WeWatch

אלון צדקני – 326281367

שם בית ספר – הרב תחומי ב' , פתח תקווה

מנחה – גולן מור

תאריך הגשה – 16.5.2022

תוכן עניינים

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.

בנוסף לכך רציתי לעסוק ולנצל את הזמן של עבודה על פרויקט גמר בכך שאבחר נושא שירחיב את אופקיי מעבר לנלמד בבית הספר במהלך השנים שלי במגמה ואכן יצירת תוסף כרום זה פרויקט שרחוק מאוד מאזור הנוחות שלי וכולו עוסק ב-HTML,JavaScript ו-CSS שהם דברים שפחות יצא לי להתנסות בהם אם בכלל. ראיתי הזדמנות בבחירת פרויקט זה כניסיון בתחום תוכנה חדש ובשילוב עם נושא שמעניין אותי ויש לי מוטיבציה לעבוד עליו בחריצות.

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*.

באמצעות כל חלקים אלו הלקוח במערכת יוכל לעקוב אחרי סטטוס המשתמש בגלישתו בכרום – באמצעות events הקיימים ב-API של תוספי כרום וכך תוכל להנגיש למשתמש בהתאם אופציות לתפעול האפליקציה דרך פופ -אפים שמשתנים לפי ה-URL של המשתמש ויעניקו לו אפשרויות בין אם זה לפתוח חדר חדש או להצטרף לחדר הקיים כבר במערכת. לאחר שהמשתמש בחדר צפייה "יוזרק" לתוך עמוד סרטון היוטיוב קוד שעוקב אחרי פעולות המשתמש ומעדכן בהתאם את מצבו בסרטון כמבוקש. בנוסף לכך קוד זה יאתר פרסומות המפריעות לחוויית הצפייה ותחסום אותם אוטומטית.

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.

מטרת הפרויקט

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. המערכת תעקוב אחרי פעילויות המשתמשים ותעדכן את השרת במידה ויוצא הלקוח מחדר הצפייה באמצעות סגירת החלון או מעבר ל-URL אחר.

דרישות מרכזיות

להלן מספר דרישות פונקציונליות ולא פונקציונליות על מנת למקד את עבודתי בפרויקט ויצירת רשימה של נקודות שעליהם אתייחס במיקוד ואדאג בפרט שהם יתקיימו בתוצר הסופי בצורה האפשרית והטובה ביותר.

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 סנכרון בסרטוני היוטיוב בין צופי החדר באמצעות הרצה משותפת של פעולות על אלמנט הסרטון באתר.

דרישות לא פונקציונאליות:

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

2. דפי ה-html שבפרויקט יהיו אסטטיים ,קלים לתפעול ומובנים למשתמש.

3. יתאפשר למשתמש חווית צפייה משתופת בצורה שתרגיש למשתמש נוחה ולא דורשת מאמץ או הכנה מראש.

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 למשתמש חדר הצפייה באתר יוטיוב ושאר הצופים יעודכנו על הצטרפותו באמצעות שינוי רשימת הצופים.

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

4. כאשר המשתמש רוצה לצאת מחדר הצפייה:

- כל שאר חברי חדר הצפייה יעודכנו על יציאת המשתמש באמצעות שינוי רשימת הצופים.

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 ובעל אופי שונה מ - socket רגיל שהתרגלתי לעבוד איתו בפייטון. גם כן קיים הקושי להסתגל לעבודה סביב API חדש בעל תנאים ודרכי עבודה ייחודיים משלו (ה-API של תוספי כרום). בנוסף לכך קיים גם הקושי לכתיבת 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 ותכננתי את הלוז כך שקודם אעסוק ואנסה להתמודד עם החלקים הקריטיים בפרויקט שבהם אני חדש לחלוטין וחסר ניסיון ורק לאחר שאצליח להתמודד עם כל הנקודות בהם רציתי להתמקד ולהבין כיצד לכתוב אותם בסביבת העבודה החדשה שלי רק אחר כך אחבר בין כל הנקודות והקוד שכתבתי לבניית פרויקט מוגמר.

תיאור תחום הידע

להלן פירוט היכולות של הפרויקט הן בצד השרת והן בצד הלקוח.

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 מהוליום – 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

שפת התכנות וסביבת העבודה

במהלך כתיבת הפרויקט שלי עסקתי בכמה שפות תוכנה שונות למספר שימושים שונים. להלן כל שפה רשימת השפות שהשתמשתי בהן, השימוש שנעשה בהן ומדוע בחרתי בשפה זו.

(1) **פייטון – Python**, בשפה זו השתמשתי אך ורק ליצירת השרת בפרויקט. השרת מנהל את כל הלקוחות במערכת ,את מצבי החדרי הצפייה ומתקשר עם ה-backend של תוסף הכרום. בחרתי בשפה זאתי לשרת מכיוון שיש לי ניסיון רב בשפה ככלל ובפרט בבניית שרתים. עקב סיבה זאת מצאת לנכון לכתוב את השרת בשפה שבה אוכל לכתוב בצורה הכי טובה וארגיש בנוח איתה לאורך כל הפרויקט.

(2) **HTML**, בשפה זו השתמשתי על מנת לכתוב את העמוד פתיחה שנפתח ברגע שהמשתמש מוריד לראשונה את התוסף ואת הפופ-אפים המופיעים למשתמש במהלך השימוש בתוסף. בחרתי בשפה זו כי היא השפה ה-default לכתיבת אתרים וקלה מאוד ללמידה ויישום בתוך הפרויקט שלי.

(3) **ג'אווה סקריפט – JavaScript**, בשפה זו השתמשי ברוב כתיבת הקוד של התוסף בין אם זה ב-backend שלה או ב-frontend. בנוסף לכך השתמשתי בה בשביל לכתוב סקריפטים של אתרי ה-HTML המופיעים בתוסף. בחרתי בשפה זו מכיוון שהיא השפה ה-default לכתיבת תוספי כרום והיא נועדה לעבודה עם html ואתרי אינטרנט ומכיוון שכל הפרויקט שלי עוסק בעבודה סביב אתר היוטיוב כתיבת הקוד ב-JS הינה הפתרון ההגיוני והטוב ביותר.

(4) **CSS**, בשפה זו השתמשתי על מנת לעצב את עמודי ה-HTML שנמצאים בפרויקט שלי. בחרתי בשפה זו מכיוון שהיא עובדת בצורה טובה עם HTML , היא שפה קלה ללמידה ולתפעול ויכולה להעניק לעיצוב הבנאלי של כתיבת HTML עיצוב ייחודי שנותנת הרגשה של עמוד יותר מוגמר ונעים לעין על מנת לשפר את חווית המשתמש.

(5) **JSON**, בשפה זו השתמשתי על מנת לכתוב את ה-manifest של תוסף הכרום. ה-manifest מציג ל-API של תוספי כרום את הסקריפטים הקיימים בתוסף, ההרשאות המיוחדות שהתוסף דורש, תמונות, כותרות, פופ-אפים וכדומה. בחרתי בשפה מכיוון שעל פי הנחיות ה-API של תוספי כרום זה השפה הדרושה לכתיבת ה-manifest של תוספי כרום.

חלוקת השפות בפרויקט שלי הינה:

CSS - **5%** , *HTML* - **8%** , *Python* - **17%** , *JavaScript* - **70%**

ניסוח וניתוח הבעיה האלגוריתמית

במהלך הפרויקט נתקלתי במספר בעיות אלגוריתמיות שלאחר מכן ביססו לי את הפרויקט ובנו אותו לצורה שבה שהוא מוצג בתוצר הסופי.

(1) כיצד ניתן לצרוך סרטון משותף על האינטרנט?

- כלומר מה הדרך המועדפת לשיתוף סרטון בתוסף על מנת לקבל חוויית צפייה משותפת נוחה, מסונכרנת ומהירה.

(2) איך למנוע את הפרסומות בסרטון היוטיוב?

- במהלך הסרטון יכולים להופיע פרסומות העוצרות את סרטון היוטיוב ומנעות את הסנכרון בין צופי החדר.

(3) איך לשמור את המידע על המשתמש בתוסף?

- ישנה בעיה ב-API של התוסף בכך שה-backend של מאתחל את עצמו מחדש כל פעם שהמשתמש לא נוגע בכרום למספר שניות על מנת לשמור על ביצועי המחשב, כך כל פעם מחדש פרטי המשתמש מאתחלים ולא נשמרים במערכת.

(4) כיצד לשמור על קשר בין התוסף לשרת?

-מכיוון שהתוסף בנוי על JavaScript ופועל בעצם על כרום אז התקשורת הבנאלית שמתאפשר דרך שימוש בתוסף הינו רק דרך WebSocket. הבעיה בשימוש תקשורת זאתי היא שהיא אינה תקשורת רציפה ולאחר כמה שניות שה-socket אינו בשימוש הוא סוגר את עצמו ואז כאשר אני רוצה לשלוח הודעה לשרת מהלקוח אך ה-socket סגור ההודעה לא תשלח והתקשורת הרציפה תיעצר. דבר נוסף לבעיה היא שכאשר הלקוח בחדר הצפייה והשרת רוצה להעביר פקודה ללקוח וה-socket שלו נסגר אז ההודעה לא תועבר וייווצר תקל בפעילות חדר הצפייה.

תיאור אלגוריתמים קיימים

(1) כיצד ניתן לצרוך סרטון משותף על האינטרנט?

- הפתרון הראשון שעלה לראשי זה להוריד את סרטון היוטיוב ל-mp4 וליצור אתר נפרד יוטיוב באינטרנט שרץ על שרת בעל חדרים והלקוחות שרוצים לצפות בסרטון יפתחו את חדר הצפייה ושם "יוקרן" להם סרטון היוטיוב שהשרת הוריד לזיכרון שלו.

- הפתרון השני שחשבתי עליו זה לעבוד עם embedded video player משמע גם כן ליצור אתר שהשרת מתפעל אך במקום להוריד את הסרטון לזיכרון השרת הוא משתמש ב-URL מיוחד של סרטון יוטיוב המאפשר באמצעות HTML להציג את הסרטון באתר נפרד בשליטת השרת וכך ניתן להריץ את הסרטון בצורה משותפת בצורה דומה ואף יותר פשוטה מהפתרון הראשון.

- הפתרון השלישי זה להריץ את הסרטון בצורה משותפת על אתר יוטיוב עצמו. משמע לשלוט את כל לקוח דרך התוסף בהתקדמות הסרטון וכך להריץ בצורה משותפת לכל הלקוחות שבחדר הצפייה. כך שכל לקוח באתר יוטיוב בסרטון מעצמו אך הצפייה נעשית בצורה משותפת לחלוטין.

(2) איך למנוע את הפרסומות בסרטון היוטיוב?

- הפתרון הראשון שחשבתי עליו הינו פשוט להוריד את הסרטון ל-mp4 וכך למנוע לחלוטין הופעת פרסומות בסרטון היוטיוב – פתרון זה הולך יד ביד עם הפתרון הראשון בבעיה הראשונה.

- הפתרון השני הינו לדרוש מהלקוח להוריד תוסף החוסם פרסומות ביוטיוב, קיימים עשרות אופציות לחוסמי פרסומות באינטרנט ככלל ביוטיוב בפרט ולכן פתרון זה ניתן ליישום בקלות ופותר את הבעיה בצורה מלאה.

- הפתרון השלישי זה ליצור בעצמי חוסם פרסומות ליוטיוב, כך שכל פרסומת המונעת התקדמות בסרטון ייחסמו אוטומטית.

(3) איך לשמור את המידע על המשתמש בתוסף?

- הפתרון הראשון זה לשמור את כל המידע על האובייקטים בקובץ נפרד וכל פעם שה-backend חוזר לתפעול הוא קורא ממנו את המידע על המשתמש וכך מעדכן את המערכת. פתרון זה קל ליישום ופשוט.

- הפתרון השני זה לעבוד בסביבת עבודה של chrome.storgae.local ודרכו לשמור את כל המידע על המשתמש. סביבת עבודה זו היא דרך ה-API של תוספי כרום כך שהיא דורשת מאמץ למידה ויישום בתוך הפרויקט.

