



# הדרכה על Roo Code

ברוכים הבאים להדרכה על התוסף Roo Code לעבודה חכמה בעזרת AI ב-Visual Studio Code

## מטרות ההדרכה:

- ללמוד את יכולות התוסף ולהבין את הפוטנציאל שלו לשיפור תהליכי עבודה
- להבין כיצד לשלב את Roo Code בסביבת העבודה היומיומית
- לראות דוגמאות מעשיות בעבודה עם Angular ו-.NET
- להכיר את היכולות המתקדמות כמו ביצוע Code Review ויצירת Unit Tests

תאריך: 7.7.2025

# מה זה Roo Code?

Roo Code הוא תוסף עוצמתי מבוסס AI ל-VSCode המאפשר להאיץ פיתוח, לבצע סקירות קוד, להריץ טסטים, לאוטומציה של דפדפן ועוד - הכל בשפה טבעית ובאינטגרציה מלאה לסביבת הפיתוח.



## פקודות טרמינל

הרצת פקודות, התקנות חבילות, הפעלת שרת פיתוח או בדיקות - הכל ישירות מהצ'אט.



## ניהול קבצים וקוד

קריאה וכתיבת קבצים ישירות במרחב העבודה, עריכת קוד, ותיקון שגיאות אוטומטי.



## תקשורת בשפה טבעית

שאל שאלות, בקש שינויי קוד או ייצור פרויקטים שלמים בשפה טבעית - בדיוק כמו שהיית מסביר למתכנת אחר.



## תמיכה במודלים מרובים

עבודה עם OpenAI, Anthropic, Google Gemini ואפילו מודלים מקומיים - הבחירה שלך.



## מצבי עבודה מותאמים

התאמת האסיסטנט לתפקידים שונים: מפתח קוד, ארכיטקט מערכת, QA, או מנהל מוצר.



## אוטומציית דפדפן

פתיחת דפדפן לבדיקת האפליקציה, ביצוע קליקים, מילוי טפסים ולכידת שגיאות.

# התקנה וקונפיגורציה



## שלב 1: התקנת התוסף

- חפשו "Roo Code" ב-VS Code Marketplace
- לחצו על "Install" להתקנת התוסף
- או התקינו דרך Open-VSX עבור VSCodium
- לחלופין, הורידו קובץ VSIX והתקינו באופן ידני



## שלב 3: פתיחת Roo Code

- לחצו על האייקון של Roo Code בסרגל הפעילות
- או השתמשו בקיצור: Ctrl+Shift+P ואז "Roo Code: Open"
- מומלץ לפתוח בטאב נפרד: "Roo Code: Open in New Tab"



## שלב 4: הגדרת פרופיל ראשוני

- בחרו מצב עבודה (Code, Architect, Ask)
- הגדירו הרשאות אוטו-אישור לפעולות
- התאימו את הכללים לנוהלי העבודה בארגון
- שמרו תצורות לפרויקטים שונים



## שלב 2: הגדרת מודל AI

- פתחו את הגדרות Roo Code (Settings)
- הוסיפו API Key למודל הרצוי (OpenAI, Anthropic וכו')
- הגדירו את המודל המועדף (GPT-4, Claude וכו')
- ניתן להשתמש במודלים מקומיים כמו Ollama/LM Studio



טיפ:

ניתן להגדיר קבצי clinerules בתיקיית הפרויקט כדי לקבוע הגדרות ספציפיות לפרויקט, כולל הגבלת הרשאות או הנחיות מותאמות אישית למצבי עבודה שונים.

# מצבי עבודה (Chat Modes)

Roo Code מציע מספר מצבי עבודה ייעודיים המאפשרים לך להתאים את סגנון האינטראקציה בהתאם למשימה הנוכחית. לכל מצב יש יכולות והרשאות שונות.



## מצב ארכיטקט (Architect)

מיועד לחשיבה מעמיקה על עיצוב וארכיטקטורת מערכת. מתמקד בתכנון ברמה גבוהה, פתרון בעיות מורכבות ותכנון מערכות. אינו כותב קוד או מריץ פקודות באופן ישיר.

