МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Економічний факультет
Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

на тему: **«Інформаційна система фіксування обліку матеріалів при сервісному обслуговуванні»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Солонінко Іван Сергійович

Керівник: кандидат психологічних наук, доцент кафедри економіко-математичного моделювання,
Зубенко Ігор Ростиславович

Рецензент: Красюк Богдан Віталійович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри економіко-математичного моделювання та інформаційних технологій _____ (проф., д.е.н. Кривицька О.Р.)
Протокол № ____ від « ___ » _____ 2022 р.

Острог, 2022

Міністерство освіти і науки України
Національний університет «Острозька академія»

Факультет: *економічний*
Кафедра: *економіко-математичного моделювання та інформаційних технологій*
Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ольга КРИВИЦЬКА
«____» _____20__ р.

ЗАВДАННЯ
на кваліфікаційний проєкт студента
Солонінко Івана Сергійовича

1. Тема роботи Інформаційна система фіксування обліку матеріалів при сервісному обслуговуванні
керівник проєкту Зубенко Ігор Ростиславович, кандидат психологічних наук, доцент кафедри економіко-математичного моделювання.

Затверджено наказом ректора НаУОА від 29 жовтня 2021 року №110

2. Термін здачі студентом закінченої роботи/проєкту: _____

3. Вихідні дані до роботи/проєкту: дана робота полягає у розробці системи фіксування обліку матеріалів при сервісному обслуговуванні

4. Перелік завдань, які належить виконати: розробка серверної частини, розробка бази даних, розробка інтерфейсу.

5. Перелік графічного матеріалу: рисунки, таблиці.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Зубенко І. Р., доцент кафедри економіко-математичного моделювання та інформаційних технологій		
2	Зубенко І. Р., доцент кафедри економіко-математичного моделювання та інформаційних технологій		
3	Зубенко І. Р., доцент кафедри економіко-математичного моделювання та інформаційних технологій		

7. Дата видачі завдання: _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1.	Постановка технічного завдання		
2.	Розробка архітектури проекту, погодження функціоналу		
3.	Розробка бази даних		
4.	Розробка макетів інтерфейсу веб застосунку		
5.	Розробка серверної частини проєкту		
6.	Розробка клієнтської частини проєкту		
7.	Реалізація веб застосунку		
8.	Попередній захист кваліфікаційної роботи/проєкту		
9.	Здача кваліфікаційної роботи/проєкту на кафедру		

Студент: _____ Іван СОЛОНІНКО

Керівник кваліфікаційної роботи: _____ Ігор ЗУБЕНКО

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: Інформаційна система фіксування обліку матеріалів при сервісному обслуговуванні

Автор: Солонінко Іван Сергійович

Науковий керівник: Зубенко Ігор Ростиславович, кандидат психологічних наук, доцент кафедри економіко-математичного моделювання.

Захищена «……»…… 20__ року.

Ключові слова: php 8.1.6, laravel 4.2.1, MySQL, MVC, користувач, адміністратор, ролі, залежності,
таблиці, github.

Короткий зміст праці:
Метою кваліфікаційної роботи було створення системи система фіксування обліку матеріалів при сервісному обслуговуванні. Даний проєкт реалізує собою систему фіксування обліку матеріалів при сервісному обслуговуванні. З можливістю додавання обладнання, зміну його розташування, стану, та даних про користувачів та надання їм різних прав доступу до системи.

Користувачами додатку може бути будь-хто, проте адміністратор може банити, видаляти та міняти роль користувача.

ANNOTATION
qualification work
for a bachelor's degree

Keywords: php 8.1.6, laravel 4.2.1, mysql, mvc, user, administrator, roles, dependencies,
table, github.

Summary of work: The purpose of the qualification work was to create a system of recording materials in service. This project implements a system of recording materials in service. With the ability to add equipment, change its location, status, and user data and give them different access rights to the system. Anyone can use the application, but the administrator can ban, delete and change the user role.

ЗМІСТ

ВСТУП

Проблема обліку інструментів та комплектуючих при сервісному обслуговуванні виникла дуже давно, а паперовий документообіг в наш час є неактуальним, так як “не дружить” з маштабуванням та є незручним для постійного моніторингу. З розвитком інформаційних технологій вищезгадані процеси вдосконалювалися проте їх автоматизація є і досі актуальною. Для її реалізації потрібно створити система, яка поєднує у собі великий спектр задач.

Для виконання задачі фіксування обліку матеріалів при сервісному обслуговуванні зазвичай використовується паперовий облік. Тому створення модулю “Inventory” є актуальною проблемою, яка потребує рішення.

Мета кваліфікаційної роботи – реалізація інформаційної система фіксування обліку матеріалів при сервісному обслуговуванні «Inventory».

Для виконання поставленої мети нами були вирішені такі завдання:.

1. Оглянути наявні аналогів.
2. Сформувати технічне завдання для проєкту «Inventory».
3. Опис предметного середовища(Технології які використовуються)
4. Проектування схеми бази даних для проєкту «Inventory» на MySQL.
5. Розробка Back-end проєкту засобами PHP, Laravel, XAMPP(phpMyAdmin).
6. Розробка Front-end проєкту засобами: html, css, js, vue.js, bootstrap template.

Об’єктом дослідження є система «Inventory», що розробляється.

Предметом дослідження є технології: PHP, Laravel, MVC, bootstrap, npm, composer, vue.js.

РОЗДІЛ 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ ТА ФОРМУВАННЯ ЗАДАЧІ ПРОЄКТУ

1.1 Огляд наявних аналогів

На ринку наявна велика кількість продуктів даної тематики, однак частина з них або спрямована на управління великою кількістю складських приміщень або не містить усіх необхідних компонентів, або не має безкоштовної версії.

Excel

Microsoft Excel (повна назва Microsoft Office Excel) — табличний процесор, програма для роботи з електронними таблицями, створена корпорацією Microsoft для Microsoft Windows, Windows NT і Mac OS. Програма входить до складу офісного пакета Microsoft Office.

Розглянемо недоліки програми:

- За відсутності автоматизації, всі дані вносяться вручну, що забирає багато часу.
- Через ручне введення даних зростає вірогідність помилки, яку не просто виявити, в разі застосування зведених таблиць. А це години робочого часу бухгалтера та невдоволеність мешканців, якщо помилку не виправлено негайно.
- Відсутність безпеки даних – основний недолік роботи в Excel. Навіть звичайне вимкнення світла призведе до втрати даних, не говорячи вже про комп'ютерний вірус чи кібератаку.

«1С:Бухгалтерия государственного учреждения 8».

Один з достатньо популярних та відомих рішень що є частиною системи «1С:Бухгалтерия государственного учреждения 8». Безкоштовної версії даного продукту немає, але дана система пропонує широкий набій функціоналу: Створення та друк штрих-кодів для автоматичного обліку активів за допомогою «1С:Печать штрихкодов».Можливості по використанню спеціальних терміналів для проведення інвентаризації. Створення відповідних документів інвентаризації у системі 1С. Автоматичне завантаження інформації з терміналу у систему 1С. Система є зручною та широко використовується.

Основними недоліками є перевантаженість інтерфейсу, відсутність

безкоштовної версії, обмежена підтримка терміналів, відсутність (або обмежена її наявність) бездротової передачі результатів інвентаризації до системи з терміналів.

Парус «Інвентаризаційний облік»

Даний продукт є частиною системи Парус та має використовуватись разом з модулем «Мобільний збір даних». Він разом з 1С є масштабними системами в яких інвентаризація є лише малою частиною. Основними функціями є:

- - Створення та друк етикеток для подальшої інвентаризації.
- - Формування інвентаризаційної комісії.
- - Підтримка Імпорт/експорт даних про об’єкти та ручне введення.
- - Формування вантаження з/на термінали інформації про об’єкти інвентаризації.

На жаль тут також не присутня безкоштовна версія. Однак можливо також вносити інформацію про місцеположення об’єкта. Підтримка терміналів також є обмеженою.

Stock and Inventory Simple

Даний продукт є безкоштовним та розповсюджується за допомогою Google Play Store. Він не є цілісною системою інвентаризації а лише автономним застосунком для пристроїв на базі платформи Android. Однак він може бути використаний для проведення інвентаризації у невеликих масштабах. Зокрема він дозволяє:

- - Імпортувати та експортувати документи у документ Excel.
- - Організовувати каталог об’єктів інвентаризації.
- - Створювати набір звітів включаючи історію об’єктів, прибутки та збитки.

Однак даний застосунок не є цілісною системою тому він неможливо проводити інвентаризацію кількома співробітниками одночасно, пристрій для роботі тільки один, немає централізованого та широкодоступного місця для збереження документів.

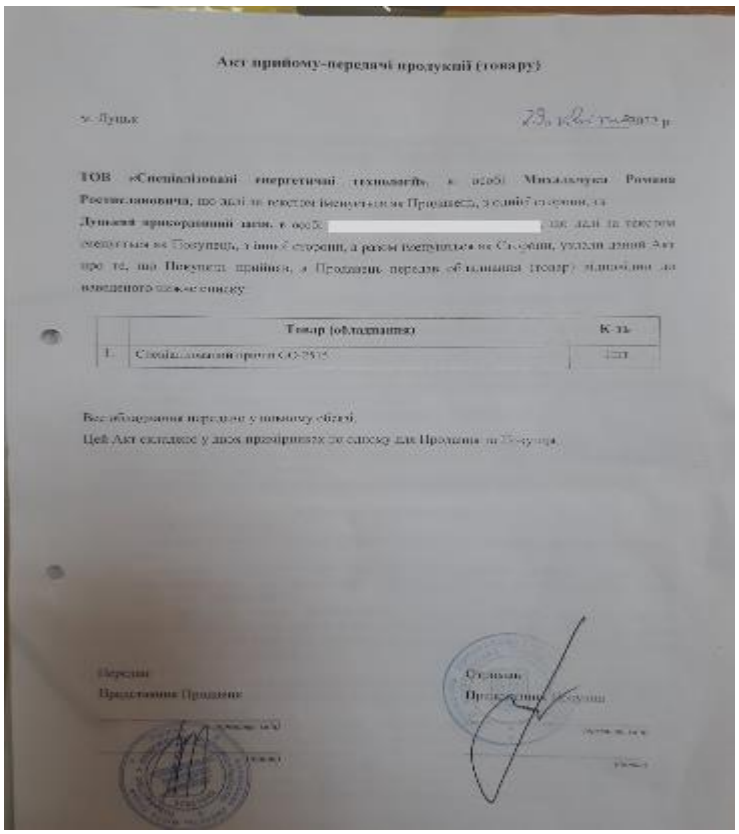
Після виконання даного аналізу та відповідно, отримання інформації про популярні та ефективні системи та методі вирішення задачі інвентаризації,

можна виділити основні переваги та недоліки. Більшість платформ, які представляють собою безпосередньо цілісну та централізовану систему інвентаризації є платними, а ті, що доступні безкоштовно, не мають достатнього набору функціональності для ефективного їх використання. Також, дані системі є обмеженими у виборі терміналів для проведення процесу. Крім того немає будь-яких можливостей по модифікації відображення та роботи терміналів відповідно до метаданих.

1.2 Постановка задачі

Аналіз бізнес-процесу

Для детального детального дослідження проблеми я проаналізував діяльність компанії “Спеціалізовані енергетичні технології”. Вона займається сервісним обслуговування інверторів струму , акумуляторів, сонячних панелей і іншого обладнання. При фіксуванні обліку підприємство використовує акти прийому-передачі матеріалів(Рис.1.1).



Акт прийому-передачі продукції (товару)

м. Луцьк

ТОВ «Спеціалізовані енергетичні технології», в особі Михалчука Романа Ростиславовича, що далі по тексту іменується як Продавець, з одної сторони, та

Дуцький юридичний ліцей, в особі [redacted] що далі по тексту іменується як Покупець, з іншої сторони, а разом іменуються як Сторони, уклали даний Акт про те, що Покупець прийняв, а Продавець передав обладнання (товар) відповідно до вказаного нижче списку:

	Товар (обладнання)	К-ть
1.	Сонячна панель ...	1шт

Все обладнання передано у повному обсязі.

Цей Акт складено у двох примірниках по одному для Продавця та Покупця.

Передав
Представник Продавця

Отримав
Представник Покупця

Рис.1.1. Акт прийому-передачі продукції(товару)

Зауважу, що при веденні обліку цінова характеристика не вказується. Сам процес передбачає 2 особи:

1. Клієнт(людина , що передає обладнання чи комплектуючі)
2. Працівник (менеджер або працівник складу, що його(їх) отримує)

В подальшому на підприємстві також передбачене переміщення обладнання між складами та його передачу працівникам. Після виконання робіт це обладнання передається клієнту, а зі складів списується. Дані про рух обладнання по складах та його передача від одного працівника до іншого теж фіксується в паперовому форматі або за допомогою Exel таблиць.

Технічне завдання

Проаналізувавши процес обліку на прикладі існуючого підприємства я зробив висновок, що для повноцінного функціонування системи достатньо кількох основних ролей:

- Адміністратор .
- Неавторизований користувач.
- Авторизований користувач .

Неавторизований користувач повинен лише мати можливість авторизуватися використовуючи відомий йому логін, або зареєструватися.

Авторизований користувач повинен мати право:

- Додавати інвентар.
- Бачити лише той інвентар, який додав сам.
- Має змогу оновлювати інформацію про обладнання, який додав.
- Мати змогу змінити свої облікові дані(password, login).

В даному випадку авторизований передбачає наявність доступу до таблиці інвентаризації та наявність у даного користувача його логіну.

Адміністратор відповідно повинен мати доступ до веб-застосунку та мати можливості:

- Заводити нових користувачів системи.
- Змінювати та переглядати інформацію про наявних користувачів.
- Переглядати список наявного обладнання.

- Банити користувачів.
- Змінювати статус, опис та місцезнаходження обладнання.
- Проводити пошук інвентарних об’єктів у системі та переглядати їх статус.
- Бачити секцію “Склади”
- Створювати нові склади.
- Завантажувати дані про наявне обладнання.
- Передавати обладнання
- Списувати (видаляти) обладнання по виконанню робіт.
- Бачити створення, передачу, переміщення та видалення обладнання.

1.3 Опис програмного середовища

Для реалізації проєкту на основі сформульованого технічного завдання було обрано стек технологій : MySQL, PHP 8, phpMyAdmin, Laravel, MVC, composer, npm(Node Package Manager) Vue.js, bootstrap template, node.js.

MySQL

MySQL — вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL має подвійне ліцензування. Вона може розповсюджуватися відповідно до умов ліцензії GPL. Але за умовами GPL, якщо якась програма використовує бібліотеки MySQL, то вона теж повинна розповсюджуватись за ліцензією GPL. В разі використання та розповсюдження програмного забезпечення з іншими вільними ліцензіями, такими як BSD, Apache, MIT та інші, MySQL дозволяє використання бібліотек MySQL за ліцензією GPL.

PHP

PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб розробок (разом із Java, .NET, JavaScript, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проєкт відкритого програмного забезпечення.

PHP інтерпретується вебсервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, тому що браузер отримує готовий html-код. Це є перевагою з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

Особливості:

PHP — мова, у код якої можна вбудовувати безпосередньо html-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не зустріне закриваючий тег.

Велика різноманітність функцій PHP дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в C або Pascal.

Наявність інтерфейсів до багатьох баз даних

- У PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, SQLite, mSQL, O racle, dbm, Hyperware, Informix, InterBase, Sybase.
- завдяки *стандарту відкритого інтерфейсу зв'язку з базами*) можна підключатися до всіх баз даних, до яких існує драйвер.

Нетрадиційність

Мова PHP здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з C, Perl. Код PHP дуже схожий на

той, який зустрічається в типових програмах мовами C або Pascal. Це помітно знижує початкові зусилля при вивченні PHP. PHP — мова, що поєднує переваги Perl та C і спеціально спрямована на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. І хоча PHP є досить молодою мовою, вона здобула таку популярність серед web-програмістів, що в наш час є найпопулярнішою мовою для створення веб застосунків (скриптів).

Наявність сирцевого коду та безкоштовність

Стратегія Open Source, і розповсюдження початкових текстів програм в масах, безсумнівно справили сприятливий вплив на багато проєктів, в першу чергу — Linux хоч і успіх проєкту Apache сильно підкріпив позиції прихильників Open Source. Сказане відноситься і до історії створення PHP, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником в розвитку проєкту PHP. Ухвалення стратегії Open Source і безкоштовне розповсюдження початкових текстів PHP надало неоціненну послугу користувачам. Окрім цього, користувачі PHP в усьому світі є свого роду колективною службою підтримки, і в популярних електронних конференціях можна знайти відповіді, навіть на найскладніші питання.

Ефективність

Ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і web. Важливою перевагою PHP є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на Perl. Проте хоч би що робили розробники PHP, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність PHP достатня для створення цілком серйозних веб застосунків.

phpMyAdmin

phpMyAdmin — веб-додаток з відкритим кодом, написаний мовою PHP із графічним веб інтерфейсом для адміністрування бази даних MySQL або

MariaDB. phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних.

На сьогоднішній день phpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, з огляду на всі нововведення СКБД MySQL. Переважна більшість українських провайдерів використовують цей застосунок як панель керування для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних.

Програма розповсюджується під ліцензією GNU General Public License і тому деякі інші розробники інтегрують його у свої розробки, наприклад XAMPP, Denwer.

Laravel

Laravel — безкоштовний, з відкритим кодом PHP-фреймворк, створений Тейлор Отвел (англ. Taylor Otwell) і призначений для розробки вебдодатків відповідно до шаблону model–view–controller (MVC).

Серед особливостей Laravel можна назвати: модульну систему упакування з виділеним менеджером залежностей Composer, різні способи для доступу до реляційних баз даних, утиліти, які допомагають в розгортанні додатків і технічного обслуговування, а також його орієнтація на синтаксичний цукор.

