**תכנון ותכנות מערכות טלפוניים חכמים תחת מערכת ההפעלה אנדרואיד**

נושא העבודה : Animengine

שם התלמיד : יאיר ישראל סאלדמן

ת"ז של התלמיד : 330766833

שם המורה : מיכל אוסטרוב

שם בית ספר ועיר : כרמים בנימינה – גבעת עדה

# תוכן עניינים וראשי פרקים

פה תוסיפו תוכן עניינים שה-word יוצר אוטומטית ע"י:

הפניות--->תוכן עניינים---->הוסף תוכן עניינים

בהמשך העבודה, אם התיק משתנה, תצטרכו להיכנס במסלול הנ"ל ולבחור "עדכן תוכן עניינים"

לא לשכוח: כדי שייווצר תוכן עניינים כזה, כל הכותרות וכותרות המשנה חייבות להיות מוגדרות/כתובות בסגנון של כותרת...

# פרק א – 'Animengine'- ייזום

## תיאור ראשוני של המערכת

האפליקציה בנויה כך שמשתמש אשר חבר באפליקציה ונכנס פעם ראשונה לאפליקציה עונה על האם מתעניין בסדרות אנימה מוכרות ביותר ועל פי כך האפליקציה בונה עבורו המלצות לאנימות אשר נראות כהכי מתאימות עבורו . בהמלצה על האנימה יופיע תיאור של האנימה וגם טריילר שמציג אותה.

## הגדרת הלקוח

המערכת מיועדת עבור אנשים אשר מתעניינים בסוגים שונים של אנימות או שרוצים לגלות טעם חדש של אנימה שלא הכירו לפני, בנוסף לאפליקציה יש משתמש "יוצר" שמשתמש זה מוסיף אנימות לפי המלצתו בנוסף לטריילר ותיאור.

## ג. הגדרת יעדים/מטרות

* לראות המלצות של אנימות לפי הטעם שלך.
* לצפות בטריילר של האנימה שהומלצה לך.
* משתמש "יוצר" מוסיף אנימות לפי בחירתו.

## בעיות, תועלות וחסכונות

* הבעיה שהאפליקציה באה לפתור היא עבור אנשים שאין להם ידע מספיק על אנימות חדשות עבורם או אנשים אשר רוצים למצוא אנימה חדשה שלא שמעו אליה.
* השירותים שהאפליקציה מאפשרת:

להארות טריילרים לאנימות

הוספה של אנימות על ידי משתמש "יוצר" , שמאושר על ידי מנהל האפליקציה

* ישנה אפליקצייה דומה בשם Rotten Tomatoes אשר מציגה פרטים על סרטים וסדרות ומביעה ביקורות ודירוג, ההבדל הוא שהאפליקציה שלי בנויה כך שהיא תמליץ למשתמש על הסדרה שהכי מתאימה לו לפי טעמו האישי בעוד שב Rotten Tomatoes המידע על הסדרות והסרטים והביקורות הן כלליות ולא מתאימות לכל משתמש בנפרד.

## האם צפויים קשיים או מגבלות בהגדרת המערכת

* תיתכן הגבלה בכמות הטריילרים והאנימות, שאותה אפתור במימוש.
* מדובר בטכנולוגיה מוכרת וידועה
* תיתכן בעיה בשימוש ב YT.

## תיחום הפרויקט

* המערכת לא נותנת המלצה על אתרים שבהם ניתן לראות את האנימות וגם לא ניתן לשתף את הטריילרים מחוץ לאפליקציה.
* במידה ויהיה לי זמן אני אוסיף לאפליקציה AI אשר הוא ימליץ למשתמש על האנימות.

# פרק ב' - ''Animengine- אפיון

## פרוט המערכת:

* המשתמש באפליקציה יוכל לקבל המלצות על אנימות ולהחליט האם לצפות בטריילרים של אותה האנימה, ולדרג האם הסדרה הינה לטעמו או לא. בנוסף המשתמש יוכל לקרוא תקציר לאותה האנימה במידה ולא הבין מהטריילר או אם הוא מעוניין לדעת עוד קצת על האנימה. את ההמלצות על האנימות האפליקציה תדע לדרג לבד על פי הסקר שאותו המשתמש מקבל עם פתיחת האפליקציה.
* למשתמש יוצר יש ממשק שונה הנותן לו את האפשרויות להוסיף אנימות על פי החלטתו שלו.

## מה היכולות שהיא תעניק למשתמש , פירוט היכולות:

**למשתמש רגיל:**

* 1. להירשם כמשתמש
  2. להתחבר כמשתמש
  3. לקבל המלצות על אנימות
  4. לצפות בטריילרים
  5. לראות תיאור על האנימה
  6. לחפש אנימות חדשות

**למשתמש** יוצר:

1. להתחבר כמשתמש יוצר
2. לעלות המלצה ותקציר לאנימה
3. לעלות טריילר לאנימה

## פירוט בדיקות קופסה שחורה (בדיקות שלא נדרש הקוד אלא רק ה APK לטובת ביצוען.)

| **מספר** | **שם הבדיקה** | **מה אמורה לבדוק** | **איך מתכננים לבדוק** |
| --- | --- | --- | --- |
| 1. | הרשמה לאפליקציה | האם כאשר המשתמש יוצר משתמש האפליקציה מעדכנת אותו כמשתמש חדש? | נרשמים לאפליקציה עם שם משתמש, סיסמא, מייל סוגרים את האפליקציה ומנסים להתחבר מחדש עם שם המשתמש והסיסמא שאיתם המשתמש נרשם. |
| 2. | התחברות לאפליקציה | האם המשתמש יכול להיכנס לאפליקציה? | לאחר שנרשמים לאפליקציה סוגרים אותה ומנסים להתחבר לאפליקציה עם שם המשתמש והסיסמא שלהם לאחר מכן מנסים לעשות אותו דבר רק ששמים סיסמא אחרת כדי לבדוק כי לא יהיה ניתן להתחבר עם סיסמא אחרת למשתמש כלשהו. |
| 3. | צפייה בטריילרים | האם הטריילר זמין ופועל כמו שצריך ללא בעיות? | להתחבר כשמתמש "רגיל" וללחוץ על הטריילר שהוא מעוניין בו ולראות האם הוא מתנגן. |
| 4. | אופציית "שכחתי סיסמא" | האם כאשר המשתמש לוחץ על כפתור "שכחתי סיסמא" האפליקציה תזהה את המייל של המשתמש? | ניצור משתמש יוצר ורגיל ונלחץ על כפתור "שכחתי סיסמא" ונראה אם האפליקציה זוכרת את המייל של המשתמש. |
| 5. | חיפוש | האם כשמחפשים אנימות באפליקציה (גם כיוצר וגם כרגיל) אמצעי החיפוש והסינון עובדים רגיל? | להתחבר לאפליקציה כמשתמש ולחפש אנימות (או כפתור אוסף אנימה למנהל). ולבדוק כשמוסיפים אנימה או מחפשים אותה מקבלים תוצאה מתאימה |
| 6. | טיפול הוספת אנימה | האם היוצר יכול להוסיף אנימה? | נתחבר לאפליקציה כמשתמש יוצר ומוסיפים אנימה יחד עם שמו ואת סוג האנימה. ומאשרים שהוא נוסף בכך שמחפשים את שמו של האנימה כמשתמש רגיל. |
| 7. | עדכון תיאור האנימה | האם יוצרים אחרים יכולים להוסיף תיאור להאנימה או לערוך אותה | נתחבר לאפליקציה כיוצר ונלך לאנימה קיימת וננסה לערוך אותה ולהוסיף תיאור לאנימה ואז נתחבר כמשתמש רגיל ונראה אם התיאור נוסף/ התעדכן. |

