

ספר פרויקט גמר

BELLE



SCHEDULE YOUR BEAUTY

מגיש: איתמר שמחה נחום

הנדסאים תל-אביב

1 תוכן עניינים

4	2	מבוא.....
5	3	הצעת הפרויקט.....
5	3.1	מסמך הצעה חתום.....
7	3.2	רקע.....
7	3.3	סקירת מצב קיים בשוק.....
7	3.4	מה הפרויקט אמור לחדש או לשפר.....
7	3.5	דרישות מערכת ופונקציונליות.....
7	3.5.1	דרישות מערכת (NFR).....
8	3.5.2	דרישות פונקציונליות (FR).....
9	3.6	בעיות צפויות במהלך הפיתוח ופתרונות.....
9	3.7	פתרון טכנולוגי נבחר.....
9	3.7.1	טכנולוגיות בשימוש.....
9	3.7.2	שפות הפיתוח.....
9	3.7.3	תיאור הארכיטקטורה הנבחרת.....
10	3.7.4	חלוקה לתוכניות ומודלים.....
10	3.7.5	סביבת השרת.....
10	3.7.6	ממשק המשתמש / לקוח – GUI.....
11	3.7.7	ממשקים למערכת אחרות / API.....
11	3.7.8	שימוש בחבילות תוכנה.....
11	3.8	שימוש במבנה נתונים.....
11	3.8.1	שיטות האחסון.....
11	3.8.2	מנגנוני התאוששות מנפילה / קריסה.....
12	3.9	תרשימי מערכת מרכזיים.....
12	3.9.1	Use Case.....
13	3.9.2	Sequence diagram.....
17	3.9.3	Dataflow.....
18	3.10	תיאור המרכיב האלגוריתמי – חישובי.....
18	3.10.1	איזה בעיה בא לפתור, איך יפתור?.....
18	3.10.2	איסוף מידע וניתוחים סטטיסטיים.....
18	3.11	אבטחת מידע.....
18	3.12	משאבים הנדרשים לפרויקט.....
18	3.12.1	מספר שעות המוקדש לפרויקט.....
18	3.12.2	ציוד נדרש.....
18	3.12.3	תוכנות נדרשות.....
18	3.12.4	ידע חדש שנדרש ללמוד לצורך הפרויקט.....
19	3.12.5	ספרות ומקורות מידע.....
19	3.13	תוכנית עבודה ושלבים למימוש הפרויקט.....
19	3.14	תכנון הבדיקות שיבוצעו.....
20	3.15	בקרת גרסאות (version control).....
22	4	מסמך SRS.....
22	4.1	רמת הארגון / העסק.....
22	4.1.1	מאפייני הארגון / העסק.....
22	4.1.2	בעלי עניין ואינטרסים תפעוליים.....
23	4.1.3	שירותי הארגון (Business Use Cases).....
23	4.1.4	תרשים לוגיקה עסקית.....
24	4.1.5	תרשים Use Case ברמת הארגון.....

25	רמת המערכת	4.2
25	רשימת תהליכי המערכת (System Use Cases)	4.2.1
26	תרשים תהליכי המערכת (System Use Case Diagram)	4.2.2
27	מפרט תהליכי המערכת (System Use Case Specification)	4.2.3
34	תרשימי SAD	5
34	ארכיטקטורה פיזית	5.1
34	תרשים פריסה (Deployment diagram)	5.1.1
34	פרטי חומרה (Nodes)	5.1.2
34	פרטי תוכנה (Software Artifacts)	5.1.3
35	ממשקים פיזיים	5.1.4
35	ארכיטקטורה לוגית	5.2
35	חלוקה לרכיבי תוכנה (Software Components)	5.2.1
36	תרשים רכיבים (Component Diagram)	5.2.2
37	פירוט רכיבים וממשקים	5.2.3
38	ארכיטקטורת תוכנה מערכתית	5.2.4
38	שגיאה! הסימניה אינה מוגדרת.	
38	תרשימי רצף לתהליכי מערכת (Sequence Diagrams)	5.3
46	מקרי בדיקה / TEST CASES	6
50	פירוט מסכי מערכת	7
50	מסכי כניסה והתחברות	7.1
50	מסך כניסה	7.1.1
52	מסך התחברות	7.1.2
55	מסך הרשמה	7.1.3
58	מסכי משתמש	7.2
58	מסך ראשי	7.2.1
61	דף בית עסק	7.2.2
63	דף קביעת תור	7.2.3
68	דף מפה	7.2.4
71	דף הגדרות	7.2.5
73	דף מעודפים	7.2.6
74	דף תורים של המשתמש	7.2.7
77	מסכי לקוח	7.3
77	יומן פגישות	7.3.1
80	מאגר לקוחות	7.3.2
82	כרטיס מטופל	7.3.3
85	דף הגדרות סוגי טיפול ומחירון	7.3.4
88	הגדרת שעות עבודה	7.3.5
91	דף הגדרת יעדים	7.3.6
94	דף דוחות	7.3.7

2 מבוא

Belle היא אפליקציה לטלפונים חכמים שנועדה להקל על קביעת פגישות בין לקוחות ונותני שירותים בתעשיית היופי. עם שוק הולך וגדל של שירותי יופי, תהליך קביעת התורים נותר לא יעיל ולא נוח. בל שואפת לתת מענה לצורך זה על ידי מתן פלטפורמה ידידותית ונגישה למשתמש ולנותן השירות לקביעת פגישות.

המערכת פותחה באמצעות React Native ושימוש במסד נתונים של Firebase. המערכת פותחה למכשירי איפון ואנדרואיד כאחד, מה שהופך אותה לנגישה למגוון רחב של משתמשים. פיתוח המערכת ב – React Native JavaScript דרש ממני להעמיק את הידע שלי בפיתוח בפלטפורמה זו על מנת ליצור אפליקציה המתאימה לשני מערכות ההפעלה השולטות בשוק באמצעות בסיס קוד אחד, פיתוח המערכת סיפק חווית למידה בעלת ערך רב.

Belle מציעה מגוון תכונות הן ללקוחות והן לספקי שירותים. לקוחות יכולים ליצור משתמש, לחפש ספקי שירות על סמך מיקום או שם העסק, לצפות בזמינות ולהזמין פגישות בזמן אמת. המערכת משתמש גם את ספקי השירות בכך שבאפשרותם ליצור פרופיל עסק, לנהל את לוחות הזמנים של העסק, את הפגישות הקיימות לו, לקבוע ימי ושעות עבודה, להגדיר את סוגי הטיפולים שהוא מציע, הצגת מאגר לקוחות והצגת סטטיסטיקות והכנסות.

הצורך בבל התעורר לאחר שיחות עם אנשים אשר עובדים בתעשייה, שהדגישו את האתגרים העומדים בפניהם בניהול פגישות ומשיכת לקוחות חדשים. עם Belle, ספקי שירות יכולים להרחיב את בסיס הלקוחות שלהם באמצעות החשיפה באפליקציה ולהגדיל את הכנסותיהם, בעוד שהלקוחות יכולים ליהנות מחוויה נוחה ויעילה.

למעשה Belle הוא פרויקט מבטיח שנותן מענה לצורך אמיתי בתעשיית היופי. עם חזון ברור, לבל יש פוטנציאל להשפיע משמעותית על נותני השירות ולהעניק להם פלטפורמה טובה יותר לעבודה.

לקוחות המערכת הינם בתי העסק ומזכרים בספר זה כ – לקוח, נותני שירות, בעל בית עסק, בתי עסק, Care giver.

משתמשי המערכת הינם הלקוחות של בתי העסק ומזכרים בספר זה כ – משתמשים.

3 הצעת הפרויקט

3.1 מסמך הצעה חתום

חוזר מנהל מה"ט 11-4-51 – נספח מס' 1 (הצעה לפרויקט גמר)

תאריך: 1.4.2022
לכבוד יחידת הפרויקטים
מה"ט

הצעה לפרויקט גמר

1. פרטי הסטודנטים

שם הסטודנט	ת.ז. 9 ספרות	כתובת	טלפון נייד	תאריך סיום הלימודים
איתמר נחום שמחה	314773250	אווה בויאר 44/2	0509221149	אוגוסט 2022

שם המכללה: מכללת הנדסאים תל אביב. סמל המכללה: _____

מסלול ההכשרה: הנדסאי תוכנה

מגמת לימוד: תוכנה מקום ביצוע הפרויקט: תל אביב

2. פרטי המנחה האישי

שם המנחה *	כתובת	טלפון נייד	תואר	מקום עבודה/תפקיד
דריו בוג'יו	מושב לבנים	0502258451	מדעי המחשב	ראש מגמת הנדסאים

חתימת הגורם המקצועי מטעם מה"ט

חתימת המנחה האישי

חתימת הסטודנט

[Type here]

דרך מנחם בגין 86 תל אביב ת.ד. 36049 מיקוד 67138
טלפון: 03-7347521 פקס: 03-7347644

חוזר מנהל מה"ט 11-4-51 – נספח מס' 1 (הצעה לפרויקט גמר)

תאריך: 1.4.2022
לכבוד יחידת הפרויקטים
מה"ט

הצעה לפרויקט גמר

1. פרטי הסטודנטים

שם הסטודנט	ת.ז. 9 ספרות	כתובת	טלפון נייד	תאריך סיום הלימודים
איתמר נחום שמחה	314773250	אווה בויאר 44/2	0509221149	אוגוסט 2022

שם המכללה: _____ : מכללת הנדסאים תל אביב. סמל המכללה: _____

מסלול ההכשרה: הנדסאי תוכנה

מגמת לימוד: תוכנה מקום ביצוע הפרויקט: תל אביב

2. פרטי המנחה האישי

שם המנחה *	כתובת	טלפון נייד	תואר	מקום עבודה/תפקיד
ד"ר בוגיו	מושב לבנים	0502258451	מדעי המחשב	ראש מגמת הנדסאים

חתימת הגורם המקצועי מטעם מה"ט

חתימת המנחה האישי

חתימת הסטודנט

[Type here]

דרך מנחם בגין 86 תל אביב ת.ד. 36049 מיקוד 67138
טלפון: 03-7347521 פקס: 03-7347644

3.2 רקע

Belle היא מערכת לטלפונים חכמים שנועדה לעזור ללקוחות ונותני השירות על מנת לקבוע פגישות וטיפולים. למעשה המערכת מחולקת לשני ממשקים, ממשק הלקוח וממשק נותן השירות. ממשק הלקוח: מאפשר ללקוח לחפש מטופל או לראות את המטופלים הנמצאים סביבו ולקבוע פגישה לעסקים השונים. ממשק נותן השירות: מאפשר לבעל העסק לנהל את הפגישות שלו, לאשר פגישות, להגדיר ימי ושעות פעילו, מאגר לקוחות ולצפות בסטטיסטיקות סוגי טיפולים מבוקשים והכנסות.

3.3 סקירת מצב קיים בשוק

נכון להיום, אכן קיימות אפליקציות דומות בשוק. אם זאת לטעמי הן אינן עושות את עבודתן נאמנה, מכיוון שהינן מונגשות כנדרש בהיבט של נוחות המשתמש, והאלמנטים אותם הן מכילות.

3.4 מה הפרויקט אמור לחדש או לשפר

פרויקט זה יפתח בפני בעלי עסק שונים חשיפה נוספת ללקוחות רבים. בעל העסק יוכל לעקוב אחר הפגישות הקודמות של המטופל להציע טיפולים רלוונטיים בהתאם ובכך ליצור תוכנית טיפול טובה יותר. יאפשר ללקוחות לראות את העסקים באזור ולקבוע טיפולים בזמן אמת ללא צורך להתקשר ולבדוק זמינות מול בית העסק.

3.5 דרישות מערכת ופונקציונליות

המערכת תופץ באמצעות פלטפורמת React Native ותתאים למכשירי טלפונים חכמים. מכשירים אלה צריכים להיות מחוברים לרשת האינטרנט בכדי להתממשק עם שרת ה-Firebase לצורך הקמת משתמש והתחברות, כמו כן העלאת והצגת תוכן.

3.5.1 דרישות מערכת (NFR)

- המערכת תרוץ על מכשירי אנדרואיד
- המערכת תרוץ על מכשירי IOS
- המערכת תומכת ברישום דרך Facebook או Gmail.
- המערכת תרוץ על מכשירי ודפדפן.
- המערכת תציג שורת חיפוש בתי עסק.
- המערכת תדרוש בדך בית העסק לעדכן את דירוג חווית הלקוח
- המערכת תציג לבעל העסק דוח הכנסות לפי סוגי טיפולים.

- המערכת תציג במסך הראשי את כפתור נתנתקות וכפתור לגישה ל פגישות קיימות.
- המערכת תציג לבעל העסק דוח לקוחות על פי חתך גילאים
- המערכת תציג לבעל העסק דוח לקוחות על פי סוגי טיפולים.

3.5.2 דרישות פונקציונאליות (FR)

- המערכת תבקש אימייל על מנת להתחבר.
- המערכת תבקש אימייל על מנת להתחבר.
- המערכת תרשום את הלקוח במאגר הנתונים לאחר ההרשמה.
- המערכת תעביר את המשתמש למסך הפתיחה במידה ורשום.
- המערכת תעביר את המשתמש לאחר ההרשמה למסך הפתיחה במידה ורשום
- המערכת תציג ללקוח בתי עסק שונים.
- המערכת תציג שורת חיפוש בתי עסק.
- המערכת תדרוש קלט מהמשתמש על מנת לחפש בעל עסק.
- המערכת תחפש ותציג את בעלי העסק על פי הקלט הנכנס בשורת החיפוש
- המערכת תציג ללקוח את פרטי בית העסק
- המערכת תציג ללקוח את התורים הזמנים של בית העסק
- המערכת תאפשר ללקוח לקבוע תור לבית העסק.
- המערכת תציג התראה ללקוח שהתור נקבע ואושר.
- המערכת תציג ללקוח את התורים העתידיים שקבע לבית העסק השונים.
- המערכת תשלח התראה ללקוח על התור הקרוב.
- המערכת תדרוש בדך בית העסק לעדכן את דירוג חווית הלקוח
- המערכת תדרוש מבעל העסק להגדיר ימי פעילות
- המערכת תדרוש מבעל העסק להגדיר שעות פעילות
- המערכת תדרוש מבעל העסק להגדיר זמן בהתאם לכל טיפול.
- המערכת תבטל תור קיים לבקשת בית העסק
- המערכת תבטל תור קיים לבקשת הלקוח
- המערכת תציג לבעל העסק את דף לקוח.
- המערכת תציג לבעל העסק פגישות קודמות בכרטיס הלקוח.
- בעת הפגישה עם הלקוח המערכת תציג את פרטי הלקוח.
- המערכת תשלח התראה לבעל העסק על פגישה קרובה.
- המערכת תבקש מבעל העסק לתאם ללקוח פגישה נוספת
- המערכת תנתק את המשתמש הקיים מהמערכת לאחר לחיצה על הכפתור ההתנתקות
- המערכת תשתמש בשרת חיצוני לאיחסון.
- בסיום הפגישה המערכת תדרוש מבעל העסק להזין סיכום פגישה.
- בעת הפגישה עם הלקוח המערכת תציג מידע אודות הטיפול אשר הלקוח הזמין

3.6 בעיות צפויות במהלך הפיתוח ופתרונות

- **בהיבט של הגנת הלקוח:** על מנת ליצור דף של בעל עסק במערכת והתחברות לשירות על בעל העסק לקבוע פגישת ייעוץ ואישור על ידי צוות Belle, רק מטפלים מורשים עם רישיון עיסוק והסמכה יוכלו להשתמש באפליקציה ובכך להגן על המשתמשים.
- **בהיבט של למידת שפה חדשה:** בנוסף ללימודים המרובים שלנו, עלי ללמוד שפת פיתוח חדשה, פלטפורמה חדשה וטכנולוגית שרת חדשה, נדאג לכך על ידי למידת קורסים מורחבים בנושא והרבה מוטיבציה.
- **בהיבט של אבטחת פרטיות המשתמשים:** מענה לנושא זה יינתן באמצעות טכנולוגית השרת Firebase שהיא מכילה ממגנונים לשמירה על התוכן הרגיש.

3.7 פתרון טכנולוגי נבחר

3.7.1 טכנולוגיות בשימוש

הטכנולוגיות בהן אני משתמשים הן JavaScript – React Native בסביבת Visual Studio Code ושרת אחסון מטעם גוגל Firebase.

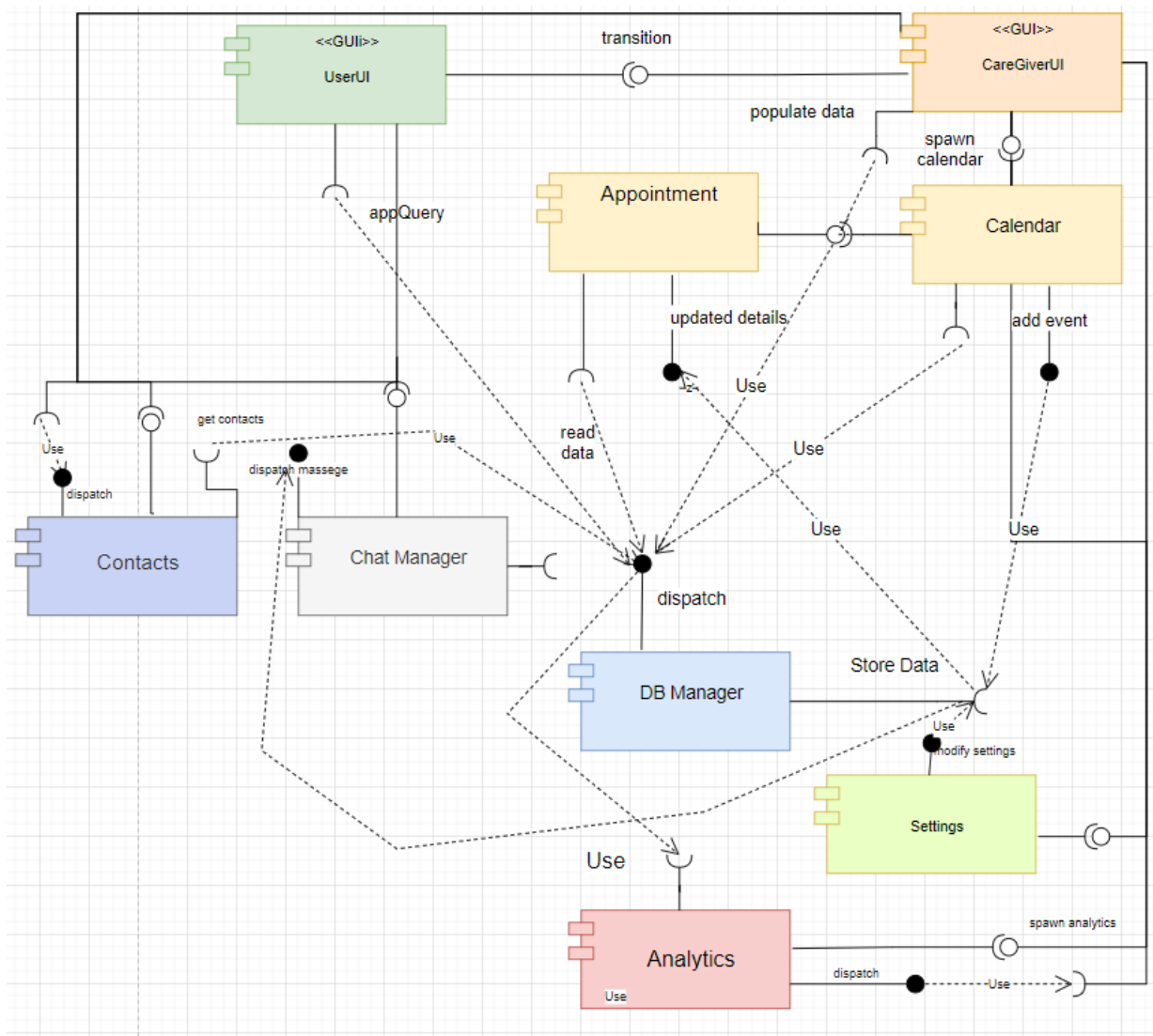
3.7.2 שפות הפיתוח

שפת הפיתוח שאותה בחרתי הינה JavaScript – React Native, בחרתי זאת מכיוון שרציתי לאתגר את עצמי בלימוד של שפה חדשה ובנוסף לאפשר תמיכה בשני מערכות ההפעלה השולטות בשוק (IOS, Android).

3.7.3 תיאור הארכיטקטורה הנבחרת

- המערכת מבוססת על מודל שלוש השכבות.
- בשכבת הלקוח יושב ה-GUI (ממשק המשתמש).
- המערכת מתקשרת באמצעות פרוטוקול מאובטח עם FIREBASE
- השרת מתקשר עם האימות של גוגל באמצעות פרוטוקול תקשורת מאובטח, בנוסף השרת מתקשר באמצעות פרוטוקול מאובטח עם מחשב המנהל.

3.7.4 חלוקה לתוכניות ומודלים



3.7.5 סביבת השרת

סביבת שרת תהיה בסביבת ענן בפלטפורמת Firebase.

3.7.6 ממשק המשתמש / לקוח – GUI

- הממשק יהיה זמין לכל משתמש שבבעלותו מכשיר חכם המחובר לרשת.
- הממשק הגרפי יהיה מועדכן באופן שוטף על פי עדכוני המערכת.

[קישור לאב טיפוס של המערכת](#)

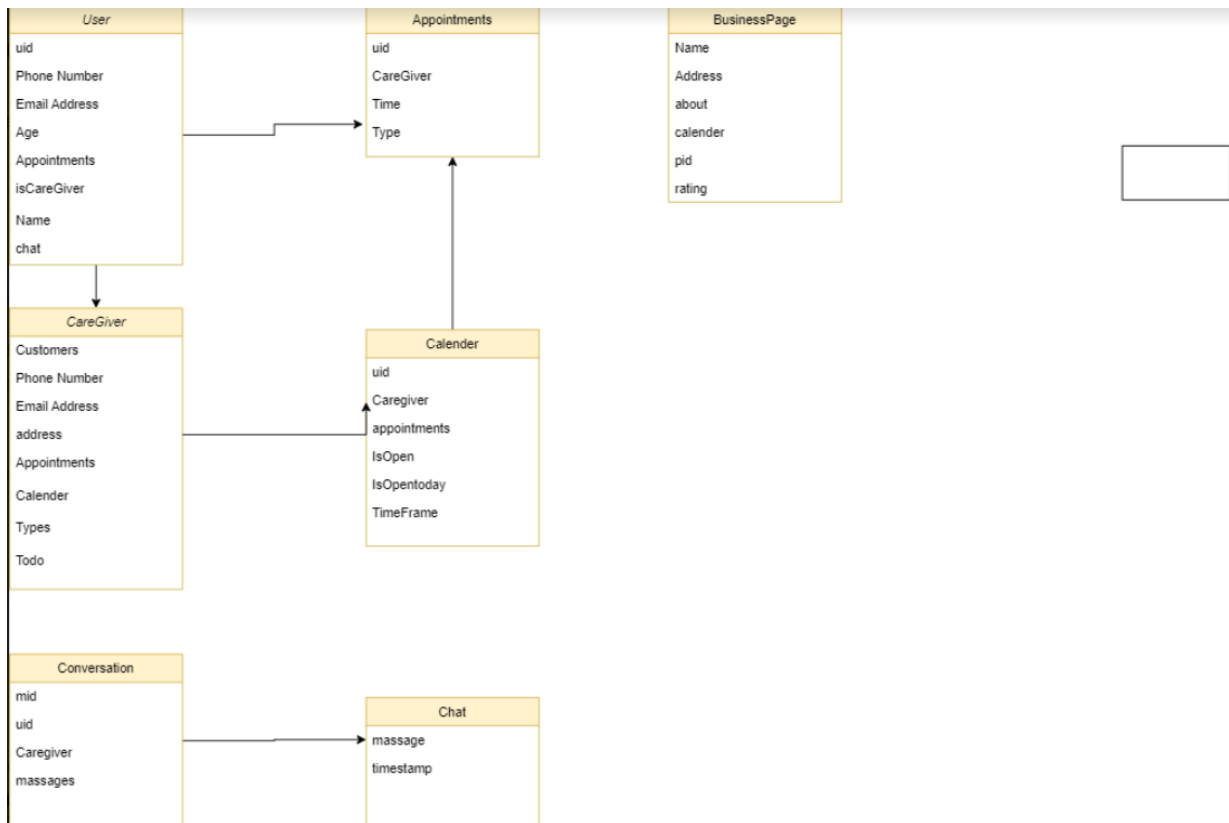
3.7.7 ממשקים למערכת אחרות / API

המערכת תתממשק למערכות API של גוגל לצורך זיהוי ואימות מטפלים.

