HABITRACKER

MINI PROJECT REPORT

Submitted by

AKSSHAYA B (202009003)

AMIRTHAVARSHINI B (202009004)

BHUVANA S (202009008)

MANJU BALA S (202009026)

in

19AD652 –FULL STACK DEVELOPMENT LABORATORY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE

SIVAKASI

MAY 2023

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATASCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of **“AKSSHAYA B (202009003) , BHUVANA S (202009008), AMIRTHAVARSHINI B (202009004)** and **MANJU BALA S (202009026)”** for the mini project titled **“HABITRACKER ”** in 19AD652 –Full Stack Development Laboratory during the sixth semester, Jan 2023 – May 2023 under my supervision.

SIGNATURE

Dr. J. Angela Jennifa Sujana,
Asso. Professor (Senior Grade)& Head,
AI&DS Department,
Mepco Schlenk Engg. College, Sivakasi

SIGNATURE

Dr. J. Angela Jennifa Sujana,
Asso. Professor (Senior Grade)& Head,
AI&DS Department
Mepco Schlenk Engg. College, Sivakasi

ABSTRACT

Habit Tracker is one of a system of goal tracking tool to track daily goals and build powerful habits. In addition, it can establish the target result to be achieved with a user-defined view to indicate consistency that has been made. It is also can measure and track anything and everything that people want to set their goals either in short terms or long terms.

Habit Tracker System can test the user by help themselves to be more discipline in order to keep track day to day habits and routines.

The main features of the Habitracker includes several functionalities like, Todo list, Timer, Tally, Mood tracking, Expenses and sleep.

With the help of the Habitracker, our daily activities can be monitored, our daily expenses for what we had bought and mood can also be tracked.

ACKNOWLEDGEMENT

We would like to express our special thanks to our project guide **Dr. J. Angela Jennifa Sujana ,** Head of the Department , Artificial Intelligence and Data Science who gave us the opportunity to do this wonderful project on the topic **“HABITRACKER”** , which also helped us in doing a lot of research and we came to know about so many things .

We are really thankful to them.

Secondly , we would also like to thank our parents and staffs who helped us a lot in finishing this project within the limited time.

We were making this project not only for marks but also to gain knowledge.

Thanks all again who helped us.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Time management is the process of organizing and planning how to divide the time between specific activities effectively. Sub-themes 1. Dealing with procrastination 2. Setting realistic goals 3. Setting priorities Steps to manage time effectively: Practice setting SMART (Specific, Measurable, Attainable, Relevant, and Time-bound) goals. a. Practice identifying your priority tasks b. Avoid multi-tasking c. Reduce distractions d. Practice your ability to beat procrastination e. Practice delegating tasks f. Taking regular breaks Outcome Students will become more focused towards their work and life. Learning good time management skill will give them buffer time at hand and reduce stress to accommodate more things that need to be done or which can be used to relax and give time to oneself.

1.2 NEED FOR THE SOLUTION

- To provide notes and motivation
- To track the mood of the user
- Automatically generate a cloud of your most common thoughts and feelings to get a better understanding of yourself.
- Find more about common conditions that can affect your mental health.

CHAPTER 2

LITERATURE REVIEW

- https://legacy.reactjs.org/docs/getting-started.html
- https://www.freecodecamp.org/news/javascript-skills-you-need-for-react-practical-examples/
- https://github.com/topics/habit-tracker
- https://github.com/Jorres/stress-relief

CHAPTER 3

IMPLEMENTATION

3.1 IMPLEMENTATION

- The habit tracker is developed using React.js and JavaScript.
- The main features of the mood tracker includes:
    - ➢ Todo
    - ➢ Timer
    - ➢ Mood tracking
    - ➢ Tally
    - ➢ Expenses
    - ➢ Sleep

- In Todo, we can add on our task for the day and delete it if we had completed it.
- Timer is used to set the timer for a specific task.
- Based on our presence of mind that is entered, our mood will be detected using the Mood tracker.
- By using Expenses, we can calculate what we had spent for the day and on which item.

3.2 TOOLS USED

- **React.js:**
    - ➢ React.js is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies.
- **JavaScript:**
    - ➢ JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat.

CHAPTER 4

CODING

4.1. CODE

<u>Index.js</u>

```
import { createRoot } from "react-dom/client";
import Main from "./components/Main";
createRoot(document.getElementById("root")).render(<Main />);
```

<u>Main.jsx</u>

```
import React from "react";
import Timer from "./bases/Timer";
import Tasks from "./bases/Tasks";
import LeaveScreen from "./bases/LeaveScreen";
import Tally from "./bases/Tally";
import Expenses from "./bases/Expenses";
import Mood from "./bases/Mood";
import NavBar from "./bases/NavBar";
import Dashboard from "./bases/Dashboard";

export default function Main() {
  const [ActiveTab, setActiveTab] = React.useState("Tasks");
  const handleTabChange = (tab) => {
    setActiveTab(tab);
  };
  return (
    <div className="main">
      <NavBar
        handleChange={handleTabChange}
        className="navbar"
        activeTab={ActiveTab}
      />
      <hr />
      <div className="main-body">
        <div className={ActiveTab === "Dashboard" ? "hide-me" : "show-me"}>
          <Dashboard />
        </div>
        <div className={ActiveTab === "Timer" ? "show-me" : "hide-me"}>
          <Timer />
        </div>
        <div className={ActiveTab === "Tasks" ? "show-me" : "hide-me"}>
          <Tasks />
        </div>
        <div className={ActiveTab === "Tally" ? "show-me" : "hide-me"}>
          <Tally />
        </div>
        <div className={ActiveTab === "Expenses" ? "show-me" : "hide-me"}>
          <Expenses />
        </div>
        <div className={ActiveTab === "Mood" ? "show-me" : "hide-me"}>
          <Mood />
        </div>
        <div className={ActiveTab === "LeaveScreen" ? "show-me" : "hide-me"}>
          <LeaveScreen />
        </div>
      </div>
    </div>
  );
}
```
Dashboard.js
```
import React from "react";
export default function Dashboard() {
  return <div></div>;
}
```
Expenses.js
```
import React, { useEffect } from "react";
const axios = require("axios");

export default function Expenses() {
  // to be saved in backend
  // ##############################################################
  const [symbol, setSymbol] = React.useState("$");
  const [expenses, setExpenses] = React.useState([]);
  const [dates, setDates] = React.useState(["01-01-2020", "01-02-2020"]);
  // ##############################################################
  useEffect(() => {
    axios
      .get("http://localhost:3001/expenses")
      .then((response) => {
        setExpenses(response.data);
      })
      .catch((err) => {
        console.log("error");
      });
  }, []);
  const addExpense = (e) => {
    if (
      e.key === "Enter" &&
      e.target.value !== "" &&
      e.target.value.slice(e.target.value.indexOf(" ")).length > 1 &&
      e.target.value.slice(1, e.target.value.indexOf(" ")) !== "" &&
      parseFloat(
        parseFloat(
          e.target.value.slice(symbol.length, e.target.value.indexOf(" ") + 1)
        ).toFixed(2)
      ) > 0
    ) {
      setExpenses([
        ...expenses,
        {
          id: expenses.length + 2,
          name: e.target.value.slice(e.target.value.indexOf(" ") + 1),
          amount: parseFloat(
            parseFloat(
              e.target.value.slice(
                symbol.length,
                e.target.value.indexOf(" ") + 1
              )
            ).toFixed(2)
          ),
          category: "other",
          date: `${new Date().getDate().toString().padStart(2, "0")}-${(
            new Date().getMonth() + 1
          )
            .toString()
            .padStart(2, "0")}-${new Date().getFullYear()}`,
        },
      ]);
      if (
        !dates.includes(
          `${new Date().getDate().toString().padStart(2, "0")}-${(
            new Date().getMonth() + 1
          )
            .toString()
            .padStart(2, "0")}-${new Date().getFullYear()}`
        )
      ) {
        setDates([
          ...dates,
          `${new Date().getDate().toString().padStart(2, "0")}-${(
            new Date().getMonth() + 1
          )
            .toString()
            .padStart(2, "0")}-${new Date().getFullYear()}`,
        ]);
      }
    }
  };
  const returnTotalOfDate = (date) => {
    let total = 0;
    expenses.forEach((expense) => {
      if (expense.date === date) {
        total += expense.amount;
      }
    });
    return total;
  };
  const deleteExpense = (id) => {
    setExpenses(expenses.filter((expense) => expense.id !== id));
  };
  const renameExpense = (e, expenseid) => {
    if (
      e.key === "Enter" &&
      e.target.value !== "" &&
      e.target.value.slice(e.target.value.indexOf(" ")).length > 1 &&
      e.target.value.slice(1, e.target.value.indexOf(" ")) !== "" &&
      parseFloat(
        parseFloat(
          e.target.value.slice(symbol.length, e.target.value.indexOf(" ") + 1)
        ).toFixed(2)
      ) > 0
    ) {
      console.log(
        parseFloat(
          parseFloat(
            e.target.value.slice(symbol.length, e.target.value.indexOf(" ") + 1)
          ).toFixed(2)
        )
      );
      setExpenses(
        expenses.map((expense) => {
          expenseid === expense.id &&
            (expense.amount = parseFloat(
              parseFloat(
                e.target.value.slice(
                  symbol.length,
                  e.target.value.indexOf(" ") + 1
                )
              ).toFixed(2)
            ));
          expenseid === expense.id &&
            (expense.name = e.target.value.slice(
            e.target.value.indexOf(" ") + 1
          ));
          return expense;
        })
      );
    }
  };
  const returnExpensesOfDate = (date) => {
    return expenses.filter((expense) => expense.date === date);
  };
  return (
    <div>
      <h1>Expenses.</h1>
      <div>
        <input
          type="text"
          placeholder={"Set Currency..."}
          maxLength="3"
          onKeyDown={(e) => {
            if (e.key === "Enter" && e.target.value !== "") {
              setSymbol(e.target.value);
            }
          }}
        />
        <br />
        <input
          onFocus={(e) =>
            e.target.value === "" ? (e.target.value = symbol) : null
          }
          onBlur={(e) => (e.target.value = "")}
          type={"text"}
          onKeyDown={(e) => addExpense(e)}
          placeholder={symbol + "2000 elon musk"}
        />
      </div>
      {expenses.length === 0 && (
        <div>
          <h2>No Expenses</h2>
        </div>
      )}
      <h2>Expenses by date:</h2>
      {dates.map((date) =>
        returnExpensesOfDate(date).length > 0 ? (
          <div key={date}>
            <h3>{date}</h3>
            <div>
              <table>
                {returnExpensesOfDate(date).map((expense) => (
                  <tbody key={expense.id}>
                    <tr key={date.indexOf(expense.date)}>
                      <td>{symbol}</td>
                      <td>{expense.amount}</td>
                      <td>{expense.name}</td>
                      <td>
                        <button onClick={() => deleteExpense(expense.id)}>
                          Delete
                        </button>
                        <input
                          onFocus={(e) =>
                            e.target.value === ""
                              ? (e.target.value = symbol + expense.amount + " ")
                              : null
                          }
                          onBlur={(e) => (e.target.value = "")}
                          type={"text"}
                          onKeyDown={(e) => renameExpense(e, expense.id)}
                          placeholder={
                            symbol + expense.amount + " " + expense.name
                          }
                        />
                      </td>
                    </tr>
                  </tbody>
                ))}
                <tfoot>
                  <tr>
                    <td style={{ borderTop: "1px solid black" }}>{symbol}</td>
                    <td style={{ borderTop: "1px solid black" }}>
                      {returnTotalOfDate(date)}
                    </td>
                    <td style={{ borderTop: "1px solid black" }}>Total</td>
                  </tr>
                </tfoot>
              </table>
            </div>
          </div>
        ) : null
      )}
    </div>
  );
}
```

