg-primary border-inner text-center text-white p-5">
26     <h4 class="text-uppercase mb-3">Birthday Cake</h4>
27     <p>Ipsum ipsum clita erat amet dolor sit justo sea eirmod diam stet sit justo amet tempor ame
28     <a class="text-uppercase text-dark" href="">Read More <i class="bi bi-arrow-right"></i></a>
29   </div>
30 </div>
31 <div class="col-lg-4 col-md-6">
32   <div class="bg-primary border-inner text-center text-white p-5">
33     <h4 class="text-uppercase mb-3">Wedding Cake</h4>
34     <p>Ipsum ipsum clita erat amet dolor sit justo sea eirmod diam stet sit justo amet tempor ame
35     <a class="text-uppercase text-dark" href="">Read More <i class="bi bi-arrow-right"></i></a>
36   </div>
37 </div>
38 <div class="col-lg-4 col-md-6">
39   <div class="bg-primary border-inner text-center text-white p-5">
40     <h4 class="text-uppercase mb-3">Custom Cake</h4>
41     <p>Ipsum ipsum clita erat amet dolor sit justo sea eirmod diam stet sit justo amet tempor ame
42     <a class="text-uppercase text-dark" href="">Read More <i class="bi bi-arrow-right"></i></a>
43   </div>
44 </div>
45 <div class="col-lg-12 col-md-6 text-center">
46   <h1 class="text-uppercase text-light mb-4">30% Discount For This Summer</h1>
47   <a href="" class="btn btn-primary border-inner py-3 px-5">Order Now</a>
48 </div>
49 </div>
50 </div>
51 </div>
52 { /* <!-- Service Start --> */ }
53 </div>
```

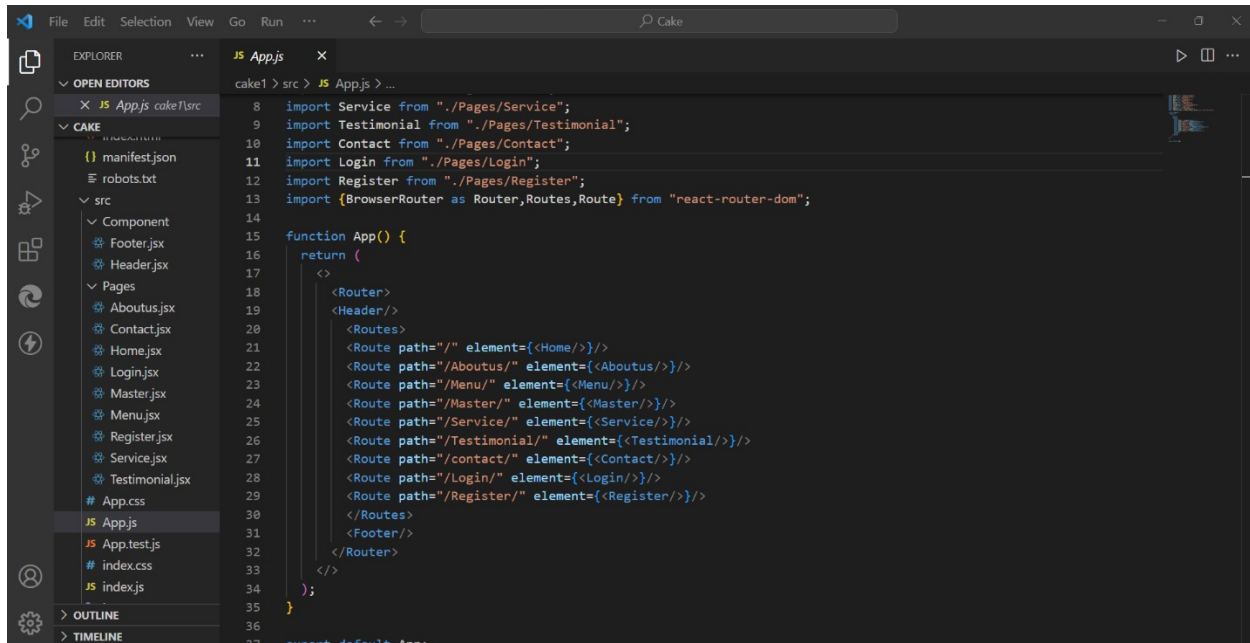
The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the project structure, including folders like 'src' and 'pages', and files like 'App.css'. The main editor area shows the content of 'App.css', which defines styles for various components of a mobile application. The code includes a root selector, a logo container, a header, and a link, all styled for a mobile screen with a width of 400px.

