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

הגשת פרויקט

B13

Group 13

StudyBuddy

|  |  |
| --- | --- |
| שם חבר.ת הצוות | תז |
| אחמד טויל | 323065847 |
| סירין פאהום | 212810071 |
| אדם תחאוק | 214311912 |
| אבראהים אסעד | 324835099 |
| מרוה עואד | 325223451 |
| עומר ערמוש | 322966326 |

### **תקציר הפרויקט**

**StudyBuddy**

היא פלטפורמת עזרה לימודית שיתופית מסוג Peer-to-Peer, שמחברת בין סטודנטים שזקוקים לעזרה אקדמית לבין עמיתים בעלי ידע ונכונות לעזור.  
 המערכת מאפשרת פרסום שאלות עם קבצים, סינון לפי נושא, קבלת עזרה בכתב או בשיחה, ושימוש בבינה מלאכותית לניסוח שאלה ברור, סיכום תשובות והצעת מקורות חיצוניים.  
 כל שאלה יכולה להיפתח לצ׳אט בזמן אמת בין השואל לעוזר, תוך שמירה על חוויית שימוש נוחה, מהירה ומעוצבת היטב.

#### משתמשים עיקריים:

* מבקש עזרה – שואל שאלות, מקבל סיוע דרך תשובות כתובות או צ׳אט, מדרג עוזרים, משתמש ב-AI לשיפור ניסוח ומעקב אישי.
* עוזר (Peer Helper) – עונה על שאלות לפי תחום התמחות, מקבל דירוגים ותגים, משתתף בצ׳אט עם שואלים.

### **מימוש – טכנולוגיות עיקריות**

|  |  |
| --- | --- |
| **חלק** | **טכנולוגיות** |
| **Frontend** | React (Next.js App Router), Tailwind CSS, Framer Motion |
| **Styling** | Tailwind CSS, globals.css, עיצוב מודרני מגיב (Responsive) |
| **Database** | Firebase Firestore (NoSQL, real-time collections: requests, answers, users, chats) |
| **Authentication** | Firebase Auth – תמיכה ב־Email/Password, Google, GitHub |
| **AI Integration** | OpenAI GPT-3.5 (/api/rephrase, /api/ai-answer, /api/ai-suggestions), |
| **Search** | Algolia + Filters לפי תגיות, נושאים, קושי, unanswered |
| **Chat** | **Gun.js (GunDB)** – הודעות נשמרות ומסתנכרנות ב־P2P דרך: gun.get('chat-room-${chatId}') תוך שימוש ב־relay ציבורי https://gun-manhattan.herokuapp.com/gun |
| **Backend** | Next.js API Routes (שימוש ב-Fetch + Env Keys), תשתית Node-Ready להרחבות עתידיות |
| **Deployment** | Vercel (Frontend + API routes), Firebase Hosting |
| **State Management** | React Context (Auth, Theme, User, Notifications, Requests) |

**GitHub:**

<https://github.com/AhmadTawil1/study-buddy>

**Vercel:**

[Study-buddy-ebon-one.vercel.app](https://study-buddy-ebon-one.vercel.app/)

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| אחמד טויל | תיק הפיתוח (מבנה פרויקט, תיעוד פונקציות עיקריות, שילוב קוד ורפרנסים, APIs) | הושלם |
| איברהים אסעד | תיק המשתמש (הסבר שימוש באתר, צילומי מסך, סרטון הדגמה) | הושלם |
| עומר ערמוש | תרחיש שימוש מלא של משתמש באתר כולל מעבר בין דפים | הושלם |
| אדם תחאוק | מבנה סופי של האתר: ארכיטקטורת קבצים + בסיס נתונים (ERD) | הושלם |
| סירין פאהום | דרישות מערכת פונקציונליות ולא פונקציונליות + ארכיטקטורה כללית | הושלם |
| מרוה עואד | דרישות מערכת פונקציונליות ולא פונקציונליות + ארכיטקטורה כללית | הושלם |

System engineer: Ahmad Tawil

במהלך העבודה חולקו המשימות בצורה שווה וברורה בין חברי הצוות לפי תחומי אחריות. כל חבר קיבל חלק עצמאי אך משלים, תוך שמירה על תקשורת בין־חברית ושיתוף פעולה שוטף. כלל המרכיבים נבדקו ועברו תיעוד וסקירה בהתאם להנחיות התרגיל.

### **3. רשימת דרישות**

#### **● דרישות פונקציונליות:**

1.1.1 The system allows users to register using email and password

1.1.2 The system validates passwords (min 8 chars, upper/lowercase, digit, special character)

1.1.3 The system checks nickname uniqueness before account creation

1.1.4 The system supports Google and GitHub OAuth for social registration

1.1.5 The system automatically creates a user profile in Firestore

1.2.1 The system allows users to log in using email/password

1.2.2 The system allows login using Google and GitHub

1.2.3 The system allows password reset via email

1.2.4 The system maintains user session across pages

1.2.5 The system redirects the user to their intended page or profile after login

1.3.1 The system protects private routes by requiring authentication

1.3.2 The system supports secure logout and session expiration

2.1.1 The system allows users to edit name, and nickname

2.1.2 The system allows users to upload and change a profile picture

2.1.3 The system allows users to add a custom bio/about section

2.1.4 The system allows users to link GitHub and LinkedIn profiles

2.2.1 The system tracks number of questions asked and answers given

2.2.2 The system calculates reputation score based on upvotes

2.2.3 The system awards badges based on contribution milestones

2.2.4 The system ranks users in a community leaderboard

2.2.5 The system displays a recent activity timeline

2.3.1 The system allows users to view their full profile

2.3.2 The system displays public versions of other users’ profiles

3.1.1 The system supports markdown input for rich text formatting

3.1.2 The system allows file attachments (images, PDFs, DOCs)

3.1.3 The system allows code snippets with syntax highlighting

3.1.4 The system requires subject/category selection (Math, CS...)

3.1.5 The system supports custom tags

3.1.6 The system allows private or anonymous posting

3.1.7 The system rephrases unclear questions using AI

3.1.8 The system calculates a clarity score and displays improvement tips

3.2.1 The system displays a dynamic feed of all questions

3.2.2 The system allows filtering by subject, status, date, and answer count

3.2.3 The system supports full-text search

3.2.4 The system allows sorting (newest, unanswered, most answered)

3.2.5 The system supports infinite scrolling for large datasets

3.2.6 The system tracks the status of each question (open, in-progress, solved)

3.3.1 The system allows community users to submit answers

3.3.2 The system supports AI-generated answers when no human response exists

3.3.3 The system allows upvoting/downvoting answers

3.3.4 The system allows the original asker to mark an accepted answer

3.3.5 The system supports nested replies to answers

3.3.6 The system allows editing and deletion of user’s own answers

3.3.7 The system sends real-time notifications when new answers are posted

4.1.1 The system displays a real-time feed of study requests

4.1.2 The system allows saving (bookmarking) requests

4.1.3 The system allows sharing links to specific requests

5.1.1 The system allows direct messaging between users

5.1.2 The system supports dedicated chat rooms per help request

5.1.3 The system syncs chat messages in real-time

5.1.4 The system saves and retrieves message history

5.2.1 The system allows users to send Zoom meeting invites

5.2.2 The system sends real-time chat notifications

#### **● דרישות לא פונקציונליות (לפי סיווג Wikipedia NFR):**

### **1. Suitability**

#### The system uses Firebase and Next.js to ensure appropriate technology choices for scalability and real-time performance.

