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

	11.06.2025	К. А. Жиданов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: Технологии и методы программирования

РАБОТУ ВЫПОЛНИЛ

Студент гр. №	3331	11.06.2025	Чмель В.М.
		подпись, дата	инициалы, фамилия

Санкт-Петербург

2025 год

Цель работы:

Разработать удобное и функциональное приложение для управления задачами — ToDo List, в котором будут такие возможности как: создание, редактирование, удаление и отметка выполнения задач, система авторизации пользователей, синхронизация данных между веб-версией и Telegram-ботом.

Результат выполненной работы:

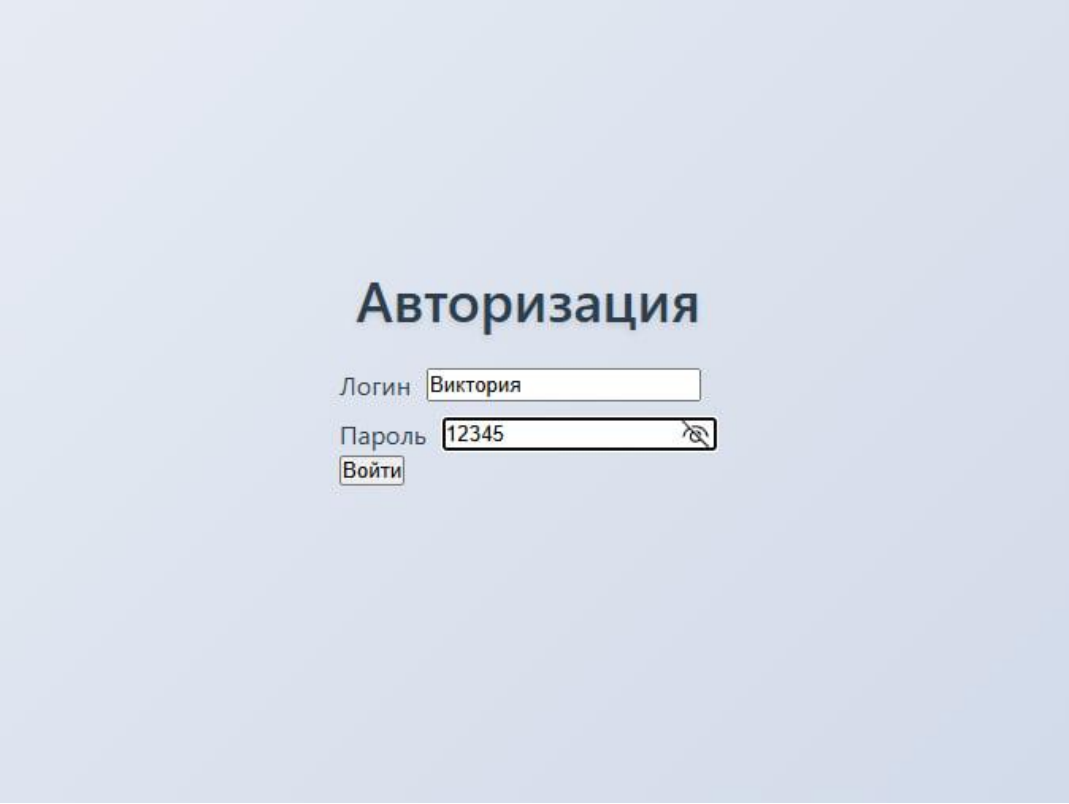

Рисунок 1. Авторизация

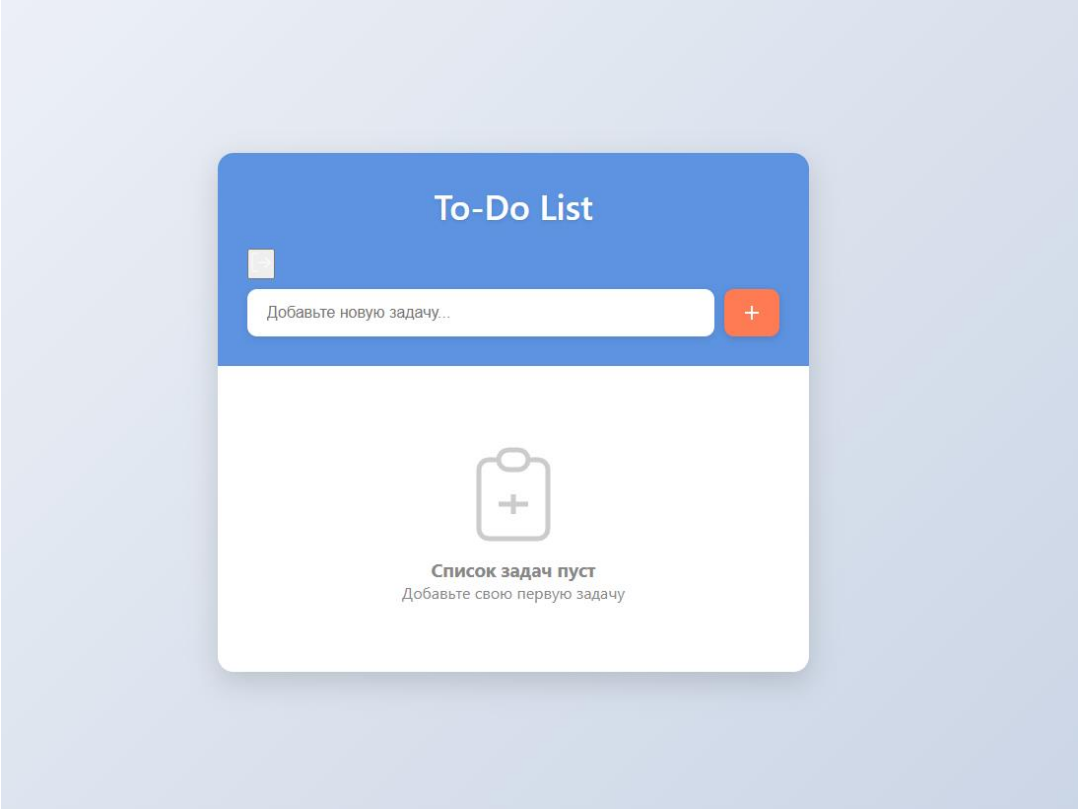

Рисунок 2.To-Do List.

