

GEEKYFREAK – comics site

“Comics are a gateway drug to literacy.” @ Art Spiegelman

Team members:

Farrukh Tulkunov	U1610069
Jasur Turaev	U1610095
Ibrokhim Orifjonov	U1610076
Nodirjon Utkirov	U1610170
Saidazimkhon Jasurbekov	U1610195

Link to GITHUB repo: <https://github.com/iuthub/design-project-geekyfreak>

Link to WEBSITE: <https://geekyfreak.000webhostapp.com/>

Brief information about site:

Simple, convenient, innovative, creative... These are the main basics on which our site was created.

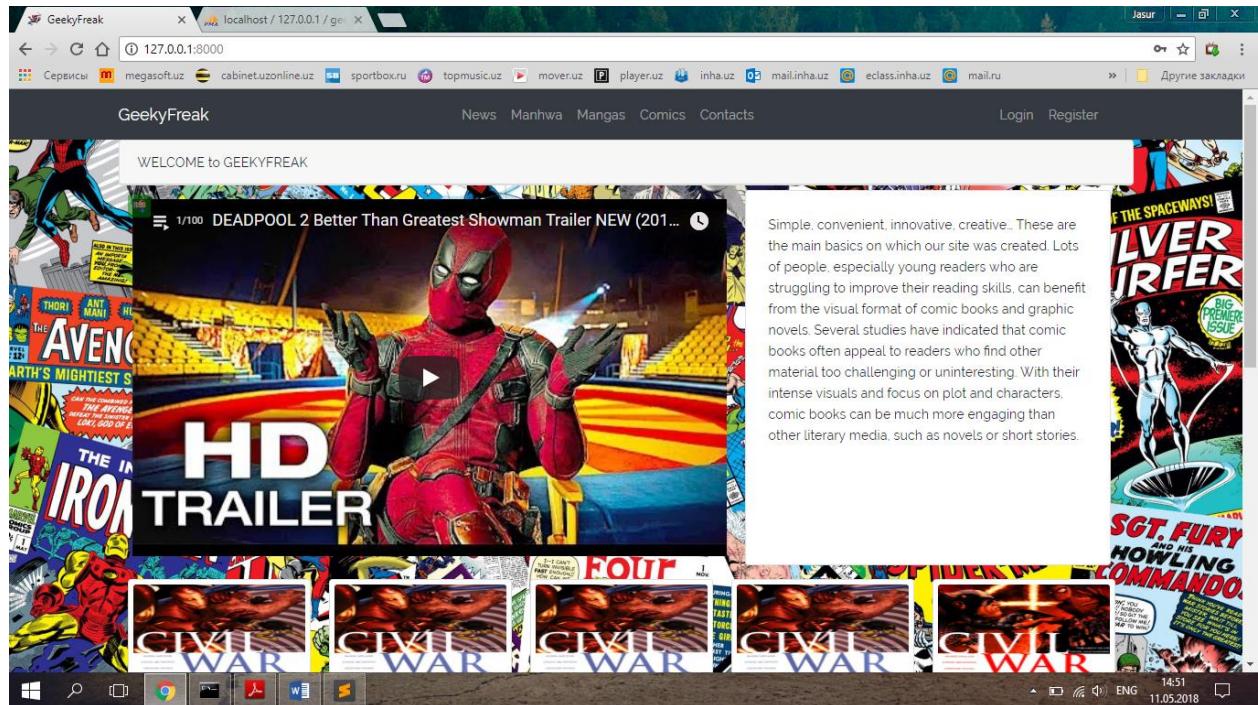
Lots of people, especially young readers who are struggling to improve their reading skills, can benefit from the visual format of comic books and graphic novels. Several studies have indicated that comic books often appeal to readers who find other material too challenging or uninteresting. With their intense visuals and focus on plot and characters, comic books can be much more engaging than other literary media, such as novels or short stories.

Our site provides with the main and the most needed information about *comics*, *manga*, *manhwa*.

You can also find news about latest releases of comics by simple keeping the watching the [video](#) on our home page.

The site has register-login operations, that means that you can have your favorite books in your device immediately without searching. In order to improve our site by giving advices you are able to connect with us via our contacts and also the comments for discussion with people worldwide are open for you just after authorizing.

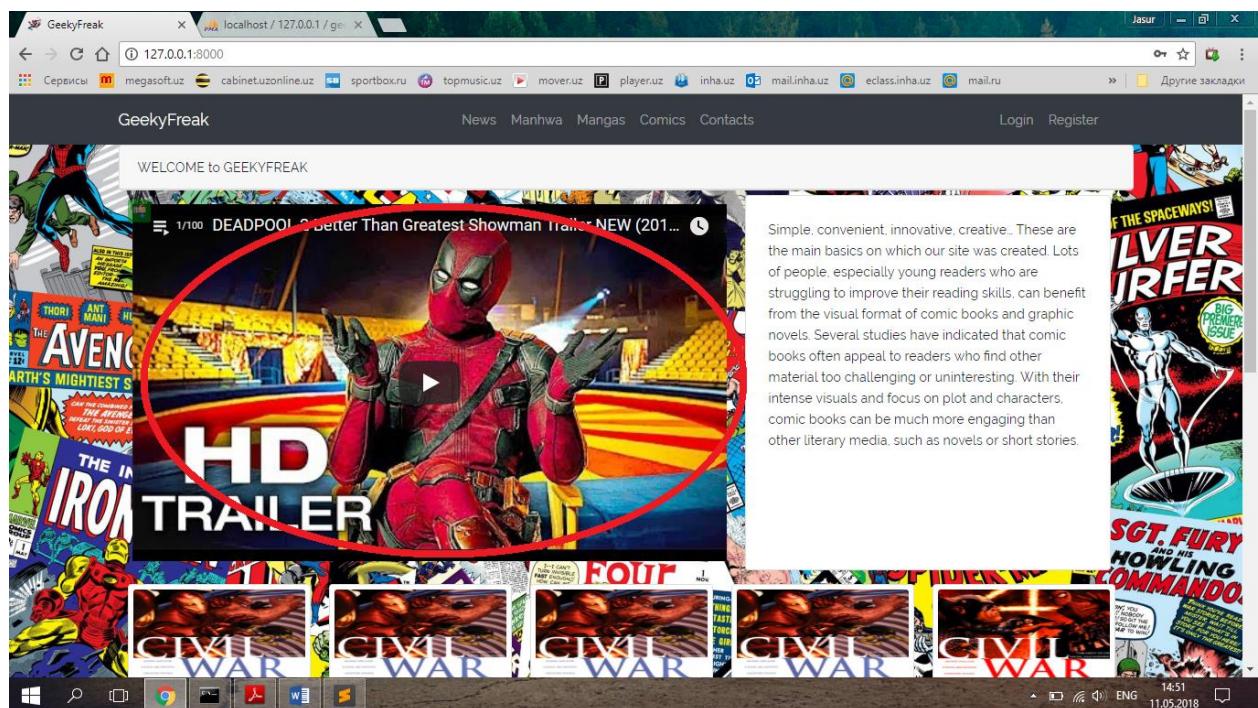
Developer side:



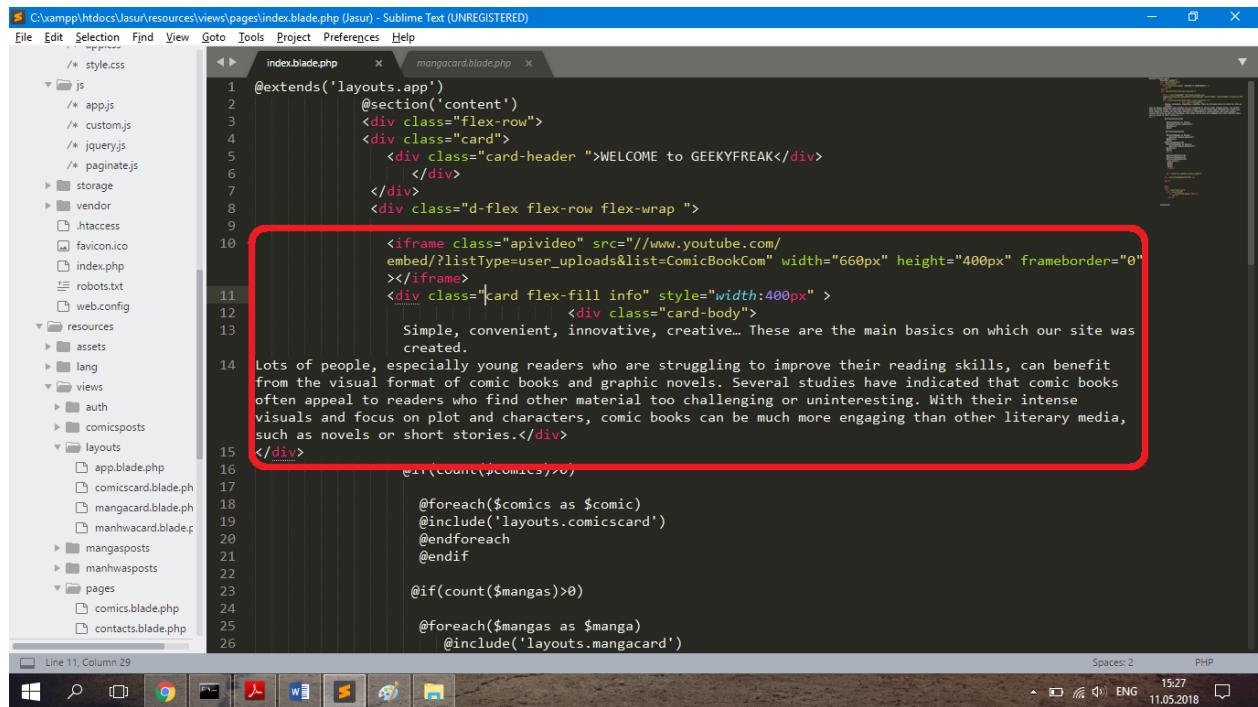
Home page consists of all comic, manga and manhwa books and each time updating video. On the top you can see the navbar on each of your pages.

