.1 שם הפרויקט

Ayin Bright – אתר לאבחון ושיפור הראייה

.2 רקע

**. 2.1 תיאור ורקע כללי**

בעידן הדיגיטלי של היום, אנשים רבים חווים אי נוחות חזותית והידרדרות עקב חשיפה ממושכת למסכים ואורח חיים בישיבה. VisionEnhance נוצרה במטרה להעצים את המשתמשים לקחת אחריות על בריאות העיניים שלהם באמצעות תוכן מרתק ואינפורמטיבי.

**.2.2 מטרות המערכת**

קדם תרגילי עיניים: - הצע מגוון תרגילי עיניים אינטראקטיביים שמטרתם לשפר את המיקוד, הגמישות ובריאות העין הכללית. - עודדו את המשתמשים לשלב את התרגילים האלה בשגרה היומיומית שלהם.

הפחת את עומס העיניים: - שתף טיפים וטכניקות למזעור עומס עיניים דיגיטלי שנגרם כתוצאה מזמן מסך ממושך, כגון כלל 20-20-20. - לספק הדרכה לגבי ארגונומיה נכונה לשימוש במחשב ובמכשיר נייד.

תמוך בהרגלים בריאים: - למד את המשתמשים על השפעת התזונה על הראייה והצע שינויים תזונתיים לתמיכה בבריאות העיניים. - לקדם שינויים באורח החיים שיכולים לשפר את חדות הראייה, כגון פעילות גופנית סדירה ושינה מספקת.

.3 סקירת מצב קיים בשוק

שוק שיפור הראייה וטיפוח העיניים מתפתח במהירות, מונע על ידי צרכי צרכנים משתנים, התקדמות טכנולוגית ודגש הולך וגובר על בריאות מונעת. האתר שלי יכול למצב את עצמו כמשאב בעל ערך בנוף זה על ידי הצעת פתרונות חדשניים, תוכן חינוכי כדי לענות על הדרישות של צרכנים מודעים לבריאות.

.4 מה הפרויקט אמור לחדש או לשפר

* לפתח תוכניות מותאמות המבוססות על הערכות אישיות, העדפות ומעקב התקדמות. זה יכול לכלול משטרי פעילות גופנית מותאמים אישית ותוכן חינוכי שמסתגל לאורך זמן ככל שהמשתמשים משתפרים.

 ליצור תרגילים מגניבים וכלים אינטראקטיביים שהופכים את שיפור הראייה למהנה ומרתק. זה יכול לכלול

אתגרים, חידונים ותגמולים עבור השלמת תרגילים או מודולים.

.5 דרישות מערכת ופונקציונאליות

.5.1 דרישות מערכת

* **סביבת הטמעה ושימוש:** מערכת מבוססת אינטרנט שתהיה נגישה מכל מכשיר עם חיבור לאינטרנט.
* **שרידות וגיבוי נתונים:** המערכת תתוכנן כך שתוכל להתמודד עם מספר משתמשים בו זמנית. נתוני המערכת יגובו על בסיס יומי כדי להבטיח שמירה ושחזור של מידע במקרה של תקלה
* **ביצועים והתמודדות עם עומסים:** המערכת תתוכנן עם דגש על ביצועים גבוהים וזמן תגובה מהיר

**.5.2 דרישות פונקציונאליות.**

**רישום משתמשים ומעקב:**

* משתמשים צריכים להיות מסוגלים ליצור חשבונות, לנהל פרופילים ולעקוב אחר ההתקדמות שלהם לאורך זמן.
* ספריית תרגילים אינטראקטיביים עם הוראות ברורות ועזרים ויזואליים.

.6 בעיות צפויות במהלך הפיתוח ופתרונות

**6.1 תיאור הבעיות**

- בעיה**:** האתר עשוי להיתקל בזמני טעינה איטיים, במיוחד עם תוכן כבד או תנועה רבה

. - פתרון: בצע אופטימיזציה של תמונות ונכסים, יישם טעינה עצלנית והשתמש ב-CDN כדי לשפר את מהירות האתר. עקוב באופן קבוע אחר מדדי ביצועים והתאם לפי הצורך.

- בעיה**:** טיפול בנתוני משתמש רגישים עלול להעלות בעיות אבטחה ופרטיות

- פתרון**:** הטמע שיטות הצפנה חזקות, פתרונות אחסון נתונים מאובטחים ועמידה בתקנות הרלוונטיות להגנה על נתונים (למשל, GDPR). בצע בקביעות ביקורות ועדכוני אבטחה.