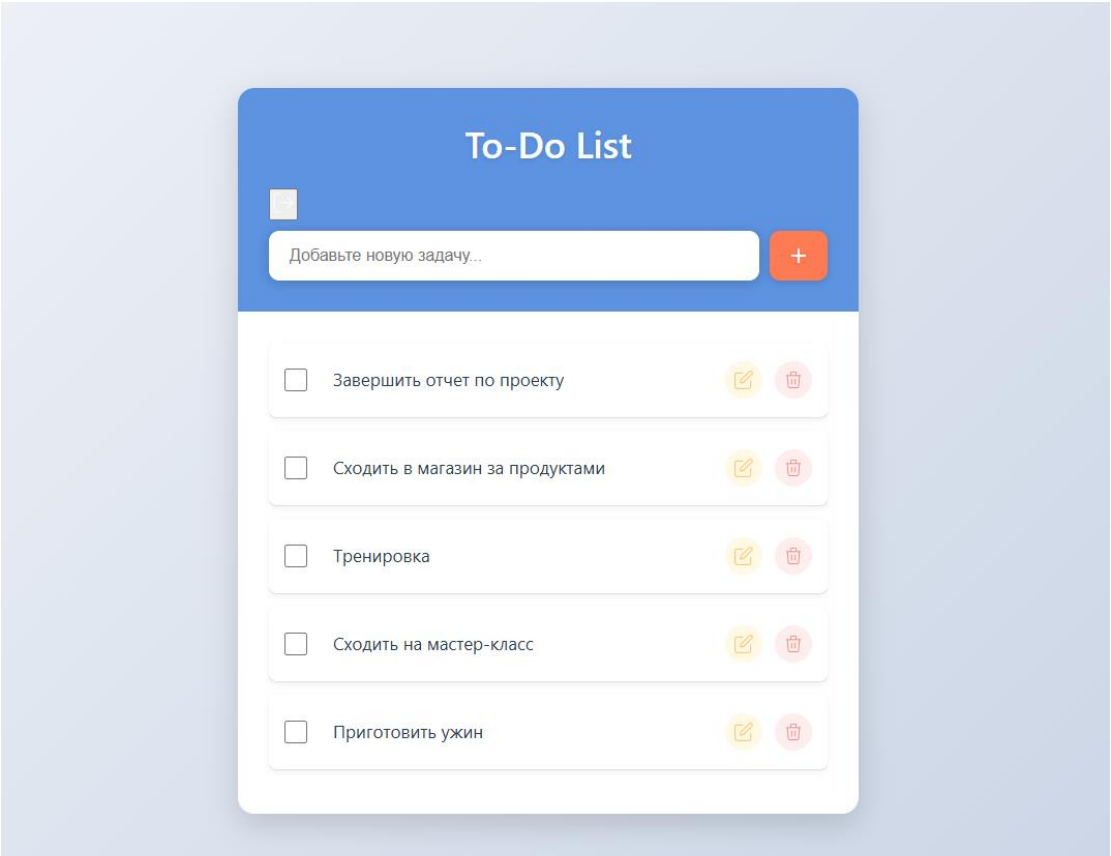

Рисунок 3. Пример записи списка дел

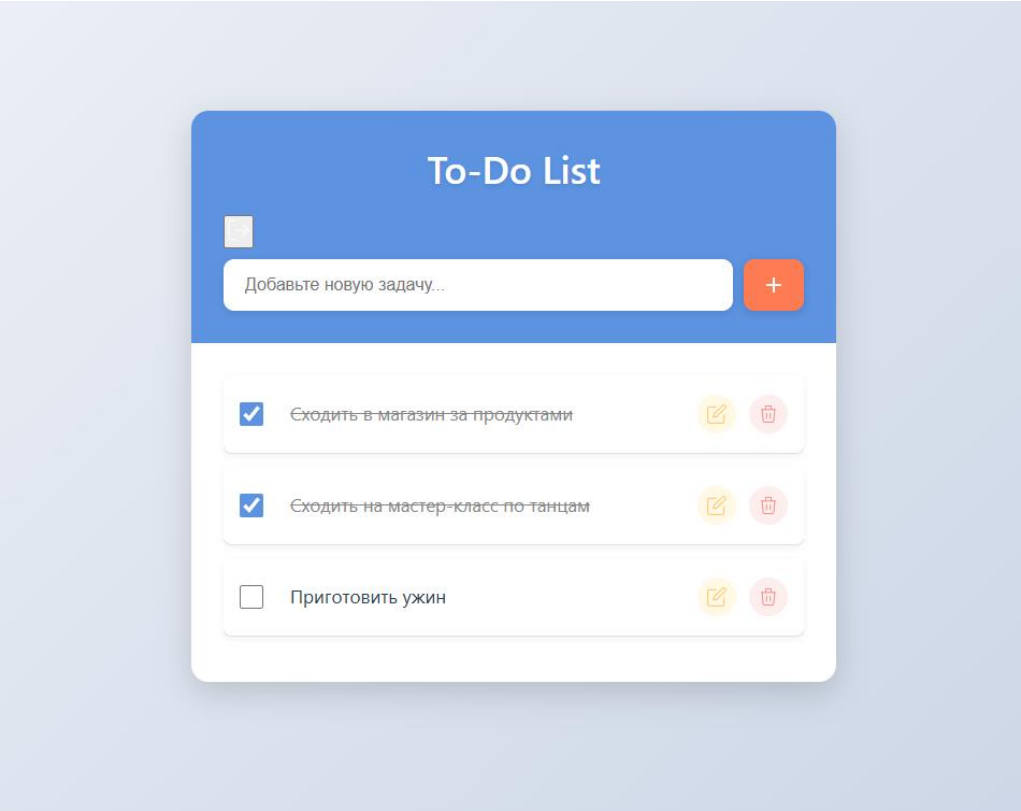

