

אפיון פרויקט - EchoRTC

מגיש: אורי כהן

כיתה: יב 2

בית ספר: הנדסאים



תוכן עניינים

תקציר מנהלים.....	6
פתיחה ותיאור כללי.....	8
הצעת הפרויקט המקורית.....	9
אפיון הפרויקט.....	10
הגדרה.....	10
קהל יעד.....	10
מטרות הפרויקט.....	11
מטרות עיקריות.....	11
מטרות מתקדמות.....	11
יעדי הפרויקט.....	11
תחום הפרויקט (תיחום).....	12
תחומים הכלולים בפרויקט.....	12
תחומים שאינם כלולים בפרויקט.....	12
חקר מוצרים.....	13
חקר פיתוחי.....	14
סביבה.....	16
חומרה.....	16
המערכת פותחה והורצה על מחשב אישי סטנדרטי, הכולל:.....	16
מערכות הפעלה.....	16
תוכנה וכלי פיתוח.....	16
סקירת טכנולוגיות הפרויקט.....	17
WebRTC.....	17
מאפשר: WebRTC השימוש ב-.....	17
צד לקוח – JavaScript.....	17
ממומשים: JavaScript באמצעות.....	17
צד שרת – Python (Signaling).....	18
בלבד. השרת Signaling ומשמש כשרת Python צד השרת פותח בשפת.....	18
אחראי על:.....	18
WebSocket - WSS.....	18
מועברות: WebSocket באמצעות.....	18

Machine Learning ועיבוד וידאו.....	18
מערכת עיבוד הווידאו מבוססת על אלגוריתמים חישוביים שפותחו מאפס, ללא שימוש בספריות חיצוניות לעיבוד תמונה או למידת מכונה. המימוש מתבסס על:	18
מסד נתונים.....	19
מסד הנתונים בשרת משמש, בין היתר, ל:	19
תיאור מפורט של המערכת.....	20
ארכיטקטורה כללית של המערכת.....	20
רכיבי המערכת המרכזיים.....	20
Client – Browser) צד לקוח.....	20
Signaling Server – צד שרת.....	21
מסד נתונים בצד השרת.....	21
Mesh Peer-to-Peer ארכיטקטורת.....	22
לדוגמה:.....	22
יתרונות גישה זו:.....	22
חסרונות:.....	22
זרימת המידע במערכת.....	23
1. ערוץ איתות (Signaling).....	23
2. ערוץ מדיה (Media).....	23
Machine Learning) שילוב תשתית עיבוד וידאו.....	23
תשתית עיבוד הווידאו פועלת בצד הלקוח, כחלק מצינור עיבוד המדיה Peers ל-MediaStream המקומי. עיבוד זה מתבצע לפני שליחת ה- אחרים, ומאפשר:	23
פרוטוקולי התקשורת במערכת.....	24
הפרוטוקולים במערכת נחלקים לשתי קבוצות עיקריות:	24
עקרונות פעולה כלליים – WebRTC.....	24
עומדים שלושה רכיבים עיקריים: WebRTC בבסיס.....	24
RTCPeerConnection.....	24
כוללים: RTCPeerConnection תפקידי.....	24
SDP – Session Description Protocol.....	25
הוא פורמט טקסטואלי המתאר את מאפייני החיבור בין שני SDP משמש לתיאור: WebRTC, SDP משתתפים. במסגרת.....	25
Offer / Answer תהליך החיבור מתבצע באמצעות מנגנון.....	25
ICE – Interactive Connectivity Establishment.....	25
אחראי על: ICE.....	25

STUN – Session Traversal Utilities for NAT.....	25
TURN – Traversal Using Relays around NAT.....	26
WebSocket Secure (WSS) – Signaling.....	26
WebSocket Secure - עושה שימוש בפרוטוקול Signaling שרת ה- ...לצורך החלפת הודעות בין הלקוחות. ערוץ זה משמש להעברת: WSS	26
הפרדה בין ערוצי תקשורת.....	26
אחד העקרונות המרכזיים במערכת הוא ההפרדה המלאה בין:	26
הפרדה זו:.....	26
בזרימת התקשורת Machine Learning שילוב עיבוד וידאו מבוסס	27
הפרדה זו מאפשרת:.....	27
סיכום הפרק.....	27
אבטחת מידע והצפנה במערכת.....	28
(Signaling) אבטחת שכבת האיתות.....	28
אמצעי האבטחה בשכבת האיתות כוללים:	28
DTLS-SRTP – הצפנת מדיה.....	29
WebRTC העברת המדיה (וידאו וקול) במערכת מתבצעת באמצעות תוך שימוש במנגנוני הצפנה מובנים: Peer-to-Peer,	29
תהליך זה מבטיח:.....	29
הפרדת תפקידים וצמצום משטח התקיפה.....	29
במערכת קיימת הפרדה ברורה בין רכיבי המערכת:	29
הפרדה זו:.....	29
STUN ו-TURN אבטחת תקשורת מול שרתי	30
TURN: גם כאשר נעשה שימוש ב-	30
פרטיות משתמשים.....	30
שמירה על פרטיות המשתמשים הייתה שיקול מרכזי בתכנון המערכת.	30
בהתאם לכך:.....	30
סיכום אבטחת המידע.....	30
אלגוריתמים מרכזיים במערכת.....	31
אלגוריתם יצירת חדר וניהול תפקידים.....	31
בעת יצירת חדר חדש במערכת, מתבצע תהליך לוגי הכולל:	31
אלגוריתם הצטרפות משתמש לחדר קיים.....	31
כאשר משתמש חדש מצטרף לחדר פעיל:	31
Mesh Peer-to-Peer אלגוריתם יצירת.....	32
ICE Candidates אלגוריתם החלפת.....	32
במהלך יצירת החיבור ולאחריו, כל משתתף:	32

אלגוריתם ניהול מדיה בזמן אמת.....	32
לאחר יצירת החיבורים, מתבצע ניהול שוטף של זרמי המדיה:.....	32
אלגוריתם ניתוק והתאוששות מחיבור שנפל.....	33
במקרה של ניתוק משתתף או נפילת חיבור:.....	33
אלגוריתם שילוב עיבוד וידאו בצד הלקוח.....	33
אחרים, ניתן לשלב Peers ל-MediaStream בצד הלקוח, לפני שליחת ה- שלב עיבוד מקדים:.....	33
סיכום האלגוריתמים.....	33
תיאור פונקציונלי של המערכת.....	34
ניהול משתמשים.....	34
הרשמה (Sign Up) תהליך הרשמה.....	34
בעת הרשמה למערכת, מתבצע התהליך הבא:.....	34
מנגנון זה נועד:.....	34
תהליך התחברות (Login).....	34
לאחר הרשמה ואימות מוצלח, המשתמש יכול להתחבר למערכת:.....	34
ניהול חדרים.....	35
יצירת חדר.....	35
הצטרפות לחדר.....	35
הרשאות ותפקידים.....	35
Host.....	35
Participant.....	35
אבטחת מידע בתהליכי משתמשים.....	36
סיכום התיאור הפונקציונלי.....	36
(User Interface) ממשק משתמש.....	37
מסך כניסה (Login).....	37
מסך הכניסה הוא נקודת ההתחלה של המערכת. במסך זה המשתמש מתבקש להזין:.....	37
מסך הרשמה (Sign Up).....	37
מסך ההרשמה מאפשר יצירת חשבון משתמש חדש במערכת. המשתמש מזין:.....	37
מסך ראשי / לובי.....	38
ממנו (Lobby), לאחר התחברות מוצלחת, המשתמש מועבר למסך הראשי ניתן:.....	38
מסך חדר (Room View).....	38
תצוגת וידאו.....	38

בקרי מדיה.....	38
צ'אט טקסטואלי.....	38
לצד שיחת הווידאו, קיים צ'אט טקסטואלי בזמן אמת, המאפשר:	38
Host – מסך ניהול.....	39
מוצג ממשק ניהול נוסף, הכולל: Host למשתמש המוגדר כ-	39
חווית משתמש ושיקולי עיצוב.....	39
בעת תכנון הממשק הושם דגש על:	39
סיכום ממשק המשתמש.....	39
Design Detailed – מפרט תכנון מפורט.....	40
המערכת בנויה בגישה מודולרית, תוך הפרדה ברורה בין:	40
מבנה כללי של הפרויקט.....	40
הפרויקט מחולק למספר תיקיות וקבצים עיקריים, כאשר כל רכיב אחראי על	
שכבה לוגית שונה במערכת:	40
Client – צד לקוח.....	41
קבצי HTML.....	41
קבצי CSS.....	41
WebRTC לוגיקת – JavaScript קבצי.....	41
ניהול חיבורים מרובים.....	42
גישה זו מאפשרת:	42
Signaling Server – צד שרת.....	42
תפקידים מרכזיים של השרת.....	42
מבנה קוד השרת.....	42
מסד נתונים.....	43
המידע הנשמר כולל:	43
(Event Handling) טיפול באירועים.....	43
לצורך: (Events) המערכת עושה שימוש באירועים	43
(Machine Learning) שילוב של עיבוד וידאו.....	43
תכנון זה מאפשר:	43
Design Detailed סיכום.....	44
עיצוב ואלגוריתמים – Machine Learning תשתית.....	45
עקרונות תכנון כלליים.....	45
בפרויקט פותחה בהתאם לעקרונות Machine Learning תשתית ה-	
הבאים:	45
בארכיטקטורה ML מיקום תשתית ה-	45
מבנה זה מאפשר:	45

קליטת פריימים ועיבוד בסיסי	46
מתבצעות פעולות חישוביות כגון: NumPy באמצעות	46
זיהוי ומעקב אחר תווי פנים – גישה אלגוריתמית	46
האלגוריתם מבצע:	46
פילטרים חישוביים בזמן אמת	46
על בסיס ניתוח התמונה ניתן להפעיל פילטרים חישוביים שונים, כגון:	46
שיקולי ביצועים	47
עיבוד וידאו בזמן אמת מחייב התייחסות לביצועים:	47
מגבלות המערכת	47
בפרויקט קיימות מגבלות ידועות: Machine Learning לתשתית ה-	47
הרחבות עתידיות	47
התשתית תוכננה כך שתאפשר בעתיד:	47
Machine Learning סיכום תשתית ה-	47
בדיקות (Testing)	48
בדיקות צד לקוח (Client Testing)	48
בדיקות חיבור מצלמה ומיקרופון	48
WebRTC בדיקות יצירת חיבור	48
בדיקות בקרי מדיה	48
בדיקות צד שרת (Server Testing)	49
Signaling בדיקות	49
בדיקות ניהול חדרים	49
בדיקות ניהול משתמשים ואימות	49
בדיקות אבטחת מידע	49
בדיקות אלו בוצעו באמצעות:	49
בדיקות תרחישי קצה (Edge Cases)	50
Machine Learning בדיקות תשתית	50
סיכום הבדיקות	50
הוראות התקנה והפעלה	51
דרישות מערכת	51
צד לקוח	51
צד שרת	51
(Signaling Server) התקנת צד שרת	51
הפעלת צד לקוח	52
הפעלת המערכת בסביבה מאובטחת	52
תקלות נפוצות ופתרון	52

מדריך למשתמש.....	53
הרשמה למערכת.....	53
התחברות למערכת.....	53
יצירת חדר חדש.....	53
הצטרפות לחדר קיים.....	53
שימוש במהלך שיחה.....	54
(Host) ניהול חדר.....	54
סיום שיחה.....	54
סיכום מדריך המשתמש.....	54
סיכום ומשוב.....	55

תקציר מנהלים

פרויקט זה עוסק בפיתוח מערכת WebRTC מתקדמת לשיחות וידאו, קול ושיתוף מסך מאובטחות, המיועדת לתקשורת בזמן אמת בין מספר משתתפים במקביל. המערכת מבוססת על ארכיטקטורת Peer-to-Peer מרובת לקוחות (Mesh), בה כל משתתף יוצר חיבור ישיר ומוצפן מול כל משתתף אחר בחדר, ללא שימוש בשרת מדיה מרכזי.

