Имена: Стефан Шиваров (фн: 9MI0600175); Анастасия Маджарова (фн: 0MI0600156); Никола Илиев (фн: 1MI0600159)
Начална година: Летен семестър *2025* **Програма:** бакалавър, (СИ) **Курс:** 3
Тема: **28.1**
Дата: 10.06.2025

Предмет: w25prj_SI _final

Имейл: stefan.shivarov.jr@gmail.com, a_madjarova@abv.bg, ndiyanov@uni-sofia.bg

преподавател: доц. д-р Милен Петров

ТЕМА: w24/28.1

1. Условие

Проектът представлява уеб-базирана платформа на име **QuoteShare**, предназначена за събиране, споделяне и организиране на вдъхновяващи или запомнящи се цитати от известни личности. Всеки регистриран потребител може да добавя собствени цитати, да създава лични колекции, да анотира избрани цитати, да ги харесва, да ги докладва при злоупотреба и да експортира избрани колекции във файл.

- Линк към GitHub: https://github.com/nikola-enter21/quoteshare

2. Въведение – извличане на изисквания

Роли:

- **Потребител**
  - Регистрация и вход
  - Добавяне на нови цитати
  - Харесване на цитати
  - Запазване на любими цитати
  - Докладване на неподходящи цитати
  - Създаване на колекции
  - Лични анотации по цитати
  - Добавяне на цитати към колекция
  - Експорт на колекцията в PDF файл
- **Администратор**
  - Преглед на общи статистики за системата
  - Преглед на ТОП 10 най-харесвани цитати
  - Преглед на всички докладвани цитати
  - Изтриване на цитати
  - Изтриване на потребители
  - Промяна на роля на потребител

- Визуализация на всички предприети действия в системата под формата на логове

Функционални изисквания:

1. Системата трябва да позволява на всеки потребител да се регистрира чрез уникално потребителско име и парола.
2. Системата трябва да валидира въведените данни при регистрация – потребителското име не може да се повтаря, а паролата трябва да бъде с минимална дължина от 6 символа.
3. След регистрация потребителят трябва да може да влезе в системата чрез формуляр за вход.
4. След успешен вход системата трябва да предоставя достъп до пълната функционалност на потребителската роля.
5. Системата трябва да позволява на потребителя да добавя нов цитат, като въведе следните задължителни данни: заглавие, съдържание и автор.
6. Системата трябва да визуализира списък с всички цитати, подредени по дата на добавяне (най-нови най-отгоре).
7. Всеки потребител трябва да може да харесва цитати, като всеки потребител има право на едно харесване на даден цитат.
8. Всеки потребител трябва да може да докладва даден цитат чрез натискане на бутон за докладване.
9. Системата трябва да позволява на всеки потребител да създава неограничен брой лични колекции от цитати.
10. Всеки потребител трябва да може да добавя и премахва цитати от собствените си колекции.
11. Системата трябва да позволява на всеки потребител да може да се добавя анотация – бележка под формата на текст към всеки цитат.
12. Системата трябва да позволява на всекки потребител да може да разглежда вече добавените анотации към цитат.
13. Всеки потребител трябва да може да експортира избрана колекция във формат PDF, като файлът съдържа списък с цитати и съответните анотации.
13. Системата трябва да предоставя отделен административен панел, достъпен само за администратор.
14. Администраторът трябва да може да вижда всички докладвани цитати.
15. Администраторът трябва да може да изтрива цитати.
16. Администраторът трябва да може да променя ролите на потребител
17. Администраторът трябва да може да изтрива потребителски акаунти
18. Администраторът трябва да може да вижда всички направени действия в системата под формата на логове
19. Администраторът трябва да може да изтрива логове
20. Системата трябва да показва на администратора статистика с:
    - общ брой активни потребители и цитати

- най-харесвани цитати

Нефункционални изисквания:

- Добра разделимост на логика, изглед и данни.
- Конфигурируемост и разширяемост на архитектурата.
- Интуитивен потребителски интерфейс.

Ползи от реализацията:

а) Ползи за разработчиците

- **Придобиване на практически опит в изграждането на уеб приложения** с трислойна архитектура MVC (модел - изглед - контролер).
- **Изучаване и използване на PHP сесии.**
- **Управление на зависимости чрез Composer**, което отговаря на реалните практики в съвременната PHP екосистема.
- **Изграждане на система с потребителски роли**, което включва автентикация и оторизация.
- **Работа с релационна база от данни (Postgres)**, включваща свързани таблици.
- **Практикуване на добри DevOps практики** - използване на Docker за пускане на база данни в контейнер, което е модерна и стандартна практика. Самото уеб приложение също е контейнеризирано. Създали сме скрипт *start.sh*, който пуска двата контейнера, свързва ги и пуска миграциите. Така приложението се пуска само с изпълнението на 1 команда - лесно и бързо.
- **Управление на проект** - проектът е документиран изцяло, лесно се конфигурира и пуска.
- **Работа в екип и разпределяне на задачи.**

а) Ползи за крайния потребител

1. **Интуитивен и лесен за използване интерфейс** за събиране и организиране на любими цитати.
2. **Персонализирани колекции с анотации**, които превръщат платформата в личен архив на вдъхновения и размисли.
3. **Функционалност за експортиране** на колекциите във формат PDF– подходящо за личен архив, разпечатване или споделяне.
4. **Възможност за социално взаимодействие чрез харесвания**, което дава индиректна оценка на популярността и въздействието на цитатите.
5. **Сигурност и качество на съдържанието**, осигурени чрез система за докладване и модериране от администратор.

3. Теория – анализ и проектиране на решението

В хода на разработване, беше предприето имплементирането на следните шаблони:

1. **Архитектурен шаблон: MVC (Model-View-Controller)**

   Приложението е организирано според **MVC (Model-View-Controller)** архитектурен шаблон, който предлага ясно разделение на отговорностите и улеснява поддръжката и разширяемостта на системата.

   Предимства от използването на MVC:

   - Улеснява тестването и модулното разширяване на функционалности.
   - Осигурява разделяне на отговорностите (separation of concerns).
   - Подобрява възможностите за работа в екип – всеки може да работи независимо по слой.

2. ***Composer – управление на зависимости***
   - Автоматично зареждане на класове чрез PSR-4 autoloading.
   - Организация на конфигурацията и разработката според съвременните PHP практики.
3. ***Docker + PostgreSQL – среда за разработка***

   В проекта се използва Docker само за стартиране на базата данни PostgreSQL. Това позволява:

   - Изолирано и контролирано стартиране на СУБД.
   - Лесна настройка без нужда от ръчно инсталиране на PostgreSQL на всяка разработваща машина.
   - Уеднаквена конфигурация за всички разработчици.

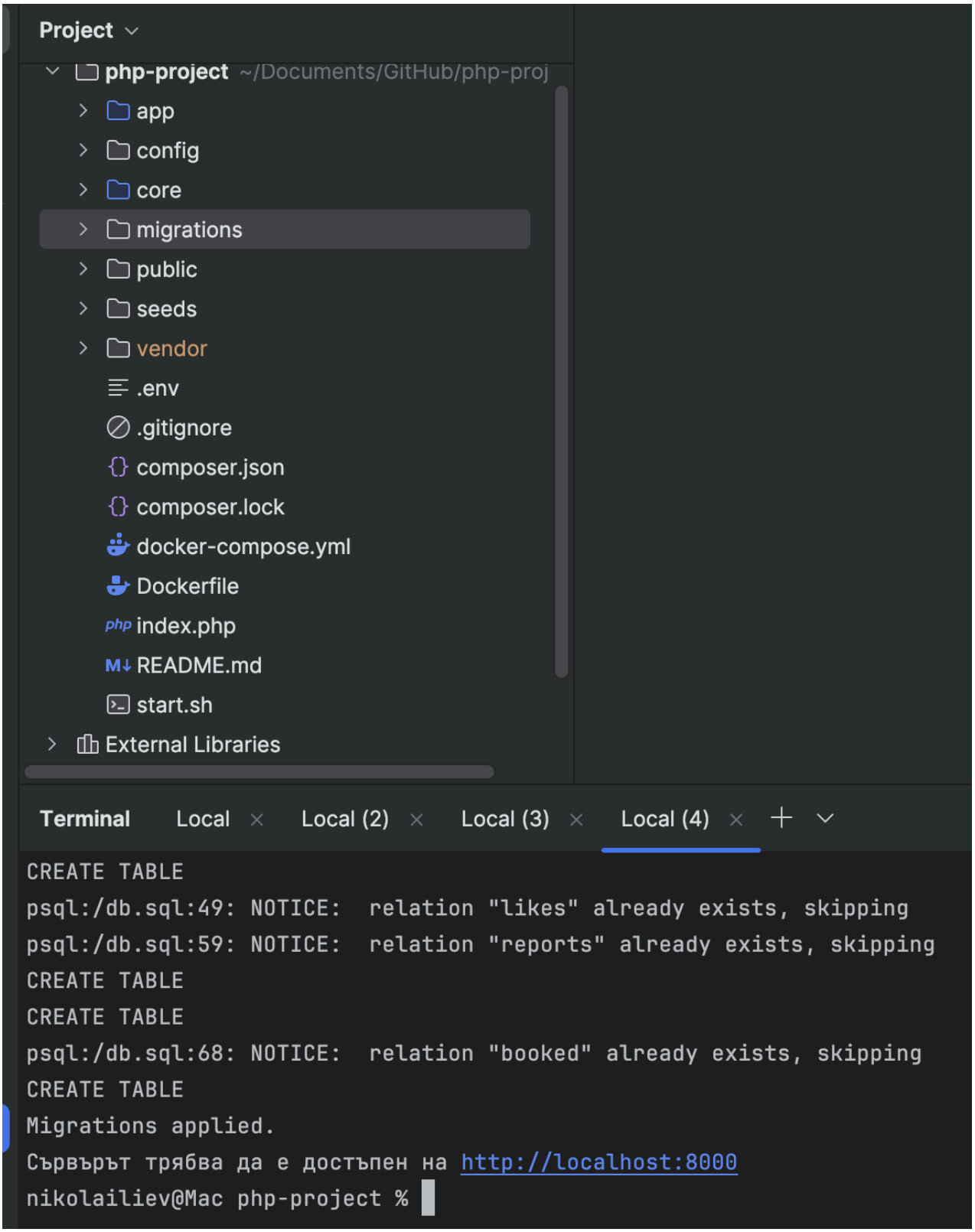
4. Dependency Injection (DI) контейнер за контролери

Контролерите в приложението не създават сами своите зависимости, а ги получават отвън чрез **DI контейнер**. Това позволява централизирано управление на свързани обекти и улеснява промени в логиката без нужда от пренаписване на самите контролери.

