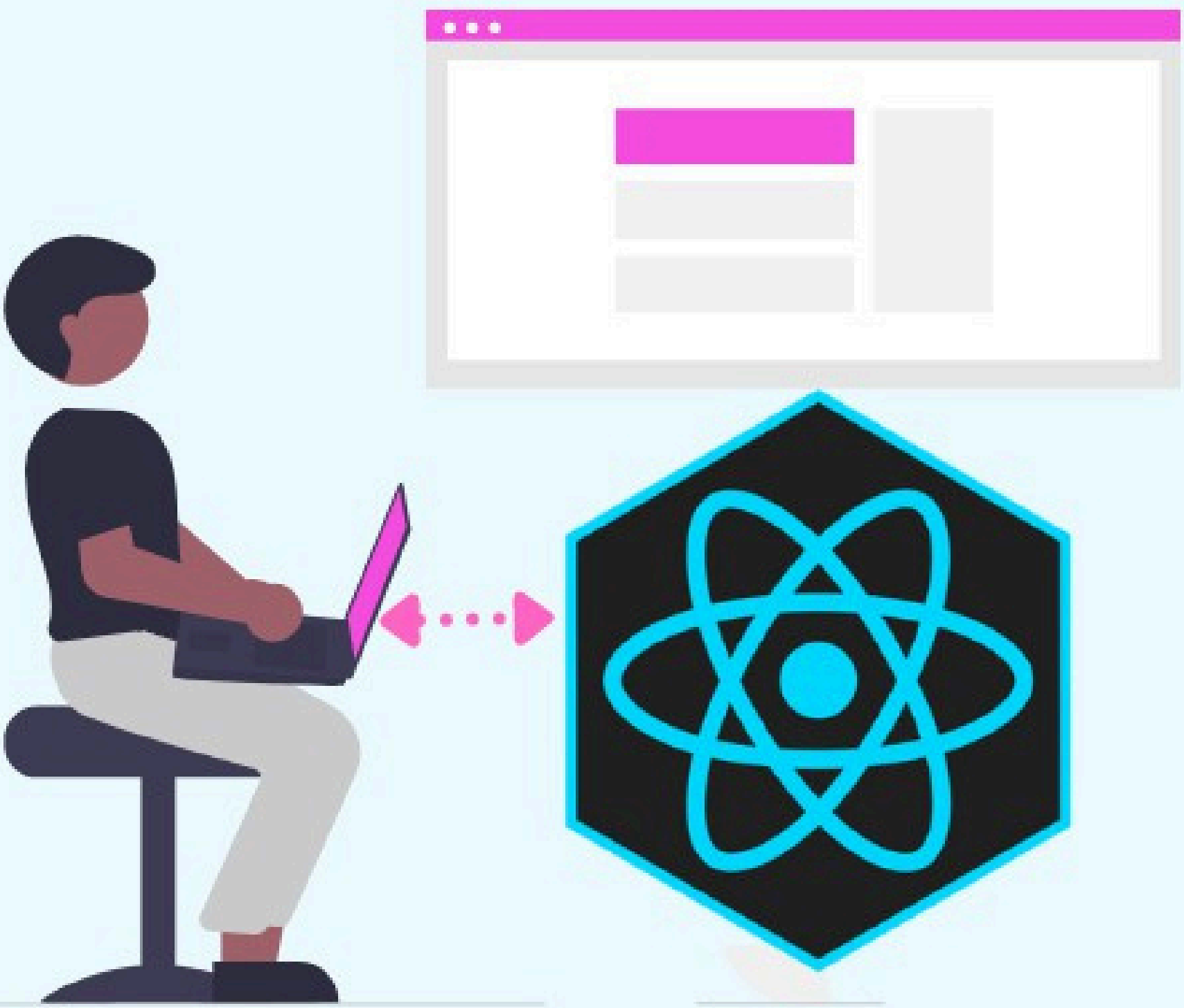
Day 26

Shifting from Theory to Practice (Project Day 2)



@oluwakemi Oluwadahunsi

What we will be doing today?

- Create the MoodContext for all moods logic
- Create the MoodSelector component
- Create the ActivityTracker section
- Create the Mood-based Suggestions section
- Create the Recent Entries section
- Create the MoodCalender page
- Create the LoadingFallback component

