Звіт про Веб-застосунок "Органайзер"

Метою цієї лабораторної роботи було створення веб-застосунку під назвою "Органайзер", призначеного для ефективного управління завданнями. Цей додаток розроблено з акцентом на забезпечення зручності для користувачів, безпеки даних та швидкодії. Він дозволяє користувачам створювати, редагувати, видаляти завдання, управляти їх пріоритетами, а також встановлювати нагадування. Рішення реалізоване з використанням сучасного технологічного стеку MERN, який складається з MongoDB, Express.js, React.js та Node.js. Завдяки цьому стеку додаток має розділену архітектуру, де фронтенд і бекенд працюють незалежно, але узгоджено через REST API.

Додаток "Органайзер" розроблений для того, щоб допомогти користувачам структурувати свої справи, забезпечуючи можливість ефективного управління часом. Основна функціональність включає створення завдань, призначення їм пріоритетів, фільтрацію за різними критеріями, а також нагадування про важливі події. Інтерфейс додатка побудований таким чином, щоб забезпечити інтуїтивність та доступність, включаючи адаптивність для різних типів пристроїв. Завдяки використанню React.js користувачі отримують швидкий і динамічний досвід роботи із застосунком, без необхідності перезавантаження сторінок.

Бекенд проекту побудований з використанням Node.js та Express.js. Серверна частина відповідає за обробку запитів, виконання логіки додатка, а також за забезпечення безпечного збереження даних. Для цього використовувалася база даних MongoDB, яка зберігає дані про завдання та користувачів у форматі документів JSON. MongoDB є NoSQL базою даних, що надає гнучкість у роботі з великими обсягами даних і дозволяє легко масштабувати додаток. Взаємодія з базою даних реалізована через бібліотеку Mongoose, яка спрощує процеси створення моделей, виконання CRUD-операцій (створення, читання, оновлення та видалення даних), а також забезпечує валідацію на рівні моделі.

Серед ключових технічних рішень особливої уваги заслуговують аспекти безпеки. Паролі користувачів шифруються за допомогою бібліотеки bcryptjs, що гарантує захист навіть у разі компрометації бази даних. Для автентифікації використовується технологія JSON Web Token (JWT). Коли користувач входить у систему, сервер створює токен, який зберігається на клієнтській стороні та використовується для підтвердження прав доступу до різних функцій додатка. Цей підхід забезпечує високий рівень захищеності та дозволяє працювати з персональними даними користувачів безпечно. Додатково, у додатку впроваджено захист від атак типу XSS та SQL-ін'єкцій.

Фронтенд додатка створено з використанням React.js, що дозволило розробити динамічний і гнучкий інтерфейс користувача. Стилізація інтерфейсу виконана за допомогою Tailwind CSS, що спрощує адаптацію додатка для роботи на різних екранах, включаючи мобільні пристрої. Також впроваджено бібліотеку React Toastify, яка відповідає за відображення інтерактивних сповіщень, що покращує взаємодію з користувачем. Для маршрутизації між сторінками використовується React Router, що забезпечує плавну навігацію без перезавантаження сторінки.

Процес розробки включав кілька етапів. Спочатку було проведено проектування, під час якого визначено основні функціональні вимоги, включаючи зручність використання, швидкість роботи та безпеку даних. Після цього здійснено розробку серверної частини додатка. Було створено REST API, що дозволяє виконувати всі необхідні операції із завданнями та користувачами. Наступним етапом стало створення клієнтської частини, яка забезпечує зручний доступ до функцій додатка. Окрему увагу приділили тестуванню:

функціональність перевірялась як вручну, так і за допомогою інструментів, таких як Postman для бекенду.

Однією з найбільших технічних викликів стала реалізація безпечної автентифікації користувачів. Використання JWT у поєднанні з bcryptjs для шифрування паролів дозволило досягти високого рівня захисту персональних даних. Додатково впроваджено middleware для перевірки токенів у кожному запиті, що гарантує, що доступ до функцій додатка отримують тільки авторизовані користувачі.

