מסמך איפיון פרוייקט - הגנת סייבר

בנושא – אפליקציה לשמיעת מוזיקה מותאמת אישית בסגנון ספוטיפיי



מגיש – עידן רטיק

**הגדרת הפרוייקט**

הפורייקט יהווה ספריית מוזיקה, שמאפשרת השמעה של שירים מתוך מגוון שירים רב. המערכת תאפשר הרשמה של משתמשים, ותאפשר לכל משתמש ליצור רשימות השמעה, ולשמור שירים ולשתף שירים באמצעות קישור. המערכת תדע לנגן שירים לבקשת המשתמש, ותדע להציג מילים על המסך, בסנכרון לניגון של המוזיקה.

בנוסף, המערכת תגבש על כל משתמש את הטעם שלו, ותמליץ לו על שירים/רשימות השמעה בהתאם לכך.

בחרתי בפרוייקט זה כיוון שהוא נראה מעניין, ומתעסק רבות בתחומי הסאונד, דבר שאני מתעניין בו ולא יצא לי לעסוק בו עדיין. האתגרים שאני צופה בפרוייקט הם סנכרון הסאונד עם המילים, שמירה ופתיחה של קבצי סאונד בצורה מהירה ויעילה והתנהלות עם קישורים – לאפשר למשתמשים לשלוח זה לזה קישורים של שירים, וכן כל הנושא של למידת מכונה כדי להמליץ למשתמש על שירים

הפרוייקט מיועד לכלל האנשים.

הפרויקט נועד ליצור חוויית מוזיקה חדשה ומלאה שמאפשרת לכל משתמש לגשת, לגלות ולשתף מוזיקה בדרך אינטואיטיבית ומותאמת אישית. אחת המטרות העיקריות היא להנגיש את כל עולם המוזיקה – מסגנונות מוכרים ועד לאמנים חדשים – בצורה שהופכת את המוזיקה לחלק בלתי נפרד מהיום-יום של כל אחד ואחת.

אשאף לספק ממשק שימוש פשוט ויפה, שדרכו ניתן לעבור בקלות בין רשימות השמעה מותאמות, לעקוב אחרי אמנים אהובים, לקבל המלצות מותאמות אישית ולקבל חוויות שמע איכותיות. בנוסף, נרצה להניח יסודות לתכנים חווייתיים נוספים כמו רשימות השמעה אוטומטיות, המלצות לפי מצב רוח וז'אנר, וגישה לתוכן בלעדי.

כדי שהמערכת תעבוד ביעילות עם משתמשים רבים, נצטרך לפתוח, לעכל, ולשלוח קבצי סאונד בצורה מהירה מאוד. בסופו של דבר המערכת תדע לנגן שיר בכל אורך באופן כמעט מיידי מרגע קבלת הבקשה, כך שהמשתמש לא יחכה, ולא יתקע באמצע שירים.

בנוסף, המערכת תיתן אפשרות של הורדת שירים למחשב של המשתמש, לכן נצטרך דרך יעילה לשלוח קבצי סאונד שלמים אל המשתמש ולהוריד אותם בצורה יעילה.

המערכת תהיה כמעט זהה לאפליקציה קיימת – ספוטיפיי – אך יהיו שינויים מסוימים בgui או בפרטים הקטנים של הפונקציונליות.

**הטכנולוגיות שבאות לידי ביטוי בפרוייקט**

אלגוריתם המלצות – כדי להתאים את חוויית המוזיקה לכל משתמש, נממש אלגוריתם למידת מכונה, שידע להמליץ בצורה טוב על שירים לפי אופי המשתמש. אלגוריתם זה יהוה חלק גדול מהחלק האלגוריתמי והקושי התכנותי בפרוייקט.

תשתיות עבודה עם מסדי נתונים מבוזרים – האפליקציה צפויה לנהל ספריית מוזיקה גדולה, הכוללת קבצי אודיו, תמונות, ומידע נוסף, כך שיכולה להידרש טכנולוגיה מתקדמת לניהול מסדי נתונים מבוזרים ולהבטחת נגישות מהירה ויציבה.

אופטימיזציה של זרמי נתונים (Streaming) – כדי להבטיח השמעה רציפה ואיכותית, ייתכן ונשלב כלים מתקדמים לניהול ותזמון זרימת נתונים. טרייד אוף בין יעילות הסטרימינג לבין איכות ההשמעה.

