



Software Engineering B.Sc. Final Project

Software Design Document

Authors:

ענבר תקדים 204117212

תבור כהן 200245587

דוד אביגד 200550010

Supervisor:

ד"ר מיכאל קיפרברג

Approved by: MK Date: _____



תוכן עניינים

• הקדמה

- מבוא
- מטרת המערכת
- תחום אחריות המערכת
- מיליון מונחים
- אילוצים

• ארכיטקטורת המערכת

- תיאור הארכיטקטורה והתוכנו
- מחזור חיים של המערכת
- SMARTIES

• סקירה ספרותית

- Server Side
 - PHP
 - NodeJS
 - Python
- Client Side
 - HTML5
 - JavaScript / ANGULAR
 - CSS3
- DB - NO/SQL
- Hybrid platform
 - iOnic
 - Native APP
- Tools

• תיקון המערכת (Design)

- Structural Design
- Interactions Design
- Software Architecture Pattern

• ניהול איכות התוכנה

- נספח א' - תיאור מבנים (Classes)
- נספח ב' - חלוקת משימות
- נספח ג' - לו"ז מפורט ותוכנו זמני
- נספח ד' - מסכי מערכת מעודכנים

הקדמה

מבוא

סקירה של המערכת:

אנו באים לפטור את בעיית מצוקת החניה על ידי הפעלת אפליקציה שיתופית בה כל אחד יכול לתרום ולהרוויח שימוש באפליקציה. משתמשים ידוחו למערכת על מועד עזיבת החניה - ומשתמשים אחרים במערכת יוכל לעשות שימוש במידע זה בזמן אמיתי או לתכנן שימוש עתידי. המערכת תקבע על דיווחים ושיתופיות מידע המתגמלת את שני הצדדים. מצד אחד מדוחה החניה הפנואה מקבל לחשבונו סمارטיז לשימוש במערכת ומצד שני, המחשף מוצא חניה פנואה.

משתמשים יכולים לתוכנן טרם היציאה מהבית את החניה וכן לחסוך זמן, כסף ועכבים.

שימוש בסмарטיז מודח שימוש הוגן במערכת - משתמש לא יוכל ניקודעובד דיווח שלא טרם למשהו אחר (חניה שהיתה תפוצה/דיווח לא אמין) , משתמש יקבל ניקוד על הדיווח רק לאחר שימוש מהצד השני אישר שהdioוח אכן היה אמיתי.

רענון השימוש במערכת:

- לכל משתמש פרופיל אישי באפליקציה.
- כשם המשתמש מתכוון לפנות חניה הוא יכול לדוח עליה ע"י תיאור מקום או תמונה.
- כל דיווח יתועד בחשבונו המשתמש ויהיה ניתן לערוך אותו כל הזמן שהואROLONETI.
- המערכת תשמר את כל הדיווחים במ Lager נתונים פנימי (להלן: מאגר הדיווחים).
- כאשר משתמש מהחשף חניה פנואה הוא פותח בקשה לחיפוש חניה באוצר ומתקבל מפה מעודכנת עם כל החניות הפנויות באוצר.
- משתמש אשר ביצע חיפוש יכול לשדרין לו חניה מתוך החניות שמופיעות על המפה.

מטרת המערכת

מטרת SmartPark מציעה פתרון מהיר, זול ושיתופי למצוקת החניה. הרעיון הוא לגורם למשתמש לרצות להשתמש במערכת ולסייע לאחרים במצבת חניה מתוך הצורך האישני שלו למצוא חניה. מטרת המערכת **העיקרית** היא לאפשר שיתופיות בין החניות הפנויות לנוהגים ולתורן בין מפנה החניה המדוח על חניה פנואה לנוהג המחשף חניה.

מטרות משנה:

- האפליקציה תאפשר **ניסיונות** לחניה הפנואה.
- האפליקציה תאפשר **שיתוף** חניה שאמורה להתפנות בעתיד.
- האלה בעליות הדלק והחניה של המשתמשים.
- חיסכון בזמן של הנהגים.
- השיתופיות תאפשר בניה של מערכת חברותית בין תושבי האזור.
- אפשרות **למציאות** חניה בתחום חניונים פרטיים.
- תהוויה פתרון אטראקטיבי לעיריות ולהנויינים שמעוניינים להפנות לקוחות לחניה קרובה בזמןני עומס.
- פלטפורמה לפרסום בענפים שונים, כגון: רכב, פנא, צרכנות וכו'.
- האפליקציה תהווה כליה לשיתוף חניה קבועה בתשלומים.
- האפליקציה תאפשר לב的日子里 חניה שאינה בשימוש בחלק משעות היום להניב הכנסה.
- שיתוף פעולה הדדי עם בעלי חניונים פרטיים.



תחום אחריות המערכת

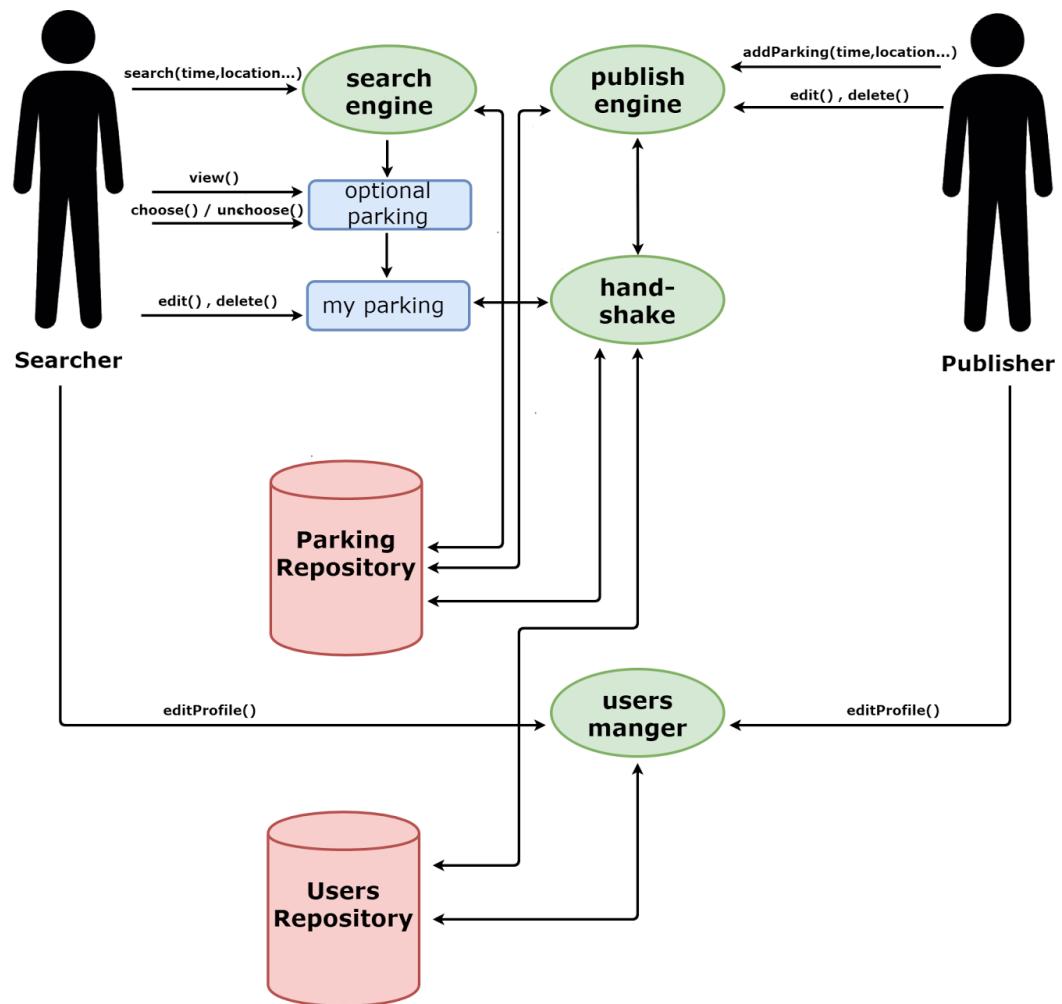
- לאחר תחיליך ההרשמה באמצעות *gPlus/Facebook*, אשר מספקת לנו את פרטי המשתמש, המערכת מתחלקת למספר חלקים:
- משתמש יכול לדוח על חניה שברצונו לפניו בשעה מסוימת, באמצעות טופס מקוון.
 - המשתמש מהצד השני יוכל לחפש חניות באמצעות טופס הזמנת חניה.
 - כאשר המשתמש מבצע חיפוש יחשפו בפניו כל החניות המתאימות לחיפוש שביצע.
 - המשתמש המCHIPש משרין לעצמו חניה על ידי בחירת החניה על המפה.
 - המערכת תספק ניוט לchniya הרצואה.
 - המערכת תוסיף ניקוד למדוח אם החניה שימושה את מחפש החניה וטוריד ניקוד למחפש החניה.
 - המערכת תתשאל את המשתמשים כדי לוודא האם נוצלה החניה שפורסמה.
 - המערכת תספק למשתמש מידע על החניות אשר האזינו לעצמו וגם את החניות אשר דיווחו עליהם.

מילון מונחים

- **משתמש** - כל אדם בעל חשבו במערכת.
- **סමארטיז** - מطبع וירטואלי לשימוש במערכת.
- **חשבון** - פרופיל המשתמש במערכת.
- **אזור** - רדיוס של X מטר מנקודה על המפה.
- **דיווח על חניה** - עדכון של תאריך, שעה ומיקום מדויק. נוצר ע"י המשתמש המCHIPש.
- **מוקד הדיווחים** - מערכת של עדכנים חדשים באיזור של המשתמש.
- **מדוחה** - משתמש המוסיף דיווח אודות חניה שתתפנה למועד הדיווחים.
- **חניה** - שטח על המפה בעל גודל (מ-6-3 קויי מדרגה) ומיקום.
- **חניה פנואה** - חניה שדווחה למועד העניינים.
- **חניה תפוצה** - חניה שדווחה כפנואה ומשתמש מיהר לתפוס אותה.
- **געץ צחוב** - חניה קרובה מאוד לאיזור בקשת החניה של המשתמש
- **געץ כתום** - חניה קרובה לאיזור בקשת החניה של המשתמש
- **געץ אדום** - חניה שאינה קרובה לבקשת החניה של המשתמש
- **מנהל מערכת** - משתמש עם הרשות מיוחדות לצורך עדכון המערכת.

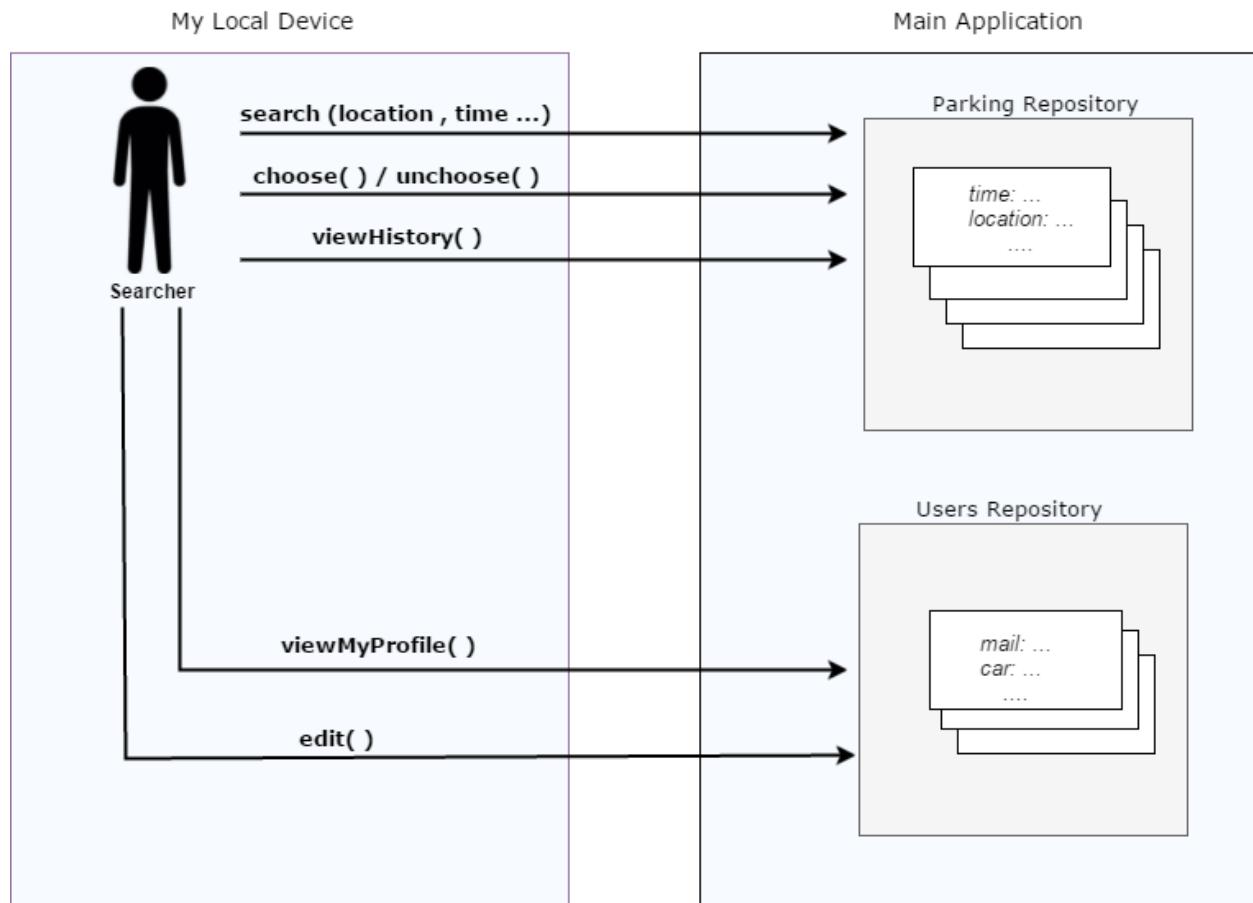
אילוצי המערכת

- **גישה לאינטרנט**
- **חשבון Gmail**
- **התקנת אפליקציית WAZE לצורך ניוט קווי (ניתן להחלפה ב-GoogleMaps)**
- **נדרש סמארטפון עם מערכת הפעלה iOS או Android.**
- **חנות גוגל או AppStore.**

Roles Diagram


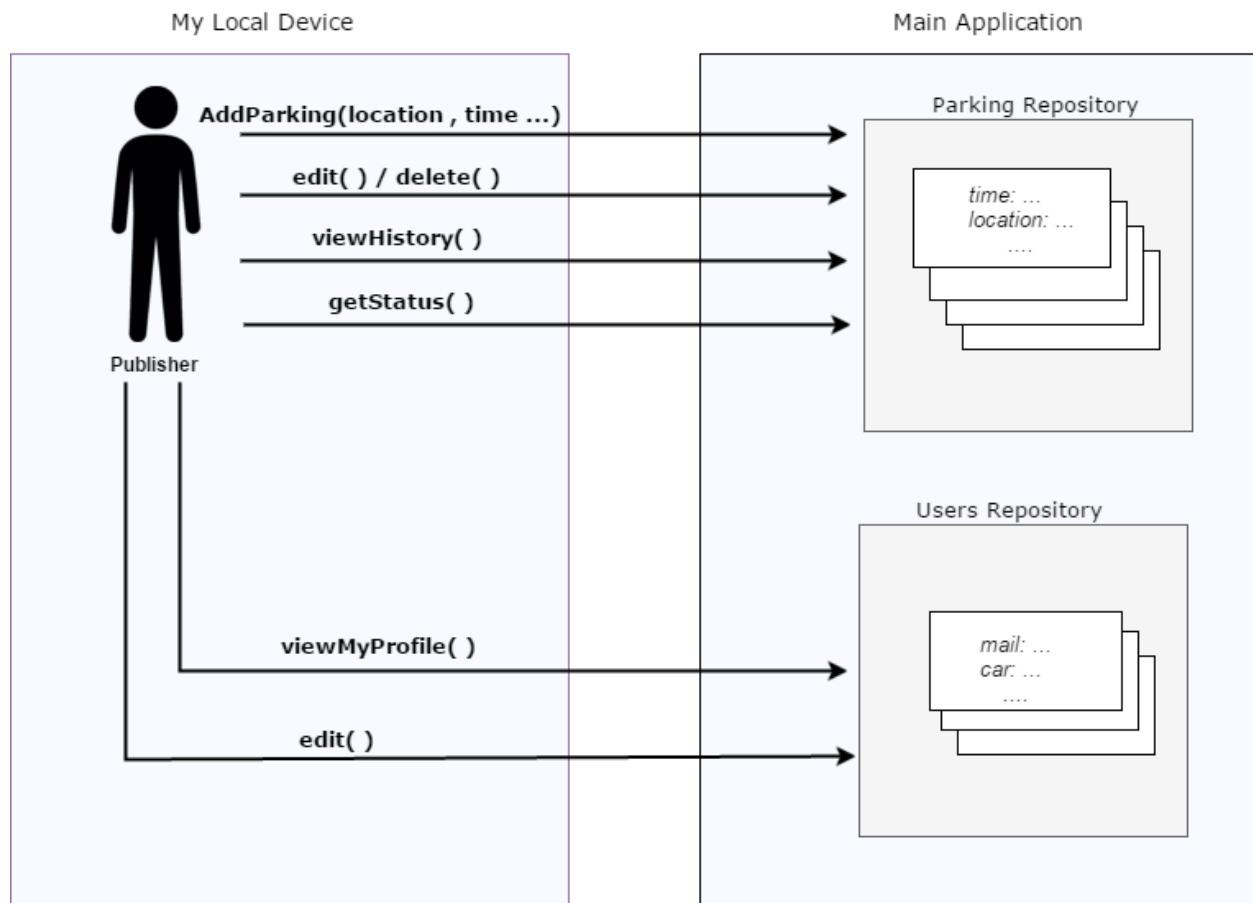


Data Diagram (SEARCHER Side)

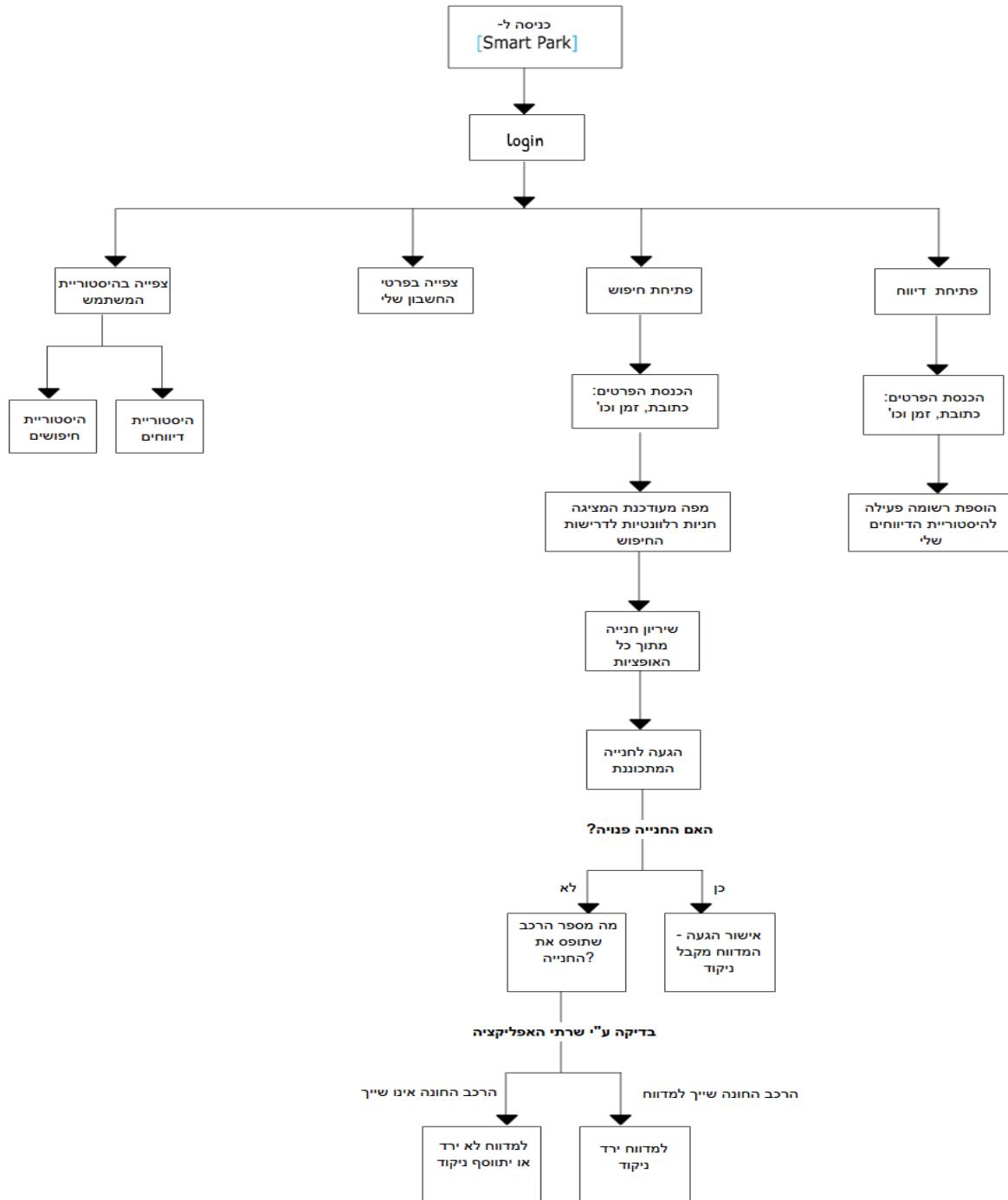




Data Diagram (PUBLISHER Side)



