

תרגיל 5

nbaim: <https://classroom.github.com/a/4Rg9eMR5>

শטרואס: <https://classroom.github.com/a/qkErr9Qk>

תאריך הגשה: 29/01/2026

חלק מבדיקה התרגיל, לו"ז יפורסם לפגישה במכלה – למרות שלא ס"ימתם לעבוד על התרגיל. חובה לכל הוצאות להגעה בשבוע האחרון באחד הימים (שני/שלישי 01/27-26). מטרת הפגישה הינה לוודא שאמת/
שולטים(ות) בחומר שהוגש או יוגש.

מסמך זה כולל הרבה הנחיות ותשובות לשאלות רבות (מניסיון של שנים קודמות), מומלץ לעבור עליו תוך כדי התקדמות בעבודה.

מטרת התרגיל היא לבנות אתר מסווג chatroom הכלול גם כשרת וגם כלקוח בערת הטכנולוגיות שנלמדו:
(express, javascript, html, css) בשילוב עם מסד נתונים mysql (mariadb).
ראו באתר הנחויות להקמת המasd נתונים docker ושם המasd נתונים.

קודם נתמקד בחלק א': **רישום משתמשים** (דף SPA שניתן כבר למש עם החומר הנלמד)
הכינו את הפרויקט לפני ההנחיות בסוף המסמך. עד שנלמד על מסדי נתונים אפשר לבנות דף SPA (html) של הרישום כמו שעשיתם עד עכשיו. דף זה יכול לשבת בתיקיה public או ב-views (ואז כתובים route שעושה route לאוטומatically). לגבי ה-backend תסתכל על הדוגמא של api-rest. בהמשך תוכלנו לשפר את מבנה הפרויקט עם המasd נתונים והפדרה מלאה routes/controllers.

יש להגיש ב-15/01 את חלק א' (אנחנו פשוט נוריד את כל ה-code בתאריך זה בבדיקה בשעה 23:59 – אחר כך תמשיכו לעשות push עד ההגשה סופית). הגשת חלק א' לא חייבת להיות error-free. המטרה היא לראות התתקומות כלומר מימוש הרישום עם טיפול בקוד. הגשה זו תחשב כ-20% מציון התרגיל. ככלור מי שלא השלים לרוב את חלק א', הציון של התרגיל לא עלה על 80%.

חלק א' : רישום משתמשים

- דף נחיתה ("/") של האתר יציג טופס login וקישור לדף רישום.
דף הרישום לאתר יכול קישור לדף login וטופס הרשמה המאפשר הזנה של:
- דואר אלקטרוני
 - שם פרטי
 - שם משפחה

לאחר הגשת הטופס (submit), צד השרת בודק אם הדואר האלקטרוני כבר בשימוש (הוא מנהל את רשימת המשתמשים הרשומים באתר). אם הדוא"ל כבר בשימוש (קיים משתמש שכבר גירש עם כתובת זו), יש להציג הודעה שגיאה (כגון "בתווך הדף עצמו".)

ולידציה צד לקוח :

- נתן לנצל את html כדי למש ולידציה צד לקוח (required, input type etc (required, input type etc))
- קלט שאינו ריק, אותיות-z-a בלבד שם ושם משפחה, דוא"ל תקין (ניתן להשתמש באימות מובנה של HTML5), אורך מינימלי 3 תווים לשם ואורך מקסימלי של 32 אותיות בכל השדות(כולל סיסמה)
 - אין חשיבות לאותיות גדולות/קטנות (case insensitive) עבור כל קלט. כלומר a@b.c שcool c-B.C
 - חלק זה יכול בניין api-rest (כלומר כל חלק זה מומש כ-views express(express-c-views) אחד בלבד)

אחרי הזנת השם והדואר אלקטרוני, עוברים בדף הפתור לדף שלב הבא להמשך רישום. דף זה מכיל טופס בעזרתו המשמש יכול לבחור סיסמה. המשמש מזין סיסמה עממית ("password" type="password") ולוחץ שוב על כפתור submit כדי לשלוח את הסיסמה לשרת ולסיטם את ההרשמה. בסיטם ההרשמה חוזרים לדף login עם הודעה מתאימה ("you registered now are"). בדף הזנת הסיסמה קיימים גם כפתור back שמחזיר לדף הזנת השם. קלט המשתמש ישמר יוצג בטופס לטוח זמן (ראו הסברים בהמשך).

