

Team:



"Power You Future"

Project: Announcement Board

Team Members

Tair	Djanibekov	U1910192
Shokhzod	Murodov	U1910251
Abboskhon	Tursunov	U1910230

[GitHub Repository](#)

[Web Site](#)

Content:

Brief description of the company	2
The project details	3
Contribution to the project	7
Database schema	15
Logins and Passwords	16
Reference	17

Brief description of the company

As part of the "enin" we completely assured that the future is behind the IT technologies. We focus on creating web products for either commercial or other purposes. We aimed to bring the customer's ideas to life, make them meaningful and stable.

For this project, we create a web application for announcing the news and orders on the board. The board is made for the small company which requested the application in which the users and employees can post the announcements that might be viewed by their colleagues. The board keeps the announcement in the database so that every announcement is official and has value, and might be considered as documents. The announcement might be of different types.

Info about Customers and Workers:

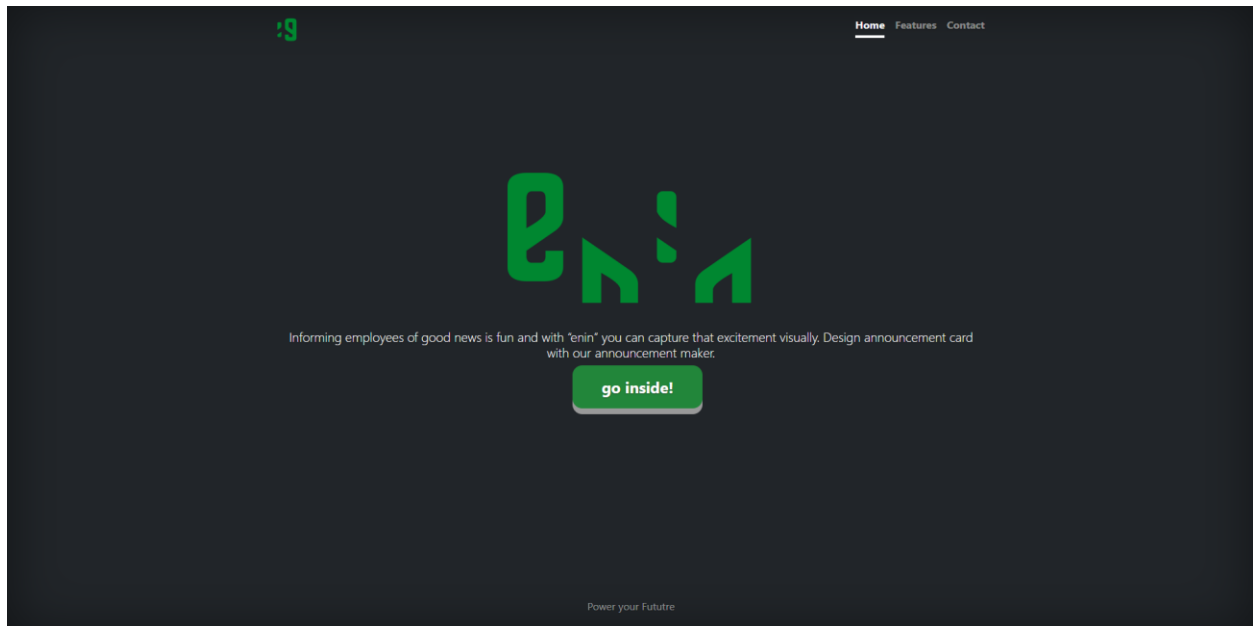
Audience (customers): Employees of the company can post the announcements under different categories and view the announcements of the colleagues. Despite registration and login, an employee can view their announcement in their profiles and view the profiles as well as the announcements of the colleagues.

Workers: Moderators can manage the content of the board, remove unnecessary or irrelevant announcements, edit and reorder the placement of the announcement in the board. Also, since the creation of the announcement, the moderator can approve the content so that it will be visible for all employees or disapprove by deleting unwilling announcements.

