**מדריך לכתיבת תיק פרויקט לפיתוח Web**

מבוא תיק הפרויקט הוא מסמך המתאר את שלבי העבודה בפיתוח האפליקציה, כולל תכנון, פיתוח ותיעוד כל החלקים של המערכת. המסמך כולל:

1. אפיון ותכנון הפרויקט.
2. שלבי פיתוח צד הלקוח.
3. שלבי פיתוח צד השרת.
4. תכנון מסד הנתונים.
5. בדיקות ואבטחת מידע.
6. סיכום והפקת לקחים.
7. הצגת פרויקט.
8. **אפיון ותכנון הפרויקט**

**1.1. שם ותיאור הפרויקט**

* **שם הפרויקט: אתר אינטרנט** Smoke
* **תיאור כללי:** האתר מאפשר למתכנתי משחקים להשתמש ב Smoke להצגת הפרויקטים שלהם ולהיחשף כך לחברות ואנשים שמחפשים מתכנתי משחקים.
* **בעיה שהפרויקט פותר:** הפרויקט מעניק פתרון אמיתי למתכנתי משחקים רבים, שמתמודדים עם תחרות גבוהה ושוק עבודה מצומצם.
* **קהל יעד:** מתכנתים וחברות למשחקי מחשב שמחפשות מתכנתי משחקים.

**1.2. דרישות מערכת**

**1 דרישות פונקציונליות (מה המשתמש יכול לעשות?)**  
✅ המשתמש יכול להירשם ולהתחבר למערכת

✅ המשתמש יכול להעלות לאתר את הפרויקטים שלו ולהיחשף למעסיקים פוטנציאליים.

✅ המערכת מציגה את הפרויקטים לפי סדר העלאתם (כרונולוגי)

✅ המשתמש יכול לצפות בפרופילים של שאר המשתמשים (על המומחיות שלהם / הפרויקטים שלהם ועוד)

✅ המשתמש יכול ליצור קשר עם ה support במקרה של תקלה או צורך בתמיכה.

**1.3. דרישות טכניות (באילו טכנולוגיות משתמשים?)**   
✅ **Frontend:** HTML, CSS, JavaScript

✅ **Backend:** Node.js, Express

✅ **Database**: SQLite

**2. תכנון מבנה האפליקציה**

**2.1 מבנה תיקיית הפרויקט**

1. **מבנה תיקיות:**

/smoke

│── /public # קבצי HTML

│── │── /functionality # קבצי JS

│── │── /styles # קבצי CSS

│── /server # קובץ הגדרות מסד הנתונים

│── │── /db # מרכז את מסד הנתונים

│── package.json # קובץ תלויות Node.js   
│── README.md # תיעוד הפרויקט

│── server.js # קובץ ניהול השרת

1. **הסבר על כל תיקייה:**

│── /public - מכיל את קוד front end

│── │── /functionality - מכיל את קוד הפונקציונליות (Java Script)

│── │── /styles - מכיל את קוד העיצוב (CSS)

│── /server - מכיל את קוד ה Data Base

│── │── /db - מכיל את ה Data Base

│── package.json - תיקיית ה Json

**2.2 תרשים ניווט בין מסכים**

1. **מסכים עיקריים:**  
 ✅ עמוד הבית → מציג את אודות האתר  
 ✅ עמוד התחברות → מאפשר כניסה למערכת   
 ✅ עמוד הפרופיל → מציג את פרופיל המשתמש

1. **תרשים זרימה של הניווט**

תמונה שמכילה טקסט, צילום מסך, גופן, מסמך

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**3. פיתוח צד הלקוח (Frontend)**

**3.1 דפי HTML**

**דפים עיקריים:**

✅ Index.html – מציג את עמוד הבית

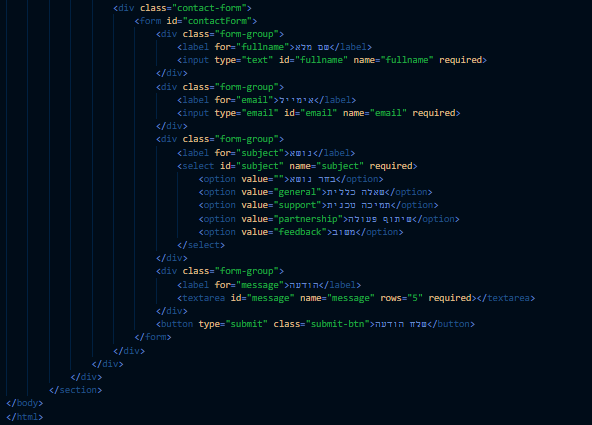
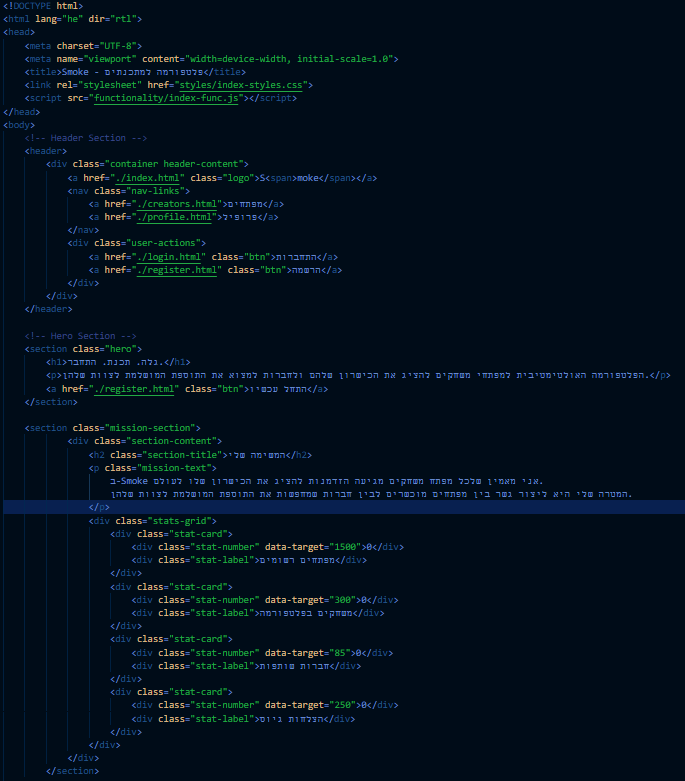
✅ login.html – מאפשר התחברות

