שאלותשמכסה את כל שלבי חיי הפרויקט:

בשאלות הבאות אין להסתפק בתשובות כן ולא. יש לתת הסבר.

**שלב אפיון וניתוח**

1. **מה הבעיה שניסית לפתור בפרויקט שלך? למי הוא מיועד?**
2. **כיצד ניגשת להגדרת הדרישות? מהן הדרישות הפונקציונליות ומהן הדרישות הלא פונקציונליות?**

**מהן דרישות פונקציונליות?**

**הגדרה**:  
הן מתארות מה המערכת צריכה לעשות - כלומר **ההתנהגות הצפויה או הפונקציות של המערכת**.

**🔹 דוגמאות:**

* המשתמש יוכל להירשם למערכת עם שם משתמש וסיסמה.
* לקוח יוכל לשלוח הודעה לשרת ולקבל תגובה.
* המערכת תאפשר שמירת קובץ למסד נתונים.
* המשתמש יוכל לאפס סיסמה באמצעות אימייל.

תחשוב על זה כ״רשימת פעולות״ שהמערכת יודעת לבצע, או תרחישים שצריכים להתאפשר.

**מהן דרישות לא פונקציונליות?**

**הגדרה**:  
הן מתארות *איך* המערכת צריכה להתנהג, **תכונות איכותיות** של המערכת – לא הפעולות עצמן אלא האופן שבו הן מתבצעות.

**🔹 דוגמאות:**

* המערכת תטפל ב־500 בקשות במקביל (scalability).
* זמן תגובה לא יעלה על 2 שניות (performance).
* המערכת תשתמש בהצפנת AES-256 לצורך שמירת נתונים (security).
* המערכת תיתמך במובייל ודפדפנים שולחניים (usability / compatibility).
* זמינות של 99.9% uptime (reliability).

אלה הדרישות שמתארות את **האיכות**, לא את היכולות.

**בעיקרון:**

* **פונקציונלי**: "מה המשתמש עושה?" → תרחישי שימוש
* **לא פונקציונלי**: "איך זה צריך לקרות?" → ביצועים, אבטחה, ממשק, זמינות, סקלביליות וכו’

**דוגמה על פרויקט של שרת מרובה לקוחות:**

| **סוג דרישה** | **דוגמה** |
| --- | --- |
| פונקציונלית | השרת יקבל חיבור TCP מלקוח וישמור את הנתונים למסד |
| לא פונקציונלית | השרת יתמוך בעד 1000 חיבורים בו־זמנית בלי לקרוס |

1. **באילו טכנולוגיות שקלת להשתמש בתחילת הדרך, ולמה בחרת דווקא בטכנולוגיות הסופיות?**
2. **מה היה האתגר הגדול ביותר באפיון, ואיך פתרת אותו?**

**אפיון וניתוח מערכת (הבנת הבעיה, דרישות ופתרונות)**

שאלות נוספות:

1. איך היית מנסח את הבעיה ב־2 משפטים למנהל מוצר שלא מבין בתכנות?
2. אילו אילוצים טכנולוגיים/חברתיים/אחרים הגבילו את הפתרון שלך?

**סוגי אילוצים עיקריים:**

1. **אילוצים טכנולוגיים**

מגבלות שנובעות מהשפה, הכלים, או סביבת העבודה.

**דוגמאות:**

* חייבים להשתמש ב־Java בלבד (כי זה קורס ב-Java).
* לא ניתן להשתמש בשרתים אמיתיים - רק ב־localhost.
* המחשב האישי איטי - אי אפשר להריץ 1000 תהליכים בו־זמנית.
* אין גישה לאינטרנט - לכן אי אפשר להשתמש ב-Firebase או API חיצוניים.
* המערכת צריכה לרוץ גם על מחשבים ישנים ⟵ דרישה לקוד יעיל מאוד.

1. **אילוצים חברתיים / אנושיים**

קשורים למשתמשים, תקשורת בצוות, או למי שצפוי להשתמש במערכת.

**דוגמאות:**

* המשתמשים הם בני נוער, אז צריך ממשק גרפי פשוט מאוד.
* המערכת תשרת אוכלוסייה דוברת ערבית, אז נדרש תרגום מלא.
* העבודה נעשית בזוגות, ובן הזוג לא תמיד היה זמין - אז חילקנו תפקידים אחרת.
* המרצה דרש שהמערכת תכלול GUI, למרות שזה לא נדרש לפתרון הטכני.

1. **אילוצים תהליכיים / לוגיסטיים / ארגוניים**

מגבלות זמן, משאבים, הנחיות פורמליות וכו'.

**דוגמאות:**

* רק חודש לפיתוח ⟵ אי אפשר לבנות מערכת של Microservices.
* אסור להשתמש בקוד מגוגל ⟵ נדרש לכתוב הכל מאפס.
* המערכת צריכה להיבחן על־ידי תלמידים אחרים ⟵ נדרש תיעוד טוב.
* אסור לשמור נתונים על ענן בגלל פרטיות.

