

הקדמה

לפני כשנה, כשנדרשנו לבחור נושא לפרויקט הגמר, התלבטנו בין מספר

נושאים כאשר כל נושא היה מאתגר בתחומו שלו .

לבסוף החלטנו לבחור בתחום חיפוש מורים פרטיים מאחר ולאחר מחקר ראינו כי תחום זה מאוד פעיל, וגם האפליקציה נותנת במה גם למי שלא מורה במקצועו .בנוסף, לדעתנו תחום זה הינו בעל פוטנציאל רב והתחומים שבהם אפשר לקבל שיעורים הם .

במהלך העבודה על הפרויקט, חקרנו רבות על המצב בשוק והרצון להשתמש באפליקציה כזאת ונחשפנו לאתגרים הרבים שיש ביצירת אפליקציה שכזאת אל מכשיר הסמארטפון.

במקביל, נחשפנו לפתרונות הקיימים היום והיות והם בסביבת הWEB זה לא נותן מענה מספק מד המשתמשים שברשותם סמארטפון

בספר זה מובא התהליך אותו עברנו, החל משלב הרעיון, שלב המחקר, שלב התכנון ועד לשלב המימוש בפועל.

ספר זה מציין את סופו של התהליך אותו עברנו במסגרת לימודינו, אך אינה מסמלת את סיומה של אפליקציית Teacher4U, אותה נמשיך לפתח ולקדם גם בעתיד .

תוכן עניינים

[1. מבוא 4](#_Toc37496337)

[2. הצעת פרויקט 5](#_Toc37496338)

[3. אפיון 21](#_Toc37496339)

[4. מסמך דרישות SRS 29](#_Toc37496340)

[Activity Diagrams – ותרשימי פעילות Use Cases – תרחישי שימוש .5 37](#_Toc37496341)

[מסמכי SAD 67](#_Toc37496342)

[6. ארכיטקטורה פיזית 67](#_Toc37496343)

[7. ארכיטקטורה לוגית 70](#_Toc37496344)

[8. תרשימי רצף ברמת רכיבים– Sequence Diagrams 72](#_Toc37496345)

[מסמכי SDD 83](#_Toc37496346)

[9. מודל ישויות במרחב הבעיה) PDOM(- 84](#_Toc37496347)

[10. תרשים מחלקות – UML 85](#_Toc37496348)

[11. תרשימי רצף ברמת עצמים. 87](#_Toc37496349)

[.)State Chart (מכונת מצבים .12 92](#_Toc37496350)

[13. בדיקות Test cases– מקרי בדיקה. 94](#_Toc37496351)

[14. פירוט מסכי האפליקציה ודוגמאות קוד 106](#_Toc37496352)

[15. מסד נתונים – Firebase 167](#_Toc37496353)

[16. ביבליוגרפיה 174](#_Toc37496354)

### 3. אפיון

##### סיפור לקוח

Teacher4u היא אפליקצייה המיועדת למכשירי טלפון חכמים המריצים את מערכת ההפעלה android.

האפליקציה משמשת רשת חברתית המפגישה בין תלמידים למורים פרטיים, כך שכל אדם יכול להעניק מהידע שלו (בין אם הוא מורה מקצועי ובין אם אדם מן השורה בעל ידע בתחום מסויים) ולקבל על כך תמורה.

כאשר משתמש נכנס לאפליקציה עליו להירשם תחילה. בשלב ההרשמה לאפליקציה יבחר המשתמש האם החשבון אותו הוא פותח יהיה חשבון תלמיד או חשבון מורה. כל בחירה תוביל להליך הרשמה מתאים.

הרשמה כתלמיד:

כאשר תלמיד נרשם לאפליקציה ייווצר פרופיל חדש עבור הנרשם ובו כל האפשרויות השונות שיהיו לו בתור תלמיד. האפשרויות יהיו:

1. מסך ייעודי המציג את פרטיו האישיים ובו האפשרות לשנות אותם.
2. מסך ייעודי המציג את ההודעות אותם קיבל התלמיד, הן מהמורים, והן מהמערכת.
3. מסך ייעודי עבור מערכת השעות של התלמיד, בו הוא יוכל לצפות בשיעורים שקבע, לשנות או/ו לבטל שיעור/ים – מה שישלח מיד הודעה למורה על כל שינוי.
4. מסך ייעודי עבור חיפוש מורים המציג את כל הסננים בהם יכול להשתמש התלמיד עבור חיפוש ממוקד יותר של מורה פרטי. הסננים יהיו:
   * המקצוע המבוקש.
   * האזור בו מלמד המורה.

לאחר הזנת הנתונים, יגיע התלמיד למסך התוצאות. כאן יופיעו כל המורים העונים על הדרישות אותם הזין התלמיד באופן מתומצת, כלומר שם המורה, המקצוע, גיל, מין והציון שלו (ראה בהמשך).

ע"י בחירה במורה יופיע הפרופיל המלא של המורה ובו כל המידע עליו. כאן יוכל התלמיד לשלוח הודעה למורה לקביעת שיעור הכולל תאריך ושעה ומלל חופשי – הודעה זו תשלח למורה שיצטרך לאשר אותה ובכך יקבע השיעור.

שעה לאחר מועד השיעור ישלח לתלמיד הודעה למסך ההודעות ובו בקשה לנתינת ציון למורה וכתיבת ביקורת קצרה על השיעור.

הרשמה כמורה:

לאחר הרשמה כמורה ייווצר עבורו פרופיל חדש ובו כל האפשרויות השונות שיהיו לו בתור מורה. האפשרויות יהיו:

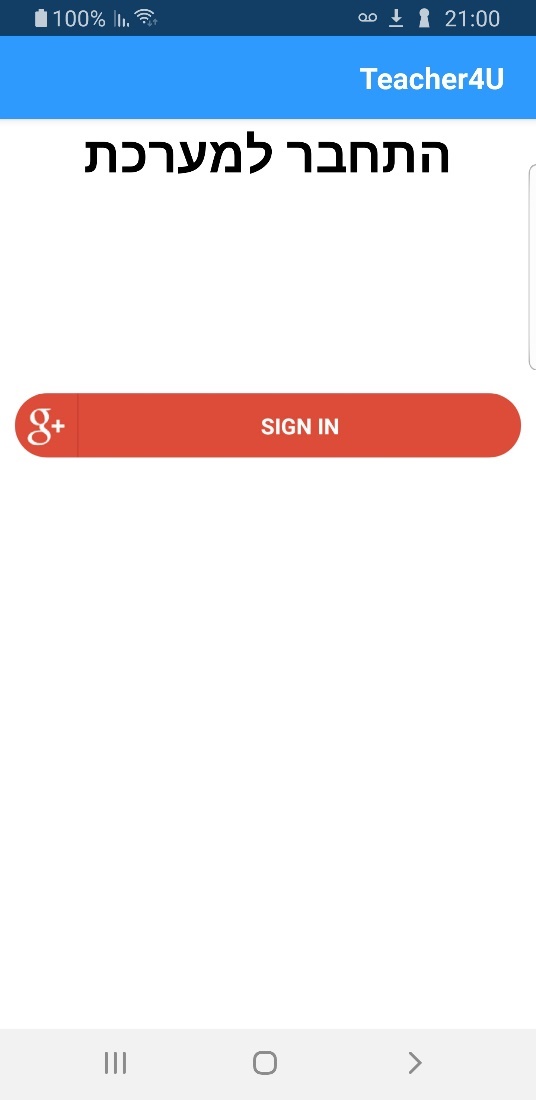
1. מסך ייעודי המציג את פרטיו האישיים ובו האפשרות לשנות אותם.
2. מסך ייעודי המציג את ההודעות אותם קיבל, הן מהתלמידים והן מהמערכת.
3. מסך ייעודי עבור מערכת השעות של המורה, בו הוא יוכל לצפות בכל השיעורים שאושרו על ידו, לשנות או/ו לבטל שיעורים – מה שישלח מיד הודעה לתלמיד על כל שינוי ובקשה לאישור שלו על השינוי (אם האישור לא יתקבל מהתלמיד עד יום לפני מועד השיעור – השיעור יבוטל והודעה מתאימה תשלח לתלמיד).
4. מסך ייעודי המציג את הביקורות אותם קיבל המורה והציון שלו באותו זמן.

כל נתוני המערכת יישמרו על גבי שרת וינוהל על ידי אתר אינטרנט המיועד למנהל המערכת.

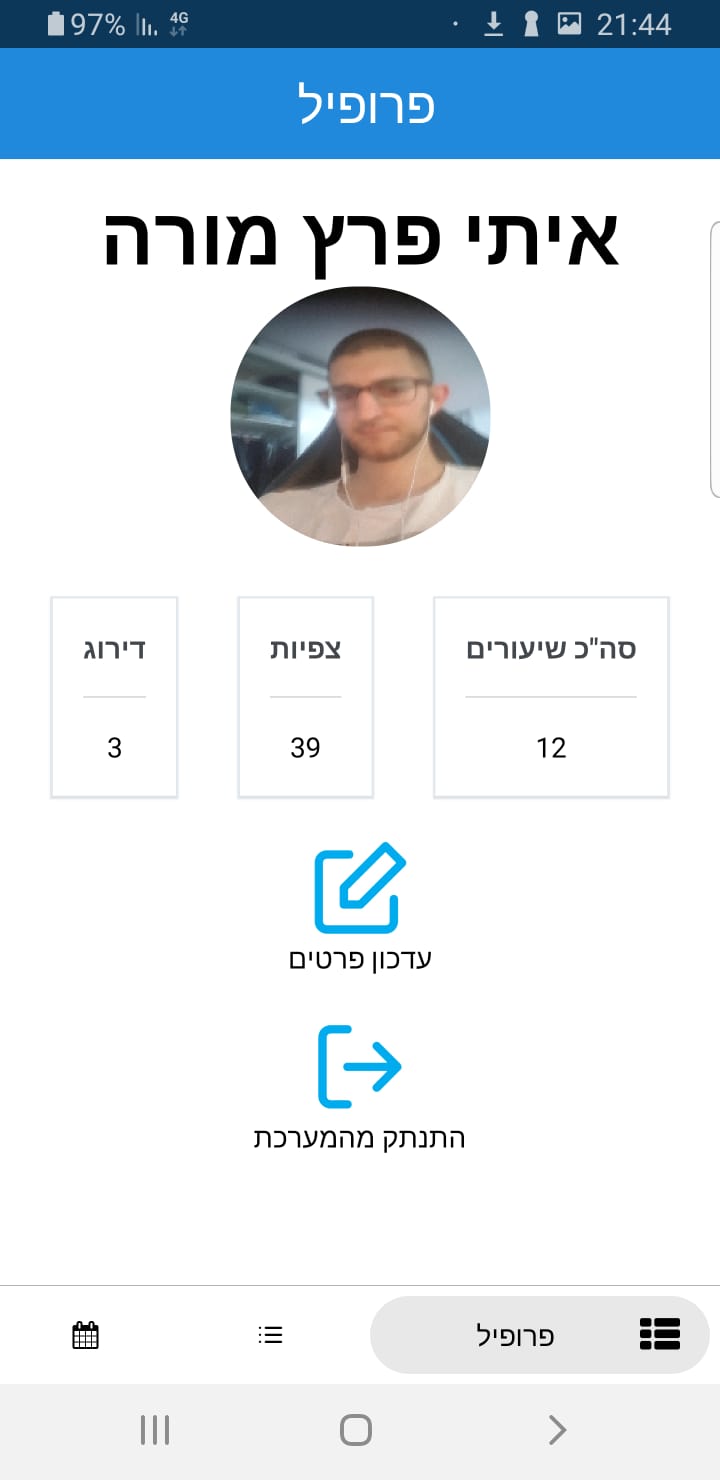
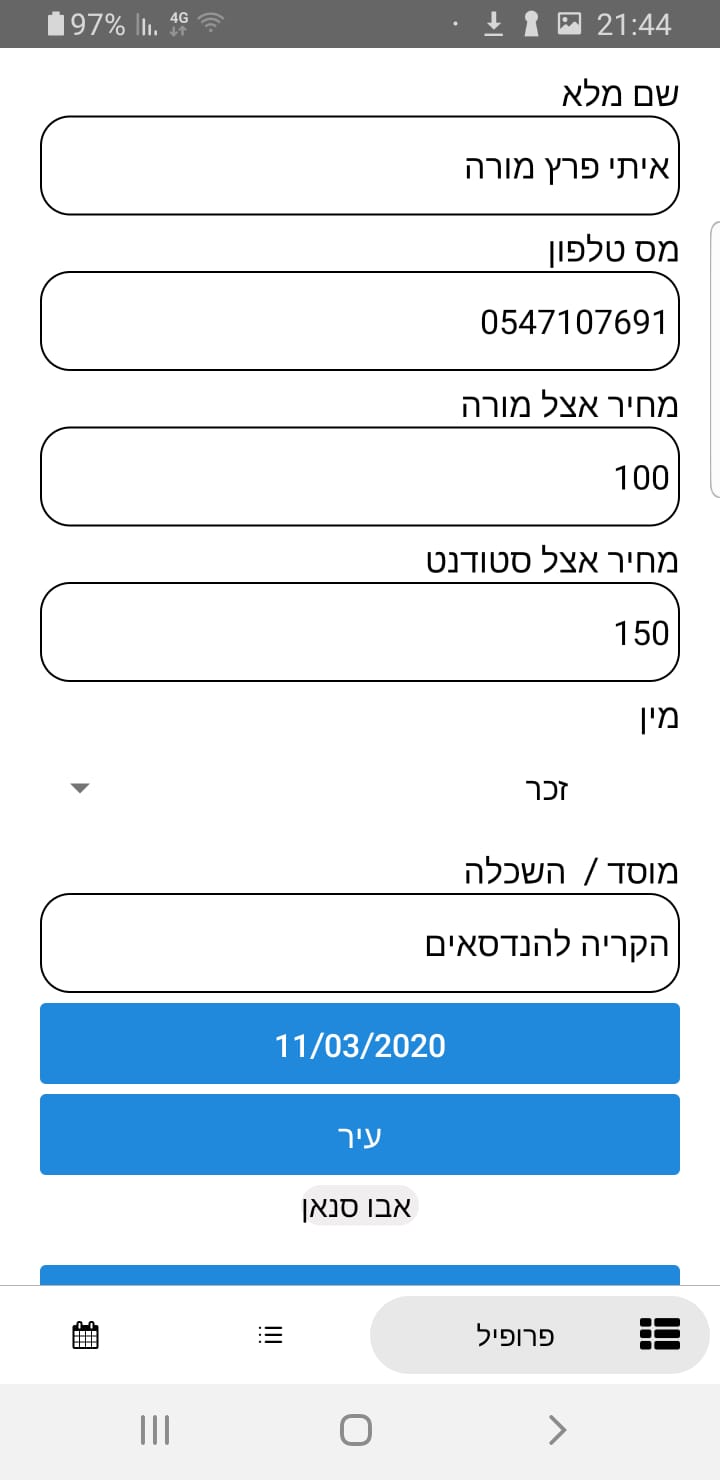
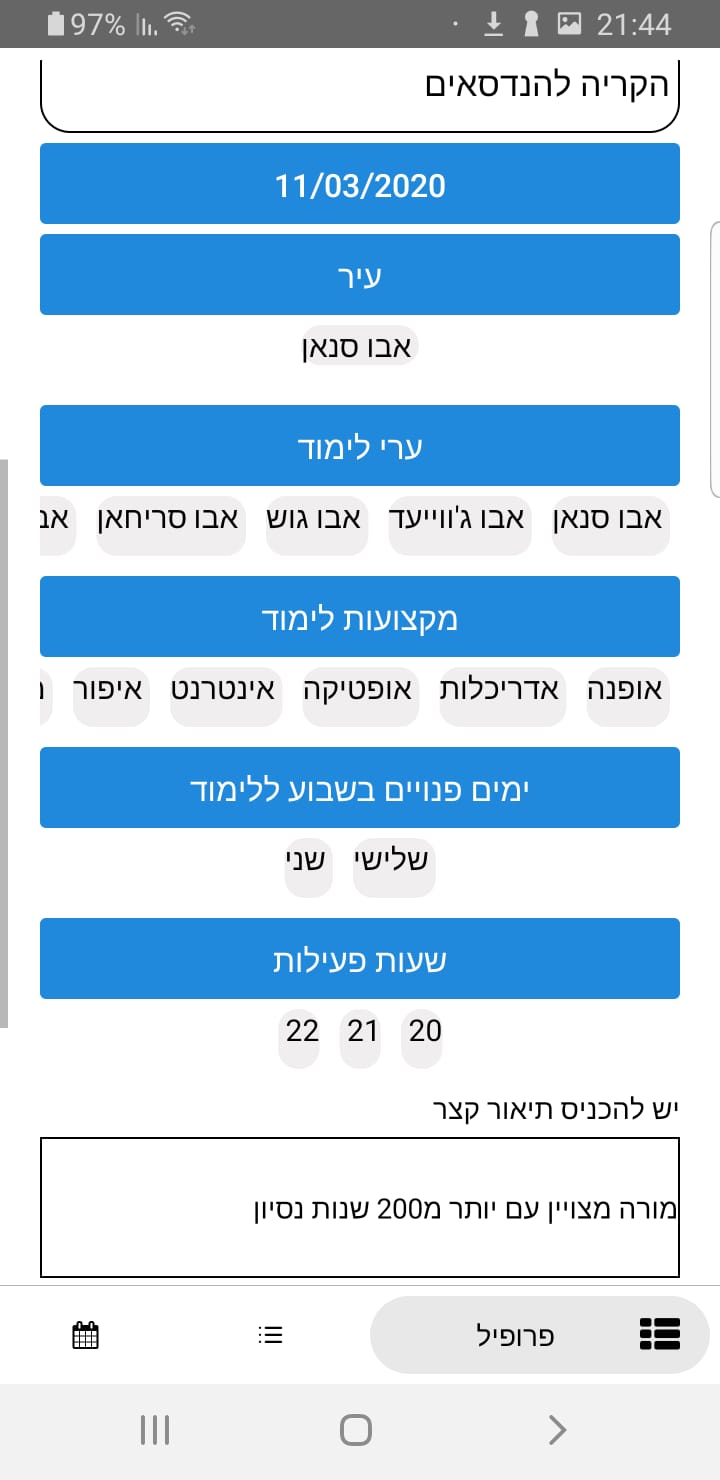
למנהל המערכת תהיה גישה לאתר על ידי הזנת שם משתמש וסיסמה. לאחר ההתחברות, יוכל המנהל לצפות בכמות המשתמשים הרשומים, ובסטטיסטיקות השימוש באפליקצייה

