



לימודי חינוך והכשרה פנימית
בראשות מנכ"ל ראשי המרכז הרבני שליד
המוזיאון תל אביב

סמינר החדש פרי תואר אלעד.
שם : רבקה מזל סימן טוב.
ת.ז. : 213286776
שם המנחה : גב' רחל דרבקין
תאריך ההגשה : 16/06/2022

תוכן

4	הצעת פרוייקט
7	מבוא
7	הרקע לפרויקט
7	תהליך המחקר
9	סקירת ספרות
10	אתגרים מרכזיים
10	הבעיה איתה התמודד התלמיד
10	הסיבות לבחירת הנושא
10	מטרות ויעדים
11	מדדי הצלחה למערכת
11	רקע תיאורטי / ספרות מקצועית
11	תיאור מצב קיים
11	ניתוח חלופות מערכת
12	תיאור החלופה הנבחרת
13	אפיון המערכת שהוגדרה
13	ניתוח דרישות המערכת
13	מודול המערכת
14	אפיון פונקציונלי
14	ביצועים עיקריים
14	אילוצים
15	תיאור הארכיטקטורה
15	הארכיטקטורה של הפתרון המוצע בפורמט של Design level Down-Top
16	תיאור הרכיבים בפתרון
16	ארכיטקטורת רשת
16	תיאור פרוטוקולי התקשורת
16	שרת-לקוח
16	תיאור הצפנות
17	ניתוח ותרשים Use / UML cases של המערכת המוצעת
17	רשימת ה- UC העיקריים של המערכת
17	תיאור ה- UC העיקריים של המערכת
18	מבנה נתונים בשימוש בפרויקט
19	הקשרים בין היחידות השונות
19	עץ מודולים
19	Use case Diagram

20	Design class Diagram
21	תיאור המחלקות המוצעות
27	רכיבי ממשק
27	תיכון המערכת
27	ארכיטקטורת המערכת
27	תיכון מפורט
27	חלופות לתיכון המערכת
28	תיאור התוכנה
28	תיאור מסכים
	תרשים מסכים המתאר את היררכיית המסכים והמעברים ביניהם (diagram flow Screen)
29	
29	תפקידו של כל מסך
36	קוד התוכנית - האלגוריתם המרכזי
40	תיאור מסד הנתונים
44	מדריך למשתמש
46	בדיקות והערכה
46	ניתוח יעילות
46	אבטחת מידע
46	מסקנות וסיכום
46	פיתוחים עתידיים
47	ביבליוגרפיה

הצעת פרוייקט

סמל מוסד : 716605

שם מכללה : סמינר החדש פרי תואר אלעד

שם הסטודנט : רבקה מזל סימן-טוב.

ת.ז הסטודנט : 213286776

שם הפרויקט : Driver PopUp.

תיאור הפרויקט :

מערכת לניהול שירותי תחבורה שיתופית הפועלת לתכנן מסלול נסיעה בצורה חכמה, כך שיהיה זה המסלול הקצר ביותר החוסך זמן לנוסעים ומקום בכבישי ישראל גם יחד.

כל משתמש יכול להזמין נסיעה לעצמו, או להציע נסיעה למשתמשים אחרים.

הזמנת הנסיעה מתבצעת על ידי ציון נקודת איסוף ונקודת יעד, המערכת מוצאת את הרכב הזמין הקרוב ביותר שהמסלול שלו יכול להשתלב היטב עם האילוצים החדשים שיתווספו לו, ללא שינוי ניכר שעשוי להפוך את המסלול ללא יעיל.

לאחר מכן, המערכת משנה את תוואי הנסיעה בהתאם לנתונים הנוכחיים.

כך תתכן אפשרות שבכל נקודה בנסיעה האלגוריתם עשוי להורות לנהג לאסוף עוד נוסעים אם הם מתאימים לפרופיל של הנסיעה.

הגדרת הבעיה האלגוריתמית :

הגדרת מוצא ויעד לכל נסיעה ומציאת המסלול הקצר ביותר בין 2 הנקודות.

חישוב דינמי לשינוי מסלול הנסיעה הקצר ביותר האפשרי שמחבר בין הנוסעים והיעדים, כך

שהמסלול לא יהיה ארוך או סיבובי מדי.

זיהוי נקודות האיסוף וההורדה של מזמיני הנסיעה ובדיקה האם הן תואמות את המסלול הקיים ולא משנות אותו יתר על המידה.

שינוי מסלול נסיעה לרכב שנמצא מתאים לבקשה חדשה של נוסע.

רקע תיאורטי בתחום הפרויקט :

אדם ממוצע המשתמש בתחבורה ציבורית, חווה לא פעם מצבים וסיטואציות מתסכלות כמו המתנה מורטת עצבים לאוטובוס מתממה, עמידה ממושכת בנסיעה מחוסר במקומות ישיבה, ולעיתים אפילו אוטובוסים מלאים שאינם יכולים לקלוט נוסעים חדשים בתחנה ולכן מדלגים עליה.

המקרים האלו גרמו לי לחשוב על פתרונות אפשריים, שיחסכו לכולנו זמן וחוסר נוחות.

שורש הבעיה נובע משתי עובדות מרכזיות :

א : זמן.

ב : מקום.

פעמים רבות ניתן לראות שהיצמדות הנהג למסלול קבוע שהוכתב לו מראש גורמת לא אחת לסיבובים מיותרים בכל רחבי העיר, סיבובים שאינם נצרכים כלל, מכיון שאין באותה עת

רבקה סימן טוב Pop-Up driver

באוטובוס אף נוסע שמעוניין לרדת או לעלות בתחנות האלו.

הגורם לכך הוא שאין לחברה המתפעלת את מערך האוטובוסים אמצעים מדויקים לדעת כמה נוסעים יעלו לכל אוטובוס, באילו נקודות עליה וירידה, ועל כן אין באפשרותה להערך לכך מראש.

כדי לתת מענה לשתי הבעיות האלו, החלטתי לבנות מערכת שתתמוך באילוצי זמן ומקום ותצור מסלול אינדיבידואלי לכל נסיעה באופן מיטבי, כמובן תוך התחשבות במקומות הפנויים הקיימים ברכב.

תהליכים עיקריים בפרויקט :

-הוספת משתמש חדש למערכת.

-הוספת נסיעה ומספר המקומות המוצעים בה.

- הגדרת מסלול בסיס לנסיעה חדשה.

- זיהוי מיקומי הרכבים הרשומים בנסיעות.

-מציאת המסלול היעיל והקצר ביותר בכל רגע נתון.

-מציאת זמן משוער עד להגעה אל היעד של כל משתמש.

-הוספת בקשה לנסיעה שיתופית.

- מציאת הנסיעה המתאימה.

- שינוי מסלול הנסיעה בהתאם.

- אישור בעל הרכב על השינוי.

- הודעה לנסע על כך שנמצאה עבורו התאמה.

- זיהוי נסיעה שהסתיימה ומחיקתה מהמערכת.

תיאור הטכנולוגיה :

צד שרת :

שפת תכנות בצד השרת : C# - שפת תכנות עילית מרובת-פרדיגמות, מונחית עצמים בעיקרה, המשלבת רעיונות כמו טיפוסיות חזקה, אימפרטיביות, הצהרתיות, פונקציונליות, פרוצדורליות וגנריות. השפה פותחה על ידי מיקרוסופט בשנת 2000 כחלק מפרויקט דוט נט.

צד לקוח :

שפת תכנות בצד הלקוח :

React Native -הוא שלד תוכנה (פריימוורק) לפיתוח ממשק משתמש שנוצר על ידי חברת פייסבוק הוא משמש לפיתוח יישומים עבור אנדרואיד, macOS, tvOS, iOS, Android, TV , פייסבוק אינטרנט, Windows ו-UWP בזכות היכולת לנצל את היכולות המובנות של מערכות ההפעלה השונות תוך שימוש בשלד התוכנה React. הוא משמש גם לפיתוח יישומי מציאות מדומה ב-Oculus.

מסד נתונים : server SQL – מערכת לניהול בסיסי נתונים.

פרוטוקולי תקשורת : HTTP, API

לוחות זמנים :

מתאריך 2021/10/02 עד תאריך 2021/12/20 חקר הפרויקט וכתיבת הצעה.

רבקה סימן טוב Pop-Up driver

מתאריך 2022/01/02 עד תאריך 2022/01/30 כתיבת מבניות הפרויקט.

מתאריך 2022/02/01 עד תאריך 2022/02/20 הקמת מסד נתונים.

האלגוריתם כתיבת 30/03/2022 תאריך עד 21/02/2022 מתאריך

מתאריך 2022/04/01 עד תאריך 2022/05/01 הקמת ממשק משתמש.

חתימת הסטודנט :



חתימת רכז המגמה :

אישור משרד החינוך :

מבוא

הרקע לפרויקט

הפרויקט עוסק בניהול מערכת של תחבורה שיתופית הפועלת לתכנן מסלול נסיעה בצורה חכמה, כך שיהיה זה המסלול הקצר ביותר החוסך זמן לנוסעים ומקום בכבישי ישראל הפקוקים. בשנים האחרונות עולה בהתמדה מספרם של הרכבים על הכביש, בפרט מספר האוטובוסים.

אדם ממוצע המשתמש בתחבורה ציבורית, חווה לא פעם מצבים וסיטואציות מתסכלות כמו המתנה מורטת עצבים לאוטובוס מתממה, עמידה ממושכת בנסיעה מחוסר במקומות ישיבה, ולעיתים אפילו אוטובוסים מלאים שאינם יכולים לקלוט נוסעים חדשים בתחנה ולכן מדלגים עליה.

המקרים האלו גרמו לי לחשוב על פתרונות אפשריים, שיחסכו לכולנו זמן וחוסר נוחות.

שורש הבעיה נובע משתי עובדות מרכזיות:

א: זמן.

ב: מקום.

פעמים רבות ניתן לראות שהיצמדות הנהג למסלול קבוע שהוכתב לו מראש גורמת לא אחת לסיבובים מיותרים בכל רחבי העיר, סיבובים שאינם נצרכים כלל, מכיון שאין באותה עת באוטובוס אף נוסע שמעוניין לרדת או לעלות בתחנות האלו.

הגורם לכך הוא שאין לחברה המתפעלת את מערך האוטובוסים אמצעים מדויקים לדעת כמה נוסעים יעלו לכל אוטובוס, באילו נקודות עליה וירידה, ועל כן אין באפשרותה להעריך לכך מראש.

כדי לתת מענה לשתי הבעיות האלו, החלטתי לבנות מערכת שתתמוך באילוצי זמן ומקום ותצור מסלול אינדיבידואלי לכל נסיעה באופן מיטבי, כמובן תוך התחשבות במקומות הפנויים הקיימים ברכב.

תהליך המחקר

קודם לכתובת הפרויקט היה עלי ללמוד שפה חדשה בה אני עתידה לכתוב את ה FrontEnd ולדעת לקדד בה - React native.

לצורך כך ערכתי עבודת מחקר, רבות בעזרת מנוע החיפוש של גוגל, קראתי חומרים רבים בתחום זה והגעתי למסקנה שטוב יותר הוא להשתמש ב Expo.

Expo היא פלטפורמת קוד פתוח ליצור אפליקציות אוניברסליות עבור אנדרואיד, iOS ו WEB תוך שימוש ב JavaScript and React Native.

קראתי בעיון את ה Documentation של Expo. שם נוכחתי לראות שלצורך הרצת האפליקציה אני צריכה להשתמש במערכת הפעלה של אנדרואיד, אם ב android device ואם ב android emulator.

מאחר ואין ברשותי טכנולוגיות מובייל האפשרות שנותרה בידי הייתה להתקין android emulator.

ערכתי חיפושים רבים ברחבי הרשת במטרה לדעת מה האפשרות הטובה ביותר מבין שלל האפשרויות של אימולטורי אנדרואיד (Andy, Android studio, BlueStacks 3, Genymotion).

