**“StudyON”** by Inanc Alp Gunalp

* ICT – Application Development
* 1st phase.
* E-mail: [r0881868@student.thomasmore.be](mailto:r0881868@student.thomasmore.be) || [inancgunalp@gmail.com](mailto:inancgunalp@gmail.com)

**Acknowledgements:**

I would like to express my graditute to Thomas More De Nayer for, giving students an important opportunity to learn new technologies & improve their skills independently.

With this excuse, we as students, can grasp the new concepts in this always changing & improving community of Software Development, by ourselves and learn the most valuable skill, self-learning, that we need to have to make our path future-proof.

Special thanks to Mr. Dieltiens for always supporting us in this journey when we needed help.

**StudyON:**

Is an interactive web application for students to input questions they have think of during class or in their study sessions. With creating their study groups and the specified courses in it, they will challenge each other for to have as much as right answers possible from the questions they have created for each other. With the questions have been inputted, they will try to earn points and rank up to earn badges or a choice of prize. Main goal of the application is to encourage students to use “active recall” technique and be more prepared for the upcoming exams.

**Problem:**

As individuals, sometimes we experience lack of motivation and get distracted which does not return the best outcomes. If we speak for the students; these outcomes can be such as, overwhelming stress and feeling of burnout, especially in the exam periods.

**Objective:**

Goal of StudyON,

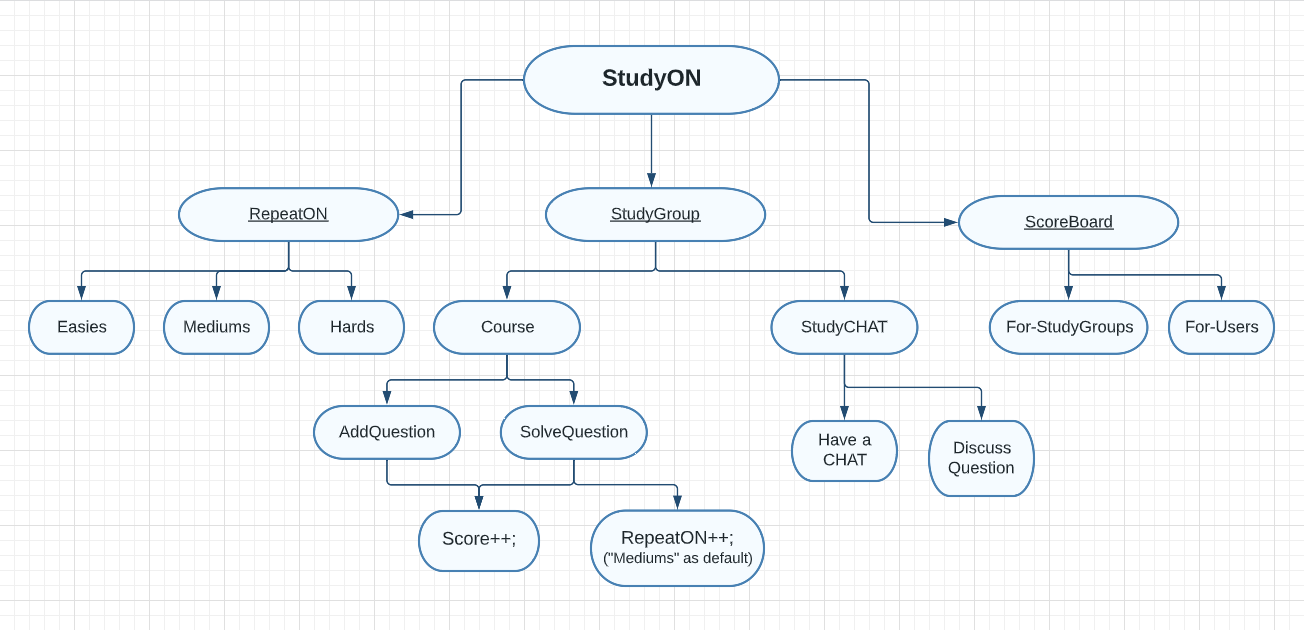
* To keep students alert enough to make them think for a question during the class hours.
* To make students interract with each other and talk more about their studies in a more casual way.
* To hack into student’s “reward system” with using scoring algorithms and make her/him look forward to find creative choices for the question she/he is planning to add.

