Проект

Real Time Chat App

автор: Кристиян Димитров

специалност, курс: Софтуерни технологии и Дизайн, 2 курс

фн: 2301681013

**Съдържание**

[**Цел на проекта** 1](#_Toc193649458)

[**Основни функционалности** 1](#_Toc193649459)

[**Целева група** 1](#_Toc193649460)

[**Обща архитектура** 1](#_Toc193649461)

[**Основни технологии** 1](#_Toc193649462)

[**Проектиране и структура** 1](#_Toc193649463)

[**Главна страница (Регистрация на потребители)** 1](#_Toc193649464)

[**PHP (Сесия и проверка за логнат потребител)** 1](#_Toc193649465)

[**Регистрация** 1](#_Toc193649466)

[**Регистрация на потребител (signup.php)** 1](#_Toc193649467)

[**JavaScript код за регистрация:** 1](#_Toc193649468)

[**JavaScript код скриване и показване на парола** 1](#_Toc193649469)

[**Връзка с база данни** 1](#_Toc193649470)

[**Вход на потребители** 1](#_Toc193649471)

[**Код за вход:** 1](#_Toc193649472)

[**PHP кода за вход (Login)** 1](#_Toc193649473)

[**JavaScript код за вход:** 1](#_Toc193649474)

[**Изход на потребители** 1](#_Toc193649475)

[**PHP обработка на logout (logout.php)** 1](#_Toc193649476)

[**JavaScript код** 1](#_Toc193649477)

[**Потребителски интерфейс** 1](#_Toc193649478)

[**PHP код за извличане на данни за текущия потребител** 1](#_Toc193649479)

[**Контент** 1](#_Toc193649480)

[**Списък с потребители** 1](#_Toc193649481)

[**JavaScript код** 1](#_Toc193649482)

[**Търсачка за потребители** 1](#_Toc193649483)

[**Редактиране на профил** 1](#_Toc193649484)

[**Chat страница с потребител** 1](#_Toc193649485)

[**Chat страница:** 1](#_Toc193649486)

[**Get-chat.php** 1](#_Toc193649487)

[**Insert-chat.php** 1](#_Toc193649488)

[**JavaScript код** 1](#_Toc193649489)

[**CSS код** 1](#_Toc193649490)

[**Глобални стилове:** 1](#_Toc193649491)

[**Стилове на страницата** 1](#_Toc193649492)

[**Стилове за формуляри** 1](#_Toc193649493)

[**Секция с потребители** 1](#_Toc193649494)

[**Стилове за търсачката** 1](#_Toc193649495)

[**Списък с потребители** 1](#_Toc193649496)

[**Оформление на скролбар** 1](#_Toc193649497)

[**Оформление на секцията с чат съобщения** 1](#_Toc193649498)

[**Responsive Design** 1](#_Toc193649499)

# **Цел на проекта**

Този проект представлява уеб приложение за чат в реално време, което позволява на потребителите да изпращат и получават съобщения мигновено.

Приложението е разработено с помощта на PHP, MySQL, HTML, CSS и JavaScript, като осигурява бързо и ефективно потребителско изживяване.

Основната му цел е да осигури лесен начин за комуникация между потребители на платформи, които поддържат интернет браузъри.

# **Основни функционалности**

* **Регистрация на потребители**: Потребителите могат да се регистрират чрез предоставяне на име, фамилия, имейл, парола и профилна снимка.Лични съобщения в реално време между потребители.
* **Вход за потребители**: Потребителите могат да влизат в приложението с имейл и парола.
* **Търсене на потребители**: Потребителите могат да търсят други потребители по име.
* **Чат в реално време**: Потребителите могат да изпращат съобщения в реално време след успешен вход.
* **Интерфейс с възможности за качване на снимка**: Регистрирането на нов потребител включва качване на профилна снимка.
* **Страница с потребителски профили и статуси**: Потребителите могат да видят информация за своето потребителско име, фамилия, статус и профилна снимка.

# **Целева група**

* Всеки потребител, който желае да комуникира в реално време с други хора.
* Платформи и уебсайтове, които искат да вградят чат функционалност.

# **Обща архитектура**

Приложението използва **клиент-сървърна архитектура**.

Клиентът (потребителският интерфейс) комуникира със сървъра, където се обработват всички логики за вход, регистрация и чат съобщения.

Данните за потребителите и съобщенията се съхраняват в MySQL база данни.

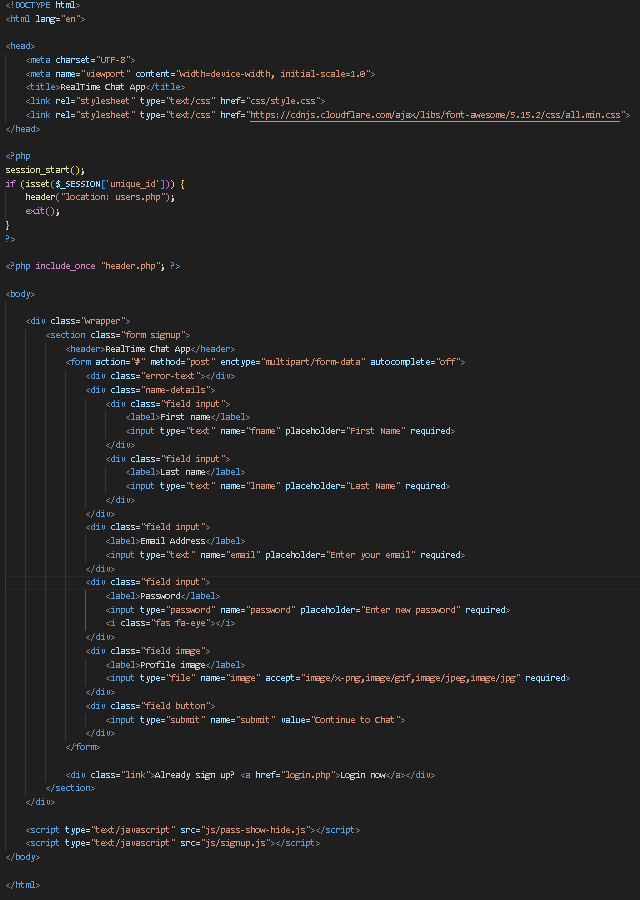
# **Основни технологии**

* **Frontend**: HTML, CSS, JavaScript.
* **Backend**: PHP за обработка на входни данни, регистрация, вход и чат съобщения.
* **Database**: MySQL за съхранение на потребителски данни и съобщения

# **Проектиране и структура**

## **Главна страница (Регистрация на потребители)**

### **PHP (Сесия и проверка за логнат потребител)**

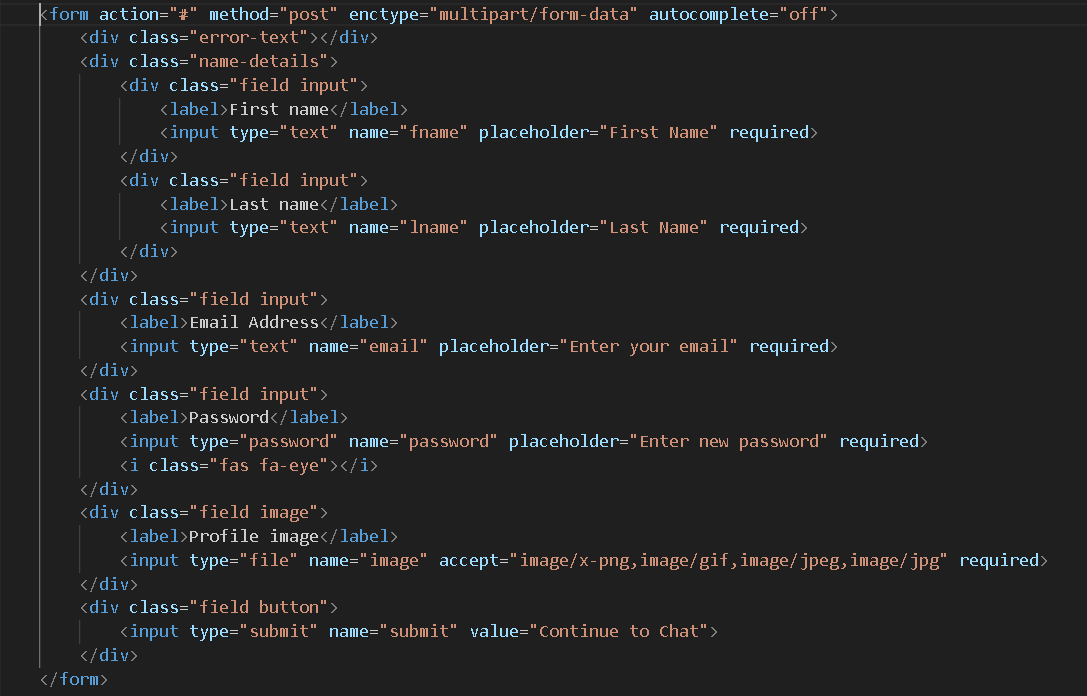


* session\_start(); → Започва **сесия**, за да може да се следи логнат потребител.
* if (isset($\_SESSION['unique\_id'])) → Проверява дали потребителят вече е логнат.
* header("location: users.php"); → Ако е логнат, пренасочва го към users.php.
* exit(); → Прекратява изпълнението на кода (за да се избегнат допълнителни операции).
* include\_once "header.php"; → Включва съдържанието на header.php (например навигация или мета данни).

### **Регистрация**

Потребителят попълва полетата за име, фамилия, имейл, парола и качва профилна снимка. При успешна регистрация потребителят се пренасочва към страницата за вход.

**Код за регистрация:**



**Атрибути на формата:**

* **action="#"**: Определя URL адреса, към който ще се изпратят данните при изпращането на формата. В този случай е зададено като #, което означава, че форма ще бъде обработена на същата страница.
* **method="post"**: Задава метода за изпращане на данните. Този метод изпраща данните чрез HTTP POST заявка, което е по-сигурен начин за изпращане на чувствителни данни (като пароли).
* **enctype="multipart/form-data"**: Този атрибут е необходим, когато формата съдържа полета за качване на файлове (като изображение), тъй като позволява изпращането на данни, които не са само текст (например изображения или документи).
* **autocomplete="off"**: Деактивира автоматичното попълване на формата от браузъра, което е полезно за предотвратяване на запазване на лични данни в браузъра, когато става дума за чувствителни данни като пароли.

**Полета в формата:**

1. **Име на полето за първо име (<div class="field input">)**:
   1. **<label>**: Показва текстовото описание на полето ("First name").
   2. **<input type="text" name="fname" placeholder="First Name" required>**: Поле за въвеждане на първото име. Атрибутът required указва, че това поле е задължително за попълване, а placeholder показва текст по подразбиране, когато полето е празно.
2. **Име на полето за фамилия (<div class="field input">)**:
   1. Същото като първото поле, но за фамилия ("Last name").
3. **Име на полето за имейл (<div class="field input">)**:
   1. Поле за въвеждане на имейл адрес. Атрибутът required отново указва, че това поле трябва да бъде попълнено.
4. **Име на полето за парола (<div class="field input">)**:
   1. Поле за въвеждане на парола. С тип password, което означава, че въведеното съдържание ще бъде скрито (обикновено с точки или звездички).
5. **Име на полето за качване на профилна снимка (<div class="field image">)**:
   1. **<input type="file" name="image" accept="image/\*" required>**: Позволява на потребителя да избере файл за качване, в този случай профилна снимка. Атрибутът accept="image/\*" указва, че се приемат само изображения (PNG, JPG, JPEG и др.).
6. **Бутон за изпращане на формата (<div class="field button">)**:
   1. **<input type="submit" name="submit" value="Continue to Chat">**: Бутон, който изпраща данните от формата. Текстът на бутона е "Continue to Chat" (Продължи към чата).