4. Използвани технологии

Компонент	***Технология***
Език за програмиране	PHP 8.4
Среда за разработка	PhpStorm
Уеб сървър	Вграден PHP сървър (php -S localhost:8000)
Бази данни	PostgreSQL (Docker образ: postgres:latest)
Управление на зависимости/библиотеки	Composer 2.8.6
Управление на Docker контейнери и образи	Docker Desktop 4.38.0

5. Инсталация, настройки и DevOps



1. Системни изисквания:
    - Docker / Docker Desktop
2. Пускане на проекта стъпка по стъпка
    - Подсигуряваме се, че docker е инсталиран и че е пуснат

- Отиваме в root директорията на проекта и пускаме скрипта **start.sh**. За да пуснем скрипта можем да напишем в терминала ***./start.sh.*** Това ще създаде два контейнера - един за базите данни (PostgreSQL) и един за уеб приложението. След това ще пусне миграциите, които ще създадат нужните таблици в базата данни **и един админ акаунт.**

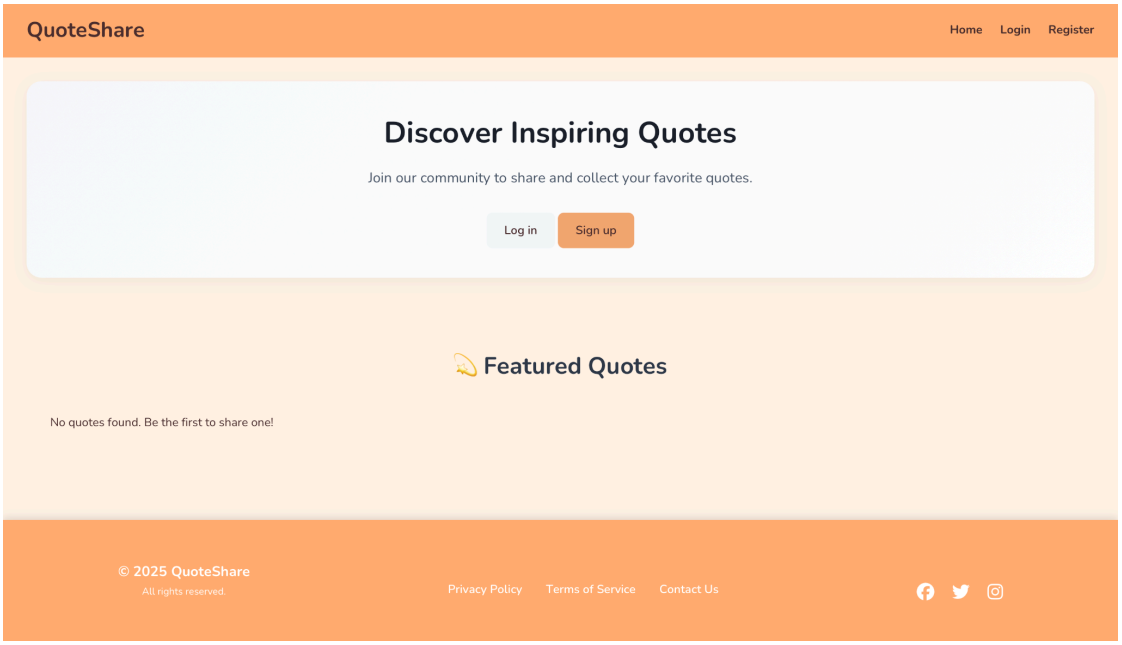
3. Отваряме проекта на адрес: http://localhost:8000

Акаунт за администратор:

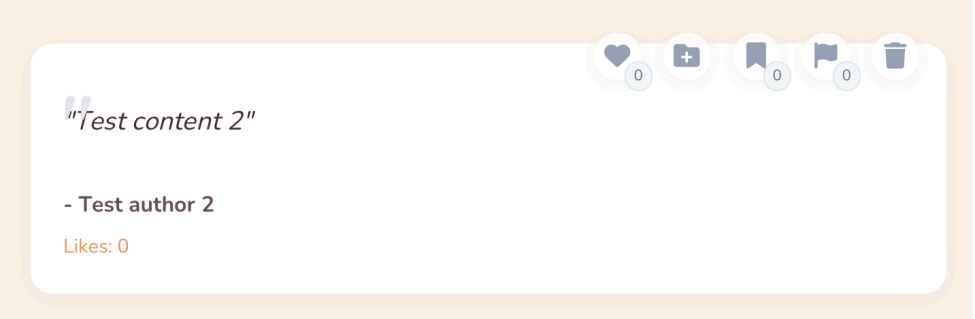
- **потребителско име:** admin@quoteshare.com
- **парола:** admin123

6. Кратко ръководство на потребителя

- Начална страница - тя има два типа в зависимост от това дали потребителят се е вписал в системата или не
    - Ако потребителят не е вписан в системата, той вижда:

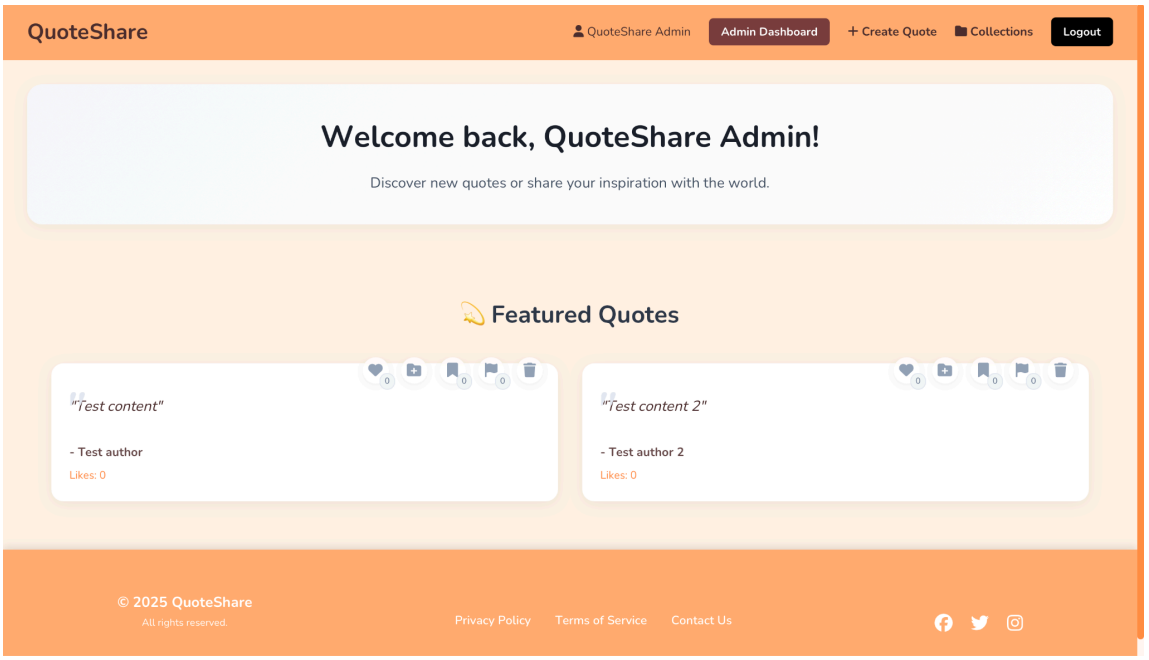


- Имаме меню най-отгоре. В лявата част на менюто виждаме името на платформата QuoteShare. Когато се натисне то ни отвежда към началната страница. В дясната част на менютото имаме Home, Login, Register. Когато потребител натисне Home, бива отведен до началната страница. При натискане на Login зареждаме страница за вход, а при натискане на Register зареждаме страница за създаване на акаунт в нашата система.
- Под навигацията имаме банер (hero section), който ясно обяснява на потребителя за какво е платформата, каква е целта. Имаме два бутона под нея за вход и за регистрация.
- Под тази секция се зареждат всички цитати, добавени от различни потребители. Всеки цитат може да бъде харесан, добавен към любими, докладван на администратора или добавен към лична колекция. Изтриването на цитат е възможно само за администратора или за потребителя, който го е създал. Можем да го добавим към наша колекция. Най-отдолу на цитата се показва общия брой харесвания.



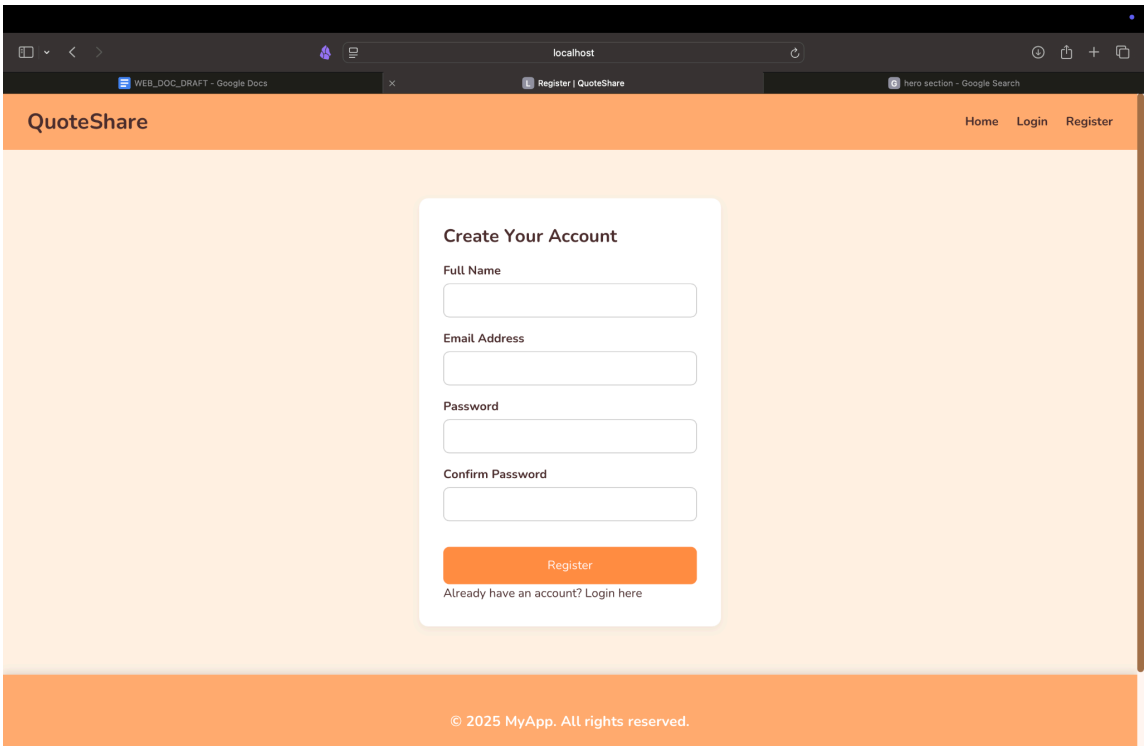
Когато не е вписан потребителя в системата и натисне тези действия - бива прехвърлен към страницата за вход.

