**טכנולוגיות אינטרנט מתקדמות – 61776 (WEB)**

הגשת פרויקט

TouRingo – Group 5

|  |  |
| --- | --- |
| **שם חברי הצוות** | **תז** |
| שלומי פרדימן | 318187002 |
| שחר ברנסון | 208608414 |
| עומר גולדשטיין | 205906258 |
| נועם ענתבי | 315033035 |

הסבר קצר על האתר

TouRingo הוא אתר תיירות, המציג למשתמש מגוון רחב של אירועים, מוזיאונים, מסעדות בערים שונות, ומאפשר לו לקבל מידע על האירועים, וכיווני דרך. בנוסף לכך מתאפשר למשתמש גם לבצע Booking עבור אירועים לבחירתו, לתת ביקורות עבור אירועים אשר היה בהן, ולהוסיף גם אירועים חדשים משלו לאתר.

בפרויקט נעשה שימוש בטכנולוגיות הבאות

* Next JS עבור הצד לקוח והצד שרת.
* React JS עבור הקומפוננטות בצד לקוח.
* Tailwind עבור העיצוב של האתר.
* MongoDB Atlas עבור הבסיס נתונים.
* Vercel עבור ה-deployment של האתר.
* Github עבור אחסון הקוד, ובקרת גרסאות.

קישורים

* [קישור לתיקיית ב-Github](https://github.com/ShlomiFridman/WebProject2025)
* [קישור לאתר ב-Vercel](https://touringo.vercel.app/)

תוכן

[סעיף 1 – משימות חברי הצוות 3](#_Toc189335755)

[סעיף 2 – דרישות פונקציונליות ולא פונקציונליות 3](#_Toc189335756)

[סעיף 3 - מבנה ואינטראקציה 4](#_Toc189335757)

[סעיף 4 – מבנה האתר \ התיקיות 5](#_Toc189335758)

[סעיף 5 - תיק למתכנת 7](#_Toc189335759)

[פונקציות מרכזיות 7](#_Toc189335760)

[API שנעשה בהם שימוש 11](#_Toc189335761)

[ספריות חיצוניות 11](#_Toc189335762)

[פירוט על הבסיס נתונים 12](#_Toc189335763)

[סביבת עבודה 13](#_Toc189335764)

[שימוש בכלי AI 13](#_Toc189335765)

[מקורות 13](#_Toc189335768)

[סעיף 6 - תיק משתמש 14](#_Toc189335769)

סעיף 1 – משימות חברי הצוות

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| שלומי פרדימן (מהנדס מערכת) | Database | Database |
| שחר ברנסון | Server API | Server API |
| עומר גולדשטיין | Pages | Pages |
| נועם ענתבי | Components | Components |

סעיף 2 – דרישות פונקציונליות ולא פונקציונליות

טבלת דרישות FR:

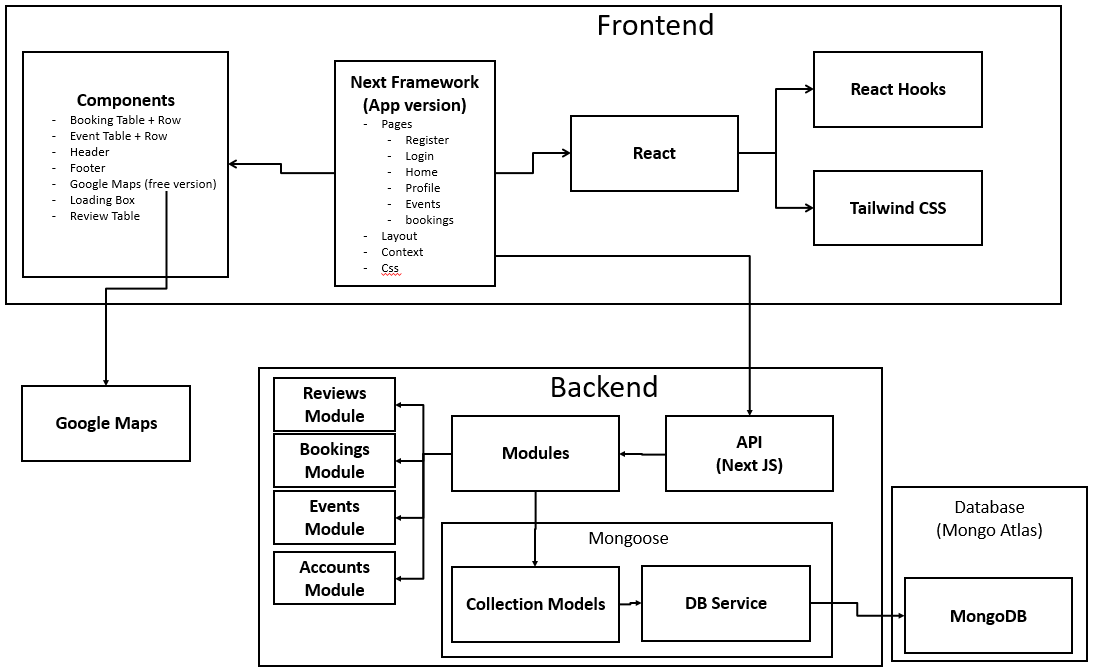
|  |  |  |
| --- | --- | --- |
| **Type** | **Requirement Description** | **No.** |
| FR | The system allows users to log in. | 1 |
| FR | The system shows a display of attractions. | 2 |
| FR | The system allows users to enter reviews for attractions. | 3 |
| FR | The system provides navigation features to attractions. | 4 |
| FR | The system allows users to book events at attractions. | 5 |

טבלת דרישות NFR-ים:

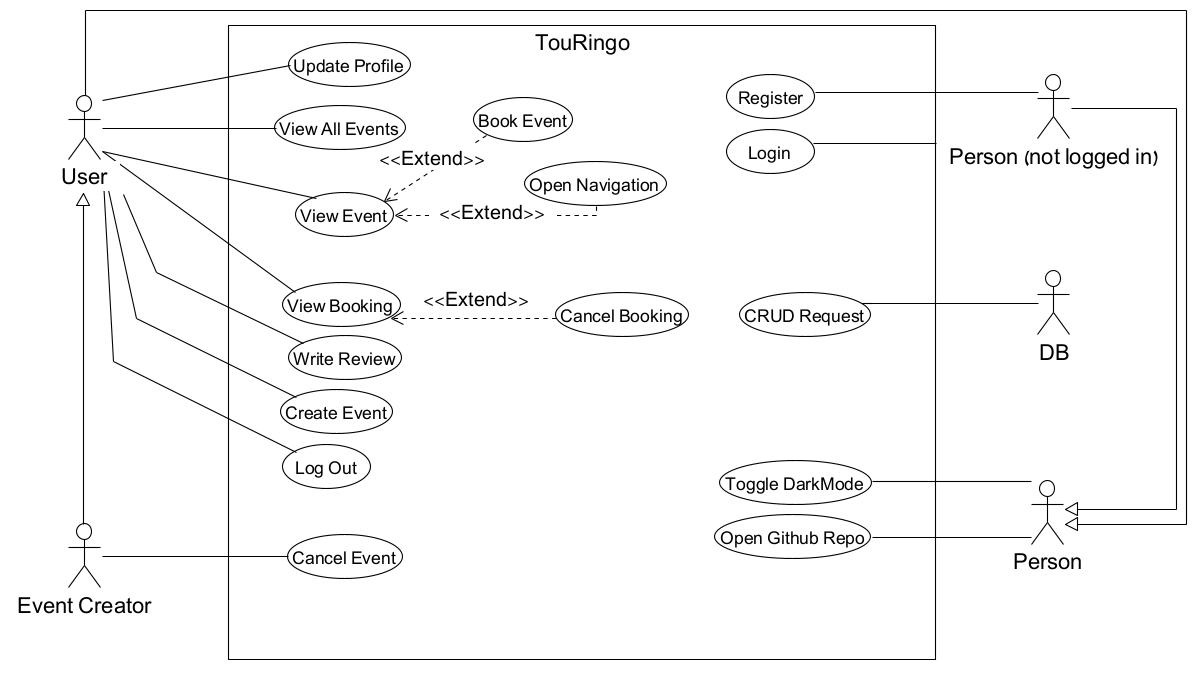
|  |  |  |
| --- | --- | --- |
| **NFR Type** | **Requirement Description** | **No.** |
| Performance | Each page should load in less than 5 seconds | 1 |
| Deployment | The product must be deployed on Vercel | 2 |
| Data integrity | The product needs to only accept review scores in range [1,5] | 3 |
| Usability | The product must have responsive design | 4 |
| Interoperability | The product needs to be compatible with outside an outside navigation service (e.g. Google Maps) | 5 |
| Usability | The product support light\dark mode | 6 |

סעיף 3 - מבנה ואינטראקציה

א. תרשים ארכיטקטורה של האתר:

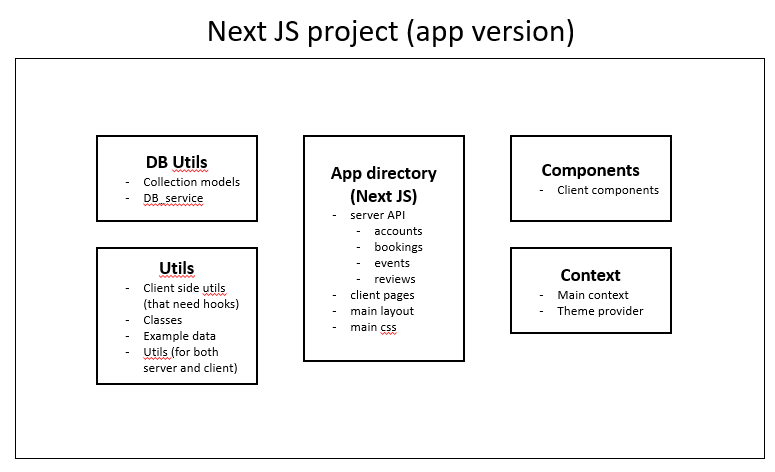


ב. דיאגרמת use case:



סעיף 4 – מבנה האתר \ התיקיות

1. דיאגרמה המתארת את התיקיות והקבצים השונים במערכת:



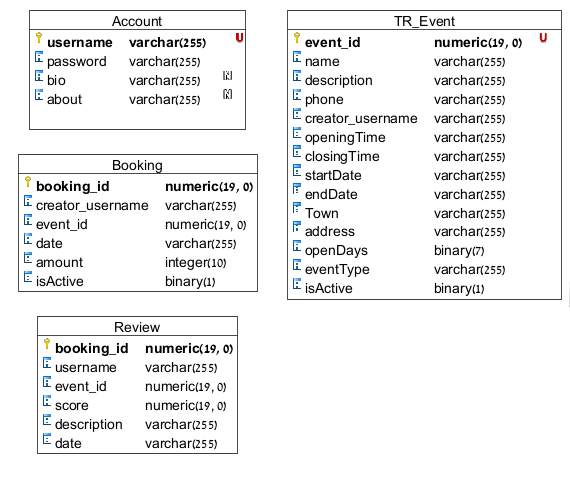
פירוט:

* App Directory: התיקייה שמכילה את קבצי דפי הצד לקוח (כגון עמוד ראשי, פרופיל וכו') ואת קבצי ה-API של הצד שרת והמודלים עבור הלוגיקה של הצד שרת (כגון של טיפול בבקשות עבור Accounts, Events וכו').
* DB Utils: התיקייה שמכילה את ה-DB-Connection (ב-DB\_Service) ואת המודלים של הבסיס נתונים (ב-Collection models) הממומשים ב-Mongoose.
* Utils: מכיל שירותי util שנעשה בהן שימוש במערכת, כגון המחלקות, פונקציות util, וכו'.
* Components: מכיל קבצי React-component שנעשה בהם שימוש בקבצי הצד לקוח, כגון EventTable, EventRow, כפתורי ה-bookings, כפתורי ה-events וכו'.
  + הם שמורים בתיקייה משלהם על מנת שלא יהיה גישה ישירה אליהם ב-routing.
* Context: מכיל את ה-ThemeProvider וה-MainContext של הצד לקוח.

