

NORTHWEST NAZARENE UNIVERSITY

Creating a Website with Laravel Framework for Northwest Nazarene University
Students to Buy and Sell Books to Each Other.

THESIS


Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

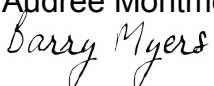
Audree Montmeny
2022


THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

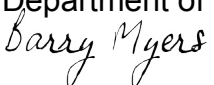
Audree Montmeny
2022

Creating a Website with Laravel Framework for Northwest Nazarene University
Students to Buy and Sell Books to Each Other.

Author: 
Audree Montmeny

Approved: 
Barry L. Myers, Ph.D., Professor of Computer Science,
Department of Mathematics & Computer Science, Faculty Advisor

Approved: 
Catherine Becker, Ph.D., Assistant Professor of English,
Department of English, Second Reader

Approved: 
Barry L. Myers, Ph.D., Chair,
Department of Mathematics & Computer Science

Abstract

Creating a Website with Laravel Framework for Northwest Nazarene University Students to Buy and Sell Books to Each Other.

MONTMENY, AUDREE (Department of Mathematics and Computer Science),
MYERS, DR. BARRY (Department of Mathematics and Computer Science)

While many websites, applications, platforms, etc., are available for current or previous students to buy and sell books to each other, none exist specifically for Northwest Nazarene University (NNU) students. The purpose of creating a website specifically for NNU students is convenience, safety, and reliability. In this project, a website was explicitly developed for NNU students using the features of the Laravel framework, including models, controllers, view, artisan, composer, and migrations. The website was built using HTML5, CSS, JavaScript, jQuery, and PHP. The project results are user login/register page, create book form, image upload, dashboard, edit, delete, and a search and sort ability has been built. Users can create a post about a book with the information of title, author, ISBN, subject, description, price, and image cover photo. User data is stored in a user table, while a separate table stores book data. While the main pages have been built, more work needs to be done to make the website more functional and user-friendly.

Acknowledgments

First, I would like to thank Zac Vineyard for generously sacrificing his time to provide help and guidance throughout the development of this project. Next, I would like to thank my friends, family, coaches, and teammates for supporting and encouraging me throughout my collegiate career as a student and athlete. Lastly, I would like to extend a special thank you to my professors, Dr. Myers, Dr. Hamilton, and Dr. McCarty, for educating and, most importantly, preparing me for the future.

Table of Contents

Abstract	iii
Acknowledgments	iv
Table of Contents	v
Table of Figures	vi
Introduction	1
Background	1
Development Environment	3
Laravel	3
Sublime Text	4
Sequel Pro and DBngin	4
Requirements	5
Website Requirements	5
Database Requirements	6
Implementation	6
Getting Started	6
Artisan	6
Composer	6
Model	8
View	8
Controller	8
Route	9
Database Implementation	10
Website Implementation	15
Welcome.blade.php	16
Dashboard.blade.php	17
Home.blade.php	18
Create.blade.php	19
Index.blade.php	21
Show.blade.php	25
Edit.blade.php	26
Conclusion	28
Future Work	29

References	31
Appendix A - Screenshots	32
Appendix B - Code	48

Table of Figures and Code

Figure 1 - Composer Setup	7
Figure 2 - Creating the Project	7
Figure 3 - Generating an Application Key	7
Figure 4 - Relationship Diagram	9
Figure 5 - Startup Database Screen	11
Figure 6 - Tables in Sequel Pro Database Application	12
Figure 7 - User Table in Database	13
Figure 8 - Books Table in Database (Pt. 1)	13
Figure 9 - Books Table in Database (Pt. 2)	14
Figure 10 - User Registration Form	15
Figure 11 - User Login Form	16
Figure 12 - Welcome.blade.php	17
Figure 13 - Dashboard.blade.php	18
Figure 14 - Home.blade.php	19
Figure 15 - Create Book Listing Form	20
Figure 16 - Store Function to Save Books to Database	21
Figure 17 - Book Posting List View	22
Figure 18 - Search Function	23
Figure 19 - Page View	24
Figure 20 - Index Function in BookController	24
Figure 21 - Show.blade.php View of Book Details	25
Figure 22 - Edit Book Post Form	27
Figure 23 - Update Function in BookController	28
Navigation.blade.php	48
Welcome.blade.php	54
Dashboard.blade.php	57
Home.blade.php	58
Create.blade.php	60
Index.blade.php	64
Show.blade.php	67
Edit.blade.php	69
BookController.php	72

Introduction

The primary goal of this project was to create a progressive website application that would allow users to safely and efficiently buy and sell books to one another. The target audience for this project is the Northwest Nazarene University (NNU) community, specifically, but not limited to, current, previous, and incoming students.

Background

Textbooks required for college courses are expensive. On average, the cost of books and supplies at a four-year institution for the 2020-21 school year was \$1,250 (U.S. Department of Education et al., 2021). While some students still elect to purchase new textbooks, many opt to purchase used textbooks to save money. Over the years, the used textbook market has become prevalent. Whether a student buys a new or used textbook, a great way to compensate for a portion of the cost is to sell the textbook to another student in need. Instead of letting textbooks that students no longer need sit around, they could be sold. Doing so would make better use of them and recover part of their expenditure.

Many websites, programs, and applications for users to buy and sell textbooks are available. Some of these popular websites include eBay, Facebook Marketplace, Chegg, Bookscout, etc... However, nothing has been developed specifically and exclusively for the NNU community yet. While popular websites, programs, and applications have a lot to offer, a website specifically for NNU students to buy and sell books to each other provides many more advantages.

The significant advantages that the NNU website offers are convenience, safety, reliability, and efficiency. The convenience of buying/selling books to others in the NNU community is substantial. Buyers or sellers do not have to go out of their way to deliver the

purchased book. Instead, they can meet in locations on campus. Keeping transactions within the community almost guarantees most users have connections in some way or another due to the small school size. Typically, if other websites are used to organize an in-person interaction to sell a book, a neutral location is chosen by both parties to meet and make the transaction. However, this would not be the case as both parties would be from NNU. Therefore, there is an additional sense of safety.

Additionally, there would be a significantly high, almost guaranteed, buyback rate. The sellers are students who have purchased books for classes they have taken, and other students, assuming that the professor does not change the course materials, will undoubtedly have to take that same class and require the same textbook. As there is an extremely high likelihood that a textbook can easily be sold once it has been finished, purchasing one would be much more comfortable for students because they would know that a portion of the money will be recovered. This also makes the option to buy a textbook, rather than rent, a lot more desirable. Moreover, students would receive their books immediately rather than wait for shipping when they order. The added stress of returning the textbook at the end of the semester will also be eliminated if the textbook is bought rather than rented.

With all of these advantages in mind, creating a website for NNU students to buy and sell books became the adopted idea for the project. This decision resulted in taking the first step of action, seeking the help of the NNU webmaster, Zac Vineyard. Zac provided guidance on where to begin, what language to use to develop the project, and the tools to use.

Development Environment

Laravel

Help from mentor, Zac Vineyard, was sought because he was responsible for developing and deploying NNU's websites. The goal was for the book-selling website to use the same environment and development plan as the other NNU websites. Zac suggested using Laravel as a framework to provide a solid foundation. With his previous experience using Sublime Text to develop websites, he also suggested that as the text editor to host the development of the website. Sequel Pro and DBngin were also used by reason of suggestion from the mentor.

Laravel was used to develop this project, along with several other technologies. Laravel is an open-source PHP (Hypertext preprocessor). A framework is essentially the starting code to build a project. By providing structure and a starting point, the framework enables developers to focus on creating excellent applications instead of worrying about the details (Installation - Meet Laravel, n.d.). It offers simple characteristics that help developers write organized, self-documenting code (Multani, 2018). Due to running on PHP, it is entirely server-side and focuses substantially on data manipulation and adhering to an MVC (Model View Controller) architecture. Laravel was used to save time and labor because it does not require having to build the whole project from scratch. However, it ended up needing a lot of additional overhead due to a lack of experience with PHP and the framework.

Unlike other frameworks, the Laravel framework is better suited to develop this project, as the website will require both front-end and back-end support to operate effectively. Furthermore, Laravel's built-in user authentication, online documentation covering the full-stack development process, and other advantages make it superior to other full-stack frameworks. The Laravel framework is an expressive framework suitable for large-scale projects.

Sublime Text

Sublime Text was used as an IDE (Integrated Development Environment) to host the project. Sublime Text supports major programming languages, including, but not limited to, the languages used to develop this project: PHP, HTML, CSS, and JavaScript. PHP was used to communicate between the website and the database on the back-end. HTML and CSS were used to interact with the browser to build the website's foundation on the front-end. JavaScript was also used on the front end to style the website. With Sublime Text, the browser can access the IDE, so web applications can be developed and debugged much faster than they would in a local environment (Hoissan, p. 11).

Sequel Pro and DBngin

The MySQL database for this project was created and managed with DBngin and Sequel Pro. DBngin is an all-in-one version management application that helps set up and manage the local database server (DBngin: Free all-in-one database version management tool). Sequel Pro is a database management application and one of the best GUI (Graphical User Interface) tools for MySQL. These applications worked together by DBngin using MySQL binaries to set up the local server so that a database could be created and managed on Sequel Pro and linked to the project in Laravel.

Before starting the project, these applications were recommended on account of functionality and usability with Laravel on a macOS platform. Both applications were developed for and supported exclusively by macOS, the platform on which this project was developed. Once implemented into the project in Laravel, all of these suggestions proved to be accurate; therefore, no other applications were considered.

Requirements

In the beginning, many requirements for the website were established to provide guidance for the criteria needed to develop the project. However, as the project's development progressed, these requirements consistently changed. The initial requirements will be exhibited and discussed in the following section.

Website Requirements

Before beginning any development on the project, the website's goals and requirements were established. Manageable and attainable project size in the time parameters given for this project were considered. Therefore, the website's goal was to provide a means for NNU students to buy and sell their used textbooks.

A user and login system became an imperative requirement to distinguish users and allow them to access and participate in the website's services. In order to be able to implement a user/login system, a database was required to store the user information. Providing users with the ability to create a post for a book was also a requirement. This post needed to contain details of the book, including title, author, ISBN, description, and a picture of the cover. The post also needed to include information on the user, including their name and email address. A function to upload and store an image needed to be developed to permit a picture in the post. A dashboard to list and allow users to view all of the postings that have also been created proved to be a requirement. When more students use the website, and there is a large amount of information available, the implementation of a search feature is crucial to ensure that students can find the books they are looking for. Also deemed essential was the ability to edit and delete a post.

Database Requirements

Database requirements were relatively simple because of Laravel's pre-build migrations and non-extensive data conditions. These pre-built migrations are "failed_jobs," "job_batches," "migrations," "password_resets," "personal_access_tokens," "sessions," and "users," and they provide basic user authentication and registration functionality. Also available are custom migrations. Only one table to hold book information was required. Therefore, only one custom migration was required for a table to store book information, titled "books." The fields that the books table was required to contain were title, author, ISBN, subject, description, price, and the path of the cover image.

Implementation

Getting Started

Artisan

Artisan is the command-line interface that is included with Laravel. The artisan command aims to help reduce common and repetitive tasks. Therefore, the artisan command helps decrease the development time and complexity of Laravel projects (Hoissan, p. 10). The artisan commands were used to set up and develop the laravel website.

Composer

The first step to running the Laravel application was to install a Composer program, a package manager for PHP. By installing composer, all of the libraries that were needed to create and run the project were included. To install Composer, a series of commands were run in the terminal (Figure 1). Installation of Composer was necessary to set up Laravel.

Composer Setup

```
Audreess-MacBook-Pro:~ audreemontmeny$ php composer-setup.php
All settings correct for using Composer
Downloading...

Composer (version 2.1.11) successfully installed to: /Users/audreemontmeny/composer.phar
Use it: php composer.phar

Audreess-MacBook-Pro:~ audreemontmeny$ php -r "unlink('composer-setup.php');"sudo mv composer.phar /usr/local/bin/composer

Parse error: syntax error, unexpected end of file in Command line code on line 1
Audreess-MacBook-Pro:~ audreemontmeny$ sudo mv composer.phar /usr/local/bin/composer
Password:
Audreess-MacBook-Pro:~ audreemontmeny$ composer

          _____
         /  ____  /
        /  /  __/
       /  /  /  /
      /  /  /  /
     /  /  /  /
    /  /  /  /
   /  /  /  /
  /  /  /  /
 /  /  /  /
/  /  /  /

Composer version 2.1.11 2021-11-02 12:10:25
```

The next step was to create the project. The command “composer create-project laravel/laravel example-app” was executed in the terminal, as seen in Figure 2. Following, an application key was generated. Generating an application key was accomplished by executing the command “php artisan key: generate” in the command line (Figure 3). After executing this command, the application was viewable in the browser.

Creating the Project

```
~/example-app — php • php artisan serve
```

```
why-not          Shows which packages prevent the given package from being installed.
Audrees-MacBook-Pro:~ audreemontmeny$ composer create-project laravel/laravel example-app
Creating a "laravel/laravel" project at "./example-app"
Installing laravel/laravel (v8.6.5)
```

Generating an Application Key

```

Publishing complete.
> @php artisan key:generate --ansi
Application key set successfully.
Audrees-MacBook-Pro:~ audreemontmeny$ cd example-app/
Audrees-MacBook-Pro:example-app audreemontmeny$ php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Fri Nov 5 13:22:43 2021] 127.0.0.1:63051 [200]: /favicon.ico
Environment modified. Restarting server...

```

Model

Laravel uses MVC architecture to separate application logic into three main things that make up an application: models, views, and controllers. Models are effectively used as data stores and provide a simple implementation for working with the database. Each database table has a corresponding "Model" which is used to interact with that table. These models can be related by using relationships just as would be typical with a standard SQL database (Row, 2018).

A Book model was created with the command "php artisan make:model Book" to interact with the database described below. The Book model provided the ability to query for the data in the books table, as well as insert new records into the table.

View

Views output data on the screen via controller or route. This is usually HTML (hypertext markup language), but views specifically separate the controller and application logic from the presentation logic. However, Laravel comes with a templating engine called Blade. Blade templates are compiled into PHP code and cached until modified, so it has virtually no overhead (Views, n.d.). Each page of the website was created with the template blade engine.

Controller

Controllers handle incoming HTTP (hypertext transfer protocol) and send a response. They direct traffic between models and views. Groups of logic related to handling requests can be contained in one controller class. It handles the requests coming from the Routes (Hoissan, p. 7).

