

**Frontend**

**React JS:** A JavaScript library for building interactive user interfaces using reusable components.

**Tailwind CSS:** A utility-first CSS framework for rapidly styling web applications with pre-defined classes.

**Zustand:** A lightweight state management library for React applications, offering a simple API for managing app state.

**Daisy UI:** A component library built on Tailwind CSS, providing pre-designed and customizable UI components.

**Backend**

Node JS

Express JS

Mongo DB

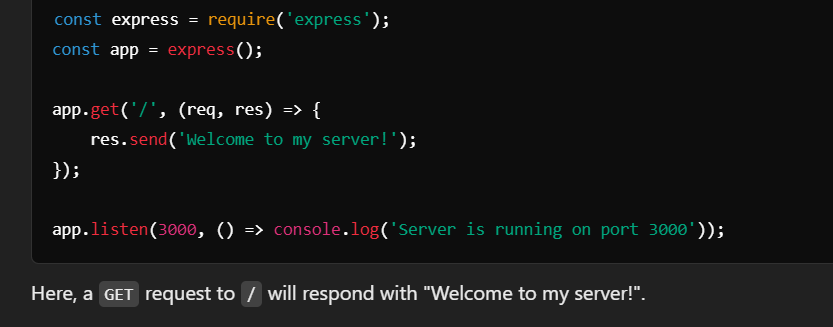
Socket.io

In backend we have used :

1. **express**:
   * A fast and minimalist web framework for Node.js, used to create server-side applications and APIs.
   * Express is like the "waiter" in a restaurant. It handles requests (e.g., a customer ordering food) and sends responses (e.g., delivering food to the customer).

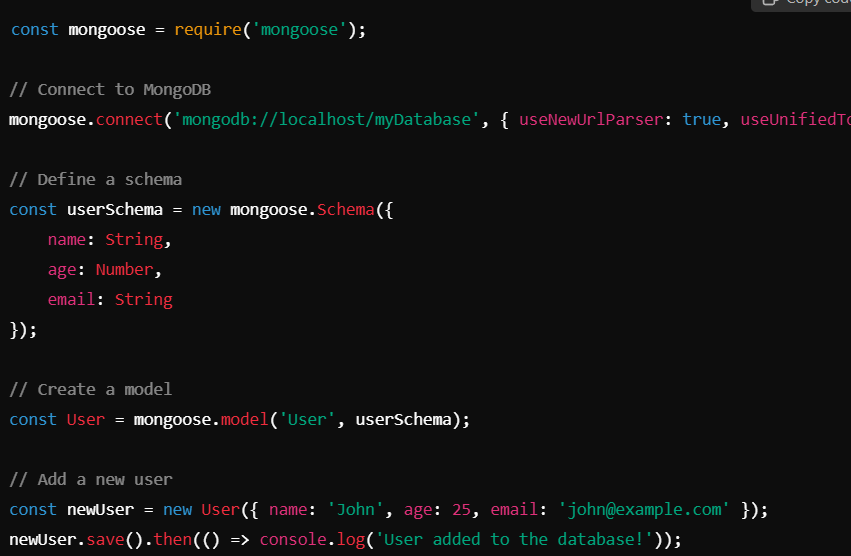
**Key Features:**

* Handles HTTP requests (GET, POST, etc.) easily.
* Allows middleware for processing requests.
* Makes API creation straightforward.



1. **mongoose**:

* A MongoDB object modeling tool, allowing you to define schemas and interact with MongoDB easily.
* Mongoose acts as a "translator" between your code and MongoDB (a database). It helps you structure your data (using schemas) and interact with MongoDB easily.



**Key Features:**

* Defines schemas for your data (like defining the structure of a form).
* Provides powerful methods to interact with the database (find, save, update, delete).

1. **dotenv**:
   * Loads environment variables from a .env file into process.env, useful for managing sensitive credentials.
   * Imagine you have a box where you keep secret keys to your house. .env is like that box, storing sensitive credentials like API keys or database passwords securely.

**Key Features:**

* + Loads environment variables from a .env file into process.env.



1. **jsonwebtoken**:

JWT is like a digital ID card. It’s a token that securely identifies a user in an application.

**Key Features:**

* Used for authentication and secure data exchange.
* Signed with a secret key to prevent tampering.
* Implements JSON Web Tokens (JWT) for secure authentication.

