МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Кафедра защиты информации

**Расчётно-графическая работа**

**по дисциплине: «Технологии и методы программирования»**

**на тему: «** *Разработка приложения* *таскера для компании, с интегрированным чат ботом для приёма заказов***»**

Выполнил: Проверил:

Студент гр. « *АБ-121* », « *АВТФ* » *ассистент кафедры ЗИ*

*Втюрин Александр Романович Медведев М. А.*

« 7 » июня 2023г. « » 20 г.

(подпись) (подпись)

Новосибирск 2023

**ОГЛАВЛЕНИЕ**

[**Задачи**: 3](#_Toc137014115)

[- Техническое задание на разработку сайта; 3](#_Toc137014116)

[- Макет проекта (при наличии). 4](#_Toc137014117)

[**Ход работы:** 5](#_Toc137014118)

[o Математическая модель алгоритмов (при наличии в работе): 6](#_Toc137014119)

[o Модель базы данных (при наличии в работе): 6](#_Toc137014120)

[o Шаблоны проектирования в работе: 6](#_Toc137014121)

[o Паттерны с примерами использования: 7](#_Toc137014122)

[**Результаты работы:** 8](#_Toc137014123)

[o Пользовательский сценарий: 8](#_Toc137014124)

[o Скриншоты с реализованным функционалом: 10](#_Toc137014125)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 14](#_Toc137014126)

[**ПРИЛОЖЕНИЕ** 15](#_Toc137014127)

**Цель:**

Целью расчётно-графической работы является разработка сайта таскера для компании, с интегрированным чат ботом для приёма заказов и заявок в тех поддержку

.

**Задачи**:

* Техническое задание на разработку сайта;
  + Структура сайта;
    - Окно добавления событий.
    - Панель переключения месяцев.
    - Сетка календаря.
    - Отображаемые события
    - Отображение размещённых заказов
  + Навигация по сайту:

Пользовательский интерфейс сайта должен обеспечивать наглядное, интуитивно понятное представление структуры размещенной на нем информации, быстрое и понятное управление задачами и календарём.

Страница отображения заказов должна отображать в понятной и удобной форме все размещённые заказы.

* Наполнение сайта:

Страница должна отображать все введённые события согласно заданному временному промежутку в понятной форме.

Модификация содержимого должна производится по средством основного интерфейса сайта.

Содержимое страницы с отображением заказов должно отображаться на основе текстового файла.

* Функциональные возможности:

В верхней части сайта должно отображаться меню перехода между страницами таскера и отображения заказов. Под навигационным меню, должна располагаться часть добавления событий с указанием имени задачи, даты начала-конца задачи в формате mm/dd/yyyy с указанием времени.

Ниже должна располагаться сетка календаря внутри которой по средствам линий отображается добавленная задача.

* Требования к дизайну:

Стиль сайта должен быть минималистичным и понятным пользователю.

Блок добавления событий должен быть явно отделён и интуитивно понятен для пользователя.

Внутри календаря должны отображаться добавленные события по средством линий (синий) с подписью названия события.

* Макет проекта (при наличии).
  + Отсутствует.

# **Ход работы:**

* Архитектура

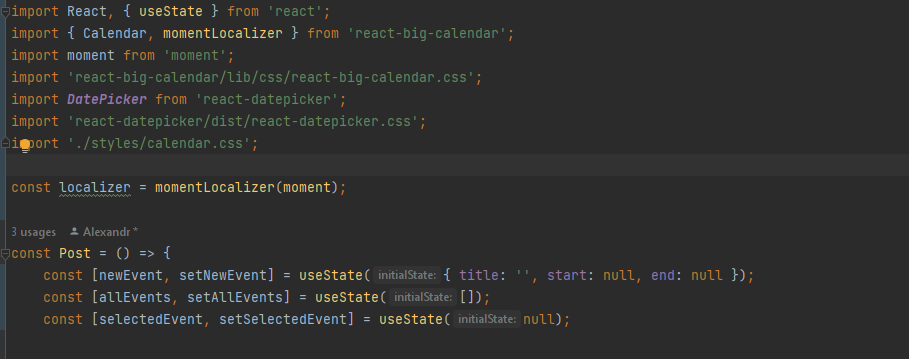
Была использована библиотека React для создания пользовательских интерфейсов (см. рис. 2.1).

Рис. 2.1 – Пример использования библиотеки React

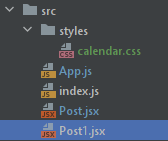
В основе архитектуры приложение лежит использование компонентов (как для отображения каждой страницы, так и для элементов на этих страницах). См. рис. 2.2 и 2.3.

Рис. 2.2 – Компоненты для отображения страницы

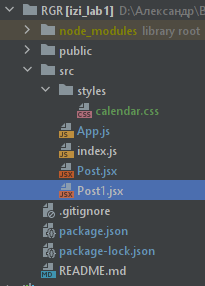
Общая организация файлов проекта выглядит следующим образом (см. рис. 2.3):

Рис. 2.3 – Общая организация файлов

В папке scr находятся файлы со всеми скриптами для страниц.

В папке styles находятся все файлы стиля страниц.

В папке node\_modulse располагаются все файлы подключённых библиотек (таких как react big calendar, datepicker и т.д.)

* Математическая модель алгоритмов (при наличии в работе):
  + - Отсутствует.
* Модель базы данных (при наличии в работе):
  + - Отсутствует.
* Шаблоны проектирования в работе:

Условный рендеринг

Начнём с условного рендеринга (conditional rendering). Условная отрисовка в React работает так же, как условия работы в JavaScript. Используйте JavaScript-операторы, например if или тернарный оператор, чтобы создать элементы, представляющие текущее состояние, и пусть React обновит пользовательский интерфейс для соответствия им.

Шаблон «провайдер»

Шаблон проектирования «провайдер» (provider) относится к возможностям React, которые появились сравнительно недавно. Как быть, если, скажем, свойства надо передать пятнадцати компонентам? В подобной ситуации полезно будет воспользоваться API React Context.

* Паттерны с примерами использования:

Conditional Rendering

Условная отрисовка в React работает так же, как условия работы в JavaScript. Используйте JavaScript-операторы, например if или тернарный оператор, чтобы создать элементы, представляющие текущее состояние, и пусть React обновит пользовательский интерфейс для соответствия им.