Рисунок 4. Выполненные задачи.

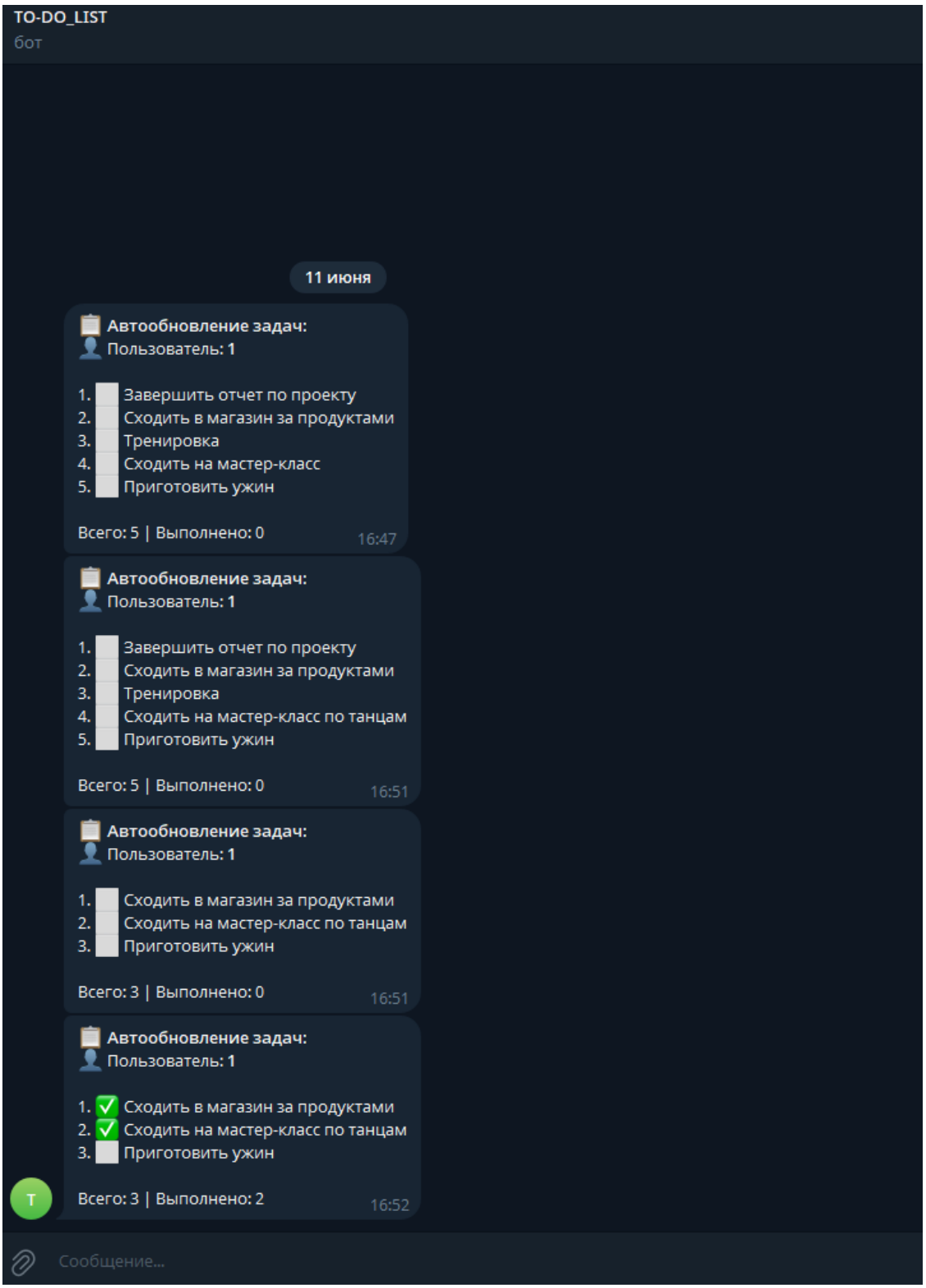

Рисунок 5. Интеграция с телеграммом через бота.