ב. פרטי המידע ודיאגרמת הבסיס נתונים:

**לא מצאנו** API **שהתאים למערכת שלנו, לכן יצרנו** FakeData **שנשמר ב-**DB **מייצג את המידע האמיתי.**

**המבנה של הנתונים ב-**MongoDB **הוא באופן הבא:**



* + היות והשתמשנו ב-NoSQL, אין קשרים ולכן לצורך נוחות לא נרמלנו את הטבלאות.

סעיף 5 - תיק למתכנת

פונקציות מרכזיות

* חיבור לבסיס נתונים: החיבור ל-MongoDB נעשה בעזרת חבילת Mongoose בתבנית עיצוב של Singleton. הקוד נמצא בקובץ src/db\_utils/dbService והחיבור ממומש במחלקת DB\_Service. הגישה לחיבור ניתנת רק על ידי הפונקציה הסטטית getInstance()

דוגמא לשימוש:

const mongoDbInstance = DB\_Service.getInstance();

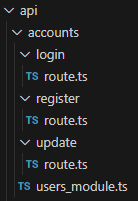
const db = await mongoDbInstance.connect();

* פירוט על המבנה של ה-API של השרת: כל נושא בפרויקט (כגון אירועים, משתמשים, ביקורות וכו') נמצא בתיקייה משלו על מנת שניתוב ה- Next js יהיה יותר נוח.

כל פעולת טיפול בבקשה נמצאת בתת-תיקייה משלה, והפונקציות של הפעולות מול ה-DB נמצאות בקובץ ה-module של אותו נושא.

בחרנו לעבוד כך על מנת שיהיה קוד יותר "נקי" בקובץ של הבקשות, וכל הפונקציות שניגשות לשרת עבור אותו נושא יהיו באותו קובץ.

דוגמא: עבור נושא Accounts יש את התיקיות:



בכל קובץ route.ts יש את הטיפול עבור אותה בקשה (כגון ב-route.ts של register יש טיפול בבקשת ה-POST של משתמש חדש) וב-users\_module.ts יש את הטיפול של הבקשה מול ה-DB.

* אתחול בסיס הנתונים: ניתן לאתחל את בסיס הנתונים על ידי בקשת GET ל-URL:

/api/dev/createExamples

ה-collections יבנו ב-DB עם הנתונים שנמצאים ב-examplesData.

* בנוגע לבקשות מהשרת:
  + במידע ובקשה לא התקבלה מסיבה כלשהי יוחזר למשתמש קוד
    - 4XX אם הסיבה היא בגלל הלקוח (מידע לא תקני או חסר)
    - 5XX אם הסיבה היא פנימית (שגיאה בצד שרת או בבסיס נתונים)

ובנוסף תהיה הודעת שגיאה ב-body.message.

במידה והבקשה תקינה תוחזר תשובה עם קוד 2XX ואם אמור לחזור תוכן כלשהו הוא יחזור ב-body.result.

* + עבור בקשות מסוג GET ו-PATCH השרת מצפה לקבל את הפרמטרים ב-URL כך שהם מופרדים על ידי '/'.

דוגמא: עבור בקשה של קבלת ה-bookings של משתמש /api/bookings/get/[username] הפרמטר הוא username.

* + עבור בקשות מסוג PUT ו-POST השרת מצפה לקבל את הפרמטרים של הבקשה דרך ה-body בשדה data שיחיל מחרוזת שמייצגת JSON שהוצפן על ידי פונקציית encryptData() הנמצאת ב-utils.

דוגמא: בקשת update של accounts מצפה לפרמטרים

interface UpdateBodyParam{

    username: string;

    updatedAccount: Account;

}

וניגשת אליהן בצורה הבאה:

        const reqBody = await request.json();

        const data = decryptData(reqBody.data) as UpdateBodyParam;

        const username = (!data) ? null : data.username;

        const account = (!data) ? null : data.updatedAccount;

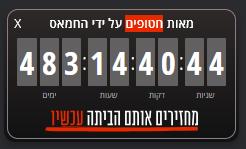
* פירוט תהליך טיפול בבקשת login בשרת על ידי שליחת בקשת PUT ל-URL

/api/accounts/login

* + השרת מצפה לקבל בקשה מסוג PUT עם הפרמטרים בצורך המתוארת מעל.
    - בחרנו לעשות את הבקשה הזו מסוג PUT בגלל סיבות אבטחת מידע.
  + במידע והפרמטרים לא תקינים יוחזר למשתמש תשובה עם קוד 400.
  + נקראת הפונקציה verifyAccount() מ-accounts\_module שבודקת האם נמצא משתמש עם אותו username ו-password.
    - במידה וכן יוחזר לבקשה תשובה עם קוד 200 ואובייקט של המשתמש ב-body.result, הסיסמה האמתית לא תהיה באובייקט אלה תוחלף ב-"#######".
    - במידה ולא קיים משתמש כזה תוחזר תשובה עם קוד 401 והודעה body.message= "Username or password incorrect"
* תהליך הצגת האירועים למשתמש:
  + בעת טעינת העמוד נשלחת בקשת GET לשרת (/api/events/getAll) ואם התשובה תקינה האירועים שהתקבלו ישמרו ב-state של events.
    - עד שהאירועים יתקבלו מהשרת מוצג למשתמש LoadingBox.
  + כל האירועים שבוטלו (isActive=false) יוסרו מהרשימה.
  + רשימת האירועים נשלחת ל-EventTable וכל אירוע מוצג ב-EventRow.
  + במידע והשתמש לחץ על אירוע, אירוע זה נשמר ב-selectedEvent על ידי קריאה ל-dispatch של ה-MainContext.
  + כאשר משתמש נכנס לעמוד של אירוע יש שני מקרים:
    - אם הוא הגיע על ידי לחיצה על EventRow המידע של האירוע יילקח מ-selectedEvent ויובאו הביקורות של אותו אירוע על ידי אובייקט זה.
    - אם הוא הגיע ידנית על ידי עריכת ה-url תשלח בקשת GET מהשרת על מנת לקבל את הנתונים של האירוע.
      * אם האירוע לא קיים המשתמש יופנה ל-404.
      * אם האירוע כן קיים תשלח בקשת GET עבור קבלת הביקורות של האירוע.
* מעבר בין עיצובי יום ולילה:
  + על ידי לחיצה על כפתור מצב יום/לילה, ניתן להחליף את העיצוב של האתר לנוחות המשתמש.
  + נשמר העיצוב ב- Local Storage של המשתמש למשתמש שחוזר לאתר.
  + הקומפוננטה Layout עטופה בתגית עיצוב על מנת לשנות ממצב יום ללילה.
  + מעבר למצב לילה מוסיף את תגית dark למסמך, מה שמעביר את כל האלמנטים במסמך למצב לילה. לחיצה נוספת מורידה את התגית למסמך.
  + בנוסף השתמשנו בתכונת dark-theme למסמך. זאת על מנת להשתמש ב – Global.css לשמירת אלמנטים מרכזיים וחוזרים למצב יום ולילה. כאפשרות נוספת לשליטה בעיצוב האלמנטים.

