

הקדמה

לפני כשנה, כשנדרשנו לבחור נושא לפרויקט הגמר, התלבטנו בין מספר אפשרויות, כאשר כל אפשרות הייתה מאתגרת בפני עצמה.

לבסוף, החלטנו לבחור בתחום **חיפוש מורים פרטיים**, מאחר ולאחר מחקר נוכחנו לדעת כי לתחום זה יש ביקוש רב – והמענה בשוק אינו מספיק אינטואיטיבי. נוסף על כך, האפליקציה נותנת במה גם למי שאינו מורה במקצועו – לחלוק מהידע שלו ולתת מענה לאנשים המבקשים הדרכה. בנוסף, לדעתנו תחום זה הינו בעל פוטנציאל רב והתחומים שבהם אפשר לקבל שיעורים הם רבים.

במהלך העבודה על הפרויקט, חקרנו ולמדנו רבות על המצב בשוק והרצון להשתמש באפליקציה מהסוג עליו עבדנו. נחשפנו לאתגרים הרבים שיש ביצירת אפליקציה שכזאת למכשיר הסמארטפון, ובכלל, לתהליך המורכב שביצירת אפליקציה כוללת העונה לבעיה מורכבת.

במקביל, נחשפנו לפתרונות הקיימים היום בשוק, והסקנו כי היות והם ניתנים בסביבת ה-WEB, הפתרונות הללו לא נותנים מענה אינטואיטיבי, נח ונגיש מספיק למשתמשים שהיום – פשוט רוצים את הכל מהסמארטפון.

בספר זה מובא התהליך אותו עברנו, החל משלב הרעיון, דרך 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. אפיון 4](#_Toc37973693)

[4. מסמך דרישות SRS – 9](#_Toc37973694)

[bvnbvnbvnbvnbvn 9](#_Toc37973695)

[Activity Diagrams – ותרשימי פעילות Use Cases – תרחישי שימוש .5 10](#_Toc37973696)

[מסמכי SAD 18](#_Toc37973697)

[6. ארכיטקטורה פיזית 19](#_Toc37973698)

[שם 20](#_Toc37973699)

[מהות 20](#_Toc37973700)

[עקיבות לדרישות 20](#_Toc37973701)

[Teacher4U Application 20](#_Toc37973702)

[לב המערכת, מותקן על המכשיר הנייד, האפליקציה מנהלת את כל דרישות המערכת ומתקשרת עם כל הגורמים החיצוניים במערכת 20](#_Toc37973703)

[Heroku NodeJS Api 20](#_Toc37973704)

[צד השרת של המערכת 20](#_Toc37973705)

[MongoDb atlas DB 20](#_Toc37973706)

[בסיס נתונים מבוסס NoSql 20](#_Toc37973707)

[Web Browser 20](#_Toc37973708)

[הדפדפן דרכו ההתחברות לממשק אדמין תתבצע 20](#_Toc37973709)

[7. ארכיטקטורה לוגית 21](#_Toc37973710)

[8. תרשימי רצף ברמת רכיבים– Sequence Diagrams 23](#_Toc37973711)

[מסמכי SDD 28](#_Toc37973712)

[9. מודל ישויות במרחב הבעיה(PDOM) 28](#_Toc37973713)

[10. תרשים מחלקות – UML 29](#_Toc37973714)

[11. תרשימי רצף ברמת עצמים. 31](#_Toc37973715)

[.(State Chart )מכונת מצבים .12 32](#_Toc37973716)

[13. בדיקות Test cases– מקרי בדיקה. – 33](#_Toc37973717)

[14. פירוט מסכי האפליקציה ודוגמאות קוד 37](#_Toc37973718)

[15. מסד נתונים – MongoDb 62](#_Toc37973719)

# 1. אפיון

## סיפור לקוח

**Teacher4U** היא אפליקציה המיועדת למכשירי טלפון חכמים המריצים את מערכת ההפעלה android.

האפליקציה משמשת כרשת חברתית המפגישה בין תלמידים למורים פרטיים, כך שכל אדם יכול להעניק מהידע שלו (בין אם הוא מורה מקצועי ובין אם הוא אדם מן השורה בעל ידע בתחום מסוים) 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 עבור מערכת השעות של התלמיד, בו הוא יוכל לצפות בשיעורים שקבע.
3. 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 והדירוג שלו (ראה בהמשך).

ע"י בחירה במורה יופיע הפרופיל המלא של המורה ובו כל המידע עליו. כאן יוכל התלמיד ליצור קשר עם המורה או להתחיל הליך קביעת שיעור הכולל תאריך ושעה ומלל חופשי – הודעה זו תשלח למורה שיצטרך לאשר אותה ובכך יקבע השיעור.

מיד לאחר קביעת השיעור יוכל התלמיד להוסיף ביקורת קצרה ולדרג את המורה.

**הרשמה כמורה**:

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

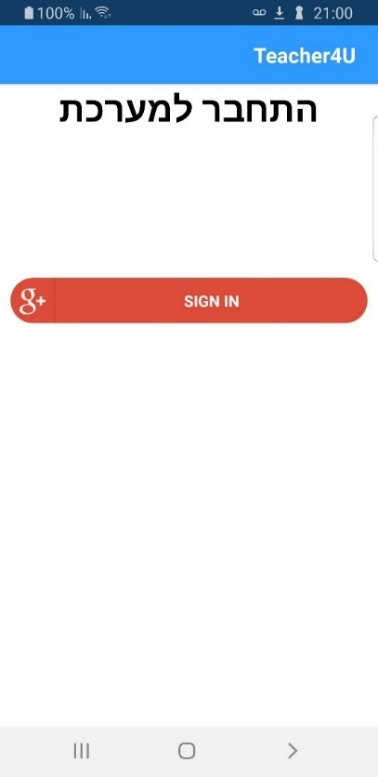
1. מסך ייעודי המציג את פרטיו האישיים ובו האפשרות לשנות אותם.
2. מסך ייעודי עבור מערכת השעות של המורה, בו הוא יוכל לצפות בכל השיעורים שנקבעו לו, לשנות או/ו לבטל שיעורים – מה שישלח מיד הודעה לתלמיד על כל שינוי ובקשה לאישור שלו על השינוי.
3. מסך ייעודי המציג את הביקורות אותם קיבל המורה והדירוג שלו באותו זמן.

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

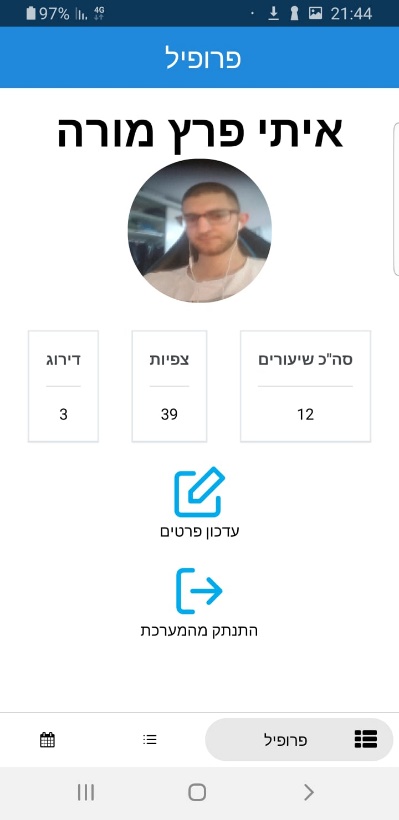
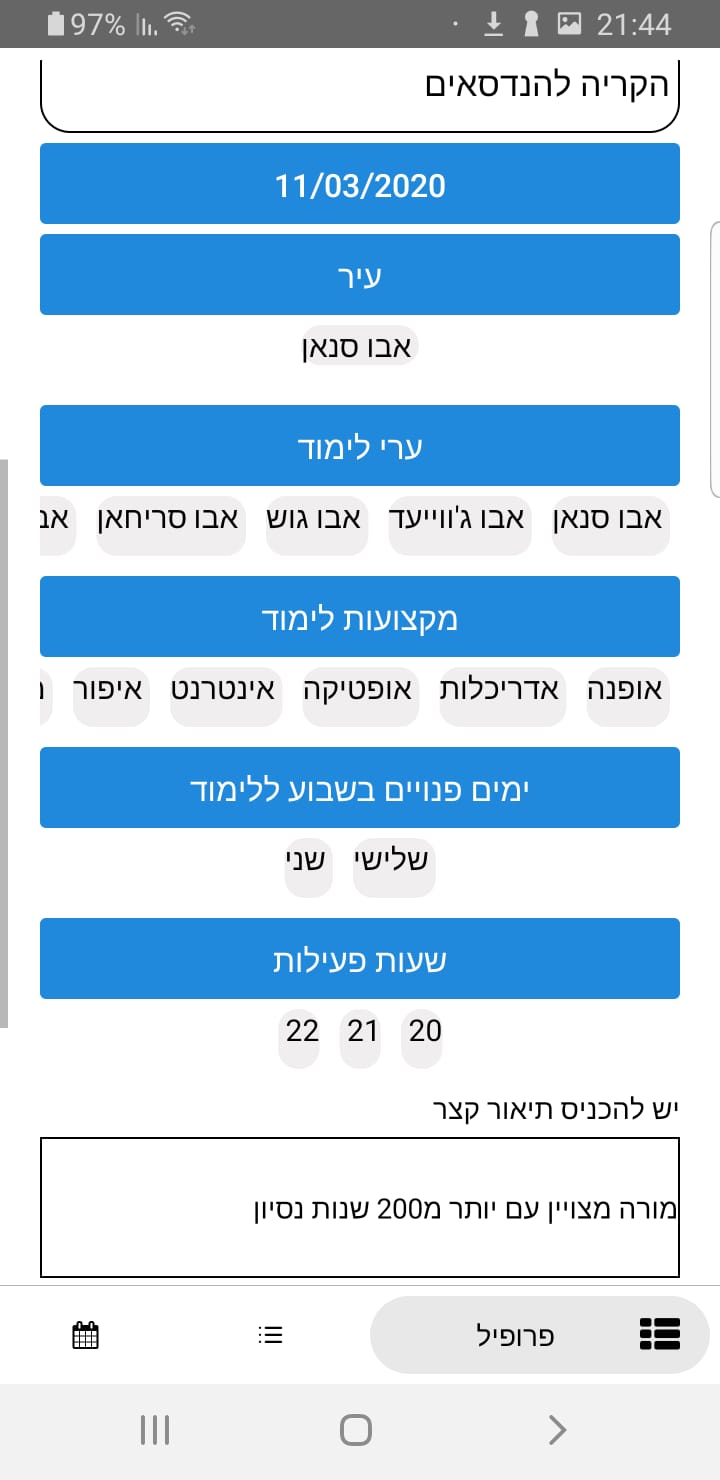
למנהל המערכת תהיה גישה לאתר על ידי חשבון גוגל שיוגדר כ'חשבון מנהל'. לאחר ההתחברות, יוכל המנהל לצפות בכמות המשתמשים הרשומים, ובסטטיסטיקות השימוש האפליקצייה.

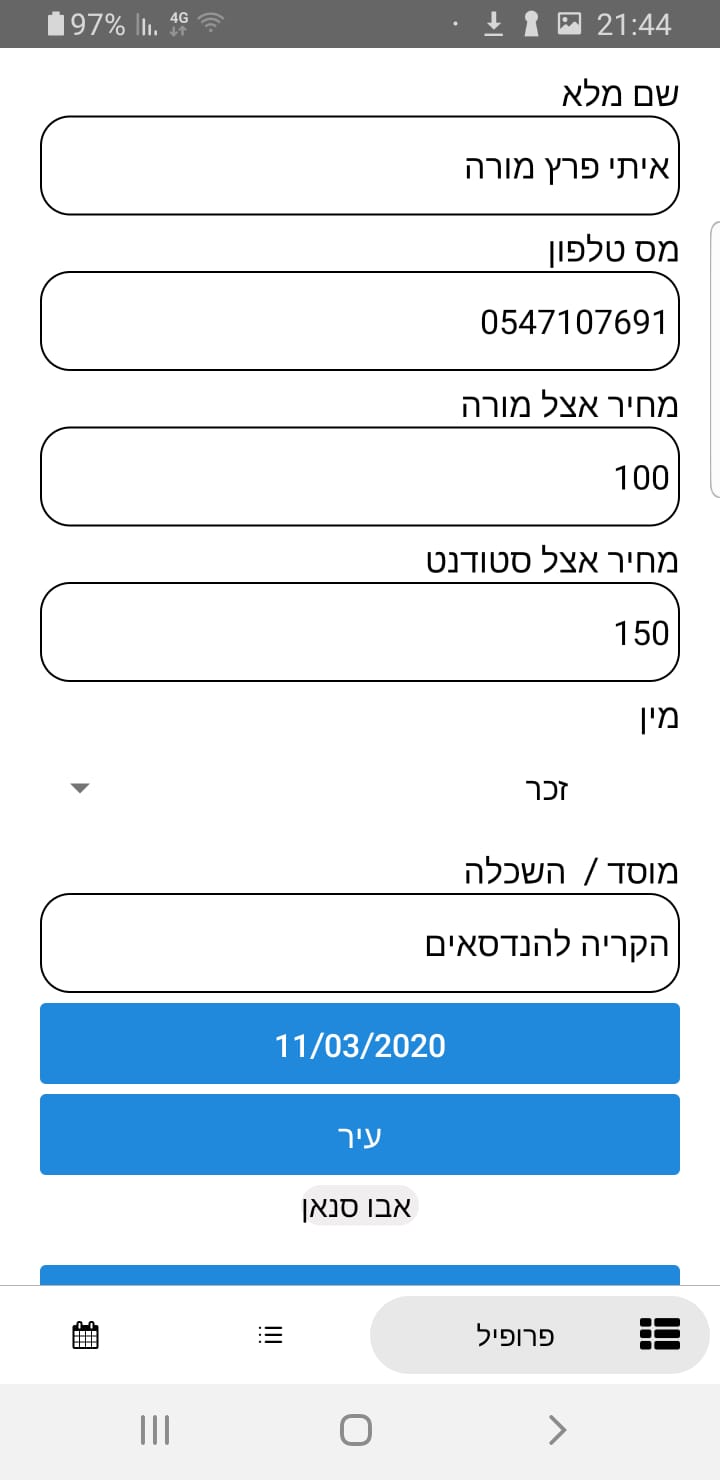
## מסכים מאופיינים:

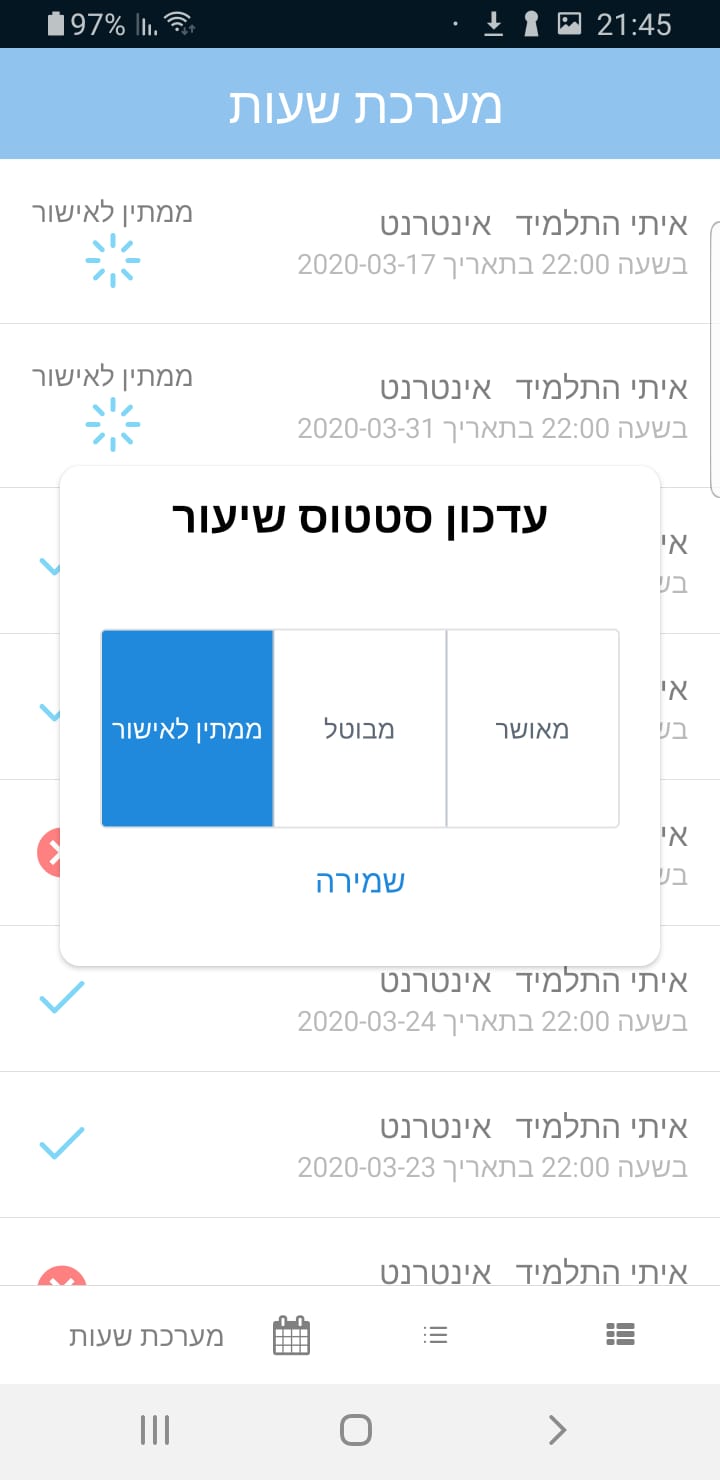
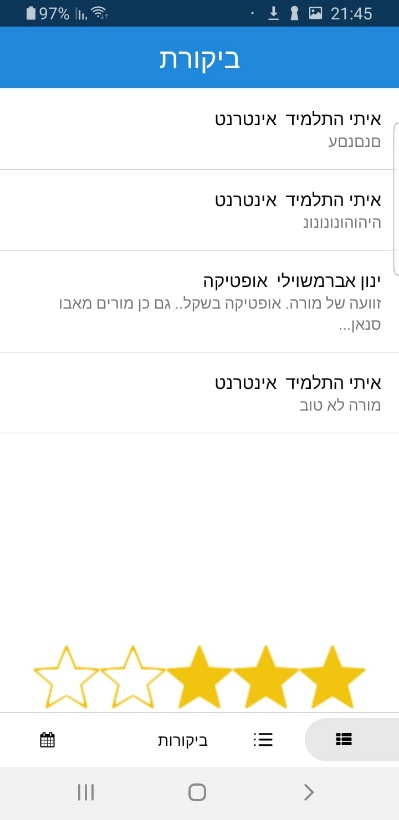
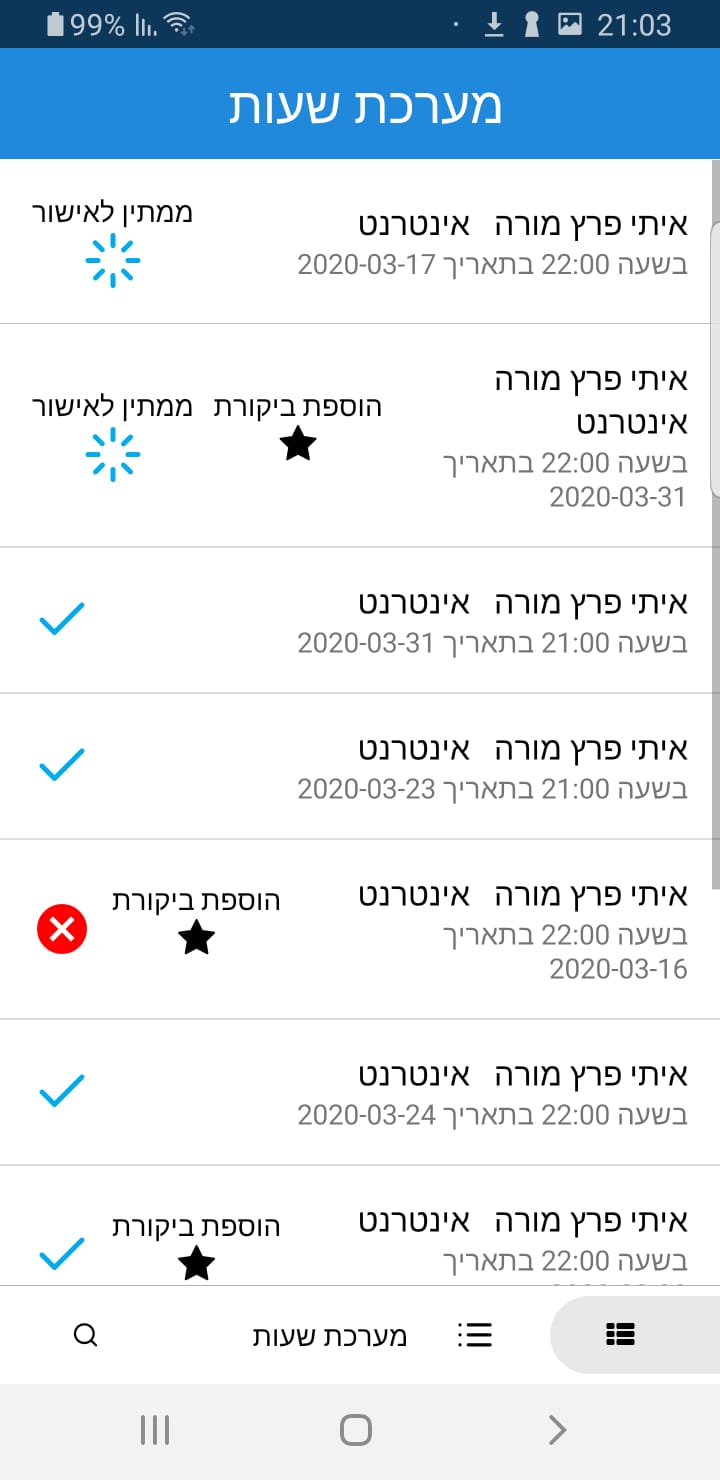
התחברות \ רישום בחירת פרופיל



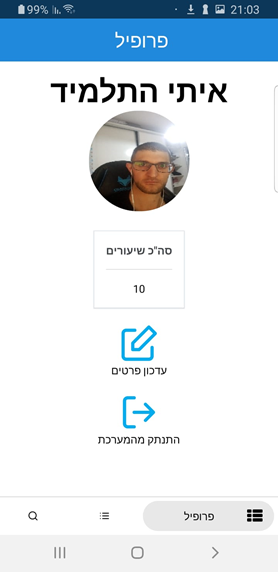
פרופיל מורה

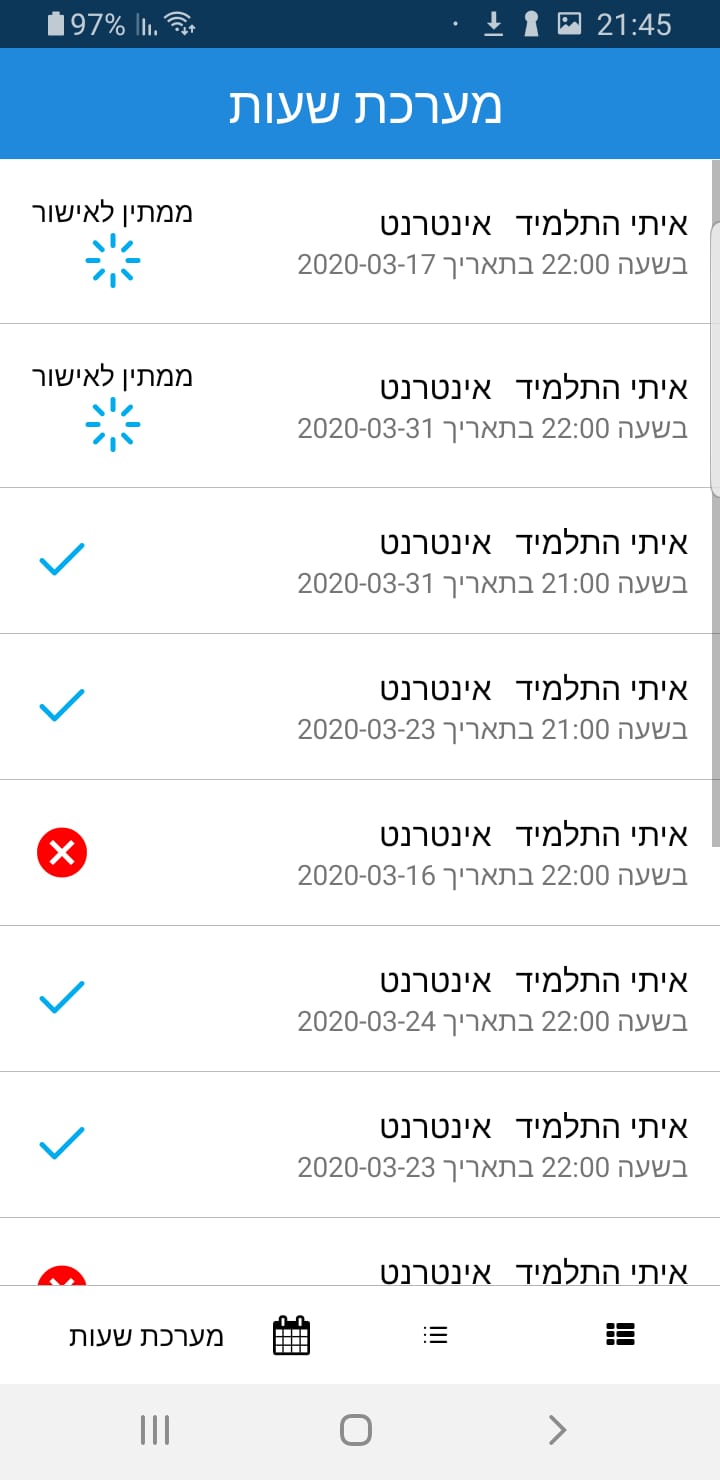
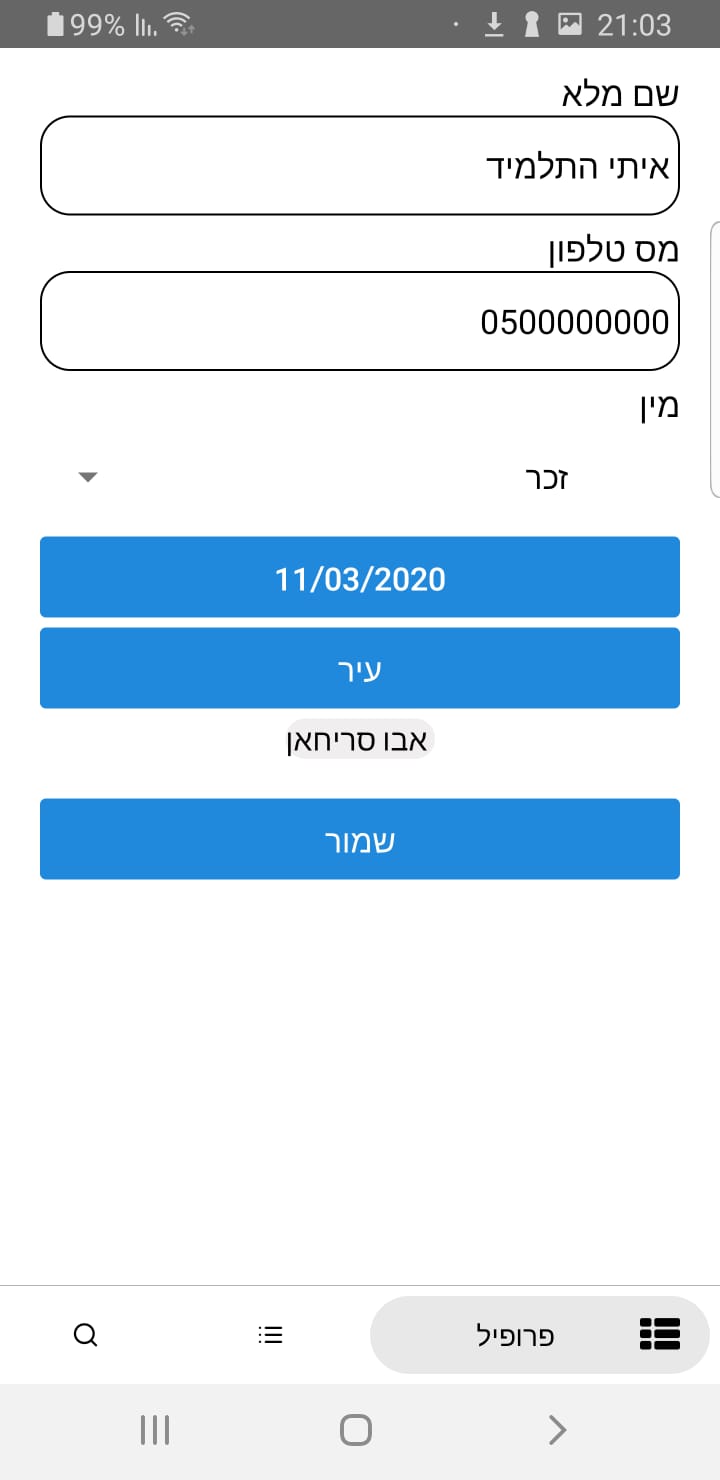


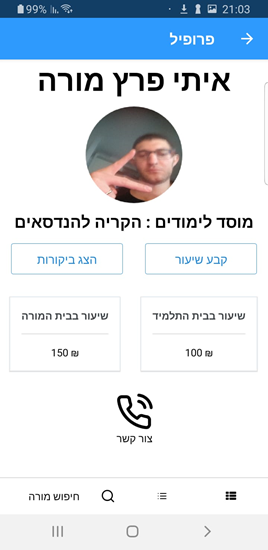
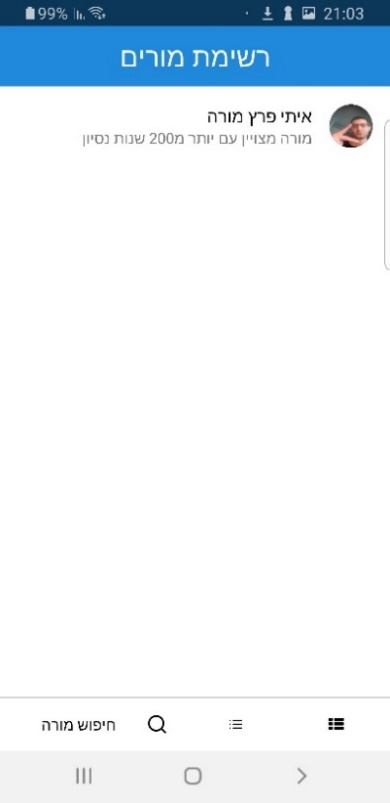


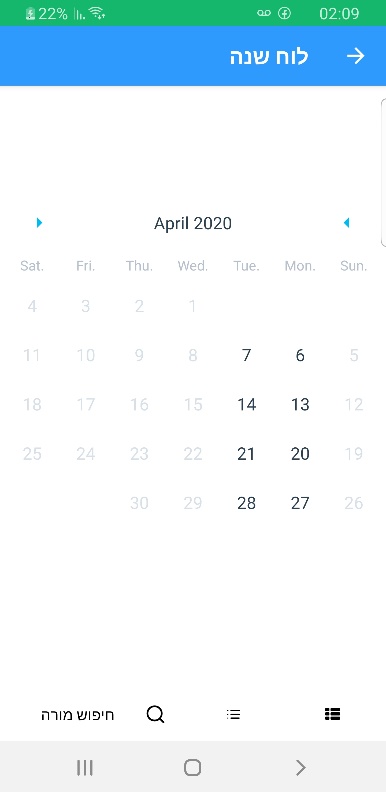
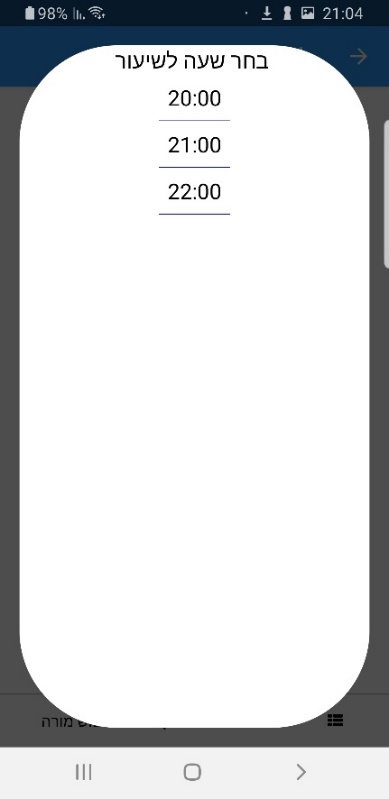
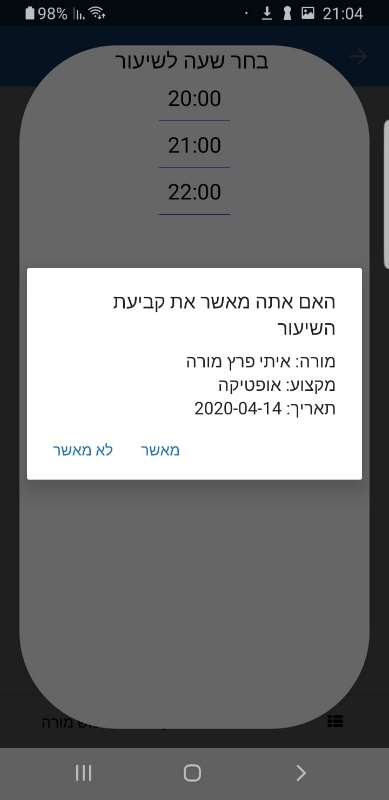


