FRONT END:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Manager</title>
</head>
<style>
body {
  font-family: Arial, sans-serif;
}

#task-form {
  margin-bottom: 20px;
}

input[type="text"], textarea, input[type="date"], select {
  margin-bottom: 10px;
  width: 70%;
}

button {
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  border: none;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

</style>
<body>
    <h1>Task Manager</h1>
    <div id="task-form">
```

```
    <input type="text" id="title" placeholder="Title">
    <textarea id="description" placeholder="Description"></textarea>
    <input type="date" id="dueDate">
    <select id="priority">
        <option value="low">Low</option>
        <option value="medium">Medium</option>
        <option value="high">High</option>
    </select><br>
    <button onclick="addTask()">Add Task</button>
</div>
<div id="task-list"> </div>
  <script>
    // script.js
        document.addEventListener('DOMContentLoaded', () => {
            getTasks();
        });

        async function getTasks() {
            const response = await fetch('/api/tasks');
            const tasks = await response.json();

            const taskList = document.getElementById('task-list');
            taskList.innerHTML = '';

            tasks.forEach(task => {
                const taskItem = document.createElement('div');
                taskItem.innerHTML = `
  <h3>${task.title}</h3>
  <p>${task.description}</p>
  <p>Due Date: ${task.dueDate ? new Date(task.dueDate).toLocaleDateString() : 'N/A'}</p>
  <p>Priority: ${task.priority}</p>
  <button onclick="deleteTask('${task._id}')">Delete</button>
`;
                taskList.appendChild(taskItem);
            });
        }

        async function addTask() {
            const title = document.getElementById('title').value;
            const description = document.getElementById('description').value;
```

```
            const dueDate = document.getElementById('dueDate').value;
            const priority = document.getElementById('priority').value;

            const response = await fetch('/api/tasks', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ title, description, dueDate, priority })
            });

            if (response.ok) {
                getTasks();
            } else {
                alert('Failed to add task');
            }
        }

        async function deleteTask(id) {
            const response = await fetch(`/api/tasks/${id}`, {
                method: 'DELETE'
            });

            if (response.ok) {
                getTasks();
            } else {
                alert('Failed to delete task');
            }
        }
    </script>
</body>

</html>
```

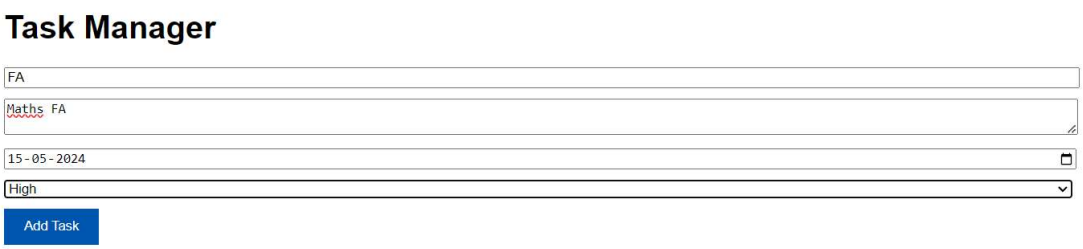
BACKEND

```
// Import required modules
const express = require('express');
const mongoose = require('mongoose');

// Initialize Express app
const app = express();
const PORT = process.env.PORT || 3000;

// Connect to MongoDB database
mongoose.connect('mongodb://localhost/taskmanager', { useNewUrlParser: true,
useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.error('MongoDB connection error:', err));

// Define Task schema
const TaskSchema = new mongoose.Schema({
  title: String,
  description: String,
  dueDate: Date,
  priority: { type: String, enum: ['low', 'medium', 'high'] }
});

// Create Task model
const Task = mongoose.model('Task', TaskSchema);

// Middleware to parse JSON bodies
app.use(express.json());

// API endpoints for CRUD operations on tasks
// Create a task
app.post('/api/tasks', async (req, res) => {
  try {
    const task = new Task(req.body);
    await task.save();
    res.status(201).json(task);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});
```

```
// Get all tasks
app.get('/api/tasks', async (req, res) => {
  try {
    const tasks = await Task.find();
    res.json(tasks);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

// Get a single task
app.get('/api/tasks/:id', async (req, res) => {
  try {
    const task = await Task.findById(req.params.id);
    if (!task) throw new Error('Task not found');
    res.json(task);
  } catch (err) {
    res.status(404).json({ message: err.message });
  }
});

// Update a task
app.patch('/api/tasks/:id', async (req, res) => {
  try {
    const task = await Task.findById(req.params.id);
    if (!task) throw new Error('Task not found');

    Object.assign(task, req.body);
    await task.save();

    res.json(task);
  } catch (err) {
    res.status(404).json({ message: err.message });
  }
});

// Delete a task
app.delete('/api/tasks/:id', async (req, res) => {
  try {
```

```
    const task = await Task.findById(req.params.id);
    if (!task) throw new Error('Task not found');

    await task.remove();
    res.json({ message: 'Task deleted' });
  } catch (err) {
    res.status(404).json({ message: err.message });
  }
});