Пример: см. в приложении Post.jsx

# **Результаты работы:**

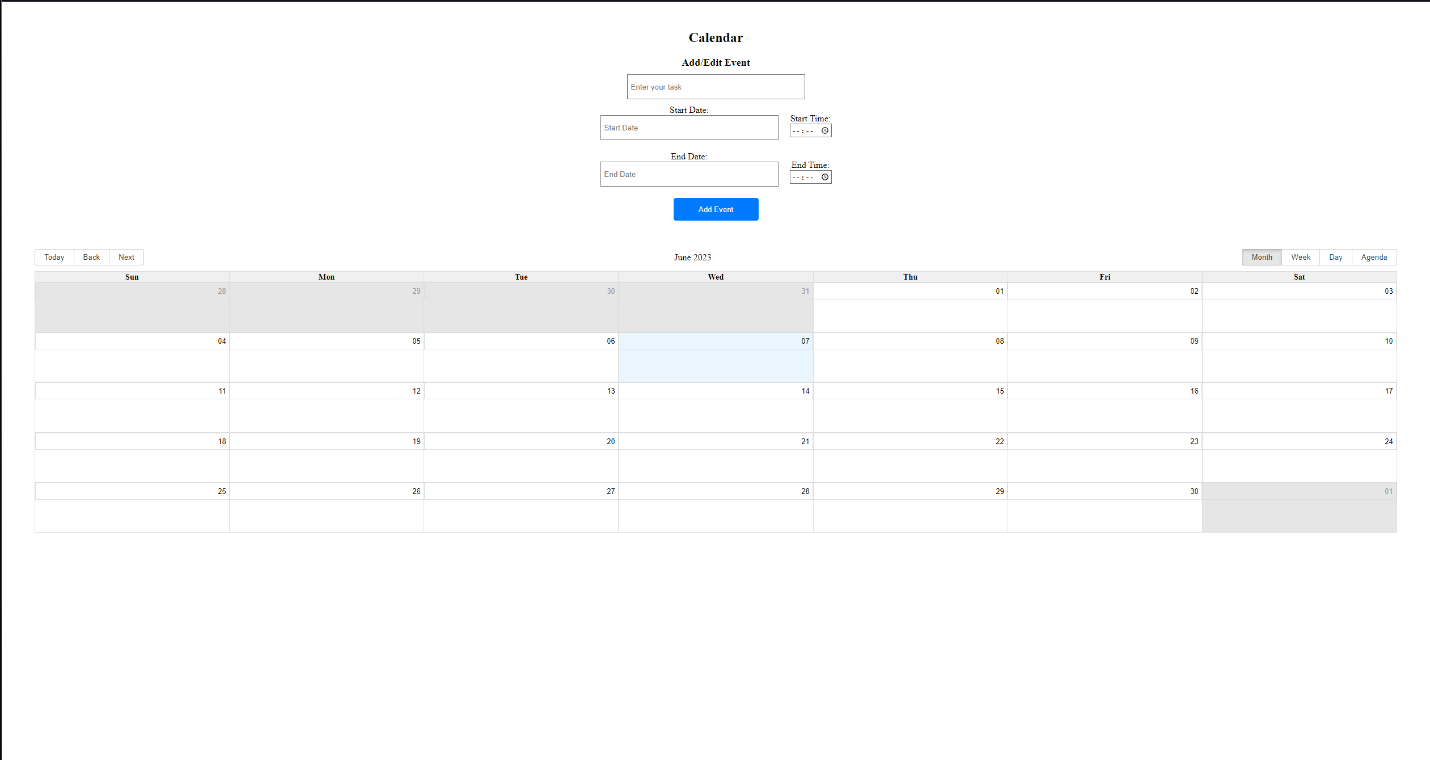
* Пользовательский сценарий:
  + Пользователь заходит на сайт и попадает на домашнюю страницу сайта (см. рис. 3.1)