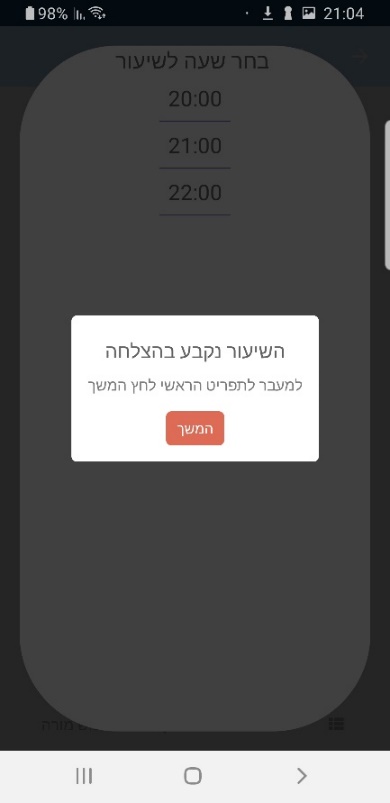
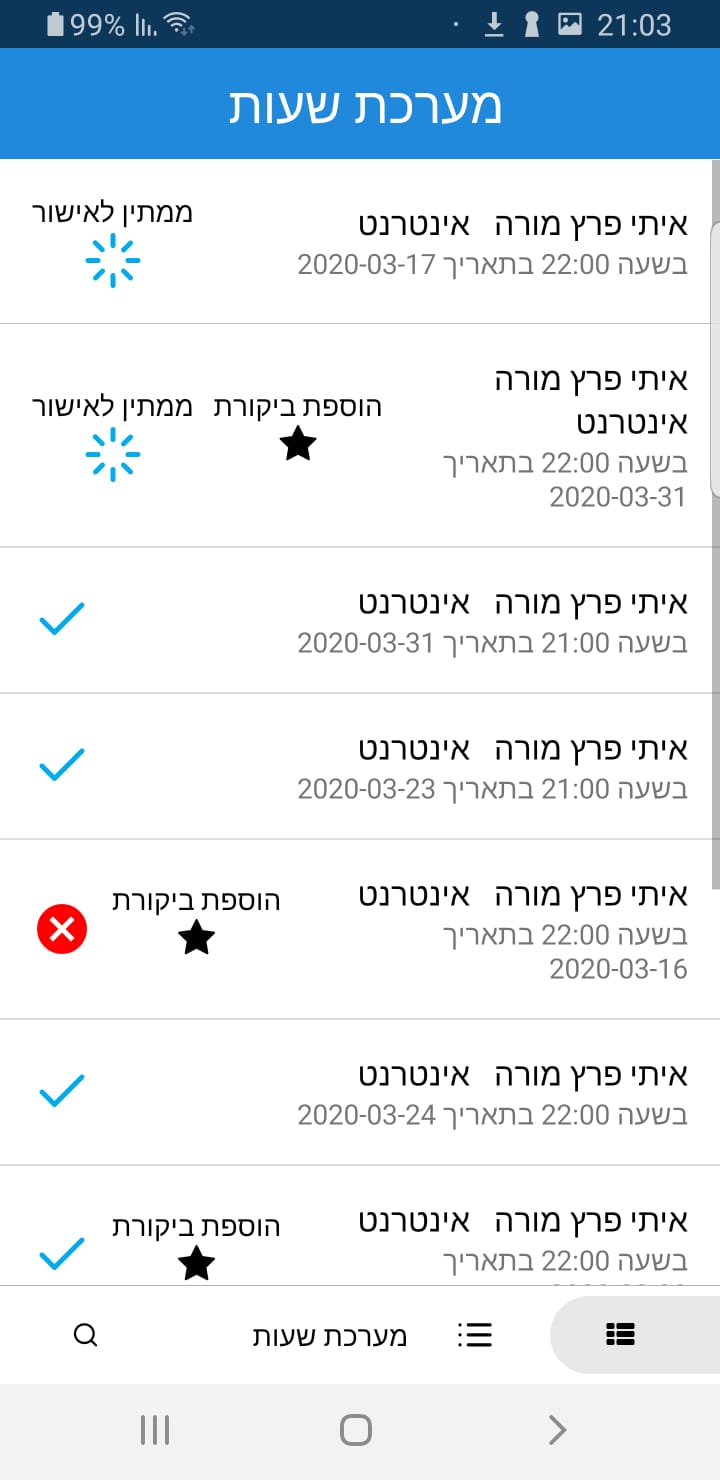
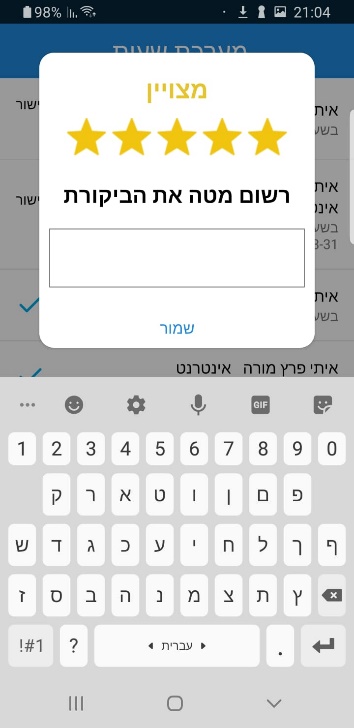
פרופיל תלמיד











# 2. מסמך דרישות SRS -

מטרת שלב הדרישות הוא יצירת תשתית למפרט התוכנה על פי צורכי הלקוח **מסיפור הלקוח לעיל**, כגון הגדרת צורכי הלקוח ו/ או בעל התוכנה. המסמך הוא מסמך פורמלי ראשוני שבו מתוארות ומפורטות הדרישות של המערכת המתוכננת. מסמך זה הינו תיאור מלא של ההתנהגות הרצויה מהמערכת שתפותח.

מבנה המסמך:

**זיהוי**– מספור של הדרישה לצרכי מעקב.

**נוסח המסמך**– תוכן הדרישה.

FR - דרישות פונקציונליות, דרישות המתארות מה המערכת צריכה לעשות ובאיזה מידע לטפל.

NFR - דרישות לא פונקציונליות, דרישות המגדירות תכונות נוספות של הפתרון שצריכות להתמלא תוך כדי מילוי הדרישות הפונקציונליות.

**מאפיינים נוספים**– מידת חשיבות הדרישה.

UC – ה-use case שאליו מתקשרת הדרישה, והתרחיש שיתאר אותה.

מקרא קיצורים:

Functional Requirements – FR – דרישות פונקציונליות:

Operation Requirements – OR – דרישות תפעוליות המתייחסות לתפעול, אינטגרציה והתנהגות צפויה של המערכת.

Data Requirements – DR – דרישות מידע המתייחסות לנתונים בהם נדרשת המערכת לטפל.

 :דרישות לא פונקציונליות – Non Functional Requirements – NFR

Performance Requirements – PR – דרישות ביצועים מדידות מהמערכת.

Quality Attributes – QA – מאפייני איכות של המערכת כגון: אמינות, זמינות, בטיחות, ביטחון ושימושיות.

Hardware Requirements – HC – אילוץ חומרה, מבנה החומרים של רכיבים.

Implementation Constraint – IC – אילוץ מימוש, אופן מימוש ספציפי בתוכנה כגון: אלגוריתם, מבנה נתונים מסוים, אופן הפעלה ספציפי וכדו.'

Management Constraint – MC – אילוץ ניהולי, תנאים בהם צריך לעמוד כגון: תקציב, לו"ז, זמינות משאבים וכדו'.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| זיהוי | נוסח | FR | NFR | UC |
| 1 | בכניסה לאפליקציה יעלה לוגו של האפליקציה למסך למספר שניות ולאחריו יפתח המסך הראשי. | OR/DR |  |  |
| 2 | במסך הראשי תופיע האפשרות להתחבר לחשבון קיים. | OR/DR |  | 1 |
| 3 | ההתחברות תתאפשר באמצעות חשבון GOOGLE. | DR | IC | 1 |
| 4 | במסך הראשי תופיע האפשרות להירשם לראשונה לאפליקציה. | OR/DR |  | 2 |
| 5 | הרישום יתאפשר באמצעות חשבון GOOGLE. | DR | IC | 2 |
| 6 | לחיצה על ה-API של GOOGLE תעביר את המשתמש לחשבון שלו. | OR | IC | 1 |
| 7 | התחברות עם חשבון גוגל לא מוכר תעביר את המשתמש למסך הרשמה. | OR |  | 2 |
| 8 | במסך ההרשמה יבחר המשתמש את סוג הפרופיל אותו מעוניין לפתוח – מורה או תלמיד. | DR |  | 2 |
| 9 | לאחר לחיצה על מורה \ תלמיד, המשתמש יעבור למסך הרישום. | DR |  | 2 |
| 10 | במסך הרישום יופיעו הנתונים אותם יידרש המשתמש למלא. כגון: מס' טלפון, כתובת, מקצוע וכו'. | OR\DR | IC | 2 |
| 11 | בסיום הליך ההרשמה, תופיע הודעה למשתמש כי הוא נרשם בהצלחה. | OR | IC |  |
| 12 | בסיום הרישום המשתמש יקבל מייל המברך אותו על הצטרפותו. | OR/DR |  |  |
| 13 | לאחר התחברות עם חשבון מורה - יפתח תפריט מורה. | OR/DR |  |  |
| 14 | מסך הבית של המורה יכיל סרגל המכיל אפשרות לעבור למסכים הבאים: מערכת שעות, פרופיל, ביקורות - על ידי לחיצה על האייקון המתאים. | DR/OR |  |  |
| 15 | בכניסה למסך מערכת שעות יפתח מסך מערכת השעות. | OR/DR |  | 5 |
| 16 | בכניסה למערכת שעות תוצג למשתמש רשימת השיעורים שנקבעו לו אי פעם. | DR |  | 5 |
| 17 | במסך מערכת שעות המורה יוכל לבטל את השיעור. | OR/DR |  | 6 |
| 18 | בעת ביטול השיעור התלמיד יקבל הודעה על כך. | OR | QA | 6 |
| 19 | בכניסה למסך הביקורות יפתח מסך הביקורות. | OR/DR |  | 9 |
| 20 | במסך הביקורות יוצג למורה הדירוג הממוצע אותו קיבל מהתלמידים. | DR |  | 9 |
| 21 | במסך הביקורות המורה יראה את הביקורות שרשמו עליו התלמידים. | DR |  | 9 |
| 22 | בכניסה למסך הפרופיל יפתח מסך הפרופיל. | OR/DR |  | 3 |
| 23 | במסך הפרופיל יוצגו למורה כל הפרטיים אותם הזין על עצמו. | DR |  | 3 |
| 24 | במסך הפרופיל יתאפשר לעדכן את הפרטים האישיים. | OR |  | 4 |
| 25 | במסך הפרופיל יוכל המורה לעדכן את האזורים בהם הוא מלמד. | OR |  | 4 |
| 26 | במסך הפרופיל תהיה האפשרות להגדיר את ימי ושעות הפעילות בהם הוא זמין. | OR |  | 7 |
| 27 | במסך הפרופיל יוכל המורה לעדכן את המקצועות אותם הוא מלמד. | OR |  | 4 |
| 28 | במסך הפרופיל יוכל המורה לראות ולעדכן את מחיר השיעור. | OR |  | 4 |
| 29 | לאחר התחברות עם חשבון תלמיד - יפתח תפריט תלמיד. | OR/DR |  |  |
| 30 | מסך הבית של התלמיד יכיל סרגל המכיל אפשרות לעבור למסכים הבאים: מערכת שעות, פרופיל, חיפוש מורה, על ידי לחיצה על האייקון המתאים. | OR/DR | IC |  |
| 31 | בכניסה למסך פרופיל יפתח מסך הפרופיל. | OR/DR |  | 10 |
| 32 | במסך הפרופיל יוצגו לתלמיד כל הפרטים אותם הזין על עצמו. | DR |  | 10 |
| 33 | במסך הפרופיל יוכל התלמיד לעדכן את פרטיו האישיים. | OR |  | 4 |
| 34 | בכניסה למסך מערכת שעות יפתח מסך מערכת השעות. | OR/DR |  | 5 |
| 35 | במסך מערכת שעות יראה התלמיד את כל השיעורים אותם קבע אי פעם. | DR |  | 5 |
| 36 | במסך מערכת שעות יוכל התלמיד לבטל את השיעור. | OR/DR |  | 7 |
| 37 | בעת ביטול השיעור המורה יקבל הודעה על כך. | OR | QA | 7 |
| 38 | בכניסה למסך חיפוש מורה יפתח מסך חיפוש מורה. | OR/DR |  | 11 |
| 39 | בכניסה למסך חיפוש מורים יוכל התלמיד לחפש מורה לפי קריטריונים. | OR/DR |  | 11 |
| 40 | במסך חיפוש מורה התלמיד יבחר את המקצוע באמצעות רשימת מקצועות. | OR | IC | 11 |
| 41 | במסך חיפוש מורה התלמיד יגדיר את האזורים מהם רוצה לקבל תוצאות. | OR | IC | 11 |
| 42 | לאחר הזנת הנתונים ולחיצה על חיפוש מורה תופיע במסך רשימת המורים המתאימים לקריטריונים. | OR/DR |  | 11 |
| 43 | בעת בחירת המורה יוצג פרופיל המורה, הכולל ציון וביקורות. | OR/DR |  |  |
| 44 | בעת לחיצה על 'ביקורות' יוצגו הביקורות אותם קיבל המורה. | OR/DR |  |  |
| 45 | במסך בחירת המורה התלמיד יקבע את השיעור לפי רשימת השעות והתאריכים שבהם המורה פנוי לשיעור. | OR/DR | IC | 12 |
| 46 | בעת קביעת שיעור על ידי לחיצה על מקש 'קבע שיעור' תשלח למורה הודעה על בקשה לאישור שיעור. | OR/DR |  | 12 |
| 47 | לאחר קביעת שיעור תופיע הודעה למסך כי השיעור נקבע בהצלחה. | OR/DR |  |  |
| 48 | בכל שלב באפליקציה ניתן לגרור את המסך כלפי מטה לריענון. | DR |  |  |
| 49 | למנהל המערכת תהיה גישה לדף הניהול דרך דף WEB. | DR |  |  |
| 50 | בכניסה לאתר הניהול יידרש המנהל להתחבר עם חשבון GOOGLE המוגדר כ'חשבון מנהל'. | OR/DR | IC |  |
| 51 | עם הזנת הפרטים הנכונים יפתח מסך הניהול. | OR/DR |  |  |
| 52 | במסך הניהול יופיעו כמות המשתמשים הרשומים, תלמידים ומורים. | DR |  |  |
| 53 | במסך הניהול יופיעו סוגי סטטיסטיקות אותם ניתן יהיה לראות. | DR |  |  |

# Activity Diagrams – ותרשימי פעילות Use Cases – תרחישי שימוש .3

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 ב-MSS, רצף זה יביא לסיום **לא** מוצלח של התרחיש. | הסתעפות חריגה |
| דרישות המוכללות בתרחיש השימוש לפי מסמך ה SRS למעלה. | עקיבה לדרישות (זיהוי) |

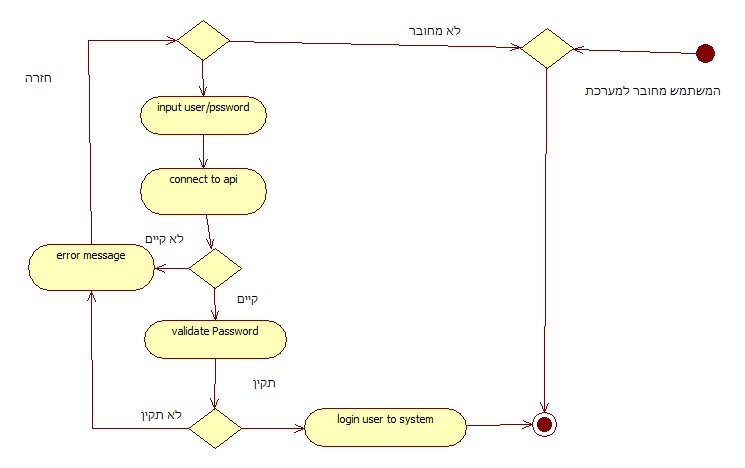
## תרשימי פעילות - Activity diagrams

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.