The video was added using API feature. The <iframe> tag was used for completion.

In this row, you can also see some brief inspirational information about site.

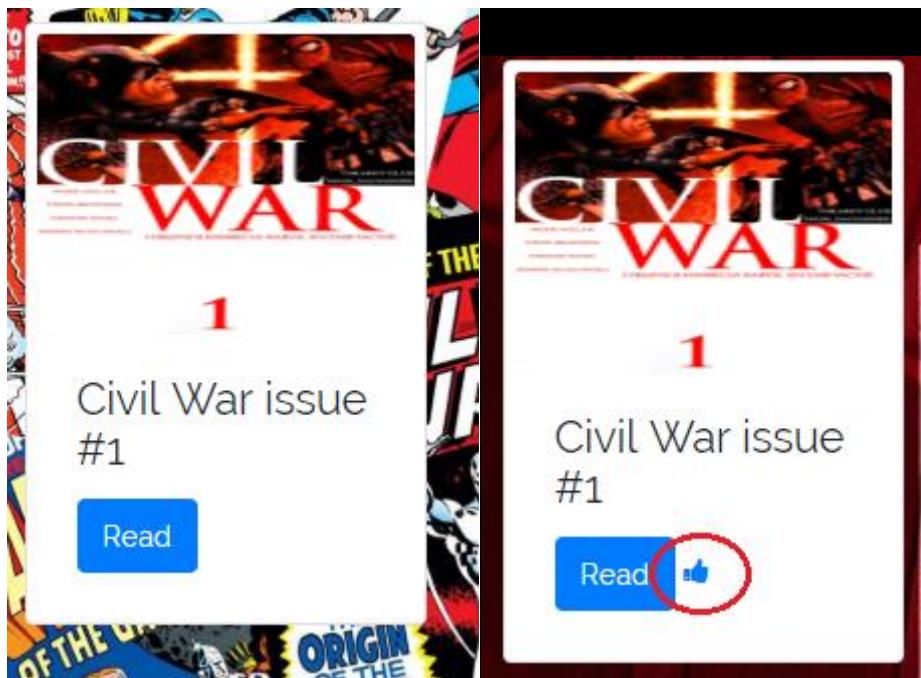


This is how we implemented with code:

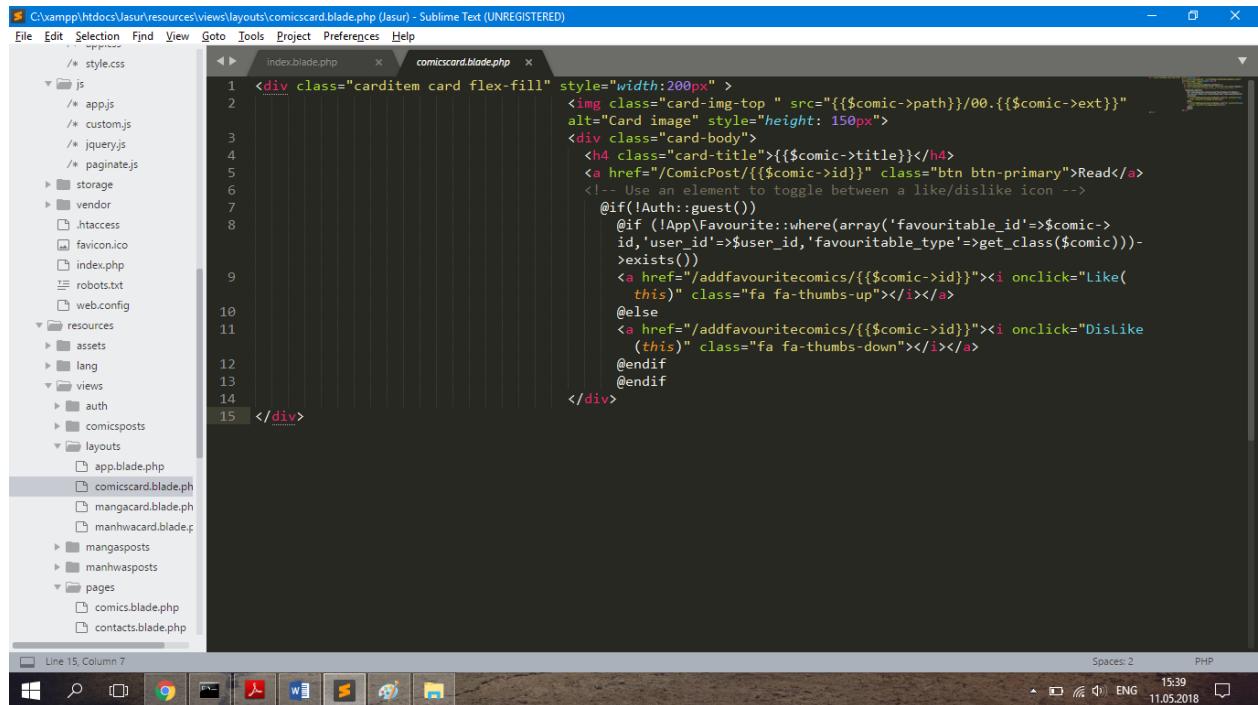


The screenshot shows two tabs open in Sublime Text: 'index.blade.php' and 'mangacard.blade.php'. The left sidebar displays a file tree for a Laravel-like application structure. The 'index.blade.php' tab contains the following code:`1 @extends('layouts.app')
2 @section('content')
3 <div class="flex-row">
4 <div class="card">
5 <div class="card-header ">WELCOME to GEEKYFREAK</div>
6 </div>
7 <div class="d-flex flex-row flex-wrap ">
8
9
10 <div class="apivideo" src="//www.youtube.com/embed/?listType=user_uploads&list=ComicBookCom" width="660px" height="400px" frameborder="0"></div>
11 <div class="card flex-fill info" style="width:400px;">
12 <div class="card-body">
13 Simple, convenient, innovative, creative... These are the main basics on which our site was created.
14 Lots of people, especially young readers who are struggling to improve their reading skills, can benefit from the visual format of comic books and graphic novels. Several studies have indicated that comic books often appeal to readers who find other material too challenging or uninteresting. With their intense visuals and focus on plot and characters, comic books can be much more engaging than other literary media, such as novels or short stories.</div>
15 @if(count($comics)>0)
16 @foreach($comics as $comic)
17 @include('layouts.comicscard')
18 @endforeach
19 @endif
20
21 @if(count($mangas)>0)
22 @foreach($mangas as $manga)
23 @include('layouts.mangacard')
24 @endforeach
25 @endif
26
27 </div>
28 </div>
29`

The comics-cards gives you possibility to read them if you are logged in and also the icon “thumbs up” allows you to add the comics book to your favorite catalogue.