Рис. 3.1 – Домашняя страница сайта

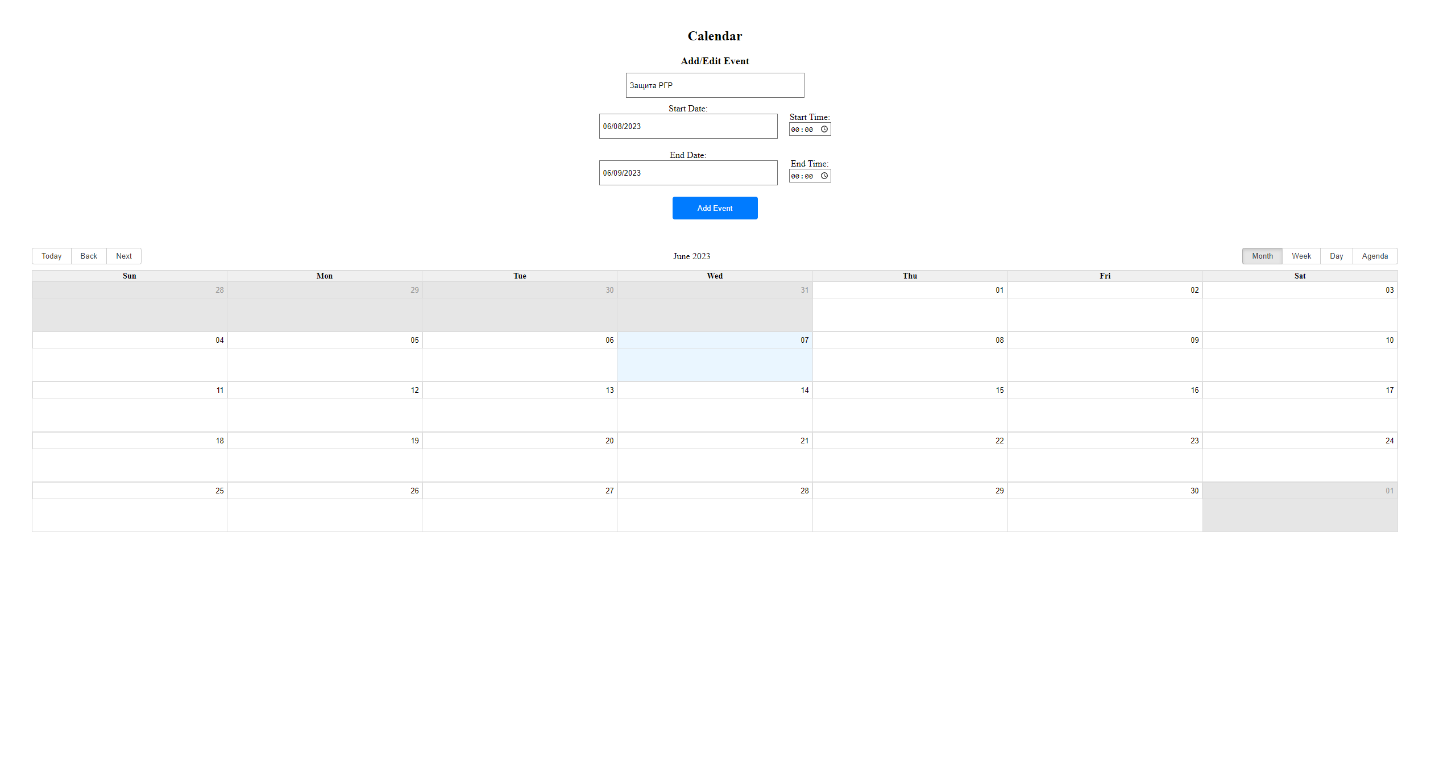
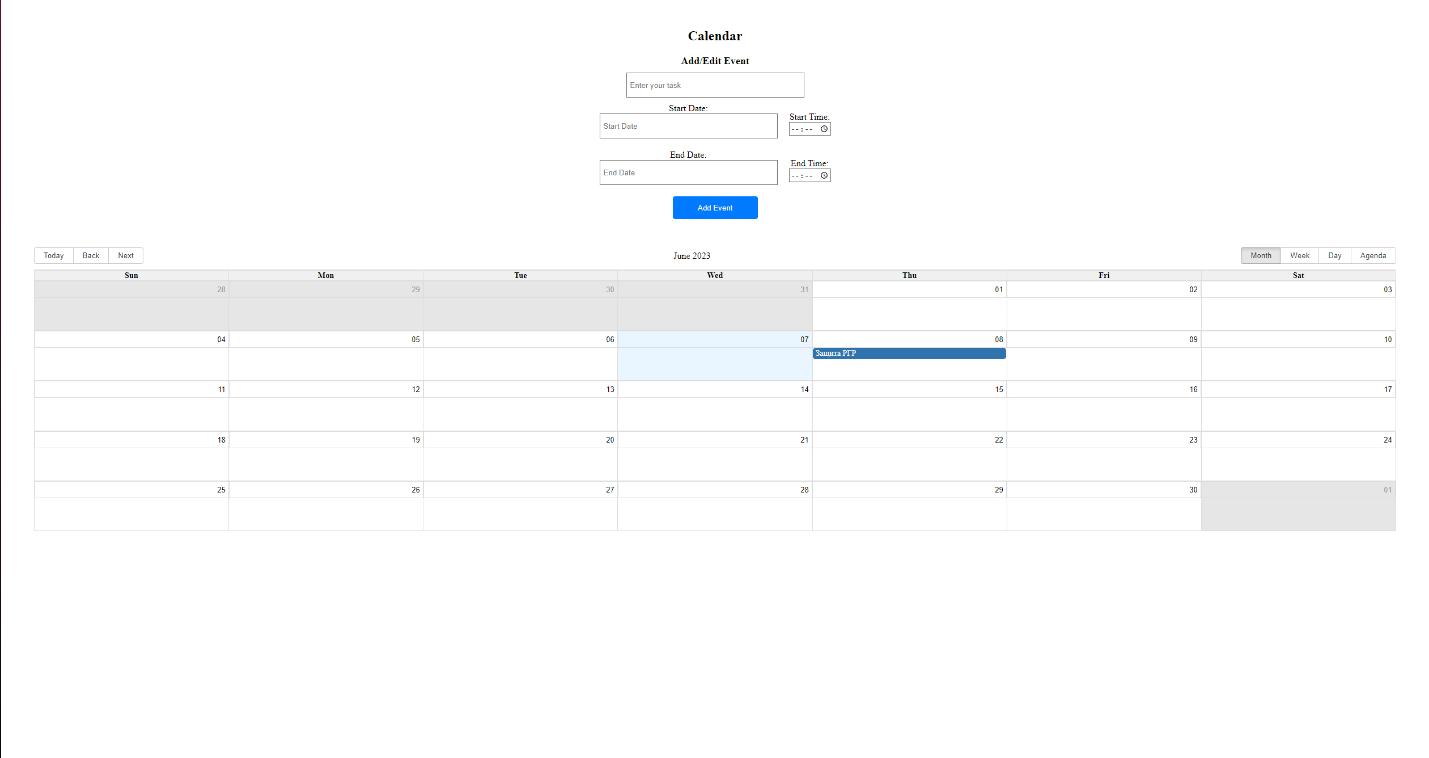
После входа на сайт можно добавлять события (см. рис. 3.2, 3.3)

Рисунок 3.2 – Добавление события

Рисунок 3.3 – Отображение добавленного события

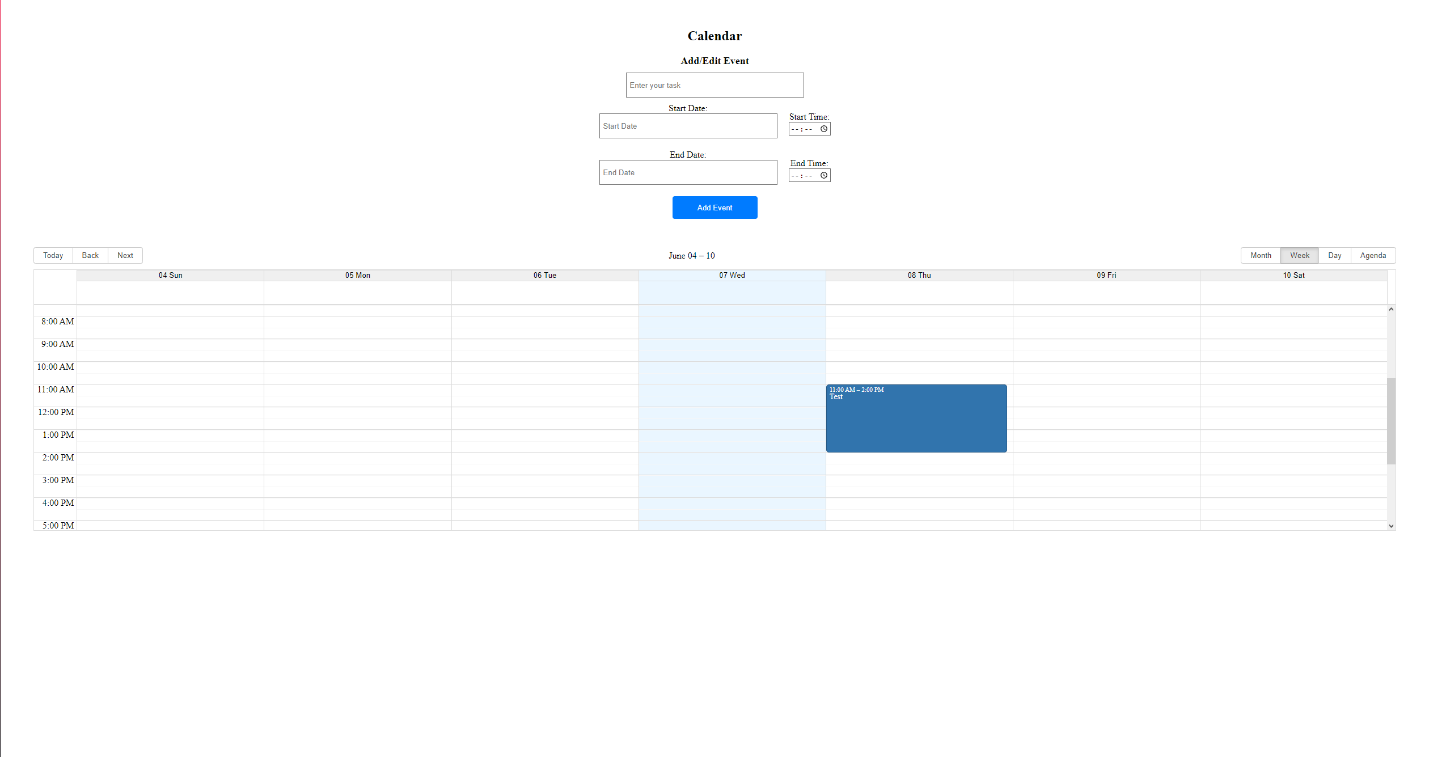
После добавления события можно переключить отображение на другой временной промежуток (см. рис. 3.4)

