

תיק פרויקט

LGFW – Let'sGetFlagWars

שם בית הספר: עמל רב תחומי חדרה

שם פרויקט: מלחמות דגלים – Let's Get Flag Wars

מגיש: טל יוסף מטרי

תעודת זהות: 213727936

שם המורה: אסף אמיר

שם המגמה: הנדסת תוכנה סייבר

תאריך הגשה: 5.4.2021

תוכן עניינים

תוכן

4	מבוא
4	מבוא
4	מטרת הפרויקט
4	תהליך המחקר
5	אילוצים ודרישות
6	בסיס הנתונים
9	טבלת uml של המחלקות והקשרים ביניהם(ירושה)
12	מסכי הפרויקט והעיצוב
12	מסכים למשתמש לא מחובר
18	מסכים למשתמש מחובר
22	ארגון קבצי הפרויקט
24	קטעי קוד מרכזיים
24	הסבר מפורט על קובץ server.js
27	הסבר על קובץ send_mails.js
28	פונקציית createTransporter()
29	פונקציית sendMail
31	הסבר על קובץ refreshTokens.js
35	Auth.js
39	קובץ res_getters.js
45	קובץ Index_logic.js
46	בקובץ find_game_client.js יש חלק מרכזי
47	קבצי הHTML
47	Index.html
49	Login.ejs
49	Sign_up.html
50	Activate_account.html
50	Dashboard.html
50	Admin_dashboard.html
51	Finding_game.html
52	Game.html
54	Reset_password.html

55	Reset_password_itself.htm
56	Add_quest.html
57	Add_regular_quest.html
57	רפלקציה אישית
58	ביבליוגרפיה

מבוא

מבוא:

ישנם משחקים באינטרנט של מציאת דגלים בעזרת פתרון חידות. הרבה אנשים אוהבים את שיטת המשחק הזאת בה יש מחשבה. פרויקט זה יאפשר לאותם אנשים להתחרות בזמן אמת בפתרון החידות, דבר שמוסיף הרבה לחוויית המשחק.

מטרת הפרויקט:

מטרת הפרויקט היא לבנות אתר אשר רץ על - localhost – 127.0.0.1:8080 port 8080. באתר יהיה אפשרות להתחבר ולהירשם בתור משתמש. בנוסף למשתמש תהיה יכולת לשנות את הסיסמה שלו במידה ושכח אותה. כאשר המשתמש יהיה מחובר לאתר הוא יוכל לדבר עם עוד משתמשים מחוברים בזמן אמת והוא יוכל לחפש משחק. כאשר המשחק נמצא לשתי המשתתפים, שניהם יהיו עם אפס פצצות ו100% חיים. במידה ואחד מהם ענה נכון על חידה הוא יקבל פצצה איתה יוכל להפציץ את השחקן שנגדו או לפוצץ את קיר השאלה ובכך לקבל אליה רמז שיעזור לו לפתור אותה. המשתתף האחרון שיישאר לו חיים ינצח ואילו השני יפסיד. בנוסף הקפדתי לשמור על פרטי המשתמשים (במיוחד בסיסמה) כמה שיותר מוגנים. כך בעצם בעזרת הפרויקט יהיה ניתן להוסיף היבט תחרותי לפתירת החידות ומציאת הדגלים.

קהל היעד של פרויקט זה הוא לאנשים שאוהבים לפצח חידות ובנוסף אוהבים להתחרות ובכך לבדוק את קישוריהם ביחס לשאר האנשים.

בנוסף למטרת הפרויקט המעשית הייתה גם מטרה נוספת והיא ללמוד ולהתפתח בתחום המחשבים, התכנות והסייבר, יישום הידע שרכשתי בדרך על ידי למידה עצמאית ובנוסף למידת שפות תכנות חדשות ורכישת ניסיון בהן.

אני רוצה לציין שהייתה מוטיבציה רבה להשלים את הפרויקט על הצד הטוב ביותר, מוטיבציה זו באה מהסיבה לאהבת המקצוע והרצון להשלים פרויקט רחב מבחינת שפות התכנות והרעיונות התיאורטיים מאחורי הקלעים.

תהליך המחקר:

הפרויקט מבוסס על אנשים בעלי תשוקה לפתירת חידות אשר רוצים להוסיף אלמנט תחרותי, לכן תחילה בניתי 8 חידות אשר כל אחת שונה, לאחר מכן

התחלתי לחקור את החומר התיאורטי יותר ולבסוף התחלתי ליישם את כל הדברים על מנת ליצור מערכת אשר תענה על מטרות הפרויקט.

אילוצים ודרישות:

- (1) אילוצי טכנולוגיה, תוכנה ותיאוריה: למידה ושימוש בכמה שפות תכנות ולימוד ויישום של ידע תיאורטי.
- (2) אילוצי זמן: לסיים את פיתוח הפרויקט בפרק זמן של פחות מחצי שנה (בערך 4 חודשים) במקביל ללימודים.
- (3) אילוצי למידה: אצטרף ללמוד באופן עצמאי חומר שיעזור לי להרכיב ולהשלים את הפרויקט באמצעות קריאת מקורות באינטרנט וסרטונים.
- (4) אילוצי משתמשים: מנהל המערכת אשר יוכל להעלות שאלות יוגדר באופן ידני בבסיס הנתונים.

בסיס הנתונים - SQLite:

את בסיס הנתונים בניתי באמצעות קובץ בשם database.db אשר בו יהיו שתי טבלאות נתונים. על קובץ מסד הנתונים אני שולט באמצעות פייתון(שפת תכנות), קובץ ייחודי אשר בתוכו יש class של טבלה רגילה ושני קלאסים נוספים אשר יורשים מטבלה רגילה ושםם usersTable(SqlTable ו- questTable(SqlTable) כיוון ששני הטבלאות הן למעשה טבלאות רגילות עם פעולות ותכונות נוספות לטבלה רגילה, אשר ייחודיות להן.

```

5 class SqlTable(object):
6     def __init__(self, database_file, table_name):
7         self.table_name = table_name
8         self.database_file = database_file

```

בתמונה למעלה ניתן לראות את "הקונסטרקטור" של המחלקה SqlTable(object) אשר יוצרת אובייקט(עצם) של טבלה רגילה בעלת התכונות הבאות: database_file, table_name תכונות אלה הכרחיות לבניית כל טבלה.

```

22 class usersTable(SqlTable):
23     def __init__(self, database_file, table_name, user_id, email, username, password, count, user_type, reset_link):
24         SqlTable.__init__(self, database_file, table_name)
25         self.user_id = user_id
26         self.email = email
27         self.username = username
28         self.password = password
29         self.count = count
30         self.user_type = user_type
31         self.reset_link = reset_link
32         connection = sqlite3.connect(self.database_file)
33         c = connection.cursor()
34         string = """CREATE TABLE IF NOT EXISTS {}({} INTEGER PRIMARY KEY AUTOINCREMENT, {} TEXT NOT NULL, {} TEXT NOT
35         c.execute(string)
36         connection.commit()
37         connection.close()

```

בתמונה למעלה ניתן לראות את "הקונסטרקטור" של המחלקה usersTable(SqlTable) אשר יורשת את התכונות של המחלקה SqlTable(object) ומקבלת בנוסף user_id, email, username, password, count, user_type, reset_link שם השדה של מספר הזיהוי של המשתמש בטבלה. email: שם השדה של האימייל של המשתמש בטבלה.

username: שם השדה של שם המשתמש בטבלה.
 password: שם השדה של הסיסמה של המשתמש בטבלה.
 count: שם השדה של הספירה (מספר לצורך זיהוי התנתקות מהמערכת) של המשתמש בטבלה.
 user_type: שם השדה של סוג המשתמש בטבלה.
 reset_link: שם השדה של "קישור איפוס הסיסמה" (בפועל יהיה שם jwt token – הסבר על כך יהיה בהמשך) של המשתמש בטבלה.
 וכמובן שאר התכונות שטבלה רגילה צריכה.

```

221 class questTable(SqlTable):
222     def __init__(self, database_file, table_name, quest_id, quest, clue, answer, image_name):
223         SqlTable.__init__(self, database_file, table_name)
224         self.quest_id = quest_id
225         self.quest = quest
226         self.clue = clue
227         self.answer = answer
228         self.image_name = image_name
229         connection = sqlite3.connect(self.database_file)
230         c = connection.cursor()
231         string = """CREATE TABLE IF NOT EXISTS {}({} INTEGER PRIMARY KEY AUTOINCREMENT, {} TEXT
232         c.execute(string)
233         connection.commit()
234         connection.close()

```

בתמונה למעלה ניתן לראות את "הקונסטרוקטור" של המחלקה questTable(SqlTable) אשר יורשת את התכונות של המחלקה SqlTable(object) גם כן אך לעומת מחלקת המשתמש היא תקבל בנוסף quest_id, quest, clue, answer, image_name הסבר התכונות:
 quest_id: שם השדה של מספר הזיהוי של השאלה/חידה בטבלה.
 quest: שם השדה של השאלה/ חידה עצמה בטבלה.
 clue: שם השדה של הרמז לשאלה/חידה בטבלה.
 answer: שם השדה של התשובה לשאלה/חידה בטבלה.
 image_name: שם השדה של שם התמונה של השאלה/חידה בטבלה.

מסד הנתונים למעשה נראה כך בטבלה:

כל מסד הנתונים נמצא בקובץ database.db

טבלת users - משתמשים

User_id	Integer Primary Key Autoincrement	שדה אשר הערך הבא ייגדל באחד אוטומטית, ייחודי(לא ייוצר מצב בו יהיו שתי משתמשים בעלי אותו מספר זיהוי) והערך מטיפוס מספר שלם(int)	מספר הזיהוי של המשתמש
Email	Text Not Null	טקסט אשר לא יכול להיות ערך ריק(null)	אימייל
Username	Text Not Null	טקסט אשר לא יכול להיות ערך ריק(null)	שם משתמש
Password	Text Not Null	טקסט אשר לא יכול להיות ערך ריק(null)	סיסמה
Count	Integer Not Null	מספר שלם(int) אשר לא יכול להיות ערך ריק(null)	ספירה(הסבר יותר מפורט לשימוש לזה יהיה בהמשך)
User_type	Integer Not Null	מספר שלם(int) אשר לא יכול להיות ערך ריק(null)	סוג המשתמש(בדיקה לזיהוי משתמש רגיל או מנהל)

Reset_link	Text Not Null	טקסט אשר לא יכול להיות ערך ריק (null)	טוקן לשחזור סיסמת המשתמש
------------	---------------	---------------------------------------	--------------------------

Quest_id	Integer Primary Key Autoincrement	שדה אשר הערך הבא ייגדל באחד אוטומטית, ייחודי (לא יוצר מצב בו יהיו שתי משתמשים בעלי אותו מספר זיהוי) והערך מטיפוס מספר שלם (int)	מספר זיהוי של השאלה
Quest	Text not null	טקסט אשר לא יכול להיות ערך ריק (null)	השאלה עצמה
Clue	Text not null	טקסט אשר לא יכול להיות ערך ריק (null)	הרמז
Answer	Text	טקסט אשר יכול להיות ערך ריק (null)	התשובה
ImageName	Text	טקסט אשר יכול להיות ערך ריק (null)	שם התמונה

טבלת quests – שאלות/חידות

טבלת uml של המחלקות והקשרים ביניהם(ירושה):

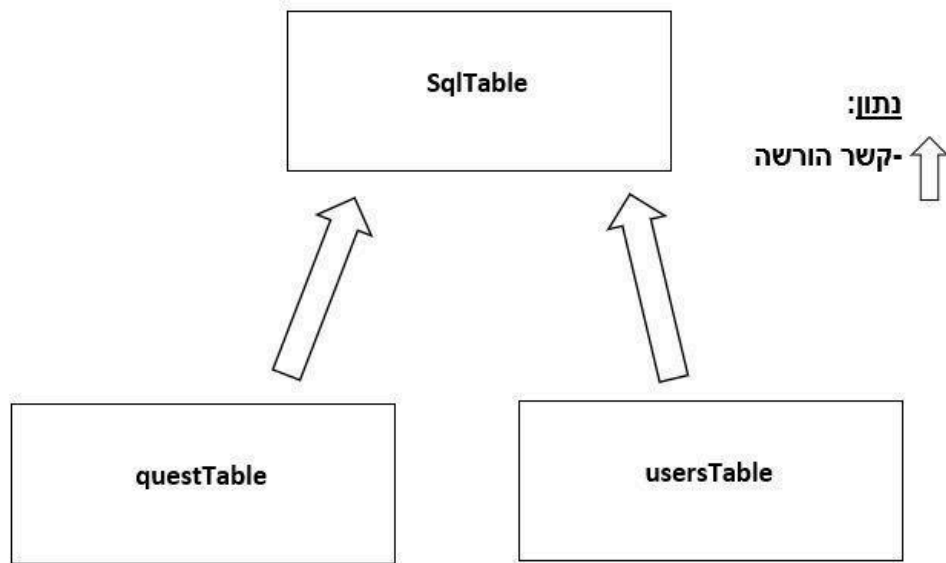
SqlTable
Database_file: String
Table_name: String
(Get_all_data_from_table(self

usersTable
Database_file: String
Table_name: String
User_id: String
Email: String
Username: String
Password: String
Cout: String
User_type: String
Reset_link: String
(Insert(self, email, username, password (1
Check_email_existance(self, email) – (2
returns boolean
Check_username_existance(self, (3
username) - returns boolean
(Delete_by_email(self, email (4
Get_max_user_id(self) – returns Integer (5
Update_by_email(self, username, (6
(password, email
Get_user_password(self, email) – (7
returns String
Change_user_password_by_user_id(self,(8
(user_id, new_password
Get_user_id_by_email(self, email) – (9
returns Integer

Get_count_by_email(self, email) – (10
 returns Integer
 Get_user_type_by_email(self, email) – (11
 returns Integer
 Get_username_by_email(self, email) – (12
 returns String
 Get_email_by_username(self, (13
 username) – returns String
 (Set_cout_by_id(self, id (14
 Check_user_password(self, email, (15
 password) – returns boolean
 Get_all_by_id(self, id) – returns tuple (16
 of user info
 Get_reset_link_by_id(self, id) – returns (17
 String
 Update_reset_link_by_id(self, id, (18
 (reset_link
 Check_if_admin(self, id) – returns (19
 boolean

questTable
Database_file: String Table_name: String Quest_id: String Quest: String Clue: String Answer: String Image_name: String
Insert_question(self, quest, clue, answer, (image_name Get_quest_by_id(self, id) – returns String Get_max_id(self) – returns Integer

כך ש:



מסכי הפרויקט והעיצוב:

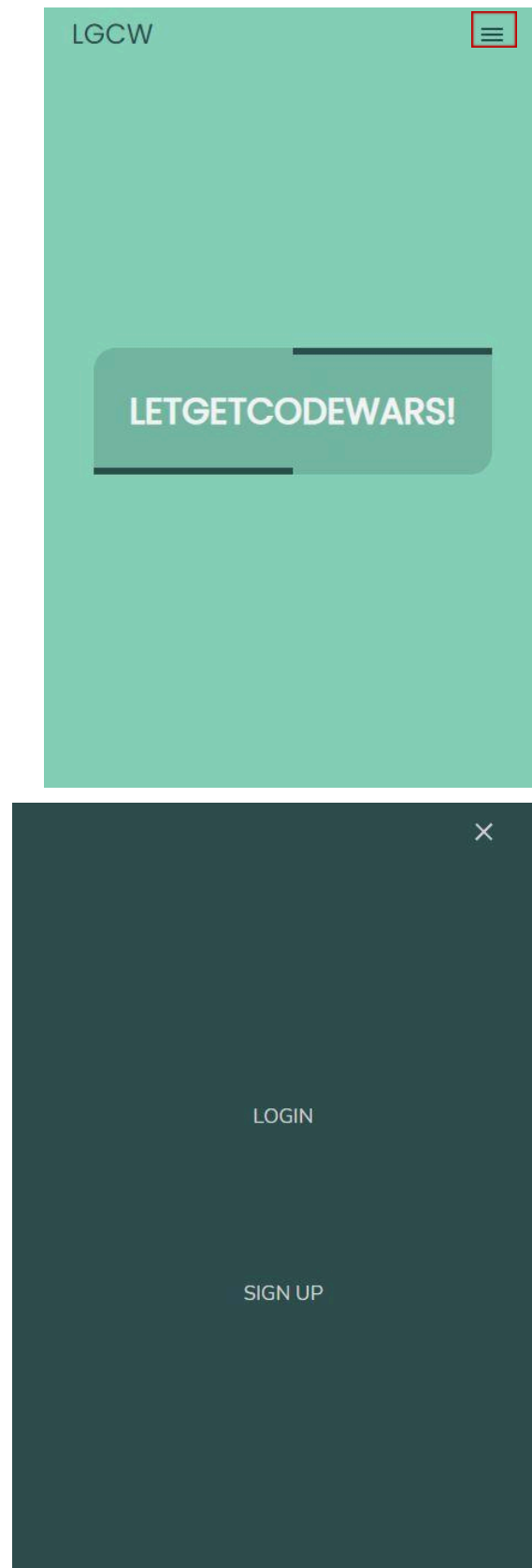
כאן אציג את כל המסכים של הפרויקט במצבים שונים לדוגמה כאשר המשתמש מחובר או לא.

