

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №4**

**з дисципліни «Інтернет-технології»**

**на тему:**

**«**Розробка бекенд частини»

Виконав:

студент гр. ПЗ2011

Кулик С. В.

Прийняв:

Андрющенко В. О.

Дніпро, 2023

**Тема.** Розробка бекенд частини.

**Завдання**

Додати до сайту, створеного в лабораторній роботі №2 JavaScript бекенд функціонал.

**HTML код сторінки**

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Додати нову наукову роботу</title>

    <link rel="stylesheet" href="styles.css">

    <script src="functional.js"></script>

</head>

<body>

    <header>

        <a href="index.html">

            <h1>Класифікація зображень за допомогою нейронних мереж</h1>

        </a>

    </header>

    <section>

        <h2>Існуючі моделі</h2>

        <ul>

            <li>

                <h3>ResNet</h3>

                <p>ResNet (Residual Network) - модель з 23.5M параметрів та 4.09B операцій.</p>

                <p>Використовується в різних галузях, включаючи комп'ютерне зорове розпізнавання та медицину.</p>

            </li>

            <li>

                <h3>Inception</h3>

                <p>Inception (GoogLeNet) - модель з 22.8M параметрів та 5.74B операцій.</p>

                <p>Широко використовується в галузі комп'ютерного зору та розпізнавання об'єктів.</p>

            </li>

            <li>

                <h3>MobileNet</h3>

                <p>MobileNet - легка модель з 4.2M параметрів та 569M операцій.</p>

                <p>Ідеальна для використання на мобільних пристроях та вбудованих системах.</p>

            </li>

            <li>

                <h3>VGG</h3>

                <p>VGG (Visual Geometry Group) - модель з 138M параметрів та 15.5B операцій.</p>

                <p>Використовується для обробки зображень в різних застосуваннях.</p>

            </li>

            <li>

                <h3>AlexNet</h3>

                <p>AlexNet - модель з 61M параметрів та 725M операцій.</p>

                <p>Одна з перших успішних глибоких нейронних мереж для класифікації зображень.</p>

            </li>

        </ul>

    </section>

    <section id="add-research">

        <h2>Додати нову наукову роботу</h2>

        <form id="add-research-form">

            <label for="model-name">Назва моделі:</label>

            <input type="text" id="model-name" name="model-name" required>

            <label for="model-author">Автор:</label>

            <input type="text" id="model-author" name="model-author" required>

            <label for="author-description">Короткий опис автора:</label>

            <textarea id="author-description" name="author-description" rows="4" required></textarea>

            <label for="author-contact">Контактні дані автора:</label>

            <input type="text" id="author-contact" name="author-contact" required>

            <label for="model-description">Короткий опис моделі:</label>

            <textarea id="model-description" name="model-description" rows="4" required></textarea>

            <label for="model-parameters">Кількість параметрів:</label>

            <input type="text" id="model-parameters" name="model-parameters" required>

            <label for="model-operations">Кількість операцій (приблизно):</label>

            <input type="text" id="model-operations" name="model-operations" required>

            <label for="model-file">Файл з дослідженням:</label>

            <label for="model-file" class="custom-file-input">

                <span>Завантажте файл</span>

                <input type="file" id="model-file" name="model-file" accept=".pdf, .doc, .docx">

            </label>

            <button type="button" onclick="submitResearch()" class="button-submit">Додати роботу</button>

