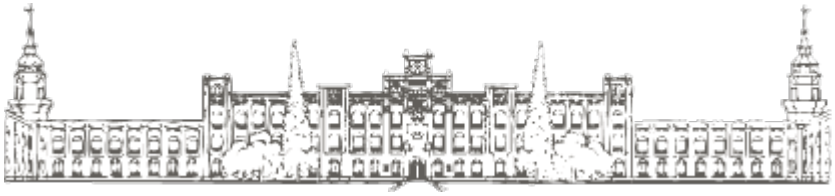

Міністерство освіти і науки України

Національний технічний університет України „КПІ

імені Ігоря Сікорського ”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики і програмної інженерії

ЗВІТ

лабораторної роботи № 4

з курсу «Основи WEB - технологій»

Тема: «Nodejs. Робота з БД. Додаток що реалізує CRUD операції в БД.»

Перевірив:

Викл. Альбрехт Й.О.

Виконав:

Студент ІП-15
Мєшков А.І

Київ 2024

Завдання

Створити додаток, що реалізує CRUD операції з БД - додавання, читання, редагування та видалення записів БД (Додаток 1. Таблиця 1).

Забезпечити роутінг запитів та виведення результатів запитів на WEB- сторінку.

Додати нові роути для виведення інформації у вигляді json-файлу (API).

Варіант 7

Варіант 7. Спроектувати базу даних про оцінки, отриманих студентами на іспитах: прізвище, група, предмет, номер квитка, оцінка, викладач.

Хід роботи

Було програму nodejs з використанням Express, EJS, MongoDB.

app.js

```
require('dotenv').config();
const express = require('express');
const connectDB = require('./utils/db');
const gradeRoutes = require('./routes/routes');

const app = express();

app.use(express.json());
app.set("view engine", "ejs")
app.use(express.static('public'));

connectDB();

app.use('/', gradeRoutes);


const PORT = process.env.PORT || 3000;
app.listen(PORT, (error) => {
  error ? console.log(error) : console.log(`Server is running on port ${PORT}`);
});
```

Db.js

```
const mongoose = require('mongoose');
require('dotenv').config();

const dbUrl = process.env.DB_URL || '';

const connectDB = async () => {
    try {
        const data = await mongoose.connect(dbUrl);
        console.log(`Database connected with ${data.connection.host}`);
    } catch (error) {
        console.log(error.message);
        setTimeout(connectDB, 5000);
    }
};


module.exports = connectDB;
```

Grade.js

```
const mongoose = require('mongoose');
```

```
const gradeSchema = new mongoose.Schema({
  lastName: { type: String, required: true},
  group: { type: String, required: true},
  subject: { type: String, required: true},
  ticketNumber: { type: Number, required: true},
  grade: { type: Number, required: true},
  professor: { type: String, required: true},
}, {timestamps: true});

module.exports = mongoose.model('Grade', gradeSchema);
```

routes.js

```
const express = require('express');
const router = express.Router();
const {getGrades, deleteGrade, getEditGrade, editGrade, getAddGrade, addGrade, getGradeJSON,
getGradesJSON} = require('../controllers/controllers');

router.get('/', (req, res) => {
    res.render('main');
});

router.get('/grades', getGrades);
router.get('/grades/json', getGradesJSON);
router.get('/grades/:id/json', getGradeJSON);

router.delete('/grades/:id', deleteGrade);
router.get('/edit/:id', getEditGrade);
router.put('/edit/:id', editGrade);

router.get('/add-grade', getAddGrade);
router.post('/add-grade', addGrade);

module.exports = router;
```