(4) כיצד לשמור על קשר בין התוסף לשרת?

- הפתרון הראשון שחשבתי עליו הינו לשלוח הודעות מהלקוח לשרת כל כמה שניות על מנת להשאיר את הקשר פעיל וכך ה-WebSocket של הלקוח לא ייסגר אף פעם לאורך פעילותו עם השרת.

- הפתרון השני הינו לעשות בדיקה לפני כל שליחת הודעה לשרת על מצב ה-WebSocket של הלקוח ובמידה והוא נסגר אז לחדש את הקשר עם השרת לטווח פעילות נוסף עד שדבר זה קורה שוב.

הפתרונות הנבחרים

(1) כיצד ניתן לצרוך סרטון משותף על האינטרנט?

החלטתי לבחור בפתרון השלישי שחשבתי עליו שהוא עבודה דרך אתר יוטיוב עצמו. כלומר בחרתי שהדרך המועדפת עלי בפרויקט לצרוך סרטון בצורה משותפת הינה דרך מניפולציה של הסרטון באתר יוטיוב בצורה מסונכרנת בין צופי החדר. בחרתי דווקא בפתרון זה מכמה סיבות:

- הסיבה ראשונה לכך היא שרציתי לתת לצופה חווית צפייה משותפת פשוטה וקלה לתפעול במלי שהמשתמש ירגיש שהוא צריך להתאמץ יותר מידי בשביל לחוות סרטון עם חבריו. יצירת אתר חיצוני שדרכו הם יצפו בסרטון הרגיש לי שנדרש למשתמש יותר מאמץ לתפעול ולפתיחת חדר צפייה ובנוסף לכך הייתי מודע למספר אתרים שמעניקים את שירות זה. רציתי לצאת מחוץ לקופסא ולחשוב על פתרון ייחודי ויצירתי לבעיה.

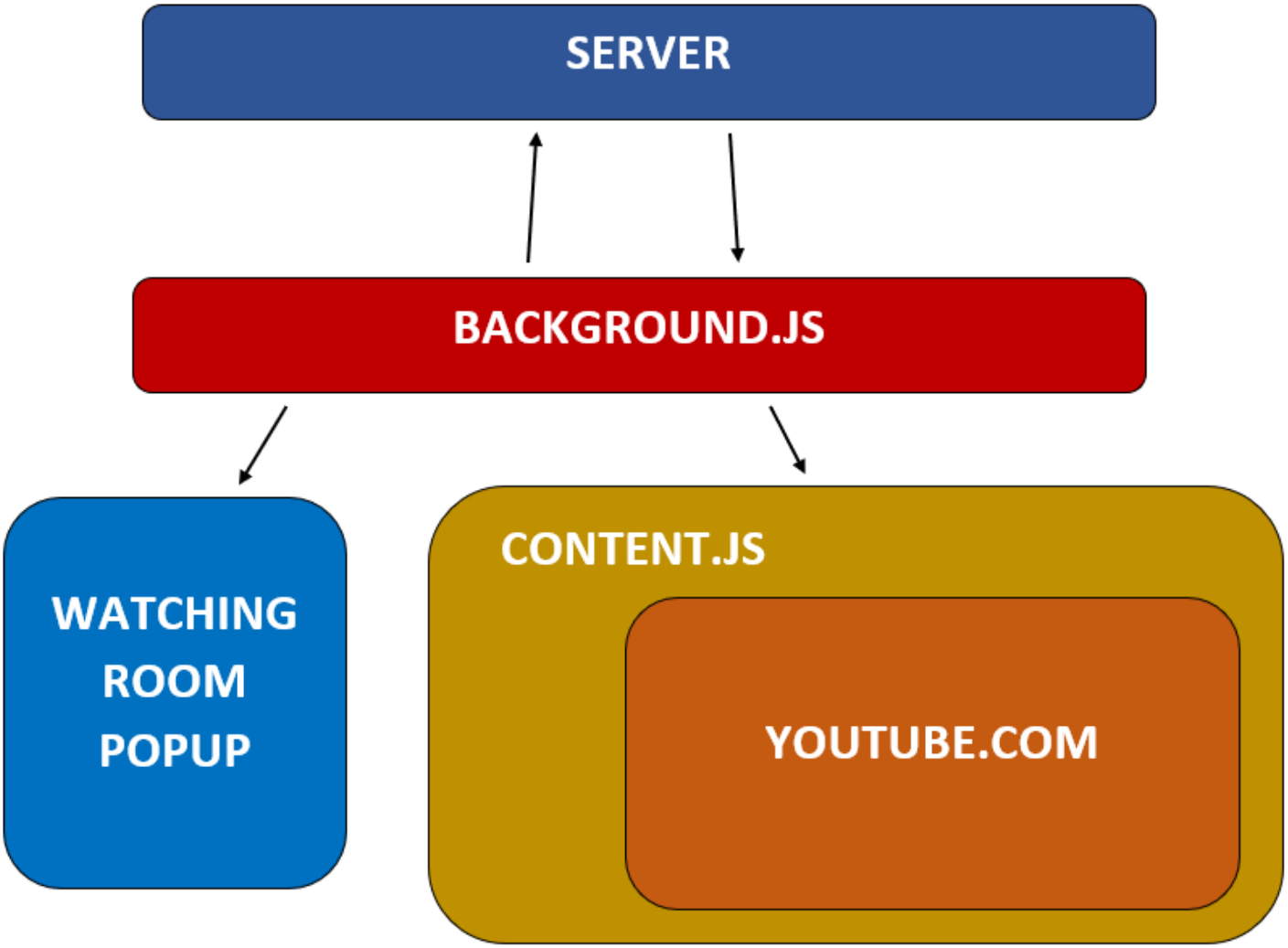
- הסיבה השנייה לכך היא שרציתי לנצל את שימושי בתוסף כרום בצורה המלאה ביותר. מכיוון שהחלטתי כבר שאני עוסק בסביבת העבודה של תוספי כרום רציתי לנצל את שימושי בתוסף בצורה המלאה ביותר. לאחר מחקר לא קצר גיליתי שבעזרת התוסף אני יכול לשלוט על אובייקטים באתרים חיצוניים ולבצע עליהם מניפולציות כך שאוכל לתת למשתמש חוויית צפייה משותפת נקייה דרך אתר יוטיוב עצמו. מכיוון שפתרון זה עוסק בסביבת העבודה של תוסף כרום רציתי לנצל כבר את ההזדמנות לכך שאני עובד על תוסף ולנצל את יכולותיו על כרום בצורה המקסימלית.

מבנה המערכת:

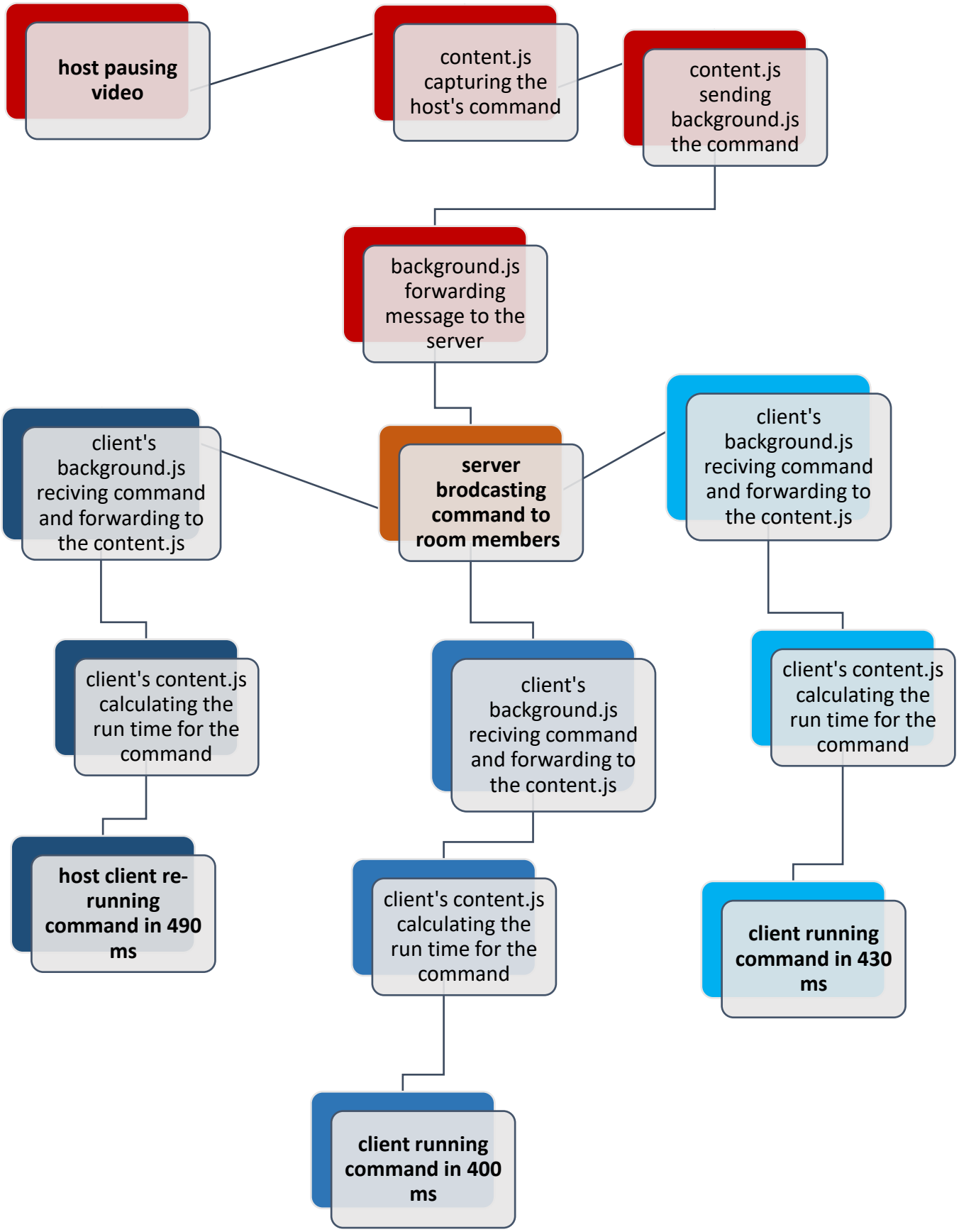
אפשרתי את חווית הצפייה המשותפת דרך יוטיוב בכך שבניתי מספר פונקציות הבנויות על events ב-backend של תוסף הכרום כך שברגע שאותר שהמשתמש בחדר צפייה הוא "מזריק" את ה-content.js לתוך אתר היוטיוב ובעזרת התקשורת בין ה-background.js וה-content.js ניתן ליצור חווית צפייה מסונכרנת בין כל המשתתפים. ה-background.js מקבל הודעות לגבי פעולות שצריך לבצע על הסרטון מהשרת ולאחר סינון הודעות שחוסך עומס על אתר היוטיוב הוא מעביר את זה הלאה ל-content.js שמבצע את הפעולות הללו על אובייקט הסרטון ש"לקח" מהאתר. מכיוון שכאשר שמזריקים קוד לתוך האתר בצורה זו יש לך גישה לכל התוכן באתר, עקב כך יכולתי להשיג את האלמנטים הרצויים שכוללים את הסרטון עצמו וכפתורי ההפעלה של האתר ובעזרתם יכולתי להריץ בקוד events שעוקבים אחרי פעולות שהמשתמש עושה על הסרטון, בין אם זה לעצור, להפעיל או להעביר את הזמן. לאחר שנקלט אחד מן הפעולות הללו במידה והלקוח הינו ה-host של חדר הצפייה, רק למשתמש שיצר את החדר יש שליטה מלאה על תפעול הסרטון. במידה והמשתמש אינו host מתבטל לו האפשרות לשלוט על הסרטון בכך שנעלם כפתורי ההפעלה והאופציה ללחוץ על הסרטון.

