**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**Факультет: Информационные технологии и управление**

**Кафедра: Компьютерная инженерия**

**Предмет: «Веб-системы и технологии»**

**Курсовая работа**

**Тема:** «**Система регистрации студентов**»

**Группа:** 680.22

**Курc:** 3

**Специальность: 050616** Информационные технологии

**Студент**: Зейналова Фидан

**Руководитель:** **асс. Халилов М. Э.**

**Зав.кафедрой: доц.Рагимова Н.А.**

**Баку – 2025**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ**

**ФАКУЛЬТЕТ**: Информационные технологии и управление

**КАФЕДРА**: Компьютерная инженерия

**Группа** 680.22 **курс** III

**Специальность** **050616** Информационные технологии

**Студент** Зейналова Фидан

**Зачетная книжка**

**Руководитель курсовой работы**  **асс. Халилов М. Э.**

**Срок выдачи**

**Дата сдачи**

**Тема курсовой работы** «**Система регистрации студентов**»

**Отзыв руководителя курсовой**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись студента**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись рук.курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись зав.кафедрой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата защиты курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Оценка**\_\_\_\_\_\_\_\_\_\_

**Пред.комиссии**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Члены комиссии**: 1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc196758474)

[**1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ** 5](#_Toc196758475)

[**1.1 ВЕБ-ПРОГРАММИРОВАНИЕ** 5](#_Toc196758476)

[**1.2 HTML** 6](#_Toc196758477)

[**1.3 CSS** 9](#_Toc196758478)

[**1.4 JavaScript** 11](#_Toc196758479)

[**2.ПРАКТИЧЕСКАЯ ЧАСТЬ** 13](#_Toc196758480)

[**2.1 Описание предметной области** 13](#_Toc196758481)

[**2.2 Основные компоненты HTML-документа** 14](#_Toc196758482)

[**2.2.1 index.html** 14](#_Toc196758483)

[**2.2.2 about.html** 17](#_Toc196758484)

[**2.2.3 contact.html** 18](#_Toc196758485)

[**2.3 CSS стили для сайта** 19](#_Toc196758486)

[**2.4 JavaScript для функционала** 28](#_Toc196758487)

[**3.РЕЗУЛЬТАТ** 32](#_Toc196758488)

[**4. ЛИТЕРАТУРА** 35](#_Toc196758489)

# 

# **ВВЕДЕНИЕ**

В современном мире информационные технологии стремительно развиваются и становятся неотъемлемой частью повседневной жизни.

Одной из ключевых областей является **веб-программирование** — создание сайтов и веб-приложений, доступных через интернет.

Веб-технологии охватывают широкий спектр задач: от простого отображения текстов и изображений до построения сложных, интерактивных платформ, обеспечивающих непрерывное взаимодействие пользователей с сервисами и информационными системами.

Развитие интернета началось с простых HTML-документов, но с течением времени веб стал более динамичным и адаптивным. Сегодняшние пользователи ожидают мгновенного отклика интерфейса, удобной навигации, безопасности передачи данных и доступности на различных устройствах — от мобильных телефонов до широкоформатных мониторов. Для решения этих задач разработчикам необходимо владеть широким набором технологий, включая HTML, CSS и JavaScript, а также понимать основы серверной разработки и работы с базами данных.

Проект **"Система регистрации студентов"** является примером практического применения базовых знаний в области веб-разработки. Он направлен на упрощение процесса регистрации студентов в образовательных учреждениях, предоставляя удобный интерфейс для ввода и обработки данных. В рамках проекта реализованы три веб-страницы, оформленные с использованием каскадных таблиц стилей (CSS) и дополненные динамическими функциями через язык JavaScript.

Данный проект позволяет продемонстрировать базовые принципы клиентской веб-разработки, такие как создание структурированной HTML-разметки, оформление визуальных элементов с помощью CSS и обеспечение интерактивности через обработку событий в JavaScript.

# **1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1 ВЕБ-ПРОГРАММИРОВАНИЕ**

**Веб-программирование —** это процесс создания сайтов и веб-приложений, которые работают через интернет и строятся на основе различных технологий. Сегодняшние веб-ресурсы выходят далеко за рамки простого отображения текста и картинок: они становятся динамичными, интерактивными и легко подстраиваются под разные устройства. В основе современной веб-разработки лежат такие технологии, как HTML, CSS, JavaScript, а также серверные языки программирования и базы данных.

**Ключевые элементы веб-разработки**

**1. Frontend (клиентская часть)**Фронтенд отвечает за всё, что пользователь видит и с чем взаимодействует на сайте. С помощью HTML создаётся структура страницы, CSS задаёт её стиль, а JavaScript добавляет жизнь — анимацию, интерактивные формы и динамическое обновление контента без перезагрузки. Большое значение уделяется адаптивности интерфейсов, чтобы сайт выглядел и работал отлично на любом устройстве, от смартфона до большого монитора.

**2. Backend (серверная часть)**Бэкенд скрыт от глаз пользователей, но именно он делает сайт «умным». Серверная логика обрабатывает запросы клиентов, работает с данными и управляет бизнес-процессами. Для этого используются языки программирования вроде Python, PHP, Java или Node.js. Сервер отправляет необходимые данные на клиентскую сторону, обеспечивает безопасность и правильную работу приложений.

**3. Базы данных**Ни одно серьёзное веб-приложение не обходится без хранения данных. Базы данных позволяют сохранять информацию о пользователях, заказах, товарах и многом другом. В зависимости от задач используются как классические реляционные базы (например, MySQL, PostgreSQL), так и современные NoSQL-решения вроде MongoDB. С помощью SQL-запросов разработчики получают, обновляют и удаляют данные.

Сегодня динамичность — неотъемлемая часть веб-разработки. Технологии типа AJAX и популярные фреймворки, такие как React и Angular, позволяют обновлять содержимое страницы на лету, не заставляя пользователя ждать перезагрузки.

С ростом мобильного трафика адаптивность стала обязательным требованием. Современный сайт обязан корректно выглядеть и работать на смартфонах, планшетах и десктопах. А ещё важно помнить об универсальной доступности — создавать сайты, которые могут использовать люди с ограниченными возможностями, включая поддержку специальных технологий управления.

## **1.2 HTML**

**HTML (HyperText Markup Language)** — это язык разметки, используемый для создания и структурирования веб-страниц. Представьте, что веб-страница — это книга. Тогда HTML — это её каркас, определяющий, где будет заголовок, абзацы, изображения и другие элементы.

**Основы разметки**

HTML-документ состоит из набора тегов, которые окружают контент и определяют его назначение. Эти теги интерпретируются веб-браузерами для отображения страницы.

**Пример базовой структуры HTML-документа:**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Моя первая страница</title>

</head>

<body>

<h1>Заголовок</h1>

<p>Это параграф текста.</p>

</body>

</html>

**Основные элементы HTML**

1. <!DOCTYPE html>

Определяет, что документ использует HTML5.

2. <html>

Корневой элемент, внутри которого находятся все остальные теги.

3. <head>

Содержит метаданные о документе (кодировку, заголовок, стили, скрипты).

4. <body>

Содержит весь видимый контент веб-страницы.

**Теги и атрибуты**

HTML состоит из множества тегов, которые помогают организовать информацию. Рассмотрим основные из них:

**1. Заголовки (<h1> - <h6>)**

Используются для создания заголовков различных уровней.

<h1>Заголовок 1</h1>

<h2>Заголовок 2</h2>

<h3>Заголовок 3</h3>

**2. Параграфы (<p>)**

Применяются для разделения текстового контента на абзацы.

<p>Это первый абзац.</p>

<p>Это второй абзац.</p>

**3. Списки**

Нумерованные (<ol>) и ненумерованные (<ul>) списки:

<ul>

<li>Элемент списка 1</li>

<li>Элемент списка 2</li>

</ul>

<ol>

<li>Первый элемент</li>

<li>Второй элемент</li>

</ol>

**4. Ссылки (<a>)**

Используются для создания гиперссылок.

<a href="https://example.com">Перейти на сайт</a>

**5. Изображения (<img>)**

Добавляют картинки на страницу.

<img src="image.jpg" alt="Описание изображения">

**6. Таблицы (<table>)**

Структурируют данные в виде таблиц.

<table border="1">

<tr>

<th>Заголовок 1</th>

<th>Заголовок 2</th>

</tr>

<tr>

<td>Ячейка 1</td>

<td>Ячейка 2</td>

</tr>

</table>

**Формы и элементы ввода**

Формы используются для сбора информации от пользователей.

<form action="submit.php" method="post">

<label for="name">Имя:</label>

<input type="text" id="name" name="name">

<br>

<label for="email">Email:</label>

<input type="email" id="email" name="email">

<br>

<button type="submit">Отправить</button>

</form>

## **1.3 CSS**

**CSS (Cascading Style Sheets)** — это язык таблиц стилей, который используется для оформления внешнего вида веб-страниц. Если представить, что HTML — это скелет страницы, то CSS — это её одежда и макияж, задающие цвета, шрифты, отступы, размеры и прочее визуальное оформление.

