

נושא פרויקט : סביבת הרצה מקוונת לשפת C

שם מלא : שרה יולי סמישקיס

ת"ז : 216764803

מנחה : אופיר שביט

שם החלופה : סייבר ומערכות הפעלה

תיכון ע"ש חיים הרצוג כפר סבא



תוכן עניינים

4	מבוא
4	ייזום
4	תיאור ראשוני של המערכת
4	הגדרת הלקוח
4	הגדרת יעדים ומטרות
5	בעיות תועלות וחסכוניות
6	סקירת פתרונות קיימים
6	סקירת טכנולוגיית הפרויקט
7	סייגים בהגדרת המערכת
7	תיחום הפרויקט
8	אפיון
8	תיאור מפורט של המערכת
8	פירוט יכולות השרת
9	פירוט יכולות הלקוח
10	פירוט יכולות המשותפות לשרת וללקוח
11	פירוט בדיקות
13	תכנון וניהול לוח זמנים
13	ניהול הסיכונים
14	תיאור תחום הידע
14	יכולות בצד השרת
18	יכולות בצד הלקוח
21	יכולות משותפות
23	ארכיטקטורה של הפרויקט
23	הארכיטקטורה המערכת תחולק לשלוש שכבות עיקריות:
23	Frontend
23	Backend
23	Database
24	החומרה
25	הטכנולוגיה הרלוונטית
25	שפות תכנות
25	מסגרות ותשתיות
25	כלים ומנגנונים נוספים
26	זרימת המידע במערכת הרשמה והתחברות
27	עריכת קוד
28	הידור והרצת קוד
29	עבודה על תרגיל
30	האלגוריתמים
30	עריכת שיתופית בזמן אמת
30	בדיקת קוד מסוכן
31	הגבלת משאבים בהרצת קוד
31	בדיקה אוטומטית של פתרונות תרגילים
32	אלגוריתמים קיימים
32	מקורות רלוונטיים
32	סביבת הפיתוח
32	כלי פיתוח
33	פירוט הסביבה וכלי בדיקה
33	פרוטוקול התקשורת
33	פירוט כלל ההודעות
34	מסכי המערכת
34	מסך כניסה (Login)
35	מסך הרשמה (Register)
36	לוח מחוונים (Dashboard)
37	מסך עריכת פרויקט

38	מסך תרגילים
39	מסך תרגיל
40	מסך ניהול
41	תרשים מסכים
41	מבני הנתונים
41	פירוט מבני הנתונים
44	פירוט מאגרי המידע
45	מסדי נתונים
46	סקירת חולשות והאיומים
46	שכבת האפליקציה
46	שכבת התעבורה
46	שכבת המערכת
47	מימוש הפרויקט
47	חלק א' - סקירת המודולים והמחלקות של המערכת
47	מודולים מיובאים
47	מחלקות שפותחו
50	חלק ב' - אלגוריתמים מרכזיים בפרויקט
50	אלגוריתם הרצת קוד C מאובטח
51	אלגוריתם Operational Transformation לעריכה שיתופית
52	אלגוריתם בדיקת תרגילי קוד אוטומטית
52	חלק ג' - מסמך בדיקות
54	בדיקות אבטחה
55	בדיקות ביצועים
55	בדיקות נוספות שבוצעו
57	מדריך למשתמש
57	פירוט קבצי המערכת - עץ קבצים
58	התקנת המערכת
58	סביבה נדרשת
58	כלים נדרשים
59	משתמשי המערכת
64	סיכום אישי / רפלקציה
64	תהליך העבודה על הפרויקט
64	תהליך הלמידה
64	כלים לעתיד
65	תובנות מהתהליך
65	סיכום
66	ביבליוגרפיה
67	נספחים
67	נספח א': פרוטוקול תקשורת WebSocket
67	הודעות מהלקוח לשרת
68	הודעות מהשרת ללקוח
69	נספח ב': שמות תיקיות וקבצים שהוגבלו לשימוש בסנדבוקס
70	נספח ג': אלגוריתם בדיקת קוד לתבניות מסוכנות
72	קוד הפרויקט:

מבוא

ייזום

תיאור ראשוני של המערכת

הפרויקט כולל פיתוח סביבת הרצה מקוונת לשפת תכנות C, המאפשרת למשתמשים לעבוד בשיתוף פעולה בזמן אמת. בסביבת ההרצה ניתן לכתוב קוד, להדר אותו ולהריץ אותו, לראות את הפלט וכן לתקשר עם משתמשים אחרים. המערכת כוללת גם אזור עם תרגילי לימוד, המאפשרים למשתמשים לתרגל את ידיעותיהם בשפת C באמצעות סדרת תרגילים ברמות קושי שונות.

המערכת מורכבת מצד שרת (Backend) שנכתב בשפת Python עם המסגרת Flask, וצד לקוח (Frontend) שנכתב באמצעות HTML, CSS ו-JavaScript. התקשורת בזמן אמת מתבצעת באמצעות Socket.IO, המאפשר תקשורת דו-כיוונית בין הלקוח לשרת. מידע של משתמשים, פרויקטים ותרגילים נשמר במסד נתונים באמצעות SQLAlchemy.

אחד היתרונות המרכזיים של המערכת הוא היכולת לעבוד בשיתוף פעולה בזמן אמת, כאשר כל המשתמשים יכולים לראות את השינויים שנעשים על ידי משתמשים אחרים באופן מיידי, וכן היכולת להדר ולהריץ קוד C בסביבה מאובטחת.

החלטתי לבחור בפרויקט זה מאחר ובעת לימודי באוניברסיטה, יצא לי ללמוד ולתרגל את שפת C, ואחד מהדברים שזכורים לי מהקורס בו למדתי את השפה, זה שבאחד מהתרגילים הייתה אפשרות להריץ את הקוד ולראות כמה זיכרון דלף בהקצאות הזיכרון שבוצעו בקוד, אך עם הנוחות שסיפק הכלי הוא היה מסובך לשימוש ורוב התלמידים ויתרו על השימוש בו, לכן אני רוצה ליצור סביבת עבודה אשר תהיה זמינה בכל מחשב בלי התקנות מסובכות, אשר תנגיש את הכלים המייעילים ומשפרים תכנות ב-C.

אתגרי פיתוח

- פרויקט זה כלל מספר אתגרים טכניים משמעותיים:
- יצירת מנגנון עריכה שיתופית בזמן אמת עם טיפול בהתנגשויות
- הקמת סביבת ריצה מאובטחת (Sandbox) להרצת קוד C, תוך מניעת קוד זדוני
- פיתוח תקשורת בזמן אמת בין לקוחות מרובים
- הגנה על המערכת מפני איומי אבטחה שונים, כולל ניסיונות SQL Injection ו-Cross-Site Scripting

הגדרת הלקוח

- המערכת מיועדת למשתמש הממוצע שמעוניין להתחיל ללמוד או לתרגל תכנות בשפת C, כגון:
- סטודנטים למדעי המחשב** - המתמודדים עם שפת C במסגרת קורסי מבוא לתכנות או קורסים ממוקדים כמו מערכות הפעלה ורשתות.
 - מורים ומרצים** - המעוניינים בכלי שיאפשר להם להדגים קוד C באופן אינטראקטיבי וללא צורך בהתקנות מסובכות.
 - מתכנתים מתחילים** - המעוניינים ללמוד שפת C באופן עצמאי.
 - צוותי פיתוח** - העובדים על פרויקטים משותפים ומחפשים סביבת עבודה שיתופית מקוונת.
- הלקוחות העיקריים הם אנשים החסרים גישה למשאבים יקרים כגון סביבות פיתוח מקצועיות, או אלה שמעדיפים סביבת עבודה פשוטה ואינטואיטיבית ללא צורך בהתקנות מורכבות.

הגדרת יעדים ומטרות

- עורך קוד שיתופי** - יצירת עורך קוד שיאפשר למספר משתמשים לערוך את אותו הקוד בו-זמנית, עם מנגנון שיתוף בזמן אמת שיציג את מיקום הסמן של כל משתמש והשינויים שהוא מבצע.

- **סביבת הדרה והרצת קוד** - פיתוח מנגנון בטוח להדרת והרצת קוד C, עם הצגת פלט ההרצה בצורה ברורה.
- **תמיכה בלמידה** - יצירת מערכת תרגילים ברמות קושי שונות, עם בדיקה אוטומטית של פתרונות ומתן משוב למשתמש.
- **תקשורת בין משתמשים** - מימוש מנגנון צ'אט בזמן אמת בין משתמשים העובדים על אותו פרויקט, ואינדיקטור המציג אילו משתמשים מחוברים כעת.
- **ניהול משתמשים ופרויקטים** - יצירת מערכת לניהול משתמשים, כולל הרשמה, התחברות ושמירת פרויקטים, עם אפשרות לשתף פרויקטים עם משתמשים אחרים.
- **אבטחה** - הטמעת מנגנוני אבטחה להגנה על המערכת ונתוני המשתמשים, כולל הצפנת סיסמאות והרצת קוד בסביבה מבודדת.

בעיות תועלות וחסכוניות

מהן הבעיות

- **מורכבות הקמת סביבת פיתוח** - הקמת סביבת פיתוח לשפת C דורשת לרוב התקנת מהדר IDE, וכלים נוספים, תהליך שעלול להיות מורכב עבור מתחילים.
- **קושי בעבודה שיתופית** - כלים מסורתיים לפיתוח בשפת C אינם מאפשרים עבודה שיתופית בזמן אמת, ודורשים שימוש במערכות ניהול גרסאות מורכבות.
- **חסור בכלי למידה אינטראקטיביים** - קיים מחסור בכלי למידה אינטראקטיביים לשפת C שמשלבים עריכת קוד, הרצה ותרגול בסביבה אחת.
- **מגבלות זמינות** - כלי פיתוח מסורתיים מוגבלים למחשב אחד ואינם זמינים מכל מקום.

מהן התועלות

- **נגישות** - המערכת זמינה מכל מחשב עם חיבור לאינטרנט, ללא צורך בהתקנות מסובכות.
- **סביבת עבודה אחידה** - כל המשתמשים עובדים באותה סביבה, מה שמונע בעיות תאימות בין מערכות הפעלה וגרסאות מהדר שונות.
- **שיתוף פעולה** - המערכת מאפשרת עבודה שיתופית בזמן אמת, כולל תקשורת ישירה בין משתמשים.
- **למידה אפקטיבית** - שילוב של עריכת קוד, הרצה ותרגול באותה סביבה מייעל את תהליך הלמידה.
- **אבטחה** - הרצת קוד בסביבה מבודדת מגנה על המערכת ועל המשתמשים מפני קוד זדוני.

אילו שירותים תציע המערכת

המערכת מציעה את השירותים הבאים:

1. **עריכת קוד** - עורך קוד עם הדגשת תחביר, השלמת קוד אוטומטית ואפשרויות עריכה מתקדמות.
2. **עריכה שיתופית** - אפשרות לעבוד על אותו קוד בו-זמנית עם משתמשים אחרים, עם הצגת שינויים בזמן אמת.
3. **הדרה והרצת קוד** - אפשרות להדר ולהריץ קוד C באופן מיידי, עם הצגת פלט ההרצה.
4. **תרגול** - סדרת תרגילים ברמות קושי שונות, עם בדיקה אוטומטית של פתרונות ומתן משוב.
5. **תקשורת** - צ'אט בזמן אמת בין משתמשים העובדים על אותו פרויקט.
6. **ניהול פרויקטים** - שמירת פרויקטים, גישה לפרויקטים קודמים ושיתוף פרויקטים עם משתמשים אחרים.

סקירת פתרונות קיימים

- מאחר והפרויקט שלי עוסק בנושאים אשר רבים מימשו כבר, אך לא שילבו בתוצר אחד קשה למצוא פתרון קיים אך אסקור את ההשראות שלי לפרויקט:
- כאשר זה מגיע למהדרי שפת C מקוונים, שני פתרונות קיימים הם:
- **Repl.it** – סביבת פיתוח מקוונת התומכת בתכנות בסי ומציעה שיתוף עריכה בזמן אמת, הרצת קוד בסביבת sandbox, אימות משתמשים ושיתוף פרוייקטים. אך עם זאת איננה מעניקה את ההתמקדות הישירה בלימוד C ולא מכילה אופציה לשלב valgrind.
 - **CoderPad** – סביבת תכנות מקוונת בעבור ראיונות טכניים, תומכת ב-C, מאפשרת שיתוף בזמן אמת בעבור תכנות בזוגות, אך גם בה לא קיימת ההתמקדות בתרגול C ושימוש ב-valgrind.
 - **Codeanywhere** – סביבת פיתוח מבוססת ענן בעבור תכנות ב-C בה כלולות אפשרות לעריכה משותפת, טרמינל פנימי, סביבת פיתוח מוגבלת אך גם לה אין שום אזור למידה המתמקד ב-C.
 - **GitHub Codespaces** – סביבת פיתוח מבוססת ענן אשר תומכת ב-C, מאפשרת עריכה בשיתוף, אך גם לא מתמקד בתרגול C.
 - **CodePen** – למרות שמתמקד בעיקר בפיתוח web, האתר כן מציע שיתוף בזמן אמת, version control ואירוח משאבים. עם זאת לא תומך ב-C.
- בהשוואה לפתרונות אלו, הפרויקט שלי מציע יתרונות ייחודיים:
1. התמקדות בשפת C עם כלים ייעודיים להדרה והרצה.
 2. שילוב של עריכה שיתופית, הרצת קוד ותרגול באותה סביבה.
 3. ממשק פשוט ואינטואיטיבי המותאם במיוחד למתחילים.
 4. מערכת תרגול מובנית עם משוב אוטומטי.
 5. אפשרות להרחבה עתידית עם כלים נוספים כמו Valgrind לניתוח זיכרון.

סקירת טכנולוגיית הפרויקט

הפרויקט משתמש במגוון טכנולוגיות קיימות ומשלב אותן באופן מיטבי:

1. **צד שרת: (Backend)**
 - שפת Python עם מסגרת Flask לניהול בקשות HTTP
 - Socket.IO לתקשורת בזמן אמת
 - SQLAlchemy לתקשורת עם מסד נתונים
 - Flask-Bcrypt להצפנת סיסמאות
 - Resource, Subprocess לניהול הרצת קוד בסביבה מבודדת
2. **צד לקוח: (Frontend)**
 - HTML, CSS ו-JavaScript
 - Socket.IO-client לתקשורת בזמן אמת
 - CodeMirror לעריכת קוד עם הדגשת תחביר
 - Bootstrap לעיצוב ממשק המשתמש
3. **תשתיות:**
 - Docker לניהול סביבת הפיתוח והפריסה
 - PostgreSQL כמסד נתונים

הפרויקט אינו מנסה להמציא טכנולוגיות חדשות, אלא משלב טכנולוגיות קיימות באופן אפקטיבי כדי ליצור פתרון מקיף ומוכוון משתמש.

סייגים בהגדרת המערכת

דרישות חומרה :

- שרתים עם ביצועים גבוהים נדרשים לתמיכה בריבוי משתמשים וקומפילציה מקבילית
- רוחב פס רשת משמעותי נדרש לתמיכה בתקשורת Socket.IO בזמן אמת

מגבלות אבטחה :

- איזון בין פונקציונליות לאבטחה בסביבת ה-Sandbox
- הגנה מפני התקפות DDOS בשירות מקוון

מורכבות טכנית :

- סנכרון מדויק נדרש בין לקוחות מרובים בעריכה שיתופית
- ניהול עומסים בקומפילציה והרצת קוד במקביל

מגבלות שפה :

- התמקדות בשפת C בלבד מגבילה את קהל היעד הפוטנציאלי

תלות בתשתית חיצונית :

- הסתמכות על מהדר C חיצוני (GCC) יוצרת תלות בגרסאות ועדכונים

תיחום הפרויקט

1. רשתות :

- תקשורת בזמן אמת באמצעות Socket.IO
- HTTP/HTTPS לבקשות REST API
- אבטחת תקשורת באמצעות TLS/SSL
- סנכרון נתונים בין משתמשים מרובים

2. מערכות הפעלה :

- סביבת הרצה מבודדת (Sandbox) לביצוע קוד
- אינטגרציה עם מערכת ההפעלה לקומפילציה והרצת קוד C
- ניהול משאבי מערכת לתמיכה במשתמשים מרובים

3. אבטחת מידע :

- יישום סביבת הרצה מבודדת (Sandbox)
- מניעת SQL Injection
- אימות משתמשים והצפנת סיסמאות
- הגבלת משאבים לקוד שמורץ במערכת

4. עריכת קוד שיתופית :

- יישום עריכה שיתופית בזמן אמת
- סנכרון שינויים וטיפול בהתנגשויות
- הצגת מיקום הסמן של משתמשים אחרים

תחומים בהם המערכת אינה עוסקת :

1. פיתוח מהדר C עצמאי
2. יצירת סביבת פיתוח מלאה (IDE) עם כל התכונות המתקדמות
3. ניהול גרסאות קוד
4. אופטימיזציה של קוד C
5. דיבוג מתקדם של תוכניות C
6. תמיכה בשפות תכנות מלבד C

אפיון

תיאור מפורט של המערכת

המערכת היא סביבת עריכה והרצה מקוונת לקוד C עם יכולות עריכה שיתופית. בכניסה למערכת, המשתמש מתבקש להתחבר או להירשם. לאחר ההתחברות, המשתמש מוצג בפני לוח מחוונים (dashboard) המציג את הפרויקטים שלו, פרויקטים משותפים ורגילי תכנות בהם החל.

המערכת מאפשרת יצירת פרויקטים חדשים, שיתוף פרויקטים עם משתמשים אחרים, ועבודה על תרגילי תכנות מובנים. בעת עבודה על פרויקט, המערכת מציגה עורך קוד עם הדגשת תחביר, רשימת משתמשים מחוברים, אפשרות לתקשורת באמצעות צ'אט, וכפתורים להדירה והרצת הקוד.

