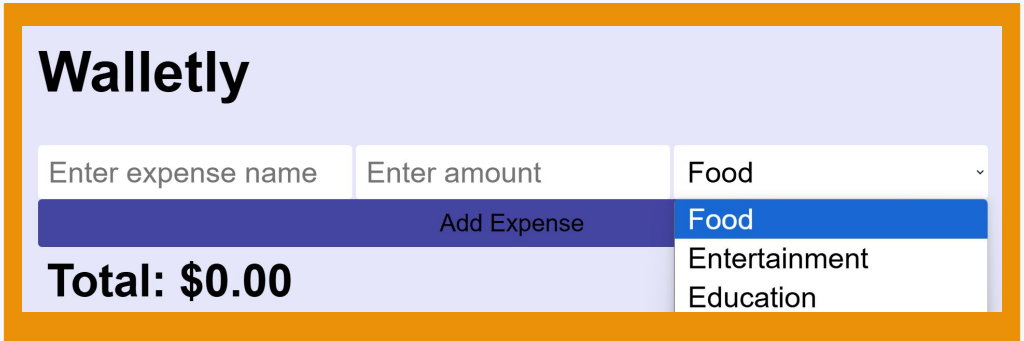
Walletly

MC Fadonougbo and Tram Nguyen

Purpose

- Money Tracking Website to see how much you spend in different categories
- Categories range from
  - Food
  - Entertainment
  - Education
  - Other
- Users input the name of the item, how much they spent on it and Which category it belongs in
- Website tracks how much user spends in total



HTML

- Divided with 4 tags
- form-container class: input fields
- expense-list id: display all expense
- total-expense id: display total amount of money.
- expense-chart id: visualize the chart.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Walletly</title>
  <link rel="stylesheet" href="pj.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  </head>
<body>
  <h1>Walletly</h1>
  <div class="form-container"> <!--inpput fields-->
    <input type="text" id="expense-name" placeholder="Enter expense name">
    <input type="number" id="expense-amount" placeholder="Enter amount">
    <select id="expense-category">
      <option value="Food">Food</option>
      <option value="Entertainment">Entertainment</option>
      <option value="Education">Education</option>
      <option value="Other">Other</option>
    </select>
    <button id="add-expense">Add Expense</button>
  </div>

  <div class="expense-list" id="expense-list"></div> <!--displayall expense-->

  <div id="total-expenses">Total: $0</div><!--display total-->

  <div class="chart-container">
    <canvas id="expense-chart"></canvas><!--visualize chart-->
  </div>
  <script src="ps.js"></script>
</body>
</html>
```

CSS

- Percents - fits portionally with any screen size
- Font-size: changes size
- Flexbox
  - Flexwrap

```css
.form-container{
    display: flex;
    flex-wrap: wrap;
    justify-content: space-evenly;

}

.chart-container {
    width: 50%;
    margin-top: 20px;
    margin: 0 auto;
}
canvas {
    max-width: 100%;
    height: 200px;
}
body {
    background-color: #e6e6fa;
    font-family: Arial, sans-serif;
    margin: 20px;
}
h1{
    margin-top: 2%;
    font-size: 70px;
    margin-bottom: 50px;
}
div.total{
    margin: 10px;
}

#expense-name, #expense-amount, #expense-category {
    padding: 12px;
    font-size: 36px;
    width: 31%;  /* width of the form container 1/3 of the page give or take*/
    border-radius: 5px;
    margin-top: -0px;
    border: 1px solid #ffffff;
    justify-content: space-around;
}
```

CSS

- Flexbox: Evenly space assets
- Width percents: portionately fit with any screen size
- Justify content (horizontally)
- Max Width: so it doesn’t exceed the space set for it

```css
}
#add-expense {
    padding: 12px;
    font-size: 30px;
    background-color: #4444a1;
    color: black;
    width: 98.5%;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
div.total{
    font-size: 56px;
    font-weight: bold;
}
.expense-item {
    font-size: 30px;
    width: 99%;
    margin: 10px 0; /* Add some spacing between list items */
    display: flex; /* To align text and the delete button nicely */
    justify-content: space-between; /* Space out the text and button */
    align-items: center;
    padding: 10px;
}
.expense-item button {
    font-size: 24px;
    padding: 5px 10px;
    border-radius: 5px;
    border: none;  /* gets rid of tiny border around delete button*/
    background-color: #4444a1;
    color: black;
    cursor: pointer;
}
```

JavaScript

1. Add an expense

```
//Add a new expense
addExpenseButton.addEventListener("click", function () {
  const name = expenseNameInput.value.trim();
  const amount = parseFloat(expenseAmountInput.value.trim()); //Convert to a number
  const category = expenseCategoryInput.value;

  //Validate the input
  if (!name || isNaN(amount) || amount <= 0) {
    alert("Please enter valid expense details!");
    return;
  }

  //Send data to server (use post request)
  fetch('http://localhost:3000/add-expense', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ name, amount, category }), //Convert expense data into JSON format
  })
    .then(response => response.json())  //Parse the server's response into a JSON object
    .then(data => {
      console.log("Server response:", data); //Log the server response
      if (data.message === 'Expense added successfully') {
        expenseNameInput.value = ""; //Clear input
        expenseAmountInput.value = "";
        printList(); //Fetch the update expense list + refresh display
      } else {
        alert('Error adding expense!');
      }
    })
    .catch(err => {
      alert('Failed to connect to server!');
    });// .catch() -> handle errors that may occur during fetch request.
});//if omitting: web may still work, but we wont see any feedback -> ?the web is broken???
```

JavaScript

2. Print the expense list.