#### Chosen tools (Tailwind, Firestore, OpenAI, Algolia) are suitable for peer-to-peer academic platforms.

### **2. Correctness**

#### The system ensures correct user flow: only authenticated users can post, reply, or chat.

#### 100% of post, reply, and chat actions require authentication

#### Questions, answers, and user data are validated on both client and server sides.

#### Data written to Firestore is consistently validated via Firestore rules and UI forms.

### **3. Appropriateness**

#### The tone and design of the platform are optimized for students and academic environments.

#### AI-generated content is labeled clearly to distinguish it from human responses.

### **4. Usability**

#### Clean and intuitive UI with consistent design using Tailwind CSS.

#### Responsive layout across desktop, tablet, and mobile.

#### Clear navigation bar, form validations, error handling, and feedback messages.

#### Features like dark mode, profile editing, question tags, and auto-suggestions improve usability.

### **5. Performance**

#### Fast Search with Algolia

#### Average query response: < 20ms

#### New questions indexed within ~2 seconds

#### Optimized Feeds

#### Lazy loading and infinite scroll reduce rendering load

#### Only 10 cards rendered per scroll → 30–40% faster UI response

#### AI & Firebase Speed

#### AI responses and Firestore actions complete in < 500ms under normal usage

### **6. Responsiveness**

#### Layout adapts fluidly to all screen sizes.

#### Optimized touch areas and form controls for mobile usage.

#### Media queries and Tailwind breakpoints used for adaptive styling.

### **7. Reliability**

#### Firebase Auth auto-persist: >99.9% session retention

#### Firestore sync in real time with offline caching fallback

#### Graceful fallback if AI or chat is unavailable

### **8. Scalability**

#### Firebase Firestore allows handling increasing volumes of users and data.

#### Algolia integration allows scalable search without Firestore query limitations.

#### Firebase Functions and extension-based architecture support future modular upgrades.

### **9. Availability**

#### Hosted on Firebase/Vercel → 99.95% uptime SLA

#### Real-time sync ensures always-fresh data

#### AI fallbacks (e.g., cached suggestions) can be used if external APIs are down.

### **10. Maintainability**

#### Modular file structure: features/, hooks/, services/, and context/.

#### Shared logic extracted into useProfileLogic, useAskFormLogic, and custom services.

#### Easily extendable architecture allows adding future features (badges, gamification,...).

#### Clean code and version control using GitHub.

#### 

#### **דרישות ממשק חיצוניות:**

### **1. API Interfaces**

* The system interacts with the **OpenAI API** to:  
  + Rephrase user questions.
  + Generate AI-based answers.
  + Suggest relevant learning resources.
* The system communicates with **Algolia Search API** to:  
  + Provide fast, indexed search of questions.
  + Enable filtering, ranking, and instant results.

### **2. Database Interface (DB)**

* Uses **Firebase Firestore** for all backend data management:  
  + Users, questions, answers, requests, profiles, reputation system.
  + Real-time data sync between clients and database.
* Uses **Firebase Storage** for:  
  + Uploading and retrieving images, files, and documents.
* All Firestore queries are filtered, paginated, and optimized.

### **3. Authentication Interface**

* **Firebase Authentication** with:  
  + Email/password login.
  + Google and GitHub OAuth providers.
  + Password reset via email.
  + Protected routes

### **4. External Libraries and Services**

* **Tailwind CSS** for styling and responsive design.
* **React InstantSearch (Algolia)** for advanced search components.
* **Framer Motion** for UI animations.
* **Lucide Icons** and **Heroicons** for consistent iconography.

### **5. Platform Access**

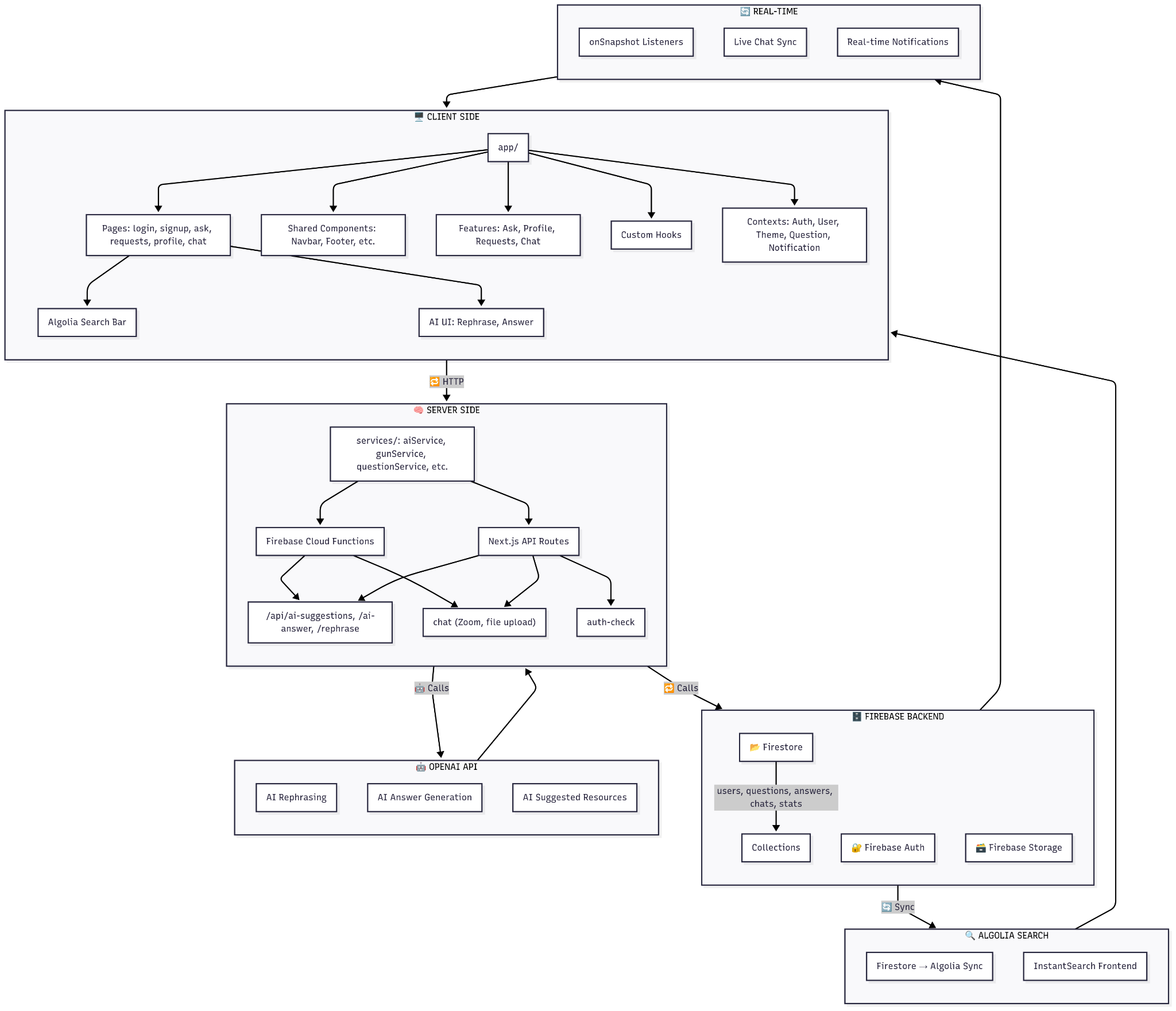
* The platform is accessible via any modern browser:  
  + Chrome, Edge, Safari.
* Mobile-friendly layout with responsive support across screen sizes.
* No installation required; fully web-based.

