

תאריך : _____

לכבוד
יחידת הפרויקטים
מה"ט

הצעה לפרויקט גמר

* יש להדפיס את כל הנתונים הנדרשים

א. פרטי הסטודנטים

שם הסטודנט	ת.ז. 9 ספרות	כתובת	טלפון נייד	תאריך סיום הלימודים
נريا דוד לוי	207786047	רימון 121 בית שאן	0522859094	

שם המכללה מכללה טכנולוגית כנרת סמל המכללה : 72228

מסלול ההכשרה : הנדסאים.

מקום ביצוע הפרויקט : מכללה טכנולוגית כנרת מגמת לימוד : תוכנה

פרטי המנהה האישי

שם המנהה *	טלפון נייד	כתובת	תואר	מקום העבודה/תפקיד	עובד/ה/תפקיד

* עבור מנהה אישי חדש יש לצרף קורTOT חיים, ניסיון מקצועי ותעודות השכלה לאישור מה"ט.

חתימת הגורם המקצועי מטעם מה"ט

חתימת המנהה האישי

חתימת הסטודנט

[Type here]

1. שם הפרויקט

BuildLocker

2. רקע

2.1. תיאור ורקע כללי

- בינויים מוגרים קיימת לעיתים קרובות בעיה של חוסר זמינות לכלי עבודה וכיוד תחזקה, עקב עלות גבוהה ושימוש מזמין בלבד. כיוון פתרונות קיימים כמו השאלה בין שכנים או השכלה מחנויות אינם נוחים, דורשים תיאום מראש ואינם מבוקרים.
- הפרויקט מציע מערכת לוגרים חכמה בבניין, המאפשרת לדירות לשאיל ולהציג ציוד באופן עצמאי באמצעות אפליקציה/אתר, תוך ניהול מלא של זמני שימוש ותשומים.

2.2. מטרות המערכת

- לאפשר לדירות גישה נוחה ומהירה לציוד לכלי עבודה מתוך הבניין.
- ליעל את תהליך ההשכרה באמצעות מערכת אוטומטית הזמינה 24/7.
- לספק למנהל המערכת כל ניהול: הוספה/עדכון ציוד, ניהול השכבות ותשומים.
- לאפשר חיבור לפי זמן שימוש בפועל.
- לשמר על בקרה ובטחה – פתיחת לוגר רק למשתמש מאומת ורישום פעולות.

3. סקירת מצב קיים בשוק, אילו בעיות קיימות

1. השכרת כלי עבודה מחנויות – דורשת נשיאה, זמינות מוגבלת, ותשלום ליום שלם גם עבור שימוש קצר.
2. השאלות בין שכנים / קבוצות יהילתיות – חסר אמון, אין בקרה על זמני החזרה, אין אחריות במקרה של נזק.
3. לוגרים חכמים קיימים (למשל לחבילות) – זמינים ויעילים אך אינם מותאמים להשאלת כלים (תחזקה, בלאי, ניטור).
4. ארונות ציוד משותפים בבניינים – ללא מגנון ניהול, נטייה לאיבוד או בלגן, ללא רישום מי לוקח מה.
5. מרכזי "ספריות כלים" עירוניים – דורשים הגעה פיזית, שעות פעילות מוגבלות ולא זמינים קרוב בבית.

בעיות עיקריות:

- חוסר נוחות וזמן
- היעדר בקרה וניהול
- תיאום ידני בין אנשים
- תשלום לא מדויק לפי זמן
- חוסר אחריות במקרה של נזק או אי החזרה

4. מה הפרויקט אמר לו חדש או לשפר

- **אוטומציה מלאה של תהליך השאלה** – שימוש בлокרים חכמים הנפתחים דרך אפליקציה/אתר ללא צורך במפגש פיזי.
- **חייב מדויק לפי זמן שימוש** – במקומות חיבוב יומי/תקופתי, המערכת מחשבת תשלום מהഫטיחה ועד ההחזרה בפועל.
- **בקורת מלאה על הציוד** – רישום מי לך,מתי, האם הוחזר, ומעקב היסטוריית שימוש.
- **шиיפור אמינות ובטיחות** – פתיחת לוקר רק למשתמשים מאומתים, שמירת לוגים ויכולת ניטור.
- **חיסכון כלכלי לדירות** – אין צורך שכל דיר קנה כלים יקרים לשימוש חד-פעמי.
- **נגישות 24/7 בבניין עצמו** – אין צורך בנסיעה לחנות או בתיאום עם שכנים.

5. דרישות מערכת ופונקציונאליות

5.1 דרישות מערכת

סביבת הטמעה ושימוש. שרידות, ביצועים והתמודדות עם עומסים.

סביבת הטמעה ושימוש :

- **שרת MySQL/Mongo.** ו Backend Node.js .
- **מספק/Web** אפליקציה לדירות ולמנהל.
- **локרים חכמים המוחברים לרשת.** (Wi-Fi / LAN)

שרידות והתמודדות עם עומסים :

- **תמיכה במספר משתמשים בויזמנית.**
- **מנגנון טיפול בתקלות תקשורת מול הלוקר.** (Retry)
- **שמירה עקבית של נתונים בסיס הנתונים.** (ACID / Transactions)
- **גיבוי נתונים תקופתי.**

5.2 דרישות פונקציונאליות

רשימת דרישות המשמש מהמערכת, מהן הפעולות בהן נדרש המערכת לתמוך

למשתמש רגיל (דיר) :

1. הרשמה והתחברות.
2. צפייה ברשימת ציוד זמין.
3. פתיחת לוקר וקבלת כל.
4. החזרת ציוד וסגורת השכלה.
5. צפייה בהיסטוריית שימוש ותשומים.