Результатом роботи став функціональний веб-застосунок, який відповідає заявленим вимогам. Додаток забезпечує зручність, швидкість і безпеку, а також є гнучким для подальших вдосконалень. Серед перспективних напрямків розвитку можна відзначити інтеграцію з іншими сервісами, наприклад, Google Calendar, та впровадження push-сповіщень для мобільних пристроїв. Проект став чудовою основою для вдосконалення навичок роботи з MERN стеком, а також для отримання досвіду розв'язання складних завдань у сфері веб-розробки.

Реалізація

1. Підключення до бази даних

Функція, яка демонструє, як додаток встановлює з'єднання з MongoDB. Використовується бібліотека Mongoose для підключення до бази даних. URI для бази зберігається у файлі `.env`

```
import mongoose from "mongoose";

const connectDB = async () => {
    try {
        const conn = await mongoose.connect(process.env.MONGO_URI);
        console.log(`MongoDB Connected: ${conn.connection.host}`.cyan.underline);
    } catch (error) {
        console.error(error.message);
        process.exit(1);
    }
}

export { connectDB }

```

2. CRUD-операції для завдань

a) Створення завдання
Функція, яка приймає дані із запиту, зберігає їх у базі та повертає створене завдання.
b) Отримання списку завдань
Функція, яка повертає список завдань, створених авторизованим користувачем.
c) Оновлення завдання
Функція, яка знаходить завдання за ID, оновлює його дані та повертає оновлену версію.
d) Видалення завдання
Функція для видалення завдання за ID або очищення всього списку завдань.

3. Аутентифікація користувачів

а) Реєстрація

Функція перевіряє, чи email унікальний, хешує пароль за допомогою bcryptjs, створює нового користувача та повертає JWT

```
const registerUser = expressAsyncHandler(async (req, res, next) => {
    const { name, email, password } = req.body;

    if (!name || !email || !password) {
        res.status(400);
        throw new Error('Please provide name, email and password');
    }

    if (password.length < 6) {
        res.status(400);
        throw new Error('Password must be at least 6 characters long');
    }

    const userExists = await User.findOne({ email });

    if (userExists) {
        res.status(400);
        throw new Error('User already exists');
    }

    const salt = await bcrypt.genSalt(10);
    const hashedPassowrd = await bcrypt.hash(password, salt);

    const user = await User.create({
        name,
        email,
        password: hashedPassowrd
    })

    if (user) {
        res.status(201).json({
            _id: user._id,
            name: user.name,
            email: user.email,
            token: generateToken(user._id)
        })
    } else {
        res.status(400);
        throw new Error('Invalid user data');
    }
})
```

б) Вхід

Функція перевіряє правильність email і пароля, створює JWT для користувача та повертає його у відповідь.

```
const loginUser = expressAsyncHandler(async (req, res, next) => {
    const { email, password } = req.body;

    if (!email || !password) {
        res.status(400);
        throw new Error('Please provide email and password');
    }

    const user = await User.findOne({ email });

    if (!user) {
        res.status(400);
        throw new Error('Invalid credentials');
    }

    if (user && (await bcrypt.compare(password, user.password))) {
        res.status(201).json({
            _id: user._id,
            name: user.name,
            email: user.email,
            token: generateToken(user._id)
        })
    } else {
        res.status(400);
        throw new Error('Invalid credentials');
    }
})
```

с)Генерація JWT

Допоміжна функція, яка створює токен на основі ID користувача.

```
const generateToken = (id) => {
    return jwt.sign({ id }, process.env.JWT_SECRET, {
        expiresIn: '30d'
    })
}
```

4. Middleware для захисту маршрутів

Middleware, який перевіряє наявність і валідність JWT токена, додає дані про користувача до запиту або відхиляє доступ.

```
const protect = expressAsyncHandler(async (req, res, next) => {
    let token;

    if (req.headers.authorization && req.headers.authorization.startsWith('Bearer')) {
        try {
            token = req.headers.authorization.split(' ')[1];
            const decoded = jwt.verify(token, process.env.JWT_SECRET);
            req.user = await User.findById(decoded.id).select('-password');
            next();
        } catch (error) {
            console.log(error);
            res.status(401);
            throw new Error('Not authorized to access this route');
        }
    }

    if (!token) {
        res.status(401);
        throw new Error('Not authorized, no token');
    }
})
```

5. Маршрутизація

a) Маршрути для завдань

Файл визначає маршрути для роботи із завданнями (`GET`, `POST`, `PUT`, `DELETE`) та підключає відповідні контролери.

b) Маршрути для користувачів

Файл визначає маршрути для реєстрації (`/register`) та входу користувачів (`/login`).