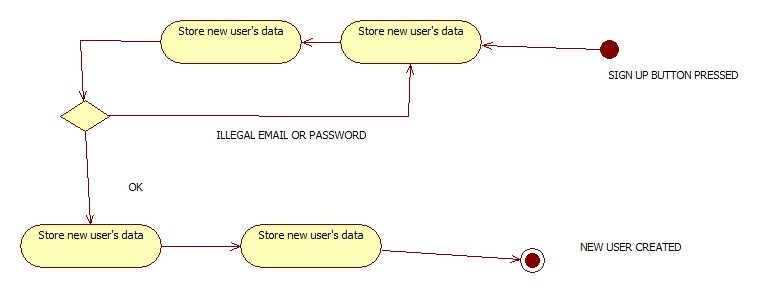
המעבר בין הפעילויות מתואר באמצעות קווים עם ראש חץ. תנאי הסתעפות או כניסה ללולאה מתואר באמצעות מעוין, שבו נרשם התנאי שיש לבדוק בכניסה להסתעפות, או תנאי יציאה מלולאה. בשונה מתרחיש זרימה ההתחלה של תרשים פעילות מסומנת בעיגול מושחר קטן, וסיום של פעילות בעיגול מושחר כפול.

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

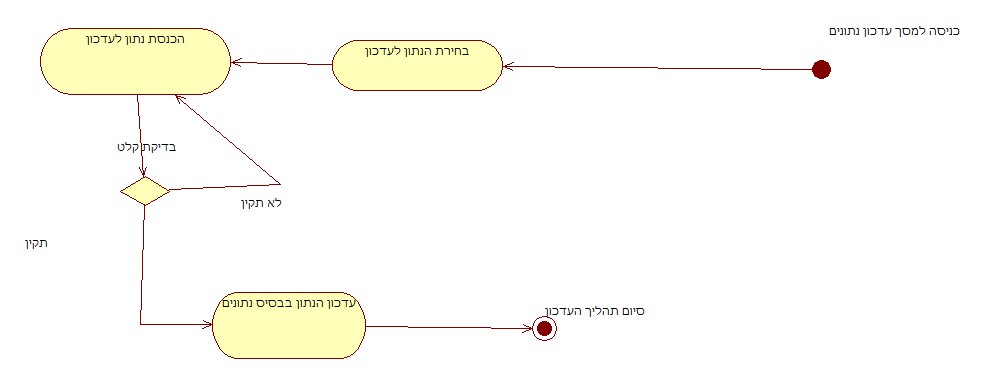
|  |  |
| --- | --- |
| LOGIN | UC-1 |
| משתמש: התחברות למערכת. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * קיים חיבור תקין לאינטרנט. * קיים חשבון פעיל למשתמש. * המשתמש אינו מחובר למערכת. | Pre-Conditions |
| המשתמש מחובר למערכת. | Post-Conditions |
| כניסה לאפליקציה. | Trigger |
| 1. המשתמש מתחבר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



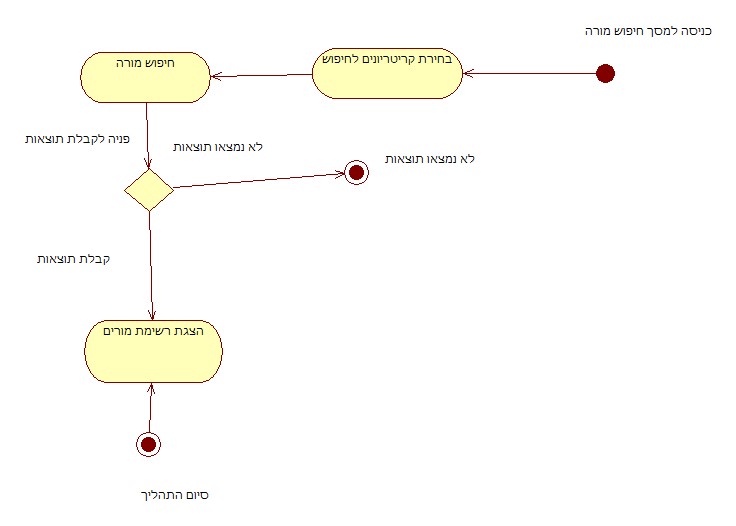
|  |  |
| --- | --- |
| SIGN UP | UC-2 |
| משתמש: התחברות ראשונית למערכת. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * קיים חיבור תקין לאינטרנט. * קיים חשבון google פעיל למשתמש. | Pre-Conditions |
| * נוצר חשבון חדש למשתמש. * המשתמש מחובר למערכת. | Post-Conditions |
| לחיצה על כפתור SIGN UP בדף ההתחברות . | 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) |



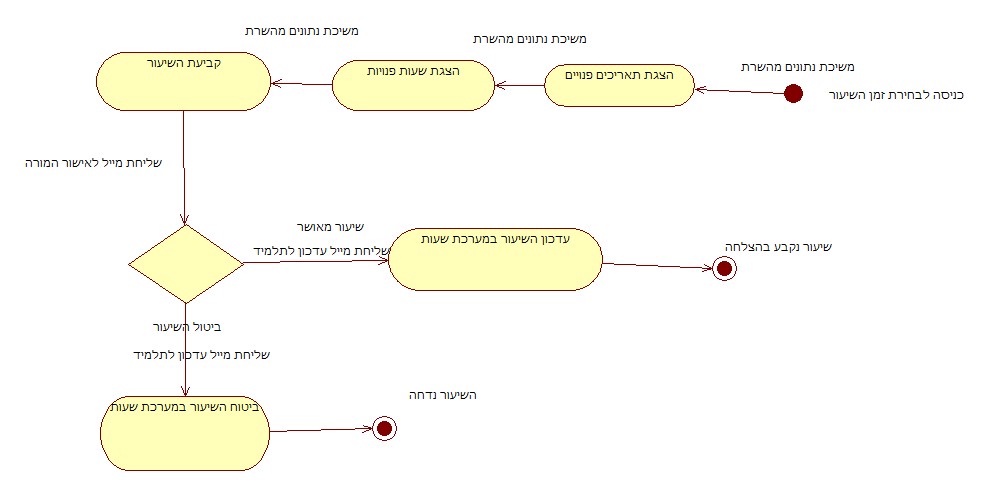
|  |  |
| --- | --- |
| עדכון נתונים | UC-3 |
| משתמש: עדכון נתונים. | שחקנים ויעדים |
| אין | ב"ע ואינטרסים |
| * התחברות למערכת. * חיבור תקין לאינטרנט. | Pre-Conditions |
| הנותנים יתעדכנו. | Post-Conditions |
| לחיצה על כפתור עדכן נתונים. | Trigger |
| 1. המשתמש בוחר את הנתון שהוא רוצה לעדכן 2. המשתמש מעדכן את הנתון . 3. המשתמש לוחץ על "לעדכון לחץ כאן". | MSS |
| חריגה- מצעד 1 של ה-MSS: הוכנס נתון לא תקין (כגון תווים במס' טלפון).  1א1- מוקפצת הודעה למשתמש: "יש להשתמש רק בספרות". | הסתעפות א' |
| 10, 28-31, 36-39 | 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 |
| * בחירת הקריטריונים לחיפוש כגון עיר, מקצוע. * 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. | MSS |
| חריגה- מצעד 23של ה-MSS:המורה דוחה את השיעור.  1א1. המערכת תשלח מייל לתלמיד עם עדכון על ביטול השיעור. | הסתעפויות |
| 10, 17, 56, 57 | עקיבה לדרישות (זיהוי) |



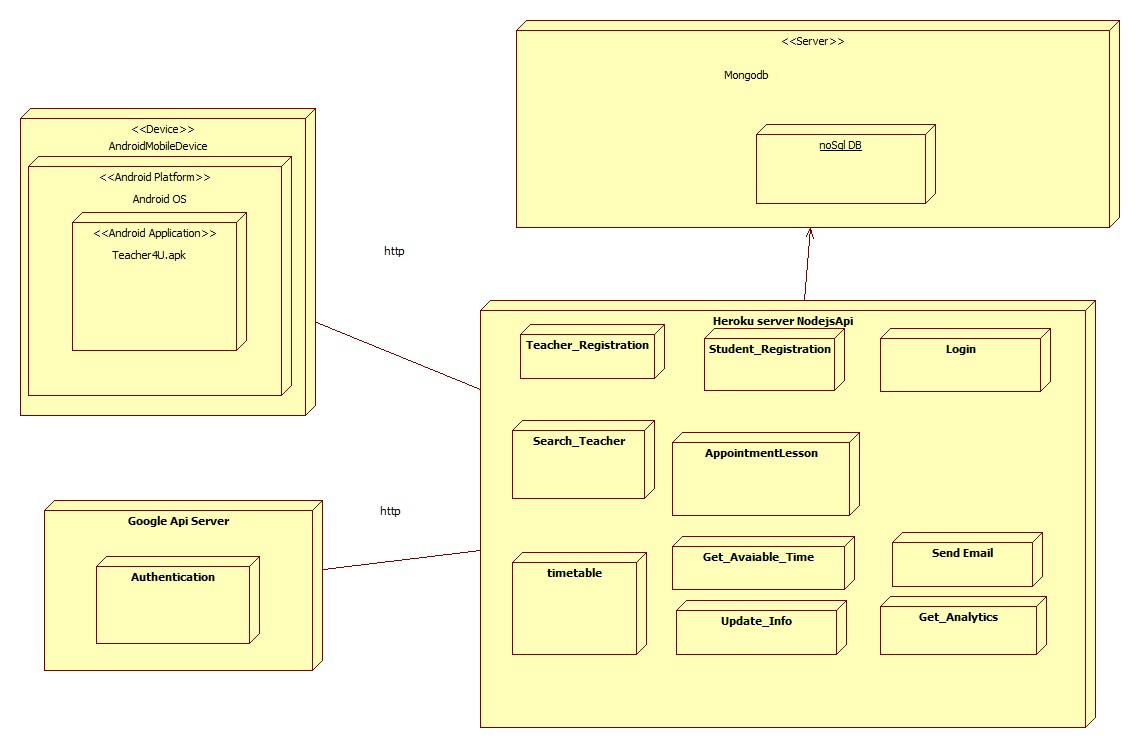
# מסמכי SAD

# 4. ארכיטקטורה פיזית

בחלק זה מובאים 4 תרשימים המתארים את המבנה הפיזי של מערכת Teacher4U.

## .1 תרשים פריסה Deployment Diagram))

תיאור הרכיבים הפיזיים במערכת כ-Nodes



## 2. 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 תהיה דרך האפליקציה ע"י פונקציות מובנות. |  |
| MongoDB atlas | שרת המכיל את מסד הנתונים של המערכת. |  |
| Computer with browser | מכשיר המריץ את הדפדפן דרכו יתחברו לממשק מנהל. |  |

## 3. 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 ה-API. |  |
| MongoDB atlas DB | בסיס נתונים מבוסס NoSQL. |  |
| Web Browser | הדפדפן דרכו ההתחברות לממשק המנהל תתבצע**.** |  |

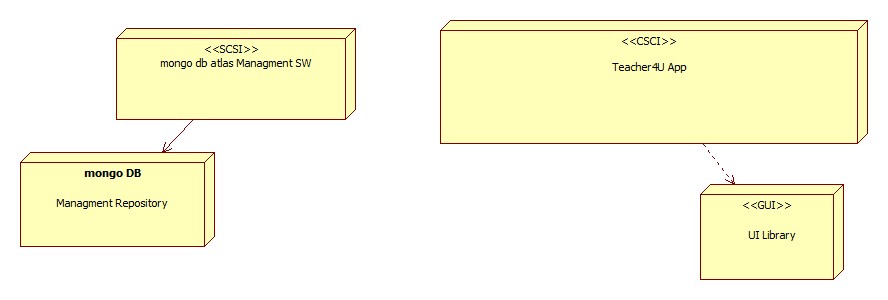
## 4. ממשקים פיזיים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |  |

# 5. ארכיטקטורה לוגית

בחלק זה מובאים 2 תרשימים המתארים את המבנה הלוגי של מערכת Teacher4U.

## 1. דיאגרמת חלוקה לרכיבי תוכנה (Software Components)



## 2. 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 |

# 6. 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.

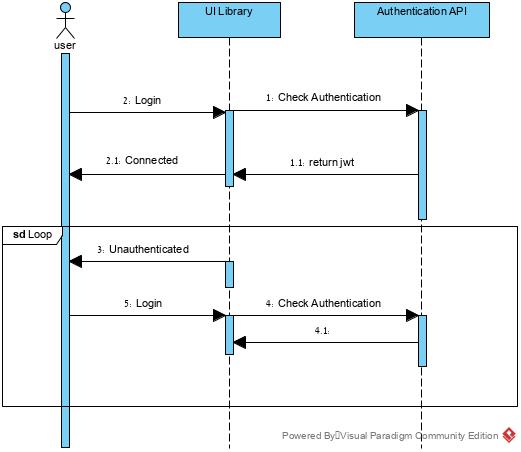
תרשים הרצף נועד לתאר תהליך ולהסביר כיצד המערכת ורכיביה מבצעים תהליך זה.

הסבר מבנה התרשים:

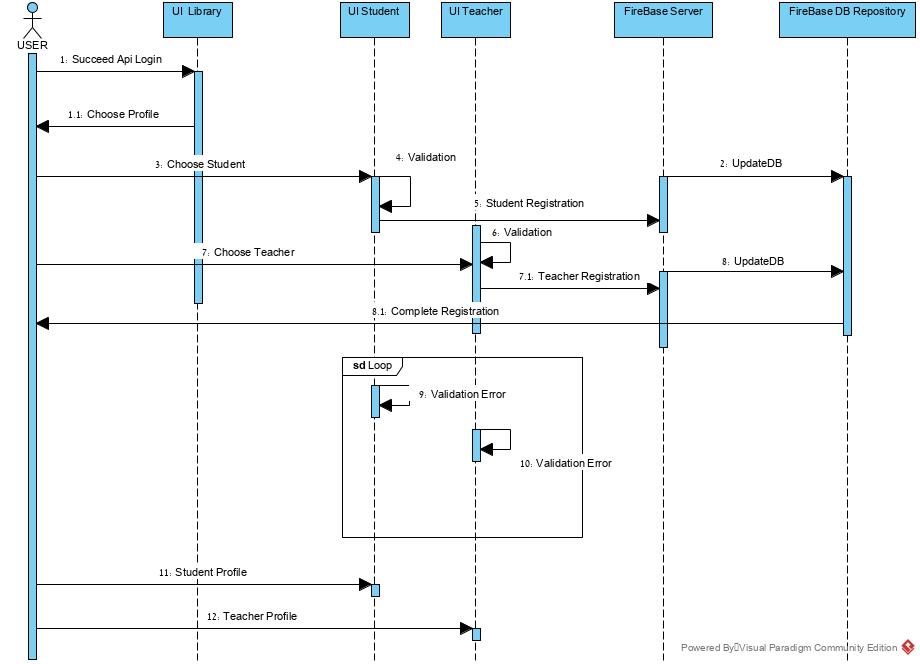
ציר הגובה המקווקו מתאר את קו החיים של האובייקט בזמן ביצוע התרחיש מלמעלה למטה. המלבנים שיופיעו על הקווים המקווקוים יתארו התממשקות של הרכיב בנקודת זמן ולפרק זמן מסויים. במשך זמן זה הרכיב במוקד הבקרה של המערכת.

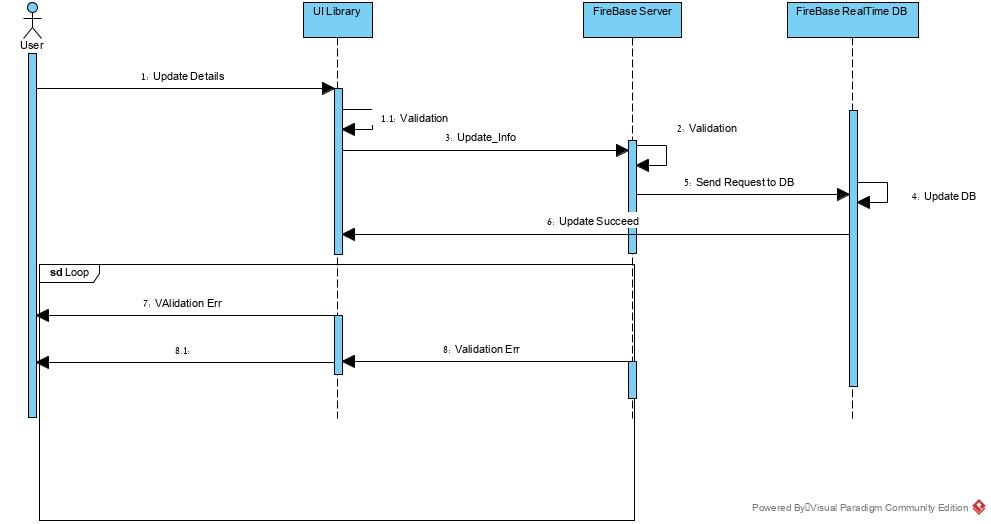
חץ עם קו מלא מתאר קריאה לרכיב או שליחת הודעה בין רכיבים, וחץ עם קו מקווקו מתאר חזרה מרכיב. בחלק מהתרשימים יופיעו גם מלבנים שיתארו אילוצי פעולה מסוימת כפי שהם מופיעים בתרחיש המימוש.