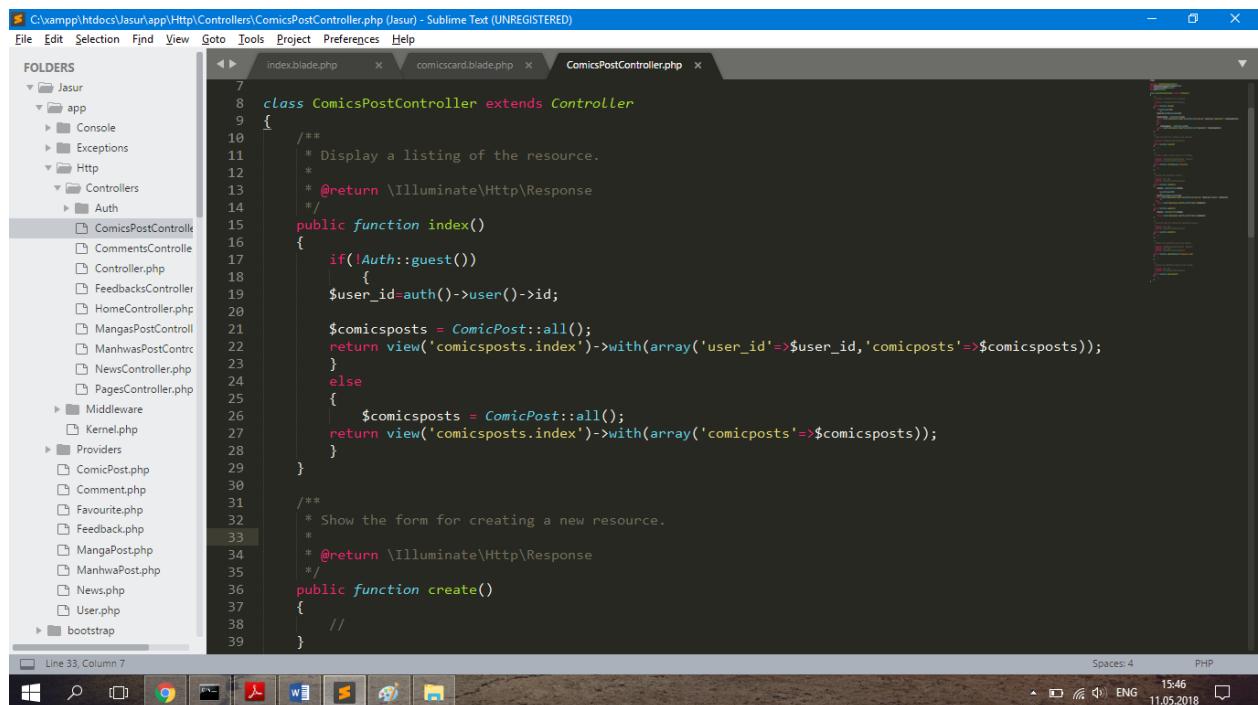
Here is the code



The screenshot shows a Sublime Text window with two tabs open: 'index.blade.php' and 'comicscard.blade.php'. The file path in the sidebar is 'C:\xampp\htdocs\Jasur\resources\views\layouts\comicscard.blade.php (Jasur) - Sublime Text (UNREGISTERED)'. The code in 'comicscard.blade.php' is a Blade template for a comic card. It includes HTML for a card item with a title, image, and body, and Blade logic for handling likes and dislikes. The code is color-coded for syntax highlighting.

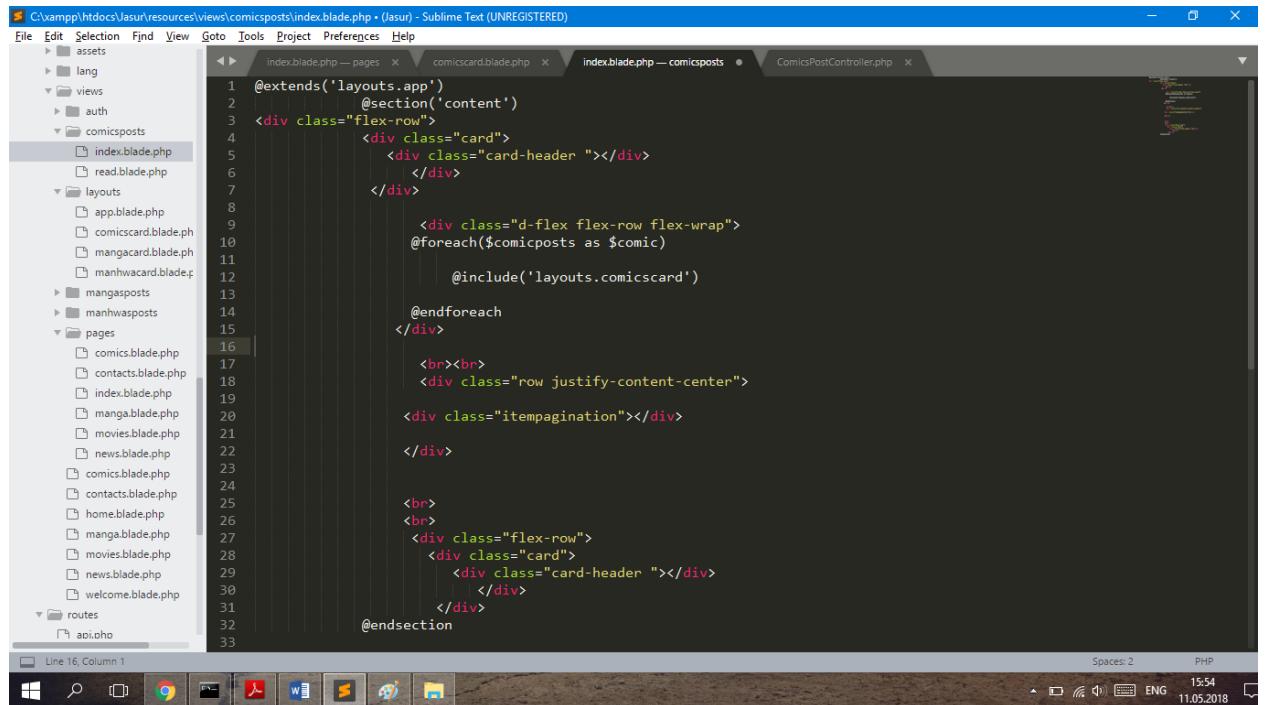
To achieve this, we needed model named “comicspost”, controller named “comicspostcontroller”, and a route to given page and most importantly a data inside database;

Route::resource('ComicPost','ComicsPostController'); this route will call the index function when /ComicPost is being opened



The screenshot shows a Sublime Text window with three tabs: 'index.blade.php', 'comicscard.blade.php', and 'ComicsPostController.php'. The file path in the sidebar is 'C:\xampp\htdocs\Jasur\app\Http\Controllers\ComicsPostController.php (Jasur) - Sublime Text (UNREGISTERED)'. The code in 'ComicsPostController.php' is a PHP controller for the 'ComicsPost' resource. It contains methods for 'index()' and 'create()'. The 'index()' method checks if the user is guest, retrieves all comics posts, and returns a view with the user_id and comicposts data. The 'create()' method is currently empty. The code is color-coded for syntax highlighting.

Here in the controller index function checks if you are logged in or not. It will get all the comics from database and pass it in array to the view comicpost. In case you are logged in it will also include your user id.