מתאפשר חווית צפייה משותפת בכך שכל המחשבים מבצעים את הפעולות בו זמנית. אני מוודה שאכן כל המחשבים מריצים את הפעולה על הסרטון בו זמנית בכך שהשרת שולח לכל המחשבים את הפקודה שעליהם לעשות על הסרטון ובנוסף לכך נותן להם את ה-UTC (Coordinated Universal Time). לאחר מכן שההודעה מועברת ל-content.js על ידי ה-background.js הוא מחשב תוך כמה זמן הוא צריך להריץ את הפעולה וכך גם מבצע אותה על השנייה המתאימה. כך מתאפשר חוויית צפייה מסונכרנת בין צופי החדר.

תרשים תקשורת כללית בין המרכיבים במערכת:



תרשים זרימה לפעולה של ה-host על סרטון היוטיוב והשפעתו על שאר חברי הקבוצה:



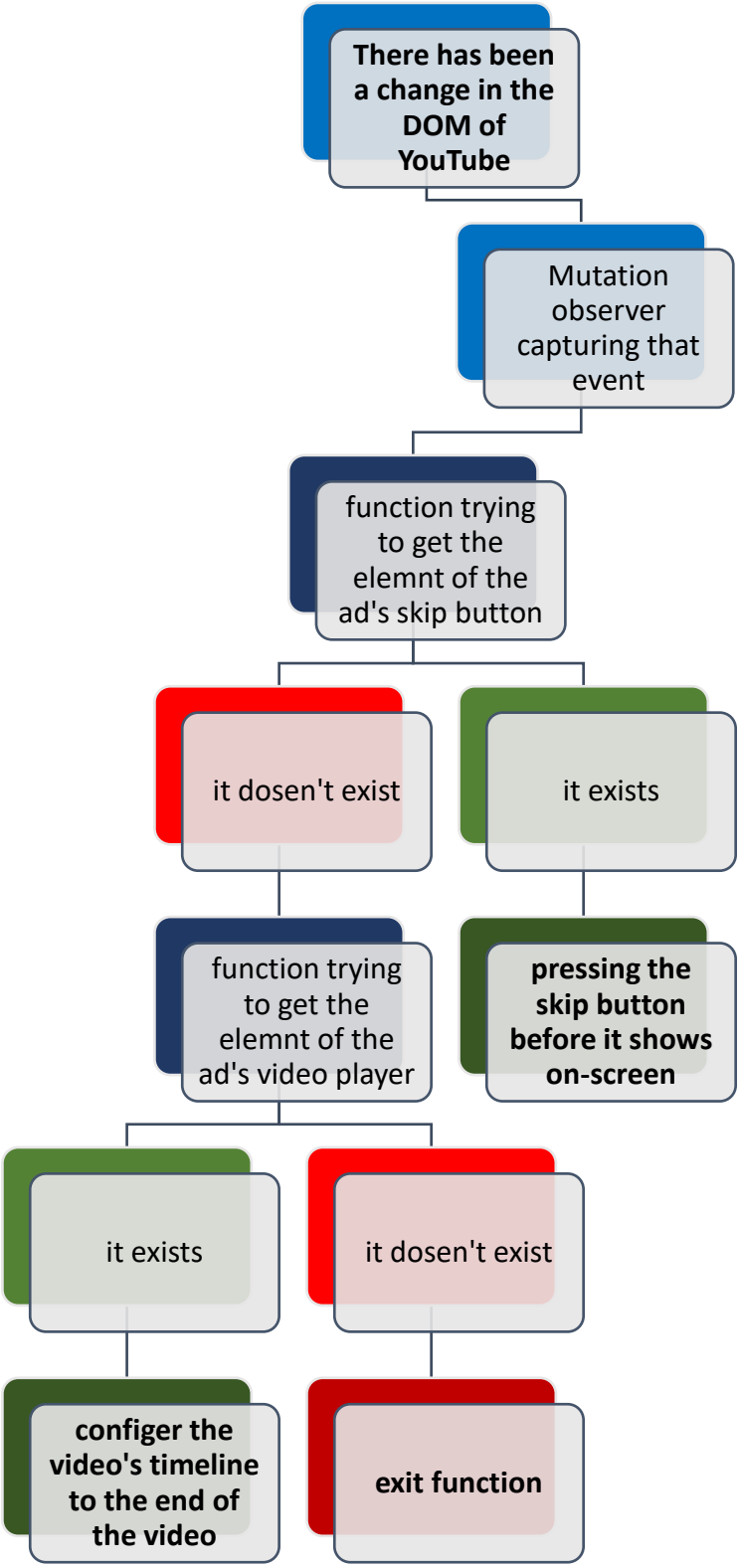
(2) איך למנוע את הפרסומות בסרטון היוטיוב?

החלטתי לבחור בפתרון השלישי לבעיה זו. כלומר מצאתי לנכון להתמודד עם בעיה זו בכך שאני אכתוב בתוך התוסף שלי חוסם פרסומות למניעת הפרסומות המפריעות להרצת הסרטון במהלכו. בחרתי בפתרון זה מכיוון שראשית בחרתי בבעיה הראשונה בפתרון של הרצת הסרטון המשותף על אתר יוטיוב ולכן הורדת הסרטון ל-mp4 היא אינה אפשרות זמינה כעט. בנוסף לכך החלטתי לא לדרוש מהמשתמש להוריד חוסם פרסומות חיצוני מכיוון שבתוך יוצר מוצר ללקוח פחות מתאים או מקובל לדרוש ממנו להוריד מוצר חיצוני על מנת לשפר את חווית המוצר שאותו אני יוצר. ולבסוף עוד סיבה לכתיבת חוסם הפרסומות בעצמי היא מכיוון שאני כותב קוד על אתר יוטיוב יש לי כבר בסיס טוב לכתוב עליו חוסם פרסומות מבלי להתאמץ יותר מידי מכיוון שכל התשתית כבר בנויה מסביב לפונקציה עצמה שתחסום פרסומות. עקב הסיבות הללו החלטתי לבחור בפתרון השלישי ואכן לכתוב חוסם פרסומות לאתר יוטיוב.

מבנה המערכת:

חוסם הפרסומות שבניתי בתוסף רץ על ה-content.js שאחראי על הרצת פקודות בסרטון המשותף ביוטיוב. הוספתי בקובץ זה MutationObserver שהוא בעצם event שמריץ פונקציה כל פעם שמתקיים שינוי ב-DOM של האתר. לאחר מכן בפונקציה אני מנסה למצוא אלמנטים של פרסומת היכולים להתקיים. אם נמצא האלמנטים המוודים שאכן ישנה פרסומת המפריעה להרצת הסרטון יש שני אפשרויות לפי סוג הפרסומת המופיעה. (1) פרסומת בעלת כפתור דילוג: במידה וזה המצב אז כפתור הדילוג נלחץ אוטומטית עוד לפני שהוא מופיע בכלל על מסך המשתמש. (2) אם זאת פרסומת ללא כפתור דילוג אז נשיג את אלמנט של סרטון הפרסומת עצמו ונגדיר ב-timeline שלה שהיא בסוף הסרטון וכך זה ידלג לחלוטין על הפרסומת. פעולות אלו קוראות במילי-שניות ככה שהמשתמש בקושי שם לב אליהם וכך חווית הצפייה המשותפת אינה נפגעת כלל.

תרשים זרימה לחוסם הפרסומות בסרטון היוטיוב:



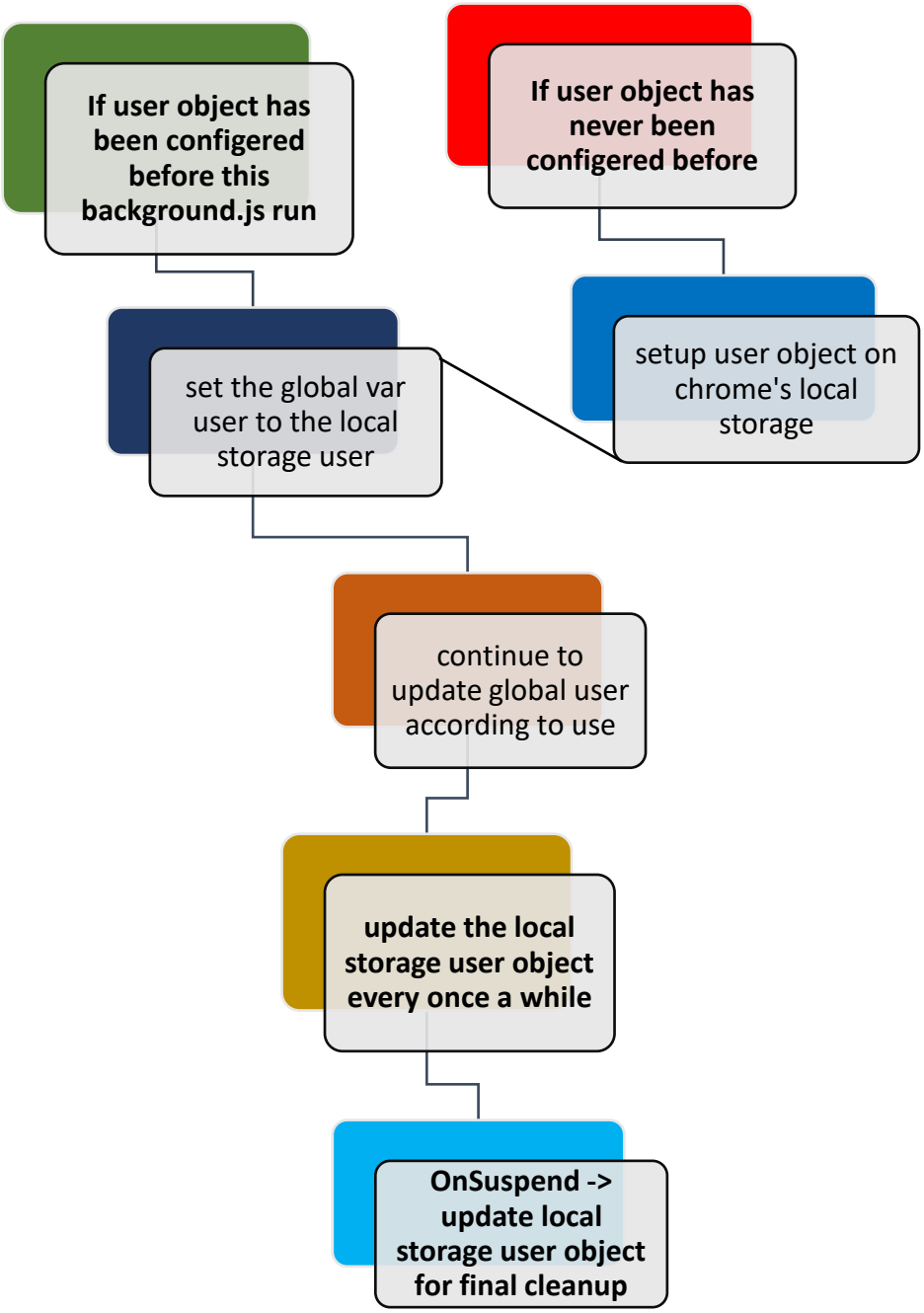
(3) איך לשמור את המידע על המשתמש בתוסף?

הפתרון שבחרתי לבעיה זו הינו הפתרון השני שהוא עבודה עם storage.local של כרום כמקום השמירה למשתנים הגלובליים של המשתמש. בחרתי דווקא בפתרון זה לעומת עבודה עם קובץ נפרד המכיל את המידע על המשתמש מכמה סיבות. ראשית ניתן לשמור באכסון של כרום אובייקטים ולא רק מילים, כך ניתן לשמור בצורה יותר נוחה את המידע כמו ה-WebSocket של הלקוח. בנוסף לכך שמירת המידע באכסון כרום יותר מאובטח מקובץ חיצוני ושמור בצורה פנימית בתוך הדפדפן לאורך כל השימוש. לבסוף העדפתי להשתמש באכסון של כרום מכיוון שהיא פועלת בצורה הרבה יותר נוחה סביב הממשק הבנוי של תוספי כרום ולכן העבודה איתה היא יותר אינטואיטיבית ותואמת לפרויקט עצמו.

