דוח פרויקט

1/9/2024

מערכת לניהול קופת חולים

מגישים
ירדן צרי 318774932
נתנאל נייזוב 325336303

הקדמה:

במערכת הבריאות המודרנית, יש חשיבות רבה לניהול יעיל של קופת חולים. המערכת המוצגת בפרויקט שלנו פותחה במטרה לספק פתרון מקיף לניהול שירותי קופת חולים, המשלב יכולות מגוונות לניהול תורים, רופאים, תרופות, מטופלים ובתי מרקחת. המערכת כוללת מספר מבנים עיקריים, ביניהם:

- **Appointment** - לניהול התורים והפגישות.
- **Date** ו **Time** -לניהול תאריכים ושעות של תורים ופעילויות שונות.
- **Doctor** - לניהול פרטי רופאים והתמחויות.
- **Drug** - לניהול מלאי תרופות (קיום תרופה במלאי ללא התייחסות לכמות) ומידע רלוונטי.
- **HMO_Admin** – מנהל קופת החולים בעת הקמתה.
- **Patient** - לניהול פרטי מטופלים, רישום תורים, היסטוריה רפואית ועוד.
- **Pharmacy** ו **Pharmacist** -לניהול בתי מרקחת ורוקחים העובדים בהם.
- **Prescription** - לניהול מרשמים רפואיים למטופלים.
- **HMO** – המבנה הראשי של המערכת שאצלו מנהלים את כל המערכת.

המערכת מתפקדת כמערכת אינטגרטיבית שמטרתה לשפר את היעילות התפעולית של קופות החולים, ולאפשר לצוות הרפואי והמנהלי לנהל את המשאבים בצורה אופטימלית. המערכת מאפשרת להוסיף, לערוך ולמחוק רשומות של מטופלים, רופאים, רוקחים ותרופות, לחפש ולמיין נתונים לפי פרמטרים שונים, ובכך מסייעת בקבלת החלטות רפואיות ובמתן שירות איכותי למטופלים.

באמצעות מערכת זו, ניתן לממש את החזון של שירותי בריאות דיגיטליים מתקדמים, המשלבים טכנולוגיה וניהול חכם לשיפור חוויית המטופל והאיכות הטיפולית.

מבני המערכת:

מבנה HMO :

המבנה HMO משמש לייצוג קופת חולים במערכת ניהול קופת חולים. המבנה מכיל תכונות שמייצגות את פרטי קופת החולים, כולל כתובת, רשימת רופאים, רשימת מטופלים, רשימת פגישות, בית מרקחת, מנהל קופת החולים, ורשימת מרשמים. בנוסף, המבנה כולל פעולות לניהול ותפעול נתוני קופת החולים במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- כתובת (char* address) – מחרוזת המייצגת את הכתובת של קופת החולים.
- מערך רופאים (Doctor* doctors) - מערך של מצביעים למבני Doctor המייצג את רשימת הרופאים העובדים בקופת החולים.
- מערך מצביעים כפולים למטופלים (Patient** patients) - מערך של מצביעים כפולים למבני Patient , המייצג את רשימת המטופלים הרשומים בקופת החולים.
- מספר הרופאים (int numDoctors) - מספר שלם המייצג את מספר הרופאים העובדים בקופת החולים.
- מספר המטופלים (int numPatients) - מספר שלם המייצג את מספר המטופלים הרשומים בקופת החולים.
- רשימת פגישות (NODE* appointmentList) - מצביע לרשימה מקושרת המייצגת את כל הפגישות המתוזמנות בקופת החולים.
- מספר פגישות (int appointmentsNum) - מספר שלם המייצג את מספר הפגישות שנקבעו בקופת החולים.
- בית מרקחת (Pharmacy* pharmacy) - מצביע למבנה Pharmacy המייצג את בית המרקחת הקשור לקופת החולים.
- מנהל קופת חולים (HMO_Admin admin) - מבנה HMO_Admin המייצג את המנהל של קופת החולים.
- רשימת מרשמים (PrescriptionNode* prescriptionListHead) - מצביע לראש הרשימה המקושרת של המרשמים בקופת החולים.
- זנב רשימת מרשמים (PrescriptionNode* prescriptionListTail) - מצביע לזנב הרשימה המקושרת של המרשמים בקופת החולים.

הפעולות הקיימות במחלקה:

- initHMO(HMO** hmo) - פעולה לאתחול מבנה הHMO, הקצאת זיכרון ואיפוס ערכים.
- freeHMO(HMO** hmo) - פעולה לשחרור כל ההקצאות שהוקצו למבנה הHMO כולל שחרור זיכרון לרשימות רופאים, מטופלים, ופגישות.
- printHMO(HMO* hmo) - פעולה להצגת כל המידע הקיים בקופת החולים כולל רופאים, מטופלים ופגישות.

