# **תרגיל מתגלגל – React + MUI**

## 

## **פרק 1 – הקמת פרויקט**

## **מה נלמד ונתרגל:**

## **מה זה React ולמה עובדים איתו.**

## **איך מקימים פרויקט חדש עם Vite.**

## **איך מתקינים ספריות צד שלישי (MUI).**

## **איך מוסיפים קומפוננטה בסיסית למסך.**

## **המשימה:**

## **צור פרויקט חדש עם Vite (npm create vite@latest my-app --template react).**

## **התקן Material UI (@mui/material @emotion/react @emotion/styled).**

## **הצג במסך כותרת (Typography של MUI) עם טקסט: *"ברוכים הבאים לאפליקציית המשימות שלי"*.**

## 

## 

## 

## 

## **פרק 2 – קומפוננטות ו־Props**

## **מה נלמד ונתרגל:**

## **יצירת קומפוננטה חדשה.**

## **שימוש ב־props להעברת מידע בין קומפוננטות.**

## **שימוש ברכיבי MUI כמו Card.**

## **המשימה:**

## **צור קומפוננטה TaskCard שתציג כרטיס (Card של MUI) עם שם משימה ותיאור.**

## **העבר את שם המשימה ותיאורה ל־TaskCard באמצעות props.**

## **במסך הראשי הצג שני כרטיסי משימות לדוגמה.**

## 

## **פרק 3 – State ו־Hooks בסיסיים**

## **מה נלמד ונתרגל:**

## **מה זה state ולמה הוא חשוב ב־React.**

## **שימוש ב־useState.**

## **ניהול רשימת משימות פשוטה.**

## **המשימה:**

## **הוסף כפתור (Button) עם טקסט: *"הוסף משימה"*.**

## **צור state (useState) שמחזיק מערך משימות.**

## **בלחיצה על הכפתור – תתווסף משימה ברירת מחדל ("משימה חדשה").**

## 

## **פרק 4 – תנאים ולולאות ברינדור**

## **מה נלמד ונתרגל:**

## **הצגה מותנית (Conditional Rendering).**

## **שימוש ב־map ליצירת קומפוננטות מתוך מערך.**

## **שימוש ב־key לכל אלמנט.**

## **המשימה:**

## **אם אין משימות – הצג טקסט: *"אין משימות כרגע"*.**

## **אם יש משימות – הצג אותן בלולאת map עם שימוש ב־TaskCard.**

## 

## **פרק 5 – useEffect וחיי קומפוננטה**

## **מה נלמד ונתרגל:**

## **מה זה useEffect.**

## **איך מגיבים לאירועים שמתרחשים ב־Lifecycle של קומפוננטה.**

## **איך מנקים משאבים (cleanup).**

## **המשימה:**

## **הצג הודעת ברוך הבא (Snackbar של MUI) כאשר האפליקציה נטענת.**

## **ההודעה תיעלם אוטומטית אחרי 3 שניות.**

## 

## **פרק 6 – טפסים וניהול קלט**

## **מה נלמד ונתרגל:**

## **Controlled Components – קלט שמנוהל ב־state.**

## **שימוש ב־TextField של MUI.**

## **איך מוסיפים מידע חדש לרשימת משימות.**

## **המשימה:**

## **צור טופס עם TextField + Button ליצירת משימה חדשה.**

## **כאשר המשתמש מקליד שם משימה ולוחץ "הוסף" – היא תתווסף לרשימת המשימות.**

## **לאחר ההוספה – השדה יתנקה.**

## 

## **פרק 7 – עיצוב ו־Styling**

## **מה נלמד ונתרגל:**

## **איך להשתמש ב־Theme מותאם אישית ב־MUI.**

## **שימוש ב־AppBar ו־Toolbar לעיצוב ראשי.**

## **Grid Layout לרספונסיביות.**

## **המשימה:**

## **הגדר Theme מותאם עם צבע ראשי וצבע משני.**

## **הוסף AppBar עם כותרת האפליקציה.**

## **הצג את המשימות ב־Grid כך שיתאימו למסכים שונים.**

## 

## **פרק 8 – מבני נתונים ומיפוי**

## **מה נלמד ונתרגל:**

## **ניהול אובייקטים מורכבים ב־state.