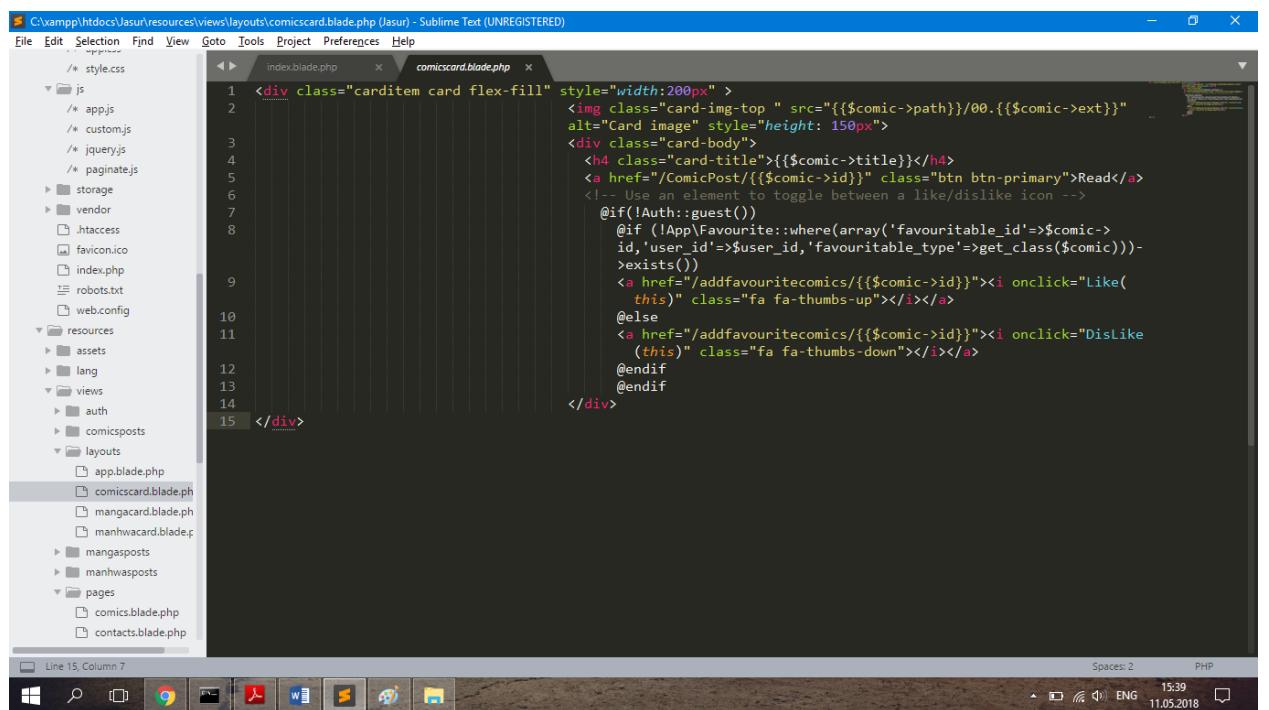


```

C:\xampp\htdocs\Jasur\resources\views\comicposts\index.blade.php (Jasur) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
index.blade.php - pages - comiccard.blade.php - index.blade.php — comicposts - ComicsPostController.php
1 @extends('layouts.app')
2     @section('content')
3     <div class="flex-row">
4         <div class="card">
5             <div class="card-header"></div>
6         </div>
7     </div>
8
9         <div class="d-flex flex-row flex-wrap">
10            @foreach($comicposts as $comic)
11                @include('layouts.comiccard')
12            @endforeach
13        </div>
14
15        <br><br>
16        <div class="row justify-content-center">
17            <div class="itempagination"></div>
18        </div>
19
20        <br>
21        <br>
22        <div class="flex-row">
23            <div class="card">
24                <div class="card-header"></div>
25            </div>
26        </div>
27
28        <br>
29        <br>
30    @endsection

```

Here as you can see in index.blade we include all the necessary tags by extending layout and also we include comiccard layout which can be seen below.



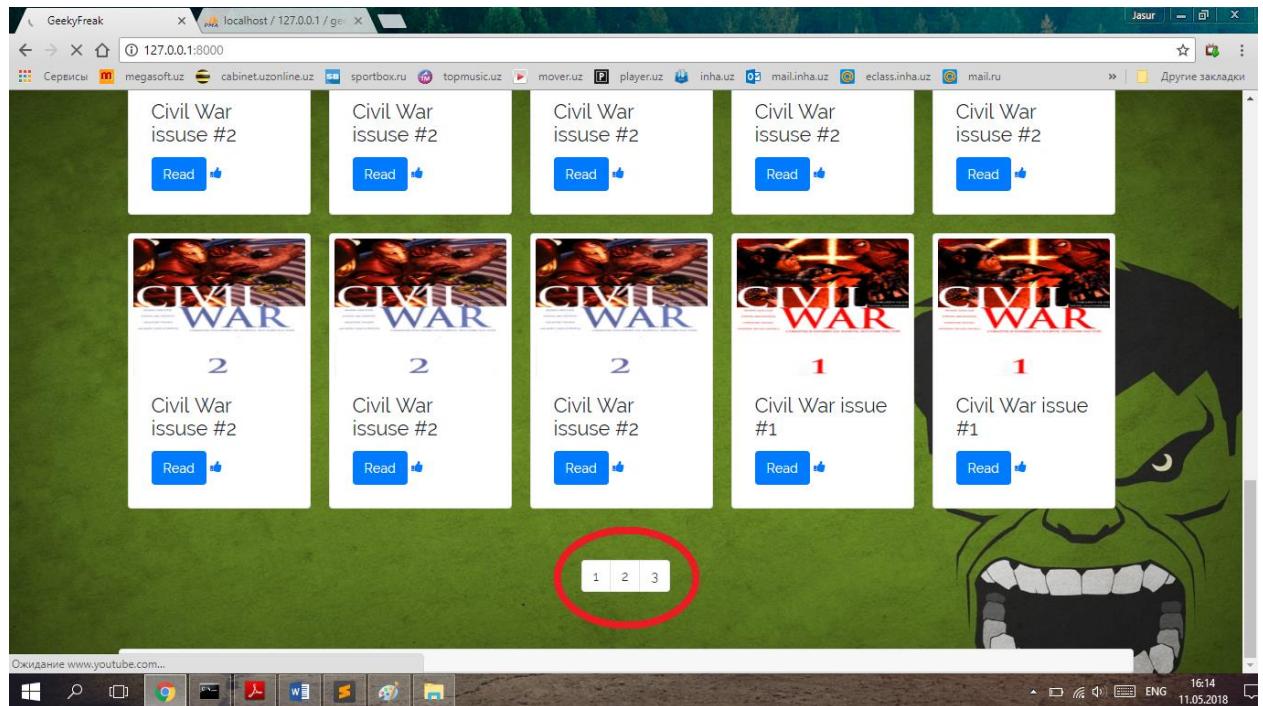
```

C:\xampp\htdocs\Jasur\resources\views\layouts\comiccard.blade.php (Jasur) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
/* style.css
/* app.js
/* custom.js
/* jquery.js
/* paginate.js
storage
vendor
.htaccess
favicon.ico
index.php
robots.txt
web.config
resources
assets
lang
views
auth
comicposts
layouts
app.blade.php
comiccard.blade.php
mangocard.blade.php
manhwacard.blade.php
mangaposts
manhwaposts
pages
comics.blade.php
contacts.blade.php
Line 15, Column 7
index.blade.php - comiccard.blade.php
1 <div class="carditem card flex-fill" style="width:200px" >
2     
4     <div class="card-body">
5         <h4 class="card-title">{{ $comic->title }}</h4>
6         <a href="/ComicPost/{{ $comic->id }}" class="btn btn-primary">Read</a>
7         <!-- Use an element to toggle between a like/dislike icon -->
8         @if(!Auth::guest())
9             @if (App\Favourite::where('favouritable_id'=>$comic->id,'user_id'=>$user_id,'favouritable_type'=>get_class($comic))->exists())
10                 <a href="/addfavouritecomics/{{ $comic->id }}"><i onclick="Like(this)" class="fa fa-thumbs-up"></i></a>
11             @else
12                 <a href="/addfavouritecomics/{{ $comic->id }}"><i onclick="DisLike(this)" class="fa fa-thumbs-down"></i></a>
13             @endif
14         @endif
15     </div>

```

In this view you can see variable \$comics, this is a variable given by controller and it is model with its own attributes which are specified in database. It will load all the requested information from database and show in our page.

To provide feature to add favorite item firstly we implemented many-to-many relation between comics and users. One user can have many favorite comics and one comic can belong to many users. So before showing button to add favorite we check if it isn't already a favorite item. If it is we provide a button to remove it from favorites.



When the number of comic-cards reaches 20, at the bottom you can see the page number and using them it's easy to reach any page you want.

JQUERY and JS were used to implement the pagination.

```
C:\xampp\htdocs\Jasur\public\js\paginate.js (Jasur) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
  session.php
  view.php
  database
  public
    / app.css
    / style.css
  js
    /* app.js
    / custom.js
    / jquery.js
    /* paginate.js
  storage
  vendor
    .htaccess
    favicon.ico
    index.php
    robots.txt
    web.config
  resources
    assets
      js
      sass
    lang
    views
      auth
    comicposts
      index.blade.php
      read.blade.php
  paginate.js
  