- printHMODoctors(HMO* hmo) - פעולה להצגת כל הרופאים הקיימים בקופת החולים.
- printHMOPatients(HMO* hmo) - פעולה להצגת כל המטופלים הרשומים בקופת החולים.
- addNewDoctorToHMO(HMO* hmo, Doctor* doctor) - פעולה להוספת רופא חדש לקופת החולים.
- *addNewPatientToHMO(HMO* hmo, Patient* patient)* - פעולה להוספת מטופל חדש לקופת החולים.
- printAppointmentsInHMO(const HMO* hmo) - פעולה להצגת כל הפגישות המתוזמנות בקופת החולים.
- createAndAddAppointmentToHMO(HMO* hmo, Appointment* newAppointment) - פעולה ליצירה והוספת פגישה חדשה בקופת החולים.
- searchDoctorByIdInHMO(const HMO* hmo, int id) - פעולה לחיפוש רופא לפי מזהה בתוך רשימת הרופאים של קופת החולים.
- searchPatientByIdInHMO(const HMO* hmo, int id) - פעולה לחיפוש מטופל לפי מזהה בתוך רשימת המטופלים של קופת החולים.
- initPrescription(const HMO* hmo, Prescription* pr) - פעולה לאתחול מרשם רפואי בתוך קופת החולים.
- printPrescription(const Prescription* pr, HMO* hmo) - פעולה להצגת מידע על מרשם רפואי מסוים.
- addPrescriptionToList(HMO* hmo, Prescription* pr) - פעולה להוספת מרשם רפואי חדש לרשימת המרשמים של קופת החולים.
- printPrescriptionList(const HMO* hmo) - פעולה להצגת כל המרשמים בקופת החולים.
- freePrescriptionList(HMO* hmo) - פעולה לשחרור כל הזיכרון של רשימת המרשמים.
- printPatientAppointments(const HMO* hmo, int id) - פעולה להצגת כל הפגישות שנקבעו למטופל מסוים לפי מזהה.
- buyDrugByPrescription(HMO* hmo) - פעולה לרכישת תרופה באמצעות מרשם רפואי בקופת החולים.
- createAppointmentApprovalFile(HMO* hmo) - פעולה ליצירת קובץ אישור פגישה עבור קופת החולים.
- writeHMOToBinaryFile(const HMO* hmo, const char* filename) - פעולה לכתיבת כל המידע של קופת החולים לקובץ בינארי.
- writeDoctorToBinaryFile(const Doctor* doctor, FILE* file) - פעולה לכתיבת מידע על רופא לקובץ בינארי.
- writePatientToBinaryFile(const Patient* patient, FILE* file) - פעולה לכתיבת מידע על מטופל לקובץ בינארי.

- writeAppointmentsToBinaryFile(NODE* appointmentList, FILE* file) - פעולה לכתיבת רשימת הפגישות לקובץ בינארי.
- writePharmacyToBinaryFile(const Pharmacy* pharmacy, FILE* file) - פעולה לכתיבת מידע על בית מרקחת לקובץ בינארי.
- writeDrugToBinaryFile(const Drug* drug, FILE* file) - פעולה לכתיבת מידע על תרופה לקובץ בינארי.
- writePharmacistToBinaryFile(const Pharmacist* pharmacist, FILE* file) - פעולה לכתיבת מידע על רוקח לקובץ בינארי.
- writePrescriptionsToBinaryFile(PrescriptionNode* head, FILE* file) - פעולה לכתיבת רשימת המרשמים לקובץ בינארי.
- writeHMOAdminToBinaryFile(const HMO_Admin* admin, FILE* file) - פעולה לכתיבת מידע על מנהל קופת חולים לקובץ בינארי.
- readHMOFromBinaryFile(HMO** hmo, const char* filename) - פעולה לקריאת כל המידע של קופת החולים מקובץ בינארי.
- readDoctorFromBinaryFile(Doctor* doctor, FILE* file) - פעולה לקריאת מידע על רופא מקובץ בינארי.
- readPatientFromBinaryFile(Patient* patient, FILE* file) - פעולה לקריאת מידע על מטופל מקובץ בינארי.
- readAppointmentsFromBinaryFile(NODE** appointmentList, FILE* file) - פעולה לקריאת רשימת הפגישות מקובץ בינארי.
- readPharmacyFromBinaryFile(Pharmacy* pharmacy, FILE* file) - פעולה לקריאת מידע על בית מרקחת מקובץ בינארי.
- readDrugFromBinaryFile(Drug* drug, FILE* file) - פעולה לקריאת מידע על תרופה מקובץ בינארי.
- readPharmacistFromBinaryFile(Pharmacist* pharmacist, FILE* file) - פעולה לקריאת מידע על רוקח מקובץ בינארי.
- readPrescriptionsFromBinaryFile(PrescriptionNode** head, PrescriptionNode** tail, FILE* file) - פעולה לקריאת רשימת המרשמים מקובץ בינארי.
- readHMOAdminFromBinaryFile(HMO_Admin* admin, FILE* file) - פעולה לקריאת מידע על מנהל קופת חולים מקובץ בינארי.
- int writeHMOToTextFile(const HMO* hmo, const char* filename) - פעולה לכתיבת כל המידע של קופת החולים לקובץ טקסט.
- void writeDoctorToTextFile(const Doctor* doctor, FILE* file) -פעולה לכתיבת מידע על רופא לקובץ טקסט.
-

- void writePatientToTextFile(const Patient* patient, FILE* file) - פעולה לכתיבת מידע על מטופל לקובץ טקסט.
- void writeAppointmentsToTextFile(NODE* appointmentList, FILE* file - פעולה לכתיבת רשימת הפגישות לקובץ טקסט.
- void writePharmacyToTextFile(const Pharmacy* pharmacy, FILE* file) – פעולה לכתיבת מידע על בית מרקחת לקובץ טקסט.
- void writeDrugToTextFile(const Drug* drug, FILE* file) - פעולה לכתיבת מידע על תרופה לקובץ טקסט.
- void writePharmacistToTextFile(const Pharmacist* pharmacist, FILE* file) - פעולה לכתיבת מידע על רוקח לקובץ טקסט.
- void writePrescriptionsToTextFile(PrescriptionNode* head, FILE* file) - פעולה לכתיבת מידע על מרשם לקובץ טקסט.
- void writeHMOAdminToTextFile(const HMO_Admin* admin, FILE* file) - פעולה לכתיבת מידע על מנהל מרפאה לקובץ טקסט.
- int readHMOFromTextFile(HMO** hmo, const char* filename) - פעולה לקריאת כל המידע של קופת החולים מקובץ טקסט.
- void readDoctorFromTextFile(Doctor* doctor, FILE* file) - פעולה לקריאת כל המידע של רופא מקובץ טקסט.
- void readPatientFromTextFile(Patient* patient, FILE* file) - פעולה לקריאת כל המידע של מטופל מקובץ טקסט.
- void readAppointmentsFromTextFile(NODE** appointmentList, FILE* file) - פעולה לקריאת כל המידע של רשימת הפגישות מקובץ טקסט.
- void readPharmacyFromTextFile(Pharmacy* pharmacy, FILE* file) - פעולה לקריאת כל המידע של בית המרקחת מקובץ טקסט.
- void readDrugFromTextFile(Drug* drug, FILE* file) - פעולה לקריאת כל המידע של תרופה מקובץ טקסט
- void readPharmacistFromTextFile(Pharmacist* pharmacist, FILE* file) - פעולה לקריאת כל המידע של רוקח מקובץ טקסט.
- void readPrescriptionsFromTextFile(PrescriptionNode** head, PrescriptionNode** tail, FILE* file) – פעולה לקריאת כל המידע של מרשם מקובץ טקסט
- void readHMOAdminFromTextFile(HMO_Admin* admin, FILE* file) - פעולה לקריאת כל המידע של מנהל מרפאה מקובץ טקסט.

