המרכז האקדמי רופין

בית-הספר להנדסה

המחלקה להנדסת חשמל ומחשבים

התוכנית להנדסת מחשבים

תכנון הנדסי

מבצע: רון מושאילוב

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

השנה: תשפ״ה

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.j**s - 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.

צד לקוח

- **components**
תיקייה זו מרכזת את הקומפוננטות החוזרות המשמשות בבנייה של ממשק המשתמש.

  - **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.

  - **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.

  - **Slidebar**: קומפוננטה להתאמת תפריט צדדי.

  - **TableContainer**: קומפוננטה להצגת נתונים בטבלאות, עם הפרדה בין קוד וסגנונות.

  - **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**
תיקייה זו מרכזת סגנונות גלובליים:

  - **images**: לאחסון תמונות.
  - **_variables.scss**: משתנים משותפים עבור SCSS כמו צבעים, פונטים.

- **utils**
תיקייה המוקדשת לפונקציות עזר כלליות:

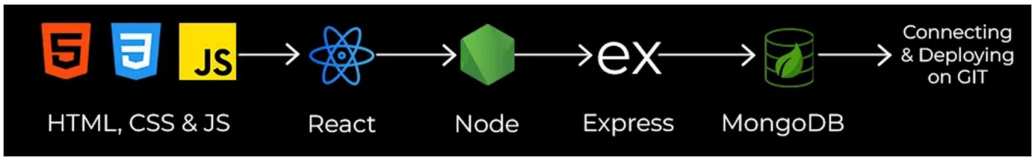
  - **data.js**: מכיל מידע כללי אשר משומש ע"י מספר רב של גורמים.
  - **functions.js**: מכיל פונקציות כלליות.
  - **globalStates.js**: הגדרות סטייט משותפות.

.

- **קבצים ראשיים**

  - **App.js, App.test.js**: מבנה ראשי של האפליקציה ובדיקות.
  - **index.js**: נקודת הכניסה של האפליקציה.

ארכיטקטורת המערכת (MERN)



MERN הוא Development Stack מבוסס JavaScript שמיועד לפיתוח יישומי Web דינמיים ומתקדמים, והוא כולל ארבע טכנולוגיות עיקריות MongoDB, Express.js, React.js ו-Node.js .

MongoDB - משמש כבסיס נתונים NoSQL לאחסון וניהול נתונים בצורה גמישה ודינמית בפורמט **JSON**.

Express.js - הוא Framework צד שרת צד שרת קל משקל המאפשר בניית API וניתוב בקשות בצורה פשוטה ויעילה.

