# פרוייקט סיום – סייבר

## מגישים:

מגישה: דניאלה בועז 209371913

חברי הקבוצה:

רמז סבוי 316491950

עמית ויזל 208349787

אור הורוביץ 316283944

גל ולטר 322980798

אבישי אלרום 207946591

## דרישות:

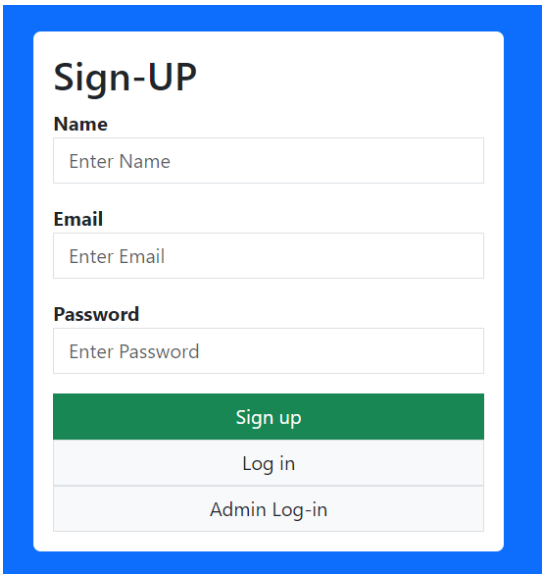
- מסד נתונים מסוג -Mysql

| id | username | email | password | prepass1 | prepass2 | prepass3 | token | isAdmin |
|---|---|---|---|---|---|---|---|---|
| 61 | <script>al... | 44222243g@g... | $2b$10$A3j8venIXbMza... | $2b$10$A... | $2b$10$A3j... | $2b$10$A... | NULL | 0 |
| 62 | daniella | daniellaboaz22... | $2b$10$E0I7KfPV09P4Q... | $2b$10$1... | $2b$10$12.L... | $2b$10$1... | 34d92aa2761817... | 0 |
| 63 | dani | danielsslaboaz2... | $2b$10$um5g9Kj86BqH... | $2b$10$u... | $2b$10$um5... | $2b$10$u... | NULL | 0 |
| 64 | aaaaaaaaa | aaaaa@a.a | $2b$10$h2JxtjKg3a4eg... | $2b$10$h... | $2b$10$h2J... | $2b$10$h... | NULL | 0 |

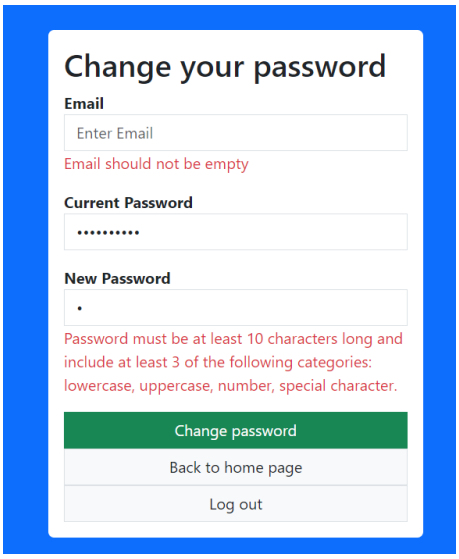
- הקמת אתר ווב מבוסס react js and node js.

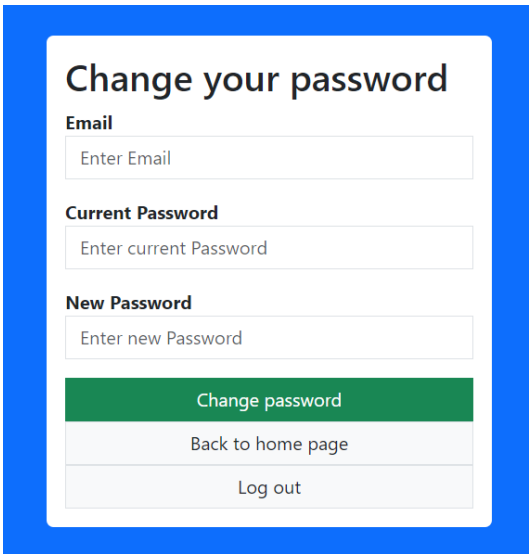
## חלק א ( פתוח עקרונות של פיתוח מאובטח):