## תכנון לוח זמנים לפרויקט

* תכנון לוח זמנים ראשוני
* עבור כל פעילות בלוח: זמן התחלה מתוכנן, זמן סיום מתוכנן , זמן התחלה בפועל, זמן סיום בפועל והערות

| **פעילות** | **זמן התחלה מתוכנן** | **זמן סיום מתוכנן** | **זמן התחלה בפועל** | **זמן סיום בפועל** | **הערות** |
| --- | --- | --- | --- | --- | --- |
| מסמך ייזום | 3.11.2024 | 4.11.2024 | 3.11.2024 | 17.11.2024 | עבר בדיוק כמו משציפתי |
| מסמך אפיון | 17.11.2024 | 18.11.2024 | 17.11.2024 | 17.11.2024 | עבר בדיוק כמו משציפתי |
| מסמך ניתוח | 20.11.2024 | 21.11.2024 | 24.11.2024 | 2.11.2024 | עבר בדיוק כמו משציפתי |
| מסמך עיצוב | 22.12.2024 | 1.1.2025 | 28.11.2024 | 26.12.2024 | עבר בדיוק כמו משציפתי |
| קוד: אקטיבי כניסה לאפליקציה | 4.1.2025 | 5.1.2025 |  |  |  |
| סקר אנימות אהובות/ מוכרות | 6.1.2025 | 10.1.2025 |  |  |  |
| חיפוש | 15.1.2025 | 25.1.2025 |  |  |  |
| אפשרויות של יוצר | 26.1.2025 | 30.1.2025 |  |  |  |
| צפייה בטריילרים ושימוש ב YT | 5.2.2025 | 15.2.2025 |  |  |  |
| המלצה למשתמש על אנימות | 20.2.2025 | 29.2.2025 |  |  |  |
| הגשה ראשונית | 1.3.2025 | 5.3.2025 |  |  |  |
| סיום תיקון שגיאות | 6.3.2025 | 1.4.2025 |  |  |  |
| הגשה סופית | 2.4.2025 | 1.5.2025 |  |  |  |

* **לוח הזמנים – הינו מסמך חי ויש לעדכנו לאורך כל הפרויקט**

## ניהול סיכונים בפרויקט

* זיהוי סיכוני הפרויקט, ניתוחם ומענה עליהם, הטבלה תמוין מסיכון גבוה לנמוך (הגבוהים באדום, בינוני –כתום , קל – צהוב) סיכון שיוסר יצבע בירוק ויעבור לתחתית המסמך

|  |  |  |  |
| --- | --- | --- | --- |
| **הסיכון** | **פירוט הסיכון** | **רמת הסיכון**  **(קל/ בינוני/ קשה)** | **תיאור דרכים (לפחות 2 ) להתמודדות עם הסיכון ולהקטין אותו** |
| עמידה בלו"ז | לא בטוח שאני אעמוד בלוח הזמנים של הפרויקט ואף אולי יהיה מצב שלא אוכל להגיש בזמן את הפרויקט. | קשה | כל יום לפחות לנסות לשבת על הקוד ועל התיק פרויקט ולנסות לקדם את הפרויקט ולבקש עזרה מהמורה שלי. |
| נתונים | דברים לא נשמרים בממסד הנתונים. | קשה | לבדוק ממצגות איך שומרים בממסד נתונים ולעבור שוב על הקוד. |
| טריילרים | לא יודע כעת כיצד להשתמש ב – YT. | קשה | ללמוד לבד באינטרנט איך עובדים עם זה. |
| ריבוי משתמשים | יותר מידי משתמשים בעלי אותו שם. | קשה | למצוא דרך לעשות בממסד נתונים פעולה שתחפש על פי שמות משתמש (או מייל). |
| חיפוש | אני לא בטוח אם האפליקציה תדע לזהות את שם האנימה. | בינוני | לבדוק האם היוצר העלה אנימה כזאת ולבדוק את ממסד הנתונים. |
| הערות | אני לא בטוח שאני אדע מתי לתעד את הקוד. | קל | לתעד כל פעולה יחד עם טענות כניסה ויציאה. |
| שימוש נכון במבני נתונים | אני לא יודע אם אני אדע מתי להשתמש במבני נתונים מתאים. | קל | בדיקה במצגות ובאינטרנט באיזה מבנה נתונים להשתמש לכל סוג של צורך. |

* **ניהול סיכונים – הינו מסמך חי ויש לעדכנו לאורך כל הפרויקט**

# פרק ג' – 'Animengine'- מסמך ניתוח

## פרוט יכולות המערכת:

הרשמה לאפליקציה

* היא מיועדת כדי ליצור ולשמור משתמש שיוכל לדרג אנימות ולקבל המלצות.
* במסך הפתיחה יש שתי שדות וארבע כפתורים אחד להתחברות עם אימייל וסיסמא שני אחרים להרשמה כמשתמש יוצר או רגיל ואחד לשכחתי סיסמא. לוחצים על כפתור הרשמה ונכנסים למסך שיש בו 4 שדות וכפתור ששם צריך לשים מייל שם פרטי ושם כינוי וסיסמא, לאחר שממלאים את כל השדות לוחצים על הכפתור ליצירת המשתמש ואז המשתמש נשמר בממסד הנתונים.

שכחתי סיסמא

* היא מיעודת עבור משתמשים רגילים או יוצרים ששכחו את סיסמאתם.
* במסך הפתיחה יש שתי שדות וארבע כפתורים אחד להתחברות עם אימייל וסיסמא שני אחרים להרשמה כמשתמש יוצר או רגיל ואחד לשכחתי סיסמא. לוחצים על כפתור "שכחתי סיסמא" ששם מופיע שדה וכפתור שמים בשדה את המייל של המשתמש ולוחצים על כפתור שינוי סיסמא לאחר מכן נשלח למייל של המשתמש הודעה לשינוי הסיסמא וככה יוכל לשנות את הסיסמא שלו.

במסך הפתיחה יש שתי שדות וארבע כפתורים אחד להתחברות עם אימייל וסיסמא שני אחרים להרשמה כמשתמש יוצר או רגיל ואחד לשכחתי סיסמא. לוחצים על כפתור "שכחתי סיסמא" ששם מופיע שדה וכפתור שמים בשדה את המייל של המשתמש ולוחצים על כפתור שינוי סיסמא לאחר מכן נשלח למייל של המשתמש

התחברות לאפליקציה

* היא מיועדת כדי להתחבר כמשתמש לאפליקציה בין עם משתמש יוצר או משתמש רגיל כך שהאפליקציה תוכל להמליץ למשתמש הספיציפי הזה.
* במסך הפתיחה יש שתי שדות ושלושה כפתורים אחד להתחברות אימייל וסיסמא, ושניים ליצירת משתמש (רגיל או יוצר). מקלידים בשדות את האימייל ואת הסיסמא של המשתמש בשדות המתאימים ובוחרים בכפתור התחברות. בודקים בממסד הנתונים אם קיים משתמש עם אותו אימייל וסיסמא אם הוא קיים הוא ימשיך לאפליקציה אם לו יופיע למשתמש הודעה כי המשתמש או הסיסמא לא מתאימים.

להוסיף אנימות

* היא מיועדת כדי שמשתמש יוצר יוכל לראות אנימות חדשות שמשתמש יוצר מעלה לאפליקציה.
* לאחר שיוצר נכנס לאפליקציה יש לו מסך שבו הוא מעלה אנימות שם הוא מזין קישור לטריילר של האנימה שרוצה להוסיף את שמו ואת סוגו ואחרי שהוא לוחץ על כפתור הוספה שם האנימה סוגה והקישור נשמרים בממסד הנתונים והמשתמשים יכולים לקבל המלצות על האנימה שהוא העלה.

לראות אנימות מומלצות

* היא מיעודת כדי שהמשתמש יוכל לראות אנימות לפי הטעם שלו.
* לאחר שהמשתמש נרשם לאפליקציה ומחובר אליה יש לו שורת אנימות מומלצות לפי דירוג של אנשים ושל עצמו והוא יכול לגלגל ולראות את האנימות ולדרג אותם בהתאם.

לחפש אנימות

* היא מיעודת כדי שהמשתמש ימצא אנימה במידה ובא לו לחקור לבד או על מנת לחפש בין האנימות שאהב.
* לאחר שהמשתמש נרשם לאפליקציה ומחובר אליה יש לו היכולת לחפש אנימות לפי שם או סוג ואז האפליקציה תביא לו את האנימה שהוא חיפש.

* קריאת תוכן עניינים לאנימה
* היא מיעודת כדי שמשתמש יוכל לקרוא על מה האנימה עוסקת ולקבל תקציר לעלילה שלה.
* לאחר שהמשתמש נרשם לאפליקציה, והתחבר אליה, ומצא אנימה מופיע בעמוד על האנימה תקציר לעלילה שלה במה היא עוסקת וסיכום קצר לפרק הראשון שלה.