A resource controller was created to handle the routes between the Book model and the views in the project with the command "php artisan make:controller BookController." The

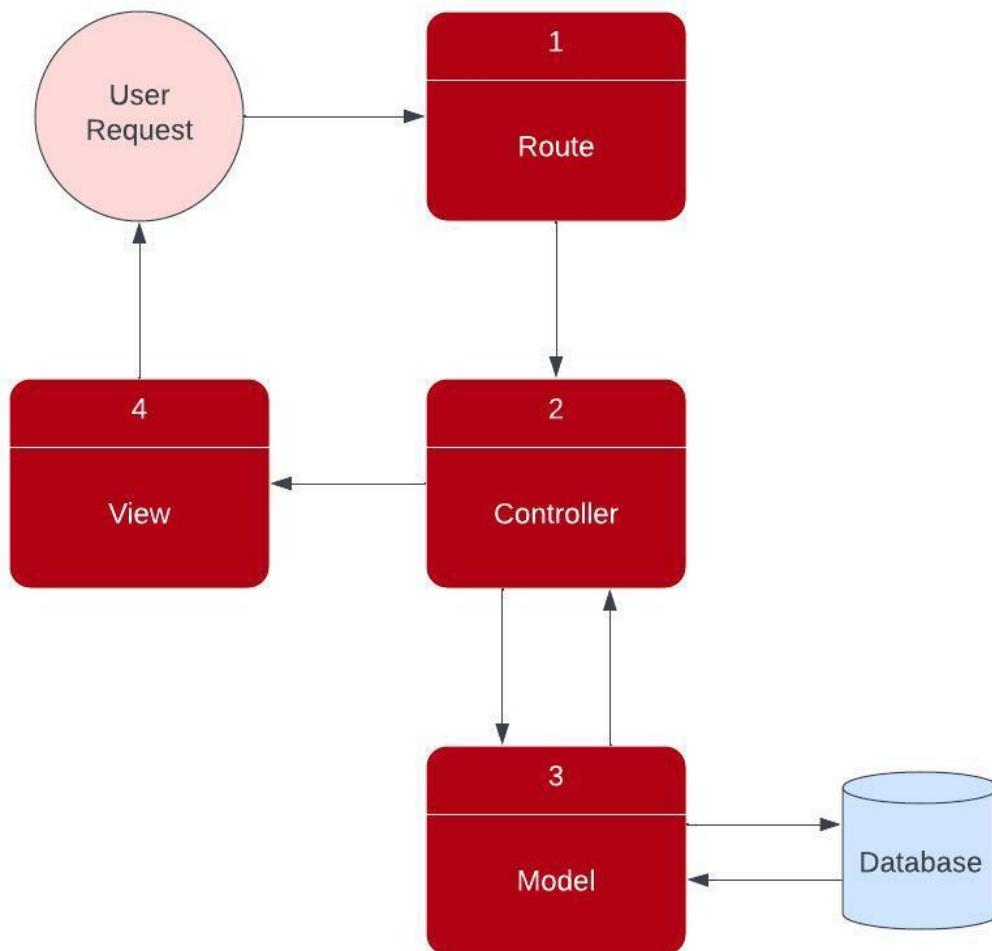
BookController was used to perform the CRUD (create, read, update, delete) operations for the route in the controller for the Books model to create the functions of the website.

Route

A route is necessary to use the controller and allowed the BookController to be put into action. When the user hits the browser request it goes to a route first then it finds the controller attached to the route. A diagram was created to exhibit the relationships between the route, controller, model, and view and how they work together in the project (Figure 4).

Figure 4

Relationship Diagram

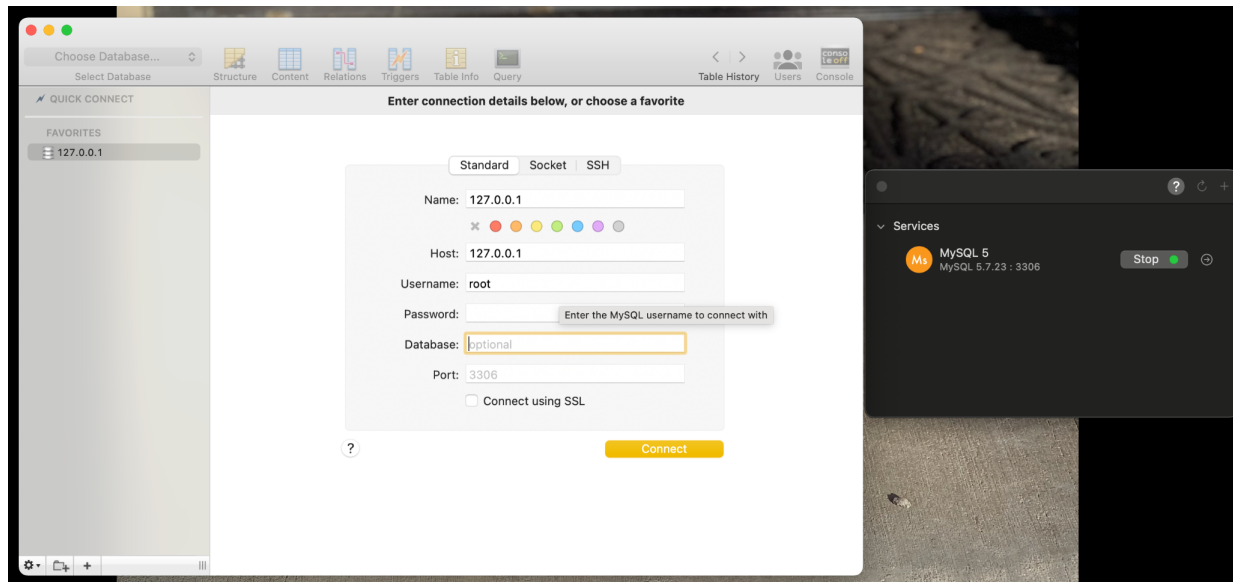


Database Implementation

Setting up the database was very simple once both the Sequel Pro and DBngin applications were installed. Generating an application key created a .env file. The .env file specifies the environment of the project. All that needed to be done to set up the database was to enter the username and host server number into the .env file shown in. Once this step was complete, the database was successfully connected to the project. A picture of the Sequel Pro (left) and DBngin (right) can be seen in Figure 5, with the username it came with and the host information. The command "php artisan migrate" was executed in the terminal. This command migrated the tables that were already in the project after creating it due to the framework of Laravel. Once the database was implemented, a local server was required to test the project. The project can be run through a local development server by executing the command "php artisan serve" in the project's terminal. Once this was complete, the website could run and be tested. While this is a great way to develop and test a website on a local machine, it is not a complete web server meant to host a project. This means that the website cannot be accessed by any other means than on the local machine with the webserver.

Figure 5

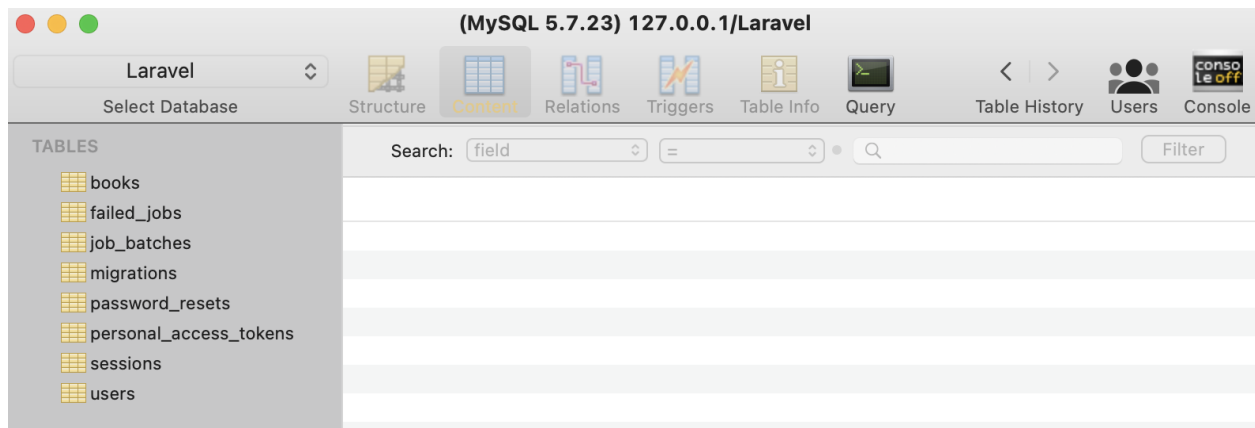
Startup Database Screen



Migrations were built-in when the project was created with Laravel, as well as the ability to make custom migrations. It comes with six migrations: "failed_jobs," "job_batches," "migrations," "password_resets," "personal_access_tokens," "sessions," and "users ." The command "php artisan migrate" was executed to create these tables. The only custom migration that needs to be run is a migration for the "books" table to store all of the book data. The command "php artisan make:migration create_books_table" was executed in the project's terminal to create this table. The "php artisan migrate" command migrated all of these fundamental tables into the Sequel Pro system used with the website. A visual of these tables in the Sequel Pro database system is shown in Figure 6.

Figure 6

Tables in Sequel Pro Database Application



By running the “php artisan make:migration create_books_table” command, a migration file was also created in the project. With the Artisan command, the database was created through the “up” method of the file. The attributes required to be stored in the books table were added to the up method. The attributes that were added to the method were “title,” “author,” “isbn,” “subject,” “description,” “price,” “inputemail,” “inputname,” and “price.” Once migrated again, these attributes were added to the books table. Code for this method can be found in Appendix B under create_books_table.php.

Additionally, Laravel uses Eloquent, an ORM (object-relational mapper), for interaction with the database (Laravel). A books model was created to interact with the books table using Eloquent. The command “php artisan make:model Books” was executed to create this model. The “Books” model also provided a means to insert, update, and delete data from the books table.

Once development was complete, and the website was employable, the database functioned correctly and stored all necessary data in the appropriate tables. A couple of people were asked to register on the website to test the user authentication and log-in functionality and have data in the database. A picture of the user table after multiple people registered on the website is shown in Figure 7. Using the function to create a book listing on the website, multiple

books were created to test the function and allow data in the books table to be present. The resulting books table, which stores all of the enterable data on each book, is shown in Figures 8 and 9.

Figure 7

User Table in Database

id	name	email	email_verified_at	password	remember_token	created_at	updated_at
1	Audree Montmeny	amontmeny@nnu.edu	NULL	\$2y\$10\$3jYw7Z5abxHmi1/6FbZ...	NULL	2022-03-29 21:41:13	2022-03-29 21:41:13
2	Ashley Bliss	ashleybliss@nnu.edu	NULL	\$2y\$10\$1aWlqvl.cCitAKVvPxXdGO...	NULL	2022-03-30 04:33:08	2022-03-30 04:33:08
3	Dominique Trevino	dtrevino@nnu.edu	NULL	\$2y\$10\$Cbw5XRuei9jJHth8Yelmd...	NULL	2022-03-31 17:42:50	2022-03-31 17:42:50
4	Emma Mulligan	emulligan@nnu.edu	NULL	\$2y\$10\$11GMrdXA.mo8Oj1Bd8g...	NULL	2022-03-31 17:43:44	2022-03-31 17:43:44
5	sidney booth	sidneybooth@nnu.edu	NULL	\$2y\$10\$5wao/ZZscYTVBedzN3ZPlr...	NULL	2022-03-31 17:44:40	2022-03-31 17:44:40
6	kellcie adams	kellcieadams@nnu.edu	NULL	\$2y\$10\$mYOKZndR3mUSCCmPTx...	NULL	2022-03-31 20:36:42	2022-03-31 20:36:42
7	Rheanna Russo	Rrusso@nnu.edu	NULL	\$2y\$10\$OT6A0QDVaY8I6gOjNSLf...	NULL	2022-04-02 00:53:36	2022-04-02 00:53:36
8	Emilie Hernandez	emiliehernandez@nnu.edu	NULL	\$2y\$10\$5sTrjutlQW9Qng6j/bGUdf...	NULL	2022-04-04 22:21:09	2022-04-04 22:21:09
9	Brittany Genuardi	bgenuardi@nnu.edu	NULL	\$2y\$10\$XNZtSGObbWmoAr854DK...	NULL	2022-04-06 02:21:28	2022-04-06 02:21:28

Figure 8

Books Table in Database (Pt. 1)

id	title	author	isbn	subject	description	price	inputemail	inputname	image
33	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
34	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
35	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
36	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
37	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
38	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
39	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
40	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
41	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
42	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
43	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
44	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
45	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
46	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
47	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
48	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
49	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
50	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
51	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
52	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
53	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
54	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
55	Test Book	Test Autho	1234567890	Cats	Everything you need to know abou...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
56	Criminal Procedure	Matthew Lippman	978-1544334752	Criminal Justice	NULL	2	ashleybliss@nnu.edu	Ashley Bliss	1331c392dee
58	Invitation to Computer Science	G. Michael Schneider	9781337561914	Computer Science	My first college textbook!	30	amontmeny@nnu.edu	Audree Montmeny	eb2701080c4
59	BRS Cell Biology and Histology	Leslie P. Gartner	1496396359	Biology	This textbook is for the human bi...	45	emulligan@nnu.edu	Emma Mulligan	69e62ead6ee

Books Table in Database (Pt. 2)

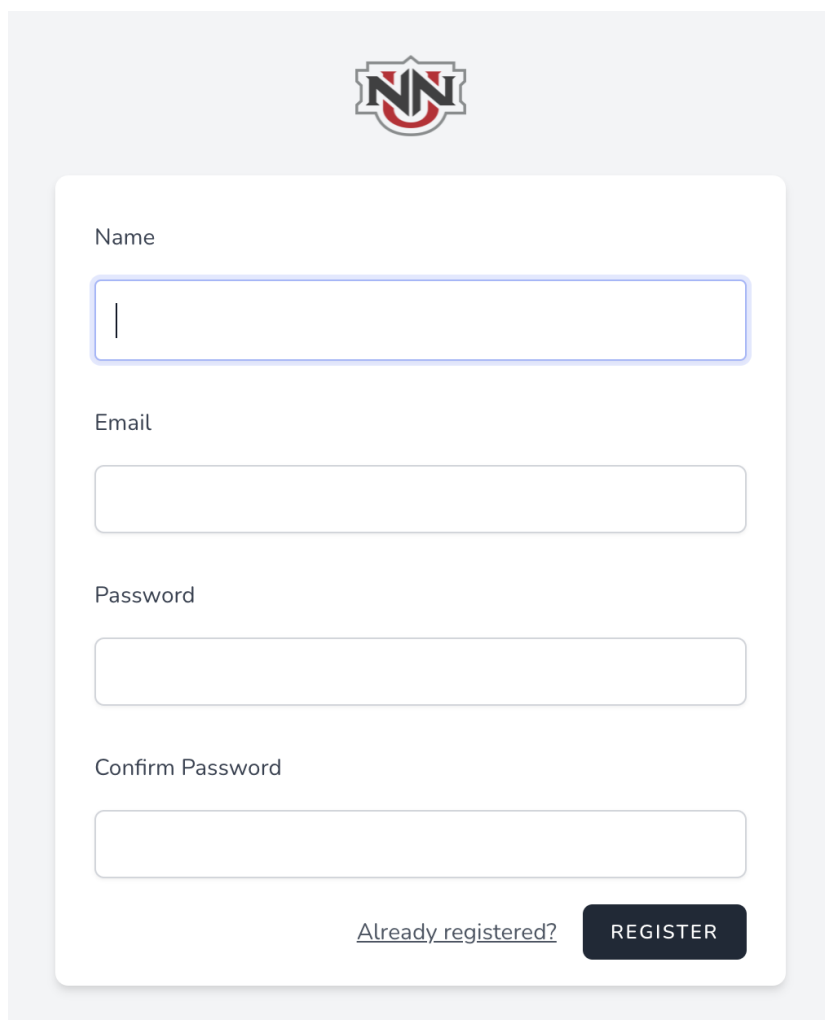
14

Website Implementation

As previously mentioned, the Laravel project came with a user authentication system, which included user registration and login functionality. A password recovery function was also included to help registered users recover their passwords if forgotten. A picture of the registration function can be seen in Figure 10, and a picture of the login function can be seen in Figure 11. However, some CSS was applied to change the style. As shown in the picture, the login function can save the user's information to make it easier for future login.