מחזור חיים של המערכת (Life cycle)



מערכת הנקודות - SMARTIES

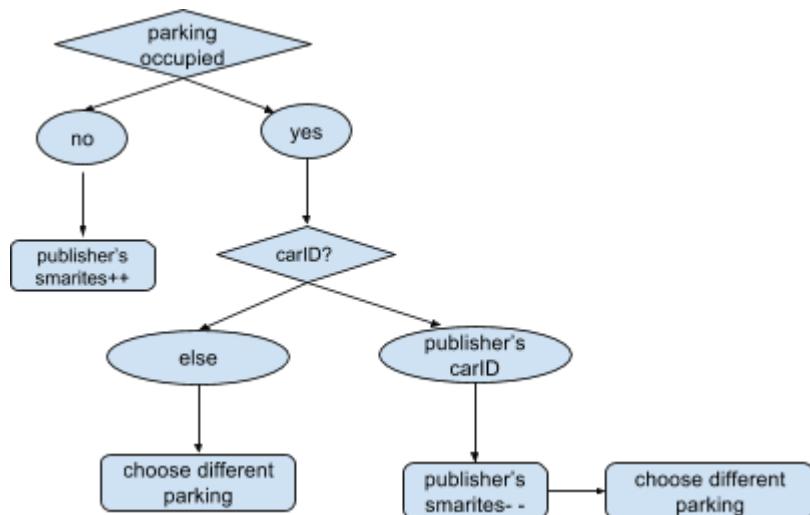
היעיון-

אפליקציית SmartPark הינה שיטופית וمعدדת שימוש וdioוח על ידי תגמול משתמשים במטבעות וירטואליים בעורთם יכול המשמש "להזמין" חניה עתידית במערכת. כל בקשה למציאת חניה שמתפנה עולה למשתמש מטבע 1 SMARTIES **וק במידה ושיתוף המידע** כל שיתוף חניה פנויה מזכה את המפתח המפנה החניה במטבע 1 SMARTIES **וק במידה ושיתוף המידע**. תרום למערכת וסיעו למשתמש אחר למצוא חניה. ישנו מספר מצבים בהם המערכת החכמה תחליט האם ליזoctה/ להוריד נקודות למשתמש. בדיקות שהמערכת תקח בחשבון-
האם בל אחד מהצדדים הגיע לחניה בזמן שנקבע מראש?
האם בל אחד מהצדדים היה אמין?
תהליך האיניות מתבצע בשלושה שלבים ו מבטיח את אמינות המערכת.

אלגוריתם המערכת-

1. **if (searcher book a parking)**
 - 1.1. searcher's **smarties--**;
2. **try ((booking ready) && (tries<2)**
 - 2.1. if (parking occupied)
 - 2.1.1. if (carID == carID(publisher))**
 - 2.1.1.1. publisher's **smarties--**;
 - 2.1.1.2. choose different parking
 - 2.1.1.3. tries++
 - 2.1.2. else**
 - 2.1.2.1. if (tries>2)
 - 2.1.2.2. choose different parking
 - 2.2. else if (!parking occupied)
 - 2.2.1. publisher's **smarties++**;
3. end

תרשים זרימת המערכת-



סקירת ספרות

הדרישות

צד שרת:

- השרת יעמוד בעומסים ויתן מענה מהיר למספר גדול של נגנים בזמן קצר.
- השרת נדרש לעבוד ביעילות ובמהירות שליפת נתונים מסודד הנתונים.
- השרת ינהל תהליכי (-Thread) בצורה אסינכורונית לקבלת תפקה מרבית בצורה מהירה מוביל לבلوم את הזרימה (Flow).
- השרת יעדכן את מסד הנתונים בצורה אוטומטית וייהי היחיד בעל הרשות גישה ישירה למסד הנתונים.
- השרת יתפעל בשירותי ענן (Heroku) ויפעל באופן רציף ללא הפסקה, גם לצורך קבלה או ביצוע של עדכונים.

צד לקוח:

- מציאות שפת תכנות שנוכל באמצעותה ליצור אפליקציה היברידית.
- פיתוח הצד לקוחiesel, מהיר ויציב ללא קפיאות ואפשרות לנתח נתונים.
- שפת התוכנות צריכה להתממשק מול api google maps.
- שפת התוכנות צריכה לאפשר עדכון של חלקי הרף ללא צורך בעינה מחדש במקש של המשתמש.
- שפת התוכנות צריכה לאפשר יצירה ממושך משתמש עשיר.

אפליקציית מובייל:

- מציאות פלטפורמה שעושה שימוש בקוד אינטרנט אך מנצלת את מערכת ההפעלה עליה היא רצה לדוגמה: ios או android.

האפשרויות

צד שרת

PHP



ראשי תיבות רקורסיביים של *PHP Hypertext Preprocessor*, PHP, שבמקור התבוססו על *Personal Home Page*. שפת תסריט המיעדת בעיקר לתוכנות יישומי אינטרנט לצד השרת, אך יכולה גם לרווח לצד המשמש. התחריר של השפה דומה לו של C והסמנטיקה דומה לו של Perl (שפה תוכנות דינמית). PHP נכתבה לראשונה ע"י רוזמוס לרודורף (Rasmus Lerdorf) בשנת 1994/5. השפה שלרדוור בנה הייתה קצר מסורבלת ושוונה מהשפה המוכרת ביום. זאב סורסקי ואנדי גוטמן, שני מפתחים ישראלים מהטכניון, פיתחו מהיסוד את שפת PHP המוכרת כיום, והעניקו לה-PHP את הפירוש הרקורסיבי: *PHP Hypertext Preprocessor*.

ל-PHP מספר יתרונות:

- השפה קלה להבנה וללמידה. במבנה השפה ניכרת השפעת שפת C, ובמידה מסוימת, גם Java.
- היכרות מוקדמת עם שפות אלו לא ספק מהוות יתרון לימודי השפה.
- בשימוש ותוכנות נכון, שפת PHP מספקת **בטחה גבוהה** מאוד וחוסמת כניסה של מחשבים אחרים למערכת.
- שפה מאוד ותיקה, נגישה ופולולארית אשר נמצאת בשימוש בקרב המונע מפתחים ברחבי העולם דבר ההופך אותה לגמישה מאוד בתהיליך יצירת פרויקט. כמו כן, בעזרה שפה זו נכתבו לא מעט סביבות פיתוח ואתרי אינטרנט מפורסמים. ביניהם - Facebook, WordPress, Wikipedia, Yahoo!, Tumblr

- ניתן להריץ את מנוע *PHP* על מגוון רחב של מערכות הפעלה ושרותים, כך ש-*PHP* אינה מוגבלת לשירותים או למערכות הפעלה מסוימות.
 - שפת *PHP* מותאמת בעיקר לעבודה עם אתרים ברשת. למעשה, ניתן לבנות אתר שלם עם קובץ *PHP* אחד.
 - שפת *PHP* היא שפת "קוד פתוח" (*Open source*), ולכן כמעט כל אחד יכול לפתח את השפה, והוא מופצת בחינוך.
 - שפת *PHP* ניתנת להטמעה ישירות בדף ה-*HTML*.
- לצד יתרונות אלו קיימים לא מעט חסרונות לשפה ובינהו-
- *PHP* לא אמורה לזרז למשכי זמן ארוכים (אינטראולים). בברית המחדל יישום מוגדר להפסיק את עצמו ברגע שהוא כבר פועל במשך 30 שניות, או אם הוא מגע לכמות מסוימת של שימוש בזיכרון. אפשר לבצע את ההגדרות הללו, וישומים יכולים להיות בנויים לזרז במשך זמן רב בהצלחה, אבל זה לא התחום שבו *PHP* מתבלט לטובה.
 - השפה לא מסוגלת להפעיל קוד במקביל. ניתן באמצעות כלים כמו *Gearman*, להعبر חלק מהעבודה לתהליכיים אחרים, אבל התהליך הזה רצ לא לצורך הטבעית ולפיכך לא מספיק טוב זאת ממש שדרך זו אינה מה *PHP* תוכננה להיות.
 - יש מעט מאוד תמיכה בניהול חבילות. הבודדים שקיים הם - *PEAR*, *Pyrus Packagist*, *PHPClasses*
 - מרגע תחילת ביצוע תהליך ב-*PHP* ועד לשינוי נתונים לא יכולים להישמר בזיכרון. ניתן לשמור את הנתונים באופן מתמשך באמצעות כלי צד שלישיים כמו מסד נתונים *Memcache* או בסיס נתונים מסורתי, אבל אז יש תקורה של תקשורת עם תהליכיים חיצוניים אלה.

Python



פייטון היא שפת תכנות דינמית מהנפוצות ביותר. פייטון תוכננה תוך שימוש DAG על קריומות הקוד, וכוללת מבנים המיעודים לאפשר ביטוי של תוכניות מורכבות בדרך קצרה וברורה. אחד המאפיינים הבולטים בתחום השפה הוא השימוש בהזחה להגדרת בלוקים של קוד (לא שימוש בסוגרים או במילים שמורות לצורך כך, כמו ברוב השפות הנפוצות).

לשפה ספרייה סטנדרטית גדולה וענפה, והיא תומכת באופן מובנה בהרחבה שלה אל שפות אחרות (כגון: *C*, *C++*, *Java* ו-*C#*).

לשפה שתי גרסאות ראשיות, פייטון 2 ופייטון 3. פייטון 2 איננה בפיתוח יותר (למעט תיקון באגים), אך עדין נתמכת, והיא הגרסה הנפוצה יותר בתעשייה.

פייטון כשפה מודרנית מכילה שלל יתרונות שהופכים אותה לפופולרית יותר ויוצר בקרב חברות תוכנה הבולטות שבהן: *Instagram*, *Pinterest*, *Django*, *Google*, *NASA*, *Yahoo*.

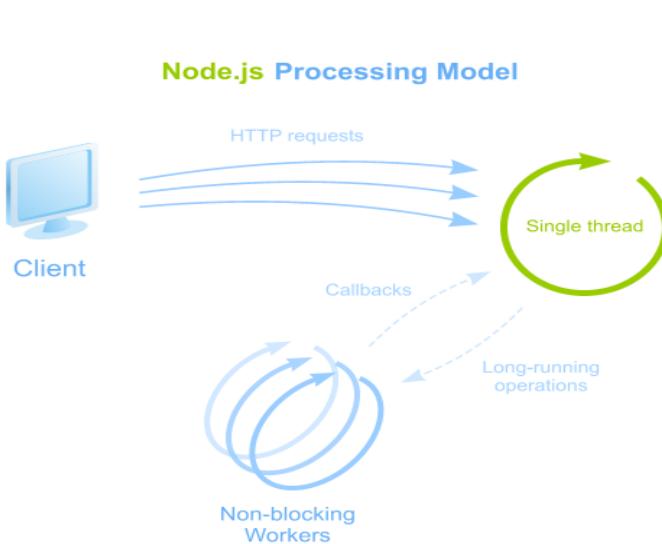
תוכניות פייטוןעשויות לכלול קבצים רבים. על מנת להריץ קוד פייטון יש לבחור מימוש: מהדר או מפרש, שיתרגם את הטקסט של התוכנית לפקודות שיתבצעו במעבד. לפיקטון קיים "מימוש-ייחוס" כלומר מערכת שהיא זו שגדירה את התנהלות של קוד שנכתב בשפה - בשם *CPython*. מימוש-היחס פועל בשני שלבים נפרדים: שלב ההידור ושלב הרצה.

בין יתרונותיה -

- שפת פיתוח נוחה לכתיבה, קריאה והבנה של הקוד. דבר שמקל על תחזוקת הקוד גם מצד מפתחים אחרים שאינם כתבו את קוד המקור.
- בעלת סינטקס מאד פשוט, ומבנה נתונים (*Data Structure*) מאוד גמישים וдинמיים.
- בעלת יכולת התכווננות והתאמאה אופטימלית לריצה מיטבית על פלטפורמות מגוונות.
- שפה ישרה לניטוח ועיבוד נתונים מקדים (*pre-processor*) בעולם ה-*BigData* הודות לאופציית הפיתוח האינטראקטיבי המאפשר לכתוב שורות קוד בחודות אשר יורכו מידית.
- פינוי הזיכרון מבוצע בצורה אוטומטית, על ידי מנגנון איסוף זבל, בעיקר תוך שימוש במניות התייחסויות, ולא דורש התייחסות מפורשת מצד המתכנת.

חרונוטיה הבולטים של השפה עדין נתונים למחלוקת:

- השפה נחשבת לאיית ביחס למתחרות אותה בעיקר בגלל הזמן הנדרש לפירוש (*Interpreted languages*) השפה מול שפות אחרות אשר עוברות הידור (*Compiled languages*) תהליך מהיר יותר משמעותית. חשוב לציין שיש אשר חולקים על עובדה זו ולטענתם מהירות ריצת התוכנית תלולה בלוגיקת כתיבת הקוד.
- השפה תומכת בכיסוס פחות משפות מונחות-עצמם אחרות. לטענת המפתחים דבר זה מקנה לה גמישות יתרה ונונן יד חופשית למפתחים.
- פיתון אינה שפה מומלצת לפיתוח אפליקציות מובייל ואינה בחירה טוביה לפיתוח יישומים עם ריבוי משימות.
- לשפה יש הגבלות בגישה *DataBase* שונים.



Node.js



שפה גמישה, מונחתית
איורוועים ואסינכרונית
המאפשרת גמישות רבה
וביצועים מצוינים יחסית
למקבילות שלה. הגרסה הראשונה של התוכנה
פותחה בשנת 2009 כתובה ב-*c++* וمبرוסטת
על *V8*.
במקור תוכננה כתיבת אפליקציות רשת
מתכוונות (*Scalable*) כגון שרת *HTTP*.
הקוד של *Node.js* אינו מורץ בדף של
הגולש, אלא בצד השרת.

- הינה מעין מנוע המרים פקודות *JavaScript* ללא דף דין הלכה ופועל בצד השרת.
- שפת קוד פתוח (*Open Source*).
- מאפשרת תכונות בגישה אסינכרונית מונחתית איורוועים שנועדה לטפל ביעילות בבקשת צד לקוח (*Requests*).



- מרים תהליכי ייחד בכל פעם ולכון אינה תוקעת את הזרימה (*Flow*) הטבעית של המערכת.
- משתמשת בסינטקס של *JavaScript* דבר הופך אותה לשוטה לכתייה, קריאה והבנה.
- מאפשרת חלוקת עבודה למודלים חיצוניים, דבר שהופך את הקוד לנקי ומסודר יותר.
- בעלת מערכת ניהול חבילות (**Packages**) המאפשרת התקנה פשוטה ומהירה של חבילות דרך *NPM*.
- שפה פופולרית עם קהילת מפתחים שיתופית נרחבת.
- ייילה מאוד בעבודה עם קבצים. (*Streaming*)
- *Node.js* שירות אינטראקט מובנה שנitin להגדרה ושינוי.
- ניתן לשמור נתונים בזיכרון בקלות ובה. כיש צורך בשיתוף נתונים בין ל��וחות שונים (לדוגמה, במשחק מרובה משתתפים) שיתוף הנתונים כבר מובנה.
- *Node.js* היא הפתרון הטוב ל-
 - *Web Socket Server*
 - *Fast file upload client*
 - *Data streaming*
 - *Ad server*
 - *Stock exchange software*

לעומת זאת,

- אינה מאפשרת עבודה בהיררכיות קוד. לדוגמה כשפונקציה אחת תלולה בשנייה מצב שעשוי להוביל ל- "*CallbackHell*".
- אינה מותאמת לעובדה עם מסד נתונים רלוונטי.
- נדרש ידע מתקוני קודם ב- *JavaScript* להבנת אבני שפת *Node.js* וכן ידע ומומחיות בתוכנות אסינכרוני.
- *Node.js* הינה פלטפורמה חדשה מאוד, ה-*API* שלה לא מספיק יציב וטרם נבדקה ביסודיות. ה-*API* של *node* משתנה בתדריות גבוהה, דבר הדורש כתיבה מחדש מוחדשת לעיתים קרובות של יישומים גדולים/ארוכי טווח.

MySQL (SQL)



MySQL (SQL) מסד נתונים יחסי, רב נימי ורב משתמשים מבוסס שפת *SQL* (*Structured Query Language*). התוכנה פותחה במקור על ידי החברה השוודית MySQL AB. בשנת 2005 רכשה חברת אורקל (אשר אחד מ מוצריה החשובים הוא בסיס הנתונים אורקל) את החברה. הרכישה העניקה לאורקל שליטה על תוכנה של מתחרה החשובה בשוק. התוכנה היא חלק מ-LAMP, אוסף תוכנות תשתיית פופולריות שעומדות בבסיסם של אתרים חשובים רבים, כגון גוגל, ויקיפדיה. מערכות ניהול תוכן ובות (כגון וורדפרס ודרופול) משתמשות בה כבסיס נתונים.

נקודות חזקה של MySQL

- קל ללימוד ולשימוש באופן ייחסי לבסיסי נתונים אחרים והוא מותאם לעבוד בצורה הטובה ביותר עם שפת הפיתוח *PHP*.
- MySQL יכולה לפעול על מספר רב של פלטפורמות ביןיהם: *UNIX, Mac OS X, Windows*.
- מסד נתונים מבוסס קוד פתוח (*Open Source*).
- מסד נתונים טבלי גמיש יותר, בעל ביצועים גבוהים יותר ואמינות גדולה יותר.
- בעל מגנוני אבטחה חזקים.

MongoDB (NoSQL)

המילה *Mongo* נגזרת מתוך המילה *Humongous* שמשמעותה - עצום. דבר המרמז על עובודה עם *BigData*. בסיס נתונים זה, הוא המוביל בעולם בקטגוריות *NoSQL*.



בסיס הנתונים נשען על מבנה של מסמך (Document-Oriented Database) בניגוד לבסיסי נתונים טבליים (כמו *MySQL* ו-*SQL Server, Oracle*) העובדים מעלהלאות הקשורות. מבנה המסמכים עובד מעלהימוש של JSON הנקרא *BSON* (עקב שמירה המידע בינהית - *Binary JSON*). זהו יישום חופשי והינו מוצר חוצה-פלטפורמות.

בסיס הנתונים כתוב בשפות *C, Javascript, C++*. מסדי נתונים לא רלציוניים פותחו על מנת לתת מענה מכךוי לעלייה במסיבית בנפח הנתונים המוטמעים ע"י המשתמשים.

תכונותיו העיקריות של MongoDB:

- אחסון ממוקד מסמך. הטעמאות סכימות גמישות לניהול מסמכים פשוטות ולא צורק בקביעת מבנים זהים לכל אחד מן המסמכים.
- תמיכת אינדקס מלאה.
- שכפול מידע אוטומטי ואמין הנצפה על פני רשתות *LAN* ו-*WAN*.
- שינוי קנה מידת אוטומטי.
- יכולות ביצוע שאילותות עשירות, המתבססות על המסמך המוטמע במסד הנתונים.
- עדכונים ושינויים במהירות ובקלות > אחסון קבצים בכל גודל ובאופן פשוט.
- תמיכה ארגונית שוטפת על ידי קהילתית *MongoDB*, בעזרת מתן ייעוץ, הנחיה והדרכה זמינה.