האפשרות הטובה ביותר שנמצאה הייתה Android studio מאחר ויש לו את הפונקציה Instant Run המאפשרת לנו להחיל שינויים בקוד ובמשאבים ביישום במהלך ביצועה. זה מפרש באופן

רבקה סימן טוב Pop-Up driver

אינטליגנטי את השינויים ויכול לספק אותם מבלי להפעיל מחדש את האפליקציה או לחבר מחדש את ה-APK.

לאחר התקנת Android Studio יצרתי אימולטור בהתאמה אישית למחשב שלי ולמטרות הפרויקט.

בשלב מתקדם יותר במהלך הפרויקט, נוכחתי לראות שכאשר אני מנסה להתחבר מהמכשיר הווירטואלי לכתובת ה-localhost שלי (בעזרת המתודה fetch), בכדי לאחזר נתונים ממסד הנתונים ומצד השרת שלי, נתקלתי בשגיאת רשת חמורה:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell.exe
Network Error
at node_modules\expo\build\environment\react-native-logs.fx.js:27:4 in error
at node_modules\regenerator-runtime\runtime.js:63:36 in tryCatch
at node_modules\regenerator-runtime\runtime.js:294:29 in invoke
at node_modules\regenerator-runtime\runtime.js:63:36 in tryCatch
at node_modules\regenerator-runtime\runtime.js:155:37 in invoke
```

כדי לפתור בעיה זו נסיתי להחליף את השימוש ב-Fetch ולהשתמש במקומו ב-axios שידוע כבטיחותי יותר אך השגיאה נותרה בעינה.

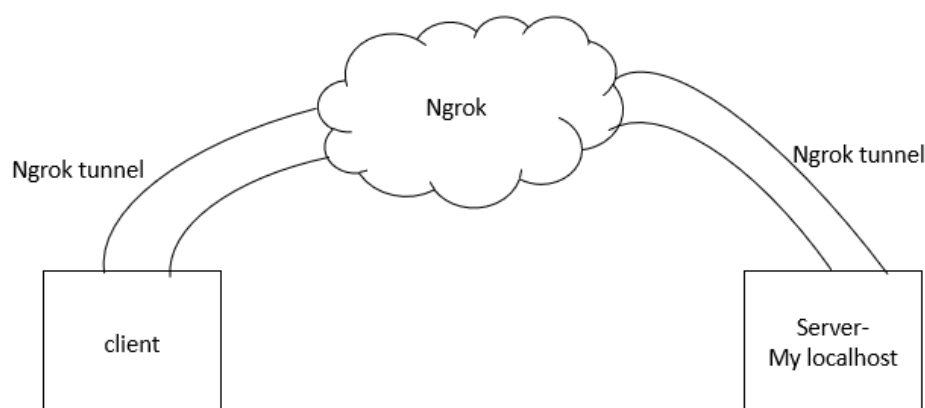
בחיפוש שערכתי לאיתור באגים דומים של מפתחים אחרים מצאתי שהבעיה הזו ידועה בקרב מפתחי expo כשמריצים על אנדרואיד דווקא ופתרון ברור וסופי עדיין לא נמצא.

לאחר עמל ויגיעה רבה גיליתי בסופו של דבר פתרון ש"עוקף" את הבעיה הזו בצורה חכמה להפליא: NGROK:

Ngrok היא נקודת קצה ברשת הניתנת לתכנות ומוסיפה קישוריות לאפליקציה ללא שינוי קוד.

היא משמשת כעין tunnel בין שני נקודות קצה ברשת בצורה מאובטחת היטב.

למעשה מדובר ביישום חוצה פלטפורמות שחושף יציאות שרת מקומיות לאינטרנט, בצורה הבאה כדלהלן:

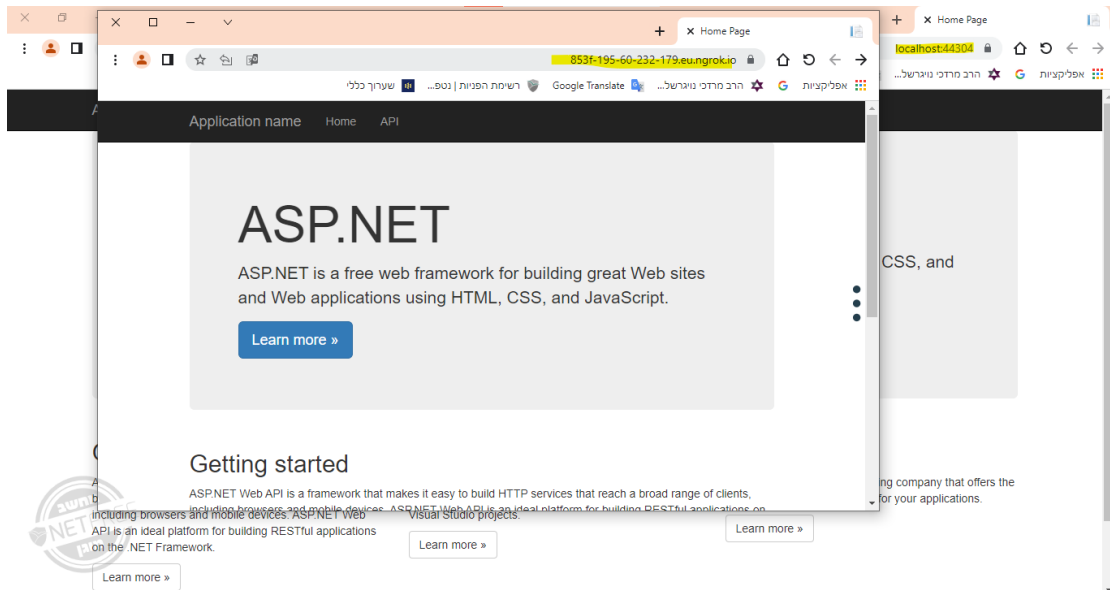


יצירת ה tunnel מתבצעת ע"י פקודה אחת שמגדירה את הפרט עליו תוקם ה"מנהרה".