למנהל מערכת / מנהל בניין :

1. ניהול משתמשים – הוספה, עדכון והרשאות.
2. ניהול ציוד – הוספה, מחיקה, עדכון ומעקב מלאי.
3. ניהול לוקרים – סטטוס פתוח/תיקון/טיפול.
4. ניהול השכבות ותשומים.
5. צפייה בלוגים ופעולות משתמשים.

6. בעיות צפויות במהלך הפיתוח ופתרונות (תפעוליות, טכנולוגיות, עומס ועוד):

6.1. תיאור הבעיה- הלו כפועל יוצא של דרישות המשמש מהתוכנה.

1. תקשורת בין השרת ללקוחים החקמים
ייתכנו ניתוקים, חוסר יציבות רשת, או עיכוב בפתחת לוקר.
2. ניהול תקלות של אי החזרה ציוד משחמש עשויה לא להחזיר ציוד בזמן או להשאיר ציוד פגום.
3. עומסים על המערכת
שימוש של מספר רב של משתמשים בויזמנית, במיוחד בשעות עומס בבניין.
4. אבטחת מידע
גישה לא מושricht ללקוחים, גניבת זהות, או ניסיון לגשת ל-**API** ללא אימונות.
5. סyncron נתונים בין הצד לקוח לצד שרת
שינויי מצבים לוקר/ציוד בזמן אמת מהיבע עדכון מהיר ואמין.
6. ניהול תשלומים לפי זמן
יש צורך במדידה מדוקיקת של זמן פתיחה ועד סגירת לוקר.

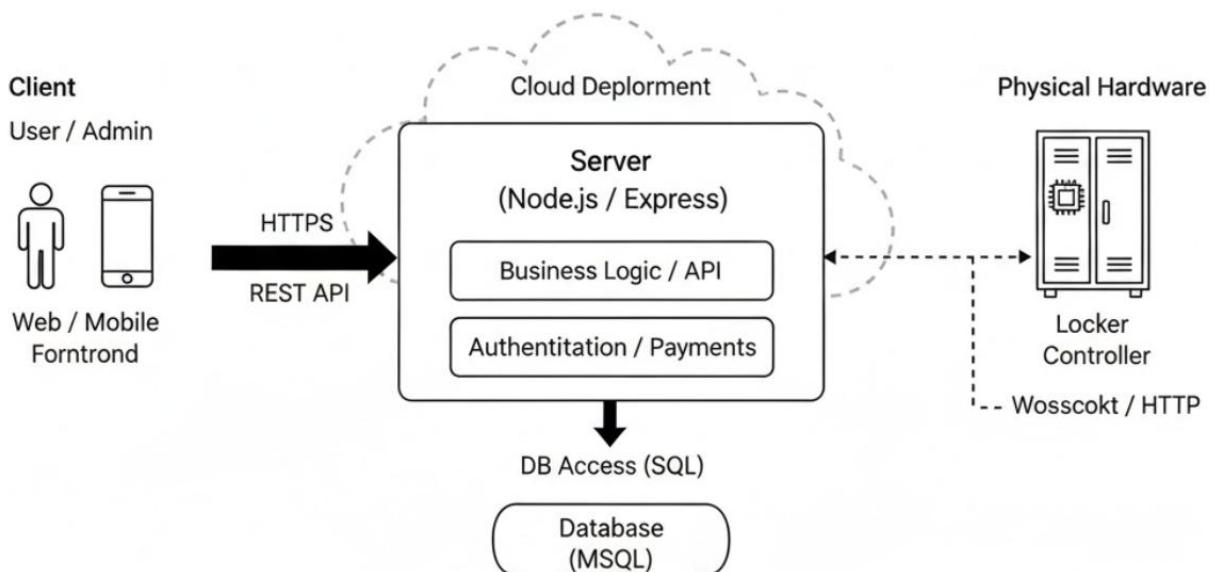
6.2. פתרונות אפשריים. (נא ציין פתרונות אפשריים וחולופות ארכיטקטוניות)

- שימוש ב- **MQTT** או **WebSockets** לשירות רציפה ויציבה בין השרת ללקוחים → אפשרות עדכון בזמן אמת וגם **fallback**.
- מגנון Retry + Queue במקרה של כשל תקשורת, הפקודה נשלחת מחדש עד לאישור תיקון.
- שימוש ב- **Caching**, **Load Balancing** ו- **Scaling** במקרה של עומסים – ניתן להרחיב את השירות בהמשך.
- אימונות משתמשים JWT + HTTPS הגנה על תקשורת המשתמשים והלקוחים.
- תיעוד לוגים ונטוור (Logging & Monitoring) במקרה של זיהות בעיות בזמן אמת.
- **חולופות:** שימוש בכלים כמו **PM2**, **Grafana**, **Kibana**,
מגנון מדידת זמן בשרת.
- השירות שומר זמן פתיחה וזמן סגירה → לא תלוי הצד לקוח.
חולופה: שירות צד שלישי למדידת טריביפים (לא חובה).
- ארכיטקטורה מודולרית(**MVC** / **Service-Repository**) מתקן תקלות והרחבת המערכת בעתיד.
- מילאה בניהול פרויקט, תיקון תקלות והרחבת המערכת בעתיד.

7. פתרון טכנולוגי נבחר:

7.1. טופולוגיה הפתרון - כלומר: פריסת המערכת , היכן יבוצע יישום המערכת (deployment), מרכיבי הפריסה. הניל ברמת מערכת (לדוgi פרויקט פיתוח אתר אינטרנט : המערכת מורכבת משרת, משק משתמש בצד הלקו,"s' DB, טווח תקשורת-אינטרנט, המערכת תיושם ברשת האינטרנט , יש להציג את דיאגרמת המערכת וכו')

Client–Server: המערכת בנויה בתצורת Frontend – אתר אינטרנט/אפליקציה, Backend Server (Node.js) – אחראי על ניהול המשתמשים, הלוקרים, התשלומים ופעולות לוגיות. Database – מאגר נתונים (MySQL / MongoDB) לשימושים ציוד, השכבות זמניות. Locker Controllers – בקרים פיזיים המחברים לאינטרנט ומקבלים פקודות פתיחה/סירה. המערכת נפרשת על שרת אינטרנט (מקומי/ענן) והлокרים מתקשרים מול השרת דרך HTTP/WebSocket.