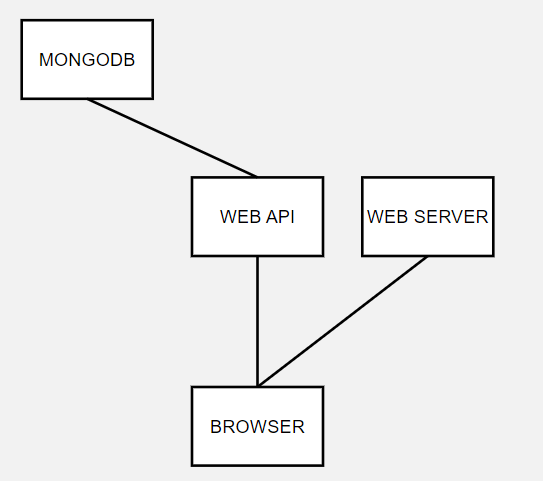
**פתרון טכנולוגי נבחר:**

7**.**1 טופולוגית הפתרון

המערכת תיושם ברשת האינטרנט וכוללת את המרכיבים הבאים:

* שרת אינטרנט: לאחסון ולהרצת האפליקציה.
* ממשק משתמש בצד הלקוח: הנגיש מדפדפנים מודרניים ומותאם למכשירים שונים.
* מסד נתונים לאחסון מידע על משתמשים, תרגילים ומעקב.
* חיבור אינטרנטי: לשם גישה למערכת מכל מקום.

**7.3 שפות הפיתוח**

* JavaScript : לשם פיתוח בצד הלקוח והשרת בשל פופולריות, ביצועים וגמישות.
* HTML לשם בניית מבנה התוכן והשלד של הדפים באתר, תוך שמירה על תקני אינטרנט נגישים ומותאמים לכלל הדפדפנים.
* CSS לעיצוב ממשק המשתמש בשל יכולתו להקל על תחזוקת הקוד ולספק אפשרויות עיצוב מתקדמות.
* MONGODB לשם ניהול ואחסון מידע במסד הנתונים הרלציוני.
* 

**7.4 תיאור הארכיטקטורה הנבחרת**

הארכיטקטורה היא Microservices המאפשרת חלוקה למודולים עצמאיים לשם פיתוח, תחזוקה ועדכון קלים יותר. כל שירות יתפקד באופן עצמאי, מה שמקל על שדרוגים ותיקונים מבלי להשפיע על המערכת כולה.

**7.5 חלוקה לתכניות ומודולים**

* **ממשק משתמש:** כולל את כל הממשקים הוויזואליים והשירותים הקשורים אליו.
* **API של שרת:** כולל את כל ה Endpoints-ליצירת, קריאה, עדכון ומחיקת נתונים.
* **מסד נתונים:** כולל את כל המודולים הקשורים לניהול הנתונים במערכת.

**7.6 סביבת השרת**

**שירות ענן**: הסביבה תופעל על שרתים וירטואליים בענן, כגון AWS או Azure לשם גמישות וסקלאביליות. השימוש בשירותי ענן יאפשר גמישות בהתאמה לעומסים משתנים והבטחת זמינות גבוהה של השירות.

**7.7 ממשק המשתמש**

הממשק יכלול אפשרויות שונים, מידע על הלקוח ומעקב באוסף התרגילים שהאתר נודע אליו.

מבוסס על React לשם פיתוח ממשק המשתמש בצד הלקוח, מאפשר פיתוח דינמי ואינטראקטיבי עם אפשרויות לשימוש חוזר בקומפוננטות ולתחזוקה קלה של הקוד.

**7.8 ממשקים למערכות אחרות API**

חיבור לAPI חיצוני להטמעת השעה הרלוונטית של המשתמש.

* 1. **שימוש בחבילות תוכנה**

Jwtלאימות ואבטחת משתמשים באמצעות עוגיות מוצפנות.

8. שימוש במבני נתונים וארגון קבצים

**8.1 מבני נתונים**

* **משתמש (User)**:

מזהה ייחודי (ID), שם משתמש, כתובת דוא"ל, סיסמא (מוצפנת), חולשות של הראייה, - שדה ראייה, טשטוש, ועיוורון צבעים

**8.2 שיטת האיחסון**

* **מסד נתונים**: MONGODB
* **איחסון בענן**:

התכנון כולל שימוש באחסון בענן לצורך גיבוי ושמירת נתונים קריטיים, כגון גיבויי מסד הנתונים, ומידע רגיש אחר.

