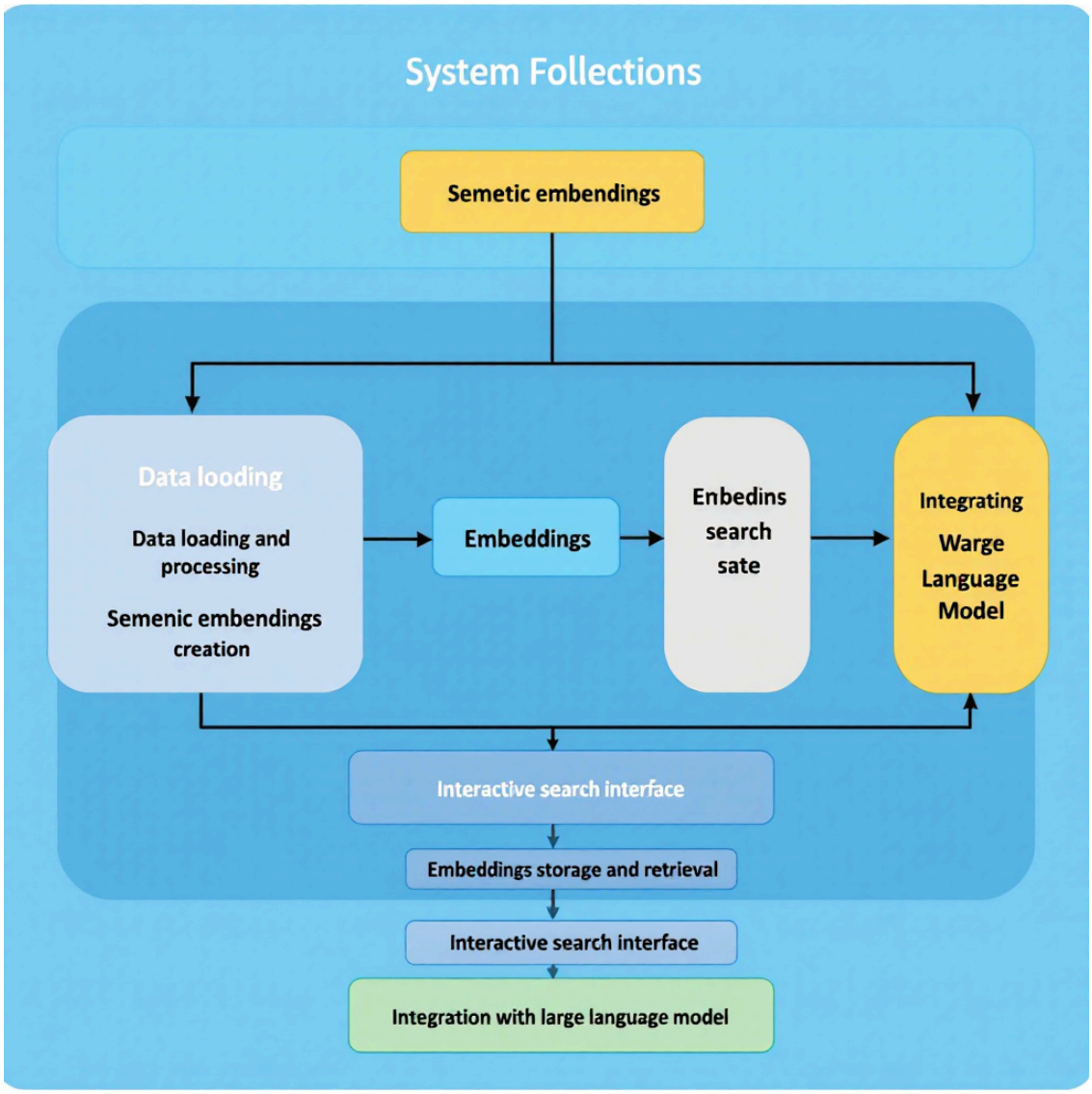
להלן התשובות שלי לשאלות בהתייחס לקוד שכתבתי בחלק א':