המערכת פועלת מתוך דפדפן אינטרנט סטנדרטי, ללא צורך בהתקנת תוכנה חיצונית, ומאפשרת יצירת חדרי שיחה מאובטחים, הצטרפות משתמשים מרובים, שיחות וידאו וקול בזמן אמת, שיתוף מסך, צ'אט טקסטואלי וניהול הרשאות מתקדם באמצעות תפקיד מארח (Host). תפקיד זה נוצר אוטומטית עם יצירת החדר ומאפשר שליטה על המשתתפים והפעולות המותרות במהלך השיחה.

בחירת הפרויקט נבעה מהרצון להעמיק בהבנת תחום התקשורת בזמן אמת, בדגש על פרוטוקולי WebRTC, רשתות מחשבים, אבטחת מידע והתמודדות עם מגבלות רשת כגון NAT ו-Firewalls. הפרויקט משלב ידע תיאורטי ומעשי בתחומים אלו, ומממש פתרון מלא מקצה לקצה הכולל צד לקוח וצד שרת.

צד הלקוח פותח בשפת JavaScript ומנצל את ממשקי ה-API WebRTC, ובפרט את RTCPeerConnection ו-MediaStreams, לצורך יצירת חיבורי מדיה מוצפנים והעברת וידאו וקול ישירות בין המשתתפים. צד השרת פותח בשפת Python ומשמש כשרת Signaling ייעודי המבוסס על WebSocket מאובטח (WSS), שתפקידו לנהל את תהליך האיתות בלבד – החלפת הודעות SDP ו-ICE Candidates – מבלי לעבד או להעביר מדיה בפועל.

לצורך התמודדות עם מגבלות רשת ותרמישים של חסימת תקשורת ישירה, המערכת עושה שימוש בפרוטוקולי ICE, STUN ו-TURN, תוך הסתמכות על שרתי TURN חיצוניים במקרה הצורך. כל תעבורת המדיה במערכת מוצפנת מקצה לקצה באמצעות מנגנוני DTLS-SRTP המובנים ב-WebRTC, דבר המבטיח רמת אבטחה ופרטיות גבוהה במיוחד.

בנוסף ליכולות התקשורת בזמן אמת, הפרויקט כולל תכנון ומימוש של תת-מערכת ייעודית לעיבוד וידאו מתקדם בצד הלקוח, המבוססת על אלגוריתמים של Machine Learning. תת-מערכת זו מיועדת לאפשר הוספת פילטרים חכמים למצלמה בזמן אמת, כגון פילטרים המבוססים על זיהוי ומעקב אחר תווי פנים.

מערכת ה-Machine Learning פותחה בגישת Low-Level, ללא שימוש בספריות עיבוד תמונה או למידת מכונה מוכנות (כגון TensorFlow, OpenCV או PyTorch), ותוך הסתמכות על ספריית NumPy בלבד לצורך חישובים מתמטיים ואלגוריתמיים. גישה זו נבחרה במכוון על מנת להבין לעומק את עקרונות הפעולה של עיבוד תמונה, זיהוי תבניות וזרימת נתונים במודלים חישוביים, ולא כהטמעה "שחורה" של ספריות חיצוניות.

כחלק מתשתית זו, נבנה מסד נתונים ייעודי שנוצר במיוחד עבור מערכת ה-Machine Learning, ומשמש לאחסון נתוני אימון, תיוגים, פרמטרים ומידע נלווה הדרוש לפיתוח ולבדיקה של האלגוריתמים. מסד נתונים זה מאפשר שליטה מלאה בנתונים, התאמה עתידית למודלים נוספים והרחבה הדרגתית של יכולות המערכת.

מערכת הפילטרים תוכננה להשתלב בעתיד ישירות בצינור עיבוד הווידאו של WebRTC בצד הלקוח, כך שעיבוד התמונה יתבצע לפני שליחת ה-MediaStream למשתתפים האחרים, ללא פגיעה בפרטיות וללא העברת מידע חזותי לשרת מרכזי.

המערכת כולה נבנתה בארכיטקטורה מודולרית, המאפשרת הפרדה ברורה בין שכבות התקשורת, האבטחה, ממשק המשתמש ועיבוד הווידאו. פרויקט זה מדגים מימוש מלא של מערכת תקשורת בזמן אמת, תוך שליטה בפרוטוקולים, באבטחת המידע, בניהול חיבורים מרובים ובתשתית Machine Learning עצמאית, ומהווה בסיס יציב להמשך פיתוח, הרחבה ושימוש מעשי במערכות תקשורת מבוזרות מתקדמות.

פתיחה ותיאור כללי

בעשור האחרון חלה עלייה משמעותית בשימוש במערכות תקשורת בזמן אמת מבוססות וידאו, הן לצרכים אישיים והן לצרכים מקצועיים וארגוניים. מערכות אלו נדרשות לספק איכות מדיה גבוהה, זמן השהיה נמוך, רמת אבטחה גבוהה ויכולת עבודה בסביבות רשת מגוונות, הכוללות מגבלות כגון NAT, Firewalls ורוחב פס משתנה.

פרויקט זה עוסק בפיתוח מערכת WebRTC לשיחות וידאו, קול ושיתוף מסך מאובטחות, הפועלת בארכיטקטורת Peer-to-Peer מרובת משתתפים (Mesh). המערכת נבנתה מתוך מטרה להבין לעומק את מנגנוני הפעולה של WebRTC ברמת הפרוטוקולים, ללא הסתמכות על ספריות עזר חיצוניות או שירותי מדיה מוכנים, ותוך שליטה מלאה בכל שכבות המערכת – החל משכבת הרשת והאיתות ועד לעיבוד המדיה בצד הלקוח.

בנוסף ליכולות התקשורת בזמן אמת, המערכת תוכננה כך שתאפשר הרחבה עתידית בתחום עיבוד הווידאו בצד הלקוח. כחלק מתכנון זה, פותחה תשתית ראשונית למערכת פילטרים למצלמה המבוססת על Machine Learning, הכוללת מנגנון זיהוי ומעקב אחר תווי פנים, אשר תשולב בעתיד במערכת ה-WebRTC עצמה.

מערכת הפילטרים פותחה בגישת "מאפס", ללא שימוש בספריות עיבוד תמונה או למידת מכונה מוכנות (כגון OpenCV או frameworks ייעודיים), ותוך הסתמכות בסיסית בלבד על ספריית NumPy לצורך חישובים מתמטיים. בנוסף, נבנה מסד

נתונים ייעודי שנוצר במיוחד עבור מערכת זו, לצורך ניהול נתוני אימון, תיוגים ונתוני עזר, כחלק מהיערכות להרחבת יכולות ה-Machine Learning בפרויקט.

הצעת הפרויקט המקורית

הצעת הפרויקט המקורית כללה פיתוח מערכת תקשורת בזמן אמת מבוססת WebRTC, שתאפשר שיחות וידאו וקול בין מספר משתתפים במקביל, ללא תלות בשרת מדיה מרכזי. המערכת תוכננה לפעול מתוך דפדפן אינטרנט, תוך שימוש בפרוטוקולים סטנדרטיים ופתוחים, ובדגש על אבטחת מידע ופרטיות המשתמשים.

לשם כך, הוגדרו המרכיבים המרכזיים הבאים:

- צד לקוח מבוסס JavaScript, האחראי על יצירת חיבורי RTCPeerConnection, ניהול זרמי מדיה (וידאו, קול ושיתוף מסך) והצגת ממשק משתמש אינטראקטיבי.
- צד שרת מבוסס Python, המשמש כשרת Signaling בלבד, באמצעות WebSocket מאובטח (WSS), לצורך תיאום החיבורים והחלפת הודעות SDP ו-ICE Candidates.
- שימוש בפרוטוקולי ICE, STUN ו-TURN לצורך התמודדות עם מגבלות רשת ומעבר NAT.
- הצפנת כלל תעבורת המדיה באמצעות מנגנוני DTLS-SRTP המובנים ב-WebRTC.
- ניהול משתמשים, הרשאות וחדרים, כולל תפקיד Host עם יכולות שליטה.

בנוסף, כבר בשלב ההצעה הוגדרה מטרה עתידית לשלב במערכת יכולות עיבוד וידאו מתקדמות בצד הלקוח, ובפרט פילטרים חכמים המבוססים על ניתוח תמונה

ו-Machine Learning. לשם כך, הוחלט לפתח תשתית עצמאית לזיהוי תווי פנים, אשר לא תישען על ספריות חיצוניות כבדות, אלא תתבסס על מימוש אלגוריתמי עצמאי ונתוני אימון שנבנו במיוחד עבור הפרויקט.

בחירה זו נועדה להעמיק את ההבנה בתחום עיבוד התמונה ולמידת המכונה, ולאפשר שליטה מלאה בלוגיקה, בנתונים ובאופן האינטגרציה העתידי עם מערכת ה-WebRTC.

אפיון הפרויקט

הגדרה

הפרויקט מוגדר כמערכת תקשורת בזמן אמת מבוססת WebRTC, המאפשרת שיחות וידאו, קול ושיתוף מסך מאובטחות בין מספר משתתפים במקביל, תוך שימוש בארכיטקטורת Peer-to-Peer מרובת לקוחות (Mesh). המערכת פועלת מתוך דפדפן אינטרנט ואינה דורשת התקנה של תוכנה חיצונית, שרת מדיה מרכזי או תשתית קניינית.

בנוסף למערכת התקשורת, הפרויקט כולל תת-מערכת עצמאית לעיבוד וידאו מבוססת Machine Learning, המיועדת להוספת פילטרים חכמים למצלמה בזמן אמת. תת-מערכת זו פותחה בגישה אלגוריתמית נמוכה (Low-Level), ללא שימוש בספריות עזר חיצוניות לעיבוד תמונה או למידת מכונה, ומתבססת על מימוש עצמאי ו-NumPy בלבד.

שילוב שתי המערכות נועד להדגים שליטה מלאה הן בתחום התקשורת בזמן אמת והן בתחום עיבוד הווידאו החישובי, תוך שמירה על אבטחה, פרטיות ויכולת הרחבה עתידית.

קהל יעד

המערכת מיועדת למגוון רחב של משתמשי קצה וגורמים טכנולוגיים, ובהם:

- משתמשים פרטיים המעוניינים בשיחות וידאו מאובטחות ללא תלות בפלטפורמות מסחריות
- צוותי עבודה מרוחקים הזקוקים לתקשורת בזמן אמת עם שליטה גבוהה בפרטיות
- מערכות לימוד מרחוק ותרגול טכנולוגי
- מפתחים וחוקרים המעוניינים ללמוד ולהתנסות ב-WebRTC ברמת הפרוטוקולים
- גורמים המעוניינים במערכות עיבוד וידאו חכמות בצד הלקוח, ללא העברת מדיה לשרת

המערכת אינה מיועדת לשימוש מסחרי רחב היקף (כגון אלפי משתמשים במקביל), אלא לפתרונות מבוקרים, לימודיים ומודולריים, בהם נדרשת שליטה מלאה בקוד ובתשתית.