הידור והרצת קוד מתבצעים בצד השרת בסביבה מבודדת, עם הגבלות על משאבי מערכת כדי למנוע שימוש לרעה. פלט ההידור וההרצה מוצג למשתמש, כולל הודעות שגיאה במידת הצורך.

לצד פרויקטים אישיים, המערכת מציעה סדרת תרגילים ברמות קושי שונות. כל תרגיל כולל תיאור, קוד התחלתי, ובדיקה אוטומטית של פתרונות.

פונקציונליות המערכת כוללת:

1. הרשמה והתחברות משתמשים
2. יצירה וניהול פרויקטים
3. עריכת קוד C עם הדגשת תחביר
4. עריכה שיתופית בזמן אמת
5. הידור והרצת קוד C
6. צ'אט בין משתמשים
7. תרגול מובנה עם בדיקה אוטומטית של פתרונות
8. ניהול הרשאות גישה לפרויקטים

אבטחת המערכת מתבססת על אימות משתמשים, הצפנת סיסמאות, הרצת קוד בסביבה מבודדת, וניקוי קלט למניעת התקפות Injection.

פירוט יכולות השרת

תקשורת Socket.IO בזמן אמת

- ניהול חיבורים מרובים של לקוחות
- העברת עדכוני עריכה בין משתמשים בזמן אמת
- סנכרון מצב הפרויקט בין כל המשתמשים
- העברת הודעות צ'אט בין משתמשים

הידור וביצוע קוד C בסביבה מאובטחת

- קבלת קוד מהלקוח וניקוי קלט
- בדיקת קוד לזיהוי דפוסים מסוכנים
- הידור הקוד באמצעות GCC בסביבה מבודדת
- הרצת הקוד המהודר עם הגבלות משאבים
- איסוף פלט ההידור וההרצה והחזרתו ללקוח

אימות משתמשים וניהול הרשאות

- רישום משתמשים חדשים והצפנת סיסמאות
- אימות משתמשים בכניסה למערכת
- יצירה וניהול סשנים
- בקרת גישה לפרויקטים לפי הרשאות

ניהול נתונים

- שמירת נתוני משתמשים, פרויקטים ותרגילים במסד נתונים
- שמירת היסטוריית קומפילציה
- ניהול פרויקטים משותפים והרשאות גישה

ניהול עריכה משותפת

- סנכרון שינויי עריכה בין כל המשתמשים
- טיפול בהתנגשויות בעריכה מקבילית
- שמירת מיקומי סמנים של כל המשתמשים

אבטחה

- הצפנת תקשורת באמצעות TLS/SSL
- בדיקת קוד לזיהוי דפוסים מסוכנים
- הגבלת משאבי מערכת בהרצת קוד
- מניעת SQL Injection

פירוט יכולות הלקוח**עריכת קוד C עם הדגשת תחביר**

- עורך קוד מבוסס CodeMirror
- הדגשת תחביר C
- מספור שורות
- השלמת קוד אוטומטית

צפייה בשיתוף פעולה בזמן אמת

- הצגת שינויי קוד של משתמשים אחרים בזמן אמת
- הצגת מיקום הסמן של משתמשים אחרים
- רשימת משתמשים מחוברים עם מחוון זמינות

הפעלת הידור והרצת קוד

- כפתורי הידור והרצה
- הכנסת קלט לתוכנית
- הצגת פלט ההידור וההרצה

ניהול פרויקטים

- יצירת פרויקט חדש
- שמירת פרויקט
- הזמנת משתמשים לשיתוף פעולה

תקשורת בין משתמשים

- צ'אט בזמן אמת בין משתמשים בפרויקט
- התראות על התחברות/התנתקות משתמשים

עבודה על תרגילים

- גישה לרשימת תרגילים
- סינון תרגילים לפי רמת קושי וסטטוס
- הגשת פתרונות וקבלת משוב אוטומטי

פירוט יכולות המשותפות לשרת וללקוח

סנכרון שינויי עריכה בזמן אמת

- זיהוי שינויים בקוד בצד הלקוח
- העברת השינויים לשרת
- עיבוד השינויים והפצתם לכל הלקוחות
- יישום השינויים בעורך הקוד של כל לקוח

טיפול בהתנגשויות

- זיהוי התנגשויות בעריכה מקבילית
- יישום אלגוריתם למיזוג שינויים מתנגשים
- עדכון כל הלקוחות בתוצאת המיזוג

שיתוף פרויקטים

- הזמנת משתמשים לשיתוף פעולה

- הגדרת הרשאות גישה
- סנכרון מצב הפרויקט בין כל המשתתפים

ניהול תרגילים

- הצגת תרגילים ובדיקת פתרונות בצד השרת
- הצגת תרגילים וממשק הגשה בצד הלקוח
- מעקב אחר התקדמות המשתמש

פירוט בדיקות

בדיקות פונקציונליות

בדיקה	מטרה	תהליך	תוצאה מצופה	תוצאה בפועל
עריכת קוד ותצוגת תחביר	לוודא שעורך הקוד מציג את הקוד בצורה נכונה עם הדגשת תחביר	הזנת קטעי קוד C שונים ובדיקת הצגתם	הקוד מוצג עם הדגשת תחביר נכונה	הקוד מוצג כראוי עם הדגשת תחביר מתאימה לפי סטנדרט C
קומפילציה והרצת קוד תקין	לוודא שקוד תקין מהודר ומורץ כראוי	הזנת קוד C תקין, הידור והרצה	הקוד מהודר ומורץ בהצלחה, הפלט מוצג כראוי	הקוד הודר והורץ בהצלחה, הפלט הוצג במסך הפלט
קומפילציה של קוד שגוי	לוודא שקוד שגוי מייצר הודעות שגיאה מתאימות	הזנת קוד C עם שגיאות תחביר, הידור	הצגת הודעות שגיאה מתאימות	הוצגו הודעות שגיאה מדויקות מהמהדר
עריכת שיתופית	לוודא שמספר משתמשים יכולים לערוך את אותו הקוד בו-זמנית	חיבור מספר משתמשים לאותו פרויקט וביצוע שינויים	השינויים נראים אצל כל המשתמשים בזמן אמת	השינויים עודכנו בזמן אמת אצל כל המשתמשים המחוברים
פתרון התנגשויות	לוודא שהמערכת מטפלת נכון בהתנגשויות עריכה	עריכה של אותו קטע קוד על ידי שני משתמשים בו-זמנית	התנגשויות מטופלות באופן חלק, ללא איבוד מידע	ההתנגשויות טופלו כראוי, השינויים מוזגו באופן תקין
תרגול תכנות	לוודא שמערכת התרגול עובדת כראוי	גישה לתרגיל, הגשת פתרון, קבלת משוב	התרגיל נטען כראוי, הפתרון נבדק ומתקבל משוב	התרגיל נטען, הפתרון נבדק והתקבל משוב מדויק
צ'אט	לוודא שמשתמשים יכולים לתקשר באמצעות הצ'אט	שליחת הודעות בין משתמשים בפרויקט	ההודעות מתקבלות אצל כל המשתמשים בפרויקט	ההודעות התקבלו בזמן אמת אצל כל המשתמשים בפרויקט
יצירת פרויקט חדש	לוודא שמשתמש יכול ליצור פרויקט חדש	לחיצה על "פרויקט חדש", הזנת שם ויצירה	פרויקט חדש נוצר ומוצג למשתמש	פרויקט חדש נוצר בהצלחה ומוצג בלוח המחוונים
הזמנת משתמש לפרויקט	לוודא שניתן להזמין משתמשים לפרויקט	הזנת שם משתמש בטופס ההזמנה	המשתמש מוזמן לפרויקט ויכול לגשת אליו	המשתמש הוזמן בהצלחה והפרויקט מופיע ברשימת

הפרויקטים המשותפים שלו				רישום משתמש חדש
משתמש חדש נוצר בהצלחה ומועבר ללוח המחוונים	משתמש חדש נוצר ומועבר ללוח המחוונים	הזנת פרטי משתמש חדש בטופס הרישום	לוודא שתהליך הרישום עובד	

בדיקות אבטחה

בדיקה	מטרה	תהליך	תוצאה מצופה	תוצאה בפועל
הצפנת תקשורת	לוודא שהתקשורת בין הלקוח לשרת מוצפנת	בדיקת תעבורת הרשת באמצעות כלי ניטור	כל התקשורת באמצעות TLS/SSL	כל התקשורת הוצפנה כנדרש, לא ניתן לקרוא את התוכן
אימות משתמשים	לוודא שרק משתמשים מורשים יכולים לגשת למערכת	ניסיון כניסה עם פרטים שגויים	המערכת דוחה ניסיונות כניסה לא מורשים	ניסיונות כניסה עם פרטים שגויים נדחו כנדרש
הרשאות גישה	לוודא שמשתמשים יכולים לגשת רק לפרויקטים שלהם או שהוזמנו אליהם	ניסיון גישה לפרויקט ללא הרשאה	המערכת מונעת גישה לפרויקטים ללא הרשאה	ניסיונות גישה לפרויקטים ללא הרשאה נחסמו
Sandbox לקוד	לוודא שקוד מסוכן לא יכול לפגוע במערכת	ניסיון הרצת קוד זדוני	המערכת מזהה קוד מסוכן ומונעת את הרצתו	קוד מסוכן זוהה ונחסם, עם הודעה מתאימה למשתמש
SQL Injection	לוודא שהמערכת מוגנת מפני SQL Injection	ניסיון הזנת קלט זדוני בשדות טופס	המערכת מנקה את הקלט ומונעת SQL Injection	ניסיונות SQL Injection נחסמו, הקלט נוקה כראוי
הגבלת משאבים	לוודא שתוכניות לא יכולות לצרוך יותר מדי משאבים	הרצת קוד עם לולאה אינסופית או צריכת זיכרון גבוהה	המערכת מגבילה את צריכת המשאבים ומפסיקה תוכניות חריגות	תוכניות שניסו לצרוך יותר מדי משאבים הופסקו אוטומטית

בדיקות ביצועים

בדיקה	מטרה	תהליך	תוצאה מצופה	תוצאה בפועל
עומסים	לוודא שהמערכת מתפקדת היטב תחת עומס	סימולציה של מספר רב של משתמשים בו-זמנית	המערכת תפקדה באופן יציב עד 50 משתמשים בו-זמנית	המערכת תפקדה באופן יציב עד 50 משתמשים בו-זמנית
זמני תגובה	לוודא שהמערכת מגיבה במהירות	מדידת זמן התגובה לפעולות שונות	זמני תגובה קצרים ועקביים	זמני תגובה היו מתחת ל-500 מילישניות לרוב הפעולות

סנכרון בזמן אמת	לוודא שעדכוני עריכה מסונכרנים בזמן אמת	מדידת הזמן בין ביצוע שינוי להופעתו אצל משתמשים אחרים	השינויים מסונכרנים תוך פחות משנייה	השינויים הופיעו בתוך 300 מילישניות בתנאי רשת רגילים
-----------------	--	--	------------------------------------	---

תכנון וניהול לוח זמנים

שלב	תיאור	זמן מתוכנן	זמן בפועל	סטטוס
1	תכנון וחקר	2 שבועות	3 שבועות	הושלם
2	הקמת תשתית בסיסית	1 שבוע	1 שבוע	הושלם
3	מערכת אימות וניהול משתמשים	1 שבוע	1.5 שבוע	הושלם
4	עורך קוד בסיסי	2 שבועות	2 שבועות	הושלם
5	תקשורת WebSocket	1 שבוע	2 שבועות	הושלם
6	עריכה שיתופית	3 שבועות	4 שבועות	הושלם
7	קומפילציה והרצה	2 שבועות	3 שבועות	הושלם
8	מערכת תרגילים	2 שבועות	2.5 שבועות	הושלם
9	פאנל ניהול	1 שבוע	1 שבוע	הושלם
10	בדיקות ותיקונים	2 שבועות	3 שבועות	הושלם

ניהול הסיכונים

הסיכון	פירוט הסיכון	רמת הסיכון	דרכי התמודדות	איך טופל בפועל
קושי בסנכרון עריכה שיתופית	התנגשויות בעריכה עלולות לגרום לאיבוד מידע	גבוהה	שימוש באלגוריתם מיזוג מתקדם, ניטור עריכות בזמן אמת	מומש אלגוריתם מיזוג יעיל, נוספה תצוגת סמנים של משתמשים אחרים
אבטחת Sandbox	קוד זדוני עלול לפרוץ מהSandbox	גבוהה	הגבלת משאבים, סינון קוד מסוכן, הרצה בסביבה מבודדת	מומש מנגנון בדיקת קוד מסוכן, הוגבלו משאבי מערכת, נוספו בדיקות אבטחה
עומס על השרת	ריבוי משתמשים עלול להעמיס על השרת	בינונית	תכנון לסקלביליות, מנגנוני איזון עומסים	שימוש בתצורת Docker מוכנה לסקלביליות, מיטוב שאליות מסד נתונים
בעיות תאימות בדפדפנים	ייתכנו הבדלים בתצוגה ובתפקוד בין דפדפנים	נמוכה	בדיקות תאימות, שימוש בספריות סטנדרטיות	שימוש ב Bootstrap-3 לתאימות, בדיקות ב-3 דפדפנים נפוצים
קשיים בהטמעת Socket.IO	חיבורים בזמן אמת עלולים להיות מאתגרים	בינונית	למידה מעמיקה של Socket.IO, קפדני של פרוטוקול ההודעות	הוקדש זמן ללמידת Socket.IO נכתב פרוטוקול מובנה ומתועד היטב
בעיות במסד הנתונים	התנגשויות ואובדן מידע במסד הנתונים	בינונית	שימוש בטרנזקציות, גיבויים תכופים	יושמו טרנזקציות לכל פעולות העדכון, הוקם מנגנון גיבוי אוטומטי

תיאור תחום הידע

יכולות בצד השרת

ניהול משתמשים ואימות

שם היכולת: הרשמה למערכת

מהות: רישום משתמש חדש במערכת עם קליטת פרטים אישיים נדרשים ובדיקת תקינות הנתונים.

אוסף יכולות נדרשות:

- ממשק משתמש – מסך הרשמה עם שדות קלט
- קליטת נתונים – שם משתמש, אימייל וסיסמה
- בדיקת תקינות - ולידציה של פורמט אימייל ועוצמת סיסמה
- הצפנה - הצפנת הסיסמה באמצעות bcrypt
- שליחה לשרת - העברת הנתונים לשרת באמצעות HTTP POST
- בדיקת תקינות מול בסיס הנתונים - בדיקת ייחודיות שם משתמש ואימייל
- קבלת תשובה מהשרת - קבלת אישור או הודעת שגיאה
- הצגת התשובה למשתמש - הצגת הודעת הצלחה או שגיאה

אובייקטים נחוצים: מודל user, ממשק הרשמה, מנגנון הצפנה, מסד נתונים

שם היכולת: התחברות למערכת

מהות: אימות זהות משתמש קיים וכניסה למערכת.

אוסף יכולות נדרשות:

- ממשק התחברות עם שדות שם משתמש וסיסמה
- ולידציה בסיסית של נתונים
- שליחת בקשה לשרת
- בדיקת קיום המשתמש במסד הנתונים
- אימות הסיסמה מול הגרסה המוצפנת
- יצירת הפעלה (session) למשתמש
- הפניה לדף הראשי או הצגת שגיאה

אובייקטים נחוצים: מודל user, מנגנון session, מנגנון אימות

ניהול פרויקטים

שם היכולת: יצירת פרויקט חדש

מהות: יצירת פרויקט קוד חדש עם הגדרת בעלות והרשאות.

אוסף יכולות נדרשות:

- קליטת שם פרויקט ותיאור
- ולידציה של ייחודיות שם הפרויקט
- יצירת רשומה חדשה במסד הנתונים
- קביעת בעלות על הפרויקט
- הגדרת הרשאות ברירת מחדל
- החזרת מזהה פרויקט למשתמש

אובייקטים נחוצים: מודל project, מודל user, מנגנון הרשאות

שם היכולת: הזמנת משתתפים

מהות: הוספת משתמשים נוספים לפרויקט קיים עם הגדרת הרשאות.

אוסף יכולות נדרשות:

- בדיקת הרשאות המזמין (בעלים או מנהל)
- חיפוש המשתמש המוזמן במסד הנתונים
- בדיקה שהמשתמש לא כבר חבר בפרויקט
- הוספת המשתמש לרשימת המשתתפים
- הגדרת רמת הרשאה (קריאה/כתיבה/ניהול)
- שליחת הודעה למשתמש המוזמן

אובייקטים נחוצים: מודל user, מודל project, מנגנון הרשאות, מערכת הודעות

עריכה שיתופית בזמן אמת

שם היכולת: סנכרון שינויי קוד

מהות: העברת שינויי קוד בין כל המשתמשים בפרויקט בזמן אמת תוך שמירה על עקביות.

אוסף יכולות נדרשות:

- קליטת פעולת עריכה מהמשתמש (הוספה/מחיקה/החלפה)
- הפיכת הפעולה לפורמט Operational Transformation
- העברת הפעולה לכל המשתמשים בפרויקט
- שום הפעולה על המסמך המקומי
- פתרון התנגשויות בעריכה
- שמירת המסמך המעודכן

אובייקטים נחוצים: מנגנון WebSocket, אלגוריתם OT, מנגנון ניהול מצב

שם היכולת: מעקב מיקום cursor

מהות: הצגת מיקום cursor של כל משתמש למשתמשים האחרים.

אוסף יכולות נדרשות:

- קליטת מיקום cursor מהמשתמש
- שידור המיקום למשתמשים אחרים
- הצגת cursor משתמשים אחרים
- עדכון מיקום בזמן אמת
- הסרת cursor עם התנתקות משתמש

אובייקטים נחוצים: מנגנון WebSocket, ממשק עורך הקוד

קומפילציה והרצה של קוד C

שם היכולת: קומפילציה מאובטחת

מהות: קומפילציית קוד C שנכתב על ידי משתמש תוך הבטחת אבטחה.