1. בחרתי בארכיטקטורה המשלבת מספר רכיבים עיקריים:
טעינה ועיבוד מקדים של נתוני ספר השירות מקובץ האקסל באמצעות ספריית Pandas
יצירת ייצוגי embedding סמנטיים למידע על הרופאים באמצעות מודל שפה רב-לשוני (distiluse-base-multilingual-cased-v2)
שמירה וטעינה של ייצוגי ה-embedding לצורך יעילות
ממשק חיפוש אינטראקטיבי המאפשר שאילתות בשפה טבעית
שילוב עם מודל שפה גדול (LLM) ליצירת תשובות בשפה אנושית על בסיס תוצאות החיפוש , להלן תרשים המתאר את הרכיבים העיקריים:



2. הנחות עבודה והחלטות:
הנחתי שהנתונים בקובץ האקסל מייצגים את כל המידע הרלוונטי הנדרש לחיפוש ספקי שירות
בחרתי להשתמש במודל embedding רב-לשוני כדי לתמוך בשאילתות בעברית
החלטתי לשמור את ייצוגי ה-embedding בקובץ כדי לחסוך זמן חישוב בהרצות חוזרות
בחרתי להשתמש במודל LLM קיים (Qwen2.5) במקום לאמן מודל ייעודי, כדי לחסוך זמן ומשאבים בשלב האב-טיפוס

3. הערכת ביצועים:
ניתן להעריך את איכות תוצאות החיפוש על ידי בדיקה ידנית של הרלוונטיות שלהן לשאילתות מגוונות
אפשר למדוד את זמן התגובה של המערכת לשאילתות כדי להעריך את היעילות
שימוש במדדים כמו precision ו-recall יכול לתת אינדיקציה כמותית לאיכות תוצאות החיפוש, בהינתן סט נתונים מתויג

4. תרחישי הצלחה וכישלון:
הפתרון צפוי להצליח כאשר השאילתות מנוסחות בצורה ברורה וכוללות מונחים וביטויים שמופיעים בנתוני ספר השירות
המערכת עלולה להיכשל כאשר השאילתות מכילות שגיאות כתיב, ראשי תיבות או מונחים שלא מופיעים בנתונים
תוצאות לא רלוונטיות עלולות להתקבל אם המידע בספר השירות אינו מלא או מעודכן

5. שיפורים אפשריים:
הוספת יכולת לחיפוש גיאוגרפי על סמך מיקום המשתמש
שיפור מנגנון ההבנה של שאילתות על ידי טיפול בשגיאות כתיב ו נרדפות
הרחבת בסיס הנתונים כך שיכלול מידע נוסף כמו שעות פעילות, התמחויות נוספות וכד'
פיתוח ממשק משתמש גרפי נוח יותר לאינטראקציה עם המערכת
שיפור מהירות החיפוש על ידי אופטימיזציה של האלגוריתמים וניצול טוב יותר של משאבי חומרה

6. שלבים להטמעה ואחזקה בסביבת ייצור
אוטומציה של תהליך הבנייה והפריסה: שימוש בכלי CI/CD כדי להבטיח עדכונים רציפים.
פריסת המערכת בשרתים ייעודיים: או שימוש בשירותי ענן כמו AWS, Azure או Google Cloud.
ניהול בסיס נתונים: העברת הנתונים מבסיס קובץ Excel לבסיס נתונים מקצועי (כמו Pinecone, Weaviate Faiss Qdrant Milvus pgvector Chroma).
אבטחת מידע: הטמעת פרוטוקולי אבטחה להגנה על הנתונים והמשתמשים.

ניטור ותקלות: שימוש בכלי ניטור למעקב אחרי ביצועי המערכת וזיהוי תקלות בזמן אמת.
תחזוקה ועדכונים: קביעת לוח זמנים לעדכוני מערכת, מודלים ונתונים.
תמיכת משתמשים: הקמת מערכת תמיכה לטיפול בפניות ושאלות המשתמשים.
דוקומנטציה: כתיבת תיעוד מסודר של הקוד, הארכיטקטורה ותהליכי העבודה לטובת מפתחים וצוותים נוספים.