SQL Vs NoSQL

NoSQL: (Non-Relational)	SQL: (Relational)
עושה את הבלתי אפשרי - NoSQL יכול לשלב כל סוג של נתונים, תוך מתן כל התכונות הדרשיות כדי לבנות יישומים עשירים בתוכן.	לא גמיש- ביום עולם ה-"Data" התרחבות והפרק למגוון מאוד. סוגי מידעים (Data Types) כמו Tweets, וידאו, GIF ואנימציות Podcasts קשה מאוד לא ניתנו לשומר על גבי בסיס נתונים רלוונטי.
כוונון בגודל - כיוול המידע מוגבנה בתוך מסד הנתונים. זה אוטומטי וסקופ. וניתן להתרחב בד בבד במידה וקהל הידע גדל או משתנה (-אזור, שפה, סוג המידע)	לא ניתן לכובונו (Scaling) - סוגי המידעים משתנים בין שפה לשפה וקהל הידע הינו דינامي ומשתנה. היכולת לככילה את בסיס הנתונים מחדש שונה או קהלה רחב יותר מאזרים שונים, היא הכרחית וחינונית ל-DB מודרני.
-\$ מידע הנשמר ב-DB לא רלוונטי הוא יותר יעיל וחושך הרבה יותר מקום ולכון לא דורש תחזוקה שוטפת של מאגרי מידע גדולים. דבר שבתווח הרחוק יוכל לעלות של בעשיית מ-DB רלוונטי.	\$\$\$\$ בטוחה הרחוק העוליות גודלות יותר. ב-DB לא רלוונטי התחזוקה השוטפת של כמה קבצים גדולה מאוד עליה הרבה יותר מתחזקה של DB המכיל צ'אנקים (Chunks) של טקסטים בלבד.

לסיכום - יתרונות מסד נתונים לא רלוונטי:

- יכולות עיבוד נפחן נתונים עצומים
- יכולות עיבוד מהירות
- תוכנות תוכנות גמישות, ידידותיות וקלות ליישום
- יכולות ניהול שינויים באופן מהיר, בזמן אמיתי ללא נזודות כשל
- יכולות ניהול מספר רב של שרתים, באופן שדרורתי ולא צורך בהתאמות מיוחדות
- תמיכה בשכפול אוטומטי המאפשרת התאוששות מהירה מאסון

ANGULARjs

היא מסגרת (framework) של JavaScript שמיועדת לכנתיבת SPA (אפליקציות מבוססות דף בודד). המשמעות של SPA היא שתוכן הדפים מוחלט בלי צורך לרענן את הדף, וכתוכאה מכך האפליקציה נראית הרבה יותר מקצועית ודומה בהתנהגותה לאפליקציות לדסקטופ.



Angular היא קודם כל מסגרת, כלומר, היא מציעה פונקציות שמקלות על עובdot המפתחים מפניהם שבסמוך להשתמש בהרבה פונקציות של JavaScript אפשר להסתפק בפונקציות פשוטות להפעלה של Angular. אングולר יכולה לעבוד בצורה MVC או MVVM. הרעיון שאングולר מיישם הוא כדלקמן: המפתח יגיד את הקשר בין אלמנט מסוים לפונקציה מסוימת או משתנה מסוים, ואングולר תעביר את המידע ביניהם, תוך התחשבות באירועים שונים.

 יתרונות:

- אין צורך להשתמש בפונקציותבולטות- אングולר מנתה את מבנה DOM ובונה את הקשרים ע"י התוכנות הספציפיות, כך שיש פחות כתיבת קוד, קוד נקי, קוד קרייא ופחות שגיאות.
- אングולר משנה ישירות את ה- DOM ולא מוסיף קוד HTML INNER, ולכן מהיר יותר.
- קישור המידע נעשה לא עברו כל control או שינוי ערך אלא בנקודות ספציפיות במהלך ריצת קוד ה JAVASCRIPT. דבר זה תורם משמעותית לביצועים משופרים.
- ישן מספר דרכים שונות לבצע את אותה פעולה- מגוון אופציונות לנוחיות המפתח.
- מכילה מאפיינים כמו Routing, Animateות וכו'.
- נתמכת באמצעות סביבות פיתוח כדוגמת IntelliJ IDEA, Visual Studio .NET IDEs, NET IDEs.
- נתמכת ע"י גוגל, וקהילה נרחבת של מפתחים.
- השימוש בפילטרים פשוט וקל.
- שימוש בתבניות קבועה.
- העברת מידע בשני כיוונים- סyncron אוטומטי של מידע בין ה- Model וה- View. כאשר הela model משתנה, אングולר יגרום לשינוי להתקדמות אוטומטית.
- שימוש ב- MVC באングולר הלוקה מקבל תשובה מהשרת, הדפספן של הקלינינט מנתה את התשובה ליצירת model ויצירת view.

 חסרונות:

- אングולר גדול ומורכב- הדריכים השונות לבצע את אותה פעולה הופך את מציאת הדרך הטובה והנוחה ביותר למורכבות.
- קשה ללמידה- יש צורך להשקיע מאמץ למדוד זאת, הרגלי הכתיבה של מפתחים שונים עשויים להקשות על חיבור חלקיקי הקוד לפתרון אחד.
- מחזור החיים של אפליקציה באングולר מסובך- על מנת להבין זאת צריך לIALIZED ולהריץ את הקוד- פעולה שאין לא אינטואיטיביות.
- כאשר הפרויקט גדל הרבה פעמים יש צורך לכתוב מחדש קוד קיים ולהחליפו בגישה אחרת.
- UI לא מוצלח- דבר זה מגביל את המגוון של טפסים, רשימות ועוד.
- אングולר נתמך בדפספן Internet Explorer רק מגירסה 8.0 .

Hyper Text Markup Language

שפת בניית אתרים לסייעון טקסט, שפת תגיות, אשר נותנת הנחיות לדפדףים בנווגע לאופן הצגתו של דף האינטרנט מבחן טקסט, תמונות, טבלאות עיצוב ועוד. זהה שפת הקוד הבסיסית ללבית ה-web עבור בניית אתרים. משמע – כל דף אינטרנט טכני שאנחנו רואים, מבוסס למעשה על HTML.



על מנת ליצור מסך עשיר ומורכב, וחווית משתמש אינטראקטיבית ומלאת חיים, נדרש התקנת **תוספים** (plugins) לדפדףים, תוספים אלה העירימו בעיות על מומחי בניית אתרים והמשתמשים כאחד: אי תאימות לפלטפורמות שונות, בעיות נגישות, פגיעה בקידום תכנים במונחי החיפוש (SEO) ועוד. HTML5 הוא גרסה מתקדמת (בהרבה) של HTML, אשר פותחה על ידי W3C עברו בניית אתרים, קריילה בינלאומית המפתחת תקנים במטרה להבטיח את צמיחת הרשת. גרסה זו כוללת יכולות עשירות של תצוגה אינטראקטיבית כחלק בלתי נפרד ממנו, וספקת כלים חדשים למורים, המיעדים להקל על עיבודת המפתחים ואנשי בניית אתרים באשר הם ולאפשר שילוב רכיבים שונים באתר – סרטוני אנימציה, וידאו, סאונד וAFXים נוספים, ללא צורך בהתקנה או הרחבה כלשהי.

יתרונות:

- שפה פשוטה ללימוד ולשימוש.
- כל דפדף תומך ב HTML.
- השפה מאפשרת שימוש בתבניות קבועה שמקלה על כתיבת דף אינטרנט.
- מותרת לשימוש על ידי כל מפתח אתרים, ללא צורך ברכישת זכויות יוצרים מחברת כלשהי, והיא ניתנת לקריאה בכל סוג המערכת.
- Loose Syntax - גמישה בסטנדרטים, לדוגמה אם תגיית שנפתחה לא נסגרה לא תופיע שגיאה.
- מאפשרת ליצור קישורים לעמודי HTML נוספים או לסוגים אחרים של נתונים. בנוסף, היא תומכת ב מולטיימדיה, וידאו, טקסט, ציליל, תמונה ועוד.

חסרונות:

- שפה סטיטית- אי אפשר ליצור דפי אינטרנט דינמיים באמצעותה בלבד.
- לעיתים, המבנה של עמוד ה-HTML קשה להבנה.
- התוכן מוצג בצורה סטנדרטית, על מנת להפוך אותו לעיצוב מותאם צריך להשתמש בשפה אחרת ותאזר באמצעותה כיצד הדף צריך להיראות כמו CSS.
- ישנן תגיוט שהפכו לחסרים שימוש בגל שפה אחרת שעבדה עם HTML החליפה את העבודה של התגיוט, לכן צריך ללמד את אותן השפות (לדוגמה CSS), בנוסף, צריך לדעת מי הן התגיוט שלא בשימוש יותר.
- נדרש כתיבת קוד רב עבור דף אינטרנט פשוט.
- שגיאות עלולות להוביל לתוצאות יקרות לתיקון.
- אבטחה- מאפייני השפה לא מתמודדים היטב עם אבטחה.
- נדרש דפדף שיתרגם אותה נכוון.
- שפה דקלרטיבית- דבר המגביל את הכוח הפונקציונלי של השפה לעומת שפות פונקציונליות.

JavaScript

שפת תכנות דינמית מונחית-עצמים המותאמת לשילוב באתר אינטרנט ורצה על ידי דפדף האינטרנט לצד הקוד. השפה מרחיבה את יכולות שפת התגיוט הבסיסית HTML ומאפשרת בכך ליצור יישומי אינטרנט מתחככים יותר. למעשה,



רוב אתרי האינטרנט המודרניים משלבים שפה זו. היא ידועה בעיקר בשפה המוטבעת בדף HTML על מנת להציג דפים דינמיים, שימושבת בהם תוכנה. קוד ה-JavaScript המשולב בדף HTML מבוצע על ידי הדפדפן.

יתרונות:

- השפה מושצת בצד הלקוון- הקוד מושץ על המעבד של המשתמש, במקום בצד השרת, דבר זה משפר את רוחב הפס ואת העומס על השרת.
- השפה קלה לישום- השפה קלילה ללימוד ומיליה תחביר שקרוב מאוד לאנגלית.
- שימוש השפה באובייקט ה- DOM מאפשר פונקציונליות רחבה ופתרון בעיות מותאמות.
- השפה לא דורשת תהיליך קומpileציה- כך שלא צריך להתקין קומpileר במכשיר. הדפדפן מפרש את השפה כשם שעושה עם תגיוט HTNL.
- שפה קלה לניפויי שגיאות (Debugging) ובדיקות- הקוד "מפורש" שורה אחר שורה, השגיאות מגיעות לצד מספרי השורה הרלוונטי לשגיאה. כך קל לאבחן שגיאות בקוד ולתקן אותן.
- שפה מהירה יחסית למשתמש הקצה (הלקוון)- התוצאות והчисובים מתבצעים בצד המשתמש כמעט מידית (תלוי במשימה), מכיוון שאין צורך לשולח חישובים לצד השרת ולחכות לקבלת תשובה- השפה הופכת ל מהירה.
- שפה מבוססת אירופאים- פירוש הדבר הוא שקוד שונה פועל בזמן אירופאים שונים. לדוגמה מתוודה A תופעל כמשתמש הקצה יلحץ על כפתור, ואילו מתוודה B תופעל כמשתמש הקצה ייזעכבר מעל אובייקט.

חסרונות:

- אבטחה- מכיוון שהקוד רץ בצד הלקוון (המשתמש) הוא יכול להיות מנוצל למטרות זדון. דפדףנים סטנדרטיים מכילים מגבלות לכך אך למרות זאת ניתן להריץ את הקוד האזוני, لكن המפתחים צריכים כתוב קוד שלא יוכל להיפרץ בקלות.
- תלות במשתמש הקצה- הקוד עשוי להיות מתרגם בצורה שונה על ידי דפדים שונים, בעוד שכתיבת קוד בצד השרת תמיד יניב פלט זהה. לכן, פעמים רבות נדרש לבדוק את האפליקציה על דפדים שונים כדי לאבחן חוסר אחידות בפונקציונליות ובממשק.
- גרסאות שונות- במהלך השנים נוספו שיפורים לשפה על מנת להפוך אותה לשפה אוניברסלית כמו שניתן, בעוד רוב הגרסאות הפכו לסטנדרט אוניברסלי, עדין קיימות גרסאות שונות שעלייה מתבססים אתרים אינטרנט רביים בעולם.
- השפה אינה מתוקנת ברמת גבוהה- יש מגוון ספריות זמינות שמקצרות את זמן הפיתוח, אבל יכולות להוביל לתוצאות שתלויות במערכות של הלקוון שעלייה מושץ הקוד.
- קוד גלוי- הקוד זמין לכל אחד שחפץ לראותו.
- חלקו הרוצה איטיים- לא משנה כמה השפה מתרגמת מהר, מבנה העמוד Javascript DOM (HTML) מאט כאשר מעובד עם קוד HTML).

אפליקציית מובייל

Hybrid platform

אפליקציה היברידית היא אפליקציה המפותחת בטכנולוגיות של פיתוח יישומי אינטרנט (WebApplication) כך שנitin להריץ אותה על כל פלטפורמה, בין אם זה מכשיר עם מערכת הפעלה אנדרואיד או עם מערכת הפעלה IOS .

אפליקציה היברידית מרגישה ונראית כמו אפליקציית native , אך למעשה זה עטוף במעטפת של web application. האפליקציה מפותחת ב html5 and javascript native אשר מספקת גישה לתוכנות שיש לפלטפורמה. במערכת הפעלה של אפל למכשירים ניידים, ה- Web view לא זהה לדיספן סאפי, ולכן יכול לגרום לעבודה מרובה עם ה- debugging .

	Hybrid	Native
App Features		
Graphics	HTML, Canvas, SVG	Native APIs
Performance	Slow	Fast
Native look and feel	Emulated	Native
Distribution	Appstore	Appstore
Device Access		
Camera	Yes	Yes
Notifications	Yes	Yes
Contacts, calendar	Yes	Yes
Offline storage	Secure file system, shared SQL	Secure file storage
Geolocation	Yes	Yes
Gestures		
Swipe	Yes	Yes
Pinch, spread	Yes	Yes
Connectivity	Online and offline	Online and offline
Development skills	HTML5, CSS, Javascript	ObjectiveC, Java



Ionic – סביבת עבודה חינמית לבניית אפליקציות היברידיות, אשר מספקת רכיבים של app Native ו- web app.



יתרונות:

- מציע מגוון רחב של רכיבי UI עוזר למפתח ליצור אפליקציה אינטראקטיבית.
- שפות התכנות הם אנגולר Html ו- Css שהן קלות לשימוש ומאפשרות אינטראקטיביות Backend.
- מציע עורך גרפי שבאמצעותו קל ונוח ליצור אפליקציה, ולהכניס קומפוננטות UI.
- באמצעות ארגז כלים זה ניתן לפתח במהירות ובקלות אפליקציה האפליקציה שנוצרת יכולה לזרע על כל הפלטפורמות רק חלק מהקוד ה-native צריך להכתב מחדש כדי שהאפליקציה תעבור על כל סוגי המכשירים.
- קל לתהזק את האפליקציה וגם להחליף סביבות פיתוח.
- ניתן להוסיר פונקציונליות חדשה לאפליקציה פעם אחת בניגוד לאפליקציה native שצריך את אותו פונקציונליות לשכפל לכל פלטפורמה.

חסרונות:

- האפליקציה לא תזרע מהר כמו אפליקציה שהיא native.
- תלוי במהירות הדפדפן
- פונקציונליות שיש בموבייל ואין בפיתוח web application יהיה פחות מהיר לפיתוח על ידי אפליקציה היברידית.
- הגרפיה תהיה יותר איטית אשר תפחית את חווית המשתמש

אפליקציות native



אפליקציית native היא אפליקציה שפותחה לפלטפורמה מסוימת והיא מפותחת בשפה מסוימת, ולכן האפליקציה יכולה לזרע רק על אותה פלטפורמה ולא על כל סוג הפלטפורמות. אפליקציה לאנדרואיד מפותחת באמצעות סביבת הפיתוח אנדרואיד סטודיו, אשר מספקת דיבוג התוכנה, ניהול הפיתוח של האפליקציה, ניהול גרסאות ושאר כלים שפתחים צריכים. יש צורך בכלים אלה לשם שאפליקציה native קשה יותר לפיתוח.

ישנן כמה דברים אשר נרווח אם מפתח אפליקציית native לאנדרואיד:
Multi-touch – מגוון רכיבי UI שמאפרות לנו לעשות מחוות על המחשב.

Fast graphic api – פלטפורמת native- מספקת את הגרפיה הכי מהירה שניתן לחווות במכשיר, וזאת חשוב מאוד לאפליקציה שיש בה זרימה של הרבה נתונים והרבה עבודה עם רכיבי ממשק משתמש.
אנימציה מהירה יותר – רלוונטי יותר למשחקים שרצים על המחשב, או בשילוב אינטראקטיבית עם מוחיקה ותמונה.

שימוש ברכיבים מובנים - יותר קל לעבוד עם רכיבים שmagics עם המ构思ר כגון מצלמה, אנשי קשר וכיוצא באלה.

קל לשימוש - אפליקציה בפיתוח אנדרואיד למשל היא מה שבדרך כלל אנשים מחפשים כיון שהיא קלה לשימוש.

תיעוד - יש הרבה מאוד מדריכים וספרים על פיתוח אנדרואיד.

יתרונות :

- אפליקציה רצה מהר על אותה פלטפורמה
- אפליקציה יותר אמינה
- חווית משתמש יותר נוחה וטובה למשתמש
- ניתן להשתמש בפונקציונליות רחבה יותר ביכולות שה构思ר מציע כמו שימוש במכשיר או בחישונים אחרים.
- ניתן לפתח תבניות עיצוב ב-native .
- אפליקציות שפותחות באנדרואיד מקבלות תמיכה בchnoot האפליקציות
- מספק כלים לפיתוח שמלים על המפתח
- java היא שפה נוחה לשימוש ומוכרת הרבה מתכנתים.
- רכיבי זען ותוכנות נוספות של המ构思ר משולבות בצורה חלקה באפליקציה ובפיתוחה.
- יש דוקומנטציה מפורטת וטובה.
- גרפיקה מהירה יותר מאשר אפליקציה היברידית.
- פיתוח בשפה java לאנדרואיד מאפשר לתוכנת בשיטה של תוכנות מונחה עצמים.

חסרים :

- פיתוח אפליקציה בjava יכולה לעבוד רק על מערכת הפעלה אנדרואיד, ככלומר נדרש לכתוב מחדש קוד למערכת הפעלה אחרת.
- נדרש צוות פיתוח ספציפי שמתמחה בפיתוח אפליקציות לאנדרואיד
- פיתוח ותחזוקה מאוד יקרה

כליים לפיתוח

יתרונות:



- מאות Packages זמינים.
- עוזר להישאר בפוקוס על הקבצים הרלוונטיים של מה שהמתקנת צריכה.
- קל ומהיר למצוא קובץ שהמתקנת רוצה לעבוד עליו.
- פעולות של פיתחה, סגירה, חיפוש וכיוצא באלה מתבצעים בצורה חלקה ומהירה בסביבת הפיתוח sublime.
- סביבת פיתוח מאוד גמישה.
- הדגשת הקוד בצבעים טוביה מאוד ונוחה למתקנת.
- חווית כתיבת קוד טובה ונוחה
- התוכנה עולה מהר, פחות משנית
- רמת אינטגרציה גבוהה בתיקון ושיפור בכתיבה העקסט, כמו תיקון שגיאות
- ניתן לעשות אותן שינויים בחלקים שונים בטקסט בו זמן קצר
- הרבה אפשרויות ופעולות זמינים בתפריט
- ניתן לפתח במהירות קובץ ולקפוץ ישר לsymbol מסוים
- טוב לפיתוח web app

חסרונות:

- אין ממש טוב לעובדה עם GIT.
- השימוש בקובץ chosz כדי להגדיר הגדרות לא נוח לשימוש ומוגבל.
- אין גרסת console זינה בסאבלים
- אין קומפיילר

יתרונות:



- מאות Packages זמינים.
- מאפשר לשים breakpoint ולדבג בתוכו js ו-nodejs .
- מותאמת לעובדה עם GIT.
- סביבת פיתוח מאוד יציבה שלא כורסת.
- יש הרבה פונקציות שעוזרות למתקנת לתכנת ביותר קלות ויעילות.
- תומך בהרבה שפות תכנות.
- עוזר למתקנת בכתיבת הקוד לדוגמה בכך שמרת את הפונקציות שניתן להפעיל על האובייקט.
- ניתן לראות שני קבועים זה לצד זה, כלומר מוך מפוץ.