### **6. Peer-to-Peer Communication Interface (Gun.js)**

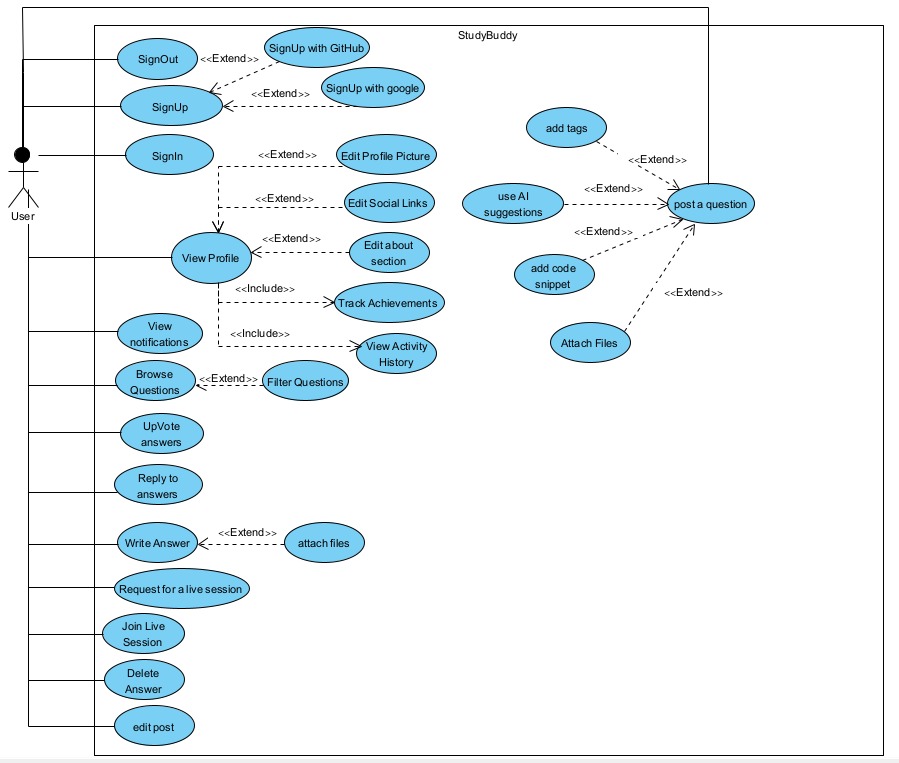
The system integrates with **Gun.js** (a decentralized, peer-to-peer database) to:

* Enable real-time communication and chat between users in a decentralized manner.
* Reduce dependency on central servers for private messaging.
* Enhance fault tolerance and real-time data sharing in chat.
* Allow message syncing even when users reconnect after disconnection.

4**.הציגו ארכיטקטורה מעודכנת של האתר (תרשים הכולל את האלמנטים המרכזיים):**

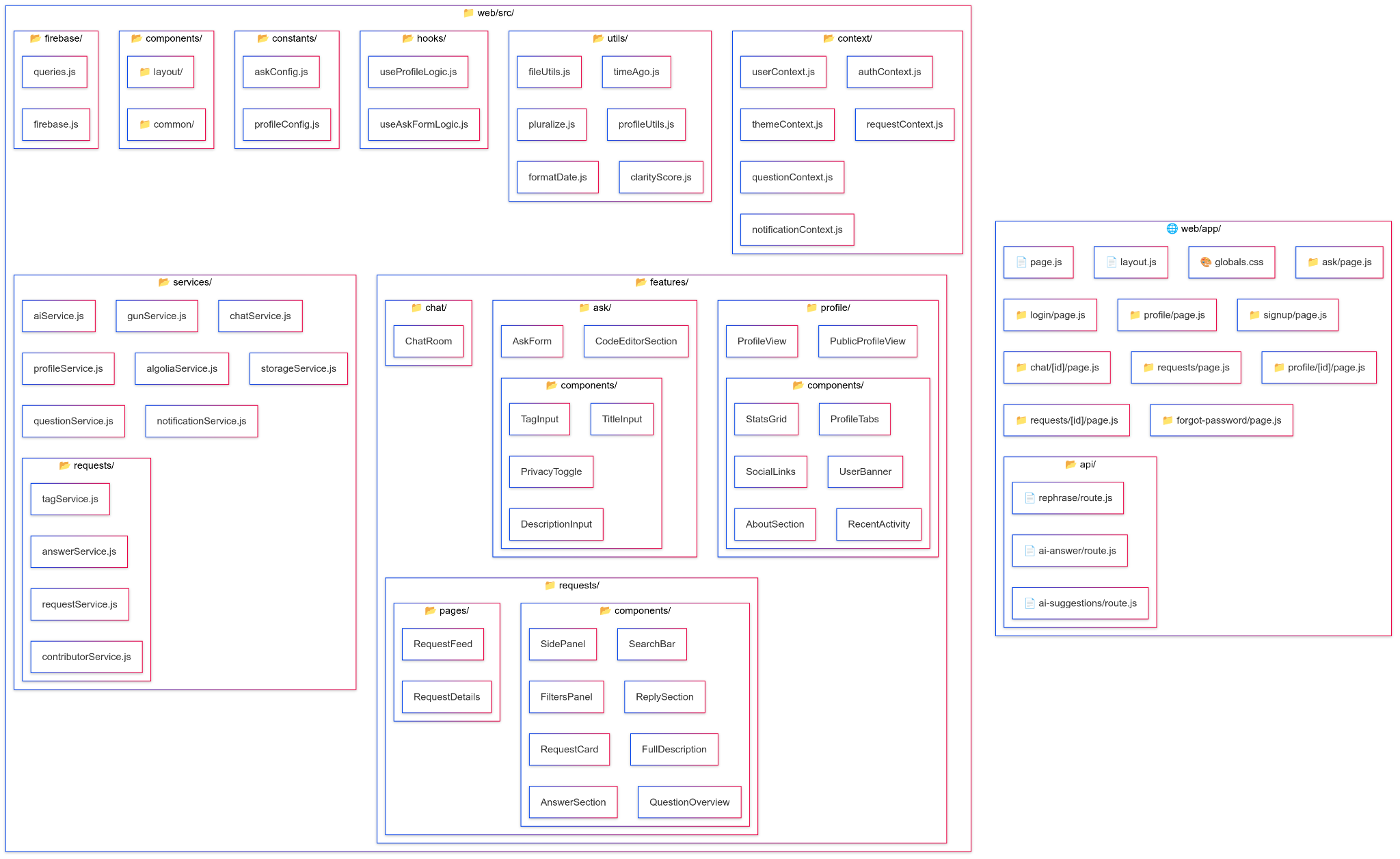


**5.דיאגרמת use case:**

****

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.



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

A diagram of a diagram

AI-generated content may be incorrect.

