**מסמך אפיון-תמונות!!!!**

1 . תאור המערכת:

**1.1.מטרת הפרויקט-**

האפליקציה נועדה לאפשר למשתמשים לנהל אלבומי תמונות בצורה נוחה ואינטואיטיבית. היא תאפשר העלאת תמונות, ארגון באלבומים, שיתוף עם אחרים וגיבוי אוטומטי בענן. המטרה היא לפשט את ניהול התמונות האישיות ולהפוך את החוויה למהנה יותר.

**1.2.קהל היעד-**

קהל היעד העיקרי של האפליקציה הוא אמהות לילדים, משפחות שרוצות לנהל את התמונות האישיות שלהן בצורה מסודרת, וחברות קטנות שצריכות לנהל גלריות תמונות לפרויקטים.

**1.3. פונקציונליות של המערכת**

1. העלאת תמונות לענן.

2. יצירת אלבומים וארגון תמונות בתוכם.

3. שיתוף אלבומים עם משתמשים אחרים.

4. חיפוש תמונות לפי תגיות או תאריכים.

5. גיבוי אוטומטי של תמונות.

6. יצירת קולאזים

**1.4 –בעיות שהמערכת פותרת-**

האפליקציה פותרת את הבעיה של ניהול תמונות מבולגן ומפוזר בין מכשירים שונים. היא מאפשרת גישה נוחה לכל התמונות ממקום אחד, מבטיחה גיבוי אוטומטי כדי למנוע אובדן תמונות, ומאפשרת שיתוף קל עם אחרים.

**1.5-טכנולוגית בשימוש-**

.1 **צד שרת (Backend):** NET 9. לבניית ה-API.

.2 **צד לקוח (Frontend):** React.js לבניית ממשק משתמש אינטראקטיבי.  
 **.3 מסד נתונים MY**SQLלאחסון נתונים של משתמשים ותמונות.  
.4 **אחסון קבצים:** שימוש ב-Amazon S3 לאחסון תמונות.

**1.6-מתודולוגית פיתוח-**

הפרויקט יפותח במתודולוגיית Agile עם חלוקה ל-5 ספרינטים בני שבועיים כל אחד. כל ספרינט יתמקד במשימות ליבה עיקריות. להלן פירוט הספרינטים הצפויים:

**ספרינט 1 - בניית תשתיות וניהול משתמשים:**

* **תכנון מודל מסד הנתונים ב-PostgreSQL לטבלאות משתמשים ואלבומים.**
* **פיתוח API בסיסי ב-.NET 9 עם endpoints לניהול משתמשים ואלבומים.**
* **הטמעת מערכת אימות משתמשים (Registration/Login) עם JWT.**
* **הגדרת אחסון תמונות ב-Amazon S3.**

**ספרינט 2 - ממשק משתמש ראשוני:**

* + **בניית רכיבי React.js לעיצוב אלבומים וגלריות.**
  + **אינטגרציה בין Frontend ל-API באמצעות Axios.**
  + **הטמעת העלאת תמונות בסיסית עם Drag & Drop.**

**ספרינט 3 - פיצ'רים ליבה:**

* + פיתוח מערכת שיתוף אלבומים עם אפשרויות גישה
  + הטמעת חיפוש תמונות לפי תגיות או תאריכים.
  + יצירת ממשק ניהול אלבומים **עם אפשרויות עריכה ומחיקה**

**ספרינט 4 – שיפורים ואבטחה**

* + הטמעת גיבוי אוטומטי יומי לתמונות
  + הוספת אופציות סינון מתקדמות (למשל לפי טקסט חופש)
  + בדיקת אבטחה ואפטימזציה ב API
  + פריסת הפליקציה בשרת AWS
  + הגדרת דומיין אישי
  + עיצוב דף הבית לשיווק ומשיכת משתמשים פוטנציאיליים

2 . ממשק ויזואלי:

עמוד התחברות-לחיצה על SignIn יעביר אותו להתחברות שתכניס אותו לאזור האישי

לחיצה על sigup תעביר אותו להרשמה שיוכל להתחיל אזור אישי משלו

**תמונה שמכילה טקסט, גופן, צילום מסך, מספר

התיאור נוצר באופן אוטומטי**

**עמוד האזור האישי- לכל אחד יהיה אזור אישי משלו שיכול לצפות בתוכן שלו יהיה לו אפשרות להעלות תמונה מהמחשב** לחיצה על 'העלאת קובץ' תפתח דיאלוג לבחירת קובץ מהמחשב ולכל תמונה יהיה **אופציה למחיקת התמונה**