✅ register.html – מאפשר הרשמה

✅ creators.html – מציג את כל המשתמשים

✅ profile.html – מציג את עמוד הפרופיל

📌 קטע קוד לדף הבית (index.html)



📌 הסבר מפורט על כל רכיב בקוד:

| **רכיב** | **תפקיד** |
| --- | --- |
| <section class="hero"> | אזור ההרואו (כותרת + טקסט פתיחה) בראש העמוד |
| <h1>, <p> | מציגים כותרת וטקסט תיאור של הפרויקט |
| <a href="./register.html"> | כפתור קריאה לפעולה להרשמה |
| <h2 class="section-title"> | מציג את שם הסקשן: "המשימה שלי" |
| <p class="mission-text"> | פסקה שמסבירה את מטרת היוזמה – סיוע למפתחים ויצירת קהילה |
| <div class="stats-grid"> | גריד שמכיל קלפים עם נתונים מספריים (כמו מספר המשתמשים) |
| <div class="stat-card"> | קלף שמציג נתון מסוים (כמות, תיאור) |
| data-target="..." | מאפיין מותאם אישית שמשמש לרוב לאנימציית ספירה עם JavaScript |
| <h2 class="section-title"> | מציג את כותרת הסקשן: "הערכים שלנו" |
| <div class="value-card"> | קלף לערך אחד (למשל קהילה, שקיפות, מקצועיות) |
| <div class="value-icon"> | הצגת אייקון ליד הערך (למשל 💛, 📣, ✨) |
| <h3>, <p> | שם הערך + תיאור קצר שלו |
| <h2 class="section-title"> | כותרת הסקשן: "צור קשר" |
| <div class="contact-method"> | כרטיס תקשורת: מציג דרך ליצור קשר (מייל, דיסקורד, טוויטר) |
| <span class="contact-icon"> | אייקון של אמצעי התקשורת (למשל ✉️ לדוא"ל) |
| <strong> | כותרת לאמצעי (למשל: "אימייל", "דיסקורד") |
| <p> | המידע עצמו (כתובת מייל, לינק וכו') |
| <input type="text" name="fullname"> | שדה להזנת שם מלא |
| <input type="email" name="email"> | שדה להזנת אימייל |
| <select name="subject"> | תיבת בחירה לנושא הפנייה (שיתוף פעולה, תמיכה וכו') |
| <textarea name="message"> | שדה טקסט חופשי להודעה |
| <button type="submit"> | כפתור לשליחת הטופס |
|  |  |

📌 קטע קוד לדף התחברות (login.html)



📌 הסבר מפורט על כל רכיב בקוד:

| **רכיב** | **תפקיד** |
| --- | --- |
| <!DOCTYPE html> | מציין לדפדפן שזהו מסמך HTML5 |
| <html lang="he" dir="rtl"> | אלמנט ראשי של הדף, מגדיר עברית וכיוון מימין לשמאל |
| <head> | חלק עליון של הדף: כולל מטא-מידע, קישורים לקבצי CSS ו-JS, כותרת הדף וכו' |
| <meta charset="UTF-8"> | קובע קידוד תווים לעברית ואנגלית – כדי שיוצגו נכון |
| <meta name=”viewport” . . . > | מאפשר התאמה למובייל – רספונסיביות |
| <title> | כותרת לשונית הדפדפן |
|  | קישור לקובץ CSS חיצוני לעיצוב |
| <script src="..." defer> | קישור לקובץ JavaScript שמופעל רק אחרי טעינת הדף |
| <div class="user-actions"> | פעולות משתמש – קישורים לעמוד התחברות והרשמה |
| <header> | אזור ראש הדף – כולל לוגו, תפריט וקישורים |
| <div class="header-content"> | עוטף את תוכן ההדר לצורך סידור ועיצוב |
| <a href="./index.html"> | קישור ללוגו – מחזיר לדף הבית |
| <nav class="nav-links"> | תפריט ניווט – כולל קישורים לדפים: יוצרים, פרופיל |
| <main class="login-main"> | החלק המרכזי בדף – מכיל את טופס ההתחברות |
| <form id="loginForm"> | טופס ההתחברות שמכיל את שדות האימייל, הסיסמה, "זכור אותי" וכפתור שליחה |
| <input type="email"> | שדה קלט לכתובת אימייל עם ולידציה |
| <input type="password"> | שדה סיסמה – התווים מוסתרים |
| <input type="checkbox"> | תיבת סימון "זכור אותי" |
| <button type="submit"> | כפתור לשליחת הטופס |
| <span class="error-message"> | הצגת הודעת שגיאה ליד שדות הקלט |
| <div class="social-login"> | קופסה נפרדת לכפתורי התחברות חברתיים |
| <button class="google-btn"> | כפתור התחברות דרך חשבון Google |
| <button class="github-btn"> | כפתור התחברות דרך GitHub |
| <div class="register-link"> | שורה בסוף הטופס – קישור לעמוד הרשמה |
| <a href="./register.html"> | קישור בפועל לעמוד הרשמה |

📌 קטע קוד לדף הרשמה (register.html)





