אפיון פרויקט: פלטפורמת בלוג

תיאור כללי

המשימה היא לפתח פלטפורמת בלוג בסיסית המאפשרת למשתמשים ליצור פוסטים ולהגיב עליהם. הפרויקט מחולק למספר חלקים, אתם תקבלו את הקוד הבסיסי עם קבצי קונטרולרים וסכימות ריקים. עליהם למלא את הקוד החסר ולהשלים את הפונקציונליות הנדרשת.

מטרות הפרויקט

- **יצירת משתמשים**: אפשרות ליצור משתמשים חדשים עם שם משתמש, אימייל ופרופיל.
- **ניהול פוסטים**: משתמשים יכולים ליצור, לקרוא, לעדכן ולמחוק פוסטים.
- **הוספת תגובות**: משתמשים יכולים להוסיף תגובות לפוסטים קיימים.
- **קשרים בין ישויות**:
  - משתמש מחזיק מערך של פוסטים (רפרנסים).
  - פוסטים מכילים מערך של תגובות (מקוננות בתוך הפוסט).

מה מצפים מהתלמידים

- **מילוי הסכימות**: על התלמידים להשלים את הסכימות של המשתמשים והפוסטים, כולל הגדרת השדות, הסוגים והקשרים ביניהם.
- **מילוי הקונטרולרים**: עליהם לכתוב את הלוגיקה בכל פונקציות הקונטרולרים כדי לממש את הפעולות הנדרשות (CRUD לפוסטים, יצירת משתמשים, הוספת תגובות וכו').
- **שימוש ב-`populate`**: במקומות המתאימים, יש להשתמש ב-`populate` כדי להביא את הנתונים המלאים מהקשרים (למשל, כאשר שולפים פוסט, להביא את פרטי המחבר והתגובות עם פרטי המגיבים).
- **ולידציות בסיסיות**: להוסיף ולידציות בסיסיות לשדות (כגון שדות נדרשים, אורך מינימלי/מקסימלי, אימייל תקין וכו').
- **ניהול שגיאות**: לטפל בשגיאות ולשלוח תגובות מתאימות ללקוח.

מבנה הפרויקט

- **מודלים (`models`)**:
  - `userModel.ts`: סכימת המשתמשים.
  - `postModel.ts`: סכימת הפוסטים.
- **קונטרולרים (`controllers`)**:
  - `userController.ts`: פונקציות לטיפול בפעולות המשתמשים.

- `postController.ts` : פונקציות לטיפול בפעולות הפוסטים.
- **נתיבים (`routes`)**:
  - `userRoutes.ts` : נתיבי המשתמשים.
  - `postRoutes.ts` : נתיבי הפוסטים.

פרטים על הארכיטקטורה

- **משתמש (`User`)**:
  - שדות:
    - `username` : מחרוזת ייחודית ונדרשת.
    - `email` : מחרוזת ייחודית ונדרשת.
    - `profile` : אובייקט המכיל `bio` ו- `socialLinks` .
    - `posts` : מערך של `ObjectId` המצביעים על פוסטים (רפרנס ל- `Post`).
- **פוסט (`Post`)**:
  - שדות:
    - `title` : מחרוזת נדרשת.
    - `content` : מחרוזת נדרשת.
    - `author` : `ObjectId` המצביע על משתמש (רפרנס ל- `User`).
    - `comments` : מערך של תגובות (מקונן בתוך הפוסט).
- **תגובה (`Comment`)**:
  - שדות:
    - `content` : מחרוזת נדרשת.
    - `author` : `ObjectId` המצביע על משתמש (רפרנס ל- `User`).
    - `createdAt` : תאריך יצירת התגובה.

דגשים למימוש

- **קשרים בין ישויות**:
  - בעת יצירת פוסט חדש, יש להוסיף את ה- `ObjectId` של הפוסט למערך `posts` של המשתמש שיצר אותו.
  - תגובות מקוננות בתוך הפוסט עצמו ואינן נשמרות כאובייקטים נפרדים במסד הנתונים.
- **שימוש ב- `populate`** :

- כאשר שולפים פוסט או פוסטים, יש להשתמש ב- `populate` כדי להביא את פרטי המחבר (`author`) ואת פרטי המגיבים בתוך התגובות.
- לדוגמה:

```
Post.find().populate({
  path: "author",
  select: "username email profile",
}).populate({
  path: "comments.author",
  select: "username",
});
```

- **ולידציות**:
  - להוסיף ולידציות לשדות, כגון:
    - שדות נדרשים (`required`).
    - אורך מינימלי ומקסימלי (`minlength` , `maxlength`).
    - פורמט תקין לאימייל (שימוש ב-validator).
    - בדיקת ייחודיות (`unique`) לשדות כמו `username` ו- `email` .
- **טיפול בשגיאות**:
  - לשלוח תגובות עם סטטוס מתאים (לדוגמה, 400 עבור בקשות לא תקינות, 404 כאשר לא נמצא וכו').
  - להחזיר הודעות שגיאה מפורטות שיעזרו בפתרון הבעיה.

דוגמאות לבקשות ותגובות

- **יצירת משתמש חדש**:
  - **בקשה**:

```
POST /users
Content-Type: application/json

{
  "username": "john_doe",
  "email": "john@example.com",
  "profile": {
    "bio": "Software developer",
    "socialLinks": ["https://twitter.com/john_doe"]
  }
}
```

  - **תגובה צפויה**:

```
{
  "_id": "ObjectId",
  "username": "john_doe",
  "email": "john@example.com",
  "profile": {
    "bio": "Software developer",
    "socialLinks": ["https://twitter.com/john_doe"]
  },
  "posts": [],
  "__v": 0
}
```

- **יצירת פוסט חדש**:
  - **בקשה**:

```
POST /posts
Content-Type: application/json

{
  "title": "My First Post",
  "content": "This is the content of my first post.",
  "author": "User ObjectId"
}
```

  - **תגובה צפויה**:

```
{
  "_id": "ObjectId",
  "title": "My First Post",
  "content": "This is the content of my first post.",
  "author": {
    "_id": "User ObjectId",
    "username": "john_doe",
    "email": "john@example.com",
    "profile": { ... }
  },
  "comments": [],
  "__v": 0
}
```

מה מצפים לקבל ולהחזיר

- **בקשות**:

- הבקשות צריכות להכיל את כל השדות הנדרשים.
- יש להשתמש ב-HTTP methods המתאימים (GET, POST, PUT, DELETE).
- **תגובות**:
  - תגובות צריכות להיות בפורמט JSON.
  - יש להחזיר את הנתונים שנוצרו או התעדכנו, כולל הקשרים (לאחר שימוש ב- `populate`).
  - במקרה של שגיאות, להחזיר אובייקט עם הודעת השגיאה וקוד סטטוס מתאים.

הנחיות נוספות

- **סדר עבודה מומלץ**:
  - מילוי הסכימות והגדרת הקשרים בין המודלים.
  - מימוש פונקציות הקונטרולרים לכל פעולה (CRUD).
  - הוספת ולידציות לשדות.
  - טיפול בשגיאות והחזרת תגובות מתאימות.
  - בדיקת הפונקציונליות באמצעות כלי כמו Postman.
- **כלים מומלצים**:
  - **Postman**: לבדיקה ושליחת בקשות לשרת.
  - **MongoDB Compass**: לבדיקת הנתונים במסד הנתונים.
- **טיפים**:
  - לשים לב לדיוק בשמות השדות והקשרים.
  - להשתמש ב- `async/await` כדי לטפל בפעולות אסינכרוניות.
  - לוודא שכל ה- `ObjectId` תקינים ונכונים.