Станом на березень 2015 року, Laravel вважався одним з найпопулярніших PHP фреймворком, разом з Symfony5, Nette, CodeIgniter, Yii2 й іншими.

Сирцевий код Laravel'a розміщується на GitHub і ліцензований відповідно до умов MIT License.

Vue.js

Vue.js— JavaScript - фреймворк з відкритим вихідним кодом для створення інтерфейсів користувача. Легко інтегрується у проекти з використанням інших JavaScript-бібліотек. Може функціонувати як веб-фреймворк для розробки односторінкових програм у реактивному стилі.

На даний момент підтримується творцем Еваном Ю. та іншими активними членами основної команди з різних компаній, таких як Netlify, Netguru, Baidu, Livestorm.

Висновки до розділу 1

Облік матеріалів та обладнання – це складний однак та дуже важливий процес для сучасного бізнесу. Її ефективне виконання та використання отриманої інформації значно впливають на можливість функціонування підприємства в цілому. Інформатизація цього процесу дозволяє значно його пришвидшити та мінімізувати помилки.

На ринку вже присутній ряд систем що слугують даній цілі і відповідно всі вони мають набір переваг та недоліків коротко описаний у цьому розділі. Однак жодна з представлених систем, які містять достатню функціональність недоступна безкоштовно, а ті що доступні є сильно обмеженими.

Було досліджено роботу дійсного підприємства, яке займається сервісним обслуговуванням, та на основі його робочих процесів сформовано технічне завдання проєкту.

Після детального аналізу технічного завдання було обрано стек технологій, за допомогою яких реалізовуватиметься система фіксування обліку матеріалів при сервісному обслуговуванні.

РОЗДІЛ 2. СТВОРЕННЯ АРХІТЕКТУРИ ПРОЄКТУ

2.1 Аналіз використовуваних даних

Посилаючись на дослідження, що були проведені для постановки задачі, дані, на основі яких функціонує бізнес-процес, можна розділити на дві групи - зовнішні та внутрішні.

Зовнішні

Це ті дані, які отримуємо при надходженні обладнання, а саме:

- ПІБ клієнта або назва підприємства, що є клієнтом
- Номер телефону
- Найменування обладнання

- Кількість обладнання
- Стан обладнання
- Дату надходження

Внутрішні

Дані, що пов’язані безпосередньо з підприємством і що виникають в процесі обслуговування, до моменту видачі, а саме:

- Хто прийняв обладнання
- Де воно знаходиться(склади та адреси)
- Кому передавалося (якщо на те виникала потреба)
- Опис обладнання
- Дати будь-яких операцій над ним

Зважаючи на цей поділ даних було

2.2 Проектування системи

Для того, щоб на основі цих даних і проаналізованих вище процесів створити якісне програмне забезпечення, яке було б ресурсо-ефективним та швидким у використанні, необхідно спочатку змоделювати ці процеси у системі та відповідно до них спроектувати архітектуру. Для зберігання великого обсягу інформації, що стосується певної області дуже зручно користуватися системами управління базами даних (СУБД). СУБД дозволяє:

- надійно зберігати інформацію;
- змінювати (додавати, видаляти, оновлювати) інформацію;
- зменшити час доступу до необхідної інформації;

Таблиці

Зважаючи на зовнішні та внутрішні дані для їх бази було вирішено створити такі таблиці:

1. Користувачі
2. Ролі
3. Інвентар
4. Склади
5. Історія операцій

Для кожної з них створено відповідні поля. Розглянемо їх детальніше.

Сутність “Користувачі” реалізована для ідентифікації користувачів. Вона має такі поля:

- Id - для того, щоб унікально визначити кожного користувача та подальшої роботи під час створення веб додатку.
- Ім’я та прізвище - персональні дані для ідентифікації особи.
- Електронна скринька та пароль - для авторизації в системі.
- Роль - для розділення прав доступу в системі.
- Токен - для реалізації початку та завершення сесії користувача.
- Дата створення та дата оновлення даних користувачем.

Вигляд таблиці в СУБД можна побачити на рисунку 2.1.

id	first_name	last_name	email	mobile_number	email_verified_at	password	role_id 1=Admin, 2=ТА/ТР	status	remember_token	created_at	updated_at
1	Super	Admin	admin@admin.com	9028187696	NULL	$2y$10$LA4wD0SPbSqhxiEt6PCwsurFpwOFqUIiP3klcZdoMRD...	1	1	NULL	2022-06-16 15:45:06	2022-06-16 15:45:06
2	ivan	Soloninko	ivan.soloninko@oa.edu.ua	0501668662	NULL	$2y$10$RvHyFkwbVVw6k8TfQB1QwO48o1LP1hf5dstKAiI.5YM...	2	1	NULL	2022-06-16 15:58:07	2022-06-16 15:58:07

Рис.2.1. Таблиця users

Ролі в додатку реалізовані для розділення прав доступу, так як цього вимагає технічне завдання. Без її реалізації кожен користувач мав би повний доступ до ресурсу, що не є безпечним та не відповідає вимогам поставленої задачі. Таблиця має такі поля:

- Id - для надання унікальності ролі.
- Назву - для опису ролі та її відображення користувачам.
- Дата створення та дата оновлення.

+ Параметри

	id	name	created_at	updated_at
Редагувати Копіювати Видалити	1	Admin	2022-06-16 15:45:06	2022-06-16 15:45:06
Редагувати Копіювати Видалити	2	User	2022-06-16 15:45:06	2022-06-16 15:45:06

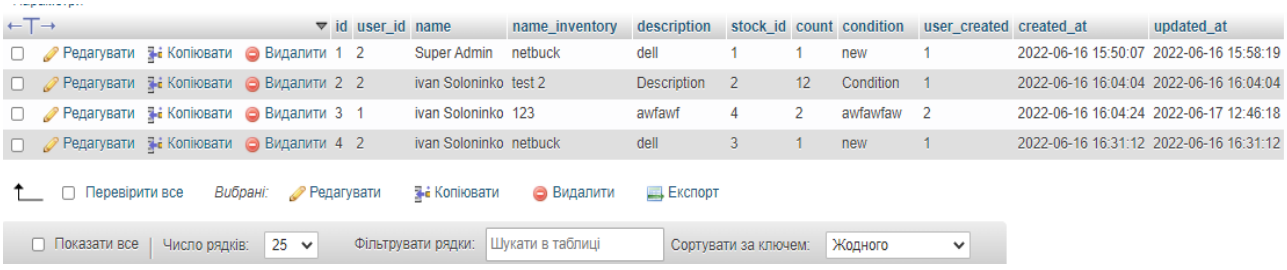
Перевірити все Вибрані: Редагувати Копіювати Видалити Ек

Рис.2.2. Таблиця roles

Таблиця була створена для реалізації сутності інвентарю та подальших дій над ним. Вона є основною в проекті і, в зв'язку з цим, пов'язана майже з усіма іншими таблицями. Її поля виглядають так:

- Id
- Id користувача, за яким закріплено та його ім'я
- Назву обладнання
- Опис
- Id складу
- Стан
- Опис
- Кількість
- Користувач, що створив
- Дату створення та оновлення

Поля id реалізовані для звернення в коді до конкретного обладнання, створення зв'язків між таблицями та реалізації подальших дій над ним. Назва, опис, стан і кількість - для загального розуміння, що саме було прийнято на сервіс.



	id	user_id	name	name_inventory	description	stock_id	count	condition	user_created	created_at	updated_at
Редагувати Копіювати Видалити	1	2	Super Admin	netbuck	dell	1	1	new	1	2022-06-16 15:50:07	2022-06-16 15:58:19
Редагувати Копіювати Видалити	2	2	ivan Soloninko	test 2	Description	2	12	Condition	1	2022-06-16 16:04:04	2022-06-16 16:04:04
Редагувати Копіювати Видалити	3	1	ivan Soloninko	123	awfawf	4	2	awfawfaw	2	2022-06-16 16:04:24	2022-06-17 12:46:18
Редагувати Копіювати Видалити	4	2	ivan Soloninko	netbuck	dell	3	1	new	1	2022-06-16 16:31:12	2022-06-16 16:31:12

Перевірити все Вибрані: Редагувати Копіювати Видалити Експорт

Показати все | Число рядків: 25 Фільтрувати рядки: Шукати в таблиці Сортувати за ключем: Жодного

Рис.2.3 Таблиця Inventories

Таблиця "Склади" була створена для постійного моніторингу місцезнаходження обладнання, так як згідно з технічним завдання обладнання може переміщатися під час сервісного обслуговування. Вона складається з таких полі:

- Id
- Назва
- Адрес
- Дата створення та дата оновлення

В базі даних виглядає так:

id	name	address	created_at	updated_at
1	Stock 1	Address 1	2022-06-16 15:45:10	2022-06-16 15:45:10
2	Stock 2	Address 2	2022-06-16 15:45:10	2022-06-16 15:45:10
3	Stock 3	Address 3	2022-06-16 15:45:10	2022-06-16 15:45:10
4	Stock 4	Address 4	2022-06-16 15:45:10	2022-06-16 15:45:10

Рис.2.4 Таблиця stocks

Так як процес сервісного обслуговування обладнання передбачає не лише отримання та видачу, а й перенесення, зміну характеристик, передачу іншим працівникам, то для ведення обліку доцільним стало створення таблиці “Історія операцій”. Вона складається з таких полів:

- Id
- Текст
- Дата створення та дата оновлення

id	text	created_at	updated_at
1	Super Admin add new inventory - netbuck	2022-06-16 15:50:08	2022-06-16 15:50:08
2	User Super Admin trade handed over inventory Super...	2022-06-16 15:50:17	2022-06-16 15:50:17
3	User Super Admin trade handed over inventory Super...	2022-06-16 15:58:19	2022-06-16 15:58:19
4	ivan Soloninko add new inventory - test 2	2022-06-16 16:04:05	2022-06-16 16:04:05
5	ivan Soloninko add new inventory - 123	2022-06-16 16:04:24	2022-06-16 16:04:24
6	ivan Soloninko add new inventory - netbuck	2022-06-16 16:31:13	2022-06-16 16:31:13
7	User ivan Soloninko trade handed over inventory iv...	2022-06-17 12:46:19	2022-06-17 12:46:19

Рис.2.6 Таблиця histories

Основним полем в таблиці є поле “текст”. Саме в ньому зберігаються дані про виконані дії над інвентарем.

Схема даних

Самі по собі таблиці не формують повноцінну базу даних. Для того, щоб додаток функціонував належним чином було розроблено схему бази даних. Вона створена на основі вимог, описаних під час постановки задачі та при формуванні технічного завдання.

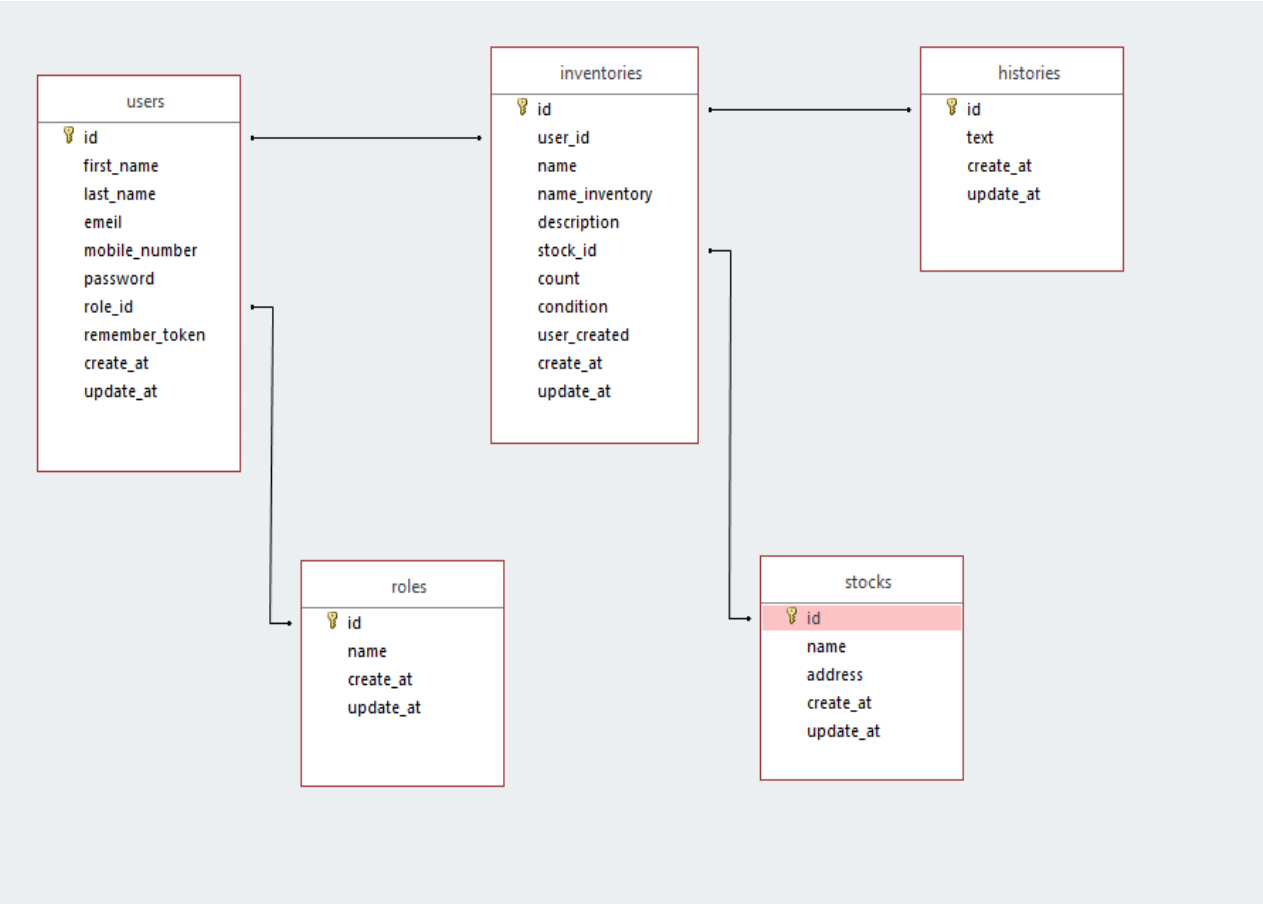


Рис.2.7. Схема даних

Для поглибленого розуміння опишемо зв'язки між таблицями. Оскільки в системі може бути багато користувачів і кількість користувачів як з правами адміністратора так і з правами клієнта не є обмеженою, то між таблицями roles і "users" вид зв'язку " один до багатьох". Цей вид зв'язку застосовується під час усіх зв'язувань, описаних на схеме. Зумовлено це тим, що:

- Один користувач може додавати необмежену кількість товарів.(users - inventories)
- В одному складі одночасно може знаходитися багато обладнання.(stocks - inventories)
- Над одним обладнанням може проводитися багато операцій, які записуються.(inventories - histories)

Після того як створення бази даних завершено перейдемо до архітектури проекту.

2.3 Архітектура проекту

Веб застосунок складається з двох частин: серверної та клієнтської.

Серверна частина

Оскільки для розробки ми обрали мову програмування PHP та фреймворк Laravel, то основою створення бекенд частини є паттерн проектування MVC (Model-View- Controller). Концепція MVC поділяє дані, представлення та обробку дій користувача на компоненти:

Модель / Model — є об'єктним представленням певної предметної області, включає дані та методи роботи з цими даними, реагує на запити з контролера, повертаючи дані та/або змінюючи свій стан. При цьому модель не містить у собі інформації про способи візуалізації даних і формати їх подання, а також не взаємодіє з користувачем безпосередньо.

Представлення/View – відповідає за відображення інформації (візуалізацію). Одні й ті ж дані можуть представлятися різними способами й у різних форматах. Наприклад, колекцію об'єктів за допомогою різних уявлень можна представити на рівні інтерфейсу користувача як в табличному вигляді, так і списком; на рівні API можна експортувати дані як у JSON, так і в XML або XSLX.

Контроллер / Controller – забезпечує зв'язок між користувачем та системою, використовує модель та уявлення для реалізації необхідної реакції на дії користувача. Як правило, на рівні контролера здійснюється фільтрація отриманих даних та авторизація - перевіряються права користувача на виконання дій або отримання інформації.

Загальну схему роботи патерну ми можемо побачити на рисунку 2.8.

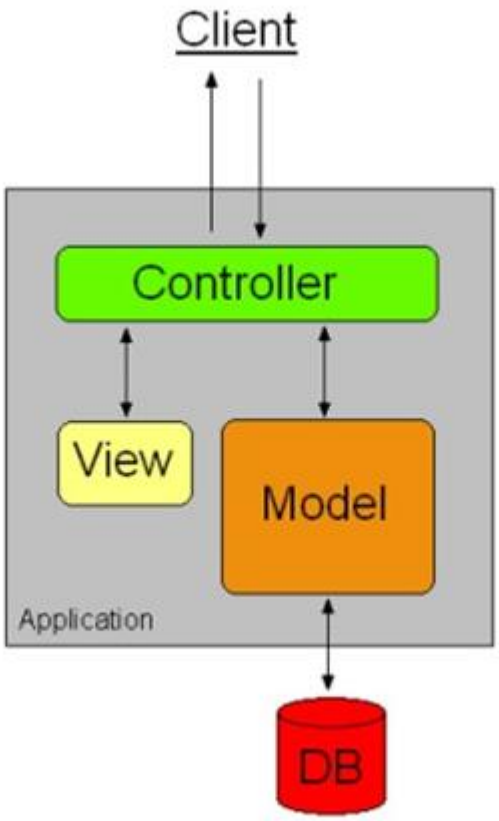


Рис.2.8. Взаємозв'язок компонентів в MVC

Дослідивши фреймворк Laravel я розширив цю схему (Рис.2.9.)

