**טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הגשת פרויקט**

**Tx2Class - B7 - 12**

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
| ראובן יצחקוב | 322855909 |
| פלג כנפי | 206680753 |
| גל מויאל | 211585641 |
| עדן פורמן | 325393817 |
| נופר בזק | 212148340 |
| אביעד סלמה | 207287988 |

קישור לתיקיית גיט ציבורי: <https://github.com/pelegc49/Web>

קישור לאתר: <https://web-w9fv.vercel.app/>

**תקציר הפרויקט:** המערכת היא אפליקציית ווב אינטראקטיבית המאפשרת למשתמשים ליצור, לערוך ולשמור דיאגרמות UML (מחלקות) באמצעות ממשק גרפי נוח. משתמשים יכולים להירשם, להתחבר, לנהל פרויקטים אישיים, לשמור היסטוריה של דיאגרמות, ולבצע פעולות כמו יצירת מחלקות, הוספת תכונות ומתודות, יצירת קשרים בין מחלקות ועוד. קיימת תמיכה במצב כהה/בהיר, עזרה מובנית, וממשק רספונסיבי המתאים גם למובייל.

**פונקציונליות מרכזיות לפי משתמשים:**

* **משתמש לא מחובר:** יכול לצפות בדף הבית, לקרוא עזרה, ולהתנסות ביצירת דיאגרמה (ללא שמירה).
* **משתמש מחובר:** יכול ליצור ולשמור פרויקטים, לצפות בהיסטוריית הפרויקטים, לערוך ולמחוק דיאגרמות, לשחזר דיאגרמות מהיסטוריה, ולשנות סיסמה.

**מימוש וטכנולוגיות:**

* **Frontend:**
  + React כולל React Router לניהול ניווט
  + Tailwind CSS ו- CSS מותאם לעיצוב רספונסיבי ונוח
  + ספריית React Flow להצגת דיאגרמות גרפיות
  + Axios לתקשורת עם ה-API
* **Backend:**
  + Node.js עם Express.js לניהול REST API
  + Firebase Auth לאימות משתמשים
  + CORS לאבטחת תקשורת בין דומיינים
* **DB:**
  + Firestore (Firebase) - מסד נתונים NoSQL בענן, לאחסון משתמשים ופרויקטים
* **Styling:**
  + Tailwind CSS, מחלקות עיצוב מותאמות, ו-CSS מודולרי לכל קומפוננטה

המערכת בנויה כך שכל משתמש מקבל חוויית עבודה אישית, מאובטחת, ונוחה – הן בדסקטופ והן במובייל.

1. עליכם להמשיך את בניית האתר לפי האלמנטים המתקדמים שלמדתם - בוצע
2. יש למנות מהנדס מערכת בכל צוות, אשר יהיה אחראי על הגדרת והקצאת המשימות בתרגיל זה.  
   נא לרשום את שם הסטודנט בתרגיל זה. על מהנדס המערכת לכתוב כיצד נעשתה חלוקת העבודה מול הצוות, מה היו המשימות של כל חבר צוות, האם היה ממשק בין חברי הצוות, והאם המשימות מולאו:

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| ראובן יצחקוב (מהנדס) | 4, תיעוד לקוד, רספונסיביות, 7, | , תיעוד לקוד, רספונסיביות, 7, |
| פלג כנפי | 5ב, תיעוד לקוד, 7, | 5ב, תיעוד לקוד, 7, |
| גל מויאל | 3, 5א, 5ב, תיעוד לקוד, | 3, 5א, 5ב, תיעוד לקוד, |
| עדן פורמן | 4, תיעוד לקוד, | 4, תיעוד לקוד, |
| נופר בזק | 2, תיעוד לקוד, 6, | 2, תיעוד לקוד, 6, |
| אביעד סלמה | 2, תיעוד לקוד, 6, | 2, תיעוד לקוד, 6, |

2. הציגו רשימת דרישות:

יש להתבסס על המבנה של asana (<https://asana.com/resources/software-requirement-document-template>

* דרישות פונקציונליות:

1. המשתמש חייב להירשם בעזרת שם משתמש, אימייל, סיסמה.

2. המשתמש יכול להתחבר לחשבון קיים בעזרת אימייל וסיסמה.

3. המשתמש יכול לשחזר סיסמה בעזרת אימייל.

4. המשתמש יכול לראות היסטוריית פרויקטים קודמים שלו.

5. למשתמש יכול ללחוץ על כפתור כדי להתחיל ליצור תרשים חדש.

6. המשתמש יכול להקליד טקסט (לפי סינטקס קבוע מראש) והמערכת תזהה מחלקות, תכונות, קשרים.

7. המערכת תמיר את הטקסט המתקבל לJSON פנימי.

8. המערכת תדע שמשפט של המשתמש מסתיים בעזרת זיהוי "." בסוף.

9. המערכת תציג דיאגרמת קלאס (בתנאי שהטקסט עומד בסינטקס שקבענו), ע"י react flow.

10. המשתמש יוכל להזיז ולשנות את הגודל של התרשים על כל חלקיו.

11. המשתמש יוכל לשמור את הפרויקט שלו.

12. המערכת תאפשר להוריד את התרשים כקובץ

13. לשונית ה "header", תהיה קבועה עבור כל האתר.

14. לשונית ה"header", תכיל כפתור על מנת לעבור ל"dark mode" או ל "light mode69", תלוי מה לחוץ.

15. לשונית ה"header", תכיל כפתור על מנת להתחבר או להירשם .

16. מסך הבית, יכיל כפתור התחברות.

17. מסך הבית, יכיל כפתור הרשמה.

18. המשתמש יוכל להתנתק מתוך לשונית ה "header".

19. מסך העזרה, יכיל שלושה כפתורים למעבר בין חלונות העזרה.

* דרישות לא פונקציונליות (בנפרד, יש לסווג דרישות לא פונקציונליות לפי wikipedia NFR).

שימושיות-usability :

1. האתר יהיה אינטואיטיבי, עם עיצוב מודרני.

