**MERN STACK**

**MONGODB**

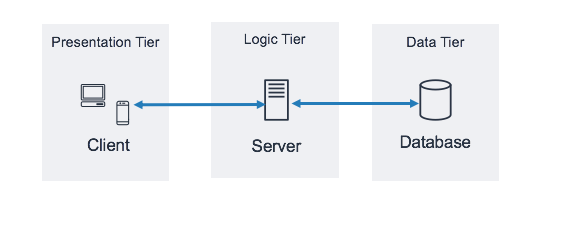
**EXPRESS**

**REACT - FRONTEND**

**NODE**

****

* Three-Tier Architecture



* Client - Frontend - User Interface - Presentation
* Backend or Server side - Server and Database

**Common Technologies Used**

* SERVER - Javascript, PHP,Python, Ruby on Rails, Java (SpringBoot), C (.Net)
* CLIENT - HTML, CSS, BOOTSTRAP, JAVASCRIPT, FLUTTER, REACT
* DATABASE
* Relational Databases (RDBMs) - mySQL, Oracle, and PostgreSQL,Microsoft Access
* Non-Relational Databases (Non-RDBMs) - mongoDB, Firebase, redis,supabase

**Creating a Node Package**

NPM - NODE PACKAGE MANAGER - npmjs.com

NPX - NODE PACKAGE EXECUTE

To create a node package, run (on your terminal) either:

1. npm init - Answer the series of questions that follow.
2. npm init -y - To skip the questions.

**Folder/File Structure**

* index.js (or another entry file): This is the main entry file of the project, where the application is typically initialised and configured. It may include setting up the server, registering routes, connecting to databases, or other necessary initialization steps.
* package.json: contains metadata about the project, including its name, version, description, author, license, and other project-specific configurations. One of the most crucial parts of the package.json file is the "dependencies" section. This section lists the packages (dependencies) that the project requires to run correctly.
* package-lock.json: a generated lock file that keeps track of the exact versions of the installed dependencies to ensure consistency among different development environments

NODE

EXPRESS - A JavaScript server-side framework

IMPORTING MODULES IN JAVASCRIPT

* ECMASCRIPT MODULES - import something from “somewhere” // React
* COMMON JS MODULES - require(“nameofpackage”) //Node

Frontend (Request) - Server (Response)

YOU (FRONTEND) - WAITER (SERVER) - CHEF (DATABASE)

REQUEST METHODS

GET REQUEST - Fetch information from the server - posts on facebook, list of videos on youtube.

POST REQUEST - Sending along payloads (additional information) - Create an account on facebook.

PUT REQUEST

PATCH REQUEST

DELETE REQUEST

Any request made through the browser url is a GET request

REQUESTS AND RESPONSES

FRONEND <-> SERVER

REQUEST <-> RESPONSE

CUSTOMER - WAITER/WAITRESS -

NODEMON

TEMPLATE ENGINES - EJS - EMBEDDED JAVASCRIPT //TEMPLATE ENGINES

JSX - JAVASCRIPT XML

**ASSIGNMENTS**

* Create a Music API
* Images and External CSS in Node application.
* Sign Up Page (firstname,lastname,email,password)
* Sign In Page (email,password)
* Dashboard Page (I am the dashboard)
* Read up on Request methods
* EJS SNIPPETS
* Checkout previous resources

**EJS - Embedded Javascript**

**ASSIGNMENTS**

* **Check out all ejx syntax**
* **Complete the student portal,display the student details in another route called dashboard,create buttons to delete and edit students**
* **Create a noise maker app.**
* **Create an account on mongoDB (May 2023, video 7)**

NOISE MAKER APP

ADD NOISE MAKERS

DISPLAY NOISE MAKERS - IN A TABLE - (+)(-) DELETE

FRONTEND <-> SERVER <-> DATABASE

MongoDB is a noSQL Database/ Non-relational Database

Cluster - Database - Collections/Table - Documents(Rows and Columns)

Schema (Structure of your Collection) (data type,constraints (unique,required,default)

Database - School Portal

Collections/Table

allStudents -

(id,

Firstname ,

Lastname,

Email,

password)

allStaffs

allCourses

Schema - Structure of your collection

Documents

Mongoose

.env (environment variables)

