TRAINOSYS

REACTJS & TYPESCRIPT

Day 4

HYBRID
TECHNOLOGY
COURSES

Course Overview

- Introduction to ReactJS and TypeScript (Day 1)
- React Component Development (Day 2)
- React Routing and Data Fetching (Day 3)
- State Management with Redux and TypeScript (Day 4)
- Advanced TypeScript and Project Development (Day 5)

Program

Day 03

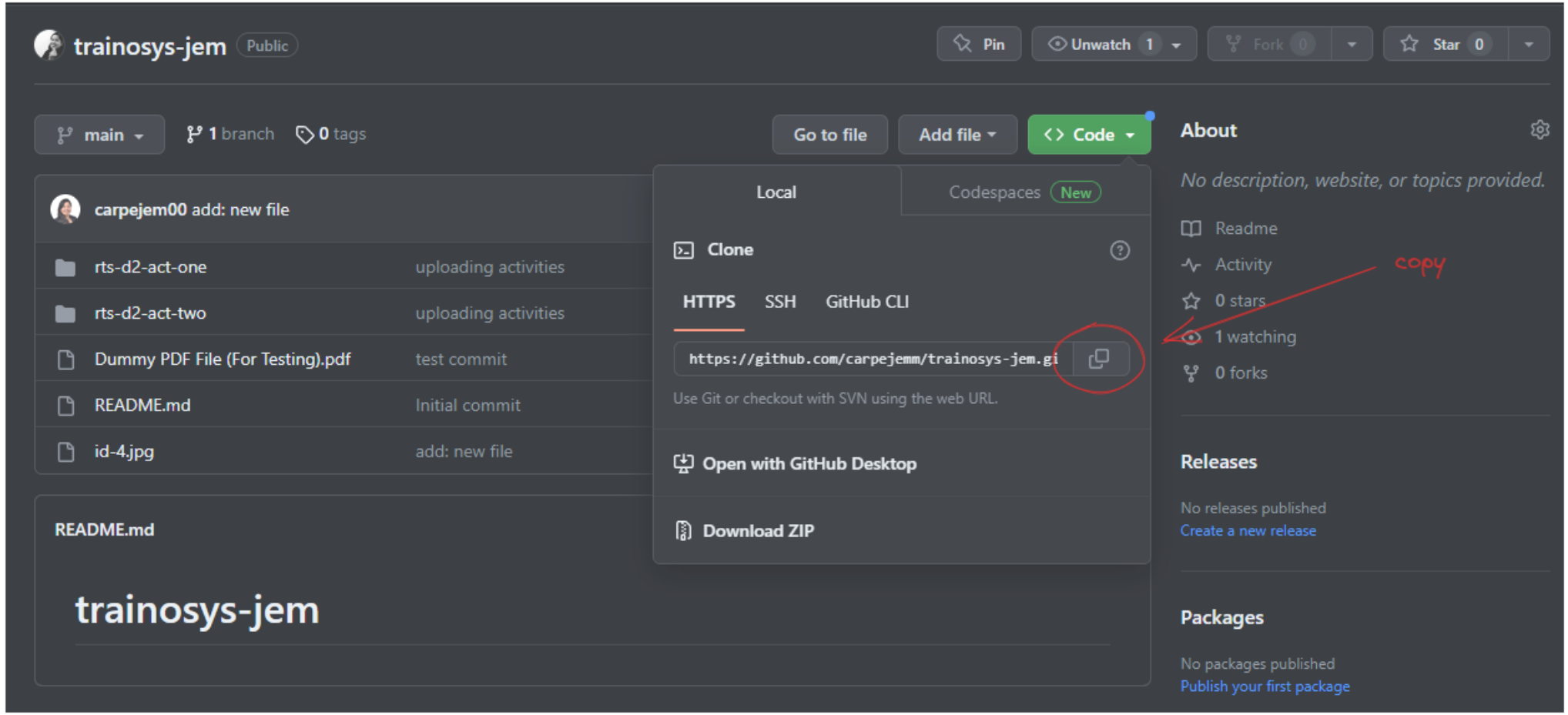
- Introduction to React Router for handling client-side routing
- Configuring routes in a TypeScript-based React Application
- Navigating between different routes with React Router
- Fetching data from APIs using TypeScript and React
- Displaying data fetched from an API in React components

Day 04

- Introduction to state management with Redux
- Setting up Redux in a TypeScript-based React application
- Creating actions and reducers with TypeScript
- Managing global state with Redux in TypeScript
- Connecting React components to Redux store using TypeScript

