Noteify

שם : חני מורגנבסר

ת.ז : 328176138

סמינר : בנות בת שבע - מודיעין עילית

המנחה : גב' רחל טוג

תאריך הגשה : 06/2025

תוכן עניינים

1.הצעת הפרויקט:

חילוץ תווים משיר

1.1 תיאור הפרויקט:

מטרת הפרויקט היא לפתח אפליקציה לחילוץ תווים משיר. בעידן הטכנולוגי של היום, כל אחד יכול ללמוד בקלות ומהבית לנגן. גם כמות השירים שיוצאת גבוהה. ולכאורה כל אחד שיודע לנגן ירצה לדעת לנגן גם את הלהיטים החדשים. הבעיה היא שאי אפשר להוציא כל יום ספרים חדשים עם כל התווים לשירים החדשים שיצאו לאחרונה. זה מסרבל, דורש מקום אחסון וכמובן – כסף...
האפליקציה תאפשר למשתמשים להעלות קובץ אודיו של שיר או להזין אותו ישירות למערכת, ובאמצאות טכנולוגיות מתקדמות בעיבוד אותות וזיהוי תבניות המערכת תפענח את התווים הנמצאים בשיר ותציג אותם למשתמש בצורה יפה ומסודרת כך שמה שיישאר לו זה פשוט לנגן...
האפליקציה תחסוך זמן רב, כסף ומקום, וכך יוכלו חובבי המוזיקה לנגן את השירים האהובים עליהם בצורה קלה, נוחה ונכונה.

1.2 תיאור הבעיה האלגוריתמית:

האתגר המשמעותי בפרויקט זה הוא לפתח אלגוריתם יעיל לחילוץ תווים משיר, תהליך שמצריך שילוב של מספר טכנולוגיות מתקדמות מעולם עיבוד האודיו וזיהוי התבניות.
אחת הבעיות המרכזיות היא עיבוד קלט האודיו, שבו האלגוריתם נדרש לנתח את האות הנכנס על מנת לזהות את התדרים המוזיקליים השונים. ניתוח זה הוא קריטי להצלחת החילוץ שכן כל השלבים הבאים מסתמכים עליו וכל שגיאה בשלב זה עשויה להוביל לתוצאות שגויות ולפגוע בדיוק התוצאה.
לאחר תהליך העיבוד יש צורך בזיהוי התווים עצמם משימה מורכבת הדורשת התחשבות בפרמטרים שונים כמו קשרים ומיקומים בין התווים, אופי וקצב המוזיקה.
בעיה נוספת היא הגיוון בסגנונות המוזיקליים השונים ובמגוון הז'אנרים הרחב כמו קלאסי, פופ, רוק ועוד.
לסיכום, כדי שהאלגוריתם יהיה אפקטיבי יש מספר דרישות מרכזיות. ראשית, היכולת לזהות תווים מתוך מגוון רחב של סגנונות מוזיקליים, מה שיבטיח שהאפליקציה תהיה שימושית עבור כל המוזיקאים. שנית, יצירת ייצוג חזותי ברור ומסודר של התווים המופקים כך שהמשתמשים יוכלו להבין את החומר בקלות ובנוחות.

1.3 רקע תאורתי:

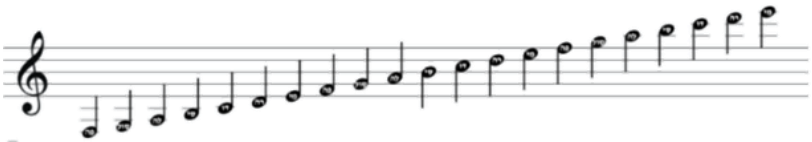
מוזיקה היא אומנות סידור הצליל והשקט בזמן.
מרכיביה העיקריים הם גובה הצליל (האחראי על מלודיה והרמוניה), קצב (מפעם, משקל וארטיקציה), דינמיקה, גוון ומרקם.
האספקטים המוסכמים והמוגדרים אשר מרכיבים כל צליל וצליל שאנו שומעים, כותבים או מנגנים במוזיקה הם גובה(תדר) משך, צבע, גוון ועוצמה או משרעת של הצליל.(בנוסף לאלה אנו שומעים גם צלילים עיליים והרמוניות.).
תדר, כאמור הוא גובה הצליל, הצלילים נעים בין נמוכים לגבוהים. ישנם צלילים שלהם אין גובה מוגדר. משך הוא זמן אורך הצליל. צבע וגוון הם תכונות המיוחסות להבדל בין צלילים המפיקים כלי נגינה ובני אדם שונים.(גם השקט הוא אספקט של מוזיקה, בהיותו שקט מדוד בין שני צלילים. למעשה, כל צליל שאנו שומעים בוקע מתוך ה"שקט" של מקום האזנה.)

הדרך שבה מועברת מוזיקה ממוזיקאי למוזיקאי היא בדרך כלל בתווים. התווים המוכרים כיום החלו להתפתח באירופה במאה ה-14 מן הניומות והתווים המרובעים שקדמו להם. בדפי התווים מצוינים כל האספקטים המוזיקליים שהוזכרו קודם-גובה, משך, עצמה, צבע וגוון. לעיתים מצוינים בתיווי גם דברים נוספים כמו קצב, אופי מוזיקה, אוירה או מהלך הרמוני.

תו מוזיקלי הוא סימן גרפי מוסכם המסמן צליל. בימינו, תווים נכתבים על חמשה כעיגולים בגבהים שונים-כל עיגול מציין את גובהו ומשכו של צליל יחיד.(מלבד התווים עצמם המציינים את הצלילים כתב התווים כולל עוד סימנים רבים המורים על אופן ביצוע המוזיקה במובנים של עצמת הנגינה, מהירותה וכלי הנגינה) ניתן לנגן בעזרת תווים במגוון כלים שונים ביניהם גיטרה, פסנתר ועוד.

המוזיקה המערבית מחלקת את האוקטבה ל12 צלילים ישנם שבעה צלילים "טבעיים" ועוד חמישה צלילים שנחשבים כשינוי של הצלילים הטבעיים שהם הדיאזים והבמולים. שמות תווי הביניים משתנים לפי סוג הסולם. סולם הוא סדרה מוגדרת של צלילים שגובהו של כל צליל בה מוגדר לפי גודל המרווח המוזיקלי בינו לבין צליל בסיס נתון. לכל סולם יש אופי מוזיקלי ייחודי.

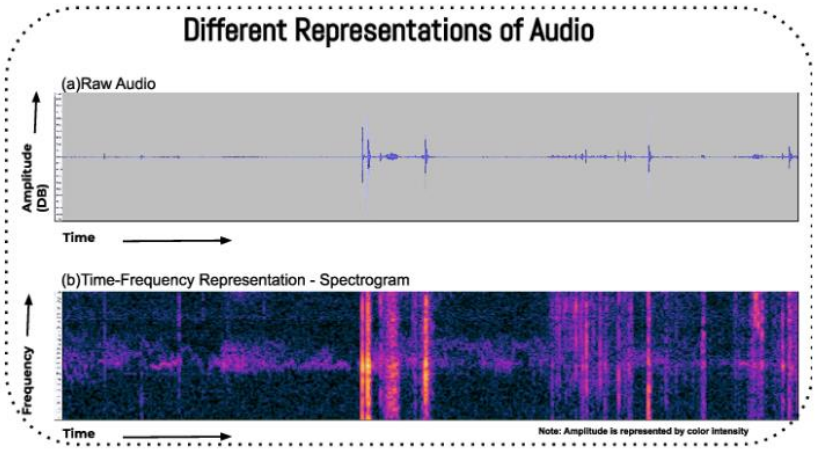
אף על פי שסולמות מרחבי העולם שונים בדרך כלל, המרווחים המוזיקליים בכל סולם מוגדרים לרוב על פי היחסים המתמטיים שבין התדירויות של כל אחד מצליליו.



הגבהה של חצי טון (למשל מדו לדו-דיאז) מחושבת על ידי הכפלתו פי:$\sqrt[12]{2}$ =1.059463094 (שורש 12 של 2) באשר החישוב מתחיל בדרך כלל מ442 הרץ לתו לה. הגבהה של טון שלם תהיה הכפלה פעמיים של חצי טון. התדירויות של התו מוכפלת במעבר לאוקטבה הבאה. הרץ(Hz) הוא יחידת מידה של SI לתדירות. הגדרת ההרץ היא -Hz=S^ 1שמשמעותו "אירוע אחד לשנייה" (100 הרץ משמעותו "100 אירועים לשנייה".)

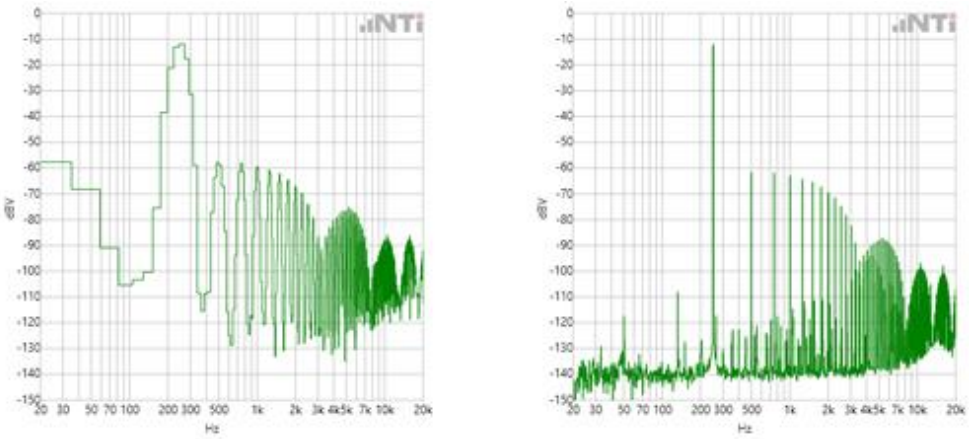
מוזיקה ספקטרלית המוגדרת בשפה טכנית, היא פרקטיקה מוזיקלית אקוסטית שבה החלטות הקומפוזיציה (מבנה של יצירה מוזיקלית) ניתנות לרוב על ידי ייצוגים סטנוגרפים וניתוח מתמטי של ספקטרום קול או על ידי ספקטרום שנוצר מתמטי. הגישה הספקטרלית מתמקדת במניפולציה של התכונות הספקטרליות חיבור ביניהן והפיכתן. בניסוח כזה, ניתוח צליל מבוסס מחשב וייצוגים של אותות אודיו מטופלים כאנלוגיים לייצוג טרמבלי של צליל.

ספקטוגרפים :כלי זה מציג את הספקטרום של אות אודיו על פני ציר הזמן, ומראה כיצד תדרים משתנים לאורך זמן.

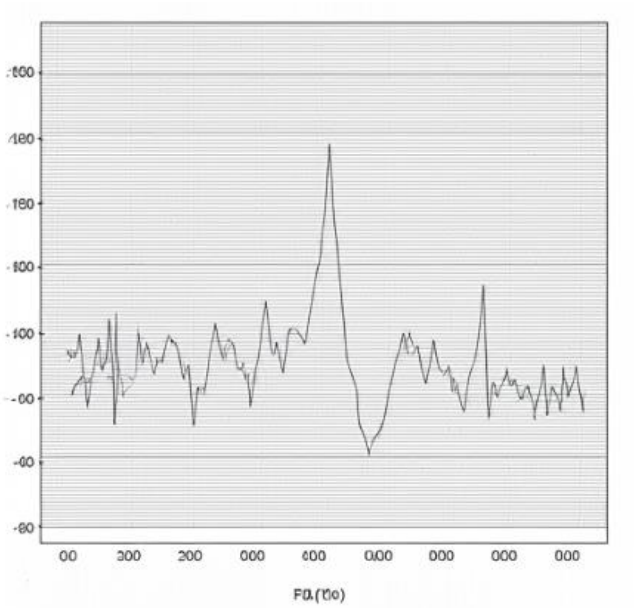


<u>ישנם כמה אפשרויות יעילות של ניתוח הספקטוגרפים. נציג כמה מהן:</u>

1. **התמרת פורייה** היא התמרה אינטגרלית המשמשת בכלי מרכזי באנליזה הרמונית. התמרת פורייה היא פרוק של פונקציה לרכיבים מחזוריים וביצוע אנליזה מתמטית לפונקציה על ידי ניתוח רכיביה. (מאפשרת לפרק צליל מורכב לתדרים הבסיסיים המרכיבים אותו.) בשל כך ניתן לראות את ההתמרה בתור מיפוי בין מרחב הזמן למרחב התדר.
התמרת פורייה מהווה כלי חשוב בניתוח צלילים משום שצליל צלול (תו בתדר בודד) הוא למעשה גל קול המתנודד בזמן בתדר מסוים. באמצעות התמרת פורייה, ניתן לראות את הספקטרום של הצליל בזמן נתון, המייצג את התדרים השונים ואת העוצמה שלהם. ההתמרה מאפשרת לנתח צלילים ולבודד את התדרים המרכיבים אותם. באופן כללי יותר, התמרת פורייה מאפשרת לאתר רכיבים מחזוריים בתוך פונקציה ולכן יש לה שימוש נרחב בניתוח אותות ובעיבוד תמונה.
התמרת פורייה מהירה (Fast Fourier Transform- FFT) היא אלגוריתם שמאפשר לחשב את ההתמרה של פורייה בצורה מהירה ויעילה יותר מאשר השיטות המסורתיות. ההתמרה של פורייה היא טכניקת ניתוח שמפרקת אות מורכב לתדרים המרכיבים אותו, אך חישוב ההתמרה באופן ישיר יכול להיות מאוד מסובך ודורש זמן רב, במיוחד כאשר האות הוא באורך גדול.
ה-FFT עובד על עקרון החלוקה והכיבוש, כאשר הוא מחלק את האות למקטעים קטנים יותר. כל מקטע נבחן בנפרד, מה שמאפשר חישוב מהיר יותר. לאחר חישוב התדרים בכל מקטע, התוצאות משולבות מחדש כדי לקבל את התמונה המלאה של התדרים והעוצמות.
ניתוח התדרים מתבצע באמצעות אלגוריתם FFT, המפיק ספקטרום של תדרים. תדרים בולטים נבחרים לצורך המרה לתווים. כל תדר מוכר ליחסו לתו מוזיקלי, לדוגמה התדר של "דו" הוא 261.63 הרץ. התווים המתקבלים נכתבים ומעובדים לכלים מוזיקליים שונים.
מעלותיו של אלגוריתם FTT הם יעילות החישוביות הגבוהה – האלגוריתם מציע שיפור משמעותי בזמן החישוב בהשוואה לשיטות המרה ישנות. בנוסף, בשל יעילותו הוא משמש במגוון רחב של תחומים כמו: עיבוד תמונות, עיבוד אודיו, תקשורת, ניצוח נתונים ועוד. וכן הוא משמש כבסיס לשיטות רבות אחרות בעיבוד אותות כגון פילטרים דיגיטליים, חלונות ואינטרפולציה.
לאלגוריתם FTT יש חסרון של מגבלת אורך האות, מכוון שבאותות ארוכים במיוחד הוא עלול להציג שגיאות עיגול שבל הטבע הנומרי של החישובים, וכן הוא דורש כמות זיכרון גדולה עבור אות באורך גדול.