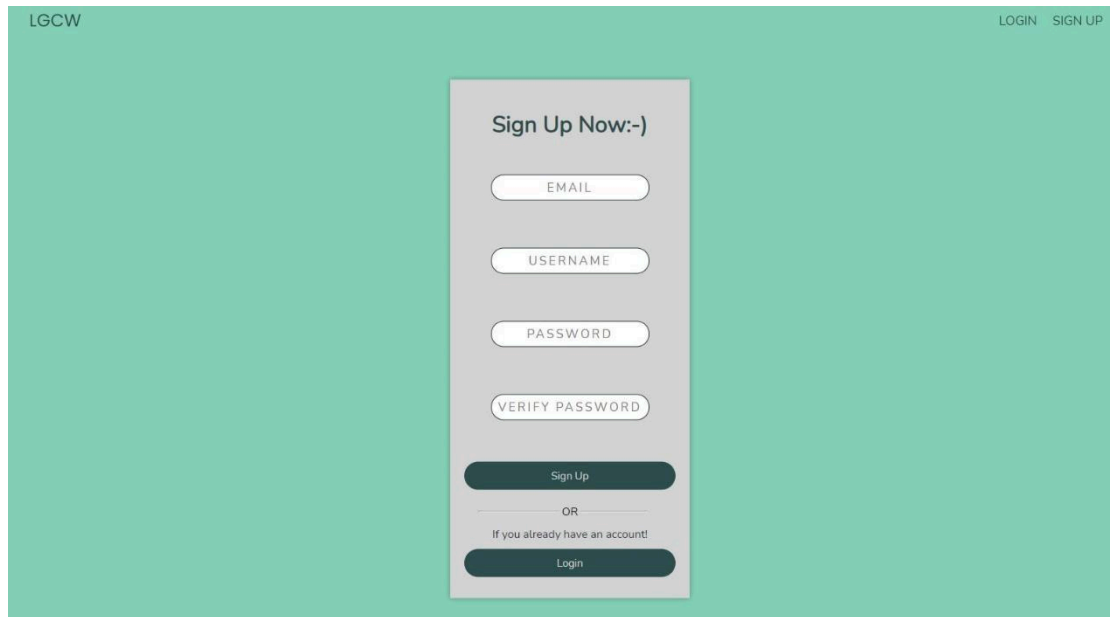
מסכים למשתמש לא מחובר:

המסך הראשי:



תצוגה למסכים יותר קטנים: כאשר לוחצים על שלוש הקווים בצד ימין

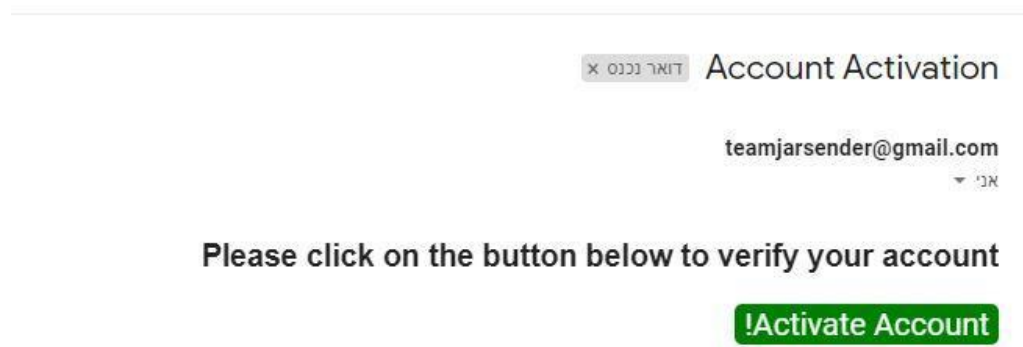




במסכים קטנים התצוגה דומה למסך הבית.
מסך אחרי הרשמה(שליחת מייל אישור הרשמה):

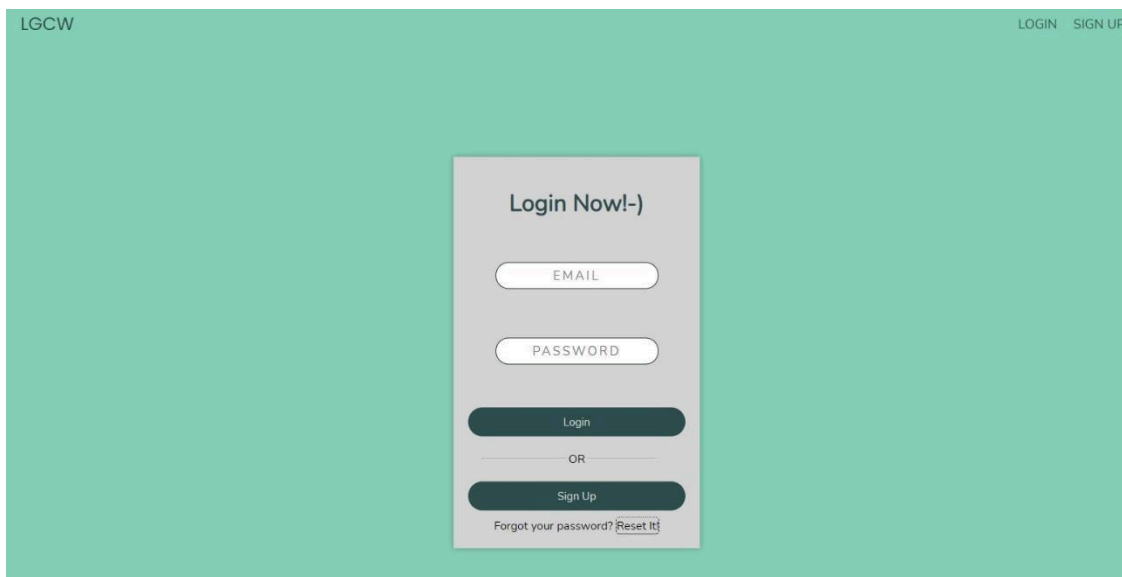
Please Activate Your Account via the Email That sent to you!

מייל האישור:



לאחר אישור ההרשמה עבר בהצלחה המשתמש יובל למסך ההתחברות.

מסך ההתחברות:

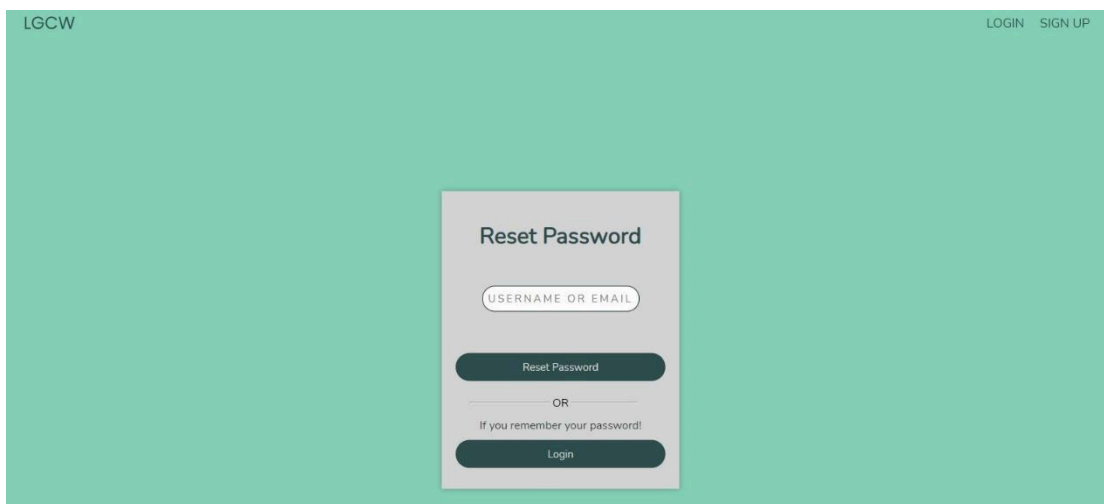


The screenshot shows a login form titled "Login Now!-" on a teal background. The form is centered and contains the following elements: a header "LGCW" in the top left and "LOGIN SIGN UP" in the top right; input fields for "EMAIL" and "PASSWORD"; a "Login" button; a separator "OR"; a "Sign Up" button; and a link "Forgot your password? Reset It!".

במסכים קטנים התצוגה דומה למסך הבית.

לאחר הכנסת פרטים נכונה האתר יוביל את המשתמש למסך הראשי לאחר התחברות.

מסך שיחזור סיסמה:



The screenshot shows a "Reset Password" form on a teal background. The form is centered and contains the following elements: a header "LGCW" in the top left and "LOGIN SIGN UP" in the top right; an input field for "USERNAME OR EMAIL"; a "Reset Password" button; a separator "OR"; a link "If you remember your password!"; and a "Login" button.

לאחר רשימת אימייל או שם משתמש יובל המשתמש למסך הבא:

Reset Password Mail Sent, go ahead and check it!

אימייל השחזור עצמו:

דואר נכנס X

Reset Your Password

teamjarsender@gmail.com

אני

תרגום הודעה

עברית

>

אנגלית

XA

!Click on the button below to reset your password

!Reset Password

לאחר לחיצה לשיחזור הסיסמה יובל המשתמש למסך הבא:

LGCW

LOGIN SIGN UP

Reset Password

PASSWORD

VERIFY PASSWORD

Reset Password

OR

If you remember your password!

Login

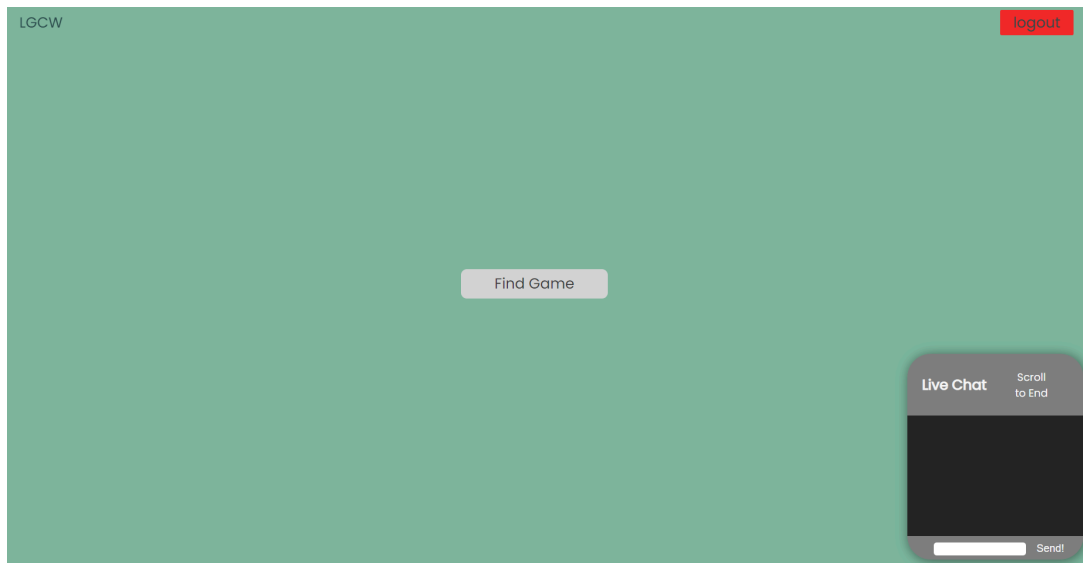
לאחר שיחזור הסיסמה בהצלחה יובל המשתמש למסך ההתחברות.

כניסה למסכי הכניסה, ההרשמה, איפוס הסיסמה ועמוד הבית בתור משתמש מחובר תוביל למסך הראשי לאחר ההתחברות.

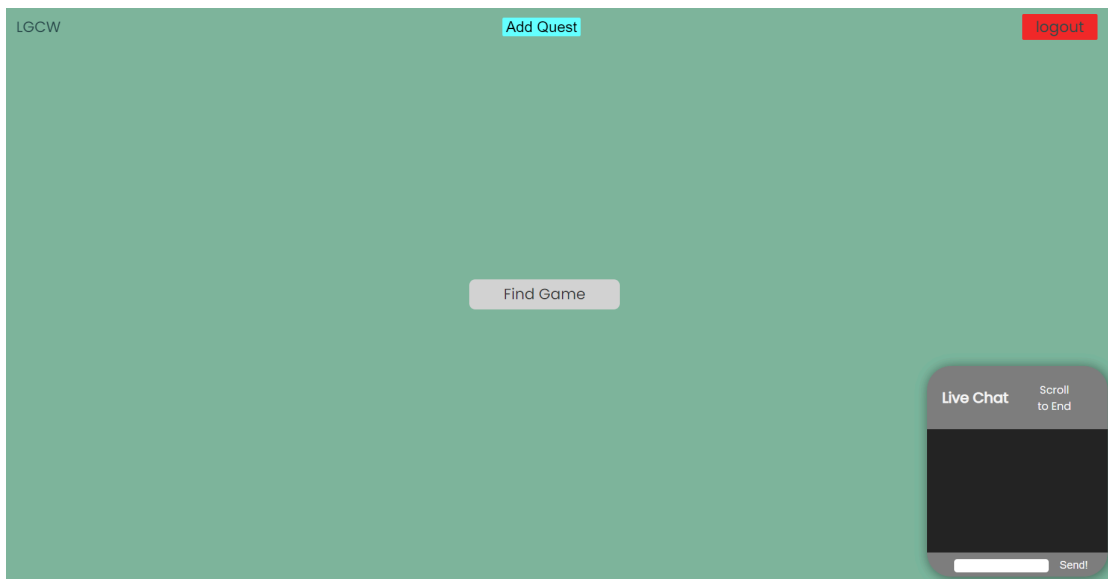
מסכים למשתמש מחובר:

מסך ראשי לאחר התחברות בהצלחה:

משתמש רגיל(לא מנהל):

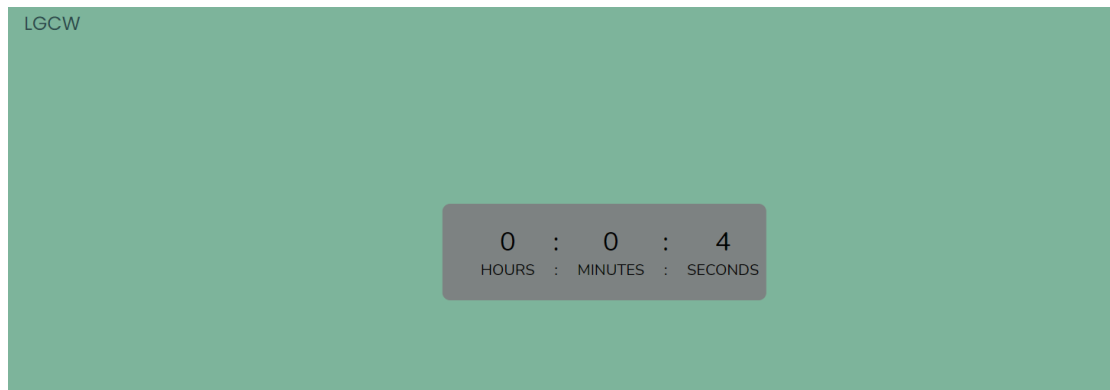


משתמש מנהל:

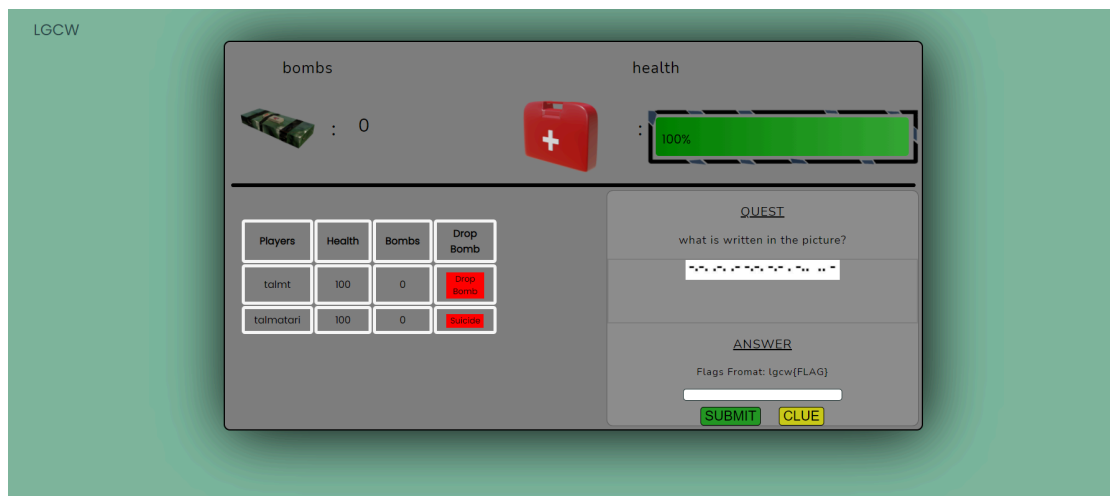


לחיצה על כפתור ההתנתקות תנתק את המשתמש מהאתר ותוביל אותו למסך ההתחברות. בצד ימין למטה ישנו צ'אט חי בו המשתמשים יכולים להתכתב אחד עם השני.

חיפוש משחק:



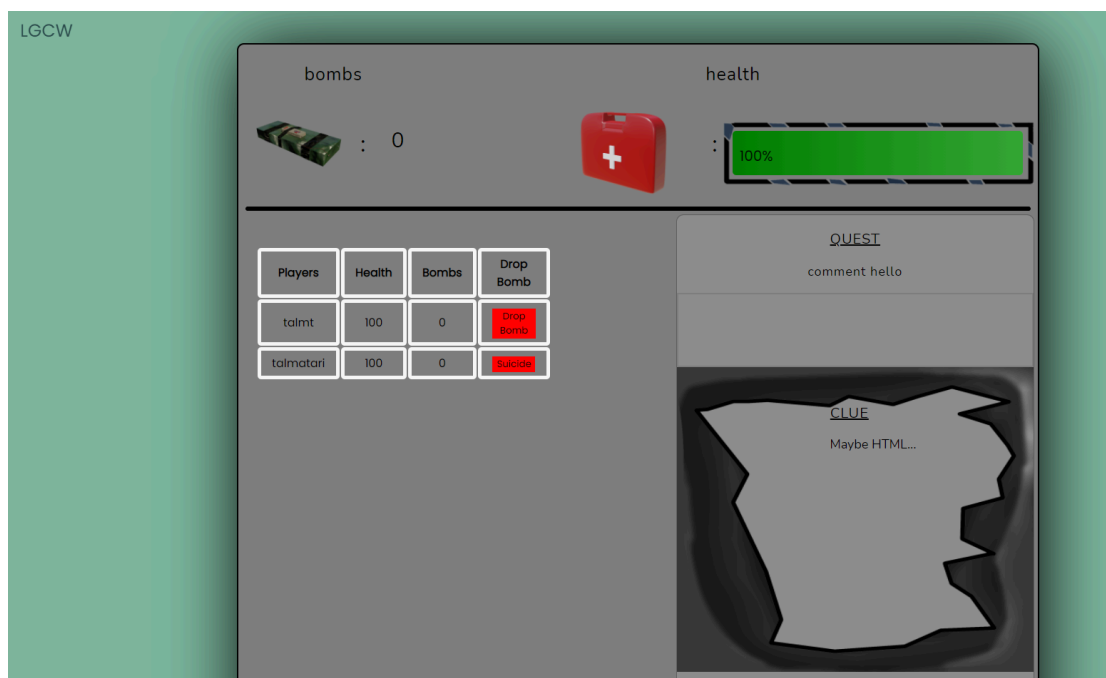
משחק עצמו לאחר מציאת משתמשים למשחק:



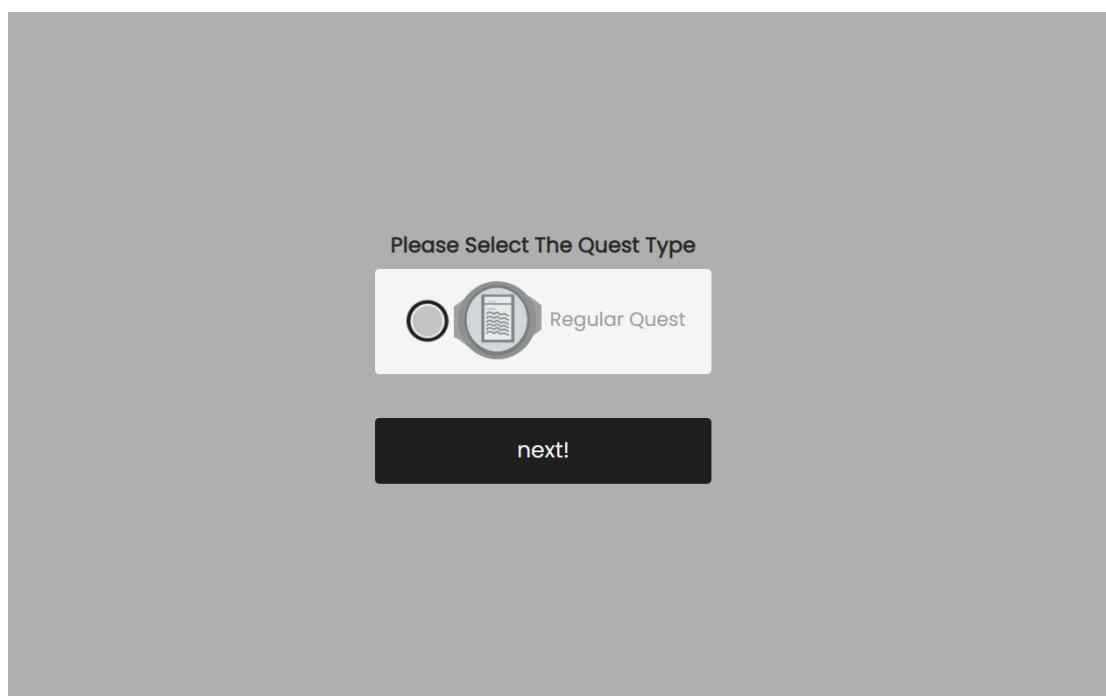
לאחר לחיצה על רמז:

שם: טל יוסף מטרי

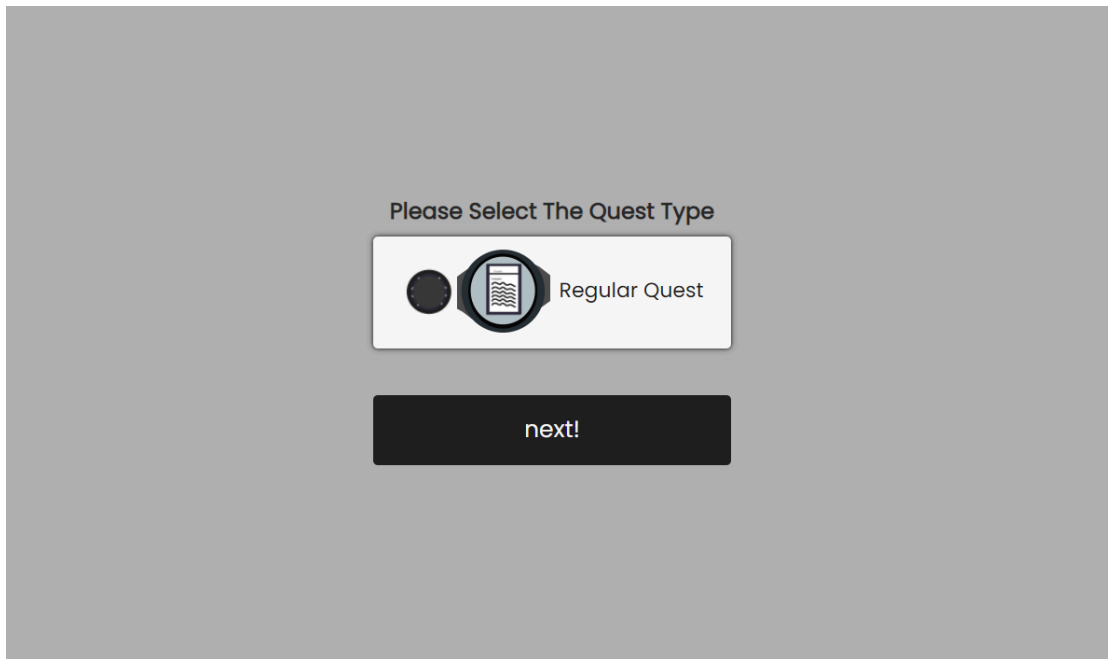
משחק דגלים רב משתתפים



הוספת שאלה:



לאחר בחירה של שאלה רגילה:



לחיצה על הבא כאשר "שאלה רגילה" מסומנת תוביל למסך הוספת השאלה.

הוספת שאלה עצמה:

Regular Quest

OPTIONAL:

☒ בחר קובץ ☐ לא נבחר קובץ

Image Preview

upload

0%

REQUIRED:

QUEST

CLUE

ANSWER

Add Quest!

*שלושת מסכים אלו הקשורים להוספת שאלה נגישים רק למנהלי המערכת.

ארגון קבצי הפרויקט:

קבצי הפרויקט מאורגנים בצורה כזו שקבצי ה-HTML אשר אחראים למבנה המסכים יהיו בתיקייה אחת המאגדת את כל הקבצים, כמו כן גם שאר תיקיות הפרויקט כך שקבצי css אשר אחראים על עיצוב דפי ה-HTML בתיקייה, קבצי javascript אשר אחראים על הלוגיקה והפעולות של קבצי ה-HTML בתיקייה וקבצי python אשר גם כן נמצאים בתיקייה ואחראים על שליטה במסד הנתונים וכלי עזר ללוגיקה שמאחורי הקלעים בשרת (back-end) בנוסף ישנם שני קבצי middleware אשר בודקות אם המשתמש אכן אמור להיות מחובר. קבצים אלו מאוגדים יחד תחת התיקייה middleware (קבצים אלו נכתבו ב-nodejs). תיקיית images מאגדת את כל התמונות הקשורות לפרויקט (חוץ מהתמונות של השאלות

אשר נמצאות בענן DropBox). תיקיית routes אחראית על הניתוב של המשתמש לפי ה-url (וה-POST method או GET). ישנם עוד כ-5 קבצים אשר אינם כלולים ומאוגדים תחת תיקייה מסוימת מהסיבה שמטרתם כללית לעומת שאר הקבצים.

קובץ ה-server.js אחראי על יצירת השרת עם כל הכלים ההכרחיים להרצת הפרויקט.

קובץ ה-create_auth_tokens.js אחראי על יצירת jwt tokens למטרות שונות במהלך הפרויקט כמו יצירת טוקנים לאחר שהמשתמש נכנס, יצירת טוקן לשחזור הסיסמה ויצירת טוקן לאישור ההרשמה.

קובץ ה-send_emails.js אחראי על שליחת אימיילים למשתמש בעת הצורך לדוגמה כדי לאשר את חשבונו או לאפס את הסיסמה שלו.

קובץ ה-.env אשר מאגד בתוכו את משתנים חשובים שעדיף לא לרשום אותם ישירות בקוד משני סיבות עיקריות. הראשונה היא שכתובת ערכים אלו בגלוי בקוד לא צעד חכם מבחינת אבטחה מסיבה שהקוד יודלף, בשימוש עם משתמשי מערכת הערכים הסודיים לא נמצאים מחוץ לשרת. הסיבה השנייה היא ששיטה זו כשמירת משתנים טובה עקב הגמישות שהיא מביאה איתה כיוון שבקלות אפשר לשנות מפתח סודי של יצירת טוקן מסוים או מפתח סיסמה המשמש מקור גישה ל-API מסויים שמקושר לחשבון.

וקובץ ה-database.db אשר מכיל בתוכו את טבלאות המידע ההכחי למערכת(מידע על משתמשים ושאלות)

כל המפורט מעלה ניתן לראות בתמונה זו:

תיקיית קבצים	24/03/2021 02:43	css
תיקיית קבצים	24/03/2021 19:15	html
תיקיית קבצים	01/04/2021 17:55	images
תיקיית קבצים	24/03/2021 02:44	javascript
תיקיית קבצים	01/01/2021 06:45	middleware
תיקיית קבצים	29/03/2021 02:06	node_modules
תיקיית קבצים	24/03/2021 02:54	python
תיקיית קבצים	27/12/2020 20:20	routes
2 KB	קובץ ENV	.env
2 KB	JavaScript File	create_auth_tokens.js
16 KB	Data Base File	database.db
1 KB	JSON File	package.json
129 KB	JSON File	package-lock.json
3 KB	JavaScript File	send_emails.js
3 KB	JavaScript File	server.js

קטעי קוד מרכזיים:

```

4 function pythonScript(data, filename){
5   return new Promise((resolve, reject) => {
6     spawn('python', [path.join(__dirname + "../python/" + filename).toString(), data]).stdout.on('data', (data) => {
7       resolve(data.toString().replace(/\r\n/gm, ""));
8     });
9   });
10 }

```

ניתן לראות בתמונה פונקציה אשר מקבלת מידע ושם קובץ(קובץ אשר נכתב בשפת התכנות python) ומחזירה הבטחה חדשה אשר מחזירה במידה והצליחה ולא הייתה שגיאה את המידע שהודפס בקובץ ה-python ללא הורדת שורות ופיצול כך למעשה הצלחתי להריץ קבצי python מתי שהייתי צריך דרך קובץ ה-nodejs.

הסבר מפורט על קובץ server.js:

```

1 //Main function
2 const main = () => {
3   //Import
4   const express = require('express');
5   const authRoutes = require('./routes/auth');
6   const resGetters = require('./routes/res_getters');
7   const chat_socket = require('./javascript/chat_handler');
8   const game_socket = require('./javascript/game_handler');
9   const path = require('path');
10  const cookieParser = require('cookie-parser');

```

ניתן לראות בתמונה זו שיצרתי פונקציה ראשית שתרוץ עם הרצת הקובץ server.js עם הפקודה node .\server.js. בנוסף ניתן לראות את כל המודולים שהשרת צריך כדי שהאתר יתפעל כמתוכנן.

מודול ה-"express" (שורה 4) אחראי על ניתוב המשתמש (routing) באתר, יכול להריץ דפים באמצעות מנועים שונים, נותן אפשרות לעבוד עם request ו-response איתם אפשר לעשות הרבה דברים שימושיים כמו להחזיר תגובה בהתאם לבקשת המשתמש.

מודול ה-"./routes/auth" (שורה 5) מייבא ניתובים שאני כתבתי בקובץ auth.js אשר נמצא בתיקיית routes. ניתובים אלו קשורים לתהליך ההתחברות, ההירשמות, אישור ההרשמה ואיפוס הסיסמה.

מודול ה-"./routes/res_getters" (שורה 6) מייבא את כל שאר הניתובים שכתבתי בקובץ res_getters.js הקשורים לניתוב המשתמש באתר בכל הדברים שלא קשורים לפעולות הכלולות בקובץ ה-"auth.js".

מודול ה-"./javascript/chat_handler" (שורה 7) מייבא את התנהגות והתנהלות הצ'אט החי הכלול בעמוד הראשי (dashboard) לאחר שהמשתמש מחובר לאתר.

מודול ה-"javascript/game_handler.js/" (שורה 8) מייבא את ההתנהגות והתגובות שהשרת מחזיר למשתמש כאשר הוא משחק במשחק(לאחר שלב חיפוש המשחק בו מספר שחקנים נמצאים).

מודול ה-"path" (שורה 9) הוא למעשה כלי עזר לעבודה עם קבצים במקומות שונים כיוון שהוא נותן את היכולת לעבוד עם פונקציות שימושיות הקשורות לנתיבים של קבצים ותקיות קבצים.

מודול ה-"cookie-parser" (שורה 10) אחראי על תפעול ושימוש בעוגיות(cookies) בשימוש עם מודול ה-"express" בכך שמודול זה עובד בתור תוכנת ביניים (middleware).

```

19  const app = express();
20  const chat_server = require('http').createServer();
21  const game_server = require('http').createServer();
22  const chat_io = require('socket.io')(chat_server, {
23    cors: {
24      origin: "http://localhost:8080",
25      methods: ["GET", "POST"]
26    }
27  });
28
29  const game_io = require('socket.io')(game_server, {
30    cors: {
31      origin: "http://localhost:8080",
32      methods: ["GET", "POST"]
33    }
34  });
35  //Port the server is running on
36  const PORT = process.env.PORT || 8080;
37  const CHAT_PORT = 3000;
38  const GAME_PORT = 5555;

```

כאן ניתן לראות שיצרתי אפליקציית אקספרס איתה אני יכול לבנות את הניתובים של בקשות HTTP ולהכניס את כל תוכנות הביניים (middlewares) (שורה 19). לאחר מכן בשורות 20 ו-21 אני יוצר שני שרתי http בעזרת המודול "http" כפי שניתן לראות, שני שרתים אלו בעצם יהיו מקושרים לחלק של הצ'אט החי והמשחק.

בשורות 22-27 ו-29-34 יצרתי שני משתנים אשר יורשים את מודול ה-"socket.io" אשר מאפשר לנהל אירועים בזמן אמת באופן דו-צדדי דבר

השימושי לצ'אט **חי** ומשחק **חי**. בנוסף אפשר להבחין שלצ'אט קישרתי את השרת שלו ולמשחק קישרתי שרת שלו גם כן. לאחר קישור השרתים הגדרתי את ה"cors" (cross origin resource sharing) כיוון שהשרתים שאחראיים על הצ'אט והמשחק שונים צריך להגדיר שהשרת הראשי יוכל לגשת אליהם לכן הגדרתי את המקור בתור השרת הראשי ואת המתודות שיאפשרו גישה אשר הם GET ו-POST כיוון שלא היה שימוש במתודות אחרות בפרויקט.

בשורות 36-38 הגדרתי את המשתנים של ערכי port שהשרתים יאזינו בהם.

לשרת הראשי הגדרתי שיישתמש במשתנה סביבתי (environment variable) ורק אם הוא לא מוגדר השרת יאזין לport 8080. לשני השרתים הנוספים הגדרתי ששרת הצ'אט יאזין בport 3000 ושרת המשחק יאזין לport 5555.

```

40 //Static files
41 app.use('/css', express.static(path.join(__dirname, '/css')));
42 app.use('/js', express.static(path.join(__dirname, '/javascript')));
43 app.use('/img', express.static(path.join(__dirname, '/images')));
44 app.use(express.json());
45 app.use(express.urlencoded({ extended: false }));
46 app.use(cookieParser());
47 app.set('view engine', 'ejs');
48 app.use(resGetters);
49 app.use(authRoutes);
50
51 chat_socket(chat_io);
52 game_socket(game_io);
53
54 //Listening as a server on a port
55 //app.listen(PORT, () => console.log('Listening on port ' + PORT));
56 chat_server.listen(CHAT_PORT, () => console.log('Chat is listening on port ' + CHAT_PORT));
57 game_server.listen(GAME_PORT, () => console.log('Game is listening on port ' + GAME_PORT));
58 app.listen(PORT, () => console.log('Listening on port ' + PORT));
59
60
61 main();

```

כאן הפונקציה הראשית נגמרת בשורה 59 ובשורה 61 הפעולה היחידה היא לקרוא לה, כיוון שרק היא רצה החלטתי לקרוא לפונקציה ראשית. בנוסף היא זו שמאתחלת, יוצרת ומפעילה את השרתים.

לפני שהפונקציה נגמרת היא עושה עוד כמה דברים חשובים בין השורות 41 ל-58.

בשורה 41 מוגדר שברגע שיש קריאה מהמשתמש השווה ל-"css/" אפליקציית האקספרס תלך ותביא את הקובץ מתיקיית ה-"css/" אשר נמצאת בתוך התיקייה הנוכחית. אותו הדבר בדיוק מוגדר בשני השורות הבאות (42 ו-43) אשר מגדירות את הגישה לקבצי התמונות וה-javascript.

בשורה 44 הגדרתי שהאפליקצייה תדע לקבל בקשת POST מהמשתמש בעלת Content-type: application/json (סוג המידע מגיע בצורת JSON).

בשורה 45 הגדרתי את אותו הדבר רק בנוגע לסוג מידע שונה אשר נקרא "x-www-form-urlencoded". אופציית ה-extended: false אומרת שהניתוח ייתבצע באמצעות ספרייה בשם querystring ולא ב-qs כאשר ההבדל ביניהם זאת תוצאת הניתוח ומה הם יכולים לנתח.

בשורה 46 אני למעשה מגדיר שהאפליקציה תוכל להשתמש בעוגיות (cookies).

בשורה 47 אני מגדיר מנוע צפייה אשר נקרא ejs והוא למעשה נותן את האפשרות להוסיף בדפי HTML את שפת התכנות javascript.

בשורה 48 ו-49 הגדרתי שהאפליקציה תשתמש בניתובים שהגדרתי לה בשני הקבצים.

בשורות 51 ו-52 אני מפעיל את שני הקבצים בעלי פונקציה עיקרית לשניהם אשר צריכה לקבל אינסטנס של שרת socket.io. לכן אני מפעיל את הפונקציה הראשית בקצים אלו ומביא להם את שרתי socket.io המתאימים לכל קובץ.

ולבסוף בשורות 56,57 ו-58 אני מגדיר שהשרתים יאזינו בפורטים שהגדרתי מקודם במשתנים.

הסבר על קובץ send_mails.js:

```

1 require('dotenv').config();
2 const nodemailer = require('nodemailer');
3 const sender = "teamjarsender@gmail.com";
4
5 const sendActivationMail = (receiver, activationToken) => {
6
7     const transporter = createTransporter();
8
9     const data = {
10         from: sender,
11         to: receiver,
12         subject: "Account Activation",
13         html: `
14             <h2>Please click on the button below to verify your account</h2>
15             <form action="${process.env.BASE_URL}/auth/activate/${activationToken}" method="GET">
16                 <button type="submit" style="background-color: green; border: none; border-radius: 5px; color: white; font-size: 20px;">
17                     </form>
18             `;
19     };

```

קובץ זה אחראי על שליחת האימיילים למשתמשים בהתאם הצורך.

בשורה 1 אני מייבא את משתני המערכת שם יש את הסיסמה לאפליקציה אשר נותנת גישה למייל השולח.

בשורה 2 אני מייבא מודול אשר עוזר בפעולת שליחת האימיילים.

בשורה 3 הגדרתי משתנה בשם שולח אשר ערכו צריך להיות האימייל השולח.

בשורה 5 יצרתי פונקציה השמורה ב `sendActivationMail`, פונקצייה זו שולחת את מייל אישור החשבון ומקבלת את המייל שצריך לקבל את האימייל וטוקן אישור.

בשורה 7 אני יוצר משתנה אשר ערכו הוא מה שהפונקציה `createTransporter()` מחזירה.

פונקציית `createTransporter()`:

```

57 function createTransporter(){
58     const transporter = nodemailer.createTransport({
59         host: "smtp.gmail.com",
60         port: 465,
61         secure: true,
62         auth: {
63             user: sender,
64             pass: process.env.EMAIL_APP_PASSWORD
65         }
66     });
67     return transporter;
68 }

```

ניתן לראות שפונקציה זאת מחזירה את האובייקט שאחרי על הגדרות ויצירת ה"מוביל" של המייל. אני משתמש כאן במודול `nodemailer` כדי ליצור את המוביל.

"Host: "smtp.gmail.com" כאן אני מגדיר שימוש בשרת האימיילים של gmail(google).

Port: 465 בחרתי להשתמש בפורט זה כיוון שהוא שימושי לשליחת מיילים.

Secure: true אופצייה זו גורמת להתחברות לשרת להתקיים באמצעות הצפנה בעלת השם TLS.

Auth שם מוגדרים שם השולח שבמקרה זה ערך השולח נמצא במשתנה `sender` והסיסמה לאפליקציה זו שמורה במשתנה מערכת.

בחזרה לפונקציה `sendActivationMail`:

משתנה המידע (`data`) מכיל את פרטי המייל עצמו.

From – מייל השולח.

To – מייל שצריך לשלוח אליו.

Subject – נושא המייל.

Html – זהו ה-html שיישלח למשתמש.

יש כותרת, וטופס למטרת GET לכתובת שהיא
"localhost:8080/auth/activate/activationToken"

קריאת GET זו תקרה רק כאשר המשתמש יילחץ על כפתור ה-submit.

```
47     const err = sendMail(transporter, data);
48
49     if(err){
50         return true;
51     }else{
52         return false;
53     }
```

בהמשך הפונקציה ניתן לראות פונקציה בשם sendMail אשר מקבלת "מוביל" אשר יצרתי והגדרתי מקודם ומידע (מה לשלוח) אשר הגדרתי מקודם גם כן.

אם לאחר הקריאה לפונקציה משתנה השגיאה (err) לא ריק תחזיר true(הייתה שגיאה) אחרת תחזיר false(לא הייתה שגיאה).

פונקציית sendMail:

```
70 function sendMail(transporter, data){
71     transporter.sendMail(data, function(err){
72         return err;
73     });
74 }
```

פונקצייה זו משתמשת ב"מוביל"(transporter) שהיא מקבלת ושולחת את המידע שהיא מקבלת. לאחר מכן יש פונקציה שמקבלת שגיאות בשליחה(אם היו) בתוך הפונקציה אני מחזיר את השגיאות. אחרי הקריאה לפונקציה אני מחזיר אמת או שקר בהתאם למה שחזר כפי שניתן לראות בתמונה הקודמת. אותו קונספט חוזר על עצמו בשליחת מייל של שחזור הסיסמה רק ששם הדבר השונה הוא המידע שנשלח.

```

38   subject: "Reset Your Password",
39   html: `
40     <h2>Click on the button below to reset your password!</h2>
41     <form action="${process.env.BASE_URL}/auth/reset_password/${resetPasswordToken}" method="GET">
42       <button type="submit" style="background-color: green; border: none; border-radius: 5px; color: white;>Reset Password</button>
43     </form>
44   `

```

ניתן לראות שהנושא וה-HTML שונים. ה-HTML שונה כיוון שעכשיו ישנו טופס שייקרא קריאת GET לכתובת הבאה
 "localhost:8080/auth/reset_password/resetPasswordToken"

```

76 module.exports = { sendActivationMail: sendActivationMail, resetPasswordMail: resetPasswordMail };

```

באמצעות שורה זו אני יכול לייבא את המודול והפונקציות הכתובות בו, במקרה זה ישנן רק שני פונקציות.

הסבר על קובץ create_auth_tokens.js:

```

1  require('dotenv').config();
2  const jwt = require('jsonwebtoken');
3
4  const createAuthTokens = (user_id, username, count, user_type) => {
5    const accessToken = jwt.sign({
6      user_id: user_id,
7      username: username
8    }, process.env.ACCESS_TOKEN_SECRET, {
9      expiresIn: "15m"
10   }); // Token is valid for 15 minutes
11   const refreshToken = jwt.sign({
12     user_id: user_id,
13     count: count,
14     user_type: user_type
15   }, process.env.REFRESH_TOKEN_SECRET, {
16     expiresIn: "7d"
17   }); // Token is valid for 7 days
18   return {refreshToken, accessToken};
19 }

```

כאן אני מייבא את משתני המערכת ואת מודול ה-"jsonwebtoken" אשר נותן אפשרות להשתמש ב-jwt-token, טוקן זה הוא בעצם JSON עם הגדרות והוא מוצפן באמצעות מפתח שמאשר שהטוקן אכן נכון. אם אננו מאושר על ידי המפתח שהוא נוצר איתו סימן שאינו נכון והוא שונה או שפג תוקפו.

ניתן לראות שבפונקציה createAuthTokens אני מקבל את כל הפרטים ההכרחיים ואיתם אני יוצר שני טוקנים בעלי ערכים שונים שהם שומרים בתוכם, מוצפנים ומוגדרים עם שני סודות שונים השמורים במשתני הסביבה וטווח זמן התוקף שלהם שונה אחד מהשני. לבסוף אני מחזיר את שני הטוקנים.

```

21  const createResetToken = (user_id, email) => {
22      const reset_password_token = jwt.sign({
23          user_id: user_id,
24          email: email
25      }, process.env.RESET_PASSWORD_TOKEN_SECRET, {
26          expiresIn: "15m"
27      }); // Token is valid for 15 minutes
28      return reset_password_token;
29  }
30
31  const createActivateAccountToken = (email, username, password) => {
32      const activate_account_token = jwt.sign({
33          email: email,
34          username: username,
35          password: password
36      }, process.env.ACTIVATE_ACCOUNT_TOKEN_SECRET, {
37          expiresIn: "15m"
38      });
39      return activate_account_token;
40  }

```

קונספט זה חוזר על עצמו בפונקציות כאשר ההבדל הוא מה שהם מקבלות, הסוד השמור במשתני המערכת וטווח הזמן עד שהם יהפכו להיות פגי תוקף.

```

41  module.exports = { createAuthTokens: createAuthTokens, createResetToken: createResetToken, createActivateAccountToken: createActivateAccountToken };

```

כמו בקובץ `send_mails.js` אני כתבתי שורה אשר בזכותה אני יכול לייבא את המודול ואיתו את הפונקציות לייצור הטוקנים השונים.

הסבר על קובץ `refreshTokens.js`:

```

1  const jwt = require('jsonwebtoken');
2  const spawn = require('child_process').spawn;
3  const path = require('path');
4  require('dotenv').config();
5  const createAuthTokens = require('../create_auth_tokens');
6  // const invalidateTokens = require('../middleware/invalidateTokens');
7
8  function pythonScript(data, filename){
9      return new Promise((resolve, reject) => {
10         spawn('python', [path.join(__dirname + "../python/" + filename).toString(), JSON.stringify(data)]).st
11         resolve(data.toString().replace(/\r\n/gm, ""));
12     });
13 }
14
15

```

בשורות 1-5 אני מייבא את המודולים הבאים: "jsonwebtoken" (הסברתי עליו מקודם), "child_process" נותן לי אפשרות להרצת תהליך חדש(כפי שניתן לראות בפונקציית `pythonScript` אני משתמש בו על מנת להריץ תהליך עם

הפקודה "python + dirname + ../../python/ + filename" אשר מריצה את קובץ הפייתון שהעברתי לפונקצייה ומעבירה את הנתונים שהיא קיבלה בתור JSON לארגומנטים כך שהראשון הוא הפקודה והשני הוא המידע לאחר מכן הפונקציה מחזירה את הפלט שהיה בקובץ הפייתון במידה ולא היו שגיאות בקריאה (אילו), מודול ה-"path" אשר הרחבתי עליו מקודם גם כן, משתני המערכת שהגדרתי ומודול יצירת הטוקנים שיצרתי.

בשורות 8-14 יש את פונקציית הpythonScript אשר הסברתי עליה מקודם.

```

16 module.exports = async(req, res, next) => {
17   accessToken = req.cookies['access-token'];
18   refreshToken = req.cookies['refresh-token'];
19   if(!accessToken || typeof accessToken === 'undefined' || accessToken === null) && (!refreshToken || typeof refresh
20     return res.sendStatus(401);
21 }
22
23 try{
24   const accessTokenDecoded = jwt.verify(accessToken, process.env.ACCESS_TOKEN_SECRET);
25   req.userId = accessTokenDecoded.user_id;
26   return next();
27 }catch{}
28
29
30 if(!refreshToken){
31   return res.sendStatus(401);
32 }
33
34
35 var refreshTokenDecoded;
36
37 try{
38   refreshTokenDecoded = jwt.verify(refreshToken, process.env.REFRESH_TOKEN_SECRET);
39 }catch{
40   return res.sendStatus(401);
41 }

```

בשורה 16 אני יוצר פונקציה המקבלת req – בקשה, res – תגובה וnext- איתה אני יכול להמשיך לmiddleware הבא במידת הצורך. בפונקציה זו אוכל להשתמש באמצעות ייבוא המודול(הקובץ). פונקציה זו היא אסינכרונית כיוון שאני צריך להשתמש בפונקציית pythonScript ולפעול רק אחראי שקובץ הפייתון סיים את הרצתו והדפיס את הפלט המבוקש דבר זה אוכל להשיג באמצעות שימוש בawait ורק לאחר שמה שבתוך await הסתיים מה שבתוך ה-then ירוץ (await יכול להיות משומש בתוך פונקציות אסינכרוניות).

בשורות 17-18 אני קורא ושומר בשני משתנים שני עוגיות(cookies) בשם "access-token" ו-"refresh-token".

בשורה 19 אני בודק אם שניהם ריקים במידה וכן אני שולח את המשתמש לעמוד עם status שאומר שאין לו גישה(401).

בשורות 23-27 אני מנסה לבדוק אם ה"access-token" תקין ויש לו תוקף במידה ויש שם כישלון כלשהו נמשיך לרוץ לשורה 30 אך אם אין שגיאה מסוימת אני מגדיר שבתוך req.userId יהיה ערך ששווה ל-user_id של המשתמש ומיד לאחר

מכן "מודיע" בתור ה-middleware שסיימתי את הבדיקה והמשתמש יוכל לגשת לדף שהוא רצה בעזרת החזרת הnext.

בשורה 30 אני בודק אם יש refreshToken במידה ואין אני מחזיר למשתמש תגובה שאין לו גישה.

בשורות 35-41 אני בודק אם ה-refreshToken תקין במידה וכן אני ממשיך בקוד אך במידה ולא אני מחזיר תגובה למשתמש האומרת שאין לו גישה.

```

43     await pythonScript(refreshTokenDecoded.user_id, '../python/get_all_user_id.py').then(user => {
44         user_info = user
45     });
46     user_info = user_info.split(", ");
47     user_id = user_info[0].slice(1, user_info[0].length);
48     username = user_info[2].slice(1, -1);
49     count = user_info[4];
50     user_type = user_info[5];
51     if(!user_info || count !== refreshTokenDecoded.count){
52         return res.sendStatus(401);
53     }
54     const tokens = createAuthTokens.createAuthTokens(user_id, username, count, user_type);
55     res.cookie('access-token', tokens.accessToken, { maxAge: 1000 * 60 * 15, httpOnly: false }); // cookie
56     res.cookie('refresh-token', tokens.refreshToken, { maxAge: 1000 * 60 * 60 * 24 * 7, httpOnly: true });
57     req.userId = user_id;
58     return next();
59 }

```

בשורה 43 אני מריץ קובץ פייתון בשם "get_all_user_id.py" אשר מחזיר את כל פרטי המשתמש לפי מספר הזיהוי של המשתמש אשר נמצא בתוקן.

קובץ get_all_user_id.py:

```

1  from sql_control import usersTable
2  import os
3  import sys
4  import json
5
6  def main():
7      # return ['user_id', 'email', 'username', 'hashed_password', 'count']
8      users_table = usersTable(os.path.join('./', 'database.db'), 'users', 'user_id', 'email', 'username', 'password',
9      details = json.loads(sys.argv[1])
10     user_info = users_table.get_all_by_id(details)
11     print(user_info)
12
13 if __name__ == "__main__":
14     main()
15

```

כפי שניתן לראות ישנה פונקציה ראשית אשר יוצרת אובייקט של טבלת משתמשים ושומרת את המידע שהועבר לקובץ דרך args במקום השני(אינדקס מספר אחד), במקרה ספציפי זה המידע הוא מספר הזיהוי של המשתמש. לאחר מכן אני קורא לפונקציה אשר נכתבה בקובץ sql_control אשר מחזירה את כל הפרטים לפי מספר הזיהוי. לבסוף אני מדפיס את מה שאני רוצה "להחזיר".

בחזרה לקובץ refreshToken.js:

פרטי המשתמש נשמרים במשתנה בשם `user_info`. כיוון שהמידע חוזר בתור `String` של `tuple` אני מוחק ממנו את כל הדברים שלא צריכים להיות בשורות 46-50. לאחר מכן אני בודק אם יש פרטים על המשתמש במידה ואין אני מחזיר תגובה שאין גישה. בדיקה נוספת היא לבדוק אם מספר `count` ששמור ב `refreshToken` שווה ל `count` שיש במסד נתונים למשתמש אם איננו שווה אני מחזיר תגובה האומרת למשתמש שאין גישה. אם שני בדיקות אלה לא גרמו להיכנס לתוך `if statement` אפשר להמשיך. בהמשך אני משתמש בפונקציה `createAuthTokens` אשר נמצאת במודול `create_auth_tokens.js`. אני שומר את שני הטוקנים בתוך משתנה בשם `tokens` ומגדיר שני עוגיות (`cookies`) אחת לכל טוקן. שם העוגיה, ערך של העוגיה והגדרותיה כמו זמן התוקף שלה (`maxAge`) והאם אפשר לגשת אליה רק בתור השרת (`httpOnly`). לבסוף אני שומר בתוך `req.userId` את מספר הזיהוי של המשתמש ומחזיר את פונקציית `next` דבר הגורם לעבור ל `middleware` הבא אם יש או להמשיך לתגובה שהמשתמש צריך לקבל לפי הניתוב. כך דרך אגב אני בודק האם המשתמש התחבר כבר ממקודם. קובץ `invalidateTokens.js`:

קובץ זה מכיל בתוכו מודול בעל פונקציה יחידה ומטרתה היא לגרום להריסת העוגיות (`cookies`) והעלאת `count` באחד ועל ידי כך בעצם לגרום להתנתקות המשתמש.

```

1  const path = require('path');
2  const spawn = require('child_process').spawn;
3
4  function pythonScript(data, filename){
5      return new Promise((resolve, reject) => {
6          spawn('python', [path.join(__dirname + '/../python/' + filename).toString(), data]).stdout.on('data', (data) => {
7              resolve(data.toString().replace(/[\r\n]+/gm, ""));
8          });
9      });
10 }
11
12 module.exports = async(req, res) => {
13     if(!req.userId){
14         return res.sendStatus(401);
15     }else{
16         let user_info;
17         await pythonScript(req.userId, '../python/get_all_user_id.py').then(user => {
18             user_info = user;
19         });
20         if(!user_info){
21             return res.sendStatus(403);
22         }
23         user_info = user_info.substring(1, user_info.length-1).split(", ");
24         await pythonScript(user_info[0].toString() + "," + (Number(user_info[4]) + 1).toString(), '../python/add');
25         if(data == "true"){
26             res.clearCookie('access-token');
27             res.clearCookie('refresh-token');
28             res.redirect('http://localhost:8080/login');
29         }else{
30             return res.sendStatus(500);
31         }
32     }
33 }

```

אני מייבא את `path` ו-`child_process` בשביל פונקציית `pythonScript` אשר הרחבתי עליה מקודם.

תחילה הפונקציה בודקת אם יש ב-req.userId ערך מסוים במידה ולא היא מחזירה תגובה של אין גישה, אחרת קובץ הפייתון get_all_user_id.py פועל ומחזיר את כל פרטי המשתמש לפי מספר הזיהוי שלו. אם אין פרטים אז אין גישה, אך אם יש פרטים ממשיכים. שורה 23 מטפלת בבעיה שהוסברה לפני כבר. אם יש פרטים אני מפעיל את קובץ הפייתון add_one_to_user_count_by_id.py קובץ זה מעלה באחד את count של המשתמש ושומר את שינוי זה במסד הנתונים כפי שניתן לראות.

```

1 import os
2 import sys
3 import json
4 from sql_control import usersTable
5
6
7 def main():
8     try:
9         users_table = usersTable(os.path.join('./', 'database.db'), 'users', 'user_id', 'email', 'username', 'password')
10        user_info = sys.argv[1].split(',')
11        # user_info => [user_id, count]
12
13        users_table.set_count_by_id(user_info[0], user_info[1])
14        print("true")
15    except:
16        print("false")
17
18
19 if __name__ == "__main__":
20     main()

```

אם מודפס true סימן שפעולה זו התבצעה בהצלחה אך אם הייתה שגיאה היא תדפיס false.

אם החזירה true אני מוחק את שני העוגיות (refresh-token – cookies ו-access-token ולבסוף מפנה את המשתמש לניתוב הכניסה).

:Auth.js

```

1 const router = require('express').Router();
2 const { body, validationResult } = require('express-validator');
3 const spawn = require('child_process').spawn;
4 const createAuthTokens = require('../create_auth_tokens');
5 const sendMails = require('../send_mails');
6 const jwt = require('jsonwebtoken');
7 const path = require('path');
8
9
10 function pythonScript(body, filename){
11     return new Promise((resolve, reject) => {
12         spawn('python', [path.join(__dirname + '/../python/' + filename).toString(), JSON.stringify(body)]).stdout.on('data', (data) => {
13             resolve(data.toString().replace(/\r\n/g, ""));
14         });
15     });
16 }

```

אני יוצר Router בקובץ נפרד server.js בשביל שהקוד יהיה יותר מסודר, Router זה נותן לי את האפשרות לכתוב את הניתובים בקובץ זה ולאחר מכן לייבא אותם ב-server.js. מודול ה-"express-validator" מוודא שהפרטים שהמשתמש הכניס מתאימים לקריטריונים מסוימים כמו אורך הסיסמה 6 תווים וכו'. שאר המודולים הוסברו מקודם באופן מפורט.

פונקציית pythonScript הוסברה גם כן.

```

19 router.post('/register', [
20   body('email').isEmail(),
21   body('password').isLength({ min:6 })
22 ],
23 (req, res) => {
24   const errors = validationResult(req);
25   if(!errors.isEmpty()){
26     res.send(errors);
27   }else{
28     if(req.body.password == req.body.verifyPassword){
29       // pythonScript(req.body, '../python/sign_up.py').then(
30       //   is_user_existing => {
31       //     if(is_user_existing == "true"){
32       //       res.redirect('/login');
33       //     }else{
34       //       res.redirect("/sign_up");
35       //     }
36       //   });
37     pythonScript([email: req.body.email, username: req.body.username ], "../python/check_user_existence.py").then(user_data =>
38       var user_details = user_data.split(", "); // user_details => [email, id] || "false" if not exist
39     if(user_details == "false"){
40       const body = req.body;
41       const activate_account_token = createAuthTokens.createActivateAccountToken(body.email, body.username, body.password);
42       const isErrors = sendMails.sendActivationMail(body.email, activate_account_token);
43       if(isErrors){
44         res.sendStatus(500).end();
45       }else{
46         res.redirect("/activate_account");
47       }
48     }else{
49       res.redirect("/sign_up");
50     }
51   });
52 }else{
53   res.send("Passwords didn't match!");
54 }
55 }
56 });

```

יש כאן ניתוב לכתובת /register במתודת POST, ניתן לראות שבתור middleware אני משתמש ב-express-validator כדי לבדוק אם האימייל הוא באמת אימייל או שהסיסמה לפחות בעלת 6 תווים.

אני שומר את השגיאות במשתנה errors, במידה ויש שגיאות מסוימות אני מחזיר כתגובה שגיאות אלה אך במידה ואין הקוד ממשיך את תהליך ההרשמה. תחילה אני בודק אם שני הסיסמאות שוות אחת לשנייה במידה ולא אני מחזיר שהסיסמאות לא תואמות, במידה וכן אני מריץ קובץ פייתון שבודק אם המשתמש קיים אם הוא לא קיים אני יוצר טוקן התחברות ושולח את המייל אם כל התהליך עבר בהצלחה אני מעביר אותו לדף "הפעלה" שמודיע שנשלח מייל אישור חשבון.

```

58 router.get("/auth/activate/:activation_token", function(req, res, next){
59   const { activation_token } = req.params;
60   if(activation_token){
61     jwt.verify(activation_token, process.env.ACTIVATE_ACCOUNT_TOKEN_SECRET, function(err, decodedToken){
62       if(err){
63         res.status(400).send("Incorrect or expired link!");
64       }else{
65         pythonScript(decodedToken, '../python/sign_up.py').then(
66           is_user_existing => {
67             if(is_user_existing == "true"){
68               res.redirect('/login');
69             }else{
70               res.redirect("/sign_up");
71             }
72           });
73       }
74     });
75   }else{
76     res.sendStatus(500);
77   }
78 });

```

יש כאן ניתוב עם מתודת GET לכתובת הURL הניתונה כאשר activation_token הוא פרמטר כדי לאשר שהמשתמש רצה להירשם ולא פג תוקפו. אם הטוקן תקין המשתמש יירשם במסד הנתונים עם הנתונים שלו אשר שמורים בטוקן התקין.

```

80 router.post('/login', function(req, res, next){
81   pythonScript(req.body, '../python/sign_in.py').then(
82     user_info => {
83       user_info = user_info.split(',');
84       if(user_info[0] == "true"){
85         const {refreshToken, accessToken} = createAuthTokens.createAuthTokens(user_info[1], user_info[2], user_info[3], user_info[4]);
86         //res.json({ accessToken: accessToken, refreshToken: refreshToken })
87         res.cookie("refresh-token", refreshToken, { maxAge: 1000 * 60 * 60 * 24 * 7, httpOnly: true });
88         res.cookie("access-token", accessToken, { maxAge: 1000 * 60 * 15, httpOnly: false });
89         res.redirect("/dashboard");
90       }else{
91         // res.send("Invalid creds!");
92         res.render(path.join(__dirname + "../html/login.ejs"), { error_message: 'Error!', counter: 1 });
93         //res.redirect('/login');
94       }
95     }
96   );
97 });

```

יש כאן פנייה למתודת POST מטופס ההרשמה לURL /login, קובץ פייתון sign_in.py בודק אם פרטיו נכונים (האימייל והסיסמה) במידה וכן הוא שומר שני טוקנים בעוגיות ומפנה את המשתמש לdashboard.

```

99 router.post("/reset_password", [
100   body('username_or_email').isEmail()
101 ], (req, res) => {
102   const errors = validationResult(req);
103   const emailOrPassword = req.body.username_or_email;
104   if(!errors.isEmpty()){
105     // username
106     pythonScript([ username: emailOrPassword ], "../python/check_user_existence.py").then(user_data => {
107       var user_details = user_data.split(", "); // user_details => [email, id] || "false" if not exist
108       if(user_details != "false"){
109         const reset_password_token = createAuthTokens.createResetToken(user_details[1], user_details[0]);
110         const isErrors = sendMails.resetPasswordMail(user_details[0], reset_password_token);
111         if(isErrors){
112           res.sendStatus(500).end();
113         }else{
114           pythonScript([ user_id: user_details[1], reset_link: reset_password_token, operation: "update" ], "../python/manage
115             if(isUpdated == "true"){
116               res.end("Reset Password Mail Sent, go ahead and check it!");
117             }else{
118               res.sendStatus(500).end();
119             }
120           });
121         }
122       }else{
123         res.sendStatus(400).end();
124       }
125     });

```

מתודת POST לכתובת /reset_password אשר בודקת בעזרת המודול express-validator אם מה שהמשתמש הכניס הוא אימייל אם איננו הכניס אימייל אני בודק אם זה אחד משמות משתמשי המערכת במסד הנתונים.

אני בודק אם המייל קיים אם הוא קיים אני מחזיר את האימייל ומספר הזיהוי של המשתמש, לאחר מכן אני שולח מייל עם reset_password_token ואני שומר טוקן זה במסד הנתונים ומדפיס הודעה שמודיעה שנשלח מייל אם מייל לא נשלח אני מודיע שהייתה שגיאה בשרת.

```

152 router.get("/auth/reset_password/:reset_token", function(req, res, next){
153   const { reset_token } = req.params;
154   if(reset_token){
155     jwt.verify(reset_token, process.env.RESET_PASSWORD_TOKEN_SECRET, function(err, decoded_token){
156       if(err){
157         res.status(400).send("Incorrect or expired link!");
158       }else{
159         res.sendFile(path.join(__dirname, "../html/reset_password_itself.html"));
160       }
161     });
162   }else{
163     res.status(500).end();
164   }
165 });
166

```

מתודת GET אשר בודקת אם הטוקן תקין, אם הוא תקין אני מציג למשתמש את דף שחזור הסיסמה אם לא תקין אני שולח הודעה שהטוקן לא תקין או שפג תוקפו.

```

167 router.post("/auth/reset_password/:reset_token", [
168   body('password').isLength({ min:6 })
169 ], function(req, res, next){
170   const errors = validationResult(req);
171   if(!errors.isEmpty()){
172     res.send(errors);
173   }else{
174     if(req.body.password == req.body.verifyPassword){
175       const { reset_token } = req.params;
176       if(reset_token){
177         jwt.verify(reset_token, process.env.RESET_PASSWORD_TOKEN_SECRET, function(err, decodedToken){
178           if(err){
179             res.status(400).send("Incorrect or expired link!");
180           }else{
181             pythonScript({ user_id: decodedToken.user_id, reset_link: reset_token, operation: "compare" }, "../python/manag
182             if(status == "true"){
183               pythonScript({ user_id: decodedToken.user_id, new_password: req.body.password }, '../python/reset_passw
184               if(status == "true"){
185                 pythonScript({ user_id: decodedToken.user_id, reset_link: "", operation: "update" }, '../python
186                 if(status == "true"){
187                   res.redirect("/login");
188                 }else{
189                   res.sendStatus(500);
190                 }
191               });
192             }else{
193               res.sendStatus(500);
194             }
195           });
196         }else{
197           res.status(400).send("Incorrect or expired link!");
198         }
199       });

```

מתודת POST שבודקת שהסיסמה לפחות 6 תווים, אם שני הסיסמאות שוות אם
 הreset_token תקין, אם הוא נמצא במסד הנתונים, מאפס את הסיסמה עם
 הסיסמה החדשה ומוחק את הreset_token ממסד הנתונים כדי שלא יהיה מצב
 שמשתמש יכול לאפס את סיסמתו כמה פעמים עם אותו קישור/מייל.

קובץ res_getters.js:

```

1  const router = require('express').Router();
2  const path = require('path');
3  const spawn = require('child_process').spawn;
4  const jwt = require('jsonwebtoken');
5  const fs = require('fs');
6  const refreshTokens = require('../middleware/refreshTokens');
7  const imagemin = require("imagemin");
8  const mozjpeg = require("imagemin-mozjpeg");
9  const isJpg = require("is-jpg");
10 const sharp = require("sharp");
11 const invalidateTokens = require('../middleware/invalidateTokens');
12 const error_page = "http://localhost:8080/error";
13
14
15 function pythonScript(data, filename){
16   return new Promise((resolve, reject) => {
17     spawn('python', [path.join(__dirname + "../python/" + filename).toString(), data]).stdout.on('data', (data) => {
18       resolve(data.toString().replace(/[\r\n]+/gm, ""));
19     });
20   });
21 }
22
23
24 const getResponse = (url, path) => {
25   router.get(url, function(req, res, next){
26     if(req.cookies["refresh-token"] == null){
27       res.sendFile(path);
28     }else{
29       res.redirect("/dashboard");
30     }
31   });
32 }

```

יורש את המודולים express, path, child_process, jsonwebtoken, fs (בשביל
 ליצור ולכתוב תוכן לתוך קבצים למשל ליצור תמונה), refreshTokens,
 imagemin (להמיר באפר (buffer) של תמונה לגלגל imagemin-mozjpeg, (עושה jpg)

אופטימיזציה לתמונה תוך כדי הימנעות מהריסתה), js-jpg (יכול לבדוק אם תמונה היא sharp, jpg) (עוזר לשנות גודל ואיכות תמונה – גם קשור לאופטימיזציה),
invalidateTokens.

משתנה error_page שומר את הניתוב לדף השגיאה.

פונקציית pythonScript הוסברה בהתחלה וgetResponse מקבלת כתובת ניתוב וכתובת של מיקום קובץ אם המשתמש לא מחובר היא מביאה אותו לאן שהוא רוצה אך אם הוא מחובר היא מנתבת אותו לכתובת /dashboard.

```

34 const protectedGetResponseWithDetails = (url, path) => {
35   router.get(url, refreshTokens, function(req, res, next){
36     res.sendFile(path);
37   });
38 }
39
40 //Regular Get Responses
41 getResponse("/", path.join(__dirname + "../html/index.html"));
42
43 getResponse("/reset_password", path.join(__dirname + "../html/reset_password.html"));
44
45 getResponse("/activate_account", path.join(__dirname + "../html/activate_account.html"));
46
47 router.get("/login", function(req, res, next){
48   if(req.cookies["refresh-token"] == null){
49     res.render(path.join(__dirname + "../html/login.ejs"), { error_message: '', counter: 0 });
50   }else{
51     res.redirect("/dashboard");
52   }
53 });
54
55 getResponse("/sign_up", path.join(__dirname + "../html/sign_up.html"));
56
57 //Protected Get Routes Using Token To Authenticate
58 router.get("/dashboard", refreshTokens, function(req, res, next){
59   const decoded_refresh = jwt.verify(req.cookies["refresh-token"], process.env.REFRESH_TOKEN_SECRET);
60   if(parseInt(decoded_refresh.user_type) === 1){
61     res.sendFile(path.join(__dirname, "../html/admin_dashboard.html"));
62   }else{
63     res.sendFile(path.join(__dirname, "../html/dashboard.html"));
64   }
65 });
66

```

פונקציית protectedGetResponseWithDetails מקבלת את אותם הפרמטרים כמו getResponse אך שהיא משתמשת בתוכנת הביניים (middleware) אשר בודקת אם המשתמש מחובר, רק אם הוא מחובר יהיה לו גישה לנתיב והקובץ ירוץ (דף הHTML).

שורות 41-45 הן ניתובים רגילים שהמשתמש לא צריך להיות מחובר כדי לגשת אליהם.

בניתוב ל/login אני מריץ את הקובץ login.ejs בלי הודעת שגיאה וספירה שווה לאפס. אם הוא מחובר אני שולח אותו לdashboard.

שורה 55 ניתוב רגיל לכתובת url /sign_up.

בניתוב לdashboard אני בודק אם המשתמש מחובר אם הוא מחובר אני בודק אם הוא מנהל במידה וכן אני מציג את הדף של המנהלים (admin_dashboard) אחרת אני יציג את דף הdashboard הרגיל.


```

67 router.get("/add_regular_quest", refreshTokens, function(req, res, next){
68   const decoded_refresh = jwt.verify(req.cookies["refresh-token"], process.env.REFRESH_TOKEN_SECRET);
69   if(parseInt(decoded_refresh.user_type) === 1){
70     res.sendFile(path.join(__dirname, "../html/add_regular_quest.html"));
71   }else{
72     res.sendStatus(403);
73   }
74 });

```

להוספת משימה אני עושה את הבדיקה אם הוא מנהל, אם כן אני שולח אותו לדף אך אם לא אני שולח בתור תגובה שאין לו גישה.

```

88 //game
89 protectedGetResponseWithDetails["/game", path.join(__dirname + "../html/game.html")];
90
91
92 //logout
93 router.post("/logout", refreshTokens, function(req, res, next){
94   invalidateTokens(req, res);
95 });
96
97 router.post("/ggpc", refreshTokens, function(req, res, next){
98   if(req.cookies["questClue"] != null){
99     res.clearCookie('questClue');
100   }
101   chars = req.body;
102   res.cookie("game-char", chars.gc, { maxAge: 1000 * 60 * 60 * 5, httpOnly:true });
103   res.cookie("player-char", chars.pc, { maxAge: 1000 * 60 * 60 * 5, httpOnly:true });
104   res.end();
105 });
106
107 router.get("/get-chars", refreshTokens, function(req, res, next){
108   game_char = req.cookies['game-char'];
109   player_char = req.cookies['player-char'];
110   res.json({ gc: game_char, pc: player_char });
111 });
112

```

רק אם משתמש מחובר הוא יכול להיות מנותב ל/game(משחק).

רק למשתמש מחובר אמור להיות גישה ליציאה. ביציאה אני משתמש במודול invalidateTokens אשר גורם להתנתקות מלאה האתר.

ggpc/ אחראי להגדיר את העוגיות של המשחק הן של השחקן והן של הלובי של המשחק. בנוסף הוא בודק אם יש רמז ממשחק קודם, אם יש הוא מוחק אותו מהעוגיות.

get-chars/ קורא את הערכים שיש בעוגיות "game-char" ו-"player-char" ושולח אותם בתגובה בתור JSON (ניתוב זה קיים כיוון שhttpOnly הוגדר כאמת, זאת אומרת שרק לשרת יש גישה לעוגיות).

```

113 router.post("/get-quest", refreshTokens, function(req, res, next){
114   pythonScript(req.body.quest_id, '../python/get_quest.py').then(quest => {
115     // console.log(quest.question);
116     // console.log(quest.clue);
117     // console.log(quest.answer);
118     // [quest_id, quest, clue, answer, image_name]
119     if(quest !== "false"){
120       quest = quest.substring(1, quest.length-1).split(', ');
121       for(var i = 0; i < quest.length; i++){
122         // quest[i] = quest[i].replace(/"/g, '');
123         if(quest[i].length > 1){
124           quest[i] = quest[i].slice(1, -1);
125         }
126       }
127       quest.splice(2, 2);
128       // res.cookie('questId', quest[0], { maxAge: 1000 * 60 * 60 * 24, httpOnly: true });
129       res.send(quest);
130     }else{
131       res.sendStatus(500);
132     }
133   });
134 });
135
136 router.post("/check_answer", refreshTokens, function(req, res, next){
137   pythonScript(JSON.stringify(req.body), "../python/check_answer.py").then(status => {
138     if(status === "true"){
139       res.send("right");
140     }else{
141       res.send("wrong");
142     }
143   });
144 });
145

```

get-quest/ אחראי על הרצת קובץ פייתון בשם get_quest.py אשר הוא מחזיר שאלה(אני מוריד חלק פרטים מהשאלה כמו רמז ופתרון כדי שהמשתמש לא יוכל לגשת למידע זה) אם יש שגיאה מסוימת זאת שגיאה של השרת.

check-answer/ בודק את התשובה של המשתמש בעזרת קובץ פייתון בשם check_answer.py אם התשובה נכונה ושווה לתשובה שבמאגר הנתונים בטבלת שאלות, השרת יחזיר "right" אחרת הוא יחזיר "wrong".

```

146 router.get("/add_quest", refreshTokens, function(req, res, next){
147   const decoded_refresh = jwt.verify(req.cookies["refresh-token"], process.env.REFRESH_TOKEN_SECRET);
148   if(parseInt(decoded_refresh.user_type) === 1){
149     res.sendFile(path.join(__dirname, "../html/add_quest.html"));
150   }else{
151     res.sendStatus(403);
152   }
153 });
154
155 router.get("/remove_game_chars", refreshTokens, function(req, res, next){
156   res.cookie('game-char', { maxAge: 0 });
157   res.cookie('player-char', { maxAge: 0 });
158   res.clearCookie('game-char');
159   res.clearCookie('player-char');
160   res.end();
161 });
162

```

add_quest/ מציג למנהל בלבד את דף המעבר של הוספת השאלות.

remove_game_chars/ הורס ומוחק את העוגיות game-char ו-player-char (הסברתי מקודם למה אני יכול לגשת אליהם רק מהשרת).

```

163 router.post("/upload_quest_image", refreshTokens, function(req, res, next){
164   const decoded_refresh = jwt.verify(req.cookies["refresh-token"], process.env.REFRESH_TOKEN_SECRET);
165   if(parseInt(decoded_refresh.user_type) === 1){
166     // pythonScript([body.imageName, body.imageFile], "../python/upload_image_dpb.py").then(date => {
167     //   console.log(date);
168     // });
169     var body = req.body;
170     var image_type = body.imageType.split('/');
171     if(image_type[0] === "image" && (image_type[1] === "png" || image_type[1] === "jpeg" || image_type[1] === "gif")){
172       var data = body.imageFile.replace(/^data:image\/\w+;base64/, "");
173       var buf = Buffer.from(data, 'base64');
174       imagemin.buffer(buf, { plugins: [() => {
175         if(isJpg(buf)){
176           return buf;
177         }else{
178           return sharp(buf).jpeg().toBuffer();
179         }
180       }, mozjpeg({ quality: 85 })] }).then(miniBuf => {
181         fs.writeFileSync("./images/image.jpg", miniBuf, function(err){
182           if(err){
183             console.log(err);
184           }
185         });
186       });
187       pythonScript(JSON.stringify({ "operation": "upload", "imageName": body.imageName, "dpbSec": process.env.DPB_SECRET }), "/
188       if(status === "true"){
189         res.sendStatus(200);
190       }else{
191         res.sendStatus(500);
192       }
193     });
194   }else{
195     res.sendStatus(500);
196   }
197 }else{
198   res.sendStatus(403);
199 }
200 });

```

מתודת POST ל"/upload_quest_image" זו נגישה רק למנהל. תחילה אני בודק אם הקובץ הוא מטיפוס של תמונה אם כן אני מוחק את ההתחלה של הbuffer אשר שווה ל"data:image;bas64" ושומר עצם של buffer במשתנה buf. אני מעביר משתנה זה לפונקציית buffer של מודול imagemin ואת הפלאגינים (השינויים שאני רוצה לבצע בו). אם הוא כבר jpeg אני מחזיר אותו באותה צורה לפלאגין הבא אך אם הוא לא jpeg אני משתמש במודול sharp כדי להעביר אותו לjpeg ואז מחזיר את הbuffer. אחרי זה אני מוריד את איכות הקובץ מ100 ל85 כדי שהוא יהיה יותר קטן ואז אני כותב buffer זה לקובץ שנמצא בתיקיית תמונות בקובץ image.jpg. לבסוף אני מריץ קובץ פייתון בשם manage_images_dpb.py, הוא מקבל את המצב אשר במקרה זה הוא העלאה (upload), שם התמונה והסוד של הגישה לחשבון הdropbox שפתחתי השמור במשתני הסביבה. אם התהליך ההעלאה צלח אני מחזיר סטטוס הצלחה אך אם הוא כשל אני שולח סטטוס שגיאה בשרת.

```

202 router.post("/add_regular_quest", refreshTokens, function(req, res, next){
203   var body = req.body;
204   const decoded_refresh = jwt.verify(req.cookies["refresh-token"], process.env.REFRESH_TOKEN_SECRET);
205   if(parseInt(decoded_refresh.user_type) === 1){
206     pythonScript(JSON.stringify(body), "../python/add_quest.py").then( status => {
207       if(status === "true"){
208         res.send("http://localhost:8080/dashboard");
209       }else{
210         res.send(error_page);
211       }
212     });
213   }else{
214     res.sendStatus(403);
215   }
216 });
217
218 router.get('/error', function(req, res, next){
219   res.sendStatus(500);
220 });
221

```

מתודת ה-POST לכתובת /add_regular_quest בודקת אם המשתמש מנהל במידה וכן השאלה נשמרת במסד הנתונים בעזרת קובץ הפיתון add_quest.py. לבסוף אם הכל הצליח השרת מחזיר את המשתמש לדף ה-`dashboard`, אם התהליך כשל הייתה שגיאה בשרת.

/error עמוד שגיאה שמחזיר קוש שגיאה של השרת.

```

222 router.post("/get_clue", refreshTokens, function(req, res, next){
223   if(!req.cookies['questClue']){
224     pythonScript(req.body.quest_id, '../python/get_quest.py').then(question => {
225       // question => [quest_id, quest, clue, answer, imageName]
226       if(question !== "false"){
227         quest = question.substring(1, question.length-1).split(', ');
228         for(var i = 0; i < quest.length; i++){
229           quest[i] = quest[i].replace(/'/g, '');
230         }
231         res.cookie('questClue', quest[2], { maxAge: 1000 * 60 * 60 * 24, httpOnly: true });
232         res.send(quest[2]);
233       }else{
234         res.sendStatus(500);
235       }
236     });
237   }else{
238     res.send(req.cookies["questClue"]);
239   }
240 });
241
242 router.post("/get_image", refreshTokens, function(req, res, next){
243   var body = req.body;
244   if(body !== null){
245     pythonScript(JSON.stringify({ "operation": "read", "imageName": body.imageName, "dpsSec": process.env.DPB_SECRET }), "../python/
246       if(status.split(",")[0] === "true"){
247         res.send(status.split(",")[1].slice(2, -1));
248       }else{
249         res.sendStatus(500);
250       }
251     });
252   }else{
253     res.sendStatus(500);
254   }
255 });

```

מתודת ה-POST לכתובת /get_clue בודקת אם יש רמז, אם אין היא בודקת מה הרמז שבמסד הנתונים בעזרת קובץ הפיתון get_quest.py בשם אשר מקבל את מספר הזיהוי של השאלה ומחזיר את כל פרטיה כולל הרמז, מגדיר עוגייה של רמז ושולח אותו למשתמש, אם יש כבר עוגייה קיימת הרמז שכתוב בה נשלח ללא פרוצדורת חיפוש השאלה במסד הנתונים.

מתודת ה-POST של כתובת /get_image מריצה את קובץ הפיתון manage_images_dpb.py עם פעולה של קריאה (read), עכשיו הקובץ יחזיר את הדבר הבא – "true, b'image", אם יש לי true לפני הפסיק סימן שהקובץ נקרא בהצלחה כל מה שנשאר הוא להחזיר כתגובה את התמונה, אם היה כשל מסוים אני מחזיר שהייתה שגיאה בשרת.

```

257 router.get("/get_clue", refreshToken, function(req, res, next){
258     if(req.cookies["questClue"] != null){
259         res.send(req.cookies["questClue"]);
260     }else{
261         res.send(null);
262     }
263 });
264
265 router.get("/remove_clue", refreshToken, function(req, res, next){
266     res.clearCookie("questClue");
267     res.end();
268 });
269
270 // router.get("/get_quest_id", function(req, res, next){
271 //     res.send(req.cookies["questId"]);
272 // });
273
274 module.exports = router;

```

/get_clue מחזיר רמז אם יש עוגייה של רמז אחרת הוא מחזיר null (ריק).

/remove_clue מוחק את העוגייה של הרמז.

קובץ Index_logic.js:

```

1  const scrolling = () => {
2      const header = document.querySelector('.main-header');
3      window.addEventListener('scroll', () => {
4          const scrollPos = window.scrollY;
5          if(scrollPos > 10){
6              header.classList.add('scrolled');
7          }else{
8              header.classList.remove('scrolled');
9          }
10     });
11 }
12
13 const main = () => {
14     scrolling();
15 }
16
17 main();
18

```

כאשר גוללים את העמוד למטה בנקודה שערך ה־scrollY של העמוד גדול מ-10 מתווספת מחלקה בשם scrolled ברגע שתנאי זה לא מתקיים היא נמחקת. בקובץ ה־css הגדרתי שאת הצבע של scrolled. קובץ זה מוסיף אלמנט של עיצוב כאשר גוללים את העמוד.

בקובץ find_game_client.js יש חלק מרכזי:

```

23 game_socket.on('found-game', () => {
24     clearInterval(timer);
25     localStorage.removeItem('startTime');
26     game_socket.emit("load-game");
27 });
28
29 game_socket.on('set-room-char', (chars) => {
30     $.get("/remove_game_chars");
31     $.when(post_chars(chars)).done(function(){
32         window.location.replace("http://localhost:8080/game");
33     });
34     // window.location.replace("http://localhost:8080/game");
35 });
36
37 function post_chars(chars){
38     return $.post("/ggpc", {
39         gc: chars.gc,
40         pc: chars.pc,
41     });
42 }

```

אם game_handler מצא משחק השעון עוצר, הזמן נמחק מהזיכרון המקומי וה־socket של ה־client משדר אליו שייטען את המשחק.

קבצי הHTML:Index.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <link rel="preconnect" href="https://fonts.gstatic.com">
5     <link href="https://fonts.googleapis.com/css2?family=Nunito&display=swap" rel="stylesheet">
6     <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
7     <link rel="stylesheet" type="text/css" href="css/index_style.css"/>
8     <meta name="viewport" content="width=device-width, initial-scale=1.0">
9     <title>INDEX</title>
10  </head>
11  <body>
12    <header class="main-header">
13      <div class="logo">
14        <a href="/">LGCW</a>
15      </div>
16
17      <input type="checkbox" class="menu-btn" id="menu-btn">
18      <label for="menu-btn" class="menu-icon">
19        <span class="menu-icon-line"></span>
20      </label>
21
22      <ul class="nav-links">
23        <li class="nav-link">
24          <a href="login">Login</a>
25        </li>
26        <li class="nav-link">
27          <a href="sign_up">Sign Up</a>
28        </li>
29      </ul>
30    </header>
31
32    <div class="slogo">
33      <h1>LetGetCodeWars!</h1>
34    </div>
35    <script type="text/javascript" src="js/index_logic.js"></script>
36  </body>
37 </html>

```

בhead יש את הכותרת, את הפונטים ואת קובץ הcss. שורה 8 אחראית על המימדים והגודל של העמוד.

בbody יש את header עם הלוגו ושני קישורים login ו-sign up. שורות 17-20 קשורות לתצוגה של הטלפון (שלוש הפסים). אחרי header יש את הכותרת הראשית ואת קובץ הjavascript.

:Login.ejs

```

32 <div class="login-form">
33 <img src="">
34 <h1>Login Now!-</h1>
35 <form action="login" method="post">
36 <label class="field-label">
37 <input type="email" class="input-box" name="email" required>
38 <div class="label-text">Email</div>
39 </label>
40 <label class="field-label">
41 <input type="password" class="input-box" name="password" required>
42 <div class="label-text">Password</div>
43 </label>
44 <p><%= error_message %></p>
45 <br>
46 <button type="submit" class="login-button">Login</button>
47 <hr>
48 <p class="or">OR</p>
49 <a href="sign_up"><button type="button" class="sign-up-button">Sign Up</button></a>
50 <p class="reset-password">Forgot your password? <a href="/reset_password">Reset It!</a></p>
51 </form>
52 </div>
53 <script type="text/javascript" src="js/index_logic.js"></script>
54 </body>
55 </html>

```

ה head וה header של index.html הוא אותו דבר חוץ מהכותרת של העמוד וקובץ ה css שמיובא.

יש כאן טופס כניסה עם מייל וסיסמה. במידה ויש שגיאה תופיע הודעת error כפתור להפעלת פעולת ה POST של הטופס לכתובת login. יש עוד שני כפתורים/קישורים להרשמה ולאיפוס סיסמה. לבסוף את אותו קובץ ה javascript שהיה ב index.html.

:Sign_up.html

```

32 <div class="sign-up-form">
33   <img src="">
34   <h1>Sign Up Now:-</h1>
35   <form action="register" method="POST">
36     <label class="field-label">
37       <input type="email" class="input-box" name="email" id="email" required>
38       <div class="label-text">Email</div>
39     </label>
40     <label class="field-label">
41       <input type="text" class="input-box" name="username" id="username" required>
42       <div class="label-text">Username</div>
43     </label>
44     <label class="field-label">
45       <input type="password" class="input-box" name="password" id="password" required>
46       <div class="label-text">Password</div>
47     </label>
48     <label class="field-label">
49       <input type="password" class="input-box" name="verifyPassword" id="verifyPassword" required>
50       <div class="label-text">Verify Password</div>
51     </label>
52     <br>
53     <button type="submit" class="sign-up-button">Sign Up</button>
54     <p id="error-message" style="display:none"></p>
55     <hr>
56     <p class="or">OR</p>
57     <p class="login-account">If you already have an account!</p>
58     <a href="login"><button type="button">Login</button></a>
59   </form>
60 </div>
61 <script type="text/javascript" src="js/index_logic.js"></script>
62 </body>
63 </html>

```

ה head וה header של index.html הוא אותו דבר חוץ מהכותרת של העמוד וקובץ הcss שמיובא.

טופס עם שדות של מייל, שם משתמש, סיסמה ואישור סיסמה(לזכור שהמשתמש יודע באמת את סיסמתו והוא יכול לרשום אותה פעם נוספת). כפתור הירשמות להפעלת פעולת הPOST של הטופס לכתובת register. בנוסף יש כפתור התחברות. לבסוף את אותו קובץ הjavascript שהיה בindex.html.

:Activate account.html

מופיע לאחר ההרשמה ומודיע לאשר אותה דרך האימייל שנשלח.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>ACTIVATE ACCOUNT</title>
5   </head>
6   <body>
7     <h2>Please Activate Your Account via the Email That sent to you!</h2>
8   </body>
9 </html>

```

:Dashboard.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <link rel="preconnect" href="https://fonts.gstatic.com">
5     <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
6     <link rel="stylesheet" type="text/css" href="css/chat.css">
7     <link rel="stylesheet" type="text/css" href="css/dashboard.css">
8     <meta name="viewport" content="width=device-width, initial-scale=1.0">
9     <meta http-equiv="X-UA-Compatible" content="ie=edge">
10    <title>DASHBOARD</title>
11    <script defer src="http://localhost:3000/socket.io/socket.io.js"></script>
12    <script defer src="js/chat_client.js"></script>
13    <script defer src="js/dashboard.js"></script>
14    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
15  </head>
16  <body>
17    <header class="main-header">
18      <div class="logo">
19        <a href="/">LGCW</a>
20      </div>
21      <!-- <button id="logout" type="button" onclick="logout()">logout</button> -->
22      <form action="logout" method="post">
23        <button type="submit" id="logout" onclick="logout()">logout</button>
24      </form>
25    </header>
26    <a href="finding_game"><button type="button" id="find-game">Find Game</button></a>
27    <div id="chat">
28      <div id="chat-title">
29        <h2>Live Chat</h2>
30        <button type="button" onclick="updateScroll()">Scroll to End</button>
31      </div>
32      <div id="message-container" onscroll="isBottom()"></div>
33      <form id="send-container">
34        <input type="text" id="message-input">
35        <button type="submit" id="send-button">Send!</button>
36      </form>
37    </div>
38  </body>
39 </html>

```

מייבא בhead את הפונטים, קבצי css, קבצי javascript.

בbody יש את הלוגו וכפתור יציאה שעובד כמו טופס בשביל להפעיל את פעולת ה POST לכתובת logout בheader, יש כפתור למציאת משחק שמוביל לדף מציאת משחק. בנוסף יש את הצ'אט החי בעמוד עם כפתור שליחה וגרירה לסוף הצ'אט.

:Admin_dashboard.html