The project details

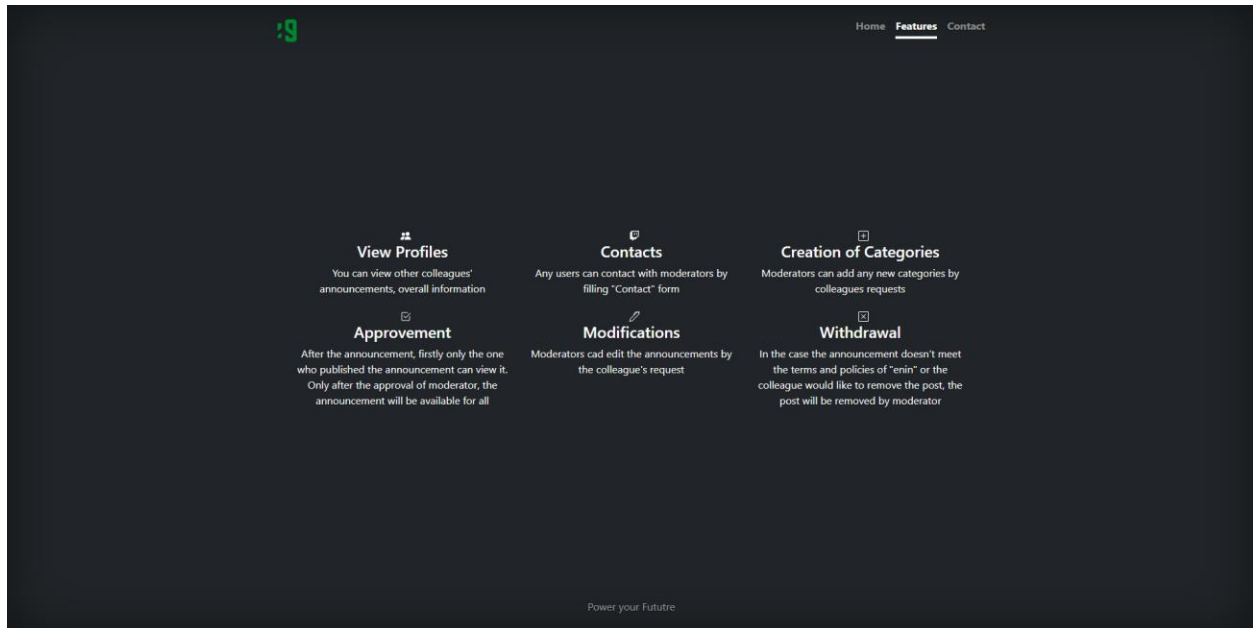
Business Scenarios: Announcements board – users can post their announcements under different categories, moderators can modify, reorder or remove announcements.

Company Landing Page (index page)



Initial landing page for the project (landing / [home.blade.php](#))

The page made by using simple html with bootstrap capabilities. The Page include the navigation bar. Since the Laravel provided the blade template the nav bar is located at separated blade.php file and include to the landing.home page. All pages and partials in the landing folder extends the master_landing.blade.php. Must be notices that this is applied also for the other directories namely for the board's and moderator's pages. By pressing "go inside" button the login page will be opened and ask to authorize the users before going to the board.



Feature's page (landing / features.blade.php)

After clicking the "Features" in the navigation bar, the corresponding page with the features of the project will be inside. The same appears with "Contact". Find the screenshots and details about implementation below in subsequent paragraph.

The routes for the landing page are not protected by the middleware so might accessed without authorization.

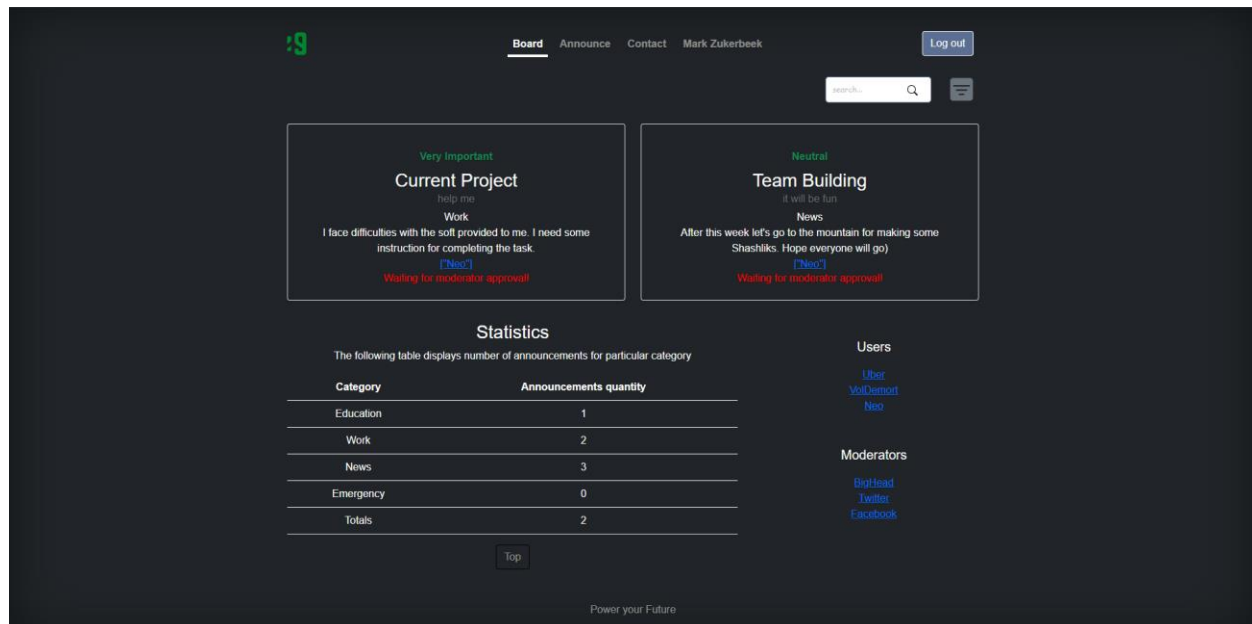
```
Route::get( uri: '/', function () {
    return view( view: 'landing.home' );
})->name( name: 'landing.home' );

Route::get( uri: '/features', function () {
    return view( view: 'landing.features' );
})->name( name: 'landing.features' );

Route::match([ 'get', 'post' ], uri: '/contact', [ ContactController::class, 'contactUsLanding' ]) -> name( name: 'landing.contact' );
```

