

# הנחיות להגשה תוכנה ברמת ה策יניות יתרה לתואר שני במדעי המחשב

## סקירה כללית

מסמך זה מגדיר את הקритריונים להגשה פרויקט תוכנה ברמת策יניות אקדמית המתאימה לסטודנטים策יניים במיוחד לתואר שני במדעי המחשב. הדרישות מתחזקות בפיתוח איקוטי, תיעוד מקיף, וב>Showcases יכולות מחקר ופיתוח ברמה גבוהה. בכל מקום שכתוב פרויקט הכוונה למטרה מסוימת או פרויקט שניתני במסגרת הקורס להגשה.

## 1. מסמכי פרויקט ותוכנו (& Planning)

### 1.1 מסמך דרישות המוצר (PRD - Product Requirements Document)

יש להציג מסמך PRD מפורט ככל:

סקירת הפרויקט והקשר (Project Overview & Background)

- תיאור ברור של מטרת הפרויקט ובוית המשמש שהוא פותח
- ניתוח שוק תחרותי והצבה אסטרטגית
- זיהוי קהל היעד והצדדים המעניינים (stakeholders)

יעדים ומazzi הצלחה (Objectives & Success Metrics)

- הגדרת יעדים מדדיים וברורים
- מדדי KPI לכימות ההשפעה וההצלחה
- קריטריוני קבלה (acceptance criteria)

דרישות פונקציונליות ולא-פונקציונליות (Functional & Non-Functional Requirements)

- רשימת תוכנות עם עדיפות ברורות
- use cases
- דרישות ביצועים, אבטחה, זמינות וscalability

הנחות, תלויות ומגבליות (Assumptions, Dependencies & Constraints)

- זיהוי מערכות חיצונית ותלוית
- מגבליות טכניות וארגוני

- פריטים מחוץ לתחום(out-of-scope items)

ציר זמן ובני דרך(Timeline & Milestones)

- לוח זמנים מפורט עם נקודות ביקורת
- משלוחים צפויים(deliverables) בכל שלב

## 1. מסמך ארכיטקטורה (Architecture Documentation)

תיעוד ארכיטקטוני מקיף הכללי:

תרשיימי בלוקים וזרימה(Block Diagrams & Architecture Diagrams)

- תרשיימי C4 Model (Context, Container, Component, Code)
- תרשיימי UML לתהליכיים מורכבים
- תרשיימי deployment ותשתיות
- ארכיטקטורה תפעולית(operational architecture)

תיעוד החלטות ארכיטקטוניות(ADRs - Architecture Decision Records)

- רצינול להחלטות ארכיטקטוניות מרכזיות
- ניתוח trade-offs ואלטרנטיבות שנשקלו

תיעוד API ו-Interfaces

- תיעוד מפורט של כל ממשק ציבורי
- סכימות נתונים וקונטראקטים

## 2. תיעוד קוד ומבנה פרויקט & Project Structure)

### 2.1 קובץ README מקייף

קובץ README בرمת user manual מלא הכללי:

הוראות התקינה(Installation Instructions)

- דרישות מערכת מפורטות(system requirements)
- הוראות התקינה שלב-אחר-שלב לסביבות שונות
- הגדרת משתני סביבה(environment variables)
- פתרון בעיות נפוצות(troubleshooting)

## Usage Instructions (הוראות הפעלה)

- הוראות הרצה למצבים שונים
- דגלים ואפשרויות CLI או GUI
- Workflowטיפוסי למשתמש

## Examples & Demonstrations (דוגמאות והדוגמאות)

- דוגמאות קוד להרצת צילומי מסך המשמש
- תרחישי שימוש נפוצים
- קישורים לסרטוני הדוגמה (אם רלוונטי)

## Configuration Guide (מדריך תצורה)

- הסבר על קבצי קונפיגורציה
- פרמטרים הנחוצים לכיוול והשעתם

## Contribution Guidelines

- הנחיות לתרומות קוד
- תקני קוד וסגנון

## License & Credits

- רישיון השימוש
- ייחוס לספריותצד ג' ותורמים

## 2.2 מבנה פרויקט מודולרי (Modular Project Structure)

ארגון נכון של מבנה הפרויקט:

### עקרונות ארגון

- חלוקה לתיקיות לפי תפקידי פונקציות ומודולים ברים
- ארגון feature-based או layered architecture
- הפרדה ברורה בין קוד, נתונים, תוכנות ותיעוד

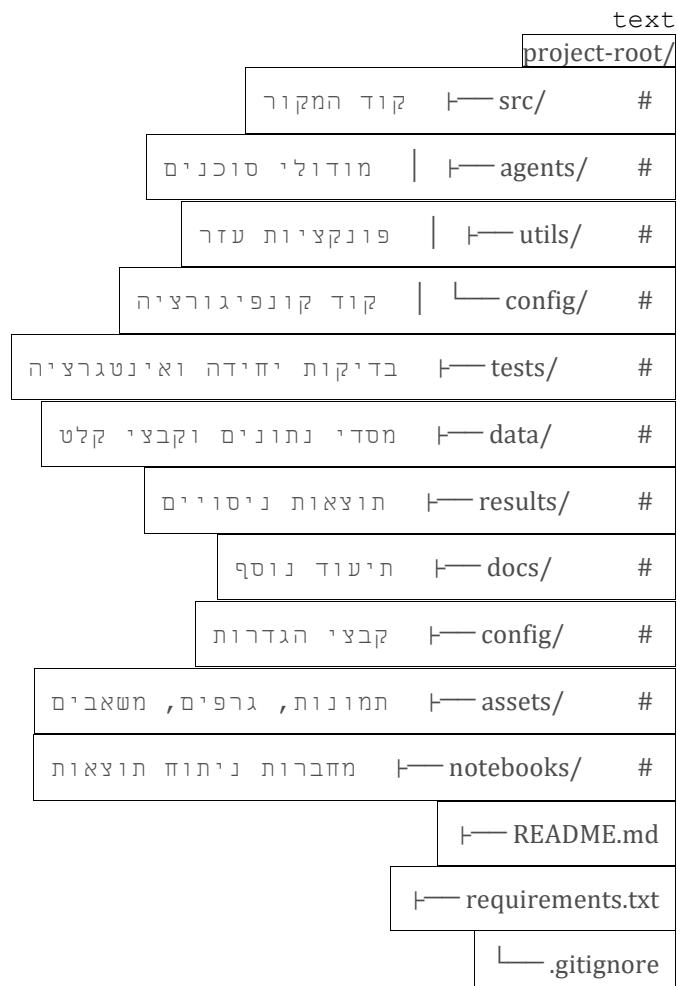
### גודל קבצים

- קבצים לא יעלסו על ~150 שורות קוד
- פירוק לפונקציות ומודולים ברורים (separation of concerns)

## Naming Conventions

- שמות תיקיות וקבצים תיאוריים ועקבאים
- שימוש באותו סגנון naming בכל הפרויקטים

## Structure Example



## 2.3 איקות קוד והערות (Code Quality & Comments)

### תקני הערות בקוד (Code Comments Standards)

- הסבר על ה-"למה" ולא רק ה-"מה" -Docstrings
- הסבר על החלטות עיצוב מורכבות
- תיעוד הנחות ותנאים מוקדמים
- עדכון הערות עם שינויי קוד

### עקרונות כתיבת קוד איקוטי

- שמות משתנים ופונקציות תיאוריים
- פונקציות קצרות ומוקדמות (single responsibility)
- הימנעות מקוד כפול (DRY - Don't Repeat Yourself)
- עקבות בסגנון קוד

## 3. ניהול קונפיגורציה ואבטחת מידע (Configuration Management & Security)

### 3.1. קבצי קונפיגורציה (Configuration Files)

ניהול הגדרות

- שימוש בקבצי קונפיגורציה נפרדים (.env, .yaml, .json)
- הימנעות מקבועים בתחום הקוד (hardcoded values)
- קבצי דוגמה (.env.example) עם ברירות מחדל
- תיעוד כל פרמטר קונפיגורציה

Git Configuration Best Practices

- שימוש ב `.gitignore` - למניעת העלאת קבצי קונפיגורציה רגילים
- קבצי template לkonfiguraciah בגרסאות שונות (dev, staging, production)

