ספר פרוייקט - cockpit

שם: שילה פרג'ון .

1 הסבר על שלבי הפרוייקט

1.1 צד שרת

השורות הראשונות מייבאות את המודולים הדרושים:
readline מאפשר לקרוא קלט מהמשתמש מהשורת הפקודה.
Server ממודול socket.io מאפשר ליצור שרת WebSocket.
http מאפשר ליצור שרת HTTP.

1.2 ניצור אינטרפייס readline לקריאת קלט מהמשתמש ע"י פונקציית createInterface
1.3 ניצור שרת HTTP ונגדיר את הגדרות CORS עבור Socket.IO:

```
const server = http.createServer();
const io = new Server(server, {
  cors: {
    origin: true,      // Allow requests from any origin
    methods: ['GET', 'POST'], // Allow specified HTTP methods
  },
});
```

1.4 השרת מתחיל להאזין על פורט 3001
1.5 ניצור פונקציה אסינכרונית בשם getInput שמקבלת שאלה מהמשתמש ומחזירה את התשובה שהמשתמש הזין:

```
async function getInput(question) {
  return new Promise((resolve) => {
    // Use readline to ask the user a question and resolve the answer
    readline1.question(question, (answer) => {
      resolve(answer);
    });
  });
}
```

1.6 יצירת לולאת אינפיניטי שממשיכה לרוץ בלולאה בלתי נגמרת. בכל תור מבקשים מהמשתמש להזין את הגובה ,זווית , ADI ואישור
1.7 הלולאה האינפיניטית ממשיכה לרוץ כל הזמן, ובכל תור בודקים אם המשתמש אישר (או לא אישר) את הפרטים שהוזנו על ידו. אם המשתמש אישר, נשלח את הפרטים ללקוחות שמחוברים באמצעות Socket.IO ונציג הודעת הצלחה בקונסול.

צד לקוח

2 ראשית התחברתי לSocket.IO שמאזין לport :

```
// Creating a connection to the Socket.IO
const socket = io.connect("http://localhost:3001");
```

2.1 הקוד הזה משתמש ב-Hooks של React בשם `useEffec

2.2 "socket.on("receive_message זהו פקודת ההאזנה של Socket.IO שמתחילה להאזין לאירוע שיש לו שם "receive_message". כשהשרת שולח את האירוע "receive_message" ללקוחות (clients), הקוד בתוך הפונקציה הזו יופעל.

2.3 בתוך הפונקציה `socket.on`, יש שלוש פעולות `dispatch` שמטרתן לעדכן את המצב (state) באמצעות Redux. הן משתמשות בפעולות Redux שניות: `setHIS`, `setAltitude`, ו-`setADI`, והן משנות את המצב בהתאם למידע שנשלח מהשרת. במקרה הזה, המידע מגיע באופן אסינכרוני מהשרת ומתעדכן במצב של הרכיב כדי שהגרפים והתצוגה יתעדכנו בהתאם לנתונים החדשים שמגיעים מהשרת.

2.4 בחרתי להשתמש ב Redux קודם כל להראות שאני שולט ב start management וגם אישית יותר נח לעבוד עם Redux שמאפשר לי עדכון מכל מקום באפליקציה.

2.5 מאפשר לי לעבור בין הטסקט לוויזואל לראות את הנתונים ע"י הכפתורים:

```
const [openComponent, setOpenComponent] = useState(false);
```

2.6 כאן ניתן לראות את הניהול של ה Redux :

client > src > redux > features > monitorSlice.jsx > monitorSlice > initialState

```
import { createSlice } from "@reduxjs/toolkit";

const monitorSlice = createSlice({
  name: "monitor",
  initialState: {
    altitude: 0,
    HIS: 0,
    ADI: 0,
  },
  reducers: {
    setAltitude: (state, action) => {
      state.altitude = action.payload;
    },
    setHIS: (state, action) => {
      state.HIS = action.payload;
    },
    setADI: (state, action) => {
      state.ADI = action.payload;
    },
  },
});

export const { setAltitude, setHIS, setADI } = monitorSlice.actions;

export default monitorSlice.reducer;
```