מבנה המערכת:

כאשר לראשונה המשתמש מוריד את התוסף נבנה אובייקט המכיל את כל המשתנים של המשתמש כגון: ID, username, WebSocket, פרטי חדר צפייה וכו'. לאחר מכן אני מציב את האובייקט הזה בתוך ה-local storage של כרום ולבסוף בכל פעם שאני משנה את האובייקט של המשתמש אני דואג לעדכן אותו באכסון הלוקלי כך שאם ה-background.js עושה unload ברגע שהוא יחזור לתפעול הוא יוצא מחדש את האובייקט המשתמש המעודכן וכך נשמר כל המידע החשוב על המשתמש לאורך השימוש בתוסף. בנוסף לכך הצבתי עוד פקודה שאמורה לוודא כי כל פרטי המשתמש נשמרו והיא קליטת ה-event שברגע ש-background.js עומד לעשות unload הוא מעלה את פרטי המשתמש לאכסון הלוקלי וכך מאבטח אכסון של כל פרטי המשתמש במצב בו הקוד עומד לעבור אתחול.

תרשים זרימה לשמירת וביצוע פרטי המשתמש ב-local storage של כרום.



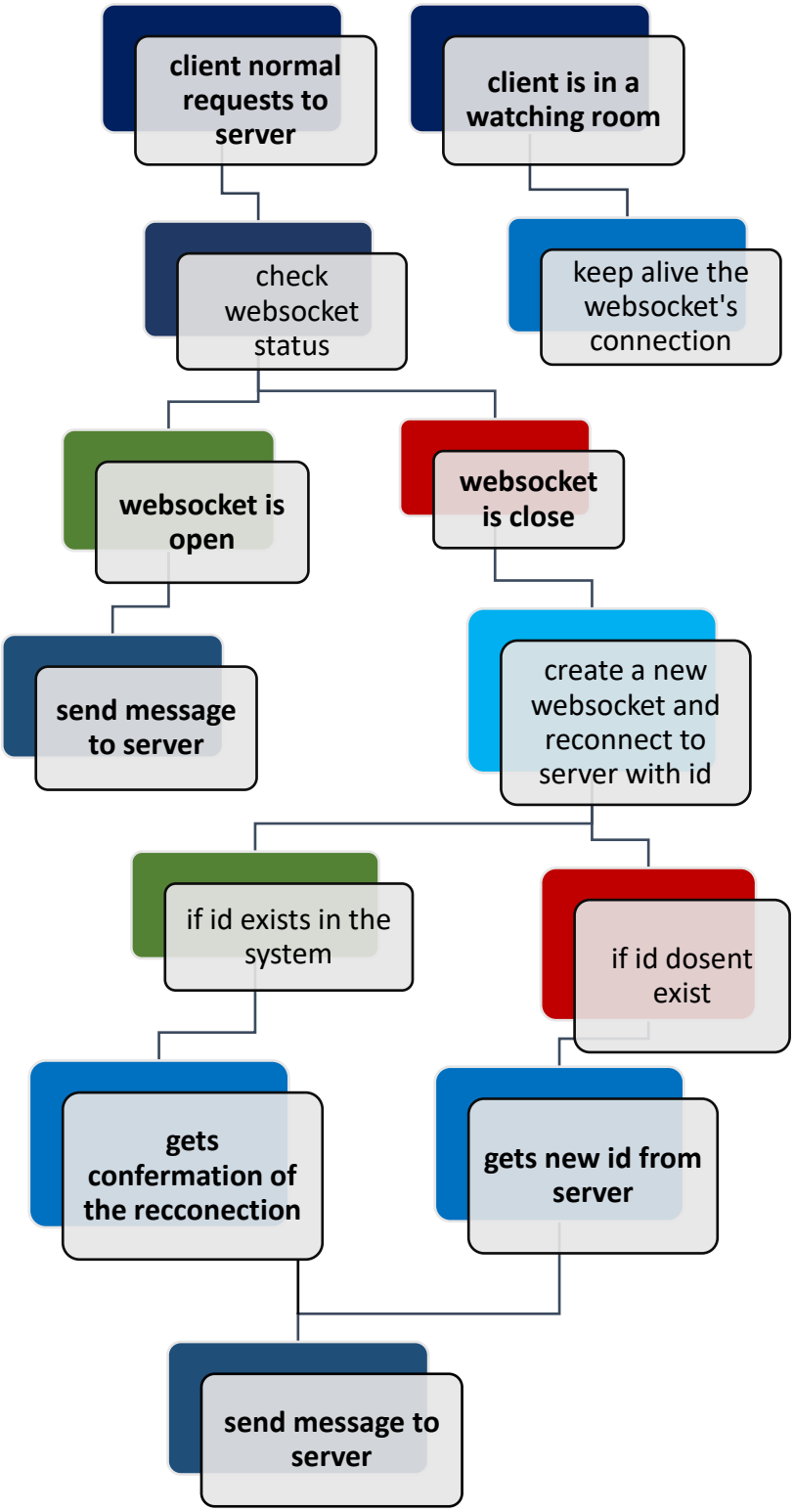
(4) כיצד לשמור על קשר בין התוסף לשרת?

הפתרון שבחרתי לבעיה זו הינה ערבוב בין שני הפתרונות הקיימים בהתאם לסיטואציה של התוסף. הגעתי למסקנה שישנם חסרונות ויתרונות לשני הפתרונות ועל מנת לפתור את הבעיה ולחשוב על פתרון מתאים צריך להשתמש בשני הפתרונות בכך שכל פתרון בסיטואציה שונה. כאשר מתקיים קשר רגיל של שאלה תשובה בין הלקוח לשרת אז לפני כל פעם שאני שולח הודעה דרך ה-WebSocket אני בודק ראשית את מצב החיבור. אם מצבו טוב אז הוא ממשיך ושולח את ההודעה אך אם הוא לא אז נשלח הודעת חיבור מחדש לשרת עם אובייקט WebSocket חדש, לאחר שדבר זה מתבצע השרת שומר מחדש את פרטי הלקוח ל-WebSocket העדכני והחדש ולאחר מכן מאשר חיבור מחדש ללקוח. לעומת זאת במצב שהלקוח החדר צפייה אז הוא צריך להיות מוכן בכל שנייה לקבל הודעה מהשרת ולכן אי אפשר להסתמך על יצירת קשר חדש כל פעם שהקודם נופל ולכן אני שולח אשלח הודעה כל כמה שניות על מנת לשמור על החיבור בין הלקוח לשרת.

מבנה המערכת:

במצב כאשר הלקוח אינו בחדר צפייה אז יצרתי פונקציה של שליחת הודעה לסרבר, הפונקציה בודקת את מצב ה-WebSocket של הלקוח, במידה והתקשורת פתוחה הוא שולח הודעה לשרת ומתקשר כרגיל. במידה והקשר בניהם במצב סגירה אז הלקוח פותח מחדש את התקשורת עם WebSocket חדש באמצעות ה-ID של הלקוח – השרת מוודא שה-ID אכן נמצא במערכת ואם הוא אכן נמצא אז מעדכן באכסון השרת ה-WebSocket של הלקוח ואם לא נמצא ה-ID אז נשלח ללקוח ID חדש ורושם אותו במערכת בצורה מחודשת. לאחר ומתקיים החיבור מחדש נשלח ההודעה הרצויה בהתחלה כרגיל.
במידה והלקוח בחדר צפייה אז כל כמה שניות נשלח לשרת הודעה דרך ה-WebSocket של הלקוח על מנת לשמור על קשר פתוח בין השניים במידה והשרת רוצה להפיץ פקודה ללקוח בנוגע לסרטון.

תרשים זרימה לתקשורת בין התוסף לשרת לפי מצב.



פיתוח הפתרונות בשכלול הקוד עם שפת התכנות

(1) כיצד ניתן לצרוך סרטון משותף על האינטרנט?

"הזרקת" הקוד לתוך סרטון היוטיוב, קבלת מידע מהמשרת ותפעול החדר מ-background.js

```
function run_room_process(){
    if (user.room_process[0]==false){
        user.room_process[0]=true;
        const tabId=parseInt(user.room[3]);
        chrome.scripting.executeScript(
            {
            target:{tabId: tabId},
            files:["content.js"],
            }

        );
        //sending content script if user is host or not
        setTimeout(notify_content_info,1000);
        user.room_process[1]=setInterval(check_status,4500);

        user.connection.onmessage=function(event){ //HERE

            var msg=String(event.data);

            console.log("got msg!!!!: ",msg);

            if(msg.includes("w.r,")){

                if (msg.includes("new_u")){
                    console.log("new member has joined")
                    msg=msg.split(','); //w.r,new_u,room_id,u1,u2,u3,u4,u5
                    msg.splice(0,3);
                    user.room_members=msg;
                    console.log("members: ",user.room_members)
                    chrome.tabs.sendMessage(user.room[3],user.room_members,function(response){ //{command:"W?"}

                    })
                }
                else if (msg.includes("left_u")){
                    console.log("member has left")
                    msg=msg.split(','); //w.r,left_u,room_id,u1,u2,u3,u4,u5
                    msg.splice(0,3);
                    user.room_members=msg;
                    console.log("members: ",user.room_members)
                    chrome.tabs.sendMessage(user.room[3],user.room_members,function(response){ //{command:"W?"}

                    })
                }
                else{
                    console.log("sending content.js"); //0-w.r,1-command,2-vid t1,3-UTC
                    chrome.tabs.sendMessage(user.room[3],msg,function(response){ //{command:"W?"}

                    })
                }
            }

        }
    }
    else{
        return
    }

}
```

מעקב ה-content.js על הסרטון ועדכון סטטוס

```
//content.js
var video
var vidUrl
var playButton
var ishost=false
var timeline
var server_order=false

//when youtube first loading - waits until the video element has loaded
document.addEventListener('yt-navigate-finish',process);

//checks if webpgae has been loaded when re-entering the page
if (document.body) process();
else document.addEventListener('DOMContentLoaded',process)


function process(){

    video = document.querySelector('video');
    playButton=document.getElementsByClassName("ytp-play-button ytp-button")[0]
    bar=document.getElementsByClassName("ytp-progress-bar-container")[0]
    video.pause();

    //ON MESSAGE FROM background.js
    chrome.runtime.onMessage.addListener(function(message,sender,sendResponce){

        console.log("msg",message);
        if(message.includes("w.r,")){
            console.log("cmd")
            run_command(message);

            return true; //stopping message port closing
        }

        else if(message=="k.a"){
           sendResponce("^");
        }

        else{

            data=message.split(",")

            //ishost

            if(data[0]=="true"){
                console.log("t");
                ishost=true;
            }
            else if(data[0]=="false"){
                console.log("f");
                ishost=false;
            }


            //vidUrl
            vidUrl=data[1]
            console.log(vidUrl)

            return true; //stopping message port closing

        }

    });
```

קליטת פקודות של המשתמש על הסרטון במידה והמשתמש הוא ה-host

```
//VIDEO EVENTS

video.onplaying=function(){//if user pressed play
    console.log("cs- ",server_order,"&& ih- ",ishost);
    if (ishost && server_order==false){
        console.log("sending onplaying");
        //letting know to server to play everyone in spesific timestamp
        chrome.runtime.sendMessage("watching_room,playing,"+video.currentTime, (response) => {

        });
    }
}
video.onpause=function(){
    console.log("cs- ",server_order,"&& ih- ",ishost);
    if (ishost && server_order==false){
        console.log("sending onpause");
        //letting know to server to pause everyone in spesific timestamp
        chrome.runtime.sendMessage("watching_room,paused,"+video.currentTime, (response) => {

        });
    }
}
video.ontimeupdate=function(){
    console.log("cs- ",server_order,"&& ih- ",ishost);
    if (ishost && server_order==false){
        console.log("sending ontimeupdate");
        //if user changed timeline
        if (Math.abs(video.currentTime-timeline)>1){

            //pauseing vid ->letting know to server that time has changed->server plays in sync
            chrome.runtime.sendMessage("watching_room,move tl,"+video.currentTime, (response) => {

            });
        }

        timeline=video.currentTime;
    }


}
```

ביטול שליטה על הסרטון במידה והמשתמש אינו host

```
//disable controls for non-host users
window.addEventListener('click', event => {
    if (event.target.matches('video')) {
        if(!ishost){
            event.stopPropagation();
        }
    }
}, true);

setInterval(disablecontrol,1000);
```

```
// DISABLE/ENBALE CONTROLS FOR USERS
function disablecontrol(){

    if(!ishost && playButton.style.display!="none"){
        console.log("disabling control")
        playButton.style.display="none";
        bar.style.display="none";
    }
    else if(ishost && playButton.style.display!=""){
        console.log("allowing control")
        playButton.style.display="";
        bar.style.display="";
    }

}
```

(2) איך למנוע את הפרסומות בסרטון היוטיוב?

