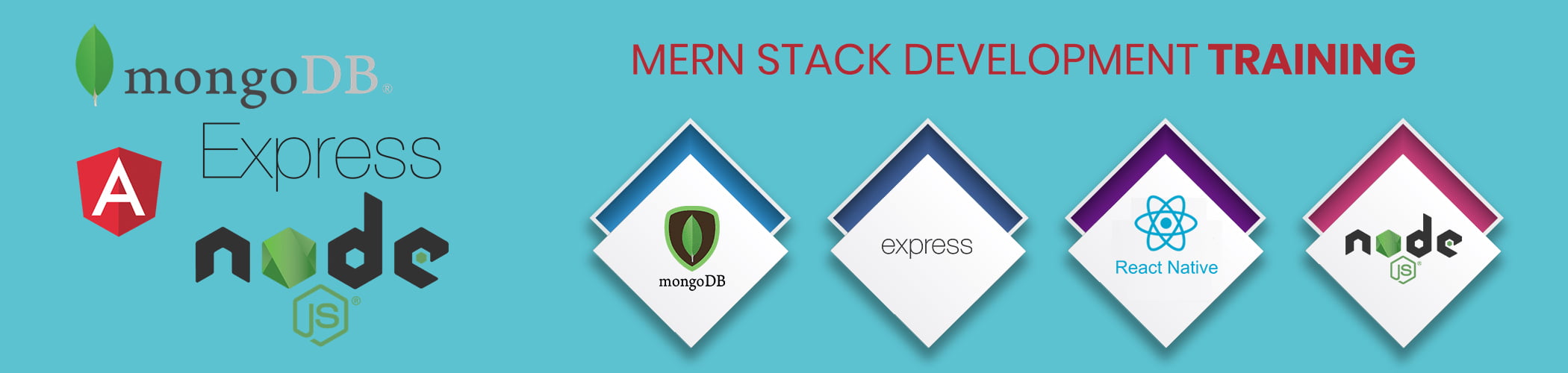
**ES – TRAINING**

**3. Enironment Setup – MERN**

****

**Content**

1. **Introduction**
2. **MERN Stack Architecture**

**3.MERN Stack components**

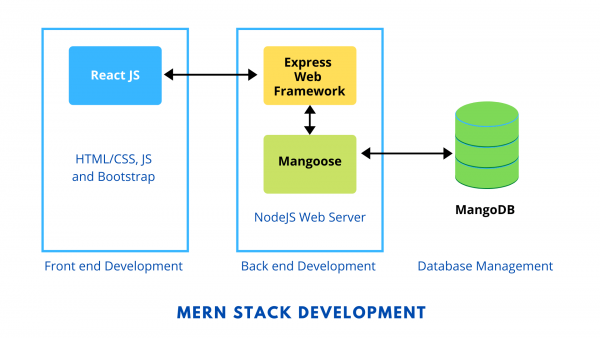
1. **VS Code Installation**
2. **Implementation of NodeJS & Express**
3. **Configuration of MongoDB & Mongoose**
4. **Setup of ReactJs**

**Introduction**

The phrase MERN stack refers to the following technologies:

* [**MongoDB**](https://www.mongodb.com/), a cross-platform document-oriented database program
* [**Express.js**](https://expressjs.com/), a web application framework for Node.js
* [**React**](https://reactjs.org/), a JavaScript library for building user interfaces
* [**Node.js**](https://nodejs.org/en/), an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser

**MERN stack Architecture**

****

**MERN Stack components**

1. **MongoDB: Cross-platform Document-Oriented Database**

MongoDB is a NoSQL database where each record is a document comprising of key-value pairs that are similar to JSON (JavaScript Object Notation) objects. MongoDB 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 MongoDB. Once MongoDB 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 (eg: querying, updating records, deleting records).

**Why use MongoDB?**

* 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.

**2. Express: Back-End Framework:**   
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 middlewares which makes the code shorter and easier to write.