**Основы CSS**

CSS позволяет разделить структуру страницы (HTML) и её визуальное представление. Правила CSS состоят из селекторов и наборов свойств, которые определяют, как элементы должны выглядеть.

Пример базового CSS-кода:

<style>

body {

background-color: #f0f0f0;

font-family: Arial, sans-serif;

}

h1 {

color: darkblue;

}

p {

font-size: 16px;

line-height: 1.5;

}

</style>

В этом примере задаются цвет фона страницы, стиль шрифта для текста, цвет заголовка и параметры для параграфов.

**Основные концепции CSS**

1. **Селекторы:** Селекторы указывают, к каким элементам на странице применяется стиль. Примеры:
2. p { color: red; } /\* Все параграфы будут красными \*/
3. #header { background-color: blue; } /\* Элемент с id="header" будет синим \*/
4. .menu-item { font-weight: bold; } /\* Все элементы с классом menu-item будут жирными \*/
5. **Свойства и значения:** Каждое CSS-правило состоит из свойства (что изменить) и значения (на что изменить). Пример:
6. p {
7. font-size: 18px; /\* Свойство: размер шрифта, Значение: 18 пикселей \*/
8. }
9. **Каскадность** Если к одному элементу применяются несколько стилей, браузер выбирает тот, который имеет более высокий приоритет. Например, стили, заданные напрямую в HTML через атрибут style, переопределяют внешние таблицы стилей.
10. **Наследование** Некоторые свойства, например цвет текста или семейство шрифтов, передаются от родительского элемента к дочерним автоматически**.**

**Важные возможности CSS**

1. **Работа с макетами  
   Современные макетные технологии позволяют легко размещать элементы на странице.**

Пример Flexbox:

.container {

display: flex;

justify-content: space-between;

}

1. **Медиа-запросы (адаптивный дизайн)**

С помощью медиа-запросов страница может менять стиль в зависимости от размера экрана устройства.

Пример:

@media (max-width: 600px) {

body {

background-color: lightgray;

}

}

1. **Анимации и переходы CSS**

позволяет создавать плавные анимации и переходы между состояниями элементов.

Пример:

button {

transition: background-color 0.3s;

}

button:hover {

background-color: green;

}

**Подключение CSS к HTML**

Существует три способа применения CSS к веб-странице:

* Встроенный стиль (через атрибут style внутри HTML-элемента):
* <p style="color: red;">Текст</p>
* Внутренний стиль (внутри тега <style> в разделе <head>):
* <style>
* p { color: blue; }
* </style>
* Внешний файл стилей (рекомендуемый способ):
* <link rel="stylesheet" href="styles.css">

## **1.4 JavaScript**

**JavaScript** — это язык программирования, используемый для создания интерактивных элементов на веб-страницах. Он позволяет динамически изменять содержимое страницы, обрабатывать действия пользователя и выполнять обмен данными с сервером без необходимости перезагрузки страницы.

**Основные возможности JavaScript**

1. **Обработка событий**  
   JavaScript позволяет реагировать на действия пользователя, такие как клики мыши, ввод данных в формы, перемещение курсора и другие события.

Пример:

<button onclick="alert('Кнопка нажата')">Нажать</button>

1. **Изменение структуры документа (DOM-манипуляции)**С помощью JavaScript можно изменять элементы страницы, добавлять новые или удалять существующие, а также изменять их свойства.

Пример:

<p id="text">Исходный текст</p>

<button onclick="document.getElementById('text').innerText = 'Текст изменён'">Изменить текст</button>

1. **Работа с формами**  
   JavaScript используется для валидации данных, введённых пользователями, до их отправки на сервер.

Пример:

<form onsubmit="return validateForm()">

<input type="text" id="name" placeholder="Введите имя">

<button type="submit">Отправить</button>

</form>

<script>

function validateForm() {

var name = document.getElementById('name').value;

if (name === '') {

alert('Поле "Имя" не должно быть пустым.');

return false;

}

return true;

}

</script>

**Базовые концепции JavaScript.**

**Переменные и типы данных:**  
JavaScript поддерживает различные типы данных: числа, строки, булевы значения, объекты, массивы и другие.

Пример:

let message = 'Текстовое сообщение';

let number = 100;

**Функции**  
Функции позволяют организовывать код в отдельные блоки для многократного использования.

Пример:

function greet(user) {

console.log('Здравствуйте, ' + user);

}

greet('Алексей');

**Условные операторы и циклы**  
JavaScript позволяет выполнять действия в зависимости от условий и повторять их заданное количество раз.

Пример:

if (isLoggedIn) {

console.log('Добро пожаловать');

} else {

console.log('Пожалуйста, авторизуйтесь');

}

for (let i = 0; i < 5; i++) {

console.log(i);

}

**Подключение JavaScript**

Скрипты JavaScript могут быть размещены:

* Внутри HTML-файла с использованием тега <script>.
* Во внешнем файле с последующим подключением через тег <script src="путь\_к\_файлу.js"></script>.

Рекомендуется размещать подключения скриптов в нижней части HTML-документа, перед закрывающим тегом </body>, чтобы сначала загружался основной контент страницы.

# 

# **2.ПРАКТИЧЕСКАЯ ЧАСТЬ**

## **2.1 Описание предметной области**

Проект представляет собой веб-приложение под названием **«Система регистрации студентов»,** предназначенное для упрощения процесса регистрации студентов и управления их данными в образовательных учреждениях.

Основной целью системы является автоматизация ввода и сбора информации о студентах, а также обеспечение удобного канала связи между пользователями и администрацией.

**Функциональные возможности сайта включают:**

* Регистрацию новых пользователей через форму на главной странице.
* Предоставление информации о назначении и возможностях системы на отдельной странице «О системе».
* Организацию канала обратной связи с администраторами через форму на странице «Контакты».

**Проект состоит из следующих компонентов:**

* **Три HTML-страницы**:
  + Главная страница (index.html) — регистрация пользователей и общее представление проекта.
  + Информационная страница (about.html) — описание системы и её ключевых функций.
  + Страница контактов (contact.html) — форма обратной связи.
* **Один файл стилей:** styles.css, отвечающий за оформление всех страниц сайта.
* **Один файл скриптов**: script.js, обеспечивающий обработку форм и взаимодействие с пользователем на стороне клиента.
* **Папка с изображениями**: images, содержащая графические материалы для визуального оформления страниц.

На текущем этапе реализации взаимодействие осуществляется только на стороне клиента, без интеграции серверной части или базы данных. В дальнейшем можно расширить функциональность за счёт серверной обработки данных, аутентификации пользователей и подключения базы данных для хранения информации.

## **2.2 Основные компоненты HTML-документа**

### **2.2.1 index.html**

**Шапка страницы с навигацией:**

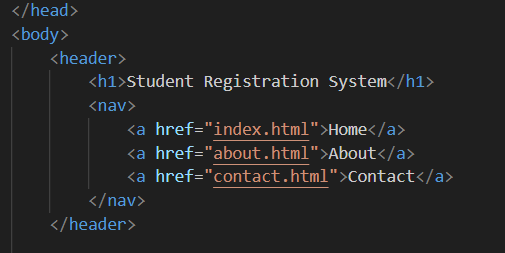


Рис.1

Эта часть создает шапку сайта с заголовком и навигационным меню, которое содержит ссылки на все три страницы сайта.

**Секция-герой:**

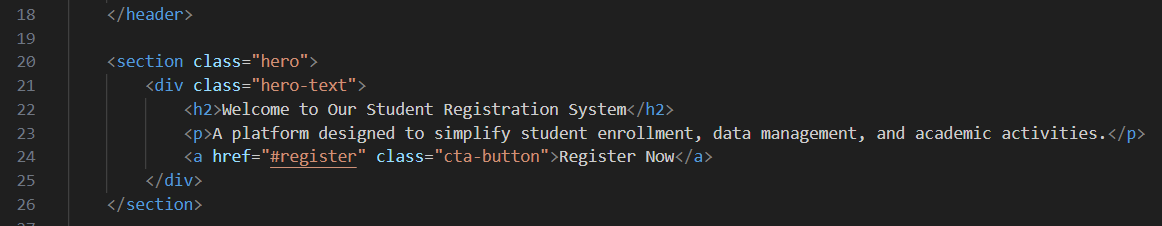


Рис.2

Эта секция создает привлекательный баннер с заголовком, кратким описанием и кнопкой призыва к действию, которая ведет к разделу регистрации ниже на странице.