        </form>

        <div id="status-message"></div>

    </section>

    <section>

        <h2>Моделі в базі даних</h2>

        <table id="models-table">

            <thead>

                <tr>

                    <th>Назва моделі</th>

                    <th>Автор</th>

                    <th>Опис наукової роботи</th>

                    <th>Контакти</th>

                    <th>Параметри</th>

                    <th>Кількість операцій</th>

                    <th>Файл</th>

                </tr>

            </thead>

            <tbody></tbody>

        </table>

    </section>

    <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

    <script>

      $(document).ready(() => {

        // Function to fetch existing models asynchronously

        const fetchModels = () => {

          $.ajax({

            type: 'GET',

            url: '/get-models',

            success: (data) => {

              // Update the models table with the retrieved data

              const modelsTable = $('#models-table tbody');

              modelsTable.empty();

              data.forEach((model) => {

                const row = `<tr>

                  <td>${model.modelName}</td>

                  <td>${model.modelAuthor}</td>

                  <td>${model.modelDescription}</td>

                  <td>${model.authorContact}</td>

                  <td>${model.modelParameters}</td>

                  <td>${model.modelOperations}</td>

                  <td><a href="/path/to/download/${model.id}">Download</a></td>

                </tr>`;

                modelsTable.append(row);

              });

            },

            error: (error) => {

              console.error('Error fetching models:', error);

            },

          });

        };

        // Fetch models on page load

        fetchModels();

        // Example: Fetch models when a button is clicked

        $('#fetch-models-btn').click(() => {

          fetchModels();

        });

      });

    </script>

    <section>

        <footer>

            <p>&copy; 2023 Класифікація за допомогою нейронних мереж</p>

        </footer>

    </section>

    <script src="my-backend-project/app.js"></script>

</body>

</html>

**Текст CSS файлу сайту**

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background-color: #f0f0f0;

}

header {

    background-color: #333;

    color: #fff;

    text-align: center;

    padding: 20px;

}

h1 {

    margin: 0;

}

section {

    margin: 20px;

    background-color: #fff;

    padding: 20px;

    border: 1px solid #ddd;

    border-radius: 5px;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

h2 {

    font-size: 24px;

    margin-top: 0;

}

ul.no-bullets {

    list-style-type: none;

    padding-left: 0;

}

ul li {

    margin-bottom: 10px;

}

table {

    width: 100%;

    border-collapse: collapse;

    margin-top: 10px;

}

th, td {

    border: 1px solid #ddd;

    padding: 8px;

    text-align: left;

}

th {

    background-color: #f2f2f2;

}

footer {

    text-align: center;

    padding: 10px;

    background-color: #333;

    color: #fff;

}

.no-bullets {

    list-style-type: none;

    padding-left: 20px; /\* Додайте відступ зліва від краю сторінки \*/

}

.no-bullets li {

    text-align: center;

}

.no-bullets li img {

    display: block;

    margin: 0 auto;

}

/\* Центрування контенту \*/

.center {

    text-align: center;

}

/\* Стилізація кнопок "Наступний" і "Попередній" \*/

.nav-button {

    background-color: transparent;

    border: none;

    font-size: 24px;

    cursor: pointer;

}

/\* Стилізація контейнера для кнопок \*/

.nav-button-container {

    display: flex;

    justify-content: space-between;

    align-items: center;

    margin-top: 20px; /\* Встановіть потрібний відступ від інформації про датасет \*/

}

.dataset-tile {

    background-color: #fff;

    padding: 20px;

    border: 1px solid #ddd;

    border-radius: 5px;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

    text-align: center;

}

/\* Стилізація зображення внутрішньо блоку \*/

.dataset-tile img {

    display: block;

    margin: 0 auto;

    width: 150px; /\* Задайте потрібну ширину \*/

    height: auto; /\* Якщо необхідно підтримувати співвідношення сторін \*/

}

/\* Стилізація заголовка та опису \*/

.dataset-tile h3 {

    font-size: 24px;

    margin-top: 0;

}

section h2 {

    font-size: 24px;

    color: #333;

}

#add-model-form {

    background-color: #f4f4f4;

    border: 1px solid #ccc;

    padding: 20px;

    border-radius: 5px;

    margin-top: 10px;

}

#add-model-form label {

    font-weight: bold;

}

#add-model-form input[type="text"] {

    width: 100%;

    padding: 10px;

    margin-bottom: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

}

#add-model-form button {

    background-color: #333;

    color: #fff;

    border: none;

    padding: 10px 20px;

    border-radius: 5px;

    cursor: pointer;