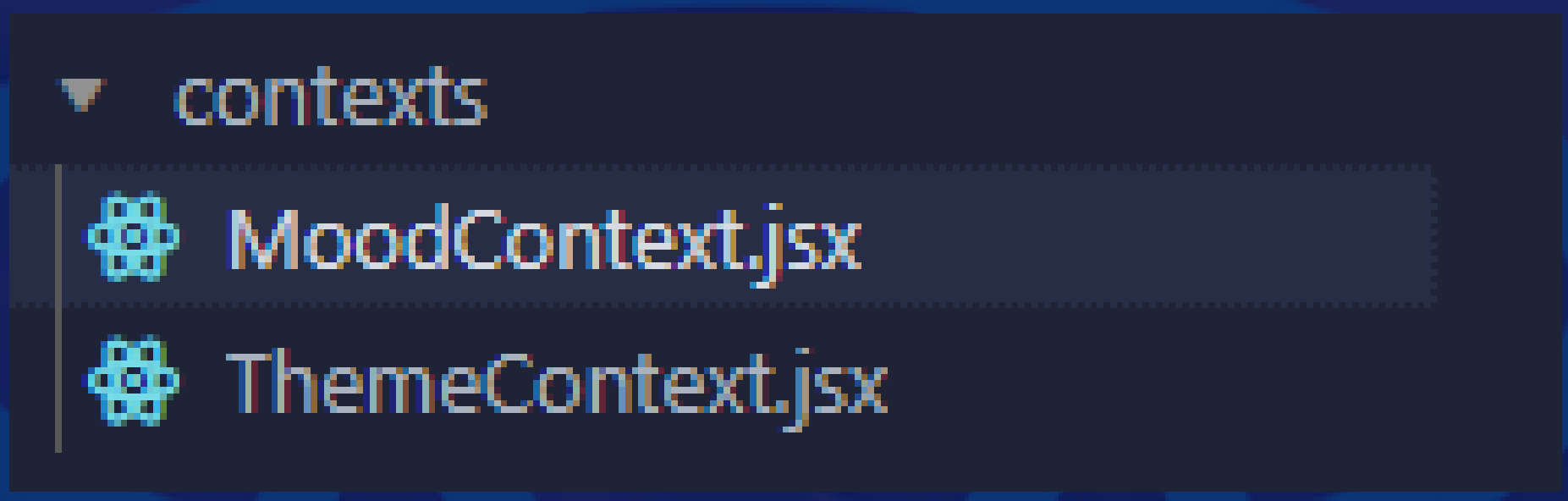
Things to note:

- There would be component codes images, some could be more than 1, I will indicate on each image which component it is together with the image number, just continue in the same component according to their numbers.
- Be sure to use correct paths for imports of components and files. If the files paths are not correct in mine, fix it in yours, else it will throw errors, our project structure may be different.

Mood Context

This is the context component that will be handling all moods logic, from selection to updating, deleting, and storing moods.

First, create a new file in your **context** folder named “**MoodContext.jsx**”



What this file will be doing:

- Manages the mood tracking state of the application, handling the log, update, store, or delete mood logics, with other related data (such as time, date, or activities associated with each mood entry).

MoodContext Image 1

```jsx
import { createContext, useState, useEffect, useCallback } from "react";

export const MoodContext = createContext();

const defaultMoods = [
  { label: "Happy", emoji: "😄", color: "#FFD700" },
  { label: "Sad", emoji: "😢", color: "#4169E1" },
  { label: "Calm", emoji: "😊", color: "#4ade80" },
  { label: "Neutral", emoji: "😐", color: "#9ca3af" },
  { label: "Excited", emoji: "🎉", color: "#FF1493" },
  { label: "Relaxed", emoji: "😌", color: "#98FB98" },
  { label: "Angry", emoji: "😠", color: "#FF4500" },
];

export const MoodProvider = ({ children }) => {
  const [moodEntries, setMoodEntries] = useState(() => {
    const storedMoodEntries = localStorage.getItem("moodEntries");
    return storedMoodEntries ? JSON.parse(storedMoodEntries) : [];
  });

  const [customMoods, setCustomMoods] = useState(() => {
    const storedCustomMoods = localStorage.getItem("customMoods");
    return storedCustomMoods ? JSON.parse(storedCustomMoods) : [];
  });

  const [activities, setActivities] = useState(() => {
    const storedActivities = localStorage.getItem("activities");
    return storedActivities ? JSON.parse(storedActivities) : [];
  });

  useEffect(() => {
    localStorage.setItem("moodEntries", JSON.stringify(moodEntries));
  }, [moodEntries]);

  useEffect(() => {
    localStorage.setItem("customMoods", JSON.stringify(customMoods));
  }, [customMoods]);

  useEffect(() => {
    localStorage.setItem("activities", JSON.stringify(activities));
  }, [activities]);

  const addMoodEntry = useCallback((entry) => {
    setMoodEntries((prevEntries) => {
      const newEntries = [...prevEntries, entry];
      localStorage.setItem("moodEntries", JSON.stringify(newEntries));
      return newEntries;
    });
  }, []);

  const addCustomMood = useCallback((mood) => {
    setCustomMoods((prevMoods) => {
      const newMoods = [...prevMoods, mood];
      localStorage.setItem("customMoods", JSON.stringify(newMoods));
      return newMoods;
    });
  }, []);
```

default moods declared

each useState hook is used to both store and get the moods logic in the localStorage

each useEffect hook handles the side effects for each logic

the addMoodEntry function handles the adding of each mood entry.

the addCustomMood function handles the adding of the user's customized mood.