📌 הסבר מפורט על כל רכיב בקוד:

| **רכיב** | **תפקיד** |
| --- | --- |
| <main class="profile-main"> | גוף העמוד – אזור התוכן הראשי לדף הפרופיל |
| <div class="container"> | קופסה כללית לסידור ועיצוב אלמנטים פנימיים |
| <section class="profile-section"> | הסקשן הראשי של הפרופיל – עוטף את כל המידע האישי של המשתמש |
| <div class="profile-card"> | כרטיס מידע של המשתמש – פרטים בסיסיים |
| <div class="avatar-wrapper"> | עוטף את התמונה של המשתמש |
| <img src="..." class="avatar"> | מציג את תמונת המשתמש |
| <h2 class="username"> | שם המשתמש (כותרת) |
| <p class="user-specialty"> | תחום התמחות (למשל: מפתח Unity) |
| <div class="user-stats"> | שורת נתונים מספריים – כמות עוקבים, דירוגים וכו' |
| <div class="user-stat"> | קלף אחד של נתון – כולל מספר וכותרת |
| <section class="details-section"> | חלק של תיאור עצמי + פרטי קשר של המשתמש |
| <p class="description"> | תיאור קצר של המשתמש, מה הוא עושה |
| <ul class="user-details"> | רשימה של פרטים נוספים כמו מיקום, אימייל, שנות ניסיון וכו' |
| <li><strong>...</strong>: ...</li> | הצגת כל שדה מודגש (כותרת) + ערך (למשל: מיקום: תל אביב) |
| <section class="projects-section"> | אזור להצגת פרויקטים של המשתמש |
| <h2 class="section-title"> | כותרת הסקשן – "הפרויקטים שלי" |
| <div class="projects-grid"> | גריד שמכיל קלפים – כל אחד מציג פרויקט |
| <div class="project-card"> | קלף פרויקט אחד – שם הפרויקט וסוגו |
| <div class="project-title"> | שם הפרויקט |
| <div class="project-type"> | סוג הפרויקט (למשל: משחק AR, פאזל) |

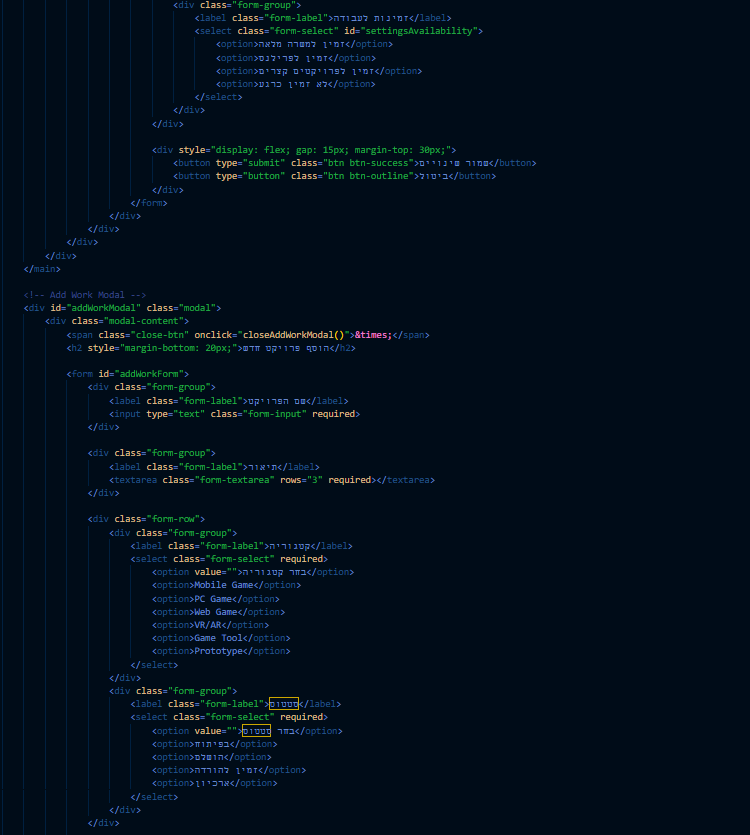
📌 קטע קוד לדף המשתמשים (creators.html)



📌 הסבר מפורט על כל רכיב בקוד:

| **רכיב** | **תפקיד** |
| --- | --- |
| <!DOCTYPE html> | מציין שמדובר במסמך HTML5 |
| <html lang="he" dir="rtl"> | הגדרת עברית וכתיבה מימין לשמאל |
| <head> ... </head> | כולל הגדרות מטא, קישורי CSS ו-JS |
| <link rel="stylesheet"...> | קישור לקובץ עיצוב CSS שמותאם לדף היוצרים |
| <script src="..."> | קישור לקובץ JavaScript שמטפל בהתנהגות הדף (סינון, פתיחת פרופילים) |
| <header> | אזור ראש האתר – כולל לוגו, תפריט ניווט ופעולות משתמש |
| <a href="./index.html"> | קישור ללוגו (דף הבית) |
| <nav class="nav-links"> | תפריט ניווט – קישורים ליוצרים ופרופיל |
| <div class="user-actions"> | קישורים להתחברות והרשמה |
| <section class="hero"> | אזור הצגה ראשוני – כולל כותרת ותיאור קצר לדף "יוצרים" |
| <h1>, <p> | טקסט שמציג את מטרת העמוד – למצוא יוצרים מתאימים לפי התמחות |
| <section class="filters-section"> | אזור סינון לפי תחומי התמחות |
| <label class="filters-label"> | תווית: "התמחויות היוצרים:" |
| <div class="specialty-tag"> | כפתור שמתאר תחום התמחות (למשל Unity, 2D, AR) |
| class="specialty-tag active" | הכפתור שנבחר כעת (כברירת מחדל) |
| data-specialty="..." (אם קיים) | מאפשר פילטרינג בצד לקוח לפי התמחות (מטופל ב-JavaScript) |
| <div id="profileModal"> | מודל קופץ להצגת פרופיל יוצר בפירוט – מוסתר כברירת מחדל (style="display:none") |
| <div class="modal-content"> | עוטף את תוכן המודל כולו |
| <span class="close-btn">×</span> | כפתור לסגירת המודל |
| <div id="modalProfileContent"> | אזור לתוכן הפרופיל שנטען דינמית (שם, כישורים, פרויקטים וכו') |