```

1  .App {
2    text-align: center;
3  }
4
5  .App-logo {
6    height: 40vmin;
7    pointer-events: none;
8  }
9
10 @media (prefers-reduced-motion: no-preference) {
11   .App-logo {
12     animation: App-logo-spin infinite 20s linear;
13   }
14 }
15
16 .App-header {
17   background-color: #282c34;
18   min-height: 100vh;
19   display: flex;
20   flex-direction: column;
21   align-items: center;
22   justify-content: center;
23   font-size: calc(10px + 2vmin);
24   color: white;
25 }
26
27 .App-link {
28   color: #61dafb;
29 }
30

```

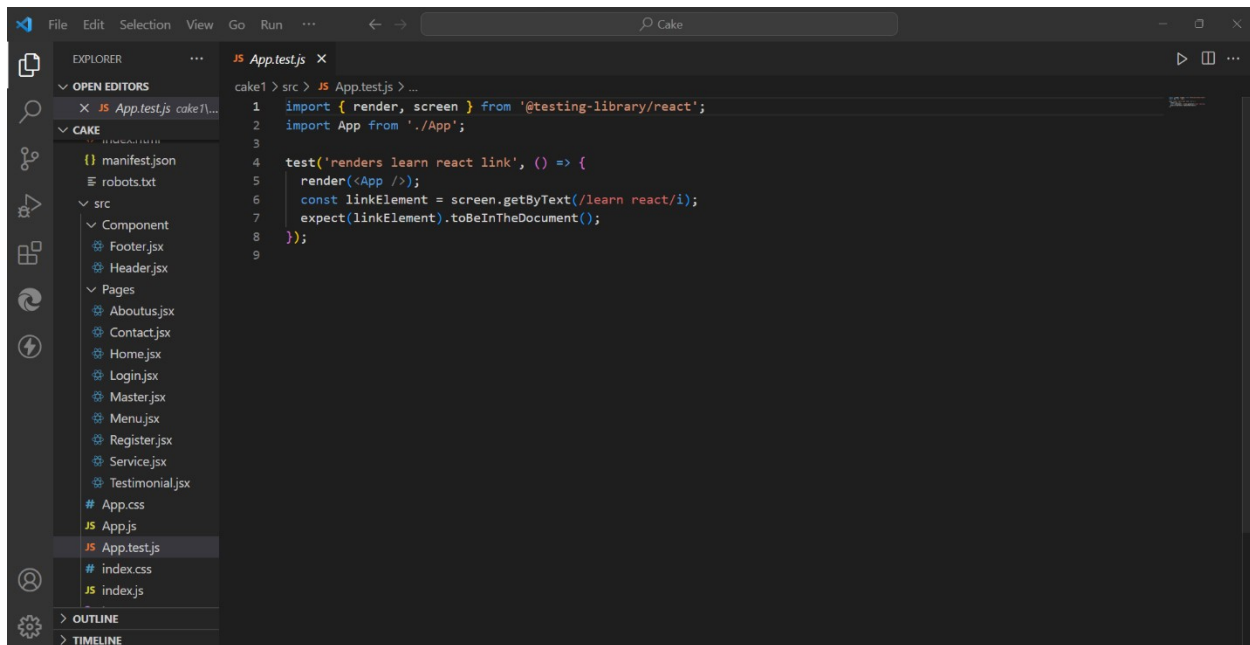

14.



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with a 'src' directory containing 'Component', 'Pages', and 'App.css'. The 'App.js' file is selected in the editor. The code in 'App.js' is as follows:

```
8 import Service from "../Pages/Service";
9 import Testimonial from "../Pages/Testimonial";
10 import Contact from "../Pages/Contact";
11 import Login from "../Pages/Login";
12 import Register from "../Pages/Register";
13 import {BrowserRouter as Router,Routes,Route} from "react-router-dom";
14
15 function App() {
16   return (
17     <>
18       <Router>
19         <Header/>
20         <Routes>
21           <Route path="/" element={<Home/>}/>
22           <Route path="/Aboutus/" element={<Aboutus/>}/>
23           <Route path="/Menu/" element={<Menu/>}/>
24           <Route path="/Master/" element={<Master/>}/>
25           <Route path="/Service/" element={<Service/>}/>
26           <Route path="/Testimonial/" element={<Testimonial/>}/>
27           <Route path="/contact/" element={<Contact/>}/>
28           <Route path="/Login/" element={<Login/>}/>
29           <Route path="/Register/" element={<Register/>}/>
30         </Routes>
31       </Router>
32     </>
33   );
34 }
35
36 export default App;
```

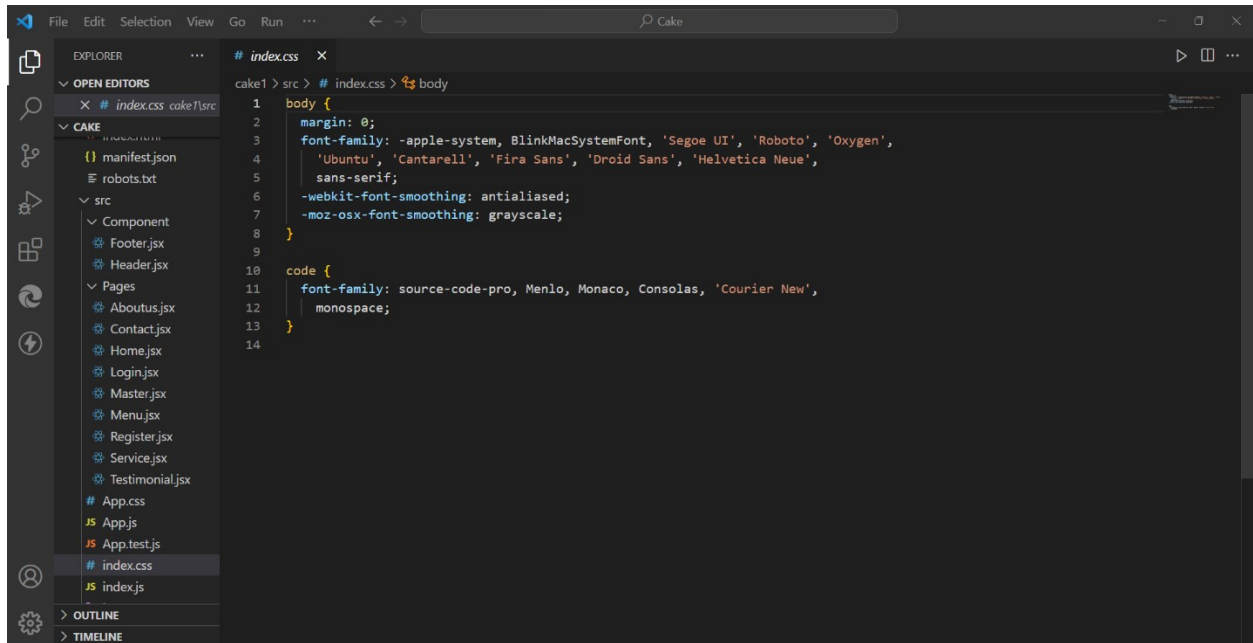
15.



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with a 'src' directory containing 'Component', 'Pages', and 'App.css'. The 'App.test.js' file is selected in the editor. The code in 'App.test.js' is as follows:

```
1 import { render, screen } from '@testing-library/react';
2 import App from './App';
3
4 test('renders learn react link', () => {
5   render(<App />);
6   const linkElement = screen.getByText(/learn react/i);
7   expect(linkElement).toBeInTheDocument();
8 });
9
```

16.