**

## **הצגת מידע נוסף בכרטיס.**

## **סימון משימה כבוצעה.**

## **המשימה:**

## **עדכן את מבנה המשימה כך שיכלול: { id: number, title: string, description: string, done: boolean }**

## 

## **הוסף Checkbox לכל משימה.**

## **אם המשימה סומנה כבוצעה – הכותרת שלה תופיע עם קו חוצה (textDecoration: line-through).**

## 

## 

## 

## **פרק 9 – ניהול State מתקדם (Context)**

## **מה נלמד ונתרגל:**

## **בעיית Prop Drilling.**

## **יצירת Context לניהול גלובלי של state.**

## **שימוש ב־useContext.**

## **המשימה:**

## **צור Context שמכיל את רשימת המשימות ואת הפונקציות עדכון, מחיקה, הוספה.**

## **גרום ל־TaskCard ו־TaskForm להשתמש ב־Context במקום ב־props.**

## **דאגו לשמור ולטעון את הנתונים ל localstorage**

## 

## **פרק 10 – Routing**

## **מה נלמד ונתרגל:**

## **התקנה ושימוש ב־react-router-dom.**

## **יצירת עמודים מרובים.**

## **ניווט בין עמודים.**

## **המשימה:**

## **צור שני עמודים:**

## **Home – הצגת המשימות.**

## **About – מידע על האפליקציה.**

## **הוסף כפתורי ניווט (Link או Button של MUI) למעבר בין העמודים.**

# **פרק 11 – Firebase**

## **מה נלמד ונתרגל**

## **מה זה Firebase ולמה משתמשים בו (Backend-as-a-Service).**

## **איך להקים פרויקט Firebase חדש.**

## **איך לחבר אפליקציית React ל־Firebase.**

## **איך לשמור ולקרוא נתונים מ־Firestore (מסד נתונים NoSQL של Firebase).**

## **איך לעדכן ולמחוק משימות ב־Firestore.**

## **איך לנהל מצבי טעינה ושגיאה באפליקציה.**

## 

## **שלבי הקמה (הסבר מפורט)**

## **יצירת פרויקט Firebase**

## **נכנסים ל־Firebase Console.**

## **לוחצים "Add project" → נותנים שם לפרויקט (למשל react-tasks).**

## **בוחרים הגדרות ברירת מחדל (ללא Google Analytics כרגע).**

## **חיבור ל־React**

## **מתקינים את ספריית Firebase: npm install firebase**

## 

## **יוצרים קובץ חדש בפרויקט: firebaseConfig.ts או firebase.js.**

## **מעתיקים לשם את קטע הקוד ש־Firebase נותן לנו (SDK setup).**

## **יוצרים משתנה db שמחבר ל־Firestore: import { initializeApp } from "firebase/app";**

## **import { getFirestore } from "firebase/firestore";**

## 

## **const firebaseConfig = {**

## **apiKey: "XXXXX",**

## **authDomain: "XXXXX.firebaseapp.com",**

## **projectId: "XXXXX",**

## **storageBucket: "XXXXX.appspot.com",**

## **messagingSenderId: "XXXXX",**

## **appId: "XXXXX"**

## **};**

## 

## **const app = initializeApp(firebaseConfig);**

## **export const db = getFirestore(app);**

## 

## **הגדרות Firestore**

## **ב־Firebase Console נכנסים ל־Firestore Database.**

## **לוחצים *Create Database* → בוחרים מצב *Test mode* (כדי לאפשר קריאות/כתיבות חופשיות בזמן הפיתוח).**

## **יוצרים אוסף חדש (Collection) בשם tasks.**

## 

## **משימות מעשיות**

## **שליפת נתונים**

## **בעת טעינת האפליקציה, שלוף את המשימות מאוסף tasks.