Запросы для нейросети:

- Как создать "TO-DO LIST" на языке js, HTML на основе имеющихся файлов
- Объясни и сделай добавление элементов в список, удаление элементов из списка, редактирование элементов
- Сделай красивое оформление сайта
- Раздели написанный код на css и html и js
- Объясни как сделать интеграцию с телеграммом
- Добавь автоотправку в телеграмм после добавления, изменения или удаления
- Как исправить ошибку отображения сообщений в телеграмме, почему они не приходят
- Добавь авторизацию пользователя на сайте по заранее записанному паролю и логину
- Сделай кпонку для выхода с сайта (чтобы заново ввести логин и пароль и попасть на сайт)
- Должно быть сохранение всех задач при обновлении страницы
- Сделай окошко для ввода новой задачи, также кнопки для удаления, добавления, редактирования и для пометки о выполненных задачах

Код проекта:

Код и описание файла index.html:

Документ представляет собой HTML-страницу для управления списком задач с авторизацией пользователей. Код разделен на три основные части:

1. Экран авторизации - видимый при первой загрузке
2. Основное приложение - скрыто до успешной авторизации
3. Модальное окно - для дополнительных взаимодействий

```
<!DOCTYPE html>
<html lang="ru">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Умный To-Do List</title>
  <link rel="stylesheet" href="styles.css">
</head>
```

- Это стандартное начало любой веб-страницы
- Указывает, что страница на русском языке
- Настраивает правильное отображение на всех устройствах
- Задает название вкладки в браузере
- Подключает файл со стилями (CSS)

```
<body>
  <!-- Экран авторизации -->
  <div id="auth-container" class="auth-container">
    <div class="auth-form">
      <h1>Авторизация</h1>
      <div class="input-group">
        <label for="username">Логин</label>
        <input
          type="text"
          id="username"
          placeholder="Введите логин"
          aria-label="Логин"
        >
      </div>
      <div class="input-group">
        <label for="password">Пароль</label>
        <input
```

```
          type="password"
          id="password"
          placeholder="Введите пароль"
          aria-label="Пароль"
        >
      </div>
      <button id="loginBtn" class="btn-login">Войти</button>
      <div id="authMessage" class="auth-message"></div>
    </div>
  </div>
```

- Показывает окно для входа
- Содержит:
  - Заголовок "Авторизация"
  - Поле для логина (обычный текст)
  - Поле для пароля (текст скрыт точками)
  - Кнопку "Войти"
  - Место для сообщений об ошибках

```
  <!-- Основное приложение (скрыто до авторизации) -->
  <div id="app-container" class="container" style="display:none">
    <header>
      <div class="header-top">
        <h1>To-Do List</h1>
        <button id="logoutBtn" class="btn-logout" aria-label="Выйти">
          <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="white">
```

```
            <path d="M9 21H5a2 2 0 0 1-2-2V5a2 2 0 0 1 2-
2h4M16 17l5-5-5-5M21 12H9"/>
          </svg>
        </button>
      </div>

      <div class="input-group">
        <input
          type="text"
          id="taskInput"
          placeholder="Добавьте новую задачу..."
          aria-label="Описание задачи"
        >
        <button id="addBtn" aria-label="Добавить задачу">
          <svg width="24" height="24" viewBox="0 0 24 24"
fill="none" stroke="currentColor" stroke-width="2">
            <path d="M12 5V19M5 12H19"/>
          </svg>
        </button>
      </div>

    </header>

    <main>
      <ul id="taskList">
        <!-- Задачи будут добавляться сюда -->
      </ul>
    </main>
  </div>
```

- Это главная часть приложения (изначально скрыта)
- Содержит:
    - Шапку с названием и кнопкой выхода
    - Поле для ввода новых задач с кнопкой добавления
    - Пустой список для отображения задач

```
<!-- Модальное окно -->
<div id="modal" class="modal"></div>
<script src="script.js"></script>
```

- Место для всплывающих окон (пока пустое)
- Подключение JavaScript-кода, который сделает приложение рабочим

```
</body>
</html>
```

Как это работает:

1. Пользователь видит только форму входа
2. После успешного входа форма скрывается, появляется список задач
3. В основном интерфейсе можно:
    - Добавлять новые задачи (кнопка "+")
    - Видеть список всех задач
    - Выходить из системы (иконка двери)

Запросы для нейронки:

- Допиши код для создания сайта на тему "TO-DO LIST" на языке js, HTML на основе имеющихся файлов
- Сделай добавление элементов в список, удаление элементов из списка, редактирование элементов
- Сделай красивое оформление сайта
- Раздели написанный код на css и html и js

- Добавь автоотправку в телеграмм после добавления, изменения или удаления
- Добавь авторизацию пользователя на сайте по заранее записанному паролю и логину
- Сделай кпонку для выхода с сайта (чтобы заново ввести логин и пароль и попасть на сайт)
- Должно быть сохранение всех задач при обновлении страницы
- Сделай окошко для ввода новой задачи, также кнопки для удаления, добавления, редактирования и для пометки о выполненных задачах

Код и описание файла styles.css:

```
:root {
  --primary: #5D93E1;
  --primary-dark: #3a6bc7;
  --secondary: #FF7B54;
  --light: #F9F9F9;
  --dark: #2C3E50;
  --gray: #E0E0E0;
  --success: #4CAF50;
  --edit: #FFC107;
  --delete: #F44336;
}
```

