DOROTRACKER

Hridya R Nair

Roll No.:29

ABSTRACT

The Pomodoro Microproject is a simple productivity application developed to help users manage their time efficiently and improve focus during study or work sessions. The system is based on the Pomodoro Technique, which divides work into fixed time intervals followed by short breaks to reduce mental fatigue and enhance concentration. The application allows users to start, pause, and reset timers for work and break sessions, providing a structured approach to time management, along with maintaining basic focus session records for productivity review. The project is developed using React for the front-end interface, Node.js with Express.js for handling server-side operations and client–server communication, MySQL for data storage, and CSS for styling and creating a user-friendly interface.

TABLE STRUCTURE

DB:pomodoro_db

1.users

Field	Description
id	Unique id of each user
name	Name of the user
email	Email id of the user used to log in
password	Password of the user that is used to log in

2.settings

Field	Description
id	Unique id of each setting
user_id	Id of the user(Foreign Key)
focus_min	Number of minutes set for focus session
break_min	Number of minutes set for break session

3.focus_log

Field	Description
id	Unique id of each focus log
user_id	Id of the user(Foreign Key)
focus_time	Number of minutes of a focus session
date	Date of the focus session

CODE

index.js

```
const express=require('express');
const cors=require("cors");
const mysql=require("mysql2");
const app=express();
app.use(cors());
app.use(express.json());
const db=mysql.createConnection({
   host:'localhost',
   user:'root',
   password:'',
   database:'pomodoro_db'
});
db.connect((err)=>{
   if(err){
      console.error("Connection Failed:",err);
      return;
   }
   console.log("Connection Successful");
});
app.set('db',db)
const authRoutes = require("./routes/auth");
const settingsRoutes = require("./routes/settings");
const focusRoutes = require("./routes/focus");
app.use("/api/auth", authRoutes);
app.use("/settings", settingsRoutes);
app.use("/focus", focusRoutes);
```

```
app.listen(5000,()=>{
   console.log("Server running on port 5000")
})
```

app.jsx

```
import { useState } from 'react'
import {BrowserRouter,Routes,Route} from "react-router-dom";
import Login from './pages/login';
import Settings from './pages/settings';
import Register from './pages/register';
import Home from './pages/home';
import History from './pages/history';
import Timer from './pages/timer';
import './App.css';
function App() {
  const [user, setUser] = useState(()=>{
   const savedUser = localStorage.getItem("user");
   return savedUser ? JSON.parse(savedUser) : null;
  });
 return (
   <BrowserRouter>
     <Routes>
        <Route path="/" element={<Login setUser={setUser}/>}/>
        <Route path="/register" element={<Register user={user}/>}/>
        <Route path="/home" element={<Home user={user}/>}/>
        <Route path="/timer" element={<Timer user={user}/>}/>
        <Route path="/settings" element={<Settings user={user}/>}/>
        <Route path="/history" element={<History user={user}/>}/>
     </Routes>
```

```
    </BrowserRouter>
  );
}
export default App
```

home.jsx

```
import { Link } from "react-router-dom";
export default function Home() {
    return (
        <div className="page-container">
  <div className="card">
    <div className="app-name">DoroTracker</div>
    <div className="navbar">
      <Link to="/timer" className="btn">Start Timer</Link>
      <Link to="/settings" className="btn">Settings</Link>
      <Link to="/history" className="btn">Focus History</Link>
    </div>
  </div>
</div>
    );
}
```

