

# RootShare

## 1. קונספט הנתונים (Data Model)

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

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

## 2. Backend API (Node.js + TS)

המטרה: API נקי שיוכל לשמש 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 המחובר (עם פילטרים פעיל/ארכיון).
- **POST /api/plants** - הוספת צמח חדש לאוסף.
- **GET /api/plants/:id** - קבלת פרטי צמח ספציפי (כולל היסטוריית הפוסטים שלו).
- **PUT /api/plants/:id** - עריכת פרטי צמח (למשל: שינוי סטטוס למת).

## Feed & Posts Routes

הפיד והחיבור החברתי.

- **GET /api/posts** - הפיד הראשי (כל הפוסטים / פוסטים של חברים).
- **POST /api/posts** - יצירת פוסט חדש.
  - **Payload:** { content, type, plantId? } (plantId, אם יש).
- **GET /api/posts?type=swap** - סינון הפיד רק להחלפות/מתנות.

---

### 3. Frontend Architecture (React + Tailwind)

גישת **Mobile First**: העיצוב הבסיסי הוא לנייד (רוחב מלא, כפתורים גדולים), ומרחיבים למסכים גדולים (RWD) אם צריך.

מבנה העמודים (Pages)

1. **Feed Page (Home)**: גלילה אינסופית של עדכונים.
2. **Inventory Page**: גריד של כרטיסיות צמחים (כמו גלריה).
3. **Plant Details**: עמוד לצמח בודד שמראה את הסטטיסטיקות שלו ואת כל הפוסטים שקושרו אליו (Timeline של גדילה).
4. **Create Post/Plant**: טופס (רצוי כ-Modal או עמוד נפרד במובייל) להוספה.

קומפוננטות מרכזיות (Components)

- **Layout**: מעטפת ראשית שמכילה את ה-**BottomNavigation** (למובייל) או **SideBar** (לדסקטופ).
- **PlantCard**: מציג תמונה, שם וסטטוס.
- **FeedPost**: הקומפוננטה המורכבת ביותר. צריכה להציג:
  - מי המשתמש.
  - על איזה צמח מדובר (לינק לצמח).
  - סוג הפוסט (Update/Swap/Giveaway - תיוג צבעוני ב-Tailwind).
- **ActionFab**: כפתור צף (+) להוספה מהירה.

#### 4. חלוקת משימות (2 מפתחים)

כדי שלא תדרכו אחד לשני על הרגליים, נחלק את זה לפי **פיצ'רים** (Full Stack) ולא לפי שכבות (Front/Back).

ניצן: האחראי על ה-Inventory (הבסיס)

- Backend: הקמת השרת, חיבור ל-DB, ומימוש ה-Auth ו-Plant CRUD.
- Frontend:

- הקמת הפרויקט (Vite + Tailwind).
- בניית עמוד ה-Inventory.
- בניית עמוד הוספת צמח.
- בניית עמוד "פרטי צמח".

ניר: האחראי על ה-Social (החיבור)

- Backend: מימוש ה-Posts CRUD, וקישור בין Post ל-Plant ב-DB.
- Frontend:
- בניית ה-Feed (עיצוב כרטיסי פוסט).
- בניית טופס "יצירת פוסט" (כולל Dropdown לבחירת צמח מהאינוונטורי של המשתמש).
- פילטורים (הצג רק החלפות/מתנות).

ניר: אחראי על הפרופיל

ניצן: אחראי על ההתחברות

---

## 5. דוגמה טכנית ל-TypeScript Interface (Data Structure)

בדי שתהיו 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
```

```
// אופציונלי - כי אולי אני סתם רוצה לשאול שאלה כללית
```

```
  ;plantId?: string
```

```
  ;type: PostType
```

```
  ;content: string
```

```
  ;[images: string
```

```
  ;createdAt: Date
```

```
{
```

## Tailwind (Mobile First) TIP:

נשאתם כותבים את ה-Classes, תמיד תכתבו קודם למובייל, ואז תוסיפו התניות למסך גדול. דוגמה לגריד של ה-Inventory:

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