```

The code in the file is a JavaScript function named \$fn.itemPaginate. It takes an options object and initializes variables like PaginationContainer, itemsToPaginate, and defaultv. It then extends the settings object with defaultv and options. It calculates the number of pages based on the total items and the items per page. It creates a navigation container and appends a list of page links. It then hides items that exceed the items per page limit. Finally, it adds click event listeners to the page links to handle navigation logic.

```

        });
      };
      $fn.itemPaginate=function(options){
        var PaginationContainer=this;
        var itemsToPaginate;
        var defaultv={
          itemsPerPage:20
        };
        var settings={};
        $.extend(settings,defaultv,options);
        itemsToPaginate=settings.itemsPerPage;
        var numberoflinks=Math.floor((itemsToPaginate.length/settings.itemsPerPage));
        $("ul class='pagination flex-wrap'").appendTo(PaginationContainer);

        for (var i = 0; i < numberoflinks; i++) {
          PaginationContainer.find("li").append("<a href='"+(i+1)*settings.itemsPerPage+"' class='page-item' onclick='topFunction()' >"+(i+1)+"</a></li>");
        }

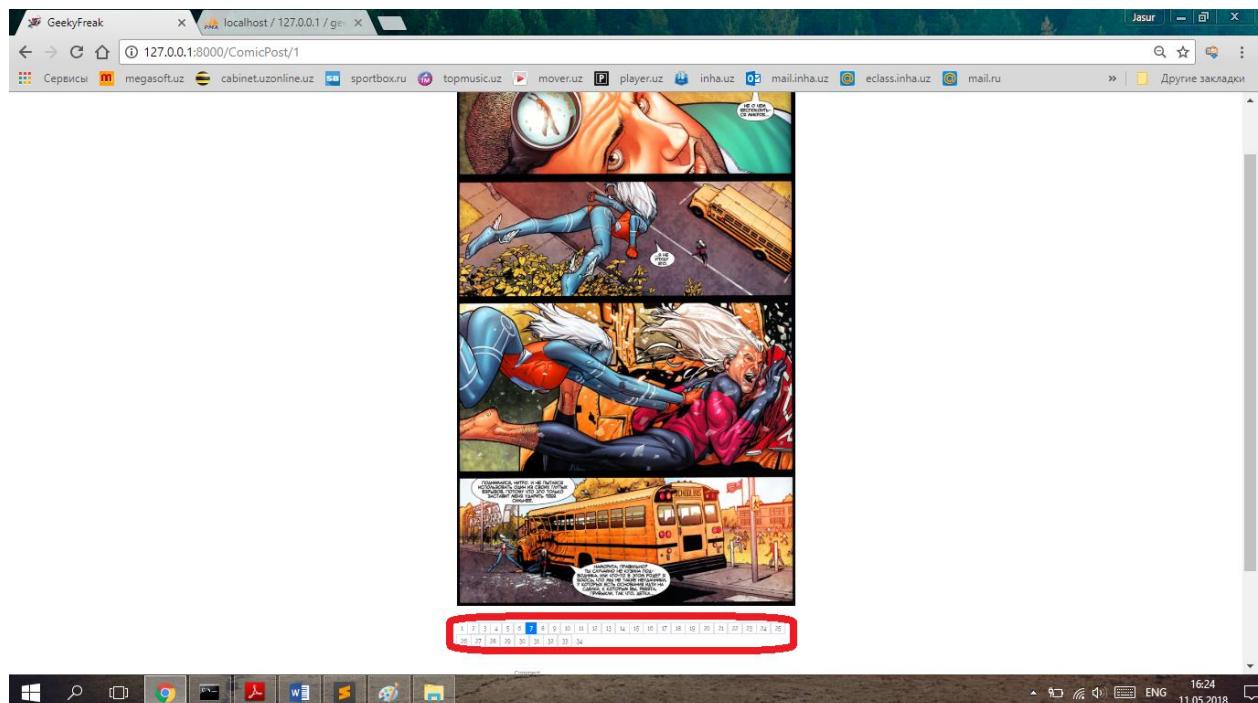
        itemsToPaginate.filter(":gt("+Math.floor((linknumber-1)*settings.itemsPerPage)+")").hide();
        PaginationContainer.find("ul li").on('click', function(event) {
          $(this).addClass('active')
          $(this).siblings().removeClass('active')

          var linknumber=$(this).text();
          var itemstohide=itemsToPaginate.filter(":lt(("+(linknumber-1)*settings.itemsPerPage)+")");
          $.merge(itemstohide, itemsToPaginate.filter(":gt(("+(linknumber*settings.itemsPerPage)-1)+")"));

          itemstohide.hide();
          var itemstoshow=itemsToPaginate.not(itemstohide);
          itemstoshow.show();
        });
      };
    })(jQuery);
  
```

Comics, manhwa and manga pages are all implemented in the same way. The only difference is that comics page has comics, manhwa has manhwa and so on. But home and favorites pages contain all of them.

The process of reading the books:



As you know, comics are the books that are created with many images. So, in order to make that books readable, we made the same principle as in pagination in the home page. The difference is that here only the images are changing instead of the whole page refreshing.

```
function($) {
    $.fn.Paginate = function(options) {
        var PaginationContainer = this;
        var itemsToPaginate;
        var defaultv = {
            itemsPerPage: 1
        };
        var settings = {};
        $.extend(settings, defaultv, options);
        itemsToPaginate = settings.itemsToPaginate;
        var numberoflinks = Math.ceil((itemsToPaginate.length / settings.itemsPerPage));
        $("ul").append("ul").prependTo(PaginationContainer);

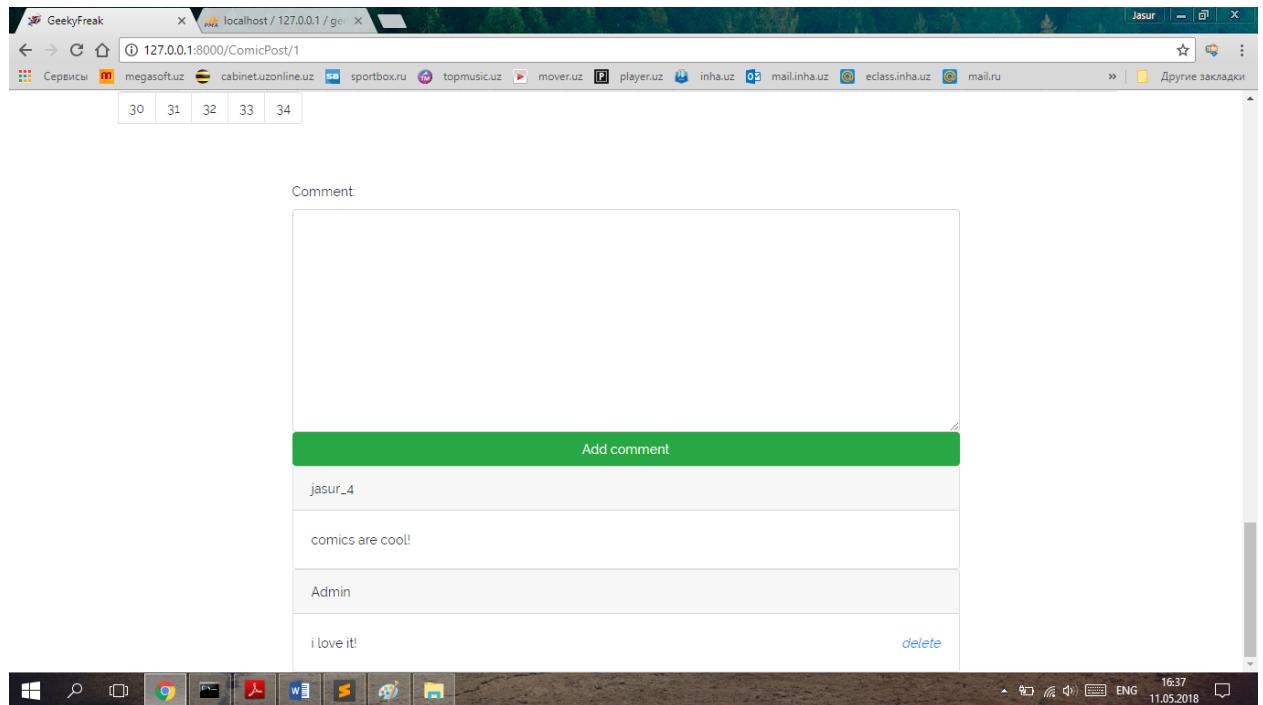
        for (var i = 0; i < numberoflinks; i++) {
            PaginationContainer.find("ul").append("<li class='page-item'><a onclick='topFunction()' class='page-link'>" + (i + 1) + "</a></li>");
        }

        itemsToPaginate.filter(":gt(" + (settings.itemsPerPage - 1) + ")").hide();
        PaginationContainer.find("ul li").on("click", function(event) {
            $(this).addClass("active")
            $(this).siblings().removeClass("active")
            var linknumber = $(this).text();
            var itemstohide = itemsToPaginate.filter(":lt(" + ((linknumber - 1) * settings.itemsPerPage) + ")");
            $.merge(itemstohide, itemsToPaginate.filter(":gt(" + ((linknumber * settings.itemsPerPage) - 1) + ")"));
            itemstohide.hide();
            var itemstoshow = itemsToPaginate.not(itemstohide);
            itemstoshow.show();
        });
    };
    $.fn.ItemPaginate = function(options) {
        var PaginationContainer = this;
        var itemsToPaginate;
        var defaultv = {
            itemsPerPage: 1
        };
        var settings = {};
        $.extend(settings, defaultv, options);
        itemsToPaginate = settings.itemsToPaginate;
        var numberoflinks = Math.ceil((itemsToPaginate.length / settings.itemsPerPage));
        $("ul").append("ul").prependTo(PaginationContainer);

        for (var i = 0; i < numberoflinks; i++) {
            PaginationContainer.find("ul").append("<li class='page-item'><a onclick='topFunction()' class='page-link'>" + (i + 1) + "</a></li>");
        }

        itemsToPaginate.filter(":gt(" + (settings.itemsPerPage - 1) + ")").hide();
        PaginationContainer.find("ul li").on("click", function(event) {
            $(this).addClass("active")
            $(this).siblings().removeClass("active")
            var linknumber = $(this).text();
            var itemstohide = itemsToPaginate.filter(":lt(" + ((linknumber - 1) * settings.itemsPerPage) + ")");
            $.merge(itemstohide, itemsToPaginate.filter(":gt(" + ((linknumber * settings.itemsPerPage) - 1) + ")"));
            itemstohide.hide();
            var itemstoshow = itemsToPaginate.not(itemstohide);
            itemstoshow.show();
        });
    };
}
```

