## 

## 

## **ראשי פרקים – מערך שיעור לוגאין ורגיסטר עם Firebase**

1. **פתיחה והצגת הנושא**
   * מה זה Firebase Authentication ולמה משתמשים בו
   * דוגמאות יומיומיות של מערכות לוגאין
2. **היכרות עם Firebase**
   * יצירת פרויקט בפיירבייס
   * סקירה קצרה של ה־Firebase Console
   * הפעלת Firebase Authentication
3. **שיטות Authentication בפיירבייס**
   * אימייל + סיסמה
   * Google / Facebook / Providers אחרים (הצגת אפשרות, התמקדות בסיסמא)
4. **הגדרת פרויקט בצד הלקוח**
   * התקנת SDK / ספרייה של Firebase
   * חיבור האפליקציה (Web / Mobile) לפרויקט Firebase
5. **Register – הרשמת משתמש חדש**
   * שימוש ב־createUserWithEmailAndPassword
   * טיפול בשגיאות (סיסמה קצרה, מייל קיים)
   * שמירת פרטי משתמש נוספים במסד (Firestore)
6. **Login – התחברות משתמש קיים**
   * שימוש ב־signInWithEmailAndPassword
   * טיפול בשגיאות (משתמש לא קיים, סיסמה שגויה)
   * שמירת Session / LocalStorage
7. **Logout – התנתקות משתמש**
   * שימוש ב־signOut
   * עדכון ממשק המשתמש
8. **ניהול מצב משתמש (User State Management)**
   * שימוש ב־onAuthStateChanged
   * הצגת תוכן בהתאם למצב (מחובר/לא מחובר)
9. **אבטחה ובונוס**
   * חשיבות של שמירת סיסמאות רק ע"י Firebase
   * כללי אבטחה בסיסיים
   * דגש על שימוש ב־Firebase Rules
10. **סיכום ותרגול**

* תרגיל: לבנות טופס Register + Login עם Firebase
* הרחבה: הוספת עמוד פרופיל אישי

## **🔹 פתיחה והצגת הנושא**

### **מה זה Firebase Authentication ולמה משתמשים בו?**

* **Firebase Authentication** הוא שירות של Google המאפשר לנו לנהל משתמשים במערכת בצורה פשוטה ובטוחה.
* הוא חוסך מאיתנו את הצורך לבנות מערכת ניהול משתמשים מלאה (שמירת סיסמאות, הצפנה, אימותים וכו’).
* הוא מספק כלים מוכנים לשימוש:  
  + התחברות עם **אימייל + סיסמה**
  + התחברות עם **Google, Facebook, GitHub** ועוד
  + ניהול **Session** (הישארות מחובר)
  + **אבטחה מובנית** (שמירת סיסמאות בצורה מוצפנת, תקני אבטחה בינלאומיים)

👉 בקיצור: Firebase Authentication מאפשר לנו **להתמקד בפיתוח האפליקציה** ולא בלוגיקה מסובכת של אבטחה וניהול משתמשים.

## **🔹 היכרות עם Firebase**

### **מה זה Firebase?**

* **Firebase** הוא פלטפורמה מבית Google המציעה כלים מוכנים לפיתוח אפליקציות Web ו־Mobile.
* השירותים המרכזיים בו:  
  + **Authentication** – לוגאין/רגיסטר וניהול משתמשים.
  + **Firestore Database** – מסד נתונים NoSQL בזמן אמת.
  + **Hosting** – העלאת האתר/האפליקציה לענן.
  + **Cloud Functions** – פונקציות צד שרת ללא צורך בשרת משלך.
  + **Analytics** – ניתוח פעילות משתמשים.

👉 Firebase נותן לנו **פתרון מלא** לאפליקציות מודרניות – גם צד לקוח וגם צד שרת.

### **יצירת פרויקט בפיירבייס**

1. ניגשים לכתובת: https://console.firebase.google.com
2. לוחצים על **Add project** (הוספת פרויקט חדש).
3. נותנים שם לפרויקט (למשל: my-login-app).
4. מסמנים אם רוצים לחבר את הפרויקט ל־Google Analytics (לא חובה, אפשר לדלג).
5. לוחצים על **Create project** ומחכים כמה שניות שהפרויקט ייבנה.