מטרות הפרויקט

מטרות עיקריות

- מימוש מערכת WebRTC מלאה לשיחות וידאו וקול מרובות משתתפים
- הבנה ומימוש של ארכיטקטורת Mesh Peer-to-Peer
- הקמת שרת Signaling עצמאי ב-Python מבוסס WebSocket
- שימוש בפרוטוקולי ICE, STUN ו-TURN להתמודדות עם NAT ו-Firewalls
- הבטחת הצפנה מקצה לקצה של תעבורת המדיה (DTLS-SRTP)
- פיתוח ממשק משתמש אינטואיטיבי מבוסס דפדפן

מטרות מתקדמות

- פיתוח תשתית Machine Learning לעיבוד וידאו בזמן אמת בצד הלקוח
- מימוש אלגוריתמים לזיהוי ומעקב אחר תווי פנים ללא ספריות עזר חיצוניות
- בניית מסד נתונים עצמאי לנתוני אימון, תיוג ופרמטרים אלגוריתמיים
- הכנת תשתית לאינטגרציה עתידית בין מערכת ה-ML ל-MediaStream של WebRTC
- שמירה על פרטיות מלאה – עיבוד הווידאו מתבצע מקומית בלבד

יעדי הפרויקט

יעדי הפרויקט הוגדרו כך שיהיו מדידים וברורים:

- יצירת חיבור WebRTC מוצפן בין כל זוג משתתפים בחדר
- ניהול מספר חיבורי RTCPeerConnection במקביל לכל משתמש
- תמיכה בשיחות וידאו, קול ושיתוף מסך
- זמני השהיה (Latency) נמוכים המאפשרים שיחה טבעית
- תפקוד תקין גם בתרחישים של חסימת P2P באמצעות TURN
- זיהוי תווי פנים בסיסי במצלמה באמצעות אלגוריתמים חישוביים
- אחסון וניהול נתוני ML במסד נתונים ייעודי שנבנה בפרויקט
- שמירה על קוד מודולרי, קריא וניתן להרחבה

תחום הפרויקט (תיחום)

תחומים הכלולים בפרויקט

- תקשורת רשת בזמן אמת
- WebRTC ו-Peer-to-Peer
- אבטחת מידע והצפנה
- Signaling ו-WebSocket
- עיבוד וידאו בצד הלקוח
- אלגוריתמים של Machine Learning
- ניהול נתונים ומסדי נתונים

תחומים שאינם כלולים בפרויקט

- עיבוד מדיה בצד שרת
- הקלטת שיחות וידאו בשרת
- שימוש בשרתי מדיה מסוג SFU / MCU
- שימוש ב-Frameworks חיצוניים ללמידת מכונה
- ניתוח Big Data או אימון מודלים כבדים בזמן אמת

חקר מוצרים

שוק מערכות התקשורת בזמן אמת כולל מגוון רחב של פתרונות מסחריים וקוד פתוח, המאפשרים שיחות וידאו, קול ושיתוף מסך בין משתמשים. בין המערכות הנפוצות ניתן למנות את Zoom, Google Meet, Microsoft Teams ומערכות נוספות, אשר משמשות מיליוני משתמשים ברחבי העולם.

מערכות אלו מציעות חוויית משתמש נוחה ויציבה, אך מתבססות לרוב על תשתיות שרת מורכבות, הכוללות שרתי מדיה מרכזיים (SFU/MCU), ניתוח ועיבוד תעבורת מדיה בצד שרת, ולעיתים אחסון מידע רגיש בתשתיות צד שלישי. כתוצאה מכך, המשתמש מאבד שליטה מלאה על נתוני המדיה, על אופן הצפנתם ועל מסלול העברתם.

בנוסף, מרבית המערכות המסחריות אינן מאפשרות למפתח להבין או לשלוט לעומק בפרוטוקולים וברכיבי התקשורת הפנימיים, שכן מדובר במערכות סגורות או מוסתרות מאחורי שכבות abstraction רבות.

הפרויקט הנוכחי שואף להציע גישה שונה: מערכת תקשורת בזמן אמת המבוססת על תקנים פתוחים, ללא שרת מדיה מרכזי, ותוך שליטה מלאה בקוד, בארכיטקטורה ובפרוטוקולי התקשורת. הבחירה ב-WebRTC נעשתה מתוך

הבנה כי מדובר בטכנולוגיה סטנדרטית, נתמכת דפדפנים, המאפשרת תקשורת מוצפנת בזמן אמת ללא צורך בהתקנת תוכנה חיצונית.

בניגוד לפתרונות קיימים, המערכת תוכננה כך שעיבוד המדיה, לרבות עיבוד וידאו מתקדם מבוסס Machine Learning, יתבצע בצד הלקוח בלבד. גישה זו מגדילה את רמת הפרטיות, מפחיתה תלות בתשתיות חיצוניות ומאפשרת שילוב אלגוריתמים חישוביים ייעודיים, אשר אינם זמינים במערכות מסחריות סטנדרטיות.

חקר פיתוחי

פיתוח מערכת תקשורת בזמן אמת מבוססת WebRTC מציב שורה של אתגרים טכנולוגיים והנדסיים, הנוגעים הן לתחום הרשתות והן לתחום עיבוד המדיה. בפרויקט זה נדרש שילוב של ידע תיאורטי עם מימוש מעשי, תוך התמודדות עם מגבלות מובנות של תקשורת מבוזרת, אבטחת מידע וביצועים בזמן אמת.

האתגר המרכזי הראשון נוגע לעצם מימוש תקשורת Peer-to-Peer בין מספר משתתפים במקביל. בניגוד לארכיטקטורות מרכזיות, בהן שרת מדיה מנהל את זרימת הווידאו והקול, ארכיטקטורת Mesh מחייבת כל לקוח לנהל מספר חיבורי תקשורת בו-זמנית. משמעות הדבר היא ניהול מספר אובייקטי RTCPeerConnection, טיפול באירועי חיבור, ניתוק והתאוששות, וכן סנכרון מצב בין כל המשתתפים בחדר.

אתגר נוסף נוגע להתמודדות עם מגבלות רשת כגון NAT ו-Firewalls. תקשורת Peer-to-Peer ישירה אינה תמיד אפשרית, ולכן נדרש שימוש בפרוטוקולים ייעודיים המאפשרים גילוי נתיבי תקשורת חלופיים. בפרויקט זה נבחר שימוש במנגנון ICE, המשלב שרתי STUN לצורך גילוי כתובת ציבורית ושרתי TURN

לתיווך במקרים של חסימה מלאה. שילוב זה מאפשר יצירת חיבור יציב גם בסביבות רשת מורכבות.

נושא אבטחת המידע מהווה אתגר מהותי נוסף. מכיוון שהמערכת עוסקת בהעברת מדיה בזמן אמת, נדרש להבטיח הצפנה מלאה של תעבורת הווידאו והקול, ללא חשיפת נתונים לשרת המרכזי. WebRTC מספק מנגנוני הצפנה מובנים באמצעות DTLS-SRTP, אך מימוש נכון מחייב הבנה של תהליך יצירת המפתחות, החלפת ה-SDP וניהול חיבור מאובטח לכל זוג משתתפים.

מעבר לאתגרי התקשורת, הפרויקט כולל גם אתגרי פיתוח בתחום עיבוד הווידאו ולמידת המכונה. פיתוח מערכת פילטרים למצלמה בזמן אמת, ללא שימוש בספריות עזר חיצוניות, מחייב מימוש עצמאי של אלגוריתמים לעיבוד תמונה, זיהוי תבניות ומעקב אחר תווי פנים. בנוסף, יש לקחת בחשבון מגבלות ביצועים, שכן עיבוד הווידאו מתבצע בזמן אמת ובצד הלקוח, במקביל לניהול חיבורי WebRTC.

אתגר משמעותי נוסף הוא בניית מסד נתונים עצמאי עבור מערכת ה-Machine Learning. מסד נתונים זה נדרש לאחסן נתוני אימון, תיוגים ופרמטרים אלגוריתמיים, ולאפשר שליטה מלאה בתהליך הלמידה והבדיקה. הבחירה לבנות מסד נתונים ייעודי, ולא להסתמך על מאגרי נתונים חיצוניים, נועדה להעמיק את ההבנה בתהליכי למידה ולשמור על התאמה מלאה לצרכי הפרויקט.

לבסוף, נדרש תכנון מוקפד של מבנה המערכת והקשר בין רכיביה. שילוב של WebRTC, Signaling, שרת מידע ותשתית Machine Learning מחייב ארכיטקטורה מודולרית, הפרדה ברורה בין שכבות, ויכולת הרחבה עתידית ללא פגיעה ביציבות המערכת. תכנון זה נועד לאפשר הוספת יכולות מתקדמות בהמשך, כגון פילטרים נוספים, אלגוריתמים חכמים יותר ותמיכה בתרחישים מורכבים יותר.

סביבה

פיתוח המערכת בוצע בסביבת עבודה מבוקרת, המאפשרת בדיקה, ניפוי שגיאות והרחבה עתידית של רכיבי המערכת. הסביבה נבחרה כך שתתמוך הן בפיתוח צד הלקוח והן בפיתוח צד השרת, תוך התאמה לעבודה עם טכנולוגיות Web ואלגוריתמים חישוביים.

חומרה

המערכת פותחה והורצה על מחשב אישי סטנדרטי, הכולל:

- מעבד רב-ליבות
- זיכרון RAM המספיק להרצת דפדפן, שרת Python ועיבוד וידאו מקבילי
- מצלמת רשת לצורך בדיקות וידאו בזמן אמת

הפרויקט אינו דורש חומרה ייעודית או מאיצי חישוב (GPU), וזאת מתוך מטרה לשמור על נגישות ולבדוק את ביצועי המערכת בתנאים ריאליים של משתמש קצה.

מערכות הפעלה

הפיתוח בוצע על מערכות הפעלה ממשפחת Windows ו-Linux. המערכת אינה תלויה במערכת הפעלה ספציפית, כל עוד קיימת תמיכה בדפדפן מודרני ובסביבת הרצה ל-Python.

תוכנה וכלי פיתוח

- עורך קוד: Visual Studio Code
 - דפדפנים לבדיקות: Google Chrome, Mozilla Firefox
 - סביבת שרת: Python 3
 - כלי בדיקה: Chrome DevTools, WebRTC Internals
- שימוש בכלים אלו מאפשר ניתוח תעבורת רשת, בדיקת חיבורי WebRTC, צפייה ב-ICE Candidates ומעקב אחר ביצועי המערכת בזמן אמת.

סקירת טכנולוגיות הפרויקט

המערכת משלבת מספר טכנולוגיות מרכזיות, כאשר כל אחת מהן נבחרה מתוך שיקול פונקציונלי, לימודי והנדסי.

WebRTC

WebRTC (Web Real-Time Communication) הוא תקן פתוח המאפשר תקשורת בזמן אמת של וידאו, קול ונתונים ישירות בין דפדפנים, ללא צורך בתוספים חיצוניים. בפרויקט זה WebRTC מהווה את ליבת מערכת התקשורת, ומאפשר העברת מדיה מוצפנת בין המשתתפים בארכיטקטורת Peer-to-Peer.

השימוש ב-WebRTC מאפשר:

- תקשורת בזמן אמת עם Latency נמוך

- הצפנה מובנית של המדיה
- תאימות מלאה לדפדפנים מודרניים
- עבודה ללא שרת מדיה מרכזי

JavaScript – צד לקוח

צד הלקוח פותח בשפת JavaScript, המהווה את הבחירה הטבעית לפיתוח אפליקציות Web אינטראקטיביות. JavaScript מאפשרת גישה ישירה ל-WebRTC APIs, ניהול MediaStreams, שליטה במצלמה ובמיקרופון, והצגת ממשק משתמש דינמי.

באמצעות JavaScript ממומשים:

- יצירת RTCPeerConnection
- ניהול זרמי וידאו וקול
- שיתוף מסך
- טיפול באירועי חיבור וניתוק
- שילוב עתידי של עיבוד וידאו מבוסס Machine Learning

Python – צד שרת (Signaling)

צד השרת פותח בשפת Python ומשמש כשרת Signaling בלבד. **השרת אחראי על:**

- ניהול חיבורי WebSocket
- החלפת הודעות SDP
- העברת ICE Candidates בין המשתתפים
- ניהול חדרים ומשתמשים

Python נבחרה בזכות פשטותה, קריאות הקוד והיכולת לפתח שרת קל משקל שאינו מעבד מדיה, אלא מתמקד באיתות בלבד.