מבנה Doctor :

המבנה Doctor משמש לייצוג רופא במערכת ניהול קופת חולים. הוא מכיל תכונות המייצגות את המידע האישי והמקצועי של הרופא וכולל פעולות לניהול ותחזוקת נתוני הרופאים במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- שם הרופא (char* name) - מחרוזת המייצגת את שמו של הרופא.
- מזהה הרופא (int id) - מספר שלם המייצג מזהה ייחודי לרופא במערכת.
- מין הרופא (char gender) - תו המייצג את מין הרופא ('M' לזכר 'F' לנקבה).
- מזהה רישיון (char licence_id[L_ID_MAX]) - מחרוזת באורך קבוע המייצגת את מזהה הרישיון של הרופא.
- התמחות (eSpecialization eS) - משתנה מסוג eSpecialization המייצג את תחום ההתמחות של הרופא (למשל: קרדיולוגיה, נוירולוגיה, וכו').

הפעולות הקיימות במחלקה:

- initDoctor(Doctor** doctor) - פעולה לאתחול מבנה Doctorכולל הקצאת זיכרון והגדרת ערכים ראשוניים.
- addDoctorToArray(Doctor** doctors, Doctor* newD, int size) - פעולה להוספת רופא חדש למערך של רופאים במערכת.
- printDoctor(const Doctor* doctor) - פעולה להצגת כל המידע הקיים על רופא מסוים.
- printDoctorBasicInfo(const Doctor* doctor) - פעולה להצגת מידע בסיסי (שם, מזהה, התמחות) על רופא מסוים.
- printAllSpecializations() - פעולה להצגת כל תחומי ההתמחות האפשריים של רופאים במערכת.
- isDoctorsEqual(const Doctor* d1, const Doctor* d2) - פעולה הבודקת אם שני רופאים זהים על פי תכונותיהם.
- isDoctorExist(const Doctor* doctors, int doctorsNum, const char* licenseId) - פעולה לבדיקה האם רופא עם רישיון מסוים קיים במערכת.
- printDoctorsBasicInfo(Doctor* doctors, int size) - פעולה להצגת מידע בסיסי על כל הרופאים במערך.
- getSpecialization() - פעולה לקבלת תחום ההתמחות.
- *getDoctorIdBylicenceId(char* licenceId, const Doctor* doctors, int size)* - פעולה למציאת מזהה הרופא על פי מזהה רישיון.
- printDoctorsBySpecialization(const Doctor* doctors, int size, eSpecialization spec) - פעולה להצגת רשימת רופאים לפי תחום התמחות מסוים.

- findDoctorById(const Doctor* doctors, int numDoctors, int id) - פעולה למציאת רופא על פי מזהה ייחודי.
- freeDoctor(Doctor* doc) - פעולה לשחרור זיכרון שהוקצה למבנה Doctor .

eSpecialization - מייצג את תחומי ההתמחות השונים של הרופאים במערכת. כולל ערכים כגון: רפואת עיניים, קרדיולוגיה, נוירולוגיה, אונקולוגיה, כירורגיה, פסיכיאטריה, דרמטולוגיה, אורטופדיה, גינקולוגיה, אורולוגיה, גסטרואנטרולוגיה, אנדוקרינולוגיה, ועוד.

מבנה Appointment:

המבנה Appointment משמש לייצוג פגישה בין רופא למטופל במערכת ניהול קופת חולים. הוא מכיל תכונות שמייצגות את פרטי הפגישה, כולל מזהי רופא ומטופל, תאריך ושעה. המבנה כולל פעולות לניהול ותפעול נתוני הפגישות במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- מזהה רופא (int doctor_id) - מספר שלם המייצג את מזהה הרופא שקבע את הפגישה.
- מזהה מטופל (int patient_id) - מספר שלם המייצג את מזהה המטופל עבורו נקבעה הפגישה.
- תאריך הפגישה (Date date) - מבנה מסוג Date המייצג את תאריך הפגישה.
- שעת הפגישה (Time time) - מבנה מסוג Time המייצג את שעת הפגישה.