7.2. טכנולוגיות בשימוש.(איזה ומדווע בכמה מילימ)

API: Node.js + Express •
- פיתוח מהיר, גמיש, נפוץ ל- React •
- משק משתמש מודרני, מהיר וקל לתחזקה. MySQL •
- שבירת נתונים בצורה עקבית ומסודרת. WebSocket / MQTT •
- תקשורת בזמן אמת עם לוקרים. JWT Authentication •
- אינומת מאובטח של משתמשים.

7.3. שפות הפיתוח:(איזה שפה ומדווע בכמה מילימ?)

- מאפשרת פיתוח מהיר בצד שרת וקל לשילוב עם הצד לקוח. JavaScript/Node.js •
- בניית UI דינמי וונת. JavaScript/React •
- להגדרת מבנה הנתונים ותקינותם. SQL •

7.4. תיאור הארכיטקטורה הנבחרת- הסבר בכמה מילימ מדווע

הארQUITקטורה בנויה לפי Routers + Services + Repository Pattern: •
- מקבל בקשות HTTP מהלקוח. Routers •
- מכיל את הלוגיקה העסקית (ניהול הזמן/ציוד). Service •
- אחראי על תקשורת עם בסיס הנתונים. Repository (DB Access) •

7.5. חלוקה לתוכניות ומודולים

- משתמשים (Users Module)
- ניהול ציוד (Items Module)
- ניהול לוקרים (Lockers Module)
- ניהול השכירות (Rentals Module)
- אוצר ניהול (Admin Module)
- Authentication & Authorization Module

7.6. סביבת השירות (מקומי, וירטואלי, ענן, שירות אירוח)

- ניתן להריץ מוקומית לפיתוח.
- בפריסה – אפשרות לשרת ענן (כמו Render, Railway, EC2).
- תמיכה ב HTTPS לתקשורת מאובטחת.

7.7. ממשק המשתמש/לקוק – GUI

The screenshot shows a user interface for creating a new user. At the top, there's a dark header bar with the title 'New User'. On the right side of the header, there are 'Hi, ליבם' and a profile icon. Below the header is a navigation sidebar on the left with the following items: 'Dashboard' (selected), 'New User' (highlighted in orange), 'Tools Shop', 'My Rentals', 'Profile', and 'Logout'. The main content area has a light gray background and features a form titled 'Create User'. The form contains four input fields arranged in a grid: 'ID (TZ)' and 'Full Name' in the top row, and 'Email' and 'Phone' in the bottom row. Below these fields is a 'Password' input field. In the bottom right corner of the form area is a red 'Create User' button. The overall design is clean and modern.

Dashboard

Welcome, nerya levi (super)

Total Buildings	Total Managers	Active Users
4	3	6

Managers

Managers List

Full Name	Email	Building ID
לי נירה דוד	neray@gmail.com	2
לי נירה דוד	nersdy@gmail.coma	3
לדכט	ndwj@njdsa.com	1

7.8. ממשקים למערכות אחרות / API :