Figure 10

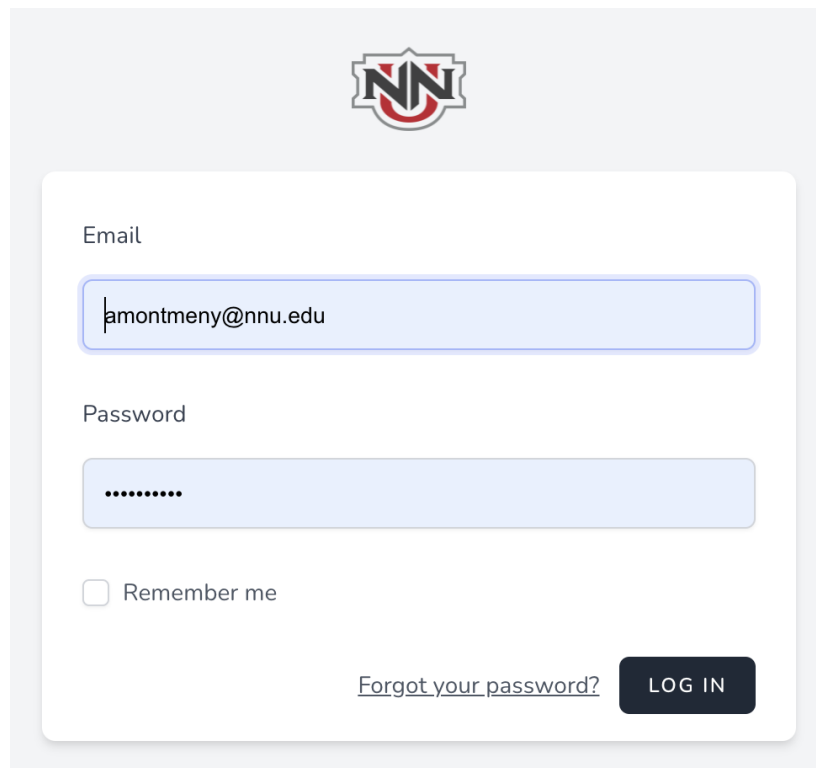
User Registration Form



The image shows a user registration form with a light gray background. At the top center is a logo consisting of a shield with the letters 'NN' inside. Below the logo, the form is a white rounded rectangle with a subtle shadow. It contains four input fields: 'Name' (with a vertical cursor), 'Email', 'Password', and 'Confirm Password'. At the bottom left of the form is a link that says 'Already registered?'. At the bottom right is a dark blue button with the word 'REGISTER' in white capital letters.

Figure 11

User Login Form

The image shows a user login form on a light gray background. At the top center is a logo consisting of a shield with the letters 'NN' inside. Below the logo is a white rectangular form with rounded corners. Inside the form, there are two input fields: the first is labeled 'Email' and contains the text 'amontmeny@nnu.edu'; the second is labeled 'Password' and contains a series of dots. Below the password field is a checkbox labeled 'Remember me'. At the bottom of the form, there is a link that says 'Forgot your password?' and a dark blue button with the text 'LOG IN' in white capital letters.

In order to begin development on the website, a controller was created for books. The BookController was created through the command “php artisan make:controller BookController.” This BookController organized all related request handling logic into one class rather than defining all related request handling logic as closures in the routes file. The BookController also contained all of the methods for the CRUD (create, read, update, delete) operations for books. However, the appropriate route had to be defined and requested in the web.php routes file for these operations to run. The code for the routes can be found under web.php in Appendix B.

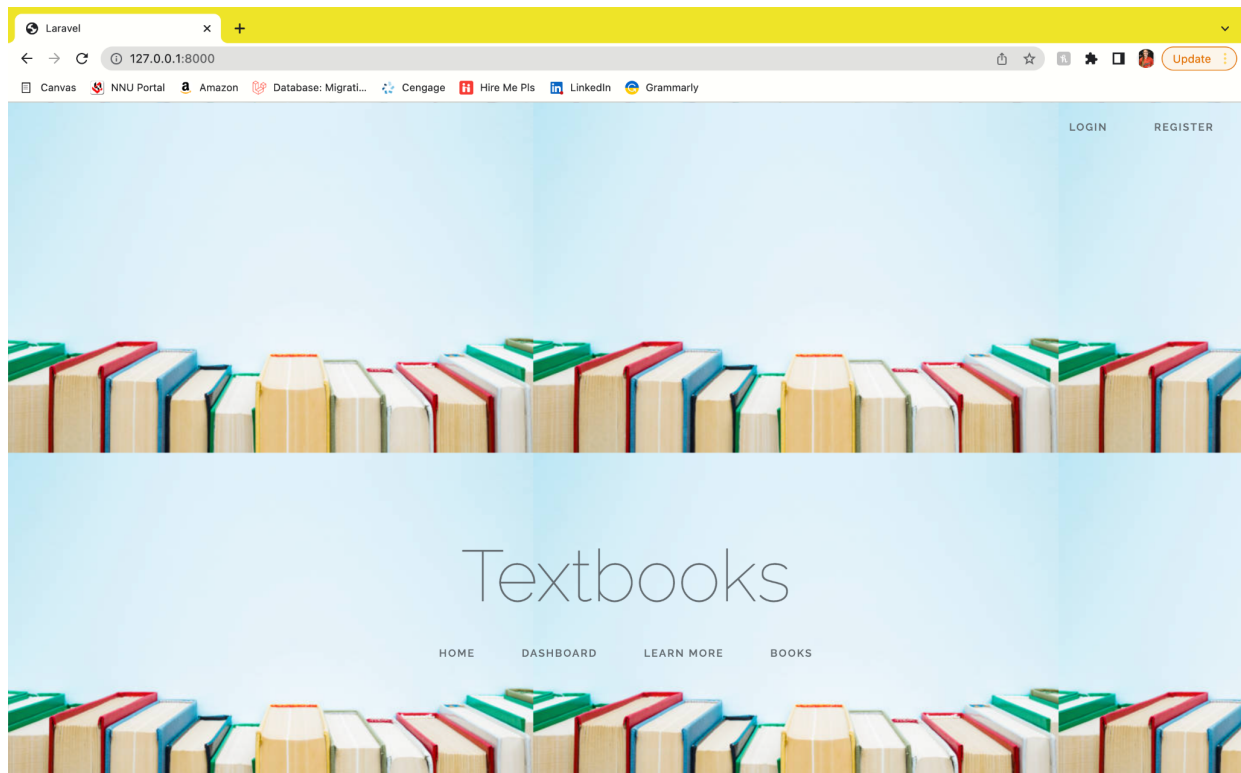
Welcome.blade.php

A basic welcome page was shipped with the project from Laravel. However, CSS was implemented to customize this welcome page toward a book theme. The purpose of the page is

to provide users with a startup page where they can log in or register for the website. In the top right of the page is a “Log in” or “Register” button for the user. If the user is already logged in, a “Home” button redirects the user to the home page. A picture of the page can be seen in Figure 12, and the code for the page can be seen under `welcome.blade.php` in Appendix B.

Figure 12

Welcome.blade.php

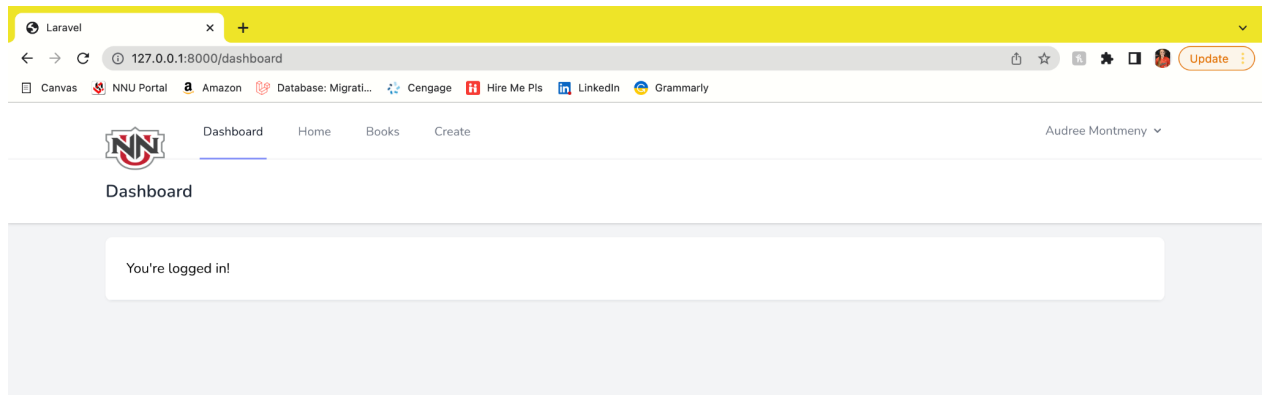


Dashboard.blade.php

Upon logging in from the welcome page, the user is directed to the dashboard page, where he or she receives a notification that they have successfully logged in. A picture of this page and the navigation bar are displayed in Figure 13, and the code can be viewed under `dashboard.blade.php` in Appendix B.

Figure 13

Dashboard.blade.php

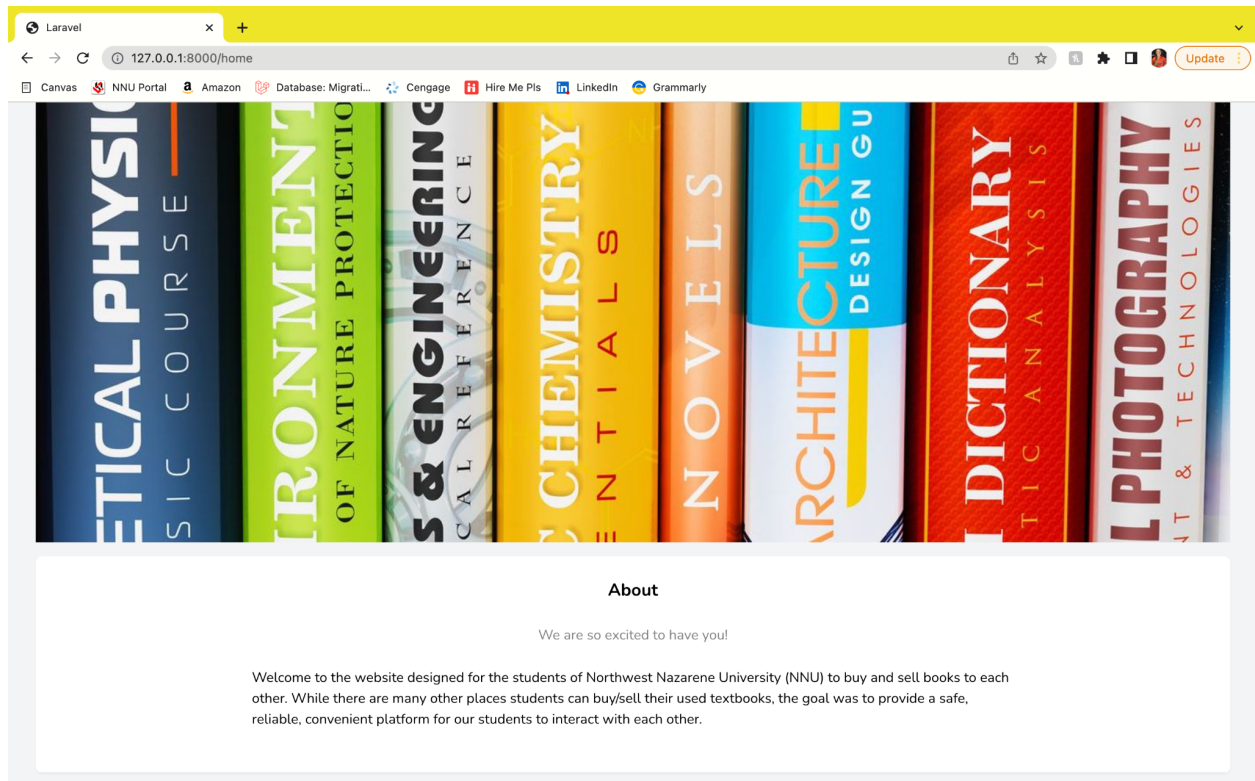


Home.blade.php

A simple home page was also created to provide background and an introduction to the website. Due to limited time, the home page was not the main priority. Therefore, minimal information and design were included, as shown in Figure 14. Code for this page can be seen under `home.blade.php` in Appendix B. Not pictured in Figure 14 is the top of the book image, header, and navigation bar on every page.

Figure 14

Home.blade.php



Create.blade.php

The create page is the page used to create book listings that users wish to sell. On this page, there is a form for the user to fill out with the fields: "Name," "Email," "Title," "Author," "ISBN," "Subject," "Description," "Price," and "Cover Image" (Figure 15). The "Cover Image" field is a function that allows the user to upload a picture of the textbook's cover. This is not a field to be filled out but rather to select and upload an image from the computer. Once all the fields of the form have been filled out, and the user clicks the "Submit" button, all of the information is stored in the books table of the database.

Figure 15

Create Book Listing Form

The screenshot shows a web application interface for creating a book listing. At the top, there is a navigation bar with a logo on the left and links for 'Dashboard', 'Home', 'Books', and 'Create' in the center. The 'Create' link is highlighted with a blue underline. On the right side of the navigation bar, the user's name 'Audree Montmeny' is displayed with a dropdown arrow. Below the navigation bar, the word 'Create' is written in a larger font. The main content area is a form with several input fields: 'Name' (with a placeholder 'First and Last'), 'Email' (with a placeholder 'Email'), 'Title' (with a placeholder 'Title'), 'Author' (with a placeholder 'Author'), 'ISBN' (with a placeholder 'ISBN'), 'Subject' (with a placeholder 'Subject'), and 'Description' (a larger text area with a placeholder '(Optional) Description of book'). Below these fields is a 'Price' field with a placeholder '\$'. At the bottom of the form, there is a 'Cover Image' section with a 'Choose File' button and the text 'No file chosen'. A 'Submit' button is located at the very bottom of the form.

