**WeWatch**



**אלון צדקני – 326281367**

**שם בית ספר – הרב תחומי ב' , פתח תקווה**

**מנחה – גולן מור**

**תאריך הגשה – 11.5.2022**

****

**תוכן עניינים**

**מבוא**

**תקציר ורציונל הפרויקט**

המוצר הינו תוסף בדפדפן הכרום המאפשר צפייה משותפת למספר משתתפים דרך אתר יוטיוב עצמו. המשתתפים יכולים לפתוח חדרי צפייה על כל סרטון יוטיוב הקיים באתר ובאמצעות שיתוף המספר הסידורי של החדר וסיסמתו כל משתמש המזין את הפרטים הנ"ל יכול להצטרף לחדר הצפייה ולחוות צפייה משותפת ומסונכרנת.

**בחרתי לעשות את פרויקט זה מכמה סיבות:**

בחרתי בפרויקט זה מכיוון שפרויקט העוסק בצפייה משותפת הינו נושא שמעניין ומושך אותי. כנער שבזמנו הפנוי מבלה במחשב יצא לי לצפות עם חברים בסרטונים משותפים במספר דרכים שונות כמו שיתוף מסך, ופתיחת אתר חיצוני המריץ סרטון במקביל לא יצא לי להיתקל בתוסף לכרום המאפשר למשתמשים לחוות בצפייה משותפת. עקב עובדה זאת חשבתי שפרויקט הגמר יכולה להיות הזדמנות טובה לנסות לכתוב תוסף כזה בעצמי וכך חשבתי על הרעיון לפרויקט.

קהל היעד של התוסף הינו כל משתמש יוטיוב שרוצה לצפות בתכנים בצורה משותפת עם עוד אנשים ללא צורך בשיתוף מסך או שימוש באתר חיצוני.

המטרות המרכזיות של התוסף שאותו בחרתי לבנות הינם: (1) המשתמש יוכל לגשת לכל סרטון הקיים באתר ולהפוך אותו בלחיצה לחדר צפייה משותף. (2) תתקיים צפייה מסונכרנת וחלקה בין חברי חדר הצפייה.

בפרויקט זה אני עובר עם ה-API של chrome extension ולכן אני מוגבל לכתיבת דברים ופונקציות המקובלות על אותו API , בנוסף לכך מכיוון שה-API של תוספי כרום פועל לפי EDA   
(event driven architecture) אז הייתי נדרש לכתוב את כל הקוד לפיכך והייתי צריך לעמוד באותם תנאים ארכיטקטורה זאת מציבה.

**מבוא ורקע כללי**

הפרויקט פועל כתוסף כרום, משמע אני פועל מתוך ה-API של chrome extensions שעובד סביב EDA (event driven architecture). התוסף בנוי מכמה חלקים שונים: (1) ***manifest.json***שאחראי להצגת ה-API של תוספי כרום על מבנה האפליקציה והשימושים המרכזיים.  
(2) ***background.js*** שמתקשר עם השרת ,מחזיק מידע על המשתמש ושם מתופעל כל האירועים האפשריים הנכללים באפליקציה בזמן גלישת המשתמש בכרום. (3) ***content.js*** שאחראי על תפעול החדר הצפייה, ובעצם יושב על ה-html של יוטיוב על מנת להוציא מידע על תפעול הסרטון ע"י המשתמש, גילוי וחסימת פרסומות המפריעים למהלך הצפייה ובנוסף לכך יוצר תקשורת מלאה עם ה-background.js ומעדכן אותו על מנת שהמידע הנחוץ יועבר לשאר חברי חדר הצפייה ע"י השרת. (4) ***popups.html*** למיניהם שאחראים על ייצוג מידע נחוץ למשתמש ונתינת אופציות למשתמש להצטרף לחדר צפייה, ליצור חדר צפייה חדש ולהתעדכן במצב חדר הצפייה בין אם זה לראות כמה צופים יש בחדר, ומה פרטי החדר (ID וסיסמא). (5) ***popups.js*** שאחראים על קישור בין המידע המוצג ב-html למידע הקשור למשתמש הנמצא ב-*background.js*.