RECAP

GITHUB


trainosys-jem
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 branch
0 tags
Go to file
Add file
Code
carpejem00 add: new file
rts-d2-act-one
uploading activities
rts-d2-act-two
uploading activities
Dummy PDF File (For Testing).pdf
test commit
README.md
Initial commit
id-4.jpg
add: new file
README.md
trainosys-jem
Local
Codespaces New
Clone
HTTPS
SSH
GitHub CLI
https://github.com/carpejemm/trainosys-jem.gi
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
copy
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

Download Resources

- > git clone https://github.com/carpejemm/trainosys-jem.git

- To download latest files
- > git pull

Windows PowerShell

```
PS C:\Users\admin\Desktop> git clone https://github.com/carpejemm/trainosys-jem.git
Cloning into 'trainosys-jem'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (38/38), done.
Receiving objects:  46% (24/52)sed 48 (delta 7), pack-reused 0
Receiving objects: 100% (52/52), 121.64 KiB | 1.32 MiB/s, done.
Resolving deltas: 100% (8/8), done.
PS C:\Users\admin\Desktop> cd .\trainosys-jem\
PS C:\Users\admin\Desktop\trainosys-jem> ls


    Directory: C:\Users\admin\Desktop\trainosys-jem


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/27/2023  10:51 PM                rts-d2-act-one
d-----         7/27/2023  10:51 PM                rts-d2-act-two
-a----         7/27/2023  10:51 PM          13264 Dummy PDF File (For Testing).pdf
-a----         7/27/2023  10:51 PM          45267 id-4.jpg
-a----         7/27/2023  10:51 PM             15 README.md


PS C:\Users\admin\Desktop\trainosys-jem> git branch
* main
PS C:\Users\admin\Desktop\trainosys-jem> git pull
Already up to date.
PS C:\Users\admin\Desktop\trainosys-jem>
```

Upload Activities

- ❑ Create new folder and put all your activities inside
- ❑ Create a github account
- ❑ > git init
- ❑ > git remote set-url origin <http-url>
- ❑ > git remote -v
- ❑ > git status
- ❑ > git add .
- ❑ > git commit –m “<commit message here>”
- ❑ > git push –set-upstream origin <master/main>

TRAINOSYS

REDUX

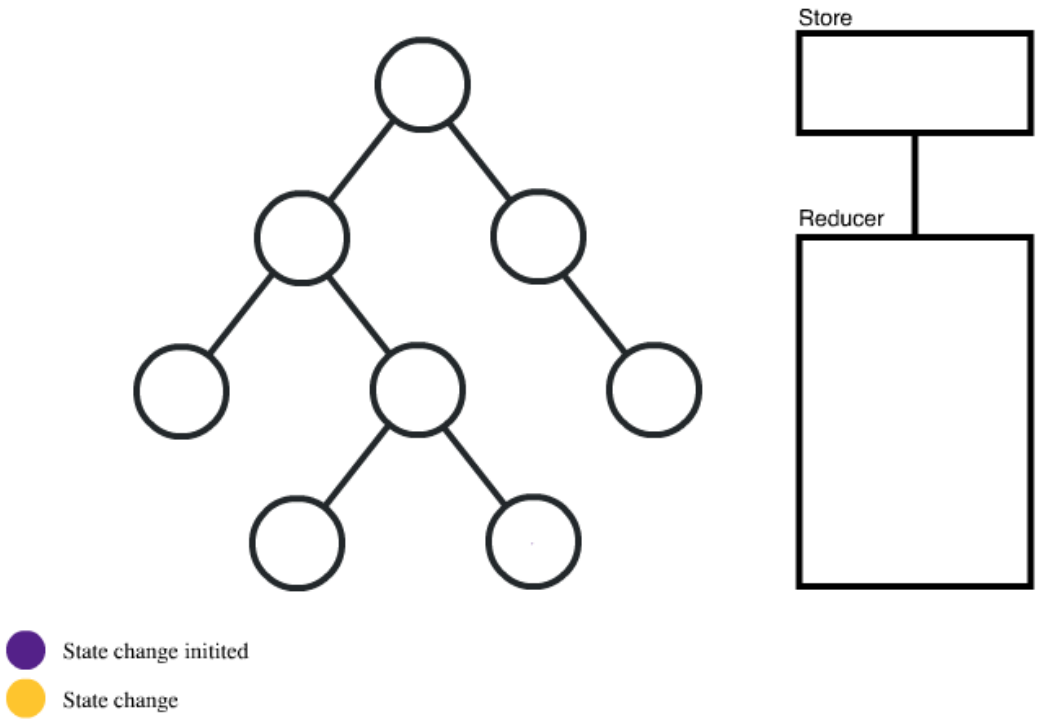
HYBRID
TECHNOLOGY
COURSES

What is Redux?

