Incture Technologies is a digital solutions company that helps businesses become more efficient by using technology and automation. It was started in 2006. then, Incture expanded its services to include cloud-based development, AI-powered automation, and data analytics. Their main product, Cherrywork®, is a platform that helps companies automate workflows and improve productivity.

**Week 1: Training Kick-off & Introduction to HTML, CSS, JS**

- In week 1 they gave told us about the training and quick introduction about react

- I regained the knowledge about html css and js like how to create semantic code and effecient and maintable code\*,

- I used CSS for styling the pages, to make the design responsive so it looks good on all devices.

- I worked with JavaScript functions to add functionality or interactivity, such as handling DOM interactions (like changing content dynamically or responding to user actions).

• Built a website and calcultor app with HTML, CSS, and JavaScript, implementing form validation and interactive UI

Week2: Introduction to React, Components, State & Props, Events

here we learnt about how to set up a React project and understand **JSX, (which is a syntax extension that lets us write HTML-like code inside JavaScript. )**

also about functional and class-based components to build reusable and modular UI elements.

- also managed component state using the useState hook, which allowed me to track and update data dynamically

- and about events **(like onclicks,onsubmit events)** and updated the UI in real time based on user interactions.

• Built a counter app in React that used the useState hook to manage the counter

value

Week 3: Conditional Rendering, React Life cycle Methods

- here we learnt about conditional rendering techniques using ternary and logical operators**, which allowed me to dynamically display content based on specific conditions.**

- I efficiently rendered dynamic lists using the map() function along with unique keys**, to prevented rendering issues.**

- I learned about the component lifecycle phases:

**- Mounting (when the component is first added to the DOM),**

**- Updating (when the component re-renders due to state or prop changes), and**

**- Unmounting (when the component is removed from the DOM).**

- I implemented the useEffect hook to **handle side effects like data fetching or DOM manipulation**. I also created custom hooks for reusable logic, which made my code more modular and maintainable.

..........Developed a **functional component to display a list of names**, later extending it to

render **user details such as ID, name, and age** in a table format

Week 4: React Context, React Router, Forms, and Validation

- we learnt the React Context API for **manging state globally which helped me avoid \*\*prop drilling.** This made it easier to \*\***pass data across multiple components without having to manually pass to every level.**

learnt React Router to create a single-page application (SPA) with seamless **navigation between different pages without full reloads.**

controlled and uncontrolled form components, learning how to **manage form inputs efficiently**. (Controlled components had their **values managed by \*\*React state,** while uncontrolled components **relied on DOM references**. )

- I also implemented form validation to ensure that user inputs met specific requirements, providing error handling and feedback when incorrect data was submitted.

• Developed a Task Management Application using React.js, focusing on implementing Context API for state management.

Week 5: State Management with Redux, Error Handling, Testing React Applications

learnt about Redux. used actions and reducers to define how the state should be modified.(**centralized store to manage the application's state, making it easier to understand, maintain, and debug.)**

-For leant about **error handling, using React’s componentDidCatch()** and **getDerivedStateFromError() lifecycle methods**.

- also **wrote unit testing using Jest to check the code**

• Developed **a React-Redux application that** **fetched data from an external API** using Redux Thunk middleware

Week 6: Performance Optimization, TypeScript– Hooks, Form Handling, API

- I learnt about React.memo, useCallback, and useMemo to optimize rendering. These hooks helped me \*\*prevent unnecessary re-renders, improving the overall performance of the app. For example, by wrapping components with **React.memo, I ensured they only re-rendered when their props changed**. ---------------**useCallback is a React hook that returns a memoized version of the callback function** that only changes if one of the dependencies has changed

- I implemented **lazy loading using React’s lazy() and Suspense by loading components only when they were needed, reducing the initial bundle size.**

- I learnt about **TypeScript with React**. This helped me **define strict types for component props, states, and API responses**, making the code more robust and easier to maintain.

• Worked on performance optimization techniques such as memoization, lazy loading, and React Profiler

Week 7 – Git, Version Control & Deployment

Week 7 was all about getting about Git and deploying our websites.