**8.3 מנגנוני התאוששות מנפילה או קריסה**

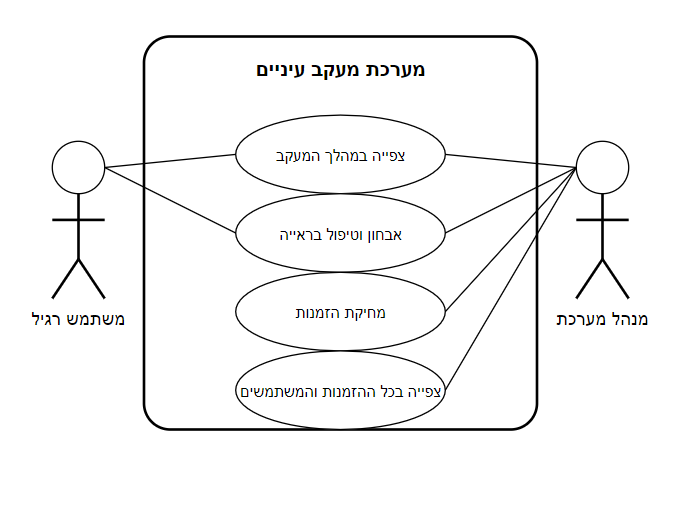
**התאוששות מנפילה/קריסה**: גיבוי תקופתי יומי של מסד הנתונים ושמירתם במיקום מאובטח בענן.

תהליך שחזור מתועד.

9. רשימת מערכת מרכזיים

**9.1 תרשים מקרה שימוש (Use Case Diagram)**

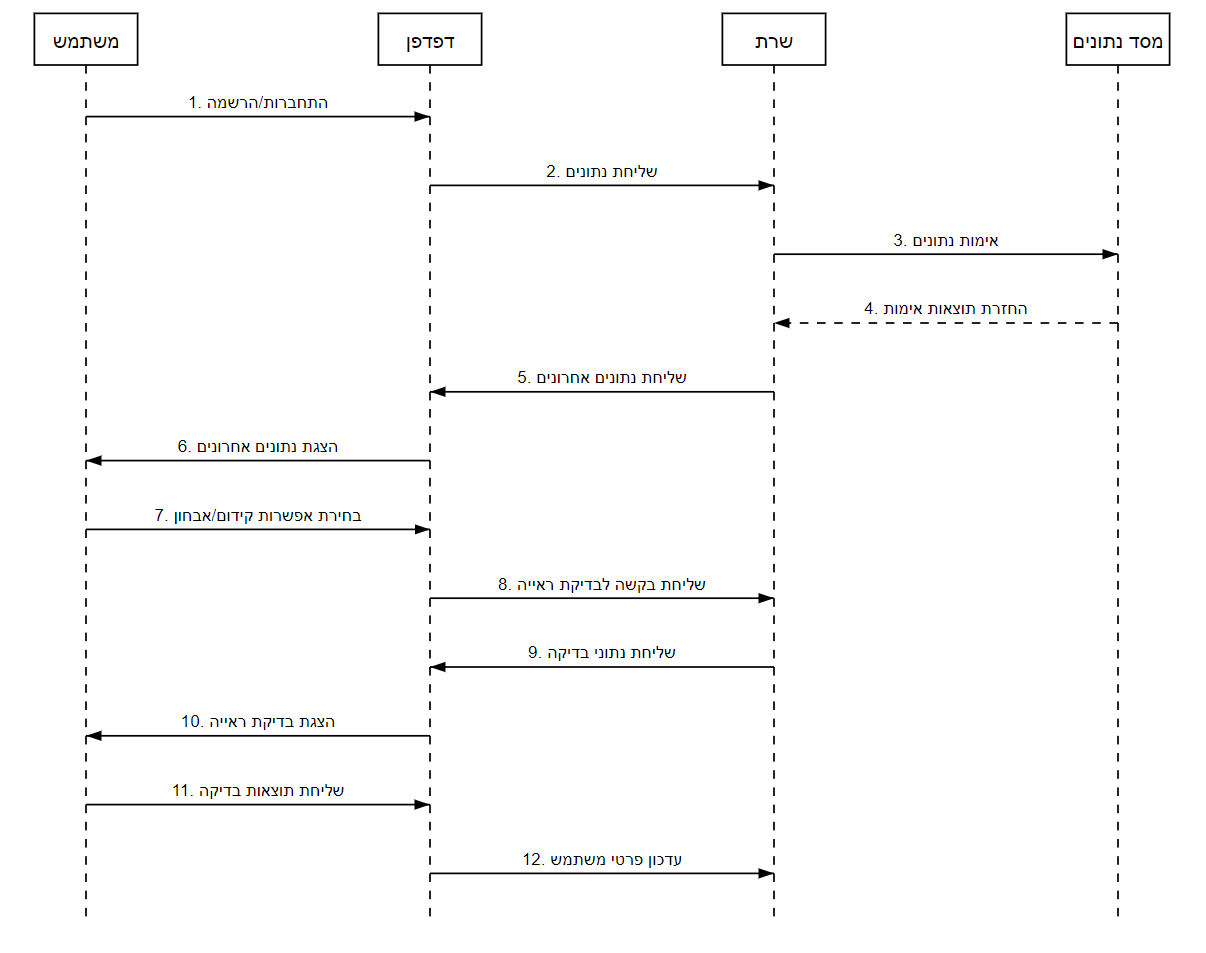
התרשים מתאר את האינטראקציות העיקריות בין המשתמשים לבין המערכת ואת ההרשאות השונות:

* **משתמש רגיל:** יכול לצפות בכל מהלך המעקב שעליו, לאבחן לשפר ולתרגל את הראייה שלו.
* **מנהל מערכת**: יכול לבצע את כל הפעולות של המשתמש הרגיל והקבלה, בנוסף למחיקת הזמנות, וצפיה ברשימת כל ההזמנות וכל המשתמשים.
* 

**9.2 תרשים רצף (Sequence Diagram)**

תרשים רצף קריאות פונקציות מרכזיות בלוגיקה העיסקית המרכזית של הפרוייקט מתאר את רצף הקריאות בפונקציות שונות בזמן ביצוע פעולות עיקריות כמו ביצוע הזמנה חדשה:

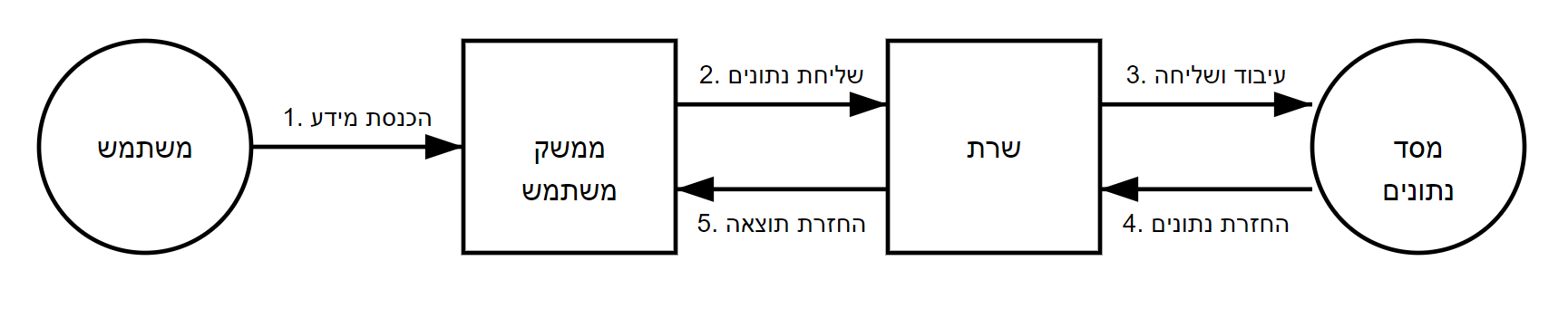
1. משתמש נרשם או מתחבר לאתר.
2. השרת מקבל את הבקשה ומבצע validation של הנתונים.
3. האתר מוביל את המשתמש לנתונים האחרונים שלו.
4. משם הוא בוחר את האפשרות שבו הוא רוצה לקדם או לאבחן.
5. המשתמש חווה גלישה מרהיבה ובודק את הראייה שלו.
6. אם יש למשתמש טעות בראייה – ניזון בערכת ומעדכן את הפרטים של המשתמש.