At the bottom of each book you can find comments box. And also, there is feature of deleting them.



The implementation was learnt from documentation of laravel.com.

Polymorphic relations were used to link several model with one association. The owning of model to return when accessing the commentable relation is determined by ORM.

A screenshot of Sublime Text showing the "CommentsController.php" file. The code implements polymorphic associations for comments. It includes methods for storing a comment on a comic post and deleting a comment from either a comic post or a manhua.

```
File Edit Selection Find View Goto Tools Project Preferences Help
C:\xampp\htdocs\Jasur\app\Http\Controllers\CommentsController.php (Jasur) - Sublime Text (UNREGISTERED)
Controllers
Auth
ComicsPostController
CommentsController
Controller.php
FeedbacksController
HomeController.php
MangaPostController
ManhwaPostController
NewsController.php
PagesController.php
Middleware
Providers
ComicPost.php
Comment.php
Favourite.php
Feedback.php
MangaPost.php
ManhwaPost.php
News.php
User.php
bootstrap
config
app.php
auth.php
broadcasting.php
cache.php
database.php
filenames.php
Line 1, Column 1
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
*/
public function storecomics(Request $request,$id)
{
    $user_id=auth()->user()->id;
    $user= User::find($user_id);

    $comics=ComicPost::find($id);
    $comment=new Comment();
    $comment->user_id=$user_id;
    $comment->commentable_id=$id;
    $comment->commentable_type=get_class ($comics);
    $comment->body=$request->comment;

    $comment->save();

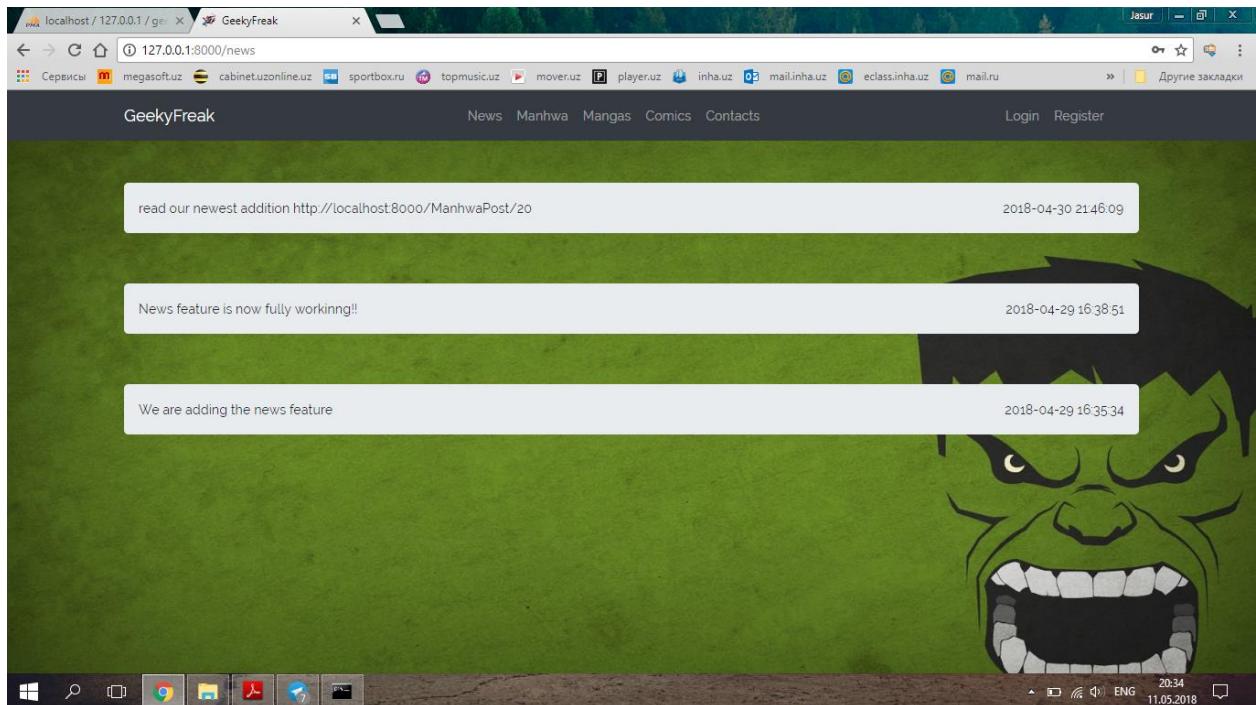
    Session::flash('success','comment has been added ');

    return redirect()->route('ComicPost.show', [$id]);
}

public function deletecomics($comment_id)
{
    $comment=Comment::find($comment_id);
    $comment->delete();
    Session::flash('success','comment has been deleted ');
    return redirect()->back();
}
public function deletemanhwa($comment_id)
{
}
Spaces: 4 PHP
16:39 11.05.2018
```

News.

In news page all the news is loaded from database and they are ordered by latest modification.



Implementation simple:

```
C:\xampp\htdocs\Jasur\resources\views\pages\news.blade.php (Jasur) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  Jasur
    app
    bootstrap
    config
    database
    public
  resources
    assets
    lang
  views
    auth
    comicsposts
    layouts
    mangaposts
    manhwaposts
  pages
    comics.blade.php
    contacts.blade.php
    index.blade.php
    manga.blade.php
    movies.blade.php
    news.blade.php
      news.blade.php
    comics.blade.php
    contacts.blade.php
    home.blade.php
    manga.blade.php
    movies.blade.php
    news.blade.php
news.blade.php
Line 9, Column 1
Spaces: 6
20:39
11.05.2018
```

```

@extends('layouts.app')
@section('content')

@if(count($newsposts)>0)

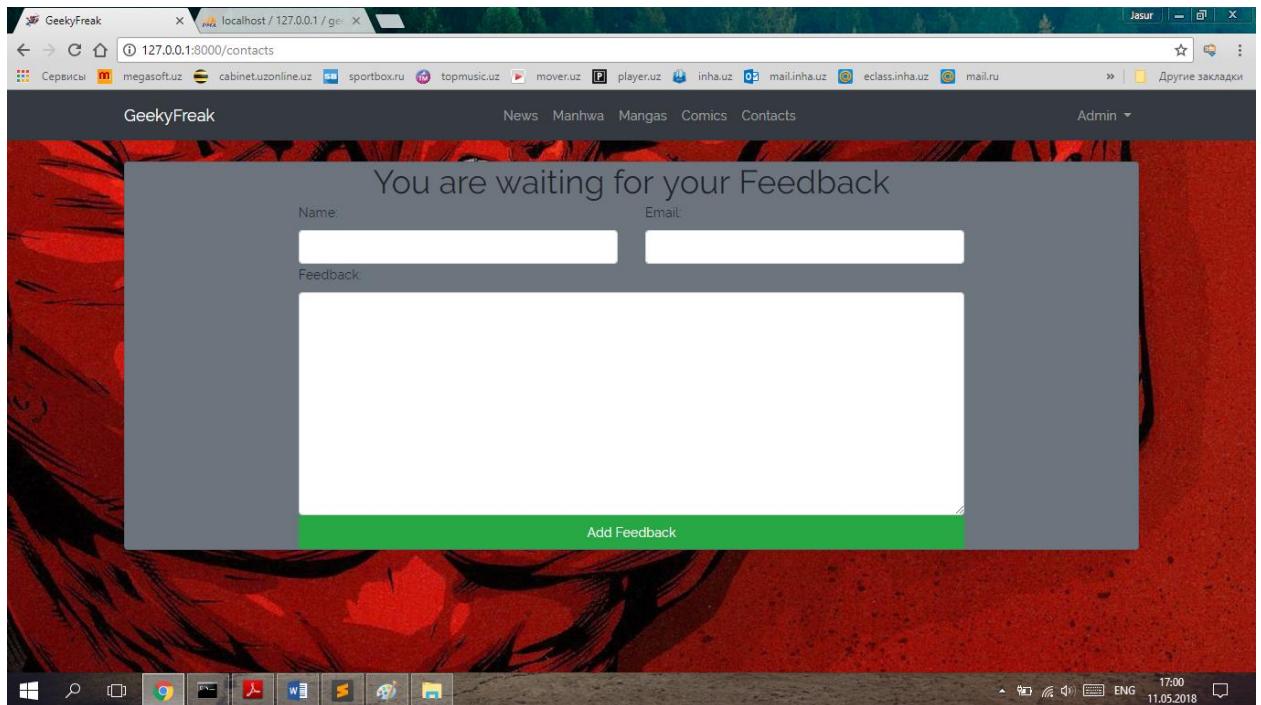
<br><br>
@foreach($newsposts as $news)

<div class="jumbotron p-3">{{$news->news_body}}<span class="float-right">{{$news->created_at}}</span></div>
<br>
@endforeach

<div class="row justify-content-center">{{$newsposts->links()}}</div>
@endif

@endsection
```