- Ако потребителят е вписан в системата, той вижда:



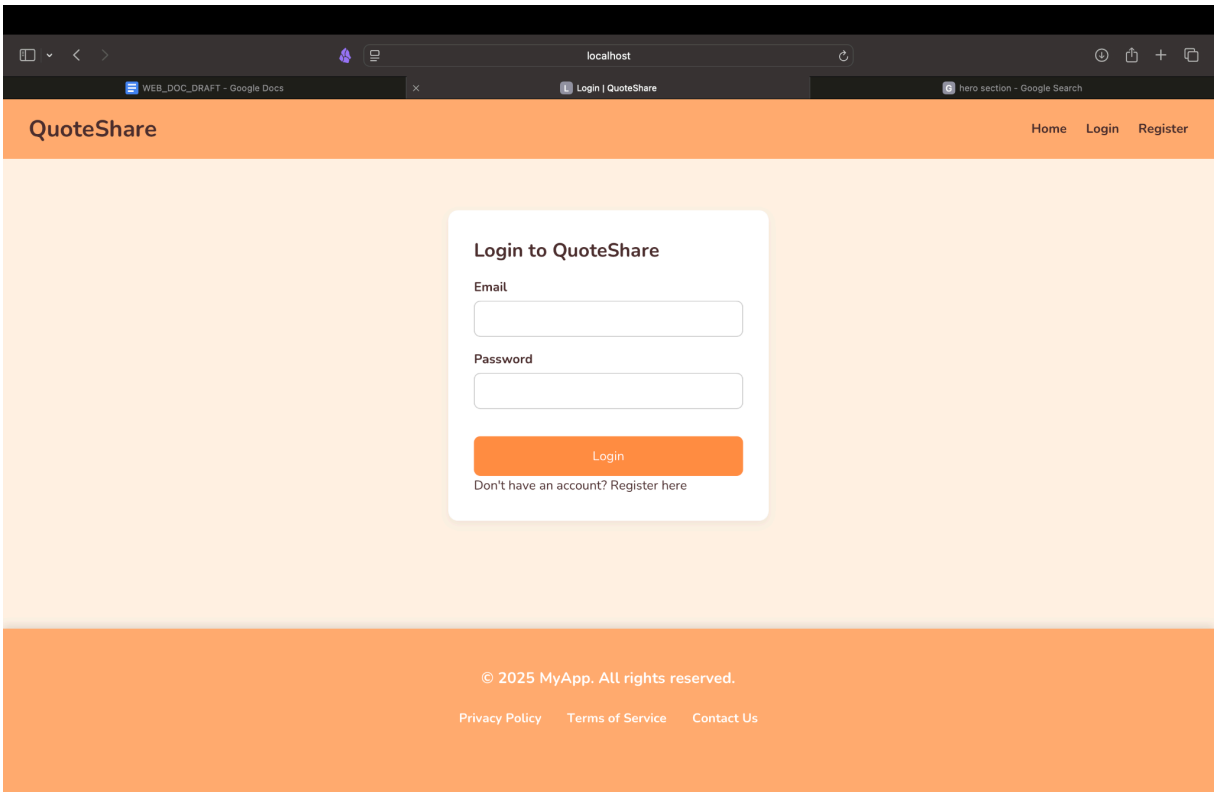
- Единствената разлика тук е, че горе дясно в менюто се премахват опциите за вход и регистрация. Сега се показва името на логнатия потребител, бутон, който води към админ панела (ако сме админ), линк към страницата за създаване на цитат, линк към страницата за създаване на колекция и бутон за изход от системата.
- Банерът вече показва лично съобщение на потребителя “Добре дошъл обратно!”. В момента потребителят е логнат и може да натиска и изпълнява бутоните, които са над всеки цитат - “Харесване/Запазване/Докладване/Добавяне към колекция”. Допълнително има бутон за изтриване, който може да бъде извикан само от администратор или ако потребителят е собственик на цитата.

- Страница за регистрация



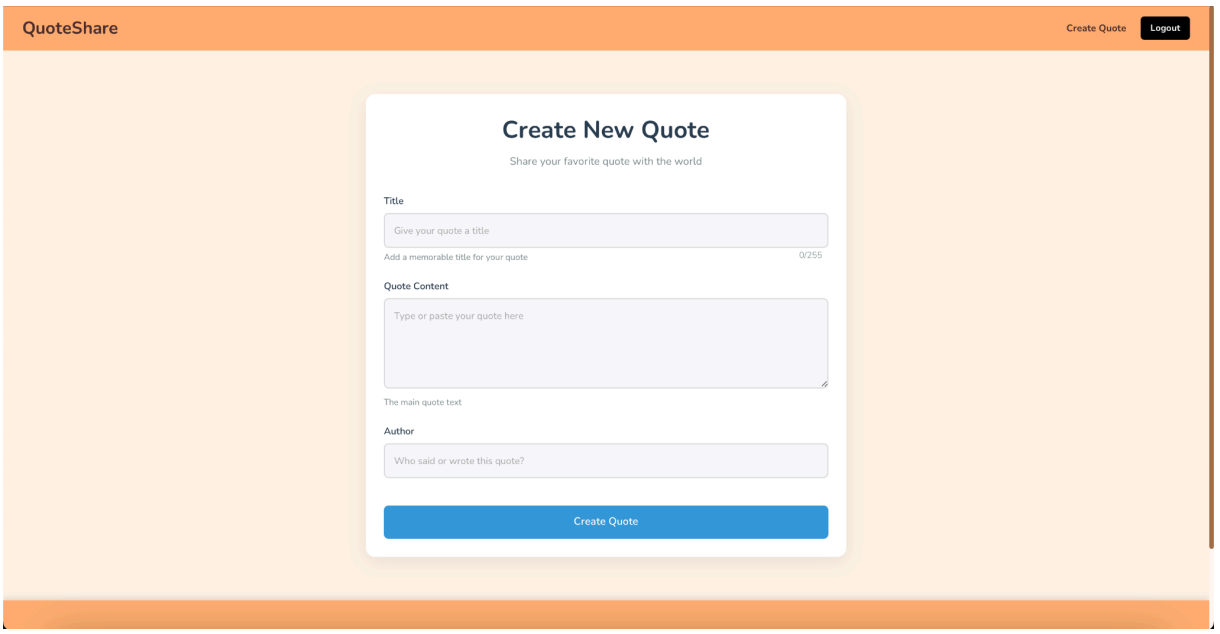
  - Потребителят въвежда своите имена, имейл адрес, парола и потвърждение на паролата. Когато се натисне бутона за регистрация формата се изпълнява и потребителят бива насочен към страницата за вход само ако данните са валидни.

- Страница за вход



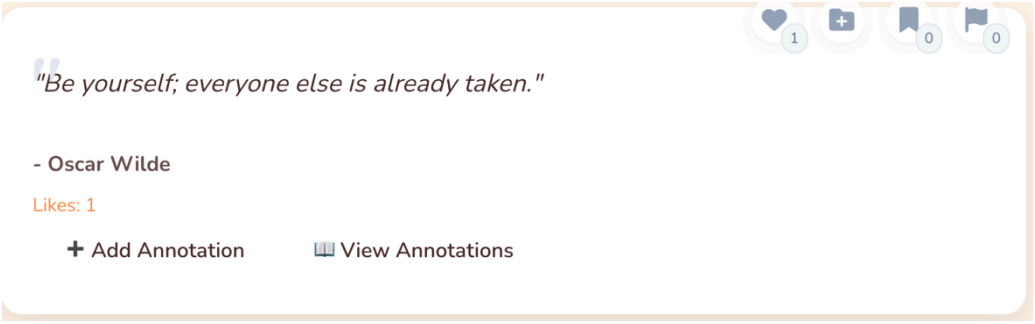
  - Потребителят въвежда своя имейл адрес и парола с които се е регистрирал в системата. След като натисне “Login” ако данните са правилни ще му бъде създадена PHP сесия и ще бъде вкаран в системата успешно.

- Създаване на цитат

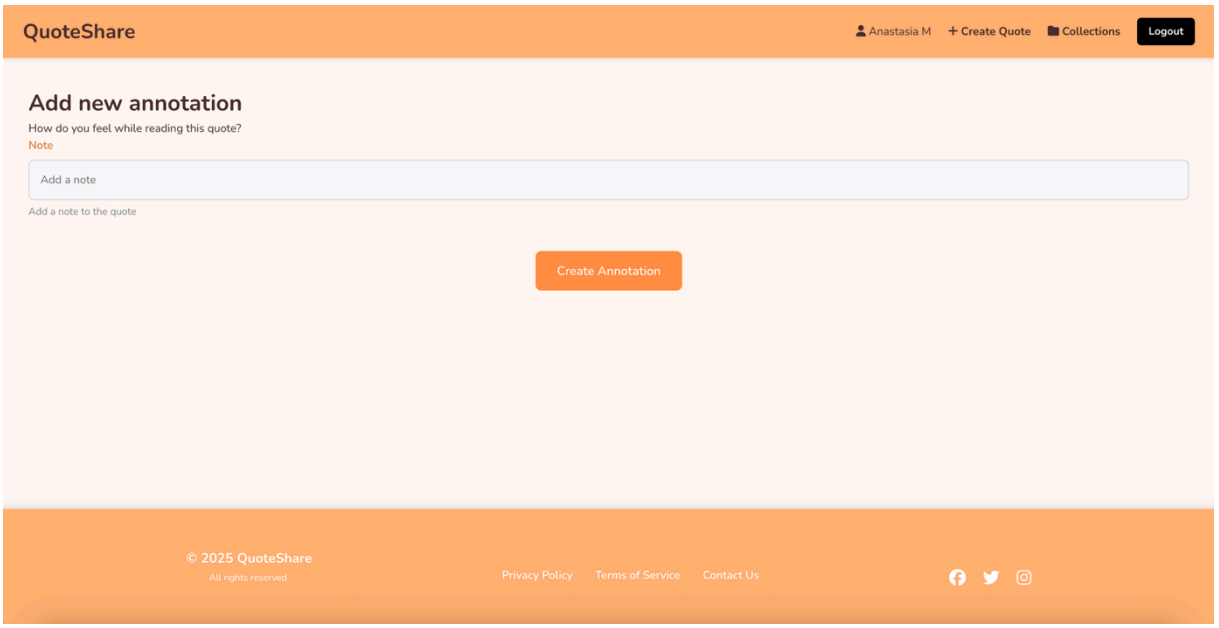


- За създаването на цитат е необходимо потребителят да въведе заглавие (title), съдържание на цитата (content) и автор на цитата (на кого е). След като бъде натиснат бутона Създай цитат ако всички данни са коректни ще бъде прехвърлен потребителя към началната страница, където ще види своя нов качен цитат.