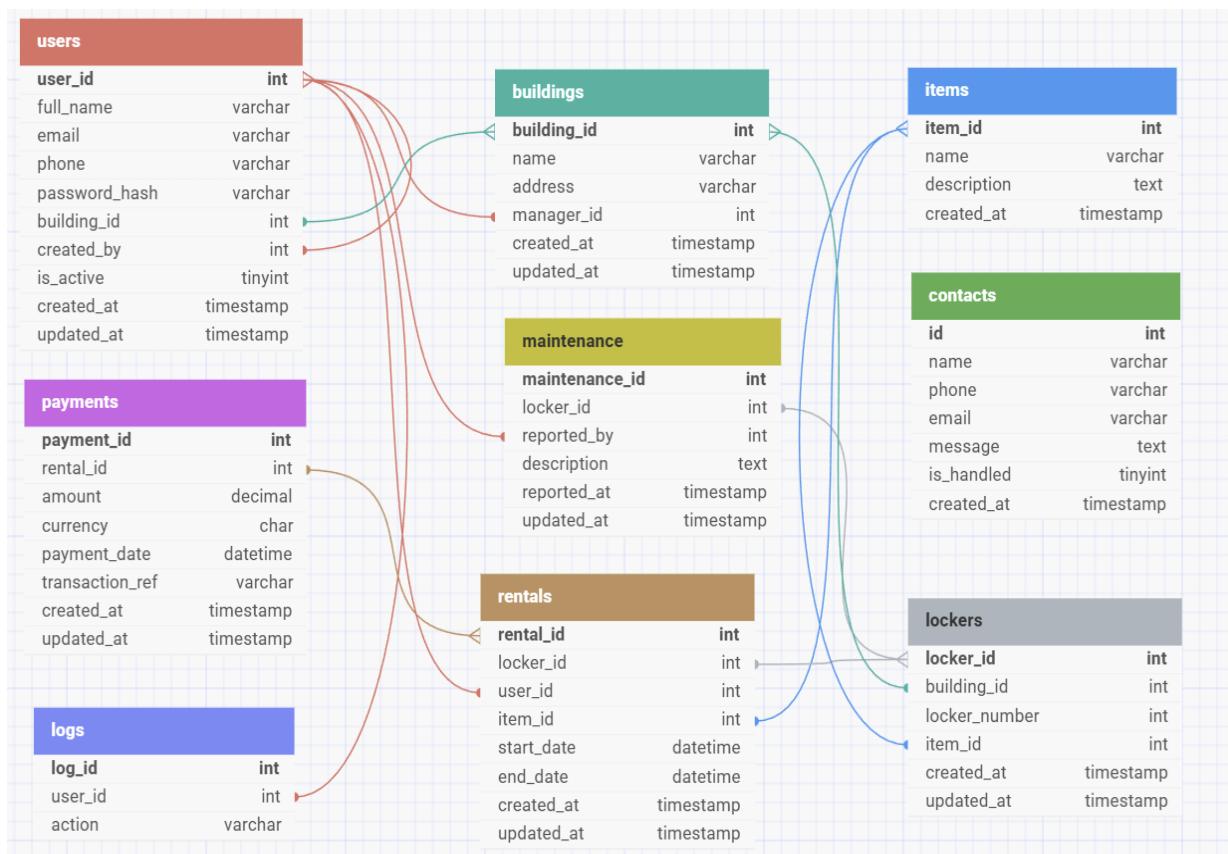
אין •

7.9. שימוש בחבילות תוכנה :

- **bcrypt** •
- **JWT** •
- **mysql2** •
- **fetch** •
- **React Query** •
- **CORS** •

8. שימוש במבנה נתונים וארגון קבצים

8.1. נא פרט את מבני הנתונים.



8.2. נא פרט את שיטת האיחסון (מאגר, קבצים ובאיוזה טכנולוגיה)

המערכת עושה שימוש בשיטת אחסון המבוססת על מאגר נתונים רלציוני (Relational Database) MySQL כפי שמוגדר בסכימת ה-SQL שלuproject.

א. מאגר נתונים MySQL – Database Storage – MySQL כל המידע במערכת נשמר במגילה טבלאות רלציוניות, הכוללות קשרים מסוג 1-N – בין ישויות הטכנולוגיה שנבחרה :

MySQL – בזכות יכולות, תמיכה ב- Foreign Keys, ביצועים טובים ויכולת Scalability. במאגר נשמרים הנתונים הבאים :

- users - נתונים משתמשים והרשאות
- buildings - פרטי בניינים ומנהלים
- lockers - לוקרים פיזיים בבניין
- items - פרטי ציוד הנמצאים בלוקרים
- rentals - השכירות פעילות והיסטוריות
- payments - תשלוםם עבור השכירות
- maintenance - תקלות ותחזוקה
- contacts - פניות משתמשים
- logs - רישום פעולות (לוגים)

אחסון קבצים (File Storage)

נכון לגרסה הנוכחית שלuproject

אין אחסון קבצים במערכת

(כמו תמונות, מסמכים או קונפיגורציות).

במידת הצורך בעתיד :

- ניתן להוסף תיקייה uploads בשרת AWS S3 : או להשתמש באחסון בענן (לדוגמה

נימוק לבחירת הטכנולוגיה

MySQL מתאים במיוחד לפרויקט של מערכת השכירה ולוקרים בלבד :

- ✓ תמיכה בקשרים בין ישויות(FK)
- ✓ שמירה על עקריות נתונים(ACID)
- ✓ נוח למימוש ותחזוקה
- ✓ זמינות גבוהה
- ✓ אינדקסים לשיפור ביצועים
- אחסון קבצים הוחלט שלא להשתמש בו בשלב זה, כדי לפחות את המערכת.

8.3. נא ציין מנגנוני התואשות מנפילה/កריישהו/תמיכה בטראנזקציות.

שימוש ב-(ACID) Transactions כדי למנוע פגיעה בעולות קרייטהיות כמו פתיחת השכירה, סגירה ותשלום מבוצעות בתוך טרנזקציה.

בנסיבות במקרה של כשל,

Foreign Keys / SET NULL UsNULL Cascading Keys – שומרים על עקריות הנתונים ומונעים רשומות יתומות במקרה של מחיקות או עדכוןים.

אינדקסים לשיפור ביצועים אינדקסים על שדות חשובים ממחוקים עומסים ומונעים קרישות בזמןן עומס.

גיבוי נתונים תקופתי – לאפשר שחזור המערכת במקרה של מחיקה או תקלת חמורה.

Checks Health ובדיקות זמן – בשרות DB (פועלות ויציבות).

מודולריות במערכת – תקלת במודול אחד אינה מפילה את כל השירות.

9. תרשימי מערכת מרכזיות

9.1 Use Case אוף השימוש

מנהל מערכת יוצר מנהל בגין

שמירת פרטי מנהל בגין מסד הנתונים
שיוך מנהל בגין

מנהל מערכת יוצר בגין בגין

שמירת פרטי בגין מסד הנתונים
עדכון רשימת בגיןים פעילים

מנהל בגין מתחבר למערכת

אימוח משתמש
טיענת הרשות מנהל בגין

מנהל בגין יוצר משתמש בגין בגין

שמירת פרטי משתמש במסד הנתונים
שיוך משתמש בגין

משתמש וגיל מתחבר למערכת

אימוח משתמש
טיענת פרטי משתמש

המשתמש צופה ברשימת ציוד זמני

B-S-שליפת ציוד זמני מה
בדיקות שמותרים אינם מושכרים

המשתמש בוחר ציוד להשכרה

בדיקה זמינות הציוד
(מני lock) נעילת הציוד

המשתמש מאשר ההשכרה

צירוף הזמנה במערכת
שיוך הזמנה ללקוח פנוי
רישום שעת התחלת ההשכרה
חישוב מחיר ראשוןי

המערכת שולחת פקودת פתיחה ללקוח

הלקוח נפתח
עדכון סטטוס: ציוד מושכר

המשתמש לוקח את הציוד

אישור פתיחת לקוח
עדכון סטטוס ההשכרה

"המשתמש בוחר "ה חוזרת ציוד

אישור ההזמנה הפעילה
הקצתה ללקוח החזרה
שליחת פקודת פתיחה ללקוח

המשתמש ניגש ללקוח

פתיחה ללקוח החזרה
התחלת מדידת זמן החזרה

המשתמש מחזיר ציוד וסגור ללקוח

אישור סגירת לקוח
רישום שעת סיום
חישוב זמן ההשכרה ותשלום סופי
עדכון סטטוס: ציוד זמני

