ספר פרויקט גמר

שם המגמה: טכנאי תוכנה

מסלול ההכשרה: מיגו מחזור ב'

שם הסטודנט: רפאל מאר ושלמה ויסמן

שם המנחה:

תאריך מסירת ספר הפרויקט:

תאריך:

לכבוד יחידת הפרוייקטים מה"ט

הצעה לפרויקט גמר

פרטי הסטודנטים

שם	ת.ז.	כתובת	נייד	שנת סיום
רפאל מאר	311625396	גבעת זאב	0559597208	2024
שלמה ויסמן	318715141	ירושלים	0585506998	2024

שם המכללה: מרכז החרדי להכשרה מקצועית ירושלים.

סמל המכללה: 72235.

מסלול הכשרה: טכנאי תוכנה.

מגמת לימוד: מיגו-תוכנה.

מקום ביצוע הפרוייקט: ירושלים.

פרטי המנחה האישי

שם	כתובת	נייד	תואר	מקום עבודה
יצחק לוינסון	מעלה מכמש 528	0528834830	Bs.c	מרכז החרדי להכשרה מקצועית

חתימת הסטודנט חתימת המנחה האישי חתימת הגורם המקצועי מטעם מה"ט

_____ _____ _____

תודות:

אנו, **רפאל מאר** ו**שלמה ויסמן**, רוצים לנצל במה זו ולהביע את תודתנו למרצה **אלכס** ולמנחה **יצחק לווינסון** על העזרה וההדרכה במהלך הכנת הפרויקט.

למוסד הלימודים **מרכז החרדי להכשרה מקצועית ירושלים**, על מסגרת וסביבת הלימודים שסופק לנו, התמיכה והמעטפת במהלך הלימודים.

לחברת **מיגו**, על ההזדמנות ללמוד תוכנה ולהשתלב בתעשיית ההיי-טק, על העזרה והעידוד במהלך כל תקופת הלימודים, הסדנאות וההרצאות.

אנו מעריכים את ההזדמנות ללמוד אצלכם, רכשנו כלים רבים לעבודה כמתכנתים וידע רב שאנו מאמינים שישמשו אותנו היטב בעתיד.

תודה על המקצועיות, האכפתיות, תשומת הלב, במהלך כל הלימודים ובמיוחד בתקופת הכנת הפרוייקט, אתם חלק משמעותי בהצלחה שלנו ועל כך תודתנו.

מודים,

רפאל מאר ושלמה ויסמן.

הצהרת עבודה עצמית

אנו, רפאל מאר ושלמה ויסמן, מצהירים כי בשיתוף פעולה יחד עבדנו על הפרויקט במשך מעל חודש ובנינו אתר מתקדם עבור מערכת לימודים מקוונת.

תקציר פרויקט:

במהלך הפרוייקט נבנה ומומש אתר לימודים מקוונים המכיל מגוון קורסים, המאפשר בין היתר לשוחח בין התלמידים בקורס (להלן 'פורום'), להעלות חומרים חדשים, להירשם לקורסים, לצפות בתכנים מוקלטים וכן הלאה.

המטרה המרכזית שלנו הייתה ליצור מערכת שנגישה עבור מאותגרי טכנולוגיה, עבור תלמידים ומרצים ללא רקע קודם בשימוש במחשב, כך שהמערכת נבנתה מתוך מחשבה על צמצום פונקציות וכפתורים, תוך מיקסום יעילות המערכת ומקצועיותה. כמו כן אף נוסף בוט עזר עבור ניווט באתר ומציאת קורסים וחומרים ספציפיים.

במהלך הפיתוח השתמשנו במגוון טכנולוגיות עבור בניית אתרים, בין היתר טכנולוגיות כמו React ו-NodeJS כדי לממש את ממשק המשתמש וצד השרת בצורה יעילה וממוטבת היטב. בעזרת טכנולוגיות אלו מימשנו אפשרויות כמו להירשם לקורסים, להעלות תכנים ולצפות בהם, וכדומה.

