**מערכת ה QR:**

* בתחילת כל יום ניגש לטבלה עם הקורסים
* לכל קורס שקיים היום מייצר לו QR ושולח אותו במייל לכתובת מוגדרת מראש (נניח info2)
* הQR גם שואב מזהה קורס ושולח אותו דרך הלינק לקוד ככה שההרשמה כבר תהיה מקושרת לאותו קורס (לדוגמה [https://example.com/signin?**courseId=501301**](https://example.com/signin?courseId=501301))
* מערכת שבודקת את הלינק של הQR לאחר היצירה שלו כדי לוודא שהוא תקין – ולהכין חדש במידה ולא.

**טבלאות מידע:**

**כולן יושבות בגוגל דרייב בפורמט של google sheets תחת אותו משתמש.**

* טבלה מזהה קורסים:
  + לכל קורס מתאימה מספר מזהה קורס (סעיף תקציבי יהיה הכי הגיוני)
  + הטבלה מכילה שם קורס, ותאריכים – משם הQR ישלוף מידע וידע האם לייצר את עצמו.
* טבלת נרשמים לכל קורס:
  + כדי למצוא אותה, נערוך אותה ככה שבשם שלה יופיע המזהה קורס, ושם נרשום מי שמתחבר

**מערכת רישום וואלידציה:**

* כאשר בנאדם סורק את הקוד:
  + דבר ראשון ניקח את המזהה קורס שהQR שלח ואת התאריך של היום, ונבדוק בטבלת קורסים שהוא אכן קורה היום. אם לא, נשלח הודעה שאי אפשר להירשם לקורס הזה היום. אחרת:
  + **אם זאת פעם ראשונה (המערכת תדע את זה מכיוון שאין לו מזהה ID בטלפון):**
  + הוא יקבל דף הרשמה שמציג שדות למלא.
  + לאחר שהוא מילא אותם, המערכת תיגש לקובץ טבלת הנרשמים לאותו קורס עם המזהה קורס( שהקוד קיבל דרך הQR).
    - תחפש את המספר טלפון
      * אם מצאה תוודא את זה מול המייל
      * אם מצאה תוודא את זה מול השם.
      * אם השם נמצא נסמן אותו
    - **אם מצאנו אותו:**
      * נייצר לו UUID
      * נשמור אותו בlocal storage
      * נוסיף למסד נתונים את המזהה הזה ונבנה לו טבלה עם הפרטים, אחד מהשדות יהיה רשימה של תאריכי רישום.
    - **אם לא מצאנו אותו:**
      * נחזיר הודעה שהפרטים לא נכונים נא למלא שוב
  + **אם לא זאת לא פעם ראשונה (כלומר יש לו כבר מזהה ID בטלפון[אולי יותר מאחד]):**
    - המערכת תעבור על כל אחד מה ID שיש לו בטלפון (למקרה שהוא רשום לכמה קורסים)
    - תיגש לטבלה של הקורס ותחפש את הטלפון, את המייל ואת השם ותנסה למצוא אותו.
    - **אם מצאנו אותו:**
      * נסמן אותו במקום המתאים ונשלח הודעה שהוא נרשם בהצלחה
    - **אם לא מצאנו אותו:**
      * נשלח הודעה שהפרטים לא נכונים, נאפשר להירשם מחדש.
        + אם הגענו לכאן זה אחד משני מקרים:

הוא מנסה להתחבר לקורס נוסף שלו, ולכן נציג את האפשרות להירשם

הוא מנסה להתחבר לקורס שהוא לא רשום אליו וככה אנחנו נמנעים מאנשים סתם לסרוק את הQR ולהירשם. **(אם הוא ינסה להירשם זה יביא אותו לכאן שוב כי הוא לא קיים בטבלאות)**

* + - * + האופציה לרישום נוסף שולחת את הבנאדם לרישום כמו של פעם ראשונה, עם תפריט בחירה לאיזה קורס. הקורס שהוא כבר רשום אליו לא יופיע (זה ימנע מאנשים לרשום את חברים שלהם וגם מכפילויות מוזרות במסד נתונים)

