**סקר ספרות ודרישות:**

**חלק א':**

1. **תקציר:**

תחומי טכנולוגיה, כמו הנדסה, מתפתחים ומתרחבים בקצב מהיר מאוד. על מנת לעמוד בקצב הגידול של תחומים אלו, על החוקרים והעוסקים בהם לנהל בצורה יעילה את הידע שגדל בצורה מהירה גם כן. על החוקרים לשמור סקירה עדכנית ככל הניתן של הבעיות הטכניות, וכן על הפתרונות שקיימים לבעיות בתחום אותו הם חוקרים.

כחלק מן האפשרויות השונות אשר פותחו על ידי נותני השירות ברחבי האינטרנט לייעול הנגישות לידע – גובש רעיון מיפוי המידע. מפת ידע הינה הצגה גרפית של מערכת מושגים, המציגה את מערכות היחסים בין אלמנטים ומושגים שונים.

דוגמה למפות ידע הן מפות ה – “Means-Ends” כפי שהוצגו במאמר Sturm et. al (2017). מפות אלו משמשות לניהול Know-How information (כיצד לבצע פעולה בצורה טובה) כאינדקס למידע, ומתמקדות ביחסי הגומלין בין בעיה לפתרונה (“means-ends”). המפות מתמקדות בפירוש הידע (לכדי מטרות ואפיונן) על מנת לבצע התקדמות לכיוון הפתרון.

1. **מבוא:**  
   במרוצת השנים, כחלק מפיתוחי הטכנולוגיות השונות, גם המידע אשר נמצא ברשת קיבל תצורה חדשה שמתעצבת ומתגברת עד לימים אלו.

ראשית, הנגישות למידע הפכה לפשוטה יותר בעבור המשתמשים השונים וכך גם העלאתו לרשת. כמות מידע גדולה מאוד החלה להיאגר במקורות האחסון וחברות שונות היו צריכות להיערך לכך בהתאם: כאשר הפצת מידע הפכה לכמעט פעולה מידית – משתמשים רבים החלו לעלות "כמעט כל דבר" לאן שרצו. בנוסף, מן העובדה שהמידע הפך נגיש יותר – משתמשים וגורמים רבים החלו להסתמך עליו בשגרת יומם ולהיעזר ברשת ככלי אינטגרלי כאשר רצו להעביר מידע בכל דרך שהיא. זאת ועוד, ניתן לראות כי חלה מגמה אשר עודדה מציאת פתרונות לבעיות שונות בכלל (החל בעיות יומיות ועד לבעיות מקצועיות) ברחבי הרשת ובפרט עלה הביקוש והצורך ב- Know-How information (=הידע כיצד לבצע משהו בצורה טובה).

כעת, שברוב הרשת ניתן למצוא בקלות כל מידע שהמשתמש חפץ למצוא: תופעת הצורך בסוג מידע כזה הולכת וגוברת עד היום.

כחלק מן האופציות השונות אשר פותחו על ידי נותני השירות ברחבי האינטרנט לייעל את הנגישות לידע – גובש רעיון מיפוי המידע (אשר תוצרו הינו מפת ידע). מפת ידע הינה הצגה גרפית של מערכת מושגים. המפה מאפשרת לחשוב על מערכות היחסים בין האלמנטים והמושגים השונים בתוך מערכת סבוכה ולהציגם באופן פשוט למשתמש ובכך להקל את החיפוש באשר יהיה. אלגוריתמים ומודלים שונים של מיפוי ידע קיימים כבר כיום ומייעלים את החיפוש. אחד המודלים אשר הוצע ע"י Sturm et. al (2017) הינו Means-ends based know-how mapping אשר ממפה את המידע על סמך ההבנה כי בין כל האלמנטים שונים במפה קיימים יחסי "אמצעי" (means) ו"תוצאה" (ends). מפות ה – ME משמשות כאינדקס לידע ולא מכילות את כל המידע הממשי, נוצרו ע"מ להתמודד עם הבעיות העולות עקב כמות מידע גדולה. המפות מתמקדות בפירוש הידע ושמירתו ע"מ לבצע התקדמות לכיוון הפתרון, שימוש בקטעי מידע הניתנים לשימוש חוזר ולא פתרון ממוקד לבעיה מסוימת.

המטרות לשימוש במפות אלו הן:

* **הפחתת עומס המידע** בעזרת בחירת האלמנט המתאים לייצוג הידע הרצוי, task לייצוג בעיה או פתרון ו- quality המשמש לאפיון ה- task אליו הוא מקושר.
* **התאמת הפתרונות והחידושים בהקשר הנכון**, ובכך ניתן להצביע על אפשרויות נוספות**.**
* **הבנת היתרונות והחסרונות של פתרונות קיימים**, ובכך לאפשר קבלת החלטות מושכלת יותר.
* **תמיכה בהתפתחות התמידית של המידע**, שמירה על עדכניות המידע עם הגעת ידע חדש.

למשתמשים במערכת שלנו יוצג המידע בצורה גרפית מה שיאפשר לדגום בעיות ופתרונות דומים או קרובים ליעד אותו הם רוצים להשיג. ניווט פיזי בין המפות השונות על גבי המידע יאופשר ובכך יקל על מחקר הבעיה ומציאת הפתרון. כיום רוב החיפוש אחר מידע ברשת מתבצע ע"ב דמיון מילולי בין המידע הקיים לבין השאילתה של המשתמש, שימוש במפות ידע מסוג זה יאפשר חיפוש מידע ע"פ קשרים נוספים בין מקורות המידע.

**מטרתנו בפרויקט זה** הינה פיתוח מערכת אשר תתמוך ביצירה, שינוי וניהול מפות ידע שכאלו.

1. **ME-MAP:**

מודל שפת ה-ME ממפה בעיות ופתרונות, ואת מערכות היחסים שבניהם. בבסיס המודל עומדים אלמנטי המודל – Task ו-Quality. לאלמנטים אלה ניתן לקשר Reference אחד או יותר למקורות מידע, ולקשר ל-Contexts אשר מספקים מידע נוסף על הסביבה או ההקשר שבו האלמנט נידון (לדוגמה, הנחות מסוימות שהונחו). ה-Reference וה-Contexts אינם מוצגים במפה באופן מידי, אך מהווים חלק מהותי מהמפה, שכן הם מספקים סימוכין לידע המוצג במפה.