```
C:\Windows\System32\cmd.exe
:\Users\WIN 10\Downloads\1>ngrok http https://localhost:44304 --host-header="localhost:44304"
```


רבקה סימן טוב Pop-Up driver

ואכן, בצורה זו הצלחתי ברוך ה' לגשת מצד הלקוח לצד השרת בצורה בטוחה ואף ללא השפעה על יעילות הזמן, כפי שניתן לראות בתמונה שלהלן:



סקירת ספרות

בכל תהליך המחקר נעזרתי רבות באתרים מקצועיים מעולים.

ברקע הטכני והתאורטי:

[/https://reactnative.dev](https://reactnative.dev)

[/https://expo.dev](https://expo.dev)

[/https://developer.android.com](https://developer.android.com)

[/https://stackoverflow.com](https://stackoverflow.com)

[/https://github.com](https://github.com)

[/https://ngrok.com](https://ngrok.com)

ברקע התכנותי התאורטי:

<https://simpledevcode.wordpress.com/2015/12/22/graphs-and-dijkstras-algorithm-c/>

<https://dev.to/russianguycoding/how-to-represent-a-graph-in-c-4cmo>

<https://developpaper.com/c-implementation-of-dijkstra-algorithms/><https://gocode.co.il/content/rn-course-5>

[/https://google.com](https://google.com)

[/https://youtube.com](https://youtube.com)

<https://developers.google.com/maps>

אתגרים מרכזיים

הבעיה איתה התמודד התלמיד במטרה להקים מערכת שתנהל באופן יעיל תחבורה שיתופית, האתגר המרכזי שעליו מתבסס שהרעיון כולו הוא למצוא את המסלול הקצר ביותר לנהג בין כל נוסעיו. הסיבות לבחירת הנושא בחרתי בנושא זה מאחר ואני משתמשת קבועה בתחבורה ציבורית, וכתוצאה מכך אני עדה לליקויים הרבים שנמצאים במערכת. החלטתי לא להישאר שם אלא לשאת עיניים לעתיד טוב יותר, ולהפיק תועלת לעצמי ולאחרים על ידי פתרון הבעיה הזו.

מטרות ויעדים

המטרה הראשונה שעמדה למולי היא להצליח להתפתח בצורה עצמאית, ללמוד גם שפות חדשות שלא הכרתי, (React Native) ללמוד על טכנולוגיות שמאז מבוקשות בשוק כיום (כגון, Google Expo Maps), לרכוש ידע בהתקנות חדשות, ולהרחיב ולשפר את המיומנות שלי במה שאני כבר יודעת, ולהעמיק את הידע עד לרמה מקצועית.

מטרת העל:

חיסכון במשאבים כגון כח אדם וזמן. הפחתת זיהום האוויר הנוצר מעודף תחבורה על הכביש.

מטרות נוספות:

- שמירת הנתונים באופן מסודר ויעיל.
- אחזור מהיר ונוח של הנתונים.
- בניית אפליקציה נעימה לעין ונוחה לשימוש.
- תכנון המערכת תוך שימת דגש על ארגון הקוד בצורה יעילה.

יעדים:

- קבלת מרחקים בין שתי נ"צ על המפה מממשק Google Maps, וחישוב המסלול הקצר ביותר בין כל הנקודות.
- אפשרות הזמנת נסיעה ע"י נוסע וקבלת תגובה מהמערכת בצורה יעילה.
- אפשרות הוספת נסיעה למערכת ע"י נהג וקבלת תגובה יעילה מהמערכת בדמות המסלול בו עליו לנסוע.

מדדי הצלחה למערכת

- כניסה מאובטחת לאזור האישי.
- הכנסת פרטי נסיעה בצורה נכונה תוך שמירת מיקומם הנכון ב-DB.
- הכנסת פרטי נהג חדש למערכת בצורה יעילה.
- הצגת מסלול קצר עבור כל נהג באופן דינאמי אך מדויק.

רקע תיאורטי / ספרות מקצועית

בעיית הנתיב הקצר ביותר היא הבעיה של מציאת נתיב בין שני קודקודים (או צמתים) בגרף כך שסכום המשקולות של הקצוות המרכיבים אותו ממוזער.

הבעיה של מציאת הנתיב הקצר ביותר בין שני צמתים במפת דרכים עשויה להיות מודל למקרה מיוחד של בעיית הנתיב הקצר ביותר בגרפים, כאשר הקודקודים תואמים לצמתים והקצוות תואמים לקטעי דרך, כל אחד משוקלל באורך של מִגָזָר.

תיאור מצב קיים

ישנן מספר אפשרויות לפתור את בעיית מציאת הנתיב הקצר איתה התמודדתי.

אפרט כמה מהם כדלהלן:

ניתוח חלופות מערכת

ישנם מספר אלגוריתמים הפועלים לחשב את המסלול הקצר ביותר בין מספר נקודות.

אלגוריתם Dijkstra-

סיבוכיות זמן ריצה: $O(E \log V)$

אלגוריתם בלמן-פורד-

חסרונות: סיבוכיות הזמן של אלגוריתם בלמן-פורד גדולה $O(V \cdot E)$

וכן עובד גם עם משקולות שליליים- מה שאצלנו יצור תוצאות שאינן נכונות.

אלגוריתם DAG :

יתרונותיו: סיבוכיות הזמן שלו יעילה יותר משאר האלגוריתמים $O(V+E)$.

חסרונותיו: אלגוריתם DAG עובד רק על גרף ממושקל ומכוון ללא מעגלים, ובגרף שלנו אי אפשר לדעת מראש האם יהיו מעגלים בגרף שיווצר ע"י נתוני המשתמש.

תיאור החלופה הנבחרת

למעשה כאשר בחנתי את נושא האלגוריתמים דלעיל לעומקו, נוכחתי לראות שאלגוריתמים אלו לא מחייבים לעבור בכל הקודקודים שברשותם, או במקרה זה- לא מחייבים לאסוף את כל הנוסעים שהזמינו נסיעה מה שמתגלה כבעייתי ביותר מפני שהאלגוריתם חייב לתת מענה לאיסוף כל הנוסעים. (איור 1)

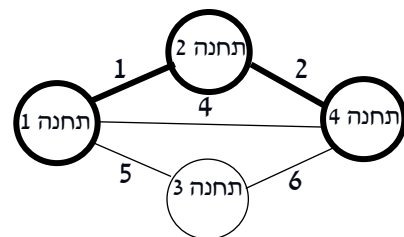
הפתרון הבא שבדקתי האם הוא יכול לפתור את הבעיה הינו שימוש באלגוריתם למציאת עץ פורש מינימלי.

אלגוריתם זה מוצא פרישה של כל הקודקודים בגרף כאשר הקשתות שמחברות ביניהן הן קשתות בעלות משקל מינימלי וללא מעגלים.

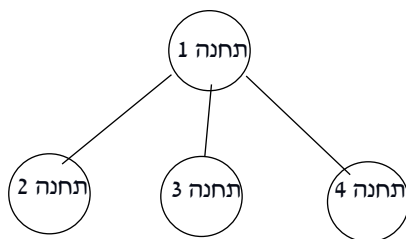
במקרה שלנו- מוצא דרך מכל מוצר שנאסף אל כל מוצר בצורה הקצרה ביותר בלי לעבור במוצר פעמיים.

אך שוב, לאחר חקירת האלגוריתם ובדיקה מדוקדקת, ראיתי שהאלגוריתם אומנם נותן מסלול בין תחנה אחת לשנייה, אך לא נותן מסלול אחד, בין כל התחנות. (איור 2)

איור 1:



איור 2:



במקרה זה רואים שהאלגוריתם אומנם מחזיר פרישה מינימלית של הגרף, אולם כפי שאפשר להיווכח, ישנו מסלול מכל תחנה אל כל תחנה, אך אין מסלול יחיד בין כל התחנות יחד.

במקרה זה רואים שהאלגוריתם אומנם יחזיר מסלול קצר אך לא כולל את כל התחנות. דבר שלא נותן מענה למטרה.

לאחר שהבנתי שהאלגוריתמים אשר הוזכרו לעיל אינם נותנים את המענה המבוקש למציאת מסלול קצר בין כל הנוסעים, גיליתי שבעיה זו היא בעיה מוכרת בקרב המתכנתים והיא נקראת "בעיית הסוכן הנוסע".

זוהי בעיה שכבר עשרות שנים מנסים למצוא לה פתרון יעיל אך לא עלה בידיהם של העוסקים בדבר לפתור אותה אלה רק בצורה נאיבית בה מחשבים את כל המסלולים האפשריים בעבור כל מוצר וכך בוחרים את המסלול הקצר מבין כולם.

סיבוכיות זמן הריצה של השימוש הנאיבי גדולה מאוד ($O(n)$).

אולם הם הצליחו למצוא קרובים לבעיה אשר אינן נותנים בהכרח פתרון אופטימלי לבעיה, אך נותנים פתרון טוב מספיק בזמן ריצה קצר בהרבה מהפתרון האופטימלי אך הלא יעיל...

קרוב אחד הוא שימוש באלגוריתם Dijkstra חמדני.

Dijkstra חמדני עובד כמעט באותה דרך בה עובד אלגוריתם Dijkstra רק השינוי הוא שחיפוש היעד הבא במסלול הוא כל פעם מהיעד הקודם.

רבקה סימן טוב Pop-Up driver

כלומר לא מחפשים מסלול קצר מהמקור בלבד, אלה מוצאים מסלול כל פעם ליעד הקרוב ביותר, וממנו אל היעד הקרוב ביותר אליו וכן הלאה עד למעבר בין כל התחנות.

אפיון המערכת שהוגדרה ניתוח דרישות המערכת

חומרה : מעבד RAM 8GB i5

עמדת פיתוח : מחשב Intel

מערכת הפעלה : Windows 10

שפות תוכנה : C# תוך שימוש בטכנולוגיות Expo, webApi, React Native.

כלי תוכנה לפיתוח המערכת: Visual Studio code, Visual studio 2019, Microsoft