חסרונות:

- סביבת פיתוח גבוהה לאט
- יש תוספים שמצריכים התקנתם וכיבויים נוספים, לא כולל מגעיהם עם ההתקנה הראשונה כברירת מחדל.

דיוון ומסקנות

צד ל��וח:

פיתוח SmartPark בצד הלוקה יקודד בשפות: *HTML5, CSS3, JavaScript*. לאחר סקירת שפות אלו, עולה כי השימוש של ארבעתן יהיה המוצלח ביותר אשר יבטיח את התוצאה הרצויה מהסיבות הבאות: טכנולוגיות עיצוב מתקדמות כמו *CSS3* וספריות עיצוב מבוססות *JavaScript* כמו *Bootstrap* ו- *jQueryUI*. אפשרו לייצור ממשק משתמש אינטראקטיבי ועיצוב דינامي מותאם-פלטפורמה (רسفונסיבי).

ליבת המערכת בצד הלוקה תפותח באמצעות *JavaScript*. שימוש בשפה זו יאפשר הרצת מגוון לולאות מהירות ומגוון אפשרויות לטיפול יעיל במערכות ומבנה נתונים שונים המבונים בשפה וכן ביצוע פקודות אРИתמטיות מהירות. זאת ועוד, *JavaScript* מאפשר שימוש בספריות ליבה מתקדמות כמו- *D3, Angular* ו- *jQuery* המאפשרות שימושה בתהליכי העבודה ומונעות כפילות קוד. יתרון נוסף של *JavaScript* על פניה האחרות שנבדקו (*C#, Android*) הינו פשוטות הבדיקה מול הממשק (*API*) של גוגל-מפות (*Google Maps*) והמשק של הפלטפורמה היברידית- *iOnic*. חשוב לציין, שההתמחות שלנו מתחלקת באופן צ�ה שני מתקנים מתחמים בפיתוח *webApp* והפתחה הנוסף מתמחה בפיתוח *mobileApp*. השימוש בין שני המסלולים הינו שילוב מנצח והבחירה בשילוב שפות תכונות אלו, ללא ספק, מקדשת את המטרה.

צד שרת:

בסקירת השפות לפיתוח הצד השרת+ בסיס נתונים (*DB*) נבדק מספר אפשרויות רלוונטיות. לכל אותן שפות יש יתרונות וחסרונות. כולן שפות הנמצאות בשימוש בפיתוח אפליקציות בימינו וככלן יכולות לענות על דרישות המערכת המתוכננת. השאלה הנשאלת היא איזו מבין השפות שתוארו לעיל היא היעילה והמתאימה ביותר לקודד את *SmartPark*?

השפה הנבחרת לפיתוח השרת- *Node.js*. יתרונותיה הבולטים של שפה זו היא אופן הריצה שלה- אסינכרונית וקלות ופשטות השפה שכן היא שפה פונקציונלית המושתת על עקרונות שפת *JavaScript*. התאמת המשלמת לעובדה מול הצד הלוקה שגם הוא יהיה כתוב ב- *JavaScript* מאפשר גמישות יתרה בממשק בין השרת ללוקה.

Node.js מכילה מגוון רב של ספריות- חבילות (*packages*) הזמינים לשימוש מיידי ומקלים על השימוש בשפה. ספריות כגון: *lodash, Asinc, D3, Promise* ועוד.. מונעות כפילות קוד ומסייעות לעובדה נוחה מול המנגנון הנוקשה של השפה (אסינכרוניות).

Node.js עובד בצורה קלה ומהירה מול ה-*API* של *mongoDB* תחת סביבת הענן של *mLab* המאפשרת את שירת הנתונים במחוזותיה. *mDB* מציע מסד נתונים לא-RELACIONAL (NoSQL) חדשני שעולה קנה אחד עם *SmartPark* זה מבחינת סוג המידע הנשמר והן מבחן אופי העבודה מולו.

נוסף על כן, מהירות ויעילות בחיפוש הודות לאנדקס נתונים (*Index*) ושיטות כמו *Sharded Files* (*Collections*) המאפשרות שלילת נתונים מהירה, ביצוע אגרגציה (*Aggregation*) ומיפוי קל של נתונים מתוך טווך נתונים אחד. חשוב לציין ש-*mongoDB* תומך בחיפוש בינהי (*Binary Search*) יתרון נוסף המאפשר לבצע חיפוש מהיר בעלות נמוכה מאוד.

פלטפורמה היברידית (משולבת):

פיתוח אפליקציה בשפת ג'אווה (*Java*) לפلتפורמת אנדרואיד אומנם מורכבת לפיתוח אך אפליקציה המפותחת בשיטה זו יוצרת מוצר אינטראקטיבי בעל יכולות מרישומות ומהירות. לעומת זאת האפליקציה היברידית מהוות פתרון קל ופשוט לפיתוח בזכות השימוש של טכנולוגיות שפות פיתוח אינטראנט- *webApp*. עם זאת, חשוב לציין כי היא רצה פחות מהר מאשר אפליקציה המköנדת באנדרואיד. למרות העובדה זו בכל זאת בחרנו לפתח את *SmartPark* בטכנולוגיה היברידית באמצעות *iOnic* הן בגלל פשטות הפלטפורמה והן השימוש המנצח בין חברי הצוות (כפי שהואר בסעיף קודם) המגיעים מהתחומיות שונות התואמות בדיקות את הפלטפורמה היברידית: *webApp dev & mobileApp dev*.

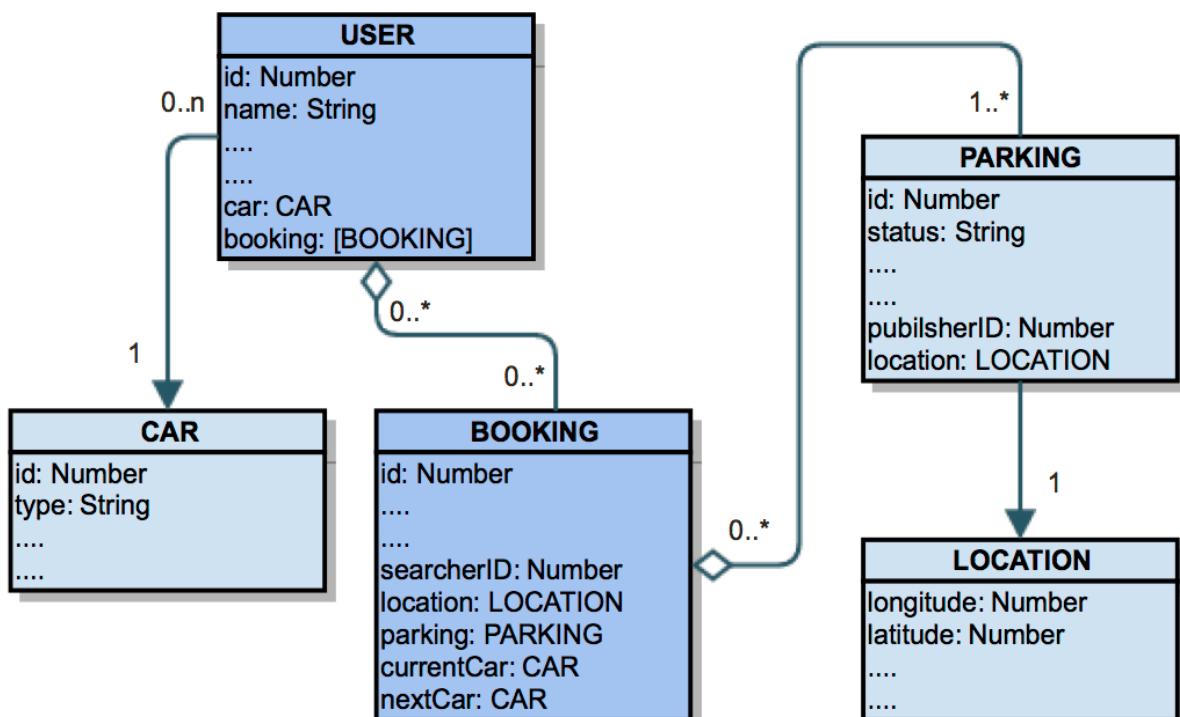
סביבת הפיתוח: ככל, לכל מפתח יש את הכליל לפיתוח שהכי נכון לו לעבוד איתו.

בהתיחס לסקירה הספרותית בנושא זה, סביבת הפיתוח *sublime* היא סביבת פיתוח נוחה ומקלה על המפתח במקוד עליון הקבצים הרלוונטיים עליהם הוא עובד. זאת ועוד חווית כתיבת הקוד ושיפור הטקסט כמו שגיאות כתיב מאפשרת לתוכנת ליעל את העבודה.

סביבת הפיתוח *visual studio* היא סביבת פיתוח שפচות מתאימה לפיתוח *app web* ובנוסח היא תוכנה איטית יותר ביחס ל- *sublime* שהיא כלי פיתוח טוב ומהיר לפיתוח *web app*.
לכן החלטנו שסביבת הפיתוח המתאימה ביותר לשוג האפליקציה שתפותה היא *sublime*.

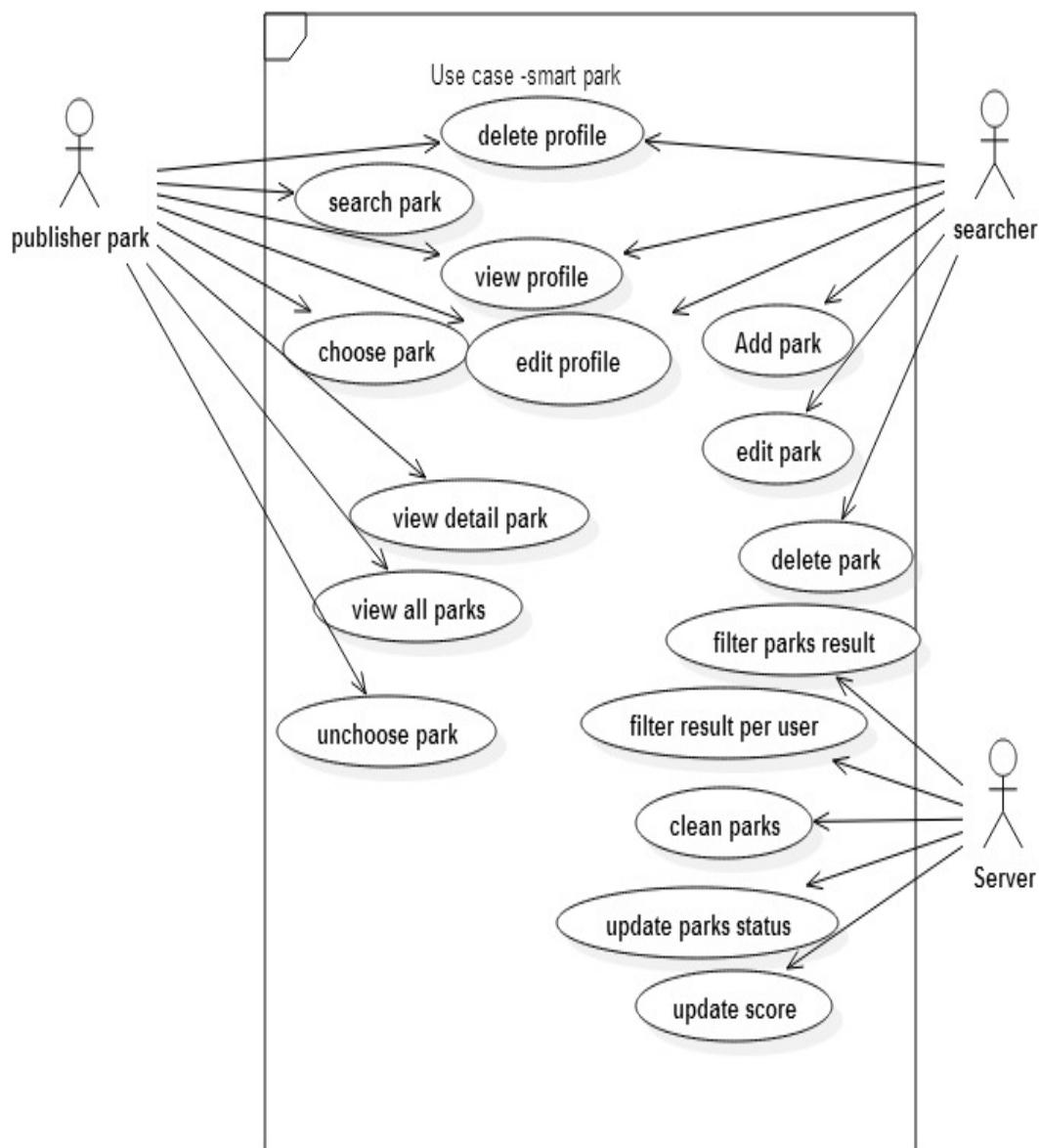
תיכון המערכת (Design)

Structural Design

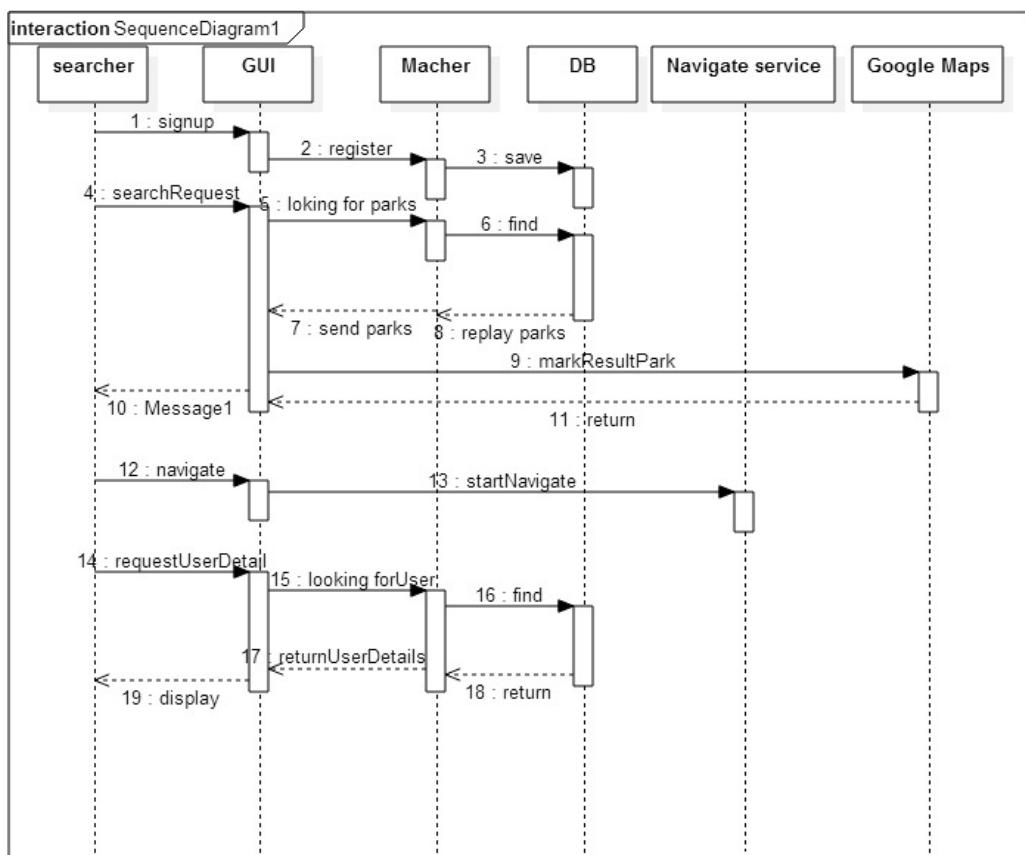
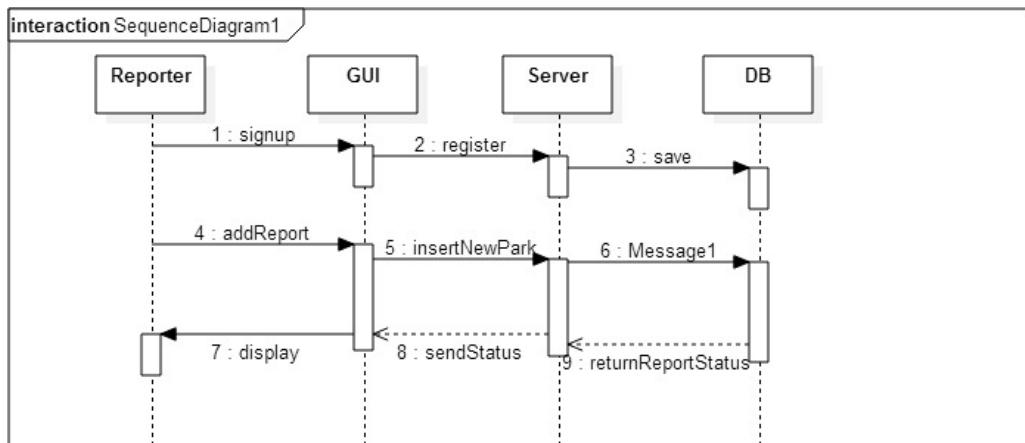