**תמונה שמכילה טקסט, צילום מסך, תרשים, גופן

התיאור נוצר באופן אוטומטי**

3 . אפיון תמציתי:

**3.1 רשימת הפונקציות המערכת:**

**1רישום משתמש**

* **שם הפונקציה:** רישום משתמש חדש למערכת.
* **Route:** POST /user/register
* **פרמטרים:**
  + Body: { username: string, password: string, email: string, role: "teacher" | "student" }
* **פלט:**
  + data: { userId: string, username: string, email: string, role: string }
  + קודי סטטוס: 201 (נוצר), 400 (בקשה שגויה), 409 (משתמש קיים).
* **לוגיקה עסקית:**
  + בדיקה אם שם משתמש או אימייל כבר קיימים.
  + גיבוב סיסמה לפני שמירה במסד הנתונים.
  + יצירת משתמש לפי סוג משתמש (מנהל/משתמש).

**2. התחברות משתמש**

* **שם הפונקציה:** התחברות משתמש קיים למערכת.
* **Route:** POST /api/login
* **פרמטרים:**
  + Body: { username: string, password: string }
* **פלט:**
  + data: { token: string, userId: string, role: string }
  + קודי סטטוס: 200 (אישור), 401 (לא מורשה).
* **לוגיקה עסקית:**
  + אימות שם משתמש וסיסמה.
  + יצירת JSON Web Token (JWT) לאימות בקשות עתידיות.

**3 העלאת תמונה**

Route:  
POST /api/upload-image

פרמטרים:  
{ "file": File, "tags": string[], "album": string } body:

פלט:  
{ "fileId": string, "url": string, "message": "File uploaded successfully" } data:

קודי סטטוס:

* Ok 200: אם התמונה הועלתה בהצלחה.
* Bad-Request 400: אם קובץ לא תואם לפורמט הרצוי (למשל, לא תמונה).
* Payload Too Large 413: אם גודל הקובץ גדול מהמגבלה(יותר מ-50MB )

לוגיקה עסקית:

* הגבלת גודל הקובץ ל-50MB.
* אם לא נבחרו תגים, מערכת תוסיף תגים אוטומטיים (ע"י AI API חיצוני )
* התמונה תישמר ב Google Cloud Storage -, עם חתימת זמן ייחודית לכל קובץ לשם אבחנה.

4 **חיפוש תמונה לפי תיוג**

Route**:**  
GET /api/search-images-by-tag

פרמטרים:  
{ "tag": string } query params:

פלט:  
 data: { "images": [ { "fileId": string, "url": string, "tags": string[], "uploadDate": string } ], "message": "Search results retrieved successfully" }

קודי סטטוס:

* Ok 200: אם התוצאות נמצאו והן מוצגות כראוי.
* Bad Request 400: אם פרמטרים לא תקינים.
* Not Found 404 : אם לא נמצאו תמונות תואמות לתיוג.

לוגיקה עסקית:

* המערכת תבצע חיפושחלקי של התיוגים (למשל, אם המשתמש הקליד "חת", המערכת תחפש את כל התמונות עם התיוג "חתולה").
* החיפוש יתבצע גם ב Google Cloud Storage וגם במסד הנתונים כדי לאחזר תמונות עם פרטי התיוג והזמן שלהן.

**5 חיפוש תמונה לפי תאריך**

Route:  
GET /api/search-images-by-date

פרמטרים:  
{ "startDate": string (yyyy-mm-dd), "endDate": string (yyyy-mm-dd) } query params:

פלט:  
data: { "images": [ { "fileId": string, "url": string, "tags": string[], "uploadDate": string } ], "message": "Search results retrieved successfully" }

קודי סטטוס:

* Ok 200: אם התוצאות נמצאו והן מוצגות כראוי.
* Bad Request 400: אם פרמטרים לא תקינים.
* Not Found 404 : אם לא נמצאו תמונות בתאריכים המבוקשים.

לוגיקה עסקית:

* חיפוש תמונה על פי תאריכים שצוינו (startDate ו- endDate)

**6 חיפוש תמונה על פי טקסט חופשי באמצעות AI**

Route:  
GET /api/search-images-by-ai

פרמטרים:  
{ "query": string } query params:

פלט:  
data: { "images": [ { "fileId": string, "url": string, "tags": string[], "uploadDate": string } ], "message": "Search results retrieved successfully" }

קודי סטטוס:

* Ok 200: אם התוצאות נמצאו ואלגוריתם ה-AI חיפש בהצלחה.
* Bad Request 400: אם פרמטרים לא תקינים (למשל, אם לא הוזן טקסט).
* Not Found 404 : אם לא נמצאו תמונות התואמות לבקשה.