בצד שרת השתמשנו במסד נתונים עבור ניהול נתונים המשתמשים, בחרנו להשתמש בMongoDB עקב הנוחות שלו בעבודה עם אתרים ושליפת נתונים, באמצעות ספריית mongose חיברנו את האתר למסד הנתונים, וביצענו מגוון אפשרויות כמו הצגת כלל הסטודנטים, הצגת כלל הקורסים ופרטיהם, כמו כן מימשנו אמצעי אבטחה למניעת רישום כפול, אישור רישום ע"י מרצה או באמצעות מייל, וכדו'.

בסיום הפרויקט התקבל אתר מקצועי יעיל ומהימן, המותאם עבור אוכלוסיות מאותגרי טכנולוגיה, המאפשר ללמוד מגוון קורסים שונים מהיצע המערכת או אף לפתוח קורס חדש ולהעלות אליו תכנים.

אנו גאים להציג את הפרוייקט ובכך להנגיש את עולם הלמידה המקוונת עבור אוכלוסיות מאותגרות ולעזור להם ללמוד בדרך המתאימה להם.

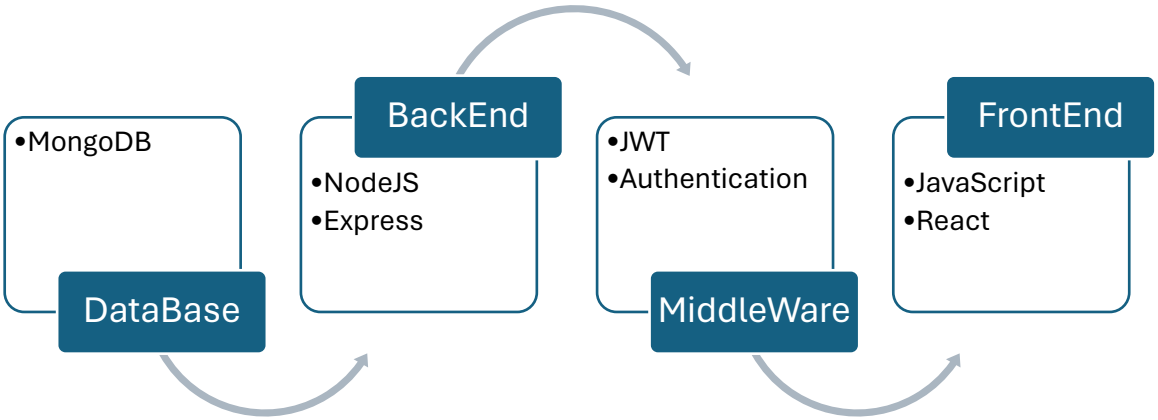
1. **שם הפרויקט**: ClassRoomPlus
2. **רקע:**
   2.1. **תיאור**:
   האתר הינו אתר לימודים מקוונים המאפשר העלאת קורסים, צפייה בהם, שיחה בין משתתפי הקורס ותגובות.
   2.2. **מטרת המערכת**:
   להקל על אוכלוסיות מאותגרות טכנולוגית ולתת להם הזדמנות ללמוד וללמד קורסים מקוונים מבלי לוותר על פונקציונליות המערכת ויעילותה.
3. **סקירת מצב קיים בשוק**
   קיימים מגוון אתרים ופלטפורמות שונות עבור למידה מקוונת, אולם נמצא כי פלטפורמות אלו משוכללות מאד ועמוסות במגוון יישומים ותכונות שעלולות להכביד על משתמשים ללא רקע קודם בשימוש במחשב.
4. **מה הפרויקט אמור לחדש או לשפר?**
   לאפשר לאוכלוסיות מאותגרות טכנולוגית הזדמנות ללמוד בלי 'ללכת לאיבוד בין כל הכפתורים והפונקציות' שבאתר, ללא עומס חזותי או טכנולוגי.
5. **דרישות**
   5.1. **דרישות מערכת**
   המערכת פותחה בסביבת web על גבי NodeJS כאשר שפת הפיתוח הינו JavaScript וסביבת העבודה הינה דרך דפדפן. השרת מבוסס Express תוך שימוש במסד נתונים באמצעות MongoDB. המערכת קיימת בקוד פתוח וניתן להטמיעו במגוון סביבות.
   5.2. **דרישות לקוח**
   בין דרישות הלקוח מהמערכת ניתן למצוא:

   <u>מרצים ותלמידים:</u>
   - רישום לקורסים
   - שיח בין מרצים ותלמידים
   - הסרת תלמיד מקורס

   <u>מרצים:</u>
   - העלאת תכנים
   - מחיקת תכנים
   - מחיקת הודעות בפורום
   - אישור רישום תלמיד

   <u>סטודנטים:</u>
   - צפייה בתכנים
   - שליחת הודעות בפורום ומחיקתן
   - הרשמה לקורס
6. **בעיות צפויות במהלך הפיתוח ופתרונות תפעוליות, טכנולוגיות, עומס, ועוד.**
   6.1. תיאור הבעיות:
   6.1.1. עומס: כאשר מספר המשתמשים באתר גדל בצורה משמעותית, עלול להתרחש בעיות ביצועים והאטות בזמני תגובה.
   6.1.2. תצוגה ורספונסיביות: הממשק מותאם למחשב ביתי כך שעלול להתרחש תצוגה לא תקינה על מסכים קטנים מידי.
   6.2. פתרונות:
   6.2.1. שימוש ובחירה בשרתים יעילים.
   6.2.2. הטמעת ספריות שמאפשרות רספונסיביות וחווית משתמש מותאמת יותר למגוון רחב של מסכים.
7. **הטכנולוגיה:**

7.1. דיאגרמת האתר



7.2. טכנולוגיות בשימוש

סביבת העבודה העיקרית היא NodeJS יחד עם ExpressJS במסגרת פיתוח האתר בשפת JavaScript. באמצעות כלים אלו וספריות נוספות כמו BootStrap סיפקנו אתר יעיל ומקצועי ונח לשימוש.

7.3. שפות הפיתוח

בחרנו להשתמש בשפת JavaScript המותאמת היטב עבור פיתוח אתר כולל צד שרת וצד לקוח, בנוסף לכך שקיימים מגוון רחב של ספריות בשפה זו התורמות רבות לפיתוח.

7.4. תיאור הארכיטקטורה

העבודה תהיה במודל שרת-לקוח כמקובל באתרי אינטרנט, עם עיבוד בצד השרת (BackEnd) ובצד הלקוח (FrontEnd). כל זה על גבי פרוטוקול HTTP, תוך הפרדה בין שרת האינטרנט לשרת ה-MongoDB מטעמי אבטחה. הארכיטקטורה הנבחרת היא שלוש שכבות: הצגה, לוגיקה עסקית ואחסון נתונים. מודל זה מספק גמישות, תחזוקה קלה, פיתוח יעיל ומשפר את האבטחה דרך בידוד המידע.

7.5. חלוקה לתוכניות ומודלים:

במסגרת בניית ומימוש האתר בצענו חלוקה למספר שכבות עבור יעילות הפיתוח ופישוטו לרמת קוד יעיל המאפשר ניתוח ודיבוג בכל שלב.

בין השכבות:

<u>שכבת הלקוח:</u>

- **רכיבי עיצוב** וממשק משתמש, כגון עיצובים ואלמנטים, טפסים, כפתורים ועוד המאפשרים לתקשר עם האתר ולהציג את המידע וביצוע מניפולציות עליו.
- **לוגיקת צד-לקוח:** קוד בJavaScript שרץ על הדפדפן בסביבת הלקוח ומבצע פעולות כמו תקשורת עם השרת, אימות פעולות הלקוח, עיבוד בקשות ועוד.

