**פרויקט שנתי**

מנחה: **פרופסור יוסי בן-אשר**

מגישים: **דני מדן וטל בוגן**

קוד מקור: **https://github.com/talbo25/fATcAT**

|  |  |
| --- | --- |
| fatcat  קערה חכמה לחיות מחמד |  |

[תיאור פרויקט 2](#_Toc53479216)

[מטרות הפרויקט 3](#_Toc53479217)

[לפתור בעיה רווחת בקרב בעלי חתולים 3](#_Toc53479218)

[כניסה לעולם ההתקנים המתממשקים לענן 3](#_Toc53479219)

[נסיון בטכנולוגיות ושפות מובילות בתעשייה 3](#_Toc53479220)

[עבודה מעשית במסגרת התואר 4](#_Toc53479221)

[פיתוח מוצר על כל שלביו 4](#_Toc53479222)

[תיאור המערכת 5](#_Toc53479223)

[דרישות פונקציונאליות 7](#_Toc53479224)

[ממשק משתמש 7](#_Toc53479225)

[קערה חכמה 8](#_Toc53479226)

[שרת 9](#_Toc53479227)

[מסד הנתונים 9](#_Toc53479228)

[דרישות לא פונקציונאליות 10](#_Toc53479229)

[תכן 11](#_Toc53479230)

[ממשק משתמש 11](#_Toc53479231)

[קערה חכמה 12](#_Toc53479232)

[השרת 15](#_Toc53479233)

[מסד הנתונים 16](#_Toc53479234)

[סביבת עבודה ושפות 19](#_Toc53479235)

[סביבות עבודה 19](#_Toc53479236)

[שפות 19](#_Toc53479237)

[שלבי ביצוע 20](#_Toc53479238)

[סיכום פרויקט 24](#_Toc53479239)

[נספחים 26](#_Toc53479240)

[נספח א' - מדריך למשתמש באפליקציית fATcAT 26](#_Toc53479241)

[נספח ב' - חיבור ראשוני של המערכת לרשת WIFI מקומית 34](#_Toc53479242)

תיאור פרויקט

הפרויקט הוא מערכת האכלה לחיות מחמד בכלל וחתולים בפרט, המקושרת בממשק ענן לאפליקציית **Mobile**.

זיהוי בעלי החיים נעשה לפי משקל, שכן מערכת זו נועדה לבעלים של חיות מחמד אשר רוצים לנהל ולעקוב אחרי התזונה של בעלי החיים שלהם, בדגש על תזונה ייחודית של בעל חיים מסוים.

הפרויקט מורכב מכמה טכנולוגיות:

* שירותי ענן
* מודל שרת-לקוח
* ממשק משתמש ל- mobile
* שימוש במסד נתונים
* שילוב חומרה ותוכנה

הפרויקט כולל שימוש בכמה שפות תכנות :

* Sub-version of **c++** for **Arduino**
* **React Native**
* **NodeJS**
* **MongoDB**

**מגישים**:

* טל בוגן
* דני מדן

מטרות הפרויקט

לפתור בעיה רווחת בקרב בעלי חתולים  
בעלי חתולים רבים נתקלים בבעיה של חתול דומיננטי שגונב את המזון מהחתול השני. כמו כן, קיימים חתולים רבים עם צרכים רפואיים שחייבים באוכל דיאטטי/בריאותי ועל הבעלים לדאוג להרגלי האכילה שלהם. הפרויקט נועד לפתור את הבעיות הללו כאשר הגורם המבדיל הוא המשקל. כמובן שניתן להרחיב את יכולת האבחנה של המכשיר בעתיד (זיהוי תמונה, RFID ועוד).

כניסה לעולם ההתקנים המתממשקים לענן

עולם ה- **Makers** הוא תחום גדול ומתפתח בעולם הטכנולוגי. אחד מהמונחים העכשוויים והנחשקים ביותר כיום הוא ה- **IOT** (Internet of Things)– קישור מוצרים יום יומיים לאינטרנט ולטלפון החכם לצורך שליטה ומעקב. בעזרת קישור לאינטרנט, נוכל להפוך את מערכת הקערה לחכמה, ולחברה לטלפון לצורך מעקב, שליטה ונוחות.

ניסיון בטכנולוגיות ושפות מובילות בתעשייה

עולם ה-**Web** וה-**Mobile** נעשה בעשור האחרון לדרך המתפתחת והנפוצה ביותר ליישום שירותים ללקוחות קצה.  
שפות מבוססות **JavaScript** ופלטפורמות כמו **NodeJS** נדרשות יותר ויותר הן בשל נוחיותן, ביצועיהן והחינמיות שלהן, דבר שמרחיב את בסיס הכותבים בהן וכפועל יוצא גם קהל התמיכה למפתחים חדשים בתחום.  
בנוסף, גם **ממשקי ענן** נעשים בשימוש נרחב בשנים האחרונות, והמגמה רק מתגברת.

עבודה מעשית במסגרת התואר

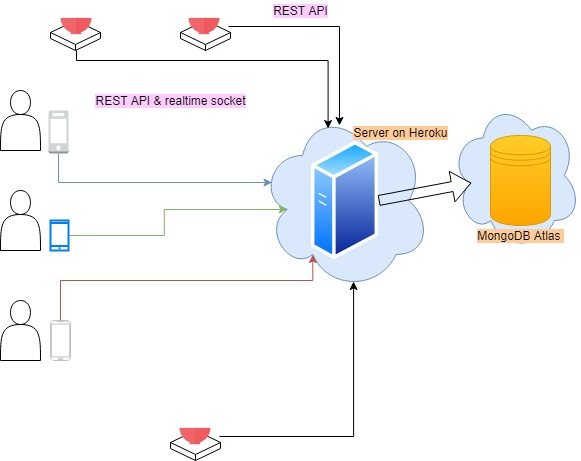
מעבר לבחינות כאלה ואחרות, שילוב בשוק העבודה היום דורש הרבה פעמים הצגת "**תיק עבודות**" בשלב הראיונות.  
אנו מקווים שפרויקט כזה יוכל להציג יכולת יצירת מערכת משלב הגדרת המטרות ועד לשלב המוצר המוגמר, וכן גם למידה עצמאית ושימוש בשפות/טכנולוגיות נדרשות.

פיתוח מוצר על כל שלביו

בפרויקט זה ניישם את השלבים הנדרשים בפיתוח מוצר: החל משלב הרעיון, דרך שלב האפיון, הדרישות, התכן, הבדיקות ועד לשלב המוצר המוגמר.  
מכיוון שזהו ניסיון ראשון בתהליך מסוג זה עבורנו, אנו בטוחים שניתקל בקשיים ובדברים לא צפויים במהלך העבודה על הפרויקט, אבל אנחנו מקווים שעם הרבה השקעה נוכל לעמוד בהם.  
אנו מקווים שתהליך זה ייתן לנו טעימה מעולם טכנולוגי רחב יותר מזה שיצא לנו להכיר עד היום.

תיאור המערכת

המערכת מורכבת מ-4 תת-מערכות הכוללות:

1. קערה חכמה
2. ממשק משתמש
3. שרת
4. מסד נתונים

הקערה החכמה מאפשרת או מונעת גישה לקערת האוכל של החתול על פי הנתונים שהמשתמש הזין למערכת בעזרת ממשק המשתמש.

הממשק יכול להיות מותקן על מערכות הפעלה **Android** ו-**IOS**.  
דרך ממשק זה יזין המשתמש את נתוני החתול וינהל את הקערות ואת החתולים שבבעלותו.

המערכת שומרת לוגים למספר אירועים הקורים לקערה החכמה:

1. החתול עלה על המשקל והחל לאכול.
2. החתול סיים לאכול וירד מהמשקל.
3. חתול לא מזוהה עלה על המשקל.
4. החתול מזוהה עלה על המשקל בשעות שלא הוגדרו עבורו.
5. החתול מפריע לחברו לאכול.
6. החתול חזר לאכול אחרי שהפריעו לו.
7. כל החתולים הלכו לאחר ההפרעה.
8. הקערה הועברה למצב "ידני".
9. הקערה הועברה למצב "אוטומטי".
10. המערכת הותחלה.

את הלוגים הללו יכול המשתמש לראות דרך ממשק המשתמש (יוצגו 50 האחרונים כרונולוגית).