- Добавяне на анотация към вече съществуващ цитат

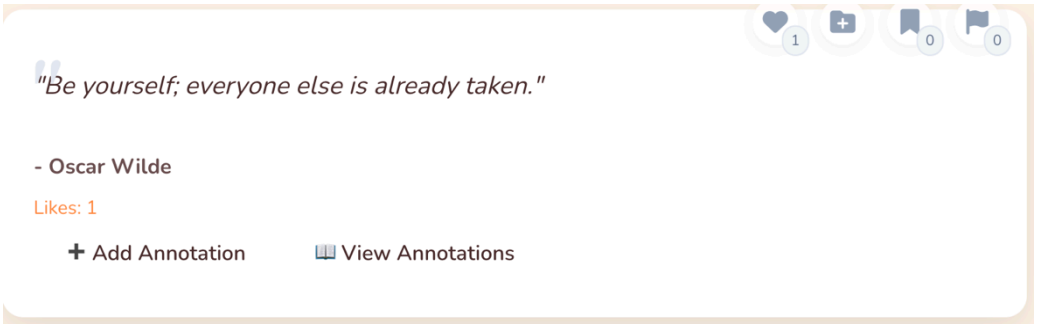


- След натискане на бутона „Add Annotation“, който е наличен за всеки цитат в началната страница, потребителят бива препратен към страница където трябва да въведе бележка за анотацията, която иска да добави

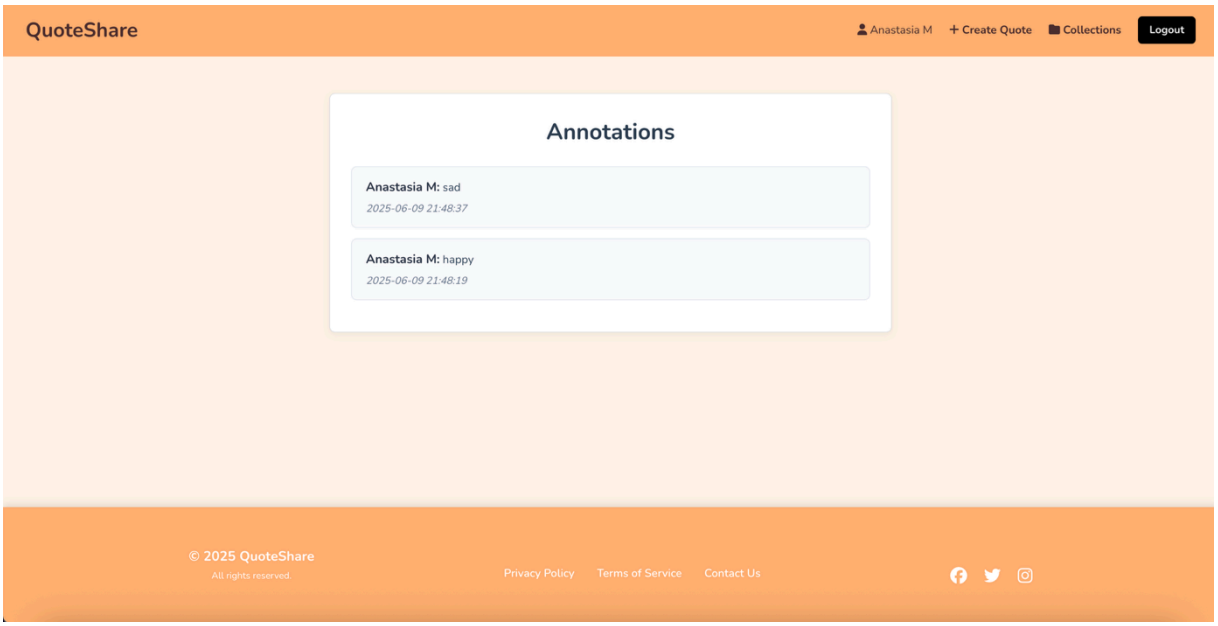


- За създаването на анотация е необходимо потребителят да въведе бележка (note). След като бъде натиснат бутона за създаване на анотация „Create Annotation“, ако всички данни са коректни, анотацията ще бъде създадена успешно и потребителят ще бъде пренасочен към началната страница.

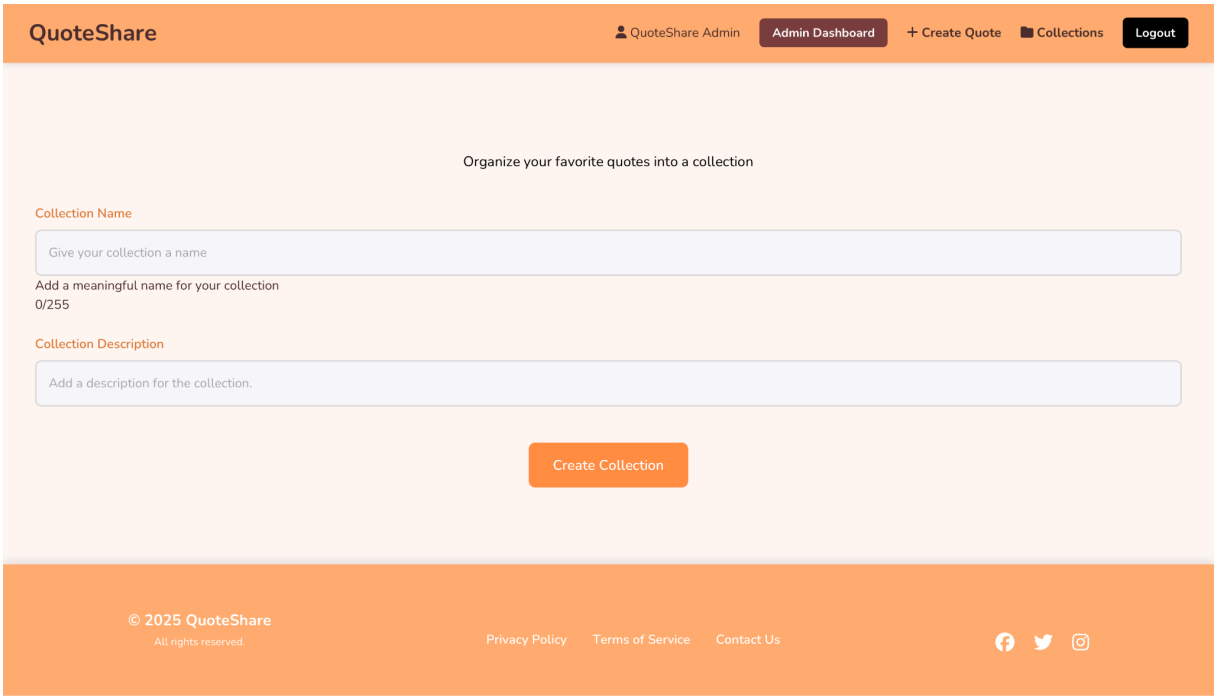
- Разглеждане на добавени анотации към цитат



- След натискане на бутона „View Annotations“ който е наличен за всеки цитат в началната страница, потребителят бива препратен към страница където може да види всички анотации добавени към даден цитат