❌ ללא יכולת לכתוב קוד    ✔️ חשיבה ארכיטקטונית



## מצב קוד (Code)

מצב ברירת המחדל המיועד לכתיבה, עריכה ותיקון קוד. מאפשר הרצת פקודות, יצירת קבצים, ואוטומציה מלאה של משימות פיתוח. אידיאלי לעבודת קידוד יומיומית ופיתוח פרויקטים.

✔️ יכולת לכתוב ולערוך קבצים    ✔️ להריץ פקודות טרמינל



## מצבים מותאמים אישית (Custom)

יצירת מצבי עבודה ייחודיים המותאמים למשימות ספציפיות. למשל: מהנדס QA לבדיקות, מנהל מוצר, או מעצב UX. ניתן להגדיר את הפרומפטים וההרשאות בכל מצב.

✔️ התאמה אישית מלאה    ✔️ הגדרת הרשאות גמישה



## מצב שאלות (Ask)

מצב המיועד לחקירה ולמידה. מתמקד במתן תשובות מעמיקות, הסברים על קוד קיים, ועזרה בפתרון בעיות. אידיאלי להבנת מושגים או לניתוח בסיס קוד קיים.

❌ ללא יכולת לכתוב קוד    ✔️ הסברים והדרכה מעמיקה

# יצירת פרויקט Angular לדוגמה

## צעדים ליצירת פרויקט:

- 1

בקשה בצ'אט

שליחת בקשה טבעית ל-Roo Code
- 2

הכנה ואיתחול

Roo Code יוצר פרויקט Angular חדש
- 3

יצירת קומפוננטות

יצירת hello-world ועדכון app.component
- 4

הפעלת האפליקציה

הרצת ng serve ישירות מהצ'אט



ניתן לבקש מ-Roo Code להוסיף יכולות נוספות או לשנות את העיצוב בבקשה הבאה



## המשתמש

צור אפליקציית Angular בסיסית עם קומפוננטה hello-world

## Roo Code

אני אצור עבורך אפליקציית Angular בסיסית עם קומפוננטת hello-world. ראשית אבדוק אם Angular CLI מותקן ואז אתחיל ביצירת הפרויקט.

## \$ התקנת Angular CLI והקמת פרויקט

```
npm install -g @angular/cli $
ng new hello-world-app --routing=false --style=css $
cd hello-world-app $
ng generate component hello-world $
ng serve $
```

## hello-world.component.ts

```
;import { Component } from '@angular/core'

@Component({
  selector: 'app-hello-world',
  templateUrl: './hello-world.component.html',
  styleUrls: ['./hello-world.component.css']
})
export class HelloWorldComponent {
  message: string = 'ברוכים הבאים לקומפוננטת Hello World!'

  constructor() {}

  changeMessage() {
    this.message = 'המסר שונה בהצלחה!'
  }
}
```

# דוגמה: פרויקט .NET חדש

## צעדים ליצירת פרויקט .NET:

- 1 בקשה בצ'אט**  
הגדרת הדרישות בשפה טבעית ל-Roo Code
- 2 יצירת פרויקט**  
Roo Code יוצר שלד עם פקודת dotnet new
- 3 הוספת קונטרולים**  
יצירת קונטרולים והוספת נקודות קצה (Endpoints)
- 4 הרצת האפליקציה**  
dotnet run ובדיקה ב-Swagger



ניתן לבקש מ-Roo Code להוסיף שכבות נוספות כמו שירותים, מודלים וחיבור למסד נתונים



## המשתמש

צור אפליקציית ASP.NET WebApi פשוטה עם קונטרולר ברירת מחדל

## Roo Code

אצור עבורך פרויקט ASP.NET WebApi בסיסי עם קונטרולר. נתחיל ביצירת הפרויקט ואז נוסיף את הקונטרולר.

