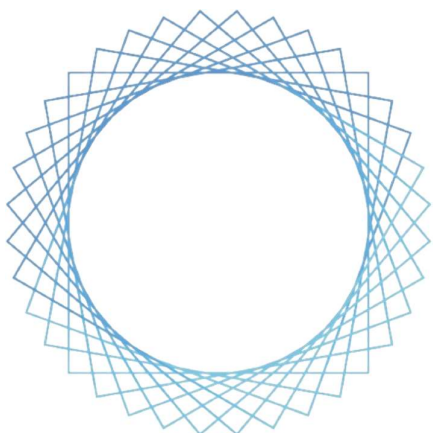


המרכז האקדמי רופין
בית-הספר להנדסה
המחלקה להנדסת חשמל ומחשבים
התוכנית להנדסת מחשבים



EZReport

פרויקט גמר הנדסי

מוגש כמילוי חלקי של הדרישות לקבלת תואר ראשון בהנדסה

נושא הפרויקט:

EZReport

תכנון בנייה ופיתוח של מערכת CRM

מבצע: רון מושאילוב

המנחים: מר תמיר דרשר (מכללת רופין), מר בן מישעלי (FiberNet)

השנה: תשפ"ה

המרכז האקדמי רופין

בית-הספר להנדסה המחלקה להנדסת חשמל ומחשבים התוכנית להנדסת מחשבים

פרויקט גמר הנדסי

מוגש כמילוי חלקי של הדרישות לקבלת תואר ראשון בהנדסה

נושא הפרויקט:

EZReport

תכנון בנייה ופיתוח של מערכת CRM

מבצע: רון מושאילוב

המנחים: מר תמיר דרשר (מכללת רופין), מר בן מישעלי (FiberNet)

השנה: תשפ"ה

הצהרה: העבודה המתוארת במסמך זה היא תוצאה של מחקר אישי שלי. כל טקסט או תוצאה שנלקחו והוכנסו לעבודה זו ממקורות אחרים מתועדים ככאלה. אני יודעת/שאי עמידה בתנאים (של עבודה עצמית, וציטוט נאות של מקורות) היא עבירה על תקנון המשמעת של בית הספר העשויה לגרור צעדים משמעתיים בפני ועדת המשמעת.

.....תאריך

.....תאריך

.....חתימת הסטודנט

.....חתימת המנחה

תקציר

חברת FiberNet מתמודדת עם אתגרים משמעותיים בניהול ובתיעוד תהליכי העבודה בשלוש תחנות עבודה מרכזיות: מחסן, ייצור ואריזה. היעדר תיעוד מספק גורם לחוסר שליטה על תהליך הייצור, אובדן חלקים, וחוסר יעילות כלכלית.

במחסן, אין תיעוד על זמן העבודה של העובדים, כמות החלקים שנאספה, או פרטי השליחה לתחנה הבאה (שם העובד, זמן השליחה). מצב זה מקשה על הערכת תפוקתו של העובד ויוצר חוסר וודאות בניהול המלאי.

במחלקת הייצור, אין מעקב אחר העובדים המטפלים בחלקים, ואובדן מעברים בין המחסן לייצור גורם לעיכובים ואובדן חלקים. כמו כן, אין תיעוד של השליחה לתחנה הבאה. במחלקת האריזה, המצב דומה: אין נתונים על קצב העבודה או על המוצרים שנשלחים ללקוחות.

הפרויקט מציע פתרון כוללני באמצעות פיתוח מערכת CRM חדשנית, שתספק:

1. **מעקב ותיעוד בזמן אמת** – רישום כל שלב במעבר חלקים בין תחנות, כולל נתוני העובד, תאריך ושעת הפעולה.
2. **חישוב זמני עבודה מדויקים** – מעקב אחר ביצועי העובדים, כדי לשפר את ניהול המשאבים והתפוקות.
3. **הפחתת אובדן ושיפור שליטה** – יצירת תמונת מצב כוללת למניעת טעויות תפעוליות וחסכון בעלויות.

ממשק המערכת תוכנן להיות אינטראקטיבי ופשוט לניהול, תוך התאמה לצרכים שונים של משתמשי הקצה. המערכת מאפשרת לעובדים בשטח להתנהל באופן עצמאי, כולל עולים חדשים מברית המועצות, באמצעות אפשרות מובנית להחלפת השפה לרוסית. התאמה זו תומכת בשילוב קל של עובדים חדשים, מגבירה את יעילותם ומצמצמת את התלות בהנחיה חיצונית.

המערכת צפויה לייעל את תהליך העבודה, לספק שקיפות ולמנוע הפסדים כספיים הנגרמים מאובדן חלקים או אי-יעילות תפעולית. היא תספק כלי ניהול חזק ואפקטיבי לשיפור קבלת ההחלטות בתחנות השונות ותתרום לשביעות רצון הלקוחות.

חילקתי את הפרויקט לשני חלקים מרכזיים:

1. שלב המחקר ובחינת תשתיות להקמת המערכת

בחנתי מספר חלופות לתשתיות הפיתוח של המערכת, הן בצד השרת והן בצד הלקוח. במסגרת המחקר, נבחנו טכנולוגיות מובילות בשוק תוך התחשבות בשיקולים טכניים, הפופולריות שלהן, והעדפות החברה.

לאחר התייעצות עם המנחים וסקול כלל הפרמטרים, נבחרו Node.js עבור צד השרת ו React-עבור צד הלקוח.

Node.js היא פלטפורמת JavaScript פופולרית, המספקת יעילות גבוהה בזכות המנוע המהיר שלה (V8), תמיכה מובנית באירועים אסינכרוניים, וקהילת מפתחים ענפה. React, שפותחה על ידי Facebook היא ספריית JavaScript מובילה לפיתוח ממשקי משתמש, המתאפיינת במהירות, פשטות וגמישות בזכות שימוש Components-.

הבחירה בטכנולוגיות אלה נועדה להבטיח ביצועים גבוהים, גמישות בהרחבת המערכת, ותמיכה רחבה מצד הקהילה המקצועית.

2. שלב המימוש והקמת שרת הנתונים

בשלב זה נבחנו האפשרויות להקים שרת עצמאי לניהול הנתונים של המערכת.

בהתאם להחלטות הפיתוח, שרת המערכת נבנה באמצעות Node.js המציעה פתרון יעיל ואינטואיטיבי לניהול API ושירותי backend. ספריות וכלים מובנים כמו Express.js שולבו לטובת מימוש קל ומהיר של שרת הנתונים, תוך שמירה על אבטחת מידע וסקלביליות.

בצד הלקוח, הממשק נבנה בטכנולוגיית React תוך שימוש ב-React Hooks לניהול מצב, וספריות מודרניות נוספות. הדגש בפיתוח היה על יצירת ממשק משתמש אינטראקטיבי ונגיש, שיתאים למגוון סוגי משתמשים, כולל עולים חדשים, עם אפשרות להחלפת שפה לרוסית.

תוצאת הפרויקט

התוצאה הסופית כללה אבטיפוס ראשוני של מערכת CRM המשלבת ממשק ניהול מבוסס React עם שרת נתונים מבוסס Node.js. המערכת תוכננה להיות מודולרית וניתנת להרחבה, כך שניתן להוסיף בקלות תכונות נוספות (כגון דוחות מותאמים אישית או מעקב סטטיסטי מתקדם). הממשק מותאם למשתמשים בעלי רקע טכנולוגי מגוון, ומתאפיין בפשטות וידידותיות למשתמש.

המערכת מספקת פתרון יעיל לניהול ותיעוד תהליכי העבודה בחברה, משפרת את ביצועי העובדים, ומייעלת את השליטה על התפוקה והעלויות.



תוכן עניינים

6	1 - הקדמה
6	2 - סקר ספרות
6	React – 2.1
7	Node.js – 2.2
7	Visual Studio Code - 2.3
8	MongoDB – 2.4
9	Postman – 2.5
10	מודל Client-Server - 2.6
11	מבנה פרויקט בסביבת הפיתוח Code Studio Visual - 2.7
11	2.7.1 – צד שרת
13	2.7.2 – צד לקוח
14	3 – תכנון הנדסי
14	3.1 – ארכיטקטורת המערכת (MERN)
15	Node.js – 3.2
16	Visual Studio Code - 3.3
18	MongoDB – 3.4
18	3.4.1 – Login to workspace
19	3.4.2 – Start working session
20	3.4.3 – Process of closing all kinds of reporting
21	3.4.4 – Send/Receive report process
22	3.4.5 – Update Event Process Sequence Diagram
23	3.4.6 – Delete Event Process Sequence Diagram
24	3.4.7 – View Event Process Sequence Diagram
25	3.4.8 – View My Events Process Sequence
26	3.4.9 – View All Events Process Sequence Diagram
27	3.6 – שיקולי התכנון
27	3.6.1 – צד שרת
29	3.6.2 – צד לקוח
31	MongoDB – 3.6.3
33	3.7 – ממשד הנתונים
33	3.7.1 – שיקולי המימוש
33	3.7.2 – תכנון ממשד הנתונים
34	3.7.3 – הסבר על דיאגרמה
35	3.8 – תקשורת ה-Backend מול ה-MongoDB
35	3.8.1 – תהליך ההתחברות לשרת ה-MongoDB
35	4 – ממשק המשתמש הסופי
35	4.1 – דף ההתחברות
35	4.2 – דף מרכז השליטה
35	4.3 – דף ההגדרות
35	4.4 – דף עמדת המחסן
35	4.5 – דף עמדת הייצור
35	4.6 – דף עמדת האריזה
35	4.7 – דף דף השגיאה
35	4.8 – דף דף לא קיים
35	4.9 – דף המנהל
39	5 – דיון ומסקנות
39	6 – אתגרים שהתקבלו בהם במהלך הפרויקט
39	7 – טכנולוגיות וכלים חדשים שהכרנו במהלך הפרויקט
39	8 – שיקולי תכנון ועיצוב הקוד
39	9 – המלצות לפרויקט להמשך
39	10 – ביבליוגרפיה

1 – הקדמה

עם התפתחות הטכנולוגיה והעלייה בדרישה ליעילות תפעולית, חברות רבות מתמודדות עם אתגרים מורכבים בניהול ותיעוד תהליכי הייצור והלוגיסטיקה שלהן. חברת FiberNet היא דוגמה בולטת לארגון הניצב בפני קשיים בניהול תהליכים בין תחנות העבודה שלה, דבר המשפיע ישירות על התפוקה, הדיוק והיעילות הכלכלית.

כיום, מעבר החלקים בין תחנות העבודה (מחסן, ייצור ואריזה) נעשה בצורה לא מתועדת, מה שמוביל לאובדן רכיבים, קושי במעקב אחר פרטי העובדים ותהליכי עבודה שאינם מדידים. בעיות אלו מונעות מהחברה לממש את מלוא הפוטנציאל שלה, ומדגישות את הצורך במערכת ניהול חדשנית המותאמת לצרכים המודרניים.

בנוסף, החברה משתמשת במערכת Priority לניהול כולל של הארגון, אך המערכת הנוכחית אינה מספקת פתרון מספק לניהול יעיל של התהליכים הפנימיים בתחנות הייצור. כדי לתת מענה לבעיה זו, נדרש פתרון שמאפשר ממשק עם Priority יחד עם מערכת מודרנית לניהול ותיעוד תהליכי העבודה.

מטלה זו דורשת פיתוח מערכת ייחודית שתספק פתרונות מעקב, ניהול ותיעוד פשוטים ואינטואיטיביים, תוך שיפור התקשורת והיעילות בין תחנות העבודה. המערכת שתפותח נועדה להיות גמישה, אינטראקטיבית, ונגישה לכל העובדים, כולל עולים חדשים, עם תמיכה בשפה הרוסית. הפרויקט נועד להוות צעד משמעותי בהפיכת סביבת העבודה ליעילה ומתקדמת יותר, עם דגש על הפחתת עלויות ושיפור התפוקה הכוללת של החברה.

