**מבוא למחשוב ענן – תרגיל בית 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **אלעד קראוז** | **לי פוטשניק** | **אורי זיו** | **דניאל ברקוביץ** | **אהרון ינוטייב** |
| 207281874 | 318158565 | 206131849 | 324304625 | 203899786 |

**קישור ל-Git של הפרויקט שלנו:** [**כאן**](https://github.com/Eladkrauz/cloud_computing_course)  
**קישור לקוד שלנו ב-Google Colab:** [**כאן**](https://colab.research.google.com/drive/1lGk9SHlHmSzi_eA6zQD8erYf3Eml1L6R?usp=sharing)

**1. בעלי תפקידים:**

|  |  |  |  |
| --- | --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** | **בדיקות קבלה** |
| לי **מהנדסת המערכת** | צ'אט בוט, התייחסות ל-Code Review | הכל | - האם הצ'אט בוט אינטראקטיבי  - האם הצ'אט בוט עונה כנדרש על שאלות שונות  - האם התשובות של הצ'אט בוט נכונות לפי מסד הנתונים  - האם הצ'אט בוט נגיש ונוח לשימוש |
| אורי | צ'אט בוט, התייחסות ל-Code Review | הכל |
| אלעד | סרטון, עיצוב המערכת ומימוש פיצ'רים לא גמורים, התייחסות ל-Code Review | הכל |
| דניאל | צ'אט בוט, תיק למתכנת, התייחסות ל-Code Review | הכל |
| אהרון | תיק למשתמש, התייחסות ל-Code Review, תכנון רעיון המערכת | הכל |

**2. מימוש ChatBot:** מימשנו בפרויקט באמצעות הספרייה **nltk.chat.util**.

**3. התייחסות לאלמנטים מנספח א':** הצגנו ויצרנו גרפים בהתאם לדרישות בנספח זה. ניתן לצפות בגרפים אלה ב-Dashboard של הפרויקט שלנו.

**4. התייחסות ל- Code Review:**

המשוב שקיבלנו ב-Code Review: בקישור [כאן](https://docs.google.com/document/d/12m60FKgf96Q8yOgMywmpJ5AuQmmRNyNi/edit). **נרשום כאן את התגובות שלנו על ההערות:**

**בדיקות קבלה:**  
בדיקות הקבלה שהם ביצעו כוללות אלמנטים שלא בהכרח היו חלק מהדרישות של הפרויקט. על כן, לא ביצענו אותם בפרויקט שלנו. אולם, מאחר והרעיונות הללו נראים רלוונטיים וענייניים, שקלנו להוסיף אותם כחלק מהפרויקט שלנו אך במסגרת הזמן שהוקצב לנו לא היה אפשרי להוסיף אלמנטים מעבר למה שהיה נדרש.

מדובר כאן על חשיבה עיצובית, ולכל פרויקט כמובן שיהיו רעיונות/אלמנטים שונים. בפרט דברים שמתאימים לפרויקט אחד לא בהכרח יהיו מתאימים לפרויקט אחר.

**קריטריונים:**  
יעילות: הוצע שניתן לשפר את יעילות הקוד על ידי שימוש במטמון, שאילתות יעילות יותר ועיבוד מקביל. זה מצביע על כך שאולי יש מקום לאופטימיזציה באופן שבו הקוד מטפל בעיבוד ואחזור נתונים. **אכן ביצענו בדיקה מקיפה לגבי איכות השאילתות שאנו שולחים למסד הנתונים.**

טיפול בשגיאות: הם ציינו שאין הודעה על שגיאות משתמש או שימוש במנגנון לטיפול בשגיאות. זה יכול להיות בעייתי מבחינת מתן חווית משתמש חלקה והבטחת שהמערכת מתנהגת באופן צפוי במקרה של קלט או שגיאות בלתי צפויות. **ביצענו בדיקה לכך שבמקרה של שגיאה אכן יש חיווי על המסך.** הוספנו עבור כל פעולה של קריאה ממסד הנתונים – חיווי על המסך עבור המשתמש במקרה של שגיאה. זה בא לידי ביטוי במצב בו יש צורך להציג גרף תוך לקיחת נתונים מהמסד והפעולה נתקלת בשגיאה. אנו מציגים הודעת פופ אפ למשתמש על כך שלא ניתן להציג את הגרף כעת.