The book information is saved to the database via the store function in the BookController.php file. A picture of this function is shown in Figure 16; however, the complete code of the file can be found under BookController.php in Appendix B. Once the book is saved

to the database, the user is redirected to the page that contains a view of all the book listings with a success message at the top that reads, “You did it!”.

Figure 16

Store Function to Save Books to Database

```
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required',
        'author' => 'required',
        'isbn' => 'required', //|integer|min:0|max:
        'subject' => '',
        'description' => '',
        'price' => '',
        'inputname' => '',
        'inputemail' => ''
    ]);

    // Create Book
    $book = new Book;
    $book->title = $request->title;
    $book->author = $request->author;
    $book->isbn = $request->isbn;
    $book->subject = $request->subject;
    $book->description = $request->description;
    $book->price = $request->price;
    $book->inputname = $request->inputname;
    $book->inputemail = $request->inputemail;

    if($request->file('image')) //function to upload an image
    {
        $file = $request->file('image');
        $extension = $file->getClientOriginalExtension();
        $filename = md5($file->getClientOriginalName() . time()).'.'.$extension;
        $file->move('images', $filename);
        $book->image = $filename;
    }

    $book->save(); //saves all information to books table

    return redirect('books')->with('success', 'You did it!');
}
```

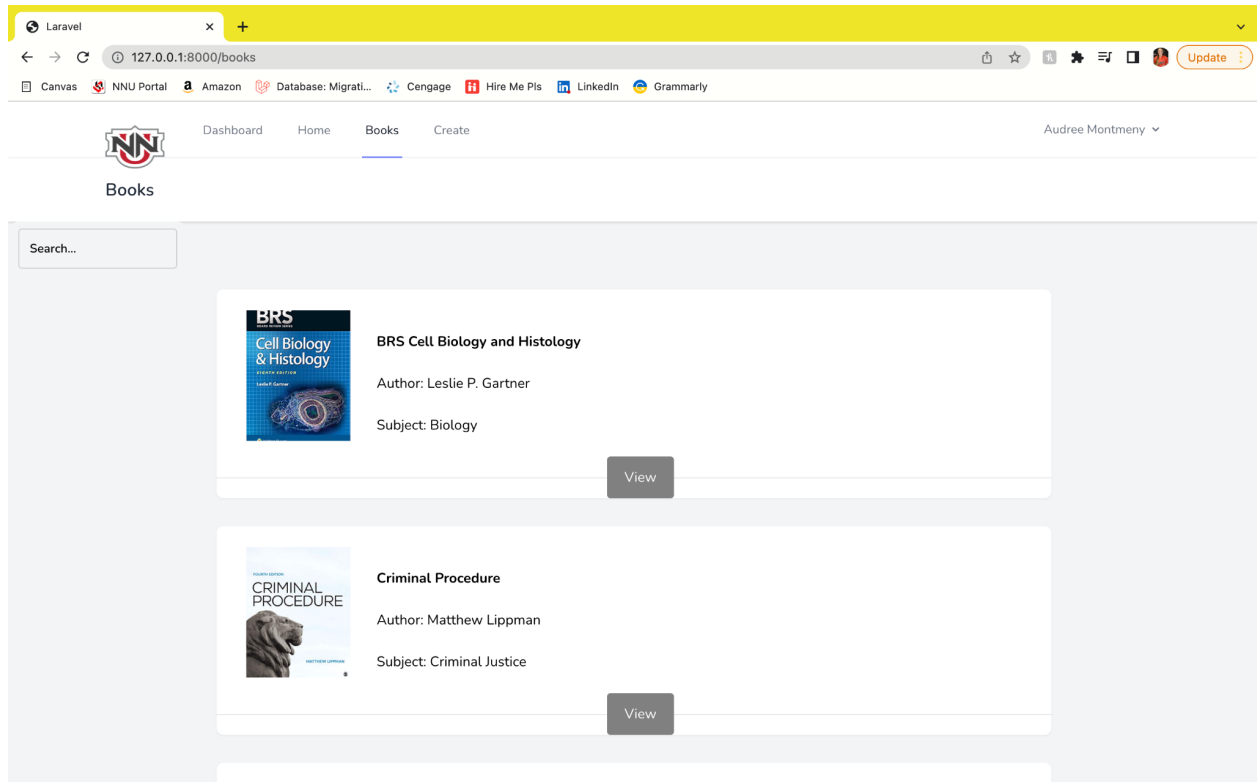
Index.blade.php

The index page displays all of the book postings that have been created in ascending order by title (Figure 17). Each posting is displayed separately with a few primary attributes,

including the cover image, title, author, and subject. Additionally, there is a button on the post with the word “view.” If this button is clicked, the user is directed to the show.blade.php page, where more information is displayed.

Figure 17

Book Posting List View



A search function is also included in the top left corner to make it easier for users to find specific books. The function takes the input text and searches the title of every book post that includes that text (Figure 18). Then, it displays only the book postings that include the text in ascending order by title. Furthermore, to make viewing book postings more ideal, the posts are organized into pages with ten postings per page using the “paginate(10)” function. This function can be seen when the user reaches the bottom of the page. The option to navigate through the pages is also given (Figure 19). A picture of this function in the BookController is shown in Figure 20.

Figure 18

Search Function

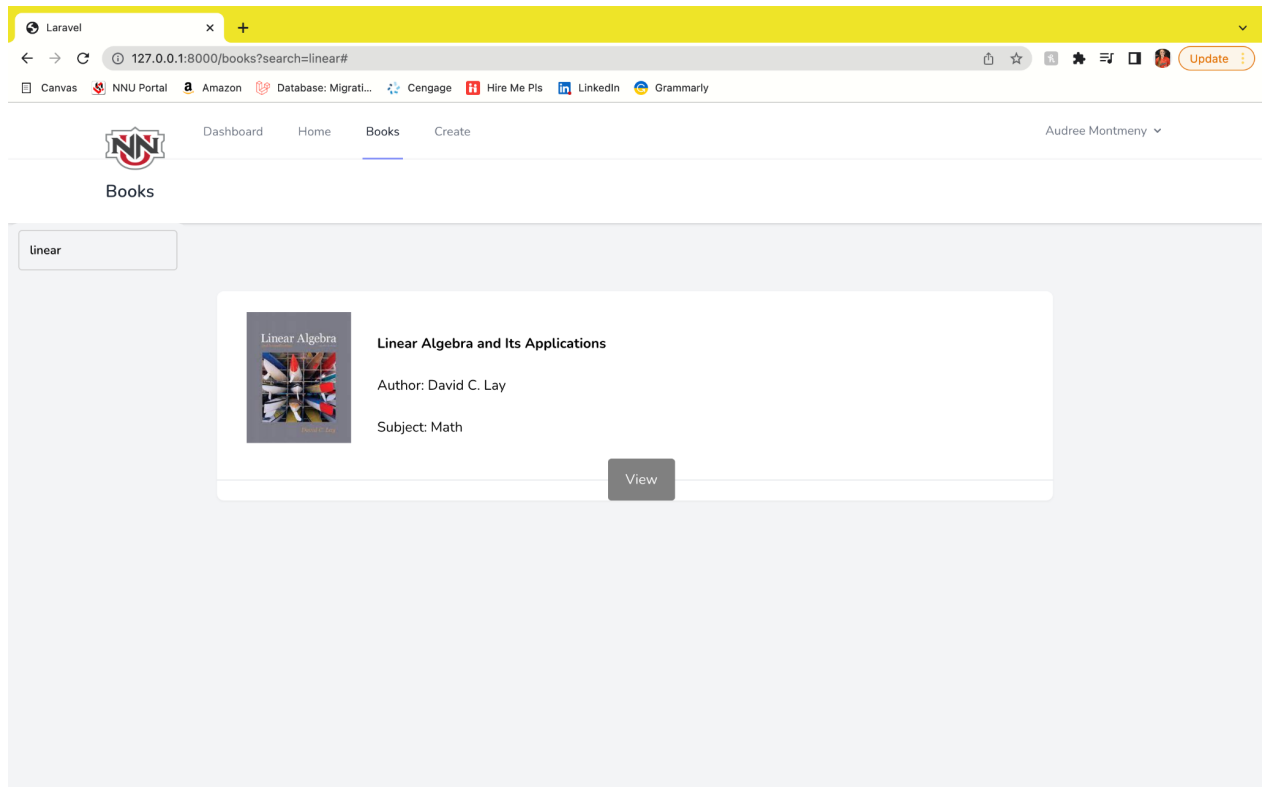


Figure 19

Page View

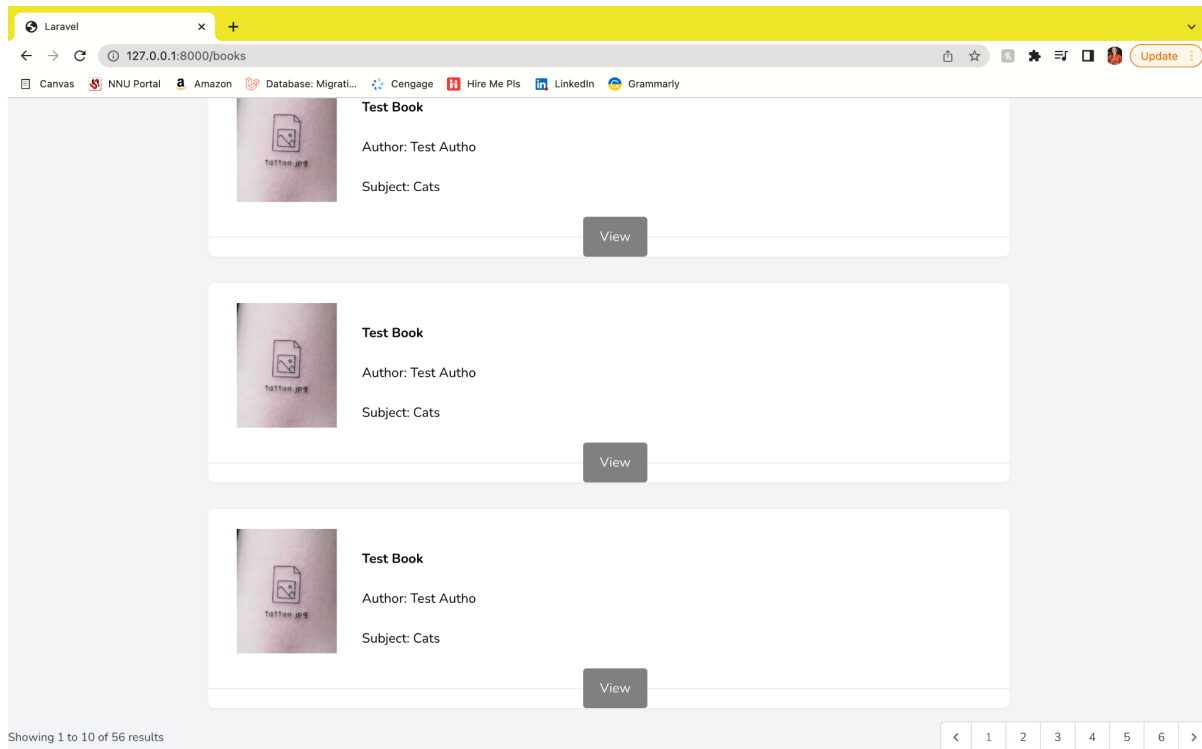


Figure 20

Index Function in BookController

```
public function index(Request $request)
{
    $books = Book::orderBy('title','asc');

    if (request('search')) {
        $books->where('title','like','%'.request('search').'%');
    }

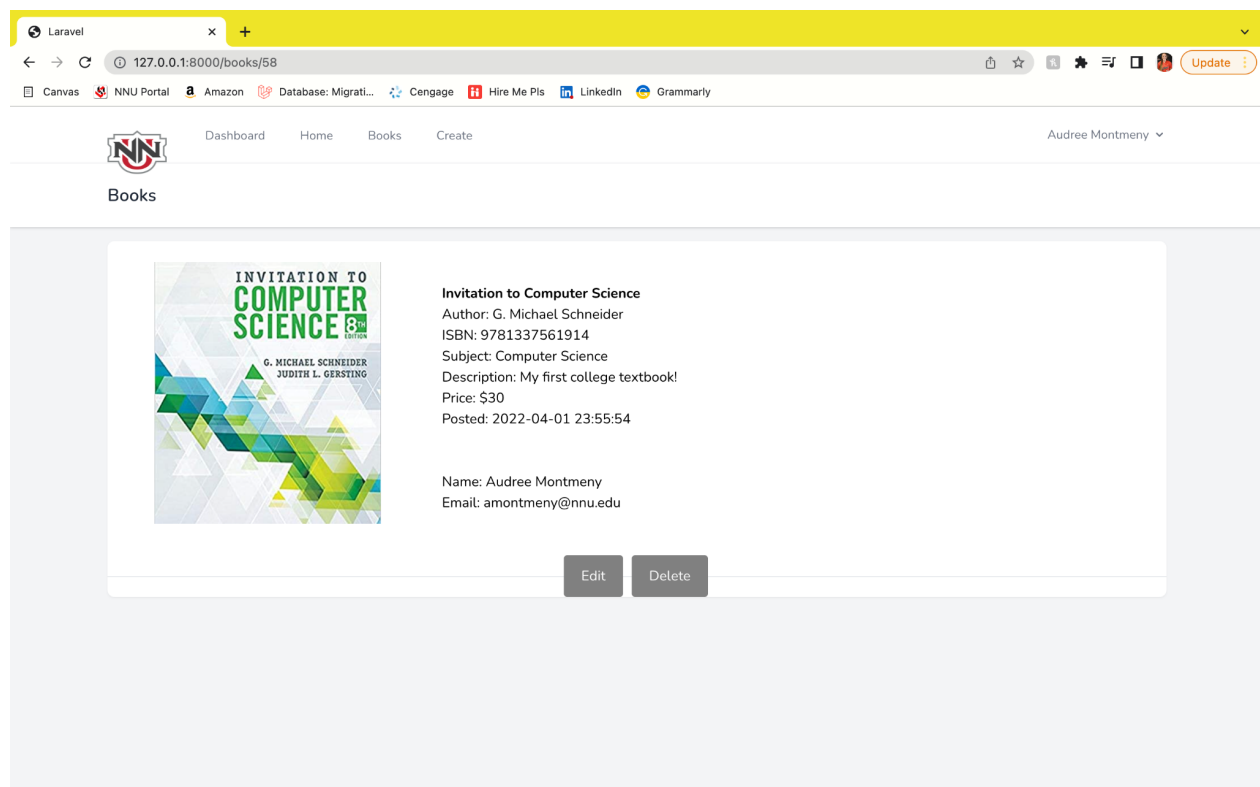
    return view('books/index', [
        'books' => $books->orderBy('title','asc')->paginate(10),
    ]);
}
```

Show.blade.php

As mentioned earlier, the show.blade.php page is displayed when the user clicks the button to view more details from the list of book postings found on the index.blade.php page. More information on the book is viewable on this page. The rest of the user's attributes assigned to the book by filling out the "create" form are displayed (Figure 21), along with buttons to edit or delete the book posting. Clicking on the "Edit" button navigates the user to the edit.blade.php. Clicking the "delete" button deletes the book post from the website and database, so it will no longer be present in the list of book postings on the index.blade.php page.