**6. יש להגיש תיק למתכנת:**

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

### **1.( handleSubmit(Ask**

**תיאור**: מטפלת בשליחת שאלה חדשה בטופס ה־Ask. מאמתת את השדות, מעלה קבצים אם קיימים, קוראת לפונקציות כמו uploadFiles, generateAnswer,createRequest, ולבסוף שומרת את הבקשה ל־Firestore.  
 **קלט**: פרטי טופס השאלה, קבצים (לא חובה)  
 **פלט**: יצירת מסמך חדש ב־Firestore  
 **מיקום**: hooks/useAskFormLogic.js

### **2. rephraseQuestion**

**תיאור**: משתמשת בממשק OpenAI כדי לנסח מחדש את השאלה שהוזנה על ידי המשתמש. נועדה לשפר בהירות ודיוק של השאלות המפורסמות.  
 **קלט**: טקסט שאלה  
 **פלט**: טקסט מנוסח מחדש  
 **מיקום**: services/aiService.js

### **3. uploadFiles**

**תיאור**: מעלה קבצים (כגון תמונות, PDF, מסמכים) לאחסון של Firebase Storage ומחזירה קישורים לקבצים.  
 **קלט**: מערך קבצים (FileList), מיקום אחסון  
 **פלט**: מערך של קישורים לצפייה/שיתוף  
 **מיקום**: services/storageService.js

### **4. fetchRequestsPaginated**

**תיאור**: מבצעת טעינה מדורגת של בקשות עזרה, לפי עמודים, נושאים, מצב שאלה וטווח זמן. מאפשרת חוויית גלילה מתמשכת.  
 **קלט**: פילטרים (subject, status, timeRange), limit, עמוד  
 **פלט**: מערך בקשות + מצביע למסמך האחרון  
 **מיקום**: services/requests/requestService.js

### **5.** **createRequest**

### **תיאור**: פונקציה זו יוצרת בקשת עזרה חדשה במסד הנתונים (Firestore). היא מאפסת שדות כמו זמן יצירה (createdAt), סטטוס פתוח (open) ושדה תגובות ריק כברירת מחדל. בנוסף, היא שומרת את הכותרת גם באותיות קטנות לצורכי חיפוש ומחזירה את הבקשה החדשה עם המזהה שלה.

### **קלט**:

### requestData – אובייקט הכולל את פרטי הבקשה (כותרת, תיאור, תגיות, קבצים וכו’)

### **פלט**:

### אובייקט בקשה חדש המכיל את כל המידע, כולל המזהה שהתקבל מהמסד (id)

### **מיקום**: services/requests/requestService.js

### **6. handleSubjectSelect**

### תיאור: פונקציה זו מטפלת בלחיצה על נושא (subject) במסנן הבקשות. אם המשתמש לוחץ על אותו נושא פעמיים – הנושא מתבטל, אחרת הוא מתעדכן. כך המשתמש יכול לסנן את רשימת הבקשות לפי נושא בודד.

### קלט:

### subject – מחרוזת שמציינת את שם הנושא שנבחר.

### פלט:

### מעדכנת את הפילטרים דרך הקריאה ל־onFilterChange עם סטייט חדש המכיל את הנושא הנבחר או null.

### מיקום: features/requests/components/FiltersPanel.js

### **7. handleAnswerSubmit**

**תיאור**:  
 פונקציה זו מטפלת בשליחת תשובה על שאלה בדף הבקשה. היא מאמתת את הקלט, מעלה קבצים ל־Firebase אם יש, יוצרת מערך קבצים מצורפים, ואז קוראת לפונקציה addAnswer כדי לשמור את התשובה במסד הנתונים. בסיום הפעולה, התוכן והקבצים מתאפסים.

**קלט**:

e – אובייקט אירוע של הטופס (onSubmit)

newAnswerContent, files, user, requestId – משתנים מקומיים מתוך הקומפוננטה

**פלט**:

אין פלט ישיר – הפעולה יוצרת תשובה חדשה במסד הנתונים ומשנה את ה־state הפנימי של הרכיב

**מיקום**:  
 features/requests/components/AnswerSection.js

### **8. handleSubmitReply**

### תיאור: פונקציה זו אחראית על שליחת תגובה (Reply) לתשובה מסוימת. היא מאמתת שהתוכן אינו ריק, ובמידה והמשתמש מחובר – קוראת לפונקציה addReply שמוסיפה את התגובה למסד הנתונים תחת מזהה התשובה. בסיום, התגובה מתאפסת ונעצרת תצוגת כתיבת תגובה.

### קלט:

### e – אובייקט אירוע של שליחת הטופס

### newReplyContent, user, answerId – משתנים מקומפוננטת התגובות

### פלט:

### אין פלט ישיר – נשלחת תגובה חדשה למסד ונעשה reset לשדות

### מיקום: features/requests/components/ReplySection.js

### **9. generateAnswer**

**תיאור**: מבצע שימוש ב־OpenAI כדי להחזיר תשובה אוטומטית לשאלה שפורסמה. שימושי כשאין מענה אנושי.  
 **קלט**: תוכן שאלה  
 **פלט**: טקסט תשובה מ־GPT  
 **מיקום**: services/aiService.js

### **10. getUserProfile**

**תיאור**: טוען את פרטי הפרופיל של המשתמש המחובר (שם, כינוי, ביוגרפיה, תמונה).  
 **קלט**: userId  
 **פלט**: אובייקט פרופיל  
 **מיקום**: services/profileService.js

### **11. getUserStats**

**תיאור**: מחזיר מידע סטטיסטי כגון כמות שאלות, תשובות, דירוג, תגיות, לתצוגה בלוח המשתמש.  
 **קלט**: userId  
 **פלט**: מספרים ותגים  
 **מיקום**: services/profileService.js

### **12. buildRecentActivity**

**תיאור**: יוצר רשימת פעילויות אחרונות של המשתמש, להצגה בכרטיסיית "היסטוריה" בפרופיל.  
 **קלט**: userId  
 **פלט**: מערך פעילויות  
 **מיקום**: utils/profileUtils.js

### **13. createNotification**

**תיאור**:  
 פונקציה זו יוצרת התראה חדשה למשתמש במאגר הנתונים (Firestore). היא מוסיפה לתיעוד גם את זמן היצירה (createdAt) ומגדירה את ההתראה כברירת מחדל כ"לא נקראה" (read: false). לאחר מכן מחזירה את ההתראה החדשה כולל מזהה (id).

**קלט**:

notificationData - אובייקט שמכיל מידע על ההתראה:  
 userId, type, message, requestId, answerId, fromUserId, ועוד

**פלט**:

אובייקט ההתראה שנוצר כולל id, createdAt ו־read

**מיקום**:  
 services/notificationService.js

### **14. findOrCreateChat**

### תיאור: הפונקציה מחפשת אם קיים צ'אט פרטי בין שני משתמשים (userA ו־userB) לפי מזהים. אם כן – מחזירה את מזהה הצ'אט הקיים. אם לא – יוצרת צ'אט חדש עם שני המשתתפים ושומרת אותו ב־Firestore עם שדה createdAt.

### קלט:

### userA – מזהה המשתמש הראשון

### userB – מזהה המשתמש השני

### פלט:

### מזהה (ID) של הצ'אט הקיים או הצ'אט החדש שנוצר

### מיקום: services/[chatService.js](http://chatservice.js)

### **ב. יש לפרט:**

### **● את כל ה -API שהשתמשתם בהם:**

## **External APIs**

### **1. OpenAI API**

* **Endpoint**: https://api.openai.com/v1/chat/completions
* **Model**: gpt-3.5-turbo
* **Authentication**: Bearer Token (OPENAI\_API\_KEY)
* **Used for**:  
  + יצירת תשובות מבוססות AI
  + ניסוח מחדש של שאלות
  + הצעת מקורות לימוד מותאמים

### **2. GunDB Relay Server**

* **Endpoint**: https://gun-manhattan.herokuapp.com/gun
* **Purpose**: תמיכה בצ'אט מבוזר בזמן אמת (P2P)
* **Service File**: gunService.js
* **Usage**: העברת הודעות בין משתמשים בצ'אט פרטי