אלמנט מודל – Task: Task הוא בעיה, או פתרון. לאלמנט ניתן שם, פועל תמציתי. Task-ים מוצגים גרפית כמלבנים במסגרת השפה.

אלמנט מודל – Quality: Quality מתאר מאפיין רצוי עבור Task. בשפת ה-ME-Map, Qualities מיוצגים גרפית על ידי אליפסות. השם שניתן ל-Quality יהיה לרוב שם תואר או שם פעולה. Quality מקושר ל-Task, אך עשוי להיות מקושר גם ל-Quality נוסף. נתייחס לקישור כזה בהמשך.

מלבד אלמנטי המודל, השפה מגדירה גם קישורים מסוגים שונים, להם משמעות סמנטית שונה, בין אלמנטי המודל.

לינק Achieved by: מיוצג ע"י חץ חד כיווני, כאשר מוצא החץ מסמן את המטרה (“ends”), וסופו באמצעי (“means”). הלינק מסמן שקיימות מספר חלופות להשגת המטרה, ושניתן לבחור באחת מהן.

לינק Consists of: מיוצג ע"י חץ חד כיווני, כאשר מוצא החץ מסמן את המטרה (“ends”), וסופו באמצעי (“means”). הלינק מסמן שבכדי להשיג את המטרה, יש צורך לפתור מספר תתי-בעיות.

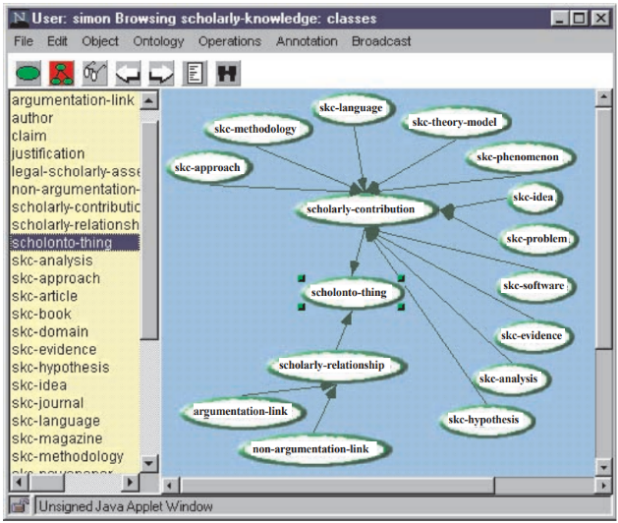
לינק Extended by: מיוצג ע"י חץ חד כיווני, כאשר מוצא החץ מסמן את המטרה (“ends”), וסופו באמצעי (“means”). הלינק מחבר בין שני Task-ים, ומסמן ש-Task האמצעי הוא למעשה הרחבה, הוספה של פרטים, או הגדרת המטרה במונחים מעשיים יותר.

לינק Association: מיוצג על ידי קו המחבר בין Task לבין Quality. הלינק מתאר את המאפיינים הרצויים בעבור ה-Task.

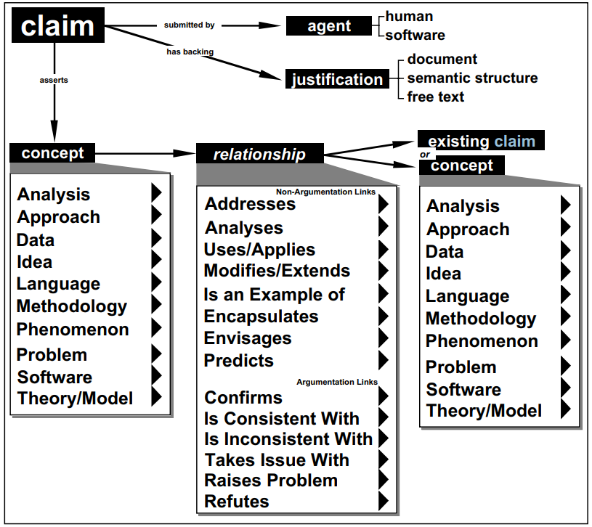
לינק Contribution: מיוצג ע"י חץ חד כיווני, עם ראש חץ מעוקל. הלינק יוצא מ-Task או Quality, ומסתיים ב-Quality. ה"תרומה", או ה-contribution של Quality היעד לאלמנט המקור מיוצגת על ידי סימנים: "++", "+", "-", "- -". התרומה לאלמנט המקור נקבעת באופיין סובייקטיבי ע"י בונה המפה.

1. **עבודות קיימות:**

**ScholOnto**, כפי שהוצג במאמר Shum et al. (2000), היא דוגמא למערכת המשמשת כשרת דיגיטלי ספרייה של מאמרים אקדמיים. השרת תומך באינטרפרטציה שונה של כל קורא למאמרים ובדו-שיח בנוגע לנושאי המאמרים, מה שמאפשר לחוקרים לתאר ולהתעמת דרך רשת סמנטית בה מתוארים התרומות וההקשרים של מאמר מסוים לשאר המאמרים בנושא.  
השרת תומך בשירותי חישוב המבוססים על תיאוריית ה-ontology וישנם ממשקים אלטרנטיביים התומכים באינטראקציה ברשתות סמנטיות גדולות.  
ע"מ להוסיף מאמר אל הספרייה על המפרסם להוסיף גם Claim על המאמר שלו. לכל claim יש "בעלים" שהוא agent (אדם או תוכנה), ישנה הצדקה כלשהי לקיום אותו claim ובעזרת claims ניתן ליצור יחסים גומלין עם claims אחרים או בין קונספטים שונים.