Continuation of the MoodContext file codes

```jsx
const addActivity = useCallback((activity) => {
    setActivities((prevActivities) => {
      const newActivities = [...prevActivities, activity];
      localStorage.setItem("activities", JSON.stringify(newActivities));
      return newActivities;
    });
  }, []);

  const updateMoodEntry = useCallback((date, updates) => {
    setMoodEntries((prevEntries) =>
      prevEntries.map((entry) =>
        entry.date === date ? { ...entry, ...updates } : entry
      )
    );
  }, []);

  const removeMoodEntry = useCallback((date) => {
    setMoodEntries((prevEntries) =>
      prevEntries.filter((entry) => entry.date !== date)
    );
  }, []);

  const getAllMoods = useCallback(() => {
    return [...defaultMoods, ...customMoods];
  }, [customMoods]);

  return (
    <MoodContext.Provider
      value={{
        moodEntries,
        addMoodEntry,
        customMoods,
        addCustomMood,
        activities,
        addActivity,
        getAllMoods,
        updateMoodEntry,
        removeMoodEntry,
      }}
    >
      {children}
    </MoodContext.Provider>
  );
};
```

the addActivity function handles the adding of the activities selected to each mood entry.

this function handles the updating of the each mood entry in the RecentEntries section.

this removeMoodEntry function handles the deleting a particular mood entry from the RecentEntries section.

this getAllMoods function handles the updating of the MoodSelection section with the new user's customized mood.

these are all values (props) exported to be used in the respectful components needing them.

@oluwakemi Oluwadahunsi

To get the emojis for the defaultMoods, on your keyboard, press the windows + . keys (for windows machines) or Control + Command + Spacebar (for mac users) to pop up the onscreen keyboard for emojis (learned something new today 😀).

Mood Selector component

Now, let's use these values in the components:

We will be creating the First section: MoodSelector component. In your "src" folder, create a new file named "**MoodSelector.jsx**".

Also, in the components folder, create a new folder named "**staticComponents**". This is where we will have all our non-dynamic components, like buttons, inputs, etc.. And create a file in it named "**MoodButton.jsx**"

Let's populate the MoodButton component first:

MoodButton Component

these are all values (props) exported to be respectful components needing them.

```jsx
import React from "react";

const MoodButton = React.memo(({ mood, onClick, isSelected }) => {
  if (!mood) {
    return null; // Don't render anything if mood is undefined
  }

  const { label, emoji, color } = mood;
  const backgroundColor = color || "#CCCCCC"; // Fallback color if not provided

  return (
    <button
      onClick={() => onClick(mood)}
      className={`p-4 rounded-lg text-center transition-colors duration-200 ${
        isSelected ? "ring-2 ring-offset-2 ring-indigo-500" : ""
      }`}
      style={{ backgroundColor }}
    >
      <span className="text-3xl" role="img" aria-label={label}>
        {emoji}
      </span>
      <p className="mt-2 text-sm font-medium">{label}</p>
    </button>
  );
});

MoodButton.displayName = "MoodButton";

export default MoodButton;
```

This component is used for styling each of the default moods. It’s a **reusable component**, we will be using it for both **MoodSelector** and **CustomMoodLabels** components.

@oluwakemi Oluwadahunsi

MoodSelector image 1

```jsx
import { useState, useContext, useCallback, useEffect } from "react";
import { MoodContext } from "../hooks/MoodContext";
import MoodButton from "./pages/staticPages/MoodButton";

function MoodSelector() {
  const { addMoodEntry, getAllMoods } = useContext(MoodContext);
  const [selectedMood, setSelectedMood] = useState(null);
  const [note, setNote] = useState("");
  const [allMoods, setAllMoods] = useState([]);

  useEffect(() => {
    const moods = getAllMoods();
    setAllMoods(moods);
  }, [getAllMoods]);

  const handleMoodSelect = useCallback((mood) => {
    setSelectedMood(mood);
  }, []);

  const handleSubmit = useCallback(
    (e) => {
      e.preventDefault();
      if (selectedMood) {
        const newEntry = {
          mood: selectedMood,
          note: note,
          date: new Date().toISOString(),
        };
        addMoodEntry(newEntry);
        alert("Mood added! Check Recent entries");
        setSelectedMood(null);
        setNote("");
      }
    },
    [selectedMood, note, addMoodEntry]
  );

  if (allMoods.length === 0) {
    return <div>Loading moods ...</div>;
  }
```