LeaveScreen.js

```
import React, { Component } from "react";

class LeaveScreen extends Component {
  state = {
    liveHours: new Date().getHours().toString().padStart(2, "0"),
    liveMinutes: new Date().getMinutes().toString().padStart(2, "0"),
    liveSeconds: new Date().getSeconds().toString().padStart(2, "0"),
    Day: new Date().getDay().toString(),
    Date: new Date().getDate().toString().padStart(2, "0"),
    Month: new Date().getMonth().toString().padStart(2, "0"),
    Year: new Date().getFullYear(),
  };
  componentDidMount() {
    this.interval = setInterval(() => {
      this.setState({
        liveHours: new Date().getHours().toString().padStart(2, "0"),
        liveMinutes: new Date().getMinutes().toString().padStart(2, "0"),
        liveSeconds: new Date().getSeconds().toString().padStart(2, "0"),
        Date: new Date().getDate().toString().padStart(2, "0"),
        Month: new Date().getMonth().toString().padStart(2, "0"),
        Year: new Date().getFullYear(),
      });
      switch (new Date().getDay()) {
        default:
          this.setState({ Day: "Error!" });
          break;
        case 0:
          this.setState({ Day: "Sun" });
          break;
        case 1:
          this.setState({ Day: "Mon" });
          break;
        case 2:
          this.setState({ Day: "Tue" });
          break;
        case 3:
          this.setState({ Day: "Wed" });
          break;
        case 4:
          this.setState({ Day: "Thu" });
          break;
        case 5:
          this.setState({ Day: "Fri" });
          break;
        case 6:
          this.setState({ Day: "Sat" });
          break;
      }
    }, 500);
  }
  render() {
    return (
      <div>
        <div className="live-clock-face">
          <span className="live-clock-hours">{this.state.liveHours}:</span>
          <span className="live-clock-minutes">{this.state.liveMinutes}</span>
          <span className="live-clock-seconds">{this.state.liveSeconds}</span>
        </div>
        <br />
        {this.state.Day}, {this.state.Date}/{this.state.Month}/{this.state.Year}
      </div>
    );
  }
}
export default LeaveScreen;
```
Mood.jsx
```
import React, { useEffect } from "react";
import axios from "axios";
// add a password manager
export default function Mood() {
  // to be saved in backend
  // ##############################################################
  const [moods, setMoods] = React.useState([]);
  const [dates, setDates] = React.useState(["01-01-2020", "02-01-2020"]);
  // ##############################################################
  useEffect(() => {
    axios
      .get("http://localhost:3001/moods")
      .then((response) => {
        setMoods(response.data);
      })
      .catch((err) => {
        console.log("error");
      });
  }, []);
  const returnMoodOfDate = (date) => {
    return moods.filter((mood) => mood.date === date);
  };
  const addMood = (e) => {
    if (
      !dates.includes(
        `${new Date().getDate().toString().padStart(2, "0")}-${(
          new Date().getMonth() + 1
        )
```

```
            .toString()
            .padStart(2, "0")}-${new Date().getFullYear()}`
        )
      ) {
        setMoods([
          ...moods,
          {
            id: moods.length + 1,
            mood: e,
            date: `${new Date().getDate().toString().padStart(2, "0")}-${(
              new Date().getMonth() + 1
            )
              .toString()
              .padStart(2, "0")}-${new Date().getFullYear()}`,
          },
        ]);
        setDates([
          ...dates,
          `${new Date().getDate().toString().padStart(2, "0")}-${(
            new Date().getMonth() + 1
          )
            .toString()
            .padStart(2, "0")}-${new Date().getFullYear()}`,
        ]);
      } else {
        alert("You already added a mood for today, rather reset it!");
      }
    };
    const setMood = (e, mood) => {
      setMoods(
        moods.map((m) => {
          if (m.id === mood.id) {
            return {
              ...m,
              mood: e,
            };
          }
          return m;
        })
      );
    };
    const lastSeven = () => {
      const lastSeven = moods.slice(-7).map((mood) => {
        return [mood.mood];
      });
      let points = 0;
      for (let i = 0; i < lastSeven.length; i++) {
        lastSeven[i][0] === "Happy Pro Max" && (points += 5);
        lastSeven[i][0] === "Happy Lite" && (points += 3);
        lastSeven[i][0] === "IDK" && (points += 1);
        lastSeven[i][0] === "ANGRY!!!" && (points -= 5);
        lastSeven[i][0] === "Sed Lyf" && (points -= 3);
        lastSeven[i][0] === "Sad" && (points -= 2);
        lastSeven[i][0] === "Zzz..." && (points += 0);
      }
      let avgPoints = points / lastSeven.length;
      let avgMood = "";
      avgPoints.toFixed(2) < -1 &&
        avgPoints.toFixed(2) >= -2 &&
        (avgMood = "Sad");
      avgPoints.toFixed(2) < -2 &&
        avgPoints.toFixed(2) >= -3 &&
        (avgMood = "Sed Lyf");
      avgPoints.toFixed(2) < -3 && (avgMood = "ANGRY!!!");
      avgPoints.toFixed(2) >= -1 &&
        avgPoints.toFixed(2) <= 1.5 &&
        (avgMood = "IDK");
      avgPoints.toFixed(2) > 1.5 &&
        avgPoints.toFixed(2) <= 3 &&
        (avgMood = "Happy Lite");
      avgPoints.toFixed(2) > 3 && (avgMood = "Happy Pro Max");
      return (
        <p>
          Total points:
          {" " + points}
          <br />
          Average:
          {" " + avgPoints.toFixed(0)}
          <br />
          Average Mood:
          {" " + avgMood}
        </p>
      );
    };
    return (
      <div>
        <h1>Mood.</h1>
        <div>
          <h2>How is your mood today?</h2>
          {moods.length === 0 && <p>No moods yet.</p>}
          <button onClick={() => addMood("Happy Pro Max")}>Happy Pro Max</button>
          <button onClick={() => addMood("Happy Lite")}>Happy Lite</button>
          <button onClick={() => addMood("IDK")}>IDK</button>
          <button onClick={() => addMood("ANGRY!!!")}>ANGRY!!!</button>
          <button onClick={() => addMood("Sed Lyf")}>Sed Lyf</button>
          <button onClick={() => addMood("Sad")}>Sad</button>
          <button onClick={() => addMood("Zzz...")}>Zzz...</button>
          {dates.map((date) => {
            return (
              <div key={date}>
                <div>
                  <p>
                    {date + " "}
                    {returnMoodOfDate(date).map((mood) => {
                      return (
                        <font key={mood.id}>
                          {mood.mood}
                          <button onClick={() => setMood("Happy Pro Max", mood)}>
                            Happy Pro Max
                          </button>
                          <button onClick={() => setMood("Happy Lite", mood)}>
                            Happy Lite
                          </button>
                          <button onClick={() => setMood("IDK", mood)}>
                            IDK
                          </button>
                          <button onClick={() => setMood("ANGRY!!!", mood)}>
                            ANGRY!!!
                          </button>
                          <button onClick={() => setMood("Sed Lyf", mood)}>
                            Sed Lyf
                          </button>
                          <button onClick={() => setMood("Sad", mood)}>
                            Sad
                          </button>
                          <button onClick={() => setMood("Zzz...", mood)}>
                            Zzz...
                          </button>
                        </font>
                      );
                    })}
                  </p>
                </div>
              </div>
            );
          })}
          <div>
            <h2>Average mood in last 7 entries:</h2>
            <div>
              <div>{lastSeven()}</div>
            </div>
          </div>
        </div>
      </div>
    );
  }
```