login.jsx

```
import { useState } from "react";
import { useNavigate,Link } from "react-router-dom";
export default function Login({ setUser }) {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
```

```
const navigate = useNavigate();
const handleLogin = async (e) => {
  e.preventDefault();
  const res = await fetch("http://localhost:5000/api/auth/login", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ email, password }),
  });
  const data = await res.json();
  if (res.ok) {
    localStorage.setItem("user", JSON.stringify(data.user));
    setUser(data.user);
    navigate("/home");
  } else {
    alert(data.message || "Login failed");
  }
};
return (
  <div className="page-container">
  <div className="card">
    <h2>Login</h2>
    <form onSubmit={handleLogin}>
      <input type="email" placeholder="Email" value={email}
        onChange={(e) => setEmail(e.target.value)} />
      <input type="password" placeholder="Password" value={password}
        onChange={(e) => setPassword(e.target.value)} />
      <button className="btn btn-submit">Login</button>
      <p>Don't have an account?<Link to="/register">Sign In</Link></p>
    </form>
  </div>
```

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

register.jsx

```
import { useState } from "react";
import { useNavigate,Link } from "react-router-dom";
export default function Register() {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();
  const handleRegister = async (e) => {
    e.preventDefault();
    const res = await fetch("http://localhost:5000/api/auth/register", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ name, email, password }),
    });
    const data = await res.json();
    if (res.ok) {
      alert("Registration successful! Please login.");
      navigate("/");
    } else {
      alert(data.message || "Registration failed");
    }
  };
  return (
    <div className="page-container">
```

```
    <div className="card">
      <h2>Create Account</h2>
      <form onSubmit={handleRegister}>
        <input type="text" placeholder="Name" value={name}
          onChange={(e) => setName(e.target.value)}/>
        <input type="email" placeholder="Email" value={email}
          onChange={(e) => setEmail(e.target.value)}/>
        <input type="password" placeholder="Password" value={password}
          onChange={(e) => setPassword(e.target.value)}/>
        <button className="btn btn-submit">Register</button>
         <p>Already have an account?<Link to="/">Log In</Link></p>
      </form>
    </div>
  </div>
  );
}
```

timer.jsx

```
import { useState, useEffect, useRef } from "react";
import { useNavigate } from "react-router-dom";
export default function Timer({ user }) {
  const [focusMin, setFocusMin] = useState(25);
  const [breakMin, setBreakMin] = useState(5);
  const [isRunning, setIsRunning] = useState(false);
  const [secondsLeft, setSecondsLeft] = useState(0);
  const [isFocus, setIsFocus] = useState(true);
  const timerRef = useRef(null);
  const navigate = useNavigate();
  // Fetch user's settings
```

```
useEffect(() => {
  if (!user) return;
  fetch(`http://localhost:5000/settings/${user.id}`)
    .then((res) => res.json())
    .then((data) => {
      setFocusMin(data.focus_min || 25);
      setBreakMin(data.break_min || 5);
      setSecondsLeft((data.focus_min || 25) * 60);
    })
    .catch((err) => console.error(err));
}, [user]);
const startTimer = () => {
  if (!isRunning) {
    setIsRunning(true);
    timerRef.current = setInterval(() => {
      setSecondsLeft((prev) => {
        if (prev <= 1) {
          clearInterval(timerRef.current);
          saveFocusSession();
          setIsFocus(!isFocus);
          setSecondsLeft((!isFocus ? focusMin : breakMin) * 60);
          setIsRunning(false);
          return (!isFocus ? focusMin : breakMin) * 60;
        }
        return prev - 1;
      });
    }, 1000);
  }
};
const stopTimer = () => {
```

```
  clearInterval(timerRef.current);
  setIsRunning(false);
};
const resetTimer = () => {
  clearInterval(timerRef.current);
  setIsRunning(false);
  setSecondsLeft((isFocus ? focusMin : breakMin) * 60);
};
const saveFocusSession = async () => {
  if (!user || !isFocus) return; // only save completed focus sessions
  await fetch("http://localhost:5000/focus", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ user_id: user.id, focus_time: focusMin }),
  });
};
const formatTime = (sec) => {
  const m = Math.floor(sec / 60);
  const s = sec % 60;
  return `${m.toString().padStart(2, "0")}:${s.toString().padStart(2, "0")}`;
};
return (
    <div className="page-container">
<div className="card">
  {/* Timer section */}
  <div className="timer-display">
              <h2 style={{fontSize: "3rem",
              fontWeight: 700,
              color: "#A18CD1",textAlign:"center"}} >Pomodoro Timer</h2>
    <h2>{isFocus ? "Focus Time" : "Break Time"}</h2>
```

```
      <h1>{formatTime(secondsLeft)}</h1>
      <div className="button-row">
        <button className="btn btn-start" onClick={startTimer}  disabled={isRunning}>
          Start
        </button>
        <button className="btn btn-pause" onClick={stopTimer}  disabled={!isRunning}>
          Stop
        </button>
        <button className="btn btn-reset" onClick={resetTimer}>
          Reset
        </button>
      </div>
       <button className="btn btn-back" onClick={() => navigate(-1)}>Back</button>
    </div>
  </div>
</div>
  );
}
```