📌 קטע קוד לדף הפרופיל (profile.html)

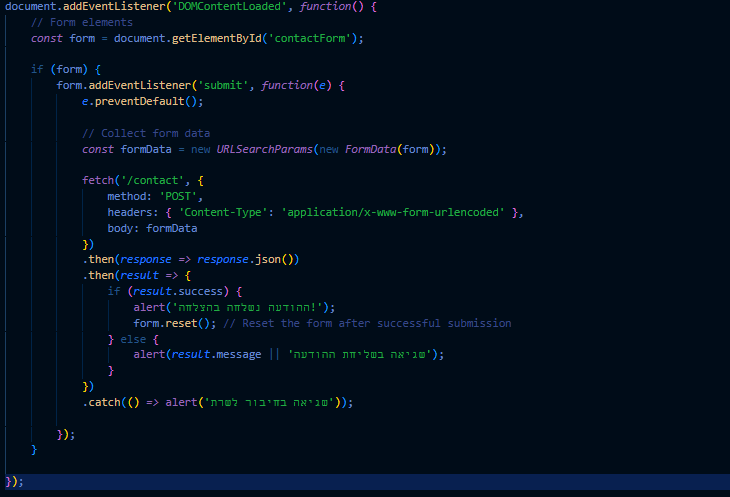


📌 הסבר מפורט על כל רכיב בקוד:

| **רכיב** | **תפקיד** |
| --- | --- |
| <header> | תפריט עליון עם ניווט ולחצני התחברות/הרשמה |
| <a href="./profile.html" class="active"> | מציין שהעמוד הנוכחי הוא "פרופיל" |
| <section class="profile-header"> | אזור כותרת הפרופיל (avatar + פרטים) |
| <div class="profile-avatar"> | מציג את תמונת הפרופיל |
| <div class="avatar-edit"> | אזור כפתור לעריכת התמונה |
| <div class="profile-details"> | מציג את פרטי המשתמש (שם, תחום וכו') |
| <main class="profile-content"> | גוף עמוד הפרופיל |
| <aside class="profile-sidebar"> | סרגל צד שמכיל כישורים ופעולות מהירות |
| <div class="skills-grid" id="skillsGrid"> | רשת הכישורים המוצגים של המשתמש |
| <input id="newSkill"> | שדה להזנת כישור חדש |
| <button onclick="addSkill()"> | כפתור להוספת כישור חדש |
| <button onclick="openAddWorkModal()"> | כפתור לפתיחת טופס הוספת עבודה |
| <div class="tab-navigation"> | ניווט בין טאב "עבודות" ל־"הגדרות" |
| <div id="worksGrid"> | תצוגת העבודות של המשתמש |
| <div class="settings-tab"> | טאב הגדרות פרופיל |
| <input id="settingsUsername"> | שדה עריכת שם משתמש |
| <select id="settingsSpecialty"> | בחירת תחום התמחות |
| <textarea id="settingsBio"> | שדה טקסט לביוגרפיה אישית |
| <select id="settingsExperience"> | בחירת רמת ניסיון |
| <select id="settingsAvailability"> | בחירת זמינות לעבודה |
| <form id="addWorkForm"> | טופס הוספת פרויקט חדש |
| <input type="text"> (בתוך addWorkForm) | שם הפרויקט |
| <textarea> (בתוך addWorkForm) | תיאור הפרויקט |
| <select> (קטגוריה) | קטגוריית הפרויקט |
| <select> (סטטוס) | סטטוס הפרויקט (טיוטה/מוכן וכו') |

**3.2 קוד JavaScript (Frontend)**

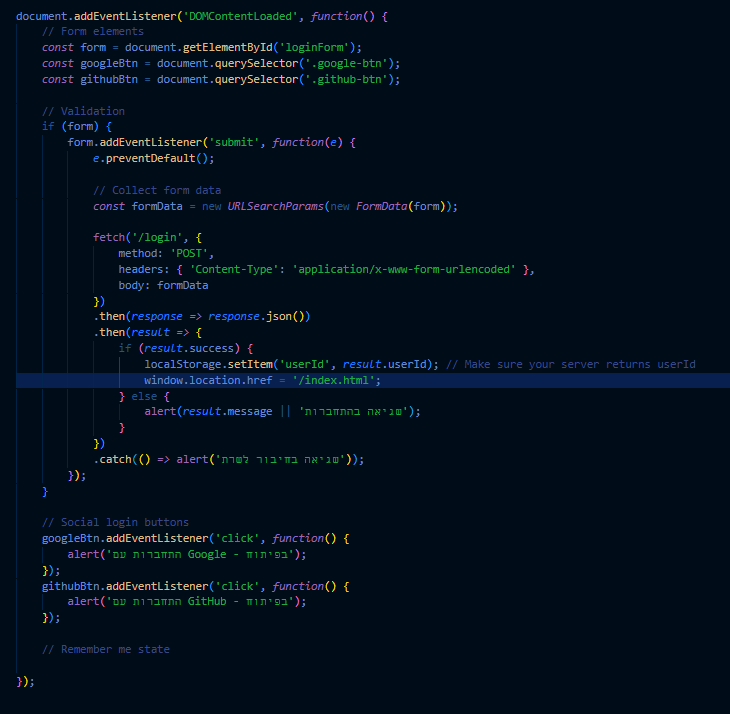
📌 קטע קוד לטיפול בדף הבית ([index-func.js](http://index-func.js))