2.7 כמו"כ אני אוהב ליצור לעצמי הוקים שיהיה נח להשתמש בהם באפליקציה ולמשוך אותם בכל קומפוננטה בלי בעיה וממש בקלות:

client > src > components > hooks > useMonitor.jsx > ...

```
import React from "react";
import { useSelector } from "react-redux";

const useMonitor = () => {
  const { altitude, HIS, ADI } = useSelector((store) => store.monitorReducers);

  return {
    altitude,
    ADI,
    HIS,
  };
};

export default useMonitor;
```

2.8 נעבור לקומפוננטה של הוויזואל, נתחיל עם החלק של **גובה המטוס** חילקתי את המספר שאני אקבל ב10, החלוקה ב-10 שביצעתי כאן בקוד בעניין של BottomA מתבצעת בגלל שהמספרים שמיוצגים על הגרף במסוף הוויזואלי גדולים מדי. לכן, במקום לעבוד עם ערכי גובה גדולים מדי, נכון להחליט לחלק את הערך של `altitude` ב-10 כך שבעצם המספר שיש לך ב-BottomA יהיה עשרות של המספר שניקח מ-altitude. זה יוצר תמציתי יותר ויותר קרוב לפיקסלים שבפועל הגובה על המסך. בכך הוא גורם לתצוגה להיות יותר אפשרית ונוחה לעבודה.

```
// Calculate a value for BottomA based on altitude
let BottomA = altitude / 10;
```

```
<div className="ml-10 border ☐border-black border-[3px] w-[4rem] h-[20rem]
  <div className="h-[30%] ">3000</div>
  <div className="h-[30%]">2000</div>
  <div className="h-[30%]">1000</div>
  <div className="">0</div>
  <div
    className={"absolute w-full h-2 border ☐border-black ☐bg-black"}
    style={{ bottom: `${BottomA}px` }}
  ></div>
</div>
{(altitude > 3000 || altitude < 0) && (
  // Show an alert for invalid altitude values
  alert("Please enter a number between 0 to 3000")
)}
</div>
```

2.9 הצגת זווית התעופה :

```
<div
  style={{ rotate: `${HIS - 360}deg` }}
  className="■bg-white text-[1.7em] ☐text-black font-bold rounde
>
  <div className="absolute inset-0 flex flex-col justify-center it
    <div className="absolute top-0 left-1/2 transform -translate-x
      0
    </div>
    <div className="absolute right-0 top-1/2 transform -translate-
      180
    </div>
    <div className="absolute top-1/2 left-0 transform -translate-y
      270
    </div>
    <div className="absolute bottom-0">90</div>
  </div>
</div>
```

3 **לזווית האופק** השתמשתי בuseState שישתנה בהתאם למספר שנקבל

3.1. בתוך ה-`useEffect`, ישנם תנאים שבודקים את הערך של `ADI`:
- אם `ADI` הוא 100, הקוד יגדיר את `circleColorClasses` ל- כחול.
- אם `ADI` הוא 0, הקוד יגדיר את `circleColorClasses` לכחול וירוק.
- אם `ADI` הוא -100, הקוד יגדיר את `circleColorClasses` לירוק בלבד.
- אם `ADI` לא תואם אף אחת מהתנאים, הוא יציג הודעת התראה למשתמש.

המטרה העיקרית של הקוד היא לשנות את הצבע של מעגל או אלמנט מסוים ברכיב בהתאם לערך של `ADI`. השימוש ב-`useEffect` `Hook` מבטיח שהשינוי בצבע יתרחש בזמן אמת בעת שינוי ב-`ADI`.

```
const [circleColorClasses, setCircleColorClasses] = useState('');

// Effect hook that runs when ADI changes
useEffect(() => {
  if (ADI === 100) {
    // Set circleColorClasses to 'bg-blue-500' for specific ADI value
    setCircleColorClasses('bg-blue-500');
  } else if (ADI === 0) {
    // Set circleColorClasses to gradient for specific ADI value
    setCircleColorClasses('bg-gradient-to-b from-blue-700 to-green-500');
  } else if (ADI === -100) {
    // Set circleColorClasses to 'bg-green-500' for specific ADI value
    setCircleColorClasses('bg-green-500');
  } else {
    // Show an alert for invalid ADI values
    alert("Please enter only these options: 100, -100, 0");
  }
}, [ADI]); // This effect depends on the ADI variable
```

<u>משאבים שנדרשו לפרויקט:</u>

- **תוכנות נדרשות:** VS Code, Redux toolkit,Socket.IO, chrome .
- **מקורות מידע:** אינטרנט, Socket.IO, youtube, stackoverflow.

תודה רבה