Figure 21

Show.blade.php View of Book Details




Edit.blade.php

The edit page allows the user to make changes in their book listing. The user is directed to the edit.blade.php page when clicking the “Edit” button on the show.blade.php page. As shown in Figure 22, the page contains the same form used to create a book listing. However, all of the information previously entered into the create form and saved in the database is already present in each field. This is done by retrieving all of the data from the database in the update function (Figure 23). Once the user makes the desired changes, the “submit” button is clicked, and the new information is saved to the database. The user is then redirected to the index page with a message saying, “It has been updated!”.

Figure 22

Edit Book Post Form

 [Dashboard](#) [Home](#) [Books](#) [Create](#) Audree Montmeny ▾

Books

Edit Post

Title

Invitation to Computer Science

Author

G. Michael Schneider

ISBN

9781337561914

Subject

Computer Science

Description

My first college textbook!

Price

30

Cover Image No file chosen

Figure 23

Update Function in BookController

```
public function update(Request $request, $id)
{
    //dd($request);
    $validated = $request->validate([
        'title' => 'required',
        'author' => 'required',
        'isbn' => 'required',
        'subject' => '',
        'description' => '',
        'price' => '',
        'inputname' => '',
        'inputemail' => ''
    ]);

    $book = Book::where('id', $id)->get();
    // $book = Book::find($id);
    $book->title = $request->title;
    $book->author = $request->author;
    $book->isbn = $request->isbn;
    $book->subject = $request->subject;
    $book->description = $request->description;
    $book->price = $request->price;
    $book->inputname = $request->inputname;
    $book->inputemail = $request->inputname;

    return redirect('books')->with('success', 'It has been updated!');
}
```

Conclusion

One of the most challenging aspects that included many roadblocks was the use of the Laravel framework. Ideally, the framework should save time and energy in developing tedious functions and features. However, there was no previous experience with the framework and minimal experience with PHP. As mentioned previously, using Laravel required a lot of extra overhead that proved to be unideal. Additionally, a considerable obstacle that was faced was the relocation of mentor, Zach Vineyard. After Christmas break, he took a different job and was no longer with NNU. This posed a significant challenge because it was harder to overcome issues

and confusion faced during the project. He was no longer easily accessible to meet with and help with the project.

A substantial amount of time and energy was spent understanding the framework and working through issues faced in the process. Another struggle was little experience with the PHP language. Essentially, it was like jumping into the deep end without first knowing how to swim.

While it was challenging, it was also gratifying. Many roadblocks were encountered due to a lack of knowledge and were very difficult to overcome. However, most of the roadblocks were overcome by spending a significant amount of time going through videos and tutorials. The hard work to overcome these obstacles made the success more rewarding. The skills and knowledge that were learned from using a framework such as Laravel were well worth all of the challenges that were faced.

Future Work

While much work has been done to create a website for NNU students to buy and sell books to each other, more work is still required for the website to be fully serviceable. While some of these features cannot be implemented until more progress has been made, others were left out because they were at the bottom of the requirements list and time was no longer available to complete them.

Currently, every user has access to all the information and functions of the website, meaning anyone can view, edit, and delete each post. This is a problem because the data needs to be protected and only allowed to be accessed by the creator, so others do not go in and edit or delete posts that are not theirs. This feature was not implemented because of the lack of development time. A feature like this can be created by only authorizing access to the edit and delete functions to the user ID that created the post.

Another thing to be addressed is the hosting of the website. Currently, the website is hosted on the local server of the machine on which the website was developed. However, the website is not accessible to anyone else. For students to access and use the website, the site and database would need to be deployed on NNU's servers.

While an automatic email service was already shipped with the Laravel framework, the website would need to be hosted on NNU's server for the feature to be operational. When a new user creates an account, the ability to create a password is functional. However, the user is not notified that they have created an account and been given access to verify their email or can change their password. While all of these features are already available, the website needs to be hosted on NNU's server for them to be operational.

While not necessary, it would be ideal for the website to contain more features and functions, so no other resources are needed to carry out buying or selling a book. Currently, the website is only functional for users to create, edit, and view posts. Contact information is provided in the post. The only way purchasing or selling a book can be carried out is for users to contact each other via email outside of the website. The only way for users to accomplish payment transactions is outside the website via cash, Venmo, PayPal, etc. Ideally, messaging capabilities and payment processing functionality would be implemented on the website. Because these features are more complicated to implement, they were not at the top of the requirements list and were deemed beyond the scope of the project.

References

Edureka. (2018). *Laravel Tutorial For Beginners | What Is Laravel? | Laravel Training Part - 1 |*

Edureka [Video]. YouTube. <https://www.youtube.com/watch?v=bkyjiXSx6WE>.

Hossain, S. (2019). Web Application Development with Laravel Framework. *Turku University of Applied Sciences*, 34.

Laravel. (n.d.). *Installation—Meet Laravel*. Retrieved April 13, 2022, from <https://laravel.com/docs/8.x/installation#why-laravel>

Laravel. (n.d.). *Views*. Retrieved April 13, 2022, from <https://laravel.com/docs/8.x/views>

Multani, A. (2018, May 23). *Getting started with Laravel. Open Source for You*.

<https://www.proquest.com/docview/2048462597/citation/5283422C21C1421DPQ/7>

U.S. Department of Education, National Center for Education Statistics, Integrated Postsecondary Education Data System (IPEDS). (2021, November). *Digest of Education Statistics. National Center for Education Statistics (NCES)*. Retrieved March 28, 2022, from https://nces.ed.gov/programs/digest/d21/tables/dt21_330.40.asp

Appendix A - Screenshots

Composer Setup

```
Audrees-MacBook-Pro:~ audreemontmeny$ php composer-setup.php
All settings correct for using Composer
Downloading...

Composer (version 2.1.11) successfully installed to: /Users/audreemontmeny/composer.phar
Use it: php composer.phar

Audrees-MacBook-Pro:~ audreemontmeny$ php -r "unlink('composer-setup.php');"sudo mv composer.phar /usr/local/bin/composer
Parse error: syntax error, unexpected end of file in Command line code on line 1
Audrees-MacBook-Pro:~ audreemontmeny$ sudo mv composer.phar /usr/local/bin/composer
Password:
Audrees-MacBook-Pro:~ audreemontmeny$ composer

  _____
 / _____ \
( (       ) \
 \_____/_____/
  _____

Composer version 2.1.11 2021-11-02 12:10:25
```

Creating the Project

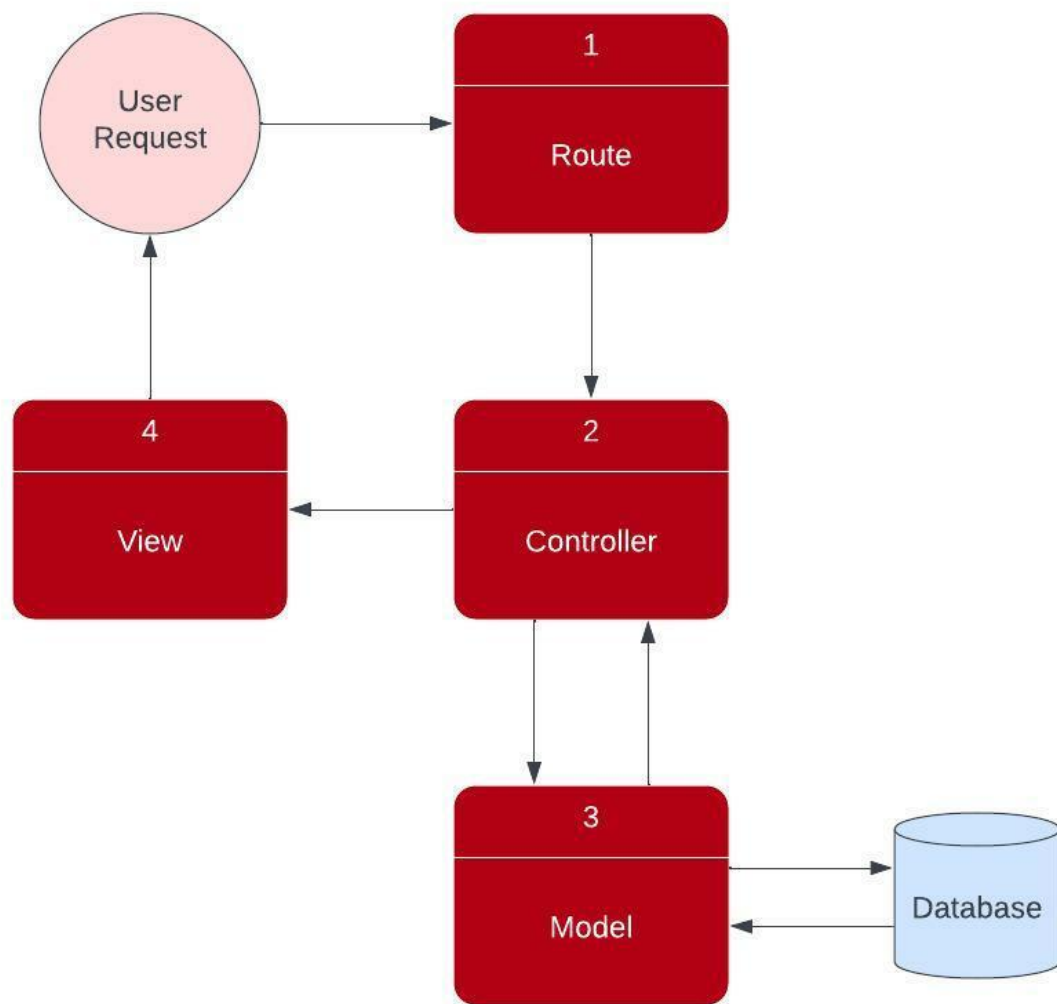
```
~/example-app — php ◀ php artisan serve

why-not          Shows which packages prevent the given package from being installed.
Audrees-MacBook-Pro:~ audreemontmeny$ composer create-project laravel/laravel example-app
Creating a "laravel/laravel" project at "./example-app"
Installing laravel/laravel (v8.6.5)
```

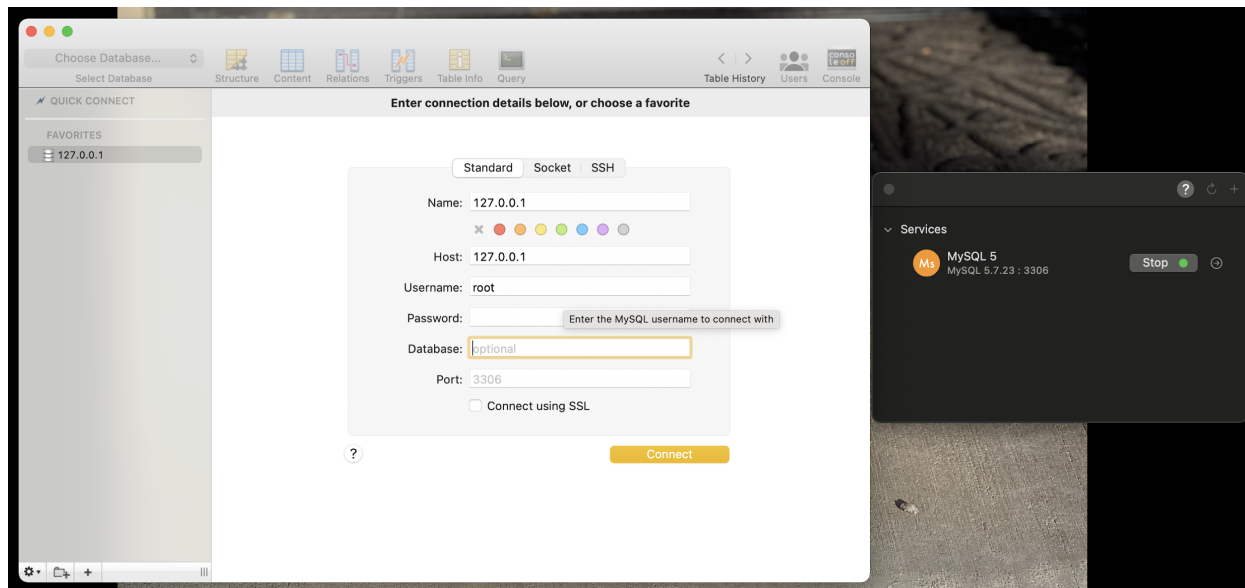
Generating an Application Key

```
publishing completed.
> @php artisan key:generate --ansi
Application key set successfully.
Audrees-MacBook-Pro:~ audreemontmeny$ cd example-app/
Audrees-MacBook-Pro:example-app audreemontmeny$ php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Fri Nov 5 13:22:43 2021] 127.0.0.1:63051 [200]: /favicon.ico
Environment modified. Restarting server...
```

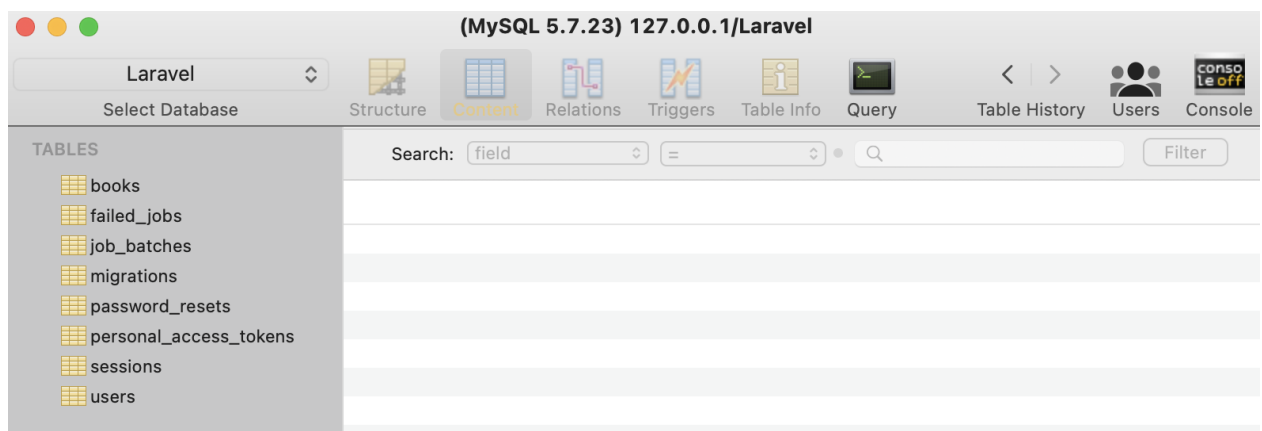
Relationship Diagram



Startup Database Screen



Tables in Sequel Pro Database Application



User Table in Database

MySQL 5.7.23) 127.0.0.1/Laravel/users

Select Database Structure Content Relations Triggers Table Info Query Table History Users Console