2 - סקר ספרות

React – 2.1

React היא ספריית JavaScript פופולרית לפיתוח ממשקי משתמש, שפותחה על ידי Facebook בשנת 2013. הספרייה מתמקדת ביצירת ממשקים דינמיים באמצעות גישה מבוססת קומפוננטות. מבנה זה מאפשר פיתוח מודולרי ושימוש חוזר בקוד, מה שמפשט את התחזוקה וההרחבה של יישומים. React מתבססת על Virtual DOM, שמבטיח ביצועים גבוהים על ידי עדכון רק החלקים הנדרשים בממשק המשתמש במקום לבצע רינדור מלא של הדף. אחד המאפיינים הבולטים של React הוא התמיכה בחד-כיווניות של זרימת הנתונים (Unidirectional Data Flow) המסייעת בהבנה ובניבוי התנהגות היישום. הטכנולוגיה זכתה לפופולריות בזכות הקהילה הרחבה שלה, התמיכה הרבה במדריכים ובכלים, והתאמתה למגוון יישומים – החל מאפליקציות קטנות ועד מערכות מורכבות. מעצם היותה פתוחה, אנדרואיד מאפשרת ע"י אמצעים פשוטים לפתח אפליקציות ושירותים באופן פשוט. כל שנדרש היא סביבת פיתוח (IDE), מחשב בסיסי, ידע בסיסי ב-Java וזמן פנוי.

Node.js – 2.2

Node.js היא פלטפורמת JavaScript רבת-עוצמה לצד השרת, שפותחה על ידי Ryan Dahl בשנת 2009. היא מתבססת על מנוע V8 של Google, המספק ביצועים גבוהים ויכולת לטפל במספר רב של בקשות בו-זמנית. (Concurrency)

אחד מיתרונותיה המרכזיים של Node.js הוא היכולת לעבוד בצורה אסינכרונית באמצעות לולאת אירועים, (Event Loop) המפחיתה את זמן התגובה ומשפרת את ניצול המשאבים.

Node.js תומכת במגוון ספריות וכלים דרך, NPM (Node Package Manager) מה שמקל על פיתוח מהיר ויעיל של יישומים.

השימוש ב-JavaScript-הן בצד הלקוח והן בצד השרת מאפשר למפתחים לעבוד באותה שפה על כלל חלקי היישום, מה שמקדם אינטגרציה ושיתוף פעולה בצוותי פיתוח.

Node.js נפוצה במיוחד בפיתוח מערכות זמן-אמת, שירותי API מבוזרים, ויישומי רשת בקנה מידה גדול בזכות סקלריותה הגבוהה.

Visual Studio Code – 2.3

Visual Studio Code (VS Code) הוא עורך קוד פתוח חוצה-פלטפורמות שפותח על ידי Microsoft והושק בשנת 2015. מאז השקתו, הפך לכלי פיתוח פופולרי בזכות קלות השימוש, הממשק האינטואיטיבי והתאמתו למגוון רחב של שפות וטכנולוגיות פיתוח.

VS Code מבוסס על Electron מסגרת המאפשרת יישומי שולחן עבודה המשתמשים בטכנולוגיות אינטרנט כמו CSS, HTML ו-JavaScript. הכלי מציע עורך קוד קל משקל אך עוצמתי, המשלב אינטגרציות רבות עם כלים וסביבות פיתוח.

אחד היתרונות המרכזיים של VS Code הוא חנות התוספים (Extensions Marketplace) המאפשרת למפתחים להתקין הרחבות המוסיפות פונקציונליות, כגון תמיכה בשפות תכנות נוספות, כלים לניפוי באגים (Debugging) אינטגרציה עם מערכות בקרת גרסאות כמו Git ותמיכה במסגרת עבודה ספציפיות כמו Node.js, React או Python.

VS Code מציע תכונות מתקדמות כמו IntelliSense המספקת השלמה חכמה של קוד והצעות מבוססות הקשר, ניפוי באגים מובנה, וסביבת מסוף משולבת (Integrated Terminal) המאפשרת ביצוע פקודות ישירות מתוך העורך.

העורך מתאים למפתחים ברמות שונות בזכות התאמתו האישית והגמישות הרבה שלו. יכולת התמיכה בפרויקטים מורכבים, המשלבים טכנולוגיות שונות, הופכת אותו לכלי אידיאלי לצוותי פיתוח מודרניים.

VS Code זוכה לשדרוגים ועדכונים תכופים, שמבטיחים התאמה לצרכים המשתנים של קהילת המפתחים העולמית. השימוש בו נפוץ במיוחד בקרב מפתחי אינטרנט, אך הוא משמש גם בתחומים אחרים, כולל פיתוח תוכנה מותאמת אישית, ניתוח נתונים, ולמידת מכונה.

בזכות השילוב של ביצועים, פשטות וגמישות Visual Studio Code הפך לאחד מכלי הפיתוח המובילים בשוק הפיתוח העכשווי.

MongoDB – 2.4

MongoDB היא מסד נתונים מבוסס מסמכים (Document-Oriented Database), שפותח על ידי MongoDB Inc בשנת 2009. מסד נתונים זה מתבסס על מבנה JSON דינמי, שמאפשר גמישות רבה באחסון נתונים לעומת מסדי נתונים רלציוניים מסורתיים.

תכונות מרכזיות של MongoDB:

מבנה נתונים גמיש:

MongoDB אינו דורש סכמות קשיחות (Schema-less) מה שמאפשר שמירה של נתונים במבנים שונים תחת אותו אוסף (Collection) גמישות זו הופכת אותו לאידיאלי ליישומים דינמיים שבהם מבנה הנתונים עשוי להשתנות.

סקלרציות (Scalability):

MongoDB תומך בחלוקה אופקית (Sharding) שמאפשרת פיזור הנתונים על פני מספר שרתים, מה שמבטיח ביצועים גבוהים ויכולת לטפל בכמויות גדולות של נתונים.

ממשק JSON דינמי:

הנתונים נשמרים בפורמט BSON (Binary JSON) שמספק יעילות באחסון ושמירה של מבנים מורכבים. ניתן לשלוח ולעדכן נתונים בקלות באמצעות שאילתות מבוססות JSON.

אינדקסים מתקדמים:

MongoDB תומך ביצירת אינדקסים על שדות ספציפיים, דבר המשפר את מהירות הביצועים בשאילתות מורכבות.

תמיכה רחבה באינטגרציות:

MongoDB משתלב בקלות עם מסגרות פיתוח פופולריות כמו Node.js, Python, React ועוד. בנוסף, קיימת תמיכה ב-Mongoose-ספריית (Object Relational Mapping) ORM המפשטת את האינטראקציה עם מסד הנתונים.

שימושים נפוצים:

MongoDB נפוץ במיוחד ביישומים מודרניים כמו מערכות ניהול תוכן, אפליקציות מבוססות זמן אמת, ומסחר אלקטרוני, בזכות הביצועים הגבוהים וגמישות מבנה הנתונים.

Postman – 2.5

Postman הוא כלי רב-עוצמה לבדיקת ממשקי API שפותח בשנת 2012 והפך במהרה לאחד הכלים הנפוצים בעולם הפיתוח. הכלי מספק ממשק פשוט ונוח לבדיקת קריאות HTTP ומיועד לשימוש על ידי מפתחים ובודקי תוכנה.

תכונות מרכזיות של Postman:

בדיקות API אינטראקטיביות:

Postman מאפשר ביצוע קריאות HTTP (GET, POST, PUT, DELETE) וכו' בקלות, תוך הגדרת פרמטרים, כותרות, וגוף הבקשה (Body).

אוטומציה של בדיקות:

ניתן לכתוב תרחישי בדיקה אוטומטיים באמצעות JavaScript ולהריץ אותם על בקשות API כדי לוודא שהתשובות תואמות את הציפיות.

ממשק משתמש אינטואיטיבי

הממשק הידידותי מאפשר למפתחים ולבודקים לאמת בקלות את הביצועים והדיוק של ה-API, ללא צורך בידע מעמיק על שפות תכנות.

:Collections

Postman מאפשר לארגן בקשות API בקבוצות (Collections), דבר שמקל על ניהול הפרויקט ושיתוף פעולה בצוותי פיתוח.

: שילוב עם CI/CD

ניתן לשלב את Postman בתהליכי אינטגרציה והפצה רציפה (CI/CD) כדי לבדוק אוטומטית ממשקי API כחלק מתהליך הפיתוח.

: תיעוד API

Postman מספק כלי תיעוד מובנה, המאפשר למפתחים לשתף את המבנה והפונקציונליות של ה-API - עם צוותים אחרים או לקוחות.

: שימושים נפוצים

Postman משמש לבדיקת ממשקי API מבוזרים, בדיקות אבטחה, ביצועי API ותיעוד.

2.6 – מודל Client-Server

מודל שרת-לקוח (Client-Server) הוא פרדיגמה נפוצה במערכות מחשוב מבוזרות, שבה שרת מרכזי מספק שירותים או משאבים ללקוחות (Clients) המחוברים אליו. מודל זה משמש בבסיסם של יישומי אינטרנט מודרניים, כולל המערכת שפותחה במסגרת פרויקט זה. בפרויקט זה, המערכת מבוססת על ארכיטקטורה של צד שרת הבנוי ב-Node.js וצד לקוח המבוסס על React. צד השרת (Server) אחראי על עיבוד הנתונים, שמירתם ב-MongoDB והתממשקות עם מערכת Priority לניהול הנתונים של חברת FiberNet. צד הלקוח (Client) מספק ממשק משתמש אינטראקטיבי, שבו העובדים יכולים לתקשר עם המערכת בצורה פשוטה ונוחה. מודל השרת-לקוח מתאפיין בהפרדה ברורה בין שכבות המערכת:

1. צד השרת:

- מתפקד כמרכז עיבוד ומאגר מידע.
- מנהל בקשות המתקבלות מצד הלקוח, מעבד אותן, ומחזיר את התשובות המתאימות.
- כולל מנגנוני אבטחת מידע לניהול גישה לנתונים רגישים ותיאום מול Priority לצורך עדכון ושיתוף נתונים.

2. צד הלקוח:

- מספק ממשק אינטראקטיבי ואינטואיטיבי לעובדים, כולל אפשרות להחלפת שפה (כגון תמיכה ברוסית).
 - שולח בקשות לשרת באמצעות פרוטוקולים סטנדרטיים (HTTP/HTTPS) ומציג את התשובות בצורה ידידותית למשתמש.
 - משתמש ב-React לצורך ניהול ממשקי המשתמש בצורה מודולרית, עם דגש על ביצועים גבוהים והתאמה אישית.
- הפרדת המשימות בין השרת ללקוח מאפשרת פיתוח ותחזוקה פשוטים יותר של המערכת, לצד שיפור ביצועי המערכת כולה.
- יתרונות מודל זה כוללים:
- **סקלריות:** ניתן להרחיב את המערכת בקלות על ידי הוספת שרתים או התאמת היישום ליותר לקוחות.
 - **אבטחת מידע:** השרת יכול לשלוט בגישה לנתונים רגישים ולשמור אותם באופן מרכזי ובטוח.
 - **גמישות בפיתוח:** השימוש ב-React וב-Node.js מאפשר התמודדות עם שינויים או שדרוגים במערכת בצורה מהירה ויעילה.
- מודל שרת-לקוח שנבחר לפרויקט זה מתאים במיוחד לצרכי המערכת, שבה נדרש עיבוד כבד בצד השרת ותמיכה בממשק משתמש אינטראקטיבי בצד הלקוח. הבחירה בטכנולוגיות אלה מבטיחה ביצועים גבוהים ותחזוקה קלה, המותאמים לדרישות החברה.

2.7 – מבנה פרויקט בסביבת הפיתוח Visual Studio Code

Visual Studio Code (VS Code) הוא עורך קוד קל משקל ופופולרי, המספק גמישות רבה לניהול פרויקטים מורכבים בסביבות פיתוח שונות. מבנה פרויקט מאורגן ומובנה בסביבת VS Code תורם ליעילות העבודה, תחזוקת הקוד ושיפור שיתוף הפעולה בצוותי פיתוח.

