

תיק למתכנת – Corelia



מבנה ע"י: .turtles group

גירסה: 1.3

תאריך שחרור הגרסה: 24/06/2025

חברי הצוות:

עינב בן שושן – 315064881

יובל לרפלד – 318186897

מתן עמרן – 316326313

עמיר איזולאי – 206695488

חנן צפיר – 319005104

◊ **קבצים עיקריים:**

- HW3_Turtles_Corelia – מחברת הפרויקט

◊ **ספריות עיקריות:**

- **FireBase** - חיבור למסד נתונים בזמן אמת.
- **Ipywidgets** - יצרת ממשק משתמש אינטראקטיבי במחברת.
- **Sendgrid** - לשיחת מיילים (כולל איפוס סיסמה).
- **google.generativeai** - שילוב עם Gemini לצ'אט בטוט.
- **matplotlib, pandas, numpy** - לניטוח והציגת נתונים.

◊ **הרשאות משתמשים:**

- **User**
 - איפוס סיסמא.
 - אפשרות לשחק בטאב המשחקיות צפיה בניקוד.
- **Admin**
 - יצרת משתמש חדש למערכת.
 - צפיה בכל נתונים המשתמשים.
 - צפיה בכל המילימ שנותרים בDatabase.

• **Admin יורש הרשות User**

פרטי דיפולט למערכת לתחזקה ובדיקות:

- **סיסמת אדמין:**
 - Username: admin
 - Password: admin1
- **סיסמת יוזר רגיל:**
 - Username: user1
 - Password: user1

❖ פונקציות מרכזיות:

שם הפונקציה	תיאור הפונקציה
add_user()	הוספה משתמש חדש ל-Firebase.
add_index_to_firebase()	שמירת אינדקס מילים/נתונים ב-Firebase.
create_temp_humidity_chart()	יצירת גרפ טמפרטורה ולחות לאורך זמן.
create_daylight_hours_chart()	יצירת גרפ שעות אוור מול פעילות המשתמש.
calculateDailyScore()	חישוב ניקוד יומי למשתמש.
DisplayDailyRanks()	ցגת דירוג יומי לכל המשתמשים.
complete_task()	סימון משימה כהושלמה ועדכון המערכת.
monitor_loop()	ניתור משתמשים מחוברים בזמן אמת.
apply_stemming()	עיבוד טקסט לקיצוץ מילים לשורש.(Stemming)
create_index_inspector()	יצירת ממשק לצפייה באינדקס המילים.
password_reset_email()	שליחת מייל איפוס סיסמה למשתמש.
update_tabs()	עדכון ממשק הטעבים בהתאם להרשאות המשתמש
fetch_users_from_firebase()	שלילת רשימת משתמשים מה DB.
handle_login()	ניהול תהליךהתחברות משתמש (תיק הסתכלות על הרשות).
handle_logout()	ניהול תהליךהתנתקות משתמש ו הסתרת כל הטעבים.
on_click_search()	הפעלת חיפוש לפי טקסט שהוזן ע"י המשתמש.
saveUserSearchQuery()	שמירת שאלחת החיפוש של המשתמש ב-FireBase (למעקב אחריו חיפושים אחרים).
on_click_lucky()	חיפוש: "I'm Feeling Lucky": פיתוח התוצאה המדורגת הראשונה.
prioritize_pages()	דירוג העמודים בחיפוש לפי רמת רלוונטיות.

◊ **תבניות מיוחדות:**

• ממשק משתמש אינטראקטיבי עם widgets

- שימוש ב-()Tab.widgets לבניית טאבים נוחים למעבר בין הדפים השונים שיש ל המערכת להציג.
- שימוש ב-()HTML.widgets ליצור טבלאות ודוחות.
- פונקציות אשר מסתירות את כל הטאבים במקורה שלא מחוברים, מציגות את הטאבים במקורה של חיבור וויאג על פי משתמש שמחובר.

• טבלת ניקוד (LeaderBoard)

- תבנית להציג דירוג יומי / חודשי:
 - DisplayDailyRanks()
 - DisplayMonthlyRanks()

• גרפים סטטיסטיים עם Matplotlib

- גרף טמפרטורה ולחות לאורך זמן - ()create_temp_humidity_chart()
- גרף שעות אור - ()create_daylight_hours_chart()

• כפתור "Feeling Lucky"

- כפתור ייחודי שМОBILE אוטומטית לתוכה עם הדירוג הגבוה ביותר - ()on_click_lucky()

• דירוג עמודים בתוצאות חיפוש (Prioritization)

- מנגנון פנימי לקביעת סדר התוצאות בחיפוש - ()prioritize_pages()

• שליחת מייל רספוניבי לאייפוס סיסמה

- תבנית מייל מעוצבת שנשלחת אוטומטית בעת אייפוס סיסמה משתמשת בפונקציה שמייצרת סיסמה רנדומלית (כਮון לאחר שבודקת שפטוי המשתמש נכונים ואומתו) - ()password_reset_email()

• Real-time User Monitoring

- לולאת רקע (thread) לניטור משתמשים מחוברים ולעדכו טבלת האדמין - ()_monitor_loop()

• מנוע חיפוש חכם עם Stemming

- עיבוד מילים לשורש (stemming) לשיפור תוצאות החיפוש –
 - apply_stemming()
 - get_search_results()

❖ קטעי קוד מעניינים:

- יצירת גרף-

```
def create_temp_humidity_chart(data):  
    fig, ax1 = plt.subplots(figsize=(6, 3))  
    ax1.set_xlabel('Day')  
    ax1.set_ylabel('Temperature (°C)', color='salmon')  
    plt.subplots_adjust(top=0.85)  
    ax1.set_title('Average Daily Temperature Last 6 Days (Indoor Sensor)', pad=20)  
  
    line1 = ax1.plot(data['days'], data['avg_temps'], color='salmon', marker='o',  
label='Temperature')  
    ax1.tick_params(axis='y', labelcolor='salmon')  
  
    ax2 = ax1.twinx()  
    ax2.set_ylabel('Humidity (%)', color='turquoise')  
    line2 = ax2.plot(data['days'], data['avg_humidity'], color='turquoise', marker='s',  
label='Humidity')  
    ax2.tick_params(axis='y', labelcolor='turquoise')  
  
    lines = line1 + line2  
    labels = [l.get_label() for l in lines]  
    ax1.legend(lines, labels, bbox_to_anchor=(0.02, 0.98), loc='upper left')  
  
    fig.tight_layout()  
    return fig_to_img(fig)
```

• שליחת סיסמה חדשה במייל

```
def send_password_reset_email(to_email, new_password):
    global SENDGRID_API_KEY, forgotPage_error_message
    try:
        sg = sendgrid.SendGridAPIClient(api_key=SENDGRID_API_KEY)
        message = Mail(
            from_email = "einavbs1@gmail.com",
            to_emails=to_email,
            subject='Reset Your Password',
            plain_text_content=f'''
                Hello,

                We received a request to reset your password.

                This is your new password:
                {new_password}

                Best regards,
                Corelia Turtles Group
            ''')
        response = sg.send(message)
        forgotPage_error_message.value = "An email has been sent to your email address with the new password."+"\n Please check your inbox and your SPAM folder."
    except Exception as e:
        forgotPage_error_message.value = "Failed to send email. Please try again later."
```

```
def handle_login_click(b):
    global currKeyUserConnected, FBconn, lastUpdateUsersTable, currUserisAdmin,
currConnectedUserName
    if not username_field.value or not password_field.value:
        login_error_message.value = "Incorrect username or password. Please try again."
        return

    entered_username = username_field.value
    entered_password = password_field.value

    usersCreds = get_user_by_username(entered_username)
    logged = False
    for key, user in usersCreds.items():
        hashed_pw = user["password"].encode()
        isAdmin = user["isAdmin"]
        #if password correct we enter the system so update last seen and showinf the
        dashboard
        if bcrypt.checkpw(entered_password.encode(), hashed_pw):
            FBconn.patch(f"{LOGIN_PATH}/{key}", {"lastLogin":
            datetime.now(TZ_IL).isoformat()})
            FBconn.patch("/Login", {"lastupdate": datetime.now(TZ_IL).isoformat()})
            lastUpdateUsersTable = FBconn.get("/Login/lastupdate", None)
            login_error_message.value = ""
            currKeyUserConnected = key
            currUserisAdmin = isAdmin
            currConnectedUserName = user["name"]
            updateTab2ContentWithUserName()
            if isAdmin:
                get_users_info_from_db()
                start_monitoring()
                show_admin_dashboard()
            else:
                show_user_dashboard()
            resetTabs()
            logged = True
            break

    if logged == False:
        login_error_message.value = "Incorrect username or password. Please try again."
```

• הציגת הטאים הרלוונטיים:

```
def resetTabs():
    global tabs, currUserisAdmin
    currPage = tabs.selected_index if hasattr(tabs, 'selected_index') else 0
    tabs.children=[tab0_content,tab1_content,tab2_content,tab3_content, tab5_content,
    tab6_content]
    if currUserisAdmin == None :
        tabs.set_title(0, 'Login Page')
    elif currUserisAdmin:
        tabs.set_title(0, 'Admin Dashboard')
    else:
        tabs.set_title(0, 'User Dashboard')
    tabs.set_title(1, 'Graphs')
    tabs.set_title(2, 'Gamification')
    tabs.set_title(3, 'Search')
    tabs.set_title(4, 'Informations Transportation')
    tabs.set_title(5, 'Chatbot Assistant')
    tabs.selected_index = currPage
```

• **ניתור משתמשים בזמן אמיתי לדף האדמין**

```
def _monitor_loop():
    global monitor_active, lastUpdateUsersTable, userInfoForAdminTable

    while monitor_active:
        monitor_active = True
        try:
            print("Monitoring for changes in user data...")
            # Get current lastupdate value from Firebase
            new_update = FBconn.get("/Login/lastupdate", None)

            if new_update and new_update != lastUpdateUsersTable:
                lastUpdateUsersTable = new_update

                # Re-fetch users from DB
                get_users_info_from_db()
                update_admin_table_only() # Refresh the admin dashboard with new data
                print("Admin dashboard updated with new user data.")
        except Exception as e:
            import traceback
            print("Error while monitoring:", e)
            traceback.print_exc()

        time.sleep(5)
```