הפעולות הקיימות במחלקה:

- *initAppointment(Appointment* appointment, const Doctor* doctors, const Patient** patients, int numPatients, int numDoctors)* - פעולה לאתחול מבנה Appointment , כולל קבלת מידע מהרופא ומהמטופל ואיפוס ערכים.
- *printAppointment(const Appointment* appointment, const Doctor* doctors, const Patient** patients, int numPatients, int numDoctors)* - פעולה להצגת כל המידע הקיים על פגישה מסוימת, כולל שמות הרופא והמטופל, תאריך ושעה.
- *areAppointmentsConflicting(const Appointment* a1, const Appointment* a2)* - פעולה לבדיקה האם שתי פגישות מתנגשות על פי תאריך ושעה, בהתחשב בחלון זמן מוגדר להתנגשות.
- addAppointment(NODE** list, Appointment* newAppointment) - פעולה להוספת פגישה חדשה לרשימה מקושרת של פגישות.
- removeAppointment(NODE** list, int index) - פעולה להסרת פגישה מהרשימה המקושרת על פי אינדקס נתון.
- printAppointments(NODE* list) - פעולה להצגת כל הפגישות ברשימה המקושרת.
- freeAppointmentList(NODE** list) - פעולה לשחרור כל המשאבים המוקצים לרשימת הפגישות המקושרת.

מבנה Date :

המבנה Date משמש לייצוג תאריך במערכת ניהול קופת חולים. המבנה מכיל תכונות שמייצגות את היום, החודש והשנה, ומשתמש בפעולות שונות לניהול ואימות נתוני תאריך.

המבנה מכיל בתוכו את התכונות הבאות :

- יום (int day) - מספר שלם המייצג את היום בחודש (1-31, בהתאם לחודש ולשנה).
- חודש (int month) - מספר שלם המייצג את החודש בשנה (1-12).
- שנה (int year) - מספר שלם המייצג את השנה.

הפעולות הקיימות במחלקה:

- *areDatesEqual(const Date* date1, const Date* date2)* - פעולה הבודקת האם שני תאריכים שווים.
- isValidDate(Date date) - פעולה הבודקת אם תאריך נתון תקין (למשל, האם התאריך קיים בפועל, כולל התחשבות בשנים מעוברות ובמספר הימים בחודש).
- printDate(const Date* date) - פעולה המציגה את התאריך בפורמט יום/חודש/שנה (DD/MM/YYYY).
- isLeapYear(int year) - פעולה הבודקת אם שנה מסוימת היא שנה מעוברת.
- daysInMonth(int month, int year) - פעולה המחשבת ומחזירה את מספר הימים בחודש נתון בשנה נתונה.
- initDate(Date* date) - פעולה לאתחול מבנה Date כולל קבלת נתונים מהמשתמש או מקור אחר והגדרת הערכים ליום, חודש ושנה.
- *isDateFormat(char* str, int* year, int* month, int* day)* - פעולה הבודקת אם מחרוזת נתונה תואמת את פורמט התאריך וממירה אותה לערכים נפרדים של יום, חודש ושנה.
- *compareDate(const Date* a, const Date* b)* - פעולה המשווה בין שני תאריכים ומחזירה ערך שלילי אם התאריך הראשון קודם לשני, 0 אם הם שווים, וערך חיובי אם התאריך הראשון מאוחר יותר.

מבנה Drug:

המבנה Drug משמש לייצוג תרופה במערכת ניהול קופת חולים. המבנה מכיל תכונות שמייצגות את שם התרופה, תאריך התפוגה שלה, ומזהה סריאלי ייחודי, וכולל פעולות לניהול, מיון וחיפוש של תרופות במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- שם התרופה (char* name) - מחרוזת המייצגת את שם התרופה.
- תאריך תפוגה (Date exp_date) - מבנה מסוג Date המייצג את תאריך התפוגה של התרופה.
- מזהה התרופה (char serial_id[S_ID_MAX]) - מחרוזת באורך קבוע המייצגת את המזהה הייחודי של התרופה.

DrugSortType משמש לייצוג סוגי מיון שונים עבור תרופות. כולל ערכים:

- **eName**: מיון לפי שם התרופה.
- **eSerialId**: מיון לפי מזהה סריאלי.
- **eExpDate**: מיון לפי תאריך תפוגה.
- **eNothing**: ללא מיון.
- **NUM_OF_SORTS**: מספר סוגי המיון האפשריים.

הפעולות הקיימות במחלקה:

- initDrug(Drug** drug) - פעולה לאתחול מבנה Drug כולל הקצאת זיכרון והגדרת ערכים ראשוניים עבור שם, תאריך תפוגה ומזהה סריאלי.
- printDrug(const Drug* drug) - פעולה להצגת כל המידע הקיים על תרופה מסוימת, כולל שם, תאריך תפוגה ומזהה סריאלי.
- printDrugs(const Drug** drugs, int size) - פעולה להצגת כל התרופות במערך נתון של תרופות.
- searchDrugBySerialId(const Drug** drugs, int numOfDrugs, char* serial_id) - פעולה לחיפוש תרופה במערך על פי מזהה סריאלי ומחזירה מצביע לתרופה אם נמצאה, אחרת מחזירה NULL.
- addDrugToArray(Drug*** drugs, Drug* newDrug, int size) - פעולה להוספת תרופה חדשה למערך תרופות קיים והרחבת המערך בהתאם.
- freeDrug(Drug** drug) - פעולה לשחרור זיכרון שהוקצה למבנה Drug כולל שחרור זיכרון לשם התרופה.
- *compareByName(const void* a, const void* b)* - פעולה להשוואה בין שתי תרופות לפי שם לצורך מיון.
- *compareBySerialId(const void* a, const void* b)* - פעולה להשוואה בין שתי תרופות לפי מזהה סריאלי לצורך מיון.
- *compareByExpDate(const void* a, const void* b)* - פעולה להשוואה בין שתי תרופות לפי תאריך תפוגה לצורך מיון.
- printSortTypes() - פעולה להצגת סוגי המיון הזמינים עבור תרופות.

<u>מבנה HMO_Admin :</u>

המבנה HMO_Admin משמש לייצוג מנהל קופת חולים. הוא מכיל תכונות שמייצגות את שם המנהל, מזהה ייחודי ושנות ותק.