**Why use Express?**

* 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
* Create a new folder to start your express project and type below command in the command prompt to initialize a package.json file. Accept the default settings and continue.

1. **React: Front-End Library**

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.

**Why use React?**

* 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.
* Developing Android/Ios Apps – With React Native you can easily code Android-based or IOS-Based apps with just the knowledge of JavaScript and ReactJS.
* You can start your react application by first installing **“create-react-app” using npm or yarn.**

1. **Node.js: JS Runtime Environment**

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

**Why use Node.JS?**

* 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.

**VS Code –Installation**

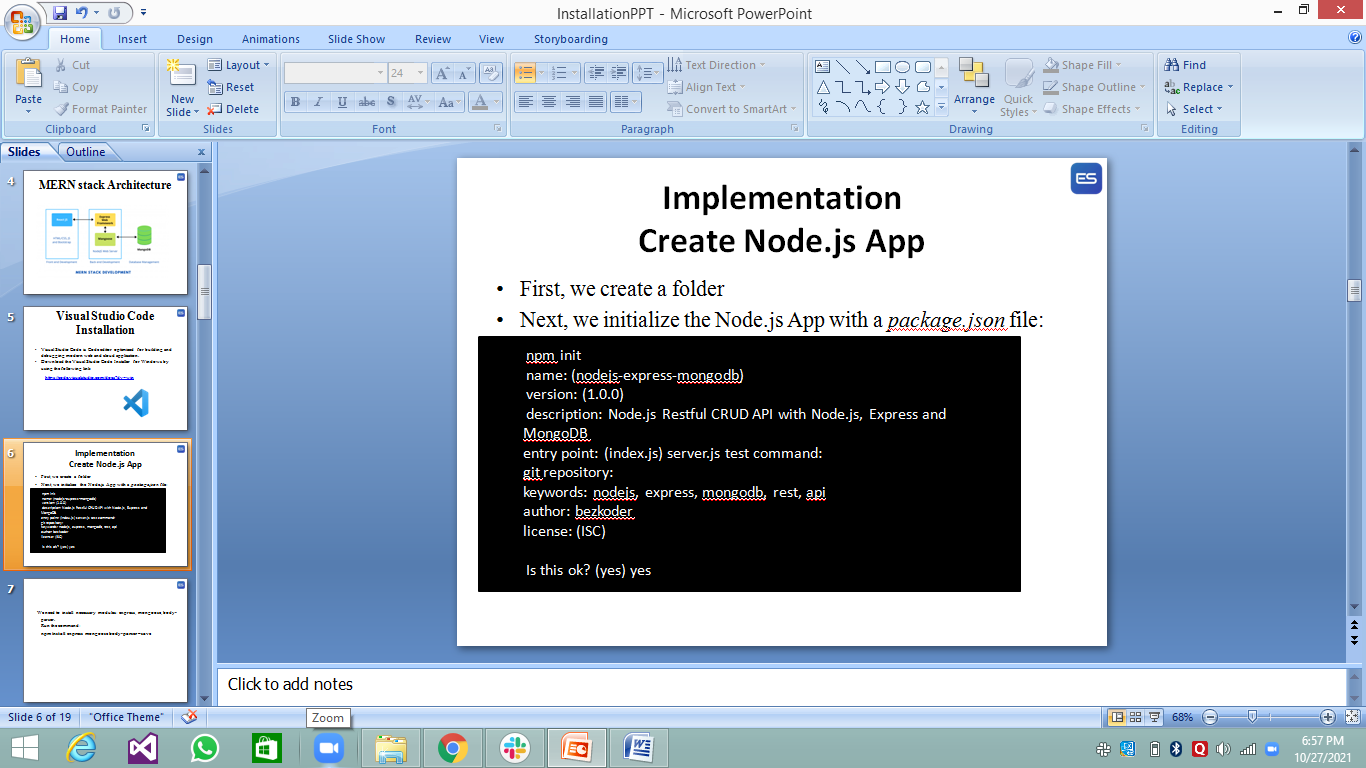
Visual Studio Code is an Integrated Development Environment made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

