TP-13

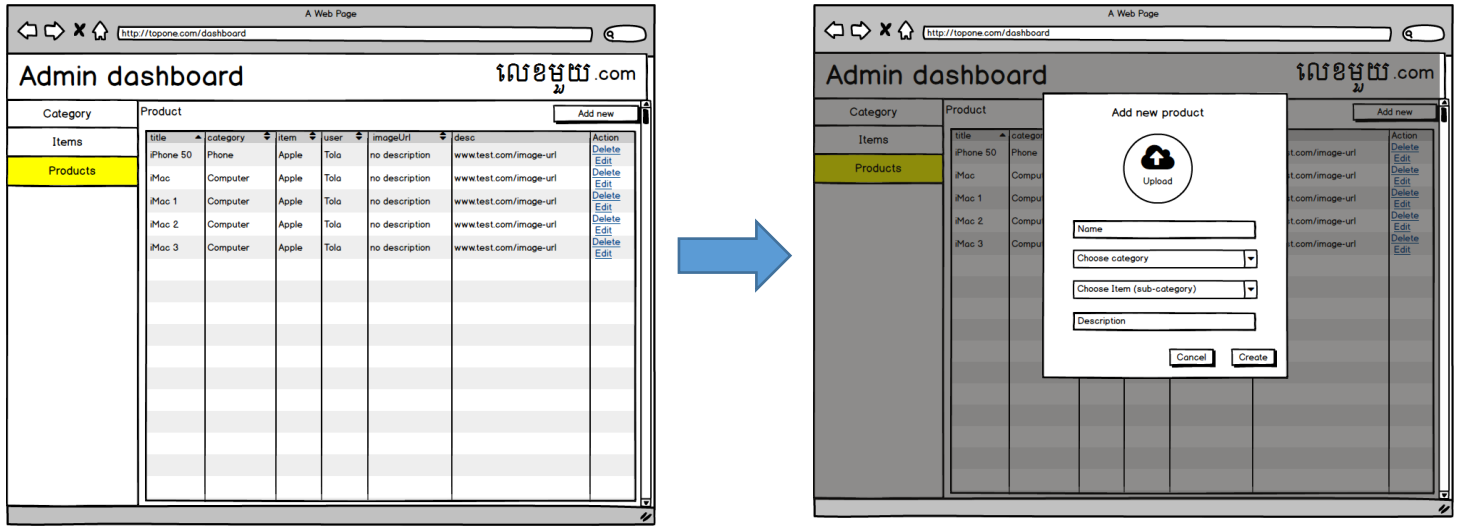
VueJS, NodeJS

CRUD APIs, Mongoose, Admin dashboard

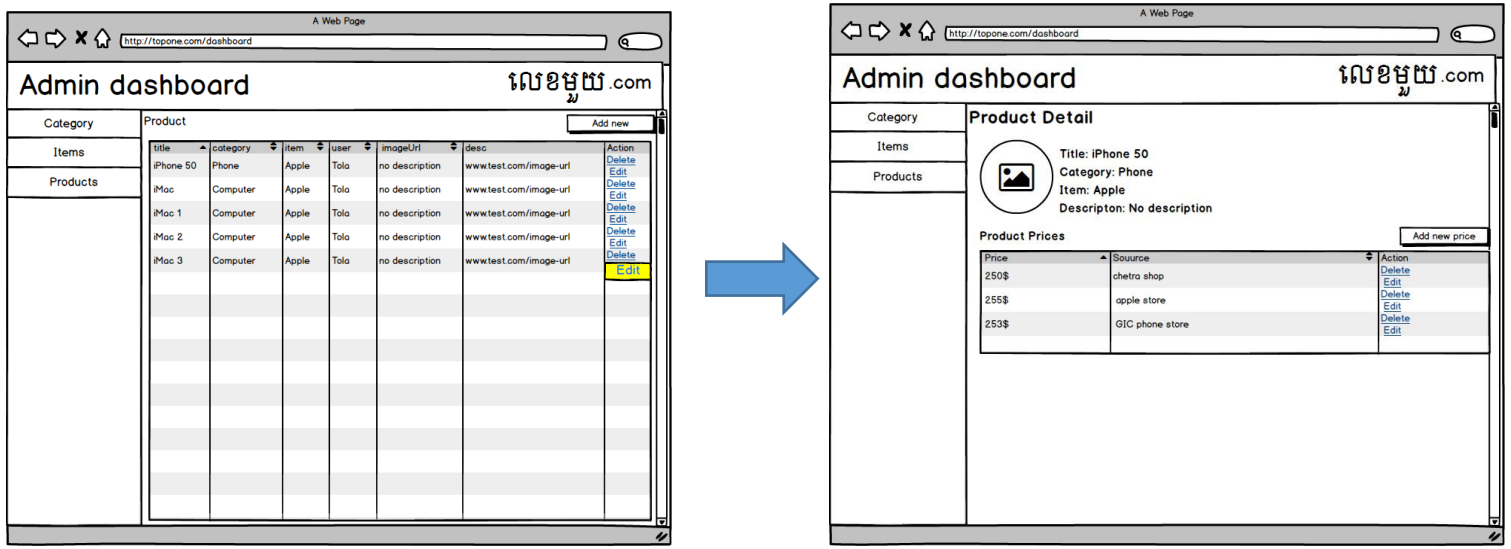
TP13 Exercise

TP13.1: Design a basic admin dashboard to manage product/product prices

Create a new product



Add a new price to a product



Getting to learn another new Thing 🤤

“Multer: upload files with Node.js and Express”