### **סקירה קצרה של ה־Firebase Console**

* **Project Overview** – עמוד הבית של הפרויקט.
* **Authentication** – ניהול משתמשים, הפעלת שיטות לוגאין שונות.
* **Firestore Database** – ניהול מסדי נתונים.
* **Storage** – אחסון קבצים (תמונות, PDF וכו’).
* **Hosting** – ניהול דפי אינטרנט שאתה מעלה.
* **Build / Release & Monitor** – כלים לניהול גרסאות, בדיקות וסטטיסטיקות.

👉 חשוב להראות לתלמידים איך מנווטים בפיירבייס קונסול ולזהות איפה נמצא כל שירות.

### **הפעלת Firebase Authentication**

1. נכנסים ל־**Authentication** בלוח הצד.
2. לוחצים על **Get started**.
3. נפתח מסך עם שיטות אימות → בוחרים **Email/Password**.
4. מפעילים את השיטה בלחיצה על **Enable** → **Save**.

📌 מכאן המערכת יודעת לאפשר הרשמה/התחברות עם מייל + סיסמה.

## **🔹 שיטות Authentication בפיירבייס**

### **1. אימייל + סיסמה (Email & Password)**

* **הכי בסיסי ונפוץ** – המשתמש יוצר חשבון עם אימייל וסיסמה.
* המערכת שומרת את הסיסמה **מוצפנת** בתוך Firebase (אנחנו לא רואים אותה).
* פקודות עיקריות:  
  + **createUserWithEmailAndPassword**() → רישום משתמש חדש.
  + signInWithEmailAndPassword() → התחברות עם חשבון קיים.
  + sendPasswordResetEmail() → שחזור סיסמה במקרה של שכחה.

📌 זוהי הדרך שבה נתחיל, כי היא הפשוטה ביותר ומתאימה כמעט לכל אפליקציה.

### **2. Google Sign-In**

* התחברות ישירה עם חשבון Google (כמו Gmail).
* יתרונות:  
  + נוח מאוד למשתמשים – אין צורך לזכור עוד סיסמה.
  + יותר מאובטח (Google מטפלת באימות).
* שימושי במיוחד באפליקציות שבהן רוצים **חוויית שימוש מהירה**.

### **3. Facebook / GitHub / Apple / Providers אחרים**

* Firebase מאפשרת גם:  
  + Facebook Login
  + GitHub Login
  + Apple ID Login
  + ועוד עשרות Providers
* היתרון → משתמש נכנס עם חשבון קיים, בלי לפתוח חשבון חדש.
* החיסרון → דורש קונפיגורציה מול אותו Provider (קצת יותר מורכב למתחילים).

### **איך זה נראה בפועל (Flow כללי)**

1. המשתמש בוחר שיטת התחברות (מייל/Google/Facebook).
2. Firebase מטפלת באימות (מול בסיס נתונים או מול Google/Facebook).
3. אם האימות הצליח → Firebase מחזירה **אובייקט משתמש (User)** עם מידע בסיסי (email, uid, displayName).
4. האפליקציה שלנו מקבלת את האובייקט → מזהה שהמשתמש מחובר → מציגה תוכן פרטי.

💡 לסיכום השיעור:

* יש **הרבה שיטות Authentication**, אבל אנחנו מתמקדים קודם כל ב־**Email + Password**, כי היא הליבה והכי פשוטה להבנה.
* אחרי שהתלמידים מבינים את זה, אפשר להרחיב גם ל־Google / Facebook.