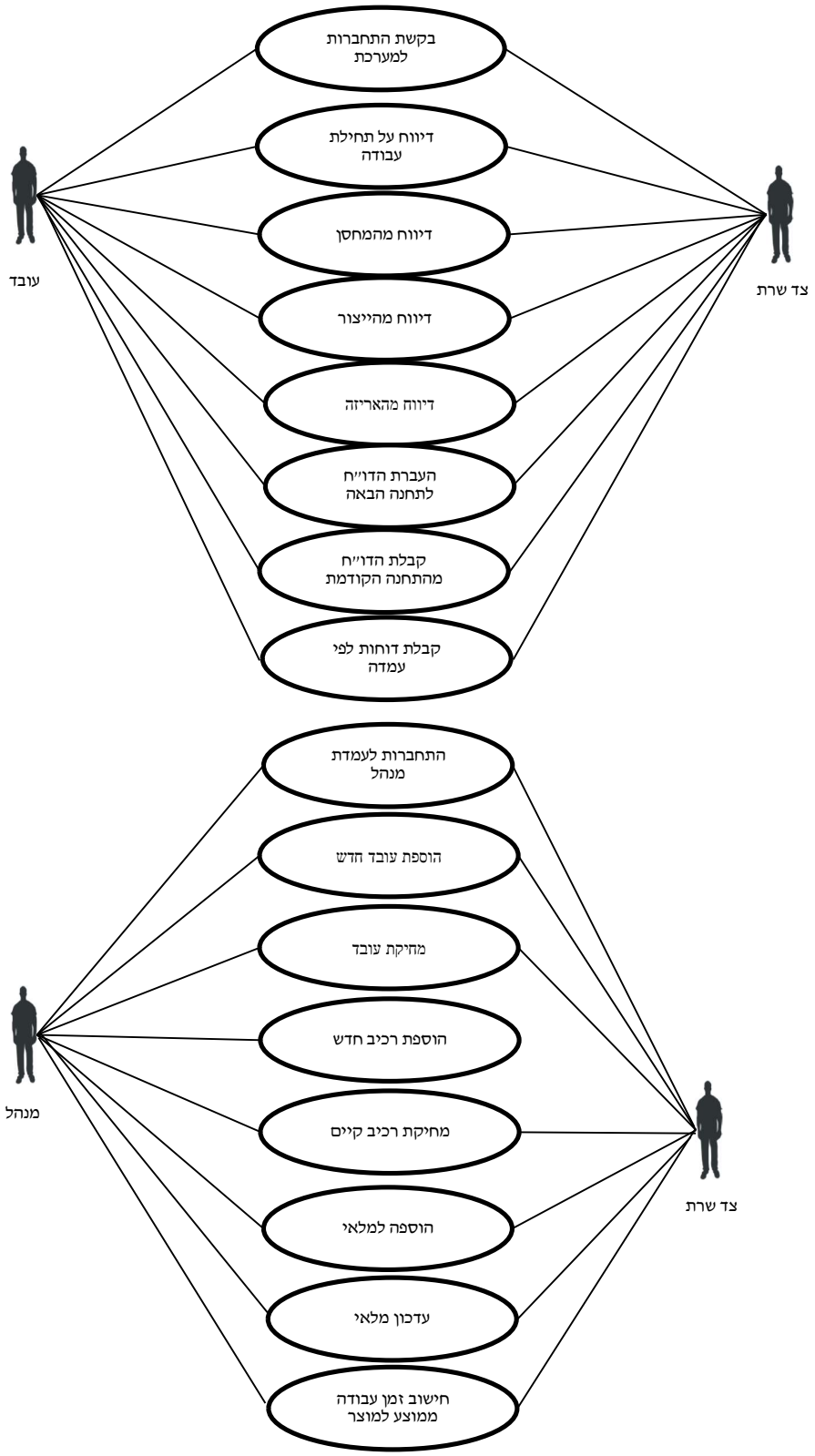
React.js - היא ספריית JavaScript מתקדמת לפיתוח ממשקי משתמש דינמיים ומודרניים.

Node.js - היא סביבה להרצת JavaScript בצד השרת המאפשרת טיפול בכמות גדולה של בקשות בו זמנית עם ביצועים גבוהים.

Stack MERN בנוי כך שכל שכבות הפיתוח מבוססות JavaScript מה שמקל על הפיתוח, האינטגרציה והתחזוקה. ה-Stack מתאים במיוחד לפיתוח יישומים דינמיים, אינטראקטיביים ויישומי Full-Stack מודרניים בזכות הגמישות והיעילות שלו, והוא פופולרי מאוד בזכות הפשטות שלו והקהילה הענפה התומכת בו.

Use Case Diagram

דיאגרמה התנהגותית אשר מנתחת תרחישי שימוש :