ייצוג ה-FFT של אות עם אורך בלוק קטן וגדול



2. Average Value Estimation (AVE)

AVE מתבסס על עקרונות פשוטים של חיבור וחלוקה. במקום לפרק אותות לתדרים קבועים כמו בFFT, נעשה שימוש בגלים קצרצרים שמתארים את האותות בצורה מקומית. הגלים הללו משתנים באורך שלהם, מה שמאפשר לנתח תדרים ברמות שונות של פירוט.

AVE מאפשרת לפצל את האות לתדרים נמוכים וגבוהים, ובכך ניתן לראות את הפרטים השונים של האות בכל רמה, זה יעיל במיוחד בניתוח אותות עם תכנים משתנים כמו מוזיקה או דיבור.

אחת היתרונות של AVE - היא מספקת מידע גם על זמן וגם על תדר. כלומר, אפשר לראות מתי מתרחשים שינויים בתדרים. זה קריטי בניתוח מוזיקה שבה תדרים משתנים מהר. שימושי לנורמליזציה של עוצמה בין קטעים שונים, או לזיהוי קטעים שקטים במיוחד.

אך מנגד היא רגישה מאוד לרעש, מה שמצוי הרבה אצל שירים. היא לא יודעת לזהות בין סוגים שונים של אותות מוזיקליים כמו כלי נגינה שונים וסגנונות מוזיקליים שונים. והחיסרון המשמעותי במיוחד לפרויקט זה בו אנו צריכים לזהות אילו תדרים מרכיבים את האות, הוא ש AVEמספק אומנם מידע על העוצמה הממוצעת, אך אינו מספק מידע על התדרים המרכיבים את האות.

3. Wavelet Transform

ניתוח באמצעות גלי וויבלטים מציע יתרון על פני FFT בכך שהוא מספק ניתוח ברמות פירוט שונות בזמן ובתדר. זה מאפשר זיהוי שינויים תדריים ומורכבות של האות, מה שיכול להיות קריטי במוזיקה שבה תדרים משתנים במהירות. היא מאפשרת ניתוח מעמיק של אותות מוזיקליים, זיהוי תכונות מורכבות ודחיסה יעילה. עם זאת, יש להבין את המורכבות שלה ואת האתגרים הכרוכים בשימוש בה כמו מורכבות גבוהה בחישוב מה שמאט את קצב הביצועים שלה, במיוחד בהשוואה להתמרת פורייה. וכן היא דורשת התאמה של הפרמטרים עבור כל אות עקב רגישותה לתנאי השפה כמו אורך האות וסוג פונקציית הגל.

4. Convolutional Neural Networks (CNN)

רשתות עצביות קונבולוציוניות מתאימות במיוחד למשימות זיהוי תבניות מורכבות באותות. הן יכולות ללמוד ולהפיק תובנות מהנתונים, ולספק דיוק גבוה בזיהוי תווים באודיו. השימוש בלמידת מכונה מאפשר ל CNN-להתמודד עם וריאציות שונות בסגנון המוזיקה ובאיכות האות . ללמידת מכונה אמנם יתרונות רבים כמו זיהוי תכונות אוטומטי וביצועים גבוהים, אך עם זאת יש לה חסרונות כמו דרישה לכמות נתונים גדולה בכדי ללמוד בצורה יעילה, כוח חישוב

גבוה מה שעלול להגביל את השימוש בהם במכשירים ניידים, בחירת ההיפר-פרמטרים הנכונים (כגון גודל המסננים, מספר השכבות וכו') יכולה להיות מורכבת והשפעה משמעותית על הביצועים.ובסוף, קשה לפעמים להבין את ההחלטות של CNN, מה שעלול להקשות על הערכת האמינות שלהן .

1.4 הליכים עיקריים בפיתוח הפרויקט:

1. קליטת קובץ אודיו מהמשתמש – אפשרות למשתמש להעלות קובץ אודיו מסוגים שונים כמו (WAV,3MP , וכו') . המרת קובץ האודיו לפורמט קובץ אחיד בשביל נוחות העיבוד.

2. עיבוד האותות – חילוץ תדרים-ניתוח האותות באמצעות אלגוריתם FFT בכדי לחלץ את התדרים המוזיקליים המרכיבים את הצליל.

עבור כל חלון יש להפעיל את האלגוריתם FFT כדי להעביר את האות מתחום הזמן לתחום התדר. התוצאה של הFFT היא מערך של תדרים עם הערכים המוחלטים שמייצגים את עצמת התדרים השונים בזמן נתון. כל תדר מייצג את הצליל או התו המוזיקלי.

3.זיהוי התדרים - זיהוי התדרים הבולטים ביותר בכל חלון והתמקדות בהם.

4.זיהוי התו המוזיקלי – המרה של כל תדר שזוהה לתו המוזיקלי על פי חישוב של התדר (למשל, תדר 440 Hz יתורגם לתו A4).

אם יש יותר מתדר אחד באותו חלון זמן, המערכת יכולה לזהות אקורד.

5.הצגת התוצאה למשתמש – המערכת תציג את התווים או האקורדים בהתאם לתוצאה שהתקבלה. אפשרות גם להציג את הגרף שהתקבל מהFFT שמציג את התדרים והעוצמה לאורך זמן.

1.5 תיאור טכנולוגיות הנדסה:

MATLAB

1.6 תיאור פרוטוקולי תקשורת:

HTTP

1.7 לוחות זמנים:

אוקטובר – ספטמבר : חקר על הנושא
נובמבר : חקר אלגוריתם FFT והיישום שלו בעיבוד אותות אודיו.
דצמבר – ינואר : מימוש וכתיבת האלגוריתם לעיבוד אותות (FFT) וזיהוי התווים.
מרץ : פיתוח ממשק משתמש להצגת התווים.

מנחת הפרויקט: רחל טוג
חתימת סטודנט:

2. תקציר/מבוא

2.1 הרקע לפרויקט

מוזיקה תמיד באה לנו בטוב, היא יכולה להוציא מאיתנו הכל. בין אם זה רגש או שמחה, עצבות וכל חוויה. מוזיקה יודעת פעמים רבות לבטא את הרגשות שלנו בצורה הטובה ביותר.

אך מה אם אנחנו רוצים להוציא מאיתנו מוזיקה? כן, לתת לנפש לשפוך ולהחליק על הקלידים או המיתרים ופשוט -לנגן?

בשאלה הזו נתקלתי הרבה פעמים כשעמדתי מול הפסנתר ורציתי לנגן, אבל לא ידעתי איך. תווים לקרוא אני יודעת ואפשר בכל מקום וזמן ללמוד. אך מאיפה יהיה לי עכשיו את התווים ללהיט האחרון שיצא ממש אתמול?

למה יש פלטפורמות רבות של שירים שכל מה שצריך זה לכתוב את שם השיר, ואילו בשביל חובבי הנגינה אין איזו אפליקציה כזו שתכתוב להם את התווים?

מהצורך הזה נולד הרעיון לפרויקט שלי.

אפליקציה קלה וידידותית שכל שעל המשתמש לעשות הוא פשוט להעלות את המנגינה שהוא רוצה לנגן, להתיישב ליד הפסנתר, וזהו.

מכאן גם הגיע הרעיון לשם הפרויקט Noteify שם שמתבסס על אפליקציית Spotify אפליקציה שכולנו מכירים שבה אפשר למצוא כל שיר שרק רוצים. רק שהתאמתי אותו לפרויקט שלי-Note – תו ו- Identify – לזהות.

2.2 תהליך המחקר

בתחילה חקרתי על התאוריה של המוזיקה הכתובה – התווים. מתי התחילו לכתוב תווים ולמה, איך החליטו על הכללים לכתיבתם, ומה עומד מאחורי השיטה.
בהמשך התעמקתי במונחים מוזיקליים כמו סולמות, תדרים ואוקטבות. איך כל תדר משפיע בצורה שונה על המנגינה ועל ההבדלים ביניהם.

לאחר שהבנתי ממה מורכב הצליל ואיך נוצרת מנגינה, בדקתי כיצד ניתן להציג צלילים בשפה טכנית. בתכנות, מציגים את הצלילים על ספקטוגרמה שהיא כלי המציג את הספקטרום (מילה כללית המשמשת בתחומים רבים לתאר בדרך כלל מצב הקשור ברציפות כלשהי.) של אות אודיו על פני ציר הזמן, ומראה כיצד תדרים משתנים לאורך זמן.

כעת, לאחר שהבנתי איך לייצג צליל בצורה טכנית, התעמקתי בנושא יצירת ספקטרום התדרים מהצליל, המספק מידע על הרכב תדריו ועוצמתו של כל גל.
גיליתי, שעל מנת למצוא את ספקטרום התדרים של אות מסוים יש מספר אלגוריתמים מוכרים, אך הבולט שלהם היה התמרת פורייה שהיא התמרה אינטגרלית המשמשת בכלי מרכזי באנליזה הרמונית. התמרת פורייה היא פירוק של פונקציה לרכיבים מחזוריים (סינוסים וקוסינוסים או לחלופין אקספוננטים מורכבים) וביצוע אנליזה מתמטית לפונקציה על ידי ניתוח רכיביה. בשל כך ניתן לראות את ההתמרה בתור מיפוי בין מרחב הזמן למרחב התדר. שיטה זו פותחה על ידי ז'אן-בטיסט ז'וזף פורייה וקרויה על שמו.

לאחר שידעתי על איזה אלגוריתם להתבסס, חיפשתי שפה אשר תדע להתנהל מולו בצורה המיטבית ביותר.
קיבלתי הפניות והמלצות על שפת MATLAB שהיא שפה אשר שייכת לחברת MathWorks. שפה זו היא שפה דור רביעי בתחום המתמטי הפנימי. התוכנה מאפשרת טיפול קל ונוח במטריצות, שימוש בפונקציות ובנתונים, מימוש אלגוריתמים על נתונים, יצירת ממשקי משתמש ויצירת קשר עם תוכנות הכתובות בשפות אחרות. מה שאופטימלי לצורך הפרויקט שהרי ייצוג הספקטוגרמה היא לבסוף כמטריצת מספרים – תחום התדר ותחום הזמן. שפה זו גם נותנת אפשרות להצגת גרפים ותרשימים שונים בצורה קלה וידידותית.

אז יש לי ידע, אלגוריתם ושפה. כל שנשאר הוא "רק" לכתוב את הקוד...

2.3 סקירת ספרות

במהלך עבודתי על הפרויקט חקרתי קראתי והעמקתי במגוון אתרים, תחומים ומקומות שונים.

תיאורית המוזיקה, פתרונות לבעיות, מונחים ורקע היסטורי:

- Wikipedia- Glossary of music terminology - הסבר תאורתי על המוזיקה והבנת הנושא, השתמשתי בעיקר כשחקרתי על נושא הפרויקט בהתחלה והכנתי את ההצעת פרויקט.
- Wikipedia- Music theory - הסבר תאורתי על המוזיקה והבנת הנושא
- Wikipedia- Music theory2 – הסבר תאורתי על המוזיקה והבנת הנושא
- YouTube – סרטונים עם הסבר על תאוריית המוזיקה וממה מורכב הצליל, עזר לי להבין מה הבעיה כשנתקלתי בתהליך ש"המציא" לי תווים שלא קיימים.
- IEEE – אתר של מאמרים שונים שהתבססתי עליהם במציאת פתרונות לביות שהתעוררו במהלך העבודה.

חקר האלגוריתם FFT :

- Wikipedia- FFT – הסבר על אלגוריתם FFT בויקיפדיה, השתמשתי בעיקר בהבנת הנושא ובהכנת הצעת הפרויקט.
- vlib.eitan.ac.il – לימוד על אלגוריתם של התמרת פורייה מהירה
- www.nti-audio.com

אתר , MATLAB קורסים מקוונים בקצב עצמי ומידע על שפה זו:

- Wikipedia- MATLAB – רקע ומידע על תוכנת מאטלאב
- MATLAB – האתר הרשמי של מאטלאב. ממנו הורדתי את התוכנה
- MATLAB courses – קורסים מקוונים בקצב עצמי של מאטלאב שעשיתי לצורך לימוד התוכנה והשימוש בה
- www.systematics.co.il – הסבר על אופן ניתוח אותות במאטלאב

פרויקטים בנושא מוזיקה וניתוח אותות, מאמרים ואתרים נוספים:

- chat Gpt
- magenta
- tandfonline
- pubs.aip

- Dixon. S
- researchgate

אתרים שמהם לקחתי מנגינות או תווים:

- tavimilim – תווימילים אתר חינמי שיש בו תווים לשירים מוכרים במילים בעברית
- tavisraeli – אתר של תווים לשירים ישראלים מוכרים בתשלום (בעיקר לצורך למידה מהאתר על דברים חשובים לאפליקציה שלי)
- YouTube – סרטונים ושירים מיוטיוב שהורדתי לצורך הפרויקט

2.4 אתגרים מרכזיים

2.4.1 הבעיות איתן התמודדתי:

- זיהוי תו נכון מתוך אות.
- פיענוח נכון של סוג התו ומשך הזמן שלו.
- הצגה גרפית מקצועית של התווים בצורה ידידותית למשתמש בדפדפן.
- אפשרות ייצוא התווים כקובץ PDF

2.4.2 הסיבות לבחירת הנושא:

- **סיבה מקצועית:** עניין אישי בתחום עיבוד אותות ואודיו, לצד רצון ליישם אלגוריתמים מתקדמים ב MATLAB-בשילוב עם ממשק משתמש מודרני ב.React-
- **סיבה אישית:** אהבה למוזיקה ונגינה, רצון להבין יותר לעומק איך יוצרים מוזיקה וממה היא מורכבת

2.4.3 מוטיבציה לעבודה:

הרצון להנגיש למוזיקאים וכלי ניתוח אוטומטיים לתווים מוקלטים, וכן להוכיח יכולת שילוב בין עולמות תוכן שונים: מתמטיקה, מוזיקה, עיבוד אותות, תכנות צד שרת וצד לקוח.

2.4.4 על איזה צורך הפרויקט עונה ואיזה פתרון הוא נותן:

הפרויקט נותן מענה לצורך בזיהוי תווים מדויק מתוך קבצי אודיו – משימה שדורשת לרוב מיומנות מוזיקלית וציוד מתאים. באמצעות הכלי שפיתחתי, כל משתמש יכול להעלות קובץ שמע ולקבל תווים מדויקים ומוצגים בצורה גרפית, מבלי לדעת לקרוא ספקטרוגרמות או להשתמש בתוכנות מקצועיות יקרות.

2.4.5 הצגת פתרונות לבעיה:

- **שימוש באלגוריתם FFT לניתוח האות:** השתמשתי באלגוריתם FFT שנותן פלט של 2 מערכים שמייצגים כל אחד את תחום הזמן ותחום התדר בהתאמה. לכל פרק זמן, מהו גובה התדר הקיים באות.
- **בעיית זיהוי סוג התו:** מקובל בעולם להתייחס לתו דו A4 כתו בגובה 440 הרץ. כאשר כל תו מחושב כפי 12√2 מהתו שקדם לו. בנוסף יש כמה סוגים של תווים, לכל משך זמן שונה 0 ציור שונה של תו. משך הזמן של כל סוג כזה ידוע וקבוע מראש, ולכן יכולתי לבדוק לפי מערך הזמן – כמה זמן היה כל תדר. , וכך לזהות את סוג התו.
- **בעיית הצגת תווים בצורה גרפית בדפדפן:**נבחנו מספר כלים להצגת תווים, כולל LilyPond ליצירת תווים בפורמט PDF, אך נבחרה לבסוף ספריית VexFlow, שמאפשרת הצגת תווים

בצורה אינטראקטיבית וישירה ב React-, עם אפשרות להתאמה דינמית על פי הנתונים שהתקבלו מהשרת.