3.7.8 שימוש בחבילות תוכנה

React Native

3.8 שימוש במבנה נתונים



3.8.1 שיטות האחסון

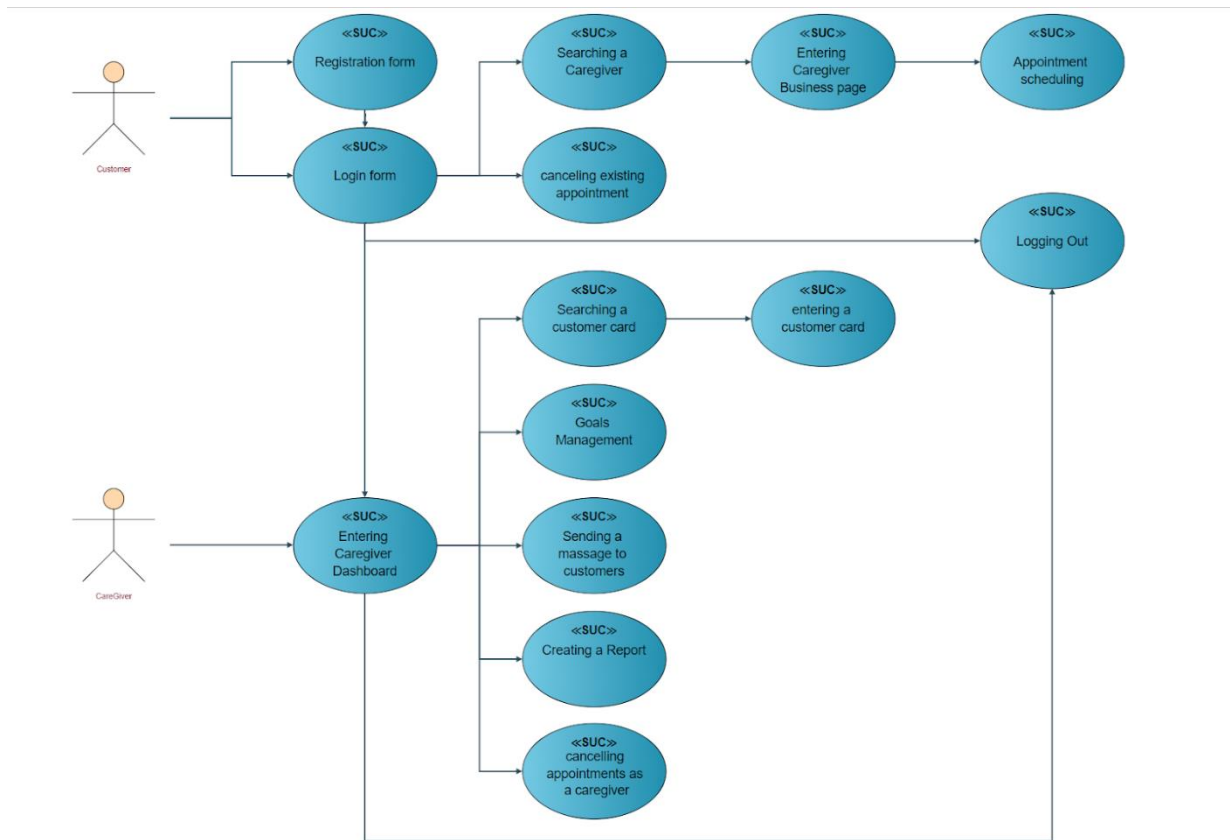
בסיס נתונים לא רציונלי Firebase no-SQL.

3.8.2 מנגנוני התאוששות מנפילה / קריסה

התאוששות מנפילות: על מנת להימנע מנפילות \ מחיקת מידע, השרת מבצע גיבוי אוטומטי על בסיס יומי לתוך הענן, ושומר את המידע בפורמט JSON

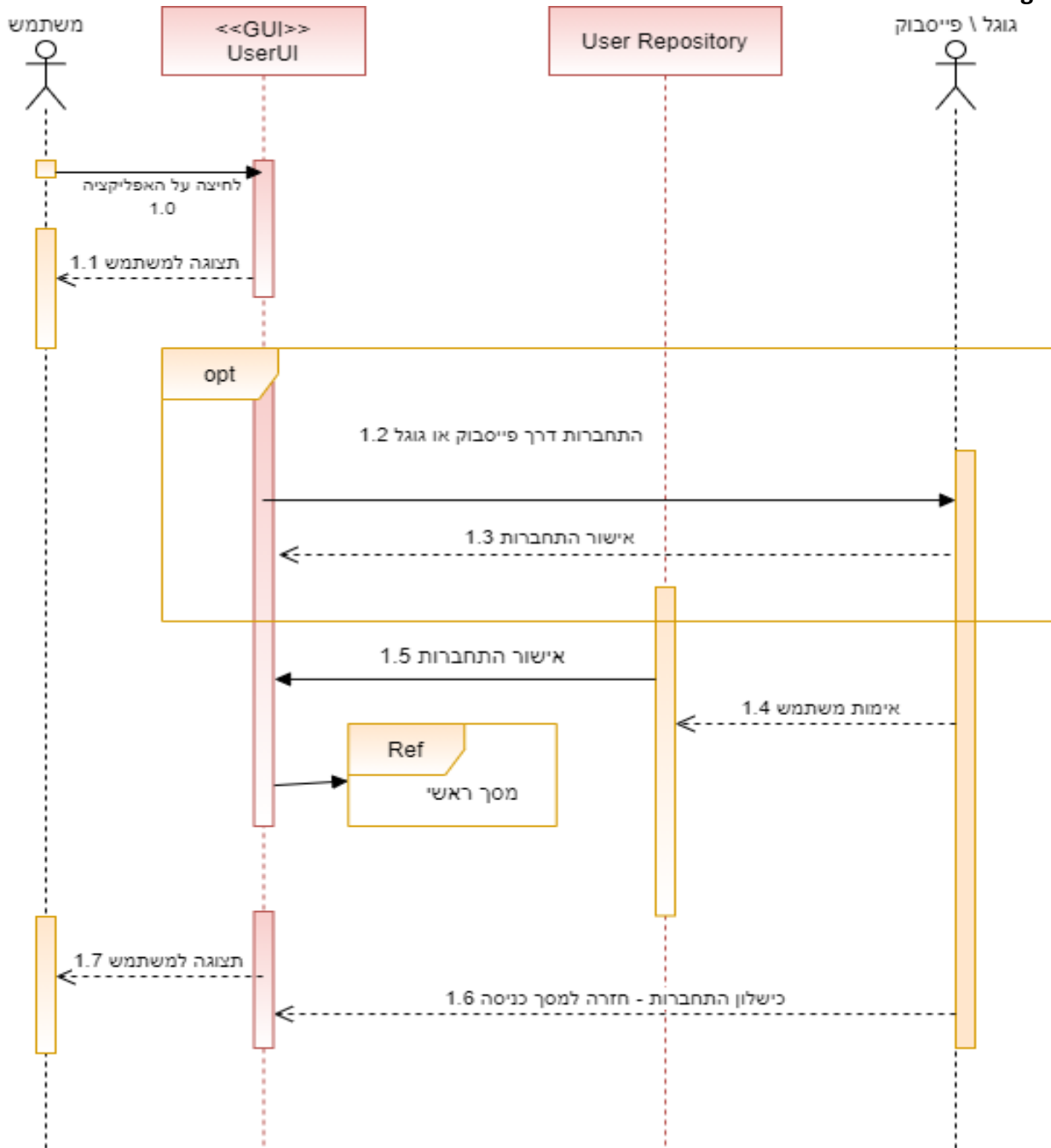
3.9 תרשימי מערכת מרכזיים

Use Case 3.9.1

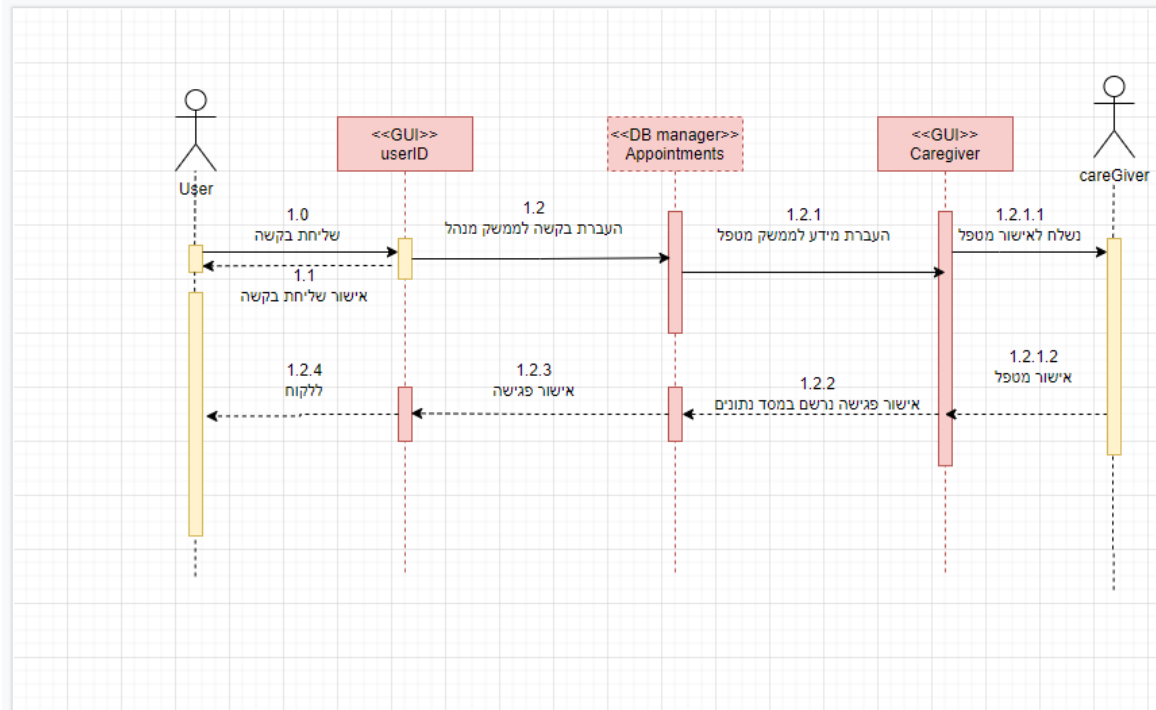


Sequence diagram 3.9.2

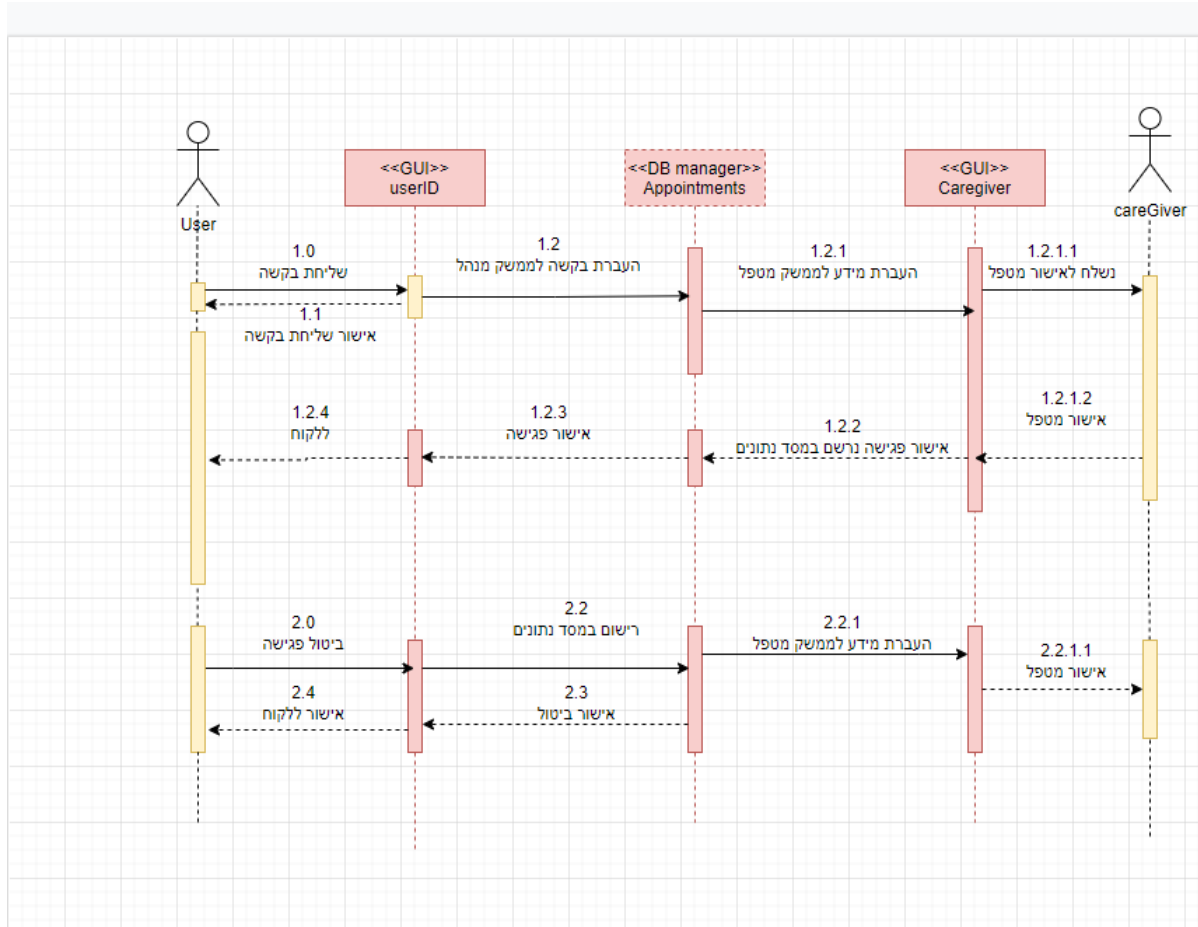
Login



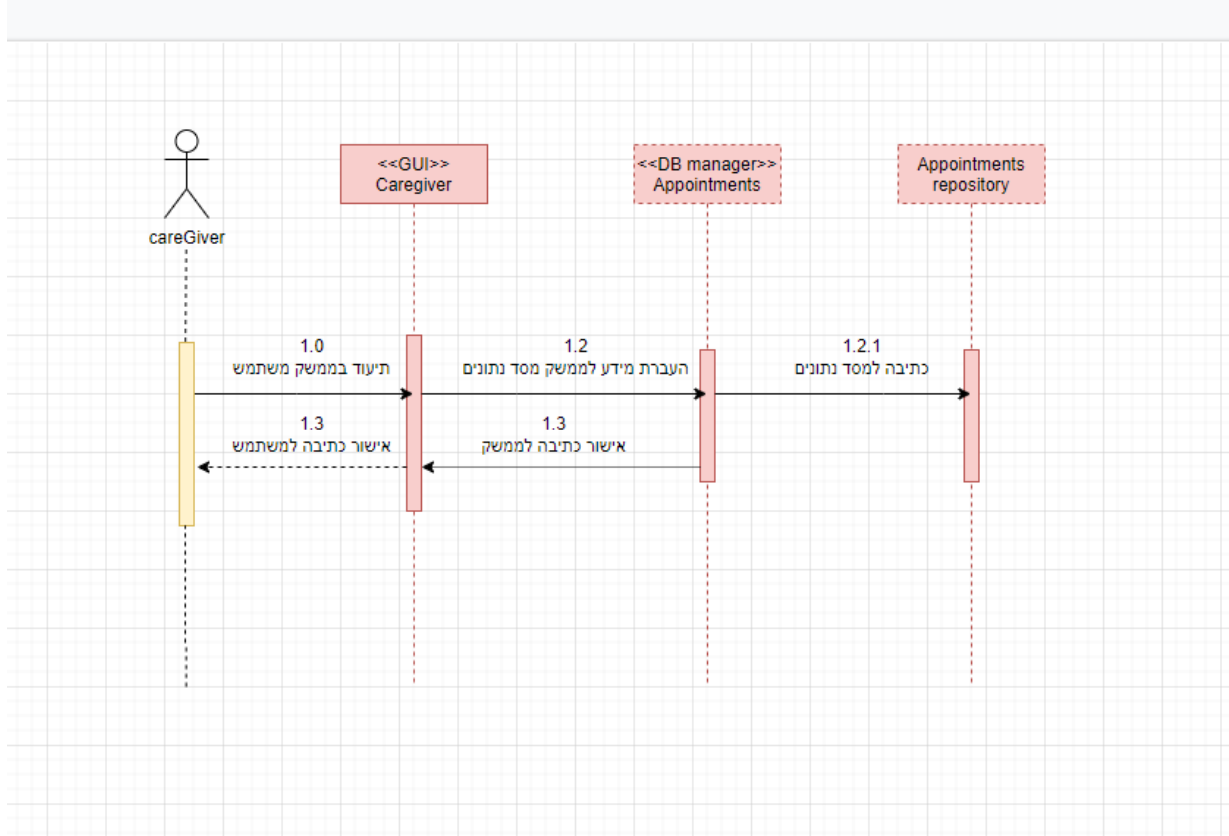
Schedule appointment

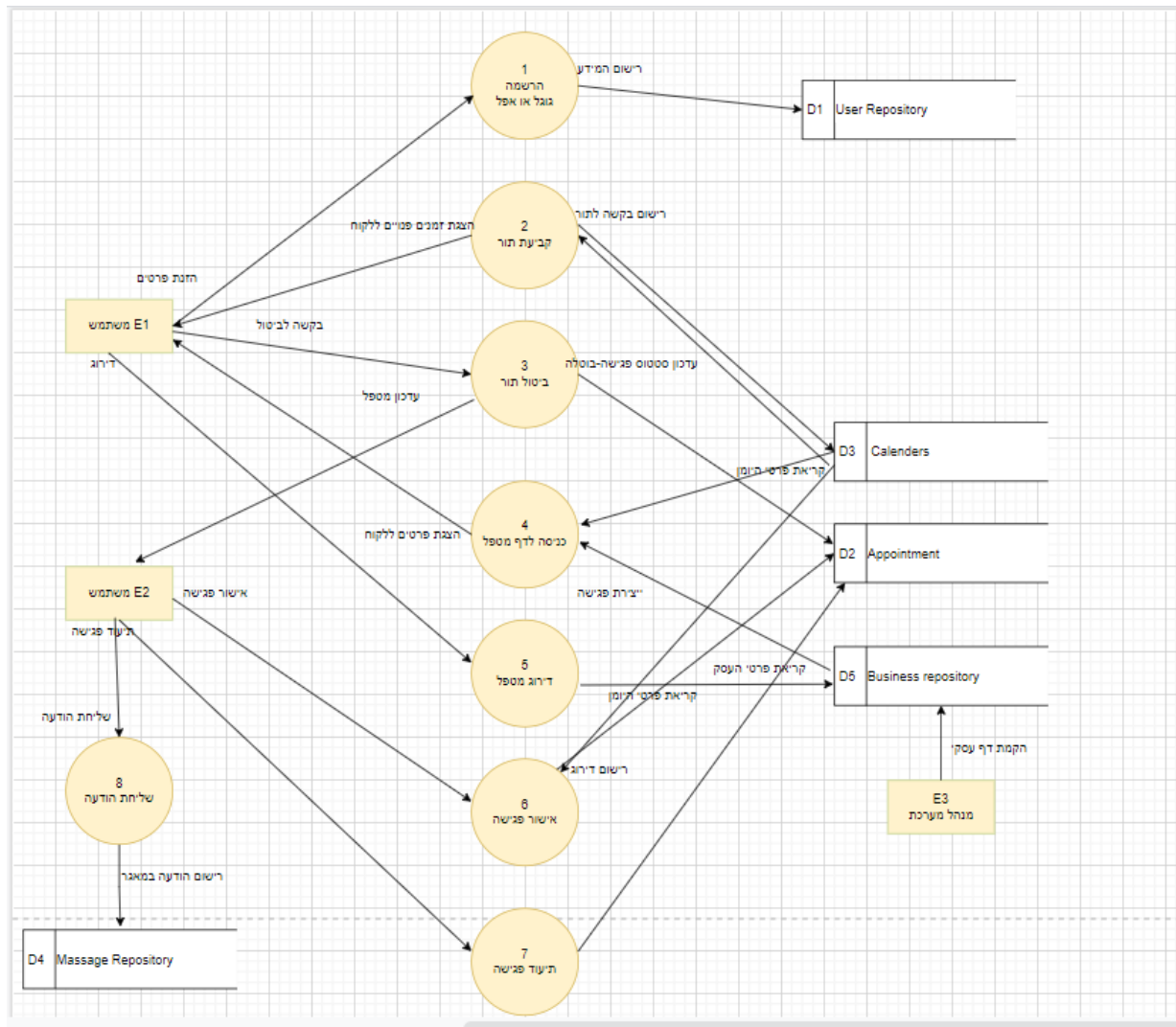


Schedule + cancel appointments



Documenting an appointment





3.10 תיאור המרכיב האלגוריתמי – חישובי

3.10.1 איזה בעיה בא לפתור, איך יפתור?

- האפליקציה נועדה לעזור למטפלים לנהל ולתעד פגישות בצורה יותר יעילה וללקוחות לקבוע פגישות.

3.10.2 איסוף מידע וניתוחים סטטיסטיים

- המערכת תאגור את המידע שהמשתמשים הזינו באפליקציה על גבי שרת ה-Firebase
- האפליקציה תעשה ניתוחים סטטיסטיים ותאגור מידע אודות המשתמשים.

3.11 אבטחת מידע

- המידע של האפליקציה יישב בשרת ה-FIREBASE.
- אבטחת השרת יכולה להיכתב על ידי מנהלי המערכת על ידי יצירת חוקים.
- המשתמש מתחבר עם פרטי משתמש וסיסמה, שבעזרתם הוא ניגש לאפליקציה.

3.12 משאבים הנדרשים לפרויקט

3.12.1 מספר שעות המוקדש לפרויקט

- +600 שעות עבודה על הפרויקט

3.12.2 ציוד נדרש

- מחשב עם מערכת הפעלה של Windows
- חיבור אינטרנט פעיל
- מכשיר סלולרי חכם

3.12.3 תוכנות נדרשות

- React-Native
- Firebase
- Visual Studio Code
- Github

3.12.4 ידע חדש שנדרש ללמוד לצורך הפרויקט

למדתי קורסים ב Udemy בהיקף של 29 שעות לימוד בשפת REACT NATIVE לצורך פיתוח בסביבת WEB. קורסים נלווים:

- <https://www.udemy.com/course/complete-react-native-mobile-development-zero-to-mastery-with-hooks>

3.12.5 ספרות ומקורות מידע

- <https://stackoverflow.com>
- [/https://firebase.react.dev](https://firebase.react.dev)
- <https://firebase.google.com/docs/web/setup>
- [/https://github.com](https://github.com)

3.13 תוכנית עבודה ושליבים למימוש הפרויקט

תאריך	שלב בפרויקט
1.1.2022	הצעת פרויקט וסיפור לקוח
1.2.2022	אב טיפוס
1.3.2022	הגדרת דרישות מערכת
1.3.2022	ניתוח דרישות מערכת
14.3.2022	ארכיטקטורת מערכת
15.4.2022	עיצוב ותוכן המערכת
1.5.2022	קידוד
23.02.2023	בדיקות יחידה
23.02.2023	בדיקות מערכת
16.03.2023	הגשת ספר פרויקט
30.03.2023	הגנה על הפרויקט

3.14 תכנון הבדיקות שיבוצעו

בדיקה	בוצע בתאריך	תוצאות
הפעלת אפליקציה	23.02.2023	תקין
הרשמת משתמש חדש	23.02.2023	תקין
התחברות משתמש רשום	23.02.2023	תקין
קביעת תור	23.02.2023	תקין
ביטול תור	23.02.2023	תקין
תיעוד פגישה	23.02.2023	תקין
יצירת דוח	23.02.2023	תקין
דירוג מטפל	23.02.2023	תקין
צפייה ברשימת לקוחות	23.02.2023	תקין
חיפוש מטפל	23.02.2023	תקין
כניסה לדף מטפל	23.02.2023	תקין
התנתקות	23.02.2023	תקין

3.15 בקרת גרסאות (version control)

בקרת גרסאות תתבצע דרך חשבון GitHub מיוחד שהוקם לצורך כך -
<https://github.com/LolipopEater/Belle>



חתימת המנחה



חתימת הסטודנט

3. הערות ראש המגמה במכללה

4. אישור ראש המגמה

תאריך: 21/04/2021



שם: דריי בוג'יו חתימה:

5. הערות הגורם המקצועי מטעם מה"ט

6. אישור הגורם המקצועי מטעם מה"ט

שם: _____ חתימה: _____ תאריך: _____

4 מסמך SRS

4.1 רמת הארגון / העסק

4.1.1 מאפייני הארגון / העסק

שם	Belle
מטרה	יצירת סביבה נוחה לניהול לקוחות ועסקים פרטיים
מרכיבים	פורטל משתמש להזמנת מקום ללקוחות ופורטל נפרד לבעלי עסק לניהול פגישות
סביבה	לקוחות ומטופלים, בעלי עסקים ונותני שירות
סדר וארגון	מאגר הלקוחות של BELLE חולק את אנשי הקשר שלו עם בעל העסק היכול לרשום ולעקוב אחרי פגישות קיימות ופגישות קיימות
אינטראקציה	המערכת הממוחשבת מאפשרת ללקוחות לקבוע תור דרך פורטל הלקוחות על ידי חיפוש המטפל ברגע קביעת התור בעל העסק מקבל אישור לקביעת תור הוא יכול לאשר או לבטל את התור בהתאם ללוח שלו. בעל העסק עוקב אחרי לקוחות ותורים קיימים.