6. Моделі даних

a) Модель завдання

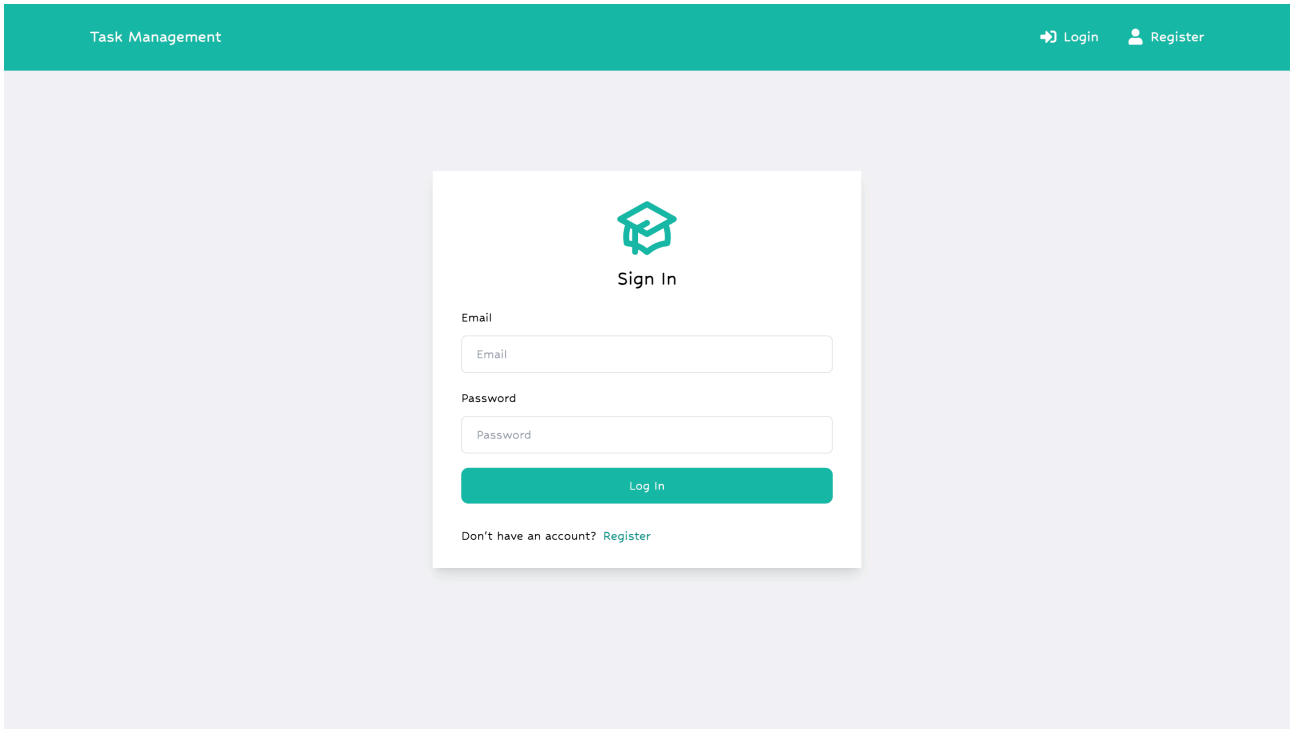
```
const taskSchema = new mongoose.Schema({
    user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User',
        required: true
    },
    title: {
        type: String,
        required: [true, 'Please add a title'],
        maxlength: 300
    },
    description: {
        type: String,
        required: [true, 'Please add a description'],
        maxlength: 1000
    },
    priority: {
        type: Number,
        required: [true, 'Please add a priority'],
        enum: [1, 2, 3, 4, 5]
    },
    startDate: {
        type: Date,
        default: Date.now
    },
    endDate: {
        type: Date,
        default: Date.now
    }
}, {
    timestamps: true
})
```