MoodContext accessd using useContext hook

getAllMoods function from MoodContext

function to select when a mood is clicked

memoized function called to log the current mood selected together with the note if any.

if no moods entry, returns this div

@oluwakemi Oluwadahunsi

MoodSelector image 2

```jsx
return (
    <>
      <div className="mb-8">
        <h2 className="text-2xl font-bold mb-4 text-gray-800 dark:text-white">
          How are you feeling?
        </h2>
        <div className="grid grid-cols-3 sm:grid-cols-5 gap-4 mb-4">
          {allMoods.map((mood) => (
            <MoodButton
              key={mood.label}
              mood={mood}
              onClick={handleMoodSelect}
              isSelected={selectedMood === mood}
            />
          ))}
        </div>
        {selectedMood && (
          <form onSubmit={handleSubmit} className="space-y-4">
            <div>
              <label
                htmlFor="note"
                className="block text-sm font-medium text-gray-700 dark:text-gray-300"
              >
                Add a note (optional)
              </label>
              <textarea
                id="note"
                rows="3"
                className="mt-1 px-4 block w-full rounded-md border border-gray-300 dark:border-none shadow-lg focus:outline-none focus:ring focus:ring-indigo-200 focus:ring-opacity-50 dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                value={note}
                onChange={(e) => setNote(e.target.value)}
              ></textarea>
            </div>
            <button
              type="submit"
              className="w-full bg-indigo-600 text-white py-2 px-4 rounded-md hover:bg-indigo-700 transition-colors duration-200"
            >
              Log Mood
            </button>
          </form>
        )}
      </div>
    </>
  );
}

export default MoodSelector;
```

maps all moods in, both the defaultMoods and the custom moods that could be created.

@oluwakemi Oluwadahunsi

ActivityTracker Component

The activity tracker component consist of the activities selected together with the moods.

We have 2 components associated with this component: “**ActivityButton.jsx**” and “**activitiesIcon.js**”

The ActivityButton component is used for styling the UI of each activity button (how each button looks like for consistent UI), while the activitiesIcons.js is a **file** where the icons used for each activity is stored.

Create these 2 files in the **staticComponents** folder.

The ActivityButton component:

@oluwakemi Oluwadahunsi

ActivityButton component

```jsx
import React from "react";

function ActivityButton({ activity, Icon, isSelected, onToggle
}) {
  return (
    <button
      onClick={() => onToggle(activity)}
      className={`flex items-center px-4 py-2 rounded-full
text-sm font-medium transition-colors duration-200 ${
        isSelected
          ? "bg-indigo-600 text-white"
          : "bg-gray-200 text-gray-800 hover:bg-gray-300
dark:bg-gray-700 dark:text-gray-200 dark:hover:bg-gray-600"
      }`}
    >
      {Icon && <Icon className="mr-2 h-4 w-4" />}
      {activity}
    </button>
  );
}

export default React.memo(ActivityButton);
```

memoized component

The component is wrapped in **React.memo** to optimize performance by preventing unnecessary re-renders. React.memo ensures that the component only re-renders if its props (activity, Icon, isSelected, or onToggle) change.

@oluwakemi Oluwadahunsi

In the activitiesIcon.js file: This basically stores icons for the activities.

activitiesicon.js

```
import {
  Dumbbell,
  Briefcase,
  Users,
  BookOpen,
  Meh,
  Palette,
  Utensils,
  Film,
  Gamepad,
  Plane,
  ShoppingBag,
  Music,
  Moon,
} from "lucide-react";

export const activityIcons = {
  Exercise: Dumbbell,
  "Good sleep": Moon,
  Work: Briefcase,
  Socializing: Users,
  Reading: BookOpen,
  Meditation: Meh,
  Hobby: Palette,
  Eating: Utensils,
  "Watching Movies": Film,
  Gaming: Gamepad,
  Traveling: Plane,
  Shopping: ShoppingBag,
  "Listening to Music": Music,
```

Now the ActivityTracker component: create this component directly in your **components** folder.