המבנה מכיל בתוכו את התכונות הבאות :

- שם המנהל (char* name) - מחרוזת המייצגת את שמו של המנהל.
- מזהה מנהל (int id) - מספר שלם המייצג מזהה ייחודי למנהל במערכת.

- שנות ותק (int seniority_years) - מספר שלם המייצג את שנות הוותק של המנהל בתפקידו.
- מין הרופא (char gender) - תו המייצג את מין הרופא ('M' לזכר 'F' לנקבה).

הפעולות הקיימות במחלקה:

- initHMOAdmin(HMO_Admin** admin) - פעולה לאתחול מבנה HMO_Admin כולל הקצאת זיכרון והגדרת ערכים ראשוניים עבור שם, מזהה ושנות ותק.
- printHMOAdmin(const HMO_Admin* admin) - פעולה להצגת כל המידע הקיים על מנהל מסוים, כולל שם, מזהה ושנות ותק.
- freeHMOAdmin(HMO_Admin* admin) - פעולה לשחרור זיכרון שהוקצה למבנה HMO_Admin כולל שחרור זיכרון לשם המנהל.

<u>מבנה Patient:</u>

המבנה Patient משמש לייצוג מטופל בקופת חולים. המבנה מכיל תכונות שמייצגות את פרטי המטופל, כולל מזהה ייחודי, שם, תאריך לידה ומגדר. בנוסף, המבנה כולל פעולות לניהול ותפעול נתוני המטופלים במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- מזהה מטופל (int id) - מספר שלם המייצג מזהה ייחודי למטופל במערכת.
- שם המטופל (char* name) - מחרוזת המייצגת את שם המטופל.
- תאריך לידה (Date birthDate) - מבנה מסוג Date המייצג את תאריך הלידה של המטופל.
- **מגדר** (char gender) - תו המייצג את מגדר המטופל (M עבור זכר F עבור נקבה).

הפעולות הקיימות במחלקה:

- initPatient(Patient** patient) - פעולה לאתחול מבנה Patient כולל הקצאת זיכרון והגדרת ערכים ראשוניים עבור מזהה, שם, תאריך לידה ומגדר.
- printPatient(const Patient* patient) - פעולה להצגת כל המידע הקיים על מטופל מסוים, כולל מזהה, שם, תאריך לידה ומגדר.
- printPatients(const Patient** patient,int size) - פעולה להצגת המידע של כל המטופלים במערך נתון של מטופלים.
- *isPatientsEqual(const Patient* p1, const Patient* p2)* - פעולה הבודקת אם שני מטופלים זהים לפי התכונות שלהם.
- isPatientExist(const Patient** patients, int patientsNum, int id) - פעולה הבודקת אם מטופל עם מזהה נתון קיים במערך המטופלים.
- findPatientById(const Patient** patients, int numPatients, int id) - פעולה למציאת מטופל במערך על פי מזהה ייחודי ומחזירה מצביע למטופל אם נמצא, אחרת מחזירה NULL.

- addPatientToArray(Patient*** patients, Patient* newP, int size) - פעולה להוספת מטופל חדש למערך מטופלים קיים והרחבת המערך בהתאם.
- freePatient(Patient** patient) - פעולה לשחרור זיכרון שהוקצה למבנה Patient כולל שחרור זיכרון לשם המטופל.

מבנה Pharmacist :

המבנה Pharmacist משמש לייצוג רוקח במערכת ניהול קופת חולים. המבנה מכיל תכונות שמייצגות את פרטי הרוקח, כולל שם, מזהה ייחודי, מספר רישיון, והרשאה למכירת תרופות עם מרשם. בנוסף, המבנה כולל פעולות לניהול ותפעול נתוני הרוקחים במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- שם הרוקח (char* name) - מחרוזת המייצגת את שמו של הרוקח.
- מזהה רוקח (int id) - מספר שלם המייצג מזהה ייחודי לרוקח במערכת.
- מספר רישיון (char licence_id[L_ID_MAX]) - מחרוזת באורך קבוע המייצגת את מספר הרישיון של הרוקח.
- האם מותר למכור תרופות מרשם (int isPermittedToSellPerscriptedDrugs) - מספר שלם המייצג האם הרוקח מורשה למכור תרופות מרשם (1 אם מורשה, 0 אם לא).

הפעולות הקיימות במחלקה:

- initPharmacist(Pharmacist** pharmacist) - פעולה לאתחול מבנה Pharmacist כולל הקצאת זיכרון והגדרת ערכים ראשוניים עבור שם, מזהה, מספר רישיון והרשאה למכירת תרופות מרשם.
- printPharmacist(const Pharmacist* pharmacist) - פעולה להצגת כל המידע הקיים על רוקח מסוים, כולל שם, מזהה, מספר רישיון והרשאה למכירת תרופות מרשם.
- *isPharmacistsEqual(const Pharmacist* p1, const Pharmacist* p2)* - פעולה הבודקת אם שני רוקחים זהים לפי התכונות שלהם. מחזירה 1 אם הרוקחים שווים, אחרת 0.
- addPharmacistToArray(Pharmacist*** pharmacists, Pharmacist* newP, int size) - פעולה להוספת רוקח חדש למערך רוקחים קיים והרחבת המערך בהתאם.
- freePharmacist(Pharmacist** ph) - פעולה לשחרור זיכרון שהוקצה למבנה Pharmacist כולל שחרור זיכרון לשם הרוקח.
- printPharmacists(const Pharmacist** pharmacists,int size) - פעולה להצגת כל הרוקחים במערך נתון של רוקחים.

