COURSENAME: BACKEND DEVELOPMENT FRAMEWORKS

FSD306

HND COT

FLEMMING EYONG

COURSE OUTLINE

MODULE 1: GENERAL OVERVIEW

- ✓ Web development technologies overview
- ✓ Database types overview

MODULE 2: INTRODUCTION TO REACT JS

- ✓ What is React Js
- ✓ Why React Js
- ✓ Installing an IDE

- ✓ React project overview
- ✓ Creating minimalist “hello world” standalone program
- ✓ What is JSX and why do we need it?

MODULE 3:REACT JS DEVELOPMENT ENVIRONMENT SE

TUP

- ✓ Getting started with Create-React-App starer Project
- ✓ Installing Project prerequisites
- ✓ Downloading project dependencies with NPM or Yarn

MODULE 4: REACT JS COMPONENTS

- ✓ React Js Components
- ✓ What are react Js Component?
- ✓ JavaScript function and Class Components in React Js
- ✓ Component life cycle
- ✓ Component props
- ✓ Parsing data from one component to another
- ✓ Styling React Js components

MODULE: ROUTING AND STATE MANAGEMENT IN REACT

JS

- ✓ Routing in React Js
- ✓ Basic state management with useRef, useMemo,useContex
  t, useEffect, useState
- ✓ Using state management libraries

MODULE 6: INTRODUCTION TO BACKEND DEVELOPMENT

- ✓ Node Js, Express Js and Mongo db setup
- ✓ Creating database connections
- ✓ Creating database models
- ✓ Using ORM in Express Js
- ✓ Crud application with Express Js
- ✓ Permissions in Express Js
- ✓ Authorization in Express Js
- ✓ Authentication in Express Js

MODULE 7:FULL STACK CONCEPT

- ✓ Integrating React Js with Express Js Back-end
- ✓ Crud application in React Js with Node Js back-end

COURSE OBJECTIVE:

By The End Of This Course Students Are Expected To Know

- ✓ Difference Between Frontend And Backend
- ✓ What React Js Is And How Is It Better
- ✓ How To Set Up Development Environment For React Js
- ✓ How To Build Web Applications With React Js
- ✓ How To Create Database Models Using Sequelize
- ✓ How To Create Api endpoints Using Express Js
- ✓ How To Render Data From Backend In React Js App

MODULE 1:

GENERAL OVERVIEW

1.1 Web development technologies overview:

Web development technologies encompass a wide range of tools, frameworks, and languages that are used to create and maintain websites and web applications. Here's an overview of some popular web development technologies.

1. **HTML (Hypertext Markup Language):** HTML is the standard markup language for creating the structure and content of web pages.

2. **CSS (Cascading Style Sheets)**: CSS is used to define the presentation an
d styling of web pages, including layout, colors, fonts, and other visual asp
ects.

3. **JavaScript**: JavaScript is a versatile programming language that enable
s interactivity and dynamic behavior in web pages. It allows you to manip
ulate page elements, handle events, perform calculations, and interact wit
h servers (Ajax).

4. **Front-End Frameworks**: Front-end frameworks like **React**, **Angular**, an

d **Vue.js** provide pre-built libraries and components to simplify and strea mline web development. These frameworks enable the creation of interacti ve user interfaces and facilitate the management of application state.

5. **Back-End Technologies**: Back-end technologies handle server-side logic and data processing. Some popular ones include:

- **Node.js**: Node.js is a JavaScript runtime environment that allows you to build server-side applications using JavaScript. It's known for its non-bloc king, event-driven architecture and is commonly used with frameworks li ke **Express.js.**
- **Ruby on Rails**: Ruby on Rails is a web application framework that uses t

he Ruby programming language. It follows the Model-View-Controller (MV C) pattern and emphasizes convention over configuration, enabling rapid development.

- **Django**: Django is a high-level Python web framework that follows the MVC architectural pattern. It provides a robust set of tools and features fo r building scalable and secure web applications.
- **ASP.NET**: ASP.NET is a web application framework developed by Micro soft, which is based on the .NET platform. It supports multiple programmi ng languages, including C# and Visual Basic, and provides various tools an d libraries for building web applications.

6. **Databases**: Databases are used to store and retrieve data in web applica
tions. Some commonly used databases include:

- **MySQL**: MySQL is an open-source relational database management syst
em (RDBMS) that is known for its speed, scalability, and ease of use.
- **PostgreSQL**: PostgreSQL is another open-source RDBMS that emphasize
s extensibility and compliance with SQL standards. It offers advanced feat
ures like support for JSON data and geospatial data types.
- **MongoDB**: MongoDB is a popular NoSQL database that stores data in fle
xible, JSON-like documents. It is known for its scalability and ability to ha
ndle unstructured and rapidly changing data.