**Секция регистрации:**

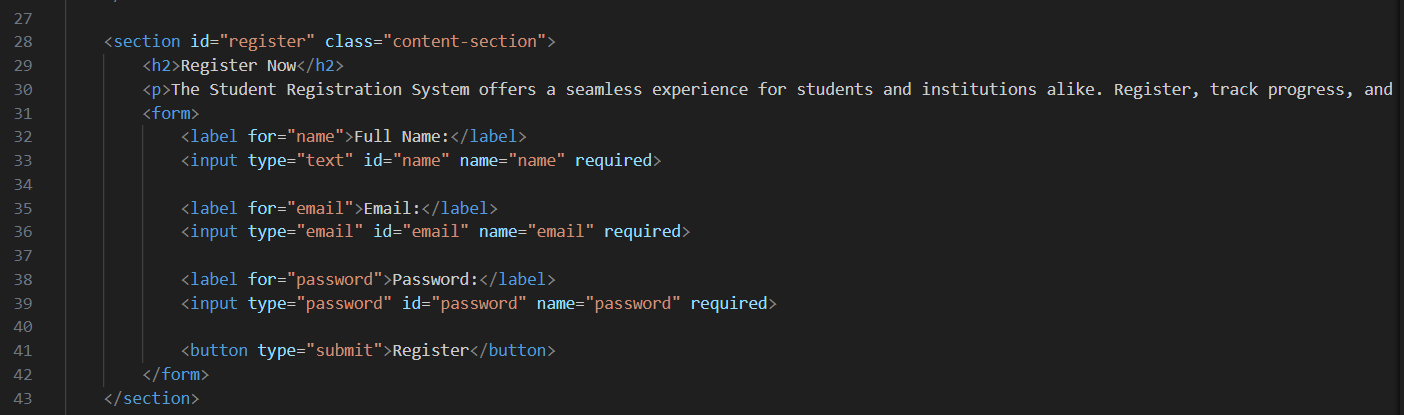


Рис.3

Этот раздел содержит форму регистрации с полями для ввода имени, email и пароля. Атрибут required обеспечивает базовую валидацию, требуя заполнения всех полей перед отправкой.

**Секция FAQ:**

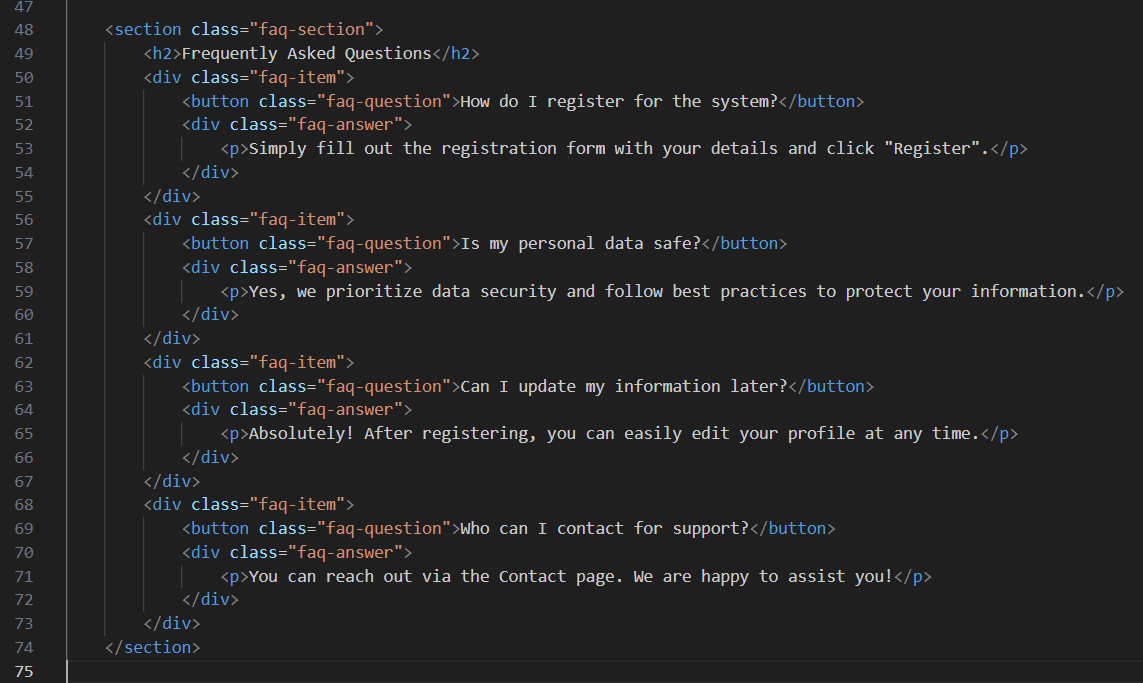


Рис.4

Здесь сделана секция с часто задаваемыми вопросами (FAQ). Каждый вопрос оформлен как кнопка (button), а ответ — как скрытый блок (div).  
Когда пользователь нажимает на вопрос, появляется соответствующий ответ.

**Подвал и подключение JavaScript:**

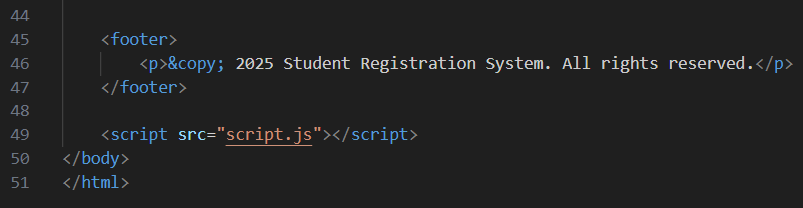


Рис.5

Подвал содержит информацию об авторских правах. В конце документа подключается файл JavaScript, который обрабатывает отправку форм.

### **2.2.2 about.html**

**Структура HTML-документа и шапка:**

Эта часть идентична структуре главной страницы, обеспечивая единообразие интерфейса, но с измененным заголовком вкладки браузера.

**Основное содержимое:**

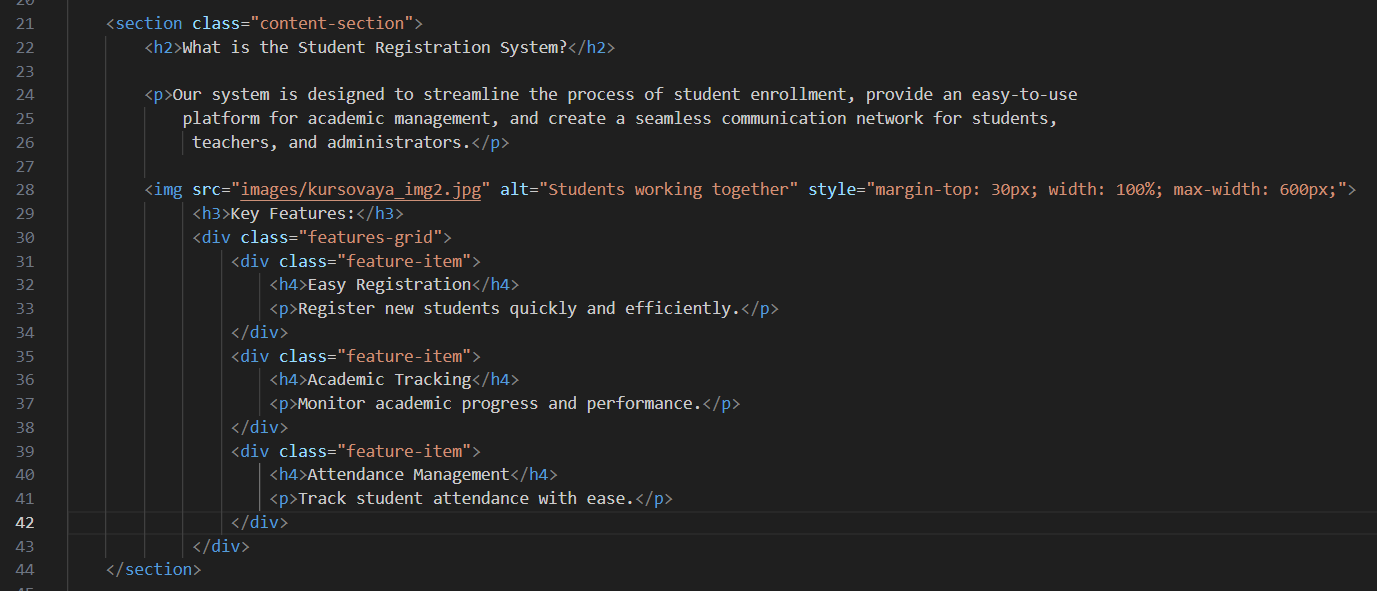


Рис.6

**Секция описания системы регистрации студентов**: короткий текст объясняет, зачем нужна система и какие задачи она решает.  
**Картинка** для визуального усиления и разбавки текста.  
**Блок с "Key Features"** — красиво разложены основные функции системы в виде сетки карточек (8 штук), каждая с заголовком и кратким описанием.

**Подвал:**

Подвал и подключение JavaScript аналогичны главной странице.

### **2.2.3 contact.html**

**Структура HTML-документа и шапка:**

Структура аналогична другим страницам, но с соответствующим заголовком страницы.