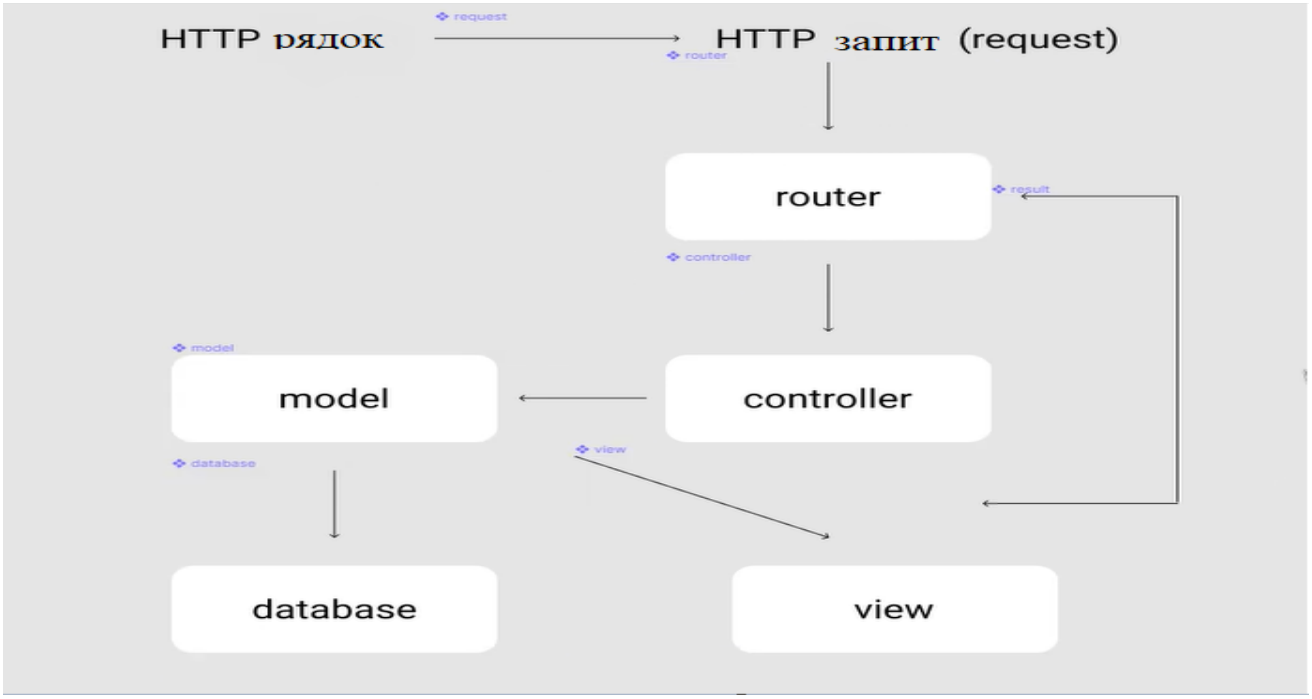


Рис.2.9. Взаємозв'язок компонентів в Laravel проєкті

Опишу детальніше:

1. Будь які взаємодії клієнта і застосунку відбуваються за допомогою HTTP рядка.

2. Для того, щоб відбулися якісь зміни на сторінці клієнту необхідно виконати HTTP запит. Це може бути натискання на кнопку, перехід за посиланням та навіть редагування HTTP рядка за допомогою клавіатури.
3. Після того, як було сформовано запит, його обробляє клас router(маршрутизатор). Саме в ньому створюються URL адреси запити. Як ми бачимо на рисунку 2.9, вони “спілкуються” не лише з контроллерами, а й можуть напряму звертатися до представлення, в залежності від того, як розробник реалізовує їх в проекті. Зазвичай не прийнято цього робити, оскільки такі запити не обробляються контроллерами і є загальнодоступними. На рисунку 2.10 можна побачити де знаходиться маршрутизація в нашому проекті.

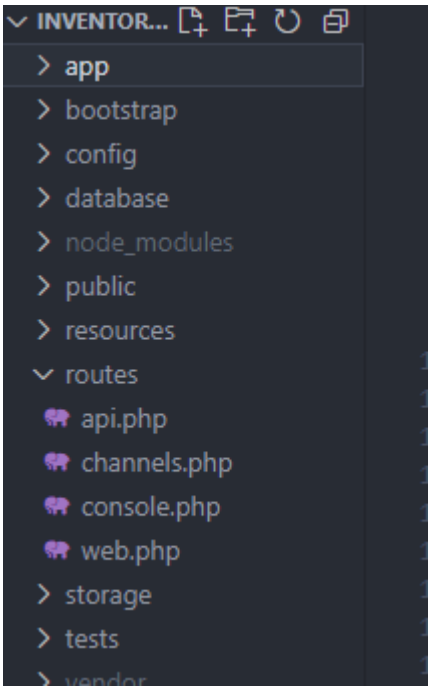


Рис.2.10 routes

4. Якщо запит “напряму” не викликає сторінку, то його обробляє відповідний контроллер.

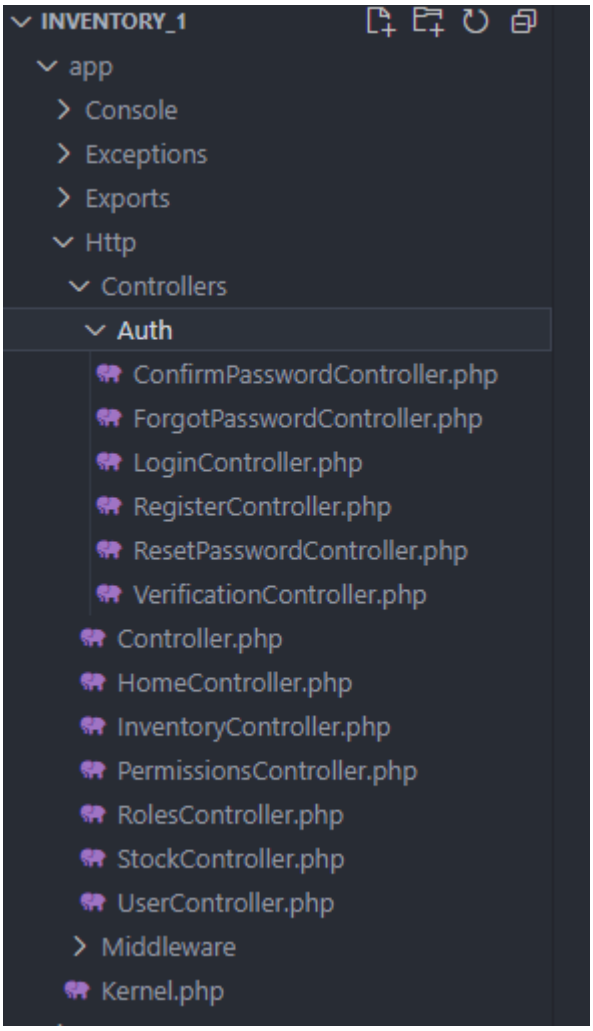


Рис.2.11 Контроллери проєкту

5. Для того, щоб контроллер повернув результати запиту йому потрібно доступитися до даних, обробити їх та передати у представлення. Сам контроллер не може звернутися до бази даних - з цим йому “допомагає” модель.
6. Модель - це представлення одного об’єкту для інтерфейсу php. Вона звертається до бази даних таредає результати контроллеру.

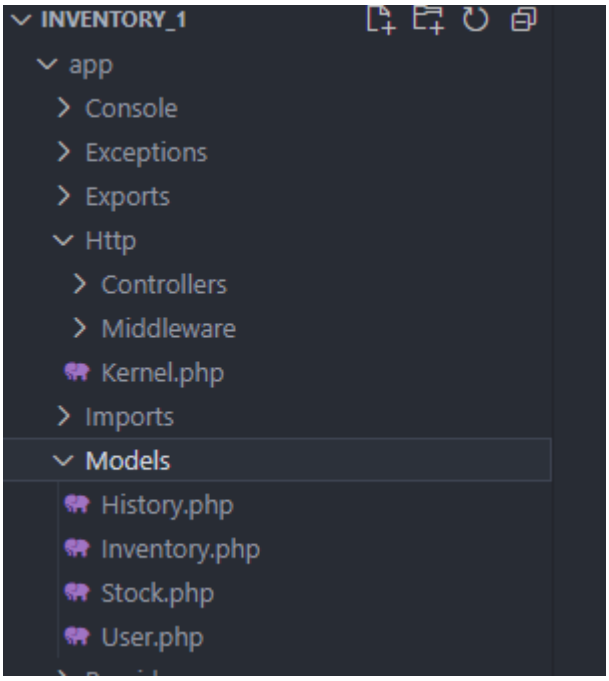


Рис.2.12 Моделі проєкту

7. Контроллер, після отримання відповіді з моделі, яка “витягнула” потрібну інформацію з бази даних, опрацьовує дані та передає їх в представлення.
8. Представлення слугує для візуалізації даних. Воно отримує результат роботи контроллера та надає йому того вигляду, який бачить клієнт після виконання запиту.

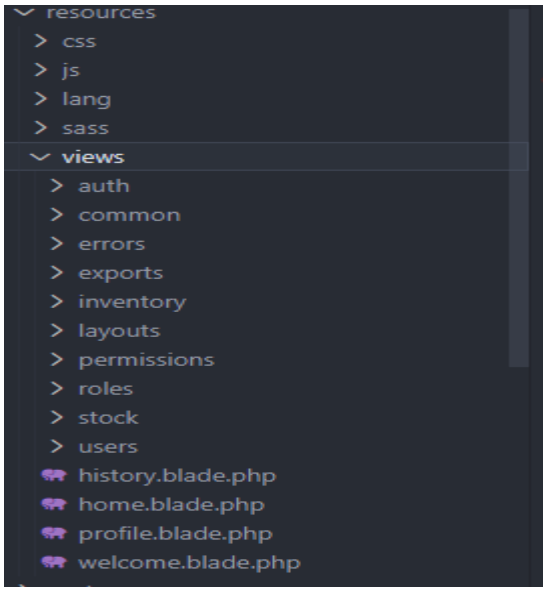


Рис. 2.13 Представлення проєкту

Саме в представленнях реалізований фронтенд проєкту.

Клієнтська частина

Перед тим, як почати реалізовувати клієнтську частину застосунку я дослідив загальнодоступні фреймворки, та обрав для роботи bootstrap. Він надає набір вже готових інструментів для розробки фронтенду. За основу мною було взяті шаблони з github.

На початковій сторінці сайту ми бачимо форму форму для авторизації, яка складається з полей для вводу даних та кнопок.

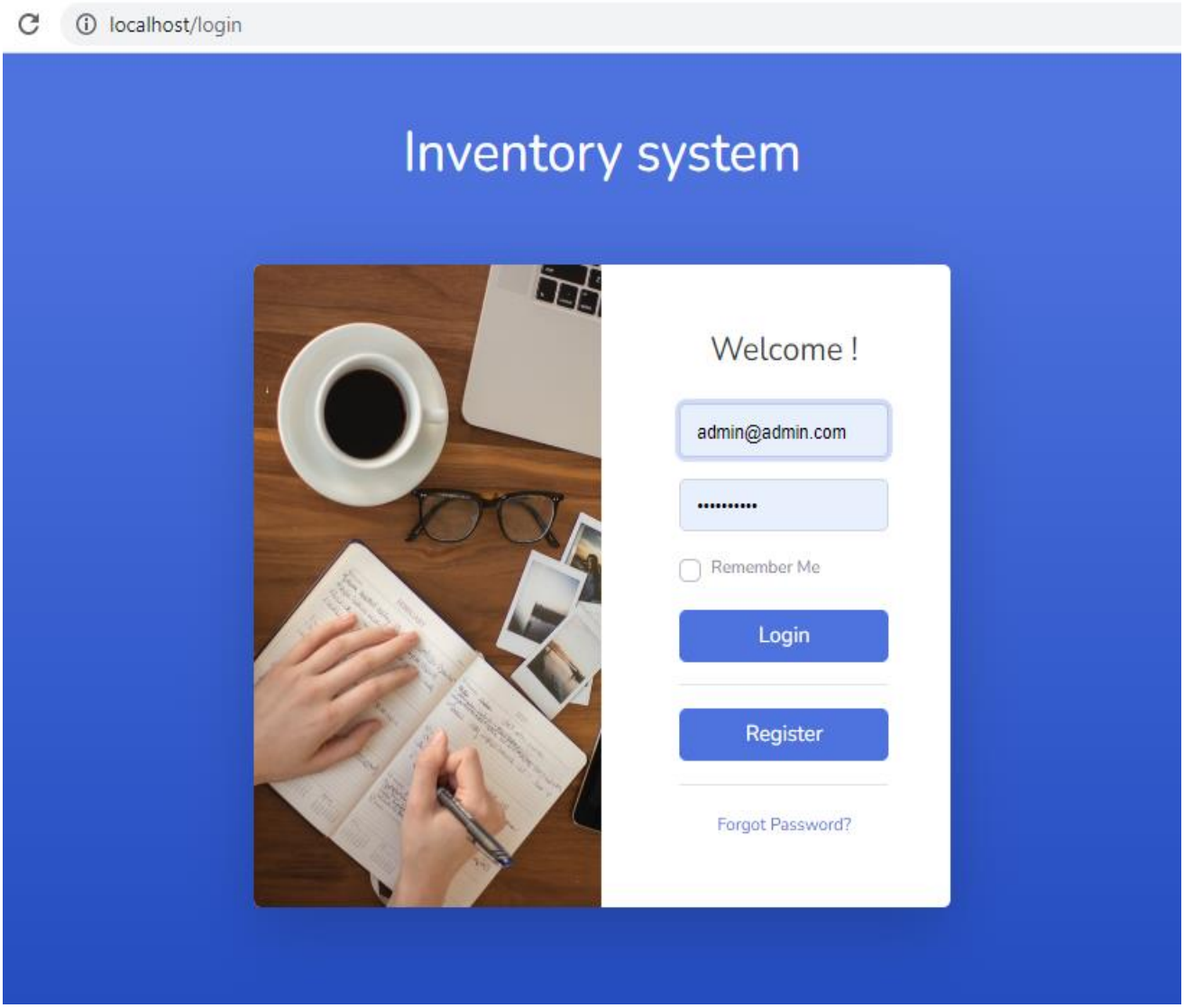


Рис.2.14. Вигляд сторінки авторизації

Після натискання на кнопку відбувається відповідна дія. Незареєстрований користувач перейде на сторінку реєстрації. Її вигляд відрізняється від сторінки

авторизації лише полями, тому перейдемо до сторінки авторизованого користувача-адміністратора.

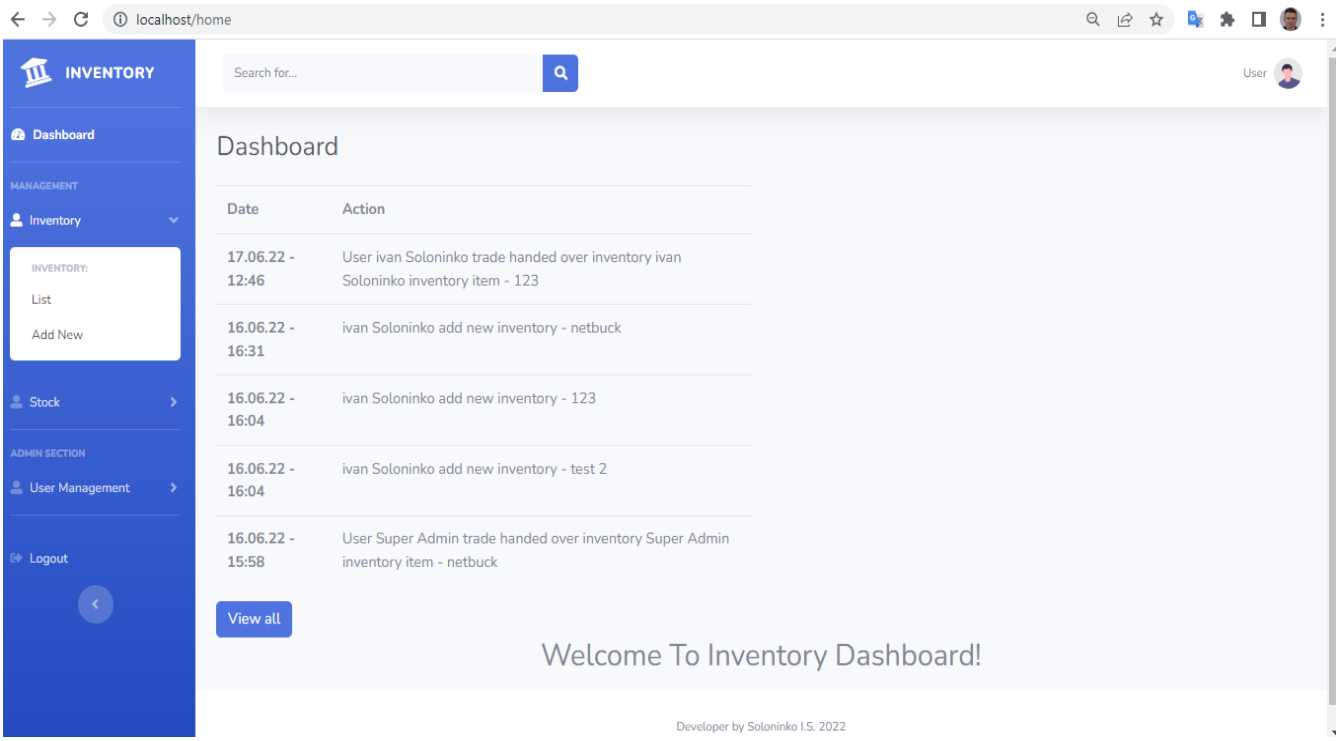


Рис.2.15 Вигляд сторінки home для адміністратора

Для звичайного користувача сторінка відрізняється лише тим, що відсутні секції “Stock” і “User Management” тому доцільніше розглядати зовнішній вигляд додатку на прикладі користувача-адміністратора.