<u>שכבת השרת:</u>

- **ניהול מסלולי הבקשות מהלקוח (Routing)**, וקביעת מימוש הבקשות ובחירת השירותים הרלוונטיים בהתאם למסלול הבקשה.
- **לוגיקת צד-שרת:** קוד בJavaScript שרץ על השרת וממשמ את בקשות הלקוח ומאחזר את הנתונים המבוקשים, תוך גישה למסד הנתונים ואחזרת הנתונים הרלוונטיים עם קיסטומם בהתאם לבקשה.
- **אבטחה**: שימוש בJWT וספריות נוספות כדי לאבטח את נתוני המשתמשים ואימותם תוך ווידוא הרשאות גישה של הלקוח בהתאם לאופי הבקשה.

<u>שכבת הנתונים:</u>

- יישום ושימוש במסד נתונים MongoDB עבור אחסון הנתונים וניהולם.

7.6. סביבת שרת

האתר קיים בקוד פתוח וניתן להריצו על מגוון סביבות שרת בהתאם לבחירת הלקוח.

7.7. ממשק המשתמש

הממשק ירוץ בדפדפן בסביבת הלקוח המאפשר לרנדר קבצי HTML ו-JavaScript וCSS.

7.8. חבילות תוכנה

7.8.1. צד לקוח:

7.8.1.1. AXIOS

7.8.1.2. BOOTSTRAP

7.8.1.3. REACT

7.8.2. צד שרת

7.8.2.1. BCRYPT

7.8.2.2. CORS

7.8.2.3. EXPRESS

7.8.2.4. JWT

7.8.2.5. VALIDATOR

7.8.2.6. mongoose

7.8.2.7. error handler

8. שימוש במבני נתונים וארגונם

8.1. פירוט מבני הנתונים

שימוש במבני נתונים שאינו רלציוני (NoSQL)
טבלאות וסכמות יוצגו בנספח מצורף.

8.2. אחסון הנתונים

נבחר לאחסן את הנתונים באמצעות MongoDB ושימוש בספריית mongonose עבור גישה לנתונים ואחזורם.

8.3. מנגנוני התאוששות מתקלות.

בוצע שימוש בספריית טיפול בשגיאות עבור מניעתן או אחזורן, כמו כן MongoDB מספק אפשרות לגבות את מסד הנתונים בענן.

9. **תרשימי מערכת**

9.1. **תרחישי שימוש**

9.2. סדר פעולות

```
+-------------+          +------+          +-------------+
| Student     |          | Site |          |  Teacher    |
+-------------+          +------+          +-------------+
      | Login/Signup       |                     |
      |--------------------->|                   |
      | View Courses       |                     |
      |--------------------->|                   |
      | Enroll in Course   |                     |
      |--------------------->|                   |
      |                    | approve enroll      |
      |                    |<---------------------|
      | Submit post        |                     |
      |--------------------->|                   |
      | View posts         |                     |
      |<---------------------|                   |
      | Communicate with   |                     |
      | Teacher            |                     |
      |--------------------->|                   |
      |                    | Forward Message     |
      |                    |--------------------->|
      |                    | Return Response     |
      |                    |<---------------------|
      | Receive Response   |                     |
      |<---------------------|                   |
      +----------------------+                   |
```

9.3. Data Flow

10. תיאור המרכיב התיכנותי

10.1. אלגוריתם

האלגוריתם יקבל מידע מהמשתמשים אודות הקורסים כגון דירוג קורסים, ממוצע זמן למידה, והשתתפות בפורום, וידרג אותם בהתאם. כך ייתן גישה והכוונה למשתמשים חדשים וקיימים לבחור את הקורסים המתאימים להם ביותר.

