מבוא למחשוב ענן – דו"ח פרויקט

פרטי המגישים (קבוצה – Ant)

* אושר בן זזון (212240238).
* עמית בניטה (315949594).
* לאוניד ויאושקוב (327312526).
* שניר יהודה (308450154).
* לב לייפר (321385064).
* אנדי לואיס ספנר (213298524).

קישורים

* תיקיית GitHub: <https://github.com/Andy-Lewis-Sapner/ITCC-Ant-SemB24.git>
* קובץ הפרויקט (Google Colab): <https://colab.research.google.com/drive/1SRQFFIRVBBW8oZ55h75jqjy09QxSaKTQ?usp=sharing>

מהות המוצר ומרכיביו

מהות המוצר היא לעזור למנהלי פרויקטים בתוכנת Onshape לבדוק נתונים וסטטיסטיקות לגבי צוות הפרויקט שהשתתף באחד מהפרויקטים שניהל.

באמצעות נתונים אלו, המנהל יוכל ללמוד כיצד לעזור לחברי צוות הפרויקט בעתיד, כך שיצליחו לפתח מוצר טוב יותר בצורה טובה יותר.

האפליקציה מכילה מספר מסכים:

**מסך login:**

המסך מכיל שני שדות טקסט, האחד עבור שם המשתמש והשני עבור הסיסמה.

* במידה ושם משתמש לא קיים או שהסיסמה לא תואמת לשם המשתמש הקיים, מקבלים הודעת שגיאה: "login failed".

מהות המסך היא רישום משתמש למערכת.

**מסך home:**

המסך מכיל 5 כפתורים: Parameters, Statistics, JSON Upload, Index Search, Chatbot אשר כל כפתור מעביר למסך שלו:

כפתור – Parameters: מעבר למסך Parameters

כפתור – Statistics: מעבר למסך Statistics

כפתור – JSON Upload: מעבר למסך JSON Upload

כפתור – Index Search: מעבר למסך Index Search

כפתור – Chatbot: מעבר למסך Chatbot

מהות המסך היא הצגת המסכים הקיימים באתר ומתן אפשרות לעבור אליהם.

**מסך Parameters:**

המסך מכיל 4 תפריטי בחירה: Type, Project, Student, Tab, שני תפריטי שנה לבחירת תאריך: Start Date, End Date וכפתור Next.

בלחיצה על הכפתור, האתר משתמש בפרמטרים ומציג גרף ולצידו טבלת נתונים.

הגרף מציג את הפעילות בפרויקט בהתאם לפילטרים שנבחרו. המסך מהווה חלק ממערכת ניתוח נתונים, המיועדת לייצר גרפים שונים בהתבסס על נתונים שנשמרו במאגר מידע.

רכיבי המסך:

* תפריט Type
* תפריט Project
* תפריט Student
* תפריט Tab
* לוח שנה Start Date: בחירת תאריך התחלה
* לוח שנה End Date: בחירת תאריך סוף
* כפתור – Next: מעבר לתצוגת גרף ולצידו טבלת נתונים

מהות המסך היא תצוגת גרף נוחה להבנה בעבור המשתמש לפי הפרמטרים שציין.

**מסך Statistics:**

המסך מממש את הדרישה שקיבלנו בתרגיל בית 3 לפי קובץ JSON מספר 3:

find the contribution of each of the users to the progress of the task

המסך מכיל 2 תפריטי בחירה: Statistic, User.

רכיבי המסך:

* תפריט Statistic
* תפריט User
* לוח שנה Start Date
* לוח שנה End Date
* כפתור – Load Selection
* כפתור – Export Data

גרף זה מציג את מספר הפעולות השליליות (מחיקות, סגירות, ביטולים) שבוצעו על ידי המשתמש/ים הנבחר/ים.

**מסך JSON Upload:**

המסך מכיל 2 שדות טקסט: Header, Text, ו-3 כפתורים: Upload JSON, Back, Next.

רכיבי המסך:

* שדה Header
* שדה Text
* כפתור – Upload JSON
* כפתור – Back
* כפתור – Next

מהות המסך היא העלאת קובץ JSON לDatabase על מנת לאפשר הצגה שלו במסכים Parameters ו- Statistics.

**מסך Search by Index** :  
המסך הזה נבנה במטרה לאפשר למשתמש לחפש מידע לפי אינדקס בתוך מאגר נתונים כלומר חיפוש מילה או חלק ממילה, על מנת לבדוק האם קיימות תוצאות ב10 המילים החשובות ביותר במילון של Onshape.   
הכוונה במילים החשובות הם אלו שמופיעות הכי הרבה פעמים במילון ובקישורים בהם המילון מכיל.  
01 המילים החשובות במילון:  
assembl, context, draw, keyboard, plan, select, shortcut, sketch, studio, type

**רכיבים עיקריים של המסך:**

1. **כותרת** -
   * הכותרת של המסך מציגה את הכיתוב "Search by Index".
2. **שדה קלט לחיפוש** -
   * שדה שבו המשתמש יכול להקליד את מונח החיפוש.
3. **כפתור חיפוש** search
   * כפתור שמפעיל את החיפוש כאשר המשתמש לוחץ עליו.
   * הכפתור כולל אייקון של זכוכית מגדלת (אייקון חיפוש), ומציג טיפ כלי (tooltip) המורה על הפונקציה שלו.
4. **אזור תוצאות החיפוש** Search results will appear here:
   * אזור שבו יוצגו תוצאות החיפוש לאחר ביצוע החיפוש.
   * התוצאות מוצגות בתוך מסגרת עם סגנון עיצובי שמגדיר את צבעי הטקסט ואת הגובה והגולל (scroll) של האזור במקרה של תוצאות רבות.

