StickyNotes

**תכנון מבנה המערכת:**

1. **-Backend** (C#.NET Core): פיתוח עם API RESTFUL שיטפל בבקשות מ-Frontend וישמור את הפתקים בבסיס נתונים. ה-API יכול לכלול פונקציות כמו הוספת פתקים, עדכון, מחיקה ושליפה.

**תכנים עיקריים**:

* **Create Note (POST /api/notes)**: יצירת פתק חדש.
* **Get Notes (GET /api/notes)**: הצגת כל הפתקים של המשתמש.
* **Update Note (PUT /api/notes/{id})**: עדכון פתק קיים.
* **Delete Note (DELETE /api/notes/{id})**: מחיקת פתק קיים.

1. **חיבור לבסיס נתונים**: שימוש ב-SQL Server כמסד נתונים וליצור טבלאות שמכילות את הפתקים (כולל שדות כמו תוכן, תאריך יצירה, תאריך עדכון וכו').

**מבנה טבלאות**:

* **Notes**: טבלה לשמירת פרטי הפתקים (ID, Title, Content, CreatedAt, UpdatedAt, UserId).
* **Users**: טבלה לשמירת פרטי המשתמשים (ID, Name, Email, Password).

)**הוספת פתרונות CACHE בעת הצורך**:

**Redis Cache**: Cache יכול לעזור לשפר ביצועים על ידי אחסון תוצאות של שאילתות נפוצות כדי למנוע גישה לבסיס הנתונים בכל פעם מחדש. זה יעזור במיוחד כאשר מספר הפתקים גדל או כאשר יש תעבורה גבוהה.)

.

1. **שירותי AI לכתיבה/ניסוח מחדש**: הטמעת שירותי AI - OpenAI המתמקדים בניתוח טקסטים ושיפורם,אשתמש בספריית http client ב. NET דרך דרכה ניתן לשלוח בקשה לapi ולבצע ניסוח מחדש של הטקסט

**שימוש ב-OpenAI API**:

* כאשר המשתמש רוצה לשפר או לשכתב טקסט, נשלח בקשה ל-OpenAI API דרך HTTP

client של .NET.

* יישום שימוש ב-**HttpClient** לשליחת הבקשות ותשובות מה-API של OpenAI. תשובות אלו ישמשו להצעות לשכתוב או שיפור תוכן הפתק.

**השלבים**:

* המשתמש יכנס עם פתק, ולאחר מכן יוכל לבחור להפעיל את שירות השכתוב.
* המערכת תשלח את הטקסט ל-OpenAI API ותקבל תשובה עם גרסה משופרת או משוכתבת של הטקסט.

1. **אבטחת המערכת**: תכנון מערכת גישה שמאמתת את המשתמשים ומגנה על המידע שנשמר.

 **JWT (JSON Web Tokens)**:

* המערכת תשתמש ב-JWT לאימות משתמשים ונתונים.
* כל בקשה ל-API תדרוש טוקן JWT בתור Authorization header.
* יש לוודא שכל משתמש מקבל טוקן JWT לאחר התחברות. כל פנייה לאחר מכן ל-API תדרוש את הטוקן כדי לבדוק את האותנטיקציה של המשתמש.

**הגנת מידע**:

* הצפנת סיסמאות עם ספריות כמו BCrypt או ASP.NET Core Identity
* יישום מדיניות גישה כדי לוודא שרק המשתמשים המורשים יכולים לגשת לנתונים שלהם.

1. **Frontend (Angular):**

**ממשק משתמש**:

* תכני ממשק המשתמש יאפשרו למשתמשים ליצור, לערוך, לשלוף ולמחוק פתקים.
* יש להוסיף תכונה המאפשרת ניסוח מחדש של הטקסט בעזרת ה-API של OpenAI.
* כל פעולה כמו יצירת פתק, מחיקה או עדכון תשלח בקשה לAPI.

**שימוש ב-AI כמו הצעות לשיפור טקסט**

* הממשק יכלול אפשרות שמציעה למשתמש ניסוח מחדש של הטקסט על ידי קישור לפונקציה המתקשרת עם OpenAI. המשתמש יוכל לבחור אם להיעזר בהצעות השכתוב או לא.

1. **בדיקות**: בדיקות יחידה ובדיקות אינטגרציה לכל רכיב במערכת.

 **בדיקות יחידה**:

* יש לבצע בדיקות יחידה (Unit Tests) על כל רכיב במערכת. כל API צריך לעבור סדרת בדיקות על מנת לוודא שהפונקציות פועלות כראוי.
* דוגמה: בדיקה על פונקציות ה-POST/GET/PUT/DELETE של ניהול הפתקים.

