GOVERNMENTCOLLEGEOFENGINEERING BARGUR,

KRISHNAGIRI-635104

(An Autonomous Institution affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Project Title: IBM-NJ-To Do App with React Hooks

Phase - I

STUDENT NM-ID : 49e50cdea1910dbaea3486fd2603547f

ROLL NO : 2303610710422302

SUBMITTED BY,

NAME:P.MADHUMITHA

MOBILE NO : 8778010311

DATE : 17.09.2025

Project Title: IBM-NJ-TO DO APP WITH REACT HOOKS

1. Problem Statement

Overview:

In today's fast-paced digital learning environment, self-driven learners often struggle to maintain consistent routines across multiple goals—especially when balancing technical skills like coding with language development such as spoken English. While many productivity apps exist, they are often bloated with features, lack personalization, or fail to support the unique needs of habit-based learners.

This project aims to build a lightweight, responsive **To-Do App using React Hooks** that empowers users to manage daily learning tasks, track progress, and build sustainable habits. The app will focus on simplicity, clarity, and habit reinforcement through intuitive design and persistent task tracking.

The Problem:

Learners pursuing self-improvement in areas like programming and spoken English face several challenges:

- **Lack of structure**: Without a clear daily plan, learners often skip practice or lose momentum.
- **No habit visibility**: Existing tools don't show progress or streaks in a motivating way.
- **Overcomplicated interfaces**: Many task apps are cluttered, making them hard to use consistently.
- **No personalization**: Learners need a tool that adapts to their goals—whether it's coding algorithms or speaking fluency.

These gaps lead to inconsistent practice, reduced motivation, and slower skill development

Project Goal:

To design and develop a **React Hooks-based To-Do App** that helps learners:

- Create and manage daily learning tasks
- Track completion status and streaks
- Filter tasks by category (e.g., Coding, English)
- Persist data across sessions using local storage
- Stay motivated through clean UI and habit feedback

The app will serve as a personal productivity companion, optimized for learners who value routine, clarity, and progress tracking.

Why React Hooks?

React Hooks offer a modern, declarative way to manage state and side effects in functional components. By using hooks like useState, useEffect, and optionally useReducer, the app can maintain clean logic, modular components, and responsive behavior—all without relying on class-based components.

Hooks also make it easier to integrate features like:

- Local storage syncing
- Dynamic filtering
- Real-time UI updates
- Component-level state isolation

This makes React Hooks the ideal foundation for building a scalable, maintainable habit-tracking app.

2. Users and Stakeholders

User / Stakeholder	Role & Interests
End Customer	Primary User. Wants to easily browse, search, filter, and find detailed information about IBM-NJ products. Needs accurate and up-to-date data.
Content Manager	Primary Admin User. Responsible for adding new products, updating prices/descriptions, marking items as out-of-stock, and managing product categories. Needs an intuitive interface to perform these tasks without coding.
Marketing Manager	Stakeholder. Wants to feature specific products, create special offers, and ensure product listings are compelling and accurate to drive

	sales.
System Administrator	Stakeholder. Responsible for the deployment, security, maintenance, and scalability of the catalog service. Needs clean APIs, monitoring, and documentation.
End Customer	Primary User. Wants to easily browse, search, filter, and find detailed information about IBM-NJ products. Needs accurate and up-to-date data.
Content Manager	Primary Admin User. Responsible for adding new products, updating prices/descriptions, marking items as out-of-stock, and managing product categories. Needs an intuitive interface to perform these tasks without coding.
Marketing Manager	Stakeholder. Wants to feature specific products, create special offers, and ensure product listings are compelling and accurate to drive
Content Manager	Primary Admin User. Responsible for adding new products, updating prices/descriptions, marking items as out-of-stock, and managing product categories. Needs an intuitive interface to perform these tasks without coding.
Marketing Manager	Stakeholder. Wants to feature specific products, create special offers, and ensure product listings are compelling and accurate to drive
Front-end Developer	Stakeholder. Consumes the Product Catalog API to build the user interface. Needs a well-documented, reliable, and performant API.

3. User Stories

Empower self-driven learners to manage and track daily tasks that reinforce consistent habits in coding, English speaking, and personal growth.

As a learner…

1. **I want to add a new task** so that I can plan my daily learning activities like “Practice Python for 30 minutes” or “Record English speaking.”
2. **I want to mark a task as complete** so that I can visually track which habits I’ve finished today.
3. **I want to delete a task** so that I can remove irrelevant or outdated goals from my list.
4. **I want to edit a task** so that I can update its title or category if my routine changes.
5. **I want my tasks to be saved automatically** so that I don’t lose progress when I refresh or close the app.
6. **I want to filter tasks by category** (e.g., Coding, English, Health) so that I can focus on specific areas of improvement.
7. **I want to see my current streak or progress** so that I stay motivated to maintain daily consistency.
8. **I want the app to work well on mobile and desktop** so that I can access my tasks wherever I’m learning.
9. **I want a clean and distraction-free interface** so that I can focus on completing my habits without unnecessary clutter.
10. **I want to receive a gentle reminder or motivational message** when I haven’t completed a task, so I stay encouraged.