מבנה Pharmacy :

המבנה Pharmacy משמש לייצוג בית מרקחת במערכת ניהול קופת חולים. המבנה מכיל תכונות שמייצגות את פרטי בית המרקחת, כולל כתובת, מזהה של הרוקח האחראי, רשימת תרופות, רשימת

רוקחים, וסוג המיון הנוכחי של התרופות. בנוסף, המבנה כולל פעולות לניהול ותפעול נתוני בתי המרקחת במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- כתובת (char* address) - מחרוזת המייצגת את כתובת בית המרקחת.
- מזהה רוקח אחראי (int responsible_pharmacist_id) - מספר שלם המייצג את המזהה של הרוקח האחראי על בית המרקחת.
- רשימת תרופות (Drug** drugs) - מערך מצביעים של תרופות המייצג את התרופות הזמינות בבית המרקחת.
- מספר התרופות (int numOfDrugs) - מספר שלם המייצג את מספר התרופות הקיימות במלאי של בית המרקחת.
- מספר רוקחים (int numOfPharmacists) - מספר שלם המייצג את מספר הרוקחים העובדים בבית המרקחת.
- רשימת רוקחים (Pharmacist** pharmacists) - מערך מצביעים של רוקחים המייצג את הרוקחים העובדים בבית המרקחת.
- סוג מיון תרופות (DrugSortType eSort) - משתנה מסוג DrugSortType המייצג את סוג המיון הנוכחי של התרופות במלאי.

הפעולות הקיימות במחלקה:

- initPharmacy(Pharmacy** pharmacy) - פעולה לאתחול מבנה Pharmacy כולל הקצאת זיכרון והגדרת ערכים ראשוניים עבור הכתובת, רשימת התרופות, רשימת הרוקחים, ומאפיינים נוספים.
- addPharmacistToPharmacy(Pharmacy** pharmacy, Pharmacist* newPharmacist) - פעולה להוספת רוקח חדש לרשימת הרוקחים של בית המרקחת.
- addDrugToPharmacy(Pharmacy** pharmacy, Drug* newDrug) - פעולה להוספת תרופה חדשה לרשימת התרופות של בית המרקחת.
- printPharmacy(const Pharmacy* pharmacy) - פעולה להצגת כל המידע הקיים על בית מרקחת מסוים, כולל כתובת, רוקח אחראי, רשימת תרופות ורוקחים.
- sortDrugs(Pharmacy** pharmacy, DrugSortType sortType) - פעולה למיון התרופות במלאי של בית המרקחת לפי סוג מיון נתון.(DrugSortType)
- printPharmacyDrugs(const Pharmacy* pharmacy) - פעולה להצגת כל התרופות במלאי של בית המרקחת.
- *searchDrug(const Pharmacy* pharmacy, const void* key)* - פעולה לחיפוש תרופה במלאי בית המרקחת לפי מפתח נתון (כגון שם, מזהה סריאלי וכו'). מחזירה מצביע לתרופה אם נמצאה, אחרת NULL.
- searchAndPrintDrug(Pharmacy* pharmacy) - פעולה לחיפוש תרופה במלאי בית המרקחת ולהצגת המידע אודותיה אם נמצאה.

<u>מבנה Prescription:</u>

המבנה Prescription משמש לייצוג מרשם רפואי במערכת ניהול קופת חולים. המבנה מכיל תכונות שמייצגות את פרטי המרשם, כולל מזהה תרופה, מינון, תאריך תפוגה של המרשם, מזהי מטופל, רופא ורוקח. בנוסף, המבנה כולל פעולות לניהול ותפעול נתוני המרשמים במערכת.

המבנה מכיל בתוכו את התכונות הבאות :

- מזהה תרופה (char drug_serial_id[L_ID_MAX]) - מחרוזת באורך קבוע המייצגת את המזהה הסריאלי של התרופה הנמצאת במרשם.
- מינון (int dose) - מספר שלם המייצג את המינון של התרופה כפי שנרשם במרשם.
- תאריך תפוגת המרשם (Date prescription_exp) - תאריך המייצג את תאריך התפוגה של המרשם.
- מזהה מטופל (int patient_id) - מספר שלם המייצג את המזהה הייחודי של המטופל עבורו ניתן המרשם.
- מזהה רופא (int doctor_id) - מספר שלם המייצג את המזהה הייחודי של הרופא שהנפיק את המרשם.
- מזהה רוקח (int pharmacist_id) - מספר שלם המייצג את המזהה הייחודי של הרוקח שטיפל במרשם.

הפעולות הקיימות במחלקה:

- isPrescriptionUsed(const Prescription* p) - פעולה הבודקת אם המרשם כבר נעשה בו שימוש. מחזירה 1 אם המרשם בשימוש, אחרת 0.

<u>מבנה Time:</u>

המבנה Time משמש לייצוג זמן (שעה ודקה) במערכת ניהול קופת חולים. המבנה מכיל תכונות שמייצגות את השעה והדקה של זמן מסוים.

המבנה מכיל בתוכו את התכונות הבאות :

- שעה (int hour) - מספר שלם המייצג את השעה.(0-23)
- דקה (int minute) - מספר שלם המייצג את הדקה.(0-59)

הפעולות הקיימות במחלקה:

- initTime(Time* time) - פעולה לאתחול מבנה Time כולל הגדרת ערכים ראשוניים עבור שעה ודקה.
- checkIfTimeValid(Time time) - פעולה שבודקת האם ערכי הזמן (שעה ודקה) תקינים (השעה בטווח 0-23 והדקה בטווח 0-59). מחזירה 1 אם הזמן תקין, אחרת 0.
- printTime(Time time) - פעולה להצגת זמן נתון.
- isTimesEqual(const Time t1, const Time t2) - פעולה הבודקת אם שני זמנים זהים לפי התכונות שלהם (שעה ודקה). מחזירה 1 אם הזמנים שווים, אחרת 0.