**התייחסות להערות האישיות:**

|  |  |  |
| --- | --- | --- |
| **ת"ז** | **הערה שלהם** | **תגובה שלנו** |
| **314813825** | **סינון נתונים:** אין פילטרים לסינון נתונים אצלם, כך שאין אפשרות לבחור מספר סטודנטים מצומצם  **חלוקת הקוד לחלקים:** עדיף לחלק את הקוד ל-sections, הקוד יהיה מובן יותר ומחולק היטב. | **סינון נתונים:** כאמור, פיצ'ר זה אינו כלול כחלק מדרישות הפרויקט.  **חלוקה:** אנו חילקנו את הקוד במהלך העבודה ובסיומה איחדנו הכל למען שתהיה תוכנית שלמה. זאת התוצאה שהם ראו. |
| **322662065** | **דרישות יבוא ויצוא:** חסר במערכת עבודה ישירה של המשתמש עם המסד. **דרישות התראה:** קיימים הסברים לכל גרף מה המשמעות שלהם, ואפשר לחשוב על זה כהתראה למשתמש | **יבוא ויצוא:** לא הייתה דרישה לעבודה ישירה של המנהל עם מסד הנתונים. היה צורך לקרוא את קובץ ה-JSON אל תוך מסד הנתונים ולעבוד עם נתונים אלו בלבד. |
| **212970156** | **בדיקת פונקציונליות:** כל הפונקציות העיקריות של התוכנה פועלות כמצופה ומשיגות את היעדים והמטרות של התרגיל. **בדיקת אינטגרציה:** השילוב בין הפונקציות עובדות יחד כראוי. | - |
| **206468795** | **האם יש גישה למשתמשים בעלי מוגבלויות?** לא **האם המערכת נוחה לשימוש?** כן. מובנת, לא מסובכת. | **גישה למשתמשים בעלי מוגבלויות** הוא אכן פיצ'ר איכותי ורלוונטי אך במסגרת הזמן, כיוון שלא היה חלק מהדרישות, לא נכלל בפרויקט שלנו. |
| **209001981** | **האם המערכת ידידותית למשתמש?** המערכת מאוד קלה להבנה ולשימוש, והניווט בה אינטואיטיבי כך שאפשר להתחיל לעבוד במהירות ובקלות. **האם המידע מוצג בצורה נכונה ואמינה?** המידע מוצג בצורה ברורה ונכונה בהתאם לקובץ ה-JSON, והפונקציות מנתחות אותו . | - |

**תיק למתכנת**

**קבצים מרכזיים:**

* **קובץ JSON** (עם נתוני הפעולות שבוצעו בפרויקט)
* **CloudProject-Tiger.ipynb**

**פונקציות מרכזיות:**

**fetch\_data\_from\_firebase()**הסבר: פונקציה זו יוצרת חיבור עם מסד הנתונים של Firebase ומחזירה את הנתונים המאוחסנים תחת נתיב ספציפי. לאחר מכן, היא מחזירה את הנתונים לעיבוד נוסף והצגתם בגרפים.

def fetch\_data\_from\_firebase():

    """

    This function fetches data from the specified Firebase database.

    It connects to the Firebase URL and retrieves data stored under the '/HW2-Tiger/' path.

    The retrieved data is stored in a list and returned.

    If no data is found, an empty list is returned.