### \$ יצירת פרויקט ASP.NET WebApi

```
dotnet new webapi -n MyApiProject $
cd MyApiProject $
dotnet add package Swashbuckle.AspNetCore $
dotnet new controller -n Products -o Controllers $
dotnet build $
dotnet run $
```

### ProductsController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace MyApiProject.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ProductsController : ControllerBase
    {
        ()private static readonly List _products = new
        {
            { Price = 99.99m, "מוצר ראשון", Id = 1, Name }, new Product
            { Price = 149.99m, "מוצר שני", Id = 2, Name } new Product
        };

        [HttpGet]
        ()public ActionResult> GetAll
        {
            ;return _products
        }

        [HttpGet("{id}")]
        public ActionResult GetById(int id)
        {
            ;var product = _products.Find(p => p.Id == id)
            ;()if (product == null) return NotFound
            ;return product
        }
    }

    public class Product
    {
        public int Id { get; set; }
        ;"" = public string Name { get; set; }
        public decimal Price { get; set; }
    }
}
```

# ביצוע Code Review עם Roo Code

Roo Code מאפשר לבצע סקירות קוד מקצועיות עם הנחיות וניתוחים חכמים לשיפור האיכות, אבטחה ויעילות הקוד. הוא יכול להפנות לבעיות, להציע פתרונות, ואף ליישם אותם אוטומטית באישורכם.

## איך לבצע Code Review?

### 1 הוסף קונטקסט

השתמש ב-mentions כדי להכניס קבצים לסקירה: `file@`  
`src/app.component.ts` או `git show HEAD~1@`

### 2 בקש ביקורת ספציפית

לדוגמה: "בצע ביקורת קוד על הקבצים האלו, התמקד בבעיות אבטחה ויעילות"

### 3 קבל הערות וסקירה

Roo Code ינתח, יספק הערות וימליץ על תיקונים

### 4 תקן באופן אוטומטי

אפשר ל-Roo Code לתקן בעיות אוטומטית או בצע את התיקונים בעצמך

// דוגמה לסקירת קוד על ידי Roo Code

```
/* Roo Code */
יש בעיית אבטחה בפונקציית executeQuery. היא פגיעה ל-SQL
/* Injection
```

```
function executeQuery(userInput) {
  const query = `SELECT * FROM users WHERE name='${userInput}'`;
  return db.execute(query);
}

function executeQuery(userInput) {
  const query = `SELECT * FROM users WHERE name = ?`;
  return db.execute(query, [userInput]);
}
```

```
/* Roo Code */
השתמשתי בפרמטרים מוכנים כדי למנוע SQL Injection
```

## יתרונות שימוש ב-Roo Code לסקירת קוד:

- איתור בעיות אבטחה, ביצועים ותחזוקה
- הצעת פתרונות אוטומטיים
- למידה מטעויות וקבלת הסברים מפורטים
- תמיכה בריבוי שפות ומסגרות פיתוח
- אינטגרציה מלאה עם סביבת VSCode

# יצירת Unit Tests אוטומטית

## איך Roo Code מייצר טסטים:

### ניתוח הקוד

זיהוי מבנה הקוד, מחלקות, תלויות, ופונקציות שדורשות בדיקה

### זיהוי תבניות בדיקה

התאמת תבניות בדיקה למבנה הקוד ולטכנולוגיה (Angular/Jest או NET/xUnit).

### יצירת מוקים ותמיכה

יצירה אוטומטית של מוקים ותמיכות נדרשות לבדיקה מבודדת

### יצירת מקרי בדיקה

יצירת מקרי בדיקה ספציפיים לכל מחלקה ומתודה עם נתוני דוגמה

### וידוא כיסוי בדיקות

מעקב אחר כיסוי הבדיקות והצעת בדיקות נוספות במידת הצורך

## שילוב מהיר

Roo Code יכול להריץ את הבדיקות באופן אוטומטי ולתקן שגיאות שמתגלות

## המשתמש

לכתוב unit tests עבור ה-Service הזה ב-Angular

## Roo Code

אשמח לכתוב עבורך unit tests ל-Service. אני אבחן את הקוד ואיצור קובץ טסט מתאים עם Jest.

### UserServiceTests.cs (.NET/xUnit)

```
using Xunit;
using Moq;
using System.Threading.Tasks;
using System.Collections.Generic;

public class UserServiceTests
{
    private readonly Mock<IRepository> _mockRepo;
    private readonly UserService _service;

    public UserServiceTests()
    {
        _mockRepo = new Mock<IRepository>();
        _service = new UserService(_mockRepo.Object);
    }