- **ייצוא התווים כקובץ PDF :** קיימות מספר שיטות ליצוא תוכן דפדפן כPDF. בפרויקט זה בוצע שימוש בספרית jspdf svg2pdf.js.

3. יעדים ומטרות

יעדים: מטרת הפרויקט היא לפתח מערכת שמסוגלת לזהות תווים מוזיקליים מתוך קובץ אודיו בצורה אוטומטית, ולהציגם באופן גרפי ברור ונגיש, כך שמשתמשים – גם ללא ידע מקצועי – יוכלו להבין את המבנה המוזיקלי של הקלטה כלשהי. ויוכלו לנגן כל שיר שירצו.
המטרה היא לשלב בין ניתוח אודיו מקצועי לבין חוויית משתמש מודרנית, על מנת להנגיש את עולם עיבוד הצליל וההמרה לתווי בצורה קלה ומועילה

מטרות:

- **ניתוח אות שמע באמצעות – MATLAB** שימוש ב FFT-ובספקטרוגרמה לזיהוי תדרים דומיננטיים לאורך זמן.
- **מיפוי תדרים לתווים מוזיקליים** –פיתוח מנגנון לזיהוי התו הקרוב ביותר לתדר שנמצא, כולל חישוב אוקטבה.
- **סינון הרמוניות ורעש** –שימוש באלגוריתם שמזהה תדר בסיסי מתוך כמה פיקים ומנפה כפולות הרמוניות.
- **מיזוג תווים זהים** –זיהוי ואיחוד של תווים שנשמעו בהמשכיות למרות סטיות קלות בתדר ובזמן.
- **סיווג משכי תווים** –קביעה של סוג התו (רבע, שמינית, וכד') לפי משך הזמן בו התו זוהה.
- **כתיבת הפלט לקובץ בפורמט קריא** –יצירת קובץ טקסט בפורמט מסודר (classified_notes.txt)שמכיל את שמות התווים והמאפיינים שלהם.
- **הקמת שרת – Node.js** קבלת קבצי אודיו מהמשתמש, הרצת קוד MATLAB בצד שרת, ושליחה חזרה של הנתונים.
- **בניית ממשק משתמש ב React-** – העלאת קבצים, קבלת תוצאות והצגה גרפית של התווים באמצעות ספריית VexFlow .

4. ממדי הצלחה

בכדי לבדוק אם הפרויקט עונה על המטרות שהוגדרו, הוגדרתי מספר מדדי הצלחה ברורים, כמותיים וברי מדידה:

- **דיוק בזיהוי תווים** –הצלחה בזיהוי לפחות 95% מהתווים האמיתיים בהשוואה לתווים שנכתבו ידנית על ידי מוזיקאי מקצועי, מתוך קובץ אודיו נתון.
- **דיוק בסיווג משך התווים** –לפחות 90% מהתווים שזוהו יסווגו על הזמן הנכון (שמינית, רבע וכו') בהתאם למציאות.

- **זמן תגובה ממוצע של המערכת** –הזמן מרגע העלאת קובץ האודיו ועד קבלת תוצאה גרפית מלאה לא יעלה על 20 שניות (לקובץ באורך של עד דקה).
- **אחוז הצלחה בהצגת התווים בדפדפן** –לפחות 95% מהקבצים המנותחים יוצגו כהלכה בצד הלקוח באמצעות ספריית VexFlow.
- **תמיכה למשתמשים בו-זמנית** –אם תבחרו לפחות 5 משתמשים שפועלים, כל אחד מעלה קובץ אחר, ללא קריסת השרת או ירידה משמעותית בביצועים.

5. אתגרים

במהלך כתיבת הפרויקט התמודדתי עם מגוון אתגרים טכניים, לוגיים וטכנולוגיים עם העבודה. להלן האתגרים המרכזיים:

אתגרים באלגוריתם:

- **הרמוניות בתדרים :**בתחילה זוהו תווים "מומצאים" שלא הופיעו בפועל במנגינה, למשל מהרמוניות בתוצאת ה .FFT-צריך היה להבין איך לאתר את התדר הבסיסי האמיתי ולסנן תדרים כפולים.
- **תווים מנותקים ולא רציפים :**תווים שנוגנו באופן רציף, זוכים נפרדים עקבו שינויים קטנים בתדר ובמשך הזמן. הפתרון דרש פיתוח מנגנון מיזוג מבוסס קרבה בתדר ובטווח זמנים.
- **סיווג משך התווים :**תרגום מדויק של משך הזמן של כל תו לסוג תו מוזיקלי (שמינית, רבע, חצי וכו') הצריך קביעת סף מדויק בין הקטגוריות, תוך התחשבות בשונות בין סוגים שונים של מוזיקה.

אתגרים טכנולוגיים:

- **שילוב בין MATLAB ל :Node.js-**צריך היה ליצור מנגנון שבו שרת Node.js יפעיל קוד MATLAB, יקלוט תוצאה ויעביר אותה ל- React בצורה אמינה. העבודה דרשה שימוש במודול child_process וטיפול בקבצים תוך שמירה על סנכרון תקין.
- **הצגה גרפית של תווים :**איתור ספרייה שתומכת בתווים מוזיקליים בצורה דינמית ואסתטית. לאחר בדיקה עם כלים שונים, נבחרה ספריית VexFlow, שדרשה למידה מעמיקה של המבנה שלה לצורך יצירת תווים לפי פורמט קלט מותאם אישית.

אתגרים בצד השרת והלקוח:

- **ניהול** למחוק קבצים חומר קבצים זמניים(כגון קובץ WAV או פלט ביניים) בכדי למנוע עומס וטעויות עתידיות.
- **ניהול חיבורים מקבילים :**הבטחת יציבות השרת גם כאשר מספר משתמשים מעלים קבצים במקביל, תוך מניעת קונפליקטים בגישה לקבצים זמניים או לקבצי תוצאה.

אתגרים באיסוף ה:Data-

- **היעדר דאטה מוכן מראש :**הפרויקט מתבסס על קלט חי משתמשים – כלומר אין קובץ תווים "אמיתי" להשוואה אוטומטית. היה צורך לבנות מבחני תקינות ידניים, להשוות תוצאה מול תווים ידניים, ולבצע שיפורים מבוססי ניסוי וטעיה.
- **השפעת איכות ההקלטה :**הקלטות באיכות נמוכה או רועשות השפיעו על תוצאות ה- FFT, והקשו על מיפוי מדויק של התדרים לתווים.

אתגרים טכניים:

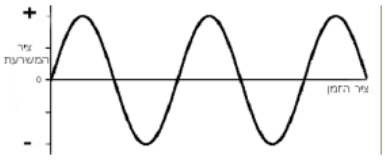
- **התקנת תוכנות:** הפרויקט מתבסס על תוכנת מאטלאב - תוכנה שהיא לא חינמית, דורשת רישיון בתוקף ומוגבלת במספר ההתקנות. בנוסף היא דורשת מעבד חזק ומתקדם מה שמנע התקנה על מחשבים שונים ועבודה על מחשב אחד במקום אחד.
- **עמידה בזמנים:** עבודה על הפרויקט תוך כדי למידה רציפה בנושאים שונים בתכנות, שיעורי בית, פרויקטים נוספים ומטלות לימודיות.

6. רקע תיאורטי

בכדי להסביר את ייצוג אות שמע בצורה ממוחשבת ולהמחיז את הקשר בין תדר וגלי קול נשתמש בצורה האופיינית של גל מחזורי, כגון גל סינוס. גל סינון הוא סוג של גל מחזורי הנע בין ערך מקסימלי למינימלי לאורך זמן. התדירות של גל הסינוס נקבעת לפי מספר המחזורים שהוא משלים בשנייה.

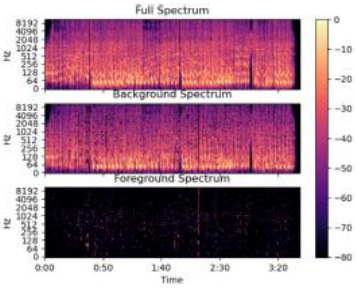
כך ניתן להשתמש בגלי סינוס לייצוג תווים או צלילים בודדים. על ידי שינוי הגובה, התדר של גל הסינוס, ניתן להפיק צלילים שונים התואמים לתווים שונים בסולם מוזיקלי. על ידי שילוב של גלי סינוסים רבים ניתן ליצור צלילים מוזיקליים מורכבים.

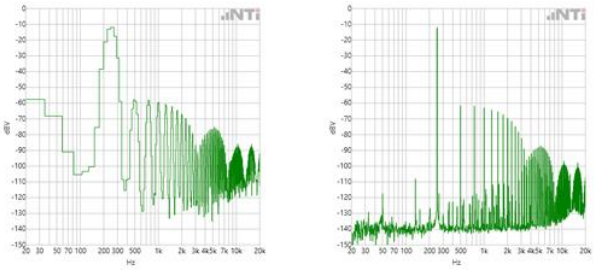
זהו ציור של גל סינוס של תו אחד החוזר על עצמו:



הנה כמה מושגים שיעזרו לנו להבין את ייצוג הגלים:
משרעת – המרחק בין הגל לציר ה-X.
תדירות – מספר הפעמים שהגל חוזר על עצמו ביחידת זמן מסוימת.
הרץ – מידה בה נמדדת התדירות לשנייה.
גל טהור – תו בודד ללא חיבור של גלים נוספים.

דרך נוספת לייצוג שמע היא ע"י ספקטוגרמה. ייצוג של ספקטוגרמה הוא טוב יותר מייצוג של גל סינוס משום שניתן על ידי עיבוד להוציא ממנו יותר מידע. ספקטוגרמה היא גרף המציג את עוצמת האות לאורך זמן עבור טווח תדרים נתון, באמצעות ספקטרום צבעים. הוא מצביע על התדרים בהם





ייצוג ה-FFT גדול וגדול עם אורך בלוק קטן של אות

האנרגיה של האות היא הגבוהה ביותר ומראה את וריאציות האנרגיה לאורך זמן.
בכדי ליצור ספקטוגרמה תוכנת ניתוח אותות מחלקת אות של תחום זמן למקטעים באורך שווה.
לאחר מכן, הוא מחיל את המרת פורייה מהירה FFT על כל מקטע, וממיר את הנתונים מתחום הזמן לתחום התדר. הספקטוגרמה היא התוצאה של הFFT של כל קטע. הנה תמונה של ספקטוגרמה וFFT:

תדרים בולטים בספקטוגרמה נבחרים לצורך המרה לתווים. כל תדר מוכר ליחסו לתו מוזיקלי, לדוגמה התדר של "דו" הוא 261.63 הרץ. התווים המתקבלים נכתבים ומעובדים לכלים מוזיקליים שונים.

7. מצב קיים

תחום זיהוי התווים ממוזיקה עוסק שנים רבות בפיתוח אלגוריתמים שונים לניתוח אודיו והמרתו לתצוגות מוזיקליות. בין הפתרונות הידועים והנפוצים ניתן לראות:

- HPS – Harmonic Product Spectrum - ספקטרום מוצרים הרמוניים
- CEPSTRUM
- YIN Algorithm
- זיהוי גובה צליל מבוסס אוטוקורלציה
- Dynamic Time Warping (DTW) לזיהוי תבניות חוזרות
- תמלול מבוסס למידת מכונה למשל Frames, CREPE, Onsets
- Spectral Flux + Peak Picking לאיתור תחלות תו
- Multi-pitch Estimation using Non-negative Matrix Factorization (NMF)

בנוסף, לאחר חיפוש מעמיק גיליתי שיש רק תוכנה אחת שיודעת להמיר שיר לתווים אך התוצאה קשה להבנה ולנגינה.

8. ניתוח חלופות מערכתי

ניתוח אלגוריתמים:

במהלך הפיתוח נבחנו מספר גישות אלגוריתמיות לזיהוי תווים מתוך אות אודיו. אחת מהן היא אלגוריתם YIN, שמבוסס על ניתוח זמן השהייה (lag) באות ומספק דיוק גבוה במיוחד תוך שמירה על עמידות לרעש – אך דורש מימוש מורכב ומרובה חישובים. גם שיטת CEPSTRUM נבחנה, המתבססת על ניתוח האות בספקטרום הלוגריתמי, אך גיליתי כי היא רגישה מאוד להפרעות ואינה מדויקת כאשר קיימים תווים חופפים. שיטת Autocorrelation, הפשוטה יותר למימוש, התגלתה כרגישה מדי להרמוניות ולרעש, מה שהוביל לטעויות בזיהוי תדרים. אלגוריתם נוסף HPS – (Harmonic Product Spectrum)נמצא כיעיל במיוחד בזיהוי תווים עם הרמוניות ברורות, אך הוא נחלש משמעותית כשמדובר בצלילים פחות מוגדרים או עם הרבה רעש רקע.

כמו כן, נבחנה האפשרות להשתמש בגישות מבוססות למידה עמוקה (Deep Learning), כגון CREPE או Onsets and Frames. גישות אלו הראו תוצאות מדויקות מאוד, גם בקטעים פוליפוניים, אך דרשו זמני ריצה ארוכים, משאבי חישוב כבדים, ואימון על דאטה נרחב.

מן הצד הטכנולוגי, הושוו מספר סביבות פיתוח -MATLAB נבחר בשל חוזקו הרב בניתוח אותות, התמיכה המקיפה שלו ב־ FFT, ובאפשרויות ההדמיה שהוא מציע. לעומתו, Python עם ספריות כמו Librosa ו Essentia-אומנם קוד פתוח ונפוץ בקהילת המוזיקה הדיגיטלית, אך דרש התאמות רבות וביצועיו היו איטיים יחסית. סביבות כמו JUCE ב־ C++ נבחנו לצרכים תעשייתיים עתידיים, אך נמצאו כמורכבות מאוד לפיתוח ראשוני. לבסוף React, נבחר עבור צד הלקוח – להצגת התווים הגרפיים בצורה נוחה ומודרנית, בעוד Node.js סיפק את הגשר לתקשורת עם MATLAB ולניהול קבצים.

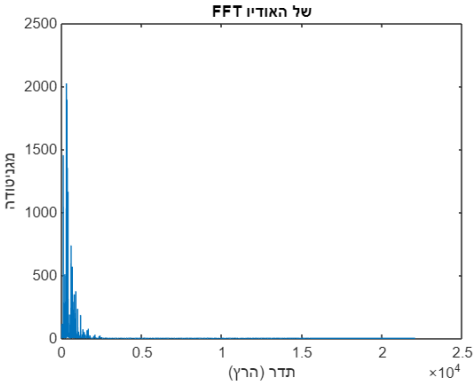
9. תיאור החלופה הנבחרת והנימוקים לבחירתה

לאחר בדיקות והתלבטויות. הוחלט לעבוד עם MATLAB לצד שרת ועם React לצד לקוח. השילוב בין MATLAB – שמתמחה בעיבוד אותות – ל־ React – שמתאים להצגת מידע בצורה אסתטית ודינמית – אפשר איזון בין דיוק ניתוח מוזיקלי לבין ממשק משתמש ידידותי. שיטות מורכבות יותר כמו CREPE או NMF אמנם מדויקות יותר בקטעים פוליפוניים, אך חורגות מדרישות הפרויקט, ודורשות אימון מודלים כבדים, זמני ריצה ארוכים ותלות בדאטה חיצוני.