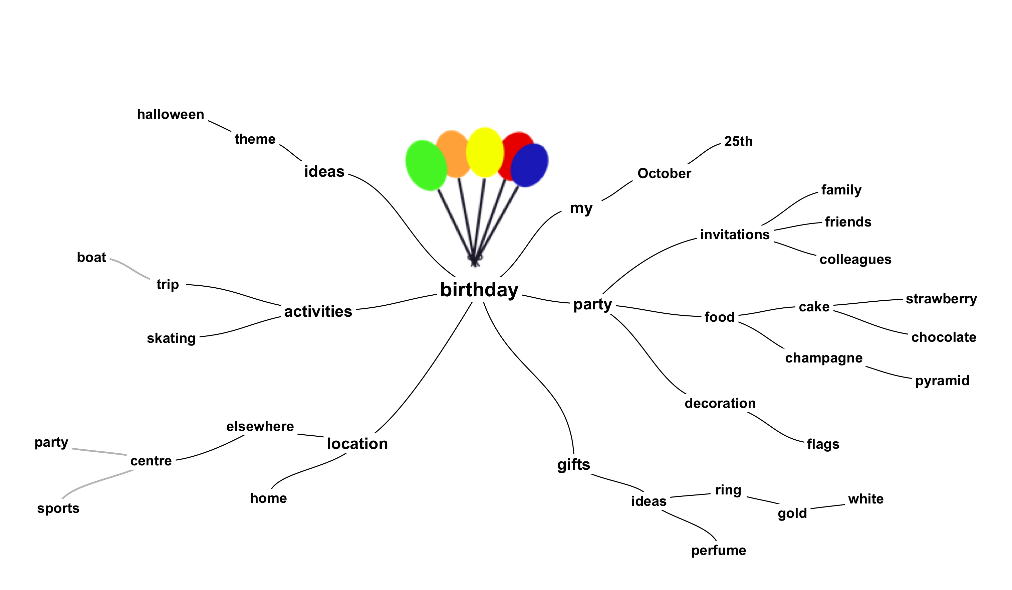


מבנה של מחלקות ב- ScholOnto ontology



מבנה של Claim ע"פ המאמר

**Mind maps** בהגדרתן הן שיטת ייצוג ותיעוד גרפי של ידע לצרכי חשיבה. הצגה היררכית של קשרים בין רכיבים שונים המרכיבים שלם מסוים, רוב המפות עוסקות בנושא אחד המופיע במרכזן. סביב רעיון המרכזי במפה מקשרים אסוציאציות, רעיונות, מושגים, משימות וכו'. המפה יכולה לשמש ככלי לייצוג ידע, לשיפור הלמידה והזיכרון ולתיאור רעיונות מורכבים. לתצוגה גרפית היררכית השפעה על האופן בו אנו מסדרים את המידע ומעניקים לו משמעות.



דוגמא למפת מידע העוסקת ביום הולדת

השוואה בין המערכות הקיימות למערכת ה- ME-Maps:

|  |  |  |  |
| --- | --- | --- | --- |
| **קריטריון** | **ScholOnto ontology** | **Mind maps** | **ME-Maps** |
| תכולת המודל | אלמנט יחיד שמכיל בתוכו את כל המידע המתאר את הנושא המוצג – ע"מ להבין יותר על הנושא יש לקרוא את המידע המצורף | כלי ויזואלי בלבד, אין אפשרות לקישור למקורות ידע נוספים או הוספת מידע על הנושא המיוצג באלמנט | ישנם כמה סוגי אלמנטים המציגים את המידע על הנושא המתואר בצורה יותר נגישה – task המציג את הנושא הרחב ו- qualities המאפיינים את הנושא.  בנוסף כל אלמנט במפה ניתן לקישור למאמרים/ דפי מידע נוספים |
| מהות המודל | ישנו מודל המתבסס על קישור האלמנטים במפה על סמך הקשרים בין המאמרים אותם הם מתארים.  הקשרים נוצרים ע"פ קריטריונים רבים המוגדרים במודל (גישות המוצגות במאמר, מידע, meta-data, רעיונות ועוד), מה שמקשה על יצירת מפה ומסבך את המודל | אין מודל מוגדר ועל כן כל האלמנטים במפה מסוג אחד ומתארים קשר היררכי בלבד | ישנו מודל מוגדר ופורמלי, בו ישנם אלמנטים מסוגים שונים המגדירים את הקשרים השונים בין הרעיונות.  ניתן לקשר בין אלמנטים שונים לא רק על בסיס קשר בין המאמרים הקשורים. לדוגמא ניתן לחבר בין task המקושר למאמר מסוים לבין quality המתאר את אותו ה- task ולא מתקשר למאמר אחר |
| רמת פירוט | המפה משמשת להצגת הקשרים בין מאמרים בשרת | המפה משמשת להרחבת הידע על נושא מסוים | המפה משמשת למציאת פתרון לבעיה מסוימת |
| קשרים במודל | כל אלמנט (claim) מקושר למאמר מסוים והקישורים בין האלמנטים מבוססים על קשר בין המאמרים בלבד | בכל מפה מוצג רעיון אחד מרכזי ואליו מקושרים רעיונות משנה | כל task מהווה מטרה/משימה שיש לבצע ובכך יכולים להיות כמה tasks במפה המתארים מטרות שונות |

1. **סקירת טכנולוגיות רלוונטיות:**

* **Back-End**: נרצה שפה או פלטפורמה אשר תאפשר לנו לשרת מספר רב של לקוחות במקביל, בעלת קהילה רחבה שתוכל להציע לנו תמיכה בתהליך הפיתוח, ועבודה נוחה מול בסיס נתונים.

נוכל לבחור מבין הטכנולוגיות NodeJS, Java, או Asp.Net.

* NodeJS היא פלטפורמה להרצת קוד JavaScript מחוץ לדפדפן, קוד פתוח, מאפשרת התמודדות עם מספר רב של מחוברים במקביל. השפה מבוססת JS מה שמאפשר עבודה בשפה זהה גם ב- front end.

השפה פופולארית ובכך ישנן המון חבילות תוכנה בהן נוכל להשתמש.

* Java היא שפה מוכרת ומבוססת, אך פיתוח מערכות web באמצעותה הוא מסורבל לעומת האלטרנטיבות, שכן היא דורשת קומפילציה. בנוסף, ניהול משימות אסינכרוניות בjava דורש מימוש של observer ו-observable, בעוד בNode.JS המנגנון ממומש כבר בשפה באמצעות Promises. בהקשרים האלו, Asp.net דומה מאוד, ודורשת קומפילציה אף היא. גם מנגנון התהליכים האסינכרוניים דומה לJava.

אם כן, נבחר עבור הbackend את NodeJS לאור התמיכה שלה בקוד אסינכרוני (שהוא חלק אינהרנטי מהשימוש בDB), ונוחות הפיתוח ללא צורך בקומפילציה. ניתן להריץ Live Server שמתעדכן בקוד החדש בכל רגע באופן אוטומטי.

* **Front-End:** גם בfront-end נרצה שפה או פלטפורמה בעלת קהילה רחבה. בנוסף, נעדיף אפשרויות אשר יסייעו לנו לכתוב קוד איכותי שיהיה קל לתחזוקה והרחבה בהמשך.