אוסף יכולות נדרשות:

- קבלת קוד מהמשתמש
- בדיקת הקוד לתבניות מסוכנות
- יצירת קובץ זמני בסביבה מבודדת
- הפעלת מקמפל GCC עם פרמטרים מוגבלים
- קליטת הודעות שגיאה וזמן קומפילציה
- מחיקת קבצים זמניים
- החזרת תוצאות קומפילציה

אובייקטים נחוצים: מנגנון sandbox, מקמפל GCC, מעבד קבצים זמניים

שם היכולת: הרצה מבודדת

מהות: הרצת הקוד הקומפל בסביבה מבודדת ומוגבלת.

אוסף יכולות נדרשות:

- יצירת תהליך מבודד
- הגבלת משאבים (זיכרון, זמן)
- הפעלת הקובץ הבינארי
- קליטת פלט ושגיאות
- הפסקת תהליך במקרה של חריגה
- מחיקת קבצים זמניים
- החזרת תוצאות הרצה

אובייקטים נחוצים: מנגנון sandbox, מגביל משאבים, מעבד I/O

מערכת תרגילים**שם היכולת:** יצירת תרגיל חדש**מהות:** יצירת תרגיל תכנות עם הגדרת מקרי בדיקה וקריטריונים.**אוסף יכולות נדרשות:**

- ממשק יצירת תרגיל עם שדות מתאימים
- הגדרת רמת קושי וקטגוריה
- הוספת מקרי בדיקה (קלט/פלט צפוי)
- הגדרת קוד התחלתי אופציונלי
- שמירת פתרון מלא לתרגיל
- ולידציה של נתוני התרגיל

אובייקטים נחוצים: מודל Exercise , מודל TestCase , ממשק יצירה**שם היכולת:** בדיקה אוטומטית של פתרון**מהות:** בדיקת פתרון תלמיד מול מקרי הבדיקה של התרגיל.**אוסף יכולות נדרשות:**

- קבלת קוד פתרון מהתלמיד
- קומפילציה של הפתרון
- הרצת הפתרון עם כל מקרה בדיקה
- השוואת הפלט לפלט הצפוי
- חישוב ציון או אחוזי הצלחה
- שמירת תוצאות הבדיקה
- עדכון התקדמות התלמיד

אובייקטים נחוצים: מנגנון קומפילציה, מעבד מקרי בדיקה, מודל Progress**פאנל ניהול****שם היכולת:** ניהול משתמשים**מהות:** מתן כלים למנהל לניהול משתמשי המערכת.**אוסף יכולות נדרשות:**

- הצגת רשימת כל המשתמשים
- חיפוש וסינון משתמשים
- צפייה בפרטי משתמש ופעילותו
- חסימה או ביטול חסימה של משתמש

- איפוס סיסמה
- מחיקת משתמש

אובייקטים נחוצים: מודל User, ממשק ניהול, מנגנון הרשאות

שם היכולת: סטטיסטיקות מערכת

מהות: הצגת נתונים סטטיסטיים על שימוש במערכת.

אוסף יכולות נדרשות:

- ספירת משתמשים פעילים
- ניתוח דפוסי שימוש
- מעקב ביצועי מערכת
- גרפים וטבלאות סטטיסטיקה
- דוחות תקופתיים
- התראות על בעיות

אובייקטים נחוצים: מנגנון analytics, גנרטור דוחות, מסד נתונים

יכולות בצד הלקוח

ממשק משתמש אינטראקטיבי

שם היכולת: ניווט במערכת

מהות: מתן ממשק נוח וידידותי לניווט בין חלקי המערכת השונים.

אוסף יכולות נדרשות:

- תפריט ראשי עם קישורים לחלקים שונים
- breadcrumb navigation למעקב מיקום
- תפריטי צד ודרופ-דאון
- כפתורי חזרה ואמלל
- חיפוש גלובלי במערכת
- עיצוב רספונסיבי

אובייקטים נחוצים: UI, CSS framework, JavaScript events

שם היכולת: הודעות משוב

מהות: הצגת הודעות משוב למשתמש על פעולות שביצע.

אוסף יכולות נדרשות:

- הודעות הצלחה בצבע ירוק

- הודעות שגיאה בצבע אדום
- הודעות אזהרה בצבע כתום
- הודעות מידע בצבע כחול
- אנימציות הופעה והעלמה
- סגירה אוטומטית לאחר זמן

אובייקטים נחוצים: notification, CSS animations, JavaScript timers

עורך קוד מתקדם

שם היכולת: הדגשת syntax

מהות: הדגשת קוד C עם צבעים המתאימים לרכיבי השפה השונים.

אוסף יכולות נדרשות:

- זיהוי מילות מפתח של C
- הדגשת מחרוזות ומספרים
- הדגשת הערות בקוד
- הדגשת סוגרים תואמים
- הדגשת שגיאות syntax
- תמיכה בעדכון בזמן אמת

אובייקטים נחוצים: מנוע syntax highlighting , רגולר expressions , עורך קוד

שם היכולת: השלמה אוטומטית

מהות: הצעת השלמות לקוד בזמן הקלדה.

אוסף יכולות נדרשות:

- זיהוי הקשר הקלדה
- הצעת מילות מפתח רלוונטיות
- הצעת שמות פונקציות ומשתנים
- הצגת רשימת אפשרויות
- בחירה באמצעות מקלדת או עכבר
- הכנסת ההשלמה לקוד

אובייקטים נחוצים: מאגר מילות מפתח, autocomplete engine, UI popup

תקשורת בזמן אמת

שם היכולת: חיבור WebSocket

מהות: יצירת וניהול חיבור WebSocket יציב לשרת.

אוסף יכולות נדרשות:

- יצירת חיבור WebSocket לשרת
- טיפול באירועי חיבור ונתק
- שליחת הודעות לשרת
- קבלת הודעות מהשרת
- Reconnection אוטומטי במקרה נתק
- ניהול מצב החיבור

אובייקטים נחוצים: WebSocket API, event handlers, reconnection logic

שם היכולת: סנכרון עדכונים

מהות: קבלת עדכוני קוד ממשתמשים אחרים וישומם בעורך המקומי.

אוסף יכולות נדרשות:

- קבלת פעולות עריכה מהשרת
- ישום הפעולות על הקוד המקומי
- שמירת מיקום cursor נוכחי
- עדכון הקוד ללא הפרעה לעריכה
- התמודדות עם התנגשויות
- שמירת consistency

אובייקטים נחוצים: code editor, API event handlers, OT algorithm

ניהול מצב אפליקציה

שם היכולת: ניהול state מקומי

מהות: ניהול מצב האפליקציה בצד הלקוח.

אוסף יכולות נדרשות:

- שמירת נתוני משתמש מחובר
- שמירת רשימת פרויקטים
- שמירת מצב עורך הקוד
- שמירת העדפות משתמש
- סנכרון עם שרת בעת הצורך
- ניקוי נתונים בהתנתקות

אובייקטים נחוצים: state management, local storage, session management

יכולות משותפות*Websocket תקשורת***שם היכולת:** WebSocket ניהול הודעות**מהות:** הגדרה ויישום של פרוטוקול תקשורת מובנה בין לקוח ושרת.**אוסף יכולות נדרשות:**

- הגדרת פורמט הודעות JSON
- routing של הודעות לפי type
- ולידציה של תוכן הודעות
- טיפול בשגיאות תקשורת
- logging של הודעות
- throttling למניעת spam

אובייקטים נחוצים: JSON parser, message router, validator*ניהול הפעלות משותפות***שם היכולת:** Operational Transformation**מהות:** אלגוריתם לסנכרון עריכה שיתופית ללא התנגשויות.**אוסף יכולות נדרשות:**

- הגדרת פעולות עריכה (insert/delete/retain)
- transformation של פעולות מתנגשות
- composition של פעולות רצופות
- שמירת היסטוריה לביטול
- ולידציה של עקביות
- recovery מכישלונים

אובייקטים נחוצים: Operation classes, Transform engine, History manager*אבטחת מידע***שם היכולת:** sanitization ו- validation**מהות:** וידוא תקינות ובטיחות של נתונים הנכנסים למערכת.**אוסף יכולות נדרשות:**

- ולידציה של פורמט נתונים
- sanitization של תוכן מסוכן

- בדיקת אורך ותוכן שדות
- מניעת XSS וSQL injection
- הצפנת נתונים רגישים
- logging של ניסיונות התקפה

אובייקטים נחוצים: validator functions, sanitizer, encryption utilities

ארכיטקטורה של הפרויקט

המערכת מתוכננת כסביבת הרצה מקוונת לשפת C עם דגש על למידה, תרגול ועבודה שיתופית.

הארכיטקטורה

המערכת תחולק לשלוש שכבות עיקריות:

Frontend

שכבת הצגה מיושמת באמצעות HTML, CSS ו-JavaScript ומספקת ממשק משתמש ידידותי עם העורך קוד, אזור תרגול, אזור פלט וצ'אט. היא מתקשרת עם שכבת היישום באמצעות HTTP ו-WebSocket.

רכיבים עיקריים:

- **עורך קוד:** מבוסס על CodeMirror עם תמיכה בהדגשת תחביר C וסנכרון בזמן אמת
- **אזור תרגול:** מציג תרגילים ומאפשר למשתמשים לעבוד עליהם
- **אזור פלט:** מציג את תוצאות ההידור וההרצה
- **צ'אט:** מאפשר תקשורת בין משתמשים
- **לוח מחוונים:** מציג את הפרויקטים של המשתמש ומאפשר ניהול שלהם

Backend

שכבת היישום מיושמת באמצעות Python עם מסגרת Flask ומטפלת בבקשות מהלקוחות, בהידור והרצת קוד, בסנכרון שינויים ובתקשורת עם מסד הנתונים.

רכיבים עיקריים:

- **בקר משתמשים:** מטפל ברישום, התחברות וניהול משתמשים
- **בקר פרויקטים:** מטפל ביצירה, עריכה וניהול של פרויקטים
- **בקר תרגילים:** מטפל בהצגה ובדיקה של תרגילים
- **מנגנון הידור והרצה:** מהדר ומריץ קוד C בסביבה מאובטחת
- **מנגנון סנכרון:** מטפל בסנכרון שינויים בין לקוחות
- **Socket.IO Server:** מאפשר תקשורת בזמן אמת בין לקוחות לשרת ובין לקוחות לבין עצמם

Database

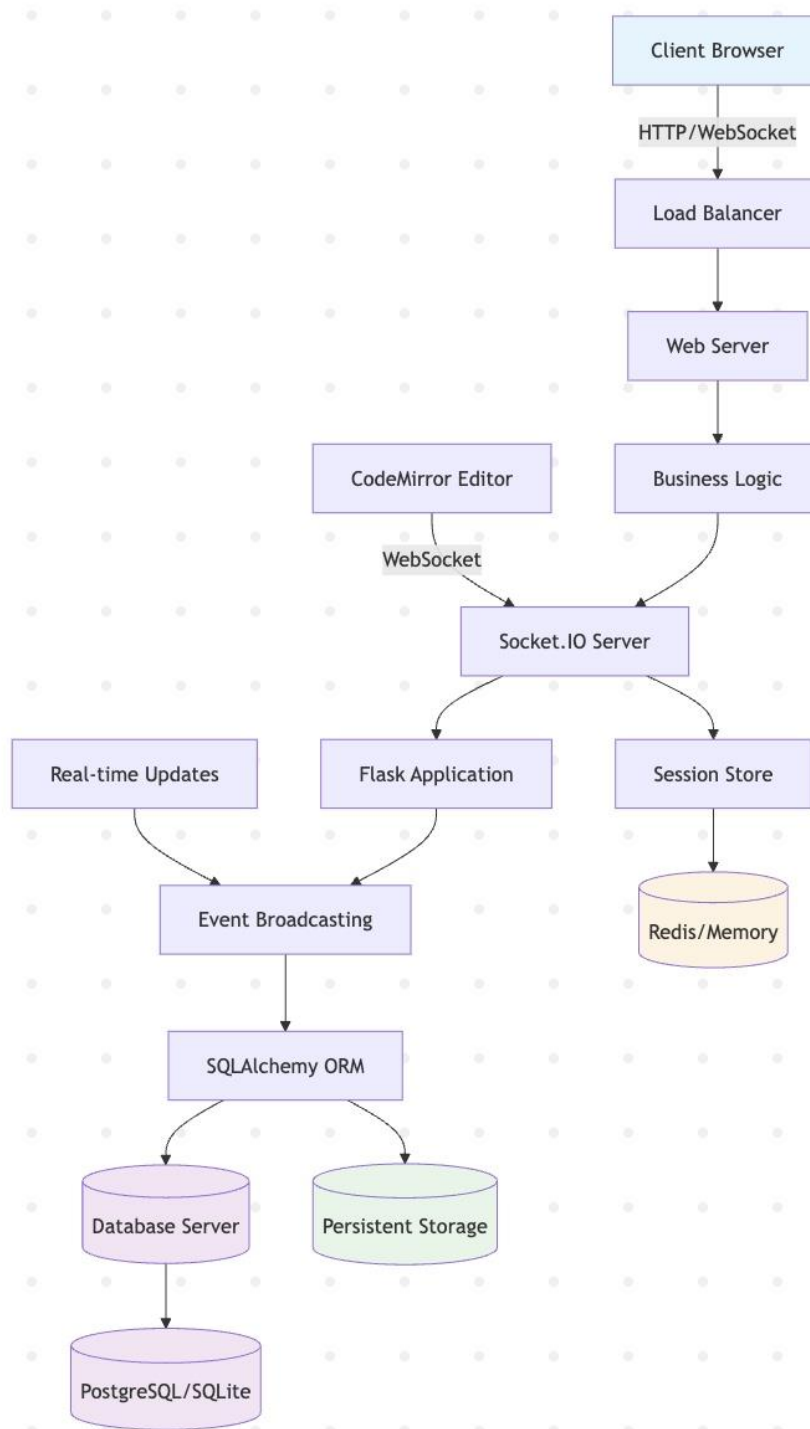
שכבת הנתונים מיושמת באמצעות מסד נתונים SQL עם ממשק SQLAlchemy ומאחסנת את כל הנתונים של המערכת.

רכיבים עיקריים:

- **טבלת משתמשים:** שומרת מידע על משתמשי המערכת
- **טבלת פרויקטים:** שומרת מידע על פרויקטים של משתמשים
- **טבלת תרגילים:** שומרת מידע על תרגילי תכנות
- **טבלת התקדמות:** שומרת מידע על התקדמות משתמשים בתרגילים

- **טבלת היסטוריית קומפילציה:** שומרת מידע על ניסיונות קומפילציה והרצה

החומרה



המערכת מתוכננת לרוץ על שרת עם משאבים מספיקים לתמיכה בריבוי משתמשים וקומפילציה מקבילית. הדרישות המינימליות הן:

- מעבד : 4 ליבות או יותר
- זיכרון : 8 GB RAM או יותר
- אחסון : 50 GB או יותר
- רוחב פס רשת : לפחות 100Mbps

בסביבת הפיתוח, המערכת יכולה לרוץ על מחשב מקומי עם Docker שמספק סביבה מבודדת להרצת השרת ומסד הנתונים.

הטכנולוגיה הרלוונטית

שפות תכנות

שפה	שימוש	נימוק הבחירה
Python	Backend development	גמישות, ספריות עשירות, קהילה גדולה
JavaScript	Frontend development	שפת הדפדפן הסטנדרטית, אקוסיסטם עשיר
HTML5	מבנה דפים	סטנדרט מודרני עם תמיכה בתכונות מתקדמות
CSS3	עיצוב ממשק	עיצוב מתקדם ורספונסיבי
C	קוד המורץ במערכת	השפה שהמערכת מיועדת ללמד

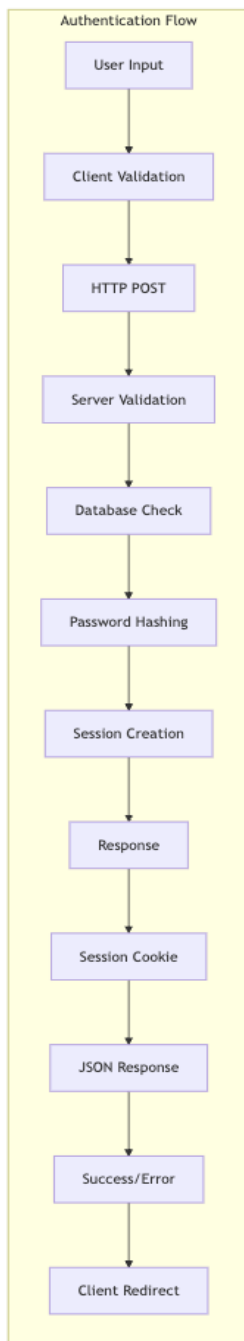
מסגרות ותשתיות

מסגרת	תפקיד	יתרונות
Flask	Web framework	פשוט, גמיש, מהיר לפיתוח
Socket.IO	Real-time communication	תקשורת דו-כיוונית יציבה
SQLAlchemy	ORM	ניהול מסד נתונים מובנה
CodeMirror	Code editor	עורך קוד מתקדם ומותאם
Bootstrap	CSS framework	עיצוב רספונסיבי מהיר

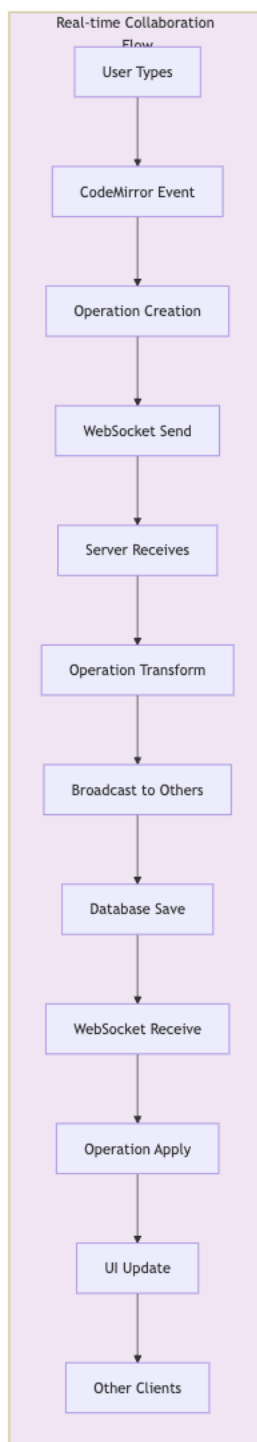
כלים ומנגנונים נוספים

- Docker : לארגון סביבת הפיתוח והריצה
- nginx : כשרת HTTP וכ-reverse proxy
- Git : לבקרת גרסאות
- pytest : לבדיקות אוטומטיות
- bcrypt : להצפנת סיסמאות
- Valgrind : לבדיקת זליגות זיכרון בקוד C