1. מסך Register של משתמשים חדשים
   a. הגדרת יוזרים חדשים
   b. הגדרת סיסמא מורכבת ( הגדרות ודרישות סיסמא מורכבת ינוהל באמצעות קובץ קונפיגורציה)
   c. סיסמא תשמר במסד הנתונים באמצעות שימוש בפונקציית HMAC + Salt
   d. הגדרת מייל למשתמש

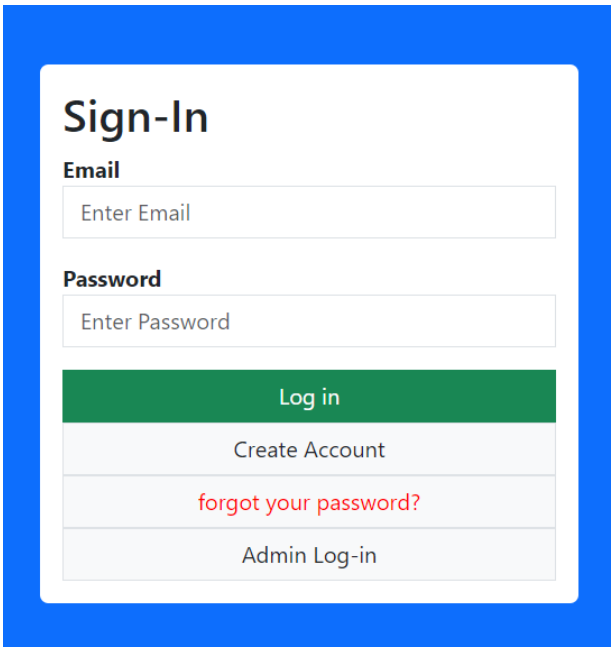


2. מסך לשינוי סיסמא עבור משתמש
   a. הזנת סיסמא קיימת
   b. הכנסת סיסמא חדשה אשר תעמוד בדרישות
   c. כפי שמוגדר בקובץ הקונפיגורציה