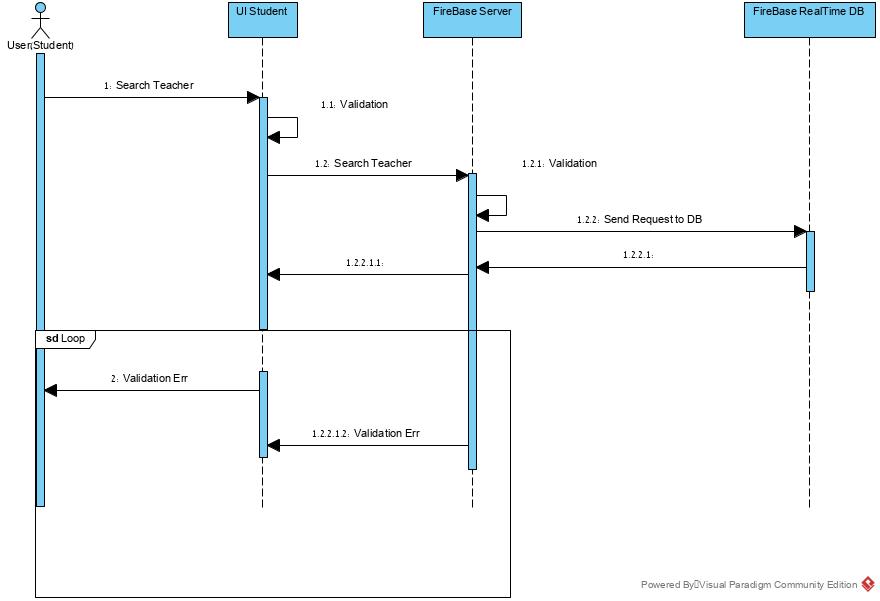
* SUC-1 LOGIN



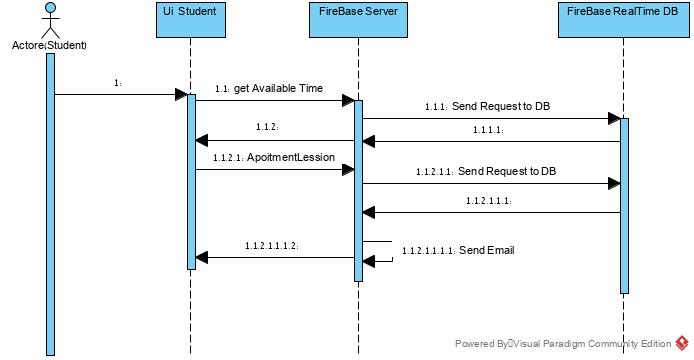
* SUC2-SIGN UP



* UC3- Update Information
* UC4-TeacherSearch



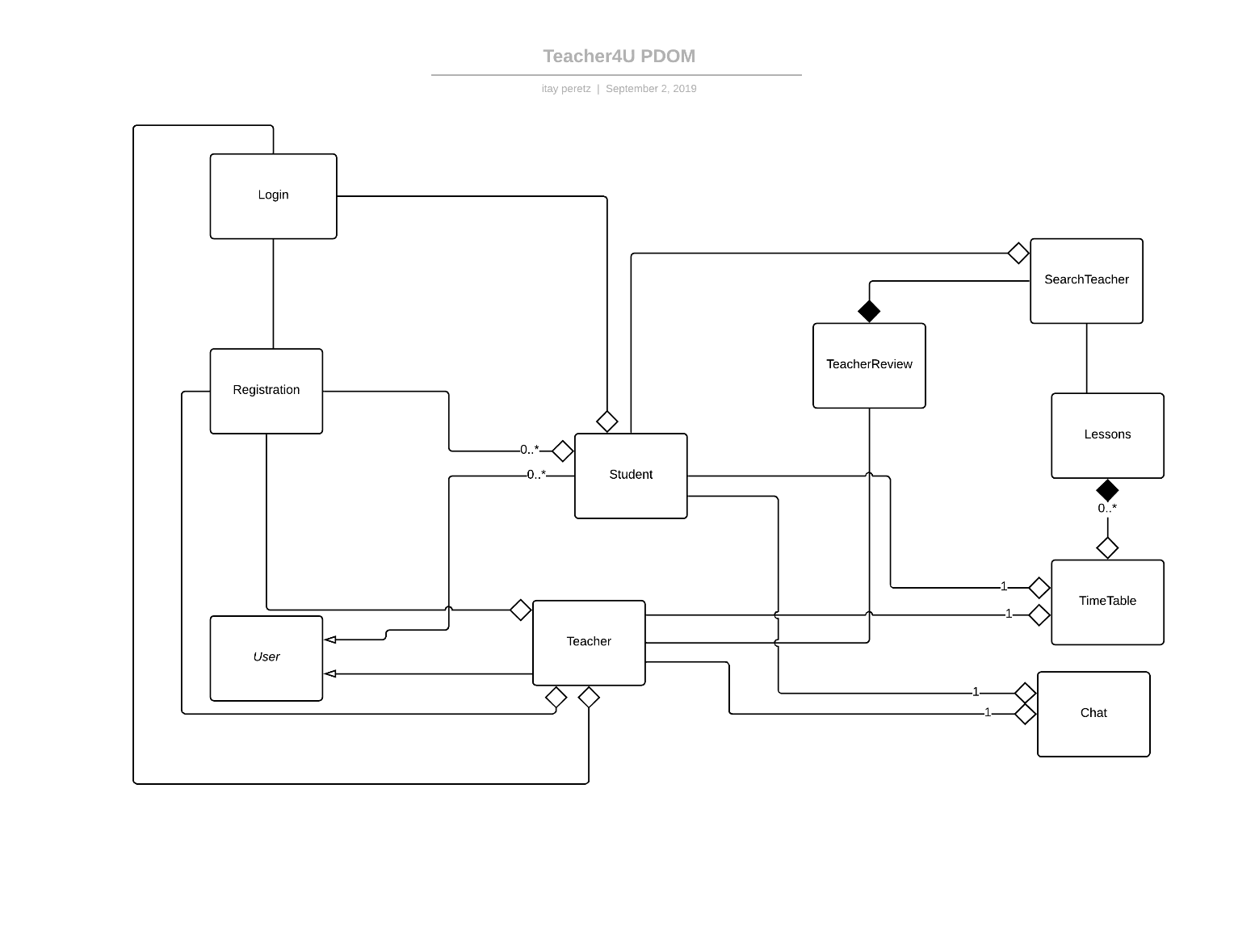
* UC5-Schedule Lesson



# מסמכי SDD

# 7. מודל ישויות במרחב הבעיה(PDOM)

להלן תרשים המתאר את תפקוד רכיבי המערכת והקשרים הלוגיים ביניהם.



# 8. 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

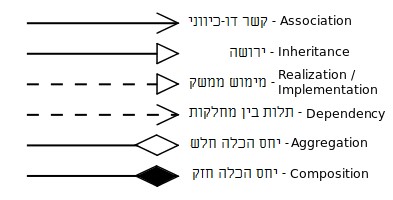
בכל צד של היחס **Inheritance:** ירושה ממחלקת בסיס.

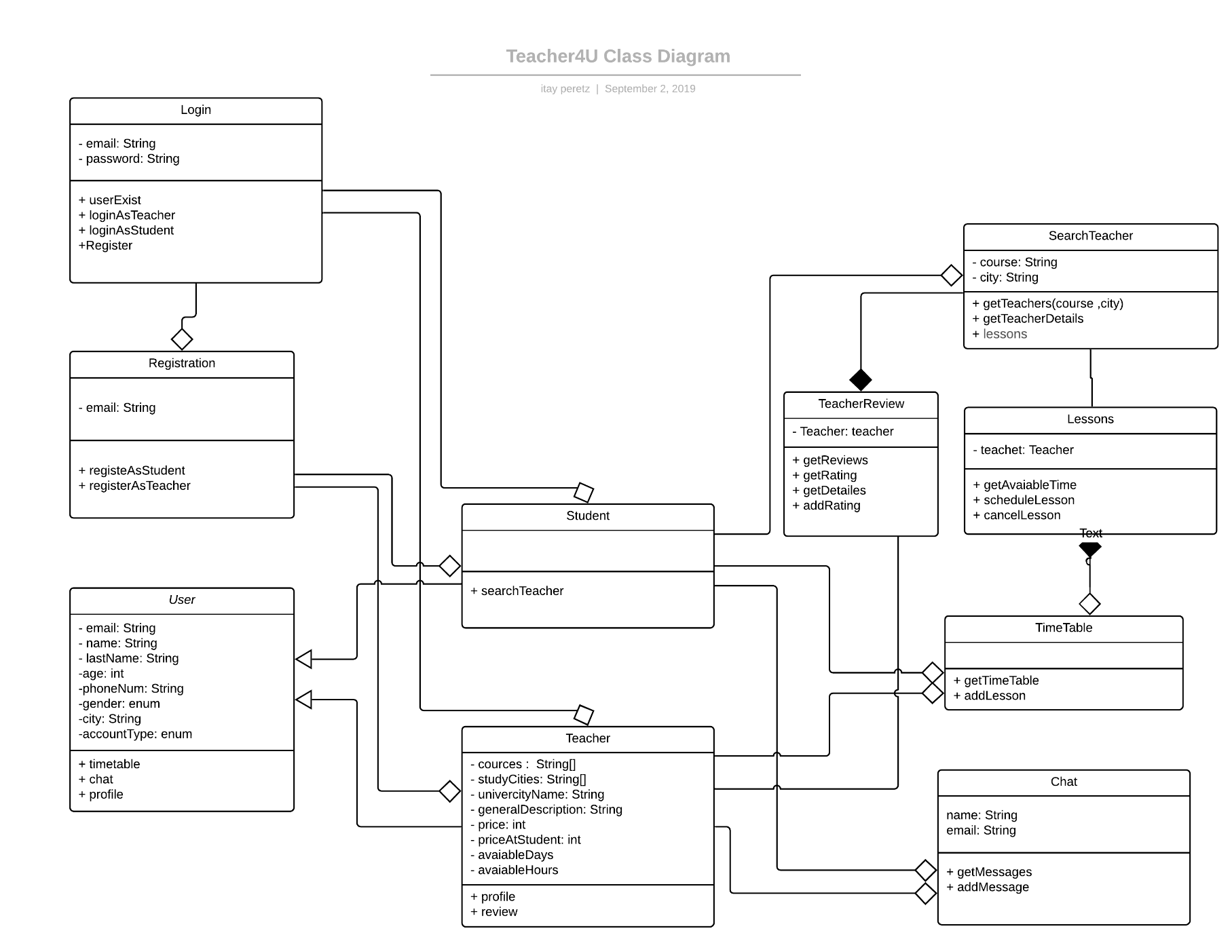
**Implementation:** מימוש ממשק. **Dependency:** תלות של מחלקה אחת במחלקה אחרת.

**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:





# 9. תרשימי רצף ברמת עצמים.

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

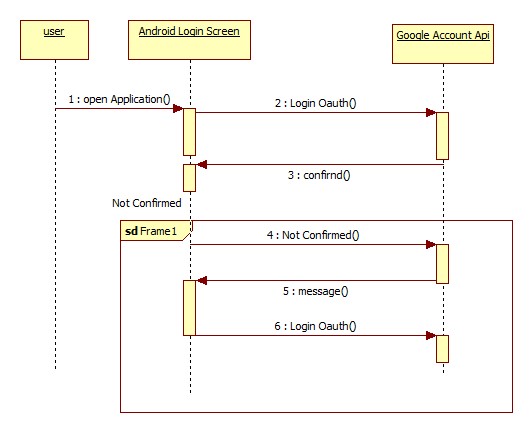
תרשים זה מספק את המבט הדינמי על ריצת התרחיש – באילו עצמים נעשה שימוש ,באיזו דרך, על ידי מי ומתי כל זה קורה על ציר הזמן.

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

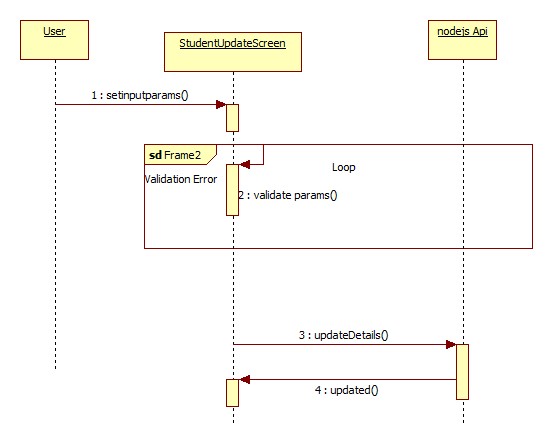
הסבר מבנה התרשים

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.



## SUC-1 Log-In

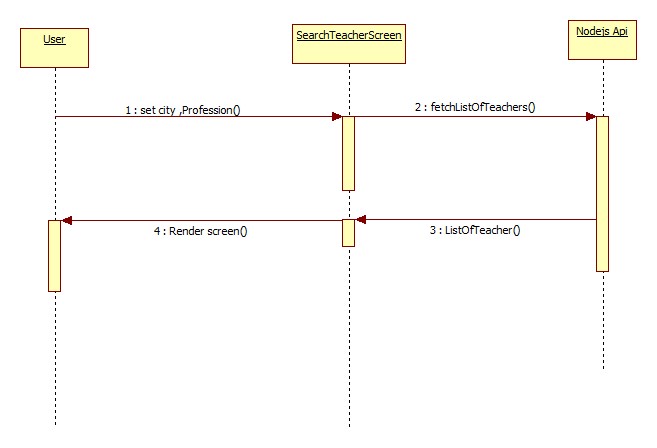


## SUC-3

## Update personal

## Info

SUC-4 קביעת שיעור



# 10. מכונת מצבים (State Charts)

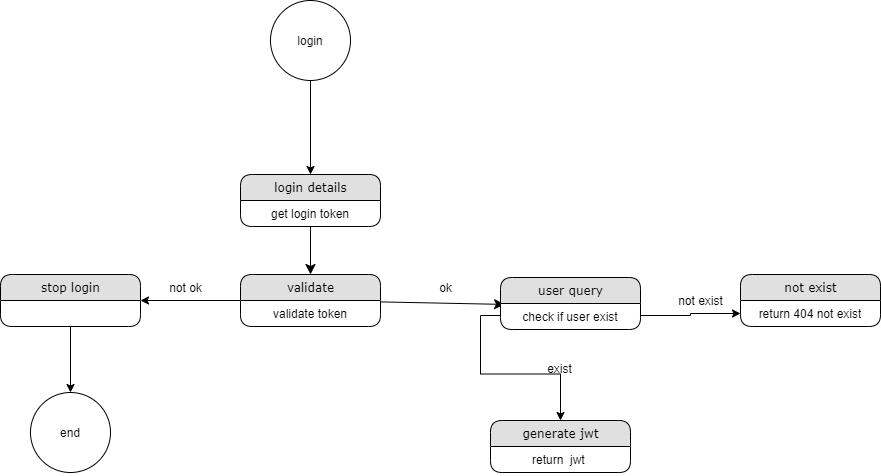
תרשים "מכונת מצבים" כדלהלן מציג את המצבים השונים של אובייקט ואת השפעת האירועים השונים על המצבים לאורך חייו של האובייקט.