У верхньому правому кутку бачимо логотип користувача користувача. натиснувши на нього спрацьовує подія і ми можемо перейти на сторінку профілю або завершити сесію. Ці кнопки відрізняються стилями від інших в додатку.

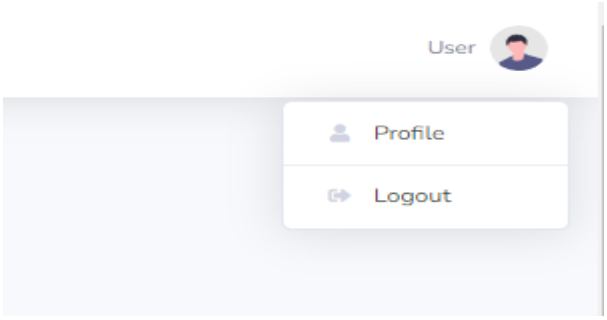


Рис.2.16 Кнопки “Profile” та “Logout”дзвініця

Оглянемо сторінку профілю.

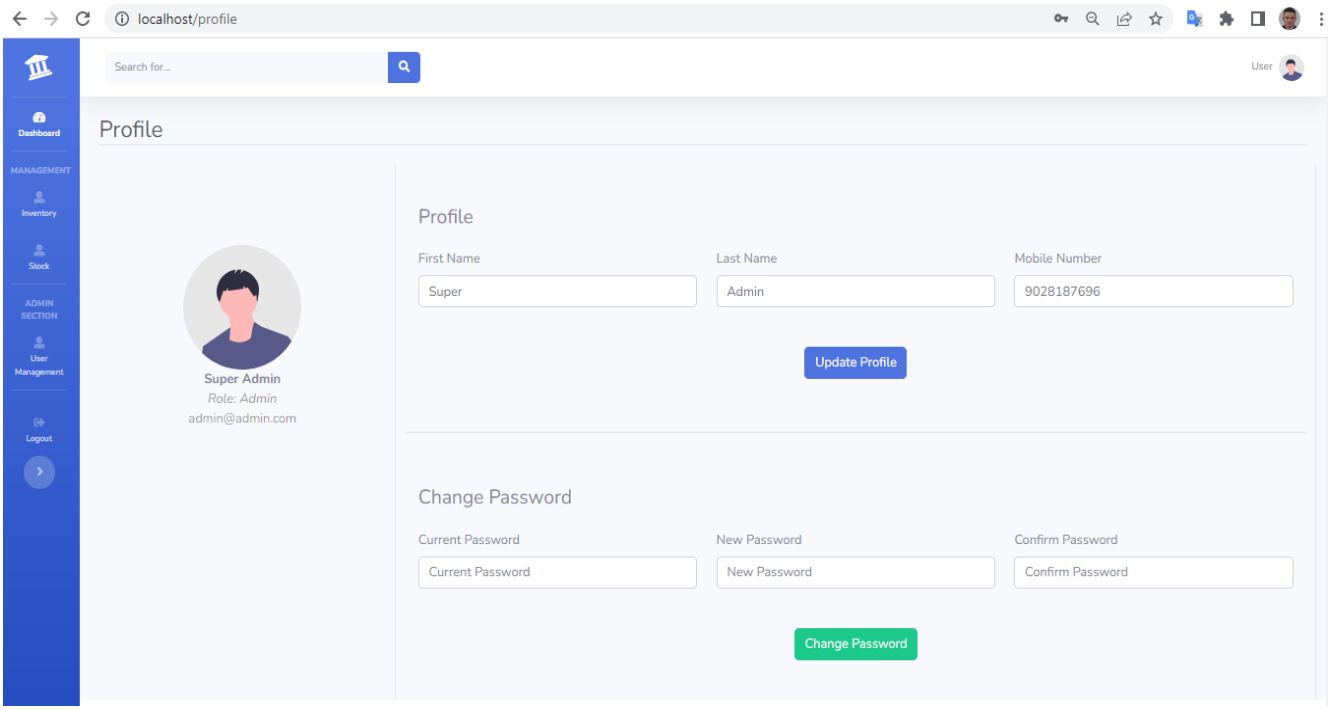


Рис.2.17. Вигляд сторінки profile

З лівої сторони розміщена панель навігації по сайту, проте - в складеному вигляді. Це зроблено для зручності і сторінка виглядає більш лаконічно. Стилізація форм для введення інформації та кнопок така ж, як при реєстрації та в інших таблицях. Для того, щоб додати обладнання, оглянути склади чи користувачів на боковій панелі були створені відповідні секції. Оскільки їх стилізація не відрізняється, то розглянемо на прикладі секції “Inventory”. Перейшовши на сторінку “inventory” ми бачимо таблицю, що складається з відповідних полів таблиці “inventories” в базі даних та кнопок.

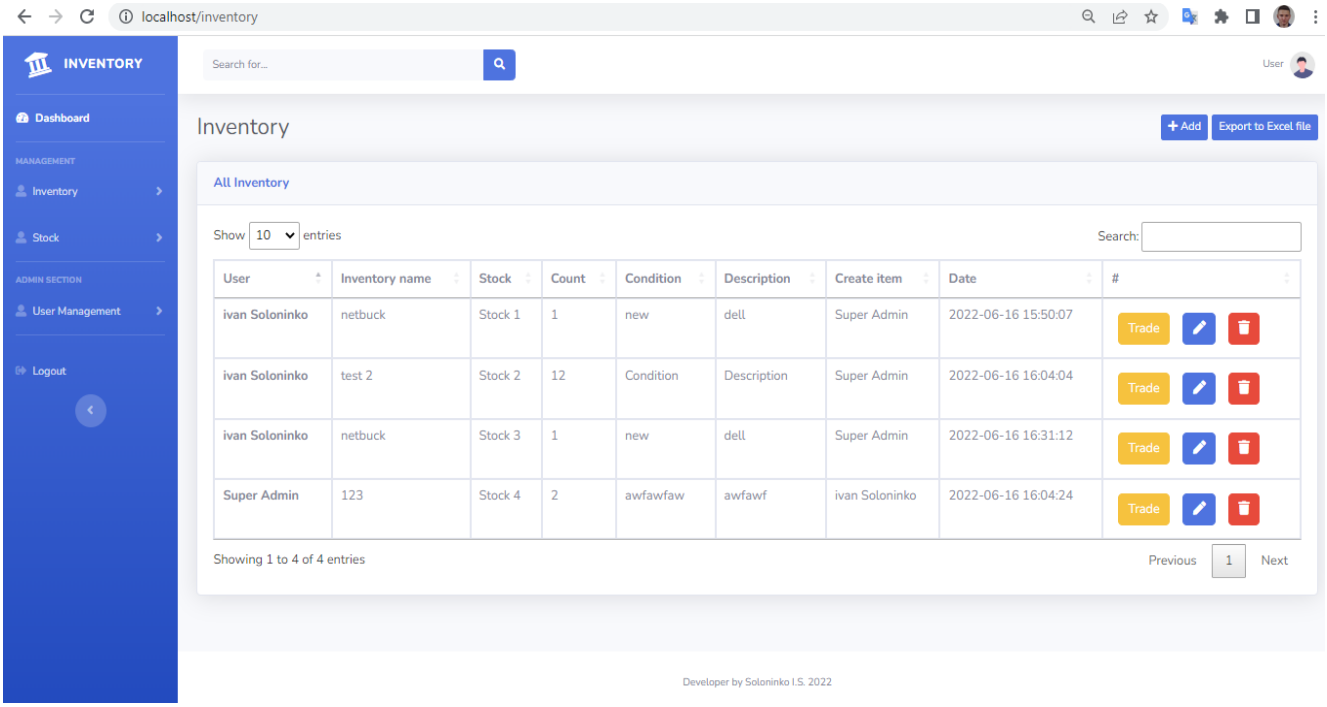


Рис.2.18. Вигляд сторінки inventory

Кнопки оформлені у різній кольоровій гамі для кращого візуального сприйняття.

Висновки до розділу 2

Зважаючи на сформовану в першому розділі задачу та технічне завдання ми визначилися з даними, які будуть заповнювати додаток та описали процедуру їх обміну в процесі.

Задля структуризації даних їх було об'єднано в таблиці. Оскільки наявність зв'язку між даними є обов'язковою(без цього додаток не може повноцінно працювати), то були описані зв'язки між ними та створена схема даних. Під час її проектування та розробки я поглибив свої знання MySQL і phpMyAdmin.

Серверна архітектура проекту формувалася на основі патерну MVC. Під час формування мною було детально досліджено його застосування у фреймворку Laravel. При розробці фронтенду проекту мною було вивчено нові інструменти, що надає платформа Bootstrap.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБ ЗАСТОСУНКУ

3.1 Засоби розробки

Visual Studio Code — засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим кросплатформовим продуктом у лінійці Visual Studio.

За основу для Visual Studio Code використовуються напрацювання вільного проєкту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовує браузерний рушій Chromium і Node.js. Примітно, що про використання напрацювань вільного проєкту Atom і на сайті Visual Studio Code, і в прес релізі, і в офіційному блозі не згадується.

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціюється як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe.Github – один з найбільших веб-сервісів для спільної розробки програмного забезпечення. Використовувався для зберігання версій проєкту.

Основні можливості та переваги програми:

• Visual Studio Code підтримує роботу з TypeScript, JavaScript, Node.js та Mono.

• Є вбудовані налагоджувач та командний рядок.

• Підтримка майже всіх мов програмування.

• Наявність вбудованої бібліотеки елементів коду.

• Автозавершення під час введення коду.

• Додавання до бібліотеки власних сніпетів.

• Підсвічування синтаксису.

• Одночасна робота з кількома проектами.

• Підтримка багатовіконного та двопанельного режимів.

• Розширення функціоналу за допомогою плагінів.

• Інтеграція з Visual Studio Team Services, GitHub та GIT.

• Наявність вбудованих засобів для тестування, складання, упаковки та розгортання програм.

• Публікація створених програмних продуктів у Microsoft Azure (через Visual Studio Team Services).

• Інтегрована система підказок.

• Командна робота над проектами.

• Широкий набір налаштувань та кроссплатформенність.

3.2 Вимоги до технічного та програмного забезпечення

Так як проект хоститься локально необхідно було вирішити яким чином зручніше підключити базу даних до мережі, так як у нас базою даних є MySQL, а веб-сервером – Apache, я обрав XAMPP. Це безкоштовна багатоплатформова збірка веб сервера з відкритим початковим кодом, що містить всі потрібні компоненти.

Для адміністрування бази даних я використав phpMyAdmin. Саме його пропонує XAMPP. phpMyAdmin — веб додаток з відкритим кодом, написаний мовою PHP із графічним веб інтерфейсом для адміністрування бази даних MySQL або MariaDB. phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Ця програма користується великою популярністю у веб розробників, оскільки дозволяє керувати базу даних MySQL без вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера

під'єднаного до інтернету без необхідності встановлення додаткового програмного забезпечення.

На сьогоднішній день phpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, з огляду на всі нововведення СУБД MySQL. Переважна більшість українських провайдерів використовують цей застосунок як панель керування для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних.

3.3 Опис програмної реалізації

Програмна реалізація проєкту «Inventory» розпочалася з підбору стеку технологій для реалізації поставленого завдання.

В першу чергу я встановив XAMP, який відповідає за серверну частину веб застосунку, та laravel, як фреймворк, за допомогою якого буде створюватися проєкт.

Для frontend частини проекту обрав bootstrap шаблон, а компілював її за допомогою laravel mix. Laravel Mix - це чистий та гнучкий API для визначення інструкцій складання Webpack для вашого Laravel-додатка з використанням кількох основних препроцесорів CSS та JavaScript.

Роботу було розпочато з створення нового проєкту. Для цього я використав команду laravel new tech.

Далі я перейшов до встановлення bootstrap адмін-панелі. За допомогою терміналу та команди

- - “git clone https://github.com/StartBootstrap/startbootstrap-sb-admin-2.git”.

Насамперед я підключив проект до бази даних за допомогою файлу .env та v-hosts(Додаток А)

Сама по собі адмін панель є лише набором стилізованих шаблонів. Отож, після редагування панелі(Додаток Б), потрібно було реалізовувати механіку роботи.

Я почав з реалізації user менеджменту. Була створена модель User, яка описувала загальну сутність користувача.(Додаток В) та відповідні контроллери(Додаток Г)

Зробивши міграцію до бази даних (Додаток Д), та прописавши маршрути в файлі web.php (Додаток Є), процедури реєстрації та авторизації користувачів працювали.

В коді я створив адміністратора і за допомогою цих даних входив в систему.

Отож процедура реєстрації та авторизації була завершена. Далі перейшов до створення таблиці інвентарю. Аналогічно як і для користувачів, я створив відповідну модель. CRUD операції над інвентарем описав у відповідних контроллерах. Провівши міграцію бази даних таблиця була реалізована. Залишилася найважливіша частина – розподіл ролей. Її я реалізував ще під час реєстрації. Кожен новий користувач при реєстрації отримує статус user, а підвищити його рівень доступу може лише адміністратор. Отож після того, як я змінив код у контролері, що відповідає за доступ до таблиці інвентарю, звичайний користувач може додавати інвентар і бачити лише те, що додав він сам. Адміністратор, в свою чергу, може бачити весь інвентар, редагувати його та, для зручності, мною було реалізовано фільтрацію інвентарю за користувачем. Також аналогічним методом було реалізовано такі сутності як histories та stock. Доступ до складів має лише адміністратор, а от історію операцій може бачити і користувач, проте йому доступна лише історія того обладнання, в якому він задіяний.

3.4 Керівництво користувачу

Для того, щоб працювати в системі inventory користувачу потрібно зареєструватися. Рис3.1

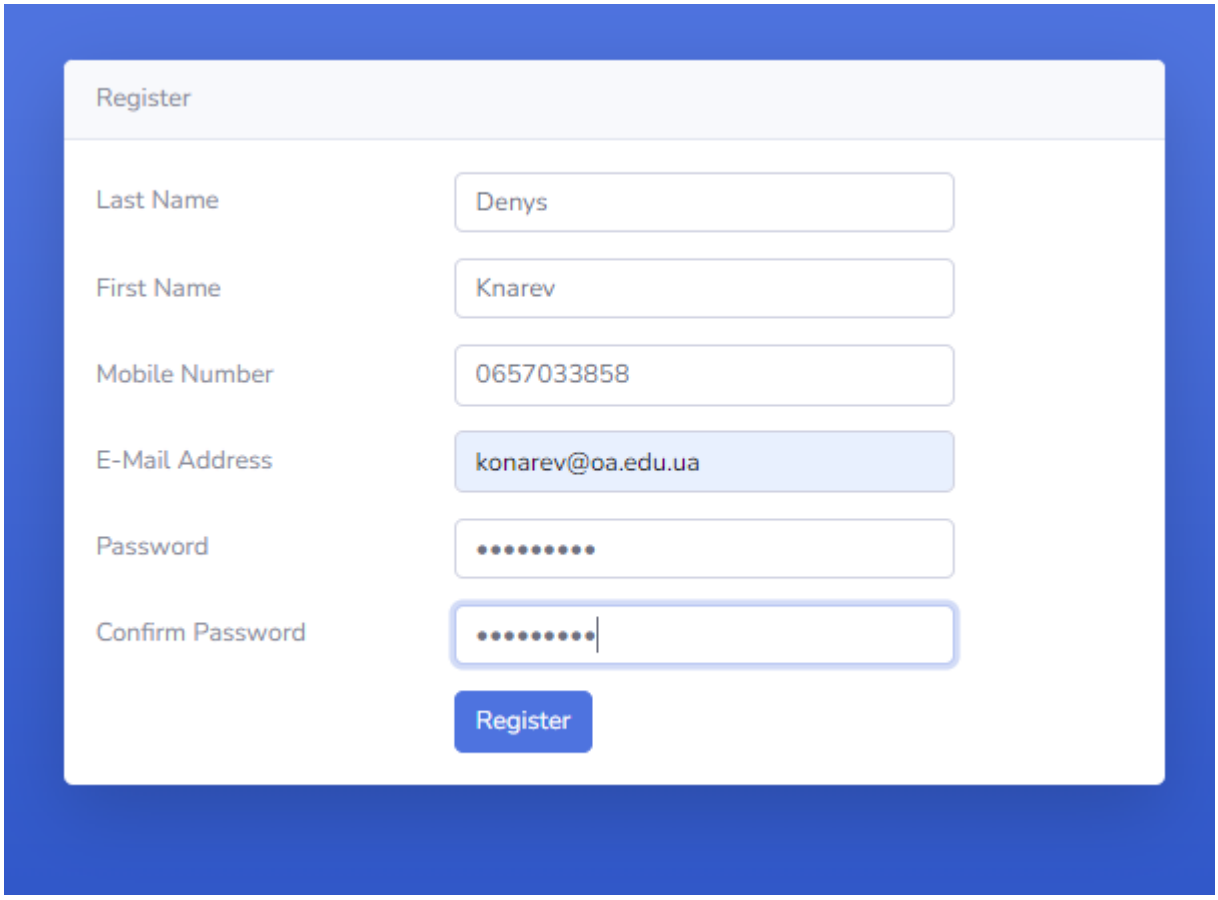


Рис.3.1 Процес реєстрації

Після успішної реєстрації користувача перенаправляє на головну сторінку.Рис.3.2

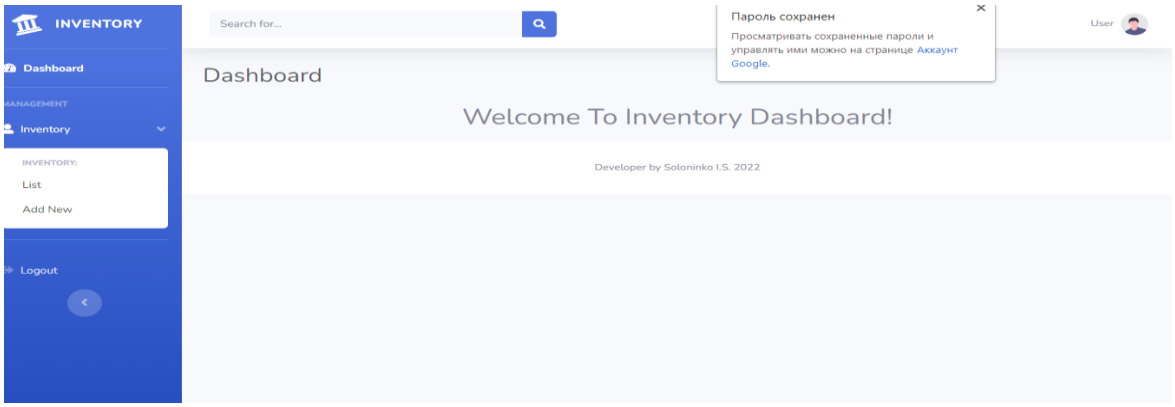


Рис.3.2 Головна сторінка авторизованого користувача.