**תקשורת עם הgoogle drive:**

* חיפוש קורס ספציפי
  + אם בשם הקורס יהיה את המזהה SAP שלו זה יהיה אידיאלי
  + במידה ולא נמצא הקורס, נבנה EXCEL חיצוני ונשלח אותו במייל.
* גישה לgoogle sheets לקריאה ולעריכה
  + זה אומר שכל הקורסים כתובים בgoogle sheets ולא בסוג אחר של קובץ,
  + אם קורס כתוב באקסל רגיל ועולה כקובץ אקסל לדרייב זה לא יעבוד

**מערכת התראות: $$**

* חיבור למייל ארגוני לשליחת הודעות (נניח info2) של תקלה, או התראה שמשתמש נרשם.
  + Log שמעדכן בכל יום את כל הפעילות שהייתה.
* התראה למייל של המשתתף שנרשם על הרשמה מוצלחת.
* מצריך שימוש בשרת לשליחת אימיילים בתשלום. יש אפשרויות חינמיות אבל הן מגבילות את כמות המיילים שניתן לשלוח ביום, עלולות להיתקע ולקחת זמן ואין תמיכה.

אופציונלי - מערכת ניהול **admin**:

* דף אדמין שמאפשר גישה ועריכה של הנתונים מהDataBase
  + מצריך התחברות מאובטחת באופן הבא:

### **תהליך התחברות מאובטחת עם express session:**

1. התחברות עם שם משתמש וסיסמא
2. המערכת מבצעת חיפוש המשתמש במסד נתונים
3. השוואת סיסמאות (הסיסמא שמאוחסנת במסד נתונים מוצפנת עם bcrypt)

#### ****אם הפרטים נכונים:****

#### השרת יוצר סשן ומזהה סשן ייחודי (session ID)

#### השרת שומר את הסשן אצלו בזיכרון.

#### נשלחת **עוגייה (Cookie)** לדפדפן של המשתמש עם ה־Session ID.

#### ****החל מאותו רגע, כל בקשה מהדפדפן תכלול את העוגייה הזו****

#### במידה והדפדפן מוצא אותה – נותן גישה

#### אם לא – שולח לדף ההתחברות

#### ****בעת ביצוע התנתקות העוגיה נמחקת ולא ניתן לגשת שוב בלי להתחבר מחדש****

#### מאפייני אבטחה של העוגייה:

#### **Http only – העוגייה נשלחת רק דרך HTTPS **ובכך** מונע גישה זדונית לעוגיה (XSS)**

#### **SameSite** מגביל התקפות בין אתרים (CSRF).

#### ניתן לקבוע תוקף לפקיעת העוגייה אחרי זמן לא פעיל.

**מקרי קצה:**

* **בנאדם שרשום לשני קורסים במקביל:** 
  + נוסיף אופציה של רישום נוסף – היא עושה את כל תהליך הרישום והוואלידציה שוב (מצליבה מידע לראות שהוא אכן צריך להירשם) ומייצרת לו UUID נוסף לקורס הזה.
* **אם בנאדם משלים מפגשים בקורס:**
  + הוא חייב להופיע בטבלה של הקורס, ואז נמצא אותו ללא בעיה, הוא יכול פשוט להירשם בתהליך הרגיל.
* **מישהו מחליף טלפון או דפדפן (כלומר נמחק לו המזהה משתמש ID):**
  + במקרה הזה הוא יצטרך להירשם רגיל (הוא בעצם בתרחיש דומה של נרשם פעם ראשונה) ואם עבד לו עד עכשיו אין סיבה שלא יעבוד, כי הפרטים שלו מקושרים בטבלה.
  + המערכת תחליף במסד נתונים את המפתח שמקשר לכל המידע עם המזהה ID החדש שנוצר.
* **מישהו מחליף מספר טלפון:**
  + זה דומה למקרה הקודם, רק שכאן כדי שזה יעבוד הוא חייב לעדכן אותנו במספר החדש ושאנחנו נעדכן אותו בטבלה של הנרשמים: אחרת המערכת לא תדע לזהות שהוא צריך להירשם לקורס (מכיוון שזה התרחיש הכי נדיר, עליו נסתמך בעיקר כדי לזהות את הבנאדם)
    - חלופה אחרת תהיה להשתמש בתעודת זהות כדי לזהות אנשים אבל לא בטוח שנרצה לבקש מהם ת"ז.
* השם בטבלה לא תואם את השם שהוא נרשם איתו כאן:
  + אנחנו מסתמכים על המספר טלפון בעיקר לוואלידציה.