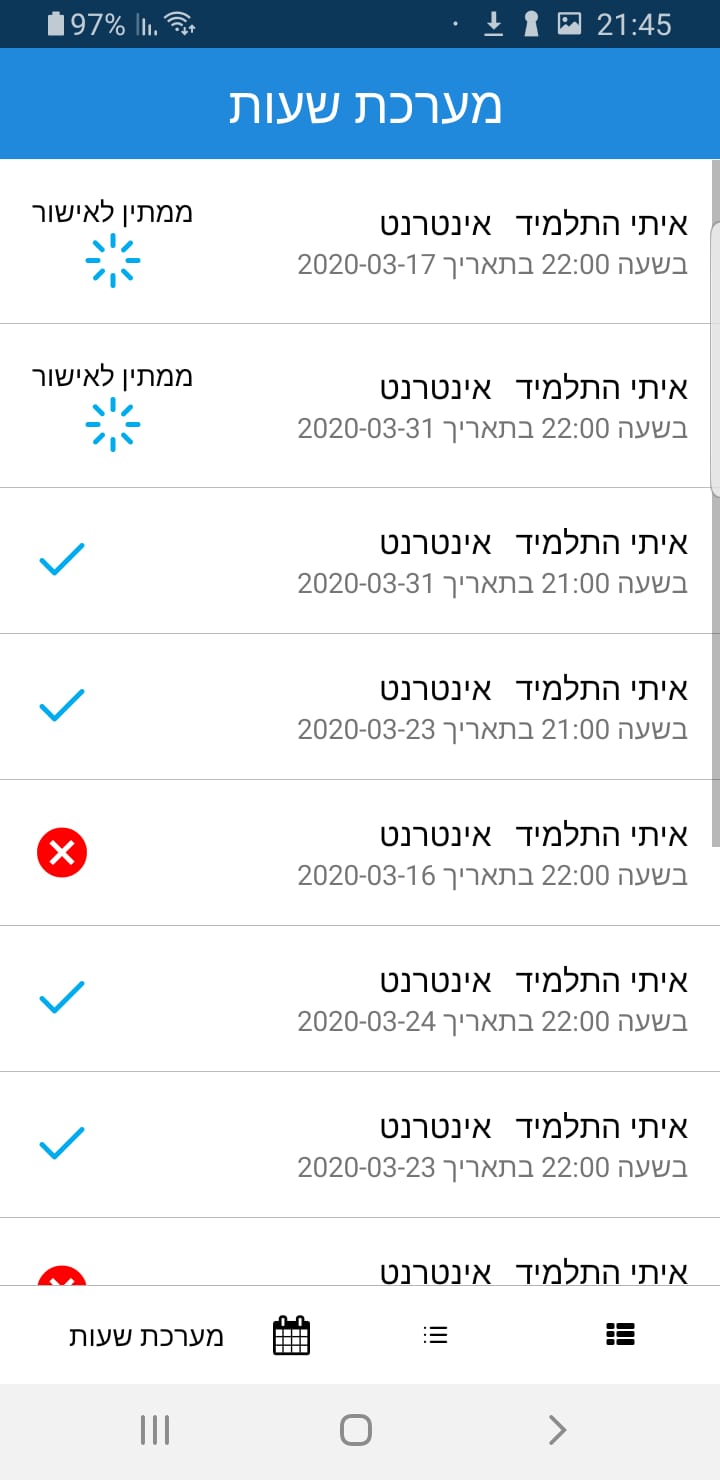
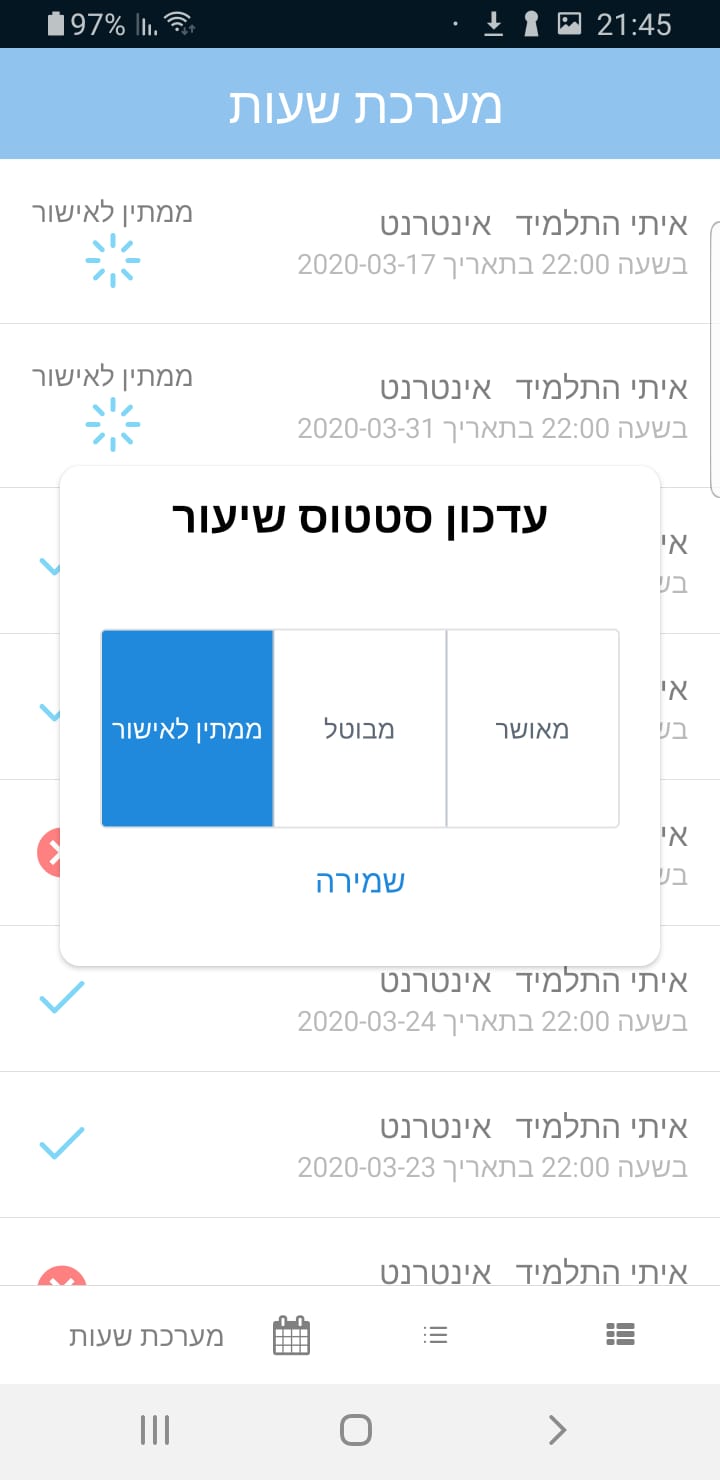
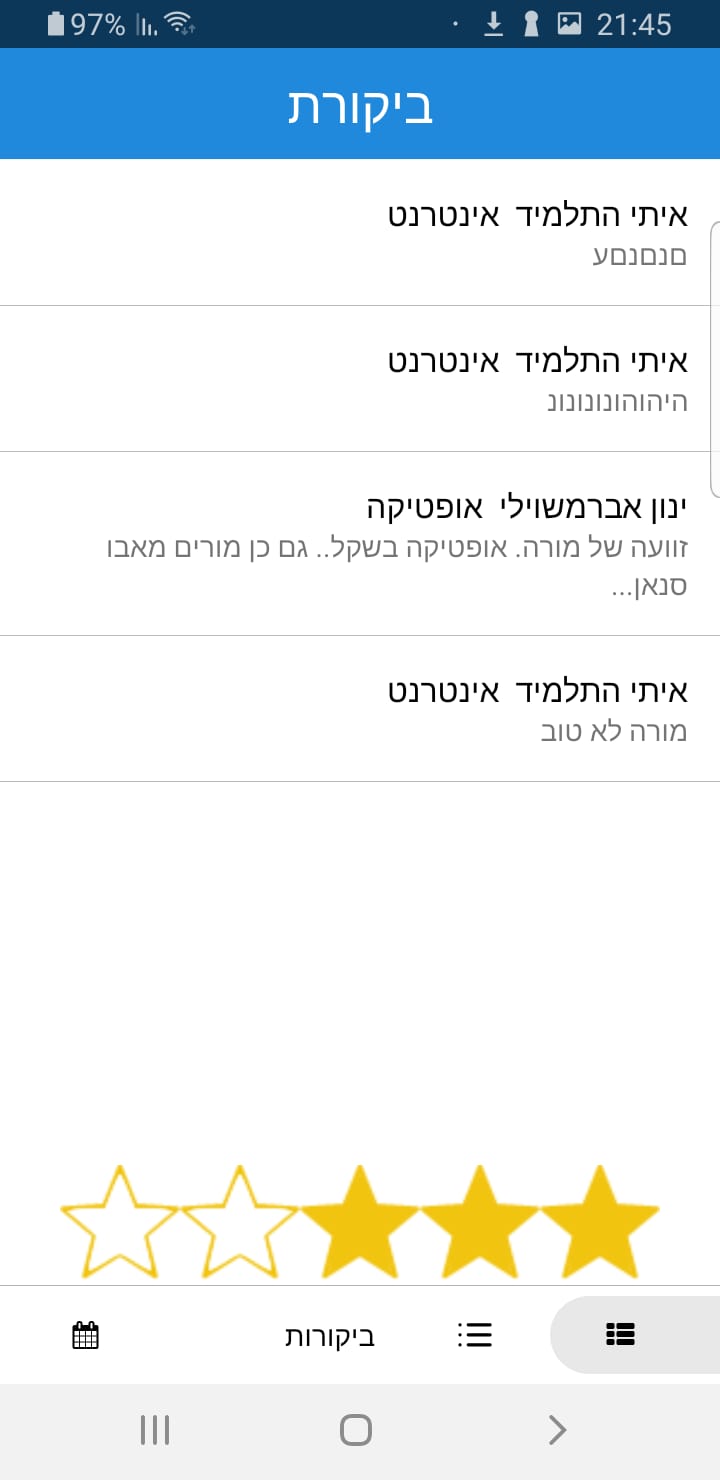
מסכים

התחברות \ רישום בחירת פרופיל

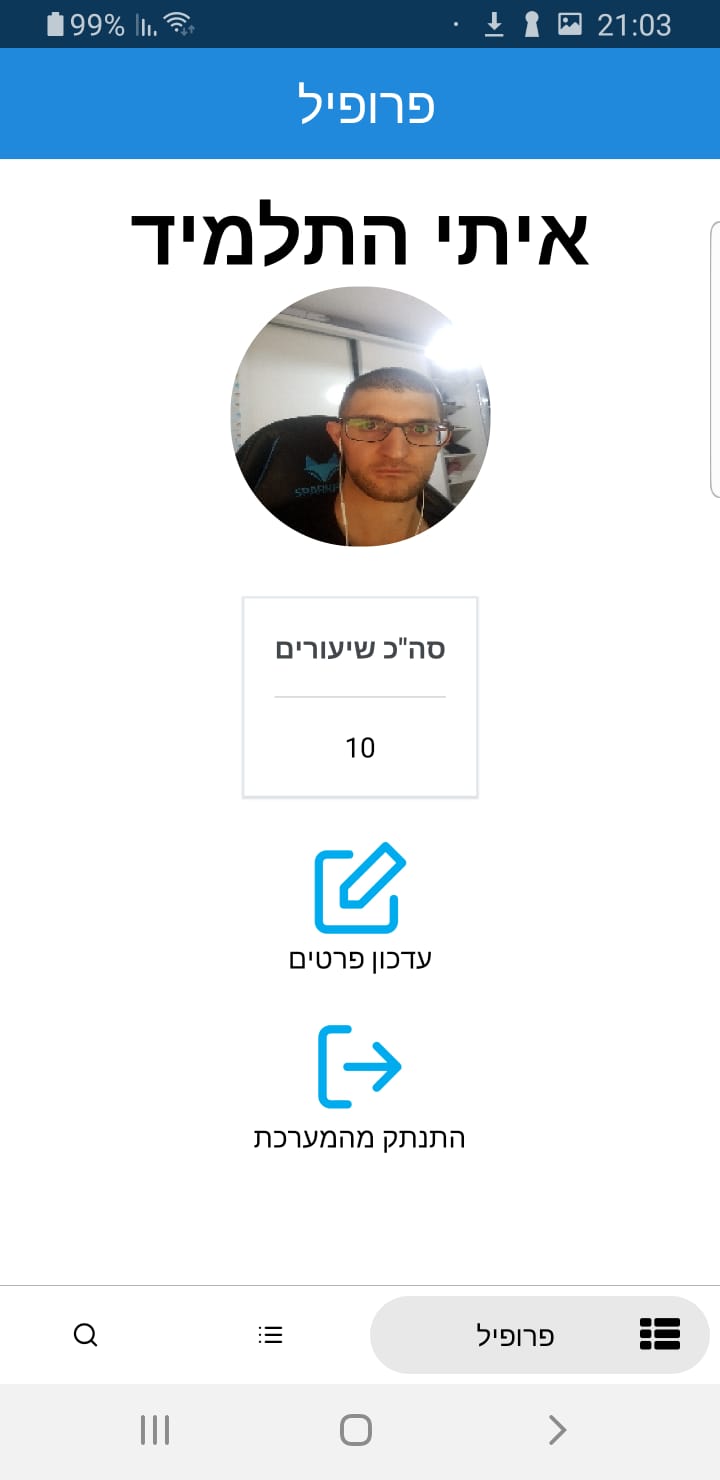
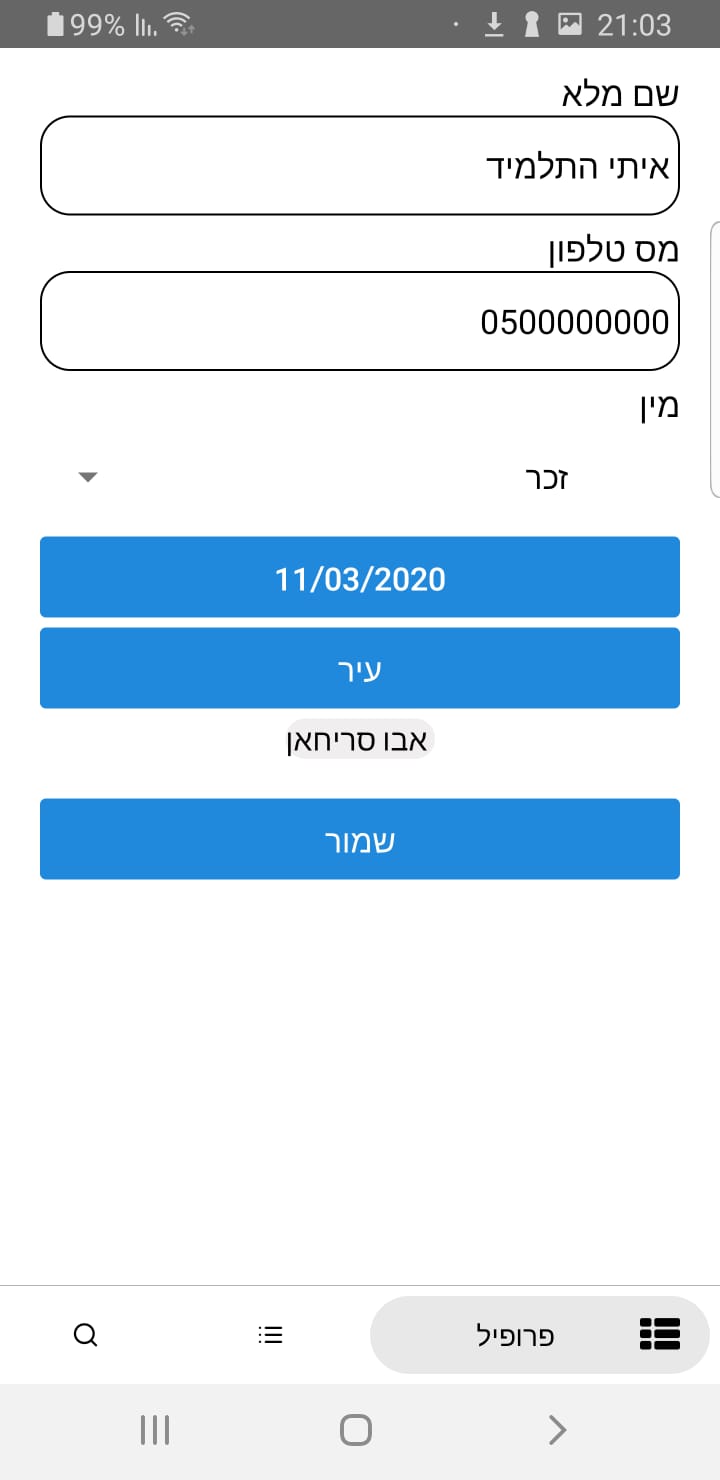
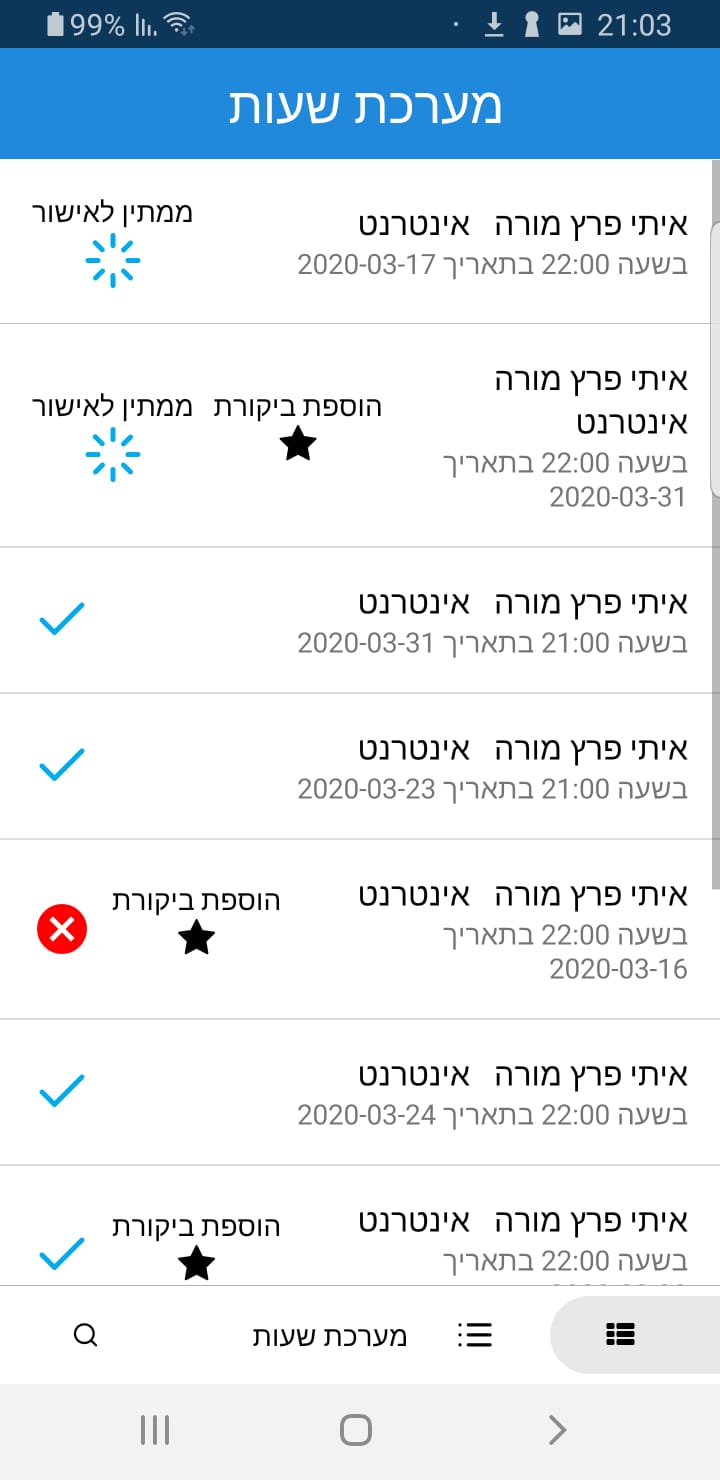
 

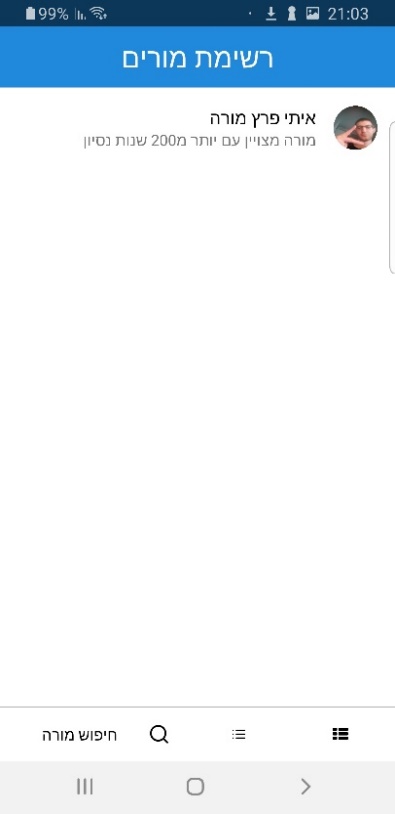
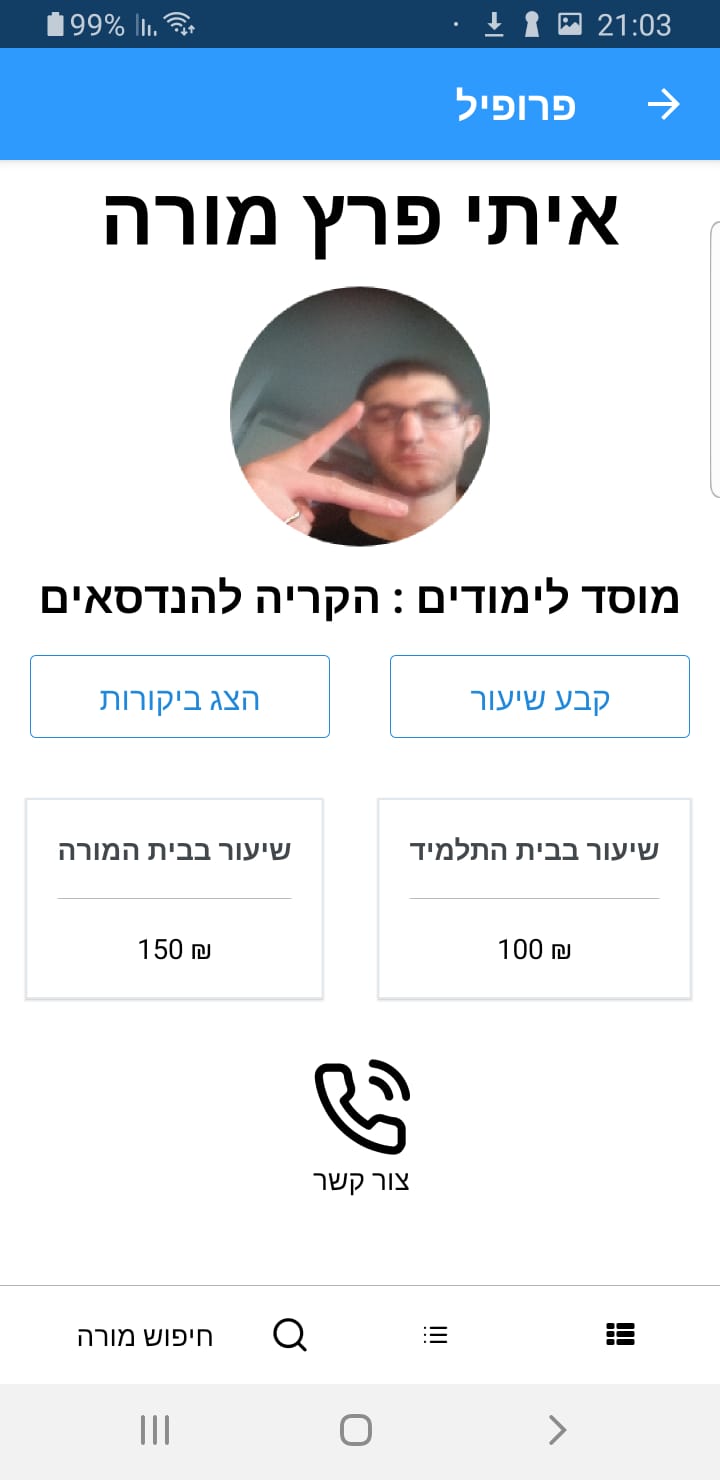
פרופיל מורה