מסד נתונים : SQL server.

עמדת משתמש מינימלית :

- חומרה : מעבד RAM 8GB i5.
- מערכת הפעלה : Win10 .
- חיבור לרשת : חובה.
- תוכנות : Android Emulator / Android device.

מודול המערכת

נושאים באחריות המערכת :

- הכנסת פרטי משתמשים חדשים למערכת.
- הוספת נסיעה חדשה למערכת ע"י משתמש נהג.
- הוספת בקשת נסיעה למערכת ע"י נוסע חדש.
- בניית מסלול אינדיבידואלי עבור כל נהג בצורה המיטבית.
- עדכון הנוסע לגבי מציאת הנהג.

רבקה סימן טוב Pop-Up driver

נושאים שאינם באחריות המערכת :

- המערכת לא אחראית לבדיקת אמיתותם של הפרטים הנוספים ע"י המשתמשים.
- המערכת לא אחראית על התשלום עבור הנהג.
- המערכת לא אחראית לטיב הנוסעים והנהגים.

אפיון פונקציונלי

`BuildDijkstraPath()` - הפונקציה המרכזית של האלגוריתם. מחשבת את המסלול הקצר ביותר לאיסוף הנוסעים עבור נהג מסוים.

`GetTheBestDriverAndInvokeDijkstra()` - הפונקציה מקבלת פרטי נוסע ומחזירה את הנהג המתאים ביותר לנסיעה זו, ומסדרת את המסלול שלו.

`GetDistance()` - הפונקציה מחזירה את המרחק בין 2 תחנות ע"י התממשקות לגוגל מטריקס.

`DistanceURL()` - הפונקציה בונה URL ל-API של Google Maps ומחזירה מרחק בין שתי נ"צ.

`SignUp()/Login()` - התחברות/רישום לאפליקציה.

`GetLocation()` - הפונקציה מחזירה את מיקום הרכב של הנהג ברגע זה.

ביצועים עיקריים

- הוספת משתמש חדש למערכת.
- הוספת נסיעה ומספר המקומות המוצעים בה.
- הוספת בקשה לנסיעה שיתופית.
- זיהוי מיקומי הרכבים הרשומים בנסיעות.
- בניית המסלול הקצר ביותר עבור כל נהג בכל רגע נתון.
- מציאת זמן משוער עד להגעה אל היעד של כל משתמש.

אילוצים

המערכת מסתמכת על נכונות הנתונים שהוכנסו ע"י המשתמשים.

המערכת מסתמכת על פונקציות מממשק google maps וכן על נתוני המיקומים של המשתמשים.

תיאור הארכיטקטורה

הארכיטקטורה של הפתרון המוצע בפורמט של Design level Down-Top

הפרויקט מחולק ל 2 חלקים :

- צד שרת : הנכתב בשפת c# . בטכנולוגיית WebApi .

- צד לקוח : הנכתב ב – React Native .

תיאור צד השרת :

צד השרת מחולק כמקובל לשכבות :

שכבת ה - DAL - Data Access Layer .

שכבת ה – BLL - Business Logic layer

החלוקה לשכבות נועדה להפריד באופן מוחלט בין הלוגיקה של הפרויקט לבין הנתונים עצמם. הפרדה זו מאפשרת לבצע שינויים בכל אחת מהשכבות בלי תלות ובלי זעזועים בשכבות האחרות. פירוט :

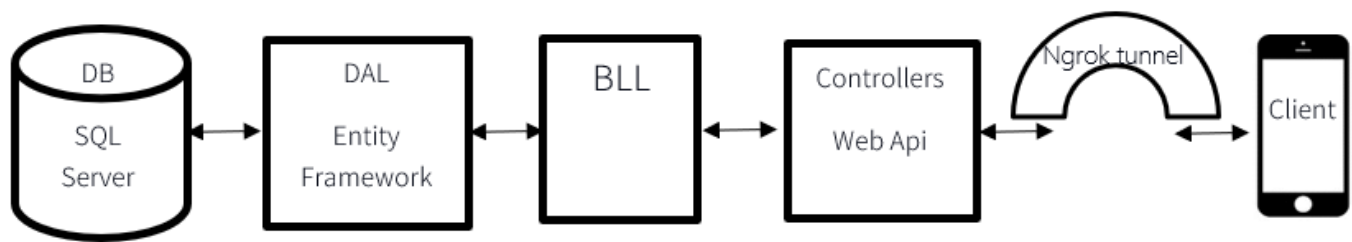
שכבת ה DAL : היא השכבה דרכה ניגשים לנתונים היושבים ב DB. היא מכילה מחלקות המייצגות את בסיס הנתונים וכן פונקציות נחוצות לתפעולו . פעולות ההתקשרות עם בסיס הנתונים נעשו בטכנולוגיית Entity framework. טכנולוגיה זו היא המקובלת לטיפול בבסיס הנתונים כאשר היא מנתחת את בסיס הנתונים, בונה מחלקות לייצוג הטבלאות ומספקת רשימות מלאות בנתוני בסיס הנתונים.

שכבת ה BLL : שכבה שבה כתובה כל הלוגיקה של הפרויקט.

בנוסף קימות מתודות השרת המחצינות את הפעולות שניתן לבצע בשרת. מתודות אלו משתמשות ב BLL ומופעלות ע"י שכבת ה GUI בצד הלקוח. מודל זה אמנם גורם טרחה טכנית לא מעטה בכתיבת הקוד ובתכנון הפונקציות אבל מספק קוד נקי, קל להבנה ונוח לשינויים ולשדרוגים.

רבקה סימן טוב Pop-Up driver

תיאור הרכיבים בפתרון



1. מסד הנתונים הבנוי מטבלאות וקשרי גומלין ביניהן.
2. שכבת הגישה לנתונים באמצעות Entity Framework.
4. שכבת BLL - בה כתוב האלגוריתם.
5. Web Api - פרוטוקול התקשורת בין צד הלקוח וצד השרת.
6. ngrok tunnel (הסבר מפורט [למעלה](#)).
7. צד לקוח JavaScript , React Native.

ארכיטקטורת רשת
לא רלוונטי.

תיאור פרוטוקולי התקשורת
HTTP

שרת-לקוח
צד השרת נכתב בטכנולוגיית Web Api ובשפת C#.

צד הלקוח נכתב ב- React Native.

תיאור הצפנות
לא רלוונטי

ניתוח ותרשים Use / UML cases של המערכת המוצעת

רשימת ה-UC העיקריים של המערכת

-הוספת משתמש חדש למערכת.

-הוספת נסיעה ומספר המקומות המוצעים בה.

- הגדרת מסלול בסיס לנסיעה חדשה.

- זיהוי מיקומי הרכבים הרשומים בנסיעות.

-מציאת המסלול היעיל והקצר ביותר בכל רגע נתון.

-מציאת זמן משוער עד להגעה אל היעד של כל משתמש.

-הוספת בקשה לנסיעה שיתופית.

- מציאת הנסיעה המתאימה.

- שינוי מסלול הנסיעה בהתאם.

- אישור בעל הרכב על השינוי.

- הודעה לנוסע על כך שנמצאה עבורו התאמה.

- זיהוי נסיעה שהסתיימה ומחיקתה מהמערכת.

תיאור ה UC-העיקריים של המערכת

UC1

- UC1 : Identifier
- Name : הוספת נסיעה חדשה.
- Description : הנהג מוסיף את פרטי הנסיעה החדשה שהוא מציע לציבור.
- הפרטים כוללים מידע אודות הרכב.
- Actors : נהג.
- Frequency : בכל עת.
- Conditions-Pre : פרטים בסיסיים לאחר התחברות או רישום.
- Conditions-Post : פרטי הנסיעה נשמרים במערכת והיא פועלת לתכנן לו את המסלול הטוב ביותר.
- Basic course of action : הנהג מוסיף נסיעה מתוך האזור האישי שלו.

UC2

- UC2 : Identifier
- Name : הוספת נוסע חדש.
- Description : בקשת נסיעה חדשה על ידי משתמש המוגדר כנוסע.
- Actors : נוסע
- Frequency : בכל עת.
- Conditions-Pre : פרטים בסיסיים לאחר התחברות או רישום.

רבקה סימן טוב Pop-Up driver

- Conditions-Post : פרטי הבקשה נשמרים במערכת והיא פועלת למצוא את הנהג הקורב ביותר שהגדרות הנסיעה שלו מתאימות לנוסע החדש.
לאחר המציאה, הודעה נשלחת לנוסע.
- Basic course of action : הנוסע מוסיף בקשה מתוך האזור האישי שלו.

UC3

- UC4 : Identifier
- Name : מסלול לאיסוף הנוסעים.
- Description : הנהג מקבל את המסלול בו עליו לנסוע נכון לעכשיו.
- Actors : נהג
- Frequency : לאחר כניסתו למערכת ורישום נסיעה חדשה.
- Conditions-Pre : רשימת הנוסעים השייכים לנסיעה זו.
- Conditions-Post : המערכת מציגה לנהג את המסלול הקצר לאיסוף הנוסעים שלו.

מבנה נתונים בשימוש בפרויקט

SortedList - מבנה נתונים המגדיר אוסף של מפתחות וערכים. למעשה מתנהג כמו מילון ובנוסף שומר על ערכים ממוינים לפי המפתח.

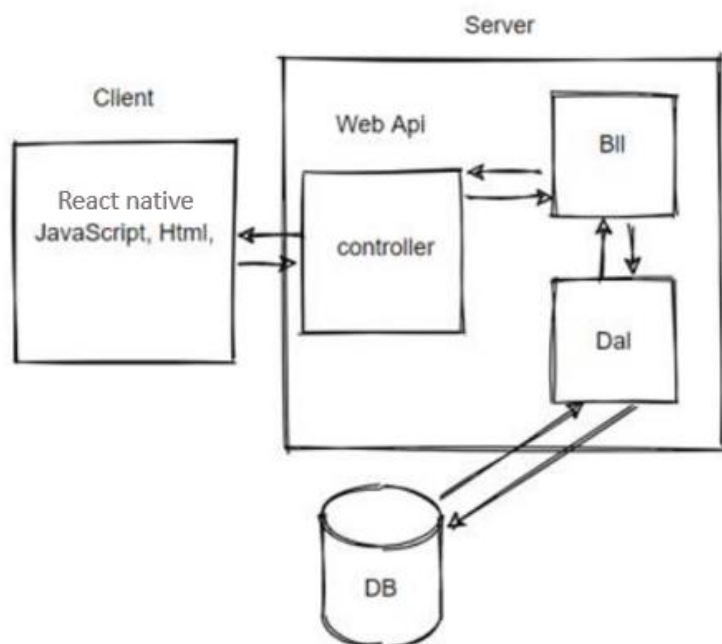
השתמשתי ברשימה ממוינת בעת שרציתי למצוא את המרחק בין תחנה אחת לשאר התחנות במסלול, תוך שמירה על היכולת להגיע לתוצאת המרחק המינימלית ביעילות של $O(1)$.

List - רשימה מקושרת היא מבנה נתונים ליניארי, שבו האלמנטים אינם מאוחסנים במיקומי זיכרון רציפים. האלמנטים ברשימה מקושרת מקושרים באמצעות מצביעים. במילים פשוטות, רשימה מקושרת מורכבת מצמתים כאשר כל צומת מכיל שדה נתונים והפניה (קישור) לצומת הבא ברשימה.

רשימה מאפשרת שמירה של נתונים בצורה סדרתית ללא צורך להגדיר מראש את גודל הרשימה. שפת C# תומכת באפשרויות רבות בכל הנוגע לרשימה, מה שהופך את הקוד לפשוט וברור יותר. השתמשתי ברשימה בכל עת שרציתי לשמור אוסף של רשומות.

רבקה סימן טוב Pop-Up driver

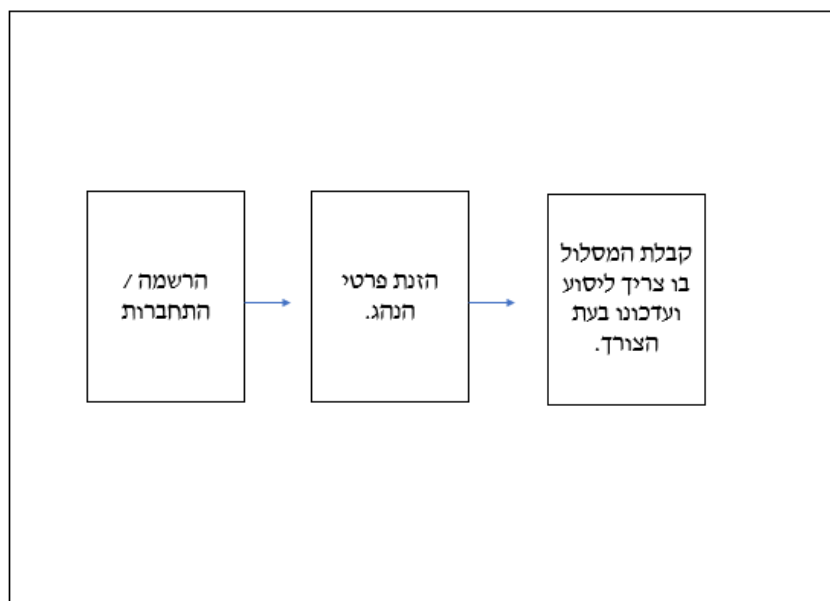
הקשרים בין היחידות השונות



עץ מודולים
Use case Diagram