דרישות פונקציונאליות

ממשק משתמש –

* 1. משתמש חדש יזוהה לפי ה-ID של המכשיר שממנו מופעל ממשק המשתמש.
  2. משתמש חדש יוכנס למערכת באופן אוטומטי, כאשר תת-המערכת "השרת" תזהה חיבור ממשק משתמש ממכשיר חדש.
  3. תת-המערכת נדרשת לממש מסך המאפשר למשתמש להוסיף נתונים למערכת:
     1. הוספת קערה למשתמש
     2. הוספת חתול לקערה
  4. תת-המערכת נדרשת לממש מסך המאפשר למשתמש לערוך את הנתונים המקושרים אליו:
     1. קערה:
        1. כינוי מזהה עבור הקערה
        2. שעות פעילות הקערה
        3. מצב הקערה "ידני" או "אוטומטי"
     2. חתול:
        1. שם החתול
        2. מין החתול
        3. משקל החתול
        4. שעות האכלת החתול
  5. תת-המערכת נדרשת להיות מחוברת לאינטרנט.
  6. תת-המערכת נדרשת לפתוח ממשק ענן לתת-מערכת השרת.
  7. תת-המערכת נדרשת לשלוח את המידע שהתקבל מהמשתמש לשרת.   
     המידע יכלול פרטיי חתול/קערה חדשים/ערוכים.
  8. תת-המערכת נדרשת לממש מסך המציג למשתמש את נתוני המערכת הקשורים אליו:
     1. הקערות הקשורות אליו
     2. החתולים הקשורים אליו
     3. הלוגים הקשורים אליו

קערה חכמה –

1. תת-המערכת נדרשת להיות מחוברת לאינטרנט.
2. תת-המערכת נדרשת לפתוח ממשק ענן ולהתעדכן בשעון מקומי.
3. תת-המערכת נדרשת לפתוח ממשק ענן לתת-מערכת "השרת".
4. תת-המערכת נדרשת לזהות עלייה וירידה במשקל שעל משטח השקילה.
5. תת-המערכת נדרשת לזהות את החתול שמגיע לאכול.   
   במידה ולא זוהה – לעדכן בלוג.
6. תת-המערכת נדרשת להחליט האם החתול שזוהה הגיע בשעות האכלה שלו ובשעות הפעילות של הקערה.   
   במידה וכן – לפתוח את המחסום ולאפשר גישה לאוכל.   
   בכל מקרה, לעדכן לוג בהחלטה.
7. תת-המערכת נדרשת לזהות הפרעה לחתול בעת שזה אוכל.   
   במידה וזוהתה הפרעה – לסגור המחסום.   
   בנוסף, לעדכן לוג באירוע.
8. תת-המערכת נדרשת לזהות את סיום האכילה של החתול.  
   ומידה וזוהה – לסגור המחסום.  
   בנוסף, לעדכן לוג באירוע.
9. במידה והתקבלה קריאת מעבר מצב - תת-המערכת נדרשת לפעול בצורה הבאה:
   1. מעבר למצב "ידני" – על המחסום להיפתח ולאפשר גישה לאוכל ( ללא תלות בנתוני השעות או המשקל)  
      בנוסף, לעדכן לוג על מעבר הקערה למצב "ידני".
   2. מעבר למצב "אוטומטי" – על המחסום לחזור להיות תלוי בנתוני השעות והמשקל.  
      בנוסף, לעדכן לוג על מעבר הקערה למצב "אוטומטי".
10. במידה וקערה חכמה הועברה למצב "ידני" ולא הוחזרה למצב "אוטומטי" – היא תוחזר למצב "אוטומטי" לאחר 20 דקות.
11. במידה והתקבלה דרישה למשקל הנוכחי תת-המערכת "הקערה" נדרשת לשלוח לתת-המערכת "השרת" את נתוני המשקל הנוכחיים ולהציגם למשתמש.

שרת –

1. תת-המערכת נדרשת לפתוח ממשק ענן עבור כל קערה חכמה.
2. תת-המערכת נדרשת לפתוח ממשק ענן עבור כל ממשק משתמש.
3. תת-המערכת נדרשת לפתוח ממשק ענן עבור תת-המערכת "מסד הנתונים".
4. תת-המערכת נדרשת לנתב בקשות ממשק המשתמש אל הקערה החכמה ולהחזיר את התשובה.
5. תת-המערכת נדרשת להעביר לשמירה במסד הנתונים את כל המידע המוזן על ידי המשתמש דרך ממשק המשתמש.
6. תת-המערכת נדרשת לעדכן את כל הקערות החכמות במידע הרלוונטי להן.
7. תת-המערכת נדרשת להעביר לשמירה במסד הנתונים את כל הלוגים המפורטים בתיאור המערכת.

מסד הנתונים –

* 1. תת-המערכת נדרשת לפתוח ממשק ענן עבור תת-המערכת "השרת".
  2. תת-המערכת נדרשת לשמור, לשלוף, למחוק ולערוך את המידע מן המשתמש ואת הלוגים מהקערות החכמות על ידי תת מערכת "השרת".

דרישות לא פונקציונאליות

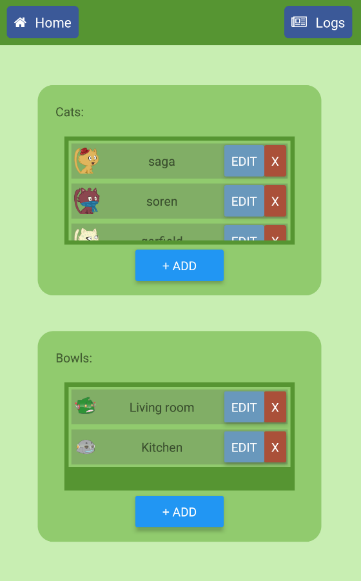
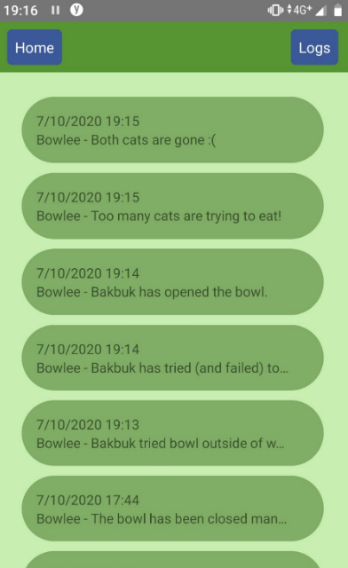
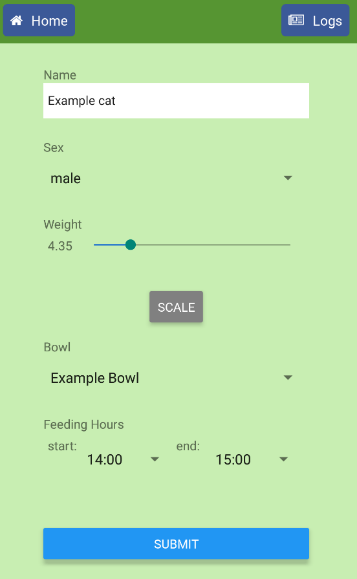
1. נדרש "קינפוג" ראשוני במפעל (**Factory Setting**) של כל קערה חכמה הכולל ID ו-KEY.
2. נדרש שמירת שמות החתולים ומשקלם לצורך זיהוי.
3. על תת-מערכת "הקערה החכמה" לקבל נתוני משקל באופן רציף.
4. כאשר קערה חכמה לא תזהה את תת-המערכת "השרת" – היא נדרשת להישאר פתוחה.
5. כל קערה חכמה תתעדכן בנתונים מתת-המערכת "השרת" בכל ~5 שניות.
6. לכל תת-מערכת "משתמש" ניתן לקשור מספר קערות.
7. לכל תת-מערכת "קערה חכמה" ניתן לקשר מספר חתולים.
8. חתול יכול להיות מקושר אך ורק לקערה אחת.
9. תתי-המערכות "השרת" ו"מסד הנתונים" יהיו זמינים 24/7.

תכן

ממשק משתמש**:**

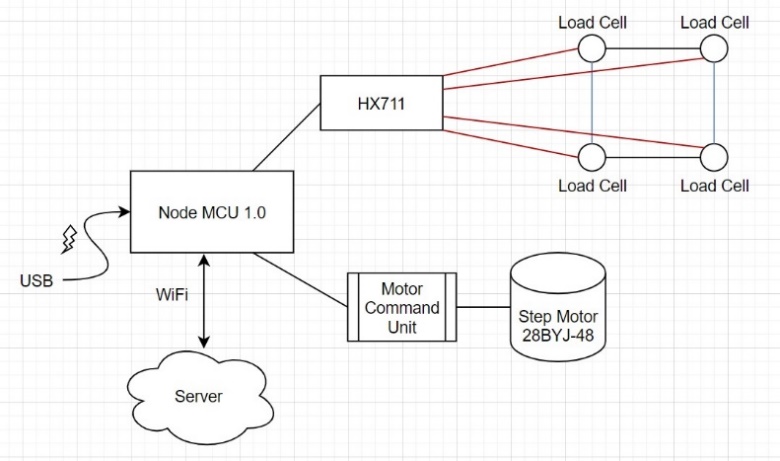
* 1. יישום האפליקציה בעזרת **React Native**.  
     הבחירה בשפה זו נעשתה משום ששפה זו מתאימה למערכות ההפעלה **Android ו-IOS** כאחד, ובעלת תמיכה נרחבת באינטרנט (הן מבחינת שאלות והן מבחינת ספריות).
  2. ממשק המשתמש מבקש נתונים עדכניים ולוגים ומעדכן נתונים באמצעות קריאות REST לAPI של תת-המערכת "השרת".  
     שימוש ב- **REST** הוא משום שזו השיטה הנוחה והנפוצה ביותר בתקשורת אינטרנטית כיום.
  3. הממשק מציג למשתמש את הנתונים המקושרים אתו בדף הבית.
  4. הממשק מאפשר למשתמש לעדכן ולהוסיף נתונים למערכת.
  5. הממשק מאפשר למשתמש לצפות בלוגים הרלוונטיים עבורו.

! מדריך למשתמש באפליקציה מצורף כנספח לספר הפרויקט !

