הנה תרשים כללי שמתאר את שלבי הפרויקט והארכיטקטורה של המערכת:

**1. הגדרת המטרה והדרישות**

* יצירת לוח דיגיטלי לבתי כנסת ומוסדות.
* הצגת מידע משתנה בזמן אמת (זמני תפילות, טקסטים, תמונות, סרטונים וכו').
* הגדרת ממשק משתמש, תמיכה ב-Routes, מערכת ניהול מרחוק, הצגת מידע מדויק לפי תוכנת "חזון-שמים".

**2. שלב ראשון - Backend (Node.js)**

* יצירת פרויקט ב-Node.js עם Express.
* יצירת Routes ליצירה, עדכון, קריאה ומחיקה של נתונים:
  + **POST** (Create): יצירת מידע חדש (זמני תפילות, תמונות וכו').
  + **PUT** (Update): עדכון נתונים קיימים.
  + **GET** (Read): שליפת נתונים להציג בריאקט.
  + **DELETE** (Delete): מחיקת נתונים.
* הגדרת JWT לאימות משתמשים והצפנת סיסמאות.
* בדיקת הפונקציות בפוסטמן (Postman).

**3. שלב שני - Frontend (React)**

* יצירת פרויקט React עם Vite (npm create vite@latest).
* הגדרת קומפוננטות לפי הארכיטקטורה:
  + **Components**: כל רכיב עצמאי (כגון לוח זמנים, לוח הנצחה).
  + **Containers**: רכיבים שמכילים קומפוננטות אחרות.
  + **Services**: מודולים שאחראים על שליפת נתונים באמצעות axios.
* הגדרת ניתוב באמצעות react-router-dom.
* הגדרת Redux לטיפול במידע גלובלי (כמו נתוני התפילות).
* שימוש ב-React-Hooks-Form לבניית טפסים (ליצירת ועריכת נתונים).
* בניית Custom Hooks למטרות שונות (לדוג' לניהול נתונים או לבצע חישובים).

**4. שלב שלישי - שילוב Backend ו-Frontend**

* שימוש ב-axios לשליפת נתונים מ-Node.js ולהציגם ב-React.
* הצגת הנתונים בצורה דינמית לפי הזמן והגדרות המשתמש.
* תזמון הנפשות (כגון הצגת טקסטים, תמונות וידאו ושמע בזמן אמת).

**5. שלב רביעי - ניהול מערכת וממשק ניהול**

* פיתוח ממשק ניהול שמאפשר למנהלי המערכת לעדכן נתונים בצורה מרחוק.
* הגדרת ממשק משתמש ידידותי וקל לתפעול, עם אפשרות לעצב את המידע המוצג (לוח זמני תפילה, לוח הנצחה).

**שלבי הפרויקט:**

1. **הגדרת פרויקט Node.js ו-Express.**
2. **יצירת Routes (POST, PUT, GET, DELETE) עבור נתונים.**
3. **יצירת פרויקט React עם Vite.**
4. **הגדרת ניתוב וקומפוננטות בריאקט.**
5. **שילוב Axios לשליפת נתונים מהשרת.**
6. **הגדרת Redux לניהול מידע גלובלי.**
7. **הגדרת אימות משתמשים באמצעות JWT.**
8. **הגדרת ממשק ניהול ועדכון נתונים מרחוק.**

**קומפוננטות (Components)**

1. **לוח זמני תפילה (PrayerTimesBoard)**:
   * תציג את זמני תפילה של היום.
   * תתעדכן אוטומטית בהתאם לשעה ותאריך.
2. **לוח הנצחה (MemorialBoard)**:
   * הצגת שמות זוכרים, תאריכים ומידע נוסף על הנפטרים.
3. **תפריט ניווט (Navbar)**:
   * תכלול כפתורים למעבר בין דפים (כגון לוח זמני תפילה, לוח הנצחה, הגדרות ועוד).
4. **כרטיס יצירת עדכון מידע (CreateUpdateCard)**:
   * טופס להצגת שדות למילוי ויצירת או עדכון נתונים (למשל: תאריך, שם, זמן תפילה, תמונה וכו').
5. **טופס הגדרות (SettingsForm)**:
   * לאפשר למשתמש להגדיר את הצג (הגדרות זמן, צבעים, עיצובים גרפיים, אפשרויות הדפסה).
6. **חיפוש (SearchBar)**:
   * אפשרות לחיפוש מידע לפי מילות מפתח (לוח זמנים, שמות בהנצחה, ועוד).
7. **מסך התחברות/הרשמה (LoginRegister)**:
   * טופס התחברות והרשמה למשתמשים עם אימות JWT.
8. **ממשק ניהול (AdminDashboard)**:
   * ניהול נתונים מרחוק (הוספה, עדכון, מחיקה של נתונים).
9. **הצגת תמונות ווידאו (MediaViewer)**:
   * רכיב להצגת תמונות ווידאו (למשל בהנצחות או סרטונים של תפילות).
10. **לוח בית כנסת (SynagogueBoard)**:

* קונפיגורציה של לוח מידע מרכזי המשלב את כל המידע הדרוש.