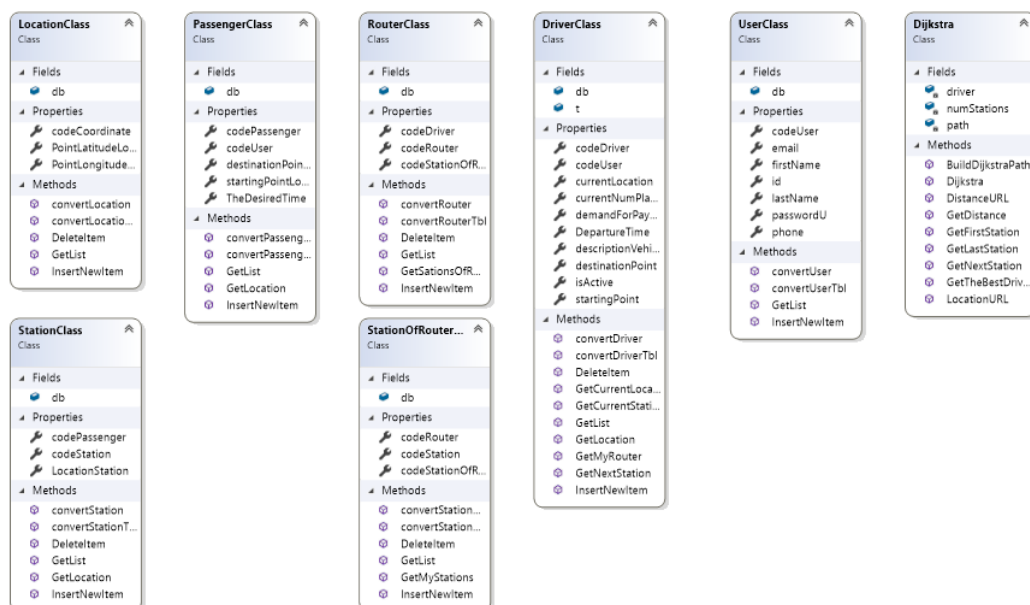


רבקה סימן טוב Pop-Up driver



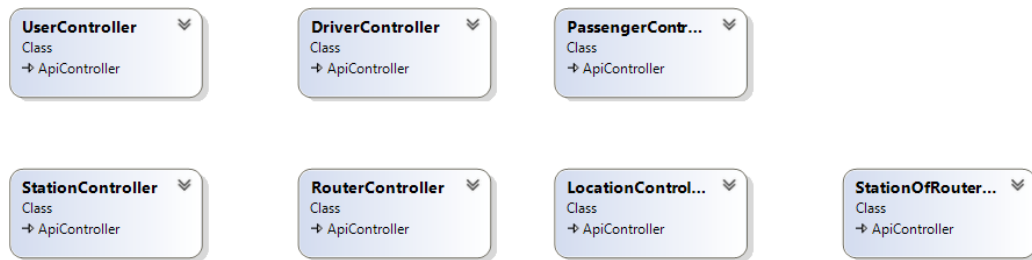
Design class Diagram

שכבת ה-BLL- האלגוריתמיקה :



שכבת API:

רבקה סימן טוב Pop-Up driver



תיאור המחלקות המוצעות

UserClass

מחלקת משתמשים

הפונקציות:

- `public static UserClass convertUser(Users_tbl u)`
פונקציה המקבלת עצם מטיפוס `User_tbl` וממירה אותו לעצם מטיפוס המחלקה.

- `public static Users_tbl convertUserTbl(UserClass u)`
פונקציה המקבלת עצם מטיפוס המחלקה וממירה אותו לעצם מטיפוס הטבלה.

- `public static void InsertNewItem(UserClass uc)`
פונקציה המקבלת עצם מטיפוס המחלקה ומוסיפה אותו לDB

- `()public static List<UserClass> GetList`
פונקציה המחזירה את רשימת המשתמשים מהDB

PassengerClass

מחלקת נוסעים

הפונקציות:

- `public static PassengerClass convertPassenger(Passengers_tbl P)`

רבקה סימן טוב Pop-Up driver

פונקציה המקבלת עצם מטיפוס Passenger_tbl וממירה אותו לעצם מטיפוס המחלקה.

- `public static Passengers_tbl convertPassengerTbl(PassengerClass P)`
המקבלת עצם מטיפוס המחלקה וממירה אותו לעצם מטיפוס הטבלה.
- `()public static List<PassengerClass> GetList`
פונקציה המחזירה את רשימת הנוסעים מהDB
- `()public static List<UserClass> GetList`
פונקציה המקבלת עצם מטיפוס המחלקה ומוסיפה אותו לDB
- `public LocationClass GetLocation(int code)`
פונקציה המקבלת קוד מיקום ומחזירה עצם מטיפוס מחלקת מיקום

DriverClass

מחלקת נהגים

הפונקציות :

- `public static DriverClass convertDriver(Drivers_tbl P)`
פונקציה המקבלת עצם מטיפוס Driver_tbl וממירה אותו לעצם מטיפוס המחלקה.
- `public static Drivers_tbl convertDriverTbl(DriverClass u)`
פונקציה המקבלת עצם מטיפוס המחלקה וממירה אותו לעצם מטיפוס הטבלה.
- `public static void InsertNewItem(DriverClass uc)`
פונקציה המקבלת עצם מטיפוס המחלקה ומוסיפה אותו לDB
- `()public static List<DriverClass> GetList`
פונקציה המחזירה את רשימת הנהגים מהDB
- `public static void DeleteItem(int id)`
פונקציה המקבלת קוד נהג ומוחקת אותו מה DB

רבקה סימן טוב Pop-Up driver

- `public LocationClass GetLocation(int code)`
פונקציה המקבלת קוד מיקום ומחזירה עצם מטיפוס מחלקת מיקום
- `()public List<StationOfRouterClass> GetMyRouter`
פונקציה המחזירה את המסלול של הנהג
- `()public StationClass GetCurrentStation`
פונקציה המחזירה את התחנה הנוכחית של הנהג
- `()public StationClass GetNextStation`
פונקציה המחזירה את התחנה הבאה של הנהג

LocationClass

מחלקת מיקומים

הפונקציות:

- `public static LocationClass convertLocation(Location_tbl u)`
פונקציה המקבלת עצם מטיפוס Location_tbl וממירה אותו לעצם מטיפוס המחלקה.
- `public static Location_tbl convertLocationTbl(LocationClass u)`
פונקציה המקבלת עצם מטיפוס המחלקה וממירה אותו לעצם מטיפוס הטבלה.
- `public static void InsertNewItem(LocationClass uc)`
פונקציה המקבלת עצם מטיפוס המחלקה ומוסיפה אותו לDB
- `()public static List<LocationClass> GetList`
פונקציה המחזירה את רשימת המיקומים מהDB

StationClass

מחלקת תחנות

הפונקציות:

רבקה סימן טוב Pop-Up driver

- `public static StationClass convert Station(Station_tbl u)`
פונקציה המקבלת עצם מטיפוס `Station_tbl` וממירה אותו לעצם מטיפוס המחלקה.

- `public static Station_tbl convertUserTbl(StationClass u)`
פונקציה המקבלת עצם מטיפוס המחלקה וממירה אותו לעצם מטיפוס הטבלה.

- `public static void InsertNewItem(StationClass uc)`
פונקציה המקבלת עצם מטיפוס המחלקה ומוסיפה אותו לDB

- `(public static List< StationClass> GetList`
פונקציה המחזירה את רשימת התחנות מהDB

- `public LocationClass GetLocation(int code)`
פונקציה המקבלת קוד מיקום ומחזירה עצם מטיפוס מחלקת מיקום

StationOfRouterClass

מחלקת תחנה למסלול

הפונקציות :

- `public static StationOfRouterClass`
`convertStationOfRouter(StationsOfRouter_tbl P)`
פונקציה המקבלת עצם מטיפוס `StationOfRouter_tbl` וממירה אותו לעצם מטיפוס המחלקה.

- `public static StationsOfRouter_tbl`
`convertStationOfRouterTbl(StationOfRouterClass P)`
מטיפוס המחלקה וממירה אותו לעצם מטיפוס הטבלה.

- `public static void InsertNewItem(StationOfRouterClass uc)`
פונקציה המקבלת עצם מטיפוס המחלקה ומוסיפה אותו לDB

רבקה סימן טוב Pop-Up driver

- `()public static List< StationOfRouterClass > GetList`

פונקציה המחזירה את רשימת התחנות למסלול מהDB

RouterClass

מחלקת מסלול

הפונקציות :

- `public static RouterClass convertRouter(Routers_tbl u)`

פונקציה המקבלת עצם מטיפוס Router_tbl וממירה אותו לעצם מטיפוס המחלקה.

- `public static Routers_tbl convertRouterTbl(RouterClass P)`

פונקציה המקבלת עצם מטיפוס המחלקה וממירה אותו לעצם מטיפוס הטבלה.

- `public static void InsertNewItem(RouterClass uc)`

פונקציה המקבלת עצם מטיפוס המחלקה ומוסיפה אותו לDB

- `()public static List<RouterClass> GetList`

פונקציה המחזירה את רשימת המסלולים מהDB

- `public static void DeleteItem(int id)`

פונקציה המקבלת קוד מסלול ומוחקת אותו מה DB

- `public static List<StationOfRouterClass> GetSationsOfRouterForDriver(int codeDriver)`

פונקציה המקבלת קוד נהג ומחזירה את רשימת התחנות למסלול שלו.

Dijkstra

מחלקת האלגוריתם העיקרי של הפרויקט- דייקסטרה.

הפונקציות :

- `public List<StationClass> BuildDijkstraPath(DriverClass driver, PassengerClass passenger)`

רבקה סימן טוב Pop-Up driver

זוהי הפונקציה שמממשת את האלגוריתם של דייקסטרה.

הפונקציה מקבלת נהג ונוסע ומחזירה את המסלול המתוקן של הנהג לאחר הוספת הנוסע.

- `public (DriverClass ,List<StationClass>)`

`GetTheBestDriverAndInvokeDijkstra(PassengerClass passenger)`

פונקציה זו מופעלת בעת הוספת נוסע למערכת.

הפונקציה מחפשת מיהו הנהג היעיל ביותר עבור הנוסע, שנמצא בסמיכות אליו והמסלול שלו ישתנה בצורה מועטה ביחס לשאר הנהגים, ומפעילה את האלגוריתם של דייקסטרה לחשב את מסלולו של הנהג שנבחר עם הוספת הנוסע החדש.

הפונקציה מקבלת נוסע ומחזירה את הנהג שנבחר ואת המסלול החדש שלו.

- `public async Task<string> GetDistance(LocationClass l1, LocationClass l2)`

הפונקציה מחזירה מרחק בין שתי מיקומים מאת `google matrix api`.

רכיבי ממשק

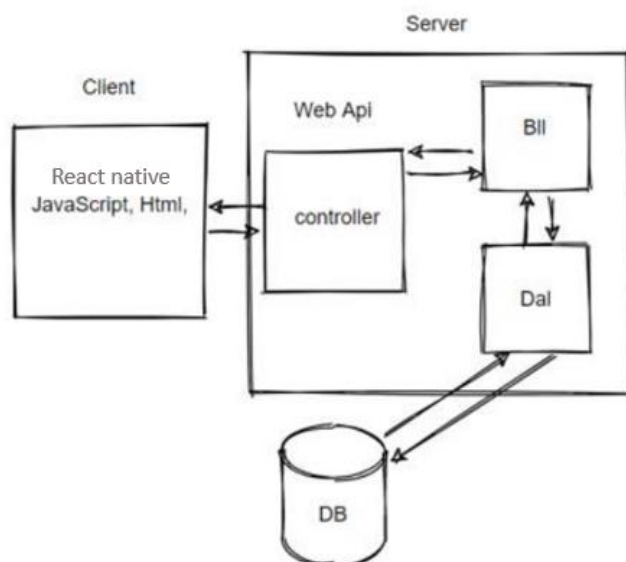
המערכת מורכבת מ-2 חלקים :

1. תוכנית הלקוח.

2. תוכנית השרת.

תיכון המערכת

ארכיטקטורת המערכת



תיכון מפורט