settings.jsx

```
import { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
export default function Settings({ user }) {
  const [focusMin, setFocusMin] = useState(25);
  const [breakMin, setBreakMin] = useState(5);
  const navigate = useNavigate();
  useEffect(() => {
    if (!user) return;
    fetch(`http://localhost:5000/settings/${user.id}`)
```

```
    .then((res) => res.json())
    .then((data) => {
      setFocusMin(data.focus_min || 25);
      setBreakMin(data.break_min || 5);
    })
    .catch((err) => console.error("Failed to fetch settings:", err));
}, [user]);
const handleSave = async () => {
  if (!user) return alert("User not logged in");
  const res = await fetch(`http://localhost:5000/settings/`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ user_id: user.id, focus_min: focusMin, break_min: breakMin }),
  });
  const data = await res.json();
  if (res.ok) alert(data.message || "Settings updated!");
  else alert(data.error || "Failed to update settings");
};
return (
 <div className="page-container">
<div className="card">
  <h2>Settings</h2>
  <form>
    <label>Focus Minutes:</label>
    <input type="number" value={focusMin}
      onChange={(e) => setFocusMin(Number(e.target.value))}/>
    <label>Break Minutes:</label>
    <input type="number" value={breakMin}
      onChange={(e) => setBreakMin(Number(e.target.value))}/>
    <button type="button" className="btn btn-submit" onClick={handleSave}>
```

```
        Save
      </button>
    </form>
    <br/>
    <button className="btn btn-back" onClick={() => navigate(-1)}>Back</button>
  </div>
</div>
  );}
```

history.jsx

```
import { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
export default function History({ user }) {
  const [logs, setLogs] = useState([]);
  const navigate = useNavigate();
  useEffect(() => {
    if (!user) return;
    fetch(`http://localhost:5000/focus/${user.id}`)
      .then((res) => res.json())
      .then((data) => setLogs(data))
      .catch((err) => console.error("Failed to fetch history:", err));
  }, [user]);
  return (
  <div className="page-container">
  <h2>Focus History</h2>
  {logs.length === 0 ? (
    <p>No focus sessions yet.</p>
  ) : (
    <div className="session-container">
```

```
    {/* Today's session */}
    {(() => {
     const today = new Date().toDateString();
     const todayLog = logs.find(log => new Date(log.date).toDateString() === today);
     if (todayLog) {
      return (
       <div className="session-card today-session">
        <h3>Today's Focus</h3>
        <p><strong>Focused Minutes:</strong> {todayLog.focus_time}</p>
        <p><strong>Break Minutes:</strong> {todayLog.break_time || 0}</p>
       </div>
      );
     }
     return null;
    })()}
    {/* Previous days */}
    <div className="session-card">
     {logs
      .filter(log => new Date(log.date).toDateString() !== new Date().toDateString())
      .map(log => (
       <div key={log.id} className="session-card prev-session">
        <p><strong>Date:</strong> {new Date(log.date).toLocaleDateString()}</p>
        <p><strong>Focused Minutes:</strong> {log.focus_time}</p>
        <p><strong>Break Minutes:</strong> {log.break_time || 0}</p>
       </div>
      ))}
    </div>
   </div>
  )}
  <br />
```

```
  <button className="btn btn-back" onClick={() => navigate(-1)}>Back</button>
</div>
  );}
```