### 3.2. אבטחת מידע (Information Security)

הגנה על API Keys וסודות

- אסור לשומר API keys בקוד המקורי
- שימוש במשתני סביבה (environment variables) בלבד
- דוגמאות קוד עם `os.environ.get("API_KEY")`
- הסתרת קבצי `.env` באמצעות `.gitignore`
- שימוש ב `secrets management tools`-כיצור

Rotation & Monitoring

- החלפת מפתחות באופן תקופתי
- ניטור שימוש בkeys API
- הגבלת הרשאות למינימום הנדרש (least privilege)

## 4. בדיקות ואיכות תוכנה (Testing & Quality Assurance)

### 4.1. יחידות בדיקה (Unit Tests)

## דרישות כיסוי בדיקות (Test Coverage)

- כיסוי מינימלי של 80-70% לקוד חדש
- כיסוי מוגבר לקוד קריטי ולוגיקה עסקית מרכזית
- בדיקות עבור edge cases וגבולות

## סוגי בדיקות נדרשות

- Statement coverage
- Branch coverage
- Path coverage

## כלי בדיקה

- שימוש ב unittest, pytest או מסגרות דומות
- automation בדיקות pipeline CI/CD
- דוחות כיסוי (coverage reports)

## 4.2 טיפול ב Edge Cases & Error-Edge Cases Handling

### זיהוי ותיעוד Edge Cases

- זיהוי תנאי גבול ומקרים קיצוניים
- תיעוד כל edge case עם תיאור, קלט צפוי ותגובה
- צילומי מסך של תקלות (אם רלוונטי)

### Error Handling Mechanisms

- defensive programming
- הודעות שגיאה ברורות ומועילות
- לוגים מפורטים לצורכי בדיקות
- graceful degradation

### תיעוד תקלות

- תיאור התקלה והסיבה
- תגובת המערכת לטיפול בשגיאה
- השפעה על המשתמש או המערכת

## 4.3 תוצאות בדיקה צפויות (Expected Test Results)

- תיעוד תוצאות הריצה צפויות לכל בדיקה

- דוחות pass/fail rates עמ"ס automated testing
- לוגים של הרצות מוצלחות וכשלות

## **(Research & Results Analysis) 5. מבחן וניתוח תוצאות**

### **1. חקר פרמטרים (Parameter Exploration)**

**ניתוח רגישות פרמטרים (Sensitivity Analysis)**

- ניסויים שיטתיים עם שינוי פרמטרים
- תיעוד השפעת כל פרמטר על התוצאות
- שימוש בשיטות כמו one-at-a-time (OAT) : partial derivatives, variance-based analysis, one-at-a-time
- זיהוי פרמטרים קריטיים שימושיים ביותר על הביצועים

**תיעוד ניסויים**

- טבלת ניסויים עם ערכי פרמטרים ותוצאות
- גרפים ממחישים (line charts, heatmaps, sensitivity plots)
- ניתוח סטטיסטי של התוצאות

### **2. מחברת ניתוח תוצאות Results Analysis Notebook)**

**עומק הניתוח**

- שימוש ב Notebook Jupyter או כלים דומים
- ניתוח מתודי של תוצאות הניסויים
- השוואת אלגוריתמים, תוצאות או גישות שונות
- הוכחות מתמטיות או ניתוחים תיאורתיים (אם רלוונטי)

**הכללת נוסחאות ותבניות**

- שימוש ב LaTeX-למשוואות ונוסחאות
- הסברים מתמטיים מפורטים למודלים ואלגוריתמים
- אסמכתאות בספרות אקדמית

### **3. הציגות ויזואלית של תוצאות Visual Data Presentation)**

**סוגי ויזואליציות**

- Bar charts להשוואות קטגוריות
- Line charts למחמות לאורך זמן
- Scatter plots ליחסים מתחאים
- Heatmaps לריאשوت פרמטרים
- Box plots להציג התפלגות
- Waterfall charts לניתוח שינוי רציפים

#### איךות גרפים

- גרפים ברורים עם תוויות מדיקות
- שימוש בצבעים עקובים ונגישים
- כיתובים (captions) ומקרא (legends)
- רזולוציה גבוהה לפרסומים

