# **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הגשת פרויקט**

**להגשה עד 12.2.25 בשעה 23:59**

**הגשת פרויקט**

**<Exam Monitoring App>**

**< B18>**

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
| מגד עווד | 324033869 |
| איה חרמה | 324886290 |
| בשאר אבראהים | 213820822 |
| עלי חיגאזי | 212704670 |

**תקציר הפרויקט**

המערכת היא Exam Monitoring System לניהול ובקרה של בחינות בזמן אמת במוסד לימודי.  
מטרת המערכת לאפשר לצוות הבחינה לנהל את כל תהליך הבחינה במקום אחד החל מיצירת הבחינה והקצאת סטודנטים ועד מעקב בזמן אמת והפקת דוחות.  
המערכת מציגה תמונת מצב חיה של הכיתה כולל מיקום כל סטודנט, סטטוס נוכחות, יציאות זמניות, העברות בין חדרים ואירועים חריגים. בנוסף קיימת תקשורת בין צוות הבחינה ותיעוד מלא של פעולות. בסיום הבחינה ניתן להפיק דוחות מפורטים וסטטיסטיקות.

**פונקציונליות לפי משתמשים**

Admin  
ניהול משתמשים במערכת  
יצירה עדכון ומחיקה של בחינות  
הקצאה אוטומטית של סטודנטים לחדרים ומושבים  
גישה מלאה לדוחות וסטטיסטיקות

Lecturer  
צפייה בדשבורד הבחינה בזמן אמת  
צפייה והורדה של דוחות הבחינה  
מעקב אחרי מצב הבחינה והתקדמותה

Supervisor  
ניהול הבחינה בפועל במהלך הריצה  
עדכון סטטוס סטודנטים הגעה יציאה סיום וחריגות  
הוספה ומחיקה של סטודנטים לבחינה  
פתיחת אירועים ושליחת הודעות  
ניהול בקשות העברה בין חדרים ומושבים

Student  
התחברות באמצעות קוד אימות שנשלח למייל  
צפייה בפרטי הבחינה האישיים והדוח האישי

**מימוש טכנולוגי**

Frontend  
React  
Vite  
React Router  
Axios  
i18next לרב לשוניות  
Chart.js לגרפים  
Google reCAPTCHA  
QR scanner  
PDF renderer  
XLSX export

Styling  
Tailwind CSS כולל תמיכה במצב כהה

Backend  
Node.js  
Express  
Mongoose  
Session authentication עם MongoStore  
Zod validation  
Helmet אבטחת כותרות  
CORS  
Compression  
Rate limiting  
WebSocket לתקשורת בזמן אמת  
PDFKit ליצירת דוחות  
Nodemailer לשליחת מיילים  
Google Generative AI לצאט

Database  
MongoDB Atlas  
אחסון משתמשים בחינות כיתות והעברות

<קישור לתיקיית גיט ציבורי : >

<קישור לאתר : >

1. עליכם לסיים את בניית האתר לפי האלמנטים המתקדמים שלמדתם וביצעתם בתרגילי בית 1+2.
2. יש למנות מהנדס מערכת בכל צוות, אשר יהיה אחראי על הגדרת והקצאת המשימות בתרגיל זה.  
   נא לרשום את שם הסטודנט בתרגיל זה. על מהנדס המערכת לכתוב כיצד נעשתה חלוקת העבודה מול הצוות, מה היו המשימות של כל חבר צוות, האם היה ממשק בין חברי הצוות, והאם המשימות מולאו:

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| עלי | * מילוי טבלת משימות * חישוב ממוצע SUS * תיקיית מתכנת בשיתוף עם בשאר * תיקונים בקוד הפרויקט | הכל |
| בשאר | * דרישות פונקציונאליות/לא פונרציונאליות/ממשק חיצוניות * תיקיית מתכנת בשיתוף עם עלי * תיקונים בקוד הפרויקט | הכל |
| מג'ד | * ארכיטקטורת המערכת + usecase * קומפננטות של המערת בשיתוף עם איה * צילום מסכי המערכת + סרטון שיווקי * טבלה ראשית והסבר על פוטיציאליות הפרויקט * תיקוהים בקוד הפרויקט | הכל |
| איה | * פסקה מסכמת * קומפננטות של המערכת בשיתוף עם מג'ד * תיקונים בקוד המערכת | הכל |

2. הציגו רשימת דרישות:

יש להתבסס על המבנה של asana (<https://asana.com/resources/software-requirement-document-template>

* דרישות פונקציונליות
* דרישות לא פונקציונליות (בנפרד, יש לסווג דרישות לא פונקציונליות לפי wikipedia NFR).
* דרישות ממשק חיצוניות.

**שם המערכת**

**Exam Monitoring System** - מערכת לניהול והשגחה על בחינות עם דשבורד חי, מפת כיתה, נוכחות, אירועים חריגים, העברות סטודנטים, דוחות, וכניסת סטודנט באמצעות OTP.

**-1 רקע ומטרת המערכת (Background + Purpose)**

**המערכת נועדה לתמוך בבחינות) בעיקר(onsite באמצעות:**

* ניהול בחינות (יצירה/התחלה/סיום)
* מעקב נוכחות סטודנטים בזמן אמת
* שליטה במצבי סטודנט (נוכח/מחוץ זמנית/סיים/וכו’)
* רישום אירועים/תקריות (incidents) במהלך הבחינה
* ניהול בקשות העברה בין חדרים (transfers)
* דוחות: צפייה, CSV, וPDF
* כניסת סטודנט בצורה מאובטחת עם OTP שנשלח למייל
* תמיכה ב (עברית/אנגלית) ו-Dark/Light Mode בצד לקוח

**-2 גבולות המערכת (Scope)**

**בתוך Scope**

* אתר React (Vite) + Tailwind UI
* שרת Node/Express עם MongoDB (Atlas)
* עבודה מבוססת Sessions (cookie) עם CORS credentials
* מודולי Exams, Dashboard, Reports, Transfers, Incidents, Messages/Chat, Student area, Admin tools
* שילוב Google reCAPTCHA בהרשמה

**מחוץ ל Scope**

* WebSocket בזמן אמת .

**-3 שחקנים (Stakeholders / Users)**

המערכת כוללת Roles מרכזיים (כפי שמוגדר בשרת):

* **Admin** מנהל מערכת: ניהול בחינות, צפייה בדשבורד, דוחות, יכולות מתקדמות.
* **Lecturer** מרצה: צפייה/ניהול בהקשר הבחינה, דוחות, ממשק דשבורד.
* **Supervisor** משגיח: עבודה בחדר מוקצה, ניהול סטודנטים במפה/נוכחות/אירועים.
* **Student** סטודנט: כניסה עם OTP וצפייה בדוח/מידע אישי לבחינה .(My Exam Report)

**-4 תיאור ארכיטקטורה וזרימות (High-level Flow)**

**Client (React) 4.1**

**עמודים עיקריים:**

* HomePage
* Login / Register כולל (reCAPTCHA)
* ) Dashboard מפת כיתה + אירועים + העברות + ניהול סטודנטים(
* ) ExamsPage רשימת בחינות(
* ManageExamsPage (Admin)
* ReportsPage (Reports + Export PDF/CSV/Excel)
* StudentMyExamReportPage

**Server (Express) 4.2**

Routes מרכזיים:

* /api/auth – login/register/logout/me + OTP לסטודנטים forgot password +
* /api/exams – CRUD לבחינות + start/end + add/delete students + עדכון attendance
* api/dashboard – snapshot / snapshot-lite / clock + add/delete students
* - api/transfers בקשות העברה ואישורן
* - api/incidents יצירת incidents
* api/messages - הודעות (לוג/פיד)
* - api/reports דוחות analytics + PDF/CSV
* - api/student מידע לסטודנט
* api/chat - צ'אט בוט

**5- דרישות פונקציונליות (Functional Requirements)**

**Authentication & Users 5.1**

**-1** המערכת תאפשר למשתמש צוות (Admin/Lecturer/Supervisor) להתחבר עם username/password .  
 **-2** המערכת תאפשר רישום משתמש חדש עם fullName, email, username, password, role.  
 **-3** המערכת תחייב reCAPTCHA בתהליך הרשמה ותשלח את token לשרת לאימות.  
 **-4** המערכת תאפשר "מי אני" לצורך טעינת משתמש מחובר ושיוך הרשאות.  
 **-5** המערכת תאפשר Logout ותסיים session פעיל.

**Student OTP Login 5.2**

**-6** המערכת תאפשר לסטודנט לבקש OTP באמצעות email + מזהה סטודנט (כפי שממומש בשרת).  
 **-7** המערכת תשלח OTP למייל באמצעות שירות אימייל SendGrid) או (dev-mode   
 **-8** המערכת תאפשר אימות OTP (verify) והתחברות סטודנט למערכת.  
 **-9** המערכת תנהל תוקף OTP (expires) ומספר ניסיונות כושלים .(fail count)

**Exams Management 5.3**

**-10** המערכת תאפשר יצירת בחינה עם פרטי בסיס (courseName, examDate, startAt, endAt, examMode, סטטוס scheduled/running/ended )  
 **-11**  המערכת תאפשר צפייה ברשימת בחינות + סינון לפי סטטוס/קריטריונים.  
 **-12** המערכת תאפשר צפייה בפרטי בחינה לפי מזהה.  
 **-13**  המערכת תאפשר התחלת בחינה (Start) ועדכון status ל running   
 **-14** המערכת תאפשר סיום בחינה (End) ועדכון status ל-ended   
 **-15**  המערכת תאפשר הוספה של סטודנטים לבחינה.  
 **-16**  המערכת תאפשר מחיקה של סטודנט מבחינה.  
 **-17**  המערכת תאפשר עדכון סטטוס נוכחותseat/classroom לכל סטודנט .(attendance patch)

**Live Dashboard (Monitoring) 5.4**

**-18**  המערכת תציג דשבורד בזמן אמת לבחינה פעילה.  
 **-19** המערכת תטען Snapshot של מצב הבחינה והסטודנטים) כולל lite snapshot להפחתת עומס(  
 **-20** המערכת תציג מפת כיתה (Classroom Map) עם מושבים וסטטוסים.  
 **-21** המערכת תאפשר שינוי סטטוס סטודנט present/temp\_out/finished/moving) וכו(   
 **-22** המערכת תנהל אירועים בזמן הבחינה (Events feed) כולל .severity   
 **-23** המערכת תאפשר ניהול/צפייה בהעברות .(Transfers panel)   
**-24** המערכת תציג “Exam overview” (נתוני על של הבחינה) כולל זמן/סטטוס.

**Incidents & Events 5.5**

**-25** המערכת תאפשר יצירת incident (אירוע חריג) עם תיאור/חומרה/סטודנט/מיקום.  
**-26** המערכת תציג פיד אירועים עם סינון/תיעדוף לפי חומרה/זמן.  
**-27**  המערכת תשמור timeline לדוח בחינה (report timeline) לצורך Reports.

**Transfers (Seat/Classroom Transfers) 5.6**

**-28** המערכת תאפשר פתיחת בקשת העברה של סטודנט מחדר/מושב ליעד אחר.  
**-29** המערכת תאפשר אישור/דחיית בקשות העברה לפי הרשאות.  
**-30** המערכת תתעד העברות בסטודנט (student file/history) לצורך דוחות.

**Reports & Exports 5.7**

**-31**  המערכת תציג Reports Page עם בחירת בחינה/פילטרים/חיפוש.  
**-32** המערכת תאפשר יצוא CSV של דוח בחינה.  
**-33** המערכת תאפשר הורדת PDF לדוח בחינה.  
**-34** המערכת תספק analytics לדוחות Reports Analytics endpoint + UI) שימושי(  
**-35**  המערכת תציג היסטוריית אירועים/סטודנטים בדוח לפי בחינה.

**Student Area (My Exam Report) 5.8**

**-36** המערכת תאפשר לסטודנט מחובר לצפות בדוח הבחינה שלו בלבד.  
**-37** המערכת תציג לסטודנט נתונים כגון סטטוס, מושב, אירועים, הערות/תיעוד (לפי הנתונים שנשמרים ב (User.studentFiles

**Chat / Messages 5.9**

**-38** המערכת תאפשר יצירת הודעות/שיחה (messages/chat route)   
**-39** המערכת תציג ממשק צ’אט כחלק מהמערכת) כולל context לפי משתמש/role אם קיים(

**Role-based Access Control 5.10**

**-40** המערכת תגביל פעולות לפי: role

* **:Admin** הכל
* **:Lecturer** צפייה/דוחות/פעולות בהתאם להקשר
* **:Supervisor** פעולות חדר/נוכחות/אירועים
* **:Student** צפייה באזור סטודנט בלבד

**-6דרישות לא פונקציונליות (NFR) לפי Wikipedia Categories**

**) Performance 6.1 ביצועים(**

**-1** טעינת Dashboard תסתיים תוך ≤ 2 שניות ברשת תקינה עבור עד 30 סטודנטים.  
Snapshot-lite **-2** יפחית payload ויאפשר polling תדיר בלי להעמיס על השרת.  
**-3**  הורדת CSV תסתיים תוך ≤ 5 שניות עבור עד 300 רשומות.

**Reliability 6.2 (אמינות)**

**-1** המערכת תשמור נתוני בחינה ונוכחות ב-MongoDB כך שלא יאבדו ברענון דף.  
**-2** במקרה כשל בשליחת מייל (SendGrid), המערכת תדווח שגיאה בצורה ברורה ותאפשר ניסיון חוזר.  
**-3** כל פעולה קריטית (start/end/attendance update) תחזיר תשובת API עקבית (success/error) לUI

**Availability 6.3 (זמינות)**

**-1** המערכת תהיה זמינה בזמן בחינה פעילה ותתמוך בתרחיש שימוש רציף (monitoring).  
**-2** המערכת תספק endpoint /health לניטור זמינות שרת.

**6.4Security (אבטחה)**

**- 1**  הרשמה תוגן באמצעות Google reCAPTCHA למניעת בוטים.  
**- 2**  גישה ל-API תוגן באמצעות session cookie + CORS credentials.  
OTP**- 3** לסטודנטים יהיה מוגבל בזמן (expiry) ויכלול ספירת ניסיונות כושלים.  
**- 4** המערכת תיישם הרשאות לפי role כדי למנוע גישה לדוחות/בחינות של אחרים.

**Usability 6.5 (שימושיות)**

**- 1** המערכת תתמוך בעברית ואנגלית.  
 **- 2** המערכת תתמוך ב-RTL כאשר השפה עברית.  
 UI **- 3** יציע Feedback ברור (Toast/Alerts/Loader) לכל פעולה.  
 **- 4** המערכת תכלול Dark/Light Mode עקבי לכל העמודים המרכזיים.

**Maintainability 6.6 (תחזוקתיות)**

**- 1**  הקוד יחולק למודולים: routes/controllers/models בשרת ורכיבי UI/hooks/services בלקוח.  
**- 2**  שימוש ב-services layer בצד לקוח יאפשר החלפת endpoints ללא שינוי UI נרחב.  
**- 3**  שימוש ב-snapshot-lite יאפשר שיפור ביצועים בלי שינוי לוגיקה פנימית.

**Portability / Compatibility 6.7 (תאימות)**

**- 1** המערכת תעבוד ב-Chrome/Edge/Firefox גרסאות מודרניות.  
**- 2** המערכת תתמוך במחשב, טאבליאים וסמארטפונים.  
**- 3**  המערכת תעבוד בפריסה Client (Vercel) + Server (Render) + DB (MongoDB Atlas)

**Scalability 6.8**

**-1** המערכת תתמוך ב-≥ 100 משתמשים מחוברים בו זמנית (polling).  
**-2** מנגנון snapshot-lite יאפשר גדילה במספר משתמשים תוך שמירה על זמני תגובה.

**7- דרישות ממשק חיצוניות**

**Interfaces – Client ↔ Server- 7.1**

* REST API תחת api
* שימוש ב - credentials (cookies) עבור session
* פורמט JSON לכל הבקשות/תגובות

**Database Interface 7.2**

* MongoDB Atlas (Mongoose Models): User, Exam, Classroom, TransferRequest
* שמירת:
  + Attendance per exam
  + Events/Incidents
  + Student files/history (בUser.studentFiles - )

**7.3Email Service Interface**

* SendGrid API (https://api.sendgrid.com/v3/mail/send)
* שימוש בENV: SENDGRID\_API\_KEY, SENDGRID\_FROM -
* : fallback dev modeהדפסה ללוג אם אין מפתחות

**Google reCAPTCHA Interface 7.4**

* Client: react-google-recaptcha
* :Server אימות token מול RECAPTCHA\_SECRET\_KEY
* :ENV
  + VITE\_RECAPTCHA\_SITE\_KEY (Client)
  + RECAPTCHA\_SECRET\_KEY (Server)

**Hosting/Deployment Interfaces 7.5**

* Client build: Vite
* Server: Node/Express
* CORS רשימת origins + תמיכה בvercel.app -

. **3רשמו פסקה מסכמת עבור התהליך שנעשה בפרויקט - פגישות עם לקוחות (במידה ויש) ,תקציר מסמכים שנאספו, מודולים מרכזיים (במידה ויש תוצרי ביניים כגון תמלולי ראיונות,מסמכים וכו, ניתן לצרף בנספח).**

במהלך הסמסטר בוצע תהליך פיתוח הדרגתי בליווי הלקוחה גב׳ יעיל ארז, כאשר בכל שלב המערכת התקדמה מרעיון ראשוני לפתרון מלא. בתרגיל בית 1 נפגשנו עמה לצורך אפיון והצגת אבטיפוס ראשוני שהתמקד בממשק בלבד, במטרה להבין את אופן העבודה בפועל בזמן בחינה ואת הצרכים של צוות הבחינה. בעקבות המשוב הוגדרו זרימות העבודה, הרשאות המשתמשים והמסכים המרכזיים. בתרגיל בית 2 הצגנו מערכת חלקית הכוללת חיבור לשרת, ניהול בסיסי של בחינות ומעקב ראשוני אחר סטודנטים, ובפגישה נוספת קיבלנו הערות ששיפרו את נוחות השימוש, סדר הפעולות והבהירות בזמן אמת. לאורך הדרך נאספו מסמכי דרישות, תרחישי שימוש והגדרות תפקידים, ונבנו מודולים מרכזיים כגון ניהול בחינות, הקצאה אוטומטית, מעקב נוכחות בזמן אמת, טיפול באירועים והודעות והפקת דוחות. בתרגיל בית 3, שהוא התוצר הסופי, הוצגה מערכת מלאה ומתפקדת המדמה ניהול בחינה אמיתי מקצה לקצה, כולל בדיקות והרצות מלאות. בפגישה האחרונה עם יעיל ארז המערכת פעלה באופן רציף וברור, והלקוחה הביעה שביעות רצון גבוהה מהתוצאה ומההתאמה לצרכים בשטח. התהליך כולו לווה בתיעוד מתמשך ובתוצרי ביניים כגון תרשימי אפיון וסיכומי פגישות, כאשר חומרים משלימים מצורפים בנספחים.

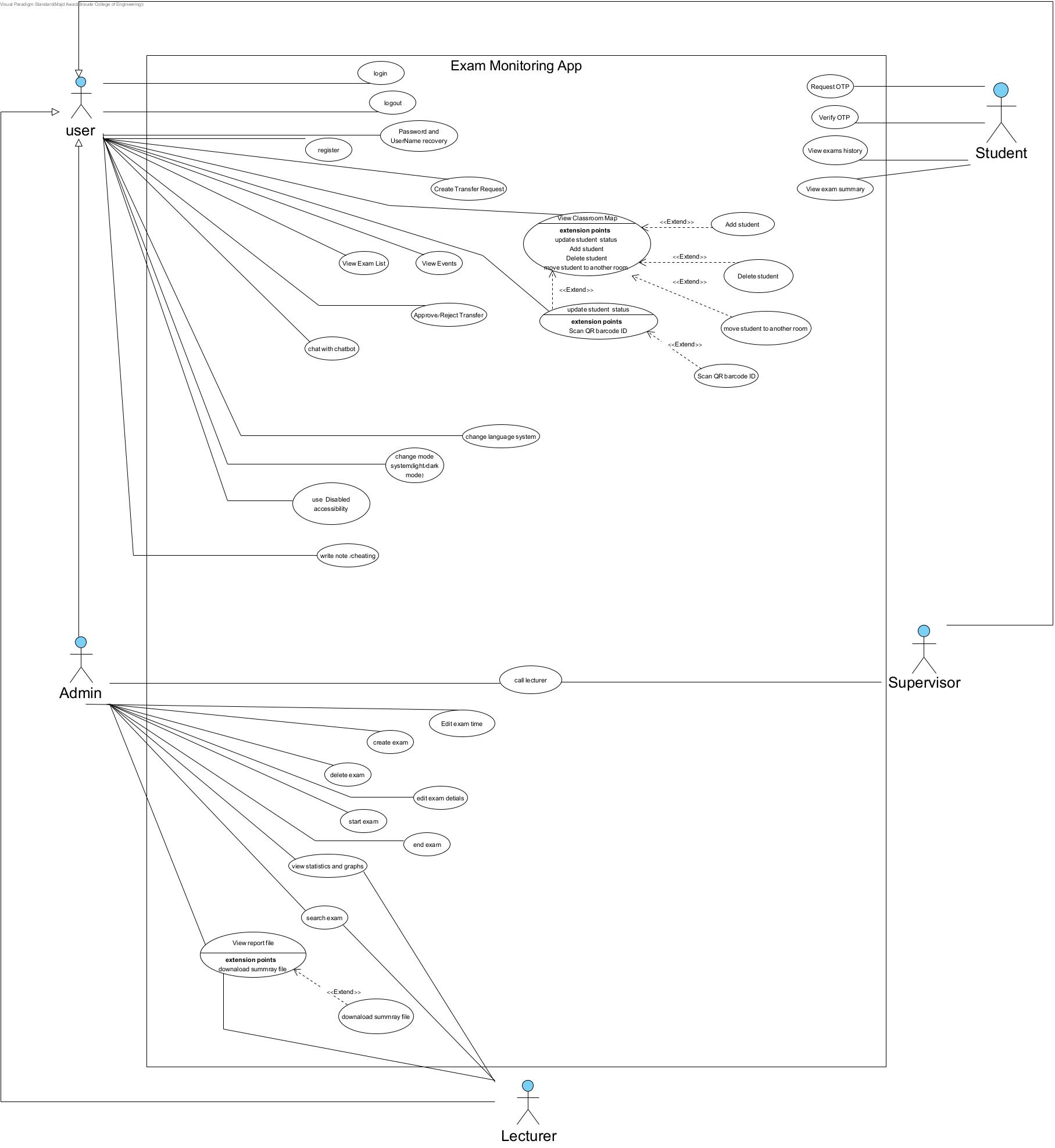
4. דיאגרמות:

תמונה שמכילה טקסט, תרשים, קבלה, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.א.הציגו ארכיטקטורה מעודכנת של האתר (תרשים הכולל את האלמנטים המרכזיים).



ב. הציגו דיאגרמת use case המתארת את השימוש באתר.



5. יש להגיש תוצרי מפגש הסטודיו:  
**א. חישוב של** **ממוצע ציון SUS שניתן לכם. הסבירו. התייחסו בקצרה להערות שקיבלתם.**

ציון ה-SUS הממוצע שהתקבל הוא 88.29 נתון הנחשב לציון "מצוין" הגבוה משמעותית מהממוצע (68). הציון מעיד על כך שהמערכת אינטואיטיבית מאוד, קלה ללמידה ומספקת חוויית משתמש חיובית עבור המשגיחים.

מהערות המשתמשים עולה כי המערכת נוחה יחסית ; המשתמשים שיבחו את העיצוב הנקי, את נוחות השימוש ב-Dashboard ואת ניהול הסטודנטים בזמן אמת. עם זאת, עלו נקודות לשיפור שמתמקדות בעיקר בהתאמה טובה יותר למובייל (כדי לאפשר ניידות מלאה בכיתה), הוספת Dark mode, וצורך בהנגשה מותאמת יותר למשגיחים מבוגרים. לסיכום, המערכת נתפסת כפתרון יעיל ומקצועי, כשהליטוש הנדרש הוא בעיקר טכני וויזואלי.