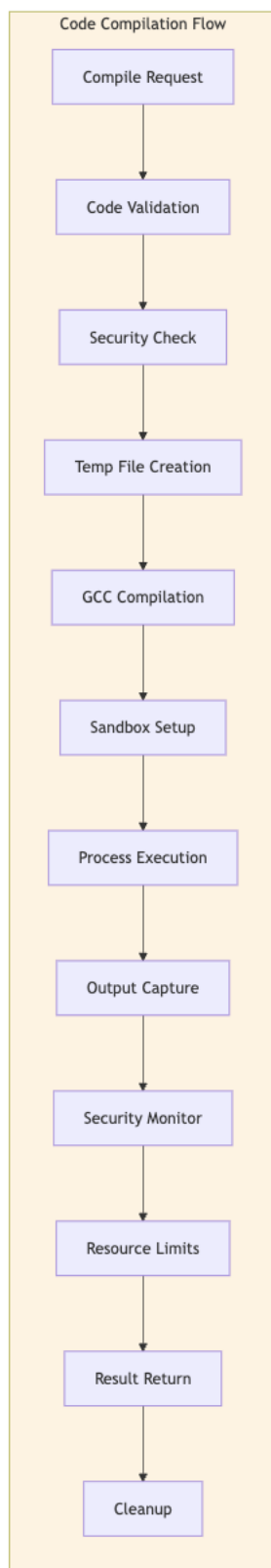
זרימת המידע במערכת הרשמה והתחברות



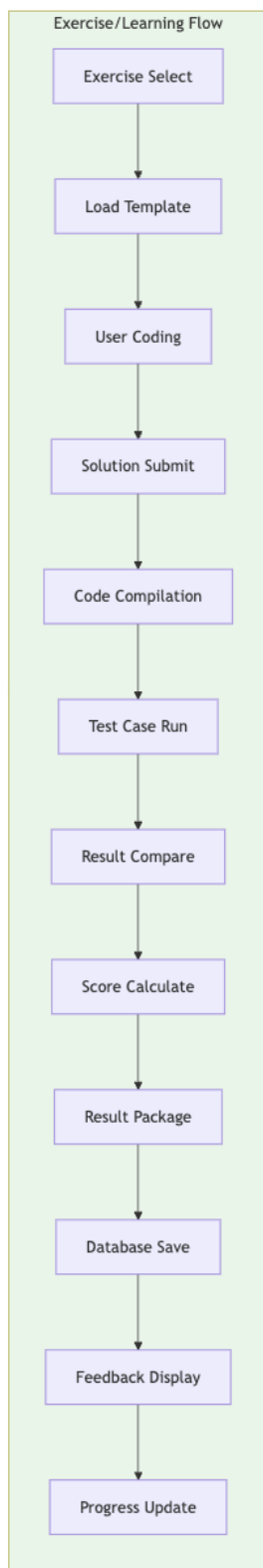
עריכת קוד



הידור והרצת קוד



עבודה על תרגיל



האלגוריתמים

עריכה שיתופית בזמן אמת

האתגר המרכזי בעריכה שיתופית הוא לשמור על סנכרון בין כל המשתמשים, תוך טיפול בשינויים מקבילים והתנגשויות. המערכת משתמשת במנגנון מבוסס על שינויים מקומיים, שנשלחים לשרת ומופצים לכל המשתמשים האחרים.

השרת מנהל מצב מרכזי של כל פרויקט, ומטפל בשינויים לפי הסדר שהם מגיעים. כאשר מתרחשת התנגשות, השרת מיישם אלגוריתם מיזוג שמנסה לשלב את השינויים של כל המשתמשים באופן הגיוני.

האלגוריתם מבוסס על שלושה סוגי פעולות עריכה:

1. **הוספה**: הוספת טקסט חדש בנקודה מסוימת
2. **מחיקה**: מחיקת טקסט קיים מנקודה מסוימת
3. **החלפה**: החלפת כל הטקסט בטקסט חדש

לכל פעולה, השרת מחשב את ההשפעה שלה על מצב הפרויקט הנוכחי, ומעדכן את המצב בהתאם. השינויים המעודכנים מופצים לכל המשתמשים, כך שכולם רואים את אותו מצב.

בדיקת קוד מסוכן

כדי להגן על המערכת מפני קוד זדוני, המערכת מיישמת אלגוריתם לבדיקת קוד מסוכן. האלגוריתם מחפש דפוסים ידועים של קוד C שעלול לסכן את המערכת, כגון:

- קריאות מערכת כמו `system()`, `popen()`
- פעולות על קבצים כמו `fopen()`, `fwrite()`, `fprintf()`
- פעולות תהליכים כמו `exec()`, `fork()`
- פעולות על מערכת הקבצים כמו `unlink()`, `remove()`, `rename()`

האלגוריתם משתמש בביטויים רגולריים כדי לזהות את הדפוסים האלה בקוד המוגש. אם נמצא דפוס מסוכן, המערכת מסרבת להריץ את הקוד ומציגה הודעה מתאימה למשתמש.

יתרונו של פתרון זה הוא שהוא פשוט יחסית ליישום ויעיל בזיהוי דפוסים ידועים. חסרונו הוא שהוא עלול לסנן גם קוד לגיטימי שמשתמש בפונקציות אלה למטרות חוקיות, ושהוא עלול להחמיץ דפוסים מסוכנים שאינם מכוסים על ידי הביטויים הרגולריים.

```
def check_for_dangerous_code(code):
    dangerous_patterns = [
        r'system\s*(\s|', # קריאות system()
        r'popen\s*(\s|', # קריאות popen()
        r'fopen\s*(\s|', # פעולות קובץ
        r'fwrite\s*(\s|', # פעולות כתיבה לקובץ
        r'fprintf\s*(\s|', # פעולות כתיבה לקובץ
        r'exec[lv][pe]\s*(\s|', # משפחת פונקציות exec
```

```

r'fork\s*\(', # יצירת תהליכים
r'unlink\s*\(', # מחיקת קבצים
r'remove\s*\(', # מחיקת קבצים
r'rename\s*\(', # שינוי שם קבצים
r'mkdir\s*\(', # יצירת תיקיות
r'rmdir\s*\(' # מחיקת תיקיות
]

```

```

import re
for pattern in dangerous_patterns:
    if re.search(pattern, code):
        return True
return False

```

הגבלת משאבים בהרצת קוד

כדי למנוע שימוש לרעה במשאבי המערכת, המערכת מיישמת אלגוריתם להגבלת משאבים בהרצת קוד. האלגוריתם משתמש במודול resource של Python כדי להגביל:

- זמן CPU מניעת לולאות אינסופיות
- שימוש בזיכרון: מניעת דליפות זיכרון ומתקפות "שיטפון זיכרון"
- גודל קובץ: מניעת יצירת קבצים גדולים

```

def set_resource_limits():
    resource.setrlimit(resource.RLIMIT_AS, (32 * 1024 * 1024, 32 * 1024 * 1024))
    resource.setrlimit(resource.RLIMIT_CPU, (2, 2))
    resource.setrlimit(resource.RLIMIT_FSIZE, (1024 * 1024, 1024 * 1024))

```

מגבלות אלה מיושמות בכל הרצת קוד, באמצעות הפרמטר `preexec_fn` של `subprocess.Popen` שמאפשר להפעיל פונקציה בתהליך הברן לפני הרצת התוכנית.

בדיקה אוטומטית של פתרונות תרגילים

המערכת מיישמת אלגוריתם לבדיקה אוטומטית של פתרונות לתרגילים. האלגוריתם מריץ את הפתרון של המשתמש עם מספר מקרי בדיקה מוגדרים מראש, ומשווה את הפלט של הפתרון עם הפלט המצופה.

התהליך כולל את השלבים הבאים:

1. הידור הקוד של המשתמש
2. הרצת הקוד עם כל מקרה בדיקה
3. השוואת הפלט בפועל עם הפלט המצופה
4. יצירת דוח תוצאות עם סטטוס כל מקרה בדיקה

מקרי הבדיקה מוגדרים בפורמט JSON שכולל:

- קלט למקרה הבדיקה
- פלט מצופה
- תיאור אופציונלי של מקרה הבדיקה

אלגוריתמים קיימים

עבור עריכה שיתופית:

- Google's Differential Synchronization
- Apache Wave's Operational Transformation
- ShareJS Implementation

עבור הרצה מאובטחת:

- Docker Containers
- LXC (Linux Containers)
- chroot jails

מקורות רלוונטיים

למה Operational Transformation ?

- **יציבות**: אלגוריתם מוכח בשימוש במערכות כמו Google Docs
- **פשטות**: יחסית פשוט ליישום לעומת אלטרנטיבות
- **ביצועים**: תגובה מהירה לשינויים

למה Linux rlimit ?

- **יעילות**: מובנה במערכת ההפעלה, אמין ומהיר
- **גמישות**: אפשרות להגדיר הגבלות מדויקות
- **אבטחה**: הגנה ברמת הליבה

סביבת הפיתוח

כלי פיתוח

- Visual Studio Code : בעבור בו לכתובת קוד Frontend, Backend ובדיקות.
- GCC (מהדר C) : לביצוע קומפילציה של קוד C ב-Backend.
- Git (בקרת גרסאות) : לניהול השינויים בקוד.
- Make (כלי בנייה) : להגדרת תהליך ההידור והבנייה של הפרויקט.

- CUnit (ספריית בדיקות C) : לבדיקות יחידה לקוד C.

פירוט הסביבה וכלי בדיקה

- macOS : מערכת ההפעלה של השרת.
- Apache/Nginx (שרת אינטרנט) : להגשת ה-Frontend
- MySQL/PostgreSQL (מסד נתונים) : לאחסון נתונים.
- Jest/Mocha (ספריות בדיקות JavaScript) : לבדיקות יחידה ואינטגרציה ל-Frontend
- Cypress/Selenium (כלי בדיקות E2E) : לבדיקות מקצה לקצה.
- Valgrind : לניתוח זיכרון.

פרוטוקול התקשורת

המערכת משתמשת בשני פרוטוקולי תקשורת עיקריים :

1. HTTP/HTTPS לבקשות סינכרוניות (REST API)
2. WebSocket לתקשורת אסינכרונית בזמן אמת

מבנה הודעות WebSocket : כל הודעה היא אובייקט JSON עם המבנה :

```
{
  "type": "message_type",
  "data": {
    "field1": "value1",
    "field2": "value2"
  },
  "timestamp": "ISO8601_timestamp",
  "user_id": "user_identifier"
}
```

פירוט כלל ההודעות

מהלקוח לשרת :

שם ההודעה	מטרה	נשלחת מ	נשלחת אל	מבנה השדות
<i>connect</i>	חיבור לשרת	Client	Server	{project_id: string}
<i>join_project</i>	הצטרפות לפרויקט	Client	Server	{project_id: string}
<i>edit</i>	פעולת עריכה	Client	Server	{type: string, position: number, text: string, project_id: string}
<i>cursor_move</i>	cursor תזוזת	Client	Server	{project_id: string, position: {line: number, ch: number}}
<i>chat_message</i>	הודעת צ'אט	Client	Server	{project_id: string, message: string}

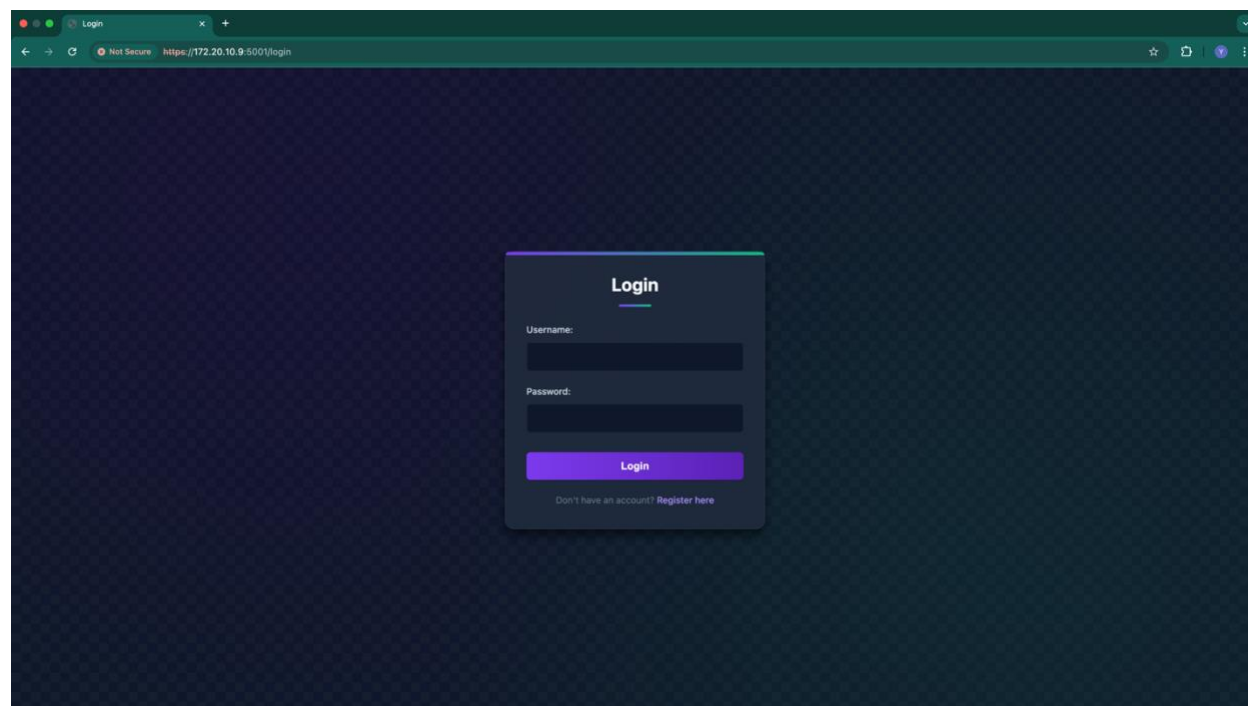
מהשרת ללקוח:

שם ההודעה	מטרה	נשלחת מ	נשלחת אל	מבנה השדות
<i>document</i>	סנכרון מסמך	Server	Client	{text: string}
<i>user_connected</i>	משתמש התחבר	Server	All Clients	{username: string, user_id: string, sid: string}
<i>user_disconnected</i>	משתמש התנתק	Server	All Clients	{username: string, sid: string}
<i>cursor_update</i>	עדכון cursor	Server	Other Clients	{user_id: string, username: string, position: object}
<i>new_chat_message</i>	הודעת צ'אט	Server	All Clients	{user_id: string, username: string, message: string, timestamp: string}
<i>edit_error</i>	שגיאת עריכה	Server	Client	{message: string}

מסכי המערכת

המערכת כוללת מספר מסכים מרכזיים:

מסך כניסה (Login)



תפקיד המסך: מתן אפשרות למשתמשים רשומים להתחבר למערכת.

מידע מוצג:

- שדה קלט לשם משתמש
- שדה קלט לסיסמה
- כפתור התחברות
- קישור למסך הרשמה
- הודעות שגיאה במידת הצורך

פעולות זמינות:

- הזנת פרטי התחברות
- לחיצה על כפתור התחברות
- מעבר למסך הרשמה

מסך הרשמה (Register)

The screenshot shows a web browser window with the title 'Register' and the URL 'https://172.20.10.9:5001/register'. The page has a dark blue background with a subtle grid pattern. In the center, there is a white registration form with the title 'Register' and a green underline. The form contains three input fields labeled 'Username:', 'Email:', and 'Password:'. Below these fields is a green 'Register' button. At the bottom of the form, there is a link that says 'Already have an account? Login here'.

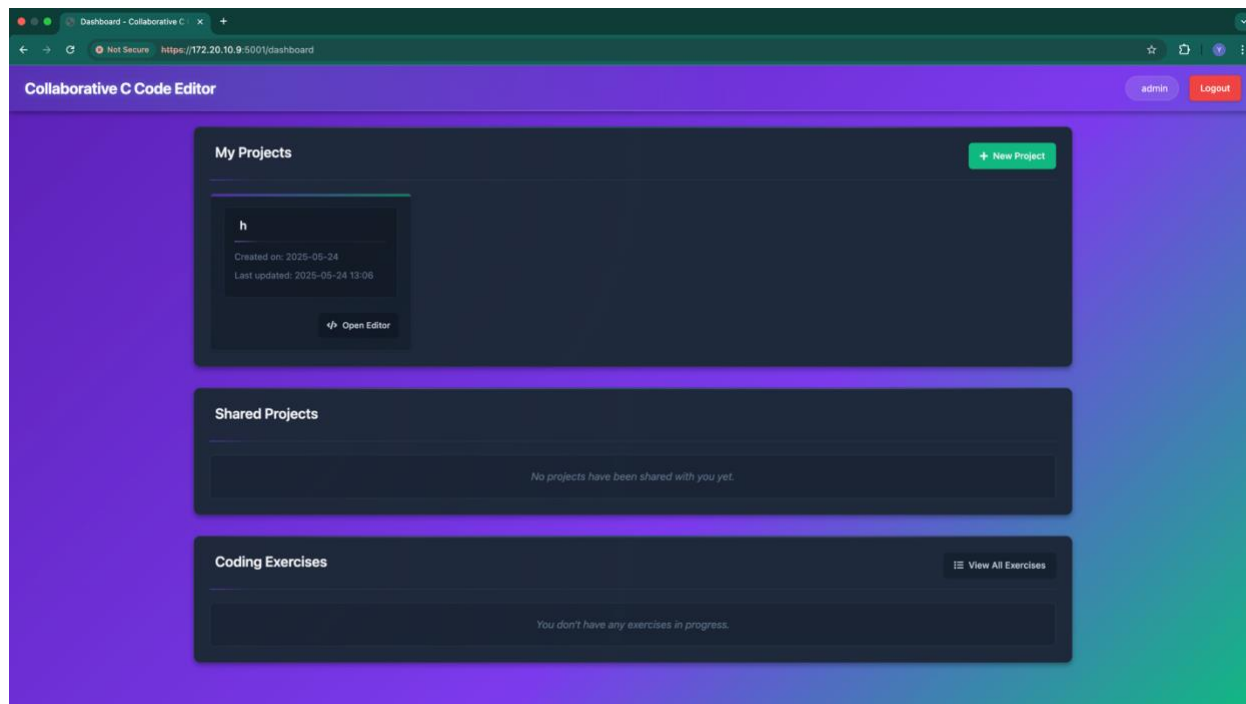
תפקיד המסך: מתן אפשרות למשתמשים חדשים להירשם למערכת.