10.2. איסוף מידע וניתוחים סטטיסטיים (אנליטיקות)

באתר שלנו, אנחנו מקיימים מערכת שמאגדת את מספר המשתתפים בכל קורס, כולל הזמן המושקע בכל קורס והשתתפות בפורום הקורס. בנוסף, אנו סוקרים את נתוני הקורסים ושומרים עליהם במערכת הנתונים שלנו. כל אלו מאפשרים לנו לספק מידע מדויק ומקיף עבור ניתוחים סטטיסטיים.

11. אבטחה

- שימוש בפרוטוקול https לצורך הצפנה של הנתונים בתקשורת בין ה-client ל-server.
- שימוש בספריית Bcrypt להצפנת סיסמאות משתמשים לפני אחסנתם במסד הנתונים.
- בקרת גישה מבוססת שם משתמש וסיסמה לממשקי הניהול וממשקי המשתמש האישיים.
- בקרת הרשאות על בסיס הגדרת הרשאות משתמש בהתאם לסוג המשתמש.
- שימוש בטוקנים בעת חיבור המשתמש עבור זיהויו תוך שימוש בתוקף מוקצב בזמן.

12. משאבים ודירשות עבור פיתוח הפרוייקט

12.1. זמני פיתוח:

הפרוייקט פותח במשך כחודשיים עם חלוקת עבודה בין חברי הצוות, בין הייתר הוקצו משימות וזמני פיתוח.

12.2. ציוד נדרש:

מחשב רב ליבות עם דיי ראם זיכרון, כמו כן חיבור לרשת וזיכרון מחשב פנוי.

12.3. תוכנות נדרשות:

Visual Studio Code

MongoDB & MongoDB Compass

Git
Postman

12.4. שפות, כלים וספריות:

JavaScript
NodeJS
Reat
Express

12.5. **ספרות ומקורות מידע**

https://react.dev/
https://nodejs.org/en
https://www.w3schools.com/js/
https://www.mongodb.com/docs

13. **תוכנית עבודה ולוח זמנים**

#	שבוע 1	שבוע 2	שבוע 3	שבוע 4	שבוע 5
תכנון וחלוקת עבודה	***				
תשתית		***	***		
פיתוח והרחבה				***	
סיום פרויקט					***

14. **רשימת הבדיקות**

בדיקה 1	האם המשתמש קיים במערכת
בדיקה 2	האם האימייל והסיסמה תקינים
בדיקה 3	האם המשתמש רשום לקורס
בדיקה 4	האם המשתמש מאושר לקורס הנוכחי
בדיקה 5	האם המשתמש רשאי למחוק את הפוסט
בדיקה 6	מה רמת ההרשאות של המשתמש
בדיקה 7	האם טוקן המשתמש תקף
בדיקה 8	האם המשתמש רשאי למחוק קורס

15. **בקרת גרסאות**

נבחר שימוש במערכת Git עבור בקרת גרסאות ומעקב אחרי שינויי קוד ומתן אפשרות לשחזור במקרה תקלת קוד וכדומה. כמו כן השתמשנו במערכת GitHub עבור אחסון בענן ואפשרות עבודה בצוות.

נספחים

1. טבלאות וסכמות נתונים

1.1. סכמת יוזר

Field Name	Type	Required	Unique	Additional Validation
firstName	String	Yes	No	'The user must have a name'
lastName	String	Yes	No	'The user must have a name'
email	String	Yes	Yes	'Please provide an email'
phone	Number	Yes	Yes	'Please provide a phone number'
password	String	Yes	No	'Must be a valid password', minLength: 6, select: false
confirmPassword	String	Yes	No	'Please confirm password', minLength: 6, validate: password match
role	String	default: 'user'	No	enum: ['admin', 'user'], 'The role must be either "admin" or "user"'

courses	Array of ObjectId references	No	No	' ref: 'Course

1.2. סכמת קורס

Field Name	Type	Required	Unique	Additional Validation
courseId	ObjectId	No	No	None
courseName	String	Yes	No	'The course must have a name'
openDate	Date	Yes	No	'Please provide start date'
endDate	Date	Yes	No	'Please provide end date'
description	String	No	No	None
price	Number	No	No	None
subscription	Array	No	No	None
subscription.userId	ObjectId	Yes	No	ref: 'User'
subscription.role	String	No	No	value: ['teacher', 'student']
contents	Array	No	No	None
contents.posts	ObjectId	No	No	ref: 'Post'
userId	String	No	No	None