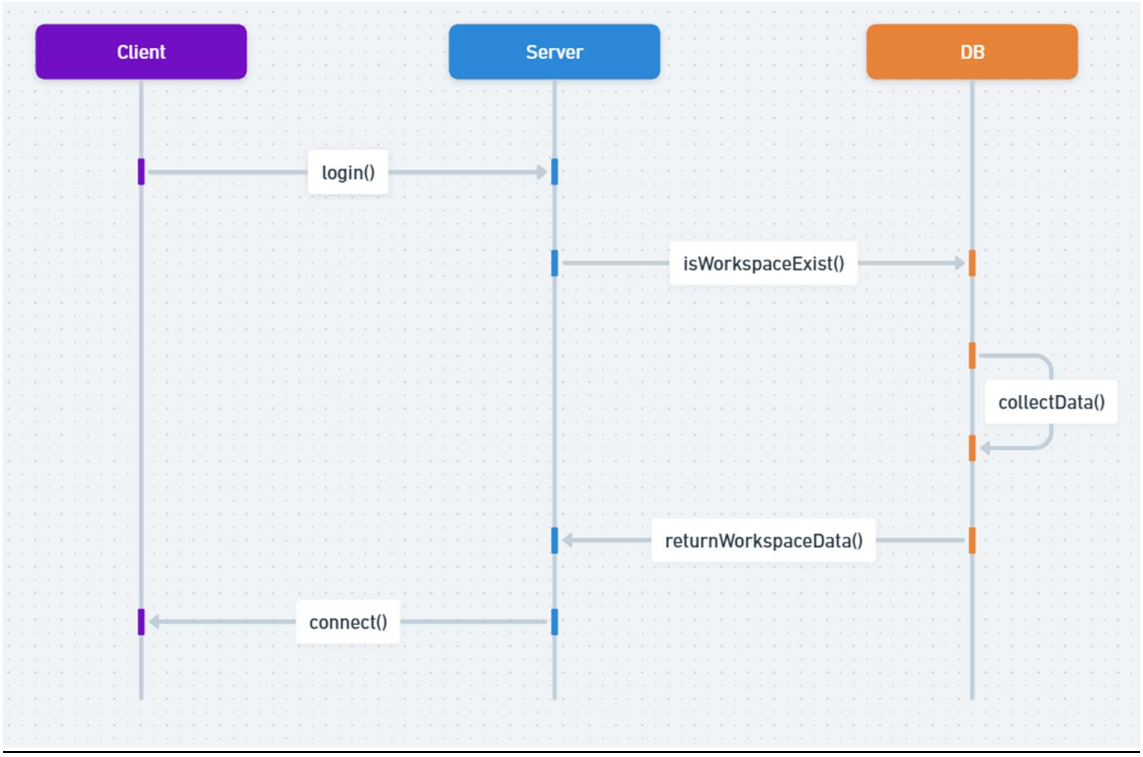
טבלה המתארת את ה-Use Case Diagram

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 שנארזו 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 לכף בשם "הוסף עובד" ויזין את פרטי העובד.
מחיקת עובד	בעמדת המנהל, המנהל ילחץ על הכפתור 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

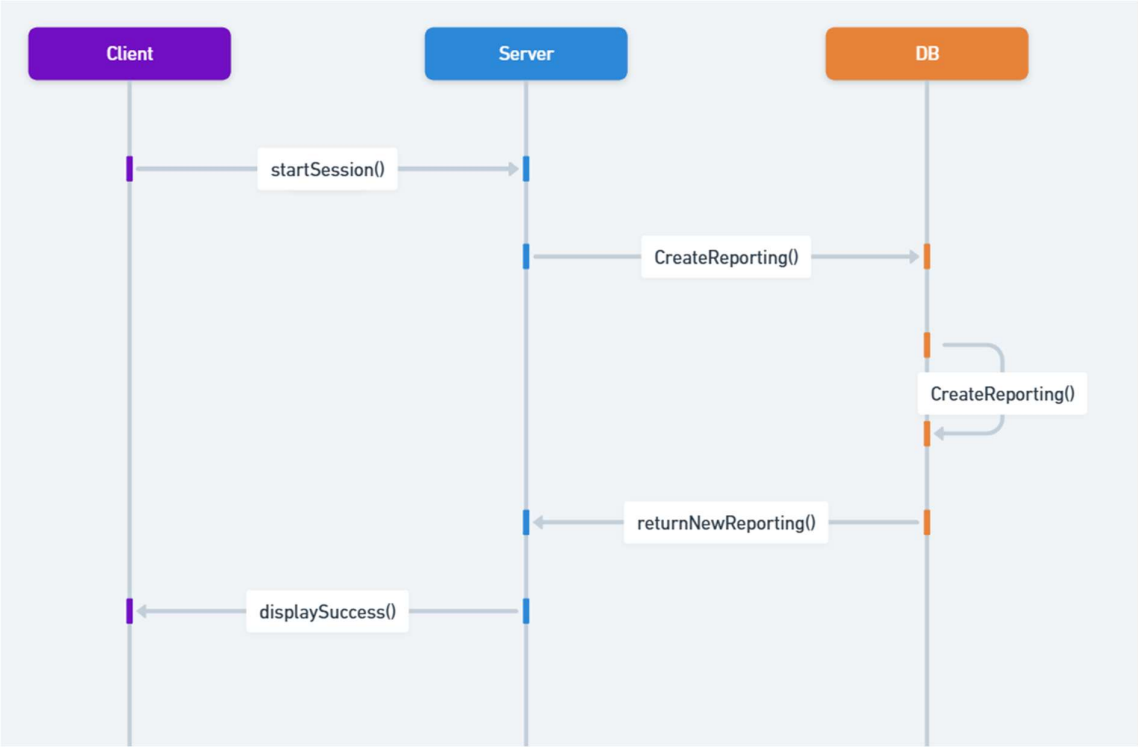
Login to workspace



תהליך זה מתאר את הליך התחברות לעמדה.