b) Модель користувача

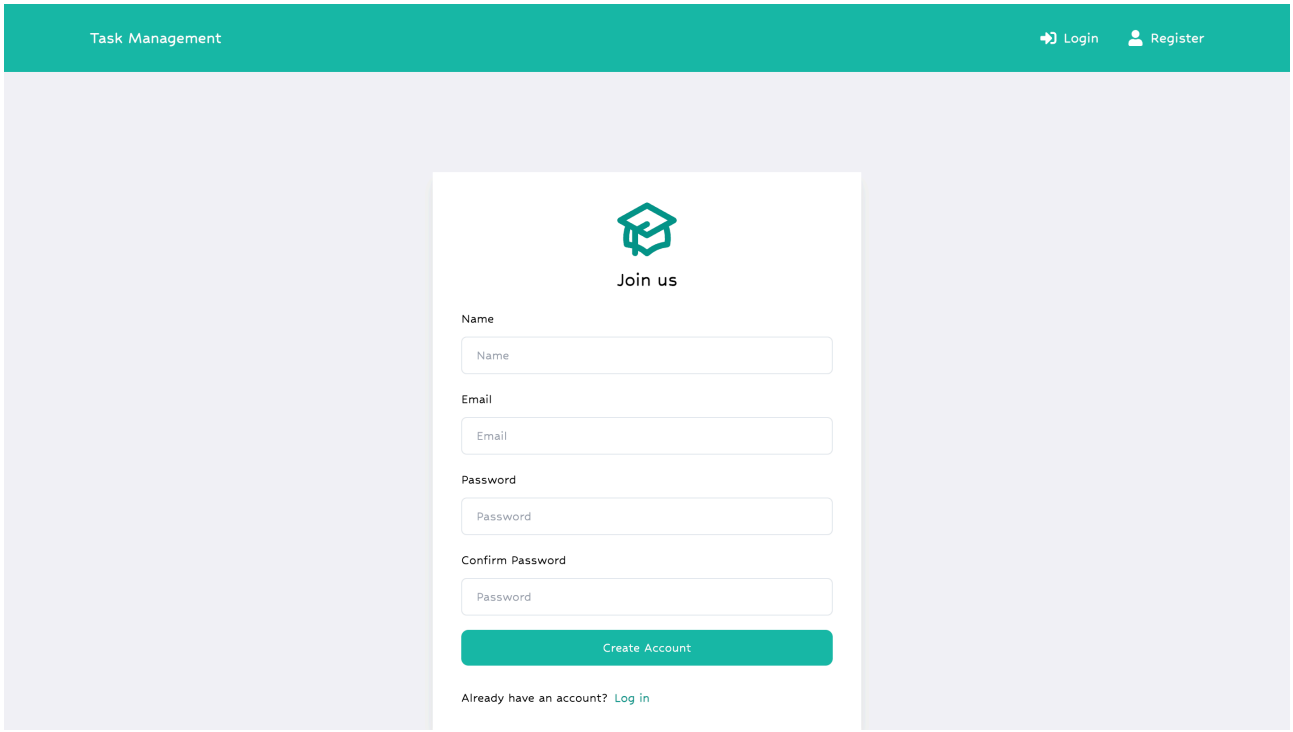
```
const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: [true, 'Please add a name'],
        maxlength: 200
    },
    email: {
        type: String,
        required: [true, 'Please add an email'],
        unique: true
    },
    password: {
        type: String,
        required: [true, 'Please add a password']
    },
}, {
    timestamps: true
})
```