controllers.js

```
const Grade = require('../models/Grade');

const getGrades = (req,res) => {
    Grade
        .find()
        .then(grades => res.render('grades', { grades }))
        .catch((error) => handleError(res,error));
}

const deleteGrade = (req,res) => {
    Grade
        .findByIdAndDelete(req.params.id)
        .then(result => res.sendStatus(200))
        .catch((error) => handleError(res,error));
}
```

```
const getEditGrade = (req,res) => {
    Grade
        .findById(req.params.id)
        .then(grade => res.render('edit', { grade }))
        .catch((error) => handleError(res,error));
}

const editGrade = (req,res) => {
    Grade
        .findByIdAndUpdate(req.params.id, req.body)
        .then(result => res.redirect(303,'/grades'))
        .catch((error) => handleError(res,error));
}

const getAddGrade = (req,res) => {
    res.render('add-grade');
}

const addGrade = (req,res) => {
    const {lastName, group, subject, ticketNumber, grade, professor} = req.body;
    const newGrade = new Grade({lastName, group, subject, ticketNumber, grade, professor});
    newGrade
        .save()
        .then(result => res.redirect('/grades'))
        .catch((error) => handleError(res,error));
}

const getGradesJSON = (req, res) => {
    Grade
        .find()
        .then(grades => res.json(grades))
        .catch((error) => handleError(res,error));
};

const getGradeJSON = (req, res) => {
    Grade
        .findById(req.params.id)
        .then(grades => res.json(grades))
        .catch((error) => handleError(res,error));

};


module.exports = {getGrades, deleteGrade, getEditGrade, editGrade, getAddGrade, addGrade,
getGradesJSON, getGradeJSON};
```

main.ejs

```
<html>
  <head>
```

```
    <title>Student Grades</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
  </head>
  <body>
    <%- include('components/_header.ejs') %>
    <h1>Welcome to the Student Grades System</h1>
    <h2>Build with Node.js, Express, EJS, MongoDB</h2>
  </body>
</html>
```

Grades.ejs

```
<html>
  <head>
    <title>Student Grades</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
  </head>
  <body>
    <%- include('components/_header.ejs') %>
    <h2>All Grades</h2>
    <ul>
      <% grades.forEach((grade) => { %>
        <li onclick="window.location='/edit/<%= grade._id %>'">
          <h3><%= grade.lastName %></h3>
          <h4>Group: <%= grade.group %></h4>
          <h4>Subject: <%= grade.subject %></h4>
          <h4>Grade: <%= grade.grade %></h4>
          <h4>by <%= grade.professor %></h4>
          <button onclick="deleteGrade('<%= grade._id %>')">Delete</button>
        </li>
      <% }); %>
    </ul>


    <script>
      async function deleteGrade(id) {
        const confirmDelete = confirm("Are you sure you want to delete this grade?");
        if (!confirmDelete) return;

        try {
          const response = await fetch(`/grades/${id}`, {
            method: 'DELETE',
          });

          if (response.ok) {
            alert('Grade deleted successfully!');
            location.reload();
          } else {
            alert('Failed to delete grade.');
          }
        } catch (error) {
```

```
          console.error('Error deleting grade:', error);
          alert('An error occurred while deleting the grade.');
        }
      }
    </script>
  </body>
</html>
```

Add-grade.ejs

```
<html>
  <head>
    <title>Student Grades</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
  </head>
  <body>
    <%- include('components/_header.ejs') %>
    <h2>Add New Grade</h2>
    <form id="addGradeForm">
      <input type="text" name="lastName" placeholder="Last Name" required />
      <input type="text" name="group" placeholder="Group" required />
      <input type="text" name="subject" placeholder="Subject" required />
      <input type="number" name="ticketNumber" placeholder="Ticket Number" required />
      <input type="number" name="grade" placeholder="Grade" required />
      <input type="text" name="professor" placeholder="Professor" required />
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
<script>
  document.getElementById('addGradeForm').addEventListener('submit', async (event) => {
    event.preventDefault();

    const formData = {
      lastName: document.querySelector('input[name="lastName"]').value,
      group: document.querySelector('input[name="group"]').value,
      subject: document.querySelector('input[name="subject"]').value,
      ticketNumber: document.querySelector('input[name="ticketNumber"]').value,
      grade: document.querySelector('input[name="grade"]').value,
      professor: document.querySelector('input[name="professor"]').value,
    };

    try {

      const response = await fetch('/add-grade', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(formData),
      });
```

```
      if (response.ok) {
        alert('Grade added successfully!');

        document.getElementById('addGradeForm').reset();
      } else {
        alert('Failed to add grade. Please try again.');
      }
    } catch (error) {
      console.error('Error:', error);
      alert('An error occurred. Please try again.');
    }
  });
</script>
```