4. MVP (Minimum Viable Product) Features

Purpose:

Create a lightweight, responsive To-Do app using React Hooks that helps users manage daily track progress, and build consistent habits.

Core MVP Features:

1. **Task Creation**
   - Users can add new tasks with a title and optional category (e.g., Coding, English).
   - Input field and "Add Task" button trigger task creation.
   - Uses useState to manage task input and list.
2. **Task Display**
   - Tasks are shown in a scrollable list with status indicators.
   - Each task includes a checkbox, title, and delete icon.
   - Responsive layout for mobile and desktop.
3. **Mark as Complete**
   - Users can toggle task status (complete/incomplete) via checkbox.
   - Completed tasks are visually styled (e.g., strikethrough or faded).
   - Uses useState to update task status.
4. **Delete Task**
   - Users can remove tasks from the list with a delete button.
   - Immediate UI update using state mutation.
5. **Local Storage Persistence**
   - Tasks are saved in localStorage to persist across sessions.
   - Uses useEffect to sync state with browser storage.
6. **Category Filtering (Optional)**
   - Users can filter tasks by category (e.g., Coding, English).
   - Simple dropdown or tab-based filter UI.
   - Uses useState to manage filter selection.

React Hooks Used:

Hook	Purpose
useState	Manage task list, input fields, filters
useEffect	Sync with localStorage, initialize tasks

UI Behavior:

- Clean, distraction-free interface

- Mobile-first responsive design

Feature	Criteria
Add Task	Task appears instantly in the list after submission
Complete Task	Checkbox toggles task status and updates UI
Delete Task	Task is removed from list immediately
Persist Tasks	Tasks remain after page refresh using localStorage
Responsive UI	App adapts to different screen sizes

- Instant feedback on task actions (add, complete, delete)

5. API Endpoint List & Wireframes

Key UI Elements:

- **Task Input**: Text field + button to add new tasks
- **Task List**: Checkbox to mark complete, delete icon
- **Filter Tabs**: Category-based filtering (optional)
- **Progress Button**: Future feature to show streaks or charts

Wireframe Overview

|

Key UI Elements:

- **Task Input**: Text field + button to add new tasks
- **Task List**: Checkbox to mark complete, delete icon
- **Filter Tabs**: Category-based filtering (optional)
- **Progress Button**: Future feature to show streaks or charts

API Endpoint List (for future backend integration):

If you expand the app beyond local storage, here's a clean RESTful API structure:

Method	Endpoint	Description
GET	/api/tasks	Fetch all tasks
POST	/api/tasks	Add a new task
PUT	/api/tasks/:id	Update task (e.g., mark complete, edit)
DELETE	/api/tasks/:id	Delete a task
GET	/api/progress/:id	Fetch weekly progress for a specific task

Optional Future Endpoints:

- POST /api/users/login – User authentication
- GET /api/tasks?category=Coding – Filter by category
- GET /api/streaks/:id – Retrieve streak count for a habit

6. Acceptance Criteria

1. Add Task

User can enter a task name in an input field.

Clicking “Add” (or pressing Enter) will add the task to the task list.Task must not be empty.

2. View Tasks

User can see a list of all tasks they have added.

Each task is displayed with its text and status (pending/completed).

3. Mark Task as Completed

User can click a checkbox/button to mark a task as completed.

Completed tasks should appear visually different (e.g., strikethrough or faded).

4. Delete Task

User can remove a task from the list.

Confirmation (optional) before deletion.

5. Edit Task (optional)

User can click on a task to edit its text.

Changes should be saved after pressing Enter or clicking Save.

6. Persistent State (optional)

Tasks remain saved after page reload using localStorage.

7. Filter/Sort (optional)

User can filter tasks by “All”, “Active”, “Completed”.

Sorting by newest/oldest (optional).

8. Responsive UI

App works properly on mobile and desktop screens.

9. Accessibility

Input fields and buttons should be accessible via keyboard.

readers can read task names.

10. Error Handling

If user tries to add an empty task, show a warning.

User and stackholders:

Student name	Role and Responsibility
L.RAMLEKHA	Define goals ,manage timeline, final presentation
L.U.RINEESHAA	Maintain documentation,compile report,create user manual
P.MADHUMITHA	Build and host web based app
S. HEMALATHA	Handle training ,testing ,tuning and performance reporting
M. THENMOZHI	Perform visualize trends,contribute to feature engineering

For API Endpoint DELETE /api/products/:id (Admin):

AC1: Deletes the product with the provided :id from the database.

AC2: Requires authentication.

AC3: Returns HTTP status code 200 with a confirmation message on success (e.g., { "message": "Product deleted successfully" }).

AC4: Returns status 404 if the product to delete is not found.

This document provides a solid foundation for your project. Good luck with building the IBM-NJ-Product Catalog