

הנחיות להגשת תוכנה ברמת הצטיינות יתרה לתואר שני במדעי המחשב

סקירה כללית

מסמך זה מגדיר את הקריטריונים להגשת פרויקט תוכנה ברמת מצוינות אקדמית המתאימה לסטודנטים מצטיינים במיוחד לתואר שני במדעי המחשב. הדרישות מתמקדות בפיתוח איכותי, תיעוד מקיף, ובהדגמת יכולות מחקר ופיתוח ברמה גבוהה. בכל מקום שכתוב פרויקט הכוונה למטלה משימה או פרויקט שניתנים במסגרת הקורס להגשה.

1. מסמכי פרויקט ותכנון (Project Documentation & Planning)

1.1 מסמך דרישות המוצר (PRD - Product Requirements Document)

יש להגיש מסמך PRD מפורט הכולל:

סקירת הפרויקט והקשר (Project Overview & Background)

- תיאור ברור של מטרת הפרויקט ובעיית המשתמש שהוא פותר
- ניתוח שוק תחרותי והצבה אסטרטגית
- זיהוי קהל היעד והצדדים המעוניינים (stakeholders)

יעדים ומדדי הצלחה (Objectives & Success Metrics)

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

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

- רשימת תכונות עם עדיפויות ברורות
- user stories ותיאורי שימוש (use cases)
- דרישות ביצועים, אבטחה, זמינות וscalability

הנחות, תלויות ומגבלות (Assumptions, Dependencies & Constraints)

- זיהוי מערכות חיצוניות ותלויות
- מגבלות טכניות וארגוניות

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

ציר זמן ואבני דרך (Timeline & Milestones)

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

1.2 מסמך ארכיטקטורה (Architecture Documentation)

תיעוד ארכיטקטוני מקיף הכולל:

(Block Diagrams & Architecture Diagrams)תרשימי בלוקים וזרימה

- תרשימי C4 Model (Context, Container, Component, Code)
- תרשימי UML לתהליכים מורכבים
- תרשימי deployment ותשתית
- ארכיטקטורה תפעולית (operational architecture)

תיעוד החלטות ארכיטקטוניות (Architecture Decision Records - ADRs)

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

תיעוד API ו-Interfaces

- תיעוד מפורט של כל ממשק ציבורי
- סכימות נתונים וקונטרקטים

2. תיעוד קוד ומבנה פרויקט (Code Documentation & Project Structure)

2.1 קובץ README מקיף

קובץ README ברמת user manual מלא הכולל:

(Installation Instructions)הוראות התקנה

- דרישות מערכת מפורטות (system requirements)
- הוראות התקנה 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
- workflow טיפוסים למשתמש

Examples & Demonstrations דוגמאות והדגמות

- דוגמאות קוד להרצה
- צילומי מסך ממשק המשתמש
- תרחישי שימוש נפוצים
- קישורים לסרטוני הדגמה (אם רלוונטי)

Configuration Guide מדריך תצורה

- הסבר על קבצי קונפיגורציה
- פרמטרים הניתנים לכיול והשפעתם

Contribution Guidelines

- הנחיות לתרומת קוד
- תקני קוד וסגנון

License & Credits

- רישיון השימוש
- ייחוס לספריות צד ג' ותורמים

2.2 מבנה פרויקט מודולרי (Modular Project Structure)

ארגון נכון של מבנה הפרויקט:

עקרונות ארגון

- חלוקה לתיקיות לפי תפקיד: `src/`, `tests/`, `docs/`, `data/`, `results/`, `config/`, `assets/`
- ארגון feature-based או layered architecture
- הפרדה ברורה בין קוד, נתונים, תוצאות ותיעוד

גודל קבצים

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

Naming Conventions

- שמות תיקיות וקבצים תיאוריים ועקביים
- שימוש באותו סגנון naming בכל הפרויקט

Structure Example

text

project-root/

src/ # קוד המקור

agents/ # מודולי סוכנים

utils/ # פונקציות עזר

config/ # קוד קונפיגורציה

tests/ # בדיקות יחידה ואינטגרציה

data/ # מסדי נתונים וקבצי קלט

results/ # תוצאות ניסויים

docs/ # תיעוד נוסף

config/ # קבצי הגדרות

assets/ # תמונות, גרפים, משאבים

notebooks/ # מחברות ניתוח תוצאות

README.md

requirements.txt

.gitignore

2.3 איכות קוד והערות (Code Quality & Comments)

תקני הערות בקוד (Code Comments Standards)