- Здесь задаются основные цвета, которые потом используются во всём приложении
- Это как палитра красок - один раз задали и везде используем

```
* {
  box-sizing: border-box; /* Правильный расчёт размеров */
  margin: 0; /* Убираем отступы */
  padding: 0;
```

```
}
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; /* Красивый шрифт */
  background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%); /* Фон с градиентом */
  min-height: 100vh;
  display: flex; /* Центрируем содержимое */
  justify-content: center;
  align-items: center;
  padding: 20px;
  color: var(--dark);
}
```

- Это "настройки по умолчанию" для всей страницы
- Делают фон приятным и центрируют содержимое

```
.container {
  width: 100%;
  max-width: 600px;
  background: white;
  border-radius: 16px;
  box-shadow: 0 10px 30px rgba(0,0,0,0.15);
  overflow: hidden;
}
```

- Задаёт стиль для "окна" нашего приложения
- Делает его похожим на карточку с тенями

```
header {
  padding: 30px;
  background: var(--primary);
```

```
  color: white;
  position: relative;
}
h1 {
  margin: 0 0 20px 0;
  font-weight: 600;
  text-align: center;
  font-size: 2.2rem;
  text-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.input-group {
  display: flex;
  gap: 10px;
  margin-top: 10px;
}
```

- Стили для верхней синей части
- Здесь будет название и поле для ввода задач

```
#taskInput {
  flex: 1;
  padding: 15px 20px;
  border: none;
  border-radius: 10px;
  font-size: 16px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
  transition: box-shadow 0.3s;
}

#taskInput:focus {
```

```
  outline: none;
  box-shadow: 0 2px 12px rgba(0,0,0,0.2);
}

#addBtn {
  background: var(--secondary);
  border: none;
  border-radius: 10px;
  width: 56px;
  cursor: pointer;
  color: white;
  transition: transform 0.2s, background 0.3s;
  box-shadow: 0 2px 8px rgba(0,0,0,0.15);
  display: flex;
  justify-content: center;
  align-items: center;
}

#addBtn:hover {
  background: #e06a45;
  transform: scale(1.05);
}

main {
  padding: 30px;
}

#taskList {
  list-style: none;
```

```
  padding: 0;
  margin: 0;
}
```

- Стили для поля ввода новых задач
- Красивая оранжевая кнопка "+", которая реагирует на наведение

```
.task-item {
  display: flex;
  align-items: center;
  padding: 18px 15px;
  border-bottom: 1px solid var(--gray);
  animation: fadeIn 0.4s;
  transition: all 0.3s;
  background: white;
  border-radius: 8px;
  margin-bottom: 12px;
  box-shadow: 0 2px 6px rgba(0,0,0,0.05);
}

.task-item:hover {
  box-shadow: 0 4px 10px rgba(0,0,0,0.1);
  transform: translateY(-2px);
}

@keyframes fadeIn {
  from { opacity: 0; transform: translateY(-10px); }
  to { opacity: 1; transform: translateY(0); }
}
```

```
.task-checkbox {
  margin-right: 15px;
  width: 22px;
  height: 22px;
  cursor: pointer;
  accent-color: var(--primary);
}

.task-content {
  flex: 1;
  padding: 0 10px;
  word-break: break-word;
  font-size: 17px;
  line-height: 1.4;
}

.task-actions {
  display: flex;
  gap: 12px;
  opacity: 0.7;
  transition: opacity 0.3s;
}

.task-item:hover .task-actions {
  opacity: 1;
}

.btn {
  background: none;
```

```
  border: none;
  cursor: pointer;
  padding: 6px;
  border-radius: 50%;
  transition: all 0.2s;
  width: 36px;
  height: 36px;
  display: flex;
  align-items: center;
  justify-content: center;
}

.btn:hover {
  transform: scale(1.1);
}
```

- Каждая задача выглядит как отдельная карточка
- При наведении она "оживает" - приподнимается

```
.btn-edit {
  background: rgba(255, 193, 7, 0.15);
}

.btn-edit:hover {
  background: rgba(255, 193, 7, 0.25);
}

.btn-delete {
  background: rgba(244, 67, 54, 0.12);
}
```

```
.btn-delete:hover {
  background: rgba(244, 67, 54, 0.2);
}

.completed .task-content {
  text-decoration: line-through;
  color: #888;
}

.edit-input {
  width: 100%;
  padding: 12px 15px;
  border: 2px solid var(--primary);
  border-radius: 8px;
  font-size: 16px;
  margin-bottom: 10px;
  box-shadow: 0 2px 6px rgba(93, 147, 225, 0.2);
}

.edit-buttons {
  display: flex;
  gap: 10px;
  justify-content: flex-end;
}

.btn-save, .btn-cancel {
  padding: 8px 20px;
  border: none;
```

```
  border-radius: 6px;
  cursor: pointer;
  font-size: 15px;
  font-weight: 500;
  transition: all 0.2s;
}

.btn-save {
  background: var(--primary);
  color: white;
}

.btn-save:hover {
  background: var(--primary-dark);
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(93, 147, 225, 0.3);
}

.btn-cancel {
  background: #f5f5f5;
  color: var(--dark);
  border: 1px solid var(--gray);
}

.btn-cancel:hover {
  background: #eaeaea;
  transform: translateY(-2px);
}
```

```
.empty-state {
  text-align: center;
  padding: 40px 20px;
  color: #888;
}

.empty-state img {
  width: 120px;
  opacity: 0.6;
  margin-bottom: 20px;
}

.share-container {
  margin-top: 20px;
  text-align: center;
}

.btn-share {
  background: var(--success);
  color: white;
  border: none;
  border-radius: 10px;
  padding: 12px 25px;
  font-size: 16px;
  cursor: pointer;
  transition: all 0.3s;
  display: inline-flex;
  align-items: center;
  gap: 8px;
```

```
  box-shadow: 0 4px 12px rgba(76, 175, 80, 0.25);
}

.btn-share:hover {
  background: #43A047;
  transform: translateY(-2px);
  box-shadow: 0 6px 15px rgba(76, 175, 80, 0.35);
}

/* Модальное окно */
.modal {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0,0,0,0.5);
  z-index: 1000;
  justify-content: center;
  align-items: center;
}
```