app.css

```
/* App */
body {
  margin: 0;
  padding: 0;
  font-family: "Inter", sans-serif;
  background: linear-gradient(135deg, #C8D6FF, #E0D4FF);
  min-height: 100vh;
  color: #333;
  display: flex;
  justify-content: center;
  align-items: center;
}
/* Center page content */
.page-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  width: 900px;
  padding: 20px;
  box-sizing: border-box;
}
/* Card */
.card {
```

```
  background: rgba(255, 255, 255, 0.3);
  backdrop-filter: blur(12px);
  padding: 30px;
  border-radius: 20px;
  box-shadow: 0 8px 20px rgba(0,0,0,0.1);
  border: 1px solid rgba(200, 200, 255, 0.3);
  width: 100%;
  text-align: center;
  box-sizing: border-box;
}

/* Section headers */
.card h2 {
  margin-top: 0;
  font-size: 22px;
  font-weight: 600;
  color: #A18CD1;
}
/* Timer */
.timer-display {
  font-size: 48px;
  font-weight: 700;
  font-family: "Courier New", monospace;
  letter-spacing: 2px;
  margin: 25px 0;
  color: #7facff;
}
/* Buttons */
.btn {
  padding: 12px 25px;
```

```
  border-radius: 12px;
  border: none;
  cursor: pointer;
  font-size: 15px;
  font-weight: 600;
  transition: 0.2s ease-in-out;
}
.btn:active {
  transform: scale(0.96);
}
.btn-start {
  background-color: #dfb3f8;
  color: #333;
}
.btn-start:hover {
  background-color: #daa8f7;
}
.btn-pause {
  background-color: #746581;
  color: #333;
}
.btn-pause:hover {
  background-color: #69577a;
}
.btn-reset {
  background-color: #D4C1FF;
  color: #333;
}
.btn-reset:hover {
  background-color: #B69EFF;
```

```
}
.button-row {
  gap: 10px;
}
/* History items */
.history-item {
  background: rgba(220, 220, 255, 0.3);
  padding: 12px 15px;
  border-radius: 12px;
  margin-bottom: 10px;
  font-size: 14px;
  box-shadow: 0 3px 8px rgba(0, 0, 0, 0.05);
}
.sessions-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 1.2rem;
  width: 500px;
}
/* Session Card */
.session-card {
  width: 500px;
  background: #fff;
  border-radius: 12px;
  padding: 1rem 1.2rem;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.08);
  display: flex;
  flex-direction: column;
  gap: 0.6rem;
```

```
  min-height: 135px;
}
.today-session {
  border-left: 5px solid #6c7ae0;
  width:400px;
}
.prev-session {
  border-left: 5px solid #a18cd1;
  width:400px;
}
/* Header */
.session-header {
  font-weight: 600;
  font-size: 1.1rem;
  color: #4b3f91;
}
/* Details layout */
.session-details {
  display: flex;
  justify-content: space-between;
  gap: 1rem;
}
/* Item */
.session-item {
  display: flex;
  flex-direction: column;
}
.label {
  font-size: 0.85rem;
  color: #555;
```

```
}
.value {
  font-size: 1rem;
  font-weight: 500;
  color: #1b042b;
}
.section-title  {
  font-weight: 600;
  font-size: 1rem;
  color: #6c7ae0;
  margin-bottom: 0.5rem;
}
.history-time  {
  font-weight: bold;
  color: #7FDBFF;
}
/* Forms */
form {
  display: flex;
  flex-direction:  column;
  gap: 12px;
  width: 100%;
}
input, select {
  padding: 12px;
  border-radius: 10px;
  border: 1px solid #C0B3FF;
  font-size: 15px;
  background-color: rgba(255,255,255,0.3);
  color: #333;
```