**ב. מסמך המכיל את תוצרי הסשן של CODEREVIEW שערכתם בכיתה, וכן** **ההתיחסות האישית של כל סטודנט (בטופס גוגל).**

**B14 צוות בודק**

**B18 צוות נבדק**

***חלק ראשון:  עבודה קבוצתית – בדיקת  האתר של הצוות השני***

1. יש לפנות לצוות ולבקש מהם את הקישור לאתר (במידה והאתר טרם נפרס, הצוות ידגים לכם את האתר באחד המחשבים שלהם).
2. כתבו התרשמות כללית של הרצת האתר (ניתן לבסס זאת על ההתנסות בשעה הקודמת):

האתר בכללי נראה ממש טוב ויפה, אהבנו את מפת הכיתה ואיך מציגים את הנתונים

האתר בנוי בצורה מצויינת, האתר מאוד פשוט לשימוש. נוח לשימוש עבור האוכלוסיה אליה מיועד הפרויקט - אנשים שהם חסרי ניסיון טכנולוגי בהכרח, תלמידים ומורים.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. בקשו מהצוות השני את הקישור לגיט ובו הקוד של האתר. רישמו את הקישור:  
   https://github.com/MajdAwad12/Exam\_Monitoring\_App
2. יש לבצע  סקירת קוד ל לקוד של המחברת שהתקבלה, לפי הפרמטרים הבאים  
    (מבוסס על :<https://www.michaelagreiler.com/code-review-checklist-2>)

נא לענות על כל קריטריון ***ולפרט*** :

יש לבצע סקירת קוד לפרויקט שהתקבל לפי הפרמטרים הבאים (מבוסס על: <https://www.michaelagreiler.com/code-review-checklist-2>)

נא לענות על כל קריטריון ולפרט:

| **קריטריון** | **הסבר** |
| --- | --- |
| **מימוש** | האם הקוד מבצע את הנדרש? האם הוא עומד במפרט הפרויקט? פרטו.   **תשובה:כן, הדרישות בוצעו רק צריך להוסיף כפתור " add student"** |
| **יעילות** | האם ניתן לשפר את יעילות הקוד? כיצד? שקלו:  • אופטימיזציה של קריאות API  • יעילות שאילתות בסיס הנתונים  • אופטימיזציה של תמונות  • רינדורים מיותרים ב-React  • גודל ה-Bundle   **תשובה: socket ל server כדי להשאיר אותו זמין כל הזמן** |
| **פשטות** | האם ניתן לפשט את הקוד? כיצד? האם ישנם קומפוננטות מורכבות מדי שניתן לפרק?   **תשובה: קיימות קומפוננטות מורכבות מדי, במיוחד ב- ClassroomMap  שמכילה לוגיקה, State ו־UI רבים. ניתן לפרק אותה לקומפוננטות קטנות יותר** |
| **מודולריות** | האם הקוד מודולרי מספיק? (חלוקה תקינה לקבצים, קומפוננטות, פונקציות, ו-utilities)  • מבנה הקומפוננטות  • ארגון ה-API routes  • הפרדת מודלים של בסיס הנתונים  • פונקציות עזר   **תשובה: באופן כללי כן** |
| **באגים וטעויות** | האם ישנם מקרים בהם הקוד לא מתנהג כצפוי? אילו?  • שגיאות בקונסול  • כשלים ברשת  • מקרי קצה   **תשובה: לא** |
| **טיפול בשגיאות** | • האם ישנן התראות על שגיאות משתמש? כן • האם ישנו שימוש במנגנוני תפיסת שגיאות (try-catch, errorfi   boundaries)? כן • האם הבדיקה ודיבוג הקוד נוחים? כן • האם הודעות המשתמש ברורות וידידותיות? כן • האם מצבי טעינה מטופלים כראוי? כן  **תשובה:** |
| **בדיקות** | • האם ישנן בדיקות יחידה (unit tests)? לא • האם ישנן בדיקות אינטגרציה? לא • אילו בדיקות ניתן להוסיף? צריך בדיקה: בעת הכנסת סטודנט חדש צריך לבדוק אם  הוא כבר קיים או לא • האם ישנה דוחות כיסוי בדיקות? לא  **תשובה:** |
| **שימושיות** | • האם הממשק שימושי (usability)? כן • האם המערכת מעוררת רצון להמשיך להשתמש בה? כן • האם ה-UI רספונסיבי במכשירים שונים? לא • האם הניווט אינטואיטיבי? כן • האם הטפסים קלים למילוי? כן  **תשובה:** |
| **תיעוד** | • האם הקוד מתועד? חלק מהקבצים כן וחלק לא • האם קיים README עם הוראות התקנה? כן • האם נקודות ה-API מתועדות? כן • האם משתני סביבה מתועדים? לא • האם קיים תיעוד פריסה? כן  **תשובה:** |
| **אתיקה ושקיפות** | • האם יש שימוש בנתוני משתמש אשר מפרים פרטיותו? לא • האם ישנו הסבר למשתמש כיצד המערכת מתנהגת? לא • האם ישנו algorithmic bias כנגד קבוצה או פרטים מסוימים? לא • האם נוהלי איסוף הנתונים שקופים? כן  **תשובה:** |
| **אבטחה** | • האם ישנו מידע אבטחה גלוי (שמות משתמש, ססמאות, API keys)? לא   • האם משתני סביבה משמשים כראוי? לא • האם מיושם ניקוי קלט (input sanitization)?  לא • האם אימות והרשאות מיושמים כראוי? באופן כללי כן • האם מחרוזת החיבור ל-MongoDB מאובטחת? כן  **תשובה:** |
| **ביצועים** | • האם שינוי עתידי בקוד יכול לגרום לפגיעה בביצועים? כיצד ניתן לשפר? כן, צריך לשמור תמיד את העבודה שבוצעה עד עכשיו , לאחר מכן לנסות דברים נוספים  • ניתוח זמן טעינת עמוד  • האם מיושם code splitting? כן • האם תמונות מאופטמות? כן • האם נעשה שימוש ב-caching? כן  **תשובה:** |
| **קריאות (Readability)** | • האם הקוד מובן בקלות? כן • אילו חלקים בקוד היו לא ברורים עבורך? החלק שלclassroom map • כיצד ניתן לשפר את קריאות הקוד? נראות טובות • האם כללי שמות עקביים? כן • האם הקוד מעוצב כראוי? כן  **תשובה:** |
| **פריסה (Deployment)** | • האם האתר פרוס בהצלחה? כן • האם תהליך הפריסה מתועד? לא • האם משתני סביבה מוגדרים כראוי? לא • האם החיבור לבסיס הנתונים עובד בסביבת ייצור? כן • האם ישנן בעיות ספציפיות לפריסה? אין תיעוד  **תשובה:** |
| **שיטות עבודה מומלצות React/Next.js** | • האם React hooks משמשים בצורה נכונה? הם עובדים עם node.js • האם ה-routing של Next.js מיושם כראוי?  • האם נעשה שימוש נכון ב-Server Components לעומת Client Components? כן • האם ה-App Router או Pages Router משמש כראוי? כן • האם נעשה שימוש ב-Next.js Image components לאופטימיזציה?כן   **תשובה:** |
| **בסיס נתונים ו-API** | • האם שאילתות ה-MongoDB מאופטמות? כן • האם סכמת בסיס הנתונים מעוצבת היטב? כן • האם ה-API routes מובנים כראוי? כן • האם מיושם אימות נתונים (validation)? כן • האם ישנם אינדקסים מתאימים בבסיס הנתונים? כן  **תשובה:** |

**ה.** האם השתמשתם בכלי AI? איזה? לא

נא לרשום קישור לפרומפט:  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

כיצד היתה החוויה? האם ה AI סייע לכם? אם כן,כיצד? אם לא, מדוע?  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***חלק שני:עבודה  עם הצוות עבורו ביצעתם את הבדיקה***

בצעו סשן של 20 דקות עם הצוות עבורו בדקתם את הקוד, עברו יחד עם נציגי הצוות על ההערות.

רשמו את התגובות להערות שהעליתם.

|  |  |  |
| --- | --- | --- |
| **הערה** | **תגובה** | **האם בכוונת הצוות לבצע שינוי בעקבות ההערה? יש לפרט. אם כן, מה השינוי. אם לא, מדוע** |
| פשטות | כןןןןןןןןןןןןן | לא |
| מימוש | המימוש עומד בדרישות | לא צריך להוסיף כפתור של סטודנט |
| מודלריות | השרת מחולק  controlers, db… וצד משתמש מחולק ל יםםלד | לא המבנה מספיק |
| יעילות | הקוד יעיל ברמה סבירה, אך ניתן לשפר ע"י הפחתת קריאות API מיותרות, ייעול שאילתות לבסיס הנתונים והקטנת רינדורים חוזרים ב-React. |  |
| באגים | אין באגים קריטיים ידועים, אך קומפוננטות גדולות ומודלים עמוסים מגדילים סיכון לטעויות ולקושי בדיבוג. |  |
| טיפול בשגיאות | קיים טיפול בסיסי בשגיאות (try/catch, הודעות), אך ניתן לשפר ע"י טיפול מרכזי בשגיאות, החזרת הודעות עקביות מה־API והפרדה בין לוגיקת שגיאות ל־UI. |  |

**צוות בודק B17**

**צוות נבדק B18**

***חלק ראשון:  עבודה קבוצתית – בדיקת  האתר של הצוות השני***

1. יש לפנות לצוות ולבקש מהם את הקישור לאתר (במידה והאתר טרם נפרס, הצוות ידגים לכם את האתר באחד המחשבים שלהם).
2. כתבו התרשמות כללית של הרצת האתר (ניתן לבסס זאת על ההתנסות בשעה הקודמת):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_האתר עובד בצורה רציפה ומתעדכן בזמן אמת בכל ביצוע פעולה, העיצוב מאוד נוח לשימוש המשתמש גם בלי הכוונה מודרכת. האתר מכיל תוכן רב ופונקציונאליות תקינה.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. בקשו מהצוות השני את הקישור לגיט ובו הקוד של האתר. רישמו את הקישור:<https://github.com/MajdAwad12/Exam_Monitoring_App>
2. יש לבצע  סקירת קוד ל לקוד של המחברת שהתקבלה, לפי הפרמטרים הבאים  
    (מבוסס על :<https://www.michaelagreiler.com/code-review-checklist-2>)

נא לענות על כל קריטריון ***ולפרט*** :

יש לבצע סקירת קוד לפרויקט שהתקבל לפי הפרמטרים הבאים (מבוסס על: <https://www.michaelagreiler.com/code-review-checklist-2>)

נא לענות על כל קריטריון ולפרט:

| **קריטריון** | **הסבר** |
| --- | --- |
| **מימוש** | האם הקוד מבצע את הנדרש? האם הוא עומד במפרט הפרויקט? פרטו.   **תשובה: כן, הקוד מבצע את הנדרש על פי ההנחיות שניתנו לפרויקט.** |
| **יעילות** | האם ניתן לשפר את יעילות הקוד? כיצד? שקלו:  • אופטימיזציה של קריאות API  • יעילות שאילתות בסיס הנתונים  • אופטימיזציה של תמונות  • רינדורים מיותרים ב-React  • גודל ה-Bundle   **תשובה:ניתן לשפר את היעילות. אין שימוש ב- useMemo עבור קומפוננטות כבדות.** |
| **פשטות** | האם ניתן לפשט את הקוד? כיצד? האם ישנם קומפוננטות מורכבות מדי שניתן לפרק?   **תשובה:  ManageExamsPage.jsx מורכב מדי וצריך לפרק אותו לקומפוננטות קטנות יותר** |
| **מודולריות** | האם הקוד מודולרי מספיק? (חלוקה תקינה לקבצים, קומפוננטות, פונקציות, ו-utilities)  • מבנה הקומפוננטות  • ארגון ה-API routes  • הפרדת מודלים של בסיס הנתונים  • פונקציות עזר   **תשובה: הקוד מודולרי ברמה טובה. יש הפרדה ברורה בין client ל-server.** |
| **באגים וטעויות** | האם ישנם מקרים בהם הקוד לא מתנהג כצפוי? אילו?  • שגיאות בקונסול  • כשלים ברשת  • מקרי קצה   **תשובה: לא זיהינו באגים בזמן הבדיקה.** |
| **טיפול בשגיאות** | • האם ישנן התראות על שגיאות משתמש?  • האם ישנו שימוש במנגנוני תפיסת שגיאות (try-catch, error boundaries)?  • האם הבדיקה ודיבוג הקוד נוחים?  • האם הודעות המשתמש ברורות וידידותיות?  • האם מצבי טעינה מטופלים כראוי?   **תשובה: יש טיפול טוב בשגיאות. כל הקונטרולרים עטופים ב-try-catch ומחזירים הודעות שגיאה ברורות.** |
| **בדיקות** | • האם ישנן בדיקות יחידה (unit tests)?  • האם ישנן בדיקות אינטגרציה?  • אילו בדיקות ניתן להוסיף?  • האם ישנה דוחות כיסוי בדיקות?   **תשובה: אין** |
| **שימושיות** | • האם הממשק שימושי (usability)?  • האם המערכת מעוררת רצון להמשיך להשתמש בה?  • האם ה-UI רספונסיבי במכשירים שונים?  • האם הניווט אינטואיטיבי?  • האם הטפסים קלים למילוי?   **תשובה: הממשק שימושי ונוח. עיצוב מודרני עם Tailwind CSS.** |
| **תיעוד** | • האם הקוד מתועד?  • האם קיים README עם הוראות התקנה?  • האם נקודות ה-API מתועדות?  • האם משתני סביבה מתועדים?  • האם קיים תיעוד פריסה?   **תשובה: קיים רדמי מפורט עדכני, מתועד כראוי, הכל מתועד** |
| **אתיקה ושקיפות** | • האם יש שימוש בנתוני משתמש אשר מפרים פרטיותו?  • האם ישנו הסבר למשתמש כיצד המערכת מתנהגת?  • האם ישנו algorithmic bias כנגד קבוצה או פרטים מסוימים?  • האם נוהלי איסוף הנתונים שקופים?   **תשובה: אין שימוש לרעה בנתונים.** |
| **אבטחה** | • האם ישנו מידע אבטחה גלוי (שמות משתמש, ססמאות, API keys)?  • האם משתני סביבה משמשים כראוי?  • האם מיושם ניקוי קלט (input sanitization)?  • האם אימות והרשאות מיושמים כראוי?  • האם מחרוזת החיבור ל-MongoDB מאובטחת?   **תשובה: המונגוס מספק הגנה בסיסית** |
| **ביצועים** | • האם שינוי עתידי בקוד יכול לגרום לפגיעה בביצועים? כיצד ניתן לשפר?  • ניתוח זמן טעינת עמוד  • האם מיושם code splitting?  • האם תמונות מאופטמות?  • האם נעשה שימוש ב-caching?   **תשובה: כל האפליקציה נטענת ביחד, ניתן לעשות יותר lazy loading** |
| **קריאות (Readability)** | • האם הקוד מובן בקלות?  • אילו חלקים בקוד היו לא ברורים עבורך?  • כיצד ניתן לשפר את קריאות הקוד?  • האם כללי שמות עקביים?  • האם הקוד מעוצב כראוי?   **תשובה: הקוד קריא ומובן. שמות משתנים ופונקציות ברורים. יש הערות מפרידות בין סקשנים. ארגון קבצים הגיוני.** |
| **פריסה (Deployment)** | • האם האתר פרוס בהצלחה?  • האם תהליך הפריסה מתועד?  • האם משתני סביבה מוגדרים כראוי?  • האם החיבור לבסיס הנתונים עובד בסביבת ייצור?  • האם ישנן בעיות ספציפיות לפריסה?   **תשובה: הפרויקט מוכן לפריסה עם קובצי vercel.json. יש README עם הוראות npm install והרצה.** |
| **שיטות עבודה מומלצות React/Next.js** | • האם React hooks משמשים בצורה נכונה?  • האם ה-routing של Next.js מיושם כראוי?  • האם נעשה שימוש נכון ב-Server Components לעומת Client Components?  • האם ה-App Router או Pages Router משמש כראוי?  • האם נעשה שימוש ב-Next.js Image components לאופטימיזציה?   **תשובה: הפרויקט משתמש ב-React+Vite ולא ב- נקסט**  **React Hooks משומשים נכון** |
| **בסיס נתונים ו-API** | • האם שאילתות ה-MongoDB מאופטמות?  • האם סכמת בסיס הנתונים מעוצבת היטב?  • האם ה-API routes מובנים כראוי?  • האם מיושם אימות נתונים (validation)?  • האם ישנם אינדקסים מתאימים בבסיס הנתונים?   **תשובה: יש indexes על**  **Exam  - status, startAt, lecturer.id, supervisors.id. סכמת הנתונים מעוצבת היטב עם Mongoose schemas מפורטים** |

**ה.** האם השתמשתם בכלי AI? איזה?\_\_gemini\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

נא לרשום קישור לפרומפט:  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_https://gemini.google.com/share/a4c615edf051\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

כיצד היתה החוויה? האם ה AI סייע לכם? אם כן,כיצד? אם לא, מדוע?  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_כלי הai תמצת לנו את עקרונות הcode review, דבר שעזר לנו במעקב אחר הקוד והעברת ביקורת בונה\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***חלק שני:עבודה  עם הצוות עבורו ביצעתם את הבדיקה***

בצעו סשן של 20 דקות עם הצוות עבורו בדקתם את הקוד, עברו יחד עם נציגי הצוות על ההערות.

רשמו את התגובות להערות שהעליתם.

|  |  |  |
| --- | --- | --- |
| **הערה** | **תגובה** | **האם בכוונת הצוות לבצע שינוי בעקבות ההערה? יש לפרט. אם כן, מה השינוי. אם לא, מדוע** |
| פשטות | הקוד יחסית ברור, אבל בחלק מהקומפוננטות יש הרבה אחריות באותו קובץ (למשל מסכים של active-exam). היה כדאי להעביר ל־utils/hooks חלק מהלוגיקה. | כן, הצוות שביקר אותנו ממליץ על חלוקה נכונה של קבצים בפרונט, עלינו לעבוד בצורה יותר נכונה ארכיטקטורית בנוגע למבנה הפרויקט. |
| מימוש | כן,המערכת בנויה לפי הדרישות של “Exam Watch” – יש התחברות, ניהול מבחנים, הפעלת מבחן, נוכחות, דוחות ובוט עוזר. יש הפרדה טובה בין מסכים ל־API. | אין שינוי |
| מודלריות | אין תגובה מצד הצוות המבקר | NA |
| יעילות | סה״כ יעיל לפרויקט לימודי. עם זאת יש מקומות שאפשר לשפר:  אם יש קריאה לקבצי CSV או DB בכל רינדור/טעינה — עדיף caching / טעינה פעם אחת  בעמודים כבדים עדיף לפצל קומפוננטות ולהימנע מרינדורים חוזרים. | כן, נתייחס לביקורת עקב פיצול הקומפוננטות ולגבי הטעינה מחדש אנחנו ככל הנראה לא נעשה שינוי כי הטעינה הכרחית |
| באגים | לא ראיתי באגים “ברורים” ישר מהעין. | למרות שלא נמצאו באגים, נרצה לעבור על המערכת שלנו ולאתר כמה שיותר באגים |
| טיפול בשגיאות | יש טיפול בשגיאות בחלק מה־routes (סטטוסים 401/400 וכו’), אבל בצד לקוח לא תמיד ברור מה המשתמש אמור לעשות עם התקלה. מומלץ להוסיף הודעות Toast/Alert עקביות בכל פעולה. | כן נעבוד על שימוש בהצגת שגיאות רלוונטיות למשתמש והכוונה לפי העקרונות של שניידרמן |

6. יש להציג מבנה סופי של האתר שלכם:

**נדרשת בכל פרויקט פריסה מלאה (deploment) של הפרויקט! ב - vercel.  
לא יתקבלו הגשות של קבצים או אתר ב -localhost .**

**טכנולוגיות**:    להלן המרכיבים הטכנולוגיים המומלצים לשימוש בפרויקט:

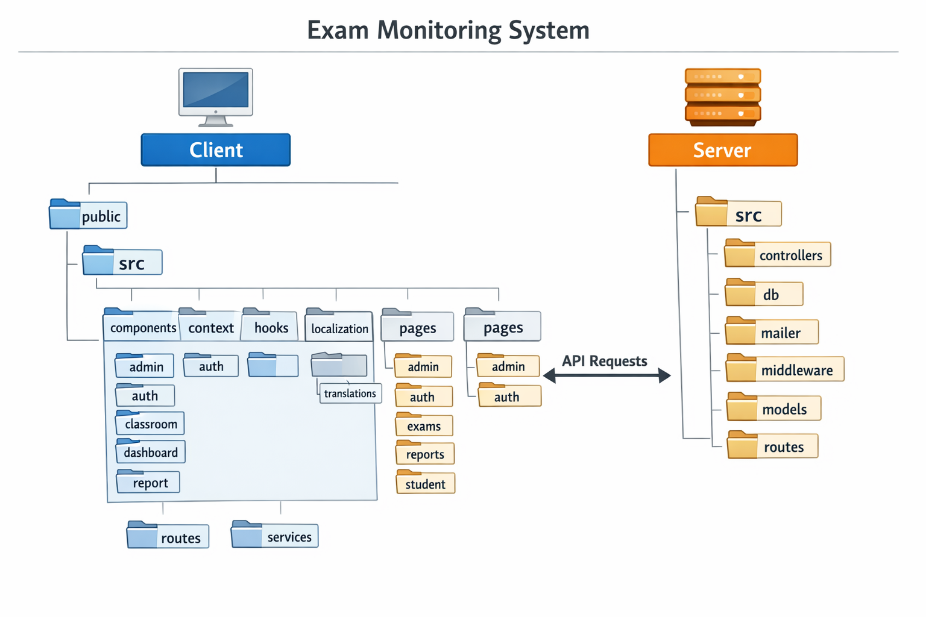
·           - front-end: React/Preact/Next with Tailwind

    - back-end:

    - option 1 - remote services/APIs

    - option 2 - node js/express deployed on remote web server

**א. האתר ימומש ב -React/Next, וכן שימוש ב Tailwind - נא להציג דיאגרמה המתארת את התיקיות והקבצים השונים. יש לפרט את הקומפוננטות השונות.**

****

**חלוקת קומפוננטות Components :**

1. **Layout :**

: components/layout/AppLayout.jsx- עוטף את הדפים עם Sidebar/Topbar.

: components/layout/Sidebar.jsx- תפריט לפי Role (admin/lecturer/supervisor).

: components/layout/Topbar.jsx- כותרת/פעולות עליונות.

1. **Navigation :**

: components/navigation/LanguageDropdown.jsx, LanguageSwitcherButton.jsx- בחירת שפה + RTL/LTR.

: components/common/ThemeToggle.jsx, AccessibilityWidget.jsx – מצב כהה/בהיר ונגישות.

1. **Auth (Login/Register) :**

: components/auth/LoginHeader.jsx, RegisterHeader.jsx- כותרות/לוגו.

: LoginForm.jsx, RegisterForm.jsx- טפסים.

: LoginCard.jsx, AuthFooter.jsx, ErrorAlert.jsx – UI- תומך.

-שירות API : services/auth.service.js

1. **Dashboard :**

: components/dashboard/ExamOverviewCard.jsx- תקציר מבחן.

: ActiveClassroomsTabs.jsx, RoomTabs.jsx, ExamTabs.jsx- ניווט פנימי (כיתות/חדרים/מבחנים).

: EventsFeed.jsx- אירועים violations/incidents

: SmartExamClock.jsx- שעון/אחוז התקדמות/סטטוס ריצה.

: TransfersPanel.jsx- בקשות העברה.

: DashboardAddDeleteStudentsCard.jsx- הוספה/מחיקה של סטודנטים.

-שירותים: dashboard.service.js, transfers.service.js, incidents.service.js, messages.service.js,  
וגם dashboard.ADD.DELETE.Students.service.js

Hook- מרכזי: hooks/useDashboardLive.js: מבצע polling/שליפות לעדכון המסך

1. **Classroom :**

: ClassroomMap.jsx – מצייר מפת כיתה לפי rows/cols.

: SeatCard.jsx – כרטיס מושב + צבע/סטטוס (present/temp\_out/moving…)

: SeatActionsModal.jsx – פעולות על סטודנט במושב.

: QrScanner.jsx – QR - לנוכחות

 utils.js – פונקציות עזר (סטטוסים, פורמט זמן, וכו’)

1. **Reports :**

**יצירה/הורדה:**

report/ExamReportPDF.jsx (PDF) -

report/ExamReportEXCEL.js (Excel) - באמצעות xlsx פורמט

**: UI Sections:**

ReportsHeaderSection.jsx, ReportsControlsSection.jsx, ReportsKpisSection.jsx,  
ReportsChartsSection.jsx, ReportsDetailsSection.jsx, ReportsErrorBanner.jsx

-שירות API: services/reports.service.js :

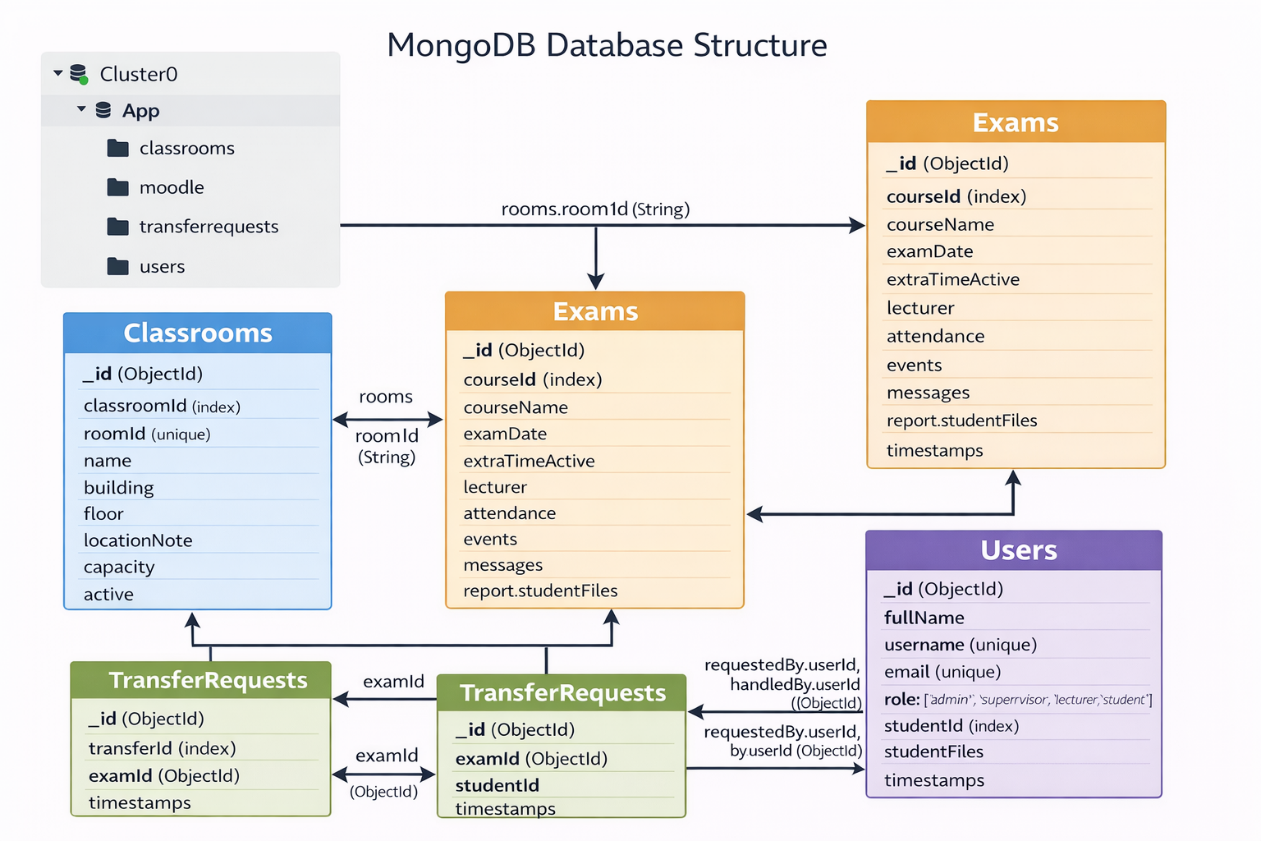
- דף pages/reports/ReportsPage.jsx :

- דוח סטודנטpages/student/StudentMyExamReportPage.jsx :

1. **Chat (AI Assistant) :**

* components/chat/FloatingChatWidget.jsx
* שירות services/chat.service.js :

**ב. יש לפרט את פריטי המידע - יש להשתמש במידע אמיתי ורלוונטי לפרויקט שלכם (בשליפה ממסד נתונים חיצוני או מ - API). יש להראות דיאגרמת מבנה DB.**

****

7. יש להגיש תיק למתכנת:

א. יש לתעד ולהסביר את כל הפונקציות המרכזיות בקוד שלכם.