- Redux is a tool for managing both data-state and UI-state in JavaScript applications
- It's ideal for Single Page Applications (SPAs) where managing state over time can be complex
- It's also framework-agnostic, so while it was written with React in mind, it can even be used with Angular or a jQuery applications

Redux - Unidirectional Flow

- React “data flow ”is called “unidirectional data flow” — data flows in one direction from parent to child
- With this characteristic, it’s not obvious how two non parent-child components would communicate in React



Store - single source of truth

- Redux offers a solution of storing all your application state in one place, called a “store”.
- Components then “dispatch” state changes to the store, not directly to other components. The components that need to be aware of state changes can “subscribe” to the store

TRAINOSYS

CODE ALONG

Let’s convert our To Do App to Redux!

HYBRID
TECHNOLOGY
COURSES

Generate a new project

npm create vite@latest / npm create vite@4.1.0

Generate a new project

- ❑ Project Name: to-do-app-redux
- ❑ > React
- ❑ > TypeScript
- ❑ > cd to-do-app-redux

Install packages

```
npm install react-redux @reduxjs/toolkit
```

Install react-redux and redux toolkit

Generate a new project

- > npm run dev

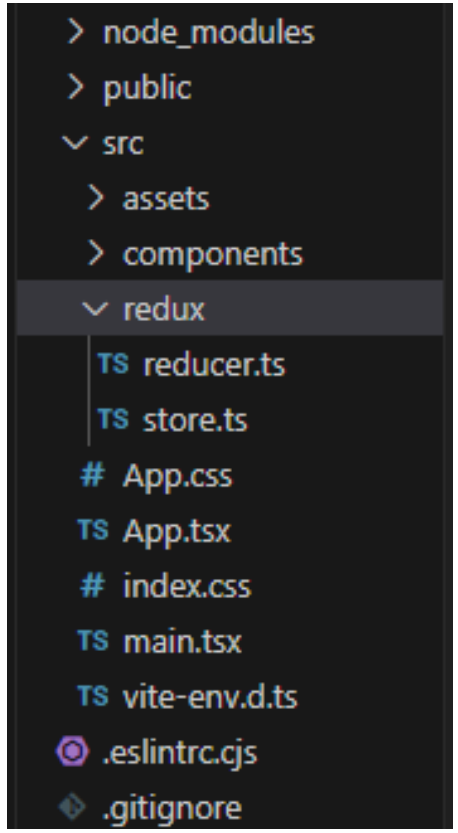
In your *main.tsx*, import the Provider from react-redux and your Redux store.

TRAINOSYS

```
TS main.tsx ×
src > TS main.tsx
1   import React from 'react'
2   import ReactDOM from 'react-dom/client'
3   import { Provider } from 'react-redux';
4   import store from './redux/store';
5   import App from './App.tsx'
6   import './index.css'
7
8   ReactDOM.createRoot(document.getElementById('root')!).render(
9     <React.StrictMode>
10      {/* Wrap the App component with the Provider */}
11      <Provider store={store}>
12        <App />
13      </Provider>
14    </React.StrictMode>,
15  )
```

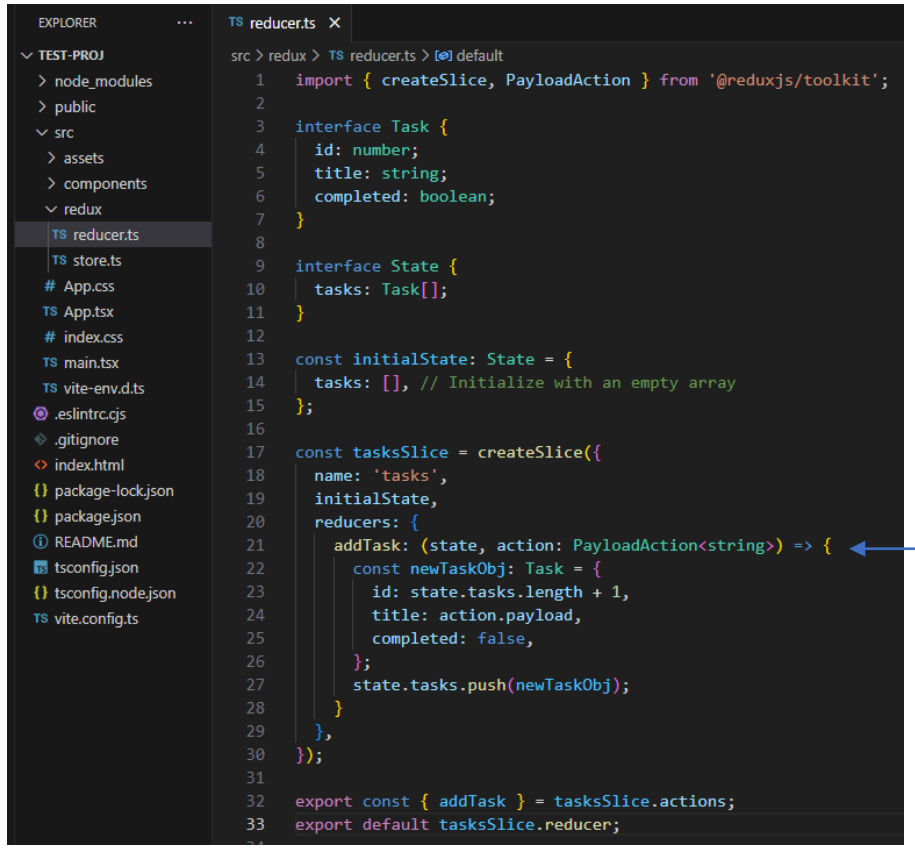
Wrap your App component with the Provider, providing the Redux store as a prop