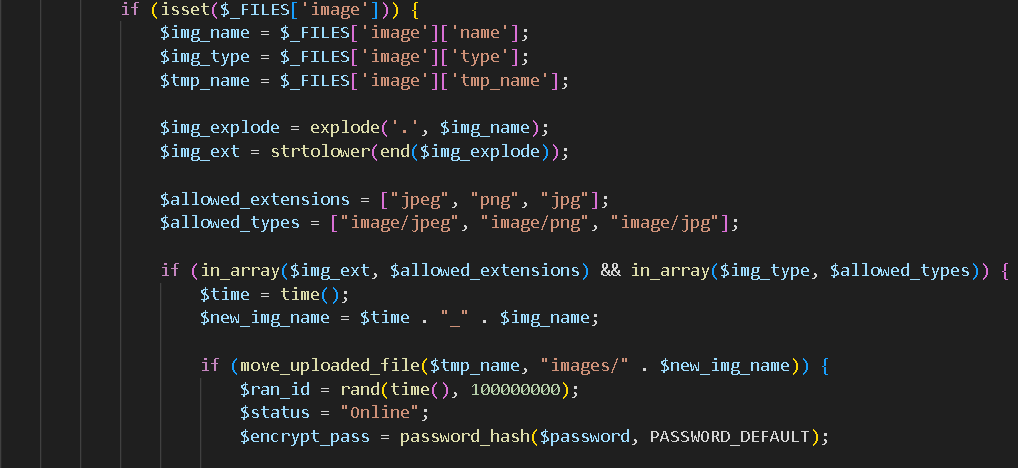
### **Регистрация на потребител (signup.php)**

|  |  |  |  |
| --- | --- | --- | --- |
| * session\_start(); – стартира сесия, която ще се използва за запазване на идентификатора на потребителя след успешна регистрация. * include\_once "config.php"; – включва конфигурационния файл, който вероятно съдържа информация за връзката с базата данни. | * error\_reporting(E\_ALL); – активира всички съобщения за грешки. * ini\_set('display\_errors', 1); – принуждава PHP да показва грешките на екрана. | * Ако $conn не съществува (няма връзка с базата данни), скриптът прекъсва изпълнението и показва грешка. | * Взимаме потребителските данни от $\_POST. * mysqli\_real\_escape\_string() се използва за защита срещу SQL инжекции. * ?? '' гарантира, че ако няма подадена стойност, ще бъде използван празен низ вместо NULL. |

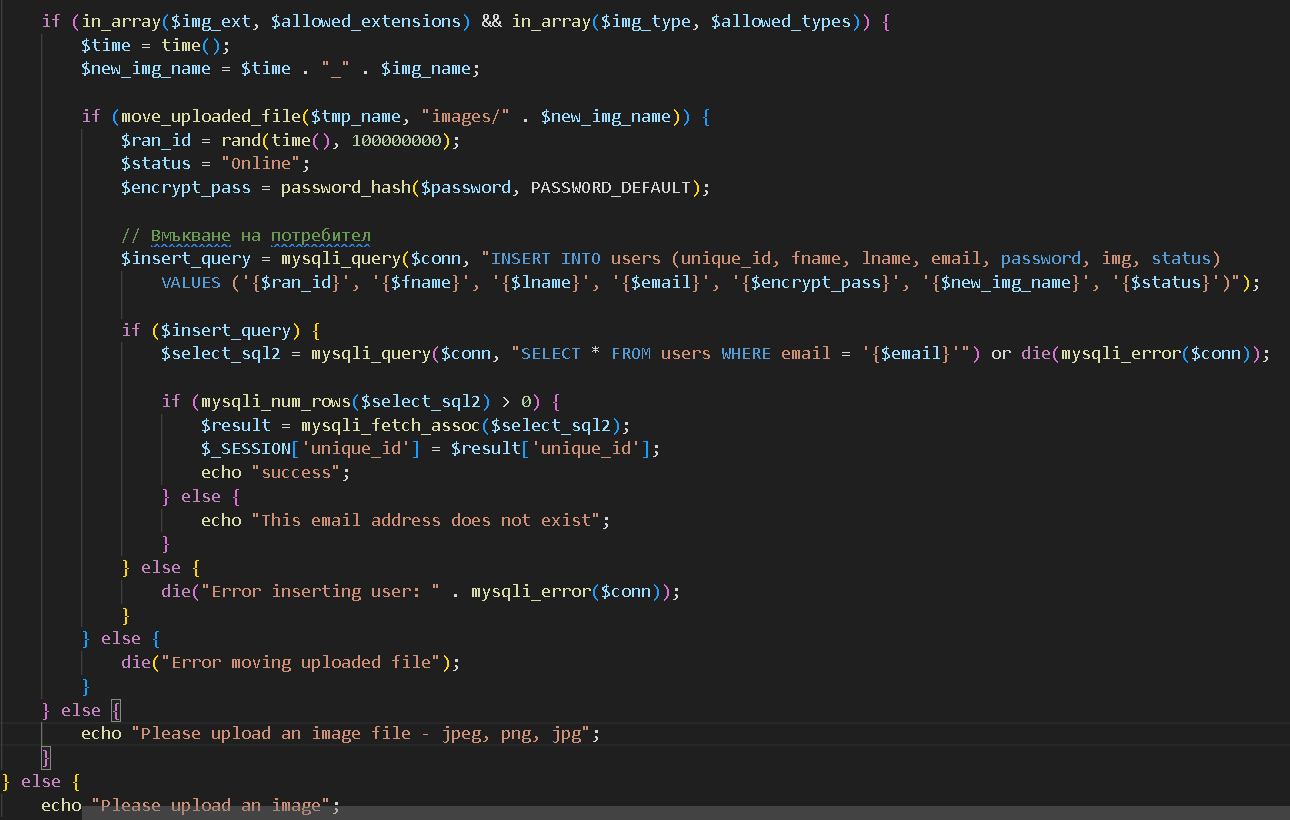
|  |  |
| --- | --- |
| * Уверяваме се, че потребителят е попълнил всички задължителни полета. | * filter\_var() проверява дали подаденият имейл има валиден формат. |



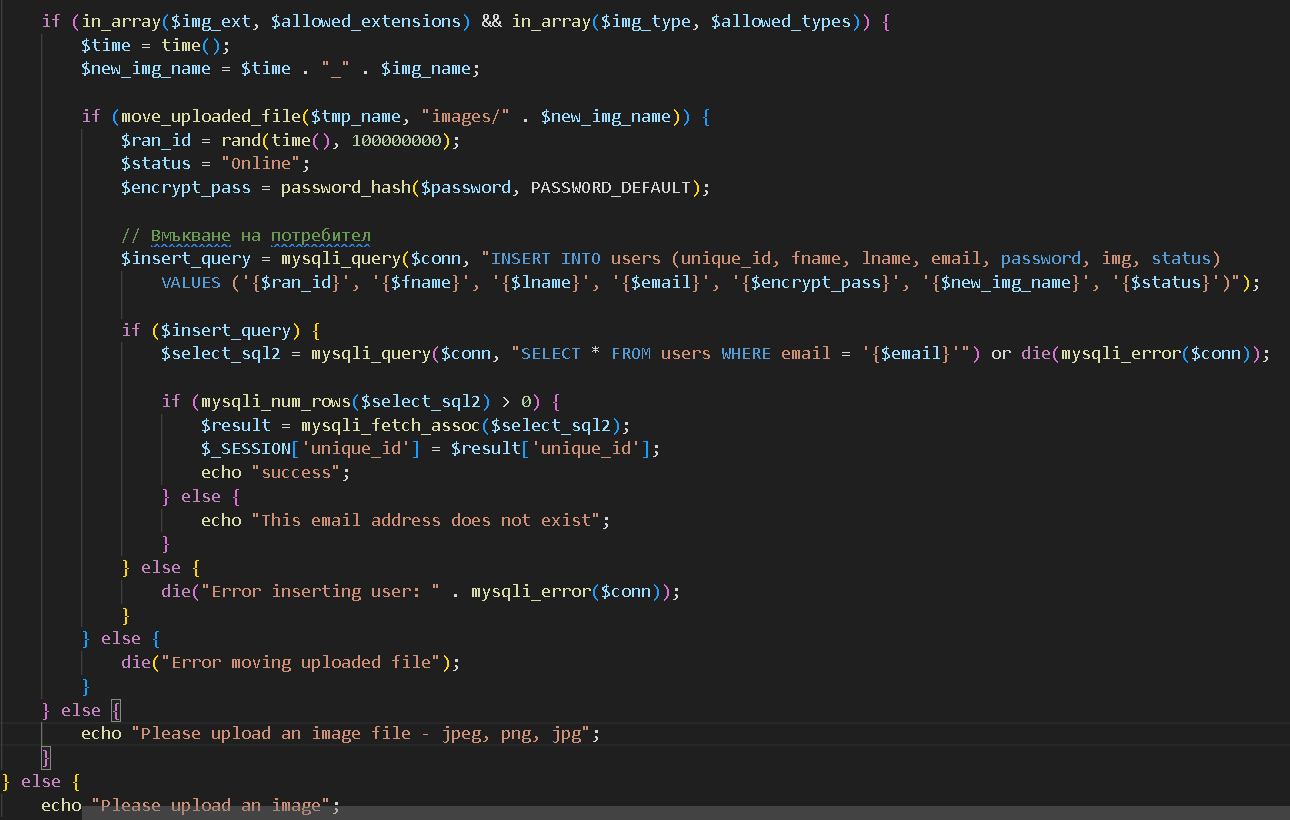
* Търсим в базата данни дали вече има потребител със същия имейл.
* Ако намерим съвпадение, връщаме съобщение, че имейлът вече съществува.



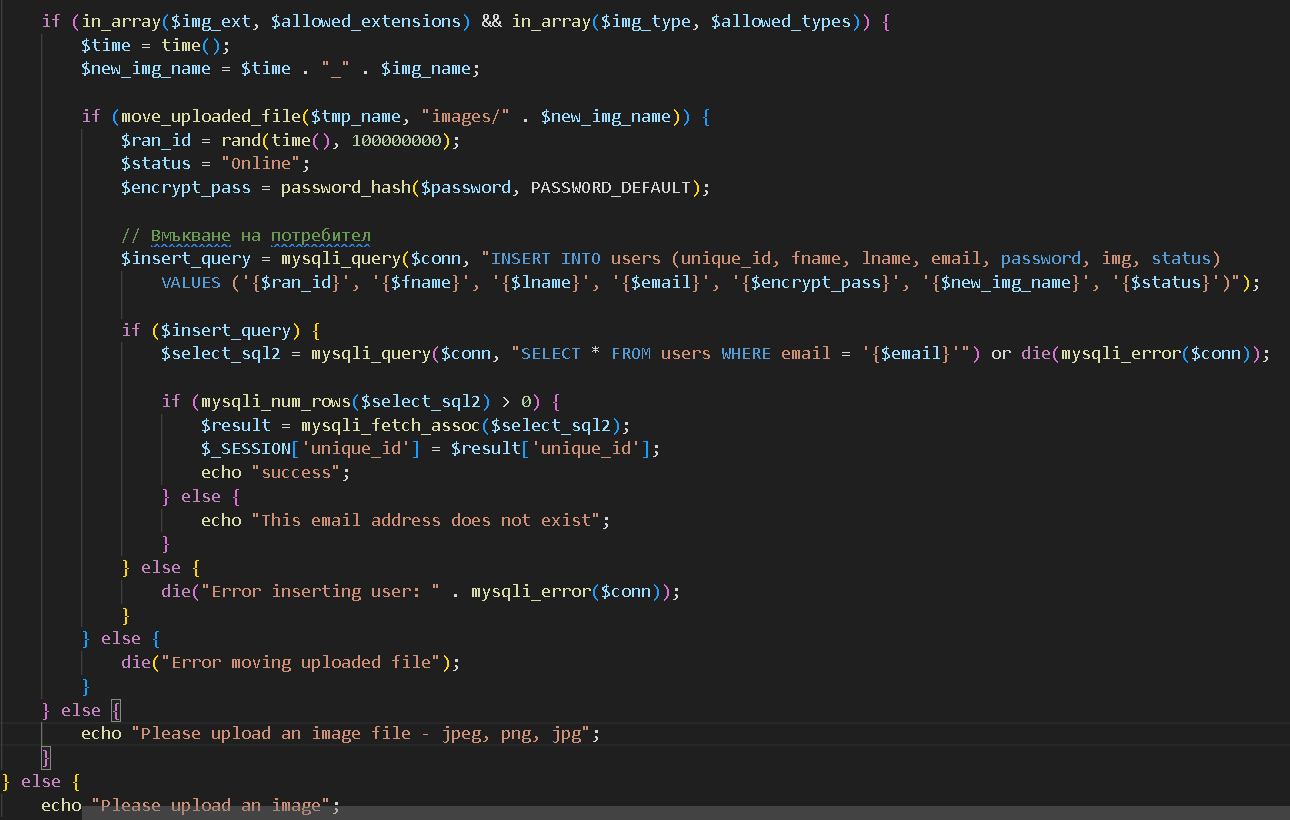
* Проверяваме дали потребителят е качил изображение.
* Взимаме информация за файла: име, тип и временна директория.
* Разделяме името на файла по точка, за да вземем разширението (jpeg, png, jpg).
* Създаваме списък с позволените разширения и MIME типове.