- Стили для всплывающих окон (подтверждение удаления и т.д.)
- Затемняет фон и показывает окно по центру

```
.modal-content {
  background: white;
  padding: 30px;
  border-radius: 16px;
```

```
  text-align: center;
  max-width: 80%;
  box-shadow: 0 10px 30px rgba(0,0,0,0.25);
}

.modal-content p {
  font-size: 18px;
  margin: 0;
  color: var(--dark);
}
```

- Кнопки с иконками (карандаш и корзина)
- Полупрозрачный фон разного цвета
- Реагируют на наведение курсора

```
@media (max-width: 480px) {
  header, main {
    padding: 20px;
  }
```

- Специальные стили для мобильных устройств
- Делают приложение удобным на маленьких экранах

```
  h1 {
    font-size: 1.8rem;
  }

  .task-actions {
    gap: 8px;
  }

  .task-item {
```

```
    padding: 15px 12px;
  }
}
```

Код и описание файла script.js:

```
// DOM элементы
const authContainer = document.getElementById('auth-container');
const appContainer = document.getElementById('app-container');
const usernameInput = document.getElementById('username');
const passwordInput = document.getElementById('password');
const loginBtn = document.getElementById('loginBtn');
const authMessage = document.getElementById('authMessage');
const logoutBtn = document.getElementById('logoutBtn');
const taskInput = document.getElementById('taskInput');
const addBtn = document.getElementById('addBtn');
const taskList = document.getElementById('taskList');
const shareBtn = document.getElementById('shareBtn');
const modal = document.getElementById('modal');

// Настройки Telegram
const TELEGRAM_BOT_TOKEN = '7790946474:AAHlaH712c4ucgsXGbeQS3CTCHXBW2mYotg';
const TELEGRAM_CHAT_ID = '1010330513';
const AUTO_SEND_DELAY = 3000;

// Конфигурация пользователей (логины и пароли)
const USERS = {
  'Виктория': '12345',  // Логин: Виктория, Пароль: 12345
```

```
  '1': '1',
};

// Загрузка состояния авторизации
let currentUser = localStorage.getItem('todoUser');
let tasks = [];
let sendTimeout = null;

// Инициализация приложения
function initApp() {
  if (currentUser && USERS[currentUser]) {
    showApp();
  } else {
    showAuth();
  }

  // Загрузка задач
  tasks                                                    =
JSON.parse(localStorage.getItem(`todoTasks_${currentUser}`
)) || [];
  renderTasks();
}

// Показать экран авторизации
function showAuth() {
  authContainer.style.display = 'flex';
  appContainer.style.display = 'none';
  currentUser = null;
  localStorage.removeItem('todoUser');
```

```
}

// Показать основное приложение
function showApp() {
  authContainer.style.display = 'none';
  appContainer.style.display = 'block';

  // Загружаем задачи для текущего пользователя
  tasks                                                    =
JSON.parse(localStorage.getItem(`todoTasks_${currentUser}`
)) || [];
  renderTasks();
}

// Авторизация пользователя
function login() {
  const username = usernameInput.value.trim();
  const password = passwordInput.value.trim();

  if (!username || !password) {
    showAuthMessage('Введите логин и пароль');
    return;
  }

  if (!USERS[username]) {
    showAuthMessage('Пользователь не найден');
    return;
  }
```

```
  if (USERS[username] !== password) {
    showAuthMessage('Неверный пароль');
    return;
  }

  // Успешная авторизация
  currentUser = username;
  localStorage.setItem('todoUser', username);
  showApp();

  // Первоначальная отправка списка при загрузке
  if (tasks.length > 0) {
    setTimeout(() => {
      sendDataToTelegram(tasks, true);
    }, 2000);
  }
}

// Показать сообщение авторизации
function showAuthMessage(message) {
  authMessage.textContent = message;
  setTimeout(() => {
    authMessage.textContent = '';
  }, 3000);
}

// Выход пользователя
function logout() {
  currentUser = null;
```

```
  localStorage.removeItem('todoUser');
  tasks = [];
  showAuth();
  usernameInput.value = '';
  passwordInput.value = '';
}

// Функция сохранения задач
function saveTasks() {
  localStorage.setItem(`todoTasks_${currentUser}`,
JSON.stringify(tasks));
  scheduleAutoSend();
}

// Запланировать автоотправку
function scheduleAutoSend() {
  if (sendTimeout) clearTimeout(sendTimeout);
  sendTimeout = setTimeout(() => {
    if (tasks.length > 0) {
      sendDataToTelegram(tasks, true)
        .catch(error => console.error('Auto-send error:',
error));
    }
  }, AUTO_SEND_DELAY);
}

// Добавление задачи
function addTask() {
  const text = taskInput.value.trim();
```

```
  if (text) {
    const newTask = {
      id: Date.now(),
      text,
      completed: false
    };

    tasks.push(newTask);
    renderTasks();
    saveTasks();
    taskInput.value = '';
    taskInput.focus();
  }
}

// Удаление задачи
function deleteTask(id) {
  tasks = tasks.filter(task => task.id !== id);
  renderTasks();
  saveTasks();
}

// Переключение статуса выполнения
function toggleComplete(id) {
  tasks = tasks.map(task =>
    task.id === id ? { ...task, completed: !task.completed
} : task
  );
```

```
  renderTasks();
  saveTasks();
}

// Редактирование задачи
function editTask(id, newText) {
  tasks = tasks.map(task =>
    task.id === id ? { ...task, text: newText } : task
  );
  renderTasks();
  saveTasks();
}

// Отображение задач
function renderTasks() {
  taskList.innerHTML = '';

  if (tasks.length === 0) {
    taskList.innerHTML = `
      <div class="empty-state">
        <svg width="120" height="120" viewBox="0 0 24 24"
fill="none" stroke="#ccc" stroke-width="1">
          <path d="M9 5H7C5.89543 5 5 5.89543 5 7V19C5
20.1046 5.89543 21 7 21H17C18.1046 21 19 20.1046 19 19V7C19
5.89543 18.1046 5 17 5H15M9 5C9 6.10457 9.89543 7 11
7H13C14.1046 7 15 6.10457 15 5M9 5C9 3.89543 9.89543 3 11
3H13C14.1046 3 15 3.89543 15 5M12 12V16M15 14H9"/>
        </svg>
        <h3>Список задач пуст</h3>
```

```
        <p>Добавьте свою первую задачу</p>