התרשימים יתארו שני תרחישים מרכזיים:

דיאר: (User)

- התחברות למערכת
- צפייה בצויד זמין בלוקרים בבניין שלו
- בחירת לוקר / כליל וביצוע השכלה
- פתיחת לוקר וקבלת הצoid
- החזרת הצoid וסיום השכלה
- צפייה בהיסטורית השכבות ותשלומים אישית

מנהל בניין: (buildingAdmin)

- התחברות לאזורי ניהול
- צפייה בניהול משתמשים השייכים לבניין (אישור / חסימה / עדכון פרטיים בסיסיים)
- ניהול לוקרים בבניין: סטטוס (available/occupied/maintenance) שיווק לokers לבניין
- צפייה בהשכבות הפעולות וההיסטוריה השכבות בבניין
- צפייה בקריאות תחזקה (maintenance) וטיפול/עדכון סטטוס שלן

מנהל מערכת ראשי: (Super)

- התחברות למשק ניהול על (Global Admin)
- ניהול משתמשים כלל מערכת (יצירה, עדכון, הגדרת תפקידים והרשאות)
- ניהול בניינים (הוספה/עדכון/шиוך מנהל בניין)
- ניהול צoid: (Items) הוספה, עדכון, הסרה, שיווק צoid לлокרים
- ניהול לוקרים ברמת מערכת (הגדרת לוקרים חדשים למערכת)
- צפייה בכלל ההשכבות, התשלומים וקריאות התחזקה בכל הבניינים
- צפייה בלוגים ובפעולות מערכת לצורך בקרה ואבטחה

9.2 Sequence diagram - רצף קראות פונקציות מרכזיות בלוגיקה העסקית המרכזית של

הפרויקט

תרחיש: משתמש מבצע השכלה של כליל בלוקר



הזרת הסידור - Sequence

User → Frontend: ○

לוח “סימן השכלה / הזרה.”

Frontend → Backend: ○

בקשת PUT /rentals/{id}/complete.

Backend → Database: ○

עדכון end_date ○

חישוב סכום סופי (לפי זמן) ○

עדכון תשלום ל-field ○

עדכון לוקר ל-field ○

Backend → Frontend: ○

: “Rental Completed”.

Data flow 9.3

- משתמש → (Frontend) משתמש (User / Admin / Super)
• המשתמש מבצע פעולות כמו התחברות, צפיה בצד, פתיחת לוקר, ניהול לוקרים וכו'.
• הממשק שולח בקשות API לשרת.
• הרשות Frontend

2. Frontend → Backend (Node.js / Express) שרת

Backend: כל פעולה של המשתמש נשלחת ל-

- בקשות Register / Login
- בקשות להתחלה השכלה
- בקשות סיום השכלה
- ניהול משתמשים / לוקרים / ציוד
- פעולות ניהול ושליחת נתונים לניהול

השרת אחראי על לוגיקה עסקית, חישובים ואמונות הרשאות (JWT).

3. Backend → Database (MySQL)

ה Backend מבצע קראות קרייה/כתיבה לכל אחת מהטבלאות:

- users •
- buildings •
- lockers •
- items •
- rentals •
- payments •
- maintenance •
- contacts •
- logs •

המידע נשמר במבנה רלציוני עם Foreign Keys וטרנסקציות.

4. Backend → Locker Controller (לוקרים חכמים)

לפעולות כמו:

פתיחה לוקר

עדכון סטטוס לוקר

קבלת תגובה על פתיחה/תקללה

השרת שולח פקודות לлокרים דרך WebSocket / HTTP.

5. Locker Controller → Backend (הлокר מוחזיר לשרת)

- “Opened” / “Failed”
- סטטוס דלט
- תקלות תחזוקה

6. משותםש → Backend → Frontend →

השרות מוחזר תוצאות ועדיונים :

- החלטת השכירה
- פתיחה לוקר
- סכום לתשלום
- היסטוריית השכירות
- נתוני ניהול

סיכום זרימת המידע

User ⇄ Frontend ⇄ Backend ⇄ Database

10. תיאור המרכיב האלגוריתמי – חישובי

10.1. איזה בעיה בא לפטור, איך יפתרו?

אין

10.2. איסוף מידע וניטוחים סטטיסטיים (אנליטיקות)

המערכת אוספת נתונים ממספר טבלאות (rentals, payments, lockers, users) ומאפשרה ביצוע ניטוחים סטטיסטיים בסיסיים, לדוגמה:

- **שימושיות המערכת:**
 - מספר השכירות בתקופה (ליום/חודש/בניין).
 - ממוצע זמן השכירה לכל צלי / לוקר.
 - **ניתוח הכנסות:**
 - סך כל הכנסות לתקופה מסוימת.
 - הכנסה ממוצעת להשכרה.
 - השוואת הכנסות בין בניינים שונים.
 - **ניתוח ניצול ציוד:**
 - אילו lokרים/כליים בשימוש גבוה.(Top used)
 - אילו כלים כמעט ולא מושברים – זיהוי ציוד לא כדאי.
- נתונים אלה יכולים להיות מוצגים בדוחות או גרפים באיזור ניהול, ולסייע בקבלת החלטות לגבי הרחבת ציוד, שינוי תעריפים או הוספה לokers בבניינים פעילים בלבד.

11. תיאור/התיכשות לנושאי אבטחת מידע

נא לציין אזוריים הדורשים אבטחה, כגון : שרת, בקרת גישה לאתר, חשבונות משתמשים, מאגרי מידע וכיוצא ניתן מענה.

נא ציין מס' מקרים ותగובות להם ניתן מענה אבטחתי. **המערכת כוללת מספר אזוריים הדורשים אבטחה** בرمגה גבוהה, לאחר שהייה מטפלת במשתמשים, לוקרים חכמים, תשומות ונתוני בניינים.