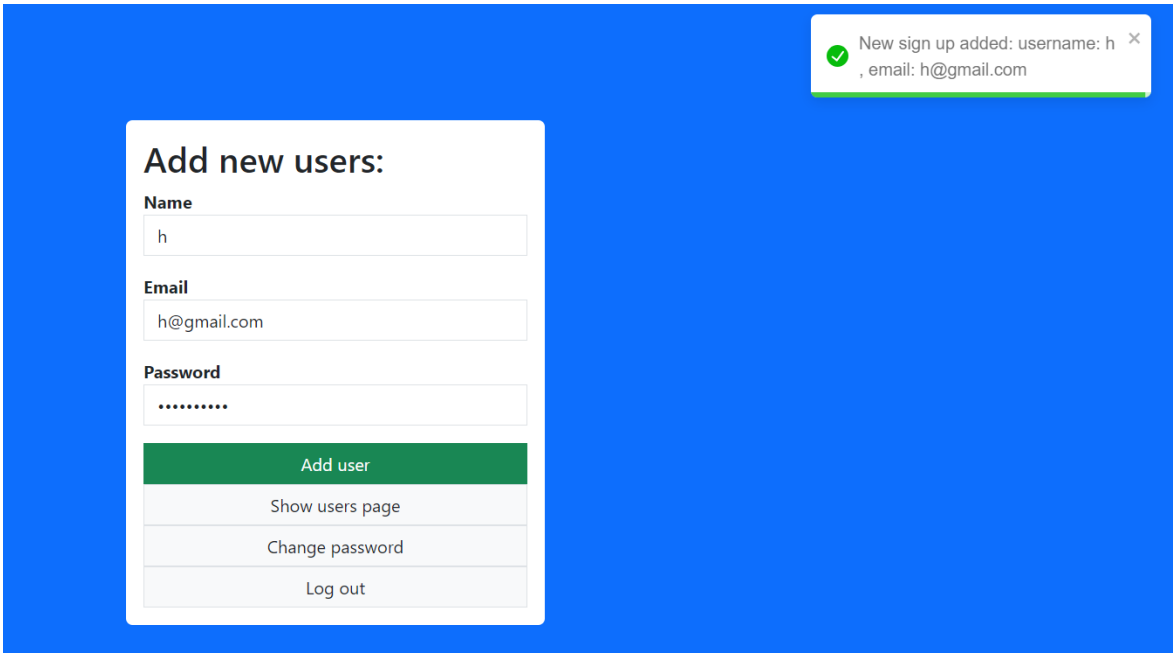


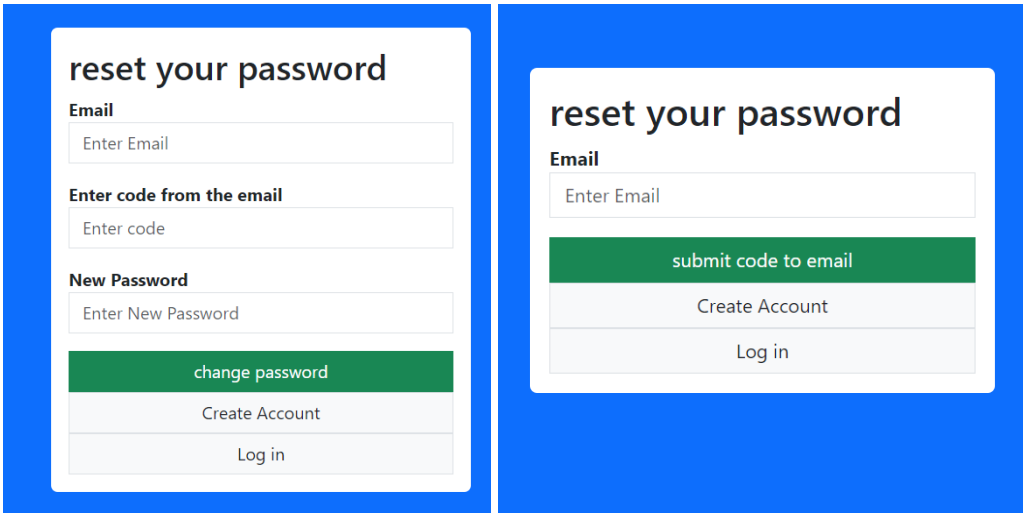

3. מסך Login " למערכת מידע " Comunication_LTD
   a. הזנת יוזר
   b. הזנת סיסמא
   c. בדיקה אם המשתמש קיים או לא והחזרת הודעה מתאימה.



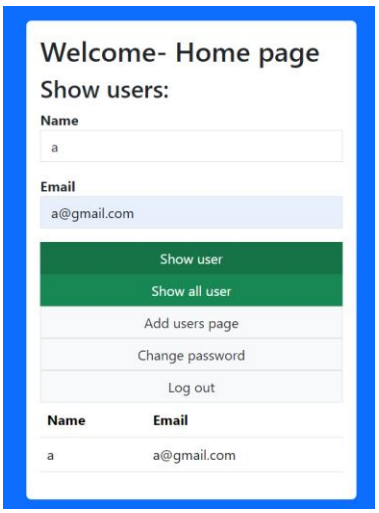
4. מסך מערכת
   a. הכנסת לקוח חדש עם פרטים חדשים.
   b. הצגה למסך את שם לקוח החדש שהוזן.