Для додання інвентарю потрібно натиснути на Inventory -> Add New

Заповнивши дані про інвентар, куди його передали та хто прийняв натискаємо кнопку save

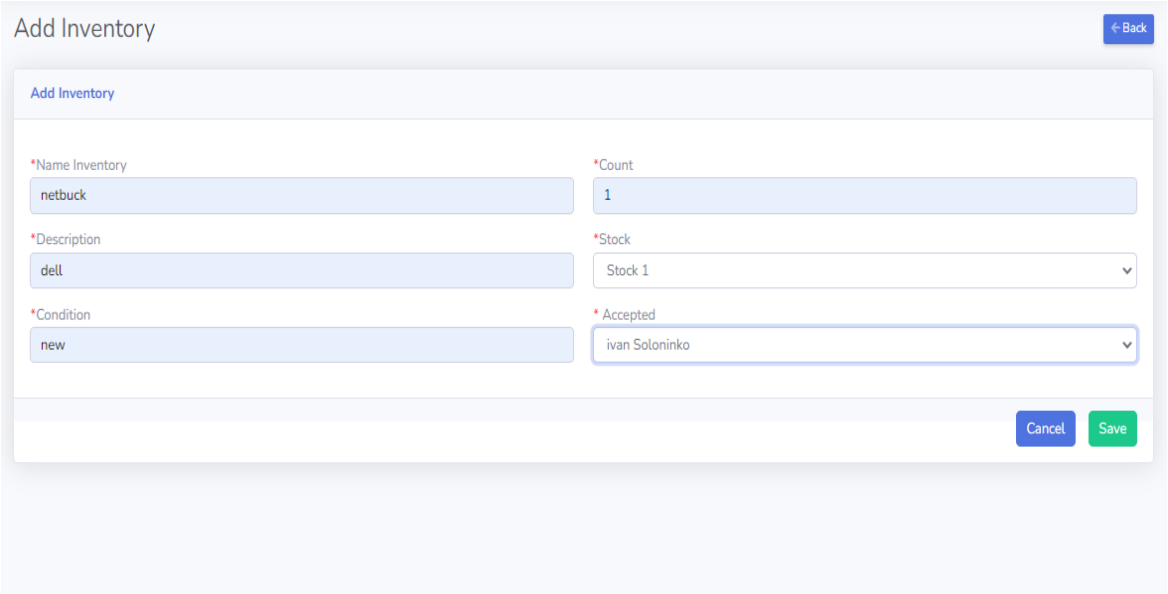


Рис.3.3. Додаємо інвентар

Вас перенаправить на сторінку з всім, доданим вами інвентарем.

Також ви можете переглянути та змінити свій профіль. Для цього потрібно перейти на сторінку profile. (Рис.3.4.)

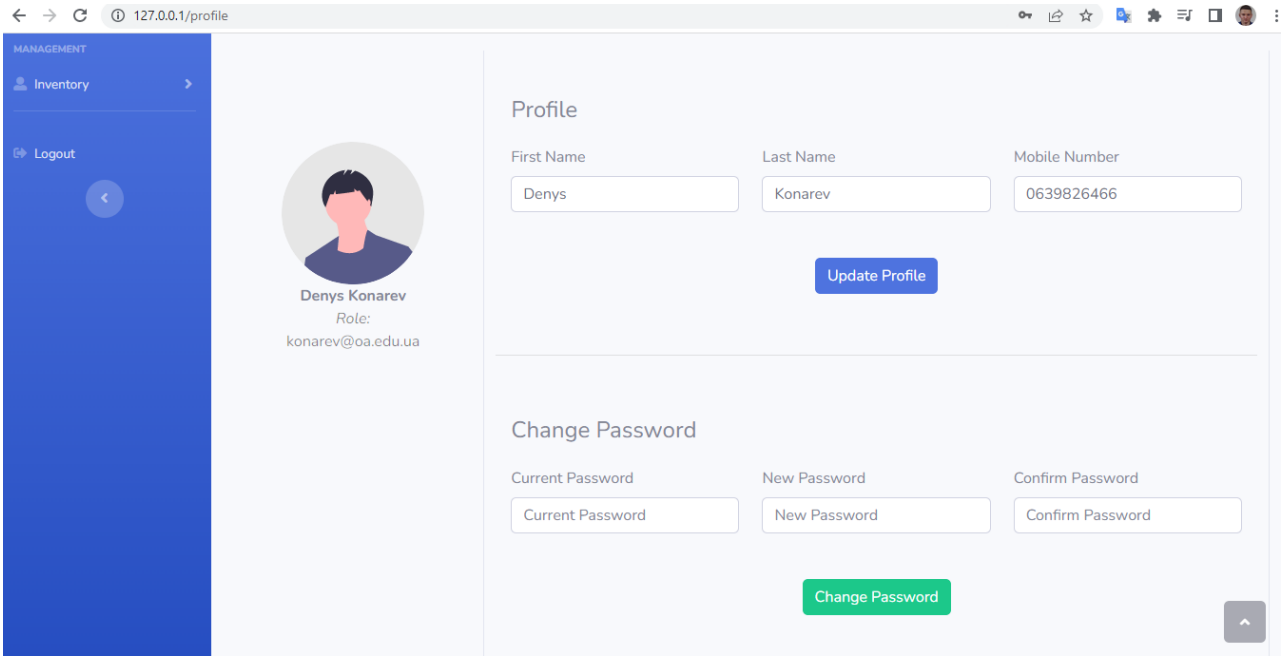


Рис.3.4.Профіль користувача

Подальші дії над інвентарем виконуються адміністратором. Для йому потрібно авторизуватися в системі під своїм логіном та паролем. На головній сторінці він одразу буде бачити дії, що ви виконали. Вигляд панелі адміністратора показано на рисунку 3.5.

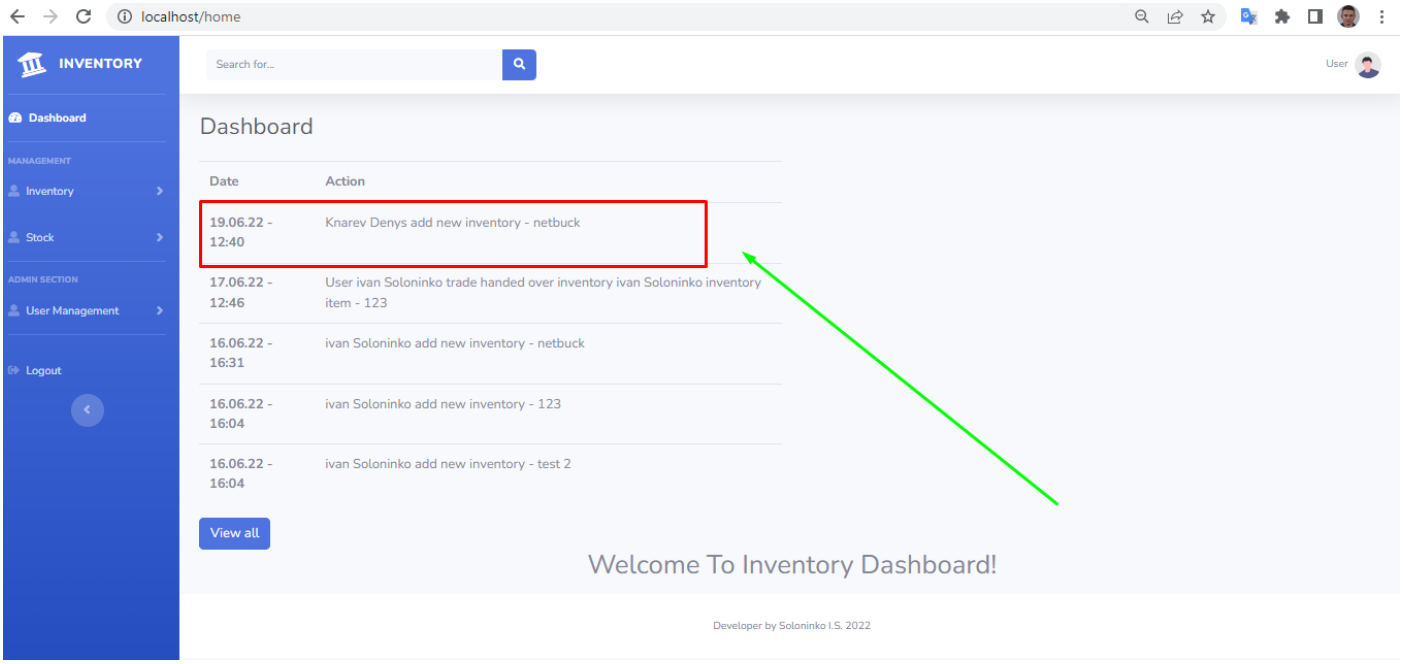


Рис.3.5.Панель користувача з правами адміністратора

Перейшовши Inventory -> List ми можемо бачити весь інвентар.

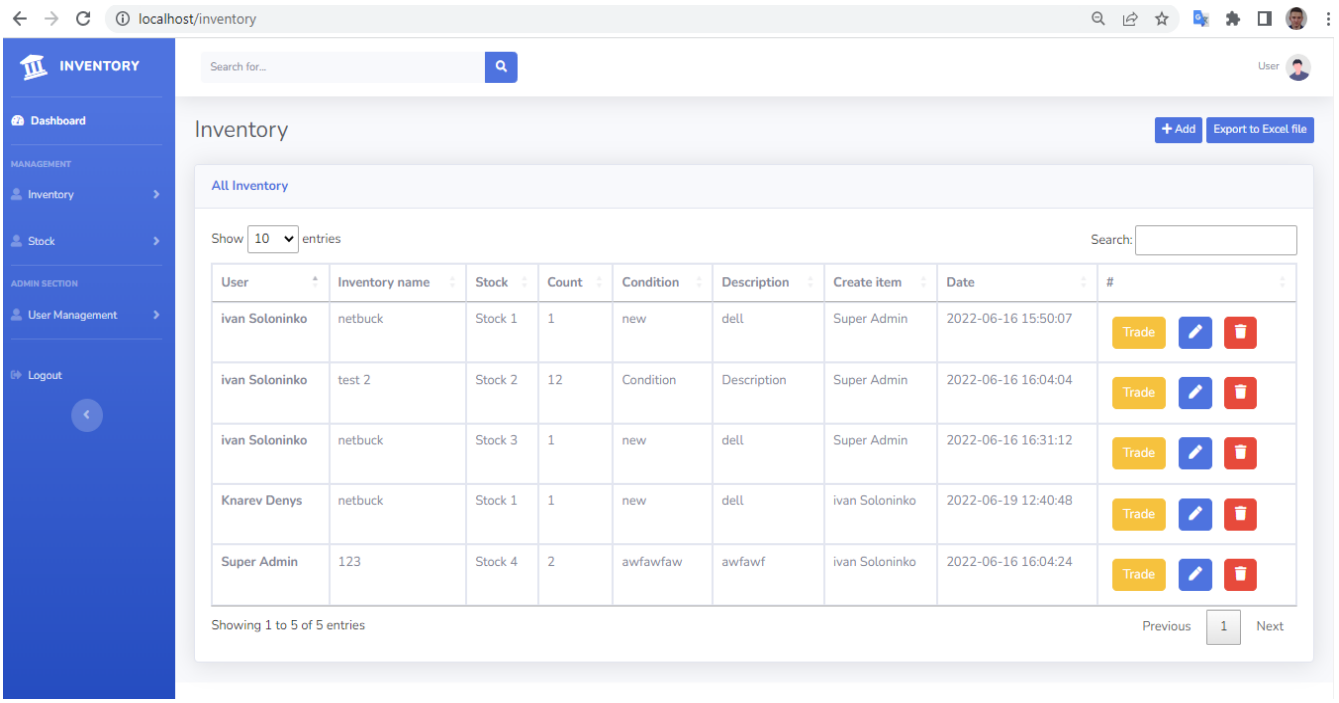


Рис. 3.6. Весь інвентар

Адміністратор може змінити місцезнаходження, стан і опис інвентарю натиснувши кнопку редагувати.

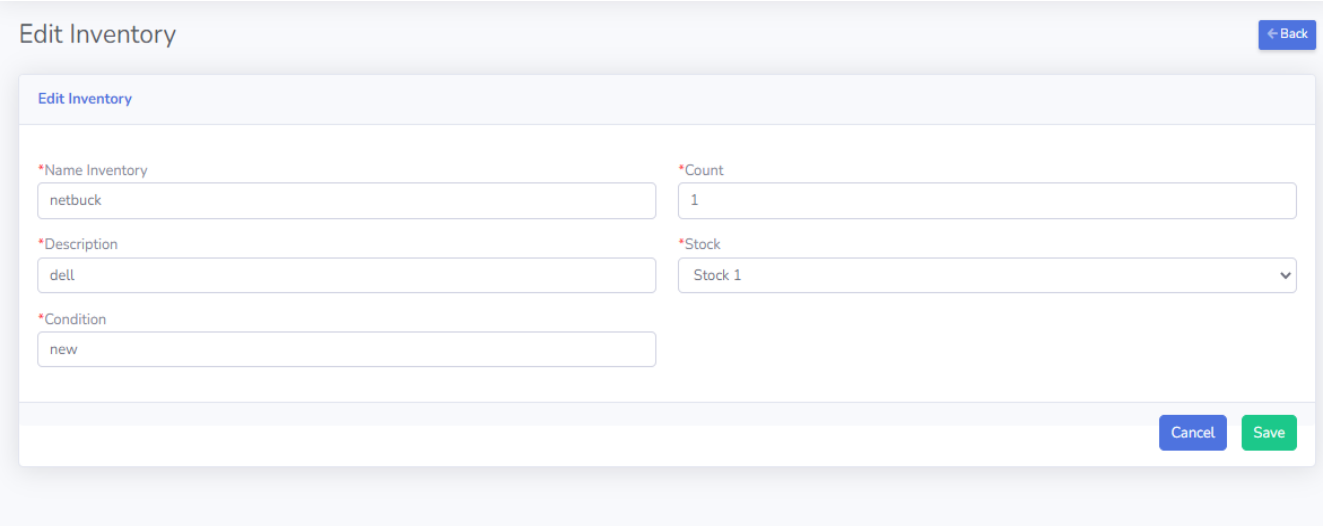


Рис.3.7. Редагування інвентарю

Для того, щоб передати інвентар достатньо натиснути кнопку “Trade”

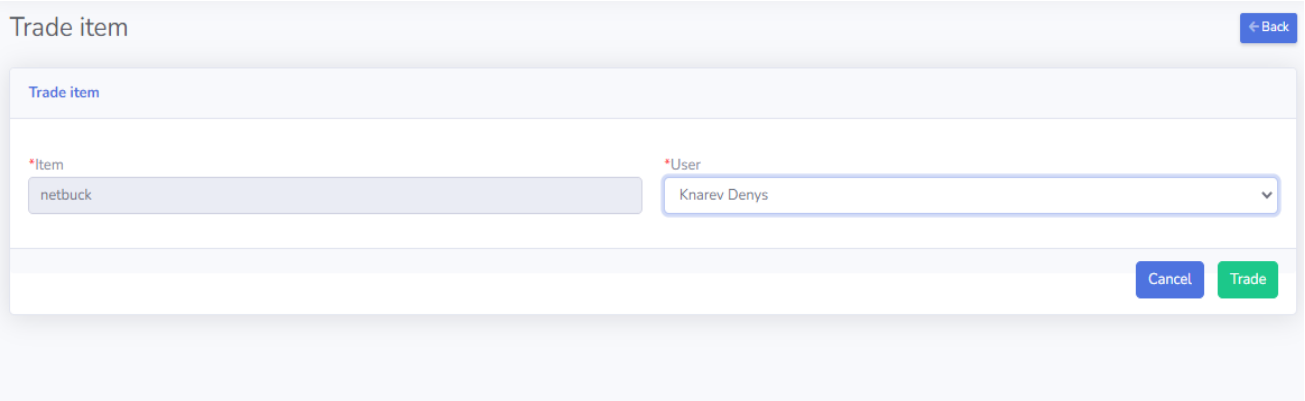


Рис.3.8. Передача інвентарю

Також можна видалити інвентар натиснувши на відповідну кнопку. Адміністратору доступна секція склади, та можливість їх створювати та редагувати.