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

# 11. בדיקות Test Cases – מקרי בדיקה

## 1 - הרשמה ראשונית – מורה ותלמיד

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 1.1 | הרשמה ראשונית לאפליקציה – הזנת כל הנתונים בצורה תקינה. | 1. התחברות ל-API של GOOGLE. 2. בחירת סוג הפרופיל – מורה או תלמיד. 3. הזנת כל השדות בצורה תקינה. 4. לחיצה על שמירת נתונים. | 1. התחברות תקינה לאפליקציה. 2. מעבר לדף הפרופיל. 3. קבלת מייל "ברוך הבא". | 12.04.20 | תקין |
| 1.2 | הרשמות ראשונית לאפליקציה – לא כל הנתונים תקינים. | 1. התחברות ל-API של GOOGLE. 2. בחירת סוג הפרופיל – מורה או תלמיד. 3. הזנת חלק מהשדות בצורה לא תקינה. 4. לחיצה על שמירת נתונים. | קבלת הודעה עבור כל שדה בנפרד על נתונים לא תקינים – לא ניתן להמשיך הלאה. | 12.04.20 | תקין |
| 1.3 | הרשמות ראשונית לאפליקציה – לא כל הנתונים מולאו. | 1. התחברות ל-API של GOOGLE. 2. בחירת סוג הפרופיל – מורה או תלמיד. 3. הזנת חלק השדות בלבד. 4. לחיצה על שמירת נתונים. | קבלת הודעה כי יש למלא את כל השדות – לא ניתן להמשיך הלאה. | 12.04.20 | תקין |

## 2 - משתמש קיים – מורה ותלמיד

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 2.1 | כניסה לאפליקציה - עם משתמש קיים שאינו מחובר. | 1. התחברות ל-API של GOOGLE. 2. מעבר מידי לדף הפרופיל | מעבר מידי לדף הפרופיל. | 12.04.20 | תקין |
| 2.2 | התחברות לאפליקציה – עם משתמש קיים מחובר. | כניסה לאפליקציה. | מעבר מידי לדף הפרופיל. | 12.04.20 | תקין |

## 3 - שינוי תמונת פרופיל – מורה ותלמיד

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 3.1 | הגדרת/החלפת תמונת פרופיל – תמונה קיימת. | 1. המשתמש לחוץ על תמונת הפרופיל במסך הפרופיל. 2. המשתמש בוחר את אפשרות הגלריה. 3. המשתמש מאתר את התמונה דרך אפליקציית סייר הקבצים ובוחר בה. | תמונת הפרופיל תוחלף ותעלה לאתר אחסון שישלח קישור שישמר ב-DB. | 12.04.20 | תקין |
| 3.2 | הגדרת/החלפת תמונת פרופיל – תמונה חדשה. | 1. המשתמש לחוץ על תמונת הפרופיל במסך הפרופיל. 2. המשתמש בוחר את אפשרות המצלמה. 3. המשתמש מצלם תמונה באמצעות אפליקציית המצלמה. | תמונת הפרופיל תוחלף ותעלה לאתר אחסון שישלח קישור שישמר ב-DB. | 12.04.20 | תקין |

## 4 – עדכון פרטים – מורה ותלמיד

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 4.1 | עדכון הפרטים האישיים של המשתמש. | 1. לחיצה על מקש 'עדכון פרטים' בדף הפרופיל. 2. שינוי הפרטים אותם מעוניין המשתמש לשנות. 3. לחיצה על מקש 'שמור'. | עדכון הפרטים ב-DB. | 12.04.20 | תקין |

## 5 - קביעת שיעור – תלמיד

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 5.1 | חיפוש מורה | 1. המשתמש יבחר מקצוע מבין רשימת המקצועות הקבועים מראש. 2. המשתמש יבחר אזור מבין רשימת האזורים הקבועים מראש. 3. המשתמש ילחץ על מקש 'המשך'. | תוצג רשימת המורים המתאימה להגדרות אותם בחר המשתמש. | 12.04.20 | תקין |
| 5.2 | בחירת מורה מתאים – צפייה בביקורות. | 1. המשתמש יבחר במורה מהרשימה המוצגת. 2. המשתמש יגיע לדף הפרטים על המורה. 3. המשתמש ילחץ על מקש 'הצג ביקורות'. | יוצגו הביקורות על המורה אותו בחר התלמיד כולל מס' הכוכבים שלו. | 12.04.20 | תקין |
| 5.3 | בחירת מורה מתאים – ע"י יצירת קשר טלפוני. | 1. המשתמש יבחר במורה מהרשימה המוצגת. 2. המשתמש יגיע לדף הפרטים על המורה. 3. המשתמש ילחץ על מקש 'צור קשר'. | המשתמש יגיע לחייגן של הטלפון עם מס' הטלפון של המורה. | 12.04.20 | תקין |
| 5.4 | קביעת שיעור – בחירת תאריך לא תקין. | 1. המשתמש ילחץ על מקש 'קבע שיעור' בדף הפרטים. 2. המשתמש יבחר תאריך לא תקין (לדוג' תאריך שכבר עבר) בלוח השנה שיוצג לפניו. | התאריך לא יבחר. | 12.04.20 | תקין |
| 5.5 | קביעת שיעור – בחירת תאריך תקין. | 1. המשתמש ילחץ על מקש 'קבע שיעור' בדף הפרטים. 2. המשתמש יבחר תאריך תקין (תאריך עתידי) בלוח השנה שיוצג לפניו. | המשתמש יועבר למסך בחירת השעה, בו יוצגו למשתמש רק השעות אותם בחר המורה כרלוונטיות ללימוד. | 12.04.20 | תקין |
| 5.6 | בחירת שעה | המשתמש יבחר שעה מתוך השעות המוצגות. | יוצג על המסך סיכום פרטי השיעור עם אפשרות לאשר או לא לאשר את השיעור. | 12.04.20 | תקין |
| 5.7 | קביעת שיעור | 1. המשתמש ילחץ על מקש 'מאשר'. 2. תוצג הודעה כי השיעור נקבע בהצלחה. 3. המשתמש ילחץ על מקש 'המשך'. | 1. השיעור יקבע בהצלחה וישמר ב-DB. 2. השיעור יופיע בדף 'מערכת שעות'. 3. השיעור יתעדכן בדף הפרופיל בתיבת 'סה"כ שיעורים' של התלמיד ושל המורה. 4. המורה יקבל מייל עם פרטי השיעור שנקבעו לו. 5. השעה אותה קבע התלמיד לא תופיע יותר עבור תלמידים נוספים. | 12.04.20 | תקין |

## 6 - מערכת שעות והוספת ביקורת – תלמיד

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 6.1 | הצגת השיעורים אותם קבע התלמיד. | לחיצה על מקש 'מערכת שעות' בסרגל התחתון שבמסך הבית. | מעבר לדף 'מערכת שעות' בו יוצגו ל השיעורים אותם קבע התלמיד. | 12.04.20 | תקין |
| 6.2 | הוספת ביקורת – כניסה למסך הוספת ביקורת. | לחיצה על מקש 'הוספת ביקורת' עבור המורה לו מעוניין התלמיד להוסיף ביקורת. | למשתמש תוצג תיבת דיאלוג עם אפשרות לדרג את המורה ע"י מס' הכוכבים המתאים לדעתו והוספת מלל חופשי. | 12.04.20 | תקין |
| 6.3 | הוספת ביקורת | 1. המשתמש יבחר את מס' הכוכבים המתאים לדעתו. 2. המשתמש יוסיף מלל חופשי בתיבת הטקסט. 3. המשתמש ילחץ על מקש 'שמור'. | 1. תוצג למשתמש הודעה כי הדירוג נוסף בהצלחה. 2. הביקורת תתווסף למסך הביקורות של המורה ולתיבת 'דירוג' בדף הפרופיל של המורה. 3. לאחר הוספת ביקורת לא יוכל התלמיד להוסיף ביקורת נוספת או להחליף את הביקורת אותה נתן. | 12.04.20 | תקין |

## 7 – צפייה בביקורות – מורה

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 7.1 | צפייה בביקורות ובדירוג שניתנו למורה. | לחיצה על מקש 'ביקורות' בסרגל התחתון. | מעבר לדף הביקורות בו יוצגו למשתמש כל הביקורות שנתינו לו ודירוג הכוכבים. | 12.04.20 | תקין |

## 8 – צפייה בשיעורים שנקבעו למורה ושינוי סטטוס שיעור – מורה

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 8.1 | צפייה בשיעורים שנקבעו למורה. | לחיצה על מקש 'מערכת שעות' בסרגל התחתון. | מעבר לדף 'מערכת שעות' בו יוצגו למשתמש כל השיעורים שנקבעו לו והסטטוס שלהם. | 12.04.20 | תקין |
| 8.2 | שינוי סטטוס שיעור. | 1. לחיצה על הסטטוס של השיעור הרצוי. 2. בחירת הסטטוס הרצוי מבין הסטטוסים שיוצגו בתיבת הדיאלוג שתופיע. 3. לחיצה על מקש 'שמירה'. | 1. סטטוס השיעור ישתנה במסך מערכת השעות של המורה והתלמיד. 2. סטטוס השיעור יעודכן ב-DB. 3. ישלח מייל לתלמיד אודות שינוי הסטטוס. | 12.04.20 | תקין |

## 9 – התנתקות מהמערכת – מורה ותלמיד

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 9.1 | ניתוק החשבון הפעיל מהמערכת. | לחיצה על מקש 'התנתק מהמערכת' שבדף הפרופיל. | המערכת תצא מהחשבון ותחזור לדף ההתחברות. | 12.04.20 | תקין |

## 10 – התחברות לאתר הניהול – חשבון מנהל

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס' | תיאור בדיקה | תהליך הבדיקה | תוצאה רצויה | תאריך הבדיקה | תוצאה |
| 10.1 | התחברות לאתר הניהול של האפליקציה – חשבון מנהל. | 1. לחיצה על מקש 'SIGN IN'. 2. התחברות ל-API של גוגל עם חשבון מנהל. | מעבר לדף הניהול. | 12.04.20 | תקין |
| 10.2 | התחברות לאתר הניהול של האפליקציה – חשבון שאינו חשבון מנהל. | 1. לחיצה על מקש 'SIGN IN'. 2. התחברות ל-API של גוגל עם חשבון שאינו מוגדר כמנהל. | קבלת הודעה על חשבון לא תקין. | 12.04.20 | תקין |

# 12. 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 הראשונית לאפליקציה, וכן בכל רגע נתון, המערכת בודקת את מצב ההתחברות של המשתמש במכשיר הנייד. במידה והמשתמש כבר היה מחובר, המערכת דואגת לחבר אותו באופן אוטומטי. 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 Exception שהמשתמש אינו מחובר ו הוא יועבר למסך ההתחברות.

התהליך מתבצע בעת הכניסה לאפליקציה באופן שאינו שקוף למשתמש.

    async config => {

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

      if (token) {

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

      }

      return config;

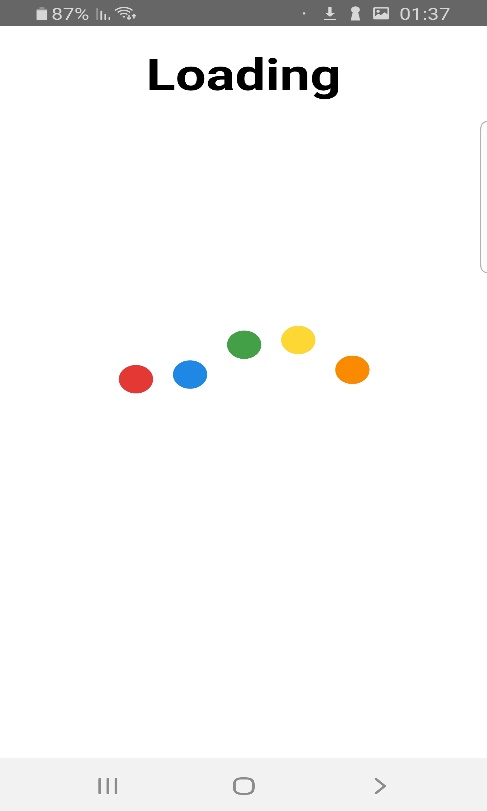
    },

    err => {

      return Promise.reject(err);

    }

  );



בכל פניה לAPI מבצעת הבדיקה של הTOKEN- ובמידה והוא קיים ותקין - מתבצעת בדיקה האם המשתמש כבר רשום. במידה וכן הוא יועבר 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");

  }

};

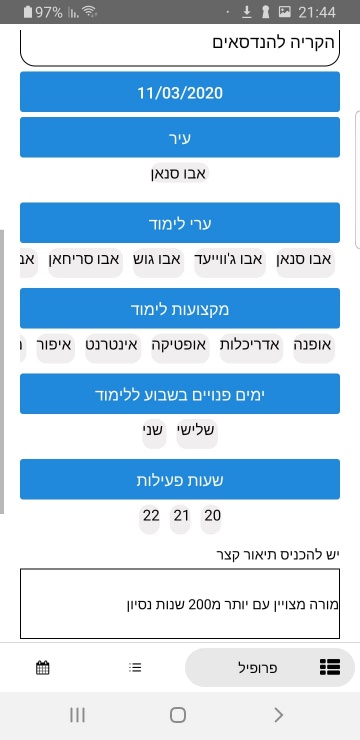
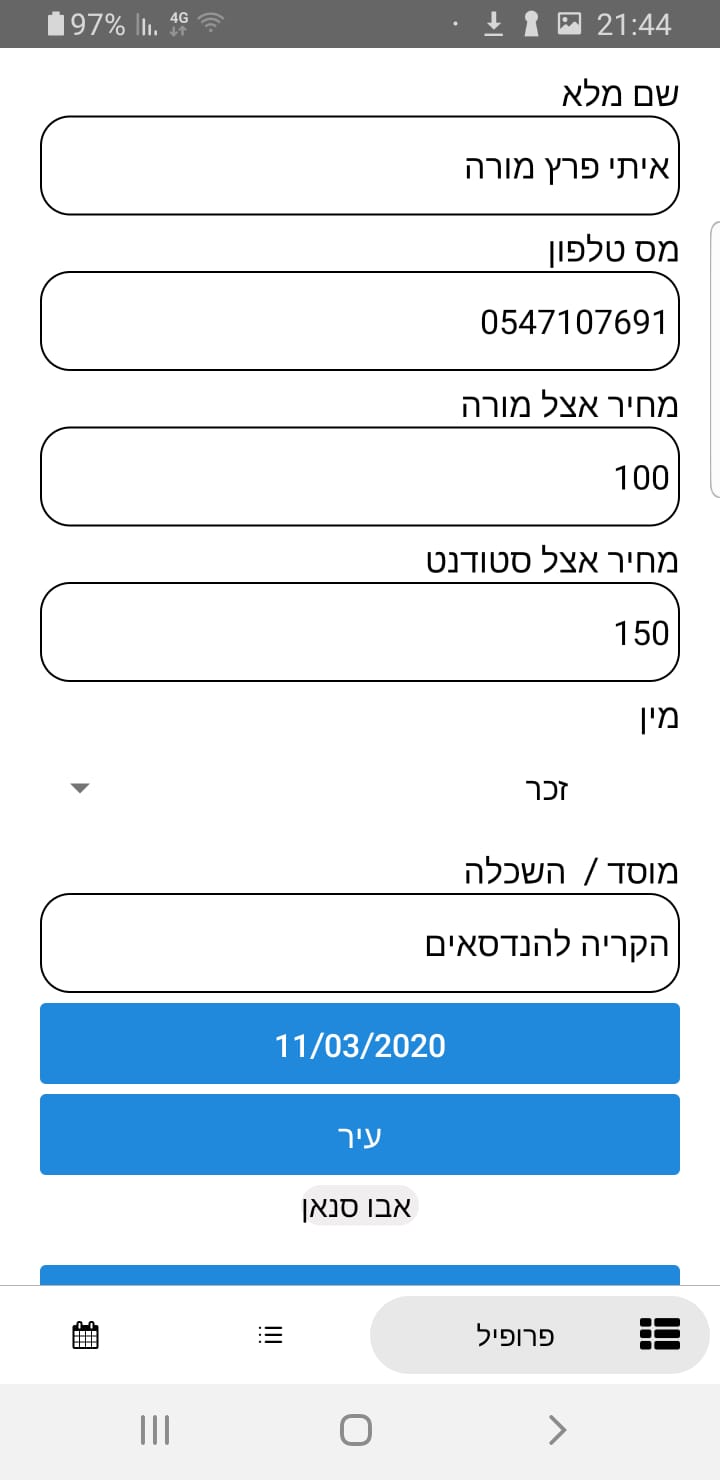
## בחירת פרופיל

כפי שניתן לראות בקוד למעלה, ברגע שלמשתמש אין משתמש של מורה או תלמיד הוא מגיע לדף בחירת פרופיל:

## מורה

### הרשמה/עדכון פרטים

במידה והמשתמש בחר בפרופיל מורה, הוא יגיע לדף הבא בו יצטרך להזין את הפרטים שלו, כגון: שם, טלפון, מין, ערי לימוד, מקצועות לימוד וכדו'.



אותה מחלקה אחראית גם על עדכון פרטי המורה.

בקוד הבא ניתן לראות את הפעולה שמאורי כפתור 'שמור',

מתבצעת שמירה של כל השדות למשתנים תחת אובייקט obj וואלדציה לכל שדות.

            <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 {

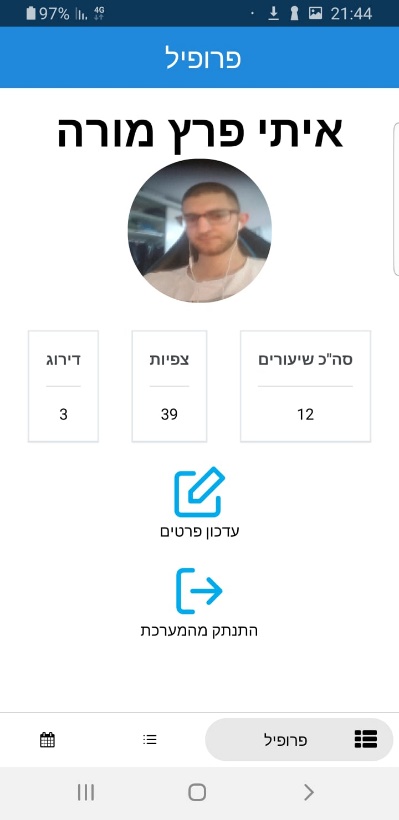
                  console.log(this.state.availablesDays);

                  this.updateDetails(obj);

                }

              }}

            />



### פרופיל

במסך הפרופיל, יכול המורה לראות את הנתונים הרלוונטיים עבורו, כגון: מס' צפיות וסה"כ שיעורים.