**

## **הצג אותן בטבלה / cards.**

## **השתמש ב־useEffect ובפונקציות של Firestore: import { collection, getDocs } from "firebase/firestore";**

## **import { db } from "./firebaseConfig";**

## 

## **async function loadTasks() {**

## **const querySnapshot = await getDocs(collection(db, "tasks"));**

## **querySnapshot.forEach((doc) => {**

## **console.log(doc.id, " => ", doc.data());**

## **});**

## **}**

## 

## **הוספת משימה חדשה**

## **כאשר המשתמש לוחץ על *הוסף משימה*, המשימה תישמר גם ב־Firestore (addDoc).**

## **import { collection, addDoc } from "firebase/firestore";**

## 

## **await addDoc(collection(db, "tasks"), {**

## **title: "משימה חדשה",**

## **description: "נוספה מפיירבייס",**

## **done: false**

## **});**

## 

## **עדכון משימה**

## **כאשר המשתמש מסמן משימה כבוצעה → עדכן אותה ב־Firestore (updateDoc).**

## **מחיקת משימה**

## **כאשר המשתמש לוחץ "מחק" → הסר את המסמך מה־Firestore (deleteDoc).**

## **מצבי טעינה ושגיאה**

## **בזמן שליפה – הצג *Spinner* או *CircularProgress* של MUI.**

## **אם יש שגיאה – הצג Alert עם טקסט מתאים.**

## 

## **התוצאה**

## **בסיום פרק זה תהיה לכם אפליקציה עם:**

## **ממשק משתמש יפה עם MUI.**

## **ניהול State בעזרת React ו־Context.**

## **Routing בין דפים.**

## **שמירה ועדכון משימות ב־Firebase Firestore בענן.**

## 

## 

## **✅ פרומפט: אפליקציית React מודרנית עם Vite, MUI, Firebase ו־React Router**

אני בונה אפליקציית **React מודרנית** באמצעות **Vite**, בשילוב:

* **Material UI (MUI)** לעיצוב,
* **React Router** לניווט בין עמודים,
* ו־**Firebase Authentication** להרשמה והתחברות.

אני רוצה ליצור כל שלב בנפרד לבדוק שעובד שהקוד בסדר ואז להמשיך. אל תכתוב קוד. אשר שהבנת .

אני רוצה ליצור כל שלב בנפרד לבדוק שעובד שהקוד בסדר ואז להמשיך. אל תכתוב קוד. אשר שהבנת .

### **📁 מבנה הקבצים כבר קיים:**

bash

CopyEdit

/src

├── /components

│ ├── TopBar.jsx

│ └── SideDrawer.jsx

├── /pages

│ ├── Login.jsx

│ ├── Register.jsx

│ ├── Profile.jsx

│ ├── Prices.jsx

│ ├── Bills.jsx

│ └── AskAgent.jsx

├── /api

│ └── api.js

├── firebase.js

├── App.jsx

└── main.jsx

.env

## **🎯 דרישות הפיתוח:**

### **1. TopBar.jsx – בר עליון:**

* השתמש ב־MUI AppBar.
* בצד **שמאל**: שם האתר (למשל “MyApp”).
* בצד **ימין**: שני כפתורים:  
  + Login → מקשר ל־/login
  + Register → מקשר ל־/register

### **2. SideDrawer.jsx – מגירה צדדית:**

* נפתחת מצד שמאל בלחיצה על ☰ ב־TopBar.
* מכילה קישורים לנתיבים:  
  + /profile → Profile
  + /prices → Prices
  + /bills → Bills
  + /ask-agent → Ask Agent

### **3. App.jsx:**

* השתמש ב־react-router-dom (Routes, Route).
* שלוט במצב פתיחה/סגירה של המגירה עם useState.
* כל דף יופיע דרך ראוטינג מלא:

jsx

CopyEdit

<Routes>

<Route path="/login" element={<Login />} />

...