}

#add-model-form button:hover {

    background-color: #555;

}

**Текст JavaScript файлу сайту**

const \_express = require('express');

const bodyParser = require('body-parser');

const sqlite3 = require('sqlite3').verbose();

const app = \_express();

const PORT = 3000;

const validateEmail = (email) => {

  // Regular expression for a simple email validation

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

  return emailRegex.test(email);

};

const validateName = (name) => {

  // Regular expression for a simple name validation (allowing letters, spaces, and hyphens)

  const nameRegex = /^[a-zA-Z\s-]+$/;

  return nameRegex.test(name);

};

const validatePhone = (phone) => {

  // Regular expression for a simple phone number validation (allowing digits, spaces, and hyphens)

  const phoneRegex = /^[0-9\s-]+$/;

  return phoneRegex.test(phone);

};

// Create SQLite database and table if not exists

const db = new sqlite3.Database('research.db');

db.run(

  'CREATE TABLE IF NOT EXISTS research (id INTEGER PRIMARY KEY AUTOINCREMENT, modelName TEXT, modelAuthor TEXT, authorDescription TEXT, authorContact TEXT, modelDescription TEXT, modelParameters TEXT, modelOperations TEXT)',

  (err) => {

    if (err) {

      console.error('Database error:', err);

    } else {

      console.log('Database is ready');

    }

  }

);

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

// Endpoint for handling form submissions

app.post('/submit-research', (req, res) => {

  const {

    'model-name': modelName,

    'model-author': modelAuthor,

    'author-description': authorDescription,

    'author-contact': authorContact,

    'model-description': modelDescription,

    'model-parameters': modelParameters,

    'model-operations': modelOperations,

  } = req.body;

  // Perform data validation here

  if (

    !modelName ||

    !modelAuthor ||

    !authorDescription ||

    !authorContact ||

    !modelDescription ||

    !modelParameters ||

    !modelOperations

  ) {

    return res.status(400).json({ error: 'All fields are required.' });

  }

  // Validate email format

  if (!(validateEmail(authorContact) || validatePhone(authorContact))) {

    return res.status(400).json({ error: 'Invalid email format for author contact.' });

  }

  // Validate name format

  if (!validateName(modelAuthor)) {

    return res.status(400).json({ error: 'Invalid name format for the author.' });

  }

  // Save the form data to the SQLite database

  const query =

    'INSERT INTO research (modelName, modelAuthor, authorDescription, authorContact, modelDescription, modelParameters, modelOperations) VALUES (?, ?, ?, ?, ?, ?, ?)';

  db.run(

    query,

    [modelName, modelAuthor, authorDescription, authorContact, modelDescription, modelParameters, modelOperations],

    function (err) {

      if (err) {

        console.error('Database insertion error:', err);

        return res.status(500).json({ error: 'Internal Server Error' });

      }

      res.status(200).json({ message: 'Research added successfully.', insertedId: this.lastID });

    }

  );

});

app.get('/get-models', (req, res) => {

  const query = 'SELECT \* FROM research';

  db.all(query, (err, rows) => {

    if (err) {

      console.error('Database query error:', err);

      return res.status(500).json({ error: 'Internal Server Error' });

    }

    res.status(200).json(rows);

  });

});