המסך הזה מיועד לשימוש באפליקציה שבה המשתמשים מחפשים ומקבלים תוצאות מתוך מאגר נתונים לפי אינדקס. הוא מספק ממשק חיפוש אינטראקטיבי ונוח, עם הצגה ברורה של התוצאות ועם עיצוב מותאם לצרכי המשתמש.  
תהליך החיפוש מתבצע כך :  
המשתמש מקליד ערך בשדה הקלט הפונקציה אוספת את מונח החיפוש משדה הקלט ומחפשת אותו בתוך רשימת נתונים אם נמצאות תוצאות שמתאימות למונח החיפוש, הן מוצגות באזור תוצאות החיפוש בפורמט מעוצב. כל תוצאה מציגה את שם האינדקס ואת מספר המופעים . אם לא נמצאות תוצאות, מוצגת הודעה מתאימה.

**מסך Chatbot**:  
  
מסך אינטראקטיבי לצ'אטבוט, שמטרתו לעזור למנהלי תוכנה ולמשתמשים אחרים לשלוף מידע חשוב מתוך מסד נתונים הקשור לפרויקטים, משתמשים ופעולות שבוצעו. הצ'אטבוט מסוגל לקבל שאילתות משתמש ולהחזיר מידע בהתאם למבנה ולתכנים הקיימים במאגר הנתונים. להלן הסבר מעמיק על המסך ומרכיביו:

**מהות המסך:**

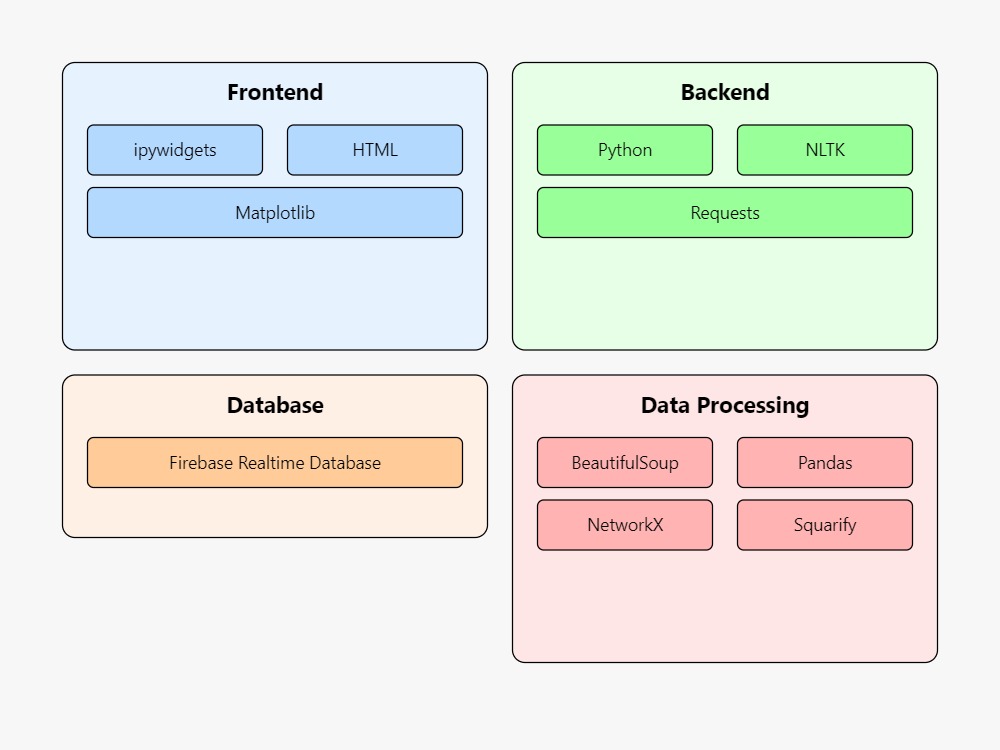
המסך מתפקד כחלון שיחה אינטראקטיבי שבו משתמשים יכולים להקליד שאילתות הקשורות לפרויקט מסוים ולקבל תשובות בצורה של הודעות טקסט מהצ'אטבוט. המטרה העיקרית היא לאפשר למשתמשים לגשת למידע אודות הפרויקט בצורה פשוטה וידידותית על ידי שימוש בשפה טבעית.  
  
המסך כולל את הרכיבים הבאים:

1. תיבת טקסט (input\_text):
   * תפקיד: תיבה זו מאפשרת למשתמש להקליד את השאלה או הפקודה שהוא רוצה לשלוח לצ'אטבוט.
2. כפתור שליחה (send\_button):
   * תפקיד: הכפתור משמש לשליחת הטקסט שהוקלד בתיבת הטקסט לצ'אטבוט. ברגע שלוחצים עליו, ההודעה נשלחת והצ'אטבוט מגיב בהתאם.
3. הסבר ראשוני (initial\_entry):
   * תפקיד: בהתחלה מוצג למשתמש הסבר על השאלות האפשריות שהוא יכול לשאול את הצ'אטבוט. זהו טקסט סטטי שמופיע בתחילת המסך כדי לעזור למשתמש להבין אילו שאלות ניתן לשאול.