</Routes>

### **4. עמודים זמניים (/pages/\*.jsx):**

בכל עמוד תכתוב:

jsx

CopyEdit

<Typography variant="h4">שם העמוד</Typography>

## **🔐 5. Firebase (קובץ firebase.js):**

* אתחל את Firebase באמצעות משתני סביבה מתוך .env
* ייבא והגדר:

js

CopyEdit

import { initializeApp } from "firebase/app";

import { getAuth } from "firebase/auth";

const app = initializeApp(firebaseConfig);

export const auth = getAuth(app);

## **📡 6. api.js:**

* כתוב פונקציות:

js

CopyEdit

export async function signIn(email, password) { ... }

export async function signUp(email, password) { ... }

* השתמש ב־signInWithEmailAndPassword ו־createUserWithEmailAndPassword
* יבא את auth מתוך firebase.js

## **🧾 7. Login.jsx ו־Register.jsx:**

* צור טופס אימייל וסיסמה (ב־MUI).
* בעת שליחה:  
  + קרא לפונקציה המתאימה מ־api.js.
  + אם הצליח → נווט ל־/profile
  + אם נכשל → הצג <Alert> של MUI עם הודעת שגיאה

## **🎨 8. עיצוב כללי:**

* השתמש ב־Material UI בלבד
* העיצוב צריך להיות **נקי**, **רספונסיבי** ו־**קל להרחבה**
* אין להשתמש ב־TypeScript

## **🎨 9. דאג לארכיטקטורה context**

מצוין! הנה **6 רעיונות לשיעורי בית מתקדמים**, כל אחד מהם בנוי כשלב עצמאי, קל לבדיקה, ומעמיק את היכולות מעבר לפרויקט הבסיס:

### **🧠 שלב 1 – שימוש ב־Context לניהול משתמש מחובר**

**מטרה**: לנהל את מצב המשתמש (logged in / not) ברחבי האפליקציה

#### **דרישות:**

* צור UserContext.jsx בתיקיית context/
* אחסן שם את המשתמש המחובר (Firebase user)
* השתמש ב־onAuthStateChanged() ב־useEffect כדי לזהות התחברות/ניתוק
* צור useUser() Hook לצורך שליפה נוחה
* הצג את שם המשתמש ב־TopBar אם מחובר

### **🔐 שלב 2 – הגנה על ראוטים פרטיים**

**מטרה**: לא לאפשר גישה לדפים כמו /profile אם המשתמש לא מחובר

#### **דרישות:**

* צור קומפוננטת <PrivateRoute> (HOC או רכיב עוטף)
* אם אין משתמש מחובר, נווט ל־/login
* עטוף את הראוטים /profile, /prices, /bills, /ask-agent עם <PrivateRoute>

### **📥 שלב 3 – שמירת פרטי פרופיל ב־Firestore**

**מטרה**: לאחסן מידע נוסף על המשתמש

#### **דרישות:**

* לאחר הרשמה, הוסף מסמך חדש ל־Firestore ב־collection בשם users
* שמור: uid, email, dateJoined
* טען את הנתונים ב־Profile.jsx והצג אותם

### **📷 שלב 4 – העלאת תמונת פרופיל ל־Firebase Storage**

**מטרה**: עבודה עם Firebase Storage

#### **דרישות:**

* טופס ב־Profile.jsx שמאפשר להעלות קובץ תמונה
* שמור את התמונה תחת users/{uid}/profile.jpg
* שלוף את ה־URL והצג ב־Avatar ב־TopBar

### **✅ שלב 5 – יצירת טופס "Ask Agent" ושמירה במסד נתונים**

**מטרה**: אימות טפסים + כתיבה ל־Firestore

#### **דרישות:**

* ב־AskAgent.jsx: טופס עם שם, אימייל, שאלה
* ולידציה עם MUI או Yup
* בעת שליחה, שמור ל־collection inquiries ב־Firestore
* הצג הודעת הצלחה או שגיאה

### **🔄 שלב 6 – ניתוק משתמש**

**מטרה**: מימוש Logout מלא

#### **דרישות:**

* ב־TopBar, אם המשתמש מחובר – הצג כפתור Logout
* בלחיצה, בצע signOut(auth)
* נקה את ה־UserContext
* נווט ל־/login

### **💡 איך אפשר להגיש?**

לכל שלב תן לתלמידים להעלות:

* קישור ל־GitHub
* סרטון קצר שמראה כל פיצ׳ר עובד
* תיעוד קצר (README) עם תיאור הפיצ׳ר