```
//Display expense list
function printList() {// send request to /expenses to retrieve the expense list
  fetch('http://localhost:3000/expenses') // Take data from server
    .then(response => response.json())
    .then(data => {
      totalSpent = 0;
      expenseList.innerHTML = ""; //Clear old list

      data.forEach(expense => {
        let expenseItem = document.createElement("div");
        expenseItem.classList.add("expense-item");
        expenseItem.innerHTML = `
          <span>${expense.name} (${expense.category}) - $${expense.amount.toFixed(2)}</span>
          <button class="delete-expense" data-id="${expense._id}">Delete</button>
        `;
        expenseList.appendChild(expenseItem);
        totalSpent += expense.amount;
      });

      totalSpentDisplay.textContent = `Total: $${totalSpent.toFixed(2)}`;
      updateChart(data);
    })
}
```

JavaScript

3. Chart

```
//Update pie chart
function updateChart(expenses) { //(printList + delete expense)
  let categoryTotals = { Food: 0, Entertainment: 0, Education: 0, Other: 0 };

  expenses.forEach(expense => {
    if (categoryTotals.hasOwnProperty(expense.category)) {
      categoryTotals[expense.category] += expense.amount;
    }
  });

  expenseChart.data.datasets[0].data = [
    categoryTotals.Food,
    categoryTotals.Entertainment,
    categoryTotals.Education,
    categoryTotals.Other,
  ];

  expenseChart.update();
}
```

```
//Pie chart
let category = ["Food", "Entertainment", "Education", "Other"];
let pieColor = ["#FF6384", "#36A2EB", "#FFCE56", "#4BC0C0"];
const myChart = document.getElementById("expense-chart").getContext("2d");
const expenseChart = new Chart(myChart, {
  type: "pie",
  data: {
    labels: category,
    datasets: [
      {
        data: [0, 0, 0, 0],
        backgroundColor: pieColor,
      },
    ],
  },
  options: {
    responsive: true,
    plugins: {
      legend: {
        position: "top",
      },
      title: {
        display: true,
        text: "Expense chart.",
      },
      tooltip: {
        callbacks: {
          label: function (context) {
            //Calc the total amount
            const total = context.chart.data.datasets[0].data.reduce((sum, value) => sum + value, 0);
            //Get the current value
            const currentValue = context.raw;
            //%
            const percentage = ((currentValue / total) * 100).toFixed(2);
            return `${context.label}: ${percentage}%`;
          }
```

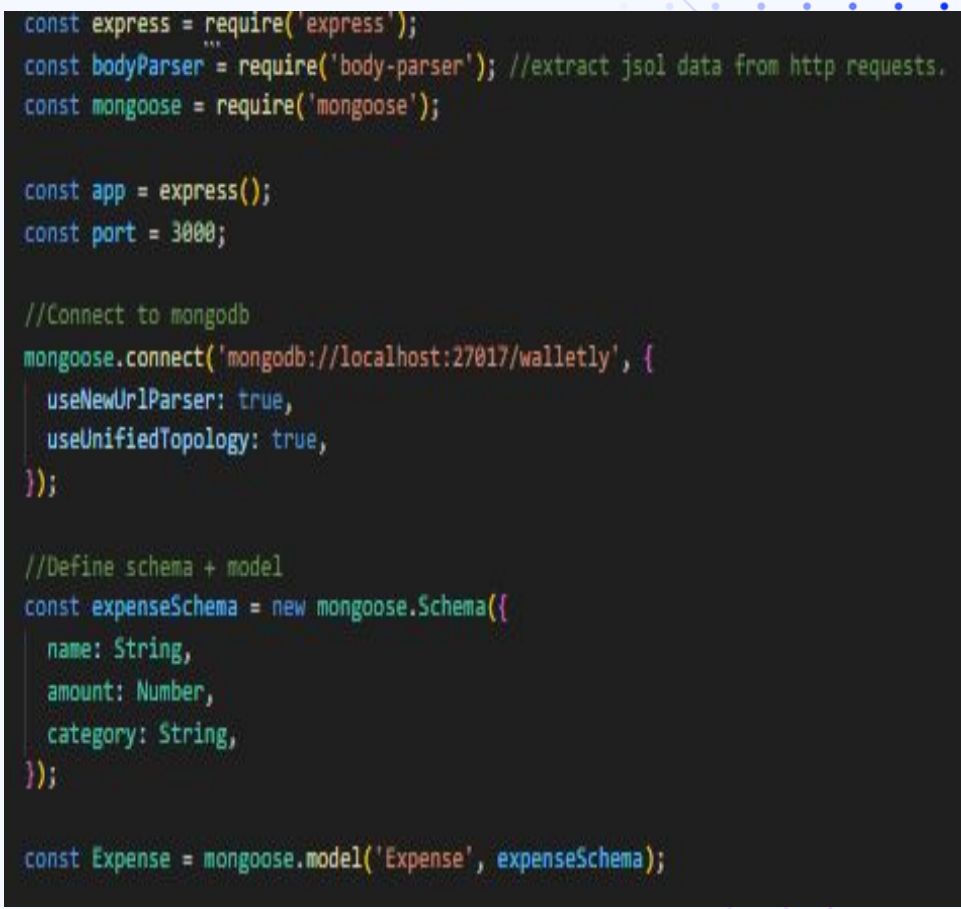
JavaScript

4. Delete an expense

