

תרגיל פרויקט – תיבת תלונות אונימיות בסיסי

צבאי

פיתוח אפליקציית HTML סטטי, טפסים, בסיס נתונים, וארქיטקטורה נקייה +
בדיקות

רקע

זה"ל בוחן מערכת המאפשרת לחילאים לשלוח תלונות אונימיות באופן דיגיטלי. המערכת לא שומרת שמות, אלא רק את תוכן התלונה, תחום התלונה (אוכל, ציוד, פקודות, אחר) ואת התאריך. רק מפקדים עם סיסמה מתאימה יכולים לצפות ברשימה של התלונות.

מטרות

- להבין איך מגישים קבצי HTML עם Express.
- לעבד עם טפסים ולקראם נתונים מהגוף (`body`).
- לתרגל שמירה של מידע למסד נתונים בענן (Supabase או MongoDB).
- لتכון ולהפריד את הקוד לארכיטקטורה נכונה ומודולרית.
- לכתוב בבדיקות יחידות לשרת.

מבנה תיקיות (מולץ)

```
complaints-app/
  └── public/          ← HTML, CSS דפי ותמונות
    ├── index.html
    ├── complaint.html
    └── style.css

  └── routes/          ← Express-ניהול נתיבי ה
    └── complaints.js

  └── controllers/    ← לוגיקה עסקית
    └── complaintsController.js

  └── models/          ← מבני נתונים למסד
    └── complaint.model.js

  └── db/              ← קובץ חיבור למסד נתונים
    └── connect.js

  └── tests/           ← בדיקות
    └── complaints.test.js

  └── .env              ← משתנים רגיסטים (חיבור למסד וסיסמה)
  └── app.js            ← קובץ ראשי של השרת
  └── package.json      ← הגדרות התלוויות של הפרויקט
```

דרישות פונקציונליות

צד לקוח

- `:index.html`
 - תיאור קצר על מטרת המערכת.
 - כפתור/ קישור המוביל לדף שליחת תלונה.
 - סיסמה וכפתור שאמורים להוביל לדף אדמין (רק אם הסיסמה נכונה)
 - קישור לדף CSS עם עיצוב יפה. שימוש לב מה קורה בשרת כשתוענים את HTML ומקשור אליו דף CSS! תעשו LOG לכל בקשה כניסה בשרת ותראו.
- `:complaint.html`

- כוורת עמוד: "שליחת תלונה אונונימית"
- טופס הכלול:
 - שדה select לבחירת תחום (אוכל, ציוד, פיקודות, אחר)
 - שדה textarea לתוך התלונה
 - כפתור שליחה

צד שרת

- הגשה של הקבצים מתוך תיקיית `public`. ניתן להשתמש בFS או בהגשת קבצים סטטיים.
 - טיפול בנתיב `POST /submit` שמקבל את תוכן הטופס. ניתן לחלץ את הדטה שנשלחה באמצעות `app.use(express.urlencoded({ extended: true }))`:
 - שמירת כל תלונה במסד נתונים בענן (Supabase MongoDB או MongoDB) או :
 - שדות: `.category, message, createdAt`
 - יצירת עמוד צפיה בתלונות (`/admin`), שמצג רק אם נשלחת סיסמה נכונה. נגישות לעמוד זה נעשית מתוך דף הבית, עם טופס שմבקש סיסמה. אם הסיסמה לא נכונה - המשתמש יונוטב לדף עם הודעה שגיאה.
 - שמירה של הסיסמה בקובץ `.env` בלבד.
-

דרישות עיצוב לדפי HTML

- הדפים צריכים להיראות אחידים, נגישים, וקריאים:
- צבע רקע עדין (למשל `#f0f0f0`).
- קונטינר מרכזי עם רקע לבן, `padding` ו-`border-radius`.
- כוורות (`h1, h2`) צריכים להיות מיושרות למרכז.
- שדות הטופס חייבים להיות מסודרים בטור אחד.

- כפטור שליחת הינה בולט ומעוצב (למשלצבע ירוק או כחול).
 - יש לדאוג להסתrema למכשיר (responsive) באמצעות יחידות יחסיות
-

דרישות לבדיקות

1. **בדיקות יחידה (Unit Tests):**
 - בדיקה של יצירת תלונה במודל הנתונים.
 - בדיקה של פונקציה שימושיפה תלונה לבסיס הנתונים.
 2. **בדיקות אינטגרציה (Integration Tests):**
 - שליחת טופס דרך `submit` / `POST` ובדיקה שהتلונה נקלטה.
 - ניסיון לגשת ל-`/admin` עם סיסמה לא נכונה - צריך להחזיר את הדף של 403 (יכולים לבדוק גם את הקוד שמחזור).
 - ניסיון לגשת ל-`/admin` עם סיסמה נכונה → אמור להחזיר את רשימת התלונות.
-

בטחת מידע בסיסית

- אין לאחסן סיסמות בקוד! יש להשתמש בקובץ `.env` בלבד.
 - אין לחושף את URI של מודל הנתונים בקוד גלוי.
-

שלבי עבודה מוצעים

1. יצירת מבנה התיקיות והקבצים.
2. חיבור ל-`Supabase` או `MongoDB` (כולל `.env`).
3. יצירת המודל (`Sequelize` או `Mongoose`) - רשות.

4. בניית הקבצים הסטטיים (`index.html`, `complaint.html`, `style.css`).
 5. בניית הנתיבים (GET, POST, /admin).
 6. כתיבת בדיקות.
 7. הרצה מקומית ופתרון באגים.
-

אתגרים למתקדמים (לא חובה)

- יצירת JSON API מקביל ל-HTML.
 - הוספת תצוגה של גרפ שマーואה כמה תלונות היו בכל תחום.
 - אפשרות למחוק תלונה (רק דרך admin).
 - הוספת timestamp אנושי (למשל "נשלח לפני 3 ימים").
-