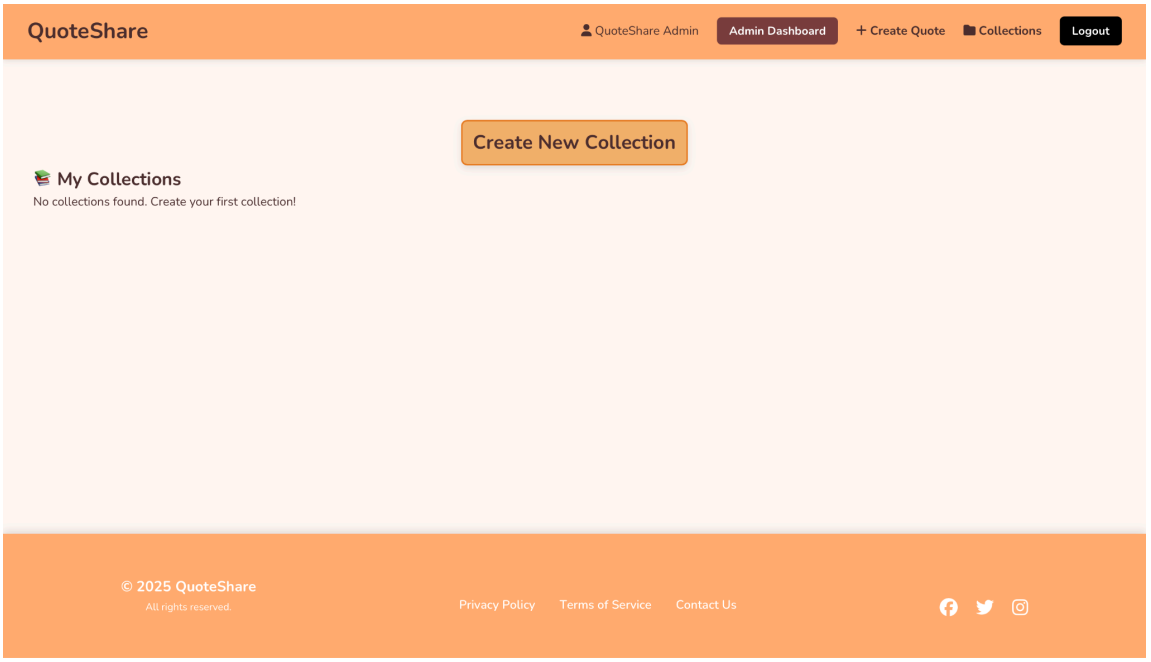


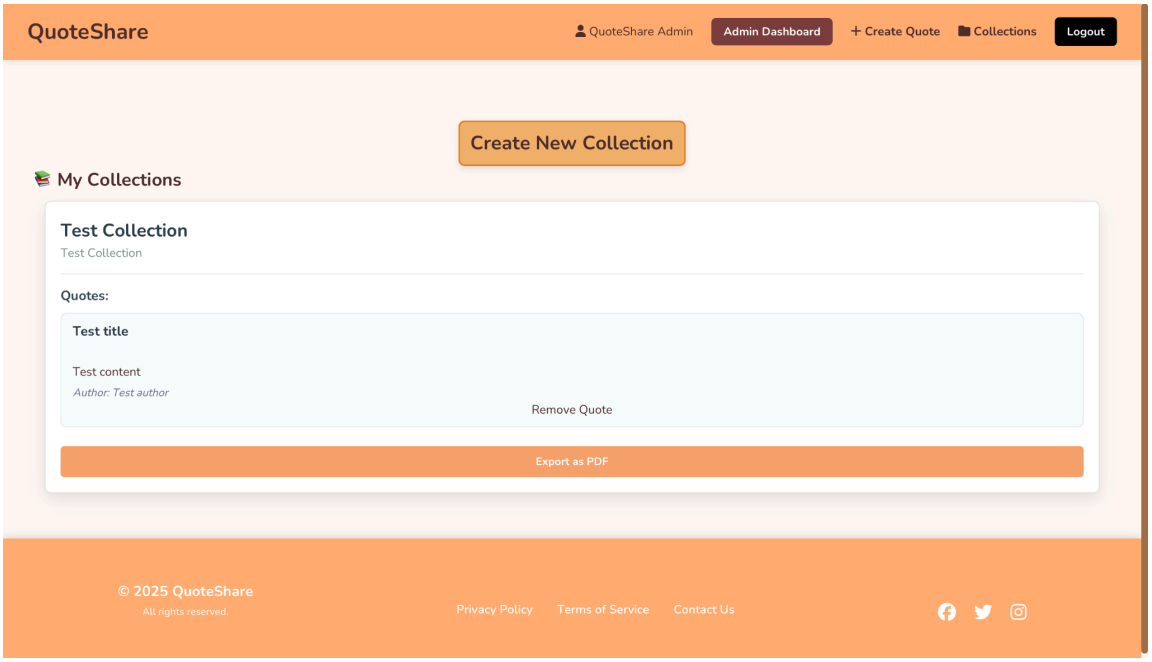
- Създаване на колекция



- За създаването на колекция е необходимо потребителят да въведе име (name) и описание (description) на колекцията. След като бъде натиснат бутона за създаване на колекцията (Create Collection), ако всички данни са коректни, колекцията ще бъде създадена успешно и потребителят ще бъде препратен към страницата с колекции /collections, където може да види новосъздадената си колекция.

- Страница „Collections“





- Най-отгоре на страницата се вижда бутонът „Create Collection“ за създаване на колекция, след натискането му потребителят ще бъде препратен към страницата /collections/create, и може да създаде колекция по начина описан в предишната точка.
- В секцията „Featured collections“ могат да се видят всички колекции на текущия потребител като всяка колекция съдържа име, описание и цитати, които са добавени.
- Потребителят може да премахва цитат от колекцията чрез натискане бутона „Remove form collection“, който се намира под съответния цитат.
- Всяка колекция след натискане на бутона „Export as pdf“, който се намира под съответната колекция , се експортира в pdf файл, който да съдържа

името, описанието и цитатите в дадена колекция. Името на pdf файла е <име на колекцията>.pdf

---

Page 1 of 1

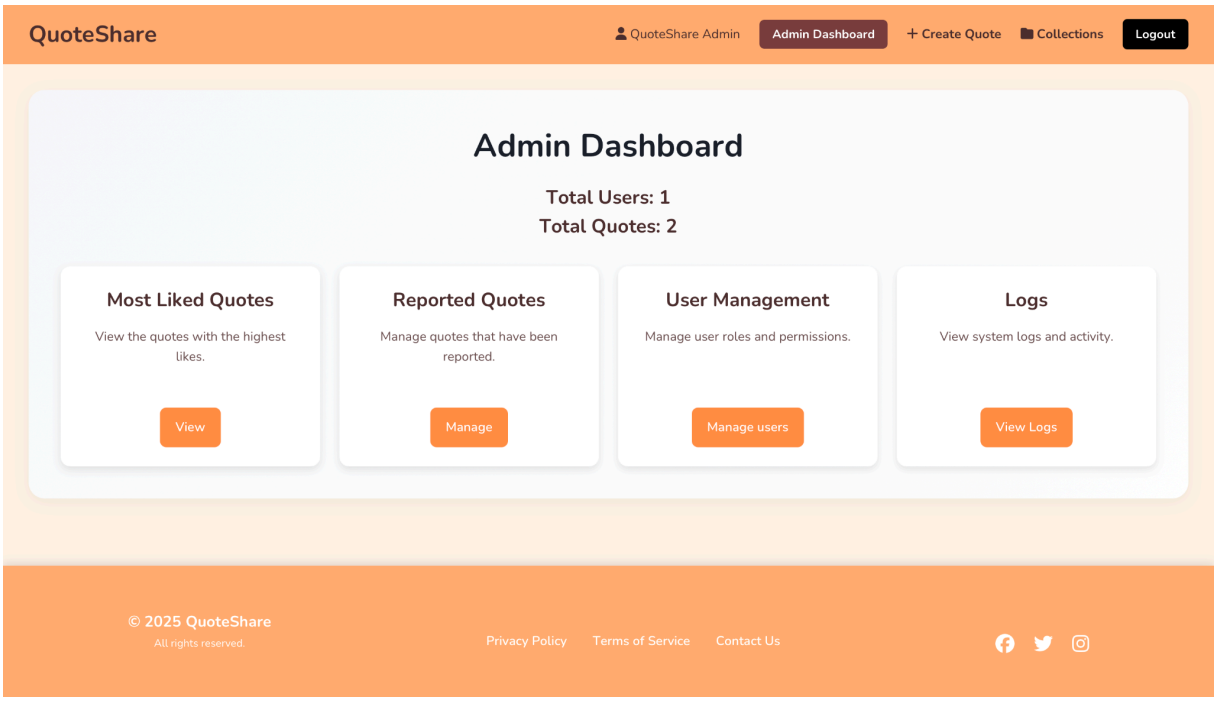
Test Collection

Test Collection

Quotes:

- **Test title**
  Test content
  Author: Test author

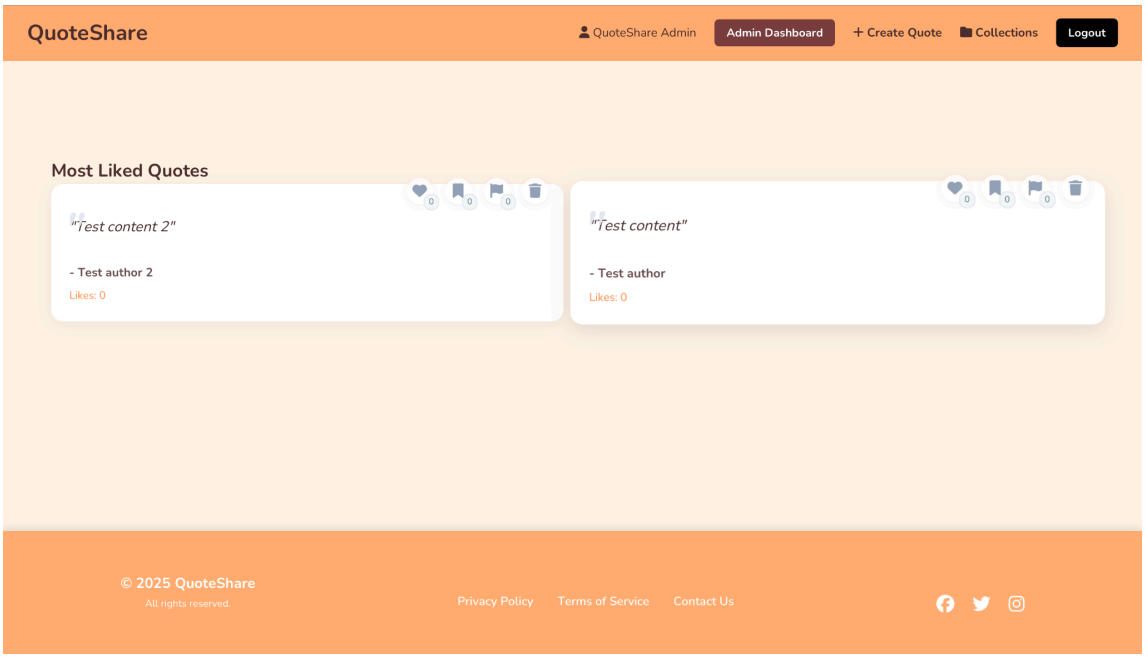
- Админ панел - начална страница



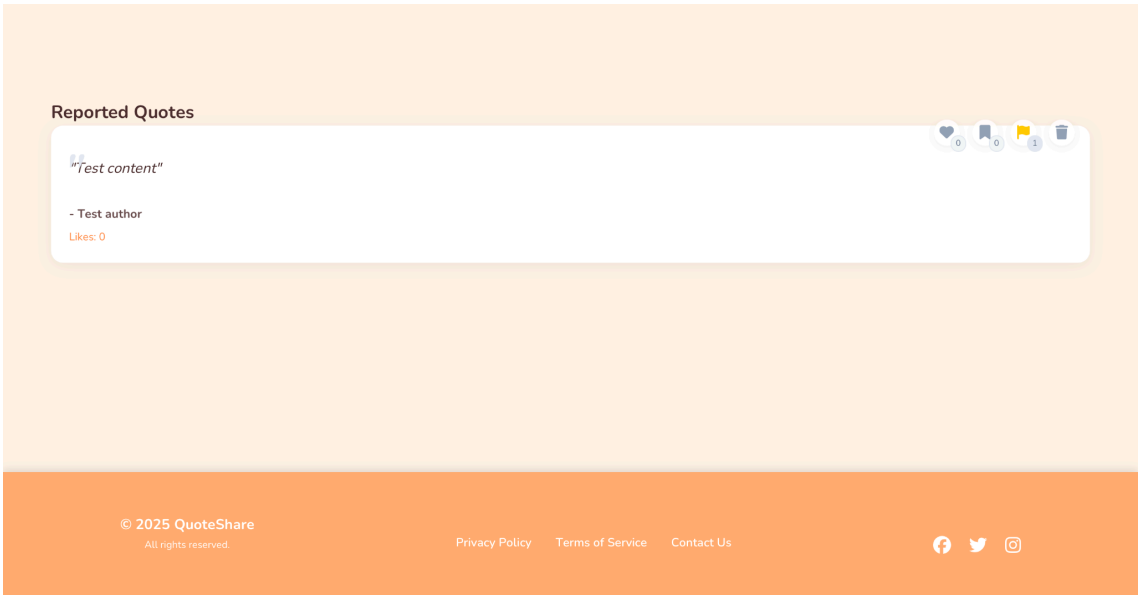
  - Админ панелът може да бъде достъпен единствено от потребители, които имат ролята “admin”.
  - В админ панела се показва общия брой потребители общия брой цитати в системата.
  - Отдолу имаме четири секции - най-харесвани цитати, докладвани цитати, управление на потребители, логове.