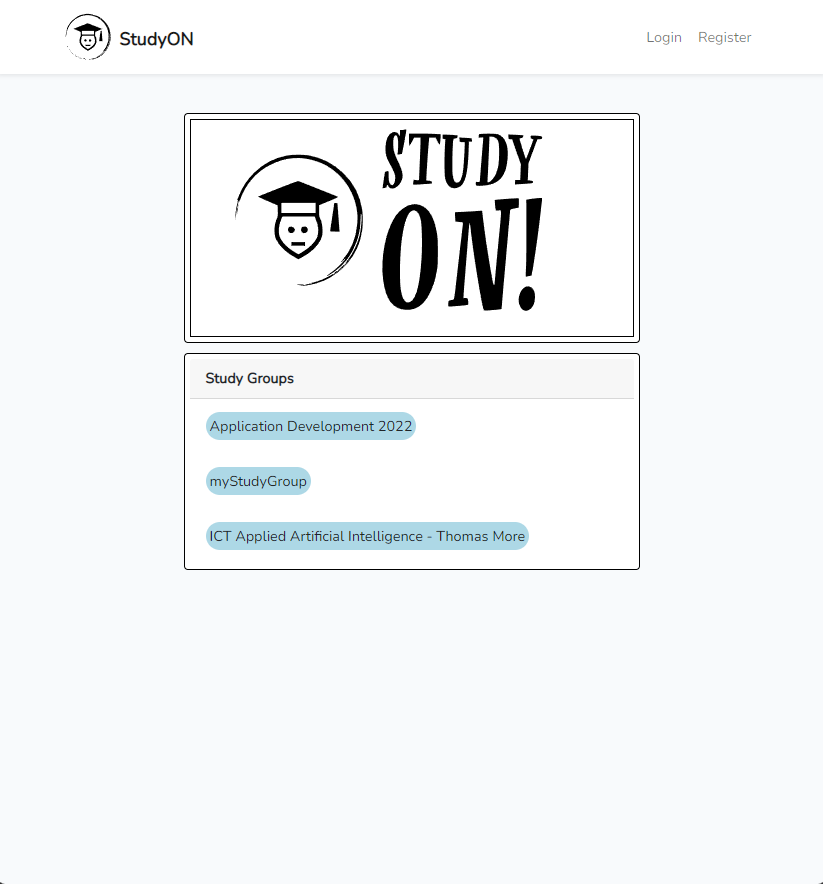
**Every drop Counts!**



**StudyON**

StudyON-BlockDiagram





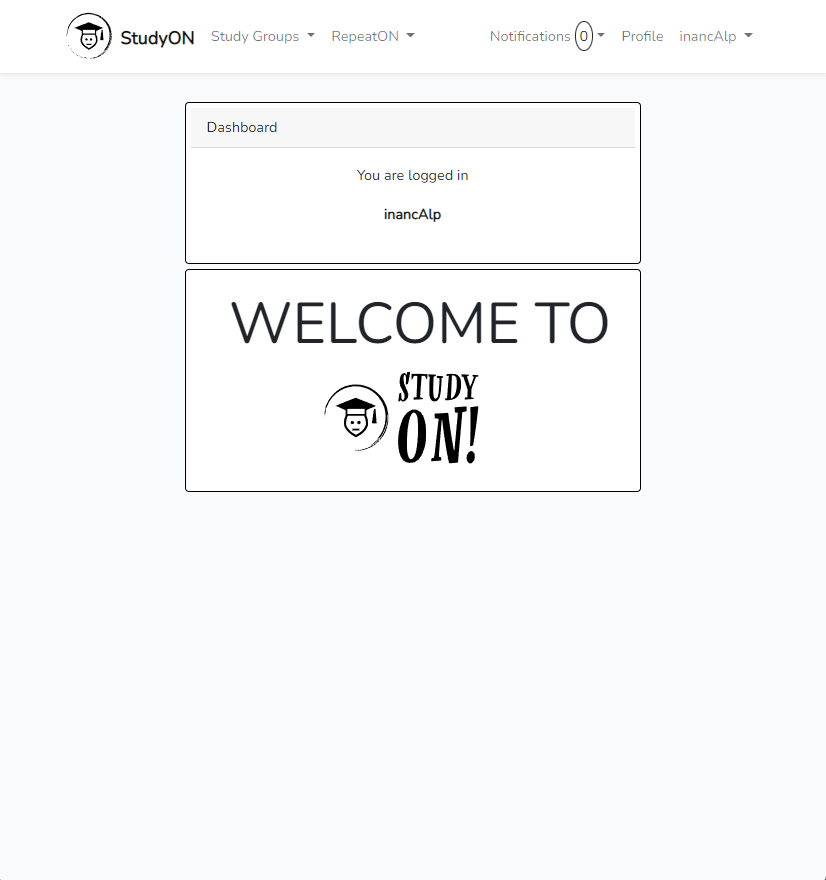
Hello, StudyON!

