



1. קונספט הנתונים (Data Model)

לפנינו שכותבים קוד, חשוב להבין את היחסות ב-Database (למשל MongoDB או PostgreSQL).

- **User**: פרטי משתמש, תמונה פרופיל.
- **Plant**: שם, סוג, תאריך שתילה, סטטוס (Active/Archived/Dead), תמונה ראשית.
- **Post**:
 - Type: Update (Growth), Swap, Giveaway
 - Relations: מקשר ל-User, ואופציונליות מקשר ל-Plant ספציפי.
 - Content: טקסט, תמונות.

2. Backend API (Node.js + TS)

המטרה: API נקי שיאפשר לשימוש בעתיד גם את האפליקציה (React Native/Flutter) וגם את האתר.

Auth Routes

- POST /api/auth/register – הרשמה.
- POST /api/auth/login – התחברות (קבלת Token).

(Plant Routes (The Inventory

ניהול "המחסן" של המשתמש.

- GET /api/plants – קבלת כל הצמחים של המשתמש המחבר (עם פילטרים: פעיל/ארביוון).
- POST /api/plants – הוספה צמח חדש לאוסף.
- GET /api/plants/:id – קבלת פרטי צמח ספציפי (כולל היסטוריה הפוסטים שלו).
- PUT /api/plants/:id – עריכת פרטי צמח (למשל: שינוי סטטוס למתח).

Feed & Posts Routes

הפייד והחיבור החברתי.

- ● הפיד הראשי (כל הפוסטים / פוסטים של חברים). GET /api/posts
 - ● ייצירת פост חדש. POST /api/posts
 - *Payload:* { content, type, plantId? } אם יש (postId, content, type, plantId).
יופיע גם בפייד וגם בדף הצמח.
סינון הפיד רק להחלפות/מתנות. GET /api/posts?type=swap
-

3. Frontend Architecture (React + Tailwind)

גישה **Mobile First**: העיצוב הבסיסי הוא לבנייד (רווח מלא, בפתחורים גדולים), ומרחיבים למסכים גדולים (RWD) אם צריך.

מבנה העמודים (Pages)

1. Feed Page (Home): גלילה אינטואטיבית של עדכונים.
2. Inventory Page: גריד של ברטיסיות צמחים (במו גלריה).
3. Plant Details: עמוד לצמח בודד שמרת את הסטטיסטיקות שלו ואת כל ה פוסטים שקשרו אליו (Timeline של גידלה).
4. Create Post/Plant: טופס (רצוי ב-Modal או עמוד נפרד בטלפון) להוספה.

קומponeנטות מרכזיות (Components)

- Layout: מעתפת ראשית שמכילה את ה-**BottomNavigation** (לטלפון) או **SideBar** (למסך).
 - PlantCard: מציג תמונה, שם וסטטוס.
 - FeedPost: הקומponeנטה המורכבת ביותר. צריכה להציג:
 - מי המשתמש.
 - על أي צמח מדובר (לינק לצמח).
 - סוג הפост (Update/Swap/Giveaway – תיוג צבעוני ב-Tailwind.).
 - כפתור צפ (+) להוספה מהירה. ActionFab
-

4. חלוקת משימות (2 מפתחים)

בדי שלא תדרכו אחד לשני על הרגליים, נחלק את זה לפי **פיז'רים** (Full Stack) ולא לפי שבבות (Front/Back).

ניצן: האחראי על ה-Inventory (הבסיס)

- .Plant CRUD-ו Auth-ו, וIMPLEMENTATION ה-DB, CHASSIS כיבור ל-DB, וIMPLEMENTATION ה-Auth •
Backend : הקמת השירות, CHASSIS כיבור ל-DB, וIMPLEMENTATION ה-Auth •
Frontend : •
 - הקמת הפרויקט (Vite + Tailwind).
 - בניית עמוד ה-Inventory.
 - בניית עמוד הוסף צמח.
 - בניית עמוד "פרטי צמח".

ניר: האחראי על ה-Social (החיבור)

- .Posts CRUD-ו, וקישור בין Post ל-Post ב-DB •
Backend : IMPLEMENTATION ה-DB, קישור בין Post ל-Post ב-DB •
Frontend : •
 - בניית ה-Feed (עיצוב ברטיסי פוסט).
 - בניית טופס "צורת פוסט" (כולל Dropdown לבחירת צמח מהאינוננטורי של המשתמש).
 - פילטרים (הציג רק החלפות/מוגנות).

ניר: אחראי על ה פרופיל

ניצן: אחראי על ההתחברות

5. דוגמה טכנית ל-TypeScript Interface

בדי שתהיינו מסונכרנים ב-`Types`

Shared Types //

```
;’export type PlantStatus = ‘active’ | ‘dead’ | ‘gifted’  
;’export type PostType = ‘update’ | ‘swap’ | ‘giveaway’
```

```
} export interface IPlant  
    ;id: string  
    ;userId: string  
    ;name: string  
    ;species: string  
    ;status: PlantStatus  
    ;imageUrl: string  
    ;createdAt: Date  
{
```

```
} export interface IPost  
    ;id: string  
    ;userId: string  
    // אופציוני – כי אולי אני סתם רוצה לשאול שאלה כללית  
    ;plantId?: string  
    ;type: PostType  
    ;content: string  
    ;[]images: string  
    ;createdAt: Date  
{
```

Tailwind (Mobile First) TIP:

בשאתם בותבים את ה-`Classes`, תמיד כתבו קודם לモבייל, ואז תוסיפו התנויות למסך גדול. דוגמה לגריד של ה-`Inventory`:

```
<"div className="grid grid-cols-2 md:grid-cols-4 gap-4>
    /* בモビיל 2 עמודות, בדסקטופ 4 עמודות */
{(</ {plants.map(plant => <PlantCard key={plant.id} plant={plant}
        <div/>
```