After clicking the "go inside" the route("board.board") will be accessed but since it is protected by the middleware "auth" the user should first authorize themselves

Services Page:



Page for announcement board (board / board.blade.php)

The announcements card might be searched by the title, and filtered by the key where first will be new or old announcements. The board contains of announcements represented as cards, table which shows the quantity of announcement of particular category, list of users, and list of moderators. By clicking on user or moderator either in the announcement card (represented as username) or in the list of user and moderator, employee can get profile with the information about the chosen user or moderator.

```
Route::get( uri: '/board', [AnnouncementController::class, 'index'])->name( name: 'board.board');
```

The route calls the function 'index' which implements the getAll() function (initialized in the UsersRepository, AnnouncementRepository, and CategoryRepository). Eventually the function return a view board.board with corresponding objects. Using the x-announcement and iterate the corresponding object, the all the announcements will be displayed.

Each card was implemented using the Announcement component, where the user using the Auth facade is checked whether he is moderator or not, since the board.board is accessed by function of AnnouncementController the corresponding objects for announcements and users are provided.

Page for the profile of the employee (board / profile.blade.php)

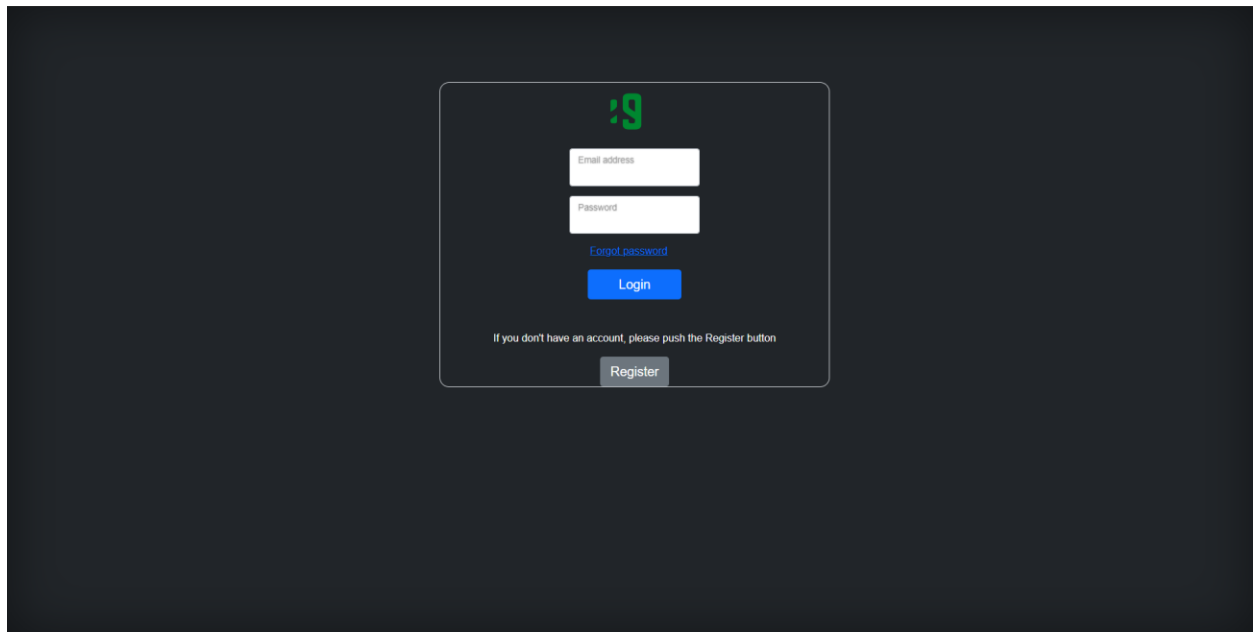
In this page the employee can view and edit his data as well as view his announcements by simply clicking his Fullname in the navigation bar.

First let's look at corresponding route, it is called in the navigation bar when the "fullName" is clicked.

```
Route::get( uri: '/profile', [AnnouncementController::class, 'indexProfile'])->name( name: 'board.profile');
```