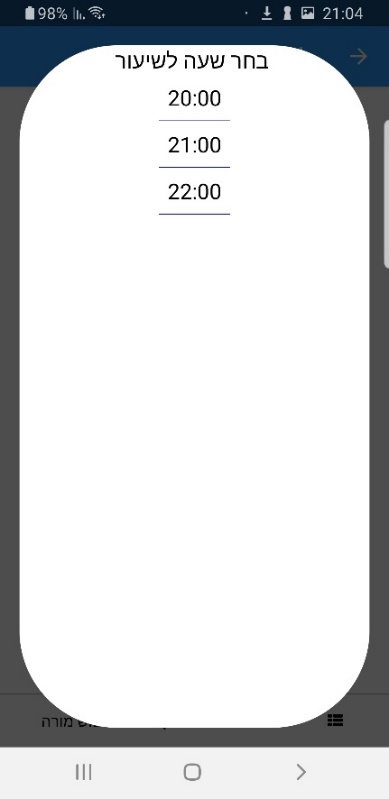
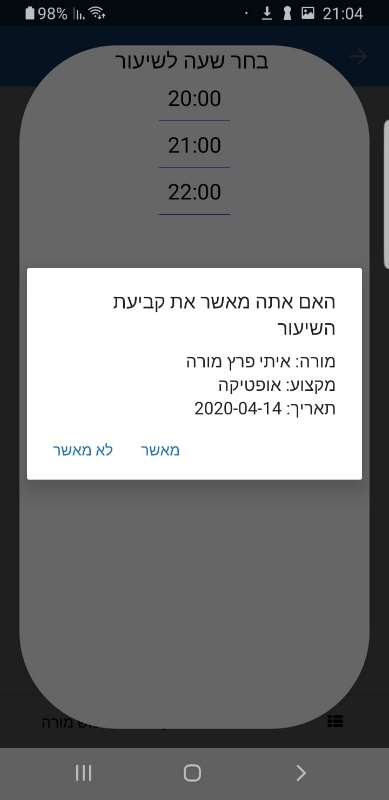
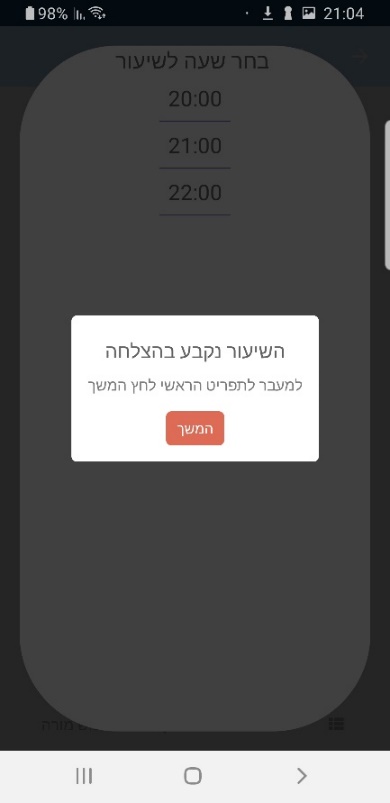
  

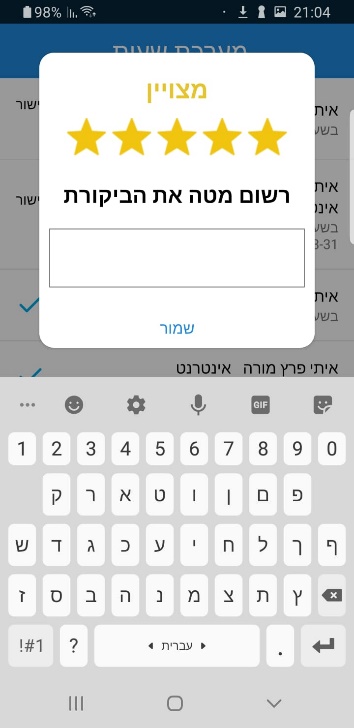
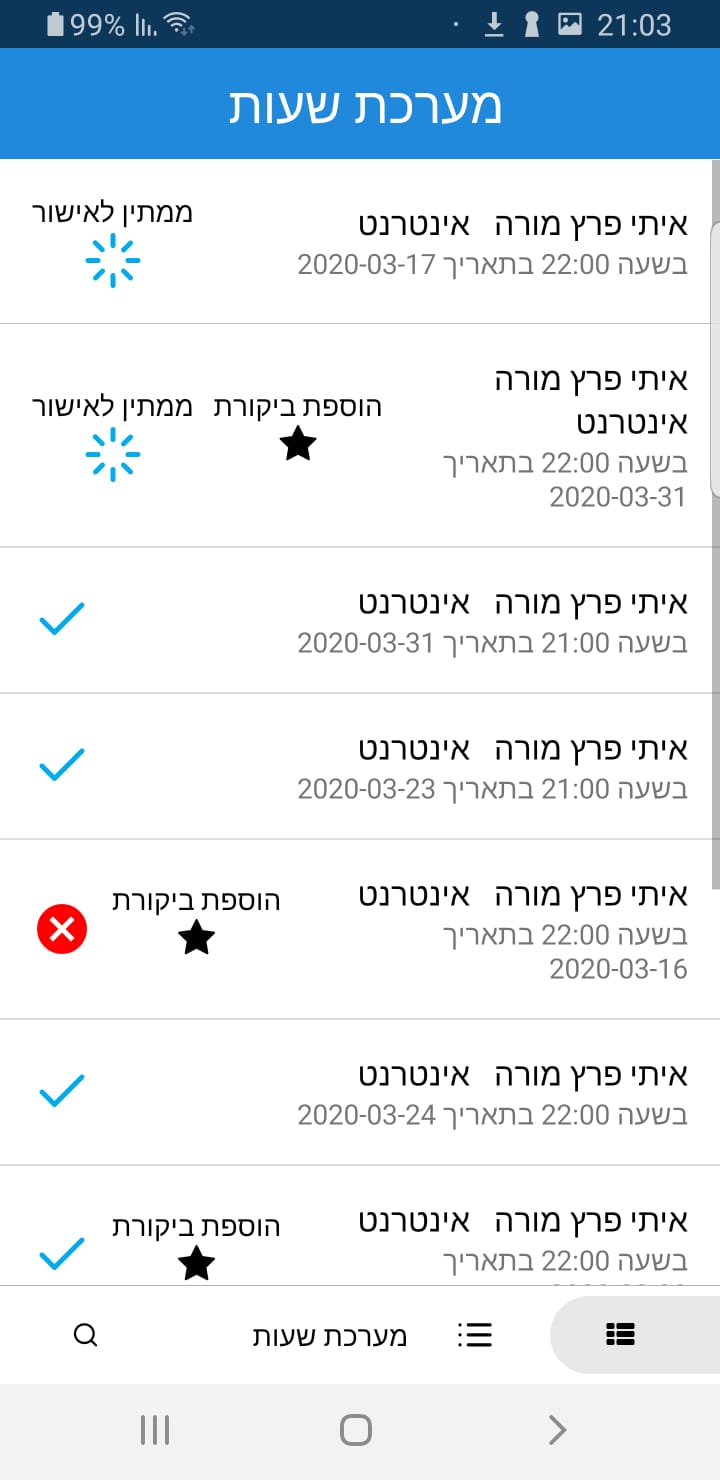
פרופיל תלמיד

המשך פרופיל תלמיד

### 4. מסמך דרישות SRS – חסר !!!!!!!!!

FG

#### Activity Diagrams ותרשימי פעילות Use Cases –

### 5. תרחישי שימוש

תרחישי שימוש – Use cases:

מטרתו העיקרית של מודל ה Use case היא לתאר את הדרישות הפונקציונאליות של האפליקציה תוך הבנה מלאה של דרישות המשתמשים, זהו תנאי הכרחי להצלחתה.

כל תרחיש מורכב מרצף של אירועים אחד או יותר, המתארים כיצד האפליקציה מתקשרת עם המשתמשים.

תרחישי השימוש משמשים לתיאור כללי של הדרישות ואת יעדיהן.

##### מבנה והסבר טבלת ה Use case

|  |  |
| --- | --- |
| שם תרחיש | מספר תרחיש |
| שמות השחקנים המשתתפים בתרחיש השימוש, שחקן הוא המשתמש באפליקציה. | שחקנים ויעדים |
| שחקנים שיש להם אינטרסים הנוגעים לפעולות האפליקציה. | ב"ע ואינטרסים |
| תנאי קדם שצריכים להתקיים טרם ביצוע תרחיש השימוש. | Pre-Conditions |
| תנאים שחייבים להתקיים לאחר השלמת התרחיש, "סיום מוצלח." | Post-Conditions |
| מה גורם להפעלת תרחיש השימוש. | Trigger |
| זהו תרחיש הצלחה ראשי – Main Successful Scenario החלק העיקרי של תיאור תרחיש השימוש, מטרתו לתאר את רצף הפעולות כדרך הקצרה והנכונה ביותר להשגת היעד, תוך הבטחת האינטרסים של בעלי העניין. | MSS |
| מתאר קצף שונה משצוין בMSS, רצף אלטרנטיבי שיביא לסיום מוצלח של תרחיש השימוש. | הסתעפות חלופה |
| מתאר קצף שונה משצוין בMSS, רצף זה יביא לסיום **לא** מוצלח של התרחיש. | הסתעפות חריגה |
| דרישות המוכללות בתרחיש השימוש לפי מסמך ה SRS למעלה. | עקיבה לדרישות (זיהוי) |

##### תרשימי פעילות - Activity diagrams

תרשים הפעילות משמש לתיאור הזרימה של התהליכים

הוא מציג את זרימת הביצוע של USECASE ומתאר אותם בצורה גרפית

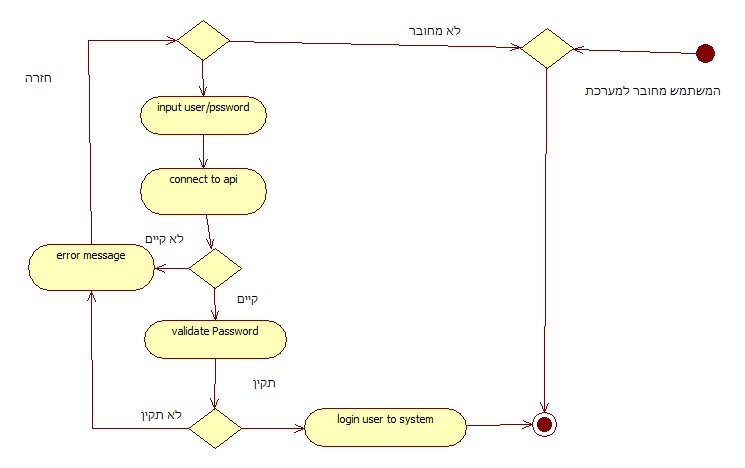
תרשים פעילות מורכב ממלבנים המייצגים כל אחד ביצוע פעילות מסוימת בתהליך. המעבר בין הפעילויות מתואר באמצעות קווים עם ראש חץ. תנאי הסתעפות או כניסה ללולאה מתואר באמצעות מעוין, שבו נרשם התנאי שיש לבדוק בכניסה להסתעפות,

או תנאי יציאה מלולאה. בשונה מתרחיש זרימה

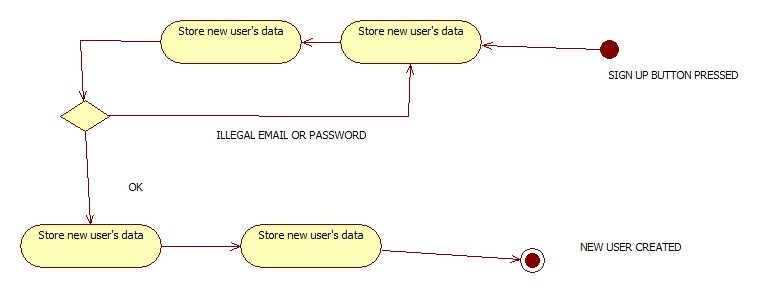
ההתחלה של תרשים פעילות מסומנת בעיגול מושחר קטן, וסיום של פעילות בעיגול מושחר כפול.

כל פעולה שמתקיימת ע"י המערכת באופן לינארי מתוארת בתוך מלבן.

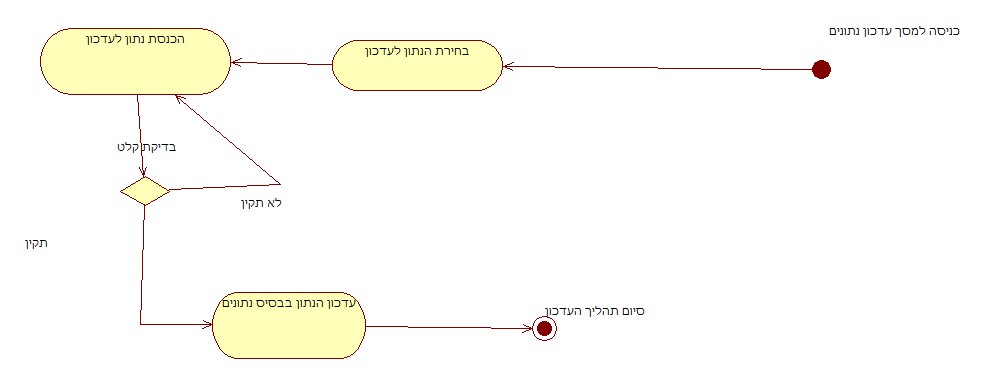
|  |  |
| --- | --- |
| LOGIN | UC-1 |
| משתמש: התחברות למערכת. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| ● קיים חיבור תקין לאינטרנט. | Pre-Conditions |
| ● קיים חשבון פעיל למשתמש. |
| ● המשתמש אינו מחובר למערכת. |
| ● המשתמש מחובר למערכת. | Post-Conditions |
| כניסה לאפליקציה | Trigger |
| 1. המשתמש מכניס את שם המשתמש וסיסמא. | MSS |
| 2. המערכת מוודאת את נתוני המשתמש מול השרת. |
| 3. פרטי המשתמש אומתו כתקינים. |
| חריגה- מצעד 3 של ה-MSS: הנתונים נתגלו כשגויים. | הסתעפות א' |
| 3א1- מוקפצת הודעה שאחד מפרטי ההזדהות אינם תקינים. |
| 3א2- נדרש למלא שוב את פרטי ההזדהות. |
| 3א3- התרחיש חוזר להתחלה. |
| 2,3,6,7 | עקיבה לדרישות (זיהוי) |



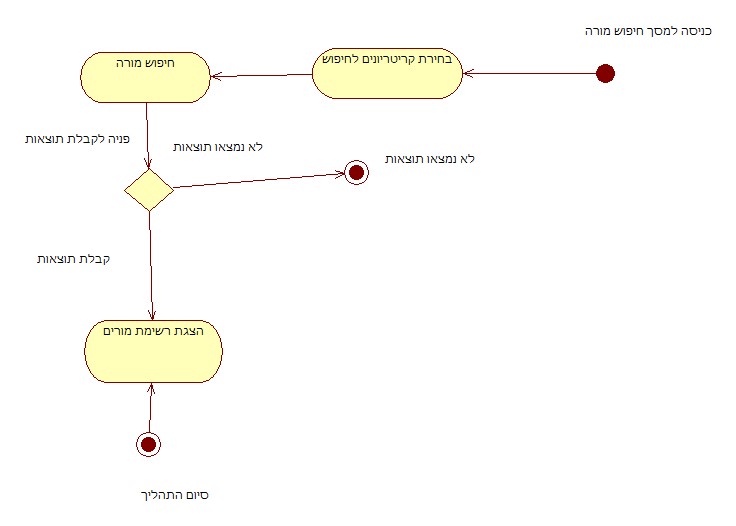
|  |  |
| --- | --- |
| SIGN UP | UC-2 |
| משתמש: התחברות ראשונית למערכת. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * קיים חיבור תקין לאינטרנט. * קיים חשבון חשבון google או facebook פעיל למשתמש. | Pre-Conditions |
| * נוצר חשבון חדש למשתמש. * המשתמש מחובר למערכת. | Post-Conditions |
| לחיצה על כפתור SIGN UP בדף ההתחברות . | Trigger |
| 1. המשתמש מכניס את כתובת הדוא"ל את הסיסמה של החשבון הרלוונטי 2. המשתמש בוחר האם הוא מורה או תלמיד 3. הזנת פרטים אישיים כגון שם, מין, גיל . 4. המערכת שומרת את נתוני המשתמש בשרת ויוצרת חשבון חדש למשתמש. | MSS |
| חריגה- מצעד 1 של ה-MSS:לא קיים חשבון FACEBOOK או GOOGLE.  המשתמש נדרש להירשם תחילה לGOOGLE או FACEBOOK. | הסתעפות א' |
| 4-8 | עקיבה לדרישות (זיהוי) |



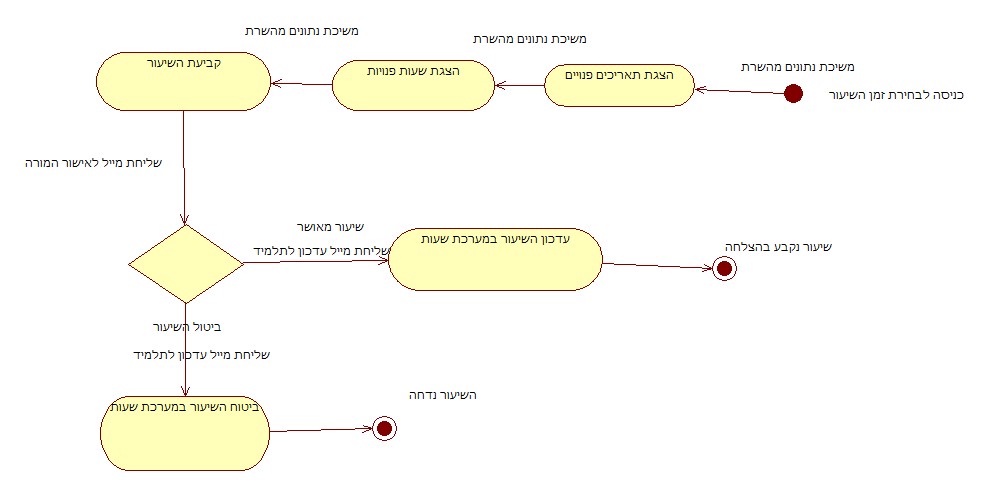
|  |  |
| --- | --- |
| עדכון נתונים | UC-3 |
| משתמש: עדכון נתונים. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * התחברות למערכת * חיבור תקין לאינטרנט | Pre-Conditions |
| ●הנותנים יתעדכנו | Post-Conditions |
| לחיצה על כפתור עדכן נתונים. | Trigger |
| 1. המשתמש בוחר את הנתון שהוא רוצה לעדכן 2. המשתמש מעדכן את הנתון . 3. המשתמש לוחץ על "לעדכון לחץ כאן" | MSS |
| חריגה- מצעד 1 של ה-MSS: הוכנס נתון לא תקין(כגון שם לטלפון)  1א1- מוקפצת הודעה למשתמש, יש להשתמש רק בספרות | הסתעפות א ' |
| 10,36-39,28-31, | עקיבה לדרישות (זיהוי) |



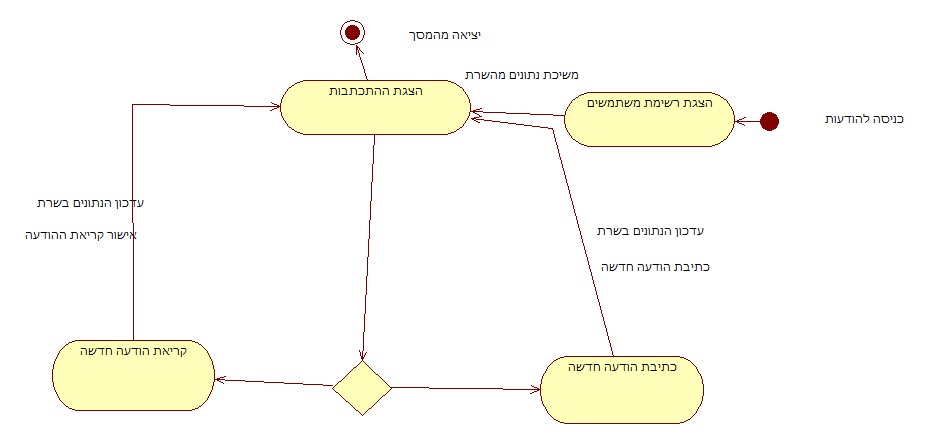
|  |  |
| --- | --- |
| חיפוש מורה | UC-4 |
| משתמש: מציאת מורה פרטי | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * המשתמש מחובר למערכת כתלמיד. * חיבור תקין לאינטרנט | Pre-Conditions |
| ● מציאת מורה . | Post-Conditions |
| 1. בחירת הקריטריונים לחיפוש כגון עיר , מקצוע. 2. לחיצה על הכפתור "חיפוש מורה". | Trigger |
| 1. המערכת פונה לשרת לקבלת הנתונים 2. המערכת שולפת מהשרת את הנתונים המתאימים. 3. המערכת תציג רשימת מורים המתאימים לחיפוש | MSS |
| חריגה- מצעד 1 של ה-MSS: לא נמצאו מורים המתאימים לקריטריונים.  1א1. יש לחפש לפי קריטריונים אחרים. | הסתעפות א' |
| 29 ,30,49-53 | עקיבה לדרישות (זיהוי) |



|  |  |
| --- | --- |
| קביעת שיעור | UC-5 |
| משתמש: קביעת שיעור פרטי עם המורה. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * קיים חיבור תקין לאינטרנט. * המשתמש מחובר כתלמיד * נמצא מורה במסך חיפוש מורה | Pre-Conditions |
| ● יקבע שיעור עם המורה .   * ישלח אל המורה מייל הממתין לאישור קביעת השיעור | Post-Conditions |
| המשתמש לוחץ על קבע שיעור במסך הפרופיל | Trigger |
| 1. המערכת תפתח לוח שנה עם התאריכים הפנויים לקביעת השיעור 2. המערכת תציג את השעות שזמינות באותו תאריך שנבחר 3. המערכת תקבע שיעור 4. המערכת תשלח מייל למורה לאישורו הסופי לקביעת השיעור   לאחר אישור המורה השיעור יעודכן במערכת שעות | MSS |
| חריגה- מצעד 23של ה-MSS:המורה דוחה את השיעור.  1א1. המערכת תשלח מייל לתלמיד עם עדכון על ביטול השיעור | הסתעפויות |
| 56,57,10,17 | עקיבה לדרישות (זיהוי) |



|  |  |
| --- | --- |
| שליחת הודעות | UC-6 |
| משתמש: מסך המאפשר למשתמש לשלוח ולקבל הודעות. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * קיים חיבור תקין לאינטרנט. * המשתמש מחובר למערכת. | Pre-Conditions |
| * האפליקציה תשלח הודעות למורה\תלמיד * האפליקציה תציג את מסך ההודעות | Post-Conditions |
| 1. לחיצה על תפריט הודעות במסך הראשי 2. לחיצה על ההודעה לצורך אישור קריאתה | Trigger |
| 1. האפליקציה שולחת בקשה לשרת לקבלת ההודעות מהמורים איתם קבעתי שיעור. 2. האפליקציה מציגה את ההודעות עם הצד השני. 3. המשתמש יוכל לאשר את קריאת ההודעה 4. המשתמש יוכל לרשום הודעות חדשות. | MSS |
| אין | הסתעפויות |
| 32-34 , 40-42 | עקיבה לדרישות (זיהוי) |



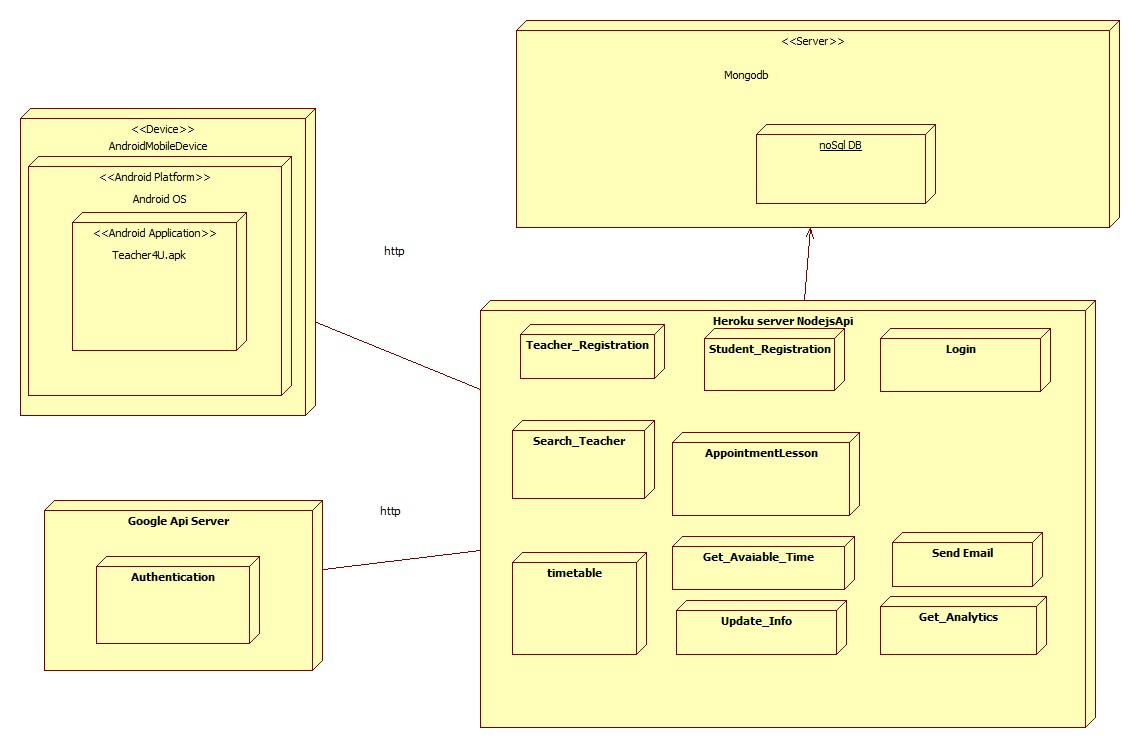
# מסמכי SAD

### 6. ארכיטקטורה פיזית

בחלק זה מובאים 4 תרשימים המתארים את המבנה הפיזי של מערכת SIN.