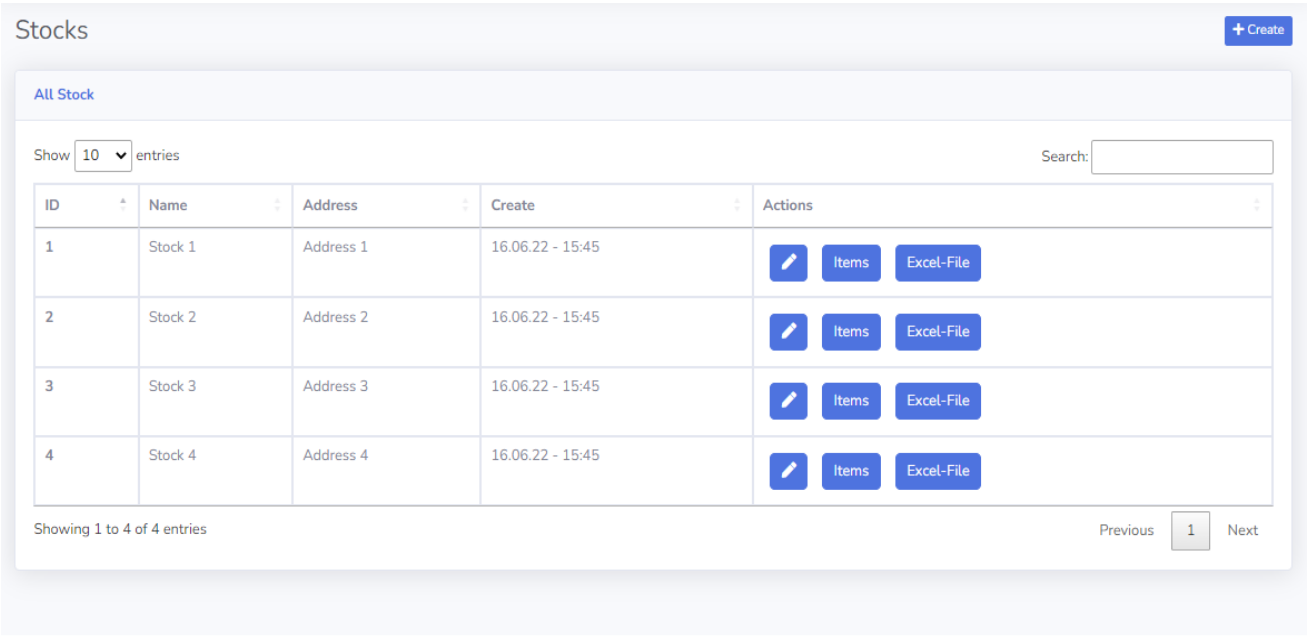


Рис.3.9. Склади

Користувачу з правами адміністратора доступна вкладка User Management

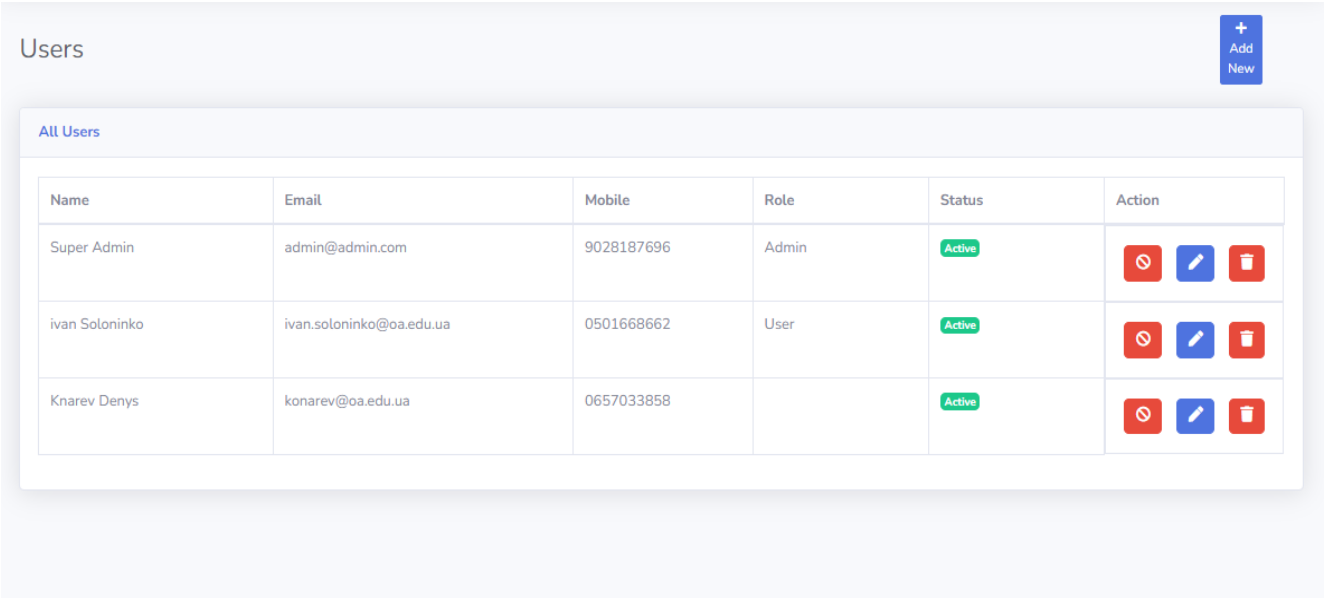


Рис.3.10. Таблиця Users

Як ми бачимо, адміністратор може банити, редагувати та видаляти користувачі.(Рис. 3.10-3.12)

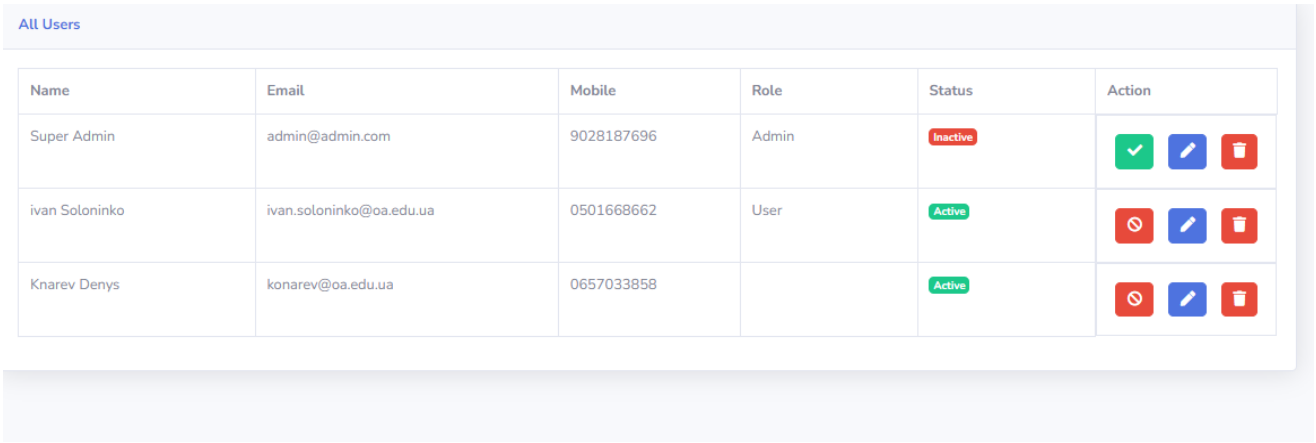


Рис.3.11. Бан користувача

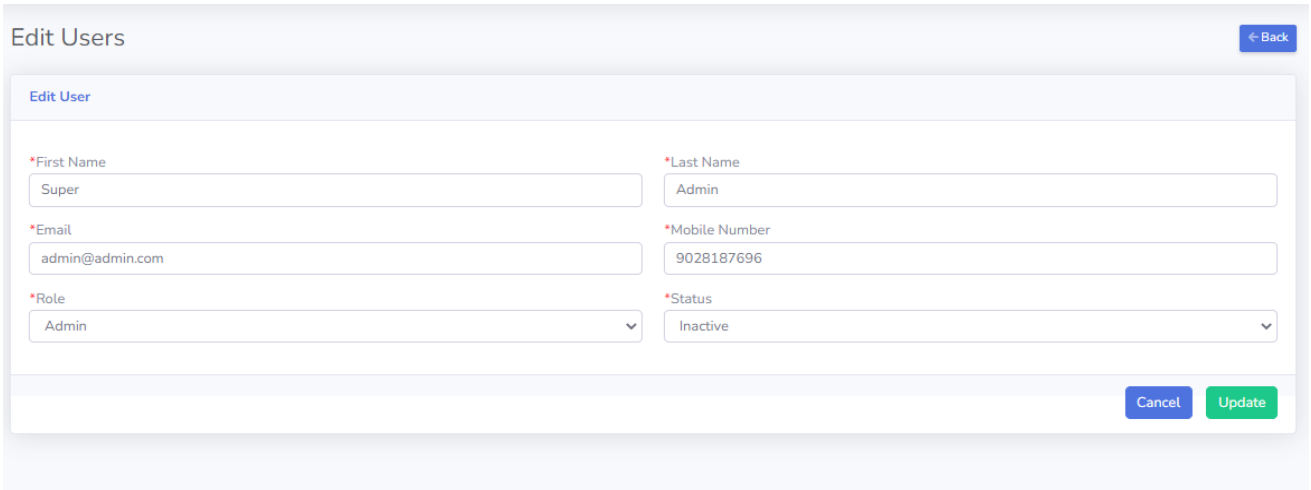


Рис. 3.11. Редагування користувача

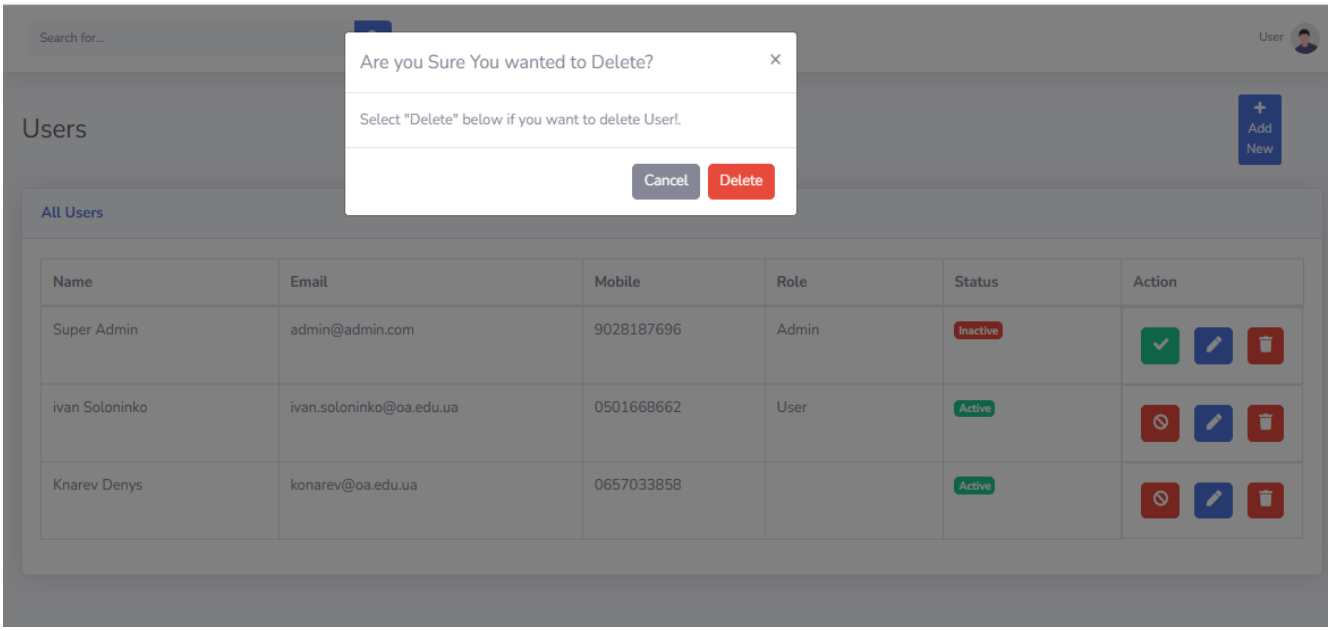


Рис. 3.12. Видалення користувача

Висновок до розділу 3

Отже, в даному розділі ми описали засоби, які використовуємо у нашому проекті. Це VS Code, Composer, XAMPP, phpMyAdmin. Розглянули їх переваги та недоліки. У розділі опис програмної реалізації ми детально поговорили в про дві основні моделі User та Inventory. Механізм їхньої роботи описують контролери, про які теж було сказано в розділі. Описане мною керівництво включало в себе керівництво як для звичайного користувача, так і для адміністратора. Разом з графічними матеріалами воно допоможе будь-кому скористатися веб-додатком.

ВИСНОВКИ

Підводячи підсумки, можна зробити такі висновки, результатом кваліфікаційної роботи стала реалізована інформаційна система фіксування обліку матеріалів при сервісному обслуговуванні, яка дає змогу додавати інвентар та проводити його облік , а також – здійснювати користувацькі налаштування. Оглянуто наявні аналоги та проведено паралелі із нашою системою.

Описано предметне середовище та технології які використовувалися в ході написання кваліфікаційної роботи, а саме: PHP 8, Laravel, MVC, Composer, Node.js, npm, bootstrap, vue.js.

Сформовано певні задачі які постали при написанні системи, і на основі цих задач було сформоване технічне завдання.

Для створення інформаційної система фіксування обліку матеріалів при сервісному обслуговуванні відбулася розробка бази даних, спочатку це відбулося на певні блок схемі, а далі за допомогою СУБД phpMyAdmin За основну систему була обрана MySQL.

Створена модель бази даних дозволяє користувачам реєструватися та додавати новий інвентар,а адміністратору – проводити його облік та подальше редагування. Було описано моделі бази даних, а саме «Users» і «Inventory».

Структура проекту як і на серверній так і на клієнтській частині є багатошаровою.

Основний функціонал серверної частини був реалізований за допомогою фреймворку Laravel на базі PHP 8.

Весь проект реалізовувався в редакторі Visual Studio Code, який є зручним та масштабованим за своїм функціоналом.

Для написання front-end був використаний безкоштовний набір інструментів з відкритим кодом Bootstrap.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HTTPS-Vue.js URL: https://vue3js.cn/vuex/ru/#что-такое-“паттерн-управления-состоянием” (дата звернення 05.04.2022).
2. Що таке MVC URL: https://uk.education-wiki.com/3886982-what-is-mvc (дата звернення 05.04.2022).
3. Вікіпедія MySQL URL: https://uk.wikipedia.org/wiki/MySQL(дата звернення 14.04.2022).
4. Вікіпедія XAMP URL: https://uk.wikipedia.org/wiki/XAMPP(дата звернення 01.04.2022).
5. Вікіпедія Vue.js URL: https://ru.wikipedia.org/wiki/Vue.js (дата звернення 02.04.2022).
6. Вікіпедія Git Hub URL: https://uk.wikipedia.org/wiki/GitHub (дата звернення 01.05.2022).
7. Вікіпедія Visual Studio Code URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення 01.04.2022).
8. Хабр URL: https://habr.com/ru/company/simbirsoft/blog/416925/ (дата звернення 08.05.2022).
9. Вікіпедія PHP URL: https://uk.wikipedia.org/wiki/PHP(дата звернення 07.04.2022).
10. Вікіпедія Node.js URL: https://uk.wikipedia.org/wiki/Node.js (дата звернення 07.05.2022).
11. Вікіпедія Bootstrap URL: https://uk.wikipedia.org/wiki/Bootstrap (дата звернення 08.04.2022).
12. GitHub URL: https://github.com/StartBootstrap/startbootstrap-sb-admin-2 (дата звернення 08.04.2022).
13. 1С логістика URL: https://solutions.1c.ru/catalog/tms/features (дата звернення 10.04.2022).
14. npm URL: https://www.npmjs.com/(дата звернення 08.04.2022).
15. Node.js URL: https://nodejs.dev/learn (дата звернення 15.05.2022).