מבנה הפרויקט משתנה בהתאם לטכנולוגיות ולדרישות היישום, אך ניתן להציג מבנה בסיסי עבור פרויקט המשלב Node.js בצד השרת React, בצד הלקוח, תוך שימוש ב-MongoDB-לניהול נתונים וב-Postman-לבדיקת ממשקי API:

2.7.1 – צד שרת

controller •

תיקייה זו מכילה קבצים לניהול לוגיקת הבקשות. כל קובץ מתייחס לחלק מסוים במערכת:

- **componentController.js** - מתמקד בניהול פעולות הקשורות לרכיבים.
- **employeeController.js** - מטפל בפעולות הקשורות לעובדים.
- **reportController.js** - מנהל לוגיקה הקשורה לדוחות.
- **workspaceController.js** - עוסק בפעולות שקשורות למרחב העבודה.

libs •

תיקייה זו נועדה לשמור לוגיקה עסקית משותפת (**Libraries**) הנמצאת בשימוש רחב במערכת:

- **componentLib.js** - מכיל פונקציות כלליות הקשורות לרכיבים.
- **employeeLib.js** - מכיל פונקציות עזר הקשורות לעובדים.
- **reportingPacking.js** - לוגיקה הקשורה לדיווחים על אריזה.
- **reportingProductionLib.js** - לוגיקה לדיווחים הקשורים לייצור.
- **reportingStorageLib.js** - פונקציות ניהול לדיווחי מחסן.
- **transferDetailsLib.js** - עוסק בפרטי ההעברות בין תחנות.
- **workspaceLib.js** - פונקציות כלליות למרחב העבודה.

model •

תיקייה זו מכילה את המודלים של הנתונים (**Models**) המבוססים על MongoDB:

- **Component.js** - מודל לניהול רכיבים.
- **Employee.js** - מודל לעובדים.
- **Enums.js** - כולל הגדרות של ערכים קבועים במערכת.
- **Report.js** - מודל לדוחות כלליים.
- **ReportingPacking.js** - מודל לדוחות אריזה.
- **ReportingProduction.js** - מודל לדוחות ייצור.
- **ReportingStorage.js** - מודל לדוחות מחסן.
- **TransferDetails.js** - מודל לפרטי העברות.
- **Workspace.js** - מודל לניהול מרחב עבודה.

• routes

תיקייה זו מרכזת את קובצי הניתוב (Routes) אשר מגדירים את נקודות הקצה של ה-API.

- **componentRoutes.js** - ניתוב לפעולות הקשורות לרכיבים.
- **employeeRoutes.js** - ניתוב לפעולות הקשורות לעובדים.
- **reportRoutes.js** - ניתוב לפעולות הקשורות לדוחות.
- **workspaceRoutes.js** - ניתוב לפעולות שקשורות למרחב העבודה.
- **connectToDB.js** - קובץ המחבר את המערכת למסד הנתונים.

• server.js

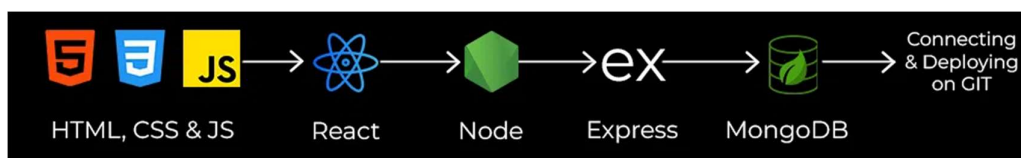
הקובץ הראשי של צד השרת. מכאן מתבצע הסטארט-אפ של השרת, טעינת ה-Controllers, הגדרת הניתובים והחיבור למסד הנתונים.

2.7.2 – צד לקוח

- **components**
 - o תיקייה זו מרכזת את הקומפוננטות החוזרות המשמשות בבנייה של ממשק המשתמש.
 - o **APIs** - קבצים המגדירים אינטראקציות עם צד השרת:
 - **components.js** – כולל קריאות API של הרכיבים.
 - **employee.js** - כולל קריאות API של העובדים.
 - **report.js** - כולל קריאות API של הדוחות.
 - **workspace.js** - כולל קריאות API של עמדת העבודה.
 - o **modals** - קומפוננטות של מודלים שמוצגים כחלונות קופצים:
 - **ManagerModal** – חלון קופץ עבור הפונקציונליות של דף המנהל.
 - **ComponentsModal** - חלון קופץ עבור רשימת הרכיבים של הדוח.
 - **CommentsModal** - חלון קופץ עבור ההערות של העמדה הקודמת.
 - **WorkSessionModal** - חלון קופץ עבור הפונקציונליות של דף ה-Dashboard.
 - o **Slidebar** - קומפוננטה להתאמת תפריט צדדי.
 - o **TableContainer** - קומפוננטה להצגת נתונים בטבלאות, עם הפרדה בין קוד וסגנונות.
 - o **Pages**: תיקייה זו מרכזת עמודים ראשיים באפליקציה.
 - **Dashboard**: עמוד מרכזי לתצוגת נתונים.
 - **errorPage**: עמוד לטיפול בשגיאות.
 - **notFoundPage**: עמודים לטיפול בעמודים חסרים.
 - **LoginPage**: עמוד כניסה למערכת.
 - **Manager**: עמוד ניהול.
 - **reportingStorage**: עמוד לדיווחים עבור המחסן.
 - **reportingProduction**: עמוד לדיווחים עבור הייצור.
 - **reportingPacking**: עמוד לדיווחים עבור האריזה.
 - **settings**: עמוד הגדרות.
 - **styles**
 - o תיקייה זו מרכזת סגנונות גלובליים:
 - o **images** - לאחסון תמונות.
 - o **_variables.scss** - משתנים משותפים עבור SCSS כמו צבעים, פונטים.
 - **utils**
 - o תיקייה המוקדשת לפונקציות עזר כלליות:
 - o **data.js**: מכיל מידע כללי אשר משומש ע"י מספר רב של גורמים.
 - o **functions.js**: מכיל פונקציות כלליות.
 - o **globalStates.js**: הגדרות סטייט משותפות.
 - **קבצים ראשיים**
 - o **App.js, App.test.js** - מבנה ראשי של האפליקציה ובדיקות.
 - o **index.js** - נקודת הכניסה של האפליקציה.

3 – תכנון הנדסי

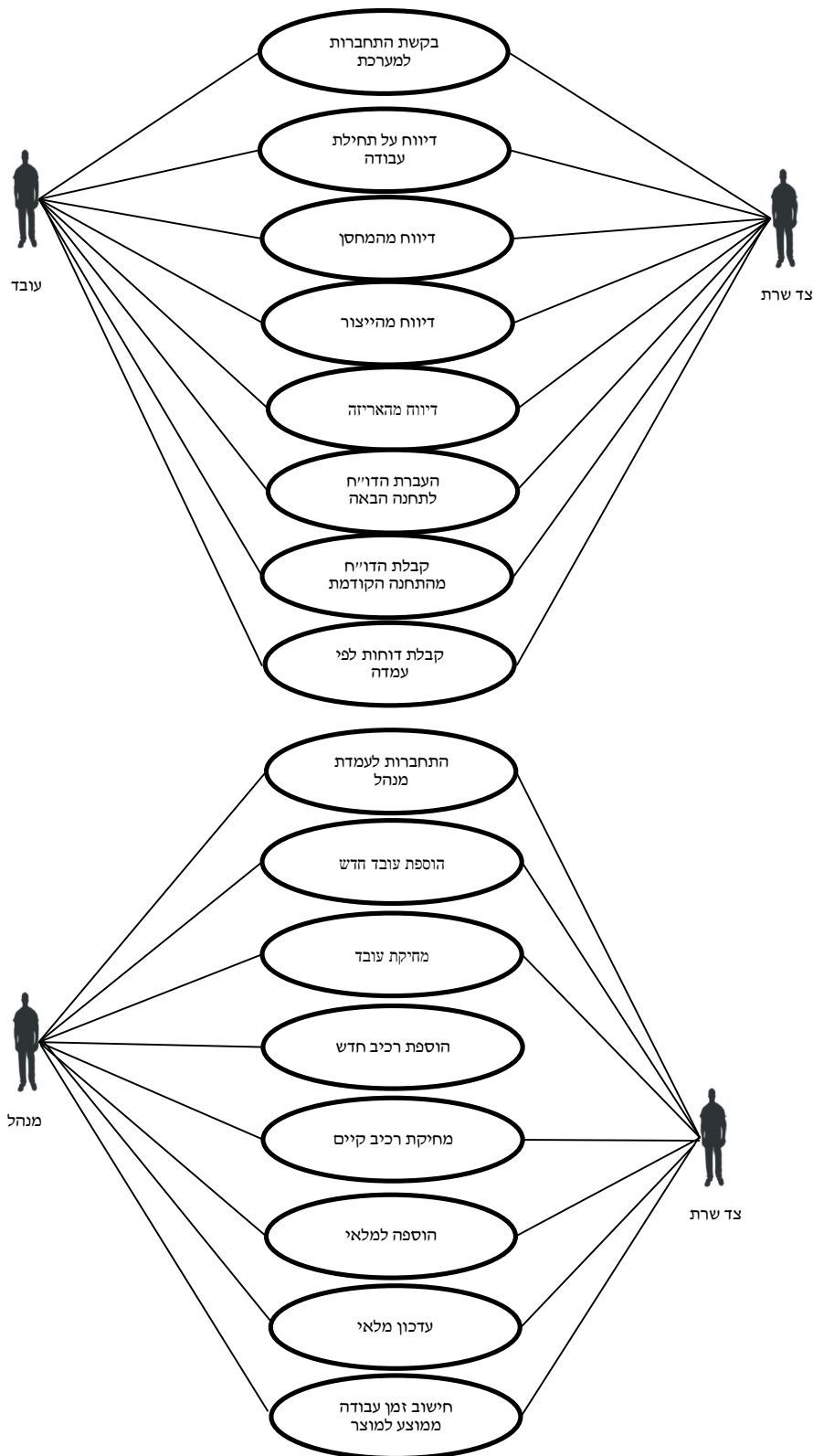
3.1 – ארכיטקטורת המערכת (MERN)



הארכיטקטורה MERN היא ערימת טכנולוגיות (Technology Stack) לפיתוח יישומים מבוססי אינטרנט. MERN הוא ראשי תיבות של React.js, Express.js, MongoDB ו-Node.js. ערימה זו מאפשרת פיתוח יישומים אינטראקטיביים וסקלאביליים בצד הלקוח ובצד השרת, תוך שימוש בשפת תכנות אחת בלבד JavaScript.