// Start the server
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

OUTPUT:



BACKEND:

Create your backend server (server.js):

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

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

// Create Product model
const Product = mongoose.model('Product', {
  name: String,
  price: Number,
});

app.use(bodyParser.json());

// Routes
app.get('/products', async (req, res) => {
  try {
    const products = await Product.find();
    res.json(products);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

app.post('/products', async (req, res) => {
  const { name, price } = req.body;
  const product = new Product({ name, price });
  try {
    const newProduct = await product.save();
```

```
    res.status(201).json(newProduct);
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
});

// Start the server
app.listen(PORT, () => {
  console.log(Server is running on http://localhost:${PORT});
});
```

FRONT END:

<!-- index.html -->

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Shopping</title>
    <style>
        /* styles.css */
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 0;
        }

        h1 {
            text-align: center;
            color: #333;
            margin-top: 20px;
        }

        #add-product-form {
            text-align: center;
            margin-top: 20px;
        }
```

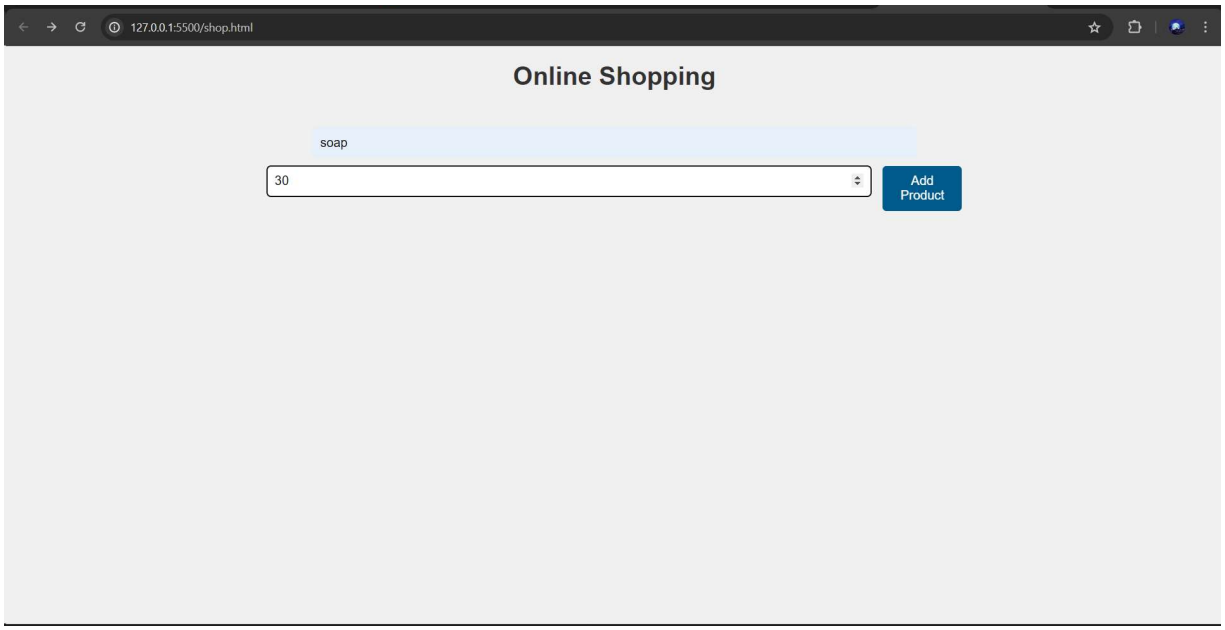
```
#product-list {
    margin-top: 20px;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}

.product {
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    padding: 20px;
    margin: 20px;
    width: 250px;
    text-align: center;
    transition: transform 0.3s ease;
}

.product:hover {
    transform: translateY(-5px);
}

.product-name {
    font-weight: bold;
    color: #333;
    font-size: 1.2em;
}

.product-price {
    color: #4CAF50;
    font-size: 1.2em;
}

.buy-button {
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
    padding: 10px 20px;
    font-size: 1em;
```