השתמשתי בטכנולוגיית API שהוא ערכה של ספריות קוד, פקודות, פונקציות ופרוצדורות מן המוכן, בהן יכולים המתכנתים לעשות שימוש פשוט, בלי להידרש לכתוב אותן בעצמם כדי שיוכלו להשתמש במידע של היישום שממנו הם רוצים להשתמש לטובת היישום שלהם.

חלופות לתיכון המערכת

יכולתי להשתמש בטכנולוגיית MVC אבל יש לה כמה חסרונות מול API :

1. MVC Net.Asp משמש ליצירת יישומי אינטרנט שמחזירים תצוגות ונתונים, אך

API Web Net.Asp משמש ליצירת שירותי HTTP מלאים בצורה קלה ופשוטה

המחזירה נתונים בלבד ולא תצוגה.

2. API Web. עוזר לבנות שירותי REST מלאים על פני Framework NET. והוא תומך

גם במשא ומתן על תוכן) זה החלטה על נתוני פורמט התגובה הטובים ביותר שיכולים

להיות מקובלים על ידי הלקוח. זה יכול להיות JSON, XML, ATOM או נתונים

מעוצבים אחרים, (אירוח עצמי שאינו ב-MVC).

3. API Web. דואג גם להחזיר נתונים בפורמט מסוים כמו XML, JSON או כל דבר

אחר המבוסס על כותרת.

רבקה סימן טוב Pop-Up driver

4. ב API Web - הבקשה ממופה לפעולות המבוססות על פעלים ב HTTP - אך ב MVC - היא ממופה לשם הפעולות.

5. API Web Net.Asp הוא מסגרת חדשה וחלק ממסגרת הליבה של NET.ASP. כריכת המודל, המסננים, הניתוב ותכונות אחרות של MVC קיימות ב API Web - שונות מ MVC - וקיימות במכלול Http.Web.System החדש. ב MVC - תכונות אלה קיימות בתוך Mvc.Web.System מכאן שניתן להשתמש ב API Web - גם עם Net.Asp וכשכבת שירות עצמאית.

6. יתר על כן, API Web הוא ארכיטקטורה קלילה, פרט ליישום האינטרנט, ניתן להשתמש בו גם עם אפליקציות למובייל.

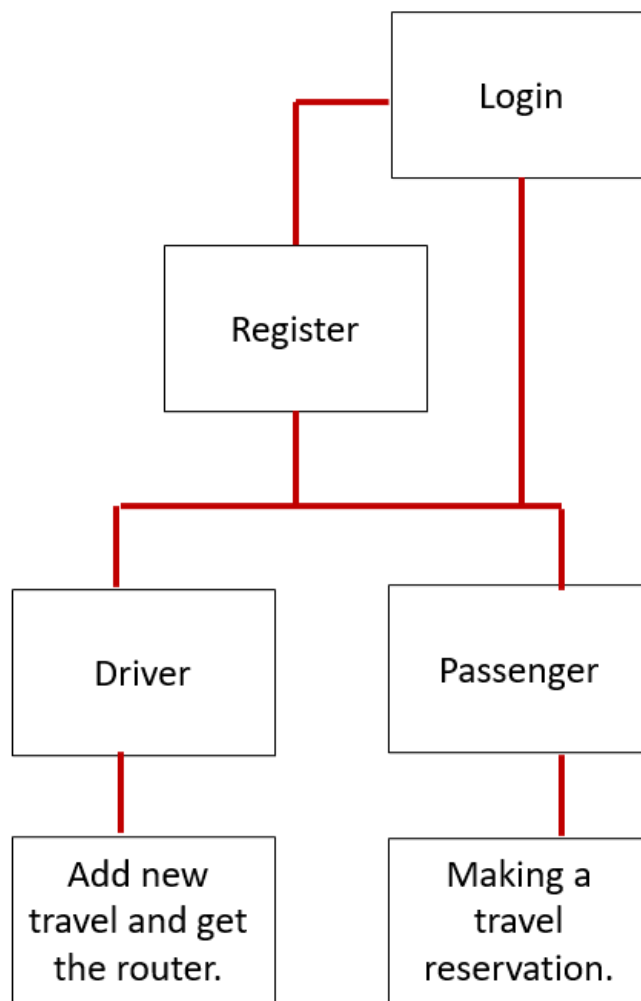
תיאור התוכנה

- סביבת עבודה :
OS: Windows 10
Visual Studio Code | Visual Studio | Android studio-emulator
- שפות תכנות :
צד השרת נכתב בטכנולוגיית web api ובשפת C#.
צד הלקוח נכתב בשפת React Native.

תיאור מסכים

- מסך כניסה למערכת
- מסך רישום משתמש חדש
- מסך התפצלות נהג/נוסע
- מסך נוסע : בקשת נסיעה חדשה.
- מסך נהג : הוספת נהג חדש. הצגת המסלול על המפה.

תרשים מסכים המתאר את היררכיית המסכים והמעברים
ביניהם (diagram flow Screen)



תפקידו של כל מסך.

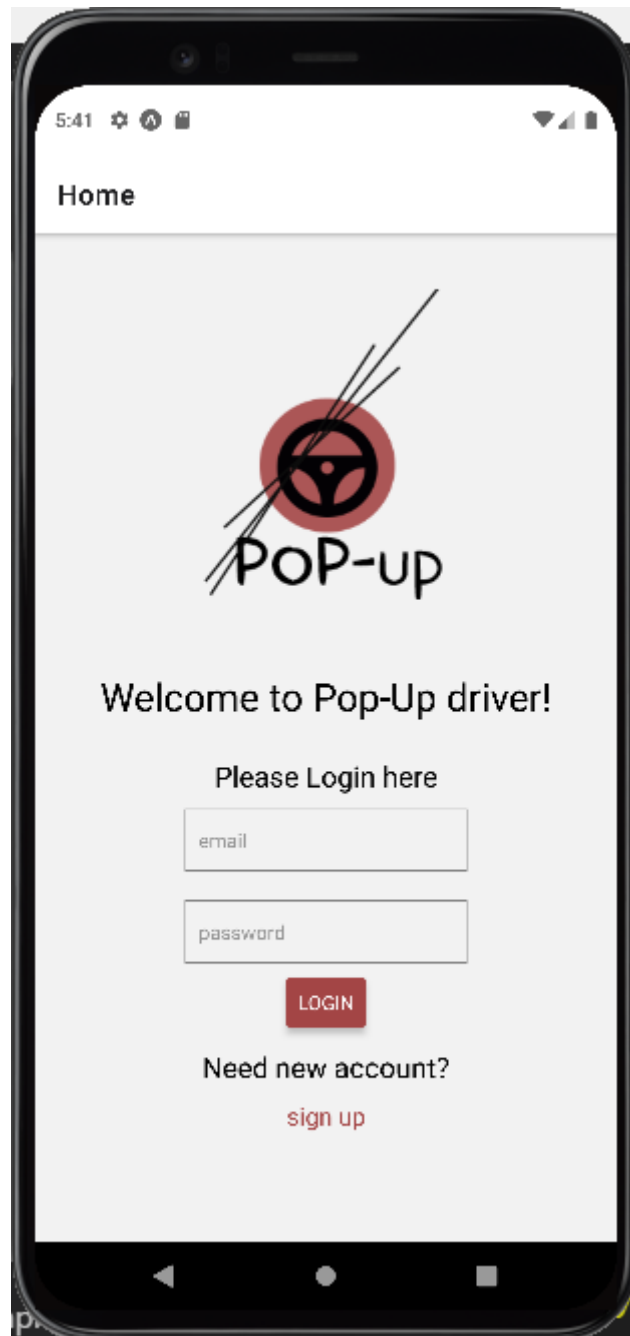
מסך הפתיחה.

מסך כניסה למערכת.

מכיל אפשרות התחברות מאובטחת לאזור האישי.

או אפשרות לצור חשבון חדש.

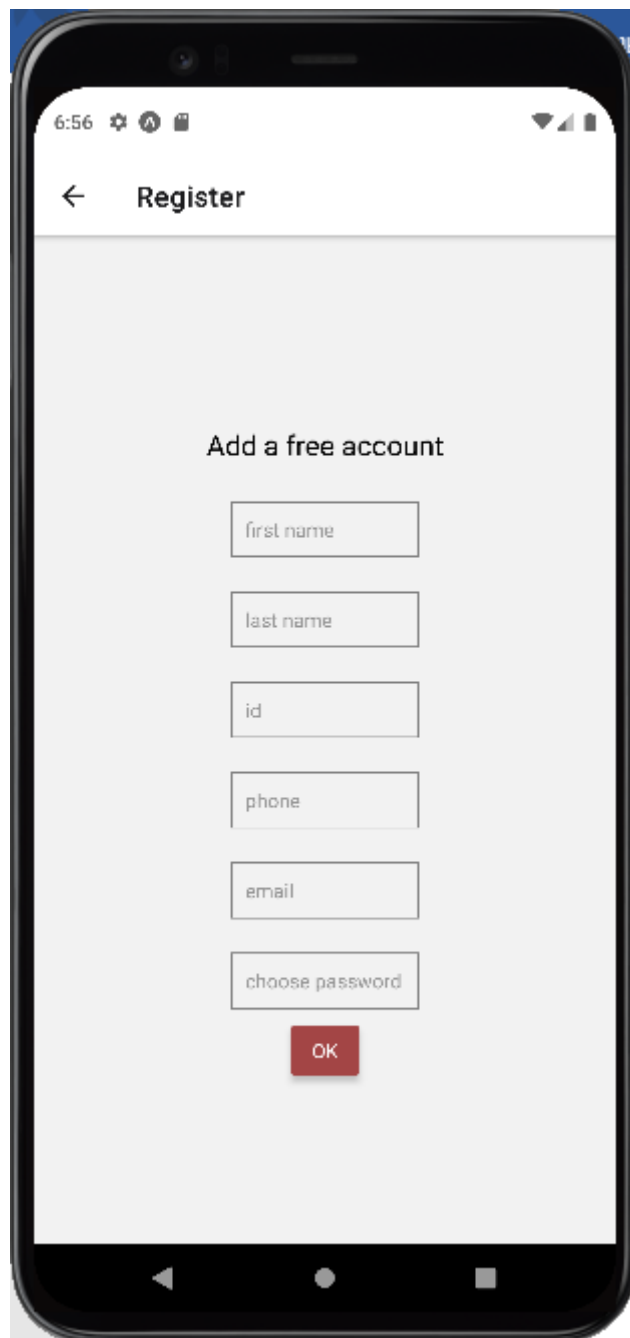
רבקה סימן טוב Pop-Up driver



על המשתמש להזין את שם המשתמש והסיסמא ובכך ליצור חיבור מאובטח לאזורו האישי.

מסך רישום משתמש חדש למערכת:

מסך זה משמש להוספת משתמש חדש למערכת.



The image shows a smartphone screen with a 'Register' app. At the top, there is a back arrow and the word 'Register'. Below this is a title 'Add a free account'. The form consists of six text input fields stacked vertically: 'first name', 'last name', 'id', 'phone', 'email', and 'choose password'. At the bottom of the form is a red button with the text 'OK'. The phone's status bar at the top shows the time 6:56 and various icons. The bottom of the phone shows the standard Android navigation bar.

על המשתמש להזין את נתוניו וללחוץ על הכפתור OK.
המשתמש החדש יוסף למערכת אם כל הנתונים שהזין נכונים ואם לא קיים שם משתמש וסיסמא זהים במערכת.

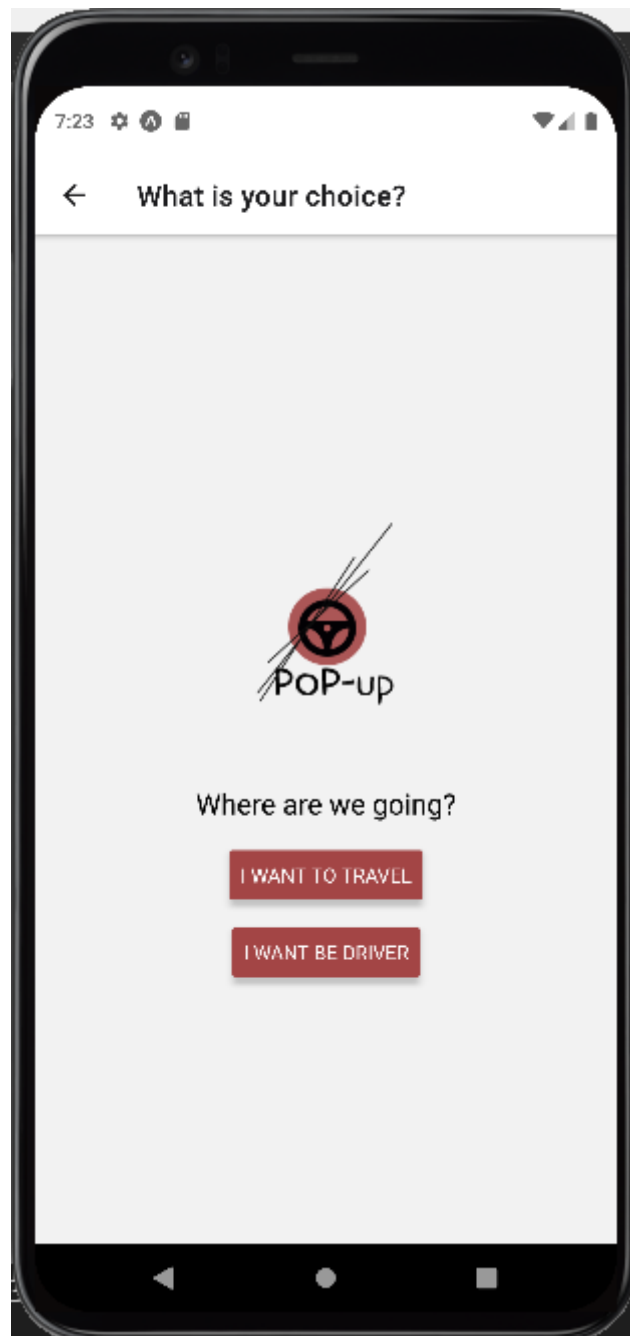
מסך ניווט:

מסך זה משמש כ

רבקה סימן טוב Pop-Up driver

נווט בין שתי האפשרויות שהאפליקציה מציעה:

נוסע / נהג



זוהי בעצם נקודת ההתפצלות של הלקוח:

האם ישמש כנהג או כנוסע.

מסך נוסע:

רביקה סימן טוב Pop-Up driver

מסך זה משמש להזנת נתונים לבקשת נסיעה חדשה.

7:01

← Passenger

starting point:

00

NEXT

destination point:

00

your time

SEND REQUEST

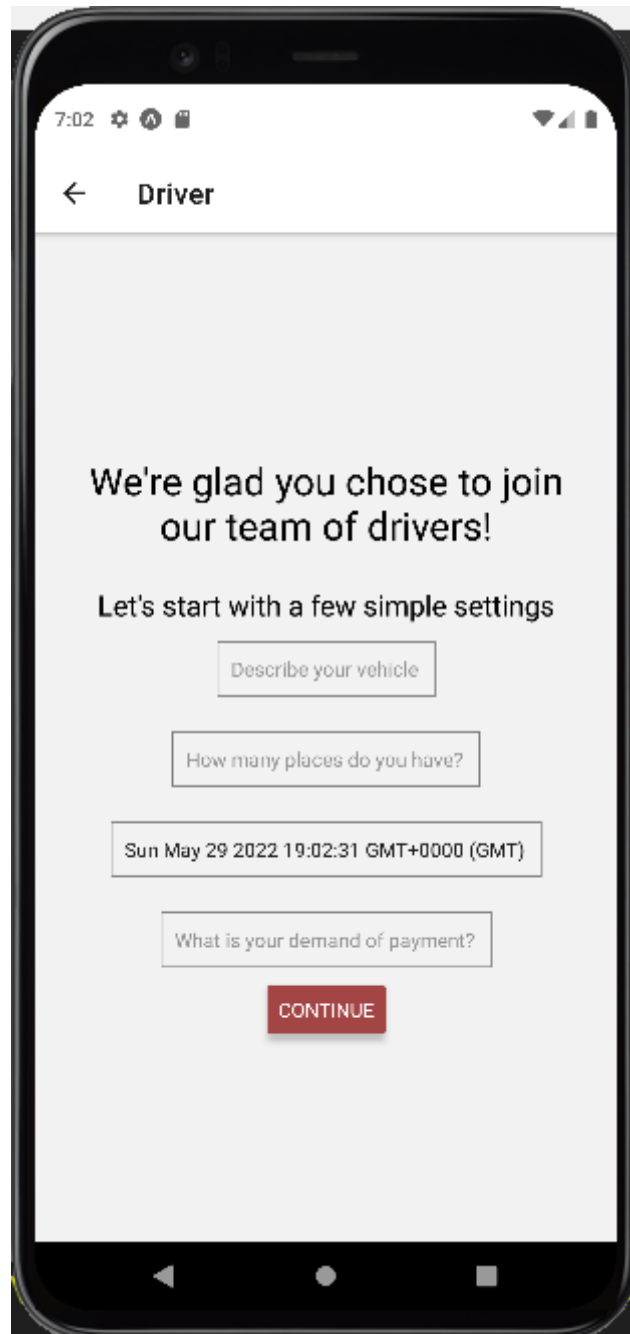
הסבר :

הנוסע לוחץ לחיצה ארוכה על המפה, במיקום הרצוי.
הלחיצה הראשונה היא נקודת המוצא שלו.
לאחר מכן עליו ללחוץ על הכפתור NEXT.
הלחיצה הבאה על המפה תזין נתונים לנקודת היעד.
לאחר מכן על הנוסע להזין את הזמן הרצוי לנסיעה.

ברירת המחדל- עכשיו.

מסך רישום נהג:

מסך זה משמש להוספת נהג חדש למערכת.



7:02

← Driver

We're glad you chose to join our team of drivers!

Let's start with a few simple settings

Describe your vehicle

How many places do you have?

Sun May 29 2022 19:02:31 GMT+0000 (GMT)

What is your demand of payment?

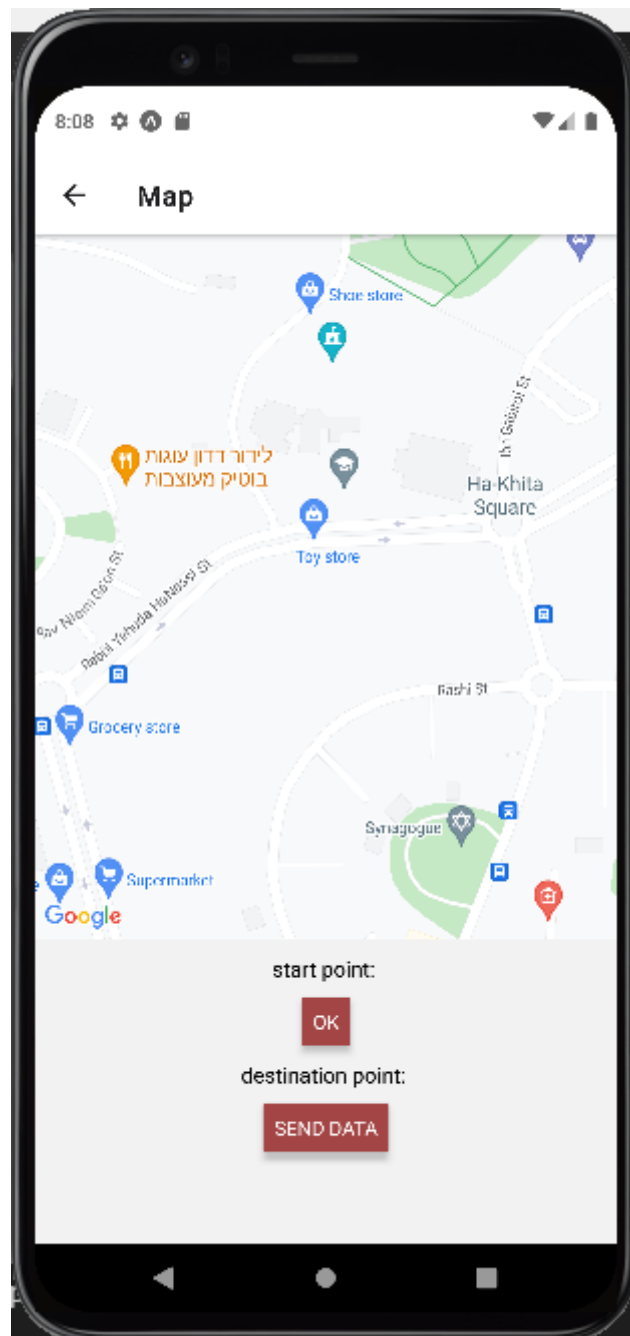
CONTINUE

המשתמש מזין את הפרטים המבוקשים.

לאחר מכן עליו ללחוץ על הכפתור CONTINUE.

המשך מסך נהג:

מסך זה משמש ללקיחת נתוני מיקום באמצעות המפה.



הנהג לוחץ לחיצה ארוכה על המפה, במיקום הרצוי.

הלחיצה הראשונה היא נקודת המוצא שלו.

לאחר מכן עליו ללחוץ על הכפתור OK.

הלחיצה הבאה על המפה תזין נתונים לנקודת היעד.

לאחר מכן עליו ללחוץ על הכפתור SEND DATA בכדי להוסיף את הנתונים למערכת.

לאחר הוספת הנהג, מסך זה ישמש להצגת המסלול על המפה, ולעצירת הנסיעה.

קוד התוכנית - האלגוריתם המרכזי

הסבר מילולי:

האלגוריתם עוסק בבעיית מציאת הדרך הקצרה עבור המסלול אותו יצטרך הנהג לעבור בכדי לאסוף את כל הנוסעים שלו, שממוקמים בנקודות שונות.

בעת שנוסע מוסיף בקשה לנסיעה, האלגוריתם מחפש מיהו הנהג הטוב ביותר מהבחינות של מיקום וזמן הגעה, ולאחר מכן מוסיפה אותו למסלול של הנהג.

בכדי לחסוך עבודה ולקצר את המסלול של הנהג, במקום שהנהג יאלץ לעבור דרך כל הנוסעים בצורה מקובעת ולא יעילה, האלגוריתם יחשב עבורו את המסלול הקצר ביותר.

חישוב המסלול נעשה באמצעות אלגוריתם דייקסטרה חמדני.

האלגוריתם עובד בצורה כזו:

האלגוריתם שולף את רשימת התחנות של הנהג.

המטרה שלנו היא להוסיף למסלול עוד שתי תחנות: תחנת איסוף נוסע, ותחנת ההורדה שלו.

ולכן, האלגוריתם מוסיף את תחנת המוצא ואת תחנת היעד בהתאמה, לסוף הרשימה.

לאחר מכן האלגוריתם פועל לתקן את הרשימה כך שבסופו של דבר, הסדר שיתקבל יהיה האופטימלי ביותר מבחינת המרחקים בין התחנות.

האלגוריתם עובר על כל רשימת התחנות ומחפש מי התחנה הקרובה ביותר למיקומו הנוכחי של הנהג.

התחנה שנמצאה היא התחנה הבאה במסלול המתוכנן.

מציאת המרחקים נעשה בעזרת הפונקציה GetDistance שמתממשקת ל google Maps.

הפונקציה מקבלת את שני המיקומים שרוצים לדעת את המרחק ביניהם ושולחת בקשה בצירוף api key מ- Google Cloud Platform (<https://console.cloud.google.com/>)

הפונקציה מחזירה את המסלול המתוכנן- רשימת התחנות.

לאחר ריצת האלגוריתם, מוחזרת התוצאה לצד הלקוח – למסך הנהג, בו הוא מתעדכן בשינוי.

הפונקציה DistanceURL()

הפונקציה מקבלת שני פרמטרים מטיפוס מחרוזת המייצגים נקודת מוצא ונקודת יעד.

הפונקציה בונה את הניתוב ל google maps api בעזרת הפרמטרים שהתקבלו, ומחזירה את הניתוב כמשתנה מטיפוס מחרוזת.

רבקה סימן טוב Pop-Up driver