ב. יש לפרט:

* את כל ה -API שהשתמשתם בהם,
* קישור ל- DB,
* סביבות מיוחדות שהתקנתם,
* קטעי קוד מיוחדים שלקחתם ממקורות שונים (כולל רפרנסים) ,
* פרומפטים שהשתמשתם בהם בכלי AI
* שמות משתמש וססמאות שהגדרתם לכל סוג משתמש, לצורך חיבור לאתר.

**תיק מתכנת client**

**תיאור כללי**

צד ה-Client בפרויקט נבנה באמצעות הטכנולוגיות הבאות:

* React.js – בניית ממשק משתמש דינמי
* Axios – שליחת בקשות HTTP לשרת
* React Router DOM – ניהול ניווט בין עמודים
* Context API – ניהול State גלובלי (משתמש מחובר)
* Bootstrap / CSS מותאם אישית – עיצוב המערכת

**תפקיד ה-Client במערכת**

ה-Client אחראי על:

* הצגת מסכי התחברות והרשמה
* ניהול Session בצד משתמש
* שליחת בקשות ל-Server דרך REST API
* הצגת נתוני בחינות, כיתות ודוחות
* ניהול טיימר בחינה בצד משתמש
* הצגת התראות בזמן אמת
* הצגת ממשק שונה לפי Role:
  + Admin
  + Lecturer
  + Supervisor
  + Student

**API .1 בהם נעשה שימוש**

|  |
| --- |
| POST /api/auth/login | התחברות משתמש  Staff (admin / lecturer / supervisor) |
| POST /api/auth/logout | יציאה מהמערכת והשמדת  session |
| GET /api/auth/me | בדיקה אם קיים משתמש מחובר והחזרת פרטיו |
| POST /api/auth/student/request-otp | שליחת קוד  OTP לסטודנט לצורך התחברות |
| POST /api/auth/student/verify-otp | אימות קוד  OTP ויצירת session |
| GET /api/exams | שליפת רשימת בחינות מהשרת |
| POST /api/exams | יצירת בחינה חדשה |
| PUT /api/exams/:id | עדכון פרטי בחינה קיימת |
| DELETE /api/exams/:id | מחיקת בחינה מהמערכת |
| GET /api/transfers | שליפת בקשות העברה בין חדרים |
| POST /api/transfers | יצירת בקשת העברה חדשה |
| PUT /api/transfers/:id | אישור או דחיית בקשת העברה |
| GET /api/incidents | שליפת אירועים חריגים |
| POST /api/incidents | יצירת דיווח אירוע במהלך בחינה |
| GET /api/dashboard | שליפת נתונים סטטיסטיים ללוח הבקרה |

**2. קישור למסד הנתונים (Database)**

ה-Client אינו מחובר ישירות ל-Database.

התקשורת מתבצעת דרך ה-Server בלבד.

שרת מחובר ל:

MongoDB Atlas (Cloud)

**3.** **סביבות עבודה מיוחדות**

Node.js

npm

React

Axios

React Router

Bootstrap

**4.** **קטעי קוד ממקורות חיצוניים (רפרנסים)**

בחלקים מסוימים נעשה שימוש בדוגמאות קוד נפוצות:

**Axios:**

<https://axios-http.com/docs/intro>

**React Router:**

https://reactrouter.com/en/main

**Bootstrap:**

<https://getbootstrap.com/>

**ניהול useEffect וטיימר:**

מבוסס על תיעוד רשמי של React Hooks.

**Local Storage / Session Handling:**

MDN Web Docs:  
<https://developer.mozilla.org/>

**5.** **שימוש בכלי AI**

במהלך פיתוח צד ה-Client נעשה שימוש בכלי AI לצורך:

* בניית קומפוננטות React
* תיקון שגיאות State
* שיפור מבנה פרויקט
* מימוש טיימר בחינה
* טיפול ב-async/await
* בניית Protected Routes

**דוגמאות לפרומפטים:**

"Create React login form with session authentication"

"How to protect routes in React based on user role?"

"Create exam countdown timer in React using useEffect"

"How to send axios request with cookies?"

**6. שמות משתמש וסיסמאות שהוגדרו**

שמות משתמש וסיסמאות לצורך בדיקה:

| Role | Username/email for student | Password | Student id |
| --- | --- | --- | --- |
| Admin | admin | Admin123 |  |
| Supervisor | leo.zein.1005 | Pass123 |  |
| Lecturer | dana.williams.1210 | Pass123 |  |
| Student | exam\_monitoirng\_app@yahoo.com | web2026application123 | 220000059 |

**תיעוד קובץclient/src/main.jsx**

הקובץ main.jsx הוא נקודת הכניסה הראשית (Entry Point) של אפליקציית ה-React בצד ה-Client.

תפקידו:

אתחול אפליקציית React

טעינת הגדרות גלובליות

חיבור האפליקציה ל־DOM

עטיפת האפליקציה ב־Providers גלובליים (Theme / Context)

הפעלת StrictMode לצורכי פיתוח

**ספריות מיובאות**

**React**

משמש לבניית קומפוננטות וניהול ממשק משתמש.

**ReactDOM**

אחראי לחיבור אפליקציית React לעמוד ה-HTML.

**i18nConfiguration**

טעינת מערכת ריבוי שפות (Internationalization).  
מגדיר תרגומים ושפת ברירת מחדל.

**App**

הקומפוננטה הראשית של המערכת.  
מכילה את ה-Routing ואת כל מבנה הדפים.

**ThemeProvider**

Context גלובלי לניהול ערכת נושא (Dark / Light Mode).  
מאפשר שינוי עיצוב בכל האפליקציה.

**הפונקציה המרכזית בקובץ**

יצירת Root וחיבור לאפליקציה

**הסבר מפורט**

**document.getElementById("root")**

מחפש את האלמנט עם id="root" בקובץ index.html.

זהו המיכל שאליו React יוזרק.

**ReactDOM.createRoot(...)**

יוצר Root חדש לפי React 18.

מאפשר Concurrent Rendering וביצועים משופרים.

**<React.StrictMode>**

כלי פיתוח שעוזר לזהות:

שימוש לא תקין ב-Lifecycle

תלויות חסרות ב-useEffect

בעיות עתידיות ב-React

**<ThemeProvider>**

עוטף את כל האפליקציה ומאפשר:

גישה ל־Theme Context מכל קומפוננטה

שינוי ערכת נושא גלובלי

שמירת בחירת משתמש

**<App />**

הקומפוננטה הראשית של המערכת.

מכילה:

React Router

Protected Routes

טעינת נתוני משתמש

דפי המערכת

**תיעוד קובץclient/src/App.jsx**

**תיאור כללי**

הקובץ App.jsx הוא הקומפוננטה הראשית של צד ה־Client.

תפקידו:

* חיבור מערכת ה־Routing לאפליקציה
* טעינת מבנה הנתיבים המרכזי
* הפעלת React Router במצב Data Router (גרסה מתקדמת)

**export default function App()**

קומפוננטה פונקציונלית שמחזירה את RouterProvider.

אין בה State או useEffect, משום שהאחריות שלה היא אך ורק אתחול מערכת הניווט.

**אחריות הקובץ בארכיטקטורה**

App.jsx מהווה:

* שכבת חיבור בין מערכת React לבין מערכת הנתיבים
* נקודת הפעלה של React Router
* גשר בין main.jsx לבין router.jsx

**תיאור לתיקיית routes**

תיקיית routes אחראית על ניהול מערכת הניווט (Routing System) של האפליקציה בצד ה־Client.

התיקייה מגדירה:

* מבנה ה־ URL של המערכת
* חלוקת הנתיבים לציבוריים ומוגנים
* ניהול הרשאות לפי Role
* טיפול בשגיאות ניווט
* ניתוב חכם לפי סוג המשתמש המחובר

המערכת מבוססת על  
React Router v6 (Data Router API) באמצעות createBrowserRouter.

**מטרת מערכת ה־Routing**

מערכת הנתיבים אחראית על:

1. מעבר בין דפים ללא רענון דפדפן (SPA)
2. שמירה על מבנה URL תקין וברור
3. הפרדה בין:
   * דפים ציבוריים (Home, Login, Register)
   * דפים מוגנים (/app)
4. ניהול הרשאות לפי Role:
   * Admin
   * Lecturer
   * Supervisor
   * Student
5. מניעת גישה לא מורשית דרך שינוי ידני של ה־URL

**עקרונות ארכיטקטורה**

**מערכת ה־Routing בנויה לפי העקרונות הבאים:**

Separation of Concerns – לוגיקת ניווט מופרדת מהקומפוננטות

Nested Layout Architecture – Layout אחד לכל אזור מוגן

Role Based Access Control (RBAC)

UX חכם באמצעות Redirect דינמי

Fallback פנימי וחיצוני

**router.jsx**

הקובץ המרכזי שמגדיר את כל נתיבי המערכת.

אחראי על:

* יצירת ה־Router
* הגדרת Nested Routes
* קישור ל־Layout הראשי
* ניתוב חכם לפי Role
* Fallback גלובלי

**AppFallback()**

תפקיד:  
ביצוע ניתוב חכם כאשר המשתמש נכנס ל־/app ללא תת־נתיב או לנתיב לא קיים.

לוגיקה:

* אם אין משתמש מחובר (me לא קיים) → הפניה ל־/login
* אם המשתמש הוא student → הפניה ל־/app/student
* אחרת → הפניה ל־/app/dashboard

משתמש ב־useOutletContext() כדי לקבל את המשתמש מה־Layout.

**createBrowserRouter([...])**

תפקיד:  
יצירת אובייקט Router מרכזי שמגדיר את כל נתיבי המערכת.

אחראי על:

* נתיבים ציבוריים (/, /login, /register)
* נתיבים מוגנים תחת /app
* Nested Routes
* טיפול בשגיאות (errorElement)
* Fallback גלובלי

**RoleGate.jsx**

קומפוננטת אבטחה (Client-side Middleware).

אחראית על:

* בדיקת הרשאות משתמש
* חסימת גישה לנתיבים מוגנים
* ביצוע Redirect במקרה של חוסר הרשאה

**RoleGate({ allow, children, fallback })**

**תפקיד כללי:**  
קומפוננטת אבטחה (Client-side Middleware) המגנה על נתיבים תחת /app לפי Role של המשתמש.

הקומפוננטה בודקת:

1. האם מצב המשתמש עדיין בטעינה
2. האם המשתמש מחובר
3. האם למשתמש יש הרשאה לגשת לנתיב

**RouteError.jsx**

קומפוננטת שגיאה.

מוצגת כאשר:

* נתיב נכשל בטעינה
* מתרחשת שגיאה ב־Loader (אם קיים)
* שגיאת ניווט

**RouteError()**

**תפקיד כללי:**  
קומפוננטת שגיאה גלובלית עבור React Router.  
מוצגת כאשר מתרחשת שגיאה בטעינת נתיב או בזמן ניווט בתוך /app.

הקומפוננטה משמשת כ־errorElement ב־Router.

**תיעוד תיקיית components**

**תיאור כללי**

תיקיית components אחראית על כל רכיבי הממשק (UI Components) של המערכת.

הרכיבים מחולקים לפי תחומים פונקציונליים, בהתאם לעקרון:

Separation of Concerns

כל תיקייה מייצגת אזור לוגי במערכת (Admin, Auth, Chat )

**תפקיד תיקיית components במערכת**

התיקייה אחראית על:

* בניית ממשק המשתמש (UI)
* יצירת קומפוננטות לשימוש חוזר
* חלוקה לאזורים לפי תפקידים במערכת
* הפרדה בין לוגיקה ויזואלית ללוגיקה עסקית
* שמירה על קוד מודולרי וקריא

**components/admin**

**תיאור כללי**

תיקיית admin מכילה קומפוננטות ייעודיות למשתמש מסוג Admin במערכת.

הרכיבים בתיקייה זו אחראים על:

* ניהול בחינות
* יצירה ועריכה של בחינות
* פעולות מערכת מתקדמות
* הצגת חלונות מודאליים (Modals) לביצוע פעולות CRUD

גישה לקומפוננטות אלו מוגבלת באמצעות RoleGate ל־admin בלבד.

**CreateExamModal.jsx**

קומפוננטת Modal ליצירת בחינה חדשה.

אחראית על:

* הצגת טופס יצירה
* קבלת נתונים מהמשתמש
* שליחת POST ל־/api/exams
* סגירת חלון לאחר הצלחה
* רענון רשימת הבחינות

**CreateExamModal()**

**תפקיד כללי:**  
קומפוננטת Modal רב-שלבית (Multi-Step Wizard) ליצירת בחינה חדשה ע"י Admin.

הקומפוננטה אחראית על:

* הזנת פרטי בחינה
* ביצוע Auto-Assignment (אופציונלי)
* ניהול והוספת חדרי בחינה
* שליחת נתונים לשרת ליצירת הבחינה

מבוססת על ModalUI כרכיב תצוגה חיצוני.

**EditExamModal.jsx**

קומפוננטת Modal לעריכת בחינה קיימת.

אחראית על:

* טעינת נתוני בחינה קיימת
* הצגת טופס עם ערכים קיימים
* שליחת PUT ל־/api/exams/:id
* עדכון נתונים בממשק

**EditExamModal()**

**תפקיד כללי:**  
קומפוננטת Modal לעריכת בחינה קיימת ע"י Admin.

הקומפוננטה מאפשרת:

* עדכון פרטי בחינה
* ניהול חדרי בחינה
* שיוך מרצה
* שיוך משגיחים לחדרים
* מחיקת בחינה
* שמירת שינויים

מבוססת על ModalUI כרכיב עיצובי משותף.

**ModalUI.jsx**

רכיב תצוגה כללי (Reusable Modal).

אחראי על:

* עטיפת תוכן בחלון מודאלי
* ניהול פתיחה וסגירה
* רקע כהה (overlay)
* מבנה עיצובי אחיד לכל המודאלים

משמש בסיס גם ל־Create וגם ל־Edit.

**ModalUI()**

**תפקיד כללי**:  
רכיב Modal כללי (Reusable UI Component) המשמש להצגת חלונות קופצים במערכת.הרכיב משמש בסיס ל־:

* CreateExamModal
* EditExamModal
* וכל מודאל נוסף במערכת

**components/chat**

תפקיד כללי:  
רכיב צ'אט צף (Floating Chat Widget) המאפשר למשתמשים — במיוחד Supervisor בזמן בחינה — לשאול שאלות ולקבל תשובות מ־AI.

הרכיב כולל:

* פתיחה/סגירה כ־Floating Bubble
* שאלות מהירות מוכנות מראש
* הזנת טקסט חופשי
* חיווי "Thinking"
* גלילה אוטומטית
* ניקוי שיחה

**sendText(text)**

שולחת הודעה ל־AI.

שלבים:

1. ניקוי הטקסט
2. הוספת הודעת משתמש ל־messages
3. חישוב זמן "חשיבה" מדומה (calcThinkingMs)
4. קריאה ל־chatWithAI
5. הוספת תשובת Bot
6. טיפול בשגיאה במקרה כשל

**send()**

פונקציה עטיפה לשליחת הטקסט מתוך שדה הקלט.

**clearChat()**

מאפס את השיחה ומחזיר להודעת פתיחה ראשונית.

**toggleQuick()**

פותח/סוגר את אזור השאלות המהירות.

שומר את המצב ב־localStorage כדי לזכור העדפת משתמש.

**components/loading**

**תפקיד כללי:**  
קומפוננטת Loader מלאה למסך (Full-Screen Center Loader) המוצגת בזמן טעינת נתונים במערכת.

הרכיב כולל:

* אנימציית מערכת מסלול (Orbit System)
* כוכבים ורקע דינמי
* טקסט טעינה
* פסי Shimmer
* אנימציית נקודות
* תמיכה ב־Dark Mode
* תמיכה ב־Reduced Motion (Accessibility)

**components/common**

תיקייה זו מכילה רכיבים כלליים (Reusable Components)  
שאינם קשורים לאזור פונקציונלי ספציפי כמו Admin או Chat.

הרכיבים כאן:

* ניתנים לשימוש חוזר בכל המערכת
* משפרים חוויית משתמש
* מוסיפים נגישות ושליטה בעיצוב
* מהווים שכבת תשתית UI כללית

**ThemeToggle.jsx**

**ThemeToggle({ className })**

**תפקיד כללי:**  
רכיב כפתור המאפשר מעבר בין מצב תצוגה בהיר (Light Mode) לכהה (Dark Mode) במערכת.

הרכיב משתמש ב־Theme Context כדי לשנות את ערכת הנושא באופן גלובלי בכל האפליקציה.

**AccessibilityWidget.jsx**

**AccessibilityWidget({ placement, tone, className })**

**תפקיד כללי:**  
רכיב נגישות פנימי (In-App Accessibility Panel) המאפשר למשתמש להתאים את חוויית השימוש במערכת לפי צרכים אישיים.

הרכיב:

* אינו משתמש בספריות חיצוניות
* מוסיף מחלקות CSS על <html>
* שומר העדפות ב־localStorage
* תומך ב־RTL
* תומך במיקום Topbar או Floating

**components/navigation**

תיקייה זו כוללת רכיבי ניווט הקשורים לשפה וללוקליזציה של המערכת.

הרכיבים כאן:

* מאפשרים מעבר בין שפות
* משנים כיוון תצוגה (LTR / RTL)
* עובדים יחד עם react-i18next
* משפיעים גלובלית על האפליקציה

**LanguageSwitcherButton.jsx**

**LanguageSwitcherButton()**

**תפקיד כללי:**  
רכיב כפתורים (Toggle Style) להחלפת שפה בין עברית (HE) לאנגלית (EN).

הרכיב מספק דרך מהירה וברורה להחלפת שפה, ללא שימוש ב־Dropdown.

**אחריות הרכיב**

* קריאה לשפה הנוכחית דרך useTranslation()
* החלפת שפה באמצעות i18n.changeLanguage()
* עדכון כיוון מסמך (document.documentElement.dir)
* עדכון lang ב־HTML
* שמירת ההעדפה (במידה ומוגדר)

**LanguageDropdown.jsx**

**LanguageDropdown()**

**תפקיד כללי:**  
רכיב Dropdown לבחירת שפה במערכת (עברית / אנגלית).

הרכיב מחובר ל־react-i18next ומאפשר שינוי שפה גלובלי בזמן אמת.

**אחריות הרכיב**

* הצגת רשימת שפות זמינות
* סימון השפה הפעילה
* שינוי שפה בלחיצה
* ניהול פתיחה/סגירה של התפריט
* תמיכה ב־RTL

**components/homepage**

תיקיית homepage מכילה את רכיבי הממשק של דף הבית הציבורי של המערכת.

זהו הדף הראשון שמשתמש רואה כאשר הוא נכנס לאתר (/).

העמוד מיועד ל:

* הצגת מיתוג המערכת
* מתן מידע כללי
* גישה ל־Login ו־Register
* הצגת פרטי יצירת קשר
* שליטה בשפה, נגישות ותצוגה

דף זה אינו דורש התחברות ולכן נחשב Public Page.

**Header.jsx**

**תפקיד כללי**

רכיב Header ראשי של דף הבית.

מהווה את אזור הניווט העליון של האתר.

**אחראי על:**

* לוגו ומיתוג
* תיאור קצר של המערכת
* כפתורי התחברות והרשמה
* שליטה בשפה (LanguageDropdown)
* שינוי ערכת נושא (ThemeToggle)
* רכיב נגישות (AccessibilityWidget)
* הצגת פרטי Helpdesk (במסכים גדולים)

**Footer.jsx**

**תפקיד כללי**

רכיב תחתון של דף הבית.

מציג מידע משלים וקישורי יצירת קשר.

**אחראי על:**

* מידע מותג
* תיאור קצר של המערכת
* פרטי Helpdesk
* קישורים לרשתות חברתיות
* זכויות יוצרים דינמיות לפי שנה

**components/classroom**

תיקיית classroom מכילה את כל הרכיבים הקשורים לניהול והצגת חדר בחינה בזמן אמת.

האזור הזה מייצג את סביבת הבחינה בפועל – כלומר:

* מפת כיתה
* מושבים
* סטטוס תלמידים
* פעולות משגיח
* אינטראקציה עם QR

זהו חלק ליבה במערכת, בעיקר עבור Supervisor בזמן בחינה.

**ClassroomMap.jsx**

ClassroomMap.jsx הוא הרכיב המרכזי של סביבת הבחינה בזמן אמת.

זהו הקומפוננטה הראשית שמנהלת:

* הצגת מפת הכיתה (5×5)
* סטטוס תלמידים לפי מושב
* רשימת תלמידים בחדר
* פעולות משגיח
* ניהול העברות בין חדרים
* סריקת QR
* רישום חריגות
* עדכוני נוכחות בזמן אמת

**patchStatus(studentId, patch)**

הפונקציה המרכזית לעדכון סטטוס תלמיד.

* עושה **optimistic update**
* שולחת עדכון לשרת (updateAttendance)
* מרעננת נתונים
* מטפלת בשגיאות

כל שינוי סטטוס (Present / Out / Finished / Not Arrived) עובר דרכה.

**applyOptimistic(studentId, patch)**

מעדכנת את ה־UI מיד לפני שהשרת מחזיר תשובה.

* משנה סטטוס זמנית
* מעדכנת זמנים
* מעלה מונה שירותים
* שומרת pending patch

זו הסיבה שהמערכת מרגישה בזמן אמת.

**requestTransfer(seat, toRoom)**

יוצרת בקשת העברה.

* מוסיפה pending סגול מידית
* שולחת לשרת (createTransfer)
* מנקה optimistic אחרי הצלחה
* עושה rollback אם נכשל

**cancelPendingTransfer(seat)**

מבטלת בקשת העברה.

* מסירה pending מידית מה־UI
* קוראת ל־API לביטול
* מסנכרנת מחדש

**QrScanner.jsx**

זה קומפוננטה שמפעילה מצלמה אמיתית וסורקת QR קודים באמצעות הספרייה html5-qrcode.

המטרה שלה:

* לפתוח מצלמה
* לזהות QR
* להחזיר את הטקסט שנסרק דרך onResult
* לטפל בשגיאות דרך onError
* לעבוד בצורה יציבה גם ב-React StrictMode (DEV)

היא גם:

* מגנה מפני race conditions (start/stop)
* מונעת כפילויות סריקה
* בודקת HTTPS לפני הפעלה
* מנקה את המצלמה בצורה בטוחה כשסוגרים

**safeStart()**

הפונקציה שמפעילה את ה-QR scanner.

מה היא עושה:

* בודקת אם הדף מאובטח (HTTPS)
* בודקת שיש תמיכה ב-camera
* יוצרת מופע של Html5Qrcode
* מפעילה את המצלמה
* מאזינה ל-QR ומפעילה onResult
* מטפלת בשגיאות ומנקה אם משהו נכשל

**safeStop()**

הפונקציה שמכבה את הסורק בצורה בטוחה.

מה היא עושה:

* עוצרת את המצלמה אם היא רצה
* מנקה את ה-video element
* מאפסת refs
* מונעת memory leaks

**SeatActionsModal.jsx**

זה המודל שנפתח כשלוחצים על מושב בכיתה.

המטרה שלו:  
לאפשר ניהול מלא של סטודנט בזמן אמת במהלך מבחן.

הוא מאפשר:

* שינוי סטטוס (Present / Out / Finished / Absent / Not arrived)
* הוספת זמן נוסף
* בקשת העברה לכיתה אחרת
* ביטול בקשת העברה
* רישום הערת משמעת / צ'יט
* צפייה במידע (זמן בשירותים, כמות יציאות וכו')

המודל חכם מאוד מבחינת הרשאות:

* בודק אם המשתמש יכול לערוך נוכחות
* נועל פעולות בזמן Transfer
* מאפשר Cancel גם כשהכל נעול
* לא חוסם כפתורים בזמן המתנה לשרת (UX מהיר)

**setStatus(status, extra)**

הפונקציה הכללית לשינוי סטטוס.

מה היא עושה:

* בודקת הרשאות
* מפעילה onSetStatus
* מחכה להצלחה
* סוגרת את המודל רק אחרי הצלחה
* מציגה שגיאה אם נכשל

משמשת ל:

* finished
* absent
* not\_arrived
* temp\_out

**setPresentOrBackToRoom()**

מטפלת בשני מצבים:

* אם הסטודנט OUT → מחזיר אותו לחדר
* אם לא → מסמן Present

היא בעצם עוטפת setStatus("present") אבל מותאמת/compilerה לפי מצב.

זה הכפתור החכם של "Present / Back to room".

**setExtraMinutes(nextExtraMinutes)**

מעדכן זמן הארכה לסטודנט.

מה עושה:

* מגביל בין 0 ל־240 דקות
* שולח extraMinutes לשרת
* סוגר מודל אחרי הצלחה

אחראי על הארכות זמן.

**submitTransfer()**

שולח בקשת העברה לכיתה אחרת.

מה עושה:

* סוגר מודל מיד (UX מהיר)
* מפעיל onRequestTransfer
* אם נכשל → מדווח דרך onActionError

יוצר Transfer Pending (הסטודנט נהיה סגול).

**cancelTransfer()**

מבטל בקשת העברה קיימת.

מה עושה:

* סוגר מודל מיד
* מפעיל onCancelTransfer
* מדווח שגיאה אם נכשל

זה הכפתור שמחזיר שליטה כשהכל נעול.

**submitNote()**

שומר הערת משמעת / אירוע.

מה עושה:

* סוגר מודל מיד
* מפעיל onCheatNote
* שומר ל-report timeline ולתיק הסטודנט

ניהול אירועים בזמן אמת.

**SeatCard.jsx**

SeatCard הוא הכרטיס שמייצג מושב אחד במפת הכיתה.

כל ריבוע במפה = SeatCard אחד.

הוא מציג:

* שם הסטודנט
* ת"ז / מספר סטודנט
* מספר מושב
* סטטוס (Present / Out / Finished / Transfer וכו')
* טיימר בזמן שירותים
* כמות יציאות לשירותים
* התראת חריגה
* בועה של תוספת זמן (+15m וכו')
* צבע סגול אם יש Transfer Pending / Moving

וכשלוחצים עליו → נפתח SeatActionsModal.