הוספה למעודפים

* היא מיעודת למשתמשים על מנת שיישמרו אנימות שאהבו ורוצים לחקור אליהם מאוחר יותר.
* לאחר שהמשתמש נרשם לאפליקציה, הוא מקבל אנימה יחד עם התיאור והכול ויש לו כפתור הוספה למעודפים (או בלחיצה על כפתור אהבתי) ואנימות אלו נכנסות לתוך החלק של אנימות מעודפות.

צפייה במועדפים

* היא מיעודת למשתמשים שאהבו סדרה שהומלצה להם או סדרה שמצאו.
* לאחר שהמשתמש נרשם לאפליקציה, הוא מקבל אנימה ואם הוא אהב אותה אז האנימה נוספת לרשימת המועדפים של אותו המשתמש כדי שיוכל לגלגל בין האנימות שהוא דירג כאהב.

צפייה בטריילר

* היא מיעודת למשתמשים שרוצים לראות חלק מהסדרה.
* לאחר שהמשתמש נרשם לאפליקציה, הוא יכול להיכנס לאנימה ולראות את הטריילר שלה על מנת להחליט אם היא טובה או לא.

הוספת דירוג לאנימה

* היא מיעודת למשתמשים שרוצים לדרג אנימה.
* לאחר שהמשתמש נרשם לאפליקציה, והוא נכנס לאנימה הוא יכול לדרג אותה בין 1 – 5 כוכבים ודירוג זה יכול להיראות על ידי כל המשתמשים באפליקציה.

צפיה בדירוגים לאנימה.

* היא מיעודת למשתמשים שרוצים לראות דירוגים של משתמשים אחרים לאפליקציה שלהם.
* לאחר שהמשתמש נרשם לאפליקציה, הוא יכול להיכנס לאנימה ולראות דירוגים של משתמשים אחרים לאותה האנימה בין אם זה כמות האנשים שלא אהבו אותה לאלה שאהבו ובין אם זה לראות דירוג ממוצע של כוכבים לאנימה.

עריכת אנימה

* היא מיעודת למשתמשים יוצרים שצריכים לערוך את האנימה שהעלו בגלל טעות.
* לאחר שהמשתמש היוצר נרשם לאפליקציה, הוא יכול להיכנס לאנימה שיצר ולבדוק אם הכול תקין שם בין אם השם שלה הטריילר והתיאור שלה.

הקראת התיאור לאנימה

* היא מיעודת למשתמשים שרוצים להקשיב ל TTS במקום לקרוא את התיאור לאנימה בעצמם.
* לאחר שהמשתמש נרשם לאפליקציה, הוא יכול להיכנס לאנימה ולראות תיאור לאנימה שנכנס אליה, במסך זה יופיע כפתור הקראה ו TTS יקריא למשתמש את כל התיאור לאנימה.

# פרק ד' –'Animengine'– תכנון מסכים מקדים

## תיאור מסכי הפרויקט:

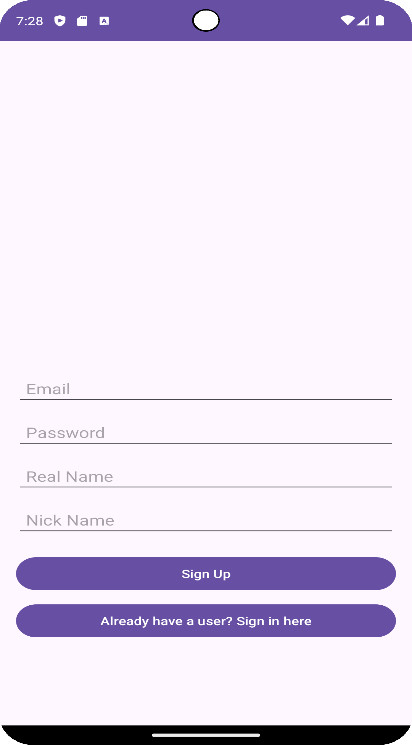
## 

## מסך כניסה:

* מסך כניסה אחראי לעזור ללקוחות לפתוח משתמש חדש באפליקציה (רגיל או יוצר) וגם אחראי על מנת שמשתמשים באפליקציה יוכלו להתחבר לאפליקציה.
* כפתור Sign in מכניס אותך לאפליקציה
* כפתור Forgot your password שולח אותך למסך לשינוי סיסמא.
* כפתור Sign up שולח אותך למסך רישום.
* כפתור Creator sign up שולח אותך למסך רישום ליוצרים.

מסך רישום למשתמשים רגילים ויוצרים (אותו עיצוב):

* מסך הרישום אחראי ליצור משתמשים על סמך המייל שלהם וסיסמתם, וגם עדכון נתוניהם בממסד הנתונים.
* כפתור Sign up שולח למסך ההמלצות
* כפתור Sign in שולח למסך כניסה.



# מסך קבלת המלצות של אנימה

* מסך זה נועד למשתמשים על מנת שידרגו את ההמלצה של האנימה שקיבלו (המסך הלבן הוא התיאור של האנימה נתונים כמו פרקים ואורך כל פרק וכמות לייקים ודיסלייקים וטריילרים).

מסך מעבר מסכים למשתמש



* מסך זה נועד עבור משתמשים כדי שיעברו בין מסכים באפליקציה כמו קטלוג אנימות, מסך המלצות, ומסך נתונים אישיים.
* כפתור Home מביא למסך המלצות
* כפתור CATALOG מביא לקטלוג אנימות
* כפתור PROFILE מביא לנתונים האישיים.

# מסך נתונים אישיים (זהה גם ליוצרים ומשתמשים)

* במסך זה מופיע הנתונים האישיים של המשתמש שם החיבה שלו שמו האמיתי המייל שלו ואפילו אפשרות לשינוי הסיסמא בעזרת סיסמא קודמת.
* כפתור Log out מביא למסך הראשי
* כפתור Your favorite Anime (יש רק למשתמשים) מביא למסך אנימות אהובות.

מסך מעבר מסכים ליוצר

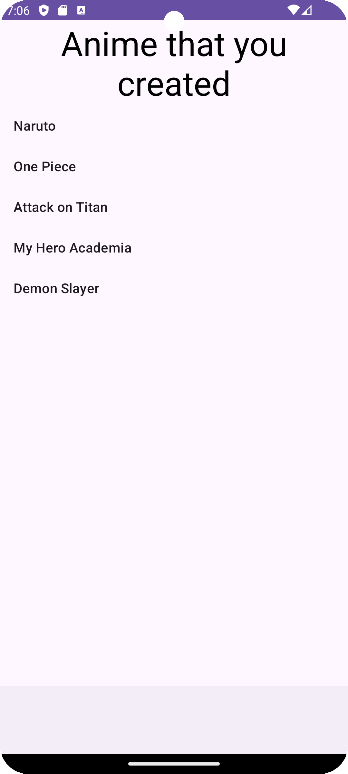


* מסך זה נועד עבור יוצרים כדי שיעברו בין מסכים באפליקציה כמו קטלוג אנימות, מסך המלצות, ומסך נתונים אישיים.
* כפתור Home מביא למסך יצירת אנימה.
* כפתור CATALOG מביא לקטלוג אנימות שיצר.
* כפתור PROFILE מביא לנתונים האישיים.



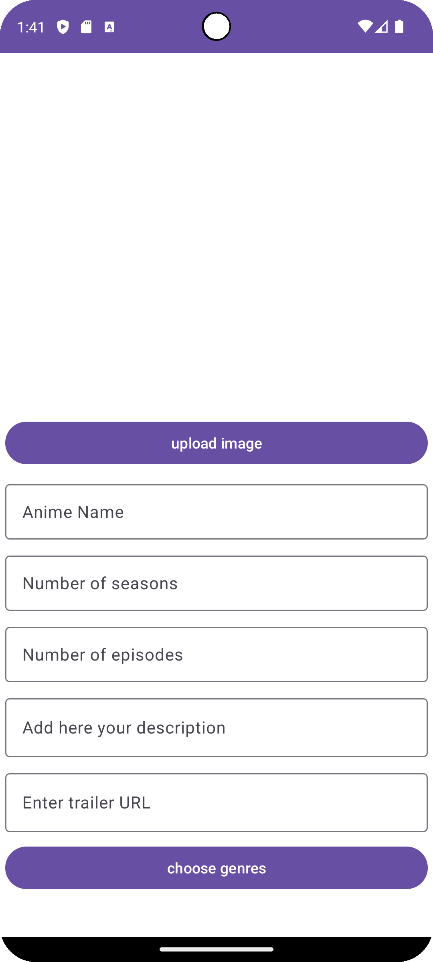
מסך קטלוג אנימות למשתמש

* מסך זה נועד למשתמש לעבור בין אנימות שונות במידה וירצה לדרגם.
* לחיצה על אנימה תכניס לאותה אנימה

מסך קטלוג אנימות ליוצר

* מסך זה נועד ליוצר שיוכל לערוך אנימות שיצר.
* לחיצה על אנימה תכניס למסך עריכה.