**9.3 תרשים זרימת נתונים (Data Flow Diagram)**

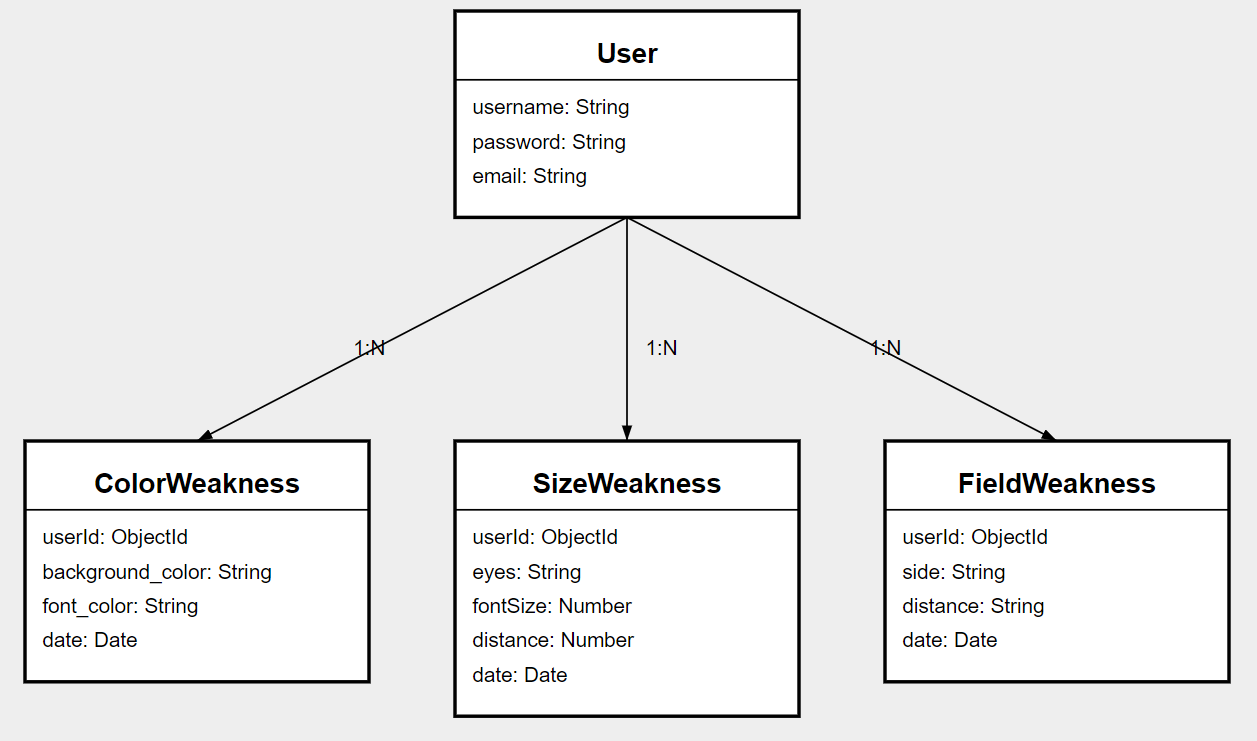
תרשים זרימת הנתונים מציג את הזרימה של נתונים בתוך המערכת:

1. משתמש מכניס מידע לממשק המשתמש (Client-Side).
2. הנתונים נשלחים לשרת (Server-Side).
3. השרת מבצע עיבוד נתונים ושולח אותם למסד הנתונים.
4. מסד הנתונים מחזיר את הנתונים המבוקשים לשרת.
5. השרת מעבד את הנתונים ומחזיר את התוצאה לממשק המשתמש.



#### **9.4**Entity-Relationship Diagram

* תרשים יחסי הגומלין והקשרים בין הטבלאות והיישויות השונות במערכת:



10 תיאור המרכיב האלגוריתמי – חישובי

* 1. **רישום משתמש ואימות**  
     - בעיה: משתמשים צריכים ליצור חשבונות ולגשת לנתונים שלהם בצורה מאובטחת.  
     - פתרון: הטמעת אלגוריתמים לרישום משתמש המאמתים סיסמאות קלט ו-hash. השתמשות באלגוריתמי אימות כדי לאמת אישורים במהלך הכניסה.
  2. **אחסון וניהול נתונים**  
     - בעיה: אחסון ואחזור ביעילות נתוני משתמש, כולל חולשות ראייה.  
     - פתרון: השתמשות בפעולות CRUD לניהול נתוני משתמש במסד נתונים.ליצור קשרים בין משתמשים וחולשותיהם באמצעות מפתחות זרים.
  3. **הערכת אבחון ראייה**  
     - בעיה: הערכת חולשות הראייה של המשתמשים על סמך הקלט שלהם.  
     - פתרון: פיתוח אלגוריתמי אבחון המנתחים את תגובות המשתמשים לבדיקות ראייה.
  4. **המלצות מותאמות אישית**  
     - בעיה: סיפוק תרגילים מותאמים כדי לעזור למשתמשים לשפר את הראייה שלהם.  
     - פתרון: יצירת מנוע המלצות שיוצר תרגילים מותאמים אישית המבוססים על נתוני משתמשים ומשוב, תוך חידוד הצעות לאורך זמן.