**transferMeta(mode)**

אחראית על הסטייל והטקסט של מצב Transfer.

אם:

* moving → מציג "Moving"
* אחרת → "Pending transfer"

ומחזירה:

* צבע עיגול (seat)
* צבע רקע של הכרטיס (card)
* label להצגה

**seatLabel(seat)**

מנקה ומעצב טקסט של מושב.

מה עושה:

* אם ריק → מחזיר —
* אם AUTO → מחזיר AUTO
* אחרת → מחזיר הערך כרגיל

**utils.js**

זה קובץ שירות (utility file) לכל קומפוננטות ה־Classroom.

הוא לא מציג UI.  
הוא לא מדבר עם שרת.

הוא מספק פונקציות עזר ל:

* המרת זמן (מילישניות → mm:ss)
* פירוק מושב לשורה/עמודה
* מיפוי סטטוס לצבעים ועיצוב
* ניקוי וסטנדרטיזציה של סטטוסים
* שליפת מזהה סטודנט בצורה בטוחה
* שליפת שם בצורה בטוחה

**msToMMSS(ms)**

ממיר מילישניות לפורמט דקות:שניות.

**parseSeat(seat)**

מפרק טקסט מושב לשורה ועמודה.

משמש ליצירת המפה ב־ClassroomMap.

**normalizeStatus(status)**

מנקה ומאחד סטטוסים מהשרת.

לדוגמה:

* temp\_out → out
* moving → waiting\_transfer

מבטיח שלכל המערכת יש שפה אחידה.

**statusMeta(status)**

הכי חשוב לעיצוב.

מקבל סטטוס ומחזיר אובייקט עם:

* label
* text
* pill (צבע תג)
* card (צבע כרטיס)
* seat (צבע עיגול קטן)

**components/auth**

התיקייה הזו אחראית על כל מה שקשור ל־Authentication בצד ה־UI:

* התחברות (Login)
* הרשמה (Register)
* הצגת שגיאות
* מבנה ועיצוב של עמודי auth

היא לא מטפלת בלוגיקה של classroom / exams / attendance.  
היא רק השכבה של הטפסים והמסגרת שלהם.

**LoginForm.jsx**

זו קומפוננטת טופס התחברות פשוטה ונקייה.

מה היא עושה בגדול:

* מקבלת username ו-password מהקומפוננטה ההורה
* מאפשרת למשתמש להזין פרטים
* שולחת את הנתונים דרך onSubmit
* חוסמת אינפוטים וכפתור בזמן טעינה
* תומכת RTL / LTR לפי השפה

**RegisterForm.jsx**

זו קומפוננטת טופס הרשמה מלאה למערכת.

מה היא עושה בגדול:

* אוספת פרטי משתמש (שם מלא, אימייל, יוזר, סיסמה, תפקיד)
* מחייבת בחירת תפקיד (Supervisor / Lecturer)
* מחייבת Google reCAPTCHA לפני שליחה
* שולחת הכל ל־onSubmit מהקומפוננטה ההורה
* מנהלת מצב טעינה (loading) ומונעת שליחה כפולה

**setField(name, value)**

מעדכנת שדה בטופס.

**handleSubmit(e)**

זו הפונקציה הכי חשובה בקומפוננטה.

מה היא עושה:

1. מונעת רענון דף
2. בודקת אם יש reCAPTCHA key
3. בודקת שהמשתמש פתר captcha
4. מונעת שליחה כפולה
5. שולחת את הנתונים ל־onSubmit
6. מאפסת את ה־captcha אחרי שליחה

**LoginCard.jsx**

זה קומפוננטת UI עוטפת (wrapper component).

התפקיד שלה:

* לתת עיצוב קבוע ל־Login / Register
* להוסיף glass / blur / shadow מודרני
* לתמוך באנימציית shake במקרה של שגיאת התחברות
* להציג את מה שמועבר דרך children

**ErrorAlert.jsx**

קומפוננטת התראות כללית להצגת הודעות למשתמש.

המטרה שלה:  
להציג הודעת שגיאה / הצלחה / מידע / אזהרה בצורה אחידה, מעוצבת וברורה.

היא בנויה להיות גמישה ותומכת בשני סגנונות שימוש:

* <ErrorAlert message="..." />
* <ErrorAlert type="success" text="..." onClose={...} />

**LoginHeader.jsx**

זה קומפוננטת UI טהורה שמציגה את החלק העליון של עמוד ההתחברות.

התפקיד שלה:

* להציג לוגו של המערכת
* להציג כותרת התחברות
* להציג תת־כותרת
* לתמוך ב־i18n (רב־שפתיות)

אין כאן state.  
אין לוגיקה עסקית.  
אין אינטראקציה.  
רק תצוגה ממותגת ונקייה.

**RegisterHeader.jsx**

קומפוננטת UI להצגת החלק העליון של עמוד ההרשמה.

בדיוק כמו LoginHeader, אבל מותאם להרשמה.

התפקיד שלה:

* להציג לוגו של המערכת
* להציג כותרת ראשית
* להציג תת־כותרת ייעודית להרשמה
* להשתמש במערכת תרגומים (i18n)

**AuthFooter.jsx**

התפקיד שלה:

* להציג טקסט footer בתחתית עמודי login / register
* להשתמש ב־i18n כדי שהטקסט יתאים לשפה (עברית / אנגלית)
* להיות חלק מה־layout של auth בלבד

אין כאן לוגיקה עסקית.  
אין state.  
אין side effects.  
רק תרגום והצגה.

**components/layout**

תיקיית layout אחראית על מבנה המעטפת (Application Shell) של האזור המוגן במערכת – כלומר כל מה שנמצא תחת ‎/app‎.

הרכיבים בתיקייה זו מגדירים:  
• מבנה עמוד ראשי לאחר התחברות  
• Sidebar ניווט קבוע  
• Topbar עליון  
• חלוקת מסך אחידה לכל הדפים המוגנים  
• העברת context פנימי בין Layout ל־Nested Routes

התיקייה מממשת עקרון חשוב בארכיטקטורה:

Nested Layout Architecture

כל הדפים תחת /app נטענים בתוך Layout משותף.

**AppLayout.jsx**

AppLayout הוא רכיב המעטפת הראשי של האזור המוגן (/app).

כל הדפים כמו:

* Dashboard
* Classroom
* Reports
* Student
* Admin

נטענים בתוכו דרך <Outlet />.

useEffect – טעינת משתמש

אחראי על:  
• שליחת GET /api/auth/me  
• שמירת המשתמש ב־state  
• טיפול במצב loading

handleLogout()

אחראית על:  
• קריאה ל־POST /api/auth/logout  
• ניקוי state  
• Redirect ל־/login

**Sidebar.jsx**

Sidebar הוא רכיב ניווט צדדי המופיע בכל עמודי /app.

הוא מציג תפריט שונה לפי Role.

**תפקיד הקומפוננטה**

• הצגת קישורים בהתאם ל־Role  
• סימון נתיב פעיל  
• ניווט בין אזורים  
• תמיכה ב־RTL  
• מצב Collapse במסכים קטנים

**buildMenuByRole(role)**

פונקציה שמחזירה מערך קישורים בהתאם ל־role.

**isActive(path)**

בודקת אם הנתיב הנוכחי תואם לקישור כדי לסמן אותו כפעיל.

**Topbar.jsx**

Topbar הוא האזור העליון של /app.

מציג מידע על המשתמש, שעון, התראות וכפתור יציאה.

**תפקיד הקומפוננטה**

• הצגת שם משתמש מחובר  
• הצגת Role  
• כפתור Logout  
• אפשרות להחלפת שפה  
• תצוגת Theme Toggle  
• הצגת זמן בחינה (אם רלוונטי)

**handleLogout()**

• קריאה ל־API  
• מחיקת session  
• ניווט ל־Login

**formatTime()**

• עיצוב זמן להצגה  
• שימוש ב־Date או טיימר חיצוני

**components/dashboard**

תיקיית **dashboard** מכילה את כל רכיבי לוח הבקרה (Dashboard) של המערכת לאחר התחברות.

האזור משמש כמרכז שליטה בזמן אמת עבור:

* Admin
* Lecturer
* Supervisor

הרכיבים בתיקייה אחראים על:  
• הצגת בחינות פעילות  
• מעבר בין בחינות וכיתות  
• ניטור אירועים בזמן אמת  
• ניהול העברות תלמידים  
• ניהול תלמידים במהלך בחינה  
• הצגת סטטיסטיקות  
• טיימר חכם לבחינה  
• מערכת Toast להתראות

ה־Dashboard הוא אזור דינמי מאוד ומתעדכן בזמן אמת.

**ActiveClassroomsTabs.jsx**

**תיאור כללי**

רכיב המאפשר ניווט מהיר בין כל הכיתות הפעילות בכל הבחינות הרצות.

מיועד בעיקר ל־Admin לצורך מעבר מהיר בין חדרים.

**תפקיד הקומפוננטה**

• איסוף כל הכיתות מכל הבחינות הפעילות  
• בניית רשימת טאבים ייחודית  
• מניעת כפילויות  
• מיון יציב לפי קורס ואז חדר  
• סימון כיתה פעילה  
• הפעלת onPick בעת בחירה

**פונקציות מרכזיות**

**norm(v)**

מנקה ערך ומחזיר מחרוזת תקינה.

**makeKey(examId, roomId)**

יוצר מפתח ייחודי לכל כיתה בפורמט:  
examId::roomId

**useMemo – בניית items**

• מעבר על כל הבחינות  
• שליפת classrooms  
• יצירת רשימה אחידה  
• סינון כפילויות  
• מיון אלפביתי

**DashboardAddDeleteStudentsCard.jsx**

**תיאור כללי**

רכיב לניהול תלמידים בזמן אמת בתוך בחינה.

מאפשר:  
• הוספת תלמיד לכיתה  
• מחיקת תלמיד מהכיתה

**תפקיד הקומפוננטה**

• שליחת בקשות ל־API  
• הצגת הודעות הצלחה / שגיאה  
• חסימת כפתורים בזמן פעולה  
• רענון נתונים דרך onChanged

**פונקציות מרכזיות**

**handleAdd(e)**

• מונע רענון דף  
• קורא ל־addStudentToExam  
• מנקה שדות לאחר הצלחה  
• מציג הודעת הצלחה

**handleDelete(e)**

• קורא ל־deleteStudentFromExam  
• מנקה שדה מזהה  
• מציג הודעת הצלחה

**EventsFeed.jsx**

**תיאור כללי**

רכיב הצגת אירועים בזמן אמת.

משלב:  
• alerts  
• events  
• חיפוש  
• סינון לפי חדר  
• Toast חכם  
• סימון Seen עבור Lecturer

**תפקיד הקומפוננטה**

• מיזוג רשימות  
• מיון לפי זמן  
• זיהוי אירוע חדש  
• סינון חכם  
• ניהול תצוגת Severity  
• מניעת הצפת Toast

**פונקציות מרכזיות**

**normalize(item, source)**

מאחד פורמט אירוע ממקורות שונים.

**shouldNotifyToast(it)**

מחליט האם להציג Toast לפי:  
• סוג  
• חומרה  
• מקור

**handleSeen(it)**

• זמין ל־Lecturer בלבד  
• קורא ל־markCallLecturerSeen  
• מבצע optimistic update

**ExamOverviewCard.jsx**

**תיאור כללי**

רכיב תצוגת פרטי הבחינה הנוכחית.

מציג:  
• שם קורס  
• תאריך התחלה וסיום  
• מרצה  
• משגיחים  
• סטטיסטיקות  
• SmartExamClock

**תפקיד הקומפוננטה**

• תצוגת מידע מרכזי  
• שילוב KPI  
• הצגת סטטוס live  
• תמיכה ב־Simulated Time

**ExamTabs.jsx**

**תיאור כללי**

טאבים למעבר בין בחינות פעילות.

**תפקיד הקומפוננטה**

• הצגת כל הבחינות הרצות  
• סימון בחינה פעילה  
• הצגת מספר חדרים  
• הצגת שעות התחלה/סיום

**RoomTabs.jsx**

**תיאור כללי**

טאבים למעבר בין חדרים בתוך בחינה.

**תפקיד הקומפוננטה**

• מעבר בין כיתות  
• הצגת מספר תלמידים בכל חדר  
• סימון חדר פעיל

**פונקציות מרכזיות**

**countInRoom(rid)**

סופר תלמידים לפי attendance.

**SmartExamClock.jsx**

**תיאור כללי**

טיימר חכם לבחינה עם חישוב Progress.

מציג:  
• זמן שנותר  
• אחוז התקדמות  
• הארכת זמן  
• מצב Extra Time

**תפקיד הקומפוננטה**

• חישוב זמן נותר  
• חישוב אחוז התקדמות  
• תמיכה ב־extraTime  
• הצגת SVG Ring

**פונקציות מרכזיות**

**toMs(v)**

ממיר Date / מספר / ISO ל־milliseconds.

**fmtHMS(ms)**

פורמט זמן HH:MM:SS.

**חישוב progress**

progress = (now - start) / (end - start)

**Toast.jsx**

**תיאור כללי**

רכיב התראות צפות.

משתמש ב־createPortal להצגה מעל כל המערכת.

**תפקיד הקומפוננטה**

• הצגת הודעה זמנית  
• סגירה אוטומטית  
• תמיכה ב־types שונים  
• תמיכה ב־i18n

**TransfersPanel.jsx**

**תיאור כללי**

רכיב לניהול בקשות העברה בין חדרים.

מאפשר:  
• אישור  
• דחייה  
• ביטול  
• צפייה בלבד לפי הרשאות

**תפקיד הקומפוננטה**

• מימוש RBAC בצד לקוח  
• סינון לפי Role  
• הצגת סטטיסטיקות incoming/outgoing  
• טיפול במצב Room Full  
• עדכון דרך onChanged

**פונקציות מרכזיות**

**canHandle(me, exam, item)**

בודקת אם המשתמש יכול לאשר/לדחות.

**canCancel(me, exam, item)**

בודקת אם המשתמש יכול לבטל.

**onApprove / onReject / onCancel**

• קריאה ל־API  
• טיפול בשגיאות  
• הצגת הודעות Flash  
• רענון נתונים

**components/report**

**תיאור כללי**

תיקיית report אחראית על מערכת הדוחות של המערכת.

האזור הזה מאפשר:

* צפייה בסטטיסטיקות בחינה
* ניתוח נתוני נוכחות
* הצגת חריגות ואירועים
* פירוט לפי חדרים
* ייצוא דוחות ל-PDF
* ייצוא דוחות ל-Excel (XLSX)

המערכת מחולקת לקומפוננטות קטנות לפי עקרון:

Separation of Concerns  
כל קומפוננטה אחראית רק על חלק אחד במסך הדוחות.

**ExamReportEXCEL.js**

**תפקיד כללי**

מודול ליצירת קובץ Excel אמיתי (XLSX) באמצעות הספרייה xlsx.

הוא יוצר קובץ עם שלושה גיליונות:

1. Summary – נתוני סיכום
2. Rooms – פירוט לפי חדרים
3. Incidents – רשימת אירועים כרונולוגית

אין כאן UI.  
זהו Utility לייצוא בלבד.

**downloadExamReportXlsx({ report, examMeta, filename })**

הפונקציה המרכזית בקובץ.

מה היא עושה:

1. יוצרת Workbook חדש
2. בונה Sheet של Summary
3. מחשבת נתונים מצטברים:
   * סך תלמידים
   * נוכחים
   * אחוז נוכחות
   * חריגות
4. בונה Sheet של Rooms
5. בונה Sheet של Incidents
6. מייצאת קובץ באמצעות XLSX.writeFile()

**safeSheetName(name)**

מנקה שם גיליון:

* מסיר תווים לא חוקיים
* מגביל ל-31 תווים (מגבלת Excel)

**autoWidthFromRows(rows)**

מחשב אוטומטית רוחב עמודות לפי תוכן.

**ExamReportPDF.jsx**

**תפקיד כללי**

קומפוננטה ליצירת דוח PDF באמצעות @react-pdf/renderer.

מייצרת מסמך A4 עם:

* כותרת בחינה
* סיכום KPI
* טבלת חדרים
* רשימת אירועים
* הערות

לא מציג UI רגיל – אלא Document ל-PDF בלבד.

**ExamReportPDF({ report, examMeta })**

הקומפוננטה הראשית.

מה היא עושה:

1. מחשבת:
   * total students
   * present
   * attendance rate
   * violations
   * incident count
2. בונה מבנה מסמך:
   * Header
   * Summary section
   * Room breakdown table
   * Incidents list (עד 20)
   * Notes

**rowSum(rooms, key)**

פונקציית עזר לסיכום ערכים מכל החדרים.

**ReportsHeaderSection.jsx**

**תפקיד כללי**

כותרת עליונה של מסך הדוחות.

מציג:

* כותרת ראשית
* תת־כותרת (אופציונלי)

קומפוננטת UI טהורה.

**ReportsKpisSection.jsx**

**תפקיד כללי**

מציג KPI עליונים של מערכת הדוחות.

כולל 5 מדדים:

* Ended Exams
* Avg Attendance
* Cheating
* Toilet
* Teacher Calls

**Kpi({ title, value, hint, tone })**

קומפוננטת משנה להצגת כרטיס KPI בודד.

tone קובע צבע (emerald / rose / sky וכו').

**ReportsControlsSection.jsx**

**תפקיד כללי**

אזור שליטה במסך הדוחות.

מאפשר:

* חיפוש בחינות
* בחירת בחינה
* הורדת PDF
* הורדת Excel
* הצגת בחינה נבחרת

**onDownloadPdf()**

מוריד דוח PDF של הבחינה הנבחרת.

**onDownloadExcel()**

מוריד דוח XLSX.

**ReportsChartsSection.jsx**

**תפקיד כללי**

אזור גרפים במסך הדוחות.

מציג 4 גרפים:

1. Attendance Line chart
2. Cheating Bar chart
3. Toilet usage chart
4. Teacher calls chart

מקבל Charts כ-Dependency Injection.

**ReportsDetailsSection.jsx**

**תפקיד כללי**

פירוט מלא של בחינה נבחרת.

כולל:

* טבלת Room breakdown
* רשימת incidents
* הערות

אם אין בחינה נבחרת → מציג הודעת הדרכה.

**ReportsErrorBanner.jsx**

קומפוננטת שגיאה פשוטה להצגת הודעת כשל בטעינת דוחות.

**ארכיטקטורה של אזור Report**

האזור בנוי בשכבות:

Header  
KPI Section  
Controls  
Charts  
Details  
Export (PDF / Excel)

ה-Container הראשי (ReportsPage) מרכיב את כולם.

הוא:

* שולף נתונים מהשרת
* מחשב סטטיסטיקות
* מעביר props לכל קומפוננטה
* מנהל state מרכזי

**Context API**

תיקיית: client/src/context

Context משמש לניהול State גלובלי באפליקציה ומונע prop drilling.

**AuthContext**

**מטרה**

ניהול מצב משתמש מחובר (Authentication State).

**אחריות מרכזית**

* שמירת המשתמש המחובר
* בדיקת session מול השרת
* ניהול logout
* ניהול role (admin / supervisor / lecturer / student)

**פונקציות מרכזיות**

**login(credentials)**

* שולחת בקשת POST ל:
* /api/auth/login
* שומרת את המשתמש ב-state
* מפעילה redirect לפי role

**logout()**

* שולחת בקשת POST ל:
* /api/auth/logout
* מאפסת state

**checkAuth()**

* שולחת GET ל:
* /api/auth/me
* בודקת אם session עדיין פעיל

**חשיבות אדריכלית**

מאפשרת הפרדת לוגיקת התחברות מרכיבי UI.  
משפרת קריאות ותחזוקת קוד.

**LocalizationContext**

**מטרה**

ניהול שפה דינמית במערכת (עברית / אנגלית).

**אחריות**

* שמירת שפה נוכחית
* החלפת שפה בזמן ריצה
* טעינת מילון תרגומים
* התאמת כיוון RTL/LTR

**פונקציות**

**changeLanguage(lang)**

* משנה שפה
* שומר ב-localStorage
* מעדכן direction במסמך

**t(key)**

* מחזירה תרגום לפי מפתח
* שולפת מתוך קובץ JSON

**קבצים**

/locales/he.json

/locales/en.json

**יתרון תכנוני**

מאפשר הוספת שפות בעתיד ללא שינוי לוגיקה עסקית.

**Custom Hooks**

תיקיית:  
client/src/hooks

Custom Hooks מבודדים לוגיקה חוזרת.

**2.1 useAuth**

עוטף את AuthContext.

מאפשר שימוש פשוט:

const { user, login, logout } = useAuth();

**2.2 useDashboard**

* מבצע קריאה ל:
* /api/dashboard
* מנהל loading + error state
* מחזיר נתונים סטטיסטיים

**2.3 useExams**

* CRUD לבחינות
* משתמש ב-fetch או axios
* מבצע:
  + GET /api/exams
  + POST /api/exams
  + PUT
  + DELETE

**2.4 useTransfers**

* ניהול בקשות מעבר
* קריאות API:
* /api/transfers

**2.5 useIncidents**

* רישום אירועים
* שליחה לשרת

**2.6 useMessages**

* מערכת הודעות
* ניהול state מקומי של הודעות

**2.7 useStudent**

* ניהול נתוני סטודנט
* התחברות OTP
* בקשת קוד

**יתרונות השימוש ב-Hooks**

✔ הפרדת לוגיקה מה-UI  
✔ קריאות גבוהה  
✔ קוד נקי  
✔ אפשרות שימוש חוזר

**Localization System**

מערכת תרגום עצמאית.

**מבנה**

locales/

he.json

en.json

דוגמה:

{

"login": "התחברות",

"logout": "התנתקות"

}

**עקרונות יישום**

* שימוש ב-Context
* שמירת שפה ב-localStorage
* התאמת כיוון:

document.dir = "rtl";

**client/src/services:**

התיקייה services היא שכבת ה-API בצד הלקוח: היא מרכזת את כל קריאות ה-HTTP לשרת, שומרת על אחידות בטיפול בשגיאות/Timeout, ומפרידה בין UI ללוגיקה של תקשורת.

**1) \_cache.js**

**מטרה:** Cache בזיכרון (in-memory) עם TTL כדי להפחית קריאות חוזרות לשרת ולהאיץ ניווט.

**פונקציות מרכזיות:**

* cacheGet(key) – מחזיר ערך מה-cache אם לא פג תוקף.
* cacheSet(key, value, ttlMs=30000) – שומר ערך עם תוקף זמן.
* cacheDel(prefixOrKey) – מוחק key יחיד או כל keys שמתחילים ב-prefix.
* fetchWithCache(key, fetcher, { ttlMs, force }) – מביא נתונים דרך fetcher ומשתמש ב-cache לפי צורך (אפשר לאלץ ריענון עם force).

**2) http.js**

**מטרה:** פונקציית HTTP אחידה לכל הפרויקט שמונעת “טעינה אינסופית” באמצעות Timeout + Parse עקבי + שגיאות עקביות.

**פונקציות מרכזיות:**

* http(url, options) – עטיפה סביב fetch עם:
  + credentials: "include" כדי לשמור session cookies.
  + JSON serialization אוטומטי ל-body.
  + Timeout (ברירת מחדל 15 שניות).
* parseResponse(res) (פנימי) – מפרש JSON/טקסט ומרים Error עם status ו-data אם res.ok נכשל.

**3) auth.service.js**

**מטרה:** כל פעולות ההתחברות/אימות (Staff + Student OTP) מול השרת, באמצעות http().

**פונקציות ו-API:**

* loginUser({username,password}) → POST /api/auth/login
* logout() → POST /api/auth/logout
* getMe() → GET /api/auth/me
* registerUser(payload) → POST /api/auth/register
* staffForgotPassword(email) → POST /api/auth/staff/forgot-password
* requestStudentOtp({email,studentId}) → POST /api/auth/student/request-otp
* verifyStudentOtp({email,studentId,otp}) → POST /api/auth/student/verify-otp
* checkUsername(username) → GET /api/auth/check-username?username=... (מחזיר {taken, exists})
* isUsernameTaken(username) – התאמה ל-UI: מחזיר boolean פשוט.

**4) admin.service.js**

**מטרה:** פעולות ניהול למשתמש Admin מול endpoints ייעודיים.

**פונקציות ו-API:**

* listUsers(role) → GET /api/admin/users?role=... (אם role נשלח).
* updateExamAdmin(examId,payload) → PUT /api/admin/exams/:examId
* deleteExamAdmin(examId,{force}) → DELETE /api/admin/exams/:examId?force=1 (אופציונלי)
* autoAssignExam(examId) → POST /api/admin/exams/:examId/auto-assign
* autoAssignDraft(payload) → POST /api/admin/exams/auto-assign-draft (הכנת טיוטת שיבוץ)

**5) chat.service.js**

**מטרה:** שליחת הודעה לצ’אט AI בצד שרת (FAQ מקומי / מודל חיצוני לפי השרת).

**פונקציה ו-API:**

* chatWithAI({message,lang}) → POST /api/chat

**6) dashboard.ADD.DELETE.Students.service.js**

**מטרה:** פעולות ניהול סטודנטים (הוספה/מחיקה) מתוך הדשבורד, כולל נרמול קלט והמרת הודעות שרת לשגיאות ידידותיות.

**פונקציות מרכזיות:**

* addStudentToExam({examId, firstName, lastName, studentId, email, roomId})  
  → POST /api/dashboard/students/add  
  מבצע בדיקות חובה (שם פרטי/משפחה/ת״ז) ונרמול של whitespace.
* deleteStudentFromExam({examId, studentId, useDelete})  
  → ברירת מחדל POST /api/dashboard/students/delete  
  או DELETE אם useDelete=true.

**טיפול שגיאות ייחודי:**

* ממפה הודעות כמו ROOM\_FULL, INVALID\_ROOM, STUDENT\_ALREADY\_IN\_CLASS לטקסט ברור למשתמש.