**הערות:**

* מידע צריך להישמר בdatabase ולא בזיכרון למקרה של קריסה.
* השירות של render כבר דואג לטפל בזה במידה והוא זה שקורס.
* לא נמחק מידע כאשר קורס נגמר, אבל כן נכניס אותו לארכיון בדאטא.

**בעיות אפשריות וטיפולן:**

**בעיה בקבצי אקסל עם המידע הפנימי שלנו:**

כדי לספק חוויית משתמש חלקה, חשוב שהקבצים שלנו בדרייב יהיו בפורמט קבוע. הקוד יתאים לפורמט הנוכחי, אך כל שינוי מבני עלול לגרום לתקלות.

אנחנו מזהים שני תרחישים עיקריים:

1. **סריקה ראשונית (ההרשמה לקורס):**

* אם הקובץ לא קיים, או שהמשתתף לא מופיע בו (או מופיע עם פרטים שגויים), המערכת לא תאשר את הרישום.
* במקרה כזה המשתמש יידרש להירשם ידנית, והמנהל יקבל הודעה על הצורך בתיקון הקובץ.

1. **סריקות חוזרות לאחר רישום מוצלח:**

* אם המשתמש כבר מחזיק מזהה (UUID) בטלפון אך הקובץ הוזז/שונה/נמחק – ההרשמה תיחשב כתקפה.
* עם זאת, תישלח למנהל הודעה שיש לעדכן את הקובץ בהתאם, כדי לשמר את רצף הנתונים.

**שימוש לרעה בלינק שמתקבל על ידי הQR (שיתוף לינק עם אנשים שלא נוכחים פיזית):**

1. **סריקת ה-QR מייצרת Token ייחודי וזמני** (Short-lived token) עבור כל סריקה.
2. ה-**Token מקושר ל-UUID** של המשתמש שסרק, ונשמר בצד השרת.
3. בעת הכניסה ללינק:
   * המערכת בודקת האם ה-UUID בדפדפן של המשתמש תואם לזה שה- token נוצר עבורו.
   * רק במקרה של התאמה, תתאפשר הרשמה לקורס.
4. משתמשים אחרים שינסו להיכנס עם אותו לינק אך עם UUID אחר – ייחסמו.
5. דוגמת לינק: [https://example.com/signin?courseId=501301**&token=abc123**](https://example.com/signin?courseId=501301&token=abc123)
6. הטוקן מתחלף כל כמה שניות ואני שומר על שניים: אחד ועל הקודם. כדי למנוע כניסה שגויה שאמורה לעבוד

חשוב לציין שבמידה ואין להם ID כי זאת ההרשמה הראשונה, אז הלינק ידע לשלוח אותם להרשמה.

יש כאן פרצה אפשרית אחת והיא שמשתתף יכנס לאחסון שלנו וימחק ידנית את ה ID מה שיאפשר לו לקבל גישה לאתר הרישום מחדש - מאפשר להירשם מרחוק (זה דורש זדון והבנה של ארכיטקטורת הקוד של המערכת – חשוב לציין את הפרצה הזאת, אבל אנחנו מאמינים שניתן להשאיר אותה בהתחשב בקהל אליו התוכנה מיועדת)

1. הגישה לGoogle Sheets נחסמת באופן זמני (כשל כלשהו בתקשורת)
   * שגיאה שהמערכת תדע לתפוס ולשלוח הודעה מתאימה שיש לנסות שנית בעוד מספר רגעים או לפנות לנציג.
   * באופן כללי המערכת תלויה לחלוטין בגישה תקינה לקבצים בגוגל דרייב, ושהקבצים כתובים בפורמט תקין:
     + כדי לחזק את נקודת החולשה הזו, המערכת תשמור עותק **מקומי מוצפן** של הקבצים הקריטיים (כגון רשימת הקורסים והנרשמים) במסד הנתונים.
2. מענה כללי לבעיות:
   * כאשר משתתף נתקל בבעיה שהמערכת לא תוכננה או לא מצליחה לטפל בה:
     1. קיים נציג תפעול שניתן לפנות אליו, יש לו גישה אל הקבצים וניתן לעשות בדיקה ידנית
     2. במידה וייתכן תרחיש שבו לא קיים נציג תפעול, אפשר יהיה להוסיף פונקציה של פניה למנהלת התכנית הרלוונטית.

