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

**תרגיל בית מס' 2**

חברי הצוות:(7**A**)

**קותיבה מחאמיד 207976382**

**פאדי סרור 322253758**

**עבדאללה עמרייה 212209670**

**אחמד מחאמיד 315038208**

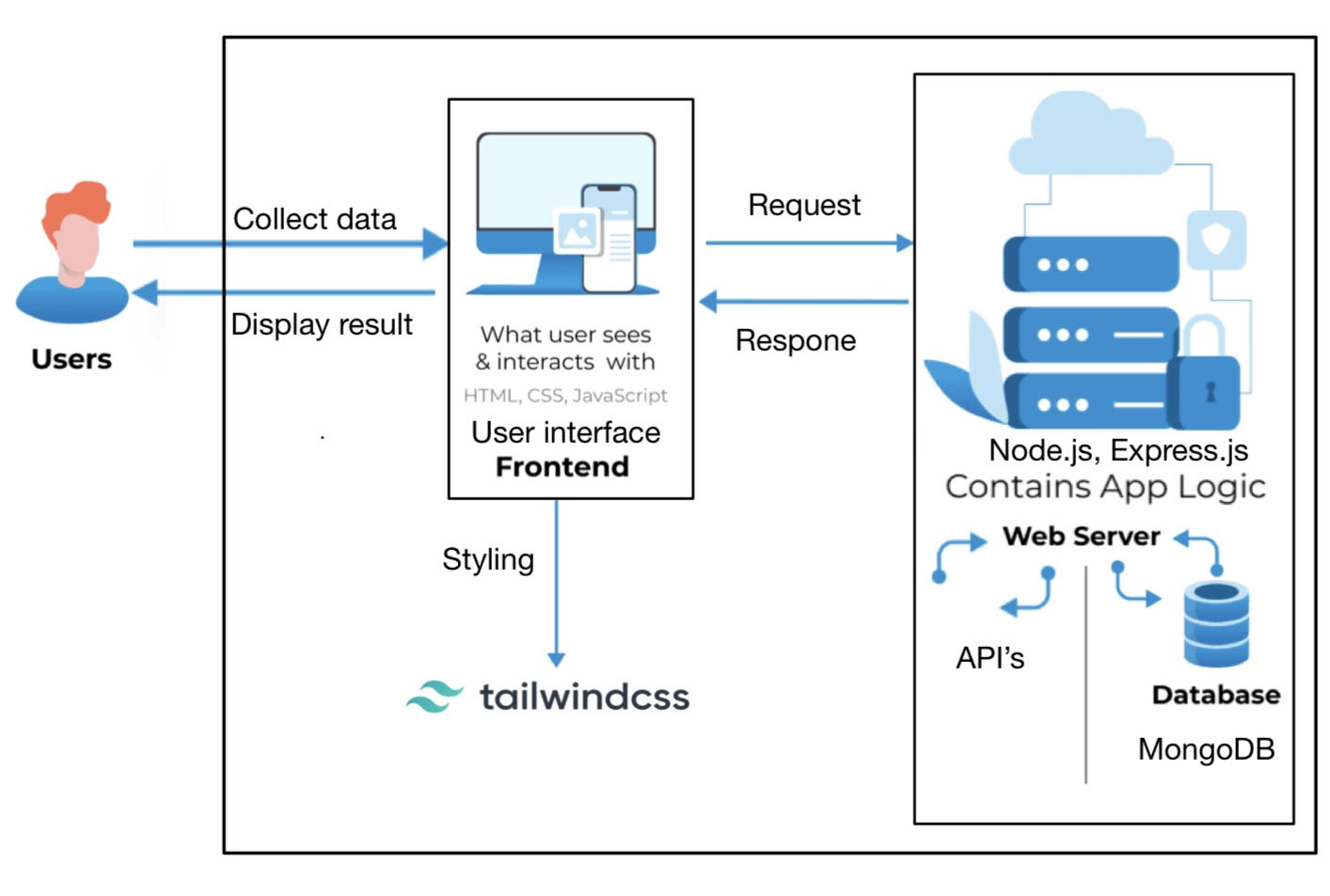
**מהנדס מערכת:** קותיבה מחאמיד.