השרת פועל בעזרת WebSocket וכך מתקשר עם הלקוחות. הוא פועל באמצעות הרצה א-סינכרונית של פונקציה המאזינה להודעות מהלקוחות ופועלת בהתאם, בין אם זה לרשום משתמש חדש למערכת, ליצור חדר צפייה חדש או לעדכן צופים על פקודות הקורות במערכת. השרת מעניק לכל משתמש חדש שמוריד את התוסף ID ייחודי ו-username משלו ורושמת את הפרטים שלו ב-dictionaries שונים. לאחר מכן באמצעות אותו ID ייחודי הוא יכול לנהל את כל המשתמשים בתוסף ולקשר את כל המידע הנחוץ באמצעות כלפי כל משתמש לפי אותם dictionaries.

**תיאור המערכת**

המערכת תעקוב אחרי הגלישה של המשתמש ותבדוק במידה והוא צופה בסרטון יוטיוב למשתמש יינתן אופציה באמצעות פופ-אפ לפתוח חדר צפייה על סרטון זה. כאשר המשתמש פותח חדר צפייה חלון הפופ -אפ ישתנה ויכיל את המספר הסידורי של החדר וסיסמתו ובנוסף כפותח החדר יינתן לו זכויות ה-Host משמע לו יינתן ההרשאות לשליטה בסרטון, להעביר זמן ולהפעיל/לעצור את הסרטון.

אם משתמש רוצה להצטרף לחדר צפייה עליו לפתוח את הפופ-אפ של התוסף ולהזין את המספר הסידורי ואת הסיסמא, במידה והפרטים נכונים יפתח לו סרטון היוטיוב בחדר הצפייה ולו כמשתמש לא תהיה אפשרות לשלוט בסרטון, רק ה-Host ראשי להריץ את הסרטון.

**דרישות פונקציונאליות:**

**1.** התוכנה תתחבר לתקשורת רציפה עם הסרבר הקיים ותקנה לכל משתמש ID ייחודי.

**2.** התוכנה תזהה אם המשתמש צופה בסרטון יוטיוב ולפיו תעניק לו את ההזמנות לפתוח חדר צפייה בעל שם וסיסמא.

**3.** יתאפשר למשתמשים להצטרף לחדרי צפייה קיימים ע"י הזנת השם וסיסמתם.

**4.** תתקיים הרצפה של פעולות בצורה מסונכרנת באמצעות ה-UTC.

**דרישות לא פונקציונאליות:**

**1.** הסרטון המשותף ירוץ בצורה חלקה ומתאומת אצל כל צופי החדר.

**2.** דפי ה-html שבפרויקט יהיו אסטטיים וקלים לתפעול.

**תרחישי שימוש**

**1.** כאשר מותקן התוכנה לראשונה:

**-** ייפתח דף הסבר על השימוש באפליקציה

**-** המשתמש יקבל ID ו-username ייחודי ע"י השרת.

**2.** כאשר המשתמש בסרטון יוטיוב ורוצה לצפות בצורה משותפת עם חבריו:

**-** יפתח לו האופציה לפתיחת חדר צפייה, בעל ID ייחודי וסיסמא.

**-** המשתמש יוכל לשתף את ה-ID והסיסמא לחבריו על מנת שיוכלו להצטרף לחדר הצפייה.

**3.** כאשר משתמש רוצה להצטרף לחדר צפייה משותפת:

**-** יינתן לו האופציה להכניס שם משתמש וסיסמא,

**->** במידה והסיסמא נכונה יפתח למש הוא יצטרף לחדר הצפייה.

**->** במידה והפרטים לא נכונים, המערכת תודיע על כך למשתמש ויינתן לו הזדמנות נוספת.

**לו"ז לפיתוח המערכת**

* 30.2 – מחקר על פיתוח תוספי כרום והבנה כללית של הארכיטקטורה של EDA בפיתוח התוסף, בניית "מיני פרויקטים" להבנה רחבה יותר.
* 1.3 – כתיבת תקשורת בסיסית בין גוגל (client) לסרבר.
* 5.3 – לכתוב את כל הפונקציות - event based של התוסף במצבי הגלישה של המשתמש.
* 25.3 – כתיבת ה-content script שמסנכרן את סרטון היוטיוב על פני כל המשתתפים.
* 1.4 – בניית GUI לכל חלונות התוסף.
* 1.5 – אופטימיזציה של כל הקוד הבנוי ותיקוני באגים ופרטים קטנים.