בנוסף, יש גם תיבת גלילה שמציגה את ההיסטוריה של השיחה.

ארכיטקטורת המערכת

**תרשים ארכיטקטורת המערכת:**



**פירוט ארכיטקטורת ומאפיינים מרכזיים:**

**ארכיטקטורת מודולים:** המערכת בנויה ממודולים נפרדים (כניסה, בית, העלאת JSON, חיפוש אינדקס, פרמטרים, סטטיסטיקות, צ'אטבוט) שמאפשרים גמישות ותחזוקה קלה.

**:ipywidgets** ליצירת ממשק משתמש דינמי ותגובתי.

**Pandas ו-Matplotlib:** לניתוח נתונים מתקדם, עיבוד וויזואליזציה של נתונים.

**Firebase Realtime Database:** מבוסס ענן לאחסון ואחזור נתונים בזמן אמת.

**NLTK**: עיבוד שפה טבעית לצורך חיפוש מתקדם וצ'אטבוט.

**:NetworkX** ניתוח רשתות לניתוח יחסים ושיתופי פעולה בין משתמשים.

**Requests**: אינטגרציה עם מערכות חיצוניות לתקשורת עם שירותים חיצוניים (כמו OnShape API).

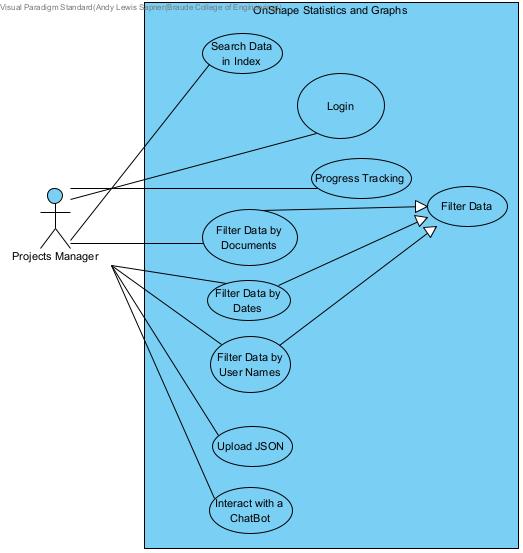
**מאפיינים נוספים:**

**יכולות ייצוא:** אפשרות לייצא נתונים לפורמטים חיצוניים לניתוח נוסף.

**מנגנון אימות משתמשים:** מערכת כניסה בסיסית לאבטחת הגישה למערכת.

**עיבוד וניתוח קבצי JSON:** יכולת לקלוט ולנתח קבצי JSON המכילים את נתוני פרויקט.

Use Case



דרישות לא פונקציונאליות

1. המערכת תשתמש ב FIRBASE לשמירת ה JSON וה INDEX בצורת JSON (Data retention )
2. ה INDEX לא יכיל מילים ללא משמעות STOPWORDS ( Quality )
3. המערכת תכיל CHATBOT שעובד עם הנתונים של ה JSON שמעלים ( Accessibility )
4. מסך הפרמטרים יפלטר את המידע לפי בחירת המשתמש ויציג בגרף ( Usability )
5. מסך הסטטיסטיקה יאפשר להוריד\לעלות את המידע המפולטר כקובץ אקסל ( Backup )
6. המערכת תתמוך בגדלים שונים של מסכים. (Adaptability).
7. רמת האפקטיביות נמדדת לפי כמות הפעולות שביצע משתמש בכל יום במהלך חיי הפרויקט, מרגע פתיחתו ועד סגירתו. (Documentation).
8. המערכת תכלול ממשק עזר כדי להקל על המשתמשים החדשים. (Usability).
9. זמן טעינת הנתונים יהיה עד 10 דקות. (Performance).

תיק תחזוקה

שמות הקבצים של הפרויקט:

* Instructions – contains the instructions about using the application.
* Installations – has the installation of needed libraries for the application.
* Imports – has the imports of all needed libraries.
* Utils:
  + DB Functionality – contains functions for DB interactions.
  + Global Variables – variables that are used for several sections.
  + Download Nltk – downloads nltk packages for the chatbot to work.
  + Index Creation – create an index of most important words in the onshape glossary.
  + Styles – contains variables with CSS styles for the application.
* All Screens – they contain the appearances of the screens and the logic behind them, for creating the content that is available on each screen.
  + Login Screen
  + Home Screen
  + Parameters Screen
  + Statistics Screen
  + Index Search Screen
  + JSON Upload Screen
  + ChatBot Screen
* Main Page – create the page on which the screen is shown.
* Main Program – the program the displays the application, creates an index if there is no index in the database, and saves the json file of the project in the database if there is no json file in the database.

