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

**הגשת פרויקט**

**<Braude News> <A7>**

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
| פאדי סרור | 322253758 |
| אחמד מחאמיד | 315038208 |
| קותיבה מחאמיד | 207976382 |
| עבדאללה עמרייה | 212209670 |

תקציר הפרויקט - עד חצי עמוד- פונקציונליות מרכזיות לפי משתמשים:

**הפרויקט שלנו הינו אתר צבירת חדשות הניתן להתאמה אישית: חדשות ממקורות שונים ומאפשרת למשתמשים להתאים אישית את פיד החדשות שלהם על סמך תחומי עניין והעדפות.**

**להלן תקציר של הפונקציונליות המרכזיות של האתר:**

1. **הוספת מאמר כמאמר מועדף: המשתמש יכול להוסיף את המאמר כ** favorites**, המשתמש נכנס לדף פרופיל שלו ושם הם יופיעו.**
2. **מחיקת מאמר מ** favorites**: המשתמש נכנס לדף פרופיל שלו והוא יכול להסיר את המאמר מ** favorites.
3. **חיפוש מאמרים: מנוע החיפוש תומך בתכונה** auto completion **שזה עוזר למשתמש בחיפוש מאמרים.**
4. **צפייה במאמרים לפי קטגוריה: אם המשתמש ביצע** Login **הוא יכול לצפות במאמרים לפי קטגוריה שהוא בוחר או אפשר לצפות במאמרים רנדומליים. (בוחר את קטגוריה** Random**)**

**מימוש- הטכנולוגיות המרכזיות בפרויקט:**

**סביבת עבודה:** fronted) – VS Code using React, JavaScript and HTML)

**סביבת עבודה:** backend) – VS Code using Node.js, Express.js and JavaScript)

**אחסון ומסד נתונים:** backend) – MongoDB)

**ספריות תקשורת צד שרת לקוח:** fronted)– Axios)

**ספריות ניווט**: (fronted)– React Router

**עיצוב:** fronted)- CSS and tailwind)

**כלי לניהול:** - git

[**https://github.com/za3bor/Web-Project.git**](https://github.com/za3bor/Web-Project.git)

[**https://finalclient-iocv.onrender.com**](https://finalclient-iocv.onrender.com)

[**https://www.morethanwallet.com/app/702**](https://www.morethanwallet.com/app/702)

**מהנדס מערכת:** קותיבה מחאמיד.

הסטודנט קותיבה מחאמיד יש לו רקע בבניית אתרים ולכן החלטנו שהוא יהיה מהנדס המערכת.

היה ממשק בין חברי הצוות, במהלך פגישות ב Microsoft teams חלקנו את העבודה למשימות לא קשורות אחת לשנייה כך שנעשו בצורה יעילה. כל העדכונים בוצעו דרך GitHub בהתאם וגם עדכון שאר חברי הקבוצה אחרי כל עדכון. בזמן הקבצת הקוד, נעשתה בדיקת איכות ותיקון לכל באג או פרצופים שנתגלו. כשהגיע זמן איחוד הקוד נכנסו ל Microsoft teams או במכללה ואיחדנו את הקוד בצורה יעילה.

שיתוף הפעולה המתמשך והתקשורת היעילה בצוות תרמו להשלמת משימות בצורה מוצלחת ומקצועית.

|  |  |  |
| --- | --- | --- |
| שם חבר הצוות | משימות שהוקצו | משימות שהושלמו |
| עבדאללה עמרייה | 1. שינוי + שיפור הקוד של מסך פרופיל ל React 2. שיפור הקוד של Server 3. שיפור הפונקציונליות של Search Bar 4. Requirements 5. DB structure diagram. | הושלמו ✔ |
| קותיבה מחאמיד | 1. שינוי + שיפור הקוד של מסך ראשי (מסך של המאמרים) ל React 2. שיפור ה DB של המערכת 3. שינוי + שיפור הקוד של מסכי Navbar ל React 4. שיפור הקוד של Server 5. Requirements | הושלמו ✔ |
| פאדי סרור | 1. שינוי + שיפור הקוד של מסכי Login + Register + ל React 2. פיקוח – בדיקות, איתור באגים. 3. מימוש עוד פונקציונליות ב server. 4. The folder and file diagram 5. A user file that includes screenshots and explanations. | הושלמו ✔ |
| אחמד מחאמיד | 1. שינוי + שיפור הקוד של Dark, light mode ל- React 2. שינוי + שיפור הקוד של "לשמור מאמר" ל React 3. Use case diagram 4. שינוי + שיפור הקוד של Forgot password ל React | הושלמו ✔ |

**Non-Functionality Requirements:**

* The system should load news articles quickly.
* The application should handle many simultaneous users without significant performance degradation.
* The system should be designed to scale horizontally to accommodate increasing user loads.
* The system should be available and reliable, with minimal downtime.
* Implement proper error handling and logging mechanisms.
* The application should be compatible with major web browsers (Chrome, Firefox, Safari, Edge) and devices (desktop, tablet, mobile).
* The user interface should be intuitive and easy to navigate.
* The system should have text is readable and user-friendly.
* Provide clear instructions and user assistance where needed.
* Regularly back up user data to prevent data loss.