Graphical user interface, application

Description automatically generated

**Authenticated User**

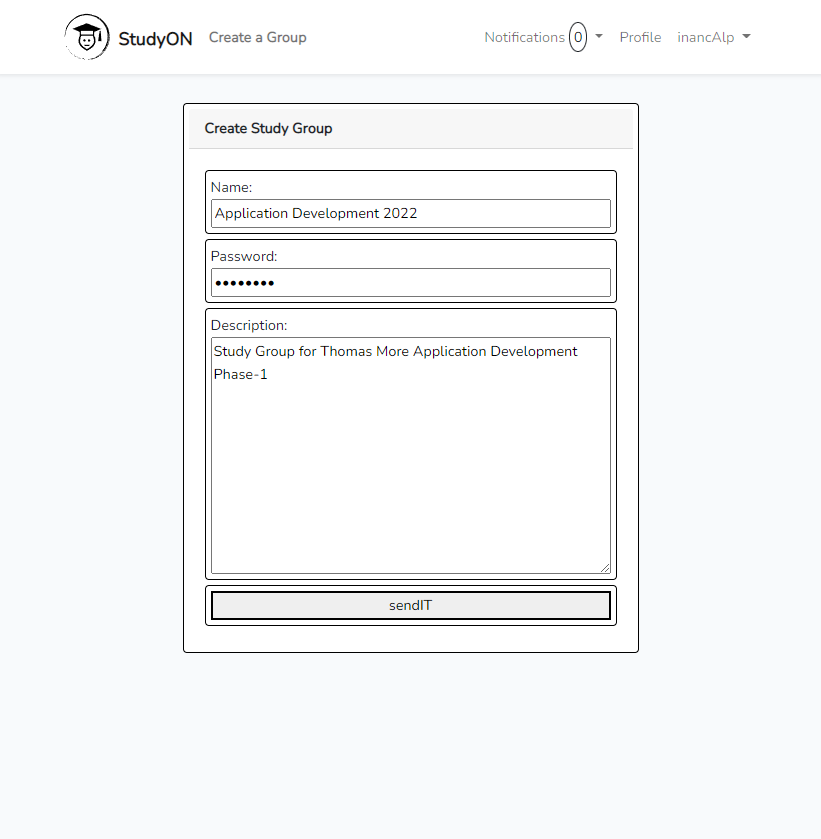
* StudyON is designed to be a user-based interactive platform.
* Users need to create account before getting involved in any of the features included.

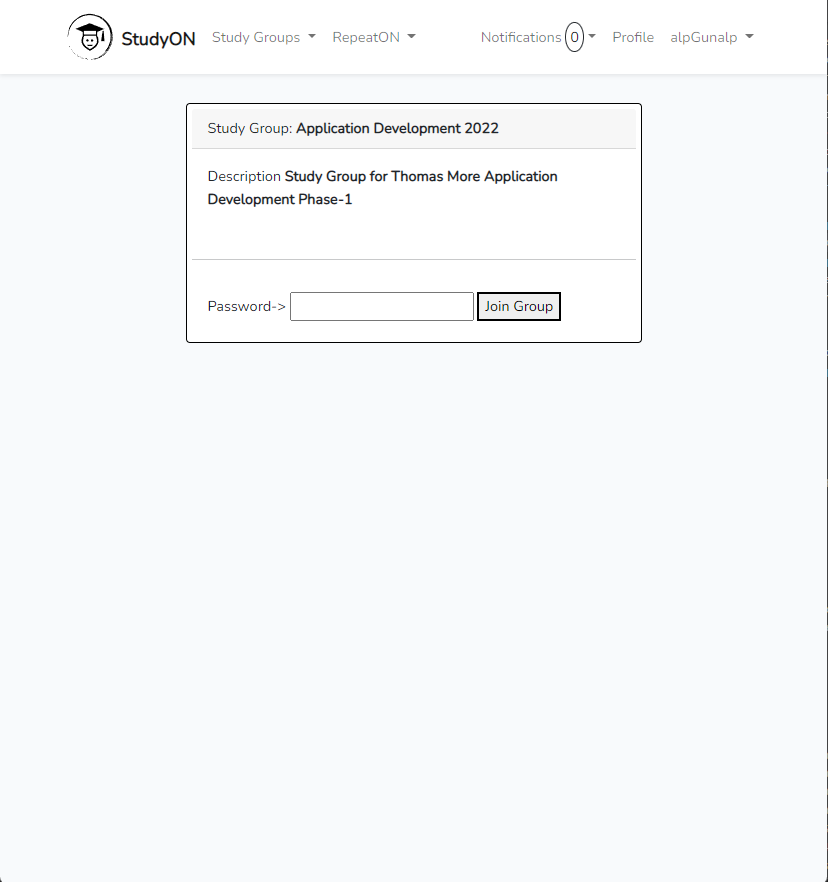


* Here above, we see the welcome page when a user is authenticated.