אובייקטים מרכזיים:

ישנם שני אובייקטים מרכזיים, אשר רוב הקבצים משתמשים בלפחות אחד מהם. אובייקטים אלו הם משתנים, אשר נקראים page ו-FBConn. המשתנה page הוא משתנה מסוג ipywidgets.Vbox, ומטרתו להכיל את המסך שמוצג בעבור המשתמש (לפי בחירותיו ופעולותיו). המשתנה FBConn הוא משתנה מסוג FirebaseApp, ומטרתו להכיל את הקישוריות למסד הנתונים Firebase. השימוש בו הוא בעבור ביצוע פעולות הקשורות למסד הנתונים, אשר הן העלאת קובץ JSON וקבלת הקובץ ממסד הנתונים. בנוסף, שמירת וקבלת האינדקס של המילים החשובות ביותר במילון Onshape.

תיעוד קצר של כל פונקציה:

נציג את השמות של הפונקציות, לפי הסדר המופיע בקובץ הפרויקט, ולידם את התיעוד:

1. connect\_to\_DB(DB\_url): Connects to a Firebase database using the provided URL.

2. saveJsonToDB(jsonFile): Saves a JSON file to the Firebase database.

3. openJsonFromDB(): Opens and returns the latest JSON file from the Firebase database.

4. fetch\_page(url): Fetches the content of a webpage given its URL.

5. index\_words(soup): Indexes words from the HTML content, counting occurrences.

6. get\_stop\_words(): Returns a list of common stop words to be excluded from indexing.

7. remove\_stop\_words(index): Removes stop words from the indexed words.

8. apply\_stemming(index): Applies stemming to the indexed words to reduce them to their root forms.

9. find\_important\_10\_words(url): Finds and returns the 10 most important words from a webpage, by their occurrences.

10. create\_important\_10\_index(important10, initial\_index): Creates an index of the top 10 most important words.

11. post\_index\_to\_DB(): Posts the created index (with the 10 most important terms) to the Firebase database.

12. get\_index(): Retrieves the last posted index from the Firebase database.

13. create\_login\_screen(): Creates the login screen UI for the application.

14. login\_username(): Creates a text input widget for the username.

15. login\_password(): Creates a password input widget.

16. json\_upload\_clicked(b): Handles the event when the JSON upload button is clicked, switches the page to the json upload page.

17. index\_search\_clicked(b): Handles the event when the Index Search button is clicked, switches the page to the index search page.

18. parameters\_clicked(b): Handles the event when the Parameters button is clicked, switches the page to the parameters screen page.

19. statistics\_clicked(b): Handles the event when the Statistics button is clicked, switches the page to the statistics screen page.

20. create\_home\_screen(): Creates the home screen UI for the application with buttons.

21. create\_parameters\_screen(): Creates the parameters screen UI where users can filter data.

22. dropdown\_graphtype(): Creates a dropdown widget for selecting the graph type.

23. project\_name(): Creates a dropdown widget for selecting the project name.

24. studet\_name(): Creates a dropdown widget for selecting the student (user) name.

25. tab\_name(): Creates a dropdown widget for selecting the tab name.

26. start\_date\_picker(): Creates a date picker widget for selecting the start date.

27. end\_date\_picker(): Creates a date picker widget for selecting the end date.

28. activities\_quantity(filters\_dict, data): Plots the quantity of activities based on the filtered data.

29. work\_hours(filters\_dict, data): Plots the work hours of users based on the filtered data.

30. relative\_work(filters\_dict, data): Plots the relative work of users on a project tab based on the filtered data.

31. shorten\_description(description): Shortens descriptions to fit within a specified format.

32. common\_activities\_in\_project(filters\_dict, data): Plots the most common activities in a project based on the filtered data.

33. filters\_data(): Creates a UI component that displays the chosen filters.

34. create\_graph(filters\_dict, data): Generates and displays a graph based on the selected filters.

35. create\_parameters\_results\_screen(filters\_dict, data): Creates the results screen based on the filtered parameters.

36. filter\_data(df, start\_date, end\_date, user): Filters the data based on selected criteria.

37. add\_value\_labels(ax, spacing=5): Adds value labels to a bar graph.

38. export\_to\_excel(data, filename): Exports data to an Excel file.

39. show\_negative\_actions(filtered\_df): Displays a graph showing the number of negative actions by each user.

40. show\_project\_stats(filtered\_df, days=7): Displays project statistics including duration, total actions, and user participation.

41. show\_user\_progress(filtered\_df): Displays a cumulative graph of user actions over time.

42. show\_user\_effectiveness(filtered\_df): Displays the average number of daily actions for each user.

43. show\_action\_types(filtered\_df): Displays a graph showing the most common types of actions performed.

44. activity\_by\_day\_of\_week(filtered\_df): Displays the distribution of activity across different days of the week.

45. action\_type\_ratio(filtered\_df): Displays a pie chart showing the ratio of different action types.

46. user\_collaboration(filtered\_df): Displays a network graph showing user collaboration based on shared documents.

47. calculate\_effective\_work\_time(filtered\_df): Calculates and displays the effective work time for each user.

48. analyze\_trends\_over\_time(filtered\_df): Displays a graph showing trends in activity over time.

49. analyze\_work\_styles(filtered\_df): Analyzes and displays user work styles based on action timing.

50. analyze\_user\_contributions(filtered\_df): Displays user contributions to task progress based on positive actions.

51. update\_statistics(statistic, start\_date, end\_date, user): Updates the statistics displayed based on the selected filters.