**7) dashboard.service.js**

**מטרה:** שליפת “תמונת מצב” לדשבורד (כבד/קל) + שעון שרת.

**פונקציות ו-API:**

* getDashboardSnapshot({examId}) → GET /api/dashboard/snapshot?examId=... (כבד יותר)
* getDashboardSnapshotLite({examId}) → GET /api/dashboard/snapshot-lite?examId=... (מומלץ ל-UI מהיר)
* getClock() → GET /api/dashboard/clock

**8) exams.service.js**

**מטרה:** פעולות בחינות ופעולות נוכחות, כולל endpoints ל-Admin.

**פונקציות ו-API:**

* getExams() → GET /api/exams
* createExam(payload) → POST /api/exams
* startExam(examId,{force}) → POST /api/exams/:id/start?force=1 (+ body {force:true} לתאימות)
* endExam(examId,{force}) → POST /api/exams/:id/end?force=1 (+ body)
* updateExamAdmin(examId,payload) → PUT /api/admin/exams/:id
* getMyReport(examId) → GET /api/exams/:id/my-report
* updateAttendance({examId, studentId, patch}) → PATCH /api/exams/:examId/attendance/:studentId
* getAdminExams(params) → GET /api/admin/exams?... (סינון לפי q/status/mode/from/to)

**9) incidents.service.js**

**מטרה:** רישום אירועי משמעת במהלך בחינה + סימון אירוע “קריאה למרצה נצפתה”.

**פונקציות ו-API:**

* logIncident(examId, studentId, payload) → POST /api/incidents/:examId
* markCallLecturerSeen(examId, targetEventId) → POST /api/incidents/:examId עם kind: "CALL\_LECTURER\_SEEN"

**10) reports.service.js**

**מטרה:** דוחות מנהליים + אנליטיקה + הורדת קבצים (PDF/CSV) דרך השרת.

**פונקציות ו-API:**

* getReportsList() → GET /api/reports
* getReportsAnalytics() → GET /api/reports/analytics
* getReportDetails(examId) → GET /api/reports/:examId
* downloadReportPdf(examId, filename) → GET /api/reports/:examId/pdf (מוריד Blob ויוצר download)
* downloadReportCsv(examId, filename) → GET /api/reports/:examId/csv (אותו עיקרון)

**11) student.service.js**

**מטרה:** מסכים לסטודנט: רשימת בחינות שהסתיימו + צפייה בדוח אישי.

**פונקציות ו-API:**

* listMyEndedExams() → GET /api/student/exams
* getMyExamReport(examId) → GET /api/student/exams/:examId/me

**12) transfers.service.js**

**מטרה:** ניהול בקשות מעבר (transfer) עבור בחינה, כולל טיפול ייחודי במקרה חדר מלא (409).

**פונקציות ו-API:**

* listTransfers(examId) → GET /api/transfers?examId=... (מחזיר items)
* createTransfer({examId, studentId, toClassroom, toSeat="AUTO", note}) → POST /api/transfers
* approveTransfer(id) → POST /api/transfers/:id/approve  
  אם מתקבל 409 → לא זורק שגיאה; מחזיר {roomFull:true, ...} כדי שה-UI יוכל להציג הודעה נכונה.
* rejectTransfer(id) → POST /api/transfers/:id/reject
* cancelTransfer(id) → POST /api/transfers/:id/cancel

**client/src/pages :**

בתיקייה pages יש 8 דפי־על (Route Pages). כל Page הוא “מסך” שמחובר ל־React Router, ובדרך כלל:

* מנהל **state** מקומי (useState)
* מבצע **טעינת נתונים** מהשרת (useEffect)
* קורא לפונקציות API דרך שכבת services/\*
* מרנדר קומפוננטות UI מתוך components/\*

**pages/auth/LoginPage.jsx**

**מטרה:** מסך התחברות שמכיל **שני זרמים**:

1. **Staff Login** (Admin/Supervisor/Lecturer) עם username + password
2. **Student Login** עם studentId + email + OTP

**רכיבים עיקריים**

* useNavigate() — ניווט למסכים אחרי התחברות.
* useTranslation() + שינוי dir של המסמך ל־RTL/LTR לפי שפה.
* useShake() — “שייק” ויזואלי בשגיאה/שדות חסרים.

**State מרכזי**

* tab — איזה טאב מוצג (“staff” / “student”)
* Staff: staffUsername, staffPassword, showForgot, forgotEmail
* Student OTP: studentId, studentEmail, otp, studentStep (שלב 1=בקשת OTP, שלב 2=אימות OTP)
* notice — הודעה אחידה (success/error/warning/info)
* isLoading — חיווי טעינה על כפתורים/טפסים

**פונקציות מרכזיות (לפי זרימה)**

* onSubmitStaff(e)  
  ולידציה בסיסית → loginUser({username,password}) → ניווט:
  + אם role === "student" → /app/student
  + אחרת → /app/dashboard
* onRequestOtp(e)  
  שולח requestStudentOtp({ email, studentId }) → מעביר לשלב אימות OTP
* onVerifyOtp(e)  
  שולח verifyStudentOtp({ email, studentId, otp }) → ניווט /app/student
* onStaffForgot()  
  שולח staffForgotPassword(email) (שליחת פרטי התחברות במייל / הודעה כללית)
* showNotice(type,text,{autoHideMs})  
  מנגנון אחיד להצגת הודעות (כולל טיימר ניקוי).

**pages/auth/RegisterPage.jsx**

**מטרה:** הרשמת משתמשי **Staff בלבד** (Supervisor/Lecturer). סטודנטים לא נרשמים כאן (בגרסת הדמו).

**State מרכזי**

* message {show,type,text} הצלחה/שגיאה
* isLoading חסימת פעולות בזמן שליחה

**פונקציות מרכזיות**

* handleSubmit(formData)  
  עושה:
  1. בדיקות צד לקוח: role קיים, התאמת סיסמאות
  2. בדיקת “שם משתמש תפוס” עם isUsernameTaken(username) (pre-check)
  3. קריאה ל־registerUser({... , captchaToken})

**דגשים**

* הקוד שולח captchaToken → בשרת יש אימות reCAPTCHA מול Google (API חיצוני).
* ניהול שפה + RTL בדומה ל־Login.

**pages/homepage/HomePage.jsx**

**מטרה:** דף הבית/לנדינג של המערכת. בדרך כלל כולל:

* הצגת המוצר והערך שלו
* כניסה ל־Login/Register
* UI בסיסי + כפתורי ניווט

**מה לבדוק בקוד שלך כשאת כותבת תיק:**  
האם הדף משתמש ב־i18n? האם הוא מפנה ל־routes נכונים? האם הוא “סטטי” או מושך מידע?

**pages/dashboard/DashboardPage.jsx**

**מטרה:** דשבורד תפעולי בזמן אמת (לרוב Supervisor/Lecturer/Admin).  
זה המסך שמציג “Snapshot” של מצב הבחינה/חדרים/נוכחות/אירועים.

**מה קורה ברמה לוגית**

* טעינת Snapshot (או “lite snapshot”) מהשרת
* הצגת כיתות/מפת מושבים/סטטוסים
* הפעלה של פעולות כמו נוכחות, שירותים, אירועים, הודעות, העברות (תלוי בהרשאות)
* Polling / Refresh: בדרך כלל Page כזה מפעיל רענון מחזורי או כפתור refresh.
* Role-based UI: חלק מהפעולות זמינות רק לתפקידים מסוימים (מגובה גם בשרת).
* State גדול → נהוג לפרק לקומפוננטות בתוך components/dashboard/\*.

**pages/exams/ExamsPage.jsx**

**מטרה:** רשימת מבחנים + כניסה לניהול/דשבורד של מבחן.

**מה Page עושה בדרך כלל**

* מביא רשימת מבחנים מהשרת (GET /api/exams)
* מציג כרטיסים/טבלה
* מאפשר פתיחה של מבחן (view) / מעבר לדשבורד
* במידה וזה Admin — ייתכן גם יצירה/עריכה/Start/End (אבל אצלך ניהול מלא נמצא בעיקר בעמוד Admin נפרד)

**pages/admin/ManageExamsPage.jsx**

**מטרה:** מסך אדמין מלא לניהול מבחנים:

* יצירת מבחן
* עריכת מבחן (כיתות, מרצה, משגיחים)
* מחיקת מבחן
* Start/End (עם בדיקות חלון זמן)
* **Auto-Assign Draft**: יצירת טיוטת חלוקה אוטומטית (בלי לשמור DB)

**עקרונות מרכזיים בקוד**

1. **טעינת נתונים דו־שלבית לביצועים**
   * קודם טוען exams (getAdminExams) כדי לפתוח מסך מהר
   * אחר כך טוען lecturers/supervisors ברקע (listUsers)
2. **ולידציות לפני שליחה**
   * תאריך התחלה/סיום תקינים
   * שם קורס
   * חייב lecturer
   * חייב לפחות כיתה אחת
   * לכל כיתה חייב משגיח
3. **ניהול כיתות (rooms) עם מזהה פנימי \_uid**
   * מאפשר עריכה/מחיקה/הוספה בלי בעיות key ב־React
4. **Optimistic UI**
   * upsertExamLocal() / removeExamLocal() לעדכן מסך מהר עוד לפני refresh מלא
5. **Start/End עם windowState**
   * windowState(exam) בודק האם עכשיו בתוך חלון המבחן (startAt-endAt)
   * Admin יכול גם “force end” אם עבר זמן (השרת מאכף הרשאה)

**פונקציות מרכזיות**

* refresh() — טוען נתונים (exams ואז users)
* submitCreateOrEdit({isEdit}) — שולח payload ליצור/לעדכן
* onDeleteExam(exam) — מחיקה רגילה/force למחיקה בזמן running
* onStartExam(exam) / onEndExam(exam) — התחלה/סיום מבחן
* onAutoAssignDraft() — קריאה ל־draft auto assign (בלי לשמור DB)

זה אחד הקבצים הכי “מדורגי-הגשה”: יש בו הרבה לוגיקה, ולידציה, טיפול בשגיאות, ופיצול UI למודלים/מודאלים.

**pages/reports/ReportsPage.jsx**

**מטרה:** מסך דוחות/היסטוריה אחרי מבחנים:

* רשימת מבחנים שהסתיימו
* KPI-ים מסכמים
* גרפים (Chart.js)
* הורדות: PDF / CSV / Excel

**מה מיוחד בקובץ**

* שימוש כבד ב־useMemo כדי לחשב:
  + סדרות לגרפים
  + KPI aggregates
  + פילטרים/חיפוש
* הורדת PDF נעשית **בצד לקוח** באמצעות @react-pdf/renderer (lazy import כדי לא להעמיס ניווט).
* הורדת Excel נעשית דרך xlsx בצד לקוח.
* יש fallback ל־Server PDF אם קיים (ניסיון שני).

**פונקציות מרכזיות**

* onDownloadPdf()יוצר blob PDF בצד לקוח, ואם נכשל מנסה להוריד מהשרת.
* onDownloadExcel()יוצר XLSX בצד לקוח.
* onDownloadCsv()קורא ל־endpoint CSV בשרת.

**pages/student/StudentMyExamReportPage.jsx**

**מטרה:** מסך סטודנט לקריאת דוח אישי **אחרי שהמבחן הסתיים** (read-only).

**זרימה**

1. listMyEndedExams() → ממלא dropdown של מבחנים סיימו
2. כשנבחר מבחן → getMyExamReport(examId) → מציג:
   * נתונים אישיים (Room/Seat/Arrived/Finished)
   * ניקוד (אם קיים)
   * timeline אישי
   * notes/transfers
   * incidents/messages

**דגשים**

* שני loaders: אחד לרשימה (loadingList) ואחד לדוח (loadingReport)
* useMemo לטונים (Badge colors) לפי score/status/severity

**כל ה-API שהמערכת משתמשת בהם (Client ⇄ Server)**

השרת מחובר ב־server/index.js לנתיבים הבאים:

**Authentication — /api/auth**

* POST /login — התחברות staff
* GET /me — מי המשתמש המחובר (דורש session)
* POST /logout
* POST /register — הרשמת staff (כולל captchaToken)
* GET /check-username
* POST /student/request-otp
* POST /student/verify-otp
* POST /staff/forgot-password

**Dashboard — /api/dashboard**

* GET /clock
* GET /snapshot
* GET /snapshot-lite
* POST /students/add
* POST /students/delete (וגם DELETE /students/delete)

**Exams — /api/exams**

* GET / — רשימת מבחנים (מסונן לפי role בשרת)
* GET /:examId
* POST / — יצירת מבחן (Admin)
* PATCH /:examId/attendance/:studentId
* POST /:examId/students
* DELETE /:examId/students/:studentId
* POST /:examId/start
* POST /:examId/end

**Transfers — /api/transfers**

* GET /
* POST /
* POST /:id/approve
* POST /:id/reject
* POST /:id/cancel

**Messages — /api/messages**

* GET /:examId
* POST /:examId
* POST /:examId/read-all

**Incidents — /api/incidents**

* POST /:examId

**Admin — /api/admin**

* GET /users?role=...
* GET /exams
* PUT /exams/:examId
* DELETE /exams/:examId
* POST /exams/auto-assign-draft
* POST /exams/:examId/auto-assign

**Reports — /api/reports**

* GET / — רשימת מבחנים שהסתיימו
* GET /analytics — נתוני גרפים
* GET /:examId — פרטי דוח מלא
* GET /:examId/pdf
* GET /:examId/csv

**Student — /api/student**

* GET /exams
* GET /exams/:examId/me

**AI Chat — /api/chat**

* POST / — צ׳אט מסייע (מוגבל/מקוצר/מגובה cache)

תיק מתכנת-SERVER

**תיאור כללי:**

**השרת בפרויקט נבנה באמצעות הטכנולוגיות הבאות:**

Node.js – סביבת הרצה ל־JavaScript בצד השרת

Express.js – Framework לבניית שרת

MongoDB – מסד נתונים מסוג NoSQL

Mongoose – ספרייה לניהול חיבור ונתונים מול MongoDB

- Express-Session מנגנון לניהול התחברות משתמשים באמצעות Session

- Connect-Mongo שמירת Sessions בתוך MongoDB

**תפקיד השרת במערכת:**

ניהול משתמשים (Admin, Lecturer, Supervisor, Student)

ביצוע תהליכי התחברות והרשמה

ניהול בחינות ומבחנים

ניהול כיתות וחדרים

טיפול בבקשות העברה בין חדרים

שמירת אירועים ודוחות בזמן אמת

שליחת מידע ל־Frontend באמצעות REST API

ניהול הרשאות ואבטחת נתיבים

**טכנולוגיות וספריות עיקריות**

-express 🡨יצירת שרת HTTP

-mongoose 🡨חיבור ל־MongoDB

-express-session 🡨 ניהול התחברות משתמשים באמצעות Session

-cors 🡨תקשורת בין שרת ללקוח

-dotenv 🡨משתני סביבה

-nodemon 🡨רענון אוטומטי

**API.1 בהם נעשה שימוש:**

APIפנימי (Express Routes)

| Method | Endpoint | תיאור |
| --- | --- | --- |
| POST | /api/auth/login | התחברות Staff |
| POST | /api/auth/logout | יציאה |
| GET | /api/auth/me | בדיקת session |
| POST | /api/auth/student/request-otp | בקשת OTP |
| POST | /api/auth/student/verify-otp | אימות OTP |
| CRUD | /api/exams | ניהול בחינות |
| CRUD | /api/transfers | ניהול העברות |
| CRUD | /api/incidents | ניהול אירועים |
| GET | /api/dashboard | נתוני דשבורד |

API חיצוני

SendGrid Mail API

<https://docs.sendgrid.com/api-reference/mail-send/mail-send>

**2.קישור למסד הנתונים (Database)**

**await connectDB();**

תפקיד:

יצירת חיבור ל־MongoDB לפני הפעלת השרת.

אופן החיבור:

החיבור מתבצע באמצעות משתנה סביבה: MONGO\_URI

**.3 סביבות עבודה מיוחדות**

Backend:

* Node.js
* Express
* Mongoose
* express-session
* connect-mongo
* dotenv
* CORS

Database:

* MongoDB Atlas

Email:

* SendGrid

**.4 קטעי קוד ממקורות חיצוניים (רפרנסים)**

בחלקים מסוימים נעשה שימוש בדוגמאות קוד נפוצות:

**ניהול Sessions עם MongoDB:**

מבוסס על תיעוד רשמי של:

* express-session
* connect-mongo

**CORS מתקדם:**

מבוסס על דוגמאות מהתיעוד של ספריית cors.

**טיימר setInterval:**

מבוסס על שימוש סטנדרטי ב־Node.js לביצוע משימות מתוזמנות.

MongoDB:  
<https://mongoosejs.com/docs/connections.html>

SendGrid:  
<https://docs.sendgrid.com/>

Express Sessions:  
<https://expressjs.com/en/resources/middleware/session.html>

**.5 שימוש בכלי AI**

במהלך הפיתוח נעשה שימוש בכלי AI (כגון ChatGPT) לצורך:

* הבנת שגיאות
* כתיבת פונקציות
* שיפור קוד
* ארגון מבנה הפרויקט
* ייעול לוגיקת טיימר הבחינות

דוגמאות לפרומפטים:

* "Write Express session with MongoDB"
* "How to implement CORS in Node.js"
* "Create exam timer in Node.js"
* "Fix async await error in Express"
* "Create secure OTP system with expiration in Node.js"
* "Implement account lockout after 3 failed login attempts"
* "How to fix MongooseServerSelectionError ETIMEDOUT"

**קובץ index.js – הקובץ הראשי של השרת**

קובץ index.js הוא הקובץ המרכזי שמפעיל את השרת, מגדיר את כל ההגדרות הראשוניות, מחבר למסד הנתונים, ומנהל את פעולת המערכת.

**ספריות חיצוניות:**

http – יצירת שרת HTTP

express – יצירת שרת Web

compression – דחיסת תגובות לשיפור ביצועים

cors – ניהול הרשאות גישה בין שרת ללקוח

dotenv – טעינת משתני סביבה

express-session – ניהול סשנים

connect-mongo – שמירת סשנים במסד MongoDB

**קבצים פנימיים:**

Exam Model – עבודה עם אובייקטי בחינה

connectDB – חיבור למסד הנתוני

קבצי Routes – ניהול נתיבי ה־API

**async function start() { ... }**

תפקיד:

פונקציה זו אחראית על הפעלת השרת והכנתו לעבודה.

פעולות:

* יצירת חיבור למסד הנתונים באמצעות connectDB()
* יצירת שרת HTTP
* קביעת פורט העבודה
* הפעלת מנגנון בדיקת זמני הבחינות
* הפעלת השרת להאזנה לבקשות

**expressApp.get("/health", ...)**

תפקיד:

בדיקת תקינות השרת, בעזרת האתר uptimerobot ששולח כל 5 דקות בקשה לשרת

על מנת שהשרת יישאר פעיל ,זה בגלל שהשרת עושה coldstart כל 15 דקות.

פעולות:

* החזרת תשובה המאשרת שהשרת פועל
* הצגת זמן הבדיקה

**setInterval(async () => { ... }, 20000);**

תפקיד:

בדיקה מחזורית של זמן הבחינות הפעילות.

פעולות:

* שליפת בחינות במצב "running"
* חישוב זמן שנותר לסיום
* יצירת התראות כאשר נשארו:
  + 30 דקות
  + 15 דקות
  + 5 דקות
* עדכון מסד הנתונים

**expressApp.use(session({...}));**

תפקיד:

ניהול חיבורי משתמשים.

פעולות:

* יצירת Session למשתמש
* שמירת סשן במסד MongoDB
* ניהול Cookies מאובטחים
* הגבלת זמן התחברות

**expressApp.use(cors({...}));**

תפקיד:

הגנה על השרת מגישה לא מורשית.

פעולות:

* בדיקת כתובת הלקוח
* אישור בקשות מורשות
* חסימת בקשות לא מורשות

**תיאור לתיקיית routes:**

תיקיית routes אחראית על הגדרת נתיבי ה־API של המערכת.  
כל קובץ בתיקייה מגדיר קבוצת נתיבים הקשורים לנושא מסוים במערכת, ומפנה לפונקציות מתאימות מתוך ה־.Controllers

כל קובץ Routes כולל:

* יצירת Router באמצעות Express
* הגדרת נתיבי GET / POST / PUT / DELETE
* חיבור לפונקציות Controller
* שילוב Middleware כאשר נדרש

**auth.routes.js**

תפקיד:

ניהול תהליכי התחברות והרשמה.

נתיבים עיקריים:

* POST /login – התחברות משתמש
* POST /register – יצירת משתמש חדש
* POST /logout – התנתקות

**admin.routes.js**

תפקיד:

ניהול פעולות מנהל מערכת.

פעולות אפשריות:

* צפייה במשתמשים
* מחיקת משתמשים
* ניהול הרשאות
* צפייה בדוחות מערכת

נתיבים אלו לרוב מוגנים באמצעות Middleware של הרשאות.

**dashboard.routes.js**

**תפקיד:**

שליפת נתונים סטטיסטיים עבור לוח הבקרה.

**פעולות:**

* מספר בחינות פעילות
* נתוני תלמידים
* סיכומי פעילות

**exams.routes.js**

**תפקיד:**

ניהול בחינות במערכת.

**פעולות:**

* יצירת בחינה
* עדכון בחינה
* מחיקת בחינה
* שינוי סטטוס
* שליפת רשימת בחינות

**transfers.routes.js**

תפקיד:

ניהול העברת תלמידים או בחינות בין כיתות/חדרים.

פעולות:

* יצירת בקשת העברה
* אישור/דחיית העברה

**messages.routes.js**

תפקיד:

מערכת הודעות פנימית.

פעולות:

* שליחת הודעה
* קבלת הודעות
* סימון כהודעה נקראה

**chat.routes.js**

תפקיד:

ניהול צ׳אט בזמן אמת בין משתמשים.

פעולות:

* שליחת הודעות צ׳אט
* שליפת היסטוריית שיחה

**incidents.routes.js**

תפקיד:

דיווח וניהול אירועים חריגים במהלך בחינה.

פעולות:

* יצירת דיווח
* צפייה בדיווחים
* עדכון סטטוס אירוע

**reports.routes.js**

תפקיד:

ניהול דו"חות בחינה.

פעולות:

* יצירת דו"ח
* שליפת דו"ח
* עדכון דו"ח

**student.routes.js**

תפקיד:

ניהול נתוני תלמידים.

פעולות:

* הוספת תלמיד
* עריכת נתונים
* שליפת מידע אישי

**תיאור לתיקיית :models**

תיקיית models אחראית על הגדרת מבנה הנתונים של המערכת במסד הנתונים .MongoDBהמודלים נבנו באמצעות ספריית Mongooseהמאפשרת יצירת סכמות (Schemas) ואובייקטים במסד.

**Classroom.js**

תפקיד

ניהול חדרי הבחינה במערכת.

האוסף נשמר בשם classrooms.

שדות מרכזיים

* classroomId
* roomId
* name
* building
* floor
* capacity
* active

**TransferRequest.js**

תפקיד

ניהול בקשות להעברת תלמידים בין חדרים במהלך בחינה

שדות מרכזיים

* transferId
* examId
* studentId
* fromClassroom / toClassroom
* status: ["pending", "approved", "rejected", "cancelled"]
* requestedBy
* handledBy
* reasonCode
* lastError

**Exam.js**

תפקיד

ניהול בחינה מלאה – כולל נוכחות, אירועים, דו"ח וסיכום.

הנתונים נשמרים באוסף moodle.

מבנה עיקרי

פרטי קורס:

* courseId
* courseName
* examDate
* examMode (onsite / online)

זמני בחינה:

* startAt
* endAt
* extraTimeActive
* extraTimeMinutesPerHour

**User.js**

תפקיד

ניהול כלל המשתמשים במערכת:

* Admin
* Supervisor
* Lecturer
* Student

הנתונים נשמרים באוסף users.

מבנה עיקרי

פרטי זיהוי:

* fullName
* username
* email
* password

**תיעוד תיקיית :middleware**

תיקיית middleware מכילה פונקציות ביניים (Middleware Functions) הפועלות בין קבלת הבקשה מהלקוח לבין הפונקציה המטפלת בה (Controller).

תפקידן העיקרי:

* אימות משתמשים
* הגנה על נתיבי API
* הוספת מידע לבקשה (req)
* שליטה בהרשאות גישה

**requireAuth.js**

**תפקיד כללי**

ה־ Middleware requireAuth אחראי לוודא שמשתמש מחובר למערכת לפני מתן גישה לנתיבי API מוגנים.

הוא מבוסס על Session שנשמר בשרת.

הקובץ requireAuth.js אחראי על:

* בדיקת משתמש מחובר
* חסימת גישה למשתמשים לא מורשים
* יצירת אובייקט משתמש אחיד עבור Controllers
* שמירה על אבטחת ה־API

הוא מהווה שכבת אבטחה מרכזית במערכת.

**תיאור לתיקיית mailer:**

מטרה: שליחת מיילים חיצוניים באמצעות SendGrid API.

פונקציות:

sendOtpEmail()

* שליחת קוד OTP.
* שימוש ב-SendGrid REST API.
* אם אין API Key → DEV MODE מדפיס לקונסול

sendStaffCredentialsEmail()

* שליחת שם משתמש וסיסמה.

**תיאור לתיקיית db:**

**connectDB.js**

מטרה: יצירת חיבור למסד MongoDB Atlas באמצעות Mongoose.

פונקציה מרכזית:

connectDB()

שלבים:

1. קריאת MONGO\_URI מתוך .env
2. קריאה ל:
3. mongoose.connect(uri)
4. טיפול בשגיאות חיבור
5. הדפסת הודעת הצלחה

קישור למסד נתונים:  
MongoDB Atlas (Cloud)

מחרוזת חיבור:

mongodb+srv://<DB\_USER>:<DB\_PASSWORD>@cluster.mongodb.net/exam\_monitoring\_hw5

נשמרת בקובץ .env ואינה מועלת ל-GitHub.

**תיאור לתיקייתcontrollers:**

ה-Controllers אחראים על לוגיקת המערכת (Business Logic).  
כל Controller מטפל בתחום פונקציונלי נפרד בהתאם לעקרון Separation of Concerns.

1.1 auth.controller.j

אחראי על Authentication והרשאות.