האלגוריתם שחוסם פרסומות ב-content.js

```
//AD BLOCKER
let observer = new MutationObserver(mutations => { //every time a change is happend in the DOM

    console.log("OBERVER")

    var skipButton=document.getElementsByClassName("ytp-ad-skip-button");
    var unskipAdd=document.querySelector(".html5-video-player.ad-showing video");

    //checking if skip button is present
    if(skipButton!=undefined && skipButton.length>0){

        console.log("AD DETECTED - SKIPPABLE");
        skipButton[0].click();
    }
    //if there is unskippable ad
    else if(unskipAdd!=undefined){

        console.log("AD DETECTED - UNSKIPPABLE");
        unskipAdd.currentTime=10000;
    }

});

observer.observe(document, { childList: true, subtree: true });
```

(3) איך לשמור את המידע על המשתמש בתוסף?

אתחול האובייקט user באכסון הלוקלי של כרום במידה וזה הרצה ראשונה של הקוד.

ובמידה והקוד קיבל Unload אז מוצב אובייקט ה-user הגלובלי מחדש.

```
var user


//setting up user's info in local storage
chrome.storage.local.get(['userLocal'], async function (result) {
    let ul = result.userLocal;
    if (ul === undefined) {
        console.log("entering user")
        // it means there was nothing before. This way you don't overwrite
        // the user object every time the backgorund.js loads.
        let ul={
            connection:undefined,
            id:"id",
            username:"username",
            room:["id","password","url","tabid","ishost"],
            room_members:[],
            in_room:false,
            room_process:[false,"intervelId"],
            current_tab:["url","id"]
        }
        user=ul;
        await chrome.storage.local.set({userLocal: ul}, function () {}); // save it in local.
        await connect();
    }
    else{
        console.log("rentering user");
        console.log("r_u ",ul);
        user=ul;
        reconnect();
    }
});
```

פונקציה לעדכון האובייקט בשמירה הלוקלית ופונקציה לקבלת האובייקט מהאכסון

```
function update_user(tmp_user){
    console.log("updating user")
    chrome.storage.local.get(['userLocal'], async function (result) {
        var userLocal = tmp_user;
        chrome.storage.local.set({userLocal: userLocal}, function () {}); // save it in local.

    });
}
```

(4) כיצד לשמור על קשר בין התוסף לשרת?

פונקציות שליחת הודעה לשרת כאשר לא מדובר על תקשורת של חדר צפייה

```
function send_message(msg){
    check_connection().then((message)=>{
        user.connection.send(msg);
    }).catch((error)=>{
        reconnect();
        user.connection.send(msg);
    });
}

function check_connection(){
    return new Promise((resolve,reject)=>{

        if (user.connection==null){
            reject("recconecting")

        }
        else if (user.connection.readyState != 1 ){
            reject("recconecting")
        }
        else{
            resolve("connection is still open");
        }
    });
}

function reconnect(){


    user.connection = new WebSocket('ws://localhost:8765 ');


    user.connection.onopen = function(e) {
        console.log("sending: ",user.id)
        user.connection.send("reconnecting,"+user.id+","+user.username);
    };

    user.connection.onmessage=function(event){

        console.log("got data ",event.data)
        if (String(event.data).includes("new id,")){
            console.log("getting new id...");

            let temp_i=(event.data.split(','))[1]
            user.id=temp_i;

            update_user(user);

        }
        else{
            console.log(event.data);
        }

    };
}
```

במצב שבוא המשתמש בחדר צפייה ונשלח הודעה לשרת כל כמה שניות על מנת לשמור על החיבור קיים זה פונקציה שרצה כל כמה שניות ומוגדרת בבעיה מספר 1 בפונקציה (run room process)

```
function check_status(){

    //if user exited room
    if (user.room_process[0]==false){

        console.log("exiting msging");
        clearInterval(user.room_process[1]);
    }
    else{
        user.connection.send("k.a") // keep alive the connection bewtween client and server
        chrome.tabs.sendMessage(user.room[3],"k.a",function(response){

        }) ;
    }


}
```

תיאור המודלים של מערכת התכנה

כפי שהזכרתי בפרק המבוא, הפרויקט בנוי מכמה מודלים וקבצים נפרדים שפועלים יחדיו על מנת ליצור את התוצר הסופי. כל קובץ בעל תפקיד שונה ופועל בזמנים שונים בהתאם למצב הלקוח במוצר.

- manifest.json

קובץ זה הוא קובץ הצגה ל-API של תוסף הכרום והוא מציג את כל הקבצים המשומשים בתוסף יחד עם הרשאות שימוש ותמונות עיצוב לתוסף.

- background.js

קובץ האחראי על כל ה-backend של תוסף הכרום. הוא מתפעל את הקשר עם השרת, events ומחזיק את כל המידע על המשתמש כגון ID, username, מידע על החדר צפייה וכו'

- content.js

אחראי לתפעול חדר הצפייה של המשתמש, קובץ זה "מוזרק" לאתר יוטיוב ואחראי להרצת הסרטון בצורה מסונכרנת, קבלת הודעות מ-background.js וחוסם פרסומות היכולות להופיע במהלך הסרטון.

- popups.js

קובץ זה מכיל את הסקריפטים לפופ-אפים הקיימים בתוסף, הוא מקשר בין מה שמוצג במסך למידע שקיים ב-background.js

- watching_room.js

קובץ זה אחראי בפרט על הפופ-אפ של חדר הצפייה ואחראי להציג ולעדכן את רשימת הצופים בחדר.

- dif_popup.html

הפופ-אפ הדיפולטי בתוסף, מאפשר למשתמש להצטרף לחדר צפייה באמצעות ID וסיסמא.

- hello.html

אתר הפתיחה המוצר למשתמש כאשר הוא לראשונה מוריד את התוסף, בעל הוראות הנחייה והסברים התחלתיים.

- in_room_popup.html

הפופ-אפ כאשר המשתמש בחדר הצפייה. הוא מכיל את ה-ID והסיסמא של החדר יחד עם רשימת צופים.

- watching_popup.html

הפופ-אפ כאשר המשתמש בסרטון יוטיוב, דרכו ניתן לפתוח חדר צפייה על הסרטון בלחיצת כפתור.

- index.css

קובץ עיצוב לפופ-אפים.

-welcome.css

קובץ עיצוב לעמוד הפתיחה.

-server.py

השרת שמנהל את התוספים, מכיל מידע על כל המשתמשים ואחראי על תפעול חדרי הצפייה.

תיעוד קוד

background.js

כותרת פונקציה	טענת כניסה	טענת יציאה	תפקיד
connect()	-	-	מתחבר לשרת ומקבל ID ו-username
check_connection()	-	-	בודק חיבור של ה-WebSocket לשרת
reconnect()	-	-	מתחבר מחדש לשרת בעזרת ה-ID של הלקוח
update_user()	אובייקט ה-user שגלובלי	-	מעדכן את אובייקט המשתמש המאוכסן לוקלית בעזרת האובייקט הגלובלי
run_room_process()	-	-	"מזריק" לתוך סרטון היוטיוב את ה-content.js ויוצר תקשורת עם השרת לגבי פקודות ופעולות שצריך לבצע בסרטון
send_message()	הודעת שליחה לשרת	-	הפונקציה קוראת ל-check_status ושולחת את ההודעה המבוקשת לשרת
check_status()	-	-	בודק אם המשתמש יצא מחדר הצפייה ומחזיק את החיבור עם השרת בחיים
notify_content_info()	-	-	מעדכן את content.js אם המשתמש ה-host של חדר הצפייה.

content.js

כותרת פונקציה	טענת כניסה	טענת יציאה	תפקיד
disable_control()	-	-	מבטל\מאפשר שליטה של המשתמש על הסרטון בהתאם להרשאות שלו
is_open()	-	-	בודק שהמשתמש עדיין בחדר הצפייה, אם לא הוא מעדכן את background.js וסוגר את החלון
run_command()	מקבל את הפעולה שעליו להריץ בסרטון	-	מבצע את הפעולה על הסרטון שאותה קיבל מ-background.js
wait_time()	מקבל את ה-UTC שעליו לבצע את הפעולה	מחזיר תוך כמה שניות עליו להריץ את הפעולה	מחשב תוך כמה זמן עליו להריץ את הפעולה על הסרטון

Watching_room.js

כותרת פונקציה	טענת כניסה	טענת יציאה	תפקיד
insert_members()	מקבל רשימת צופים בחדר	-	מעדכן את ה-popup של חדר הצפייה בהתאם לכמות הצופים בחדר

server.py

כותרת פונקציה	טענת כניסה	טענת יציאה	תפקיד
create_new_id()	את אובייקט ה-websocket של הלקוח	ID חדש של המשתמש	מבטל\מאפשר שליטה של המשתמש על הסרטון בהתאם להרשאות שלו
create_new_username()	ID של המשתמש	שם משתמש ללקוח	בודק שהמשתמש עדיין בחדר הצפייה, אם לא הוא מעדכן את background.js וסוגר את החלון
create_new_room()	ה-ID של הלקוח שיצר את החדר וה-URL שהחדר נמצא בו	מחזיר את ה-ID של החדר צפייה החדש	מבצע את הפעולה על הסרטון שאותה קיבל מ-background.js
generate_comb()	רשימת בוליאנים של אפשרויות עם אותיות גדולות קטנות, מספרים וסמלים	מחזיר קומבינציה רנדומלית	יוצר קומבינציה רנדומלית של סמלים אותיות ומספרים בהתאם לתנאי הכניסה
cmd_utc()	-	מחזיר את הUTC שבוא צופי החדר יבצעו את הפעולה.	הפונקציה מוציאה את זמן ה-UTC שבו חברי הצוות יבצעו את פעולת הסרטון
broadcast()	הודעה שרוצים להעביר	-	שולחת לכל חברי חדר הצפייה את ההודעה המבוקשת

השוואת העבודה עם פתרונות ויישומים קיימים

לעניין שיתוף סרטונים וצפייה משותפת עם חברים קיימים בשוק מספר פתרונות אחרים. הראשונה הינה אתר watch together זה אתר שדרכו ניתן לצפות בסרטונים ולפתוח חדרי צפייה בעזרת הכנסת ה-URL של הסרטון. בנוסף לכך האופציה השנייה לחוויית צפייה משותפת היא דרך מספר יישומים כמו ZOOM, discord, skype וכו. שם ניתן לשתף מסך וכך לראות עם חברים סרטונים.

הסיבה בה בחרתי בפרויקט זה דווקא כפי שציינתי כבר בהתחלה זה שלא מצאתי מוצר ברמה שיווקית כמו שאני יצרתי. קוב האפליקציות והדרכים לצפייה משותפת דורשות אפליקציה חיצונית ולוקחות יחסית זמן להריץ ולצפות ביחד בסרטונים. דרך התוסף שלי שבנוי על כרום ניתן בעזרת לחיצת כפתור אחת לפתוח חדר צפייה. והדבר לא דורש URL או כלום, בכל מקרה המשתמש רואה סרטונים ביוטיוב וכך אם בא לו לראות את הסרטון יחד עם חבר הוא רק פותח את פופ-אפ של התוסף ופותח על האתר עצמו חדר צפייה, ללא התעסקות או מאמץ כלל.