2. ייעשה שימוש ב־ToolTips להסבר הפקודות (ביצירת התרשים).

תפקוד- Performance:

1. התרשים הנוצר יתעדכן בזמן אמת.

אבטחה- Performance :

1.יהיה שימוש ב- captcha בזמן ההרשמה.

2.אימות חוזר של סיסמה בעת ההרשמה.

זמינות- Availability :

1. המערכת תהיה זמינה לשימוש בכל עת, למעט בפרקי זמן קצרים שיידרשו לצורך ביצוע תחזוקה ועדכונים מתוכננים מראש

תמיכה- Compatibility :

1. האתר יתמוך בדפדפן אינטרנט.

2. האתר יהיה רספוניבי ויתמוך גם במסכים קטנים (טלפונים חכמים).

תחזוקה- Maintainability:

1. הקוד מופרד לקומפוננטות.

2. תיעוד לקוד יישמר לשם תחזוקה עתידית.

* דרישות ממשק חיצוניות.

ממשק משתמש-UI:

1. header קבוע.

2. כפתורי ניווט.

3. טפסים לרישום והתחברות.

4. אזור הקלדה (אפליקציה, התחברות, הרשמה).

5. canvas גרפי לבניית התרשים.

6. התממשקות עם עכבר ומקלדת.

ממשק קלט-input :

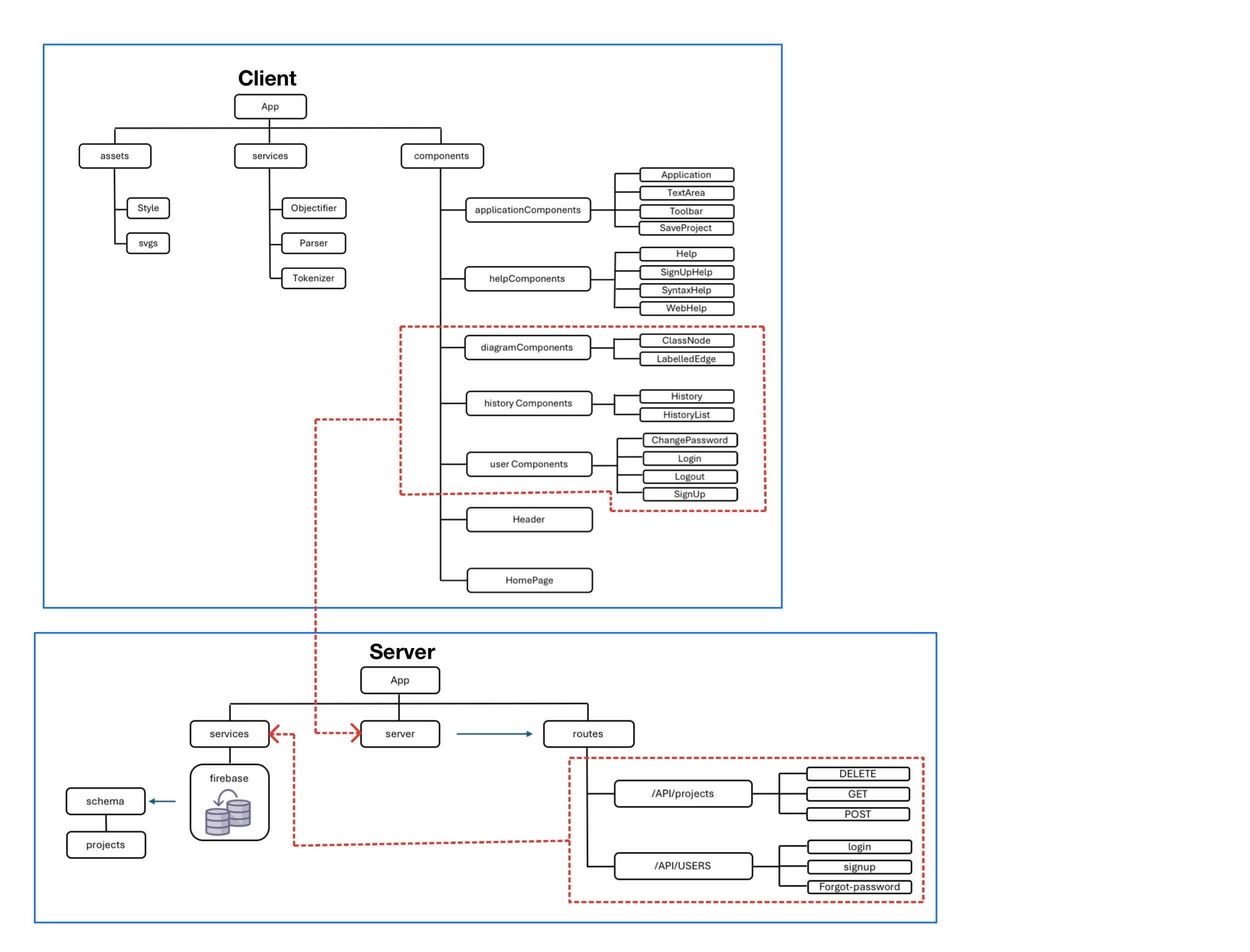
1. טקסט (אפליקציה, התחברות, הרשמה) באזור ההקלדה.
2. עכבר (לחיצה על כפתורים).