מסדי נתונים וקטוריים מתמחים באחסון יעיל וחיפוש של אמבדינגים וקטוריים בעלי ממדים גבוהים. הם מספקים אופטימיזציות ספציפיות בהשוואה למסדי נתונים מסורתיים או אינדקסים וקטוריים עצמאיים אחסון האמבדינגים במסד נתונים וקטורי יכול לאפשר חיפושי דמיון מהירים כדי למצוא מידע רלוונטי בנתוני האקסל.

אמבדינגים גרפיים המאוחסנים במסדי נתונים וקטוריים יכולים למדל ביעילות מבנים מורכבים של יחסים, שעשויים להיות קיימים בחלק ממערכי הנתונים של אקסל. ממדי הווקטור לוכדים את האופן שבו ישויות שונות מקושרות זו לזו.

בעוד שאמבדינגים בודדים הם קטנים יחסית (קילובייטים), דרישות האחסון יכולות להצטבר במהירות כאשר מטפלים באלפי אמבדינגים. מסדי נתונים וקטוריים מתוכננים לטפל ביעילות בתרחיש זה ולהתרחב היטב.

ביישום RAG טיפוסי, משתמשים במודלים של אמבדינג לצורך קליטת נתונים והבנת הנחיות המשתמש האמבדינגים המתקבלים מאוחסנים לאחר מכן במסד הנתונים הווקטורי יחד עם מטא-נתונים לצורך אחזור מאוחר יותר במהלך תהליך ה-RAG.

חשוב לבחור מודל אמבדינג וקטורי המתאים היטב לסוגי הנתונים הספציפיים ולמקרה השימוש מודל האמבדינג אחראי להפיכת נתוני האקסל הלא מובנים לווקטורים בעלי משמעות סמנטית.

**אלגוריתם לחיפוש רופא באמצעות אמבדינג וסינון תוצאות**

בהינתן שחיפוש האמבדינג מחזיר תוצאות של רופאים במומחיות הרצויה גם מחוץ לעיר המבוקשת, ניתן לנצל זאת כדי לבנות אלגוריתם יעיל יותר. להלן תיאור של אלגוריתם כזה:

1. קלט:
   - עיר מבוקשת
   - מומחיות מבוקשת
   - מספר התוצאות המקסימלי הרצוי
2. חיפוש אמבדינג:
   - השתמש במודל אמבדינג כדי לחפש רופאים לפי המומחיות המבוקשת
   - המודל יחזיר רופאים רלוונטיים גם מחוץ לעיר המבוקשת
3. סינון תוצאות לפי עיר:
   - סנן את תוצאות חיפוש האמבדינג כך שיכללו רק רופאים מהעיר המבוקשת
   - אם מספר התוצאות המסוננות עומד במספר התוצאות המקסימלי הרצוי, החזר את התוצאות וסיים את האלגוריתם
4. הרחבת תוצאות לערים סמוכות:
   - אם מספר התוצאות המסוננות נמוך ממספר התוצאות המקסימלי הרצוי, הרחב את הסינון כך שיכלול גם רופאים מערים סמוכות
   - מיין את הערים הסמוכות לפי מרחק מהעיר המבוקשת
   - הוסף בהדרגה רופאים מהערים הסמוכות לתוצאות, עד להשגת מספר התוצאות המקסימלי הרצוי
5. החזרת תוצאות מסוננות:
   - החזר את רשימת הרופאים המסוננת, הכוללת רופאים מהעיר המבוקשת וערים סמוכות במידת הצורך
   - אם לא נמצאו מספיק רופאים גם לאחר הרחבת החיפוש לערים סמוכות, החזר את כל התוצאות שנמצאו

האלגוריתם מנצל את היכולת של מודל האמבדינג למצוא רופאים רלוונטיים במומחיות הנדרשת, גם מחוץ לעיר המבוקשת. לאחר מכן, הוא מסנן את התוצאות כדי לתעדף רופאים מהעיר המבוקשת. אם אין מספיק תוצאות, הוא מרחיב בהדרגה את החיפוש לערים סמוכות, עד לקבלת מספר התוצאות הרצוי.

גישה זו מאפשרת לנצל את העושר של תוצאות האמבדינג תוך מתן עדיפות לרופאים מהעיר המבוקשת והסביבה הקרובה. זה יכול לספק למשתמשים המלצות רלוונטיות יותר, גם כאשר אין התאמות מושלמות בעיר עצמה.