דחיסת מבנה:

בחרנו לדחוס את מבנה HMO_Admin באמצעות אלגוריתם RLE .
את הדחיסה ביצענו במספר שלבים:

1. ראשית את המבנה HMO_Admin נסדר לתוך מאגר בתים רציף בסדר הבא:

   אורך השם (Int)

   שם (char array)

   מזהה (Int)

   שנות ותק (Int)

   מגדר (char)

2. נדחוס באמצעות RLE (Run-length encoding)

   הנתונים המסודרים נדחסים לאחר מכן באמצעות האלגוריתם הבא:

   נחזור על כל בייט במאגר המסודר.

   עבור כל בייט, נספור בתים זהים עוקבים (עד 255).

   נכתוב שני בתים למאגר הדחוס:

   ספירת בתים זהים עוקבים (1 Byte)

   ערך הבתים עצמו (1 Byte)

3. כתיבה לקובץ:
   נכתוב את גודל הנתונים הדחוסים (Int)
   נכתוב את הנתונים הדחוסים לתוך מערך (Byte array).

דוגמא לדחיסה:

מקור: AAABBBCCCCDDDD

דחוס: 3A 3B 4C 4D

פעולות מרכזיות במערכת:

על כל הפעולות הקיימות במערכת פירטנו בחלק של תיאור המבנים של הפרוייקט, להלן רשימה של הפעולות המרכזיות אותם המשתמש יכול לבצע במערכת:

הדפסת קופת החולים :מציגה את כל המידע הקיים על קופת החולים, כולל רופאים, חולים, בתי מרקחת, מרשמים ופגישות.

הדפסת מידע על בית המרקחת :מציגה את כל המידע על בית המרקחת הקיים בקופת החולים, כולל תרופות ורוקחים.

הוספת רופא חדש :מאפשרת למשתמש להוסיף רופא חדש למערכת על ידי הזנת כל הפרטים הנדרשים.

הוספת מטופל חדש :מאפשרת למשתמש להוסיף מטופל חדש למערכת על ידי הזנת כל הפרטים הנדרשים.

הוספת תרופה חדשה לבית המרקחת :מאפשרת למשתמש להוסיף תרופה חדשה לרשימת התרופות של בית המרקחת במערכת.

תזמון פגישה חדשה :מאפשרת למשתמש ליצור פגישה חדשה בין מטופל לרופא במערכת.

הוספת רוקח חדש :מאפשרת למשתמש להוסיף רוקח חדש לבית המרקחת.

הדפסת כל הרופאים :מציגה את כל הרופאים הרשומים במערכת.

הדפסת כל המטופלים :מציגה את כל המטופלים הרשומים במערכת.

הדפסת כל הרוקחים :מציגה את כל הרוקחים הרשומים במערכת.

הדפסת כל התרופות :מציגה את כל התרופות הרשומות בבית המרקחת של המערכת.

הדפסת כל הפגישות :מציגה את כל הפגישות שתוזמנו במערכת.

הדפסת כל הפגישות של מטופל מסוים :מציגה את כל הפגישות שתוזמנו עבור מטופל מסוים על פי תעודת זהות.

מיון תרופות :מאפשרת למשתמש למיין את רשימת התרופות בבית המרקחת לפי קטגוריות שונות (שם, מספר סידורי, תאריך תפוגה).

חיפוש תרופה :מאפשרת למשתמש לחפש תרופה מסוימת בבית המרקחת ולהציג את פרטיה.

יצירת מרשם חדש :מאפשרת למשתמש ליצור מרשם חדש לרופא, מטופל ותרופה מסוימת.

הדפסת כל המרשמים :מציגה את כל המרשמים שנוצרו במערכת.

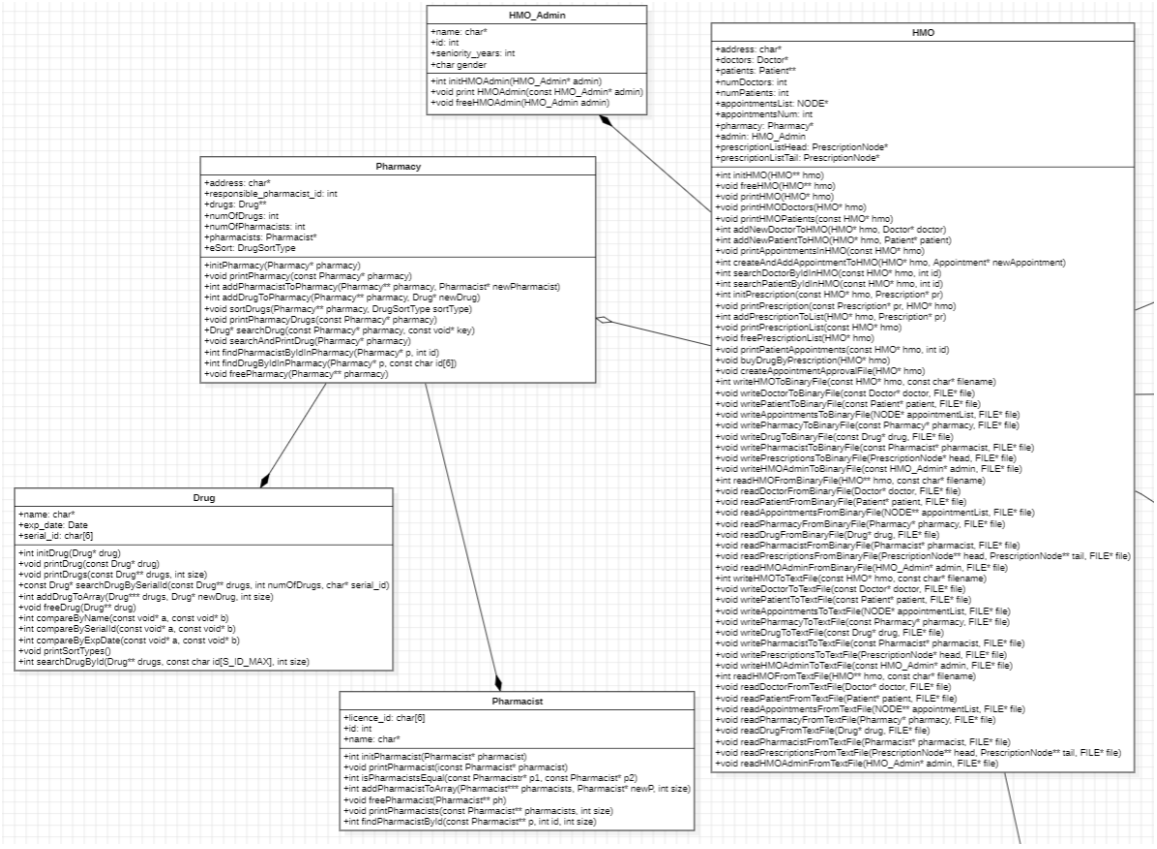
קניית תרופה לפי מרשם :מאפשרת למשתמש לקנות תרופה עבור מטופל על פי מרשם שניתן מראש.