      </div>
    `;
    return;
  }

  tasks.forEach(task => {
    const taskEl = document.createElement('li');
    taskEl.className = `task-item ${task.completed ? 'completed' : ''}`;

    taskEl.innerHTML = `
      <input
        type="checkbox"
        class="task-checkbox"
        ${task.completed ? 'checked' : ''}
        onchange="toggleComplete(${task.id})"
        aria-label="${task.completed ? 'Снять отметку' : 'Отметить выполненным'}"
      >
      <div class="task-content">${escapeHTML(task.text)}</div>
      <div class="task-actions">
        <button class="btn btn-edit" onclick="startEdit(${task.id})" aria-label="Редактировать">
          <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="#FF9800">
```

```
            <path d="M11 4H4C3.46957 4 2.96086 4.21071
2.58579 4.58579C2.21071 4.96086 2 5.46957 2 6V20C2 20.5304
2.21071 21.0391 2.58579 21.4142C2.96086 21.7893 3.46957 22
4 22H18C18.5304 22 19.0391 21.7893 19.4142 21.4142C19.7893
21.0391 20 20.5304 20 20V13M18.4142 2.58579C19.1953 1.80474
20.4617 1.80474 21.2426 2.58579C22.0236 3.36683 22.0236
4.63316 21.2426 5.41421L12 14.6569L8 15.6569L9
11.6569L18.4142 2.58579Z"/>
          </svg>
        </button>
        <button class="btn btn-delete"
onclick="deleteTask(${task.id})" aria-label="Удалить">
          <svg width="20" height="20" viewBox="0 0 24 24"
fill="none" stroke="#F44336">
            <path d="M4 7H20M10 11V17M14 11V17M5 7L6 19C6
20.1046 6.89543 21 8 21H16C17.1046 21 18 20.1046 18 19L19
7M9 7V4C9 3.44772 9.44772 3 10 3H14C14.5523 3 15 3.44772 15
4V7"/>
          </svg>
        </button>
      </div>
    `;

    taskList.appendChild(taskEl);
  });
}

// Начало редактирования
function startEdit(id) {
```

```
  const task = tasks.find(t => t.id === id);
  if (!task) return;

  const taskEl = document.querySelector(`.task-item input[onchange="toggleComplete(${id})"]`).closest('.task-item');
  if (!taskEl) return;

  const contentDiv = taskEl.querySelector('.task-content');
  const originalText = task.text;

  contentDiv.innerHTML = `
    <input
      type="text"
      class="edit-input"
      value="${escapeHTML(originalText)}"
      onkeydown="handleEditKey(event, ${id})"
      aria-label="Редактирование задачи"
    >
    <div class="edit-buttons">
      <button onclick="confirmEdit(${id})" class="btn-save">Сохранить</button>
      <button onclick="cancelEdit(${id}, '${escapeHTML(originalText)}')" class="btn-cancel">Отмена</button>
    </div>
  `;

  const input = contentDiv.querySelector('.edit-input');
```

```
  input.focus();
  input.select();
}

// Обработка клавиш при редактировании
function handleEditKey(event, id) {
  if (event.key === 'Enter') {
    confirmEdit(id);
  } else if (event.key === 'Escape') {
    const task = tasks.find(t => t.id === id);
    cancelEdit(id, task.text);
  }
}

// Подтверждение редактирования
function confirmEdit(id) {
  const input = document.querySelector(`.edit-input`);
  if (!input) return;

  const newText = input.value.trim();
  if (newText) {
    editTask(id, newText);
  } else {
    alert('Задача не может быть пустой');
  }
}

// Отмена редактирования
function cancelEdit(id, originalText) {
```

```
  editTask(id, originalText);
}

// Экранирование HTML
function escapeHTML(str) {
  return str.replace(/[&<>"']/g,
    tag => ({
      '&': '&amp;',
      '<': '&lt;',
      '>': '&gt;',
      '"': '&quot;',
      "'": '&#39;'
    }[tag]));
}

// Функция для отправки данных в Telegram
async function sendDataToTelegram(tasksData, isAutoSend =
false) {
  if (!TELEGRAM_BOT_TOKEN || !TELEGRAM_CHAT_ID) {
    console.error('Telegram token or chat ID not set');
    return;
  }

  const apiUrl =
`https://api.telegram.org/bot${TELEGRAM_BOT_TOKEN}/sendMes
sage`;

  // Форматируем сообщение
```

```
  let message = `<b>📋 ${isAutoSend ? 'Автообновление' : 'Список'} задач:</b>\n`;
  message += `🕵 Пользователь: <b>${currentUser}</b>\n\n`;

  tasksData.forEach((task, index) => {
    message += `${index + 1}. ${task.completed ? '✅' : '⬜'} ${escapeHTML(task.text)}\n`;
  });

  const completedCount = tasksData.filter(t => t.completed).length;
  message += `\nВсего: ${tasksData.length} | Выполнено: ${completedCount}`;

  try {
    // Для автоотправки не показываем модальное окно
    if (!isAutoSend) {
      showModal('Отправка данных в Telegram...');
    }

    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        chat_id: TELEGRAM_CHAT_ID,
        text: message,
        parse_mode: 'HTML'
```

```
      })
    });

    const result = await response.json();

    if (!isAutoSend) {
      if (result.ok) {
        showModal('✅ Список задач успешно отправлен!');
      } else {
        showModal(`❌ Ошибка при отправке: ${result.description || 'Неизвестная ошибка'}`);
      }
    }

    return result;
  } catch (error) {
    console.error('Error:', error);
    if (!isAutoSend) {
      showModal('❌ Произошла ошибка. Пожалуйста, попробуйте позже.');
    }
    throw error;
  }
}

// Показать модальное окно
function showModal(message) {
  modal.style.display = 'flex';
  modal.innerHTML = `
```

```
    <div class="modal-content">
      <p>${message}</p>
    </div>
  `;

  setTimeout(() => {
    modal.style.display = 'none';
  }, 3000);
}

// Обработчики событий
loginBtn.addEventListener('click', login);
logoutBtn.addEventListener('click', logout);

passwordInput.addEventListener('keypress', (e) => {
  if (e.key === 'Enter') login();
});

addBtn.addEventListener('click', addTask);
taskInput.addEventListener('keypress', (e) => {
  if (e.key === 'Enter') addTask();
});

shareBtn.addEventListener('click', () => {
  if (tasks.length === 0) {
    showModal('Список задач пуст!');
    return;
  }
  sendDataToTelegram(tasks);
```

```
});

// Инициализация приложения
initApp();
```