Use Case Diagram – 3.2

דיאגרמה התנהגותית אשר מנתחת תרחישי שימוש :



3.3 – טבלה המתארת את ה- Use Case Diagram

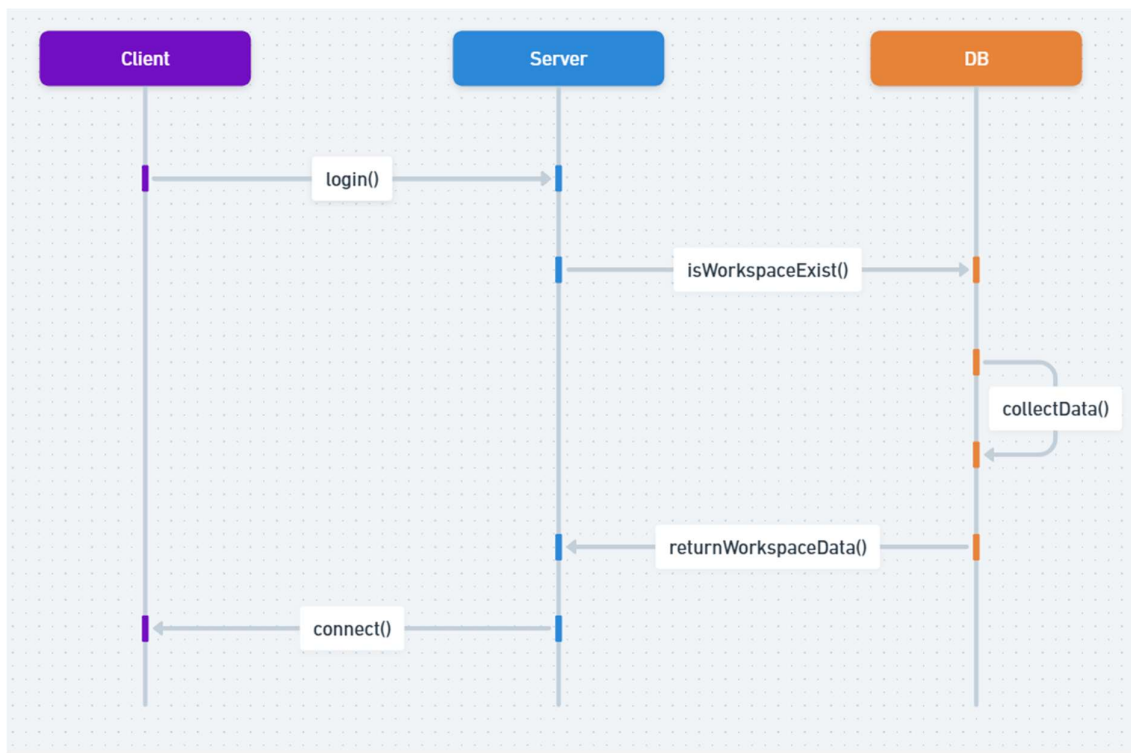
שירותים שהמערכת מספקת	תיאור
בקשת התחברות למערכת	בכל כניסה לאפליקציה, יידרש המשתמש להתחבר למערכת ע"י הקלדת מספר עמדה שעבורה הוא רוצה להתחבר.
דיווח על תחילת עבודה	משתמש אשר רוצה להתחיל לעבוד ידווח על תחילת עבודה כדי שבסיס הנתונים ידע מי העובד, מתי הוא מתחיל ובאיזה עמדה.
דיווח מהמחסן	משתמש אשר רוצה לסיים ולדווח על פעולתו ימלא את המלאי ברכיבים ואז ילחץ על "שליחה".
דיווח מהייצור	משתמש אשר רוצה לסיים ולדווח על כמות הרכיבים שיוצרו והערות במידת הצורך ואז ילחץ על "שליחה".
דיווח מהאריזה	משתמש אשר רוצה לסיים ולדווח על פעולתו כמות הרכיבים שנארוזו והערות במידת הצורך ואז ילחץ על "שליחה".
העברת הדו"ח לתחנה הבאה	משתמש אשר רוצה להעביר את הדו"ח לעמדת העבודה הבאה ישלח אותה והדו"ח ויעלם.
קבלת הדו"ח מהתחנה הקודמת	משתמש אשר רוצה לקבל דו"ח עמדת העבודה הנוכחית ילחץ על קבלה והדו"ח יועבר לתחנה הנוכחית ויהיה מוכן לעבודה.
קבלת דוחות לפי עמדה	כאשר המשתמש יתחבר לעמדה, לאחר שהעמוד יעלה האפליקציה תיגש באופן אוטומטי שוב לשרת כדי לבקש את הדוחות לפי התחנה.
התחברות לעמדת מנהל	משתמש אשר מתחבר לעמדת מנהל יזין סיסמה מיוחדת וכך יוכל להתחבר לעמדת המנהל.
הוספת עובד חדש	בהנחה והעובד בעמדת המנהל, המנהל ילחץ על הכפתור היעודי לכך בשם "הוסף עובד" ויזין את פרטי העובד.
מחיקת עובד	בהנחה והעובד בעמדת המנהל, המנהל ילחץ על הכפתור היעודי לכך בשם "הסר עובד" ויזין



את פרטי העובד.	
הוספת רכיב חדש	בהנחה והעובד בעמדת המנהל, המנהל ילחץ על הכפתור היעודי לכף בשם "הוסף רכיב" ויזין את פרטי הרכיב.
מחיקת רכיב קיים	בהנחה והעובד בעמדת המנהל, המנהל ילחץ על הכפתור היעודי לכף בשם "הסר רכיב" ויזין את פרטי הרכיב.
הוספה למלאי	בהנחה והעובד בעמדת המנהל, המנהל ילחץ על הכפתור היעודי לכף בשם "הוסף כמות" ויזין את פרטי הרכיב והכמות להוספה.
עדכון מלאי	בהנחה והעובד בעמדת המנהל, המנהל ילחץ על הכפתור היעודי לכף בשם "עדכן כמות" ויזין את פרטי הרכיב והכמות החדשה.
חישוב זמן עבודה ממוצע למוצר	בהנחה והעובד בעמדת המנהל, המנהל ילחץ על הכפתור היעודי לכף בשם "חשב ממוצע" ויזין את מספר הפק"ע וכך יקבל את כמות הזמן הממוצעת לייצור רכיב אחד.

Sequence Diagram – 3.4

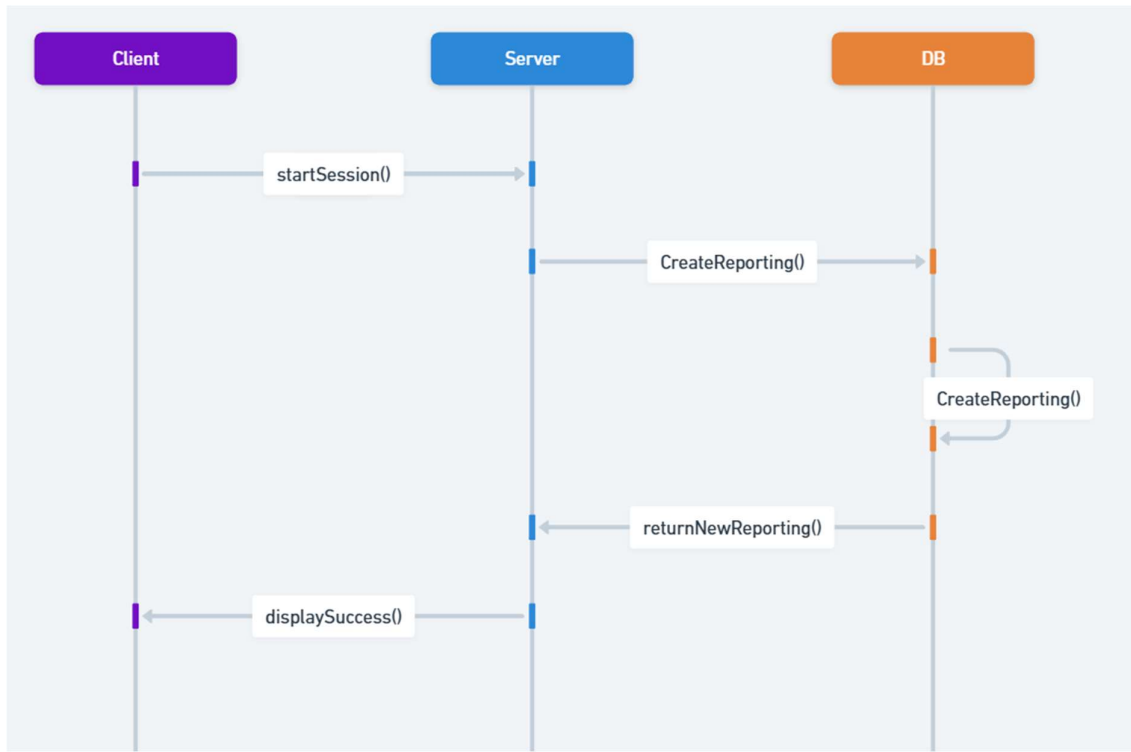
Login to workspace – 3.4.1



תהליך זה מתאר את הליך התחברות לעמדה.

- בשלב הראשון המשתמש יקליד את מספר העמדה
- בשלב השני, נשלחת בקשה לשרת לאימות העמדה וקבלת שם העמדה
- בשלב השלישי, המידע חוזר ואם הכל תקין יעלה הדף עם הפרטים הרלוונטים.

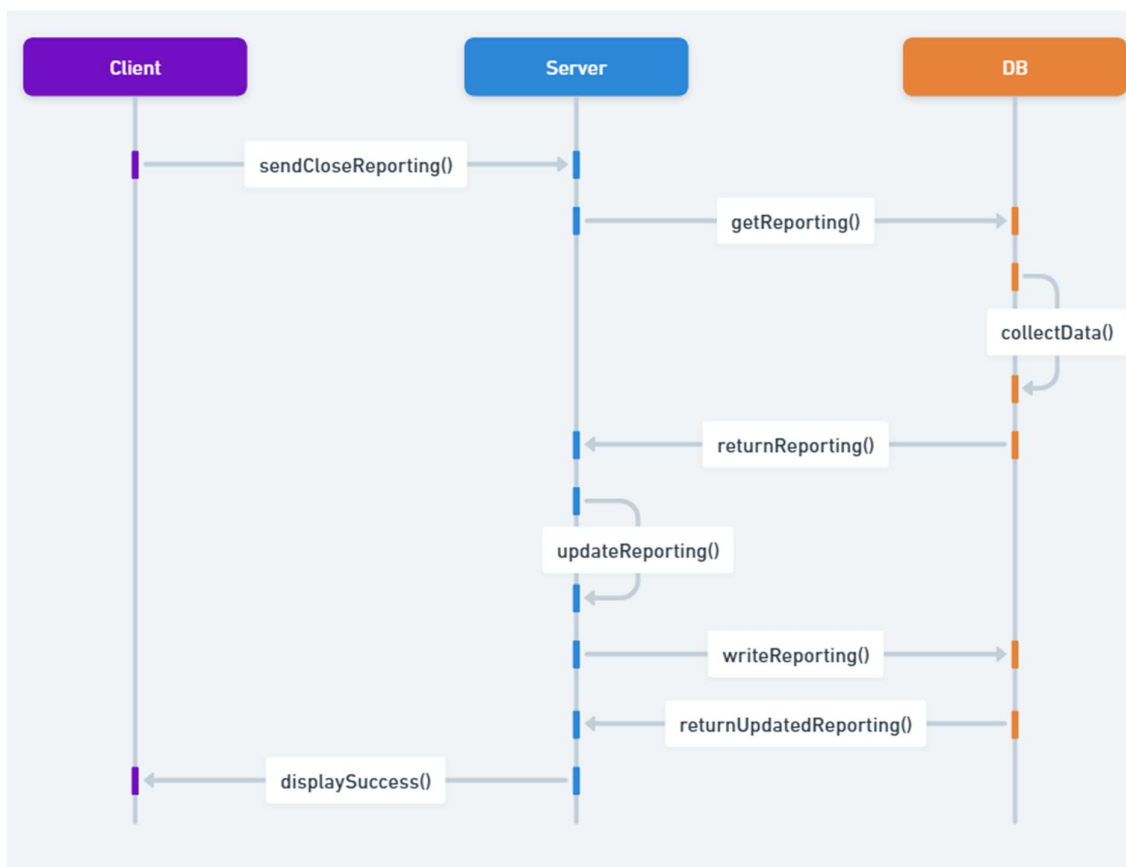
Start working session – 3.4.2



תהליך זה מתאר את הליך תחילת עבודה של העובד

- בשלב הראשון, המשתמש לוחץ על כפתור הדיווח
- בשלב השני, נפתח חלון המבקש להקליד מספר עובד שנעשה ע"י סורק ברקודים
- בשלב השלישי, תשלח בקשה לשרת לבדיקת המשתמש
- בשלב הרביעי, יוחזר מידע על תהליך הבדיקה ואם הכל עבר כשורה החלון יסגר והמשתמש יראה
toas שיגיד לו שהכל תקין אחרת החלון לא יסגר ומולו תופיע סיבת השגיאה