מידע מוצג:

- שדה קלט לשם משתמש
- שדה קלט לכתובת אימייל
- שדה קלט לסיסמה
- כפתור הרשמה
- קישור למסך התחברות
- הודעות ולידציה

פעולות זמינות:

- הזנת פרטי הרשמה
- ולידציה בזמן אמת של השדות
- לחיצה על כפתור הרשמה
- מעבר למסך התחברות

לוח מחוונים (Dashboard)

תפקיד המסך: מרכז השליטה הראשי של המשתמש במערכת.

מידע מוצג:

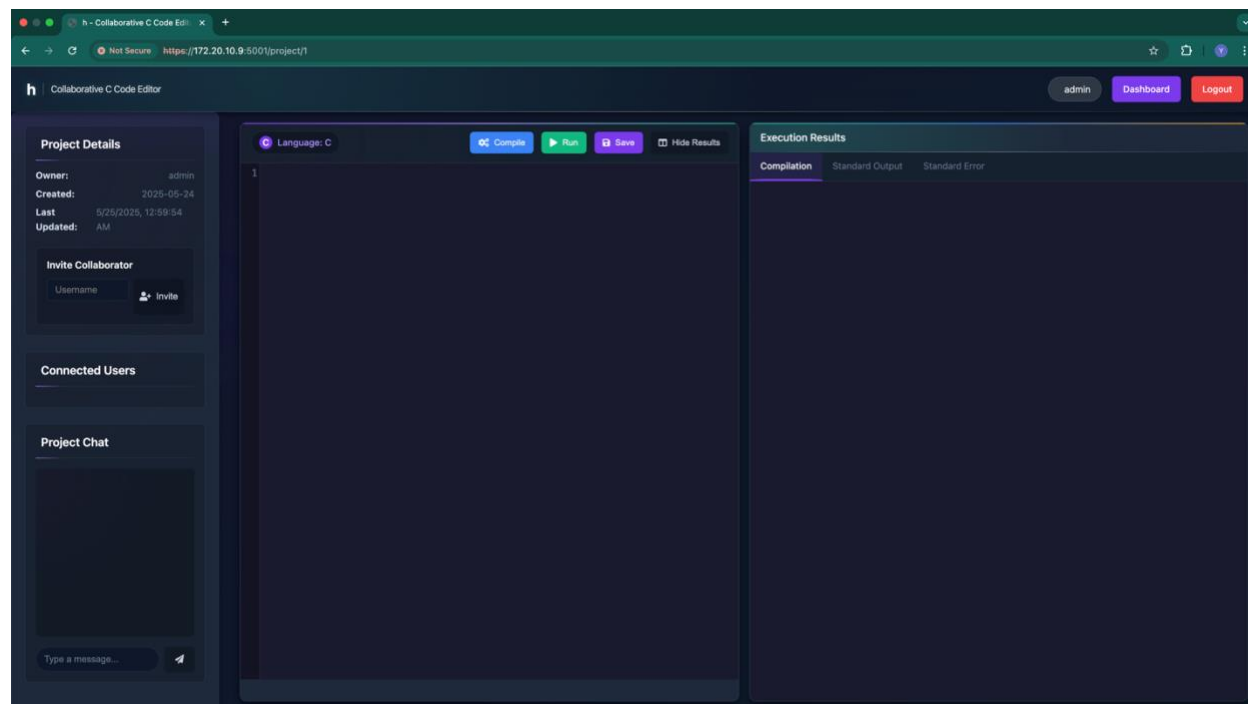
- רשימת פרויקטים בבעלות המשתמש
- רשימת פרויקטים משותפים
- רשימת תרגילים בתהליך
- סטטיסטיקות אישיות
- תפריט ניווט ראשי

פעולות זמינות:

- יצירת פרויקט חדש
- פתיחת פרויקט קיים לעריכה
- מעבר לדף תרגילים
- מעבר לפאנל ניהול

- התנתקות מהמערכת

מסך עריכת פרויקט



תפקיד המסך: הלב של המערכת - עריכה שיתופית של קוד C

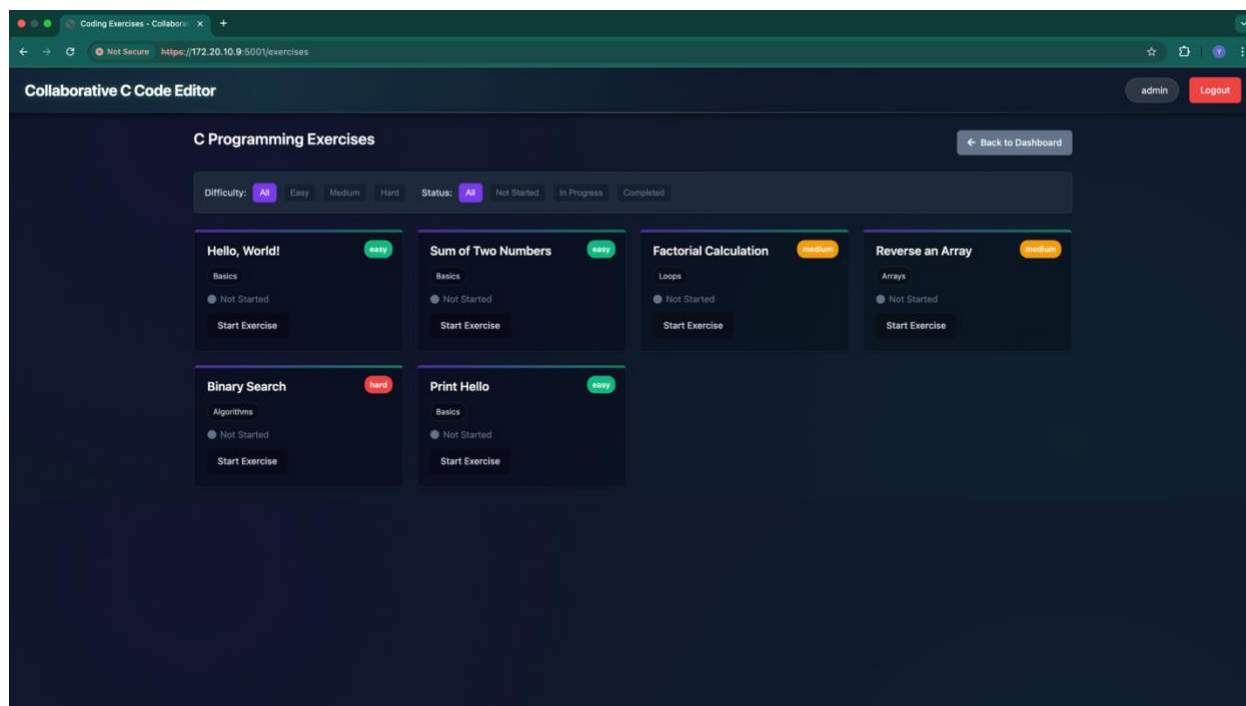
מידע מוצג:

- עורך קוד עם syntax highlighting
- רשימת משתמשים מחוברים
- אזור צ'אט לתקשורת
- פאנל תוצאות הרצה
- כפתורי פעולה (קומפילציה, הרצה, שמירה)

פעולות זמינות:

- עריכת קוד בזמן אמת
- קומפילציה של הקוד
- הרצת הקוד עם או בלי קלט
- שליחת הודעות צ'אט
- שמירת הפרויקט
- הזמנת משתתפים נוספים

מסך תרגילים



תפקיד המסך: הצגת רשימת התרגילים הזמינים במערכת.

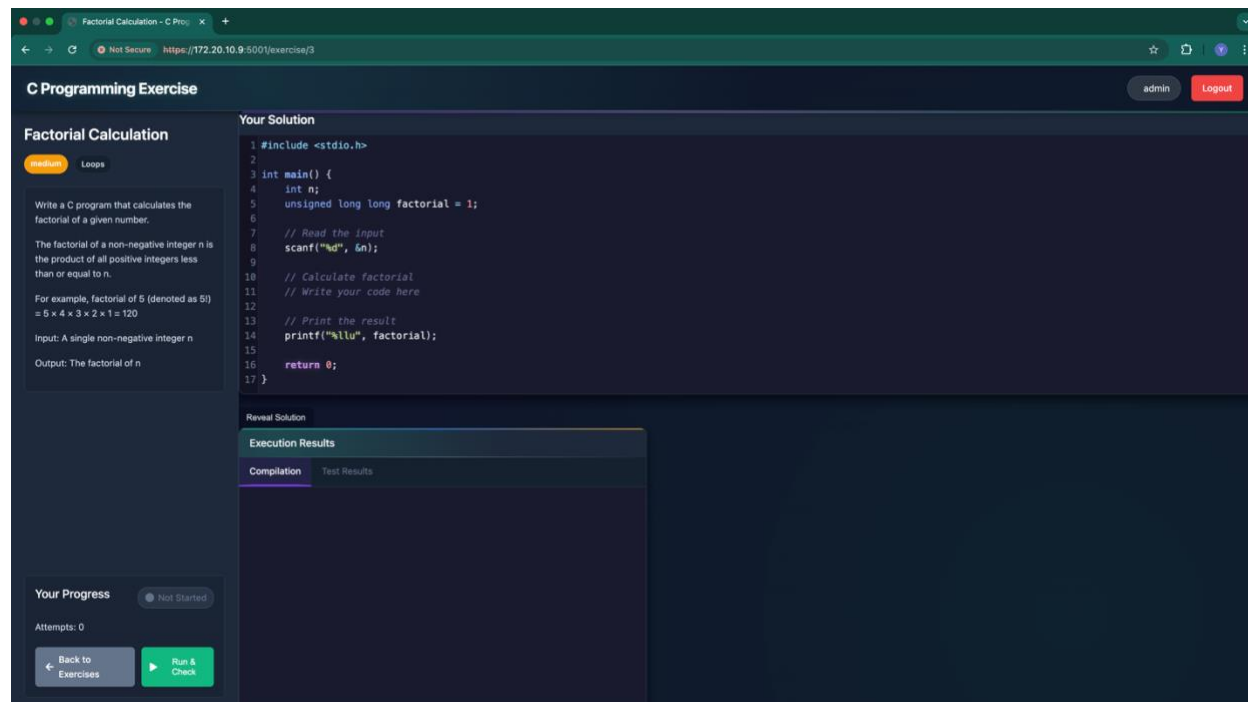
מידע מוצג:

- רשימת כל התרגילים
- מידע על כל תרגיל
- סטטוס התקדמות
- אפשרויות סינון וחיפוש
- כפתורי פעולה לכל תרגיל

פעולות זמינות:

- סינון תרגילים לפי קושי
- סינון תרגילים לפי סטטוס
- חיפוש תרגילים
- תחילת עבודה על תרגיל
- המשך עבודה על תרגיל קיים
- חזרה לדשבורד

מסך תרגיל



תפקיד המסך: פתרון תרגיל ספציפי עם בדיקה אוטומטית.

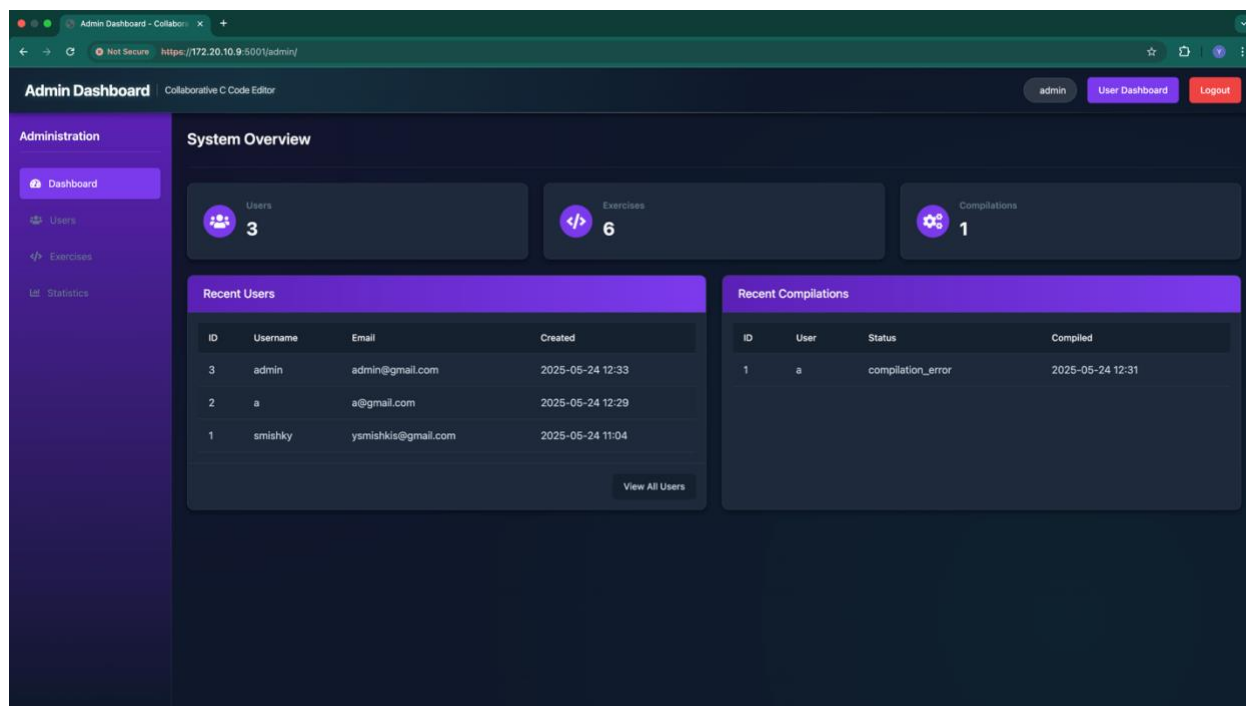
מידע מוצג:

- תיאור התרגיל ודרישותיו
- דירוג קושי וקטגוריה
- עורך קוד עם קוד התחלתי
- אזור תוצאות בדיקה
- פאנל התקדמות אישית
- פתרון

פעולות זמינות:

- עריכת קוד הפתרון
- הרצה ובדיקה של הפתרון
- הצגת פתרון מלא
- חזרה לרשימת תרגילים
- מעבר לתרגיל הבא/קודם

מסך ניהול



תפקיד המסך: כלי ניהול למנהלי המערכת.

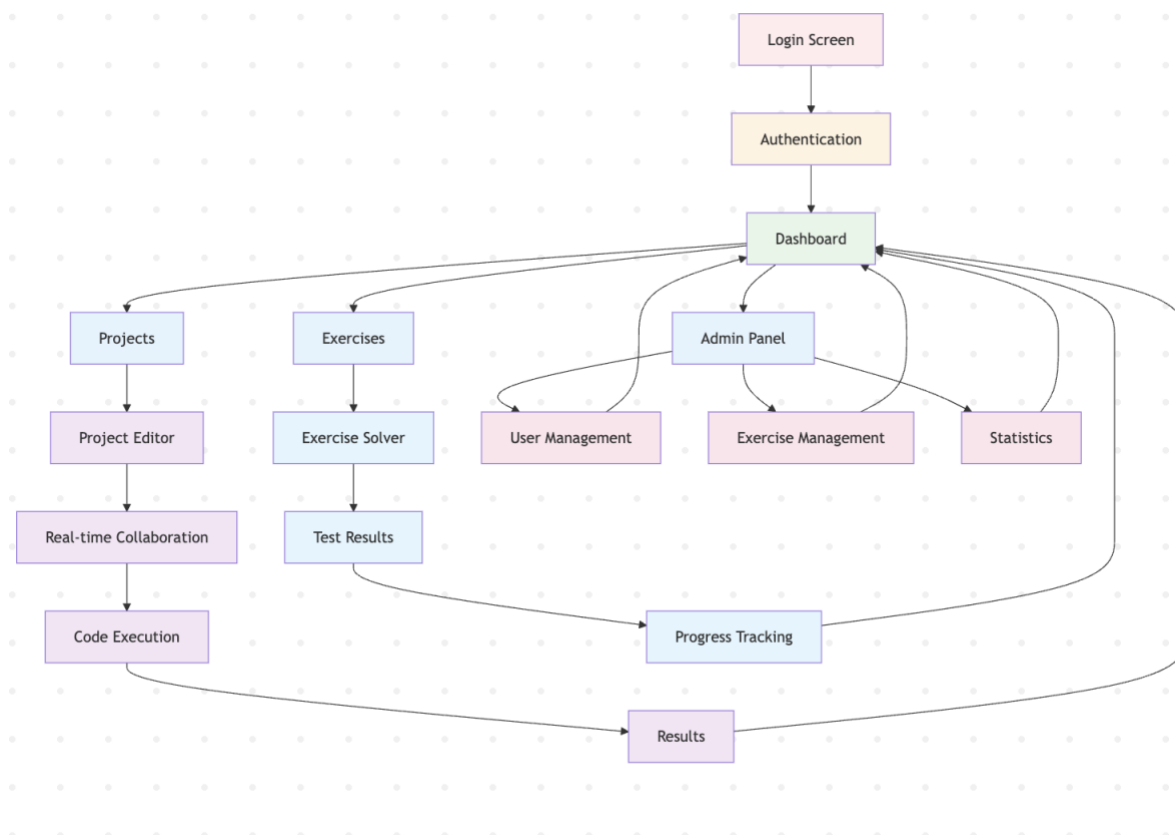
מידע מוצג:

- סטטיסטיקות מערכת כלליות
- רשימת משתמשים
- רשימת תרגילים
- גרפים ודוחות
- לוגים ומעקב פעילות

פעולות זמינות:

- ניהול משתמשים
- יצירה ועריכה של תרגילים
- צפייה בסטטיסטיקות מפורטות
- ניהול תוכן ומודרציה
- ייצוא נתונים ודוחות של תרגילים

תרשים מסכים



מבני הנתונים

פירוט מבני הנתונים

המערכת משתמשת במספר מבני נתונים מרכזיים, המיוצגים כמודלים ב-SQLAlchemy-

User

מייצג משתמש במערכת.