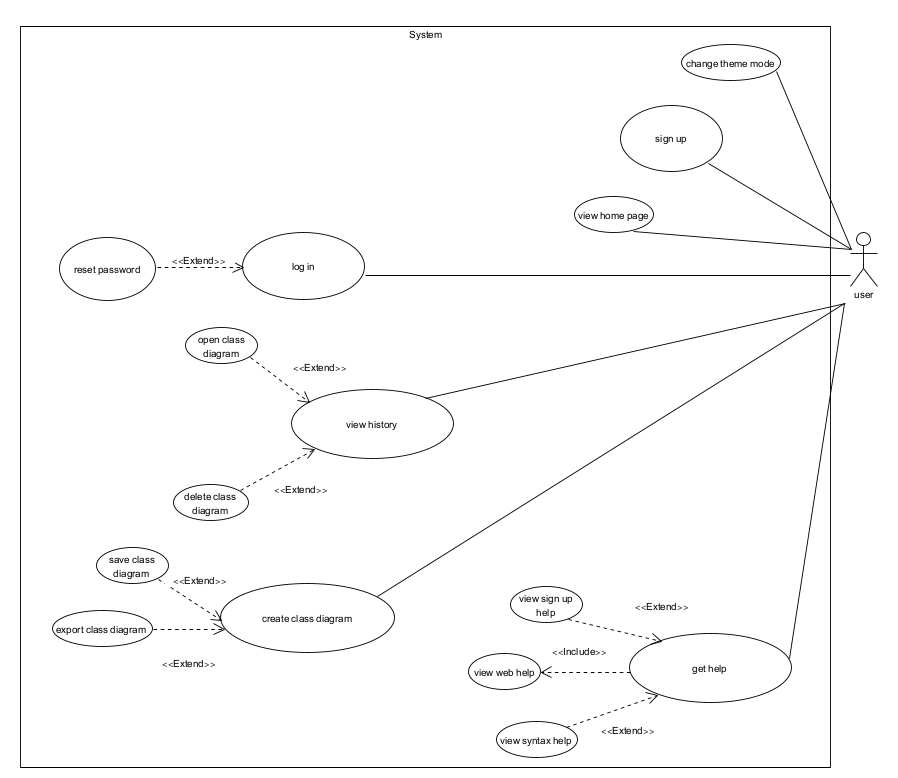
ממשק פלט-output:

1. הצגת תרשים גרפי (class diagram) מבוסס JSON.
2. אפשרות הורדה לקובץ (SVG/PNG).
3. תצוגת הודעות שגיאה בטופס (לדוגמה: "סיסמה שגויה").

ממשק שירותים צד שלישי:

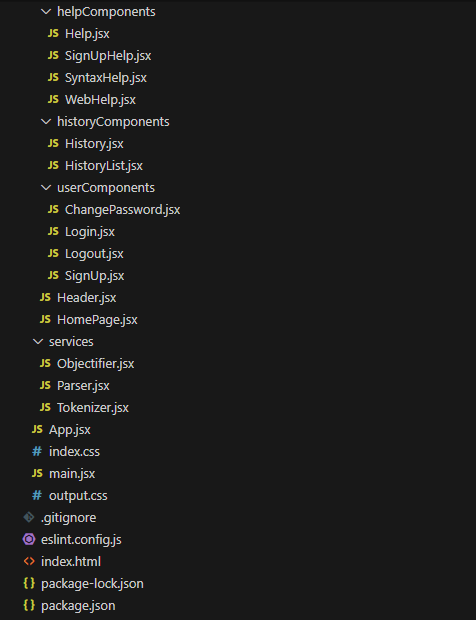
1. שימוש בfirebase .
2. שימוש בdom-to-image לצילום תגית HTML

3. הציגו ארכיטקטורה מעודכנת של האתר (תרשים הכולל את האלמנטים המרכזיים).

4. הציגו דיאגרמת use case המתארת את השימוש באתר.

5. יש להציג מבנה סופי של האתר שלכם:   
א. האתר ימומש ב -React/Preact/Next, וכן שימוש ב Tailwind - נא להציג דיאגרמה המתארת את התיקיות והקבצים השונים. יש לפרט את הקומפוננטות השונות.

A screenshot of a computer program

AI-generated content may be incorrect.  
  
A black background with white text

AI-generated content may be incorrect.

פירוט הקומפוננטות הוא להלן:

**Frontend Components:**

1.Application Management : Application.jsx (Front/src/components/applicationComponents/)

* קומפוננטה ראשית לעורך דיאגרמות UML
* מטפלת בהזנת טקסט, בהצגת הדיאגרמה ובשמירת פרויקטים
* מנהלת את האינטראקציה בין קלט הטקסט לדיאגרמה הוויזואלית

2. Diagram Components :

* ClassNode.jsx (Front/src/components/diagramComponents/)
* מציגה צמתים של מחלקות בדיאגרמת ה-UML
* מציגה שם מחלקה, תכונות ומתודות בעיצוב מתאים
* LabelledEdge.jsx (Front/src/components/diagramComponents/)
* מטפלת בהצגה הגרפית של קשרים בין מחלקות
* מציירת קווים עם תוויות ריבוי (one/many)

3.Help System Components :

* Help.jsx (Front/src/components/helpComponents/)
* קומפוננטת הורה לכל תוכן העזרה
* מאפשרת ניווט בין חלקי העזרה השונים
* WebHelp.jsx (Front/src/components/helpComponents/)
* מדריך התחלה מהירה למשתמשים חדשים
* מסביר את השימוש הבסיסי באפליקציה
* SyntaxHelp.jsx (Front/src/components/helpComponents/)
* תיעוד מקיף של תחביר דיאגרמת ה-UML
* כולל דוגמאות לכל הפקודות הנתמכות
* SignUpHelp.jsx (Front/src/components/helpComponents/)
* מדריך לתהליך יצירת חשבון
* מסביר אימות ואבטחת משתמש

4. User Management Components :

* Login.jsx (Front/src/components/userComponents/)
* מטפל באימות משתמשים
* כולל אפשרות לאיפוס סיסמה
* SignUp.jsx (Front/src/components/userComponents/)
* רישום משתמש חדש עם אימות אימייל
* כולל אימות CAPTCHA
* ChangePassword.jsx (Front/src/components/userComponents/)
* מאפשר למשתמשים לשנות סיסמה
* כולל ולידציה ובדיקות אבטחה
* Logout.jsx (Front/src/components/userComponents/)
* מטפל בתהליך יציאה מהמערכת
* כולל קישור לשינוי סיסמה
* UserProfile.jsx (Front/src/components/userComponents/)
* תצוגה בסיסית של פרופיל המשתמש

