תכנות מערכות מתקדם - תרגיל מספר 4

בתרגיל זה שני חלקים

<u>חלק א</u>

בחלק זה עליכם להמשיך להשתמש בפרוייקט ה JIRA מהתרגילים הקודמים, ולתכנן בו את חלק ב של תרגיל זה.

תהליך הפיתוח של חלק ב יהיה בצמוד לפרוייקט הJIRA, ואתם חייבים להשתמש ב JIRA **<u>לאורך כל הדרך בזמן אמת</u>**. הדבר ייבדק על ידי הבודק.

אם ברצונכם להמשיך להשתמש ב Github מהתרגילים הקודמים אתם יכולים לעשות זאת. במקרה זה עליכם ליצור branch נפרד עבור תרגילים קודמים ולא לגעת בו (ב branch הנפרד) - כדי שהבודק יוכל לבדוק אותו.

ואז אתם ממשיכים לעבוד על ה main branch בשביל תרגיל 4.

לחילופין, אתם יכולים לא לגעת ב main branch וליצור branch חדש שיהיה ה main branch עבור תרגיל 4.

בכל מקרה, שימו לב לא לגעת בbranch שמיועד לתרגילים הקודמים לאחר מכן - כדי לא לפגוע בימי ה"חסד".

אם אתם רוצים, אתם יכולים ליצור repository נפרד עבור תרגיל 4.

אם החלפתם קבוצה ואין באפשרותיכם להשתמש בעקבות כך בפרויקט ה JIRA (או Github וכו') מהתרגילים הקודמים, צרו פרוייקטים חדשים בהתאם עם הקוד מהתרגילים הקודמים.

בפרט, עליכם להקפיד על:

- חלוקה ל epics
- כתיבת users stories
- כתיבת tasks
- יצירת ספרינט, מינוי scrum master מבין חברי הקבוצה, תכנון הספרינט, קיום מפגשי סטטוס לאורך הספרינט (לא כל יום, אבל לפחות פעמיים בשבוע) חובה לתעד ב JIRA את כל הפעולות הללו ואת התוצאות שלהן. למשל, התובנות והמסקנות של מפגשי הסטטוס חייבים להיות מפורטים ומתועדים.
- עדכון סטטוס משימות (in progress, code review, done) כולל עדכון של הactivity של משימות **<u>תוך כדי</u>** ביצוע המשימה. שימו לב, עליכם להוסיף את הסטטוס code review, הוא לא מגיע ברירת מחדל. משימה עוברת ל code-review כאשר סיימתם אותה ופתחתם לה pull request. באפשרותכם כעת לשייך את המשימה ב JIRA אל חבר צוות אחר שצריך לעשות review לקוד שלכם. לחילופין, תוכלו להשאיר את המשימה אצלכם ול"תייג" את החבר צוות האחר ב activity של הissue ב JIRA. בכל מקרה, עליכם להגדיר בצורה ברורה מי לבסוף מעביר את המשימה ל done.
- שיוך משימות לסטודנט שמבצע אותן **<u>לפני שהוא מתחיל לעבוד עליהן</u>**.
- עליכם לזהות אילו משימות "תוקעות" משימות אחרות (is blocked by) ולהביא את זה לידי ביטוי בפיצ'ר הרלוונטי בJIRA.

- שיוך **<u>כל</u>** המשימות ל feature branches. כלומר, יש ל Github ול JIRA התממשקות כך שאם רושמים את שם הJIRA issue בשם של ה branch והמשימות, JIRA יודע להציג את זה בJIRA issue אוטו׳.

<u>חלק ב</u>

תרגיל זה הינו אבן הדרך האחרונה של הפרויקט, בה נשזור ביחד את כל רכיבי הפרויקט למערכת אחת שלמה העובדת ביחד.

עליכם לכתוב אפליקציית Web בעזרת HTML, CSS, JS ו React, **<u>וגם</u>** אפליקציית Android.
האפליקציות יציגו מידע אמיתי ודינמי, יתמכו במשתמשים שונים וידברו מול שרת.
המטרה של האפליקציות להיות הכי קרוב שאפשר לאפליקציות שירותי צפיה בסרטים אמיתיות (דמוי נטפליקס).