Search: id

id	name	email	email_verified_at	password	remember_token	created_at	updated_at
1	Audree Montmeny	amontmeny@nnu.edu	NULL	\$2y\$10\$3jYw7ZSaxHm1l/6FbZ...	NULL	2022-03-29 21:41:13	2022-03-29 21:41:13
2	Ashley Bliss	ashleybliss@nnu.edu	NULL	\$2y\$10\$1aWqvl.cCitAKVvPxXdGO...	NULL	2022-03-30 04:33:08	2022-03-30 04:33:08
3	Dominique Trevino	dtrevino@nnu.edu	NULL	\$2y\$10\$CbwSXRuei9/JHth8Yelmd...	NULL	2022-03-31 17:42:50	2022-03-31 17:42:50
4	Emma Mulligan	emulligan@nnu.edu	NULL	\$2y\$10\$11GMrDXA.mo8Qj18d8g...	NULL	2022-03-31 17:43:44	2022-03-31 17:43:44
5	sidney booth	sidneybooth@nnu.edu	NULL	\$2y\$10\$wao/ZZscYTV8edzN3ZPlr...	NULL	2022-03-31 17:44:40	2022-03-31 17:44:40
6	kelcie adams	kelcieadams@nnu.edu	NULL	\$2y\$10\$mY0KZndR3mUSCCmPTx...	NULL	2022-03-31 20:36:42	2022-03-31 20:36:42
7	Rheanna Russo	Rrusso@nnu.edu	NULL	\$2y\$10\$0T6A0QDVaY8I6gOJNSLf...	NULL	2022-04-02 00:53:36	2022-04-02 00:53:36
8	Emilie Hernandez	emiliehernandez@nnu.edu	NULL	\$2y\$10\$5sTrjutlQW9Qng6j/bGUdf...	NULL	2022-04-04 22:21:09	2022-04-04 22:21:09
9	Brittany Genuardi	bgenuardi@nnu.edu	NULL	\$2y\$10\$XNZtSOGbbWmoAr854DK...	NULL	2022-04-06 02:21:28	2022-04-06 02:21:28

TABLE INFORMATION

- created: 3/29/22
- updated: 4/5/22
- engine: InnoDB
- rows: 9
- size: 16.0 KiB
- encoding: utf8mb4
- auto_increment: 10

9 rows in table; 1 row selected

Books Table in Database (Pt. 1)

MySQL 5.7.23) 127.0.0.1/Laravel/books

Select Database Structure Content Relations Triggers Table Info Query Table History Users Console

Search: id

id	title	author	isbn	subject	description	price	inputemail	inputname	image
33	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
34	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
35	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
36	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
37	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
38	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
39	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
40	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
41	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
42	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
43	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
44	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
45	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
46	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
47	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
48	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
49	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
50	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
51	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
52	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
53	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
54	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
55	Test Book	Test Autho	1234567890	Cats	Everything you need to know about...	10	lonlycatlady@gmail.com	Joan	1d695d730ce
56	Criminal Procedure	Matthew Lippman	978-1544334752	Criminal Justice	NULL	2	ashleybliss@nnu.edu	Ashley Bliss	1331c392dee
58	Invitation to Computer Science	G. Michael Schneider	9781337561914	Computer Science	My first college textbook!	30	amontmeny@nnu.edu	Audree Montmeny	eb2701080c4
59	BRS Cell Biology and Histology	Leslie P. Gartner	1496396359	Biology	This textbook is for the human bi...	45	emulligan@nnu.edu	Emma Mulligan	69e62ead6ee

TABLE INFORMATION

- created: 3/29/22
- updated: 4/3/22
- engine: InnoDB
- rows: 55
- size: 16.0 KiB
- encoding: utf8mb4
- auto_increment: 60

55 rows in table; 1 row selected

Books Table in Database (Pt. 2)

[illegible]

User Registration Form



Name

Email

Password

Confirm Password

[Already registered?](#)

REGISTER

User Login Form



Email

amontmeny@nnu.edu

Password

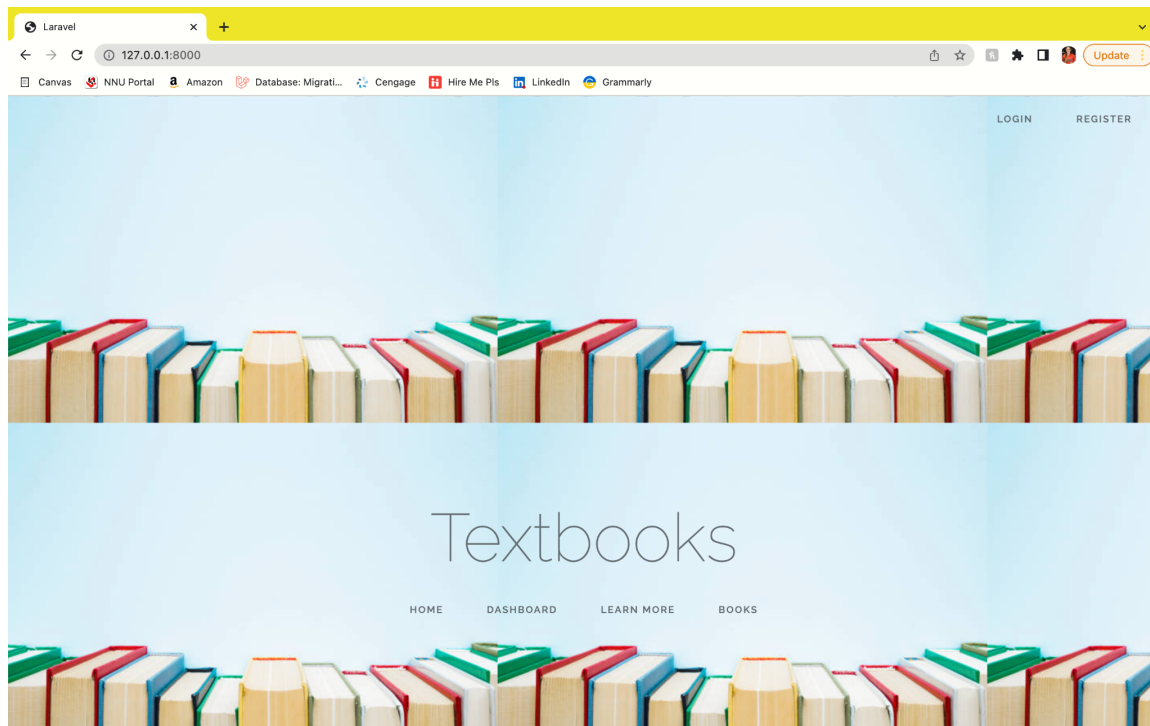
.....

☐ Remember me

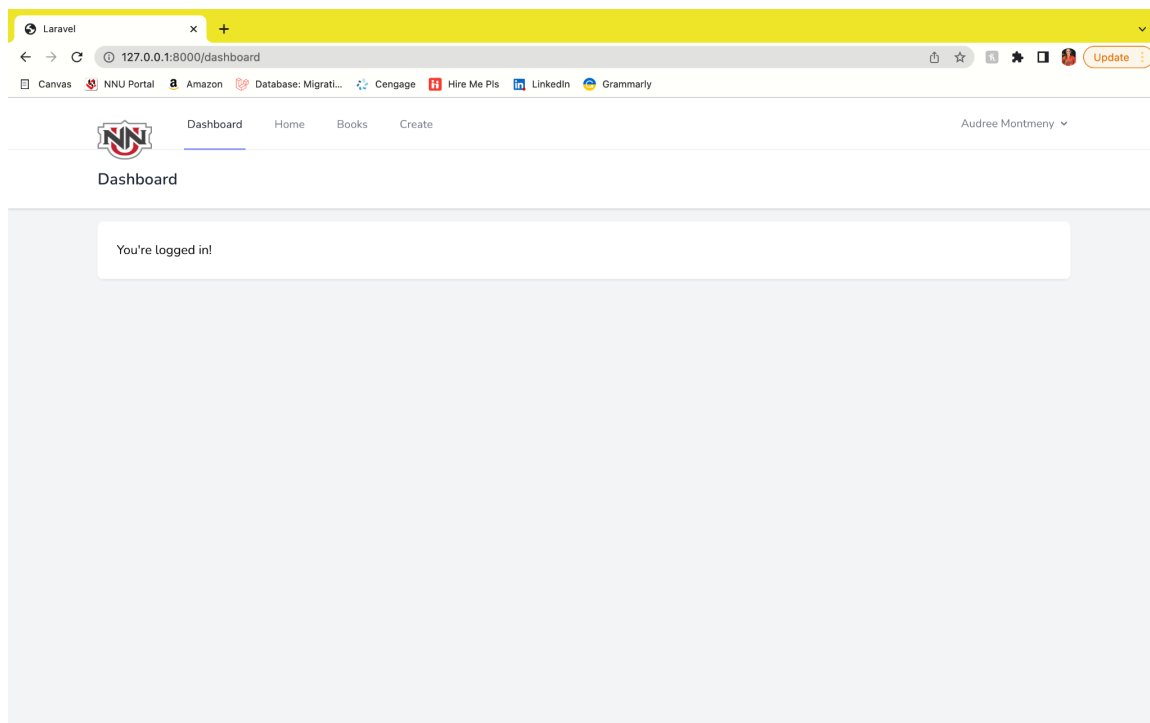
[Forgot your password?](#)