**Секция контактов:**

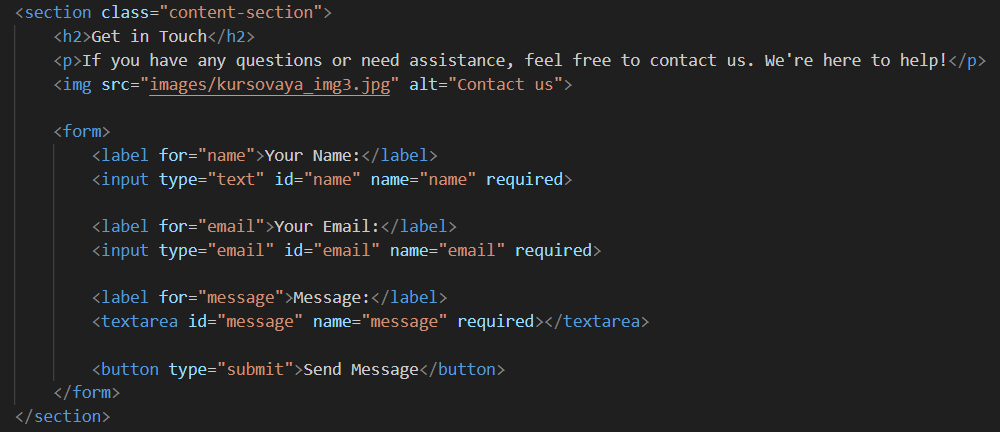


Рис.7

Эта секция содержит заголовок, краткий текст, изображение и форму обратной связи с полями для имени, email и текстового сообщения. В отличие от формы регистрации, здесь используется элемент <textarea> для многострочного ввода текста.

**Подвал:**

Подвал и подключение JavaScript идентичны другим страницам.

## **2.3 CSS стили для сайта**

**Общие стили для тела страницы:**



Рис.8

Устанавливает шрифт Arial, удаляет стандартные отступы, задает синий фон и добавляет отступ сверху для размещения фиксированной шапки.

**Стили для шапки:**

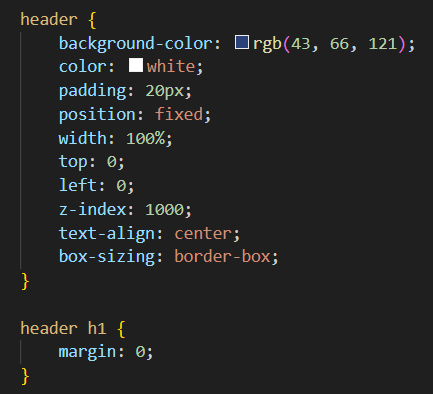


Рис.9

Шапка имеет темно-синий фон, белый текст и фиксированное позиционирование, чтобы оставаться видимой при прокрутке.

**Стили для навигации:**

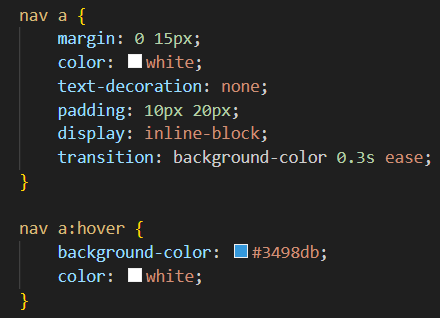


Рис.10

Ссылки навигации имеют отступы, белый цвет и эффект плавного изменения фона при наведении.

**Стили для контентных секций:**

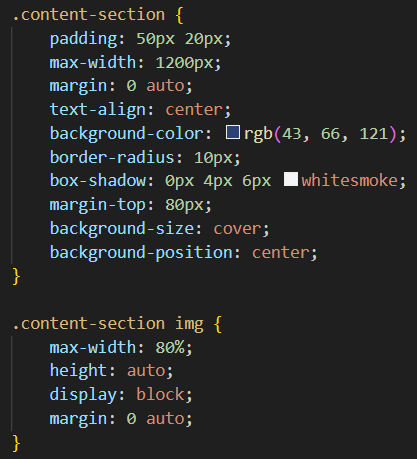


Рис.11

Контентные секции имеют отступы, ограниченную ширину, выравнивание по центру, темно-синий фон и тень. Изображения ограничены по ширине с сохранением пропорций.

**Стили для текстовых элементов:**

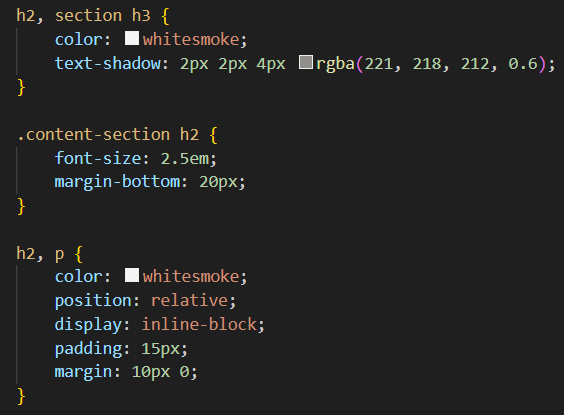


Рис.12

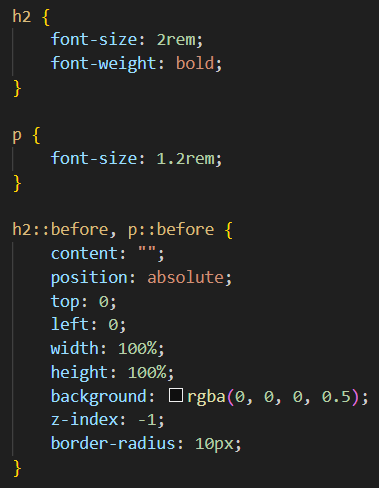


Рис.13

Заголовки и параграфы имеют светлый цвет и темный полупрозрачный фон, создаваемый с помощью псевдоэлемента ::before.

**Стили для FAQ:**

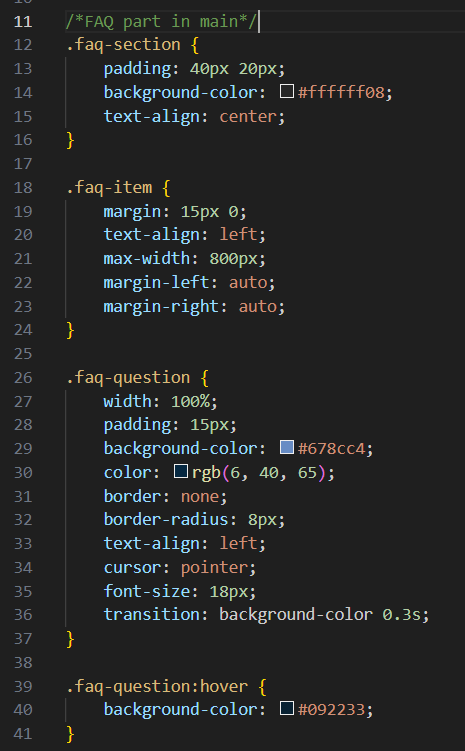


Рис.14

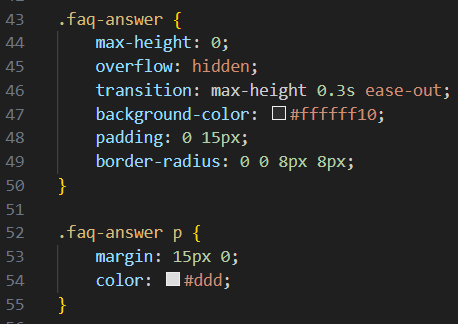


Рис.15

1. **.faq-section:** Добавляет отступы и полупрозрачный фон для всей секции с вопросами.
2. **.faq-item:** Центрирует каждый вопрос и ограничивает его ширину.
3. **.faq-question:** Кнопка с вопросом имеет синий фон, скругленные углы, и плавное изменение цвета при наведении.
4. **.faq-answer:** Ответы скрыты по умолчанию, плавно раскрываются с анимацией при клике, с полупрозрачным фоном.
5. **.faq-answer p:** Текст внутри ответов светло-серый, с отступами.

Когда кликаешь на вопрос, его ответ плавно раскрывается, создавая интерактивный и стильный элемент на странице.