52. on\_button\_click(b): Handles the event when the “Load Selection” button is clicked.

53. create\_statistics\_screen(): Creates the statistics screen UI for the application.

54. create\_index\_search\_screen(): Creates the index search screen UI for the application.

55. create\_json\_upload\_screen(): Creates the JSON upload screen UI for the application.

56. getUsers(): Retrieves and lists all users from the chatbot data.

57. getCountsOfTopActions(user\_input): Returns the counts of the most frequent actions based on user input.

58. getLastDateOfProject(): Returns the latest date that appears in the data.

59. getActionsOfAUser(user\_input): Finds the common actions that a user has made.

60. getDurationOfProject(): Returns the time difference between the start and the end of the project.

61. getNumberOfDocuments(): Retrieves the amount of different documents that appear in the data.

62. getTabsCountInProject(user\_input): Returns the counters of each tab that appears in the data.

58. getUsersWithTheirMostPopularAction(): Retrieves all users and identifies the most popular action each user has performed. It returns a string listing each user with their most frequent action.

59. getAllAvailableQuestions(patterns): Takes a list of patterns (questions) and returns a formatted string listing all available questions that the chatbot can answer.

60. CustomChat.respond(user\_input): Overrides the `respond` method of the `Chat` class to provide custom responses based on user input. It matches the input against predefined patterns and returns an appropriate response. If no match is found, it returns a default response.

61. create\_chatbot\_screen(): Creates the chatbot UI screen. It includes an input field for user messages, a send button, and a scrollable chat history display. The chatbot interacts with the user based on predefined patterns and responses.

62. home\_screen\_clicked(b): Handles the click event for the “Home” button, transitioning the user back to the home screen.

63. disconnect\_button\_clicked(b): Handles the click event for the “Disconnect” button, logging the user out and returning to the login screen.

64. chatbot\_clicked(b): Handles the click event for the “Chatbot” button, transitioning the user to the chatbot screen.

65. create\_menu(): Creates the navigation menu for the application, allowing users to navigate between different screens, including Home, Parameters, Statistics, JSON Upload, Index Search, and Chatbot. It also includes a disconnect button.

ביקורת עמיתים

במהלך הצגת הפרויקט בשבוע 7, קיבלנו את הביקורת הבאה מהעמיתים:

1. **חיבור מסכים**:
   * ההערה הייתה שהחיבור בין המסכים דורש שיפור. התגובה שלנו לכך הייתה לבצע בדיקה נוספת על חיבורי המסכים, לתקן את הבעיות שנתגלו, ולוודא שהמעבר בין המסכים חלק ונוח למשתמש.  
     בשלב הסטודיו, המערכת הייתה בשלב הכנתה, לכן המסכים הוצגו בנפרד. כמובן שלאחר מכן, הם חוברו למערכת אחת.
2. **עיצוב מסכים וכפתורים**:
   * הביקורת כללה הערות על העיצוב של המסכים והכפתורים, במיוחד בהתאמה בין העיצוב לפונקציונליות. בעקבות כך, ביצענו עדכונים לעיצוב הכפתורים והמסכים, בוצעו שינויים בצבעי הרקע ובצבעי הכפתורים תוך שמירה על פשטות ושימושיות.
3. **הצגת ערכים בעמודת עיון**:
   * ניתנה הערה על כך שחלק מהערכים בעמודות מסוימות לא היו מוצגים בצורה ברורה. לשם כך, שיפרנו את התצוגה כך שהערכים יהיו ברורים ונגישים למשתמש.
4. **ארגון וסדר**:
   * הייתה הערה על הארגון של האלמנטים במסכים. בעקבות כך, ביצענו ארגון מחדש של האלמנטים ושיפרנו את הסדר הפנימי של המסכים.  
     שינויים של הWidgets בוצע כך שיהיה בפריסה נוחה יותר למשתמש

**ציון SUS:**

הציון שחושב עבור ה -SUS (System Usability Scale) של המערכת היה  **8.096**. ציון זה מצביע על כך שהמערכת היא קלה ונוחה לשימוש, ומספקת חוויית משתמש חיובית.

**התייחסות לציון ה-SUS:**

1. **חוזקות המערכת**:
   * הציון הגבוה מראה שהמערכת שלנו קלה לשימוש, במיוחד בזכות השיפורים שביצענו לאחר הביקורת. זה גם מדגיש את הפשטות והתאמה למשתמשים שאינם טכנולוגיים.  
     המידע המופק יעזור למנהל הפרויקט להפיק מסקנות לגבי הצוות ופיקוח עליו
2. **תחומים לשיפור**:

* למרות הציון הגבוה, תמיד יש מקום לשיפור, במיוחד בנוחות המשתמש והפונקציונליות של המערכת.

אתגרים במהלך העבודה והתמודדות

במהלך הפרויקט ולאחר ה-design thinking היו סעיפים שבסופו של דבר החלטנו לממש ולהכניס למערכת והיו כאלה שלא הכנסנו מסיבות שונות כמו תיעדוף מטלות אחרות בפרויקט וכו'.

אחד מהאתגרים היה לממש סעיפים כמו התראות חיות, תצוגת לוח זמנים ותרומות חדשניות ולאחר שדנו בהם הגענו למסקנה שהם היו דורשים תשומת לב גדולה יותר וזמן רב יותר לפתח אותם, מפאת חוסר זמן והעדפה לסעיפים אחרים, החלטנו לוותר עליהם ולהתמקד במשימות אחרות.