## למה זה חשוב?

מי שמזהה אילוצים ויודע להסביר איך הם השפיעו על הפתרון – מראה **חשיבה הנדסית מערכתית**. הוא לא פשוט ״זרם עם הקוד״ - אלא חשב והתאים את עצמו למצב.

1. האם ביצעת סקר מערכות דומות קיימות? מה למדת מהן?
2. איך ידעת שמערכת כזו בכלל נדרשת? האם יש "כאב" אמיתי שהיא פותרת?
3. אילו נתונים דרושים למערכת שלך כדי לפעול? מהיכן הם מגיעים?

**"אילו נתונים דרושים למערכת שלך כדי לפעול?"**

הכוונה כאן היא לשאול:

1. מהו **המידע שהמערכת שלך צריכה כדי לפעול כראוי**?
2. בלי אילו נתונים היא לא תוכל להתחיל / להמשיך לפעול?

**🔹 סוגי נתונים אפשריים:**

* פרטי משתמשים (שם משתמש, סיסמה, אימייל)
* נתוני התחברות (Token, Session ID)
* הגדרות מערכת (קונפיגורציה, מפתחות API)
* מידע עסקי (הזמנות, מוצרים, לקוחות, הודעות)
* נתוני זמן אמת (מיקום, חיישנים, קלט מהמשתמש)
* מפתחות הצפנה או סיסמאות פנימיות

1. **"מהיכן הם מגיעים?"**

השאלה הזו בודקת: **מאיפה מגיע המידע הזה? מי אחראי עליו? ואיך הוא מוזן למערכת?**

**🔸 דוגמאות:**

* מהקלט של המשתמש (GUI, CLI)
* מקובץ חיצוני (config.json, CSV)
* ממסד נתונים (MySQL, MongoDB וכו')
* מ־API חיצוני (לדוג' שירות מזג אוויר, שירות תשלום)
* משרת אחר (בתקשורת TCP בין רכיבים)
* מטבלת זיכרון משותפת או cache

**למה השאלה הזו חשובה?**

1. היא מוודאת **שהבנת מה המערכת באמת צריכה כדי לפעול**.
2. היא חושפת אם הייתה **הפרדה ברורה בין קלט, עיבוד ופלט**.
3. היא בודקת אם חשבת על **הזנה, אימות, ואבטחת הנתונים**.

**דוגמה לתשובה טובה:**

"המערכת שלי מקבלת נתוני משתמש מהטופס ב־GUI (שם וסיסמה), ושולחת אותם לשרת. השרת מאמת את הנתונים מול מסד הנתונים (MySQL). בנוסף, השרת משתמש בקובץ הגדרות config.json שמכיל את מפתח ההצפנה ואת כתובת ה־DB. בלי הקובץ הזה, המערכת לא תוכל לפעול."

זו תשובה שמעידה על:

* הבנה של מקורות נתונים
* אחריות רכיבים
* תלות במידע חיצוני

**שלב התכנון**

1. **תאר את מבנה המערכת: אילו רכיבים יש בה, ואיך הם מתקשרים ביניהם?**
2. **כיצד תכננת את העבודה עם משאבים משותפים (כמו קבצים או זיכרון)? אילו מנגנוני סנכרון השתמשת ולמה דווקא בהם?**
3. **באילו שלבים שקלת חלופות – למשל למסד נתונים או הצפנה – ומה היו השיקולים לבחירה הסופית?**
4. **אם היית צריך לתעד את הארכיטקטורה למתכנת שימשיך אחריך – איך היית מסביר אותה?**

**תכנון מערכת וארכיטקטורה (תקשורת, סנכרון, מבנה)**

1. אילו תהליכים רצים במקביל? איך אתה מבטיח שהם לא מתנגשים?
2. אם מחר יבקשו ממך להוסיף מודול חדש – איפה במערכת תכניס אותו?
3. האם קיימת תלות הדדית (Coupling) גבוהה בין רכיבים? איך ניתן להפחית אותה?
4. איך תיארת לעצמך את המסלול של בקשה מהרגע שלחצו על כפתור בממשק – ועד שהנתונים חזרו?
5. תאר את תהליך/פעולה מורכבת, שלב אחר שלב, מבחינת תכנון. (לדוגמא login) בחר בפעולה שמיחדת את הפרויקט שלך מפרויקטים אחרים. (הזרמת וידאו, העתקת קובץ ועוד...)
6. בחר מודול שיש בו מספר מחלקות והפרד אותו למודולים שונים וגרום לתוכנית לרוץ כמו שצריך. אם אין לך אף מודול כזה אז הסביר מדוע זה לא אפשרי. אחרת בצע זאת. זה בודק האם אתה מבין את הקשרים בין הרכיבים של המערכת את האחראיות של כל מחלקה ואת יחסי הגומלין ביניהן. רשום מה עשית

**שלב המימוש**

1. **מהי חלוקת האחריות בין הקליינט לשרת אצלך? ואיך טיפלת בבעיות כמו טיפול בו-זמני במספר קליינטים?**
2. **האם נתקלת במרוץ תהליכים (Race Condition)? איך זיהית אותו ואיך פתרת אותו?**
3. **תאר לי כיצד בנית את מערכת ההרשאות או ההצפנה – ולמה בחרת בפתרון הזה דווקא?**
4. **אם היית צריך לשנות טכנולוגיה באמצע הפרויקט – איך זה השפיע עליך?**
5. באיזה חלק בקוד אתה הכי גאה, ולמה?
6. אילו ספריות השתמשת בהן? האם הבנת לעומק מה הן עושות?
7. איפה הקוד שלך עשוי להיכשל – ומה תעשה במקרה כזה?
8. האם יש מקום שבו בחרת לוותר על פיצ’ר כדי לא להסתבך? אם כן – תסביר למה.