 **בדיקות אינטגרציה**:

* יש לוודא שהמערכת כולה פועלת בצורה אינטגרטיבית, כלומר שהשירותים עובדים כראוי יחד.
* לדוגמה, לוודא שהטוקן JWT מועבר כראוי ומאומת בכל הבקשות בAPI

**.7 פריסה לענן**

* יש לתכנן כיצד המערכת תופרש בשלב מאוחר יותר, במקרה של גידול במשתמשים או נתונים.
* פתרונות ענן כמו **Azure** או  **AWS** מאפשרים גמישות עם הגדלת משאבים באופן דינאמי.
* תכנון למערכת  **CI/CD** על מנת להבטיח עדכונים אוטומטיים.

**1.תכנון ה-Backend (C# + .NET Core)**

על ה-API לעבד את הבקשות מה-Frontend ולבצע את הפעולות הנדרשות על הפתקים.

API:

| **פעולה** | **סוג בקשה** | **נתיב** | **תיאור** |
| --- | --- | --- | --- |
| קבלת כל הפתקים | GET | /api/notes | שליפת כל הפתקים |
| קבלת פתק לפי ID | GET | /api/notes/{id} | שליפת פתק ספציפי |
| יצירת פתק חדש | POST | /api/notes | הוספת פתק חדש |
| עדכון פתק | PUT | /api/notes/{id} | עדכון תוכן של פתק |
| מחיקת פתק | DELETE | /api/notes/{id} | מחיקת פתק |
| **כתיבת פתק עם AI** | POST | /api/notes/generate | יצירת תוכן בעזרת AI |
| **ניסוח מחדש של פתק** | POST | /api/notes/rewrite | שימוש ב-AI לניסוח מחדש |

**2.תכנון בסיס הנתונים (SQL Server)**

כדי ליישם אימות (Authentication) ו-הרשאות גישה (Authorization), נצטרך להוסיף טבלת משתמשים וליצור קשר בין המשתמשים לפתקים.

**טבלת משתמשים (Users)**

נוסיף טבלת Users שתשמור את פרטי המשתמשים.

מבנה הטבלה:

| **עמודה** | **סוג נתון** | **תיאור** |
| --- | --- | --- |
| **Id** | **int (Primary Key)** | **מזהה ייחודי למשתמש** |
| **Username** | **nvarchar(100)** | **שם משתמש** |
| **Email** | **nvarchar(255)** | **אימייל ייחודי** |
| **PasswordHash** | **nvarchar(255)** | **סיסמה מוצפנת** |
| **Role** | **nvarchar(50)** | **תפקיד המשתמש (User, Admin)** |
| **CreatedAt** | **datetime** | **תאריך יצירה** |

קשר בין משתמשים לפתקים כל פתק יהיה משויך למשתמש שיצר אותו.  
לכן, נוסיף עמודת UserId לטבלת Notes.

**טבלת הפתקים**

| **עמודה** | **סוג נתון** | **תיאור** |
| --- | --- | --- |
| **Id** | **bigint (Primary Key)** | **מזהה ייחודי לפתק** |
| **Title** | **nvarchar(255)** | **כותרת הפתק** |
| **Content** | **nvarchar(MAX)** | **תוכן הפתק** |
| **UserId** | **int (Foreign Key)** | **מזהה המשתמש שיצר את הפתק** |
| **CreatedAt** | **datetime** | **תאריך יצירה** |
| UpdatedAt | datetime | תאריך עדכון |

**. 3 שירותי AI לניסוח מחדש**

שימוש בשירות OpenAI כדי לבצע ניסוח מחדש של טקסטים

**4 .אבטחת המערכת**

מימוש מערכת אימות על ידי JWT (JSON Web Tokens) לכל הבקשות המגיעות מה-Frontend כדי להבטיח שהגישה למידע תהיה רק למי שיש לו הרשאות מתאימות

שלבים בפיתוח האימות

* יצירת API להרשמה (/api/auth/register) – שמירת משתמש חדש.
* יצירת API להתחברות (/api/auth/login) – אימות משתמש + יצירת טוקן JWT.
* יצירת API ל "שכחתי סיסמא"
* הוספת Middleware לניתוב בקשות לפי הרשאות משתמשים.
* הוספת Authorization כך שכל משתמש יכול לגשת רק לפתקים שלו.