    """

    result = FBconn.get('/HW2-Tiger/', None)

    if result:

        data = [value for value in result.values()]

        return data, result

    else:

        return [], []

**send\_email\_with\_attachment()**הסבר: פונקציה זו שולחת מייל עם קובץ מצורף, ומאפשרת למשתמשים לשתף נתונים חזותיים או דוחות שנוצרו על ידי האפליקציה ישירות מהממשק.

def send\_email\_with\_attachment(subject, body, attachment\_path, recipient\_email):

    """

    This function sends an email with an attachment.

    It uses the smtplib library to send an email from a Gmail account.

    The email includes a subject, body, and an attachment.

    """

    msg = MIMEMultipart()

    msg['From'] = EMAIL\_ADDRESS

    msg['To'] = recipient\_email

    msg['Subject'] = subject

    # Attach the email body

    msg.attach(MIMEText(body, 'html'))

    # Attach the file

    attachment = MIMEBase('application', 'octet-stream')

    with open(attachment\_path, 'rb') as file:

        attachment.set\_payload(file.read())

    encoders.encode\_base64(attachment)

    attachment.add\_header('Content-Disposition', f'attachment; filename={os.path.basename(attachment\_path)}')

    msg.attach(attachment)

    # Connect to the SMTP server and send the email

    with smtplib.SMTP(SMTP\_SERVER, SMTP\_PORT) as server:

        server.starttls()

        server.login(EMAIL\_ADDRESS, EMAIL\_PASSWORD)

        server.send\_message(msg)

**create\_main\_dashboard()**הסבר: פונקציה זו מייצרת את ה-Dashboard הראשי של האפליקציה, ומספקת למשתמש כפתורים המובילים לגרפים וניתוחים של נתונים הקשורים לפעילויות ניהול הפרויקטים שלהם.

def create\_main\_dashboard():

    """

    This function creates the main dashboard with buttons for different actions.

    The buttons are arranged in a grid layout.

    """

    # Create buttons for the dashboard

    dashboard\_buttons = [

        widgets.Button(description='Actions by User', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Actions by Document', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Number of Actions Over Time', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Actions by Tab', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Personal Assistant', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Progress Patterns', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Collaboration Signs', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Working Hours', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='User Contributions', layout=widgets.Layout(width='200px', height='75px')),

        widgets.Button(description='Student Activities', layout=widgets.Layout(width='200px', height='75px'))

    ]

    # Arrange buttons in a grid

    dashboard\_grid = widgets.GridBox(

        children=dashboard\_buttons,

        layout=widgets.Layout(

            width='100%',

            grid\_template\_columns='repeat(5, 250px)',

            justify\_content='center',

            align\_items='center',

            grid\_gap='10px',

            padding='20px'

        )

    )

    # Title HTML widget

    title = widgets.HTML(value=f"""

        <div style='background: linear-gradient(to right, #ffffff, #6EC373, #ffffff); padding: 20px; border-radius: 10px; color: black; text-align: center;'>

            <h1 style='color: black'>Welcome to OnShape, Amir Cohen</h1>

            <h2 style='color: black'>{(datetime.now(pytz.timezone('Asia/Jerusalem'))).strftime('%Y-%m-%d %H:%M:%S')}</h2>

            <h3 style='color: black'>On the dashboard, you can explore different aspects of your project management data. Check out actions by user, see which documents are accessed the most, track the number of actions over time, and dive into actions by tab. It's all about giving you a clear view of what's happening in your project!</h3>

        </div>

    """)

    # Combine title and grid into a VBox with a white background

    dashboard\_content = widgets.VBox([title, dashboard\_grid], layout=widgets.Layout(

        padding='20px',

        width='100%',

        background\_color='white'

    ))

    # Wrap the content in a Box to set the background color and ensure full width

    dashboard\_box = widgets.Box([dashboard\_content], layout=widgets.Layout(

        width='100%',

        height='auto',

        background\_color='white',

        padding='20px'

    ))

    # Function to navigate to the dashboard (placeholder, can be customized)

    def go\_to\_dashboard(button):

        display\_main\_dashboard()

    # Assign the navigation function to all buttons

    for button in dashboard\_buttons:

        button.on\_click(go\_to\_dashboard)

    return dashboard\_box

**plot\_actions\_by\_user()**הסבר: פונקציה זו יוצרת תרשים עוגה המייצג חזותית את התפלגות הפעולות שבוצעו על ידי משתמשים שונים, ועוזרת לזהות אילו משתמשים הם הפעילים ביותר.

def plot\_actions\_by\_user(save\_path='actions\_by\_user.png'):

    """

    This function plots the number of actions by user and saves the plot as an image.