NavBar.jsx

```
import React from "react";

export default function NavBar(props) {
  return (
    <div>
      <div className="header-logo">
        <svg
          width="128"
          height="128"
          x="0"
          y="0"
          viewBox="0 0 24 24"
          style={{ enableBackground: "new 0 0 512 512" }}
          className="logo"
        >
          <g>
            <path
              d="M18.314,7.548l-2.346-1.4.022-3.24A1.5,1.5,0,0,0,15.5,0h-
7A1.5,1.5,0,0,0,8,2.908v3.22L5.653,7.549A5.531,5.531,0,0,0,3,12.255V18.5A5.506,5.506,0,0,0,8.5,24h7A5.50
6,5.506,0,0,0,21,18.5V12.274A5.526,5.526,0,0,0,18.314,7.548ZM18,18.5A2.5,2.5,0,0,1,15.5,21h-
7A2.5,2.5,0,0,1,6,18.5V12.255a2.513,2.513,0,0,1,1.206-2.139l3.071-1.859A1.5,1.5,0,0,0,11,6.974V3h1.99l-
.028,3.99a1.5,1.5,0,0,0,.732,1.3l3.085,1.836A2.513,2.513,0,0,1,18,12.274Zm-2-3v1A1.5,1.5,0,0,1,14.5,18h-
5A1.5,1.5,0,0,1,8,16.5v-1A1.5,1.5,0,0,1,9.5,14h5A1.5,1.5,0,0,1,16,15.5Z"
              fill="#000000"
              data-original="#000000"
            ></path>
          </g>
        </svg>
      </div>
      <div className="header">
        <h1 className="header-heading">{props.activeTab}.</h1>
      </div>
      <div className="navbar">
        <span
          className="navbar-item"
          onClick={() => props.handleChange("Dashboard")}
        >
          <svg
            width="128"
            height="128"
            x="0"
            y="0"
            viewBox="0 0 24 24"
            style={{ enableBackground: "new 0 0 512 512" }}
          >
            <g>
              <path
                d="M16.773,2.9c-.747-.634-1.53-1.3-2.327-
2.024A3.354,3.354,0,0,0,11.731.031,3.264,3.264,0,0,0,9.4,1.516h0a18.708,18.708,0,0,0-
2.126,5.02,2.458,2.458,0,0,0-3.795-.163,9.159,9.159,0,0,0-
2.61,6.493A11.024,11.024,0,0,0,9.133,23.64,11.667,11.667,0,0,0,11.99,24,11.122,11.122,0,0,0,23.127,12.87
4C23.127,8.3,20.183,5.8,16.773,2.9Zm.218,16.386a7.974,7.974,0,0,1-.9.6,4.486,4.486,0,0,0,.44-1.919c0-
1.927-1.343-4.164-3-6.125a2.025,2.025,0,0,0-3.079,0C8.1,14.6,6.627,17.582,8,20a8.1,8.1,0,0,1-4.129-
7.127,5.951,5.951,0,0,1,1.345-3.9q.212.348.455.673a2.3,2.3,0,0,0,2.375.87,2.364,2.364,0,0,0,1.8-
1.826,17.255,17.255,0,0,1,2.091-
5.567A.259.259,0,0,1,12.132,3a.367.367,0,0,1,.295.093c.824.749,1.633,1.437,2.405,2.091,3.189,2.708,5.295
,4.5,5.295,7.686A8.081,8.081,0,0,1,16.991,19.288Z"
                fill="#000"
                data-original="#000000"
              ></path>
            </g>
          </svg>
          <div className="navbar-button-text">Today</div>
        </span>
        <span
          className="navbar-item"
          onClick={() => props.handleChange("Timer")}
        >
          <svg
            width="128"
            height="128"
            x="0"
            y="0"
            viewBox="0 0 24 24"
            style={{ enableBackground: "new 0 0 512 512" }}
          >
            <g>
              <path
                d="m16.632 24h-9.264a4.379 4.379 0 0 1 -3.319-1.523 4.307 4.307 0 0 1 -1-3.451 12.207
12.207 0 0 1 3.934-7.026 12.207 12.207 0 0 1 -3.935-7.025 4.307 4.307 0 0 1 1-3.451 4.379 4.379 0 0 1
3.32-1.524h9.264a4.378 4.378 0 0 1 3.318 1.523 4.3 4.3 0 0 1 1 3.448 12.235 12.235 0 0 1 -3.938 7.029
12.234 12.234 0 0 1 3.94 7.03 4.3 4.3 0 0 1 -1 3.447 4.378 4.378 0 0 1 -3.32 1.523zm0-21h-9.264a1.382
1.382 0 0 0 -1.046.48 1.3 1.3 0 0 0 -.307 1.048c.337 2.243 1.746 4.362 4.188 6.3a1.5 1.5 0 0 1 0 2.352c-
2.442 1.933-3.851 4.052-4.188 6.3a1.3 1.3 0 0 0 .307 1.048 1.382 1.382 0 0 0 1.046.472h9.264a1.384 1.384
0 0 0 1.046-.481 1.294 1.294 0 0 0 .307-1.044c-.335-2.236-1.745-4.355-4.191-6.3a1.5 1.5 0 0 1 0-
2.348c2.446-1.946 3.856-4.065 4.191-6.3a1.3 1.3 0 0 0 -.307-1.045 1.384 1.384 0 0 0 -1.046-.482z"
                fill="#000"
                data-original="#000000"
              ></path>
            </g>
          </svg>
          <div className="navbar-button-text">Timer</div>
        </span>
        <span
          className="navbar-item"
          onClick={() => props.handleChange("Tasks")}
        >
          <svg
            width="128"
            height="128"
            x="0"
            y="0"
            viewBox="0 0 24 24"
            style={{ enableBackground: "new 0 0 512 512" }}
          >
            <g>
              <path
                d="M4.5,7A3.477,3.477,0,0,1,2.025,5.975L.5,4.62a1.5,1.5,0,0,1,2-
2.24L4.084,3.794A.584.584,0,0,0,4.5,4a.5.5,0,0,0,.353-
.146L8.466.414a1.5,1.5,0,0,1,2.068,2.172L6.948,6A3.449,3.449,0,0,1,4.5,7ZM24,3.5A1.5,1.5,0,0,0,22.5,2h-
8a1.5,1.5,0,0,0,0,3h8A1.5,1.5,0,0,0,24,3.5ZM6.948,14l3.586-3.414A1.5,1.5,0,0,0,8.466,8.414l-
3.613,3.44a.5.5,0,0,1-
.707,0L2.561,10.268A1.5,1.5,0,0,0,.439,12.39l1.586,1.585A3.5,3.5,0,0,0,6.948,14ZM24,11.5A1.5,1.5,0,0,0,2
2.5,10h-8a1.5,1.5,0,0,0,0,3h8A1.5,1.5,0,0,0,24,11.5ZM6.948,22l3.586-3.414a1.5,1.5,0,0,0-2.068-2.172l-
3.613,3.44A.5.5,0,0,1,4.5,20a.584.584,0,0,1-.416-.206L2.5,18.38a1.5,1.5,0,0,0-
2,2.24l1.523,1.355A3.5,3.5,0,0,0,6.948,22ZM24,19.5A1.5,1.5,0,0,0,22.5,18h-
8a1.5,1.5,0,0,0,0,3h8A1.5,1.5,0,0,0,24,19.5Z"
                fill="#000"
                data-original="#000000"
              ></path>
            </g>
          </svg>
          <div className="navbar-button-text">Todo</div>
        </span>
        <span
          className="navbar-item"
          onClick={() => props.handleChange("Tally")}
        >
          <svg
            width="128"
            height="128"
            x="0"
            y="0"
            viewBox="0 0 24 24"
            style={{ enableBackground: "new 0 0 512 512" }}
          >
            <g>
              <path
                d="M23.871,6.733c-.336-.758-1.225-1.1-1.98-.763l-.891,.395V1.5c0-.829-.672-1.5-1.5-1.5s-
1.5,.671-1.5,1.5V7.697l-2,.888V1.5c0-.829-.672-1.5-1.5-1.5s-1.5,.671-1.5,1.5V9.916l-2,.888V1.5c0-.829-
.671-1.5-1.5-1.5s-1.5,.671-1.5,1.5V12.134l-2,.888V1.5c0-.829-.671-1.5-1.5-1.5s-1.5,.671-1.5,1.5V14.353l-
2.108,.935c-.757,.336-1.099,1.223-.763,1.979,.249,.56,.797,.892,1.372,.892,.203,0,.41-.041,.608-
.129l.891-.395v4.865c0,.828,.671,1.5,1.5,1.5s1.5-.672,1.5-1.5v-6.196l2-
.888v7.083c0,.828,.671,1.5,1.5,1.5s1.5-.672,1.5-1.5V14.085l2-.888v9.302c0,.828,.672,1.5,1.5,1.5s1.5-
.672,1.5-1.5V11.867l2-.888v11.521c0,.828,.672,1.5,1.5,1.5s1.5-.672,1.5-1.5V9.648l2.108-.935c.758-
.336,1.099-1.223,.763-1.979Z"
                fill="#000"
                data-original="#000000"
              ></path>
            </g>
          </svg>
          <div className="navbar-button-text">Plus-Minus</div>
        </span>
        <span
          className="navbar-item"
          onClick={() => props.handleChange("Expenses")}
        >
          <svg
            width="128"
            height="128"
            x="0"
            y="0"
            viewBox="0 0 24 24"
            style={{ enableBackground: "new 0 0 512 512" }}
          >
            <g>
              <path
                d="M16.771,22c-1.637,0-3.111-.409-4.537-.805-1.289-.357-2.506-.695-3.733-.695-1.364,0-
2.142,.146-2.954,.38-1.316,.38-2.703,.124-3.801-.703-1.109-.834-1.745-2.108-1.745-
3.495V7.749C0,5.556,1.381,3.555,3.437,2.771c1.343-.512,2.618-.771,3.791-
.771,1.636,0,3.11,.409,4.537,.805,1.29,.357,2.508,.695,3.735,.695,1.362,0,2.14-.145,2.951-.379,1.315-
.382,2.702-.124,3.802,.703,1.109,.834,1.745,2.108,1.745,3.495v8.933h0c0,2.193-1.382,4.194-3.438,4.978-
1.344,.512-2.619,.771-3.791,.771Zm-8.271-
4.5c1.636,0,3.11,.409,4.535,.805,1.29,.357,2.507,.695,3.735,.695,.805,0,1.721-.193,2.723-.575,.915-
.349,1.506-1.202,1.506-2.174V7.318c0-.435-.201-.835-.549-1.097-.339-.255-.768-.335-1.167-.218-
1.224,.353-2.319,.497-3.783,.497-1.636,0-3.11-.409-4.537-.805-1.29-.357-2.508-.695-3.735-.695-.806,0-
1.722,.193-2.723,.575-.915,.349-1.506,1.202-
1.506,2.174v8.933c0,.436,.201,.835,.549,1.098,.338,.254,.764,.335,1.166,.217,1.225-.353,2.321-
.497,3.786-.497Zm5.5-3v-5c0-.607-.365-1.154-.926-1.386-.561-.233-1.205-.104-1.635,.325l-2,2c-.586,.585-
.586,1.536,0,2.121,.422,.421,1.032,.541,1.561,.354v1.585c0,.829,.672,1.5,1.5,1.5s1.5-.671,1.5-
1.5ZM5.5,7c-.828,0-1.5,.672-1.5,1.5s.672,1.5,1.5,1.5,1.5-.672,1.5-1.5-.672-1.5-1.5-
1.5Zm11.5,2.5c0,.828,.672,1.5,1.5,1.5s1.5-.672,1.5-1.5-.672-1.5-1.5-1.5-1.5,.672-1.5,1.5Zm-11.5,3.5c-
.828,0-1.5,.672-1.5,1.5s.672,1.5,1.5,1.5,1.5-.672,1.5-1.5-.672-1.5-1.5-
1.5Zm11.5,2.5c0,.828,.672,1.5,1.5,1.5s1.5-.672,1.5-1.5-.672-1.5-1.5-1.5-1.5,.672-1.5,1.5Z"
                fill="#000"
                data-original="#000000"
              ></path>
            </g>
          </svg>
          <div className="navbar-button-text">Expenses</div>
        </span>
        <span
          className="navbar-item"
          onClick={() => props.handleChange("Mood")}
        >
          <svg
            width="128"
            height="128"
            x="0"
            y="0"
            viewBox="0 0 24 24"
            style={{ enableBackground: "new 0 0 512 512" }}
          >
            <g>
              <path

d="M8.5,7A2.587,2.587,0,0,1,11,9.667C11,11,9.881,11,8.5,11S6,11,6,9.667A2.587,2.587,0,0,1,8.5,7ZM13,9.66
7C13,11,14.119,11,15.5,11S18,11,18,9.667A2.587,2.587,0,0,0,15.5,7,2.587,2.587,0,0,0,13,9.667ZM24,12A12,1
2,0,1,0,12,24,12.013,12.013,0,0,0,24,12Zm-3,0a9,9,0,1,1-9-9A9.011,9.011,0,0,1,21,12Zm-5.5,1h-
7a1.468,1.468,0,0,0-1.268,2.2A5.865,5.865,0,0,0,12,18a5.865,5.865,0,0,0,4.767-
2.8A1.468,1.468,0,0,0,15.5,13Z"
                fill="#000"
                data-original="#000000"
              ></path>
            </g>
          </svg>
          <div className="navbar-button-text">Moods</div>
        </span>
        <span
          className="navbar-item"
          onClick={() => props.handleChange("LeaveScreen")}
        >
          <svg
            width="128"
            height="128"
            x="0"
            y="0"
            viewBox="0 0 509.348 509.348"
            style={{ enableBackground: "new 0 0 512 512" }}
          >
            <g>
              <g>
                <path

d="M488.935,188.541C437.397,109.024,349.407,60.662,254.652,59.773C159.898,60.662,71.908,109.024,20.37,18
8.541   c-27.16,39.859-27.16,92.279,0,132.139c51.509,79.566,139.504,127.978,234.283,128.896   c94.754-
0.889,182.744-49.251,234.283-128.768C516.153,280.919,516.153,228.429,488.935,188.541z M436.199,284.541
c-39.348,62.411-107.769,100.488-181.547,101.035c-73.777-0.546-142.198-38.624-181.547-101.035   c-12.267-
18.022-12.267-41.712,0-59.733c39.348-62.411,107.769-100.488,181.547-101.035
c73.777,0.546,142.198,38.624,181.547,101.035C448.466,242.829,448.466,266.519,436.199,284.541z"
                  fill="#000"
                  data-original="#000000"
                ></path>
                <circle
                  cx="254.652"
                  cy="254.674"
                  r="85.333"
                  fill="#000"
                  data-original="#000000"
                ></circle>
              </g>
            </g>
          </svg>
          <div className="navbar-button-text">Sleep?</div>
        </span>
      </div>
    </div>
  );
}
```