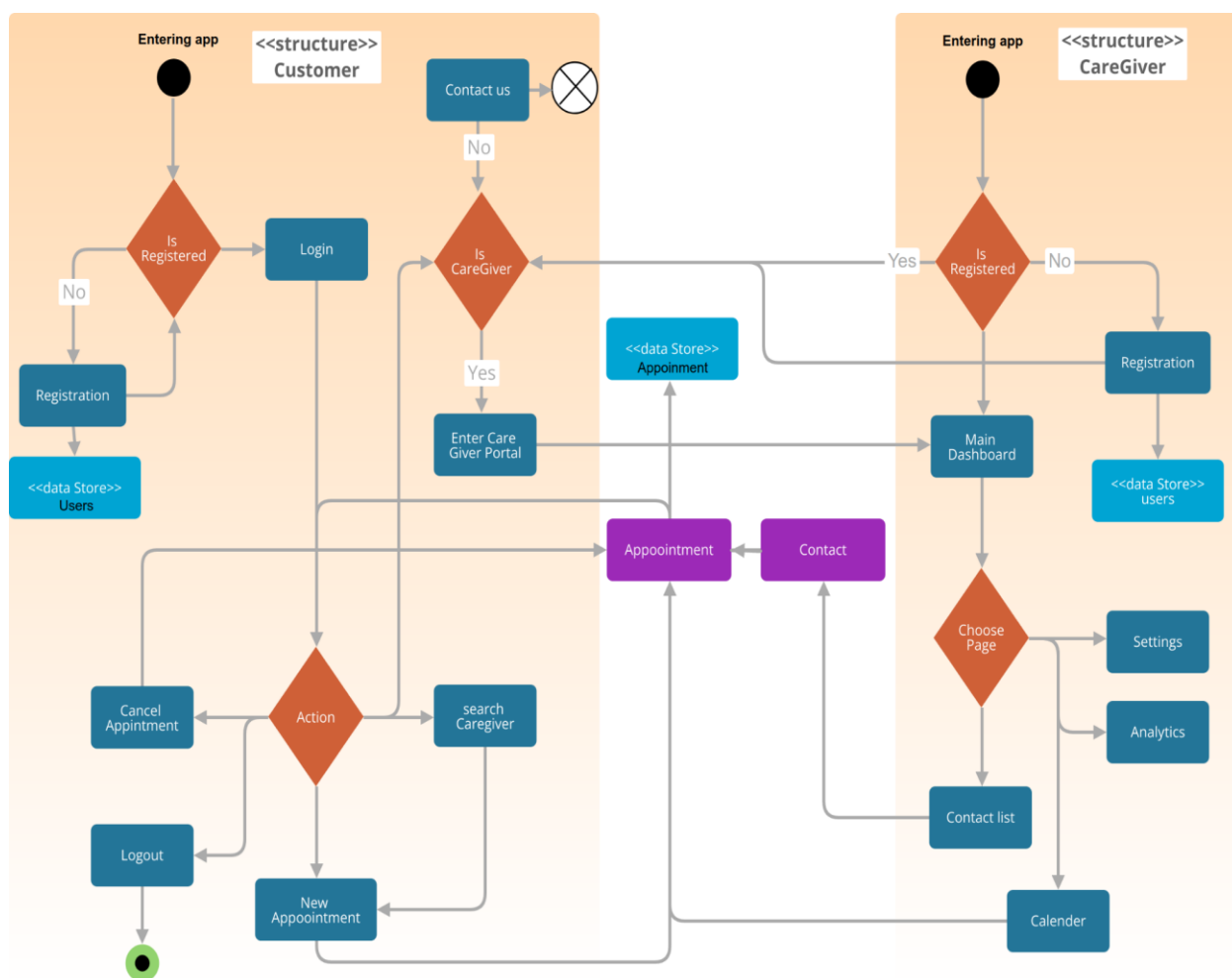
4.1.2 בעלי עניין ואינטרסים תפעוליים

בעל עניין	אינטרסים תפעוליים [מאפייני איכות]	תפקיד
מטפל(בעל עסק)	מעקב אחר תורים ותיעוד תורים חדשים על מנת לשמור על בטחון הלקוח קיימים וקבלת תורים חדשים [שימושיות, זמינות, אמינות, בטיחות]	משתמש
לקוח	יכולת לנווט בפורטל המטפלים ולחפש את בעל העסק הספציפי ולקבוע תור\לבטל תור קיים- לראות מידע על המטפל [שימושיות, זמינות]	משתמש
בונה דף העסק	בניית הדף העסקי לבעל העסק המכיל מידע\מידע\סרטונים בפגישה בתשלום [שימושיות, אמינות]	מפעיל
בעלי האפליקציה	גביית תשלום על כל שימוש [שלמות המידע – Integrity]	-
תקן בטיחות	שמירת המידע של הלקוחות הרשומים במאגר הלקוחות של BELLE [בטיחות]	-

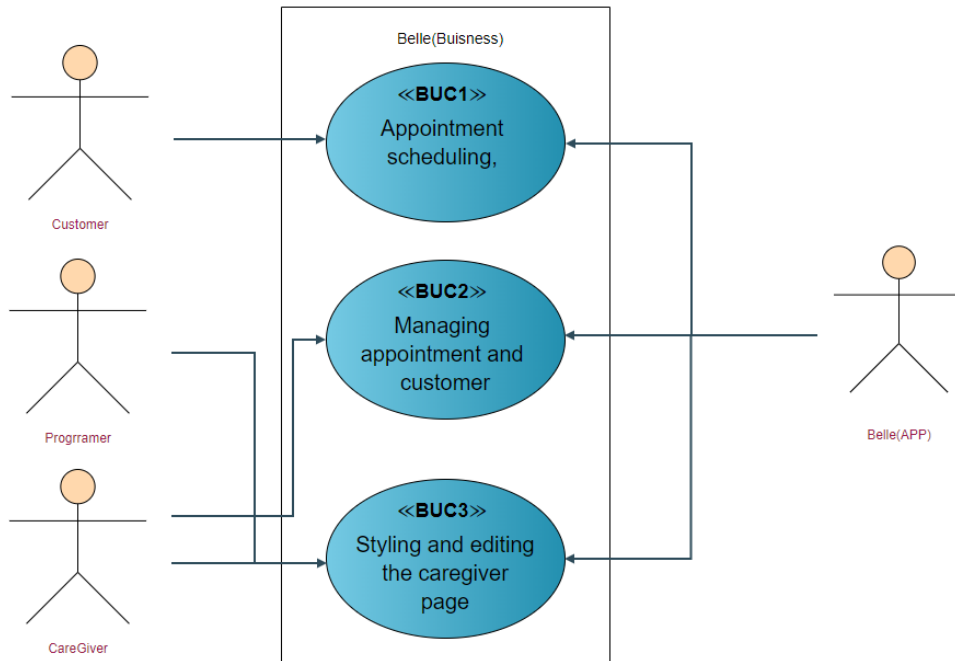
4.1.3 שירותי הארגון (Business Use Cases)

זיהוי	שם ה-UC	שחקנים/ראשי/ים	תיאור קצר
BUC-1	קביעת תורים וקבלת מידע	לקוח	חיפוש בעל עסק מטפל קיים וקביעת תור בשעה הרצויה
BUC-2	ניהול תורים קיימים וברטיסי לקוח	מטפל (בעל עסק)	ניהול ברטיסי לקוח הכוללים פגישות קודמות, מידע אודות הלקוח, רגישויות והמשך טיפול, הוצאת דוחות.
BUC-3	ניהול דפי לקוח	מפעיל	עיצוב ברטיס הלקוח על ידי דרישות המטפל

4.1.4 תרשים לוגיקה עסקית



4.1.5 תרשימים Use Case ברמת הארגון

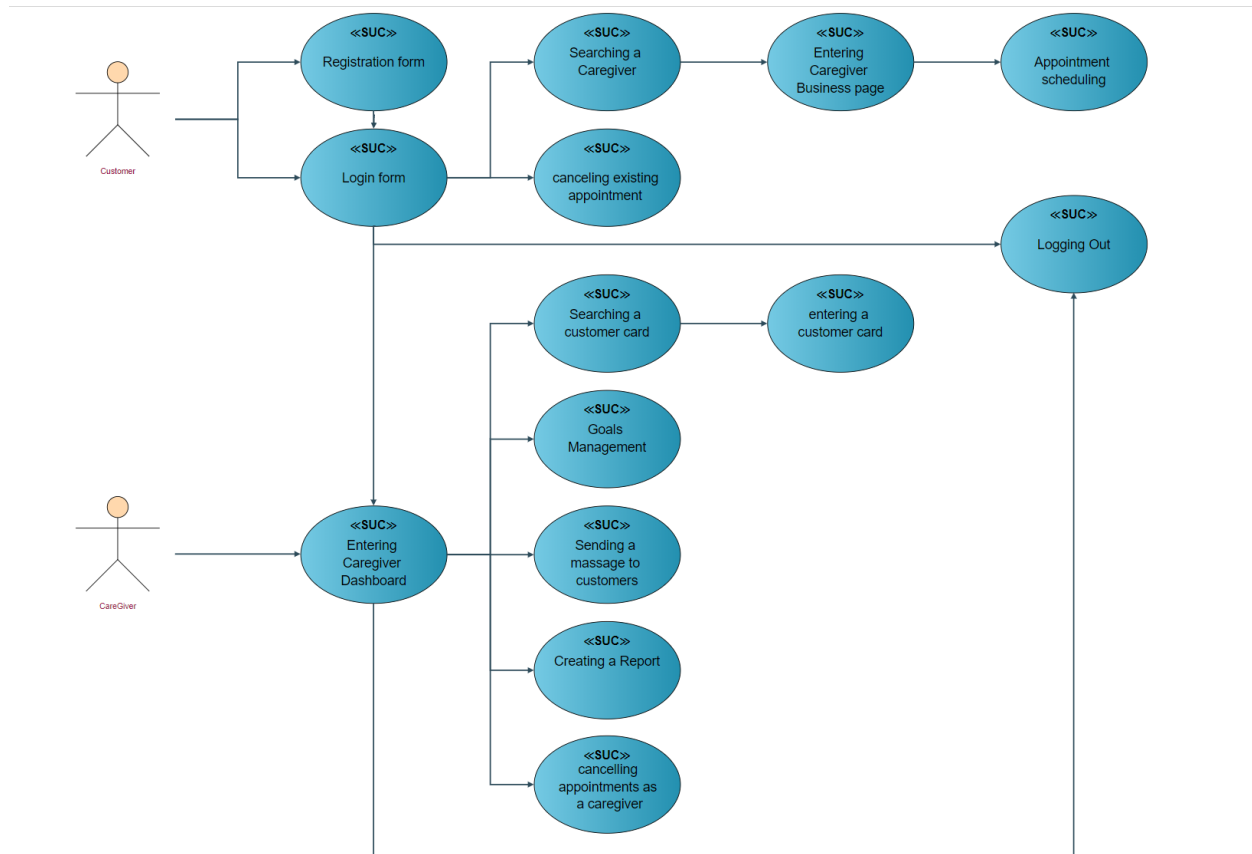


4.2 רמת המערכת

4.2.1 רשימת תהליכי המערכת (System Use Cases)

זיהוי	שם ה-UC	שחקנים ראשיים	תיאור קצר
SUC-1	כניסה לאפליקציה	לקוח, מטפל	כניסה לאפליקציה באמצעות פרטים מזהים- מייל + סיסמה
SUC-2	הרשמה לאפליקציה	לקוח, מטפל	הרשמה לאפליקציה, באמצעות פרטים מזהים: דואר אלקטרוני, מספר טלפון, גיל, ומין
SUC-3	קביעת תור	לקוח	קביעת תור לפגישה עם בעל העסק הרצוי
SUC-4	התנתקות מהאפליקציה	לקוח, מטפל	לחיצת על כפתור ההתנתקות ויציאה מהאפליקציה
SUC-5	כניסה לדף עסק	לקוח	כניסה לדף המראה את פרטי בית העסק, מיקום, שעות פעילות, תורים פנויים
SUC-6	חיפוש בית עסק	לקוח	חיפוש בית עסק על פי שם באמצעות שורת החיפוש
SUC-7	ביטול תור קיים	לקוח	בפורטל המשמתמש אפשרות לבטל תורים קיימים
SUC-8	כניסה לפורטל ניהול בית העסק	מטפל	מעבר מפורטל הלקוח אל פורטל ניהול בית העסק.
SUC-9	חיפוש כרטיס לקוח	מטפל	חיפוש הלקוח על פי פרמטרים מזהים : שם, מספר טלפון, דואר אלקטרוני
SUC-10	כניסה לכרטיס לקוח	מטפל	כניסה לכרטיס לקוח ספציפי ועריכתו על פי תאריך הפגישה
SUC-11	הנפקת דוחות	מטפל	הנפקת דוחות שונים על פי סוגי טיפול ופרקי זמן
SUC-12	הגדרת יעדים	מטפל	הגדרת יעדים ושעות עבודה
SUC-13	ביטול תורים נכנסים	מטפל	ביטול כניסת תורים חדשים מאפליקציית BELLE לאלתר

4.2.2 תרשים תהליכי המערכת (System Use Case Diagram)



4.2.3 מפרט תהליכי המערכת (System Use Case Specification)

כניסה לאפליקציה	SUC-1
<ul style="list-style-type: none"> ● לקוח: להתחבר למשתמש ולקבוע תור ● מטפל: להתחבר לפורטל הניהול 	שחקנים ויעדים
<ul style="list-style-type: none"> ● מטפל: ניהול לקוחות ● לקוח: קביעת תור מהיר 	ב"ע ואינטרסים
<ul style="list-style-type: none"> ● הלקוח או המטפל רשומים לאפליקציה ● הלקוח או המטפל מחוברים לאפליקציה 	pre-conditions
<ul style="list-style-type: none"> ● הלקוח מחובר לאפליקציה ● המטפל מחובר לאפליקציה ● הלקוח רואה או המטפל רואים מטפלים ● ניתן לחפש מטפל 	post-conditions
המשתמש מכניס את פרטי הזיהוי משתמש וסיסמה	trigger
<ol style="list-style-type: none"> 1. המערכת מצגיה למשתמש טופס כניסה 2. המשתמש מזין את הסיסמה שלו ואת האימייל שלו 3. המערכת מוודאת שהפרטים תקינים ומלאים 4. המערכת מאמתת פרטים עם מסד הנתונים לבדוק שהפרטים נכונים 5. המערכת מציגה ללקוח את פורטל המשתמש עם הפרטים האישיים שלו 	MSS
<p><u>חלופה מצעד 3 של MSS: פרטי המשתמש אינם תקינים / מלאים</u></p> <p>1א3. המערכת מציגה את הטופס בציון הפרטים שאינם תקינים</p> <p>2א3. חזרה לצעד 2</p>	הסתעפות א'
<p><u>חלופה מצעד 4 של MSS: אחד הפרטים אינם תואמים למסד הנתונים</u></p> <p>1א4. המערכת מתריעה שאחד הפרטים אינם נכונים</p>	הסתעפות ב'
תפעוליות:	עקיבות
אחרות: 1,2	לדרישות

הרשמה לאפליקציה	SUC-2
● <u>לקוח</u> : לקבוע תור או להפוך להיות מטפל ולהשתמש באפליקציית ניהול	שחקנים ויעדים
● <u>מטפלים</u> : לקוחות קובעים פגישות על ידי פורטל הלקוח	ב"ע ואינטרסים
● הלקוח לא רשום במאגרי החברה	pre-conditions
● פרטי הלקוח נרשמים במאגר ● חשבון לקוח נוצר ● הלקוח יכול לחפש את המטפל שלו ולקבוע תור	post-conditions
הלקוח נרשם על ידי מילוי פרטיו האישיים שם, מייל, תאריך לידה, מין >פלאפון.	trigger
1. המערכת מבקשת מהלקוח פרטי הרשמה 2. הלקוח מכניס את הפרטים האישיים שלו שם, מייל, גיל, מין. 3. המערכת בודקת האם הפרטים תקינים 4. המערכת בודקת האם המייל והפלאפון הנייד בשימוש על ידי בדיקת מסד הנתונים 5. המערכת רושמת את הלקוח במסד הנתונים ומחברת אותו לאפליקציה	MSS
<u>חלופה מצעד 3 של MSS</u> : אחד הפרטים שהכנסו לא תקינים 1א3. המערכת מתריעה איזה ערך מצריך תיקון 2א3 חזרה למצעד 2 של MSS	הסתעפות א'
<u>חלופה: מצעד 4 של MSS</u> : אחד הפרטים שהוכנסו שייכים ללקוח רשום 1א4 המערכת תתריעה ללקוח שאחד הפרטים שהשתמש בהן תפוס	הסתעפות ב'
תפעוליות: 11 אחרות: 6,7,8,9,10	עקיבות לדרישות

SUC-3	קביעת תור
שחקנים ויעדים	<ul style="list-style-type: none"> ● לקוח: קביעת תור בזמן הרצוי ● מטפל: קביעת תור ללקוח
ב"ע ואינטרסים	<ul style="list-style-type: none"> ● מטפל: המשך טיפול וקביעת תור נוסף ● לקוח: קביעת תור לטיפול
pre-conditions	<ul style="list-style-type: none"> ● הזמן הנבחר אינו תפוס על ידי לקוח אחר ● הזמן הנבחר אינו חסום על ידי המטפל
post-conditions	<ul style="list-style-type: none"> ● הלקוח שלח למטפל בקשה לאישור התתור ● לאחר אישור התור התור נרשם אצל הלקוח ● לאחר אישור התור התור נמצא ביומן המטפל ● לאחר אישור התור נחסם ביומן
trigger	הלקוח בוחר במסגרת הזמן המתאימה לו ושולח את הבקשה
MSS	<ol style="list-style-type: none"> 1. המערכת מציגה ללקוח את הזמנים האפשריים בהן יוכל לקבוע תור 2. הלקוח בוחר מסגרת זמן המתאימה לו 3. הלקוח שולח למטפל את הבקשה לתור 4. המערכת שולחת למטפל בקשה לאישור התור 5. המטפל מאשר את התור 6. המערכת מוסיפה את התור ללוח השעות של המטפל 7. המערכת מוסיפה את התור לרשימת התורים בפורטל הלקוח 8. המערכת חוסמת את הזמן מאושר ביומן של המטפל
עקיבות לדרישות	תפעוליות: 40,20,21,19,18 אחרות:

SUC-4	התנתקות מהאפליקציה
שחקנים ויעדים	<ul style="list-style-type: none"> ● לקוח: יציאה מהאפליקציה ● בעל עסק: יציאה מהאפליקציה
ב"ע ואינטרסים	<ul style="list-style-type: none"> ● לקוח ● בעל עסק
pre-conditions	<ul style="list-style-type: none"> ● להיות מחובר למערכת
post-conditions	<ul style="list-style-type: none"> ● יציאה מהמערכת
trigger	המשתמש לוחץ על כפתור ההתנתקות בפורטל הלקוח
MSS	<ol style="list-style-type: none"> 1. המערכת מציגה ללקוח את כפתור היציאה 2. המשתמש לוחץ על כפתור היציאה 3. המערכת מנתקת את הלקוח וסוגרת את הסשן הנוכחי
עקיבות לדרישות	תפעוליות: 58,59 אחרות:

SUC-5	כניסה לדף עסק
שחקנים ויעדים	● לקוח
ב"ע ואינטרסים	● מטפל: פרסום ● לקוח: קבלת מידע על המטפל
pre-conditions	● לקוח רשום ● לקוח מחובר למערכת
post-conditions	● כניסה לדף המטפל ● פרטי המטפל מוצגים ● אפשרות לקבוע תור מוצגת למשתמש
trigger	משתמש לוחץ על שם המטפל
MSS	1. המערכת המערכת טוענת את דף בית העסק
הסתעפות א'	חלופה בצעד 1 של MSS: המערכת אינה מוצאת את הנתונים של בית העסק 1א1. המערכת מתריאה על שגיאת נתונים
עקיבות לדרישות	תפעוליות: 16,17,18,19 אחרות:

SUC-6	ביטול תור
שחקנים ויעדים	● לקוח: שחקן משני ● מטפל: שחקן
ב"ע ואינטרסים	● מטפל: ביטול תור מסיבה אישית\ עיסקית ● לקוח: ביטול תור מסיבות אישיות
pre-conditions	● משתמש מחובר לאפליקציה ● תור קיים
post-conditions	● התור מתבטל
trigger	● הלקוח לוחץ על כפתור הביטול
MSS	1. המשתמש פותח את רשימת התורים שלו 2. המשתמש לוחץ על כפתור הביטול ומאשר 3. המערכת נעדכנת את בעל העסק שהתור בוטל 4. המערכת מוחזרת את התור מרשימת התורי העתידיים של המשתמש
עקיבות לדרישות	תפעוליות: 42,41 אחרות:

SUC-7	התחברות לפורטל ניהול
שחקנים ויעדים	● מטפל: כניסה לפורטל ניהול
ב"ע ואינטרסים	● מטפל: ניסה לפורטל ניהול
pre-conditions	● משתמש רשום ● משתמש מחובר ● משתמש רשום במטפל
post-conditions	● כניסה לפורטל הניהול
trigger	הלקוח לוחץ על הכפתור למעבר לפורטל הניהול
MSS	1. המערכת מציגה למשתמש כפתור 2. המשתמש לוחץ על כפתור המעבר לפורטל הניהול 3. המערכת מוודאת שהמשתמש המחובר אכן אם גישה לפורטל ניהול 4. המערכת מעבירה את הלקוח לדף הניהול
הסתעפות א'	חלופה בצעד 3 של MSS: לקוח ללא גישה. 3א1. המערכת מציגה התרעה מתאימה ללקוח 3א2. חזרה לצעד 1
עקיבות לדרישות	תפעוליות: 6 אחרות:

SUC-8	חיפוש כרטיס לקוח
שחקנים ויעדים	● בעל עסק: חיפוש כרטיס לקוח ותיעוד פגישה
ב"ע ואינטרסים	● בעל עסק: תיעוד פגישות מילוי פרטי לקוח ושמירה במאגר לקוחות
pre-conditions	● משתמש מחובר ● משתמש מטפל רשום
post-conditions	● כרטיס הלקוח מוצג ● פגישות עבר מוצגות בכרטיס הלקוח
trigger	המטפל מכניס את פרטי הזיהוי של הלקוח בשורת החיפוש
MSS	1. המערכת מחפשת במאגר הלקוחות של המשתמש את כרטיס הלקוח 2. המערכת מציגה את הלקוחות הקיימים ברשימת הלקוחות התואמים לנתונים שהוקלדו 3. המשתמש לוחץ על כרטיס הלקוח הרצוי 4. המערכת מציגה את כרטיס הלקוח עם הפרטים המזהים למטפל 5. המערכת מציגה פגישות קודמות
הסתעפות א'	חלופה בצעד 2 של MSS: לקוח לא קיים 3א1. הודעת שגיאה שהמשתמש לא נמצא ברשימת לקוחות 3א2. הצעה ליצירת כרטיס לקוח חדש 3א3. חזרה לצעד 1
עקיבות לדרישות	תפעוליות: 48,49,50,51,63,64,65 אחרות: 64