7. Фронтенд

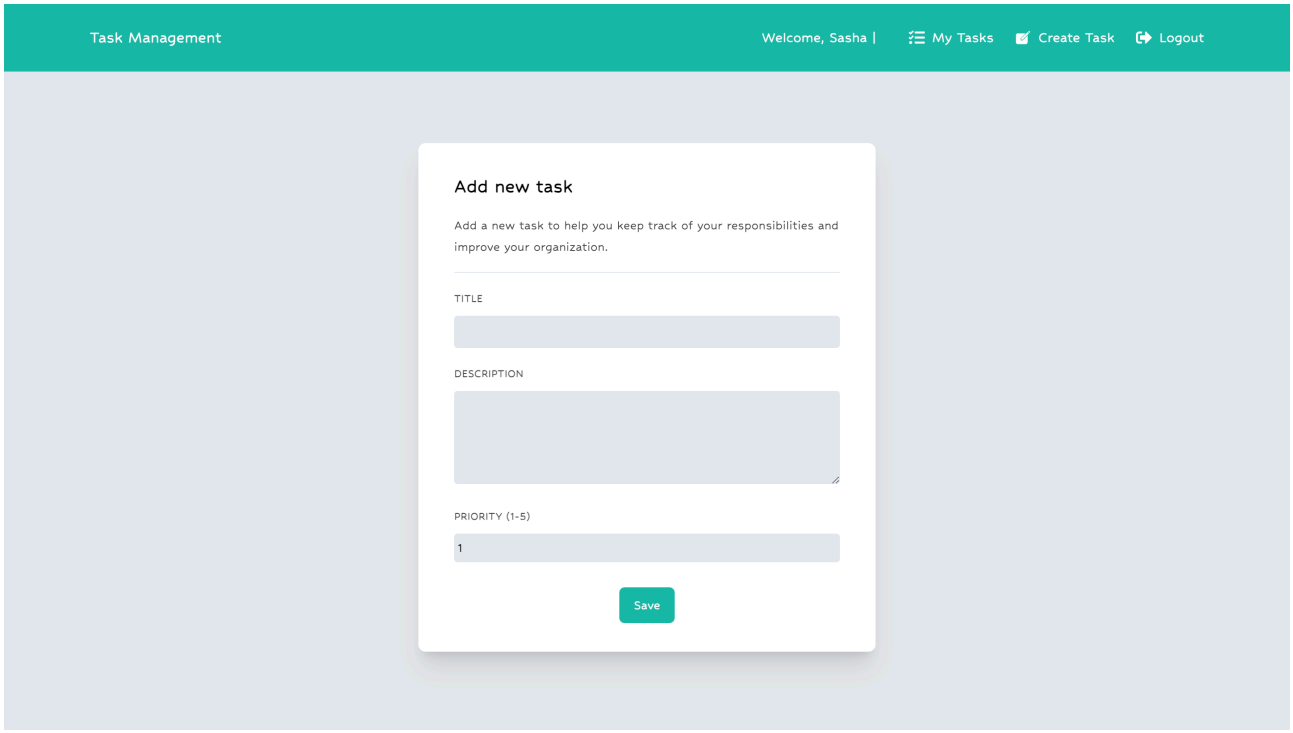
Login Page



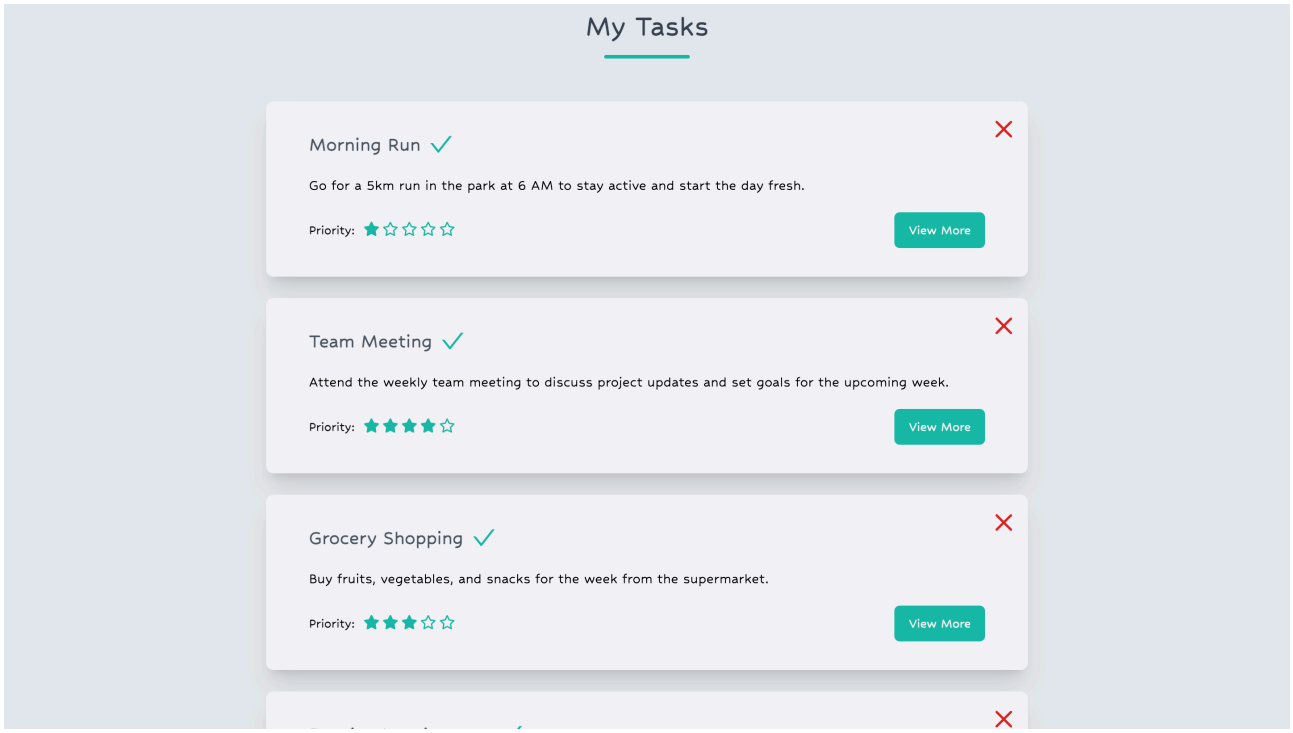