מסך יצירת אנימה

* מיעוד ליוצרים שיוכלו להוסיף אנימות חדשות למשתמשים רגילים שיוכלו לדרג ולגלות אנימות חדשות.
* כפתור Upload image מעביר לגלריה.
* כפתור Choose geners מביא לאוסף סוגי אנימות שעל היוצר לבחור.

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
מסך שינוי סיסמא

* מיעוד למשתמשים ויוצרים שרוצים לשנות את הסיסמא למשתמש שלהם.
* כפתור Change password מחזיר למסך הראשי.

# מסך עריכת אנימה

* במסך זה מופיע למנהלים על מנת שיערכו את האנימות שכבר יצרו במידה ונפלה טעות באחד מהנתונים בה.
* לחיצה על כפתור edit this anime מחזיר לתפריט של היוצר.

# 

מסך בחירת סוגי אנימות

* מיעוד ליוצר כשהוא יוצר אנימה עליו לבחור על איזה סוג של אנימה מדובר.
* כל כפתור מוסיף לשדה הסוג אנימה את הסוג.

מסכי משתמש רגיל

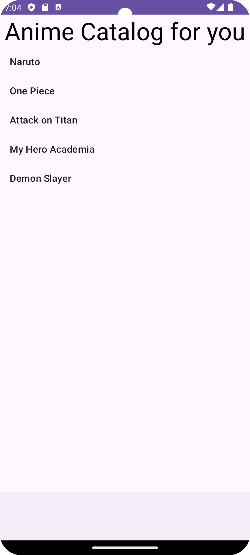








****

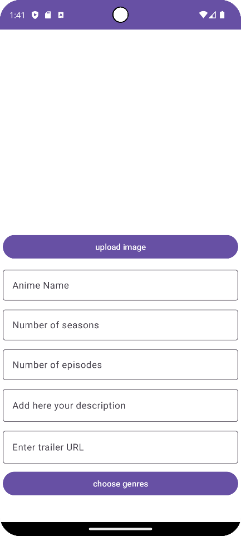


מסכי היוצר:



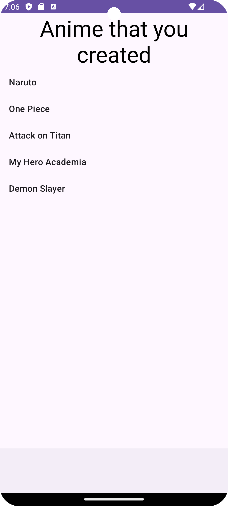


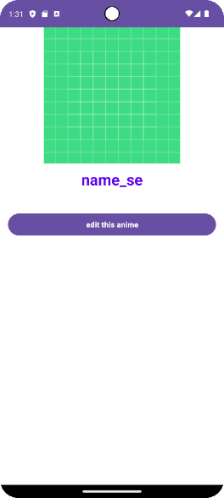












# פרק ה' - 'Animengine' - העיצוב

## ‪Databases | Python‬‏א. תיאור הארכיטקטורה של המערכת המוצעת Database (firebase)

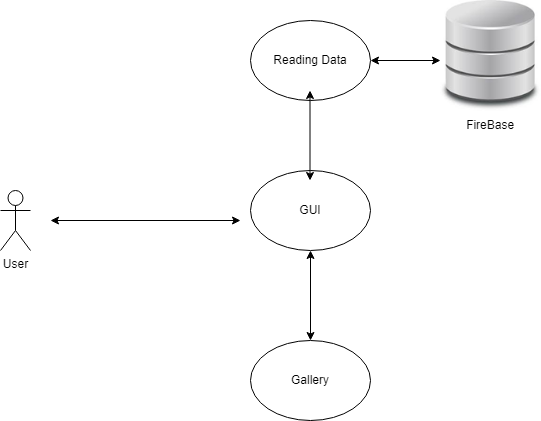
Server  
השרת שמתקשר עם ממסד הנתונים

## ‪User mobile phone - Ecommerce & Shopping Icons‬‏‪User mobile phone - Ecommerce & Shopping Icons‬‏‪User mobile phone - Ecommerce & Shopping Icons‬‏ משתמשים של האפליקציה

## ב. תיאור הטכנולוגיה הרלוונטית

* ממסד נתונים FireBase – בחרתי להשתמש בממסד הנתונים Firebase בגלל שאני יכול לשמור בו מידע ביתר קלות בגלל הממשק שלו עם אנדרואיד, בנוסף האפליקציה שלי לא דורשת שמירת יותר מידי נתונים ולכן אני יכול לא להשתמש בטבלת נתונים. ואישית יותר נוח לי לעבוד עם Firebase בגלל הממשק שלו.

## ג. תיאור מודולים בהם נעשה שימוש



* 1. פעולות ראשיות במחלקות (טענות כניסה ויציאה)

|  |  |  |
| --- | --- | --- |
| שם המחלקה User | | |
| תיאור המחלקה | המחלקה מייצגת את המשתמש |
| שם משתמש | String name |
| מייל | String mail |
| סיסמא | String password |
| שם חיבה | String nickname |
| האם יוצר | boolean creator |
| מזהה | String id |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| getName | אין | הפונקציה מחזירה את הערך של השדה name של המשתמש. |
| setName() | String Name | הפונקציה מעדכנת name את השדה |
| getMail() | אין | הפונקציה מחזירה את הערך של השדה .mail של המשתמש |
| setMail() | String Mail | הפונקציה מעדכנת .mail את השדה |
| getPassword() | אין | הפונקציה מחזירה את הערך של השדה .password |
| setPassword() | String password | הפונקציה מעדכנת passwordאת |
| getnickname() | אין | הפונקציה מחזירה את הערך של השדה .nickname |
| Setnickname() | String nickname | הפונקציה מעדכנת nicknameאת |
| getIscreator() | אין | הפונקציה מחזירה את הערך של השדה creator אם המשתמש הוא יוצר)) |
| Setcreator() | boolean creator | הפונקציה מעדכנת .creatorאת |
| getId() | אין | הפונקציה מחזירה את הערך של השדה .id של המשתמש |
| setId() | String id | הפונקציה מעדכנת .idאת |
| InsertUser() | אין | אובייקט המשתמש נוסף למסד הנתונים תחת הקשר עם המזהה שלו. |

|  |  |
| --- | --- |
| שם המחלקה | DB |
| תיאור המחלקה | המחלקה מייצגת את הפעולות שעושים על ממסד הנתונים |
| המשתנה שמחבר לממסד הנתונים | static FirebaseDatabase db |
| המשתנה עוסק ב SDK של האפליקציה ומשמשת לאימות משתמשים | static FirebaseAuth au |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| getAU() | אין | אם au הוא null, אז ייווצר אובייקט חדש של Firebase Authentication. אם כבר קיים אובייקט, יוחזר האובייקט הקיים. |
| getDB() | אין | אם dbהוא null, אז ייווצר חיבור חדש ל-Firebase Realtime Database. אם החיבור כבר קיים, יוחזר החיבור הקיים |
| SetUser() | User user | הפונקציה מגדירה אובייקט User בממסד נתונים תחת מפתח ה-id של המשתמש |
| setLikes() | Map<String, Integer> likes, String uid | הצומת Likes מעודכן במסד הנתונים עבור משתמש עם המזהה uid |
| setAnime() | Anime anime, InputStream inputStream | הפונקציה מוסיפה אנימה לרשימת האנימות גם ליוצר שיצר אותה וגם למשתמש רגיל. |
| getKey() | אין | הפונקציה מחזירה מפתח ייחודי חדש שנוצר עבור הצומת Anime במסד הנתונים של Firebase |
| getBytes() | InputStream inputStream | הפונקציה מחזירה את מערך הבתים שנוצר. |