נוכל לבחור באחת מבין האפשרויות הבאות: Angular, React, VueJS ו - jQuery.

* Angular –היא framework של JavaScript, מבוססת TypeScript. מאפשרת שינויים ב- DOM בצורה דינאמית בעת הריצה. מקלה את הפיתוח לעומת JS. Angular היא שפה שמפותחת, נתמכת, וגם מפתחים באמצעותה ב-Google, מה שמבטיח תמיכה והמשך פיתוח בעתיד הנראה לעין. בנוסף, Angular מגיעה out of the box עם מימושים ופתרונות לבעיות שניתקל בהן, כמו router, form validation, httpClient ועוד.

אפשרויות נוספות שעומדות בפנינו הן Vue, jQuery ו-React. ארבע האפשרויות הן למעשה frameworks מבוססי JavaScript.

* React היא למעשה ספרייה שנועדה לפיתוח UI ע"י Facebook. ככזאת, היא אינה מגיעה עם כלל הכלים הנדרשים שמגיעים עם Angular, ותדרוש מאתנו שימוש בספריות נוספות, מה שייצור תלות בחבילות חיצוניות ונקודות כשל אפשריות במקרה של שדרוגים או עדכונים בחבילות אלה. מבחינת ביצועים, React עדיפה על פני Angular, אך האפליקציה שלנו אינה דורשת ביצועים מהירים בצד הלקוח, אשר יממש תהליך ארוכים באמצעות GUI, ולא חישובים ארוכים ואוטומטיים.

לעומת React, Angular תומכת בdependency Injection, ומאלצת את המפתחים לחלוקה ברורה של MVC. שתי עובדות אלה מאלצות את הפיתוח להיות מודולרי ופשוט לבדיקה.

* Vue.Js היא framework צעיר יחסית, ללא חברה גדולה שעומדת מאחוריו, או קהילת פיתוח גדולה וחזקה כמו שיש ל - React ול - Angular. לכן ישנן כרגע פחות חבילות מוכנות, ופחות תמיכה.
* jQuery היא ספרייה של JavaScript אשר מאפשרת לבצע מניפולציה על ה - DOM באופן דינאמי בזמן הריצה.

בדומה ל - React, שימוש ב - jQuery ידרוש מאתנו שימוש בחבילות נוספות, שכן מדובר בחבילה בלבד, ולא ב - framework שלם. כמו כן, אין את ההפרדה המוגדרת ל -MVC. בנוסף, Angular תומכת ב - data binding, בעוד ב-jQuery יש צורך לטפל בכל מקום שבוא נרצה לקשר בין data לבין תצוגה באופן ידני.

אם כן, נבחר ב-Angular לאור העובדה שמדובר ב-framework עם תמיכה גדולה, שכולל פונקציונאליות רבה שנזדקק להן. כמו כן, החלוקה המוגדרת ל-MVC תאפשר לנו לכתוב קוד בצורה יעילה יותר.

* **DBs:** נתייחס תחילה לשלוש דרכים עקרוניות לניהול המידע בבסיס הנתונים.

1. בסיס נתונים רלציוני: זהו בסיס הנתונים שנמצא בשימוש הנרחב ביותר נכון להיום. ביסוד בסיס הנתונים מוגדרות טבלאות (רלציות) בעלות מבנה מוגדר, אשר בהן מסודר המידע. בכל רלציה מוגדר מפתח (שדה אחד או יותר) אשר מזהה באופן חד-ערכי רשומה בטבלה.
2. בסיס נתונים גרפי: בסיס נתונים גרפי מתאים מאוד במקרים בהם המידע שנשמר והיחסים שבין פרטי המידע, מיוצגים בקלות באמצעות גרף (לדוגמה – רשתות חברתיות, תשתית כבישים ועוד).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **רלציוני** | **גרפי** | **מסמכים** |
| מבנה מוגדר עבור כל פיסת מידע | כן. מוגדר בעבור כל רלציה | לא. מוגדר על הקשר בין כל קודקוד וקודקוד. | לא. לכל מסמך עשוי להיות מבנה משלו. |
| עבודה מול שפות OOP | דורש התאמה, שכן המידע של כל אובייקט פרוש על פני מספר טבלאות. | כל Node ניתן להציג כאובייקט OOP, אך לא כל אובייקט ניתן להציג כגרף. ההתאמה תלויה בעולם התוכן. | המעבר משפת תכנות OOP הינו טריוויאלי. |
| קהילה ותמיכה | זהו המודל המקובל והנפוץ ביותר בעולם. | ישנה תמיכה מסויימת. | ישנה תמיכה מסויימת. |
| תמיכה בשינוי מבנה המידע | אפשרי, אך משפיע על כלל המידה הקיים, וידרוש שינויים בשאילתות קיימות. | ניתן לשנות את מבנה המידע | בגלל שאין מבנה שמוגדר מראש, אין שום בעיה לשנות את המבנה במהלך הזמן. |

1. בסיס נתונים מבוסס מסמכים: בבסיס נתונים זה נשמרים מסמכים בעלי שדות, ובשדות אלו נשמר המידע. עבור כל מסמך ישנו ID אשר מזהה אותו באופן חד ערכי. ישנו סינטקס אשר מאפשר הרצת שאילתות כדוגמת "החזר את כל המסמכים שיש להם שדה number וערכו הוא 2".

זו הנקודה לציין שלוש עובדות מהותיות הקשורות לאופן קבלת ההחלטה לגבי סוג בסיס הנתונים שנבחר:

* 1. אחת הדרישות שעלו בזמן הראיונות עם הלקוח, הייתה תמיכה בשינוי מודל המפות העומד בבסיס הפרויקט. לאור דרישה זו, נעדיף בסיס נתונים אשר יאפשר לנו לשנות את מבנה המידע הנוכחי למבנה אחר.
  2. הפרויקט שלנו מתממשק לפרויקט נוסף, אשר יעסוק בביצוע חיפושים על גבי המפות אשר יאוחסנו בבסיס הנתונים שלנו. האלגוריתם בבסיס פרויקט החיפוש אינו כתוב עדיין, ולכן אין אפשרות לדעת בבירור אילו שאילתות יריץ על המפות. נכון לשלב זה, צוות הפרויקט טוען שירצה לקבל מפות שלמות, ולא חלקי מפות, או אלמנטים מסוימים מתוך המפות.
  3. ישנם שני אלגוריתמים חיצוניים נוספים אשר מסתמכים על האופן שבו מאוחסנות המפות בבסיס הנתונים. אלגוריתם אחד מוודא את תקינות המפה מבחינת הסינטקס שלה, ואלגוריתם שני מאפשר לייצר מפה חדשה שהיא הפשטה של מפה קיימת (כלומר, בעלת מספר Task-ים נמוך יותר, באופן שמציג את התמונה ברמת הפשטה גבוהה יותר). שני האלגוריתמים עובדים כרגע עם ייצוג של JSON של המפות. ההתממשקות לאלגוריתמים אלה תדרוש מאתנו להמיר הצגות אחרות חזרה לייצוג JSON-י, כדי לאפשר את התמיכה.