שדה	סוג	תיאור
<i>id</i>	Integer	מזהה ייחודי אוטומטי
<i>username</i>	String(20)	שם משתמש, ייחודי
<i>email</i>	String(120)	כתובת דוא"ל, ייחודית
<i>password</i>	String(60)	סיסמה מוצפנת
<i>created_at</i>	DateTime	תאריך יצירת המשתמש

קשרים:

- **owned_projects** פרויקטים שהמשתמש יצר
- **collaborated_projects** פרויקטים שהמשתמש משתתף בהם
- **exercise_progress** התקדמות המשתמש בתרגילים

Project

מייצג פרויקט במערכת.

שדה	סוג	תיאור
<i>id</i>	Integer	מזהה ייחודי אוטומטי
<i>name</i>	String(100)	שם הפרויקט
<i>content</i>	Text	תוכן הקוד של הפרויקט
<i>created_at</i>	DateTime	תאריך יצירת הפרויקט
<i>updated_at</i>	DateTime	תאריך עדכון אחרון
<i>owner_id</i>	Integer (FK)	מזהה הבעלים של הפרויקט

קשרים:

- **owner** המשתמש שיצר את הפרויקט
- **Collaborators** משתמשים המשתתפים בפרויקט
- **Documents** מסמכים השייכים לפרויקט

Document

מייצג מסמך בפרויקט.

שדה	סוג	תיאור
<i>id</i>	Integer	מזהה ייחודי אוטומטי
<i>name</i>	String(100)	שם המסמך
<i>content</i>	Text	תוכן המסמך
<i>project_id</i>	Integer (FK)	מזהה הפרויקט
<i>user_id</i>	Integer (FK)	מזהה המשתמש שיצר את המסמך
<i>created_at</i>	DateTime	תאריך יצירת המסמך
<i>updated_at</i>	DateTime	תאריך עדכון אחרון

קשרים:

- **Project** הפרויקט שהמסמך שייך אליו

Exercise

מייצג תרגיל תכנות.

שדה	סוג	תיאור
<i>id</i>	Integer	מזהה ייחודי אוטומטי

כותרת התרגיל	String(200)	<i>title</i>
תיאור התרגיל	Text	<i>description</i>
רמת קושי (easy, medium, hard)	String(20)	<i>difficulty</i>
קטגוריה	String(50)	<i>category</i>
קוד התחלתי	Text	<i>initial_code</i>
קוד פתרון	Text	<i>solution_code</i>
מקרי בדיקה בפורמט JSON	Text	<i>test_cases</i>

קשרים:

- **Progress** התקדמות משתמשים בתרגיל

ExerciseProgress

מייצג התקדמות של משתמש בתרגיל.

שדה	סוג	תיאור
<i>id</i>	Integer	מזהה ייחודי אוטומטי
<i>user_id</i>	Integer (FK)	מזהה המשתמש
<i>exercise_id</i>	Integer (FK)	מזהה התרגיל
<i>status</i>	String(20)	סטטוס (not_started, in_progress, completed)
<i>user_code</i>	Text	הקוד של המשתמש
<i>attempts</i>	Integer	מספר ניסיונות
<i>last_attempt</i>	DateTime	זמן הניסיון האחרון
<i>completed_at</i>	DateTime	זמן השלמת התרגיל

קשרים:

- **User** המשתמש
- **Exercise** התרגיל

CompilationHistory

מייצג היסטוריית קומפילציה והרצה.

שדה	סוג	תיאור
<i>id</i>	Integer	מזהה ייחודי אוטומטי
<i>user_id</i>	Integer (FK)	מזהה המשתמש
<i>project_id</i>	Integer (FK)	מזהה הפרויקט
<i>document_id</i>	Integer (FK)	מזהה המסמך
<i>exercise_id</i>	Integer (FK)	מזהה התרגיל
<i>code</i>	Text	הקוד שהודר/הורץ
<i>compiled_at</i>	DateTime	זמן הקומפילציה
<i>compilation_output</i>	Text	פלט הקומפילציה
<i>execution_output</i>	Text	פלט ההרצה

(success, סטטוס, compilation_error, runtime_error) String(20) | status

ChatMessage

מייצג הודעת צ'אט.

שדה	סוג	תיאור
<i>id</i>	Integer	מזהה ייחודי אוטומטי
<i>user_id</i>	Integer (FK)	מזהה המשתמש
<i>project_id</i>	Integer (FK)	מזהה הפרויקט
<i>message</i>	Text	תוכן ההודעה
<i>sent_at</i>	DateTime	זמן שליחת ההודעה

קשרים:

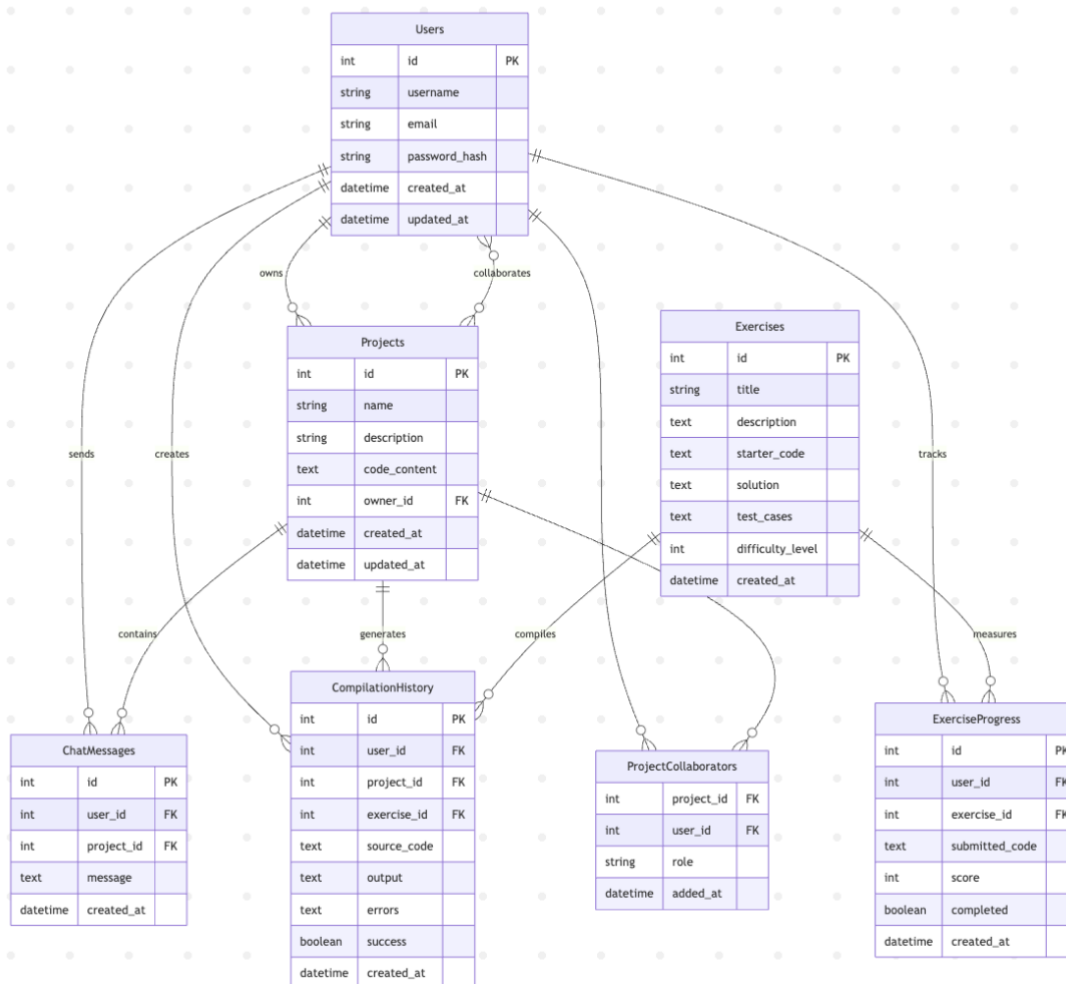
- User המשתמש ששלח את ההודעה
- Project הפרויקט שההודעה שייכת אליו

פירוט מאגרי המידע

המערכת משתמשת במסד נתונים SQL, מיושם באמצעות מנגנון ORM של SQLAlchemy מסד הנתונים כולל את הטבלאות הבאות:

1. **user** שומרת מידע על משתמשי המערכת
2. **project** שומרת מידע על פרויקטים
3. **project_collaborators** טבלת קשר בין משתמשים לפרויקטים (many-to-many)
4. **document** שומרת מידע על מסמכים בפרויקטים
5. **exercise** שומרת מידע על תרגילי תכנות
6. **exercise_progress** שומרת מידע על התקדמות משתמשים בתרגילים
7. **compilation_history** שומרת מידע על ניסיונות קומפילציה והרצה
8. **chat_message** שומרת מידע על הודעות צ'אט

מסדי נתונים



אינדקסים לביצועים:

- users.username : Username
- users.email : Email
- projects.owner_id : Owner
- exercise_progress(user_id, exercise_id) : User-Exercise
- compilation_history.compiled_at : Compilation-Time

סקירת חולשות והאיומים

המערכת חשופה למספר איומים אבטחה פוטנציאליים, והיא מיישמת מנגנוני הגנה כדי להתמודד איתם:

שכבת האפליקציה

מנגנון הגנה	תיאור	איום
SQLAlchemy ORM שמנקה קלט אוטומטית	הזרקת קוד SQL זדוני לשאילתות	<i>SQL Injection</i>
ניקוי קלט, הגבלת תגיות HTML מותרות	הזרקת קוד JavaScript זדוני	<i>Cross-Site Scripting (XSS)</i>
CSRF token	ביצוע פעולות בשם משתמש מאומת	<i>Cross-Site Request Forgery (CSRF)</i>
הצפנת סיסמאות עם bcrypt מגבלות על ניסיונות כניסה	ניסיונות לנחש סיסמאות או לעקוף את מערכת האימות	<i>התקפות על מערכת האימות</i>
סינון קוד מסוכן, סביבת הרצה מבודדת, הגבלת משאבים	הרצת קוד C שמנסה לפגוע במערכת	<i>הרצת קוד זדוני</i>
הגבלת קצב בקשות, הגבלת משאבים לכל משתמש	התקפות מניעת שירות	<i>DOS/DDOS</i>

שכבת התעבורה

מנגנון הגנה	תיאור	איום
שימוש ב-TLS/SSL להצפנת תקשורת	יירוט תקשורת בין לקוח לשרת	<i>Man-in-the-Middle</i>
שימוש בטוקנים מוצפנים, תפוגת סשן	גניבת זהות סשן	<i>Session Hijacking</i>
הצפנת כל התקשורת באמצעות TLS/SSL	האזנה לתעבורת רשת	<i>Packet Sniffing</i>

שכבת המערכת

מנגנון הגנה	תיאור	איום
סביבת הרצה מבודדת, הגבלת משאבים, סינון קוד מסוכן	הרצת קוד שמנסה לגשת למערכת הקבצים או למשאבים אחרים	<i>הרצת קוד זדוני</i>
הרצת קוד בהרשאות מינימליות, סביבת הרצה מבודדת	ניסיון להשיג הרשאות גבוהות יותר	<i>Privilege Escalation</i>
הגבלת זמן CPU, זיכרון וגודל קובץ לכל הרצת קוד	ניסיון לצרוך יותר מדי משאבי מערכת	<i>שימוש יתר במשאבים</i>

המערכת מיישמת שכבות הגנה מרובות כדי להתמודד עם איומים אלה. היא משתמשת בכלים ושיטות סטנדרטיים ומוכחים, כמו Flask-Bcrypt, SQLAlchemy וגם TLS/SSL ומיישמת מנגנוני אבטחה ייחודיים כמו סינון קוד מסוכן וסביבת הרצה מבודדת.

מימוש הפרויקט

חלק א' - סקירת המודולים והמחלקות של המערכת

מודולים מיובאים

שם המודול	תפקיד המודול
<i>Flask</i>	פריימוורק לפיתוח אפליקציית ווב בשפת Python
<i>Flask-SocketIO</i>	הרחבה של Flask המאפשרת תקשורת בזמן אמת באמצעות WebSockets
<i>SQLAlchemy</i>	ORM (Object Relational Mapper) להתממשקות עם מסד נתונים
<i>Bcrypt</i>	ספריה להצפנת סיסמאות
<i>JSON</i>	ספריה לטיפול בפורמט JSON
<i>Subprocess</i>	ספריה להרצת פקודות מערכת הפעלה
<i>Resource</i>	ספריה לניהול משאבי מערכת הפעלה

מחלקות שפותחו

User

תפקיד המחלקה: ניהול פרטי משתמש במערכת

תכונות המחלקה:

- id : מזהה ייחודי למשתמש
- Username : שם המשתמש
- Email : כתובת הדוא"ל של המשתמש
- password : סיסמה מוצפנת
- created_at : תאריך יצירת החשבון
- owned_projects : קשר לפרויקטים שהמשתמש יצר
- collaborated_projects : קשר לפרויקטים שהמשתמש משתתף בהם
- exercise_progress : קשר להתקדמות המשתמש בתרגילים

פעולות במחלקה:

- `__repr__` : מחזיר מחרוזת המייצגת את המשתמש

Project

תפקיד המחלקה: שמירה וניהול של פרויקטי קוד

תכונות המחלקה:

- Id : מזהה ייחודי לפרויקט
- Name : שם הפרויקט
- Content : תוכן הפרויקט
- created_at : תאריך יצירת הפרויקט

- updated_at : תאריך עדכון אחרון של הפרויקט
- owner_id : מזהה המשתמש שיצר את הפרויקט

פעולות במחלקה:

- __repr__ : מחזיר מחרוזת המייצגת את הפרויקט

Exercise

תפקיד המחלקה: ניהול תרגילי קוד במערכת

תכונות המחלקה:

- Id : מזהה ייחודי לתרגיל
- Title : כותרת התרגיל
- Description : תיאור התרגיל
- difficulty : רמת קושי
- category : קטגוריה
- initial_code : קוד התחלתי לתרגיל
- solution_code : פתרון לתרגיל
- test_cases : מקרי בדיקה
- progress : קשר להתקדמות משתמשים בתרגיל

פעולות במחלקה:

- __repr__ : מחזיר מחרוזת המייצגת את התרגיל

ExerciseProgress

תפקיד המחלקה: מעקב אחר התקדמות משתמשים בתרגילים

תכונות המחלקה:

- Id : מזהה ייחודי
- user_id: מזהה המשתמש
- exercise_id : מזהה התרגיל
- status : סטטוס
- user_code : הקוד שהמשתמש כתב
- attempts : מספר הניסיונות
- last_attempt : תאריך הניסיון האחרון
- completed_at : תאריך השלמת התרגיל

פעולות במחלקה:

- __repr__ : מחזיר מחרוזת המייצגת את ההתקדמות

CompilationHistory

תפקיד המחלקה: שמירת היסטוריית קומפילציה והרצת קוד

תכונות המחלקה:

- Id : מזהה ייחודי
- user_id : מזהה המשתמש
- project_id : מזהה הפרויקט
- document_id : מזהה המסמך
- exercise_id : מזהה התרגיל
- code : הקוד שהורץ
- compiled_at : תאריך הקומפילציה
- compilation_output : פלט הקומפילציה
- execution_output : פלט ההרצה
- status : סטטוס

פעולות במחלקה:

- __repr__ : מחזיר מחרוזת המייצגת את ההיסטוריה

ChatMessage

תפקיד המחלקה: ניהול הודעות צ'אט בין משתמשים בפרויקט

תכונות המחלקה:

- Id : מזהה ייחודי להודעה
- user_id : מזהה המשתמש ששלח את ההודעה
- project_id : מזהה הפרויקט
- message : תוכן ההודעה
- sent_at : זמן שליחת ההודעה
- User : קשר למשתמש ששלח את ההודעה
- Project : קשר לפרויקט

פעולות במחלקה:

- __repr__ : מחזיר מחרוזת המייצגת את ההודעה

AdminBlueprint

תפקיד המחלקה: ניהול דפי האדמין

תכונות המחלקה:

- `admin_required` : דקורטור לבדיקת הרשאות אדמין

פעולות במחלקה:

- `check_admin()` : בדיקה האם המשתמש הוא אדמין
- `index()` : דף הבית של האדמין
- `users()` : ניהול משתמשים
- `exercises()` : ניהול תרגילים
- `stats()` : הצגת סטטיסטיקות

פונקציות עזר מרכזיות:

`execute_code_impl()`

- **תפקיד:** הרצה מאובטחת של קוד C
- **פרמטרים:** `code, project_id, user_input, compile_only`
- **החזרה:** תוצאות קומפילציה והרצה
- **תכונות מיוחדות:** `sandboxing, resource limiting, security checks`

`check_for_dangerous_code()`

- **תפקיד:** בדיקת קוד לתבניות מסוכנות
- **פרמטרים:** `code (string)`
- **החזרה:** True אם נמצא קוד מסוכן
- **תכונות מיוחדות:** `regex patterns` לזיהוי פונקציות מערכת

`set_resource_limits()`

- **תפקיד:** הגדרת הגבלות משאבים לתהליך
- **פרמטרים:** ללא
- **החזרה:** ללא
- **תכונות מיוחדות:** הגבלת זיכרון CPU, וקבצים

חלק ב' - אלגוריתמים מרכזיים בפרויקט

אלגוריתם הרצת קוד C מאובטח

הסבר על היכולת: אלגוריתם זה מאפשר למשתמש להריץ קוד C בסביבה מאובטחת תוך הגנה על המערכת מפני קוד זדוני. האלגוריתם כולל שלושה שלבים עיקריים:

1. בדיקת אבטחה של הקוד
2. קומפילציה של הקוד

3. הרצת הקוד בסביבה מבודדת עם הגבלת משאבים

תיאור מפורט של האלגוריתם :

1. המערכת יוצרת תיקייה זמנית לאחסון קבצי הקוד והקובץ הבינארי המקומפל
2. הקוד נכתב לקובץ זמני
3. מתבצעת בדיקת אבטחה לאיתור תבניות קוד מסוכנות למשל `system()`, `popen()` וכו'
4. אם מתגלות תבניות מסוכנות, התהליך נעצר ומוחזרת הודעת שגיאה
5. הקוד מקומפל באמצעות GCC עם דגלים ספציפיים להגדרת סטנדרט, ספריות וכו'
6. אם יש שגיאות קומפילציה, הן מוחזרות למשתמש
7. לפני הרצת הקוד מוגדרות מגבלות משאבים באמצעות `resource.setrlimit`:
 - הגבלת זיכרון ל-MB32
 - הגבלת זמן CPU ל-2 שניות
 - הגבלת גודל קבצים ל-MB1
8. הקוד מורץ בתהליך נפרד עם קלט מהמשתמש (אם יש)
9. פלט ההרצה נאסף ומוחזר למשתמש
10. התיקייה הזמנית נמחקת

קוד זה מיישם עקרונות של "סביבת חול (Sandbox)" להרצת קוד לא אמין תוך הגנה על המערכת המארחת.