חיפוש שיר מתוך רשימת שירים – המערכת תחזיק מספר רב של שירים, והמשתמש יחפש בתוכה את השיר שהוא רוצה. יהיה צורך לממש אלגוריתם חיפוש מחרוזות, כדי להבטיח שחיפוש זה יהיה יעיל ומהיר.

כל הטכנלוגיות הללו קיימות ומוכרות בשוק ונעשה עליהם מחקר רב. לא צפויים קשיים במציאת הסברים/שיעורים על נושאים אלה באינטרנט, עיקר הקושי יהיה לא בהמצאת האלגוריתמים אלא במימוש.

**תיחום הפרוייקט**

הפורייקט מתעסק רבות הן ברשתות והן במערכות הפעלה.

רשתות – שימוש בפרוטוקול מותאם בשכבת האפליקציה על UDP.

השימוש בUDP הוא כיוון שבהעברת קבצי סאונד פספוס מסוים לא ישפיע על איכות ההשמעה, וUDP מציע תעבורה יותר מהירה אך פחות מדויקת ביחס לTCP, בדיוק כפי שהפורייקט דורש. מסמך המתאר את הפרוטוקול בשכבת האפליקציה יצורף בנפרד.

מערכות הפעלה – שימוש בsubprocesses ות׳רדים כדי לייעל תקשורת עם מספר משתמשים במקביל.

בנוסף, שימוש רב במערכת קבצים כדי לשמור קבצי סאונד ולקמפרס אותם בצורה יעילה, ושימוש באמצעי סאונד המחוברים למחשב.

המערכת תטפל במידה מוחלטת בשגיאות ברשתות, אך במידה מסוימת בשגיאות הקשורות במערכת ההפעלה. הפרוייקט לא כולל שום התעסקות עם התקני Bluetooth למינהם ורכיבי חומרה אחרים.

המערכת לא תקבל input דרך מיקרופון, והמשתמש לא יכול להעלות קבצים מהמחשב שלו.

**תיאור מפורט של המערכת**

מערכת המוזיקה דמוית ספוטיפיי מציעה חוויית שימוש מהנה ונגישה, תוך התאמה אישית והזרמת מוזיקה באיכות גבוהה. המערכת מתוכננת לפשט את התהליך עבור המשתמשים – למצוא מוזיקה שהם אוהבים, לקבל המלצות, לבנות רשימות השמעה אישיות ולשמור על נוחות וגמישות בעבודה עם המוזיקה שלהם. הפרויקט שם דגש על חוויית משתמש קלה, שירותי נתונים מתקדמים ותמיכה במגוון רחב של פלטפורמות. להלן תיאור הרכיבים המרכזיים:

**ממשק משתמש (User Interface)**

ממשק נקי ואינטואיטיבי – המסך הראשי מציג רשימות השמעה מומלצות, שירים מושמעים לאחרונה, והמלצות מותאמות אישית. הממשק מאפשר מעבר מהיר בין מסכי חיפוש, ספריית מוזיקה אישית, ושירותי מוזיקה מתקדמים.

חיפוש מתקדם – החיפוש מאפשר למשתמשים לאתר שירים לפי שם, אמן, ז'אנר או אלבום, כולל אפשרות להשלמה אוטומטית ולהצגת תוצאות מהירות. תוצאות החיפוש מתעדכנות בהתאם להרגלי האזנה קודמים.

פרופיל משתמש מותאם אישית – פרופיל המשתמש מציג היסטוריית האזנה, רשימות השמעה אישיות, אמנים מועדפים וסטטיסטיקות על השירים הנבחרים.

**ניהול ספריית המוזיקה**

מסד נתונים מרכזי – ספריית המוזיקה כוללת קבצי אודיו, תמונות ומידע נוסף על כל שיר (שם, אמן, תאריך הוצאה, ז'אנר, אורך וכו'). הספרייה המרכזית מאפשרת גישה מהירה וקלה לכל שיר בקטלוג.

טכנולוגיית אחסון מבוזרת –

בזכות מבנה נתונים מבוזר, ניתן לתמוך במיליוני משתמשים פעילים מבלי להעמיס על השרתים, וכך לשמור על חוויית שימוש חלקה גם בעומסים כבדים.