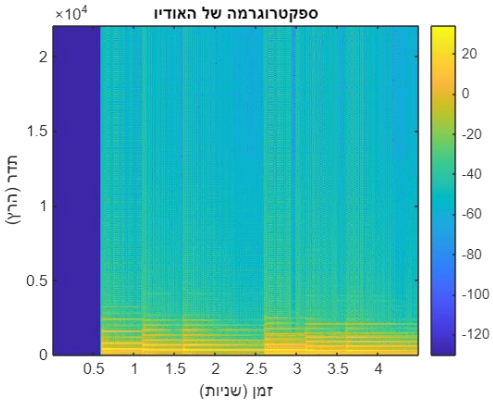
אלו שלבי האלגוריתם:

שלב 1. לאחר העלאה של השיר על ידי המשתמש הוא מומר לקובץ מונו אם צריך.

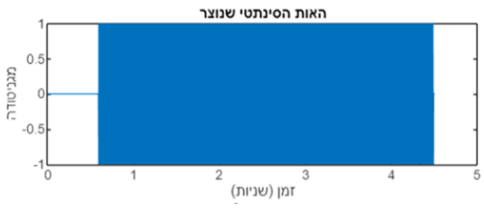
שלב 2. החלת FFT על קובץ השמע: החלת FFT על כל חלון זמן: כל חלון הופך לספקטרום תדרים:

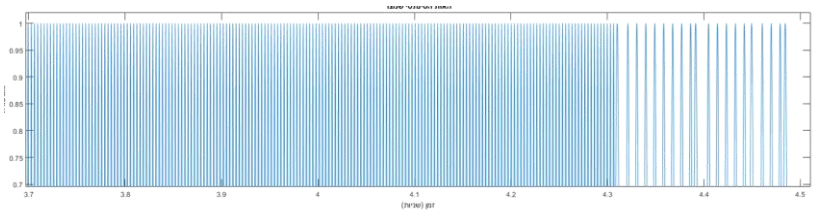


שלב 3. יצירת ספקטוגרמה: הספקטוגרמה מציגה את התדרים השונים של הצלילים לאורך זמן ומספקת תמונה ויזואלית שמאפשרת להבין את המבנה הספקטרלי של הצלילים שנשמעים.

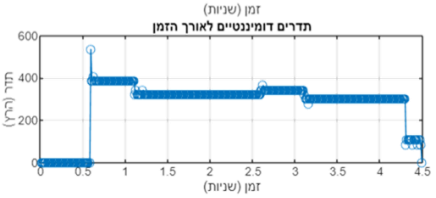


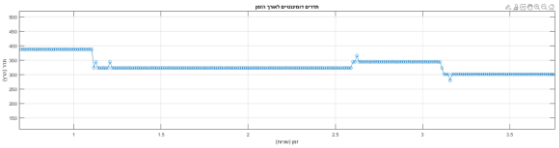
שלב 4. ביצוע סינתזה על קובץ השמע: הגרף מציג את הגל הסינתטי שנוצר, כאשר ייצרנו אותו עבור כל מקטע זמן על פי התדר הדומיננטי שנמצא

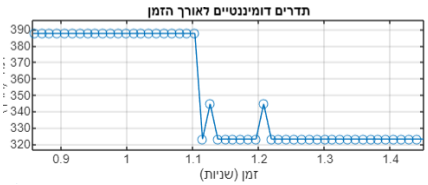




שלב 5. מציאת תדרים דומיננטיים: הגרף מציג את התדרים הדומיננטיים לאורך הזמן. כל נקודה בגרף מייצגת את התדר הדומיננטי במקטע הזמן המתאים







שלב 6. זיהוי תווים: התו נמצא באמצעות השוואה לתדרים הקרובים ביותר. אם המגניטודה גבוהה יותר מהסף שנקבע, אנו מזהים תו ומציגים את הזמן, התו והתדר.

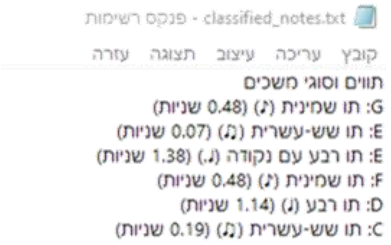
```
זיהוי תווים:
תדר: 387.5977 הרץ, זמן: 0.97524 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 0.98685 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 0.99846 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0101 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0217 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0333 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0449 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0565 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0681 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0797 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.0913 שניות, תו: G
תדר: 387.5977 הרץ, זמן: 1.1029 שניות, תו: G
תדר: 322.998 הרץ, זמן: 1.1146 שניות, תו: E
תדר: 344.5312 הרץ, זמן: 1.1262 שניות, תו: F
תדר: 322.998 הרץ, זמן: 1.1378 שניות, תו: E
תדר: 322.998 הרץ, זמן: 1.1494 שניות, תו: E
תדר: 322.998 הרץ, זמן: 1.161 שניות, תו: E
תדר: 322.998 הרץ, זמן: 1.1726 שניות, תו: E
```

שלב 7. דחיסת תווים: אחרי שמזוהים התווים, אנו מחדשים את השמות ומעבדים אותם בצורה דחוסה, אם יש שינוי בתו, נשמור את התו ואת משך הזמן שלו.

שלב 8 . זיהוי סוג התו לפי משך הזמן:

```
תדר: 387.5977 הרץ (G#) מ-0.01161 שניות
תדר: 387.5977 הרץ (G) מ-0.01161 שניות
תדר: 387.5977 הרץ (G) מ-0.02322 שניות
תדר: 387.5977 הרץ (G#) מ-0.01161 שניות
תדר: 387.5977 הרץ (G) מ-0.47605 שניות
תדר: 322.998 הרץ (E) מ-0.01161 שניות
תדר: 322.998 הרץ (E) מ-0.06966 שניות
תדר: 322.998 הרץ (E) מ-0.01161 שניות
תדר: 344.5312 הרץ (F) מ-0.02322 שניות
תדר: 322.998 הרץ (E) מ-1.38168 שניות
תדר: 322.998 הרץ (E) מ-0.01161 שניות
תדר: 322.998 הרץ (E) מ-0.47605 שניות
תדר: 322.998 הרץ (E) מ-0.01161 שניות
תדר: 322.998 הרץ (D) מ-0.03483 שניות
תדר: 322.998 הרץ (C#) מ-0.01161 שניות
תדר: 322.998 הרץ (D) מ-1.13768 שניות
תדר: 322.998 הרץ (C) מ-0.18576 שניות
תדר: 322.998 הרץ (C#) מ-0.01161 שניות
```

שלב 9. כתיבת התווים בצורה מסודרת לקובץ .txt



10. איפיון המערכת

10.1 ניתוח דרישות המערכת

סביבת פיתוח : Node.js , React ,MATLAB

חומרה:

- מחשב בעל מעבד i5 ומעלה
- זיכרון RAM של לפחות GB8
- שטח אחסון פנוי של לפחות GB2

עמדת פיתוח:

- מחשב אישי הכולל MATLAB מותקן עם Signal Processing Toolbox
- Node.js ו־ npm מותקנים
- עורך קוד Visual Studio Code
- ספריות vexflow ,jspdf svg2pdf.js

מערכת ההפעלה: Windows 10 או 11

חיבור לרשת: נצרך לצורך חיבור לחשבון מאטלאב בלבד.

תוכנות:

- MATLAB
- Node.js
- Vs code
- Google Chrome להצגת הממשק

10.2 מודול המערכת

- המערכת עוסקת בקבלת מנגינה בקובץ שמע מהמשתמש והצגת התווים לנגינתו, החל משלב קבלת המנגינה, המרתו לקובץ מונו ניתוח השיר ועד הצגת התווים בצורה גרפית יפה וקלה להבנה.
- המערכת לא עוסקת בזיהוי תווים מתוך שיר מלא הרמוניות וכלים.

10.3 איפיון פונקציונלי

- UploadForm – פונקציה שמקבלת את קובץ השמע מהמשתמש ושולחת אותו לשרת.
- analyze_audio – האלגוריתם הראשי בשרת שמנתח את אות השמע וכותב את התוצאה לקובץ json
- NotesDisplay – פונקציה שמטפלת בהצגת התווים בדפדפן ולוקחת את המידע מקובץ ה Json שיצר השרת.
- exportToPDF -פונקציה שמייצאת את התווים לקובץ PDF

10.4 ביצועים עיקריים

- העלאת קובץ שמע בפורמט MP3 או WAV
- הפעלת האלגוריתם FFT
- מציאת התדר הבולט
- מציאת התו המתאים
- סיווג התו ומשך הזמן
- כתיבת התוצאה לקובץ TXT וJSON
- הצגת התוצאה בצורה גרפית ויפה
- ייצוא התווים כקובץ PDF

10.5 אילוצים

- המערכת לא תומכת בכל הסוגים השונים של קבצי השמע למעט קבצי mp3 ו- wav .
- המערכת מזהה נכון רק ממנגינות ללא כלים נוספים ברקע.
- שירים ארוכים לוקחים יותר זמן.
- המערכת חייבת את תוכנת MATLAB עם רישיון בתוקף מותקן על המחשב.

11. תאור האריבטורה

11.1 תיאור הארכיטקטורה

המערכת בנויה מארכיטקטורה תלת-שכבתית הכוללת:

1. **שכבת ממשק משתמש**
2. **שכבת שרת**
3. **שכבת עיבוד וניתוח (MATLAB)**

תיאור השכבות:

Frontend: שימוש המשתמש

- המשתמש מעלה קובץ אודיו דרך דפדפן אינטרנט (React)
- לאחר זמן קצר מתקבלת תצוגה גרפית של תווים
- למשתמש יש אפשרות להורדת התווים כקובץ.

Backend: תקשורת עם השרת

- קובץ האודיו נשלח לשרת Node.js מקומי
- השרת מאחסן את הקובץ ומריץ תהליך MATLAB לניתוחו
- עם סיום הריצה, השרת קורא את קובץ JSON של התווים ושולח אותו חזרה ל React

MATLAB: עיבוד ב MATLAB

- קובץ האודיו מומר ל־WAV
- מתבצע חיתוך למקטעים וניתוח FFT
- זיהוי התדר הדומיננטי לכל מקטע
- מיפוי התדר לתו מוזיקלי ואוקטבה
- קביעת משך התו וסיווגו (שמינית, רבע וכו')
- שמירת הפלט לקובץ טקסט והמרה ל־JSON

[משתמש]

↓

[React] — טופס העלאת קובץ

↓

[Node.js] — טיפול בהעלאה והפעלת MATLAB

↓

[MATLAB] — עיבוד הקובץ והפקת התווים

↑

[Node.js] — שליחת JSON ל-Frontend

↑

[React] — הצגת תווים גרפית באמצעות VexFlow וייצוא הקובץ

11.2 תיאור הרכיבים בפתרון

המערכת מבוססת על ארכיטקטורה מפוצלת (client-server) הכוללת שלושה רכיבים עיקריים:

1. צד לקוח React.js – (Frontend)

- **React** היא ספריית JavaScript מבית Meta (פייסבוק), המאפשרת בניית ממשקי משתמש אינטראקטיביים בצורה מודולרית באמצעות קומפוננטות.

- היא מתאימה לפיתוח יישומים דינמיים, בהם יש צורך בריענון חלקי של המסך (Virtual DOM).
- בפתרון שלנו React ,משמש להצגת טופס העלאת הקובץ, לשליחת הבקשה לשרת ולקבלת התוצאה – תווים מוזיקליים מוצגים בגרפיקה בעזרת ספריית . **VexFlow** וייצוא קובץ הPDF בעזרת ספריית jspdf svg2pdf.js

2. צד שרת Node.js + Express – (Backend)

- **Node.js** היא סביבת זמן ריצה מבוססת JavaScript הפועלת בצד השרת, המאפשרת ביצועים גבוהים בעיבוד בקשות רשת בצורה אסינכרונית.
- **Express** היא מסגרת עבודה קלה ל Node.js המשמשת ליצירת API וניתוב בקשות.
- בפתרון זה, השרת אחראי על קבלת קובץ האודיו, הפעלת קוד MATLAB לניתוחו, והחזרת תוצאות ללקוח. הוא גם דואג לקרוא את קובץ ה JSON-שנוצר לאחר הריצה ולהחזיר אותו לממשק המשתמש.

3. רכיב ניתוח MATLAB

- **MATLAB** היא סביבת תכנות מתקדמת לניתוחים נומריים, גרפיקה ואלגוריתמים מתמטיים. היא פופולרית בתחומי עיבוד אותות, תמונות, הנדסה ומחקר מדעי.
- המערכת משתמשת ב־ MATLAB לצורך:
  - המרת האודיו לפורמט מתאים
  - חיתוך למקטעים
  - ביצוע FFT
  - זיהוי תדרים
  - מיפוי לתווים והפקת JSON
- MATLAB מופעלת על ידי פקודת מערכת מתוך Node.js והקוד המרכזי מבוצע בקובץ analyze_audio.m

11.3 ארכיטקטורת רשת

בפרויקט קיימת ארכיטקטורת רשת בסיסית מסוג Client-Server, הפועלת בסביבה מקומית (localhost) אך ניתנת להרחבה בקלות לרשת רחבה או שרת ענן. הארכיטקטורה כוללת את הרכיבים הבאים:

- **לקוח** : **(Client)** אפליקציית React הפועלת בדפדפן. המשתמש מעלה דרכה את קובץ האודיו, ומקבל את תוצאות ניתוח התווים מהשרת.
- **שרת** : **(Server)** אפליקציית Node.js עם Express שפועלת כשרת HTTP ומאזינה לבקשות HTTP מסוג POST להעלאת קובץ. השרת מפעיל תהליך MATLAB על הקובץ ושולח חזרה את הפלט.
- **רשת תקשורת** : תקשורת HTTP בין הלקוח לשרת (לרוב על פורט 3000/5000). בסביבת פיתוח localhost

- **גישה לקבצים** : המערכת מתבססת על תקשורת קבצים בין Node.js ל־ MATLAB ולא דרך socket או API פנימי. הקובץ מועבר לסקריפט MATLAB לעיבוד.

ארכיטקטורת הרשת הזו שומרת על הפרדה בין שכבות ומאפשרת ניהול תקשורת פשוט, עם אפשרות לשדרוג עתידי לתקשורת בזמן אמת או שירותי ענן.

11.4 תיאור פרוטוקולי תקשורת

HTTP

11.5 שרת-לקוח והתקשורת ביניהם

התקשורת בין הלקוח (React) לשרת (Node.js + Express) מתבצעת באמצעות פרוטוקול **HTTP** סטנדרטי, כאשר הנתונים מועברים בפורמט **JSON** בקשת ההעלאה מתבצעת דרך **HTTP POST** באמצעות טופס שמכיל את קובץ האודיו, ונשלחת לכתובת מוגדרת בשרת.

