### ****1. מבוא לפרויקט****

* **תיאור כללי:** הצגת מטרת הפרויקט בקצרה והפתרון המוצע.
* **רקע:** הסבר על הצורך במערכת ניהול תורים רפואיים עם בוט AI.
* **מטרות הפרויקט:** מה הסטודנטים מתכוונים להשיג בטווח הזמן הקצר של 20 שעות.

### ****2. ניתוח דרישות****

* **דרישות פונקציונליות:** רשימת הפונקציות המרכזיות של המערכת (למשל הזמנת תור, ביטול תור, ייעוץ רפואי ראשוני וכו').
* **דרישות לא פונקציונליות:** כמו אבטחת מידע, ביצועים ועמידה בעקרונות ניהול גרסאות (GitHub).
* **דרישות טכנולוגיות:** הטכנולוגיות המותרות על פי ההנחיות מהקורס.

### ****3. תכנון ראשוני – ארכיטקטורת המערכת****

* **מבנה צד הלקוח (Frontend):** הסבר קצר על עמודי הממשק ותפקודם (HTML, CSS, JavaScript).
* **מבנה צד השרת (Backend):** תיאור המבנה והתפקידים של Node.js.
* **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):** שם הסטודנט
* **Backend (API והלוגיקה):** שם הסטודנט
* **מסד נתונים (MongoDB):** שם הסטודנט
* **אינטגרציית AI (API):** שם הסטודנט
* **אחראי ניהול גרסאות (GitHub):** שם הסטודנט

### ****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 הסטודנטים למדו בתהליך הפיתוח הקצר.
* המלצות לשיפורים במערכת להמשך הפיתוח.