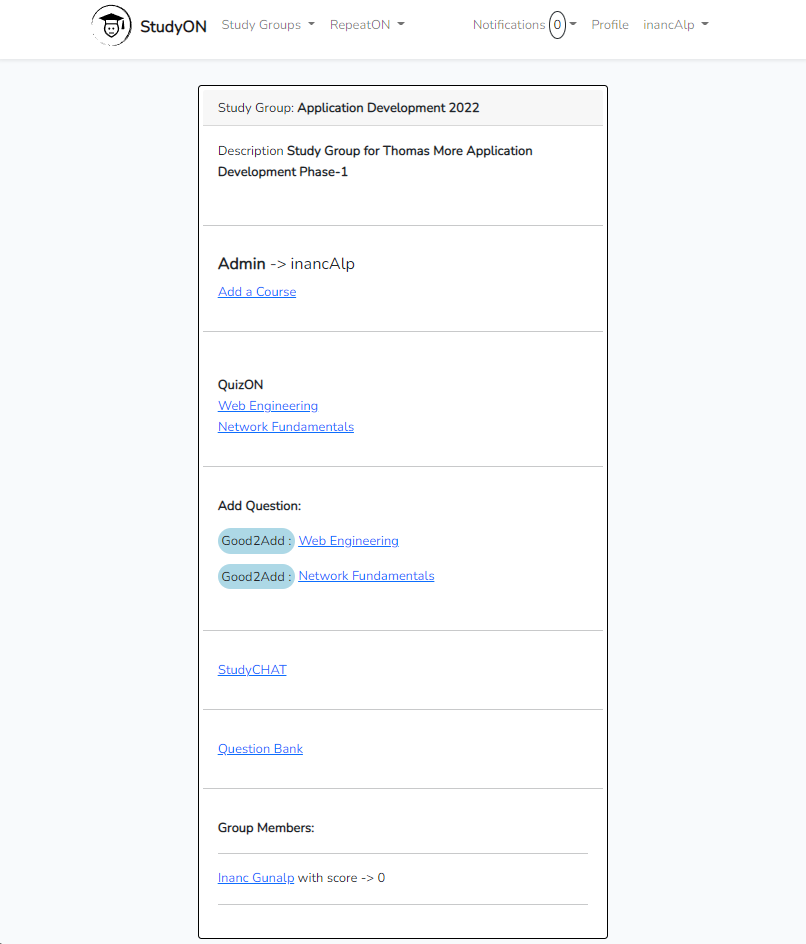
**Study Groups**

* User can either join a study group or create a new one.





* Inside of a StudyGroup, user can find most of the core concepts of StudyON.



**QuizON**

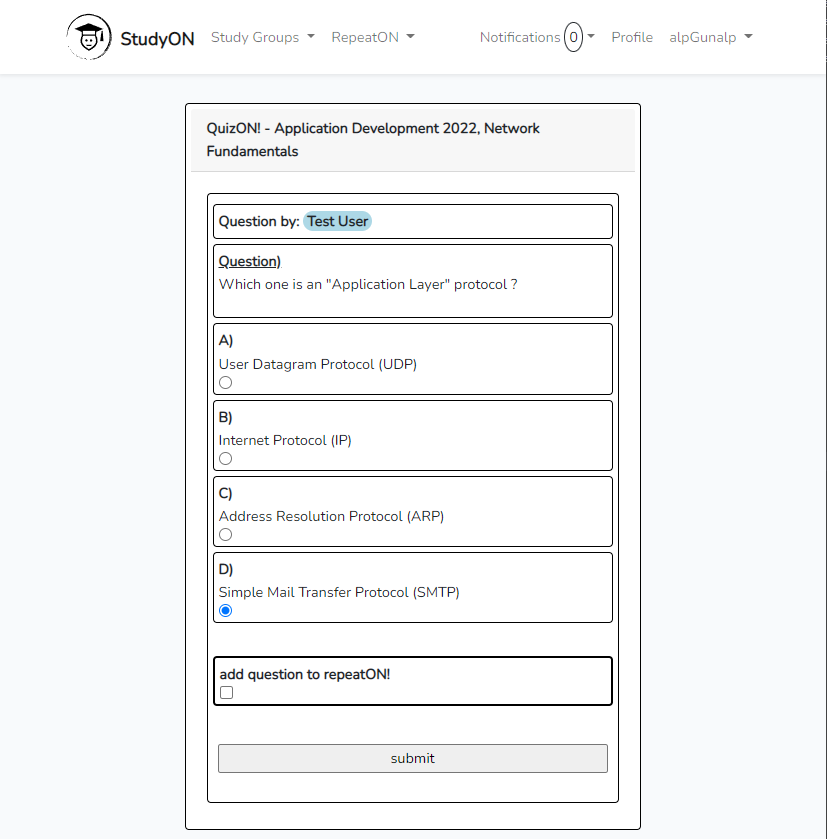
* User can add & solve questions of the week, for the releated courses.
* When a question is added, it will appear in the releated course’s QuizON page.
  + User will earn 5 points each for adding a question and for solving a question.
* Users should not wait too long to solve the questions though!
  + Questions will disappear automatically from QuizON, 1 week after being added.
  + Which means, chance to earn points from the questions will also disappear.
* But no need to worry xxx-tra much!
  + Deleted QuizON questions will have their places ready inside of the QuestionBank for all members of that one StudyGroup.