SUC-9	כניה לכרטיס לקוח
שחקנים ויעדים	● בעל עסק: חיפוש כרטיס לקוח ותיעוד פגישה
ב"ע ואינטרסים	● בעל עסק: תיעוד פגישות מילוי פרטי לקוח ושמירה במאגר לקוחות
pre-conditions	● משתמש מחובר ● משתמש מטפל רשום
post-conditions	● כרטיס הלקוח מוצג ● פגישות עבר מוצגות בכרטיס הלקוח
trigger	המטפל לוחץ על כרטיס הלקוח לאחר חיפוש (SUC-9)
MSS	1. המערכת מושבת את המידע מהשרת על פגישות קיימות 2. המערכת מציגה את כרטיס הלקוח 3. המשתמש לוחץ על כרטיס הלקוח הרצוי 4. המערכת מציגה את כרטיס הלקוח עם הפרטים המזהים 5. המערכת מציגה פגישות קודמות המערכת נותן אפשרות לערוך פגישה קיימת
עקיבות לדרישות	תפעוליות: 52,48,49,50,63,64,65 אחרות: 64,51

SUC-10	הנפקת דוחות
שחקנים ויעדים	● מטפל: הנפקת דוחות מעקב
ב"ע ואינטרסים	● מטפל: ניהול הכנסות ● מטפל: ניהול לקוחות חדשים\לקוחות פעילים
pre-conditions	● משתמש מחובר ● משתמש מטפל
post-conditions	● הנפקת דוח המאגד את הנתונים הרלוונטיים
trigger	המשתמש נכנס לטאב של "Analytics"
MSS	1. המערכת מעלה את בסיס הנתונים של המשתמש 2. המערכת מנפיקה דוח על פי החודש הנוכחי ומציגה אותו למסך
עקיבות לדרישות	תפעוליות: 53,54,55 אחרות:

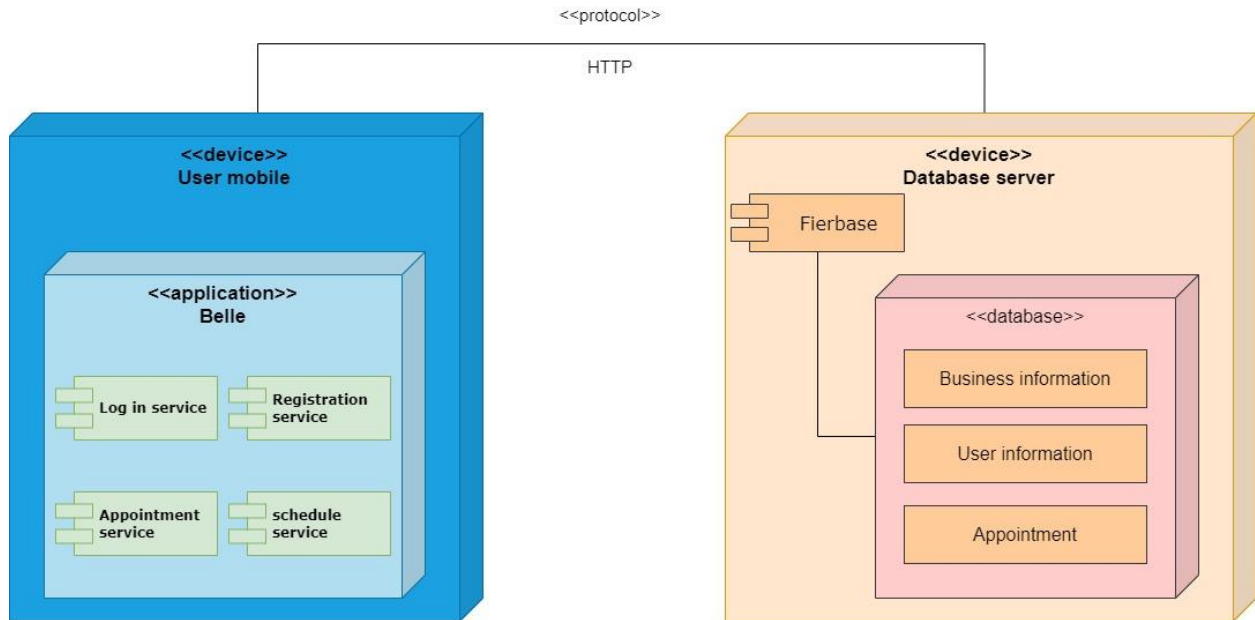
SUC-11	הגדרת יעדים ושעות עבודה
שחקנים ויעדים	● מטפל: הגדרת שעות עבודה ויעדים
ב"ע ואינטרסים	● מטפל: הגדרת שעות לתפוסת פגישה ביומן ● מטפל: הגדרת יעדים לדוחות יומיים
pre-conditions	● מטפל מחובר לפורטל ניהול
post-conditions	● זמן הפגישה מוגדר לפי השעות שהגדיר הלקוח.
trigger	המטפל לוחץ על הטאב "הגדרות" בפורטל הניהול
MSS	(1) המשתמש נכנס להגדרות (2) המשתמש בוחר ב"פגישה" (3) המשתמש מכניס את משך הפגישה הרצוי (4) המשתמש לוחץ על כפתור האישור (5) המערכת מעדכן את אתר BELLE כך שכול הפגישות הנכנסות ייתפסו זמן ביומן בהתאם למספר שהוכנס.
הסתעפות א'	חלופה בצעד 4 של MSS: המספר שהוכנס אינו תואם את הפורמט 3א1. הודעת שגיאה: "נא להכניס אורך" 3א2. הצעה ליצירת כרטיס לקוח חדש 3א3. חזרה לצעד
עקיבות לדרישות	תפעוליות: 66 אחרות: 33,20,22

SUC-12	חיפוש המטפל
שחקנים ויעדים	● לקוח
ב"ע ואינטרסים	● מטפל: שלקוחותימצאו אותו ויקבעו תור ● לקוח: לקבוע תור אצל המטפל הרצוי
pre-conditions	● משתמש מחובר ● הכנסת קלט
post-conditions	● מציאת דף המטפל הרצוי אם קיים
trigger	● הלקוח לוחץ על שורת החיפוש ומקליד את שם המטפל
MSS	1. הלקוח מקליד את שם המטפל 2. המערכת מחפשת במערכת את דף בית העסק על פי קלט המשתמש 3. המערכת נכנסת לדף העסק הרצוי (5-incl. SUC)
הסתעפות א'	חריגה בצעד 2 של MSS: מטפל לא קיים במאגר 1אS. המרכת תציג הודעה שבעל העסק לא קיים במערכת
עקיבות לדרישות	תפעוליות: 16,17,14 אחרות:

5 תרשימי SAD

5.1 ארכיטקטורה פיזית

5.1.1 תרשים פריסה (Deployment diagram)



5.1.2 פרטי חומרה (Nodes)

שם	מהות	עקיבות לדרישות
Database server	שרת הנתונים של המערכת Firebase	
User mobile	טלפון חכם של המשתמש המחובר לאינטרנט	

5.1.3 פרטי תוכנה (Software Artifacts)

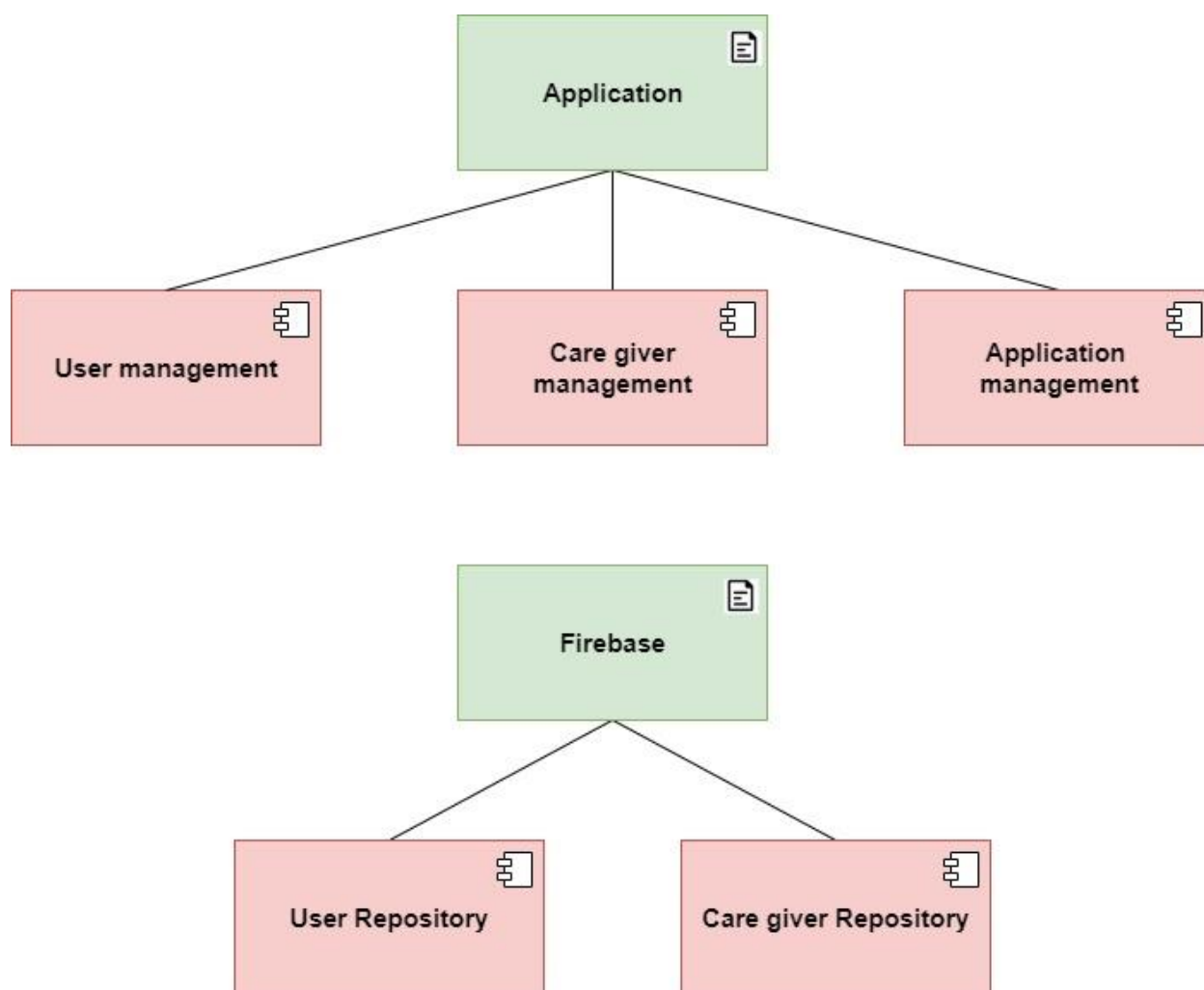
שם	מהות	עקיבות לדרישות
Registration service	הרשמה למערכת Belle	
Log in service	התחברות למערכת Belle	Registration service
Schedule service	קביעת תורים	Log in service
Appointment service	תצוגה של פגישות קיימות במערכת	Log in service, Schedule service

5.1.4 ממשקים פיזיים

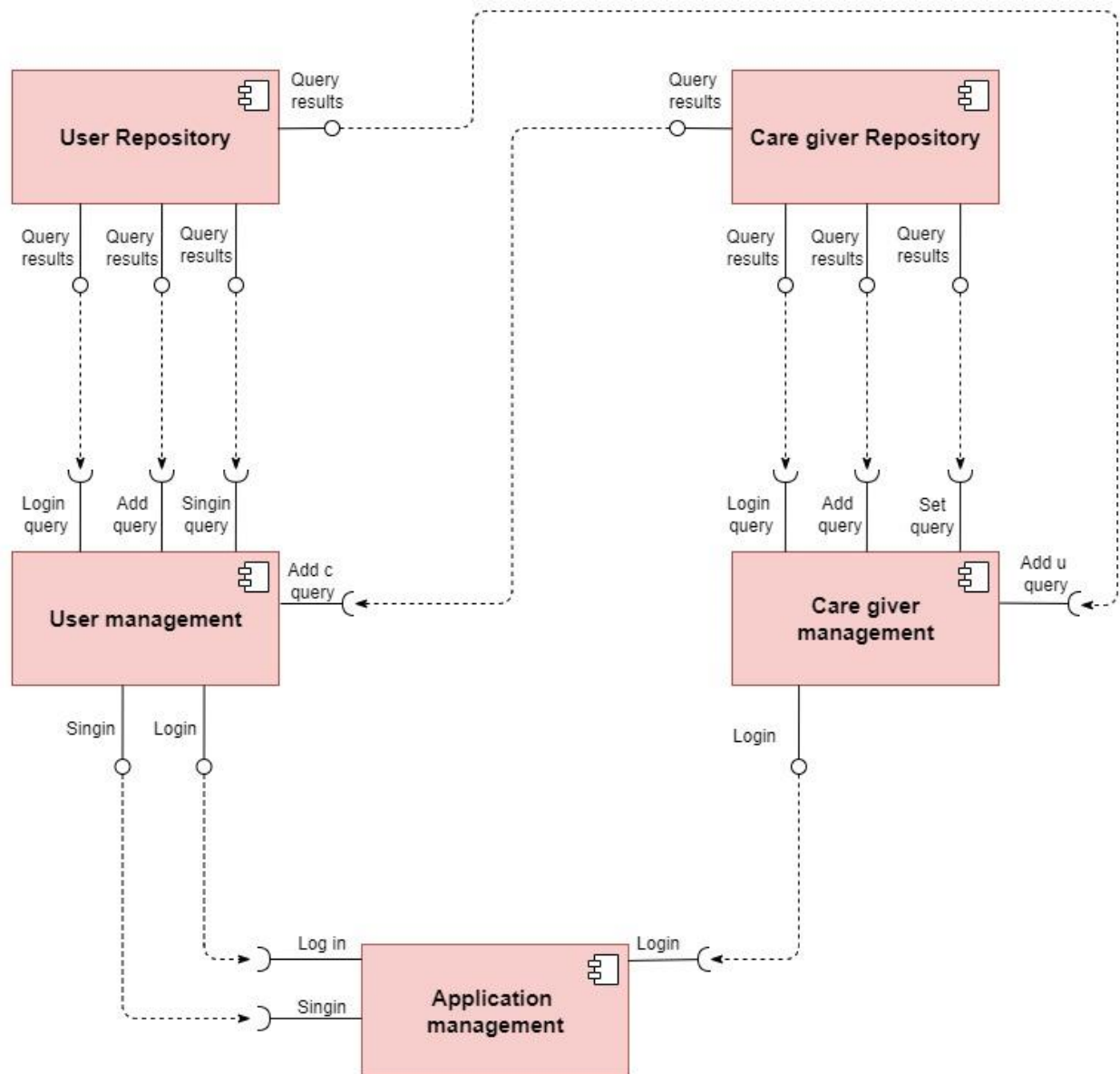
רכיב	רכיב	תכנים	תווך / פרוטוקל	עקיבות לדרישות
Database server	User mobile	> פקודות / שאילתות > עדכון מידע < תצוגת מידע > אימות משתמש	אינטרנט / HTTP	

5.2 ארכיטקטורה לוגית

5.2.1 חלוקה לרכיבי תוכנה (Software Components)



5.2.2 תרשים רכיבים (Component Diagram)



5.2.3 פירוט רכיבים וממשקים

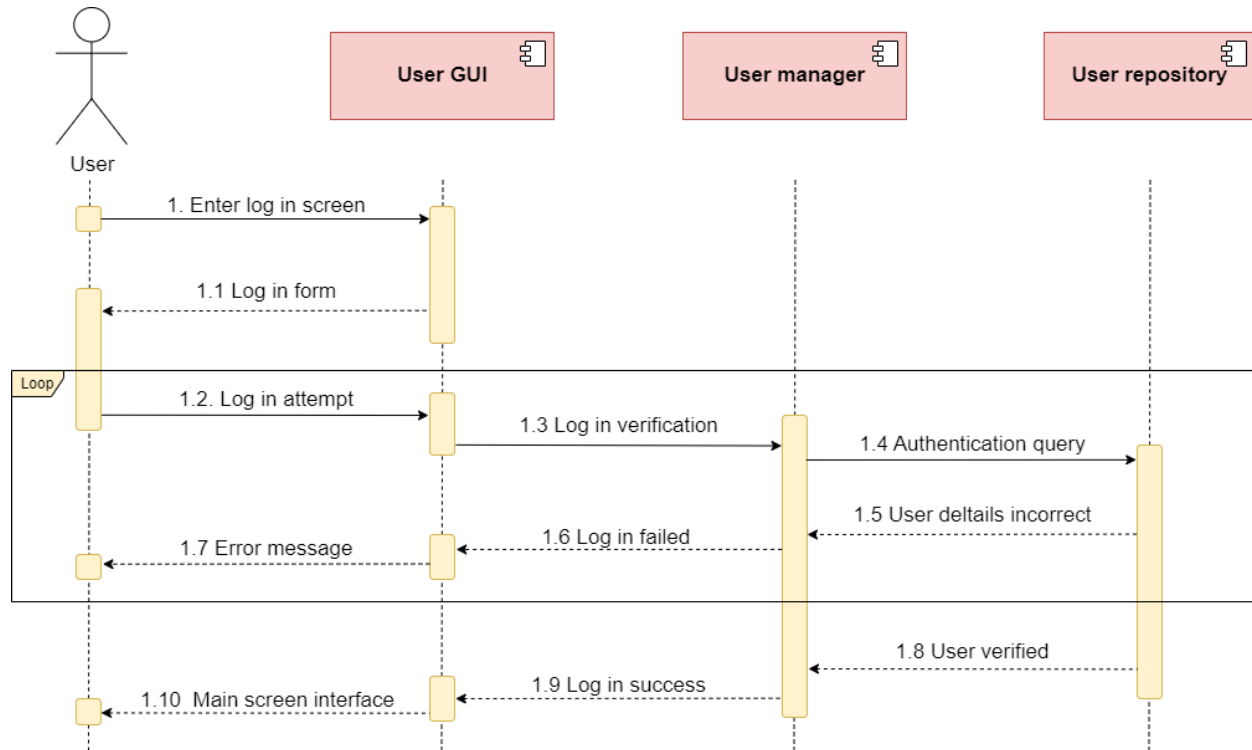
Firebase				
שם	תפקיד	ממשקים		
		סוג	שם	פירוט
User Repository	מאגר נתונים של משתמשי האפליקציה	מסופק	Query results	התחברות המשתמש למערכת
		מסופק	Query results	הוספת פגישה חדשה למשתמש
		מסופק	Query results	הרשמת משתמש חדש למערכת
		מסופק	Query results	עדכן פגישה אצל המשתמש
Care giver Repository	מאגר נתונים של לקוחות האפליקציה	מסופק	Query results	התחברות לקוח למערכת
		מסופק	Query results	עדכון פגישה חדשה במערכת
		מסופק	Query results	עדכון שעות וימי הפעילות

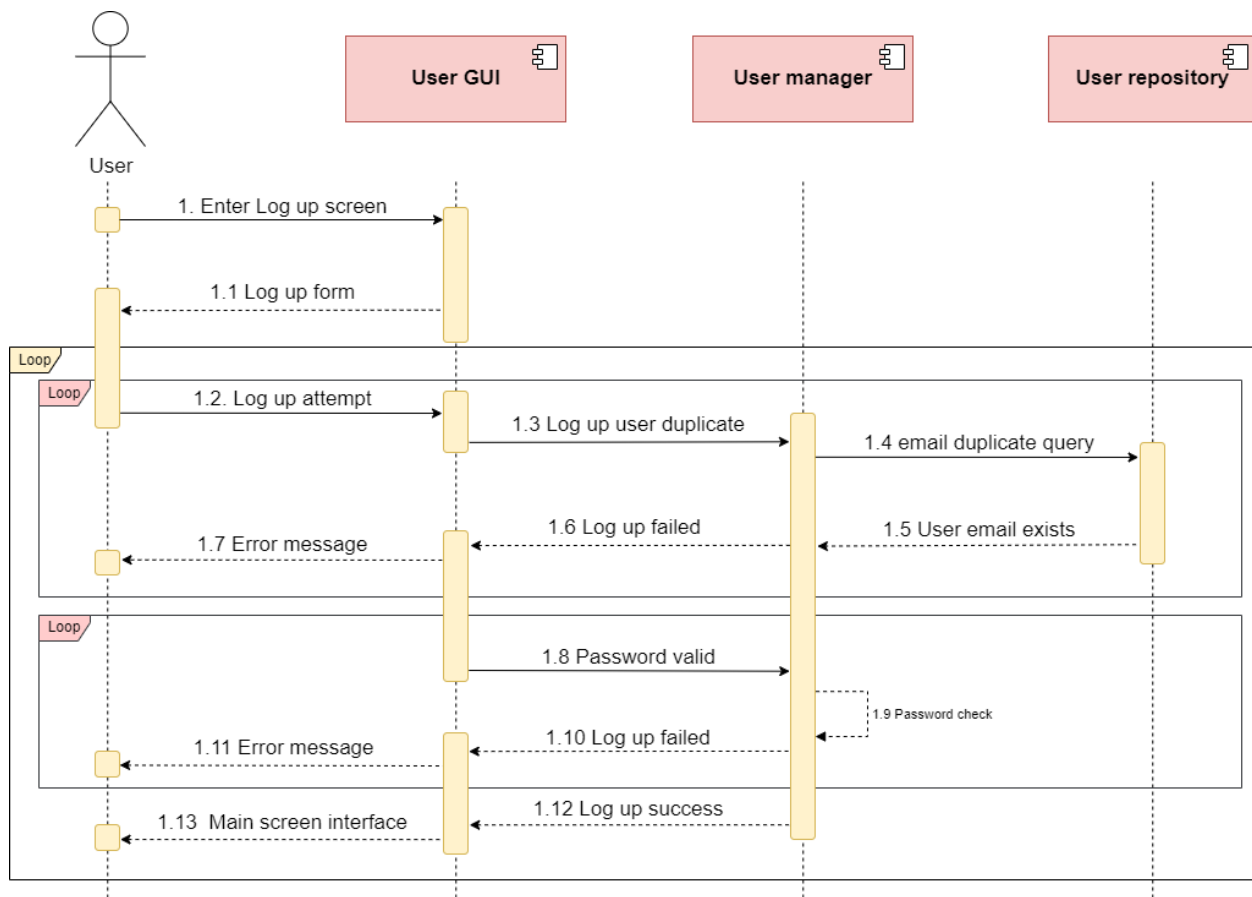
Application management				
שם	תפקיד	ממשקים		
		סוג	שם	פירוט
Application management	ממשק משתמש גרפי	נדרש	Login	התחברות משתמש למערכת
		נדרש	Singin	הרשמת משתמש למערכת
		נדרש	Login	התחברות לקוח למערכת