```
//Delete an expense
expenseList.addEventListener("click", function (event) {
  if (event.target.classList.contains("delete-expense")) {
    const id = event.target.getAttribute("data-id");

    fetch(`http://localhost:3000/delete-expense/${id}`, { method: 'DELETE' })
      .then(response => response.json())
      .then(data => {
        if (data.message === 'Expense deleted successfully') {
          printList(); //Refresh the list after deletion
        } else {
          alert('Error deleting expense!');
        }
      })
  }
});
```

Server side

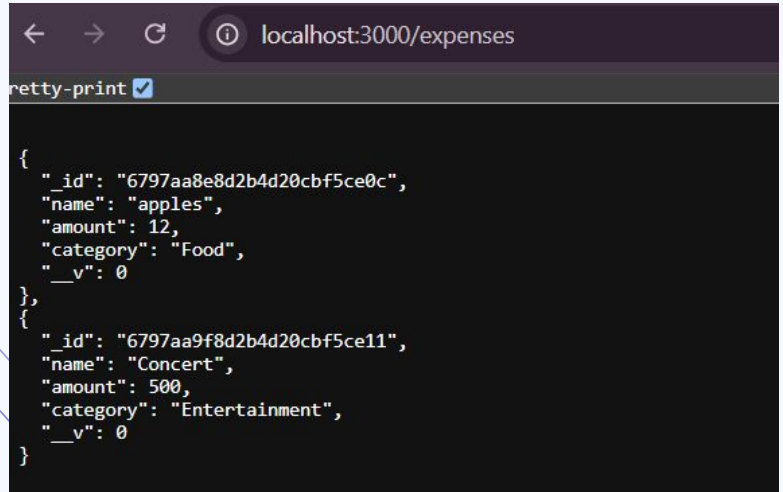
- Node.JS, Express, and Mongoose
- Utilizes Mongo database
  - Stores data
- Simple Mongoose Schema
  - Name
  - Amount
  - Category
- Status
  - 200 - valid requests
  - 400 - errors



```
const express = require('express');
const bodyParser = require('body-parser'); //extract jsol data from http requests.
const mongoose = require('mongoose');

const app = express();
const port = 3000;

//Connect to mongodb
mongoose.connect('mongodb://localhost:27017/walletly', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

//Define schema + model
const expenseSchema = new mongoose.Schema({
  name: String,
  amount: Number,
  category: String,
});

const Expense = mongoose.model('Expense', expenseSchema);
```

Server side

- Uses Try and catch blocks for requests and throwing errors
- Can add and remove items from database
- Add - POST (uses request and response)
- Expenses - GET
- Delete - DELETE




```
//Main Endpoint for API
app.get('/', (req, res) => {
  res.send('Welcome to Walletly API. Use /expenses, /add-expense, or /delete-expense/:id');
});

// Endpoint to add expense
app.post('/add-expense', async (req, res) => {
  try {
    const { name, amount, category } = req.body;
    const newExpense = new Expense({ name, amount, category });
    await newExpense.save();
    res.status(200).json({ message: 'Expense added successfully' });
  } catch (error) {
    res.status(400).json({ message: 'Error adding expense' });
  }
});

//Endpoint to get expense list
app.get('/expenses', async (req, res) => {
  try {
    const expenses = await Expense.find();
    res.status(200).json(expenses);
  } catch (error) {
    res.status(400).json({ message: 'Error fetching expenses' });
  }
});

//Endpoint to delete expense
app.delete('/delete-expense/:id', async (req, res) => {
  try {
    await Expense.findByIdAndDelete(req.params.id);
    res.status(200).json({ message: 'Expense deleted successfully' });
  } catch (error) {
    res.status(400).json({ message: 'Error deleting expense' });
  }
});

//Start server
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
```