**גישות מקביליות לקבצי google sheets (concurrency):**

* הקוד ניגש לעמודה של התאריך הנוכחי ומחפש תא ספציפי לערוך אותו. גוגל מתייחס לזה כשתי פעולות שונות ואין כאן התנגשות. כלומר לא יקרה מצב שמידע הולך לאיבוד (הממשק של גוגל יודע להתמודד עם התרחיש שלנו)

**עומס גישות על השרת של גוגל:**

* ניתן לבצע עד 100 גישות לשניה, לפני שמתחילה האטה (כשעוברים את ה200 ייתכן ונקבל שגיאה שיש לנו יותר מידי גישות) גם בתרחיש הכי קיצוני אנחנו לא מתקרבים לכמות אנשים כזאת שבא כולם סורקים את הקוד בו זמנית באותה שניה. אני יכול להתמודד עם זה על ידי בניה של מערכת תורים במקרה של עומס כדי למנוע חסימת גישה, אבל אין צורך.

Stack (באיזה טכנולוגיות נעשה שימוש):

* Front: Html + CSS
* Back: Javascript: Node.Js, Express.js
* Database: PostgreSQL
* Google sheets API v4
* QR generation: python
* Email notifier: Nodemailer

תכנון זמן

| **משימה** | **זמן מוערך** | **שבוע** | **שלב** |
| --- | --- | --- | --- |
| תכנון סופי וסגירת מסמך + פלואו | 6-8 שעות | שבוע 1 | 1 |
| הקמת בסיס קוד: Node.js, Express, מבנה פרויקט | 6-8 שעות | שבוע 1 | 2 |
| חיבור Google Sheets API + ניהול גישה/שגיאות | 10-12 שעות | שבוע 2 | 3 |
| שליחת מיילים אוטומטית QR Code + | 6-8 שעות | שבוע 2 | 4 |
| בניית טופס, וולידציה , LocalStorage) פיתוח דף הרשמה ראשונה) | 10-12 שעות | שבוע 3 | 5 |
| טיפול במקרי קצה נפוצים (מס' קורסים, שינוי טלפון וכו’) | 6-8 שעות | שבוע 3 | 6 |
| מערכת לוגים + התראות מייל | 6-8 שעות | שבוע 4 | 7 |
| טופס עריכה: פאנל אדמין מאובטח (Express-session, bcrypt,) | 12-14 שעות | שבוע 4–5 | 8 |
| כתיבת מדריך משתמש + תיעוד תרחישים | 6-8 שעות | שבוע 5 | 9 |
| בדיקות כוללות + Debug לכל התרחישים | 10-12 שעות | שבוע 5 | 10 |

הערה:

* שלב 8 הוא אופציונלי, המטרה שלו היא לאפשר עריכה של מסד הנתונים באמצעות פאנל אדמין – זה דורש הרבה עבודה בעיקר מהכיוון של אבטחה. זה לא חובה, ואפשר קודם לראות שהמערכת עובדת ונכונה לנו ורק לאחר מכן להוסיף את זה.

סיכום:

אם בונים את פאנל האדמין:

בין 78 ל 98 שעות, ביום עבודה שבו יש לי עוד משימות מסביב אם אני מצליח להקדיש כ3-4 שעות מלאות של עבודה זה יקח כ 25 ימי עבודה = 5 שבועות

אם לא בונים את פאנל האדמין:

בין 66 ל 84 שעות, ביום עבודה שבו יש לי עוד משימות מסביב אם אני מצליח להקדיש כ3-4 שעות מלאות של עבודה זה יקח כ 21 ימי עבודה = 4 שבועות