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

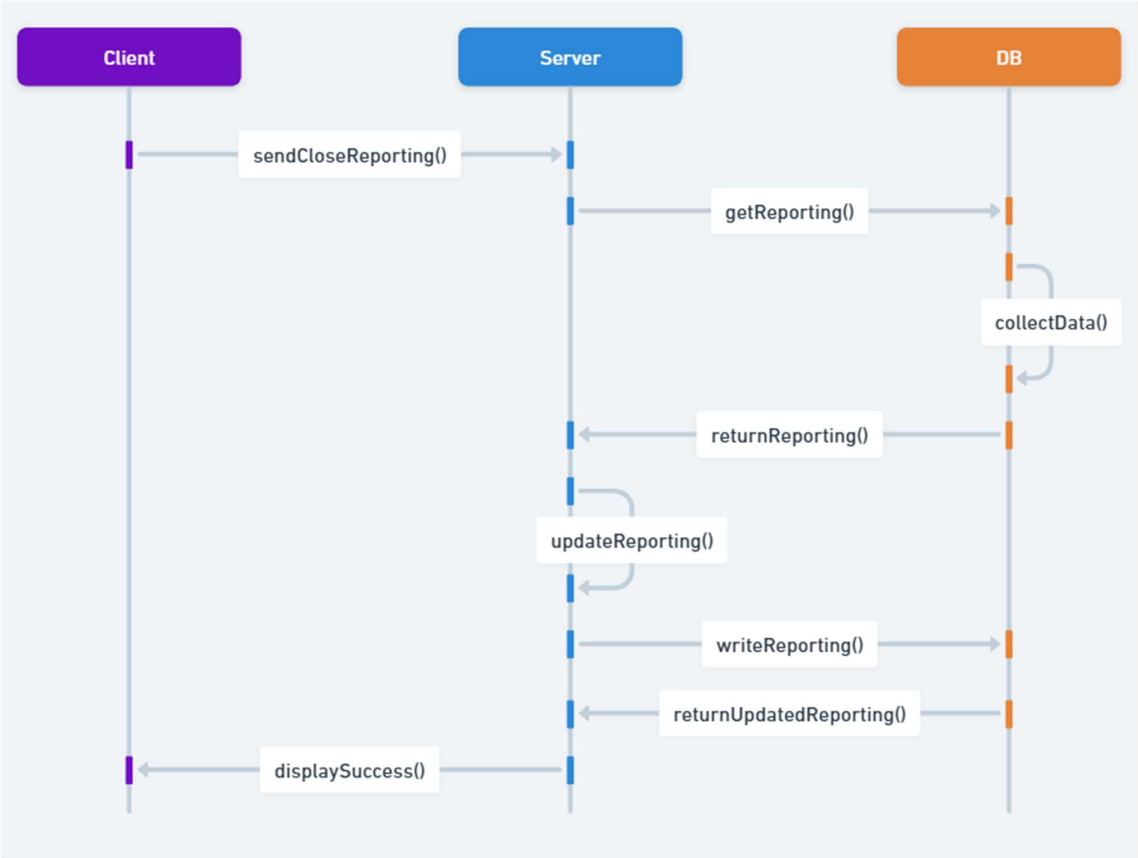
Start working session



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

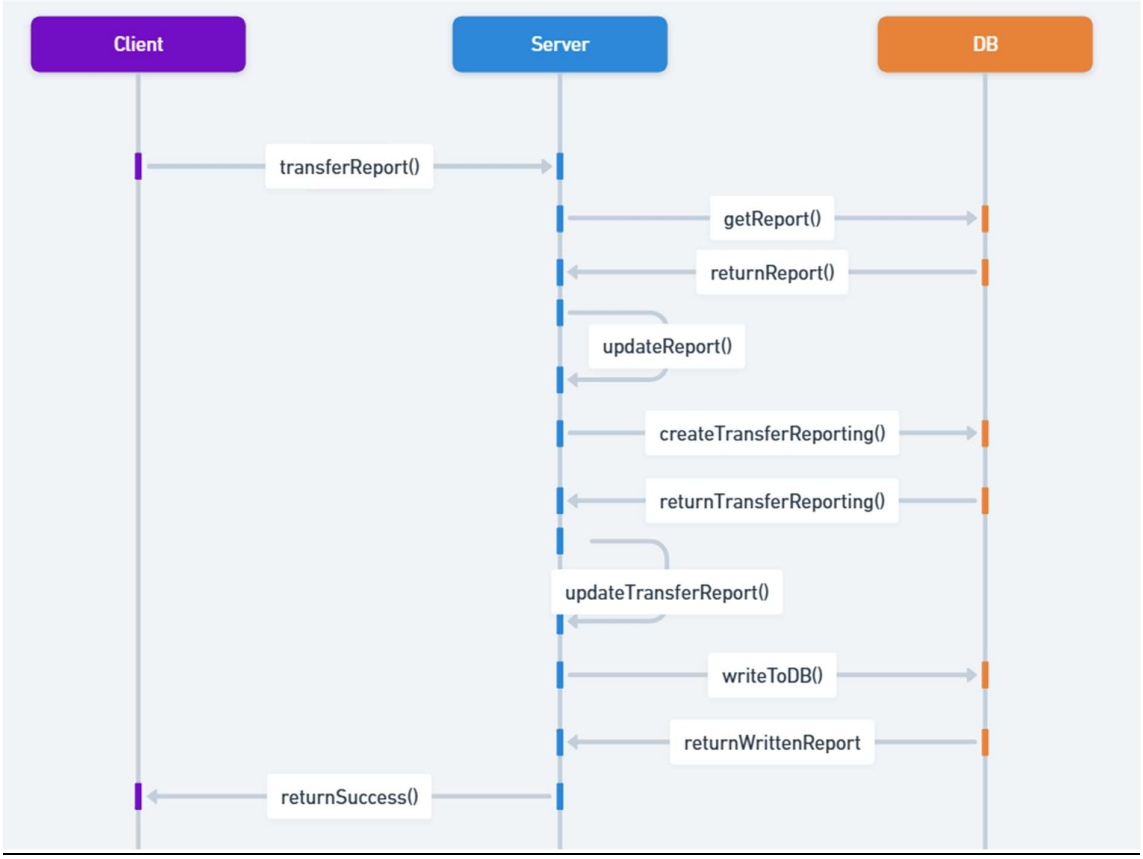
Process of closing all kinds of reporting



תהליך זה מתאר את הליך סיום ודיווח של המחסן, 