How to upload my image to a server?? 🤔

Tutorial source:

https://blog.logrocket.com/multer-nodejs-express-upload-file/

❖ Creating our frontend

index.html

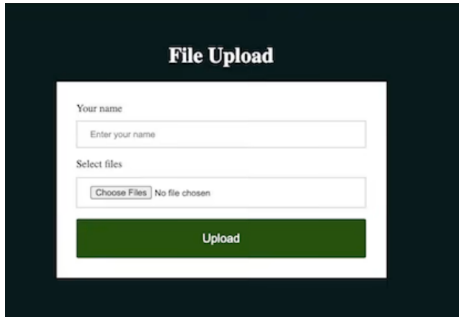
```
<body>
    <div class="container">
        <h1>File Upload</h1>
        <form id='form'>
            <div class="input-group">
                <label for='name'>Your name</label>
                <input name='name' id='name' placeholder="Enter your name" />
            </div>
            <div class="input-group">
                <label for='files'>Select files</label>
                <input id='files' type="file" multiple>
            </div>
            <button class="submit-btn" type='submit'>Upload</button>
        </form>
    </div>
    <script src='./script.js'></script>
</body>
```

script.js

```
// script.js
const form = document.getElementById("form");

form.addEventListener("submit", submitForm);

function submitForm(e) {
    e.preventDefault();
    const name = document.getElementById("name");
    const files = document.getElementById("files");
    const formData = new FormData();
    formData.append("name", name.value);
    for(let i =0; i < files.files.length; i++) {
            formData.append("files", files.files[i]);
    }
    fetch("http://localhost:5000/upload_files", {
        method: 'POST',
        body: formData,
        headers: {
          "Content-Type": "multipart/form-data"
        }
    })
        .then((res) => console.log(res))
        .catch((err) => ("Error occured", err));
}
```



❖ Setting up the server

```
npm init -y
```

```
npm i express
```

server.js

```
// server.js
const express = require("express");

const app = express();
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.post("/upload_files", uploadFiles);
function uploadFiles(req, res) {
    console.log(req.body);
}
app.listen(5000, () => {
    console.log(`Server started...`);
});
```

❖ Install and configure Multer

```
npm i multer
```

server.js

```
const multer = require("multer");
const upload = multer({ dest: "uploads/" });
...
```

we'll use Multer to intercept incoming requests on our API and parse the inputs to make them available on the req object:

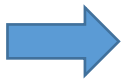
```
app.post("/upload_files", upload.array("files"), uploadFiles);

function uploadFiles(req, res) {
    console.log(req.body);
    console.log(req.files);
    res.json({ message: "Successfully uploaded files" });
}
```

❖ Here is what you got

```
app.post("/upload_files", upload.array("files"), uploadFiles);

function uploadFiles(req, res) {
    console.log(req.body);
    console.log(req.files);
    res.json({ message: "Successfully uploaded files" });
}
```



```
Server started...
[Object: null prototype] { name: 'Images' }
[
  {
    fieldname: 'files',
    originalname: 'undraw_confirmed_81ex.svg',
    encoding: '7bit',
    mimetype: 'image/svg+xml',
    destination: 'uploads/',
    filename: '5cb132f75d66fe877a7e9974bfcdf47b',
    path: 'uploads/5cb132f75d66fe877a7e9974bfcdf47b',
    size: 6128
  },
  {
    fieldname: 'files',
    originalname: 'Atomic_Habits_-_Chapter_1_Excerpt.pdf',
    encoding: '7bit',
    mimetype: 'application/pdf',
    destination: 'uploads/',
    filename: '901a1b930918ca9df759ae5092c10779',
    path: 'uploads/901a1b930918ca9df759ae5092c10779',
    size: 6829248
  },
  {
    fieldname: 'files',
    originalname: 'WhatsApp Image 2020-12-16 at 09.30.13.jpeg',
    encoding: '7bit',
    mimetype: 'image/jpeg',
    destination: 'uploads/',
    filename: '2920d45355c44410e82778b8316ae662',
    path: 'uploads/2920d45355c44410e82778b8316ae662',
    size: 30517
  }
]
```

Good luck 👌

In-class practice:: Finish the remaining APIs in category (+ bonus score)

- **Get by ID**

http://localhost:3001/category/id/:id

- **Get all**

http://localhost:3001/category/all

- **Update**

http://localhost:3001/category/update

- **Delete**

http://localhost:3001/category/delete

☑ **Create**

http://localhost:3001/category/create

☑ **Get categorized items**

http://localhost:3001/category/delete

In-class practice:: Finish the remaining APIs in item (+ bonus score)

- **Get by ID**

http://localhost:3001/item/id/:id

- **Get all**

http://localhost:3001/item/all

- **Update**

http://localhost:3001/item/update

- **Delete**

http://localhost:3001/item/delete

☑ **Create**

http://localhost:3001/item/create

In-class practice:: Finish the remaining APIs in product (+ bonus score)

- **Get by ID**

http://localhost:3001/product/id/:id

- **Get all**

http://localhost:3001/ product/all

- **Update**

http://localhost:3001/product/update

- **Delete**

http://localhost:3001/product/delete

☑ **Create**

http://localhost:3001/product/create

Practice:: Create an API to add price in product (+ bonus score)

routes\price.js (http://localhost:3001/price/create)

```
const priceService = require('../services/price');
```

```
router.post('/create', auth.ensureSignedIn, async (req, res) => {
  const { product, price, source } = req.body;
  const result = await priceService.create({ product, price, source })
  res.json(result);
})
```

services\price.js

```
const Prices = require("../models/prices");
```

```
const create = async (newPrice) => {
  // to do
  const createdPrice = await Prices.create(newPrice);
  return createdPrice;
}
```

models\prices.js (Price Model)

```
var pricesSchema = new mongoose.Schema({
  product: {
    type: Schema.Types.ObjectId,
    ref: 'Products',
    required: true
  },
  price: Number,
  source: String,
}, {
  timestamps: true,
});
```