5. History and Project Management :

* History.jsx (Front/src/components/historyComponents/)
* מציג רשימת פרויקטים שנשמרו
* מספק ממשק לניהול פרויקטים
* HistoryList.jsx (Front/src/components/historyComponents/)
* מטפל באימות משתמשים
* כולל אפשרות לאיפוס סיסמה

6. Application Tools :

* TextArea.jsx (Front/src/components/applicationComponents/)
* רכיב להזנת טקסט עבור הגדרת UML
* מטפל בשינויים ומעדכן בהתאם
* Toolbar.jsx (Front/src/components/applicationComponents/)
* סרגל כלים אנכי עם כלי דיאגרמות UML
* כולל רמזים עם מידע תחבירי
* SaveProject.jsx (Front/src/components/applicationComponents/)
* חלון קופץ לשמירת פרויקטים
* מטפל בהזנת שם ובשמירה

7. Core UI Components :

* Header.jsx (Front/src/components/)
  + סרגל ניווט ראשי
  + כולל לוגו, מתג מצב כהה, ובקרות משתמש
* HomePage.jsx (Front/src/components/)
  + דף נחיתה עם הצגת האפליקציה
  + כולל קריאה לפעולה למשתמשים חדשים

7. Core Services :

* Tokenizer.jsx (Front/src/services/)
* ניתוח לקסיקלי של טקסט הקלט
* יצירת טוקנים עבור ה-parser
* טיפול בהערות ורווחים
* דיווח על שגיאות כולל מספרי שורות
* Parser.jsx (Front/src/services/)
* מימוש כללי הדקדוק (Grammar)
* יצירת עץ תחביר מופשט (AST)
* אימות תחבירי
* הודעות שגיאה מפורטות
* Objectifier.jsx (Front/src/services/)
* המרה מ-AST לדיאגרמה
* יצירת צמתים וקשרים (nodes and edges)
* טיפול בקשרים בין מחלקות
* ניהול סידור ויזואלי של הדיאגרמה

**Backend Services:**

1. server.js (Back/) - Main server application

2. API Routes :

* api.js (Back/routes/) - נקודות קצה כלליות
* projects.js (Back/routes/) - נקודות קצה לניהול פרויקטים
* users.js (Back/routes/) - נקודות קצה לניהול משתמשים

3. Services - firebase.js (Back/services/) - אינטגרציה עם Firebase לאחסון נתונים

ב. יש לפרט את פריטי המידע - יש להשתמש במידע אמיתי ורלוונטי לפרויקט שלכם (בשליפה ממסד נתונים חיצוני או מ - API).

את הDB אנו שומרים בשירות Firestore של Firebase, שהוא DB מבוסס מסמכים.

אנחנו שומרים פרויקטים של משתמשים בDB הזה.

פרויקט בנוי משם הפרויקט, רכיבי הדיאגרמה (צמתים, קשתות, מיקומים שמורים של צמתים), הטקסט שהוכנס כקלט לתוכנה, תמונה של הפרויקט בפורמט base64.

לכל משתמש, יש מסמך שהשם שלו הוא מזהה המשתמש (UID) שבו שמורים כל הפרויקטים שלו, המפתחות במסמך הם שמות הפרויקטים, והערכים הם הפרויקטים עצמם במבנה שמתואר למעלה.

כל המסמכים האלו נמצאים תחת קולקציה (Collection) projects.

את המשתמשים עצמם אנו שומרים בשירות Authentication של firebase, שירות זה מאפשר לנהל משתמשים באופן מאובטח.

API:

POST /api/projects

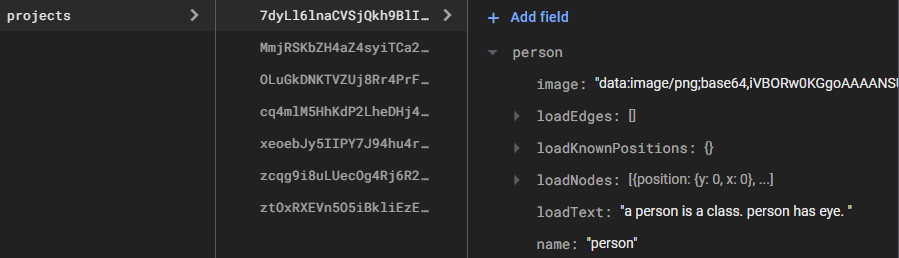
מתודה ליצירת פרויקט. מצפה להעברת הפרויקט ואת המשתמש בגוף (body) הבקשה.

GET /api/projects/:uid

מתודה לשליפת כל הפרויקטים של המשתמש בעל המזהה uid.

DELETE /api/projects/:uid/:projName

מתודה למחיקת פרויקט ספציפי



6. יש להגיש תיק למתכנת:

א. יש לתעד ולהסביר את כל הפונקציות המרכזיות בקוד שלכם.

ב. יש לפרט:

* את כל ה -API שהשתמשתם בהם,
* קישור ל- DB,
* סביבות מיוחדות שהתקנתם,
* קטעי קוד מיוחדים שלקחתם ממקורות שונים (כולל רפרנסים) ,
* פרומפטים שהשתמשתם בהם בכלי AI
* שמות משתמש וסיסמאות שהגדרתם לכל סוג משתמש, לצורך חיבור לאתר.

**.עבור צד שרת – back :**

1. נתבים- Routers
2. (projects.js)Delete :

קלט : uid (מזהה משתמש), projName (שם הפרויקט) .

פלט: מחזיר את רשימת הפרויקטים המעודכנת או הודעת שגיאה.

פעולה: שולף את מסמך הפרויקטים של המשתמש מה-firebase מוחק את המסמך הרצוי ומעדכן את המסמך .

1. Get (projects.js):

קלט : uid (מזהה משתמש).

פלט: מחזיר JSON עם כל הפרויקטים של המשתמש.

פעולה: שולף את כל הפרויקטים של המשתמש מה-firebase .

1. (projects.js) post :

קלט : פרויקט לעדכן.