Рисунок 3.4 – Отображение события на временном отрезке «неделя»

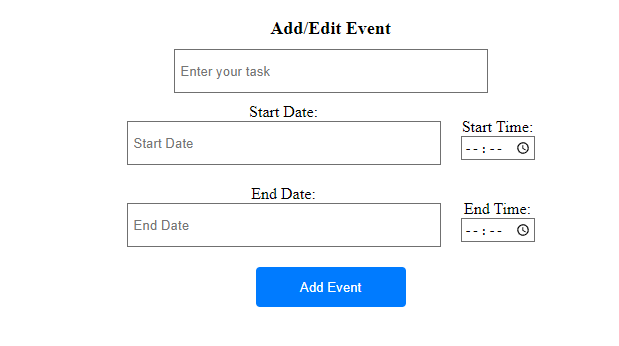
* Скриншоты с реализованным функционалом:
* Меню добавления событий в верхней части экрана (см. рис. 4.1)

Рисунок 4.1 - Меню добавления событий

* Календарь, отображающий добавленные события на промежутке месяц (см. рис. 4.2)

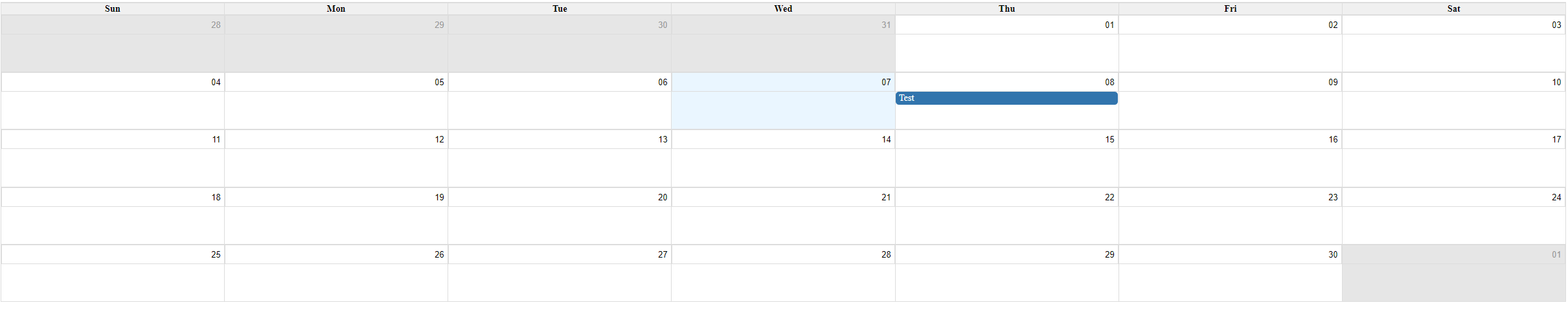


Рисунок 4.2 – Календарь отображающий события на промежутке месяц

* Календарь, отображающий добавленные события на промежутке неделя (см. рис. 4.3)

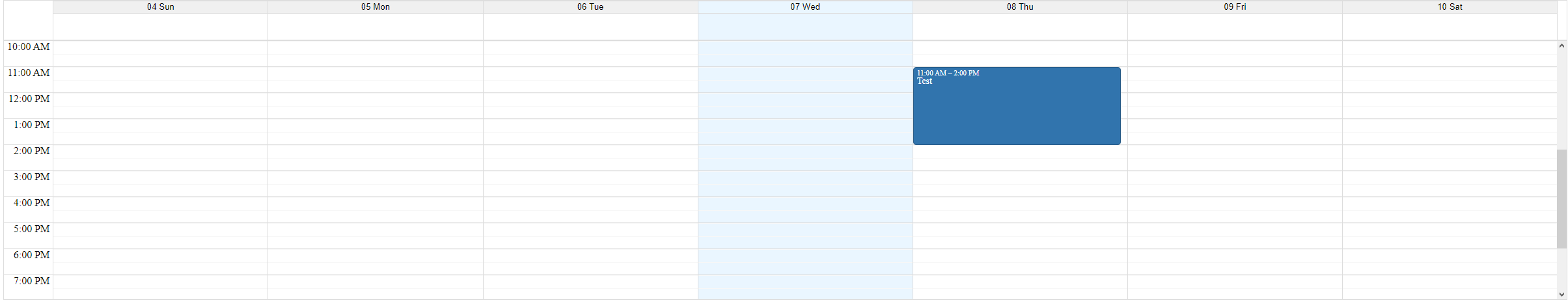


Рисунок 4.3 - Календарь отображающий события на промежутке неделя

* Календарь, отображающий добавленные события на промежутке день (см. рис. 4.4)