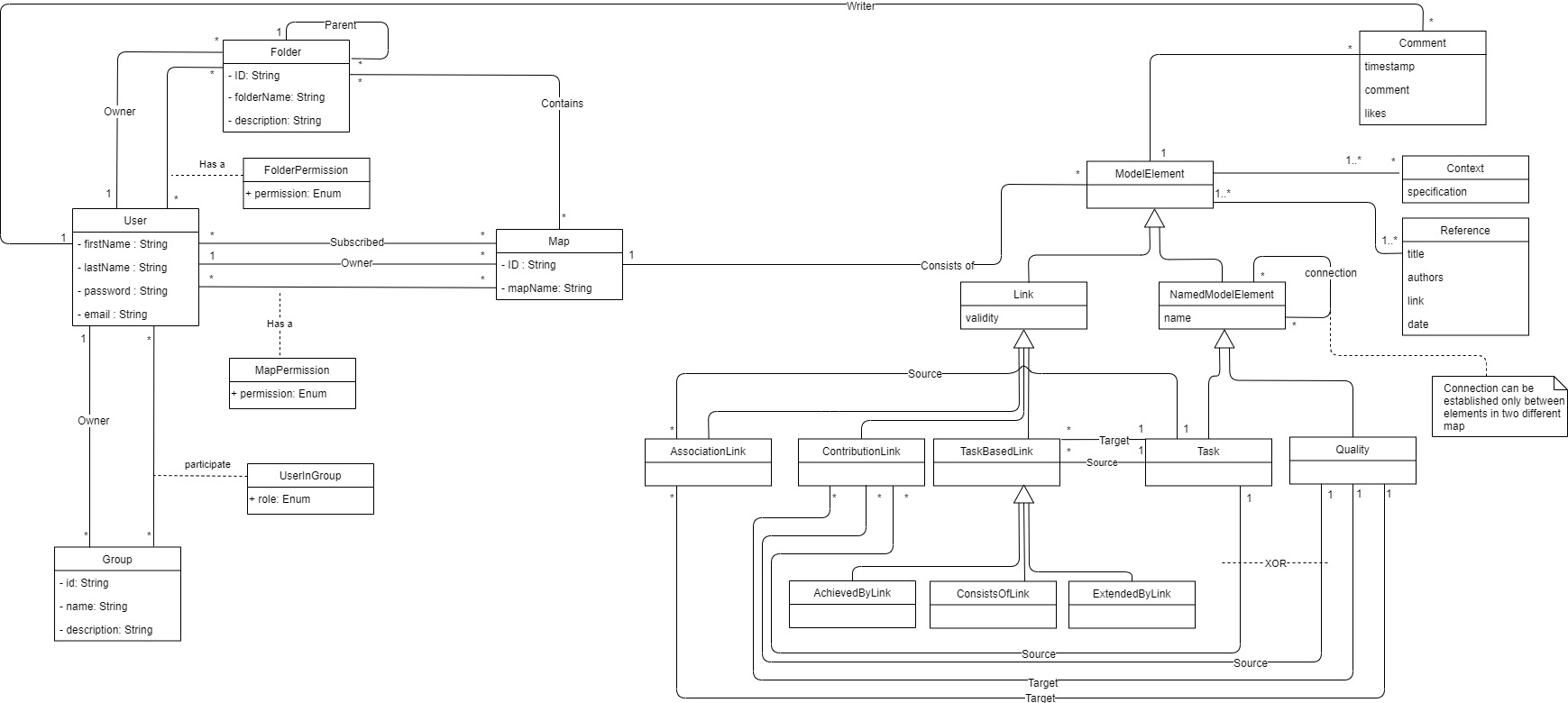


Figure 1 Class Diagram

אם נתבונן בשרטוט הclass diagram, נשים לב שניתן לחלק את המערכת לשני חלקים עיקריים:

* 1. חלק העוסק במפות. חלק זה מתאים מאוד למימוש באמצעות בסיס נתונים מבוסס גרף, שכן כל מפה היא למעשה גרף.
  2. חלק העוסק בניהול הרשאות ומשתמשים. חלק זה מתאים למימוש באמצעות בסיס נתונים רלציוני קלאסי.

בשלב זה אנו מתלבטים בין שתי אפשרויות. האפשרות הראשונה היא לממש בפרויקט שני בסיס נתונים, כאשר אחד מהם גרפי, ומנהל את המידע הנוגע למפות, והשני רלציוני ומנהל את המידע הנוגע להרשאות.

ארכיטקטורה כזאת תאפשר לנו לבצע בקלות שינויים במבנה המפות בעתיד, תמיכה בשאילתות גרפיות, וכמובן תאפשר ניהול הרשאות. מצד שני, החזקת שני בסיסי נתונים תגדיל את מורכבות הפרויקט, וכמובן תהפוך שאילתות הנוגעות גם להרשאות וגם למפות עצמן למורכבות יותר, בשל הצורך לקבל מידע בשני ייצוגים שונים כדי לקבל החלטה.

האפשרות השנייה היא לנהל בסיס נתונים אחד בלבד, אשר ינהל את שני חלקי המערכת. יתרונות הארכיטקטורה הם פשטות המערכת הומוגניות של המידע. החיסרון של פתרון כזה הוא שלפחות אחד (או שני) החלקים ייוצג באופן שאינו מיטבי.

עבור ארכיטקטורה זו, נבחר בבסיס נתונים של מסמכים, אשר מהווה פשרה סבירה עבור שני חלקי המערכת.