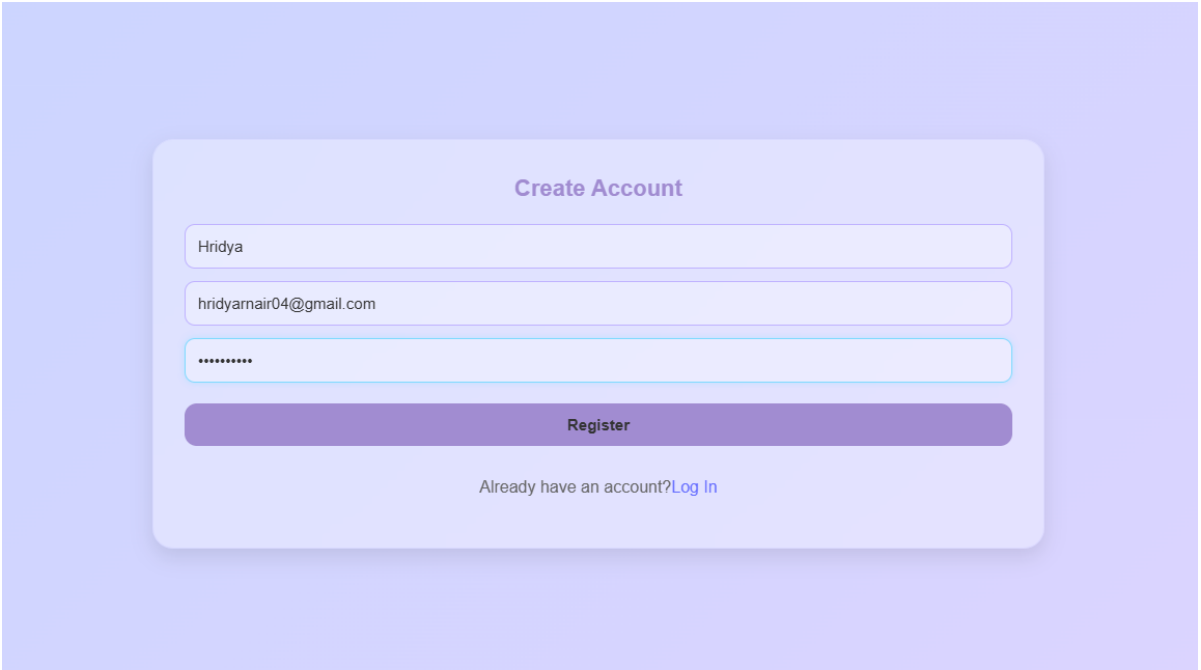
```
}
input:focus, select:focus {
  outline: none;
  border-color: #7FDBFF;
  box-shadow: 0 0 6px rgba(127,219,255,0.4);
}
.btn-submit {
  background-color: #A18CD1;
  color: #333;
  margin-top: 8px;
}
.btn-back {
  background-color: #A18CD1;
  color: #fff;
}
.btn-back:hover { background-color: #8B79C7; }
.page-container p {
  text-align: center;
  font-size: 16px;
  color: #666;
}
.btn-submit:hover {
  background-color: #8B79C7;
}
/* Navbar */
.navbar {
  padding: 15px 25px;
  background: rgba(240, 240, 255,0.3);
  backdrop-filter: blur(10px);
  display: flex;
```

```
  justify-content:  center;
  gap: 20px;
}
.navbar a {
  margin-left:  0;
}
.navbar a:hover { color: #7FDBFF; }
/* Table */
.page-container table {
  width: 100%;
  border-collapse:  collapse;
  background: rgba(255,255,255,0.3);
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
}
.page-container th,
.page-container td {
  padding: 12px 15px;
  text-align:  left;
  font-size: 16px;
}
.page-container th {
  background: #A3E4D7;
  color: #333;
}
.page-container tr:nth-child(even)  { background: rgba(200,220,255,0.2); }
.page-container tr:hover { background: rgba(180,200,255,0.3); }
/* Responsive */
@media (max-width: 480px) {
```