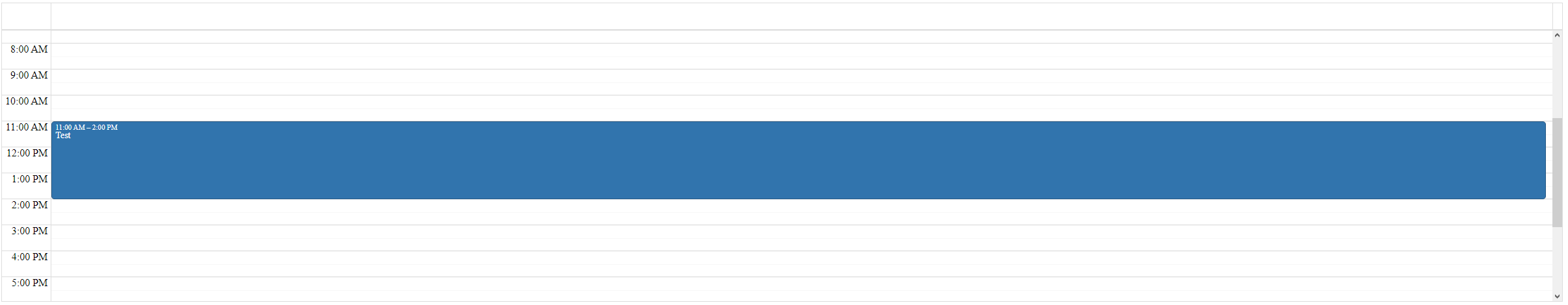


Рисунок 4.4 - Календарь отображающий события на промежутке день

* Меню переключения временных промежутков (см. рис. 4.5)

Рисунок 4.5 - Меню переключения временных промежутков

* Работа бота ассистента по приёму заказов (см. рис. 4.6)

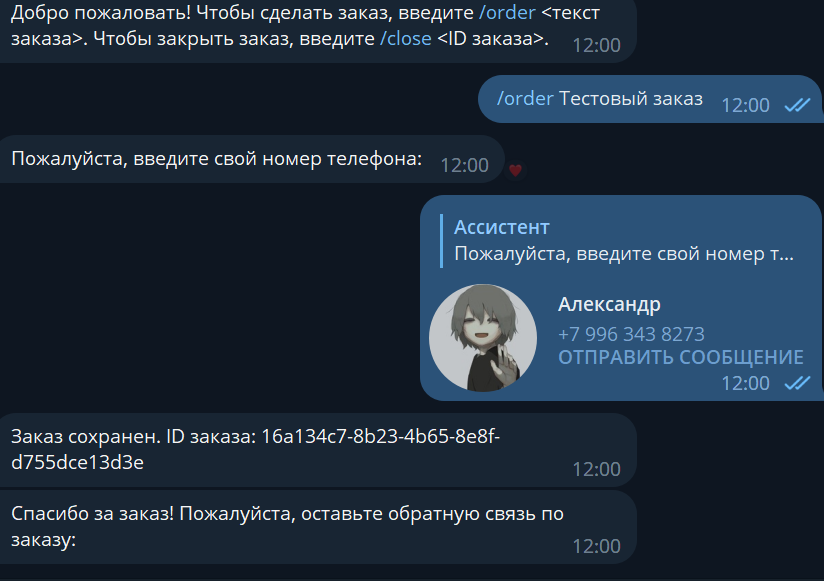


Рисунок 4.6 - Работа бота ассистента по приёму заказов

* Работа бота для просмотра и работы с заказами (см. рис 4.7)

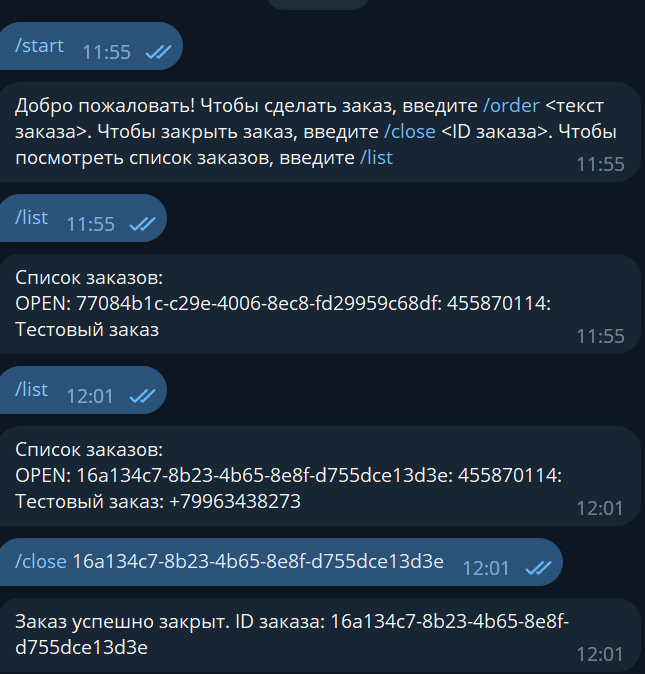


Рисунок 4.7 - Работа бота для просмотра и работы с заказами

**Вывод**

В результате выполнения расчётно-графической работы был разработан сайт таскер для компании, с интегрированным чат ботом для приёма заказов

* Разработан сайт таскер с помощью библиотек React и React Big Calendar
* Разработан дизайна сайта:
  + - Разработан дизайн меню добавления событий и их редактирования
    - Разработан дизайн для календаря выбора срока события
    - Разработан дизайн календаря отображающего события на разных временных промежутках
* Разработано динамическое изменение контента страниц сайта

Разработанный сайт позволяет добавлять и отслеживать задачи в удобном и наглядном формате

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Бондаренко Сергей Олегович Современные интерактивные веб-приложения - построение пользовательского интерфейса с React // Вестник науки и образования. 2018. №5 (41). URL: <https://cyberleninka.ru/article/n/sovremennye-interaktivnye-veb-prilozheniya-postroenie-polzovatelskogo-interfeysa-s-react> (дата обращения: 05.05.2023).
2. Яровая Екатерина Владимировна АРХИТЕКТУРНЫЕ РЕШЕНИЯ В БИБЛИОТЕКИ РЕАКТ // Столыпинский вестник. 2022. №5. URL: <https://cyberleninka.ru/article/n/arhitekturnye-resheniya-v-biblioteki-reakt> (дата обращения: 20.04.2023).
3. Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. СОВРЕМЕННЫЕ ФРЕЙМВОРКИ ДЛЯ РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЙ // Интеллектуальные технологии на транспорте. 2020. №4 (24). URL: <https://cyberleninka.ru/article/n/sovremennye-freymvorki-dlya-razrabotki-web-prilozheniy> (дата обращения: 23.04.2023).
4. Ахмеджанова Заррина, Гафурова Парвина Применение html и css для создания интерактивных Веб сайтов // Евразийский Союз Ученых. 2019. №4-3 (61). URL: https://cyberleninka.ru/article/n/primenenie-html-i-css-dlya-sozdaniya-interaktivnyh-veb-saytov (дата обращения: 22.04.2023).
5. Самоучитель CSS URL: <http://htmlbook.ru/samcss> (дата обращения: 23.04.2023).
6. React Big Calendar documentation URL: <https://github.com/jquense/react-big-calendar> (дата обращения: 05.05.2023).

