# Task 2 – Learn Full-Stack Basics

Internship Submission by: Eman Qamar

Internship Program: YoungDev Fullstack Internship

## Objective

Learn the fundamentals of both Frontend (ReactJS) and Backend (Node.js + Express + MongoDB) using the provided video tutorials. This helps in understanding how a full-stack web application works — from the user interface (frontend) to the server and database (backend).

## Part 1 – 10 Things I Learned from ReactJS (Frontend)

1. React is a JavaScript library used for building dynamic and interactive user interfaces.

2. React works through components, which are small, reusable UI blocks.

3. JSX allows us to write HTML code inside JavaScript, making code cleaner and more readable.

4. Props help pass data from one component to another, especially from parent to child.

5. State is used to store and manage data inside components.

6. Hooks like useState and useEffect make it easy to manage state and perform actions when data changes.

7. The Virtual DOM improves performance by updating only the parts of the page that actually change.

8. React Router helps in creating multiple 'pages' in a single-page web app.

9. Context API or Redux are used for managing shared data across components.

10. React makes it easier to integrate APIs to fetch and display data from a backend server.

### Code Snippet (Frontend – ReactJS Example)

import React, { useState } from 'react';  
  
function WelcomeMessage() {  
 const [name, setName] = useState('Eman');  
  
 return (  
 <div>  
 <h1>Hello, {name}! Welcome to React</h1>  
 <button onClick={() => setName('Developer')}>Change Name</button>  
 </div>  
 );  
}  
  
export default WelcomeMessage;

This React component uses state (useState) to change the name dynamically when the button is clicked, showing how React updates the UI automatically.

## Part 2 – 10 Things I Learned from Backend (Node.js + Express + MongoDB)

1. Node.js allows JavaScript to run on the server-side, not just in browsers.

2. Express.js is a framework that simplifies building web servers and REST APIs.

3. The HTTP methods (GET, POST, PUT, DELETE) are used for creating, reading, updating, and deleting data.

4. Middleware in Express handles requests before sending a response (e.g., logging, authentication).

5. MongoDB is a NoSQL database that stores data in flexible, JSON-like documents.

6. Mongoose is used to connect Node.js with MongoDB and define schemas and models.

7. CRUD operations (Create, Read, Update, Delete) are the core of backend development.

8. Environment variables (.env) help protect sensitive data like passwords or API keys.

9. Tools like Postman help test and debug APIs easily.

10. Backend development handles authentication, database communication, and routing between frontend and database.

### Code Snippet (Backend – Node.js + Express + MongoDB Example)

const express = require('express');  
const mongoose = require('mongoose');  
  
const app = express();  
app.use(express.json());  
  
mongoose.connect('mongodb://localhost:27017/testdb')  
 .then(() => console.log('MongoDB Connected'))  
 .catch(err => console.error(err));  
  
const UserSchema = new mongoose.Schema({  
 name: String,  
 email: String  
});  
const User = mongoose.model('User', UserSchema);  
  
app.post('/addUser', async (req, res) => {  
 const { name, email } = req.body;  
 const newUser = new User({ name, email });  
 await newUser.save();  
 res.send('User added successfully!');  
});  
  
app.get('/users', async (req, res) => {  
 const users = await User.find();  
 res.json(users);  
});  
  
app.listen(3000, () => console.log('Server running on port 3000'));

This backend code connects to a MongoDB database, defines a User model, and sets up two API routes — one to add a user and another to fetch all users. It shows the basic structure of a real backend server.

## Final Summary

By completing this task, I learned how both frontend and backend work together in a full-stack web app. React JS controls what users see and interact with, while Node.js, Express, and MongoDB manage how data is stored, processed, and delivered. This helped me understand the complete flow of a web application — from the browser to the server and then the database.