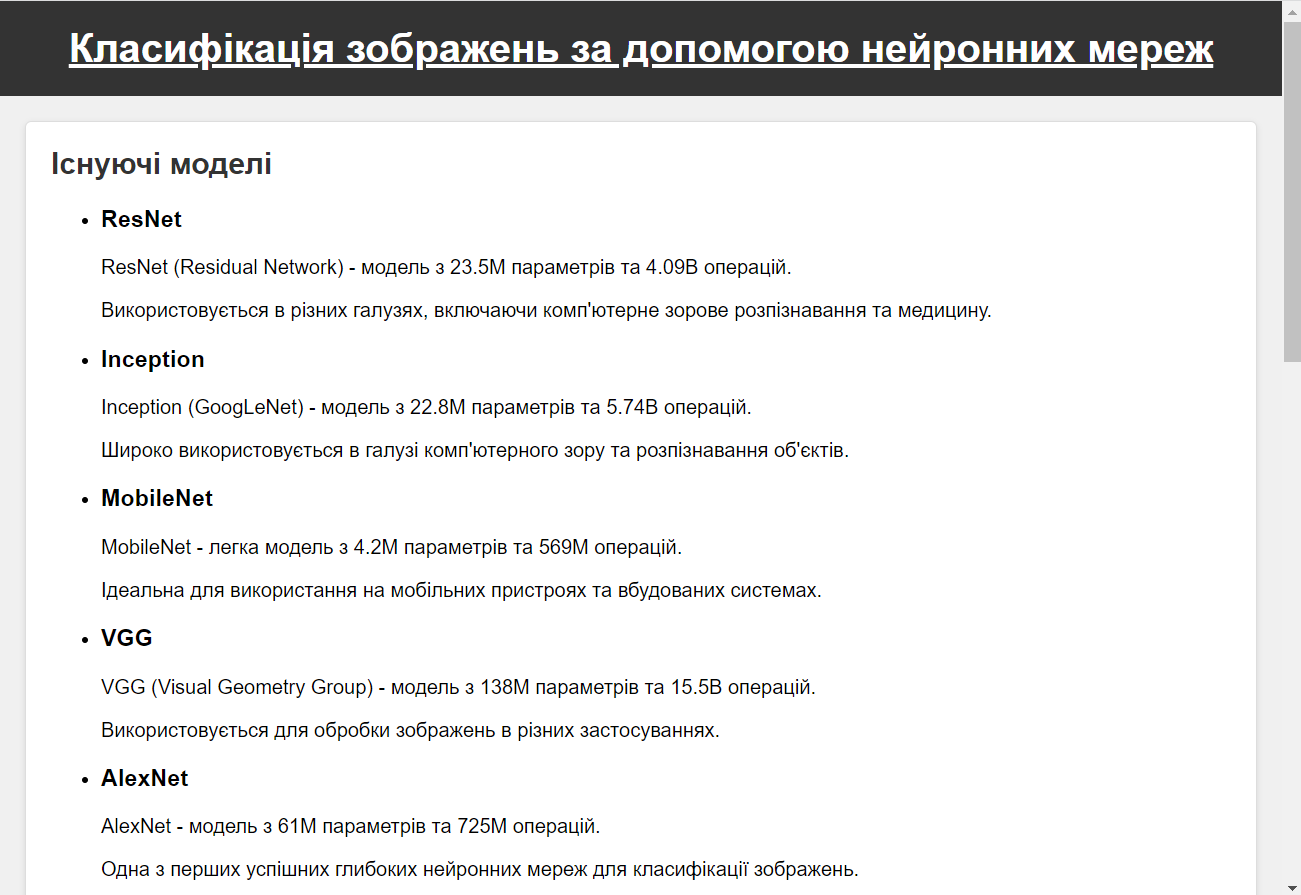
// Start the server

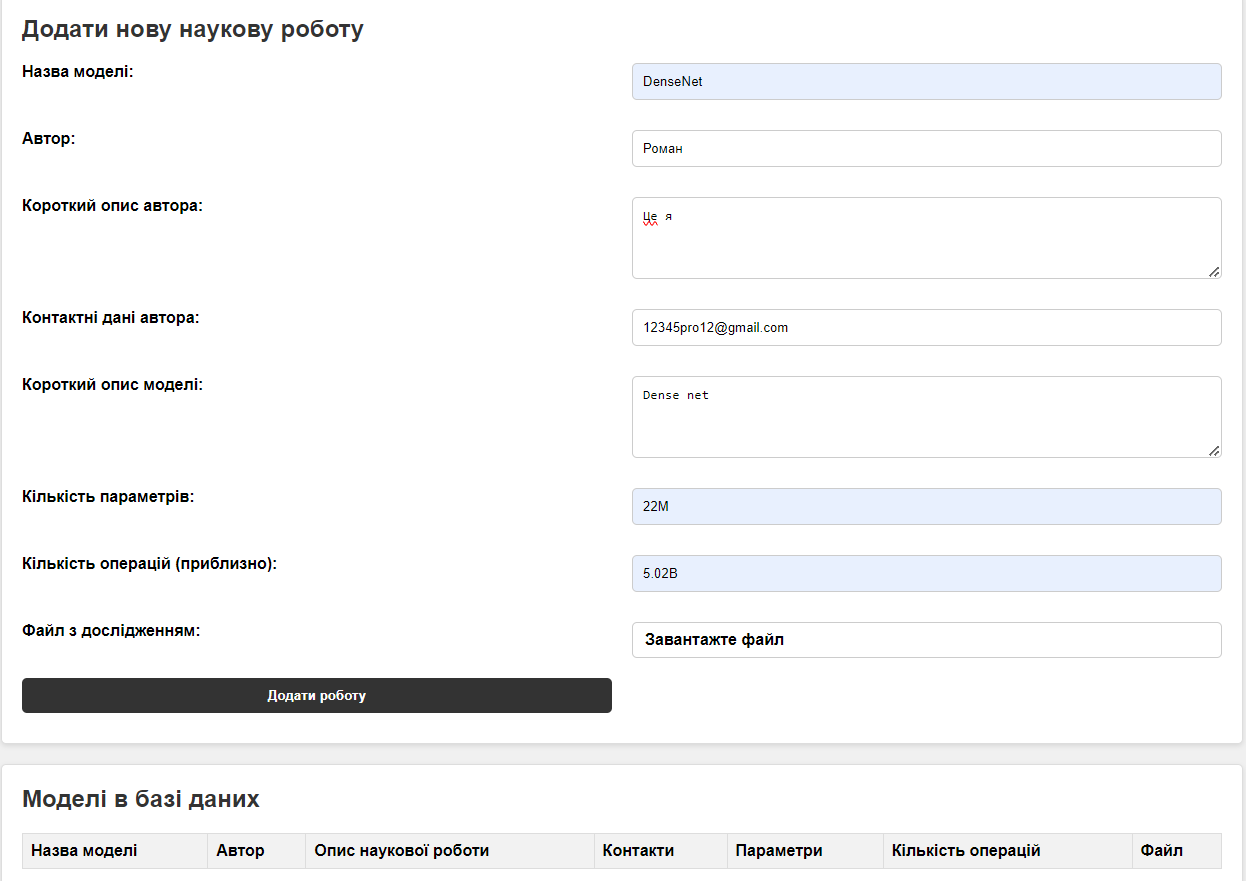
app.listen(PORT, () => {

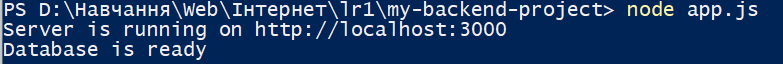
  console.log(`Server is running on http://localhost:${PORT}`);

});

**Демонстрація сайту**





**Висновки**

Розробка бекенду на Node.js є захоплюючим та дуже популярним аспектом веб-розробки. Особливу увагу слід звернути на асинхронну природу Node.js та його події, які дозволяють ефективно обробляти багато операцій. Зазвичай для створення серверів та обробки запитів використовується фреймворк Express.js. Добре організована маршрутизація та використання middleware дозволяють реалізувати додаткову логіку перед або після обробки запитів.

Взаємодія з базами даних здійснюється за допомогою різних бібліотек, таких як SQLite, MySQL або MongoDB. Важливо валідувати введені дані та ефективно обробляти помилки, щоб забезпечити надійність та безпеку додатка. Безпека є ключовим аспектом, тому слід уникати атак та забезпечувати використання HTTPS для захисту даних.