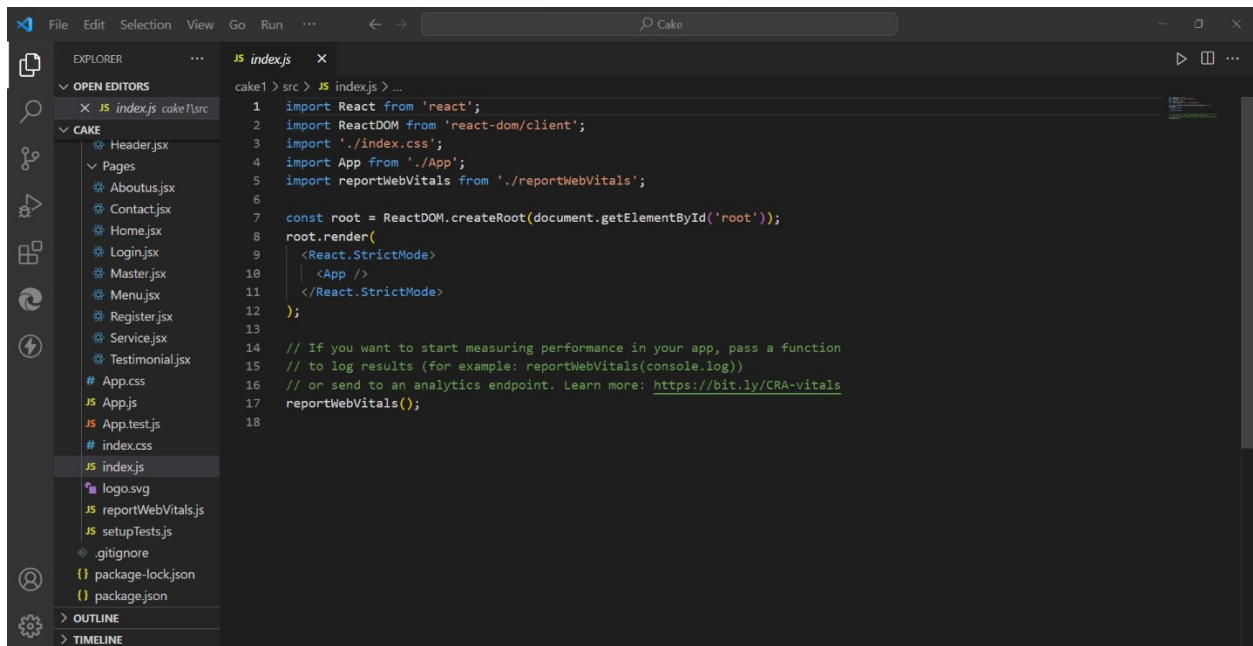
The screenshot shows the Visual Studio Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure with the following files and folders:

- cake1 > src > # index.css
- cake1 > src > # index.css > body
- cake1 > src > # index.css > code

The editor window displays the content of the `index.css` file:

```
1 body {  
2   margin: 0;  
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',  
4     'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',  
5     sans-serif;  
6   -webkit-font-smoothing: antialiased;  
7   -moz-osx-font-smoothing: grayscale;  
8 }  
9  
10 code {  
11   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',  
12   monospace;  
13 }  
14
```

17.



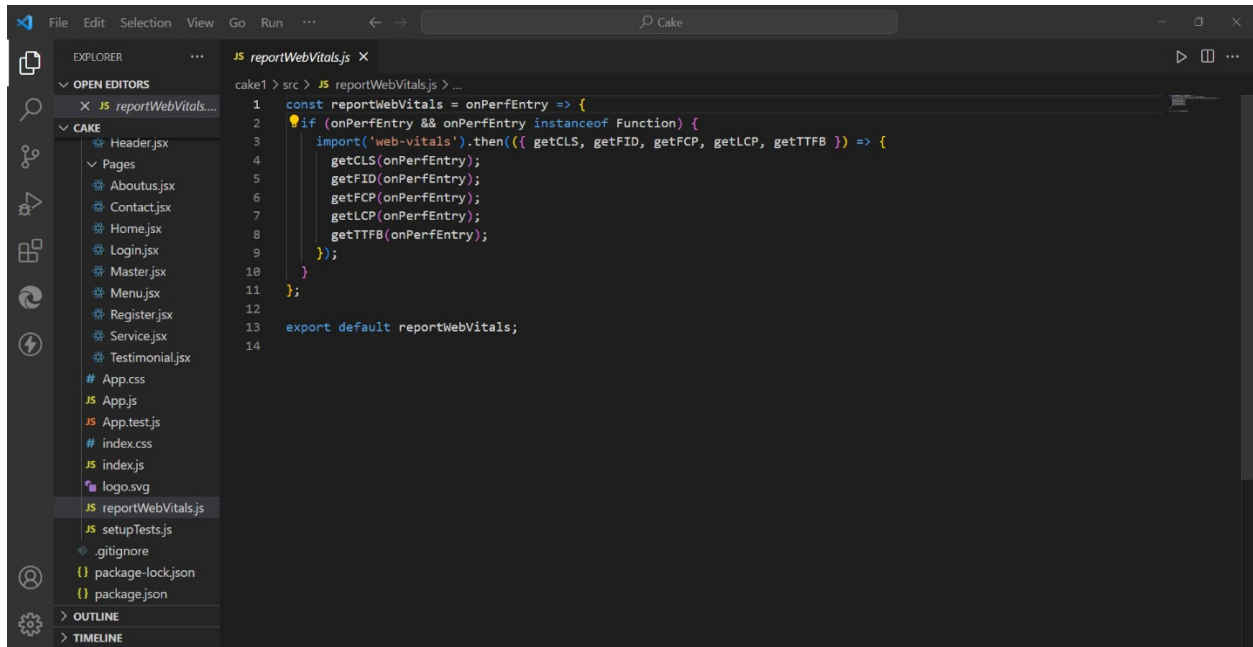
The screenshot shows the Visual Studio Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure with the following files and folders:

- cake1 > src > # index.js
- cake1 > src > # index.js > ...