לוגיקה עסקית:

* החיפוש יתבצע בעזרת GoogleCloudVisionAPI או AWS Rekognitionאו כאשר הטקסט החופשי שהוזן ישמש כבסיס לחיפוש תמונות תואמות.
* AI מנתח את הטקסט ומחפש תמונות שנבדקו והוזנו מראש עם תיוגים רלוונטיים.

**7 עריכת תמונה**

Route:  
POST /api/edit-image

פרמטרים:  
{ "fileId": string, "requestText": string } body:

פלט:  
{ "fileId": string[], "url": string[], "message": "Image edited successfully" } data:

קודי סטטוס:

* Ok 200: אם העריכה בוצעה בהצלחה.
* Bad Request 400: אם פרמטרים לא תקינים (למשל, טקסט בקשה לא ברור או תמונה לא נמצאה).
* Internal Server Error 500: אם הייתה שגיאה בצד ה-API של הבינה המלאכותית.

לוגיקה עסקית:

* הפונקציה תקבל טקסט חופשי שכולל את בקשת המשתמש לעריכת התמונה (כמו הוספת טקסט או שיפור האיכות).
* המערכת תשלח את בקשת המשתמש יחד עם מזהה התמונה ל-API של בינה מלאכותית DeepAI שיבצע את העריכה.
* לאחר ביצוע העריכה, התמונה המעובדת תוחזר למערכת עם לינק להורדה או לשיתוף.

**8 יצירת קולאז' מתמונות**

Route:  
POST /api/create-collage

פרמטרים:  
body: { "imageIds": string[], "layout": string (optional, values: ["grid", "stacked"]), "size": { "width": number, "height": number } }

פלט:  
{ "collageId": string, "url": string, "message": "Collage created successfully" } data:

קודי סטטוס:

* Ok 200: אם הקולאז' נוצר בהצלחה.
* Bad Request 400: אם אחד או יותר מזהים של התמונות לא תקינים.
* Unprocessable Entity 422: אם המידות שנבחרו לא תואמות את פרופורציות התמונה.

לוגיקה עסקית:

* יצירת קולאז' מתוך תמונות שהוזנו, בהתבסס על פרמטרי layout ו-size שנבחרו.
* ניתן לבחור בין מבני סידור שונים כמו "grid" או "stacked".
* תהליך ייצור הקולאז' יעשה על גבי שרת API של בינה מלאכותית

**9 הורדת תמונה**

Route:  
GET /api/download-image

פרמטרים:  
{ "fileId": string } query params:

פלט:

* קובץ התמונה(File)
* { "message": "Image shared successfully" } data:

קוד סטטוס:

* Ok 200: אם ההורדה הצליחה

לוגיקה עסקית:

* הצגת התמונה שנשמרה ב Google Cloud Storage להורדה לפי מזהה קובץ.

פונקציה: שיתוף תמונה

Route:  
POST /api/share-image

פרמטרים:  
body: { "fileId": string, "platform": string (values: ["email", "facebook", "twitter"]), "recipient": string (optional, for email only) }

פלט:  
{ "message": "Image shared successfully" } data:

קודי סטטוס:

* Ok 200: אם השיתוף בוצע בהצלחה.
* Bad Request 400: אם פרמטרים לא תקינים (למשל, פלטפורמה לא נתמכת או fileId לא תקין)
* Internal Server Error 500: אם לא ניתן לשלוח את התמונה לפלטפורמה שבחרה המשתמש

לוגיקה עסקית:

* אם נבחרה פלטפורמת שיתוף כמו דוא"ל, תשלח הודעת דוא"ל עם לינק לתמונה.

המשתמש יוכל לבחור האם לשתף דרך פלטפורמת דוא"ל או רשתות חברתיות ולקבוע אם נדרש לשלוח את התמונה למישהו מסוים (למשל, בכתובת דוא"ל).