- Админ панел - най-харесвани цитати

- В тази страница показваме ТОП 10 най-харесвани цитати

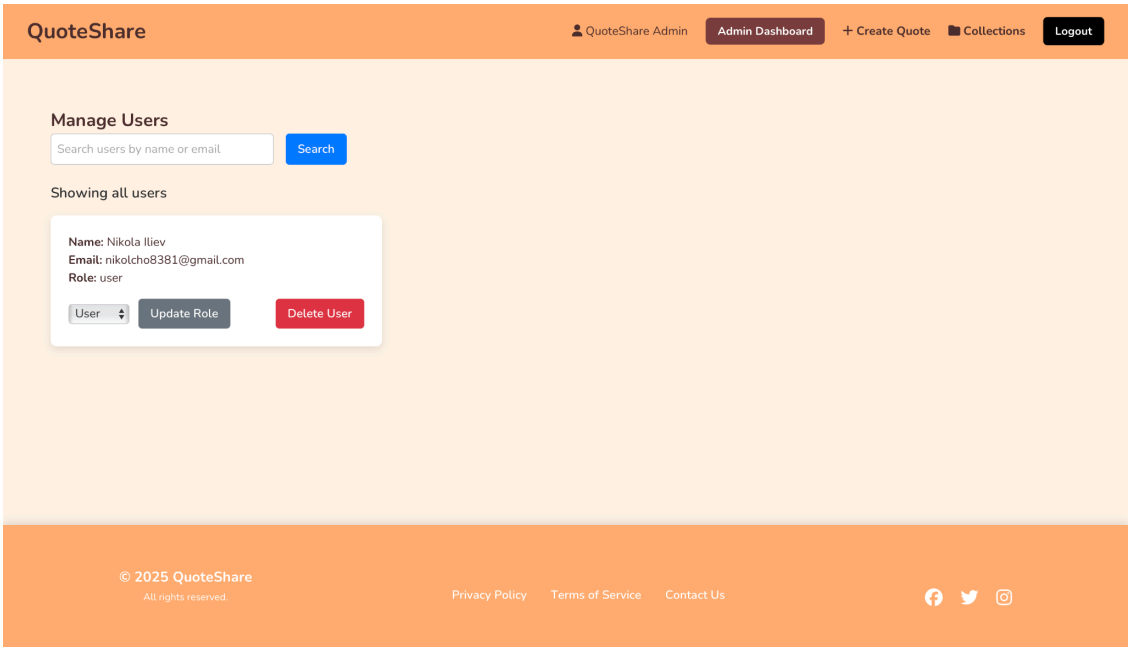


- Админ панел - докладвани цитати
  - В тази страница показваме всички докладвани цитати и администраторът може да реши да изтрие даден цитат ако прецени, че има причина.

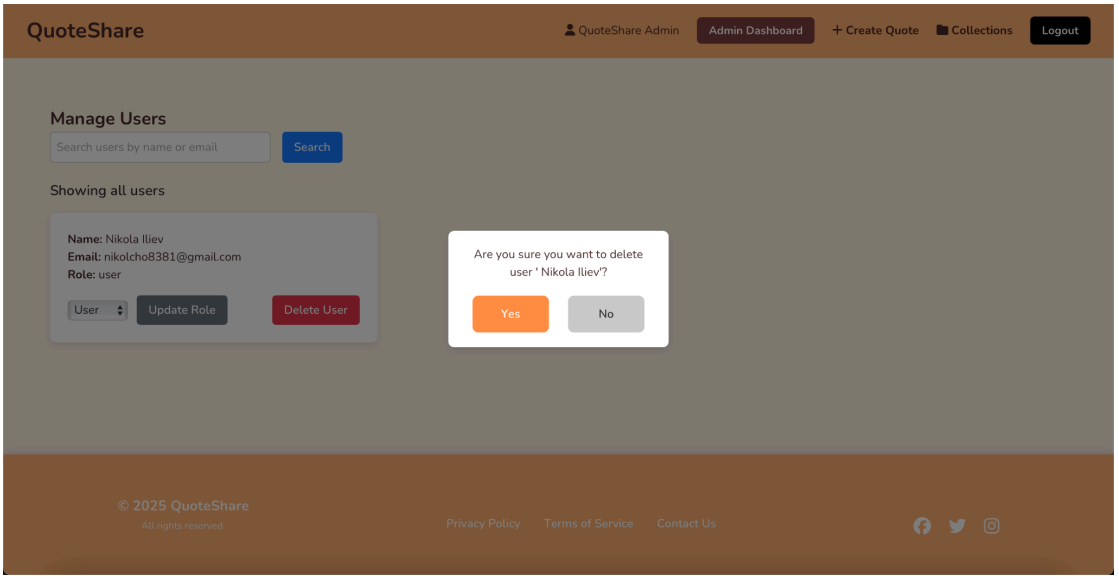


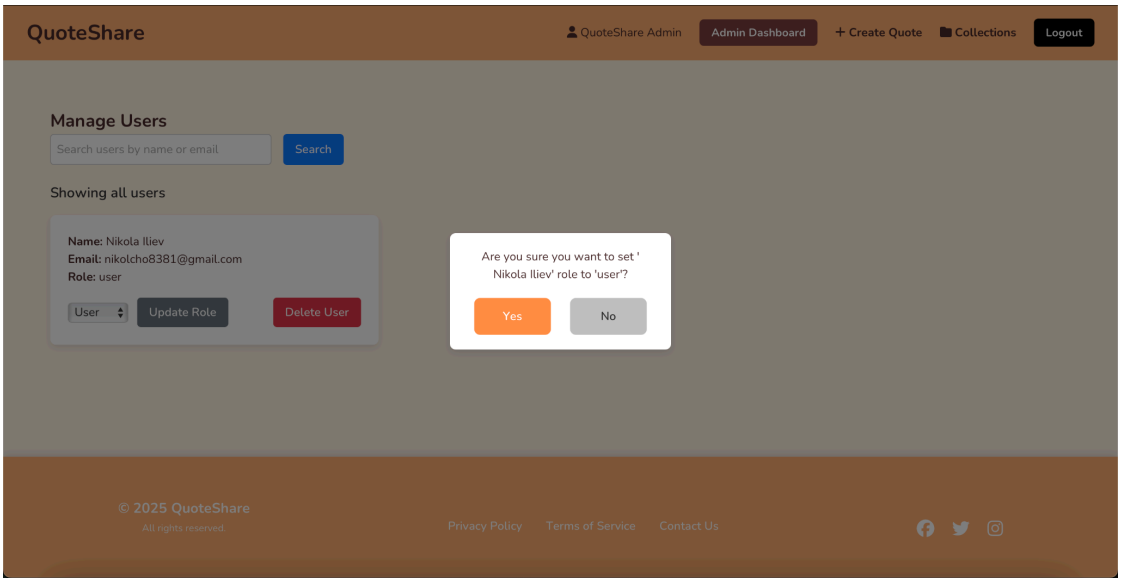
- Админ панел - управление на потребители

- В тази страница показваме всички регистрирани потребители в системата. Можем да изтриваме даден акаунт и да модифицираме неговата роля - дали да е потребител или администратор. Също така можем да търсим потребители по име или имейл адрес. Това е много удобно, когато имаме твърде много потребители и искаме да се ориентираме бързо.

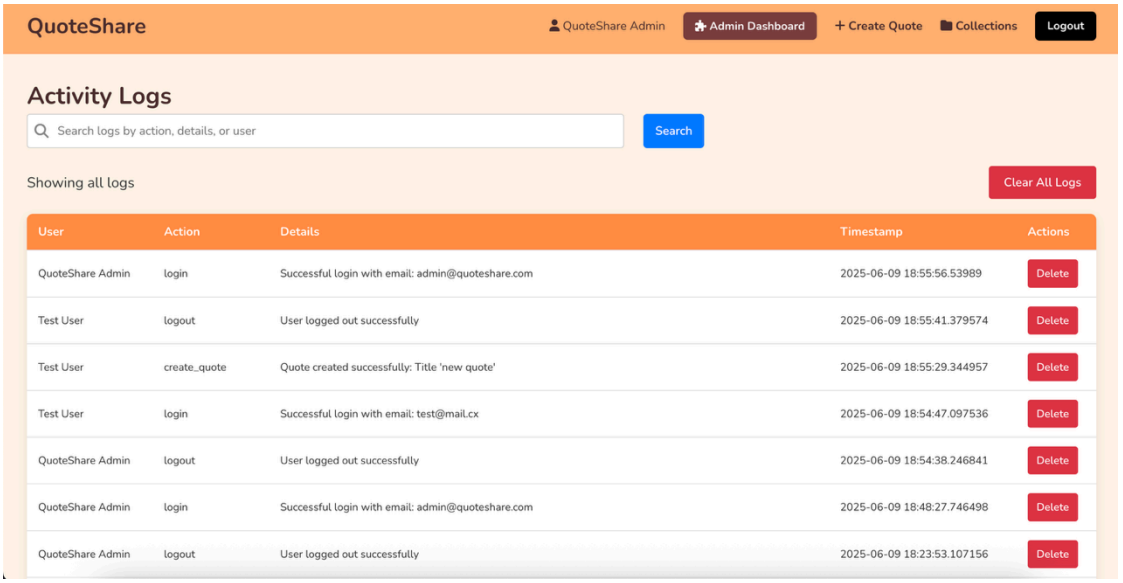


- Когато изтриваме потребител или променяме неговата роля системата отваря прозорец за потвърждение:

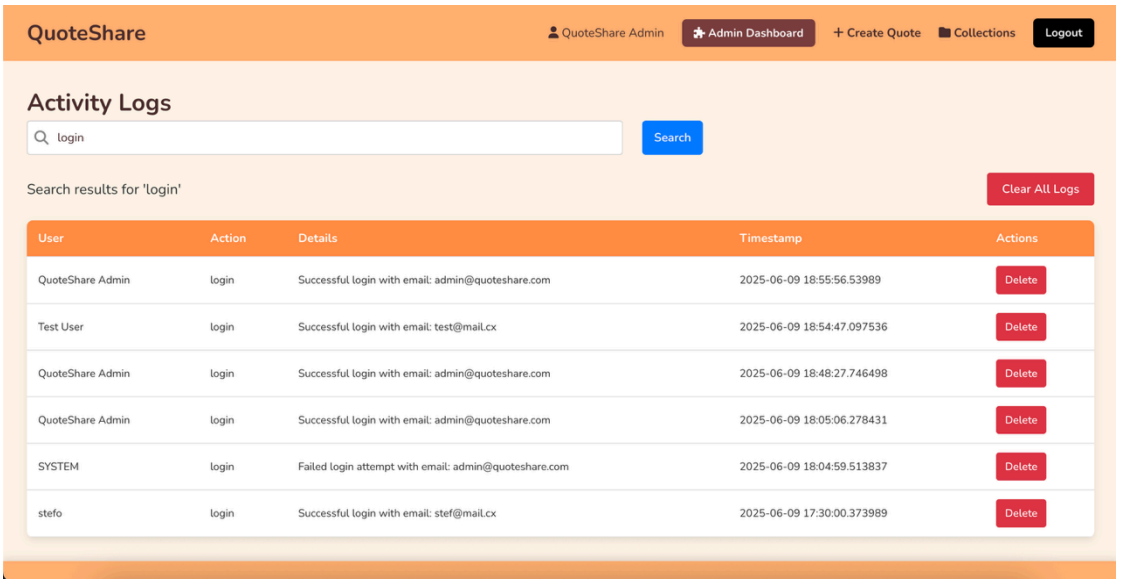

QuoteShare
QuoteShare Admin
Admin Dashboard
+ Create Quote
Collections
Logout
Manage Users
Search users by name or email
Search
Showing all users
Name: Nikola Iliev
Email: nikolcho8381@gmail.com
Role: user
User
Update Role
Delete User
Are you sure you want to delete user ' Nikola Iliev'?
Yes
No
© 2025 QuoteShare
All rights reserved.
Privacy Policy
Terms of Service
Contact Us


QuoteShare
QuoteShare Admin
Admin Dashboard
+ Create Quote
Collections
Logout
Manage Users
Search users by name or email
Search
Showing all users
Name: Nikola Iliev
Email: nikolcho8381@gmail.com
Role: user
User
Update Role
Delete User
Are you sure you want to set ' Nikola Iliev' role to 'user'?
Yes
No
© 2025 QuoteShare
All rights reserved.
Privacy Policy
Terms of Service
Contact Us

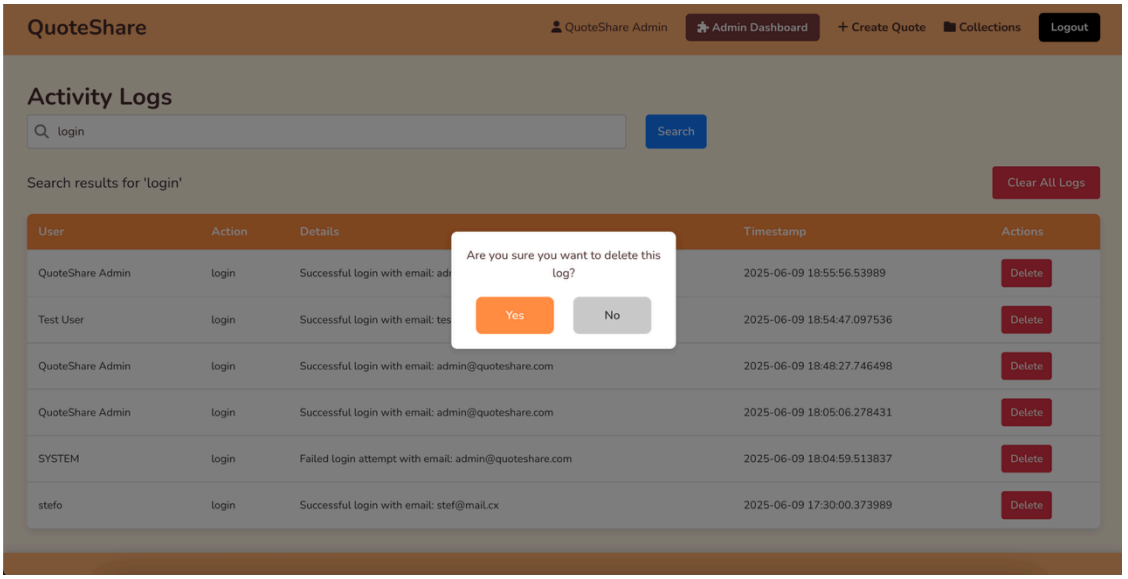
- Админ панел – логове на дейностите
    - В тази страница показваме всички дейности, които са били изпълнени в нашата система. За всяка дейност получаваме информация от кой потребител е била извършена, тип на дейността, описание и време на извършване.



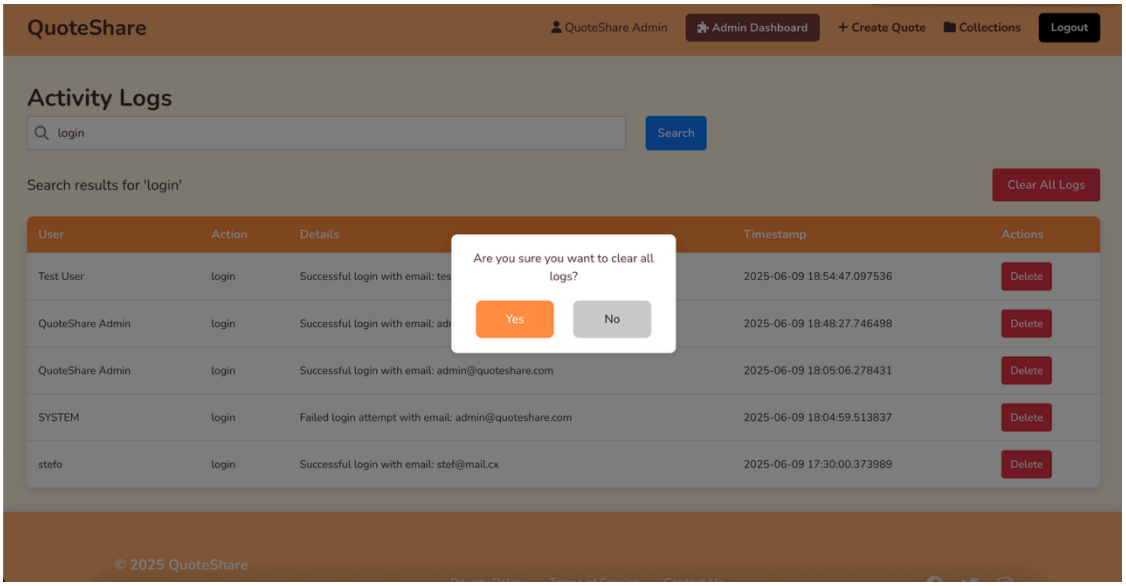
    - Аналогично на потребителите, тук отново можем да търсим(филтрираме), използвайки търсачката над таблицата с логовете.

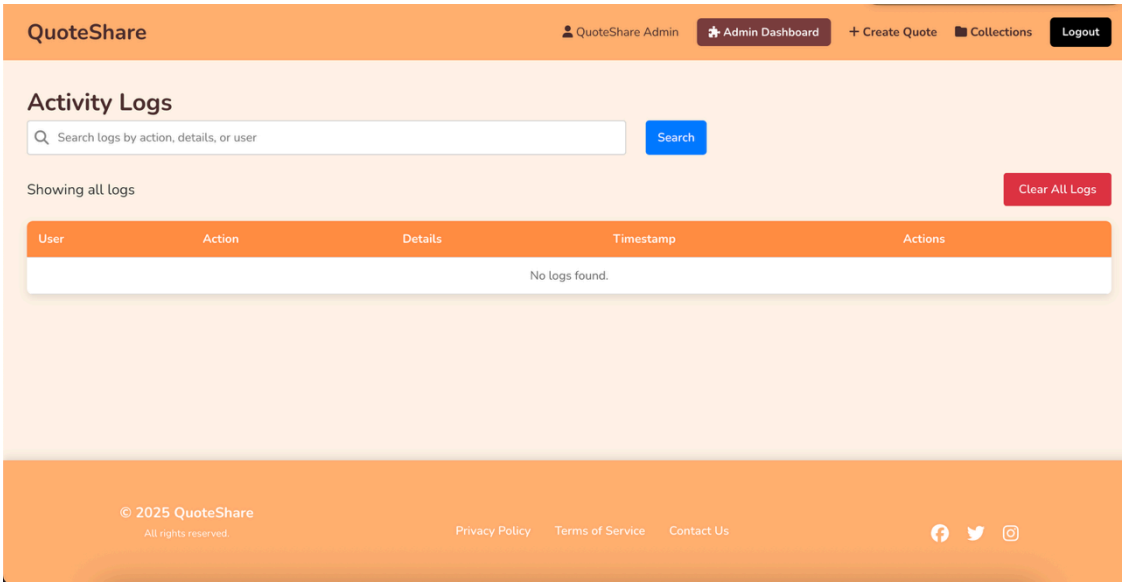


- При опит за изтриване на лог, системата изкарва диалог за потвърждение. При потвърждение, логът бива изтрит.