**Стили для "Key Features":**

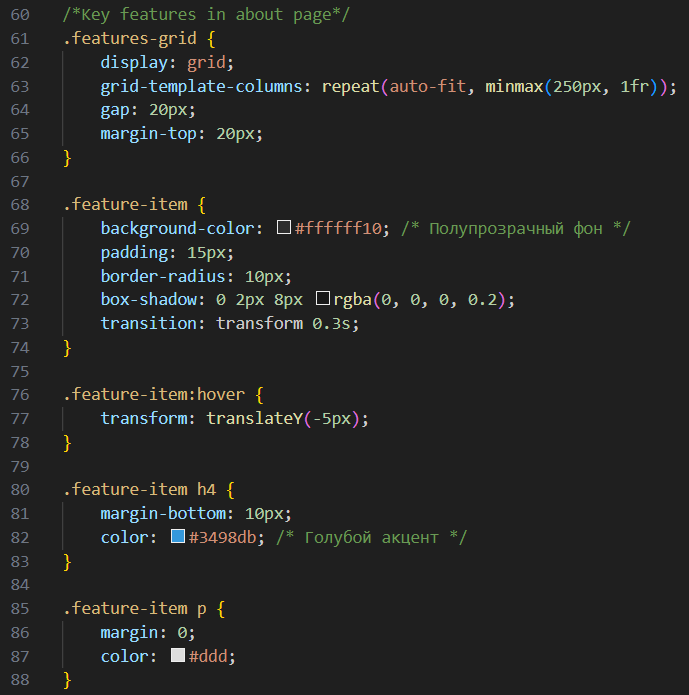


Рис.16

1. **.features-grid**: Создает сетку с элементами, которые автоматически адаптируются по размеру экрана (минимум 250px для каждого элемента) и имеют промежуток 20px между собой.
2. **.feature-item:** Каждый элемент имеет полупрозрачный фон, скругленные углы, небольшую тень для глубины и плавный эффект при наведении, который поднимет элемент вверх на 5px.
3. **.feature-item:hover:** Элемент слегка поднимается при наведении, что создает эффект интерактивности.
4. **.feature-item h4:** Заголовок внутри элемента получает голубой акцент и отступ снизу.
5. **.feature-item p:** Текст внутри элемента имеет светло-серый цвет и нулевой отступ.

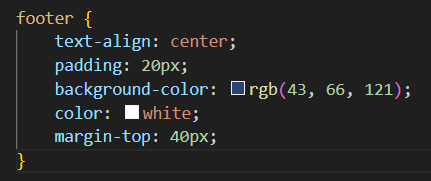
**Стили для подвала:**

Рис.17

Подвал имеет центрированный текст, отступы, темно-синий фон и светлый текст.

**Стили для секции-героя:**

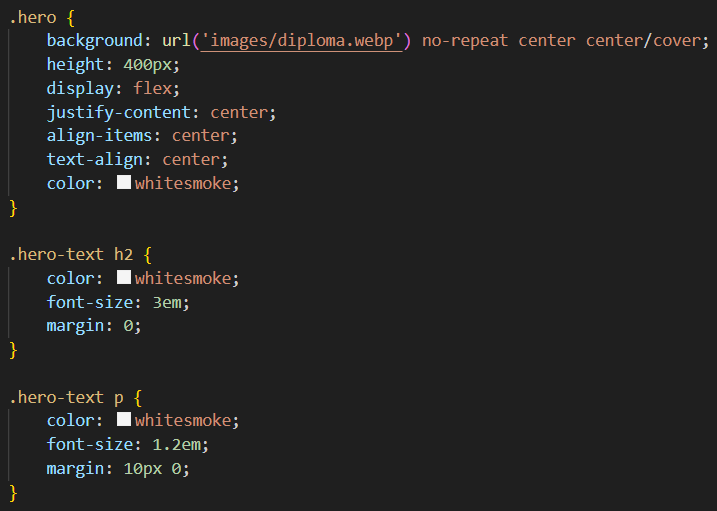


Рис.18

Секция-герой имеет фоновое изображение, фиксированную высоту и центрированное содержимое с использованием flexbox.

**Стили для кнопок:**

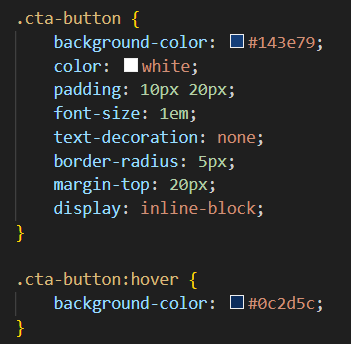


Рис.19

Кнопка призыва к действию имеет синий фон, белый текст, закругленные углы и эффект затемнения при наведении.

**Стили для форм:**

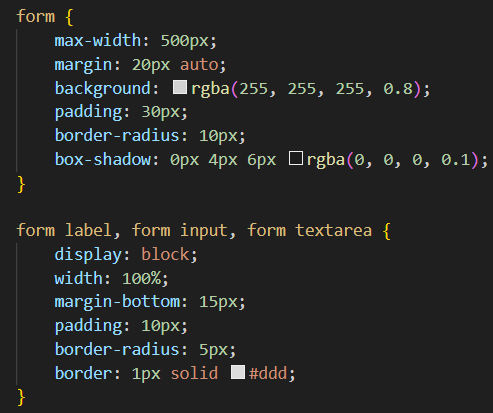


Рис.20

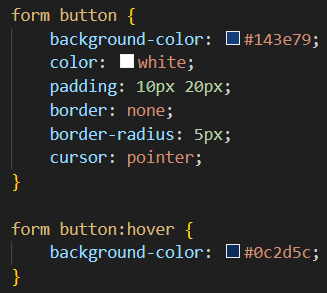


Рис.21

Формы имеют светлый полупрозрачный фон, тень и ограниченную ширину. Поля ввода занимают всю ширину и имеют отступы и закругленные углы. Кнопка отправки имеет стили, аналогичные другим кнопкам сайта.

## **2.4 JavaScript для функционала**

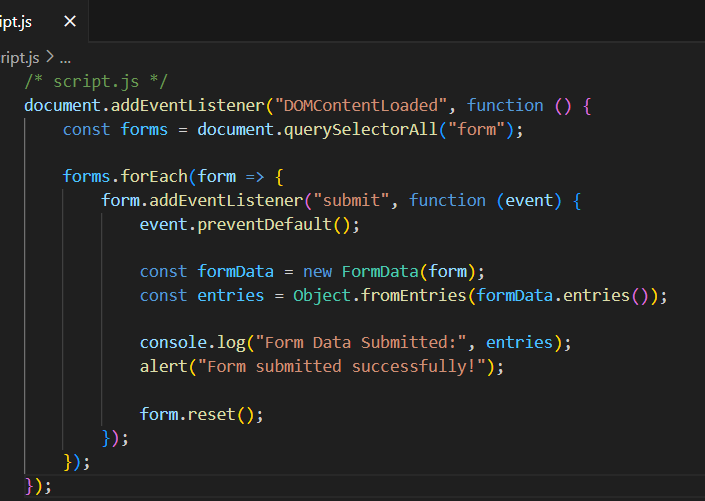


Рис.22

document.addEventListener("DOMContentLoaded", function () {

* Ждём, пока весь HTML загрузится, чтобы скрипт не начал искать формы до того, как они появятся в DOM.

const forms = document.querySelectorAll("form");

* Получаем **все элементы <form>** на странице.
* querySelectorAll возвращает **List**, похожий на массив, который можно перебирать.

forms.forEach(form => {

* Запускаем цикл по всем найденным формам. Даже если форм 10 — всё схвачено.

form.addEventListener("submit", function (event) {

* Навешиваем обработчик на **событие отправки** (submit) каждой формы.

event.preventDefault();

* Блокируем **стандартное поведение**: форма **не будет отправляться на сервер**, и страница **не будет перезагружаться**.
* Теперь всё в руках JavaScript'а.

const formData = new FormData(form);

* Создаём объект FormData, в котором автоматически собраны все <input>, <select>, <textarea> и т.п. из формы.
* FormData — это как корзина, в которую собрали всё, что ввёл пользователь.

const entries = Object.fromEntries(formData.entries());

* Преобразуем FormData в обычный объект JavaScript ({ ключ: значение }), чтобы было удобно работать.
* entries() возвращает пары ключ-значение, а Object.fromEntries() превращает их в объект.

console.log("Form Data Submitted:", entries);

* Выводим собранные данные в консоль. Типа: "Вот что заполнил пользователь".

alert("Form submitted successfully!");

* Показываем пользователю сообщение — чисто для обратной связи.

form.reset();

* Сбрасываем форму — все поля становятся пустыми.
* Можно сказать, форма "обнуляется".

JavaScript-код выполняет следующие действия:

1. Дожидается полной загрузки DOM-структуры страницы
2. Находит все формы на странице с помощью querySelectorAll
3. Для каждой формы добавляет обработчик события submit
4. При отправке формы:
   * Предотвращает стандартное поведение (перезагрузку страницы) с помощью event.preventDefault()
   * Создает объект FormData для сбора данных формы
   * Преобразует данные в объект JavaScript с помощью Object.fromEntries
   * Выводит собранные данные в консоль для отладки
   * Показывает пользователю сообщение об успешной отправке
   * Сбрасывает форму для нового ввода

Этот код работает одинаково для всех форм на сайте, как для формы регистрации на главной странице, так и для формы обратной связи на странице контактов.