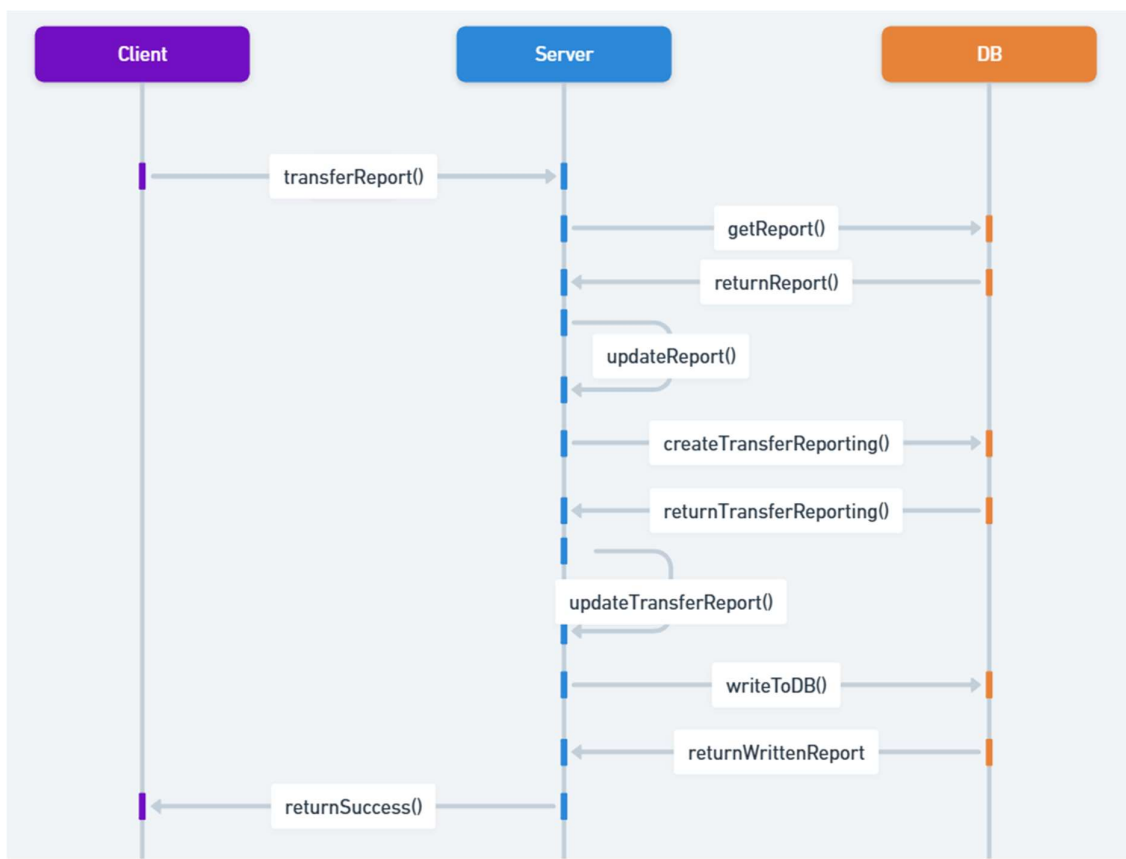
Process of closing all kinds of reporting – 3.4.3



תהליך זה מתאר את הליך סיום ודיווח של המחשב, ייצור ואריזה (לכולם הליך זהה)

- בשלב הראשון, נשלחת בקשה אל השרת עם כל הפרטים הרלוונטים
- בשלב השני, השרת מקבל את הפרטים ומחפש את הדו"ח הרלוונטי
- בשלב השלישי, השרת מתחיל לעדכן את הדיווח, לעדכן רכיבים ולהוריד אותם מהמלאי (עבור המחשב בלבד), מעדכן את הדו"ח משייך בין הדיווח לדו"ח ומחזיר תשובה למשתמש
- בשלב הרביעי, במידה והכל הצליח המשתמש יוחזר אוטומטית לדף הראשי וחלון toast יופיע מולו עם הודעה שהכל הצליח

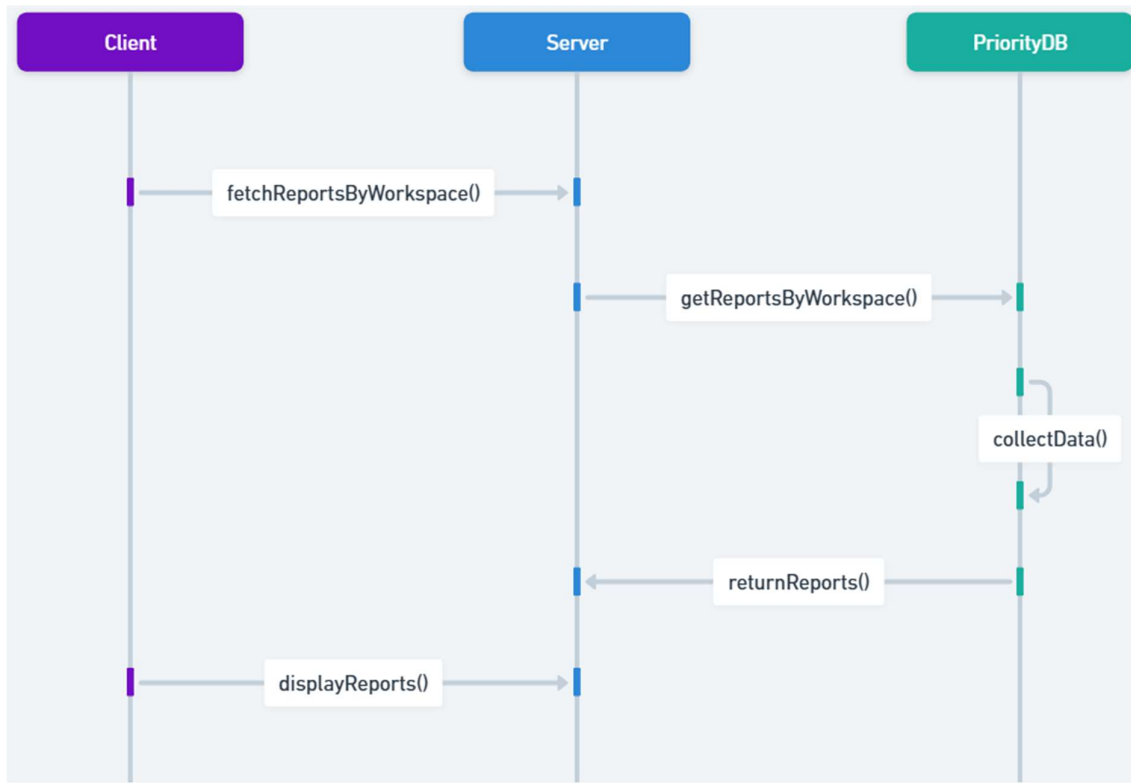
Send/Receive report process – 3.4.4



תהליך זה מתאר את הליך שליחה/קבלה

- בשלב הראשון, המשתמש ילחץ על כפתור החץ ויפיע מולו חלון שישאל אם הוא רוצה לשלוח
- בשלב השני, תישלח לשרת בקשה לשליחה/קבלה
- בשלב השלישי, השרת יזהה אם הדו"ח ממתין לקבלה ולכן הוא יפעל לקבל אותו ולפי כך יעדכן את הנתונים, אחרת הוא יבין שהדו"ח צריך להישלח ולכן הוא ישלח את הדו"ח לתחנה הבאה, יעדכן את הנתונים ויעבור את הדו"ח למצב "המתנה לקבלה".
- בשלב הרביעי, המשתמש יראה את הדו"ח נעלם עם הודעת toas שתגיד לו שהדו"ח נשלח/התקבל

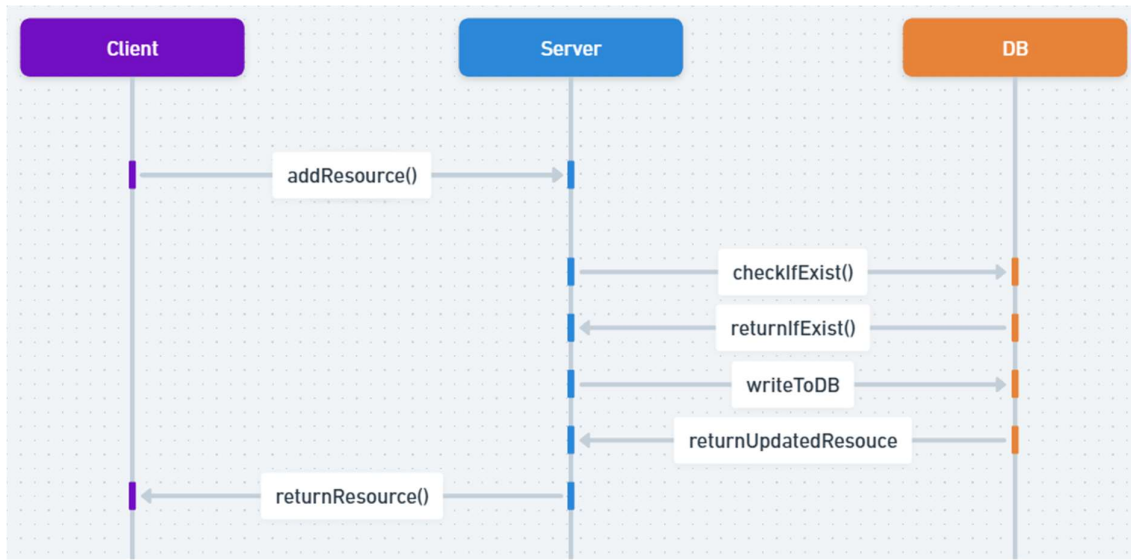
Update Event Process Sequence Diagram – 3.4.5



תהליך זה מתאר את הליך בקשת דוחות

- בשלב הראשון, האתר ישלח לבד בקשה לשרת לקבל את הדוחות
- בשלב השני, נמשכים הדוחות העדכניים לפי התחנה
- בשלב השלישי, יופיע מול המשתמש טבלה עם כל הדוחות והפרטים הרלוונטים

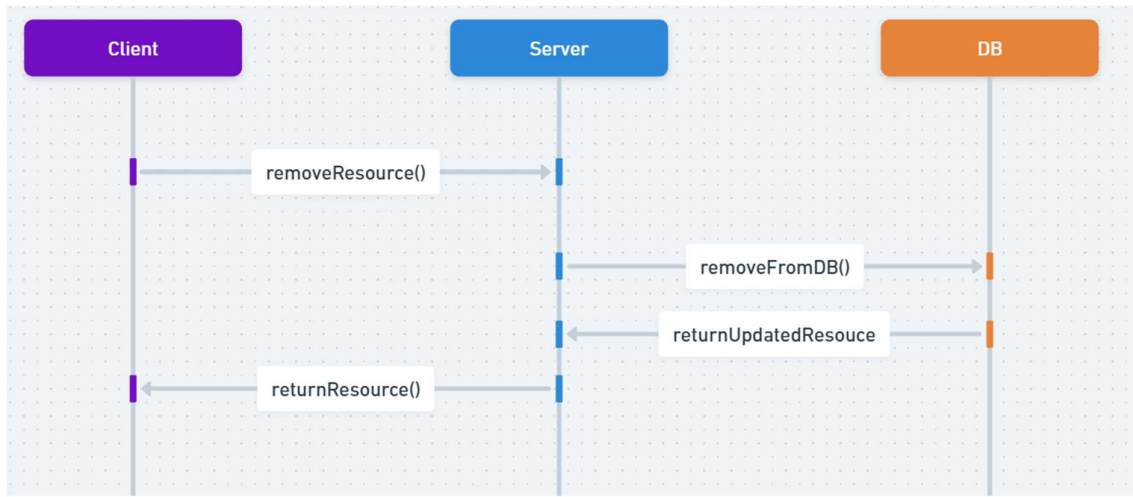
Delete Event Process Sequence Diagram – 3.4.6



תהליך זה מתאר הוספה של עובד ורכיב

- בשלב הראשון, המנהל יבחר את הפעולה הרצויה וימלא בחלון modal שיופיע מולו את הפרטים הרלוונטים וישלח את הבקשה
- בשלב השני, השרת יבדוק שכל הנתונים תקינים ואין מקרה של כפילות
- בשלב השלישי, במידה והכל תקין השרת יוסיף את הרכיב לבסיס הנתונים וישלח חזרה למשתמש הודעה לפי אם הפעולה הצליחה או לא