לעומת זאת שאר הפתרונות בשוק הם בעלי יותר יכולות כמו צא'ט או אפילו שיחה קולית ובעל UI יותר מלא ומיוחד לאפליקציה, אבל בתוסף אצלי רוב ה-UI מוסתר בפופ-אפ ואין דרך לתקשר בצורה ישירה עם צופי החדר אך זה נעשה כפי שהוא מכיוון שהעדפתי לשמור על האפליקציה כמה שיותר Low profile על מנת לחזק את היתרון שבפשטות והנוחות שהיא מציעה למשתמשים ולקוחות.

הערכת הפתרון לעומת התכנון והמלצות לשיפורו

אני אישית נורא מרוצה מהפתרון והתוצר הסופי של האפליקציה. נהניתי מכל רגע שעבדתי עליו ולמדתי דרכו המון על עולם חדש שלא יצא לי להתנסות בו לפני. אני מאמין כי בצאתי את כל המטרות המרכזיות שתכננתי ובסופו אם מסתכלים על הפרויקט הזה כפרויקט ראשון שלי בתחום הידע של תוספי כרום,HTML,JavaScript, CSS,WebSocket, ועבודה אסינכרונית התוצר הסופי מעולה. בנוסף לכך בסופו של דבר התפקוד באפליקציה מרגיש אינטואיטיבי ופשוט ושומר על נגישות קלה ונוחות למשתמש. אלו היו מטרות מרכזיות שלפיהם אני אישית קבעתי את מידת שביעות הרצון שלי לגבי הפרויקט.

בעתיד אם היה לי עוד זמן לעבוד על הפרויקט או אם הייתי מתחיל מחדש לאחר כל הידע שצברתי בעולם תכנות הזה הייתי מוסיף אפשרות תקשורת בין צופי חדר הצפייה, הייתי מוסיף אופציה שבא כולם יכולים לתפעל את הסרטון בו זמנית וכל הנראה אני מוסיף אופציה לעיצוב המשתמש כמו תמונת פרופיל, שם משתמש שהלקוח קובע ואפשרויות תפעול לפי נוחות והעדפת המשתמש.

תיאור של הממשק למשתמש – הוראות הפעלה

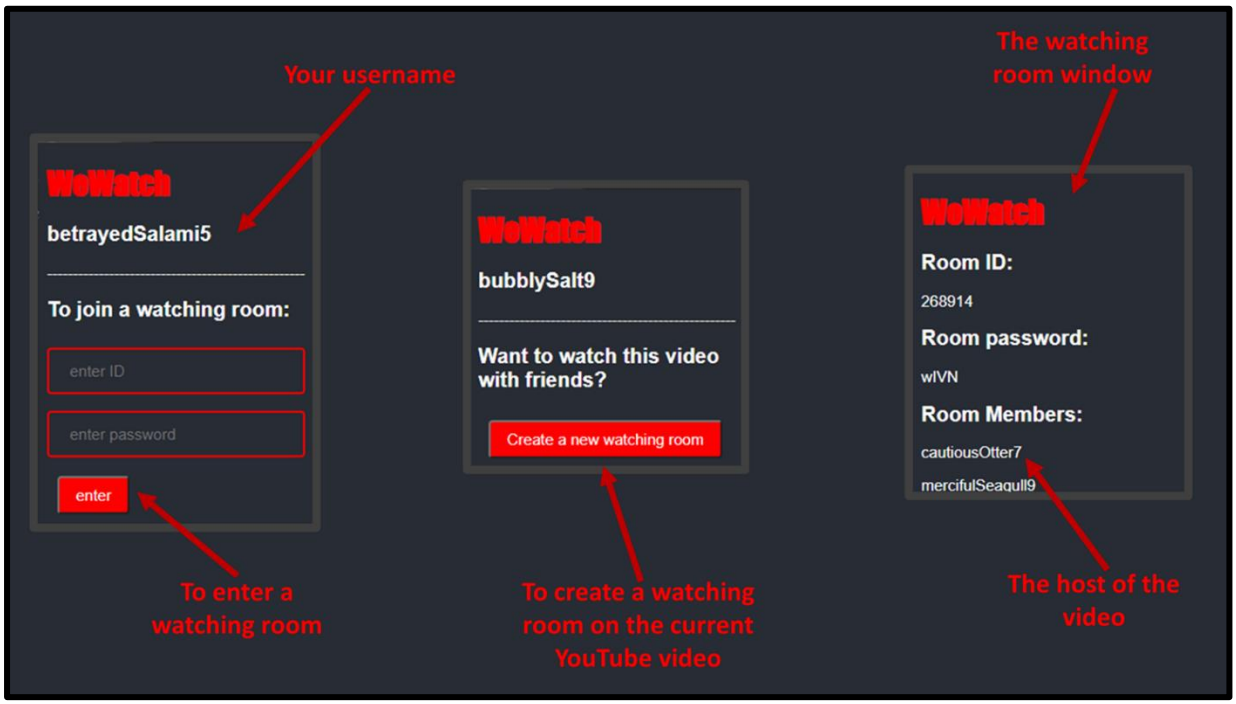
שלום צופה יקר, ברוכה הבאה ל-WeWatch – חווית הצפייה המשותפת הכי נוחה שקיימת.

ממשק התוכנה פשוט מאוד, כאשר אתה רוצה לפתוח חדר צפייה כל מה שעליך לעשות זה להיות בסרטון יוטיוב -> ללחוץ על הפופ-אפ של התוסף ומשם לפתוח חדר צפייה על אותו סרטון יוטיוב. לאחר מכך תקבל בחלון הפופ-אפ ID וסיסמא לחדר שדרכם חבריך יכולים להצטרף ביחד איתך לחוויית הצפייה המשותפת.

כאשר אתה רוצה להצטרף לחדר צפייה כל מה שעליך לעשות זה לפתוח את חלון הפופ-אפ דרך כל אתר שהוא לא סרטון צפייה, שם תוכל להכניס את פרטי החדר ואם הם נכונים תוכוון אוטומטית לחדר צפייה עם חבריך.

כאשר אתה בחדר צפייה רק ל-host של החדר יש שליטה על הסרטון, אפשר לדעת מי זה לפי האדם הראשון ברשימה, לפיו יקבע פעולות ההרצה של הסרטון והזמן, כל צופה ראשי לכוון אישית את גודל המסך, השמע, איכות וכו'.

להלן כל חלונות הפופ-אפים בתוסף עם הסברים תפקודיים על כל אחד מהם:



מבט אישי על העבודה ותהליך הפיתוח

בהתחלה רק כשהתחלתי לעבוד על בניית תוסף ולתכנן את הפרויקט הייתי מאוד לחוץ מהעובדה שאני בונה פרויקט גמר בסביבת עבודה חדשה לגמרי, בשפות חדשות ומבנה שונה. אך עם הזמן התחלתי ללמוד ולהתקדם בהבנה שלי בנושאים הרבים החדשים ולאט לאט הרגשתי וראיתי את התפתחות שעברתי במהלך כתיבת הפרויקט. עצם העובדה שעבדתי בחומר שונה ובסביבת עבודה חדשה לגמרי גרמה לזה שפחות הספקתי להתייחס לכל הפרטים הקטנים ולא הספקתי להוסיף פיצ'רים נוספים שנותנים תחושה של מוגמרות. בנוסף לכך כתיבת תוסף זה תהליך שאין עליו המון מידע באינטרנט או המון עוסקים בתחום ולכן היה מאוד קל לפתוח תוסף פשוט שלדוגמא יכול להחליף רקע אך להגיע מזה לפרויקט הגורם לצפייה משותפת ביוטיוב לקח המון זמן והמון חקירה אישית שבאה מניסיון וטעיה. הייתי מאוד פעיל באתרי עזרה כמו stack overflow ונעזרתי בכמה אנשים שעזרו לי להבין כיצד עליי לפתח את הקוד על מנת להתקדם.

אך למרות כל הקשיים שעברתי והתהליך הארוך שלקחתי על עצמי על מנת להוציא את תוצר זה אני מאוד גאה בתוצאה הסופית שהגעתי עליה, בסופו של יום נהניתי מאוד מהתהליך שעברתי עם עצמי לאורך כתיבת הפרויקט ודרכו יצא לי לראות את הלמידה שלי בתיכון מתממשת בפרויקט שיצא מחוץ לתוכנית הלימודים.

יצא לי ללמוד בזכות הפרויקט הזה המון דברים חדשים כמו עבודה ב-EDA, להכיר JavaScript, לעבוד עם אתרים באמצעות HTML ו-CSS, תקשורת של WebSocket, כיצד לשנות אובייקטים באתרים באמצעות קוד ואיך לכתוב תוסף לכרום. בסופו של דבר הידע שצברתי בתחומים חדשים שווה יותר מכל פרויקט אחר שהייתי יכול לעשות באזור הנוחות שלי ולכן אני מאוד מרוצה מהתוצר הסופי והעבודה שיצא לי לעשות במהלך החמישה חודשים שעברו.

ביבליוגרפיה

במהלך הפרויקט נעזרתי מספר אתרים על מנת ללמוד את כל החומר החדש שעבדתי דרכו. להלן רשימת האתרים הללו:

- https://developer.chrome.com/docs/extensions/mv3

אתר הסבר ומדריך לכתיבת תוסף סביב ה-API של תוספי כרום, שם נמצאים כל הפיצ'רים של תוספי כרום וכיצד לעבוד איתם.

- https://stackoverflow.com

אתר בלוגים המלא בשאלות ותשובות בכל תחום, שם נעזרתי בחברי הקהילה ועברתי על שאלות ישנות בכל התחומים של תוספי כרום, JavaScript ו-WebSocket.

- https://developer.mozilla.org

אתר עם הסברים על פונקציות ודרכי עבודה ב-JavaScript, WebSocket, web design וכו'.