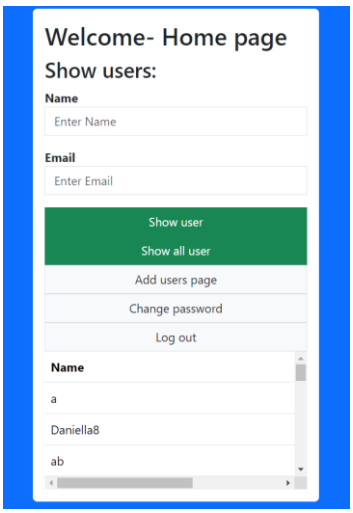


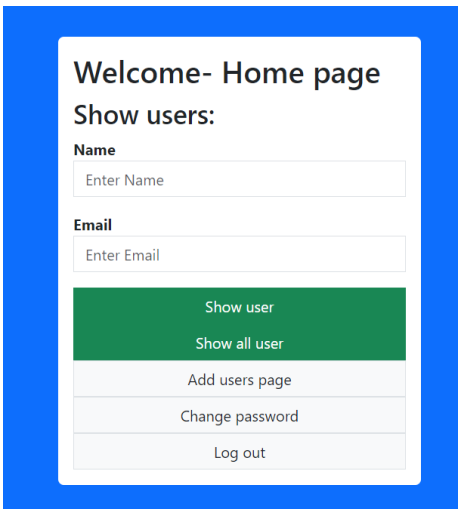

5. מסך "שכח סיסמא"
   a. המשתמש מפעיל אופציה זאת
   b. המערכת מייצרת ערך אקראי ושולחת אותו למייל של המשתמש
   c. הערך האקראי חייב להיות מוגדר באמצעות SHA-1
   d. המשתמש מזין ערך זה על מנת שיוכל להגיע לחלון שינוי סיסמא של משתמש.




מסך הצגת משתמשים:

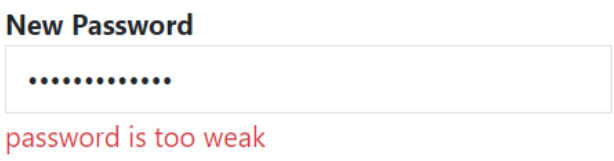


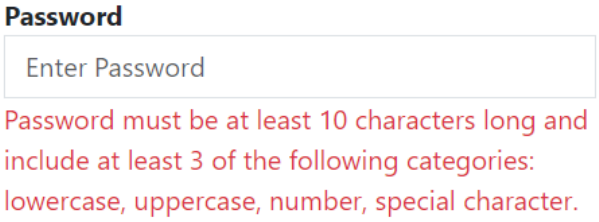



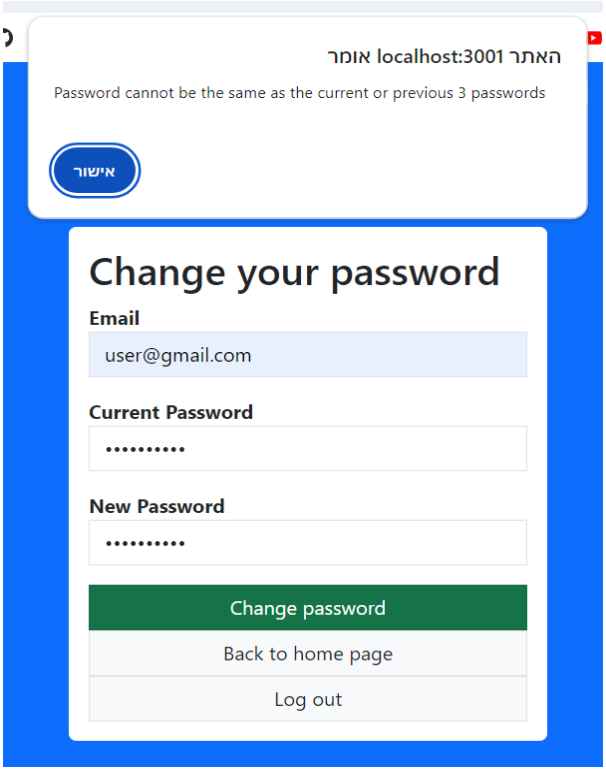



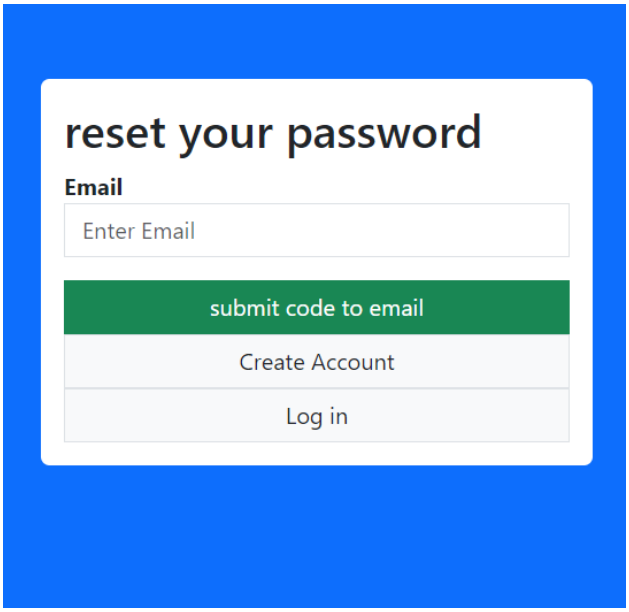

## קובץ קונפיגורציה לניהול סיסמא ( ערכים ניתנים לשינוי על מנהל המערכת)

1. אורך סיסמא : [10]
2. סיסמא מורכבת : [אותיות גדולות , קטנות, ספרות, תווים מיוחדים]
3. היסטוריה : [3 פעמים]
4. מניעת שימוש במילון [.....]
5. מספר ניסיונות בשלב ה [3] Login



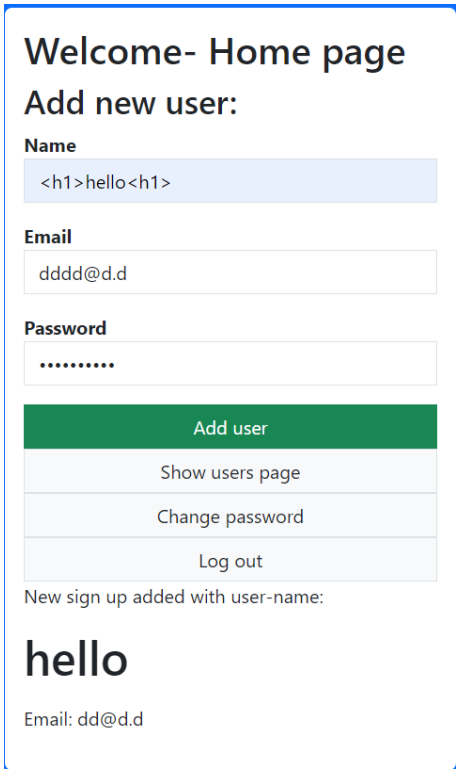



# חלק ב (שימוש בטכניקות XSS + :Sqli)

*הצגת דוגמא לשימוש בהתקפה מסוג Stored XSS בסעיף 4 (מסך מערכת) מחלק א:




בנוסף, בחלק של הצד שרת הוספנו שורת קוד שעוזרת במניעת התקפות מסוג זה:

```
const helmet = require("helmet");
```

*הצגת דוגמא לשימוש בהתקפה מסוג Sqli על סעיף 1 + סעיף 3 + סעיף 4 מחלק א של הפרויקט:

סעיף -3 מסך התחברות:

הכנסת מייל קיים לדוגמא: d@d.d עם הסיסמא: '1'='1 OR 'email=' -- -' (התחברות בהצלחה)

*הצגת פתרון נגד הפרצות בסעיף 1 על ידי שימוש בקידוד של תווים מיוחדים:

(הצגת דוגמא לשימוש בהתקפה מסוג Stored XSS בסעיף 4 (מסך מערכת) מחלק א)

קוד פגיע:

```
app.post("/signup", (req, res) => {
  hashPassword(req.body.password).then((newPassword) => {

      /*const values = [
        validator.escape(req.body.name[0]),
        validator.escape(req.body.email[0]),
        newPassword,
        newPassword,
        newPassword,
        newPassword,
      ];*/


      sendQueryCommit(
       /* "INSERT INTO loginvulnerable (`username`, `email`, `password`, `prepass1`, `prepass2`, `prepass3` ) VALUES (?)",
        values*/
      `INSERT INTO loginvulnerable (username, email, password, prepass1, prepass2, prepass3) VALUES
      ('${req.body.name[0]}', '${req.body.email[0]}',
      '${newPassword}', '${newPassword}', '${newPassword}', '${newPassword}')`
      )
```

קוד ללא encode של html, הורדנו את validator.escape וישר השתמשנו במשתנים.