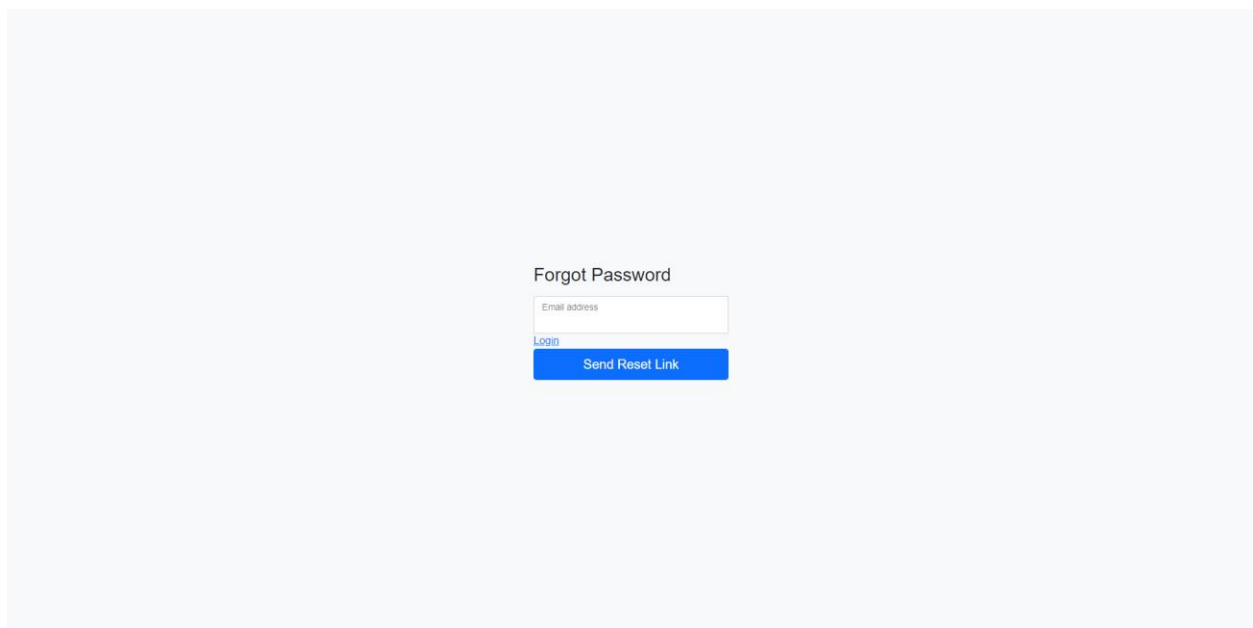
The route was assigned with the function 'indexProfile' which is method of AnnouncementController class. In this function we obtain the object of announcement, user, and categories (since these data is needed). In this class we use Auth Facade for getting the user object, and since we initialize that user can have multiple announcement namely by the function announcement() (hasMany inside) we can get the announcement of particular user in this case the user object which was obtained by the Auth Facade. So, we got all necessary information, and in the profile.blade.php using the Auth Facade we get the entities of the user, and by iterating usersAnnouncemetn populate separately each announcement of the registered user.

Registration and “Sign In” Pages

A screenshot of a login page with a dark background. In the center, there is a white rounded rectangle containing a green logo at the top. Below the logo are two white input fields labeled 'Email address' and 'Password'. Under the password field is a blue link that says 'Forgot password'. Below that is a blue button labeled 'Login'. At the bottom of the white box, there is a line of text: 'If you don't have an account, please push the Register button', followed by a grey button labeled 'Register'.

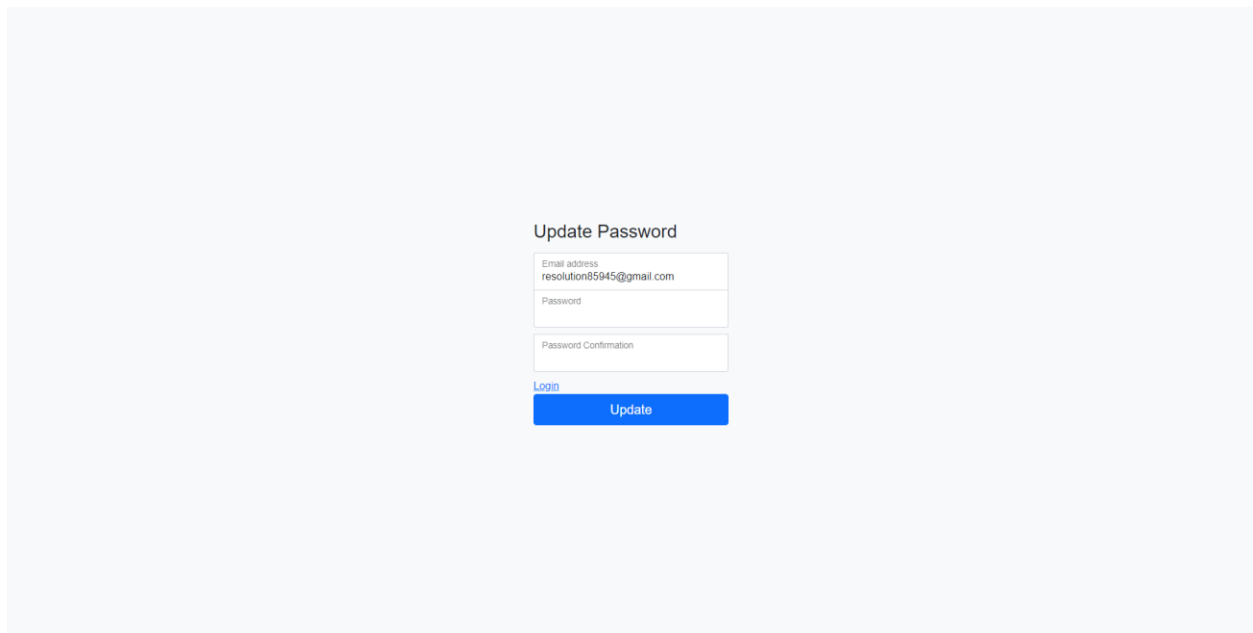
Page for Login (auth / login.blade.php)

The employee can login through by filling the form with authorized email and corresponding password and pressing “Login”. Moreover, if the user forgot the password by pressing the link “Forgot password, the corresponding page will be opened.