📌 **הסבר מפורט:**

| **שורת קוד** | **מה היא עושה** |
| --- | --- |
| document.addEventListener('DOMContentLoaded', function() { | מפעיל את הקוד רק אחרי טעינת הדף |
| const form = document.getElementById('contactForm'); | מאחסן את טופס יצירת הקשר במשתנה |
| if (form) { | בודק אם הטופס קיים בדף |
| form.addEventListener('submit', function(e) { | מאזין לאירוע שליחת הטופס |
| e.preventDefault(); | מונע שליחה רגילה של הטופס |
| const formData = new URLSearchParams(new FormData(form)); | יוצר את הנתונים לשליחה מהטופס |
| fetch('/contact', { | שולח בקשת POST לשרת בכתובת /contact |
| method: 'POST', | מגדיר את שיטת השליחה כ-POST |
| headers: { 'Content-Type': 'application/x-www-form-urlencoded' }, | מגדיר את סוג התוכן |
| body: formData | מצרף את נתוני הטופס לבקשה |
| }).then(response => response.json()) | ממיר את תגובת השרת ל-JSON |
| .then(result => { | מטפל בתוצאה שהגיעה |
| if (result.success) { | בודק אם השליחה הצליחה |
| alert('ההודעה נשלחה בהצלחה!'); | מציג הודעה שההודעה נשלחה |
| form.reset(); | מאפס את הטופס לאחר שליחה מוצלחת |
| } else { | אם הייתה שגיאה מצד השרת |
| }).catch(() => alert('שגיאה בתקשורת לשרת')); | מציג שגיאה במקרה של כשל בתקשורת |

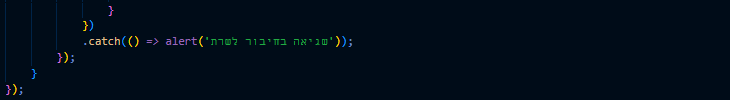
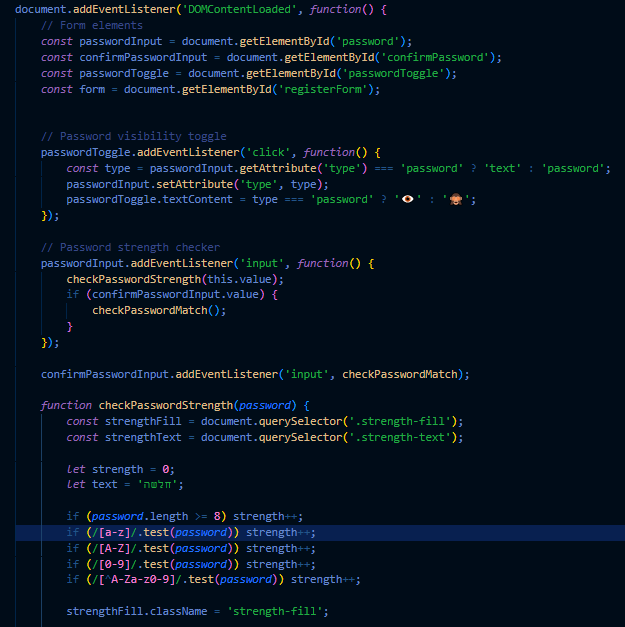
📌 קטע קוד לטיפול בטופס התחברות (login-func.js)



📌 **הסבר מפורט:**

| **שורת קוד** | **מה היא עושה** |
| --- | --- |
| document.addEventListener('DOMContentLoaded', function() { | מפעיל את הקוד רק לאחר טעינת הדף |
| const form = document.getElementById('loginForm'); | שומר את טופס ההתחברות במשתנה |
| const googleBtn = document.querySelector('.google-btn'); | שומר את כפתור Google במשתנה |
| const githubBtn = document.querySelector('.github-btn'); | שומר את כפתור GitHub במשתנה |
| if (form) { | בודק אם הטופס קיים בדף |
| form.addEventListener('submit', function(e) { | מאזין לשליחת הטופס |
| e.preventDefault(); | מונע את שליחת הטופס הרגילה |
| const formData = new URLSearchParams(new FormData(form)); | אוסף את הנתונים מהטופס |
| fetch('/login', { | שולח בקשת POST לכתובת /login |
| method: 'POST', | מציין שהבקשה היא בשיטת POST |
| headers: { 'Content-Type': 'application/x-www-form-urlencoded' }, | מגדיר סוג תוכן לשליחה |
| body: formData | מצרף את הנתונים לבקשה |
| }).then(response => response.json()) | ממיר את התגובה ל-JSON |
| .then(result => { | ממשיך לטפל בתוצאה שהתקבלה |
| if (result.success) { | בודק אם ההתחברות הצליחה |
| localStorage.setItem('userId', result.userId); | שומר את מזהה המשתמש בדפדפן |
| window.location.href = '/index.html'; | מעביר את המשתמש לדף הבית |
| } else { | אחרת – התחברות נכשלה |
| }).catch(() => alert('שגיאה בתקשורת לשרת')); | מציג שגיאה אם הבקשה נכשלה |
| googleBtn.addEventListener('click', function() { | מאזין ללחיצה על כפתור Google |
| alert('בפיתוח - התחברות Google'); | מציג הודעת alert זמנית |
| githubBtn.addEventListener('click', function() { | מאזין ללחיצה על כפתור GitHub |
| alert('בפיתוח - התחברות GitHub'); | מציג הודעת alert זמנית |

📌 קטע קוד לטיפול בטופס הרשמה (register-func.js)