אזוריים הדורשים אבטחה
שרת(Backend API)

- שימוש ב- HTTPS לכל התקשרות.

JWT Tokens. אימומת משתמשים באמצעות.

(SQL Injection / XSS) סינון בקשות (Validation) למניעת הזרקות.
(Role-Based Access Control)

: User / buildingAdmin / SuperAdmin. הפרדה בין תפקידי משתמש.

כל תפקיד יכול לבצע רק פעולות המותרות לו.

חסימת פעולות ניהול בפני משתמש רגיל.

חשבונות משתמשים

סיסמאות מוצפנות באמצעות bcrypt.

מנגנון חסימת משתמשים לא פעילים (is_active)

מאגר נתונים(Database)

שימוש ב- Foreign Keys לשימירת עקביות נתונים.

הרשאות DB מוגבלות – השרת בלבד יכול לקרוא/לכתוב.

גיבוי תקופתי ושהזור במקורה של קרישה.

локרים חכמים(Locker Controller)

שליחת פקודות מאובטחות מהשרת בלבד.

אימומת בין השירות לlokers כדי למנוע פתיחה לא מורשתית.

מקרה 1 – ניסיון גישה ללא התחרבות
תגובה :

ה- Backend בודק JWT אם חסר או לא תקין → החזרת שגיאת 401.(Unauthorized)

מקרה 2 – ניסיון של משתמש רגיל לבצע פעולה ניהול
לדוגמה : מחיקת לוקר או ניהול משתמשים.

תגובה :

בדיקת תפקיד (Role) בצד השירות → החזרת 403 (Forbidden).

מקרה 3 – ניסיון הזרקת קוד (SQL Injection)
תגובה :

כל שאלות DB מבוצעות באמצעות Prepared Statements. בצד השירות מונע הכנסת קלט מסוכן. בנוסף Validation

מקרה 4 – סיסמה שנגנבה או נחשפה
תגובה :

המערכת שומרת סיסמאות רק לאחר הצפנה (bcrypt), ולכן לא ניתן לשחזר אותן. בנוסף קיימת אפשרות להשבית משתמש (is_active=0).

מקרה 5 – ניסיון פתיחת לוקר לצורך לא חוקית
תגובה :

רף השירות יכול לשולח פקודות פתיחה ; כל פקודה נבדקת מול רשומות השכלה פעילה. ללא השכלה פעילה – הפקודה נדחתה.

מקרה 6 – נפילת שירות או קרייסטי DB
תגובה :

- שימוש בגיבויים אוטומטיים
- תמייכה בטרנסקציות למניעת נזק לנזונים
- אפשרות לשחזר מלא מטוזן Backup

12. משאבים הנדרשים לפרויקט:

12.1. מספר שעות המוקדש לפרויקט, חולקת עבודה ביו חברי הצוות

300

12.2 נדרש ציוד

- מחשבים:** מחשבים איסיים לכל חבר צוות לצורך פיתוח, קומפול ובדיקה.
שרת פיתוח/בדיקה: סביבת ענן בסיסית (גנו, AWS EC2, Render, Railway או AWS Database) להטמעת השירות וה-
מודול לוקר (אופציונלי אך מומלץ): לוח פיתוח פיזי דוגמת Arduino/Raspberry Pi/ESP32 עם מנעיל
-Locker Controller (Solenoid Lock): אלקטודוני Wi-Fi/LAN המדמה את פעולת ה-Locker Controller.

רשות: חיבור אינטגרטיבי יאפשר תקשורת בין הרשות לבין השירות ללוקר.

12.3. תוכנות נדרשות

VS Code, MySQL, Insomnia, IDE

12.4. ידע חדש שנדרש ללמידה לאורך ביצוע הפרויקט

- React Hooks**: שימוש ב-React Hooks מציב ניהול מצב ונתונים מתקדמות (כגון Query).

12.5. ספרות ומקורות מידע

W3
MUI

13. תכנית עבודה ושלבים למימוש הפרויקט

שלב 1 – אפיון וניתוח דרישות (Requirement Analysis)

- הגדרת מטרות הפרויקט
 - זיהוי תרחישים שימושיים (Use Cases)
 - הגדרת תפקידים משתמשים (User / buildingAdmin / SuperAdmin)
 - אפיון מבנה מסד הנתונים (DB)
 - אפיון תקשורת בין המערכת ללקוחים חכמים

שלב 2 – תכנון ארכיטקטורה (System & DB Design)

- תכנון ארכיטקטורת Client-Server
 - יישרת תרשימי ו-ERD, Use Case
 - תכנון API endpoints
 - בניית סכימת MySQL עם קשרים PK/FK
 - הרכבת Frontend ו-Backend אמצעים מוכרים

שלב 3 – הקמת מסד נתונים(Database Implementation)

- **ץ' זיהוי הלקוחות** – יזירת הטבלאות users, buildings, lockers, items, rentals, payments, maintenance, contacts, contacts logs

- הגדרת טרנזקציות וายידקסים
- בדיקת שלמות קשרים(Constraints)

שלב 4 – פיתוח(Node.js / Express)

- בניית שכבה API מלאה
- יישום לוגיקה עסקית:
 - השכבות
 - תשלומים
 - סטטוס ללקוחות
 - ניהול משתמשים
- יישום אימות משתמשים(JWT)
- פיתוח חיבור למאגר הנתונים
- פיתוח תקשורת עם הלקוחות(WebSocket / HTTP)

שלב 5 – פיתוח(React)

- בניית מסכי משתמש:
 - התחברות
 - דף ציוד זמין
 - ניהול השכבות
 - היסטוריות תשלומים
- בניית מסכי ניהול:
 - ניהול לקוחות
 - ניהול משתמשים
 - תקלות תחזוקה
- מסכי Super Admin: ניהול בניינים וציוד
- שימוש ב망שך MUI לעיצוב אחיד