1. תרשים פריסה Deployment Diagram ) )

תיאור הרכיבים הפיזיים במערכת כ Nodes



1. פריטי חומרה( Nodes )

תיאור רכיבים פיזיים המשתמשים את המערכת לביצוע משימתה.

|  |  |  |
| --- | --- | --- |
| **שם** | **מהות** | **עקיבות לדרישות** |
| Android Mobile Device | מכשיר המריץ מערכת אנדרואיד להפעלת המערכת |  |
| Application Server (Heroku) | שרת המנהל את מסד הנתונים של המערכת. מבוסס NoSql ,שליטה על שרת זה יהיו דרך האפליקציה ע"י שימוש בפונקציות מובנות |  |
| Mongodb atlas | שרת המכיל את מסד הנתונים של המערכת. |  |
| Computer with browser | מכשיר המריץ את הדפדפן דרכו יתחברו לממשק אדמין |  |

1. פרטי תוכנה (Software Artifacts )

תיאור רכיבי התוכנה המשתמשים את המערכת לביצוע פעולותיה.

|  |  |  |
| --- | --- | --- |
| שם | מהות | עקיבות לדרישות |
| Teacher4U Application | לב המערכת, מותקן על המכשיר הנייד, האפליקציה מנהלת את כל דרישות המערכת ומתקשרת עם כל הגורמים החיצוניים במערכת |  |
| Heroku NodeJS Api | צד השרת של המערכת אחראי על קבלת בקשות דרך הAPI |  |
| MongoDb atlas DB | בסיס נתונים מבוסס NoSql |  |
| Web Browser | הדפדפן דרכו ההתחברות לממשק אדמין תתבצע |  |

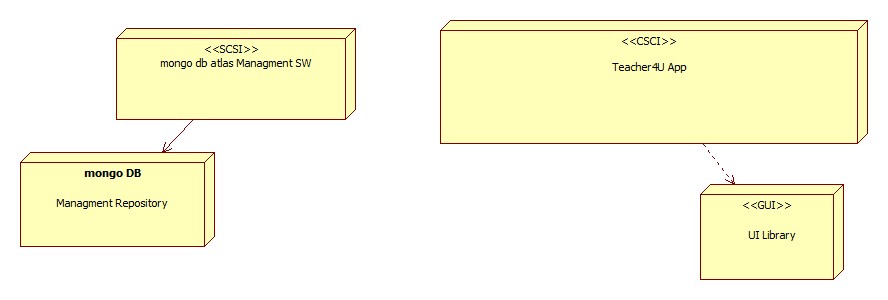
1. ממשקים פיזיים-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HWCI | שם | תכנים | תווך\פרוטוקול | עקיבות לדרישות |
| User Mobile | Teacher4U Mobile App | >התחברות  >ביצוע פעולות באפליקציה כגון  חיפוש מורים , קביעת שיעור, שליחת הודעות | אינטרנט / HTTP |  |
| Application Server | Real-Time DB Server | >קבלה ועדכון נתונים לפי שאילתות | אינטרנט / HTTP |  |
| Computer | Management console in browser | הוצאת דוחות סטטיסטים על המערכת | אינטרנט / HTTP |  |

### 7. ארכיטקטורה לוגית

בחלק זה מובאים 2 תרשימים המתארים את המבנה הלוגי של מערכת SIN.

1. דיאגרמת חלוקה לרכיבי תוכנה) Software Components(



1. פירוט רכיבים וממשקים

* Smartphone Android Application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ממשקים | | | תפקיד | שם |
| פירוט | שם | סוג |
| פקודות משתמש | User Commands | מסופק | ממשק המשתמש | UI Library |
| תצוגת משתמש | User Display | נדרש |
| תצוגת תחזוקה | Main. Display | נדרש |
| הקפצת מודעות | Notifications | מסופק |

* Mongodb Management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ממשקים | |  | תפקיד | שם |
| פירוט | שם | סוג |
| שאילתות על מורים | Teacher info | מסופק | שמירה ושליפת נתונים אודות  מורים פרטיים, הודעות ,פרטי משתמש ועוד.  שאילתות על מסד הנתונים ודוחות תחזוקה/ניהולים | Mongo DB  Management  Repository |
| שאילתות כל הודעות | message info | מסופק |
| שאילתות על סטודנטים | Student info | מסופק |
| שאילתות על תאריכים פנויים | Calendars Queries | מסופק |
| דוחות כלליים | Management Reports | מסופק |

### 8. תרשימי רצף ברמת רכיבים– Sequence Diagrams

מטרתו של תרשים רצף היא להציג אינטראקציה בין רכיבים לאורך ציר הזמן בתרחיש מסוים.

תרשים זה מספק את המבט הדינמי על ריצת התרחיש – באילו רכיבים נעשה שימוש, באיזו דרך ,על ידי מי ומתי כל זה קורה על ציר הזמן.

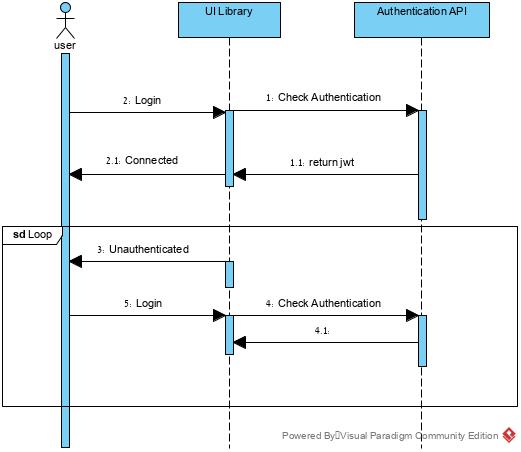
תרשים הרצף נועד לתאר תהליך ולהסביר כיצד המערכת ורכיביה מבצעים תהליך זה.

הסבר מבנה התרשים:

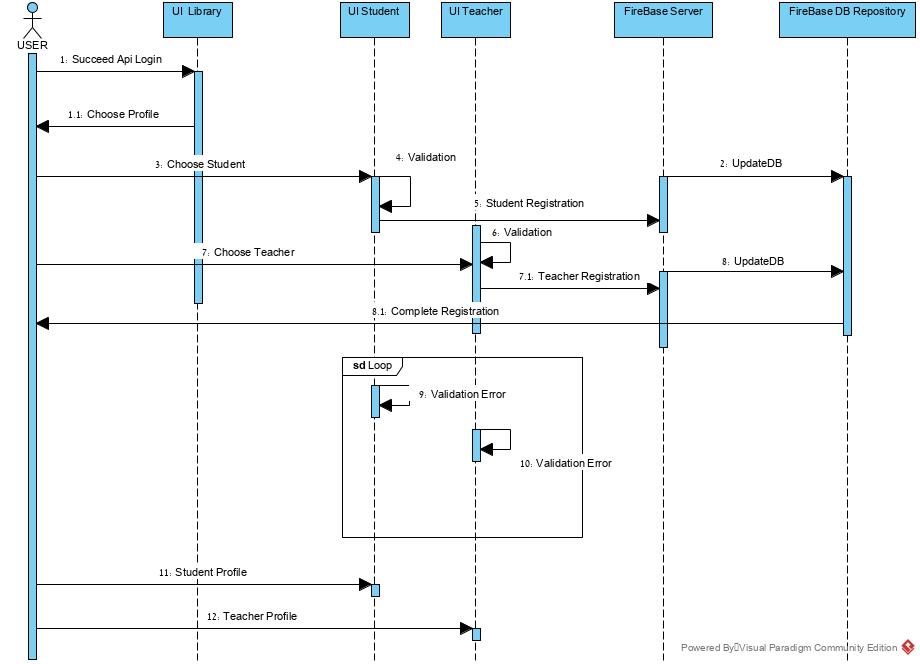
ציר הגובה המקווקו מתאר את קו החיים של האובייקט בזמן ביצוע התרחיש מלמעלה למטה. המלבנים שיופיעו על הקווים המקווקוים יתארו התממשקות של הרכיב בנקודת זמן ולפרק זמן מסויים. במשך זמן זה הרכיב במוקד הבקרה של המערכת.

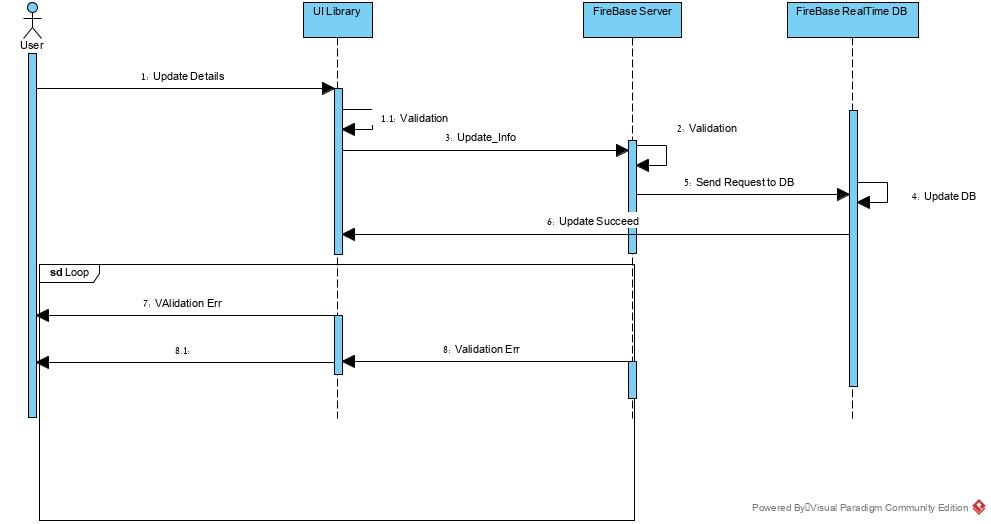
חץ עם קו מלא מתאר קריאה לרכיב או שליחת הודעה בין רכיבים, וחץ עם קו מקווקו מתאר חזרה מרכיב. בחלק מהתרשימים יופיעו גם מלבנים שיתארו אילוצי פעולה מסוימת כפי שהם מופיעים בתרחיש המימוש.

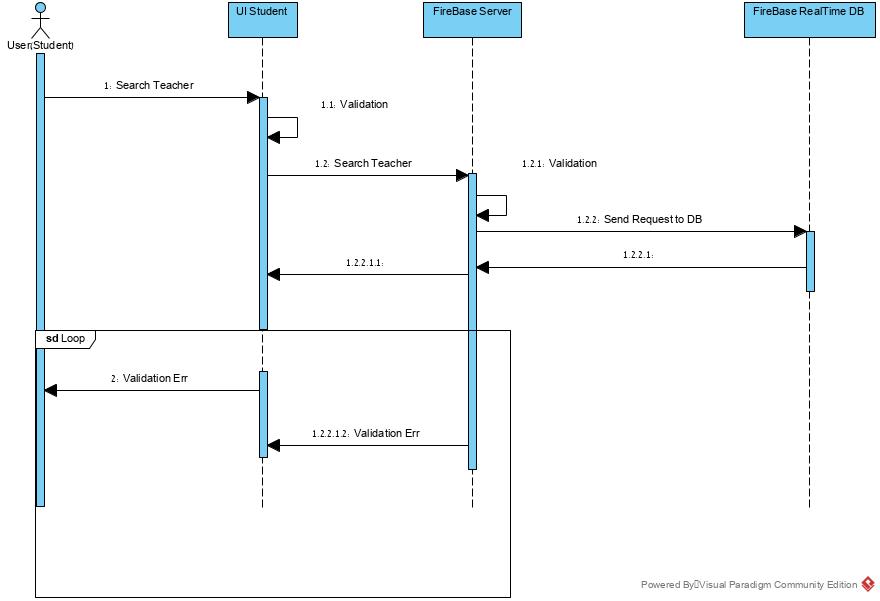
* SUC-1 LOGIN



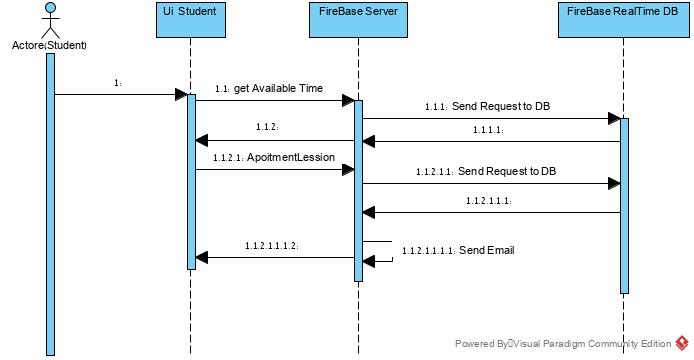
* SUC2-SIGN UP



* UC3- Update Information
* UC4-TeacherSearch



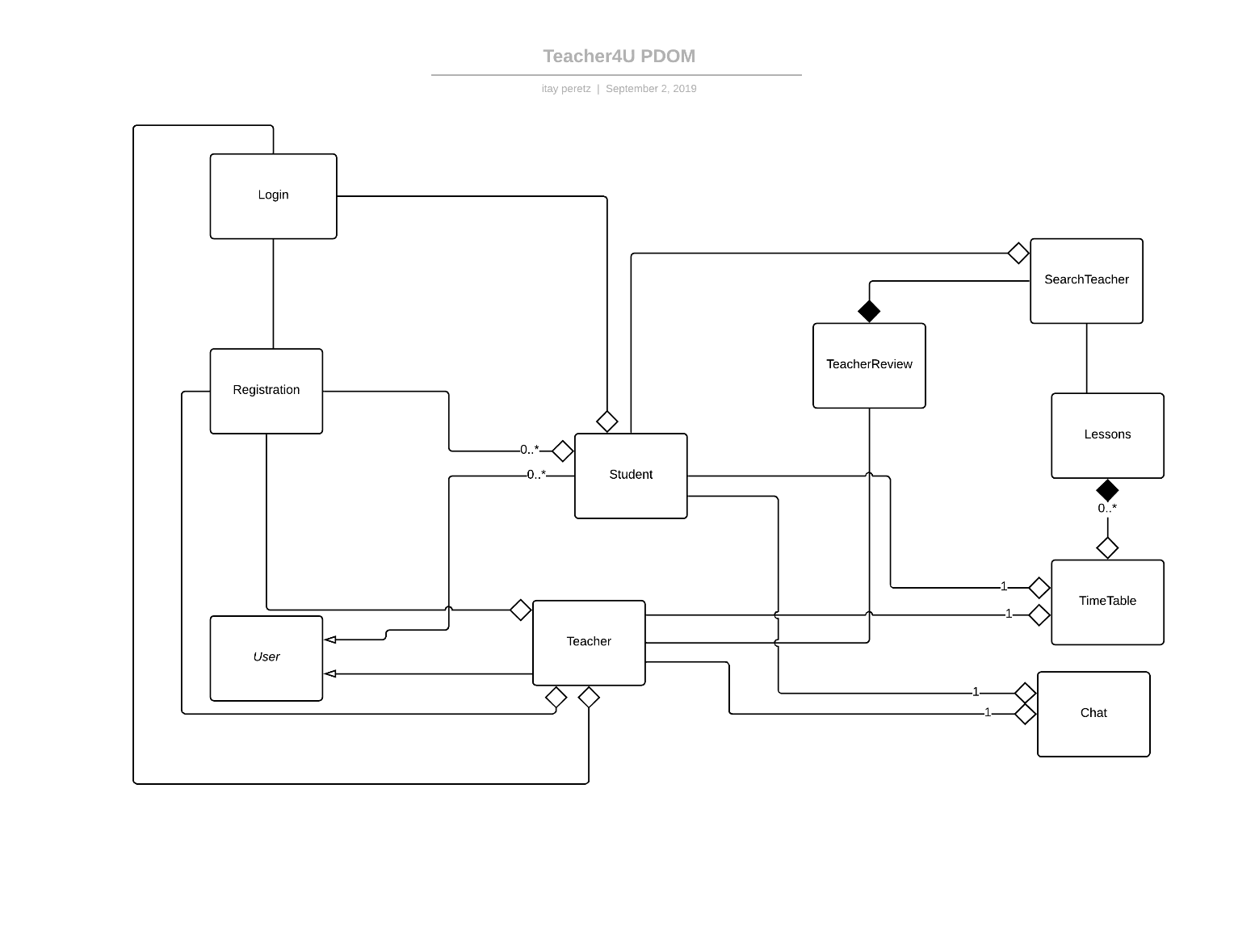
* UC5-Schedule Lesson



# מסמכי SDD

### 9. מודל ישויות במרחב הבעיה(PDOM)

להלן תרשים המתאר את תפקוד רכיבי המערכת והקשרים הלוגיים ביניהם.



## 10. תרשים מחלקות – UML

דיאגרמת ה UML מתארת את המחלקות הקיימות במערכת, שדות הנתונים בכל מחלקה וכן את הפעולות )פונקציות( שניתן להפעיל על אותה מחלקה.

היחסים בין מחלקות מתוארים באמצעות קווים המחברים בין המחלקות, סוגי הקשרים מתוארים כדלהלן-

**Association:** יחס כלשהו בין המחלקות. המספרים בקצוות מציינים מספר או טווח העצמים

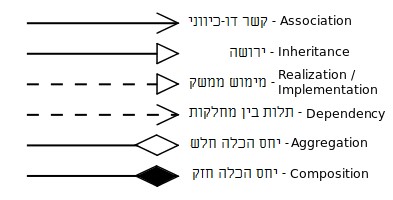
בכל צד של היחס **Inheritance:** ירושה ממחלקת בסיס.

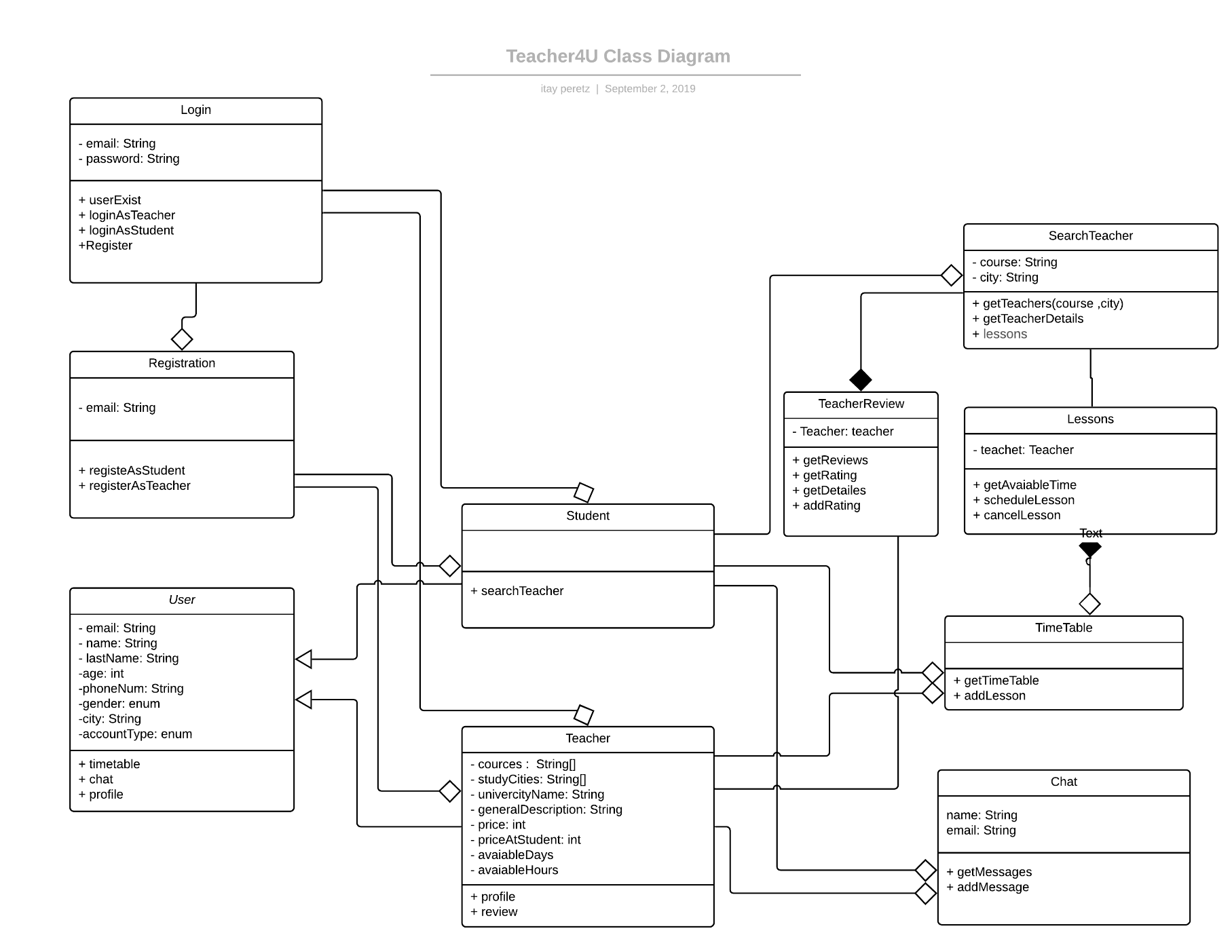
**Implementation:** מימוש ממשק. **Dependency:** תלות של מחלקה אחת במחלקה אחרת.