WebSocket - WSS

WebSocket מאפשר תקשורת דו-כיוונית רציפה בין לקוח לשרת. בפרויקט נעשה שימוש ב-WSS (WebSocket Secure), על מנת להבטיח הצפנה של שכבת האיתות.

באמצעות WebSocket מועברות:

- הודעות JSON
- נתוני SDP
- ICE Candidates
- הודעות מערכת וניהול

Machine Learning ועיבוד וידאו

מערכת עיבוד הווידאו מבוססת על אלגוריתמים חישוביים שפותחו מאפס, ללא שימוש בספריות חיצוניות לעיבוד תמונה או למידת מכונה. **המימוש מתבסס על:**

- NumPy לחישובים מתמטיים
- עיבוד מטריצות תמונה
- אלגוריתמים לזיהוי ומעקב אחר תווי פנים

גישה זו מאפשרת שליטה מלאה בלוגיקה, הבנת תהליך הלמידה והכנה לאינטגרציה עתידית עם זרמי וידאו חיים.

מסד נתונים

מסד הנתונים של המערכת ממוקם בצד השרת ומשמש לצרכים תפעוליים וניהוליים של מערכת ה-WebRTC. מסד נתונים זה מהווה רכיב מרכזי בניהול המערכת ואחראי על אחסון וניהול מידע לוגי, שאינו כולל מדיה חזותית או קולית.

מסד הנתונים בשרת משמש, בין היתר, ל:

- ניהול משתמשים (הרשמה, התחברות ואימות)
- שמירת פרטי חדרים פעילים
- שיוך משתמשים לחדרים

- ניהול תפקידי משתמש (Host / Participant)
- שמירת פרטי חיבורים ולוגיקה תפעולית
- תמיכה בתהליכי בקרה, ניתוק והתחברות מחדש

הבחירה למקם את מסד הנתונים בצד השרת בלבד, ולמנוע אחסון של מדיה או נתונים חזותיים, נעשתה מתוך שיקולי אבטחת מידע ופרטיות. כל תעבורת הווידאו והקול מועברת ישירות בין המשתתפים בתקשורת Peer-to-Peer מוצפנת, מבלי לעבור דרך מסד הנתונים או להישמר בשרת.

מבנה זה מאפשר שמירה על ארכיטקטורה קלה ויעילה, תוך הפרדה ברורה בין שכבת האיתות והניהול לבין שכבת המדיה, ומאפשר הרחבה עתידית של מנגנוני ניהול ואימות ללא פגיעה בפרטיות המשתמשים.

תיאור מפורט של המערכת

ארכיטקטורה כללית של המערכת

המערכת נבנתה בארכיטקטורה משולבת מסוג Server-Client לצורך איתות (Signaling) ו-Peer-to-Peer להעברת מדיה, במבנה Mesh מרובה משתתפים. ארכיטקטורה זו מפרידה באופן ברור בין שכבת הניהול והאיתות לבין שכבת העברת המדיה, ומאפשרת תקשורת מאובטחת ויעילה ללא תלות בשרת מדיה מרכזי.

במבנה זה, שרת מרכזי אחראי אך ורק על תיאום הקשרים בין המשתתפים, בעוד שכל תעבורת הווידאו והקול מועברת ישירות בין הדפדפנים של המשתתפים, בצורה מוצפנת מקצה לקצה.

רכיבי המערכת המרכזיים

צד לקוח (Client – Browser)

צד הלקוח הוא אפליקציית Web הפועלת בדפדפן, ומפותחת בשפת JavaScript. כל לקוח אחראי על:

- גישה למצלמה ולמיקרופון
- יצירת זרמי מדיה (MediaStreams)
- יצירת חיבורי RTCPeerConnection
- ניהול מספר חיבורים במקביל (אחד לכל משתתף בחדר)
- הצפנה והעברת מדיה בזמן אמת
- הצגת ממשק משתמש (ווידאו, צ'אט, כפתורים)

כל לקוח במערכת פועל כ-Peer עצמאי, ומנהל חיבור נפרד מול כל Peer אחר בחדר.

צד שרת – Signaling Server

שרת ה-Signaling פותח בשפת Python ומשמש כשרת תיאום בלבד. תפקידו הוא:

- קבלת חיבורי WebSocket מאובטחים (WSS)
- ניהול חדרים ומשתתפים
- העברת הודעות SDP בין לקוחות
- העברת ICE Candidates בין לקוחות

- שמירת מידע לוגי וניהולי במסד הנתונים

השרת אינו מעביר, מעבד או שומר מדיה, ובכך מצטמצם משטח התקיפה ונשמרת פרטיות המשתמשים.

מסד נתונים בצד השרת

מסד הנתונים משמש את שרת ה-Signaling בלבד, ואינו מעורב בהעברת מדיה. הוא אחראי על:

- ניהול משתמשים והרשאות
- שיוך משתמשים לחדרים
- שמירת מצב לוגי של חדרים פעילים
- תמיכה בתהליכי אימות ובקרה

הפרדה זו מבטיחה שמידע רגיש כגון וידאו וקול לעולם אינו נשמר או עובר דרך השרת.

Mesh Peer-to-Peer ארכיטקטורת

במערכת זו מיושמת ארכיטקטורת Mesh מלאה, בה כל משתתף יוצר חיבור Peer-to-Peer ישיר מול כל משתתף אחר.

לדוגמה:

- בחדר עם 2 משתתפים – חיבור אחד

- בחדר עם 3 משתתפים – כל לקוח מנהל 2 חיבורים
- בחדר עם 4 משתתפים – כל לקוח מנהל 3 חיבורים

יתרונות גישה זו:

- פרטיות מלאה – אין שרת מדיה
- Latency נמוך
- הצפנה נפרדת לכל חיבור
- שליטה מלאה בזרימת המדיה

חסרונות:

- עומס חישובי גובר עם ריבוי משתתפים
- שימוש מוגבר ברוחב פס בצד הלקוח

בחירה זו נעשתה במודע, בהתאם ליעדי הפרויקט ולדגש על לימוד, שליטה ואבטחה.

זרימת המידע במערכת

המערכת עושה שימוש בשני ערוצי תקשורת נפרדים:

1. ערוץ איתות (Signaling)

- מבוסס WebSocket מאובטח (WSS)
- משמש להעברת:
 - SDP Offer / Answer
 - ICE Candidates
 - הודעות ניהול
- עובר דרך שרת ה-Signaling בלבד

2. ערוץ מדיה (Media)

- מבוסס WebRTC Peer-to-Peer
- כולל:
 - וידאו
 - קול
 - שיתוף מסך
- מוצפן באמצעות DTLS-SRTP
- אינו עובר דרך השרת

שילוב תשתית עיבוד וידאו (Machine Learning)

תשתית עיבוד הווידאו פועלת בצד הלקוח, כחלק מצינור עיבוד המדיה המקומי. עיבוד זה מתבצע לפני שליחת ה-MediaStream ל-Peers אחרים, ומאפשר:

- ניתוח פריימים מהמצלמה
- הפעלת פילטרים חישוביים בזמן אמת
- שמירה על פרטיות מלאה (ללא שליחת נתונים לשרת)

תשתית זו מופרדת לוגית ממנגנון התקשורת, אך תוכננה כך שתוכל להשתלב באופן טבעי בזרימת המדיה של WebRTC.

פרוטוקולי התקשורת במערכת

מערכת ה-WebRTC מבוססת על שילוב של מספר פרוטוקולי תקשורת, אשר פועלים יחד לצורך יצירת חיבור מאובטח, יציב ובזמן אמת בין המשתתפים. בפרויקט זה נעשה שימוש בפרוטוקולים אלו בצורה ישירה ומודעת, תוך הבנה של תפקידו של כל רכיב בתהליך.

הפרוטוקולים במערכת נחלקים לשתי קבוצות עיקריות:

- פרוטוקולי איתות וניהול חיבור
- פרוטוקולי העברת מדיה והצפנה

WebRTC – עקרונות פעולה כלליים

WebRTC אינו פרוטוקול בודד, אלא סט תקנים וממשקי API המאפשרים תקשורת בזמן אמת בין דפדפנים. תפקידו העיקרי הוא לאפשר העברת וידאו, קול ונתונים בצורה ישירה בין לקוחות, ללא צורך בשרת מדיה מרכזי.

בבסיס WebRTC עומדים שלושה רכיבים עיקריים:

- RTCPeerConnection – ניהול חיבורי Peer-to-Peer
- MediaStream – ניהול זרמי מדיה (וידאו, קול, מסך)
- DataChannel – העברת נתונים (לא בשימוש מרכזי בפרויקט זה)

RTCPeerConnection

RTCPeerConnection הוא האובייקט המרכזי האחראי על יצירת וניהול חיבור Peer-to-Peer בין שני משתתפים. בפרויקט זה, כל לקוח יוצר מופע RTCPeerConnection נפרד עבור כל משתתף אחר בחדר, בהתאם לארכיטקטורת Mesh.

תפקידי RTCPeerConnection כוללים:

- ניהול משא ומתן על תצורת החיבור
- החלפת פרטי מדיה ויכולות קידוד
- טיפול בהצפנה ובאימות
- ניהול חיבורי ICE

SDP – Session Description Protocol

SDP הוא פורמט טקסטואלי המתאר את מאפייני החיבור בין שני משתתפים. במסגרת WebRTC, SDP משמש לתיאור:

- סוגי מדיה (וידאו, קול)

- קודקים נתמכים
- פרטי הצפנה
- פרטי רשת ראשוניים

תהליך החיבור מתבצע באמצעות מנגנון Offer / Answer:

1. אחד המשתתפים יוצר SDP Offer
2. ה-Offer נשלח למשתתף השני דרך שרת ה-Signaling
3. המשתתף השני מחזיר SDP Answer
4. לאחר השלמת התהליך, החיבור יכול להתחיל

ICE – Interactive Connectivity Establishment

ICE הוא מנגנון המשמש למציאת נתיב התקשורת הטוב ביותר בין שני משתתפים. מכיוון שמשתתפים רבים נמצאים מאחורי NAT או Firewall, חיבור ישיר אינו תמיד אפשרי.

ICE אחראי על:

- איסוף כתובות רשת אפשריות (Candidates)
- בדיקת זמינות כל נתיב
- בחירת הנתיב היעיל ביותר

STUN – Session Traversal Utilities for NAT

שרת STUN מאפשר ללקוח לגלות את כתובתו הציבורית כפי שהיא נראית מחוץ לרשת המקומית. מידע זה משמש כחלק מתהליך ICE, ומאפשר ניסיון חיבור ישיר בין שני לקוחות.

STUN אינו מעביר מדיה ואינו מתווך את התקשורת, אלא משמש לגילוי כתובות בלבד.

TURN – Traversal Using Relays around NAT

במקרים בהם חיבור ישיר אינו אפשרי (למשל עקב Firewall מחמיר), נעשה שימוש בשרת TURN. שרת זה משמש כמתווך, דרכו מועברת תעבורת המדיה.

בפרויקט זה נעשה שימוש בשרת TURN חיצוני רק כמוצא אחרון, כאשר חיבור Peer-to-Peer ישיר אינו מתאפשר.

WebSocket Secure (WSS) – Signaling

שרת ה־Signaling עושה שימוש בפרוטוקול WSS - WebSocket Secure לצורך החלפת הודעות בין הלקוחות. ערוץ זה משמש להעברת:

- הודעות SDP
- ICE Candidates
- הודעות ניהול וחדרים

כל התקשורת בערוץ זה מוצפנת באמצעות TLS, אך אינה כוללת מדיה.