API שנעשה בהם שימוש

נעשה שימוש ב-API של [Bring Them Home Now](https://stories.bringthemhomenow.net/counter-he). על מנת להציג את ה-counter מאז ה-7 באוקטובר במערכת שלנו.

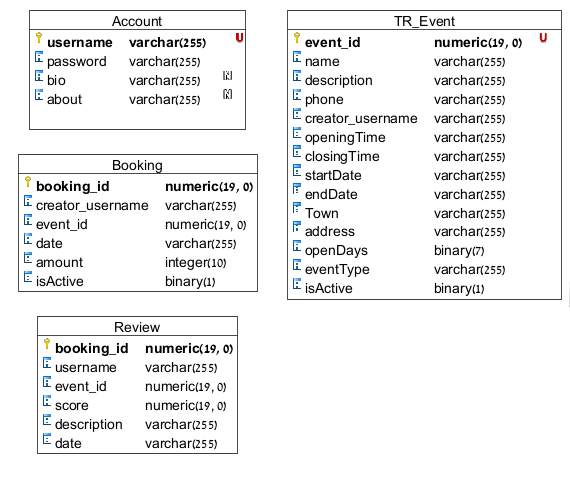


הספריות החיצוניות שנעשה בהן שימוש

* + - Next js: לבניית הפרויקט, ספציפית ב-Next js מסוג App.
    - Mongoose – כל התקשורת עם ה-MongoDB atlas נעשתה בעזרת חבילה זו, כולל בניית ה-schemes והמודלים של ה-collections.
    - crypto-js – נעשה שימוש בחבילה זו על מנת להצפין את המידע שעובר לשרת בבקשות ה-POST וה-PUT, ולפענח אותן בצד של השרת.
    - Tailwind – לצורך עיצוב הצד לקוח.
    - Eslint – לצורך זיהוי בעיות בקוד, אכיפת סגנון כתיבה עקבי, ווידוי עמידה בסטנדרטים הנפוצים בשוק (כגון אלה של גוגל).

פירוט על הבסיס נתונים

* + היות והשתמשנו ב-NoSQL, אין קשרים ולכן לצורך נוחות לא נרמלנו את הטבלאות.



* + הנתונים שהשתמשנו בהם הם נתונים שיוצרו ע"י בינה מלאכותית (ChatGPT), הכנסנו prompt המכיל את המבנה של בסיס הנתונים שלנו ומידע נוסף הרלוונטי על ישראל. ביקשנו לפחות 5 דוגמאות עבור כל טבלה כדי שנוכל למלא את התוכן. בנוסף מצאנו תמונות מגוגל בשביל האירועים.

סביבת עבודה

עבדנו עם סביבת העבודה VS-Code עם ה-extensions שהוצגו בקורס:

* HTML template
* Live Server
* Auto Rename Tag
* Lorem ipsum
* Backticks
* Multi-cursor
* Htmltagwrap
* Import cost

שימוש בכלי AI

נעזרנו בכלי ChatGPT זה על מנת ללמוד לתכנת ב- MongoDB וקבלת דוגמאות איך להשתמש בחבילה של Mongoose, הפרופטים היו:

* + - Mongoose, how to create schema and models
    - Mongoose, how to create singleton connection
    - Mongoose, given collection 'Account {username: String, password: String} and a connection, give example for simple CRUD
      * בנוסף נעזרנו לצאט ליצירת דוגמאות לבסיס הנתונים כפי שפירטנו קודם.

שלחנו ל-ChatGPT את הקוד של המחלקות, וביקשנו שיבנה לפחות 5 דוגמאות לכל מחלקה.