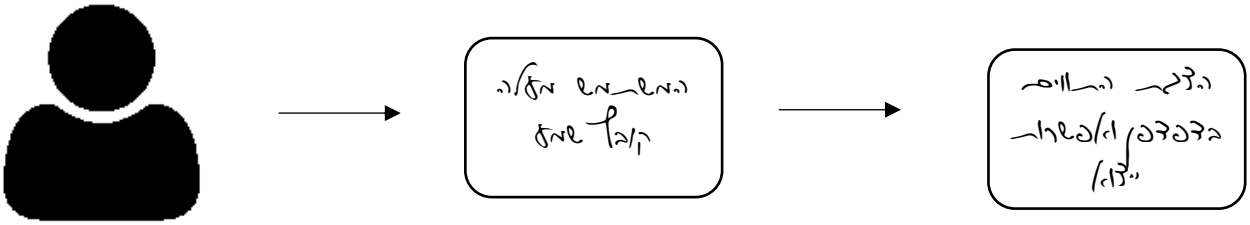
בצד השרת, ספריית **multer** משמשת לפריסת הקובץ מהבקשה והעברתו לתהליך עיבוד ב-MATLAB לאחר השלמת הניתוח, התוצאות נכתבות לקובץ JSON והשרת שולח את הנתונים חזרה ללקוח באמצעות **HTTP Response** בפורמטJSON

הטכנולוגיות המרכזיות בתקשורת כוללות:

- **React** - שולחת בקשות HTTP באמצעות fetc
- **Node.js + Express** - מאזינים לבקשות HTTP ומחזירים תגובה מתאימה.
- **פורמט** – **JSON** פורמט הנתונים לתקשורת בין לקוח לשרת.

12. ניתוח ותרשים UML / Use cases של המערכת המוצעת

12.1 תיאור ה-UC העיקריים של המערכת



12.2 הצגת use case עבור כל הפונקציות העיקריות בפרוויקט

Use Case 1 - 🗀 העלאת קובץ אודיו

- **שחקן עיקרי** : משתמש

- **מטרה** : העלאת קובץ אודיו לעיבוד
- **תיאור** : המשתמש בוחר קובץ בפורמט נתמך (MP3/WAV) דרך ממשק React ולוחץ על כפתור ההעלאה.
- **תוצאה רצויה** : הקובץ נשלח בהצלחה לשרת דרך HTTP POST
- **חריגים** : קובץ לא נתמך, קובץ ריק, בעיית רשת.

Use Case 2 - ⚙ עיבוד קובץ אודיו והפקת תווים

- **שחקן עיקרי** : שרת Node.js + MATLAB
- **מטרה** : ניתוח תדרים וזיהוי תווים
- **תיאור** : השרת מפעיל את הסקריפט analyze_audio.m שמעבד את הקובץ : המרה ל-WAV, יצירת ספקטרוגרמה, חישוב , FFT זיהוי תווים וסיוגם וכתיבת התוצאה לקובץ JSON
- **תוצאה רצויה** : יצירת קובץ JSON עם תווים מסווגים לפי שם, סוג תו (משך), ואוקטבה.
- **חריגים** : קובץ פגום, בעיה בהרצת MATLAB, קובץ ללא תוכן מוזיקלי ברור.

Use Case 3 - שליחת תוצאת ניתוח ללקוח

- **שחקן עיקרי** : שרת
- **מטרה** : החזרת תוצאות העיבוד ללקוח
- **תיאור** : לאחר סיום העיבוד, השרת קורא את קובץ התוצאה ומחזיר את תוכנו כ־ JSON לממשק React
- **תוצאה רצויה** : קבלת אובייקט JSON תקני בלקוח.
- **חריגים** : קובץ תוצאה לא נמצא, שגיאת תקשורת, שגיאת פענוח.JSON

Use Case 4 - הצגת תווים על גבי חמשה

- **שחקן עיקרי** : ממשק משתמש (React + VexFlow)
- **מטרה** : הצגת תווים בצורה גרפית קריאה
- **תיאור** : ה־ JSON מהשרת מומר לאובייקטים גרפיים על ידי VexFlow ומוצג בחלון המשתמש עם חמשה, מפתח סול, וסמלי תווים מתאימים.
- **תוצאה רצויה** : תצוגה מדויקת של כל תו במיקום ובמשך הנכון.
- **חריגים** : תווים חסרים, תו לא מזוהה, כשל בהצגה גרפית.

Use Case 5 - ייצא התווים שזוהו כקובץ PDF

- **שחקן עיקרי** : ממשק משתמש (React + jspdf svg2pdf.js)
- **מטרה** : לייצא את התווים שזוהו מקובץ השמע כקובץ PDF הניתן להדפסה או לשיתוף
- **תיאור** : המשתמש לוחץ על כפתור הייצוא והמערכת מורידה לו למחשב את התווים שזוהו בקובץ PDF בעזרת ספריית jspdf svg2pdf.js.
- **תוצאה רצויה** : קובץ PDF עם התווים שזוהו בצורה גרפית בתיקיית ההורדות של המשתמש
- **חריגים** : לא זוהו תווים, תקלה בהפקת PDF (יכול להגרם אם הייתה תקלה בהצגה הגרפית בדפדפן).

12.3 מבני נתונים שבשימוש בפרויקט

MATLAB:

MATLAB היא שפה פחות מוכרת לסטודנטים וקצת שונה משפות מוכרות אחרות כמו C# או PYTON. אך עם זאת מבני הנתונים שלה דומים לשפות אלו. בפרויקט שלי מבני הנתונים העיקריים היו:

- **מערכים Arrays**
  מערכים הם מבני נתונים המקבילים לאורכים קבועים של סוגי נתונים. המערך מקבל מבנה קבוע, כלומר כל האלמנטים במערך הם מאותו סוג.
  דוגמא מהפרויקט - מערך של תדרים: note_frequencies = [261.63, 277.18, 293.66, 311.13, 329.63];
  מערך זה מאחסן את התדרים של התווים, בו כל תדר הוא ערך נומרי.

- **תאים Call Arrays**
  תאים מאפשרים לאחסן נתונים מסוגים שונים (כגון מיתרים, מספרים וכו') בתוך אלמנטים שונים.
  בכדי לשמור תווים עם סוגים שונים של נתונים השתמשתי בתאים: detected_notes = cell(size(dominantFrequencies));
  כאן כל תא בתוך detected_notes מכיל תו (כמו E, D, C-) או ערך ריק. גישה לתא נעשית באמצעות {} ולא ().

- **מפות Containers.Map**
  השתמשתי במפה כדי לאחסן קשר בין תדרים לתווים. מפה היא מבנה נתונים שבו כל מפתח מקושר לערך (בדומה ל- HashMap בשפות אחרות).לדוגמא-
  המפה note_dict מקשרת כל תדר לשם התו המתאים: note_dict = containers.Map(note_frequencies, notes_names);

- **רשימות Lists**
  השתמשתי ברשימות כשתיארתי תווים שנמצאו ברצפים שונים או באורך משתנה כמו

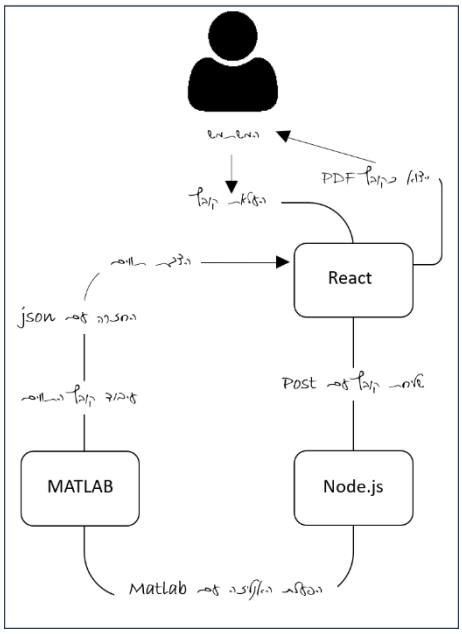
ב repeated_notes: repeated_notes = {};
רשימה זו אינה מוגבלת בגודל ומאפשרת הוספה של תווים בזמן הריצה.

React + Node.js:

בצד לקוח ובשרת השתמשתי במבני הנתונים המוכרים כמו

- fs, JSON objects, Array לקריאה, פירוק ואכסון הנתונים מה JSON
- Buffer לקליטת קבצי אודיו שהועלו על ידי המשתמש
- VexFlow Note objects, Objects לצורך הצגת התווים בדפדפן.

12.4 הקשרים בין היחידות השונות



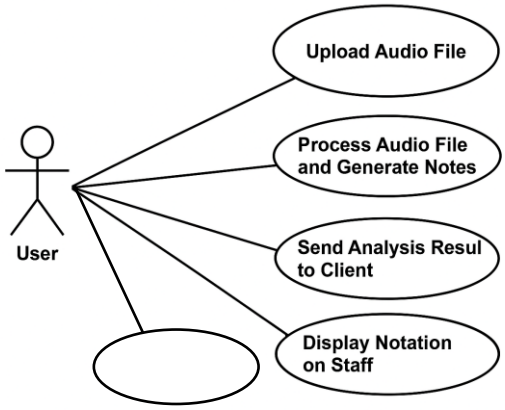
12.5 עץ מודולים

📦 מערכת זיהוי תווים
├── 🎧 לקוח (React)
│ ├── 📁 העלאת קובץ אודיו
│ ├── 📁 שליחת בקשה לשרת (API Service)
│ ├── 📁 קבלת פלט JSON
│ └── 📁 הצגת תווים גרפית (VexFlow)
│ ___ 📁 ייצוא התווים כקובץ PDF

├── 🌐 שרת (Node.js + Express)
│ ├── 📁 קבלת קובץ מהלקוח (multer)
│ ├── 📁 הפעלת MATLAB דרך child_process
│ ├── 📁 קריאת פלט ממאטלאב (classified_notes.txt)
│ └── 📁 המרת טקסט ל-JSON ושליחה חזרה ללקוח

└── מנתח (MATLAB)
├── המרת קובץ אודיו (mp3 → wav)
├── חישוב FFT וזיהוי תדרים דומיננטיים
├── מיפוי תדרים לתווים
├── סיווג תווים לפי משך וזיהוי אוקטבה
└── כתיבת תוצאה לקובץ classified_notes.json

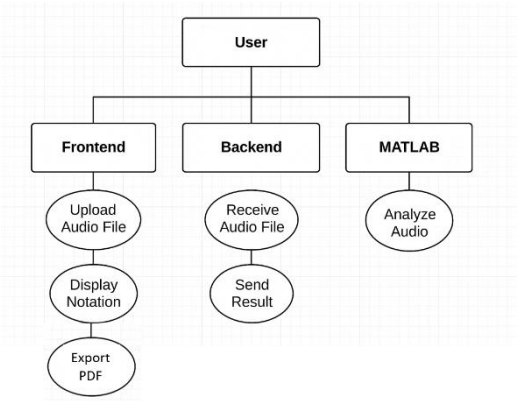
12.6 Use case Diagram



12.7 רשימת Use cases

- העלאת קובץ על ידי המשתמש
- עיבוד קובץ השמע וזיהוי התווים
- שליחת התוצאה בחזרה ללקוח
- הצגת התווים בדפדפן
- ייצוא התווים כקובץ PDF

12.8 UML



13. רכיבי ממשק

דפדפן (Firefox, edge, chrome)

14. תיכון המערכת

14.1 ארכיטקטורת המערכת

WebApi ארכיטקטורת Client-Server

14.2 תיכון מפורט

JavaScript , React – **Client**

Node.js, MATLAB **- Server**

14.3 חלופות המערכת

Angular, Python, C++

15. תיאור התוכנה

15.1 סביבות עבודה

- R2024a - MATLAB
- 1.99.3 - Visual Studio Code

15.2 שפות תכנות

- v22.12.0 - Node.js
- 10.9.0 - NM
- ^19.1.0 - React
- R2024a - MATLAB

16. תיאור המסכים

מסך האפליקציה – מסך ובו כפתור לבחירת שיר וכפתור נוסף לשליחת השיר לזיהוי. לאחר שהתווים מזוהים הם מוצגים מתחת בצורה ויזואלית מקצועית ויש אפשרות של ייצוא התווים בקובץ PDF.

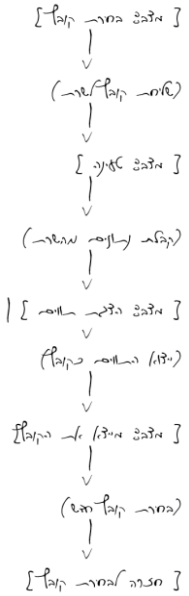
17. תרשים מסכים המתאר את היררכיית המסכים והמעברים ביניהם

בפרויקט קיים ממשק משתמש מבוסס מסך יחיד בו כל השלבים מתבצעים על פני **אותו מסך React** המעברים בין המצבים נעשים באופן דינמי בהתאם לפעולות המשתמש ומצב המערכת.

תיאור מצבי המסך:

1. **מצב התחלתי – בחירת קובץ:**
   - המשתמש מתבקש לבחור קובץ שמע לניתוח.
   - מוצג כפתור "בחר קובץ" וכפתור "שלח."
2. **מצב טעינה – ניתוח מתבצע:**
   - לאחר שליחת הקובץ לשרת, מוצגת אינדיקציה של טעינה.
   - לא ניתן לבצע פעולות נוספות בזמן זה.
3. **מצב תוצאה – הצגת תווים:**
   - לאחר קבלת התוצאה מהשרת, מוצגים התווים שנמצאו באמצעות רכיב גרפי - VexFlow.
   - יש אפשרות לייצא את התווים בקובץ PDF
   - ניתן לבחור קובץ חדש ולהתחיל תהליך נוסף.