Tally.jsx

```
import React, { useEffect } from "react";
import axios from "axios";

export default function Tally() {
  // to be saved in backend
  // ##############################################################
  const [tallies, setTallies] = React.useState([]);
  const [categories, setCategories] = React.useState(["other", "work", "home"]);
  // ##############################################################
  useEffect(() => {
    axios
      .get("http://localhost:3001/tallies")
      .then((response) => {
        setTallies(response.data);
      })
      .catch((err) => {
        console.log("error");
      });
  }, []);
  const minusCount = (id) => {
    setTallies(
      tallies.map((tally) => {
        if (tally.id === id) {
          tally.val > 0 && tally.val--;
        }
        return tally;
      })
    );
  };
  const deleteTally = (id) => {
    setTallies(tallies.filter((tally) => tally.id !== id));
  };
  const addCount = (id) => {
    setTallies(
      tallies.map((tally) => {
        if (tally.id === id) {
          tally.val++;
        }
        return tally;
      })
    );
  };
  const newTally = (e) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTallies([
        ...tallies,
        {
          id: tallies.length + 1,
          category: categories[0],
          val: 0,
          name: e.target.value,
        },
      ]);
    }
  };
  const addCategory = (e) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setCategories([...categories, e.target.value]);
    }
  };
  const deleteCategory = (category) => {
    setCategories(categories.filter((cat) => cat !== category));
    setTallies(
      tallies.map((tally) => {
        if (tally.category === category) {
          tally.category = "other";
        }
        return tally;
      })
    );
  };
  const returnTallyOfCategory = (category) => {
    return tallies.filter((tally) => tally.category === category);
  };
  const addTallyOfCategory = (e, category) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTallies([
        ...tallies,
        {
          id: tallies.length + 1,
          category: category,
          val: 0,
          name: e.target.value,
        },
      ]);
    }
  };
  const moveCategory = (id, category) => {
    setTallies(
      tallies.map((tally) => {
        if (tally.id === id) {
          tally.category = category;
        }
        return tally;
      })
    );
  };
  const renameTally = (e, id) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTallies(
        tallies.map((tally) => {
          if (tally.id === id) {
            tally.name = e.target.value;
          }
          return tally;
        })
      );
    }
  };
  return (
    <div>
      <h1>Tally Counter.</h1>
      <div>
        <h2>All Tallies:</h2>
        <input type="text" onKeyDown={newTally} placeholder="Add tally." />
        <ul>
          {tallies.map((tally) => (
            <li key={tally.id}>
              <button onClick={() => minusCount(tally.id)}>-</button>
              {tally.val}
              <button onClick={() => addCount(tally.id)}>+</button>
              <button onClick={() => deleteTally(tally.id)}>Delete</button>
              {tally.name}
              <input
                type="text"
                onKeyDown={(e) => renameTally(e, tally.id)}
                placeholder="Rename..."
              />
              <select
                onChange={(e) => moveCategory(tally.id, e.target.value)}
                value={tally.category}
              >
                {categories.map((category) => (
                  <option key={category} value={category}>
                    {category}
                  </option>
                ))}
              </select>
            </li>
          ))}
        </ul>
      </div>
      <div>
        <h2>Tallies of Category:</h2>
        {categories.map((category) => (
          <div key={category}>
            <h3>
              # {categories.indexOf(category) + 1} | {category} | (
              {returnTallyOfCategory(category).length} Tallies.){" "}
            </h3>
            {category !== categories.slice(0, 1)[0] && (
              <div>
                <button onClick={() => deleteCategory(category)}>Delete</button>
                <input
                  type="text"
                  onKeyDown={(e) => addTallyOfCategory(e, category)}
                  placeholder="Add tally."
                />
              </div>
            )}
            <p>
              {returnTallyOfCategory(category).length === 0 &&
                "You dont have any tallies in this category."}
            </p>
            <ul>
              {returnTallyOfCategory(category).map((tally) => (
                <li key={tally.id}>
                  <button onClick={() => minusCount(tally.id)}>-</button>
                  {tally.val}
                  <button onClick={() => addCount(tally.id)}>+</button>
                  <button onClick={() => deleteTally(tally.id)}>Delete</button>
                  {tally.name}
                  <input
                    type="text"
                    onKeyDown={(e) => renameTally(e, tally.id)}
                    placeholder="Rename..."
                  />
                  <select
                    onChange={(e) => moveCategory(tally.id, e.target.value)}
                    value={tally.category}
                  >
                    {categories.map((category) => (
                      <option key={category} value={category}>
                        {category}
                      </option>
                    ))}
                  </select>
                </li>
              ))}
            </ul>
          </div>
        ))}
      </div>
      <h3>
        Add Category:
        <br />
        <input
          type="text"
          onKeyDown={addCategory}
          placeholder="Add category."
        />
      </h3>
    </div>
  );
}
```
Tasks.jsx
```
import React, { useEffect } from "react";
const axios = require("axios");

export default function Tasks() {
  // to be saved in backend
  // ##############################################################
  const [tasks, setTasks] = React.useState([]);
  const [categories, setCategories] = React.useState(["other", "work", "home"]);
  // ##############################################################
  useEffect(() => {
    axios
      .get("http://localhost:3001/tasks")
      .then((response) => {
        setTasks(response.data);
      })
      .catch((err) => {
        console.log("error");
      });
  }, []);
  const addTask = (e) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTasks([
        ...tasks,
        {
          id: tasks.length + 1,
          category: categories[0],
          name: e.target.value,
          done: false,
          subTasks: [],
        },
      ]);
    }
  };
  const deleteTask = (id) => {
    setTasks(tasks.filter((task) => task.id !== id));
  };
  const renameTask = (e, taskid) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTasks(
        tasks.map((task) => {
          taskid === task.id && (task.name = e.target.value);
          return task;
        })
      );
    }
  };
  const doneTask = (id) => {
    setTasks(
      tasks.map((task) => {
        if (task.id === id) {
          task.done = !task.done;
        }
        return task;
      })
    );
  };
  const addSubTask = (e, taskid) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTasks(
        tasks.map((task) => {
          taskid === task.id &&
            (task.subTasks = [
              ...task.subTasks,
              {
                id: task.subTasks.length + 1,
                name: e.target.value,
                done: false,
              },
            ]);
          return task;
        })
      );
    }
  };
  const deleteSubTask = (id, e) => {
    setTasks(
      tasks.map((task) => {
        e === task.id &&
          (task.subTasks = task.subTasks.filter(
            (subTask) => subTask.id !== id
          ));
        return task;
      })
    );
  };
  const renameSubTask = (e, subtaskid, taskid) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTasks(
        tasks.map((task) => {
          if (task.id === taskid) {
            task.subTasks.map((subtask) => {
              if (subtask.id === subtaskid) {
                subtask.name = e.target.value;
              }
              return subtask;
            });
          }
          return task;
        })
      );
    }
  };
  const doneSubTask = (subtaskid, taskid) => {
    setTasks(
      tasks.map((task) => {
        if (task.id === taskid) {
          task.subTasks.map((subtask) => {
            if (subtask.id === subtaskid) {
              subtask.done = !subtask.done;
            }
            return subtask;
          });
        }
        return task;
      })
    );
  };
  const returnTaskOfCaterory = (category) => {
    return tasks.filter((task) => task.category === category);
  };
  const addTaskOfCategory = (e, category) => {
    if (e.key === "Enter" && e.target.value !== "") {
      setTasks([
        ...tasks,
        {
          id: tasks.length + 1,
          category: category,
          name: e.target.value,
          done: false,
          subTasks: [],
        },
      ]);
    }
  };
  const addCategory = (e) => {
    if (e.key === "Enter" && e.target.value !== "") {
      if (categories.indexOf(e.target.value) === -1) {
        setCategories([...categories, e.target.value]);
      } else {
        alert("Category already exists.");
      }
    }
  };
  const deleteCategory = (category) => {
    setCategories(categories.filter((c) => c !== category));
    setTasks(
      tasks.map((task) => {
        if (task.category === category) {
          task.category = "other";
        }
        return task;
      })
    );
  };
  const moveTask = (taskid, category) => {
    setTasks(
      tasks.map((task) => {
        if (task.id === taskid) {
          task.category = category;
        }
        return task;
      })
    );
  };
  return (
    <div>
      <h1>Tasks.</h1>
      <div>
        <h2>All Tasks | ({tasks.length} Tasks.)</h2>
        <input type="text" placeholder={"Add task..."} onKeyDown={addTask} />
        <br />
        <p>{tasks.length === 0 && "You don't have any tasks yet!"}</p>
        <ul type="disc">
          {tasks.map((task) => (
            <li key={task.id} style={{ opacity: task.done && 0.5 }}>
              <button onClick={() => doneTask(task.id)}>
                {task.done ? "Done!" : "To be done"}
              </button>
              <button onClick={() => deleteTask(task.id)}>Delete</button>
              <input
                type="text"
                onKeyDown={(e) => renameTask(e, task.id)}
                placeholder={"Rename..."}
              />
              <select
                onChange={(e) => moveTask(task.id, e.target.value)}
                value={task.category}
              >
                {categories.map((category) => (
                  <option key={category} value={category}>
                    {category}
                  </option>
                ))}
              </select>
              {task.name}
              <br />
              <input
                type="text"
                onKeyDown={(e) => addSubTask(e, task.id)}
                placeholder={"Add sub task..."}
              />
              <ul type="circle">
                {task.subTasks.map((subTask) => (
                  <li key={subTask.id} style={{ opacity: subTask.done && 0.5 }}>
                    <button onClick={() => doneSubTask(subTask.id, task.id)}>
                      {subTask.done ? "Done" : "To be done"}
                    </button>
                    <button onClick={() => deleteSubTask(subTask.id, task.id)}>
                      Delete
                    </button>
                    <input
                      type="text"
                      onKeyDown={(e) => renameSubTask(e, subTask.id, task.id)}
                      placeholder={"Rename..."}
                    />
                    {subTask.name}
                  </li>
                ))}
              </ul>
            </li>
          ))}
        </ul>
      </div>
      <div>
        <h2>Tasks of category:</h2>
        {categories.map((category) => (
          <div key={category}>
            <h3>
              # {categories.indexOf(category) + 1} | {category} | (
              {returnTaskOfCaterory(category).length} Tasks.)
            </h3>
            {category !== categories.slice(0, 1)[0] && (
              <div>
                <button onClick={() => deleteCategory(category)}>Delete</button>
                <input
                  type="text"
                  placeholder={"Add task in category..."}
                  onKeyDown={(e) => addTaskOfCategory(e, category)}
                />
              </div>
            )}
            <p>
              {returnTaskOfCaterory(category).length === 0 &&
                "You don't have any tasks in this category yet!"}
            </p>
            <ul type="disc">
              {returnTaskOfCaterory(category).map((task) => (
                <li key={task.id} style={{ opacity: task.done && 0.5 }}>
                  <button onClick={() => doneTask(task.id)}>
                    {task.done ? "Done!" : "To be done"}
                  </button>
                  <button onClick={() => deleteTask(task.id)}>Delete</button>
                  <input
                    type="text"
                    onKeyDown={(e) => renameTask(e, task.id)}
                    placeholder={"Rename..."}
                  />
                  <select
                    onChange={(e) => moveTask(task.id, e.target.value)}
                    value={task.category}
                  >
                    {categories.map((category) => (
                      <option key={category} value={category}>
                        {category}
                      </option>
                    ))}
                  </select>
                  {task.name}
                  <br />
                  <input
                    type="text"
                    onKeyDown={(e) => addSubTask(e, task.id)}
                    placeholder={"Add sub task..."}
                  />
                  <ul type="circle">
                    {task.subTasks.map((subTask) => (
                      <li
                        key={subTask.id}
                        style={{ opacity: subTask.done && 0.5 }}
                      >
                        <button
                          onClick={() => doneSubTask(subTask.id, task.id)}
                        >
                          {subTask.done ? "Done!" : "To be done"}
                        </button>
                        <button
                          onClick={() => deleteSubTask(subTask.id, task.id)}
                        >
                          Delete
                        </button>
                        <input
                          type="text"
                          onKeyDown={(e) =>
                            renameSubTask(e, subTask.id, task.id)
                          }
                          placeholder={"Rename..."}
                        />
                        {subTask.name}
                      </li>
                    ))}
                  </ul>
                </li>
              ))}
            </ul>
          </div>
        ))}
      </div>
      <h3>
        Add more categories:
        <br />
        <input
          type="text"
          placeholder={"Type category name here..."}
          onKeyDown={addCategory}
        />
      </h3>
      <div className="sidebar">
```

```
        <h2>Categories.</h2>
        <div className="sidebar-card-short">
          <h3>All</h3>
        </div>
        {categories.map((category) => (
          <div key={category} className="sidebar-card-short">
            <h3>
              # {categories.indexOf(category) + 1} | {category}
            </h3>
          </div>
        ))}
      </div>
    </div>
  );
}
```