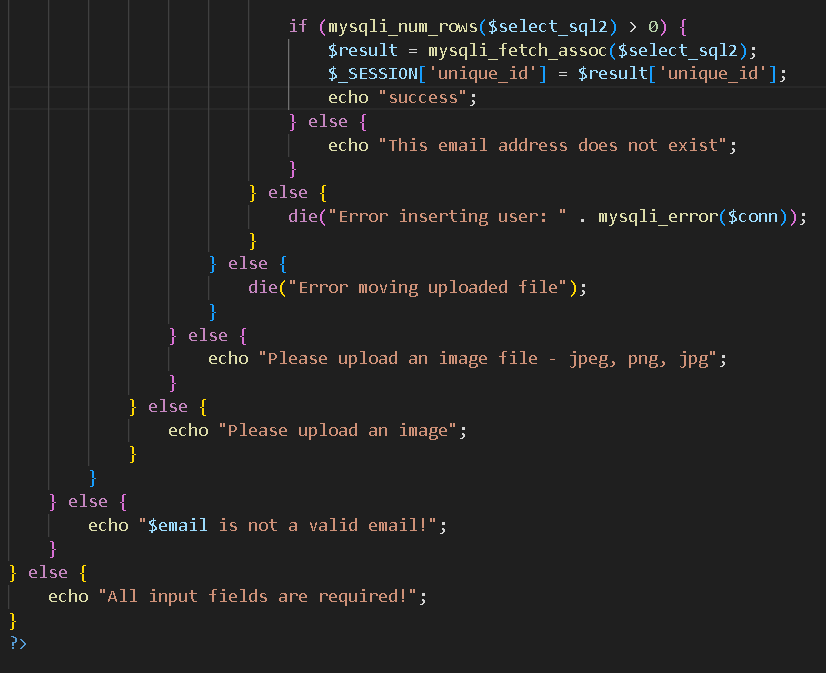
* Проверяваме дали разширението и типът на файла са валидни.`
* Създаваме уникално име за изображението, като добавяме времеви отметки.



* Преместваме каченото изображение в папката images/.
* rand(time(), 100000000); генерира уникален ID за потребителя.
* password\_hash($password, PASSWORD\_DEFAULT); хешира паролата за сигурност.
* Вмъкваме новия потребител в базата данни.



* Проверяваме дали заявката е успешна.
* Ако регистрацията е успешна, извличаме данните на новия потребител и го вписваме автоматично в системата чрез $\_SESSION['unique\_id'].



* Ако не успеем да добавим потребителя в базата, показваме грешка.
* Ако потребителят не качи снимка, показваме грешка.
* Ако имейлът не е валиден, връщаме грешка.
* Ако някое поле е празно, връщаме грешка.

### **JavaScript код за регистрация:**

|  |  |  |  |
| --- | --- | --- | --- |
| * **form**: Избира HTML формата в елемента с клас .signup form. * **continueBtn**: Избира бутона за изпращане на формата (елементът <input> с клас .button). * **errorText**: Избира елемента, в който ще се показват съобщения за грешка (предполага се, че е в <div class="error-text">). | * **onsubmit**: Задава обработчик на събитието, когато формата се изпрати. * **e.preventDefault()**: Това спира изпращането на формата по стандартния начин (през HTTP), така че данните да не се изпратят и страницата да не се презареди. Вместо това, ще се използва AJAX за изпращане на данните. | * **onclick**: Задава обработчик за натискането на бутона. Когато потребителят кликне върху бутона за продължаване, се изпълнява следният код. | * **xhr = new XMLHttpRequest()**: Създава нов обект за AJAX заявка. * **xhr.open("POST", "php/signup.php", true)**: Подготвя заявка с метод **POST** към PHP скрипта signup.php, който ще обработи данните. Третият параметър true указва, че заявката е асинхронна (няма да блокира изпълнението на други операции на страницата). |

|  |  |
| --- | --- |
| * **xhr.onload**: Задава обработчик на събитие, което ще се стартира, когато заявката е изпълнена (или в случай на грешка). * **xhr.readyState === XMLHttpRequest.DONE**: Проверява дали заявката е приключила (стойност 4 за завършено състояние). * **xhr.status === 200**: Проверява дали HTTP статус кодът е **200** (успешно изпълнена заявка). * **let data = xhr.response**: Взема отговора от сървъра (например текст или JSON).   + Ако отговорът е "success", потребителят бива пренасочен към страницата users.php, което означава, че регистрацията е успешна.   + Ако отговорът не е "success", се показва съобщение за грешка в елемента с клас .error-text. | * **let formData = new FormData(form)**: Събира всички данни от формата, включително файловете (като изображения), в обект FormData. * **xhr.send(formData)**: Изпраща събраните данни чрез **AJAX** към сървъра. |

### **JavaScript код скриване и показване на парола**



* **pswrdField** – Избира **полето за парола** (<input type="password">).  
  **toggleIcon** – Избира **иконата за превключване** (<i>), намираща се в .form .field.
* Ако **паролата е скрита (type="password")** → променяме на **text** (показване на паролата).  
  Ако **паролата е показана (type="text")** → променяме обратно на **password** (скриване на паролата).
* **toggleIcon.classList.add("active")** – Добавя клас "active" (може да променя стила на иконата).  
   **toggleIcon.classList.remove("active")** – Премахва "active", когато паролата отново се скрие.

## **Връзка с база данни**

Този код се използва за **установяване на връзка с базата данни** и **стартиране на сесията**, ако тя не е стартирана.Той се включва в други файлове (include\_once "config.php";), за да може всеки скрипт да използва връзката към базата.

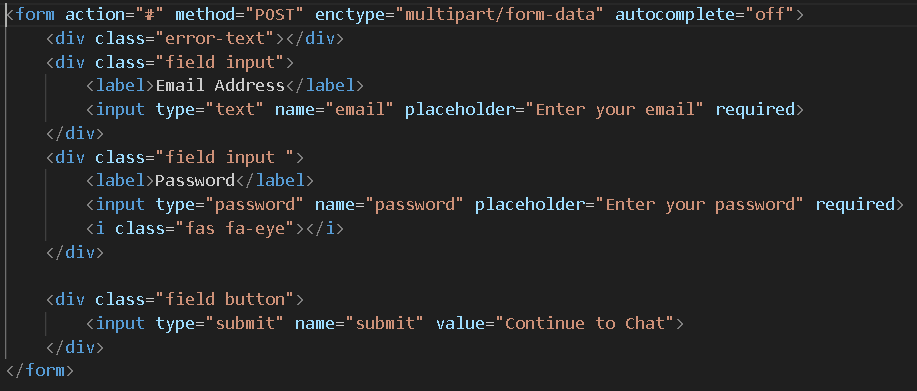
|  |  |  |
| --- | --- | --- |
| * **session\_status()** проверява състоянието на сесията. * Ако **няма активна сесия (PHP\_SESSION\_NONE)**, тогава се стартира с **session\_start();**.   + Това гарантира, че **няма да се стартира сесия повече от веднъж**, дори ако config.php се включва няколко пъти. | * **$hostname = "localhost";** – Това означава, че базата данни се намира **локално** на същия сървър. * **$username = "root";** – Стандартно потребителско име за локални сървъри (например XAMPP, WAMP). * **$password = "";** – По подразбиране **няма парола** за MySQL в XAMPP. * **$dbname = "phpchatapp\_db";** – Това е **името на базата данни**, която приложението използва. | * **mysqli\_connect()** прави опит за връзка към базата данни с подадените параметри.   + Ако връзката е успешна, в променливата $conn се съхранява **връзката с базата**. |

|  |  |
| --- | --- |
| * Ако връзката **не може да бъде установена** (!$conn означава "ако $conn е false"),   + Тогава **показва грешка** чрез mysqli\_connect\_error(), което връща описание на проблема. | * mysqli\_set\_charset($conn, "utf8mb4"); настройва кодировката на **utf8mb4**.   + **Без тази настройка, българските букви и емоджитата може да се показват грешно в чата!** |

## **Вход на потребители**

Потребителите могат да се логват с имейл и парола, ако са вече регистрирани.

### **Код за вход:**



**Атрибути на формата:**

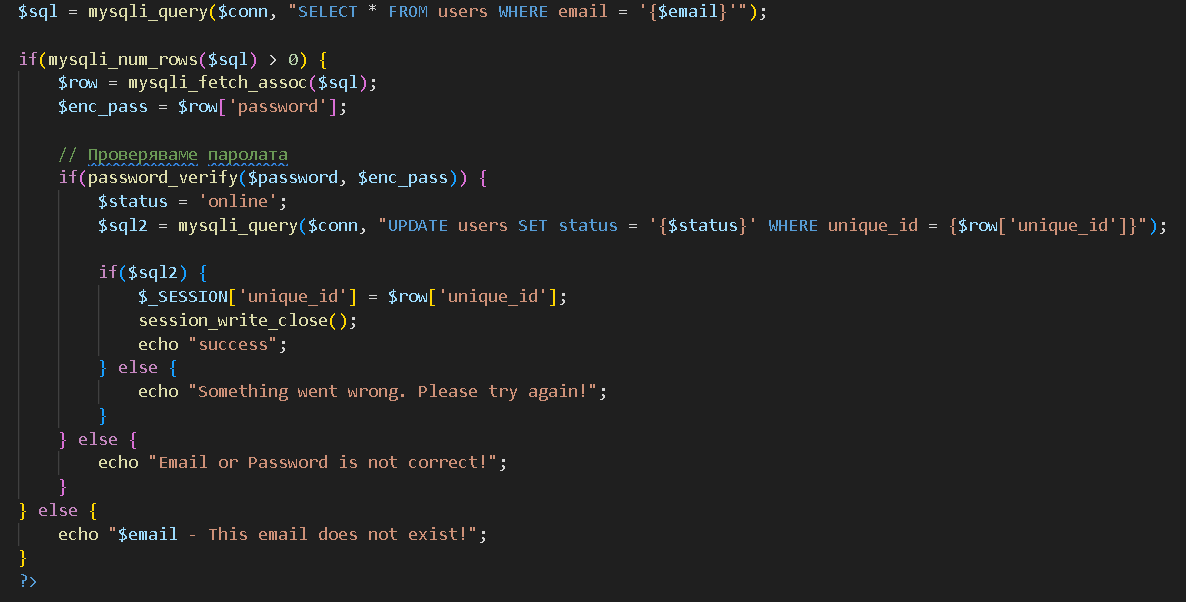
* **action="#"**: Определя URL адреса, към който ще се изпратят данните при изпращането на формата. В този случай е зададено като #, което означава, че форма ще бъде обработена на същата страница.
* **method="post"**: Задава метода за изпращане на данните. Този метод изпраща данните чрез HTTP POST заявка, което е по-сигурен начин за изпращане на чувствителни данни (като пароли) , тъй като данните не се показват в URL адреса.
* **enctype="multipart/form-data"**: Този атрибут е необходим за качване на файлове чрез формата (макар в този случай формата да не съдържа полета за файлове, този атрибут може да бъде добавен за бъдеща разширяемост).
* **autocomplete="off"**: Деактивира автоматичното попълване на формата от браузъра, което е полезно за предотвратяване на запазване на лични данни в браузъра, когато става дума за чувствителни данни като пароли.