|  |  |
| --- | --- |
| שם המחלקה | Anime |
| תיאור המחלקה | המחלקה Anime מייצגת אובייקט של סדרת אנימה במערכת. |
| String name | שם האנימה |
| long likes | מספר הלייקים שהאנימה קיבלה |
| long dislikes | מספר הדיסלייקים שהאנימה קיבלה |
| String description | תיאור האנימה |
| String creator\_id | מזהה היוצר שיצר את האנימה. |
| String anime\_id | מזהה ייחודי לאנימה |
| int episodes | מספר הפרקים באנימה |
| int seasons | מספר העונות באנימה |
| List<String> genres | רשימה של ז'אנרים שהאנימה משתייכת אליהם. |
| Creator<Anime> CREATOR | אובייקט המאפשר ליצור אובייקטים מסוג Anime מ-Parcel |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| upload\_anime() | InputStream input\_stream | הפעולה תעלה את פרטי האנימה יחד עם תמונה ל-Firebase Storage ותעדכן את הנתונים ב-Firebase Realtime Database. |
| getBytes() | InputStream input\_stream | הפעולה תקרא נתונים מתוך InputStream ותמיר אותם למערך בתים (byte[]). |
| describeContents() | אין | הפעולה מחזירה ערך שלם 0 המייצג שאין לאובייקט שדות מיוחדים למעבר ב-Parcel |
| writeToParcel() | Parcel parcel, int i | הפעולה כותבת את כל השדות של אובייקט ה-Anime לתוך אובייקט מסוג Parcel |

|  |  |
| --- | --- |
| שם מחלקה | AnimeComperator |
| תיאור המחלקה: | **AnimeComperator** היא מחלקה המממשת את הממשק Comparator ומספקת מנגנון השוואה מותאם אישית עבור אובייקטים מסוג Anime. |
| Map<String,Integer> likes | מייצגת מפת העדפות שמכילה את רמת הלייקים עבור כל ז'אנר (Genre) |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| setLikes() | Map<String, Integer> likes | השוואות עתידיות בין אנימות יתבצעו בהתבסס על המפה החדשה |

|  |  |
| --- | --- |
| שם המחלקה: | StorageConnection |
| תיאור המחלקה: | המחלקה מטפלת באינטראקציות עם  Firebase Storage  כמו העלאת והורדת קבצים/תמונות. |
| |  | | --- | |  |  |  | | --- | | StorageReference storageRef | | אובייקט הפניה לתיקיה ספציפית ב Firebase Storage. |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| requestFile () | String name, OnSuccessListener<byte[]> lambda | מוריד קובץ כמערך בתים (byte array)  Firebase Storage מ- ומפעיל את המאזין שהתקבל במקרה של הצלחה. |
| uploadImage() | String name, byte[] img | מעלה את מערך הבתים כתמונה ל-Firebase Storage, עם טיפול במקרים של הצלחה או כישלון בעזרת מאזינים. |

|  |  |
| --- | --- |
| שם המחלקה: | Favorites |
| תיאור המחלקה: | המחלקה מייצגת רשימת מועדפים של משתמש, הכוללת מזהה, שם חיבה ורשימת המועדפים. |
| favorites <List<String | רשימת המועדפים של המשתמש. |
| String nickname | שם החיבה של המשתמש. |
| id String | מזהה ייחודי של המשתמש. |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| ()InsertDB | אין | מוסיפה את אובייקט המועדפים למסד הנתונים תחת הצומת  Favorites  כשהמפתח הוא המזהה של המשתמש. |

|  |  |
| --- | --- |
| שם המחלקה: | EngineController |
| תיאור המחלקה: | המחלקה מנהלת את האנימות, העדפות המשתמש (מועדפים ודיסלייקים), ואת התקשורת עם Firebase. |
| Map<String, Integer> likes | מפה לשמירת מספר הלייקים לכל אנימה |
| ArrayList<String> favourites | רשימת האנימות המועדפות של המשתמש |
| ArrayList<String> disliked | רשימת האנימות שהמשתמש סימן כדיסלייק |
| ArrayList<Anime> disliked\_anime | רשימת אובייקטי אנימה שהמשתמש סימן כדיסלייק |
| PriorityQueue<Anime> queue | תור עדיפויות לניהול אנימות בהתאם להעדפות המשתמש |
| AnimeComperator comparator | קומפרטור למיון אנימות לפי לייקים |
| Anime current\_anime | האנימה הנוכחית |
| MutableLiveData<Anime> shared\_anime | אובייקט LiveData לשיתוף האנימה הנוכחית עם ה-UI |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| ()listen\_to\_anime | אין | מאזינה לשינויים בנתוני האנימות ומעדכנת את התור בהתאם להעדפות המשתמש. |
| ()contain | ArrayList<Anime> arr, String id | בודקת אם רשימת האנימות מכילה אנימה מסוימת לפי מזהה ומחזירה true או false. |
| ()getCurrent\_anime | אין | מחזירה את האנימה הנוכחית או את האנימה הבאה לפי סדר העדיפויות. |
| ()like\_set | boolean isLike | מעדכנת את העדפת המשתמש לאנימה הנוכחית (לייק/דיסלייק), כולל עדכון ב-Firebase |

|  |  |
| --- | --- |
| מחלקה: | SearchListAdapter |
| תיאור המחלקה: | המחלקה אחראית להציג רשימת אנימות (Anime) ברשימה מתגלגלת (RecyclerView) ומאפשרת תגובה לאירועים כמו לחיצות. |
| ArrayList<Anime> mList | רשימת האובייקטים מסוג Anime המוצגת ב-RecyclerView. |
| Context context | הקשר (Context) של הפעילות או הפרגמנט שמשתמשים באדפטר זה. |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| onInterceptTouchEvent() | RecyclerView rv, MotionEvent e | יוצר ומחזיר ViewHolder חדש שמכיל את מבנה התצוגה עבור פריט ברשימה. |
| onBindViewHolder() | MyViewHolder holder, int position | קושר את הנתונים לפריט המתאים ברשימה (לדוגמה, שם ותמונה). |

|  |  |
| --- | --- |
| שם המחלקה: | Server\_logger |
| תיאור המחלקה: | מחלקה זו אחראית על תהליכי התחברות של משתמשים דרך שרת HTTP ודרך Firebase. היא מספקת פונקציות לניהול התחברות, שליחת סיסמאות לאיפוס, ותמיכה בתקשורת עם השרת. |
| FirebaseAuth myAuth | מופע של Firebase Auth לביצוע אימות והתחברות משתמשים. |

|  |  |  |
| --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה |
| ()log\_in\_with\_server | String email, String pass, Context context | שולח בקשת POST לשרת עם המייל והסיסמה ומבצע אימות. אם ההתחברות מצליחה, המשתמש מחובר דרך Firebase, והנתונים מעודכנים ב-Firebase Realtime Database. |
| ()send\_reset\_password\_mail | String email, Context context | שולח קישור לאיפוס סיסמה למייל שהוזן. במקרה של הצלחה, מציג הודעה למשתמש ומפנה אותו חזרה לעמוד ההתחברות. |
| doInBackground() | Object[] objects | מבצע קריאה לפונקציה log\_in\_with\_server בתהליך רקע. מחזיר true אם הפעולה הצליחה או false אם היא נכשלה. |

|  |  |
| --- | --- |
| שם המחלקה: | Server\_signer |
| תיאור המחלקה: | מחלקה זו אחראית על רישום משתמשים דרך שרת HTTP ו-Firebase. היא כוללת פונקציות לרישום משתמש חדש ולניהול תקשורת עם השרת והמסד הנתונים. |
| String[] gen | מערך המכיל את רשימת הז'אנרים (Genres) המוגדרים מראש, אשר משמשים לאיפוס או הגדרה של העדפות משתמש במסד הנתונים. |

|  |  |  |
| --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה |
| sign\_up\_with\_server() | String uname, String email, String pass, String nick, boolean isCreator, Context context | שולח בקשת POST לשרת לרישום משתמש חדש. במקרה של הצלחה, המשתמש נרשם ל-Firebase, פרטי המשתמש נשמרים ב-Firebase Realtime Database, ונוצר חיבור למערכת עם הגדרות מתאימות (יוצר או משתמש רגיל). |
| doInBackground() | Object[] objects | מבצע קריאה לפונקציה sign\_up\_with\_server() בתהליך רקע. מחזיר true אם הפעולה הצליחה, אחרת false. |