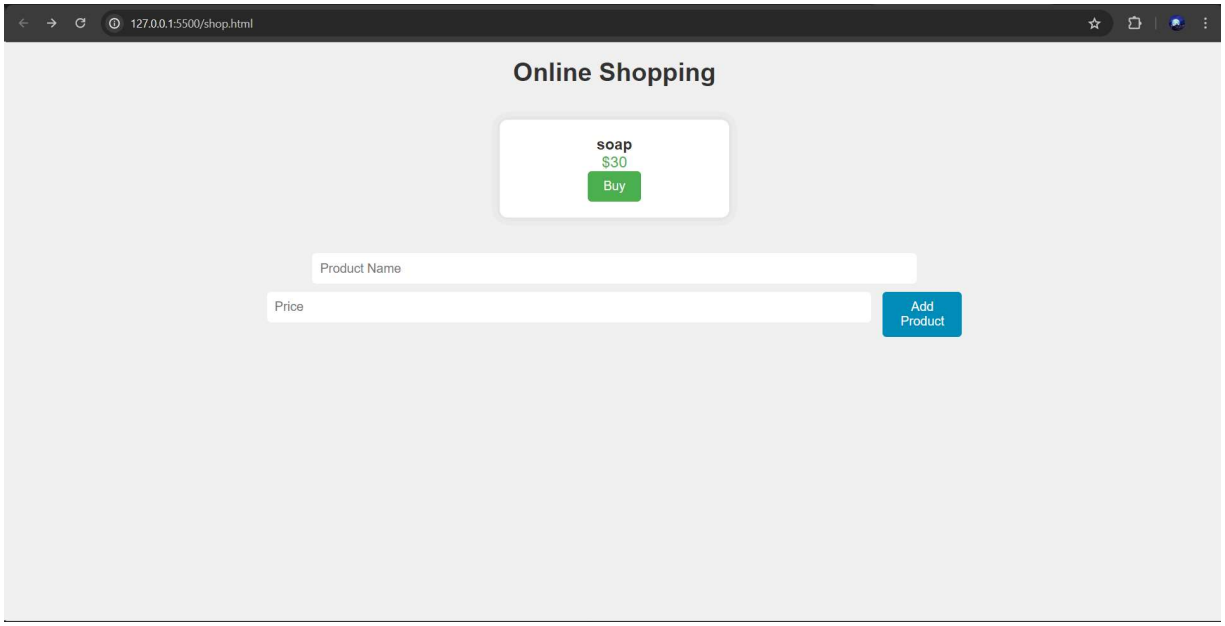
```
            cursor: pointer;
            transition: background-color 0.3s ease;
        }

        .buy-button:hover {
            background-color: #45a049;
        }

        #add-product-form input[type="text"],
        #add-product-form input[type="number"],
        #add-product-form button {
            padding: 10px;
            margin: 5px;
            border: none;
            border-radius: 5px;
            font-size: 1em;
        }

        #add-product-form input[type="text"],
        #add-product-form input[type="number"] {
            width: calc(50% - 25px);
        }

        #add-product-form button {
            background-color: #008CBA;
            color: white;
            cursor: pointer;
            width: 100px;
            transition: background-color 0.3s ease;
        }

        #add-product-form button:hover {
            background-color: #005A8D;
        }
    </style>

</head>

<body>
    <h1>Online Shopping</h1>
```

```
    <div id="product-list"></div>
    <form id="add-product-form">
        <input type="text" id="name" placeholder="Product Name">
        <input type="number" id="price" placeholder="Price">
        <button type="submit">Add Product</button>
    </form>
    <script>
            const productList = document.getElementById('product-list');
            const addProductForm = document.getElementById('add-product-form');

            // Fetch products
            async function fetchProducts() {
                try {
                    const response = await fetch('/products');
                    const products = await response.json();
                    productList.innerHTML = products.map(product => <div>${product.name}:
$${product.price}</div>).join('');
                } catch (error) {
                    console.error('Error fetching products:', error);
                }
            }

            // Add product
            addProductForm.addEventListener('submit', async (event) => {
                event.preventDefault();
                const name = document.getElementById('name').value;
                const price = document.getElementById('price').value;
                try {
                    const response = await fetch('/products', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json',
                        },
                        body: JSON.stringify({ name, price }),
                    });
                    if (response.ok) {
                        fetchProducts();
                        addProductForm.reset();
                    } else {
                        const errorMessage = await response.text();
```

```
                console.error('Error adding product:', errorMessage);
            }
        } catch (error) {
            console.error('Error adding product:', error);
        }
    });

    // Initial fetch
    fetchProducts(); const productList = document.getElementById('product-list');
const addProductForm = document.getElementById('add-product-form');

// Function to create a new product div
function createProductDiv(name, price) {
    const productDiv = document.createElement('div');
    productDiv.classList.add('product');
    productDiv.innerHTML = `
        <div class="product-name">${name}</div>
        <div class="product-price">$${price}</div>
        <button class="buy-button">Buy</button>
    `;
    return productDiv;
}

// Add product
addProductForm.addEventListener('submit', async (event) => {
    event.preventDefault();
    const name = document.getElementById('name').value;
    const price = document.getElementById('price').value;

    // Create and append the new product div
    const newProductDiv = createProductDiv(name, price);
    productList.appendChild(newProductDiv);

    // Reset form
    addProductForm.reset();
});
  </script>
</body>

</html>
```

OUTPUT:


127.0.0.1:5500/shop.html
Online Shopping
soap
30
Add Product


127.0.0.1:5500/shop.html
Online Shopping
soap
$30
Buy
Product Name
Price
Add Product