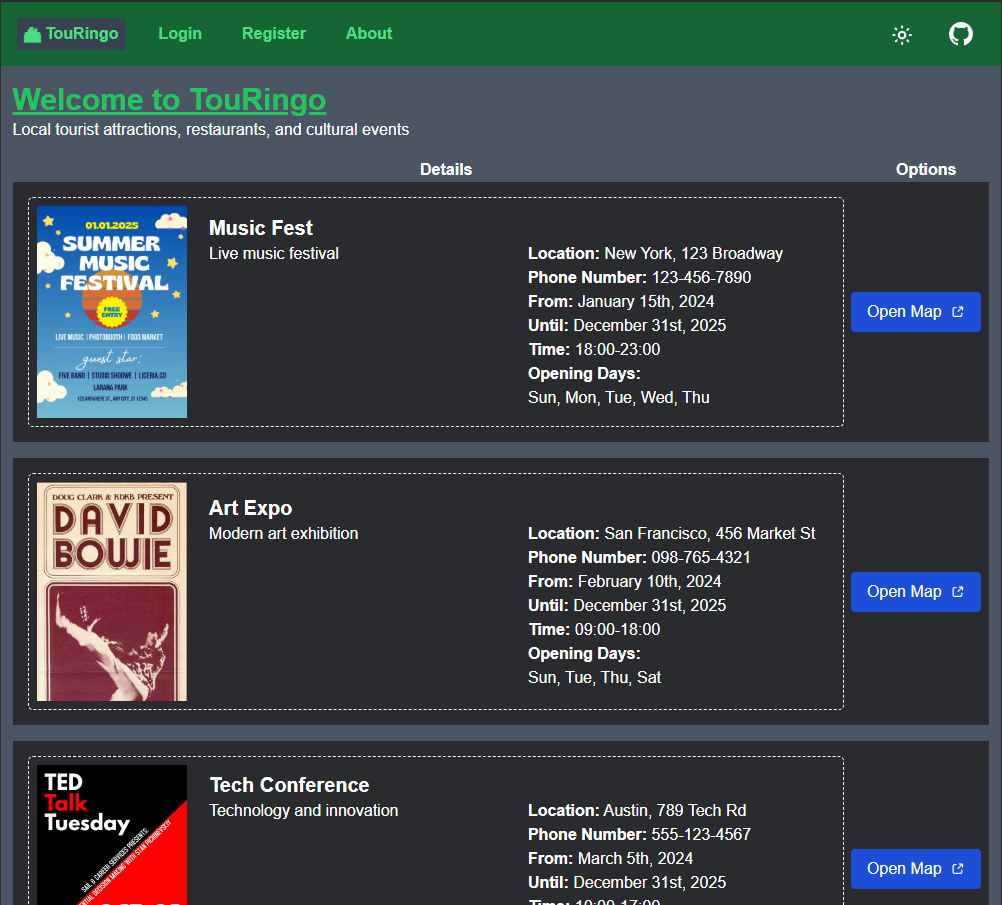
מקורות

מקורות שנעשה בהן שימוש במהלך הפיתוח:

* + - <https://mongoosejs.com/docs/>
    - <https://nextjs.org/docs>
    - <https://react.dev/>
    - <https://www.typescriptlang.org/docs/>
    - <https://tailwindcss.com/>
    - המצגות של הקורס.

סעיף 6 - תיק משתמש

צילומי מסך והסברים



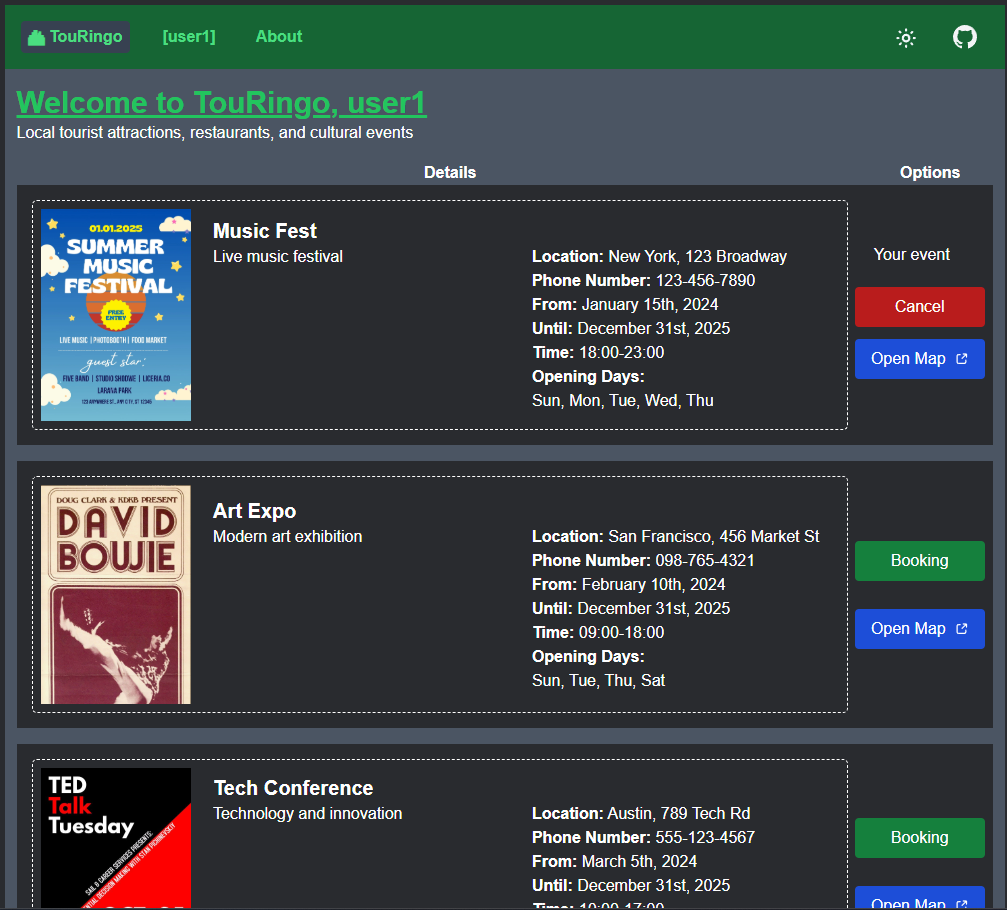
* עם הכניסה הראשונית לאתר נגיע לדף הבית, שימו לב שניתן לצפות בתוכן האתר, מידע על האטרקציות וכן גישה לגוגל מפות. בנוסף ניתן להיכנס לעמוד של כל אטרקציה. כדי לפתוח את הפונקציונאליות המלאה באתר יש להירשם ולהתחבר.