**Functionality Requirements:**

* Users should be able to register an account.
* Users should be able to log in using email/password.
* Users should be able to reset their password.
* Users should have individual profiles.
* Users can customize their profiles by adding interests, preferred news sources, and keywords.
* The system should aggregate news from various sources.
* Users should be able to customize their news feed by selecting preferred topics, sources, or keywords.
* The news feed should be dynamic and update in real-time based on user preferences.
* Users should have the ability to save articles.
* Saved articles should be organized into profile.
* The system should provide a search feature for users to find specific articles, topics, or sources.
* The system should provide auto completion feature in search.
* The application should provide a responsive and user-friendly design across various devices.

A diagram of a web service

Description automatically generated

1. **Frontend**:

* **ReactJS:** Create the user interface using ReactJS for structure, Tailwind for styling, and JavaScript for interactivity.
* **Tailwind CSS:** Utilize Tailwind CSS for quickly styling your UI components.
* **JavaScript:** Implement client-side logic for handling user interactions, making API requests, and updating the UI dynamically.

1. **Backend**:
   * **Node.js**: Use Node.js to build the backend of your application. It's efficient for handling asynchronous I/O operations.
   * **Express.js**: A web application framework for Node.js that simplifies routing, middleware, and other common tasks.
2. **Database**:
   * **MongoDB**: A NoSQL database that stores data in JSON-like documents. It's flexible and scalable, suitable for various types of applications.
   * **Mongoose**: An Object Data Modelling (ODM) library for MongoDB and Node.js, providing a higher level of abstraction for interacting with MongoDB.
3. **APIs**:
   * **External APIs**: Integrate with external APIs for fetching data or performing specific actions. Use JavaScript's fetch API or libraries like Axios to make HTTP requests.
4. **Application Architecture**:
   * **Client-Server Architecture**: Follow the traditional client-server architecture where the frontend interacts with the backend through HTTP requests.
5. **Deployment:**

* **Server (Dashboard render):** Handles backend logic, requests, and database interactions.
* **Client (Vercel):** Hosts frontend code for user interface and interaction.

**דיאגרמת מבנה DB.**

|  |
| --- |
| user |
| ID (ObjectID) |
| username (String) |
| password (String) |
| email (String) |
| phone (String) |
| category (Array of String) |
| country (String) |
| jobTitle (String) |
| bio (String) |