## **Internal APIs (Next.js API Routes)**

### **1. /api/ai-answer**

* **Method**: POST
* **Purpose**: הפקת תשובה לשאלה באמצעות GPT
* **Input**: { question: string }
* **Output**: { answer: string }
* **Used in**:  
  + aiService.js → generateAnswer
  + AnswerSection.js, FullDescription.js, QuestionOverview.js

### **2. /api/rephrase**

* **Method**: POST
* **Purpose**: ניסוח מחדש לשאלות לא ברורות
* **Input**: { prompt: string }
* **Output**: { rephrased: string }
* **Used in**:  
  + aiService.js → rephraseQuestion
  + TitleInput.js, DescriptionInput.js

### **3. /api/ai-suggestions**

* **Method**: POST
* **Purpose**: הצעת מקורות לימוד מתאימים
* **Input**: { question, description }
* **Output**: { suggestions: [{ title, description, url }] }
* **Used in**: RequestDetails.js

## **Firebase Services**

### **1. Firebase Authentication**

* **Purpose**: התחברות וניהול משתמשים
* **Features**: התחברות עם מייל/סיסמה, Google, GitHub
* **Used throughout**: כל הרכיבים שדורשים אימות

### **2. Cloud Firestore**

* **Purpose**: מסד נתונים בזמן אמת
* **Collections**:  
  + users, requests, answers, replies, notifications, chats, savedQuestions
* **Features**: סנכרון חי, שאילתות, מיון, סינון

### **3. Firebase Storage**

* **Purpose**: אחסון קבצים
* **Used for**: העלאת קבצים לשאלות/תשובות
* **Features**: מעקב אחר התקדמות, קישורי הורדה

### **4. Firebase Analytics**

* **Purpose**: ניתוח שימוש וביצועים
* **Used for**: מעקב אחרי מעורבות המשתמשים

### **● קישור ל- DB,**

[**DATABASE LINK**](https://console.firebase.google.com/u/0/project/study-buddy-3a00a/firestore/databases/-default-/data/~2Frequests~2F0edsRptNnf0EqSiNd3GX)

### **● סביבות מיוחדות שהתקנתם :**

* **Next.js (React Framework – App Router)** **תיאור**: סביבת פיתוח מודרנית לבניית אפליקציות Full-Stack עם React. השתמשנו בגרסת App Router של Next.js, הכוללת רנדרינג בצד השרת, נתיבים מבוססי תיקיות, תמיכה ברכיבי שרת ולקוח, ושילוב קל עם Firebase ו־API פנימיים.
* **Firebase CLI + Firebase Emulator Suite** **תיאור**: השתמשנו ב־Firebase עבור מסד נתונים (Firestore), אחסון קבצים (Firebase Storage), ניהול משתמשים (Firebase Auth), ונוטיפיקציות. באמצעות Firebase CLI ביצענו בדיקות לוקאליות, ניהלנו את בסיס הנתונים, ובדקנו כללים של הרשאות.
* **Vercel (Deployment Platform)** **כתובת פריסה**: <https://study-buddy-ebon-one.vercel.app>/  
   **תיאור**: פרסנו את האתר על Vercel עם אינטגרציה ל־GitHub. כל Push מבצע Deployment אוטומטי, כולל תמיכה בפונקציות Serverless, CI/CD, ו־Preview לכל Pull Request.
* **Thunder Client / Postman** **תיאור**: נבחרו ככלים לבדיקת API פנימיים וחיצוניים – בעיקר עבור בדיקת תקשורת עם שרת OpenAI, קבלת תשובות AI, ושימוש ב־API הפנימי שלנו ב־Next.js.
* **GunDB Relay Server (P2P Chat)** **תיאור**: לצורך מימוש צ’אט מבוסס Peer-to-Peer השתמשנו ב־GunDB. הוא מאפשר תקשורת ריל־טיים מבוזרת בין משתמשים ללא שרת מרכזי, דרך ריליי ציבורי בכתובת:  
   https://gun-manhattan.herokuapp.com/gun
* **Firebase Console + Firestore Rules Debugger** **תיאור**: כלי ניהול גרפי לצפייה, עריכה ובקרה על הנתונים בזמן אמת. השתמשנו ב־Firestore Console לניטור השאילתות, וב־Rules Debugger לבדוק הרשאות בהתאם לרמות גישה.
* **React DevTools + Chrome DevTools** **תיאור**: כלי פיתוח לבדיקת זרימת נתונים בקומפוננטות, פרופיל ביצועים של האפליקציה, ניטור סטייטים ו־Context, ובדיקת רינדור חוזר.

### **● קטעי קוד מיוחדים שלקחתם ממקורות שונים (כולל רפרנסים) :**

### **1. שילוב GunDB לצ'אט P2P**

* **מקור** GunDB Docs
* **שימוש**: אפשרנו יצירת צ'אט בזמן אמת בין משתמשים באמצעות GunDB – ספריית קוד פתוח המאפשרת תקשורת Peer-to-Peer ללא שרת מרכזי. השתמשנו בריליי ציבורי של Gun.
* **רפרנס**: <https://gun.eco/docs/API>

### **2. שימוש ב־onSnapshot לקבלת נתונים בזמן אמת**

* **מקור**: Firebase Docs
* **שימוש**: ניטור חי של שינויים באוסף הבקשות (requests) עם סינון לפי קריטריונים (נושא, סטטוס). אפשר למשתמש לראות עדכונים ללא רענון עמוד.
* **רפרנס**: <https://firebase.google.com/docs/firestore/query-data/listen>

**3. אינטגרציה עם OpenAI API**

* **מקור**: OpenAI API Reference
* **שימוש**: שליחת בקשות ל־ChatGPT (gpt-3.5-turbo) ליצירת תשובות אוטומטיות, שיפור ניסוח השאלה והצעת מקורות לימוד רלוונטיים. נעשה שימוש ב־Bearer Token דרך קובץ סביבה.
* **רפרנס**: [OpenAI – Chat Completions API](https://platform.openai.com/docs/guides/chat)

### **4. Sign in with Google (Firebase Authentication)**

* **מקור**: Firebase Authentication Docs
* **שימוש**: התחברות מהירה דרך חשבון Google .פותח באמצעות signInWithPopup עם ספק Google של Firebase.
* **רפרנס**: <https://firebase.google.com/docs/auth/web/google-signin>

### **5. שליחת מייל לאיפוס סיסמה**

* **מקור**: Firebase Web API Docs
* **שימוש**: שליחת מייל אוטומטי עם קישור לאיפוס סיסמה מתוך עמוד “שכחתי סיסמה”.
* **רפרנס**: <https://firebase.google.com/docs/auth/web/manage-users>

### **6. ניטור מצב התחברות המשתמש**

* **מקור**: Firebase Docs
* **שימוש**: מאזין ל־auth (onAuthStateChanged) על מנת לקבוע אם המשתמש מחובר ולטעון את התוכן בהתאם, כולל הפניית משתמשים לאזורי גישה מוגבלים.
* **רפרנס**: <https://firebase.google.com/docs/auth/web/manage-users#get_the_currently_signed-in_user>

### 

### **● פרומפטים שהשתמשתם בהם בכלי AI:**

### **1. AI Answer Generation Prompt**

**Location:** /api/ai-answer/route.js  
 **Purpose:** Generates an AI-based answer for the student’s submitted question.  
 **Main Use Case:** When a user submits a question, this prompt sends it to OpenAI and returns a generated answer using GPT.

**Prompt Format:**

[

{

role: 'user',

content: question

}

]  
 **Configuration:**

* **Model:** gpt-3.5-turbo
* **Max Tokens:** 300
* **Temperature:** 0.7 (Balanced — combines clarity with creativity)

### **2. Question Rephrasing Prompt**

**Location:** /api/rephrase/route.js  
 **Purpose:** Rephrases vague or unclear questions or descriptions submitted by users.  
 **Main Use Case:** Helps users improve the phrasing of their questions for better clarity and understanding.