```
//build the specific url of the distance request
public string DistanceURL(string origin, string dest)
{
    string API_KEY = "AIzaSyBmGUxqJwr..._gwAeu-5Nso";
    string URL = "https://maps.googleapis.com/maps/api/distancematrix/json?origins=" + origin +
        "&destinations=" + dest +
        "&units=imperial&mode=driving" +
        "&key=" + API_KEY;
    return URL;
}
```

הפונקציה GetDistance()

פונקציה אסינכרונית הפועלת למצוא מרחק בין שתי קואורדינטות.

הפונקציה מקבלת שני פרמטרים מטיפוס מיקום, ומחזירה את המרחק ביניהם באמצעות שליחת בקשת http ל google distance matrix api.

```
//get the distance between two coordinats from google matrix
public async Task<string> GetDistance(LocationClass l1, LocationClass l2)
{
    HttpClient http = new HttpClient();

    string strL1 = l1.PointLatitudeLocationX + "" + l1.PointLongitudeLocationY;
    string strL2 = l2.PointLatitudeLocationX + "" + l2.PointLongitudeLocationY;

    var responseDistance = await http.GetAsync(DistanceURL(strL1, strL2));

    if (responseDistance.IsSuccessStatusCode)
    {
        var myResult = responseDistance.Content.ReadAsStringAsync();
        jsonFormat jsonResult= JsonConvert.DeserializeObject<jsonFormat>(myResult.Result);
        return jsonResult.rows[0].elements[0].distance.text;
    }

    //if not success:
    return "";
}
```

הפונקציה GetTheBestDriverAndInvokeDijkstra()

הפונקציה מקבלת אובייקט מטיפוס נוסע, מוצאת עבורו את הנהג המיטבי- מבחינת יעילות מרחק וזמן, ולאחר מכן מפעילה את הפונקציה BuildDijkstraPath(), שבונה את המסלול לנהג שנמצא, אודותיה יפורט בהמשך.

הפונקציה מחזירה את הנהג ששויד לנוסע, ואת המסלול המתוקן שלו.

רבקה סימן טוב Pop-Up driver

```
//get the best driver to the current passenger
//add the passenger and fix the router
public async Task<(DriverClass, List<StationClass>>> GetTheBestDriverAndInvokeDijkstra(PassengerClass passenger)
{
    LocationClass sourcePassenger = passenger.GetLocation((int)passenger.startingPointLocationX);
    List<DriverClass> drivers = DriverClass.GetList();
    List<DriverClass> RelevantDriversList = drivers; ;//the list of the relevant drivers
    DriverClass minDriver = new DriverClass();
    List<StationClass> TheWinPath = new List<StationClass>();
    double minDistance = 0;
    foreach (var item in drivers)
    {
        // get the distance from the starting point to the passenger
        double DisStarting = await GetDistance(sourcePassenger, item.GetLocation((int)item.startingPoint));

        //if this distance not in the range-we remove it from the RelevantDriversList
        if (DisStarting > (passenger.TheDesiredTime.Value.Hour*60*60+ passenger.TheDesiredTime.Value.Minute*60+ passenger.TheDesiredTime.Value.Second));
            RelevantDriversList.Remove(item);

        //if the driver not active- remove from the RelevantDriversList
        if (!(bool)item.isActive)
            RelevantDriversList.Remove(item);
    }

    //find the driver that the change in him router is the lower
    foreach (var item in RelevantDriversList)
    {
        LocationClass driverLocation = item.GetCurrentLocation();
        List<StationClass> DijkstraPath = new List<StationClass>();

        double sumTimes = 0;
        DijkstraPath = BuildDijkstraPath(item, passenger);
        for (int i = 0; i < DijkstraPath.Count()-1; i++)
            //get the distance between all the stations in the router and sum them
            sumTimes += await GetDistance(DijkstraPath[i].GetLocation(), DijkstraPath[i + 1].GetLocation());

        if (sumTimes < minDistance)
        {
            minDistance = sumTimes;
            minDriver = item;
            TheWinPath = DijkstraPath;
        }
    }
    return (minDriver, TheWinPath);
}
```

הפונקציה BuildDijkstraPath()

הפונקציה מקבלת אובייקט מטיפוס נוסע ואובייקט מטיפוס נהג.

הפונקציה משתמשת במבנה הנתונים sortedList בכדי לייעל את התהליך.

הפונקציה מוסיפה למסלול של הנהג את התחנות של הנוסע ומתקנת אותו כך שעדיין יהיה זה המסלול הקצר ביותר.

הפונקציה מחזירה את המסלול החדש.

רבקה סימן טוב Pop-Up driver

```
SortedList<double, StationClass> sortedDistanceStations = new SortedList<double, StationClass>();

//build the path:
public async Task<List<StationClass>> BuildDijkstraPath(DriverClass driver, PassengerClass passenger)
{
    int codeRouter = RouterClass.GetList().FirstOrDefault(x => x.codeDriver == driver.codeDriver).codeRouter;
    LocationClass driverCurrentLocation = driver.GetCurrentLocation();
    LocationClass pasStart = passenger.GetLocation((int)passenger.startingPointLocationX);
    LocationClass pasDest = passenger.GetLocation((int)passenger.destinationPointLocationX);
    StationClass s1 = StationClass.GetList().FirstOrDefault(x => x.LocationStation == pasStart.codeCoordinate);
    StationClass s2 = StationClass.GetList().FirstOrDefault(x => x.LocationStation == pasDest.codeCoordinate);
    List<StationClass> path = new List<StationClass>();
    List<StationClass> sourcePath = driver.GetMyRouter();
    sourcePath.Add(s1);
    sourcePath.Add(s2);

    while (sourcePath != null)
    {
        var nextNode = await CalcNextStation(sourcePath, driverCurrentLocation);
        path.Add(nextNode);
        sourcePath.Remove(nextNode);
    }

    //add the new stations to the database:
    StationOfRouterClass o = new StationOfRouterClass();
    o.codeRouter = codeRouter;
    o.codeStation = s1.codeStation;
    StationOfRouterClass.InsertNewItem(o);