|  |  |
| --- | --- |
| שם המחלקה: | MenuFragment |
| תיאור המחלקה: | מחלקה זו אחראית על ניהול תפריט הניווט באפליקציה, כולל מעבר בין עמודים שונים על פי לחיצות המשתמש. היא מטפלת גם בהבדל בין משתמש רגיל ליוצר. |
| String ARG\_PARAM1 | מחרוזת המכילה את שם הפרמטר הראשון שעובר ל-Fragment באמצעות Bundle. |
| String ARG\_PARAM2 | מחרוזת המכילה את שם הפרמטר השני שעובר ל-Fragment באמצעות Bundle. |
| LinearLayout mHome | אלמנט גרפי שמייצג את כפתור הבית בתפריט. |
| LinearLayout mCatalog | אלמנט גרפי שמייצג את כפתור הקטלוג בתפריט. |
| LinearLayout mProfile | אלמנט גרפי שמייצג את כפתור הפרופיל בתפריט. |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| newInstance() | String param1, String param2 | יוצר מופע חדש של MenuFragment עם פרמטרים שמועברים באמצעות Bundle. |
| onCreateView() | LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState | יוצר את התצוגה של התפריט ומחבר את האלמנטים הגרפיים לפעולות שלהם (Listeners). |

|  |  |
| --- | --- |
| שם המחלקה: | TrailerService |
| תיאור המחלקה: | מחלקה זו משתמשת בשירות (Service) לניהול והשמעת טריילרים של אנימה. המחלקה אחראית על גישה למידע על הטריילר והשמעתו ברקע תוך שימוש במנגנוני Android Service. |
| String trailer\_id | מזהה ייחודי לטריילר. |
| String video\_url | הקישור לסרטון הטריילר (YouTube). |
| boolean isPlaying | מציין האם הטריילר מנוגן כרגע. |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| startTrailerService() | String trailer\_id, String video\_url | מפעיל את ה-Service עם מזהה הטריילר וכתובת הסרטון. מתחיל לנגן את הטריילר ברקע. |
| stopTrailerService() | אין | מפסיק את ה-Service ואת השמעת הטריילר. |
| isServiceRunning() | אין | בודק אם ה-Service פעיל ומחזיר ערך בוליאני בהתאם. |
| bindService() | אין | מבצע חיבור ל-Service כדי לאפשר שליטה מרחוק על הטריילר (לדוגמה: עצירה, השמעה מחדש). |
| getTrailerStatus() | אין | מחזיר מידע אם הטריילר מתנגן כרגע או עצור (isPlaying). |

|  |  |
| --- | --- |
| שם המחלקה: | Trailer |
| תיאור המחלקה: | מחלקה זו משתמשת בשירות (Service) לניהול והשמעת טריילרים של אנימה. המחלקה אחראית על גישה למידע על הטריילר והשמעתו ברקע תוך שימוש במנגנוני Android Service. |
| String trailerId | מזהה ייחודי לטריילר, לרוב מספר או מזהה ייחודי ממסד הנתונים. |
| String videoUrl | כתובת ה-URL של הסרטון (YouTube). |
| boolean isPlaying | מציין האם הטריילר מנוגן כרגע. |

|  |  |  |
| --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה |
| stopTrailerService() | אין | מפסיק את השירות ואת השמעת הטריילר ברקע. |
| isServiceRunning() | אין | מחזיר ערך בוליאני המצביע אם ה-Service פעיל או לא. |
| bindService() | אין | מבצע חיבור ל-Service כדי לאפשר שליטה מרחוק על הטריילר (לדוגמה: עצירה, הפעלה מחדש). |

|  |  |
| --- | --- |
| שם המחלקה: | TTSManager |
| תיאור: | מחלקה זו אחראית על ניהול המרת טקסט לדיבור (Text-To-Speech) בתוך האפליקציה. המחלקה תומכת בשפות שונות, התאמת מהירות הדיבור, והפסקת הדיבור בעת הצורך. |
| TextToSpeech tts | מופע של TextToSpeech המנהל את פעולות ההמרה מטקסט לדיבור. |
| Locale language | השפה בה ה-TTS ידבר (ברירת מחדל: אנגלית). |
| Float pitch | גובה הצליל (Pitch) של הקול (ברירת מחדל: 1.0). |
| Float speechRate | מהירות הדיבור (Speech Rate), ערכים גבוהים יותר יגרמו לדיבור מהיר יותר. |
| Boolean isInitialized | מציין האם ה-TTS התחיל לעבוד בהצלחה. |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | טענת כניסה | טענת יציאה |
| initializeTTS() | Context context | מאתחל את ה-TTS עם ההגדרות הנוכחיות ומוודא שהוא מוכן לשימוש. |
| Speak() | String text | קורא את הטקסט בקול רם בהתאם להגדרות הקול והמהירות. |
| stopSpeaking() | אין | עוצר את הדיבור הנוכחי באופן מיידי. |
| shutdownTTS() | אין | משחרר את משאבי ה-TTS כאשר אין בו צורך. |

* 1. Threading – שם ה thread ומה מבצע (וסנכרון עם שאר במערכת):

שם הAnimeProcessorThread - thread :

* תיאור : ה Thread - AnimeProcessorThread אחראי על עיבוד לייקים, דיסלייקים, ועדכון תור האנימות (PriorityQueue) בזמן אמת, מבלי לשבש את חוויית המשתמש בממשק. הוא גם מתעדכן בבסיס הנתונים של Firebase ומשתמש בנתונים אלו לעדכון המידע המוצג למשתמש.
* סנכרון עם המערכת : עדכון PriorityQueue: עיבוד הנתונים והוספה לתור האנימות מתבצע באופן עצמאי ב-Thread. עם זאת, כדי למנוע מצב של קריאה וכתיבה בו-זמנית, נעשה שימוש באובייקט מסונכרן או lock. לדוגמה:

שם ה :TrailerPlaybackThread – thread

* תיאור: ה Thread - TrailerPlaybackThread משמש להשמעת טריילרים ברקע באמצעות TrailerService. הוא טוען את הטריילר מהשרת או מהקובץ המקומי ומבצע השמעה ברקע, תוך עדכון סטטוס ההשמעה (נגן/עצור).
* סנכרון עם המערכת: משתמש בממשק Callback כדי לעדכן את המשתמש אם יש שגיאה בהשמעה או אם הסרטון הסתיים.

## ד. תרשים UML (להוסיף רק אחרי שהפרויקט מוכן להגשה!!!)

## ה. סיכום אבני היסוד כפי שבאות לידי ביטוי בפרויקט

1. **activities** – פעילות היא מחלקה המייצגת מסך אחד ביישום וכל יישום יכול להכיל כמה פעילויות. לדוגמא: האקטיביטי SignUpActivity מאפשר למשתמשים להירשם לאפליקציה על ידי הזנת פרטים כמו שם, כתובת אימייל וסיסמה, ולאחר מכן שומר את הנתונים בבסיס הנתונים של Firebase.
2. **intents** – מסרים משמשים כדי לעבור מפעילות (מסך אקטיביטי) אחד לפעילות אחרת, יש שני סוגים של מסרים מפורשים ומרומזים. במסר מפורש מציינים באופן מפורש לאיזה פעילות רוצים לעבור ומאיזה פעילות עוברים. במסר מרומז לא מציינים במפורש לאיזה פעילות רוצים לעבור אלא את שצריך להתבצע. לדוגמא : בעת ההרשמה באקטיביטי SignUpActivity, נשלח Intent מפורש לאקטיביטי SelectActivity, שבו המשתמש בוחר באנימות המועדפות עליו.
3. **בסיס נתונים** – ישנם סוגים שונים של בסיסי נתונים שונים כמו למשל,

SQLite, Firestore, Firebase SQLserver ועוד. אני בחרתי להשתמש ב Firebase שהוא בסיס נתונים אשר שומר את הנתונים בתוך קבצי JSON ולכן הוא גם בסיס נתונים מרוחק שעובד על שרת מה שמאפשר ריבוי משתמשים באפליקציה. לדוגמא: האנימה נשמרת בבסיס הנתונים של Firebase בפורמט JSON תחת הצומת **Anime**, עם מזהה ייחודי, שם, תיאור, ומספר פרקים

1. **Service** – **שירות (Service)** הוא פעולה או מערכת המציעה פתרון או פונקציה מסוימת למשתמשים או לאפליקציות. שירותים יכולים לכלול פעולות כמו שליחה וקבלה של נתונים, תמיכה טכנית, או ביצוע חישובים. לדוגמא : TrailerService מנהל השמעת טריילר של אנימה ברקע תוך שמירה על ביצועי האפליקציה.
2. **Web Services** – הורדת נתוני מידע(לא תמונות/ מוזיקה) מהאינטרנט תוך שימוש ב- API של אתר. שימוש בנתונים שהורדו. דוגמה: שווי מטבע חוץ, רשימת בתי עסק באזור... לתת דוגמה מהפרויקט.
3. **AcvtivityForResult עם launcher** – שימוש ב-ActivityForResult עם Activity Result Launcher הוא המנגנון המודרני (שלאחר API 30) לשיתוף פעולה בין Activities באפליקציה. במקום המנגנון הישן (שכולל startActivityForResult), כיום משתמשים ב-ActivityResultLauncher, שהוא דרך פשוטה, מאובטחת וגמישה יותר לטפל בתוצאות שמגיעות מ-Activity אחרת.