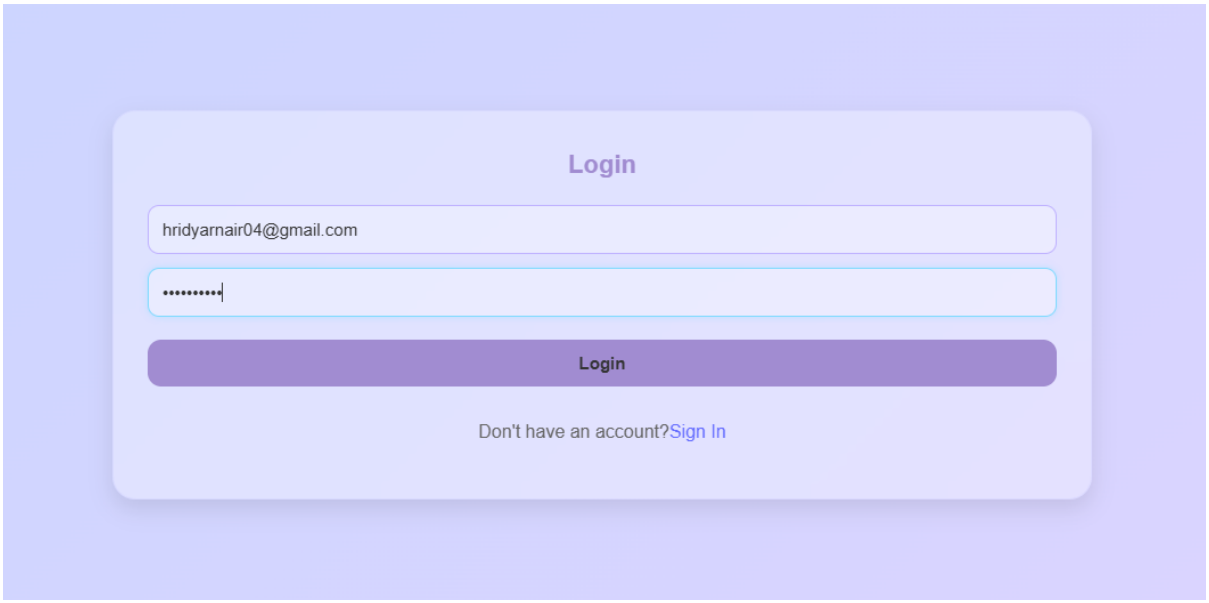
```
  .timer-display  { font-size: 38px; }
  .btn { padding: 10px 18px; }
  .card { padding: 20px; }
}
/* App name style */
.app-name {
  font-family: 'Inter', 'Poppins', sans-serif;
  font-size: 3rem;
  font-weight: 700;
  color: #A18CD1;
  text-align: center;
  text-shadow: 2px 2px 6px rgba(0,0,0,0.1);
  letter-spacing: 2px;
  margin: 0.5em 0;
}
```