* Visual Studio Code is Code editor optimized for building and debugging modern web and cloud application.
* Download the Visual Studio Code Installer for Windows by using the following link

[**https://code.visualstudio.com/docs?dv=win**](https://code.visualstudio.com/docs?dv=win)

**Implementation  
Create Node.js App**

* First, we create a folder
* Next, we initialize the Node.js App with a **package.json** file:

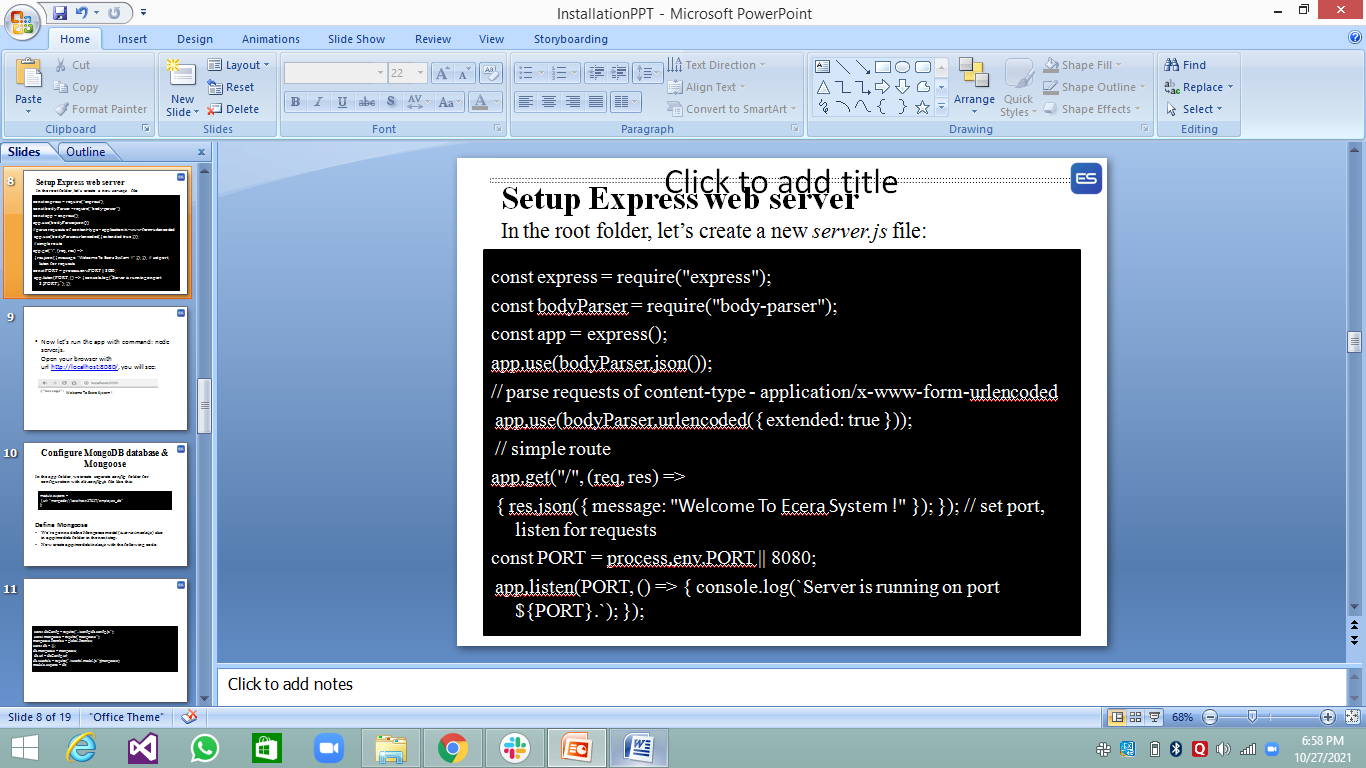


We need to install necessary modules: express, mongoose, body-parser.  
Run the command:

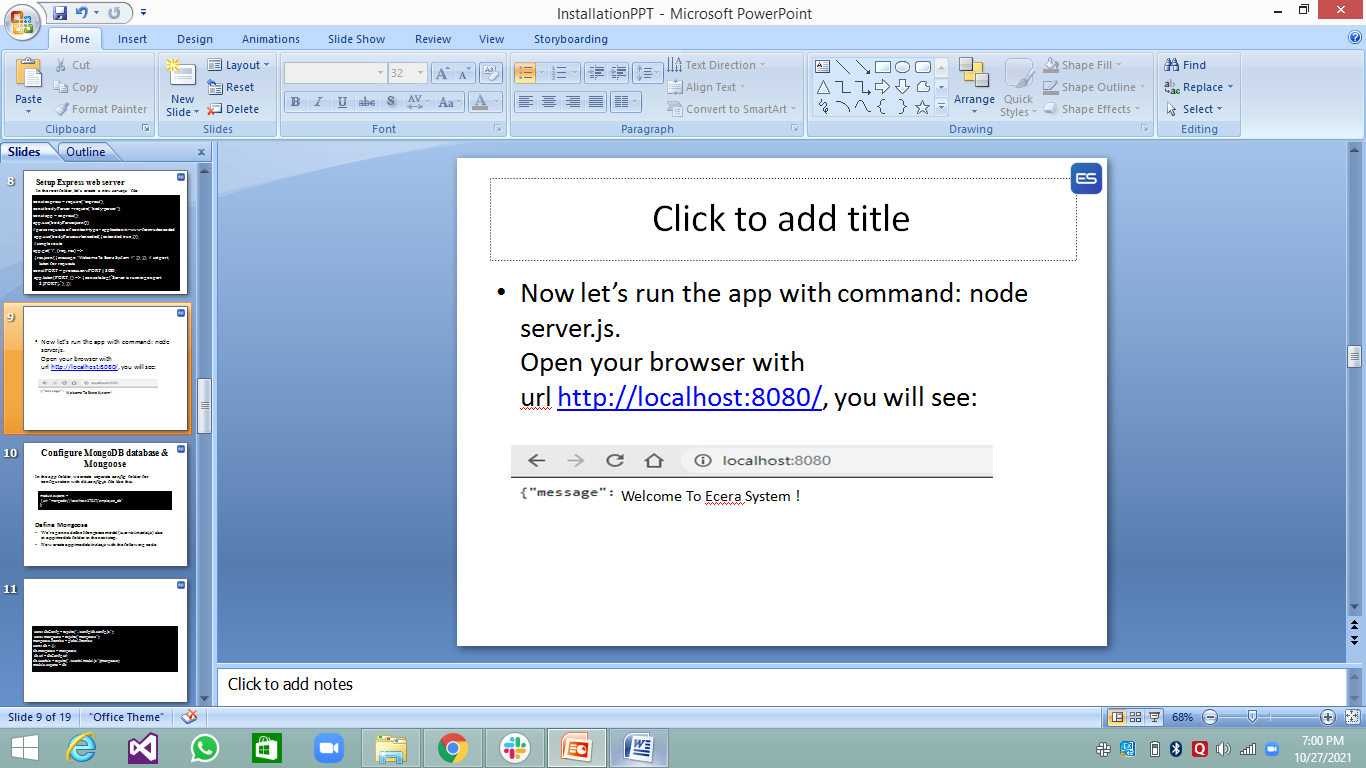
**npm install express mongoose body-parser –save**

**Setup Express web server**

In the root folder, let’s create a new **server.js** file:

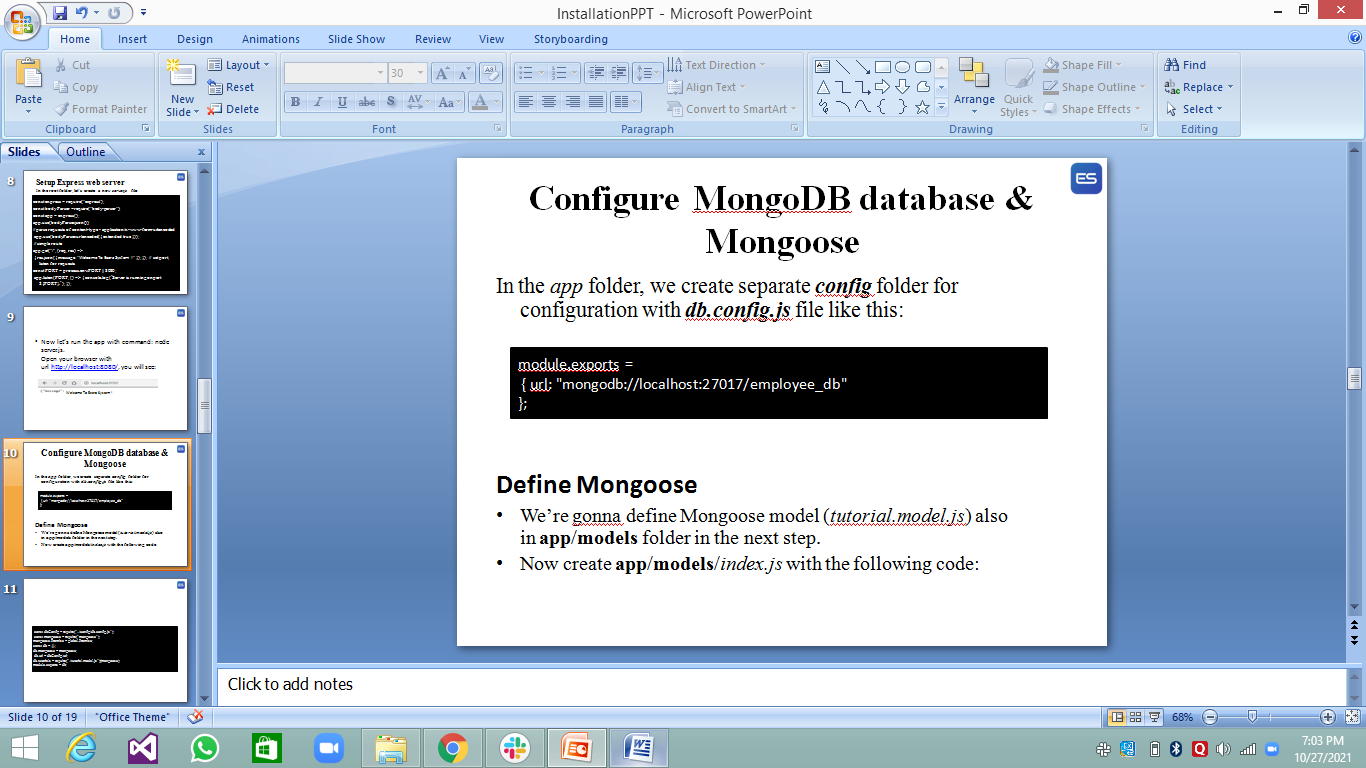


* Now let’s run the app with command: node server.js.  
  Open your browser with url <http://localhost:8080/>, you will see:



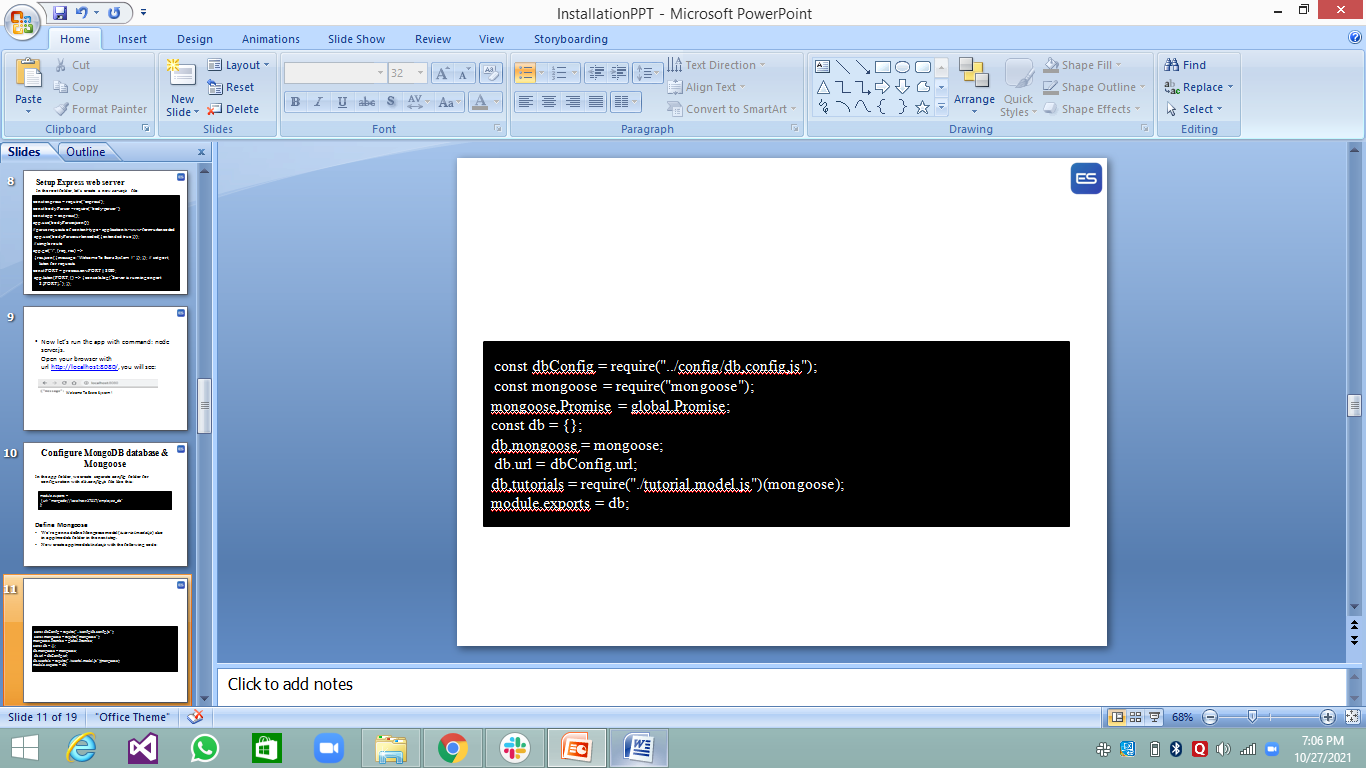
**Configure MongoDB database & Mongoose**

In the *app* folder, we create separate ***config*** folder for configuration with ***db.config.js***file like this:

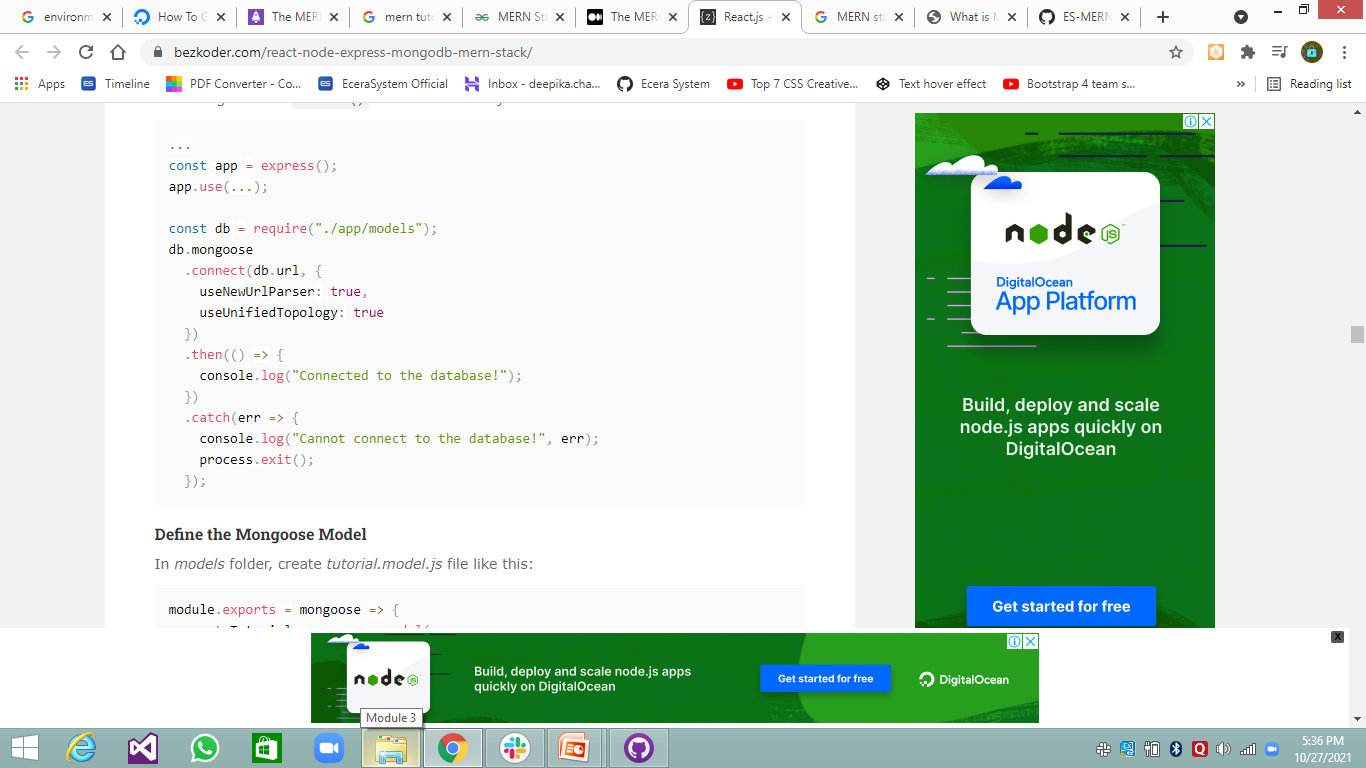


**Define Mongoose**

* We’re gonna define Mongoose model (*tutorial.model.js*) also in **app**/**models** folder in the next step.
* Now create **app**/**models**/*index.js* with the following code:

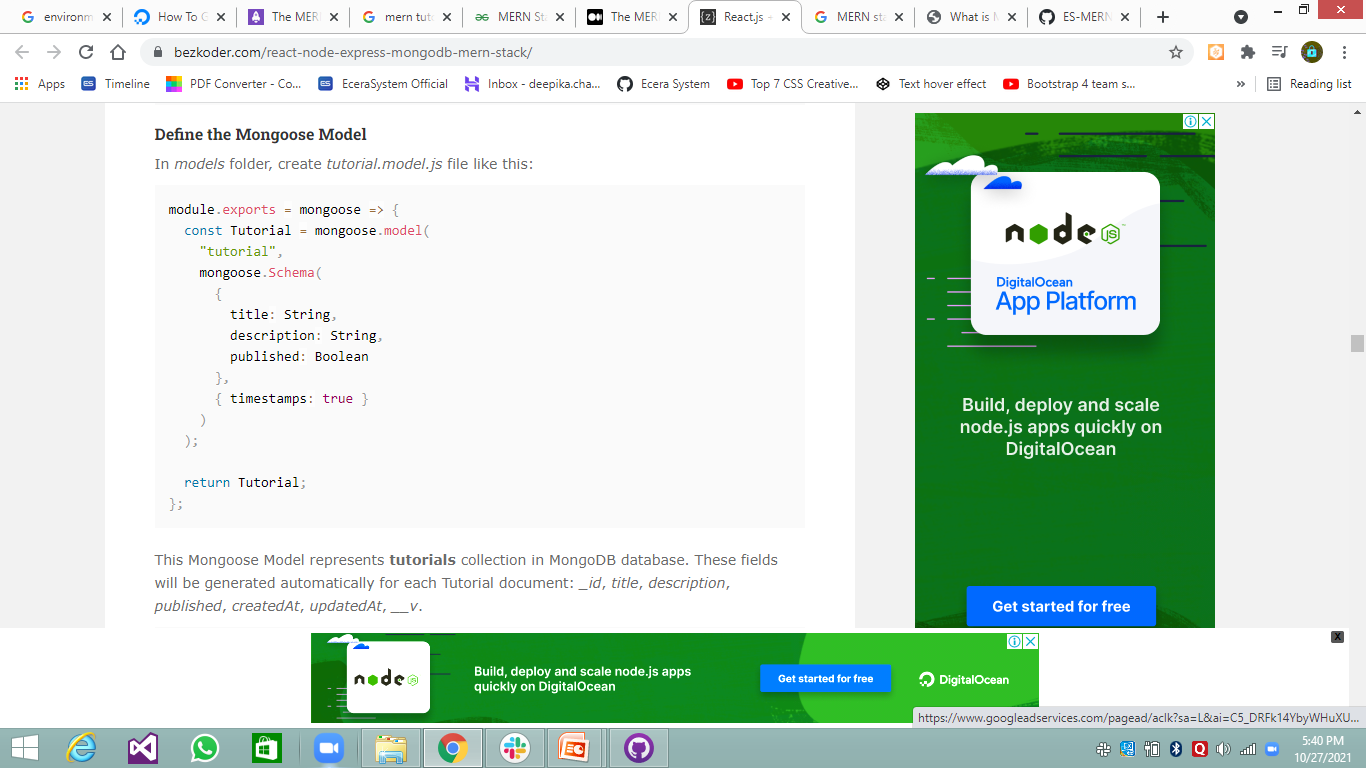


Don’t forget to call connect() method in **server.js:**

****

**Define the Mongoose Model**

In *models* folder, create *tutorial.model.js* file like this:



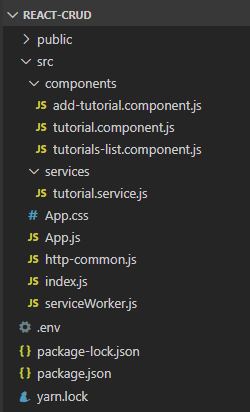
**Run the Node.js Express Server**

Run our Node.js application with command: node server.js.

**React.js Front-end**

**Technology**

* React 16
* react-router-dom 5.1.2
* bootstrap 4.4.1
* **Project Structure**
* – **package.json** contains 4 main modules: react, react-router-dom & bootstrap.  
  – App is the container that has Router & navbar.  
  – There are 3 components: TutorialsList, Tutorial, AddTutorial.  
  – TutorialDataService has methods for sending HTTP requests to the Apis.  
  – **.env** configures *port* for this React CRUD App.



**Implementation**

**Setup React.js Project**

* Open cmd at the folder you want to save Project folder, run command:  
  **npx create-react-app react-crud**

When you first run create-react-app. you’ll end up with a folder like this -> fig

**src/api.js** - You’ll probably need to make calls to a backend API at some point. Put all that code here. If it gets too unwieldy in one file, make a src/api directory and put the area-specific API files under there – like userApi.js, productApi.js, etc.

**src/components** - All your Presentational (aka Dumb) components go here. These are the simple stateless ones that just take props.

**src/containers** - The Container components go here. These are the stateful ones, and the ones that make the API calls. If you’re using Redux, these are the ones that are connected to the store. Notice that CSS and tests are in the same folder as their respective components.

**src/images** - Put the images in one place to start with.

**src/index.js** - This is where you initialize the app and call ReactDOM.render, so it makes sense to keep this at the top level.

**src/utils** - You’ll probably end up with miscellaneous utility functions – error handlers, formatters, and the like. I usually put them in a file inside utils so I can access them easily.

Index.js -> App.js -> calls components inside this file