פונקציות מרכזיות:

login(req, res)

* התחברות משתמש Staff (admin / supervisor / lecturer).
* בודק username + password.
* מיישם מנגנון חסימה (Lockout) לאחר 3 ניסיונות כושלים.
* יוצר session באמצעות express-session.
* שומר את המשתמש ב-MongoDB session store.

logout(req, res)

* משמיד session.
* מנקה cookie בשם sid.

me(req, res)

* מחזיר מידע על המשתמש המחובר.
* מבוסס על session פעיל.

register(req, res)

* יצירת משתמש חדש מסוג supervisor או lecturer.
* בודק ייחודיות username ו-email.

studentRequestOtp(req, res)

* שליחת קוד OTP לסטודנט.
* מייצר קוד אקראי.
* שומר hash של הקוד במסד.
* שולח מייל דרך mailer.js.

studentVerifyOtp(req, res)

* אימות OTP.
* בדיקת תוקף.
* יצירת session במקרה הצלחה.

staffForgotPassword(req, res)

* שליחת פרטי התחברות דרך מייל (לצרכי קורס בלבד).

1.2 dashboard.controller.js

* מחזיר נתונים סטטיסטיים עבור Dashboard.
* סופר בחינות, משתמשים, אירועים.
* משתמש בפעולות Mongoose כגון countDocuments.

1.3 exams.controller.js

* יצירה, עדכון, מחיקה ושליפה של בחינות.
* CRUD מלא.
* שימוש ב-Model Exam.

1.4 transfers.controller.js

* ניהול בקשות מעבר בין חדרי בחינה.
* בדיקת זמינות חדרים.
* עדכון סטטוס בקשה.

1.5 incidents.controller.js

* רישום אירועי משמעת.
* שמירת אירוע במסד.
* שיוך לסטודנט ולבחינה.

1.6 messages.controller.js

* מערכת הודעות פנימית.
* שליחת הודעות בין משתמשים.
* שליפת היסטוריית הודעות.

1.7 chat.controller.js

* ניהול צ׳אט בזמן אמת (במידה וקיים).
* שמירת הודעות בצד שרת.

1.8 reports.controller.js

* הפקת דוחות.
* חישובי סטטיסטיקות.
* שליפת מידע מצטבר.

1.9 student.controller.js

* שליפת מידע סטודנטים.
* עדכון פרטי סטודנט.
* חיבור בין Student Model ל-Exam.

1.10 admin.controller.js

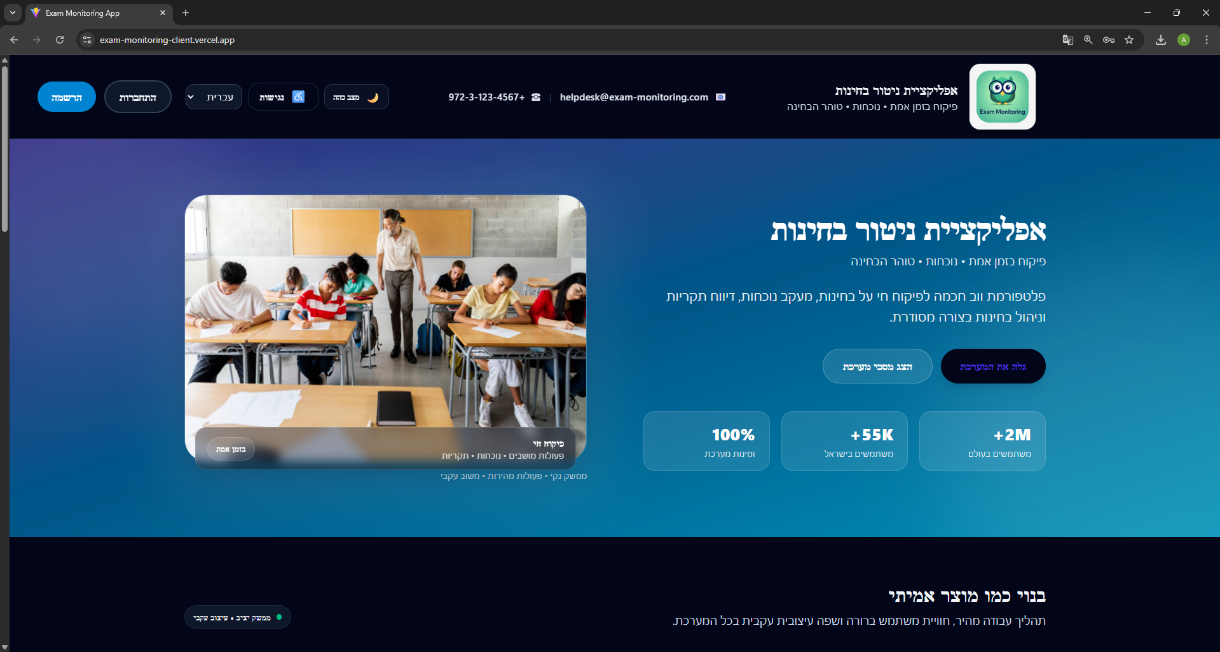
* ניהול משתמשים.
* מחיקה / שינוי הרשאות.
* שליטה מלאה על המערכת

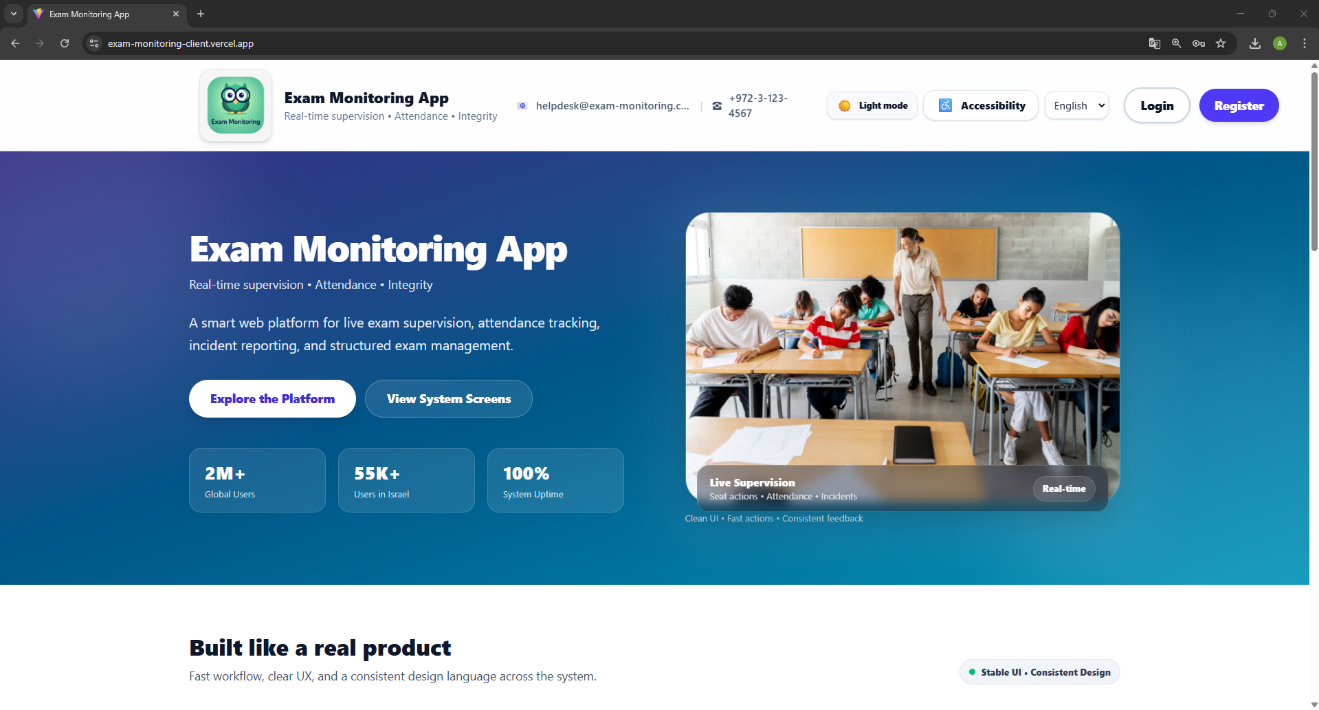
1. יש להגיש תיק משתמש הכולל צילומי מסך והסברים כיצד להשתמש באתר שלכם. נא לכלול סרטון של כדקה-דקה וחצי , שיווקי, המציג את האתר.

קישור לסרטון :

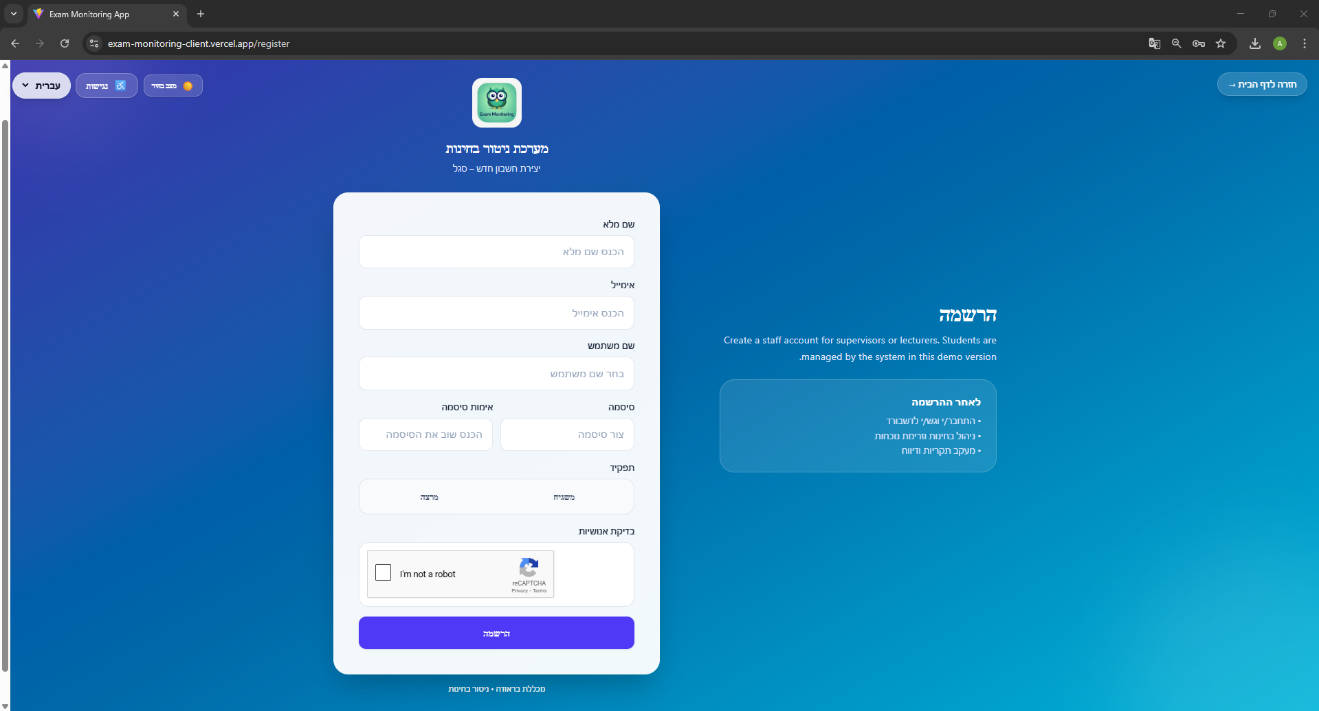
**העמוד הראשי של האתר**

**בראש העמוד אפשר לבחור שפה (אנגלית או עברית) גם יש אופציה לבחור במצב darkmode או מצב רגיל .**



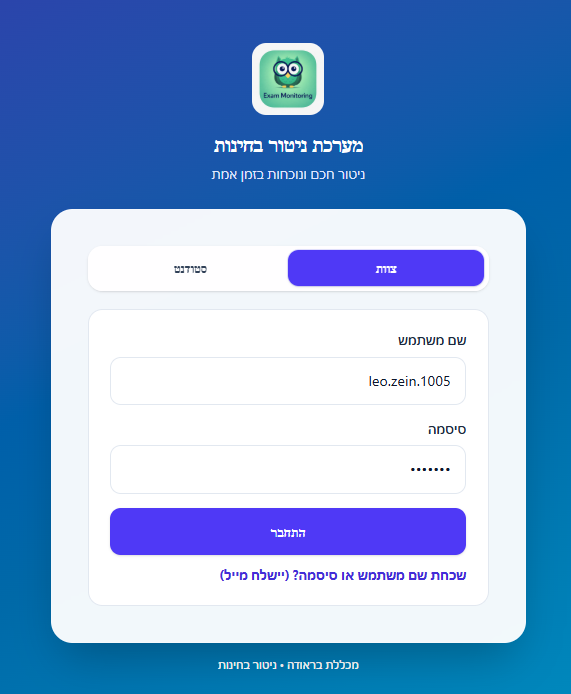


**2- נלחץ על כפתור הרשמה כדי שנעשה חשבון חדש**



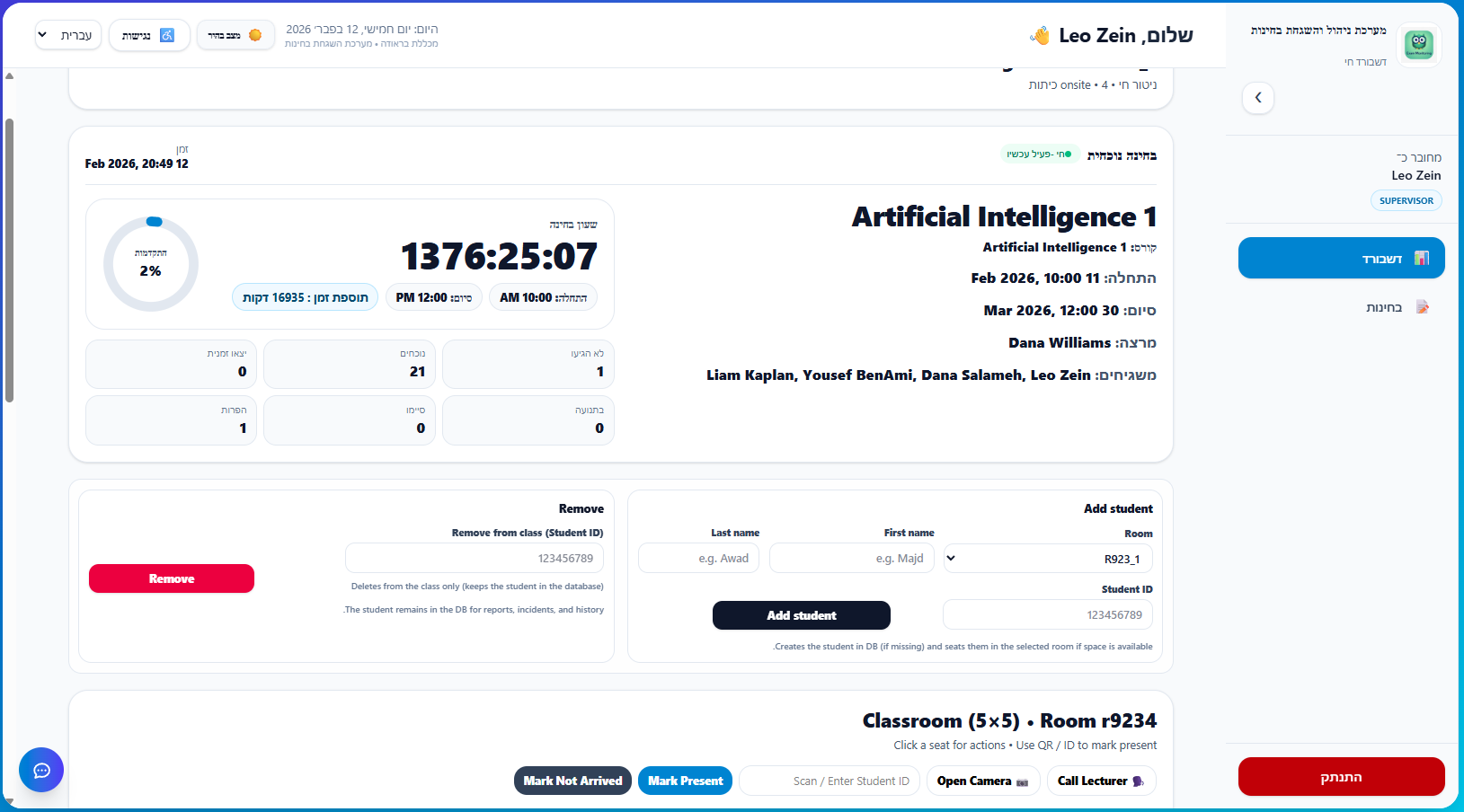
**נמלא פרטים שלנו ונעשה חשבון חדש.**

**3-נחזור לעמוד הראשי כדי שנתחבר למערכת כמשגיח (נתחבר עם החשבון של המרצה שקיים כי יש בו נתונים כגון בחינות וקורסים כי אם ניכנס עם החשבון שיצרנו עכשיו יהיה ריק לכן לא יהיה אפשרות לעשות הסבר עליו )**

****

**4- אחרי התחברות כמשגיח המסך הנפתח הוא הדשבורד מראה לנו את הבחינה הנוכחית (בצד שמאל) עם הטיימר שלה (השעון הזה הוא טיימר שידור חי לבחינה)**

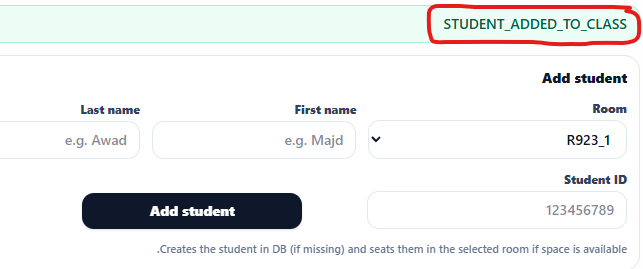
**Role : כל המשתמשים במערכת**



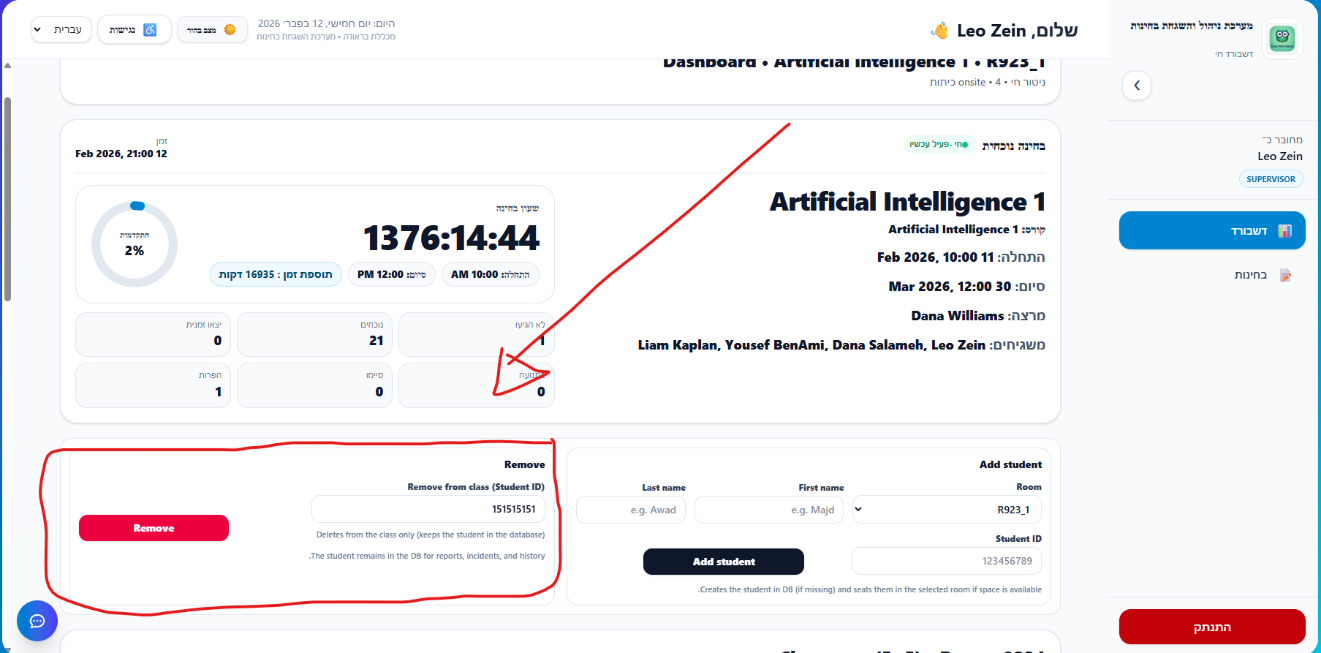
**4.1 בדשבורד המשגיח יכול להוסיף כל סטודנט לכיתה שלו (המשגיח יבחר את הכיתה וימלא פרטי הסטודנט וילחץ על כפתור הוספת סטודנט) והמשגיח יקבל הודעה לאחר הוספת הסטודנט על המסך**



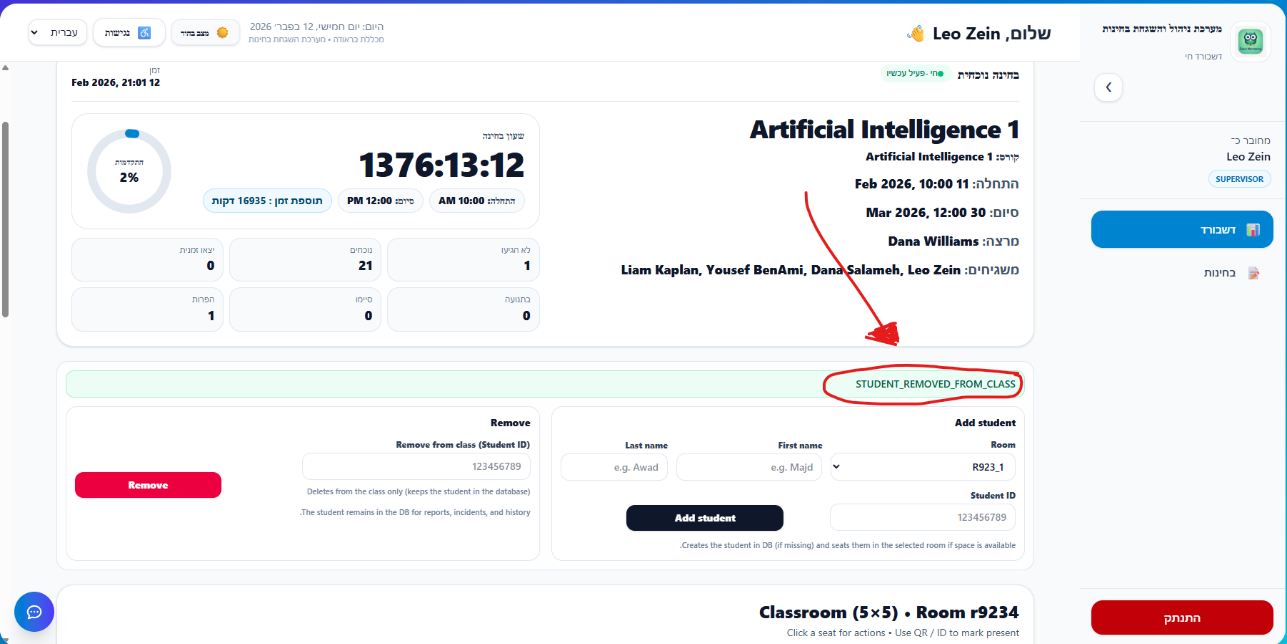
**והמשגיח יקבל הודעה לאחר הוספת הסטודנט על המסך**



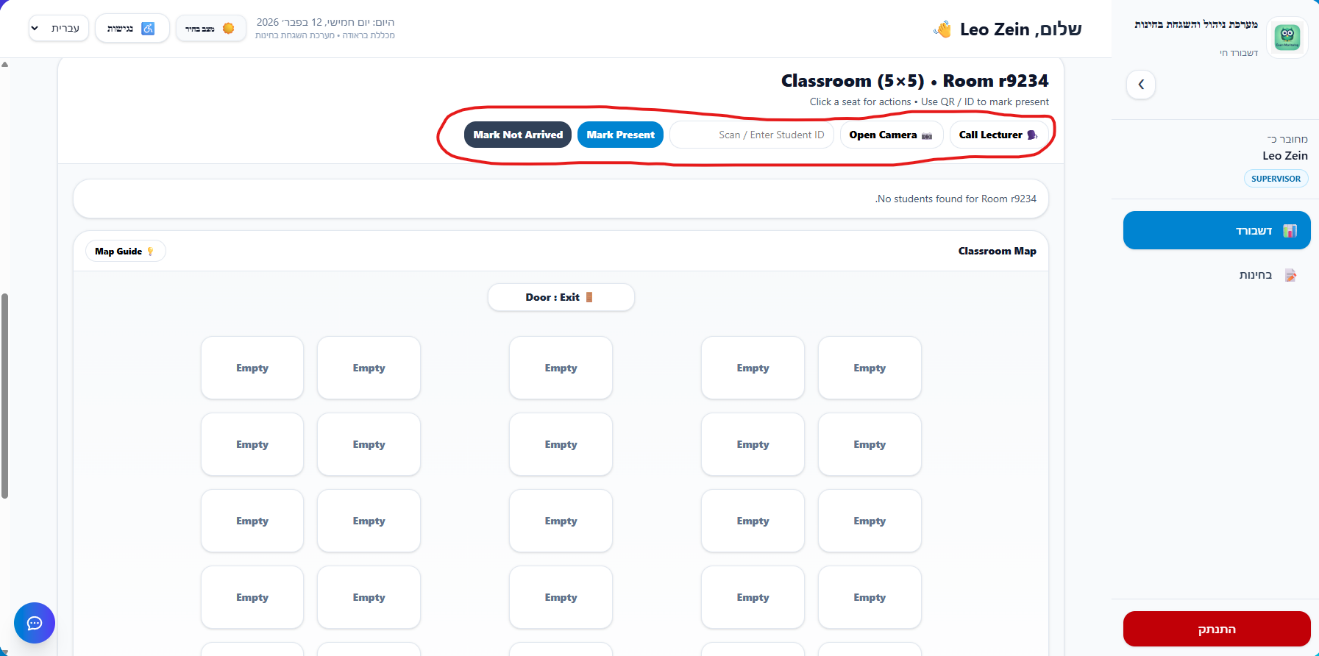
**4.2 גם המשגיח יכול למחוק סטודנט מהכיתה רק במילוי ת.ז שלו**