**ПРИЛОЖЕНИЕ**

Листинг программы

index.js

import React from 'react';  
import ReactDOM from 'react-dom';  
import App from './App';  
  
  
ReactDOM.***render***(  
<App/>,  
 ***document***.getElementById('root')  
);

App.js

import React, {useState} from 'react'  
import './styles/calendar.css'  
import Post from "./Post";  
function App() {  
  
 return (  
 <Post/>  
);  
}  
  
export default App;

Post.jsx

import React, { useState } from 'react';  
import { Calendar, momentLocalizer } from 'react-big-calendar';  
import moment from 'moment';  
import 'react-big-calendar/lib/css/react-big-calendar.css';  
import ***DatePicker*** from 'react-datepicker';  
import 'react-datepicker/dist/react-datepicker.css';  
import './styles/calendar.css';  
  
const localizer = momentLocalizer(moment);  
  
const Post = () => {  
 const [newEvent, setNewEvent] = useState({ title: '', start: null, end: null });  
 const [allEvents, setAllEvents] = useState([]);  
 const [selectedEvent, setSelectedEvent] = useState(null);  
  
 const handleAddEvent = () => {  
 if (!newEvent.title || !newEvent.start || !newEvent.end) {  
 alert('Please enter all fields');  
 return;  
 }  
  
 const updatedEvents = [...allEvents, { ...newEvent, id: Date.now() }];  
 setAllEvents(updatedEvents);  
 setNewEvent({ title: '', start: null, end: null });  
 };  
  
 const handleEditEvent = () => {  
 if (!selectedEvent || !selectedEvent.title || !selectedEvent.start || !selectedEvent.end) {  
 alert('Please select an event and enter all fields');  
 return;  
 }  
  
 const updatedEvents = allEvents.map((event) => {  
 if (event.id === selectedEvent.id) {  
 return selectedEvent;  
 }  
 return event;  
 });  
  
 setAllEvents(updatedEvents);  
 setSelectedEvent(null);  
 };  
  
 const handleDeleteEvent = () => {  
 if (!selectedEvent) {  
 alert('Please select an event');  
 return;  
 }  
  
 const updatedEvents = allEvents.filter((event) => event.id !== selectedEvent.id);  
  
 setAllEvents(updatedEvents);  
 setSelectedEvent(null);  
 };  
  
 const handleSelectSlot = (slotInfo) => {  
 const start = slotInfo.start;  
 const end = slotInfo.end;  
  
 setNewEvent({ ...newEvent, start, end });  
 setSelectedEvent(null);  
 };  
  
 const handleSelectEvent = (event) => {  
 setSelectedEvent(event);  
 setNewEvent({ title: event.title, start: event.start, end: event.end });  
 };  
  
 return (  
 <div className="App">  
 <h1>Calendar</h1>  
 <h2>Add/Edit Event</h2>  
 <div>  
 <form className="Inputbox">  
 <input  
 value={newEvent.title}  
 className="input"  
 onChange={(e) => setNewEvent({ ...newEvent, title: e.target.value })}  
 type="text"  
 id="task"  
 placeholder="Enter your task"  
 required  
 />  
 <div className="datetime-container">  
 <div className="date-input">  
 <label htmlFor="start-date">Start Date:</label>  
 <DatePicker  
 selected={newEvent.start}  
 onChange={(date) => setNewEvent({ ...newEvent, start: date })}  
 dateFormat="MM/dd/yyyy"  
 placeholderText="Start Date"  
 className="input"  
 required  
 />  
 </div>  
 <div className="time-input">  
 <label htmlFor="start-time">Start Time:</label>  
 <input  
 type="time"  
 id="start-time"  
 value={newEvent.start ? moment(newEvent.start).format('HH:mm') : ''}  
 onChange={(e) => {  
 const startTime = moment(newEvent.start).set({  
 hour: e.target.value.split(':')[0],  
 minute: e.target.value.split(':')[1],  
 });  
 setNewEvent({ ...newEvent, start: startTime });  
 }}  
 />  
 </div>  
 </div>  
 <div className="datetime-container">  
 <div className="date-input">  
 <label htmlFor="end-date">End Date:</label>  
 <DatePicker  
 selected={newEvent.end}  
 onChange={(date) => setNewEvent({ ...newEvent, end: date })}  
 dateFormat="MM/dd/yyyy"  
 placeholderText="End Date"  
 className="input"  
 required  
 />  
 </div>  
 <div className="time-input">  
 <label htmlFor="end-time">End Time:</label>  
 <input  
 type="time"  
 id="end-time"  
 value={newEvent.end ? moment(newEvent.end).format('HH:mm') : ''}  
 onChange={(e) => {  
 const endTime = moment(newEvent.end).set({  
 hour: e.target.value.split(':')[0],  
 minute: e.target.value.split(':')[1],  
 });  
 setNewEvent({ ...newEvent, end: endTime });  
 }}  
 />  
 </div>  
 </div>  
 {selectedEvent ? (  
 <div className="button-container">  
 <button onClick={handleEditEvent} className="submit">  
 Update Event  
 </button>  
 <button onClick={handleDeleteEvent} className="submit delete">  
 Delete Event  
 </button>  
 </div>  
 ) : (  
 <button onClick={handleAddEvent} className="submit">  
 Add Event  
 </button>  
 )}  
 </form>  
 </div>  
 <div className="calendar-container">  
 <Calendar  
 localizer={localizer}  
 events={allEvents}  
 startAccessor="start"  
 endAccessor="end"  
 style={{ height: '100%' }}  
 selectable  
 onSelectSlot={handleSelectSlot}  
 onSelectEvent={handleSelectEvent}  
 />  
 </div>  
 </div>  
 );  
};  
  
export default Post;

calendar.css

/\* calendar.css \*/  
  
.App {  
 text-align: center;  
 margin-top: 50px;  
}  
  
h1 {  
 font-size: 24px;  
 margin-bottom: 20px;  
}  
  
h2 {  
 font-size: 18px;  
 margin-bottom: 10px;  
}  
  
.Inputbox {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 margin-bottom: 20px;  
}  
  