קוד פרויקט

background.js

```
var user


//setting up user's info in local storage
chrome.storage.local.get(['userLocal'], async function (result) {
    let ul = result.userLocal;
    if (ul === undefined) {
        console.log("entering user")
        // it means there was nothing before. This way you don't overwrite
        // the user object every time the backgorund.js loads.
        let user={
            connection:undefined,
            id:"id",
            username:"username",
            room:["id","password","url","tabid","ishost"],
            room_members:[],
            in_room:false,
            room_process:[false,"intervelId"],
            current_tab:["url","id"]
            }
        chrome.storage.local.set({userLocal: user}, function () {}); // save it in local.
    }
    else{
        console.log("rentering user");
        console.log("r_u ",ul);
        user=ul;
    }
});




//WHEN THE EXTENTION IS BEING INSTALLED
chrome.runtime.onInstalled.addListener(async () => {
    //opens welcome page
    let url = chrome.runtime.getURL("htmls/hello.html");

    let tab = await chrome.tabs.create({ url });


    user= await get_user();
    console.log("u: ",user);

    //connect to server and recive id
    connect();

});
```

```
//WHEN BACKGROUND.JS IS ABOUT TO GO IDLE
chrome.runtime.onSuspend.addListener(function (){
    /*
    if(user.in_room){
        chrome.tabs.sendMessage(user.room[3],"close cjs",function(response){ //{command:"W?"}
        })
        clearInterval(user.room_process[1]);
        user.room_process[0]=false;
    }
    */
    update_user(user);//upload user before going idle
});



//WHEN THE ACTIVE TAB CHANGES CHROME
chrome.tabs.onActivated.addListener( function(activeInfo){


    //checks current tab and update the popup accordingly
    chrome.tabs.get(activeInfo.tabId, function(tab){
        var u = tab.url;
        //console.log("OnActivated-you are here: "+u);

        user.current_tab[0]=u;
        user.current_tab[1]=tab.id;


        if(u==user.room[2] && user.in_room && (user.current_tab[1]==user.room[3] || user.room[3]=="tabid")){

            chrome.action.setPopup({popup: 'htmls/in_room_popup.html'});
            user.room[3]=tab.id;



            console.log("onActivated did it")
            run_room_process();
        }
        else if(String(u).includes("https://www.youtube.com/watch")){


            chrome.action.setPopup({popup: 'htmls/watching_popup.html'});
            user.current_tab[0]=u;



            //console.log("set current tab to: "+ current_tab)
        }
        else{
            chrome.action.setPopup({popup: 'htmls/dif_popup.html'});
        }

        update_user(user);
    });
```

```
//WHEN THE TAB IS UPDATED -I NEED TO TAKE INTO ACCOUNT IF USER GOES BACKTAB I NEEED TO CHANGE ROOM TO FALSE!
chrome.tabs.onUpdated.addListener(function (tabId, change, tab) {


    //console.log("UPDATE")
    //checks if you changed current tab - change is the object of things that have changes and does not have id - id stays the same!!
    //console.log("change.url: "+ change.url)
    if (tab.active && change.url) {

        user.current_tab[0]=change.url;
        user.current_tab[1]=tab.id;


        if (String(change.url)==user.room[2] && user.in_room && (user.current_tab[1]==user.room[3] || user.room[3]=="tabid")){
            chrome.action.setPopup({popup: 'htmls/in_room_popup.html'});

            user.room[3]=tab.id;



            //console.log("onUpdated did it, room[3]: "+room[3])
            run_room_process();
        }
        else if(String(change.url).includes("https://www.youtube.com/watch")){


            chrome.action.setPopup({popup: 'htmls/watching_popup.html'});

            user.current_tab[0]=change.url;





        }
        else{
            chrome.action.setPopup({popup: 'htmls/dif_popup.html'});
        }


        update_user(user);
    }


});
```

```
//WHEN CLOSING A TAB
chrome.tabs.onRemoved.addListener (function(tabId) {


    //console.log(user.username)


    if (user.in_room){
        //console.log("current tab: "+current_tab[0]+" room tab: "+room[2])
        if (user.current_tab[0]==user.room[2]){

            console.log("OnRemoved - user went out of watching room");
            user.room_process[0]=false;




            check_connection();
            user.connection.send("exit_room,"+user.room[0]+","+user.id);

            user.in_room=false;
            user.room=["id","password","url","tabid"];
            user.room_members=[];

            update_user(user);
        }

    }

});

//WHEN NAVIGATION IS COMMITTED
chrome.webNavigation.onCommitted.addListener(function(details) {



    console.log(user.username)


    //if a room tab has been refreshed
    if(["reload", "link", "typed", "generated"].includes(details.transitionType) && details.url==user.room[2] &&details.tabId==user.room[3]){ //NEED TO FIX -> JUST WHEN RELOADING YOU SHOULD REBOT
        //content.js rebot
        console.log("REBOTING");
        clearInterval(user.room_process[1]);
        user.room_process[0]=false;

        update_user(user);

        run_room_process();
    }



});
```

```
//LISTEN TO MESSAGE OVER CONTENT SCRIPT
chrome.runtime.onMessage.addListener(function(message,sender,sendResponse){



    //in-room commands (if user is host)
    if (message.split(',')[0]=="watching_room" && user.in_room &&user.room[4]){
        console.log("in here")
        data=message.split(',');

        if(data[1]=="playing"){
            console.log("PLAYING");

            check_connection();
            user.connection.send("w.r,"+user.room[0]+","+user.id+",play,"+data[2]); //0-w.r,1-room id,2-user id,3-command,4-vid tl
        }

        else if(data[1]=="paused"){
            console.log("PAUSED");

            check_connection();
            user.connection.send("w.r,"+user.room[0]+","+user.id+",pause,"+data[2]);
        }

        else if (data[1]=="move tl"){
            console.log("MOVED TL");

            check_connection();
            user.connection.send("w.r,"+user.room[0]+","+user.id+",move tl,"+data[2]);
        }

        else if (data[1]=="exited"){

            console.log("User exited watching room - content script");

            user.room_process[0]=false;


            check_connection();
            console.log("OnContentScript - user went out of watching room");
            user.connection.send("exit_room,"+user.room[0]+","+user.id);

            user.in_room=false;
            user.room=["id","password","url","tabid"];
            user.room_members=[]

            update_user(user);


        }
    }
```

```
//everything else
else{

    //notify that the user in watching room
    if(message=="in watching room"){
        //give popup room info
        sendResponse(user.room[0]+","+user.room[1]+","+user.room_members.toString());


    }

    else if(message=="username"){
        sendResponse(user.username)
    }

    //create new watching room
    else if (message == 'create new watching room') { //need to exit old one when created new one

        check_connection();
        user.connection.send("create_room,"+user.current_tab[0]+","+user.id);

        user.connection.onmessage=function(event){
            var data=event.data;
            data=data.split(',')
            user.room[0]=data[0];
            user.room[1]=data[1];
            user.room[2]=data[2];
            user.room[3]=user.current_tab[1];
            user.room[4]=true;


            //need to add username list here
            user.room_members.push(user.username)
            console.log(user.room_members)

            user.in_room=true;

            update_user(user);

            chrome.action.setPopup({popup: 'htmls/in_room_popup.html'});
            sendResponse("^");

            run_room_process();


        }


    }

    //enters room
    else if(String(message).includes("enter room,")){ //need to exit old one when created new one
        var msg=message.split(',');
```

```
        //enters room
        else if(String(message).includes("enter room,")){ //need to exit old one when created new one
            var msg=message.split(',');

            //console.log("JOINING ROOM,"+data[1]+","+data[2])
            check_connection();
            user.connection.send("join_room,"+msg[1]+","+msg[2]+","+user.id);

            user.connection.onmessage=function(event){
                let data=event.data.split(',');
                //data=data.split(',');
                if(data[0]=="TRUE"){

                    user.room[0]=msg[1];
                    user.room[1]=msg[2];
                    user.room[2]=data[1];
                    user.room[4]=false;
                    user.in_room=true;
                    let temp=data;

                    temp.splice(0,2);
                    user.room_members=temp;

                    update_user(user);

                    //console.log("eve data "+String(event.data))
                    sendResponse(String(event.data));


                }
                else{
                    sendResponse("false id or password");
                }


            }

        }

    }
    return true; //stopping message port closing

});
```

```
//-------------------------DEFS-------------------------------

function connect(){



    user.connection = new WebSocket('ws://localhost:8765'); //ws:// 10.30.56.204:8765

    user.connection.onopen = function(e) {
        user.connection.send("get_id");
    };
    user.connection.onmessage=function(event){
        let data=event.data.split(',');

        let temp_i=data[0];
        user.id=temp_i;

        let temp_u=data[1];
        user.username=temp_u;

        update_user(user);

        console.log("connection: ",user.id,",",user.username);
    };
    //setInterval(keep_alive,10000)

}

function check_connection(){

    console.log(user.id);
    if (user.connection==null){
        reconnect();

    }
    else if (user.connection.readyState != 1 ){
        reconnect();
    }
    else{
        console.log("connection is still open");
    }

}
```

```
function reconnect(){

    user.connection = new WebSocket('ws://localhost:8765 ');

    user.connection.onopen = function(e) {
        console.log("sending: ",user.id)
        user.connection.send("reconnecting,"+user.id+","+user.username);
    };

    user.connection.onmessage=function(event){

        console.log("got data ",event.data)
        if (String(event.data).includes("new id,")){
            console.log("getting new id...");

            let temp_i=(event.data.split(','))[1]
            user.id=temp_i;

            update_user(user);

        }
        else{
            console.log(event.data);
        }

    };
}

async function get_user(){
    return new Promise(function (res, rej) {
      chrome.storage.local.get(['userLocal'], async function (result) {
          let userLocal = result.userLocal;
          res(userLocal);
      });
    });
}

function update_user(tmp_user){
    console.log("updating user")
    chrome.storage.local.get(['userLocal'], async function (result) {
        var userLocal = tmp_user;
        chrome.storage.local.set({userLocal: userLocal}, function () {}); // save it in local.

    });
}
```

```
function run_room_process(){
    if (user.room_process[0]==false){
        user.room_process[0]=true;
        const tabId=parseInt(user.room[3]);
        chrome.scripting.executeScript(
            {
            target:{tabId: tabId},
            files:["content.js"],
            }

        );
        //sending content script if user is host or not
        setTimeout(notify_content_info,1000);
        user.room_process[1]=setInterval(check_status,4500);

        user.connection.onmessage=function(event){

            var msg=String(event.data);

            console.log("got msg!!!!: ",msg);

            if(msg.includes("w.r,")){

                if (msg.includes("new_u")){
                    console.log("new member has joined")
                    msg=msg.split(','); //w.r,new_u,room_id,u1,u2,u3,u4,u5
                    msg.splice(0,3);
                    user.room_members=msg;
                    console.log("members: ",user.room_members)
                    chrome.tabs.sendMessage(user.room[3],user.room_members,function(response){

                    })
                }
                else if (msg.includes("left_u")){
                    console.log("member has left")
                    msg=msg.split(','); //w.r,left_u,room_id,u1,u2,u3,u4,u5
                    msg.splice(0,3);
                    user.room_members=msg;
                    console.log("members: ",user.room_members)
                    chrome.tabs.sendMessage(user.room[3],user.room_members,function(response){

                    })
                }
                else{
                    console.log("sending content.js"); //0-w.r,1-command,2-vid tl,3-UTC
                    chrome.tabs.sendMessage(user.room[3],msg,function(response){

                    })
                }
            }

        }
    }
    else{
        return
    }

}
```

```
function check_status(){

    //if user exited room
    if (user.room_process[0]==false){

        console.log("exiting msging");
        clearInterval(user.room_process[1]);
    }
    else{
        user.connection.send("k.a") // keep alive the connection bewtween client and server
        chrome.tabs.sendMessage(user.room[3],"k.a",function(response){

        }) ;
    }


}

function notify_content_info(){

    console.log("N I H ",user.room[4])

    chrome.tabs.sendMessage(user.room[3],String(user.room[4]+","+user.room[2]),function(response){ //{command:"W?"}
    })
}
```