    StationOfRouterClass oo = new StationOfRouterClass();
    oo.codeRouter = codeRouter;
    oo.codeStation = s1.codeStation;
    StationOfRouterClass.InsertNewItem(oo);

    return path;
}
```

הפונקציה CalcNextStation()

הפונקציה מקבלת רשימת תחנות ומיקום.

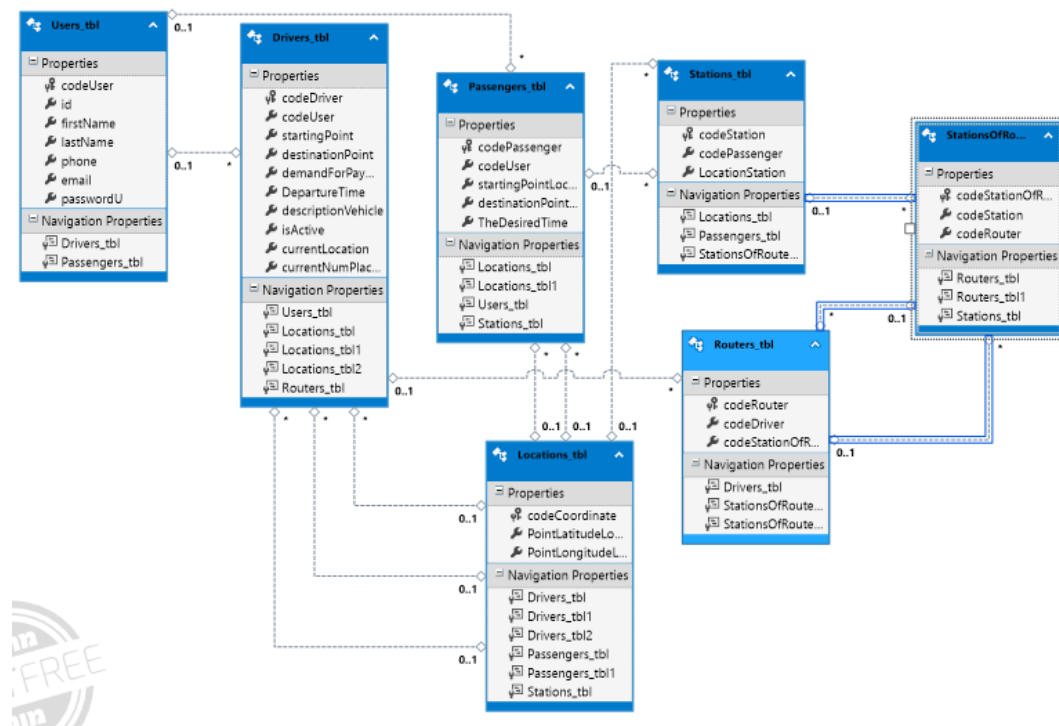
הפונקציה מחשבת מי התחנה הבאה במסלול.

הפונקציה מחזירה את התחנה.

```
public async Task<StationClass> CalcNextStation(List<StationClass> path, LocationClass dest)
{
    foreach (var item in path)
    {
        sortedDistanceStations[await GetDistance(dest, item.GetLocation())] = item;
    }
    return sortedDistanceStations.First().Value;
}
```

רביקה סימן טוב Pop-Up driver

תיאור מסד הנתונים



פירוט הטבלאות בDB:

Users_tbl

מכילה את פרטי המשתמשים.

פרטי המשתמש נקלטים למערכת בעת כניסת משתמש חדש.

CodeUser - מיספור רץ.

שם השדה	טיפוס השדה	תאור	Nullable?	מפתח
CodeUser	int	קוד מזהה		PK
FirstName	string	שם פרטי		
LastName	string	שם משפחה		
PasswordU	string	סיסמא		
Email	string	מייל		
Phone	string	טלפון		
id	string	מ.ז		

רבקה סימן טוב Pop-Up driver

Passengers tbl:

מכילה את פרטי הנוסעים.

פרטי הנוסע נקלטים בעת רישום נוסע חדש למערכת.

CodePassenger - מיספור רץ.

שם השדה	טיפוס השדה	תאור	Nullable?	מפתח
CodePassenger	int	קוד מזהה		PK
codeUser	int	משתמש		FK
StartingPoint	int	נקודת מוצא		FK
DestinationPoint	int	נקודת יעד		FK
TheDesireTime	time	זמן היציאה		

Driver tbl:

מכילה את פרטי הנהגים.

פרטי הנהג נקלטים למערכת בעת כניסת נהג חדש

CodeDriver - מיספור רץ.

שם השדה	טיפוס השדה	תאור	Nullable?	מפתח
CodeDriver	int	קוד מזהה		PK
codeUser	int	משתמש		FK
startingPoint	int	נקודת מוצא		FK
destinationPoint	int	נקודת יעד		FK
demandForPayment	float	דרישת תשלום		
DepartureTime	time	זמן משוער		
descriptionVehicle	string	תיאור הרכב		
isActive	bool	האם הנסיעה פעילה		
currentLocation	int	מיקום נוכחי		FK
currentNumPlacesInVehicle	int	מס מקומות פנויים		

Location tbl

רבקה סימן טוב Pop-Up driver

מכילה את אובייקט המיקום המורכב משתי נקודות.

שדה של מיקום נוסף בעת הוספת נסיעה, נהג ונוסע.

codeCoordinate - מיספור רץ.

שם השדה	טיפוס השדה	תאור	Nullable?	מפתח
codeCoordinate	int	קוד מזהה		PK
PointLatitudeLocationX	float	קו הרוחב		
PointLongitudeLocationY	float	קו האורך		

Stations_tbl:

מכילה את פרטי התחנות.

פרטי התחנה נקלטים למערכת הוספת תחנה חדשה למסלול.

codeStation - מיספור רץ.

שם השדה	טיפוס השדה	תאור	Nullable?	מפתח
odeStation	int	קוד מזהה		PK
codePassenger	int	קוד נוסע		FK
LocationStation	int	מיקום התחנה		FK

Routers_tbl:

מכילה את פרטי המסלולים.

מסלול מכיל את כל התחנות השייכות לו.

codeRouter - מיספור רץ.

שם השדה	טיפוס השדה	תאור	Nullable?	מפתח
codeRouter	int	קוד מזהה		PK
codeDriver	int	קוד נהג		FK

רבקה סימן טוב Pop-Up driver

codeStationOfRouter	int	קוד תחנה בטבלת תחנות למסלול		FK
---------------------	-----	-----------------------------------	--	----

StationOfRouter_tbl:

טבלת תחנות למסלול.

טבלה זו משמשת כרשימת תחנות לכל מסלול.

codeStationOfRouter - מיספור רץ.

שם השדה	טיפוס השדה	תאור	Nullable?	מפתח
codeStationOfRouter	int	קוד מזהה		PK
codeStation	int	קוד תחנה		FK
codeRouter	int	קוד מסלול		FK

מדריך למשתמש

בעת הכניסה לאפליקציה, ישנה אפשרות למשתמש להתחבר לאזורו האישי ע"י הקשת שם משתמש וסיסמא.

במקרה שמדובר במשתמש חדש, ניתנת האפשרות להירשם למערכת.

לאחר מכן מופיע מסך שמוגדרת בו נקודת פיצול בצורה הבאה :

במקרה שהמשתמש מעוניין להוסיף בקשת נסיעה - נוסע :

עליו ללחוץ על הכפתור : "I want to travel".

לאחר מכן יופיע מסך שבו יוגדרו שדות להזנת הנתונים הבאים :

- נקודת מוצא של הנוסע.
- נקודת יעד של הנוסע.
- הזמן הרצוי לנסיעה המבוקשת. (כברירת מחדל- ערך זה מאותחל לשעה הנוכחית).

נתונים נקודת המוצא והיעד מוזנים באמצעות לחיצה ארוכה על המפה שמוצגת על המסך.

לאחר שהנוסע מזין את הנתונים לאפליקציה, עליו ללחוץ על הכפתור "send request".

הבקשה נשלחת למערכת.

המערכת פועלת לאתר את הנהג המתאים ביותר לנתונים שהוזנו.

לאחר מציאת הנהג המבוקש, המערכת מיידעת את הנוסע.

במקרה שהמשתמש מעוניין להוסיף הצעת נסיעה - נהג :

עליו ללחוץ על הכפתור : "I want to be driver".

לאחר מכן יופיע מסך שבו יוגדרו שדות להזנת הנתונים הבאים :

- הזמן שבו יצא לנסיעה. (כברירת מחדל- ערך זה מאותחל לשעה הנוכחית).
- מספר המקומות הפנויים הקיימים ברכבו.
- תיאור הרכב.
- דרישת התשלום לצירוף נוסע.

לאחר מילוי נתונים אלו, עליו ללחוץ על הכפתור "continue".

בעקבות כך, נפתח מסך נוסף להזנת שני הנתונים הבאים :

- נקודת מוצא של הנהג.
- נקודת יעד של הנהג.

נתונים אלו מוזנים באמצעות לחיצה ארוכה על המפה שמוצגת על המסך.

לאחר מכן עליו ללחוץ על הכפתור "send request".

הבקשה נשלחת למערכת.

הנהג מוסף למערכת ובעקבות כך, הנהג מקבל משוב בדמות מסלול שעליו לנסוע בעקבותיו.

רבקה סימן טוב Pop-Up driver

הכפתור "stop the travel" המופיע על המסך במהלך נסיעת הנהג מאפשר לו לעצור את האפשרות להוסיף עוד נוסעים למסלול שלו.

בדיקות והערכה

לאחר הרצת האלגוריתם נבחנו כל האילוצים ומקרי הקצה שדרושים כדי להביא להתאמה אופטימלית. כאשר הופיעו טעויות ובאגים בביצוע האלגוריתם נבדק הקוד שוב עד להטבת הקוד.

ניתוח יעילות

ניתוח הסיבוכיות:

האלגוריתם הראשי- דייקסטרה חמדני, הוא עיקר הסיבוכיות בתוכנה והיא: $O(N^2)$

כאשר N מסמן את מספר התחנות השייכות למסלול.

אבטחת מידע

המידע של המשתמשים מוגן ע"י שם משתמש וסיסמא השמורים במערכת.

מסקנות וסיכום

בזמן זה של סיום הפרויקט אני מרגישה שעברתי תהליך ענק עם עצמי.

אם נחזור לאחור, לרגעי התכנון והבניה הראשונית, ההרגשה המבוהלת מעט, הידיעה שאני עומדת מול פרויקט גדול מאד שידרוש ממני ללמוד ולרכוש מיומנויות רבות בנושאים שבכלל לא הכרתי ולא התנסיתי מעולם.

ידעתי שאני צריכה ללמוד שפה חדשה לגמרי, שאין לי בה שום מושג, ידעתי שאני אצטרך לעבוד בסביבת עבודה זרה לחלוטין ממה שהורגלתי אליה.

יחד עם כל זה, התרגשתי מהאתגר שהוצב בפניי וידעתי שאני אעשה הכל כדי להצליח.

וברוך ה', הצלחתי!

רכשתי ידע מקצועי רב ב react native ובסביבת פיתוח של אנדרואיד,

למדתי לעצב ולדבר בשפת מובייל,

למדתי להשתמש בממשק google maps ובאפשרויות המוצעות בו.

עמדתי בפני באגים משמעותיים לבד, ויכולתי להם.

כמו כן, ההתעסקות והדיון על האלגוריתם: מי היעיל ביותר מבין שאר האפשרויות, איזו דרך היא הטובה ביותר למימוש, מבנה הנתונים שאשתמש בו וכו', כל אלא שפשפו את החשיבה האלגוריתמים שלי ואת הלוגיקה התכנותית שרכשתי במהלך הלימודים.

אני מרגישה שנקודת היציאה שלי מהפרויקט היא נקודת ציון משמעותית בדרכי המקצועית, הידע והמיומנות שרכשתי לפעול לבד, להעז ללמוד דברים חדשים לגמרי, לנסות, לנסות שוב ולהצליח בעזרת ה'.

פיתוחים עתידיים

בשלב זה האפליקציה עוסקת בניהול מערך של תחבורה שיתופית אך לא מתערבת בדרכי התשלום בין הנוסע לנהג.

רבקה סימן טוב Pop-Up driver

בעתיד אני שואפת להוסיף מערכת לניהול תשלום באמצעות אשראי (ע"י pay-pal).
כמו כן, להוסיף לנוסע אפשרות לבחור את מגדר הנהג הרצוי.

ביבליוגרפיה

[/https://reactnative.dev](https://reactnative.dev)

[/https://expo.dev](https://expo.dev)

[/https://developer.android.com](https://developer.android.com)

[/https://stackoverflow.com](https://stackoverflow.com)

[/https://github.com](https://github.com)

[/https://google.com](https://google.com)

[/https://youtube.com](https://youtube.com)

<https://developers.google.com/maps/documentation/distance-matrix/distance-matrix>