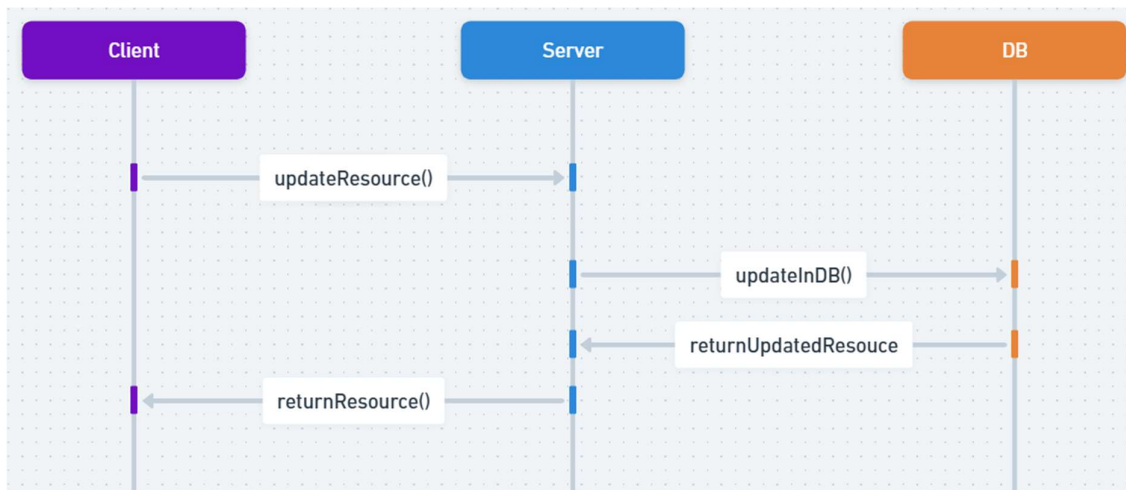
View Event Process Sequence Diagram – 3.4.7



תהליך זה מתאר מחיקה של עובד ורכיב

- בשלב הראשון, המנהל יבחר את הפעולה הרצויה וימלא בחלון modal שיופיע מולו את הפרטים הרלוונטים וישלח את הבקשה
- בשלב השני, השרת יבדוק שכל הנתונים תקינים
- בשלב השלישי, במידה והכל תקין השרת ימחק את הרכיב מבסיס הנתונים וישלח חזרה למשתמש הודעה לפי אם הפעולה הצליחה או לא

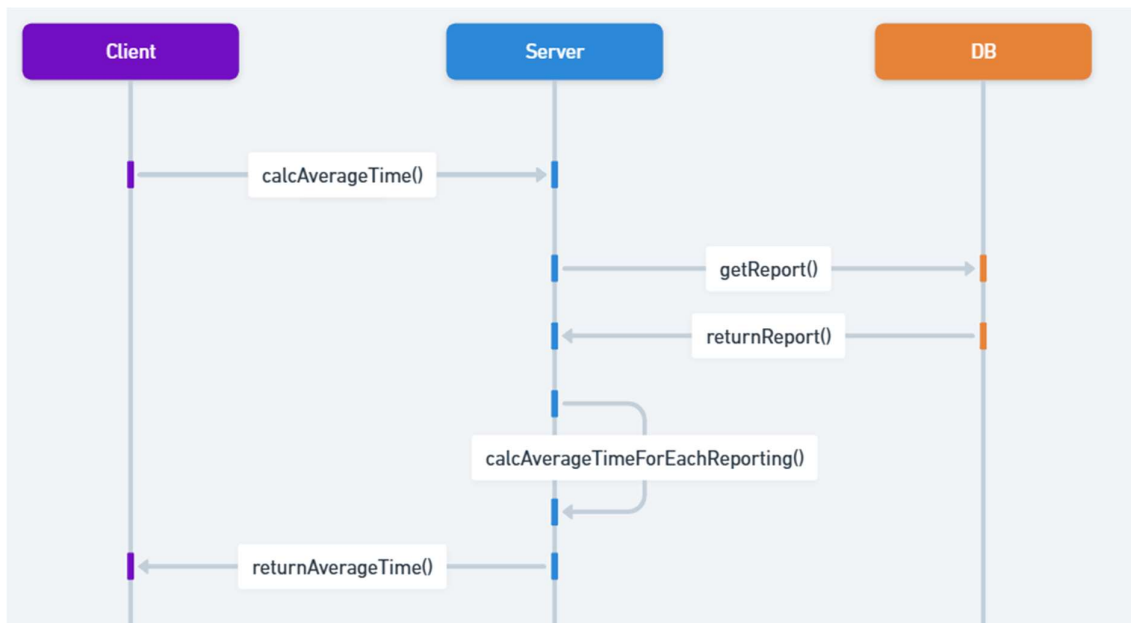
View My Events Process Sequence – 3.4.8



תהליך זה מתאר עדכון של רכיב

- בשלב הראשון, המנהל יבחר את הפעולה הרצויה וימלא בחלון modal שיופיע מולו את הפרטים הרלוונטים וישלח את הבקשה
- בשלב השני, השרת יבדוק שכל הנתונים תקינים ושהרכיב אכן קיים
- בשלב השלישי, במידה והכל תקין השרת יעדכן את הרכיב מלאי הרכיב בסיס הנתונים בהתאם לבקשה וישלח חזרה למשתמש הודעה לפי אם הפעולה הצליחה או לא

View All Events Process Sequence Diagram – 3.4.9



תהליך זה מתאר חישוב זמן ממוצע לייצור רכיב

- בשלב הראשון, יבחר את הפעולה הרצויה, יקליד מספר פק"ע וישלח בקשה לשרת
- בשלב השני, השרת יפנה לבסיס הנתונים כדי למשוך את כל הדיווחים של הייצור שקיימים עבור אותו הדו"ח
- בשלב השלישי, השרת יספור כמה זמן סך הכל לקח לכל דיווח עד שהסתיים וכמה רכיבים כל דו"ח ייצר
- בשלב הרביעי, מתבצע החישוב הממוצע והוא נשלח חזרה למשתמש
- בשלב החמישי, תופיע מול המשתמש הודעה עם כמות הזמן הממוצעת לרכיב

3.6 – שיקולי תכנון

3.6.1 – צד שרת

בפרויקט זה בחרנו לממש את צד השרת באמצעות פלטפורמת **Node.js** בחירה זו התבססה על מספר שיקולים מרכזיים:

1. **סקלABILיות (Scalability)**

- Node.js מספק תמיכה מעולה לעומסים גבוהים ולמספר רב של בקשות בו-זמנית, בזכות מנגנון ה-Event Loop האסינכרוני שלו.
- הפלטפורמה מתאימה במיוחד למערכות מבוזרות וליישומים עם תעבורת נתונים גבוהה.

2. **יעילות וביצועים**

- Node.js מתבסס על מנוע V8 של Google שמאפשר הרצת JavaScript במהירות גבוהה.
- המבנה האסינכרוני של הפלטפורמה תורם לצמצום זמני התגובה, במיוחד במערכות עם גישה תדירה למסדי נתונים או שירותים חיצוניים.

3. **שימוש בשפה אחידה**

- בחירה ב-Node.js מאפשרת עבודה ב-JavaScript הן בצד השרת והן בצד הלקוח (React).
- גישה זו מפשטת את תהליך הפיתוח, התחזוקה ושיתוף הפעולה בין חברי הצוות.

4. **תמיכה בהתממשקות**

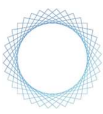
- Node.js תומך במגוון ספריות וכלים המאפשרים התממשקות קלה עם MongoDB באמצעות ספריית (Mongoose) ומערכות צד שלישי כמו-Priority ERP.
- הפלטפורמה מתאימה במיוחד לפיתוח RESTful APIs הדרושים לצורך תקשורת עם צד הלקוח.

5. **תחזוקה ופיתוח מהיר**

- ספריית (Node Package Manager) NPM מספקת גישה לאלפי ספריות מוכנות, מה שמאפשר פיתוח מהיר ויעיל.
- פלטפורמת Node.js נתמכת על ידי קהילה פעילה ורחבה, מה שמקל על מציאת פתרונות ותמיכה.

6. **אבטחה**

- Node.js תומכת במנגנונים מתקדמים לניהול גישה מאובטחת, כולל הצפנה, אימות משתמשים, וניהול תעבורת נתונים באמצעות פרוטוקולים מאובטחים (HTTPS).
- הספריות המובנות ותוספים כמו Helmet.js מסייעים בחיזוק אבטחת היישום.



7. ניהול עומסים וזמינות

- Node.js מתאימה ליישומים הנדרשים לפעול 24/7 עם תחזוקה מינימלית.
- מנגנוני Load Balancing ויכולת הטיפול במספר רב של בקשות מבטיחים זמינות גבוהה.

8. התאמה למסדי נתונים מבוזרים

- הבחירה ב-MongoDB כבסיס נתונים משתלבת היטב עם Node.js בזכות המבנה מבוסס JSON והיכולת לבצע שאילתות דינמיות בצורה יעילה.

9. עלויות

- Node.js היא פלטפורמה בקוד פתוח, מה שמפחית את עלויות הפיתוח.
- התחזוקה הקלה והביצועים הגבוהים מפחיתים את ההוצאות התפעוליות.

10. תמיכה בניטור ותיעוד

- ישנם כלים מתקדמים לניטור תפקוד השרת כגון PM2 ו-Loggly-המאפשרים מעקב אחר ביצועים, סטטיסטיקות ותיעוד שגיאות.

בסופו של דבר, הבחירה ב-Node.js-עבור צד השרת סיפקה לי פתרון יעיל, גמיש ומודרני, שתואם את דרישות המערכת ואת האתגרים הטכנולוגיים בפרויקט.

3.6.2 – צד לקוח

בפרויקט זה בחרתי לממש את צד הלקוח באמצעות React בחירה זו התבססה על מספר שיקולים מרכזיים:

1. פיתוח מבוסס קומפוננטות

- React מאפשרת בניית ממשק משתמש בצורה מודולרית על בסיס קומפוננטות (Components).
- מבנה זה מקל על פיתוח, תחזוקה ושימוש חוזר בקוד, במיוחד בפרויקטים גדולים ומורכבים.

2. מהירות וביצועים

- React מתבססת על Virtual DOM שמאפשר רינדור מהיר ויעיל על ידי עדכון רק החלקים שנדרשים.
- ביצועים אלו חשובים במיוחד ליישומים עם ממשקי משתמש דינמיים ותדירות גבוהה של עדכונים.

3. זרימת נתונים חד-כיוונית (Unidirectional Data Flow)

- React משתמשת במודל חד-כיווני לזרימת הנתונים, מה שמקל על מעקב אחר שינויים במצב היישום ומפחית באגים.

4. תאימות עם RESTful APIs

- React משתלבת בקלות עם APIs מבוססי REST שמפותחים בצד השרת (Node.js). ניתן לבצע בקשות נתונים (GET, POST) וכו' ולרנדור את הנתונים המתקבלים באופן דינמי בממשק המשתמש.

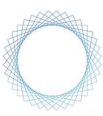
5. ניהול מצב (State Management)

- React מספקת כלים מובנים לניהול מצב קומפוננטות באמצעות **React Hooks**.
- עבור יישומים מורכבים, ניתן לשלב כלים מתקדמים לניהול מצב כמו Redux או Context API שמאפשרים שיתוף נתונים בין חלקים שונים של האפליקציה.

6. תמיכה בעיצוב רספונסיבי

- React מאפשרת פיתוח ממשקי משתמש מותאמים לכל סוגי המכשירים, מה שמבטיח חוויית משתמש אופטימלית על מחשבים, טאבלטים וסמארטפונים.
- שימוש בספריות עיצוב מתקדמות כמו Material-UI או Bootstrap מיעיל את הפיתוח.

7. תחזוקה ופיתוח מהיר



- React בנויה בצורה מודולרית, מה שמקל על עדכון קומפוננטות ושדרוג המערכת מבלי לפגוע בתפקוד הכללי.

- הספרייה נהנית מקהילה רחבה שמספקת תוספים, מדריכים ודוגמאות.

8. גמישות והתאמה אישית

- React מאפשרת שליטה מלאה במבנה ובפונקציונליות של ממשק המשתמש.
- ניתן לשלב ספריות חיצוניות לצורך ניהול נתונים, עיצוב גרפי או פונקציות מתקדמות נוספות.