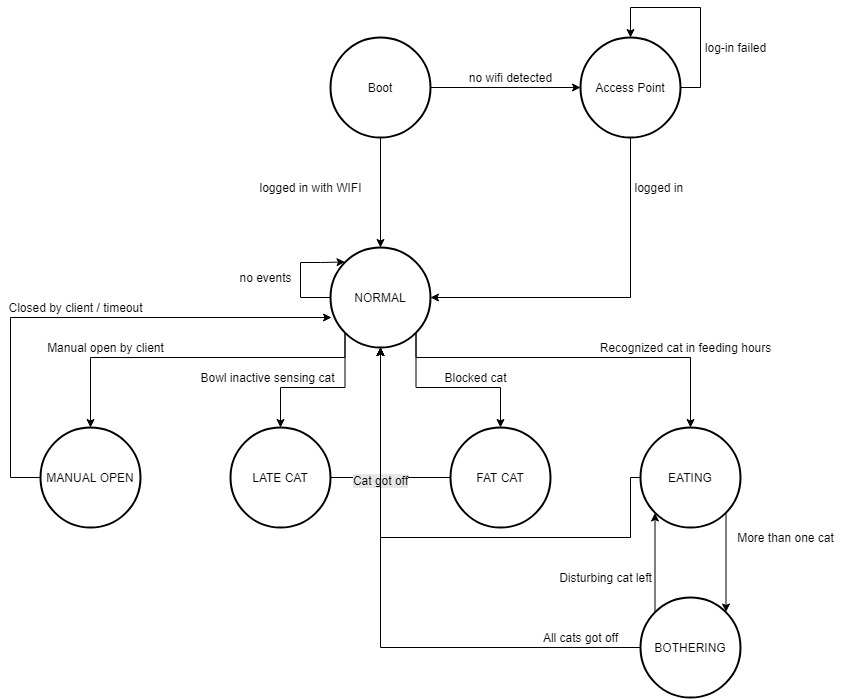


קערה חכמה :

1. אספקת המתח לבקר הקערה החכמה נעשה באמצעות חיבור  
    **micro USB**.
2. הקערה החכמה מורכבת ממספר חלקים:
   1. כרטיס ארדואינו – עליו רצה הקושחה המנהלת את שאר החלקים בתת המערכת.
   2. 4 חיישני משקל המחוברים למשטח עץ – חלק זה מעביר את נתוני השקילה לרכיב הארדואינו.
   3. מנוע המחובר לשני חצאי קערות – חלק זה הוא בעצם החוסם את הגישה לקערת האוכל. מקבל פקודת פתיחה או סגירה מרכיב הארדואינו.
3. רכיב הארדואינו כולל כרטיס WIFI המאפשר חיבור לרשת לוקאלית. (מדריך לחיבור הקערה לWIFI מקומי מצורף כנספח)



1. לקוד הקערה הוגדרו 6 מצבים:
   * 1. **Normal** – המערכת מבצעת שקילה באופן שוטף ומבקשת עדכון מהשרת כל 5~ שניות. הקערה נעולה.
     2. **Eating** – המערכת זיהתה חתול מוכר שבא לאכול בתוך שעות ההאכלה שלו. הקערה פתוחה.
     3. **Fat Cat** – המערכת זיהתה חתול זר או חתול מוכר שמנסה לאכול מחוץ לשעות ההאכלה שלו. הקערה נעולה.
     4. **Late cat** – המערכת זיהתה חתול מנסה לאכול מחוץ לשעות הפעילות של הקערה. הקערה נעולה.
     5. **Manual open** – המערכת עברה למצב ידני. הקערה פתוחה.
     6. **Bothering** – המערכת זיהתה חתול נוסף שעלה על המשקל בזמן שחתול אוכל. הקערה סגורה.

להלן תרשים המסביר את המעברים בין המצבים:

קיימים עוד שני מצבי עזר לצורך התרשים שהם:  
מצב Boot – המצב ההתחלתי של המערכת. איפוס המשקל, הגדרת המשתנים וכו'.  
מצב Access Point – מצב שבו המערכת מחפשת WIFI מקומי – מחכה למשתמש שיגדיר. הקערה פתוחה.

המעברים בין המצבים (מלבד Boot ו-Access Point) נעשים לפי שינויים בעלייה/ירידה במשקל שמזהה הקערה או לפי קריאות מתת-המערכת השרת.

1. קערה חכמה מבצעת קריאת **REST** לתת-מערכת השרת בכל 5~ שניות לצורך קבלת נתונים עדכניים עבורה. הנתונים כוללים:
   1. שעות הפעילות שלה.
   2. מערך המכיל את משקלי החתולים המקושרים אליה.
   3. מערך המכיל את שעות האכלת החתולים המקושרים אליה.
   4. מערך המכיל את שמות החתולים המקושרים אליה.
   5. שדה בוליאני המתאר את מצב הקערה (**ידני** או **אוטומטי**).
   6. שדה בוליאני המתאר האם יש בקשה לשקילה מיידית מהקערה.
2. קריאות מתת-המערכת "השרת" לקערה חכמה מבוצעות על ידי שינוי שדות בתגובת עדכון הנתונים הרגיל:
   1. שינוי למצב "**ידני**" או "**אוטומטי**" - נעשה על ידי שינוי שדה **method** למצב הרצוי.
   2. בקשת שקילה מיידית – שינוי שדה ה-**weight** ל-TRUE.
3. לצורך שליחת לוג הקערה החכמה תשלח קריאת **REST** לתת-המערכת "השרת" עם תיאור האירוע ופרטי השעה והתאריך שהאירוע התרחש.
4. המערכת משתמשת בספרייה שפונה ל-**NTP** (Network Time Protocol) ולפיו קובעת את השעה ואת התאריך.

השרת:

1. את השרת החלטנו לכתוב בסביבת **NodeJS**.   
   הסביבה מהירה מאוד ומאפשרת בקלות יחסית ליצור קוד א-סינכרוני. כמו כן, מכיוון שזהו קוד-פתוח, קיימת תמיכה רבה, ותמיד תחת עדכון בהתאם לדרישות שיש בשוק העבודה.   
   עובדת עם ספרית **npm** שזו ספריה נוחה מאוד לניהול מודלים. לא במקרה, זו אחת הסביבות הנפוצות ביותר כיום בשוק העבודה.
2. השרת יושב על **פלטפורמת ענן** **Heroku** הזמינה 24/7.
3. השרת מספקAPI עבור ממשקי המשתמש ועבור הקערות החכמות.
4. השרת מתקשר עם מסד הנתונים מסוג **MongoDB** בעזרת ספריית **mongoose**.
5. השרת מגשר בין כל רכיבי המערכת.
6. על מנת לעדכן את ממשק המשתמש ללא קריאה מצידו – נעשה שימוש בספריית **socket.io**.  
   זו יוצרת מעין צינור האזנה משני הכיוונים לקריאות מסוימות.

מסד הנתונים**:**

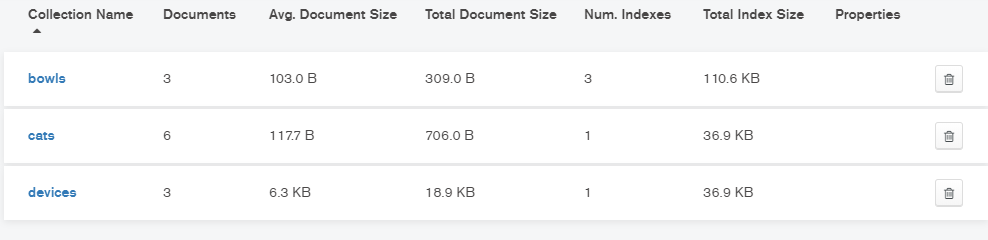
1. מסד הנתונים הוא לא-רלציוני מסוג **MongoDB**.

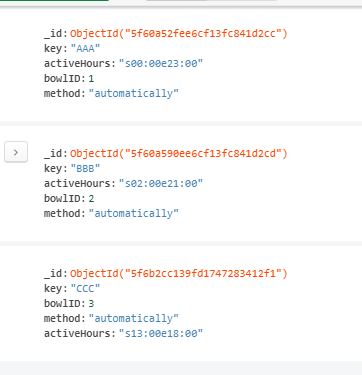
למרות שכל השדות בבסיס הנתונים שתכננו היו ידועים לנו מראש, החלטנו להשתמש במסד נתונים לא-רלציוני מכמה סיבות עיקריות:

1. דינאמי יותר להוספת שדות. (**JSON**)
2. עובד הרבה עם **אפליקציות Mobile** ו-**Web**.
3. ניסיון עם מסד נתונים רלציוני מקורסים קודמים ורצון להתנסות.
4. היקף השימוש של הטכנולוגיה בשוק העבודה.

בחרנו להשתמש ב-**MongoDB** מתוך רצון להיכרות עם אחד משירותי מסדי נתונים המוכרים והנפוצים ביותר בשוק כיום.

1. מורכב מ-3 **Collections**:
   * 1. Cats
     2. Bowls
     3. Devices



1. ה-Documents בכל אחד מה-collections בנויים בצורה הבאה:
   * 1. Cats’ documents:
        1. \_id
        2. Name
        3. Weight
        4. Sex
        5. feedingHours
        6. bowID
     2. bowls’ documents:
        1. \_id
        2. bowlID
        3. key
        4. activeHours
        5. method
     3. Devices’ documents:
        1. \_id
        2. deviceID
        3. logs[]:
           1. date
           2. info
        4. bowls[]:
           1. bowlID
           2. name
2. מסד הנתונים יושב על **פלטפורמת הענן** של **mongoDB** הנקראת **mongoDB Atlas**.