    It fetches data from Firebase, creates a pie chart showing the number of actions by each user,

    and saves the plot as a PNG file.

    """

    data, \_ = fetch\_data\_from\_firebase()

    if data:

        df = pd.DataFrame(data)

        with plot\_output:

            plot\_output.clear\_output()

            fig, ax = plt.subplots(figsize=(6, 4))  # Adjust the figsize for better layout

            colors = sns.color\_palette("Greens", as\_cmap=True)(np.linspace(0.5, 1, df['User'].nunique()))  # Apply the color palette

            user\_counts = df['User'].value\_counts()

            user\_counts.plot(kind='pie', ax=ax, colors=colors, autopct='%1.1f%%', startangle=90)

            ax.set\_title('Number of Actions by User', fontsize=16)

            ax.set\_ylabel('')  # Remove y-label for a cleaner look

            ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

            plt.show()

            fig.savefig(save\_path, format='png')  # Save the plot as an image

    else:

        show\_popup("We currently cannot display this graph at the moment.\nPlease try again later.", title="Error")

**display\_virtual\_assistant()**הסבר: פונקציה זו מגדירה ומציגה ממשק צ'אט בוט המאפשר למשתמשים ליצור אינטראקציה עם העוזר הווירטואלי, לבצע שאילתות בנתוני הפרויקט ולקבל תגובות על סמך דפוסים וסטטיסטיקות מוגדרות מראש.

def display\_virtual\_assistant():

    """

    This function displays the page for the personal virtual assistant.

    """

    clear\_output()

    virtual\_assistant\_description = "This is your project management assistant. Ask me anything about your project's data."

    page = create\_page('Personal Virtual Assistant', virtual\_assistant\_description, additional\_widgets=None)

    display(page)

    # Fetch data from Firebase

    \_, data = fetch\_data\_from\_firebase()

    # Process the data

    users = Counter([item.get('User') for item in data.values()])

    documents = Counter([item.get('Document') for item in data.values() if item.get('Document')])

    tabs = Counter([item.get('Tab') for item in data.values() if item.get('Tab')])

    descriptions = Counter([item.get('Description') for item in data.values() if item.get('Description')])

    # Calculate some statistics

    total\_actions = len(data)

    unique\_users = len(users)

    unique\_documents = len(documents)

    unique\_tabs = len(tabs)

    # Set patterns for the chatbot

    patterns = set\_patterns(total\_actions, users, documents, tabs, descriptions, unique\_users, unique\_documents)

    chatbot = Chat(patterns, reflections)

    # Create interactive elements for the chat

    user\_input\_widget = widgets.Text(placeholder='Type your question here...')

    send\_button = widgets.Button(description='Send')

    chat\_output = widgets.Output()

    # Display the initial greeting

    with chat\_output:

        print("Hello! I'm the Project Management Assistant. How can I help you today?")

    def on\_send\_button\_clicked(b):

        user\_input = user\_input\_widget.value

        user\_input\_widget.value = ''

        with chat\_output:

            if user\_input.lower() in ['exit', 'bye', 'goodbye']:

                print("ChatBot: Thank you for using the Project Management Assistant. Goodbye!")

                return

            print(f"You: {user\_input}")

            response = chatbot.respond(user\_input)

            if response:

                print("ChatBot:", response)

            else:

                specific\_response = handle\_specific\_user\_query(user\_input, users, data)

                print("ChatBot:", specific\_response if specific\_response is not None else "This type of question is not supported.")