📌 **הסבר מפורט:**

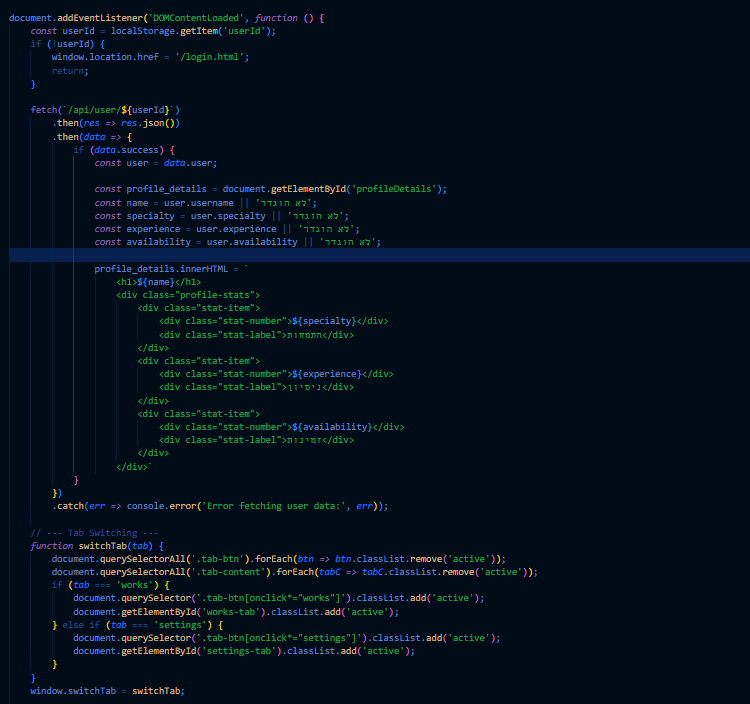
| **שורת קוד** | **מה היא עושה** |
| --- | --- |
| document.addEventListener('DOMContentLoaded', function() { | מפעיל את הקוד רק אחרי שהדף נטען |
| const passwordInput = document.getElementById('password'); | שומר את שדה הסיסמה |
| const confirmPasswordInput = document.getElementById('confirmPassword'); | שומר את שדה אישור סיסמה |
| const passwordToggle = document.getElementById('passwordToggle'); | שומר את כפתור הצגת הסיסמה |
| const form = document.getElementById('registerForm'); | שומר את טופס ההרשמה |
| passwordToggle.addEventListener('click', function() { | מאזין ללחיצה על כפתור הצגת הסיסמה |
| const type = passwordInput.getAttribute('type') === 'password' ? 'text' : 'password'; | מחליף את סוג השדה מסיסמה לטקסט ולהפך |
| passwordInput.setAttribute('type', type); | מגדיר את סוג השדה החדש |
| passwordToggle.textContent = type === 'password' ? '👁️' : '🙈'; | משנה את האייקון בהתאם למצב |
| passwordInput.addEventListener('input', function() { | מאזין להזנת סיסמה חדשה |
| checkPasswordStrength(this.value); | קורא לפונקציית בדיקת חוזק סיסמה |
| if (confirmPasswordInput.value) { checkPasswordMatch(); } | אם יש סיסמה חוזרת, בודק אם הן תואמות |
| confirmPasswordInput.addEventListener('input', checkPasswordMatch); | מאזין להזנה בשדה אישור סיסמה |
| function checkPasswordStrength(password) { | פונקציה לבדיקת חוזק סיסמה |
| const strengthFill = document.querySelector('.strength-fill'); | תופס את פס הצבע של חוזק הסיסמה |
| const strengthText = document.querySelector('.strength-text'); | תופס את טקסט הדירוג |
| let strength = 0; | קובע את ערך החוזק ההתחלתי |
| let text = 'חלשה'; | קובע טקסט התחלה |
| if (password.length >= 8) strength++; | בודק אם אורך הסיסמה מספיק |
| if (/[a-z]/.test(password)) strength++; | אם יש אותיות קטנות – מוסיף נקודה |
| if (/[A-Z]/.test(password)) strength++; | אם יש אותיות גדולות – מוסיף נקודה |
| if (/[0-9]/.test(password)) strength++; | אם יש ספרות – מוסיף נקודה |
| if (/[^A-Za-z0-9]/.test(password)) strength++; | אם יש תווים מיוחדים – מוסיף נקודה |
| switch(strength) { case ... } | קובע צבע וטקסט לפי דרגת חוזק |
| strengthFill.classList.add('weak'/'fair'/'good'/'strong') | משנה את צבע הפס לפי חוזק |
| strengthText.textContent = \חוזק הסיסמה: ${text}`;` | מציג את חוזק הסיסמה בטקסט |
| function checkPasswordMatch() { | בודק אם הסיסמאות תואמות |
| if (confirmPassword === password) | אם הסיסמאות שוות |
| confirmPasswordInput.setCustomValidity(''); | מסיר שגיאה |
| confirmPasswordInput.classList.remove('valid'); | מסיר מחלקת שגיאה |
| else { | אם הסיסמאות לא תואמות |
| confirmPasswordInput.setCustomValidity('סיסמאות אינן תואמות'); | מגדיר שגיאה |
| confirmPasswordInput.classList.add('invalid'); | מסמן שגיאה ב־CSS |
| if (form) { | בודק אם הטופס קיים |
| form.addEventListener('submit', function(e) { | מאזין לשליחת טופס |
| e.preventDefault(); | מונע שליחה רגילה |
| const formData = new URLSearchParams(new FormData(form)); | אוסף את נתוני הטופס |
| fetch('/register', { | שולח את הטופס לשרת |
| method: 'POST', | מציין POST |
| headers: { 'Content-Type': 'application/x-www-form-urlencoded' }, | מגדיר סוג התוכן |
| body: formData | מצרף את הנתונים |
| }).then(res => res.json()) | הופך את התגובה ל-JSON |
| .then(result => { | ממשיך בטיפול בתוצאה |
| if (result.success) { | אם ההרשמה הצליחה |
| window.location.href = '/'; | מעביר לדף הבית |
| }).catch(() => alert('שגיאה בתקשורת לשרת')); | מציג שגיאה כללית אם נכשלה הבקשה |