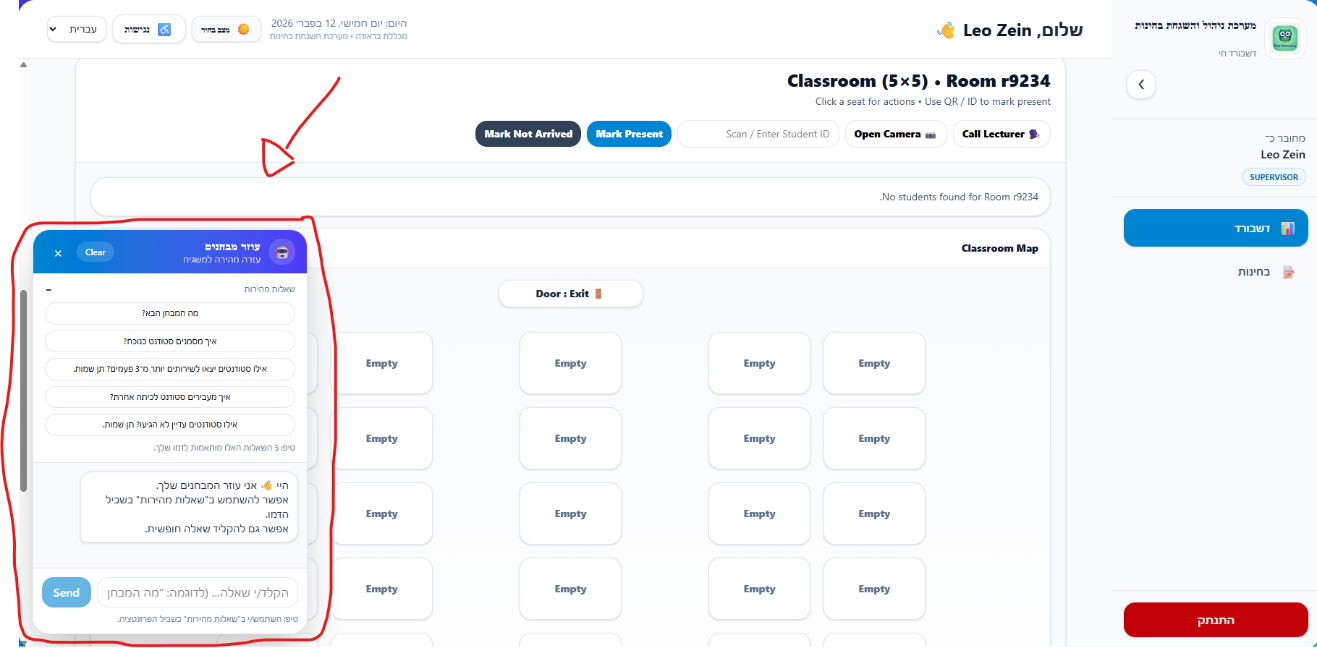
**וגם יקבל הודעה שהסטודנט נמחק מהכיתה**

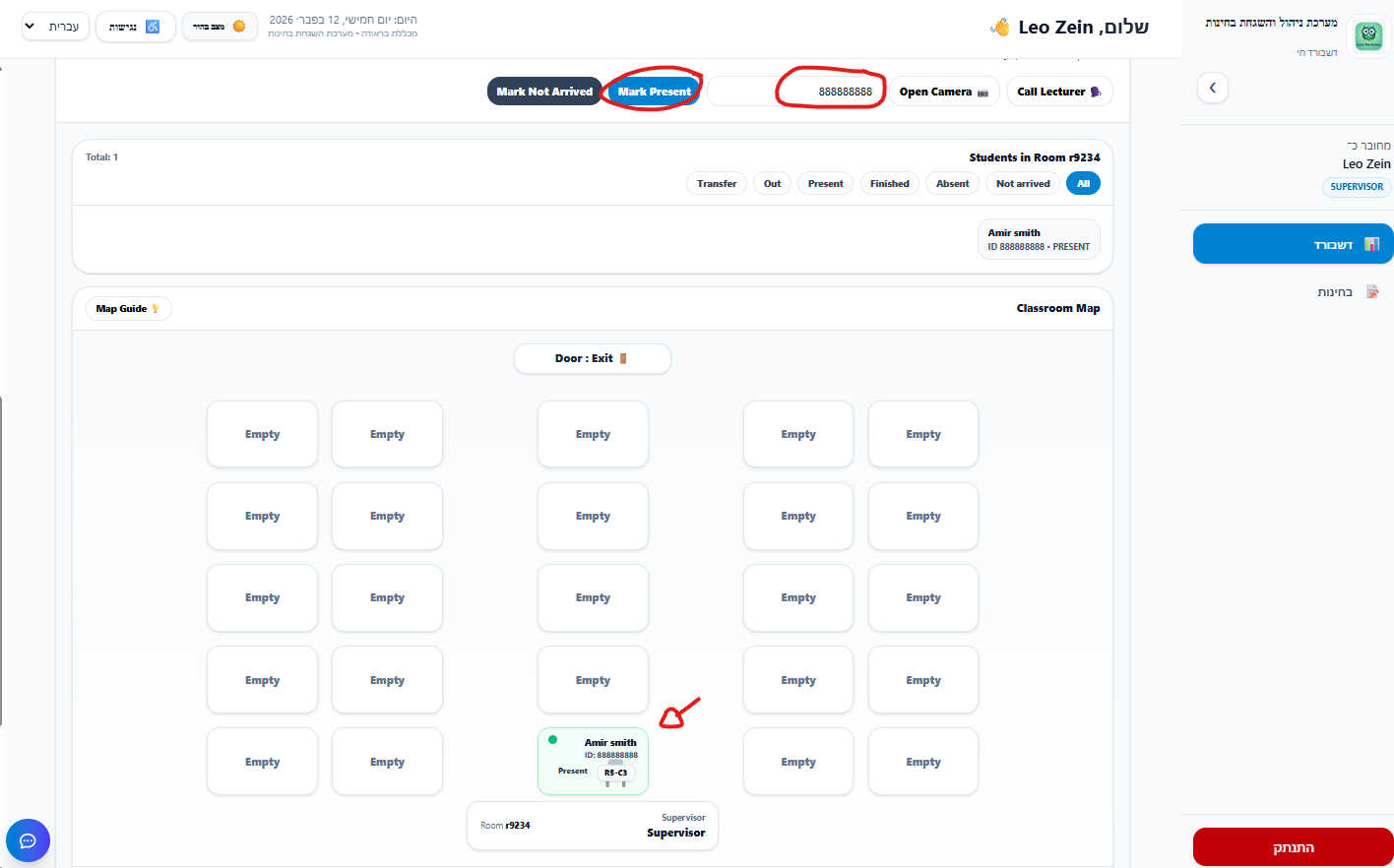


**5- נגלגל למטה בעמוד הדשבורד ויש אפשריות למשגיח – לקרוא למרצה – לפתוח מצלמה כדי לזהות סטודנט או יכול להזין את ת.ז של הסטודנט במקום הברקוד ויש שתי אפשריות לסמן סטודנט כנוכח או כנעדר.**



**5.1 שים לב יש בזווית התחתית השמאלית לוגו של הודעה , זה צאטבוט שעוזר למשתמש בלהשתמש במערכת אם הוא לא ידע איך לבצע פעולה מסוימת**



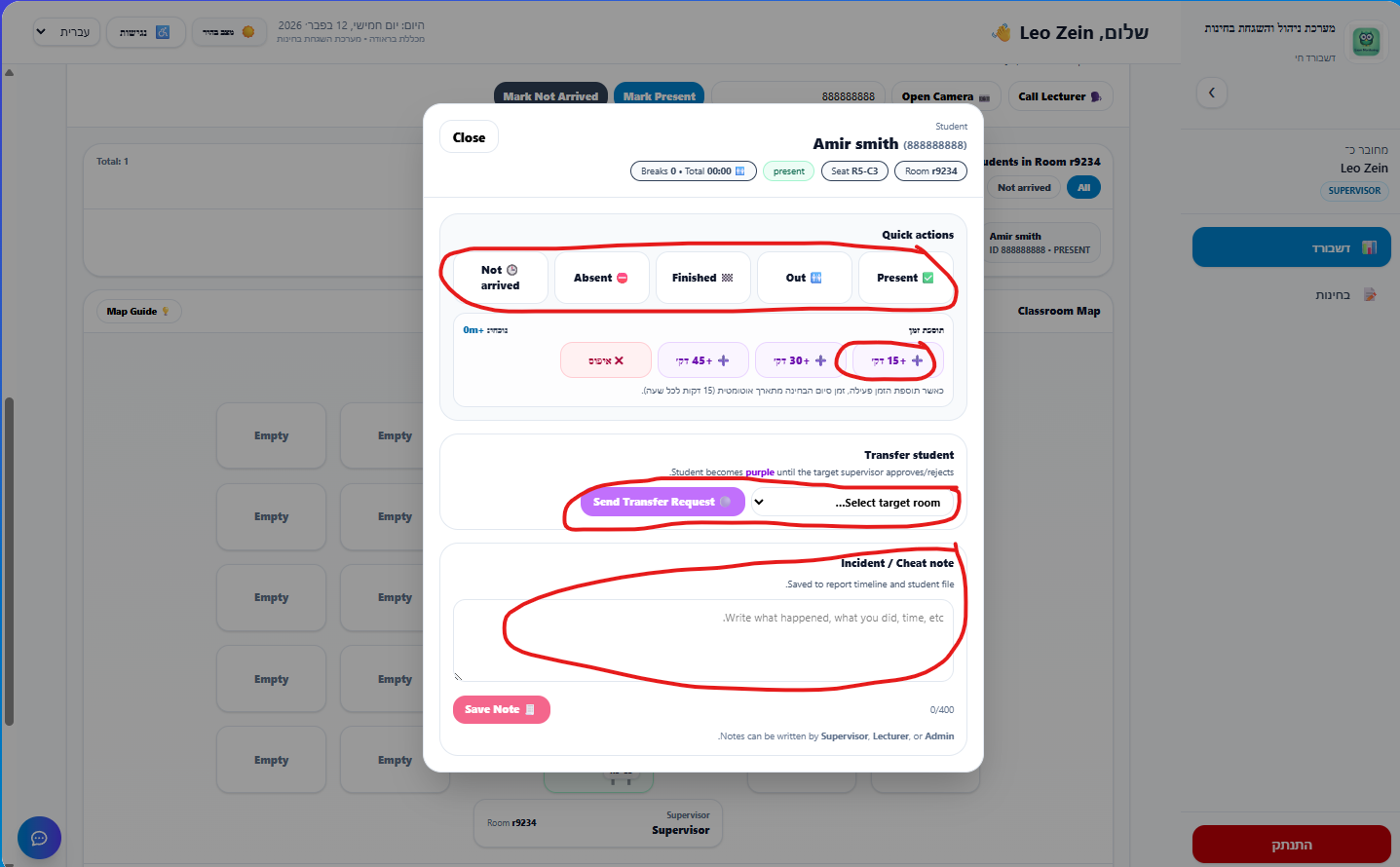
**6 אחרי שהוספתי סטודנט וסימנתי כנוכח , הוא התווסף אוטומטי למפת הכיתה והמערכת הושיבה אותו במקום באופן אוטומטי למקום שלו (המערכת קובעת מראש את כל סטודנט והמקום שלו)  
**

**6.1 כשלוחצים על הסטודנט במפה , יהיה למשגיח כמה אפשריות לשנות בסטודנט הזה**

**לסמן אותו אם יצא לשירותים – אם סיים את המבחן – אם נעדר – אם עדיין לא הגיע**

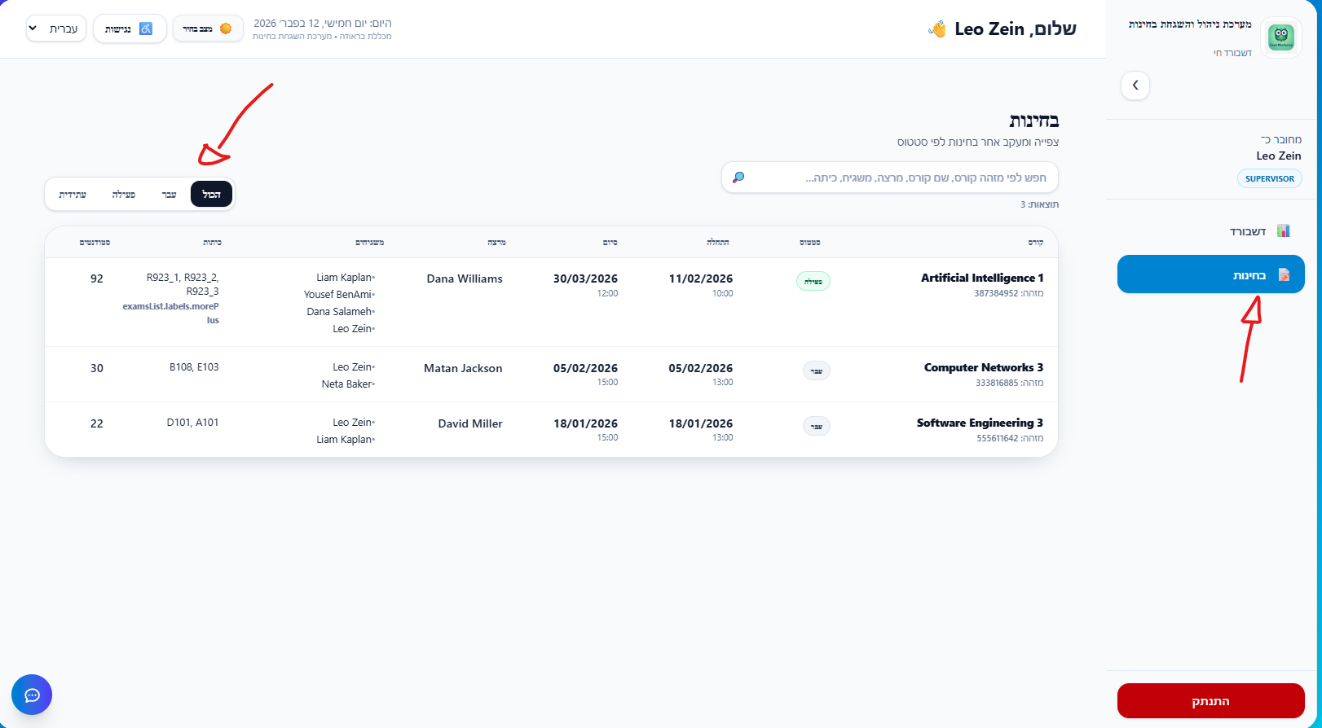
**לתת לו תוספת זמן (כמובן המרצה יגיד למשגיח אם אפשר לתת לו תוספת זמן )**

**או לבקש להעביר אותו לכיתה אחרת**

**או לכתוב עליו הערות**   


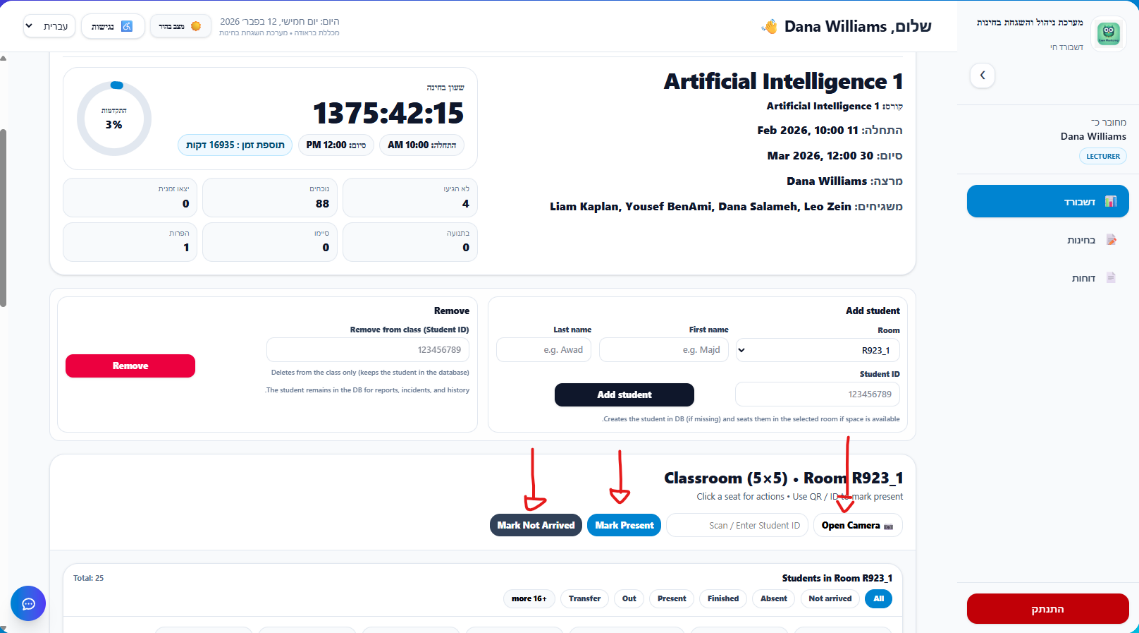
**7-** **נעבור לעמוד "בחינות"** : עמוד זה יציג רק את הבחינות של המשגיח , עם אפשרות לסנן בראש העמוד בצד שמאל (כל הבחינות או בחינות שעברו וכך ...) בעמוד זה אין פעולות רק הצגת מידע

Role : כל המשתמשים במערכת



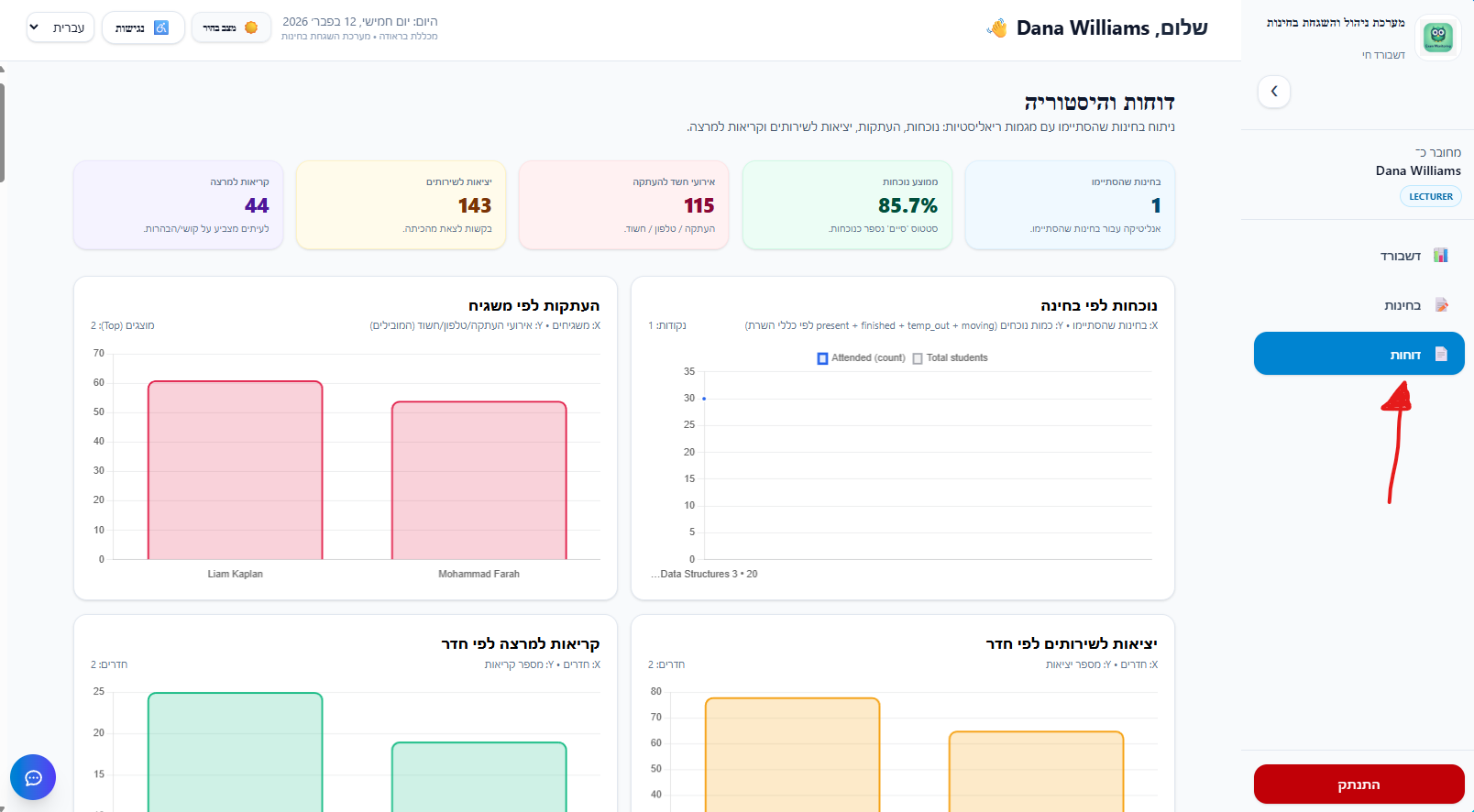
**8. כרגע נתחבר כמרצה**

**ונשים לב שאין את הכפתור שקורא למרצה בדשבורד שהיה אצל המשגיח**

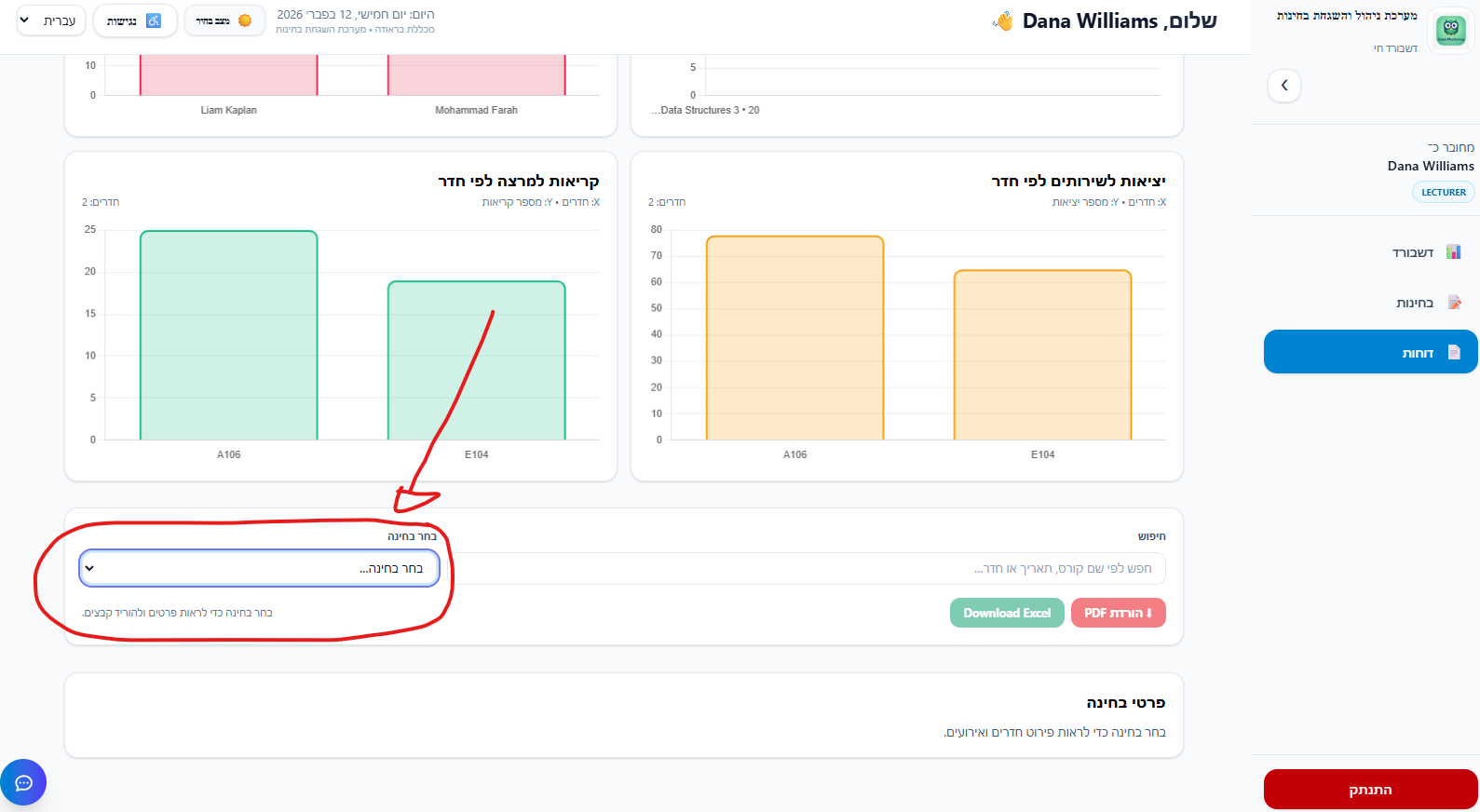


**8.1 ניכנס לעמוד הדוחות אצל המרצה :**

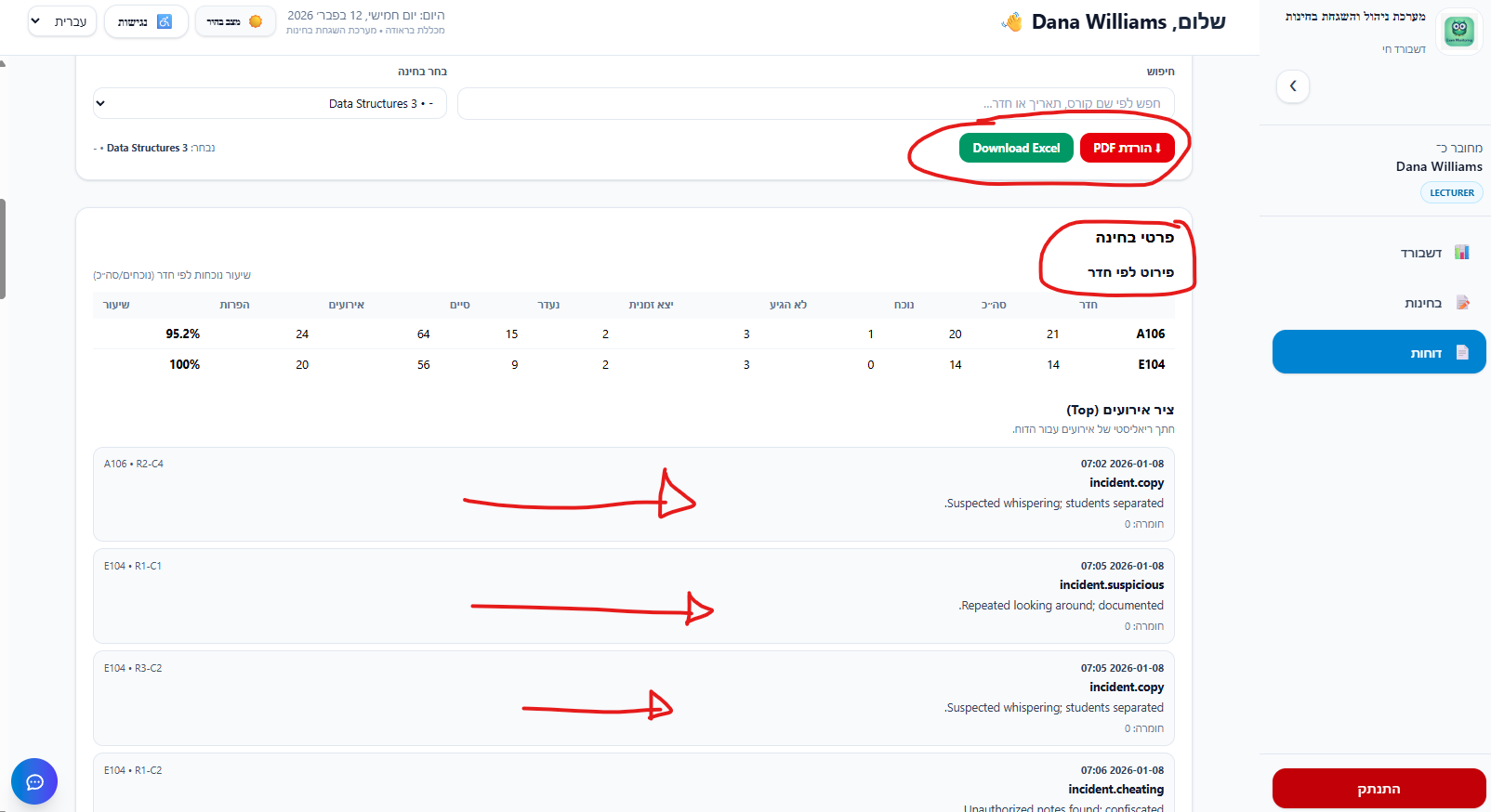
**Role : מרצה ומנהל המערכת**

****

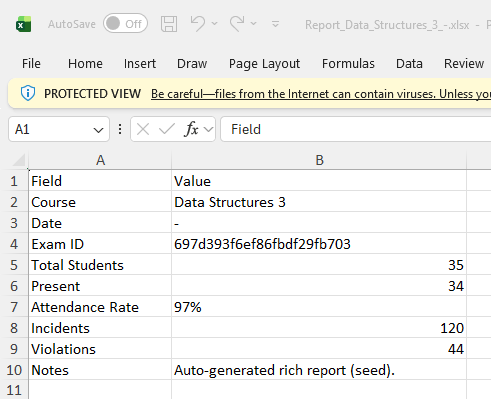
**8.2 נגלגל למטה בדף הדוחות ובחר את הבחינה שנרצה לראות את הנתונים שלה**