ActivityTracker Image 1

```jsx
import React, {useState, useContext, useCallback, useEffect, useMemo}from "react";
import { MoodContext } from "../hooks/MoodContext";
import ActivityButton from "./pages/staticPages/ActivityButton";
import { activityIcons } from "./pages/staticPages/activitiesIcon";

const defaultActivities = [
  "Exercise", "Good sleep", "Work", "Socializing", "Reading", "Meditation",
  "Hobby", "Eating", "Watching Movies", "Gaming", "Traveling", "Shopping",
  "Listening to Music",
];

function ActivityTracker() {
  const { moodEntries, updateMoodEntry, activities, addActivity } =
    useContext(MoodContext);
  const [selectedActivities, setSelectedActivities] = useState([]);
  const [customActivity, setCustomActivity] = useState("");

  const lastEntry = useMemo(() => {
    return moodEntries.length > 0 ? moodEntries[moodEntries.length - 1] : null;
  }, [moodEntries]);

  useEffect(() => {
    if (lastEntry && lastEntry.activities) {
      setSelectedActivities(lastEntry.activities);
    }
  }, [lastEntry]);

  const handleActivityToggle = useCallback(
    (activity) => {
      setSelectedActivities((prev) => {
        const newActivities = prev.includes(activity)
          ? prev.filter((a) => a !== activity)
          : [...prev, activity];

        if (lastEntry) {
          updateMoodEntry(lastEntry.date, { activities: newActivities });
        }

        return newActivities;
      });
    },
    [lastEntry, updateMoodEntry]
  );
```

this sets the state for custom activity added

handles the select and unselect events of each of the activities

@oluwakemi Oluwadahunsi

ActivityTracker image 2

```
const handleCustomActivityAdd = useCallback(
    (e) => {
      e.preventDefault();
      if (customActivity && !activities.includes(customActivity)) {
        addActivity(customActivity);
        setSelectedActivities((prev) => {
          const newActivities = [...prev, customActivity];
          if (lastEntry) {
            updateMoodEntry(lastEntry.date, { activities: newActivities });
          }
          return newActivities;
        });
        setCustomActivity("");
      }
    },
    [customActivity, activities, addActivity, lastEntry, updateMoodEntry]
  );

  const allActivities = useMemo(
    () => [...defaultActivities, ...activities],
    [activities]
  );

  return (
    <div className="mb-8">
      <h2 className="text-2xl font-bold mb-4 text-gray-800 dark:text-white">
        Activities
      </h2>
      <div className="flex flex-wrap gap-2 mb-4">
        {allActivities.map((activity) => (
          <ActivityButton
            key={activity}
            activity={activity}
            Icon={activityIcons[activity]}
            isSelected={selectedActivities.includes(activity)}
            onToggle={handleActivityToggle}
          />
        ))}
      </div>
```

@oluwakemi Oluwadahunsi

ActivityTracker image 3

```jsx
<form onSubmit={handleCustomActivityAdd} className="mb-4">
        <label htmlFor="custom-activity" className="sr-only">
          Add custom activity
        </label>
        <input
          id="custom-activity"
          type="text"
          value={customActivity}
          onChange={(e) => setCustomActivity(e.target.value)}
          placeholder="Add custom activity"
          className="w-full py-2 px-4 rounded-md border border-gray-300 dark:border-none
                     shadow-lg focus:outline-none focus:ring focus:ring-indigo-200
                     focus:ring-opacity-50 dark:bg-gray-700 dark:border-gray-600
                     dark:text-white" />
        <button
          type="submit"
          className="mt-2 bg-indigo-600 text-white py-2 px-4 rounded-md hover:bg-indigo-700
                     transition-colors duration-200" >
          Add Custom Activity
        </button>
      </form>
    </div>
  );
}

export default React.memo(ActivityTracker);
```

It uses the **MoodContext** to access and manipulate mood entries (**moodEntries**), update mood entries (**updateMoodEntry**), retrieve activities (**activities**), and add new activities (**addActivity**).

handleCustomActivityAdd function allows users to add custom activities. If the activity doesn't already exist. Consuming the **addActivity** function from **MoodContext**, it adds the custom activity to the list and selects it for the current mood entry.

@oluwakemi Oluwadahunsi

MoodSuggestions Comonponent

This component displays engaging activities suggestions based on the current mood chosen.

It makes use of a file named “**suggestions.js**”.
Create this file in the **staticComponents** folder.

In the file, type these out:

suggestions.js image 1

```
export const moodSuggestions = {
  Happy: [
    "Celebrate your good mood",
    "Share your happiness with others",
    "Try a new hobby",
    "Express gratitude",
    "Set new goals",
  ],
  Excited: [
    "Channel your energy into a creative project",
    "Plan an adventure",
    "Share your enthusiasm with friends",
    "Start learning something new",
    "Write down your ideas",
  ],
  Calm: [
    "Practice mindfulness",
    "Do some light stretching",
    "Listen to soothing music",
    "Enjoy a relaxing hobby",
    "Spend time in nature",
  ],
```

@oluwakemi Oluwadahunsi

suggestions.js image 2

```
Relaxed: [
    "Read a book",
    "Take a leisurely walk",
    "Practice deep breathing",
    "Enjoy a warm bath",
    "Do some gentle yoga",
  ],
  Neutral: [
    "Take a short walk",
    "Write in a journal",
    "Call a friend",
    "Try a new recipe",
    "Organize your space",
  ],
  Sad: [
    "Talk to someone you trust",
    "Practice self-care",
    "Watch a comforting movie",
    "Listen to uplifting music",
    "Engage in light exercise",
  ],
  Angry: [
    "Take deep breaths",
    "Exercise to release tension",
    "Write down your feelings",
    "Practice progressive muscle relaxation",
    "Engage in a calming activity",
  ],
};
```

@oluwakemi Oluwadahunsi

MoodSuggestions.js

```jsx
import React, { useContext, useMemo } from "react";
import { MoodContext } from "../hooks/MoodContext";
import { useTheme } from "../hooks/ThemeContext";
import { moodSuggestions } from "./pages/staticPages/moodSuggestions";

const Suggestion = React.memo(({ text }) => {
  const { darkMode } = useTheme();
  return (
    <li className={`mb-2 ${darkMode ? "text-gray-200" : "text-gray-800"}`}>
      {text}
    </li>
  );
});

Suggestion.displayName = "Suggestion";

function MoodSuggestions() {
  const { moodEntries } = useContext(MoodContext);
  const { darkMode } = useTheme();

  const latestMood = useMemo(() => {
    if (moodEntries.length === 0) return null;
    return moodEntries[moodEntries.length - 1].mood.label;
  }, [moodEntries]);

  const currentSuggestions = useMemo(() => {
    if (!latestMood || !moodSuggestions[latestMood]) return null;
    return moodSuggestions[latestMood];
  }, [latestMood]);

  if (!currentSuggestions) {
    return null;
  }

  return (
    <div
      className={`mb-8 p-6 rounded-lg shadow-md ${
        darkMode ? "bg-gray-800" : "bg-white"
      }`}
    >
      <h2
        className={`text-xl sm:text-2xl font-bold mb-4 ${
          darkMode ? "text-white" : "text-gray-900"
        }`}
      >
        Mood-Based Suggestions for {latestMood}
      </h2>
      <ul className="list-disc pl-5">
        {currentSuggestions.map((suggestion, index) => (
          <Suggestion key={index} text={suggestion} />
        ))}
      </ul>
    </div>
  );
}

export default React.memo(MoodSuggestions);
```

@oluwakemi Oluwadahunsi

RecentEntries Comonponent

This component displays all recent entries of moods by date.
It makes use of a component named "**MoodEntry.jsx**". This component is used to style each entry card and also makes use of the activitiesicon.js file.
Create this file in the **staticComponents** folder.

MoodEntry.jsx image 1

```jsx
import React from "react";
import { format, parseISO } from "date-fns";
import { activityIcons } from "./activitiesIcon";
import { X } from "lucide-react";

function MoodEntry({ entry, onRemove }) {
  return (
    <div className="bg-gray-100 dark:bg-gray-700 p-4 rounded-lg relative">
      <button
        onClick={onRemove}
        className="absolute top-2 right-4 text-gray-500 hover:text-red-500 transition-colors duration-200"
        aria-label="Remove entry"
      >
        <X size={20} />
      </button>
      <div className="flex justify-between items-center mb-2">
        <span className="font-semibold text-gray-700 dark:text-gray-300">
          {format(parseISO(entry.date), "h:mm a")}
        </span>
        <span className="text-2xl mt-4 sm:mt-6" role="img" aria-label={entry.mood.label}>
          {entry.mood.emoji}
        </span>
      </div>
      <p className="text-lg font-medium text-gray-800 dark:text-white mb-2">
        {entry.mood.label}
      </p>
```

MoodEntry.jsx image 2

```jsx
{entry.note && (
        <p className="text-sm text-gray-600 dark:text-gray-400 mb-2">
          {entry.note}
        </p>
      )}
      {entry.activities && entry.activities.length > 0 && (
        <div className="flex flex-wrap gap-2">
          {entry.activities.map((activity, i) => {
            const IconComponent = activityIcons[activity] || null;
            return (
              <span
                key={i}
                className="inline-flex items-center bg-blue-100 text-blue-800 text-xs font-semibold px-2.5 py-0.5 rounded dark:bg-blue-200 dark:text-blue-800"
              >
                {IconComponent && <IconComponent className="mr-1 h-4 w-4" />}
                {activity}
              </span>
            );
          })}
        </div>
      )}
    </div>
  );
}

export default React.memo(MoodEntry);
```

Now, let’s create the RecentEntries component.