📌 קטע קוד לטיפול בדף היוצרים (creators-func.js)

📌 **הסבר מפורט:**

| **שורת קוד** | **מה היא עושה** |
| --- | --- |
| document.addEventListener('click', function(e) {...}) | מאזין לכל קליק בדף |
| if (e.target.classList.contains('btn-small')) | בודק אם נלחץ כפתור "צפה בפרופיל" |
| const card = e.target.closest('.creator-card'); | מוצא את כרטיס היוצר שנלחץ |
| const creatorIndex = parseInt(card.getAttribute('data-creator-index')); | מאחזר את אינדקס היוצר מתוך ה-HTML |
| const creator = creators[creatorIndex]; | שומר את אובייקט היוצר מתוך המערך |
| modalContent.innerHTML = createEnhancedModalContent(creator); | ממלא את תוכן המודל לפי היוצר שנבחר |
| modal.style.display = 'flex'; | מציג את המודל על המסך |
| document.body.style.overflow = 'hidden'; | חוסם גלילה של הדף כשמודל פתוח |
| document.addEventListener('keydown', function(event) {...}) | מאזין להקשת מקש במקלדת |
| if (event.key === 'Escape' && modal.style.display === 'flex') | בודק אם נלחץ ESC כשמודל פתוח |
| modal.style.display = 'none'; | סוגר את המודל |
| document.body.style.overflow = 'auto'; | מחזיר את הגלילה של הדף |
| function createCreatorCard(creator) { ... } | יוצרת HTML עבור כרטיס יוצר |
| data-creator-index="${creators.indexOf(creator)}" | שומר אינדקס יוצר בתור data attribute |
| ${creator.name}, ${creator.avatar} וכו' | ממלא שדות בכרטיס כמו שם, תמונה וכו' |
| ${creator.skills.map(...).join('')} | ממיר מערך כישורים לתגיות HTML |
| <button class="btn btn-small">צפה בפרופיל</button> | כפתור לפתיחת מודל פרופיל |
| function createEnhancedModalContent(creator) | מחזירה HTML מורחב של פרופיל יוצר |
| const stars = '★'.repeat(...) + '☆'.repeat(...) | מציג דירוג כוכבים לפי ציון |
| ${creator.skills.map(...)} | מציג תגיות כישורים במודל |
| ${creator.projects.map(...)} | מציג את כל הפרויקטים של היוצר |
| ${experienceText[creator.experience]} | ממיר קוד ניסיון (למשל 'mid') לתיאור |
| ${locationText[creator.location]} | ממיר קוד מיקום (למשל 'remote') לשם מקום |
| ${creator.description} | מציג תיאור כללי של היוצר |
| ${creator.phone}, ${creator.email} | מציג פרטי יצירת קשר |
| const activeSpecialty = document.querySelector('.specialty-tag.active').textContent; | מקבל את התחום שנבחר כפעיל |
| let filteredCreators = [...creators]; | יוצר עותק של מערך היוצרים |
| if (activeSpecialty !== 'הכל') { ... } | בודק אם בחרו פילטר מסוים |
| creator.skills.some(skill => ...) | מסנן יוצרים לפי התאמת כישורים |
| renderCreators(filteredCreators); | מציג את היוצרים במסך |
| document.querySelectorAll('.specialty-tag').forEach(...) | מאזין ללחיצה על תגיות סינון |
| tag.classList.add('active'); | מסמן תגית כלחוצה |
| filterCreators(); | מפעיל את סינון היוצרים |
| modal.addEventListener('click', (e) => { if (e.target === modal) ... }) | סוגר את המודל בלחיצה מחוץ לחלון |
| closeBtn.addEventListener('click', () => { ... }) | סוגר את המודל בלחיצה על X |
| modal.style.display = 'none'; | מחביא את המודל |
| document.body.style.overflow = 'auto'; | מחזיר את אפשרות הגלילה |

📌 קטע קוד לטיפול בדף הפרופיל (profile-func.js)