## **🔹 הגדרת פרויקט בצד הלקוח**

### **1. התקנת SDK / ספרייה של Firebase**

#### **לפרויקט Web (JavaScript):**

* **נכנסים לתיקייה של הפרויקט (למשל my-app).**
* **מתקינים את Firebase SDK עם npm:**

**npm install firebase**

**או אם עובדים עם <script> רגיל (ללא npm):**

**<script type="module">**

**import { initializeApp } from "https://www.gstatic.com/firebasejs/11.0.1/firebase-app.js";**

**import { getAuth } from "https://www.gstatic.com/firebasejs/11.0.1/firebase-auth.js";**

**</script>**

#### **לפרויקט Mobile (React Native / Android / iOS):**

* **ב־React Native:**

**npm install firebase**

* **ב־Android (Kotlin/Java): מוסיפים dependencies בקובץ build.gradle.**
* **ב־iOS (Swift): מוסיפים דרך CocoaPods.**

**(בשיעור אפשר להתמקד קודם כל ב־Web, כדי שזה יהיה פשוט וברור).**

### **2. חיבור האפליקציה לפרויקט Firebase**

**לאחר שיצרנו פרויקט בפיירבייס:**

1. **נכנסים ל־Project Overview → לוחצים על כפתור Add app.**
2. **בוחרים את הפלטפורמה:**
   * **Web (</>)**
   * **Android**
   * **iOS**
3. **נותנים שם לאפליקציה (למשל login-demo).**
4. **Firebase מציג לנו קטע קוד עם פרטי החיבור (config):**

**// Import SDKs שהתקנו קודם**

**import { initializeApp } from "firebase/app";**

**import { getAuth } from "firebase/auth";**

**// 🔹 קוד חיבור לפרויקט**

**const firebaseConfig = {**

**apiKey: "AIzaSyD7-XXXXXXXXXXXXXXX",**

**authDomain: "my-login-app.firebaseapp.com",**

**projectId: "my-login-app",**

**storageBucket: "my-login-app.appspot.com",**

**messagingSenderId: "1234567890",**

**appId: "1:1234567890:web:abc123xyz"**

**};**

**// אתחול החיבור**

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

**const auth = getAuth(app);**

**export { auth };**

**📌 עכשיו האפליקציה שלנו מחוברת לפרויקט Firebase ומוכנה לשימוש ב־Authentication.  
 בשלב הבא אפשר להתחיל לכתוב את הפונקציות של Register ו־Login.**