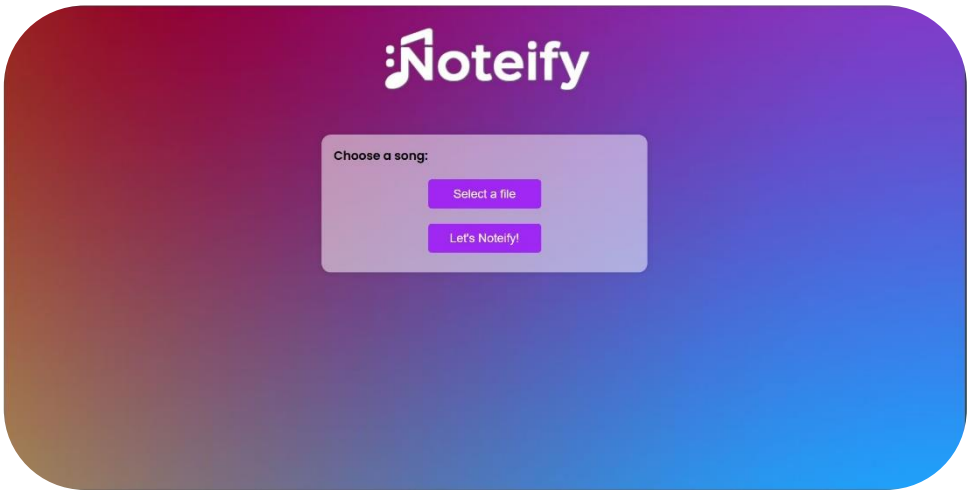
תרשים זרימת המסך:

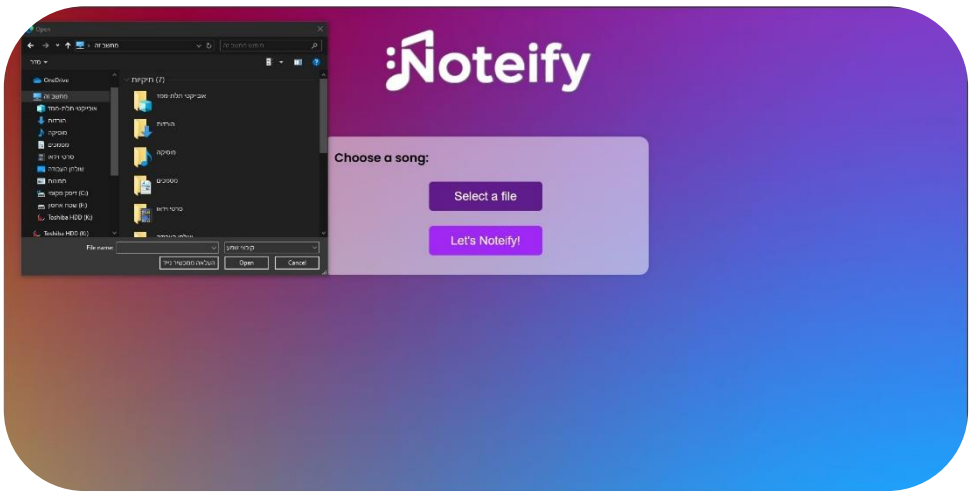


18. פרוט המסכים

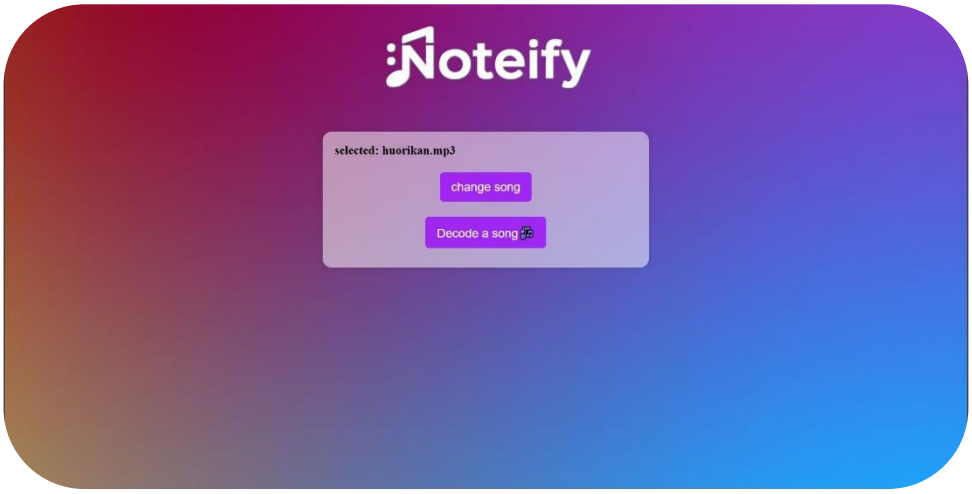
האפליקציה נוחה מאוד למשתמש וכוללת מסך אחד שבו נמצא כל המידע.

שלב 1: בחירת שיר. על המשתמש ללחוץ על כפתור "select a file". ולבחור שיר מתוך המחשב.

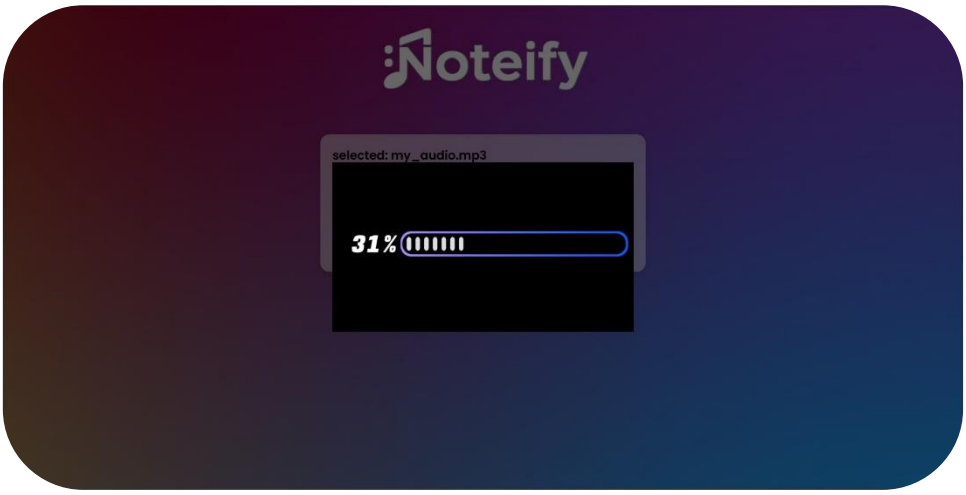




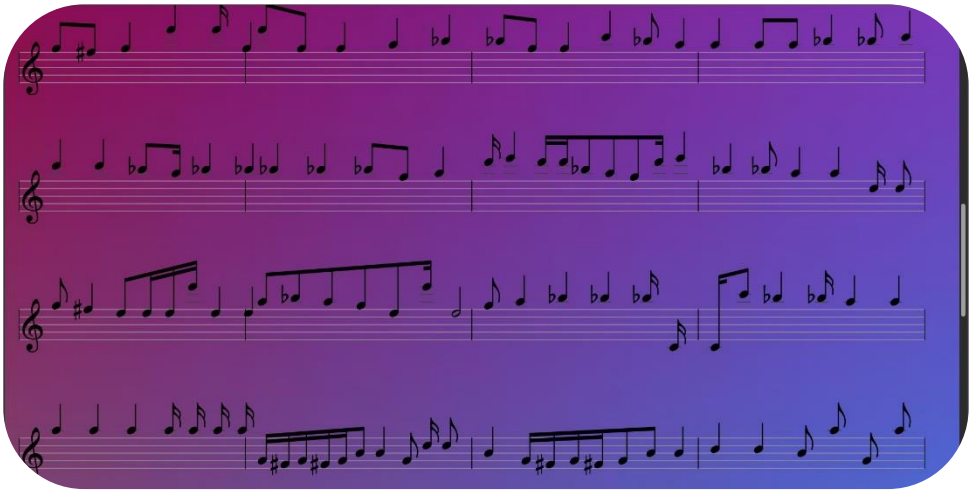
שלב 2: שם השיר יופיע למעלה וכפתור "select a file" יהפוך לכפתור "change song" בשביל לאפשר למשתמש, מכל סיבה שהיא, להחליף שיר.

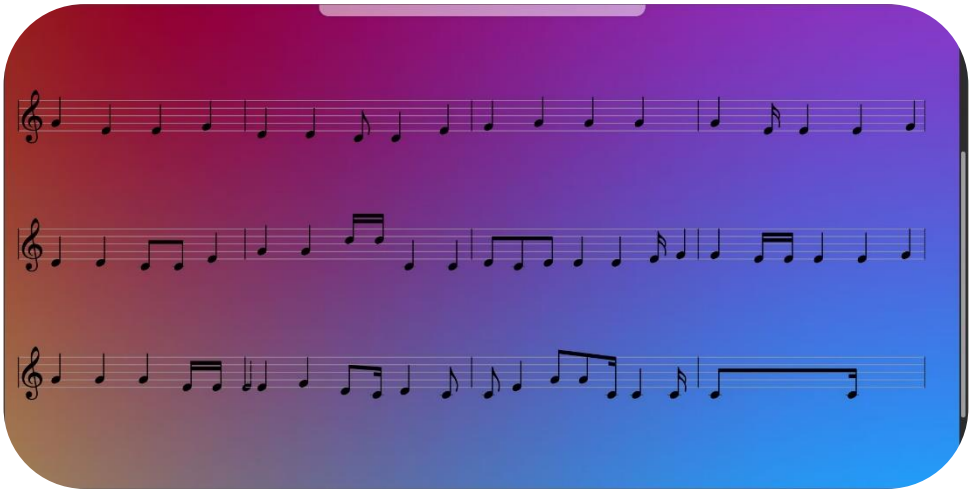


שלב 3: לאחר בחירת השיר הרצוי, על המשתמש ללחוץ על כפתור "Lat's Noteify". המערכת תתחיל לנתח את השיר, ועל המסך יופיע טיימר התקדמות, כשברקע מושמע השיר הנבחר.

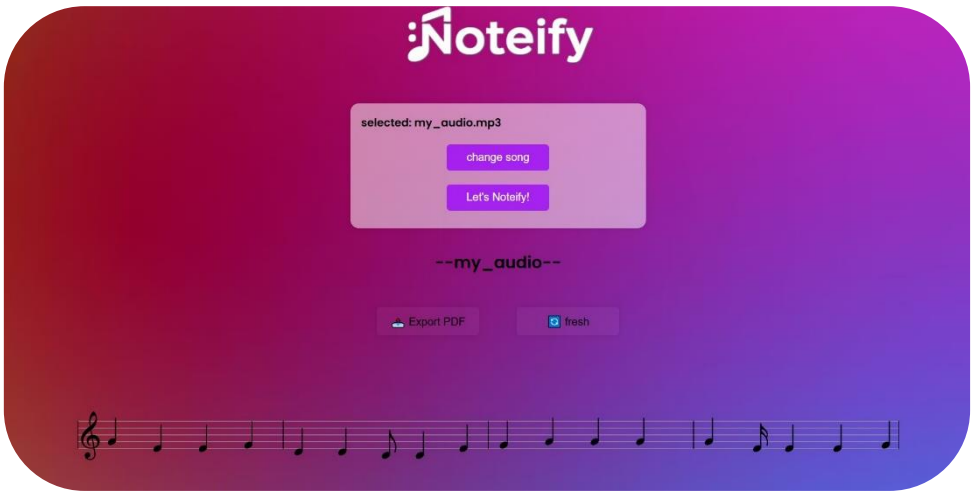


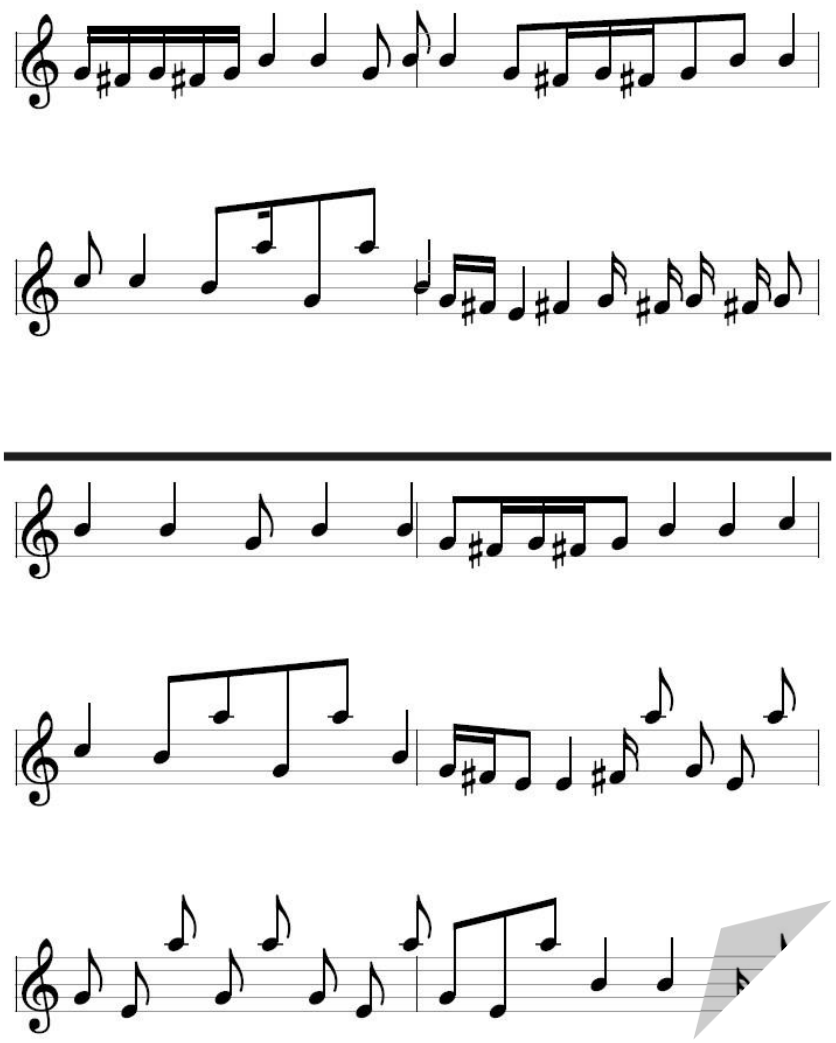
שלב 4: המערכת תציג למשתמש את שם השיר ואת התווים בצורה גרפית ומקצועית. (בדוגמא 2 שירים שונים, אחד מורכב יותר והשני פשוט יותר)





שלב 5 : אפשרות המשתמש לייצא את התווים בקובץ PDF לצורך הדפסה או שיתוף, הקובץ ירד וימצא בתיקיית ההורדות של המכשיר.





19. קוד התוכנית

Server.js

matlab-audio-server > JS server.js > ...

```
const fs = require('fs');
const path = require('path');
const express = require('express');
const multer = require('multer');
const cors = require('cors');
const { exec } = require('child_process');

const app = express();

app.use(cors());

// הגדרת multer לאחסון הקבצים בתיקיית uploads
const upload = multer({ dest: 'uploads/' });

app.use(express.json());

app.post('/upload', upload.single('audio'), (req, res) => {

  console.log('Received file:', req.file);  // מוודא שהקובץ התקבל
  const file = req.file;
  if (!file) {
    return res.status(400).send('No file uploaded');
  }
  if (!req.file) {
    return res.status(400).json({ error: 'No file uploaded' });
  }

  console.log(req.body); // הדפסת הנתונים המתקבלים

  const audioFilePath = req.file.path;
  console.log('📂 Uploaded file:', req.file);

  const matlabCommand = `matlab -batch "analyze_audio('${audioFilePath}')"`; // ביצוע פקודת MATLAB

  exec(matlabCommand, (err, stdout, stderr) => {
    if (err) {
      console.error('❌ MATLAB error:', stderr);
      return res.status(500).json({ error: 'MATLAB processing failed' });
    }

    const resultPath = path.join(__dirname, 'uploads', 'classified_notes.json');

    if (!fs.existsSync(resultPath)) {
      return res.status(404).json({ error: 'Result file not found' });
    }

    fs.readFile(resultPath, 'utf8', (err, data) => {
      if (err) {
        return res.status(500).json({ error: 'Failed to read result file' });
      }

      return res.json(JSON.parse(data));
    });
  });
});

app.listen(3001, () => {
  console.log('Server running on port 3001');
});
```

