Etsuko Wardrobe API

Etsuko Wardrobe was inspired by the constant swapping of clothes for nights out. It is named after the building I live in, as this is where it has been happening, but would be easily adaptable for other communities. However a fairly important assumption is that you live with the community, as the website is more for viewing what’s on offer as opposed to handling the specifics of the swap.

This API is to handle the server side of Etsuko Wardrobe. Its main functions are handling clothes, owners and reservations data; and the upload of images when adding new clothes.

There is currently no authentication as it is on a private network, but it is something I will add as I look to make it public.

**Base URL: localhost:3000**

# Clothes

Clothes are displayed as cards and can be loaded depending on their clothing category.

GET /clothesData

Simply returns the clothesData json file to be accessed client side.

**Endpoint URL: localhost:3000/clothesData**

Request Headers: None

Request Parameters: None

Response Format: JSON file

Error Handling: 400, 'Client Error, try a different request'

Code:

app.get('/clothesData', function (request, response) {

try{

response.json(clothesFile)

}

catch (error) {

response.status(400).send('Client Error, try a different request');

}

});

Example Response:

[

{

"name": "navy mesh top",

"owner": "Rose",

"description": "strappy floral mesh top, can be a little itchy",

"size": "M",

"image": "blue\_top.jpg",

"type": "top"

},

{

"name": "blue floral top",

"owner": "Sorine",

"description": "baby blue floral mesh strappy top",

"size": "S",

"image": "blue\_strap\_top.jpg",

"type": "top"

},

{

"name": "grey top",

"owner": "Sorine",

"description": "grey mesh top, need top for underneath",

"size": "S",

"image": "grey\_top.jpg",

"type": "top"

},

{

"name": "halter top",

"owner": "Sorine",

"description": "stripy halterneck top",

"size": "S",

"image": "halter\_top.jpg",

"type": "top"

}

]

POST /addClothesData

Adds the text parts of a new clothing item to the clothesData file.

**Endpoint URL: localhost:3000/addClothesData**

Request Headers: 'Content-Type': 'application/json'

Request Parameters: None

Response Format: status

Error Handling: 200, 'Clothes data received, refresh to see it in the list' or 400, 'Client Error, try a different request'

Code:

app.post('/addClothesData', function (request, response) {

try{

const newClothes = request.body;

clothesFile.push(newClothes);

fs.writeFileSync('data/clothesData.json', JSON.stringify(clothesFile, null, 2));

response.status(200).send('Clothes data received, refresh to see it in the list');

}

catch (error) {

response.status(400).send('Client Error, try a different request');

}

});

Example Response:

'Clothes data received, refresh to see it in the list'

POST /uploadImage

Separately handles the upload of the image into client/images/ using multer and adds the file path to the clothesData JSON file.

**Endpoint URL: localhost:3000/uploadImage**

Request Headers: method: 'POST', body: formData

Request Parameters: formData passed in -> upload.single('image')

Response Format: status

Error Handling: 200, 'Image uploaded successfully' or 400, 'Client Error, try a different request'

Code:

app.post('/uploadImage', upload.single('image'), (request, response) => {

try{

const imagePath = `${request.file.filename}`;

const lastItem = clothesFile[clothesFile.length - 1];

if (lastItem) {

lastItem.image = imagePath;

}

fs.writeFileSync('data/clothesData.json', JSON.stringify(clothesFile, null, 2));

response.status(200).send('Image uploaded successfully');

}

catch (error) {

response.status(400).send('Client Error, try a different request');

}

});

Example Response:

'Image uploaded successfully'

# Owners

Each clothing item has one individual owner, noted by a button at the bottom of the card. The button takes them to the top of the page where their details are displayed.

GET /ownerData

Returns the ownerData JSON file.

**Endpoint URL: localhost:3000/ownerData**

Request Headers: None

Request Parameters: None

Response Format: JSON file

Error Handling: 'Error loading owner:', error.message

Code:

sync function loadOwner(person) {

try {

const response = await fetch('ownerData');

if (!response.ok) {

throw new Error(`Error: ${response.status}`);

}

const ownerData = await response.json();

const ownerDiv = document.getElementById('ownerdiv');

ownerData.forEach(owner => {

if (owner.name === person) {

ownerDiv.innerHTML = `

<div class="card-body">

<p class="card-title">${owner.name}</p>

<p class="card-text">email: ${owner.email}</p>

<p class="card-text">room number: ${owner.room}</p>

</div>

`;

}

})

}

catch (error) {

console.error('Error loading owner:', error.message);

}

}

Example Response:

[

{

"name": "Rose",

"room": "207",

"email": "rose.laird@durham.ac.uk"

},

{

"name": "Sorine",

"room": "202",

"email": "sorine.furnace@durham.ac.uk"

},

{

"name": "Liam",

"room": "204",

"email": "liam.mays@durham.ac.uk"

},

{

"name": "Max",

"room": "206",

"email": "max@durham.ac.uk"

},

{

"name": "Megan",

"room": "212",

"email": "megan.le@durham.ac.uk"

}

]

# Reservations

Clothes can be reserved by clicking on the plus button on the clothing card. The pop up also shows when the clothing item is already reserved, so the user knows when not to book it (unless they’re ready to fight!)

GET /reservations/:id

Returns the dates for where a reservation already exists in the reservationsData file for a specific clothing item, identified by its name.

**Endpoint URL: localhost:3000/reservations/:id**

Request Headers: None

Request Parameters: clothesName

Response Format: JSON file

Error Handling: 400, 'Client Error, try a different request'

Code:

app.get('/reservationsFile/:id', function (request, response) {

try{

const clothesName = request.params.id;

const unavailableDates = reservationsFile.filter(reservation => reservation.name === clothesName);

response.json(unavailableDates);

}

catch (error) {

console.error('Error handling reservations data:', error.message);

response.status(400).send('Client Error, try a different request');

}

});

Example response: id=”navy mesh top”

[

{

"name": "navy mesh top",

"date": "26/01/2024"

},

{

"date": "2024-03-09",

"name": "navy mesh top"

}

]

POST /addReservation

Adds reservation to reservationData file

**Endpoint URL: localhost:3000/addReservation**

Request Headers: method: 'POST',

headers: {

'Content-Type': 'application/json'

}

Request Parameters: body: reservationJSON,

Response Format: status

Error Handling: 200, 'Reservation data received' or 400, 'Client Error, try a different request'

Code:

app.post('/addReservation', function (request, response) {

try {

const newReservation = request.body;

reservationsFile.push(newReservation);

fs.writeFileSync('data/reservationsData.json', JSON.stringify(reservationsFile, null, 2));

response.status(200).send('Reservation data received');

} catch (error) {

response.status(400).send('Client Error, try a different request');

}

});

Example input:

{

"date": "2024-02-02",

"name": "blue floral top"

}

Example response:

'Reservation data received'