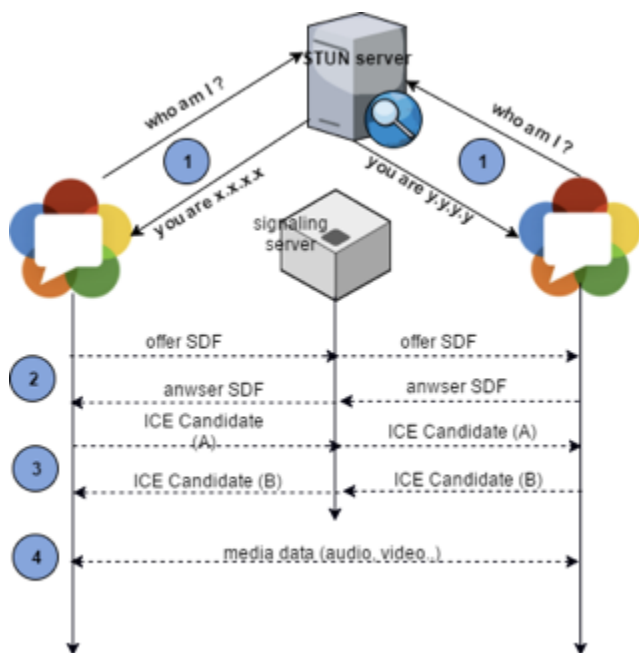
הפרדה בין ערוצי תקשורת

אחד העקרונות המרכזיים במערכת הוא ההפרדה המלאה בין:

- ערוץ האיתות (דרך השרת)
- ערוץ המדיה (Peer-to-Peer)

הפרדה זו:

- מגדילה פרטיות
- מצמצמת עומס שרת
- משפרת אבטחה
- מאפשרת Scalability מבוקרת



שילוב עיבוד וידאו מבוסס Machine Learning בזרימת התקשורת

לצד פרוטוקולי התקשורת, המערכת תוכננה כך שתאפשר שילוב של עיבוד וידאו חישובי בצד הלקוח, כחלק מצינור העברת המדיה של WebRTC. עיבוד זה אינו מהווה חלק מפרוטוקולי התקשורת עצמם, אלא פועל בשכבה לוגית נפרדת, לפני שליחת ה-MediaStream ל-Peers אחרים.

במהלך פעולת המערכת, פריימים המתקבלים מהמצלמה המקומית יכולים לעבור עיבוד מקדים באמצעות אלגוריתמים חישוביים, כגון זיהוי ומעקב אחר תווי פנים, טרם קידודם והעברתם באמצעות WebRTC. תהליך זה מתבצע באופן מקומי בדפדפן, ואינו משפיע על תהליך האיתות (Signaling), על מנגנון ICE או על תצורת החיבור בין המשתתפים.

הפרדה זו מאפשרת:

- שמירה על תקינות פרוטוקולי התקשורת
- מניעת תלות בין שכבת התקשורת לשכבת העיבוד
- שמירה על פרטיות מלאה, שכן נתוני הווידאו המעובדים אינם מועברים לשרת
- אפשרות להרחבה עתידית של אלגוריתמי העיבוד ללא שינוי במנגנוני החיבור

פירוט מלא של תשתית ה-Machine Learning, האלגוריתמים והשילוב העתידי שלהם במערכת יוצג בפרק ייעודי במסמך.

סיכום הפרק

שילוב הפרוטוקולים WebRTC, SDP, ICE, STUN, TURN ו-WebSocket מאפשר למערכת ליצור חיבורי תקשורת בזמן אמת, מאובטחים ועמידים למגבלות רשת. הבנת תפקידו של כל פרוטוקול והאינטראקציה ביניהם מהווה בסיס הכרחי למימוש נכון של מערכת Peer-to-Peer מרובת משתתפים.

אבטחת מידע והצפנה במערכת

אבטחת מידע מהווה מרכיב מרכזי בתכנון ובמימוש המערכת, לאור העובדה שהיא עוסקת בהעברת מדיה בזמן אמת, הכוללת וידאו וקול רגישים. בפרויקט זה הושם דגש על הצפנה מלאה, הפרדת שכבות תקשורת וצמצום חשיפת מידע, תוך שימוש במנגנוני האבטחה המובנים בתקני WebRTC ובפרוטוקולי תקשורת מאובטחים.

עקרון מנחה בתכנון המערכת הוא שמידע חזותי וקולי אינו נשמר ואינו מעובד בצד השרת, אלא מועבר ישירות בין המשתתפים בתקשורת Peer-to-Peer מוצפנת.

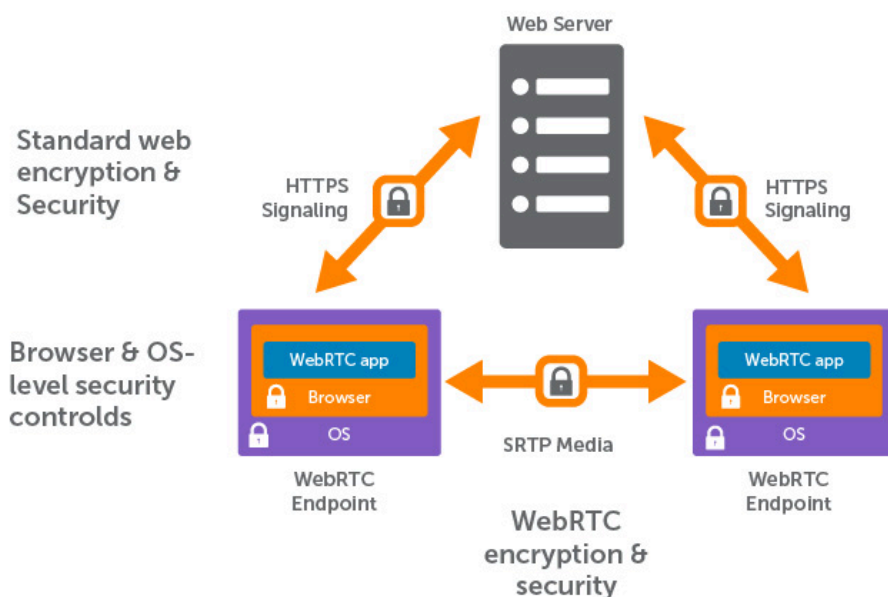
אבטחת שכבת האיתות (Signaling)

שכבת האיתות במערכת מבוססת על פרוטוקול (WebSocket Secure (WSS, הפועל מעל TLS. שכבה זו אחראית על העברת הודעות ניהול, SDP ו-ICE Candidates בין הלקוחות, אך אינה כוללת העברת מדיה.

אמצעי האבטחה בשכבת האיתות כוללים:

- שימוש ב-TLS להצפנת תעבורת WebSocket
- מניעת האזנה או שינוי הודעות איתות (Man-in-the-Middle)
- העברת מידע בפורמט JSON מבווקר
- הפרדה בין מידע ניהולי לבין נתוני מדיה

שימוש ב-WSS מבטיח כי גם שכבת התיאום בין המשתתפים מוגנת מפני יירוט או שינוי זדוני.



הצפנת מדיה – DTLS-SRTP

העברת המדיה (וידאו וקול) במערכת מתבצעת באמצעות WebRTC Peer-to-Peer, תוך שימוש במנגנוני הצפנה מובנים:

- DTLS - Datagram Transport Layer Security – משמש לאימות והחלפת מפתחות הצפנה בין ה-Peers
- SRTP - Secure Real-time Transport Protocol – משמש להצפנת תעבורת המדיה עצמה

תהליך זה מבטיח:

- הצפנה מקצה לקצה (End-to-End Encryption)
- יצירת מפתחות הצפנה נפרדים לכל חיבור Peer-to-Peer
- מניעת גישה לנתוני המדיה מצד השרת

שרת ה-Signaling אינו מחזיק במפתחות ההצפנה ואינו מסוגל לפענח את המדיה.

הפרדת תפקידים וצמצום משטח התקיפה

במערכת קיימת הפרדה ברורה בין רכיבי המערכת:

- השרת אחראי על איתות וניהול בלבד
- הלקוחות אחראים על יצירת חיבורים והעברת מדיה
- מסד הנתונים מכיל מידע לוגי בלבד, ללא מדיה

הפרדה זו:

- מצמצמת את משטח התקיפה
- מפחיתה סיכוני דליפת מידע
- מאפשרת תחזוקה מאובטחת ופשוטה יותר

אבטחת תקשורת מול שרתי STUN ו-TURN

שרתי STUN ו-TURN משמשים לצורך גילוי נתיבי תקשורת והתמודדות עם חסימות רשת. שרתים אלו אינם מעבדים או שומרים מדיה באופן קבוע, ותפקידם מוגבל לתיווך חיבור בלבד במקרים חריגים.

גם כאשר נעשה שימוש ב-TURN:

- המדיה מועברת בצורה מוצפנת
- השרת אינו מסוגל לפענח את התוכן
- השימוש ב-TURN מתבצע רק כאשר אין אפשרות לחיבור ישיר

פרטיות משתמשים

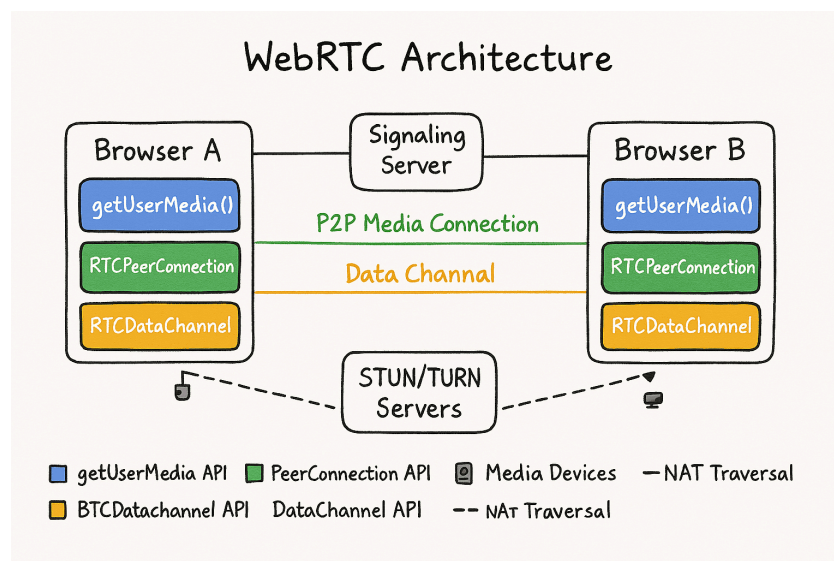
שמירה על פרטיות המשתמשים הייתה שיקול מרכזי בתכנון המערכת. **בהתאם לכך:**

- לא מתבצע רישום או הקלטה של שיחות
- לא נשמרים נתוני וידאו או קול
- עיבוד הווידאו (כולל פילטרים חישוביים) מתבצע מקומית בצד הלקוח בלבד

גישה זו מבטיחה שליטה מלאה של המשתמש במידע האישי שלו ומונעת חשיפה מיותרת.

סיכום אבטחת המידע

מערכת זו עושה שימוש במנגנוני אבטחה תקינים ומוכחים, תוך שילוב של הצפנה מקצה לקצה, הפרדת שכבות והימנעות מאחסון מדיה בשרת. תכנון זה מאפשר יצירת מערכת תקשורת בזמן אמת מאובטחת, פרטית ועמידה בפני איומים נפוצים ברשת.