#### כלי ויזואליזציה

- Matplotlib, Seaborn, Plotly לגרפים
- Tableau או Power BI לדשبورדים
- D3.js לויזואלייזציות אינטראקטיביות

## 6. ממשק משתמש ו-UX (User Interface & UX)

### 6.1 קרייטורי איזוטקס/וּס

#### Usability Criteria

- |                             |                   |   |
|-----------------------------|-------------------|---|
| קלות למידה ושימוש במערכת    | Learnability:     | • |
| כיצוע משימות בייעילות       | Efficiency:       | • |
| קלות חזרה למערכת לאחר הפסקה | Memorability:     | • |
| הגנה מפני טעויות משתמש      | Error Prevention: | • |
| שביעות רצון ועיצוב אסתטי    | Satisfaction:     | • |

#### Nielsen's 10 Heuristics

- Visibility of system status
- Match between system and real world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition over recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognize and recover from errors

## **6.2 תיעוד ממתק**

- צילומי מסך של כל מסך ומצב
  - תיאור workflow של משתמש
  - הסברים על אינטראקציות ופידבק
  - (accessibility considerations)

# **(Version Control & פיתוח ותיעוד גרסאות ניהול)**

## 7.1 Git Best Practices

- היסטוריה commits ברורה עם הודעות שימושיות
  - שימוש ב branches -לתוכנות חדשות
  - Pull Requests עם code reviews
  - גרסאות מרכזיות tagging

## **(Prompt Engineering Log)**

תיעוד תהליכי הפיתוח עם

- רישימת כל הפרומפטים המשמשו לבניית הפרויקט
  - תיאור הקשר והמטרה של כל פרומפט
  - דוגמאות לפלטטים שהתקבלו
  - שיפורים איטרטיביים של פרומפטים
  - best practices מהניסיו



## 8. עלויות ומחיר (Cost Analysis & Pricing)

### 8.1 ניתוח עלויות (Cost Breakdown)

שיעור כ- API Tokens

- ספירת tokens בכניסה ויציאה (input/output tokens)
- עלות למיליאן tokens (per Mtokens)
- עלות כוללת לפי מודל ושירות

טבלת עלויות

| מודול    | Input Tokens | Output Tokens | עלות כוללית           |
|----------|--------------|---------------|-----------------------|
| GPT-4    | 1,245,000    | 523,000       | \$45.67               |
| Claude 3 | 890,000      | 412,000       | \$32.11               |
| **       | 2,135,000    | 935,000       | **\$77.78**   סה"כ ** |

אופטימיזציה

- אסטרטגיות להפחיתת שימוש בתokens - batch processing
- בחירת מודלים(cost-effectiveness) לפי יעילות

### 8.2 ניהול תקציב

- תחזית עלויות לסקאלה
- monitoring של שימוש בזמן אמיתי
- התראות על חריגה מתקציב

## 9. הרחבה & Maintainability

### 9.1 נקודות הרחבה (Extension Points & Hooks)

ארQUITECTורת Plugins

- הגדרת interfaces בורורים להרחבה lifecycle hooks (beforeCreate, afterUpdate, etc.)

- middleware
- API-first design

### תיעוד הרחבה

- הדרכה לפיתוחplugins
- דוגמאות לתוספים conventions
- הל הרחבה בטוחה

## 9.2 Maintainability

### קוד ניתן לתחזקה

- -separation of concernsModularity
- של קומפוננטות Reusability
- Analyzability - כלות לבנייה וניתוח
- - כלות בדיקה Testability

## 10. תקני איכות ISO/IEC 25010 (ISO Quality Standards)

### 10.1 מאפייני איכות מוצר מוצר (Product Quality Characteristics)

#### הפרויקט יוערך לפי 8 מאפייני איכות:

|    |  |
|----|--|
| 1. | Functional Suitability (ההתאמה פונקציונלית)  |
|    | Completeness: •<br>Correctness: •<br>Appropriateness: •                                  |
| 2. | Performance Efficiency (יעילות ביצועים)  |
|    | Time behavior: •<br>Resource utilization: •<br>Capacity: •                               |
| 3. | Compatibility (תאימות)   |
|    | Interoperability: •<br>Coexistence: •  |
| 4. | Usability (שימושיות)   |
|    | Learnability, Operability, Accessibility •<br>User error protection •<br>UI Aesthetics • |