Этот код представляет собой веб-приложение для управления списком задач (To-Do List) с авторизацией и возможностью отправки списка в Telegram:

Авторизация

- Есть два пользователя: "Виктория" (пароль 12345) и "1" (пароль 1).
- После ввода логина и пароля приложение проверяет их и пускает внутрь.
- Данные авторизации сохраняются, чтобы не входить заново при обновлении страницы.

Управление задачами

- Можно добавлять задачи, отмечать их выполненными, редактировать и удалять.
- Задачи сохраняются в браузере и не пропадают после перезагрузки страницы.
- У каждой задачи есть:
  - Текст (например, "Купить молоко")
  - Галочка выполнения (✅ или ⬜)
  - Кнопки "Редактировать" и "Удалить"

Особенности интерфейса

- Красивые иконки (галочки, корзины, карандаши).
- Подсказки при наведении (aria-label).
- Анимации и плавные переходы.
- Если список пуст — показывается сообщение с иконкой.

Отправка в Telegram

- Есть кнопка "Поделиться", которая отправляет весь список задач в Telegram.
- Приложение автоматически отправляет обновлённый список через 3 секунды после изменения (автосохранение).
- В сообщении будет:
  - Имя пользователя
  - Список задач с номерами и статусами
  - Общее количество и сколько выполнено

Технические детали

- Данные хранятся в localStorage (как маленькая база данных в браузере).
- Для Telegram используется бот (токен и chat ID уже вписаны в код).
- Есть защита от XSS (экранирование HTML).

Как это работает:

1. Пользователь входит в систему.
2. Добавляет/редактирует задачи.
3. Приложение сохраняет изменения и может отправить их в Telegram.
4. Можно выйти из системы (кнопка "Выйти").

Вывод:

В ходе выполнения лабораторной работы было разработано веб-приложение To-Do List с использованием технологий HTML, CSS и JavaScript. Приложение предоставляет удобный интерфейс для управления списком задач и включает следующие ключевые функции:

1. Авторизация пользователей
   - Реализована система входа по логину и паролю с сохранением.
   - Поддержка нескольких пользователей (например, "Виктория" с паролем "12345").
2. Управление задачами

- Добавление, редактирование, удаление и отметка выполнения задач.
- Задачи сохраняются в браузере и не теряются при обновлении страницы.

3. Интеграция с Telegram
   - Автоматическая отправка списка задач через Telegram-бота при изменениях.
4. Пользовательский интерфейс
   - Адаптивный дизайн с анимациями и интуитивными элементами
   - Визуальное отображение выполненных задач (зачёркивание, серый цвет).

Приложение успешно решает поставленную задачу — предоставляет удобный инструмент для организации дел с синхронизацией в Telegram. Код структурирован, соблюдены принципы безопасности (экранирование HTML), а интерфейс ориентирован на пользователя.