**Aggregation:** יחס הכלה חלש: מחלקה מכילה מצביע לעצם, כאשר העצם המוצבע יכול להתקיים ללא המחלקה המכילה.

**Composite Aggregation:** יחס הכלה חזק: מחלקה מכילה עצם, כאשר העצם המוכל מתקיים רק עם העצם החיצוני.

להלן תיאורים גרפיים של היחסים הקיימים





## 11. תרשימי רצף ברמת עצמים.

מטרתו של תרשים רצף היא להציג אינטראקציה בין עצמים הקיימים במערכת לאורך ציר הזמן בתרחיש מסוים.

תרשים זה מספק את המבט הדינמי על ריצת התרחיש – באילו עצמים נעשה שימוש ,באיזו דרך, על ידי מי ומתי כל זה קורה על ציר הזמן.

תרשים הרצף נועד לתאר תהליך ולהסביר כיצד המערכת ורכיביה מבצעים תהליך זה.

הסבר מבנה התרשים

ציר הגובה המקווקו מתאר את קו החיים של האובייקט בזמן ביצוע התרחיש מלמעלה למטה. המלבנים שיופיעו על הקווים המקווקוים יתארו פונקציות של העצם הנקראות בנקודת זמן ולפרק זמן מסוים. במשך זמן זה האובייקט במוקד הבקרה של המערכת.

חץ עם קו מלא מתאר קריאה לפונקציה או שליחת הודעה בין אובייקטים, וחץ עם קו מקווקו מתאר חזרה מפונקציה. בחלק מהתרשימים יופיעו גם מלבנים שיתארו אילוצי פעולה מסוימת כפי שהם מופיעים בתרחיש המימוש.

#### .(State Chart )מכונת מצבים .12

תרשים "מכונת מצבים" כדלהלן מציג את המצבים השונים של אובייקט ואת השפעת האירועים השונים על המצבים לאורך חייו של האובייקט.

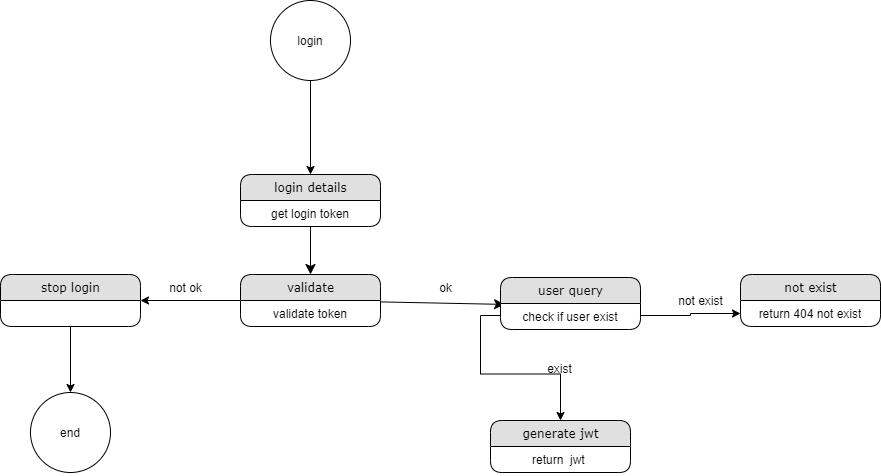
התרשים הינו נחוץ עבור המחלקות בהם ישנם מצבים שונים ורבים וכאשר נדרש לחדד את הכללים למעבר ממצב למצב )לפי הפעולות המופעלות(. מכונת המצבים פותחה על ידי דוד הראל בשנת

1987 ובשנות ה-90 פיתוח זה הפך לחלק מתקן ה-UML.



להלן תרשים המתאר את רכיבי מכונת המצבים עבור המחלקה

.Login



## 13. בדיקות Test cases– מקרי בדיקה. – אין!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

* בדיקה של כל תרחישי השימוש לעיל .
* תרחישי השימוש המתייחסים לניווט נבדקו בשטח, היינו במתחם הקריה ללימודי הנדסה וטכנולוגיה .
* תרחישי ניווט דינאמי )בין 2 משתמשים( נבדק מול 2 מכשירי נייד התומכים בגרסה המתאימה.

##### TC-1 Login

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
| OK | 20.5.19 | התחברות לאפליקציה.  מעבר למסך ראשי. | 1. המשתמש מזין שם משתמש תקין וקיים במערכת. 2. הזנת סיסמא תקינה 3. לחיצה על התחברות. | התחברות לאפליקציה ,הזנת נתונים תקינים | 1.1 |
| OK | 20.5.19 | תוקפץ הודעה למשתמש ששם המשתמש שהוזן אני תקין. | 1. הזנת שם משתמש לא חוקי. 2. הזנת סיסמא תקינה. 3. לחיצה על התחברות. | התחברות לאפליקציה ,שם משתמש לא חוקי | 1.2 |
| OK | 20.5.19 | תוקפץ הודעה למשתמש ששם המשתמש שהוזן אינו קיים במערכת. | 1. הזנת שם משתמש חוקי אך לא קיים במערכת. 2. הזנת סיסמא תקינה. 3. לחיצה על התחברות. | התחברות לאפליקציה ,שם משתמש לא קיים במערכת. | 1.3 |
| OK | 20.5.19 | תוקפץ הודעה למשתמש ,סיסמא שגויה. | 1. הזנת שם משתמש תקין. 2. הזנת סיסמא שגויה. 3. לחיצה על התחברות. | התחברות לאפליקציה ,סיסמא שגויה. | 1.4 |

##### TC-2 Sign-up

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
| OK | 20.5.19 | נוצר משתמש חדש במערכת.  המשתמש מקבל הודעת הרשמות מוצלחת.  המשתמש מועבר למסך הראשי. | 1. המשתמש לוחץ על כפתור ההרשמות. 2. המשתמש מזין שם משתמש תקין. 3. מזין סיסמא תקינה. 4. )רשות( המשתמש מזין פרטים אישיים. | התחברות ראשונית לאפליקציה, נתונים תקינים | 2.1 |
| OK | 20.5.19 | תוקפץ הודעה  למשתמש ששם  המשתמש שהוזן אני תקין. | 1. המשתמש לוחץ על כפתור ההרשמות. 2. המשתמש מזין שם משתמש לא תקין. 3. מזין סיסמא תקינה. 4. )רשות( המשתמש מזין פרטים אישיים. | התחברות ראשונית לאפליקציה, שם משתמש אינו חוקי. | 2.2 |
| OK | 20.5.19 | תוקפץ הודעה למשתמש, סיסמא לא תקינה. | 1. המשתמש לוחץ על כפתור ההרשמות. 2. המשתמש מזין שם משתמש תקין. 3. מזין סיסמא לא תקינה. 4. )רשות( המשתמש מזין פרטים אישיים. | התחברות ראשונית לאפליקציה, סיסמא אינה חוקית. | 2.3 |
| OK | 20.5.19 | תוקפץ הודעה למשתמש, המשתמש קיים במערכת. | 1. המשתמש לוחץ על כפתור ההרשמות. 2. המשתמש מזין שם משתמש תקין וקיים במערכת 3. מזין סיסמא תקינה. 4. )רשות( המשתמש מזין פרטים אישיים. | התחברות ראשונית לאפליקציה ,משתמש קיים. | 2.4 |

3-TC שכחתי סיסמא

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
| OK | 20.5.19 | נשלח למייל למשתמש עם לינק לשחזור סיסמא. | 1. המשתמש לוחץ על כפתור "שכחתי סיסמא " 2. המשתמש מזין שם משתמש  3. המשתמש לוחץ על כפתור "שחזור סיסמא " 4.המשתמש מזין את כתובת המייל הקיים במערכת | שכחתי סיסמא, שחזור התבצע בהצלחה | 3.1 |
| OK | 20.5.19 | המשתמש בחר סיסמא קבועה חדשה ותקינה | 1. המשתמש מזין את הסיסמא הזמנית שקיבל במייל 2. המשתמש בוחר סיסמא חדשה קבועה | שכחתי סיסמא, הזנת סיסמא זמנית | 3.2 |
| OK | 20.5.19 | תוקפץ הודעה למשתמש ששם המשתמש שהוזן אני תקין. | 1. המשתמש לוחץ על כפתור "שכחתי סיסמא " 2. המשתמש מזין שם משתמש שלא קיים | שכחתי סיסמא, הזנת שם משתמש שלא קיים | 3.3 |

4-TC איתור מיקום נוכחי

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
| OK | 20.5.19 | המערכת מעדכנת בשרת ובמכשיר המשתמש את  נתוני המיקום הנוכחי של המשתמש. | 1. המשתמש מתחיל   תהליך ניווט או במהלך ניווט או התחבר בהצלחה למערכת   1. המערכת מאתרת ומחשבת את מיקום המשתמש על פי Beacons שבסביבה | איתור מיקום נוכחי, אותר מקום נוכחי בהצלחה | 4.1 |
| OK | 20.5.19 | 1. המערכת מודיעה על תקלה וממשיכה לחפש התקנים 2. המערכת רושמת בשרת את פרטי התקלה | 1. המערכת סורקת Beacons בסביבת המשתמש להתחברות 2. המערכת לא מאתרת Beacons להתחברות בסביבה | איתור מיקום נוכחי, לא  אותרו Beacons להתחברות | 4.2 |
| OK | 20.5.19 | 1. המערכת מודיעה על תקלה וממשיכה לחפש התקנים 2. המערכת רושמת בשרת את פרטי התקלה | 1. המערכת סורקת Beacons בסביבת המשתמש 2. המערכת איתרה Beacon אך לא מצליחה להתחבר אליו | איתור מיקום נוכחי, לא הצליח להתחבר ל-  Beacon | 4.3 |

5-TC מידע על מבנה

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
| OK | 20.5.19 | המערכת מציגה את הפרטים על המבנה במסך ייעודי עבור המשתמש | 1. המשתמש לוחץ על כפתור האינפורמציה במסך הראשי 2. המערכת סורקת את מיקום המשתמש 3. המערכת מקבלת פרטים על המבנה שבו המשתמש נמצא כרגע | מידע על המבנה, קבלת מידע תקין על המבנה | 5.1 |
| OK | 20.5.19 | 1. המערכת מודיעה על תקלה וממשיכה לחפש התקנים | 1. המערכת סורקת את מיקום המשתמש 2. לא אותר מיקום המשתמש | מידע על המבנה, לא אותר מיקום המשתמש | 5.2 |

6-TC מידע על מיקום במבנה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **הערות** | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  | OK | 3.6.19 | 1. המשתמש בוחר מיקום המערכת מציגה את פרטי מתוך שורת החיפוש במסך המיקום במסך יעודי עבור הראשי או מתוך רשימת המשתמש.   המיקומים בדף המידע על  המבנה .   1. המערכת שולפת את הנתונים מהשרת בהסתמך על מיקום המשתמש . | | מידע על מיקום, קבלת מידע תקין על המיקום | 6.1 |

7-TC מידע על משתמש חבר

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **תוצאה** | **תאריך בדיקה** | **תהליך מקרה הבדיקה תוצאה רצויה מתאימה** | **תיאור מקרה בדיקה** | **מזהה** |
| OK | 3.6.19 | 1. לחיצה על שם המערכת מציגה את דף החבר מתוך רשימת הפרופיל של החבר החברים בתפריט הצד .במסך יעודי עבור 2. הקפצת חלונית ובה המשתמש.  דף הפרופיל של החבר .  3. הבאת הנתונים מהשרת אודות החבר. | מידע על משתמש חבר, קבלת מידע תקין. | 7.1 |

8-TC חיפוש והוספת חברים

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **תוצאה** | | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | | | **תהליך מקרה הבדיקה** | | **תיאור מקרה בדיקה** | **מזהה** | | |
| OK | | 3.6.19 | המערכת תשלח לתיבת הדואר של החבר את בקשת החברות. הודעה תוקפץ לחבר בעת התחברות לאפליקציה במידה ולא פג תוקף ההודעה. | | | 1. לחיצה על סמן חיפוש/הוספת חברים מתוך תפריט הצד . 2. המערכת פותחת חלונית עם שדה להזנת כתובת המייל של   החבר .3.  הזנת כתובת תקינה של חבר חדש. | | חיפוש והוספת חברים, כתובת אימייל תקינה וחבר חדש. | 8.1 | | |
| OK | | 3.6.19 | המערכת מקפיצה הודעה למשתמש שהחבר כבר קיים ברשימת החברים. | | | 1. לחיצה על סמן חיפוש/הוספת חברים מתוך תפריט הצד . 2. המערכת פותחת חלונית עם שדה להזנת כתובת המייל של   החבר .3.  הזנת כתובת תקינה של חבר קיים. | | חיפוש והוספת חברים, כתובת אימייל תקינה וחבר קיים. | 8.2 | | |
| OK | | 3.6.19 | | | המערכת מקפיצה הודעה למשתמש שלא נמצא המשתמש במערכת. | 1. לחיצה על סמן חיפוש/הוספת חברים מתוך תפריט הצד . 2. המערכת פותחת חלונית עם שדה להזנת כתובת המייל של   החבר .3.  הזנת כתובת תקינה של חבר לא קיים במערכת. | | חיפוש והוספת חברים, כתובת אימייל תקינה וחבר לא קיים במערכת. | | | 8.3 |
| OK | | 3.6.19 | | | המערכת מקפיצה הודעה למשתמש, האימייל אינו חוקי.. | 1. לחיצה על סמן חיפוש/הוספת חברים מתוך תפריט הצד . 2. המערכת פותחת חלונית עם שדה להזנת כתובת המייל של   החבר .3.  הזנת כתובת אימייל לא חוקית. | | חיפוש והוספת חברים, כתובת אימייל לא חוקית. | | | 8.4 |

10/9-TC הנפקת דוחות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **תוצאה** | **תאריך בדיקה** | **תהליך מקרה הבדיקה תוצאה רצויה מתאימה** | **תיאור מקרה בדיקה** | **מזהה** |
| OK | 3.6.19 | 1. פתיחת קובץ ה- הנתונים מוצגים בטבלה   HTML באופן תקין   1. קובץ הJS מייבא את הנתונים הרלוונתים מהשרת | דף HTML העושה שימוש בקוד JS מנפיק דוחות מוגדרים מראש ממסד הנתונים | 9-10.1 |

12/11-TC ניווט ליעד במבנה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **הערות** | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
| **כל ניווט במערכת הינו ניווט בין 2 נקודות, רק**  **בניווט דינאמי הנקודות**  **משתנות במהלך הניווט.** | OK | 20.5.19 | המערכת מחשבת את הכיוון הכללי ליעד והמרחק .  המערכת מתעדכנת במיקום המשתמש בכל כמה רגעים ,ומעדכנת בשרת ובתצוגת המשתמש. בהגעה ליעד מוקפצת הודעה. | 1. המערכת מקבלת בקשה לתחילת ניווט )סטטי/דינאמי( | ניווט בין 2 נקודות ,תהליך ניווט מוצלח. | 12.1 |
|  | OK | 20.5.19 | המערכת שומרת את נתוני הניווט בשרת ובמכשיר המשתמש. מוקפצת הודעה מתאימה. | 1. המערכת מקבלת בקשה לתחילת ניווט   )סטטי/דינאמי.(   1. המערכת מקבלת בקשה מהמשתמש להפסקת הניווט. | ניווט בין 2 נקודות ,תהליך ניווט מופסק ע"י משתמש. | 12.2 |
|  | OK | 20.5.19 | המערכת שומרת את נתוני הניווט בשרת ובמכשיר המשתמש. מוקפצת הודעה מתאימה. | 1. המערכת מקבלת בקשה לתחילת ניווט   )סטטי/דינאמי.(   1. המערכת מקבלת בקשה להפסקת הניווט )כיבוי אפלי', קריסת האפליקציה( | ניווט בין 2 נקודות ,תהליך ניווט מופסק ע"י מערכת | 12.3 |

13-TC ניווט דינאמי בין משתמשים

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  |  |  | עדכון דינאמי של היעד אצל 2 המשתמשים במהלך הניווט.  בעט הגעה לקרבה מוקפצת הודעה. | 1. 2 המשתמשים מאשרים תהליך של ניווט דינאמי. 2. המערכת מתעדכנת במיקום המשתמשים בכל כמה רגעים וכן מעדכנת את תצוגת המשתמשים. | ניווט דינאמי, תהליך ניווט מוצלח. | 13.1 |
|  |  |  |  | לא מתחיל ניווט, המשתמש השני מקבל הודעה על סירוב ניווט דינאמי. | 1. אחד המשתמשים לא מאשר ניווט דינאמי. | ניווט דינאמי, אחד לא מאשר ניווט. | 13.2 |
|  |  |  |  | המערכת ממשיכה לנווט דינאמית בין המשתמשים ,בהגעה לקרבה תוקפץ הודעה. | 1. 2 המשתמשים מאשרים תהליך של ניווט דינאמי. 2. המערכת מתעדכנת במיקום המשתמשים בכל כמה רגעים וכן מעדכנת את תצוגת המשתמשים. 3. אחד או שני המשתמשים סוגרים את חלון   האפליקציה ומאשר ניווט ברקע. | ניווט דינאמי, תהליך ניווט ברקע. | 13.3 |
|  |  |  |  | הניווט מופסק אצל הצד המבטל, נשמרת רשומה בשרת המתארת את הניווט .  המשתמש השני מקבל הודעה על הפסקת הניווט אצל החבר, הניווט אצלו מופסק גם ורשומה עדכנית נוספת לשרת. | 1. 2 המשתמשים מאשרים תהליך של ניווט דינאמי. 2. המערכת מתעדכנת במיקום המשתמשים בכל כמה רגעים וכן מעדכנת את תצוגת המשתמשים. 3. אחד או שני המשתמשים מפסיקים הניווט. | ניווט דינאמי, אחד  המשתמשים הפסיק לשתף. | 13.4 |

14-TC ניווט ברקע

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.2019 | המערכת ממשיכה לאסוף נתונים אודות הניווט ,מתקיים ניווט ברקע. | 1. המשתמש סוגר את חלון האפליקציה במהלך ניווט פעיל. 2. המערכת ממשיכה   לסרוק ולנווט ליעד עד לסיום של מערכת ההפעלה. | ניווט ברקע, המשתמש ממזער את חלון האפליקציה. | 14.1 |

15-TC עדכון רכיב

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.19 | הרכיב מתעדכן בשרת. | 1. הזנת שם הרכיב ו-UID חד חד ערכי. 2. הזנת נתונים על הרכיב. 3. לחיצה על כפתור הוספה. | עדכון רכיב, הרכיב עודכן במערכת בהצלחה. | 15.1 |
|  |  | OK | 20.5.19 | לא ניתן ללחוץ על כפתור עדכון. | 1. בחרית רכיב מתוך רשימה. 2. לא נבחרו שדות לעדכון. | עדכון רכיב, לא בוצעו שינויים. | 15.2 |
|  |  | OK | 20.5.19 | המשתמש מועבר לדף עדכון/הוספה/הסרה של מיקום. | 1. לאחר לחיצה על כפתור עדכון רכיב, ינתן אופציה לשייוך מיקומים קרובים. 2. לחיצה על מעבר לדף עדכון המיקומים. | עדכון רכיב )חלופה א ,('שייוך מיקומים לרכיב. | 16.3 |

16-TC הוספת רכיב

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.19 | הרכיב מתעדכן בשרת. | 1. הזנת שם הרכיב ו-UID חד חד ערכי. 2. הזנת נתונים על הרכיב. 3. לחיצה על כפתור הוספה. | הוספת רכיב, הרכיב נוסף במערכת בהצלחה. | 16.1 |
|  |  | OK | 20.5.19 | תוקפץ הודעה, הרכיב קיים במערכת. הרכיב לא יתווסף. | 1. הזנת שם הרכיב ו-UID חד חד ערכי. 2. הזנת נתונים על הרכיב. 3. לחיצה על כפתור הוספה. | הוספת רכיב, רכיב קיים במערכת. | 16.2 |
|  |  | OK | 20.5.19 | המשתמש מועבר לדף עדכון/הוספה/הסרה של מיקום. | 1. לאחר לחיצה על כפתור הוספת רכיב, יינתן אופציה לשיוך מיקומים קרובים. 2. לחיצה על מעבר לדף עדכון המיקומים. | הוספת רכיב )חלופה א ,('שיוך מיקומים לרכיב. | 16.3 |

17-TC הסרת רכיב

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.19 | הנתונים על הרכיב נמחקים מהשרת .המשתמש מקבל הודעת אישור על הסרת הרכיב מהמערכת. | 1. המשתמש בוחר מתוך רשימה את הרכיב שברצונו להסיר. 2. לחיצה על כפתור הסרת רכיב. | הסרת רכיב, הוסר בהצלחה מהמערכת | 17.1 |