A screenshot of a login box

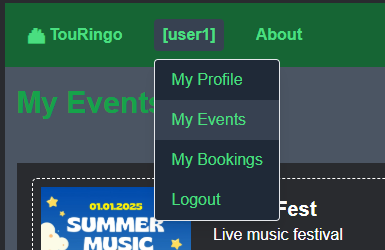
Description automatically generated A screenshot of a login box

Description automatically generated

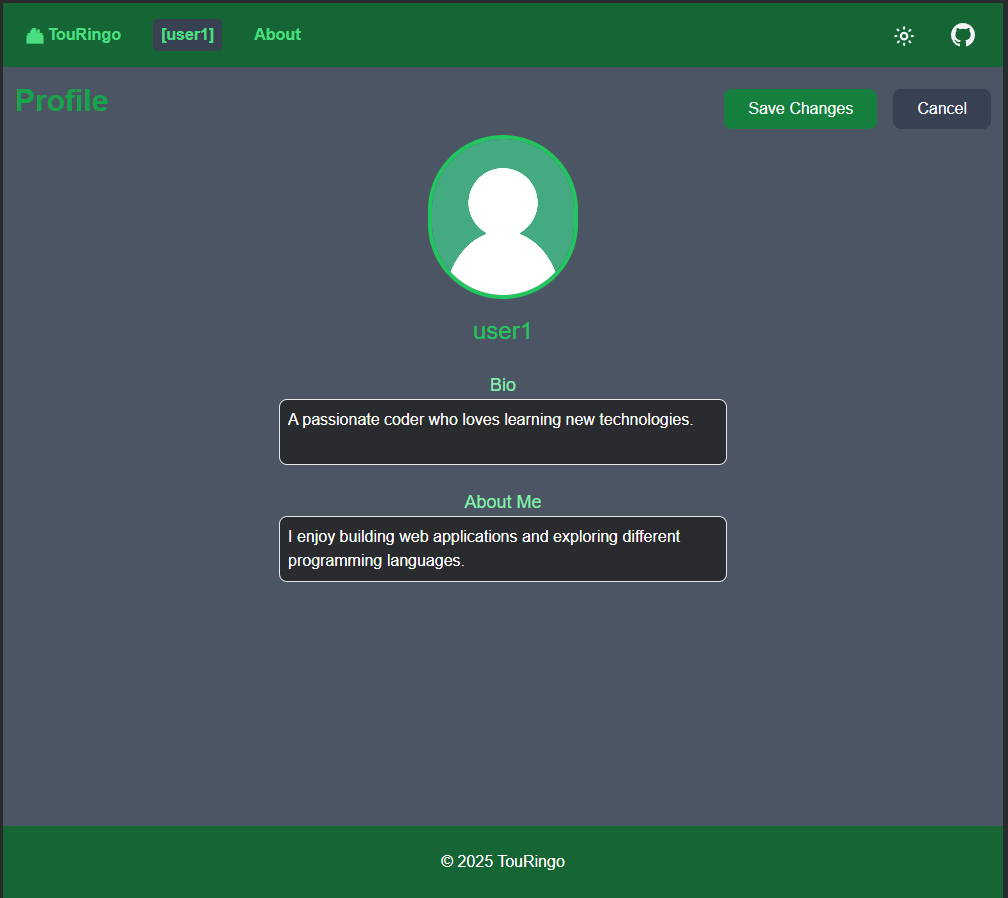
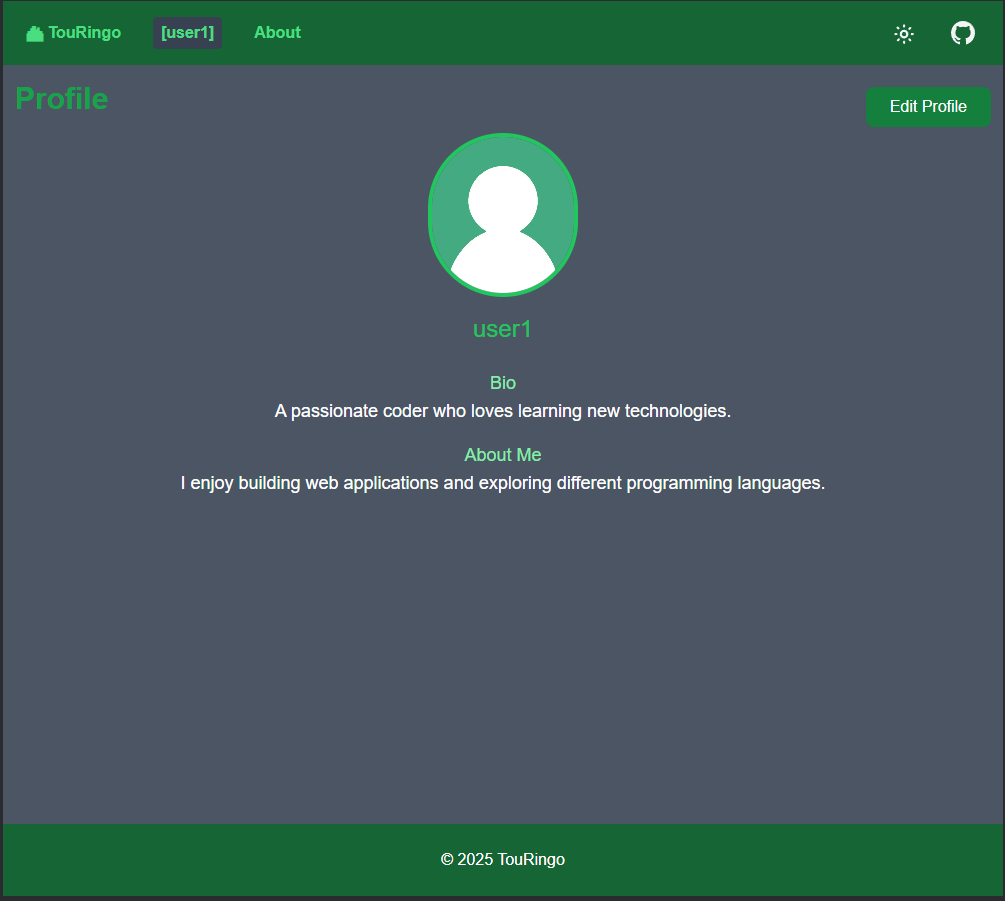
* כל משתמש מוזמן למלא את הפרטים כרצונו להתחברות ושימוש באתר.