A screenshot of a 'Forgot Password' page with a light grey background. The title 'Forgot Password' is centered at the top. Below it is a white input field labeled 'Email address'. Under the input field is a blue link that says 'Login'. Below that is a blue button labeled 'Send Reset Link'.

Page for Forgot Password (auth / forgot_password.blade.php)

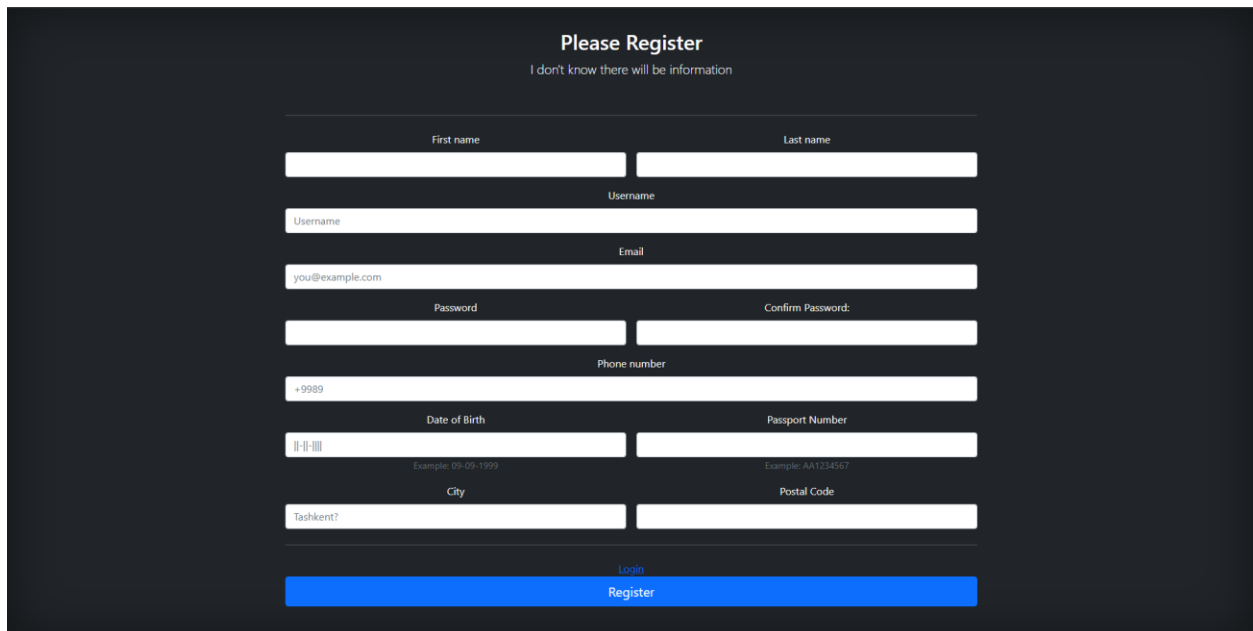
The employee should input the email for which the password will be reset. After clicking the “Send Reset Link” the link for reset password will be provided by sending the email message to the employee’s email.



The screenshot shows a web form titled "Update Password" centered on a light gray background. The form contains three input fields: "Email address" with the value "resolution85945@gmail.com", "Password", and "Password Confirmation". Below these fields is a small blue link labeled "Login" and a prominent blue button labeled "Update".

Page for Reset Password (auth / reset_password.blade.php)

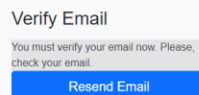
After clicking the button “Reset Password” user will be redirected to the page above where he will be required to provide new password and confirm it. After pressing “Update” the employee will be redirected to the board.



The screenshot shows a dark-themed registration form titled "Please Register" with the subtitle "I don't know there will be information". The form includes several input fields: "First name" and "Last name" (split), "Username", "Email" (with the example "you@example.com"), "Password" and "Confirm Password:" (split), "Phone number" (with a "+9989" prefix), "Date of Birth" (with a calendar icon and example "09-09-1999"), "Passport Number" (with example "AA1234567"), "City" (with example "Tashkent?"), and "Postal Code". At the bottom, there is a small blue link labeled "Login" and a large blue button labeled "Register".

Page for Register (auth / register.blade.php)

New users should register by providing necessary information. The all fields are validated as required. After filling the space and pressing "Register" the site will verify the provide email by loading the page below.