**Полета в формата:**

1. **Поле за имейл (<div class="field input">)**:
   1. **<label>Email Address</label>**: Показва етикет с текст "Email Address", който описва това, което потребителят трябва да въведе.
   2. **<input type="text" name="email" placeholder="Enter your email" required>**: Поле за въвеждане на имейл адрес.
      1. **type="text"**: Означава, че това е текстово поле.
      2. **name="email"**: Задава името на полето, което се използва за извличане на стойността му на сървърната страна при обработка на формата.
      3. **placeholder="Enter your email"**: Това е текстът, който се показва в полето, когато е празно, за да подскаже на потребителя каква информация трябва да бъде въведена.
      4. **required**: Указва, че полето е задължително и не може да бъде изпратено празно.
2. **Поле за парола (<div class="field input">)**:
   1. **<label>Password</label>**: Показва етикет с текст "Password", който описва полето за парола.
   2. **<input type="password" name="password" placeholder="Enter your password" required>**: Поле за въвеждане на парола.
      1. **type="password"**: Използва се за паролни полета, като въведената стойност ще бъде скрита (показвана обикновено със звездички или точки).
      2. **name="password"**: Името на полето, което ще бъде използвано на сървъра за извличане на стойността на паролата.
      3. **placeholder="Enter your password"**: Текстът, който се показва в полето, когато то е празно.
      4. **required**: Указва, че полето за парола е задължително за попълване.
3. **Бутон за изпращане на формата (<div class="field button">)**:
   1. **<input type="submit" name="submit" value="Continue to Chat">**: Бутон, който изпраща данните от формата.
      1. **type="submit"**: Определя, че това е бутон за изпращане на формата.
      2. **name="submit"**: Името на бутона (може да бъде полезно за обработка на данни на сървъра).
      3. **value="Continue to Chat"**: Текстът, който се показва върху бутона, когато потребителят трябва да натисне, за да изпрати формата.

### **PHP кода за вход (Login)**

|  |  |
| --- | --- |
| * **session\_start();** – Стартира PHP сесията, за да можем да използваме $\_SESSION. * **include\_once "config.php";** – Включва конфигурационния файл, където е връзката с базата * **Проверява дали връзката с базата ($conn) е успешна.**  Ако има грешка – **прекъсва скрипта (die()) и показва съобщение**. * Проверява дали заявката **идва чрез POST метод** (ако не – грешка). * Проверява дали **email и password** са попълнени (ако не – грешка). * **Филтрира въведените данни** чрез mysqli\_real\_escape\_string() (защита срещу SQL инжекции). | * **Изпълнява SQL заявка**, която **търси потребител с въведения email**. * mysqli\_num\_rows($sql) > 0 – Проверява дали е намерен потребител. * mysqli\_fetch\_assoc($sql) – Извлича потребителските данни като **асоциативен масив**. * $enc\_pass = $row['password']; – Взима криптираната парола от базата. |



* password\_verify($password, $enc\_pass) – Сравнява въведената парола с **хешираната парола** от базата.  
   Ако **паролата е вярна**, продължава с логването.
* Ако входът е успешен, задава **статуса на потребителя** на "online".
* UPDATE users SET status = 'online' WHERE unique\_id = ... – Обновява статуса в базата.

|  |  |
| --- | --- |
| * Ако **обновяването на статуса е успешно**, * Създава **сесия** и запазва **unique\_id** в $\_SESSION. * Връща "success", което ще позволи **пренасочване към users.php**. | * Ако паролата не съвпада – **показва грешка**. * Ако email не е намерен – **съобщава, че няма такъв акаунт**. |

### **JavaScript код за вход:**

|  |  |  |  |
| --- | --- | --- | --- |
| * **form**: Избира формата, която е вътре в елемента с клас .login. * **continueBtn**: Избира бутона за изпращане на формата (елемент <input> в контейнера с клас .button). * **errorText**: Избира елемента с клас .error-text, в който ще се показват съобщения за грешка (ако има такива). | * **onsubmit**: Това е обработчик на събитието, което се задейства, когато потребителят натисне бутона за изпращане на формата. * **e.preventDefault()**: Тази команда спира стандартното действие на браузъра при изпращане на формата, което би довело до презареждане на страницата. Вместо това, ще използваме **AJAX** за асинхронно изпращане на данните. | * **onclick**: Това е обработчик на събитието, което се задейства, когато потребителят натисне бутона за влизане. | * **xhr = new XMLHttpRequest()**: Създава нов обект за **AJAX** заявка. * **xhr.open("POST", "php/login.php", true)**: Подготвя **POST** заявка към PHP скрипта login.php, който ще обработи данните. Третият параметър true указва, че заявката ще бъде асинхронна (няма да блокира останалия код на страницата). |

|  |  |
| --- | --- |
| * **xhr.onload**: Задава обработчик за събитие, което ще се стартира, когато заявката бъде изпълнена. * **xhr.readyState === XMLHttpRequest.DONE**: Проверява дали заявката е завършена. * **xhr.status === 200**: Проверява дали статус кодът на HTTP отговор е **200**, което означава успешно изпълнение на заявката. * **let data = xhr.response**: Взима отговорът от сървъра (който може да бъде текст или JSON).   + Ако отговорът е "success", потребителят ще бъде пренасочен към страницата users.php, което предполага успешен вход в приложението.   + Ако отговорът е нещо друго (например грешка), ще се покаже съобщение за грешка в елемента с клас .error-text. | * **let formData = new FormData(form)**: Събира всички данни от формата, включително стойности на полета за текст и файлове (ако има). * **xhr.send(formData)**: Изпраща данните на сървъра чрез **AJAX** заявка. |

## **Изход на потребители**

**Изходът от системата се реализира чрез PHP и JavaScript.**

* **PHP скрипт (logout.php)**: Обновява статуса на потребителя в базата данни на "offline now", след което унищожава сесията и пренасочва потребителя към страницата за вход.
* **JavaScript код**: Изпраща асинхронна GET заявка към logout.php, обработва отговора и ако всичко е успешно, пренасочва потребителя към login.php.

### **PHP обработка на logout (logout.php)**

|  |  |
| --- | --- |
| * Проверява дали в сесията има **unique\_id** → ако има, значи потребителят е влязъл.   + Ако няма, **пренасочва към страницата за вход** (login.php). * Запазва уникалния идентификатор на потребителя в променлива $unique\_id.   + echo се използва за **тестови цели** – показва ID-то на потребителя. * **Изпълнява SQL заявка**, която променя статуса на потребителя в базата.   + UPDATE users SET status = 'offline now' WHERE unique\_id = '{$unique\_id}' | * Ако заявката към базата е успешна, извежда **"Status updated successfully."**   + **session\_unset();** – Изчиства всички сесийни променливи.   + **session\_destroy();** – Унищожава сесията.   + **header("Location: ../login.php");** – **Пренасочва потребителя** към страницата за вход. * Ако има проблем с заявката, **показва грешката** чрез mysqli\_error($conn).   + Това помага при **отстраняване на проблеми (debugging)**. * Ако **потребителят вече не е логнат** (unique\_id не е в сесията),   + **Пренасочва го към страницата за вход (login.php)**. |

### **JavaScript код**

|  |  |  |
| --- | --- | --- |
| * Тук се избира HTML елемент с клас logout, който е бутон или линк за изход (logout). | * Прикрепване на събитие към бутона за изход | * Създаване на XMLHttpRequest обект |

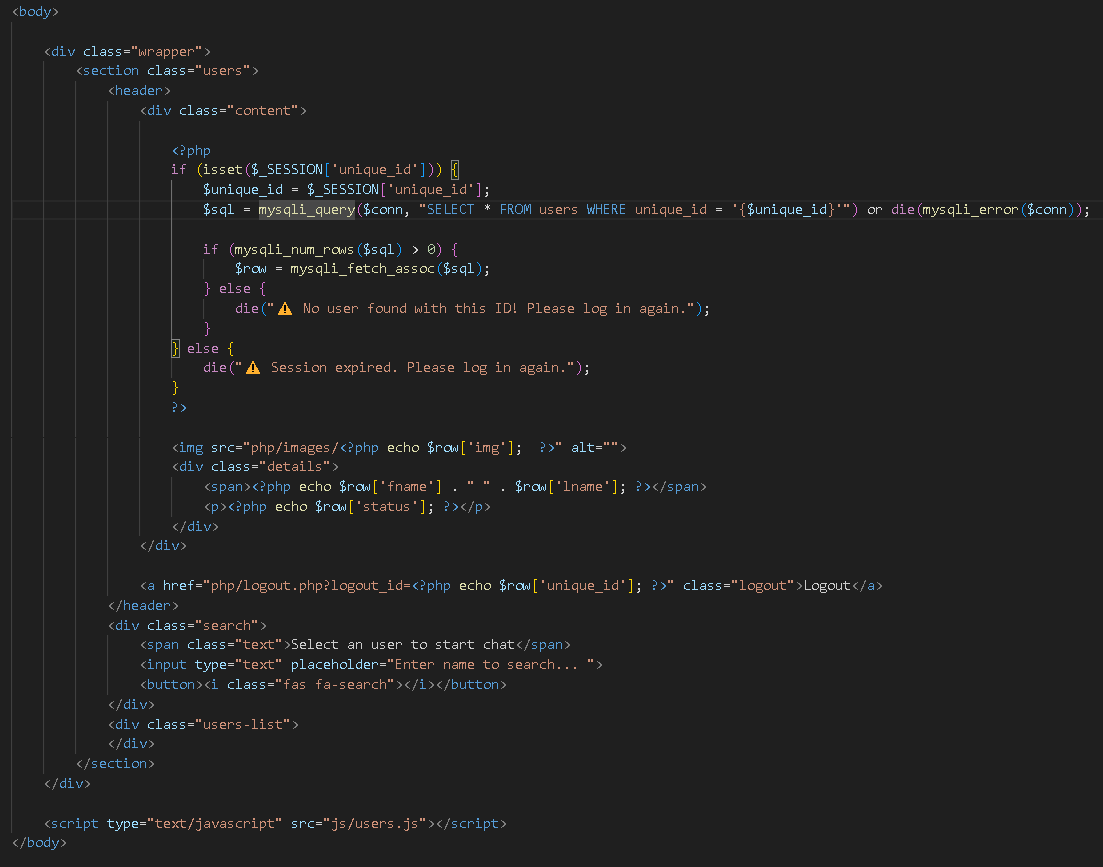
|  |  |
| --- | --- |
| * Отваряне на заявка (HTTP GET към logout.php) | * Проверка на готовността и статус на заявката |

|  |  |
| --- | --- |
| * **xhr.responseText.trim()**: Взима текстовия отговор от сървъра и премахва всички ненужни интервали в началото и края на текста. * **console.log("📩 Logout Response:", response);**: Извежда отговора на сървъра в конзолата, което е полезно за дебъгване (разработка). | * Ако отговорът от сървъра е "success", което означава, че изходът е успешен, тогава потребителят ще бъде пренасочен към страницата за вход (login.php) с помощта на **window.location.href**. * Ако отговорът не е "success", това означава, че нещо е било неуспешно. В този случай се показва съобщение за грешка, което информира потребителя защо изходът не е успял. |

## **Потребителски интерфейс**

Този код представлява част от HTML документ, който се използва за показване на информация за потребител, с възможност за търсене на други потребители и изход от сесията.