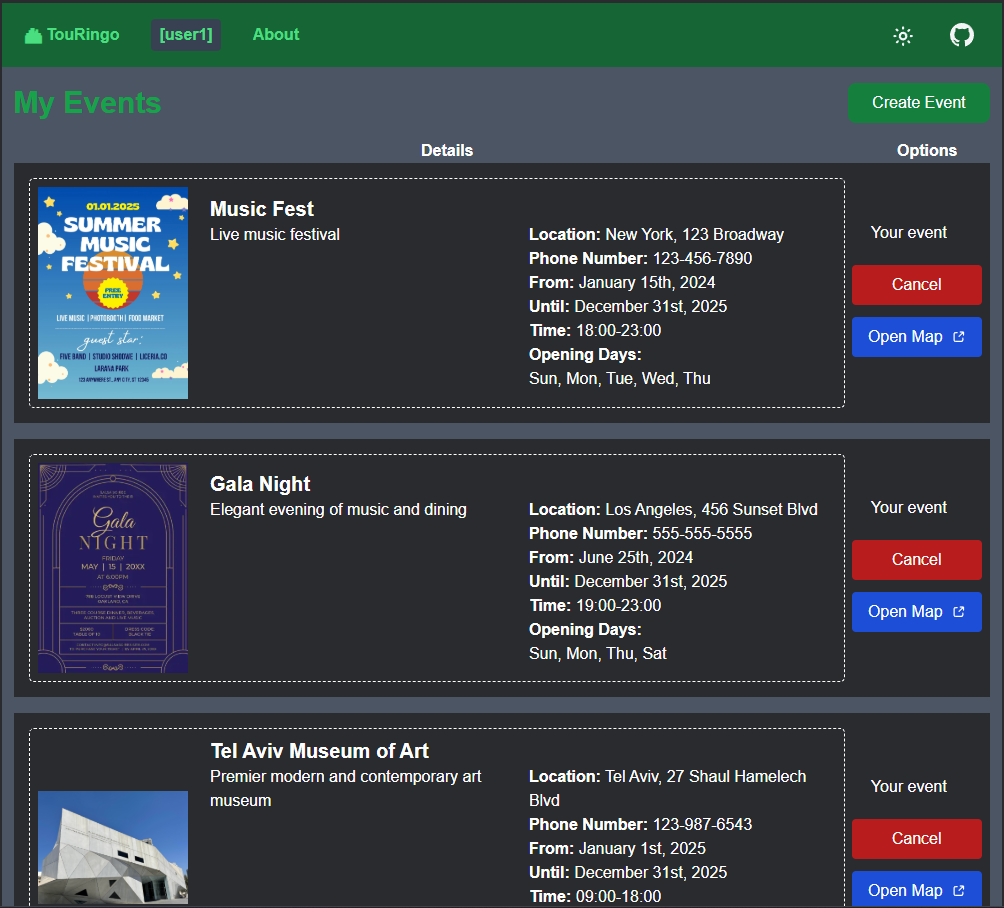
* כעת לאחר ההתחברות ניתן לראות אופציות נוספת בנוגע לאטרקציות כמו יצירת אירוע וביטולה או ביצוע "Booking".
* עבור ניווטים נוספים ניתן ללחוץ על שם המשתמש שלכם ויפתח תפריט נוסף.



* האופציה הראשונה היא צפייה בפרופיל שלנו.
* השנייה היא צפייה באירועים שהמשתמש יצר.
* השלישית היא צפייה ב-booking שהמשתמש עשה.

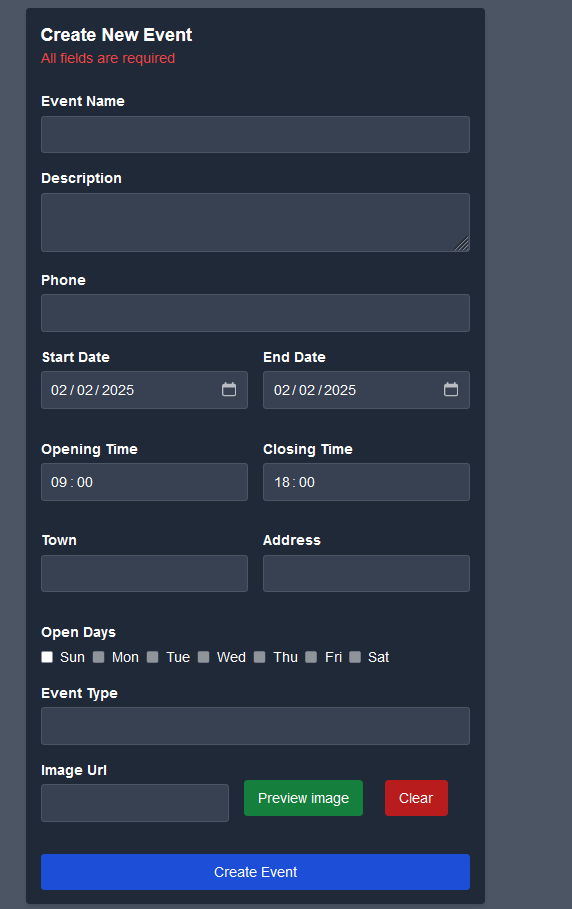


* יש את האופציה לעריכת שדות ה- Bio וה- About Meע"י לחיצה על כפתור "Edit Profile" למעלה בצד ימין, ניתן לשמור את השינויים שביצעתם ע"י לחיצה על הכפתור המתאים.