**Frontend – Angular. 5**

נתחיל בבניית ה-UI שבו המשתמשים יוכלו ליצור לערוך פתקים. הממשק יהיה פשוט וכולל את הפעולות הבסיסיות כמו הוספה, עדכון, מחיקה, והצגת פתקים קיימים.

1.המסך הראשי:

* **רשימת פתקים**: תצוגה של פתקים קיימים עם כותרת ותוכן (תצוגת הפתקים יהיו ממויינים לפי סדר יצירתם הכי חדש בראש המסך)
* כל פתק יכול להציג את תאריך היצירה ותוכן הטקסט.
* כפתור **יצירת פתק חדש**: בראש הדף, כפתור “צור פתק חדש” שיפתח טופס ליצירה.
* כפתור “**ניסוח מחדש**”: כפתור מתחת לכל פתק, אשר ישלח את הטקסט לשירות ה-AI כדי לשדרג או לשנות את הניסוח.
* **כפתור "מחיקה**"- נוסיף כפתור מחיקה ליד כל פתק ברשימה שיבצע את פעולות המחיקה כאשר נלחץ עליו ,כל פתק יוכל להימחק באמצעות כפתור זה והמערכת תעדכן את רשימת הפתקים על פי השינויים.
* **שדה חיפוש** –בראש הדף יהיה שדה חיפוש שיאפשר למשתמש לחפש פתקים לפי מילות מפתח בכותרת או בתוכן .(החיפוש יפעל בזמן אמת,כלומר ברגע שהמשתמש יקליד בטקסט, המערכת תסנן את הפתקים ותציג רק את הפתקים התואמים לחיפוש.)

2. יצירת פתק/עריכה:

• שדות טקסט: שדה כותרת ושדה תוכן.

• כפתור “שמור”: לחיצה על הכפתור תיצור או תעדכן את הפתק.

• כפתור “בטל”: יציאה מהמסך ללא שמירה.

3. עיצוב:

• צבעים רכים ונעימים לעיניים

• כפתורים גדולים וברורים עם עיצובים מינימליסטיים.

**רשימת פתקים**

חיפוש פתק

הקלד/י טקסט כאן

צור פתק חדש

14.02.25

**קניות**

חלב

לחם

טיטולים

ניסוח מחדש

מחיקה

13.02.25

**תזכורות כללי:**

-לקבוע תור לטיפת חלב

-להירשם לגן

-לחדש ביטוח לרכב

מחיקה

ניסוח מחדש

בלחיצה על מחיקה יופיע המסך הבא :

בלחיצה על מחיקה :

האם את/ה בטוח/ה שברצונך למחוק פתק זה ?

(לאחר מחיקה לא יתאפשר לשחזר פתק זה )

לא

כן

בלחיצה על "צור פתק חדש"

התאריך ישתל אוטומטית

כותרת

תוכן הפתק

שמור

בטל

**6 . בדיקות**

בדיקות לכל רכיב במערכת, כולל:

**בדיקות יחידה**: לכל פונקציה במערכת (כמו יצירת פתק, עדכון, ניסוח מחדש של טקסט).

**בדיקות אינטגרציה**: לבדוק איך כל הרכיבים עובדים יחד (Frontend, Backend ו-AI).

**בדיקות יחידה**-

**.1בדיקות לשירות ניהול פתקים (NoteService)**

**מטרת הבדיקות**: לבדוק שהלוגיקה של ה-Service עובדת נכון  
**נבדוק:**  
✔️ שליפת פתק לפי מזהה  
✔️ הוספת פתק חדש  
✔️ עדכון פתק קיים  
✔️ מחיקת פתק

**רשימת טסטים לדוגמא:**

| **#** | **בדיקה** | **קלט** | **תוצאה צפויה** |
| --- | --- | --- | --- |
| 1 | שליפת פתק קיים | Id=1 | מחזיר פתק קיים |
| 2 | שליפת פתק לא קיים | Id=99 | מחזיר null |
| 3 | הוספת פתק חדש | {Title: "Test", Content: "Hello"} | הפתק נוסף בהצלחה |
| 4 | עדכון פתק קיים | {Id:1, Title: "Updated"} | הפרטים מתעדכנים |
| 5 | מחיקת פתק קיים | Id=1 | הפתק נמחק |

**. 2בדיקות לשירות אימות משתמשים (AuthService)**

**מטרת הבדיקות**: לבדוק אם המשתמשים עוברים תהליך התחברות נכון  
**נבדוק:**  
✔️ התחברות עם פרטי משתמש תקינים  
✔️ התחברות עם שם משתמש שגוי  
✔️ התחברות עם סיסמה שגויה  
✔️ הרשאה נכונה בטוקן