**Prompt Format:**

[

{

role: 'system',

content: 'You rephrase unclear student question descriptions into clearer versions.'

},

{

role: 'user',

content: `Rephrase this: ${prompt}`

}

]

**Configuration:**

* **Model:** gpt-3.5-turbo
* **Max Tokens:** 150
* **Temperature:** 0.7  
   **Used In:** TitleInput.js, DescriptionInput.js, and [aiService.js](http://aiservice.js).

### **3. Learning Resource Suggestions Prompt**

**Location:** /api/ai-suggestions/route.js  
 **Purpose:** Suggests relevant learning resources (videos, articles, documentation) based on the user's question.  
 **Main Use Case:** Automatically suggests 3–5 high-quality educational resources that match the user’s question and description.

**Prompt Text:**

You are an expert tutor. Suggest 3–5 high-quality, relevant online resources (videos, articles, or documentation).

For each, provide a title, a short description, and a direct link.

Format your response as a JSON array like:

[

{ "title": "...", "description": "...", "url": "..." },

...

]

Question: ${question}

Description: ${description}

**Configuration:**

* **Model:** gpt-3.5-turbo
* **Max Tokens:** 500
* **Temperature:** 0.7  
   **Used In:** RequestDetails.js via the getSuggestions function.

### **● שמות משתמש וססמאות שהגדרתם לכל סוג משתמש, לצורך חיבור לאתר.**

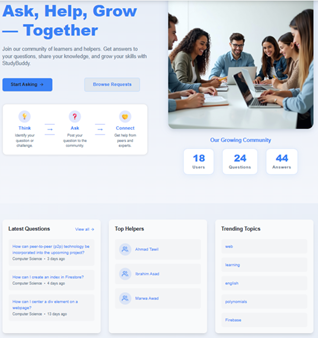
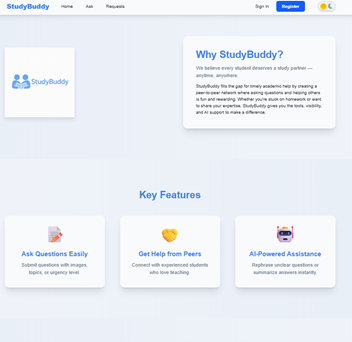
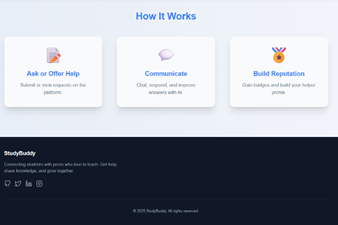
Email : [user@gmail.com](mailto:user@gmail.com)

Password: UserUser123!

7. תיק משתמש :

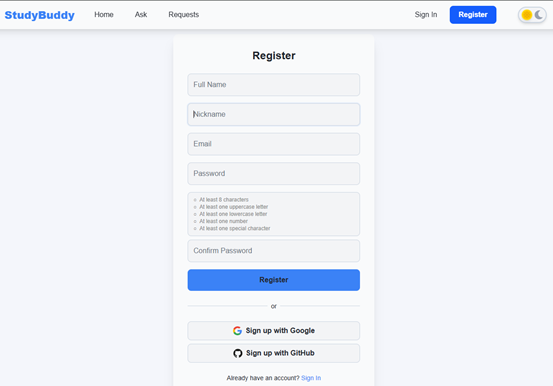
**Home Page:**

When entering the StudyBuddy website, the user is greeted with a welcoming header displaying the slogan "Ask, Help, Grow - Together" that inspires them to join and participate. At the top right corner of the page, there are buttons for Registering and Log In, allowing new users to quickly create an account and returning users to access their profile and activity. Next to these buttons, there is also a light/dark mode toggle, represented by a sun or moon icon, which allows the user to switch between light and dark display modes for a more comfortable and personalized browsing experience. Below the header, two main buttons appear: "Start Asking", which allows the user to immediately submit a new question to the community, and "Browse Requests", which lets the user explore and help with existing questions posted by others. Further down, the page displays community statistics, including the total number of users, questions, and answers, emphasizing the platform's active and growing environment. The next section shows Latest Questions, providing a quick overview of the most recent questions asked, helping users stay updated and easily choose where to contribute. Following that, a Top Helpers section highlights the most active and supportive members of the community, motivating others to become involved and recognized. Finally, at the bottom of the page, a Trending Topics section presents the most popular discussion topics, encouraging users to explore areas of interest and join ongoing conversations. This long, scrollable home page is designed to showcase the vibrant community, promote engagement, and make it easy for new and returning users to start interacting right away. the footer includes a brief description of StudyBuddy’s mission to connect students with peers who love to teach, encourage sharing knowledge, and foster growth together. It also contains social media icons (Twitter, LinkedIn, Instagram, and GitHub), allowing users to follow or connect through external networks. At the bottom, there is a copyright notice confirming all rights reserved to StudyBuddy

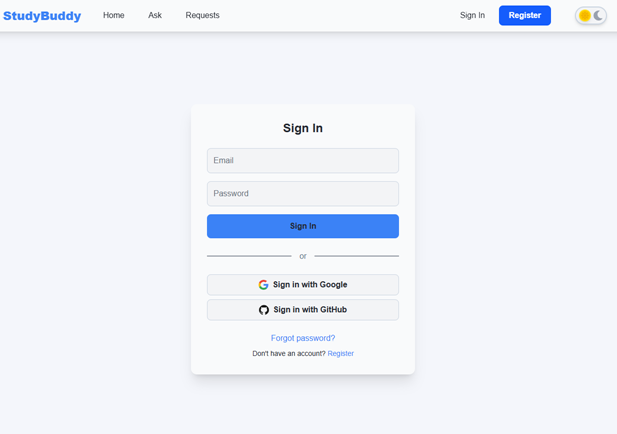
**Register Screen:**

When the user clicks **Register**, they are directed to the registration page, where they can create a new StudyBuddy account. At the top of the form, fields are provided for **Full Name**, **Nickname**, **Email**, **Password**, and **Confirm Password**. The password field includes clear guidelines indicating that the password must contain at least 8 characters, one uppercase letter, one lowercase letter, one number, and one special character, helping users create a secure password. After filling in the fields, the user clicks the blue **Register** button to create their account. Alternatively, users have the option to register using third-party accounts, with dedicated buttons for **Sign up with Google** and **Sign up with GitHub**, allowing faster and easier access without manually entering details. At the bottom of the form, a link labeled **Sign In** is available for users who already have an account, so they can easily switch to the login page



**Sign In Screen**

When the user clicks **Sign In**, they are taken to the login page where they can access their StudyBuddy account. The form includes fields for **Email** and **Password**, allowing users to enter their credentials securely. After filling in their details, the user clicks the blue **Sign In** button to enter the system. Alternatively, users can choose to sign in using a third-party account by clicking **Sign in with Google** or **Sign in with GitHub**, providing a faster and more convenient login option. Below these buttons, there is a **Forgot password?** link, which helps users recover their account if they can’t remember their password. At the bottom of the form, there is a prompt for new users: **Don’t have an account? Register**, with a clickable link that redirects to the registration page