ניהול קובצי אודיו באיכות גבוהה ודחוסה –

השירים נשמרים בפורמט דחוס באיכות שמע גבוהה, כדי לחסוך במשאבים ולהבטיח חוויית שמיעה איכותית.

**מערכת המלצות**

למידת מכונה להתאמה אישית – מערכת ההמלצות משתמשת באלגוריתמים של למידת מכונה כדי לנתח את הרגלי האזנה של המשתמש ולהתאים עבורו שירים, רשימות השמעה ואמנים.

פילוח דמוגרפי מתקדם – המלצות המוזיקה יכולות להתבצע לפי פילוחים דמוגרפיים שונים (גיל, מיקום וכו'), כל זאת תוך שמירה על פרטיות המשתמש.

ה**זרמת מדיה**

פרוטוקול הזרמה מותאם (Streaming Protocol) – המערכת משתמשת בפרוטוקול הזרמה שמספק שמע איכותי ותואם לרוחב הפס של המשתמש, ומבטיח השמעה חלקה בכל תנאי.

מנגנון השמעה רציף – השירים נטענים מראש במטמון חכם כדי לשמור על השמעה רציפה ולטעון שירים באופן מיידי, במיוחד במצבים של שינויי חיבור או חיבור בלתי יציב.

ניהול איכות שמע דינמי – המערכת מתאימה את איכות השמע בזמן אמת בהתאם לאיכות חיבור האינטרנט, ומאפשרת האזנה באיכות גבוהה ככל האפשר בתנאים הקיימים, כולל אפשרות להורדת איכות כדי לשמור על זרימה רציפה במצבי עומס.

**מערכת ניהול משתמשים ואבטחה**

ניהול הרשאות ותפקידי משתמשים – תמיכה ברמות גישה שונות והתאמה אישית של ההגדרות.

אבטחת מידע מקיפה – המערכת משתמשת בהצפנה על כל המידע, הן השמור במסד זיכרון והן המידע העובר בתקשורת, כולל שימוש באלגוריתמי הצפנה מוכרים כגון RSA ו AES.

אימות דו-שלבי (2FA) – לשיפור האבטחה, המערכת מציעה אפשרות לאימות דו-שלבי כדי להבטיח גישה בטוחה לחשבונות מכל מכשיר.

**סוגי משתמשים והרשאות**

לכל משתמש ניתנות היכולות לנגן שירים, לאהוב שירים, להוריד שירים, ליצור רשימות השמעה, לשמור רשימות השמעה של משתמשים אחרים, לשתף שירים בעזרת קישור, לחפש שירים/זמרים/אלבומים/ז׳אנרים ולעצב את הפרופיל שלו (לשנות שם משתמש, לכתוב תיאור להוסיף תמונה וכו׳)

למשתמש בהרשאת מנהל תינתן האופציה להוריד/להוסיף שירים ,למחוק משתמשים, לשנות כל דבר בנוגע לשירים או משתמשים אחרים.

המטרה היא שמנהל יוכל לתפעל את האפליקציה מתוך הGUI עצמו ולא רק מתוך הקוד.

אופציונלי –

למשתמש בהרשאת זמר, תינתן האפשרות להעלות שירים/אלבומים, למחוק שירים/אלבומים שהעלה, לערוך שמות של שירים וכדו׳.

**בדיקות**

בדיקת חיפוש שירים:

מטרה: לוודא שחיפוש לפי שם שיר, שם אמן או ז'אנר פועל כראוי.

איך לבדוק: להזין מילות חיפוש שונות ולוודא שתוצאות החיפוש מתאימות ומציינות את הקריטריונים שהוזנו (למשל חיפוש לפי שם אמן יציג את כל השירים של אותו אמן בלבד.

בדיקת השמעת שירים:

מטרה: לוודא שמשתמש יכול להפעיל ולעצור שירים, להריץ קדימה ואחורה.

איך לבדוק: להפעיל שיר, לעצור אותו, לנסות להריץ 10 שניות קדימה ואחורה, ולוודא שהשיר ממשיך מהמיקום הנכון.

