הסבר קוד שעון צומת

הרצת הקוד:

וודא שיש לך node.js מותקן על המחשב.
הורד למכשיר האנדרואיד שלך את האפליקציה Expo Go
פתח את הפרויקט ב visual studio , פתח טרמינל והרץ npm i
הרץ את הפקודה npm start ואז יפתח לך ברקוד לסריקה
פתח את האפליקציה בפלאפון וסרוק את הקוד, תופיע לך האפליקציה המקושרת לקוד כך כאשר תבצע שינויים ותשמור תוכל ישר לראות אותם בתוך האפליקציה.

בנוסף כרגע הקוד מקושר לפרויקט בEXPO דרכו אפשר לייצר גרסאות APK לקוד.
צריך להזדהות בטרמינל על ידי הפקודה expo login
ואז להכניס מייל raziel674@gmail.com
סיסמא : raz13791379
בשביל ליצור APK הרץ את הפקודה eas build --platform android
בסיום התהליך יופיע לך קישור להורדת קובץ הAPK שנוצר.

מכאן תוכל להשתמש באפליקציה כאשר שעון דולק בקרבת מקום ולחפש אותו בעזרת כפתור ה: חפש שעון חדש.

לכל שאלה נוספת ניתן לפנות לרזיאל סבתי:

0544272291

ניהול המידע של טבלאות התזמונים:

בקובץ screen/EditTable.tsx

יצרנו מטריצה דו-ממדית שמכילה מערכים של לוחות זמנים. כל שורה במטריצה מייצגת קבוצה נפרדת של לוחות זמנים השייכת למפסק ספציפי.

```
const [schedules, setSchedules] = useState<Schedule[][]>([[], []]);
```

זהו המערך הנוכחי של לוחות זמנים שאנו עובדים עליו, המתייחס לשורה ספציפית במטריצה, כלומר למפסק ספציפי.

```
const [schedule, setSchedule] = useState<Schedule[]>(schedules[curIndex]);
```

בשביל לקבוע על איזה מפסק אנו עובדים יצרנו dropdown עם בחירת מפסק וכל השינויים שהמשתמש יעשה יתעדכנו בשורה המתאימה במטריצה שמרכזת את כל מידע התזמונים.
בנוסף כך נדע איזה קבוצת תזמונים קיימת יש להציג למשתמש על המסך.

```
<DropdownComponent
  data={[
    { label: " 1 מפסק", value: "0" },
    { label: " 2 מפסק", value: "1" },
  ]}
  currentValue={String(curIndex)}
  onValueChange={(value) => {
    setCurIndex(Number(value));
    setSchedule(schedules[curIndex]);
  }}
/>
```

נעבור לקומפוננטה אחרת בה אנו מנהלים את השינויים עבור כל מפסק.
מוסיפים/מוחקים תזמון ומעדכנים את הערכים בתוך תזמון ספציפי.

קובץ – components/TableComponent

הקומפוננטה מנהלת מצב מקומי של התזמונים עבור המפסק, המאפשר עדכונים מהירים ויעילים.

```
const [localSchedules, setLocalSchedules] = useState<Schedule[]>(schedules);
```

סנכרון עם המטריצה שמנהלת את כלל התזמונים והמפסקים בEditTable.tsx

בכל פעם שהתזמונים משתנים ב EditTable.tsx, המצב המקומי מתעדכן בהתאם.

```
useEffect(() => {
  setLocalSchedules(schedules);
  setSchedule(schedules);
}, [schedules]);
```

פעולות ניהול:

1. <u>הוספת תזמון חדש:</u>
יוצר תזמון חדש עם ערכי ברירת מחדל.
מוסיף את התזמון החדש למערך הקיים.
מעדכן הן את המצב המקומי והן את מצב ההורה.

```
const addSchedule = () => {
  const newSchedule: Schedule = {
    ...default_values,
    scheduleID: localSchedules.length + 1,
  };
  const updatedSchedules = [...localSchedules, newSchedule];
  setLocalSchedules(updatedSchedules);
  setSchedule(updatedSchedules);
};
```

2. <u>מחיקת תזמון:</u>
מוחק את התזמון האחרון מהמערך.
מעדכן הן את המצב המקומי והן את מצב ההורה.

```
const deleteSchedule = () => {
  const updatedSchedule = localSchedules.slice(0, -1);
  setLocalSchedules(updatedSchedule);
  setSchedule(updatedSchedule);
};
```

3. <u>עדכון תזמון ספציפי:</u>

מעדכן שדה ספציפי בתזמון מסוים.

אם השדה המתעדכן אינו isActive והתזמון מסומן כצרוב לשעון, הוא יהפוך ללא צרוב.