### **PHP код за извличане на данни за текущия потребител**



* PHP код, който проверява дали сесията съдържа уникален идентификатор на потребителя (unique\_id).
  + Ако е така, се извършва SQL заявка, за да се извлекат данни за потребителя от таблицата users в базата данни, като използва уникалния идентификатор ($unique\_id).
  + Ако не може да се изпълни заявката, ще се изведе съобщение за грешка.
* Втория if проверява дали заявката е върнала резултати (дали има запис с такъв уникален идентификатор).
  + Ако има резултат, извлича данните за потребителя и ги поставя в променливата $row.
  + Ако заявката не върне резултати (няма такъв потребител с този идентификатор), ще прекрати изпълнението на скрипта и ще изведе съобщение за грешка.
* Ако няма активна сесия (т.е. потребителят не е влязъл в системата), скриптът ще прекрати изпълнението и ще покаже съобщение за изтекла сесия.

### **Контент**

|  |  |
| --- | --- |
| * Изобразява снимка на потребителя, като пътят до изображението се извлича от базата данни ($row['img']). | * Показва името на потребителя ($row['fname'] и $row['lname']) и неговия статус ($row['status']) в контейнер с клас details. |

|  |  |  |
| --- | --- | --- |
| * Връзка за изход от системата, когато потребителят кликне на този линк, ще бъде изпратен към logout.php със зададен уникален идентификатор (logout\_id) на потребителя. * Това ще сигнализира на сървъра да приключи сесията и да изведе потребителя от системата. | * Създава секция за търсене на потребители, която съдържа текстово поле за въвеждане на име, бутон за търсене и икона за търсене (с помощта на Font Awesome). | * Контейнер за списъка с потребители, който може да бъде попълнен динамично с резултати от търсенето чрез JavaScript. |

### **Списък с потребители**

* Тази заявка избира **всички потребители**, **освен** текущия (WHERE NOT unique\_id = {$outgoing\_id}).
* **Сортира ги по user\_id в низходящ ред (ORDER BY user\_id DESC)**, така че **най-новите потребители** да се показват **първи**.

|  |  |
| --- | --- |
| * Изпълнява SQL заявката с помощта на mysqli\_query(), като използва активната връзка с базата данни ($conn). * mysqli\_query() връща **резултатен обект**, който съдържа данните от заявката. | * **mysqli\_num\_rows($query)** проверява **колко реда** (потребители) са върнати от заявката. * Ако **няма други потребители**, mysqli\_num\_rows($query) == 0, и в този случай:   + **$output** получава текста "No users are available to chat", което означава, че няма налични потребители за чат. |

### **JavaScript код**

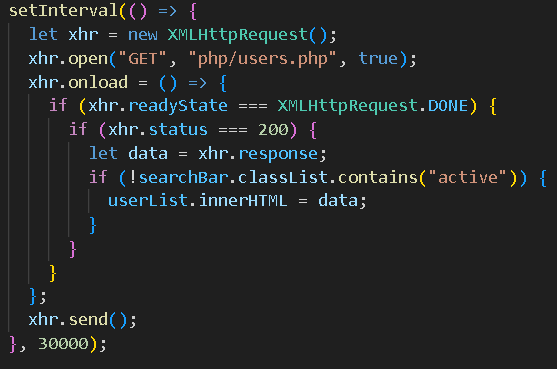
**Функционалността на полето за търсене** – показване/скриване и изпращане на заявка за търсене.

**Търсене на потребители в базата данни** – чрез AJAX заявка към php/search.php.

**Автоматично обновяване на списъка с потребители** – чрез периодична AJAX заявка към php/users.php на всеки **30 секунди**.

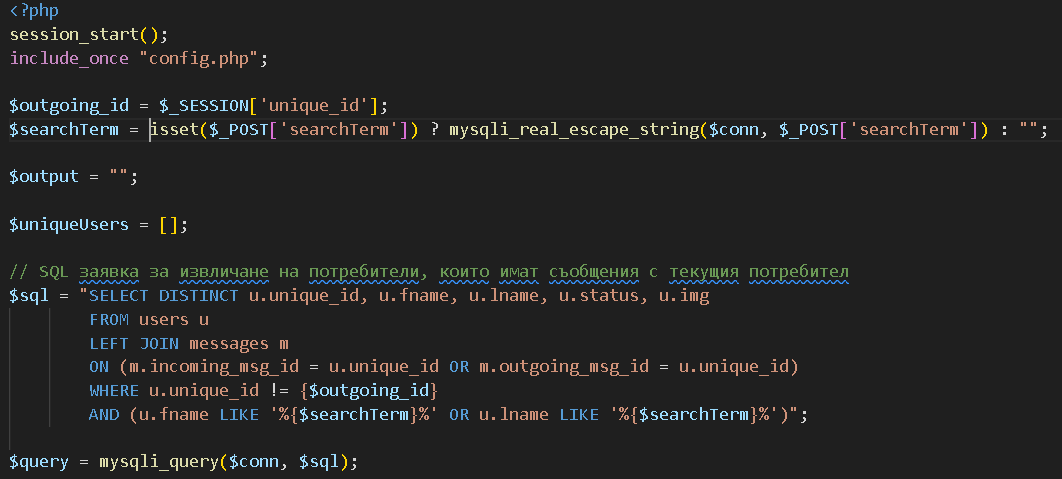
|  |  |
| --- | --- |
| * **searchBar** – Това е полето за търсене (<input>), което ще използваме, за да въвеждаме име на потребител. * **searchIcon** – Това е бутонът (<button>), който потребителят натиска, за да активира полето за търсене. * **userList** – Това е <div class="users-list">, който съдържа списъка с потребители. | **При клик върху searchIcon (бутон за търсене):**   * Добавя/премахва (toggle) CSS клас show на searchBar, което вероятно визуално показва или скрива полето. * Добавя/премахва (toggle) CSS клас active на searchIcon, което може да промени иконата (например от лупа на „X“). * Фокусира полето (searchBar.focus();). * Ако полето за търсене вече съдържа текст, изчиства го (searchBar.value = "";). |

|  |  |
| --- | --- |
| * Този код се изпълнява **всеки път, когато потребителят натисне клавиш в полето за търсене**. * Целта му е **да проверява дали в полето има текст** и да добавя/премахва CSS клас "active". | * **Създава AJAX заявка** (new XMLHttpRequest()). * **Отваря POST заявка към php/search.php** (където ще търсим потребители). * **Когато получи отговор (onload)**:   + Проверява дали заявката е завършена (xhr.readyState === XMLHttpRequest.DONE).   + Проверява дали отговорът е успешен (xhr.status === 200).   + Получените данни (xhr.responseText) се записват в data.   + Замества съдържанието на userList с резултатите от search.php. * Изпраща заявката със стойността на searchTerm като POST параметър ("searchTerm=" + searchTerm). * Това означава, че когато потребителят въвежда име, автоматично се търси в базата данни и резултатите се показват в реално време! |

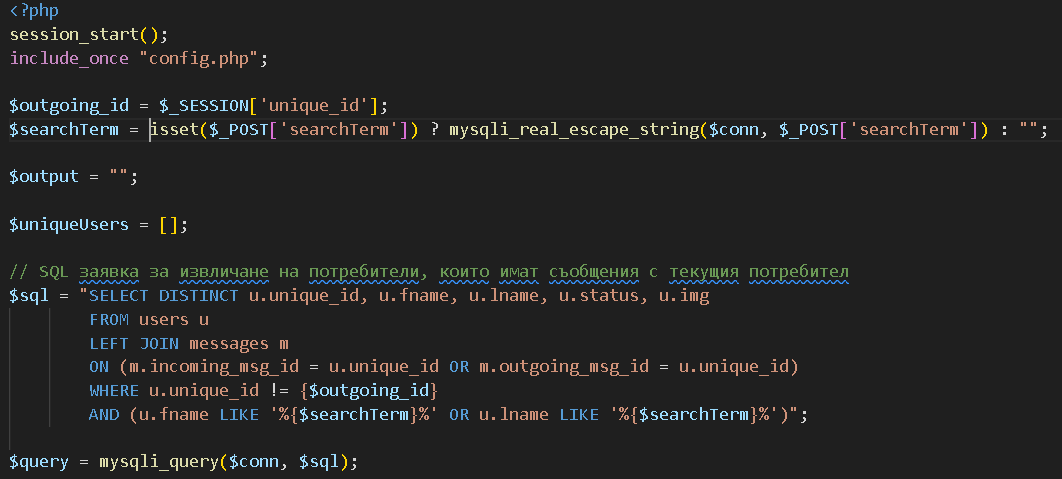


* На всеки **30 секунди (30000 милисекунди)** се изпълнява тази функция.
* Създава **AJAX GET заявка** към php/users.php, който връща списъка с потребители.
* Когато получи отговор (xhr.status === 200), замества userList.innerHTML = data.
* Ако потребителят търси (searchBar.classList.contains("active")), обновяването се спира, за да не се изтрие текущото търсене.
* Това осигурява динамично обновяване на списъка с потребители, така че новите потребители се появяват автоматично, без презареждане на страницата!

### **Търсачка за потребители**



* **session\_start();** – Започва сесията, за да можем да използваме $\_SESSION (запазените данни на потребителя).
* **include\_once "config.php";** – Включва конфигурационния файл (обикновено съдържа връзката към базата данни).
* Тук вземаме **уникалния идентификатор (ID) на текущия потребител**, за да можем да го сравняваме с други потребители в базата данни.
* Проверяваме дали в заявката има **searchTerm** (текста, който потребителят е въвел в търсачката).
  + **Ако има**, използваме mysqli\_real\_escape\_string($conn, $searchTerm), за да **предпазим от SQL инжекции**.
  + **Ако няма**, просто задаваме $searchTerm = "".
* **$output** – ще съдържа HTML кода с резултатите от търсенето.
* **$uniqueUsers** – масив за **съхранение на вече добавени потребители** (за да не се повтарят).



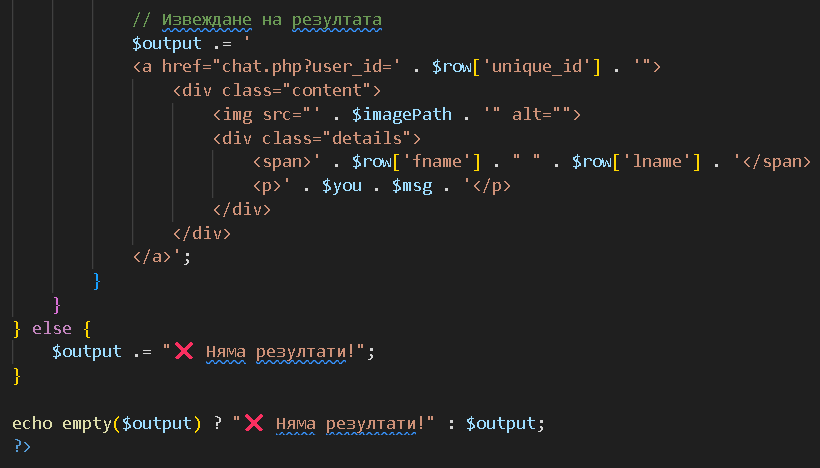
* Тук търсим потребители, **с които настоящият потребител е имал съобщения**.
* Използваме LEFT JOIN messages, за да **вземем само тези, с които има чат**.
* **WHERE u.unique\_id != {$outgoing\_id}** – Изключваме текущия потребител (да не се показва сам на себе си).
* **Търсим по име или фамилия** (u.fname LIKE '%{$searchTerm}%' OR u.lname LIKE '%{$searchTerm}%').
* $query - изпълнява заявката към MySQL и запазваме резултата в $query.

|  |  |
| --- | --- |
| * **Проверяваме дали има резултати** (mysqli\_num\_rows($query) > 0).   + **Ако има**, обхождаме всеки намерен потребител (while ($row = mysqli\_fetch\_assoc($query))). | * **Проверяваме дали вече сме добавили този потребител в $uniqueUsers** (за да не се повтаря). |

|  |  |
| --- | --- |
| * Търсим **последното съобщение** между текущия потребител и този потребител (ORDER BY msg\_id DESC LIMIT 1). * Съхраняваме резултата в $row2. | * **Ако има съобщение**, го запазваме. * **Ако няма**, показваме "No messages available". |

|  |  |
| --- | --- |
| * Ако съобщението е **по-дълго от 28 символа**, го **отрязваме** и добавяме (...). | * Ако последното съобщение **е изпратено от текущия потребител**, добавяме "You: ". |

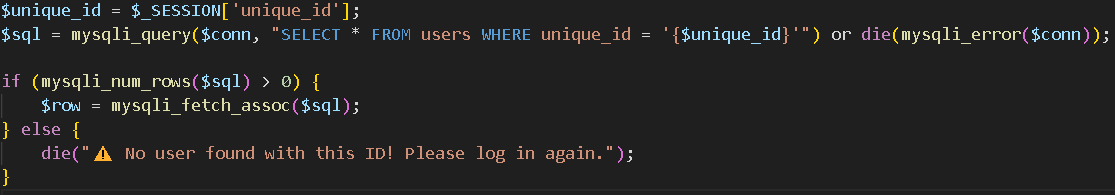
|  |  |
| --- | --- |
| * Проверяваме дали потребителят е **офлайн**, за да приложим съответен CSS стил. | * Генерираме HTML **линк** за всеки намерен потребител. * Показваме **снимка, име, статус и последното съобщение**. |



* Ако **не са намерени потребители**, показваме "❌ Няма резултати!".
* Ако има, извеждаме **списък с потребители**.