**ניהול סיכונים**

הסיכונים המרכזיים שלי במהלך הפרויקט זה להתמודד וללמוד את כל המידע החדש שנכלל בהכנת הפרויקט הזה. התקשורת עוברת באמצעות WebSocket שפועל בצורה ואופי שונה מ - socket רגיל שהתרגלתי לעבוד איתו בפייטון. בנוסף הקושי להסתגל לעבודה סביב API ובנוסף כתיבת פרויקט ש-70 אחוז ממנו בשפה שחדשה לי (JavaScript) עקב סיכונים אלו החלטתי לפני ההתחלה של כתיבת הפרויקט עצמו לעשות מספר מיני פרויקטים שיעזרו לי להבין את כל החומר החדש ותכנני את הלוז בכל שקודם אעסוק בכל הנושאים שאני פחות מנוסה בהם ורק אחר כך אחבר בין כל הנקודות לבניית פרויקט מוגמר.

**תיאור תחום הידע - פרק מילולי**

**פירוט יכולות בצד השרת**

* הרשמה למערכת - משתמש חדש

רישום משתמש חדש לשרת המוריד את התוסף לראשונה בכרום.

* קליטת התחברות WebSocket חדש לשרת.
* נתינה ללקוח ID ייחודי.
* נתינה ללקוח username ייחודי שישמש אותו לאורך כל השימוש ורישומו במערכת.
* רישום פרטי ה-ID ,ה-username וה-WebSocket הנוכחי שבאמצעותו אותו לקוח תקשר עם הסרבר.
* אובייקטים נחוצים: תקשורת, מבנה נתונים
* התחברות מחדש של לקוח לשרת

במידה וה-WebSocket של הלקוח התנתק והלקוח רוצה לשלוח הודעה חדשה לשרת.

* נוצר בדיקה של ה-ID שאליו משתייך הלקוח ובמידה והוא קיים רושם את הלקוח מחדש במערכת עם ה-WebSocket החדש שדרכו התחבר.
* במידה ואין את אותו ה-ID המדובר השרת מעניק ללקוח ID ו-username חדש ורושם אותו במערכת.
* אובייקטים נחוצים: תקשורת, מבנה נתונים
* פתיחת חדר צפייה חדש

במידה והלקוח רוצה לפתוח סרבר הוא פונה לשרת על מנת לפתוח ולרשום את החדר.

* יצירת ID וסיסמא לחדר.
* רישום פרטי החדר במערכת: ID, סיסמא, משתתפים, URL של הסרטון, הזמן האחרון שה-host צופה הפיץ לשרת פקודה.
* שליחת ה-ID והסיסמא ללקוח.
* אובייקטים נחוצים: תקשורת, מבנה נתונים
* הצטרפות לקוח לחדר חדש

המידה ולקוח רוצה להצטרף לחדר צפייה הוא שולח את פרטי החדר לשרת.

* אימות פרטי החדר שהלקוח הכניס בפופ-אפ, במידה והפרטים נכונים, מוסיף אותו לחדר במערכת ובמידה ולא שולח ללקוח שהפרטים לא נכונים.
* מודיע לשאר חברי החדר שמשתמש חדש הצטרף אליו.
* אובייקטים נחוצים: תקשורת, מבנה נתונים
* יציאה של לקוח מהחדר

במידה והלקוח יוצא מהחדר הוא מודיע על כך לשרת.

* הורדה של הלקוח מהמערכת ברשימת הצופים בחדר הצפייה.
* העברת המסר על יציאת הלקוח לשאר המשתתפים בחדר.
* אובייקטים נחוצים: תקשורת, מבנה נתונים
* יציאה של לקוח שהוא ה-Host של חדר הצפייה

במידה וה-Host של חדר הצפייה יוצא הוא מודיע על כך לשרת.

* הורדה של הלקוח מהמערכת ברשימת הצופים בחדר הצפייה.
* העברת המסר על יציאת הלקוח לשאר המשתתפים בחדר.
* העברת הרשאות ה-Host ללקוח הצטרף לחדר אחריו.
* אובייקטים נחוצים: תקשורת, מבנה נתונים
* סגירת חדר צפייה

במידה ויוצא הלקוח האחרון מחדר הצפייה.

* השרת מוודא שאכן אף אחד לא שנשאר בחדר הצפייה.
* מחיקת חדר הצפייה מהמערכת.
* אובייקטים נחוצים: תקשורת, מבנה נתונים
* העברת broadcast לחברי החדר

במידה והלקוח עושה פעולה על הסרטון (מפעיל, עוצר, מעביר זמן) הוא מודיע על כך לשרת.