Edit.ejs

```
<html>
  <head>
    <title>Student Grades</title>
    <link rel="stylesheet" type="text/css" href="http://localhost:3000/css/styles.css">
  </head>
  <body>
    <%- include('components/_header.ejs') %>
    <h2>Edit Grade</h2>
    <form id="editGradeForm">
      <input type="text" name="lastName" value="<%= grade.lastName %>" required />
      <input type="text" name="group" value="<%= grade.group %>" required />
      <input type="text" name="subject" value="<%= grade.subject %>" required />
      <input type="number" name="ticketNumber" value="<%= grade.ticketNumber %>" required />
      <input type="number" name="grade" value="<%= grade.grade %>" required />
      <input type="text" name="professor" value="<%= grade.professor %>" required />
      <button type="submit">Update Grade</button>
    </form>
  </body>
</html>

<script>
  document.getElementById('editGradeForm').addEventListener('submit', async (event) => {
    event.preventDefault();

    const gradeId = "<%= grade._id %>";
    const formData = {
      lastName: document.querySelector('input[name="lastName"]').value,
      group: document.querySelector('input[name="group"]').value,
      subject: document.querySelector('input[name="subject"]').value,
      ticketNumber: document.querySelector('input[name="ticketNumber"]').value,
      grade: document.querySelector('input[name="grade"]').value,
      professor: document.querySelector('input[name="professor"]').value,
    };
```

```
    try {

      const response = await fetch(`/edit/${gradeId}`, {
        method: 'PUT',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(formData),
      });

    } catch (error) {
      console.error('Error:', error);
      alert('An error occurred. Please try again.');
    }
  });
</script>
```

_header.ejs

```
<header>
    <nav>
      <a href="/">Home</a>
      <a href="/grades">Posts</a>
      <a href="/add-grade">New Post</a>
    </nav>
</header>
```

1. Отримані результати

На рис 4.1-4.4 можна побачити сторінки сайту.