## **Редактиране на профил**

Този PHP код представлява **страница за редактиране на профил** в чат приложение. Той проверява дали потребителят е влязъл в системата, извлича неговите данни от базата и позволява промяната на име, фамилия, имейл, парола и профилна снимка.



* $\_SESSION['unique\_id'] съдържа **уникалния идентификатор** на потребителя.
* Създаваме SQL заявка, за да вземем всички негови данни от базата (users).
* mysqli\_fetch\_assoc($sql); ни дава масив $row, съдържащ информацията на потребителя (име, имейл и др.).
* Ако няма потребител с такъв ID → извежда грешка **"No user found"**.

|  |  |
| --- | --- |
| * $\_POST['update'] означава, че потребителят е натиснал бутона за запазване. * Взимаме въведените в полетата данни (fname, lname, email). * Използваме mysqli\_real\_escape\_string(), за да предотвратим SQL инжекции. | * Ако полето за нова парола НЕ е празно → **шифроваме новата парола** с password\_hash(). * Ако няма нова парола → **оставяме старата**. |

|  |  |
| --- | --- |
| * Проверяваме дали потребителят е качил нова снимка. * Ако да: * Проверяваме дали разширението е jpg, jpeg или png. * Проверяваме дали не е по-голяма от **5MB**. * Ако всичко е наред → записваме изображението в папката **"php/images/"**. * Ако не е качено ново изображение → **оставяме старото**. | * Създаваме SQL заявка, която **актуализира** информацията на потребителя. * Ако успешно я изпълним → **пренасочваме потребителя към страницата users.php**. * Ако има грешка → показваме съобщение. |

## **Chat страница с потребител**

Този PHP код **зарежда страницата за чат между двама потребители** в чат приложението. Той:

* Проверява дали потребителят е влязъл в системата.
* Взима ID на другия потребител, с когото ще се чатят.
* Зарежда неговата информация (име, снимка, статус).
* Създава HTML структурата за чат полето.

### **Chat страница:**

|  |  |
| --- | --- |
| * Взимаме user\_id от **URL адреса** (chat.php?user\_id=12345).   + **mysqli\_real\_escape\_string()** предпазва от **SQL инжекции**. * **Търсим в базата данни** потребител с това unique\_id. * **Ако има такъв потребител**, запазваме информацията в $row. * **Ако не съществува**, пренасочваме към **страницата със списък на потребителите (users.php)**. | * **Стрелка за връщане назад** (<a href="users.php">). * **Показваме профилната снимка** (<img src="php/images/..."). * **Изписваме името и статуса на потребителя**. |



* incoming\_id – **ID на потребителя, към когото ще изпращаме съобщението**.
* message – **поле за въвеждане на текст**.
* **Когато натиснем бутона**, съобщението ще се изпрати **чрез AJAX заявка**.

### **Get-chat.php**

**Този PHP код обработва заявките за извличане на чат съобщения между двама потребители в реално време.**

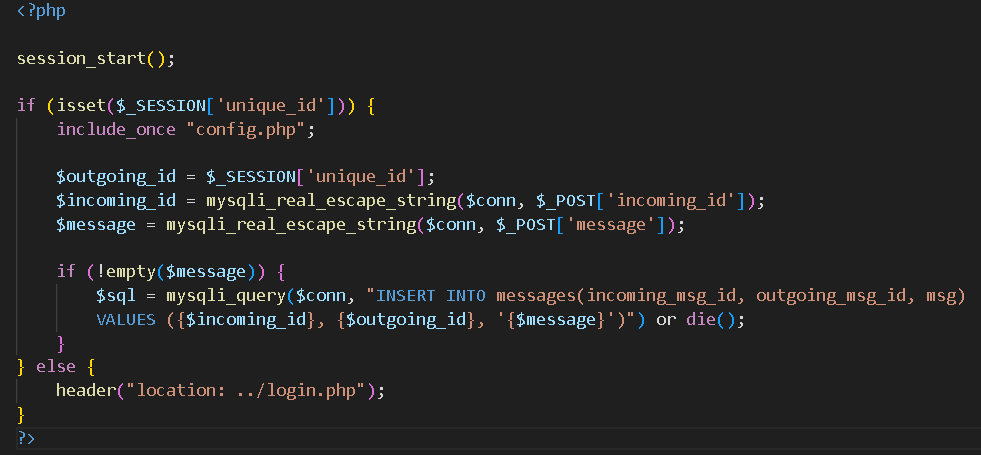
|  |  |
| --- | --- |
| * **$outgoing\_id** – ID на текущия (логнатия) потребител. * **$incoming\_id** – ID на другия потребител (човека, с когото чатим). * Използваме mysqli\_real\_escape\_string(), за да **предотвратим SQL инжекции**. | * **Обединяваме (LEFT JOIN) таблицата messages и users** → Така можем да вземем и профилната снимка (users.img). * **Филтрираме само съобщенията между двамата потребители** * (outgoing\_msg\_id = $outgoing\_id AND incoming\_msg\_id = $incoming\_id) **ИЛИ** * (outgoing\_msg\_id = $incoming\_id AND incoming\_msg\_id = $outgoing\_id) * **Подреждаме резултатите по msg\_id (от най-старото към най-новото).** |

|  |  |
| --- | --- |
| * **Обхождаме всички съобщения** от заявката. * Проверяваме дали съобщението е **изпратено от текущия потребител (outgoing)**: * **Ако е изпратено от нас**, показваме балонче вдясно (outgoing). * **Ако е получено от другия потребител**, показваме го вляво (incoming) + **неговата профилна снимка**. | * Ако все още няма чат съобщения, показваме съобщение **"No messages available"**. * **Изпращаме HTML кода** обратно към AJAX заявката, която го вмъква в чата. |

### **Insert-chat.php**

**Този PHP код се използва за изпращане на съобщения в чат приложението.**

|  |  |
| --- | --- |
| * **session\_start();** – Започваме сесията, за да проверим дали потребителят е влязъл. * **Проверяваме isset($\_SESSION['unique\_id'])** | * **$outgoing\_id** – ID на изпращащия (логнатия) потребител. * **$incoming\_id** – ID на получателя (другия потребител в чата). * **$message** – Самото съобщение. * **Използваме mysqli\_real\_escape\_string()**, за да избегнем **SQL инжекции**. |



* Ако **съобщението не е празно**, записваме го в базата.
* Изпълняваме **SQL заявка**, за да **запишем съобщението в базата**.
* messages има три колони:
* **incoming\_msg\_id** → ID на получателя
* **outgoing\_msg\_id** → ID на изпращача
* **msg** → Самото съобщение
* Ако потребителят **не е логнат**, го **пренасочваме** към login.php.

### **JavaScript код**

Този JavaScript код управлява чата в реално време:

* Обработва **изпращането на съобщения**
* Автоматично **зарежда нови съобщения**
* Управлява **интеракцията със скролиране**

|  |  |  |
| --- | --- | --- |
| * **form** – Взима формата за изпращане на съобщения. * **incoming\_id** – ID на получателя, взето от скритото поле в HTML (<input type="hidden" name="incoming\_id" value="...">). * **inputField** – Полето за въвеждане на текст. * **sendBtn** – Бутонът за изпращане. * **chatBox** – Кутията, където се показват съобщенията. | * Спираме **презареждането на страницата**, когато потребителят натисне Enter или бутона "Изпрати". * Чатът работи **чрез AJAX**, без да се опреснява страницата. | * Ако потребителят **пише нещо**, бутонът "Изпрати" става **активен**. * Ако изтрие текста, бутонът **става неактивен**. |

|  |  |
| --- | --- |
| * Изпращаме **POST заявка** към php/insert-chat.php. * Когато съобщението бъде записано в базата: * **Изчистваме полето за писане (inputField.value = "")**. * Не скролираме автоматично надолу (само при първото зареждане). | * Ако потребителят **е в чата**, добавяме class="active". * Ако потребителят **излезе от чата**, махаме active. * Това е полезно, за да **не скролира автоматично, докато четем стари съобщения**. |

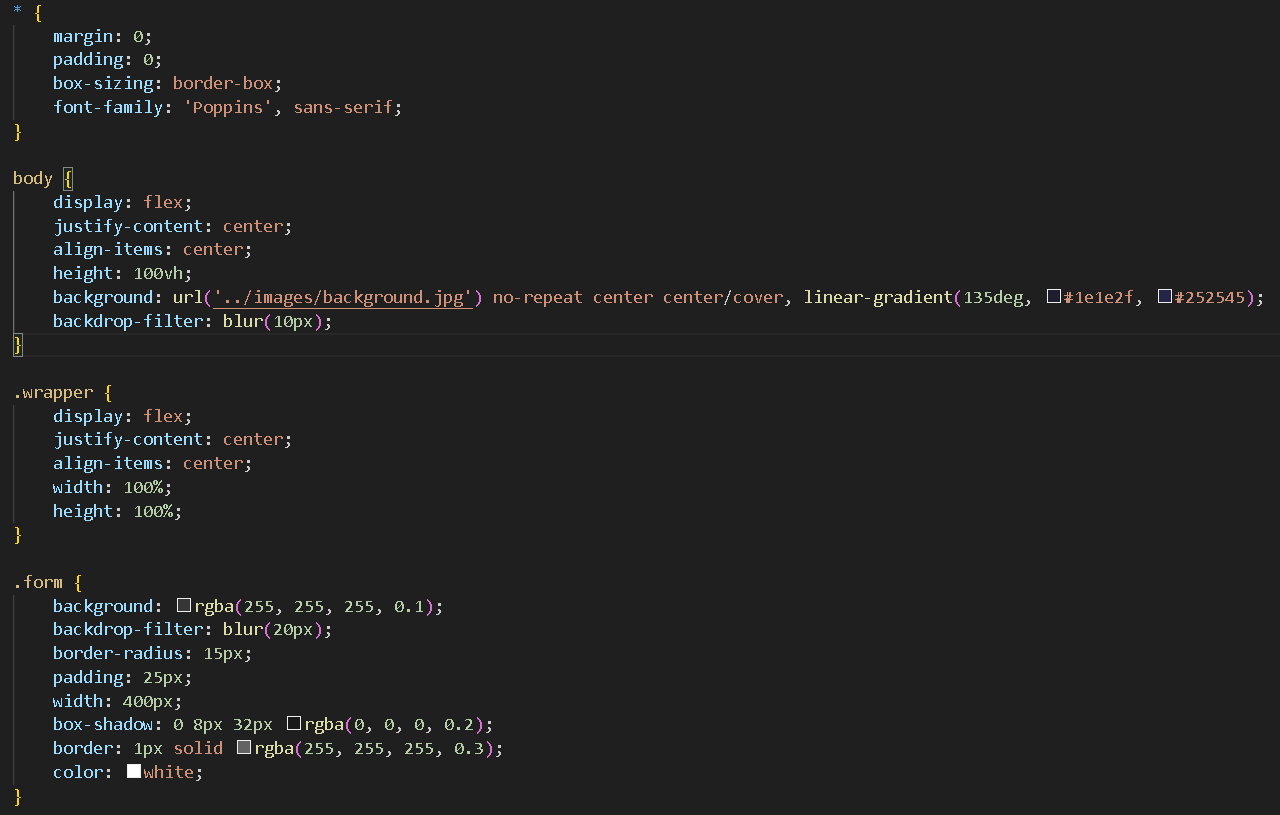
|  |  |
| --- | --- |
| * **На всеки 500ms** (0.5 сек.) изпращаме заявка към php/get-chat.php, за да **обновим съобщенията**. * **Зареждаме чата само веднъж до последното съобщение** (чрез isFirstLoad). * Ако потребителят **не е скролирал нагоре**, ще вижда последното съобщение автоматично. | * Скролира чата **до най-новото съобщение**. * Използва се **само при първото зареждане**. |