16. Laravel URL: https://www.tutorialspoint.com/laravel/index.htm (дата звернення 05.03.2022).

ДОДАТОК А

Файли .env та httpd-vhosts.conf

Лістінг програми

Аркушів 4

Острог 2022

Файл .env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:mwD39BVaRAPub5pESo6W6BPSHYQ2Jn71WX1lGOmvIT8=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
```

```
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"



AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false
```

```
PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1


MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Файл httpd-vhosts.conf

```
# Virtual Hosts
#
# Required modules: mod_log_config

# If you want to maintain multiple domains/hostnames on your
```

```
# machine you can setup VirtualHost containers for them. Most configurations
# use only name-based virtual hosts so the server doesn't need to worry about
# IP addresses. This is indicated by the asterisks in the directives below.
#
# Please see the documentation at
# <URL:http://httpd.apache.org/docs/2.4/vhosts/>
# for further details before you try to setup virtual hosts.
#
# You may use the command line option '-S' to verify your virtual host
# configuration.

#
# Use name-based virtual hosting.
#
##NameVirtualHost *:80
#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for all requests that do not
# match a ##ServerName or ##ServerAlias in any <VirtualHost> block.
#
##<VirtualHost *:80>
    ##ServerAdmin webmaster@dummy-host.example.com
    ##DocumentRoot "E:/xampp/htdocs/dummy-host.example.com"
    ##ServerName dummy-host.example.com
    ##ServerAlias www.dummy-host.example.com
    ##ErrorLog "logs/dummy-host.example.com-error.log"
    ##CustomLog "logs/dummy-host.example.com-access.log" common
##</VirtualHost>

##<VirtualHost *:80>
    ##ServerAdmin webmaster@dummy-host2.example.com
    ##DocumentRoot "E:/xampp/htdocs/dummy-host2.example.com"
    ##ServerName dummy-host2.example.com
    ##ErrorLog "logs/dummy-host2.example.com-error.log"
    ##CustomLog "logs/dummy-host2.example.com-access.log" common
##</VirtualHost>

 <VirtualHost *:80>
```

```
    ServerAdmin webmaster@dummy-host.example.com

   DocumentRoot "E:\xampp\htdocs\tech\public"

    ServerName tech.com

    ErrorLog "logs/tech.log"

</VirtualHost>
```

ДОДАТОК Б

Основний frontend адмін панелі

Лістінг програми

Аркушів 12

Острог 2022

login.blade.php

```
@extends('auth.layouts.app')

@section('title', 'Login')

@section('content')
<div class="row justify-content-center">

    <div class="text-center mt-5">
        <h1 class="text-white">Welcome To  Inventory</h1>
    </div>

    <div class="col-xl-10 col-lg-12 col-md-9">
        <div class="card o-hidden border-0 shadow-lg my-5">
            <div class="card-body p-0">
                <!-- Nested Row within Card Body -->
```

```
                <div class="row">
                    <div class="col-lg-6 d-none d-lg-block bg-login-image"></div>
                    <div class="col-lg-6">
                        <div class="p-5">
                            <div class="text-center">
                                <h1 class="h4 text-gray-900 mb-4">Log in!</h1>
                            </div>

                            @if (session('error'))
                                <span class="text-danger"> {{ session('error')
}}</span>
                            @endif

                            <form method="POST" action="{{ route('login') }}">
                                @csrf
                                <div class="form-group">
                                    <input id="email" type="email" class="form-
control form-control-user @error('email') is-invalid @enderror" name="email"
value="{{ old('email') }}" required autocomplete="email" autofocus
placeholder="Enter Email Address.">

                                        @error('email')
                                            <span class="invalid-feedback"
role="alert">

                                                <strong>{{ $message }}</strong>
                                            </span>
                                        @enderror
                                </div>
                                <div class="form-group">
                                    <input id="password" type="password"
class="form-control form-control-user @error('password') is-invalid @enderror"
name="password" required autocomplete="current-password" placeholder="Password">

                                    @error('password')
                                        <span class="invalid-feedback"
role="alert">

                                            <strong>{{ $message }}</strong>
                                        </span>
                                    @enderror
                                </div>
```

```
                                <div class="form-group">
                                    <div class="custom-control custom-checkbox
small">
                                        <input class="custom-control-input"
type="checkbox" name="remember" id="customCheck" {{ old('remember') ? 'checked' :
'' }}>

                                        <label class="custom-control-label"
for="customCheck">Remember
                                            Me</label>
                                    </div>
                                </div>
                                <button class="btn btn-primary btn-user btn-block">
                                    Login
                                </button>
                            </form>
                            <hr>

                            <a href="{{ route('register')}}" type="button"
class="btn btn-primary btn-user btn-block">
                                    Register
</a>
                            <hr>
                            <div class="text-center">
                                <a class="small"
href="{{route('password.request')}}">Forgot Password?</a>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>



</div>
@endsection
```

register.blade.php

```
@extends('auth.layouts.app')

@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Register') }}</div>

                <div class="card-body">
                    <form method="POST" action="{{ route('register') }}">
                        @csrf

                        <div class="row mb-3">
                            <label for="last_name" class="col-md-4 col-form-label
text-md-end">{{ __('Last Name') }}</label>

                            <div class="col-md-6">
                                <input id="last_name" type="text" class="form-
control @error('last_name') is-invalid @enderror" name="last_name" value="{{
old('last_name') }}" required autocomplete="last_name" autofocus>

                                @error('last_name')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="first_name" class="col-md-4 col-form-label
text-md-end">{{ __('First Name') }}</label>

                            <div class="col-md-6">
                                <input id="first_name" type="text" class="form-
control @error('first_name') is-invalid @enderror" name="first_name" value="{{
old('first_name') }}" required autocomplete="first_name" autofocus>
```

```
                                @error('first_name')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="mobile_number" class="col-md-4 col-form-
label text-md-end">{{ __('Mobile Number') }}</label>

                            <div class="col-md-6">
                                <input id="mobile_number" type="text" class="form-
control @error('mobile_number') is-invalid @enderror" name="mobile_number"
value="{{ old('mobile_number') }}" required autocomplete="mobile_number" autofocus>

                                @error('mobile_number')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="email" class="col-md-4 col-form-label text-
md-end">{{ __('E-Mail Address') }}</label>

                            <div class="col-md-6">
                                <input id="email" type="email" class="form-control
@error('email') is-invalid @enderror" name="email" value="{{ old('email') }}"
required autocomplete="email">

                                @error('email')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
```

```
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="password" class="col-md-4 col-form-label
text-md-end">{{ __('Password') }}</label>

                            <div class="col-md-6">
                                <input id="password" type="password" class="form-
control @error('password') is-invalid @enderror" name="password" required
autocomplete="new-password">

                                @error('password')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="password-confirm" class="col-md-4 col-form-
label text-md-end">{{ __('Confirm Password') }}</label>

                            <div class="col-md-6">
                                <input id="password-confirm" type="password"
class="form-control" name="password_confirmation" required autocomplete="new-
password">
                            </div>
                        </div>

                        <div class="row mb-0">
                            <div class="col-md-6 offset-md-4">
                                <button type="submit" class="btn btn-primary">
                                    {{ __('Register') }}
                                </button>
                            </div>
                        </div>
                    </form>
                </div>
```

```
            </div>
        </div>
    </div>
</div>
@endsection
```

app.blade.php

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

{{-- Include Head --}}
@include('common.head')

<body id="page-top">

    <!-- Page Wrapper -->
    <div id="wrapper">

        <!-- Sidebar -->
        @include('common.sidebar')
        <!-- End of Sidebar -->

        <!-- Content Wrapper -->
        <div id="content-wrapper" class="d-flex flex-column">

            <!-- Main Content -->
            <div id="content">

                <!-- Topbar -->
                @include('common.header')
                <!-- End of Topbar -->

                <!-- Begin Page Content -->
                @yield('content')
                <!-- /.container-fluid -->

            </div>
            <!-- End of Main Content -->
```

```
            <!-- Footer -->
            @include('common.footer')
            <!-- End of Footer -->

        </div>
        <!-- End of Content Wrapper -->

    </div>
    <!-- End of Page Wrapper -->

    <!-- Scroll to Top Button-->
    <a class="scroll-to-top rounded" href="#page-top">
        <i class="fas fa-angle-up"></i>
    </a>

    <!-- Logout Modal-->
    @include('common.logout-modal')

    <!-- Bootstrap core JavaScript-->
    <script src="{{asset('js/app.js')}}"></script>

    <!-- Custom scripts for all pages-->
    <script src="{{asset('admin/js/sb-admin-2.min.js')}}"></script>
    <script
src="{{asset('//cdn.datatables.net/1.12.1/js/jquery.dataTables.min.js')}}"></script
>

    @yield('scripts')
</body>

</html>
```

Home.blade.php

```
@extends('layouts.app')

@section('title', 'Dashboard')
```

```
@section('content')
<div class="container-fluid">

    <!-- Page Heading -->
    <div class="d-sm-flex align-items-center justify-content-between mb-4">
        <h1 class="h3 mb-0 text-gray-800">Dashboard</h1>

    </div>

    <div class="row">
        <div class="col-md-12">
            <h2 class="text-center mb-3">Welcome To  Inventory Dashboard!</h2>
        </div>
    </div>

    <!-- Content Row -->
    <div class="row">

</div>
@endsection
```

profile.blade.php

```
@extends('layouts.app')

@section('title', 'Profile')

@section('content')
    <div class="container-fluid">

        <!-- Page Heading -->
        <div class="d-sm-flex align-items-center justify-content-between mb-4
border-bottom">
            <h1 class="h3 mb-0 text-gray-800">Profile</h1>
        </div>

        {{-- Alert Messages --}}
        @include('common.alert')

        {{-- Page Content --}}
```

```
        <div class="row">
            <div class="col-md-3 border-right">
                <div class="d-flex flex-column align-items-center text-center p-3
py-5">
                    <img class="rounded-circle mt-5" width="150px" src="{{
asset('admin/img/undraw_profile.svg') }}">
                    <span class="font-weight-bold">{{ auth()->user()->full_name
}}</span>
                    <span class="text-black-50"><i>Role:
                            {{ auth()->user()->roles
                                ? auth()->user()->roles->pluck('name')->first()
                                : 'N/A' }}</i></span>
                    <span class="text-black-50">{{ auth()->user()->email }}</span>
                </div>
            </div>
            <div class="col-md-9 border-right">
                {{-- Profile --}}
                <div class="p-3 py-5">
                    <div class="d-flex justify-content-between align-items-center
mb-3">
                        <h4 class="text-right">Profile</h4>
                    </div>
                    <form action="{{ route('profile.update') }}" method="POST">
                        @csrf
                        <div class="row mt-2">
                            <div class="col-md-4">
                                <label class="labels">First Name</label>
                                <input type="text" class="form-control
@error('first_name') is-invalid @enderror"
                                    name="first_name" placeholder="First Name"
                                    value="{{ old('first_name') ? old('first_name')
: auth()->user()->first_name }}">

                                @error('first_name')
                                    <span class="text-danger">{{ $message }}</span>
                                @enderror
                            </div>
                            <div class="col-md-4">
                                <label class="labels">Last Name</label>
                                <input type="text" name="last_name"
```

```
                                    class="form-control @error('last_name') is-
invalid @enderror"
                                    value="{{ old('last_name') ? old('last_name') :
auth()->user()->last_name }}"
                                    placeholder="Last Name">

                                @error('last_name')
                                    <span class="text-danger">{{ $message }}</span>
                                @enderror
                            </div>
                            <div class="col-md-4">
                                <label class="labels">Mobile Number</label>
                                <input type="text" class="form-control
@error('mobile_number') is-invalid @enderror" name="mobile_number"
                                    value="{{ old('mobile_number') ?
old('mobile_number') : auth()->user()->mobile_number }}"
                                    placeholder="Mobile Number">
                                @error('mobile_number')
                                    <span class="text-danger">{{ $message }}</span>
                                @enderror
                            </div>
                        </div>
                        <div class="mt-5 text-center">
                            <button class="btn btn-primary profile-button"
type="submit">Update Profile</button>
                        </div>
                    </form>
                </div>

                <hr>
                {{-- Change Password --}}
                <div class="p-3 py-5">
                    <div class="d-flex justify-content-between align-items-center
mb-3">
                        <h4 class="text-right">Change Password</h4>
                    </div>

                    <form action="{{ route('profile.change-password') }}"
method="POST">
                        @csrf
```

```
                        <div class="row mt-2">
                            <div class="col-md-4">
                                <label class="labels">Current Password</label>
                                <input type="password" name="current_password"
class="form-control @error('current_password') is-invalid @enderror"
placeholder="Current Password" required>
                                @error('current_password')
                                    <span class="text-danger">{{ $message }}</span>
                                @enderror
                            </div>
                            <div class="col-md-4">
                                <label class="labels">New Password</label>
                                <input type="password" name="new_password"
class="form-control @error('new_password') is-invalid @enderror" required
placeholder="New Password">
                                @error('new_password')
                                    <span class="text-danger">{{ $message }}</span>
                                @enderror
                            </div>
                            <div class="col-md-4">
                                <label class="labels">Confirm Password</label>
                                <input type="password" name="new_confirm_password"
class="form-control @error('new_confirm_password') is-invalid @enderror" required
placeholder="Confirm Password">
                                @error('new_confirm_password')
                                    <span class="text-danger">{{ $message }}</span>
                                @enderror
                            </div>
                        </div>
                        <div class="mt-5 text-center">
                            <button class="btn btn-success profile-button"
type="submit">Change Password</button>
                        </div>
                    </form>
                </div>
            </div>

        </div>
```

```
    </div>
@endsection
```

alert.blade.php

```
{{-- Message --}}
@if (Session::has('success'))
    <div class="alert alert-success alert-dismissible" role="alert">
        <button type="button" class="close" data-dismiss="alert">
            <i class="fa fa-times"></i>
        </button>
        <strong>Success !</strong> {{ session('success') }}
    </div>
@endif

@if (Session::has('error'))
    <div class="alert alert-danger alert-dismissible" role="alert">
        <button type="button" class="close" data-dismiss="alert">
            <i class="fa fa-times"></i>
        </button>
        <strong>Error !</strong> {{ session('error') }}
    </div>
@endif
```

footer.blade.php

```
<footer class="sticky-footer bg-white">
    <div class="container my-auto">
        <div class="copyright text-center my-auto">
            <span>Developer by Soloninko I.S.     2022</span>
        </div>
    </div>
</footer>
```

head.blade.php

```
<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>{{ config('app.name', 'Inventory') }} | @yield('title')</title>

    {{-- ICON --}}
    <link rel="shortcut icon" type="image/jpg" href="{{ asset('images/icon.png') }}"/>

    <!-- Font Awesome UI KIT-->
    <script src="https://kit.fontawesome.com/f75ab26951.js" crossorigin="anonymous"></script>

    <link
        href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i"
        rel="stylesheet">

    <!-- Custom styles for this template-->
    <link href="{{asset('css/app.css')}}" rel="stylesheet">
    <link href="{{asset('admin/css/sb-admin-2.min.css')}}" rel="stylesheet">
    <link href="{{asset('//cdn.datatables.net/1.12.1/css/jquery.dataTables.min.css')}}" rel="stylesheet">

</head>
```

header.blade.php

```
<nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top shadow">

    <!-- Sidebar Toggle (Topbar) -->
```

```
    <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-circle mr-
3">
        <i class="fa fa-bars"></i>
    </button>

    <!-- Topbar Search -->
    <form
        class="d-none d-sm-inline-block form-inline mr-auto ml-md-3 my-2 my-md-0
mw-100 navbar-search">
        <div class="input-group">
            <input type="text" class="form-control bg-light border-0 small"
placeholder="Search for..."
                aria-label="Search" aria-describedby="basic-addon2">
            <div class="input-group-append">
                <button class="btn btn-primary" type="button">
                    <i class="fas fa-search fa-sm"></i>
                </button>
            </div>
        </div>
    </form>

    <!-- Topbar Navbar -->
    <ul class="navbar-nav ml-auto">

        <!-- Nav Item - User Information -->
        <li class="nav-item dropdown no-arrow">
            <a class="nav-link dropdown-toggle" href="#" id="userDropdown"
role="button"
                data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                <span class="mr-2 d-none d-lg-inline text-gray-600
small">User</span>
                <img class="img-profile rounded-circle"
                    src="{{asset('admin/img/undraw_profile.svg')}}">
            </a>
            <!-- Dropdown - User Information -->
            <div class="dropdown-menu dropdown-menu-right shadow animated--grow-in"
                aria-labelledby="userDropdown">
                <a class="dropdown-item" href="{{ route('profile.detail') }}">
                    <i class="fas fa-user fa-sm fa-fw mr-2 text-gray-400"></i>
                    Profile
```

```
                </a>
                <div class="dropdown-divider"></div>
                <a class="dropdown-item" href="#" data-toggle="modal" data-
target="#logoutModal">
                    <i class="fas fa-sign-out-alt fa-sm fa-fw mr-2 text-gray-
400"></i>
                    Logout
                </a>
            </div>
        </li>

    </ul>

</nav>
```

logout-modal.blade.php

```
<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel"
    aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
                <button class="close" type="button" data-dismiss="modal" aria-
label="Close">
                    <span aria-hidden="true">×</span>
                </button>
            </div>
            <div class="modal-body">Select "Logout" below if you are ready to end
your current session.</div>
            <div class="modal-footer">
                <button class="btn btn-secondary" type="button" data-
dismiss="modal">Cancel</button>
                <a class="btn btn-primary" href="{{ route('logout') }}"
                    onclick="event.preventDefault();
                    document.getElementById('logout-form').submit();">
                    Logout
                </a>
```

```
                <form id="logout-form" action="{{ route('logout') }}" method="POST"
style="display: none;">
                    {{ csrf_field() }}
                </form>
            </div>
        </div>
    </div>
</div>
```

ДОДАТОК В

Моделі

Лістінг програми

Аркушів 2

Острог 2022

Inventory.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Inventory extends Model
```

```
{
    use HasFactory;

    protected $fillable =
        [
            'name',
            'user_id',
            'name_inventory',
            'description',
            'stock',
            'condition'
        ];
}
```

User.php

```
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'first_name',
        'last_name',
```

```
        'email',
        'mobile_number',
        'role_id',
        'status',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    /**
     * Get the user's full name.
     *
     * @return string
     */
    public function getFullNameAttribute()
    {
        return "{$this->first_name} {$this->last_name}";
    }

}
```

History.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class History extends Model
{
    use HasFactory;

    protected $guarded = [];
}
```

Stock.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Stock extends Model
{
    use HasFactory;

    protected $guarded = [];

    public function iventory()
    {
        return $this->hasMany(Inventory::class);
    }
}
```

ДОДАТОК Г

Контроллери

Лістінг програми

Аркушів 15

Острог 2022

RegisterControlle.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use App\Models\User;
use Illuminate\Foundation\Auth\RegistersUsers;
```

```php
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Register Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new users as well as their
    | validation and creation. By default this controller uses a trait to
    | provide this functionality without requiring any additional code.
    |
    */

    use RegistersUsers;

    /**
     * Where to redirect users after registration.
     *
     * @var string
     */
    protected $redirectTo = RouteServiceProvider::HOME;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
```