Page for Email Verification (auth / verify_email.blade.php)

The user required to press "Resend Email" button, so that the corresponding email will be send to the provided email. There by pressing the "Verify Email" user authenticate his email and prove that his email is valid and exist.

For the authorization functionality we modified the FortifyServiceProvider, used given by the Fortify routes, enable features in the config/fortify.php as well as add App\Providers\FortifyServiceProvider::class in the config/ app. In the route service provider, we initialize HOME as /board. We add some routes to the group and protect them by the auth and verified.

```
Route::group([
    "middleware" => ["auth", "verified"]
], function () {

    Route::get( uri: '/board', [AnnouncementController::class, 'index']->name( name: 'board.board'));

    Route::get( uri: '/announce', [AnnouncementController::class, 'announcement']->name( name: 'board.announce'));

    Route::get( uri: '/profile', [AnnouncementController::class, 'indexProfile']->name( name: 'board.profile'));

    Route::match(['get', 'post'], uri: '/contact8', [ContactController::class, 'contactUs']->name( name: 'board.contact'));

    Route::get( uri: '/profileForeign/{id}', [AnnouncementController::class, 'foreign']->name( name: 'profileForeign'));

});
```

Create Announcement Page:

Page for creating new announcement (board / announce.blade.php)

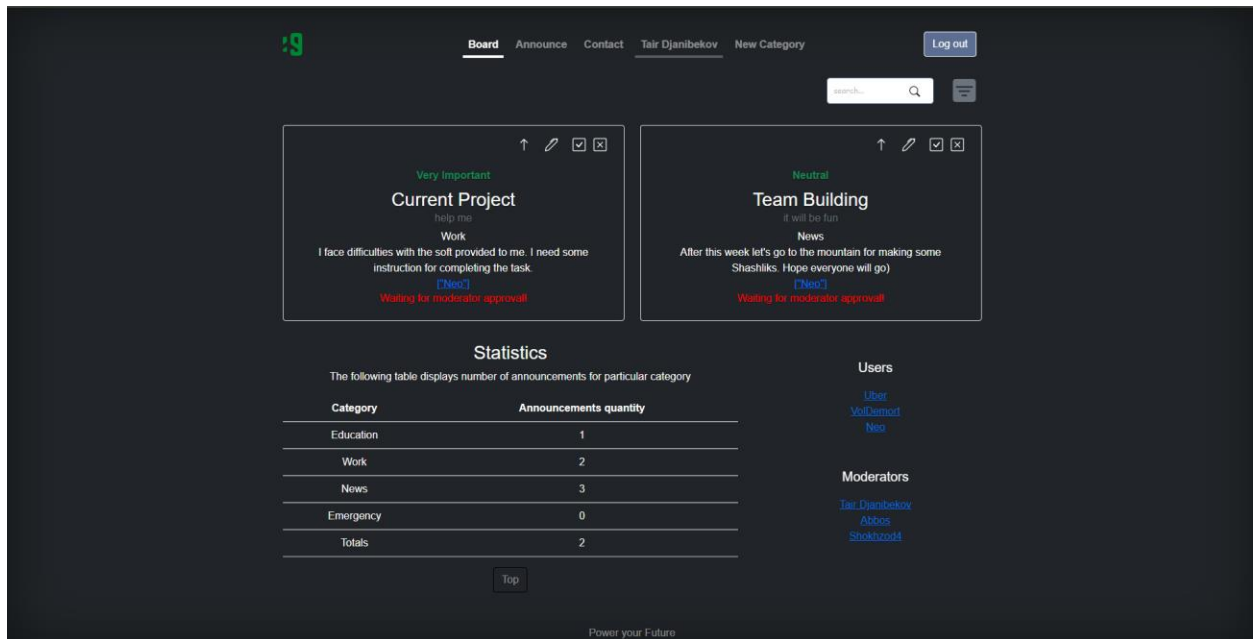
By filling the form and pressing “Announce” user will create announcement which will be displayed in the board only for the him and moderators who can approve the announcement which will make announcement visible for all users.

The page is accessed by the route below. The route call the function ‘announcement’ the method of

```
Route::get( uri: '/announce', [AnnouncementController::class, 'announcement']->name( name: 'board.announce');
```

AnnouncementController class. In this function, all categories are obtained through the CategoryRepository and assigned to the object categories which is parsed to the board.announce.

Process the Announcement (Moderator) Page:



The screenshot shows the 'Board' page for a moderator. The top navigation bar includes 'Board', 'Announce', 'Contact', 'Tair Djanibekov', 'New Category', and a 'Log out' button. A search bar is located on the right. The main content area features two announcement cards. The first card, titled 'Current Project', is marked 'Very Important' and contains text about work difficulties. The second card, titled 'Team Building', is marked 'Neutral' and contains text about a mountain trip. Both cards have a 'Waiting for moderator approval' status. Below the cards is a 'Statistics' section with a table showing announcement counts by category. To the right of the statistics is a list of 'Users' and 'Moderators'. A 'Top' button is at the bottom of the statistics section.