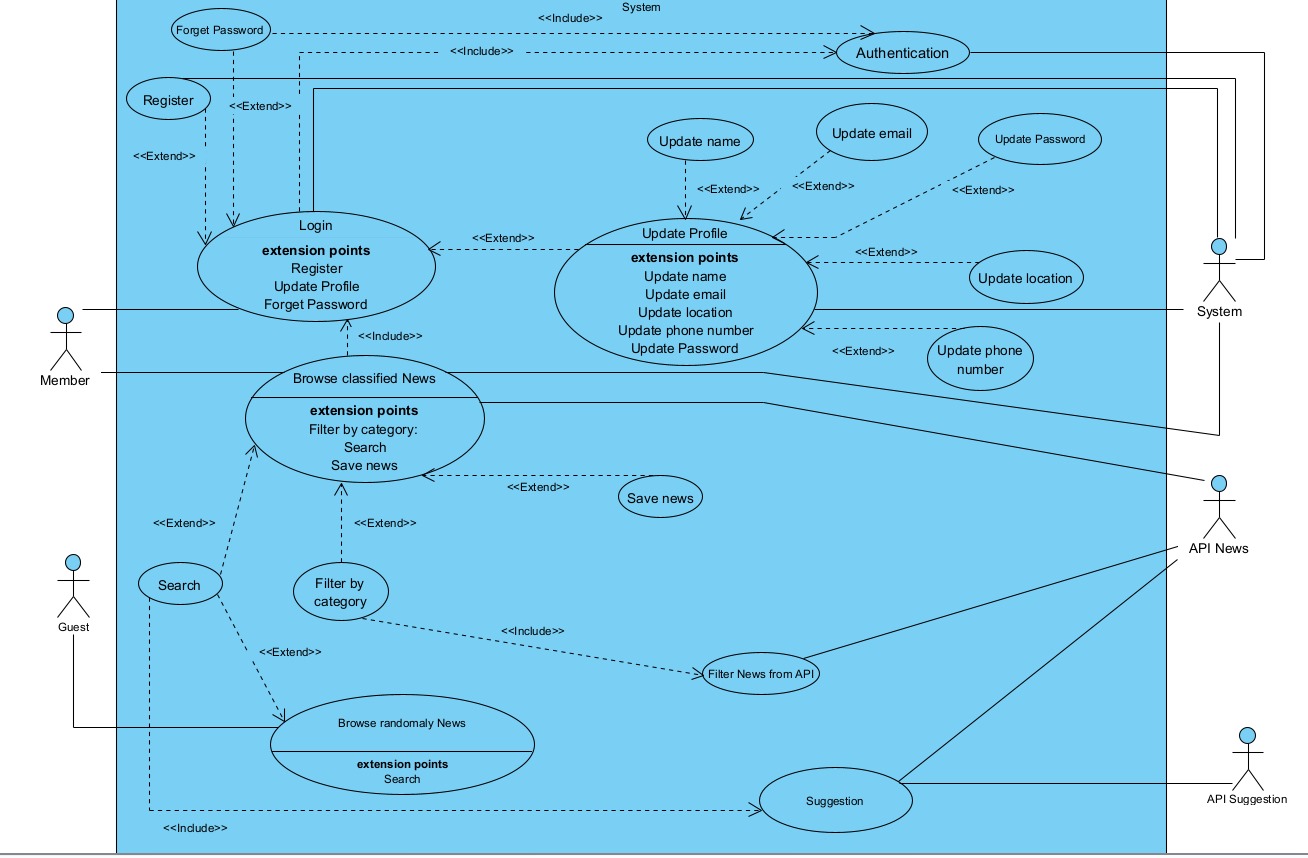
|  |
| --- |
| article |
| ID (ObjectID) |
| username (String) |
| author (String) |
| title (String) |
| description (String) |
| url (String) |
| urltoImage (String) |
| publishedAt (String) |
| isFavorite (Boolean) |

**דיאגרמה המתארת את התיקיות והקבצים השונים.**

finalclient/  
┣ node\_modules/   
┃ ┣ .bin/

┣ public/  
┃ ┣ favicon.ico  
┃ ┣ index.html  
┃ ┣ manifest.json  
┃ ┗ robots.txt  
┣ src/  
┃ ┣ App.css  
┃ ┣ App.js  
┃ ┣ ArticleList.js  
┃ ┣ ForgotPassword.js  
┃ ┣ index.css  
┃ ┣ index.js  
┃ ┣ InterestModal.js  
┃ ┣ Login.js  
┃ ┣ navbar.js  
┃ ┣ Profile.js  
┃ ┣ ProfileCard.js  
┃ ┣ profileNavbar.js  
┃ ┣ Register.js  
┃ ┣ reportWebVitals.js  
┃ ┣ savenews.js  
┃ ┣ ShowArticle.js  
┃ ┗ useDarkSide.js  
┣ .ignore  
┣ package-lock.json  
┣ package.json  
┣ README.md  
┗ tailwind.config.js

**Use case**



**תיק מתכנת:**

**Server:**

**function handleDataRequest():**

This function handles incoming HTTP requests for news articles based on a specified category. It makes an asynchronous call to the News API using Axios, retrieves the articles, and sends them back to the client as a JSON response. Any errors during this process are logged, and an appropriate error response is sent to the client.

**function handleSearchRequest():**

This handleSearchRequest function processes search requests for news articles. It logs a message indicating the start of fetching search data, then defines an asynchronous request handler. Within this handler, it retrieves the search query from the request, makes an API call to the News API using Axios with the search query, filters the articles to remove any with null fields, and sends the filtered articles back to the client as a JSON response. In case of an error during the process, it logs the error and sends an internal server error response to the client.

**function handleLoginRequest(mongoDB):**

This handleLoginRequest function handles login requests by checking the provided username and password against a MongoDB collection named 'user'. It expects a MongoDB client (mongoDB) to be passed as an argument. Within the request handler, it retrieves the username and password from the request query, logs them, and then queries the 'user' collection in MongoDB to find a matching user. If a user is found, it logs a successful login message, updates the 'login' field for the user, and sends a JSON response indicating a successful login. If no user is found or the credentials are invalid, it logs a login failure message and sends an appropriate JSON response. In case of any errors during this process, it logs the error and sends an internal server error response to the client.