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 המחסן 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

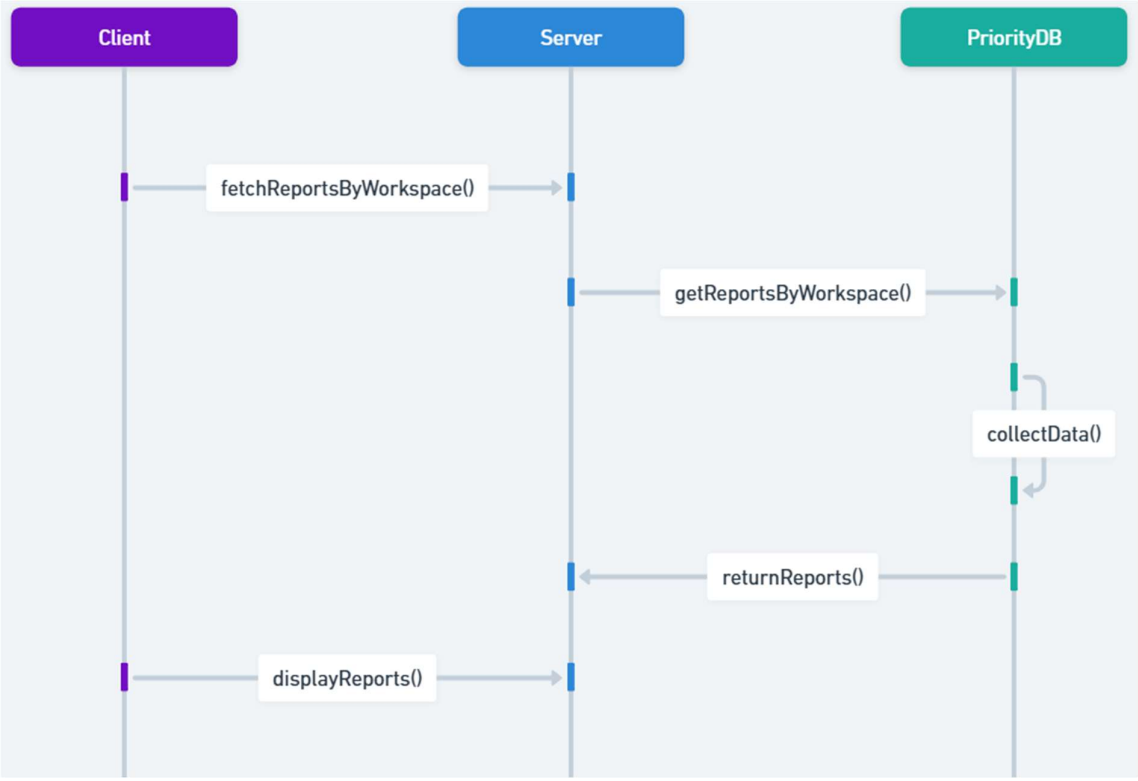
Send/Receive report process



תהליך זה מתאר את הליך שליחה/קבלה

- בשלב הראשון, המשתמש ילחץ על כפתור החץ ויופיע 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