18-TC עדכון נקודת עניין

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.19 | הקפצת הודעה אם השדות שעודכנו במערכת. עדכון הנתונים בשרת. | 1. בחית המיקום מתוך רשימה. 2. בחירת השדות שיש לעדכן. 3. הכנסת נתונים. 4. לחיצה על כפתור עדכון מיקום | עדכון מיקום קיים | 18.1 |
|  |  | OK | 20.5.19 | לא ניתן ללחוץ על כפתור עדכון. | 1. בחית המיקום מתוך רשימה. 2. לא נבחרו שדות לעדכון. | עדכון מיקום קיים, לא בוצעו שינויים. | 18.2 |

19-TC הסרת נקודת עניין

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.19 | הנתונים על המיקום נמחקים מהשרת. המשתמש מקבל הודעת אישור על הסרת המיקום. | 1. המשתמש בוחר מתוך רשימה את המיקום שברצונו להסיר. 2. לחיצה על כפתור הסרת מיקום. | הסרת מיקום קיים | 19.1 |

20-TC הוספת נקודת עניין

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.19 | המיקום מתעדכן בשרת. | 1. הזנת שם המיקום. 2. הזנת נתונים על המיקום. 3. לחיצה על כפתור הוספה. | הוספת מיקום, המיקום נוסף במערכת | 20.1 |
|  |  | OK | 20.5.19 | תוקפץ הודעה, המיקום קיים במערכת. המיקום לא יתווסף. | 1. הזנת שם המיקום. 2. הזנת נתונים על המיקום. 3. לחיצה על כפתור הוספה. | הוספת מיקום, מיקום קיים במערכת. | 20.2 |

21-TC יצירת קשר בין משתמשים

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **הערות** |  | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | | **תיאור מקרה בדיקה** | **מזהה** |
|  |  | OK | 20.5.19 | נוצר קישור בין המשתמשים ,מוקפצת מודעה למשתמשים אם אופציה להתחלת ניווט.  . | 1. משתמש א 'לוחץ על כפתור ההתקשרות עם החבר. 2. משתמש ב' מקבל הודעה מחבר א.' 3. משתמש ב' לוחץ על ההודעה ומועבר לדף החבר | | המשתמשים חברים ומחוברים למערכת, אחד המשתמשים שולח מסרון Poke. | 21.1 |
|  |  | OK | 20.5.19 | המסרון ממתין בשרת להתחברות מחודשת של המשתמש המקבל, באים ההודעה לא ישנה מידי ,המשתמש מקבל ההודעה בהתחברות מחודשת. | | 1. משתמש א' לוחץ על כפתור ההתקשרות עם החבר. | אחד המשתמשים אינו מחובר למערכת | 21.3 |

22-TC הודעת הנהלה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **הערות** | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
|  | OK | 20.5.19 | כל המשתמשים המחוברים במבנה מקבלית את ההודעה שנשלחה ע"י ההנהלה במידה ולא פג התוקף. | 1. משתמש מנהל מזין את תוכן ההודעה. 2. לחיצה על כפתור שליחה. 3. חלון אישור לפני שליחה. | הקפצת הודעה מנהלתית בהצלחה. | 22.1 |

23-TC הקפצת הודעה למשתמש

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **הערות** | **תוצאה** | **תאריך בדיקה** | **תוצאה רצויה מתאימה** | **תהליך מקרה הבדיקה** | **תיאור מקרה בדיקה** | **מזהה** |
| **בדיקה זאת**  **נעשית עבור**  **כל התרחישים שכוללים**  **בתוכם הקפצת הודעת Notification** | OK | 20.5.19 | המשתמש מקבל את ההודעה שנשלחה ע"י המערכת. | 1. המשתמש הגיע ליעד ו/או המערכת רוצה להקפיץ הודעה למשתמש. 2. המערכת מקבלת בקשה להקפצת הודעה למשתמש. 3. המערכת שולחת הודעה למכשיר המשתמש. 4. האפליקציה מאשרת הגעת ההודעה. | הקפצת הודעה ,המשתמש קיבל. | 23.1 |

## 14. פירוט מסכי האפליקציה ודוגמאות קוד

להלן פירוט מסכי האפליקציה, כל מסך יפורט בתקציר מילולי קצר, תמונות צילומי מסך וקוד רלוונטי.

רשימת המסכים המפורטים -

* התחברת
* בחירת פרופיל
* מורה
  + רישום
  + פרופיל
  + ביקורות
  + שעות
  + פרטים
* תלמיד
  + רישום
  + פרופיל
  + שעות
  + מורה
    - שיעור
    - ביקורות

##### התחברות

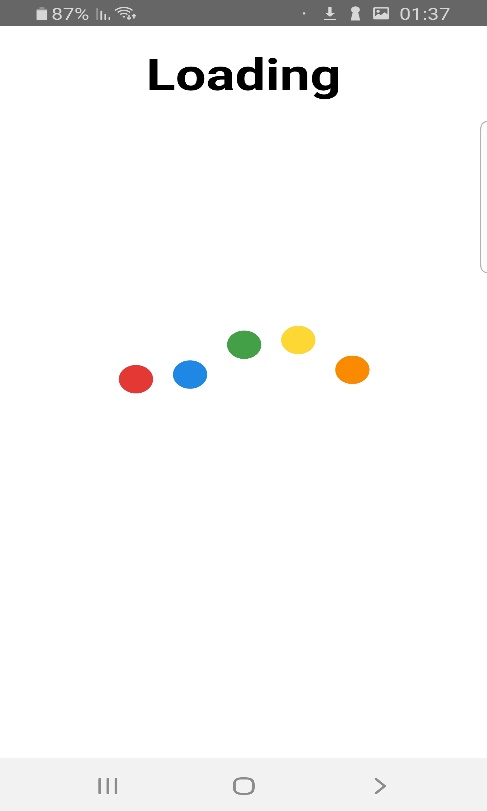
בכניסה ראשונית לאפליקציה, וכן בכל רגע נתון, המערכת בודקת את מצב ההתחברות של המשתמש במכשיר הנייד. במידה והמשתמש כבר היה מחובר, המערכת דואגת לחבר את המשתמש באופן אוטומטית. במידה ואינו מחובר המשתמש מועבר למסך ההתחברות וההרשמה של האפליקציה .

להלן הקוד שבודק את מצב החיבור של המשתמש מול המערכת.

נעשית בדיקה של השדה token במחלקה WelcomeScreen,

במידה ולא יהיה טוקן תקין בcache יצא Exeption קרי המשתמש אינו מחובר ומועבר למסך ההתחברות.

התהליך מתבצע בעת הכניסה לאפליקציה באופן ששקוף למשתמש

    async config => {

      const token = await AsyncStorage.getItem("token");

      if (token) {

        config.headers.Authorization = `Bearer ${token}`;

      }

      return config;

    },

    err => {

      return Promise.reject(err);

    }

  );

בכל פניה לAPI מבצעת הבדיקה של הTOKEN ובמידה והוא קיים והוא תקין מתבצעת בדיקה לראות האם המשתמש כבר רשום ובמידה וכן הוא יעבור לדף המשתמש שלו

const loginMethod = async navigation => {

  try {

    const response = await api.get("/api/login/verifyToken");

    if (response.data.profile === "student") {

      navigation.navigate("StudentMenu");

    } else if (response.data.profile === "teacher") {

      navigation.navigate("TeacherMenu");

    } else {

      navigation.navigate("loginFlow");

    }

  } catch (error) {

    console.log(error);

    navigation.navigate("loginFlow");

  }

};

##### בחירת פרופיל

כמו שרואים בקוד למעלה ברגע שלמשתמש אין יוזר של מורה או תלמיד הוא מגיע לדף בחירת פרופיל

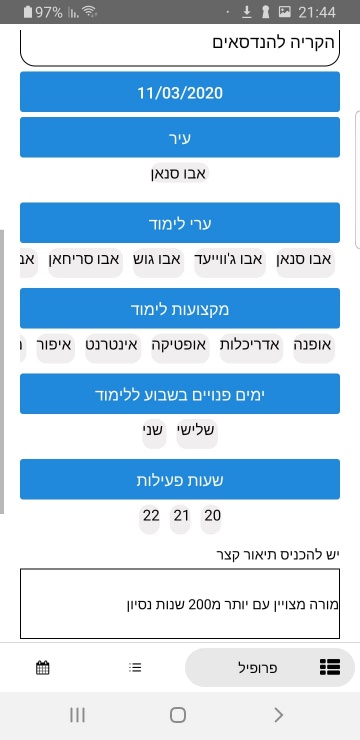


##### מורה

###### הרשמה / עדכון פרטים

במידה והמשתמש בחר בפרופיל מורה הוא יגיע לדף הבא ויצטרך להזין את הפרטים שלו

כגון שם , טלפון , מין , ערי לימוד , מקצועות לימוד ...



אותה מחלקה אחראית גם על העדכון פרטי המורה

בקוד מטה אפשר לראות את הפעולה שמאורי כפתור שמור

מתבצעת שמירה של כל השדות למשתנים תחת אובייקט obj וואלדציה לשדות

            <Button

              title="שמור"

              style={{ size: 15 }}

              onPress={() => {

                const obj = {

                  fullName: this.state.fullName,

                  phone: this.state.phone,

                  priceAtStudent: this.state.priceAtStudent,

                  price: this.state.price,

                  gender: this.state.gender,

                  city: this.state.city,

                  courses: this.state.courses,

                  studyCities: this.state.studyCities,

                  university: this.state.university,

                  generalDescription: this.state.generalDescription,

                  availablesDays: this.state.availablesDays,

                  avaiablesHours: this.state.avaiablesHours,

                  age: this.state.datePickerTitle,

                  rating: 0,

                  profile: "teacher"

                };

                if (

                  FormValidation(

                    obj.fullName,

                    obj.phone,

                    obj.priceAtStudent,

                    obj.price

                  ) ||

                  !this.state.fullName ||

                  !this.state.phone ||

                  !this.state.priceAtStudent ||

                  !this.state.price ||

                  !this.state.gender ||

                  !this.state.city ||

                  this.state.courses.length == 0 ||

                  this.state.studyCities.length == 0 ||

                  !this.state.university ||

                  !this.state.generalDescription ||

                  this.state.availablesDays.length == 0 ||

                  !this.state.avaiablesHours.length ||

                  !this.state.datePickerTitle

                ) {

                  Alert.alert("שיאה בנתונים ", "יש למלא את כל השדות");

                } else if (!this.props.Teacher.fullName) {

                  console.log("not this.props.Teacher.fullName");

                  this.printDetails(

                    obj,

                    this.props.navigation.getParam("access\_token")

                  );

                } else {

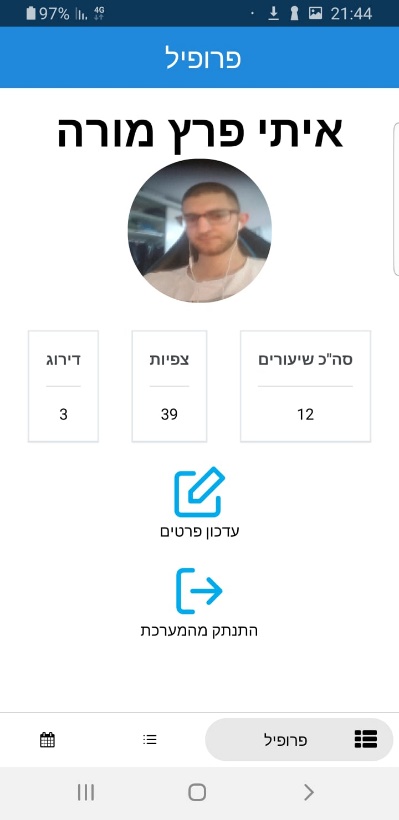
                  console.log(this.state.availablesDays);

                  this.updateDetails(obj);

                }

              }}

            />



###### פרופיל

במסך הפרופיל המורה רואה את הנתונים עליו כגון מס צפיות וסה"כ שיעורים

בנוסף הוא יכול לעדכן את הפרטים שלו ולשנות את תמונת הפרופיל

בקוד מטה אפשר לראות את תהליך הבניה של המסך

SELECTPIC אחראי על צילום התמונה כתוצאה מצליחה על אייקון התמונה

            <Text h1 style={{ textAlign: "center", marginTop: 10 }}>

              {this.props.fullName}

            </Text>

            <SelectPic

              isVisable={this.state.visable}

              profile="teacher"

              close={() => this.setState({ visable: false })}

            />

            <TouchableOpacity onPress={() => this.setState({ visable: true })}>

              {srcpic}

            </TouchableOpacity>

          </View>

          <View style={styles.cards}>

            <Card title={`סה"כ שיעורים`}>

              <Text style={styles.text}>{this.state.sumOfLessons}</Text>

            </Card>

            <Card title="צפיות">

              <Text style={styles.text}>{this.state.views}</Text>

            </Card>

            <Card title="דירוג">

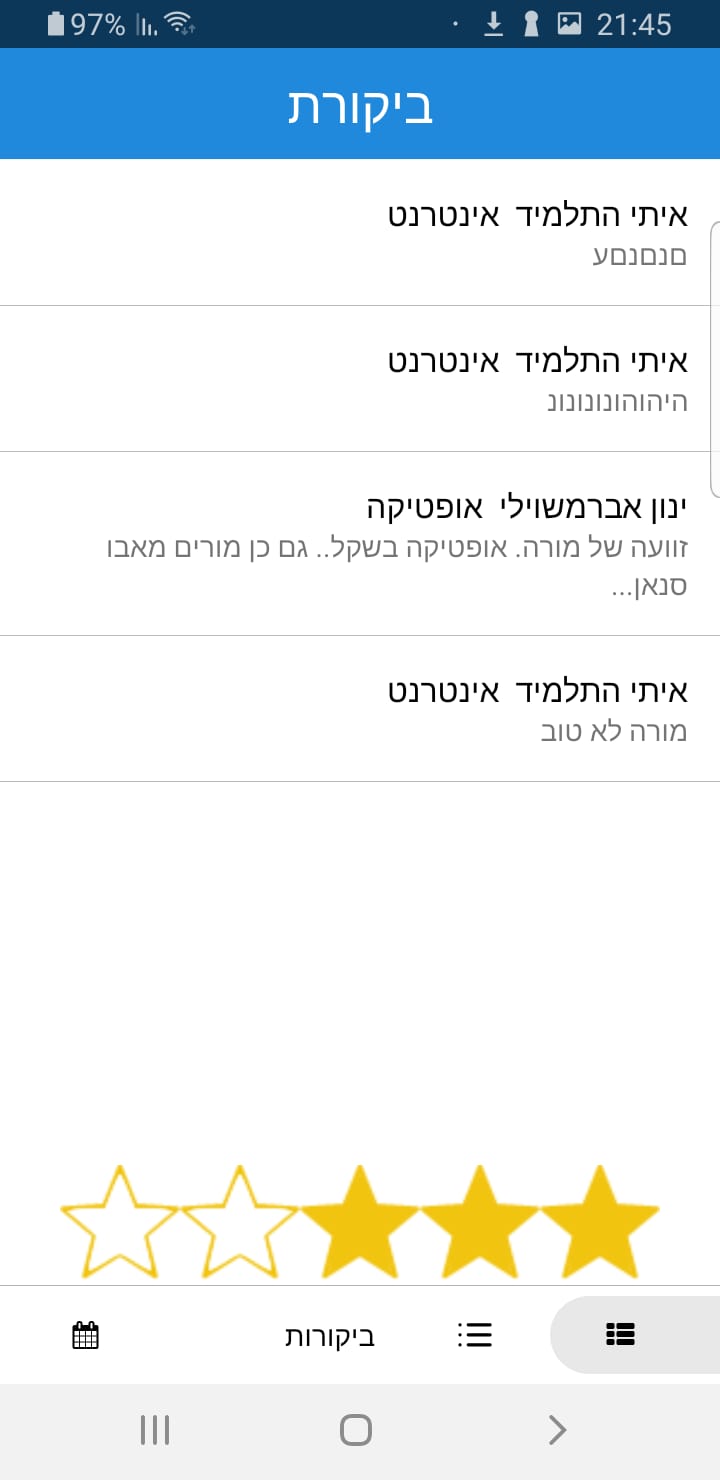
              <Text style={styles.text}>{this.state.ratingAverage}</Text>

            </Card>

          </View>

###### ביקורת

במסך הבא אפשר לראות את הביקורות שהמורה קיבל ואת הממוצע בכוכביות



הקוד מטה מראה את הבניה של המסך לפי הנתונים שחזרו מהשרת

        {reviewList.map((l, i) => (

          <ListItem

            key={i}

            title={l.studentName}

            subtitle={l.review}

            bottomDivider

          />

        ))}

      </View>

      <Rating

        ratingCount={5}

        imageSize={60}

        showRating

        ratingBackgroundColor="white"

        readonly

        startingValue={ratingSum}

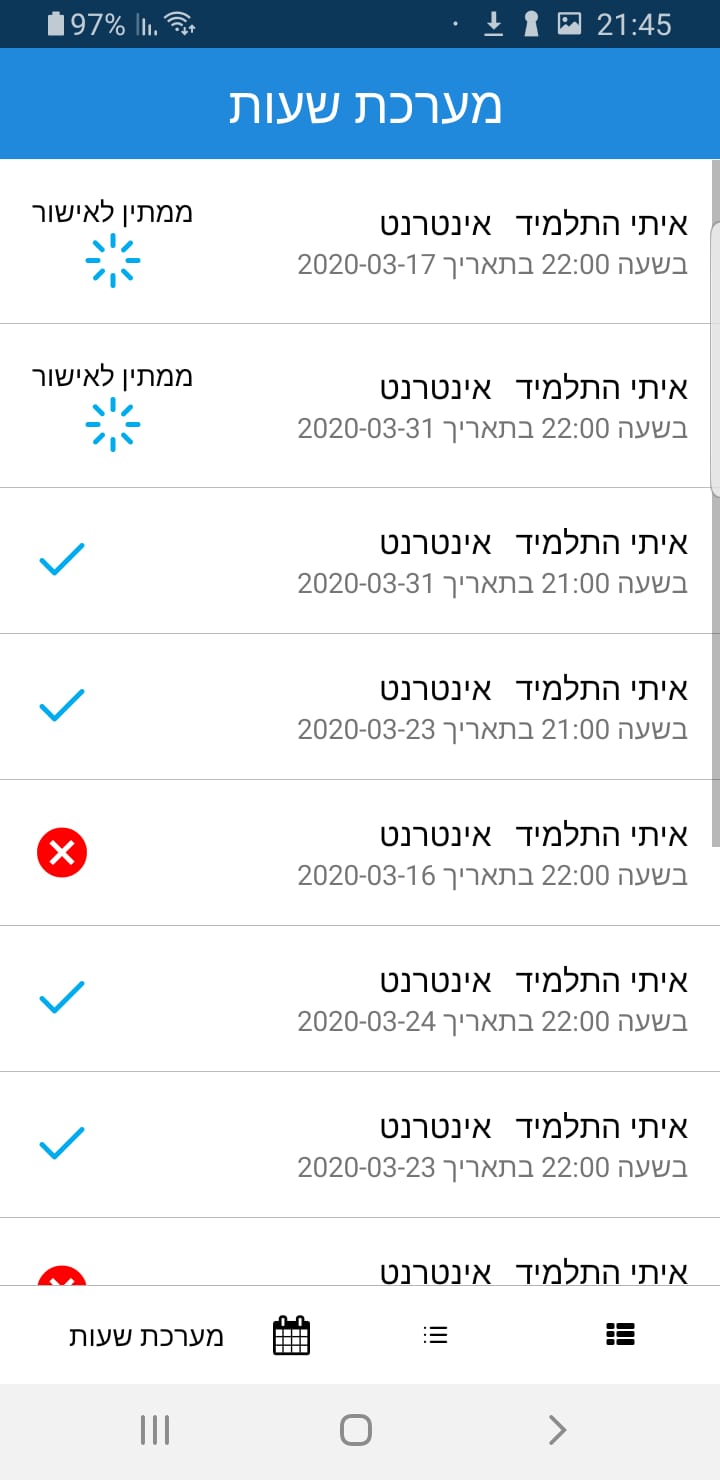
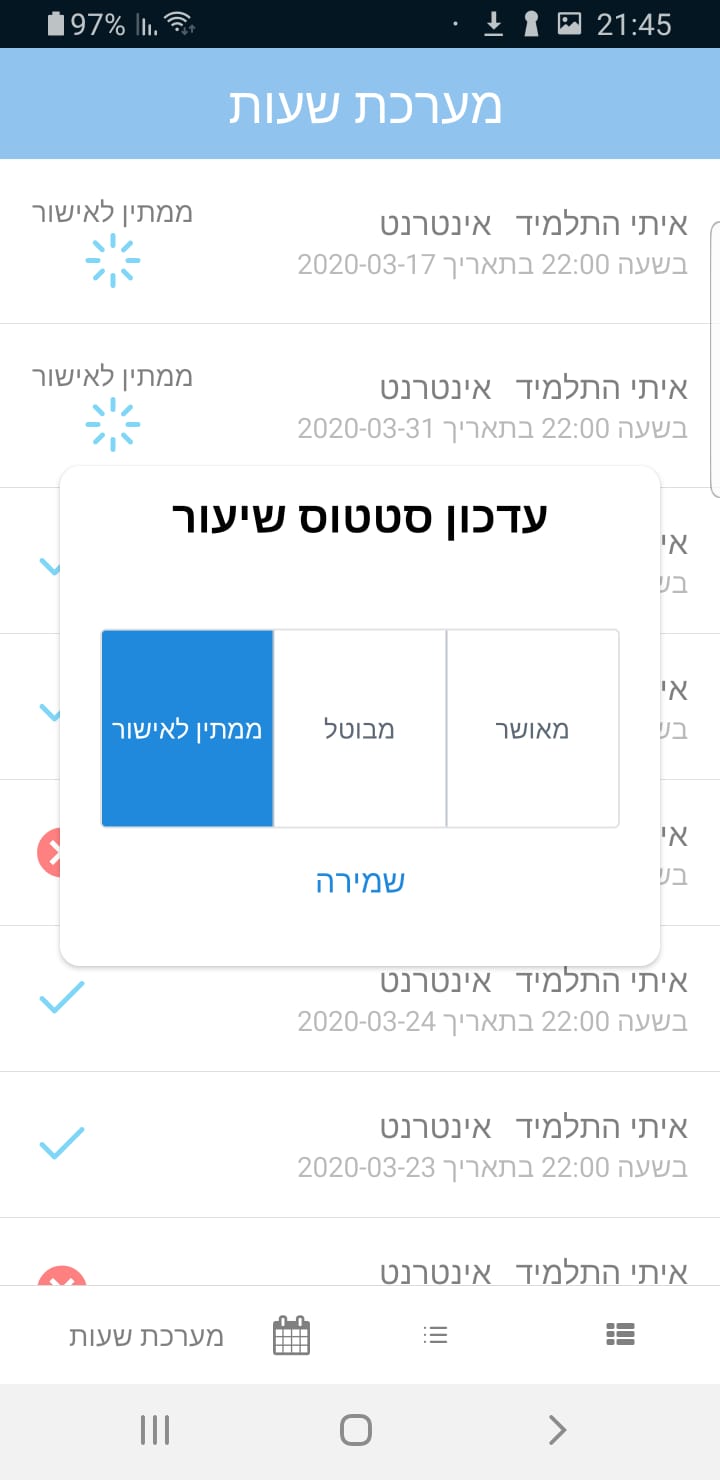
      />