Timer.jsx

```
import React from "react";

export default function Timer() {
  const [h, setH] = React.useState(1);
  const [m, setM] = React.useState(0);
  const [s, setS] = React.useState(0);
  const [started, setStarted] = React.useState(false);
  const [startTime, setStartTime] = React.useState(null);
  const [isFocus, setIsFocus] = React.useState(true);
  const [saveH, setSaveH] = React.useState(0);
  const [saveM, setSaveM] = React.useState(0);
  const [saveS, setSaveS] = React.useState(0);
  const [lastClick, setLastClick] = React.useState("Reset");
  // to be saved in backend
  const [totalTimeFocused, setTotalTimeFocused] = React.useState(0);
  const [totalTimeBreak, setTotalTimeBreak] = React.useState(0);
  const [lastHistory, setLastHistory] = React.useState([
    {
      date: new Date(),
      focus: 0,
      break: 0,
      history: [],
    },
  ]);
  // -
  const startTimer = () => {
    setStarted(true);
    setStartTime(new Date());
    lastHistory[0].date.getDate() === new Date().getDate()
      ? (lastHistory[0].history = [
          "(" +
            new Date().getHours().toString().padStart(2, "0") +
            ":" +
            new Date().getMinutes().toString().padStart(2, "0") +
            ") Started: " +
            saveH.toString().padStart(2, "0") +
            ":" +
            saveM.toString().padStart(2, "0") +
            ":" +
            saveS.toString().padStart(2, "0"),
          ...lastHistory[0].history,
        ])
      : setLastHistory([
          {
            date: new Date(),
            focus: 0,
          break: 0,
          history: [
            "(" +
              new Date().getHours().toString().padStart(2, "0") +
              ":" +
              new Date().getMinutes().toString().padStart(2, "0") +
              ") Started: " +
              saveH.toString().padStart(2, "0") +
              ":" +
              saveM.toString().padStart(2, "0") +
              ":" +
              saveS.toString().padStart(2, "0"),
          ],
        },
        ...lastHistory,
      ]);
  setLastClick("Start");
};
const continueTimer = () => {
  setStartTime(new Date());
  setStarted(true);
  lastHistory[0].history = [
    "(" +
      new Date().getHours().toString().padStart(2, "0") +
      ":" +
      new Date().getMinutes().toString().padStart(2, "0") +
      ") Resumed. ",
    ...lastHistory[0].history,
  ];
  setLastClick("Continue");
};
const stopTimer = () => {
  setStarted(false);
  isFocus
    ? setTotalTimeFocused(
        totalTimeFocused +
          (new Date().getTime() - startTime.getTime()) / 60000
      )
    : setTotalTimeBreak(
        totalTimeBreak + (new Date().getTime() - startTime.getTime()) / 60000
      );
  lastHistory[0].history = [
    "(" +
      new Date().getHours().toString().padStart(2, "0") +
      ":" +
      new Date().getMinutes().toString().padStart(2, "0") +
      ") Stopped. " +
      ((isFocus ? " (Focused: " : " (Break: ") +
        ((new Date() - startTime) / 60000).toFixed(0) +
        (((new Date() - startTime) / 60000).toFixed(0) === 1
          ? " min.)"
          : " mins.)")),
    ...lastHistory[0].history,
  ];
  setLastClick("Stop");
};
const resetTimer = () => {
  lastHistory[0].history = [
    "(" +
      new Date().getHours().toString().padStart(2, "0") +
      ":" +
      new Date().getMinutes().toString().padStart(2, "0") +
      ") Reset to: " +
      saveH.toString().padStart(2, "0") +
      ":" +
      saveM.toString().padStart(2, "0") +
      ":" +
      saveS.toString().padStart(2, "0"),
    ...lastHistory[0].history,
  ];
  setH(saveH);
  setM(saveM);
  setS(saveS);
  setStarted(false);
  setLastClick("Reset");
};
React.useEffect(() => {
  if (started) {
    const interval = setInterval(() => {
      if (s === 0) {
        if (m === 0) {
          if (h === 0) {
            setH(0);
            setM(0);
            setS(0);
            setStarted(false);
            lastHistory[0].history = [
              "(" +
                new Date().getHours().toString().padStart(2, "0") +
                ":" +
                new Date().getMinutes().toString().padStart(2, "0") +
                ") Finished: " +
                saveH.toString().padStart(2, "0") +
                ":" +
                saveM.toString().padStart(2, "0") +
                ":" +
                saveS.toString().padStart(2, "0"),
              ...lastHistory[0].history,
            ];
            isFocus
              ? setTotalTimeFocused(
                  totalTimeFocused +
                    (new Date().getTime() - startTime.getTime()) / 60000
                )
              : setTotalTimeBreak(
                  totalTimeBreak +
                    (new Date().getTime() - startTime.getTime()) / 60000
                );
            setLastClick("Ended");
          } else {
            setH(h - 1);
            setM(59);
            setS(59);
          }
        } else {
          setM(m - 1);
          setS(59);
        }
      } else {
        setS(s - 1);
      }
    }, 1000);
    return () => clearInterval(interval);
  }
```

```
}, [
  started,
  h,
  m,
  s,
  saveH,
  saveM,
  saveS,
  isFocus,
  startTime,
  lastHistory,
  totalTimeBreak,
  totalTimeFocused,
]);
if (h < 0) {
  setH(0);
  setM(0);
  setS(0);
}
if (m < 0) {
  setM(m + 60);
  setH(h - 1);
}
if (s < 0) {
  setS(s + 60);
  setM(m - 1);
}
if (m > 59) {
  setM(m - 60);
  setH(h + 1);
}
if (s > 59) {
  setS(s - 60);
  setM(m + 1);
}
const setTime = (hr, min, sec) => {
  setH(hr);
  setM(min);
  setS(sec);
  setSaveH(hr);
  setSaveM(min);
  setSaveS(sec);
  setLastClick("Reset");
};
const randomKaomoji = () => {
  const kaomoji = [
    "(っ °Д °;)っ",
    "ヽ(´▽`)/",
    "(▀̿Ĺ̯▀̿ ̿)",
    "ヽ(゜Д゜)ﾉ",
    "Σ(⊙ω⊙ノ)/",
    "Σ(O_O)",
    "w(゜Д゜)w",
    "(#°Д°)",
    "(ʘ ͜ʖ ʘ)",
    "╭つ ◕_◕ ╮つ",
    "(ﾉ◕ヮ◕)ﾉ",
    "━Σ(゜Д゜|||)━",
    "?(☻○ ☻)╯",
    "lim ∆t → 0",
  ];
```

```
    kaomoji.reverse();
    return kaomoji[Math.floor(Math.random() * kaomoji.length)];
  };
  return (
    <div>
      <h2
        className={
          lastClick === "Ended" || lastClick === "Stop"
            ? "main-look-timer timer-blink"
            : lastClick === "Reset"
            ? "alt-look-timer"
            : "main-look-timer"
        }
      >
        {lastClick === "Reset"
          ? h.toString().padStart(2, "0") === "00"
            ? m.toString().padStart(2, "0") === "00"
              ? s.toString().padStart(2, "0") === "00" &&
                m.toString().padStart(2, "0") === "00" &&
                h.toString().padStart(2, "0") === "00"
                ? randomKaomoji()
                : s.toString().padStart(2, "0")
              : m.toString().padStart(2, "0") +
                ":" +
                s.toString().padStart(2, "0")
            : h.toString().padStart(2, "0") +
              ":" +
              m.toString().padStart(2, "0")
          : h.toString().padStart(2, "0") === "00"
          ? m.toString().padStart(2, "0") === "00"
            ? s.toString().padStart(2, "0") === "00" &&
              m.toString().padStart(2, "0") === "00" &&
              h.toString().padStart(2, "0") === "00"
              ? "End."
              : s.toString().padStart(2, "0")
            : m.toString().padStart(2, "0") +
              ":" +
              s.toString().padStart(2, "0")
          : h.toString().padStart(2, "0") + ":" + m.toString().padStart(2, "0")}
      </h2>
      <div className="timer-controls-container">
        {(lastClick === "Reset" || lastClick === "Ended") && (
          <div>
            <button
              onClick={startTimer}
              disabled={h === 0 && m === 0 && s === 0 && true}
            >
              <svg
                width="128"
                height="128"
                x="0"
                y="0"
                viewBox="0 0 24 24"
                style={{ enableBackground: "new 0 0 512 512" }}
              >
                <g>
                  <path
                    d="M20.463,7.713l-9.1-
6.677A5.317,5.317,0,0,0,2.9,5.323V18.677a5.311,5.311,0,0,0,8.46,4.287l9.105-6.677a5.315,5.315,0,0,0,0-
8.574Zm-1.774,6.155-9.1,6.677A2.317,2.317,0,0,1,5.9,18.677V5.323a2.276,2.276,0,0,1,1.27-
2.066A2.328,2.328,0,0,1,8.223,3a2.3,2.3,0,0,1,1.362.455l9.1,6.677a2.316,2.316,0,0,1,0,3.736Z"
                    fill="#00213a"
                    data-original="#000000"
                  ></path>
                </g>
              </svg>
            </button>
          </div>
        )}
        {(lastClick === "Start" || lastClick === "Continue") && (
          <div>
            <button onClick={stopTimer}>
