Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2025-26

Class:	TE-AIDS	Semester:	V
Course Code:	CSC502	Course Name:	WC

Name of Student:	Divya Davane
Roll No. :	14
Assignment No.:	6
Title of Assignment:	React development using back-end.
Date of Submission:	
Date of Correction:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Completeness	5	5
Demonstrated Knowledge	3	3
Legibility	2	2
Total	10	10

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Completeness	5	3-4	1-2
Demonstrated Knowledge Legibility	3	2	1
Legibility	2	1	0

Checked by

Name of Faculty :

Signature : Bharat

Date : 6/10/25

WC assignment - 6

NAME:________ STD.:______ DIV.:______ DATE : PAGE :

You are building a React functional component that needs to fetch data from an external API and display it. Use the useEffect hook to perform this side effect. Write a React component that fetches a list users from https://jsonplaceholder.typicode.com/users and displays their names in a list. Explain how useEffect manages the side Effect in your implementation.

Ans

```
import React, { useState , useEffect } from "react";
function UserList () {
const [users, setUsers] = useState ([]);
const [loading, setloading] = useState (true);
useEffect (() => {
fetch ("https://jsonplaceholder.typicode.com/users")
.then ((response) => response.json())
.then ((data ) => {
setUsers (data);
setloading (false);
})
.catch ((error) => {
console.error("Error fetching users:", error);
setLoading (false);
});
}, []);
if (loading) {
return <p> Loading users..... </p>
}
return (
 <div>
 <h2> UserList </h2>
```

NAME: STD.: DIV.: DATE : PAGE :

```
<ul>
{users.map((user) => (
<li key = {user.id}>{user.name}</li>
))}
</ul>
</div>
);
}
export default UserList;
```

Q.2. You are developing a simple web application to manage a library of books. Using the MVC architecture, implement the functionality to add a new book and display the list of books. Describe how you seperate the responsibilities between the Model view, and controller in your implementation.

Ans • Model (Book.js)

```
class Book {
constructor(title, author) {
this.title = title;
this.author = author;
}
}
class Library {
constructor () {
this.books = [];
}
addBook (book) {
this.books.push(book);
}
```

WORLD STAR

For Educational Use

NAME:_______________ STD.:________ DIV.:__________

DATE :

PAGE :

```
getBooks() {
  return this.books;
  }
  }

module.exports = { Book, Library };
```

• Controller (libraryController.js)

```
const { Book, Library } = require ('./Book');
const library = new Library ();
function addBookController (title, author) {
const book = new Book(title, author);
library.addBook(book);
}
function getBooksController() {
return library.getBooks();
}
module.exports = { addBookController; getBooksController };
```

• View (index.html)

```
<!DOCTYPE html>
<html>
 <body>
 <h2> Library </h2>
 <input id = "title" placeholder="Title">
<input id ="author" placeholder=" Author">
<button onclick="addBook()"> Add Book </button >
<ul id = "bookList"></ul>
```

NAME:________________ STD.:________ DIV.:________ DATE : PAGE :

```
<script>
async function addBook() {
const title = document.getElementById('title').value;
const ~~title~~ author = document.getElementById('author').value;
await fetch('/addBook', {
method: 'POST',
headers: {'content-Type': 'application/json'},
body: JSON, stringify({ title, author })
});
displayBooks();
}

async function displayBooks() {
const res = await fetch('/books');
const books = await res.json();
const list = document.getElementById('bookList');
list.innerHTML = '';
books.forEach(b => list.innerHTML += `<li>${b.title}
by ${b.author}</li>`);
}
displayBooks();
</script>
</body>
</html>
```

NAME: STD.: DIV.: DATE : PAGE :

.3 A Differentiate between MVC, FLUX and Redux

Feature	MVC (Model-View controller)	Flux	Redux
Architecture	classic pattern with three components: Model, View, controller	Unidirectional data flow pattern introduced by facebook for React.	Predictable state container based on Flux, with stricter rules
Data flow	Bidirectional	Unidirectional	Unidirectional
State Management	Each Model holds its own state	Stores hold state; Dispatcher coordinates updates	Single immutable global state
Scalability	Hard for large apps	Better, but dispatcher adds complexity.	Very scalable and predictable
Debugging	Harder due to multiple updates	Easier with unidirectional flow	Very easy, supports time-travel debugging
Usage	General apps, not React specific	Common in React	Mostly with React, can be used elsewhere

DATE :

PAGE :

NAME: STD.: DIV.:

Q.4. You are building a React application that manages shopping cart with multiple products. Use Redux to ma the state of the cart, including adding, removing, and updating items. Implement the Redux store, actions & reducers and explain how Redux helps in managing state across the application

Ans 1. Actions (cartActions.js)

```
export const ADD_ITEM="ADD_ITEM";
export const REMOVE_ITEM="REMOVE_ITEM";
export const UPDATE_ITEM="UPDATE_ITEM";
export const addItem = (item)=>({type: ADD_ITEM, payload:
item });
export const removeItem=(id)=>({type: REMOVE_ITEM, payloa
: id });
export const updateItem=(item)=>({type: UPDATE_ITEM, payload:
item });
```

2. Reducer (cartReducer.js)

```
import {ADD_ITEM, REMOVE_ITEM, UPDATE_ITEM} from "./
cart Actions";
const initialstate ={cart: [] };
export const cartReducer =(state = initialstate, action) => {
 switch (action.type) {
 case ADD_ITEM:
   return {...state, cart: [...state.cart, action.payload]};
 case REMOVE_ITEM;
   return {...state, cart: [...start.cart.filter(i=>i.id !==
 action.payload) };
 case UPDATE_ITEM:
 return {
   ...state,
```

NAME:______________ STD.:________ DIV.:________ DATE : PAGE :

```
cart: state.cart.map(i=> i.id ===action.payload.id ? action.
payload : i)
};
default: return state;
}
};
```

Q.5 Given a complex web application, propose a strategy using advanced React features like Ref and Hooks to manage component states across multiple layers. Explain the benifits of this approach.

Ans

- Strategy Using advanced react features :
  i) Use useState & useReducer for local and complex state management within components
  ii) Use useContext to provide global state to deeply nested components without prop drilling
  iii) Use useRef to persist values across renders (eg form inputs timers) without triggering re-renders.
  iv) Use custom Hooks to encapsulate reusable logic across multiple components
  v) Combine with useEffect for side effects like API calls or syncing state with external sources.

- Benefits :

NAME:________ STD.:______ DIV.:______ DATE : PAGE :

i) centralized and predictable state management.
ii) Reduces prop drilling and improves code readability
iii) efficient re-renders by isolating updates to affected components.
iv) Reusable logic via custom Hooks
v) Persistent references with useRef without unnecessary re-renders