.input {  
 width: 300px;  
 height: 30px;  
 padding: 5px;  
 margin-bottom: 10px;  
}  
  
.datetime-container {  
 display: flex;  
 align-items: center;  
 margin-bottom: 10px;  
}  
  
.date-input {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 margin-right: 20px;  
}  
  
.time-input {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
}  
  
.submit {  
 width: 150px;  
 height: 40px;  
 background-color: #007bff;  
 color: #fff;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
}  
  
.submit:hover {  
 background-color: #0056b3;  
}  
  
.calendar-container {  
 height: 500px;  
 margin: 50px;  
}  
  
.rbc-calendar {  
 height: 100%;  
}  
  
.rbc-header {  
 font-size: 14px;  
 font-weight: bold;  
 text-align: center;  
 background-color: #f0f0f0;  
 border: 1px solid #ddd;  
 border-width: 1px 0;  
 z-index: 0;  
}  
  
.rbc-header > div {  
 padding: 10px;  
}  
  
.rbc-row-content {  
 display: flex;  
 flex-direction: column;  
 align-items: stretch;  
 flex-grow: 1;  
 overflow-y: auto;  
 position: relative;  
}  
  
.rbc-date-cell {  
 width: 100%;  
 height: 100%;  
 padding: 5px;  
 border: 1px solid #ddd;  
 text-align: right;  
 cursor: pointer;  
 position: relative;  
 z-index: 1;  
}  
  
.rbc-date-cell:hover {  
 background-color: #f5f5f5;  
}  
  
.rbc-event {  
 background-color: #007bff;  
 color: #fff;  
 font-size: 14px;  
 padding: 2px 5px;  
 border-radius: 4px;  
 cursor: pointer;  
}  
  
.rbc-today {  
 font-weight: bold;  
}  
  
.react-datepicker {  
 font-size: 14px;  
 border: 1px solid #ccc;  
 border-radius: 4px;  
 position: absolute;  
 z-index: 9999;  
 background-color: #fff;  
 height: auto;  
}  
  
.react-datepicker\_\_header {  
 background-color: #f0f0f0;  
 border-bottom: 1px solid #ccc;  
 padding: 10px;  
 font-weight: bold;  
 text-align: center;  
}  
  
.react-datepicker\_\_month-container {  
 display: flex;  
 flex-wrap: wrap;  
 justify-content: center;  
 position: relative;  
 z-index: 2;  
}  
  
.react-datepicker\_\_month {  
 margin: 0 10px;  
}  
  
.react-datepicker\_\_day-names {  
 display: flex;  
 justify-content: space-between;  
 margin-bottom: 10px;  
}  
  
.react-datepicker\_\_day-name {  
 flex-basis: 0;  
 flex-grow: 1;  
 text-align: center;  
}  
  
.react-datepicker\_\_week {  
 display: flex;  
 justify-content: space-between;  
 margin-bottom: 10px;  
}  
  
.react-datepicker\_\_day {  
 flex-basis: 0;  
 flex-grow: 1;  
 text-align: center;  
 cursor: pointer;  
}  
  
.react-datepicker\_\_day:hover {  
 background-color: #f5f5f5;  
}  
  
.react-datepicker\_\_day--selected {  
 background-color: #007bff;  
 color: #fff;  
}  
  
.react-datepicker\_\_day--today {  
 font-weight: bold;  
}  
  
.react-datepicker\_\_time-container {  
 display: flex;  
 justify-content: center;  
 margin-top: 10px;  
}  
  
.react-datepicker\_\_time {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
}  
  
.react-datepicker\_\_time-box {  
 display: flex;  
 align-items: center;  
}  
  
.react-datepicker\_\_time-box input[type='time'] {  
 width: 70px;  
 height: 30px;  
 margin-right: 5px;  
 padding: 5px;  
}  
  
.react-datepicker\_\_time-box label {  
 margin-right: 5px;  
}  
  
.react-datepicker\_\_navigation {  
 background-color: transparent;  
 border: none;  
 cursor: pointer;  
}  
  
.react-datepicker\_\_navigation--previous {  
 float: left;  
}  
  
.react-datepicker\_\_navigation--next {  
 float: right;  
}

orderbot.js

const TelegramBot = ***require***('node-telegram-bot-api');  
const fs = ***require***('fs');  
const path = ***require***('path');  
const { v4: uuidv4 } = ***require***('uuid');  
  
// Токен вашего бота, полученный от BotFather  
const token = '6239041673:AAGfUhZTyYUupwFRJHUKLaKL-VjsyPqH9SQ';  
  
  
// Путь к файлу заказов в папке проекта  
const ordersFilePath = path.join(***\_\_dirname***, 'orders.txt');  
  
// Создание экземпляра бота  
const bot = new TelegramBot(token, { polling: true });  
  
// Обработчик команды /start  
bot.onText(/**\/**start/, (msg) => {  
 const chatId = msg.chat.id;  
 bot.sendMessage(  
 chatId,  
 'Добро пожаловать! Чтобы сделать заказ, введите /order <текст заказа>. Чтобы закрыть заказ, введите /close <ID заказа>.'  
 );  
});  
  
// Обработчик команды /order  
bot.onText(/**\/**order (.+)/, (msg, match) => {  
 const chatId = msg.chat.id;  
 const orderText = match[1];  
 const orderId = uuidv4(); // Генерация уникального ID заказа  
 const userId = msg.from.id; // Получение номера пользователя  
  
 // Запрос номера телефона  
 bot.sendMessage(chatId, 'Пожалуйста, введите свой номер телефона:', {  
 reply\_markup: {  
 keyboard: [[{ text: 'Отправить номер телефона', request\_contact: true }]],  
 one\_time\_keyboard: true,  
 },  
 });  
  
 // Обработка введенного номера телефона  
 bot.on('contact', (contactMsg) => {  
 const phoneNumber = contactMsg.contact.phone\_number;  
  