|                                |  |
|--------------------------------|--|
| .5.) אמינות Reliability        | <ul style="list-style-type: none"> <li>• Maturity: בשלות המערכת</li> <li>• Availability: זמינות</li> <li>• Fault tolerance: עמידות בתקלות</li> <li>• Recoverability: יכולת התאושש</li> </ul> |
| .6.) אבטחה Security            | <ul style="list-style-type: none"> <li>• Confidentiality, Integrity, Authenticity</li> <li>• Accountability, Non-repudiation</li> </ul>  |
| .7.) תחזוקתיות Maintainability | <ul style="list-style-type: none"> <li>• Modularity, Reusability, Analyzability</li> <li>• Modifiability, Testability</li> </ul>   |
| .8.) ניידות Portability        | <ul style="list-style-type: none"> <li>• Adaptability, Installability, Replaceability</li> </ul>   |

## 11. רשימת בדיקה סופית(Final Submission Checklist)

### תיעוד

|  |
|--|
| <ul style="list-style-type: none"> <li>• PRD מפורט עם כל המרכיבים</li> <li>• תיעוד ארכיטקטורה עם תרשימי בלוקים</li> <li>• README מקיים ברמה מהירה user manual</li> <li>• תיעוד API מלא</li> <li>• ספר פרומטפים מתוועד</li> </ul> |
|--|

### קוד

|   |
|---|
| <ul style="list-style-type: none"> <li>• מבנה פרויקט מודולרי ומאורגן</li> <li>• קבועים לא עולים על 150 שורות</li> <li>• הערות קוד מקיפות ובסיסיות -docstrings</li> <li>• עקביות בסגנון קוד</li> </ul> |
|---|

### קונפיגורציה

|  |
|--|
| <ul style="list-style-type: none"> <li>• קבוע config נפרדים מהקוד</li> <li>• קבוע env.example עם דוגמאות</li> <li>• אין API keys בקוד המקורי</li> <li>• מעודכן .gitignore</li> </ul> |
|--|

### בדיקות

|  |
|--|
| <ul style="list-style-type: none"> <li>• +70%覆盖率 unit tests</li> </ul> |
|--|

- תיעוד edge cases
- ניהול error handling מקייף
- דוחות בדיקה אוטומטיים

## מחקר וניתוח

- ניסויים עם שינויי פרמטרים
- ניתוח רגישות מתועדת
- מחברת ניתוח עם גרפים
- נוסחאות מתמטיות (אם רלוונטי)

## ויזואלייזציה

- גרפים איקטיים של תוצאות
- צילומי מסך שימוש משתמש
- תרשימי ארכיטקטורה ברורים

## עלויות

- טבלת שימוש בtokens
- ניתוח עלויות מפורט
- אסטרטגיות אופטימיזציה

## הרחבה

- extension points מתחעים
- דוגמאות לפיתוחplugins
- ממשקים ברורים להרחבה

## כלליים

- Git history מסודרת
- License מצורף
- Credits לספריותצד ג'
- deployment הוראות

## 12. מקורות ותקנים נוספים

מומלץ להתייחס לתקנים הבאים:

- MIT Software Quality Assurance Plan

- |                                      |   |
|--------------------------------------|---|
| ISO/IEC 25010 Software Quality Model | • |
| Google Engineering Practices         | • |
| Microsoft REST API Guidelines        | • |
| Nielsen's Usability Heuristics       | • |
- 

**הערה חשובה:** מסמך זה מציג רמת מצוינות גבוהה במיוחד במיוחד. לא כל סעיף הוא מחייב במלואו, אך ככל שיותר קритריונים מתקיימים, כך הציון והערכת האיכות יהיו גבוהים יותר. התמקדו בעומק, במקצועיות ובהדגמת יכולות מחקר ברמה אקדמית גהדגמת יכולות מחקר ברמה אקדמית גבוהה. מומלץ להשתמש בכל ה [LLM](#) לעזרה בהשלמת הפרוייקט.

מובהר – כי חלק מהבדיקה יתכן ויישנה שימוש בסוכני AI לביצוע הבדיקה