Registration Page



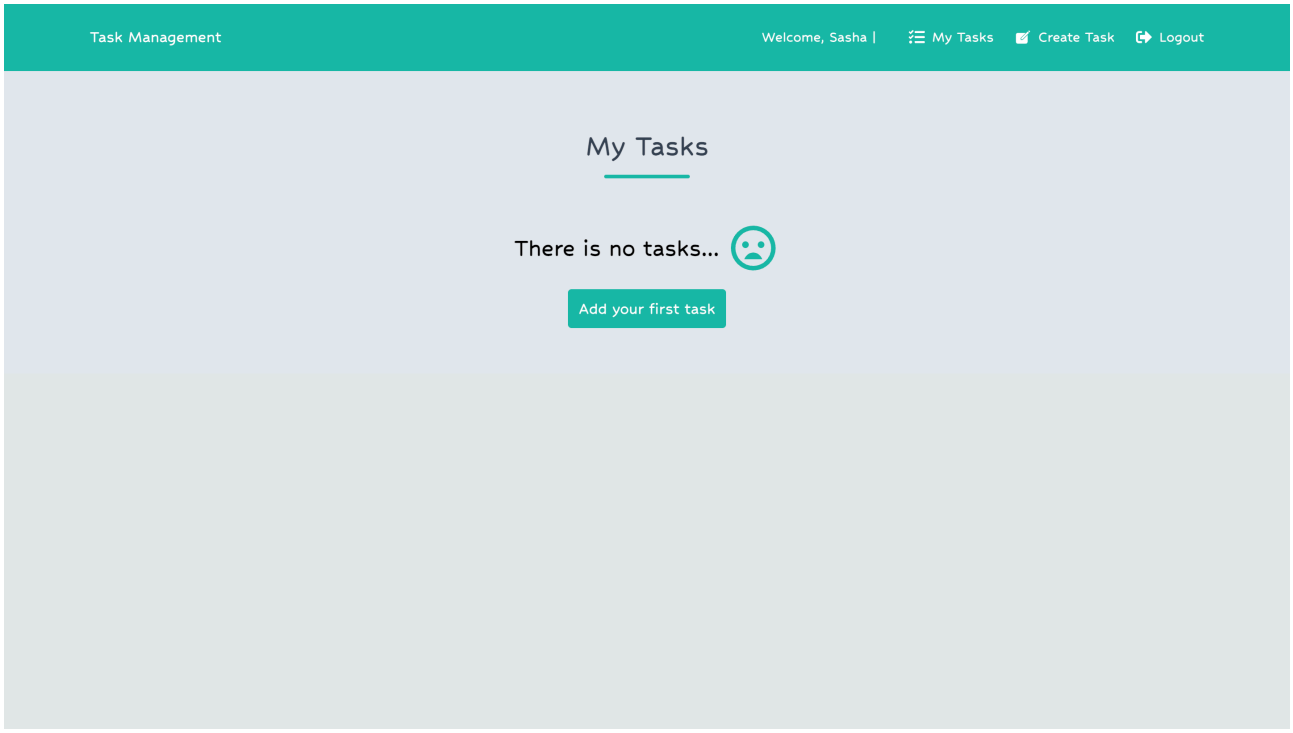
Create Task



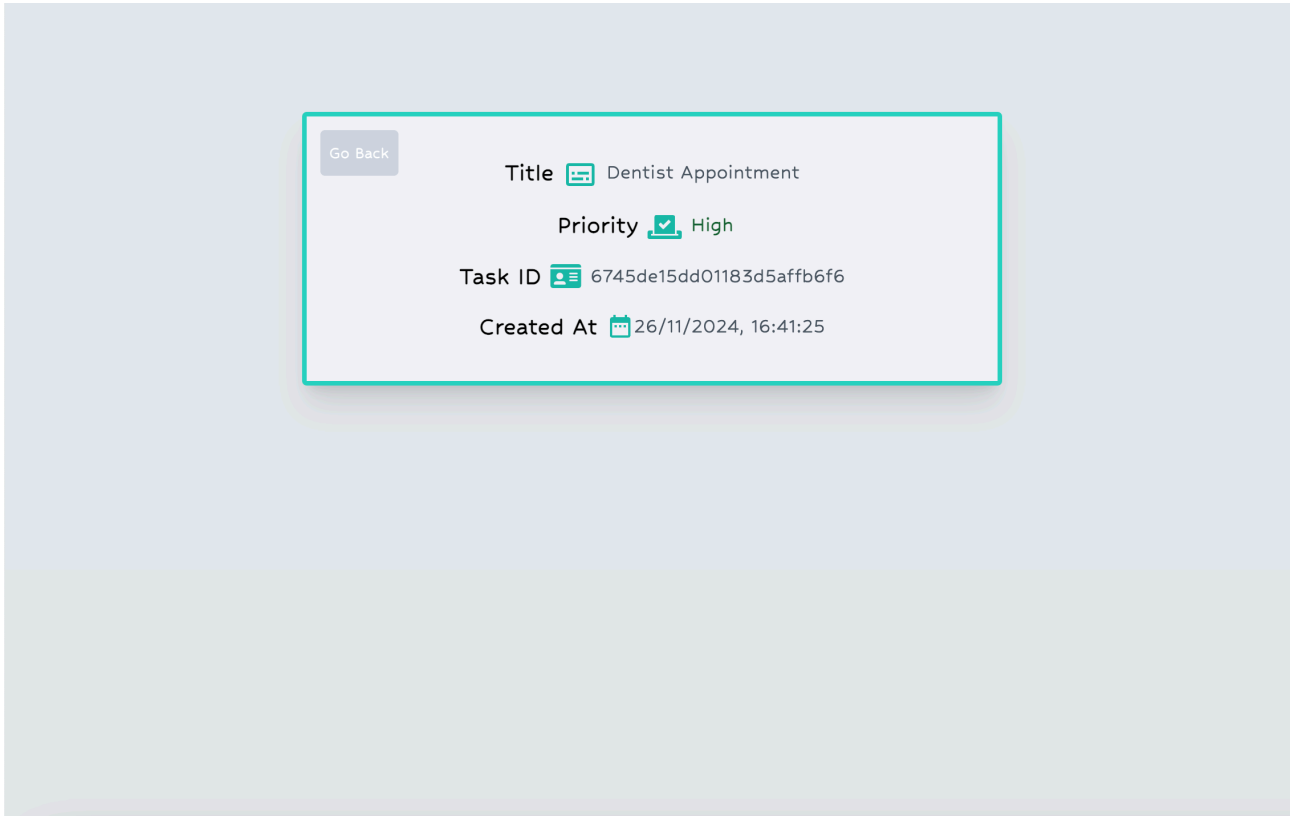
My Tasks



Empty Task List



Single Task



Посилання на проєкт https://github.com/stozhok/IS