לדוגמא : בעת העלאת תמונה ב-**CreateActivity**, משתמשים ב-ActivityResultLauncher כדי לקבל את תמונת האנימה שהמשתמש בחר בגלריה.

1. **אפליקציית רב משתתפים –** אפליקציה שמאפשרת למספר משתמשים לשתף פעולה בזמן אמת, לדוגמה, משחקים מקוונים או צ'אט קבוצתי. לדוגמא : משתמשים יכולים לעדכן לייקים או דיסלייקים על אנימות בזמן אמת, ומידע זה מסונכרן בין משתמשים שונים בעזרת Firebase Realtime Database.
2. **- Fragment** רכיב UI עצמאי שמאפשר חלוקה של האקטיביטי למסכים קטנים. דוגמה: רכיב בתצוגת אנימציה או רשימה שנמצא בתוך אקטיביטי, ומנהל את הצגת הנתונים.לדוגמא: **MenuFragment** מנהל תפריט ניווט שמפנה את המשתמש למסכים כמו דף הבית (Engine), קטלוג האנימות (SearchActivity), או עמוד פרופיל המשתמש (UserProfileActivity).
3. **Threads**  - תהליכים נפרדים שמתבצעים במקביל לפעולה הראשית, לדוגמה, חישוב בזמן אמת או עבודה עם קבצים/background processing. לדוגמא : מחלקת **EngineController** משתמשת ב-Thread לחישוב לייקים בזמן אמת ולהשוואת אנימות על פי העדפות המשתמש.
4. **SharedPreferences** - מה זה? למה השתמשתם בפרויקט?
5. **Async Task**  - משימה אסינכרונית שמבצעת פעולות ברקע (כמו קריאת נתונים או חיבור לשרת) מבלי להפריע לממשק המשתמש. לדוגמא : Server\_logger משתמש ב-AsyncTask כדי לשלוח בקשת התחברות לשרת HTTP ברקע מבלי לשבש את חוויית המשתמש באפליקציה.
6. **TextToSpeech** - מאפשר להפוך טקסט לדיבור. לדוגמא: בעת צפייה בפרטי האנימה, מנוע TTS מקריא את תיאור האנימה למשתמש בעזרת מחלקת TTSManager.

* להוסיף קוד לדוגמה בכל סעיף (לפי מה שכתבתם שיש אצלכם בפרויקט) – אחרי שלב הקידוד! אסור קוד על רקע שחור

## תיאור סביבת הפיתוח

* שפת התכנות שנבחרה לפיתוח הפרויקט היא Java 1.8
* Android studio גרסת Koala 2024.1.1
* פירוט כלי הפיתוח הנדרשים לפיתוח ופירוט הסביבה והכלים הנדרשים לבדיקות:  
  Android studio עם API לפחות 33, באמולטור תהיה את האפליקציה, שרת Firebase.

## תיאור האלגוריתמים המרכזיים בפרויקט (לבחור 2-3):

* **הבעיות האלגוריתמיות בפרוייקט:**
* 1. איך לשמור טריילרים בממסד הנתונים הבעיה היא כיצד לאחסן מידע על טריילרים במבנה היררכי ונגיש בממסד הנתונים של Firebase, כולל קישורים לטריילרים.

1. איך לנגן את הטריילר האתגר הוא לנהל השמעה של סרטוני טריילרים בצורה חלקה באפליקציה, תוך שימוש במנגנוני Android, כמו Service, ותוך שמירה על ביצועי האפליקציה.   
   3. איך למנוע ממשתמשים להיכנס לאפליקציה כיוצרים נדרשת בדיקה ובקרה על סוג המשתמש (משתמש רגיל או יוצר), כך שרק משתמשים מאומתים ומורשים יוכלו לגשת לתכונות המיועדות ליוצרים.הפנייה למקורות רלוונטיים.

* **תיאור אלגוריתמים קיימים לפתרון הבעיות:**
  + - 1. **שמירת טריילרים במסד הנתונים**

**תיאור הבעיה:** יש צורך לאחסן מידע על טריילרים של אנימה, כולל קישורים לסרטונים ותכונות נוספות, באופן היררכי ונגיש בממסד הנתונים של Firebase.   
**הפתרון האלגוריתמי**: השיטה שנבחרה היא שימוש במבנה היררכי בפורמט JSON שבו לכל אנימה יש תת-צומת ייעודי בשם "טריילרים". כל טריילר נשמר כעצם (Object) עם מזהה ייחודי, קישור לסרטון ותכונות נוספות. יתרונות השיטה: מבנה הנתונים קל לניהול ולהבנה. ניתן לשלוף ולעדכן נתונים בקלות בעזרת Firebase SDK. מאפשר הרחבה ללא שינויים גדולים במבנה.   
**חסרונות השיטה:** ביצועים עשויים להיפגע כאשר יש ריבוי טריילרים או שאילתות מורכבות. ייתכנו אתגרים בניהול נתונים במקרה של קשרים מורכבים בין האובייקטים. דרכים חלופיות: טבלה שטוחה: שמירת טריילרים כישויות נפרדות עם מזהה של האנימה. יתרון: מאפשר שאילתות גמישות. חיסרון: מצריך שאילתות מורכבות יותר כדי לשחזר את ההקשר. קישורים הדדיים: אחסון מזהים מקושרים בין אובייקטים. יתרון: מאפשר גמישות רבה יותר בקשרים. חיסרון: מסובך למימוש ולעדכון.

* + - 1. **ניגון טריילר באפליקציה**  
          **תיאור הבעיה:** יש לנהל את ניגון הטריילר בצורה חלקה תוך שמירה על ביצועי האפליקציה ושימוש במנגנוני Android כמו Service.  
          **הפתרון האלגוריתמי:** שימוש ב-Service שמיועד לניגון הטריילרים. השירות נטען ברקע ומתחיל לנגן את הסרטון באמצעות MediaPlayer או YouTube Player API. השירות נשלט על ידי האפליקציה ויכול לעצור, להתחיל מחדש, או לבצע השהיה של הניגון. יתרונות השיטה: מפריד את פעולת ניגון הטריילר משאר האפליקציה. מאפשר לנגן את הטריילר ברקע גם אם המשתמש עובר בין מסכים. מאפשר חיסכון במשאבים על ידי עצירת השירות כאשר הוא אינו בשימוש.   
         **חסרונות השיטה:** מצריך התעסקות רבה יותר בניהול חיי השירות (Lifecycle). ייתכן עיכוב קל בניגון בשל יצירת ה-Service. דרכים חלופיות: AsyncTask או Thread: ניגון הטריילר בתהליך רקע ייעודי. יתרון: פשוט יחסית למימוש. חיסרון: לא אידיאלי לניהול מתמשך כמו ניגון ברקע. BroadcastReceiver: שימוש בשידורים לקבלת פקודות לניגון, הפסקה ועצירה. יתרון: מאפשר תקשורת בין חלקי האפליקציה. חיסרון: מסובך יותר ממימוש שירות.

3. **מניעת כניסה של משתמשים כיוצרים לא מורשים.** **תיאור הבעיה:** יש לוודא שמשתמשים רגילים לא יוכלו לגשת לפונקציות המיועדות ליוצרים בלבד. הפתרון האלגוריתמי: שימוש בשדה displayName של Firebase Authentication. כאשר משתמש נרשם, שדה זה מעודכן ל-"creator" או "user". כל פעם שהמשתמש מתחבר, האפליקציה בודקת את הערך ומאפשרת גישה בהתאם לתכונות המותרות לסוג המשתמש.