```
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'first_name' => ['required', 'string', 'max:255'],
            'last_name' => ['required', 'string', 'max:255'],
            'mobile_number' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\Models\User
     */
    protected function create(array $data)
    {
        return User::create([
            'first_name' => $data['first_name'],
            'last_name' => $data['last_name'],
            'mobile_number' => $data['mobile_number'],
            'email' => $data['email'],
            'role_id' => '2',
            'password' => Hash::make($data['password']),
        ]);
    }
}
```

LoginControlle.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
```

```php
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = RouteServiceProvider::HOME;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

HomeControlle.php

```php
<?php
```

```php
namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use App\Rules\MatchOldPassword;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Http;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        return view('home');
    }

    /**
     * User Profile
     * @param Nill
     * @return View Profile
     * @author Shani Singh
     */
    public function getProfile()
    {
```

```php
        return view('profile');
    }

    /**
     * Update Profile
     * @param $profileData
     * @return Boolean With Success Message
     * @author Shani Singh
     */
    public function updateProfile(Request $request)
    {
        #Validations
        $request->validate([
            'first_name'    => 'required',
            'last_name'     => 'required',
            'mobile_number' => 'required|numeric|digits:10',
        ]);

        try {
            DB::beginTransaction();

            #Update Profile Data
            User::whereId(auth()->user()->id)->update([
                'first_name' => $request->first_name,
                'last_name' => $request->last_name,
                'mobile_number' => $request->mobile_number,
            ]);

            #Commit Transaction
            DB::commit();

            #Return To Profile page with success
            return back()->with('success', 'Profile Updated Successfully.');

        } catch (\Throwable $th) {
            DB::rollBack();
            return back()->with('error', $th->getMessage());
        }
    }
```

```
    /**
     * Change Password
     * @param Old Password, New Password, Confirm New Password
     * @return Boolean With Success Message
     * @author Shani Singh
     */
    public function changePassword(Request $request)
    {
        $request->validate([
            'current_password' => ['required', new MatchOldPassword],
            'new_password' => ['required'],
            'new_confirm_password' => ['same:new_password'],
        ]);

        try {
            DB::beginTransaction();

            #Update Password
            User::find(auth()->user()->id)->update(['password'=>
Hash::make($request->new_password)]);

            #Commit Transaction
            DB::commit();

            #Return To Profile page with success
            return back()->with('success', 'Password Changed Successfully.');

        } catch (\Throwable $th) {
            DB::rollBack();
            return back()->with('error', $th->getMessage());
        }
    }
}
```

InventoryControlle.php

```
<?php

namespace App\Http\Controllers;
```

```php
use App\Models\Inventory;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Spatie\Permission\Models\Permission;

class InventoryController extends Controller
{
    public function list()
    {
        $user = Auth::user();

        if ($user->hasRole('Admin') == true) {
            $inventory = Inventory::query()->orderBy('created_at')->get();

            return view('inventory.list', compact('inventory'));
        }

        $inventory = Inventory::query()->where('user_id', '=', $user->id)-
>orderBy('created_at')->get();

        return view('inventory.list', compact('inventory'));
    }

    public function add()
    {
        return view('inventory.add');
    }

    public function edit($id)
    {
        $inventory = Inventory::query()->where('id', $id)->first();

        return view('inventory.edit', compact('inventory'));
    }

    public function delete($id)
    {
        Inventory::query()->where('id', $id)->delete();
```

```
        return redirect()->route('inventory.list')->with('success', 'Delete
Inventory.');

    }

    public function store(Request $request)
    {
        // Validations
        $request->validate([
            'name_inventory' => 'required',
            'description' => 'required',
            'stock' => 'required',
            'condition' => 'required',
        ]);

        try {

           Inventory::create([
                'name' => Auth::user()->first_name . ' ' . Auth::user()->last_name,
                'user_id' => Auth::user()->id,
                'description' => $request->description,
                'name_inventory' => $request->name_inventory,
                'stock' => $request->stock,
                'condition' => $request->condition,
            ]);

            return redirect()->route('inventory.list')->with('success', 'User
Created Successfully.');

        } catch (\Throwable $th) {
            // Rollback and return with Error
            DB::rollBack();
            return redirect()->back()->withInput()->with('error', $th-
>getMessage());
        }
    }

    public function change($id,Request $request)
    {
```

```
        // Validations
        $request->validate([
            'name_inventory' => 'required',
            'description' => 'required',
            'stock' => 'required',
            'condition' => 'required',
        ]);

        try {

            Inventory::where('id',$id)->update([
                'description' => $request->description,
                'name_inventory' => $request->name_inventory,
                'stock' => $request->stock,
                'condition' => $request->condition,
            ]);

            return redirect()->route('inventory.list')->with('success', 'Inventory
Update Successfully.');

        } catch (\Throwable $th) {
            // Rollback and return with Error
            DB::rollBack();
            return redirect()->back()->withInput()->with('error', $th-
>getMessage());
        }
    }
}
```

RolesControlle.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Spatie\Permission\Models\Permission;
use Spatie\Permission\Models\Role;
```

```php
class RolesController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
        $this->middleware('permission:role-list|role-create|role-edit|role-delete',
['only' => ['index']]);
        $this->middleware('permission:role-create', ['only' =>
['create','store']]);
        $this->middleware('permission:role-edit', ['only' => ['edit','update']]);
        $this->middleware('permission:role-delete', ['only' => ['destroy']]);
    }

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $roles = Role::paginate(10);

        return view('roles.index', [
            'roles' => $roles
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
```

```
        $permissions = Permission::all();

        return view('roles.add', ['permissions' => $permissions]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        DB::beginTransaction();
        try {
            $request->validate([
                'name' => 'required',
                'guard_name' => 'required'
            ]);

            Role::create($request->all());

            DB::commit();
            return redirect()->route('roles.index')->with('success','Roles created
successfully.');
        } catch (\Throwable $th) {
            DB::rollback();
            return redirect()->route('roles.add')->with('error',$th->getMessage());
        }

    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
```

```
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $role = Role::whereId($id)->with('permissions')->first();

        $permissions = Permission::all();

        return view('roles.edit', ['role' => $role, 'permissions' =>
$permissions]);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        DB::beginTransaction();
        try {

            // Validate Request
            $request->validate([
                'name' => 'required',
                'guard_name' => 'required'
            ]);

            $role = Role::whereId($id)->first();

            $role->name = $request->name;
```

```
            $role->guard_name = $request->guard_name;
            $role->save();

            // Sync Permissions
            $permissions = $request->permissions;
            $role->syncPermissions($permissions);

            DB::commit();
            return redirect()->route('roles.index')->with('success','Roles updated
successfully.');
        } catch (\Throwable $th) {
            DB::rollback();
            return redirect()->route('roles.edit',['role' => $role])-
>with('error',$th->getMessage());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        DB::beginTransaction();
        try {

            Role::whereId($id)->delete();

            DB::commit();
            return redirect()->route('roles.index')->with('success','Roles deleted
successfully.');
        } catch (\Throwable $th) {
            DB::rollback();
            return redirect()->route('roles.index')->with('error',$th-
>getMessage());
        }
    }
}
```

UserControlle.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use App\Exports\UsersExport;
use App\Imports\UsersImport;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Spatie\Permission\Models\Role;
use Illuminate\Support\Facades\Hash;
use Maatwebsite\Excel\Facades\Excel;
use Illuminate\Support\Facades\Validator;
use Illuminate\Validation\Rules\Password;

class UserController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
        $this->middleware('permission:user-list|user-create|user-edit|user-delete', ['only' => ['index']]);
        $this->middleware('permission:user-create', ['only' => ['create','store', 'updateStatus']]);
        $this->middleware('permission:user-edit', ['only' => ['edit','update']]);
        $this->middleware('permission:user-delete', ['only' => ['delete']]);
    }

    /**
     * List User
     * @param Nill
```

```
     * @return Array $user
     * @author Shani Singh
     */
    public function index()
    {
        $users = User::with('roles')->paginate(10);
        return view('users.index', ['users' => $users]);
    }

    /**
     * Create User
     * @param Nill
     * @return Array $user
     * @author Shani Singh
     */
    public function create()
    {
        $roles = Role::all();

        return view('users.add', ['roles' => $roles]);
    }

    /**
     * Store User
     * @param Request $request
     * @return View Users
     * @author Shani Singh
     */
    public function store(Request $request)
    {
        // Validations
        $request->validate([
            'first_name'    => 'required',
            'last_name'     => 'required',
            'password'     => 'required',
            'email'         => 'required|unique:users,email',
            'mobile_number' => 'required|numeric|digits:10',
            'role_id'       =>  'required|exists:roles,id',
            'status'       =>  'required|numeric|in:0,1',
        ]);
```

```
        DB::beginTransaction();
        try {

            // Store Data
            $user = User::create([
                'first_name'    => $request->first_name,
                'last_name'     => $request->last_name,
                'email'         => $request->email,
                'mobile_number' => $request->mobile_number,
                'role_id'       => $request->role_id,
                'status'        => $request->status,
                'password'      => Hash::make($request->password)
            ]);

            // Delete Any Existing Role
            DB::table('model_has_roles')->where('model_id',$user->id)->delete();

            // Assign Role To User
            $user->assignRole($user->role_id);

            // Commit And Redirected To Listing
            DB::commit();
            return redirect()->route('users.index')->with('success','User Created
Successfully.');

        } catch (\Throwable $th) {
            // Rollback and return with Error
            DB::rollBack();
            return redirect()->back()->withInput()->with('error', $th-
>getMessage());
        }
    }

    /**
     * Update Status Of User
     * @param Integer $status
     * @return List Page With Success
     * @author Shani Singh
     */
```

```php
    public function updateStatus($user_id, $status)
    {
        // Validation
        $validate = Validator::make([
            'user_id'    => $user_id,
            'status'     => $status
        ], [
            'user_id'    =>  'required|exists:users,id',
            'status'     =>  'required|in:0,1',
        ]);

        // If Validations Fails
        if($validate->fails()){
            return redirect()->route('users.index')->with('error', $validate->errors()->first());
        }

        try {
            DB::beginTransaction();

            // Update Status
            User::whereId($user_id)->update(['status' => $status]);

            // Commit And Redirect on index with Success Message
            DB::commit();
            return redirect()->route('users.index')->with('success','User Status Updated Successfully!');
        } catch (\Throwable $th) {

            // Rollback & Return Error Message
            DB::rollBack();
            return redirect()->back()->with('error', $th->getMessage());
        }
    }

    /**
     * Edit User
     * @param Integer $user
     * @return Collection $user
     * @author Shani Singh
```

```php
     */
    public function edit(User $user)
    {
        $roles = Role::all();
        return view('users.edit')->with([
            'roles' => $roles,
            'user'  => $user
        ]);
    }

    /**
     * Update User
     * @param Request $request, User $user
     * @return View Users
     * @author Shani Singh
     */
    public function update(Request $request, User $user)
    {
        // Validations
        $request->validate([
            'first_name'    => 'required',
            'last_name'     => 'required',
            'email'         => 'required|unique:users,email,'.$user->id.',id',
            'mobile_number' => 'required|numeric|digits:10',
            'role_id'       =>  'required|exists:roles,id',
            'status'       =>  'required|numeric|in:0,1',
        ]);

        DB::beginTransaction();
        try {

            // Store Data
            $user_updated = User::whereId($user->id)->update([
                'first_name'    => $request->first_name,
                'last_name'     => $request->last_name,
                'email'         => $request->email,
                'mobile_number' => $request->mobile_number,
                'role_id'       => $request->role_id,
                'status'        => $request->status,
            ]);
```

```
            // Delete Any Existing Role
            DB::table('model_has_roles')->where('model_id',$user->id)->delete();

            // Assign Role To User
            $user->assignRole($user->role_id);

            // Commit And Redirected To Listing
            DB::commit();
            return redirect()->route('users.index')->with('success','User Updated
Successfully.');

        } catch (\Throwable $th) {
            // Rollback and return with Error
            DB::rollBack();
            return redirect()->back()->withInput()->with('error', $th-
>getMessage());
        }
    }

    /**
     * Delete User
     * @param User $user
     * @return Index Users
     * @author Shani Singh
     */
    public function delete(User $user)
    {
        DB::beginTransaction();
        try {
            // Delete User
            User::whereId($user->id)->delete();

            DB::commit();
            return redirect()->route('users.index')->with('success', 'User Deleted
Successfully!.');

        } catch (\Throwable $th) {
            DB::rollBack();
            return redirect()->back()->with('error', $th->getMessage());
```

```
        }
    }

    /**
     * Import Users
     * @param Null
     * @return View File
     */
    public function importUsers()
    {
        return view('users.import');
    }

    public function uploadUsers(Request $request)
    {
        Excel::import(new UsersImport, $request->file);

        return redirect()->route('users.index')->with('success', 'User Imported Successfully');
    }

    public function export()
    {
        return Excel::download(new UsersExport, 'users.xlsx');
    }

}
```

StockController.php

```
<?php

namespace App\Http\Controllers;

use App\Exports\StockExport;
use App\Models\History;
use App\Models\Stock;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
```

```php
use Maatwebsite\Excel\Facades\Excel;
use Throwable;

class StockController extends Controller
{
    public function index()
    {
        return view('stock.index', [
            'stocks' => Stock::query()->orderBy('id', 'desc')->get(),
        ]);
    }

    public function create()
    {
        return view('stock.create');
    }

    public function store(Request $request)
    {
        $this->validate($request, [
            'name' => 'required',
            'address' => 'required',
        ]);

        try {
            $stock = Stock::query()->create($request->only('name', 'address'));

            History::query()->create([
                'text' => "Admin: ".Auth::user()->full_name." create new stock
$stock[name]",
            ]);

            return redirect()->route('stock.index')->with('success', 'Stock Created
Successfully.');
        } catch (Throwable $error) {
            DB::rollBack();
            return redirect()->back()->withInput()->with('error', $error-
>getMessage());
        }
    }

    public function edit(Stock $stock)
    {
        return view('stock.edit', [
```

```
            'stock' => $stock,
        ]);
    }

    public function update(Request $request, Stock $stock)
    {
        $this->validate($request, [
            'name' => 'required',
            'address' => 'required',
        ]);

        try {
            $stock->update($request->only('name', 'address'));

            History::query()->create([
                'text' => "Admin: ".Auth::user()->full_name." edit stock
$stock[name]",
            ]);

            return redirect()->route('stock.index')->with('success', 'Stock Updated
Successfully.');
        } catch (Throwable $error) {
            DB::rollBack();
            return redirect()->back()->withInput()->with('error', $error-
>getMessage());
        }
    }

    public function export(Stock $stock)
    {
        return Excel::download(new StockExport($stock['id']), 'stock.xlsx');
    }

    public function items(Stock $stock)
    {
        return view('stock.items', [
            'stock' => $stock,
            'items' => $stock->iventory,
        ]); }}
```

ДОДАТОК Є

Маршрутизація

Лістінг програми

Аркушів 4

Острог 2022

Web.php

```php
<?php

use App\Http\Controllers\HomeController;
use App\Http\Controllers\InventoryController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Route;

/*
```

```
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return redirect()->route('login');
});

Auth::routes(['register' => true]);

Route::group(['middleware' => 'auth'], function () {
    Route::resource('stock', \App\Http\Controllers\StockController::class)->except('show', 'destroy');
    Route::get('stock/{stock}/export', [\App\Http\Controllers\StockController::class, 'export'])->name('stock.export');
    Route::get('stock/{stock}/items', [\App\Http\Controllers\StockController::class, 'items'])->name('stock.items');


    Route::get('history', [HomeController::class, 'showHistory'])->name('history');

    Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');
    Route::get('inventory', [InventoryController::class, 'list'])->name('inventory.list');
    Route::get('inventory/add', [InventoryController::class, 'add'])->name('inventory.add');
    Route::post('inventory/store', [InventoryController::class, 'store'])->name('inventory.store');
    Route::post('inventory/change/{id}', [InventoryController::class, 'change'])->name('inventory.change');

    Route::get('inventory/delete/{id}', [InventoryController::class, 'delete'])->name('inventory.delete');
    Route::get('inventory/edit/{id}', [InventoryController::class, 'edit'])->name('inventory.edit');
```

```
    Route::get('inventory/{inventory}/trade', [InventoryController::class,
'showFormTrade'])->name('inventory.trade');
    Route::post('inventory/{inventory}/trade', [InventoryController::class,
'tradeItem'])->name('inventory.trade');
    Route::get('inventory/export', [InventoryController::class, 'export'])-
>name('inventory.export');
});

// Profile Routes
Route::prefix('profile')->name('profile.')->middleware('auth')->group(function(){
    Route::get('/', [HomeController::class, 'getProfile'])->name('detail');
    Route::post('/update', [HomeController::class, 'updateProfile'])-
>name('update');
    Route::post('/change-password', [HomeController::class, 'changePassword'])-
>name('change-password');
});

// Roles
Route::resource('roles', App\Http\Controllers\RolesController::class);

// Permissions
Route::resource('permissions', App\Http\Controllers\PermissionsController::class);

// Users
Route::middleware('auth')->prefix('users')->name('users.')->group(function(){
    Route::get('/', [UserController::class, 'index'])->name('index');
    Route::get('/create', [UserController::class, 'create'])->name('create');
    Route::post('/store', [UserController::class, 'store'])->name('store');
    Route::get('/edit/{user}', [UserController::class, 'edit'])->name('edit');
    Route::put('/update/{user}', [UserController::class, 'update'])-
>name('update');
    Route::delete('/delete/{user}', [UserController::class, 'delete'])-
>name('destroy');
    Route::get('/update/status/{user_id}/{status}', [UserController::class,
'updateStatus'])->name('status');


    Route::get('/import-users', [UserController::class, 'importUsers'])-
>name('import');
    Route::post('/upload-users', [UserController::class, 'uploadUsers'])-
>name('upload');

    Route::get('export/', [UserController::class, 'export'])->name('export');
```

```
});
```