analyze_audio.m – MATLAB

matlab-audio-server > analyze_audio.m

```
function analyze_audio(filePath)

    %-----------------------------------
    %שלב 1: קריאת קובץ האודיו
    %-----------------------------------
    fprintf("🔍 Analyzing file: %s\n", filePath);
    [audioData, sampleRate] = audioread(filePath);                          % קריאת הקובץ לאודיו
    if size(audioData, 2) > 1                                               % אם האודיו הוא סטריאו, נבחר את הערוץ הראשון
        audioData = audioData(:, 1);
    end

    %-----------------------------------------
    % שלב 2: המרת קובץ אם אינו בפורמט WAV
    %-----------------------------------------
    if strcmpi(filePath(end-2:end), 'mp3')
        [~, fileName, ~] = fileparts(filePath);                             % קבלת שם הקובץ בלי סיומת
        outputfilePath = fullfile(fileparts(filePath), [fileName, '_converted.wav']);  % שמירה באותו תיקיה עם שם חדש
        audiowrite(outputfilePath, audioData, sampleRate);
        filePath = outputfilePath;                                          % המומר WAV-עדכון נתיב הקובץ ל
    end

    %-----------------------------------------
    % שלב 3: קריאת הספקטרוגרמה של האודיו
    %-----------------------------------------
    window = 1024;                                                          % גודל החלון
    overlap = 512;                                                          % חפיפות
    nfft = 2048;                                                            % מספר חישובי ה-FFT
    [S, F, T] = spectrogram(audioData, window, overlap, nfft, sampleRate);  % T=זמן, F=תדרים

    %-----------------------------------
    %שלב 4: חישוב התדרים הדומיננטיים
    %-----------------------------------
    [~, maxIndex] = max(abs(S));                                            % מציאת המגניטודה המקסימלית
    dominantFrequencies = F(maxIndex);                                      % התדרים הדומיננטיים
    synthesizedAudio = zeros(size(audioData));                              % אתחול האות
    dt = 1 / sampleRate;                                                    % מרווח דגימה

    for k = 1:length(T)-1                                                   % יצירת גל סינוסי בתדר הדומיננטי
        t = T(k):dt:T(k+1)-dt;                                              % זמן למקטע
        freq = dominantFrequencies(k);
        synthesizedAudio(round(T(k)*sampleRate):round(T(k+1)*sampleRate)-1) = ...
            sin(2 * pi * freq * t);
    end

                                                                            % יצירת מילון מתאים בין תדרים לתווים
    note_frequencies = [261.63, 277.18, 293.66, 311.13, 329.63, 349.23, 369.99, 392.00, 415.30, 440.00, ...
                        466.16, 493.88, 523.25, 554.37, 587.33, 622.25, 659.25, 698.46, 739.99, ...
                        783.99, 830.61, 880.00, 932.33, 987.77, 1046.50, 1108.73, 1174.66, ...
                        1244.51, 1318.51];

    notes_names = {'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B', 'C5', 'C#5', 'D5', ...
                   'D#5', 'E5', 'F5', 'F#5', 'G5', 'G#5', 'A5', 'A#5', 'B5', 'C6', 'C#6', 'D6', ...
                   'D#6', 'E6'};

    note_dict = containers.Map(note_frequencies, notes_names);              % יצירת מילון של תדרים לתווים
    magnitude_threshold = 0.01;                                             % רף מינימלי לעוצמת האות

    %-------------------------------------------------
    % שלב 5:זיהוי תווים בהתבסס על התדרים והעוצמות
    %-------------------------------------------------
    detected_notes = cell(size(dominantFrequencies));
    for i = 1:length(dominantFrequencies)
        if max(abs(S(:, i))) > magnitude_threshold                          % אם המגניטודה גבוהה מספיק
            [~, index] = min(abs(note_frequencies - dominantFrequencies(i)));  % מצא את התדר הקרוב ביותר
            detected_notes{i} = note_dict(note_frequencies(index));         % שמור את התו
        end
    end

    %-----------------------------------------------
    %שלב 6: דחיסת תווים ומעקב אחר משך הזמן
    %-----------------------------------------------
    beatDuration = 0.5;                                                     % נניח קצב של 120 BPM (60/120 = 0.5 שניות)
    compressed_notes = {};
    durations_windows = [];

    silence_threshold = 0.05;                                               % ערך עוצמה מתחתיו נחשב שתיקה
    if ~isempty(detected_notes)
        current_note = detected_notes{1};
        current_duration = 1;

        for i = 2:length(detected_notes)
            current_magnitude = max(abs(S(:, i)));                          % העוצמה בחלון הנוכחי
            if strcmp(detected_notes{i}, current_note) && current_magnitude > silence_threshold
                current_duration = current_duration + 1;                    % אם אותו תו ואין שתיקה
            else
                compressed_notes{end+1} = current_note;                     % אחרת: סוגרים תו ומתחילים חדש
                durations_windows(end+1) = current_duration;
                current_note = detected_notes{i};
                current_duration = 1;
            end
        end

        compressed_notes{end+1} = current_note;                             % הוספת התו האחרון
        durations_windows(end+1) = current_duration;
    end

    %-----------------------------------------------
    % שלב 7: איחוד תווים חוזרים ומפוצלים
    %-----------------------------------------------
```

```
    min_duration_seconds = 0.1;                                                 % סף זמן מינימלי לתו
    i = 1;
    while i < length(compressed_notes)
        duration_seconds = durations_windows(i) * (window - overlap) / sampleRate;          % חישוב הזמן של התו בשניות
        if duration_seconds < min_duration_seconds                                          % אם הזמן קטן מהסף
            if i < length(compressed_notes) && strcmp(compressed_notes{i}, compressed_notes{i+1})
                durations_windows(i+1) = durations_windows(i+1) + durations_windows(i);
                compressed_notes(i) = [];
                durations_windows(i) = [];
                continue;
            end
        end
        i = i + 1;
    end
    %--------------------------------------------------
    % שלב 8: סיווג התווים לפי המשך שלהם
    %--------------------------------------------------
    final_classified_notes = {};
    final_durations_seconds = [];

    for i = 1:length(compressed_notes)
        note = compressed_notes{i};
        duration_seconds = durations_windows(i) * (window - overlap) / sampleRate;              % חישוב זמן בשניות

        if duration_seconds > 0.09
            halfNoteDurationThreshold = beatDuration * 1.8;                                     % סף לזיהוי תו חצי
            quarterNoteDuration = beatDuration * 0.9;                                           % משך זמן משוער של רבע

            if duration_seconds >= halfNoteDurationThreshold
                final_classified_notes{end+1} = note;
                final_durations_seconds(end+1) = quarterNoteDuration;
                final_classified_notes{end+1} = note;
                final_durations_seconds(end+1) = quarterNoteDuration;
            else
                final_classified_notes{end+1} = note;
                final_durations_seconds(end+1) = duration_seconds;
            end
            %disp(['תו: ', note, ' (', num2str(duration_seconds), ' שניות)']);
        end
    end

    %-------------------------------------------------------
    % שלב 9: כתיבת התוצאות לקובץ classified_notes.txt
    %-------------------------------------------------------
    [fileDir, ~, ~] = fileparts(filePath);
    outputFileClassified = fullfile(fileDir, 'classified_notes.txt');
    fileID = fopen(outputFileClassified, 'w');
    if fileID == -1
        disp('שגיאה: לא ניתן לפתוח את הקובץ לכתיבה.');
        return;
    end

    fprintf(fileID, '(בשניות) תווים ומשכי זמן\n');

    for i = 1:length(final_classified_notes)
        duration_seconds = final_durations_seconds(i);
        fprintf(fileID, 'תו: %s, 2.%f :משך שניות\n', final_classified_notes{i}, duration_seconds);
    end

    fclose(fileID);
    disp(['סיווג התווים נכתב לקובץ: ', outputFileClassified]);

    %---------------------------------------------
    % שלב 10: יצירת מבנה JSON עם התוצאות
    %---------------------------------------------
    output.notes = struct('note', final_classified_notes', 'duration', num2cell(final_durations_seconds'));

    [~, fileName, ~] = fileparts(filePath);                                                 % יצירת נתיב לקובץ JSON
    jsonOutputPath = fullfile(pwd, 'uploads', 'classified_notes.json');
    if ~exist('uploads', 'dir')
        mkdir('uploads');
        disp('תיקייה uploads נוצרה.');
    end

    fid = fopen(jsonOutputPath, 'w');                                                       % כתיבת התוצאה לקובץ JSON
    if fid == -1
        disp('שגיאה: לא ניתן לפתוח את הקובץ _classified_notes.json לכתיבה.');
        return;
    end

    fwrite(fid, jsonencode(output));
    fclose(fid);
    disp(['הנתונים נכתבו לקובץ: ', jsonOutputPath]);
end
```

APP.js-קומפוננטת האב צד לקוח

```
promusic > src > JS App.js > ...
import React, { useState, useEffect } from 'react';
import UploadForm from './components/UploadForm';
import NotesDisplay from './components/NotesDisplay';
import './App.css';


function App() {
  const [analysisResult, setAnalysisResult] = useState(null);
  const [vexFlowLoaded, setVexFlowLoaded] = useState(false);


  useEffect(() => {
    // Check if VexFlow is already loaded
    if (window.Vex) {
      setVexFlowLoaded(true);
    } else {
      const script = document.createElement('script');
      script.src = 'https://unpkg.com/vexflow@4.2.1/build/cjs/vexflow.js'; // Use a specific version
      script.async = true;
      script.onload = () => {
        console.log("VexFlow loaded successfully");
        setVexFlowLoaded(true);
      };
      script.onerror = () => {
        console.error("Failed to load VexFlow");
        setVexFlowLoaded(false);
      };
      document.body.appendChild(script);

      return () => {
        if (document.body.contains(script)) {
          document.body.removeChild(script);
        }
      };
    }
  }, []);

  const parseNotes = (data) => {
    if (typeof data === 'string') {
      const lines = data.split('\n');
      return lines.map(line => {
        const [pitch, duration] = line.split(':');
        return {
          note: pitch.trim(),
          duration: duration.trim().replace(/[^\w\s]/g, '')
        };
      });
    }

    if (Array.isArray(data?.notes)) {
      return data.notes.map(note => ({
        note: note.note,
        duration: note.duration
      }));
    } else if (data && typeof data === 'object' && data !== null) {
      return Object.values(data).map(item => ({
        note: item.note, // adjust property names if needed
        duration: item.duration
      }));
    }

    return [];
  };

  return (

    <div className="App">

      <header>
        <img src="/logo.png" alt="לוגו האתר" style={{ width: '300px', padding: '20px' }} />
      </header>

      <UploadForm setAnalysisResult={setAnalysisResult} />

      {analysisResult?.filename && (
        <h2>--{analysisResult.filename.replace(/\.[^/.]+$/, "")}--</h2>
      )}

      {analysisResult && vexFlowLoaded && (
        <div className="result">
          <NotesDisplay notes={parseNotes(analysisResult) || []} />

        </div>
      )}
      {analysisResult && !vexFlowLoaded && (
        <p>Loading VexFlow...</p>
      )}
    </div>
  );
}
export default App;
```

UploadForm.js

promusic > src > components > JS UploadForm.js > ...

```javascript
import React, { useState, useRef } from "react";
import "./UploadForm.css";

function UploadForm({ setAnalysisResult }) {
  const [file, setFile] = useState(null);
  const [loading, setLoading] = useState(false);
  const fileInputRef = useRef(null);
  const [audioFile, setAudioFile] = useState(null);
  const audioRef = useRef(null);



  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!file) return alert("Please select a song file.");

    const formData = new FormData();
    formData.append("audio", file);

    const audioURL = URL.createObjectURL(file);
    audioRef.current.src = audioURL;


    try {
      await audioRef.current.play();

      setLoading(true);
      try {
        const response = await fetch("http://localhost:3001/upload", {
          method: "POST",
          body: formData,
        });

        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }


        audioRef.current.pause();
        audioRef.current.currentTime = 0;
        await audioRef.current.play();

        const result = await response.json();
        setAnalysisResult({ ...result, filename: file.name });
      } catch (error) {
        alert("Error sending file to server.");
        console.error("Error:", error);
        setAnalysisResult({ error: error.message });
      } finally {
        setLoading(false);
      }
    } catch (err) {
      alert("Failed to play audio.");
      console.error(err);
    }
  };


  const handleButtonClick = () => {
    fileInputRef.current.click(); // הפעל את חלון בחירת הקובץ
  };

  return (
    <form className="upload-form" onSubmit={handleSubmit}>
      <label htmlFor="file-upload-real" className="file-upload-label">
        {file ? `selected: ${file.name}` : "Choose a song:"}
      </label>
      <input
        id="file-upload-real"
        type="file"
        accept="audio/*"
        onChange={(e) => setFile(e.target.files[0])}
        ref={fileInputRef}
        style={{ display: 'none' }} // הסתר את כפתור ברירת המחדל
      />
      <button type="button" onClick={handleButtonClick} className="custom-file-button">
        {file ? "change song" : "Select a file"}
      </button>
      <button type="submit" disabled={loading}>
        {loading ? "Noteify..." : "Let's Noteify!"}
      </button>
      <audio ref={audioRef} hidden />
      {loading && (
        <div className="overlay">
          <img src="/note-loading.gif" alt="Loading animation" className="loader-gif" />
        </div>
      )}
    </form>
  );
}

export default UploadForm;
```