9. תמיכה באפליקציות אינטראקטיביות

- React מתאימה במיוחד לפיתוח יישומים אינטראקטיביים, בהם המשתמשים נדרשים להזין נתונים, לשנות הגדרות או לנווט בין מסכים בצורה דינמית.

10. אינטגרציה עם כלים מתקדמים

- React משתלבת עם כלים כמו Webpack או Babel לניהול תהליכי בנייה, וספריות כמו Axios או Fetch לצורך קריאות API.
- בנוסף, ישנה תמיכה מלאה בתיעוד וסטטיסטיקות באמצעות ספריות כגון React Developer Tools.

11. תמיכה בשפות וקהלים שונים

- React מאפשרת בקלות להוסיף תמיכה בריבוי שפות כגון, i18n מה שמקל על יצירת חוויית משתמש מותאמת לקהלים מגוונים.

12. תמיכה בקוד פתוח ועדכונים שוטפים

- React היא ספרייה בקוד פתוח הנתמכת על ידי Facebook. עדכונים ושיפורים משוחררים באופן שוטף, עם תמיכה נרחבת מהקהילה.

בסופו של דבר, הבחירה ב React-עבור צד הלקוח סיפקה לנו פתרון מודרני, גמיש ויעיל, שמאפשר למשתמשים אינטראקציה חלקה עם המערכת תוך שמירה על ביצועים גבוהים וחוויית משתמש מצוינת.

MongoDB – 3.6.3

בפרויקט זה בחרנו להשתמש ב **MongoDB**-כמסד הנתונים של המערכת. הבחירה התבססה על מספר שיקולים מרכזיים:

1. מבנה נתונים גמיש (Schema-less)

- MongoDB אינו דורש מבנה נתונים קשיח (Schema), מה שמאפשר גמישות רבה בשמירה וניהול של נתונים משתנים.
- גישה זו מתאימה במיוחד לפרויקטים דינמיים שבהם מבנה הנתונים עשוי להשתנות לאורך זמן.

2. פורמט (JSON (BSON

- MongoDB משתמש ב-BSON (Binary JSON) פורמט קל משקל ואינטואיטיבי לאחסון נתונים.
- קלות השימוש בפורמט זה מאפשרת אינטגרציה חלקה עם JavaScript וטכנולוגיות כמו Node.js ו-React.

3. סקלריות גבוהה (Scalability)

- MongoDB תומכת בחלוקה אופקית (Sharding) המאפשרת פיזור הנתונים על פני מספר שרתים.
- תכונה זו מבטיחה יכולת לטפל בכמויות נתונים גדולות תוך שמירה על ביצועים גבוהים.

4. תמיכה בשאילתות דינמיות

- MongoDB מאפשרת ביצוע שאילתות גמישות ומתקדמות, כגון חיפושים על פי שדות מרובים, סינון, ומיון.
- השאילתות מתבצעות בצורה יעילה ומאפשרות שליפת נתונים במהירות גבוהה.

5. תמיכה במערכות מבוזרות

- MongoDB תוכנן לפעול בצורה מיטבית במערכות מבוזרות, מה שמתאים במיוחד לארכיטקטורות מודרניות כמו MERN.
- תכונה זו מבטיחה יתירות גבוהה ואמינות במקרה של תקלות או איבוד נתונים.

6. אינטגרציה עם Node.js

- MongoDB משתלב בקלות עם Node.js באמצעות ספריות כמו Mongoose שמספקות כלים לניהול הסכמות ומודלים.
- החיבור הישיר עם השרת מאפשר זרימת נתונים חלקה ותגובה מהירה לקריאות API.

7. פשטות התקנה וניהול

- MongoDB הוא מסד נתונים קל להתקנה ולתחזוקה, עם ממשק משתמש נוח לניהול מסדי הנתונים.
- תכונות מובנות כמו גיבוי, שחזור וניהול אינדקסים מקלות על תחזוקת הנתונים.

8. תמיכה במבני נתונים מורכבים

- MongoDB מאפשר אחסון נתונים מורכבים ומקוננים (Nested Documents) בתוך מסמך יחיד, דבר המפחית את הצורך בחיבורים (Joins) ומייעל את הביצועים.

9. ניהול ביצועים ואינדקסים

- MongoDB תומכת באינדקסים מתקדמים, שמאפשרים אופטימיזציה של שאלות ושיפור זמן התגובה של המערכת.
- ניתן ליצור אינדקסים מותאמים אישית עבור שדות מסוימים או קבוצות שדות.

10. אבטחת מידע

- MongoDB מציע תכונות מובנות של אבטחת מידע, כולל הצפנת נתונים, בקרת גישה, ותמיכה בתעבורת HTTPS.
- ניתן ליישם שכבות נוספות של אבטחה כדי לשמור על מידע רגיש במערכת.

11. קהילה רחבה ותמיכה

- MongoDB נתמך על ידי קהילה רחבה וכלים מתקדמים לניטור, תיעוד ואופטימיזציה של מסדי נתונים.
- קל למצוא מדריכים, דוגמאות ותוספים שמתאימים ליישומים מגוונים.

12. התאמה ליישומים מודרניים

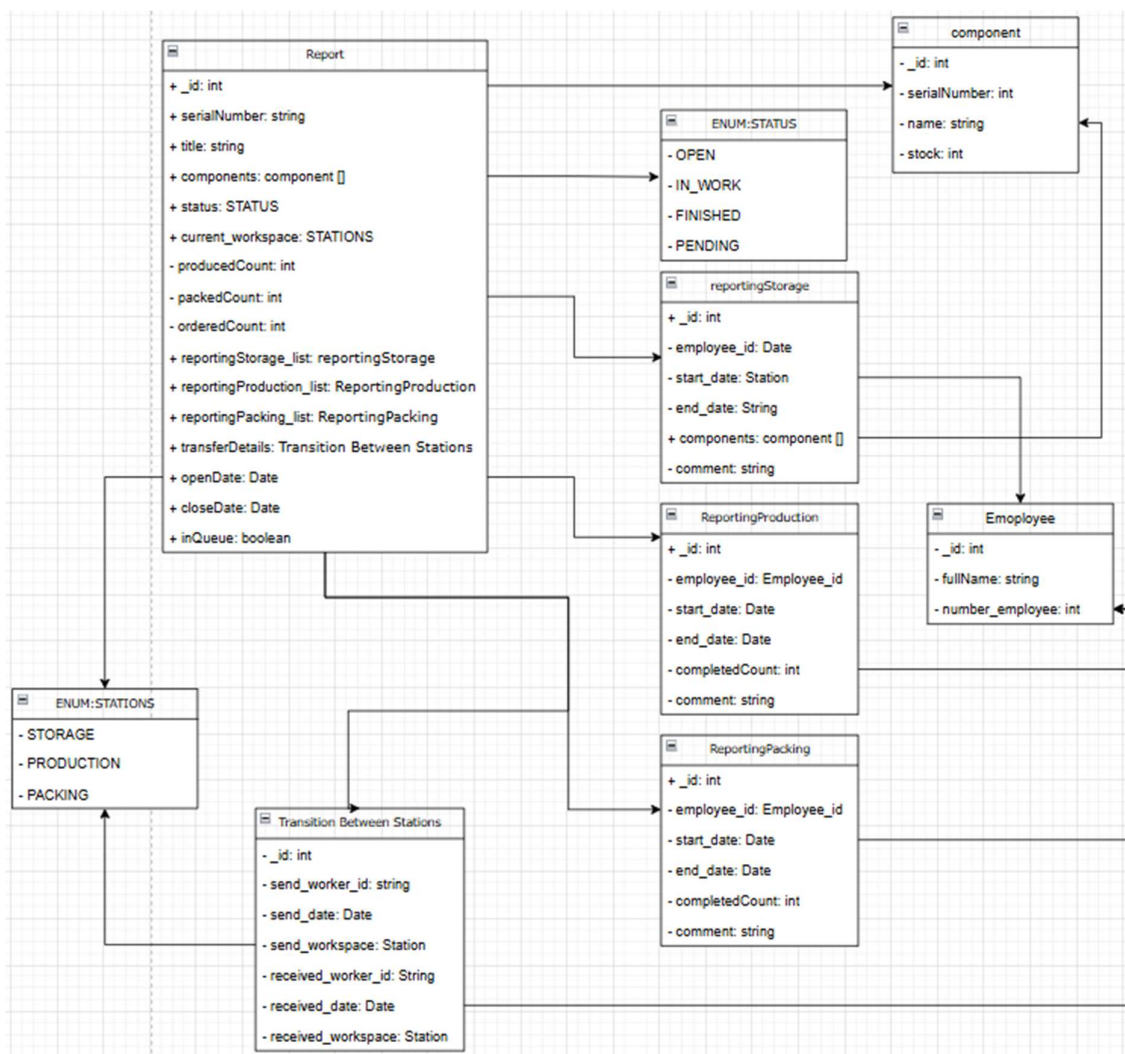
- MongoDB מתאים במיוחד ליישומים מבוססי זמן אמת, מערכות ניהול תוכן, אפליקציות אינטרנט, ומערכות מבוזרות.

3.7 – ממסד הנתונים

3.7.1 – שיקולי השימוש

MongoDB הוא פתרון אידיאלי לפרויקט שלי מכיוון שהוא משתלב בצורה מושלמת עם Node.js השרת שבחרתי ומאפשר עבודה עם נתונים במבנה JSON המתאים בדיוק לצרכי המערכת שלי. מסד הנתונים הזה מציע גמישות מרבית בניהול נתונים, מה שמקל על תיעוד תהליכי העבודה במחסן, ייצור ואריזה, גם כאשר הדרישות עשויות להשתנות לאורך זמן MongoDB. תומך בשאילתות מתקדמות, כך שניתן לשלוף בקלות נתונים על עובדים, פרטי מעברים בין תחנות וזמני עבודה לצורך ניתוח ובקרה. הפשטות בניהול והשילוב עם ספריית Mongoose מאפשרים לי להגדיר מודלים עבור כל תחנה בצורה יעילה, והסקלאביליות הגבוהה תבטיח שהמערכת תתמודד עם גידול בעומסים בעתיד.

3.7.2 – תכנון ממסד הנתונים



3.7.3 – הסבר על הדיאגרמה

Report – Schema אשר מייצגת את הצורה שבה ישמרו הדוחות בבסיס הנתונים.

STATIONS - ENUM שבו נשמרים התחנות האפשריות בבסיס הנתונים .

TransferDetails - Schema שמתארת את פרטי ההעברה.

STATUS - ENUM שבו נשמרים המצבים האפשריים לדו"ח להיות בהם.

reportingStorage – דיווח אשר מכיל רק את פרטי הדיווח הספציפי לעובד ספציפי במחסן, בדו"ח ישמר רק ה-id שלו.

reportingProduction – דיווח אשר מכיל רק את פרטי הדיווח הספציפי לעובד ספציפי בייצור, בדו"ח ישמר רק ה-id שלו.

reportingPacked – דיווח אשר מכיל רק את פרטי הדיווח הספציפי לעובד ספציפי באריזה, בדו"ח ישמר רק ה-id שלו.

Component – Schema אשר מייצגת את הצורה שבה ישמרו הרכיבים בבסיס הנתונים.

Employee – Schema אשר מייצגת את הצורה שבה ישמרו העובדים בבסיס הנתונים.

3.8 – תקשורת ה Backend מול ה-MongoDB

ההתחברות לבסיס הנתונים מתבצעת באמצעות ספריית **Mongoose** המשמשת כ-ORM (Object Relational Mapping) לניהול אינטראקציה עם MongoDB. Mongoose מאפשרת חיבור יעיל ופשוט באמצעות כתובת ההתחברות (Connection String) ומספקת כלים לניהול סכמות, ולביצוע שאילתות ותפעול נתונים בצורה מסודרת ומאובטחת.