תכניות אינטראנט א' - תשפ"י

מطبع הדברים משתמשים רבים להירשם בו זמנית. נדרש לבצע את בדיקת קיום הדוא"ל לפני כל הוספה ממשית למסד/מבנה נתונים של השירות (כלומר לא רק בבדיקה ראשונית) וכן יש לבדוק שוב את קיום הכתובת מייל בסיסים הרישום של המשתמש מצד שרת. אם הכתובת קיימת יש להציג את המשתמש להתחלה הרישום והודעה מתאימה ("this email is already in use, please choose another one") כאשר קלט המשתמש הוזן ב-`req` כל עוד הקוקיז קיים (ראה פסקה הבאה).

קיימות דרישת להשתמש בקוקיז על מנת לשמור את פרטי המשתמש לטווח זמן של 30 שניות (מהרגע שהמשתמש עבר למסך השני של הרישום). הגדרו מספר זה כ-`const REGISTER_TIMEOUT = 30000` בתחלת התכנית כדי שנוכל לשנות אותו בקבילות. כל עוד המשתמש נמצא בטוח זמן זה, אנחנו תמיד נציג את הטופס הראשון עם הקלט שהוא כבר הינו ונאפשר להשלים את רישום. בדף הזנת הסיסמה, אם הוא חזר למסך הראשון, או אולי סגר את הדפדפן, ווחזר לכתובת URL של שלב ראשון של הרישום, הטופס יכלול את הקלט שלו כל עוד לא עברו `REGISTER_TIMEOUT` שניות. אם עבר הזמן, שליחת לשרת להשלמת הרישום של דף הזנת הסיסמה, יעביר אותו למסך הראשון עם טופס מאופס, במקומם להשלים את הרישום.

לאחר סיום הרישום שדות הטופסים מאופסים וכך נאפשר רישום משתמש חדש. מומלץ לבדוק את האתר עם `REGISTER_TIMEOUT` ארוך על מנת לבדוק את כל התסרים האפשריים.

הכתובת לדף רישום הינו `/register`. יש למשמש את "/" - דף "נוחות" - כך שיביא את דף ה-`login` אך כדי שהוסבר בשיעור דף ה-`login` לא יהיה זמין למשתמש כבר מחובר ויעביר אותו ל-`chatroom`.

סיום הרישום מעביר לדף login (זהו URL אחר "/" כלומר route אחר). לאחר הצלחה ברישום, נציג דף חדש `login` עם הודעה בהתאם "you are registered".

חלק ב': chatroom

דף chatroom

זמן רק למשתמש מחובר, אחרת אין להציג אותו בכלל ועוברים למסך `login`. כאשר משתמש מתחבר נציג לו את ההודעות של `chat` בסדר יורד (תאריך עדכני ביותר קודם), כל הודעה תוצג יחד עם שם הפרטיו של המשתמש.

דף זה יכול:

- טקסט: `username` `Welcome <username>` כאשר `first name` מכיל את שם של המשתמש המחבר
- `text input` וכפתור לשילוח הודעה לצ'אט
- כפתור ליציאה מה-`chat` (logout) ומעבר למסך `login`
- צ'אט: רשימת ההודעות שנשלחו עם שם השולח, תאריך וזמן בפורמט קרייא, ואת הודעה

הצ'אט מתעדכן כל 10 שניות: הגדרו מספר זה כ-`const POLLING` בתחלת התכנית כדי שנוכל לשנות אותו בקבילות. יחד עם מגנון `polling`, יש להשתמש ב-api `rest` על מנת לחוץ הודעה עדכנית. מומלץ לבדוק את האתר עם ערכים שונים (כגון 5 דק.) על מנת לבדוק את כל התסרים האפשריים. יש לדאוג לכך שכל הגולשים רואים את אותו תוכן של הודעות באותו סדר. תוכן הצ'אט מתעדכן מיד אחרי הוספה חדשה להודעה (ולא רק בסיבוב הבא של `polling`).