קוד מאובטח:

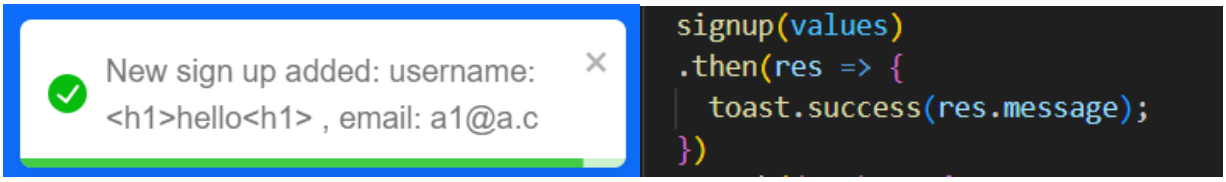
שימוש ב validator.escape על מנת לוודא קידוד של תווים מיוחדים:

```
app.post("/signup", (req, res) => {
  hashPassword(req.body.password).then((newPassword) => {

      const values = [
        validator.escape(req.body.name[0]),
        validator.escape(req.body.email[0]),
        newPassword,
        newPassword,
        newPassword,
        newPassword,
      ];

      sendQueryCommit(
        "INSERT INTO login (`username`, `email`, `password`, `prepass1`, `prepass2`, `prepass3` ) VALUES (?)",
        values
      )
```

בנוסף, הצגנו את המשתנים במסך המערכת על ידי התראה מסוג toast שגם היא מקודדת תווים מיוחדים למניעת xss:



*הצגת פתרון נגד הפרצות בסעיף 4 + סעיף 1 + סעיף 3 מחלק א על ידי שימוש ב Parameters או שימוש ב :Stored procedures

סעיף 1- מסך הרשמה: סעיף 4- מסך מערכת: (נשלח לאותו מקום כמו בסעיף 1)

קוד פגיע:

```
app.post("/signup", (req, res) => {
  hashPassword(req.body.password).then((newPassword) => {

    /*const values = [
      validator.escape(req.body.name[0]),
      validator.escape(req.body.email[0]),
      newPassword,
      newPassword,
      newPassword,
      newPassword,
    ];*/

    sendQueryCommit(
     /* "INSERT INTO loginvulnerable (`username`, `email`, `password`, `prepass1`, `prepass2`, `prepass3` ) VALUES (?)",
      values*/
    `INSERT INTO loginvulnerable (username, email, password, prepass1, prepass2, prepass3) VALUES
    ('${req.body.name[0]}', '${req.body.email[0]}',
    '${newPassword}', '${newPassword}', '${newPassword}', '${newPassword}')`
    )
```

קוד מאובטח:

```
const values = [
  validator.escape(req.body.name[0]),
  validator.escape(req.body.email[0]),
  newPassword,
  newPassword,
  newPassword,
  newPassword,
];

sendQueryCommit(
  "INSERT INTO login (`username`, `email`, `password`, `prepass1`, `prepass2`, `prepass3` ) VALUES (?)",
  values
)
```

סעיף 3- מסך התחברות:

קוד פגיע:

```
loginRouter.post("/login", async (req, res) => {
  try {
    const email = req.body.email;
    const password = req.body.password;

    const result = await sendQuery(`SELECT password FROM loginvulnerable WHERE email = '${email}' And password ='${password}'`);
```

קוד מאובטח:

```
loginRouter.post("/login", async (req, res) => {
  try {
    const email = validator.escape(req.body.email);
    const password = (req.body.password);

    const result = await sendQuery("SELECT password FROM login WHERE email = ?", [email]);
```