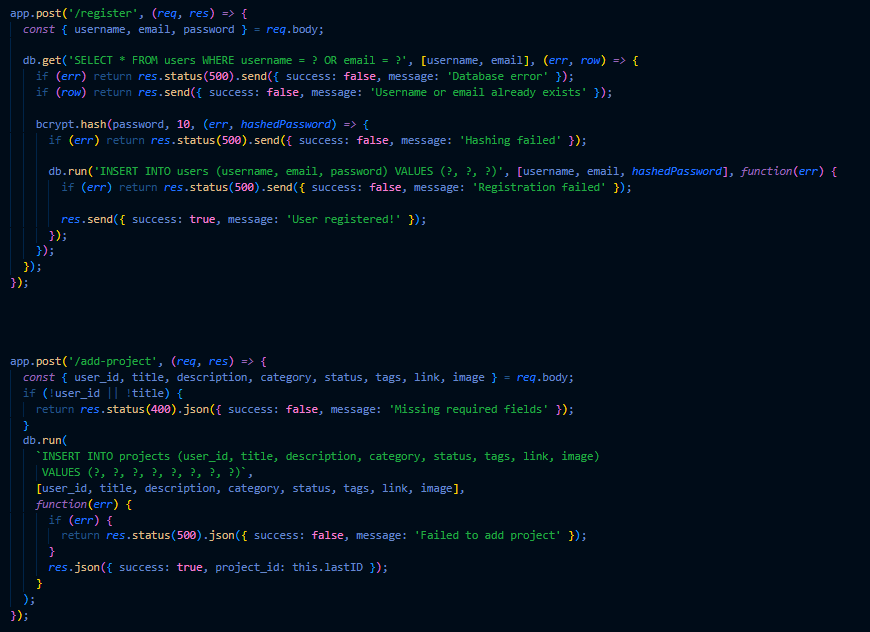
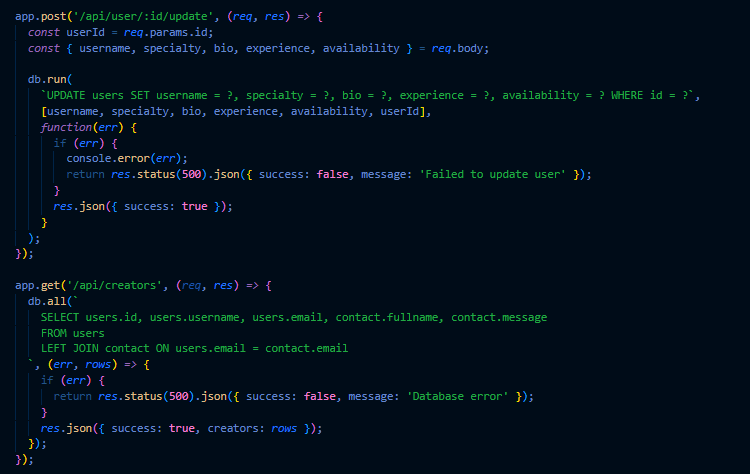
📌 **הסבר מפורט:**

| **שורת קוד** | **מה היא עושה** |
| --- | --- |
| document.addEventListener('DOMContentLoaded', function () { | מפעיל את הקוד רק לאחר טעינת הדף |
| const userId = localStorage.getItem('userId'); | שולף מזהה משתמש מהדפדפן |
| if (!userId) { window.location.href = '/login.html'; return; } | מעביר לדף התחברות אם אין userId |
| fetch('/api/user/' + userId) | מבקש פרטי משתמש מהשרת |
| .then(res => res.json()) | ממיר את התשובה ל-JSON |
| if (data.success) { const user = data.user; ... } | ממשיך רק אם התשובה הצליחה |
| profile\_details.innerHTML = ... | מכניס את פרטי המשתמש לתוך הדף |
| function switchTab(tab) { ... } | מחליף בין טאב העבודות להגדרות |
| window.switchTab = switchTab; | מאפשר קריאה לפונקציה מה-HTML |
| const skillsGrid = document.getElementById('skillsGrid'); | מזהה את גריד הכישורים |
| const newSkillInput = document.getElementById('newSkill'); | מזהה את שדה קלט הכישור |
| window.addSkill = function () { ... } | מוסיף כישור חדש לגריד |
| window.removeSkill = function (el) { ... } | מסיר כישור קיים בלחיצה על כפתור X |
| window.openAddWorkModal = function () { ... } | פותח את המודל |
| window.closeAddWorkModal = function () { ... } | סוגר את המודל ומאפס את הטופס |
| addWorkForm.addEventListener('submit', function (e) { ... }) | מאזין לשליחת טופס העבודה החדשה |
| worksGrid.appendChild(workCard); | מוסיף את העבודה לגריד בדף |
| const settingsForm = document.querySelector('#settings-tab form'); | מזהה את טופס ההגדרות |
| settingsForm.addEventListener('submit', function (e) { ... }) | מאזין לשליחת טופס ההגדרות |
| const userId = localStorage.getItem('userId'); | שולף את מזהה המשתמש |
| const newName = ... עד newAvailability = ... | שומר את כל הערכים החדשים מהטופס |
| const formData = new URLSearchParams(); ... | יוצר את גוף השליחה |
| fetch('/api/user/'+userId+'/update', {...}) | שולח את הנתונים לשרת |
| .then(response => response.json()) | ממיר את התשובה ל-JSON |
| if (result.success) {...} | אם הצליח – מעדכן בדף |
| else { alert(...) } | אם נכשל – מציג שגיאה |
| window.openAvatarModal = function () { alert('...'); } | מדמה פתיחה של מודל לתמונת פרופיל |

**4. פיתוח צד השרת (Backend)**

**4.1 יצירת שרת (Node.js)**

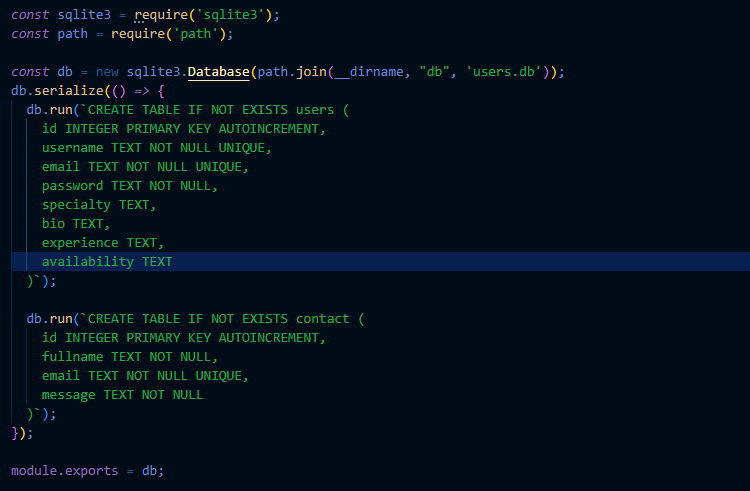
📌 קטע קוד ליצירת שרת ([server.js](http://server.js))



**5. תכנון מסד הנתונים**

**5.1 מבנה מסד נתונים**

📌 קטע קוד למסד נתונים (server/[db.js](http://db.js))



**6. בדיקות ואבטחת מידע**