Create a file in the components folder and name it “**RecentEntries.jsx**”

@oluwakemi Oluwadahunsi

RecentEntries image 1

```
1 import React, { useCallback, useContext, useMemo } from "react";
2 import { MoodContext } from "../hooks/MoodContext";
3 import { format, parseISO } from "date-fns";
4 import MoodEntry from "./pages/staticPages/MoodEntry";
5
6
7
8 function RecentEntries() {
9   const { moodEntries, removeMoodEntry } = useContext(MoodContext);
10
11   const groupedEntries = useMemo(() => {
12     return moodEntries.reduce((acc, entry) => {
13       const date = format(parseISO(entry.date), "yyyy-MM-dd");
14       if (!acc[date]) {
15         acc[date] = [];
16       }
17       acc[date].push(entry);
18       return acc;
19     }, {});
20   }, [moodEntries]);
21
22   const handleRemoveEntry = useCallback(
23     (entryDate) => {
24       removeMoodEntry(entryDate);
25       alert("Mood entry removed successfully");
26     },
27     [removeMoodEntry]
28   );
```

@oluwakemi Oluwadahunsi

RecentEntries image 2

```jsx
return (
    <div className="mt-12">
      <h2 className="text-lg sm:text-2xl font-bold mb-4 text-gray-800 dark:text-white">
        Recent Entries
      </h2>
      <div className="space-y-8 max-h-[600px] overflow-y-auto pr-8 mx-auto">
        {Object.entries(groupedEntries)
          .sort(([dateA], [dateB]) => new Date(dateB) - new Date(dateA))
          .map(([date, entries]) => (
            <div
              key={date}
              className="bg-white dark:bg-gray-800 shadow-md rounded-lg p-4"
            >
              <h3 className="text-lg font-semibold mb-2 text-gray-800 dark:text-white">
                {format(parseISO(date), "MMMM d, yyyy")}
              </h3>
              <div className="space-y-4">
                {entries.map((entry, index) => (
                  <MoodEntry
                    key={`${date}-${index}`}
                    entry={entry}
                    onRemove={() => handleRemoveEntry(entry.date)}
                  />
                ))}
              </div>
            </div>
          ))}
      </div>
    );
}

export default React.memo(RecentEntries);
```

@oluwakemi Oluwadahunsi

Calender Comonponent

This component displays the current mood for the day on a calender.

It makes use of a component named "**CalendarDay.jsx**". This component is used to enter the selected mood into the MoodCalender components.

Create this file in the **staticComponents** folder.

CalendarDay.jsx

```jsx
import React from "react";
import { format } from "date-fns";

const CalendarDay = ({ day, moodEntry, isSelected, onSelect }) => (
  <button
    onClick={() => onSelect(day)}
    className={`p-2 text-center rounded-lg ${
      moodEntry ? moodEntry.mood.color : "bg-gray-200 dark:bg-gray-700"
    } ${isSelected ? "ring-2 ring-indigo-500" : ""}`}
    aria-label={`Select date: ${format(day, "MMMM d, yyyy")}`}
  >
    <span className="block text-sm">{format(day, "d")}</span>
    {moodEntry && (
      <span
        className="block text-2xl"
        role="img"
        aria-label={moodEntry.mood.label}
      >
        {moodEntry.mood.emoji}
      </span>
    )}
  </button>
);

export default React.memo(CalendarDay);
```

@oluwakemi Oluwadahunsi

Create a file named “**MoodCalendar.jsx**” in components folder.

MoodCalendar.jsx image 1

```jsx
import React, { useState, useContext, useMemo } from "react";
import { MoodContext } from "../hooks/MoodContext";
import {
  format,
  startOfMonth,
  endOfMonth,
  eachDayOfInterval,
  isSameDay,
  isSameMonth,
  addDays,
} from "date-fns"
import CalendarDay from "./pages/staticPages/CalendarDay";

CalendarDay.displayName = "CalendarDay";

function MoodCalendar() {
  const { moodEntries } = useContext(MoodContext);
  const [selectedDate, setSelectedDate] = useState(null);
  const today = new Date();

  const { daysInMonth, moodEntriesMap } = useMemo(() => {
    const monthStart = startOfMonth(today);
    const monthEnd = endOfMonth(today);
    const firstDayOfWeek = addDays(monthStart, -monthStart.getDay());
    const lastDayOfWeek = addDays(monthEnd, 6 - monthEnd.getDay());

    const days = eachDayOfInterval({
      start: firstDayOfWeek,
      end: lastDayOfWeek,
    });

    const entriesMap = new Map(
      moodEntries.map((entry) => [
        format(new Date(entry.date), "yyyy-MM-dd"),
        entry,
      ])
    );

    return { daysInMonth: days, moodEntriesMap: entriesMap };
  }, [moodEntries, today]);

  const getMoodForDay = (day) => {
    return moodEntriesMap.get(format(day, "yyyy-MM-dd"));
  };
```