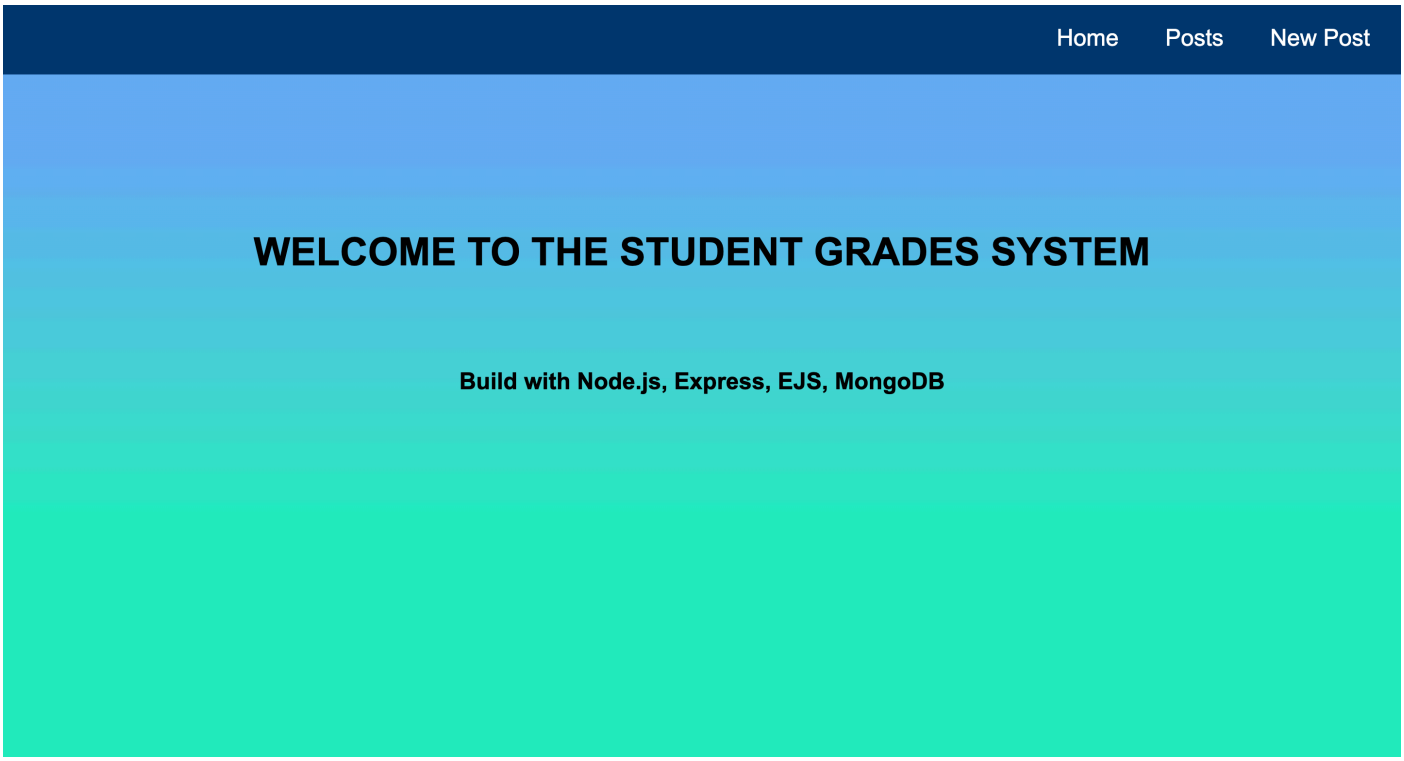

Рис. 4.1. Головна сторінка

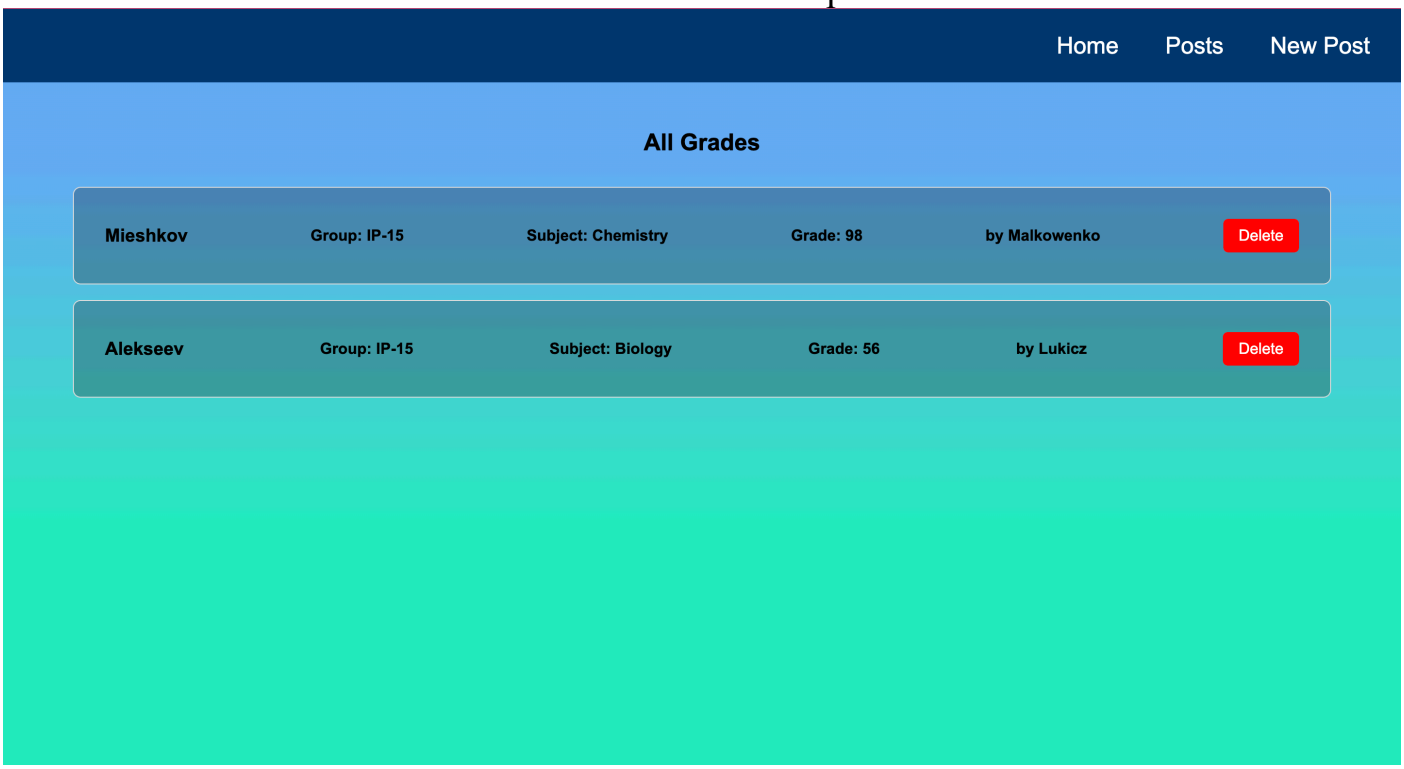

Рис. 4.2. Сторінка оцінок

Рис. 4.3. Сторінка додавання

Рис. 4.4. Сторінка оновлення

На рис 4.5 можна побачити базу даних.