Let’s create the Redux Store



Inside your *src* directory, create a new directory called *redux*.
Inside the *redux* directory, create two files: ***reducer.ts*** and ***store.ts***.

Define the Initial State and Reducer



Define the initial state of your application

Let's try our *addTask* function here

Create a reducer using Redux Toolkit's *createSlice* function

Export our reducers and actions

In *store.ts*, import the configureStore function from Redux Toolkit

In store.ts, the configureStore function from Redux Toolkit implicitly combines reducers to manage multiple state slices in the Redux store.

TRAINOSYS

TS store.ts | TS App.tsx

src > TS App.tsx > Task > id

```
1  import React from 'react';
2  import { useSelector, useDispatch } from 'react-redux';
3  import { RootState } from './redux/store';
4  import { addTask } from './redux/reducer';
5  import Task from './components/Task/Task';
6  import AddTask from './components/AddTask/AddTask';
7  import './App.css';
8
9  interface Task {
10   id: number;
11   title: string;
12   completed: boolean;
13 }
14
15 const App: React.FC = () => {
16   const tasks: Task[] = useSelector((state: RootState) => state.tasks.tasks);
17   const dispatch = useDispatch();
18
19   return (
20     <div style={{ textAlign: 'center', margin: '50px'}}>
21       <h1>Task Manager</h1>
22       <AddTask onAddTask={(task) => task.trim() !== '' && dispatch(addTask(task))} />
```

In your *App.tsx*, update your component to use Redux Toolkit's *useSelector* and *useDispatch* hooks to access the state and dispatch actions respectively

Add our imports from *store* and *reducer*

This is the selector function. It receives the entire Redux state (*state*) as its parameter, and it returns the specific part of the state that we want to extract.

Dispatches an action to add the new task to the Redux store using the addTask action creator. The dispatch function is provided by Redux and used to dispatch actions to the store's reducer.

TS store.ts | TS AddTask.tsx | TS App.tsx

src > components > AddTask > TS AddTask.tsx > AddTask

```typescript
import React, { useState, ChangeEvent } from 'react';
import { useDispatch } from 'react-redux';
import { addTask } from '../../redux/reducer';
import './add-task.css';

interface AddTaskProps {
  onAddTask: (task: string) => void;
}

const AddTask: React.FC<AddTaskProps> = () => {
  const [newTask, setNewTask] = useState('');
  const dispatch = useDispatch();

  const handleInputChange = (event: ChangeEvent<HTMLInputElement>) => {
    setNewTask(event.target.value);
  };

  const handleAddTask = () => {
    if (newTask.trim() !== '') {
      dispatch(addTask(newTask));
      setNewTask('');
    }
  };

  return (
    <div className="add-task-container">
      <input
        type="text"
        value={newTask}
        onChange={handleInputChange}
        className="add-task-input"
        placeholder="Enter a new task"
      />
      <button onClick={handleAddTask} className="add-task-button">
        Add Task
      </button>
    </div>
  );
};

export default AddTask;
```

Local state

initializes the dispatch

the dispatch function is used to dispatch the *addTask* action with the new task as its payload when the "Add Task" button is clicked. This action is then processed by the Redux store's reducer, updating the state and adding the new task to the task list.

ACTIVITY 1

Activity 1

- Apply the React Redux in your doneTask, editTask and deleteTask function (To Do App Activity)

ACTIVITY – Kahoot!