NotesDisplay

promusic > src > components > JS NotesDisplay.js > ...

```
import React, { useState, useEffect, useRef } from 'react';
import { Renderer, Stave, StaveNote, Voice, Formatter, Vex } from "vexflow";
import jsPDF from "jspdf";
import { svg2pdf } from "svg2pdf.js";

const NotesDisplay = ({ notes }) => {
  const [notesState, setNotes] = useState(notes);
  const vexRef = useRef(null);
  const staveHeight = 150;
  const beatsPerMeasure = 4;
  const beatValue = 4;
  const ticksPerBeat = 1024 / beatValue;
  const requiredTicksPerMeasure = ticksPerBeat * beatsPerMeasure;
  const quarterDuration = 0.48;

  const handleRefresh = () => {
    window.location.reload();
  };

  const getDurationType = (durationSeconds) => {
    if (durationSeconds >= 3.5 * quarterDuration) return "w";
    if (durationSeconds >= 1.8 * quarterDuration) return "h";
    if (durationSeconds >= 0.85 * quarterDuration) return "q";
    if (durationSeconds >= 0.4 * quarterDuration) return "8";
    if (durationSeconds >= 0.2 * quarterDuration) return "16";
    return "q";


  };

  const processedNotes = notesState.map(note => {
    if (note && note.note && note.duration) {
      const durationSeconds = parseFloat(note.duration);
      const noteName = note.note;
      let octave = 4;
      let pitch = noteName.toUpperCase();
      const octaveMatch = noteName.match(/[A-Ga-g][#b]?\d/);

      if (octaveMatch) {
        pitch = octaveMatch[0].slice(0, -1).toLowerCase();
        octave = parseInt(octaveMatch[0].slice(-1));
      }

      const formattedNote = `${pitch}/${octave}`;
      const durationType = getDurationType(durationSeconds);

      // כאן נבדוק אם האוקטבה היא 5, אם כן נשמה את הרגל הפוך
      const stemDirection = octave < 5 ? Vex.Flow.Stem.UP : Vex.Flow.Stem.DOWN;

      return {
        keys: [formattedNote],
        duration: durationType,
        originalTicks: durationType === 'w' ? ticksPerBeat * 4 :
          durationType === 'h' ? ticksPerBeat * 2 :
            durationType === 'q' ? ticksPerBeat :
              durationType === '8' ? ticksPerBeat / 2 :
                durationType === '16' ? ticksPerBeat / 4 : ticksPerBeat,
        stemDirection: stemDirection // הוספתי את כיוון הרגל
      };
    }
    return null;
  }).filter(Boolean);


  useEffect(() => {
    if (!processedNotes || processedNotes.length === 0 || !vexRef.current) return;

    const vexContainer = vexRef.current;
    vexContainer.innerHTML = '';

    try {
      const renderer = new Renderer(vexContainer, Renderer.Backends.SVG);
      const context = renderer.getContext();
      const width = vexContainer.clientWidth - 20;
      let currentX = 10;
      let currentY = 60;
      let measureNotes = [];
      let currentMeasureTicks = 0;
      let stave;
      let firstStaveInLine = true;
      const staveWidth = 300;
      let totalHeight = 0;

      let beamGroup = [];
      let allBeams = [];

      processedNotes.forEach((note, index) => {
        const staveNote = new StaveNote({
          keys: [note.keys[0].toLowerCase()],
          duration: note.duration
        });

        if (note.keys[0].includes("#")) {
          staveNote.addModifier(new Vex.Flow.Accidental("#"), 0);
        } else if (note.keys[0].includes("b")) {
          staveNote.addModifier(new Vex.Flow.Accidental("b"), 0);
        }
```

```
98
99          if (note.stemDirection) {
100           staveNote.setStemDirection(note.stemDirection);  // הוספתי את כיוון הרגל
101         }
102
103         if (note.duration === "16" || note.duration === "8") {
104           beamGroup.push({ staveNote, note });
105         } else {
106           if (beamGroup.length >= 2) {
107             applyBeamStemDirection(beamGroup);
108             allBeams.push(new Vex.Flow.Beam(beamGroup.map(n => n.staveNote)));
109           }
110           beamGroup = [];
111         }
112
113         if (index === processedNotes.length - 1 && beamGroup.length >= 2) {
114           applyBeamStemDirection(beamGroup);
115           allBeams.push(new Vex.Flow.Beam(beamGroup.map(n => n.staveNote)));
116           beamGroup = [];
117         }
118
119         measureNotes.push(staveNote);
120         currentMeasureTicks += note.originalTicks;
121
122         if (currentMeasureTicks >= requiredTicksPerMeasure || index === processedNotes.length - 1) {
123           stave = new Stave(currentX, currentY, staveWidth);
124           if (firstStaveInLine) {
125             stave.addClef('treble').setContext(context).draw();
126             firstStaveInLine = false;
127           } else {
128             stave.setContext(context).draw();
129           }
130           stave.setEndBarType(Vex.Flow.Barline.TYPE_SINGLE);
131
132           const voice = new Voice({ num_beats: beatsPerMeasure, beat_value: beatValue });
133           voice.setStrict(false);
134           voice.addTickables(measureNotes);
135           new Formatter().joinVoices([voice]).format([voice], stave.width - 20);
136           voice.draw(context, stave);
137
138           allBeams.forEach(beam => beam.setContext(context).draw());
139
140           currentX += staveWidth;
141           measureNotes = [];
142           currentMeasureTicks = 0;
143           allBeams = [];
144           beamGroup = [];
145
146           if (currentX + staveWidth + 10 > width) {
147             currentY += staveHeight + 20;
148             currentX = 10;
149             firstStaveInLine = true;
150           }
151         }
152       });
153
154       totalHeight = currentY + staveHeight + 40;
155       renderer.resize(width + 20, totalHeight);
156     } catch (error) {
157       console.error('Error initializing VexFlow', error);
158       vexContainer.innerHTML = 'Error initializing VexFlow. Please check your setup.';
159     }
160   }, [processedNotes]);
161
162   const applyBeamStemDirection = (group) => {
163     const octaveNumbers = group.map(n => parseInt(n.note.keys[0].match(/\/(\d+)/)[1]));
164     const stemDownCount = octaveNumbers.filter(oct => oct >= 5).length;
165     const stemDirection = stemDownCount > group.length / 2 ? Vex.Flow.Stem.DOWN : Vex.Flow.Stem.UP;
166     group.forEach(({ staveNote }) => staveNote.setStemDirection(stemDirection));
167   };
168
169   const exportToPDF = async () => {
170     const svgElement = document.querySelector("svg");
171     if (!svgElement) return alert("ERROR! can't find note.");
172
173     const svgRect = svgElement.getBoundingClientRect();
174     const svgWidth = svgRect.width;
175     const svgHeight = svgRect.height;
176
177     const pdf = new jsPDF({ orientation: "portrait", unit: "pt", format: "a1" });
178     const pageWidth = pdf.internal.pageSize.getWidth();
179     const pageHeight = pdf.internal.pageSize.getHeight();
180     const margin = 0;
181
182     const availablePageWidth = pageWidth - 2 * margin;
183     const availablePageHeight = pageHeight - 2 * margin;
184     const scale = availablePageWidth / svgWidth;
185     const originalPageHeightEquivalent = availablePageHeight / scale;
186
187     let yOffset = 0;
188
189     while (yOffset < svgHeight) {
190       const clonedSvg = svgElement.cloneNode(true);
191       const viewBoxHeight = Math.min(originalPageHeightEquivalent, svgHeight - yOffset);
192       clonedSvg.setAttribute("viewBox", `0 ${yOffset} ${svgWidth} ${viewBoxHeight}`);
193       clonedSvg.setAttribute("width", svgWidth);
194       clonedSvg.setAttribute("height", viewBoxHeight);
195
196       await svg2pdf(clonedSvg, pdf, {
197         xOffset: margin,
198         yOffset: margin,
199         scale: scale,
200       });
```

```
      yOffset += originalPageHeightEquivalent;
      if (yOffset < svgHeight) pdf.addPage();
    }

    pdf.save("Noteify_export_PDF.pdf");
  };

  return (
    <div className="p-4" style={{ width: '100%', overflowX: 'auto' }}>
      {processedNotes.length > 0 && (
        <div className="flex items-center gap-4 mb-4">
          <button
            id="PDF_export"
            onClick={exportToPDF}
            className="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700 transition"
          >
            📥 Export PDF
          </button>
          <button
            id="PDF_export"
            onClick={handleRefresh}
            className="bg-gray-500 text-white px-4 py-2 rounded hover:bg-gray-600 transition"
          >
            🔄 fresh
          </button>
        </div>
      )}
      {processedNotes.length > 0 ? (
        <div ref={vexRef} className="mb-6"></div>
      ) : (
        <p>העלה שיר כדי לראות את התווים כאן.</p>
      )}
    </div>
  );
};


export default NotesDisplay;
```

20. תיאור מסד הנתונים

חלק הפרויקט, מידע אודות התווים המוזיקליים מזוהה באמצעות קוד MATLAB המנתח קובץ שמע. תוצאות הניתוח נשמרות בקובץ טקסט בפורמט אחיד, אשר מומר לאחר מכן לקובץ JSON קובץ זה משמש כבסיס להצגת התווים בצד הלקוח, באמצעות ממשק גרפי מבוסס React וספריית VexFlow

הקובץ בנוי כמערך notes ובו לכל תו יש מפתח – note -שם התו, וערך – duration – משך הזמן של התו בשניות. צד הלקוח (React) מציג את התווים בצורה גרפית באמצעות ספריית.VexFlow

matlab-audio-server > uploads > {} classified_notes.json > [ ] notes

```
{"notes":[
{
    "note":"G",
    "duration":0.47600907029478456
},
{
    "note":"E",
    "duration":0.45
},
{
    "note":"E",
    "duration":0.45
},
{
    "note":"F",
    "duration":0.47600907029478456
},
{
    "note":"D",
    "duration":0.45
},
{
    "note":"D",
    "duration":0.45
},
```

21. מדריך למשתמש

ברוך הבא ל־ Noteify!

שלום וברכה,
תודה שהצטרפת אל *Noteify*, אפליקציית הדגל של CM לזיהוי תווים מתוך שירים – כי גם למוזיקה מגיע תרגום ברור ונוח!

Noteify פותחה מתוך אהבה למוזיקה ומתוך מטרה להקל על תהליך זיהוי וכתיבת תווים בצורה אוטומטית, חכמה ונעימה לשימוש.
השקענו מחשבה רבה בחוויית המשתמש, וכעת תורך להפיק את המקסימום.

הוראות שימוש:

בחר שיר – לחץ על כפתור "Select a file"ובחר קובץ שמע (MP3 או.WAV)

אישור טעינה – שם הקובץ יופיע מעל הכפתור, כך שתדע שהטעינה הצליחה.

רוצה להחליף שיר? - לחץ על "Change song"כדי לבחור קובץ חדש.

התחל ניתוח –לחץ על ."Let's Noteify!"

המתן לעיבוד –האפליקציה תזהה את התווים מתוך השיר. זה לוקח רק רגע.

התווים מוצגים –ברגע שהתהליך מסתיים, תראה את התווים מוזיקליים על גבי תווי נגינה מקצועיים.

ייצוא לקובץ PDF - בלחיצה על "Export PDF" תוכל לשמור את התווים בקובץ באיכות גבוהה, שירד אוטומטית לתיקיית ההורדות בשם: Noteify_export_PDF.pdf

קדימה לנגינה !

רוצה לנתח שיר חדש? - לחץ על כפתור הריענון (🔄) והתחל שוב מהשלב הראשון.

תמיכה טכנית:

לכל שאלה, בקשה או תקלה – אנו זמינים עבורך בפלטפורמות המוכרות.
צוות התמיכה של CM ישמח לעזור.

בהצלחה,
צוות Noteify - תווים שמתנגנים נכון

22. בדיקות והערכה

ממטרות הפרויקט:

- זיהוי אוטומטי של תווים מתוך קובץ שמע
- הצגה גרפית של התווים בממשק ידידותי
- אפשרות לייצוא לפורמט PDF לצורך הדפסה או שיתוף
- ממשק נוח וגמיש גם לאדם שאינו מקצוען בתחום המוזיקה והנגינה

לאחר בדיקה נמצא כי הפרויקט עומד בדרישות כולן: הוא מזהה תווים תוך 6-10 שניות (לקובץ של עד דקה) בצורה מדויקת, מציג אותם בצורה מקצועית, נוח לשימוש, קל להבנה, וידידותי למשתמש.

23. ניתוח יעילות

האלגוריתם שפותח במסגרת פרויקט זה מבצע ניתוח תדרים וזיהוי תווים מוזיקליים מתוך קובץ אודיו באמצעות המרת האות הדיגיטלי לתחום התדר (FFT) , זיהוי תווים לפי תדרים דומיננטיים, סיווג לפי משך זמן והפקת קובץ JSON . האלגוריתם מאורגן בצורה מודולרית ומחולק לשלבים לוגיים ברורים, המאפשרים תחזוקה קלה והרחבה עתידית.

מבחינת סיבוכיות זמן, האלגוריתם נשען על חישוב FFT לכל חלון זמן, FFT פועל בזמן $O(N \log N)$ מה שמביא לסיבוכיות של $O(T \cdot N \log N)$ כאשר:

- T מספר החלונות (frames)
- N גודל החלון (2048)נקודות.

שאר השלבים האחרים כמו מיפוי תדרים לתווים, סיווג משכים וכתיבה לקובץ מבוצעים בסיבוכיות זניחה של O(T) או פחות.

ביצועים אלו נחשבים טובים עבור ניתוח אודיו.

לסיכום, האלגוריתם שומר על איזון טוב בין **פשטות, יעילות ודיוק פונקציונלי** .הוא מתאים במיוחד לפרויקטים העוסקים בזיהוי מלודיות חד־קוליות או אנליזות בסיסיות של קבצי מוזיקה. הוא בנוי בצורה שניתן יהיה להרחיבו, בעזרת מספר שיפורים, גם לניתוח מתקדם יותר של מוזיקה מורכבת, בצורה קלה.

24. אבטחת מידע

לא רלוונטי

25. מסקנות

ביצוע הפרויקט היה מעבר לעוד משימה אקדמאית, הוא היה מעין סטאז' מרוכז בעולם האמיתי של הפיתוח, כזה שלא מסתכם בשאלות אמריקאיות, אלא באתגרים, תסכולים... ותחושת סיפוק.

כשניגשתי לתכנון, הרגשתי כאילו אני עומדת מול משהו ענק – חיבור בין מוזיקה, קוד, רצון ויכולת. הדרישות, התכנון, הדוקומנטציה – הכול נראה בהתחלה כמו תזמורת ללא מנצח...

אבל אחרי שנשמתי עמוק, מיפיתי את כל הדרישות וחילקתי את המשימות לחלקים קטנים – גיליתי שזו בכלל סימפוניה שאפשר לנצח עליה.

דרך הפרויקט רכשתי המון כלים ומיומנות שלא היו נלמדות בקורס רגיל: איך ללמוד לבד שפה חדשה כמעט מאפס, ואיך לחפש מידע ועזרה בפלטפורמות השונות. איך לנתח קובץ אודיו לעומק, איך להבחין בין תו אמיתי לרעש רקע, ואיך להציג את כל זה בצורה נקייה, קריאה ואפילו נעימה לעין. למדתי על מבני קבצים, אלגוריתמים, תיאום בין צד שרת ללקוח, ואפילו על איך לגרום ל־ VexFlow לנגן לפי החליל שלי.

למדתי גם לעבוד לבד – כלומר: להיות גם המפתחת, גם הבודקת, גם זו שמתעצבנת כשהדברים לא עובדים, וגם זו ששמחה כשפתאום יש סנכרון מושלם בין השיר לתו. ההתמודדות עם באגים, שגיאות סנכרון ו"מאיפה הוא *המציא* את התו הזה, למען ה'??!!" – חישלה אותי לכל פרויקט עתידי.

למדתי להתמודד עם כל סוגי ותחומי התכנות, החל משלב הרעיון, המחקר והפיתוח ועד לשלב הבדיקות והשקת הפרויקט. מה שנתן לי ניסיון ואפשרות לעבוד בכל תחום בעתיד.

לסיכום, לאחר שעות רבות של קוד, ניסויים והרבה אוזן קשבת (לפעמים תרתי משמע) אני מרגישה שפרויקט זה נתן לי קפיצה אמיתית מבחינה טכנית, מקצועית ואישית. ובעזרת ה', זו רק ההתחלה.

> **לא נכשלתי, הצלחתי למצוא 10,000 דרכים איך זה לא עובד**
>
> **תומס אלווה אדיסון**

26. פיתוחים עתידיים

במהלך העבודה התברר כי הפוטנציאל הטמון באלגוריתם שפותח רחב ומסקרן, וכי ניתן להמשיך לפתח ולהעמיק בו בכיוונים מגוונים:

הרחבת התמיכה לכלי נגינה נוספים – על מנת לאפשר תמיכה גם לכלי מיתר או נשיפה. כי גם לחליל צד ולבנג'ו מגיעה במה!

ממשק אינטראקטיבי בדפדפן – הוספת יכולת לנגן את התווים שזוהו, לצבוע תווים לפי סוג או אורך, ולאפשר תיקונים על ידי המשתמש.

תמיכה ביצירה מרובת ערוצים - כדי שנוכל לנתח רביעיית מיתרים או את היצירה הבאה של מוצרט בלי להזיע.

אינטגרציה עם אפליקציה עתידית - שתשמע צליל, תזהה את התו, תציע לו אקורד תואם – ואם נשאר זמן, גם תציע כוס קפה😉

27. בביבליוגרפיה

- Wikipedia- Glossary of music terminology
- Wikipedia- Music theory
- Wikipedia- Music theory2
- YouTube
- IEEE
- Wikipedia- FFT
- vlib.eitan.ac.il
- Wikipedia- MATLAB
- MATLAB
- MATLAB courses
- www.systematics.co.il
- chat Gpt
- magenta
- tandfonline
- pubs.aip