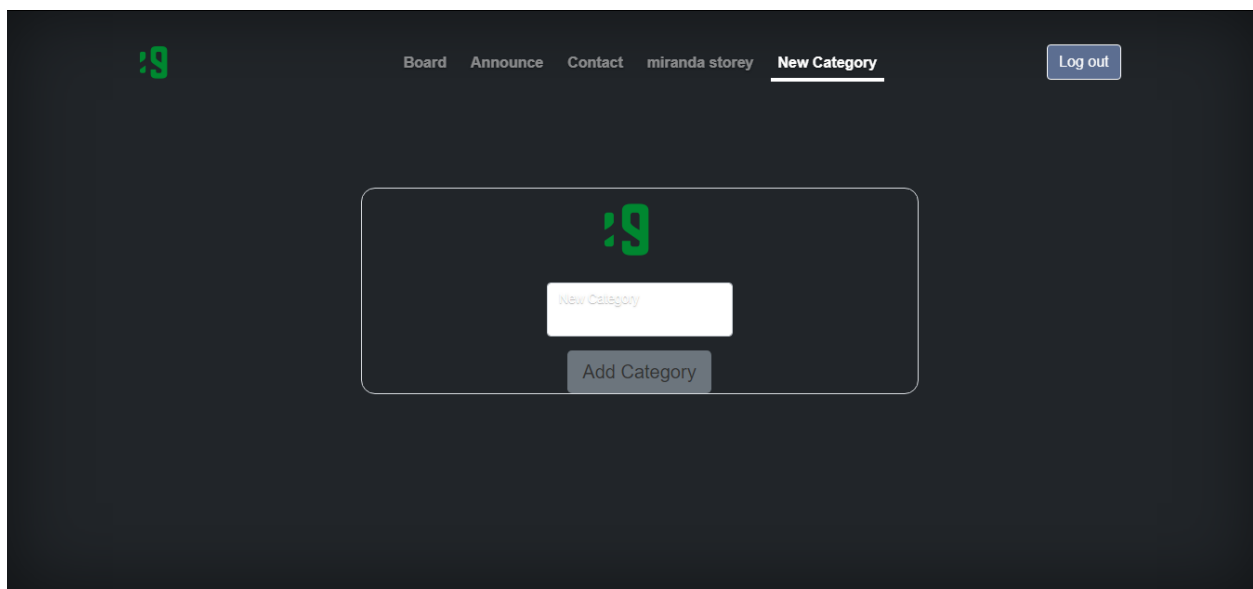
Category	Announcements quantity
Education	1
Work	2
News	3
Emergency	0
Totals	2

Users: User, VolDement, Neo

Moderators: Tair Djanibekov, AlKos, Shokhuzat

Page for the moderator (moderator / board.blade.php)

In the moderator page we have almost the same page as in the board for user, with the exception there are buttons in each announcement card. The first button put the announcement at first position. Second button open the page for editing the announcement, third page approve the announcement and make it visible for all employees. Last button simply deletes the announcement. Moreover, moderator has the same functionality as a simple user, like announcing and viewing. With exception moderator can edit the user.



The screenshot shows the 'New Category' page for a moderator. The top navigation bar includes 'Board', 'Announce', 'Contact', 'miranda storey', 'New Category', and a 'Log out' button. The main content area features a large green ':9' logo, a text input field labeled 'New Category', and an 'Add Category' button.

Page for adding new category (moderator / create_category.blade.php)

Moderator can create new category by simply filling the form.

So, the implementation part of buttons in the boards template mainly was using ajax requests. We tried to make things faster, and minimize numbers of reload and traffic, so for simple operation as delete, modify, and reorder, we are sending ajax requests to the server. The server processes incoming data thanks to Sanctum package, so we can use api authorization and check what type of user is sent request thanks to authorize method provided by laravel. Then, we can inform user that request was processed successfully using alert method in javascript. Also, simple api of jquery makes our life easier when we reorder announcement, delete it, and etc. Logic for announcements is written considering type of user. Only users who are moderators are able to see button for modifying announcements and link for new category is available for moderators, too.

Contact Us Page:

9

Home Features Contact

Contact

We are glad to receive something from you

Location:
Mirzo-Ulugbek,
Tashkent

Email:
enin@enin.online

Call:
+998909988409

Your Name

Your Email

Subject

Message

Send Message

Power your Future

"Contact Us" page before authorization

"Contact Us" page after authorization

The employee can contact the company through the page "Contact Us" either after authorization or before. If the user already authorized the Name and Email input space will be already populated with the data of the corresponding employee. By filling the form and pressing "Send Message" button the employee will send a message to the email of the company, namely to enin@enin.online.