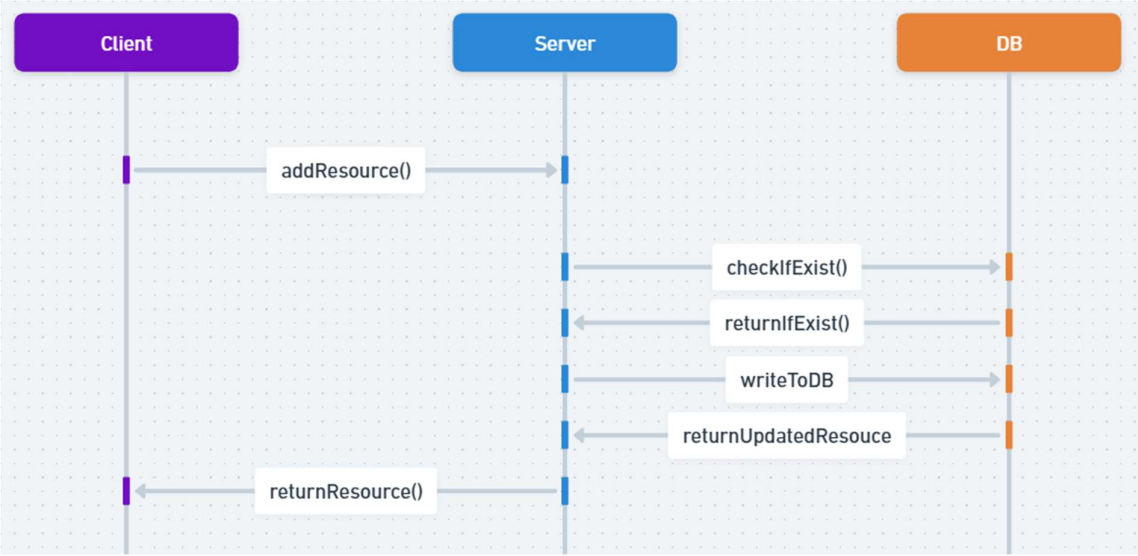
Update Event Process Sequence Diagram



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

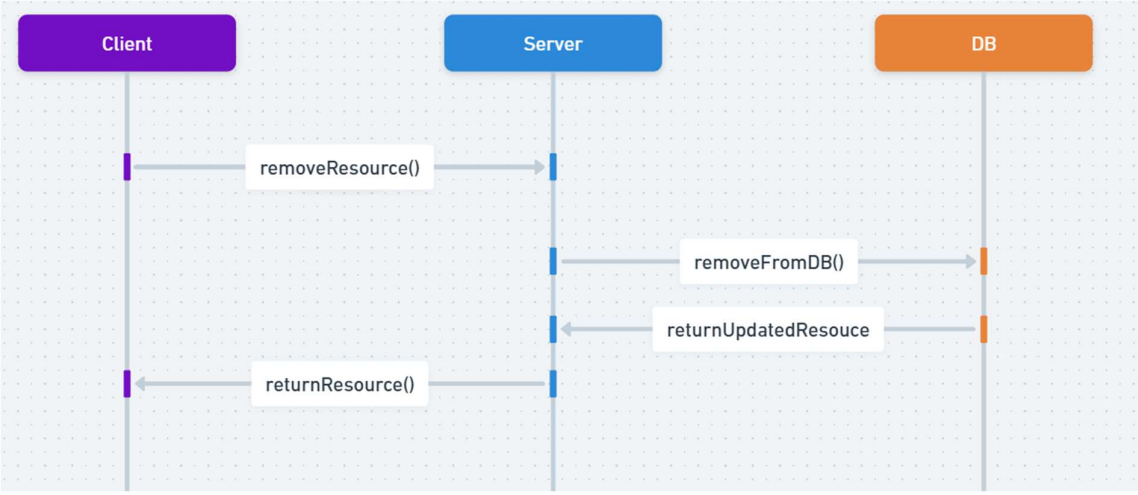
Delete Event Process Sequence Diagram



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

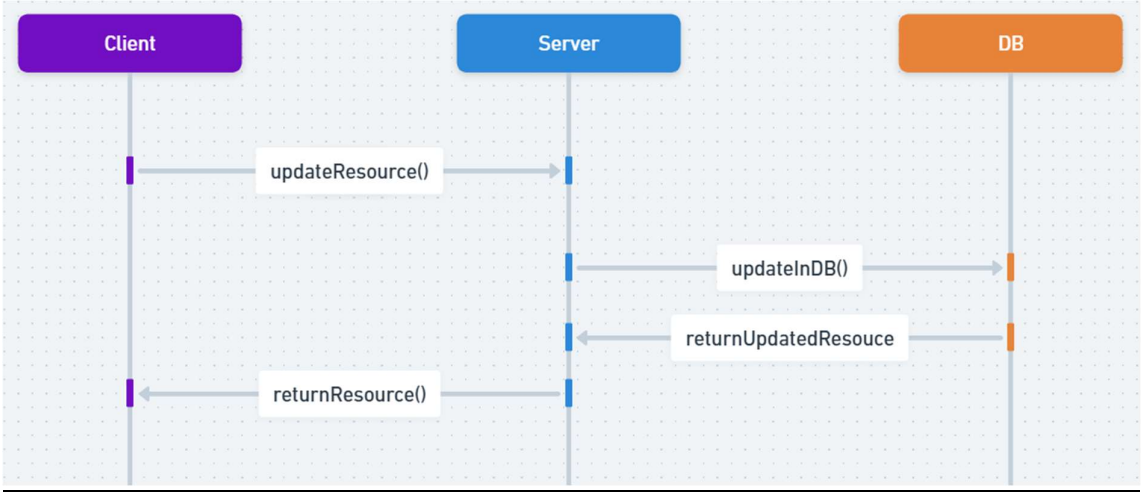
View Event Process Sequence Diagram



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

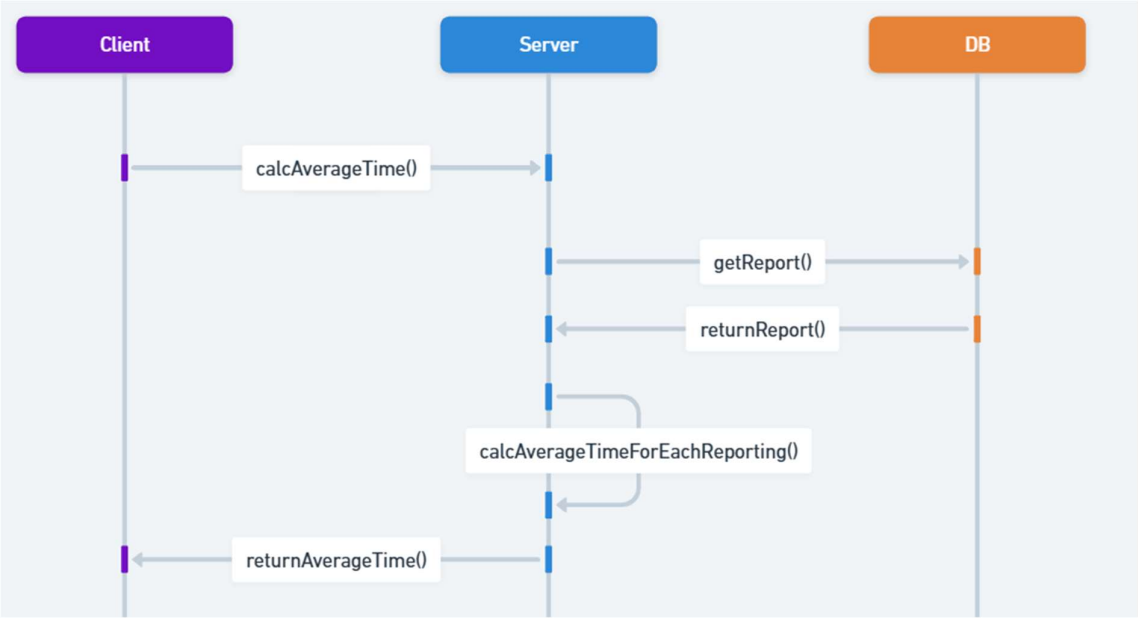
View My Events Process Sequence



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



תהליך זה מתאר חישוב זמן ממופע 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

שיקולי תכנון

צד שרת

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

1. **סקלאביליות (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 NPM (Node Package Manager) 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.

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' ולרנדר 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 מייעל 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

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. **סקלאביליות 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.

