משימה: פיתוח מערכת לניהול רשימת ספרים עם Node.js, Express ו-TypeScript

---

תיאור הפרויקט:

אתם נדרשים לפתח יישום מלא לניהול רשימת ספרים אישית. היישום יאפשר למשתמשים להירשם, להתחבר ולנהל את רשימת הספרים שלהם. מכיוון שאין מנגנון טוקן או ניהול סשנים, המשתמשים יספקו את מזהה המשתמש שלהם (`userId`) עם כל בקשה הנוגעת לנתונים האישיים שלהם.

דרישות:

1. **הרשמה והתחברות:**

- משתמשים יכולים להירשם למערכת עם שם משתמש וסיסמה.
- הסיסמאות יישמרו באופן מוצפן באמצעות `bcrypt` .
- בעת ההרשמה, יוחזר למשתמש מזהה ייחודי (`userId`) שנוצר באמצעות `uuidv4` .
- המשתמשים צריכים לספק את ה- `userId` שלהם בכל בקשה כדי לזהות את עצמם.

2. **ניהול רשימת ספרים אישית:**

- משתמשים יכולים להוסיף ספרים לרשימה האישית שלהם.
- כדי להוסיף ספר, המשתמש יספק שם ספר, והשרת יפנה ל-API חיצוני (כגון Open Library API) כדי לקבל מידע נוסף על הספר.
- הספר יתווסף לרשימת הספרים של המשתמש ויישמר בקובץ JSON המקומי.
- משתמשים יכולים לצפות ברשימת הספרים שלהם באמצעות סיפוק ה- `userId` .
- משתמשים יכולים לעדכן או למחוק ספרים מהרשימה שלהם על ידי סיפוק `userId` ו- `bookId` .

3. **נקודות קצה בשרת:**

- `POST /register` – הרשמת משתמש חדש.
- גוף הבקשה: `{ "username": "שם משתמש", "password": "סיסמה" }`
- תגובה: `{ "userId": "מזהה-משתמש-ייחודי" }`
- `POST /login` – התחברות משתמש קיים.
- גוף הבקשה: `{ "username": "שם משתמש", "password": "סיסמה" }`
- תגובה: `{ "userId": "מזהה-משתמש-ייחודי" }` (אם ההתחברות הצליחה)
- `GET /books` – קבלת רשימת הספרים של המשתמש.
- פרמטרים של שאילתה: `?userId=מזהה-משתמש-ייחודי`
- תגובה: `[{ אובייקטים של ספרים }]`
- `POST /books` – הוספת ספר חדש לרשימה של המשתמש.
- גוף הבקשה: `{ "userId": "מזהה-משתמש-ייחודי", "bookName": "שם הספר" }`
- השרת יפנה ל-API חיצוני לקבלת מידע על הספר.
- תגובה: `{ "bookId": "מזהה-ספר-ייחודי", "book": { אובייקט ספר } }`

- `PUT /books/:bookId` – עדכון פרטי ספר.
  - גוף הבקשה: `{ "userId": "מזהה-משתמש-ייחודי", "updatedData": { ... } }`
- `DELETE /books/:bookId` – מחיקת ספר מהרשימה של המשתמש.
  - גוף הבקשה: `{ "userId": "מזהה-משתמש-ייחודי" }`

4. **בסיס הנתונים:**

- הנתונים יישמרו בקובץ JSON מקומי באמצעות `jsonfile` .
- הקובץ ישמור את פרטי המשתמשים ורשימות הספרים שלהם.

5. **פרונט-אנד:**

- ממשק משתמש שיאפשר הרשמה, התחברות וניהול רשימת הספרים.
- הממשק ידרוש מהמשתמש להזין את ה- `userId` שלו לאחר ההתחברות לפעולות נוספות.
- פיתוח ב-Vanilla JavaScript עם TypeScript.

6. **אינטגרציה עם API חיצוני:**

- השרת ישתמש ב-API חיצוני כמו Open Library API כדי לקבל פרטי ספרים.
- בעת הוספת ספר, השרת יביא מידע נוסף וישמור אותו.

7. **ספריות נוספות:**

- `bcrypt` להצפנת סיסמאות.
- `uuidv4` ליצירת מזהים ייחודיים.
- `jsonfile` לקריאה וכתיבה של נתונים לקובץ JSON.

הוראות מפורטות:

1. הקמת פרויקט השרת:

- אתחול פרויקט Node.js עם TypeScript:

```
mkdir book-list-app
cd book-list-app
npm init -y
npm install express bcrypt uuidv4 jsonfile axios
npm install -D typescript @types/express @types/node @types/bcrypt @types/uuidv4
@types/jsonfile @types/axios ts-node
npx tsc --init
```

- הגדרת הסקריפט `start` ב- `package.json` :

```
} :"scripts"
"start": "ts-node src/index.ts"
{
```

2. פיתוח השרת:

- **הרשמה (`POST /register`):**
  - קבלת `username` ו- `password` מגוף הבקשה.
  - הצפנת הסיסמה באמצעות `bcrypt` .
  - יצירת `userId` ייחודי עם `uuidv4` .
  - שמירת המשתמש בקובץ JSON.
  - החזרת `userId` למשתמש.
- **התחברות (`POST /login`):**
  - אימות שם משתמש וסיסמה.
  - השוואת הסיסמה באמצעות `bcrypt.compare` .
  - אם ההתחברות הצליחה, החזרת `userId` למשתמש.
- **Middleware לזיהוי משתמש:**
  - מכיוון שאין מנגנון טוקן, ודאו שכל בקשה המכילה נתונים אישיים כוללת את ה- `userId` בגוף הבקשה או בפרמטרי השאילתה.
  - וידוא קיום ה- `userId` בשרת ואימותו.
- **נקודות הקצה לניהול ספרים:**
  - `GET /books` **– קבלת רשימת הספרים של המשתמש:**
    - קבלת `userId` מפרמטרי השאילתה.
    - שליפת רשימת הספרים של המשתמש מהקובץ JSON.
    - החזרת הרשימה למשתמש.
  - `POST /books` **– הוספת ספר חדש:**
    - קבלת `userId` ו- `bookName` מגוף הבקשה.
    - שימוש ב- `axios` לפנייה ל-API החיצוני לקבלת פרטי הספר.
    - יצירת `bookId` ייחודי עם `uuidv4` .
    - הוספת הספר לרשימת הספרים של המשתמש בקובץ JSON.
    - החזרת פרטי הספר וה- `bookId` למשתמש.
  - `PUT /books/:bookId` **– עדכון ספר:**
    - קבלת `userId` מגוף הבקשה ו- `bookId` מפרמטרי הנתיב.
    - וידוא שהספר שייך למשתמש.
    - עדכון פרטי הספר בקובץ JSON.
  - `DELETE /books/:bookId` **– מחיקת ספר:**

- קבלת `userId` מגוף הבקשה ו- `bookId` מפרמטרי הנתיב.
- וידוא שהספר שייך למשתמש.
- מחיקת הספר מרשימת הספרים של המשתמש בקובץ JSON.

3. אינטגרציה עם API חיצוני:

- שימוש ב- `axios` לפנייה ל-Open Library API:

```
axios.get(`https://openlibrary.org/search.json?title=${bookName}`)
```

- עיבוד הנתונים המתקבלים ושמירתם (למשל, כותרת, מחבר, תמונת כריכה).

4. ניהול נתונים עם `jsonfile` :

- שימוש ב- `jsonfile` לקריאה וכתיבה לקובץ JSON (למשל, `database.json`).
- טיפול בשגיאות אפשריות בעת גישה לקובץ.

5. פיתוח הפרונט-אנד:

- **הרשמה והתחברות:**
  - יצירת טפסים לקבלת שם משתמש וסיסמה.
  - לאחר ההרשמה או ההתחברות, הצגת ה- `userId` למשתמש והסבר שעליו להשתמש בו בבקשות הבאות.
- **ניהול ספרים:**
  - טופס להזנת `userId` כדי לצפות ברשימת הספרים.
  - אפשרות להוסיף, לעדכן ולמחוק ספרים באמצעות הזנת `userId` ופרטי הספר.
- **TypeScript:**
  - שימוש ב-TypeScript עבור הסקריפטים בפרונט-אנד.
  - קומפילציה ל-JavaScript לצורך תאימות לדפדפנים.

6. שיקולי אבטחה:

- למרות שהעברת `userId` בבקשות אינה מאובטחת ליישום אמיתי, זה מספק למשימה זו.
- ודאו שבצד השרת נעשית ולידציה של `userId` ושמשתמשים אינם יכולים לגשת או לשנות נתונים של משתמשים אחרים.

הגשה:

- **מאגר Git:**
  - העלו את כל קבצי הפרויקט למאגר Git (GitHub, GitLab וכו').
  - הקפידו על הודעות קומיט משמעותיות ומסודרות.

- **קובץ README:**
  - כללו הוראות להרצת הפרויקט:
    - התקנת התלויות (`npm install`).
    - הרצת השרת (`npm start`).
    - כל הגדרה נוספת אם נחוצה.
  - תיאור קצר של הפרויקט והפונקציונליות שלו.

טיפים:

- **ארגון הקוד:**
  - שמרו על קוד נקי ומאורגן.
  - השתמשו בשמות משתנים ופונקציות ברורים.
- **TypeScript:**
  - הגדירו ממשקים (Interfaces) עבור נתוני משתמשים וספרים.
  - נצלו את היתרונות של TypeScript למניעת שגיאות.
- **בדיקות:**
  - בדקו את היישום בכל שלב.
  - ודאו שכל נקודות הקצה עובדות כמצופה.
  - בדקו שהנתונים האישיים מנוהלים כראוי.
- **חוויית משתמש:**
  - מכיוון שהמשתמש צריך להזין את ה- `userId` שלו לעיתים קרובות, שקלו דרכים להפוך זאת לידידותי יותר בממשק (לדוגמה, שמירת ה- `userId` ב-Local Storage).

הערה:

ביישום אמיתי, היינו משתמשים בטוקנים או מנגנון סשנים לניהול אימות משתמשים בצורה מאובטחת. העברת מזהה המשתמש בבקשות אינה מומלצת ליישומים בייצור. עם זאת, לצורך המשימה, זה מקובל בשל המגבלות.

בהצלחה!