בדיקת רשימות השמעה אישיות:

מטרה: לבדוק שהמשתמש יכול ליצור, לערוך ולמחוק רשימות השמעה אישיות.

איך לבדוק: ליצור רשימת השמעה, להוסיף לה שירים, להסיר חלק מהם, למחוק את הרשימה ולוודא שכל הפעולות נשמרות ומוצגות נכון.

בדיקת פרופיל משתמש:

מטרה: לבדוק שהיסטוריית ההאזנה, אמנים מועדפים וסטטיסטיקות על השירים נשמרים ומתעדכנים.

איך לבדוק: להאזין למספר שירים, לבדוק אם הם מופיעים בהיסטוריית האזנה, לסמן אמנים מועדפים ולוודא שהסטטיסטיקות מתעדכנות בהתאם.

בדיקת שמירת מידע במסד הנתונים:

מטרה: לוודא שכל המידע האישי של המשתמשים (כמו רשימות השמעה ופרופיל) נשמר כראוי.

איך לבדוק: לבצע פעולות שימור שונות כמו יצירת רשימת השמעה, חיפוש ושמירת העדפות אישיות, ולוודא שהנתונים נשמרים נכון על ידי בדיקה ישירה במסד הנתונים.

בדיקת התאמה אישית של המלצות:

מטרה: לוודא שהמלצות השירים מותאמות להעדפות והיסטוריית האזנה של המשתמש.

איך לבדוק: ליצור פרופיל חדש, להאזין למספר שירים בז'אנר מסוים ולראות אם ההמלצות משקפות את סוגי השירים שהושמעו.

בדיקת תגובתיות בממשק החיפוש:

מטרה: לבדוק שהממשק מגיב במהירות גם כאשר המשתמשים מרבים להשתמש בפונקציית החיפוש.

איך לבדוק: לבצע חיפושים רבים ברצף ובאופן מקבילי, ולוודא שהממשק מגיב במהירות ושאינו מציג עיכובים או תקלות.

בדיקת הרשאות משתמשים:

מטרה: לוודא שניהול הרשאות עובד ושמשתמשים עם הרשאות שונות מוגבלים בהתאם.

איך לבדוק: לנסות לבצע פעולות כמשתמש רגיל, משתמש מנהל, וכהורה עם בקרת הורים, ולוודא שכל קבוצה מוגבלת לפעולות המותרות לה.

בדיקת ניגון רציף והזרמת מדיה:

מטרה: לוודא שהמערכת יכולה לנגן שירים באופן רציף גם כאשר איכות האינטרנט משתנה.

איך לבדוק: לנגן שיר ולהקטין את מהירות האינטרנט במהלך ההשמעה, לבדוק אם יש עצירות, ולוודא שהאיכות משתנה בהתאם למהירות.

בדיקת כניסת משתמש:

מטרה: לוודא שהמשתמש יכול להתחבר עם פרטי ההתחברות שלו בצורה בטוחה, כולל אימות דו-שלבי.

איך לבדוק: לבצע כניסה ויציאה של משתמש ממכשירים שונים, לנסות גישה עם אימות דו-שלבי ממכשיר חדש, ולוודא שהגישה מאובטחת.

בדיקת שמירת ההגדרות האישיות:

מטרה: לוודא שהגדרות משתמש, כמו איכות השמעה והגדרות תצוגה, נשמרות ונטענות כראוי.

איך לבדוק: לשנות את ההגדרות, להפעיל מחדש את האפליקציה, ולוודא שההגדרות נשמרו ונראות כפי שהוגדרו.

בדיקת טעינת מסכים ורכיבי ממשק:

מטרה: לוודא שכל רכיבי הממשק נטענים בזמן סביר וללא עיכובים מיותרים.

איך לבדוק: לפתוח את המסכים השונים (חיפוש, פרופיל, ספריית השירים), לוודא שכל הרכיבים נטענים במהירות, ולבדוק שאין איטיות בטעינת נתונים.