The editor window displays the content of the `index.js` file:

```
1 import React from 'react';  
2 import ReactDOM from 'react-dom/client';  
3 import './index.css';  
4 import App from './App';  
5 import reportWebVitals from './reportWebVitals';  
6  
7 const root = ReactDOM.createRoot(document.getElementById('root'));  
8 root.render(  
9   <React.StrictMode>  
10     <App />  
11   </React.StrictMode>  
12 );  
13  
14 // If you want to start measuring performance in your app, pass a function  
15 // to log results (for example: reportWebVitals(console.log))  
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals  
17 reportWebVitals();  
18
```

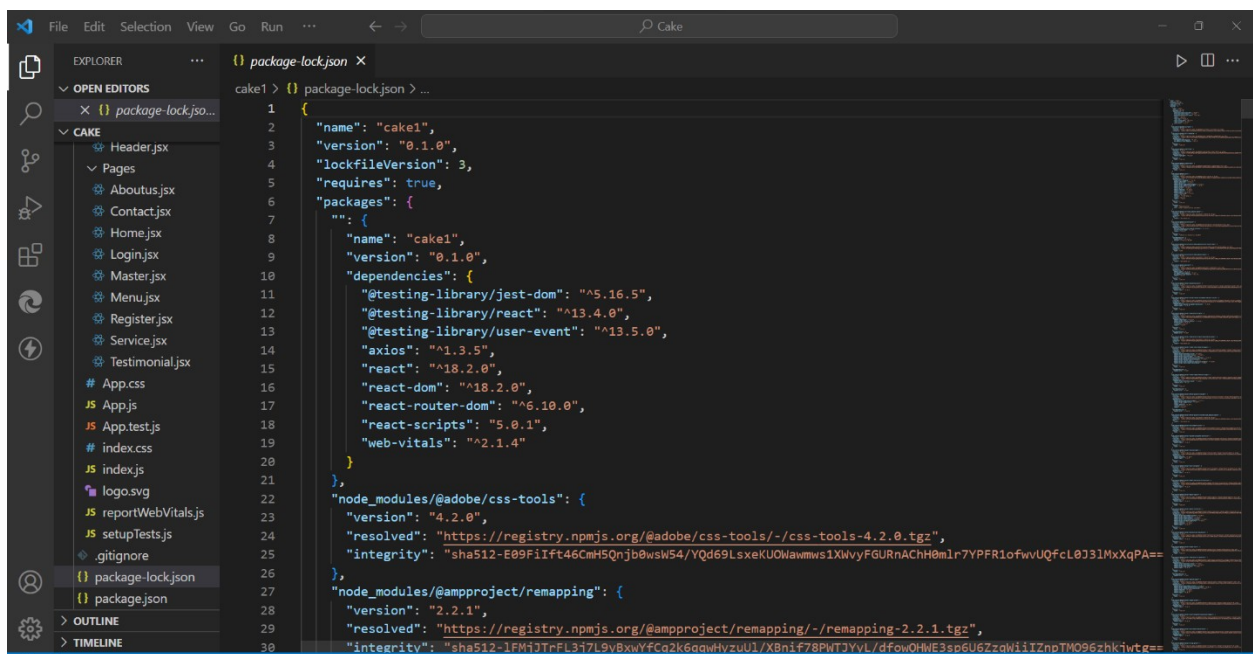
18.



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with folders like 'Pages' and 'CAKE'. The file 'reportWebVitals.js' is selected. The editor displays the following code:

```
1 const reportWebVitals = onPerfEntry => {
2   if (onPerfEntry && onPerfEntry instanceof Function) {
3     import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
4       getCLS(onPerfEntry);
5       getFID(onPerfEntry);
6       getFCP(onPerfEntry);
7       getLCP(onPerfEntry);
8       getTTFB(onPerfEntry);
9     });
10  }
11 };
12
13 export default reportWebVitals;
```

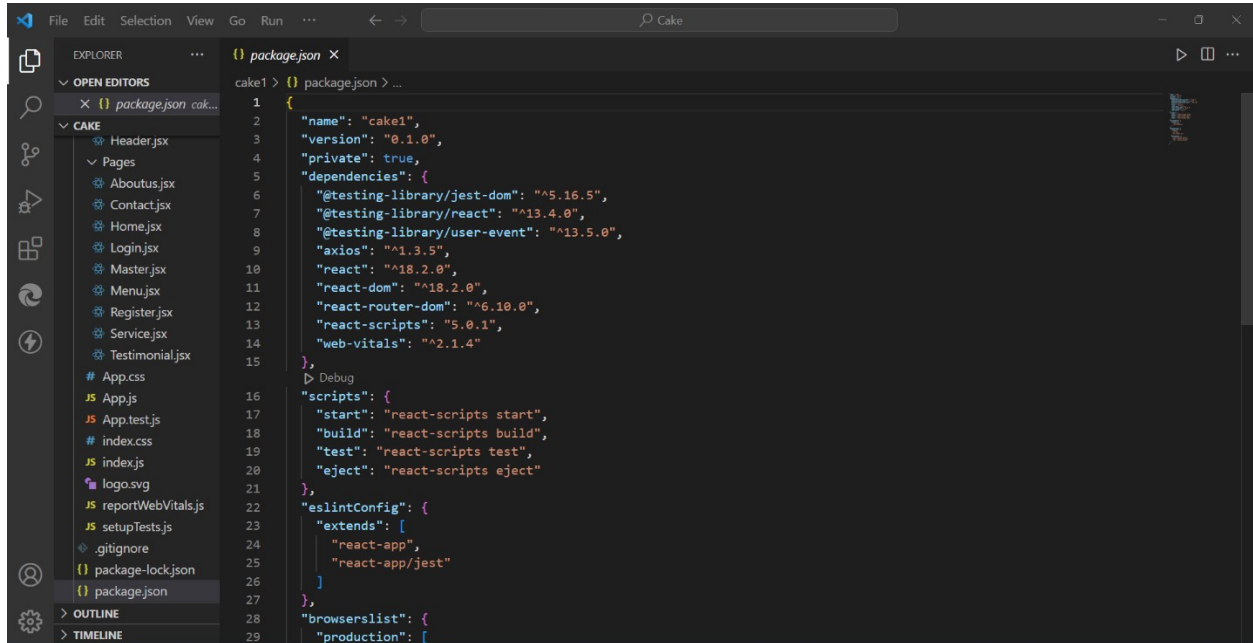
19.