TRAINOSYS

CODE ALONG – TO DO APP REDUX ACTIVITY

HYBRID
TECHNOLOGY
COURSES

TRAINOSYS

CODE ALONG – DAY 3 ACTIVITY ANSWERS

HYBRID
TECHNOLOGY
COURSES

Generate a new project

npm create vite@latest / npm create vite@4.1.0

Generate a new project

- ❑ Project Name: rts-d3-act-one
- ❑ > React
- ❑ > TypeScript
- ❑ > cd rts-d3-act-one
- ❑ > npm install / npm i
- ❑ > npm run dev

```
58  return (
59    <>
60      <Routes>
61        <Route path="/" element={<RootPage />}>
62          <Route path="/all" element={<PokemonList pokemonList={pokemonList} />} />
63          <Route path="/home" element={<HomePage />} />
64          <Route path="/about" element={<AboutPage />} />
65          <Route path="/profile" element={<ProfilePage />} />
66
67          {/* Route for the individual Pokemon's page */}
68          <Route path="/pokemon/:name" element={<PokemonComponent pokemonList={pokemonList} />} />
69        </Route>
70      </Routes>
71
72    </>
73  )
```

EXPLORER

POKEDEX-REACT-ROUTER
- node_modules
- public
- src
  - assets
  - components
    - About
    - Home
    - Pokemon
      - Pokemon.tsx
    - PokemonList
      - PokemonList.tsx
      - style.module.css
    - Profile
    - RootPage
  - App.css
  - App.tsx
  - index.css
  - main.tsx
  - vite-env.d.ts
- .eslintrc.cjs
- .gitignore
- index.html
- package-lock.json
- package.json
- README.md
- tsconfig.json
- tsconfig.node.json
- vite.config.ts

TS App.tsx | TS Pokemon.tsx

src > components > Pokemon > TS Pokemon.tsx > Pokemon

```typescript
import React from 'react';
import { useParams } from 'react-router-dom';

interface PokemonProps {
  pokemonList: Pokemon[];
}

interface Pokemon {
  name: string;
  height: number;
  id: number;
  img: string;
  types: string[];
}

const Pokemon: React.FC<PokemonProps> = ({ pokemonList }) => {
  const { name } = useParams<{ name: string }>();

  const selectedPokemon = pokemonList.find((pokemon) => pokemon.name === name);

  if (!selectedPokemon) {
    return <div>Pokemon not found!</div>;
  }

  return (
    <div>
      <h3>{selectedPokemon.name}</h3>
      <h4>{selectedPokemon.id}</h4>
      <img src={selectedPokemon.img} alt={selectedPokemon.name} />
      <ul>
        {selectedPokemon.types.map((type) => (
          <li key={type}>{type}</li>
        ))}
      </ul>
    </div>
  );
};

export default Pokemon;
```

TS App.tsx | TS RootPage.tsx

src > components > RootPage > TS RootPage.tsx > RootPage

```typescript
import { Link, Outlet } from 'react-router-dom';
import './rootpage.css';

const RootPage: React.FC = () => {
  return (
    <header className='m-container'>
      <h2 className='title'>
        <Link to="/">Pokedex</Link></h2>
      <div className='button-container'>
        <p className='load'>
          <Link to="/all">All Pokemons</Link>
        </p>
        <p className='filter'>
          <Link to="/home">Home</Link>
        </p>
        <p className='load'>
          <Link to="/about">About</Link>
        </p>
        <p className='filter'>
          <Link to="/profile">Profile</Link>
        </p>
      </div>
      <main>
        <Outlet />
      </main>
    </header>
  );
};

export default RootPage;
```

TRAINOSYS

CODE ALONG – LET’S MAKE A REACT PORTFOLIO PROJECT

HYBRID
TECHNOLOGY
COURSES

Generate a new project

- ❑ Project Name: react-portfolio-project
- ❑ > React
- ❑ > TypeScript
- ❑ > cd react-portfolio-project
- ❑ > npm install / npm i
- ❑ > npm run dev

Generate a new project

- ❑ Create Login and Registration Page
- ❑ > apply react-router, react hooks etc.
- ❑ Upon login / registration – create a Home / Landing page
- ❑ > https://www.behance.net/gallery/139185709/Job-Interview-Pre-Hire-Assessment-App/modules/786638349
- ❑ Upper right corner – create a user profile page

TRAINOSYS
Training the Future Today

Reach Us!

Visit Us

12th/F The Trade & Financial Tower Unit 1206
32nd Street & 7th Avenue Bonifacio Global City,
Taguig 1634 Philippines

Email Us

inquiry@trainosys.com

Browse Our Website

www.trainosys.com

TRAINOSYS

Training the Future Today