MoodCalendar.jsx image 2

```jsx
return (
    <div className="mb-8">
      <h2 className="text-2xl font-bold mb-4 text-gray-800 dark:text-white">
        Mood Calendar
      </h2>
      <div className="grid grid-cols-7 gap-2 mb-4">
        {["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"].map((day) => (
          <div
            key={day}
            className="text-center font-bold text-gray-600 dark:text-gray-400"
          >
            {day}
          </div>
        ))}
        {daysInMonth.map((day, index) => (
          <CalendarDay
            key={index}
            day={day}
            moodEntry={getMoodForDay(day)}
            isSelected={selectedDate && isSameDay(day, selectedDate)}
            onSelect={setSelectedDate}
            isCurrentMonth={isSameMonth(day, today)}
          />
        ))}
      </div>
      {selectedDate && (
        <div className="mt-4 p-4 bg-gray-100 dark:bg-gray-700 rounded-lg">
          <h3 className="text-lg font-semibold mb-2">
            {format(selectedDate, "MMMM d, yyyy")}
          </h3>
          {getMoodForDay(selectedDate) ? (
            <>
              <p>Mood: {getMoodForDay(selectedDate).mood.label}</p>
              <p>Note: {getMoodForDay(selectedDate).note || "No note added"}</p>
            </>
          ) : (
            <p>No mood logged for this day</p>
          )}
        </div>
      )}
    </div>
  );
}

export default React.memo(MoodCalendar);
```

@oluwakemi Oluwadahunsi

LoadingFallback Comonponent

This component is designed as a loader for when a loading state is true. It uses the tailwindcss class of **animate-spin** for its animation.

CalendarDay.jsx

```jsx
function LoadingFallback() {
  return (
    <div className="flex items-center justify-center h-64">
      <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-indigo-500"></div>
    </div>
  );
}

export default LoadingFallback;
```

@oluwakemi Oluwadahunsi

Updating App.jsx

Let’s update our **App.jsx** file to make use of these components.

We will be using the React lazy and Suspense methods to lazy load these components for explanation purposes. it is a simple project which does not need these methods, but I want to use this medium to explain how they are being used.

Now, go into your **App.jsx** file and update it this way:

@oluwakemi Oluwadahunsi

```jsx
import { lazy, Suspense, useState } from "react";
import Header from "./components/Header";
import { ThemeProvider } from "./contexts/ThemeContext";
import LoadingFallback from "./components/staticComponents/LoadingFallback";

const MoodSelector = lazy(() => import("./components/MoodSelector"));
const MoodCalendar = lazy(() => import("./components/MoodCalender"));
const ActivityTracker = lazy(() => import("./components/ActivityTracker"));
const MoodSuggestions = lazy(() => import("./components/MoodSuggestions"));
const RecentEntries = lazy(() => import("./components/RecentEntries"));

function App() {
  const [activeTab, setActiveTab] = useState("today");
  return (
    <ThemeProvider>
      <div className="min-h-screen bg-gradient-to-br from-purple-400 to-indigo-600 dark:from-gray-600 dark:to-gray-900 transition-colors duration-200 shadow-2xl">
        <div className="max-w-4xl mx-auto px-4 sm:px-6 lg:px-8 py-6 sm:py-8 lg:py-12">
          <Header activeTab={activeTab} setActiveTab={setActiveTab} />
        </div>
        <main className="p-4 sm:p-6 lg:p-8">
          <Suspense fallback={<LoadingFallback />}>
            {activeTab === "today" && (
              <div className="space-y-6 sm:space-y-8">
                <MoodSelector />
                <ActivityTracker />
                <MoodSuggestions />
                <RecentEntries />
              </div>
            )}
            {activeTab === "calendar" && <MoodCalendar />}

          </Suspense>
        </main>
      </div>
    </ThemeProvider>
  );
}
export default App;
```

@oluwakemi Oluwadahunsi

This file utilizes React's **lazy loading** to optimize performance by only loading components when needed. It uses the **useState** hook to manage the currently **active tab** ("**today**" or "**calendar**"), allowing the user to toggle between different views.

Suspense from React displays a LoadingFallback while lazy-loaded components are being fetched.

The active tab for **Today** is the landing page which consist of the **MoodSelector**, **ActivityTracker**, **MoodSuggestions**, and **RecentEntries** components.

While the active tab for **Calender** renders the **MoodCalendar** component.

The active tabs had been updated already in the Header component we did yesterday (part 1), so you don’t need to update anything again.

@oluwakemi Oluwadahunsi

We will stop here today.

We will continue with other parts of the projects tomorrow.

@oluwakemi Oluwadahunsi

I hope you found this material useful and helpful.

Remember to:

Like

Save for future reference

&

Share with your network, be helpful to someone 👌

@oluwakemi Oluwadahunsi

Hi There!

Thank you for reading through

Did you enjoy this knowledge?

💼 Follow my LinkedIn page for more work-life balancing and Coding tips.

🌐 LinkedIn: Oluwakemi Oluwadahunsi

kodemaven-portfolio.vercel.app