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

עקרונות כתיבת קוד איכותי

- שמות משתנים ופונקציות תיאוריים
- פונקציות קצרות וממוקדות (single responsibility)
- הימנעות מקוד כפול (DRY - Don't Repeat Yourself)
- עקביות 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 בתוך 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 רגישים
- קבצי template לקונפיגורציה בגרסאות שונות (dev, staging, production)

3.2 אבטחת מידע (Information Security)

הגנה על API Keys וסודות

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

Rotation & Monitoring

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

24.בדיקות ואיכות תוכנה (Testing & Quality Assurance)

4.1 יחידות בדיקה (Unit Tests)

דרישות כיסוי בדיקות (Test Coverage)

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

סוגי בדיקות נדרשות

- Statement coverage
- Branch coverage
- Path coverage למסלולים קריטיים

כלי בדיקה

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

4.2 טיפול ב-Edge Cases ותקלות (Edge Cases & Error Handling)

זיהוי ותיעוד Edge Cases

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

Error Handling Mechanisms

- defensive programming עם בדיקות קלט
- הודעות שגיאה ברורות ומועילות
- לוגים מפורטים לצורך debugging
- graceful degradation במקרי כשל

תיעוד תקלות

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

4.3 תוצאות בדיקה צפויות (Expected Test Results)

- תיעוד תוצאות הרצה צפויות לכל בדיקה

- דוחות automated testing עם pass/fail rates
- לוגים של הרצות מוצלחות וכושלות

5. מחקר וניתוח תוצאות (Research & Results Analysis)

5.1 חקר פרמטרים (Parameter Exploration)

ניתוח רגישות פרמטרים (Sensitivity Analysis)

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

תיעוד ניסויים

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

5.2 מחברת ניתוח תוצאות (Results Analysis Notebook)

עומק הניתוח

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

הכללת נוסחאות ומתמטיקה

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

5.3 הצגה ויזואלית של תוצאות (Visual Data Presentation)

סוגי ויזואליזציות

<ul style="list-style-type: none"> • Bar charts להשוואות קטגוריות • Line charts למגמות לאורך זמן • Scatter plots לזיהוי מתאמים • Heatmaps לרגישות פרמטרים • Box plots להצגת התפלגויות • Waterfall charts לניתוח שינויים רציפים
--

איכות גרפית

<ul style="list-style-type: none"> • גרפים ברורים עם תוויות מדויקות • שימוש בצבעים עקביים ונגישים • כיתובים (captions) ומקרא (legends) • רזולוציה גבוהה לפרסומים
--

כלי ויזואליזציה

<ul style="list-style-type: none"> • Matplotlib, Seaborn, Plotly לגרפים • Tableau או Power BI לדשבורדים • D3.js לויזואליזציות אינטראקטיביות
--

6. מחשק משתמש ו-UX (User Interface & UX)

6.1 קריטריוני איכות UX/UI

Usability Criteria

<ul style="list-style-type: none"> • Learnability: קלות למידה ושימוש במערכת • Efficiency: ביצוע משימות ביעילות • Memorability: קלות חזרה למערכת לאחר הפסקה • Error Prevention: הגנה מפני טעויות משתמש • Satisfaction: שביעות רצון ועיצוב אסתטי

Nielsen's 10 Heuristics

<ul style="list-style-type: none"> • 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 ממשק

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

7.ניהול גרסאות ותיעוד פיתוח (Version Control & Development Documentation)

7.1 Git Best Practices

- היסטוריית commits ברורה עם הודעות משמעותיות
- שימוש ב branches לתכונות חדשות
- Pull Requests עם code reviews
- tagging של גרסאות מרכזיות

7.2 ספר הפרומפטים (Prompt Engineering Log)

תיעוד תהליך הפיתוח עם AI

- רשימת כל הפרומפטים המשמעותיים ששימשו לבניית הפרויקט
- תיאור הקשר והמטרה של כל פרומפט
- דוגמאות לפלטים שהתקבלו
- שיפורים איטרטיביים של פרומפטים
- best practices שהופקו מהניסיון

מבנה מומלץ

text

prompts/

README.md סקירה כללית

architecture/ פרומפטים לתכנון ארכיטקטורה

code-generation/ פרומפטים לייצור קוד

testing/ פרומפטים לבדיקות

documentation/ פרומפטים לתיעוד

8. עלויות ותמחור (Cost Analysis & Pricing)

8.1 ניתוח עלויות (Cost Breakdown)