**שלב הבדיקות**

1. **איך בדקת את המערכת? האם כתבת בדיקות אוטומטיות או בדקת ידנית? פרט**
2. **כיצד בדקת תרחישים של עומס (stress testing) או תקלות תקשורת? פרט**
3. **האם בדקת מה קורה כאשר משתמש מנסה לבצע פעולה חריגה או זדונית? פרט**
4. **איזה באגים גילית רק לאחר זמן, וכיצד איתרת אותם?**

**בדיקות, אבטחה ואיכות**

1. כתבת בדיקות ידניות או אוטומטיות? תן דוגמה לבדיקה ובאג שמצאת דרכה.
2. איך בדקת מה קורה אם השרת לא עונה? או אם יש נתק ברשת?
3. איך בדקת תרחישים זדוניים? למשל נסיונות SQL Injection או הצפה (flood)?
4. אילו דרכי גיבוי או שיחזור מידע קיימות במערכת שלך?
5. מה קורה אם שני קליינטים מנסים לבצע אותה פעולה במקביל? ניסית לבדוק? פרט

**שאלות עומק ואותנטיות**

1. **אם הייתי מבקש ממך לשכתב את הרכיב המרכזי של הפרויקט – איך היית עושה זאת אחרת?**
2. **איזה חלק מהקוד כתבת בעצמך, ואיפה השתמשת בקוד גנרי או עזרה ממודלים? איך אתה מבטיח שהבנת אותו לעומק?**
3. **אם הייתה לך עוד שבוע עבודה – מה היית משפר או מוסיף?**
4. **איזה עיקרון למדת תוך כדי הבנייה, שלא ידעת לפני?**

**הבנה אותנטית, מקוריות ולמידה**

1. אילו חלקים בפרויקט לא ממש הבנת בהתחלה, ורק תוך כדי תהליך למדת?
2. אילו החלטות טכנולוגיות היו הכי קשות? האם בדיעבד היית מחליף אותן?
3. אם תבנה את המערכת מחדש – מה תעשה אחרת?
4. מה הפתיע אותך בתהליך העבודה? מה לא ציפית שיקרה?
5. האם אתה מרגיש שכתבת את הקוד הזה לבד, או רק שיפצת קוד קיים? תסביר.

## שאלות על ביצועים, יעילות וצריכת משאבים

**יעילות אלגוריתמית:**

1. האם יש קטעי קוד שאתה יודע שהם לא יעילים? אם כן – למה לא שיפרת אותם?
2. מהו סדר הזמן (Time Complexity) של פעולה מרכזית במערכת שלך? הסבר למה.
3. אם יגיעו פי 10 יותר לקוחות – איזה חלק במערכת שלך יקרוס ראשון?
4. איך טיפלת בלולאות או שליפות מהמסד כדי שלא יבצעו עבודה כפולה?
5. האם השתמשת במבני נתונים מתאימים (HashMap, Queue וכו) כדי לשפר ביצועים?

**צריכת זיכרון:**

1. האם אתה שומר מידע מיותר בזיכרון לאורך זמן? תן דוגמה.
2. איך טיפלת בניקוי של משתנים, buffers או חיבורים לא נחוצים?
3. יש לך קובץ Log או קבצים זמניים? האם בדקת כמה נפח הם צורכים?
4. האם שקלת שימוש במבנה נתונים דחוס או בנוי מראש כדי לחסוך בזיכרון?

**ניתוח ביצועים בפועל:**

1. האם מדדת את זמן הביצוע של פעולה מסוימת? מה גילית?
2. אילו כלים השתמשת כדי לבדוק ביצועים? (לוגים, timers, פרופיילר וכו')
3. תאר מקרה שבו המערכת הייתה "איטית" – מה עשית כדי לגלות את הסיבה?
4. האם יש חלק במערכת שרץ במקביל (Threading) לשיפור ביצועים? האם זה עזר?

**עומסים ואופטימיזציה:**

1. כתבת מערכת מבוססת שרת – איך טיפלת בעומסים של קליינטים מרובים?
2. אילו משאבים היו צוואר הבקבוק (CPU, זיכרון, דיסק, תקשורת)?
3. האם שקלת שימוש ב־Caching? תסביר למה כן או לא.
4. האם היה שלב שבו אופטימיזציה פגעה בפשטות הקוד? מה בחרת לעשות?