אלגוריתמים מרכזיים במערכת

מערכת ה-WebRTC המפותחת בפרויקט נשענת על מספר אלגוריתמים מרכזיים, אשר אחראים על יצירת החיבורים, ניהול המשתתפים, העברת המדיה והתמודדות עם תרחישי קצה כגון ניתוק והתחברות מחדש. אלגוריתמים אלו אינם מתבססים על מנגנון יחיד, אלא על שילוב של תהליכים סינכרוניים וא-סינכרוניים הפועלים יחד בזמן אמת.

אלגוריתם יצירת חדר וניהול תפקידים

בעת יצירת חדר חדש במערכת, מתבצע תהליך לוגי הכולל:

1. משתמש יוזם בקשה ליצירת חדר
2. שרת ה-Signaling מקצה מזהה חדר ייחודי
3. המשתמש מוגדר אוטומטית כ-Host
4. פרטי החדר נשמרים במסד הנתונים בצד השרת
5. החדר מוכן לקבלת משתתפים נוספים

תפקיד ה-Host כולל הרשאות ניהול, כגון שליטה על משתתפים, ניתוק משתמשים וניהול מצב החדר.

אלגוריתם הצטרפות משתמש לחדר קיים

כאשר משתמש חדש מצטרף לחדר פעיל:

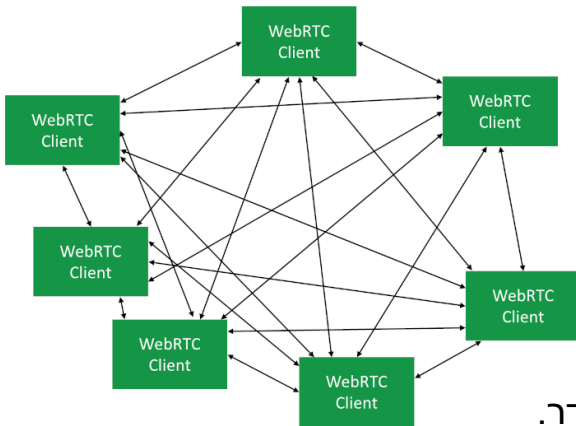
1. המשתמש יוצר חיבור WebSocket לשרת ה-Signaling
2. מתבצע אימות והרשאה
3. השרת שולח למשתמש את רשימת המשתתפים הקיימים בחדר
4. המשתמש מתחיל תהליך יצירת חיבור Peer-to-Peer מול כל משתתף קיים
5. המשתתפים הקיימים מתעדכנים בהצטרפות המשתמש החדש

אלגוריתם זה מבטיח שכל המשתתפים יישארו מסונכרנים לגבי מצב החדר.

אלגוריתם יצירת Mesh Peer-to-Peer

בלב המערכת נמצא אלגוריתם יצירת רשת Mesh, בו כל משתתף יוצר חיבור RTCPeerConnection נפרד מול כל משתתף אחר.

שלבי האלגוריתם:



1. יצירת RTCPeerConnection חדש
2. יצירת SDP Offer
3. שליחת ה-Offer דרך שרת ה-Signaling
4. קבלת SDP Answer מה-Peer השני
5. התחלת החלפת ICE Candidates
6. מעבר למצב חיבור פעיל

תהליך זה חוזר על עצמו עבור כל זוג משתתפים בחדר.

אלגוריתם החלפת ICE Candidates

במהלך יצירת החיבור ולאחריו, כל משתתף:

- אוסף ICE Candidates מקומיים
- שולח אותם ל-Peer השני דרך שרת ה-Signaling
- בודק זמינות של כל נתיב תקשורת
- בוחר את הנתיב האופטימלי

אלגוריתם זה מאפשר התאמה דינמית לתנאי הרשת ומבטיח חיבור יציב ככל האפשר.

אלגוריתם ניהול מדיה בזמן אמת

לאחר יצירת החיבורים, מתבצע ניהול שוטף של זרמי המדיה:

- קליטת וידאו וקול מהמצלמה והמיקרופון
- קידוד המדיה בהתאם ליכולות ה-Peer
- שליחת המדיה דרך חיבורי WebRTC
- הצגת המדיה המתקבלת בממשק המשתמש

אלגוריתם זה פועל ברציפות ודורש סנכרון מדויק בין חיבורים מרובים.

אלגוריתם ניתוק והתאוששות מחיבור שנפל

במקרה של ניתוק משתתף או נפילת חיבור:

1. זיהוי מצב disconnected / failed
2. עדכון המשתתפים האחרים
3. סגירת חיבורי RTCPeerConnection רלוונטיים
4. ניסיון ICE Restart במידת הצורך
5. חיבור מחדש או הסרת המשתתף מהחדר

מנגנון זה מאפשר המשך עבודה יציב גם בתנאי רשת משתנים.

אלגוריתם שילוב עיבוד וידאו בצד הלקוח

בצד הלקוח, לפני שליחת ה-MediaStream ל-Peers אחרים, ניתן לשלב שלב עיבוד מקדים:

1. קליטת פריימים מהמצלמה
2. עיבוד חישובי (פילטרים, ניתוח תווים)
3. יצירת זרם וידאו מעובד
4. שליחת הזרם ל-WebRTC

אלגוריתם זה מופרד לוגית משכבת התקשורת ואינו משפיע על מנגנוני החיבור עצמם.

סיכום האלגוריתמים

שילוב האלגוריתמים במערכת מאפשר יצירת פתרון תקשורת בזמן אמת, יציב, מאובטח ומודולרי. האלגוריתמים תוכננו כך שיפעלו יחד בצורה יעילה, תוך שמירה על הפרדה בין שכבות, פרטיות המשתמש ויכולת הרחבה עתידית.

תיאור פונקציונלי של המערכת

ניהול משתמשים

המערכת כוללת מנגנון ניהול משתמשים מלא, המאפשר הרשמה, התחברות ואימות משתמשים בצורה מאובטחת. מנגנון זה נועד לוודא כי רק משתמשים מורשים יכולים ליצור חדרים, להצטרף לשיחות ולבצע פעולות ניהול.

כל משתמש במערכת מזוהה באמצעות כתובת דואר אלקטרוני וסיסמה, המשמשים כאמצעי הזדהות ייחודי.

תהליך הרשמה (Sign Up)

בעת הרשמה למערכת, מתבצע התהליך הבא:

1. המשתמש מזין כתובת דואר אלקטרוני וסיסמה
2. הנתונים נשלחים לשרת בצורה מאובטחת
3. השרת שומר את פרטי המשתמש במסד הנתונים
4. נשלחת הודעת דואר אלקטרוני לאימות החשבון
5. המשתמש נדרש לאשר את החשבון באמצעות קישור ייעודי

רק לאחר השלמת תהליך האימות, ניתן להתחבר למערכת ולהשתמש בשירותיה.

מנגנון זה נועד:

- למנוע יצירת משתמשים מזויפים
 - להבטיח תקינות כתובות דוא"ל
- לחזק את רמת האבטחה הכללית של המערכת

תהליך התחברות (Login)

לאחר הרשמה ואימות מוצלח, המשתמש יכול להתחבר למערכת:

1. הזנת כתובת דוא"ל וסיסמה
2. אימות הנתונים מול מסד הנתונים
3. פתיחת Session מאובטח

4. מתן גישה לפונקציות המערכת בהתאם להרשאות המשתמש
במקרה של כשל בהתחברות, תוצג הודעת שגיאה מתאימה למשתמש.

ניהול חדרים

המערכת מבוססת על קונספט של חדרים (Rooms), כאשר כל חדר מייצג סשן תקשורת עצמאי.

יצירת חדר

- משתמש מחובר יכול ליצור חדר חדש
- השרת מקצה מזהה חדר ייחודי
- יוצר החדר מוגדר אוטומטית כ-Host
- החדר נרשם כמופעל וזמין להצטרפות

הצטרפות לחדר

- משתמשים אחרים יכולים להצטרף לחדר באמצעות מזהה או קישור
- מתבצע אימות הרשאות
- לאחר הצטרפות מתחיל תהליך יצירת חיבורי WebRTC מול שאר המשתתפים

הרשאות ותפקידים

במערכת קיימים תפקידי משתמש שונים:

Host

- יצירת חדר
- ניהול משתתפים
- שליטה על מצב החדר

Participant

- הצטרפות לחדר
- שליחת וקבלת וידאו וקול
- שימוש בצ'אט ושיתוף מסך

חלוקה זו מאפשרת שליטה ברורה על תפקידי המשתמשים במהלך השיחה.

אבטחת מידע בתהליכי משתמשים

כל תהליכי ההרשמה, ההתחברות וניהול החדרים מתבצעים דרך ערוצי תקשורת מאובטחים (HTTPS / WSS). סיסמאות אינן נשמרות בצורה גלויה, והמערכת אינה חושפת פרטים רגישים למשתמשים אחרים.

סיכום התיאור הפונקציונלי

מנגנון ניהול המשתמשים והחדרים מהווה שכבת בסיס קריטית לפעולת המערכת. שילוב של אימות משתמשים, הרשאות וחלוקת תפקידים מאפשר שימוש מאובטח, מבוקר ומסודר במערכת התקשורת בזמן אמת.

ממשק משתמש (User Interface)

ממשק המשתמש של המערכת תוכנן כך שיהיה ברור, אינטואיטיבי ונוח לשימוש, גם עבור משתמשים שאינם בעלי רקע טכני. עיצוב הממשק נועד לאפשר גישה מהירה לפונקציות המרכזיות של המערכת, תוך שמירה על חוויית שימוש רציפה בזמן שיחה.

הממשק פועל מתוך דפדפן אינטרנט, ואינו דורש התקנה של תוכנה חיצונית. כל הפעולות מתבצעות בזמן אמת, תוך סנכרון מלא בין המשתתפים בחדר.

מסך כניסה (Login)

מסך הכניסה הוא נקודת ההתחלה של המערכת. במסך זה המשתמש מתבקש להזין:

- כתובת דואר אלקטרוני
- סיסמה

לאחר הזנת הפרטים ולחיצה על כפתור התחברות, מתבצע אימות מול השרת. במקרה של פרטים שגויים, תוצג הודעת שגיאה מתאימה.

מסך זה כולל גם קישור למסך ההרשמה עבור משתמשים חדשים.

מסך הרשמה (Sign Up)

מסך ההרשמה מאפשר יצירת חשבון משתמש חדש במערכת. המשתמש מזין:

- כתובת דואר אלקטרוני
- סיסמה
- אישור סיסמה

לאחר השלמת תהליך ההרשמה, נשלחת הודעת דואר אלקטרוני לאימות החשבון. עד לאישור החשבון, לא ניתן להתחבר למערכת.

מנגנון זה תורם לאבטחת המערכת ולמניעת שימוש לרעה.

מסך ראשי / לובי

לאחר התחברות מוצלחת, המשתמש מועבר למסך הראשי (Lobby), ממנו ניתן:

- ליצור חדר חדש
- להצטרף לחדר קיים באמצעות מזהה או קישור
- לצפות במידע בסיסי על המשתמש המחובר

מסך זה מהווה מעבר בין שלב האימות לבין שלב השיחה עצמה.

מסך חדר (Room View)

מסך החדר הוא המסך המרכזי במערכת, ובו מתבצעת השיחה בזמן אמת. מסך זה כולל:

תצוגת וידאו

- תצוגת Grid של משתתפי החדר
- הצגת וידאו חי מכל משתתף
- התאמת גודל הווידאו למספר המשתתפים

בקרי מדיה

- הפעלה / השתקה של מיקרופון
- הפעלה / כיבוי של מצלמה
- שיתוף מסך
- יציאה מהחדר

הבקרים זמינים בכל עת ומאפשרים שליטה מיידית במהלך השיחה.

צ'אט טקסטואלי

לצד שיחת הווידאו, קיים צ'אט טקסטואלי בזמן אמת, המאפשר:

- שליחת הודעות בין המשתתפים

- הצגת היסטוריית הודעות במהלך השיחה
- תקשורת שקטה במקביל לשיחה הקולית

הצ'אט עושה שימוש בערוץ האיתות ואינו משפיע על תעבורת המדיה.