אלגוריתם Operational Transformation לעריכה שיתופית

הסבר על היכולת: אלגוריתם זה מנהל עריכת קוד שיתופית בזמן אמת בין מספר משתמשים. הוא מאפשר סנכרון תוכן העורך בין משתמשים שונים ופתרון התנגשויות.

תיאור מפורט של האלגוריתם :

1. כל שינוי בעורך הטקסט מייצר פעולת עריכה (operation) מסוג :
 - הוספה: (insert) הוספת טקסט במיקום מסוים
 - מחיקה: (delete) מחיקת טקסט ממיקום מסוים
 - החלפה: (replace) החלפת כל התוכן
2. פעולת העריכה נשלחת לשרת עם המידע הבא :
 - סוג הפעולה
 - מיקום בטקסט
 - הטקסט הרלוונטי
 - מזהה הפרויקט
3. השרת מקבל את הפעולה ומבצע את השינוי במסמך המשותף
4. השרת משדר את המסמך המעודכן לכל המשתמשים החוברים לאותו הפרויקט
5. לקוחות מקבלים את המסמך המעודכן ומעדכנים את העורך שלהם
6. כדי למנוע "עריכות מרוצות" (race conditions), השרת מבצע את העריכות בסדר קבלתן

אלגוריתם זה מבטיח שכל המשתמשים רואים את אותו תוכן בכל רגע נתון, גם כאשר מספר משתמשים עורכים את המסמך במקביל.

אלגוריתם בדיקת תרגילי קוד אוטומטית

הסבר על היכולת: אלגוריתם זה מאפשר בדיקה אוטומטית של פתרונות משתמשים לתרגילי קוד באמצעות מקרי בדיקה מוגדרים מראש.

תיאור מפורט של האלגוריתם:

1. המערכת מקבלת את הקוד שכתב המשתמש ואת רשימת מקרי הבדיקה
 2. הקוד נבדק תחילה מבחינת אבטחה להגנה מפני קוד זדוני
 3. הקוד מקומפל עם אותן הגדרות כמו באלגוריתם הרצת הקוד
 4. אם הקומפילציה נכשלת, מוחזרת שגיאת הקומפילציה
 5. עבור כל מקרה בדיקה:
 - קוד המשתמש מורץ עם הקלט המוגדר במקרה הבדיקה
 - הפלט מהריצה משווה לפלט המצופה
 - נשמרת תוצאת הבדיקה (עבר/נכשל/תקוע)
 6. המערכת מחזירה סיכום של כל מקרי הבדיקה, כולל מידע על מה עבר ומה נכשל
 7. אם כל מקרי הבדיקה עברו בהצלחה, התרגיל מסומן כ"הושלם"
- האלגוריתם מאפשר משוב מיידי למשתמשים על פתרונותיהם ומאפשר למידה אינטראקטיבית.

חלק ג' - מסמך בדיקות

בדיקות שתוכננו בשלב האפיון

בדיקת עריכת קוד ותצוגת תחביר

מטרת הבדיקה: לוודא שהעורך מאפשר הזנת קוד C ומציג את התחביר בצורה נכונה.

מה בוצע בפועל: הזנת קטעי קוד שונים ובדיקת תצוגת הצבעים והפורמט.

תוצאות הבדיקה: העורך מציג כראוי את התחביר באמצעות ספריית CodeMirror מילות מפתח, משתנים, מחרוזות ופונקציות מוצגים בצבעים שונים לשיפור הקריאות.

בעיות שהתגלו ופתרון: לא התגלו בעיות משמעותיות.

בדיקת קומפילציה והרצת קוד

מטרת הבדיקה: לוודא שהמערכת מסוגלת להדר ולהריץ קוד C תקין ולהציג הודעות שגיאה מתאימות עבור קוד לא תקין.

מה בוצע בפועל:

- נכתבו והורצו תוכניות C פשוטות (כגון Hello World)
- נכתבו והורצו תוכניות עם לולאות ותנאים
- נכתבו והורצו תוכניות עם שגיאות תחביר
- נכתבו והורצו תוכניות עם שגיאות זמן ריצה

תוצאות הבדיקה :

- קוד תקין הודר והורץ בהצלחה
- המערכת הציגה הודעות שגיאה ברורות לקוד לא תקין
- המערכת זיהתה ותיעדה שגיאות זמן ריצה

בעיות שהתגלו ופתרון :

- בעיות ראשוניות בפרמטרים של GCC - נפתרו על ידי הוספת דגלים מתאימים ל-GCC
- בעיות בזיהוי ספריות - נפתרו על ידי הוספת נתיבי include מפורשים
- תקיעות בתוכניות אינסופיות - נפתרו על ידי הוספת הגבלת זמן ריצה

בדיקת עריכה שיתופית

מטרת הבדיקה : לוודא שמספר משתמשים יכולים לערוך את אותו הקוד במקביל ולראות את השינויים בזמן אמת.

מה בוצע בפועל : חיבור של מספר משתמשים לאותו פרויקט ובדיקת סנכרון השינויים.

תוצאות הבדיקה : המשתמשים יכלו לראות את השינויים של אחד השני בזמן אמת. מיקום הסמן של כל משתמש הוצג למשתמשים אחרים.

בעיות שהתגלו ופתרון :

- ניתוקים מזדמנים בחיבור - WebSocket נפתרו על ידי הוספת מנגנון התחברות מחדש אוטומטי
- עיכובים בסנכרון בעת עומס - נפתרו על ידי ייעול העברת הנתונים

בדיקת פתרון התנגשויות

מטרת הבדיקה : לוודא שהמערכת מסוגלת לטפל נכון בעריכה מקבילית של אותו קטע קוד.

מה בוצע בפועל : יצירת התנגשויות מכוונות על ידי עריכה מקבילית של אותו קטע קוד על ידי שני משתמשים שונים.

תוצאות הבדיקה : המערכת הצליחה לפתור התנגשויות על ידי החלת השינויים לפי סדר הגעתם לשרת.

בעיות שהתגלו ופתרון :

- בעיות בסדר החלת שינויים - נפתרו על ידי שיפור אלגוריתם ה-OT
- איבוד שינויים בתנאי תזמון מסוימים - נפתרו על ידי שיפור מנגנון הסנכרון

בדיקת אזור תרגול תכנות C

מטרת הבדיקה : לוודא שהמשתמשים יכולים לגשת לתרגילים, להגיש פתרונות ולקבל משוב.

מה בוצע בפועל : פתרון מספר תרגילים ובדיקת המשוב והדירוג.

תוצאות הבדיקה: המשתמשים הצליחו לגשת לתרגילים, להגיש פתרונות ולקבל משוב אוטומטי על פתרונותיהם.

בעיות שהתגלו ופתרון:

- בעיות במקרי בדיקה מורכבים - נפתרו על ידי שיפור מנגנון השוואת הפלט
- בעיות בתצוגת התקדמות המשתמש - נפתרו על ידי עדכון ממשק המשתמש

בדיקות אבטחה

בדיקת הצפנת תקשורת

מטרת הבדיקה: לוודא שהתקשורת בין הלקוח לשרת מוצפנת באמצעות TLS/SSL.

מה בוצע בפועל: מעקב אחר תעבורת הרשת ווידוא שהיא מוצפנת.

תוצאות הבדיקה: כל התקשורת בין הלקוח לשרת מוצפנת כנדרש באמצעות TLS/SSL.

בעיות שהתגלו ופתרון:

- בעיות בתצורת SSL - נפתרו על ידי עדכון תצורת השרת והתעודות

בדיקת אימות משתמשים

מטרת הבדיקה: לוודא שתהליכי הרשמה, כניסה ואיפוס סיסמה פועלים כראוי.

מה בוצע בפועל:

- ניסיון כניסה עם פרטים נכונים ושגויים
- בדיקת תהליך הרשמה
- בדיקת חוזק הסיסמאות

תוצאות הבדיקה: מערכת האימות פועלת כראוי, סיסמאות מאוחסנות בצורה מוצפנת ומאובטחת.

בעיות שהתגלו ופתרון:

- חולשה בתהליך איפוס סיסמה - נפתרה על ידי הוספת שכבת אימות נוספת
- אפשרות ליצור סיסמאות חלשות - נפתרה על ידי הוספת בדיקת חוזק סיסמה

בדיקת הרשאות גישה

מטרת הבדיקה: לוודא שמשתמשים יכולים לגשת רק לפרויקטים שיש להם הרשאה אליהם.

מה בוצע בפועל: ניסיון גישה למשאבים ללא הרשאה מתאימה.

תוצאות הבדיקה: המערכת מונעת גישה לפרויקטים שלמשתמש אין הרשאה אליהם.

בעיות שהתגלו ופתרון:

- מספר נקודות גישה ללא אכיפת הרשאות - נפתרו על ידי הוספת בדיקות הרשאה בכל הנקודות הקריטיות

בדיקות ביצועים*בדיקת עומסים*

מטרת הבדיקה: לוודא שהמערכת מתפקדת תחת עומס של משתמשים רבים.

מה בוצע בפועל: סימולציה של פעילות מרובת משתמשים ומדידת זמני תגובה.

תוצאות הבדיקה: המערכת תפקדה באופן סביר עד 100 משתמשים במקביל.

בעיות שהתגלו ופתרון:

- עומס על בסיס הנתונים - נפתר על ידי הוספת cache
- עומס בעת קומפילציה מרובה - נפתר על ידי הגבלת כמות הקומפילציות במקביל

2.בדיקת זמני תגובה

מטרת הבדיקה: לוודא שעדכונים בעריכה השיתופית מופצים במהירות.

מה בוצע בפועל: מדידת הזמן בין ביצוע שינוי להופעתו אצל משתמשים אחרים.

תוצאות הבדיקה: זמן התגובה הממוצע היה פחות מ-200 מילישניות בתנאים רגילים.

בעיות שהתגלו ופתרון:

- עיכובים ברשתות איטיות - נפתרו באמצעות אופטימיזציה של גודל ההודעות המועברות

בדיקות נוספות שבוצעו*בדיקת מגוון דפדפנים*

מטרת הבדיקה: לוודא שהמערכת פועלת באופן תקין במגוון דפדפנים.

מה בוצע בפועל: בדיקת המערכת בדפדפנים Chrome, Firefox, Safari ו-Edge.

תוצאות הבדיקה: המערכת תפקדה כראוי בכל הדפדפנים, עם הבדלים מינוריים בתצוגה.

בעיות שהתגלו ופתרון:

- בעיות תאימות ב - Safari - נפתרו על ידי התאמת ה-CSS והתסריטים

בדיקת תגובתיות במכשירים ניידים

מטרת הבדיקה: לוודא שהממשק מתאים למכשירים ניידים ומסכים קטנים.

מה בוצע בפועל: בדיקת המערכת במגוון מכשירים ורזולוציות מסך.

תוצאות הבדיקה: הממשק הותאם למכשירים ניידים באמצעות CSS רספונסיבי.

בעיות שהתגלו ופתרון:

- בעיות בתצוגת עורך הקוד במסכים קטנים - נפתרו על ידי התאמת העיצוב והוספת אפשרות לגלילה

בדיקת שימוש בזיכרון ו-CPU

מטרת הבדיקה: לוודא שהמערכת לא צורכת יותר מדי משאבים.

מה בוצע בפועל: מדידת צריכת זיכרון ו-CPU במצבי שימוש שונים.

תוצאות הבדיקה: צריכת המשאבים הייתה בגבולות הסבירים, עם עליות צפויות בעת קומפילציה.

בעיות שהתגלו ופתרון:

- דליפות זיכרון בתהליכי קומפילציה - נפתרו על ידי שיפור ניהול המשאבים

מדריך למשתמש

פירוט קבצי המערכת - עץ קבצים

```

VCCE/
|
├── server.py          # קובץ השרת הראשי
├── models.py          # הגדרות מודלים של מסד הנתונים
├── admin.py           # Blueprint לפאנל האדמין
├── exercise_manager.py # ניהול תרגילים ופונקציות עזר
├── requirements.txt    # dependencies רשימת
├── docker-compose.yml  # הגדרות Docker
├── Dockerfile          # הגדרות container לבניית
├── .gitignore          # Git-קבצים להתעלמות ב
├── README.md           # תיעוד בסיסי
├── cert.pem            # SSL תעודת
├── key.pem             # פרטי SSL מפתח
|
├── static/             # קבצים סטטיים
|   ├── style.css       # עיצוב ראשי של הממשק
|   └── script.js        # JavaScript לקוח
|
├── templates/          # HTML תבניות
|   ├── index.html      # (דף הבית) עורך קוד
|   ├── login.html      # דף התחברות
|   ├── register.html   # דף הרשמה
|   ├── dashboard.html  # לוח מחוונים
|   ├── editor.html     # עורך פרויקט
|   ├── exercises.html  # רשימת תרגילים
|   ├── exercise.html   # דף תרגיל בודד
|   ├── new_project.html # יצירת פרויקט חדש
|   |
|   └── admin/          # תבניות פאנל אדמין
|       ├── index.html  # דשבורד אדמין
|       ├── users.html  # ניהול משתמשים
|       ├── exercises.html # ניהול תרגילים
|       ├── new_exercise.html # יצירת תרגיל חדש
|       ├── edit_exercise.html # עריכת תרגיל
|       └── stats.html  # סטטיסטיקות
|
├── certs/              # SSL תיקיית תעודות
|   ├── cert.pem
|   └── key.pem
|
└── migrations/         # קבצי מיגרציה של מסד הנתונים
    └── versions/

```

התקנת המערכת

סביבה נדרשת

- Python 3.9 או גרסה חדשה יותר
- GCC (GNU Compiler Collection) מותקן על השרת
- PostgreSQLSCLite
- Docker and Docker compose

כלים נדרשים

התקנת תלויות בסיסיות (Ubuntu) :

```
sudo apt update
sudo apt install python3 python3-pip python3-venv
sudo apt install postgresql postgresql-contrib
sudo apt install gcc build-essential
sudo apt install git
sudo apt install docker.io docker-compose
```

התקנת תלויות Python :

יצירת virtual environment

```
python3 -m venv venv
source venv/bin/activate
```

התקנת packages

```
pip install -r requirements.txt
```

הוראות התקנה

שלב 1: שכפול הפרויקט

```
git clone <repository-url>
cd VCCE
```

שלב 2: הגדרת מסד נתונים

התחברות ל-PostgreSQL

```
sudo -u postgres psql
```

יצירת מסד נתונים ומשתמש

```
CREATE DATABASE code_editor;
CREATE USER code_editor_user WITH PASSWORD 'your_password';
GRANT ALL PRIVILEGES ON DATABASE code_editor TO code_editor_user;
```

\q

שלב 3: הגדרת משתני סביבה

יצירת קובץ .env :

```
DATABASE_URL=postgresql://code_editor_user:your_password@localhost:5432/code_editor
SECRET_KEY=your_very_secret_key_here
ALLOWED_ORIGINS=*
DEBUG=True
```

שלב 4: אתחול מסד הנתונים

```
python models.py
```

שלב 5: הרצת השרת

```
python server.py
```

התקנה עם Docker (מומלץ!!!!):

```
docker-compose up -d
```

נתונים התחלתיים

תרגילים לדוגמה: המערכת יוצרת אוטומטית 5 תרגילי דוגמה:

1. Hello, World!
2. Sum of Two Numbers
3. Factorial Calculation
4. Reverse an Array
5. Binary Search

הגדרות ברירת מחדל:

- פורט שרת: 5001
- מאפשר חיבורים HTTPS
- לוגים ברמת DEBUG
- גיבוי אוטומטי יומי

משתמשי המערכת

משתמש רגיל

יצירת חשבון והתחברות:

1. גישה למערכת :
 - פתח דפדפן וגש לכתובת : <https://localhost:5001>

○ תוצג דף הכניסה הראשי

2. הרשמה למערכת :

- לחץ על "Register here" בדף הכניסה
- מלא את הפרטים הנדרשים :
- שם משתמש (באנגלית, ללא רווחים)
- כתובת אימייל תקפה
- סיסמה (מינימום 6 תווים)

- לחץ על "Register"
- במקרה של הצלחה, תועבר לדף הכניסה

3. התחברות :

- הזן את שם המשתמש והסיסמה
- לחץ על "Login"
- תועבר ללוח המחוונים האישי

עבודה עם פרויקטים :

1. יצירת פרויקט חדש :

- בלוח המחוונים, לחץ על "New Project"
- הזן שם לפרויקט
- לחץ על "Create Project"
- תועבר לעורך הקוד

2. עריכת קוד :

- כתוב את הקוד בעורך
- השתמש ב- Ctrl+Enter להרצה מהירה
- השתמש ב- Ctrl+S לשמירה
- עקוב אחר משתמשים מחוברים ברשימה הימנית

3. קומפילציה והרצה :

- לחץ על "Compile" לקומפילציה בלבד
- לחץ על "Run" להרצה מלאה
- אם התוכנית מחכה לקלט, הזן אותו בשדה הקלט
- צפה בתוצאות בלשוניות המתאימות

4. שיתוף פרויקט :

- בעורך הפרויקט, הזן שם משתמש בשדה "Invite Collaborator"
- לחץ על "Invite"
- המשתמש המוזמן יראה את הפרויקט ברשימת הפרויקטים המשותפים

עבודה עם תרגילים :

1. גישה לתרגילים :

- מלוח המחוונים, לחץ על "View All Exercises"
- או לחץ על תרגיל ספציפי ברשימת התרגילים בתהליך

2. סינון וחיפוש :

- השתמש בכפתורי הסינון לפי קושי (Easy/Medium/Hard)
- השתמש בכפתורי הסינון לפי סטטוס (Not Started/In Progress/Completed)
- כל הסינונים ניתנים לשילוב

3. פתרון תרגיל :

- לחץ על "Start Exercise" או "Continue" בתרגיל הרצוי

- קרא את התיאור והדרישות בצד שמאל
- כתוב את הפתרון בעורך הקוד
- לחץ על "Run & Check" לבדיקה אוטומטית

4. הצגת תוצאות :

- לאחר הרצה, תוצג טבלת תוצאות מפורטת
- כל מקרה בדיקה מציג: קלט, פלט צפוי, פלט בפועל, סטטוס
- תרגיל מושלם כאשר כל מקרי הבדיקה עוברים

5. הצגת פתרון :

- לאחר השלמת התרגיל (או בכל עת), ניתן ללחוץ על "Reveal Solution"
- יוצג הפתרון המלא עם הסברים

תקשורת וצ'אט:

1. צ'אט בפרויקט :

- בעורך הפרויקט, השתמש באזור הצ'אט בצד ימין
- כתוב הודעה ולחץ Enter או על כפתור השליחה
- כל המשתתפים בפרויקט יראו את ההודעה

2. מעקב אחר משתמשים :

- ברשימת המשתמשים המחוברים, תוכל לראות מי מחובר כרגע
- צבעי cursor מציגים מיקום כל משתמש בקוד

ניהול חשבון:

1. שינוי פרטים :

- כרגע לא קיימת אפשרות לשינוי פרטים דרך הממשק
- ניתן לפנות למנהל המערכת

2. התנתקות :

- לחץ על "Logout" בפינה הימנית העליונה
- תועבר לדף הכניסה

משתמש מנהל

גישה לפאנל ניהול:

1. התחברות כמנהל :

- השתמש בחשבון עם הרשאות מנהל
- מלוח המחוונים, תראה כפתור "Admin Panel"
- לחץ עליו לכניסה לפאנל הניהול

ניהול משתמשים:

1. צפייה במשתמשים :

- בפאנל האדמין, לחץ על "Users"
- תוצג רשימת כל המשתמשים במערכת
- ניתן לחפש משתמש ספציפי

2. ניהול משתמש :

- לחץ על כפתור העין לצפייה בפרטי המשתמש

ניהול תרגילים:

1. **צפייה בתרגילים :**
 - לחץ על "Exercises" בתפריט האדמין
 - תוצג רשימת כל התרגילים
 - ניתן לחפש ולסנן תרגילים
2. **יצירת תרגיל חדש :**
 - לחץ על "New Exercise"
 - מלא את הפרטים הנדרשים :
 - כותרת התרגיל
 - רמת קושי (Easy/Medium/Hard)
 - קטגוריה
 - תיאור מפורט
 - קוד התחלתי
 - קוד הפתרון המלא
 - מקרי בדיקה בפורמט JSON
3. **עריכת תרגיל :**
 - לחץ על כפתור העריכה ליד התרגיל
 - ערוך את הפרטים הנדרשים
 - שמור את השינויים
4. **מחיקת תרגיל :**
 - לחץ על כפתור המחיקה
 - אשר את המחיקה בחלון המאשר

פורמט מקרי בדיקה:

```
[
  {
    "input": "5 7",
    "expected_output": "12"
  },
  {
    "input": "10 -3",
    "expected_output": "7"
  }
]
```

צפייה בסטטיסטיקות:

1. **דשבורד אדמין :**
 - דף הבית של האדמין מציג סטטיסטיקות כלליות :
 - מספר משתמשים במערכת
 - מספר תרגילים
 - מספר קומפילציות שבוצעו
2. **סטטיסטיקות מפורטות :**
 - לחץ על "Statistics" לנתונים מפורטים יותר

- גרפים המציגים חלוקת תרגילים לפי קושי
- נתוני הצלחת קומפילציות

ניטור פעילות:

1. משתמשים מחוברים :

- צפייה במשתמשים הפעילים כרגע
- מעקב אחר פעילותם האחרונה

סיכום אישי / רפלקציה

תהליך העבודה על הפרויקט

עבודה על פרויקט זה הייתה מסע מרתק של למידה והתפתחות. בתחילת הדרך, הצבתי לעצמי אתגר משמעותי: ליצור סביבת עבודה מקוונת לשפת C שתהיה ידידותית למשתמש, בטוחה, ומאפשרת עבודה שיתופית בזמן אמת. היעד היה להנגיש את הכלים המתקדמים לפיתוח בשפת C גם למתכנתים מתחילים.

התחלתי בחקירה מעמיקה של הטכנולוגיות הקיימות. מצאתי שקיימות מספר פלטפורמות לעריכת קוד מקוונת, אך אף אחת מהן לא התמקדה ספציפית בשפת C או לא הציעה את השילוב של עריכה שיתופית, סביבת הרצה מאובטחת ואזור תרגול ייעודי.

הצלחות רבות ליוו את הפרויקט. הצלחתי ליישם מערכת התחברות מאובטחת, עורך קוד שיתופי עם סנכרון בזמן אמת, וסביבת הרצה מבודדת לקוד C. במיוחד אני גאה במערכת התרגילים האוטומטית, שמאפשרת למשתמשים לתרגל את כישורי התכנות שלהם ולקבל משוב מיידי.

האתגרים היו רבים גם הם. ההתמודדות עם שיתוף קוד בזמן אמת הציבה אתגרים טכניים מורכבים, במיוחד בהיבטי סנכרון ופתרון התנגשויות. יישום סביבת הרצה מאובטחת היה מאתגר גם כן, שכן שפת C מאפשרת גישה ישירה למשאבי המערכת, מה שמהווה סיכון אבטחתי משמעותי. פתרתי את הבעיות הללו באמצעות מחקר מעמיק, התייעצות עם מומחים, ויישום של פתרונות מתקדמים כמו Operational Transformation ומגבלות משאבים.

תהליך הלמידה

הפרויקט דרש ממני להעמיק בתחומים רבים ומגוונים:

1. **טכנולוגיות Full-Stack:** למדתי לעומק את Flask ו- SQLAlchemy בצד השרת, ו- JavaScript עם SocketIO בצד הלקוח.
2. **תכנות בזמן אמת:** למדתי על תקשורת WebSocket והאלגוריתמים המורכבים הנדרשים לסנכרון עריכה שיתופית.
3. **אבטחת מידע:** העמקתי בנושאי סביבות מבודדות (Sandboxing), הגבלת משאבים, ומניעת התקפות כגון SQL Injection.
4. **הרצת קוד מרוחק:** למדתי כיצד להריץ ולנהל תהליכי קומפילציה והרצה באופן מאובטח.
5. **עיצוב ממשק משתמש:** שיפרתי את מיומנויות ה- HTML, CSS, ו- JavaScript שלי ליצירת ממשק ידידותי ותגובתי.

כל השלבים האלה דרשו למידה עצמית נרחבת. קראתי מאמרים, צפיתי בהרצאות, התייעצתי בפורומים, וערכתי ניסויים רבים. זו הייתה למידה שלא היתה מתאפשרת במסגרת הלימודים הרגילה בכיתה.

כלים לעתיד

הפרויקט העניק לי כלים וידע רב שאקח איתי להמשך דרכי:

1. **מיומנויות פיתוח Full-Stack:** הבנה עמוקה של פיתוח שרת-לקוח API, ותקשורת בזמן אמת.
2. **יכולת תכנון ופיתוח מערכות מורכבות:** למדתי כיצד לחלק בעיה גדולה למודולים ולשלבים, ולתכנן ארכיטקטורה מסודרת.

3. **הבנה עמוקה של אבטחת מידע**: רכשתי ידע מעשי רב על אבטחת יישומי ווב ומניעת איומים.
4. **ניהול פרויקט**: פיתחתי מיומנויות בתכנון, מעקב וניהול של פרויקט מורכב לאורך זמן.
5. **למידה עצמית**: שיפרתי את יכולת הלמידה העצמית שלי, שתשרת אותי בכל אתגר עתידי.

תובנות מהתהליך

במהלך הפרויקט נעזרתי רבות בקהילת המפתחים. הצטרפתי לפורומים של Flask ו-SocketIO, התייעצתי עם מומחי אבטחה, ושיתפתי קטעי קוד עם עמיתים לקבלת משוב. המידע הפתוח והנכונות של מפתחים לעזור זה לזה היו משמעותיים מאוד להצלחת הפרויקט.

נתקלתי בבעיות שלא צפיתי מראש, כמו למשל הקושי בפתרון התנגשויות בעריכה שיתופית או האתגרים בהגנה על המערכת מפני קוד זדוני. אך כל בעיה הייתה הזדמנות ללמידה ולשיפור.

במבט לאחור, אם הייתי מתחיל את הפרויקט מחדש, הייתי מקדיש יותר זמן לתכנון הארכיטקטורה והבנת האלגוריתמים של עריכה שיתופית לפני שהתחלתי בכתובת הקוד. הייתי גם שם דגש רב יותר על כתיבת בדיקות אוטומטיות מההתחלה.

אילו היו לי משאבים נוספים, הייתי מרחיב את הפרויקט עם תכונות כמו:

1. תמיכה בשפות תכנות נוספות מלבד C
2. מערכת להערכת קוד איכותית יותר
3. אפשרויות לדיבאג מתקדם
4. אינטגרציה עם מערכות ניהול גרסאות כמו Git
5. אפשרויות להוראה מרחוק עם יכולות שיתוף מסך ווידאו

סיכום

הפרויקט "סביבת הרצה מקוונת לשפת C" היה מסע של התפתחות והעצמה. למדתי לא רק על טכנולוגיות ויישומים, אלא גם על עצמי כמפתח וכלומד. פיתחתי סבלנות, התמדה ויכולת להתמודד עם אתגרים מורכבים.

אני מאמין שהתוצר הסופי מגשים את המטרה העיקרית: הנגשת כלי פיתוח מתקדמים בשפת C למתכנתים בכל הרמות. המערכת היא ידידותית למשתמש, בטוחה, ומאפשרת שיתוף פעולה - בדיוק מה שחזיתי בתחילת הדרך.

אני מודה למנחה שלי, אופיר שביט, על ההכוונה והתמיכה לאורך כל הדרך. תודה גם למשפחתי וחבריי שתמכו בי ועזרו לי להתמיד גם ברגעים הקשים. ותודה מיוחדת לקהילת המפתחים הפתוחה, שהידע והניסיון שלה היו משאב יקר ערך עבורי.

ביבליוגרפיה

1. **Flask Project.** (2023). *Flask Quickstart - Request Routing and Templates*. Flask Documentation. <https://flask.palletsprojects.com/en/2.3.x/quickstart/#routing>
2. **Flask Project.** (2023). *Flask Sessions and User Authentication*. Flask Documentation. <https://flask.palletsprojects.com/en/2.3.x/quickstart/#sessions>
3. **Flask Project.** (2023). *Flask Blueprints for Application Modularization*. Flask Documentation. <https://flask.palletsprojects.com/en/2.3.x/blueprints/>
4. **Socket.IO.** (2023). *Server API - Rooms and Event Handling*. Socket.IO Documentation. <https://socket.io/docs/v4/server-api/#socketjoin>
5. **Socket.IO.** (2023). *Emitting Events and Broadcasting*. Socket.IO Documentation. <https://socket.io/docs/v4/emit-events/>
6. **SQLAlchemy.** (2023). *ORM Tutorial - Declaring Models and Relationships*. SQLAlchemy Documentation. https://docs.sqlalchemy.org/en/20/tutorial/orm_data_manipulation.html
7. **SQLAlchemy.** (2023). *Relationship Configuration - Many-to-Many*. SQLAlchemy Documentation. https://docs.sqlalchemy.org/en/20/orm/basic_relationships.html#many-to-many
8. **CodeMirror.** (2023). *User Manual - Basic Usage and C-like Mode*. CodeMirror Documentation. <https://codemirror.net/5/doc/manual.html#usage>
9. **CodeMirror.** (2023). *C-like Mode for Syntax Highlighting*. CodeMirror Documentation. <https://codemirror.net/5/mode/clike/>
10. **Mozilla Developer Network.** (2023). *Using WebSockets for Real-time Communication*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications
11. **PostgreSQL Global Development Group.** (2023). *PostgreSQL Tutorial - Creating Tables and Relationships*. PostgreSQL Documentation. <https://www.postgresql.org/docs/current/tutorial-table.html>

נספחים

נספח א': פרוטוקול תקשורת WebSocket

הודעות מהלקוח לשרת

connect

```
{  
  "type": "connect",  
  "data": {  
    "project_id": "string"  
  }  
}
```

join_project

json

```
{  
  "type": "join_project",  
  "data": {  
    "project_id": "string"  
  }  
}
```

edit

```
{  
  "type": "edit",  
  "data": {  
    "type": "insert|delete|retain",  
    "position": "number",  
    "text": "string",  
    "project_id": "string"  
  }  
}
```

cursor_move

```
{  
  "type": "cursor_move",  
  "data": {
```

```
"project_id": "string",  
"position": {  
  "line": "number",  
  "ch": "number"  
}  
}  
}
```

chat_message

```
{  
  "type": "chat_message",  
  "data": {  
    "project_id": "string",  
    "message": "string"  
  }  
}
```

הודעות מהשרת ללקוח

document

```
json  
{  
  "type": "document",  
  "data": {  
    "text": "string"  
  }  
}
```

user_connected

```
{  
  "type": "user_connected",  
  "data": {  
    "username": "string",  
    "user_id": "string",  
    "sid": "string"  
  }  
}
```

user_disconnected

```
{
  "type": "user_disconnected",
  "data": {
    "username": "string",
    "sid": "string"
  }
}
```

cursor_update

```
{
  "type": "cursor_update",
  "data": {
    "user_id": "string",
    "username": "string",
    "position": {
      "line": "number",
      "ch": "number"
    }
  }
}
```

new_chat_message

```
{
  "type": "new_chat_message",
  "data": {
    "user_id": "string",
    "username": "string",
    "message": "string",
    "timestamp": "ISO8601_string"
  }
}
```

נספח ב': שמות תיקיות וקבצים שהוגבלו לשימוש בסנדבוקס

תיקיות אסורות:

- קובצי הגדרות מערכת - /etc/

- /var/ - נתונים משתנים של המערכת
- /usr/ - (תוכנות משתמש (קריאה בלבד מותרת -
- /home/ - תיקיות בית משתמשים אחרים -
- /root/ - תיקיית מנהל המערכת -
- /proc/ - מידע על תהליכים -
- /sys/ - מידע על מערכת -
- /dev/ - קובצי התקנים -

קבצים מוגבלים:

- /etc/passwd - רשימת משתמשים -
- /etc/shadow - סיסמאות מוצפנות -
- /etc/hosts - הגדרות רשת -
- /-כל קובץ ההתחלה ב

הרחבות קבצים אסורות:

- .sh - shell script קובצי -
- .py - Python קובצי -
- .pl - Perl קובצי -
- .php - PHP קובצי -

נספח ג': אלגוריתם בדיקת קוד לתבניות מסוכנות

רגקסים לזיהוי פונקציות מסוכנות:

```
dangerous_patterns = [
    r'system\s*\(',      # system() calls
    r'popen\s*\(',      # popen() calls
    r'fopen\s*\(',      # File operations
    r'fwrite\s*\(',      # File write operations
    r'fprintf\s*\(',    # File write operations
    r'exec[lv][pe]\s*\(', # exec family functions
    r'fork\s*\(',        # Process creation
    r'unlink\s*\(',      # File deletion
    r'remove\s*\(',      # File deletion
    r'rename\s*\(',      # File renaming
    r'mkdir\s*\(',       # Directory creation
    r'rmdir\s*\('       # Directory removal
]
```

אלגוריתם הבדיקה:

```
def check_for_dangerous_code(code):  
    import re  
    for pattern in dangerous_patterns:  
        if re.search(pattern, code):  
            return True  
    return False
```

קוד הפרויקט :

קוד הפרוייקט יצורף בעמודים הבאים, בתחילת כל section של קוד נמצא שם הקובץ.