סביבת עבודה ושפות

סביבות עבודה:

* חמרה:
  + בקר ארדואינו **ESP8266** (הכולל חיבור WiFi)
  + 4 רכיבי **Load Cell** לקליטת מתח (strain)
  + ממיר/מגבר **HX711**
  + מנוע **Step Motor 28BYJ-48**
  + ספק מתח **micro USB**
* עורכים:
  + **Arduino Software (IDE)** - עבודה מול הארדואינו נעשתה על ידי תוכנה ייעודית לרכיב זה.
  + **Visual Studio Code**
  + Sublime
* תוכנת בדיקת שרת:
  + **Postman**
* מסד נתונים:
  + **MongoDB**
* תוכנת GUI למסד הנתונים:
  + **MongoDB compass**
* פלטפורמות ענן:
  + **Heroku** – עבור השרת
  + **MongoDB Atlas** - עבור מסד הנתונים

שפות:

* **C++** - עבור הארדואינו
* **React Native** – עבור האפליקציה fAtcAt
* **NodeJS** – עבור השרת

שלבי ביצוע

1. קנייה ובניית רכיבי חומרה להכנת הקערה והמשקל.

התחלנו בהזמנת ערכה בסיסית של **ארדואינו** יחד עם חיישני המשקל (**Load cells**).  
הבחירה בארדואינו הייתה אינטואיטיבית, משום שזהו פתרון נפוץ מאד ליישום אבות-טיפוס למוצרים בסגנון הקערה שלנו – הן מבחינת סיבוכיות המוצר והן מבחינה כלכלית.  
עקב זמן משלוח גדול (שרק החמיר לאחר התפרצות המגיפה בהזמנות מאוחרות יותר) לוח הזמנים של הפרויקט נדחק.

1. בנייה ראשונית של המוצר (ללא קוד חכם, אלא רק בדיקת חיבור).

תחילה בדקנו כי הרכיבים והלוגיקה מתפקדים. בפועל, שלב זה ארך זמן גדול מהמצופה עקב תקלות ייצור ברכיבים עצמם. בגלל חוסר ידע בתחום האלקטרוניקה, לקח לנו זמן רב להבין שהרכיבים תקולים, ושהבעיות שנתקלנו בהן אינן בגלל תוכנה. גם הבניה עצמה הייתה מסורבלת עקב חוסר בעזרים אלקטרונים, ציוד הרכבה וכמובן חוסר ידע. לאחר שהשגנו מספר כלים ורכיבים חלופיים (שגם הגיעו לאחר זמן רב) הצלחנו ליצור דגם ראשוני של משקל ששולט על שער הנפתח ונסגר בעזרת מנוע, בהתאם למשקל הנוכחי.

A picture containing table

Description automatically generated

1. קידוד ראשוני של רכיב הארדואינו.

מכיוון שזו שפה המבוססת על C++, שלב זה לא היה קשה למדי. עם זאת, השילוב של ספריות מתאימות לשליטה על הרכיבים הצריכו הרבה ניסוי וטעייה.

1. סקיצה ראשונית של האפליקציה.  
   בשלב זה יצרנו רישומים ראשוניים של עמודי האפליקציה – מסך בית, מסכי עריכת נתונים ומסך לוגים.  
   לאחר מכן התחלנו לעבוד על הקוד והבחירה בשפה נפוצה כל כך הייתה לנו לעזר – שכן יש תמיכה רחבה של קהילת ה- **React Native**.
2. הקמת השרת.  
   השלב הראשון היה לדעת אילו קריאות API בסיסיות אנו צריכים הן לצרכי ממשק המשתמש והקערה החכמה והן לצרכי בדיקת המצב הנוכחי של השרת.  
   לצרכי בדיקת קריאות לשרת נעזרנו בתוכנת **POSTMAN** - שיוצרת קריאות ובכך ניתן לנתח את השליחה והתשובה המתקבלת בצורה מבוקרת יותר.
3. חיבור האפליקציה והרכיב הארדואינו עם השרת (מעבר ל-**Heroku**).

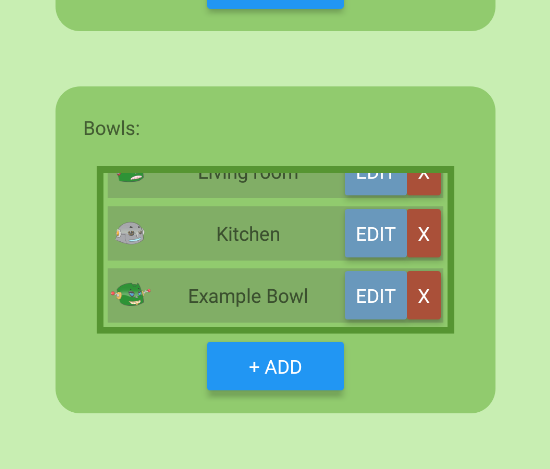
לאחר מספר ניסיונות עם רכיב נפרד לבקר לקליטת WiFi, הגענו למסקנה כי מוטב להשתמש בבקר חדש מסוג **NodeMCU** הכולל רכיב WiFi מובנה. צעד זה הצריך שכתוב קל של הקוד ומספר שינויים באופן החיבור של הרכיבים (עקב שינויים בהספק החשמלי). לדוגמה, המנוע

לאחר מכן, בעזרת ספריה ייעודית, יכולנו בקלות להתחבר לרשת ומשם לשרת באמצעות קריאות **REST API**.

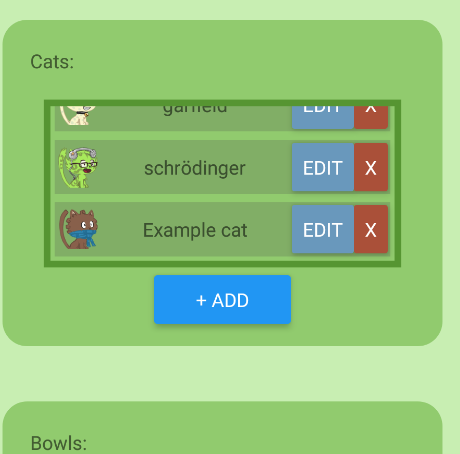
הרצון לחבר את הארדואינו לשרת דרש מעבר למחשב חיצוני שיהיה נגיש 24/7. לכן בחרנו ב-**Heroku** המספק שירותי ענן (בחינם) ועובד עם מגוון רחב של סביבות עבודה. לצורך ייעול תהליך העדכון, בחרנו לקשר את **Heroku** ל-**repository** שלנו ב-**github**. כך לאחר כל עדכון של הקוד יכולנו מיידית לבדוק מול השרת בפועל.

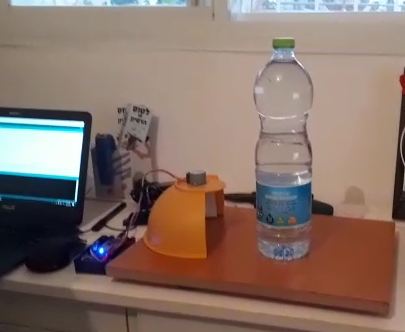
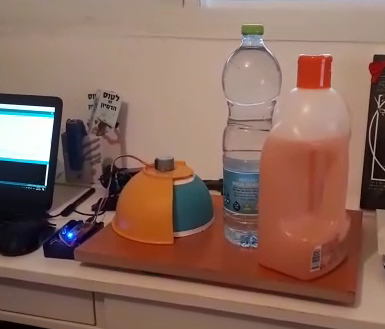
1. לאחר שימוש ב-**DB** מקומי, עברנו לשימוש ב-**MongoDB**.  
   בהתחלה הזנו נתונים ידנית בעזרת התוכנה **MongoDB Compass** , המציגה בממשק **GUI** את מסד הנתונים.  
   לאחר מכן בדקנו את השפעות קריאות השרת על מסד הנתונים.  
   ולבסוף בעזרת האפליקציה.
2. עבודה על קוד רכיב הארדואינו לפי נתונים פיקטיביים (פארסינג לנתונים ועבודה לפיהם).

לצורך בדיקת החומרה והתכנה, בדקנו בעזרת נתונים פיקטיביים שהלוגיקה מתפקדת כראוי ושהתקשורת בין השרת למשקל תקינה.

1. בדיקת המערכת.

הרצנו מספר בדיקות:

* הוספת קערה
* הוספת חתול
* עריכת קערה
* עריכת חתול
* הסרת קערה
* הסרת חתול
* בקשת משקל
* צפייה בלוגים
* עדכון לוגים
* חתול טוב על משקל
* חתול שמן על משקל
* חתול מפריע לחתול לאכול
* פתיחה ידנית
* סגירה ידנית
* סגירה אוטומטית לפי טיימר
* שעות פעילות קערה



ברצף התמונות הנ"ל (הנע משמאל לימין) מוצגת בדיקה של הפרעה לחתול.  
בתמונה הראשונה נראה החתול אוכל מהקערה הפתוחה.  
בתמונה השניה חתול אחר מפריע לחתול הראשון לאכול.  
בתמונה השלישית, כתוצאה מההפרעה, הקערה נסגרת.

1. בדיקת נתונים אמיתיים:  
   לצערנו, כאשר הגענו לשלב הבדיקות עם חתולים אמיתיים גילינו שהקערות קטנות מדי ולא מאפשרות באמת אכילה. לכן נשארנו בשלב הבדיקות בעזרת בקבוקים. עם זאת, הקערה הסגורה אכן מנעה גישה למזון בפנים.