- Системата предоставя и опция за изчистване на всички логове



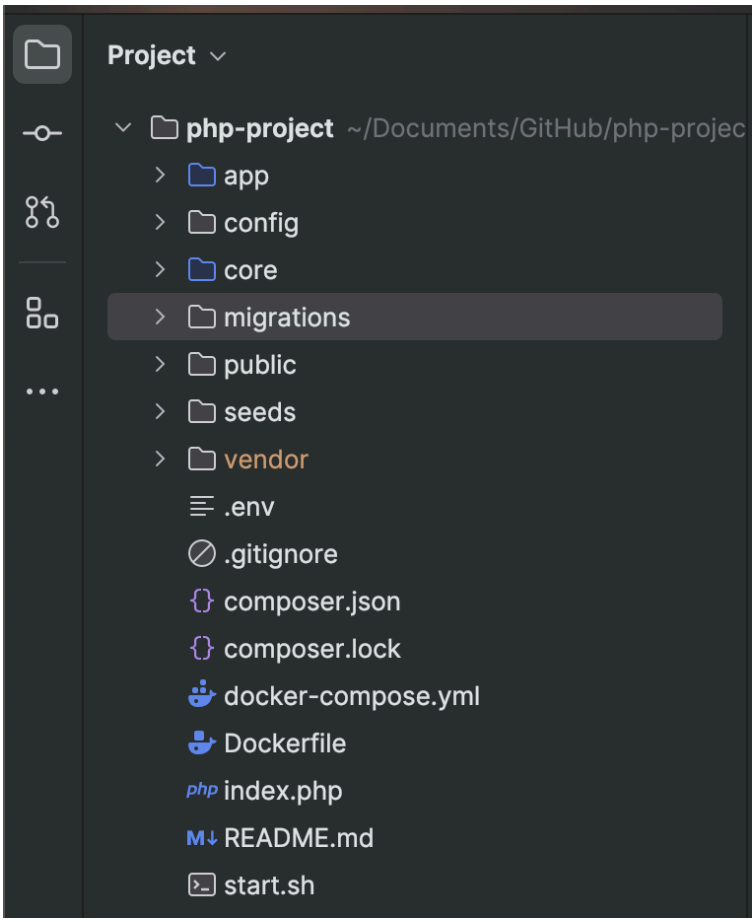


7. Примерни данни

Акаунт за администратор, генериран при стартиране на системата:

- **потребителско име:** admin@quoteshare.com
- **парола:** admin123

8. Описание на програмния код



- **app**: съдържа моделите, контролерите, изгледите на проекта и middleware, който защитава частни ресурси - изисква автентикация/сесия.
    - Controllers: AdminController, HomeController, UserController, CollectionController, QuoteController и т.н. Всеки контролер има в себе си метод, който бива извикан, когато се зареди даден URL адрес и се използва специфичен HTTP method (GET/POST/DELETE/PATCH).
    - Middlewares: AuthMiddleware, AdminMiddleware: Съдържа логиката, която се изпълнява преди зареждането на всяка заявка. Ако дадената заявка/ресурс е защитена с този middleware и ако потребителят не е вписан в системата ще възникне грешка за потребителя и ще бъде пренасочен към страницата за вход. AdminMiddleware проверява дали потребителят изпращащ заявката има администраторски права.
    - Models: UserModel/QuoteModel/CollectionModel/LogModel. В себе си съдържат функции, които общуват с базата дани (четене/създаване/триене и т.н.)
    - Views: Съдържа php файлове, които са самите изгледи, които се зареждат на даден адрес. Всеки контролер може да зареди даден изглед от тази папка. Тъй като футърът и навигацията са общи сме създали папка /partials, където имаме nav.php и footer.php. Те биват импортвани от

различни страници. Така ако променим навигацията или футъра - ще бъде направена промяната и във всички други страници автоматично.

- **config**: съдържа файл, в който лесно конфигурираме уеб приложението какви параметри за използва при създаване на връзка с базите данни
- **core**: главни класове, които са необходими за работата на приложението
    - **App**: Стартовата точка на приложението, която извиква Router и обработва глобални грешки.
    - **Router**: Отговаря за маршрутирането на HTTP заявки към съответните контролери и методи. Поддържа GET/POST маршрути, динамични параметри и middleware-и.
    - **Request**: Абстрахира входната заявка – метод, път, query параметри, body, параметри от URL и достъп до сесията.
    - **Response**: Използва се за изграждане на отговор – текст, JSON, HTTP статус, пренасочване или зареждане на view файл.
    - **Middleware**: Абстрактен клас, който се използва за междинна логика като автентикация, логове и достъп, преди контролерът да бъде изпълнен.
    - **Database**: Управлява връзката с PostgreSQL базата данни и предоставя методи за изпълнение на заявки (SELECT, INSERT и др.).
    - **Session**: Обработва PHP сесиите – set, get, remove, destroy. Използва се за вход, съобщения и други състояния между заявки.
    - **Container: Dependency Injection контейнер**, който съхранява и създава нужните обекти при поискване (напр. контролери, бази и др.).
- **migrations:**
    - db.sql - съдържа всички нужни таблици, които са необходими, за да работи приложението коректно.
    - setup.sh - това е скрипт, който се извиква от start.sh и се използва, за да пусне миграциите в базата данни.
- **public/assets:**
    - съдържа всички нужни css файлове, които се импортват от различни страници
- **public/js:**
    - съдържа всички нужни JavaScript скриптове, които се използват от различни страници, с цел по – добро потребителско преживяване

- **seeds:**
    - съдържа скрипт, който като се пусне регистрира един нормален потребител в системата и един администратор с цел по-лесно тестване.
- **composer.json/composer.lock:**
    - composer.json: Основният конфигурационен файл на Composer. В него се описват зависимостите на проекта (пакети и библиотеки), PSR-4 autoloading правила, име на проекта и версии.
    - composer.lock: Генерира се автоматично от Composer при инсталиране/ъпдейт на зависимости. Заключва точните версии на инсталираните библиотеки, за да осигури еднаква работеща среда при всеки, който клонира проекта и изпълни `composer install`.
- **docker-compose.yml:** Дефинира и управлява многоконтейнерна Docker среда. В нашия случай създава два сървиса – PostgreSQL база данни (fmi_db) и PHP приложение (php_app). Свързани са в обща мрежа, използват споделени обеми и работят заедно. Позволява стартиране на цялата среда с една команда.
- **Dockerfile:** Описва как да се изгради PHP контейнер за приложението. Използва php:8.4-cli образ, добавя нужните разширения (pdo, pdo_pgsql), копира файловете, инсталира Composer и стартира вграден PHP сървър на порт 8000. Това е основата на сървиса php_app.
- **index.php:** Главният входен файл на PHP приложението. Зарежда Composer autoload, създава dependency injection контейнер, регистрира модели, контролери и маршрути, и стартира Router-а. Това е централната точка, от която тръгва цялото приложение.
- **start.sh:** Bash скрипт, който автоматизира стартирането на проекта. Изпълнява `docker compose up --build`, изчаква контейнерите, и по желание стартира миграции чрез `./migrations/setup.sh`. Удобен за бързо пускане на локално разработване.

9. Приноси на студента, ограничения и възможности за бъдещо разширение

- **Измисляне и създаване на MVC архитектурата и структурата на проекта**: Всички заедно
- **Контейнеризиране на базите данни и уеб приложението с Docker**: Никола
- **Регистрация и вход**: Никола
- **Добавяне на нов цитат:** Никола

- **Харесване, Запазване и Докладване на цитати**: Никола
- **Добавяне на анотации към цитати**: Анастасия
- **Добавяне на цитати към колекция**: Анастасия
- **Създаване и анотиране на лични колекции**: Анастасия
- **Експорт на дадена колекция в PDF**: Анастасия
- **Преглед на общи статистики за системата в админ панела**: Стефан
- **Преглед на всички докладвани цитати в админ панела**: Стефан
- **Изтриване на цитати през админ панела**: Стефан

- **Преглед и изтриване на логове през админ панела**: Стефан

10. Какво научих (най-важните неща, които сте научили по време на курса и при разработването на проекта-за всеки студент)

Никола:
По време на разработката на проекта придобих опит в създаването на уеб приложения с помощта на MVC архитектура. Научих как се реализира регистрация и вход чрез PHP сесии, както и как се имплементират действия като харесване, запазване и докладване на съдържание. Усъвършенствах уменията си за работа с контролери, модели и изгледи.

Допълнително научих как да конфигурирам и използвам Docker, като контейнеризирах базата данни и уеб приложението, така че целият проект да може да се стартира с една команда - бързо и лесно. Това ми даде реална представа за добрите DevOps практики и изграждането на преносима и лесна за настройка среда за разработка.

Надградих уменията си по JavaScript, включително работа с fetch, асинхронни заявки, динамично обновяване на съдържание, както и HTML/CSS за изграждане на интуитивен и отзивчив потребителски интерфейс. Проектът ми помогна също така да подобря уменията си за работа в екип.

Анастасия:
Разработването на проекта ми помогна да придобия множество нови знания и практически умения. Научих се да създавам уеб приложения с използване на PHP и архитектурата MVC, която включва работа с контролери, изгледи и модели.

По време на процеса успях значително да надградя вече усвоените знания от курса по HTML, CSS и JavaScript. Проектът беше отлична възможност да обединя и приложа всички придобити умения на практика, създавайки работещо приложение, готово за реална употреба от крайни потребители.

Не на последно място, подобрих и уменията си за работа в екип. Работата по проекта изискваше постоянна комуникация и съгласуване с колегите, което допринесе за по-ефективен и успешен екипен процес.

Стефан:
По време на разработката на този PHP проект научих много практически неща, които допълниха теоретичните ми познания. За първи път използвах PHP в реален проект и ми

беше изключително интересно да разбера как работи неговата структура – от изграждането на контролери и модели до работата с шаблони за изгледи. Също така се запознах с основни концепции около организацията на MVC архитектурата.

Работата по проекта ми помогна да усъвършенствам уменията си за работа в екип чрез използването на Git и GitHub – от създаване на отделни branch-ове и pull request-и до разрешаване на конфликти и водене на ясна комуникация чрез commit съобщения и кодови ревюта. Освен това ми беше полезно да използвам Docker за контейнеризация и изграждане на предвидима разработваща среда, както и да автоматизирам процеси чрез shell скриптове и SQL миграции.

Целият процес ми даде реална представа как се изгражда и поддържа пълноценна уеб система в екипна среда.

11. Използвани източници

[1]: Как се използва Composer

[2]: Как се създава примерно MVC с Composer

[3]: Какво е PDO и защо ни е нужно?

[4]: Как се работи с Postgres в PHP

[5]: Сесии в PHP

[6]: Как се пуска Postgres в Docker

[7]: PHP образ в Doker

Предал (подпис): …….

/1MI0600159, Никола Илиев, СИ, 3/

Предал (подпис): …….

/0MI0600156, Анастасия Маджарова, СИ, 3/

Предал (подпис): …….

/9MI0600175, Стефан Шиваров, СИ, 3/

Приел (подпис): …….

/проф. д-р *Милен Петров*/