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with folders like 'Pages' and 'CAKE'. The file 'package-lock.json' is selected. The editor displays the following code:

```
1 {
2   "name": "cake1",
3   "version": "0.1.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "cake1",
9       "version": "0.1.0",
10      "dependencies": {
11        "@testing-library/jest-dom": "^5.16.5",
12        "@testing-library/react": "^13.4.0",
13        "@testing-library/user-event": "^13.5.0",
14        "axios": "^1.3.5",
15        "react": "^18.2.0",
16        "react-dom": "^18.2.0",
17        "react-router-dom": "^6.10.0",
18        "react-scripts": "5.0.1",
19        "web-vitals": "^2.1.4"
20      },
21      "devDependencies": {
22        "node_modules/@adobe/css-tools": {
23          "version": "4.2.0",
24          "resolved": "https://registry.npmjs.org/@adobe/css-tools/-/css-tools-4.2.0.tgz",
25          "integrity": "sha512-E89FiIft46CmH5Qnjb0wsW54/YQd69LsxKUDWawmms1XWvyFGURnAchH0m1r7YpFR1ofwUQfcl0J31MxXqPA==",
26          "dev": true
27        },
28        "node_modules/@ampproject/remapping": {
29          "version": "2.2.1",
30          "resolved": "https://registry.npmjs.org/@ampproject/remapping/-/remapping-2.2.1.tgz",
31          "integrity": "sha512-kRnG4c58gE101KZqtk7A1yTdN52g88QAH1oL3K6dsxpX9S03oYy3UMW5lNKdNpS7B58g30GZPYFb7hg==",
32          "dev": true
33        }
34      }
35    }
36  }
```

20.

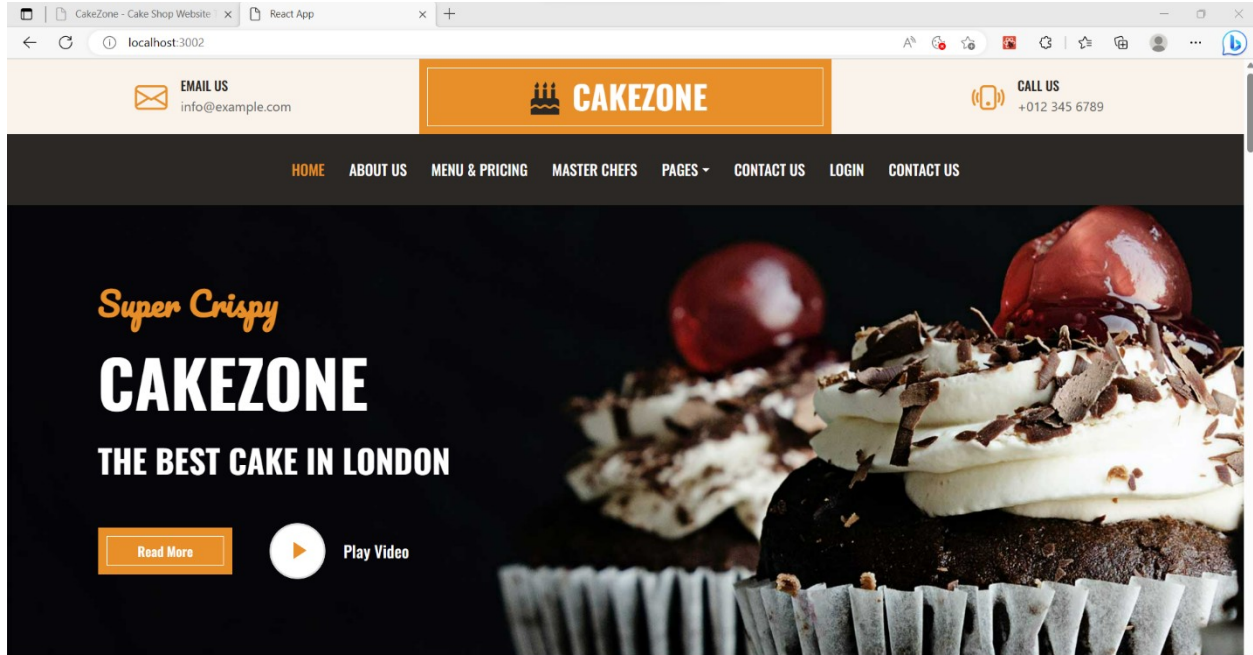


The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'cake1' with a 'package.json' file selected. The code editor displays the contents of 'package.json' for 'cake1'. The file is a JSON object with the following structure:

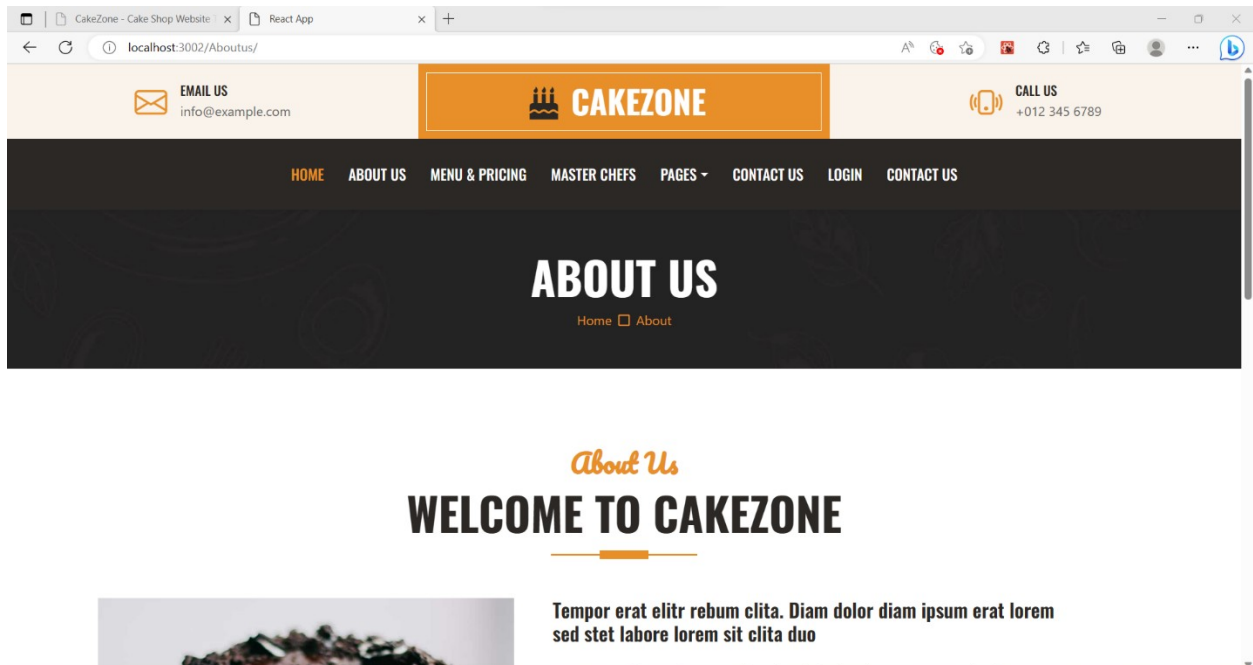
```
{
  "name": "cake1",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.3.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.10.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [

```

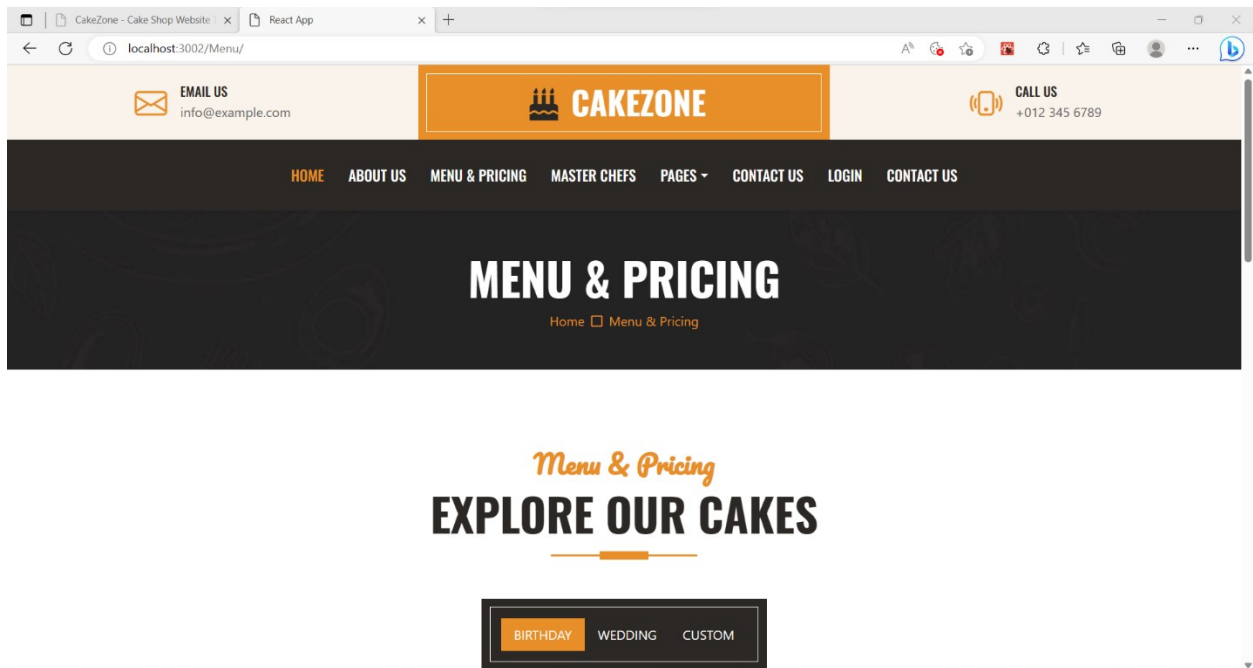
21.



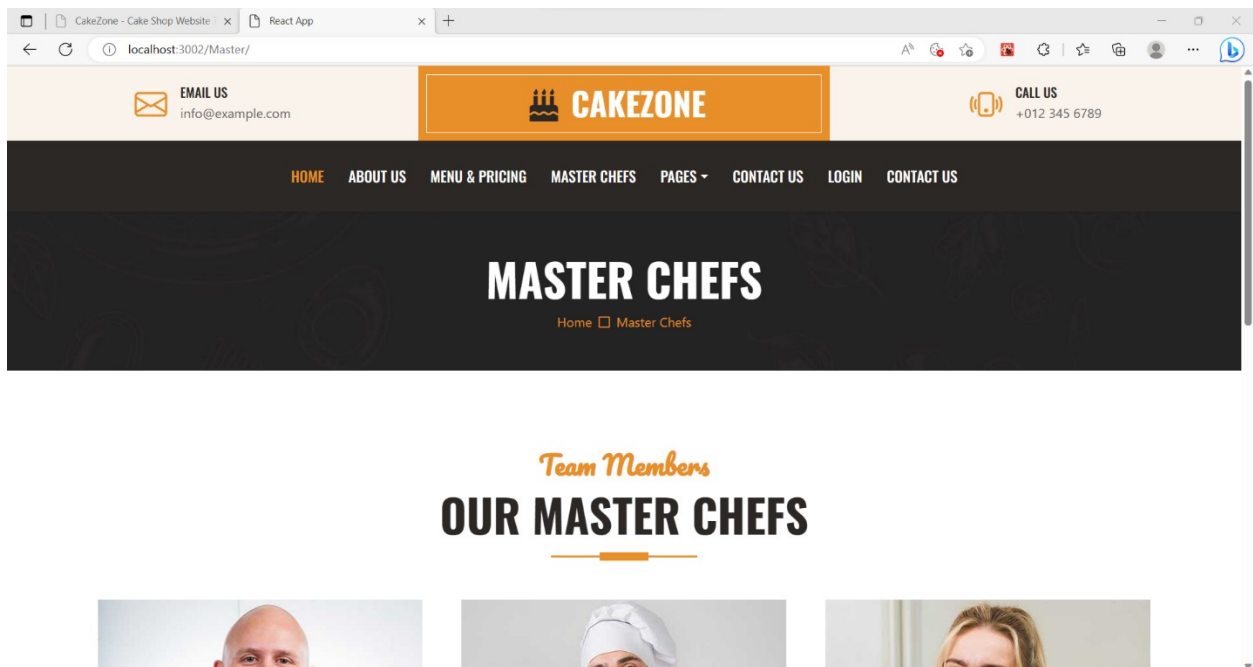
22.



