**CHAPTER 2**

**BACKGROUND THEORY**

**2.1. Next.js Framework**

Next.js is a popular open-source framework for building server-rendered React applications. It was created by Vercel (formerly known as Zeit). It's designed to make the development of React applications easier by providing features that enhance performance, SEO, and developer experience. The framework is based on various concepts and theories that contribute to its functionality and design philosophy.

Next.js is built on top of React, a JavaScript library for building user interfaces. React allows developers to create reusable UI components and efficiently update the user interface when data changes.

One of the key features of Next.js is its support for Server-Side Rendering. SSR allows web pages to be pre-rendered on the server and sent to the client as HTML. This improves page load times and can enhance SEO, as search engines can easily index the content.

Next.js automatically splits your JavaScript bundles into smaller, optimized chunks. This means that only the code necessary for a particular page is loaded, reducing initial load times, and improving performance.

Next.js provides a file-based routing system. You can create pages by simply adding components to the pages directory. The framework handles the routing for you, making it easy to create a logical and SEO-friendly URL structure.

Next.js allows you to create API routes easily. These routes can serve as a backend for your application, making it possible to build full-stack applications within the same codebase.

**2.2. Express.js**

Express.js is a widely used open-source web application framework for Node.js. It's designed to simplify the process of building web applications and APIs by providing a robust set of tools and utilities.

Express.js is built on top of Node.js, a server-side JavaScript runtime. Node.js

allows developers to run JavaScript code on the server, enabling server-side programming using the same language as the client-side (browser) code. At the core of Express.js is the concept of middleware. Middleware are functions that can be added to the request-response pipeline to perform various tasks such as authentication, data validation, logging, and more. Middleware functions can be added globally to the entire application or to specific routes.

Express.js provides a simple and flexible routing system. Routes allow you to define how your application responds to different HTTP methods (GET, POST, PUT, DELETE, etc.) and URL patterns. Each route can have a corresponding handler function that gets executed when the route is matched. Express.js follows the principles of Representational State Transfer (REST). It supports various HTTP verbs, such as GET, POST, PUT, and DELETE, which correspond to CRUD (Create, Read, Update, Delete) operations. This makes it easier to build RESTful APIs. Express.js provides request and response objects that encapsulate information about incoming HTTP requests and outgoing HTTP responses. These objects are used to access parameters, headers, body data, and other relevant information.

While not a core feature, Express.js can be used with various template engines (like EJS, Handlebars, or Pug) to render dynamic HTML content on the server before sending it to the client. Express.js is often used to build RESTful APIs due to its simplicity and flexibility. With Express, you can define routes and handlers that interact with a database or other data sources to expose API endpoints.

The underlying philosophy of Express.js is to provide minimal abstractions and a lightweight framework that allows developers to quickly build web applications and APIs. Its modular approach, middleware architecture, and focus on simplicity have made it a foundational tool in the Node.js ecosystem.

**2.3. Google Authentication**

Google authentication, often referred to as Google Sign-In, is a method of allowing users to log in to websites and applications using their Google accounts. It's built on top of the OAuth 2.0 protocol and OpenID Connect, and it provides a secure and convenient way for users to access various online services without needing to create new accounts for each service.

OAuth 2.0 is a widely used open standard for access delegation. It allows third-party applications to obtain limited access to a user's account without exposing

their credentials. In the context of Google authentication, OAuth 2.0 is used to grant access to a user's Google account data.

Google Authentication is secure because it uses industry-standard protocols and practices. It ensures that the application never directly handles the user's credentials. Instead, the authentication flow happens on Google's servers, and the application only receives tokens that grant access to the user's data.

Google Authentication is widely used by developers to provide a seamless and secure login experience for users while leveraging the robust identity infrastructure provided by Google. It's especially popular in applications that want to offer users the convenience of logging in with their existing Google accounts rather than creating new credentials.

**2.4. MongoDB**

MongoDB is a popular open-source NoSQL database management system that is designed for flexibility, scalability, and ease of development. Unlike traditional relational databases, MongoDB uses a document-oriented data model that stores data in a flexible, JSON-like format called BSON (Binary JSON).

MongoDB falls under the category of NoSQL databases. NoSQL stands for "not only SQL," and these databases are designed to handle large volumes of unstructured or semi-structured data. They offer flexible schemas and are well-suited for applications that require scalability, performance, and agility.

In MongoDB, data is stored in documents, which contain data fields and values. These documents are like JSON objects and are stored in a collection. Unlike rows in relational databases, documents in MongoDB can have varying structures within the same collection.

BSON is a binary representation of JSON data that MongoDB uses to store and transmit documents. BSON supports additional data types like dates and binary data, making it more suitable for storage and querying.

MongoDB supports the standard CRUD operations (Create, Read, Update, Delete) for working with documents. The operations are performed using the MongoDB Query Language (MQL) which allows you to filter, project, and manipulate documents.

MongoDB's architecture and design are aimed at addressing the challenges of

modern applications, such as handling diverse and rapidly changing data, supporting

horizontal scalability, and providing developers with a flexible and efficient way to interact with their data.

**2.5. Async and Await**

Asynchronous programming is a programming paradigm used to handle tasks that may not be completed immediately or tasks that involve waiting for some external event to occur, like I/O operations (reading/writing files, making network requests, etc.). In traditional synchronous programming, the program execution waits for each task to be completed before moving on to the next one. Asynchronous programming, on the other hand, allows the program to continue executing other tasks while waiting for certain operations to finish.

In many programming languages, asynchronous programming was traditionally handled using callbacks. However, callbacks can lead to complex and hard-to-read code, especially when dealing with multiple asynchronous operations or nested callbacks. This is where concepts like async and await come into play.

An async function is a special type of function that can be used to define asynchronous behavior. When you declare a function as async, it means that it will always return a Promise. The Promise represents a future value that may or may not be available immediately. Inside an async function, you can use the await keyword to pause the execution of the function until the awaited Promise is resolved.

The await keyword is used within an async function to pause the execution of the function until the awaited Promise is resolved. This allows you to write asynchronous code that resembles synchronous code in terms of readability and structure. When you use await, the event loop is not blocked; other tasks can continue executing while the awaited Promise is being resolved.

async and await are features in languages like JavaScript (in both Node.js and browser environments) and other languages with asynchronous capabilities. They are widely used to handle asynchronous operations in a more structured and human-readable manner.