    [Fact]
    public async Task GetAllUsers_ReturnsUsers()
    {
        // Arrange
        var expectedUsers = new List<User>
        {
            new User { Id = 1, Name = "שם 1" },
            new User { Id = 2, Name = "שם 2" }
        };

        _mockRepo.Setup(repo => repo.GetAllAsync())
            .ReturnsAsync(expectedUsers);

        // Act
        var result = await _service.GetAllUsersAsync();

        // Assert
        Assert.Equal(expectedUsers.Count, result.Count);
        Assert.Equal(expectedUsers[0].Name, result[0].Name);
    }
}
```

### user.service.spec.ts (Angular)

```
import { TestBed } from '@angular/core/testing';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { UserService } from '../user.service';

describe('UserService', () => {
    let service: UserService;
    let httpMock: HttpTestingController;

    beforeEach(() => {
        TestBed.configureTestingModule({
            imports: [HttpClientTestingModule],
            providers: [UserService]
        });
        service = TestBed.inject(UserService);
        httpMock = TestBed.inject(HttpTestingController);
    });

    it('should be created', () => {
        expect(service).toBeTruthy();
    });

    it('should retrieve users from API', () => {
        const mockUsers = [
            { id: 1, name: 'שם 1' },
            { id: 2, name: 'שם 2' }
        ];

        service.getUsers().subscribe(users => {
            expect(users).toEqual(mockUsers);
        });

        const req = httpMock.expectOne('api/users');
        expect(req.request.method).toBe('GET');
        req.flush(mockUsers);
    });
});
```

\$ הרצת הטסטים

# להרצת טסטים ב-Angular  
ng test \$

# להרצת טסטים ב-.NET  
dotnet test \$

Made with Genspark



# יכולות מתקדמות: אוטומציות ודפדפן



## Model Context Protocol (MCP)

הוספת כלים מותאמים אישית ל-Roo Code להרחבת היכולות שלו ואינטגרציה עם מערכות וכלים נוספים.

- חיבור לשירותים חיצוניים כמו Jira או AWS
- יצירה אוטומטית של כלים חדשים לפי דרישה
- התאמה לתהליכי עבודה ספציפיים בארגון
- הרחבה קלה של יכולות האסיסטנט



## Browser Automation

Roo Code מאפשר לפתוח ולהפעיל דפדפן ישירות מהצ'אט, ובכך להאיץ תהליכי בדיקות וניפוי באגים.

- פתיחת דפדפן לבדיקת אפליקציה מקומית
- ביצוע פעולות: קליקים, טפסים, ניווט
- לכידת צילומי מסך ושגיאות קונסול
- הפעלת בדיקות end-to-end אוטומטיות

### </> דוגמת שימוש:

```
// פקודה לצ'אט
צור כלי חדש שיאפשר לי לחפש ולקבל נתונים מה-API של Jira ולסנן משימות לפי סטטוס וקדימות.
```

### </> דוגמת שימוש:

```
// פקודה לצ'אט
בדוק את האפליקציה בדפדפן, נווט לעמוד הבית ולחץ על כפתור ההרשמה. צלם מסך של התוצאה והדפס שגיאות קונסול אם יש.
```

## שילוב יכולות מתקדמות



ניתן לשלב את שתי היכולות יחד - למשל, ליצור כלי MCP שמפעיל בדיקות דפדפן אוטומטיות לפי לוח זמנים, או שולף נתונים מ-API חיצוני ומציג אותם בדפדפן.

# Modes מותאמים אישית (Custom Modes)

Custom Modes מאפשרים לך להתאים את Roo Code לתפקידים ומשימות ספציפיות, עם הנחיות, הרשאות וכלים מותאמים לכל מצב עבודה.

## דוגמאות למצבים מותאמים:

### QA Engineer 🛠️

מתמחה בכתיבת טסטים, חיפוש באגים ואיתור מקרי קצה שלא נלקחו בחשבון

### Code Reviewer 🔍

מתמקד בסקירת קוד מעמיקה, איתור בעיות אבטחה ושיפור יעילות הקוד

### UI/UX Designer 🎨

מומחה בעיצוב ממשקים, נגישות ושיפור חוויית משתמש

### Product Manager 📋

עוזר בתכנון מוצרים, כתיבת סיפורי משתמש ותיעדוף יכולות

### טיפ שימושי 💡

תוכל לשתף ולייבא מצבים מותאמים אישית מהקהילה ב-Reddit (r/RooCode) או ליצור קבצי clinerules לשמירת מצבים לשימוש חוזר בפרויקטים שונים.



## איך יוצרים מצב מותאם אישית:

### 1 פתח צ'אט והקלד פקודה

הקלד בצ'אט: 'Create a new mode for [שם התפקיד]'  
לדוגמה: 'Create a new mode for QA Engineer'

### 2 התאם את הפרומפט

עבור לטאב 'Prompts' בתפריט העליון  
התאם את התיאור, ההוראות והפרסונה של המצב

### 3 הגדר הרשאות וכלים

בחר אילו כלים המצב יוכל להשתמש בהם  
הגבל פעולות מסוכנות או הוסף יכולות מיוחדות

### 4 שמור והתחל להשתמש

בחר את המצב החדש מהתפריט הנפתח בתחתית הצ'אט  
הפעל עם מודל AI מתאים לצרכי המשימה

# טיפים ושיטות עבודה מומלצות



## סקירה ידנית לפעולות קריטיות

בצעו בחינה ידנית של קוד שנוגע ללוגיקה עסקית מורכבת או אבטחה. השתמשו ב-Roo Code כמסייע, אך אל תוותרו על הבקרה האנושית בנקודות קריטיות.



## בדיקת אוטומציות בקוד חדש

השתמשו בסביבת פיתוח נפרדת לבדיקת הקוד שנוצר ע"י Roo Code. כך תוכלו לוודא את איכות הפתרונות לפני שילובם בענף הראשי של הפרויקט.



## הגדרות הרשאות מתאימות

התאימו את רמת האוטונומיה של Roo Code לפרויקט. בפרויקטים רגישים הקפידו על אישור ידני של כל פעולה, ובפרויקטים פחות קריטיים אפשר להגדיר auto-approve לפעולות נפוצות.



## שמירת פרומפטים מוצלחים

צרו לעצמכם ספריית פרומפטים יעילים לשימוש חוזר. כאשר מצליחים לנסח בקשה שהביאה לתוצאות מעולות, שמרו אותה לשימושים עתידיים דומים.



## תיעוד באמצעות Roo Code

בקשו מ-Roo Code ליצור תיעוד איכותי תוך כדי העבודה. הוא יכול ליצור README, הסברים על הקוד, ואפילו לסייע בכתיבת דוקומנטציה טכנית או מדריכי משתמש.



## שימוש במספר מצבים

נצלו את היכולת לעבור בין מצבי עבודה שונים: השתמשו במצב Architect לתכנון ראשוני, עברו למצב Code ליישום, וסיימו עם מצב QA מותאם אישית לבדיקות.

## המלצה מיוחדת:

השתמשו בפונקציית file, @folder@ ו-git@ כדי לתת ל-Roo Code הקשר מלא על הפרויקט ולקבל תוצאות מדויקות יותר.

# שאלות? משאבים נוספים והעמקה

## משאבים מומלצים:

**Roo Code GitHub**   
[github.com/qpd-v/Roo-Code](https://github.com/qpd-v/Roo-Code)

**קהילת Reddit**   
[r/RooCode](https://r/RooCode)

**קורסים ומדריכים**   
קורס Udemy: שימוש ב-Roo Code

**דוקומנטציה רשמית**   
[Roo Code Documentation](#)

## דיון ושאלות

- שאלות על התקנה והגדרת מודלי AI
- טיפים לאינטגרציה בסביבת הפיתוח הקיימת
- דיון על יכולות ספציפיות או שימושים מיוחדים
- חוויות משתמשים ודוגמאות מהשטח

 [הרם יד לשאלה](#)

 **בהצלחה!**

המשיכו לחקור את יכולות Roo Code ולשפר את תהליכי הפיתוח שלכם!