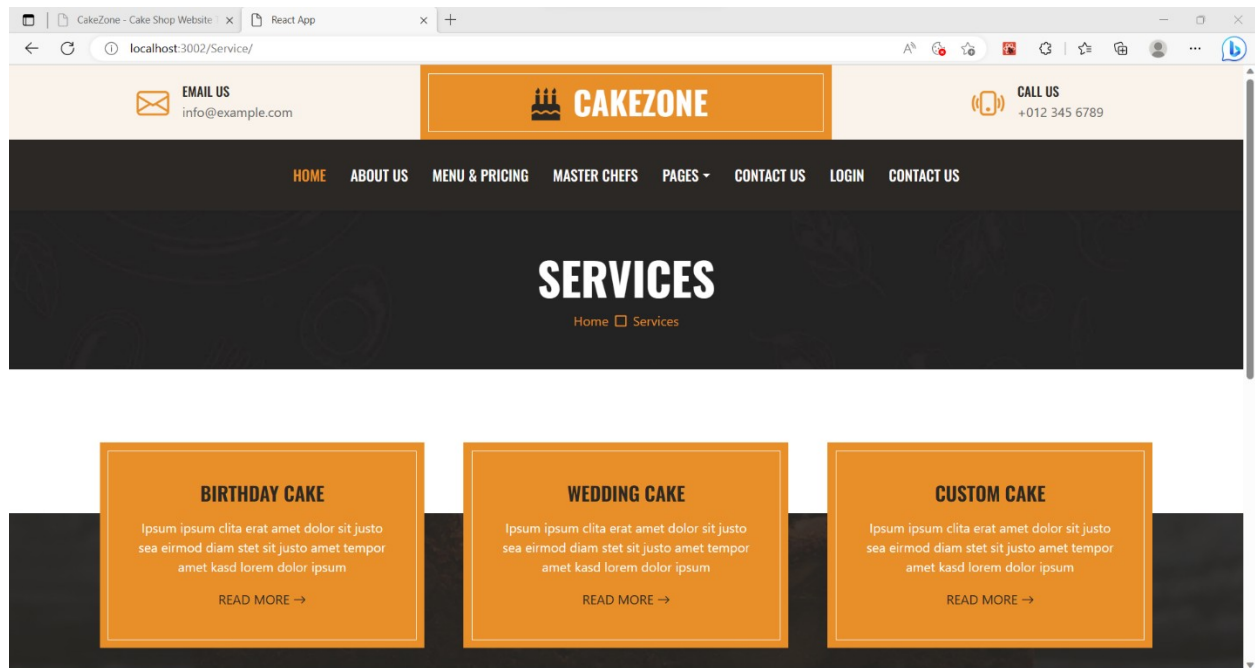
23.



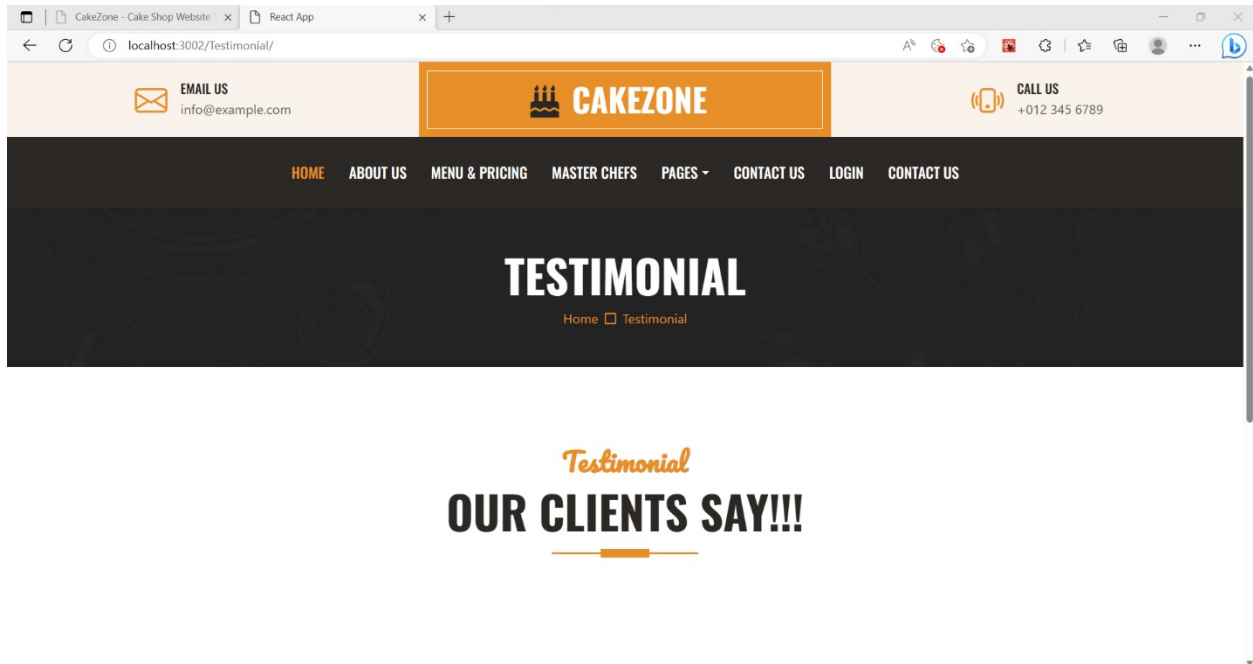
24.



