Flask + React Project (Create Recipes)

1. Pip install flask, flask_RESTx, flask_sqlalchemy, flask_jwt_extended
2. **Config.py:** This will have all the configuration related to the project
3. **Libraries Using**
    a. **Flask:** For creating the web application
    b. **Flask_RESTx:** for building the rest api with the documentation and validation
    c. **FLASK_SQLAlchemy:** for database interaction using SQLalchemy ORM
    d. **FLASK_JWT_extended:** for handling authentication via JSON web tokens(JWT)
    e. **Python-decouple:** The python decouple library is used to manage configuration setting for your python application. It helps you to separate configuration parameter(like Database setting, API keys and other sensitive information) from your source code ,making cleaner and secure
4. **Set FLASK_APP = main.py:** This command is used to set the flask app environment variable. It tells the flask which python file to run/use as the entry point of our application
5. **$:FLASK_APP=”main.py” :** This is equivalent command in powershell for setting the environment variable to “main.py” after this we can use the flask run directly to run our application
6. **Serializer :** The serializer is the process of converting data into a format suitable for transmission and storage. The process of serializatoin is crucial aspect of working with the database. In the context of web application serialization is commonly used to convert the data into JSON(Javascript object notation) or xml format.
7. **FLASK migration init:** Creates the migration repository
8. **FLASK db upgrade:** The flask db upgrade command applies the migration to the database. This means it runs the migration scripts that have not been applied to data with current state of the models
9. **make_response:** make_response in python is equal to the response_entity in java where we sent the object with the status code
10. **Unittest:** this is python build in library for writing and running the test. There are three methods in unitest library

a. **SetUp:** This method is called before each individual test method to set up a state that is shared among the states. This ensures that each test starts with a clean state
b. t**earDown method:** This method is called after each individual test method to clean up any state that should not persist between tests, This ensures that each test does not affet the other.
c. **Method that starts with "test__":** These are those method which is having name like "test_hello_world" means these method always starts with "test__" . These are the actual method that contain the assertion to check the correctness of the code.Each test method typically starts with 'test__' and is executed by the unitest framework.
d. Execution Flow:

    For each test method:

    1. Call the SetUp method
    2. Run the test method
    3. Call the tearDown method

12. **Refresh Tokens:** Refresh tokens are those tokens which is used to create the new access token, if the current access token is expired
13. **CORS (Cross Origin Resourse Sharing):** If we need aur api should work with client/Frontend we have to set up the CORS policy of our backend. Because our browsers does not allow request from the one domain (i,e localhost:3000) where react is running, so to communicate with the other domain i,e(localhost: 5000) our backend is running. In informal way we have to say that to aur backend that the application running at localhost:3000 i,e our react application is our friend. It is not harmful for us
14. **Proxy in package.json:** When we set the " proxy::http://localhost:5000 in our package.json of react app we does not have to specify the url everytime we call the api for example to call the api like "http:localhost:5000/recipes: we just call "/recipes" and its done. **It also helps in CORS policy.**
15. ReactDom.render() method is deprecated that why we use :

    const root = ReactDon.CreateRoot(document.getElementById("root")

root.render(<App />)

16. **React Router Dom:**

    BrowserRouter is used so that our application should use the routing of application

    Switch is used to exactly match the url of the page

    Note: In react version 6 the switch is replaced by routes

17. **Fetch Request of POST Types:**

```
fetch("/auth/signup", {
method: "POST",
Headers:{
"Content-Type": "application/json"
},
Body: JSON.strinify({username, email, password})
})
.then(res => res.json())
.then(data => Console.log(data))
```

18. **React-Bootstrap:**

    We have used the react-bootstrap library for this project. Because react-bootstrap has better react compability than only bootstrap.

    The components provided by react-bootstrap are more naturally integrated into a react-project, which allows you to use them as a react components with props rather than relying on bootstrap HTML and javaScript

    It can be more efficient in Bundle Size because it can be use for the tree shaking

19. **Tree Shaking & Bundle Size in react :**

a.

Tree shaking is a technique used in javaScript bundler(like WebPack) to eliminate the dead or unused code from the final bundle.

The term tree shaking comes form the idea of shaking a tree to remove the dead leaves. In this context it means removing code that is never used in your application

How it Works :

a. When you import a module it might contain many functions or the components, However you may only use the small portion of them
b. Tree Shaking analyses the code to determine which parts of a module are actually being used and removes the rest during the build process
c. The result in a smaller and more efficiend bundle becaus unneccessary code in not included.