ההוראות הבאות מפרטות את הדרישות **<u>משתי האפליקציות</u>**.

כעת אפרט את המסכים והפונקציונאליות המתבקשת.

מבחינה עיצובית - הנראות הויזואלית של המסכים צריכה להיות **בהשראת** אפליקציית נטפליקס האמיתית.

<u>יש לייצר את המסכים הבאים:</u>

- מסך הבית למשתמשים לא מחוברים.
- מסך התחברות.
- מסך הרשמה (אצלנו מספיק שהוא יהיה דומה למסך ההתחברות רק עם שדות מתאימים. כלומר שנרשמים צריך לתת נתונים שלא נותנים בעת התחברות, כמו שם, תמונה וכו׳).
- מסך ראשי לאחר התחברות (המסך בו רואים את הסרטים שניתן לצפות בהם).
- מסך מידע על סרט (אפשר שיהיה מסך נפרד ואפשר שזה יהיה ״חלון״ על גבי המסך הראשי). שימו לב, במסך זה יהיה המידע המלא על הסרט וגם את רשימת הסרטים המומלצים (שהשרת מתרגיל 2 ימליץ עליהם)
- מסך צפייה בסרט (נגן וידאו).
- מסך תוצאות חיפוש.
- מסך ניהול.

בעת לחיצה על כפתור ההתחברות בעמוד הראשי למשתמשים לא מחוברים, יוצג **מסך התחברות** פשוט, בו יש להזין שם משתמש וסיסמא או מעבר למסך ההרשמה.
במסך ההרשמה נקבל מקום להזין שם משתמש, סיסמא, וידוא סיסמא, שם לתצוגה, תמונה.
מבחינת נראות, אנא הסתכלו על מסכי ההתחברות של נטפליקס (או שירות דומה).
אין להתעסק עם חיבור לחשבון גוגל - אנחנו מנהלים את המשתמשים בעצמנו בפרויקט.

מסך ראשי לאחר התחברות יכלול את האלמנטים הויזואלים הבאים:

- תפריט עליון.
- נגן וידאו עליון שמנוגן אוטומטית. יש להציג סרט אקראי מתוך רשימת הסרטים שחוזרים ב: http://foo.com/api/movies
- סרטים לפי קטגוריות. יש להציג את הסרטים והקטגוריות, לפי מה שחוזר מ: http://foo.com/api/movies

לחיצה על סרט תפתח את "מסך מידע על סרט" ותציג את המידע על הסרט כולל ההמלצות.
לחיצה על צפיה בסרט תפתח את מסך "מסך צפייה בסרט (נגן וידאו)".

בתפריט העליון ניתן יהיה לראות את עמוד הבית שלאחר ההתחברות (כפי שהוגדר לעיל), ניתן יהיה לצפות בסרטים לפי קטגוריות, ניתן יהיה לבצע חיפוש ולראות תוצאות חיפוש מתאימות.
רמז: אומנם המשפט הקודם מתאר לכאורה שלושה מסכים שונים, אבל בפועל, ברגע שמימשתם מסך אחד - מימשתם כמעט את שלושתם. וודאו שאתם מבינים את הרמז הזה.

בנוסף, תהיה אפשרות להתנתק בתפריט העליון, ואם משתמש המחובר הינו משתמש מנהל, הוא יוכל דרך התפריט העליון גם לעבור למסך הניהול.
במסך הניהול ניתן יהיה:

- להוסיף/לערוך/למחוק קטגוריות.
- להוסיף/לערוך/למחוק סרטים.

כמו כן, יש לממש כפתור בתפריט העליון של מסך הבית לאחר התחברות אשר משנה את ה theme של האפליקציה מdark mode ל light mode ולהפך. כלומר, בעת שינוי ההגדרה, הצבעים של האפליקציה ישתנו.