7. **Web Servers**: Web servers, such as **Apache** and **Nginx**, handle the processing and delivery of web requests. They are responsible for serving static files, managing connections, and routing incoming requests to the appropriate application.

8. **DevOps Tools**: DevOps tools facilitate the development, deployment, and management of web applications. Some popular ones include:

- **Git**: Git is a widely used version control system that enables collaboration and code management.
- **Docker**: Docker is a platform that allows you to package applications and their dependencies into containers, ensuring consistency and portability

across different environments.

- **Jenkins**: Jenkins is an open-source automation server that helps automa
te various stages of the software development lifecycle, including building,
testing, and deployment.

9. **Cloud Services**: Cloud platforms, such as **Amazon Web Services (AWS)**,
Microsoft Azure, and **Google Cloud Platform (GCP)**, provide a wide array o
f services and infrastructure for hosting, scaling, and managing web applic
ations.

It's important to note that the web development landscape is constantly e volving, and new technologies and frameworks emerge regularly. Therefore, it's advisable to stay updated with the latest trends and advancements in the field.

1.2 Database types overview

A database is a structured collection of data that is organized, managed, a nd accessed using specific software and techniques. It provides a way to st ore, retrieve, modify, and analyze large amounts of data efficiently. Datab ases are widely used in various applications, including web development, b usiness systems, scientific research, and more.

Here's an overview of different types of databases:

1. **Relational Databases (RDBMS)**: Relational databases are the most comm
on type of databases. They store data in structured tables with predefined
 schemas, where data is organized into rows and columns. Relational data
bases use SQL (Structured Query Language) to manage and manipulate th
e data. Examples of relational database management systems (RDBMS) inc
lude MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.

2. **NoSQL Databases**: NoSQL (Not Only SQL) databases are non-relational

databases that provide flexible data models and horizontal scalability. They are designed to handle large-scale data and high-velocity data ingestion. NoSQL databases offer different data models, including key-value stores, d ocument stores, columnar databases, and graph databases. Examples of N oSQL databases include MongoDB, Cassandra, Couchbase, and Redis.

3. **Key-Value Stores**: Key-value stores are simple databases that store dat a as a collection of key-value pairs.They are highly scalable and efficient fo r retrieving data based on keys. Examples include Redis, Riak, and Amazo n DynamoDB.

4. **Document Databases**: Document databases store data in flexible, semi-s tructured documents, typically inJSON or XML formats. They provide hig h flexibility and are suitable for handling unstructured or rapidly changing data. Examples include MongoDB, Couchbase, and Elasticsearch.

5. **Columnar Databases**: Columnar databases store data in columns rather than rows, which allows for efficient data compression and retrieval. They are well-suited for analytical workloads and data warehousing. Examples include Apache Cassandra, Apache HBase, and Vertica.

6. **Graph Databases**: Graph databases are designed to store and process hi

ghly interconnected data, such as social networks, recommendation syste
ms, and knowledge graphs. They represent data as nodes, edges, and prop
erties, enabling efficient traversal and querying of graph structures. Exam
ples include Neo4j, ArangoDB, and Amazon Neptune.

7. **In-Memory Databases**: In-memory databases store data primarily in th
e main memory (RAM) rather than on disk, resulting in faster data access
 and processing. They are used for high-performance applications that req
uire real-time data processing. Examples include Redis, Memcached, and S
AP HANA.

8. **Time-Series Databases**: Time-series databases are optimized for storing
and analyzing time-stamped or time-series data, such as sensor data, log
files, and financial data. They provide efficient storage, retrieval, and anal
ysis of data points over time. Examples include InfluxDB, Prometheus, and
TimescaleDB.

9. **NewSQL Databases:** NewSQL databases aim to combine the benefits of t
raditional relational databases with the scalability and performance of No
SQL databases. They provide ACID (Atomicity, Consistency, Isolation, Dura
bility) compliance while supporting distributed architectures. Examples incl
ude Google Spanner, CockroachDB, and NuoDB.

It's worth noting that these database types are not mutually exclusive, an
d many databases blur the lines between different categories. The choice of
 a database type depends on the specific requirements of the application, i
ncluding data structure, scalability, performance, and consistency needs.

MODULE 2: INTRODUCTION TO REACT JS

2.1 Getting Started with React

In this section, we will create a minimalistic, stand-alone "Hello World" React example. We will be installing an integrated development environ ment (IDE), and we will cover crucial concepts such as JSX, the DOM, th e React virtual DOM, Babel, and ES5/ES6. First, though, we'll talk about what React is and why to use it. Although it may be more gratifying to jump right in and start writing code, paying attention to the key concep ts will help you understand React better. The entire book relies on you u nderstanding the ideas in this section.