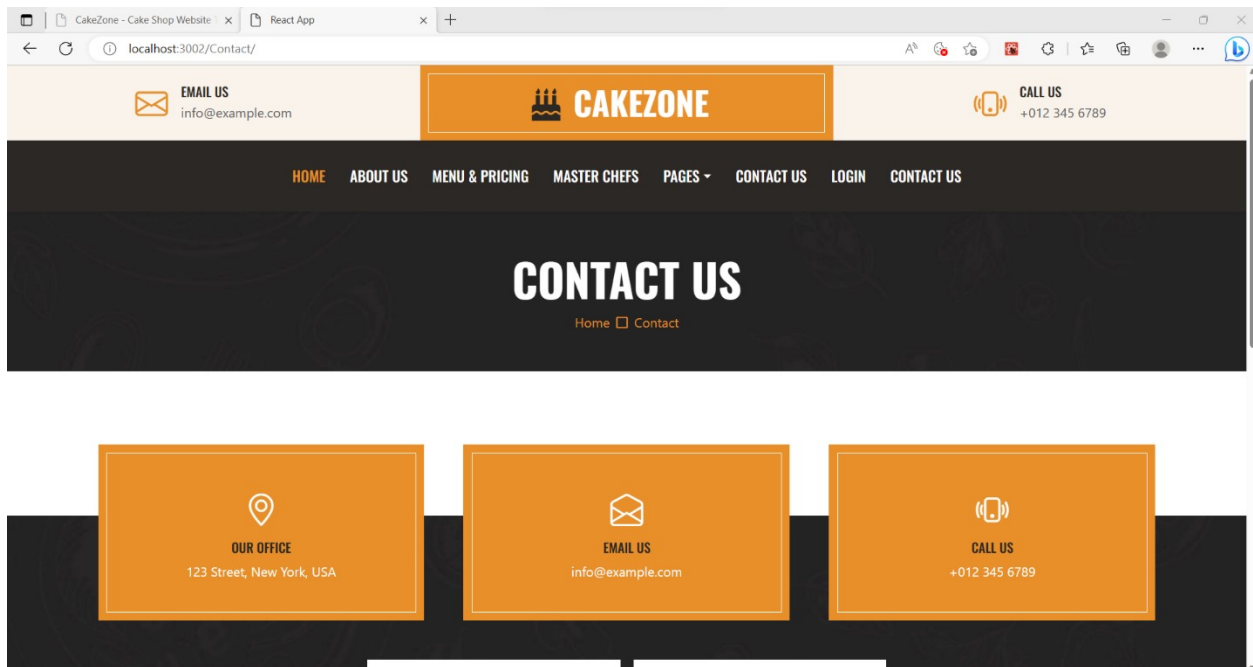
25.



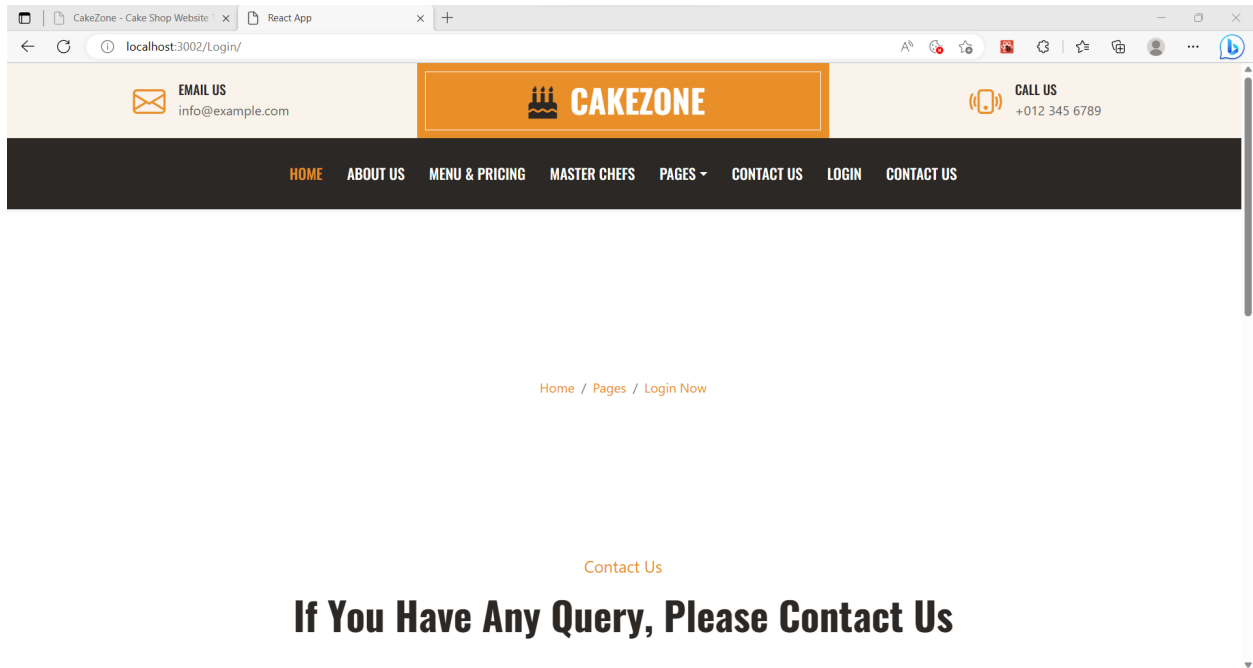
26.



27.



28.



REFERENCES:

- ❖ <https://www.w3schools.com>
- ❖ <http://www.google.com>
- ❖ <https://getbootstrap.com/docs/4.0/getting/started/download/>
- ❖ <https://docs.djangoproject.com>