###### מערכת שעות

במסך הבא

יוצגו השיעורים שנקבעו עם המורה ואת הסטטוס שלהם

המורה יכול לשנות את הסטטוס באמצעות לחיצה על האייקון של הסטטוס

קטע הקוד המתאר את בניית המסך

כאשר **TimeTableStatus**  זה באובייקט האחראי על שינוי הסטטוס

        <View>

          {timeTable.map((l, i) => (

            <View key={i}>

              <ListItem

                key={i}

                rightElement={

                  <TimeTableStatus status={l.status} tableId={l.id} />

                }

                title={l.name + "   " + `${l.cource}`}

                subtitle={`בשעה ${l.time}:00 בתאריך ${l.date}`}

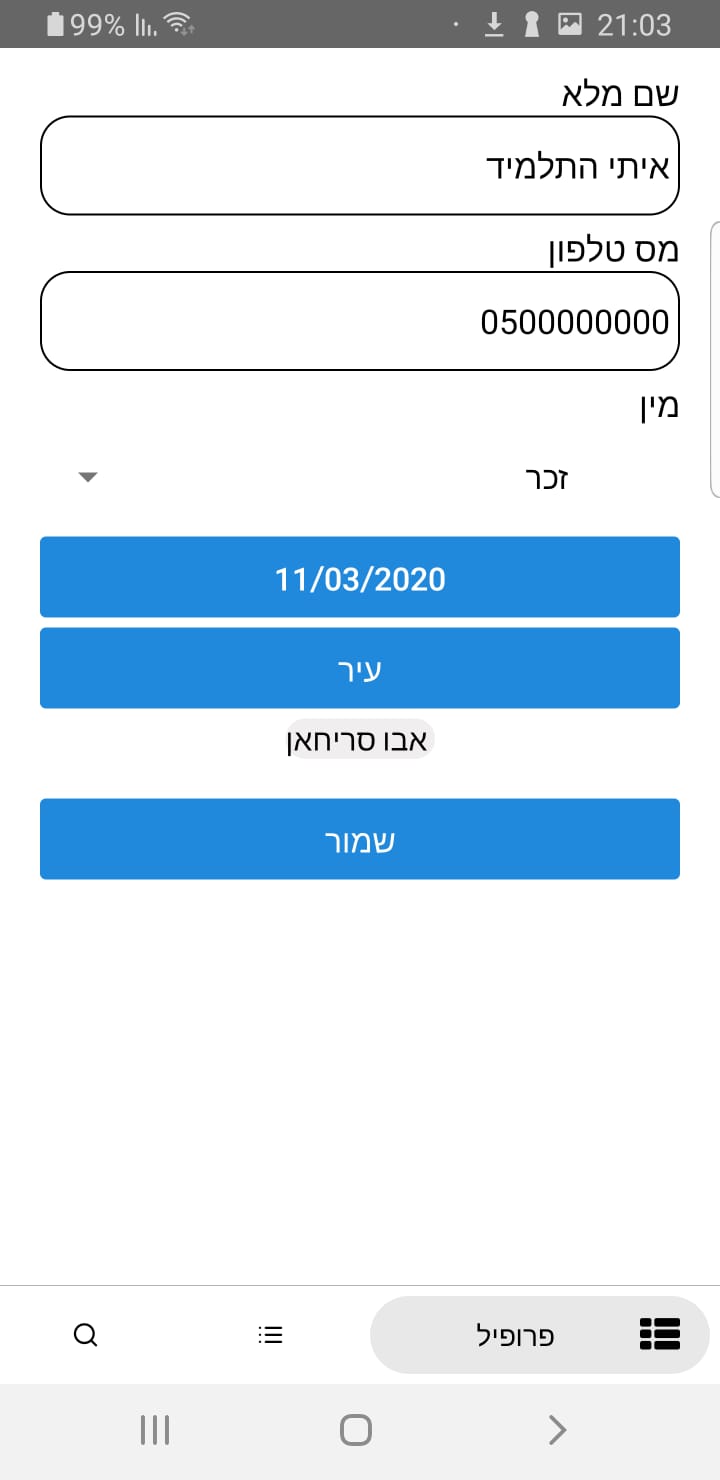
                bottomDivider

              />

            </View>

          ))}

        </View>



##### תלמיד

###### הרשמה / עדכון פרטים

במידה והמורה בחר בפרופיל תלמיד הוא יגיע למסך ההרשמה

ויצטרך למלא מספר פרטים כגון עיר , טלפון , שנת לידה

קטע הקוד מטה אחראי על ההרשמה ועל עדכון הפרטים

<Button

              title="שמור"

              buttonStyle={{ marginTop: 5 }}

              onPress={() => {

                const obj = {

                  fullName: this.state.fullname,

                  phone: this.state.phone,

                  city: this.state.city,

                  gender: this.state.gender,

                  age: this.state.datePickerTitle,

                  profile: "student"

                };

                if (

                  !FormValidation(obj.fullName, obj.phone, obj.city, obj.age)

                ) {

                  Alert.alert("שיאה בנתונים ", "יש למלא את כל השדות");

                  return;

                } else if (!this.props.fullName) {

                  this.printDetails(

                    obj,

                    this.props.navigation.getParam("access\_token")

                  );

                } else {

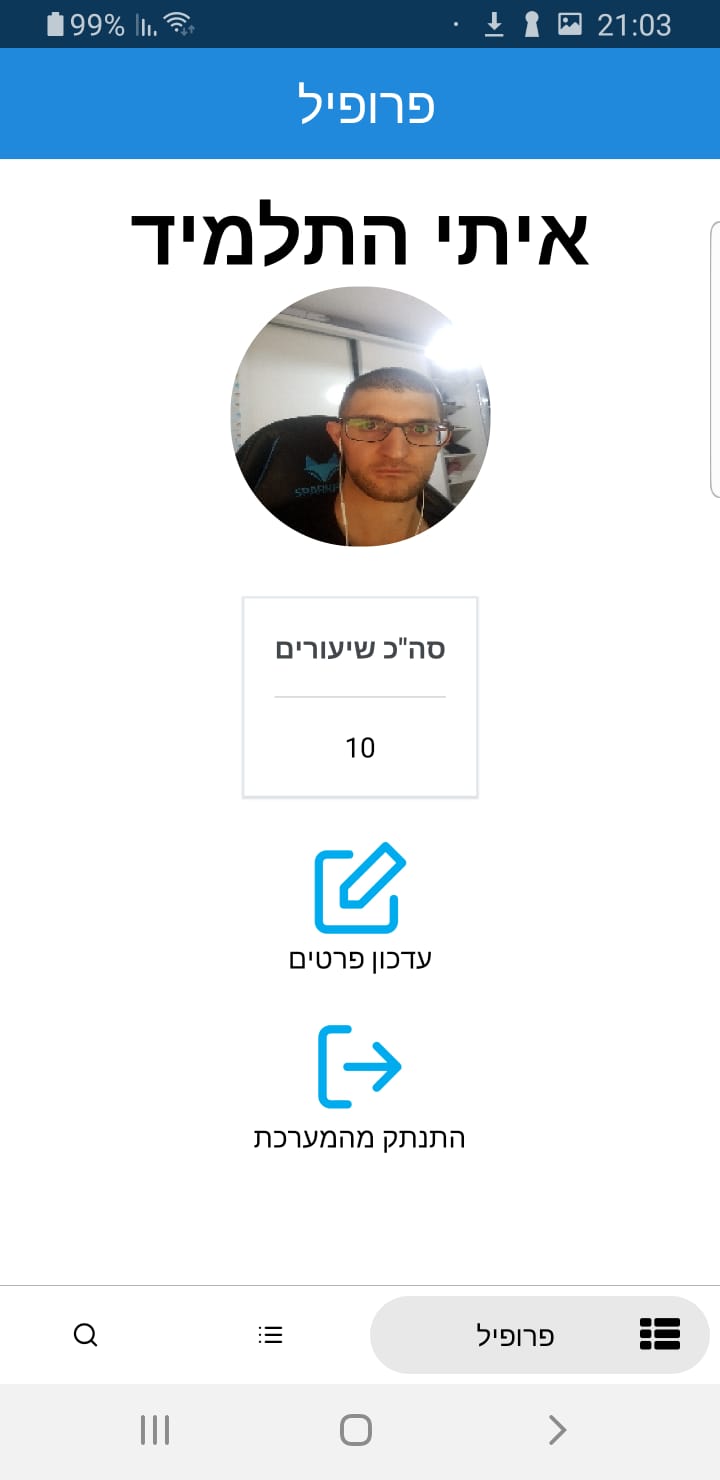
                  this.updateDetails(obj);

                }

              }}

            />

###### פרופיל



התלמיד יגיע למסך הבא לאחר שהוא יבחר בפרופיל תלמיד

הוא יכול לראות את כמות השיעורים שנקבעו ,

לעדכן את הפרטים ולבחור תמונה לפרופיל

בקוד מטה אפשר לראות את הבניה של המסך

כמות השיעורים מקבלת בAPI this.state.sumOfLessons

          <View>

            <Text h1 style={{ textAlign: "center", marginTop: 10 }}>

              {this.props.name}

            </Text>

            <SelectPic

              profile="student"

              isVisable={this.state.visable}

              close={() => this.setState({ visable: false })}

            />

            <TouchableOpacity onPress={() => this.setState({ visable: true })}>

              {srcpic}

            </TouchableOpacity>

          </View>

          <View style={styles.cards}>

            <Card title={`סה"כ שיעורים`}>

              <Text style={styles.text}>{this.state.sumOfLessons}</Text>

            </Card>

          </View>

          <TouchableOpacity

            style={{ marginTop: 20 }}

            onPress={() => {

              this.props.navigation.navigate("studnetDetails");

            }}

          >

            <Icon name="edit" type="feather" color="#00aced" size={50} />

            <Text>עדכון פרטים</Text>

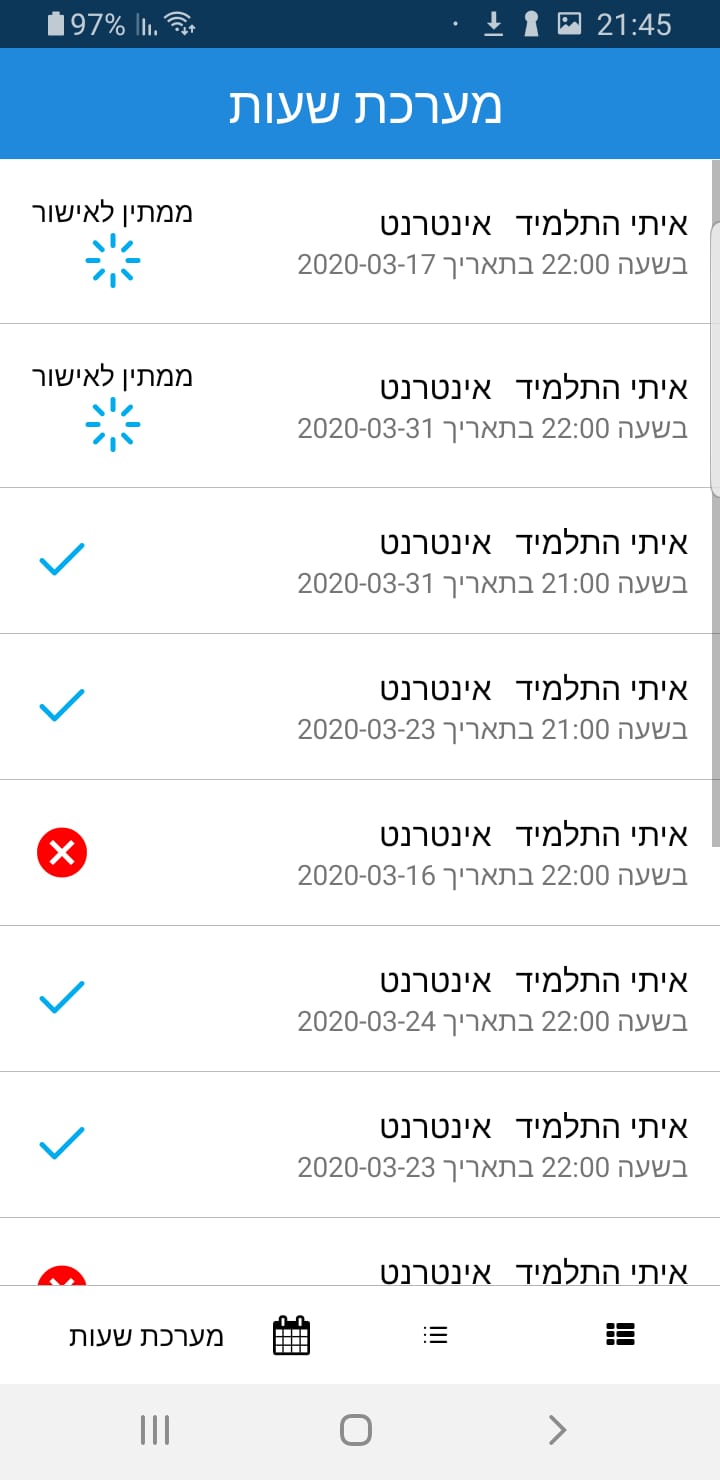
          </TouchableOpacity>

          <SignOutIcon />

###### מערכת שעות

במסך הבא

יוצגו השיעורים שנקבעו עם המורה ואת הסטטוס שלהם



מבחינת קוד מדובר על אותו הקוד רק שלתלמיד אין הרשאה לשנות את סטטוס השיעור

###### חיפוש מורה

מסך חיפוש המורה

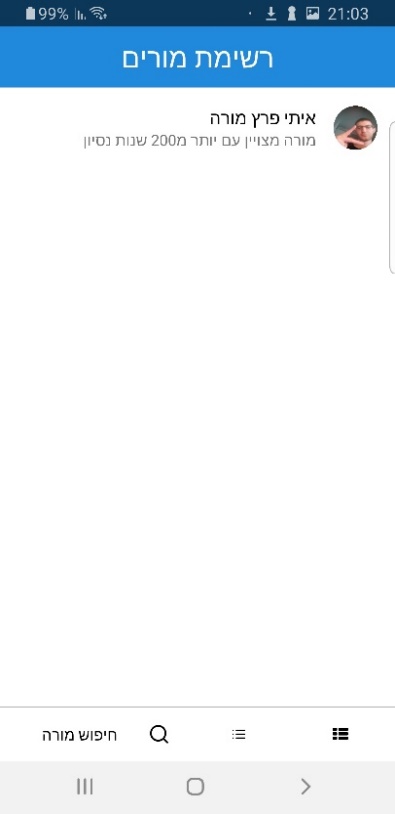
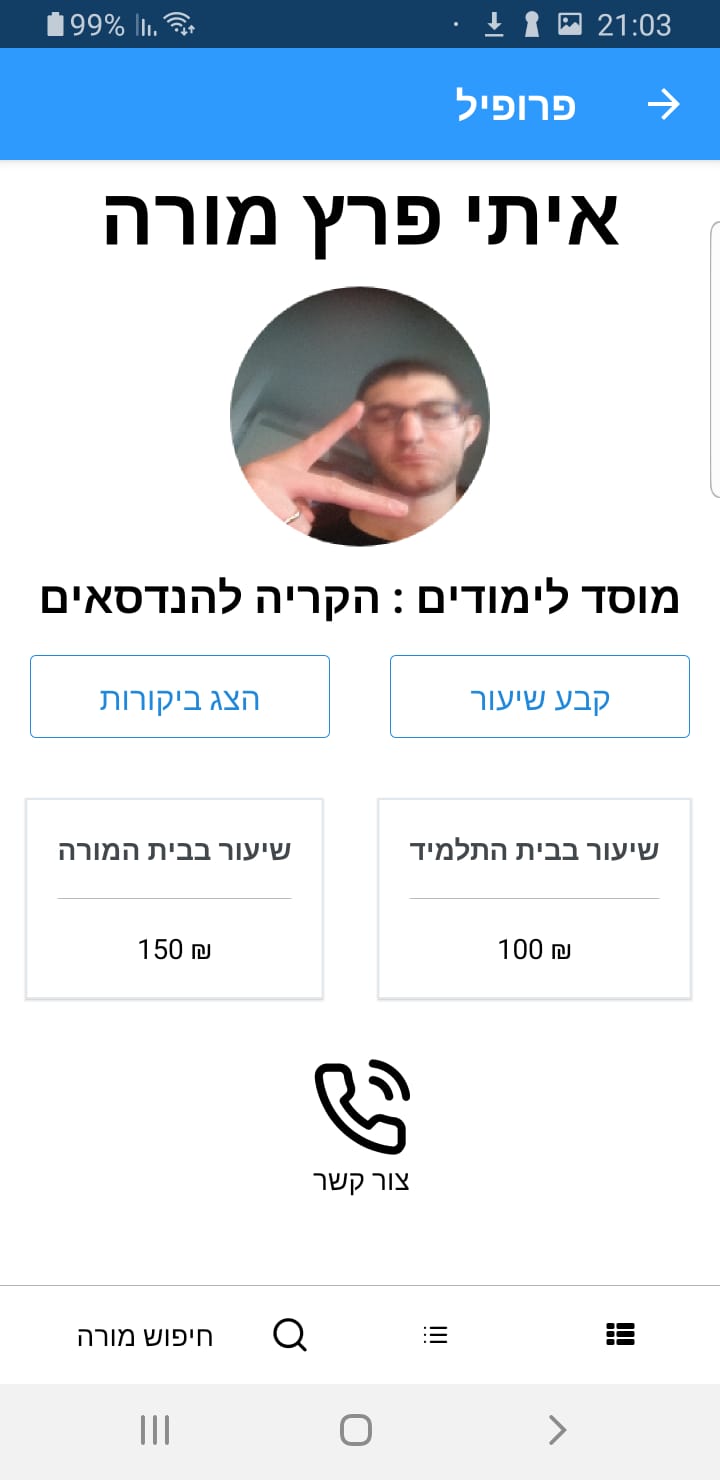
יש לבחור בעיר ובמקצוע למצוא שם מורה

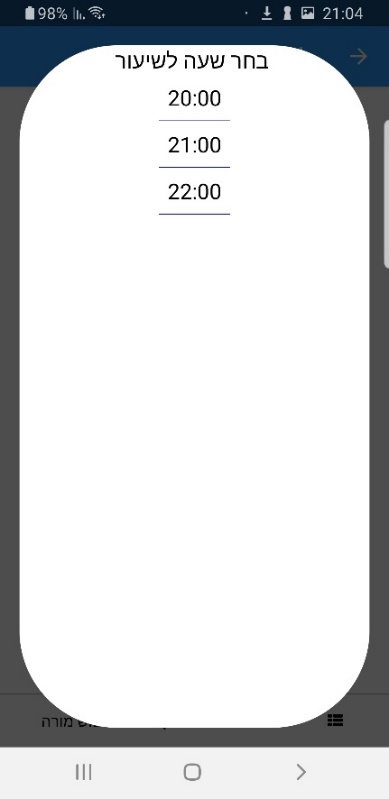
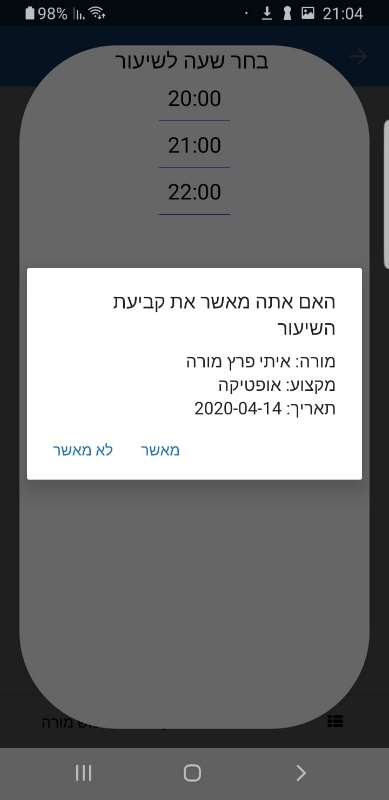
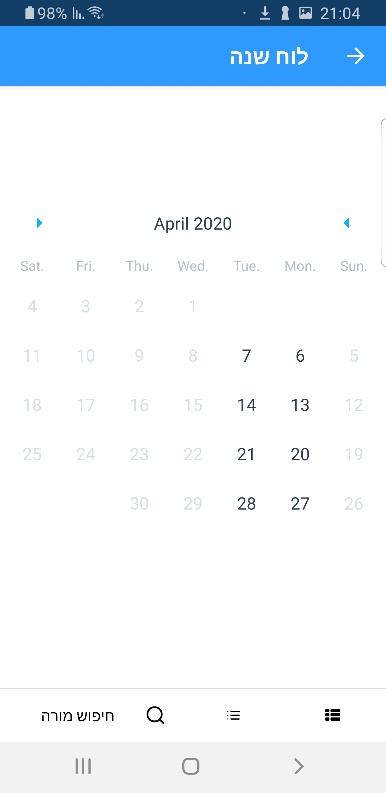
רשימת הערים נמצאת בקובץ JSON מרשות האוכלוסין והאגירה