add-question

Graphical user interface

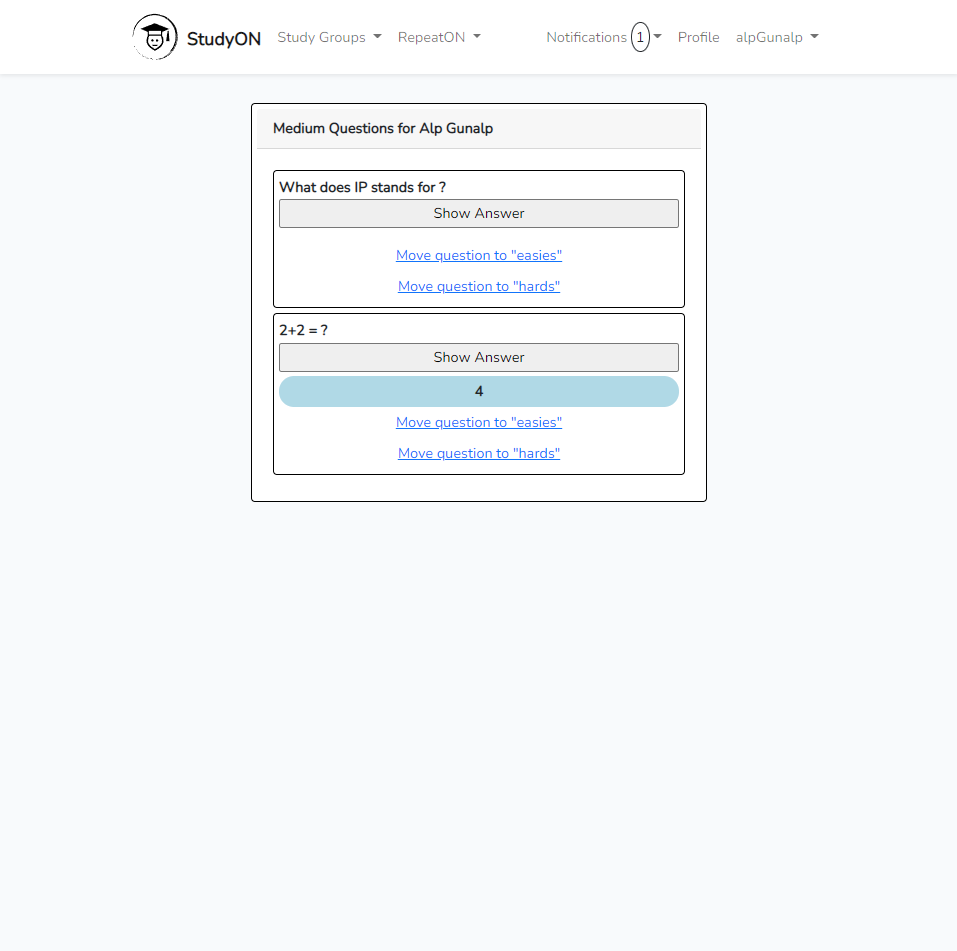
Description automatically generated

solve-question



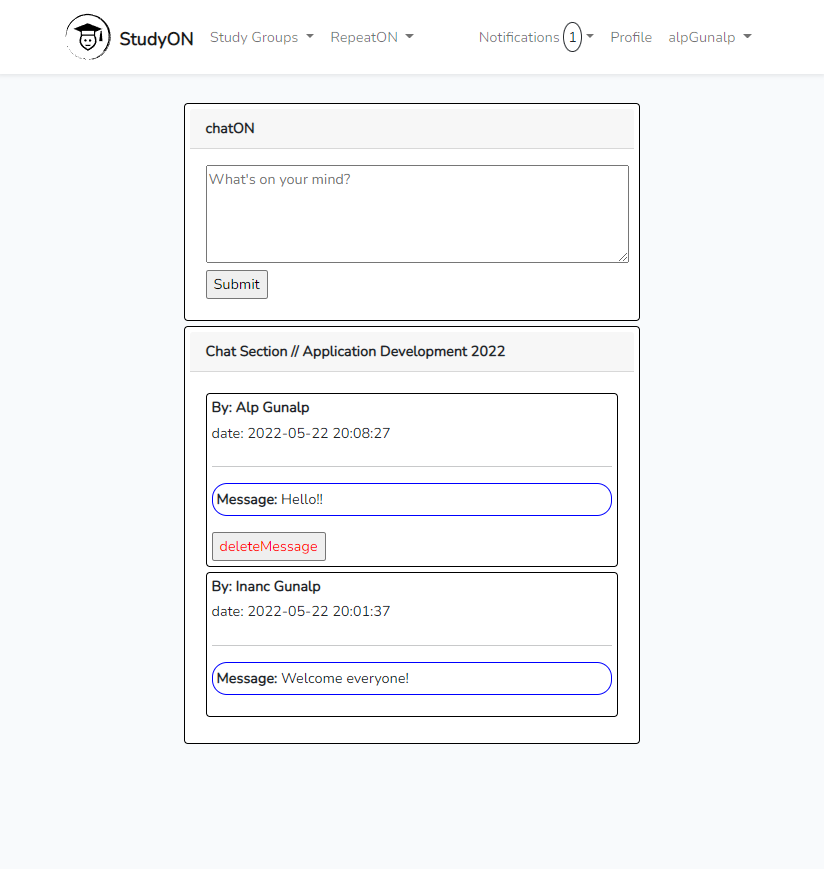
**RepeatON**

* During QuizON session, if users find any of the question interesting and want to work on it again later, they can simply add it to their RepeatON list.
  + Any question from any StudyGroup that user is a member of, can have its place in RepeatON section for user to work on any different topic collectively.
* RepeatON questions will drop into medium level list as default, but users can easily move the questions to preffered level.