שלב 6 – בדיקות(Testing)

- בדיקות יחידה – (Unit Tests) בדיקת פונקציות לוגיקה מרכזיות
- בדיקות תהליכי – (Full Flow) התחברות, השכבה, פתיחת לקוח, תשלום
- בדיקות תקשורת ללקוח – שרת
- בדיקות עומס ותגובה לביעוות רשת
- תיקון באגים ויפורור ביצועים

שלב 7 – אינטגרציה בין המודולים

- Frontend ↔ Backend ↔ Database
- שילוב מלא בין Frontend וBackend
 - בדיקת שליחת פקודות ללקוחות וקליטת תשובות
 - בדיקה מקיפה של תרחישים אמיתיים

שלב 8 – תיעוד

- תיעוד תהליכי הפיתוח
- כתיבת הספר הטכני
- הוספה תרשימים(ERD, Sequence, Use Case, Data Flow)
- הצגת הפרויקט ומסקנות

14. **14.1. תכנון הבדיקות שיבוצעו**

נא פרט בטבלה, בדיקות תהליכי ברמת משתמש בהן נדרש המערכת לעמוד .(full Flow)

14.2. נא פרט בטבלה, מס מייצג של בדיקות ייחודית למודולים המרכזיים בהן נדרש המערכת לעמוד. (unit test)

סודול מרכזי	מספר בדיקות ייחודיה	תיאור הבדיקה	תוצאה צפיה
התחברות משתמש (Login)	3	בדיקות עם סיסמה נכונה, לא נכון, משתמש לא קיים	הצלחה/שגיאה
הרשאות משתמש	2	בדיקה של גישה למודולים לפי סוג משתמש	רף משתמשים מורשים יכולם למשתמש
ניהול לקוחות	4	בדיקה של כניסה ללקוח, מחיקה, עדכון סטטוס, בדיקת לקוח תפוס	ЛОкер מתעדכן בהתאם ללקוח
ניהול ציוד	3	בדיקה של הוספה ציוד, מחיקה, עדכון ציוד נסוף/מעודכן/נמחק בהצלחה	הצלחה
השרתות ללקוח/ציוד	3	בדיקה של השכלה, החזרה, מצב לקוח לאחר השכלה	סטטוס נכון אחריו פעולה
הציגת היסטוריה	2	בדיקות הציגת היסטורית השכבות נתוניים מוצגים נכון לתאריכים	נתונים מוצגים נכון לפני תאריכים
התראות / הודעות	2	בדיקה של שליחת התראה במצב חריגת המשמש הودעה מתקבלת על ידי המשמש	הודעה מתקבלת על ידי המשמש

15. בקרת גרסאות (version control)

<https://github.com/Nerya253/BuildLocker>

חתימת המנהה האישי

חתימת הסטודנט

ב. הערות ראש המגמה במכללה

ג. אישור ראש המגמה

שם : _____ חתימה : _____ תאריך : _____

ד. הערות הגורם המקצועי מטעם מה"ט

ה. אישור הגורם המקצועי מטעם מה"ט

שם : _____ חתימה : _____ תאריך : _____

נספח להצעת הפרויקט:

קוויים מוחמים בבחירה נושאים והיקפי עבודה בפרויקט הגמר.

1. **דגשים ארכיטקטוניים ושיקולים במימוש :**
 - 1.1. מומלץ להתנסות בארכיטקטורות הקשורות שימוש בתצורת שרת ללקוח.
 - 1.2. שימוש ב- design pattern במודולי התוכנה השונים- באיזוריהם מתאימים.
 3. דגש על הפרדה בין לוגיקה עסקית לצד לקוח וצד שרת. (FrontEnd,Backend,ClientBL,ServerBL)
 4. חלופות ארכיטקטוניות נדרשות לתמוך או לספק מענה לדרישות כגון:
 - 1.4.1. תמיינה והתמודדות בויסות עומסים .
 - 1.4.2. תמיינה והתמודדות עם שיקולי אבטחה והגנה על מידע.
 - 1.4.3. תמיינה בשירידות והתאוששות מתקלה(טרנזקציות שמירה למאגר, והתאוששות)
 5. תמיינה בשיקולי חווית משתמש (צד ניהול מערכת הצד משתמש קצה)
 6. תמיינה היכן שנitin בניהול פרופילי משתמשים.
2. **שפת מימוש הפרויקט -**

ישנו משקל גובה במימוש הפרויקט ביותר משפת מימוש אחת לפרויקט , תוך מתן דגש ליתרונות היחסיים של כל שפה, עבור מודול תוכנה המכלי הפרויקט.
למשל במקרה של תצורת שרת ללקוח (אתר אינטרנט) :
 - 2.1.1. לצד ניהול העסקי של השרת, בחירה בשפות עיליות JAVA, C# או nodeJS.
 - 2.1.2. לתכונה חישובית אלגוריתמית- מימוש בשפת native C, C++
 - 2.1.3. לצד לקוח AngularJS,Asp.net וכו'
3. **מאגר נתונים :Database**

3.1. ישנה חשיבות גבוהה להתנסות בעבודה עם מאגרי נתונים למשל, מאגר רלציוני ומ Lager FS Based .
למשל:
 - 3.1.1. עבור מאגר רלציוני נבחר ב-' Sql server, Sqlite, etc
 - 3.1.2. עבור מאגר לא רלציוני נבחר ב- mongoDB או NoSql

3.2. ישנה חשיבות רבה להגדרת שכבות גישה למאגר הנתונים כזו שתנהל מרכיבים טרנזקטיביים וסנכרון. נין להשתמש גם במסגרת frameworks קיימים כדוגמת dotNet.