**לו״ז**

|  |  |  |  |
| --- | --- | --- | --- |
| **משימה** | **פירוט** | **דד ליין** | **ידע נדרש** |
| בסיס פיתוחי | בניית כל האובייקטים הדרושים, יצירת הקבצים הצפויים. | 1/1 |  |
| תקשורת א׳ | בניית פרוטוקול ומימוש תקשורת בכל הקשור להזרמת מוזיקה. | 12/2 | Streaming |
| תקשורת ב׳ + מוסד נתונים | בניית פרוטוקול ומימוש תקשורת בכל הקשור לניהול משתמשים והרשמה, ושמירה במסד נתונים. בשלב זה תתווסף הצפנה. | 1/3 |  |
| מסד נתונים של שירים | הקמת מסד נתונים עם כל השירים הדרושים ושמירתם בצד שרת בצורה טובה ויעילה. | 1/4 | שימוש בספוטיפיי api, והתעסקות עם קבצי mp4. |
| Gui | ממשק משתמש גרפי בהרשמה, ובאפליקציה עצמה. | 10/4 | שימוש בפלטפורמת gui. |
|  |  |  |  |
|  |  |  |  |

**ניהול סיכונים**

|  |  |  |  |
| --- | --- | --- | --- |
| **סיכון** | **תיאור** | **דרגת סיכון** | **דרכי התמודדות** |
| עומס על שרתים | נפילות או עיכובים בביצוע פעולות עקב כמות משתמשים גבוהה או עומסי תעבורה. | גבוהה | - שימוש בשירותי ענן לניהול דינמי של מ  שאבים. - יישום Load Balancer. - ניטור ביצועים והתרעות בזמן אמת. |
| SQL Injection | מתקפות שבהן מוזרק קוד זדוני למסדי הנתונים דרך קלט משתמש. | גבוהה מאוד | - אימות והגבלת קלט משתמשים. - שימוש בפרמטרים מוכנים מראש (Prepared Statements). - ניטור ואיתור פרצות. |
| חוסר יעילות בקוד | קוד שאינו אופטימלי עלול לגרום לביצועים איטיים ולבעיות סקלאביליות. | בינונית | - ביצוע Code Reviews תכופים. - שימוש בפרופיילרים לאיתור צווארי בקבוק בקוד. - בדיקות עומסים אוטומטיות. |
| דליפת נתונים | דליפת מידע אישי או עסקי בעקבות פריצות או טעות בקוד. | גבוהה מאוד | - הטמעת הצפנת נתונים (SSL/TLS ו-AES). - שימוש במדיניות הרשאות מוגבלת (RBAC). - בדיקות חדירות תקופתיות. |
| תקלות ב-API חיצוניים | שיבושים בתקשורת או חוסר זמינות של שירותי צד שלישי, כגון שירותי תמלוגים או זרימת מוזיקה. | בינונית | - כתיבת קוד שמזהה ומטפל באי זמינות זמנית (Retry Mechanism). - שימוש ב-Circuit Breaker Design Pattern. |
| חשיפת מפתחות API | מפתחות API גלויים בקוד יכולים לשמש גורמים זדוניים. | גבוהה | - אחסון מפתחות API בקבצים מוצפנים (כגון .env). - הגבלת שימוש לפי כתובת IP או כמות בקשות מוגדרת מראש. |
| בעיות תאימות דפדפנים | תפקוד לא תקין של המערכת בדפדפנים או במכשירים מסוימים. | נמוכה | - ביצוע בדיקות תאימות לדפדפנים שונים (Cross-Browser Testing). - שימוש בתקנים מודרניים כמו HTML5 ו-CSS3. |
| שימוש יתר במשאבי מערכת | שימוש לא מבוקר בזיכרון או במעבד עלול לגרום לקריסות. | בינונית | - ביצוע בדיקות עומסים ותקלות (Stress Testing). - יישום ניהול יעיל של משאבים בתוכנה (Garbage Collection). |

**פירוט יכולות**

**יכולות בצד שרת**

**ניהול משתמשים**

תיאור: ניהול הרשמה, התחברות וניהול פרטי משתמשים.

פעולות נדרשות:

יצירת והפעלת REST API לניהול פעולות משתמשים (התחברות, הרשמה, עריכה).

הטמעת אימות (Authentication) עם JWT.

שמירת סיסמאות בצורה מאובטחת.

אובייקטים נחוצים:

מודל נתונים לניהול משתמשים (..

**ניהול מוזיקה**

תיאור: אחסון וניהול קבצי מוזיקה ומידע על שירים.