## **CSS код**

### **Глобални стилове:**

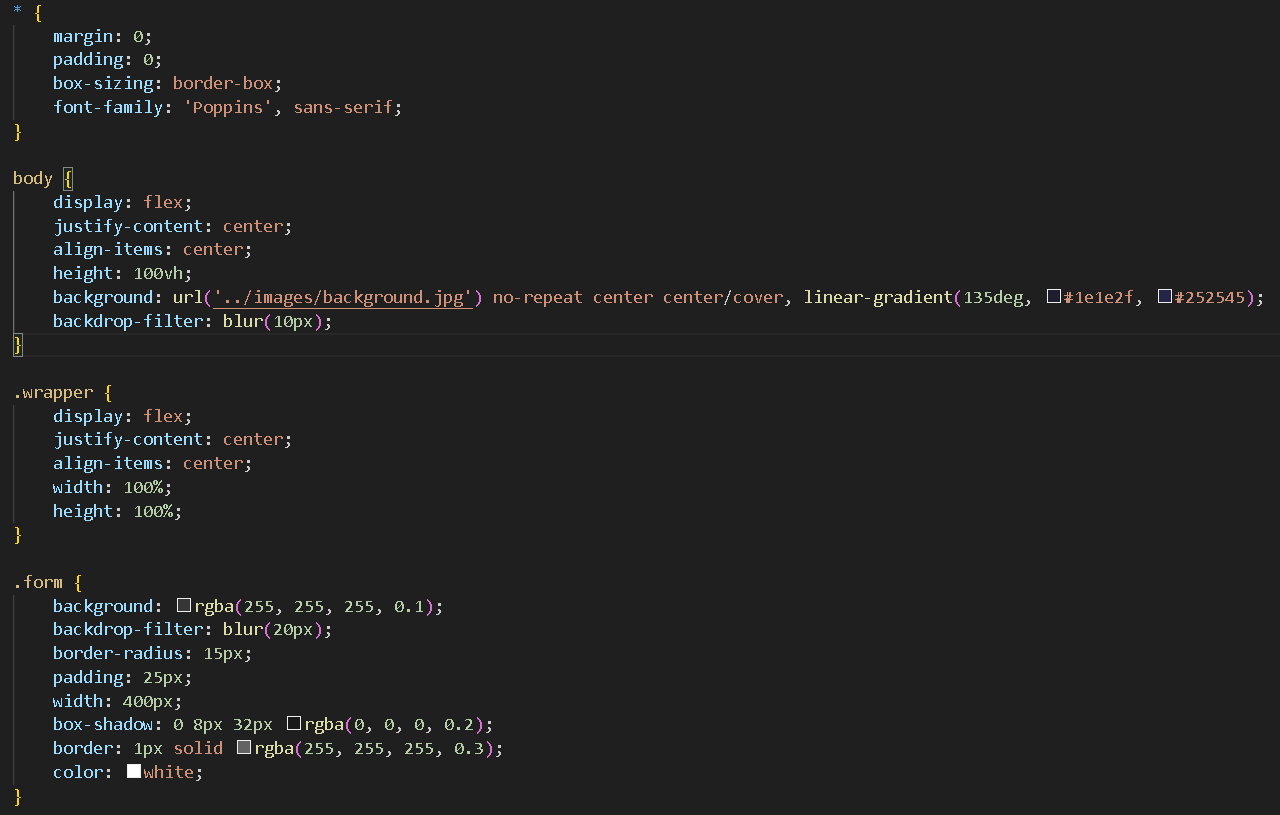
**@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600;700&display=swap');**

Този ред импортира шрифта Poppins от Google Fonts с различни тежести (тънък, нормален, дебел и т.н.). Шрифтът ще се използва в цялото приложение.

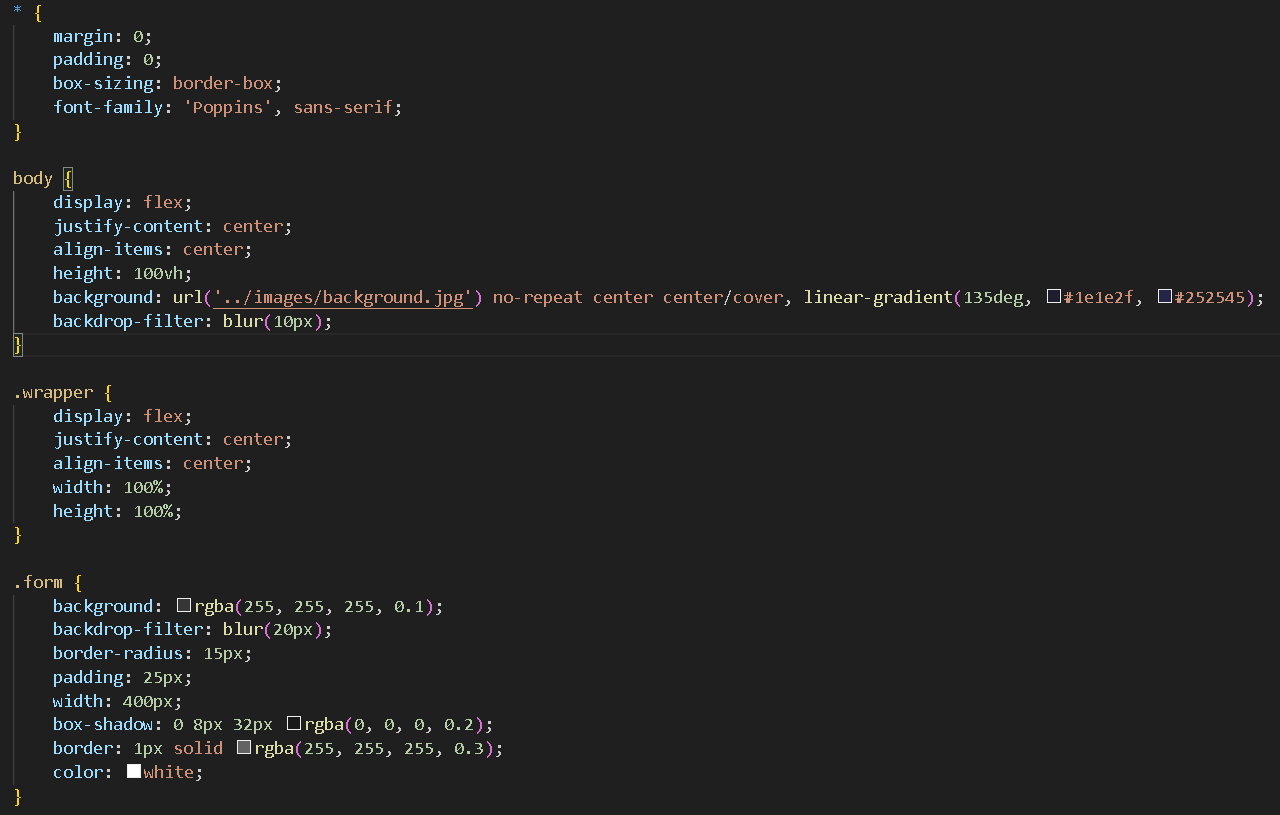


* Тази част нулира **всички отстъпи (margin)** и **паддинги (padding)** на елементите в документа, така че всички елементи да започват от нулеви стойности.
* box-sizing: border-box; настройва всички елементи да се измерват включително вътрешните отстъпи и граници, за да се избегнат неочаквани размери.
* Задава глобалния шрифт на страницата на *Poppins*.

### **Стилове на страницата**



* display: flex; създава флексбокс контейнер, който позволява лесно центриране на съдържанието.
* justify-content: center; и align-items: center; центрират съдържанието хоризонтално и вертикално.
* height: 100vh; прави височината на елемента на цялата височина на екрана.
* background: задава фоново изображение и градиент, който обхваща целия фон.
* backdrop-filter: blur(10px); прилага размазване на фона, което дава ефект на размътен фон зад съдържанието.



* Това е контейнерът, който ще обвие всички елементи. Настройва го като флексбокс, който центрира всички елементи в него както хоризонтално, така и вертикално.

### **Стилове за формуляри**

|  |  |  |  |
| --- | --- | --- | --- |
| * background: rgba(255, 255, 255, 0.1); задава полупрозрачен фон с лек бял оттенък. * backdrop-filter: blur(20px); дава ефект на размазване на фона, зад самата форма. * border-radius: 15px; прави ъглите на формата заоблени. * padding: 25px; добавя вътрешни отстояния около съдържанието в формата. * width: 400px; задава ширината на формата. * box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2); добавя сенка, която да създаде дълбочина. * border: 1px solid rgba(255, 255, 255, 0.3); добавя тънка бяла граница с полупрозрачност. * color: white; задава цвета на текста в формата да бъде бял. | * font-size: 26px; задава размера на шрифта на заглавието. * font-weight: bold; прави текста удебелен. * text-align: center; центрира текста. * margin-bottom: 20px; добавя разстояние под заглавието. *  text-transform: uppercase; преобразува текста в главни букви. | * Добавя разстояние под всяко поле и задава позиционирането като **relative** за по-лесно поставяне на икони. | * display: block; прави етикетите да заемат цялото налично пространство. * font-size: 14px; задава размера на шрифта. * margin-bottom: 5px; добавя разстояние под етикетите. *  color: #fff; прави текста на етикетите бял. |

|  |  |
| --- | --- |
| * width: 100%; прави входните полета да заемат цялата ширина на родителския елемент. * padding: 12px; добавя вътрешни отстояния за по-голямо пространство в полето. * border-radius: 8px; заобля ъглите на полетата. * border: none; премахва стандартната граница на полето. * background: rgba(255, 255, 255, 0.2); дава леко полупрозрачен фон. * color: white; задава белия текст. * outline: none; премахва синята рамка, която се появява при фокусиране на полето. * transition: all 0.3s ease-in-out; добавя плавен преход за стиловете при взаимодействие | * background: rgba(255, 255, 255, 0.3); дава полупрозрачен бял фон. * cursor: pointer; променя курсора при задържане върху бутона. * transition: 0.3s; добавя плавен преход при задържане върху бутона. * font-size: 16px; задава размера на шрифта на бутона. * font-weight: bold; прави текста удебелен. *  text-transform: uppercase; превръща текста в главни букви. |