Firstly, the corresponding routes were created. These routes might be accessed by get and post method.

```
Route::match(['get', 'post'], url: '/contactB', [ContactController::class, 'contactUs'])->name('board.contact');
Route::match(['get', 'post'], url: '/contact', [ContactController::class, 'contactUsLanding'])->name('landing.contact');
```

We assign the functions "contactUs" and "contactUsLanding" for the contacting through after and before authorization, respectively. Both methods are members of ContactController class. Both check whether the post or get requested was processed. For get the functions return the view for corresponding page (in the landing or in the board). For the post, both functions call the "postContact(\$request)" function where the request is validated and parsed to the method in the UserRepository – "sendContactUs", where parameter for the function is the object of ContactUs class. The class ContactUs provides us the ability to notify through the email since it is extending the Notification class. The user is notified that the email was sent eventually.

Contribution to the project

Tair Djanibekov U1910192

Created fundament for the project, by creating all blades with components and corresponding routes, add authorization through the fortification, made migrations and corresponding Models, created seeder for the categories, made Repository and Controllers for the Announcements, made email verification.

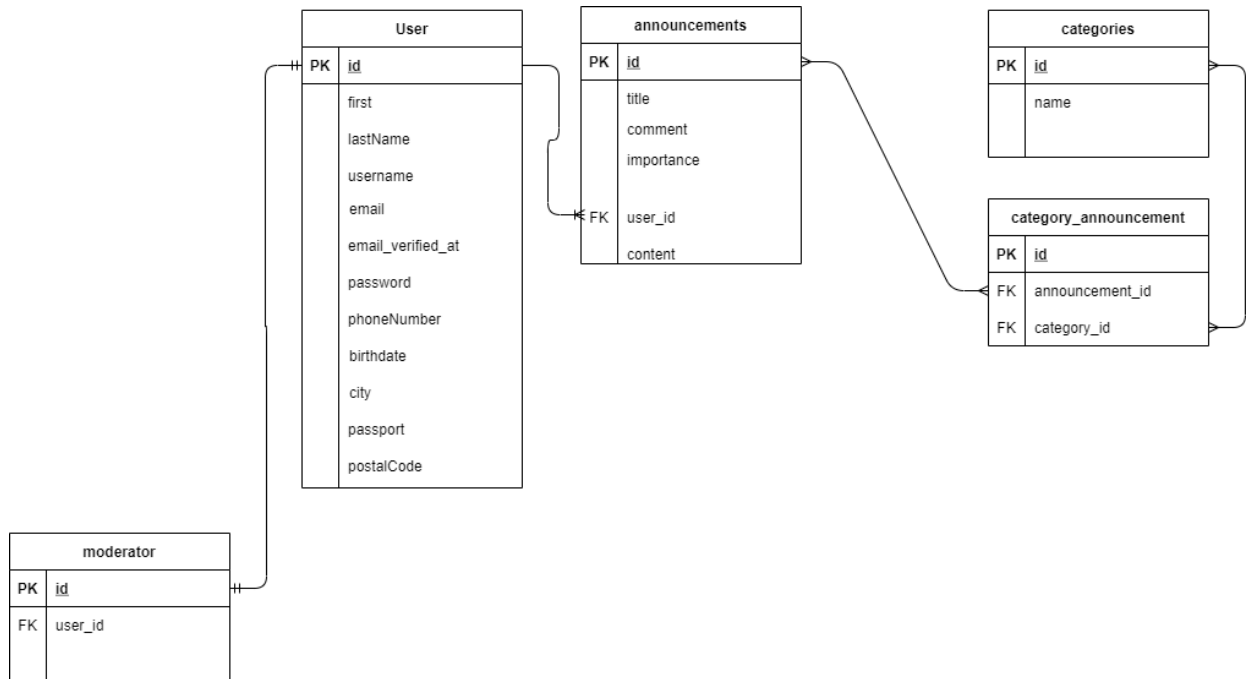
Abboskhon Tursunov U1910230

Created polices, blades, and functions for moderators, made significant contribution to the style, extended the capability of the functions in some blades, added javascript interactivity using jQuery, made notifications and "Contact Us" functionality, made Policy. Installed Sanctum for API.

Shokhzod Murodov U1910251

Add searching in the board, creating new category for the moderators, made validation for all forms, hosted the project, implemented filter (sorting) in the board, fixed bugs and problems in notifications and polices, made some changes in structure of Controllers. Resolve some major issues in the Sanctum. Resolve most of the merging conflicts.

Database schema



One to one relation between user and moderator, since we need to identify the moderators along the users. It is better than simply adding new column for every user.

One to many relation between user and announcement, one user can have multiple announcement.

Many to many relation between announcement and category, since one announcement can have multiple categories and one category can have many announcements. This resolved by creating additional table with announcement and categories id.

Logins and Passwords

We don't find free service for providing email notification, so that we used mailtrap.io

Moderator: tair.djanibekov@gmail.com password: 123Aa123

Users:

For simplicity password for all users is the same

asd@asd.ddd

shaxa882@gmail.com

asd@qwe.qwe

aaa@aaa.aaa

Actually, you can get just email from the board of any user and login through it since password is the same. Creation will lead to problems since we can not send the email to real mail, we didn't find free service for this issue

References

1. Laravel documentation
2. Bootstrap for styling and for some components
3. Some stack overflow issues