1. **ממשק משתמש וניסיון**  
   - בעיה: הקפדה על ממשק אינטואיטיבי וידידותי למשתמש.  
   - פתרון: הטמעת אלגוריתמים של מעקב אינטראקציות לאיסוף ניתוחים וטעינה דינמית של תוכן בהתבסס על העדפות המשתמש לחוויה מותאמת אישית.
2. **דיווח וניתוח**  
   - בעיה: סיפוק תובנות לגבי בריאות הראייה וההתקדמות של המשתמשים.  
   - פתרון: פיתוח אלגוריתמים לניתוח נתונים כדי לעקוב אחר מגמות לאורך זמן ולהפיק טבלאות המשקפים את ההתקדמות של המשתמשים ותחומים לשיפור.

**סיכום**  
האתר כולל אלגוריתמים לניהול משתמשים, טיפול בנתונים, הערכת חזון, המלצות מותאמות אישית, עיצוב ממשק משתמש, אמצעי אבטחה ודיווח. כל הרכיבים עובדים יחד כדי לספק פלטפורמה מקיפה ומאובטחת לאבחון ושיפור הראייה.

11. תיאור/התייחסות לנושאי אבטחת מידע

* גישה לחשבונות משתמשים על ידי שימוש בשם משתמש וסיסמא
* גישה לחשבון מנהל על ידי חלוקה לרמות הרשאה
* הגנה מפני מתקפות MONGO Injection בעזרת מסנן קלט משתמש.
* שמירת נתוני משתמש מוצפנים בטוקן בתוך עוגיות בדפדפן
* הענקת תוקף קצר לטוקנים
* הצפנת סיסמאות בבסיס הנתונים

## 12.משאבים הנדרשים לפרויקט:

#### **שעות עבודה:**

* תכנון כ-10 שעות
* פיתוח אתר: כ-200 שעות
* עיצוב ממשק משתמש: כ-40 שעות
* בדיקות ותחזוקה: כ-60 שעות

#### **ציוד נדרש:**

#### מחשבי פיתוח עם חיבור אינטרנט

* שרת לפיתוח והטמעה

#### **תוכנות נדרשות:**

* Visual Studio Code
* Node.js, React, NEXT.JS
* Mongo
* github desktop

#### **ידע נדרש ללימוד:**

* אבטחת מידע באתרי אינטרנט
* ספריית MUI
* ספריית TAILWIND
* ספריית framer-motion

#### **ספרות ומקורות מידע:**

* GOOGLE
* דוקומנטציה של MUI
* STACK OVERFLOW

13. תכנית עבודה ושלבים למימוש הפרויקט

#### **שלב 1: ניתוח דרישות ותכנון מפורט**

* איסוף דרישות מהלקוח
* ניתוח דרישות והגדרת מקרי שימוש - Use Cases
* תכנון מפורט של ממשק המשתמש, מסכים ותזרים עבודה
* תכנון מפורט של מבנה מסד הנתונים
* תכנון ארכיטקטורת המערכת
* תכנון חלוקת עבודה בצוות ולו"ז מפורט

#### **שלב 2: פיתוח ראשוני**

* הקמה ראשונית של השרתים וסביבת הפיתוח
* פיתוח מוקדם של מסך כניסה והרשמה
* יצירת מסד נתונים ראשוני עם טבלאות בסיסיות
* פיתוח בסיסי של ממשק ניהול תוכן פשוט

#### **שלב 3: פיתוח מלא של מודולים עיקריים**

* פיתוח מלא של מודול ניהול משתמשים
* פיתוח מלא של מודול ניהול חולשות הראייה
* השלמת מסד נתונים עם כל הטבלאות והקשרים

#### **שלב 4: פיתוח ממשק משתמש מלא**

* עיצוב גרפי מלא של כל המסכים
* יישום רספונסיביות מלאה
* פיתוח ממשק משתמש מלא עם קישור למודולים
* בדיקות ממשק משתמש ותיקון באגים

#### **שלב 5: בדיקות ושליפת באגים**

* בדיקות יחידה ואינטגרציה של כל המודולים
* בדיקות מערכת כולל מקרי קצה
* תיקון באגים וייצוב המערכת
* בדיקות ביצועים ועומסים