התחברות והרשמה:

- חלק מהפעולות מול השרת יכולות להתבצע רק על ידי משתמש שרשום למערכת והזדהה מולה.
- יתרה מכך, חלק מהפעולות יכולות להתבצע רק על ידי משתמש שהזדהה מול המערכת והוא בעל הרשאות "מנהל".
- למשל, כל משתמש רשום יכול לצפות בסרט, לקבל המלצות וכו', אבל רק מנהל יכול ליצור/לערוך/למחוק סרט.
- כאשר משתמש מזדהה מול המערכת, **הוא מקבל בחזרה JWT, ומאותו הרגע לכל הפעולות מול השרת תמיד יצורף הJWT של הלקוח.**
- עליכם לחשוב לכל פעולה האם היא:
  - יכולה להתבצע על ידי משתמש שאינו מחובר למערכת
  - יכולה להתבצע רק על ידי משתמש רשום
  - יכולה להתבצע רק על ידי מנהל

  ולהגן על הגישה לפעולות בעזרת JWT בהתאם. (**רמז**: הJWT צריך לבטא לא רק את שם המשתמש של

המשתמש המחובר אלא גם את הסוג שלו)

* JWT נלמד בתרגילון

<u>כיצד בא לידי ביטוי בAPI?</u>
עבור הכתובת: http://foo.com/api/users
- פעולת Post יוצרת משתמש חדש

עבור הכתובת: http://foo.com/api/tokens
- פעולת Post יוצרת JWT למשתמש שרשום למערכת

בסוף התרגיל יש דוגמאות לעבודה מול הAPI.

<u>הערות:</u>

עליכם להדגים את הרצת כל המערכת כולה ביחד בעמודי wiki בתוך גיטהאב. (צרו תיקייה בשם wiki ושם שימו הסברים ותיעוד להכל)
כלומר, להראות תהליך מסודר של קימפול כל הקוד שצריך לקמפל והרצת כל השרתים והאפליקציות בעזרת docker-compose.
כולל כל ההסברים, הפקודות וצילומי מסך מתאימים.
יש להדגים בwiki מספר תהליכים עיקריים:
- הרמת כל הסביבה
- התחברות והרשמה
- יצירת/עריכת/מחיקת סרטים וצפיה בהם בקליינטים השונים.

נדרש עיצוב 'סביר'. יינתן בונוס על עיצוב מעולה. יירדו נקודות על עיצוב מינימלי. (ניתן לקבל יותר מ 100 בתרגיל זה בעקבות בונוס אך לא ניתן לקבל יותר מ 100 בציון הסופי בפרויקט או בקורס. במידה וציון הפרויקט הסופי יהיה מעל 100, הציון יעוגל ל 100)
שימו לב שלכל סביבה יש "שפה" עיצובית. כלומר, אפליקציית נטפליקס באינטרנט נראית שונה מאיך שהיא נראית בטלפון, כי אלו שתי סביבות שונות עם "אופי" שונה. גם המימוש שלכם מבחינה עיצובית צריך להיות תואם לסביבה.
למשל, באינטרנט התפריט הוא תפריט עליון. עם זאת, בטלפון, התפריט אולי "נשלף" מצידי המסך בעת לחיצה על כפתור.

אנו מעודדים אתכם להגדיל ראש ולהגיש ממשק משודרג אשר מכיל מעבר למינימום הנדרש, אך חייב לכלול את המינימום שהוגדר.

יש לכתוב אפליקציית React המורכבת מ components מתאימים.

יש לחלק את הקוד חלוקה טובה ונכונה לרכיבים, ולא (למשל), פשוט ליצור רכיב אחד שיכיל את כל הקוד.

יש לממש את אפליקציית הAndroid בארכיטקטורת MVVM בעזרת Room ו Repository pattern.