מסך ניהול – Host

למשתמש המוגדר כ-Host מוצג ממשק ניהול נוסף, הכולל:

- רשימת משתתפים מחוברים
- זיהוי תפקיד המשתתפים
- אפשרות לניתוק משתמשים
- שליטה בהרשאות בסיסיות

ממשק זה מאפשר ניהול מסודר של השיחה ושמירה על סדר ותקינות.

חווית משתמש ושיקולי עיצוב

בעת תכנון הממשק הושם דגש על:

- מינימליזם ופשטות
- תגובתיות (Responsive Design)
- הצגת מידע ברור ולא עמוס
- זמינות פונקציות קריטיות בכל שלב

הממשק תוכנן כך שיתאים גם לשימוש במסכים שונים וברזולוציות שונות.

סיכום ממשק המשתמש

ממשק המשתמש מהווה חלק מרכזי בהצלחת המערכת, ומאפשר למשתמשים לנצל את יכולות התקשורת והניהול בצורה נוחה, ברורה ובטוחה. שילוב של ממשק אינטואיטיבי עם תשתית טכנולוגית מתקדמת יוצר מערכת שמישה, יעילה ונגישה.

Design Detailed – מפרט תכנון מפורט

פרק זה מציג פירוט מעמיק של מבנה הפרויקט, חלוקת הקוד למודולים והאחריות של כל רכיב במערכת. מטרת הפרק היא להציג את מבנה המערכת ברמת הקוד, ולהדגים כיצד הרכיבים שתוארו בפרקים הקודמים ממומשים בפועל.

המערכת בנויה בגישה מודולרית, תוך הפרדה ברורה בין:

- צד לקוח (Client)
- צד שרת (Signaling Server)
- מסד נתונים
- רכיבי תשתית ותצורה

מבנה כללי של הפרויקט

הפרויקט מחולק למספר תיקיות וקבצים עיקריים, כאשר כל רכיב אחראי על שכבה לוגית שונה במערכת:

- קבצי צד לקוח (HTML / CSS / JavaScript)
- קבצי צד שרת (Python)
- קבצי תצורה
- רכיבי מסד נתונים

חלוקה זו מאפשרת תחזוקה נוחה, קריאות קוד והרחבה עתידית של המערכת.

צד לקוח – Client

צד הלקוח הוא אפליקציית Web הפועלת בדפדפן, ואחראית על כל הלוגיקה הקשורה לחוויית המשתמש ולתקשורת בזמן אמת.

קבצי HTML

קבצי ה-HTML אחראים על:

- מבנה המסכים
- הצגת רכיבי ממשק
- שילוב קבצי JavaScript ו-CSS

קבצים אלו מגדירים את שלד הממשק בלבד, בעוד הלוגיקה ממומשת בקבצי JavaScript.

קבצי CSS

קבצי העיצוב אחראים על:

- פריסת מסכים
- עיצוב תצוגת וידאו
- התאמה למסכים שונים (Responsive Design)

העיצוב נשמר פשוט וברור, במטרה להבליט את פונקציונליות המערכת.

קבצי JavaScript – לוגיקת WebRTC

קבצי ה-JavaScript מהווים את ליבת צד הלקוח, וכוללים:

- יצירת RTCPeerConnection
- ניהול MediaStreams
- טיפול ב-SDP Offer / Answer
- החלפת ICE Candidates
- ניהול חיבורי WebSocket
- עדכון ממשק המשתמש בזמן אמת

הלוגיקה מחולקת לפונקציות ברורות, כאשר כל פונקציה אחראית על שלב אחר בתהליך החיבור.

ניהול חיבורים מרובים

בשל ארכיטקטורת Mesh, צד הלקוח מנהל מספר חיבורי RTCPeerConnection במקביל. לשם כך נעשה שימוש במבנה נתונים הממפה בין מזהי משתמשים לחיבורי PeerConnection פעילים.

גישה זו מאפשרת:

- יצירה והסרה דינמית של חיבורים
- ניהול חיבורים לפי משתתף
- טיפול יעיל בניתוקים והתחברויות מחדש

צד שרת – Signaling Server

שרת ה-Signaling ממומש בשפת Python, ואחראי על תיאום החיבורים בין הלקוחות.

תפקידים מרכזיים של השרת

- קבלת חיבורי WebSocket
- ניהול חדרים ומשתתפים
- העברת הודעות SDP
- העברת ICE Candidates
- אינטראקציה עם מסד הנתונים

השרת אינו מעבד מדיה ואינו שומר נתוני וידאו או קול.

מבנה קוד השרת

קוד השרת מחולק למודולים לוגיים, כגון:

- מודול חיבורי WebSocket
- מודול ניהול חדרים
- מודול ניהול משתמשים
- מודול גישה למסד הנתונים

חלוקה זו מאפשרת תחזוקה נוחה והרחבה עתידית של רכיבי המערכת.

מסד נתונים

מסד הנתונים משמש לאחסון מידע לוגי בלבד, ואינו כולל מדיה.

המידע הנשמר כולל:

- פרטי משתמשים
- סטטוס אימות
- שיוך משתמשים לחדרים
- תפקידי משתמשים (Host / Participant)

גישה למסד הנתונים מתבצעת דרך שכבת גישה ייעודית, המונעת גישה ישירה מהלוגיקה העסקית.

טיפול באירועים (Event Handling)

המערכת עושה שימוש באירועים (Events) לצורך:

- חיבור וניתוק משתמשים
- קבלת הודעות WebSocket
- שינוי מצב חיבור WebRTC
- עדכון ממשק המשתמש

גישה מבוססת אירועים מאפשרת תגובתיות גבוהה והתאמה לשינויים בזמן אמת.

שילוב של עיבוד וידאו (Machine Learning)

תשתית עיבוד הווידאו ממוקמת בצד הלקוח, ומופרדת מהלוגיקה של WebRTC. העיבוד מתבצע על פריימים לפני שליחתם ל-MediaStream.

תכנון זה מאפשר:

- הוספת פילטרים חישוביים
- הרחבת אלגוריתמים

- שמירה על קוד נקי ומודולרי

פירוט האלגוריתמים עצמם יוצג בפרק ייעודי.

סיכום Design Detailed

המפרט המפורט מדגים כיצד הרעיונות הארכיטקטוניים של המערכת ממומשים בפועל בקוד. חלוקה מודולרית, הפרדת אחריות וניהול חיבורים מבוקר מאפשרים מערכת יציבה, מאובטחת וניתנת להרחבה.

תשתית Machine Learning – עיצוב ואלגוריתמים

פרק זה מתאר את תשתית עיבוד הווידאו והאלגוריתמים החשובים שפותחו כחלק מהפרויקט, לצורך הוספת פילטרים חכמים למצלמה בזמן אמת. תשתית זו תוכננה לפעול בצד הלקוח בלבד, ללא תלות בשרת וללא שימוש בספריות למידת מכונה או עיבוד תמונה חיצוניות.

מטרת הפרק היא להציג את הגישה ההנדסית שנבחרה, את מבנה האלגוריתמים ואת אופן השילוב העתידי שלהם במערכת ה-WebRTC.

עקרונות תכנון כלליים

תשתית ה-Machine Learning בפרויקט פותחה בהתאם לעקרונות הבאים:

- עיבוד מקומי בלבד בצד הלקוח
- הימנעות מהעברת נתונים חזותיים לשרת
- מימוש אלגוריתמי עצמאי (Low-Level)
- שימוש מינימלי בספריות עזר – NumPy בלבד
- התאמה לעיבוד בזמן אמת (Real-Time)

עקרונות אלו נבחרו מתוך רצון להבין לעומק את תהליכי עיבוד התמונה והחישוב, ולא להסתמך על מימושים מוכנים.

מיקום תשתית ה-ML בארכיטקטורה

תשתית ה-Machine Learning ממוקמת בצד הלקוח, כחלק מצינור עיבוד הווידאו המקומי. האלגוריתמים פועלים על פריימים המתקבלים ישירות מהמצלמה, לפני קידוד המדיה והעברתה באמצעות WebRTC.

מבנה זה מאפשר:

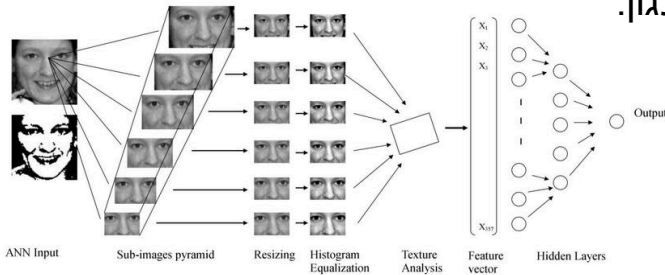
- עיבוד וידאו לפני יצירת MediaStream
- שליחת וידאו מעובד ל-Peers אחרים
- שמירה על הפרדה מלאה בין שכבת התקשורת לשכבת העיבוד

האלגוריתמים אינם משפיעים על תהליך ה-Signaling, על ICE או על יצירת החיבורים עצמם.

קליטת פריימים ועיבוד בסיסי

תהליך עיבוד הווידאו מתחיל בקליטת פריימים מהמצלמה המקומית. כל פריים מיוצג כמטריצה דו-ממדית או תלת-ממדית של ערכי פיקסלים.

באמצעות NumPy מתבצעות פעולות חישוביות כגון:



- המרת צבעים
- נרמול ערכים
- חיתוך אזורי עניין (ROI)
- חישובים סטטיסטיים על אזורים בתמונה

פעולות אלו מהוות בסיס לאלגוריתמים מתקדמים יותר.

זיהוי ומעקב אחר תווי פנים – גישה אלגוריתמית

המערכת כוללת תשתית ראשונית לזיהוי ומעקב אחר תווי פנים, הממומשת בגישה חישובית ישירה. הגישה אינה מבוססת על רשתות נוירונים מוכנות או מודלים מאומנים מראש, אלא על ניתוח תבניות בתמונה.

האלגוריתם מבצע:

- זיהוי אזורי עניין בתמונה
- ניתוח שינויים בין פריימים עוקבים
- מעקב אחר מיקום יחסי של תווי פנים

גישה זו מאפשרת הבנה מלאה של הלוגיקה, במחיר דיוק נמוך יותר לעומת מודלים מתקדמים – בחירה מודעת בהתאם ליעדי הפרויקט.

פילטרים חישוביים בזמן אמת

על בסיס ניתוח התמונה ניתן להפעיל פילטרים חישוביים שונים, כגון:

- הדגשת אזורים מסוימים בתמונה
- הוספת שכבות גרפיות
- שינוי מאפיינים חזותיים של אזור פנים

פילטרים אלו פועלים בזמן אמת, ולכן האלגוריתמים תוכננו להיות קלים חישובית ולהימנע מפעולות כבדות.

שיקולי ביצועים

עיבוד וידאו בזמן אמת מחייב התייחסות לביצועים:

- שמירה על קצב פריימים יציב
- הימנעות מלולאות כבדות
- שימוש בפעולות וקטוריות של NumPy
- עיבוד חלקי של פריימים במידת הצורך

שיקולים אלו מאפשרים הפעלת האלגוריתמים במקביל לניהול חיבורי WebRTC.

מגבלות המערכת

לתשתית ה-Machine Learning בפרויקט קיימות מגבלות ידועות:

- דיוק נמוך יחסית למודלים מבוססי Deep Learning
- רגישות לתנאי תאורה
- התאמה לסצנות פשוטות בלבד

מגבלות אלו נלקחו בחשבון כחלק מתהליך הפיתוח, והוגדרו כחלק מהתיחום ההנדסי של הפרויקט.