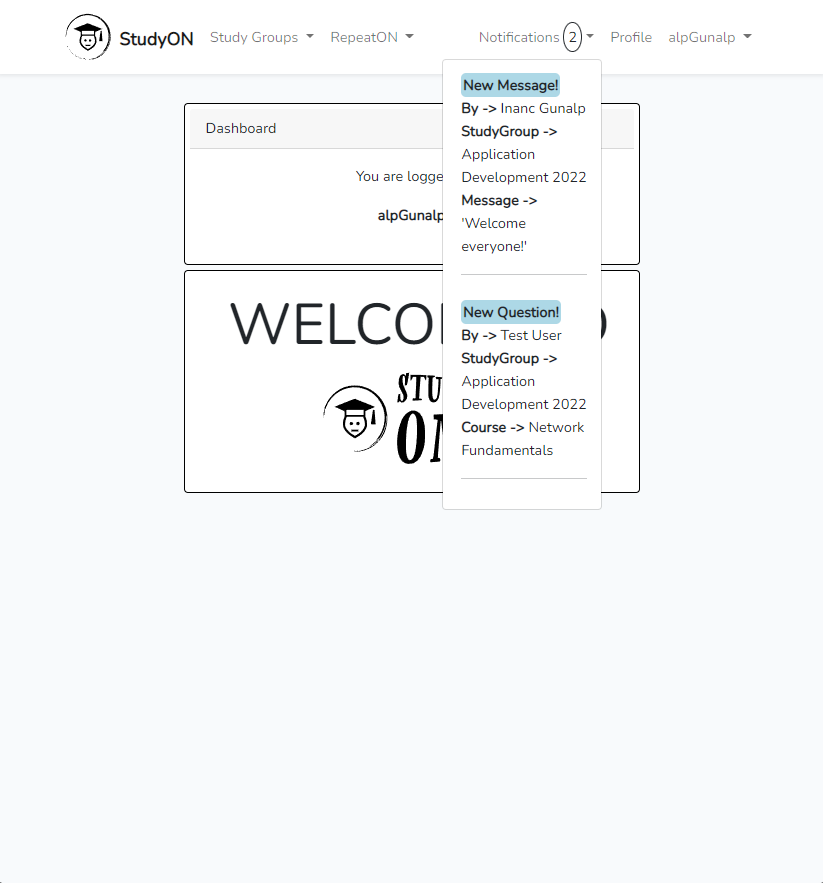
**StudyChat**

* Users can discuss about questions.
  + Or they can just have a chat!



**Notifications**

* Users will be notified for any question added & any message sent from other users, which are taking place in their StudyGroups.



**StudyON-Technical Details**

* Choice of Technology = Laravel (PHP Framework)
* Laravel comes with the design pattern called MVC.
  + Model–View–Controller