1.3. סכמת פוסטים

Field Name	Type	Required	Unique	Additional Validation
userId	ObjectId	No	No	ref: 'User'
courseId	ObjectId	Yes	No	ref: 'Course', 'Must have a valid classid'
postData	String	Yes	No	None
postFiles	Array	No	No	files: { type: String }
createdAt	Date	default: now	No	None

2. קטעי קוד נבחרים

2.1. צד שרת:

2.1.1. אימות משתמשים

```
const signToken = id => {
    return jwt.sign({id, iat: Date.now()}, process.env.JWT_SECRET, {
        expiresIn: process.env.JWT_EXPIRES_IN
    });
};

const createSendToken = (user, statusCode, res) => {
    const token = signToken(user._id);
    const cookieOptions = {
        expires: new Date(
            Date.now() + process.env.JWT_EXPIRES_IN * 24 * 60 * 60 * 1000
        ),
        httpOnly: true,
        secure: true
    }

    res.cookie('jwt', token, cookieOptions);
    res.status(statusCode).json({
        status: 'success',
        token,
        data: {
```

```
            user
        }
    });
};
exports.register = asyncHandler(async (req, res, next) => {
    const {email, password, confirmPassword, firstName, lastName, phone, role} =
req.body
    if (!email || !password || !confirmPassword || !firstName || !lastName ||
!phone || !role) return next(new AppError(403, 'Request details are missing'))
    const newUser = await User.create({
        email,
        password,
        confirmPassword,
        firstName,
        lastName,
        phone,
        role
    }).catch(err => {
        return next(new AppError(403, 'Email or Phone already exists'))
    })
    createSendToken(newUser, 201, res)
})

exports.login = asyncHandler(async (req, res, next) => {
    const {email, password} = req.body
    if (!email || !password) return next(new AppError(403, 'Email or password is
missing'))
    const user = await User.findOne({email}).select('+password')
    if (!user || !await user.checkPassword(password, user.password)) return
next(new AppError(403, 'Email or password is not correct 1'))

    createSendToken(user, 201, res)

})
const protect = asyncHandler(async (req, res, next) => {
    const token = req.headers.cookie.split('=')[1]
    if (!token) return next(new AppError(403, 'Please login '))
    const decoded = await promisify(jwt.verify)(token, process.env.JWT_SECRET)
    if (!decoded) return next(new AppError(403, 'Please login '))
    const {id} = decoded
    const user = await User.findById(id)
    if (!user) return next(new AppError(400, 'Please register'))
    req.user = user
    next()
})

exports.restrictTo = (...role) => {
    return async (req, res, next) => {
        if (!role.includes(req.user.role)) {
            return next(
                new AppError(403, 'You do not have permission to perform this
action')
            );
        }
    };
};
```