Interactions Design

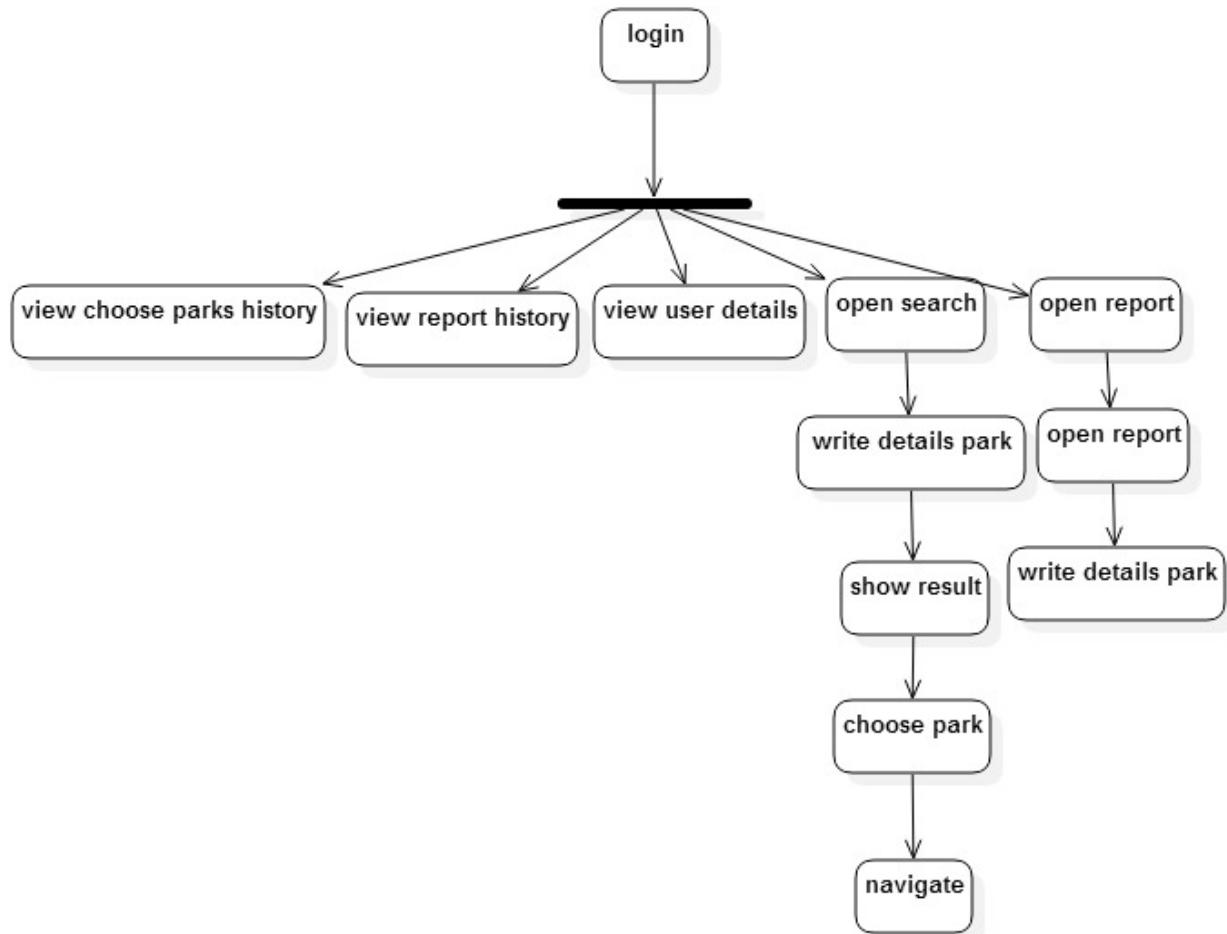
- **Use Cases:**



- **Sequence Diagram**

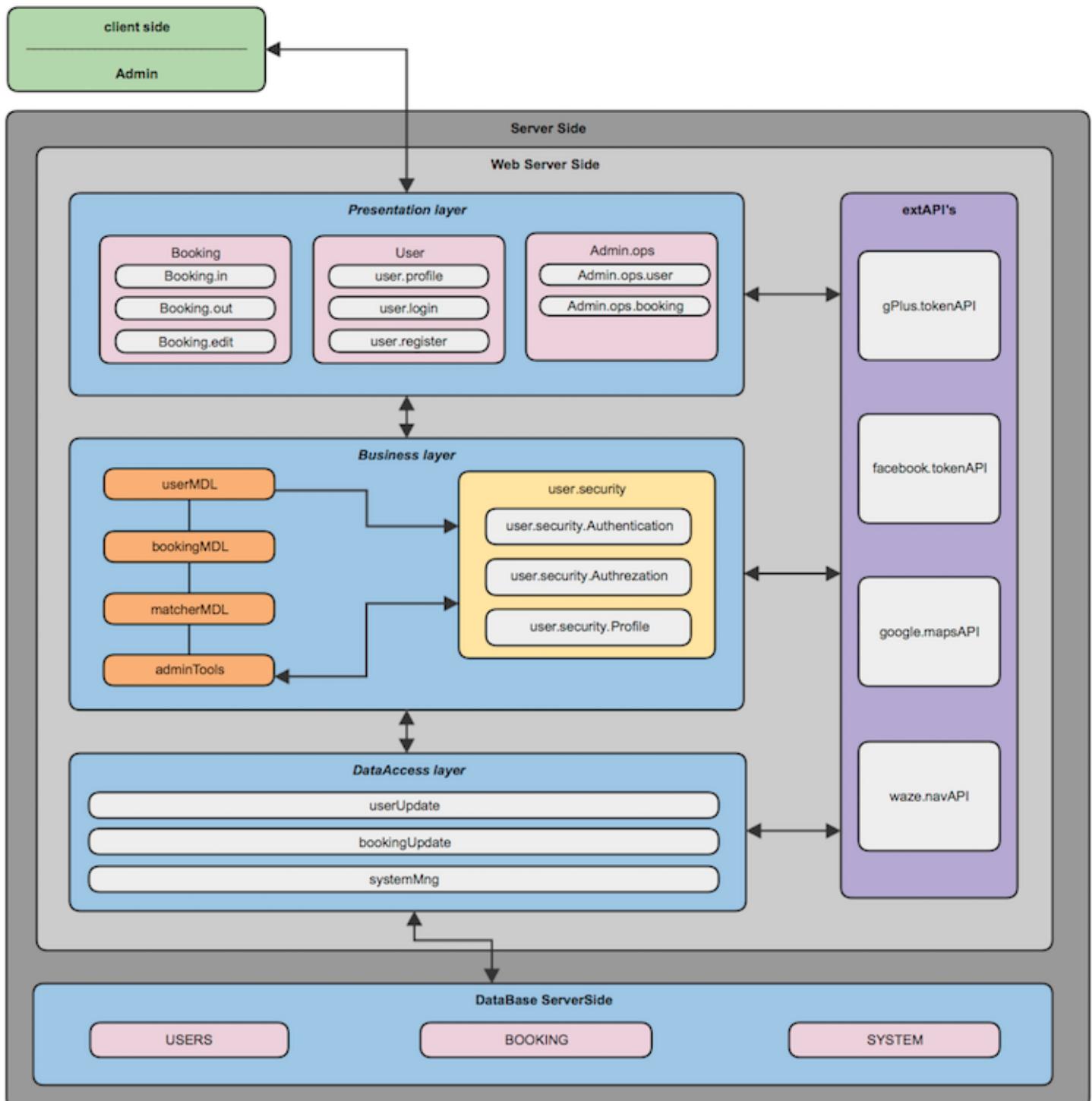


- **Activity Diagram**

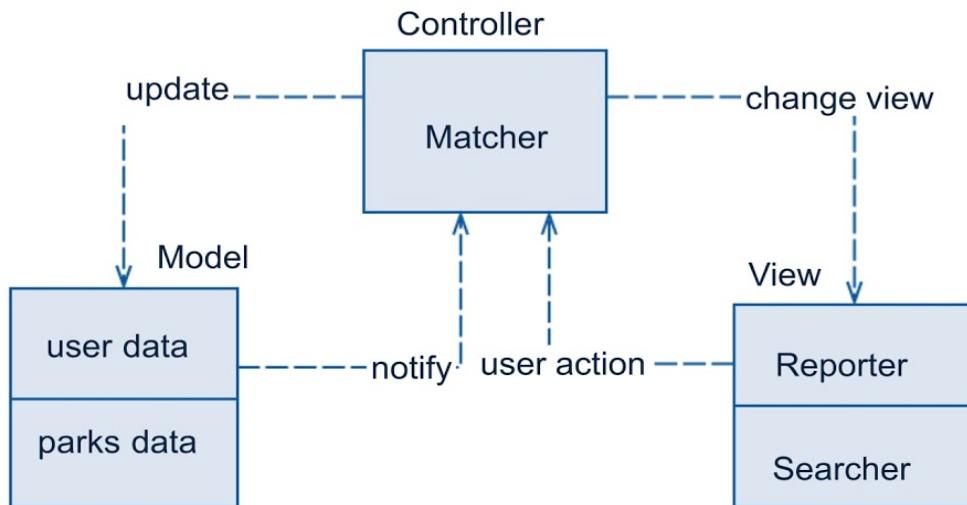


Software Architecture Pattern

- Three-tier: Data, Logic and Presentation Layers



- **MVC: Model, View, Controller structure**



ניהול איכויות התוכנה

שיתוף הקוד בין חברי הצוות יעשה באמצעות פלטפורמת GITHUB - עבור הפרויקט יפותח Repository משותף לכל חברי הקבוצה.

- **תיעוד באגים:** כל GANG יתעדר כ-issue ב-Github. הנושא-hueIsI יכול את תמצית הבאג ולאיזה חלק במערכת הוא קשור. בגוף-hueIsI יהיה מפורט הבאג כולל תיאור באילו מקרים הבאג הופיע.
- **חלוקת באגים לאנשי הצוות:** כל מתכנת יהיה אחראי על הבאגים בתחום הקידוד שלו במערכת. אך עם זאת, אנו עובדים בשיתוף פעולה ולכן רוצים שלכל GANG יהיה תיעוד במערכת וכן אחד יוכל לעזור לשני. כל מתכנת אצלנו בקבוצה מכיר את החזוקות של חברי הצוות, ולכן ניתן לשיך GANG מסוים לחבר צוות מסוים במידה והוא בקיא יותר בתחום בו קיים GANG. נעשה זאת ע"י mention ב-issue ונציין את שם חבר הצוות הרלוונטי. מעבר לכך, בכל פגישה שבועית של חברי הצוות- נעבור על כל ה-issue-im הפתוחים ונראה כיצד אפשר לפתור אותם יחדיו.
- **מעקב אחרי תיקוני באגים:** כשבא Gang יתוקן-hueIsI ייסגר. בנוסף עבור GANG שיפתח מחדש יבוצע reopen. עבור GANG יוגדר מי מטפל בו. בפגישה השבועית של כל חברי הצוות נעבור על הרשימה כך יהיה מעקב שבועי על באגים שנשארו פתוחים GANG ובאGs לא ישאר פתווח מעל שבוע.

תיאור מבנים (Classes)

As our project coded in JavaScript which is a functional language code and not OOP-driven language, this field doesn't match our project.
 The project will use in objects structure similaire as the DB's objects structure.
 Therefore, the main objects will be:

- **USER** ={
 id: Number,
 name: String,
 Email: String,
 phone: Number,
 ranking: Number,
 phoneInvisible: Boolean,
 defaultAddress: String,
 token: String,
 img: url(String),
 smarties: Number,
 handicapped: Boolean,
 car: {CAR},
 bookings: {[BOOKING]}
 }
- **CAR** ={
 id: Number,
 type: String,
 color: String,
 cube_size: Number
 }
- **LOCATION** ={
 country: String,
 city: String,
 street: String,
 number: Number,
 longitude: Number,
 latitude: Number
 }
- **PARKING** ={
 id: Number,
 date: Date,
 time: Date.time,
 published: Date,
 status: String,
 occupied: Boolean,
 location: {LOCATION},
 waiting: Number,
 handicapped: Boolean,
 descriptionTXT: String,
 descriptionIMG: url(String),
 repeat: Boolean,
 interval: Date,
 cube_size: Number,
 pubilsherID: Number,
 }
- **BOOKING** ={
 id: Number,
 DOI: Date,
 TOI: Date.time,
 date: Date,
 time: Date.time,
 radius: Number,
 location: {LOCATION},
 comments: String,
 searcherID: Number,
 parking: {PARKING},
 currentCar: {CAR},
 nextCar: {CAR}
 }

הגדרת מישימות ותפקידים

:Base •

- חילוקה לתקיות development, production, server, client
- פיתוח רפואיטורי משותף בגיט והגדרת אבני דרך, בעיות ומישימות.
- ייצירת פרויקט ב-herokuApp.com
- ייצירת פרויקט ב- goDaddy / ftp cloud
- הקמת db והגדרת קולקשיין ב-mLab
- ביצוע בדיקות וטיפול בשגיאות

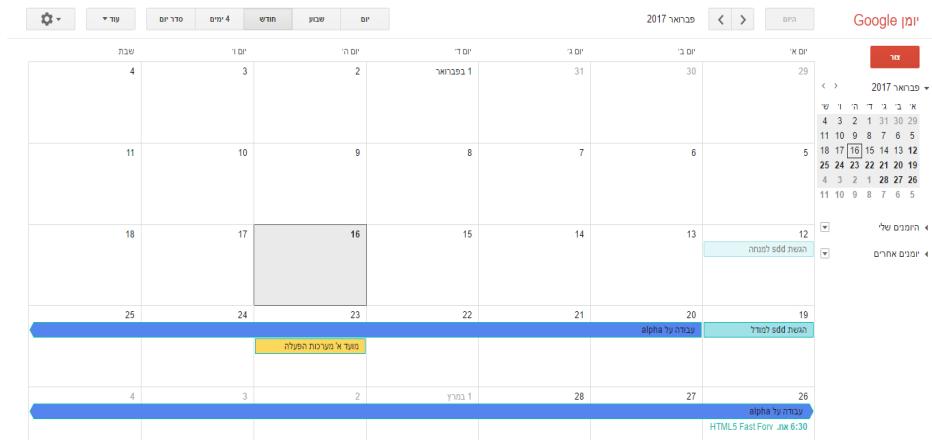
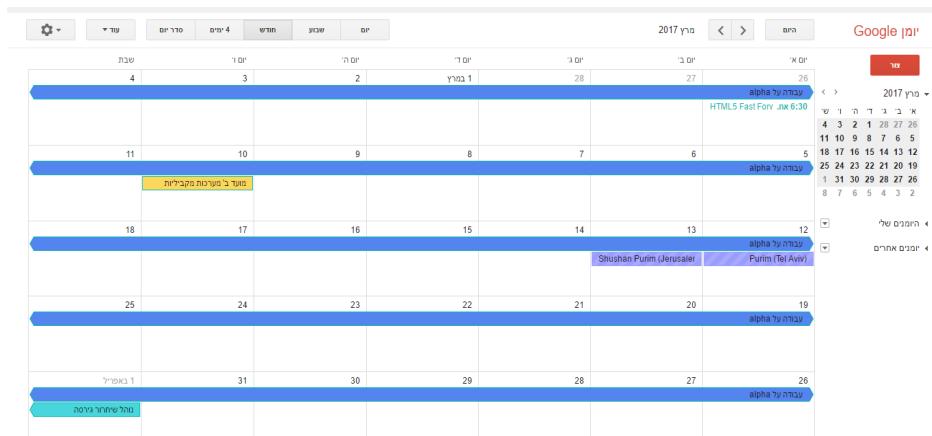
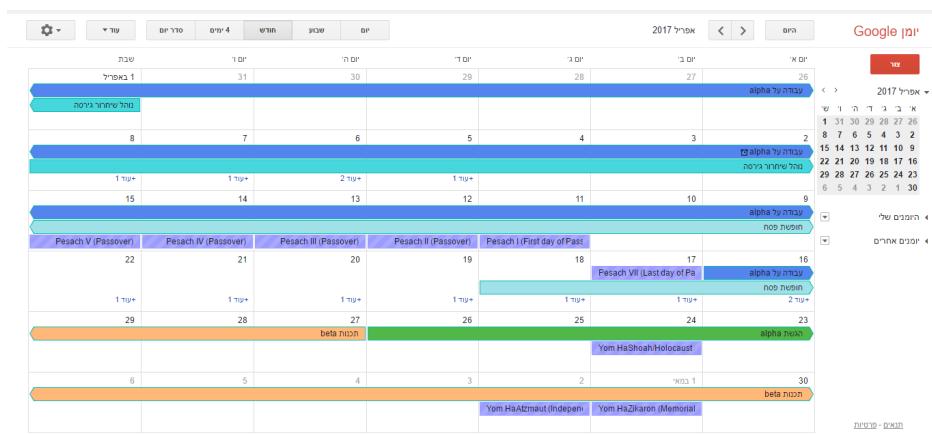
:Client side •

- screens: (בנייה כל המסכים)
- ייצירת מסכים flow MAIN
- ייצירת מסכים flow IN
- ייצירת מסכים flow OUT
- ייצירת מסכים flow Menu
- ייצירת מסכים flow History
- :login/register
- Google
- Facebook
- Primitive way

- Waze API (עבודה עם API של ניוט)
- Google API (עבודה עם המפות - Google maps API)
- לצור marks על המפה
- למחוק marks על המפה
- לתרגם מהרוואות לקוואורדיינטות (Longitude/ Latitude)

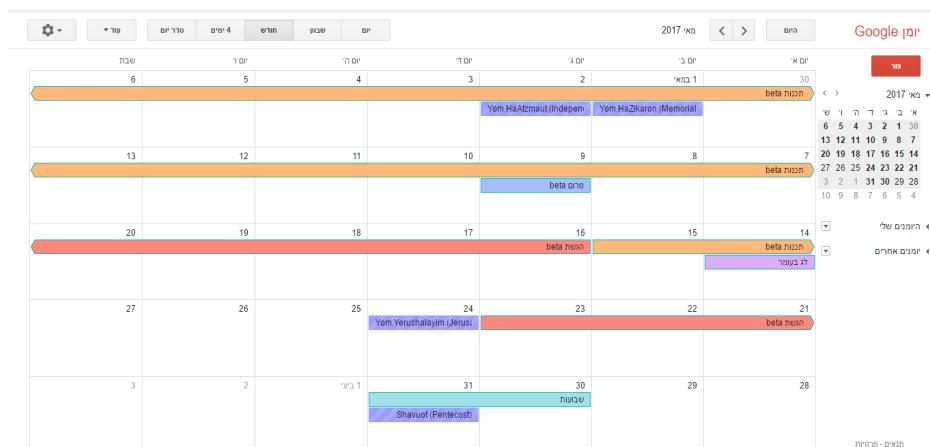
:Server side •

- הקמת Express Server + Routs
- מימוש מודולים ופונקציונליות לכל מודול לפי דרישות פונקציונליות (FRS):
- User (הציגה של נתונים שמאימים ממסד הנתונים. דוגמה: צפייה בדיוחים שלי, צפייה בפרטיו החנינה שנבחרה)
- Booking (הכנסת דיוח, פיתוח חיפוש, בחירת החנינה הרצויה)
- Matcher (מודול החיפוש, ניקוד ולחיצת היד)
- Admin (מודול ניהול משתמשים)
- System (ניקוי / מחיקת חניות לא רלוונטיות, פונקציות tasks קבועות)
- Push notifications

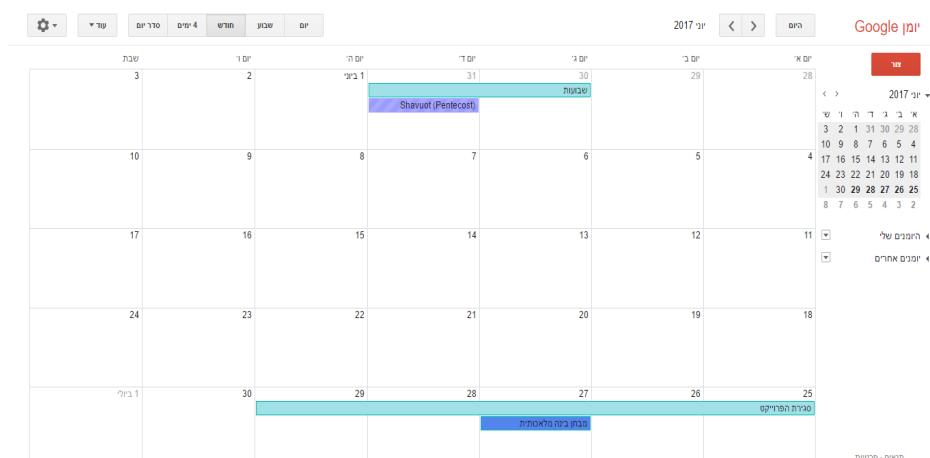
נספח ג'
לוי'ז מפורט ותכנון זמנים
פברואר:

מרץ:

אפריל:




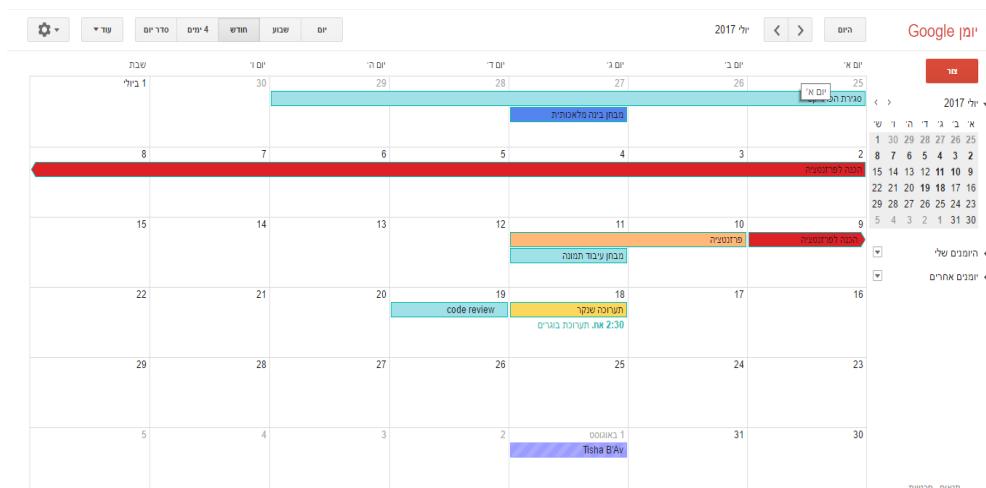
מאי:



יוני:



יולי:





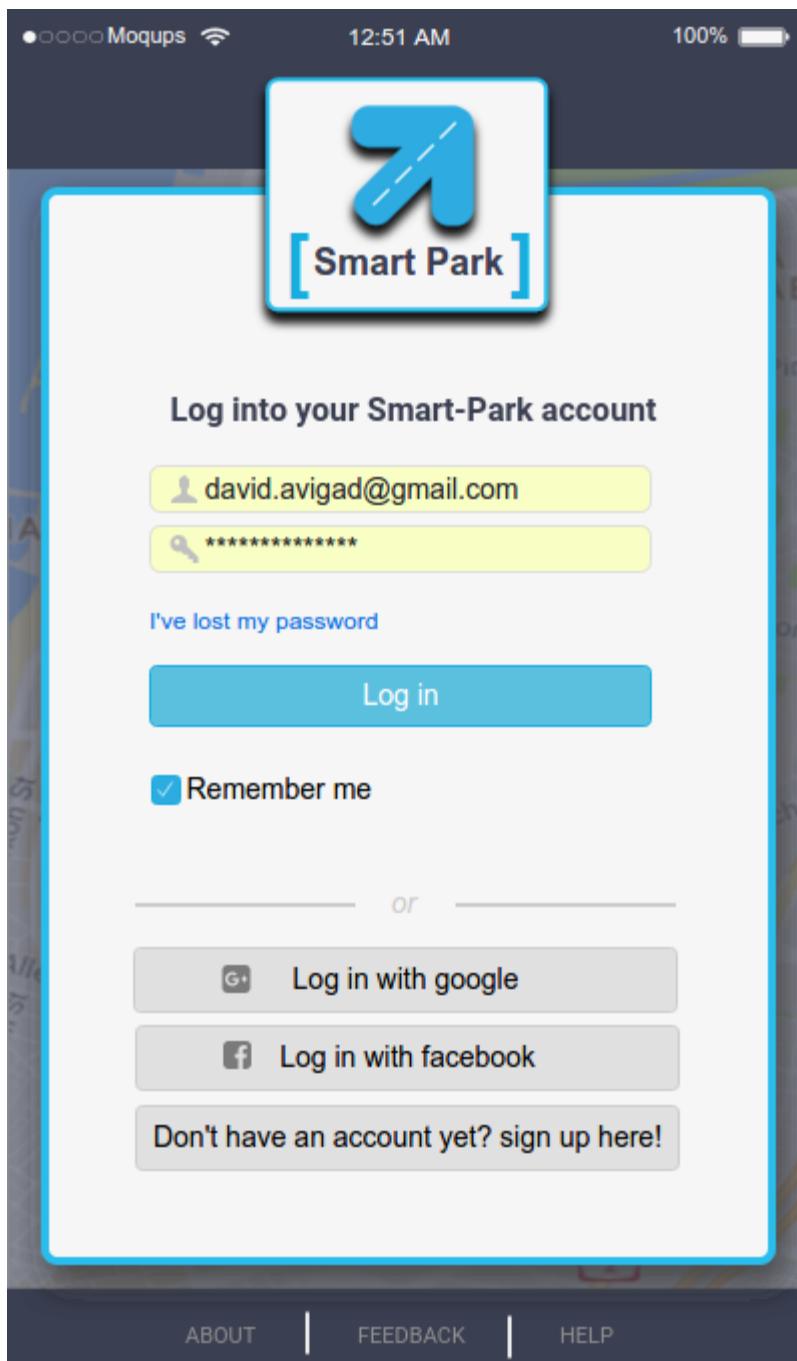
ספטמבר:

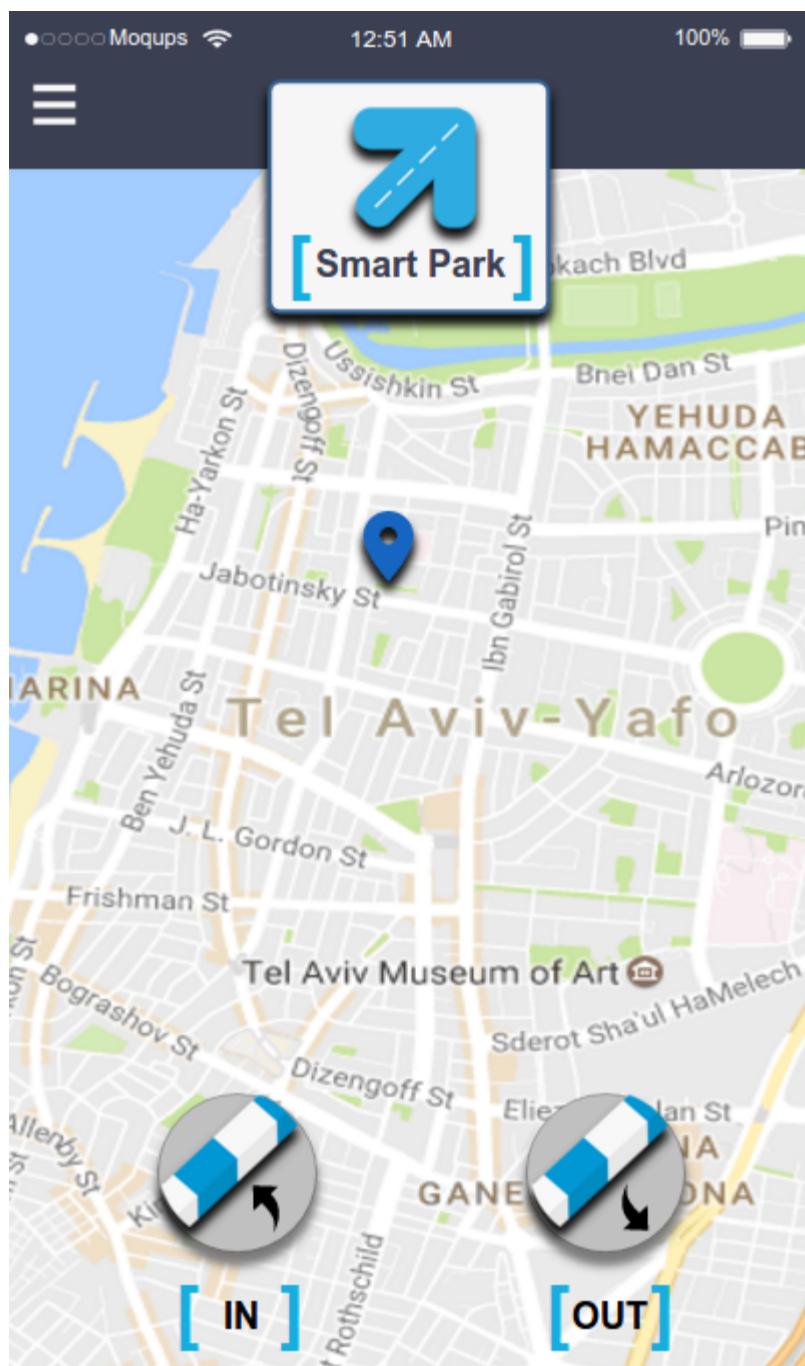
אוקטובר:

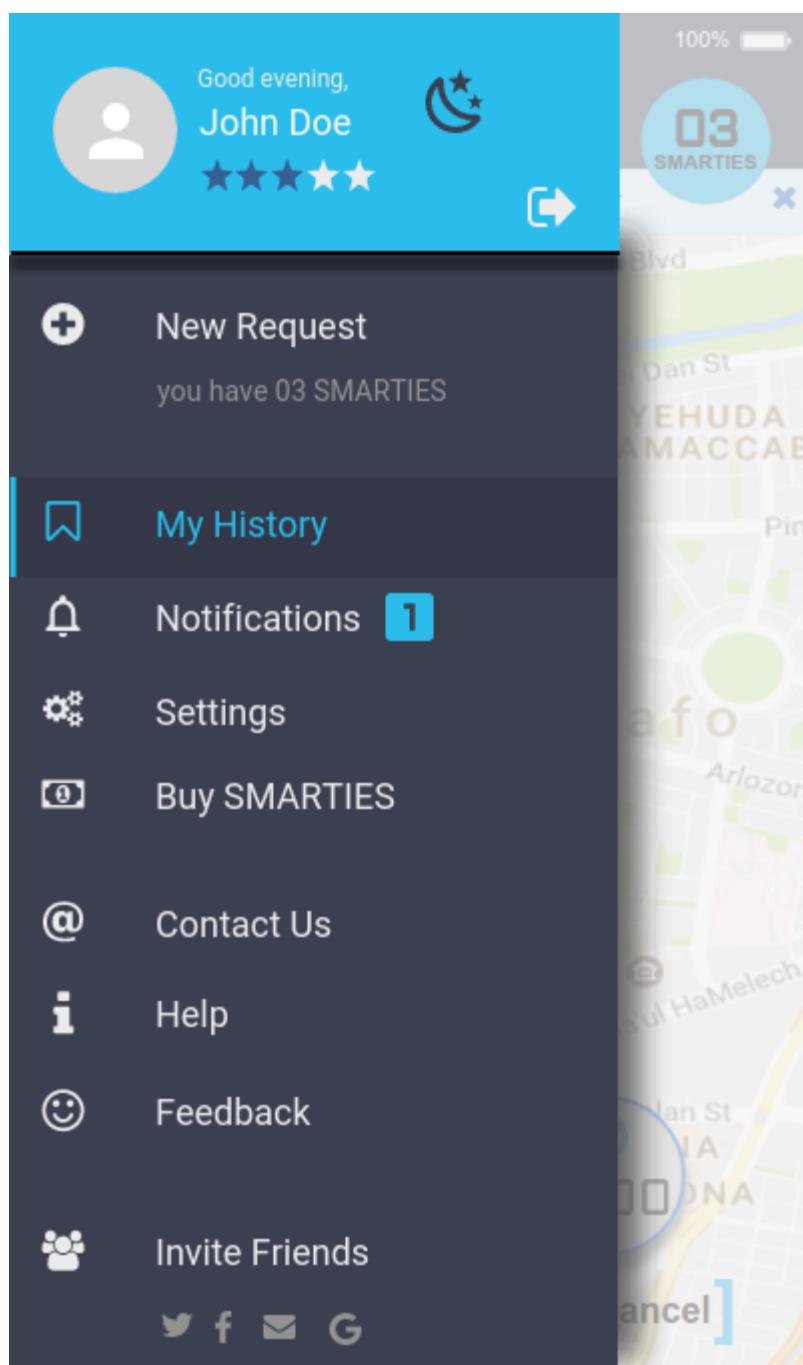


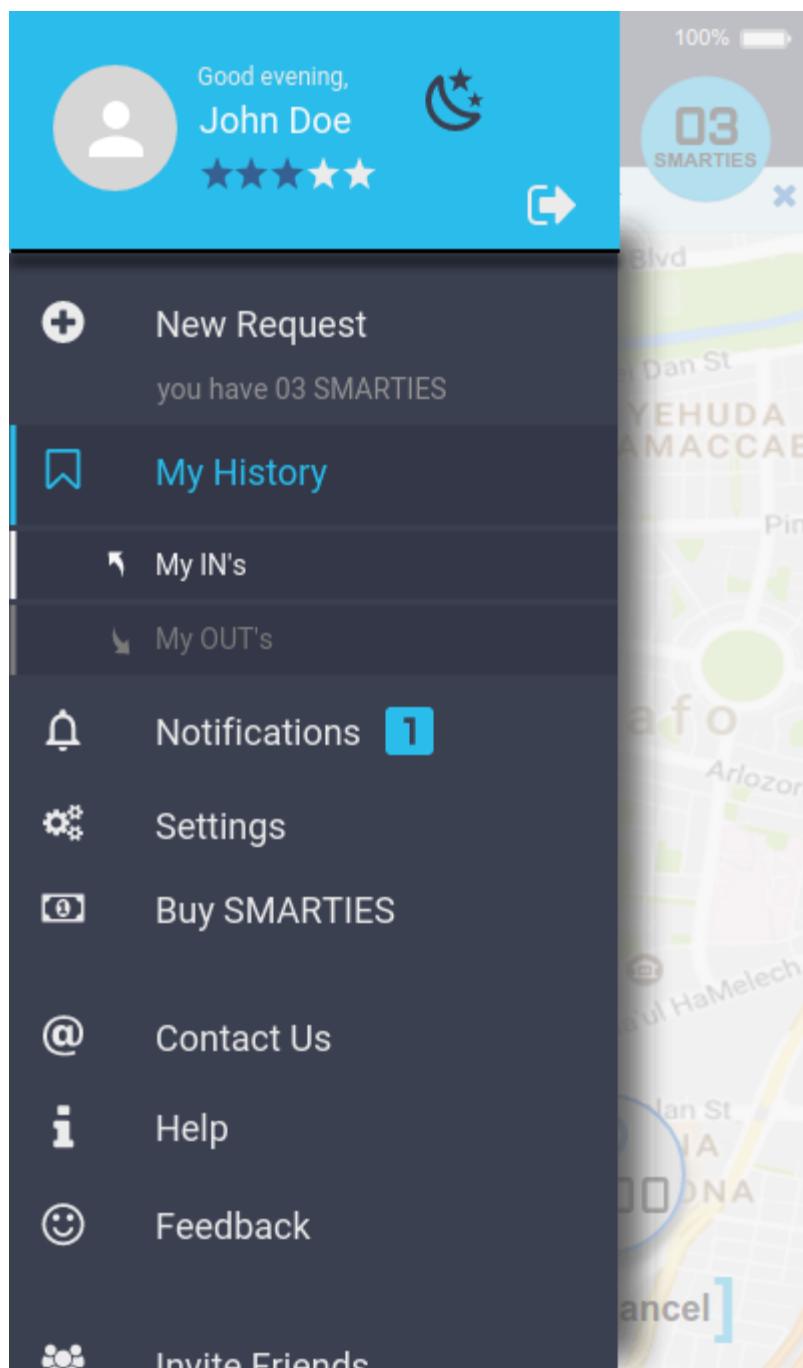
נספח ד'

מסכי מערכת מעודכנים











[Smart Park]

A screenshot of a mobile application interface for "Smart Park". The top navigation bar includes icons for signal strength, Moqups, time (12:51 AM), battery level (100%), and a menu icon. The title "My OUT's (21)" is displayed in the center. A circular badge in the top right corner shows "23 SMARTIES". The main content area lists several parking posts, each with a colored circle (green or orange) and a location name and timestamp. Some posts include a brief message.

- [Arlozorov st. #17] 2 hours ago >
09/12/17 at 15:55
Easy Parking space would be appr...
- [Arlozorov st. #17] 2 hours ago >
09/12/17 at 15:55
Easy Parking space would be appr...
- [Ben Yehuda st. #10] 3 days ago >
06/12/17 at 20:00
Please don't be late!!
- [Arlozorov st. #17] 2 hours ago >
09/12/17 at 15:55
Easy Parking space would be appr...
- [Ben Yehuda st. #10] 3 days ago >
30/11/17 at 20:00
Please don't be late!!
- [Ben Yehuda st. #10] 10 days ago >
25/11/17 at 20:00
Please don't be late!!



[Smart Park]

A screenshot of a mobile application interface for "Smart Park". The top status bar shows signal strength, 12:51 AM, and 100% battery. The title bar says "[My IN's (16)]". A circular badge in the top right corner indicates 23 SMARTIES. The main content area lists six parking notifications, each with a timestamp, location, and a short message.

- Arlozorov st. #17** 2 hours ago >
09/12/17 at 15:55
Easy Parking space would be appr...
- Arlozorov st. #17** 2 hours ago >
09/12/17 at 15:55
Easy Parking space would be appr...
- Ben Yehuda st. #10** 3 days ago >
06/12/17 at 20:00
Please don't be late!!
- Arlozorov st. #17** 2 hours ago >
09/12/17 at 15:55
Easy Parking space would be appr...
- Ben Yehuda st. #10** 3 days ago >
30/11/17 at 20:00
Please don't be late!!
- Ben Yehuda st. #10** 10 days ago >
25/11/17 at 20:00
Please don't be late!!





[Smart Park]

•○○○○ Moqups 12:51 AM 100%

[SmartPark - OUT]

03 SMARTIES

Date 09 12 17

Time 16 00

Street Rotshild Blvd. City Zone 01

No. 16

City Tel - Aviv Jaffa

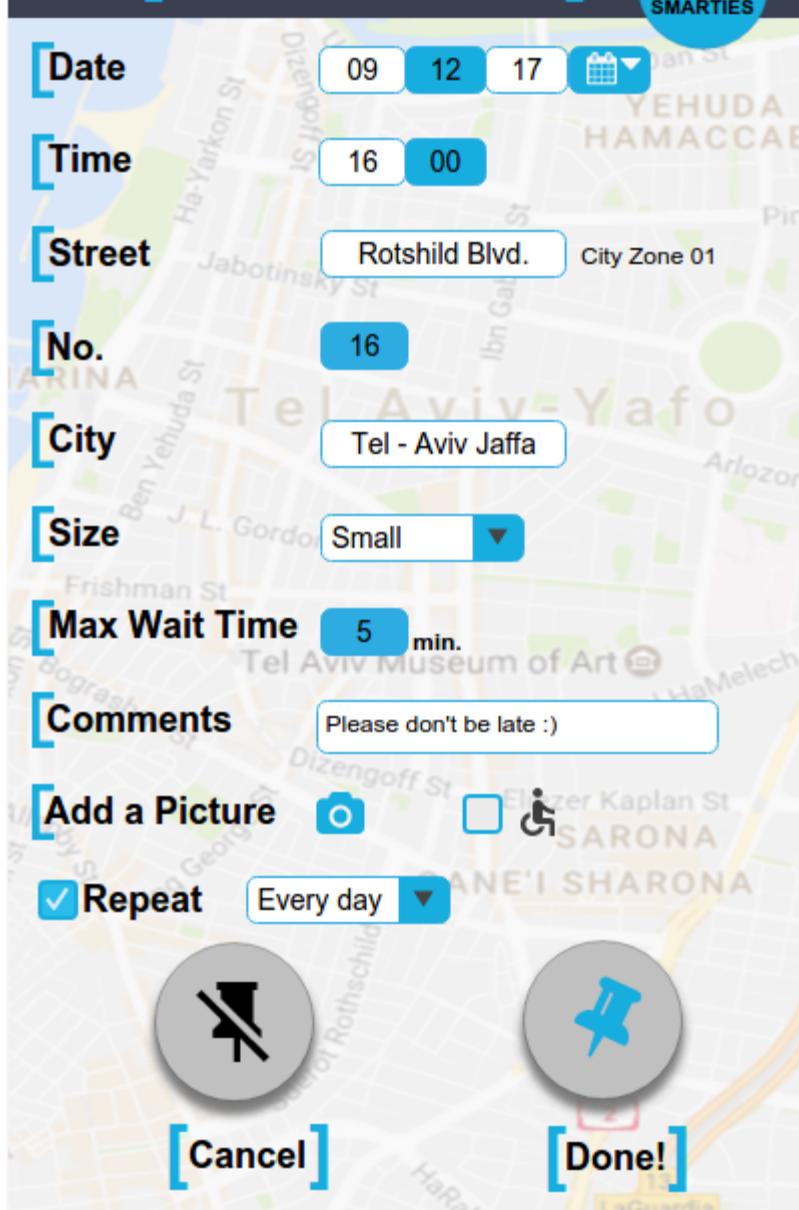
Size Small

Max Wait Time 5 min.

