תיק פרויקט: חשבון בנק

## אפיון ותכנון הפרויקט

**שם ותיאור הפרויקט:**

* **שם הפרויקט:** חשבון בנק
* **תיאור כללי:** חשבון הבנק מאפשר לאנשים ליצור חשבון בנק על שמם (או על שם לידיהם) , ולעקוב שם אחרי ההוצאות וההכנסות שלהם בצורה מסודרת.
* **בעיה שהפרויקט פותר:** הרבה אנשים מעוניינים לעקוב אחרי דמי הכיס של ילדיהם, ואין להם איך לעשות זאת בצורה מסודרת, אתר זה יאפשר להם לכתוב כמה יש לילד ב"בנק" וכך הוא יוכל להשתמש בו כשירצה וההורים יורידו מהחשבון. (זה יכול להתאים לעוד אנשים ולא רק להורים, לדוגמא לאנשים שרוצים לחסוך ולראות ויזואלית כמה הם מבזבזים, אך הפרויקט בנוי בעיקר להורים במחשבה ששם יהיה השימוש הנפוץ ביותר.)
* **קהל יעד:** בעיקר הורים, אך יכול להתאים לעוד אנשים שמעוניינים לעקוב אחרי ההוצאות שלהם בצורה מסודרת.

**דרישות מערכת:**

דרישות פונקציונליות:

* המשתמש יכול להירשם ולהתחבר למערכת.
* המשתמש יכול להכניס כמה כסף הוא מכניס או מוציא.
* המערכת מציגה את ההכנסות וההוצאות של המשתמש ומראה לו כמה כסף יש לו כרגע בבנק.

דרישות טכניות:

* **Frontend:** HTML, CSS, JavaScript
* **Backend:** Node.js, Express
* **Database**: MongoDB
* **Session Management:** express-session

## תכנון ומבנה האפליקציה

**מבנה תיקיית הפרויקט:**

מבנה תיקיות:

/Bankaccount

I — / images #קבצי תמונות

I—/ node\_modules #הספריות שהותקנו של node.js

I—/ public # קבצי html

I—/ server//db #קבצי צד שרת

I—/style #קבצי css

**הסבר על כל תיקיה:**

📁 images - מכילה תמונות סטטיות.

📁 node\_modules - ספרייה פנימית של Node.js שמכילה את כל הספריות שהותקנו.

📁 public – דפי HTML שמוצגים למשתמש.

📁 server/db - מסד נתונים וקבצים לטיפול בנתונים בצד שרת.

📁 style - קבצי עיצוב CSS.

**תרשים ניווט בין המסכים:**

מסכים עיקריים:

* עמוד בית- מסביר על מה האתר ומה אפשר לעשות בו.
* עמוד הרשמה - מאפשר הרשמה לאתר.
* עמוד התחברות - מאפשר התחברות לאתר.
* עמוד חשבון אישי- מציג את חשבון הבנק של המשתמש.

**תרשים זרימה של הניווט:**

[המשתמש נכנס לאתר]

↓

[עמוד הבית]

↓

[האם למשתמש יש חשבון?]

↓ ↓

[כן] [לא]

↓ ↓

[התחברות] [רישום משתמש]

↓ ↓

[האם ההתחברות הצליחה?]

↓ ↓

[כן] [לא]

↓ ↓

[עמוד החשבון] [הודעת שגיאה]

↓

[אפשרויות פעולה:]

- הוספת כסף

- הורדת כסף

- צפייה בפעולות קודמות

↓

[שמירה במסד הנתונים]

↓

[המשתמש ממשיך / מתנתק]

## 

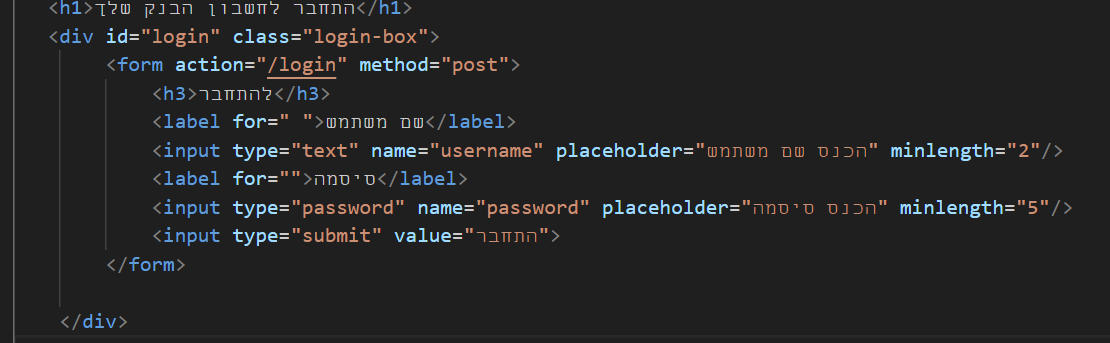
## פיתוח צד הלקוח

**דפי HTML**

דפים עיקריים:

* מציג את עמוד הבית -index.html
* מאפשר הירשמות לאתר- signUp.html
* מאפשר התחברות לאתר- logIn.html
* מציג את חשבון הבנק האישי- account.html

קטע קוד לדף ההתחברות:



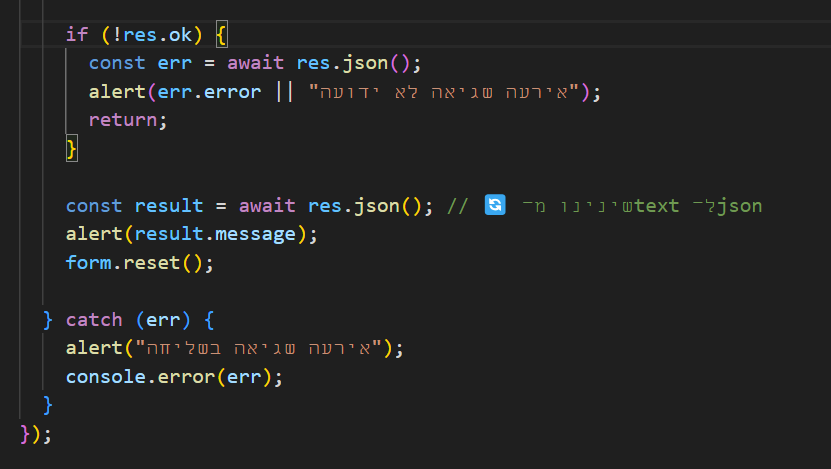
הסבר מפורט על כל רכיב בקוד:

* <form> – יוצר טופס לקליטת נתונים מהמשתמש ושליחתם לשרת.
* <label> – תווית שמסבירה למשתמש מה להזין בכל שדה קלט.
* <input type="text"> – שדה קלט לטקסט רגיל, למשל שם משתמש.
* <input type="password"> – שדה קלט שמסתיר את הטקסט (לסיסמה).
* placeholder – טקסט שמוצג בתוך שדה הקלט עד שמתחילים להקליד.
* minlength – מגדיר את מספר התווים המינימלי שהמשתמש חייב להקליד.
* <input type="submit"> – כפתור ששולח את הטופס.
* name – מזהה את שדה הקלט לצורך שליחה ועיבוד הנתונים בשרת.
* <h1>, <h3> – כותרות לדף ולטופס, מסבירות מה מטרת הדף.
* <div> – בלוק עיצובי שמאגד את הטופס (משמש לסידור ועיצוב ב־CSS).

**קוד (javaScript Fronted)**

קטע קוד לטיפול בטופס צור קשר:





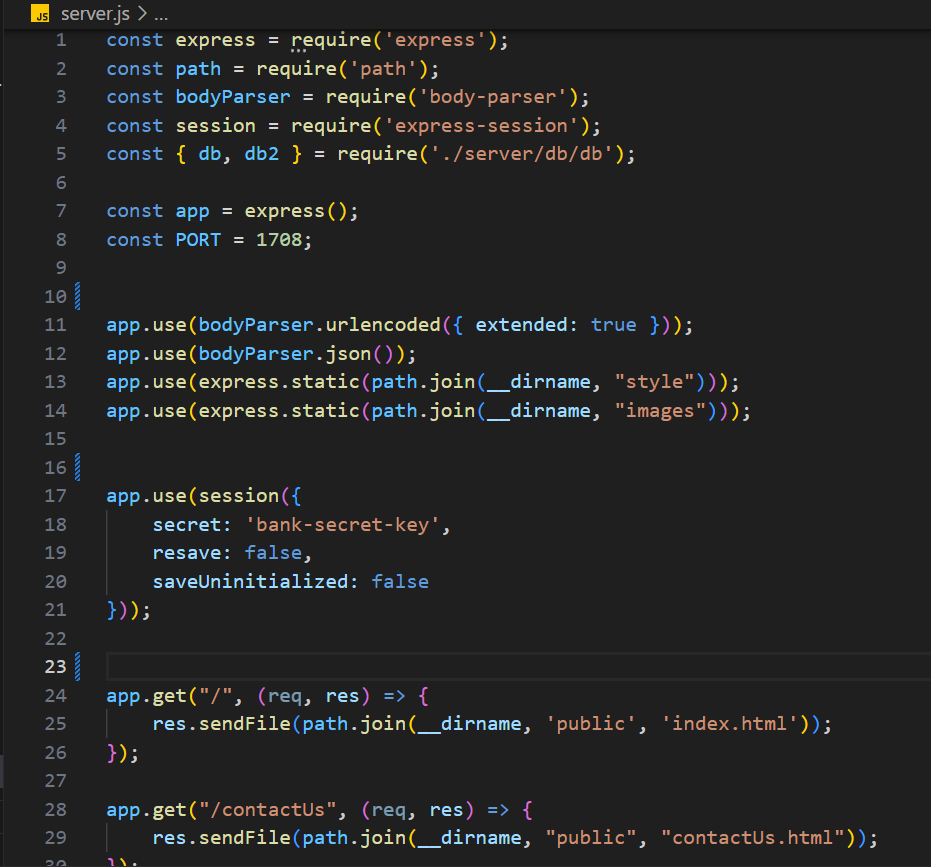
**הסבר על הקוד**

* document.getElementById('contactForm').addEventListener('submit', ...) – מאזין לשליחת טופס.
* e.preventDefault() – מונע ריענון עמוד ברירת מחדל.
* אוסף ערכי שדות מהטופס (firstName, secondName, eMail, message).
* שולח בקשת POST לשרת בכתובת /contactUs עם הנתונים כ־JSON.
* בודק אם התגובה מהשרת לא תקינה ומציג התראה.
* אם התגובה תקינה, מציג הודעת הצלחה ומנקה את הטופס.
* בודק שגיאות תקשורת ומציג התראה במקרה של כשל.

## פיתוח צד שרת

**יצירת שרת (Node.js)**

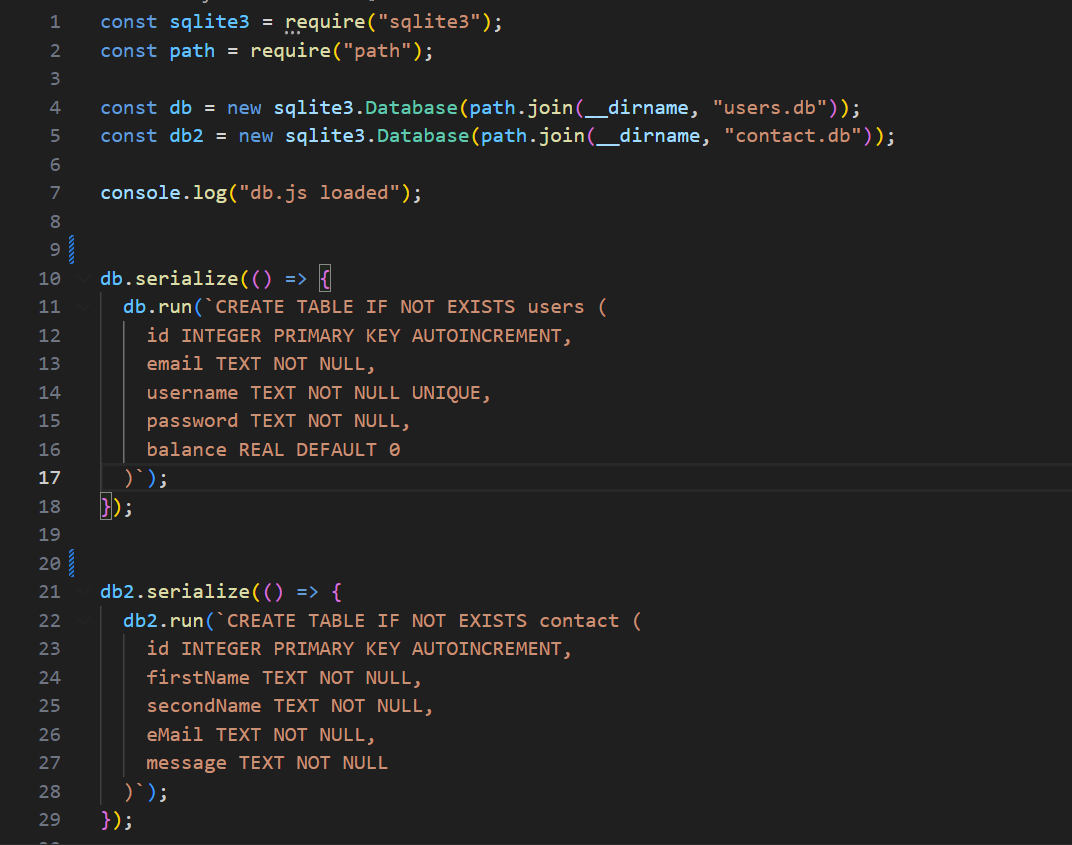
קטע קוד ליצירת שרת (server.js) באמצעות Express