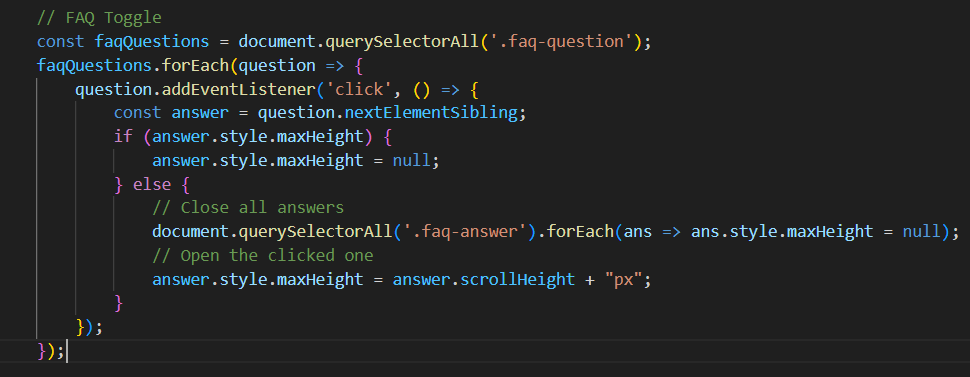


Рис.23

const faqQuestions = document.querySelectorAll('.faq-question');

* Эта строка находит все элементы с классом .faq-question (которые являются кнопками для вопросов) и сохраняет их в переменную faqQuestions в виде коллекции List.

faqQuestions.forEach(question => {

* Цикл, который перебирает все элементы в коллекции faqQuestions. Для каждого вопроса будет выполняться функция, которая принимает один элемент question в качестве параметра.

question.addEventListener('click', () => {

* Для каждого вопроса добавляется обработчик события, который срабатывает при клике. Когда пользователь кликает на вопрос, выполняется функция.

const answer = question.nextElementSibling;

* Находит следующий элемент после question, который в нашем случае является элементом с ответом (то есть .faq-answer). Сохраняет его в переменную answer.

if (answer.style.maxHeight) {

answer.style.maxHeight = null;

}

* Проверяет, если у ответа уже установлена высота (maxHeight), то она сбрасывается (делаем скрытым ответ).

else {

* Если же у ответа нет заданной высоты (то есть он скрыт), то выполняется следующий код.

// Close all answers

document.querySelectorAll('.faq-answer').forEach(ans => ans.style.maxHeight = null);

* Эта строка закрывает все ответы на других вопросах, устанавливая их maxHeight в null, чтобы они скрылись.

// Open the clicked one

answer.style.maxHeight = answer.scrollHeight + "px";

* Открывает только тот ответ, по которому был произведен клик. answer.scrollHeight — это полная высота содержимого элемента (включая скрытые части), и мы задаем maxHeight на основе этой высоты, чтобы ответ плавно раскрылся.