1. **ביבליוגרפיה:**

* Sturm, A., Gross, D., Wang, J., & Yu, E. (2017). Means-ends based know-how mapping. Journal of Knowledge Management, 21(2), 454-473.‏
* Shum, S. B., Motta, E., & Domingue, J. (2000). ScholOnto: an ontology-based digital library server for research documents and discourse. International Journal on Digital Libraries, 3(3), 237-248.‏

**חלק ב':**

1. **מטרת המערכת:**

נפריד את המערכת לשני חלקים: שימוש מפות ה- ME וממשק המערכת.

המטרות של שימוש במפות ME: תחומי טכנולוגיה כמו הנדסה מתפתחים ומתרחבים בקצב מהיר מאוד, ע"מ לעמוד בקצב הגדילה של תחומים אלו על החוקרים והעוסקים בהם לנהל בצורה יעילה את הידע שגדל בצורה מהירה גם כן. על החוקרים לשמור סקירה עדכנית ככל הניתן על הבעיות הטכניות, וכן גם על הפתרונות שקיימים לפתרון הבעיות בתחום אותו הם חוקרים.

מפות ה – ME משמשות כאינדקס לידע ולא מכילות את כל המידע הממשי, אלו נוצרו ע"מ להתמודד עם הבעיות העולות עקב כמויות המידע הגדולות, ומתמקדות ביחסי הגומלין בין בעיה לפתרונה (“means-ends”).

המפות מתמקדות בפירוש הידע ושמירתו ע"מ לבצע התקדמות לכיוון הפתרון, שימוש בקטעי מידע הניתנים לשימוש חוזר ולא פתרון ממוקד לבעיה מסוימת.

**הפחתת עומס המידע** בעזרת בחירת האלמנט המתאים לייצוג הידע הרצוי, task לייצוג בעיה או פתרון ו- quality המשמש לאפיון ה- task אליו הוא מקושר.

**התאמת הפתרונות והחידושים בהקשר הנכון**, ובכך ניתן להצביע על אפשרויות נוספות**.**

**הבנת היתרונות והחסרונות של פתרונות קיימים**, ובכך לאפשר קבלת החלטות מושכלת יותר.

**תמיכה בהתפתחות התמידית של המידע**, שמירה על עדכניות המידע עם הגעת ידע חדש.

מטרת ממשק המערכת: המערכת שלנו תשמש כפלטפורמה לחיפוש וניהול הידע הממודל כמפות. במערכת המשתמש יוכל ליצור ולעדכן מפות וכן לגשת אל המידע אשר ממודל במפה.

**הצגה גרפית ומילולית** – אחת המטרות המרכזיות של המערכת היא הצגת המפות בצורה בה היא תהיה למשתמשים ברורה ונוחה לעבודה. ההצגה הגרפית מבוססת על המודל המתואר. בנוסף ניתן לייצג מפה בצורה מילולית באופן תבניתי המתאר את הקשרים במפה.

**ניהול מפות** – אדם יוכל ליצור מפות לפי מה שהוא רואה לנכון, עם הזמן יוכל לשנות מפות קיימות, כלומר להוסיף או להוריד אלמנטים שונים, להוסיף קישורים למאגרי ידע נוספים. יוכל לנהל את המפות לפני הנושאים המרכזיים המגדירים את המפה בנוחות. בנוסף יתאפשר חיפוש מילולי על המפות. קישור בין אלמנטים שונים במפות ובכך קישור בין- נושאי.