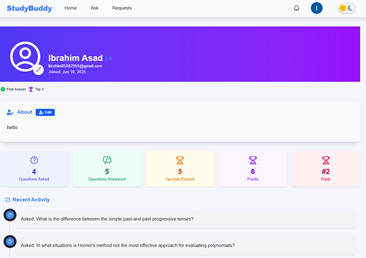
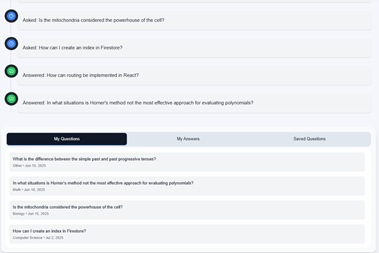
**Profile Screen:**

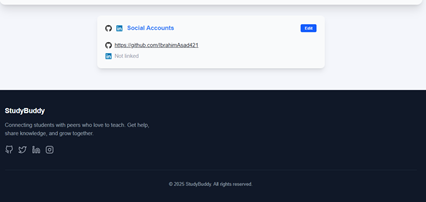
After signing in, users can access their personal **Profile page**, which presents a detailed overview of their activity and achievements on StudyBuddy. At the top of the profile, the user’s **name**, **email**, and **join date** are displayed on a vibrant background, along with status badges (such as First Answer or Top 3), giving a sense of progress and recognition. An **Edit** button next to the name allows the user to update their information. Below this header, there is an **About** section, where the user can write a short personal description and update it via the **Edit** button

Underneath, the profile displays key activity statistics in colorful cards: **Questions Asked**, **Questions Answered**, **Upvotes Earned**, **Points**, and **Rank**, helping users track their engagement and contributions. Further down, a **Recent Activity** timeline visually summarizes the latest actions, including questions asked and answers provided, each with a clear icon to distinguish between them

At the bottom, there is a tabbed section allowing users to switch between **My Questions**, **My Answers**, and **Saved Questions**, providing easy access to their past contributions and saved content for future reference. Each question entry shows its title, category, and posting date for quick identification

Finally, the **Social Accounts** area allows users to connect or display links to their GitHub and LinkedIn profiles, with an **Edit** button for updating these links





**Ask a Question Page:**

On the **Ask a Question** page, users are encouraged to clearly and specifically post new questions to the community. At the top of the form, there is a **Title** field with a clarity score indicator and an option to **Rephrase with AI** to help improve the question title automatically. Below, the **Description** box allows users to provide detailed explanations of their question, also with an AI rephrase option for better clarity

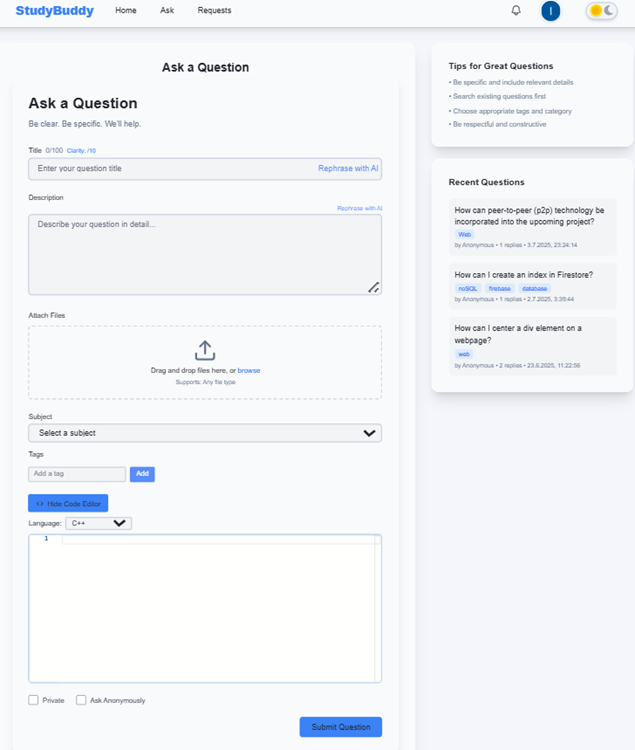
Users can upload supporting materials using the **Attach Files** section, where they can drag and drop files or browse their device; any file type is supported. Next, a **Subject** dropdown helps categorize the question properly, and **Tags** can be added to improve searchability and relevance

A key feature on this page is the **Code Snippet** section, which can be expanded to show a dedicated code editor. Users can select the programming language from a dropdown (for example, C++, JavaScript, etc.) and paste or write their code directly into the editor, making it especially useful for technical and coding-related questions

Below the main inputs, there are two checkboxes: **Private**, which makes the question visible **only to users who are logged in**, and **Ask Anonymously**, which hides the user’s identity when posting

On the right side, a **Tips for Great Questions** panel provides helpful guidelines such as being specific, searching for similar questions first, choosing appropriate tags and categories, and maintaining respectful and constructive language. Underneath, a **Recent Questions** section lists the latest posts from other users, making it easy to see if related questions have already been addressed

Finally, at the bottom of the form, the user clicks the blue **Submit Question** button to publish their question to the community



**Requests Page:**

On the **Requests** page, users can browse, filter, and contribute to help requests from other community members. At the top, a large header invites users to engage with existing questions and provide answers. Just below, a **search bar** allows users to search for specific requests using keywords

Underneath, category filters (such as **Math**, **Physics**, **Computer Science**, **Chemistry**, **Biology**, and **Other**) help users narrow down the displayed questions by subject. There is also a dropdown filter to choose between all questions or specific groups, and a toggle to **Show only unanswered** requests, making it easier for helpers to find questions that still need responses

Each help request is shown as a card displaying the **title**, **author name**, **date and time**, and a brief preview of the question. Tags under each card highlight related topics (for example, web, noSQL, firebase), and a small bookmark icon allows users to save questions for later. A comment icon indicates the number of answers the request has received, and a **View** link opens the full question details

On the right side, a **Popular Tags** panel shows the most used tags and the number of questions under each, enabling quick topic-based exploration. Below that, a **Top Contributors** panel lists users with the most answers, encouraging engagement and friendly competition among community members

A screenshot of a computer

AI-generated content may be incorrect.Further down on the right, a **Need Help?** box prompts users to ask their own question if they don’t find what they're looking for, with a prominent **Ask a Question** button linking directly to the ask page

**Request Details Page:**

When user clicks **View** on a specific help request from the Requests page, they are taken to the **Request Details** page, which provides a complete view of that question and its responses. At the top, a **Back to Requests** button allows users to easily return to the list of all requests. The main section displays the **question title**, along with the author’s name, date and time, and tags related to the question (such as noSQL, firebase, database). The user can also see the number of answers, and they have the options to **Edit**, **Save**, or **Share** the question for broader visibility