|  |  |  |
| --- | --- | --- |
| * text-align: center; – центрира текста на линковете. * margin-top: 15px; – добавя отстояние над линка, за да не е залепен за другите елементи. * font-size: 14px; – задава размера на шрифта, за да бъде по-добре четим. | * color: #fff; – задава бял цвят на текста на линковете. * text-decoration: none; – премахва стандартното подчертаване на линковете. * font-weight: 500; – прави текста малко по-дебел. *  transition: color 0.3s ease-in-out; – създава плавен преход при промяна на цвета. | * color: rgba(255, 255, 255, 0.7); – при задържане върху линка цветът става малко по-светъл (70% прозрачност), което създава приятен визуален ефект. |

|  |  |
| --- | --- |
| * Това задава цвета на **placeholder** текста (текстът, който се показва в полето, когато е празно) вътре в елементите с клас .field и тип input. Цветът е бял с 60% прозрачност, благодарение на rgba(255, 255, 255, 0.6). | * Когато входното поле в .field е фокусирано (например, когато потребителят кликне върху него или започне да пише), фонът на полето се променя на бяло с 30% прозрачност (за да се създаде ефект на акцентиране). |

|  |  |
| --- | --- |
| * **position: absolute;** - Иконата ще бъде позиционирана абсолютно спрямо родителския контейнер .field. * **right: 15px;** - Иконата ще бъде разположена на 15px от десния ръб на родителя. * **top: 68%;** - Иконата ще бъде разположена на 68% от височината на родителя. * **transform: translateY(-50%);** - Тази стойност коригира вертикалното позициониране, за да я центрира по височина в родителския контейнер, като я премества нагоре с 50% от височината на самата икона. * **cursor: pointer;** - Когато потребителят премине с мишката над иконата, курсорът ще се променя на "ръка", което показва, че иконата е кликаема. * **color: rgba(255, 255, 255, 0.8);** - Иконата ще бъде с бял цвят, но с 80% непрозрачност. * **transition: color 0.3s;** - Това осигурява плавен преход на цвета на иконата за 0.3 секунди. | * Това задава как ще изглежда иконата при **hover** (когато потребителят премине с мишката върху иконата): * **color: rgba(255, 255, 255, 1);** - Когато потребителят премине с мишката над иконата, цветът ѝ ще стане чисто бял (rgba(255, 255, 255, 1)), без прозрачност. |

### **Секция с потребители**

|  |  |
| --- | --- |
| * **Полупрозрачен фон** – background: rgba(255, 255, 255, 0.1); * **Размазване на фона** – backdrop-filter: blur(20px); * **Заоблени ръбове** – border-radius: 15px; * **Цвят на текста** – color: white; * **Сянка за стилен ефект** – box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2); | * display: flex; – подрежда елементите в ред. * align-items: center; – центрира съдържанието вертикално. * justify-content: space-between; – разпределя елементите така, че да има пространство между тях. * border-bottom: 1px solid #e6e6e6; – добавя долна граница за разделяне на съдържанието. |

|  |  |  |  |
| --- | --- | --- | --- |
| * Задава на изображенията височина и ширина от **50px**. | * display: flex; → Позволява на дъщерните елементи да бъдат подредени в редица. * align-items: center; → Центрира елементите вертикално. | * color: #fff; → Задава бял цвят на текста. * margin-left: 20px; → Добавя **20px** отстояние отляво. | * font-size: 18px; → Задава размер на шрифта **18px**. * font-weight: 500; → Прави шрифта **полу-дебел**. |

|  |  |
| --- | --- |
| * display: block; → Позволява бутона да заема цялата ширина. * background: rgba(255, 255, 255, 0.3); → Прозрачен бял фон. * color: #fff; → Бял цвят на текста. * border-radius: 5px; → Заоблени ъгли. * transition: 0.3s; → Плавен ефект при промяна. | * При задържане върху бутона, фонът става **по-ярък**. |

### **Стилове за търсачката**

|  |  |  |  |
| --- | --- | --- | --- |
| * display: flex; → Подравнява елементите в редица. * justify-content: space-between; → Оставя разстояние между елементите. | * Определя **размера** на текста в полето за търсене. | * opacity: 0; → Скрива полето по подразбиране. * pointer-events: none; → Забранява взаимодействие. * transition: all 0.2s ease; → Прави анимацията **плавна**. | * Когато класът **.show** се добави, полето става **видимо**. |

|  |  |
| --- | --- |
| * cursor: pointer; → Курсорът става **ръка**. * border-radius: 0 5px 5px 0; → Закръгля само **десните** ъгли. | * При **активиране**, бутонът става **тъмен**. |

### **Списък с потребители**

|  |  |  |
| --- | --- | --- |
| * max-height: 350px; → Ограничаваме височината до **350px**. * overflow-y: auto; → Добавя **скролбар**, ако е нужно. | * Скрива **скролбара**. | * display: flex; → Подравнява аватара и името в редица. * transition: 0.3s; → Добавя **плавен ефект** при промяна. |

|  |  |  |
| --- | --- | --- |
| * Премахва **границата** от последния елемент. | * Профилните снимки са **кръгли** и **40px** големи. | * Текстът на последното съобщение е **сив**. |

### **Оформление на скролбар**

|  |  |  |
| --- | --- | --- |
| * Скролбарът става **тънък** (5px). | * Създава **зона** за скрол. | * Цветът на скролбара става **прозрачно бял**. |

### **Оформление на секцията с чат съобщения**

|  |  |  |
| --- | --- | --- |
| * Подобен стил като .users, но за секцията с чата. | * display: flex; – подрежда елементите хоризонтално. * align-items: center; – центрира съдържанието вертикално. * padding: 18px 30px; – осигурява вътрешно разстояние. | * Определя размера на аватара в хедъра на чата. |

|  |  |
| --- | --- |
| * **position: relative;** — задава относително позициониране на елементите в зоната. * **min-height: 400px; и max-height: 400px;** — задава минимална и максимална височина от 400px. * **max-width: 500px;** — задава максимална ширина от 500px. * **overflow-y: auto;** — разрешава вертикално скролиране, ако съобщенията надвишават височината. * **padding: 10px 30px 20px 30px;** — задава отстояния около съдържанието. * **background: #d3d3d3;** — задава светлосив фон. * **box-shadow: inset 0 32px 32px -32px rgb(0 0 0 / 5%), inset 0 -32px 32px -32px rgb(0 0 0 / 5%);** — добавя вградена сянка в областта за чат. * **scroll-behavior: smooth;** — прави скролирането плавно. | * **position: absolute;** — позиционира елемента абсолютно спрямо родителския елемент. * **top: 45%; left: 50%;** — позиционира елемента по средата на родителския елемент. * **width: calc(100% - 50px);** — задава ширина, която е 100% от родителя минус 50px. * **text-align: center;** — подравнява текста по центъра. * **transform: translate(-50%, -50%);** — премества елемента, за да го центрира точно в средата. |

|  |  |
| --- | --- |
| * **margin: 15px 0;** — задава разстояние от 15px между съобщенията. | * **word-wrap: break-word;** — позволява на текста да се разчупва, когато е необходимо. * **padding: 8px 16px;** — задава разстояния около текста. * **box-shadow: 0 0 32px rgb(0 0 0 / 8%), 0rem 16px 16px -16px rgb(0 0 0 / 10%);** — добавя сянка към съобщението. |

|  |  |  |
| --- | --- | --- |
| * **display: flex;** — използва Flexbox за подравняване. | * **margin-left: auto;** — подравнява детайлите вдясно. * **max-width: calc(100% - 130px);** — задава максимална ширина на детайлите. | Ю     * **background: #3e424b;** — тъмен фон. * **color: #fff;** — бял цвят на текста. * **border-radius: 18px 18px 0 18px;** — скругляване на ъглите. |

|  |  |  |
| --- | --- | --- |
| * **height: 35px; width: 35px;** — задава размерите на изображението. | * **display: flex;** — използва Flexbox за подравняване. * **align-items: flex-end;** — подравнява елементите в долния край. | * **margin-right: auto; margin-left: 10px;** — подравнява детайлите вляво с отстояние от 10px отдясно. * **max-width: calc(100% - 130px);** — задава максимална ширина. |

|  |  |  |
| --- | --- | --- |
| * **background: #fff;** — бял фон. * **color: #3e424b;** — тъмно сив цвят на текста. * **border-radius: 18px 18px 18px 0;** — скругляване на ъглите. | * **padding: 18px 30px;** — отстояния около полето за въвеждане. * **display: flex; justify-content: space-between;** — използва Flexbox за подравняване на елементите и разстояние между тях. | * **opacity: 1;** — прави бутона напълно непрозрачен. * **pointer-events: auto;** — позволява кликане върху бутона, когато е активен. |

|  |  |
| --- | --- |
| * **color: #ffffff;** — бял цвят на текста. * **width: 55px;** — задава ширината на бутона. * **background: #2d2f35;** — задава тъмен фон на бутона. * **font-size: 19px;** — задава размер на шрифта. * **cursor: pointer;** — променя курсора на ръка, когато е върху бутона. * **opacity: 0.7;** — задава прозрачен ефект на бутона. * **pointer-events: none;** — забранява кликането върху бутона, когато не е активен. * **border-radius: 0 5px 5px 0;** — скруглява десните ъгли. | * **height: 45px; width: calc(100% - 58px);** — задава височина на 45px и ширина, която е 100% минус ширината на бутона. * **font-size: 16px;** — задава размера на шрифта на 16px. * **border: 1px solid #e6e6e6;** — задава светлосива рамка. * **outline: none;** — премахва контурите при фокусиране. * **border-radius: 5px 0 0 5px;** — скруглява левите ъгли. |

### **Responsive Design**

|  |  |  |  |
| --- | --- | --- | --- |
| * За елементите с класове .form и .users, се задава отстояние от 20px навсякъде, за да се осигури по-добро разстояние при малки екрани. | * Заглавието на формата ще бъде подравнено по център, за да изглежда по-добре на мобилни екрани. | * За елементите с клас .name-details вътре във формата, се променя подреждането на елементите от ред (по подразбиране) на колонка. Това ще направи полетата във формата да са подредени вертикално. | * Премахва отстоянието отдясно на първото поле в .name-details, за да се изравни с другите елементи, когато те са подредени вертикално. |

|  |  |  |  |
| --- | --- | --- | --- |
| * Премахва отстоянието отляво на последното поле в .name-details, за да се изравни с останалите елементи. | * Намалява размерите на изображението в заглавието на .users на 45x45px, за да се побере по-добре на мобилни устройства. | * Променя размера на шрифта и отстоянията на бутона за излизане в заглавието на .users, за да изглежда по-компактно на малки екрани. | * Добавя отстояние от 15px отляво за .details елементите в .content на класовете .users и .users-list. Това създава малко разстояние между съдържанието и лявата страна. |

|  |  |  |  |
| --- | --- | --- | --- |
| * Добавя отстояние от 10px вдясно на линковете в .users-list, за да осигури по-добро подравняване на елементите. | * Намалява отстоянията на заглавието в .chat-area до 15px отгоре и отдолу и 20px отстрани, за да се направи по-компактно на малки екрани. | * Задава минимална височина от 400px за .chat-box и променя отстоянията, за да направи зоната за чат по-удобна за мобилни екрани. | * Намалява размера на шрифта на текстовете в съобщенията на 15px, за да се събере повече текст на малкия екран. |

|  |  |  |  |
| --- | --- | --- | --- |
| * Задава максимална ширина от 230px за изходящите съобщения, за да не излизат извън екрана. | * Задава максимална ширина от 265px за входящите съобщения, като дава малко повече пространство на съобщенията от други потребители. | * Намалява размерите на изображенията на входящите съобщения (например, аватарите) до 30x30px. | * Добавя отстояние от 20px около формата за съобщение в .chat-area. |

|  |  |
| --- | --- |
| * Променя височината на полето за текст на 40px и ширината му да бъде 100% от ширината на родителя, минус 48px (за да остави място за бутона). | * Задава ширина от 45px за бутона за изпращане на съобщението. |