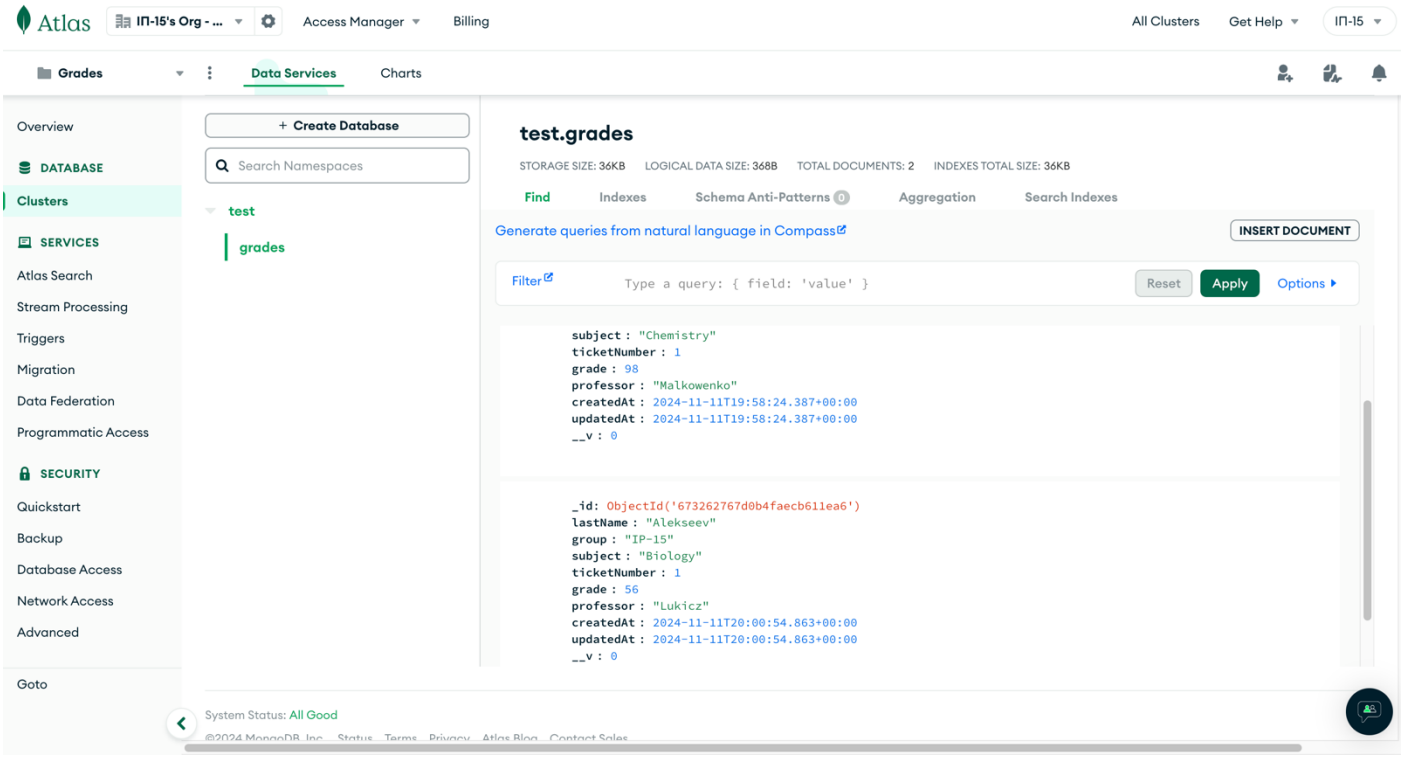

Рис. 4.5. База даних

ВИСНОВОК

Під час виконання даної лабораторної роботи створено додаток на основі Node.js, Express, MongoDB, що реалізує CRUD-операції для управління даними студентських оцінок. Використано маршрутизацію для обробки запитів до різних сторінок, де можна переглядати, додавати, редагувати та видаляти оцінки. Окрім того, додано маршрути для API, які повертають дані у форматі JSON, що дозволяє легко інтегрувати додаток з іншими сервісами або фронтендом.