3.3. ישנה חשיבות גבוהה להתנסות של התלמיד והتمודדותו עם יכולות חישוביות במסגרת מכלולי הפרויקט. ניתן לשלב היבטים אלגוריתמיים או לחילופין ניתוחים וחיתוכים סטטיסטיים בסיסיים מעל מאגר נתונים, למשל :

 - 4.1. במקרה של אתרים כניסה משתמשים, גיאומטריה חישובית וכו'.
4. **רכיב אלגוריתמי חישובי -**

ישנה חשיבות רבה להתנסות של התלמיד והتمודדותו עם יכולות חישוביות במסגרת מכלולי הפרויקט. ניתן לשלב היבטים אלגוריתמיים או לחילופין ניתוחים וחיתוכים סטטיסטיים בסיסיים מעל מאגר נתונים, למשל :

 - 4.1. במקרה של אתרים כניסה משתמשים, גיאומטריה חישובית וכו'.

5. בדיקות תוכנה:

5.1. יש לגוזר מדרישות המוצר אוסף בדיקות שיופרדו לפחות לשתי קטגוריות מרכזיות ויכסו

את מרבית הקוד:

5.1.1. בדיקות יחידה (Unit-Test) - אלה הן אותן בדיקות אותן ימשח המפתח ברמת פנים מכלולי התוכנה ועד לרמת הפקנציות הציבוריות באותו מכלולי תוכנה.

5.1.2. בדיקות תהליכי (Full Flow) - הללו עוסקו בעיקר בבדיקה בקשר שבין מכלולי תוכנה מרכזיים ויבחנו את הפקנציונליות האינטגרטיבית של המוצר, מקצתה לפחות.

5.1.3. גישות מבורכות לתחילה ניתן לאמץ מתוך גישות שונות

למשל:

code a little test a little .5.1.4.1

, test driven development, .5.1.4.2

Regression Tests .5.1.4.3

פרקים מרכזיים נוספים בספר הפרויקט:

16. דרישות מערכת ופונקציונליות

16.1. הנחות יסוד

16.1.1. הנחות יסוד הקשורות לסביבה הטכנולוגית ולתפקיד.

16.2. דרישות מערכת

סביבת הטעמה ושימוש. שרידות, ביצועים והתמודדות עם עומסים.

16.3. דרישות פונקציונליות.

רשימת דרישות המשמש מהמערכת, מהן הפעולות בהן נדרש המערכת לתמוך.

2. חלופות ארכיטקטוניות-

דיון בבחירה ארכיטקטורתם ברמת המערכת וברמת מכלולי התוכנה.

5.2.1. ברמת המערכת- חשוב להציג בספר הפרויקט בחינה של מספר חלופות

ארქיטקטוניות (פחות 3) בהן יבחן התלמיד את האפשרויות השונות תחת מספר אילוצים רלוונטיים נושא הפרויקט, בין יתר השיקולים ניתן לשוקל: התמודדות עם עומסים וויסותם, שרידות, יכולת שימוש, זמינות טכנולוגית ועוד. יש לשוקל שילוב במקומות המתאימים של design patterns מקובלים, הן ברמת המערכת והן ברמת מכלולי התוכנה.

5.2.2. ברמת מכלולי התוכנה- ניתן להציג דיוון חלופות מצומצם יותר, אך לשוקל בחיבור שילוב של design patterns מקובלים במקומות המתאימים (להימנע משימוש מיותר).

6. טופולוגיה הפתרון הנבחר- הצגה סכימטית של פרישת המערכת.

7. ארכיטקטורה נבחרת: הצגה בגישה UML את פריטות מרכבי הפתרון בחלוקת למכלולי תוכנה ראשיים ומשניים כמו-גס הדיאגרמות הרלוונטיות.

7.1. שימוש במבני נתונים וארגון קבצים

7.1.1. נא פרט את מבני הנתונים.

7.1.2. נא פרט את שיטת האיחסון (מאגר, קבצים ובאיוזה טכנולוגיה)

7.1.3. נא ציין מנגנוני התאוששות מנפילוחקריסיה ותמייה בטראנסקיוט.

7.2. תרשימים זרימת המידע במערכת

Use Cases .7.2.1

Sequence diagram .7.2.2

Data flow .7.2.3

7.3. חלופות שפת מימוש -

במסגרת ספר הפרויקט חשוב להציג בבדיקה של מספר חלופות עבור שפהות מיומוש הפרויקט. הניל צריך לכלול דרישות אותן יגדיר התלמיד בבחירה השפה המתאימה. בין יתר השיקולים ניתן לכלול: זמני ריצה, היבטי אבטחה והגנה, הגנה על זכויות יוצרים (ビニアリ או interpreter, קלות בימוש, התאמה למשקי משתמש או צד שרת) וכו'.

8. חלופות אבטחתיות והגנה –

במסגרת ספר הפרויקט יש לבחון לפחות 3 חלופות אבטחתיות להגנה ושמירה על נתונים, יש לכלול התמודדות עם מקרים ותוגבות בתהילך הבדיקה (פחות 10 מקרים).

.8.1.1. יש לשים לב- לפרטוקולי תקשורת, ssl###, http ,https ,ssl### .

.8.1.2. מכלולי תוכנה צד שלישי (אנטי וירוס, מצפינים, firewalls וכו').

.8.1.3. ברמת הקוד- ווידוא סכימות הודעות בין מכלולי התוכנה ובתקשורת ביניהם.

2.8. פירוט בדיקות תוכנה ואופן ביצוע (STP-DOC) - לכלול את רשיימת בדיקות התוכנה, בדיקות ייחודית, בדיקות תהליכי Flow full מסמך תכנון בדיקות ובדיקות. הניל יוצג בטבלה : תיאור הבדיקה, תוצאה רצiosa, תוצאה מתקבלת.

ניתן לאמץ את נוהל מפתח באופן מושכל!

http://www.methodacloud.com/content/pages/kit_maxsum/H_Guide-map.asp