כחלק מתהליך העבודה על הפרויקט, חלק מהסעיפים חולקו אישית לחברי צוות.

עוד אתגרים מרכזים שנתקלנו בהם היו כאשר חברי הצוות שקיבלו לממש חלק מהסעיפים כחלק מחלוקת העבודה, נתקלו בקשיים שנבעו בעיקר מחוסר ידע וניסיון בכלים וספריות שונות כמו לדוגמה matplotlib. בעת המפגשים הצוותים שהתבצעו בתדירות תכופה כדי להתעדכן ולעבוד יחד על התקדמות הפרויקט, הוצפו הבעיות וחברי צוות אחרים שהיו מנוסים יותר באותם כלים לקחו על עצמם לעזור ולקדם את אותם מטלות שחברי הצוות שהיו אחראיים עליהם נתקלו בבעיות. שיטת הפעולה הזאת תרמה רבות להתקדמות הפרויקט והצלחת האתגרים השונים.

בנוגע למטלת ה-code review, המשוב והערות שקיבלנו מהצוות השני, לא היה לנו אתגרים מיוחדים ובעיות שעלו כתוצאה מהם. סעיף 1 היה לטובה, סעיף 2 נבדק ויצא תקין, סעיף 3 תוקן ולא נתקלנו בבעיה וסעיף 4 הגענו להחלטה שאין צורך לטפל בו.

א.

Divergent Thinking:

1. תצוגת לוח זמנים (תצוגה שתכלול את עבודת המשתמשים בטווח זמן מסוים).
2. תצוגת בעיות ושינויים (כוללת את כמות הפעולות השליליות שביצע משתמש, למשל מחיקה של אלמנטים מהמודלים).
3. תצוגות דוחות וסטטיסטיקות (תצוגה עם דוחות מסוגים שונים, כמו של כמות הזמן שהושקעה על הפרויקט, כמות ה-tabs לפרויקט , זיהוי פערים בין התכנון לביצוע וכדומה).
4. תצוגת מסמכים פעילים ולא פעילים (מסמכים שבוצעו בהם שינויים בטווח זמן מוגדר האחרון וכאלו שלא).
5. תצוגת התקדמות המשתמשים (כוללת גרפים המראים התקדמות בשימוש באפליקציה, למשל בכמות הפעולות מסוג כלשהו, או עד כמה גדל השימוש בפעולות של import ו-edit על העתקים).
6. התראות חיות: תוספת של התראות בזמן אמת על בעיות או התקדמות יוצאת דופן.
7. תצוגה שמראה את התרומות החדשניות והשיפורים של המשתמשים (אוסף רעיונות שנצבר במהלך הפרויקט).

Convergent Thinking:

תצוגת לוח זמנים (תצוגה שתכלול את עבודת המשתמשים בטווח זמן מסוים, את התפלגות העבודה לאורך הזמן: כמות השעות שהוקצו לעבודה מול השעות שבוצעו בפועל. תציג אזורי עומס כדי לזהות משתמשים שעובדים מעבר ליכולת שלהם ולסייע להם.