2.1.2. רישום לקורס וביטול רישום

```
exports.subscribe = asyncHandler(async (req, res, next) => {
    const { _id } = req.params;
    let userId = req.user._id;
    if (req.user.role === 'teacher') {
        userId = req.body.userId;
    }

    const course = await Course.findById(_id);
    const subscriptionIndex = course.subscription.findIndex(sub => sub.userId ===
userId);
    const courseUpdate = subscriptionIndex === -1
        ? { $addToSet: { subscription: { userId: userId, role: undefined } } }
        : { $set: { [`subscription.${subscriptionIndex}.role`]: 'student' } };

    const userUpdate = { $push: { courses: _id } };

    const { course: updatedCourse, user } = await updateCourseAndUser(_id, userId,
courseUpdate, userUpdate);

    res.status(201).json({
        status: 'success',
        course: updatedCourse,
        user
    });
});

exports.subDelete = asyncHandler(async (req, res, next) => {
    const { _id } = req.params;
    let userId = req.user._id;
    if (req.user.role === 'teacher') {
        userId = req.body.userId;
    }

    const courseUpdate = { $pull: { subscription: { userId } } };
    const userUpdate = { $pull: { courses: _id } };

    const { course, user } = await updateCourseAndUser(_id, userId, courseUpdate,
userUpdate);

    res.status(201).json({
        status: 'success',
        course,
        user
    });
});
```

2.1.3. סכמת פוסטים

```
const postSchema = new mongoose.Schema({
    userId: {type: mongoose.Schema.Types.ObjectId, ref: "User"},
    courseId: {
        type: mongoose.Schema.Types.ObjectId, ref: "Course",
        required: [true, "Must have a valid classid"]
    },
    postData: {
        type: String, required: true,
```

```
    },
    postFiles: {type: Array, files: {type: String}},
    createdAt: {type: Date, default: Date.now}
});
```

2.1.4. צינרור (routing) קורסים

```
const router = express.Router()

router.route('/')
    .get(courseControllers.getAllCourses)
    .post(authControllers.protect, courseControllers.addCourse)

router.route('/:_id')
    .get(authControllers.protect, courseControllers.getCourseByID)
    .put(authControllers.protect, courseControllers.updateCourse)
    .delete(authControllers.protect, courseControllers.deleteCourse)

router.route('/subscribe/:_id')
    .put(authControllers.protect, courseControllers.subscribe)
    .delete(authControllers.protect, courseControllers.subDelete)
module.exports = router
```

2.2. צד לקוח:

2.2.1. קרוסלת קורסים

```
const Carousel = ({courses}) => {
  const carouselRef = useRef(null);
  const [startIndex, setStartIndex] = useState(0);

  const scrollLeft = () => {
    if (startIndex > 0) {
      setStartIndex(startIndex - 1);
    }
  };

  const scrollRight = () => {
    if (startIndex < courses.courses.length - 3) {
      setStartIndex(startIndex + 1);
    }
  };

  if (!Array.isArray(courses.courses) || courses.courses.length === 0) {
    return <div className="carousel-container">No courses available</div>;
  }

  const visibleCourses = courses.courses.slice(startIndex, startIndex + 3);
  const courseCards = visibleCourses.map(course => (
      <CourseCard key={course._id} course={course}/>
  ));

  return (
      <div className="carousel-container">
        <div className="carousel" ref={carouselRef}>
          {courseCards}
          <div className="carousel-buttons">
```

```
            <button className="carousel-button"
onClick={scrollLeft}><BsChevronLeft/></button>
            <button className="carousel-button"
onClick={scrollRight}><BsChevronRight/></button>
          </div>
        </div>
      </div>
  );
};
```

2.2.2. תשתית Footer עבור קומפוננטה

```
    return (
        <footer className="footer">
            <div className="footer-section company-info">
                <h2 className="company-name">Company Name</h2>
                <p>&copy; 2024 Company Name. All rights reserved.</p>
            </div>
            <nav className="footer-section nav">
                <ul className="nav-list">
                    <li className="nav-item"><a href="#home" className="nav-
link">Home</a></li>
                    <li className="nav-item"><a href="#about" className="nav-
link">About</a></li>
                    <li className="nav-item"><a href="#services" className="nav-
link">Services</a></li>
                    <li className="nav-item"><a href="#contact" className="nav-
link">Contact</a></li>
                </ul>
            </nav>
            <div className="footer-section social-media">
                <a href="https://facebook.com" className="social-link">Facebook</a>
                <a href="https://twitter.com" className="social-link">Twitter</a>
                <a href="https://instagram.com" className="social-
link">Instagram</a>
            </div>
            <div className="footer-section contact-info">
                <p>Email: info@company.com</p>
                <p>Phone: (123) 456-7890</p>
            </div>
        </footer>
    );
};
```