## תכנון מסד הנתונים

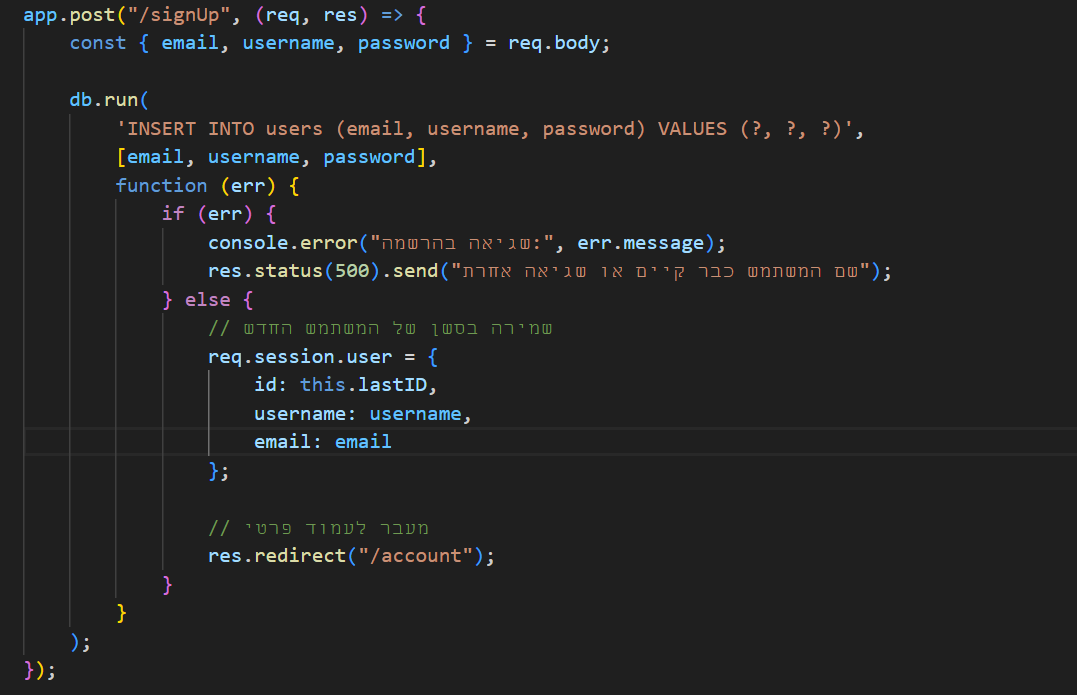
**מבנה מסד הנתונים:**

קטע קוד למסד נתונים:

* const path = require("path") - מאפשר לבנות נתיב קבצים באופן תקני בין מערכות הפעלה.
* new sqlite3.Database(...) - פותח (או יוצר) שני קבצי מסד נתונים: אחד ל־users, השני ל־contact.
* db.serialize(...).run(...) - יוצר טבלת users אם לא קיימת, עם עמודות: מזהה, אימייל, שם משתמש, סיסמה, יתרה.
* db2.serialize(...).run(...) - יוצר טבלת contact אם לא קיימת, עם עמודות: שם פרטי, שם משפחה, אימייל, הודעה.

## 

## בדיקת אבטחה ומידע

בדיקת טפסים:

* app.post("/signUp", ...) - מטפל בבקשה של הרשמה חדשה מהלקוח.
* const { email, username, password } = req.body - שולה את הנתונים מהטופס שנשלח.
* db.run(...) - מנסה להוסיף משתמש חדש למסד הנתונים (users).
* אם יש שגיאה (למשל שם משתמש שכבר קיים), מחזיר שגיאה ללקוח (res.status(500)).
* אם ההרשמה הצליחה - שומר את פרטי המשתמש ב־session.
* ואז מעביר את המשתמש לדף החשבון (/account).