Mandatory Opgave UX

Opgaven har to dele:

- Del 1: Lav en frontend (UI) til den udleverede server.
- Del 2: Lav en heuristisk evaluering af en anden gruppes frontend (Jakob Nielsens Heuristik & Steve Kruug's trunk test).
- Hver gruppe skal aflevere:
  1. Koden for jeres frontend.
  2. Rapporten med den heuristiske gennemgang af den anden gruppes UI.

Del 1: UI (frontend) til den udleverede server (backend).

I skal lave forsiden til en webshop, der sælger brugt tøj og sko.

Siden skal have følgende funktionalitet:

- Vise produkterne til salg på webshoppen.
- Afgrænse produkterne efter søgetermer eller type af produkt.
- Jeres webshop skal have en underside eller et flow, der gør det muligt at oprette nye produkter.

Bemærk:

- I behøver **IKKE** at lave funktionalitet der understøtter køb af produkter.

Om opgaven:

- Hver gruppe bestemmer selv om i bruger REACT eller standard HTML+JS til at lave hjemmesiden.
- I må gerne anvende AI til at lave hjemmesiden.
- Opgaven bliver bedømt som enten 'bestået' eller 'ikke bestået' og kræves bestået for at man kan gå til eksamen i faget.

Serveren har tre endpoints, der understøtter den ønskede funktionalitet funktionaliteter.

Her følger en genemgang af den.

Tutorial: Running the Server and Using the API

Installing and Running the Server

This exercise includes a Node.js Express server that serves data for a used clothes marketplace.

Prerequisites:
- Node.js installed (Download from nodejs.org)
- A terminal (Command Prompt, PowerShell, or macOS/Linux terminal)
- A code editor (VS Code recommended)

Step 1: Install Dependencies
- Open a terminal
- CD to the project folder
- run: 'npm install' to install the dependencies

Step 2: Start the Server
- Run the following command: node server.js

After running this, the server will start on http://localhost:5500

Step 3: Verify the Server is Running
Open a web browser and go to:
http://localhost:5500/clothes

You should see a JSON response with the available products.

2. API Endpoints

```
GET /clothes
Returns a list of all available clothing items.
```

Example Request:
GET http://localhost:5500/clothes

Example Response:

```
[
  {
    "id": 1,
    "title": "Vintage Denim Jacket",
    "description": "Classic blue denim jacket with a worn-in look.",
```

```
    "size": "M",
    "price": 35,
    "category": "Jackets",
    "seller": "Emma Green",
    "location": "New York",
    "imageUrl": "/static/images/item_1.png"
  }
]
```

```
GET /clothes/search?query={search_term}
Searches for clothes by title or description.
```

Example Request:
GET http://localhost:5500/clothes/search?query=denim

Example Response:

```
[
  {
    "id": 1,
    "title": "Vintage Denim Jacket",
    "description": "Classic blue denim jacket with a worn-in look.",
    "size": "M",
    "price": 35,
    "category": "Jackets",
    "seller": "Emma Green",
    "location": "New York",
    "imageUrl": "/static/images/item_1.png"
  }
]
```

```
POST /clothes
Adds a new clothing item to the marketplace.
```

Request Body (JSON):

```
{
  "title": "Black Hoodie",
  "description": "Comfortable black hoodie with front pocket.",
  "size": "L",
  "price": 20,
  "category": "Sweaters",
  "seller": "John Doe",
  "location": "San Francisco"
}
```

Example Request:

```
POST http://localhost:5500/clothes
Content-Type: application/json
```

Example Response:

```
{
  "id": 11,
  "title": "Black Hoodie",
  "description": "Comfortable black hoodie with front pocket.",
  "size": "L",
  "price": 20,
  "category": "Sweaters",
  "seller": "John Doe",
  "location": "San Francisco",
  "imageUrl": "/static/images/item_2.png"
}
```

Note: The id is automatically assigned, and the image is randomly chosen.

JavaScript method examples:

Here are a few examples of how to access these endpoints using javaScript.

```
const API_BASE_URL = 'http://localhost:5500/clothes';

// Fetch all clothing items
async function getAllClothes() {
    const response = await fetch(API_BASE_URL);
    return response.json();
}

// Search for clothing items by title or description
async function searchClothes(query) {
    const response = await
fetch(`${API_BASE_URL}/search?query=${encodeURIComponent(query)}`);
    return response.json();
}

// Add a new clothing item
async function addClothingItem(item) {
    const response = await fetch(API_BASE_URL, {
        method: 'POST',
        headers: {
```

```
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(item)
    });
    return response.json();
}
```

Del 2: Heuristisk evaluering af user interfaces

1. Byt jeres user interface kode med en anden gruppe
   (se gruppeoversigten for at se hvilken gruppe i bytter med).
2. Gennemgå den anden gruppes user interface med Jakob Nielsens Heuristik.
   husk at gennemgå alle 10 punkter af den heuristiske evaluering og skriv både severity samt forslag til forbedring. Tag et screenshot til jeres rapport for hvert punkt i finder.
3. Brug forsiden af den anden gruppes website og gennemgå den med Steve kruugs trunk test. Skriv ned for hvert punkt, hvis i har forbedringsforslag.

Aflever:

- Aflever jeres egen gruppes UI-kode samt jeres evaluering af den anden gruppes UI.
- I afleverer på Learnit.