יש לאפשר למשתמש לעורוך או למחוק כל הודעה שהוא שלח.

- כדי לאפשר למשתמש עריכת הודעה (פעולה זו תמומש ב-fetch) תוך כדי זה שיש עדכנים בדף, מומלץ ליצור אלמנט של ה-`DOM` שלא מתרכב עם תוכן ההודעות המתעדכן בו בזמן.
- כדי לטפל בעניין העדכנים מומלץ להשתמש ב-mode paranoid `mode` של `fetch` (`rest api`) אלה כתופס רגיל עם `route` של `express`. לגבי מחיקה פעולות הוספה ומחיקה ימומנשו לאפשר (`rest api`) אלה כתופס רגיל עם `route` של `express`. נציג נציג למשתמש הודעה כגו"ן "Are you sure? Deleting a message cannot be undone". יחד עם תוכן ההודעה. לאחר הוספה או מחיקה (או ביטול), נחזיר את דף ה-`chat`.

ניהול מבנה הנתונים של ההודעות:
חשוב לעדכן את תוכן הצ'אט רק אם יש עדכן מצד שרת. כלומר יש צורך להחזק תאריך עדכן לקליל הצד לקוח על מנת

תכניות אינטראנט א' - תשפ"י

להשוות בין שרת-לקוח ולבקש מהשרת עדכנים. נציג גם שמהינה הנתונים נמצא לצד שרת ומומלץ לא להחזיק מבנה דומה בצד לקוח אלא לסמור על פעולות rest api על מנת לקבל תוקן תמיד עדכני. ווסף כי לצורך פשטות אפשר לרענן את כל תוכן ההודעות על כל עדכן במקום לבצע עבודה כירוגית ב-dom. במקרה זה אין להשתמש ב-web socket.

סיכום סשן:

נניח והמשתמש התנתנק: מצב זה יכול לקרות כאשר המשתמש פתח שני דפים והוא מתנתנק באחד מהם, או שהוא מחק קוקיז, או שהוא session-shlo פג תוקף. במקרה זה אנחנו רוצים להעביר אותו לדף login. מספיק לגלות זאת בתקשורתם של השירות: ככלומר בעבר לדף, בשילוחה הודעות, חילוץ הודעות, חיפוש או בכל פעולה רשות (כגון מנגנון polling שיכל לטפל בהז תוך כדי). אז גודאג להעביר את המשתמש לדף login (בצד שרת קיימת פעולה redirect, שימוש לבצד לקוח עושים זאת ע"י response.redirect ובצד שרת window.location())

:Rest api חשוב לאמת את המשתמש בכל פעולה של rest api וכן למנוע פעולות סדוניות כגון מחיקת הודעות של משתמשים אחרים.

חיפוי

זמן רק למשתמש מחובר, כמו כל דף הקשור ל-chat. דף זה יאפשר חיפוי טקסט בתוך מסד הנתונים לפי תוכן הודעות. ניתן למש את החיפוי באותו דף של ה **chatroom** או בנפרד.

באופן כללי חיבים לאפשר למשתמש גישה בין דפים על ידי כפתורי כניסה בדף עצם, או לינקים של האתר עצמו ולא להתבסס על הcptorios של הדף (לדוגמא דף תוצאות חיפוי חיב כפתור להמשך גישה/חזרה לצ'אט).