   Tree shaking helps reduce the size of the javascript files that are sent to the client which in turn improves the load time and overall performance.

b**. Bundle Size:**

Bundle Size is the size of the javascript file that the bundler like webpack, Rollup etc generates for your react application. When we use create-react-app in the default bundler webpack create the new react app under the hood.

20. **Use Navigate hook in react** :

    The useNavigate hook is a new addition to the react router 6. It's a replacement for the useHistory and useLocation hooks in previous versions of react router.

    The useNavigator hook provides the simple and intuitive API for navigating between pages in your react application

    To use useNavigate we have to first import it like import {useNavigate } from 'react-router-dom' and consider the example :

```
Function myComponent = () =>{
        const navigate = useNavigate();
        const handleClick = () =>{
                navigate("/other-pages")
        }
        return( <button onClick = {handleClick}> Go to the other page
</button>);
    }
```

In this example we're using the useNavigate hook to create navigate function that we can use to navigate to a different page.
We then use this function in the handleClick function to navigate to the 'other-page' URL when the button is clicked
In thi project we are using the useNavigate hook so that when some person successfully login then he should be redirected to the new page

21. **Keep in mind**

```
const [recipes,setRecipes ] = useState ([])
useEffect (() =>{
fetch("http::localhost:5000/recipe/recipes")
.then(res = > res.json())
.then(data =>{
        console.log(data )
        setRecipes(data)
        console.log(recipes))
})
}, [])
```

This code will not print the updated recipes done through the setRecipe in console

This is because of Asynchronous State update
The SetRecipe (data) call schedule an update to the recipes state but this update doesn't happen immediately.
React batches state updates and applies them in the next render cycle

Therefore logging recipes immediately after the calling 'setRecipes' will show the old value (which is still the empty array)

How to fix the above issue

If we want to see the updated code recipes in the console.log we should log it in the next render cycle,

i, e we can use the another useEffect hoot which will print the value of recipe when the current value of the recipe changes

I,e we are inserting the value of recipe into the dependency array of another useEffect hook as follows:

```
useEffect (() =>{
console.log(recipes)
},[recipes]);
```

22. In this project we have used the modal to get the pop up window when someone click on the update recipe page.

GlobalLogic®
A Hitachi Group Company

Flask + React Project.

pip install Flask Flask-restx Flask-sqlalchemy Flask-jwt-extended

Config.py :- This file will have all the configuration related to the project.

Libraries Using.

Flask :- for creating the web application.

Flask-RESTx - for building rest api with the documentation and validation.

Flask-SQLAlchemy :- for database interaction using SQLalchemy ORM

Flask-JWT-extended - for handling authentication via JSON Web tokens (JWT)

Authentication:

GlobalLogic®
A Hitachi Group Company

library is used to manage configuration setting for your python application.

It helps you to separate configuration parameter (like database setting, API keys and other sensitive information) from your source code, making code cleaner and secure.

set Flask_APP = main.py This command is used to set the flask app environment variable. It tells the flask which python file to run/use as the entry point of our application.

$env: Flask_APP = "main.py" This is equivalent command in Powershell for setting the environment variable to "main.py" after setting this we can use the flask run directly to run our application.

GlobalLogic®
A Hitachi Group Company

Q Serializer :-

The Serialization is the process of converting data into a format suitable for transmission and storage.

The process serialization is crucial aspect of working with database.

In the context of web applications, Serialization is commonly used to convert the data into JSON (Javascript Object Notation) or XML format.

flask migration init :- creates the migrations repository

flask db upgrade :- The flask db upgrade command applies the migrations to the database. This means it runs the migration scripts that have not yet been applied to bring the database schema up to data with current state of the models.

GlobalLogic®
A Hitachi Group Company

Unittest :- this is python built-in-library for writting and running test.

There are three methods in unitest library in the
① SetUp
② teardown
③ Method that starts with test.

① Setup method :- this method is called before each individual test method to set up a state that is shared among the states.
This ensures that each test starts with a clean state.

② tearDown method :- This method is called after each individual test method to clean up any state that should not persist between tests. This ensures that each test does not affect the others.

③ Method that starts with "test" :- There are those method which is having

name like the "test_hello_world" means these method always start with "test_"

These are the actual test method that contain assertions to check the correctness of the code. Each test method typically starts with 'test_' and is executed by the unittest framework.

Execution flow.

for each test method:-

① Call the setUp method.

② Run the test method

③ Call the tearDown method.

* Refresh tokens are those tokens which is used to create the new access token

GlobalLogic®
A Hitachi Group Company

CORS (Cross origin resource sharing)
If we need our api should work with client/frontend we have to set up the CORS policy of our backend.