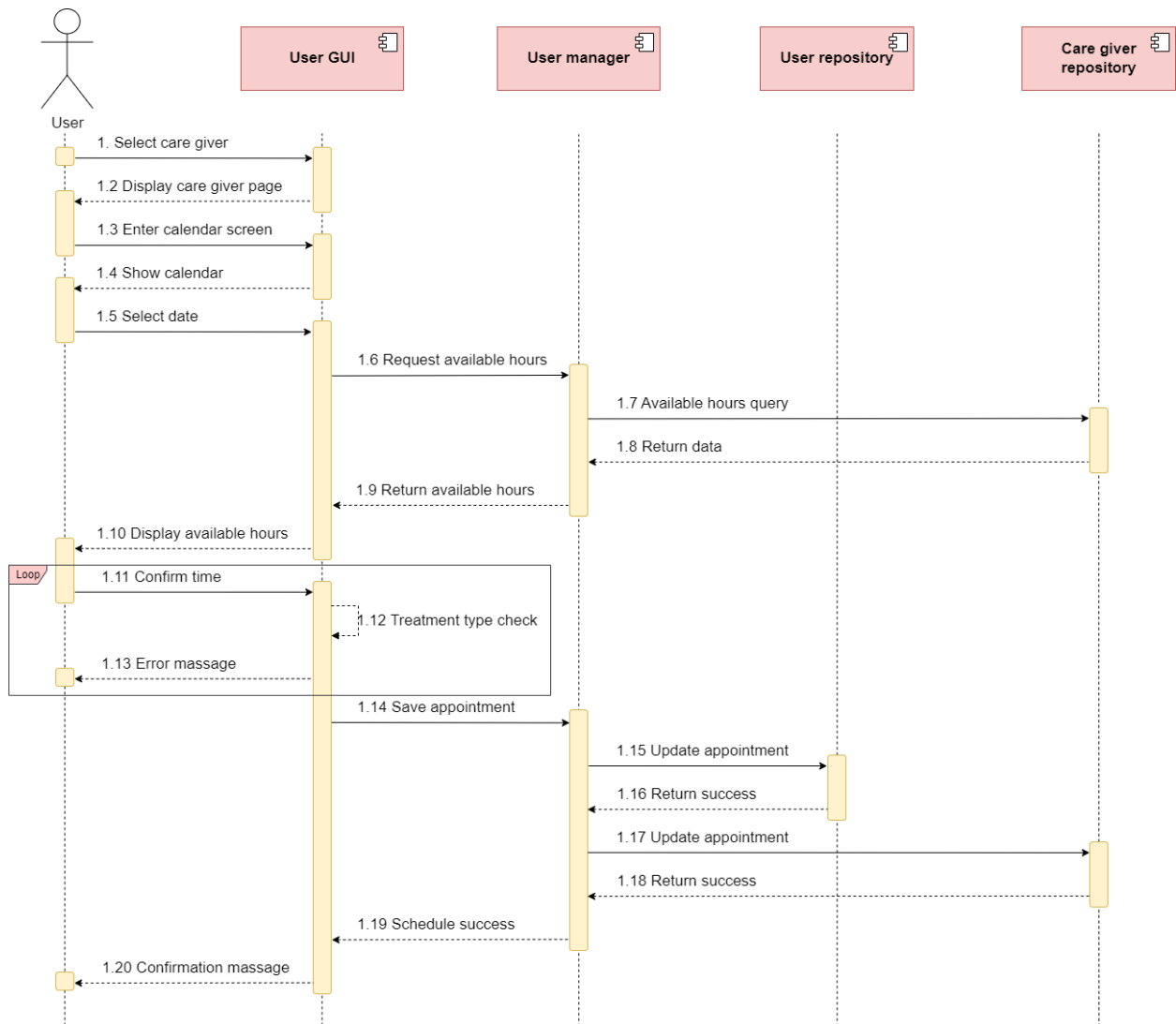
Application				
שם	תפקיד	ממשקים		
		סוג	שם	פירוט
User management	מאגר נתונים של משתמשי האפליקציה	נדרש	Login query	שליחת שאילתה לחיבור משתמש למערכת
		נדרש	Singin query	שליחת שאילתה לרישום משתמש למערכת
		נדרש	Add query	שליחת שאילתה להוספת פגישה חדשה למשתמש
		נדרש	Add c query	שליחת שאילתה להוספת הפגישה אצל העסק
		מסופק	Login	התחברות משתמש למערכת
		מסופק	Singin	הרשמת משתמש חדש למערכת
Care giver management	מאגר נתונים של לקוחות האפליקציה	נדרש	Login query	שליחת שאילתה לחיבור משתמש למערכת
		נדרש	Singin query	שליחת שאילתה לרישום משתמש למערכת
		נדרש	Add query	שליחת שאילתה להוספת פגישה חדשה למשתמש
		נדרש	Add u query	שליחת שאילתה להוספת הפגישה אצל המשתמש
		מסופק	Login	התחברות לקוח למערכת

5.3 תרשימי רצף לתהליכי מערכת (Sequence Diagrams)

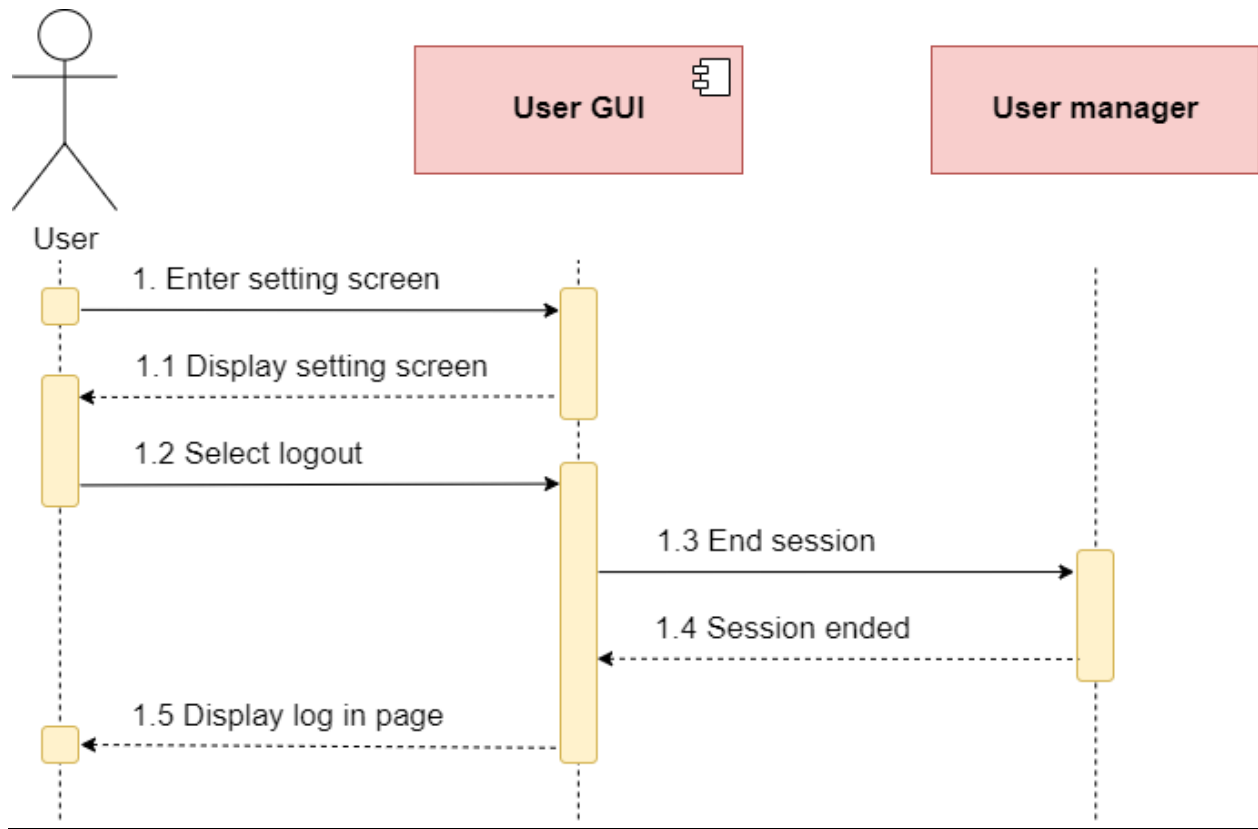
SUC-1



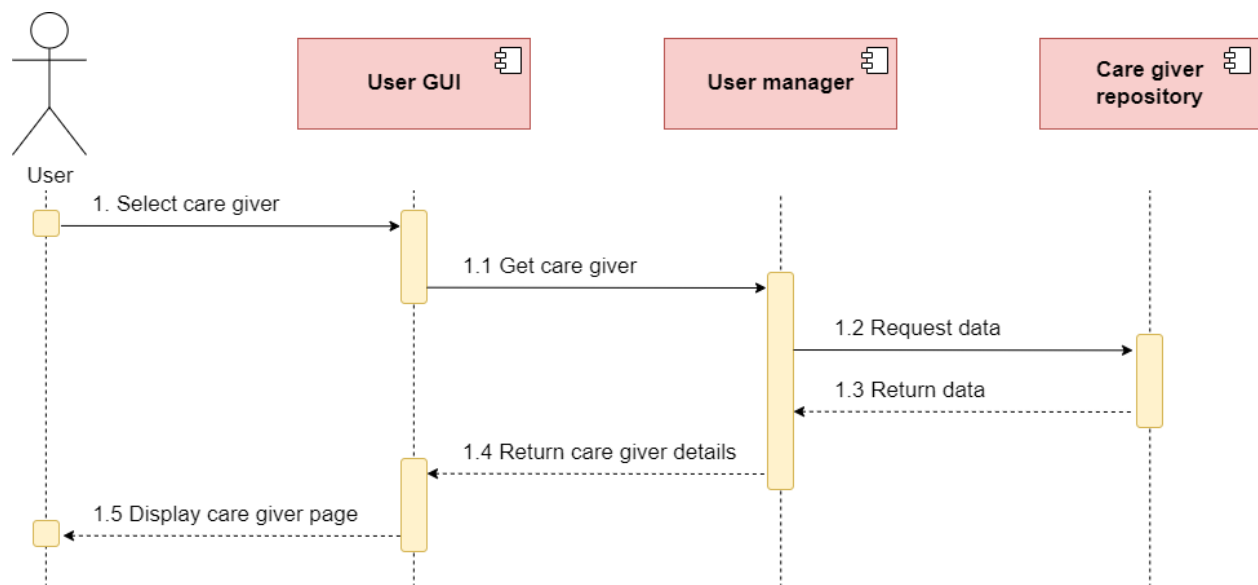


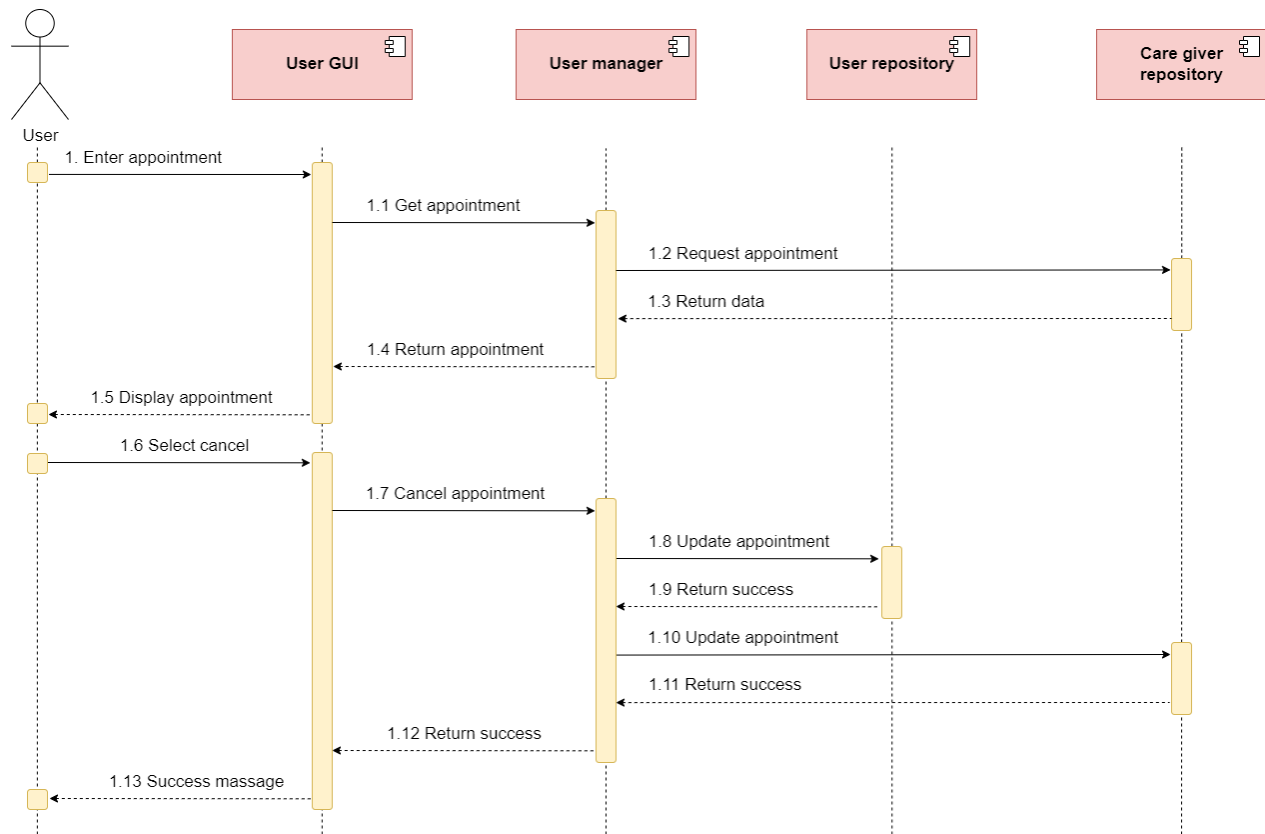


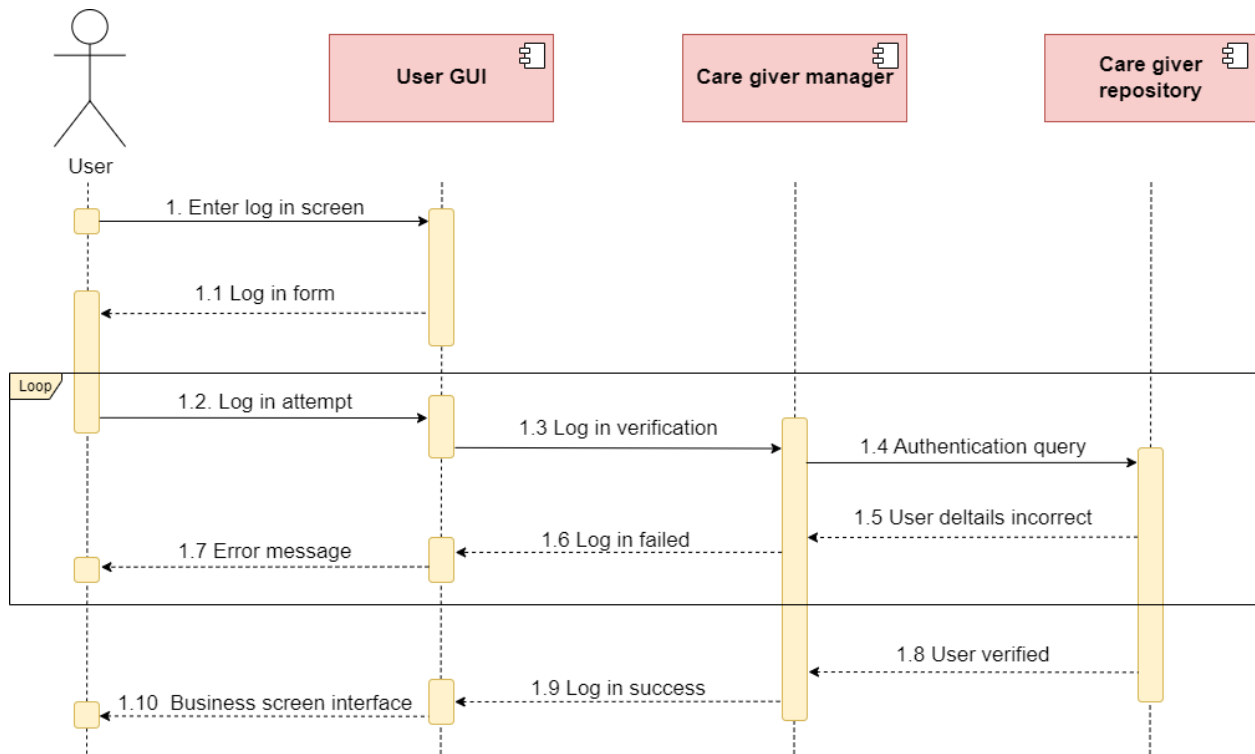
SUC-4

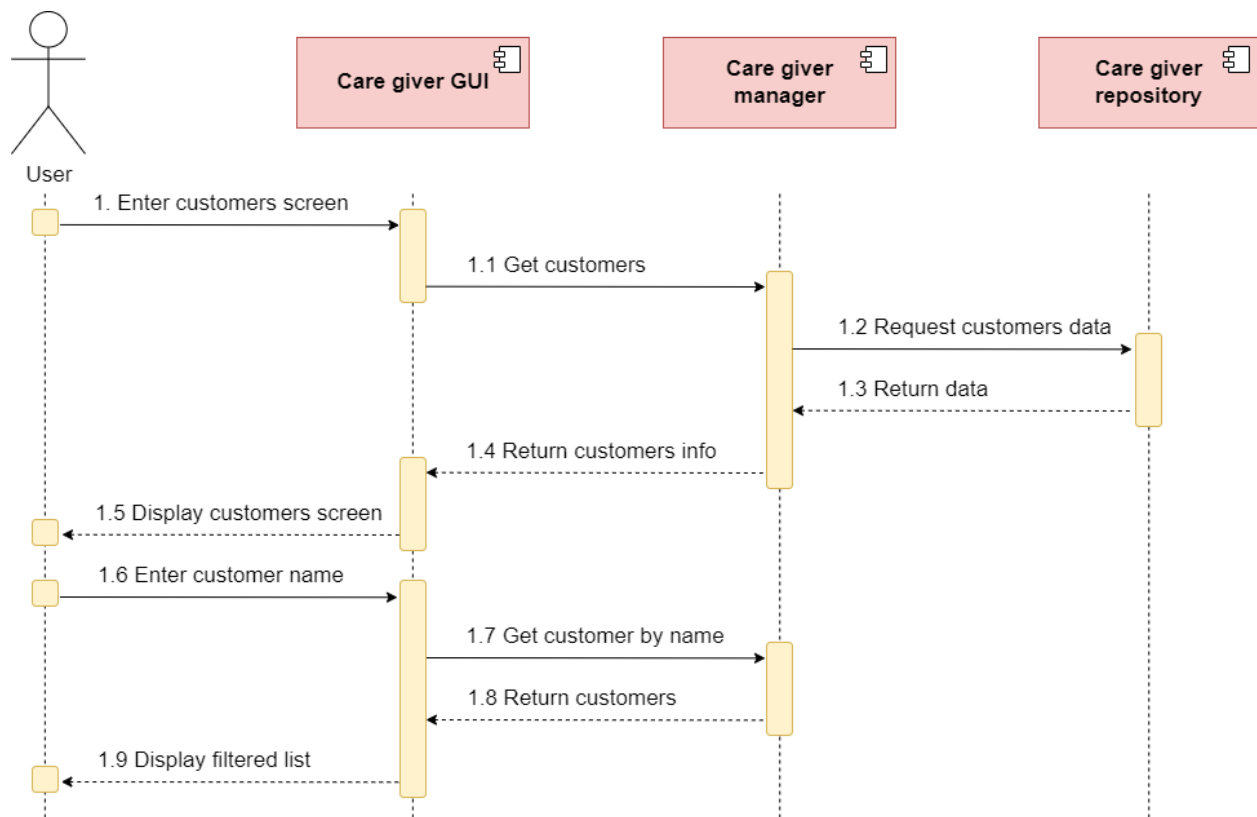


SUC-5

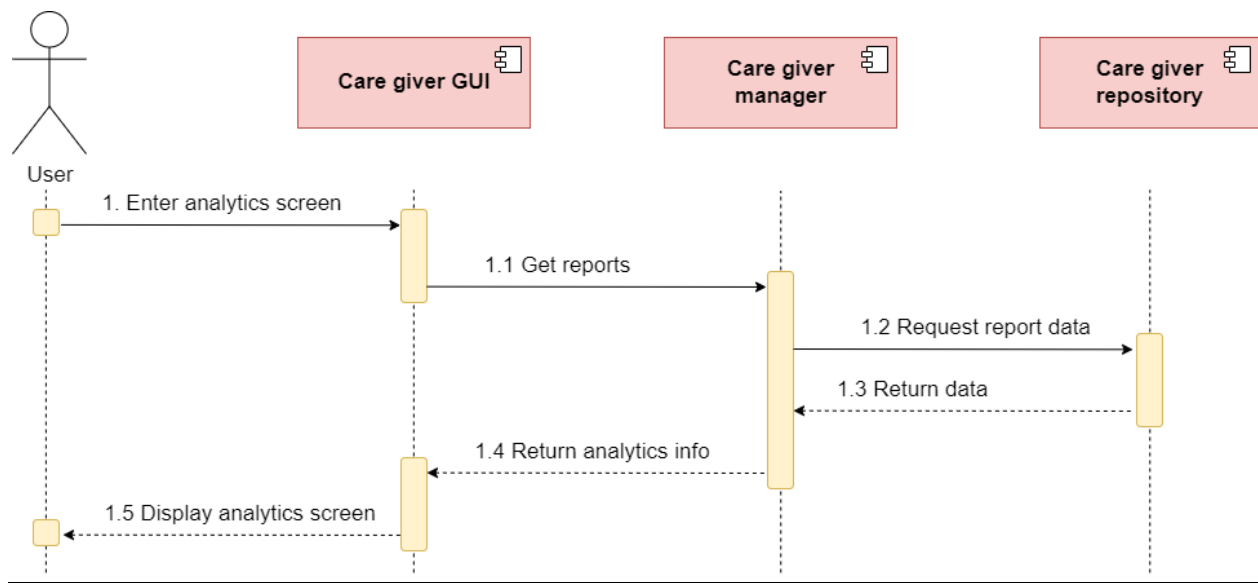




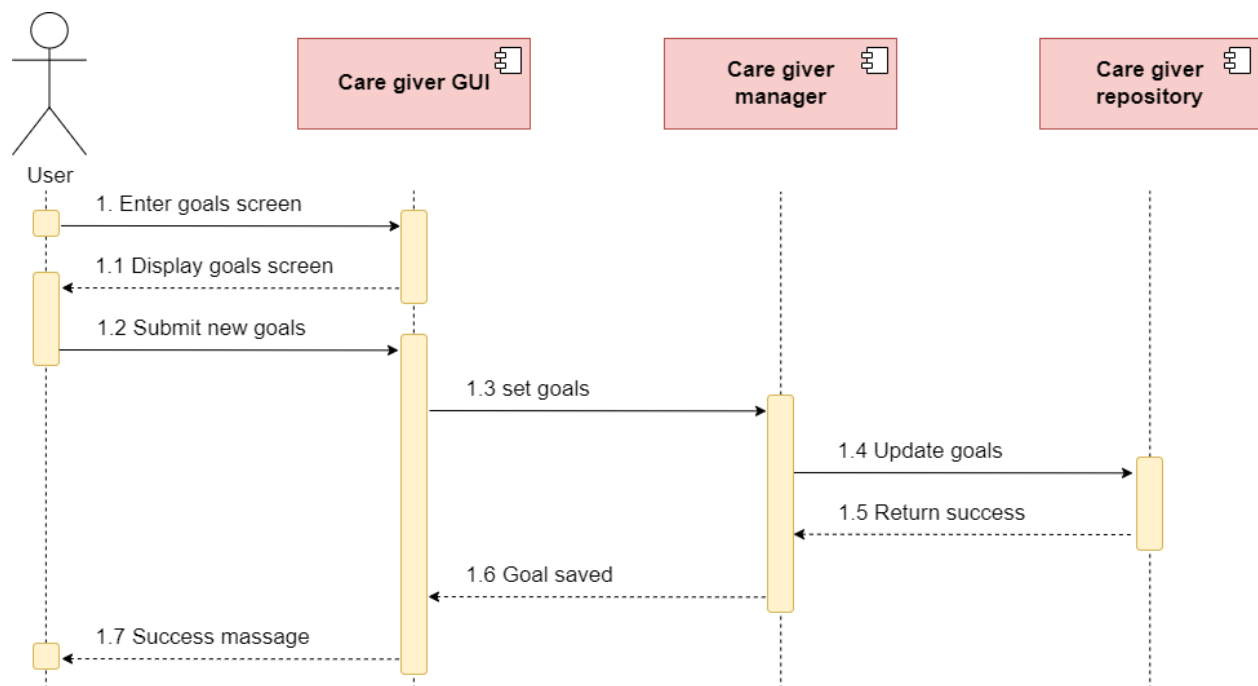




SUC-10



SUC-11



6 מקרי בדיקה / Test Cases

TC 1 - Login					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
1.1	התחברות	המשתמש לוחץ על כפתור ההתחברות במסך הכניסה.	מעבר למסך ההתחברות.	23.02.2023	תקין
1.2		המשתמש מכניס את פרטי ההתחברות איתם נרשם למערכת, כתובת מייל וסיסמה.	התחברות ומעבר למסך הבית.	23.02.2023	
1.3		הלקוח מכניס את פרטי הלקוח שלו ולוחץ על כפתור התחבר כלקוח.	התחברות ומעבר למסך ניהול עסק.	23.02.2023	
1.4		המשתמש מכניס פרטים שגויים במסך ההתחברות.	השארות במסך ההתחברות עם הפועה של הודעת שגיאה.	23.02.2023	
1.5		המשתמש מכניס את פרטי ההתחברות איתם נרשם למערכת, כתובת מייל וסיסמה ולוחץ על כפתור התחבר כלקוח.	השארות במסך ההתחברות עם הפועה של הודעת שגיאה.	23.02.2023	

