МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

КАФЕДРА 33

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

Канд. тех. наук, доцент	подпись, дата	Жиданов К.А.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине: Основы программирования

РАБОТУ ВЫПОЛНИЛ
Студент гр. 3331 подпись, дата М.Ф. Купфер
№
подпись, дата инициалы, фамилия

Санкт-Петербург 2025

Цель работы

Разработать веб-приложение для управления задачами (To-Do List) с использованием JavaScript (Node.js + React), обеспечивающее:

- Регистрацию и аутентификацию пользователей
- Добавление, редактирование, удаление и отметку выполнения задач
- Сохранение задач в базе данных (MongoDB)
- Интеграцию с Telegram-ботом для уведомлений и управления задачами

Задачи

Настройка серверной части (Backend)

- Создать сервер на Node.js с Express
- Реализовать REST API для работы с задачами
- Настроить аутентификацию пользователей (JWT)
- Подключить базу данных MongoDB для хранения задач
- Интегрировать Telegram-бота для уведомлений

1. Серверная часть (Node.js/Express)

```
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const cors = require('cors');
const TelegramBot = require('node-telegram-bot-api');

const app = express();
app.use(express.json());
app.use(cors());


const TELEGRAM_BOT_TOKEN = '7888410573:AAHOJ7zpJ7I9X5SzCxrfu-638d4uTGhV91Y';
const TELEGRAM_BOT_USERNAME = 'project3331_bot';
const bot = new TelegramBot(TELEGRAM_BOT_TOKEN, { polling: true });


mongoose.connect('mongodb://localhost:27017/todolist', {
```

```
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('MongoDB connection error:', err));

const UserSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  telegramChatId: { type: String, default: null }
});

const TaskSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  title: { type: String, required: true },
  description: { type: String, default: '' },
  completed: { type: Boolean, default: false },
  createdAt: { type: Date, default: Date.now },
  dueDate: { type: Date, default: null }
});

const User = mongoose.model('User', UserSchema);
const Task = mongoose.model('Task', TaskSchema);


bot.onText(/\/start/, (msg) => {
  const chatId = msg.chat.id;
  const welcomeMessage = `□ Привет! Я бот для управления задачами
(@${TELEGRAM_BOT_USERNAME}).\n\n` +
                         `Чтобы привязать аккаунт, отправь мне команду:\n` +
                         `/auth ТВОЙ_КОД_АУТЕНТИФИКАЦИИ\n\n` +
                         `Код можно получить в веб-приложении.`;
  bot.sendMessage(chatId, welcomeMessage);
});

bot.onText(/\/auth (.+)/, async (msg, match) => {
  const chatId = msg.chat.id;
  const authToken = match[1];

  try {
    const decoded = jwt.verify(authToken, process.env.JWT_SECRET || 'secret_key');
    const user = await User.findById(decoded.userId);

    if (user) {
      user.telegramChatId = chatId;
```

```
      await user.save();
      bot.sendMessage(chatId, '✅Ваш аккаунт успешно привязан!\nТеперь вы будете получать уведомления о задачах.');
    } else {
      bot.sendMessage(chatId, '❌Пользователь не найден.');
    }
  } catch (err) {
    bot.sendMessage(chatId, '❌Неверный код аутентификации. Пожалуйста, получите новый код в веб-приложении.');
  }
});

const authenticate = async (req, res, next) => {
  const token = req.header('x-auth-token');
  if (!token) return res.status(401).json({ message: 'No token, authorization denied' });

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET || 'secret_key');
    req.user = await User.findById(decoded.userId);
    next();
  } catch (err) {
    res.status(401).json({ message: 'Token is not valid' });
  }
};


app.post('/api/register', async (req, res) => {
  try {
    const { username, password } = req.body;


    const existingUser = await User.findOne({ username });
    if (existingUser) {
      return res.status(400).json({ message: 'Username already exists' });
    }

    const hashedPassword = await bcrypt.hash(password, 10);
    const user = new User({ username, password: hashedPassword });
    await user.save();

    const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET || 'secret_key', { expiresIn: '1h' });
    res.json({ token, username: user.username });
  } catch (err) {
    res.status(400).json({ message: err.message });
```

```
  }
});

app.post('/api/login', async (req, res) => {
  try {
    const { username, password } = req.body;
    const user = await User.findOne({ username });
    if (!user) return res.status(400).json({ message: 'User not found' });

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(400).json({ message: 'Invalid credentials' });

    const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET ||
'secret_key', { expiresIn: '1h' });
    res.json({
      token,
      username: user.username,
      telegramLinked: !!user.telegramChatId
    });
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});



const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
  console.log(`Telegram bot @${TELEGRAM_BOT_USERNAME} is active`);
});
```

2. Клиентская часть (React)

Создайте компонент для работы с Telegram-ботом:

```
import React, { useState } from 'react';
import { useAuth } from '../context/AuthContext';
import axios from 'axios';
import {
  Button,
  Dialog,
  DialogTitle,
  DialogContent,
  DialogContentText,
```

```
  DialogActions,
  Typography,
  IconButton,
  Box
} from '@mui/material';
import TelegramIcon from '@mui/icons-material/Telegram';

const TelegramConnect = () => {
  const { currentUser } = useAuth();
  const [open, setOpen] = useState(false);
  const [authToken, setAuthToken] = useState('');
  const [isConnected, setIsConnected] = useState(currentUser?.telegramLinked ||
false);

  const handleClickOpen = async () => {
    try {
      const res = await axios.get('/api/telegram-auth');
      setAuthToken(res.data.token);
      setOpen(true);
    } catch (err) {
      console.error('Error generating auth token:', err);
    }
  };

  const handleClose = () => {
    setOpen(false);
  };

  const handleConnectSuccess = () => {
    setIsConnected(true);
    handleClose();
  };

  return (
    <Box sx={{ display: 'flex', alignItems: 'center', gap: 1 }}>
      <Typography variant="body1">
        Telegram: {isConnected ? 'Connected ✓' : 'Not connected'}
      </Typography>

      {!isConnected && (
        <>
          <IconButton
            color="primary"
            onClick={handleClickOpen}
            title="Connect Telegram"
```

```
        >
          <TelegramIcon fontSize="large" />
        </IconButton>

        <Dialog open={open} onClose={handleClose}>
          <DialogTitle>Connect Telegram Account</DialogTitle>
          <DialogContent>
            <DialogContentText>
              To connect your Telegram account:
            </DialogContentText>
            <ol>
              <li>Start a chat with
<strong>@{TELEGRAM_BOT_USERNAME}</strong></li>
              <li>Send the following command:</li>
            </ol>
            <Box sx={{
              p: 2,
              mt: 2,
              mb: 2,
              backgroundColor: '#f5f5f5',
              borderRadius: 1,
              wordBreak: 'break-all'
            }}>
              <code>/auth {authToken}</code>
            </Box>
            <DialogContentText>
              After successful connection, this dialog will close automatically.
            </DialogContentText>
          </DialogContent>
          <DialogActions>
            <Button onClick={handleClose}>Cancel</Button>
            <Button
              onClick={() => {
                window.open(`https://t.me/${TELEGRAM_BOT_USERNAME}`,
'_blank');
                handleConnectSuccess();
              }}
              color="primary"
            >
              Open Telegram
            </Button>
          </DialogActions>
        </Dialog>
      </>
    )}
```

```
    </Box>
  );
};

export default TelegramConnect;
```