**function handleRegistrationRequest(mongoDB):**

This function handles user registration requests by checking if all required fields are present and if the username is available. If the requirements are met, it inserts the new user into the MongoDB collection and sends a success message. If any validation fails or an error occurs, it sends an appropriate error response.

**function handleArticleRemoveRequest(mongoDB):**

This handleArticleRemoveRequest function processes requests to remove articles from a MongoDB collection. It checks for the presence of required fields (username and title) in the request body and removes the specified article from the 'article' collection. It then sends a success message if the removal is successful or appropriate error responses if validation fails or errors occur during the process.

**Client:**

const ArticleList = ({ category }) => {

// Define state variables for loading status and articles list

const [loading, setLoading] = useState(true);

const [articles, setArticles] = useState([]);

// Fetch articles when the component mounts or when the category changes

useEffect(() => {

const fetchArticles = async () => {

try {

setLoading(true); // Set loading state to true while fetching data

setArticles([]); // Clear the articles array to show loading state or empty state

const defaultCategory = category;

let response;

// Fetch articles based on the default category

if (defaultCategory ==='Random'){

response = await axios.get(${ipAddress}/api/random);

}

else{

response = await axios.get(${ipAddress}/api/search?query=${defaultCategory});

}

// Update the articles state with the fetched data

setArticles(response.data.articles);

setLoading(false);

} catch (error) {

console.error('Error fetching data:', error);

setLoading(false);

}

};

fetchArticles();

}, [category]);

**// Function to fetch articles based on username**

const fetchArticles = async (username) => {

try {

const response = await fetch(${ipAddress}/api/articles/${username});

if (response.ok) {

const contentType = response.headers.get('content-type');

if (contentType && contentType.includes('application/json')) {

const data = await response.json();

setArticles(data.articles);

} else {

console.warn('Response is not JSON:', await response.text());

// Handle the case where the response is not JSON, e.g., by displaying an error message

}

} else {

throw new Error('Failed to fetch articles');

}

} catch (error) {

console.error('Error fetching articles:', error);

}

};

**// Function to remove a favorite article**

const RemoveFav = async (article, articleIndex) => { // Pass article and articleIndex as parameters

try {

const response = await axios.post(${ipAddress}/api/remove-article, {

username: localStorage.getItem('name'), // Use localStorage directly here

title: article.title,

});

if (response.status === 200) {

const updatedArticles = [...articles];

updatedArticles.splice(articleIndex, 1); // Remove the article from the list

setArticles(updatedArticles); // Update the state

} else {

throw new Error('Failed to remove favorite');

}

} catch (error) {

alert('Error removing favorite');

console.error('Error removing favorite:', error);

}

};

**Function for search**

useEffect(() => {

if (searchTerm.trim() !== '' && inputClicked) {

fetch(https://api.datamuse.com/sug?s=${searchTerm})

.then(response => {

if (!response.ok) {

throw new Error('Network response was not ok');

}

return response.json();

})

.then(data => {

setSuggestions(data);

setShowSuggestions(data.length > 0);

})

.catch(error => console.error('Error fetching suggestions:', error));

} else {

setSuggestions([]);

setShowSuggestions(false);

}

}, [searchTerm]);

**ספריות שהשתמשנו:**

1. const cors = require('cors');

The cors middleware is used to enable Cross-Origin Resource Sharing (CORS) in your Node.js server. CORS is a security feature implemented by web browsers to restrict web pages from making requests to a different domain than the one that served the page.

1. const express = require('express');

Express.js is a popular Node.js web application framework that simplifies the process of building web applications and APIs. It provides a robust set of features for handling HTTP requests, routing, middleware integration, and more.

1. const MongoDB = require('./mongodb');

In Node.js, require is used to import modules or files into your application. When you use require('./mongodb'), it looks for a file named 'mongodb.js' or 'mongodb/index.js' in the current directory ('./') and imports the functionality exported by that file.

API שהשתמשנו:

<https://newsapi.org/> for articles

<https://www.datamuse.com/api/> for search engine

mongodb+srv://fadisrourza3bor:7BswDK9jiKHfdt32@webproject.pmqjqdn.mongodb.net/WebProject

**\*הערה חשובה:**

צריך לחקות דקה לפני תחילת הבדיקה של האתר כי השתמשנו ב deploy בחינם.

**תיק משתמש:**

**A screenshot of a login box

Description automatically generated**

**A screenshot of a login screen

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a search engine

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**