TC 2 – Sing-up					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
2.1	הרשמה	המשתמש לוחץ על כפתור ההרשמה במסך הכניסה.	מועבר למסך ההרשמה.	23.02.2023	תקין
2.2		המשתמש מכניס פרטים אישיים ובנוסף מייל וסיסמה.	התחברות ומעבר למסך הבית.	23.02.2023	
2.3		המשתמש מכניס פרטים שגויים במסך ההתחברות.	השארות במסך ההרשמה עם הפועה של הודעת שגיאה	23.02.2023	

TC 3 - Schedule					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
3.1	קביעת פגישה	המשתמש לוחץ על בית העסק	מועבר למסך בית העסק.	23.02.2023	תקין
3.2		המשתמש לוחץ על Schedule.	המשתמש מועבר ללוח שנה המציג את הפגישות האפשריות.	23.02.2023	
3.3		המשתמש בוחר את כל הפרטים הרלוונטיים ולחץ אישור.	המשתמש מקבל הודעה כי הפגישה נרשמה.	23.02.2023	
3.4		המשתמש לוחץ אישור ללא לבחור את כל שדות החובה.	המשתמש מקבל הודעת שגיאה כי עליו למלא את כל השדות	23.02.2023	

TC 4 – User menu					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
4.1	תפריט ניווט משתמש	המשתמש לוחץ על אייקון	מועבר למסך הצגת בתי עסק.	23.02.2023	תקין
4.2		המשתמש לוחץ על אייקון המפה.	מועבר למסך המפה המציג את בתי העסק באזור.	23.02.2023	
4.3		המשתמש לוחץ על אייקון ההגדרות	מועבר למסך ההגדרות מערכת.	23.02.2023	

TC 4 – User menu					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
4.1	תפריט ניווט משתמש	המשתמש לוחץ על אייקון	מועבר למסך הצגת בתי עסק.	23.02.2023	תקין
4.2		המשתמש לוחץ על אייקון המפה.	מועבר למסך המפה המציג את בתי העסק באזור.	23.02.2023	
4.3		המשתמש לוחץ על אייקון ההגדרות	מועבר למסך ההגדרות מערכת.	23.02.2023	

TC 5 – User setting					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
5.1	מסך הגדרות	המשתמש לוחץ על כפתור התנתקות.	המערכת מנתקת את המשתמש ויוצאת למסך ההתחברות.	23.02.2023	תקין
5.2		המשתמש לוחץ על פגישות.	מועבר למסך המציג את כלל הפגישות של המשתמש.	23.02.2023	
4.3		** להוסיף **		23.02.2023	

TC 6 – Care giver clients					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
6.1	מסך הלקוחות של בית העסק	הלקוח לוחץ על אייקון הלקוחות.	מועבר למסך ניהול לקוחות.	23.02.2023	תקין
6.2		המשתמש בוחר מטופל.	מועבר למסך המציג את פרטי הלקוח והטיפול של הלקוח.	23.02.2023	
6.3		** להוסיף **		23.02.2023	

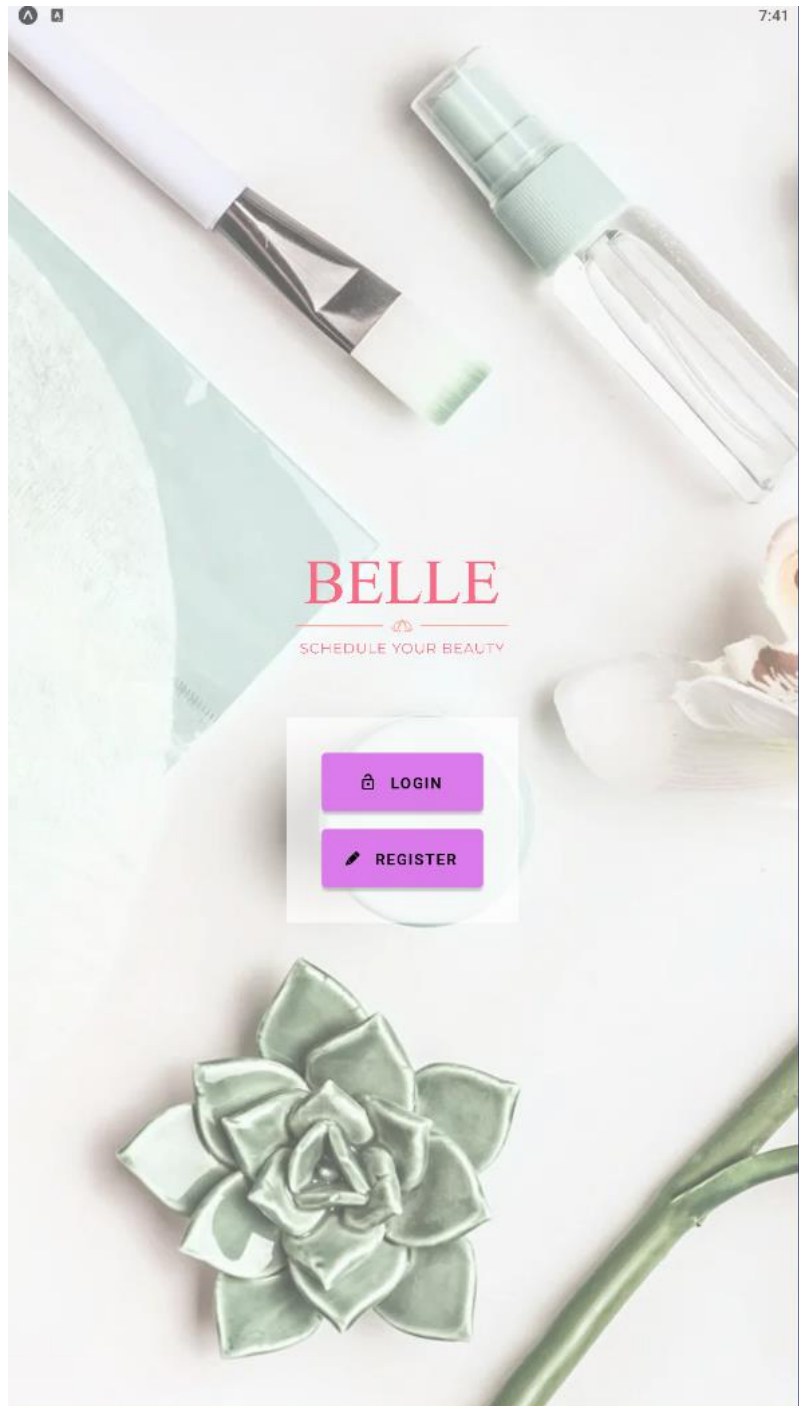
TC 7 – Care giver schedule					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
7.1	בית עסק מוסיף פגישה ללקוח	הלקוח לוחץ על המשתמש מרשימת המטופלים.	מועבר למסך פרטי הלקוח.	23.02.2023	תקין
7.2		הלקוח לוחץ כל כפתור קביעת תור.	מועבר למסך לוח שנה לקביעת תאריך ושעה.	23.02.2023	
7.3		הלקוח ממלא את הפרטים הרלוונטיים ולוחץ אישור.	מתקבלת הודעה כי הפגישה נרשמה.	23.02.2023	
7.4		הלקוח לא ממלא את כל השדות החובה בטופס ומנסה ללחוץ אישור.	מתקבלת הודעה כי לא כל שדות החובה מולאו.	23.02.2023	

TC 8 – Care giver reports					
מספר בדיקה	תיאור הבדיקה	תהליך הבדיקה	תוצאה רצויה	תאריך בדיקה	תוצאה
8.1	מסך	הלקוח לוחץ על אייקון הדוחות.	מועבר למסך הדוחות.	23.02.2023	תקין
8.2	דוחות בית העסק	הלקוח בוחר חודש ולוחץ אישור.	מציג את הדוחות הרלוונטיים לחודש הנבחר.	23.02.2023	

7 פירוט מסכי מערכת

7.1 מסכי כניסה והתחברות

7.1.1 מסך כניסה



```

export const AccountScreen = ({ navigation }) => {
  return (
    <AccountBackground>
      <AccountCover />

      <Image
        style={styles.logo}
        source={require("../../../assets/logo.png")}
      />
      <AccountContainer>
        <AuthButton
          title="Login"
          onPress={() => {
            navigation.navigate("Login");
          }}
        >
          Login
        </AuthButton>
        <Spacer size="large" />
        <RegisterButton
          title="Register"
          onPress={() => {
            navigation.navigate("Register");
          }}
        >
          Register
        </RegisterButton>
      </AccountContainer>
    </AccountBackground>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center",
  },
  logo: {
    width: 200,
    height: 200,
    resizeMode: "contain",
  },
});

```



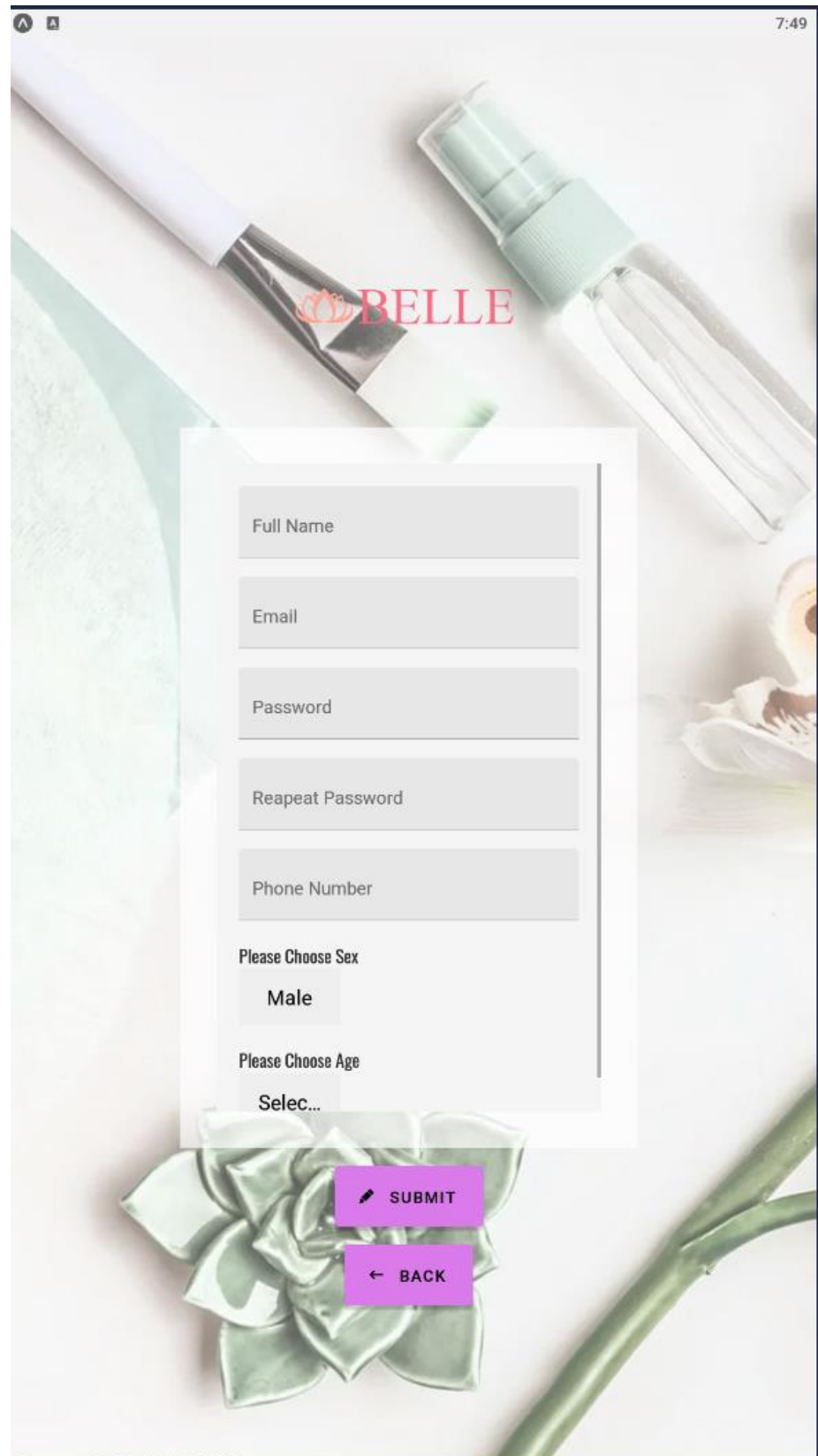
```

return (
  <LoginBackground>
    <LoginCover />
    <Image
      style={styles.logo}
      source={require("../assets/logo.png")}
    />
    <LoginContainer>
      <TextField
        label="Email"
        value={userName}
        onChangeText={(t) => setUserEmail(t)}
      />
      <Spacer size="large" />
      <TextField
        label="Password"
        value={password}
        secureTextEntry={true}
        onChangeText={(t) => setPassword(t)}
        secure={true}
      />
      {error && (
        <Spacer size="large">
          <Text variant="error">{error}</Text>
        </Spacer>
      )}
      <Spacer size="large" />
      <Spacer size="large" />
      {isLoading ? (
        <ActivityIndicator size="small" color="#0000ff" />
      ) : (
        <Submit title="Login" onPress={onSubmit}>
          Login
        </Submit>
      )}
      <Spacer size="large"></Spacer>
      {isLoading ? (
        <ActivityIndicator size="small" color="#0000ff" />
      ) : (
        <SubmitPartner title="Login As Partner" onPress={onSubmitPartner}>
          Login As Partner
        </SubmitPartner>
      )}
    </LoginContainer>
    <Spacer size="large" />

```

```
        <Back title="Register" onPress={onBack} icon="keyboard-backspace">
          Back
        </Back>
      </LoginBackground>
    );
  };
  const styles = StyleSheet.create({
    container: {
      flex: 1,
      alignItems: "center",
      justifyContent: "center",
    },
    logo: {
      width: 200,
      height: 200,
      resizeMode: "contain",
    },
  });
});
```

7.1.3 מסך הרשמה



Full Name

Email

Password

Repeat Password

Phone Number

Please Choose Sex

Male

Please Choose Age

Selec...

SUBMIT

← BACK

```

return (
  <SafeArea>
    <RegisterBackground>
      <RegisterCover />
      <Image
        style={styles.logo}
        source={require("../../../../../assets/MIni_logo.png")}
      />
      <RegisterContainer>
        <StyledScrollView>
          <TextField
            label="Full Name"
            value={name}
            onChangeText={(t) => setName(t)}
          />
          <Spacer size="large" />
          <TextField
            label="Email"
            value={userEmail}
            onChangeText={(t) => setUserEmail(t)}
          />
          <Spacer size="large" />
          <TextField
            label="Password"
            value={password}
            onChangeText={(t) => setPassword(t)}
            secureTextEntry={true}
          />
          <Spacer size="large" />
          <TextField
            label="Reapeat Password"
            value={reapeatedPassword}
            onChangeText={(t) => setRepeatedPassword(t)}
            secureTextEntry={true}
          />
          <Spacer size="large" />
          <TextField
            label="Phone Number"
            value={phone}
            onChangeText={(t) => setPhone(t)}
            keyboardType="numeric"
          />
          <Spacer size="large" />
          <Text>Please Choose Sex</Text>
          <SelectDropdown

```



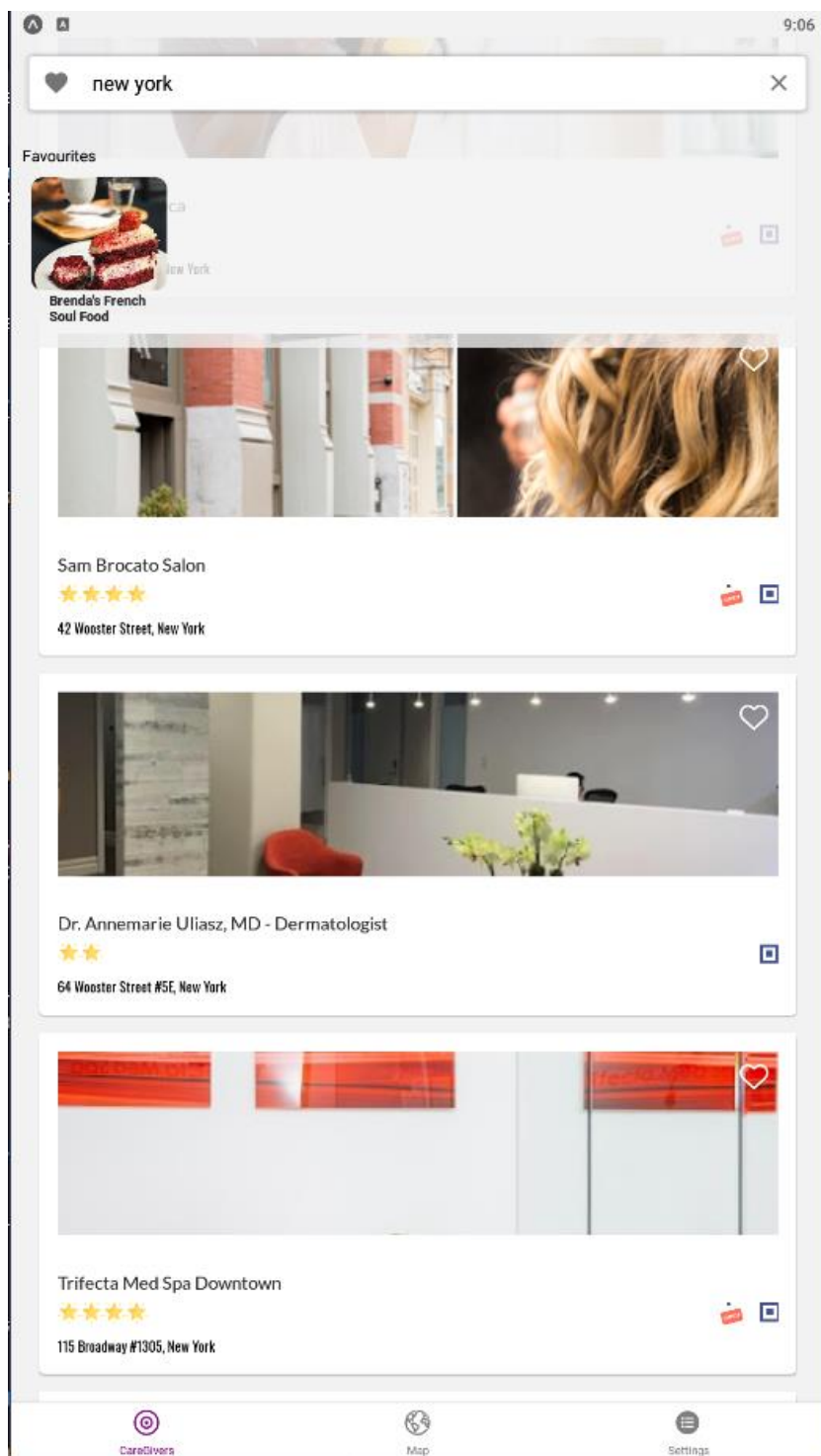
```

        data={Sexlist}
        defaultValue={sex}
        onSelect={(selectedItem, index) => setSex(selectedItem)}
        buttonStyle={sty}
        dropdownStyle={{ backgroundColor: "#fafafa" }}
        rowStyle={{ justifyContent: "flex-start" }}
    />
    <Spacer size="large" />
    <Text>Please Choose Age</Text>
    <SelectDropdown
        data={ageItems}
        defaultValue={age}
        onSelect={(selectedItem, index) => setAge(parseInt(selectedItem))}
        buttonStyle={sty}
        dropdownStyle={{ backgroundColor: "#fafafa" }}
        rowStyle={{ justifyContent: "flex-start" }}
    />
    <Spacer size="large" />
</StyledScrollView>
</RegisterContainer>
{error && (
    <ErrorContainer size="large">
        <Text variant="error">{error}</Text>
    </ErrorContainer>
)}
{isLoading ? (
    <ActivityIndicator size="small" color="#0000ff" />
) : (
    <Spacer size="large">
        <Submit title="Register" onPress={onSubmit}>
            Submit
        </Submit>
    </Spacer>
)}
<Spacer size="large" />
<Back title="Register" onPress={onBack} icon="keyboard-backspace">
    Back
</Back>
</RegisterBackground>
</SafeArea>
);
};

```

7.2 מסכי משתמש

7.2.1 מסך ראשי



```

export const CareGiversScreen = ({ navigation }) => {
  const { error: locationError } = useContext(LocationContext);
  const { CareGivers, isLoading, Featured, error } =
    useContext(CareGiversContext);
  const [isToggled, setIsToggled] = useState(false);
  const { favourites } = useContext(FavouritesContext);

  const hasError = locationError || error;
  const Wrap = Platform.OS === "android" ? TransAndroid : TransIOS;
  const onPress = () => {
    setIsToggled(!isToggled);
  };

  return (
    <SafeArea>
      <Wrap>
        <Search isFavouritesToggled={isToggled} onFavouritesToggle={onPress} />
        {isToggled && (
          <FavouriteBar
            favourites={favourites}
            onNavigate={navigation.navigate}
          />
        )}
        <RecommendedBar
          recommended={Featured}
          onNavigate={navigation.navigate}
        />
      </Wrap>
      {hasError && (
        <ErrorV>
          <Spacer position="left" size="large">
            <Text variant="error">
              Something went wrong retrieving the data
            </Text>
          </Spacer>
        </ErrorV>
      )}

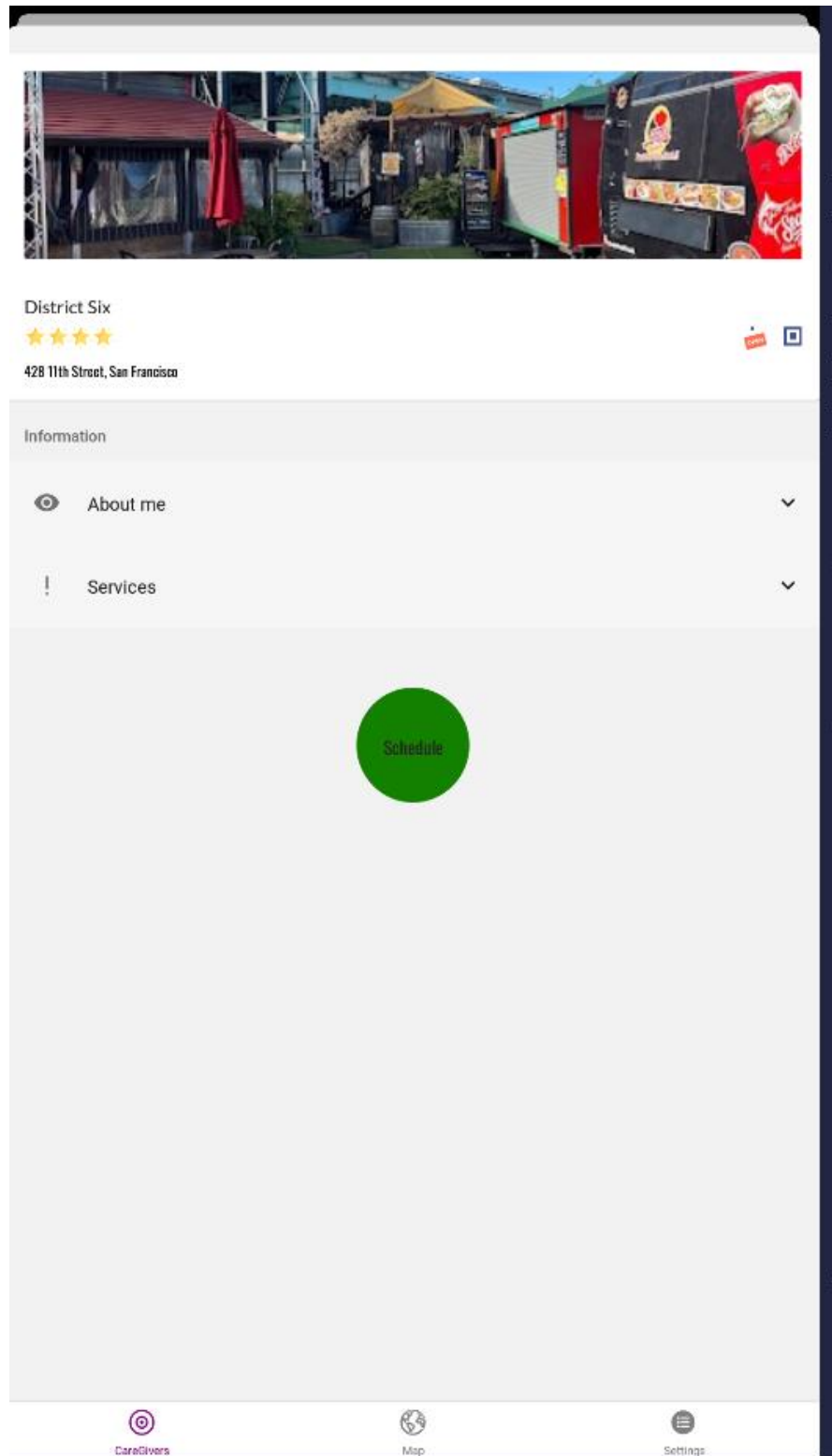
      {!hasError && !isLoading ? (
        <CareGiverList
          data={CareGivers}
          renderItem={({ item }) => {
            return (
              <TouchableOpacity
                onPress={() =>