הרחבות עתידיות

התשתית תוכננה כך שתאפשר בעתיד:

- שילוב מודלים מתקדמים יותר
- שיפור אלגוריתמי זיהוי
- אופטימיזציה נוספת לביצועים
- שילוב עמוק יותר עם MediaStream של WebRTC

כל זאת מבלי לשנות את מבנה הארכיטקטורה הקיים.

סיכום תשתית ה-Machine Learning

תשתית ה-Machine Learning בפרויקט מדגימה גישה אלגוריתמית בסיסית לעיבוד וידאו בזמן אמת, תוך שליטה מלאה בלוגיקה ובזרימת הנתונים. שילוב תשתית זו במערכת WebRTC מדגיש את היכולת להרחיב מערכת תקשורת בזמן אמת לעולמות של עיבוד חישובי, תוך שמירה על פרטיות, אבטחה ומודולריות.

בדיקות (Testing)

בדיקות המערכת מהוות שלב מרכזי בתהליך הפיתוח, ונועדו לוודא כי כלל רכיבי המערכת פועלים בהתאם לאפיון, עומדים בדרישות הפונקציונליות והלא-פונקציונליות, ומספקים חוויית שימוש יציבה ובטוחה. הבדיקות בפרויקט בוצעו בשלבים שונים, תוך התמקדות הן בצד הלקוח והן בצד השרת.

הבדיקות בוצעו בסביבה מבוקרת, תוך שימוש בדפדפנים מודרניים ובכלי ניתוח ייעודיים ל-WebRTC.

בדיקות צד לקוח (Client Testing)

בדיקות חיבור מצלמה ומיקרופון

- בדיקת גישה למצלמה ולמיקרופון לאחר מתן הרשאות
- בדיקת תגובה במקרה של סירוב הרשאות
- בדיקת החלפת התקני קלט בזמן ריצה

בדיקות יצירת חיבור WebRTC

- בדיקת יצירת RTCPeerConnection בין שני משתתפים
- בדיקת יצירת Mesh בחדרים עם מספר משתתפים
- בדיקת קבלת זרמי וידאו וקול מכל Peer

בדיקות בקרי מדיה

- בדיקת השתקת מיקרופון והחזרתו לפעולה
- בדיקת כיבוי והפעלת מצלמה
- בדיקת שיתוף מסך והפסקתו
- בדיקת יציאה מהחדר וסגירת חיבורים

בדיקות צד שרת (Server Testing)

בדיקות Signaling

- בדיקת חיבור WebSocket מאובטח (WSS)
- בדיקת שליחת וקבלת הודעות SDP
- בדיקת העברת ICE Candidates בין לקוחות
- בדיקת ניתוב הודעות בין משתתפים בחדר

בדיקות ניהול חדרים

- בדיקת יצירת חדר חדש
- בדיקת הצטרפות ויציאה מחדר
- בדיקת עדכון רשימת משתתפים בזמן אמת

בדיקות ניהול משתמשים ואימות

- בדיקת תהליך הרשמה (Sign Up)
- בדיקת שליחת אימייל לאימות משתמש
- בדיקת חסימת התחברות למשתמש לא מאומת
- בדיקת תהליך התחברות (Login)
- בדיקת הרשאות Host לעומת Participant

בדיקות אבטחת מידע

- בדיקת הצפנת תקשורת (WebSocket (WSS
- בדיקת קיום חיבור HTTPS בצד הלקוח
- בדיקת הצפנת מדיה באמצעות DTLS-SRTP
- וידוא שאין מעבר מדיה דרך השרת

בדיקות אלו בוצעו באמצעות:

- Chrome DevTools
- WebRTC Internals
- ניתוח תעבורת רשת

בדיקות תרחישי קצה (Edge Cases)

- הצטרפות משתמש לחדר קיים בזמן שיחה פעילה
- יציאת משתמש פתאומית מהחדר
- נפילת חיבור רשת והתאוששות
- מעבר אוטומטי ל-TURN בעת חסימת P2P
- ריבוי משתתפים ויצירת Mesh מלא

בדיקות תשתית Machine Learning

- בדיקת קליטת פריימים מהמצלמה
- בדיקת עיבוד פריימים בזמן אמת
- בדיקת השפעת העיבוד על קצב הפריימים
- בדיקת יציבות המערכת בעת הפעלת פילטרים

בדיקות אלו נועדו לוודא שעיבוד הווידאו אינו פוגע ביציבות התקשורת.

סיכום הבדיקות

תהליך הבדיקות הוכיח כי המערכת פועלת בהתאם לאפיון, מספקת חיבורי תקשורת יציבים, שומרת על אבטחת מידע ופרטיות, ומתמודדת בהצלחה עם

תרחישי שימוש שונים. שילוב בדיקות פונקציונליות, בדיקות אבטחה ובדיקות ביצועים תרם ליציבות הכללית של המערכת.

הוראות התקנה והפעלה

פרק זה מתאר את שלבי ההתקנה, ההגדרה וההפעלה של המערכת, ומיועד למשתמשים המעוניינים להריץ את המערכת בסביבה מקומית לצורכי בדיקה, פיתוח או שימוש.

המערכת מבוססת על טכנולוגיות Web ושרת Signaling קל משקל, ואינה דורשת התקנה של רכיבי תוכנה כבדים או תשתיות מיוחדות.

דרישות מערכת

צד לקוח

- דפדפן מודרני התומך ב-WebRTC (Google Chrome / Mozilla Firefox)
- מצלמת רשת ומיקרופון
- חיבור אינטרנט פעיל

צד שרת

- Python 3

- גישה לרשת לצורך חיבורי WebSocket
- מערכת הפעלה Windows או Linux

התקנת צד שרת (Signaling Server)

1. יש לוודא ש-Python 3 מותקן במערכת
2. להוריד את קבצי השרת מתיקיית הפרויקט
3. להתקין חבילות נדרשות (במידת הצורך)
4. להפעיל את שרת ה-Signaling באמצעות פקודת הרצה ייעודית
5. לוודא שהשרת מאזין לחיבורי WebSocket מאובטחים (WSS)
6. לפתוח port forwarding

לאחר הפעלת השרת, הוא מוכן לקבל חיבורי לקוחות.

הפעלת צד לקוח

1. לפתוח את קבצי ה-Client בדפדפן
2. לאחר גישה למצלמה ולמיקרופון
3. לוודא חיבור תקין לשרת ה-Signaling
4. להתחבר למערכת באמצעות משתמש קיים או ליצור משתמש חדש

הפעלת המערכת בסביבה מאובטחת

המערכת פועלת באמצעות HTTPS ו-WSS, ולכן יש להפעיל אותה בסביבה מאובטחת או מקומית התומכת בפרוטוקולים אלו. הפעלה זו נדרשת לצורך גישה ל-WebRTC בדפדפנים מודרניים.

תקלות נפוצות ופתרון

- אין גישה למצלמה – יש לבדוק הרשאות בדפדפן
- לא נוצר חיבור וידאו – לבדוק חיבור רשת ושרת Signaling
- משתתף אינו שומע קול – לבדוק הגדרות מיקרופון ורמקולים
- חיבור נופל – לבדוק מעבר אוטומטי ל-TURN

מדריך למשתמש

פרק זה מתאר את אופן השימוש במערכת מנקודת מבטו של משתמש הקצה, ומסביר כיצד לבצע פעולות עיקריות בצורה פשוטה וברורה.

הרשמה למערכת

1. לפתוח את מסך ההרשמה
2. להזין כתובת דואר אלקטרוני וסיסמה
3. לאשר את ההרשמה באמצעות קישור שנשלח לדוא"ל
4. להתחבר למערכת לאחר האימות

התחברות למערכת

1. להזין כתובת דואר אלקטרוני וסיסמה
2. ללחוץ על כפתור התחברות
3. במקרה של שגיאה – לבדוק פרטים או הודעת מערכת

יצירת חדר חדש

1. לאחר התחברות, לבחור באפשרות "יצירת חדר"

2. החדר ייווצר אוטומטית
3. המשתמש יוגדר כ-Host
4. ניתן לשתף את קישור החדר עם משתמשים נוספים

הצטרפות לחדר קיים

1. להזין מזהה חדר או קישור
2. לאשר הצטרפות
3. להתחיל שיחה לאחר יצירת החיבורים

שימוש במהלך שיחה

- הפעלה / כיבוי מצלמה
- הפעלה / השתקת מיקרופון
- שיתוף מסך
- שליחת הודעות בצ'אט
- יציאה מהחדר

ניהול חדר (Host)

- צפייה ברשימת משתתפים
- ניתוק משתתף
- שליטה בהרשאות בסיסיות

סיום שיחה

1. לחיצה על "יציאה מהחדר"
2. סגירת החיבורים
3. חזרה למסך הראשי

סיכום מדריך המשתמש

המדריך מאפשר שימוש פשוט וברור במערכת, גם למשתמשים ללא ידע טכני. שילוב של ממשק אינטואיטיבי עם תיעוד ברור תורם לחווית שימוש חיובית ולתפעול תקין של המערכת.

סיכום ומשוב

פרויקט זה מהווה מימוש מלא של מערכת תקשורת בזמן אמת מבוססת WebRTC, המשלבת ידע תיאורטי ומעשי בתחומי רשתות מחשבים, אבטחת מידע, פיתוח Web ועיבוד וידאו חישובי. במהלך העבודה על הפרויקט תוכננה ונבנתה מערכת שלמה מקצה לקצה, הכוללת צד לקוח, צד שרת, ניהול משתמשים, ניהול חדרים, הצפנת מדיה ותשתית לעיבוד וידאו מתקדם.

אחת המטרות המרכזיות בפרויקט הייתה להבין לעומק כיצד פועלות מערכות תקשורת בזמן אמת, ולא להסתפק בשימוש בספריות או שירותים מוכנים. בחירה זו הובילה למימוש עצמאי של שרת Signaling, ניהול חיבורי Peer-to-Peer בארכיטקטורת Mesh, והיכרות מעמיקה עם פרוטוקולי WebRTC, ICE, STUN, TURN ו-SDP.

בנוסף, הפרויקט כלל פיתוח תשתית לעיבוד וידאו מבוסס Machine Learning בצד הלקוח, בגישה אלגוריתמית נמוכה וללא שימוש בספריות חיצוניות ייעודיות. תשתית זו מדגימה הבנה של עקרונות עיבוד תמונה, ניתוח פריימים בזמן אמת ושיקולי ביצועים, ומהווה בסיס להרחבות עתידיות של המערכת.

במהלך הפיתוח ניתנה חשיבות רבה לאבטחת מידע ולשמירה על פרטיות המשתמשים. המערכת תוכננה כך שתעבורת המדיה תועבר ישירות בין המשתתפים בתקשורת מוצפנת מקצה לקצה, ללא אחסון או עיבוד מדיה בצד השרת. גישה זו מצמצמת את משטח התקיפה ומחזקת את רמת האבטחה הכוללת.

תהליך העבודה על הפרויקט תרם לפיתוח מיומנויות תכנון, ארכיטקטורה, ניתוח בעיות והפרדת שכבות מערכת. ההתמודדות עם אתגרים כגון ניהול חיבורים מרובים, תרחישי קצה ברשת ועיבוד וידאו בזמן אמת חיזקה את ההבנה ההנדסית ואת היכולת לפתח מערכות מורכבות בצורה מסודרת ואחראית.

לסיכום, הפרויקט השיג את מטרותיו ואף מעבר לכך. הוא מדגים מערכת תקשורת בזמן אמת יציבה, מאובטחת ומודולרית, ומהווה בסיס ראוי להמשך פיתוח, הרחבה ושיפור. העבודה על הפרויקט תרמה רבות לידע, לניסיון ולבשלות המקצועית, והיותה תהליך למידה משמעותי ומעמיק.