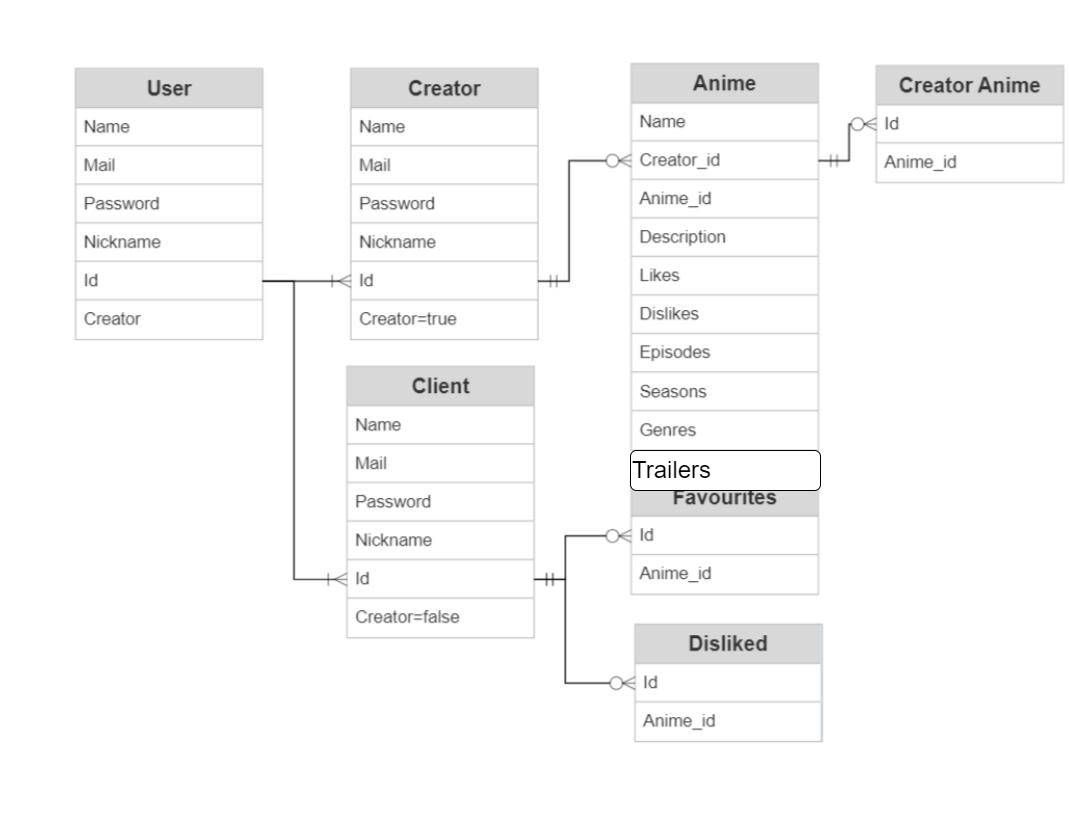
**יתרונות השיטה:** פתרון פשוט לשימוש בשדה קיים במערכת Firebase Authentication. מאפשר בקרה מדויקת על הרשאות המשתמש. מתממשק בקלות עם נתוני Firebase Realtime Database. חסרונות השיטה: אם השדה אינו מוגדר כראוי בזמן ההרשמה, המשתמש עלול לקבל הרשאות לא נכונות. מחייב תחזוקה שוטפת של השדה במקרים של שינויים במשתמשים. דרכים חלופיות: מנגנון הרשאות מותאם אישית: שמירת הרשאות במסד הנתונים לפי מזהה המשתמש. יתרון: גמישות רבה יותר במימוש הרשאות.  **חיסרון:** מצריך שאילתות נוספות לבדיקת הרשאות בכל פעם שהמשתמש מבצע פעולה. Token-based Authorization: יצירת טוקן גישה מותאם אישית לכל משתמש. יתרון: מאפשר רמות אבטחה גבוהות יותר. חיסרון: מורכב יותר ליישום ולעדכון.

להוסיף קוד לכל אלגוריתם אחרי שלב הקידוד  
אסור קוד על רקע שחור

## תיאור מבני הנתונים:

**בסיס נתונים:**

* **Firebase** תאור ההיררכיה של עץ הנתונים:



## אתגרים בפרויקט:

1. ללמוד איך לנגן ולהציג טריילרים מהאינטרנט באמצעות קישור.
2. ללמוד איך להשתמש ב Firebase ולשמור ולשלוף ממנו נתונים.
3. שימוש בצד שרת.

# פרק ו' - 'Animengine' - בדיקות

פירוט הבדיקות :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **שם הבדיקה** | **מטרת הבדיקה** | **מה נדרש לבצע** | **מתי** | **מה בוצע בפועל** |
| יצירת משתמש | אמורה לוודא שכאשר יוצרים משתמש הוא נשמר בבסיס הנתונים | הבדיקה מתבצעת על ידי לחיצת כפתור "צור משתמש" ומילוי הפרטים שמספיקים על מנת ליצור משתמש. לאחר מכן נכנסים מחדש לאפליקציה ומנסים להתחבר למשתמש. | 16.11 | המשתמש שיצרתי נשמר בבסיס הנתונים. |
| יצירת משתמש |  |  |  |  |
| מעבר בין סדרה לסדרה | אמורה לוודא כי האפליקציה מביאה למשתמש סדרות על פי לוגיקה. | יש להיכנס למשתמש רגיל ולדרג אנימות, במידה ואותה האנימה חוזרת על עצמה מספר פעמים הדבר שגוי וצריך לתקן זאת מיידית. | 11.1 | לאחר כתיבה ארוכה הצלחתי לגרום לכך שאנימה לא חוזרת ברצף לאותו המשתמש. |
| אפשרות של משתמש יוצר | בדיקה האם משתמש יוצר יכול ליצור אנימה, לעלות את כל מה שקשור לאנימה ועוד. בנוסף לבדוק האם האנימה נשמרת בבסיס הנתונים. | יש להיכנס למשתמש יוצר ולנסות להעלות סדרה, אם ההעלת הסדרה יש להיכנס למשתמש רגיל על מנת לראות כי בקטלוג האנימות התווספה האנימה הנוכחית. | 20.1 | לאחר שיצרתי את הסדרה היא נוספה כמו שצריך. |
|  |  |  |  |  |
|  |  |  |  |  |

* פירוט הבדיקות שהופיעו בפרק האפיון , **חייבים לבצע כל בדיקה מינימום פעמיים ולדווח בטבלה את תוצאות הבדיקה**
* יש לתעד כל בדיקה שמבוצעת, במידה וחוזרים על בדיקה – כל חזרה תתועד בשורה נפרדת עם תוצאות הבדיקה

# פרק ח' - 'שם המערכת' – מדריך למשתמש

## מדריך למשתמש

במידה ויש יותר מסוג משתמש אחד (לדוגמא מנהל מערכת ומשתמש קצה) יש לבנות לכל אחד מהם מדריך:

## מדריך למפתח

גירסאות, התקנות נדרשות לספריות (לדוגמה: JTDS,OpenCV), ספריות שצריך לעשות להן implement בגראדל.

**עץ הקבצים של הפרויקט:**

# פרק ט' - 'שם המערכת' – מבט אישי

## מבט אישי על העבודה ועל תהליך פיתוחה:

* אתגרים שהיו לי אורך הפרויקט
* אירועים מעניינים שקרו במהלך הפיתוח
* התמודדות עם קשיים
* הערכת הפתרון לעומת התכנון והמלצות לשיפורו

**יש להשלים לפחות לעמוד אחד !!!**

# פרק י' - 'שם המערכת' – ביבליוגרפיה

1. קישורים למקורות באינטרנט
2. קישורים לספרות מקצועית ספציפית לנושא העבודה (רשימת ספרים, מאמרים והפנייה לכתובות אתרים המכילים חומר רקע ששימש לפיתוח העבודה)

**לפחות 10 קישורים !!!**

# פרק יא' – נספח: תהליכי חקר בפרויקט (למי שיש!).

תיאור מחקרים שבוצעו בפרויקט לטובת פתרונות שונים.

# פרק יב' – נספח: הקוד המלא.

כאן להוסיף את כל המחלקות שלכם (רק JAVA, לא כולל קבצי XML ואחרים). מעל כל קובץ – שם הקובץ.