              <svg
                width="128"
                height="128"
                x="0"
                y="0"
                viewBox="0 0 24 24"
                style={{ enableBackground: "new 0 0 512 512" }}
              >
                <g>
                  <path
                    d="M7,0A4,4,0,0,0,3,4V20a4,4,0,0,0,8,0V4A4,4,0,0,0,7,0ZM8,20a1,1,0,0,1-
2,0V4A1,1,0,0,1,8,4Z"
                    fill="#000305"
                    data-original="#000000"
                  ></path>
                  <path
                    d="M17,0a4,4,0,0,0-4,4V20a4,4,0,0,0,8,0V4A4,4,0,0,0,17,0Zm1,20a1,1,0,0,1-
2,0V4a1,1,0,0,1,2,0Z"
                    fill="#000305"
                    data-original="#000000"
                  ></path>
                </g>
              </svg>
            </button>
          </div>
        )}
        {lastClick === "Stop" && (
          <div>
            <button onClick={continueTimer}>
              <svg
                width="128"
                height="128"
                x="0"
                y="0"
                viewBox="0 0 24 24"
                style={{ enableBackground: "new 0 0 512 512" }}
              >
                <g>
                  <path
                    d="M20.463,7.713l-9.1-
6.677A5.317,5.317,0,0,0,2.9,5.323V18.677a5.311,5.311,0,0,0,8.46,4.287l9.105-6.677a5.315,5.315,0,0,0,0-
8.574Zm-1.774,6.155-9.1,6.677A2.317,2.317,0,0,1,5.9,18.677V5.323a2.276,2.276,0,0,1,1.27-
2.066A2.328,2.328,0,0,1,8.223,3a2.3,2.3,0,0,1,1.362.455l9.1,6.677a2.316,2.316,0,0,1,0,3.736Z"
                    fill="#00213a"
                    data-original="#000000"
                  ></path>
                </g>
              </svg>
            </button>
            <button onClick={resetTimer}>
              <svg
                width="128"
                height="128"
                x="0"
                y="0"
                viewBox="0 0 24 24"
                style={{ enableBackground: "new 0 0 512 512" }}
              >
                <g>
                  <path

d="M1.611,12c.759,0,1.375,.57,1.485,1.32,.641,4.339,4.389,7.68,8.903,7.68,5.476,0,9.827-4.917,8.867-
10.569-.453-2.665-2.148-5.023-4.523-6.313-3.506-1.903-7.48-1.253-
10.18,1.045l1.13,1.13c.63,.63,.184,1.707-.707,1.707H2c-.552,0-1-.448-1-1V2.414c0-.891,1.077-1.337,1.707-
.707l1.332,1.332C7.6-.115,12.921-1.068,17.637,1.408c3.32,1.743,5.664,5.027,6.223,8.735,1.122,7.437-
4.633,13.857-11.86,13.857-6.021,0-11.021-4.457-11.872-10.246-.135-.92,.553-1.754,1.483-1.754Z"
                    fill="#00223c"
                    data-original="#000000"
                  ></path>
                </g>
              </svg>
            </button>
          </div>
        )}
      </div>
      <br />
      <div
        className={
          lastClick === "Reset" || lastClick === "Ended"
            ? "timer-reco-visible"
            : "timer-reco-invisible"
        }
      >
        <button onClick={() => setTime(0, 30, 0)}>30 min</button>
        <button onClick={() => setTime(1, 0, 0)}>1 hr</button>
        <button onClick={() => setTime(2, 0, 0)}>2 hrs</button>
        <p>------</p>
        <button onClick={() => setTime(h + 1, m, s)}>+1 hr</button>
        <button onClick={() => setTime(h - 1, m, s)}>-1 hr</button>
        <p>------</p>
        <button onClick={() => setTime(h, m + 15, s)}>+15 min</button>
        <button onClick={() => setTime(h, m - 15, s)}>-15 min</button>
        <p>------</p>
        <button onClick={() => setTime(h, m + 1, s)}>+1 min</button>
        <button onClick={() => setTime(h, m - 1, s)}>-1 min</button>
      </div>
      <div
        className={
          lastClick === "Ended" || lastClick === "Reset"
            ? "timer-input-visible"
            : "timer-input-invisible"
        }
      >
        <div>
          <input
            type="number"
            placeholder={h.toString().padStart(2, "0")}
            onChange={(e) =>
              e.target.value > 0
                ? e.target.value < 99
                  ? setTime(parseInt(e.target.value), m, s)
                  : setTime(99, m, s)
                : setTime(0, m, s)
            }
            min="0"
            max="99"
          />
          :
          <input
            type="number"
            placeholder={m.toString().padStart(2, "0")}
            onChange={(e) =>
              e.target.value > 0
                ? e.target.value < 59
                  ? setTime(h, parseInt(e.target.value), s)
                  : setTime(h, 59, s)
                : setTime(h, 0, s)
            }
            min="0"
            max="59"
          />
          :
          <input
            type="number"
            placeholder={s.toString().padStart(2, "0")}
            onChange={(e) =>
              e.target.value > 0
                ? e.target.value < 59
                  ? setTime(h, m, parseInt(e.target.value))
                  : setTime(h, m, 59)
                : setTime(h, m, 0)
            }
            min="0"
            max="59"
          />
        </div>
      </div>
      <br />
      <div className="sidebar">
        <div
          className="timer-details sidebar-card"
          onClick={() => setIsFocus(!isFocus)}
        >
          <div className="visible-sidebar-card-info">
            # Focus: {totalTimeFocused.toFixed(0)}
            {totalTimeFocused.toFixed(0) === 1 ? " min" : " mins"} <br /> #
            Break: {totalTimeBreak.toFixed(0)}
            {totalTimeBreak.toFixed(0) === 1 ? " min" : " mins"}
          </div>
          {isFocus ? (
            <div className="invisible-sidebar-card-info">
              The coffee needs your attention. <br />
              Click to switch to break mode.
            </div>
          ) : (
            <div className="invisible-sidebar-card-info">
              The polar bear is back from his hibernation. Are you?
            </div>
          )}
        </div>
        <div className="timer-details sidebar-card-long">
          <div>
            <div>
              History |
              {lastHistory[0].date.getDate() +
                "-" +
                (lastHistory[0].date.getMonth() + 1) +
                "-" +
                lastHistory[0].date.getFullYear() ===
              new Date().getDate() +
                "-" +
                (new Date().getMonth() + 1) +
                "-" +
                new Date().getFullYear()
                ? " Today"
                : " " +
                  lastHistory[0].date.getDate().toString().padStart(2, "0") +
                  "-" +
                  (lastHistory[0].date.getMonth() + 1)
                    .toString()
                    .padStart(2, "0") +
                  "-" +
                  lastHistory[0].date.getFullYear().toString().padStart(4, "0")}
            </div>
            <ul className="timer-history-timeline">
              {lastHistory[0].history.map((item, index) => (
                <li key={index} className="timer-history-item">
                  <nav>{item}</nav>
                </li>
              ))}
              <li className="timer-history-item">
                The lazy organism seems to have become responsible!
              </li>
            </ul>
          </div>
        </div>
        <div className="timer-details sidebar-card-long">
          <div>
            <div>Focussed time:</div>
            <ul className="timer-history-timeline">
              {lastHistory.map((item, index) => (
                <li key={index} className="timer-history-item">
                  <nav>
                    {item.date.getDate() +
                      "-" +
                      (item.date.getMonth() + 1) +
                      "-" +
                      item.date.getFullYear() ===
                    new Date().getDate() +
                      "-" +
                      (new Date().getMonth() + 1) +
                      "-" +
                      new Date().getFullYear()
                      ? " Today"
                      : " " +
                        item.date.getDate().toString().padStart(2, "0") +
                        "-" +
                        (item.date.getMonth() + 1).toString().padStart(2, "0") +
                        "-" +
                        item.date
                          .getFullYear()
                          .toString()
                          .padStart(4, "0")}{" "}
                    | focus {item.focus}m, break {item.break}m
                  </nav>
                </li>
              ))}
            </ul>
```

```
          </div>
        </div>
      </div>
    </div>
  );
}
```