2.2 What Is React?

React (also known as ReactJS) is a JavaScript library developed by Faceb ook (https:// github.com/facebook/react) to create user interfaces for th

e Web. React was invented by **Jordan Walke**, who was working at Facebo

ok Ads at the time. It competes with other front-end development fram

eworks and libraries such as jQuery, Angular, Vue.js, Svelte, etc.

2.3 **Why React?**

Did you know?

– React is a favorite among developers. In fact, according to a StackOve

rFlow survey, React.js has been the "most loved" web framework for two consecutive years.

– The need for React developers has ballooned; according to Indeed.com there are close to 56,000 open React developer positions.

– The React library itself is lighter than competing Web frameworks suc h as Ember or Angular, and fast.

– React is easy to get started with.

2.3.1 what is an IDE(integrated development environment)

An IDE is a software application that provides comprehensive set of tools and features to assist developers in writing, testing and debugging softa re.

2.3.2 what is JSX?

JSX stands for JavaScript XML, it is an extension of the JavaScript lang uage that allows developers to write HTML-like code within JavaScript. With JSX developers can define the structure and layout of UI componen ts using familiar HTML-like syntax, including tags, attributes, and neste d elements **e.g const element = <h2> Hello Full stack </h2>;**

2.3.3 what is DOM?

Document object model(DOM) is a programming interface for web docu ments, representing the structure of an HTML or XML document as a tr ee-like structure. It provides a way for programs or scripts to dynamical ly access and manipulate the content, structure, and style of a web page.

2.3.4 React Virtual DOM:

This is a concept and implementation used by the react JavaScript librar y. It is a light weight representation of the actual DOM that react main tains internally for efficient rendering and updating of components.

2.3.5 what is Babel?

Babel is tool commonly used in React Js ecosystem(as well as other javaS cript projects) for transcompiling JavaScript code. It allows developers to write modern javaScript code using latest language features and syntax while ensuring compatibility with older browsers and environments that may not support those feature.

MODULE 3

3.1 Installing an IDE

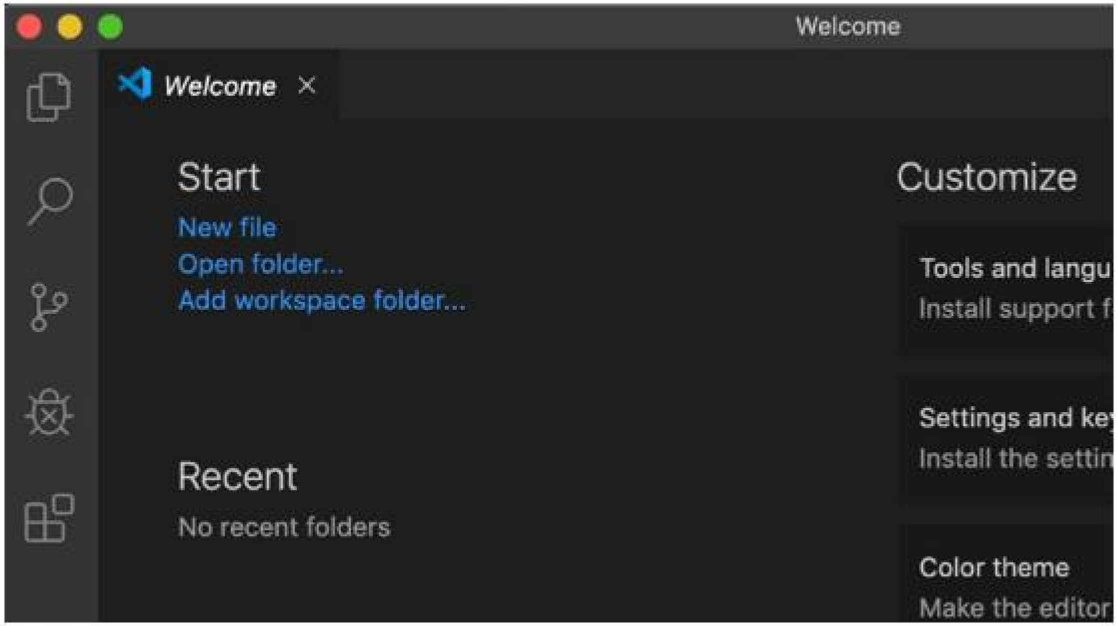
When working with code, it's recommended that you use an IDE.

3.2 Why Do I Even Need an IDE?

You don't "need" an IDE; you can always write your code with a text edi
tor. However, an IDE helps write people code and increases productivity.
This is done by providing common needed functionality for writing code,
 such as source code editor, build automation tools, and a debugger, as w
ell as many other functionalities. When it comes to IDEs, there are many
 to choose from.For this course, I selected Visual Studio Code (VS Code) si
nce it's a lightweight, free, cross-platform (Linux, macOS, Windows) edit
or that can be extended with plugins.