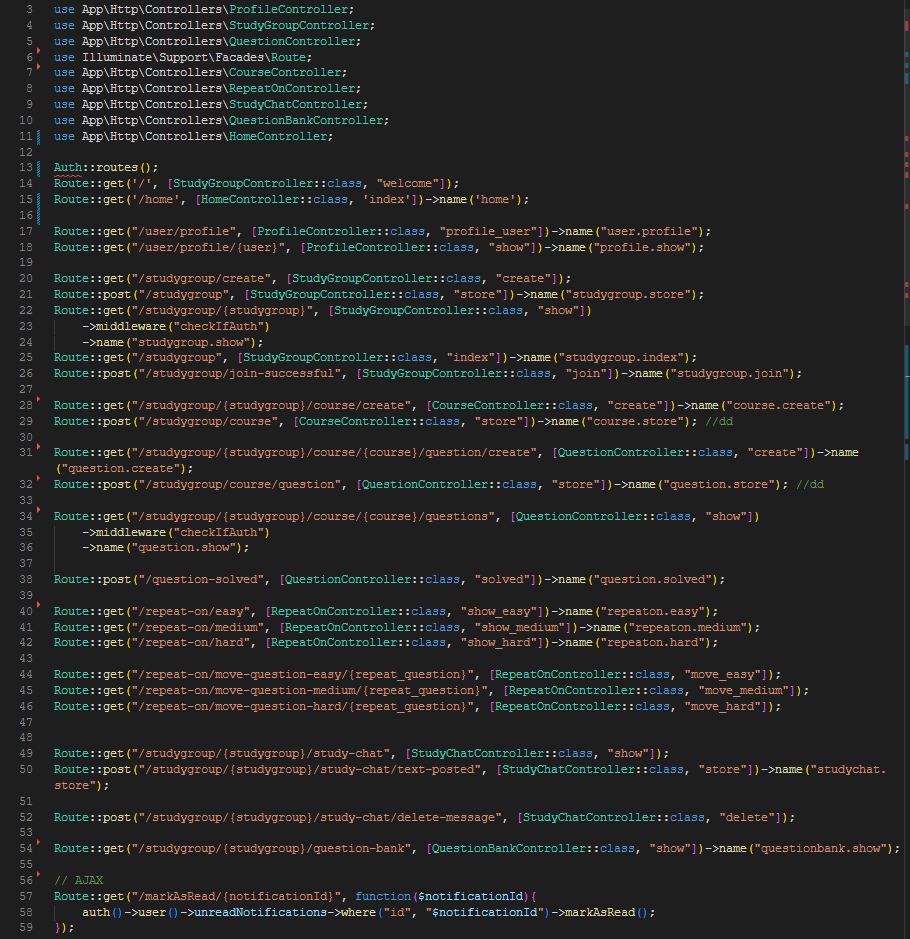
**What is MVC?**

Let’s explain this with some examples from StudyON!

Below, there are 2 examples that can be found.

* One for HTTP POST REQUEST
* One for HTTP GET REQUEST

But before the examples, you can find a brief explanation of a web.php file right below this page.



**File the “web.php”**

In Laravel, “Route” objects play one of the most important roles to keep our code tidy and clean!

* When a URI been active, releated Route method lets us to reach the necessary Class including the necessary method which decides what will happen next!
* Route reads the URI, then uses the method has been called from the releated Controller Class.
  + Route (“URI”, [ControllerClass, “controllerMethod”])->name(“routeName”)
  + “routeName” can be used as a formAction or a hyperTextReference instead of using URIs.

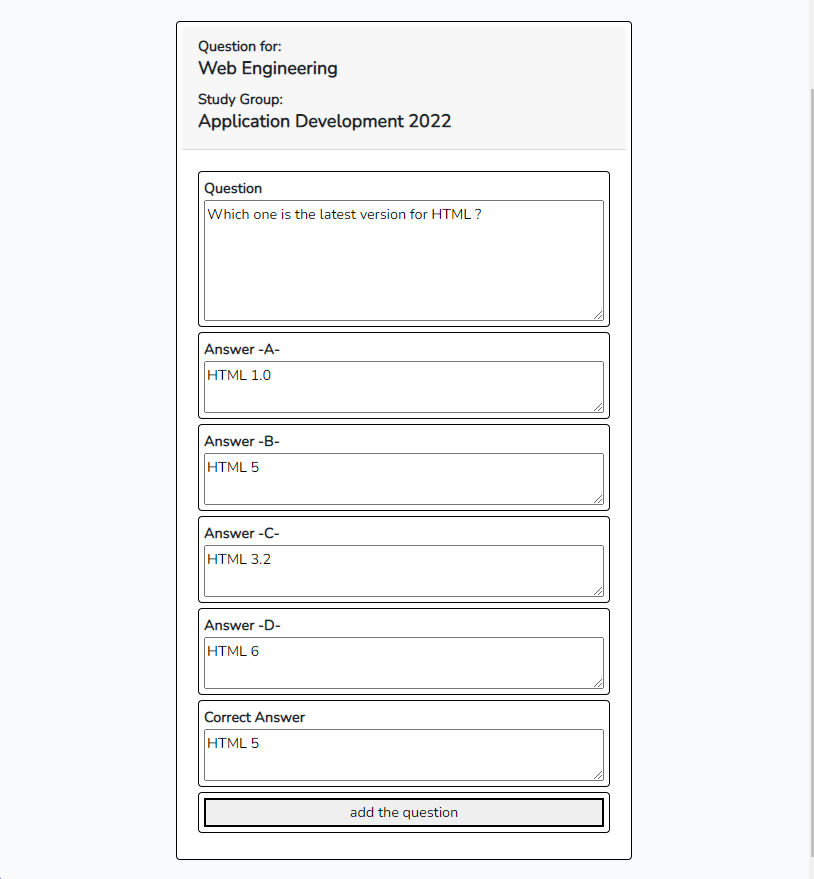
In the examples below, we will be seeing What happens when a user wants to,

* Reach a QuizON page (GET Method)
* Add a question for QuizON (POST Method)