## **🔹 Register – הרשמת משתמש חדש**

### **1. שימוש ב־createUserWithEmailAndPassword**

**פונקציה מובנית של Firebase שמקבלת Email + Password ויוצרת משתמש חדש.**

**דוגמה בסיסית:**

**import { auth } from "./firebase";**

**import { createUserWithEmailAndPassword } from "firebase/auth";**

**const registerUser = async (email, password) => {**

**try {**

**const userCredential = await createUserWithEmailAndPassword(auth, email, password);**

**// אובייקט המשתמש שנוצר**

**const user = userCredential.user;**

**console.log("משתמש נרשם בהצלחה:", user.uid);**

**} catch (error) {**

**console.error("שגיאה בהרשמה:", error.message);**

**}**

**};**

### **2. טיפול בשגיאות (Errors)**

**Firebase מחזיר קודים שונים בשגיאות – חשוב להציג הודעות מתאימות למשתמש.**

**דוגמה:**

**try {**

**await createUserWithEmailAndPassword(auth, email, password);**

**} catch (error) {**

**switch (error.code) {**

**case "auth/email-already-in-use":**

**alert("האימייל כבר רשום במערכת");**

**break;**

**case "auth/invalid-email":**

**alert("האימייל שהוזן לא תקין");**

**break;**

**case "auth/weak-password":**

**alert("הסיסמה חייבת להיות לפחות באורך 6 תווים");**

**break;**

**default:**

**alert("שגיאה לא ידועה: " + error.message);**

**}**

**}**

### **3. שמירת פרטי משתמש נוספים במסד (Firestore)**

**מעבר לאימייל/סיסמה, לרוב רוצים לשמור שם מלא, טלפון, תאריך הרשמה וכו’.  
 בשביל זה נשתמש ב־Firestore Database:**

**import { db } from "./firebase";**

**import { doc, setDoc } from "firebase/firestore";**

**const saveUserData = async (userId, name, phone) => {**

**await setDoc(doc(db, "users", userId), {**

**name: name,**

**phone: phone,**

**createdAt: new Date()**

**});**

**};**

**שילוב בתוך ההרשמה:**

**try {**

**const userCredential = await createUserWithEmailAndPassword(auth, email, password);**

**const user = userCredential.user;**

**// שמירת פרטי משתמש נוספים במסד**

**await saveUserData(user.uid, name, phone);**

**alert("נרשמת בהצלחה!");**

**} catch (error) {**

**console.error(error);**

**}**

### **📌 סיכום של שלב Register**

1. **יוצרים חשבון עם createUserWithEmailAndPassword.**
2. **מטפלים בשגיאות בצורה ידידותית למשתמש.**
3. **שומרים נתונים נוספים (פרופיל אישי) במסד Firestore.**

## **🔹 Login – התחברות משתמש קיים**

### **1. שימוש ב־signInWithEmailAndPassword**

**זוהי הפונקציה של Firebase שמבצעת אימות למשתמש שכבר נרשם.**

**import { auth } from "./firebase";**

**import { signInWithEmailAndPassword } from "firebase/auth";**

**const loginUser = async (email, password) => {**

**try {**

**const userCredential = await signInWithEmailAndPassword(auth, email, password);**

**const user = userCredential.user;**

**console.log("התחברות הצליחה:", user.uid);**

**return user;**

**} catch (error) {**

**console.error("שגיאה בהתחברות:", error.message);**

**}**

**};**

### **2. טיפול בשגיאות נפוצות**

**Firebase מחזיר קודים שונים – אנחנו נתפוס אותם ונציג הודעות ברורות למשתמש.**

**try {**

**const userCredential = await signInWithEmailAndPassword(auth, email, password);**

**console.log("התחברות הצליחה", userCredential.user);**

**} catch (error) {**

**switch (error.code) {**

**case "auth/user-not-found":**

**alert("המשתמש לא קיים במערכת");**

**break;**

**case "auth/wrong-password":**

**alert("סיסמה שגויה");**

**break;**

**case "auth/invalid-email":**

**alert("האימייל שהוזן לא תקין");**

**break;**

**default:**

**alert("שגיאה לא ידועה: " + error.message);**

**}**

**}**

### **3. שמירת Session / LocalStorage**

**🔹 כברירת מחדל, Firebase שומר session של המשתמש עד לסגירת הדפדפן.  
 אם רוצים לשמור התחברות קבועה (כמו "זכור אותי"), משתמשים ב־Persistence:**

**import { setPersistence, browserLocalPersistence } from "firebase/auth";**

**const loginUser = async (email, password) => {**

**try {**

**await setPersistence(auth, browserLocalPersistence);**

**const userCredential = await signInWithEmailAndPassword(auth, email, password);**

**console.log("התחברות הצליחה:", userCredential.user);**

**} catch (error) {**

**console.error("שגיאה בהתחברות:", error.message);**

**}**

**};**

* **browserLocalPersistence → נשמר ב־LocalStorage (גם אחרי סגירת הדפדפן).**
* **browserSessionPersistence → נשמר רק כל עוד החלון פתוח.**
* **inMemoryPersistence → נשמר בזיכרון בלבד (נעלם מיד בסגירה/רענון).**

### **📌 סיכום שלב Login**

1. **מבצעים התחברות עם signInWithEmailAndPassword.**
2. **מטפלים בשגיאות כמו משתמש לא קיים או סיסמה שגויה.**
3. **מנהלים את ה־Session לפי הצורך (ברירת מחדל / LocalStorage / SessionStorage).**

## **🔹 Logout – התנתקות משתמש**

### **1. שימוש בפונקציה signOut**

**פיירבייס מספקת פונקציה פשוטה לניתוק משתמש מהמערכת:**

**import { auth } from "./firebase";**

**import { signOut } from "firebase/auth";**

**const logoutUser = async () => {**

**try {**

**await signOut(auth);**

**console.log("המשתמש התנתק בהצלחה");**

**alert("התנתקת מהמערכת");**

**} catch (error) {**

**console.error("שגיאה בהתנתקות:", error.message);**

**}**

**};**

### **2. מה קורה אחרי Logout?**

* **ה־Session שנשמר נמחק (בין אם זה ב־LocalStorage או Session).**
* **auth.currentUser הופך ל־null.**
* **אם יש listener של מצב משתמש (onAuthStateChanged) – הוא יתעדכן אוטומטית.  
   למשל:**