3.3 How to Install VS Code?

To get started, go to the VS Code download page: **https://code.visualstudio.com/download** Choose your platform, and once the download is complete, open VS Code. You should see the Welcome page shown below.



3.4 Creating a Minimalistic Stand-Alone "Hello World" Program

It's time to create our new "Hello World" app and run the project in ou r browser. To create a new file in VS Code, select New File. Then, paste i n the following code and save the file anywhere you want:

```
<html>
 <head>
 <meta charSet="utf-8">
 <title>Hello world</title>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/react/17.
0.0/
umd/react.production.min.js"></script>
 <script src="https://unpkg.com/react-dom@17.0.0/umd/react-dom.
production.min.js"></script>
 <script src="https://unpkg.com/@babel/standalone/babel.min.js
">
```

```
 </script>
 </head>
 <body>
 <div id="app"></div>
 <script type="text/babel">
 ReactDOM.render(
<h1>Hello World</h1>,
 document.getElementById('app')
 );
 </script>
 </body>
</html>
```

3.5 Getting Started with the Create-React-App (CRA) Starter Project

In the previous exercise, we created a simple "Hello World" application fr

om scratch, and we talked about how it works behind the scenes. We cov ered topics such as JSX, DOM, the React virtual DOM, Babel, ES5/ES6, and **SPA**s. Creating a React app was easy and took just few lines of code. However, to create a real-life application based on one page (an SPA) w ith multiple views, many user gestures, lists with hundreds or thousands of items, member areas, and other common pieces that are common in today's applications, there is much more we need to learn. In this book, my goal is to set you up with a large toolbox full of the best libraries and tools so you can build a top-grade React application and get the most o ut of React. For that, we need to look at several popular libraries and to ols in the world of React.

Head over to https://github.com/facebook/create-react-app to downloa d the create-react-app boilerplate project.

CRA has already made some decisions for us. For example, the build tool is a tool called **Webpack** over **Rollup** or **Parcel.** Task Runners is set up wit h **NPM scripts** over other tools such as Gulp. CSS, JS, and Jest are used as the defaults.

3.5.1 Create-React-App Prerequisites

CRA needs a node js runtime environment. Note that CRA doesn't come with all packages needed to build full web applications by default, howev er we can get whatever package we need using **NPM(Node Package Man ager).**

Install Node and NPM on a Mac/PC

you can install node js for both Mac or PC from here https://nodejs.org/en/ To check if installation was successful run **node -v** in your comman d prompt or terminal

3.5.2 Creating a new React Js Project Using Create-React_App

To use CRA we need to install it first, open your terminal or command p rompt, run the command **npm install -g create-react-app**

Make you're your computer is connected to the internet.

To create "hello world" app using CRA

➢ Create an empty project folder called **school**

➢ Open the school folder using vs code

- Activate a new terminal in vs code
- Run the command **npx create-react-app helloworld**

Note: Npx is the task runner for NPM

The command above will create a new react js application with the nam e **helloworld**.

3.6 React Js Application Structure

When you create a React application using Create React App (CRA), it ge nerates a default project structure with various directories and files. Her e's an overview of the structure and the function of each directory and fi le:

node_modules: This directory contains all the project's dependencies. It is created and managed by the package manager (e.g., npm or yarn) based on the dependencies specified in the package.json file.

public: This directory contains static assets that are directly copied to th e build folder during the build process. It typically includes the HTML file (index.html) that serves as the entry point for the application. Other ass ets like images, icons, and fonts can also be placed in this directory.

src: This is the main directory where most of the application's code resid es. It contains the source code files for the React components, styles, and other application-specific logic.

index.js: This is the entry point for the application. It renders the root co
mponent into the DOM and sets up the React rendering environment.

App.js or App.jsx: This is the default component that serves as the root o
f the application. You can modify or replace this component to define th
e structure and behavior of your application.

index.css or index.scss: This is the default CSS or SCSS file for the applica
tion. You can add global styles or import other CSS/SCSS files here.

components: This directory is commonly used to store reusable React co
mponents. You can create subdirectories within it to organize component
s based on their functionality or purpose.

assets: This directory can be used to store static assets like images, icons,
 or JSON files that are used within the application.

package.json: This file contains metadata about the project and specifies i
ts dependencies, scripts, and other configuration options. It is also used b
y the package manager to install, manage, and build the project.

package-lock.json or yarn.lock: These files are automatically generated an
d provide deterministic dependency resolution, ensuring consistent install
ations across different machines.

README.md: This file typically contains information about the project, i ncluding setup instructions, usage guidelines, and other relevant details.

These are the core directories and files typically found in a React applica
tion created using Create React App. However, it's important to note th
at the structure can vary depending on the specific requirements and pr
eferences of the project. As you develop your React application, you may
create additional directories and files to organize your code, assets, and
other resources.