OUTPUT SCREENS

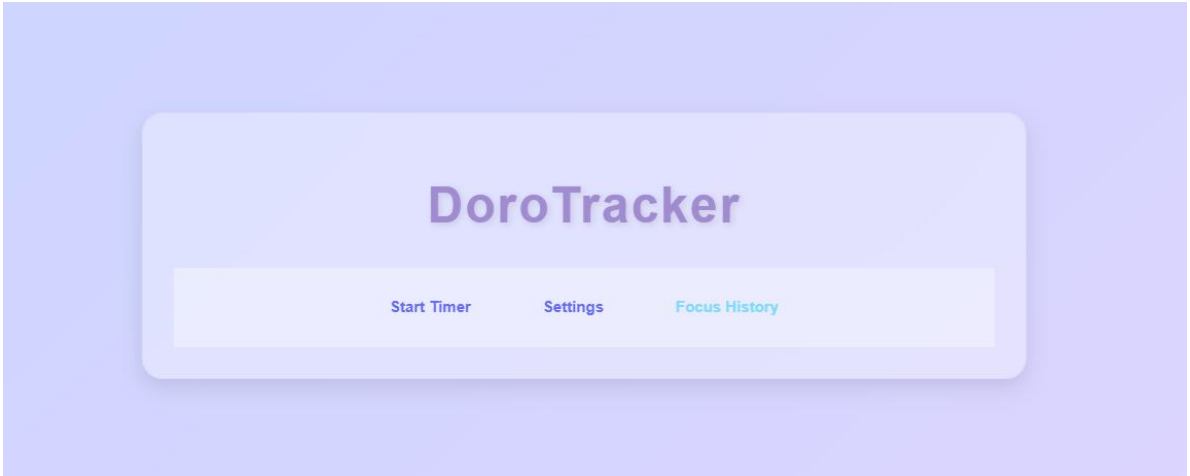
SIGN IN



LOGIN

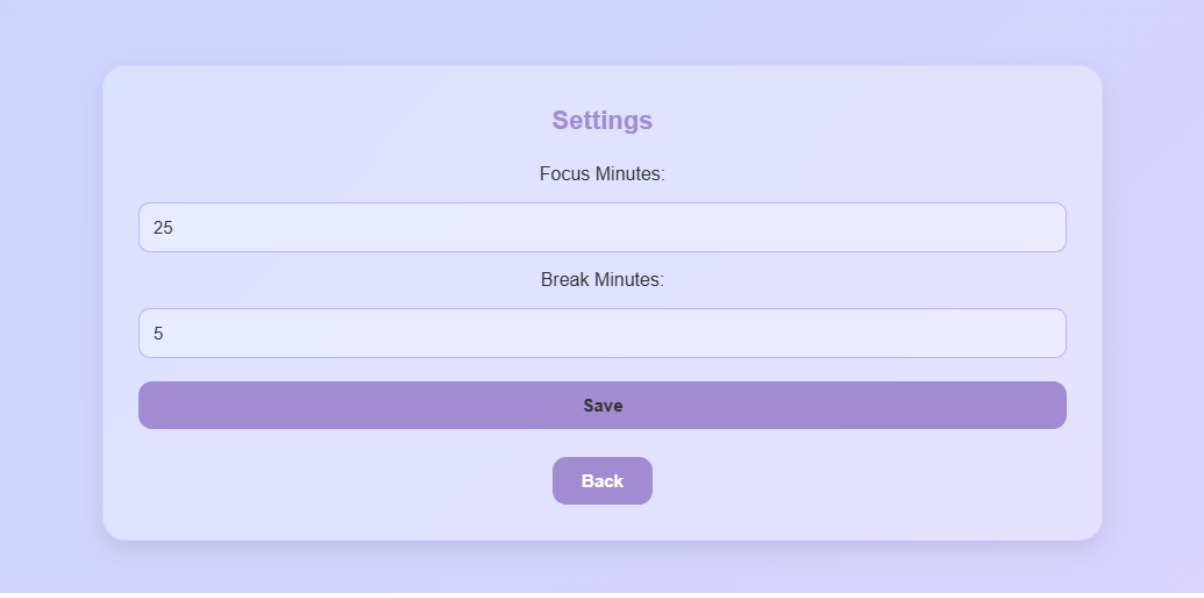

Login
hridyarnair04@gmail.com
Login
Don't have an account?Sign In

HOME


DoroTracker
Start Timer
Settings
Focus History

POMODORO TIMER

SETTINGS



FOCUS HISTORY

\