**מודלים (Models) ב-Backend**

1. **מודל משתמש (User Model)**:
   * **שדות**:
     + שם משתמש, סיסמא (מוצפנת), אימייל, תפקיד (מנהל/משתמש).
     + שימוש באימות JWT.
   * **פונקציות**:
     + יצירת משתמש חדש, התחברות, עדכון פרטי משתמש.
2. **מודל לוח זמני תפילה (PrayerTime Model)**:
   * **שדות**:
     + תאריך, זמן תפילה (שחרית, מנחה, ערבית), מיקום גיאוגרפי (כדי לחשב את זמני התפילה).
   * **פונקציות**:
     + יצירת זמן תפילה חדש, עדכון, מחיקה.
3. **מודל לוח הנצחה (Memorial Model)**:
   * **שדות**:
     + שם הנפטר, תאריך פטירה, תיאור נוסף (הנצחה, משפחה).
   * **פונקציות**:
     + הוספה, עדכון, מחיקה של נתונים.
4. **מודל הגדרות (Settings Model)**:
   * **שדות**:
     + צבעים, עיצוב גרפי, הגדרות דינמיות אחרות (תצוגה משתנה לפי זמן, עיצוב הלוח).
   * **פונקציות**:
     + עדכון הגדרות המערכת.
5. **מודל אירועים (Event Model)**:
   * **שדות**:
     + סוג האירוע (תפילה מיוחדת, חג, תערוכה).
     + תאריך ושעה, תיאור.
   * **פונקציות**:
     + הוספת אירוע חדש, עדכון, מחיקה.
6. **מודל התחברות/הרשמה (Auth Model)**:
   * **שדות**:
     + פרטי משתמשים, סיסמאות מוצפנות.
   * **פונקציות**:
     + אימות משתמשים, יצירת טוקן JWT.

**ארכיטקטורת פרויקט**

1. **Backend**:
   * Express.js עבור ניהול ה-API.
   * JWT לאימות משתמשים.
   * MongoDB או כל DB אחר לאחסון נתונים.
2. **Frontend**:
   * React עבור הממשק המשתמש.
   * Redux לניהול המידע הגלובלי (למשל, זמני תפילה).
   * React Router לניהול ניתובים בין דפים.
   * Axios לשליפת נתונים מהשרת.
   * React-Hooks-Form לניהול טפסים.
   * Custom Hooks למודולציה של לוגיקה כמו חישוב זמני תפילה.

**1. תיקיית Components (קומפוננטות)**

כל רכיב צריך לקבל נתונים בצורה דינמית ולתמוך בממשק משתמש נוח:

* **PrayerTimesBoard.jsx** – הצגת זמני תפילה עם עדכון אוטומטי.
* **MemorialBoard.jsx** – הצגת רשימת נפטרים.
* **Navbar.jsx** – תפריט ניווט לכלל הדפים.
* **CreateUpdateCard.jsx** – טופס להוספת נתונים.
* **SettingsForm.jsx** – טופס הגדרות לעיצוב הלוח.
* **SearchBar.jsx** – רכיב חיפוש דינמי.
* **LoginRegister.jsx** – טופס התחברות/הרשמה.
* **AdminDashboard.jsx** – לוח ניהול נתונים.
* **SynagogueBoard.jsx** – שילוב כל הרכיבים יחד.

*מה לעשות?*

* הגדר useState ו-useEffect להצגת נתונים.
* השתמש בprops לקבלת מידע מבחוץ.
* הוסף axios למשיכת מידע מהשרת.

**2. תיקיית Containers (מיכלים)**

אלה רכיבים שעוטפים כמה קומפוננטות:

* **MainLayout.jsx** – מסגרת לכל הדפים.
* **DashboardContainer.jsx** – מנהל את תצוגת לוח הניהול.
* **PrayerContainer.jsx** – מכיל את לוח זמני התפילה.

*מה לעשות?*

* שלוף נתונים מה-Redux או מהשרת.
* נהל סטייט גלובלי עם useContext או Redux.

**3. תיקיית Services (שירותים)**

כאן מנהלים קריאות API עם axios:

* **prayerService.js** – שליפת זמני תפילה.
* **userService.js** – התחברות/הרשמה.
* **settingsService.js** – שליפת הגדרות.

*מה לעשות?*

* כתוב פונקציות GET, POST, PUT, DELETE.
* השתמש ב-axios.create לניהול בסיס כתובת API.

**שלבים להמשך**

1. התחל ממילוי הנתונים בקומפוננטות המרכזיות.
2. חבר את הקריאות ל-API דרך services.
3. השתמש ב-Redux לניהול מידע כללי כמו משתמשים וזמני תפילה.
4. עצב עם Tailwind או CSS מתאים.