**import { onAuthStateChanged } from "firebase/auth";**

**onAuthStateChanged(auth, (user) => {**

**if (user) {**

**console.log("משתמש מחובר:", user.email);**

**} else {**

**console.log("אין משתמש מחובר");**

**}**

**});**

### **3. יישום באפליקציה**

**בממשק משתמש (UI) מקובל לשים כפתור התנתק.**

* **בלחיצה על הכפתור → מפעילים את logoutUser().**
* **המערכת מנתקת את המשתמש ומחזירה אותו למסך Login/Register.**

### **📌 סיכום שלב Logout**

1. **שימוש ב־signOut(auth) לניתוק המשתמש.**
2. **המערכת מעדכנת אוטומטית את מצב המשתמש.**
3. **הממשק חוזר למסך התחברות/הרשמה.**

**💡 עכשיו יש לנו זרימה מלאה:**

* **Register → יצירת משתמש חדש.**
* **Login → התחברות למערכת.**
* **Session → שמירת מצב המשתמש.**
* **Logout → התנתקות וניקוי session.**

## **🔹 ניהול מצב משתמש (User State Management)**

### **1. שימוש ב־onAuthStateChanged**

**פונקציה מובנית של Firebase שמאזינה לשינויים במצב המשתמש:**

**import { auth } from "./firebase";**

**import { onAuthStateChanged } from "firebase/auth";**

**// מאזין לשינויי מצב משתמש**

**onAuthStateChanged(auth, (user) => {**

**if (user) {**

**console.log("מחובר:", user.email);**

**} else {**

**console.log("לא מחובר");**

**}**

**});**

**🔹 מתי היא מופעלת?**

* **אחרי Register מוצלח.**
* **אחרי Login מוצלח.**
* **אחרי Logout.**
* **אחרי רענון הדף (שימוש ב־Session או LocalStorage).**

### **2. הצגת תוכן בהתאם למצב (מחובר/לא מחובר)**

**נרצה שהמערכת תראה דברים שונים לפי מצב המשתמש:**

#### **דוגמה בסיסית ב־HTML + JS**

**<div id="loginSection">**

**<h2>התחברות</h2>**

**<!-- טופס התחברות -->**

**</div>**

**<div id="appSection" style="display:none;">**

**<h2>ברוך הבא!</h2>**

**<button onclick="logoutUser()">התנתק</button>**

**</div>**

**onAuthStateChanged(auth, (user) => {**

**if (user) {**

**document.getElementById("loginSection").style.display = "none";**

**document.getElementById("appSection").style.display = "block";**

**} else {**

**document.getElementById("loginSection").style.display = "block";**

**document.getElementById("appSection").style.display = "none";**

**}**

**});**

### **3. יישום ב־React (אופציונלי למי שמתקדם)**

**נשמור state של המשתמש:**

**import { useEffect, useState } from "react";**

**import { auth } from "./firebase";**

**import { onAuthStateChanged } from "firebase/auth";**

**function App() {**

**const [user, setUser] = useState(null);**

**useEffect(() => {**

**const unsubscribe = onAuthStateChanged(auth, (currentUser) => {**

**setUser(currentUser);**

**});**

**return () => unsubscribe();**

**}, []);**

**return (**

**<div>**

**{user ? (**

**<div>**

**<h2>שלום {user.email}</h2>**

**<button onClick={() => auth.signOut()}>התנתק</button>**

**</div>**

**) : (**

**<h2>אנא התחבר</h2>**

**)}**

**</div>**

**);**

**}**

**export default App;**

### **📌 סיכום שלב ניהול מצב משתמש**

1. **מאזינים עם onAuthStateChanged לכל שינוי במצב המשתמש.**
2. **בהתאם למצב → מציגים מסך Login/Register או את האפליקציה הפנימית.**
3. **המשתמש נשאר מחובר גם אחרי ריענון הדף אם שמרנו Session ב־LocalStorage.**