פלט: הודעת הצלחה או שגיאה.

פעולה: בודק אם קיים מסמך פרויקטים למשתמש, אם כן מוסיף את הפרויקט החדש ואם לא יוצר מסמך חדש .

1. post (users.js):

קלט : אימייל, סיסמה.

פלט: פרטי משתמש או שגיאה.

פעולה: מאמת פרטי התחברות באמצעות Firebase Auth.

1. post (users.js):

קלט : אימייל, סיסמה.

פלט: פרטי משתמש או שגיאה.

פעולה: מאמת פרטי התחברות באמצעות Firebase Auth.

1. Post /login (users.js):

קלט : אימייל, סיסמה.

פלט: פרטי משתמש או שגיאה.

פעולה: מאמת פרטי התחברות באמצעות Firebase Auth.

1. Post /signup (users.js):

קלט : אימייל, סיסמה.

פלט: פרטי משתמש או שגיאה.

פעולה: יוצר משתמש חדש באמצעות Firebase Auth.

1. Post /forgot-password (users.js):

קלט : אימייל.

פלט: הודעת הצלחה או הודעת שגיאה.

פעולה: שולח מייל לאיפוס סיסמה באמצעות Firebase Auth.

1. Post /change-password (users.js):

קלט : אימייל, סיסמה ישנה, סיסמה ישנה.

פלט: הודעת הצלחה או הודעת שגיאה.

פעולה: מאמת מחדש את המשתמש ומעדכן את הסיסמה באמצעות Firebase Auth.

1. **עבור צד לקוח – front :**
2. Application() (Application.jsx) :

קלט : אין.

פלט: אין.

פעולה: מנהל את כל הלוגיקה של האפליקציה כולל עיבוד טקסט, יצירת דיאגרמות, ושמירת פרויקטים.

1. useEffect(Application.jsx):

קלט : text (הטקסט שהוזן).

פלט: מעדכן את התרשים.

פעולה: המנוע המרכזי של האפליקציה - הופך טקסט לדיאגרמה.

1. SaveProject (SaveProject.jsx):

קלט : שם הפרויקט .

פלט: הודעת הצלחה או שגיאה.

פעולה: יוצר אובייקט פרויקט, יוצר תמונה של הדיאגרמה ושולח לשרת.

**הסבר קישור למסד נתונים:**

הפרויקט משתמש בFirebase לאחסון משתמשים והיסטוריית דיאגרמות.

הקישור עצמו מתבצע דרך הקובץ (firebaseConfig.js) שהוא קובץ קונפיגורציה עם פרטי הפרויקט.

**סביבות מיוחדות שהתקנו :**

* Node.js (גרסה 18+)
* React (Vite)
* TailwindCSS
* Firebase (npm install firebase)
* Express
* axios
* dotenv
* react-router-dom
* cors
* body-parser

**קטעי קוד ממקורות חיצוניים :**

* קוד הקפצ'ה (captcha) – מבוסס על דוגמאות מStackOverflow- , אשר שופרו והותאמו לצרכינו .

רפרנס : <https://stackoverflow.com/questions/25861262/rotate-html5-canvas-text-from-its-center-point>

* קריאות Firebase Auth:

לפי הוראות השימוש של Firebase .

**רפרנס:**  
האתר הרשמי של ReactFlow אשר מכיל דוגמאות<https://reactflow.dev/>  
הגיטהב של dom-to-image <https://github.com/tsayen/dom-to-image>

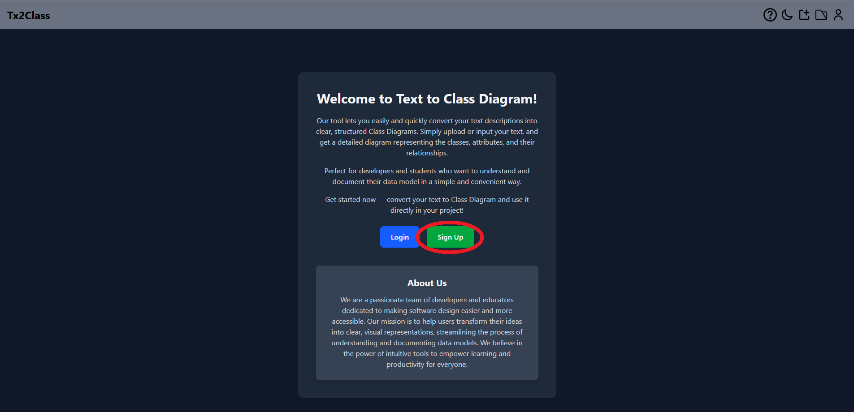
**פרומפטים ל-AI:**איך משתמשים בAPI של firebase? תן דוגמאות בבקשה  
איך מייצרים אפקט hover בעיצוב של tailwind?

**שמות משתמש וסיסמאות שהגדרנו: (ניתן גם ליצור משתמש)**

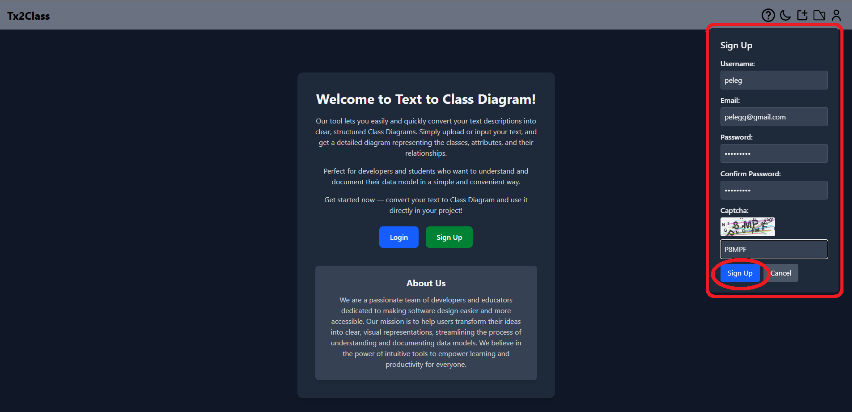
[a@a.com](mailto:a@a.com) : aaaaaa  
[asd@gmail.com](mailto:asd@gmail.com) : asdasd