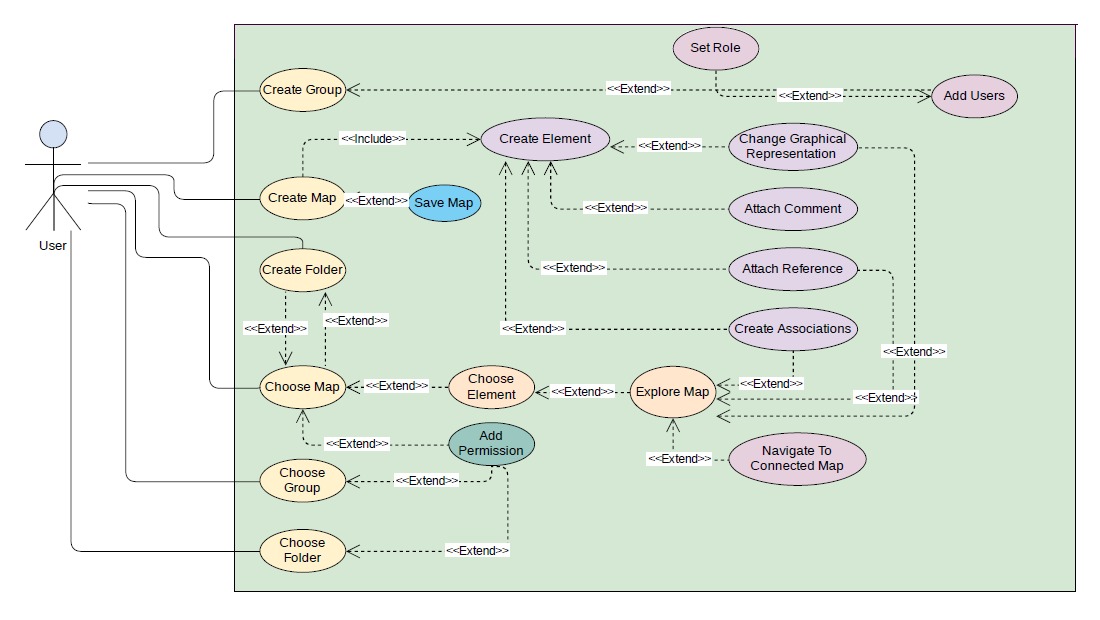
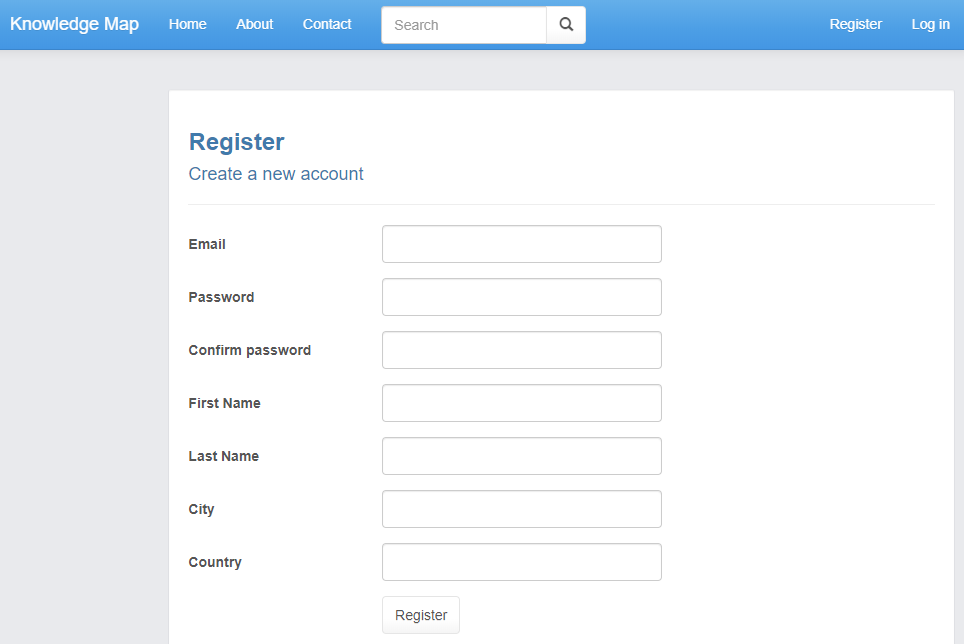
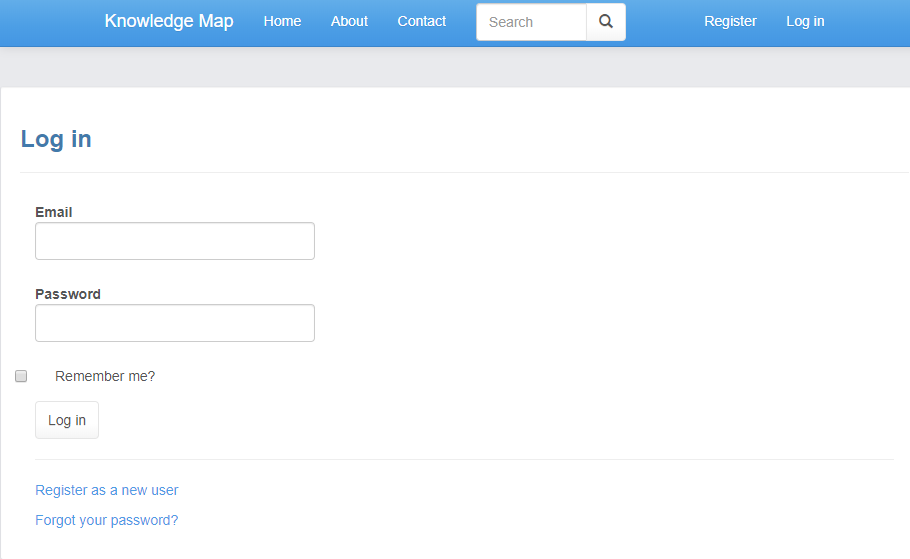
**ניהול משתמשים והרשאות** – כחלק מהמאמץ לשיתוף הידע יתאפשר למשתמש לשתף מפות שהוא יצר ולהעניק הרשאות קריאה/כתיבה למשתמשים אחרים. תתאפשר יצירת קבוצות "לימוד" להן יוענקו הרשאות זהות על מפה או קבוצת מפות. כתוצאה משיתוף המפות יקבלו המשתמשים גם קישור נוח למקורות הידע עצמם הממודל במפות.

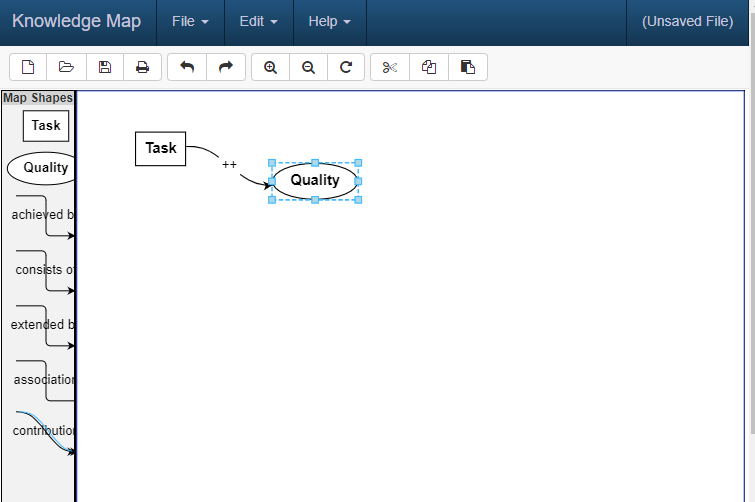
**ניהול דו שיח** – כתיבת הערות על אלמנט מסוים (אם בהגדרת ערך של task או קישור של quality) ובכך ניתן ליצור דיון על אותו פרט ובכך להעשיר את המידע הניתן על האלמנט.

1. **סיכום תהליך חקר הדרישות:**

המערכת שאנו בונים הינה בעבור לקוח (המנחה ארנון שטורם) לכן רוב חקר הדרישות נעשה בראיונות מולו. במהלך כמה פגישות בהן הציג הלקוח את המערכת הרצויה גזרנו ממה שהוא דרש את מסמך הדרישות שנשלח אליו לתיקונים שלו, כך כמה פעמים. בנוסף לפגישות עם הלקוח, סקרנו את המאמר Sturm et al. (2017) בו מתואר המודל של מפות ה- ME, ולפי מודל זה הוספנו דרישות נוספות למערכת.

ישנו אבטיפוס ראשוני למערכת, לאחר סקירה נוספו דרישות נוספות אל מסמך הדרישות.