* יצירת זמן UTC שבא כל הלקוחות יבצעו את הפעולה שה-Host יזם.
* העברת מסר באמצעות שליחת הפקודה לWebSocket הרשומים כלקוחות בחדר הצפייה.
* אובייקטים נחוצים: תקשורת

**פירוט יכולות בצד הלקוח**

* התחברות לשרת ולמערכת

קבלת ID מהשרת להצטרפות הלקוח החדש

* בדיקת ה-Event שבא הלקוח הוריד את התוסף לראשונה.
* יצירת WebSocket חדש וחיברו לשרת.
* קבלת ה-ID וה-username מהשרת ורישומם כמשתנים גלובליים.
* אובייקטים נחוצים: תקשורת
* התחברות מחדש לשרת

במידה וה-WebSocket של הלקוח מתנתק.

* יצירת WebSocket חדש ושליחת בקשה להתחברות מחדש באמצעות ה-ID הקיים.
* קבלת ההודעה מהשרת וקבלת ID ו-username חדש במידת הצורך או קבלת אישור להתחברות מחדש למערכת.
* אובייקטים נחוצים: תקשורת
* פתיחת חדר צפייה חדש

במידה והלקוח נמצא בסרטון יוטיוב ומעוניין לפתוח חדר צפייה

* הופעת פופ-אפ המציע למשתמש לפתוח חדר צפייה.
* במידה והמשתמש מעוניין ולוחץ על כפתור הפתיחה נשלח הודעה לשרת עם ה-URL של הסרטון ובקשה לפתיחת חדר צפייה.
* נפתח למשתמש פופ-אפ חדש שמכיל את ה-ID של החדר, סיסמתו ורשימת המשתתפים הקיימים בחדר.
* נפתח למשתמש הלקוח הראשות Host בחדר הצפייה.
* אובייקטים נחוצים: תקשורת, ממשק משתמש
* שימוש בחדר צפייה כ-Host

במידה והלקוח הינו Host בחדר הצפייה הנוכחי

* שליטה מלאה על זמן הצפייה.
* שליטה מלאה על כפתור ההפעלה.
* במידה והמשתמש סוגר את החלון או עובר על לאתר אחר הוא מוצא אוטומטית מן החדר.
* אובייקטים נחוצים: web scraping
* שליחת פקודות של ה-Host לשרת

כאשר המשתמש עושה פקודה על הסרטון באתר היוטיוב שלו.

* באמצעות content script שהוזרק לתוך האתר נאסף מידע על הפקודות שמשתמש מבצע.
* ה-script מודיע ללקוח על הפקודה ולאחר מכן היא מועברת לשרת על מנת לבצע broadcast לכך לשאר צופי החדר.
* אובייקטים נחוצים: web scraping, תקשורת
* שימוש בחדר צפייה כלא Host

במידה והלקוח אינו Host בחדר הצפייה הנוכחי.

* אין שליטה כלל על אף פרט בסרטון חוץ מהווליום – כל כפתורי התפעול נעלמים מן המסך.
* במידה והמשתמש סוגר את החלון או עובר על לאתר אחר הוא מוצא אוטומטית מן החדר.
* אובייקטים נחוצים: web scraping
* הצטרפות לקוח לחדר חדש

במידה ומשתמש חדש הצטרף לאותו חדר הצפייה של הלקוח

* קבלת הודעה מהשרת דרך ה-WebSocket של הלקוח על הצטרפות משתמש חדש לחדר.
* עדכון הפופ-אפ והוספת המשתמש החדש לרשימת הצופים בחדר.
* אובייקטים נחוצים: ממשק משתמש, תקשורת
* יציאה של לקוח מהחדר

במידה ומשתמש יוצא מחדר הצפייה של הלקוח

* קבלת הודעה מהשרת דרך ה-WebSocket של הלקוח על יציאה של משתמש מהחדר.
* עדכון הפופ-אפ והסרת המשתמש מרשימת הצופים בחדר.
* אובייקטים נחוצים: תקשורת
* חסימת פרסומות

במידה ובמהלך הסרטון מופיע פרסומת שעוצרת את הסרטון.

* קליטת שינוי באובייקטים באתר.
* מציאת אובייקט ה-DOM של הפרסומת.
* דילוג ישיר על הפרסומת בחזרה לסרטון.
* אובייקטים נחוצים: web scraping