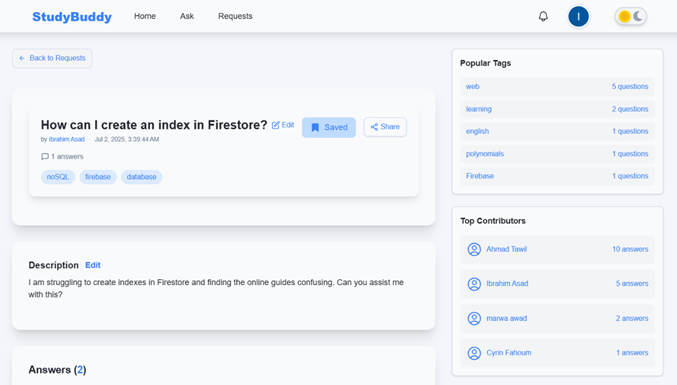
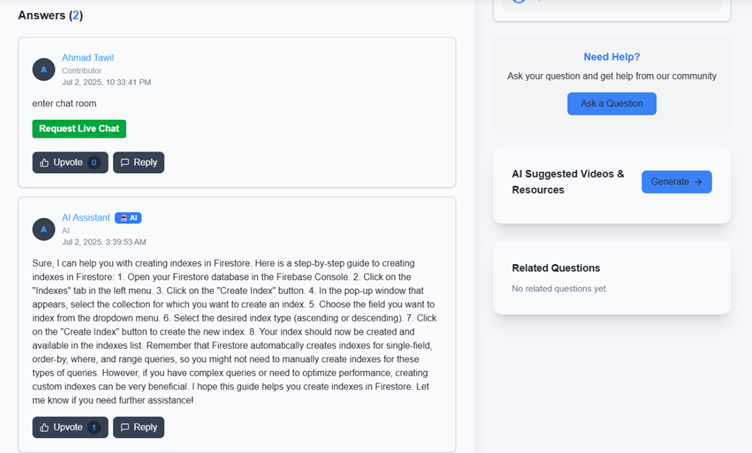
Below, the **Description** section shows the full text submitted by the question author, with an **Edit** link to allow modifications if needed

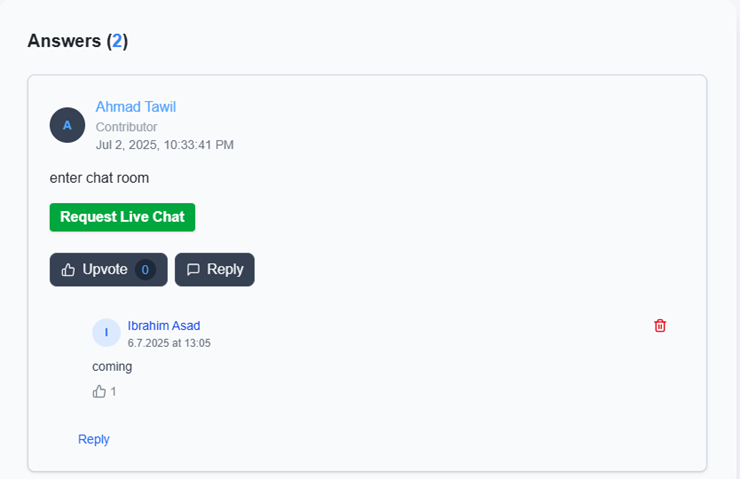
The **Answers** section follows, showing each response in a card layout. Each answer card displays the responder’s name, role (e.g., Contributor), date and time, and the answer content. Users can interact with answers by clicking **Upvote** to support helpful responses or **reply** to engage in further discussion. If live help is needed, a **Request Live Chat** button may appear, allowing for real-time support

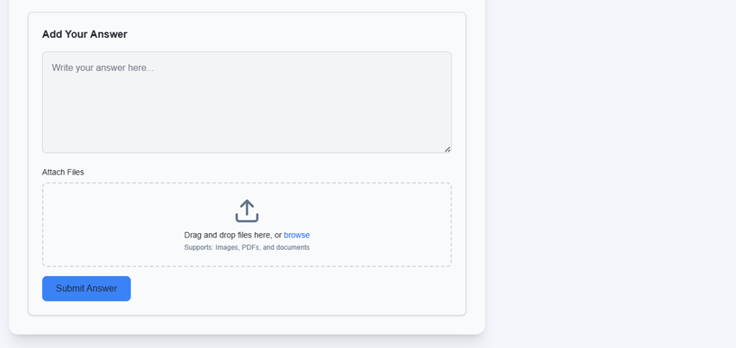
Answers can be posted by community members or generated by the integrated **AI Assistant**, which provides automated, detailed, step-by-step guidance when appropriate

On the right side, there are supporting panels including **Popular Tags** (to explore related topics), **Top Contributors** (showing the most active community members), and a **Need Help?** box with a quick link to ask a new question. There is also an **AI Suggested Videos & Resources** section with a **Generate** button to provide further learning materials, and a **Related Questions** box (which may be empty if no similar posts exist yet)

At the bottom of the page, an **Add Your Answer** form allows users to submit their own response. This form includes a large text area for writing the answer and an **Attach Files** section supporting images, PDFs, and documents. Users finalize their submission by clicking the blue **Submit Answer** button







**Live Chat:**

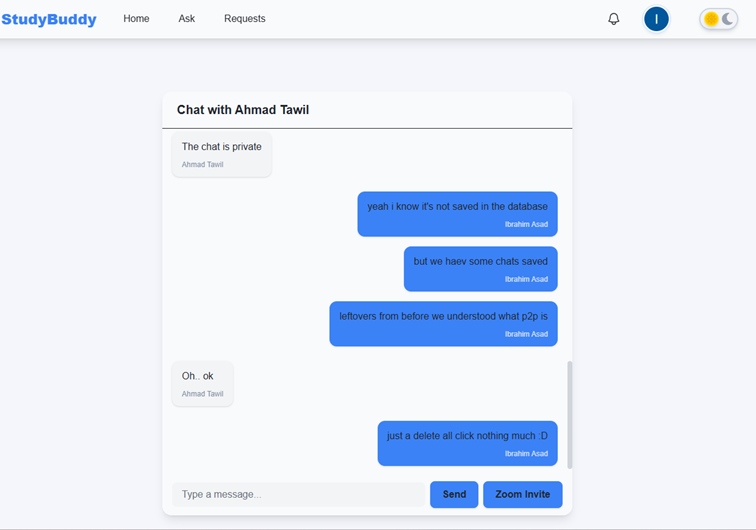
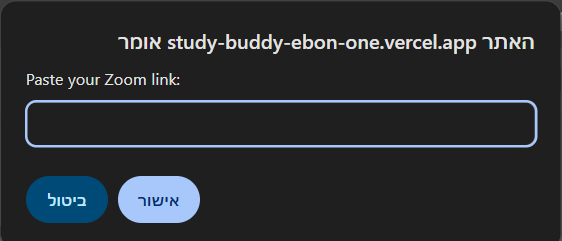
When a user views their **own post** (a question they created), they have the ability to invite contributors who posted answers to start a **private live chat**. The option to request a live chat is visible **only to the original post creator** when they are viewing their own question page. This ensures that not just any user can initiate live chats, but only the question owner can invite specific answer contributors for more detailed or real-time support

When the post creator clicks **Request Live Chat** under a specific answer, a notification is sent to the selected contributor inviting them to start a private, real-time chat session. Once the contributor accepts, both users are taken to a dedicated **Chat page**, where they can exchange messages instantly

At the top of this page, the chat title displays the contributor's name (e.g., *Chat with Ahmad Tawil*), and it is clearly labeled as private. The messages are shown in a simple conversational format with speech bubbles, including sender names for clarity

At the bottom, there is a text input field labeled **Type a message...**, with **Send** and **Zoom Invite** buttons. Clicking **Zoom Invite** opens a pop-up dialog where the user can **Paste your Zoom link**, confirm (אישור), or cancel (ביטול). This makes it easy to continue the discussion over a live video call if needed

The chat is not saved in the database and is only visible to the two involved participants, ensuring privacy

**סרטון :**

[**https://www.dropbox.com/scl/fi/ewjf0vygz7v0ymdbrm2ls/pitch\_elevator\_B13.mp4?rlkey=haj94kyywfsozvmx8zauqw3ld&st=slqksyvu&dl=0**](https://www.dropbox.com/scl/fi/ewjf0vygz7v0ymdbrm2ls/pitch_elevator_B13.mp4?rlkey=haj94kyywfsozvmx8zauqw3ld&st=slqksyvu&dl=0)