סיכום פרויקט

בפרויקט זה יצרנו מערכת האכלה לחיות מחמד, שנועדה לספק לבעליהן אפשרות נוחה וקלה לנהל מספר קערות האכלה דרך ממשק **Mobile**.

כאשר בחרנו ליצור פרויקט זה המטרה העיקרית שלנו הייתה הרכבת מוצר נדרש.  
כמובן היינו בעלי ידע מאד מועט על תהליכי בניית מוצר.  
השלבים אומנם היו לנו מוגדרים די בבירור מבחינה כרונולוגית, אך חוסר הניסיון הפרקטי שלנו שמר לנו הרבה הפתעות ועיכובים למהלך העבודה.

השילוב בין הרכבת רכיבי חומרה ייעודיים למטרה, בניית ממשק משתמש וחיבור המערכת באמצעות שרת נתנו לנו תמונה גדולה יותר על העולם הטכנולוגי, בדגש על עולם ה-**IoT**.

בפרויקט זה נתקלנו בהרבה מאד קשיים, הן טכנולוגיים והן לוגיסטיים.  
השימוש בטכנולוגיות זרות לנו היה לא פעם מתסכל. גם בעיות עם חומרה תקולה או פשוט לא תואמת היוו מכשול מספר פעמים לאורך העשייה. וכמובן שהמצב הגלובלי לא תרם לצורכי מפגש על העבודה והשגת רכיבים.

מצד שני, עמדנו במטרה שהצבנו לפנינו בתחילת הפרויקט, ולמדנו על טכנולוגיות רבות, כלים שימושיים ודרכי עבודה.

במבט לאחור, הפרויקט תרם ל"השכלה בעולם הטכנולוגיה" שלנו רבות:

ברמת התכנון, העבודה על מערכת שלמה, הצבת יעדים, עבודה בצוות והתחשבות בצרכים לוגיסטיים.

ברמת התוכנה, העבודה במודל של שרת-לקוח, שימוש במספר פלטפורמות ענן, שימוש בספריות חיצוניות לצרכינו, ניסיון עם תוכנות לצרכי בדיקה ועורכי טקסט שונים.

ברמת החומרה, ההתעסקות עם האלקטרוניקה, הרבה ניסוי וטעיה בבניית החומרה של המערכת.

וכמובן, החשיפה לכלים נוספים ורבים שלבסוף לא השתמשנו בהם לצורכי הפרויקט תרמה לנו לידע רחב יותר.

למרות שהמערכת הפיזית לא מתאימה בגודלה לחתולים בפועל, עדיין ניתן לראות בה בתור מודל ראשוני מוצלח, וע"י החלפה של הקערות (לזוג קערות גדולות יותר) או שינוי המנגנון – ניתן לקבל מערכת שתתאים גם לחתולים גדולים יותר. בכל מקרה, אנו מסתכלים על המערכת כ**בחינת היתכנות** (**Proof of Concept**), שניתן ליישם למערכת שלמה.

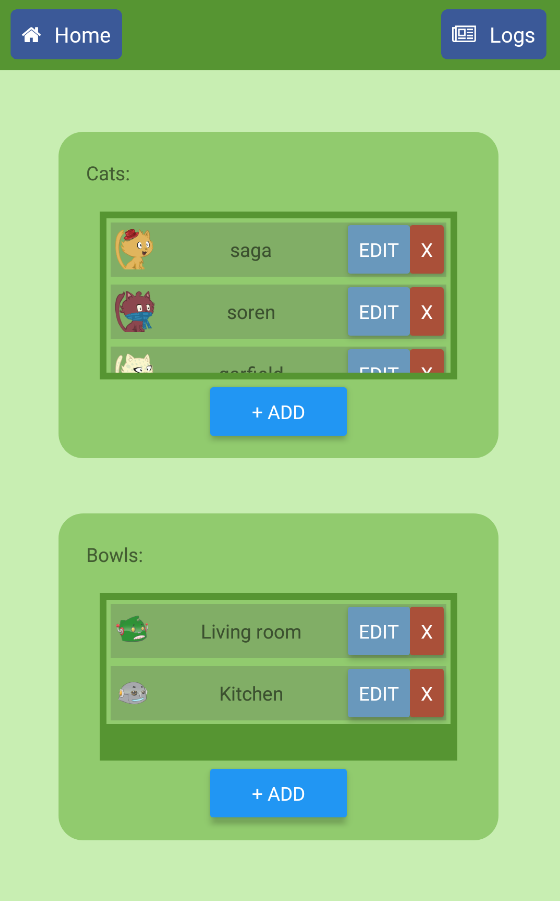
לסיכום, אנו שמחים שבחרנו לבצע פרויקט זה – הן מבחינת הידע שרכשנו והן מבחינת התוצאות.

נספחים

נספח א' - מדריך למשתמש באפליקציית fATcAT

האפליקציה נועדה למשתמש לנהל את השימוש בקערות ה-**fATcAT**.  
המשתמש נקבע לפי המכשיר (זהו סימן ההיכר מול השרת).

בעזרת האפליקציה ניתן להתחבר לקערות ולשייך אליהן חתולים.  
לכל קערה ניתן להגדיר שעות פעילות (שרק בהן תוכל להיפתח) וכינוי שלפיו תוצג הקערה במכשיר המשתמש.  
ולכל חתול יש להגדיר שם, מין, משקל ושעות האכלה.

**מסך הבית** :

**חלון ניהול חתולים**

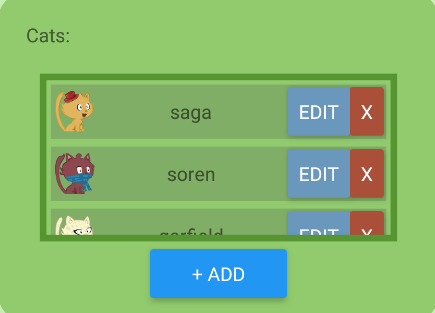
**חלון ניהול קערות**

**תפריט ניווט**

**תפריט ניווט:**

* צפייה בלוגים מהקערות
* חזרה למסך הבית

**חלון ניהול חתולים**:



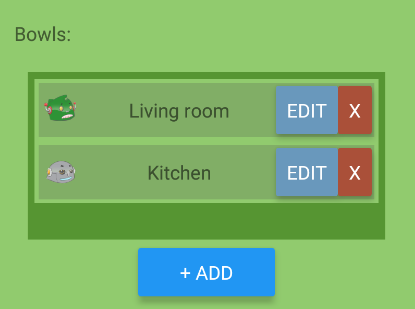
**עריכת חתול**

**הסרת חתול**

**הוספת חתול קערה**

* עריכת חתול – שינוי נתוני החתול.
* הסרת חתול – הסרת חתול ממאגר הנתונים של המערכת.
* הוספת חתול – הוספת חתול חדש למאגר הנתונים של המערכת.

חלון ניהול קערות:



**הסרת קערה**

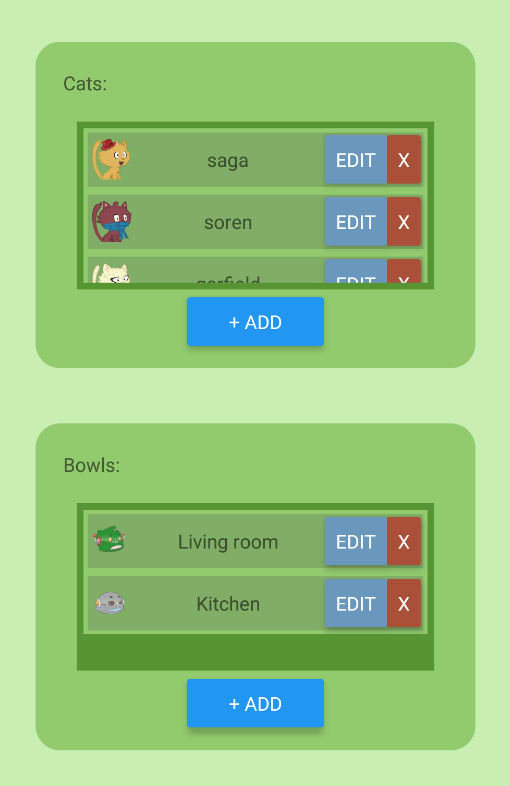
**הוספת קערה קערה**

**פתיחת קערה**

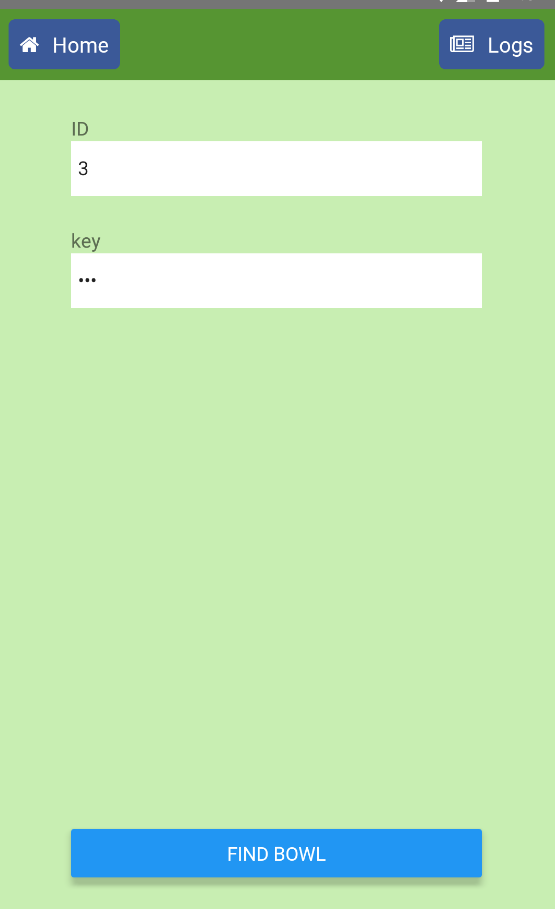
**עריכת קערה**

* עריכת קערה – שינוי שם הקערה או שעות הפעילות שלה.
* הסרת קערה – הסרת קערה ממאגר הקערות של המכשיר.
* הוספת קערה – הוספת קערה חדשה למאגר המכשיר.
* פתיחת קערה – לחיצה על האייקון תפתח את הקערה ותשאיר אותה במצב זה (ללא תלות במשקל הנמדד או בשעות הפעילות),עד לאחד משני המצבים:
  + המשתמש ילחץ שוב
  + יחלפו 10 דקות