# **3.РЕЗУЛЬТАТ**

**Страница: Home**

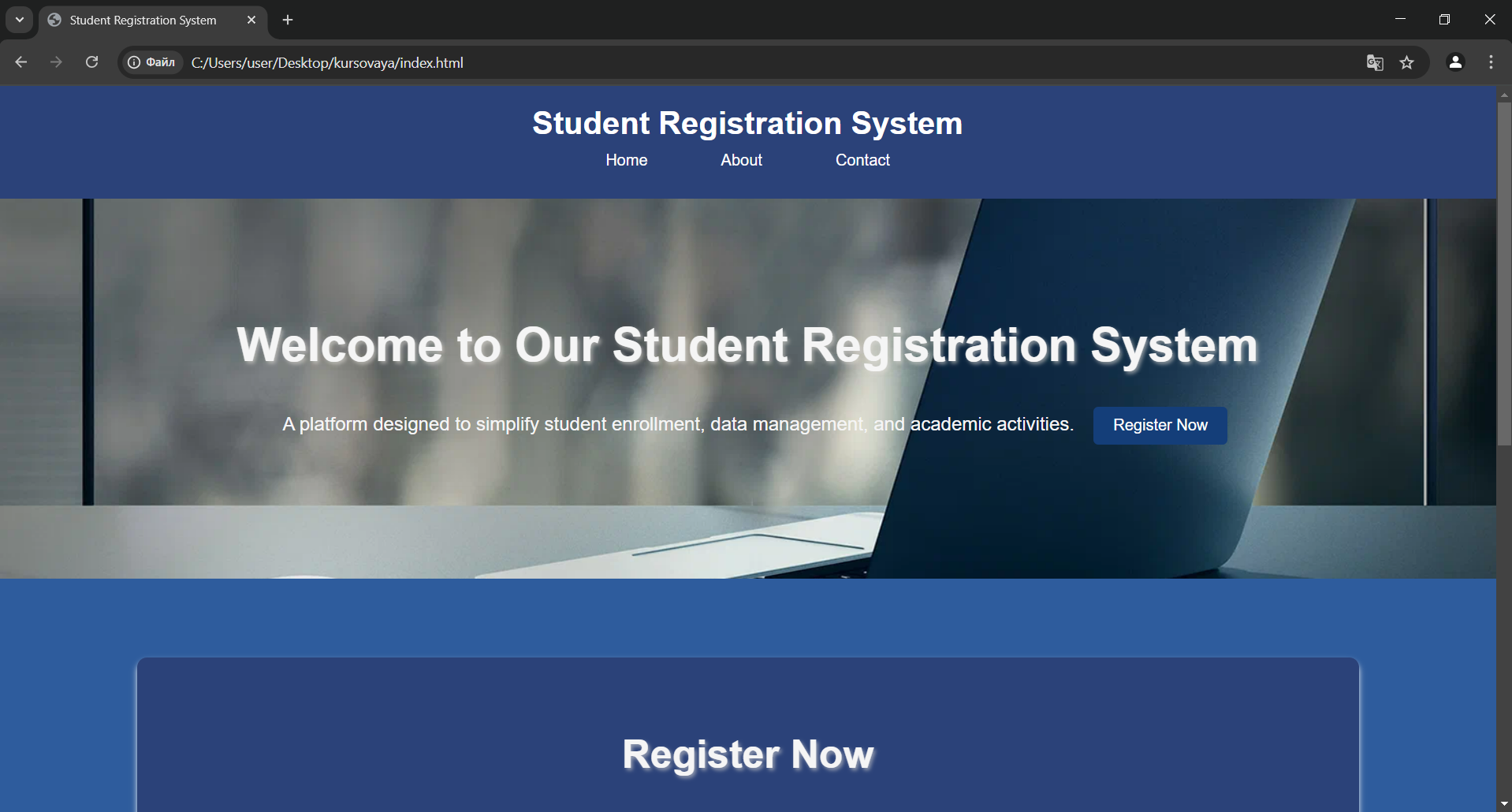


Рис.24

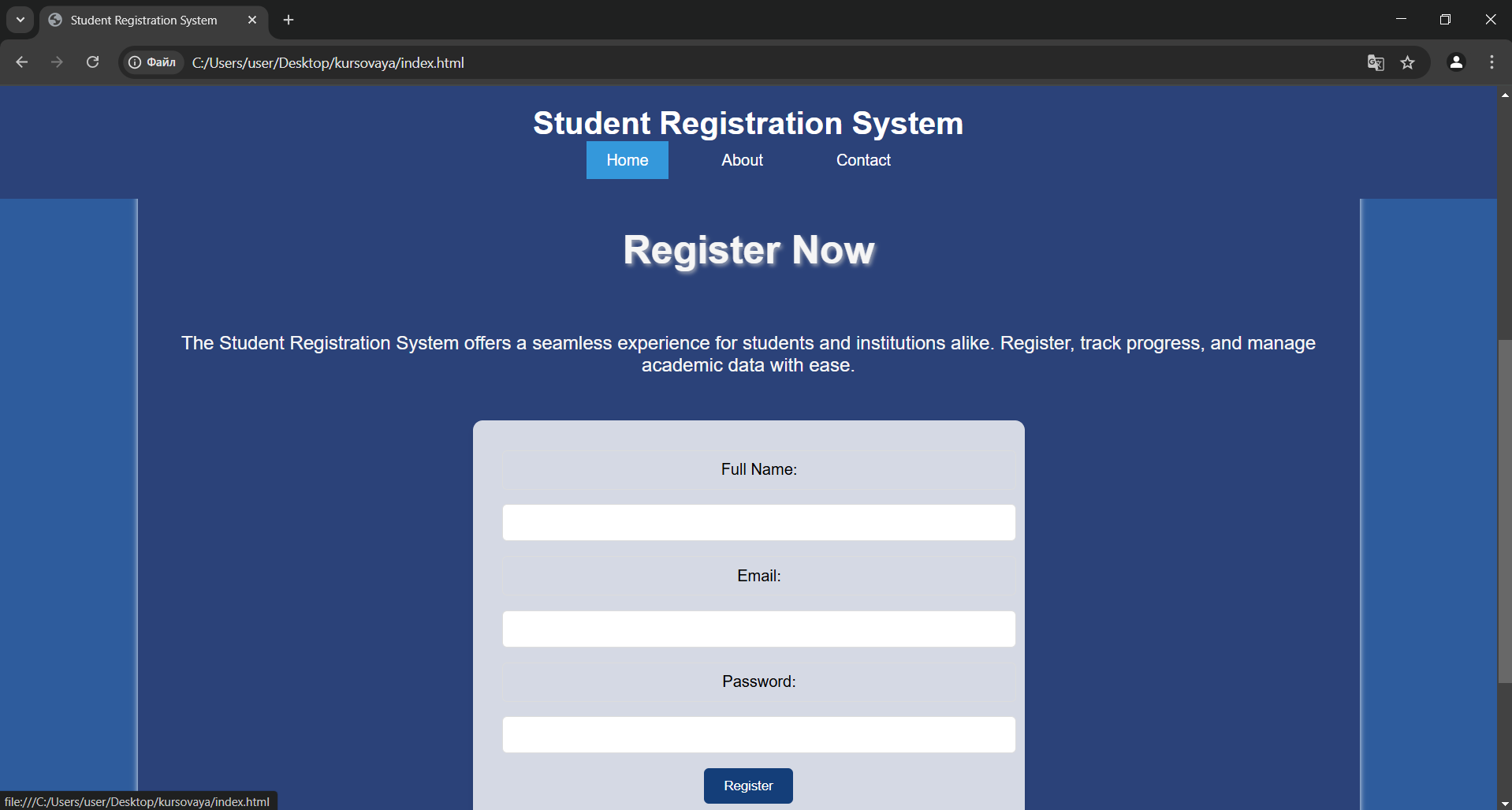


Рис.25

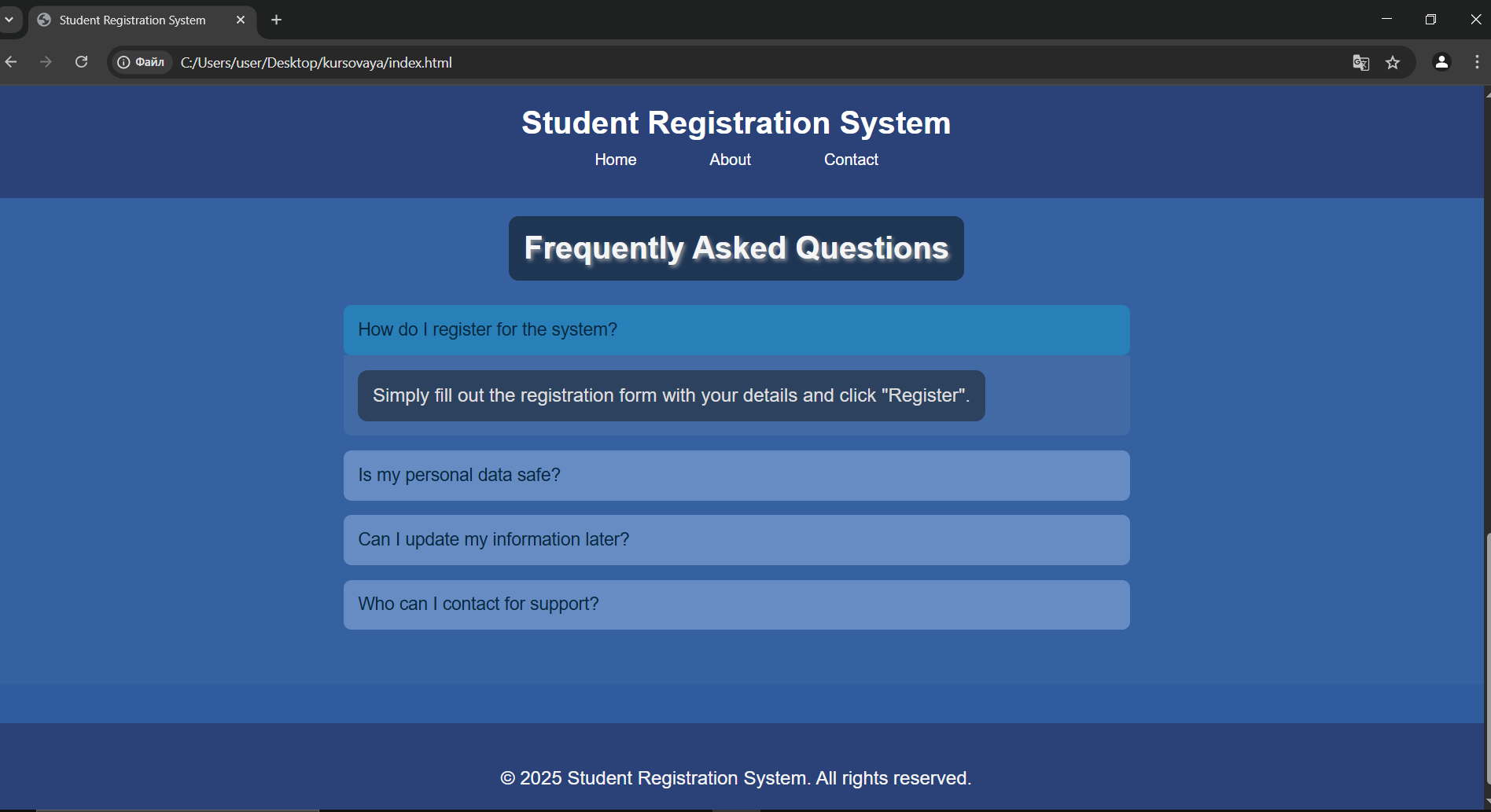


Рис.26

**Страница: About**



Рис.27

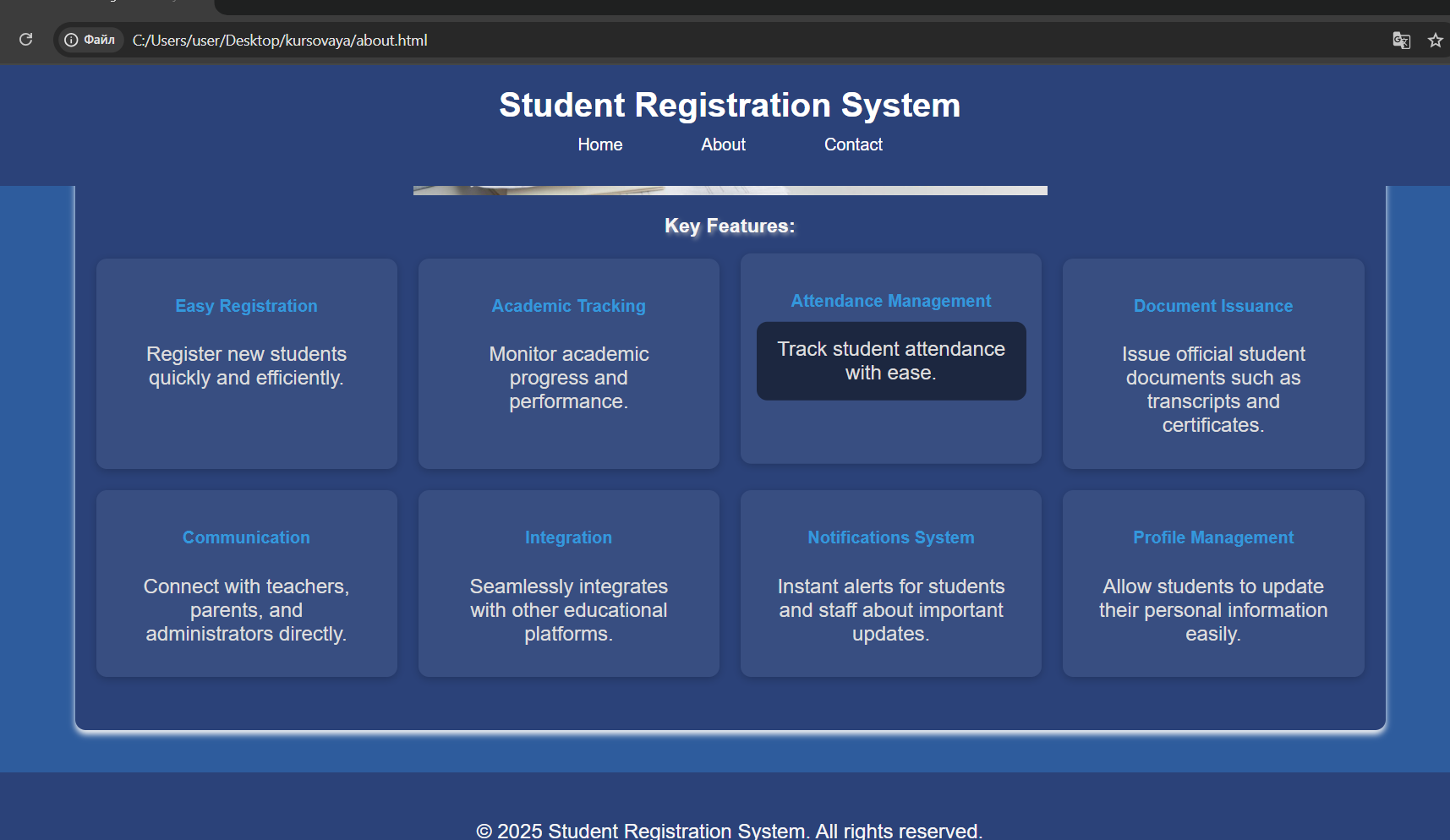


Рис.28

**Страница: Contact**

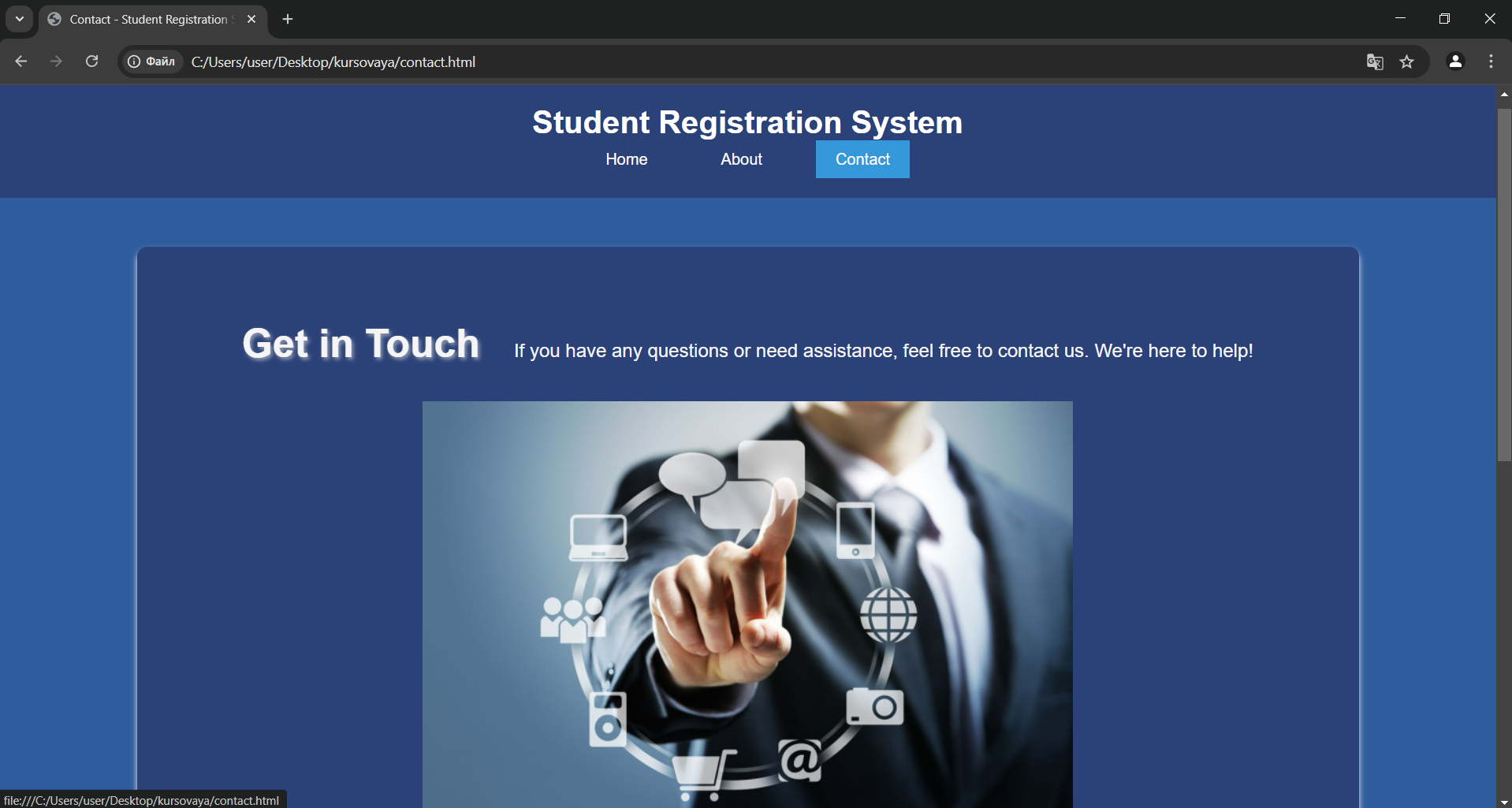


Рис.29

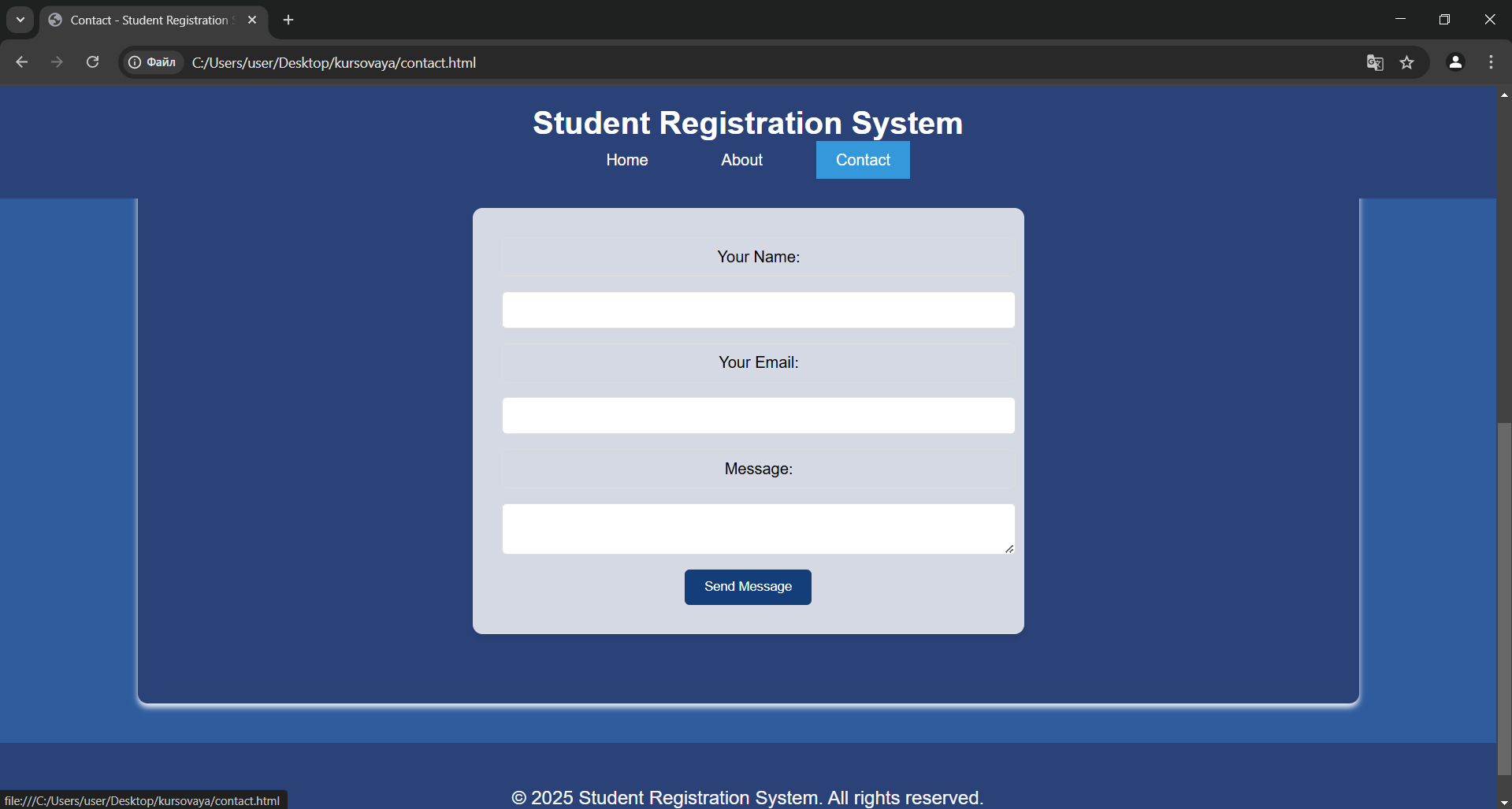


Рис.30

# 

# **4. ЛИТЕРАТУРА**

1. Фримен Э., Робсон Э. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. — СПб.: Питер, 2021.
2. Макфарланд Д. JavaScript. Полное руководство. — М.: Вильямс, 2020.
3. Кейт Дж. CSS: карманный справочник. — М.: Символ-Плюс, 2019.
4. Флэнаган Д. JavaScript. Подробное руководство. — СПб.: Питер, 2022.
5. Койн С. Проектирование адаптивных веб-интерфейсов. — М.: Диалектика, 2020.
6. Mozilla Developer Network (MDN) — [https://developer.mozilla.org](https://developer.mozilla.org/) — официальная документация по HTML, CSS и JavaScript.
7. Bootstrap — [https://getbootstrap.com](https://getbootstrap.com/) — фреймворк для адаптивной верстки.
8. Unsplash — [https://unsplash.com](https://unsplash.com/) — бесплатный ресурс изображений для оформления сайтов.

**Ссылка на GitHub:** <https://github.com/Cerezzas/student-regisrtration-system.git>