בנוסף, הוא יוכל לעדכן את הפרטים האישיים שלו ולשנות תמונת הפרופיל.

בקוד למטה ניתן לראות את תהליך הבנייה של המסך.

SELECTPIC אחראי על צילום התמונה כתוצאה מלחיצה 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, ניתן לראות את הביקורות שקיבל המורה

ואת הדירוג 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

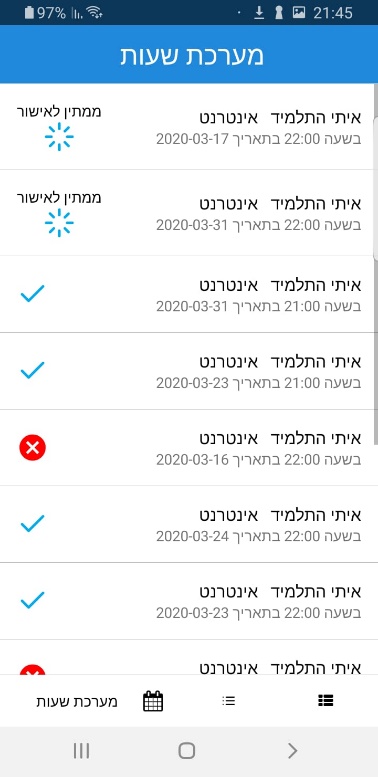
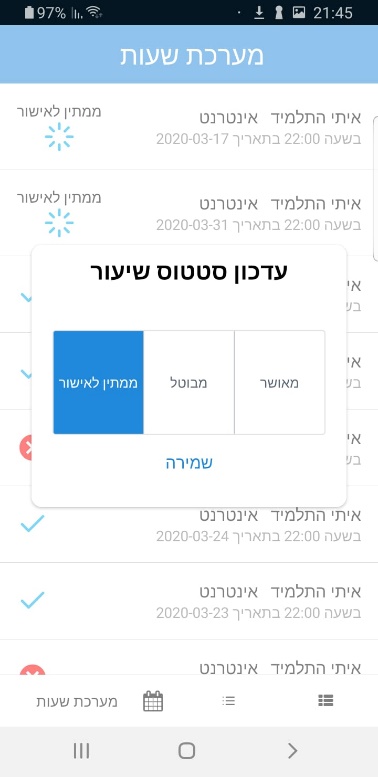
        startingValue={ratingSum}

      />

### מערכת שעות

במסך הבא יוצגו השיעורים שנקבעו למורה ואת הסטטוס שלהם.

המורה יכול לשנות את הסטטוס ע"י לחיצה על האייקון של הסטטוס.



קטע הקוד הבא מתאר את בניית המסך, כאשר**TimeTableStatus**  הוא האובייקט 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>

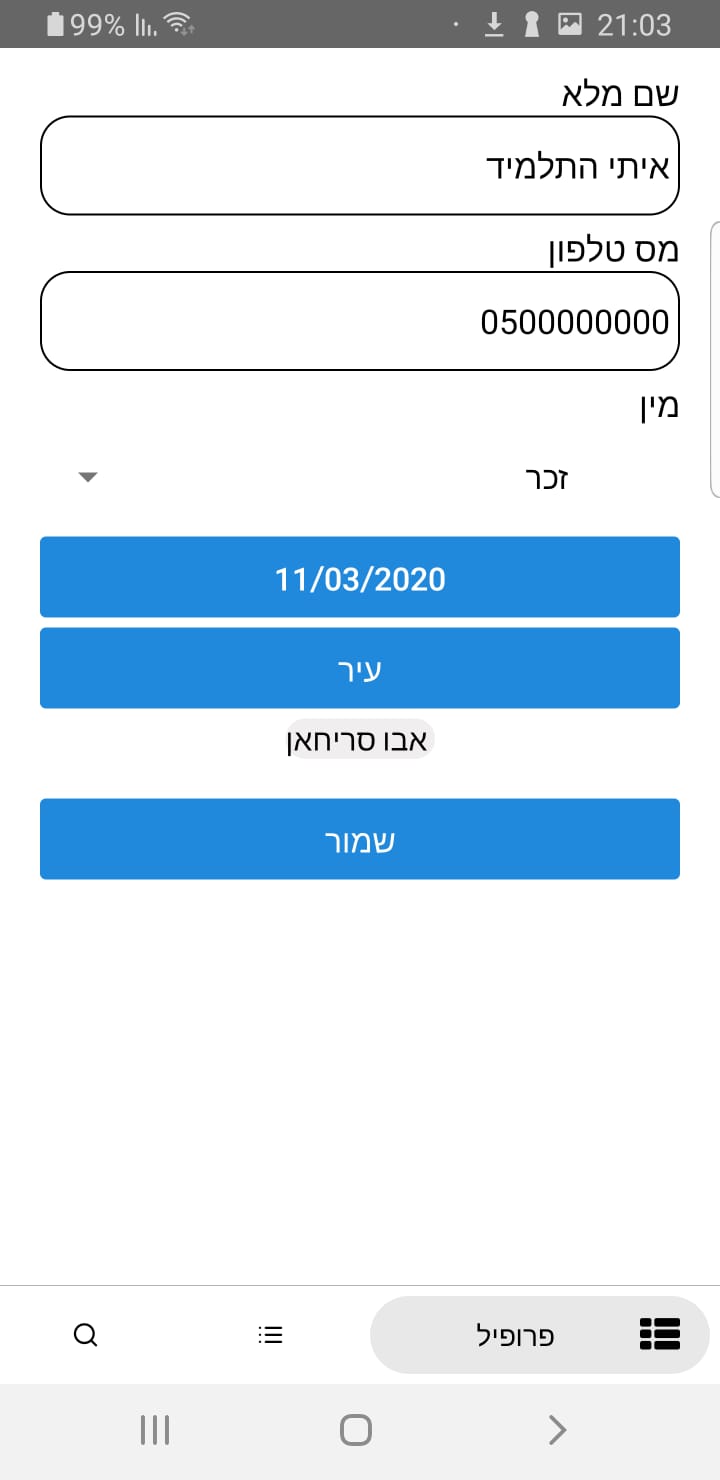
          ))}

        </View>

## תלמיד

### הרשמה / עדכון פרטים

במידה והמשתמש בחר בפרופיל תלמיד הוא יגיע למסך ההרשמה

ויצטרך למלא מספר פרטים, כגון: עיר, טלפון, שנת לידה וכדו'.

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 {

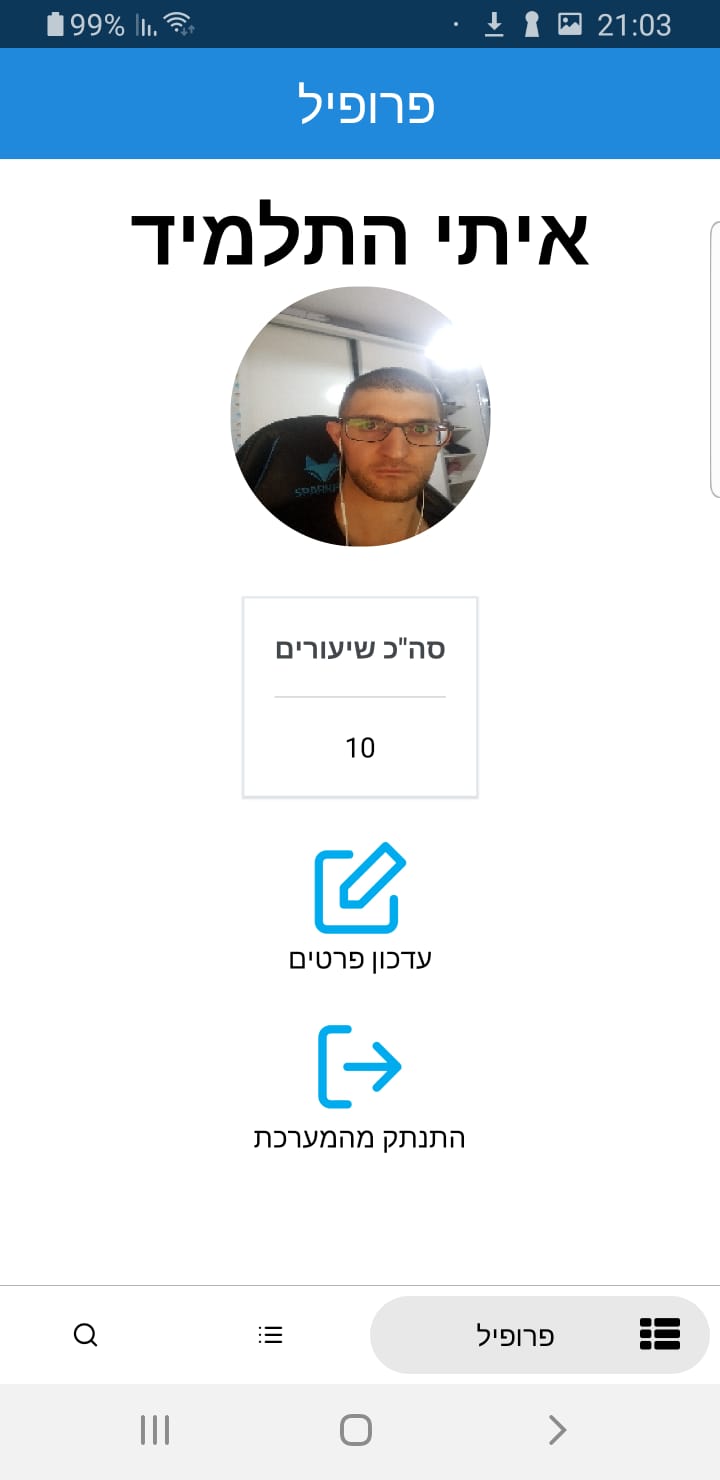
                  this.updateDetails(obj);

                }

              }}

            />

### פרופיל



התלמיד יגיע למסך הבא לאחר שיבחר בפרופיל תלמיד.

במסך זה יוכל לראות את כמות השיעורים שקבע,

לעדכן את הפרטים ולבחור תמונה לפרופיל.

בקוד למטה אפשר לראות את הבניה של המסך.

כמות השיעורים מתקבלת בפונקציה 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");

            }}

          >

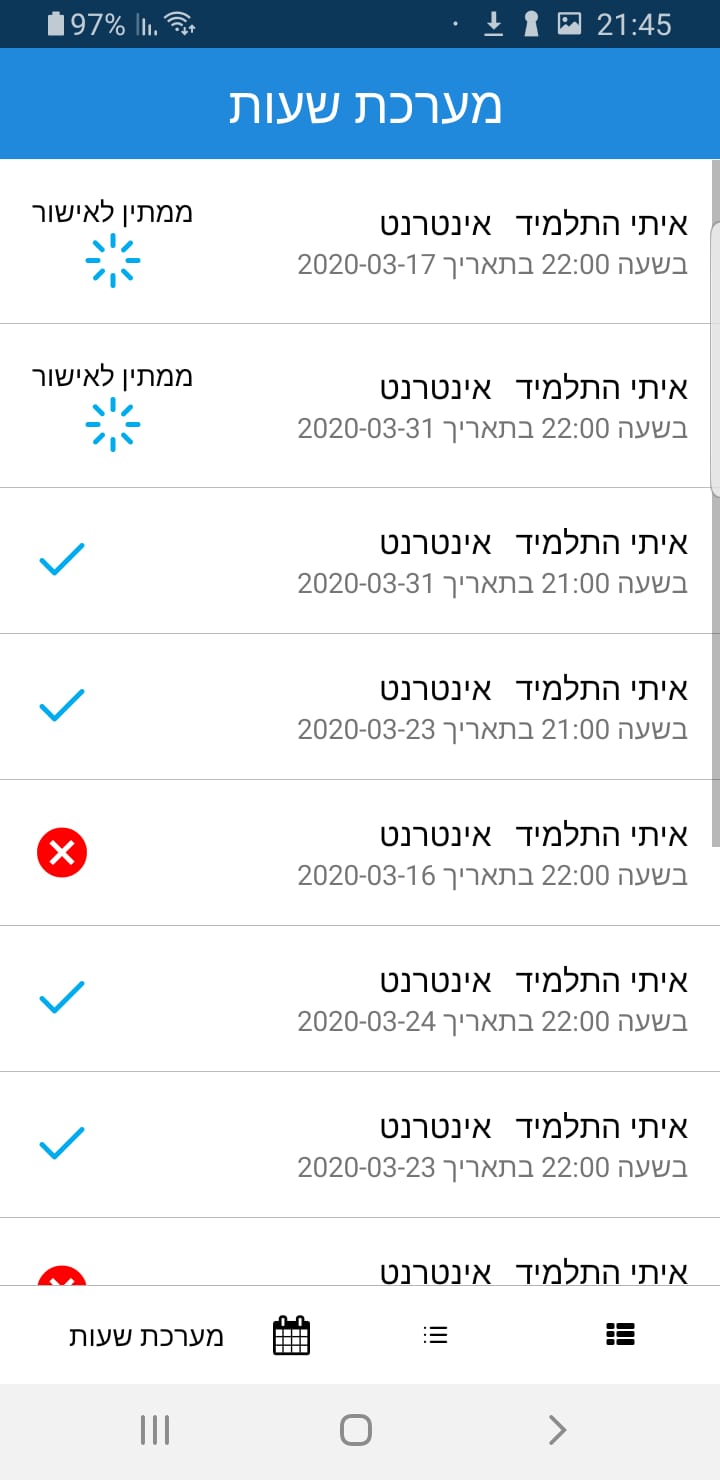
            <Icon name="edit" type="feather" color="#00aced" size={50} />

            <Text>עדכון פרטים</Text>

          </TouchableOpacity>

          <SignOutIcon />

### מערכת שעות



במסך הבא יוצגו השיעורים שנקבעו ואת הסטטוס שלהם.

מבחינת קוד, מדובר על אותו הקוד כמו מסך מערכת שעות של המורה.

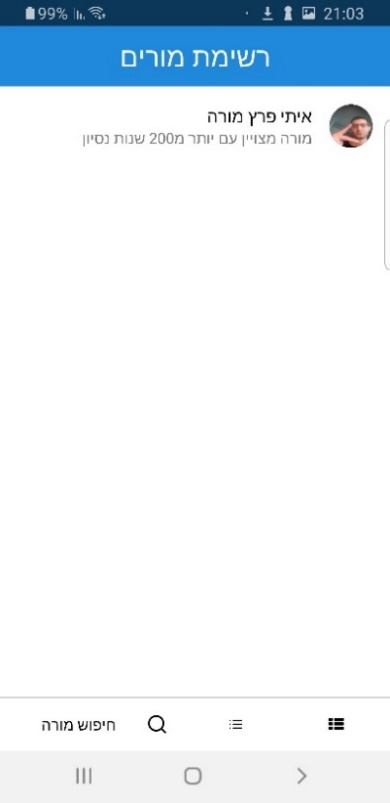
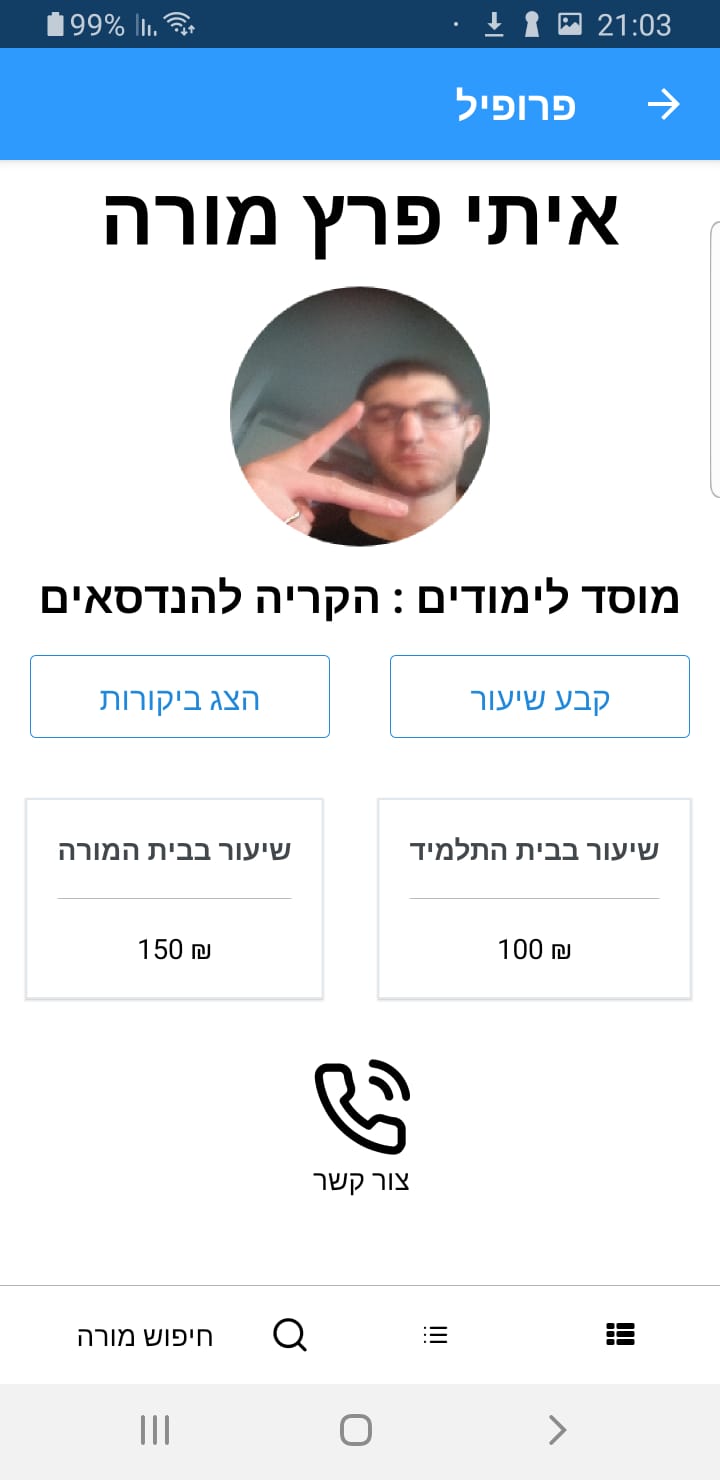
### חיפוש מורה

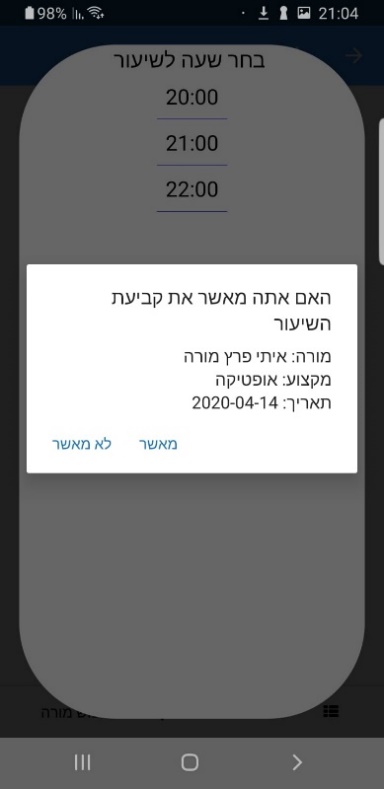
במסך זה יש לבחור בעיר ובמקצוע למציאת מורים רלוונטיים.

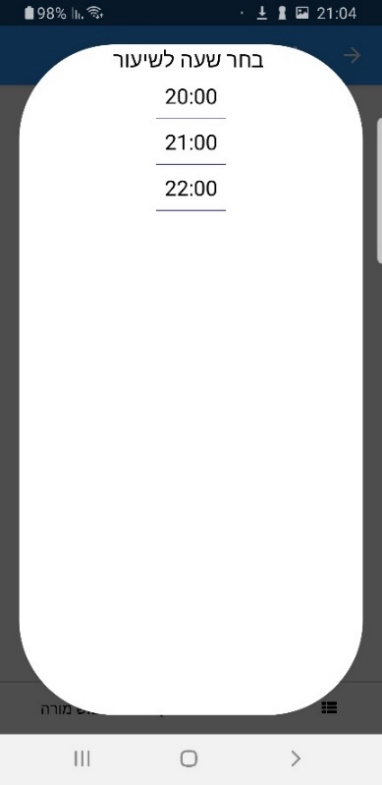
רשימת הערים נמצאת בקובץ JSON מרשות האוכלוסין והאגירה, ורשימת המקצועות מכילה את רוב המקצועות שקיימים.

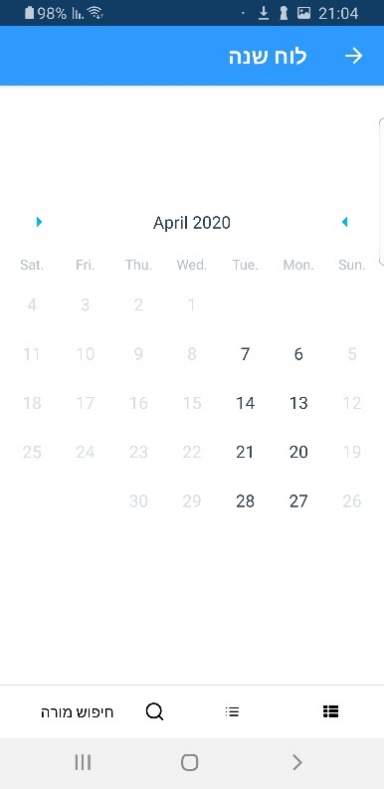
במסכים הבאים ניתן לראות את הסדר של קביעת השיעור:

1. חיפוש מורה לפי מיקום ומקצוע.
2. בחירת מורה מהרשימה המתקבלת.
3. לחיצה על מקש 'ביקורות' תציג את הביקורות על המורה.
4. לחיצה על 'קבע שיעור' תציג לוח שנה בו מסומנים הימים בהם זמין המורה.
5. לאחר לחיצה על היום, יתקבלו השעות בהם זמין המורה.
6. לאחר אישור השיעור, תתקבל הודעה כי השיעור נקבע בהצלחה.









קטעי הקוד הבאים אחראים על תהליך חיפוש המורה.

**חיפוש מורה:**

הקוד מציג את 2 הלשוניות של בחירת הערים וברגע שהמשתמש בוחר בעיר או במקצוע יתבצע set שיגדיר את העיר/מקצוע בזיכרון, שמאוחר יותר ישמש לצורף ה-QUERY 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 ה-API,

ברגע שהמשתמש ילחץ על מסך המורה תתרחש קריאה של onpress שתבחר את המורה המבוקש, 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={{

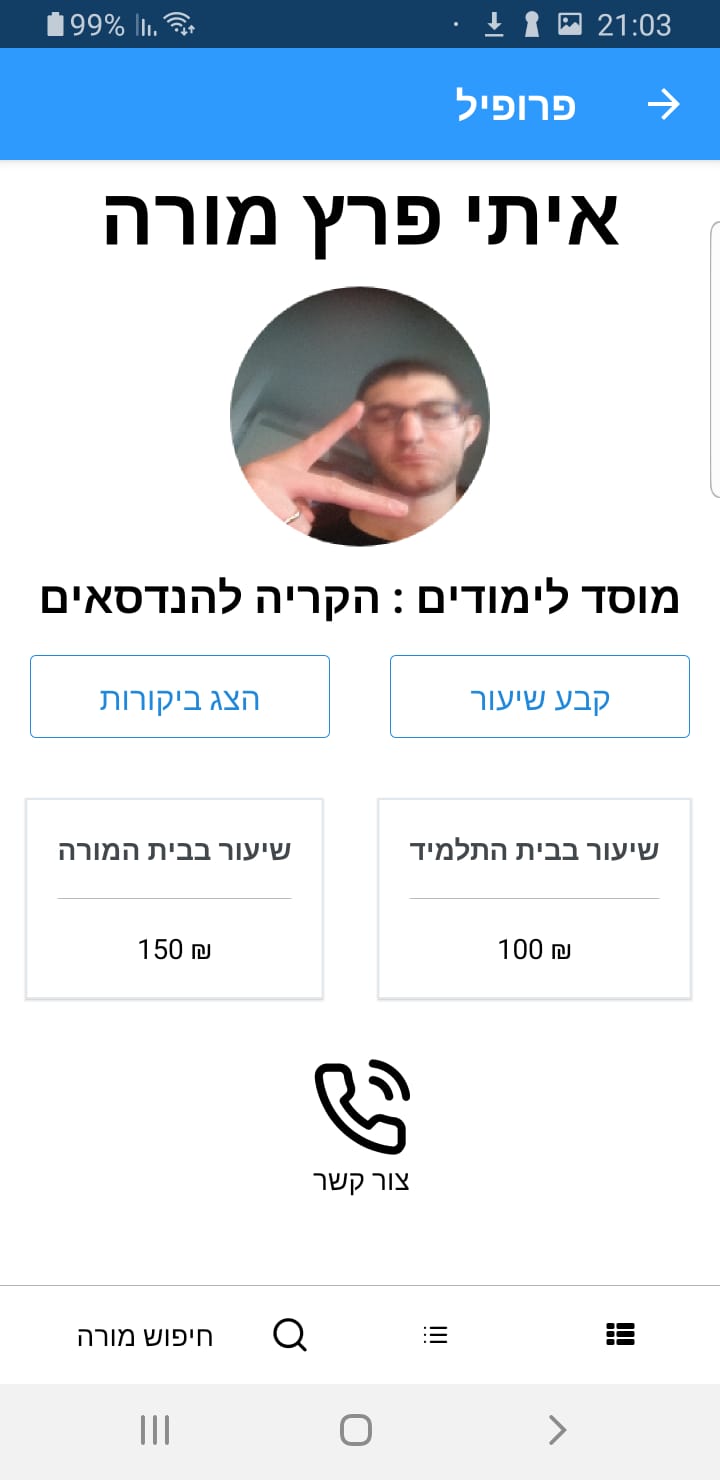
                source: { uri: l.pic || "https://i.imgur.com/XlQzxuT.png" }

              }}

              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 ניתן לראות את מחיר השיעור בבית התלמיד ואצל

המורה.

קטע הקוד הבא מציג את האפשרויות של המסך למעלה.

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

לחיצה על 'צור קשר' תפעיל את Linking.openURL שיציג את מספר הטלפון בחייגן של הטלפון.

   <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}`);

          }}

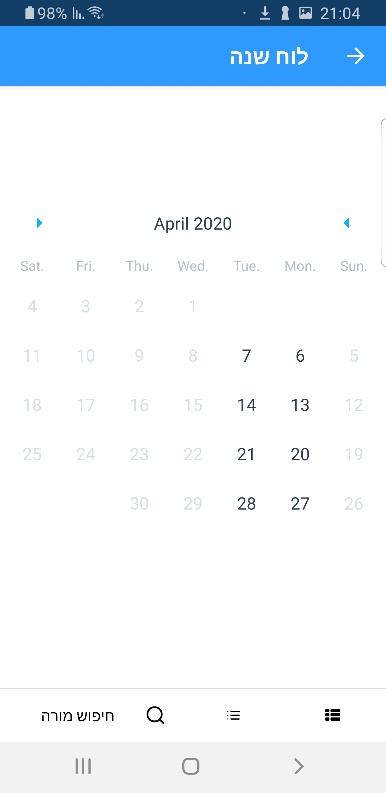
        >

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

          <Text>צור קשר</Text>

        </TouchableOpacity>

**קביעת יום לשיעור:**

במסך הבא יוצגו כל הימים שהמורה מלמד בהם.

כל הימים הלא רלוונטיים יוצגו בצבע אפור ולא ניתנים לבחירה.

לחיצה על התאריך המבוקש תבצע פנייה ל-API, שיבדוק באיזה שעות המורה פנוי, כאשר מתבעת בדיקה גם לגבי שעות אשר נתפסו על ידי שיעורים אחרים שנקבעו.

הקוד הבא אחראי על ההצגה של לוח השנה.

כאשר הוא מקבל כפרמטר את הימים בהם המורה פנוי ומציג אותם למשתמש.

אפשר לראות שה-MIN DATE הוא היום, כך שלא ניתן יהיה לקבוע שיעור ליום שעבר.

   <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);

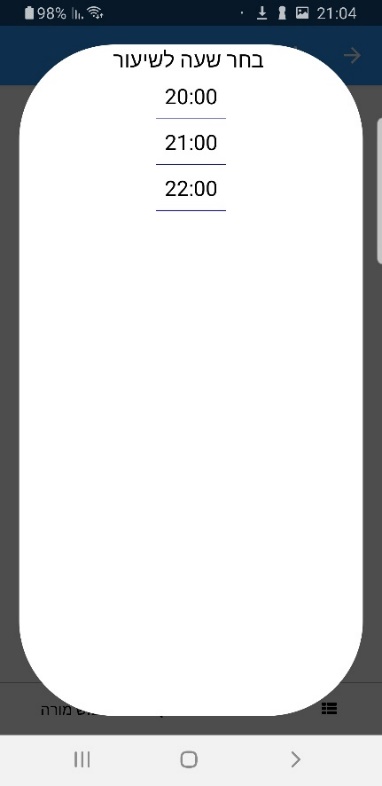
          }

        }}

        hideExtraDays={true}

        markedDates={monthList}

      />

לאחר לחיצה על התאריך יתקבל מה-API רשימת השעות בהם המורה זמין, מה שיקפיץ בחלון חדש על המסך את האפשרות לבחור מסך.

אפשר לראות שבעת לחיצה מתבצעת קריאה ל-API, לאחר קבלת התשובה, הפרמטר SETVISIABLE יהיה שווה ל-TRUE, כך שיוצג החלון לבחירת השעות.

חלון האחראי על הצגת השעות הפנויות.

Flatlist אחראי על קבלת מערך ויצירת אלמנטים ממנו, ובמקרה הזה כל אלמנט מייצג 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 יאושר, יקפוץ חלון המבקש את האישור הסופי 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);

                      }

                    }}

                  >

לאחר 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

const axios = require("axios");

const conf = require("../../server.config");

const url = "https://api.imgur.com";

const apiGur = axios.create({

  baseURL: url,

  headers: {

    Authorization: conf.APIGURID

  }

});

const uploadPicture = async image => {

  const response = await apiGur.post("/3/upload", {

    image

  });

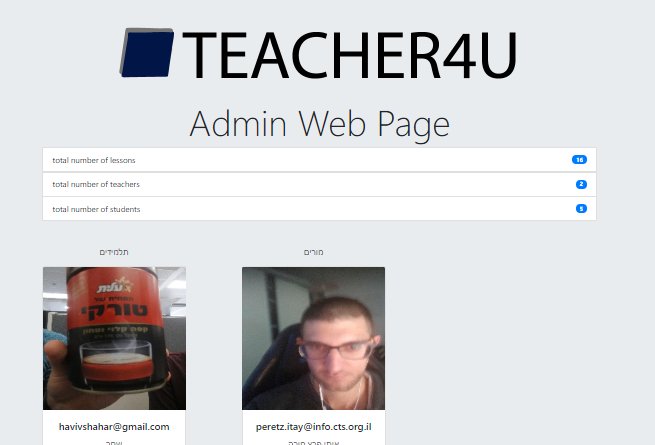
  return response.data.data.link;

};

module.exports = uploadPicture;

דפי הנהלה

בדפי ההנהלה אפשר לראות נתונים שונים על השימוש באפליקציה



הקוד לבניית הHTML נמצא מטה

<div class="jumbotron text-center">

  <div class="container">

    <img src="logomini.png" alt="Smiley face" />

    <h1 class="display-3">Admin Web Page</h1>

    <div style="margin-bottom: 50px;">

      <ul class="list-group">

        <li class="list-group-item d-flex justify-content-between align-items-center">

          total number of lessons

          <span class="badge badge-primary badge-pill"><%= timetables.length %></span>

        </li>

        <li class="list-group-item d-flex justify-content-between align-items-center">

          total number of teachers

          <span class="badge badge-primary badge-pill">

            <%= teachers.length %></span>

        </li>

        <li class="list-group-item d-flex justify-content-between align-items-center">

          total number of students

          <span class="badge badge-primary badge-pill"><%= students.length %></span>

        </li>

      </ul>

    </div>

    <div class="container">

      <div class="row">

        <ul class="list-group" style="margin-right: 400px;">

          <p>תלמידים </p>

          <% students.forEach(function(student){ %>

          <div class="card" style="width: 18rem;">

            <% if (student.pic) { %>

            <img src=<%= student.pic %> class="card-img-top" alt="..." />

            <% }  %>

            <% if (student.pic == undefined || student.pic == "") { %>

            <img src="https://i.imgur.com/jzAfnJ9.png" class="card-img-top" alt="..." />

            <% }  %>

            <div class="card-body">

              <h5 class="card-title"><%= student.email %></h5>

              <p class="card-text">

                <%= student.fullName %>

              </p>

            </div>

          </div>

          <% }) %>

        </ul>

        <ul class="list-group">

          <p>מורים </p>

          <% teachers.forEach(function(teacher){ %>

          <div class="card" style="width: 18rem;">

            <% if (teacher.pic) { %>

            <img src=<%= teacher.pic %> class="card-img-top" alt="..." />

            <% }  %>

            <% if (teacher.pic == undefined || teacher.pic == "") { %>

            <img src="https://i.imgur.com/jzAfnJ9.png" class="card-img-top" alt="..." />

            <% }  %>

            <div class="card-body">

              <h5 class="card-title"><%= teacher.email %></h5>

              <p class="card-text">

                <%= teacher.fullName %>

              </p>

            </div>

          </div>

          <% }) %>

        </ul>

      </div>

    </div>

## 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