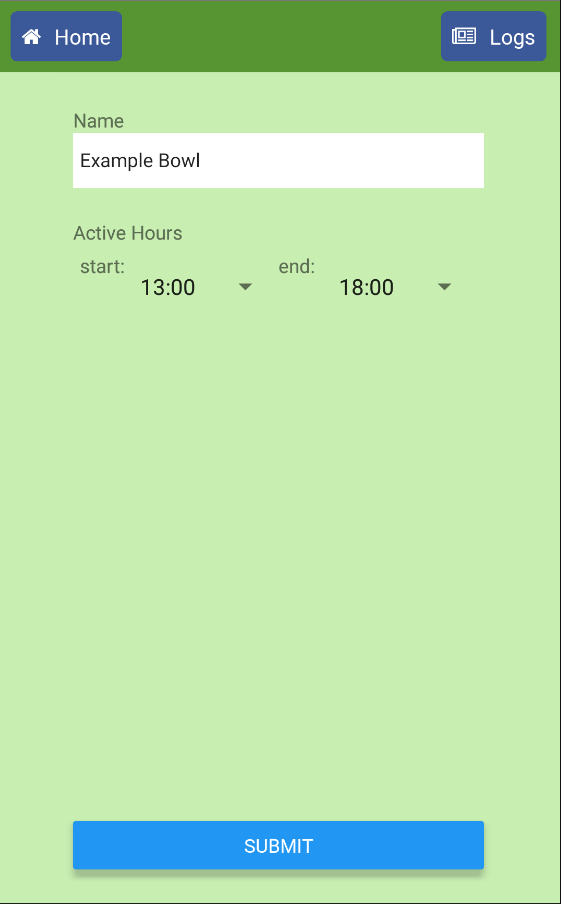
**תהליך הוספת קערה**



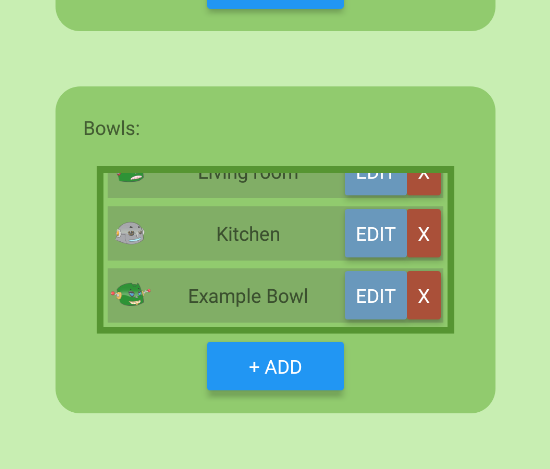
1. נלחץ על הוספת קערה



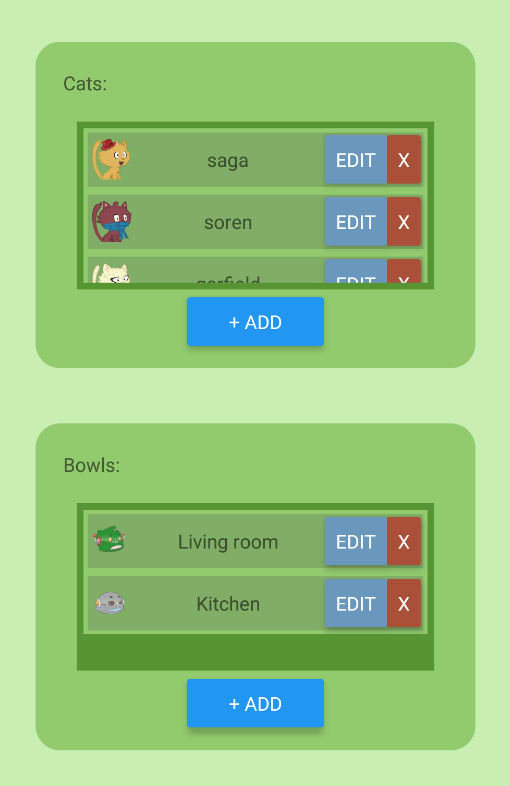
1. בחלון שנפתח נמלא את השדות ID ו-KEY   
   לפי הנתונים שניתנו עם הקערה.  
     
   למשל:  
   ID:3  
   KEY:CCC  
     
   ולאחר מכן נלחץ על FIND BOWL



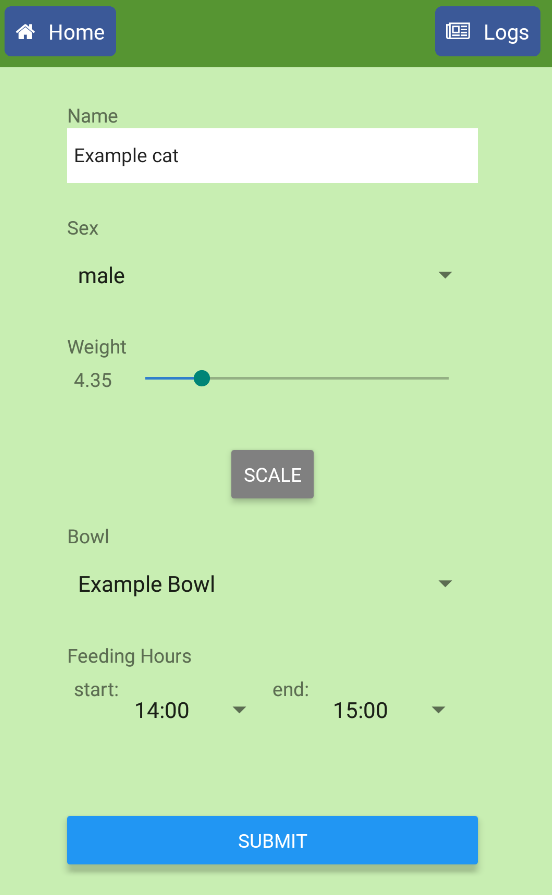
1. בשלב זה נמלא את נתוני הקערה – כינוי ושעות הפעילות  
     
   ולאחר מכן נלחץ על SUBMIT



1. בשלב זה ננותב למסך הבית.  
     
     
     
   כאן נוכל לראות את הקערה החדשה שהוספנו

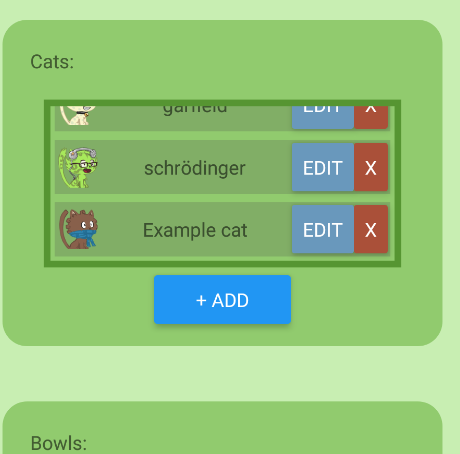
**תהליך הוספת חתול**

1. במסך הבית נבחר להוסיף חתול

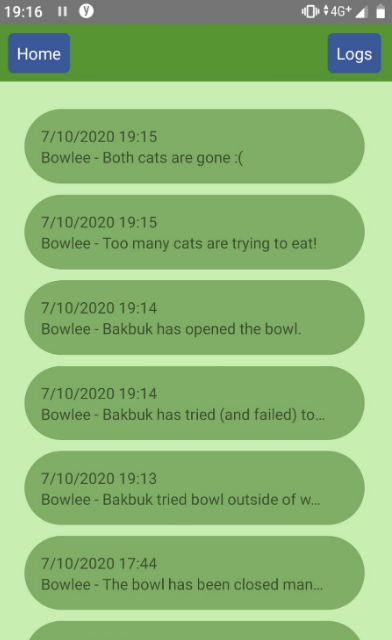


1. בשלב זה נמלא את המידע על החתול  
     
     
   במידה ונרצה למדוד את החתול שעל המשקל כעת,  
   נוכל להשתמש בכפתור הSCALE

ולבחור מאיזו קערה לקבל משקל נוכחי  
   
  
   
 לאחר שנסיים נלחץ על SUBMIT



1. בשלב זה ננותב לדף הבית, שם נוכל לראות את החתול החדש



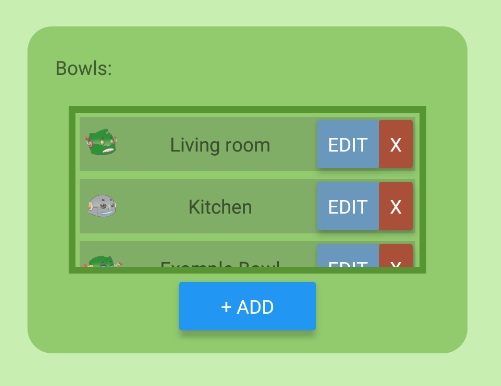
**צפייה בלוגים**:

ניתן לצפות ב-50 הלוגים האחרונים שהתקבלו מהקערות המקושרות למכשיר.  
  