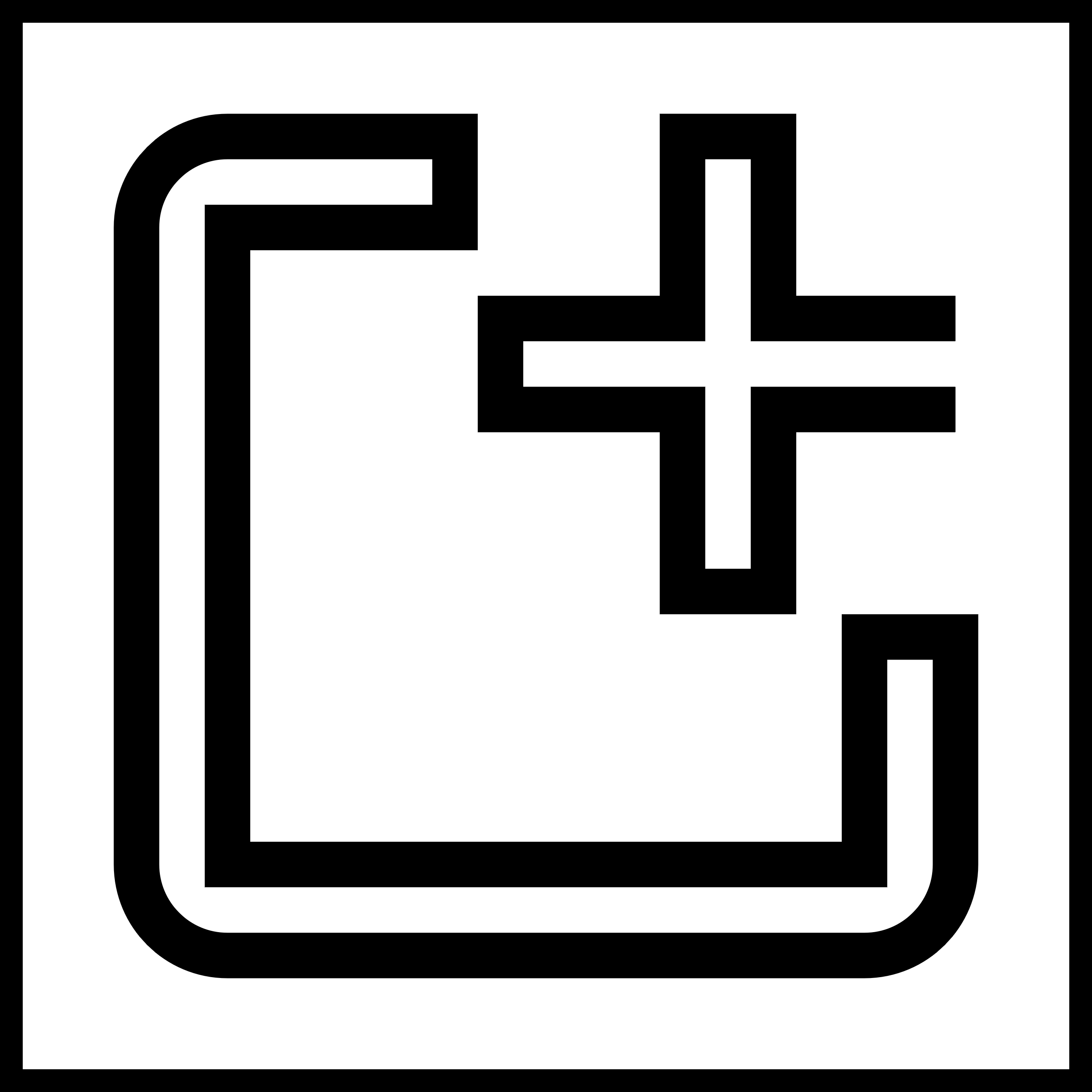
7. יש להגיש תיק משתמש הכולל צילומי מסך והסברים כיצד להשתמש באתר שלכם. נא לכלול סרטון של כדקה-דקה וחצי , שיווקי, המציג את האתר.

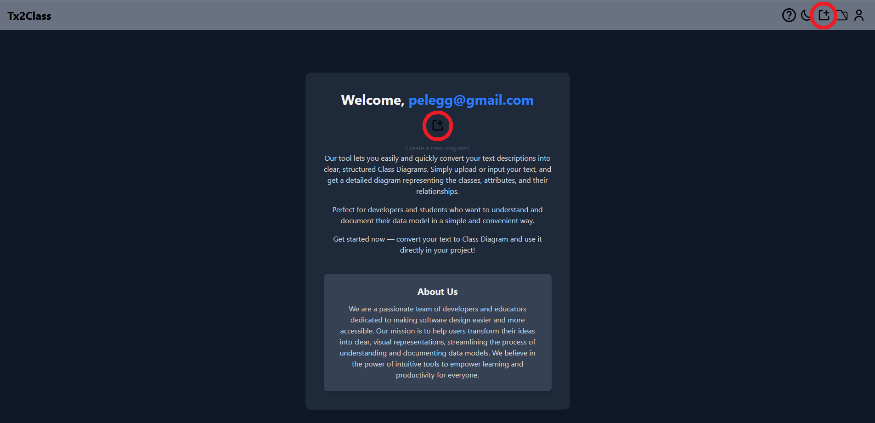
בכניסה לאתר, המשתמש ינותב לדף הבית. יש להרשם תחילה לאתר על ידי לחיצה על כפתור Sign Up:



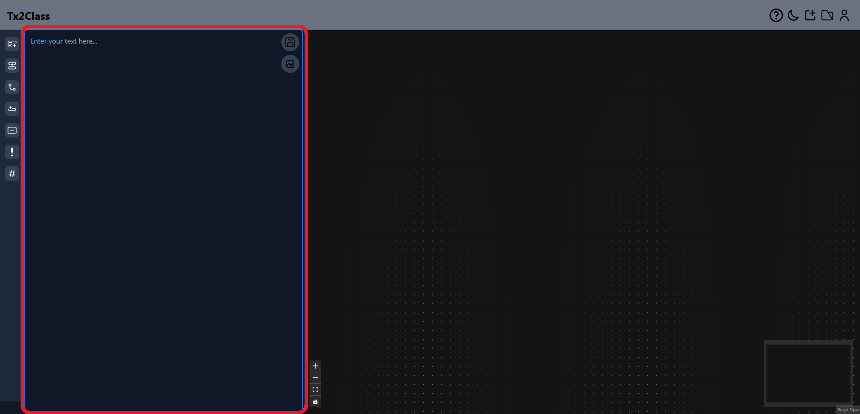
ולאחר מכן מילוי הפרטים ולחיצה על Sign Up:



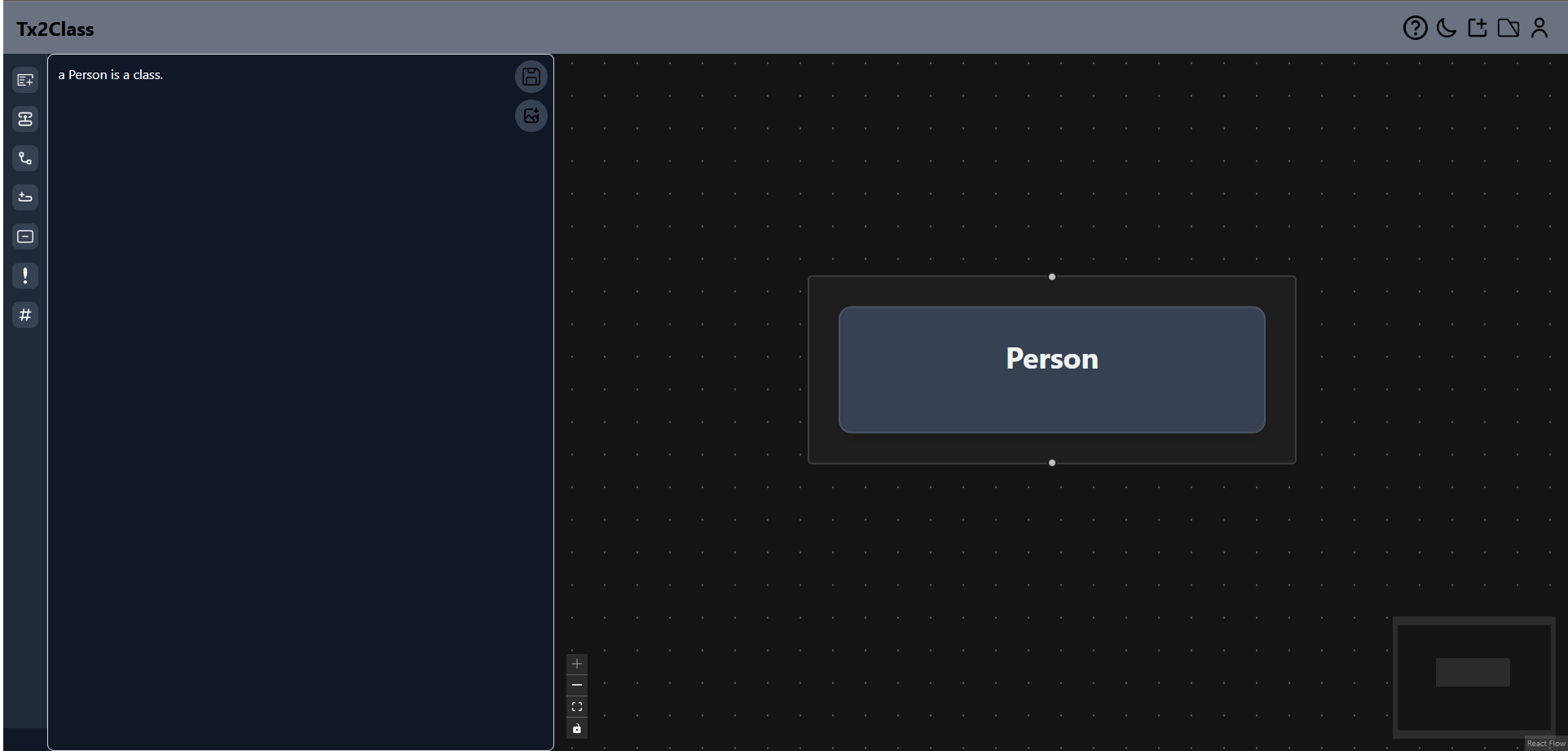
כעת, יש ללחוץ על כפתור  הממוקם במרכז המסך או בheader:

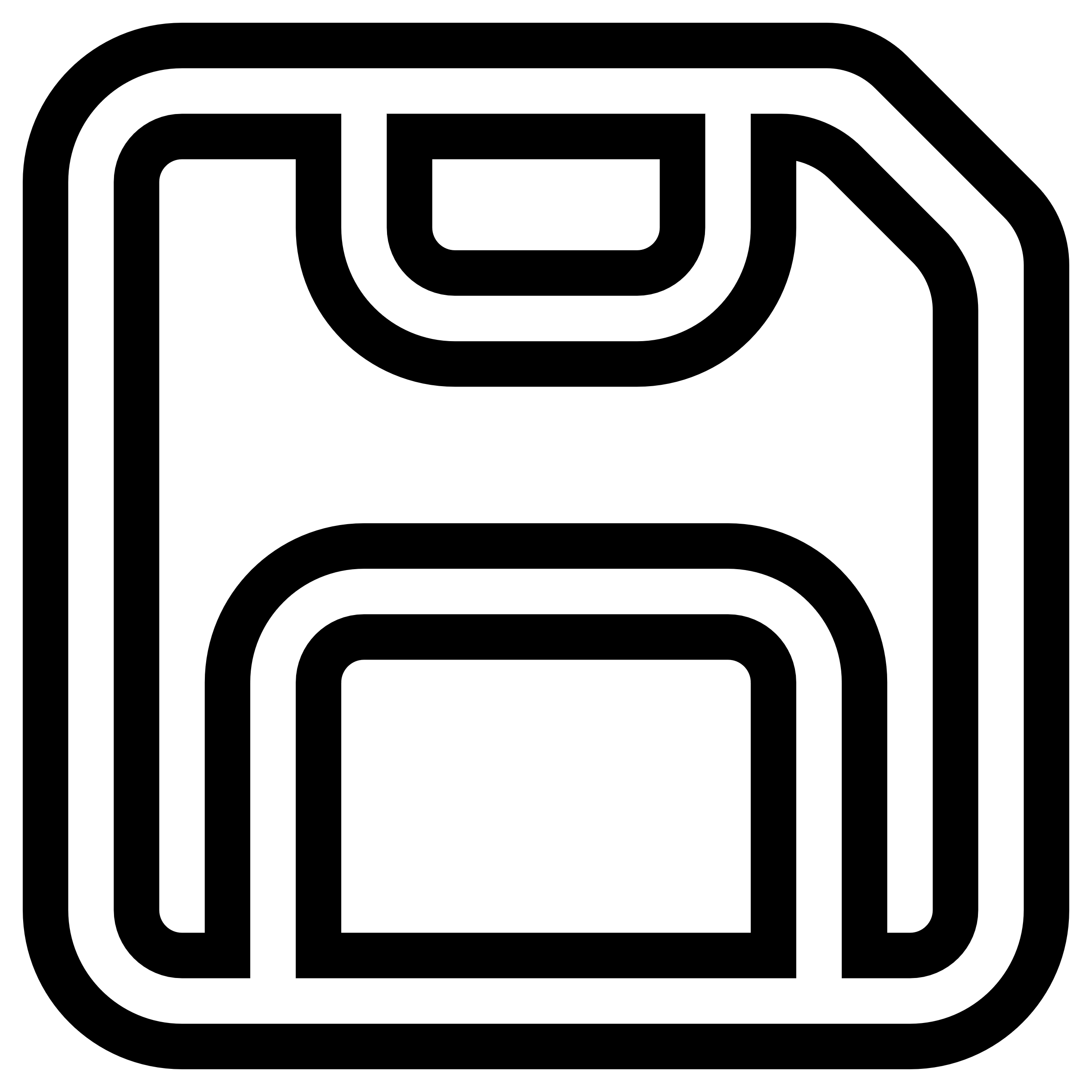


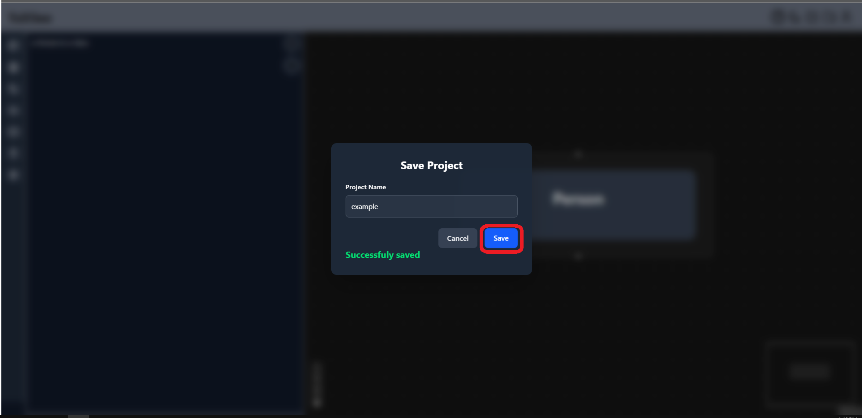
עכשיו ניתן לרשום פקודות ליצירת מחלקות בtextArea משמאל:



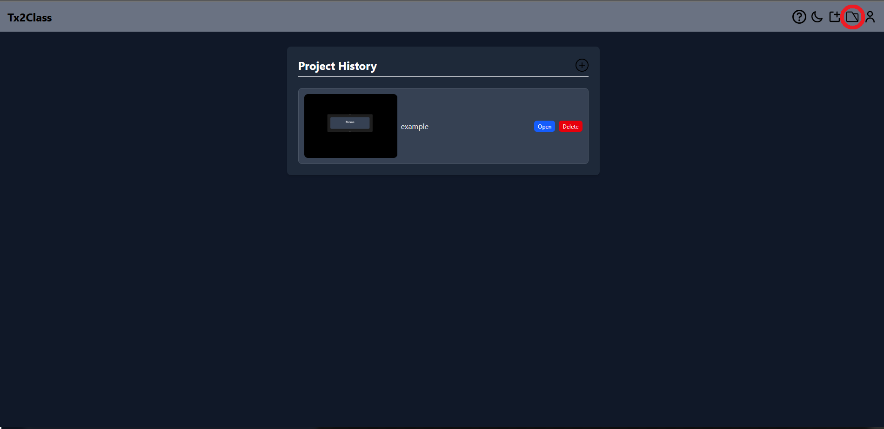
לדוגמה, לאחר כתיבת "a Person is a class.", נוצרת קובייה עם Person בצד ימין. ניתן לגרור אותה.



ניתן לשמור את הדיאגרמה כתמונה, כדי לעשות זאת יש ללחוץ על כפתור  שנמצא בתוך הtextArea.  
ניתן לשמור את הדיאגרמה כפרויקט, כדי לעשות זאת יש ללחוץ על כפתור  שנמצא בתוך הtextArea ולאחר מכן לתת שם לפרויקט בחלונית pop-up וללחוץ על save.



על מנת למצוא את הפרויקט הזה וגם את שאר הפרויקטים של המשתמש, יש ללחוץ על כפתור  הנמצא בheader.



ניתן כעת לפתוח כל פרויקט על ידי לחיצה על Open.

או למחוק אותו על ידי לחיצה על Delete.