```

<header class="main-header">
  <div class="logo">
    <a href="/">LGCW</a>
  </div>
  <!-- <button id="logout" type="button" onclick="logout()">logout</button> -->
  <button type="button" id="add-quest">Add Quest</button>
  <form action="logout" method="post">
    <button type="submit" id="logout" onclick="logout()">logout</button>
  </form>
</header>

```

ההבדל היחידי הוא שבדף המנהל יש בheader כפתור שמפנה לדף הוספת שאלה.

:Finding_game.html

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <link rel="preconnect" href="https://fonts.gstatic.com">
5   <link href="https://fonts.googleapis.com/css2?family=Nunito&display=swap" rel="stylesheet">
6   <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
7   <link rel="stylesheet" type="text/css" href="css/finding_game.css">
8   <meta name="viewport" content="width=device-width, initial-scale=1.0">
9   <title>Finding Game</title>
10  <script defer src="http://localhost:5555/socket.io/socket.io.js"></script>
11  <script defer src="js/find_game_client.js"></script>
12  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
13 </head>
14 <body onload="sendUsername()">
15   <header class="main-header">
16     <div class="logo">
17       <a href="/">LGCW</a>
18     </div>
19   </header>
20   <div id="container">
21     <div id="timer">
22       <ul>
23         <li><span id="hours"></span>Hours</li>
24         <li><span id="colon">:</span></li>
25         <li><span id="minutes"></span>Minutes</li>
26         <li><span id="colon">:</span></li>
27         <li><span id="seconds"></span>Seconds</li>
28       </ul>
29     </div>
30   </div>
31 </body>
32 </html>

```

בhead מייבא את הפונטים, קובץ css וjavascript.

בbody יש את הheader עם הלוגו וטיימר עם שעות דקות ושניות.

:Game.html

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>GAME</title>
5      <link rel="preconnect" href="https://fonts.gstatic.com">
6      <link href="https://fonts.googleapis.com/css2?family=Nunito&display=swap" rel="stylesheet">
7      <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
8      <link rel="stylesheet" type="text/css" href="css/game.css"/>
9      <meta name="viewport" content="width=device-width, initial-scale=1.0">
10     <script defer src="http://localhost:5555/socket.io/socket.io.js"></script>
11     <script defer src="js/game_client.js"></script>
12     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
13   </head>

```

מייבא את הפונטים, קובץ css וקבצי javascript.

```

15   <header class="main-header">
16     <div class="logo">
17       <a href="/">LGCW</a>
18     </div>
19   </header>
20   <div id="game-container">
21     <div id="player-status">
22       <div id="bombs-status">
23         <p id="object-text">Bombs</p>
24         <div>
25           
26           <p id="colon">:</p>
27           <p id="number-bombs">0</p>
28         </div>
29       </div>
30       <div id="health-status">
31         <p id="object-text">Health</p>
32         <div>
33           
34           <p id="colon">:</p>
35           <!--  -->
36           <!-- <div id="health-progress-bar-placement"> -->
37           <!--  -->
38         </div>
39
40         <div id="health-progress-container">
41           <div id="health-progress">
42             <div id="health-back">
43               <div id="bar">
44                 </div>
45               <p id="health-text">100%</p>
46             </div>
47             
48           </div>
49         </div>
50       </div>

```

בbody יש את הלוגו בheader, מצב השחקן שם יש את מספר הפצצות שלו ואת סטטוס החיים שלו מ 0-100 עם תמונות לאלמנט העיצוב.

```
56 <hr>
57 <div id="below-player-status">
58   <div id="players-div">
59     <table id="players">
60       <tr>
61         <th>Players</th>
62         <th>Health</th>
63         <th>Bombs</th>
64         <th>Drop Bomb</th>
65       </tr>
66     </table>
67   </div>
68   <div id="quest">
69     <p id="quest-text">Quest</p>
70     <p id="quest-itself"></p>
71     <div class="image-preview" id="image-preview">
72       <img src="" alt="quest image" id="quest-img">
73     </div>
74     
75     <div id="clue-div">
76       <p id="clue-text">Clue</p>
77       <p id="clue-itself"></p>
78     </div>
79     <form id="answer-form">
80       <label class="field-label">
81         <p id="answer-text">Answer</p>
82         <p id="flag-format">Flags Fromat: lgcw{FLAG}</p>
83         <input type="text" class="input-box" id="answer" name="flag" required>
84       </label>
85       <button type="button" id="submit-flag" onclick="checkAnswer()">Submit</button>
86       <button type="button" id="clue" onclick="getClue()">Clue</button>
87     </form>
88   </div>
89 </div>
90 </div>
91 </body>
92 </html>
```

ואת החלק שמתחת לנתונים של השחקן כמו חיים, שמות, פצצות והפלת פצצות על שחקנים אחרים בתוך טבלה כפי שניתן לראות. והקונטיינר של השאלה שם יש את השאלה עצמה, רמז שיכול להופיע אם השחקן בחר להשתמש בפצצה שלו ומקום לענות על השאלה.

:Reset_password.html

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <link rel="preconnect" href="https://fonts.gstatic.com">
5      <link href="https://fonts.googleapis.com/css2?family=Nunito&display=swap" rel="stylesheet">
6      <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
7      <link rel="stylesheet" type="text/css" href="css/reset_password.css"/>
8      <meta name="viewport" content="width=device-width, initial-scale=1.0">
9      <title>RESET PASSWORD</title>
10   </head>
11   <body>
12     <header class="main-header">
13       <div class="logo">
14         <a href="/">LGCW</a>
15       </div>
16
17       <input type="checkbox" class="menu-btn" id="menu-btn">
18       <label for="menu-btn" class="menu-icon">
19         <span class="menu-icon-line"></span>
20       </label>
21
22       <ul class="nav-links">
23         <li class="nav-link">
24           <a href="/login">Login</a>
25         </li>
26         <li class="nav-link">
27           <a href="/sign_up">Sign Up</a>
28         </li>
29       </ul>
30     </header>

```

ה head וה header של index.html הוא אותו דבר חוץ מהכותרת של העמוד וקובץ ה css שמיובא.

```

32   <div class="reset-password-form">
33     <img src="">
34     <h1>Reset Password</h1>
35     <form action="reset_password" method="POST">
36       <label class="field-label">
37         <input type="text" class="input-box" name="username_or_email" id="username_or_email" required>
38         <div class="label-text">Username or Email</div>
39       </label>
40       <br>
41       <button type="submit" class="reset-password-button">Reset Password</button>
42       <p id="error-message" style="display:none"></p>
43       <hr>
44       <p class="or">OR</p>
45       <p class="login-account">If you remember your password!</p>
46       <a href="/login"><button type="button">Login</button></a>
47     </form>
48   </div>
49   <script type="text/javascript" src="js/index_logic.js"></script>
50 </body>
51 </html>

```

ויש את הטופס לאיפוס הסיסמה בעזרת שימוש בשם המשתמש או במייל. לאחר לחיצה על איפוס סיסמה תופעל POST לכתובת reset_password עם מה שהמשתמש כתב. כפתור לאופציה של התחברות אשר מוביל לדף ההתחברות וקובץ ה javascript של Index.html.

:Reset_password_itself.html

```

33 <div class="reset-password-form">
34 <img src="">
35 <h1>Reset Password</h1>
36 <form method="POST" id="reset-password-form">
37 <label class="field-label">
38 <input type="password" class="input-box" name="password" id="password" required>
39 <div class="label-text">password</div>
40 </label>
41 <label class="field-label">
42 <input type="password" class="input-box" name="verifyPassword" id="verifyPassword" required>
43 <div class="label-text">Verify Password</div>
44 </label>
45 <br>
46 <button type="submit" class="reset-password-button">Reset Password</button>
47 <p id="error-message" style="display:none"></p>
48 <hr>
49 <p class="or">OR</p>
50 <p class="login-account">If you remember your password!</p>
51 <a href="login"><button type="button">Login</button></a>
52 </form>
53 </div>
54 <script type="text/javascript" src="/js/index_logic.js"></script>
55 </body>
56 </html>

```

ה head וה header של index.html הוא אותו דבר חוץ מהכותרת של העמוד, קובץ css וקובץ javascript שמיובא.

בנוסף יש את טופס איפוס הסיסמה שם המשתמש מקיש את סיסמתו החדשה פעמיים (מוסבר למה פעמיים מקודם בהסבר על טופס ההרשמה). כפתור להפעלת פעולת ה POST לכתובת שמוגדרת לעמוד הנוכחי בקובץ javascript המיובא ושמו reset_password_init.js. כפתור שמוביל לדף ההתחברות.

:Add_quest.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Add Quest</title>
5     <link rel="preconnect" href="https://fonts.gstatic.com">
6     <link href="https://fonts.googleapis.com/css2?family=Nunito&display=swap" rel="stylesheet">
7     <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
8     <link rel="stylesheet" type="text/css" href="css/add_quest.css"/>
9     <meta name="viewport" content="width=device-width, initial-scale=1.0">
10    <script defer src="js/add_quest.js"></script>
11  </head>
12  <body>
13    <div class="wrapper">
14      <div class="quest_type_wrap">
15        <h2>Please Select The Quest Type</h2>
16        <div class="quest_type_container">
17          <label class="radio_container_quest_type">
18            <input type="radio" name="quest_type" class="quest_radio" id="regular_quest">
19            <div class="radio_inner">
20              </img>
21              <div class="type_quest_icon_wrapper">
22                
23                </img>
24              </div>
25              <div class="info">Regular Quest</div>
26            </div>
27          </label>
28        </div>
29        <p id="not_selected_message">You didn't select something!</p>
30        <button type="button" class="next" onclick="checkSelectedRadio()">Next!</button>
31      </div>
32    </div>
33  </body>
34 </html>
```

דף לבחירת סוג השאלה שהמנהל רוצה להוסיף (יש רק שאלות עם טקסט ותמונה – שאלות רגילות), למעשה זהו דף מעבר מהdashboard להוספת שאלה.

:Add regular quest.html

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>add regular quest</title>
5      <link rel="preconnect" href="https://fonts.gstatic.com">
6      <link href="https://fonts.googleapis.com/css2?family=Nunito&display=swap" rel="stylesheet">
7      <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
8      <link rel="stylesheet" type="text/css" href="css/add_regular_quest.css"/>
9      <meta name="viewport" content="width=device-width, initial-scale=1.0">
10   </head>
11   <body>
12     <div class="add-regular-quest-form">
13       <!-- <img src="" -->
14       <h1>Regular Quest</h1>
15       <p class="note">Optional:</p>
16       <hr>
17       <form class="form" id="upload-form">
18         <input type="file" name="input-file" id="input-file">
19         <div class="image-preview" id="image-preview">
20           <img src="" alt="Image Preview" class="image-preview-itself" id="image-preview-itself">
21           <span class="image-preview-text" id="image-preview-text">Image Preview</span>
22           <span class="bad-file-type" id="bad-file-type">Bad File Type - File is Not an Image!</span>
23         </div>
24         <input type="submit" class="button" id="submit-image-btn" value="upload">
25       </form>
26       <div class="progress-bar" id="progress-bar">
27         <div class="progress-bar-fill" id="progress-bar-fill">
28           <span class="progress-bar-text" id="progress-bar-text">0%</span>
29         </div>
30       </div>
31     </div>

```

שורות 17-30 מציגות את האפשרות להעלאת תמונה שתהיה עם השאלה.

```

32   <form id="add-quest-form" method="post">
33     <p id="required-text">Required:</p>
34     <hr>
35     <label class="field-label">
36       <input type="text" class="input-box" name="quest" required>
37       <div class="label-text">Quest</div>
38     </label>
39     <label class="field-label">
40       <input type="text" class="input-box" name="clue" required>
41       <div class="label-text">Clue</div>
42     </label>
43     <label class="field-label">
44       <input type="text" class="input-box" name="answer" required>
45       <div class="label-text">Answer</div>
46     </label>
47     <hr>
48     <button type="submit" class="button" id="add-quest-button">Add Quest!</button>
49   </form>
50 </div>
51 <script src="js/add_regular_quest.js"></script>
52 </body>
53 </html>

```

שאר הקובץ הוא טופס שצריך להכניס שאלה, רמז ותשובה.

רפלקציה אישית:

למדתי המון דברים מהפרויקט הזה כמו איך להתנהל עם עוגיות, מתודות שונות, הצורה של שמירת הסיסמה של המשתמש בתור hash עם salt (דבר שעוזר מאוד לאבטחת הסיסמה כיוון שאם מאגר הנתונים שלי נחשף גם אם יגלו סיסמה של

אדם אחד זה ישפיע רק עליו, כל סיסמה מוצגת באופן שונה גם אותה סיסמה בדיוק) ואיך לעבוד עם פונקציות אסינכרוניות.

למדתי איך לעצב את דפי הHTML עם קבצי css ולהוסיף את הלוגיקה של הדפים עם קבצי javascript.

אני מסכם את תהליך הלימוד ובניית הפרויקט שלי בתור משהו חוויתי ומספק עקב הידיעה שהצלחתי לבנות מערכת שלמה (אתר קטן) שכל הקבצים והפונקציות עובדות יחד כאשר בתמונה הכוללת נוצר הפרויקט.

אמנם היה קושי מבחינת הזמן בגלל שהיו עוד עבודות, פרויקטים ומבחנים בבית הספר אך החוויה הכללית של סיום הפרויקט היא טובה.

כמובן שאם היה עוד זמן הייתי משפר את המראה של הדפים, מכניס מערכת של רמות ורמות מנהלים, מכניס עוד סוגי חידות שונות, מוסיף אפקטים של קול למשחק.

למדתי מעבודה על הפרויקט שחשוב לנצל כל טיפת זמן על מנת לסיים משהו שאתה מתחיל ולהתמיד ללא הפסקה עד הסוף.

אם הייתי צריך לשנות את צורת העבודה שלי, לא הייתי עושה זאת כיוון שכל הטעויות בדרך והלמידה העצמאית שלי שיפרו אותי בתחומים שהפרויקט שלי נגע בהם.

ביבליוגרפיה:

[/https://socket.io](https://socket.io)

[/https://www.npmjs.com](https://www.npmjs.com)

[/https://www.geeksforgeeks.org](https://www.geeksforgeeks.org)

[/https://stackoverflow.com](https://stackoverflow.com)

[/https://www.w3schools.com](https://www.w3schools.com)

[/https://www.youtube.com](https://www.youtube.com)

[/https://medium.com](https://medium.com)

אלו הם האתרים העיקריים שחקרתי ולמדתי מהם נושאים ושפות תכנות שהיו בתוך הפרויקט.