LOG IN

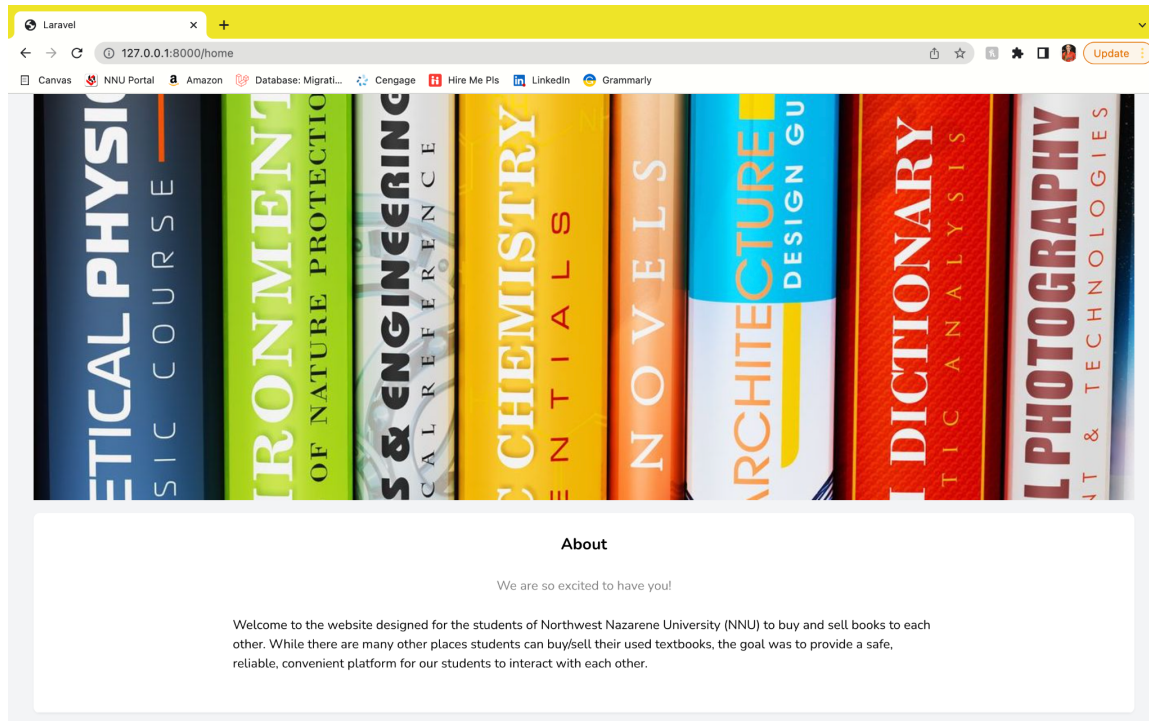
Welcome.blade.php




Dashboard.blade.php



Home.blade.php



Create Book Listing Form



DashboardHomeBooksCreate

Audree Montmeny ▾

Create

Name

First and Last

Email

Email

Title

Title

Author

Author

ISBN

ISBN

Subject

Subject

Description

(Optional) Description of book

Price

\$

Cover Image

Choose File

No file chosen

Submit

Store Function to Save Books to Database

```
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required',
        'author' => 'required',
        'isbn' => 'required', //|integer|min:0|max:
        'subject' => '',
        'description' => '',
        'price' => '',
        'inputname' => '',
        'inputemail' => ''
    ]);

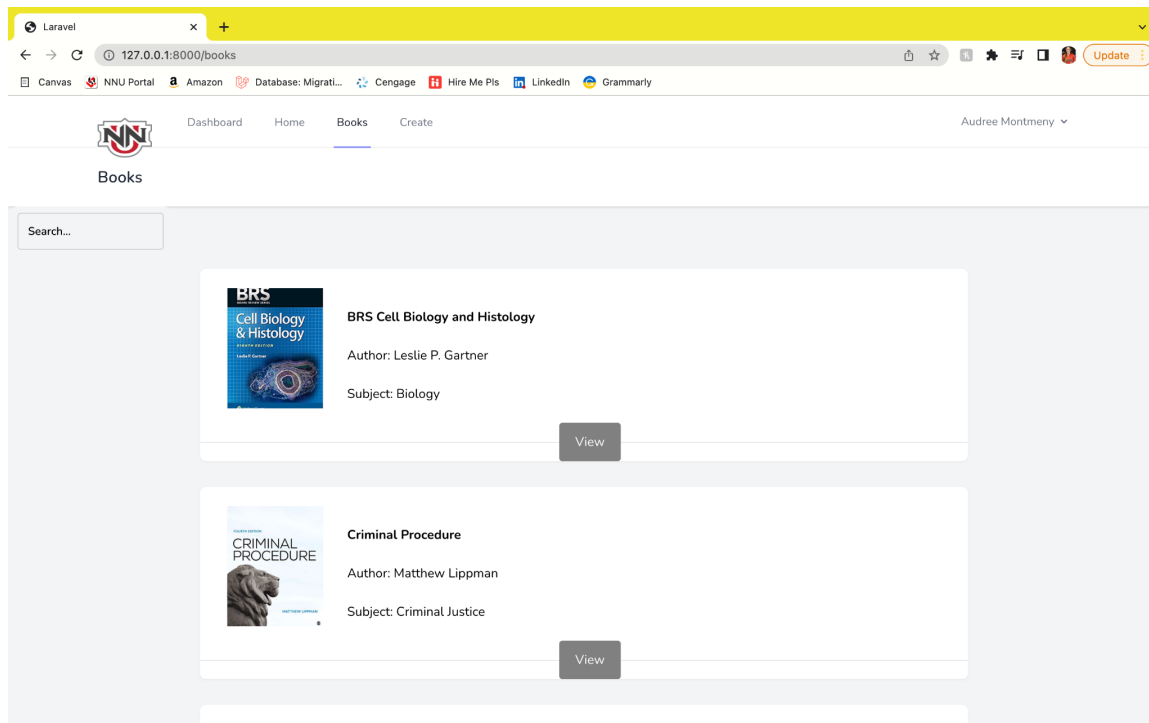
    // Create Book
    $book = new Book;
    $book->title = $request->title;
    $book->author = $request->author;
    $book->isbn = $request->isbn;
    $book->subject = $request->subject;
    $book->description = $request->description;
    $book->price = $request->price;
    $book->inputname = $request->inputname;
    $book->inputemail = $request->inputemail;

    if($request->file('image')) //function to upload an image
    {
        $file = $request->file('image');
        $extension = $file->getClientOriginalExtension();
        $filename = md5($file->getClientOriginalName() . time()).'.'.$extension;
        $file->move('images', $filename);
        $book->image = $filename;
    }

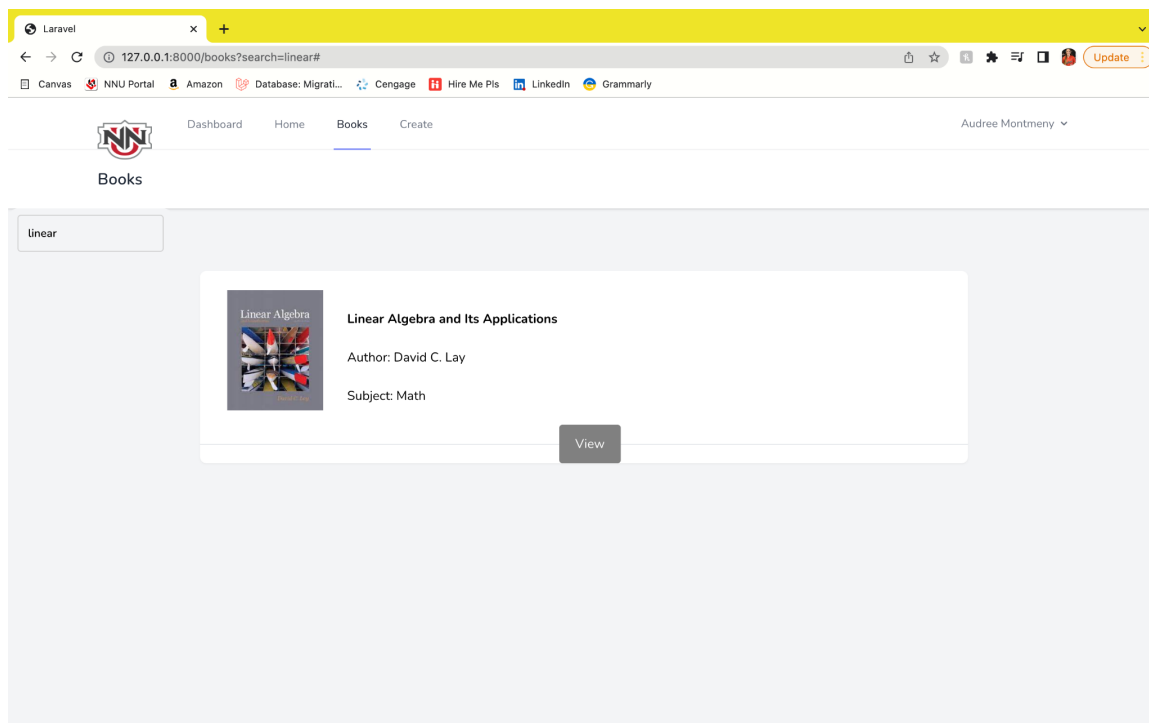
    $book->save(); //saves all information to books table

    return redirect('books')->with('success', 'You did it!');
}
```

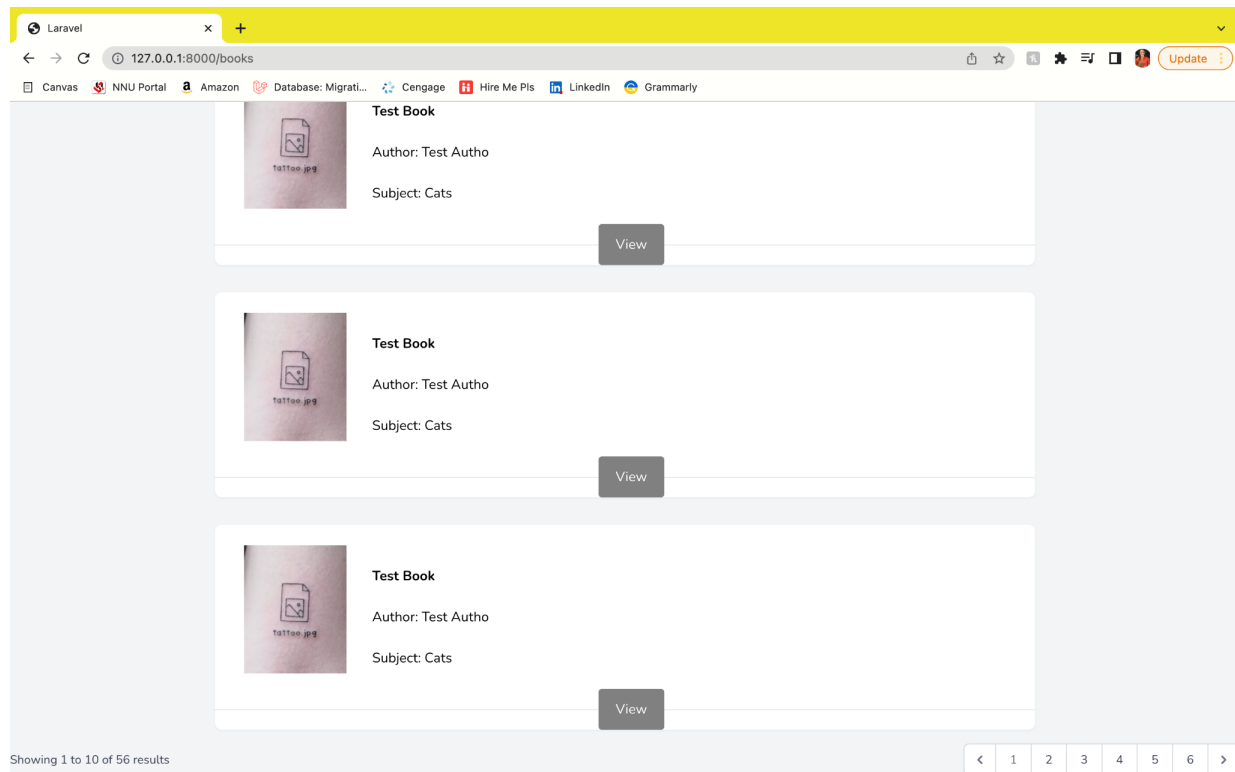
Book Posting List View



Search Function



Page View



Index Function in BookController

```
public function index(Request $request)
{
    $books = Book::orderBy('title','asc');

    if (request('search')) {
        $books->where('title','like','%'.request('search').'%');
    }

    return view('books/index', [
        'books' => $books->orderBy('title','asc')->paginate(10),
    ]);
}
```

Show.blade.php View of Book Details

Laravel

127.0.0.1:8000/books/58

Update

Canvas

NNU Portal

Amazon


Database: Migrati...

Cengage

Hire Me Pls

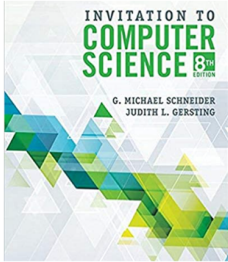
LinkedIn

Grammarly

 Dashboard Home Books Create

Audree Montmeny

Books




Invitation to Computer Science

Author: G. Michael Schneider
ISBN: 9781337561914
Subject: Computer Science
Description: My first college textbook!
Price: \$30
Posted: 2022-04-01 23:55:54

Name: Audree Montmeny
Email: amontmeny@nnu.edu

EditDelete

Edit Book Post Form

 [Dashboard](#) [Home](#) [Books](#) [Create](#) Audree Montmeny ▾

Books

Edit Post

Title

Invitation to Computer Science

Author

G. Michael Schneider

ISBN

9781337561914

Subject

Computer Science

Description

My first college textbook!

Price

30

Cover Image No file chosen

Update Function in BookController

```
public function update(Request $request, $id)
{
    //dd($request);
    $validated = $request->validate([
        'title' => 'required',
        'author' => 'required',
        'isbn' => 'required',
        'subject' => '',
        'description' => '',
        'price' => '',
        'inputname' => '',
        'inputemail' => ''
    ]);

    $book = Book::where('id', $id)->get();
    // $book = Book::find($id);
    $book->title = $request->title;
    $book->author = $request->author;
    $book->isbn = $request->isbn;
    $book->subject = $request->subject;
    $book->description = $request->description;
    $book->price = $request->price;
    $book->inputname = $request->inputname;
    $book->inputemail = $request->inputname;

    return redirect('books')->with('success', 'It has been updated!');
}
```

Appendix B - Code

Navigation Bar

A navigation bar was already provided with the Laravel framework. The navigation bar included a layout to add desired pages. There was a settings dropdown in the top right corner with the name of the user who is logged in. The dropdown arrow reveals the option to log out. In the end, the navigation bar had Home, Dashboard, Books, and Create tabs. The navigation bar has a very simple design and structure.

navigation.blade.php

```
<nav x-data="{ open: false }" class="bg-white border-b
border-gray-100">

    <!-- Primary Navigation Menu -->

    <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

        <div class="flex justify-between h-16">

            <div class="flex">

                <!-- Logo -->

                <div class="flex-shrink-0 flex items-center">

                    <a href="{{ route('dashboard') }}">

                        <x-application-logo class="block h-10 w-auto
fill-current text-gray-600" />

                    </a>

                </div>

            </div>

        </div>

    </div>
```

```

<!-- Navigation Links -->

<div class="hidden space-x-8 sm:-my-px sm:ml-10
sm:flex">

    <x-nav-link :href="route('dashboard')"
:active="request()->routeIs('dashboard')">

        {{ __('Dashboard') }}

    </x-nav-link>

    <x-nav-link :href="route('home')"
:active="request()->routeIs('home')">

        {{ __('Home') }}

    </x-nav-link>

    <x-nav-link :href="route('books.index')"
:active="request()->routeIs('books.index')">

        {{ __('Books') }}

    </x-nav-link>

    <x-nav-link :href="route('books.create')"
:active="request()->routeIs('books.create')">

        {{ __('Create') }}

    </x-nav-link>

</div>

</div>

<!-- Settings Dropdown -->

<div class="hidden sm:flex sm:items-center sm:ml-6">

```

```

<x-dropdown align="right" width="48">
    <x-slot name="trigger">
        <button class="flex items-center text-sm
font-medium text-gray-500 hover:text-gray-700 hover:border-gray-300
focus:outline-none focus:text-gray-700 focus:border-gray-300
transition duration-150 ease-in-out">
            <div>{{ Auth::user()->name }}</div>

            <div class="ml-1">
                <svg class="fill-current h-4 w-4"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20">
                    <path fill-rule="evenodd"
d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414
1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
clip-rule="evenodd" />
                </svg>
            </div>
        </button>
    </x-slot>

    <x-slot name="content">
        <!-- Authentication -->
        <form method="POST" action="{{
route('logout') }}">
            @csrf

```

```

        <x-dropdown-link :href="route('logout')"
            onclick="event.preventDefault();

this.closest('form').submit();">

            {{ __('Log Out') }}
        </x-dropdown-link>
    </form>
</x-slot>
</x-dropdown>
</div>

<!-- Hamburger -->
<div class="-mr-2 flex items-center sm:hidden">
    <button @click="open = ! open" class="inline-flex
items-center justify-center p-2 rounded-md text-gray-400
hover:text-gray-500 hover:bg-gray-100 focus:outline-none
focus:bg-gray-100 focus:text-gray-500 transition duration-150
ease-in-out">

        <svg class="h-6 w-6" stroke="currentColor"
fill="none" viewBox="0 0 24 24">
            <path :class="{ 'hidden': open, 'inline-flex':
! open }" class="inline-flex" stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16"
/>

```

```

        <path :class="{ 'hidden': ! open,
'inline-flex': open }" class="hidden" stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6L12 12" />

        </svg>

        </button>

    </div>

</div>

</div>

<!-- Responsive Navigation Menu -->

<div :class="{ 'block': open, 'hidden': ! open}" class="hidden
sm:hidden">

    <div class="pt-2 pb-3 space-y-1">

        <x-responsive-nav-link :href="route('home')"
:active="request()->routeIs('home')">

            {{ __('Home') }}

        </x-responsive-nav-link>

        <x-responsive-nav-link :href="route('dashboard')"
:active="request()->routeIs('dashboard')">

            {{ __('Dashboard') }}

        </x-responsive-nav-link>

        <x-responsive-nav-link :href="route('books.index')"
:active="request()->routeIs('books.index')">

            {{ __('Books') }}

```

```

        </x-responsive-nav-link>

        <x-responsive-nav-link :href="route('books.create')"
:active="request()->routeIs('books.create')">
            {{ __('Create') }}
        </x-responsive-nav-link>
    </div>

    <!-- Responsive Settings Options -->
    <div class="pt-4 pb-1 border-t border-gray-200">
        <div class="px-4">
            <div class="font-medium text-base text-gray-800">{{
Auth::user()->name }}</div>
            <div class="font-medium text-sm text-gray-500">{{
Auth::user()->email }}</div>
        </div>

        <div class="mt-3 space-y-1">
            <!-- Authentication -->
            <form method="POST" action="{{ route('logout') }}">
                @csrf

                <x-responsive-nav-link :href="route('logout')"
                    onclick="event.preventDefault();
this.closest('form').submit();">

```

```

                {{ __('Log Out') }}
            </x-responsive-nav-link>

        </form>

    </div>

</div>

</div>

</nav>