# **🔹 אבטחה ובונוס ב־Firebase Authentication**

### **1. Firebase Rules – גישה למסד נתונים רק למשתמשים מחוברים**

* **ברירת מחדל: כל אחד יכול לקרוא/לכתוב במסד (לא מאובטח!).**
* **כדי להגן על הנתונים, Firebase מאפשרת להגדיר Security Rules ל־Firestore / Storage.**
* **דוגמה: לאפשר גישה למסד רק למשתמשים מחוברים:**

**{**

**"rules": {**

**"users": {**

**"$uid": {**

**".read": "request.auth != null && request.auth.uid == $uid",**

**".write": "request.auth != null && request.auth.uid == $uid"**

**}**

**}**

**}**

**}**

**🔹 מה זה אומר?**

* **request.auth != null → רק משתמש מחובר רשאי לגשת.**
* **request.auth.uid == $uid → משתמש יכול לקרוא/לכתוב רק את הנתונים של עצמו.**

### **2. אימות דו־שלבי (2FA) ו־Providers נוספים**

#### **אימות דו־שלבי (2FA):**

* **Firebase תומכת באימות עם טלפון (SMS).**
* **המשתמש מתחבר עם מייל/סיסמה → המערכת שולחת קוד לטלפון → המשתמש מזין את הקוד → נכנס.**
* **מעלה רמת אבטחה משמעותית (נפוץ בבנקים / אפליקציות כספים).**

#### **Providers נוספים (Google, Facebook, GitHub, Apple):**

* **יתרון: המשתמש לא צריך לזכור סיסמה חדשה.**
* **קל מאוד להתחיל עם Google Sign-In:**
  + **מפעילים את Google provider ב־Firebase Console.**
  + **משתמשים ב־signInWithPopup(auth, googleProvider).**
* **דוגמה:**

**import { GoogleAuthProvider, signInWithPopup } from "firebase/auth";**

**const provider = new GoogleAuthProvider();**

**const loginWithGoogle = async () => {**

**try {**

**const result = await signInWithPopup(auth, provider);**

**console.log("התחברות עם גוגל הצליחה:", result.user);**

**} catch (error) {**

**console.error("שגיאה בהתחברות עם גוגל:", error.message);**

**}**

**};**

### **3. שמירה על UX טוב**

* **טעינה (Loading): בזמן שהמערכת מתחברת ל־Firebase (Login/Register), מציגים ספינר במקום שהמשתמש יחכה בלי פידבק.**
* **הודעות ברורות:**
  + **במקום לזרוק שגיאה טכנית ("auth/wrong-password") → להציג הודעה אנושית ("הסיסמה שהוזנה שגויה").**
* **שמירה על מצב משתמש: אם המשתמש מחובר → להפנות אותו אוטומטית למסך הבית, בלי לדרוש Login מחדש.**
* **כפתור "זכור אותי" → שימוש ב־browserLocalPersistence כדי לשמור Session גם אחרי סגירת הדפדפן.**

## **📌 סיכום יחידת אבטחה ובונוס**

1. **Firebase Rules – מגנים על מסד הנתונים כך שרק משתמשים מחוברים יכולים לגשת, וכל משתמש ניגש רק לנתונים שלו.**
2. **2FA + Providers – אפשרות להוסיף שכבת אבטחה עם SMS או לאפשר התחברות נוחה עם Google / Facebook.**
3. **UX טוב – טעינה, הודעות ברורות, וזכירת המשתמש לשיפור חווית ההתחברות.**

**💡 עכשיו בעצם יש לנו מערך שיעור שלם שמתחיל מהבנת לוגאין ורגיסטר, עובר דרך Firebase, כולל Register, Login, Logout, ניהול מצב משתמש, ועד אבטחה ו־UX.**