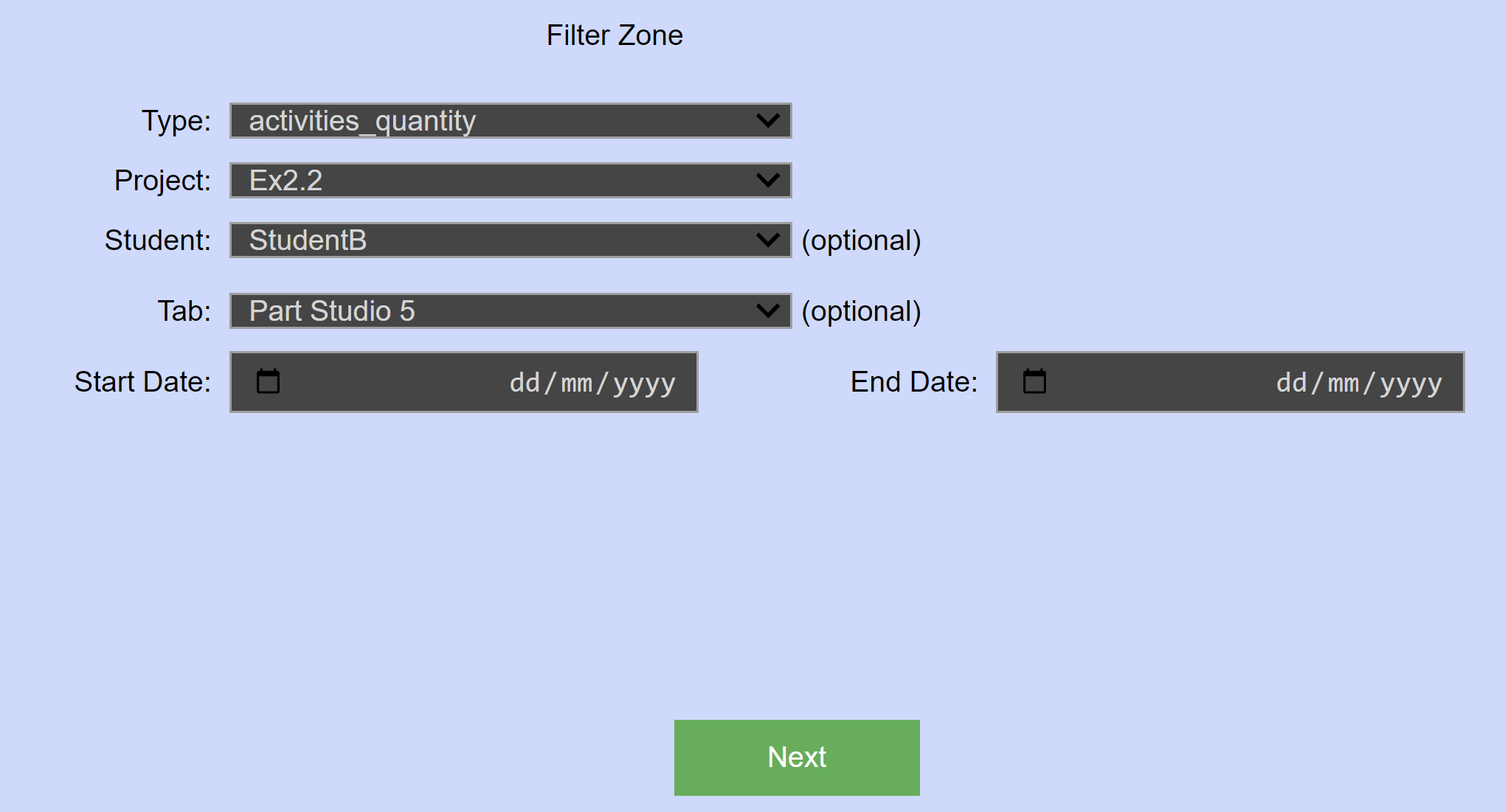
תצוגת הלוח זמנים צריכה להיות ברורה וקלה להבנה, וגם צריכה לספק אפשרויות לצפייה בנתונים ברמה של משתמשים ספציפיים או קבוצות קטגוריות שונות).

|  |  |  |
| --- | --- | --- |
| הערה | תגובה | האם בכוונת הצוות לבצע שינוי בעקבות ההערה? יש לפרט. אם כן, מה השינוי. אן לא, מדוע. |
| עיצוב יפה. | מודים על כך. | אין צורך בשינויים בעקבות הערה חיובית זו. |
| הייתה שגיאה במסך הlogin לאחר שבוצע login שגוי. | לא ראינו שגיאה בעת הרצת הקוד במחשבים שלנו. | לא נבצע שינוי, כי לא זוהתה השגיאה. |
| מלל הjson בpreview לא מכיל תמיכה בשפה העברית. | אכן אין תמיכה בהעלאת קובץ אשר מכיל אותיות מהשפה עברית. | כן, ננסה לשנות את האופן בו קובץ הjson מוצג עבור המשתמש. |
| בחירה של סוג הגרף (למשל pie chart וכדומה). | לא ניתן לבחור סוג גרף, אך אפשר לבחור קטגוריות של גרפים (המידע שיוצג). | לא, מכיוון שזה יחייב שינויים רבים. בנוסף, לא כל סוג מידע שאנו מציגים יכול להתבטא בסוגים שונים של גרפים. |

ב.

שקיפות אלגוריתמית

במערכת שלנו, בעמוד הפרמטרים ניתן להציג גרף המבוסס ע"פ הפרמטרים הבאים:



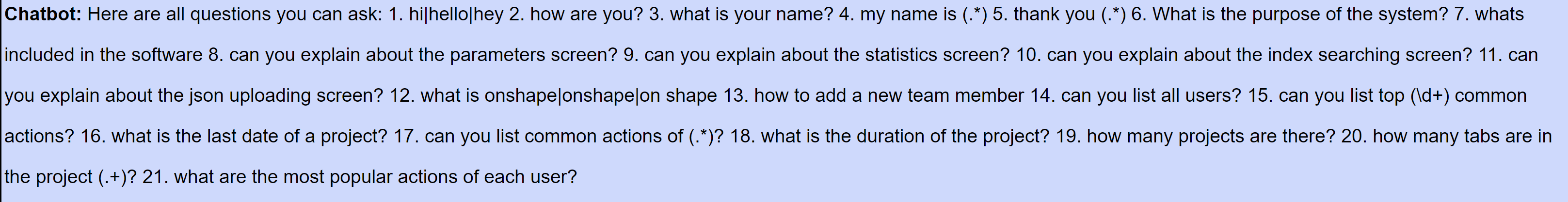
למעשה הנתונים המוצגים למשתמש לבחירה הם הנתונים שנשמרים במערכת.

במערכת שלנו בעמוד הסטטיסטיקות המשתמש יכול לגשת לנתוני פעילות ושימוש (מתוך רשימה: פעולות שליליות, נתוני פרויקט, התקדמות משתמש, יעילות משתמש, סוגי פעולות, פעילויות לפי יום בשבוע, יחס סוג פעולה, שת"פ משתמשים, יעילות זמן עבודה, טרנדים לאורך הזמן, סגנונות עבודה, תרומת משתמש) של משתמשים אחרים לפי תאריכים מבוקשים, אלה למעשה הנתונים הנאספים שהמשתמש יכול לגשת אליהם.