```

welcome.blade.php

```

<!doctype html>

<html lang="{{ config('app.locale') }}">

    <head>

        <meta charset="utf-8">

        <meta http-equiv="X-UA-Compatible" content="IE=edge">

        <meta name="viewport" content="width=device-width,
initial-scale=1">

        <title>Laravel</title>

        <!-- Fonts -->

        <link

href="https://fonts.googleapis.com/css?family=Raleway:100,600"

rel="stylesheet" type="text/css">

        <!-- Styles -->

```



```
<style>

    html, body {

        background-image: url("/images/background.jpeg");

        color: #636b6f;

        font-family: 'Raleway', sans-serif;

        font-weight: 100;

        height: 100vh;

        margin: 0;

    }

    .full-height {

        height: 100vh;

    }

    .flex-center {

        align-items: center;

        display: flex;

        justify-content: center;

    }

    .position-ref {

        position: relative;

    }

    .top-right {

        position: absolute;

        right: 10px;

        top: 18px;

    }

    .content {
```

```

        text-align: center;

        position: absolute;

        bottom: 140px;
    }

    .title {

        font-size: 84px;
    }

    .links > a {

        color: #636b6f;

        padding: 0 25px;

        font-size: 12px;

        font-weight: 600;

        letter-spacing: .1rem;

        text-decoration: none;

        text-transform: uppercase;
    }

    .m-b-md {

        margin-bottom: 30px;
    }
</style>
</head>
<body>

    <div class="flex-center position-ref full-height">

        @if (Route::has('login'))

            <div class="top-right links">

                @if (Auth::check())

```

```

        <a href="{{ url('/home') }}">Home</a>

    @else

        <a href="{{ url('/login') }}">Login</a>

        <a href="{{ url('/register') }}">Register</a>

    @endif

</div>

@endif

<div class="content">

    <div class="title m-b-md">

        Textbooks

    </div>

    <div class="links">

        <a href="{{ url('/home') }}">Home</a>

        <a href="{{ url('/dashboard') }}">Dashboard</a>

        <a href="{{ url('/home') }}">Learn More</a>

        <a href="{{ url('/books') }}">Books</a>

    </div>

</div>

</div>

</body>

</html>

```

Dashboard.blade.php

```
<x-app-layout>
```

```

<x-slot name="header">

    <h2 class="font-semibold text-xl text-gray-800
leading-tight">

        {{ __('Dashboard') }}

    </h2>

</x-slot>


<div class="py-12">

    <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

        <div class="bg-white overflow-hidden shadow-sm
sm:rounded-lg">

            <div class="p-6 bg-white border-b border-gray-200">

                You're logged in!

            </div>

        </div>

    </div>

</div>

</x-app-layout>

```

Home.blade.php

```

<x-app-layout>

    <x-slot name="header">

        <h2 class="font-semibold text-xl text-gray-800
leading-tight">

            {{ __('Home') }}

        </h2>

```

```

    </x-slot>

<div class="max-w-10xl mx-auto sm:px-6 lg:px-8">
    
</div>

<div class="py-12">
    <div class="max-w-10xl mx-auto sm:px-6 lg:px-8">
        <div class="bg-white overflow-hidden shadow-sm
sm:rounded-lg">
            <div class="p-6 bg-white border-b border-gray-200">
                <h1 class="about">About</h1>
                <h2 class="about-head">We are so excited to have
you!</h2>
                <p class="about-content">Welcome to the website
designed for the students of Northwest Nazarene University (NNU) to
buy and sell books to each other. While there are many other places
students can buy/sell their used textbooks, the goal was to provide a
safe, reliable, convenient platform for our students to interact with
each other.

                </p>
            </div>
        </div>
    </div>
</div>

```

```
</x-app-layout>
```

Create.blade.php

```
<x-app-layout>
```

```
    <x-slot name="header">
```

```
        <h2 class="font-semibold text-xl text-gray-800
leading-tight">
```

```
            {{ __('Create') }}
```

```
        </h2>
```

```
    </x-slot>
```

```
<div class="py-12">
```

```
    <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
```

```
        <div class="bg-white overflow-hidden shadow-sm
sm:rounded-lg">
```

```
            <div class="p-6 bg-white border-b border-gray-200">
```

```
                @if ($errors->any())
```

```
                    <div class="alert alert-danger">
```

```
                        <ul>
```

```
                            @foreach ($errors->all() as $error)
```

```
                                <li>{{ $error }}</li>
```

```
                            @endforeach
```

```
                        </ul>
```

```

        </div>

    @endif

    {!! Form::open(['url' => '/books', 'method' =>
'POST', 'enctype' => 'multipart/form-data']) !!}

    <div class="form-group">

        {!!Form::label('inputname', 'Name')!!}

        {!!Form::text('inputname', '', ['class' =>
'form-control', 'placeholder' => 'First and Last'])!!}

    </div>

    <div class="form-group">

        {!!Form::label('inputemail', 'Email')!!}

        {!!Form::text('inputemail', '', ['class' =>
'form-control', 'placeholder' => 'Email'])!!}

    </div>

    <div class="form-group">

        {!!Form::label('title', 'Title')!!}

```

```

        {{Form::text('title', '', ['class' =>
'form-control', 'placeholder' => 'Title'])}}}

    </div>

    <div class="form-group">

        {{Form::label('author', 'Author')}}

        {{Form::text('author', '', ['class' =>
'form-control', 'placeholder' => 'Author'])}}}

    </div>

    <div class="form-group">

        {{Form::label('isbn', 'ISBN')}}

        {{Form::text('isbn', '', ['class' =>
'form-control', 'placeholder' => 'ISBN'])}}}

    </div>

    <div class="form-group">

        {{Form::label('subject', 'Subject')}}

        {{Form::text('subject', '', ['class' =>
'form-control', 'placeholder' => 'Subject'])}}}

```



```
</div>
```

```
<div class="form-group">
```

```
  {{Form::label('description', 'Description')}}
```

```
  {{Form::textarea('description', '', ['class'
```

```
=> 'form-control', 'placeholder' => '(Optional) Description of  
book'])}}}
```

```
</div>
```

```
<div class="form-group">
```

```
  {{Form::label('price', 'Price')}}
```

```
  {{Form::text('price', '', ['class' =>
```

```
'form-control', 'placeholder' => '$'])}}}
```

```
</div>
```

```
<div class="form-group">
```

```
  {{Form::label('image', 'Cover Image')}}
```

```
  {{Form::file('image')}}
```

```
</div>
```

```
        {{Form::submit('Submit', ['class' => 'btn
btn-primary'])}}
```

```
    {!! Form::close() !!}


```

```
    </div>


```

```
    </div>


```

```
    </div>


```

```
    </div>


```

```
</x-app-layout>
```

Index.blade.php

```
<x-app-layout>
```

```
    <x-slot name="header">
```

```
        <h2 class="font-semibold text-xl text-gray-800
leading-tight">
```

```
            {{ __('Books') }}
```

```
        </h2>
```

```
    </x-slot>
```

```

<div class="relative flex lg:inline-flex items-center bg-gray-100
rounded-xl px-3 py-2">

    <form method="GET" action="#">

        <input type="text" name="search" placeholder="Search..."
            class="bg-transparent placeholder-black font-semibold
text-sm"

            value="{{ request('search') }}">

    </form>

</div>

@if(count($books) > 0)

    @foreach($books as $book)

        <div class="py-12">

            <div class="max-w-5xl mx-auto sm:px-6 lg:px-8">

                <div class="bg-white overflow-hidden shadow-sm
sm:rounded-lg">

                    <div class="p-6 bg-white border-b border-gray-200">

                        <br />

```

```

        <div style="position: relative; left: 40px;">
            <h1 style="font-weight: bold;"><a
href="/books/{{ $book->id }}">{{ $book->title }}</a></h1>

            <br />
            Author: {!!$book->author!!}
            <br />
            <br />
            Subject: {!!$book->subject!!}
            <br />
            <br />
        </div>

        <a href="/books/{{ $book->id }}" class="btn
btn-default" style="position: relative; left: 425px;">View</a>

    </div>
</div>
</div>
</div>

@endforeach

{{ $books->links() }}

@else

```

```

        <p>No books found</p>
    @endif

```

```

</x-app-layout>

```

Show.blade.php

```

<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800
leading-tight">
            {{ __('Books') }}
        </h2>
    </x-slot>
    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="bg-white overflow-hidden shadow-sm
sm:rounded-lg">
                <div class="p-6 bg-white border-b border-gray-200">
                    
                    <br />
                    <h1 style="font-weight: bold; position: relative;
left: 100px;">{{ $books->title }}</h1>

```

```

<div style="position: relative; left: 100px;">

    Author: {!!$books->author!!}<br/>

    ISBN: {!!$books->isbn!!}<br/>

    Subject: {!!$books->subject!!}<br/>

    Description: {!!$books->description!!}<br/>

    Price: ${!!$books->price!!}<br/>

    Posted: {!!$books->created_at!!}

    <br><br/>

    <br/>

</div>

<div style="position: relative; left: 100px;">

    Name: {!!$books->inputname!!}<br/>

    Email: {!!$books->inputemail!!}<br/>

</div>

<br />

<br />

<div style="position: relative; left: 500px;">

    <a href="/books/{{ $books->id }}/edit" class="btn
btn-default">Edit</a>

    {!!Form::open(['url' => ['/books', $books->id],
'method' => 'POST', 'class' => 'pull-right'])!!}

    {{Form::hidden('_method', 'DELETE')}}

    {{Form::submit('Delete', ['class' => 'btn
btn-danger'])}}

    {!!Form::close()!!}

```

```

        </div>

        </div>

    </div>

</div>

</div>

</x-app-layout>

```

Edit.blade.php

```

<x-app-layout>

    <x-slot name="header">

        <h2 class="font-semibold text-xl text-gray-800
leading-tight">

            {{ __( 'Books' ) }}

        </h2>

    </x-slot>

    <div class="py-12">

        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

            <div class="bg-white overflow-hidden shadow-sm
sm:rounded-lg">

                <div class="p-6 bg-white border-b border-gray-200">

                    Edit Post

                </div>

            </div>

        </div>

    </div>

    @if ($errors->any())

        <div class="alert alert-danger">

```

```

        <ul>

            @foreach ($errors->all() as $error)

                <li>{{ $error }}</li>

            @endforeach

        </ul>

    </div>

@endif

{!! Form::open(['url' => ['/books', $books->id],
'method' => 'POST']) !!}

    <div class="form-group">

        {{Form::label('title', 'Title')}}

        {{Form::text('title', $books->title, ['class'
=> 'form-control', 'placeholder' => 'Title'])}}

    </div>

    <div class="form-group">

        {{Form::label('author', 'Author')}}

        {{Form::text('author', $books->author,
['class' => 'form-control', 'placeholder' => 'Author'])}}

    </div>

```



```

<div class="form-group">

    {{Form::label('isbn', 'ISBN')}}

    {{Form::text('isbn', $books->isbn, ['class'
=> 'form-control', 'placeholder' => 'ISBN'])}}

</div>

<div class="form-group">

    {{Form::label('subject', 'Subject')}}

    {{Form::text('subject', $books->subject,
['class' => 'form-control', 'placeholder' => 'Subject'])}}

</div>

<div class="form-group">

    {{Form::label('description', 'Description')}}

    {{Form::textarea('description',
$books->description, ['class' => 'form-control', 'placeholder' =>
'(Optional) Description of book'])}}

</div>

```

```

<div class="form-group">

    {{Form::label('price', 'Price')}}

    {{Form::text('price', $books->price, ['class'
=> 'form-control', 'placeholder' => 'Price'])}}

</div>

<div class="form-group">

    {{Form::label('image', 'Cover Image')}}

    {{Form::file('image')}}

</div>

{{Form::hidden('_method', 'PUT')}}

{{Form::submit('Submit', ['class' => 'btn
btn-primary'])}}

{!! Form::close() !!}

</div>

</div>

</div>

```

```
</div>
</x-app-layout>
```

BookController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use App\Models\Book;

class BookController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    /**
     * The attributes that aren't mass assignable.
     *
     * @var array
     */
    protected $guarded = [];

    public function index(Request $request)
```

```

{
    $books = Book::orderBy('title','asc');

    if (request('search')) {
        $books->where('title','like','%'.request('search').'%');
    }

    return view('books/index', [

        'books' => $books->orderBy('title','asc')->paginate(10),

    ]);

}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    return view('books.create');
}

```

```

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required',
        'author' => 'required',
        'isbn' => 'required', //|integer|min:0|max:
        'subject' => '',
        'description' => '',
        'price' => '',
        'inputname' => '',
        'inputemail' => ''

    ]);

    // Create Book

    $book = new Book;

    $book->title = $request->title;
    $book->author = $request->author;
    $book->isbn = $request->isbn;

```

```

$book->subject = $request->subject;

$book->description = $request->description;

$book->price = $request->price;

$book->inputname = $request->inputname;

$book->inputemail = $request->inputemail;


if($request->file('image')) //function to upload an image
{
    $file = $request->file('image');

    $extension = $file->getClientOriginalExtension();

    $filename = md5($file->getClientOriginalName() .
time()).'.'.$extension;

    $file->move('images', $filename);

    $book->image = $filename;

}


$book->save(); //saves all information to books table


return redirect('books')->with('success', 'You did it!');
}


/**
 * Display the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response

```

```

    */

//created on 11/18
// may be wrong idk

    public function show($id)
    {

        $books = Book::find($id);

        // method to select a book

        return view ('books/show')->with('books', $books);

    }

/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
    public function edit($id)
    {

        $books = Book::find($id); // method to select a book

        return view ('books/edit')->with('books', $books);

    }

```

```

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

    $validated = $request->validate([
        'title' => 'required',
        'author' => 'required',
        'isbn' => 'required',
        'subject' => '',
        'description' => '',
        'price' => '',
        'inputname' => '',
        'inputemail' => ''
    ]);

    $book = Book::where('id', $id)->get();

    // $book = Book::find($id);

    $book->title = $request->title;

    $book->author = $request->author;

```



```

$book->isbn = $request->isbn;

$book->subject = $request->subject;

$book->description = $request->description;

$book->price = $request->price;

$book->inputname = $request->inputname;

$book->inputemail = $request->inputname;


if($request->file('image')) //function to upload an image
{
    $file = $request->file('image');

    $extension = $file->getClientOriginalExtension();

    $filename = md5($file->getClientOriginalName() .
time()).'.'.$extension;

    $file->move('images', $filename);

    $book->image = $filename;
}


return redirect('books')->with('success', 'It has been
updated!');
}


/**
 * Remove the specified resource from storage.
 *
 * @param int $id

```

```

    * @return \Illuminate\Http\Response
    */
    public function destroy($id)
    {
        $book = Book::find($id); // method to select a book

        $book->delete();

        return redirect('books')->with('success', 'Book has been
Removed');
    }
}

```