Comments Please don't be late :)

Add a Picture

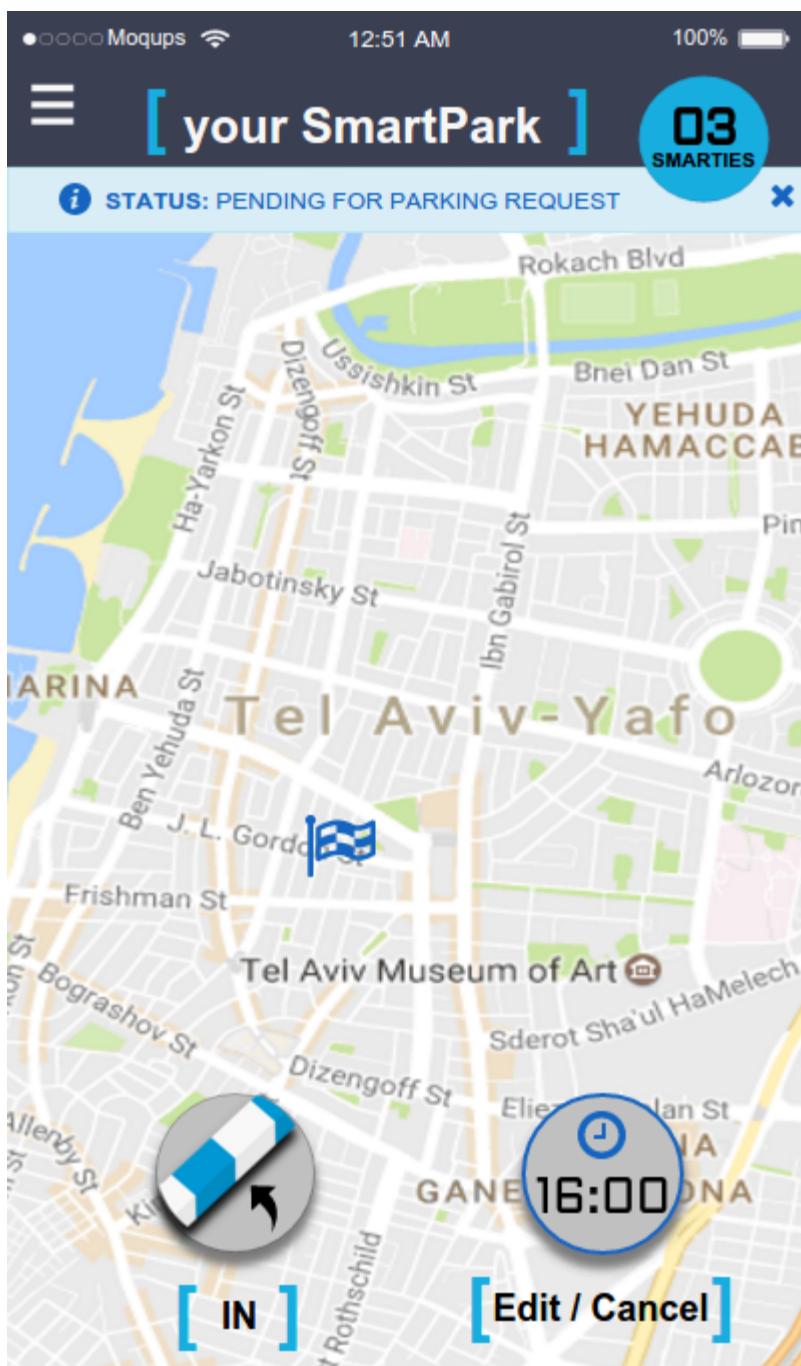
Repeat Every day



The form is a mobile application interface for booking a parking space. It includes fields for Date (09/12/17), Time (16:00), Street (Rotshild Blvd.), Number (16), City (Tel-Aviv Jaffa), Size (Small), Max Wait Time (5 minutes), Comments (Please don't be late :)), and Repeat (Every day). There are buttons for adding a picture and canceling or confirming the booking. The background shows a map of Tel Aviv-Yafo with the specific location marked.

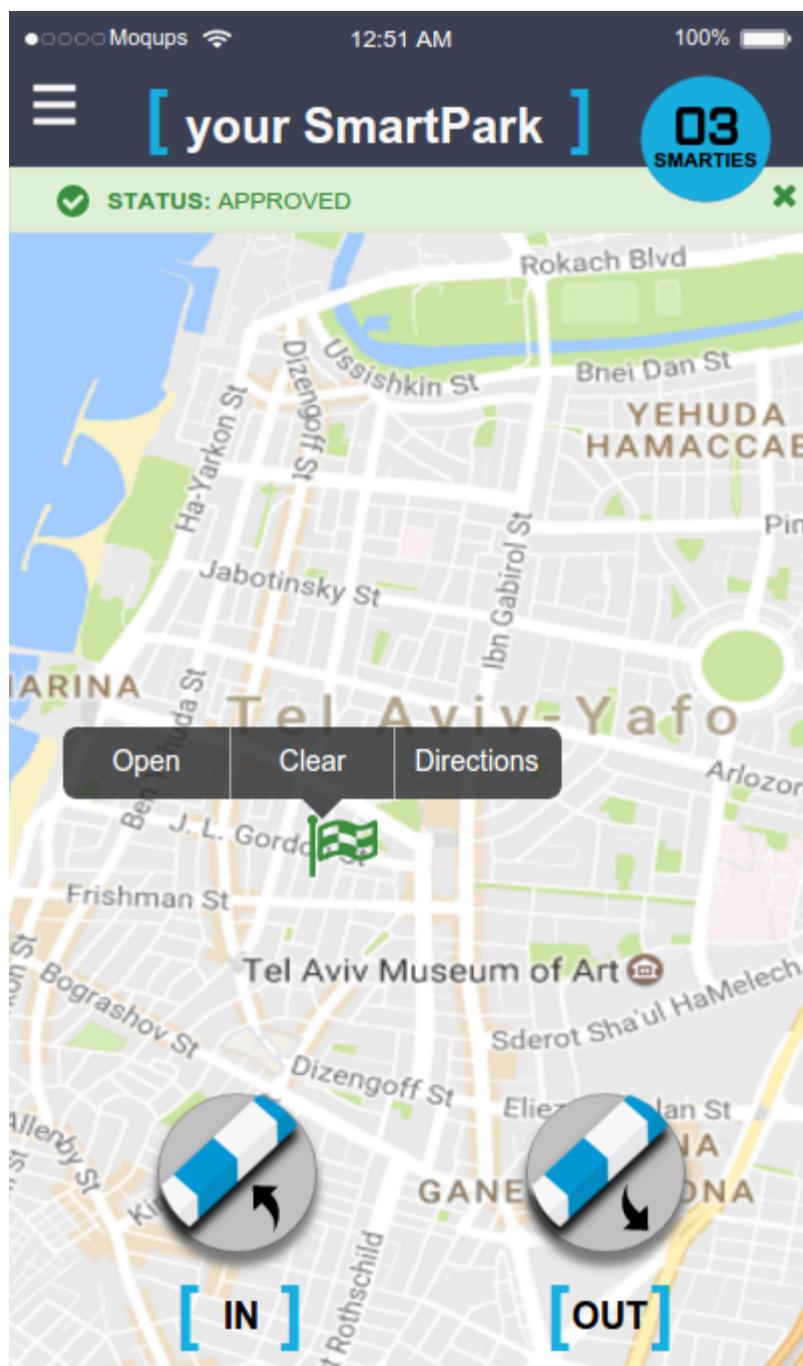


[Smart Park]



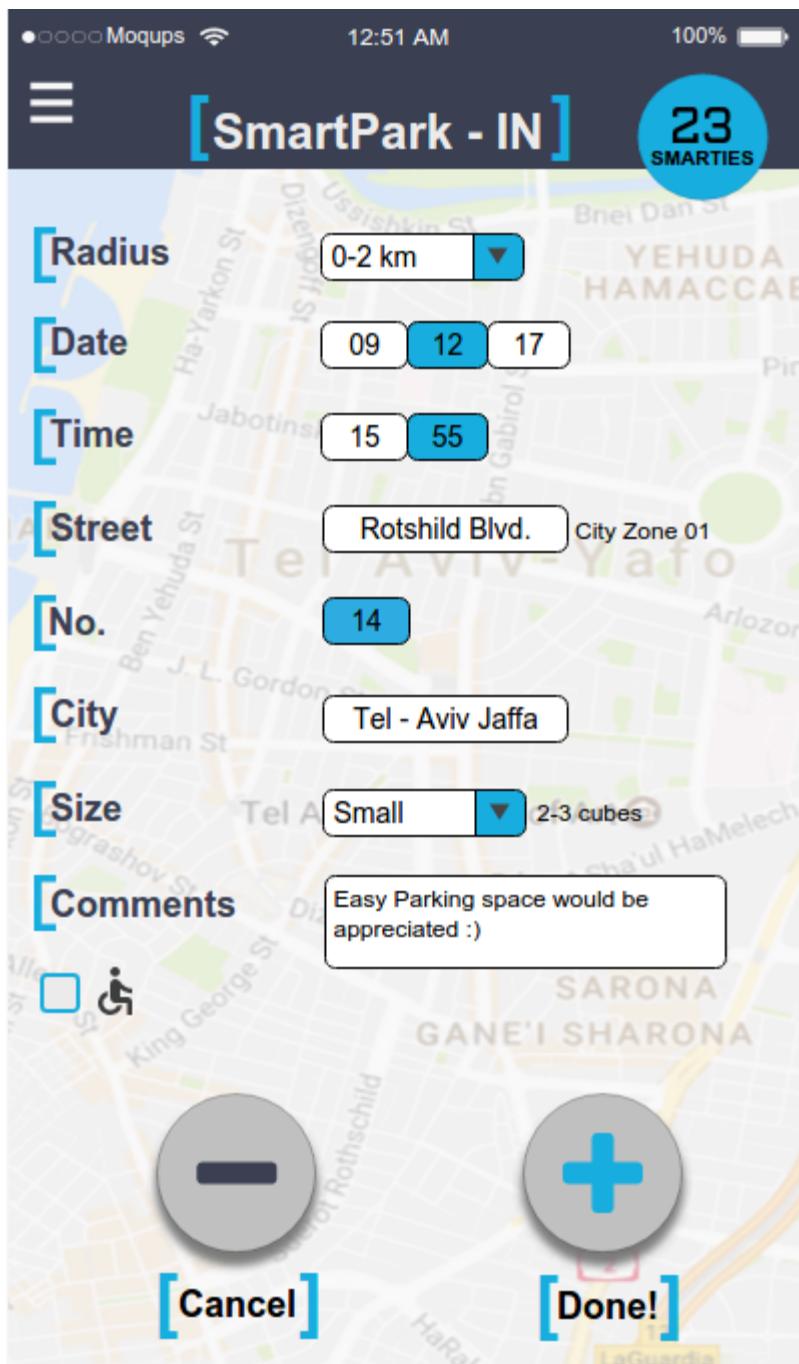


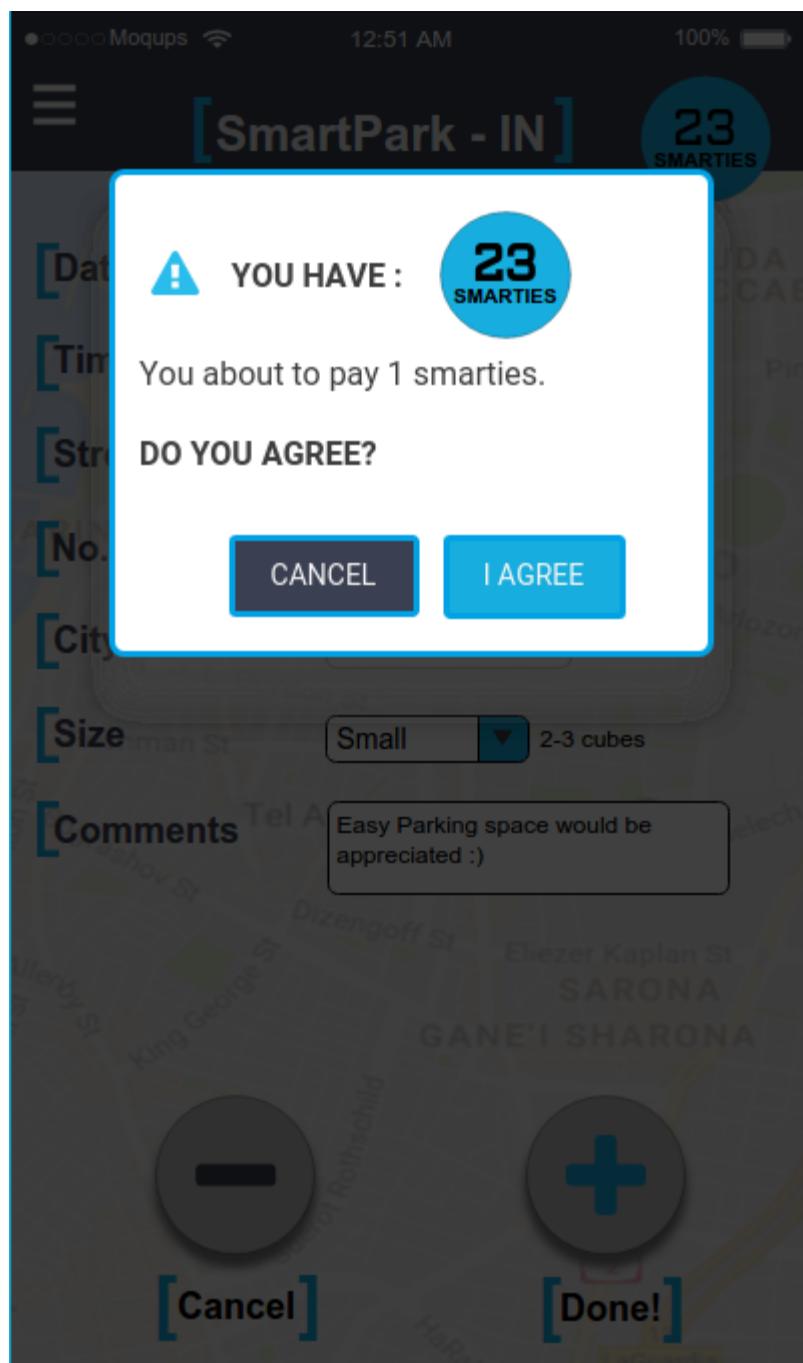






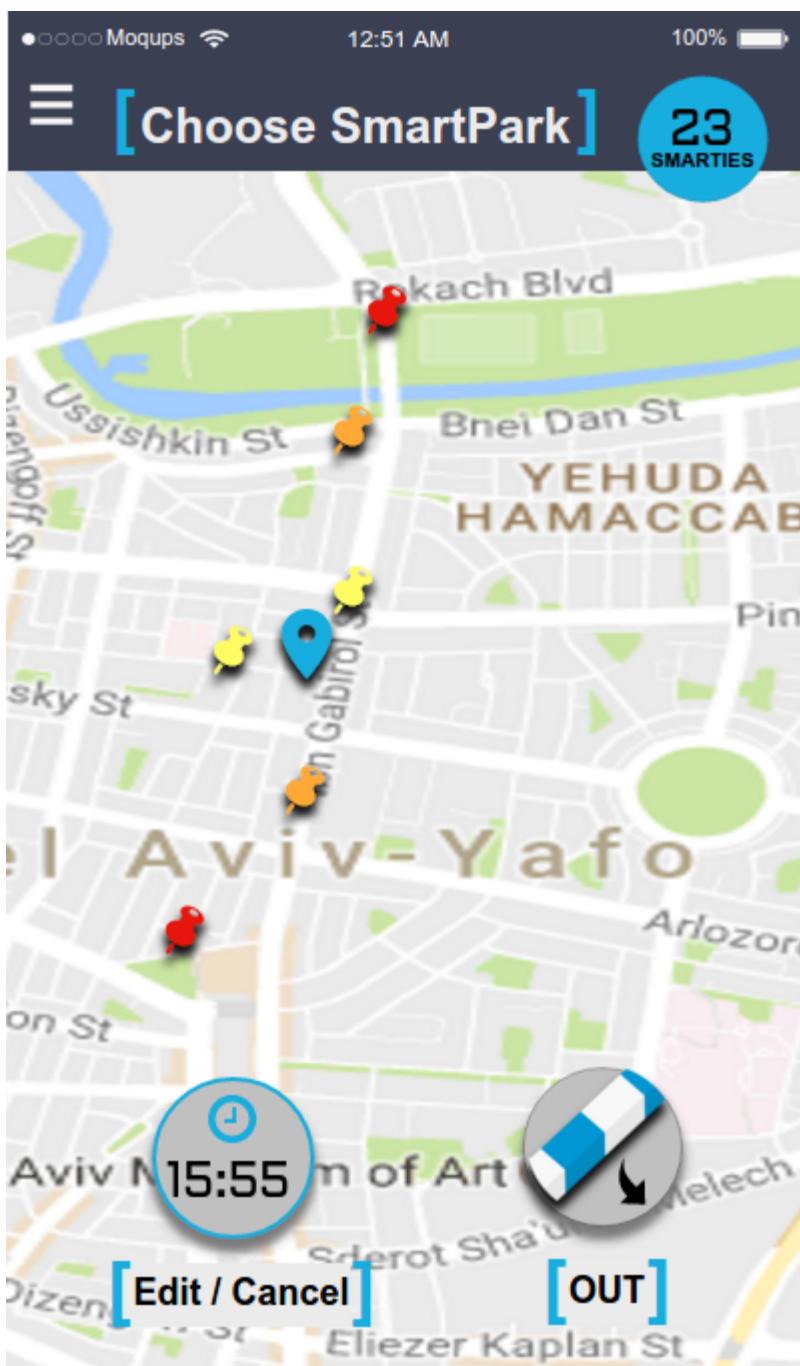
[Smart Park]

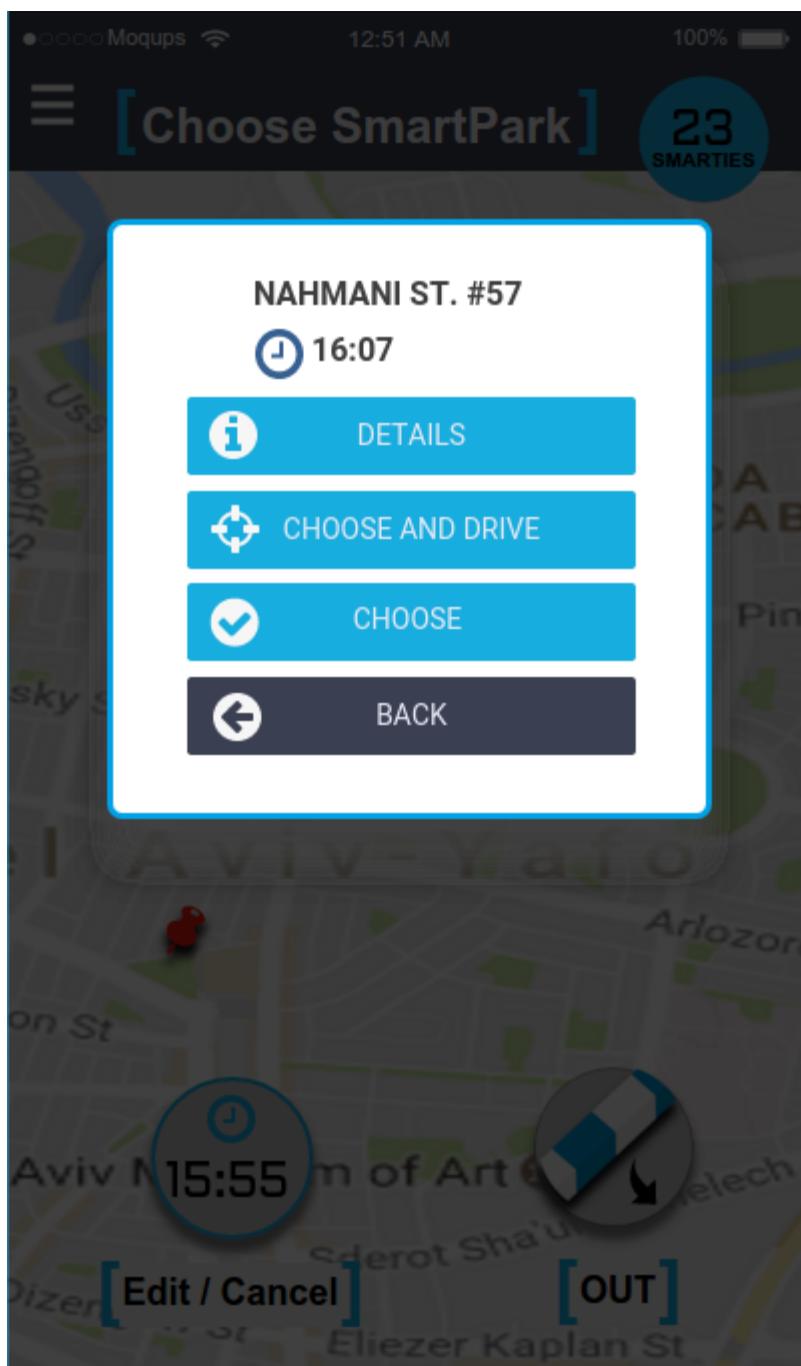






[Smart Park]







•○○○ Moqups 12:51 AM 100%

[SmartPark - Details]

23 SMARTIES

Date 09 12 17

Time 16 00

Street Nahmani st. City Zone 01

No. 57

City Tel - Aviv Jaffa

Size Medium 4-5 cubs

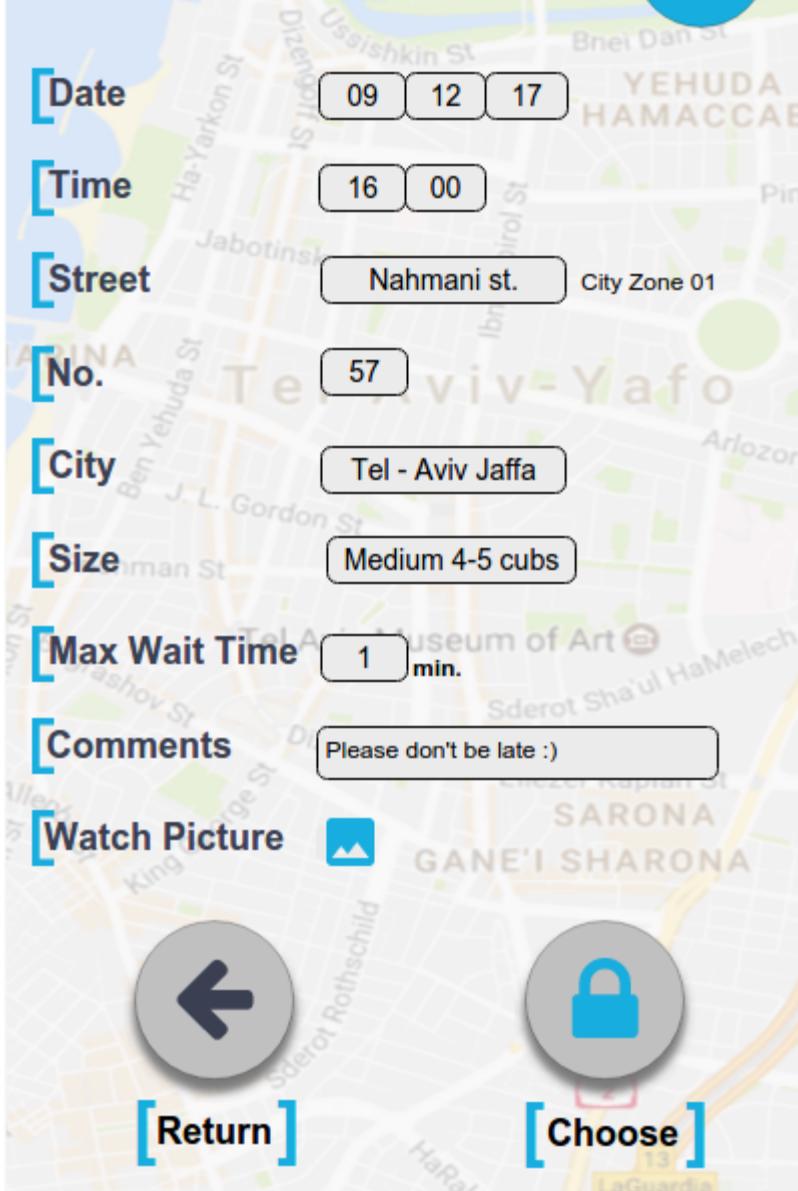
Max Wait Time 1 min.