במסך ההרשמה וההתחברות, עליכם לממש ולידציה על שדות הקלט.
כלומר, כל השדות הם שדות חובה, ויש לממש לוגיקה הגיונית לשאר השדות ולהציג חיווי ויזואלי ברור למשתמש לאיזה ערכים מותר להזין לפני ההזנה - וכן הודעה מתאימה כאשר הוא הזין ערכים לא מתאימים. למשל, סיסמא באורך של 8 ספרות לפחות, שילוב של תווים ואותיות וכו׳.
לא ניתן להירשם לאפליקציה או לבצע התחברות בלי מילוי נכון של השדות.
(מיותר לציין, שהולידציה הזאת אמורה הייתה להיות ממומשת גם בצד השרת בתרגיל 3, בכל פונקציה שהיא עם הולידציות המתאימות לפונק׳)

יש לוודא שהלוגיקה של התמונה במסך ההרשמה עובדת (בחירת תמונה מציגה את התמונה שנבחרה, אימות סיסמא וכו'). המשתמש בוחר תמונה קיימת מהמחשב/טלפון/מצלמה.

לאחר הרשמה תקינה לאפליקציה, הזנה נכונה של פאטי המשתמש (ושלהם בלבד) תכניס את המשתמש לאפליקציה דרך מסך ההתחברות. יש להגן על גישה לעמודים המצריכים הרשאות (למשל, מסך ניהול רק מנהל יכול לגשת אליו ומסכים הדורשים משתמש מחובר יוצגו רק אם המשתמש עבר התחברות בהצלחה). יש להגן על עמוד אלו, כך שניסיון לגשת אליהם בלי להתחבר בהתאם מציג את מסך ההתחברות.
אל תזינו/תאחסנו מידע רגיש כלשהו בפרויקט (למשל, סיסמאות אמיתיות, הודעות רגישות, סרטים, תמונות וכו')

המידע שהמשתמש נרשם/מתחבר איתו (שם, תמונה וכו') יופיע במיקום המתאים במסך הראשי.

יש לממש את הלוגיקה הרלוונטית בעמוד הראשי.
למשל, התנתקות מעבירה את המשתמש למסך ההתחברות ומנתקת אותו מהאפליקציה.

מבחינת ריאקט:
חובה להשתמש ב useState, useRef.
חובה להשתמש ב Router (כלומר, האפליקציה נטענת פעם אחת בלבד ואין רפרוש במעבר בין העמודים).

הדגש העיקרי בתרגיל זה שעל המערכת להיות שלמה ומתפקדת ⇐ כמו אפליקציה אמיתית.
למשל, אם משתמש מזין שם משתמש/סיסמא לא נכונים - יש להציג לו הודעה ברורה מתאימה.
אם יש בעיית ולידציה בהזנת נתונים - יש להציג לו את זה.
אם יש כבר שם משתמש כזה כשהוא מנסה להירשם - יש להציג לו את זה.
וכו' וכו' וכו'.

המשתמש צריך להבין לכל אורך הדרך כשהוא עשה משהו לא תקין - ושימוש תקין לאורך האפליקציה בתרחישים השונים צריך לעבוד.

ל Web, מותר להשתמש ב HTML, CSS, BOOTSTRAP, JAVASCRIPT, REACT בלבד. פניה לעמוד הבית של השרת, מחזירה את אפליקציית הReact. (**רמז**: עשיתם את זה בתרגילון)

אסור להשתמש בשום סיפרייה/מודול שלא הופיע במצגות הקורס (או חומרי העזר) או קיבל אישור מפורש לשימוש בפורום הקורס. אין לבקש אישור מילולי מהמרצה/מתרגלים - רק דרך הפורום.
אין צורך לבקש אישור על דברים טריוויאלים ומובנים בשפה. למשל, מבני נתונים וכו׳.

חובה לעבוד בREACT. לא מספיק רק לשים את ה html ושאר הקוד בסביבה של REACT, אלא ממש להשתמש בתשתית. כלומר, לחלק את הקוד ל components. הדבר ייבדק על ידי הבודק.

חובה להגיש את התרגיל בזוגות או בשלשות. הגשה ביחידים (מכל סיבה שהיא), תגרור תקרה של ציון מקסימלי של 75 בכל תרגיל שיוגש ביחידים. לא ניתן יהיה לקבל ציון מעל 75 בתרגיל שהוגש ביחידים.