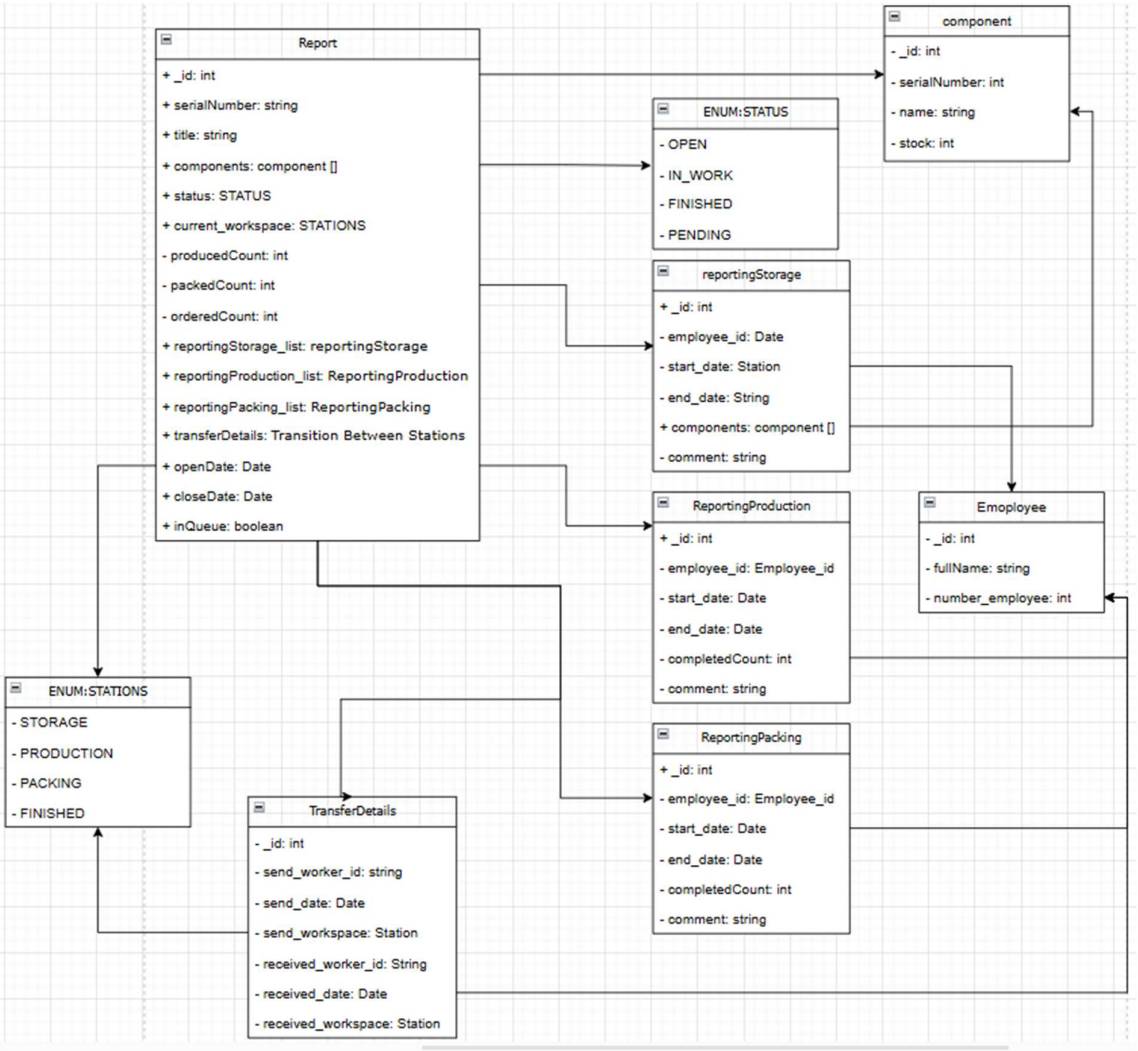
7. **פשטות התקנה וניהול**
   - MongoDB הוא מסד נתונים קל להתקנה ולתחזוקה, עם ממשק משתמש נוח לניהול מסדי הנתונים.
   - תכונות מובנות כמו גיבוי, שחזור וניהול אינדקסים מקלות על תחזוקת הנתונים.
8. **תמיכה במבני נתונים מורכבים**
   - MongoDB מאפשר אחסון נתונים מורכבים ומקוננים (Nested Documents) בתוך מסמך יחיד, דבר המפחית את הצורך בחיבורים (Joins) ומייעל את הביצועים.
9. **ניהול ביצועים ואינדקסים**
   - MongoDB תומכת באינדקסים מתקדמים, שמאפשרים אופטימיזציה של שאילתות 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.

ממסד הנתונים

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.

תכנון ממסד הנתונים



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.