```

```

        navigation.navigate("CareGiverDetail", {
            CareGiver: item,
            navigation: navigation,
        })
    }
}
>
<FaceInView>
    <Spacer position="bottom" size="large">
        <CareGiverInfoCard CareGivers={item} />
    </Spacer>
</FaceInView>
</TouchableOpacity>
);
}}
keyExtractor={(item) => item.name}
/>
) : (
    <Activity animating={isLoading} color={Colors.blue300} size={"large"} />
)}
</SafeArea>
);
};

```



```

return (
  <SafeArea>
    <ScrollView>
      <CareGiverInfoCard CareGivers={CareGiver} />
      {useParams.isInProgram && (
        <List.Section title="Information">
          <List.Accordion
            title="About me"
            left={(props) => <List.Icon {...props} icon="eye" />}
          >
            <View>
              <Text variant="body">{CareGiver.name}</Text>
              <Text variant="caption">{useParams.about}</Text>
            </View>
          </List.Accordion>
          <List.Accordion
            title="Services"
            left={(props) => <List.Icon {...props} icon="exclamation" />}
          >
            {useParams.types.map((item) => (
              <List.Item key={item} title={item} />
            ))}
          </List.Accordion>
        </List.Section>
      )}
      <ScheduleWrap>
        <Schedule
          title="Schedule"
          onPress={() =>
            navigation.navigate("Schedule", {
              info: CareGiver,
              navigation: navigation,
            })
          }
        >
          <Text>Schedule</Text>
        </Schedule>
      </ScheduleWrap>
    </ScrollView>
  </SafeArea>
);
};

```

7.2.3 דף קביעת תור

March 2023

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1

Please Pick Service Type

Select Type

↶ BACK
SCHEDULE ➤

```

export const CareGiverScheduleScreen = ({ route, navigation }) => {
  const [selected, setSelected] = useState(new Date());
  const [startTime, setStartTime] = useState(new Date());
  const [endTime, setEndTime] = useState(undefined);
  const [timeSelected, setTimeSelected] = useState(undefined);
  const [loading, setLoading] = useState(false);
  const [status, setStatus] = useState("send");
  const [selectedType, setSelectedType] = useState("Select Type");

  const Auth = useContext(AuthenticationContext); //get user setting
  const schedulerInfo = useContext(SchedulerContext); // fetch Data From Context
  const onBack = () => {
    navigation.goBack();
  };

  const showError = (errorMessage) => {
    Alert.alert(
      "Error",
      errorMessage,
      [
        {
          text: "OK",
          onPress: () => console.log("OK pressed"),
          style: "cancel",
        },
      ],
      { cancelable: false }
    );
  };

  const ScheduleSuccess = (errorMessage) => {
    Alert.alert(
      "Status",
      errorMessage,
      [
        {
          text: "OK",
          onPress: () => onBack(),
          style: "cancel",
        },
      ],
      { cancelable: false }
    );
  };

  const onSchedule = () => {

```



```

    if (selectedType === "Select Type") {
      showError("You have not selected type!");
      return;
    }
    setLoading(true);
    setTimeout(() => {
      schedulerInfo.schedule(
        Auth.user.uid,
        info.placeId,
        timeSelected.getTime(),
        selectedType,
        setStatus,
        ScheduleSuccess
      );
      setLoading(false);
    }, 1000);
  };
  const { info } = route.params;

  useEffect(() => {
    //initialize Context with current scheduler
    schedulerInfo.changeID(info.placeId);
  }, [route]);

  const onChoose = (day) => {
    //var workingHours= getfunction call a get function to retrieve the specific
    time the caregiver works on.
    //const workingHours = working_hours[day.getDay()];

    const start = new Date(day.dateString);
    schedulerInfo.changeDate(start, info.placeId);

    const workingHours = schedulerInfo.working_hours[start.getDay()]; //get the
    working hours of the chosen day to render timeframes

    start.setHours(
      parseInt(workingHours.substring(0, 2)),
      parseInt(workingHours.substring(2, 4)),
      0
    );

    const end = new Date(day.dateString);

    end.setHours(
      parseInt(workingHours.substring(5, 7)),

```

```

        parseInt(workingHours.substring(7, 9)),
        0
    );

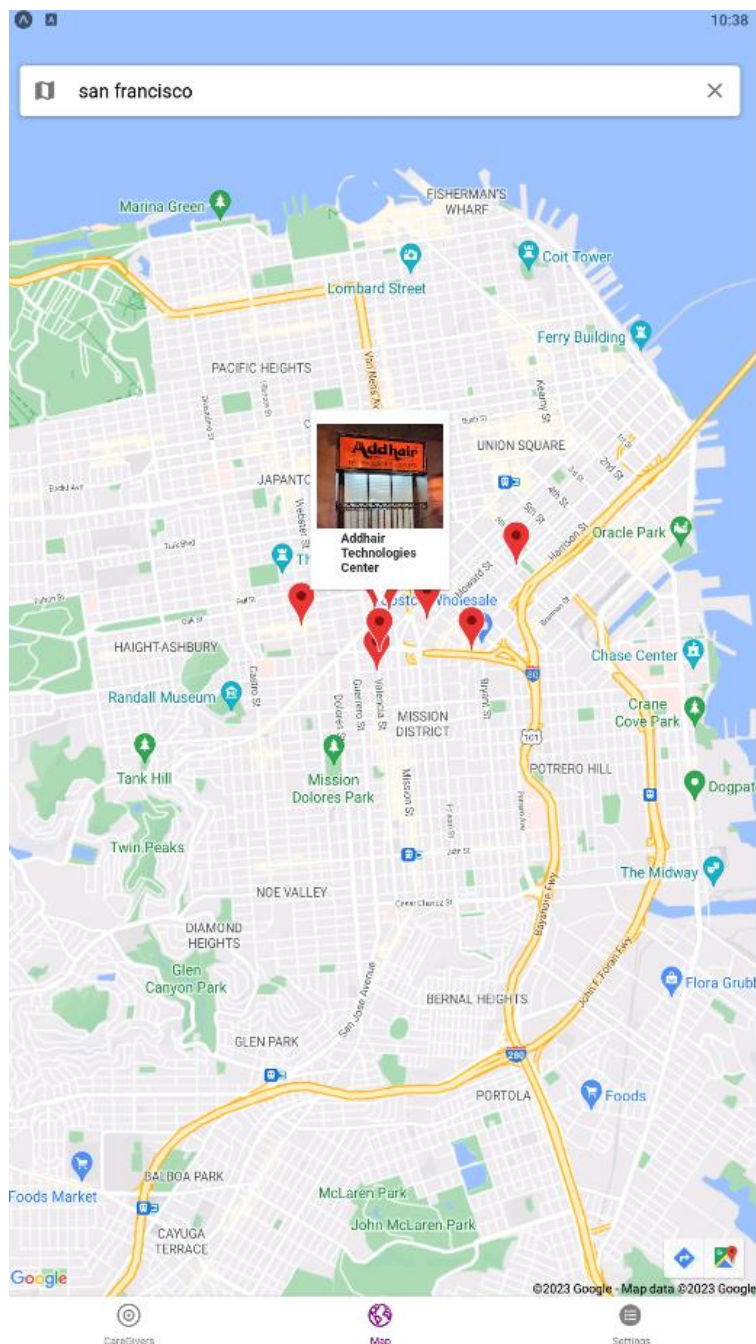
    setStartTime(start); // current date and time
    setEndTime(end); // end time
};

return (
    <SafeArea>
        <ScheduleCalendar
            workingDays={schedulaerInfo.working_hours}
            update={setSelected}
            select={onChoose}
            markedDates={schedulaerInfo.markedDates}
        />
        <ScrollView>
            <Text>{info.placeId}</Text>
            <TimeSlotButtons
                startTime={startTime}
                endTime={endTime}
                interval={60}
                setTimeSelected={setTimeSelected}
                appointments={schedulaerInfo.appointments}
            />
        </ScrollView>
        <PickerContainer>
            <Text>Please Pick Service Type</Text>
            <Picker
                selectedValue={selectedType}
                onValueChange={(itemValue, itemIndex) => {
                    setSelectedType(itemValue);
                }}
                enableHapticFeedback={true}
            >
                <Picker.Item
                    key={"Select Type"}
                    label={"Select Type"}
                    value={"Select Type"}
                />
                {schedulaerInfo.types.map((type) => (
                    <Picker.Item key={type.id + 1} label={type} value={type} />
                ))}
            </Picker>
        </PickerContainer>

```

```
<ButtonsContainer>
  <BackButton onClick={onBack} />
  {timeSelected && (
    <ScheduleButton
      loading={loading}
      onClick={onSchedule}
      status={status}
    />
  )}
</ButtonsContainer>
</SafeArea>
);
};
```

7.2.4 דף מפה



```

export const MapScreen = ({ navigation }) => {
  const { location } = useContext(LocationContext);
  const { CareGivers = [] } = useContext(CareGiversContext);
  const [latDelta, setLatDelta] = useState(0);
  const { lat, lng, viewport } = location;

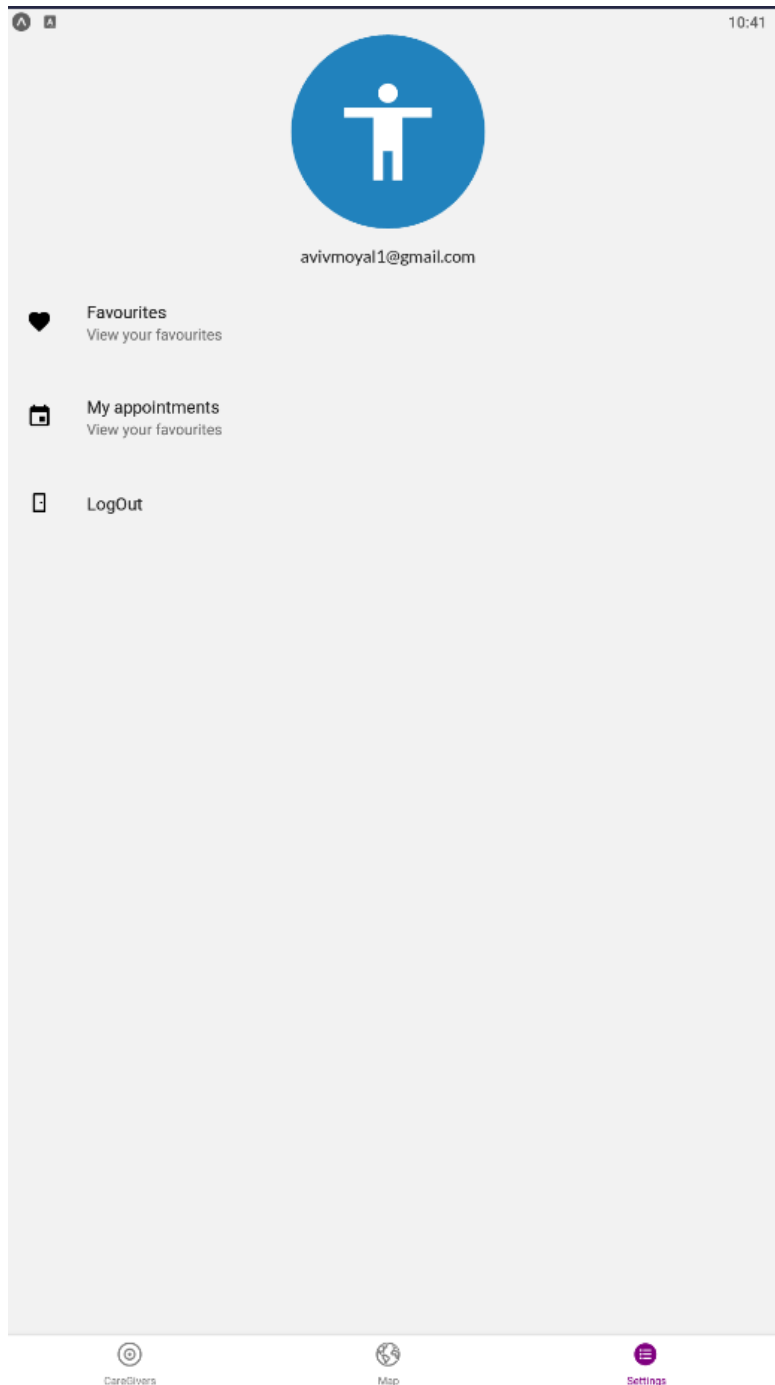
  useEffect(() => {
    const northeastLat = viewport.northeast.lat;
    const southwestLat = viewport.southwest.lat;
    const latDelta = northeastLat - southwestLat;
    setLatDelta(latDelta);
  }, [location, viewport]);

  return (
    <>
      <Search />
      <Map
        region={{
          latitude: lat,
          longitude: lng,
          latitudeDelta: latDelta,
          longitudeDelta: 0.02,
        }}
      >
        {CareGivers.map((CareGiver) => {
          return (
            <Marker
              key={CareGiver.name}
              title={CareGiver.name}
              coordinate={{
                latitude: CareGiver.geometry.location.lat,
                longitude: CareGiver.geometry.location.lng,
              }}
            >
              <Callout
                onPress={() =>
                  navigation.navigate("CareGiverDetail", {
                    CareGiver,
                  })
                }
              >
                <MapCallout CareGiver={CareGiver} />
              </Callout>
            </Marker>
          );
        })}
      </Map>
    </>
  );
}

```

```
    }  
  }  
  </Map>  
</>  
);  
};
```

7.2.5 דף הגדרות



```

return (
  <SafeArea>
    <AvatarContainer>
      <Avatar.Icon size={180} icon="human" backgroundColor="#2182BD" />
      <Spacer position="top" size="large">
        <Text variant="label">{user.email}</Text>
      </Spacer>
    </AvatarContainer>

    <List.Section>
      <SettingItem
        style={{ padding: 16 }}
        title="Favourites"
        description="View your favourites"
        left={({props}) => <List.Icon {...props} color="black" icon="heart" />}
        onPress={() => navigation.navigate("Favourites")}
      />
      <SettingItem
        style={{ padding: 16 }}
        title="My appointments"
        description="View your favourites"
        left={({props}) => (
          <List.Icon {...props} color="black" icon="calendar" />
        )}
        onPress={() => navigation.navigate("My Appointments")}
      />
      <SettingItem
        style={{ padding: 16 }}
        title="LogOut"
        left={({props}) => <List.Icon {...props} color="black" icon="door" />}
        onPress={() => press()}
      />
    </List.Section>
  </SafeArea>
);
};

```




10:43

← Favourites



Brenda's French Soul Food



652 Polk Street, San Francisco



District Six



428 11th Street, San Francisco



TAYLOR / MONROE



448 Grove Street, San Francisco



CareOvers

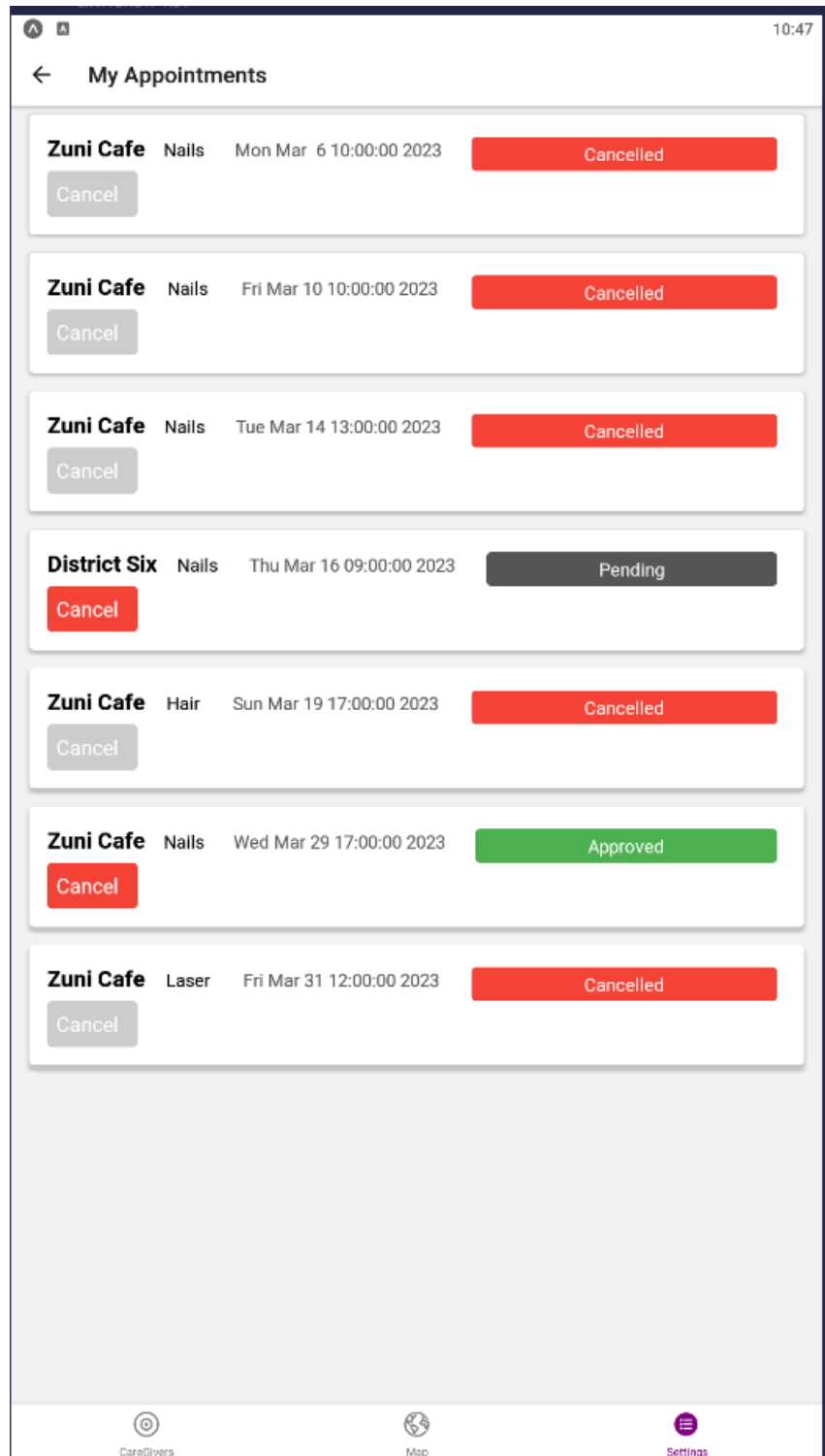


Map



Settings

7.2.7 דף תורים של המשתמש



```

export const AppointmentList = ({ navigation }) => {
  const { getAppointment, appointments, setAppointments } = useContext(
    AuthenticationContext
  );
  const { cancelAppointment } = useContext(SchedulerContext);
  const [flag, setFlag] = useState(false);
  useEffect(() => {
    getAppointment();
  }, [flag]);

  const cancel = async (appointment) => {
    await cancelAppointment(
      appointment.care,
      appointment.customer,
      appointment.date._seconds,
      appointment.id
    );
    setTimeout(() => {
      //set time out the code went too fast and did not wait for the updated
appointment list
      setFlag(!flag);
    }, 1000);
  };

  const renderAppointment = ({ item }) => {
    const time = new Date(item.date._seconds * 1000);
    const title = item.name;
    const type = item.type;
    const status = item.status;
    const cancelled = item.cancelled;
    const approved = item.approved;
    const statusLabel = cancelled
      ? "Cancelled"
      : approved
      ? "Approved"
      : "Pending";
    return (
      <AppointmentItem>
        <AppointmentTitle>{title}</AppointmentTitle>
        <AppointmentType>{type}</AppointmentType>
        <AppointmentTime>{time.toLocaleString()}</AppointmentTime>
        <Status status={status} cancelled={cancelled} approved={approved}>
          {statusLabel}
        </Status>
        <CancelButton onPress={() => cancel(item)} disabled={cancelled}>

```

```

        <CancelButtonText>Cancel</CancelButtonText>
      </CancelButton>
    </AppointmentItem>
  );
};

return (
  <View>
    {appointments.length === 0 ? (
      <Text>No appointments found</Text>
    ) : (
      <FlatList
        data={appointments}
        keyExtractor={(item) => item.date._seconds}
        renderItem={renderAppointment}
      />
    )}
  </View>
);
};

```

7.3 מסכי לקוח

7.3.1 יומן פגישות

11:13

March 2023

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1

Nails withAt 12:00:00CancelCancelled

nails with Itamar Nahum SimhaAt 17:00:00Confirm!CancelPending

gL7h8xKXDacQe7LA5STaEakZXy93

Dashboard

Analytics

Calendar

Customers

Settings

```

export const PartnerAppointmentList = ({ navigation }) => {
  const { appointments, stat } = useContext(PartnerSchedulerContext);
  const { cancelAppointment, confirmAppointment } = useContext(
    PartnerSchedulerContext
  );
  const [flag, setFlag] = useState(false);
  const cancel = async (appointment) => {
    await cancelAppointment(
      appointment.care,
      appointment.customer,
      appointment.date._seconds,
      appointment.id
    );
    setTimeout(() => {
      //set time out the code went too fast and did not wait for the updated
appointment list
      setFlag(!flag);
    }, 1000);
  };

  const approve = async (appointment) => {
    confirmAppointment(appointment);
  };

  const renderAppointment = ({ item }) => {
    const time = new Date(item.date._seconds * 1000);
    const title = item.name;
    const type = item.type;
    const status = item.status;
    const cancelled = item.cancelled;
    const approved = item.approved;
    const statusLabel = cancelled
      ? "Cancelled"
      : approved
      ? "Approved"
      : "Pending";
    return (
      <AppointmentItem>
        <AppointmentType>
          {type} with {item.customerName}
        </AppointmentType>
        <AppointmentType> At {time.toLocaleTimeString()}</AppointmentType>
        {approved === false && cancelled == false ? (
          <ConfirmButton onPress={() => approve(item)}>
            <ButtonText>Confirm!</ButtonText>