פעולות נדרשות:

יצירת שירות להעלאת קבצים ויצירת מטא-דאטה.

פיתוח API לחיפוש מוזיקה על בסיס פרמטרים כמו שם שיר או ז'אנר.

אובייקטים נחוצים:

מודל נתונים לשירים (.

שירותי צד שרת לעיבוד והעלאת קבצים.

**ניהול פלייליסטים**

תיאור: שמירת פלייליסטים מותאמים אישית וניהולם.

פעולות נדרשות:

פיתוח REST API ליצירה, עדכון ומחיקה של פלייליסטים.

עיבוד בקשות משתמשים לשיתוף פלייליסטים.

אובייקטים נחוצים:

מודל נתונים לניהול פלייליסט.

לוגיקת צד שרת לשיתוף נתונים עם משתמשים אחרים.

**סטרימינג של מוזיקה**

תיאור: העברת קבצי מוזיקה בצורה רציפה למשתמשים.

פעולות נדרשות:

פיתוח שירות Streaming מבוסס על פרוטוקולים כמו HLS.

כתיבת API לניהול בקשות סטרימינג של שירים.

אובייקטים נחוצים:

מחלקה ייעודית לניתוב קבצים בתצורת זרימה.

**יכולות בצד לקוח**

**ניהול חשבון משתמש**

תיאור: יצירת ממשק הרשמה, התחברות וניהול פרטים אישיים.

פעולות נדרשות:

פיתוח דף התחברות והרשמה.

שילוב בקשות HTTP לשרת עם מידע משתמש.

אובייקטים נחוצים:

רכיבי UI לטפסים

פונקציות API בצד לקוח לניהול בקשות

**חיפוש מוזיקה**

תיאור: חיפוש שירים על בסיס פרמטרים כמו שם אמן, שיר או ז'אנר.

פעולות נדרשות:

יצירת שדה חיפוש והצגת תוצאות בצורה ברורה.

קריאה ל-API לקבלת תוצאות חיפוש בזמן אמת.

אובייקטים נחוצים:

רכיב חיפוש

פונקציה שמתקשרת ל-API

**ניהול פלייליסטים**

תיאור: יצירת ממשק לניהול פלייליסטים ושירים בתוכם.

פעולות נדרשות:

פיתוח דפים או מודולים ליצירה וסידור פלייליסטים.

אינטגרציה עם API ליצירת רשימות השמעה ושמירתן.

אובייקטים נחוצים:

רכיב UI לניהול פלייליסטים.

פונקציות צד לקוח לביצוע פעולות CRUD על פלייליסטים

**נגן מוזיקה**

תיאור: ממשק להשמעת מוזיקה עם שליטה בסיסית כמו השהיה, עצירה, דילוג קדימה/אחורה וניהול עוצמת קול.

פעולות נדרשות:

פיתוח רכיב גרפי לנגן המוזיקה.

שילוב פונקציות לניהול זרימה ופעולות משתמש.

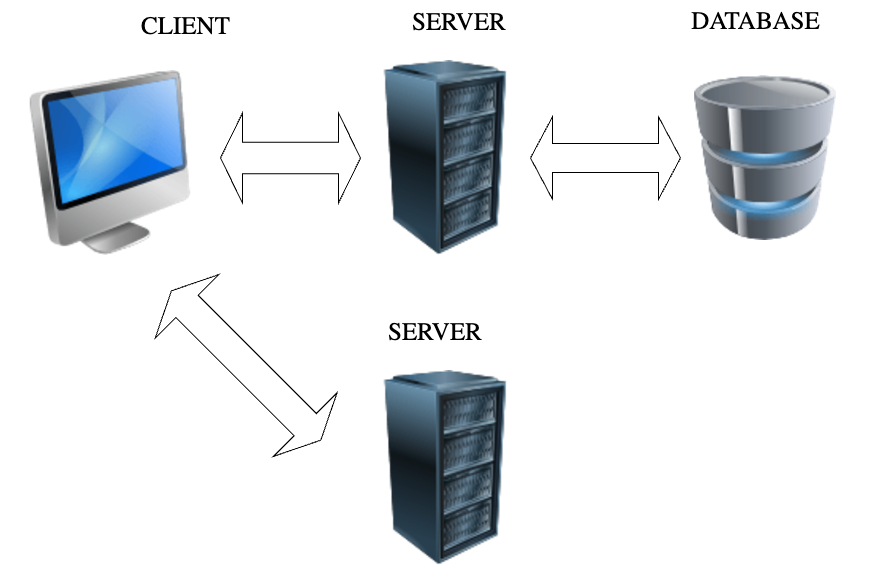
אובייקטים נחוצים:

רכיב נגן מוזיקה

פונקציה לניהול סטרימינג

**ארכיטקטורת תקשורת**

בארכיטקטורה שנשתמש יש שני שרתים. שרת אחד שמנהל את השירותים הקשורים לאפליקציה עצמה, ושרת אחד שעוסק בlogin, וsignup.



תרשים 1 – ארכיטקטורת תקשורת

כאשר שרת פותח את האפליקציה לראשונה, הוא יתחבר לשרת שמתעסק ברישום משתמשים.

מול שרת זה יתבקש המשתמש להיכנס בתור משתמש רשום, ובעזרת תקשורת עם שרת כזה ייצור משתמש בעצמו. למעשה שרת זה מטפל אך ורק בכל הקשור לרישום משתמשים לרבות שליחת שגיאות בזמן ההרשמה, 2FA וכל היוצא בזאת.

לאחר שההרשמה עברה בהצלחה, בעזרת הפרוטוקול שיוגדר בהמשך, ישלח למשתמש את הכתובת של השרת שמנהל את שירותי האפליקציה.

**מדוע?**

לארכיטקטורה זו שתי יתרונות עיקריים.

יעילות – בגלל ששני שרתים פועלים, הם מחלקים בינהם את העומס. משתמשים אשר כבר מחוברים יתקשרו עם שרת האפליקציה ולכן לא ירגישו אם במקביל הרבה משתמשים רוצים להירשם שכן הם מתקשרים מול שרת ההרשמות.

תחזוקה וסדר – מערכת המפרידה באופן מובהק בין שימוש בשירותי האפליקציה לבין הרשמה, קלה יותר לתחזוק ומודולורית יותר, והתכנות יהיה מסודר וקריא יותר.

**איך זה בא לידי ביטוי בהצפנה?**

תהיינה לחיצת יד הן לשרת הנתונים והן לשרת האפליקציה. בלחיצת היד עם שרת ההרשמות, תהיה החלפת מפתחות בעזרת RSA, מפתחות אשר ישמשו לsession התקשורת עם שרת ההרשמות. כאשר המשתמש יופנה לשרת האפליקציה, תתבצע החלפת מפתחות נוספת עם שרת האפליקציה.

כלומר לשרת ההרשמות ולשרת האפליקציה יש מפתחות שונים בתקשורת עם כל משתמש, כאשר המשתמש יודע שהוא צריך להשתמש במפתח מסוים לתקשורת עם שרת ההרשמות ובמפתח אחר לתקשורת עם האפליקציה.

המטרה של שימוש זה בהצפנה הוא למזער נזקים – במקרה ופורצים את אחד מהשרתים, שהמידע של השרת השני לא יהיה חשוף.

במידה ולאחר סיום ההתחברות הראשונית המשתמש ירצה לעשות מהלך הקשור למסד הנתונים כגון החלפת סיסמא, שם משתמש או מייל, המשתמש ידע לפנות לשרת ההרשמות כדי לנהל בקשות אלה מולו.

בפרוטקולי התקשורת יפורט כיצד כל זה נעשה.

**מסדי נתונים**

המערכת מתנהל עם שני מסדי נתונים.

מסד נתונים ראשון יחזיק בתוכו אך ורק שמות משתמש, סיסמאות ומיילים. הסיסמאות תהיינה מקודדות באמצעות פונקציית hash, עם שימוש במלח ופלפל. למסדר נתונים זה יהיה ניתן לגשת אך ורק דרך השרת המטפל בהרשמות.

מסד הנתונים השני ישמור את פרטי המשתמש ברמת האפליקציה – רשימות השמעה, שירים אהובים וכיוצא בזאת.

עדיין לא הוחלט אם יהיה מסד נתונים שלישי – המכיל בתוכו את כל השירים, ובאופן כללי איך לשמור את השירים.

מסד הנתונים הראשון יהיה בsql.

מסד