חובה לעבוד בגיט לאורך כל התרגיל. כל סטודנט עובד מהמחשב שלו ומהמשתמש הנפרד שלו. לא ניתן לעבוד ביחד מאותו מחשב או מאותו משתמש או ללא גיט. חובה לעבוד בשיטת feature branches בלבד ולתחזק branch ראשי שאליו ממרג׳ג׳ים את הfeature branches. חובה למרג׳ג׳ קוד לbranch הראשי על ידי pull request בלבד, וחובה שמאשר הpull request יהיה שונה ממי שביקש את ה pull request (במידה כמובן שאכן הבקשה אושרה - ייתכן ומי שבוחן את בקשת הpull ימצא בעיות במימוש שיש לתקן טרם אישור הבקשה). כל pull request חייב לעבור אישור של כל שאר חברי הקבוצה.
כלומר, אם אתם זוג - אחד פותח pull request <u>והשני עובר על הקוד **ונותן הערות** (אם ישנן) **ורק אם הקוד תקין**</u> מאשר את הrequest. אם אתם שלשה - אחד פותח pull request והשניים האחרים צריכים לעבור על הקוד ונותנים הערות (אם ישנן) **<u>ורק אם הקוד תקין</u>** שניהם מאשרים את הrequest. דרך אגב, לgithub יש פיצ׳ר מובנה שמאפשר לעשות את זה. אפשר להשתמש בו אבל לא חובה. (הוא זמין בחינם רק למי שעשה את החשבון כחשבון סטודנט, ואנחנו משתמשים **<u>רק</u>** בכלים חינמיים). אם עשיתם חשבון מתאים, אתם יכולים להשתמש בזה וזה פשוט לא יאפשר למי שפתח את הpull request לאשר לעצמו. אם בחרתם שלא להשתמש בפיצ׳ר המובנה, באחריותכם להקפיד לבצע את הלוגיקה שהוגדרה (שאחד פותח PR והאחרים מאשרים).

(למרג׳ג׳ == לעשות merge)

חובה לתכנן ולחלק את המשימות בJIRA כספרינט, ולתחזק אותן בהתאם להתקדמות בפרויקט. אלה דרישות מהותיות ומשקלם בתרגיל משמעותי.
אני ממליץ **<u>לא</u>** לחלק את העבודה בצורה שבה מישהו מהקבוצה עושה רק React ומישהו עושה רק Android. בצורה כזאת לא תתנסו בשתי הטכנולוגיות - וחבל.

את התרגיל יש להגיש למודל לתיבת ההגשה על ידי הגשת קובץ טקסט בשם details.txt עם שמות ות.ז. של המגישים וקישור לגיט. שימו לב, חובה על הקובץ להיות בפורמט הבא:

Israel Israeli 123456789
Israela Israeli 012345678
Israeli Israeli 012345679
LINK TO GITHUB HERE

בלי רווחים נוספים, בלי שורות נוספות, ובשפה האנגלית בלבד. אי הגשה של קובץ ה details.txt הנ״ל או הגשתו באופן שונה ממה שהוגדר, **תגרור הורדה של 20 נקודות בציון התרגיל.**

אנא דאגו לכך שהגיט שלכם יהיה במצב private לכל אורך הסמסטר. יש להוסיף את מייל המתרגלים (שהם פירסמו) עם הרשאות צפייה בGithub ובjira.

שאלות לגבי התרגיל יש לשאול בפורום בלבד. פניות בנושאים פרטיים יש לשלוח למתרגלים.

עבודה עצמאית של המגישים בלבד. העתקות יבדקו הן ידנית והן ע״י מערכת אוטומטית, ויטופלו ישירות על ידי המחלקה.

על הגיט להכיל קובץ README שמסביר איך להריץ את הקוד וכן מסביר את תהליך העבודה שלכם בקצרה.

אין לאחסן (או לשלוח) מידע חשוב/רגיש כלשהו (כמו סיסמאות אמיתיות או טקסט חשוב) אל השרת או בשרת, כי התרגיל לא לוקח בחשבון דגשים רבים שיש לקחת בהקשר של ה security של האפליקציה.