Because our browsers does not allow the request from one domain i,e (localhost:3000) where react is running to communicate with other domain i,e (localhost:5000) where our backend is running.

So in informal way we have to say that to our backend that the application running at localhost:3000 i,e our react application is our friend. It will not harmful to us.

* Proxy in package.json.

When we set the "proxy": "http://localhost:5000" in our package.json of react app we does not have to specify the

GlobalLogic®
A Hitachi Group Company

api for example to call the api like "http://localhost:5000/recipes" we just call "/recipes" and its done. It also helps in the CORS policy

* ReactDOM.render() method is deprecated thatwhy we use :-

```
const root = ReactDOM.createRoot(document
    .getElementById('root');

root.render(<App/>).
```

* React Router DOM

BrowserRouter is used so that our application should use the routing of application.

Switch is used exactly match the URL of the page.

Note:- In react version 6 the

GlobalLogic
A Hitachi Group Company

Fetch API Requests for different types.

POST Method :-

```
fetch("/auth/signup", {
    method: "POST",
    headers: {
        "Content-Type": "application/json"
    },
    body: JSON.stringify({username, email, password})
})
.then(res => res.json())
.then(data => Console.log(data))
```

GlobalLogic®
A Hitachi Group Company

* React-Bootstrap.

We have used the react-bootstrap library in this project. Since because react-bootstrap has better react compability than only bootstrap.

the components provided by react-bootstrap are more naturally integrated into a react-project, which allows you to use them as a react components with props, rather than relying on bootstraps HTML & javascript.

It can be more efficient in Bundle size because it can be used for tree shaking

* Tree Shaking & Bundle Size in React

Tree shaking is a technique used in javascript bundler (like webpack) to eliminate dead or unused code from the final bundle.

The term tree shaking comes from the

the dead ~~ti~~ leaves. In this context, it means ~~re~~ removing code that is never used in your application.

How it works :-

(i) When you import a module it might contain many functions or the components. However you may only use the small portion of them

(ii) Tree Shaking analyses the code to determine which parts of a module are actually being used and removes the rest during the build process

(iii) This result in a smaller and more efficient bundle because unnecessary code is not included.

Tree Shaking helps reduce the size of the javascript files that are sent to the client, which in turn improves load times and overall performance.

GlobalLogic
A Hitachi Group Company

Bundle Size :-

Bundle size is the size of the javascript file(s) that the bundler like (webpack, Rollup, etc) generates for your react application.

When we use create-react-app the default bundler webpack create the new react app under the hood.

UseNavigate hook in react.

The useNavigate hook is a new addition to react Router 6. It's a replacement for the useHistory and useLocation hooks in previous version of react router.

The useNavigate hook provides a simple and intuitive API for navigating between pages in your react application.

and consider the example.

```
function myComponent() {

    const navigate = useNavigate();
    const handleClick = () => {
        navigate('/other-page');
    };
return (
<button onClick={handleClick}>Go to
    other page.</button>
);
}
```

In this example, we're using the useNavigate hook to create a navigate function that we can

GlobalLogic®
A Hitachi Group Company

use to navigate to a different page.

We then use this function in the handleClick function to navigate to the '/other-page' URL when the button is clicked.

In this project we are using the useNavigate hook so that when some person successfully login then he should be redirected to the new page

GlobalLogic
A Hitachi Group Company

* Keep in mind

- ~~UseEffect~~

```
const [recipes, setRecipes] = useState([])
useEffect(() => {
    fetch('http://localhost:5000/recipe/recipes')
    .then(res => res.json())
    .then(data => {
        console.log(data)
        setRecipes(data)
        console.log(recipes)
    })
}, [])
```

GlobalLogic®
A Hitachi Group Company

this code will not print the updated recipes done through the setRecipe in console.

this is because of Asynchronous state update

The SetRecipes(data) call schedules an update to the recipes state but this update doesn't happen immediately.

React batches state updates and applies them in the next render cycle.

Therefore, logging recipes immediately after calling 'setRecipes' will show the old value (which is still the initial empty array)

How to fix above issue

If we want to see the updated

rendee cycle.

i,e we can use another useEffect hook which will print the value of recipe when the current value of the recipe changes.

i,e we are inserting the value of recipe into the dependency array of another useEffect hook as follows:-

```
useEffect(() => {
    console.log(recipes)
}, [recipes]);
```

In this project we have used the modal to get the pop up window when someone click on the update recipe tab.