

תרגיל פרויקט – תיבת תלונות אנונימיות בבסיס

צבאי

פיתוח אפליקציית Node.js + Express עם HTML סטטי, טפסים, בסיס נתונים, וארכיטקטורה נקייה + בדיקות

רקע

צה"ל בוחן מערכת המאפשרת לחיילים לשלוח תלונות אנונימיות 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 ברשימת התלונות.

מטרות

- להבין איך מגישים קבצי HTML עם Express.
- לעבוד עם טפסים ולקרוא נתונים מהגוף (**body**).
- לתרגל שמירה של מידע למסד נתונים בענן (Supabase או MongoDB).
- לתכנן ולהפריד את הקוד לארכיטקטורה נכונה ומודולרית.
- לכתוב בדיקות ליחידות ולשרת.

מבנה תיקיות (מומלץ)

complaints-app/

| | |
|---------------------------|-------------------------------------|
| public/ | ← ותמונות HTML, CSS דפי |
| | |
| — 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 רגילים) חיבור למסד וסיסמה |
| app.js | ← קובץ ראשי של השרת |
| package.json | ← הגדרות התלויות 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. שימו לב מה קורה בשרת כשטוענים את הHTML ומקושר אליו דף CSS! תעשו LOG לכל בקשה נכנסת בשרת ותראו.

• `complaint.html`:

- כותרת עמוד: "שליחת תלונה אנונימית"
- טופס הכולל:
- שדה select לבחירת תחום (אוכל, ציוד, פקודות, אחר)
- textarea לתוכן 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 הדאטה שנשלח בעזרת:


```
app.use(express.urlencoded({ extended: true }));
```
- שמירת כל תלונה במסד נתונים בענן (Supabase או MongoDB):
 - שדות: `category, message, createdAt`.
- יצירת עמוד צפייה בתלונות (`admin/`), שמוצג רק אם נשלחת סיסמה נכונה. ניווט לעמוד הזה ייעשה מתוך דף הבית, עם טופס שמבקש סיסמה. אם הסיסמה לא נכונה - המשתמש ינותב לדף עם הודעת שגיאה.
- שמירה של הסיסמה בקובץ `env`. בלבד.

דרישות עיצוב לדפי HTML

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

- כפתור השליחה יהיה בולט ומעוצב (למשל בצבע ירוק או כחול).
 - יש לדאוג להתאמה למובייל (responsive) באמצעות יחידות יחסיות
-

דרישות לבדיקות

1. בדיקות יחידה (Unit Tests):

- בדיקה של יצירת תלונה במודל הנתונים.
- בדיקה של פונקציה שמוסיפה תלונה לבסיס הנתונים.

2. בדיקות אינטגרציה (Integration Tests):

- שליחת טופס דרך `POST /submit` ובדיקה שהתלונה נקלטה.
 - ניסיון לגשת ל-`admin/` עם סיסמה לא נכונה - צריך להחזיר את הדף של 403 (יכולים לבדוק גם את הקוד שמוחזר).
 - ניסיון לגשת ל-`admin/` עם סיסמה נכונה → אמור להחזיר את רשימת התלונות.
-

אבטחת מידע בסיסית

- אין לאחסן סיסמאות בקוד! יש להשתמש בקובץ `env` בלבד.
 - אין לחשוף את URI של מסד הנתונים בקוד גלוי.
-

שלבי עבודה מוצעים

1. יצירת מבנה התיקיות והקבצים.
2. חיבור ל-MongoDB או Supabase (כולל `env`).
3. יצירת המודל (Mongoose או Sequelize) - 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 API JSON מקביל ל-HTML.
 - הוספת תצוגה של גרף שמראה כמה תלונות היו בכל תחום.
 - אפשרות למחוק תלונה (רק דרך `admin`).
 - הוספת timestamp אנושי (למשל "נשלח לפני 3 ימים").
-