**3.2 אימות והרשאות**

1.אפלקצייה מבצעת אימות באמצעות JWT

2. התפקידים הקימיים במערכת הוא המורה-המשתמש הפרטי יש לו גישה לאזור האישי שלו ואפשרות שיתוף עם אחרים לפי הרשאת גישה שמחליט

**3.2 אפליקצית נהול**

**דוחות**: הפקת דוחות פעילות משתמשים, סטטיסטיקות שימוש

**ניהול משתמשים**: CRUD למשתמשים + שיוך תפקידים

**הרשאות**: עדכון הרשאות למשתמשים

**הגדרות מערכת**: ניהול פמרטרים גלובליים (למשל: מכסות אחסון)

3 .תרשים מבנה הטבלאות:

**טבלת user**

|  |  |  |  |
| --- | --- | --- | --- |
| תיאור | מאפיינים | סוג נתונים | שם עמודה |
| מזהה יחודי למשתמש | PK, AUTO INCREMENT | INT | id |
| שם פרטי | NOT NULL | VARCHAR(50) | first\_name |
| שם משפחה | NOT NULL | VARCHAR(50) | last\_name |
| אימייל | UNIQUE | VARCHAR(100) | email |
| סיסמה | NOT NULL | VARCHAR(9) | password |
| תפקיד | NOT NULL | ENUM | "user"/"admin" |
| תאריך יצירת משתמש | DEFAULT NOW() | DATETIME | created\_at |
| טלפון |  | VARCHAR(10) | Phon\_number |

**טבלת Files**

|  |  |  |  |
| --- | --- | --- | --- |
| תיאור | מאפיינים | סוג נתונים | שם עמודה |
| מזהה ייחודי לתמונה | PK, AUTO INCREMENT | INT | id |
| מזהה המשתמש שהעלה את התמונה | FK | INT | user\_id |
| שם הקובץ | NOT NULL | VARCHAR(100) | file\_name |
| מזהה אלבום שאליו שייך הקובץ |  | INT | album\_id |
| נתיב הקובץ | NOT NULL | VARCHAR(255) | file\_path |
| גודל הקובץ |  | INT | file\_size |
| תאריך העלאת תמונה | DEFAULT NOW() | DATETIME | uploaded\_at |
| תאריך עדכון אחרון | DEFAULT NOW() | DATETIME | updated\_at |



**טבלת Albums**

|  |  |  |  |
| --- | --- | --- | --- |
| תיאור | מאפיינים | סוג נתונים | שם עמודה |
| מזהה ייחודי לאלבום | PK, AUTO INCREMENT | INT | id |
| מזהה המשתמש שיצר את האלבום | FK | INT | user\_id |
| שם האלבום | NOT NULL | VARCHAR(100) | album\_name |
| תאריך יצירת האלבום | DEFAULT NOW() | DATETIME | created\_at |

**טבלת Tags**

|  |  |  |  |
| --- | --- | --- | --- |
| תיאור | מאפיינים | סוג נתונים | שם עמודה |
| מזהה ייחודי לתמונה | PK, AUTO INCREMENT | INT | id |
| שם התיוג | NOT NULL | VARCHAR(100) | album\_name |

**טבלת AlbumsFiles:**

|  |  |  |  |
| --- | --- | --- | --- |
| תיאור | מאפיינים | סוג נתונים | שם עמודה |
| מזהה אלבום | FK | INT | album\_id |
| מזהה קובץ | FK | INT | file\_id |

|  |  |  |  |
| --- | --- | --- | --- |
| תיאור | מאפיינים | סוג נתונים | שם עמודה |
| מזהה התיוג | FK | INT | tag\_id |
| מזהה קובץ | FK | INT | file\_id |

**טבלת FilesTags:**

**טבלת Messages:**

|  |  |  |  |
| --- | --- | --- | --- |
| תיאור | מאפיינים | סוג נתונים | שם עמודה |
| מזהה ייחודי להודעה | PK, AUTO INCREMENT | INT | id |
| מזהה המנהל שהכניס את ההודעה | FK | INT | user\_id |
| תוכן ההודעה | NOT NULL | VARCHAR(255) | message |
| תאריך יצירת ההודעה | DEFAULT NOW() | DATETIME | created\_at |
| האם ההודעה פעילה | DEFAULT TRUE | BOOLEAN | is\_active |

קשרים בין הטבלאות:

* Users → Files קשר Many-to-Many
* Users → Albums קשר One-to-Many
* Files ↔ Tags קשר Many-to-Many
* Albums→ Files קשר Many-to-Many
* ~~Users → Messages קשר One-to-Many~~

**5 הוגדר למעלה!!!**

**תמונה**

**משתמש**

**אלבום**