The “Contacts” were also added as in other sites in order to provide the user with feedback if he/she has questions, complains, suggestions or gratitude.



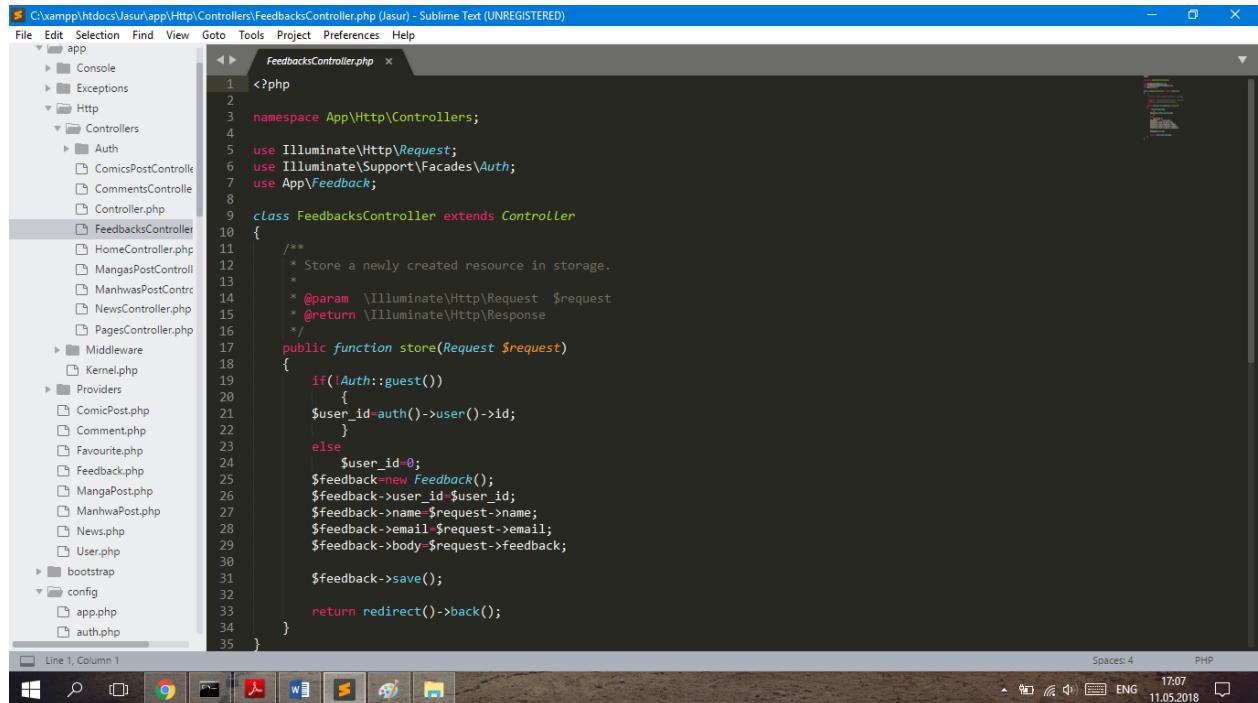
The implementation was made with simple forms.

```
C:\xampp\htdocs\Jasur\resources\views\pages\contacts.blade.php (Jasur) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
> lang
views
  > auth
  > comicposts
    index.blade.php
    read.blade.php
  > layouts
    app.blade.php
    comiccard.blade.php
    mangocard.blade.php
    manhwacard.blade.php
  > mangaposts
  > manhwaposts
  > pages
    comics.blade.php
    contacts.blade.php
    index.blade.php
    manga.blade.php
    movies.blade.php
    news.blade.php
    comics.blade.php
    contacts.blade.php
    manga.blade.php
    movies.blade.php
    news.blade.php
    welcome.blade.php
  > routes
    api.php
    channelroute.php
Line 26, Column 25
1 @extends('layouts.app')
2   @section('content')
3     <br>
4
5       <div class="card bg-secondary m-auto">
6         <h1 class="m-auto">You are waiting for your Feedback</h1>
7         {{Form::open(['route'=>'feedback.store'], 'method'=>'POST')}}
```

```
<div class="row justify-content-center">
  <div class="col-md-4">
    {{Form::label('name', 'Name:')}}
    {{Form::text('name', '', ['class'=>'form-control', 'row'=>'1'])}}
  </div>
  <div class="col-md-4">
    {{Form::label('email', 'Email:')}}
    {{Form::text('email', '', ['class'=>'form-control', 'row'=>'1'])}}
  </div>
<div class="row justify-content-center">
  <div class="col-md-8">
    {{Form::label('feedback', 'Feedback:')}}
    {{Form::textarea('feedback', '', ['id'=>'article-ckeditor', 'class'=>'form-control', 'row'=>'5'])}}
    {{Form::submit('Add Feedback', ['class'=>'btn btn-success btn-block'])}}
  </div>
{{Form::close()}}
```

```
@endsection
```

Form sends the message to database, feedback.store function. Here is it:

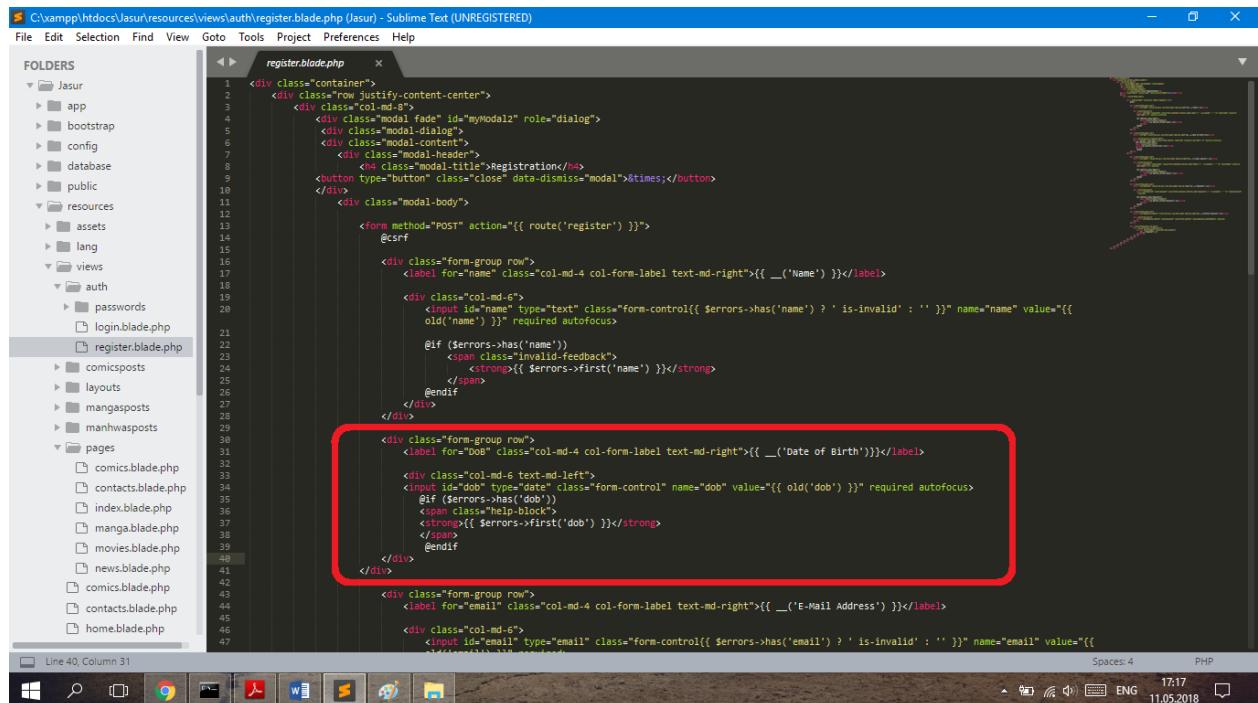


The screenshot shows the Sublime Text editor with the file `FeedbacksController.php` open. The code defines a `FeedbacksController` class that extends `Controller`. It contains a `store` method which checks if the user is a guest or authenticated. If authenticated, it creates a new `Feedback` object, sets its user ID, name, email, and body from the request, and saves it to storage. Finally, it redirects back.

```
<?php  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Auth;  
use App\Feedback;  
  
class FeedbacksController extends Controller  
{  
    /**  
     * Store a newly created resource in storage.  
     *  
     * @param \Illuminate\Http\Request $request  
     * @return \Illuminate\Http\Response  
     */  
    public function store(Request $request)  
    {  
        if(!Auth::guest())  
        {  
            $user_id=auth()->user()->id;  
        }  
        else  
            $user_id=0;  
        $feedback=new Feedback();  
        $feedback->user_id=$user_id;  
        $feedback->name=$request->name;  
        $feedback->email=$request->email;  
        $feedback->body=$request->feedback;  
  
        $feedback->save();  
  
        return redirect()->back();  
    }  
}
```

Finally, register-login operations are made easily as they are built-in laravel project. We just slightly modified it. “Date of birth” option was added. Also, you can notice the popup window, that was created using Modal class in bootstrapping.

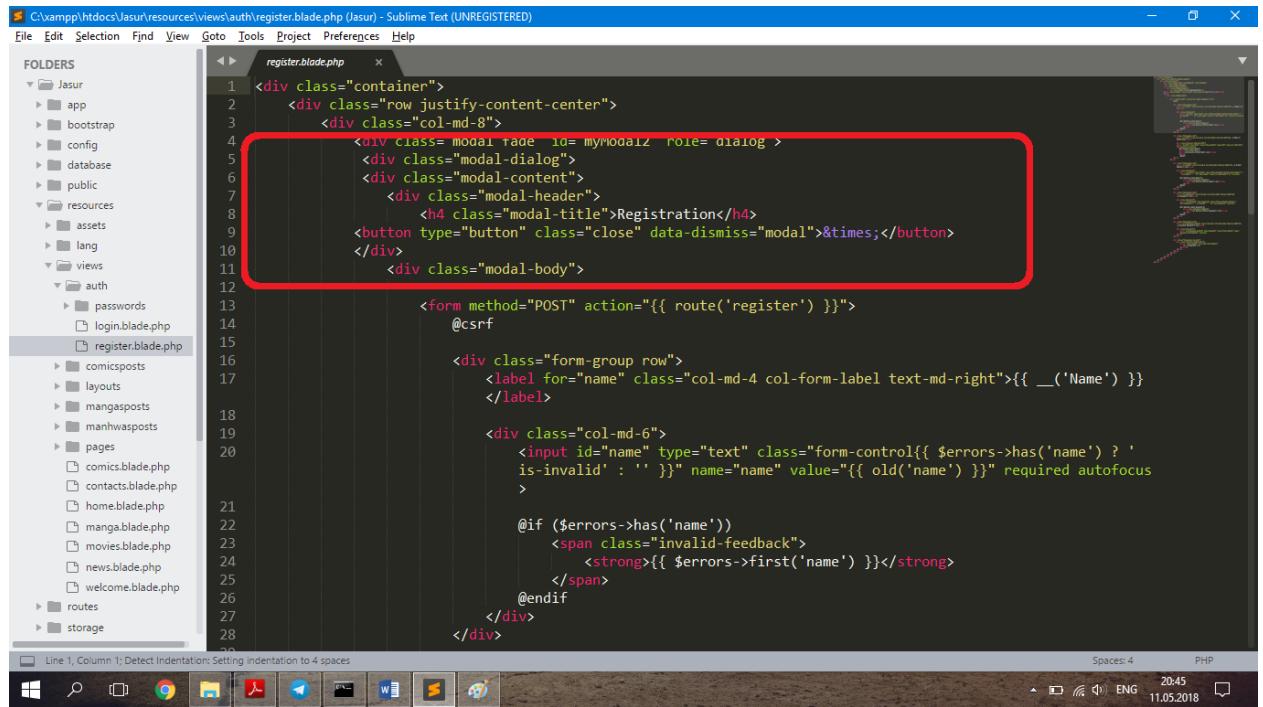
Date of birth:



The screenshot shows the Sublime Text editor with the file `register.blade.php` open. It displays a modal dialog for registration. A red box highlights the `<input id="dob" type="date" ...>` line, which is part of a form group for the date of birth. The code also includes validation logic for the name and email fields.

```
<div class="container">  
    <div class="row justify-content-center">  
        <div class="col-md-8">  
            <div class="modal fade" id="myModal2" role="dialog">  
                <div class="modal-dialog">  
                    <div class="modal-content">  
                        <div class="modal-header">  
                            <h4 class="modal-title">Registration</h4>  
                            <button type="button" class="close" data-dismiss="modal">&times;</button>  
                        </div>  
                        <div class="modal-body">  
                            <form method="POST" action="{{ route('register') }}>  
                                @csrf  
  
                                <div class="form-group row">  
                                    <label for="name" class="col-md-4 col-form-label text-md-right">{{ __('Name') }}</label>  
                                    <div class="col-md-6">  
                                        <input id="name" type="text" class="form-control{{ $errors->has('name') ? ' is-invalid' : '' }}" required autofocus>  
  
                                        @if ($errors->has('name'))  
                                            <span class="invalid-feedback">  
                                                <strong>{{ $errors->first('name') }}</strong>  
                                            </span>  
                                        @endif  
                                    </div>  
                                </div>  
  
                                <div class="form-group row">  
                                    <label for="dob" class="col-md-4 col-form-label text-md-right">{{ __('Date of Birth') }}</label>  
                                    <div class="col-md-6 text-md-left">  
                                        <input id="dob" type="date" class="form-control" name="dob" value="{{ old('dob') }}" required autofocus>  
                                        @if ($errors->has('dob'))  
                                            <span class="help-block">  
                                                <strong>{{ $errors->first('dob') }}</strong>  
                                            </span>  
                                        @endif  
                                    </div>  
                                </div>  
  
                                <div class="form-group row">  
                                    <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>  
                                    <div class="col-md-6">  
                                        <input id="email" type="email" class="form-control{{ $errors->has('email') ? ' is-invalid' : '' }}" name="email" value="{{ old('email') }}>  
                                    </div>  
                                </div>  
                            </form>  
                        </div>  
                    </div>  
                </div>  
            </div>  
        </div>  
    </div>  
</div>
```

Modal addition:



The screenshot shows the 'register.blade.php' file in Sublime Text. A red box highlights the modal registration code from line 4 to line 28. The code defines a modal dialog with a title, close button, and form fields for name and password.

```
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="modal fade" id="myModal" role="dialog">
                <div class="modal-dialog">
                    <div class="modal-content">
                        <div class="modal-header">
                            <h4 class="modal-title">Registration</h4>
                        <button type="button" class="close" data-dismiss="modal">&times;</button>
                        </div>
                        <div class="modal-body">

                            <form method="POST" action="{{ route('register') }}>
                                @csrf

                                <div class="form-group row">
                                    <label for="name" class="col-md-4 col-form-label text-md-right">{{ __('Name') }}
```

Summary of each member's contribution to the project:

Overall, the project was fairly done by whole group. There were many minor additions and changes by each member, but here are the main contributions of every person:

Nodirjon Utkirov: started the project, created the layouts, worked on css, backgrounds

Jasur Turaev: register-login models, API from YouTube

Saidazimkhon Jasurbekov: created routes models for comics' migrations

Ibrohim Orifjonov: extended comics to manga and manhwa

Farrukh Tulkunov: comments, pagination, favorite, and some changes to layouts

You can check site in authorized view (profiles):

1) Login: jasur.turayev@mail.ru

 Password: definition

2) Login: ftulkunov@gmail.com

 Password: qwerty