שימוש ב-API Tokens

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

טבלת עלויות

text				מודל
עלות כוללת	Input Tokens	Output Tokens		
	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 להפחתת שימוש ב-tokens-
- batch processing לשיפור יעילות
- בחירת מודלים לפי cost-effectiveness

8.2 ניהול תקציב

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

9. הרחבה ו Extensibility (Extensibility & Maintainability)

9.1 נקודות הרחבה (Extension Points & Hooks)

ארכיטקטורת Plugins

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

- מנגנוני middleware
- API-first design

תיעוד הרחבה

- הדרכה לפיתוח plugins
- דוגמאות לתוספים
- conventions להרחבה בטוחה

9.2 Maintainability

קוד ניתן לתחזוקה

- Modularity | separation of concerns
- Reusability של קומפוננטות
- Analyzability קלות להבנה וניתוח
- Testability קלות בדיקה

10. תקני איכות (ISO Quality Standards)

10.1 מאפייני איכות מוצר (Product Quality Characteristics)

הפרויקט יוערך לפי 8 מאפייני איכות:

1. Functional Suitability (התאמה פונקציונלית)
 - Completeness: כיסוי כל התכונות הנדרשות
 - Correctness: דיוק התוצאות
 - Appropriateness: התאמה לשימושים
2. Performance Efficiency (יעילות ביצועים)
 - Time behavior: זמן תגובה
 - Resource utilization: שימוש במשאבים
 - Capacity: יכולת להתמודד עם עומסים
3. Compatibility (תאימות)
 - Interoperability: עבודה עם מערכות אחרות
 - Coexistence: במקביל למערכות אחרות
4. Usability (שימושיות)
 - Learnability, Operability, Accessibility
 - User error protection
 - UI Aesthetics

Reliability (אמינות)	5.
Maturity: בשלות המערכת	•
Availability: זמינות	•
Fault tolerance: עמידות בתקלות	•
Recoverability: יכולת התאושש	•
Security (אבטחה)	6.
Confidentiality, Integrity, Authenticity	•
Accountability, Non-repudiation	•
Maintainability (תחזוקתיות)	7.
Modularity, Reusability, Analyzability	•
Modifiability, Testability	•
Portability (ניידות)	8.
Adaptability, Installability, Replaceability	•

11. רשימת בדיקה סופית (Final Submission Checklist)

תיעוד

PRD מפורט עם כל המרכיבים	•
תיעוד ארכיטקטורה עם תרשימי בלוקים	•
README מקיף ברמת user manual	•
תיעוד API מלא	•
ספר פרומפטים מתועד	•

קוד

מבנה פרויקט מודולרי ומאורגן	•
קבצים לא עולים על 150 שורות	•
הערות קוד מקיפות וdocstrings-	•
עקביות בסגנון קוד	•

קונפיגורציה

קבצי config נפרדים מהקוד	•
קבצי <code>.env.example</code> עם דוגמאות	•
אין API keys בקוד המקור	•
<code>.gitignore</code> מעודכן	•

בדיקות

unit tests עם כיסוי +70%	•
--------------------------	---

- תיעוד edge cases
- error handling מקיף
- דוחות בדיקה אוטומטיים

מחקר וניתוח

- ניסויים עם שינוי פרמטרים
- ניתוח רגישות מתועד
- מחברת ניתוח עם גרפים
- נוסחאות מתמטיות (אם רלוונטי)

ויזואליזציה

- גרפים איכותיים של תוצאות
- צילומי מסך ממשק משתמש
- תרשימי ארכיטקטורה ברורים

עלויות

- טבלת שימוש ב-tokens
- ניתוח עלויות מפורט
- אסטרטגיות אופטימיזציה

הרחבה

- extension points מתועדים
- דוגמאות לפיתוח plugins
- ממשקים ברורים להרחבה

כללים

- Git history מסודרת
- License מצורף
- Credits לספריות צד ג'
- הוראות deployment

12. מקורות ותקנים נוספים

חומלץ 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. לא כל סעיף הוא מחויב במלואו, אך ככל שיותר קריטריונים מתקיימים, כך הציון והערכת האיכות יהיו גבוהים יותר. התמקדו בעומק, במקצועיות ובהדגמת יכולות מחקר ברמה אקדמית גהדגמת יכולות מחקר ברמה אקדמית גבוהה. מומלץ להשתמש בכלי LLM לעזרה בהשלמת הפרויקט.

מובהר – כי כחלק מהבדיקה יתכן וייעשה שימוש בסוכני AI לביצוע הבדיקה