\* In this week, we first learnt about basic Git commands like git init, git clone, git add, git commit, and git push to push our code.

\* We also saw how to create pull requests and manage code

\* Then we created Git repositories for our projects and started building them.

\* After building, we first deployed the dist folder (the build version) to Netlify just to check if it's working fine.

\* Later, we deployed the whole project to Netlify to make sure everything like routing and project structure is working properly.

🗓 Week 8 & 9 – Amazon Clone Assignment

Week 8 and 9 were all about the major project. Everyone got different topics, and I got the Amazon Clone.

First, I created components like the navbar and banner.

Then I used **props and state properly to manage data between components.**

I added components like product cards for each item and features like register, login, add to cart, and checkout.

I used useEffect for handling side effects like fetching/storing cart data and some DOM updates.

For styling, I used CSS and Flexbox to match the look and feel of the original Amazon layout.

Finally, I pushed the code to GitHub and deployed it on Netlify.

week 10 and 11 final asesment

they gave 1 week time to prepare for the assesment later we had 3-hour online assessment via Evalgator platform.

which Included MCQs about core concepts covered in internship.and also Coding challenges, including web-based

tasks(like creating a responsive website) and multiple React/JavaScript problems.

then we Participated in peer review to evaluate each others projects and get feedback from others

then the Final was scores released

🗓 Week 12 – MVC Architecture, JSON, SAP BTP Basics

Week 12 was mainly about SAP basics and MVC architecture.

**SAPUI5 is a JavaScript framework and UI library developed by SAP for building cross-platform, enterprise-grade web applications**

First, we learnt about the MVC pattern, **which helps in separating the business logic, UI, and data management. This makes our app more organized and easy to manage**.

Then we studied SAP UI5, which is a JavaScript framework to build responsive enterprise apps. It’s mainly used in **SAP environments to develop user interfaces that follow the Fiori design principles.**

We also covered JSON — **we learnt its syntax and how it’s used in real-world APIs.** We worked with fetch and axios to get and parse JSON data.

* **Explored SAP BTP is a cloud platform for building and running enterprise applications**. It provides tools and services for Data management, App development, Integration with SAP systems

In web apps, we often **get data from APIs** in JSON format.

fetch() and axios are JavaScript tools for making **HTTP requests**.

Week 13 – SAP Cloud Foundry, Cloud Connectors, Business Application Studio, UI5

 **Cloud Foundry** is a platform-as-a-service (PaaS) that hosts and manages applications in the **SAP BTP (Business Technology Platform)**.

 You deployed your UI5 app using **BAS**, which runs on **Cloud Foundry**.

SAP BAS (Business Application Studio) is a cloud-based development environment for building and extending applications on the SAP Business Technology Platform (BTP)

**Cloud-based IDE:**

**Dev Spaces:**

It utilizes "dev spaces," which are virtual environments tailored for specific development scenarios, including SAP Fiori, SAP S/4HANA extensions, and more

Explored UI5 framework and its reusable components

Practiced using the SAP UI5 Demokit for interface development

Week 14 :

In SAPUI5, an ODataModel is a model that facilitates the interaction with an OData service (a REST API) on the server. It allows you to perform CRUD (Create, Read, Update, Delete) operations on data by sending requests to the OData service and managing the data returned.

ou added **form elements** like sap.m.Input, sap.m.Button, and attached **event handlers** such as:

* onAddProduct()
* onDeleteProduct()
* onEditProduct(

sap.m.BusyDialog refers to a UI control that displays a dialog box indicating that the application is currently busy and processing data, effectively blocking user interaction with the UI. This dialog provides feedback to the user about ongoing operations and helps prevent them from attempting other actions

**Week 15: SAP UI5 – Routing, Navigation & Fragments**

* Set up navigation between multiple views using routing in manifest.json.
* Created reusable UI parts like dialog boxes and forms using **fragments**.
* Used navTo and onNavBack to move between pages in the app.
* Made the code cleaner and the app easier to use by organizing routes and reusing components.