אני מזכיר, אין להעלות שום סיסמאות, מידע פרטי, רגיש או סודי, מפתחות, טוקנים, API Keys וכו׳ וכו׳ וכו׳ בשום מקום בפרויקט ובקורס. לא בgithub, לא בקוד, לא בREADME, לא ב WIKI לא בJIRA או בDOCKER או ב DB ולא בשום מקום אחר. זיכרו שלרוב הדברים יש היסטוריה - והמידע מופיע גם שם, אפילו אם מחקתם אותו.

אל תעלו לגיט את תיקיית ה node_modules (ובוודאי שלא שום מידע פרטי או רגיש, כמו סיסמאות או כל דבר אחר).

בקורס אנחנו עובדים אך ורק עם משאבים חינמיים, חוקיים ו open source.
לדוגמא, במידה ואתם משתמשים בקבצים כמו תמונות או סרטים, אתם יכולים להשתמש רק בדברים שמותר וחוקי ושיש לכם רשות להשתמש בהם. באחריותכם לוודא זאת. לדוגמא, יש למשל מאגרי iconים חינמיים וחוקיים של גוגל המותרים לשימוש מה שיכול להיות רלוונטי ל react ול android.
בכל שלבי הפרוייקט אנחנו לא משתמשים בשום דבר שאינו חוקי, חינמי ו open source.
למשל, אתם לא משתמשים בסרטים אמיתיים או במידע על סרטים אמיתיים.
מבחינתי, אתם יכולים לצלם את הדבר הכי סתמי כמו הרצפה/שמיים/קיר וזה יהיה הסרטונים שאצלכם באפליקציות.
אתם גם לא שמים סרטים בגיט או ב db, אלא רק להסביר לבודק איך לעשות את זה.

עבודה עצמית בלבד. הנושא ייבדק ויאכף. שימו לב, התרגיל הוא שלכם ואתם צריכים לממש אותו ולא כלים מבוססים Gen AI. שימוש בכלים מסוג זה באופן שכלים אלו מממשים עבורכם חלקים מהתרגיל שקול לעבודה שאינה עצמית ועצמאית. הנושא ייבדק וייאכף.

בהצלחה

<u>נספח:</u>

<u>נדגים עבודה מול השרת בעזרת קוד לדוגמא:</u>

נניח ואנחנו רוצים לפנות לשרת בעזרת JAVASCRIPT בצורה א-סינכרונית.
נוכל להשתמש בדוגמאת הקוד הבאה:

```html
<html>

<body>
    <script>
        async function foo() {
            const data = {
                username: 'guest',
                password: '123456'
            }

            const res = await fetch('http://localhost:5000/api/Tokens', {
                'method': 'post',
                'headers': {
                    'Content-Type': 'application/json',
                },
                'body': JSON.stringify(data)
            })
        }

        foo()
    </script>
</body>

</html>
```

ואז לחלץ את הJWT מהתשובה ולקבל את הפרטים של משתמש על ידי שליחת בקשה נוספת בצירוף הJWT.
כעת, נוכל לצרף לפניות עתידיות לשרת את ה JWT כדי לאמת את זהותנו.

דוגמא נוספת -
נניח ובמסך הבית של אפליקציית ה React, היכן שמוצגת רשימת הסרטים, מימשתם את רשימת הסרטים באופן הבא. (הקוד מפושט לשם קריאות)

יצרתם state שנקרא videos:

```
const [videos, setVideos] = useState([]);
```

ואז, עבור כל video שכזה יצרתם קומפוננטה והעברתם לה את פרטי הוידאו:

```
const listItems = videos.map((video) =>
```

```
    <Video video={video} />
  );
```

ואז הצגתם את רשימת הסרטים:

```
<div className="videos">
     {listItems}
</div>
```

כדי להפוך את הקוד הזה לדינמי שעובד מול השרת, מביאים את רשימת הסרטים מהשרת (כפי שהוסבר לעיל עם fetch), ומעדכנים את משתנה הstate עם הרשימה, כלומר:

```
const data = await response.json();
setVideos(data);
```

וזהו. ריאקט ידאג לרנדר אוטו' את הרשימה בהתאם למה שהגיע מהשרת.