מעדכן הן את המצב המקומי והן את מצב ההורה.

```
const updateSchedule = (id: number, field: string, newValue: string | number) => {
  const updatedSchedules = localSchedules.map((schedule) => {
    if (schedule.scheduleID === id) {
      if (field !== "isActive" && schedule.isActive) {
        return { ...schedule, [field]: newValue, isActive: false };
      } else {
        return { ...schedule, [field]: newValue };
      }
    }
    return schedule;
  });

  setLocalSchedules(updatedSchedules);
  setSchedule(updatedSchedules);
};
```

4. <u>עדכון תזמון לפי ערך ה"חזרות":</u>

מעדכן מספר שדות בתזמון בהתבסס על מצב החזרה החדש (newRepMode).

מתאים את השדות השונים לפי סוג החזרה (יומי, שבועי, חודשי, וכו').

מבצע עדכון מרוכז של כל השדות הרלוונטיים.

```
const batchUpdateSchedule = (id: number, newRepMode: string) => {
  // ... קוד עדכון מרובה ...
};
```

הצגת טבלת התזמונים:

קובץ components\DisplaySchedules.tsx

הקומפוננטה אחראית על הצגת רשימת התזמונים ומאפשרת עריכה של כל תזמון. היא מקבלת כ- props את רשימת התזמונים ,(schedules)
פונקציה לעדכון תזמון בודד ,(updateSchedule)
ופונקציה לעדכון מרובה של תזמון.(batchUpdateSchedule)

הצגת התזמונים בפועל:

1. **מיפוי התזמונים** :הקומפוננטה עוברת על מערך ה - schedules ומציגה כל תזמון בשורה נפרדת.
2. **מבנה השורה** :כל שורת תזמון מכילה :
   - מספר זיהוי(ID)
   - סוג החזרה (יומי, שבועי, וכו')
   - ימים או תאריכים (תלוי בסוג החזרה)
   - זמני הפעלה וכיבוי
   - מצבי זמן (זמן מקומי, לפני/אחרי שקיעה)
   - סטטוס צרוב/לא צרוב
3. **תצוגה דינמית** :התצוגה משתנה בהתאם לסוג החזרה שנבחר. למשל, עבור חזרה שבועית יוצגו ימי השבוע, ועבור חזרה חודשית יוצגו ימים בחודש.

מנגנון העדכון

1. **פתיחת מודל** :כאשר המשתמש לוחץ על אחד מהשדות, נפתח מודל עם Picker מתאים.
2. **בחירת ערך** :המשתמש בוחר ערך חדש מתוך ה - Picker.
3. **עדכון מצב מקומי** :הערך הנבחר מעדכן תחילה את המצב המקומי של הקומפוננטה.
4. **עדכון התזמון** :לאחר מכן, הערך החדש מועבר לפונקציית העדכון המתאימה.

פונקציות לתקשורת עם הרכיב:

קובץ components\ TableFunctionality.ts.ts

קומפוננטה זו מכילה פונקציות לתקשורת עם הרכיב. היא מאפשרת שליחה וקבלה של נתוני תזמון, הגדרת מצבי המפסק – דלוק/מכובה וניהול החיבור עם הרכיב.
הקומפוננטה משתמשת ב fetch API-לביצוע בקשות HTTP ומטפלת בשגיאות תקשורת.

1. פונקציה זו ממירה את התזמונים הקיימים במפסק הספציפי למבנה JSON לשליחה לרכיב.

```
function createJson(schedule: Schedule[], swID: number)
```

2. פונקציה זו שולחת את התזמונים במפסק ספציפי לרכיב.
תהליך:
הפיכת כל התזמונים לפעילים (isActive = true).
יצירת JSON מנתוני התזמון.
שליחת בקשת POST לרכיב עם הנתונים.
עדכון מצב החיבור והתזמונים בהתאם לתגובה, כלומר אם לא קיבלנו אישור מהרכיב שהנתונים התקבלו כנראה התקשורת מול הרכיב התנתקה ונציג הודעה למשתמש בהתאם בנוסף נשנה את הערכים חזרה ל- (isActive = false).

```
export async function sendCurrentData(
  schedule: Schedule[],
  setSchedule: React.Dispatch<React.SetStateAction<Schedule[]>>,
  swID: number,
  setConnectToChip: React.Dispatch<React.SetStateAction<boolean>>
)
```

3. פונקציה זו מבקשת את התזמונים הנוכחיים שצרובים על הרכיב במפסק ספציפי.
תהליך:
שליחת בקשת GET לרכיב.
פענוח התגובה והמרתה למערך של אובייקטי Schedule.
עדכון מצב התזמונים באפליקציה.

```
export async function getCurrentData(
  setSchedule: React.Dispatch<React.SetStateAction<Schedule[]>>,
  swID: number,
  setConnectToChip: React.Dispatch<React.SetStateAction<boolean>>
)
```

4. פונקציה זו מגדירה את מצב המפסק – ידני / תזמונים

תהליך:

שליחת בקשת GET לרכיב עם המצב הרצוי.
עדכון מצב החיבור בהתאם לתגובה. שוב אם הרכיב לא מגיב נקפיץ חיווי למשתמש שכנראה התקשורת נותקה ויש להתחבר מחדש.

```
export async function setSwitchMode(
  switchID: number,
  mode: string,
  setConnectToChip: React.Dispatch<React.SetStateAction<boolean>>
)
```

5. פונקציה זו מגדירה את מצב המפסק – דלוק/כבוי

תהליך:

שליחת בקשת GET לרכיב עם המצב הרצוי.
עדכון מצב החיבור בהתאם לתגובה. שוב אם הרכיב לא מגיב נקפיץ חיווי למשתמש שכנראה התקשורת נותקה ויש להתחבר מחדש.

```
export async function setLightMode(
  switchID: number,
  state: string,
  setConnectToChip: React.Dispatch<React.SetStateAction<boolean>>
)
```

בדיקה האם אנחנו מחוברים לרכיב:

המטרה היא לדעת האם המשתמש התחבר לשעון, כדי להתחבר לשעון המשתמש צריך לצאת מהאפליקציה ולכן בכל פעם שיוצא מהאפליקציה ואז חוזר, נשלח בקשת get לשעון ואם חוזרת תשובה זה אומר שאנחנו מחוברים.
addEventListener תופס אירוע של שינוי מצב אפליקציה, ממצב לא פעיל לפעיל או ממצב של "background" למצב פעיל.
בנוסף בכל פעם כשה status משתנה ממחובר ללא מחובר או להיפך, יוצג חלון קטן למשך שנייה שיודיע לנו האם התחברנו או התנתקנו.

```
useEffect(() => {
  const subscription = AppState.addEventListener(
    "change",
    async (nextAppState) => {
      if (
        appState.current.match(/inactive|background/) &&
        nextAppState === "active"
      ) {
        const controller = new AbortController();
        const timeoutId = setTimeout(() => controller.abort(), 1500);

        try {
          // Make the fetch request with a timeout
          const response = await fetch(`${ZAC_URL}/Get?Status`, {
            method: "GET",
            signal: controller.signal,
          });

          // Clear the timeout if the request completed successfully
          clearTimeout(timeoutId);

          if (response.ok) {
            const data = await response.json();
            if (data.hasOwnProperty("zacName")) {
              setConnectToChip(true);
```

```
            } else {
              setConnectToChip(false);
            }
          }
        } catch (error: any) {
          setConnectToChip(false);
        }
      }
      appState.current = nextAppState;
    }
  );

  return () => {
    subscription.remove();
  };
}, []);
```

<u>הרשאת האפליקציה לתקשר עם הרכיב:</u>

קובץ android\app\src\main\AndroidManifest.xml

קובץ קריטי באפליקציית Android שמכיל מידע חיוני על האפליקציה שלך, כולל:

1. **הרשאות**: ההרשאות שהאפליקציה זקוקה להן, כמו גישה לאינטרנט או לקרוא קבצים.
2. **הגדרות האפליקציה**: פרטים על האפליקציה, כמו שם, אייקון, ומאפייני תצורה אחרים.

הקובץ מאפשר למערכת ההפעלה להבין איך האפליקציה צריכה להתנהג ומה הדרישות שלה.

בתוך קובץ זה יש הפנייה לקובץ אחר שנראית כך :

```
<application
    android:networkSecurityConfig="@xml/network_security_config"
    ...>
```

זה אומר שהאפליקציה תשתמש בהגדרות האבטחה מהקובץ network_security_config.xml בעת ביצוע חיבורי רשת.

הקובץ הזה יושב ב - android\app\src\main\res\xml\network_security_config.xml

ונראה כך:

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">192.168.4.1</domain>
    </domain-config>
</network-security-config>
```

הסבר על ההגדרות בקובץ:

<domain-config cleartextTrafficPermitted="true":

cleartextTrafficPermitted="true": מאפשר חיבורי HTTP (לא מאובטחים) לדומיינים המוגדרים בתוך בלוק זה. ברירת המחדל היא ש-Android מונע חיבורי HTTP לא מאובטחים, כדי לשפר את האבטחה, ולכן יש להפעיל את ההגדרה הזו במפורש אם רוצים להשתמש ב-HTTP.

<domain includeSubdomains="true">192.168.4.1</domain>:

"includeSubdomains="true": מאפשר את החיבור לכתובת IP זו ולכל תתי-דומיינים שלה. במקרה שלך, מאחר וכתובת ה-IP היא 192.168.4.1, הכוונה היא לאפשר חיבור לכתובת זו בכל תצורה שלה (למשל, אם יש תתי-דומיינים או סביבות משנה).

**** חשוב לציין שללא הקבצים האלה וההגדרות שבתוכן לא ניתן יהיה לתקשר עם הרכיב! ****