CHAPTER 5

RESULTS AND DISCUSSIONS

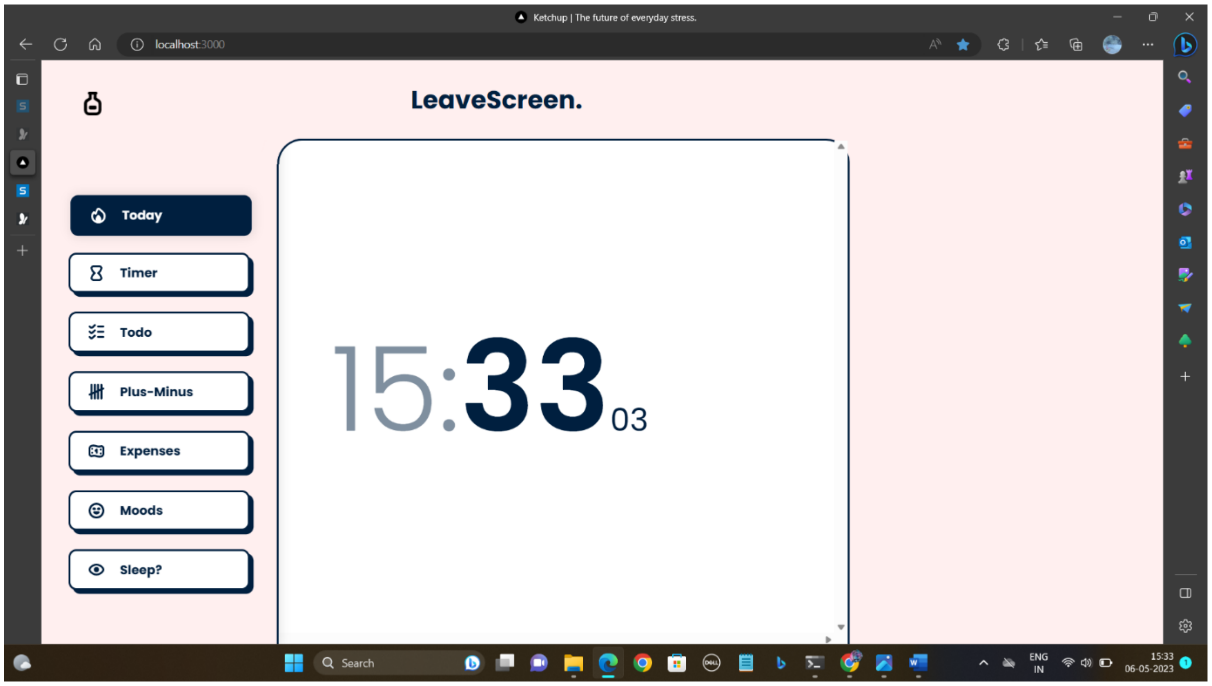
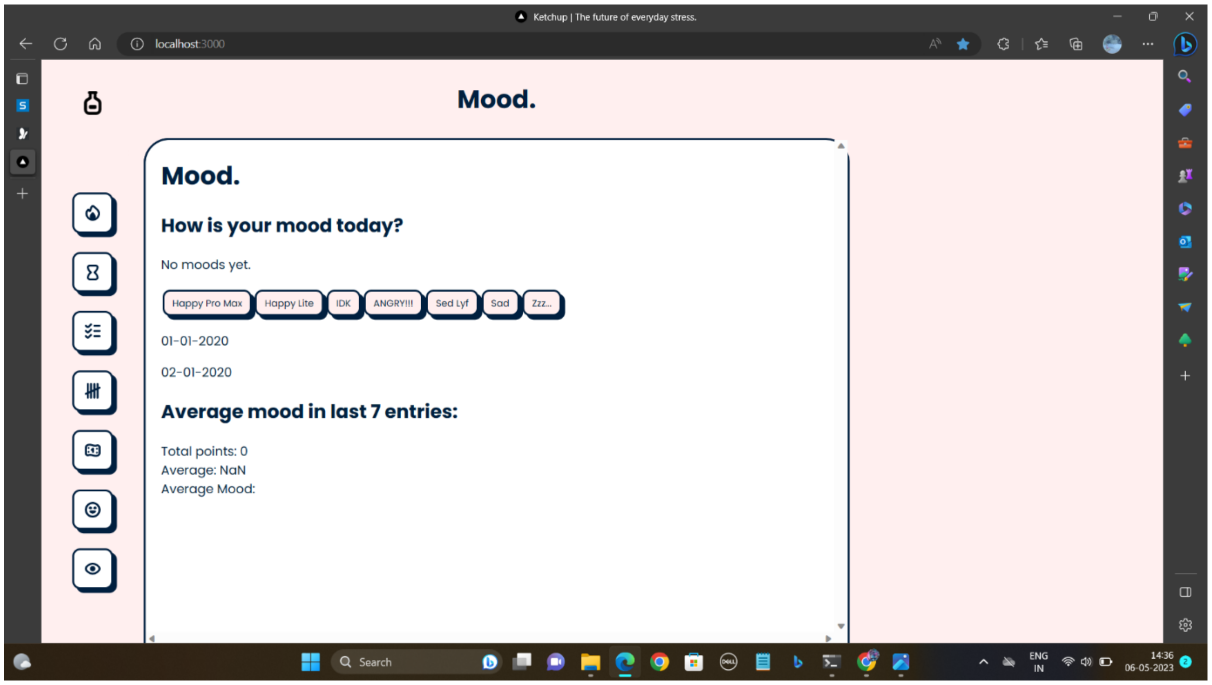


Fig.1: LeaveScreen



Mood.

How is your mood today?

No moods yet.

Happy Pro Max | Happy Lite | IDK | ANGRY!!! | Sed Lyf | Sad | Zzz...

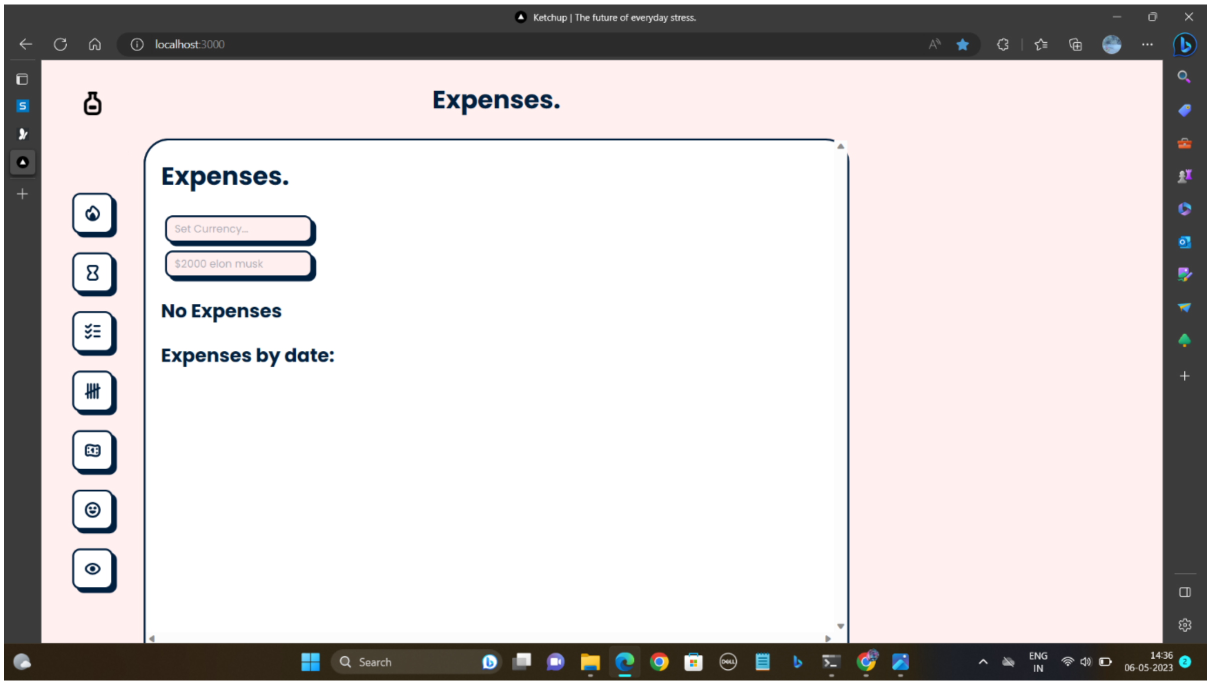
01-01-2020

02-01-2020

Average mood in last 7 entries:

Total points: 0
Average: NaN
Average Mood:

Fig.2: Mood Tracker



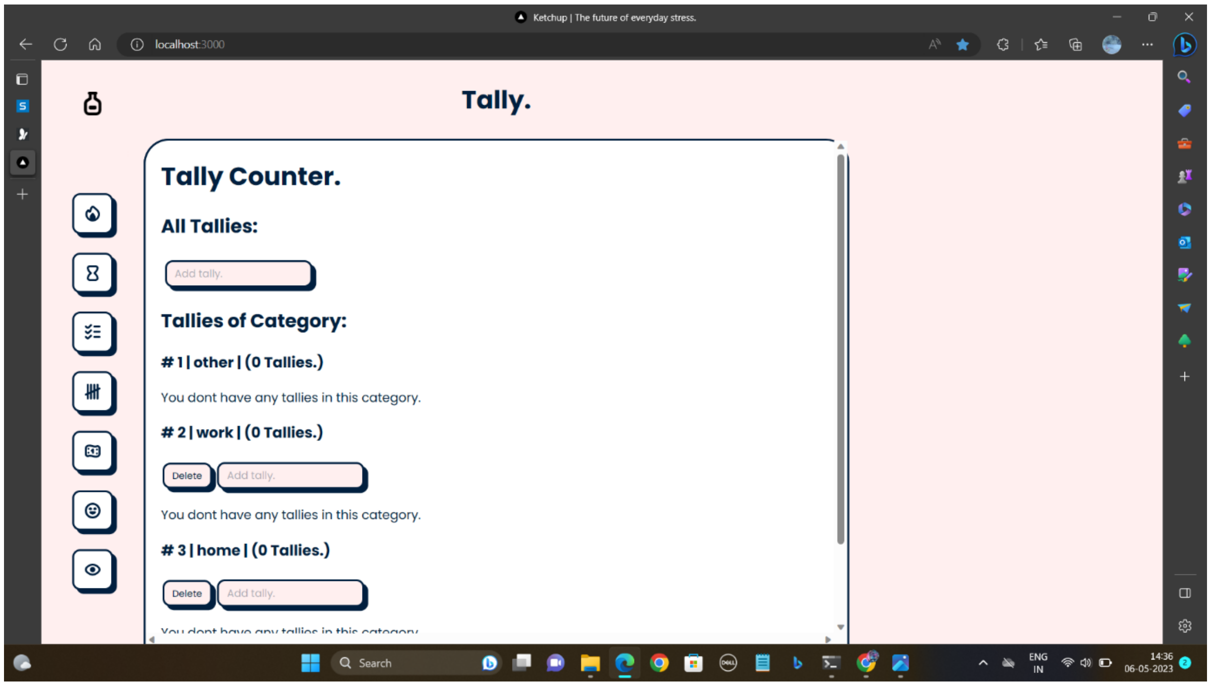
Expenses.

Expenses.

No Expenses

Expenses by date:

Fig.3: Daily expenses



Tally.

Tally Counter.

All Tallies:

Tallies of Category:

1 | other | (0 Tallies.)

You dont have any tallies in this category.

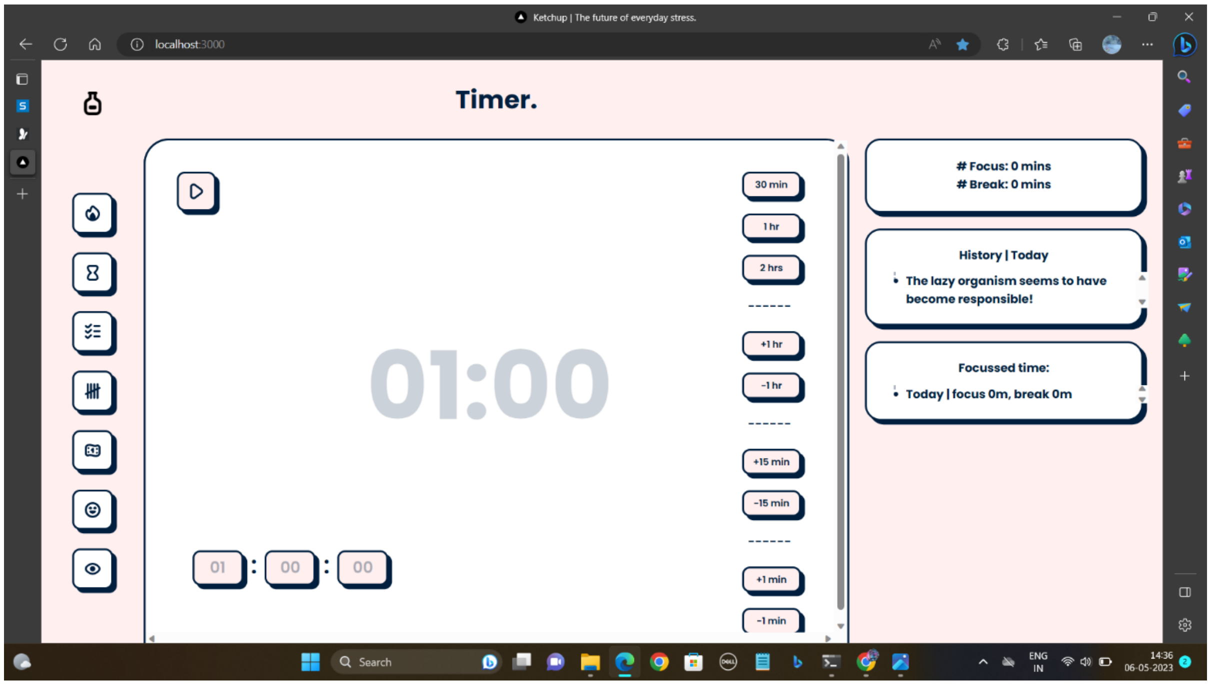
2 | work | (0 Tallies.)

You dont have any tallies in this category.

3 | home | (0 Tallies.)

You dont have any tallies in this category.

Fig.4: Tally of what we had completed



Focus: 0 mins
Break: 0 mins

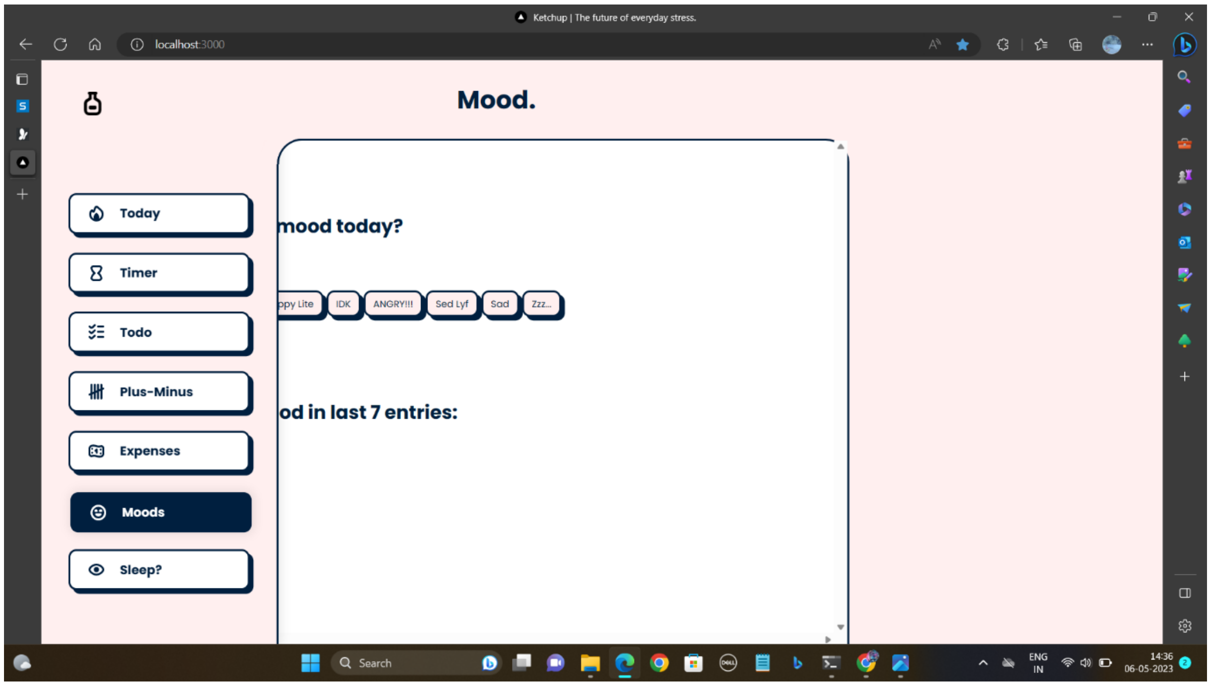
History | Today

- The lazy organism seems to have become responsible!

Focussed time:

- Today | focus 0m, break 0m

Fig.5: Timer



mood today?

od in last 7 entries:

Fig.6: Display of all the features

CHAPTER 6

CONCLUSION

We have developed a mood tracker, anxiety helper and mood-boosting journal, all rolled into one. The main purpose is to track your emotions over time to discover your mental health and become aware of your own mind.

REFERENCES

- https://legacy.reactjs.org/docs/getting-started.html
- https://www.freecodecamp.org/news/javascript-skills-you-need-for-react-practical-examples/
- https://github.com/topics/habit-tracker
- https://github.com/Jorres/stress-relief

****