 // Сохранение заказа в файл  
 fs.appendFile(  
 ordersFilePath,  
 `OPEN: ${orderId}: ${userId}: ${orderText}: ${phoneNumber}\n`,  
 (err) => {  
 if (err) {  
 ***console***.error(err);  
 bot.sendMessage(chatId, 'Произошла ошибка при сохранении заказа.');  
 } else {  
 bot.sendMessage(chatId, 'Заказ сохранен. ID заказа: ' + orderId);  
 bot.sendMessage(  
 chatId,  
 'Спасибо за заказ! Пожалуйста, оставьте обратную связь по заказу:'  
 );  
 }  
 }  
 );  
 });  
});  
  
// Обработчик команды /close  
bot.onText(/**\/**close (.+)/, (msg, match) => {  
 const chatId = msg.chat.id;  
 const orderId = match[1];  
  
 // Чтение файла заказов  
 fs.readFile(ordersFilePath, 'utf8', (err, data) => {  
 if (err) {  
 ***console***.error(err);  
 bot.sendMessage(chatId, 'Произошла ошибка при чтении заказов.');  
 } else {  
 const orders = data.trim().split('\n');  
 let foundOrder = false;  
  
 for (let i = 0; i < orders.length; i++) {  
 const order = orders[i];  
 const orderData = order.split(':');  
 const orderStatus = orderData[0].trim();  
 const existingOrderId = orderData[1].trim();  
  
 if (orderStatus === 'OPEN' && existingOrderId === orderId) {  
 const updatedOrder = order.replace('OPEN', 'CLOSED');  
  
 orders[i] = updatedOrder;  
 foundOrder = true;  
 break;  
 }  
 }  
  
 if (!foundOrder) {  
 bot.sendMessage(chatId, 'Заказ с указанным ID не найден или уже закрыт.');  
 return;  
 }  
  
 // Обновление файла заказов  
 fs.writeFile(ordersFilePath, orders.join('\n'), 'utf8', (err) => {  
 if (err) {  
 ***console***.error(err);  
 bot.sendMessage(chatId, 'Произошла ошибка при закрытии заказа.');  
 } else {  
 bot.sendMessage(  
 chatId,  
 'Заказ успешно закрыт. ID заказа: ' + orderId  
 );  
 }  
 });  
 }  
 });  
});  
  
// Запуск бота  
bot.on('polling\_error', (error) => {  
 ***console***.error(error);  
});

order admin.js

const TelegramBot = ***require***('node-telegram-bot-api');  
const fs = ***require***('fs');  
const path = ***require***('path');  
  
// Токен вашего бота, полученный от BotFather  
const token = '6100351907:AAGDnfTFiWfe2rSp-MKKdaEohLUS\_JoNlI8';  
  
  
// Путь к файлу заказов в папке проекта  
const ordersFilePath = path.join(***\_\_dirname***, 'orders.txt');  
  
// Создание экземпляра бота  
const bot = new TelegramBot(token, { polling: true });  
  
// Обработчик команды /start  
bot.onText(/**\/**start/, (msg) => {  
 const chatId = msg.chat.id;  
 bot.sendMessage(  
 chatId,  
 'Добро пожаловать! Чтобы сделать заказ, введите /order <текст заказа>. Чтобы закрыть заказ, введите /close <ID заказа>. Чтобы посмотреть список заказов, введите /list'  
 );  
});  
  
// Обработчик команды /order  
bot.onText(/**\/**order (.+)/, (msg, match) => {  
 const chatId = msg.chat.id;  
 const orderText = match[1];  
 const orderId = generateOrderId(); // Генерация уникального ID заказа  
  
 // Сохранение заказа в файл  
 fs.appendFile(  
 ordersFilePath,  
 `OPEN: ${orderId}: ${chatId}: ${orderText}\n`,  
 (err) => {  
 if (err) {  
 ***console***.error(err);  
 bot.sendMessage(chatId, 'Произошла ошибка при сохранении заказа.');  
 } else {  
 bot.sendMessage(chatId, 'Заказ сохранен. ID заказа: ' + orderId);  
 }  
 }  
 );  
});  
  
// Обработчик команды /close  
bot.onText(/**\/**close (.+)/, (msg, match) => {  
 const chatId = msg.chat.id;  
 const orderId = match[1];  
  
 // Чтение файла заказов  
 fs.readFile(ordersFilePath, 'utf8', (err, data) => {  
 if (err) {  
 ***console***.error(err);  
 bot.sendMessage(chatId, 'Произошла ошибка при чтении заказов.');  
 } else {  
 const orders = data.trim().split('\n');  
 let foundOrder = false;  
  
 for (let i = 0; i < orders.length; i++) {  
 const order = orders[i];  
 const orderData = order.split(':');  
 const orderStatus = orderData[0].trim();  
 const existingOrderId = orderData[1].trim();  
  
 if (orderStatus === 'OPEN' && existingOrderId === orderId) {  
 const updatedOrder = order.replace('OPEN', 'CLOSED');  
  
 orders[i] = updatedOrder;  
 foundOrder = true;  
 break;  
 }  
 }  
  
 if (!foundOrder) {  
 bot.sendMessage(  
 chatId,  
 'Заказ с указанным ID не найден или уже закрыт.'  
 );  
 return;  
 }  
  
 // Обновление файла заказов  
 fs.writeFile(ordersFilePath, orders.join('\n'), 'utf8', (err) => {  
 if (err) {  
 ***console***.error(err);  
 bot.sendMessage(chatId, 'Произошла ошибка при закрытии заказа.');  
 } else {  
 bot.sendMessage(  
 chatId,  
 'Заказ успешно закрыт. ID заказа: ' + orderId  
 );  
 }  
 });  
 }  
 });  
});  
  
// Обработчик команды /list  
bot.onText(/**\/**list/, (msg) => {  
 const chatId = msg.chat.id;  
  
 // Чтение файла заказов  
 fs.readFile(ordersFilePath, 'utf8', (err, data) => {  
 if (err) {  
 ***console***.error(err);  
 bot.sendMessage(chatId, 'Произошла ошибка при чтении заказов.');  
 } else {  
 const orders = data.trim().split('\n');  
 const openOrders = orders.filter((order) => order.includes('OPEN'));  
  
 if (openOrders.length === 0) {  
 bot.sendMessage(chatId, 'Нет доступных заказов.');  
 } else {  
 bot.sendMessage(chatId, 'Список заказов:\n' + openOrders.join('\n'));  
 }  
 }  
 });  
});  
  
// Запуск бота  
bot.on('polling\_error', (error) => {  
 ***console***.error(error);  
});  
  
// Генерация уникального ID заказа  
function generateOrderId() {  
 return ***Math***.random().toString(36).substr(2, 10);  
}