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. מימוש דף Login + Register והפונקציונליות שלהם.  2. מימוש פונקציונליות עבור Search Bar.  3. מימוש הפונקציונליות עבור הוספת נתונים ל DB.  4. פיקוח - בדיקות, איתור באגים. | הושלמו ✔ |
| קותיבה מחאמיד | 1. מימוש Server.  2. יצירת DB למערכת.  3.שליפת נתונים מ API והצגתם בדף הראשי.  4. שיפר ה HTML של דף הראשי (דף המאמרים) והטעמתו באמצעות JS. | הושלמו ✔ |
| פאדי סרור | 1. מימוש ארכיטקטורה של האתר (diagram).  2. שיפר ה HTML של דף המאמר והטעמתו באמצעות JS.  3. מימוש פונקציונליות של Dropdown (category) והצגת הנתונים לפי מה שבחר המשתמש או לפי מה שרשום ב DB שלו.  4. יצירת הלוגו של האתר. | הושלמו ✔ |
| אחמד מחאמיד | 1. מימוש Use case  2.מימוש דף פרופייל של המשתמש  3. מימוש כפתור favorite כך שמוסיפים את המאמר ל פרופייל של המשתמש והצגתו.  4. מימוש הפונקציונליות עבור הוספת נתונים ל DB. | הושלמו ✔ |

2. הציגו ארכיטקטורה של האתר (תרשים הכולל את האלמנטים המרכזיים).

1. **Frontend**:

**HTML/CSS/JavaScript:** Create the user interface using HTML for structure, CSS (possibly with Tailwind CSS) for styling, and JavaScript for interactivity.

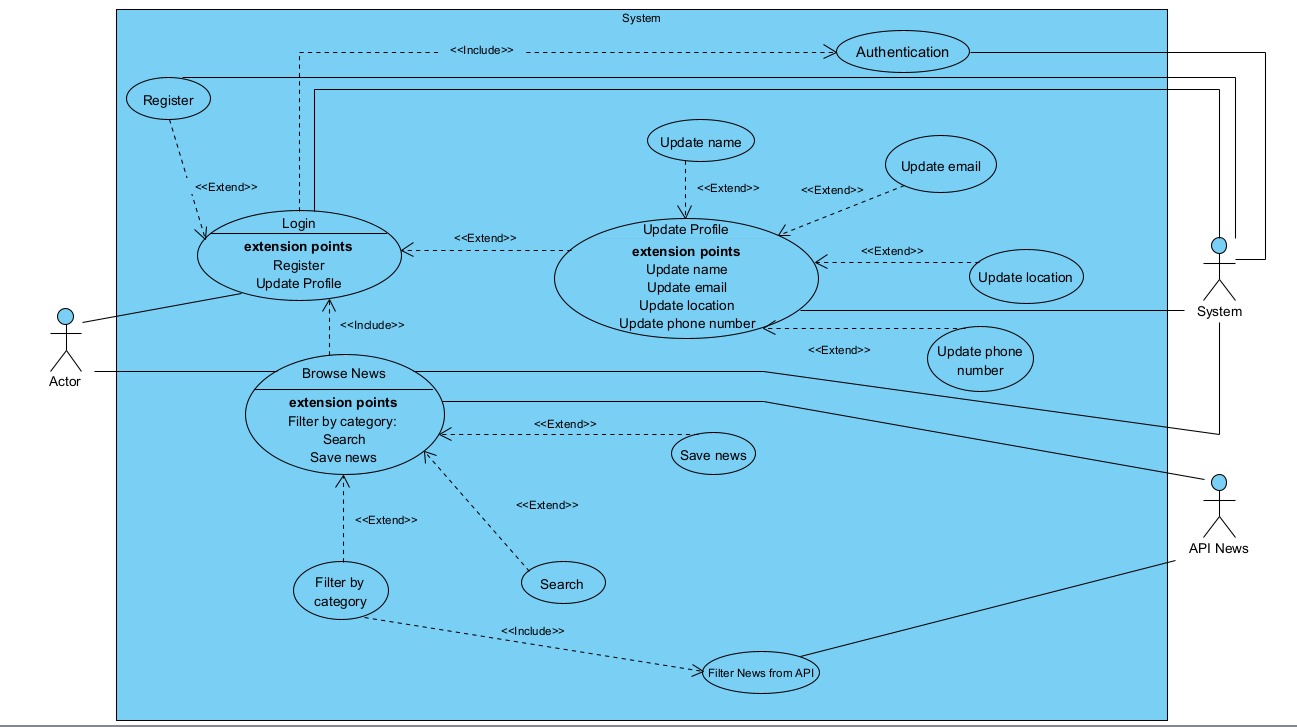
**Tailwind CSS:** Utilize Tailwind CSS for quickly styling your UI components.

**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 Modeling (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.

3. הציגו דיאגרמת use case המתארת את השימוש באתר.

****