**רשימת טסטים לדוגמא:**

| **#** | **בדיקה** | **קלט** | **תוצאה צפויה** |
| --- | --- | --- | --- |
| 1 | התחברות עם פרטי משתמש נכונים | Username: test, Password: 1234 | מחזיר טוקן |
| 2 | התחברות עם שם משתמש לא קיים | Username: unknown | מחזיר שגיאה |
| 3 | התחברות עם סיסמה לא נכונה | Username: test, Password: wrong | מחזיר שגיאה |
| 4 | בדיקת הרשאות בטוקן | Token: valid | הרשאה תקינה |

**.3 בדיקות למערכת מסד הנתונים (Repository Layer)**

**מטרת הבדיקות**: לוודא שה-Repository מתקשר נכון עם ה-Database  
**נבדוק:**  
✔️ הוספה ושמירה של נתונים במסד  
✔️ שליפת נתונים  
✔️ מחיקת נתונים

**רשימת טסטים לדוגמא:**

| **#** | **בדיקה** | **קלט** | **תוצאה צפויה** |
| --- | --- | --- | --- |
| 1 | שמירת פתק חדש במסד | {Title: "DB Test"} | הפתק נשמר |
| 2 | שליפת פתק מהמסד | Id=1 | מחזיר פתק קיים |
| 3 | מחיקת פתק | Id=1 | הפתק נמחק מהמסד |

**4.תכנון בדיקות עומסים (Load Testing) למערכת הפתקים**

**מטרת הבדיקות: לוודא שהמערכת עומדת בעומס גבוה של בקשות ולמנוע קריסות.  
נבדוק:  
✔️ כמה בקשות המערכת יכולה לטפל בו-זמנית  
✔️ מה זמן התגובה תחת עומס כבד  
✔️ איך המערכת מתמודדת עם שימוש אינטנסיבי (עשרות/מאות משתמשים בו-זמנית)  
✔️ האם יש נזילות זיכרון או קריסות**

**בדיקות אינטגרציה-**

**.1 בדיקות עומסים לשירות ניהול פתקים (Notes API)**

**בדיקות לדוגמא:**

| **#** | **בדיקה** | **כמות בקשות** | **תוצאה צפויה** |
| --- | --- | --- | --- |
| **1** | **קריאה של פתק יחיד בו-זמנית** | **100 בקשות לשנייה** | **זמני תגובה תקינים** |
| **2** | **הוספת פתקים מרובים** | **500 פתקים במקביל** | **כל הפתקים נשמרים בלי שגיאות** |
| **3** | **מחיקה בו-זמנית** | **50 מחיקות במקביל** | **זמן תגובה תקין** |
| **4** | **עדכון פתקים במקביל** | **200 עדכונים בשנייה** | **ללא קריסה** |

**איך לבדוק?**

**להריץ בדיקות עם משתמשים וירטואליים רבים ולמדוד ביצועים**

**. 2 בדיקות עומסים למסד הנתונים (Database Load)**

**בדיקות לדוגמא:**

| **#** | **בדיקה** | **כמות בקשות** | **תוצאה צפויה** |
| --- | --- | --- | --- |
| **1** | **הוספה וקריאה של פתק במקביל** | **1000 שאילתות בו-זמנית** | **זמן תגובה תקין** |
| **2** | **קריאה של כל הפתקים** | **100 שאילתות SELECT** | **ביצועים תקינים** |
| **3** | **עומס גבוה ממספר משתמשים** | **1000 משתמשים מחוברים** | **ללא נפילות** |

**איך לבדוק?**

**להפעיל שאילתות כבדות ולבדוק זמן תגובה**

**להשתמש בכלים כמו SQL Profiler כדי לנתח ביצועים**

**.3 בדיקות עומסים לאימות משתמשים (Auth Load Testing)**

**בדיקות לדוגמא:**

| **#** | **בדיקה** | **כמות בקשות** | **תוצאה צפויה** |
| --- | --- | --- | --- |
| **1** | **התחברות משתמשים** | **1000 התחברויות בשנייה** | **זמני תגובה תקינים** |
| **2** | **יצירת טוקנים** | **500 יצירות טוקן בשנייה** | **ללא קריסות** |
| **3** | **בדיקת הרשאות תחת עומס** | **1000 קריאות API עם טוקנים** | **ללא שגיאות הרשאה** |

**איך לבדוק?**

**שימוש בכלי עומסים כמו Gatling או k6**

**לוודא שהשרת לא קורס תחת כמות התחברויות גדולה**