ורשימת המקצועות מכילה את רוב המקצועות שקיימים

במסכים מטה אפשר לראות את הסדר של קביעת השיעור

1. חיפוש מורה לפי מיקום ומקצוע
2. בחירת מורה מרשימה
3. לחיצה על הצג ביקורות תציג את הביקורות על המורה
4. לחיצה על קבע שיעור תעביר ללוח שנה שבו מסומנים הימים שהוא זמין בהם
5. לאחר לחיצה על היום יתקבלו השעות שהוא זמין בהם
6. לאחר אישור השיעור תתקבל הודעה שהשיעור נקבע בהצלחה

קטעי הקוד הבאים אחראים על התהליך

**חיפוש מורה -**

הקוד מציג את 2 הלשוניות של בחירת הערים וברגע שהמשתמש בוחר בעיר או במקצוע יתבצע set שיגדיר את העיר \ מקצוע בזיכרון שמאוחר יוזר ישמש לצורף הQUERY לחיפוש המורה

 <SearchInput

        title="לחץ כאן לבחירת מקצוע"

        inputText="לחץ כאן לבחירת מקצוע"

        label={profession.name}

        data={professions}

        onSelect={name => {

          setProfession(name);

        }}

      />

      <Text style={styles.text}>עיר</Text>

      <SearchInput

        title="לחץ כאן לבחירת אזור"

        inputText="לחץ כאן לבחירת אזור"

        label={city.name}

        data={cities}

        onSelect={name => {

          setCity(name);

        }}

      />

      <TouchableOpacity style={styles.button}>

        <Button

          title="המשך"

          onPress={() => {

            if (city == 0 || profession == 0) {

              Alert.alert("חסרים פרטים", "יש לבחור עיר ומקצוע");

              return;

            }

            fetchListOfTeachers(city, profession, navigation);

          }}

        />

**רשימת מורים -**

קטע קוד הבא אחראי על הצגת כל המורים לפי הפרמטרים שנבחרו בחלון הקודם מול הAPI

ברגע שהמשתמש ילחץ על מסך המורה תתרחש קריאה של onpress ויבחר המורה והמשתמש יעבור למסך שנקרא teacher profile

        {teacherList.map((l, i) => (

          <TouchableOpacity

            onPress={() => {

              l.profession = profession;

              selectTeacher(l);

              navigation.navigate("TeacherProfile");

            }}

            key={i}

          >

            <ListItem

              keyExtractor={item => item.fullName}

              key={l => l.name}

              leftAvatar={{

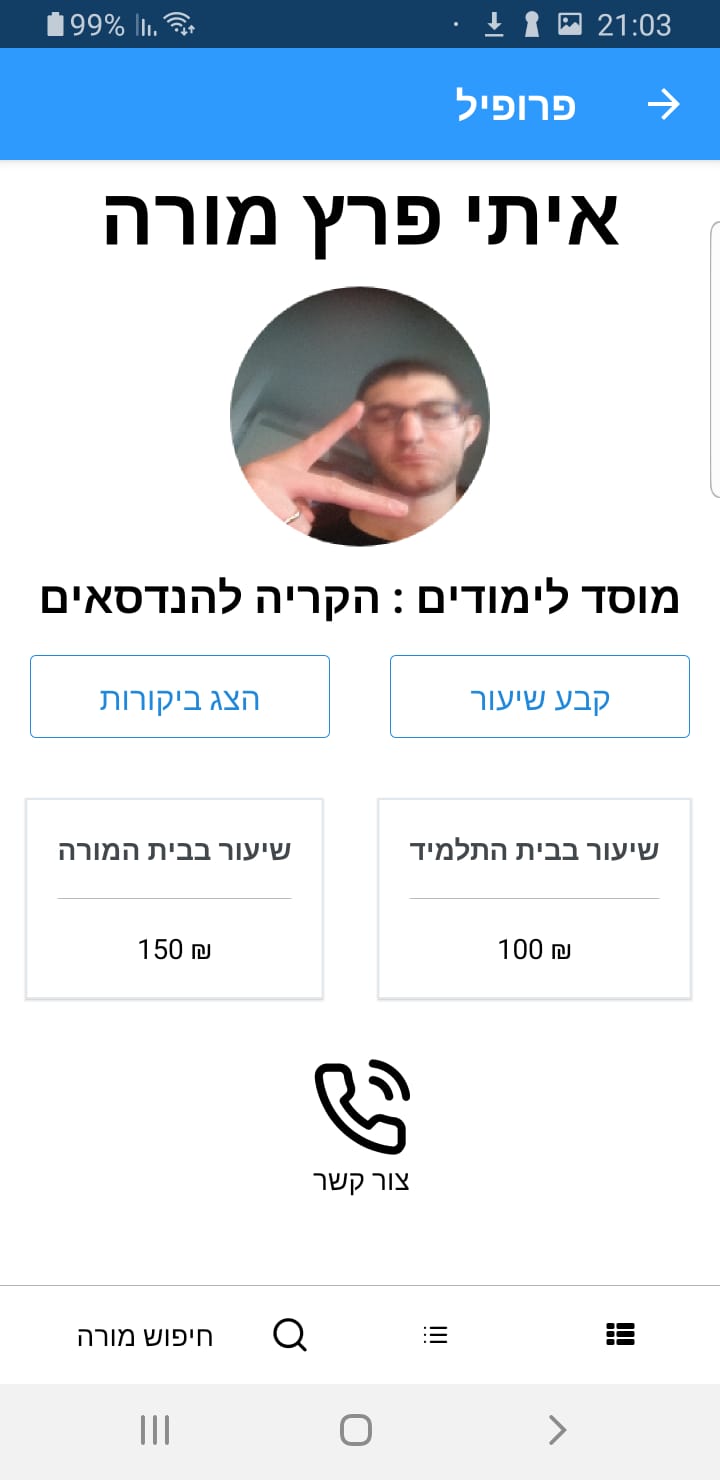
                source: { uri: l.pic || "https://i.imgur.com/XlQzxuT.png" }

              }}

              title={l.fullName}

              subtitle={l.generalDescription}

              bot

**הצגת פרופיל המורה לצורך קביעת שיעור**

במסך הבא המשתמש יוכל לראות את הביקורות על המורה

וליצור איתו קשר דרך האייקון של הטלפון

בנוסף אפשר לראות את מחיר השיעור בבית התלמיד ואצל

המורה

קטע הקוד מטה מציג את האפשרויות של המסך מעלה

לחיצה על קבע שיעור תעביר את המשתמש למסך של CALENDER ולחיצה על הצג ביקורת תעביר את המשתמש למסך ביקורות

לחיצה על צור קשר תפעיל את Linking.openURL שיציג את מספר הטלפון במסך חיוג

   <Button

            title="קבע שיעור"

            type="outline"

            containerStyle={{ margin: 15, width: 150 }}

            onPress={() => {

              this.props.navigation.navigate("Calender");

            }}

          />

          <Button

            containerStyle={{ margin: 15, width: 150 }}

            title="הצג ביקורות"

            type="outline"

            onPress={() => {

              this.props.getReviews(this.state.email);

              this.props.navigation.navigate("StudentReview", {

                email: this.state.email

              });

            }}

          />

        </View>

        <View style={styles.cards}>

          <Card title={`שיעור בבית התלמיד`}>

            <Text style={styles.text}>{this.state.price + " ₪"}</Text>

          </Card>

          <Card title="שיעור בבית המורה">

            <Text style={styles.text}>{this.state.priceAtStudent + " ₪"}</Text>

          </Card>

        </View>

        <TouchableOpacity

          style={{ marginTop: 30 }}

          onPress={() => {

            Linking.openURL(`tel:${this.state.phone}`);

          }}

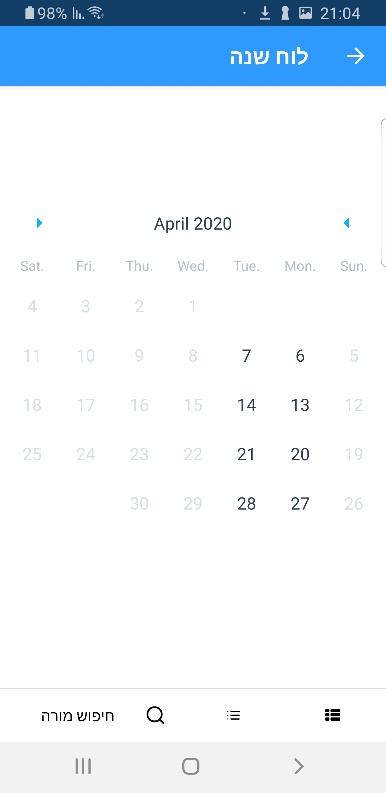
        >

          <Feather name="phone-call" size={50} />

          <Text>צור קשר</Text>

        </TouchableOpacity>

**קביעת יום לשיעור**

במסך הבא יוצגו כל הימים הפנויים שהמורה מלמד בהם

כל הימים הלא פנויים יוצגו כאשר הם מאופררים

לחיצה על התאריך המבוקש לבצע פניה לAPI

שיבדוק באיזה שעות המורה פנוי כאשר מתבעת בדיקה גם

לגבי שעות אשר נתפסו על ידי שיעורים אחרים שנקבעו

הקוד למטה אחראי על ההצגה של הלוח שנה

כאשר הוא מקבל כפרמטר את הימים שבהם המורה פנוי ומציג אותם למשתמש

אפשר לראות שהMIN DATE הוא היום , ככה שלא יהיה אפשר לקבוע שיעור לאתמול

   <Calendar

        onMonthChange={month => {

          setCurrentDate(month.dateString);

          setMonthsList(

            getMonthsArray(

              month.year.toString(),

              month.month,

              availablesDays,

              currentDate

            )

          );

        }}

        // Collection of dates that have to be marked. Default = {}

        current={currentDate}

        minDate={new Date()}

        //  displayLoadingIndicator

        // Handler which gets executed on day press. Default = undefined

        onDayPress={async ({ dateString }) => {

          try {

            await getAvaiablehours(email, dateString, setAvailablesHouers);

            setSelectetDate(dateString);

            setVisable(true);

          } catch (error) {

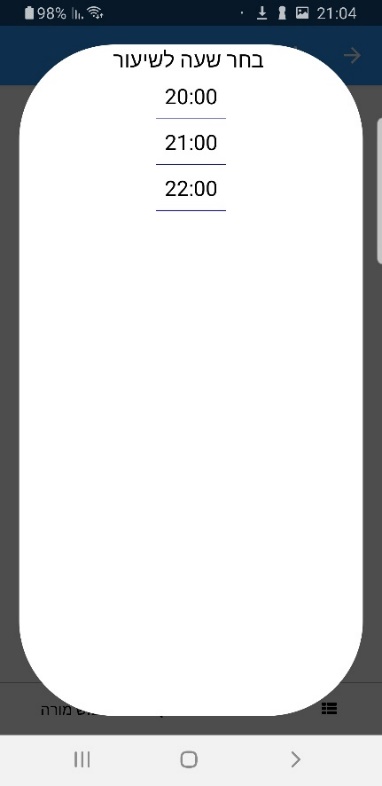
            console.log(error);

          }

        }}

        hideExtraDays={true}

        markedDates={monthList}

      />

לאחר לחיצה על התאריך יתקבל מהAPI רשימת שעות שהמורה זמין בהם

וזה יקפוץ בחלון חדש על המסך

אפשר לראות שבעת לחיצה מתבצעת קריאה ואז לאחר התשובה SETVISIABLE

שווה לTRUE שזה אחראי על ההצגה של החלון על השעות

חלון האחראי על הצגת השעות הפנויות

Flatlist אחראי על קבלת מערך ויציאת אלמנטים מאותו המערכת ובמקרה הזה כל אלמנט הוא שעה

      <Modal

        isVisible={visable}

        onBackdropPress={() => setVisable(false)}

        transparent={true}

        animationType="slide"

        onBackButtonPress={() => {

          setVisable(false);

        }}

 <Text style={{ textAlign: "center", fontSize: 20 }}>

            {" "}

            בחר שעה לשיעור

          </Text>

          <FlatList

            data={availablesHouers}

            keyExtractor={(key, i) => i.toString()}

            renderItem={({ item }) => {

במידה והשיעור יאופר יקפוץ חלון המבקש את האישור לקביעת השיעור

   Alert.alert(

                          "האם אתה מאשר את קביעת השיעור",

                          `מורה: ${teacherfullName}\nמקצוע: ${profession.name}\nתאריך: ${selectetDate}   `,

                          [

                            {

                              text: "מאשר",

                              onPress: () => {

                                apointmentLesson(

                                  studentName,

                                  studentEmail,

                                  teacherfullName,

                                  email,

                                  profession.name,

                                  selectetDate,

                                  item

                                );

                                setVisableOK(true);

                              }

                            },

                            {

                              text: "לא מאשר"

                            }

                          ],

                          { cancelable: false }

                        );

                        // setVisable(false);

                      } catch (error) {

                        console.log(error);

                      }

                    }}

                  >

ולאחר האישור

יתקבל עדכון שהשיעור נקבע בהצלחה

  <AwesomeAlert

          show={true}

          showProgress={false}

          title="השיעור נקבע בהצלחה"

          message="למעבר לתפריט הראשי לחץ המשך"

          closeOnTouchOutside={false}

          closeOnHardwareBackPress={false}

          showCancelButton={false}

          showConfirmButton={true}

          confirmText="המשך"

          confirmButtonColor="#DD6B55"

          onCancelPressed={() => {}}

          onConfirmPressed={() => {

            try {

              setVisableOK(false);

              setVisable(false);

              navigation.popToTop();

            } catch (error) {

              console.log(error);

            }

          }}

        />

##### צד שרת

###### שליחת מייל

בכל קביעת שירות , שינוי סטטוס שיעור , הרשמה נשלח מייל למשתמשים

המייל נשלח באמצעות השרת דרך SMTP

זה מתבצע באמצעות EVENTS שהם מתאימים לתהליכים אסינכרונים

const EventEmitter = require("events").EventEmitter;

const emitter = new EventEmitter();

const sendEmailOnRegistration = require("./send\_email\_on\_registration");

const sendEmailOnChangeStatus = require("./send\_email\_on\_changelessonstatus");

const sendEmailOnNewLesson = require("./send\_email\_on\_newLesson");

// const someOtherListener = require('./some\_other\_listener');

// const doSomethingEntirelyDifferent = require('./do\_something\_entirely\_different');

emitter.on("user-registered", sendEmailOnRegistration);

emitter.on("teacher-changestatus", sendEmailOnChangeStatus);

emitter.on("teacher-apointmentLesson", sendEmailOnNewLesson);

module.exports = emitter;

דוגמה לבניית שלד המייל

module.exports = obj => {

  const { teacherName, studentName, date, time, teacherEmail } = obj;

  const htmlBody = mailgen(teacherName, studentName, date, time);

  const subject = `  נקבע עבורך שיעור חדש - Teacher4U`;

  const mailService = new MailService(

    teacherEmail,

    subject,

    "heyy there",

    htmlBody

  );

  Logger.info(`new lesson , sending email to ${teacherEmail}`);

  mailService.sendEmail();

};

###### JWT

כאשר משתמש מתחבר בפעם הראשונה מתבצעת פניה לשרתים של GOOGLE

כדי לאמת את הטוקן שהתקבל , מדובר על שיטת אימות שנקראת OAUTH2

כדי שהתהליך לא יחזור על עצמו אנחנו יוצרים JWT עם ASYMETRIC KEY

כאשר המשתמש יפנה אלינו בפעם הבאה אנחנו נבצע VALIDATION לJWT במקום לפנות לGOOGLE

/\* eslint-disable no-undef \*/

const jwt = require("jsonwebtoken"),

  fs = require("fs"),

  privateKey = fs.readFileSync(\_\_dirname + "/../security/private.key"),

  cert = fs.readFileSync(\_\_dirname + "/../security/public.pem"); // get public key

module.exports = class JsonWebToken {

  constructor(obj) {

    this.obj = obj;

  }

  createJwt() {

    const token = jwt.sign(JSON.stringify(this.obj), privateKey, {

      algorithm: "RS256"

    });

    return token;

  }

  verifyJwt() {

    const response = jwt.verify(this.obj, cert);

    return response;

  }

};

###### העלאת תמונות

כאשר המשתמש בוחר תמונת פרופיל

אנחנו משתמשים בשירות של IMGUR שמקבל תמונה בbase64 ומחזיר את הURL של התמונה

דפי הנהלה )משתמש הנהלה(

הנפקת דוחות

## 15. מסד נתונים – MongoDb

בסיס נתונים NOSQL אשר יושב בענן של MONGO ATLAS

הנתונים נשמרים בו בתצורה של JSON וזה מאוד עוזר

כי על העבודה שלנו בפרויקט היא בJAVASCRIPT וזה אומר שהנתונים

נמצאים בjson באופן אוטומטי

בנוסף לnodejs יש יכולת חיבור מעולה על הבסיס נתונים הזה מה שנותן

ביצועים מעולים ויכולת לטפל בהמון בקשות בו זמנית

mongodb והסברים כללים-

**מערכת שעות -**

**נשמרים כל השיעורים והסטטוס שלהם**

const TimeTableSchema = new mongoose.Schema({

  teacherEmail: {

    type: String,

    required: true

  },

  teacherName: {

    type: String,

    required: true

  },

  studentEmail: {

    type: String,

    required: true

  },

  studentName: {

    type: String,

    required: true

  },

  hasReview: {

    type: Boolean,

    required: false

  },

  cource: {

    type: String,

    required: true

  },

  date: {

    type: String,

    required: true

  },

  time: {

    type: Number,

    required: true

  },

  status: {

    type: String,

    default: "awating"

  }

});

**פרופיל מורה-**

כל הפרטים של המורה נשמרים

const TeacherSchema = new mongoose.Schema({

  age: {

    type: String,

    required: true

  },

  email: {

    type: String,

    required: true

  },

  gender: {

    type: String,

    required: true,

    enum: ["נקבה", "זכר"]

  },

  fullName: {

    type: String,

    required: true

  },

  views: {

    type: Number,

    required: true

  },

  city: {

    type: String,

    required: true

  },

  phone: {

    type: String,

    required: true

  },

  pic: {

    type: String,

    required: false

  },

  courses: {

    type: [String],

    required: true

  },

  studyCities: {

    type: [String],

    required: true

  },

  university: {

    type: String,

    required: true

  },

  generalDescription: {

    type: String,

    required: true

  },

  priceAtStudent: {

    type: Number,

    required: true

  },

  price: {

    type: Number,

    required: true

  },

  availablesDays: {

    type: [String],

    required: true,

    default: undefined

  },

  avaiablesHours: {

    type: [String],

    required: true,

    default: undefined

  },

  rating: {

    type: Number,

    default: 0

  },

  profile: {

    type: String,

    required: true

  }

});

  }

**פרופיל תלמיד-**

כל הפרטים של התלמיד נשמרים

עבור כל מפתח משתמש נשמר המפתחות של החברים const mongoose = require("mongoose");

const StudentSchema = new mongoose.Schema({

  age: {

    type: String,

    required: true

  },

  email: {

    type: String,

    required: true

  },

  gender: {

    type: String,

    required: true

  },

  fullName: {

    type: String,

    required: true

  },

  pic: {

    type: String,

    required: false

  },

  city: {

    type: String,

    required: true

  },

  phone: {

    type: String,

    required: true

  },

  profile: {

    type: String,

    required: true

  }

});

**ביקורות –**

בהוספת ביקורת מתווסף LESSONID כדי שלא יהיו 2 ביקורות על אותו שיעור

const RatingSchema = new mongoose.Schema({

  lessonId: {

    type: String,

    required: true

  },

  rating: {

    type: String,

    required: true

  },

  review: {

    type: String,

    required: true

  },

  teacherEmail: {

    type: String,

    required: true

  },

  studentName: {

    type: String,

    required: true

  },

  cource: {

    type: String,

    required: true

  }

});

אבטחת מידע

כל הנתונים שבבסיס נתונים יושבים בענן מאובטח של MONGO ATLAS

ככה שהגישה לDB תתבצע רק עם יוזר וסיסמה ייעודיים

בנוסף

הקוד עצמו רץ בסביבה של HEROKU שהם בין החברות המובילות בשוק להרצת מערכות API על ענן

ובאמצעות עבודה עם משתני סביבה אנחנו דואגים שהסיסמאות לא יזלגו לבחוץ ככה שגם במקרה שמשתמש השיג את הקוד הוא לא יוכל לעשות איתו כלום כי הוא לא כולל את הסיסמאות

מצורף קובץ ההגדרות ואפשר לראות שלא רשום יוזר וסיסמה בכל הפרוייקט

  BASE\_API: process.env.BASE\_API,

  PORT: process.env.PORT || 3000,

  MONGO\_CONNECTION\_STRING:

    process.env.MONGO\_CONNECTION\_STRING\_TEST ||

    process.env.MONGO\_CONNECTION\_STRING,

  SMTP\_USER: process.env.SMTP\_USER,

  SMTP\_PASS: process.env.SMTP\_PASS,

  SMTP\_HOST: process.env.SMTP\_HOST,

  SMTP\_PORT: process.env.SMTP\_PORT,

  REDIS\_CONNECTION\_STRING: process.env.REDIS\_CONNECTION\_STRING,

  REDIS\_PORT: process.env.REDIS\_PORT,

  REDIS\_PASSWORD: process.env.REDIS\_PASSWORD,

  OAUTH\_FACEBOOK: process.env.OAUTH\_FACEBOOK,

  OAUTH\_GOOGLE: process.env.OAUTH\_GOOGLE,

  CLIENT\_ID: process.env.CLIENT\_ID,

  APIGURID: process.env.APIGURID

**Authentication**

ההזדהות לשירות הוא באמצעות 2 אפשרויות

1. התחברות דרך GOOGLE וביצוע שליחה של הaccess token
2. שליחה של JWT שנחתם על ידי השרת (בכל בקשה לapi)

כמו שאפשר לראות יהיה קשה לעבור את ההגנות האלו

את ההזדהות של גוגל כמעט ולא אפשרי לפרוץ

ומבחינת השירות

אני משתמש בתעודות אבטחה אסימטריות כאשר אני מצפין את הJWT

ואז אני יוכל לבצע VALIDATION לJWT באמצעות הPUB וככה אני ידע שאין משהוא שמתחזה למשתמש אחר