content.js

```
//content.js
var video
var vidUrl
var playButton
var ishost=false
var timeline
var server_order=false

//when youtube first loading - waits until the video element has loaded
document.addEventListener('yt-navigate-finish',process);

//checks if webpgae has been loaded when re-entering the page
if (document.body) process();
else document.addEventListener('DOMContentLoaded',process)


function process(){

    video = document.querySelector('video');
    playButton=document.getElementsByClassName("ytp-play-button ytp-button")[0]
    bar=document.getElementsByClassName("ytp-progress-bar-container")[0]
    video.pause();

    //ON MESSAGE FROM background.js
    chrome.runtime.onMessage.addListener(function(message,sender,sendResponce){

        console.log("msg",message);
        if(message.includes("w.r,")){
            console.log("cmd")
            run_command(message);

            return true; //stopping message port closing
        }

        else if(message=="k.a"){
           sendResponce("^");
        }

        else{

            data=message.split(",")

            //ishost

            if(data[0]=="true"){
                console.log("t");
                ishost=true;
            }
            else if(data[0]=="false"){
                console.log("f");
                ishost=false;
            }


            //vidUrl
            vidUrl=data[1]
            console.log(vidUrl)

            return true; //stopping message port closing

        }

    });
```

```
//disable controls for non-host users
window.addEventListener('click', event => {
    if (event.target.matches('video')) {
        if(!ishost){
            event.stopPropagation();
        }
    }
}, true);

setInterval(disablecontrol,1000);



//VIDEO EVENTS

video.onplaying=function(){//if user pressed play
    console.log("cs- ",server_order,"&& ih- ",ishost);
    if (ishost && server_order==false){
        console.log("sending onplaying");
        //letting know to server to play everyone in spesific timestamp
        chrome.runtime.sendMessage("watching_room,playing,"+video.currentTime, (response) => {

        });
    }
}
video.onpause=function(){
    console.log("cs- ",server_order,"&& ih- ",ishost);
    if (ishost && server_order==false){
        console.log("sending onpause");
        //letting know to server to pause everyone in spesific timestamp
        chrome.runtime.sendMessage("watching_room,paused,"+video.currentTime, (response) => {

        });
    }
}
video.ontimeupdate=function(){
    console.log("cs- ",server_order,"&& ih- ",ishost);
    if (ishost && server_order==false){
        console.log("sending ontimeupdate");
        //if user changed timeline
        if (Math.abs(video.currentTime-timeline)>1){

            //pauseing vid ->letting know to server that time has changed->server plays in sync
            chrome.runtime.sendMessage("watching_room,move tl,"+video.currentTime, (response) => {

            });
        }

        timeline=video.currentTime;
    }


}

setInterval(is_open,1000);
```

```
    //AD BLOCKER
    let observer = new MutationObserver(mutations => { //every time a change is happend in the DOM

        console.log("OBERVER")

        var skipButton=document.getElementsByClassName("ytp-ad-skip-button");
        var unskipAdd=document.querySelector(".html5-video-player.ad-showing video");

        //checking if skip button is present
        if(skipButton!=undefined && skipButton.length>0){

            console.log("AD DETECTED - SKIPPABLE");
            skipButton[0].click();
        }
        //if there is unskippable ad
        else if(unskipAdd!=undefined){

            console.log("AD DETECTED - UNSKIPPABLE");
            unskipAdd.currentTime=10000;
        }

    });

    observer.observe(document, { childList: true, subtree: true });



}


// DISABLE/ENBALE CONTROLS FOR USERS
function disablecontrol(){

    if(!ishost && playButton.style.display!="none"){
        console.log("disabling control")
        playButton.style.display="none";
        bar.style.display="none";
    }
    else if(ishost && playButton.style.display!=""){
        console.log("allowing control")
        playButton.style.display="";
        bar.style.display="";
    }

}

//IF URL IS STILL OPEN
function is_open(){

    if (location.href!=vidUrl && vidUrl!=null){
        console.log("exiting content.js from window")
        chrome.runtime.sendMessage("watching_room,exited", (response) => {

        });
        window.close()
    }
}
```

```
// *****HELPING FUNCTIONS*****

//runs the command on spesific UTC time
function run_command(msg){

    const data=msg.split(','); //0-w.r,1-command,2-vid tl,3-UTC

    const cmd=data[1];
    const t_t_r=wait_time(data[3]);

    video.currentTime=parseInt(data[2]);
    server_order=true;

    if(cmd=="play"){
        console.log("playing at: ",t_t_r)
        setTimeout(function(){
            video.play();
            server_order=false;
        },t_t_r);
    }

    else if(cmd=="pause"){
        console.log("pausing at: ",t_t_r)
        setTimeout(function(){
            video.pause();
            server_order=false;
        },t_t_r);
    }

    else if (cmd=="move tl"){
        console.log("moving tl at: ",t_t_r)
        if(video.paused==false){
            setTimeout(function(){
                video.currentTime=parseInt(data[2]);
                video.play();
                server_order=false;
            },t_t_r);

        }
        else{
            setTimeout(function(){
                video.currentTime=parseInt(data[2]);
                server_order=false;
            },t_t_r);
        }
    }
}


function wait_time(e_time){
    const now=new Date().getTime(); //gets milisecs since Unix Epoch in 1.1.1970 - UTC
    return (parseInt(e_time)-now);
}
```

popups.js

```
var username

window.onload=function(){

    chrome.runtime.sendMessage("username", (response) => {

        let temp=response;
        username=temp;
        console.log("popups got username: ",username);

        document.getElementById("username").innerHTML=username;


    });

    //CREATE ROOM
    if (document.getElementById("createRoom")){
        document.getElementById("createRoom").addEventListener("click",function(){

            chrome.runtime.sendMessage("create new watching room", (response) => {

                window.close()
            });

        });
    }

    //ENTER ROOM
    if (document.getElementById("enterRoom")){
        document.getElementById("enterRoom").addEventListener("click",function(){

            var room_id=document.getElementById("room_id").value;
            var room_password=document.getElementById("room_password").value;


            chrome.runtime.sendMessage("enter room,"+room_id+","+room_password, (response) => {

                var data=response
                data=data.split(',')

                document.getElementById("didfind").innerHTML=data[0];

                if(data[0]=="TRUE"){
                    console.log("d1 "+data[1])
                    window.open(data[1])
                }

            });
        });
    }
```

watching_room.js

```
var room_id=""
var room_password=""

//get room details and nofity the bg that the user is in the watching room

chrome.runtime.sendMessage("in watching room", (response) => {
    let data=response.split(',');
    console.log("pre: ",data)
    document.getElementById("room_id").innerHTML=data[0];
    document.getElementById("room_password").innerHTML=data[1];

    data.splice(0,2);
    console.log("Data: ",data)
    insert_members(data);


});

chrome.runtime.onMessage.addListener(function(message,sender,sendResponse){
    console.log("got new user");

    let data=message.split(','); //w.r,new_u,room_id,u1,u2,u3,u4,u5
    console.log("before splic: ",data)
    data.splice(0,3);
    console.log("after: ",data)
    insert_members(data);
   sendResponse();

});


function insert_members(data){
    console.log("inseting members")
    let index=1;
    for (const u_name of data){

        console.log(u_name,",",index)
        if(document.getElementById(index.toString())){ // if there is allready a member there -> modifies it

            document.getElementById(index.toString()).innerHTML=u_name;
        }

        else{// else -> creates a new element
            console.log("creating new element "+index.toString())
            var para = document.createElement("p");
            para.setAttribute("id",index.toString())
            var node = document.createTextNode(u_name.toString());
            para.appendChild(node);
            var element = document.getElementById("members");
            element.appendChild(para);

        }
        index=index+1;
    }
}
```

server.py

```
#from math import comb
import websockets
import asyncio
import string
import random
import time
from random_username.generate import generate_username




#SETTING UP VARIABLES AND FUNCTIONS#

global ids
ids={"ID":"x"} #ID=8TTTF  #to delete do del ids["ID"]

global usernames
usernames={"ID":"uname"}

global rooms
rooms={"ID":(["password","url","current_time"],["x","y","z"])} #ID=6FFTF , "password=4TTTF"

global room_members
room_members={"ID":["u1","u2","u3"]}

global used_combs
used_combs=[]

def create_new_id(client):
    global ids
    new_id=generate_comb(8,True,True,True,False)
    ids[new_id]=client
    return new_id

def create_new_username(id):
    global usernames
    new_uname=generate_username()[0]

    while True:
        if new_uname not in usernames.values():
            usernames[id]=new_uname
            return new_uname


def create_new_room(soc_id,url):
    global rooms
    new_id=generate_comb(6,False,False,True,False)
    password=generate_comb(4,True,True,True,False)
    rooms[new_id]=([password,url,"current_time"],[soc_id])
    room_members[new_id]=[usernames[soc_id]]
    return new_id
```

```
def generate_comb(length,lowercase,uppercase,digits,symbols):
    global used_combs
    lst=""
    #list of all lowercase, uppercase, digits and symbols
    if lowercase:
        lst+=string.ascii_lowercase
    if uppercase:
        lst+=string.ascii_uppercase
    if digits:
        lst+=string.digits
    if symbols:
        lst+=string.punctuation
    while True:
        temp=random.sample(lst,length)
        comb="".join(temp)
        if comb not in used_combs:
            used_combs.append(comb)
            return comb

#returns time since Unix Epoch in 1.1.1970 - UTC
def cmd_utc():
    delay=500
    return (int(round(time.time() * 1000)+delay)) #rounding to ms

async def broadcast(msg):
    global rooms

    if "new_u" in msg:

        data=msg.split(',') #w.r,new_u,room_id,u1,u2,u3,u4,u5
        for uId in (rooms[data[2]][1]):
            try:
                await ids[uId].send(msg)
            except Exception as e :
                print("exe:")
                print(e)

    elif "left_u" in msg:

        data=msg.split(',') #w.r,left_u,room_id,u1
        for uId in (rooms[data[2]][1]):
            try:
                await ids[uId].send(msg)
            except Exception as e :
                print("exe:")
                print(e)

    else:
        data=msg.split(',') #0-w.r,1-room id,2-user id,3-command,4-vid tl
        rooms[data[1]][0][2]=data[4]#setting current time in watching room
        #print(f"set current time in room: {data[1]} to : {rooms[data[1]][0][2]}")
        utc=cmd_utc()

        for uId in (rooms[data[1]][1]):
            try:
                await ids[uId].send((str(data[0])+","+str(data[3])+","+str(data[4])+","+str(utc))) #0-w.r,1-command,2-vid tl,3-UTC
            except Exception as e :
                print("exe:")
                print(e)
```

```
async def listen(websocket,path):
    global rooms
    global ids
    global used_combs

    try:
        async for message in websocket:

            if ("w.r" in message):

                if("k.a" in message): #keep connection alive
                    data=message.split(",") #w.r,k.a,uId,rId
                    re=data[2] in rooms[data[3]][1]
                    await websocket.send(str(re))

                else:
                    await broadcast(message) #maybe needs await



            else:

                if (message=="get_id"):

                    soc_id=(create_new_id(websocket))
                    u_name=create_new_username(soc_id)
                    print(F"NEW USER: {soc_id} - {u_name}")
                    await websocket.send(soc_id+","+u_name)

                elif("reconnecting" in message):

                    data=message.split(',')
                    print(data)
                    check_id=data[1]
                    username=data[2]
                    print(f"{data[1]} wants to recconect")
                    if check_id in ids:
                        print("OK")
                        ids[check_id]=websocket
                        usernames[check_id]=username
                        print(f"reconnecting {data[1]} by id")
                        await websocket.send("reconnected you by id")

                    else:
                        new_id=(create_new_id(websocket))
                        ids[new_id]=websocket
                        usernames[new_id]=username
                        print(f"gave {data[1]} new id -> {new_id}")
                        await websocket.send("new_id,"+new_id)

                        #await websocket.send(soc_id)

                if ("create_room," in message):

                    data=message.split(',')
                    url=data[1]
                    soc_id=data[-1]
                    room_id=create_new_room(soc_id,url)
                    print(f"room {room_id} was created by {soc_id}")
                    await websocket.send(room_id+","+rooms[room_id][0][0]+","+rooms[room_id][0][1]) #id,password
```

```
            if ("join_room," in message):
                data=message.split(',')
                room_id=data[1]
                room_password=data[2]
                soc_id=data[-1]
                try:
                    if (rooms[room_id][0][0]==room_password):

                        print(f"{soc_id} has joined room {room_id}")
                        room_members[room_id].append(usernames[soc_id])

                        str_members=','.join(room_members[room_id])
                        await broadcast(f"w.r,new_u,{room_id},{str_members}")

                        rooms[room_id][1].append(soc_id)

                        print(rooms)

                        await websocket.send(f"TRUE,{rooms[room_id][0][1]},{str_members}") #TRUE,url,u1,u2,u3...

                        utc=cmd_utc()
                        await websocket.send("w.r,move tl,"+rooms[room_id][0][2]+","+utc) #0-w.r,1-command,2-vid tl,3-UTC
                    else:
                        await websocket.send("FALSE")
                except:
                    await websocket.send("FALSE")

            if ("exit_room" in message):

                data=message.split(',')
                room_id=data[1]
                soc_id=data[-1]

                print(f"user: {soc_id} exited room {room_id}")

                if (rooms[room_id][1].index(soc_id)==0):
                    nxt_host=rooms[room_id][1][1]
                    print(f"assigning new host to room {room_id}")
                    await ids[nxt_host].send("host")


                rooms[room_id][1].remove(soc_id)
                str_members=','.join(room_members[room_id])
                room_members[room_id].remove(usernames[soc_id])

                await broadcast(f"w.r,left_u,{room_id},{str_members}") #w.r,left_u,room_id,u1

                #checks if room is empty
                if not (rooms[room_id][1]):
                    # need to delete room id and password from used combs

                    #delete room
                    del rooms[room_id]

                print(rooms)

            else:
                pass
                #print ("Received and echoing message: "+message)
            #await websocket.send(message)

    except websockets.exceptions.ConnectionClosed:
        print("A CLIENT HAS DISSCONECTED")
```

```
start_server = websockets.serve(listen, "0.0.0.0", 8765)

print("WebSockets echo server starting") #FLUSH=TRUE
asyncio.get_event_loop().run_until_complete(start_server)

print("WebSockets echo server running")
asyncio.get_event_loop().run_forever()
```

https://github.com/OakTZ/WeWatch.git

להלן קישור ל-GitHub שבוא קיים כל הקוד הסופי והמעודכן ביותר