Dotenv

Npm i mongoose dotenv

MONGODB STRUCTURE

Cluster - Collections - Database - Table/Documents

Cabinet of Documents

School’s Database - Information of students, Information of staffs, Information of courses

E-commerce - Products, Buyers,Sellers

Database Schema - Structure of your database

Model - table name, schema

Model - Data Structure

School Portal Case Study

Firstname - String,Required

Lastname - String, Required

Email - String,Required,Unique

Password - String, Required

creationDate - Date,default:Date.now

ASSIGNMENT

COMPLETE THE SCHOOL PORTAL, BUT SAVE THE INFORMATION INTO THE DATABASE INSTEAD, BE ABLE TO DELETE AND ALSO EDIT FROM THE DATABASE

**MODULARIZATION**

**MVC ARCHITECTURE - MODEL - VIEW - CONTROLLER (Routes)**

**MODEL - SCHEMA**

**VIEW - WHAT IS BEING DISPLAYED TO THE USER**

**CONTROLLER - LOGIC**

**ROUTES -**

**AUTHENTICATION - To confirm is a user exists.**

**AUTHORIZATION - Gives the user access and privileges to protected resources**

1. **The user is registered**
2. **Whenever your user registers, you want to hash their passwords.**
3. **Authenticate the user.**

**BLOG**

**A user can post blogs**

**blog\_collection**

**allBlogs = [**

**{title:”how to become a billionaire in 100 days”,body:”lorem ipsum dolor siwhdihwid”,userid:”1”},**

**{title:”how to become a billionaire in 100 days”,body:”lorem ipsum dolor siwhdihwid”id:”2”},**

**{title:”how to become a billionaire in 100 days”,body:”lorem ipsum dolor siwhdihwid”,id:”1”},**

**]**

**users\_collection**

**allUsers = [**

**{firstname:”Abolade”,lastname”SQI”,email:”aboladesqi@gmail.com”,password:”fish”,userid:”1”},**

**{firstname:”Abolade”,lastname”SQI”,email:”aboladesqi@gmail.com”,password:”fish”.”2”},**

**{firstname:”Abolade”,lastname”SQI”,email:”aboladesqi@gmail.com”,password:”fish”,,”id:”1”},**

**]**

**Platforms to register**

**render.com**

**npmjs**

**cloudinary**

**POSTMAN**

**INSOMNIA**

**THUNDERCLIENT**

**Deploying to Vercel.**

1. **Create a file named vercel.json in the root folder of your application as follows**

**{**

**"version": 2,**

**"builds": [**

**{**

**"src": "server.js", // Replace this with the entry point of your server file**

**"use": "@vercel/node" // Specifies that this is a Node.js server**

**}**

**],**

**"routes": [**

**{**

**"src": "/(.\*)",**

**"dest": "server.js" // Replace this with the same entry point used in the "builds" section**

**}**

**]**

**}**

1. **Push to GitHub**
2. **Deploy to Vercel, add environment variables as well**

Using Nodemailer

Nodemailer https://nodemailer.com/about/

Sending email from the server side

1. Set up your node application

2. Run npm i nodemailer

3. Set up a route to test it out

4. Import nodemailer

5. Nodemailer requires two main things to work, the transporter (config for the email) as follows

let transporter = nodemailer.createTransport({

service : 'gmail',

auth : {

user : 'oyeniranoluwafemi36@gmail.com',

pass : ''

}

})

* service - The email service to be used
* user - Where the email is to be sent from
* pass - A passkey not necessarily the actual email address password. Temporary passwords can be created

as follows:

1. Navigate to your Google accounts > manage your Google account

2. Navigate to security > Signing in to Google > Turn on 2-step verification > App passwords

3. Select ‘other’ in the option, give it a name, and then generate and keep the password safe as

you won't be able to access it again

and mailoptions (the actual info about the mail) as follows

let mailoptions = {

from : 'oyeniranoluwafemi36@gmail.com',

to : 'oluwafemijohn1000@gmail.com',

subject : 'Testing Nodemailer',

text : 'The body of the message',

html :

}

6. Call on sendMail to send the message as follows

transporter.sendMail(mailoptions)

.then((info)=>{

console.log('success');

console.log(info);

})

.catch((err)=>{

console.log(err);

})