****

**8.2 אחרי שבחרנו בחינה יוצג לנו את הפרטים שלה וגם עם אפשרות להוריד את הדוח כקובץ אקסל או "פי די אף"**

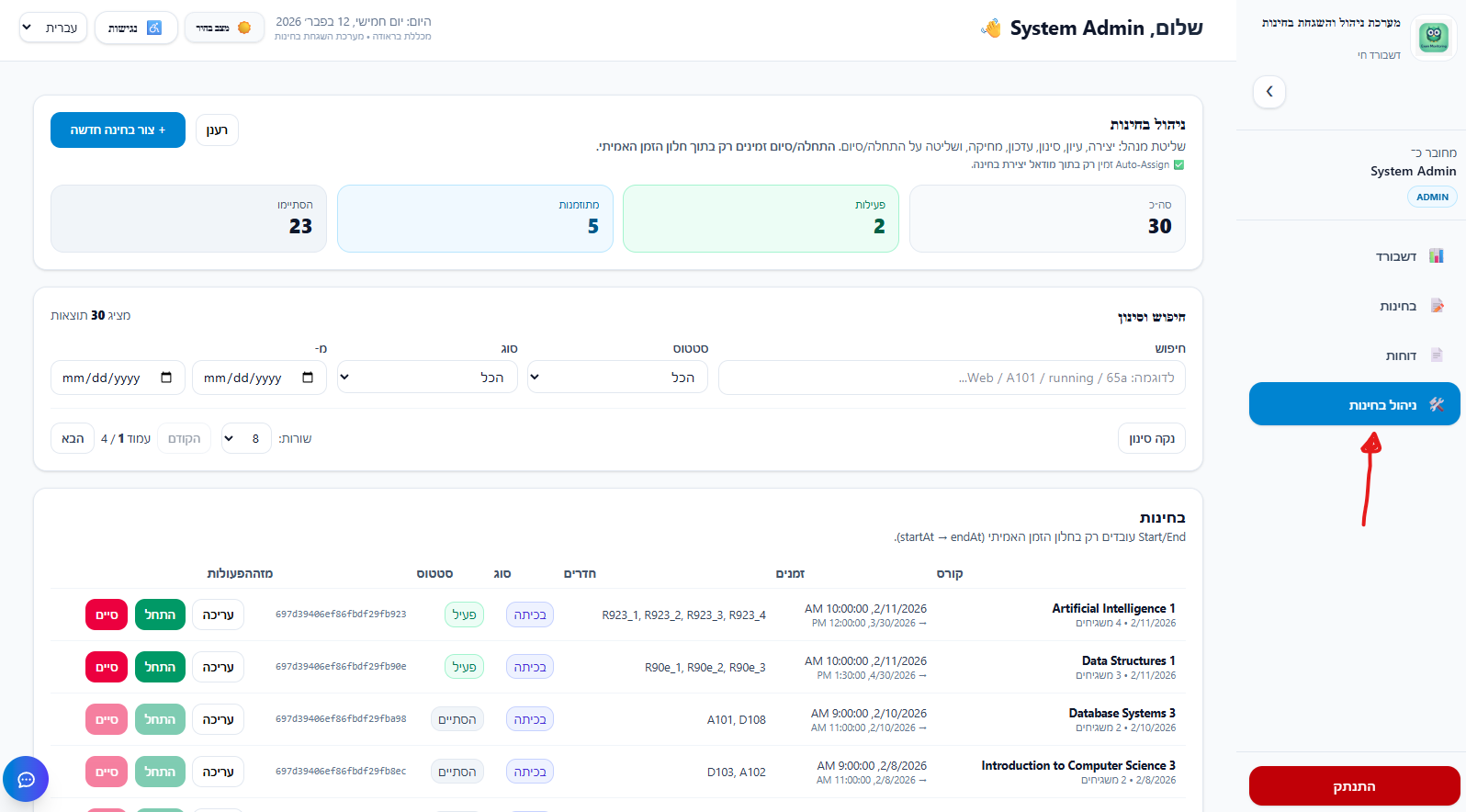


**הנה תכני קבצי הוורד והפי די אף שאפשר להוריד :**

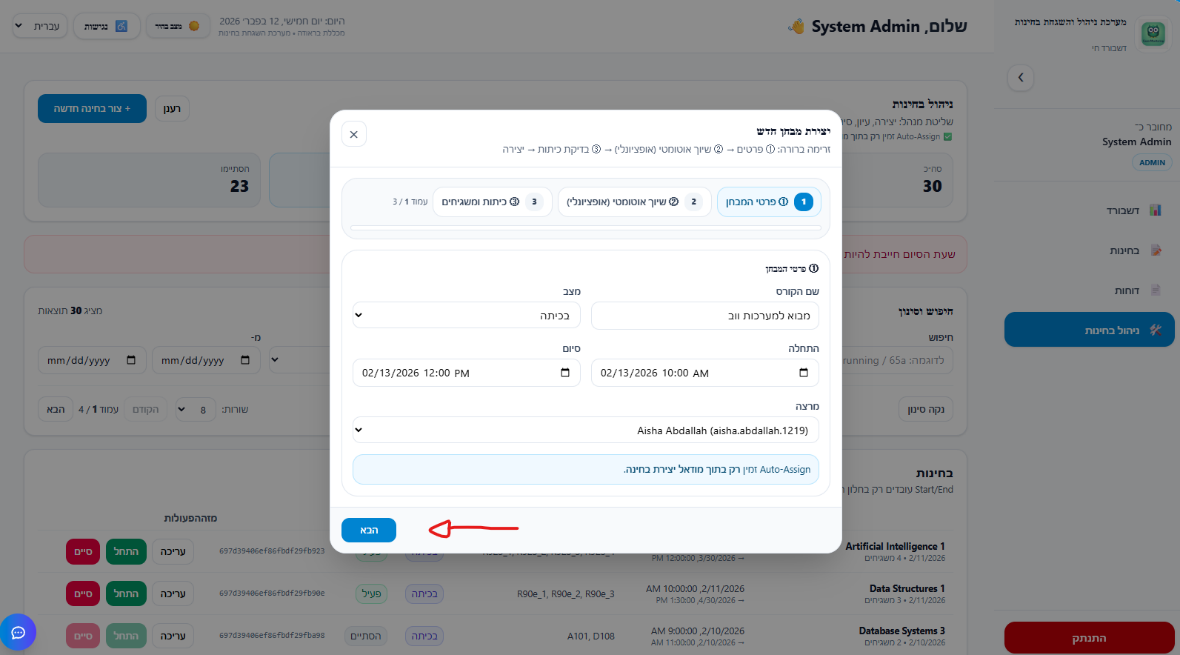


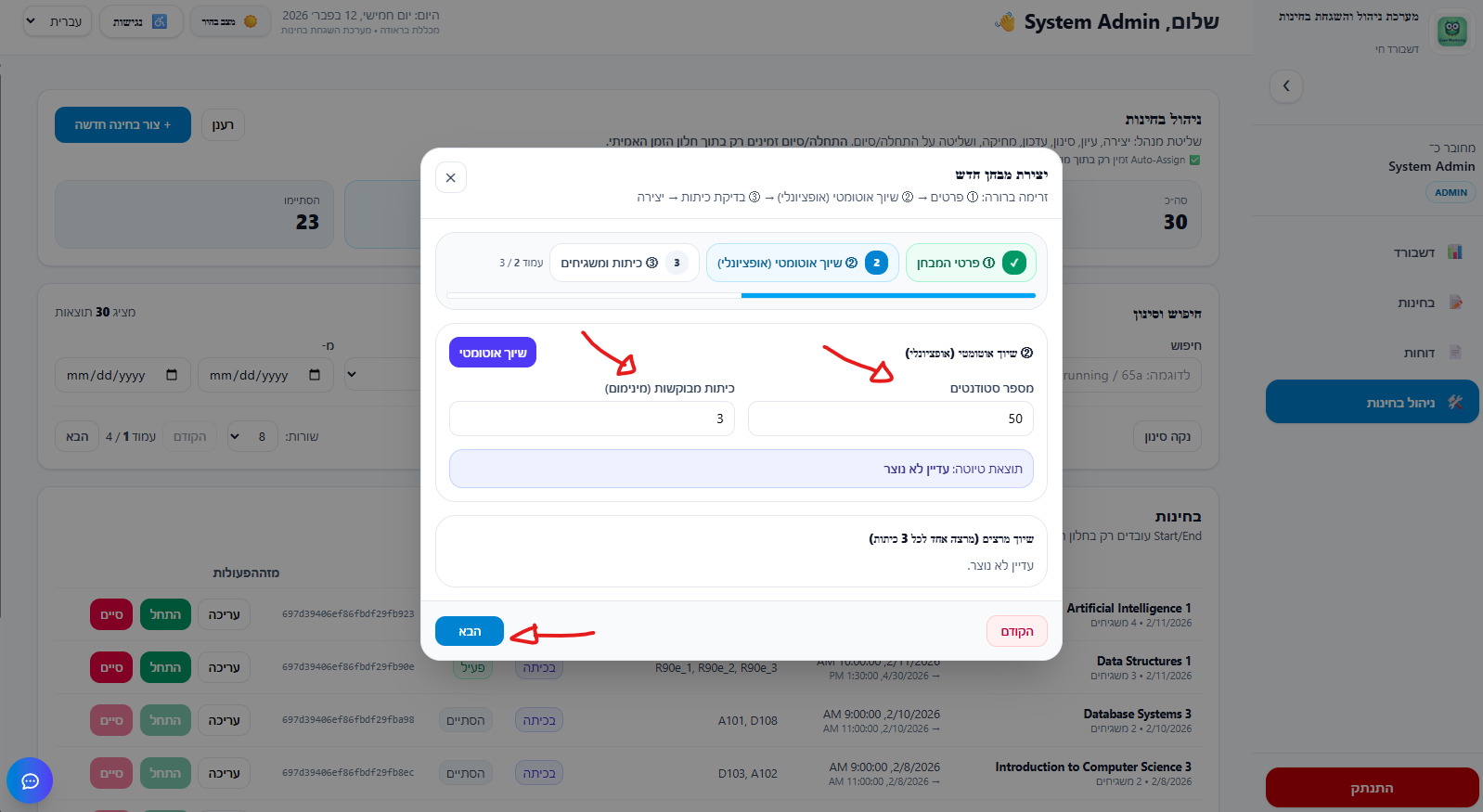
**9 . נתחבר כרגע כמנהל מערכת , ולמנהל יש 4 עמודים שהם : דשבורד , בחינות , דוחות , ניהול בחינות , את שלושת העמודים הראשונים הסברנו למעלה (אותם עמודים אצל המרצה)**

**9.1 נכנס עכשיו לעמוד "ניהול בחינות"**

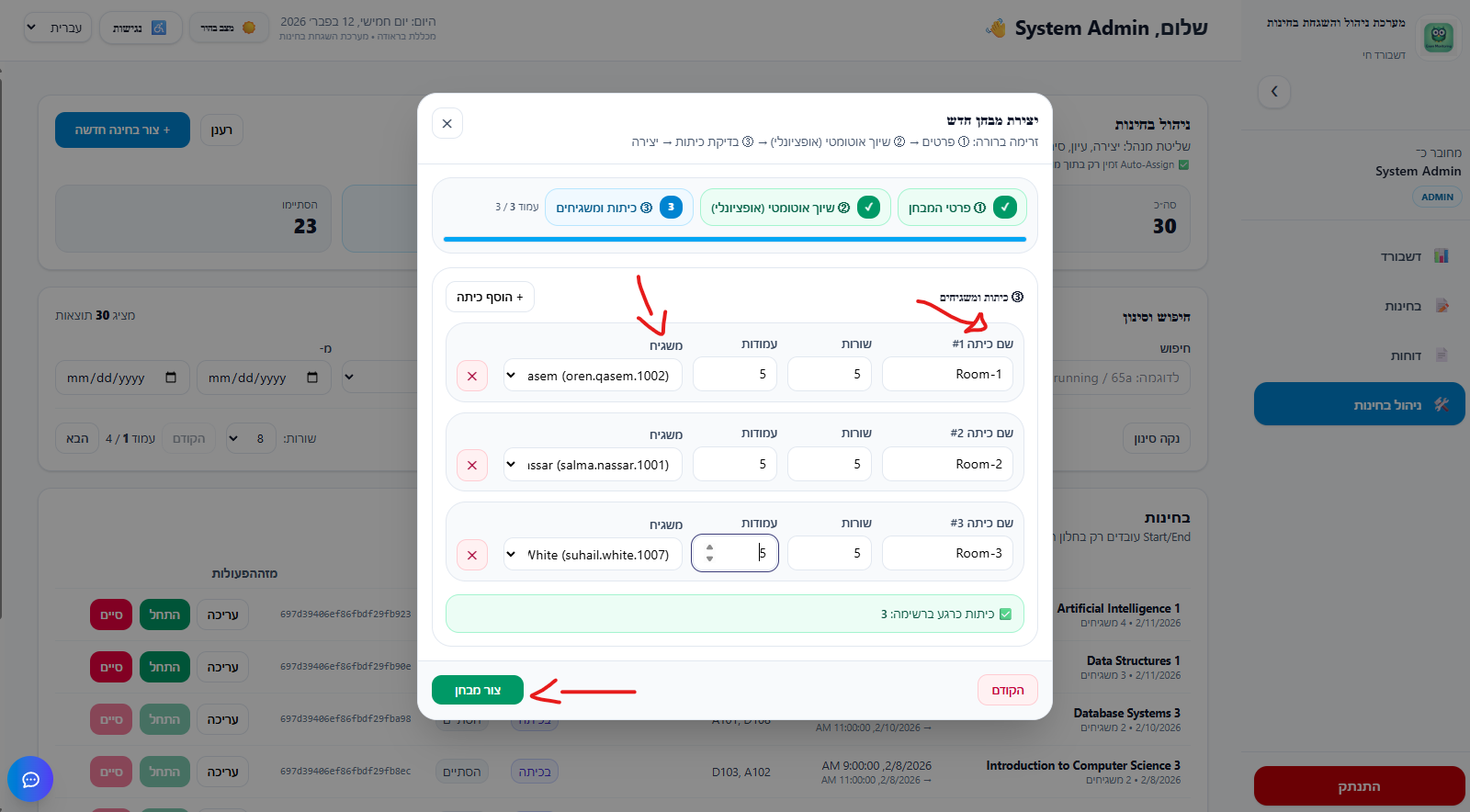


**9.3 נלחץ על הכפתור "צור בחינה חדשה" בראש העמוד בצד שמאל כדי להוסיף מבחן חדש במערכת , נפתח חלון בו צריך למלא פרטי המבחן ואז לוחצים "הבא"**

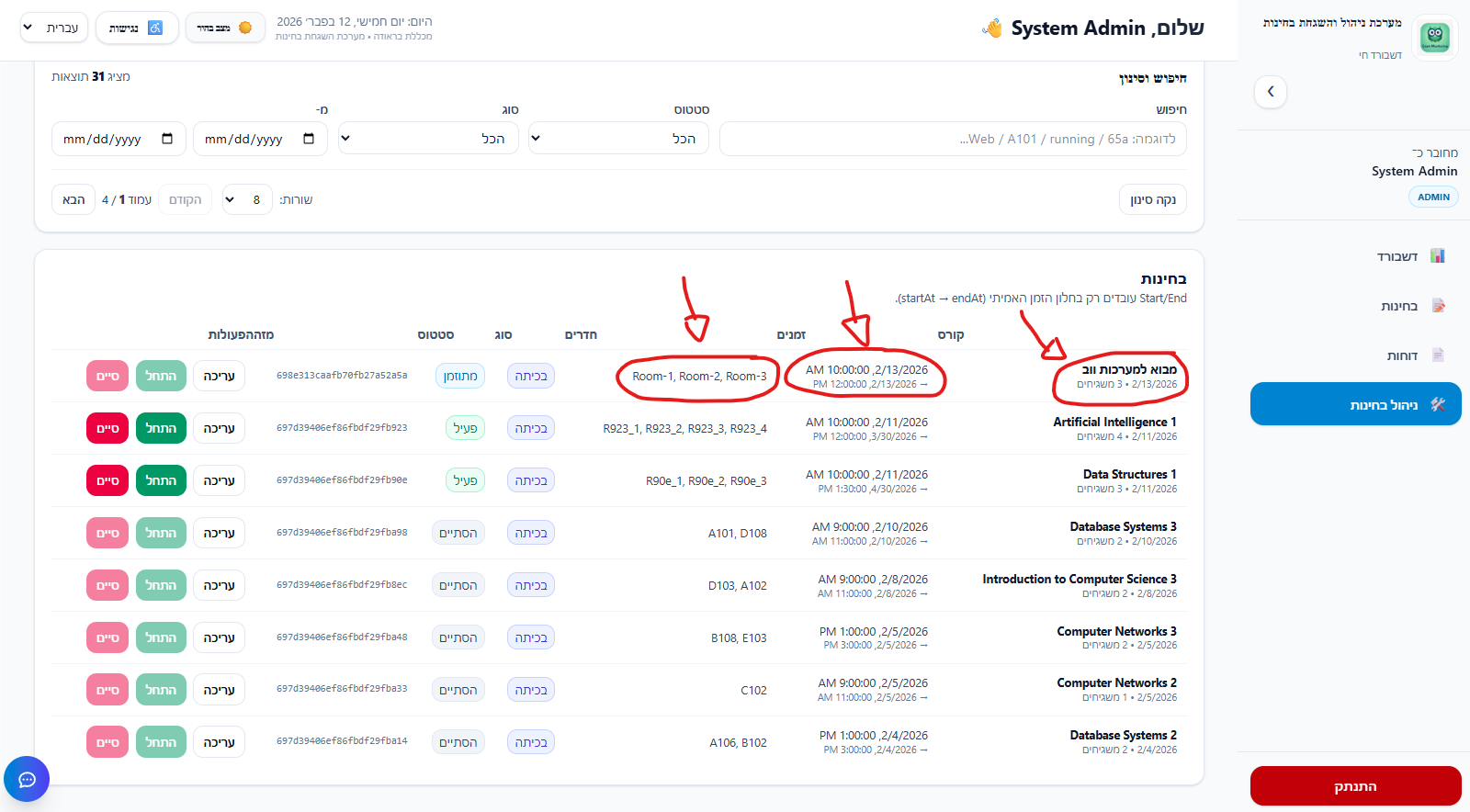


**9.4 אחרי שלוחצים הבא נמשיך במילוי הפרטים ולוחצים "הבא"   
**

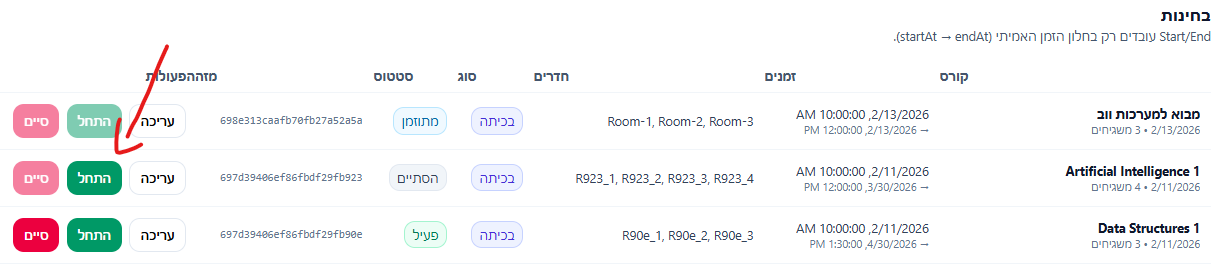
**9.5 מופיע לנו את החלק האחרון במילוי הפרטים (בו בוחרים כיתות ומשגיחים) ולוחצים "יצור מבחן"**



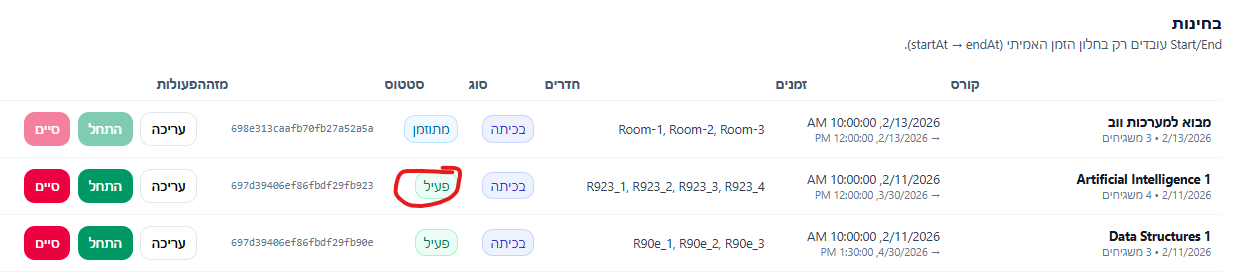
**9.6 אחרי הוספת המבחן נוכל לראות אותו באותו עמוד "ניהול בחינות" עם הפרטים שלו**



**הערה : רואים שמנהל מערכת יכול לסיים מבחן "פעיל" , מנהל המערכת משתמש בכפתור זה רק במקרים חריגים , כי כל בחינה מתחילה באופן אוטומטי בזמן שלה ומסתיימת בזמן שלה.**

דוגמה : עצרתי את המבחן השני ברשימה (וכמובן אני יכול ללחוץ על "התחיל" והמבחן ממשיך)  


**הנה הוא המשיך אחרי שלחצתי על "התחיל"**



9. בשבוע של 21.1 תציגו את תוצרי הפרויקט שלכם. כל צוות יציג 15 דקות את המערכת, וכן אלמנטים מרכזיים של העבודה (סעיפים 1-4 של מסמך זה).  
יש להציג את הקוד והאתר - אין צורך להכין מצגת!  
בסיום ההצגה נשאל את חברי הצוות שאלות בנוגע להצגה ולנושאי הקורס.

מטרת ההצגה היא לתת משוב ומיקוד לקראת הצגת הפרויקט.

יש להשתבץ להצגה בקישור הבא, חובת ההצגה על כל חברי הצוות!

להזכירכם – **ציון אישי של 30%** יינתן לכל אחד מהמציגים לפי הצגה ומענה על שאלות:

<https://docs.google.com/spreadsheets/d/1AAiXOpiXwto890SZs3tb5Pv2KITqEmgo/edit?gid=778935068#gid=778935068>

יש להגיש את כל התוצרים ב - GIT repository של הצוות , בתיקיית Project.

בנוסף, יש לוודא כי תיקיית ה- GIT ציבורית וכוללת את כל הקבצים של הפרויקט, כולל המסמכים, המצגת וכן הוראות הרצה.

# יש לכלול בתחילת מסמך הפרויקט את דף השער הבא: **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הנחיות הגשה:**

1.באחריותכם לוודא שהגשתם את כל התוצרים כנדרש ושהם הגיעו ליעדם.

לא תתאפשר בדיקה מחודשת של העבודה עקב טעויות בעת ההגשה!

2 .יש להגיש את התרגיל בקבוצות שהוגדרו בקורס.

3 .יש להגיש את התוצר הסופי של מסמך הפרויקט בפורמט WORD שייכלל בריפו של הפרויקט.

4.ניתן להפנות שאלות במייל לצוות הקורס, נא לשלוח שאלות לכלל הצוות . נא לציין את שם הקורס.

חלוקת ניקוד:

