1. מבוא לפרויקט

תיאור כללי:

מערכת זו פותחה במטרה לספק פתרון חכם לניהול חולים והמלצות רפואיות באמצעות אינטגרציה של בינה מלאכותית. המערכת מאפשרת למשתמשים לקבל המלצות מותאמות אישית לבדיקות רפואיות, בהתאם לנתוניהם האישיים, ההיסטוריה הרפואית והנחיות קליניות עדכניות.

המערכת כוללת ממשק למטופלים לצורך רישום וניהול המידע האישי שלהם, וכן ממשק נפרד למנהלים רפואיים, המאפשר מעקב אחר חולים, ניתוח נתונים, והפקת דוחות מותאמים אישית.

הפלטפורמה מבוססת על טכנולוגיות עדכניות, כולל **MongoDB** לניהול מסד הנתונים, **Node.js** בצד השרת, js,css,html ו bootstrap Icons

בצד הלקוח. כמו כן, שולב בינה מלאכותית לשיפור תהליך קבלת ההחלטות הרפואיות, באמצעות מודלים חכמים של ניתוח נתונים.

המערכת נועדה לייעל את תהליכי האבחון והמעקב, להקל על צוותים רפואיים, ולהציע חוויית שימוש ידידותית ואינטואיטיבית למשתמשים.

רקע:

בעידן הדיגיטלי, תחום הרפואה עובר מהפכה משמעותית עם כניסתן של מערכות מבוססות בינה מלאכותית, שמטרתן לשפר את איכות השירות הרפואי ולהפחית עומסים על צוותים רפואיים. הצורך בניהול חכם של מידע רפואי ומתן המלצות מותאמות אישית הופך לקריטי, במיוחד כאשר מערכות הבריאות מתמודדות עם גידול בכמות המטופלים וצורך באבחון מהיר ומדויק.

מערכת זו פותחה במטרה לתת מענה לצורך זה באמצעות שילוב של טכנולוגיות מתקדמות לניהול נתונים, ניתוח מידע רפואי, ומתן המלצות מבוססות AI. היא מספקת ממשק ידידותי למשתמשים המאפשר להם להזין נתונים רפואיים, לקבל תובנות רלוונטיות, ולעקוב אחר מצבם הבריאותי.

יתר על כן, המערכת מספקת למנהלים רפואיים וכלל הצוותים הרפואיים גישה לכלים מתקדמים לניהול חולים, מעקב אחר נתונים סטטיסטיים, וזיהוי מגמות רפואיות. בכך, היא מאפשרת למוסדות רפואיים לשפר את תהליכי קבלת ההחלטות ולייעל את השירות הרפואי הניתן למטופלים.

מטרות הפרויקט:

מטרת הפרויקט היא לפתח מערכת רפואית חכמה המשלבת בינה מלאכותית לצורך ניתוח נתונים רפואיים והמלצה על בדיקות רפואיות מתאימות. המערכת נועדה לסייע הן למטופלים והן לצוותים רפואיים בקבלת החלטות רפואיות מבוססות נתונים, תוך שיפור תהליכי האבחון והמעקב הרפואי.

באמצעות טכנולוגיות מתקדמות, המערכת מאפשרת:
מתן המלצות מותאמות אישית – בהתאם להיסטוריה הרפואית ולפרופיל הבריאותי של המשתמש.
ייעול תהליכי עבודה רפואיים – הפחתת העומס על רופאים וצוותים רפואיים על ידי ניתוח אוטומטי של נתונים.
שיפור ניהול המידע – ארגון נתונים רפואיים בצורה חכמה ונגישה למטופלים ולמנהלים רפואיים.
שימוש באינטגרציה חכמה – חיבור בין מסדי נתונים רפואיים, API חיצוניים ואלגוריתמים .

המטרה הסופית של המערכת היא לספק כלי עוצמתי שיתרום לבריאות הציבור, ישפר את איכות חייהם של המטופלים, ויאפשר למערכת הבריאות להתמודד בצורה טובה יותר עם האתגרים הקיימים.

2. ניתוח דרישות

- **דרישות פונקציונליות:**

מערכת מאפשרת למשתמש:

מנהל :

- הוספת מרפאה חדשה
- עדכון ומחיקת מרפאות קיימות
- הוספת תחום רפואי חדש
- עדכון ומחיקת תחומי רפואה קיימים
- הוספת משתמש חדש
- עדכון ומחיקת משתמש קיים
- זימון תור חדש
- ביטול תור קיים
- הוספת רופא חדש
- מחקית ועדכון רופא קיים
- לענות על פניות שנשלחו ממטופלים
- למחוק / לעדכן סטטוס הפיה

מטופל :

- כניסה למערכת
- הרשמה למערכת
- צפיה בנתונים אישיים
- עריכת נתונים אישיים

- צפייה בהיסטוריה שיחות, ביקורים,
- קבלת המלצות רפואיות
- יצירת קשר
- צפיה בנתוני הפניה
- בוט AI מבצעת ניתוח נתונים של המטופל
- ניהול שיחה עם בוט AI:
  - שאלות מותאמות (אבחון, תחושות, מצבו הראשוני תסמינים)
  - הכוונה
  - המלצות רפואיות
- התחברות למערכת עם אימות (מבוסס jwt)
- הרשמה למערכת כולל הצפנה (hashing) לסיסמה
- שמירת נתונים למסד הנתונים (תורים, שיחות, משתמשים וכו...)

- **דרישות לא פונקציונליות:**

1. ביצועים (Performance)

המערכת תספק תגובה מהירה לכל פעולה, עם זמן טעינה ממוצע של פחות מ-3 שניות לכל בקשה.
המערכת תתמוך בעיבוד נתונים בזמן אמת לצורך הצגת המלצות מהירות.

2. אבטחה (Security)

הצפנת נתונים לצורך אבטחת מידע המטופלים.
אימות משתמשים באמצעות הרשאות וגישה מבוססת תפקידים .

3. זמינות (Availability)

המערכת תהיה זמינה 24/7 .

4. יכולת הרחבה (Scalability)

המערכת תוכל לתמוך במספר גדול של משתמשים פעילים בו-זמנית ללא פגיעה בביצועים.

5. שימושיות (Usability)

הממשק יהיה אינטואיטיבי וידידותי למשתמש, עם עיצוב נגיש ונוח.
תמיכה ברזולוציות שונות ומכשירים מגוונים (Responsive Design).

- **דרישות טכנולוגיות:**

1. צד שרת (Backend)

Node.js – לניהול השרת וה-API.
MongoDB – כבסיס נתונים NoSQL לאחסון וניהול מידע רפואי.
Mongoose – לניהול סכמות הנתונים ולשאילתות מול מסד הנתונים.
JWT (JSON Web Token – לאימות משתמשים וניהול הרשאות גישה.
bcrypt – להצפנת סיסמאות משתמשים.

2. צד לקוח (Frontend)

full stack– לפיתוח ממשק משתמש דינמי ואינטראקטיבי.
Fetch API – לשליחת בקשות API לשרת.

3. אינטגרציה עם AI
Gemini AI – לחיבור המודלים של הבינה המלאכותית עם ה-Backend.

4. אבטחת מידע ((Security
COR – לניהול הרשאות גישה ל-API.

5. Version Control:

Git, GitHub – לניהול קוד, שיתוף פעולה בין מפתחים, וניהול גרסאות המערכת.

3. תכנון ראשוני – ארכיטקטורת המערכת

```
/frontend
├── /ai-bot (בוט ה-AI)
│   ├── bot.html
│   ├── bot.css
│   ├── bot.js
│
├── /appointment-details (פרטי תור)
│   ├── appointment-details.html
│   ├── appointment-details.css
│   ├── appointment-details.js
│
├── /authorization (אזור הרשאות)
│   ├── authorized.html
│
├── /dashboard-user (דאשבורד למשתמש)
│   ├── dashboard-user.html
│   ├── dashboard-user.css
│   ├── dashboard-user.js
│   ├── contact.html
│   ├── myContact.html
│
├── /dashboard (דאשבורד לניהול)
│   ├── dashboard.html
│   ├── dashboard.css
│   ├── dashboard.js
│   ├── deleteItem.js
│   ├── modal-details.css
│   ├── table.css
│
├── /appointments (ניהול תורים)
│
├── /clinics (ניהול מרפאות)
│   ├── clinics.html
│   ├── add-new-clinic.html
```

```
│
├── medical-field/ (ניהול תחומים רפואיים)
│   ├── medical-fields.html
│   ├── add-new-medicalfield.html
│
├── success-modal/ (מודל הצלחה)
│   ├── success-modal.css
│   ├── successModal.js
│
├── users/ (ניהול משתמשים)
│   ├── AdmnContacts.html
│
├── doctor/ (עמוד מידע לרופא)
│   ├── doctor.html
│   ├── doctor.css
│   ├── doctor.js
│
├── error/ (עמוד שגיאה)
│   ├── html.404
│
├── history/ (היסטוריית תורים)
│   ├── history.html
│   ├── history.css
│   ├── history.js
│
├── home/ (עמוד ראשי)
│   ├── home.html
│   ├── home.css
│   ├── home.js
│
├── login/ (עמוד התחברות)
│   ├── login.html
│   ├── login.css
│   ├── login.js
│
├── medical-appointment/ (זימון תור רפואי)
│   ├── medical-appointment.html
│   ├── medical-appointment.css
│   ├── medical-appointment.js
│
├── patient/ (פרטי מטופל)
```

```
│   ├── patient.html
│   ├── patient.css
│   ├── patient.js
│
├── register/ (עמוד הרשמה)
│   ├── register.html
│   ├── register.css
│   ├── register.js
│
├── search-info/ (חיפוש מידע)
│   ├── search-info.html
│   ├── search-info.css
│   ├── search-info.js
│
├── user-info/ (פרטי משתמש)
│   ├── user-info.html
│   ├── user-info.css
│   ├── user-info.js
│
├── assets/ (קבצים משותפים)
│   ├── logo.png
│   ├── navbar.css
│   ├── styles.css
│   ├── script.js
│   ├── template.html
│
├── auth.js (קובץ לאימות משתמשים)
├── .gitignore (התעלמות מקבצים שלא רוצים להעלות ל-Git)
```

- **מבנה צד השרת (Backend):**

```
├── backend/
│   ├── config/ (mongodb קונפיגורציה לחיבור למסד הנתונים)
│   │   ├── connectToDB.js
│   ├── models/
│   │   ├── User.js (כל משתמש – אדמין, מטופל ורופא)
│   │   ├── Appointment.js (תור)
│   │   ├── ChatLogs.js (שיחות עם הבוט)
│   │   ├── Clonics.js (מרפאות)
│   │   ├── MedicalFiled.js (תחום רפואי)
│   ├── routes/
│   │   ├── authRoutes.js (נתיבים להתחברות והרשמה)
│   │   ├── userRoutes.js (נתיבים למשתמש)
│   │   ├── appointmentRoutes.js (נתיבים לתור – זימון, ביטול, עדכון)
│   │   ├── botRoutes.js (נתיבים לשיחות עם הבוט)
│   │   ├── authClinics (נתיבים למרפאות)
│   │   ├── contactRoutes.js (נתיבים ליצירת קשר)
│   │   ├── dashboardRoutes.js (נתיבים לדאשבורד)
│   ├── middleware/
```

```
│   │   ├── error.js
│   │   ├── validdataObjectId.js.js
│   │   ├── verfyToken.js
│   ├── /controllers
│   │   ├── authController.js (בקרת הרשאות – רישום, התחברות, אימות)
│   │   ├── userController.js (בקרת משתמשים – צפייה, עריכה, מחיקה)
│   │   ├── appointmentController.js (בקרת תורים – יצירה, ביטול, עדכון)
│   │   ├── chatLogsController.js (בקרת ניהול שיחות עם הבוט)
│   │   ├── Controller.jsClinics(בקרת הרשאות – מרפאות)
│   │   ├── contactController.js (יצירת קשר)
│   │   ├── dashboardController.js (דאשבורד)
│   │   ├── chatLogsController.js (תחום רפואי)
│   ├── /services
│   │   ├── botServices.js
│   ├── index.js  (קובץ ראשי להפעלת השרת, חיבור למסד הנתונים והגדרות כלליות)
│   ├── .env (קובץ של משתני סביבה – מכיל את הקישור למסד הנתונים ו-secret_key של JWT)
├── package.json (מידע על הפרויקט – מכיל תלויות וגרסאות של הספריות)
├── package-lock.json (מידע על גרסאות מותאמות של הספריות שהותקנו)
```

- **מסד הנתונים (Database):** תיאור המבנה של ה-Collections ב-MongoDB.
- **אינטגרציית הבוט (AI):** חיבור ל-API (כגון Hugging Face/OpenAI).
- **מערכת ההתחברות (Authentication):** שימוש ב-JWT לאימות מאובטח.

4. חלוקת עבודה ותוכנית זמנים (20 שעות)

סך כל השעות: 20 שעות

- **שעה 1:** הגדרת פרויקט ב-GitHub וחלוקת משימות.
- **שעה 2-4:** בניית מבנה בסיסי של Frontend.
- **שעה 5-7:** פיתוח צד השרת (API) ב-Node.js.
- **שעה 8-10:** יצירת מסד נתונים ובדיקת חיבורים ל-MongoDB.
- **שעה 11-13:** שילוב הבוט הרפואי דרך API חיצוני.

- **שעה 14-16:** פיתוח מערכת ההתחברות עם JWT.
- **שעה 17-18:** שילוב כל החלקים יחד (Full Integration).
- **שעה 19:** ביצוע בדיקות מערכת (Testing).
- **שעה 20:** העלאת הפרויקט ל-GitHub והכנת מצגת להצגה.

📌 **גישה 1: חלוקה לפי שכבות (Layers) – קלאסית ומסודרת**

יתרונות: מיקוד והתמקצעות בכל שכבה, קל לשלב את החלקים בסיום.
חסרונות: דורשת תאום חזק בין כל השכבות.

- **אחראי Frontend (ממשק המשתמש):** בניית עמודי האתר, כתיבת קוד ב-HTML, CSS, ו-JavaScript.
- **אחראי Backend (שרת ולוגיקה):** פיתוח ה-API ב-Node.js, חיבור למסד הנתונים והגדרת הנתיבים (routes).
- **אחראי מסד נתונים (Database):** יצירת ה-Collections והגדרת הקשרים בין הנתונים ב-MongoDB.
- **אחראי אינטגרציית AI:** חיבור ל-API של הבוט והטמעת הלוגיקה מול השרת.
- **אחראי אבטחה ואימות משתמשים:** פיתוח מערכת אימות (JWT) ושמירת נתונים מאובטחת.

🚀 **גישה 2: חלוקה לפי תכונות (Features) – אג'ילית וממוקדת בתוצרים**

יתרונות: כל סטודנט אחראי על תכונה מלאה – רואים תוצאה מהירה.
חסרונות: דורשת הבנה רחבה של כל החלקים (Frontend + Backend + DB).

- **תכונת ההתחברות והרשמה:** אחראי על מסך ההתחברות וההרשמה כולל אימות (Frontend + Backend + Database).
- **תכונת ניהול התורים:** אחראי על הזמנה, צפייה וביטול תורים (Frontend + Backend + Database).
- **תכונת הבוט הרפואי:** אחראי על השילוב של הבוט מול הלקוח והשרת (Frontend + Backend + AI Integration).
- **תכונת ניהול משתמשים (Admin):** אחראי על מסכי ניהול משתמשים ומעקב אחר היסטוריית תורים.

🔁 **גישה 3: חלוקה לפי תהליכים (Processes) – גישת DevOps קלה**

יתרונות: עבודה מקבילית עם חלוקה טבעית לפי שלבי הפיתוח.
חסרונות: דורשת הבנה מעמיקה של כל התהליך אצל כל אחד מהסטודנטים.

- **צוות איסוף הדרישות ותכנון:** אחראי על כתיבת מסמכי דרישות ותכנון ארכיטקטורה בסיסית.
- **צוות הפיתוח (Coding):** אחראי על כתיבת הקוד לכל החלקים – Frontend ו-Backend.

- **צוות האינטגרציה והבדיקות (Integration & Testing):** אחראי על חיבור החלקים, בדיקות ואימותים.
- **צוות הפריסה (Deployment):** אחראי על העלאת הפרויקט ל-GitHub והכנת מצגת הסיום.

⚙️ **גישה 4: חלוקה לפי מומחיות (Expertise) – מותאמת ליכולות הצוות**

יתרונות: כל סטודנט מתמקד במה שהוא הכי טוב בו.
חסרונות: תלות גבוהה בכל סטודנט, פחות למידה רב-תחומית.

- **מומחה Frontend:** אחראי על עיצוב הממשק וחוויית המשתמש.
- **מומחה Backend:** אחראי על כתיבת ה-API וניהול התקשורת מול מסד הנתונים.
- **מומחה Database:** בונה את מסד הנתונים ומנהל את כל שאילתות המידע.
- **מומחה אבטחה:** מתמקד במערכת האימות ובשמירה על אבטחת הנתונים.
- **מומחה AI:** מתמחה באינטגרציה עם הבוט ובניתוח תשובות ה-AI.

💡 **המלצה:**

בפרויקט קצר של 20 שעות, **גישה 2 (חלוקה לפי תכונות) או גישה 4 (לפי מומחיות)** הן היעילות ביותר. הן מאפשרות לעבוד במקביל ולהציג תוצרים מהר.

5. חלוקת תפקידים בצוות

- **Frontend (עמודים והאינטראקציה):** ירין /רונן/אייבק/מוחמד /מוסא
- **Backend (API והלוגיקה):** ירין /רונן/אייבק/מוחמד
- **מסד נתונים (MongoDB):** שם הסטודנט
- **אינטגרציית AI (API):** ירין /רונן/אייבק
- **אחראי ניהול גרסאות (GitHub):** רונן

6. קריטריונים להצלחה

- **פונקציונליות מלאה של המערכת:** הזמנה, ביטול תורים וייעוץ רפואי באמצעות הבוט.
- **עמידה בזמנים:** סיום הפרויקט ב-20 שעות בלבד.
- **עבודה צוותית נכונה:** ביצוע Code Reviews והגשת Pull Requests.
- **איכות הקוד:** שימוש בעקרונות מודולריות, קריאות ושמירה על אבטחת מידע.
- **תיעוד:** כתיבת הערות בקוד ותיעוד בסיסי.

7. אתגרים צפויים ופתרונות מוצעים

- **בעיית חיבור ל-API:** בדיקה עם Postman לפני השילוב בקוד.
- **עיכוב בעבודה בצוות:** קיום פגישות קצרות בתחילת ובסוף כל שלב.
- **בעיות בממשק המשתמש:** בדיקות רציפות וקבלת פידבק מהצוות.

8. מסקנות והצעות לשיפור עתידי

מה למדנו בתהליך הפיתוח?

במהלך פיתוח המערכת, למדנו כיצד לשלב בין טכנולוגיות שונות ליצירת מערכת רפואית חכמה ויעילה. התמודדנו עם אתגרים כמו ניהול מידע רפואי בצורה מאובטחת, יצירת אלגוריתמים מבוסס AI למתן המלצות, ושיפור חוויית המשתמש במערכת. בנוסף, הבנו את החשיבות של תכנון מקדים, עבודה בצוות, וניהול גרסאות בצורה מסודרת.

המלצות לשיפורים עתידיים:

1. **שיפור חוויית המשתמש (UX/UI)** – התאמת הממשק למשתמשים שונים (מטופלים, רופאים, מנהלים) ושיפור נוחות השימוש.
2. **הרחבת יכולות הבינה המלאכותית** – שילוב אלגוריתמים מתקדמים יותר לצורך דיוק גבוה יותר בהמלצות רפואיות וגם שיעבוד על כל המחלות האפשריים .
3. **תמיכה בשפות נוספות** – הוספת תמיכה בממשק רב-לשוני להנגשת המערכת למשתמשים בינלאומיים.
4. **פיתוח אפליקציה ניידת** – יצירת גרסת מובייל שתאפשר למשתמשים גישה נוחה למידע הרפואי מכל מקום.
5. **שיפור האבטחה** – הוספת אימות דו-שלבי (2FA) והקשחת הגנות למניעת פריצות וגניבת מידע.
6. **אינטגרציה עם מערכות רפואיות חיצוניות** – חיבור למערכות מידע בבתי חולים ובמרפאות לצורך שיתוף נתונים בצורה מאובטחת.
7. **ניתוח נתונים מתקדם** – פיתוח דוחות רפואיים וגרפים להצגת מגמות בריאותיות בזמן אמת.