סוגי הלוגים:

* חתול התחיל לאכול
* חתול סיים לאכול
* חתול שמן ניסה לאכול
* חתול ניסה להפריע לחתול אוכל
* פתיחה ידנית
* סגירה ידנית
* המערכת הודלקה

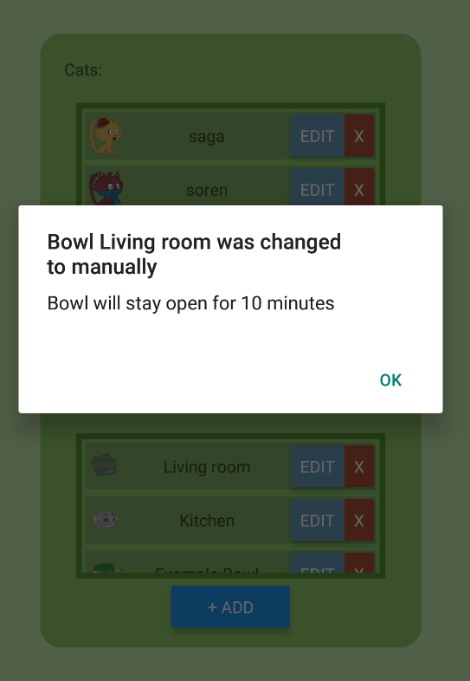
לחיצה על לוג ארוך תפתח אותו לצפייה

גלילת העמוד כלפי מטה תרענן את הלוגים.

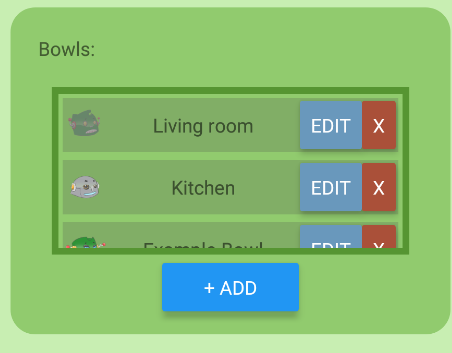
**שינוי מצב הקערה**

ברירת המחדל היא שהקערה במצב "אוטומטי" – כלומר, נפתחת ונסגרת לפי הנתונים שהוזנו לה (משקלי ושעות החתולים ושעות הפעילות של הקערה עצמה).

על מנת לשנות את מצב הקערה יש ללחוץ על התמונה שלה.