    send\_button.on\_click(on\_send\_button\_clicked)

    # Layout for the chat interface

    chat\_interface = widgets.VBox([user\_input\_widget, send\_button, chat\_output], layout=widgets.Layout(align\_items='center'))

    display(chat\_interface)

**קטעי קוד ותבניות עיצוב  
טיפול בשגיאות בקריאות מול מסד הנתונים**הסבר: דפוס זה מבטיח שאם אחזור הנתונים של Firebase נכשל או חוזר ריק, המשתמש מקבל הודעה ברורה, ומונע שגיאות בלתי צפויות.

data, \_ = fetch\_data\_from\_firebase()

    if not data:

        show\_popup("We currently cannot display this graph at the moment.\nPlease try again later.", title="Error")

        return

**שימוש ב-Caching לטיפול יעיל בנתונים**הסבר: המרת נתונים שנשלפו ל-DataFrame ואחסון במטמון מאפשרים שימוש חוזר יעיל במספר פונקציות ללא אחזור נתונים מיותר, מה שמשפר את הביצועים.

if data:

    df = pd.DataFrame(data)

    user\_counts = df['User'].value\_counts()

**יצירת גרפים דינמיים עם פונקציונליות לשמירה**הסבר: הקוד יוצר באופן דינמי גרפים ומיד שומר אותם כתמונות, מה שמאפשר למשתמשים לייצא ולשתף נתונים חזותיים ישירות מהממשק.

def plot\_actions\_by\_user(save\_path='actions\_by\_user.png'):

    fig.savefig(save\_path, format='png')  # Save the plot as an image

**גרפים גמישים עם אינטראקטיביות למשתמש**הסבר: דפוס עיצוב זה מאפשר למשתמשים לקיים אינטראקציה עם הממשק כדי ליצור גרפים ספציפיים, עם הפונקציונליות הנוספת של שליחת גרפים אלו באמצעות דואר אלקטרוני, מה שמשפר את האינטראקטיביות והשימושיות של האפליקציה.

display\_plot\_with\_send\_email(plot\_actions\_by\_user, 'actions\_by\_user.png', 'Actions by User', actions\_by\_user\_description)

**עוזר וירטואלי מותאם אישית עם עיבוד שפה טבעית**הסבר: השילוב של צ'אט בוט מבוסס NLP מאפשר למשתמשים לבצע שאילתות בנתוני הפרויקט באמצעות שיחה, מה שהופך את המערכת ליותר ידידותית ואינטואיטיבית למשתמשים שאינם טכניים.

chatbot = Chat(patterns, reflections)

response = chatbot.respond(user\_input)

**עיצוב רספונסיבי עם ווידג'טים וגרידים**הסבר: השימוש ב-GridBox ובווידג'טים מבטיח שה-Dashboard מגיב, עם כפתורים ואלמנטים מיושרים על פני גדלי מסך שונים, ומשפרים את חווית המשתמש הכוללת.

dashboard\_grid = widgets.GridBox(children=dashboard\_buttons, layout=widgets.Layout(grid\_template\_columns='repeat(5, 250px)'))

**יצירת דפים מודולריים עבור תצוגות שונות**הסבר: הגישה המודולרית ליצירת עמודים מאפשרת הוספה או שינוי קלה של תוכן, מה שהופך את המערכת לגמישה וניתנת לתחזוקה.

def create\_page(title\_text, description, additional\_widgets=None):

    children = [back\_button, seperator, page\_title, seperator]

    if additional\_widgets:

        children.extend(additional\_widgets)

**תיק למשתמש**

**הסבר כללי:**המערכת מדמה אתר מבוסס אינטרנט להצגת פעולות משתמשים על מסמכים שוניםהמאוחסנים במסד נתונים שלFirebase .

המערכת מעבדת את הנתונים המגיעים ממסד הנתונים ובעקבות זה המשתמשים יכולים לצפות במגוון גרפים, כגון פעולות לפי משתמש, פעולות לפי מסמך, פעולות לאורך זמן, פעולות לפי Tab ועוד. על כל גרף המערכת מציגה הסבר למשתמש על הנתונים המוצגים. המערכת גם מאפשרת למשתמשים לשלוח דוחות במייל עם גרפים.