1. **Usecases:**
   1. יצירת מפה
   2. מתן הרשאה – יצירת תיקייה
   3. חקר תחום בעזרת המפות
2. **דרישות פונקציונאליות:**
   1. ניהול משתמשים:
      1. כל גולש יוכל ליצור לעצמו משתמש באתר, אשר יכיל פרטים מזהים שיסיעו לזהות אותו, ולחפש אחריו במערכות חיפוש משתמשים באתר.
      2. משתמש יוכל לשנות חלק מהפרטים אשר הזין בזמן ההרשמה.
      3. משתמש יוכל למחוק את עצמו מהמערכת.
      4. התחברות לאתר תתבצע באמצעות דוא"ל וסיסמה.
      5. יש אפשרות לשחזר את הסיסמה באמצעות שליחת הסיסמה במייל עמו נרשם המשתמש.
   2. ניהול מפות:
      1. מפה מורכבת מקבוצה של ישויות – Tasks ו-Qualities, וקבוצה של לינקים.
      2. Task –בעיה, או אמצעי להשגת מטרה (כתלות במיקום בהיררכיה). מיוצג גרפית כמרובע.
      3. Quality – מייצג מאפיין רצוי של Task. מיוצג גרפית כאליפסה.
      4. לינק מסוג Consists of מחבר בין Tasks. מיוצג ע"י חץ חד כיווני, עליו הכיתוב Consists of.
      5. לינק מסוג Extended by מחבר בין Tasks. מיוצג ע"י חץ חד כיווני, עליו הכיתוב Extended by.
      6. לינק מסוג Association. מחבר בין Task לQuality. מיוצג ע"י קו, ללא כיתוב.
      7. לינק מסוג Contribution. יכול לצאת מQuality או Task, ותמיד מכוון לQuality. מיוצג ע"י חץ חד כיווני, עם סימני + או - המעידים על תרומת המקור ליעד הלינק.
      8. המשתמש שיצר את המפה מוגדר להיות ה-owner שלה. הוא זה שיכול לתת הרשאות עריכה או קריאה למשתמשים אחרים.
      9. משתמש בעל הרשאות עריכה יוכל לערוך מפה – להוסיף, למחוק או לערוך אלמנטים הנמצאים בה.
      10. כל משתמש בעל הרשאות קריאה, כתיבה או Owner על מפה, יוכל לקרוא ולהשאיר Comments על אלמנטים במפה.
      11. ניתן יהיה לייצא מפה מהאתר למחשב המשתמש בייצוג JSON או כתמונה.
      12. יהיה ניתן לערוך את האלמנטים המרכיבים את המפה באמצעות ממשק גרפי (דומה לdraw io) או בממשק טקסטואלי, המייצג את המפה.
      13. ניתן יהיה לשייך אלמנט במפה למקור ידע – מסמך או כמה מהם
      14. משתמש יוכל להירשם למפה. ההרשמה תביא לכך שהמשתמש יקבל התרעות במייל כאשר המפה אליו נרשם עברה עריכה.
   3. ניהול קבוצות:
      1. קבוצה היא אוסף של משתמשים, המשמש למתן הרשאות על מפות באופן מהיר ויעיל (לעומת מתן הרשאות באופן פרטני לכל משתמש בנפרד).
      2. משתמש שיוצר קבוצה מוגדר כOwner שלה.
      3. Owner יכול להוסיף משתמשים לקבוצה, שיקבלו הרשאות manager או member.
      4. משתמש manager יכול להוסיף members או managers לקבוצה.
   4. ניהול תיקיות:
      1. תיקייה היא אוסף של מפות ותיקיות, אשר משתמש יוצר עבור עצמו בכדי לתת הרשאות למשתמשים או קבוצות באופן יעיל (מתן הרשאות על מספר מפות בפעולה אחת).
   5. ניהול המודל:
      1. ניתן יהיה לערוך את המודל המגדיר את מבנה המפה, ע"י הכנסת link חדש או הכנסת Entity חדשה.
3. **חלוקת פונקציונאליות לחברי צוות:**
   1. ניהול משתמשים + עריכת מפות (סעיף b, תת-סעיפים v ו-vi) – עילי.
   2. ניהול מפות וניהול קבוצות – סער.
   3. ניהול תיקיות וניהול המודל –אורן.
4. **דרישות לא פונקציונליות:**
   1. המערכת תעבוד על כל מערכת הפעלה, ללא צורך בהתקנה.
   2. המערכת תתממשק לשני אלגוריתמים קיימים – אלגוריתם לבדיקת תקינות מפה, ואלגוריתם להפשטה של מפה.
   3. המערכת תתממשק לאלגוריתם לביצוע חיפוש בגרפים אשר נמצא בפיתוח.
5. **תכנון ממשק:**
6. יצירת משתמש חדש:
7. כניסה למערכת :



1. יצירת מפה:
2. **פירוט התוצרים המתוכננים:**

במסגרת הפרוייקט נפתח אפליקציית web אינטראקטיבית, אשר תאפשר למשתמש ליצור מפות, לחקור אותן, ולדון בהן עם קולגות באמצעות מערכת התגובות.

1. **אתגרים וסיכונים:**
   1. בשלב זה לא ברור מהו הפיתרון הנכון עבור בחירת בסיס הנתונים. סוגיה זו נידונה לעומק בסעיף סקירת הטכנולוגיות. בימים הקרובים נקבל החלטה סופית.
   2. התממשקות לאלגוריתם החיפוש – זהו אתגר מפני שהאלגוריתם עצמו אינו כתוב, והצוות בשלב זה לא יודע לומר איזה סוגים של שאילתות ייצטרך. דרישות אלה של הצוות כמובן משליכות על הבחירות הטכנולוגיות שלנו ועל הAPI שנדרש לפתח.
2. **כלים ושיטות:**
   1. עבור הfrontend בחרנו בAngular.
   2. עבור ממשק העריכה הגרפי של המפות, נבחרה ספריית GoJS.
   3. עבור הbackend בחרנו בNodeJS.
   4. עבור בסיס הנתונים טרם התקבלה החלטה סופית.
3. **לו"ז מעודכן עבור הפרויקט:**
   1. דצמבר:
      1. בחירת DB.
      2. הקמת DB עם מפות קיימות.
      3. תמיכה של backend וfrontend בהצגת מפות.
      4. נקודת ביקורת עם צוות אלגוריתם החיפוש.
   2. ינואר:
      1. עריכת מפות ושמירה.
      2. משתמשים והרשאות.
      3. נקודת ביקורת עם צוות אלגוריתם החיפוש.
   3. פברואר:
      1. תקופת מבחנים
   4. מרץ:
      1. תיקיות וקבוצות.
      2. נקודת ביקורת עם צוות אלגוריתם החיפוש.
   5. אפריל:
      1. תמיכה בשינוי המודל.
      2. אינטגרציה מלאה עם אלגוריתם החיפוש.
   6. מאי:
      1. ספר פרויקט.