2.2.3. עיצוב דשבורד

```
.dashboard-container {
    width:  600px;
    margin: auto;
    padding: 20px;
    text-align: center;
    background-color: #f9f9f9;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    display:flex;
    flex-direction: column;
```

```
    margin-top: 100px;

  }

  .dashboard-title {
    font-size: 24px;
    font-weight: bold;
    margin-bottom: 20px;
  }

  .dashboard-button {
    width: 100%;
    padding: 10px 20px;
    margin: 5px;
    font-size: 16px;
    border: none;
    border-radius: 5px;
    background-color: #007bff;
    color: #fff;
    cursor: pointer;
    transition: background-color 0.3s;
  }
```

2.2.4. רישום משתמש

```
const Subscription = ({ user, courseId, showSubscription, setShowSubscription }) =>
{
    const [course, setCourse] = useState({});
    const [isLoading, setIsLoading] = useState(true);
    const [error, setError] = useState(null);
    const [subscriptionStatus, setSubscriptionStatus] = useState(null);
    console.log(user.token)

    useEffect(() => {
        const fetchData = async () => {
            try {
                const response = await
axios.get(`http://localhost:3000/courses/${courseId}`,
                    { headers: { jwt: `${user.token}` },
                        withCredentials: true });
                setCourse(response.data.courses);
                setIsLoading(false);
            } catch (error) {
                console.error('Error fetching data:', error);
                setError(`Error fetching course data:
${error.response.data.error}`);
                setIsLoading(false);
            }
        };
        fetchData();
    }, [courseId]);

    const handleSubscribe = async () => {
        try {
            const response = await axios.put(
                `http://localhost:3000/courses/subscribe/${courseId}`,
```

```
                { userId: user.data.user._id, role: 'student' },
                { withCredentials: true }
            );
            if (response.data.status === 'success') {
                setSubscriptionStatus('subscribed');
            } else {
                setError('Failed to subscribe to the course');
            }
        } catch (error) {
            console.error('Subscription error:', error);
            setError('Failed to subscribe to the course');
        }
        setShowSubscription(false);
    };

    const handleSubscribeNo = () => {
        setShowSubscription(false);
    };

    if (isLoading) {
        return <div>Loading...</div>;
    }

    return (
        <>
            <div>
                {subscriptionStatus === 'subscribed' ? (
                    <p>You have successfully subscribed to this course!</p>
                ) : null}
                {error && <p>Error: {error}</p>}
            </div>
            <div>
                {showSubscription && (
                    <div className="confirm-modal">
                        <p>Are you sure that you want to subscribe?</p>
                        <div className="buttons">
                            <button className="confirm"
onClick={handleSubscribe}>Yes</button>
                            <button className="cancel"
onClick={handleSubscribeNo}>No</button>
                        </div>
                    </div>
                )}
            </div>
        </>
    );
};

export default Subscription;
```

2.2.5. קומפנטת אימות מייל

```
const Verifi = ({userId}) => {
    const navigate = useNavigate();

    const handleApproveuser = async () => {
        console.log('handleApproveStudent');
        try {
```

```
            console.log('userId:', userId);
            const response = await
axios.put(`http://localhost:3000/mail/verfieduser/${userId}`);
            console.log('Student approved:', response.data);
        } catch (error) {
            console.log('Error approving student:', error);
        }
    };

    useEffect(() => {
        handleApproveuser();
    }, []);

    return (
        <div>
            <h1></h1>
            <button className="btn btn-primary" onClick={() => navigate('/login')}>
                Back to Home
            </button>
        </div>
    );
}
```

תודה רבה

רפאל ושלמה.