המערכת מאפשרת אינטראקציה בין המשתמש לchat bot.

**מסכים וניווט:  
נציין שבכל אחד מן הגרפים משתמש יכול לחזור בחזרה למצב הראשי ולשלוח את הגרף במייל, עם כותרת מסודרת למסמך הנשלח במייל.**

|  |  |
| --- | --- |
| **מסך ראשי** | המסך הראשי כולל כפתורים לגרפים שונים: "פעולות לפי משתמש", "פעולות לפי מסמך", "מספר פעולות לאורך זמן", "עוזר אישי", "דפוסי התקדמות", "שיתוף פעולה בין משתמשים", " שעות עבודה", "תרומת משתמשים", "סיווג המשתמשים" ו"פעולות לפי טאב". |
| **פעולות לפי משתמש** | לחיצה על כפתור זה תציג תרשים עוגה המציג את מספר הפעולות שבוצעו על ידי כל משתמש. |
| **פעולות לפי מסמך** | לחיצה על כפתור זה תציג תרשים עמודות המציג את מספר הפעולות שבוצעו על כל מסמך. |
| **מספר פעולות לאורך זמן** | לחיצה על כפתור זה תציג תרשים קו המציג את מספר הפעולות שבוצעו לאורך זמן. |
| **פעולות לפי טאב** | לחיצה על כפתור זה תציג תפריט לבחירת מסמך ואז תציג תרשים עמודות של מספר הפעולות שבוצעו לפי טאב עבור המסמך שנבחר. |
| **עוזר אישי** | מסך להצגת הצ'אט בוט של האפליקציה. |
| **דפוסי התקדמות** | מסך להצגת יצירת פיצ'רים וטאבים חדשים של המשתמשים לאורך הזמן. |
| **שיתוף פעולה בין משתמשים** | מסך להצגת תרומות ועבודה חופפים (על אותו טאב/מסמך) בידי מספר משתמשים. |
| **שעות עבודה** | מסך להצגת התפלגות שעות העבודה עבור כל סטודנט, בין עבודה ביום חול, עבודה בסוף שבוע ועבודה בשעות הלילה. |
| **תרומת משתמשים** | מסך להצגת תרומות של כל משתמש על פי הפעולות השכיחות ביותר בפרויקט. |
| **סיווג המשתמשים** | מסך להצגת פעולות המשתמשים לפי פעולות "יצירתיות", "ניהוליות", "צפייה" ו"אחרות". כמו כן, הצגת המשתמש בעל התרומה הגדולה ביותר, וכן הצגת המשתמש בעל כמות התרומות היצירתיות הגדולה ביותר, לצורך הצגת המשתמש בעל ה-Performance הטוב ביותר. |

**שגיאות אפשריות:**

|  |  |
| --- | --- |
| **שגיאת חיבור ל-Firebase** | אם המערכת לא מצליחה לקבל נתונים מה Firebase היא מחזירה רשימה ריקה ולא מציגה נתונים. מציגה הודעה בהתאם. |
| **שגיאת שליחת מייל** | אם המערכת נתקלת בשגיאה בזמן שליחת מייל, היא מציגה הודעת שגיאה ומבקשת מהמשתמש להזין כתובת מייל תקינה. |
| **נתונים לא תקינים** | אם הנתונים המתקבלים מה-Firebase מכילים שדות לא תקינים או חסרים, המערכת מנסה לטפל בשגיאות על ידי מחיקת שורות לא תקינות או החלפתן בערכים ברירת מחדל. |