```

```

        </ConfirmButton>
      ) : (
        <></>
      )}

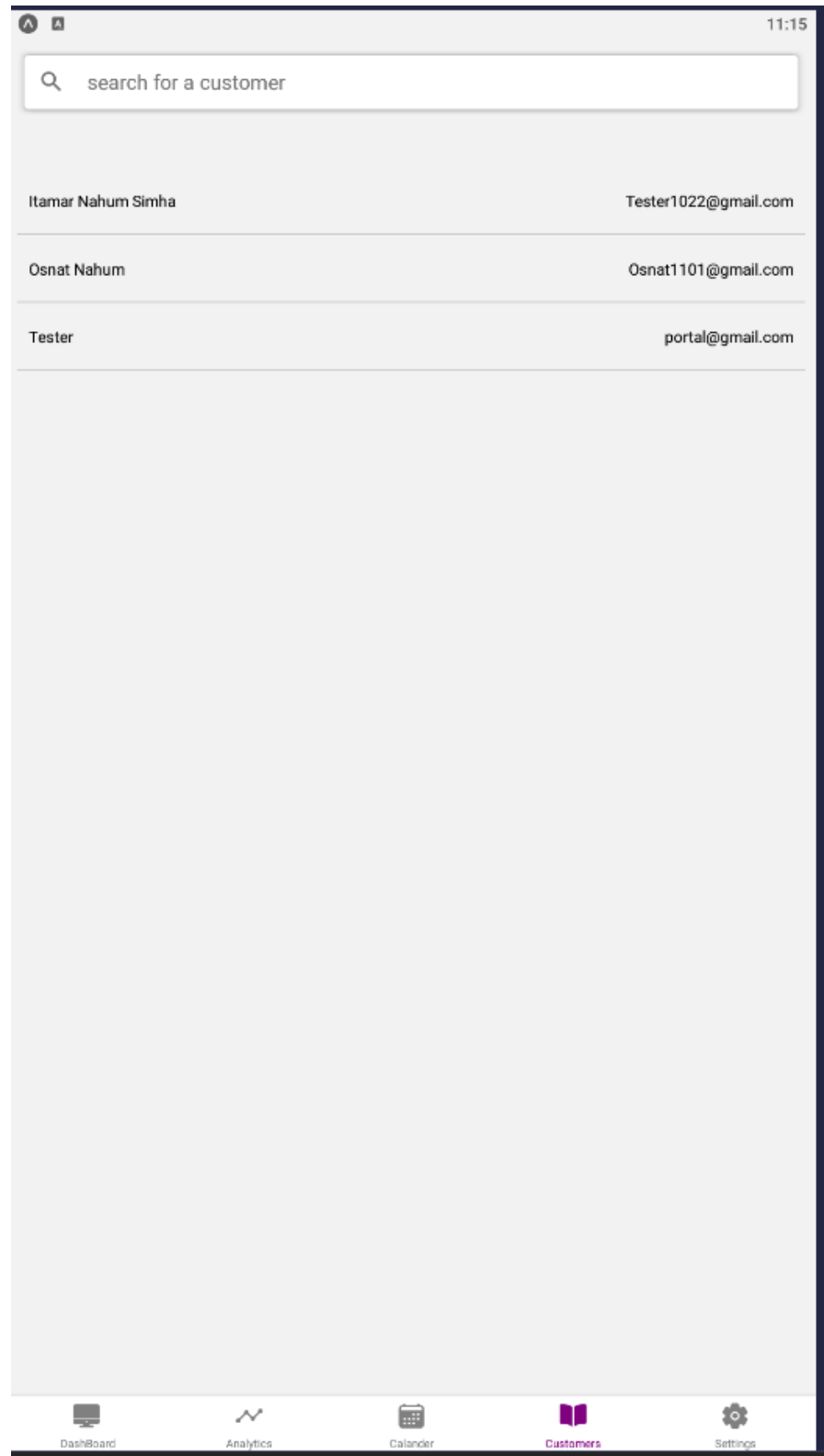
      <CancelButton onPress={() => cancel(item)} disabled={cancelled}>
        <CancelButtonText>Cancel</CancelButtonText>
      </CancelButton>

      <Status status={status} cancelled={cancelled} approved={approved}>
        {statusLabel}
      </Status>
    </AppointmentItem>
  );
};

return (
  <View>
    {appointments.length === 0 ? (
      <Text>No appointments found</Text>
    ) : (
      <FlatList
        data={appointments}
        keyExtractor={(item) => item.date._seconds}
        renderItem={renderAppointment}
      />
    )}
  </View>
);
};

```

7.3.2 מאגר לקוחות




```

const CustomerLastAppointmentDate = styled.Text``;

export const CustomerList = ({ onPress }) => {
  const { customersFiltered } = useContext(CustomersContext);

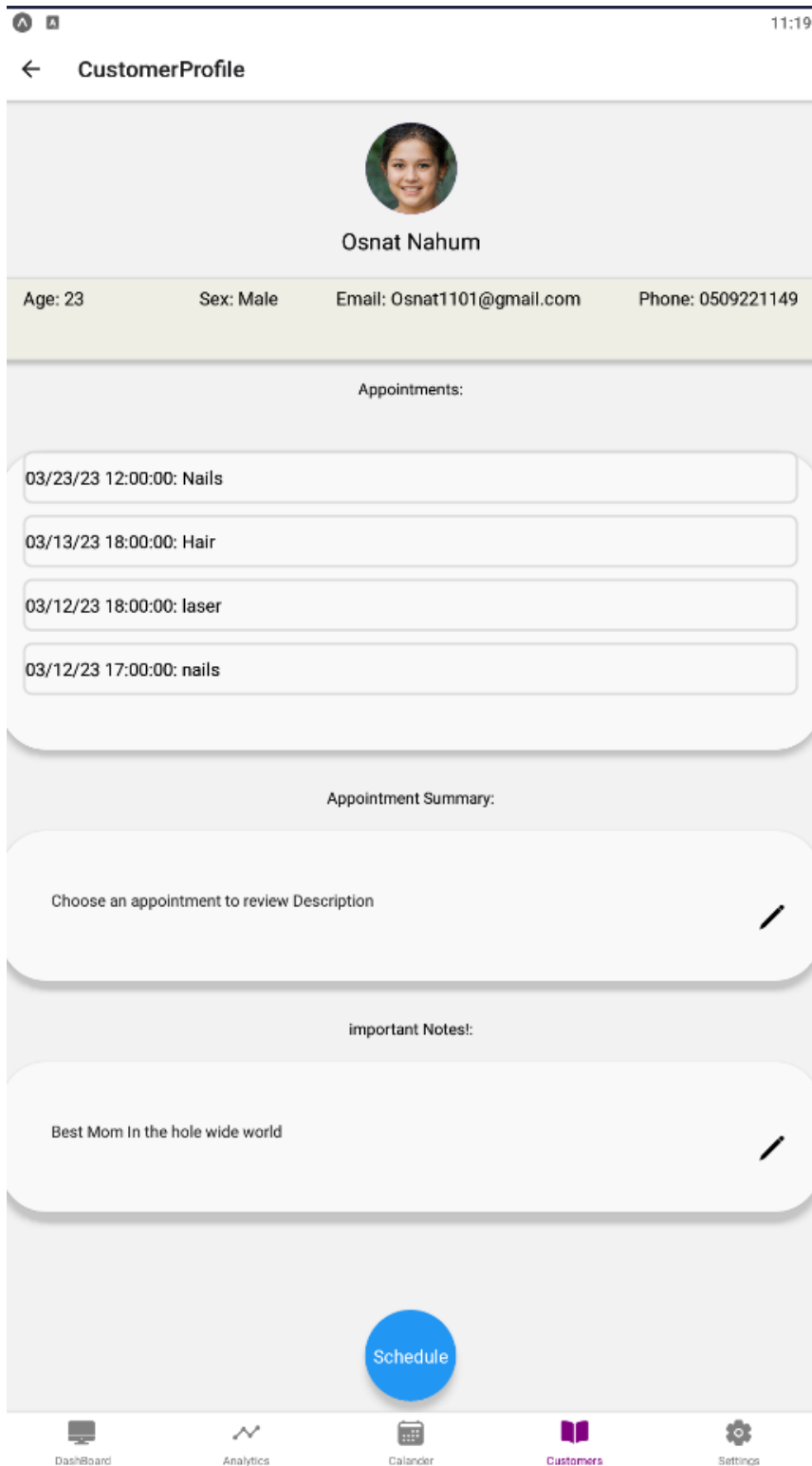
  const handlePress = (customer) => {
    console.log("Customer pressed:", customer);
    onPress(customer);
  };

  const renderItem = ({ item }) => {
    return (
      <CustomerItemContainer onPress={() => handlePress(item)}>
        <CustomerName>{item.customer}</CustomerName>
        <CustomerLastAppointmentDate>{item.Email}</CustomerLastAppointmentDate>
      </CustomerItemContainer>
    );
  };

  return (
    <CustomerListContainer>
      <FlatList
        data={customersFiltered}
        renderItem={renderItem}
        keyExtractor={(item) => item.customer}
      />
    </CustomerListContainer>
  );
};

```

7.3.3 ברטיס מטופל



```

useEffect(() => {
  setChosen(undefined);
  setDescription("Choose an appointment to review Description");
}, []);

const handleNav = () => {
  navigation.navigate("Scheduler", {
    customer: customer,
    placeId: storeID,
  });
};

return (
  <Container>
    <AvatarContainer>
      <Avatar.Image
        size={80}
        source={{
          uri: "https://this-person-does-not-exist.com/gen/avatar-
11849bbf3c0f23ba109dc65f7b58eee8.jpg",
        }}
      />
      <Name>{customer.customer}</Name>
    </AvatarContainer>
    <DetailsContainer>
      <DetailsText>Age: {customer.age}</DetailsText>
      <Spacer position="top" size="large" />
      <DetailsText>Sex: {customer.sex}</DetailsText>
      <DetailsText>Email: {customer.Email}</DetailsText>
      <DetailsText>Phone: {customer.phone}</DetailsText>
    </DetailsContainer>

    <Spacer position="top" size="large" />
    <Text>Appointments:</Text>
    <ContainerAppointmentsSummary>
      <AppointmentsList customer={customer} place={storeID} />
      <Spacer position="top" size="large" />
      <Spacer position="top" size="large" />
      <Text>Appointment Summary:</Text>
      <Summary />
      <Spacer position="top" size="large" />
      <Spacer position="top" size="large" />
      <Text>important Notes!:</Text>
      <Notes customer={customer}></Notes>
    </ContainerAppointmentsSummary>
  </Container>
);

```

```
    <ScheduleButton onPress={() => handleNav()}>
      <ScheduleButtonText>Schedule</ScheduleButtonText>
    </ScheduleButton>
  </Container>
);
};
```

7.3.4 דף הגדרות סוגי טיפול ומחירון

← Services

Please Add or Remove a Service

Halr

Price: 65

REMOVE

nails

Price: 13

REMOVE

Nails

Price: 65

REMOVE

laser

Price: 90

REMOVE

Add a new service:

Enter service name

Price: Enter

ADD

SUBMIT

Dashboard

Analytics

Calendar

Customers

Settings

```

export const SettingTypesScreen = () => {
  const [services, setServices] = useState(["nails", "laser", "Hair"]);
  const [Prices, setPrices] = useState([50, 30, 25]);
  const [newService, setNewService] = useState("");
  const [newServicePrice, setNewServicePrice] = useState("");
  const { types, prices, updateTypes, setResponseFlag } = useContext(
    PartnerSchedulerContext
  );
  const handleAddService = () => {
    if (newService.trim() && newServicePrice.trim()) {
      setServices([...services, newService]);
      setPrices([newServicePrice]);
      setNewService("");
      setNewServicePrice("");
    }
  };

  const handleRemoveService = (service) => {
    const serviceIndex = services.indexOf(service); // get index of the service
    // to be removed
    const updatedServices = [...services]; // create a copy of the services array
    updatedServices.splice(serviceIndex, 1); // remove the service at the given
    // index from the copy of the services array
    const updatedPrices = [...Prices]; // create a copy of the prices array
    updatedPrices.splice(serviceIndex, 1); // remove the price at the given index
    // from the copy of the prices array
    setServices(updatedServices); // update the state of the services array with
    // the updated copy
    setPrices(updatedPrices); // update the state of the prices array with the
    // updated copy
  };

  const handleSubmit = () => {
    updateTypes(services, Prices);
  };

  useEffect(() => {
    setServices(types);
    setPrices(prices);
  }, [types]);

  return (
    <Container>
      <Title>Please Add or Remove a Service</Title>
      {services.map((service, index) => (

```

```

        <ServiceContainer key={service}>
          <ServiceText>{service}</ServiceText>
          <PriceInputContainer>
            <PriceInputLabel>Price:</PriceInputLabel>
            <PriceInputField
              value={String(Prices[index])}
              onChangeText={(price) => {
                const updatedPrices = [...Prices];
                updatedPrices[index] = Number(price);
                setPrices(updatedPrices);
              }}
              keyboardType="numeric"
            />
          </PriceInputContainer>
          <Button title="Remove" onPress={() => handleRemoveService(service)} />
        </ServiceContainer>
      )}}
    <InputContainer>
      <InputLabel>Add a new service:</InputLabel>
      <InputField
        value={newService}
        onChangeText={setNewService}
        placeholder="Enter service name"
      />
      <PriceInputContainer>
        <PriceInputLabel>Price:</PriceInputLabel>
        <PriceInputField
          value={newServicePrice}
          onChangeText={setNewServicePrice}
          placeholder="Enter service price"
          keyboardType="numeric"
        />
      </PriceInputContainer>
      <Button title="Add" onPress={handleAddService} />
    </InputContainer>
    <ButtonContainer>
      <Button title="Submit" onPress={handleSubmit} />
    </ButtonContainer>
  </Container>
);
};

```

7.3.5 הגדרת שעות עבודה

Hours settings

11:30

Set your Working hours

Day	From	To	Not Working
Sunday	09:00	18:00	<input type="checkbox"/>
Monday	09:00	18:00	<input type="checkbox"/>
Tuesday	09:00	18:00	<input type="checkbox"/>
Wednesday	09:00	18:00	<input type="checkbox"/>
Thursday	09:00	18:00	<input type="checkbox"/>
Friday	09:00	12:00	<input type="checkbox"/>
Saturday	08:00	18:00	<input checked="" type="checkbox"/>

Submit

Dashboard

Analytics

Calendar

Customers

Settings


```

return (
  <List.Item
    title={day.name}
    titleStyle={styles.titleStyle}
    right={() => (
      <View style={styles.row}>
        <View style={styles.timeButtonContainer}>
          <Button
            title={day.startTime}
            onPress={() =>
              showDatePicker(index, "isDatePickerVisibleStart")
            }
            disabled={day.disabled}
            color={day.disabled ? "gray" : "purple"}
          />
          <DateTimePickerModal
            isVisible={day.isDatePickerVisibleStart}
            mode="time"
            onConfirm={(date) => handleConfirmStart(date, index)}
            onCancel={() =>
              hideDatePicker(index, "isDatePickerVisibleStart")
            }
          />
        </View>

        <View style={styles.timeButtonContainerEnd}>
          <Button
            title={day.endTime}
            onPress={() => showDatePicker(index, "isDatePickerVisibleEnd")}
            disabled={day.disabled}
            color={day.disabled ? "gray" : "purple"}
          />
          <DateTimePickerModal
            isVisible={day.isDatePickerVisibleEnd}
            mode="time"
            onConfirm={(date) => handleConfirmEnd(date, index)}
            onCancel={() => hideDatePicker(index, "isDatePickerVisibleEnd")}
          />
        </View>
        <View style={styles.checkboxContainer}>
          <Checkbox
            status={day.disabled ? "checked" : "unchecked"}
            onPress={() => toggleDay(index)}
            color="purple"
          />
        </View>
      </View>
    )
  />
)

```

```
        </View>
      </View>
    })
  />
);
};
```

7.3.6 דף הגדרת יעדים

11:31

← Goals

Aim High! Set up your goals for each Treatment

Treatment Name:	Number of meetings:
Hair	<input type="text" value="20"/>
nails	<input type="text" value="5"/>
Nails	<input type="text" value="6"/>
laser	<input type="text" value="1"/>

UPDATE GOALS

Dashboard

Analytics

Calendar

Customers

Settings

```

export const Goals = () => {
  const [goals, setGoals] = useState(initialGoals);
  const [treatments, setTreatments] = useState([
    "hair",
    "nails",
    "laser",
    "zona",
  ]); // Replace with your initial goals array

  const { types, treatmentGoals, setTreatmentGoals, updateUserGoals } =
    useContext(PartnerSchedulerContext);

  const updateGoals = () => {
    updateUserGoals(goals);
  };
  const isNumeric = (value) => {
    return !isNaN(parseFloat(value)) && isFinite(value);
  };

  useEffect(() => {
    // initialize array to make sure flow wont break in case theres no Goals.
    if (treatmentGoals.length === 0) {
      const newArray = Array.from(types, () => 0);
      setGoals(newArray);
    } else if (treatmentGoals.length < types.length) {
      const zerosArray = Array(types.length - treatmentGoals.length).fill(0);
      const newGoals = treatmentGoals.concat(zerosArray);
      setGoals(newGoals);
    } else {
      setGoals(treatmentGoals);
    }
    setTreatments(types);
  }, []);

  const onChangeGoal = (text, index) => {
    if (text === "") {
      const newGoals = [...goals]; //placeholder
      newGoals[index] = 0; //setDefaultValue
      setGoals(newGoals);
    } else if (!isNumeric(text)) {
      Success("Please insert Number!!!");
      return;
    } else {
      const newGoals = [...goals];
      newGoals[index] = parseInt(text);
    }
  };

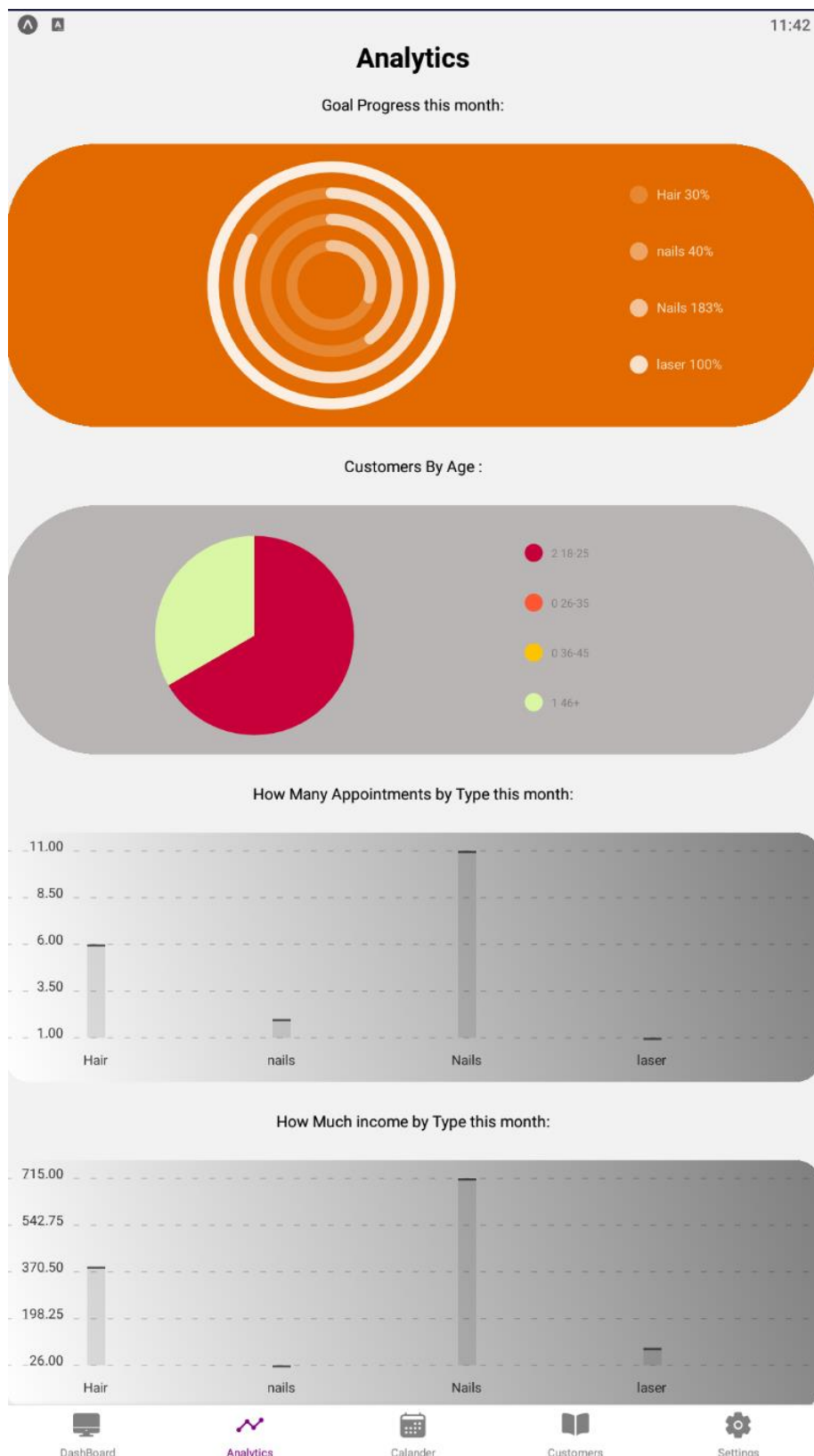
```

```

        setGoals(newGoals);
    }
};

return (
    <Container>
        <Title>Aim High! Set up your goals for each Treatment</Title>
        <SubTitles>
            <LeftItem>
                <Text>Treatment Name:</Text>
            </LeftItem>
            <RightItem>
                <Text>Number of meetings:</Text>
            </RightItem>
        </SubTitles>
        {treatments.map((treatment, index) => (
            <TreatmentContainer key={index}>
                <TreatmentName>{treatment}</TreatmentName>
                <GoalInput
                    keyboardType="numeric"
                    value={goals[index].toString()}
                    onChangeText={(text) => onChangeGoal(text, index)}
                />
            </TreatmentContainer>
        ))}
        <UpdateButton title="Update Goals" onPress={updateGoals} />
    </Container>
);
};

```



```

import React, { useContext, useEffect } from "react";
import {
  View,
  Text,
  StyleSheet,
  Dimensions,
  ScrollView,
  ActivityIndicator,
} from "react-native";
import {
  ProgressChart,
  PieChart,
  BarChart,
  LineChart,
} from "react-native-chart-kit";
import { SafeArea } from "../../components/utility/safe-area.component";
import styled from "styled-components/native";
import {
  Container,
  Title,
  ScrollViewContent,
  ChartContainer,
  ChartBarContainer,
  ChartTitle,
  SubText,
} from "../../styles/analytics.style";
import { PartnerSchedulerContext } from
"../../services/schedulaer/partner.scheduler.context";
import { AnalyticsContext } from
"../../services/analytics/analytics.context.provider";
import { CustomersContext } from "../../services/customers/customers.context";
import AsyncStorage from "@react-native-async-storage/async-storage";

export const AnalyticsScreen = () => {
  const {
    fetchData,
    Sort,
    progressData,
    ageData,
    appointmentData,
    incomeByTypeData,
    isLoading,
    setIsLoading,
  } = useContext(AnalyticsContext);

```

```

const { customers, getCustomers } = useContext(CustomersContext);

const BarchartConfig = {
  backgroundGradientFrom: "#000000",
  backgroundGradientFromOpacity: 0,
  backgroundGradientTo: "#000000",
  backgroundGradientToOpacity: 0.5,
  color: (opacity = 1) => `rgba(0, 0, 0, ${opacity})`,
  strokeWidth: 2, // optional, default 3
  barPercentage: 0.5,
  useShadowColorFromDataset: false, // optional
};

const chartConfig = {
  backgroundColor: "#e26a00",
  backgroundGradientFrom: "#e26a00",
  backgroundGradientTo: "#e26a00",
  decimalPlaces: 2, // optional, defaults to 2dp
  color: (opacity = 1) => `rgba(255, 255, 255, ${opacity})`,
  labelColor: (opacity = 1) => `rgba(255, 255, 255, ${opacity})`,
  barRadius: 50,
};

const incomeData = {
  labels: ["5 Mar", "10 Mar", "15 Mar", "20 Mar", "25 Mar"],
  datasets: [
    {
      data: [800, 1200, 1000, 1600, 2000],
      color: (opacity = 1) => `rgba(255, 0, 0, ${opacity})`,
      strokeWidth: 2,
    },
  ],
};

return (
  <SafeArea>
    <Container>
      <Title>Analytics</Title>
      <ScrollViewContent>
        <SubText>Goal Progress this month:</SubText>
        <ChartContainer>
          <ProgressChart
            data={progressData}
            width={Dimensions.get("window").width}
            height={250}
            strokeWidth={10}
            radius={35}
            chartConfig={chartConfig}

```



```

        hideLegend={false}
        center={[0, 50]}
      />
    </ChartContainer>
    <SubText>Customers By Age :</SubText>
    <ChartContainer>
      <PieChart
        data={ageData}
        width={Dimensions.get("window").width}
        height={220}
        chartConfig={chartConfig}
        accessor={"population"}
        backgroundColor={"#b9b5b5"}
        paddingLeft={"15"}
        center={[25, 5]}
        absolute
      />
    </ChartContainer>
    <SubText>How Many Appointments by Type this month:</SubText>
    <ChartBarContainer>
      <BarChart
        data={appointmentData}
        width={Dimensions.get("window").width}
        height={220}
        yAxisLabel=""
        chartConfig={BarchartConfig}
        verticalLabelRotation={0}
      />
    </ChartBarContainer>
    <SubText>How Much income by Type this month:</SubText>
    <ChartBarContainer>
      <BarChart
        data={incomeByTypeData}
        width={Dimensions.get("window").width}
        height={220}
        yAxisLabel=""
        chartConfig={BarchartConfig}
        verticalLabelRotation={0}
      />
    </ChartBarContainer>
  </ScrollViewContent>
</Container>
</SafeArea>
);
};

```