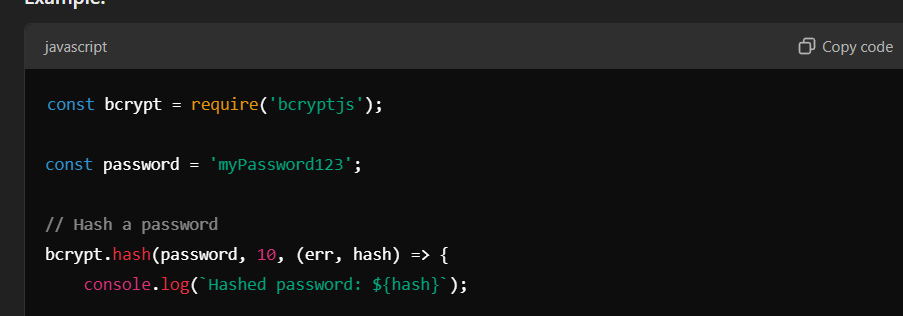


1. **bcryptjs**:
   * A library to hash passwords and compare them securely.

Bcrypt.js is like a "safe vault" for passwords. Instead of storing plain text passwords (dangerous!), it stores hashed (encrypted) versions.

**Key Features:**

* Hashes passwords to enhance security.
* Verifies hashed passwords during login.

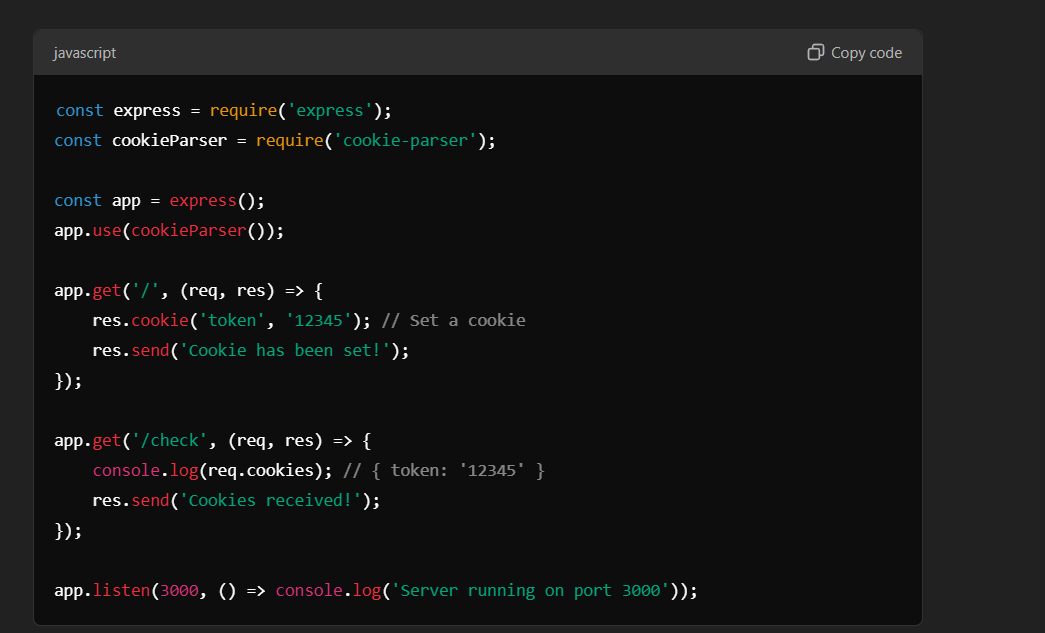
****

1. **cookie-parser**:

Cookie-parser helps "read the cookies" sent by the client. Think of cookies as small pieces of data stored in your browser to remember things like your login session.

**Key Features:**

* Parses cookies attached to incoming requests.
* Useful for session and user state management.
* Parses cookies attached to incoming requests, useful for handling session management.



1. **cloudinary**:

* Cloudinary is like a cloud storage service for images and videos. It allows you to upload, store, and manage media files.

**Key Features:**

* Easily handles image/video uploads.
* Provides URLs for the uploaded media.
* Provides easy integration with Cloudinary's cloud-based image and video management services.

1. **socket.io**:

* Socket.io is like a two-way walkie-talkie. It enables real-time communication between a client (browser) and a server.

**Key Features:**

* Facilitates instant messaging, notifications, or live updates.
* Uses WebSockets for fast, bidirectional communication.
* A library for enabling real-time, bidirectional communication between a client and a server.

These libraries work together in a MERN stack project to handle the server, database, authentication, real-time messaging, and more.

Imagine building a chat app:

* **Express**: Manages routes and handles requests.
* **Mongoose**: Stores chat messages in MongoDB.
* **JWT**: Secures user authentication.
* **Socket.io**: Enables live messaging.
* **Cloudinary**: Manages media uploads.

**Nodemon** is a utility that monitors for changes in your Node.js application and automatically restarts the server whenever a file is modified. This is especially useful during development because it eliminates the need to manually stop and start the server every time you make a change in your code.

**Why Use Nodemon?**

* **Efficiency**: Saves time by automatically restarting the server on code changes.
* **Convenience**: Avoids the manual need to stop and restart the server repeatedly, improving developer workflow.

In package.Json

With "type": "module", you can use **ES Module syntax** with import and export instead of require and module.exports.