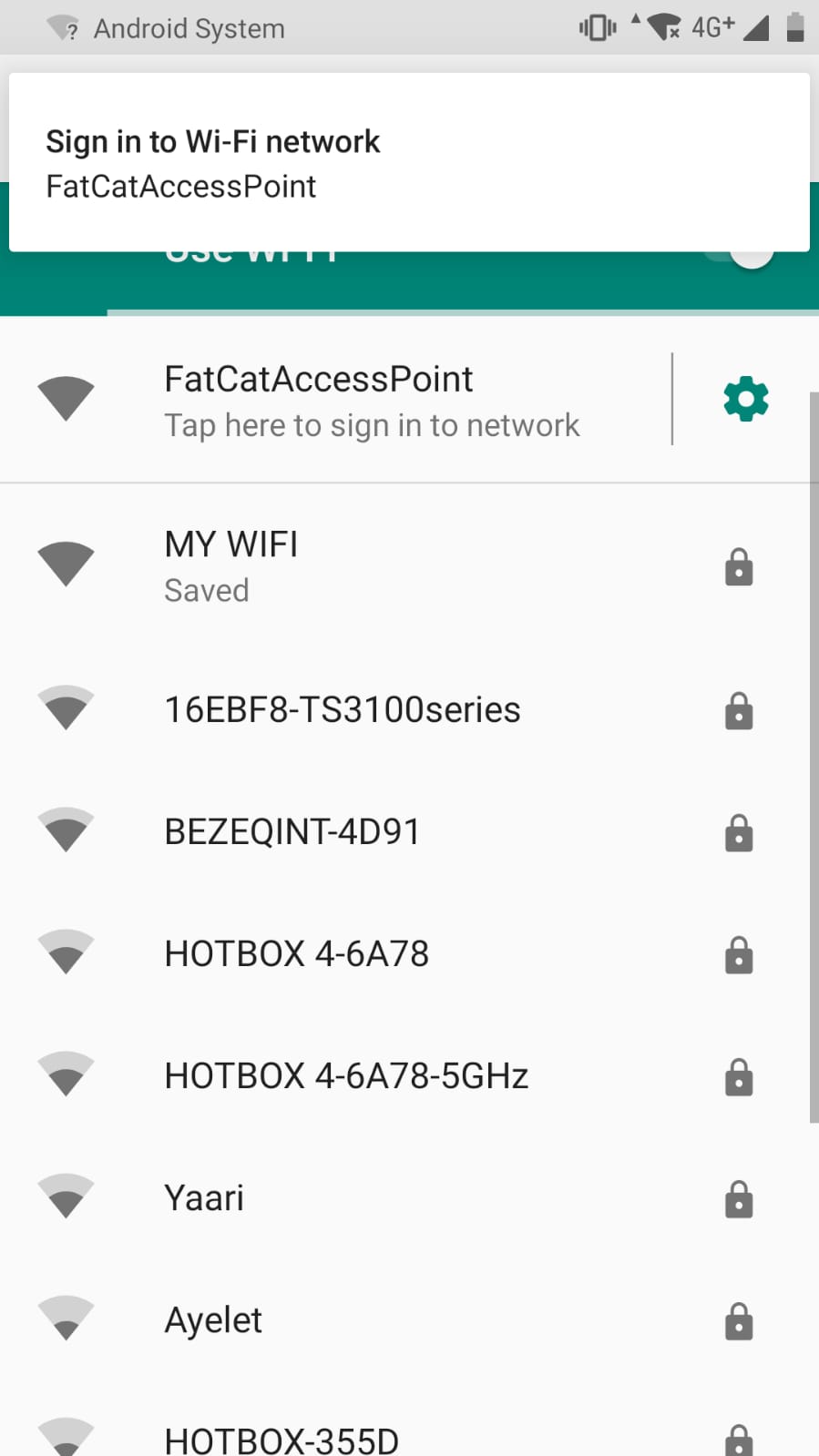


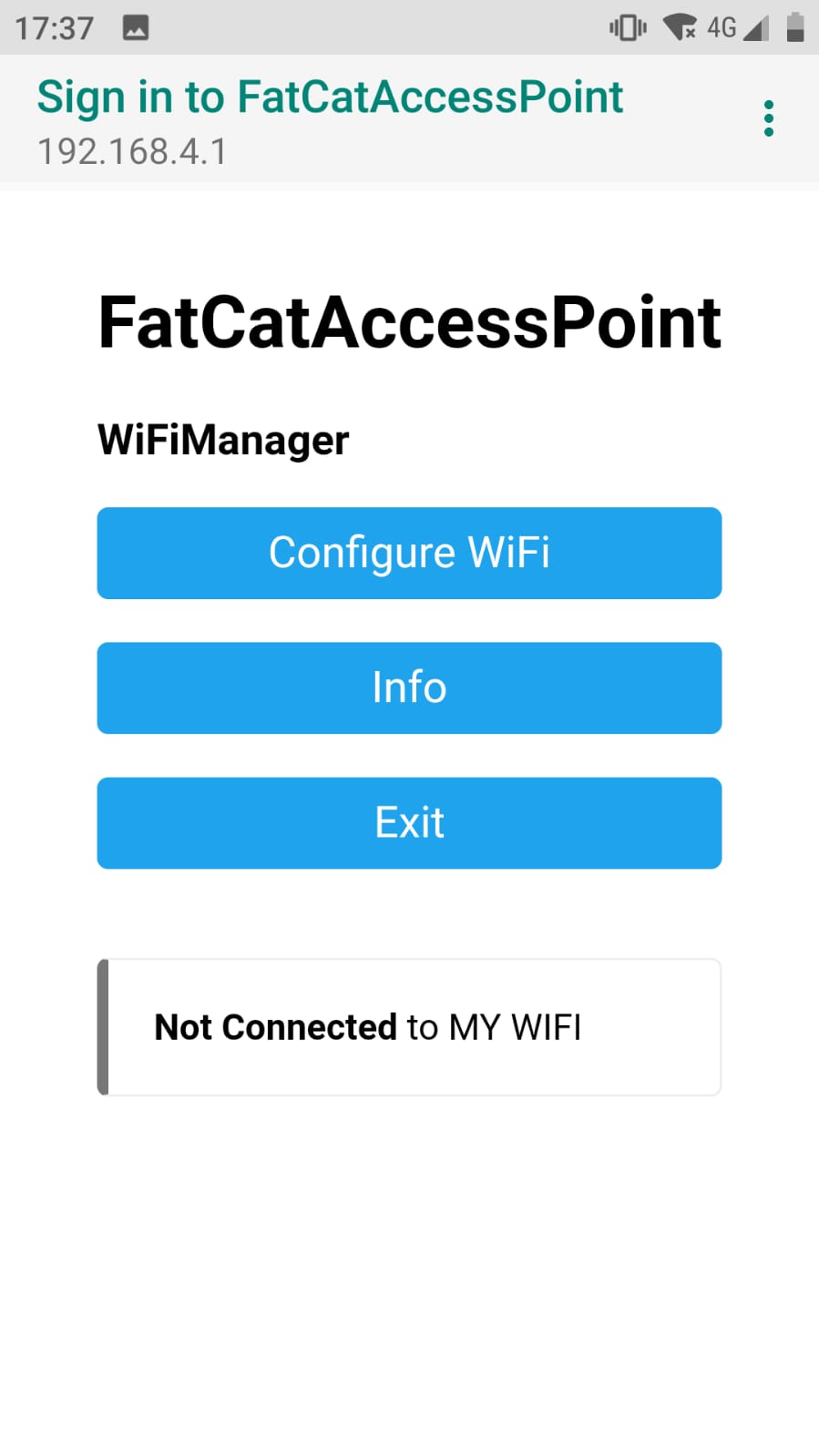
לאחר מכן תוצג בפנינו הההודעה הבאה

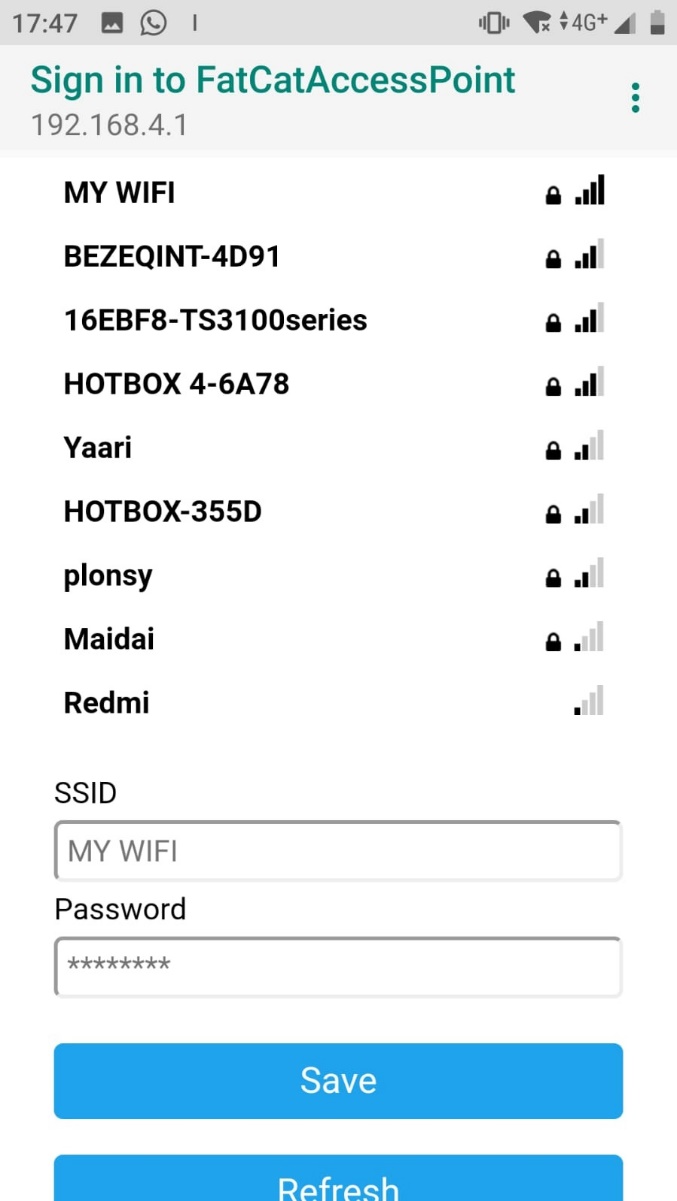


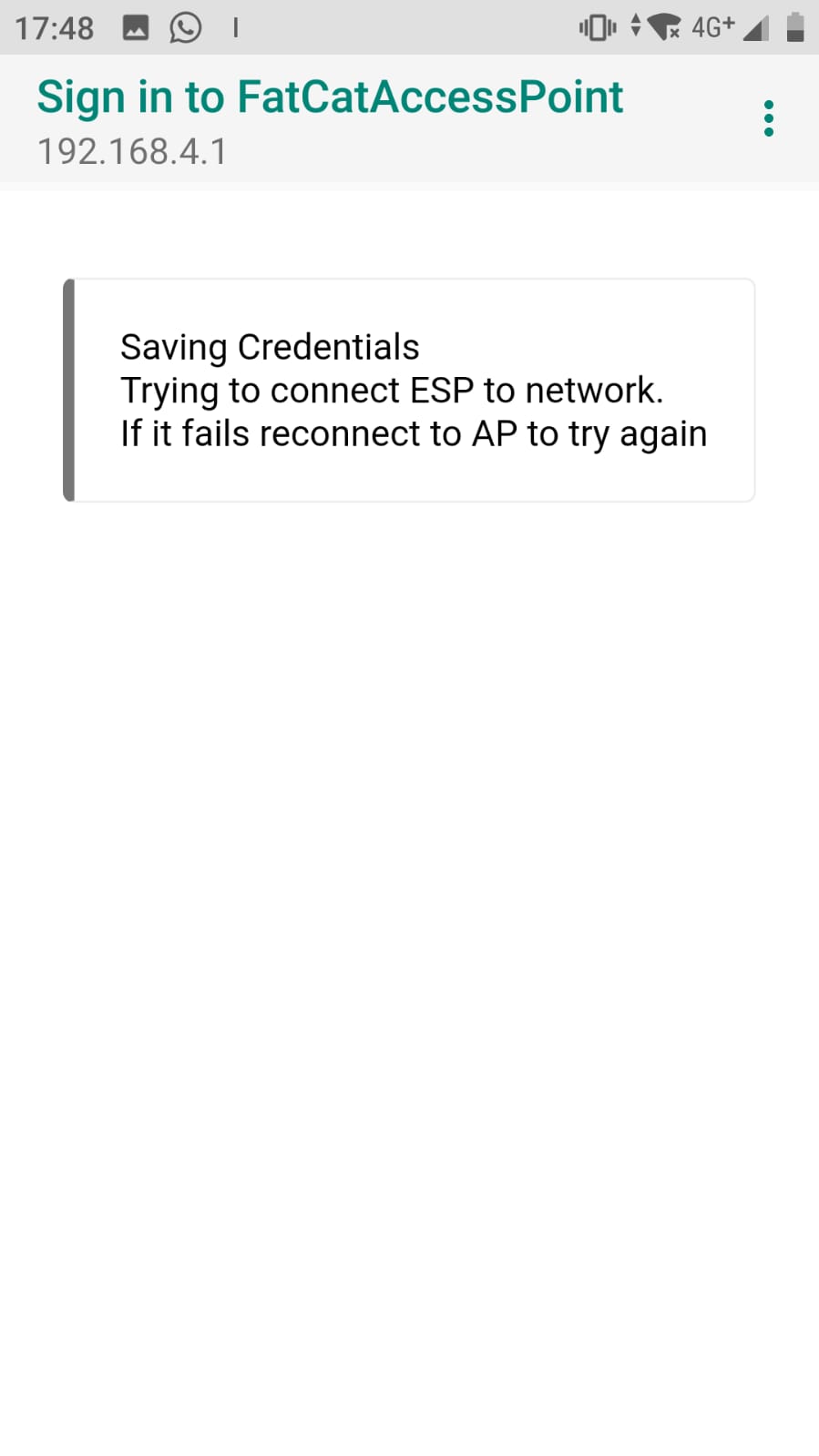
כאשר הקערה במצב "ידני" נראה את תמונת הקערה בגוון אפור

לאחר 20 דקות, במידה ואף משתמש המקושר לקערה זו לא החזיר את הקערה למצב "אוטומטי", הקערה תוחזר למצב זה אוטומטית.

נספח ב' - חיבור ראשוני של המערכת לרשת WIFI מקומית

1. מספר שניות לאחר חיבור המשקל לחשמל, המערכת תפתח את הקערה ותחפש רשת אינטרנט אלחוטית שהיא מזהה. אם המערכת לא מצליחה להתחבר או שהיא אינה מזהה רשתות כלל, המערכת תיצור נקודת גישה ממנה ניתן להגדיר את הרשת הביתית. במידה והצליחה להתחבר, תדלג לשלב 4.
2. לאחר חיבור לרשת באמצעות טלפון חכם / טאבלט / מחשב, המשתמש יתבקש להתחבר לנקודת הגישה. אם לא מופיעה בקשה כזו, יוכל להיכנס בדפדפן לכתובת 192.168.4.1.



1. נכנסים להגדרת רשת ומכניסים את פרטי הרשת הביתית (ניתן לבחור את שם הרשת מבין רשימה של הרשתות שהטלפון מזהה).
2. לאחר ההתחברות, הקערה תיסגר (ותעבור למצב "חכם") והרשת תישמר בזיכרון של המערכת ותתחבר אוטומטית החל מההפעלה הבאה.