יצירת קובץ אישור לפגישה :יוצרת קובץ אישור לפגישה עבור המטופל כדי לאשר את המועד והזמן.

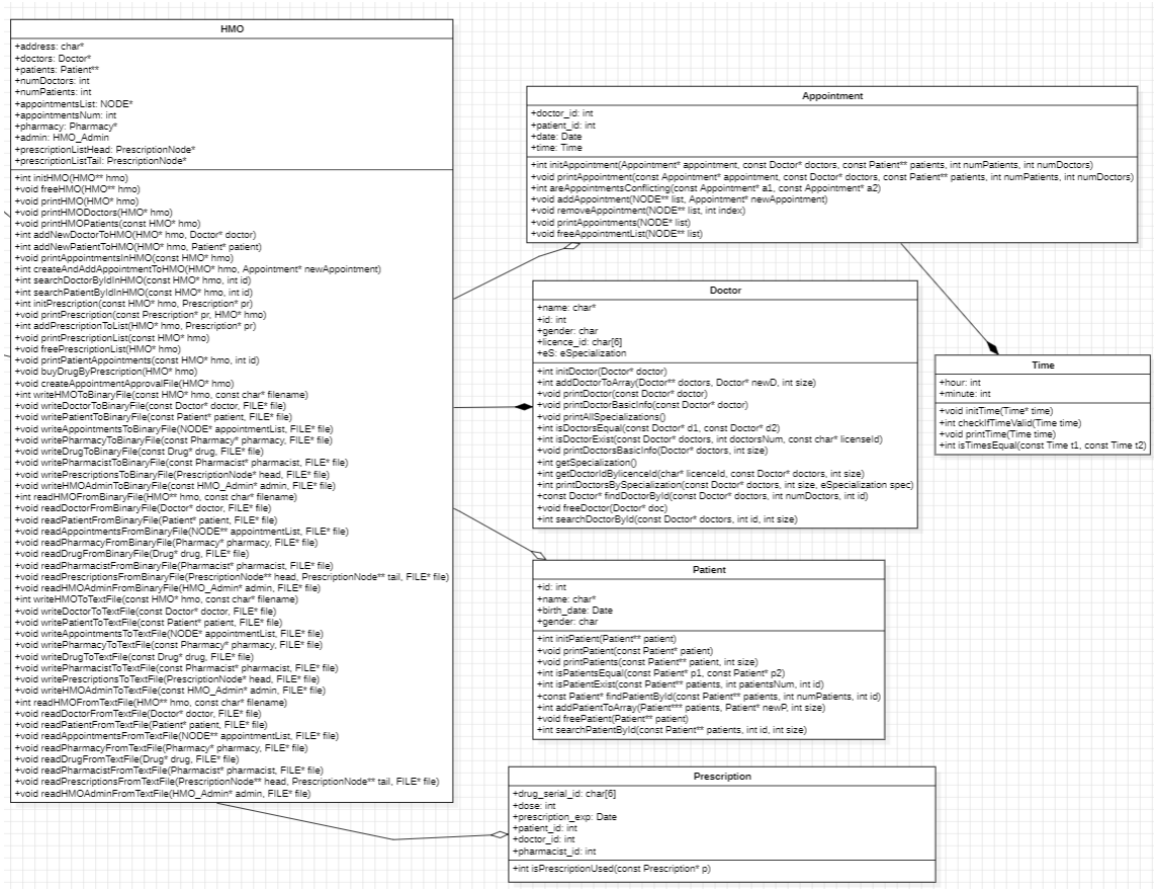
שרטוט המערכת:

לאחר תחילת העבודה על הפרוייקט הבנו שנצטרך לבצע שינויים בשרטוט.
לא הצלחנו לצלם את הדיאגרמה בצורה שתהיה קריאה לכן נצרף אותה בשתי תמונות ובנוסף נצרף את הקובץ לzip.

חלק שמאלי של השרטוט:



חלק ימני של הסרטוט



חלוקת אחריות:

בחרנו לחלק את העבודה באופן שבו שנינו נוכל לגעת בכל הנושאים, חילקנו בינינו את המבנים ואת הפעולות המיוחדות בנינו יחד. חשוב לציין שעזרנו אחד לשני לתכנן ולבנות החלקים אחד בהתאם לשני, ובנוסף פתרנו תקלות יחד ככה שחלוקת העבודה הייתה שווה.

תחום	אחראי
Appointment	נתנאל
Date	נתנאל
Doctor	נתנאל
Drug	ירדן
HMO	נתנאל + ירדן
HMO_Admin	ירדן
Patient	ירדן
Pharmacy	ירדן
Pharmacist	ירדן
Prescription	נתנאל
Time	ירדן