יש למש:

- כניסה לצ'אט (login)
- יציאה מהצ'אט (logout)
- הוספה הודעה
- מחיקה ועריכת הודעה ע"י המשתמש על ההודעה
- חיפוי הודעות לפי טקסט בהודעות
- קבלת הציאת על ידי הלקוח ועדכן הדף rest api()
- טיפול שגיאות שונות (input validation)
- טיפול בסיסים ששן (מעבר כל דפי הציאת: מעבר אוטומטי ל-login) ולהפרק מעבר אוטומטי לציאת במידה והמשתמש כבר מחובר
- טיפול בשגיאות תקשורת (הודעה רלוונטית למשתמש)

הצעת תכנית עבודה לחילק ב':

- פתחת את דף ה-login (שימוש ב-session) ואחרי login מוצלח להציג דף סמלי.
- פתחת ולבדוק (בעזרת REST API postman) ניהול הודעות (חילוץ, הוספה, חיפוי, מחיקה, עדכן – אפשר לעבד עם משתמש קבוע סמלי לצורך בדיקה)
- להוסיף את כל נושא אימונות/הרשאות (שימוש בסשן)
- פתחת את הדפים וקוד צד ל��וח לתצוגה ופעולות של הציאת מול ה-rest api

הערות כלליות:

חשוב לשמר על מסד נתונים עקבתי ולא תוכן מיותר (לדוגמא לדאוג שלא לשמר מידע חלקי או זמני שלא יהיה רלוונטי להמשך). נצלו את השאלות: ניתן למיין/לسان/לשלב בקשות בבקשת אחת, אל לבצע פעולות בקוד javascript אם ניתן לעשות זאת כבר ב-libs (כגון מיוו).

בדקו את האתר שלכם עם 2 דפים ומשתמשים שונים! בדקו את עניין סגירת הסשן בכל מצב / פעולה אפשרית באתר.

ראו את המסמך שגיאות נפוצות שבדרייב עם התרגילים.

יש להציג את כל הפרויקט ללא תיקייה node_modules. שימוש לב לא לשוכח להכין את הקובץ package.json או תמונות.css (כגון מיין).

לא לשכו trim לכל קלט שколо רוחים נחשב ריק.

יש גישות שונות לאיתר מסוג זה, ואין ספק שהאטר חלקי ואפשר להתמקד בדברים אחרים, אך מותר להציג שינויים בדרישות, פשוט כתבו לי ואם תקבלו אישור מני, תודיעו לבודק.

התרגיל משלב את כל החומר הנלמד ובכל זאת, לא מדובר בתרגיל גדול אבל כן מורכב בארכיטקטורה. רצוי קודם להבין לעומק את הקוד לדוגמא באתר. אנחנו נחזיר על כל החומר בשבעים אחרונים לסמינר ונדריך אתכם על מנת לסייע את התרגיל בזמן. מאוד מומלץ להיות נכון בשיעורים. הניסיון מוכיח שעבודה עצמית במקביל לקורס יכולה להאריך את הזמן ולסייע לכם בצורה משמעותית.

בהצלחה!

Short Guide: How to Initialize your Repo

1. Open the GitHub repo in Webstorm, you have an empty project, add a README.md file and write your names and email (the README.md format is explained here : <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>)
2. Open the terminal and type
`npm install -g express-generator`
`express -e`
At this point you have an express project that can run.
The code includes 2 routes (index.js and users.js), we recommend cleaning up and remove these files and write your own routes (irrelevant code such as code examples from class in your submission is penalized).
If you have trouble with this step, create another express/ejs project from WebStorm menu as shown in class, and drag all the files into your empty project.
3. Optionally add a run configuration as shown in class, or simply type in terminal:
`npm start`
Open your browser on <http://localhost:3000> and make sure you see a welcome page
4. Now you want to add the required libraries (you might need more libraries later)
`npm install cookie express-session mysql2 mariadb sequelize`
If you have errors (ERR), look for solution in the sequelize slides.
5. create the *models* and *controllers* folders:
Until we learn about databases you can write the models similarly to 11-MVC-shop example (note the separation between routes and controllers). Copy from this repo the models/index.js into your folder and add the DB initialization code in your app.js. Regarding the REST api backend, look at the rest api example.
6. Set up your database as instructed in the website