בנוסף, בעמוד העלאת קובץ JSON ניתן להעלות קובץ מסוג JSON והוא יישמר בdatabase.

בעמוד חיפוש האינדקס, ניתן לחפש מילים מסוימות ותופיע כמות הפעמים שהן מופיעות.

בעמוד הChatbot ישנו פירוט על 21 שאלות שהתשובות אליהן ניתן למענה ודאי:



כמו כן, קיים תיעוד (קצר) בקוד אשר מסביר מה כל אלגוריתם מבצע וכיצד.

התייחסות למשובים על הצגת הפרויקט

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| הגרפים שמוצגים למנהל מבלבלים | לא | לדעתנו, הגרפים מסודרים היטב ומכילים מידע מפורט ומובן. |
| הגרפים שמוצגים למנהל קצת מסורבלים לעין | כן | הגרפים יכולים להיות מסורבלים (במסך הפרמטרים) בעקבות גודלם הקטן על המסך, לכן ניתן להגדיל את ממדי הגרף. |
| באחד מהגרפים המוצגים למנהל תפריט המפתח עולה על הגרף | כן | בדומה להערה הקודמת, הגדלה של ממדי הגרף תגרום לסידור טוב יותר של האלמנטים של הגרף. |
| יש לסדר את הגרפים כדי שיהיו ברורים יותר ללקוח | לא | הגרפים מסודרים במקום במסך שנוח עבור משתמש לצפות בהם. בנוסף, אין צורך למשתמש לגלול במסך כדי לראות את הגרפים המוצגים. |
| להוסיף עוד גרפים (מעניינים) | לא | למרות שתמיד ניתן להוסיף גרפים, לעניות דעתנו, כמות הגרפים הקיימת במסך הפרמטרים ומסך הסטטיסטיקות יכול להספיק עבור מנהל הפרויקט על מנת להפיק לקחים מתאימים. |
| לסדר את דף הבית | לא | דף הבית מסודר ובעל כתב גדול כדי שיהיה ניתן לזהות את כל הרכיבים בו. לכן, אנו לא רואים צורך בשינוי דף זה. |

מקורות

**מקורות מידע:**

1. **ChatGPT** שימש ליצירת קטעי קוד והסברים הקשורים ל-HTML, גרפים ב-Python, וטיפול בשגיאות.
2. **GeeksforGeeks** שימש למידע והדרכות בנושא פיתוח אתרים ותכנות ב-Python.
3. **Cloud AI** כמו GPT
4. **Stack Overflow** שימש לפתרון בעיות תכנותיות, במיוחד בנושא Python, תוך שימוש בפתרונות שנכתבו על ידי קהילת המפתחים.
5. **W3Schools** שימש ללמידה והבנה של HTML, CSS, ופיתוח אתרים בצורה כללית.

**פרומפטים בChatGPT:**

1. **פרומפט** "I will give you this widget, make it a HTML style"

**תגובה:** נוצר קוד HTML ו-CSS לעיצוב ווידג'ט נתון, במטרה ליצור עיצוב רספונסיבי ואסתטי.

1. **פרומפט** "תכין לי פלוט בפייתון עבור x וy "

**תגובה:** נוצר קוד ב-Python ליצירת גרף באמצעות Matplotlib, הכולל גרף של x נגד y.

1. **פרומפט**"What this x error means, fix and add"

**תגובה:** הסביר את הסיבה לשגיאה, סיפק פתרון והציע קוד נוסף למניעת בעיות דומות.

1. **פרומפט**"Add more" פרומפט זה יכול לבוא בצורות שונות והוא תמיד בא לאחר בקשה מסויימת שהייתה טובה ורצוי עוד ממנה.

**תגובה:** הוצעו הצעות להרחבת הפונקציונליות או לשיפור הקוד הקיים, כולל תכונות נוספות, אופטימיזציות או טיפול טוב יותר בשגיאות.

1. **פרומפט**"add comments for the code here the code"

**תגובה** **:** תיעדתי לך את הקוד לבקשתך

1. **פרומפט** “איך יוצרים כפתור פשוט ב-Jupyter באמצעות ipywidgets?

**תגובה :**כדי ליצור כפתור פשוט ב-Jupyter באמצעות ipywidgets, תוכל להשתמש במחלקה Button מ-ipywidgets. הנה דוגמת קוד פשוטה: ...

1. **פרומפט** "איך משתמשים ב-DatePicker ב-Jupyter כדי לאפשר למשתמש לבחור תאריך?"

**תגובה :** כדי להשתמש ב-DatePicker ב-Jupyter ולהציע למשתמש אפשרות לבחור תאריך, תוכל להשתמש במחלקת DatePicker מתוך מודול ipywidgets. הנה דוגמת קוד פשוטה: ...

1. **פרומפט** "איך יוצרים Tab Widget ב-Jupyter שמכיל תוכן שונה בכל כרטיסייה

**תגובה :** כדי ליצור Tab Widget ב-Jupyter עם תוכן שונה בכל כרטיסייה, תוכל להשתמש במחלקת Tab מ-ipywidgets. הנה דוגמת קוד שמראה איך לעשות זאת: ...