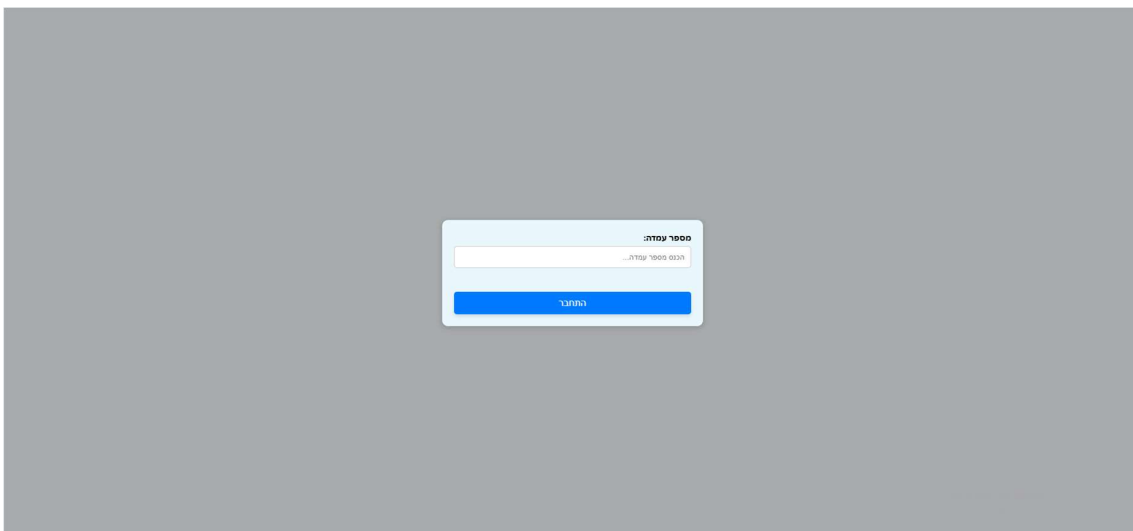
3.8.1 – תהליך ההתחברות לשרת MongoDB

```

1  const mongoose = require('mongoose');
2
3  const connectToDB = async () => {
4    try{
5      await mongoose.connect(process.env.MONGO_URI);
6      console.log('MongoDB connected successfully' + '\n');
7    } catch(err){
8      console.error('Error connecting to MongoDB:', err.message);
9      process.exit(1); // Exit process with failure
10   }
11 }
12
13 module.exports = connectToDB;
14
```

4 – ממשק המשתמש הסופי

4.1 – דף ההתחברות




The screenshot shows a login form titled "ספרי עמוד:" (Book Page:). It contains a text input field with the placeholder text "הכנס סופרי עמוד..." (Enter book page...). Below the input field is a blue button labeled "התחבר" (Login).

4.2 – דף מרכז השליטה

חפש לפי מספר סידורי...

עמדת מחסן

שליחה	סטוס דרישה	החילול עבודה	תאריך פתיחה	החמנו	מארח	יצוא	חומה נוכחית	סטטוס	מספר סידורי	שם
			1/1/2024	100	0	0	Storage	IN_WORK	SN-1001	Tester
			1/1/2024	50	0	0	Storage	OPEN	SN-1004	All way
			1/3/2024	100	0	0	Storage	OPEN	SN-1006	Boisga
			1/1/2024	100	0	0	Storage	OPEN	SN-1010	Daily Production Report
			1/1/2024	100	0	0	Storage	OPEN	SN-1012	Daily Production Report
			1/1/2024	100	0	0	Storage	OPEN	SN-1013	Daily Production Report
			1/1/2024	100	0	0	Storage	OPEN	SN-1011	Daily Production Report
			1/1/2024	100	0	0	Storage	OPEN	SN-118	Bilby
			1/1/2024	100	0	0	Storage	OPEN	SN-116	Annie rules
			1/1/2024	100	0	0	Storage	OPEN	SN-117	Bob
			1/1/2024	100	0	0	Storage	OPEN	SN-114	Daily Pikachu
			1/1/2024	100	0	0	Storage	OPEN	SN-115	Two piece



פקעו"ת

חור

הגדרות

יציאה

Activate Windows
Go to Settings to activate Windows.

4.3 – דף ההגדרות

בחר שפה

☒ עברית
☐ English
☐ Русский

[חזור](#)



4.4 – דף עמדת המחסן

רשימת רכיבים

הפש רכיב לפי שם או מספר...

שם: גליל

רשימת רכיבים: 105

כמות: 5

שם: בקוק

רשימת רכיבים: 103

כמות: 10

מספר רכיב:

הכנס מספר רכיב...

כמות:

הערות:

הערה...

חזור

הצג רכיבים בהמשך

שלח דיווח

הוסף רכיב

כל הרכיבים במערכת

הפש לפי מספר רכיב...

שם: גליל

מספר רכיב: 101

כמות: 600

שם: בקוק

מספר רכיב: 103

כמות: 90

שם: קליף

מספר רכיב: 102

כמות: 220

4.5 – דף עמדת הייצור

דיווח חדש מספר 0007

כמות יחידות

הערות

כמות יחידות

הערות

מספר עובד

מקט

חוקים

הוסמך

100

SN-1003

0

100

חזור

הצג הערות

שלח

4.6 – דף עמדת האריזה

דיווח חדש מספר 0007

כמות יחידות

הערות

כמות יחידות

הערות

מספר עובד

מקט

נארוז

ייצור

100

SN-1002

0

0

חזור

הצג הערות

שלח

4.7 – דף השגיאה

שגיאה

אירעה שגיאה במערכת. אנא נסה להתחבר מחדש

התחבר מחדש

4.8 – דף לא קיים

דף לא קיים - 404

אופס! הדף לא קיים

חזרה לעמוד הראשי

4.9 – דף המנהל

פקטוריות לא בעבודה

מספר סידורי	כותרת	סטטוס	תחנה
SN-1003	Daily Packing Report	OPEN	Production
SN-1004	All way	OPEN	Storage
SN-1007	Ramzi	OPEN	Production
SN-1008	Jonny Daily tasks	OPEN	Production
SN-1006	Bosga	OPEN	Storage
SN-1010	Daily Production Report	OPEN	Storage
SN-1012	Daily Production Report	OPEN	Storage
SN-1009	Daily Production Report	OPEN	Production
SN-1013	Daily Production Report	OPEN	Storage
SN-1011	Daily Production Report	OPEN	Storage

פקטוריות בעבודה

מספר סידורי	כותרת	סטטוס	תחנה
SN-1002	Daily Production Report	IN_WORK	Packing
SN-1001	Tester	IN_WORK	Storage

עובדים

הוסף עובד

רשם עובד

רכיבים

הוסף רכיב

רשם רכיב

הוסף כמות

עדכן כמות

פקטוריות

הוסף פקטוריה

פקטוריות ממתנות

מספר סידורי	כותרת	סטטוס	תחנה
-------------	-------	-------	------

פקטוריות שהסתיימו

מספר סידורי	כותרת	סטטוס	תחנה
-------------	-------	-------	------

5 – דיון ומסקנות

מטרת הפרויקט הייתה לפתח מערכת CRM שתיתן מענה לניהול יעיל של תהליכי העבודה בחברת FiberNet. המערכת נועדה להתמודד עם אתגרים מרכזיים, כמו היעדר תיעוד של מעברי תחנות, חוסר מעקב אחר זמן העבודה של העובדים, ותפקוד מבולגן של מערכת הניהול הקיימת. באמצעות המערכת שפותחה, החברה תוכל לשפר את תהליך הייצור, להקטין הפסדים כלכליים ולשפר את השקיפות התפעולית.

תוצאות הפרויקט כוללות מערכת המורכבת משרת Backend המבוסס על Node.js מסד נתונים MongoDB, צד לקוח מבוסס React, וממשק התממשקות עם מערכת Priority. המערכת מספקת מעקב מלא על תהליכי העבודה, תיעוד מעברים בין תחנות, וחישוב זמן עבודה לעובדים בצורה שקופה ומדויקת.

עם זאת, במהלך הפיתוח נתקלנו באתגרים טכניים, כגון הבטחת סנכרון מלא בין המערכת שלנו ל-Priority, וכן פיתוח תמיכה מרובת שפות. חלק מהיעדים, כמו יצירת דוחות מתקדמים או אוטומציה מלאה של חלק מהתהליכים, לא הושגו בשל מגבלות זמן ומשאבים.

למרות האתגרים, המערכת שפותחה מספקת מענה מוצק לבעיות המרכזיות שהוגדרו בתחילת הפרויקט ומהווה תשתית חזקה להרחבות עתידיות. החברה תוכל כעת לשפר את תהליכי העבודה שלה באופן מדויק ומודרני, תוך התייעלות תפעולית ושיפור חוויית העובדים והלקוחות.

6 – אתגרים שנתקלנו בהם במהלך הפרויקט

1. למידת טכנולוגיות חדשות

חלק ניכר מהטכנולוגיות שבהן השתמשנו בפרויקט, כגון Node.js, React, MongoDB, והתממשקות עם מערכת, Priority, לא נלמדו במהלך התואר. היה עלינו להשקיע זמן רב בלמידה עצמית, הבנת העקרונות הבסיסיים, והתמודדות עם בעיות טכניות שנבעו מהיעדר ניסיון קודם.

2. שינויים ושיפורים חוזרים

במהלך מימוש המערכת, גילינו פעמים רבות דרכים יעילות ונכונות יותר לביצוע פעולות שכבר מומשו. מצב זה גרם לנו לבנות מחדש חלקים משמעותיים מהמערכת מספר פעמים, דבר שהשפיע על לוחות הזמנים ועל עומס העבודה.

3. חוסר רצינות מצד סמנכ"ל החברה

אחד האתגרים המשמעותיים בפרויקט היה חוסר שיתוף פעולה ורצינות מצד סמנכ"ל החברה. לעיתים קרובות, לא קיבלנו את המידע והתמיכה הנדרשים בזמן, דבר שהאט את תהליך הפיתוח ויצר תחושת תסכול.

למרות האתגרים הללו, הצלחנו להשלים את הפרויקט תוך עמידה ברוב היעדים שהצבנו לעצמנו. תהליך ההתמודדות עם האתגרים תרם רבות לצבירת ידע מעשי ולפיתוח מיומנויות חדשות שישמשו אותנו בעתיד.

7 – טכנולוגיות וכלים חדשים שהכרנו במהלך הפרויקט

- Node.js
- React
- MongoDB
- Postman

8 – שיקולי תכנון ועיצוב הקוד

• כיצד השתמשנו בכלים שקיבלנו בזמן הלימודים

לא נעשה שום שימוש בכלים שקיבלנו בזמן הלימודים. בתואר לא נחשפנו שטכנולוגיות האלו אבל לשמחתי היו לי חברים שעובדים בתחום זה בתעשייה, את המנחה שלי מצד רופין, את המנחה שלי מצד החברה, אינטרנט ו-chatGPT ששאלתי שנעזרתי בהם ובסופו של דבר ע"י שילוב של הדעות השונות של כולם למדתי והצלחתי לסיים את הפרויקט.

• המצב בתעשייה

פיתוח ב-Node.js ו-React פופולארי מאוד בתעשייה כיום, הבעיה העיקרית היא המשרות ללא ניסיון כי אחרי הפרויקט הזה החלטתי לקחת את הקרייה שלי לכיוון של Full-Stack אבל בגלל שאין לי ניסיון זה מאוד קשה.

9 – המלצות לפרויקט להמשך

• הוספת סוגי אירועים נוספים

לדעתי צריך לתת לאתר לעבוד בחברה במשך חודש ולקבל תגובות מצד העובדים ולבצע שדרוגים בהתאם, זה גם מה שתכננה חברת FiberNet מלכתחילה, אבל כבר עכשיו אני יכול לחשוב על כמה רעיונות.

• פיצול עמדת הייצור לעמדות קטנות יותר

בזמן שביקרנו בחברה נחשפנו למחלקות השונות ושמנו לב שמחלקת הייצור כוללת בתוכה ייצור של המון תחנות קטנות בתוכה שמייצרות כל אחד משהו שונה בצורה שונה (ריתוך, הלחמה...) ואפילו עמדות מיוחדות עבור חברות שרוצות לבצע פעולה באופן עצמאי ולכן פירוק לתחנות קטנות יותר יכול להיות יעיל ביום מן הימים.



10 – ביבליוגרפיה

- [1] [Node.js](#)
- [2] [React](#)
- [3] [MongoDB](#)
- [4] [Postman](#)
- [5] [ExpressJS](#)
- [6] [Mongoose](#)