הבחירה הבאה היא עמוד האירועים שלנו, 

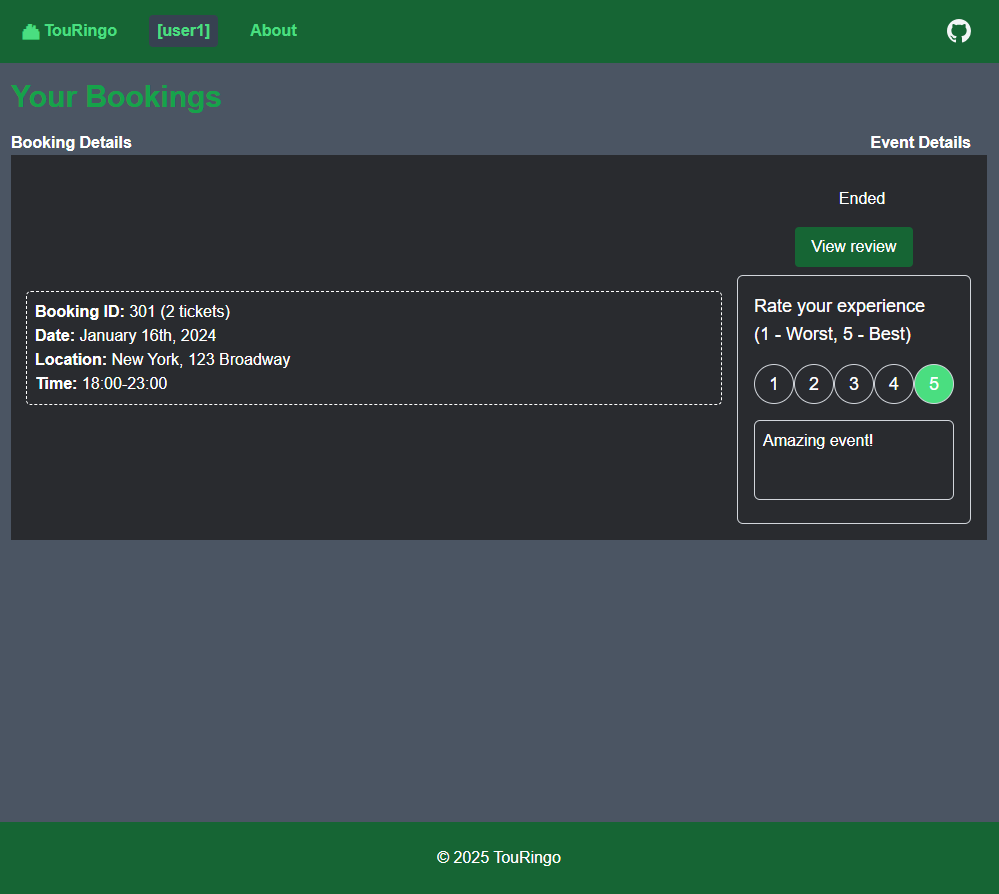
* מוצג למשתמש האירועים שיצרנו, המשתמש יכול לבחור לבטל את האירועים שלו, על ידי לחיצה על כפתור cancel.

ע"י לחיצה על “Create Event” נפתח אירוע חדש.



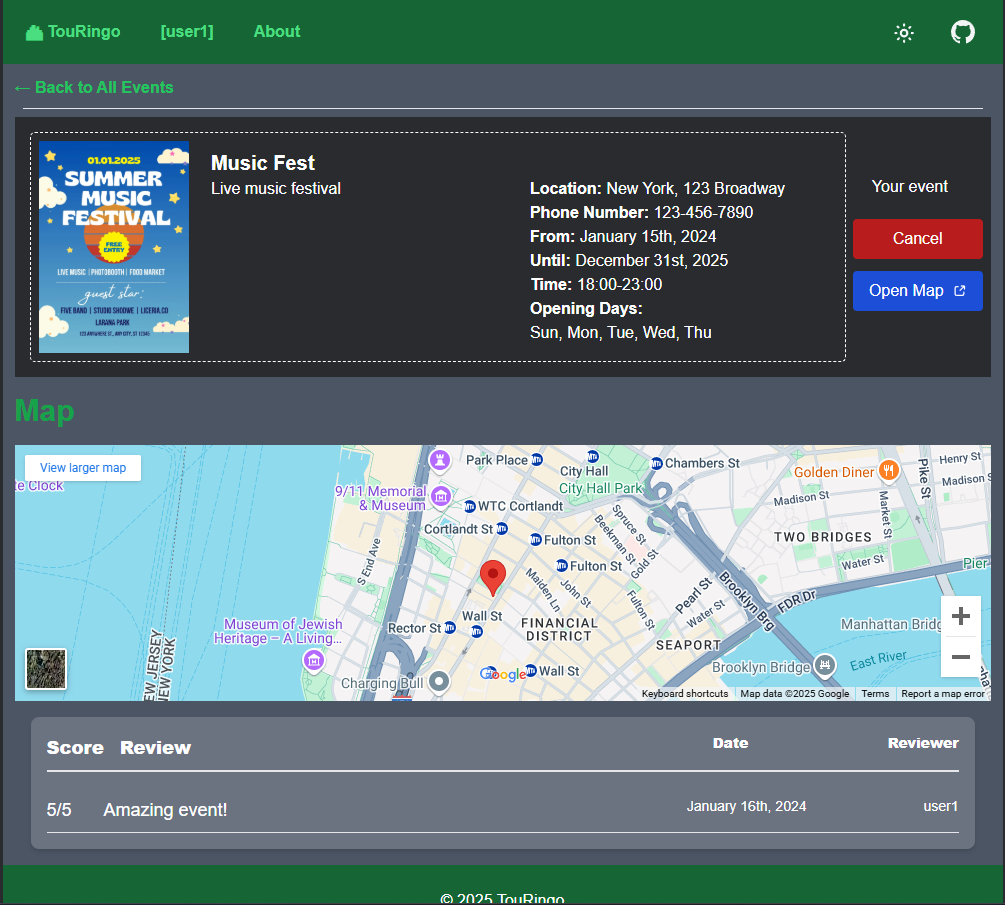
* בטופס נמלא את המידע הנדרש והאירוע שלנו מוכן לפרסום באתר.   
  יש את האופציה לביטול האירוע.

הבחירה הבאה היא "My Booking" שם נוכל לצפות באירועים אליהם שמרנו מקום.



* לחיצה על כפתור "View Review" תאפשר לכם לראות את הביקורת שאתם השארתם על האירוע.  
  לחיצה על מלבן האירוע תעביר אותנו לעמוד של האירוע.

נחזור לעמוד של אירוע



* בעמוד האירוע ניתן לראות בקטן חלון של גוגל מפות, כך נוכל לראות את הניווט בצורה נוחה מהאתר שלנו. בנוסף ניתן לצפות בכל הביקורות שהשאירו על אותו אירוע.
* במידה ואתם רוצים לצפות במפה המלאה, האופציה של "Open Map" תפתח חלונית נוספת לגוגל מפות.

בעמוד ה-About ניתן לראות את הפרטים שלנו, נושא הפרויקט, והכלים המרכזיים שנעשה בהם שימוש במערכת.

