Concepts to be known for this assignment:

◇ Core Concepts

- Components
- JSX
- Props
- State
- Conditional Rendering
- Lists + Keys
- Events
- Forms & Controlled Inputs
- Lifting State Up
- Component Reusability
- Folder Structure
- Error handling
- API Integration (fetch/axios)
- Routing (nested routes, dynamic routes, params)
- Navigation, Links

◇ Hooks

- useState
- useEffect
- useContext
- useReducer
- useMemo
- useCallback
- useRef
- Custom Hooks

◇ Advanced Concepts

- Performance Optimization
- Debouncing/Throttling

- Memoized Components (React.memo)
- LocalStorage usage
- Authentication Flow (basic mock)
- Protected Routes
- Global State Management
- Custom Context Providers
- Reusable Form Components
- Modal implementation
- Light/Dark Theme
- Error Boundary (optional)
- Loading & Skeleton UI
- Pagination (optional)

◇ Additional Good-to-Know

- Folder structuring (pages, components, context, hooks, services)
- Project architecture
- Clean code conventions

ASSIGNMENT: "SMART-TASK PRO — A Productivity + Notes + Learning Dashboard"

A full React web app that feels like a real SaaS product

This assignment includes:

✓ All hooks
✓ Routing
✓ Themes
✓ API integration
✓ Global context

✓ Reducers
✓ Memoized values
✓ LocalStorage
✓ Modal
✓ Protected routes
✓ Reusable components
✓ Real interactions
✓ Real project structure

🎯 SCENARIO: “SMART-TASK PRO”

You are building a **personal productivity dashboard** for students & developers.

The app allows users to:

🏠 1. Home Dashboard

- Show greeting based on time (“Good morning Yashas”)
- Task progress summary
- Notes count
- Upcoming reminders
- Theme toggle (light / dark)

📝 2. Task Manager

The user can:

- Add tasks
- Edit tasks
- Delete tasks
- Mark complete/incomplete
- Filter: All / Completed / Pending
- Search tasks (with debouncing)
- Persist tasks in LocalStorage

📓 3. Notes App

- Create notes
- View notes
- Edit notes
- Pin/unpin
- Search notes
- Auto-save every 3 seconds (useEffect)

📚 4. Learning Area

- Shows topics (React, JS, APIs…)
- Each topic has a "Learn More" page using **dynamic routing**
- Fetch topic details from a mock API

👤 5. Profile Page

- Shows name, email, profile picture
- Allows editing the details
- Uses useRef for profile image upload preview

🔐 6. Auth (Mock)

- Login page
- Logout
- "Protected Routes" so only logged-in users can see Dashboard
- Save login session in localStorage

⚙️ 7. Settings

- Theme
- Font size
- Clear all tasks
- Reset app

📌 REQUIRED ROUTES

Students must create these routes using React Router:

Route	Page
/	Landing page
/login	Login form
/dashboard	Home dashboard (protected)
/tasks	Task manager
/notes	Notes app
/learn	All topics
/learn/:topicId	Topic detail page (dynamic)
/profile	User profile
/settings	Settings page

📌 REQUIRED COMPONENTS

create **at least 10+ components**, including:

- Navbar
- Sidebar
- TaskCard
- NoteCard
- Modal
- SearchBar
- Loader / Skeleton
- ThemeToggle
- ProtectedRoute wrapper
- ProfileEditor

🚀 MUST-USE REACT CONCEPTS

Below is what concepts MUST be used, and where.

1️⃣ useState — Everywhere

Examples:

- Task input
- Notes input
- Login form
- Theme selection
- Search box
- Profile fields

🟦 Hint:
Use useState wherever the UI needs to change **immediately when a user types** or interacts.

2️⃣ useEffect — For side-effects

Use in these cases:

- Auto-save notes every 3 seconds
- Fetch learning topics from mock API
- Load tasks from localStorage
- Save tasks to localStorage whenever tasks change
- Apply theme to document.body

🟦 Hint:
Ask: “Does something need to happen when state changes or component loads?” → useEffect.

3️⃣ useContext — Global App State

Create AppContext or AuthContext to share:

- User info
- Theme
- Auth status
- Global loading state

🟦 Hint:
If **3 or more components need the same value**, useContext.

4️⃣ useReducer — Task State Management

Use reducer for:

- Add task
- Edit task
- Delete task
- Mark complete
- Clear all

🟦 Hint:
When state becomes “complex” → prefer useReducer over many useState.

5️⃣ useMemo — Expensive calculations

Use for:

- Counting completed tasks
- Filtering tasks
- Searching notes

🟦 Hint:
If something is “slow” & recalculates unnecessarily → wrap in useMemo.

6️⃣ useCallback — Prevent re-render loops

Use for:

- Passing functions to child components (SearchBar, TaskCard)

🟦 Hint:
If a child re-renders too much → wrap callback in useCallback.

7️⃣ useRef — Non-UI values

Use for:

- Input focus (auto focus on search)
- Storing timer IDs
- Profile picture upload preview

🟦 Hint:
If the value should NOT cause re-render → useRef.

8️⃣ Custom Hook

Create at least one:

- useLocalStorage
- useDebounce
- useFetch

🟦 Hint:
If you repeat logic → turn it into a custom hook.

9️⃣ React Router

Use for multi-page app.

🟦 Hint:
Show how pages can be split into multiple routes.

🔟 React.memo

Use for:

- TaskCard
- NoteCard

🟦 Hint:
If a component receives same props repeatedly → wrap in React.memo.

🎯 FINAL OUTCOME

After completing this assignment, the student will understand:

✓ How to structure a real React project
✓ How to use **every important hook**
✓ How to manage global state
✓ How to create reusable components
✓ How to build real routes
✓ How to work with APIs
✓ How to optimize performance

✓ How to implement localStorage

✓ How to build a full working multi-page application