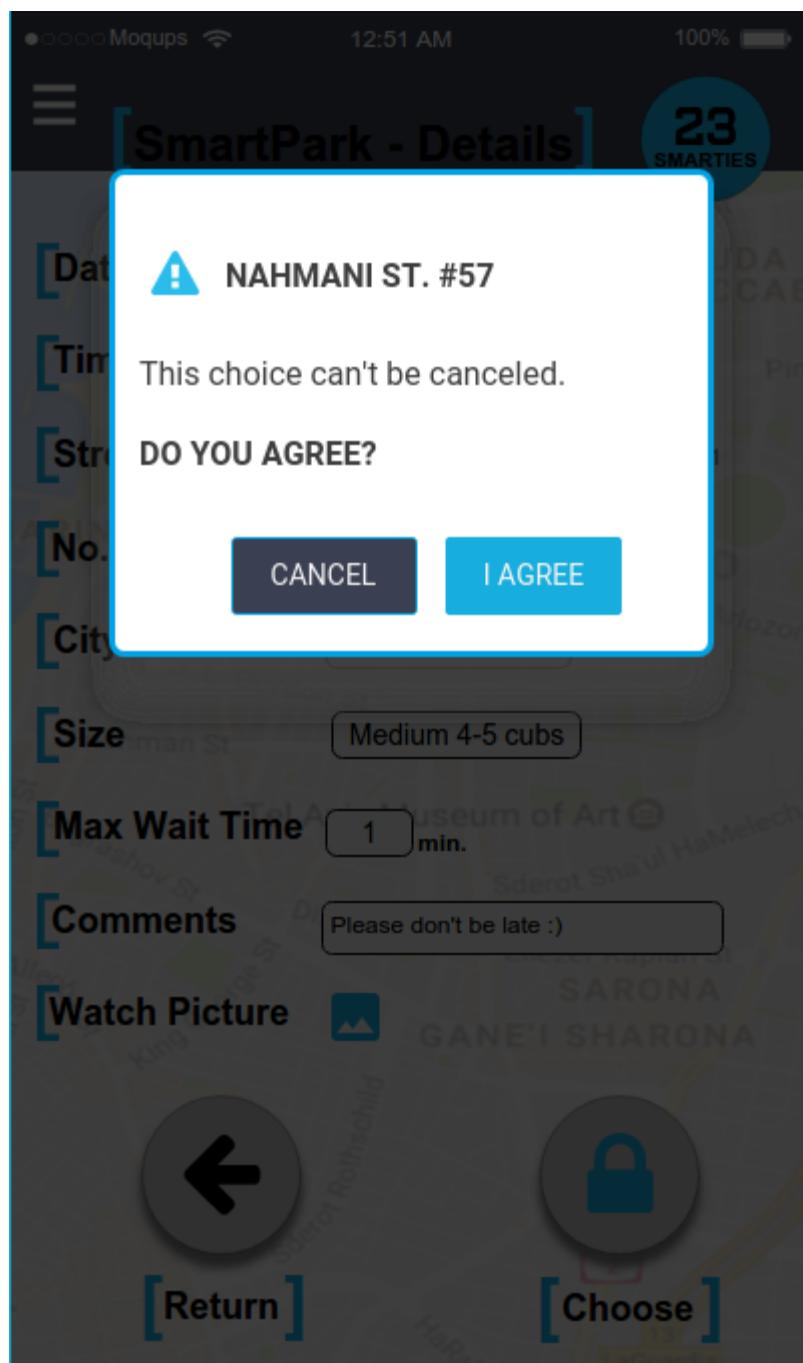
Comments Please don't be late :)

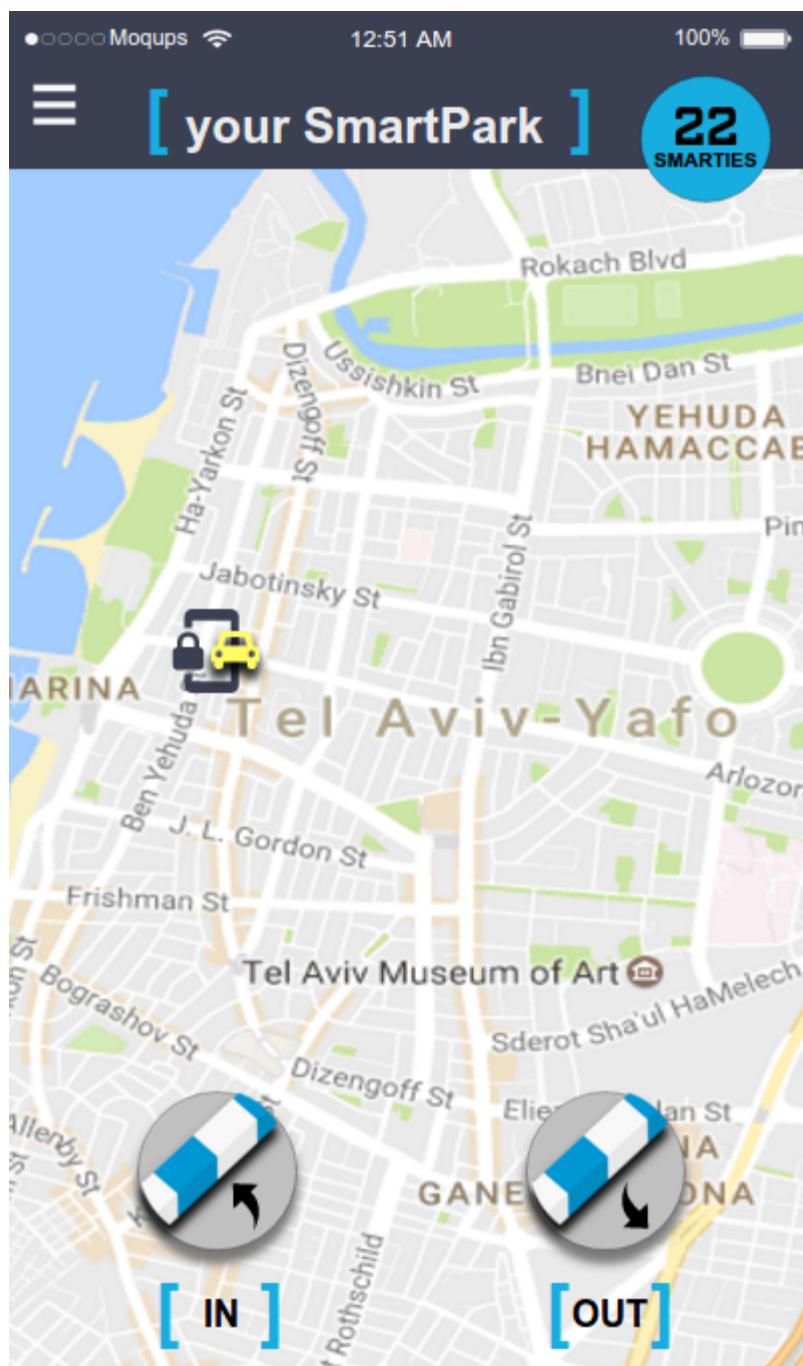
Watch Picture

Return

Choose

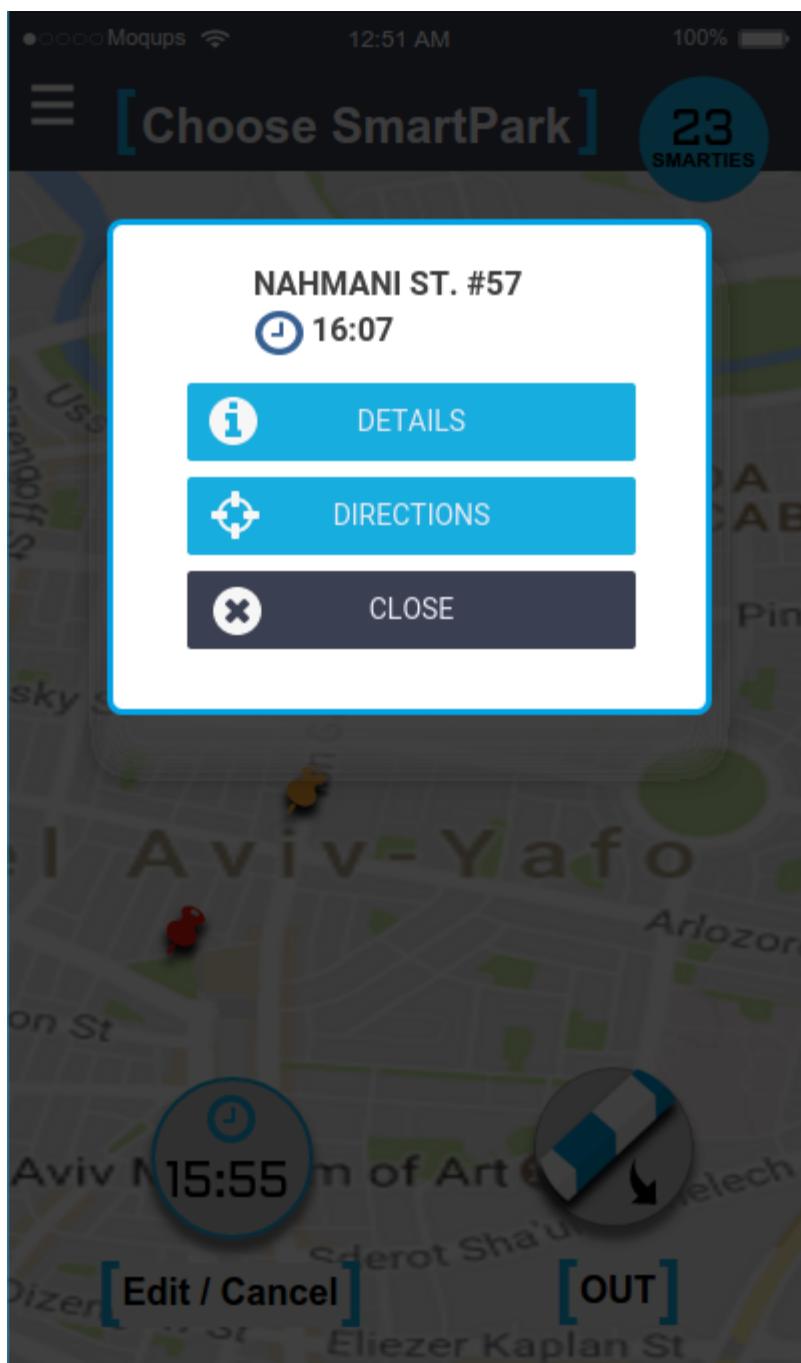








[Smart Park]





•○○○ Moqups 12:51 AM 100%

[SmartPark - Details]

22 SMARTIES

Date 09 12 17

Time 16 00

Street Eben Gevirol st. City Zone 01

No. 18

City Tel - Aviv Jaffa

Size Medium 4-5 cubs

Max Wait Time 1 min.

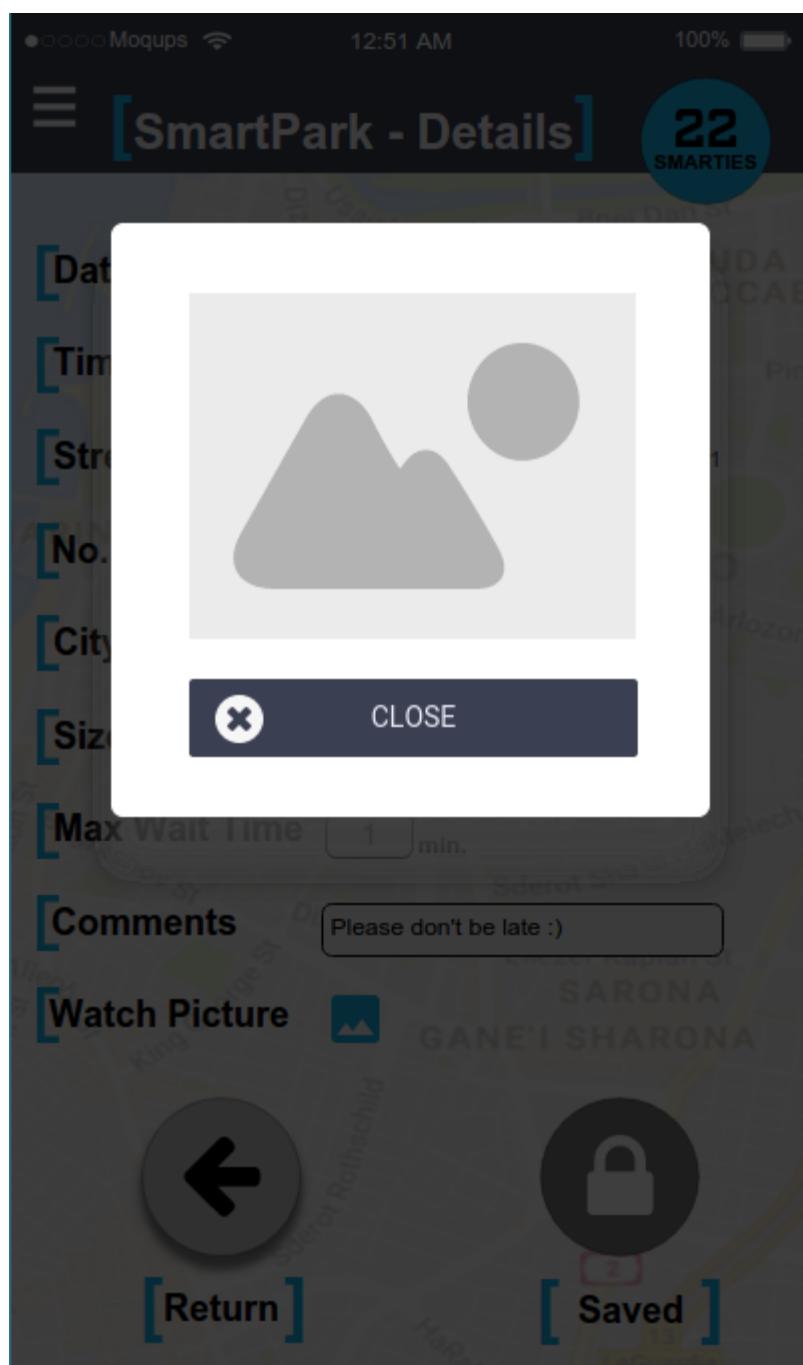
Comments Please don't be late :)

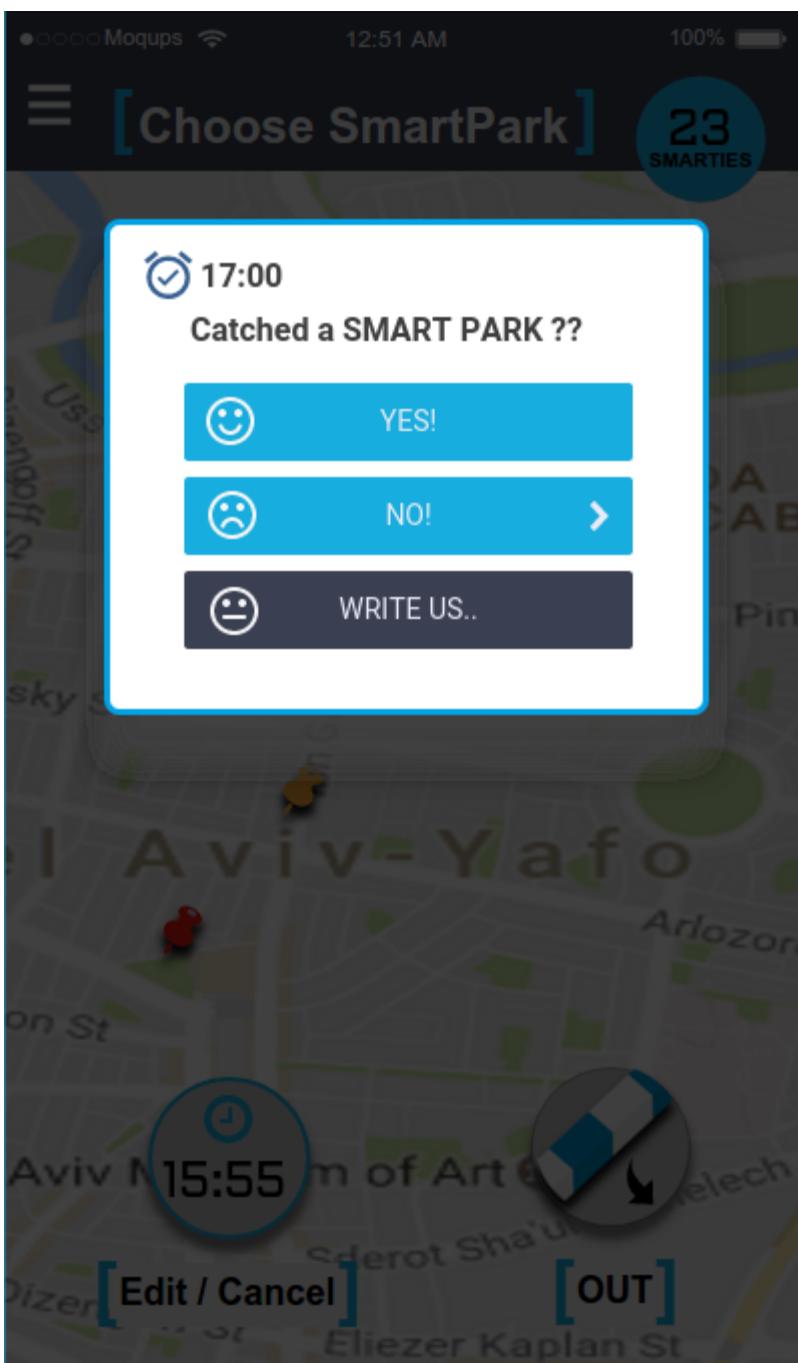
Watch Picture

Return

Saved

A detailed map of Tel Aviv-Yafo is visible in the background, showing various streets and landmarks. The map includes labels for Ha-Yarkon St, Jabotinsky St, Ben Yehuda St, King George St, Sderot Rothschild, and LaGuardia St. The city center is labeled "Tel Aviv-Yafo".







[Smart Park]