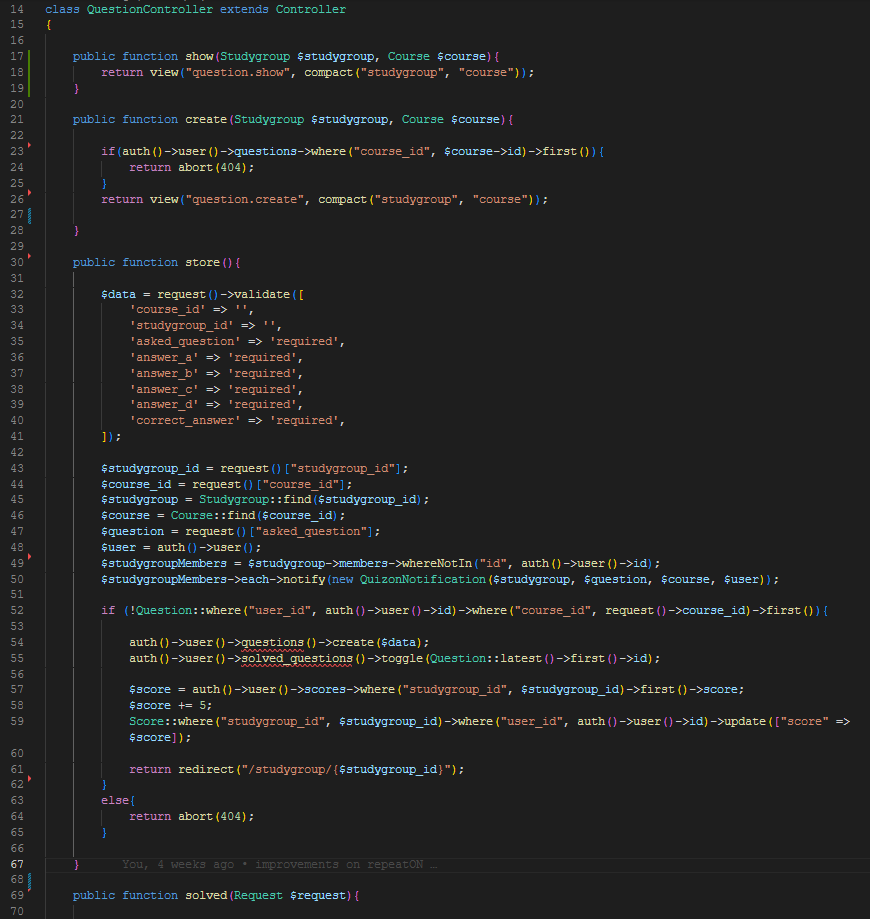
**Example POST Request**

User wants to add a question for the choice of course.

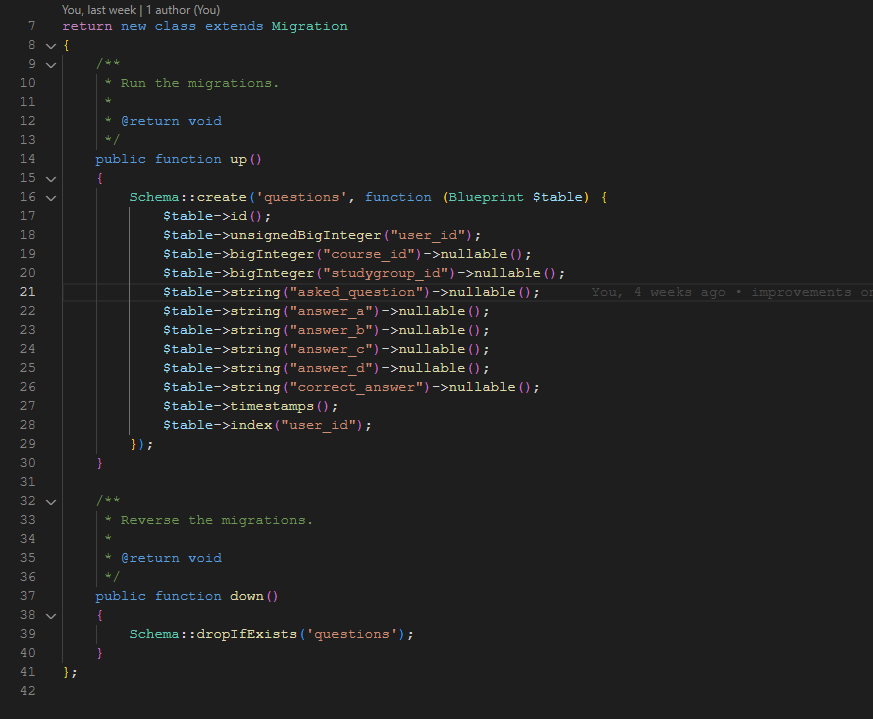
* User opens the view page.
* User writes down the necessary information for the question.



* When user submits the form, it will be directed to the releated controller method.
* As can be seen from the image below, any green colored code represents a model that interracts with the controller method.



* Studygroup Class finds the collection of data for the releated studyGroup ID from Database.
* Course Class finds the collection of data for the releated course ID from Database.
* Notice that “request()” & “auth()” are also Models that we are using.
  + Laravel gives us the freedom of choice in many aspects, even with the style of code that we want to execute!
* QuizonNotification Class*,* with using “new” in front, we can create a new entry for our Notifications table.
* Question Class conditional lets us check if user have already input a question for the releated course.
* Score Class lets us manipulate the scorePoint for the user if we lead it to right direction as seen.
* With using controller methods, we are able to reach the necessary models & update the releated database table via using our Migration BluePrints which are representing our DataBase Tables.



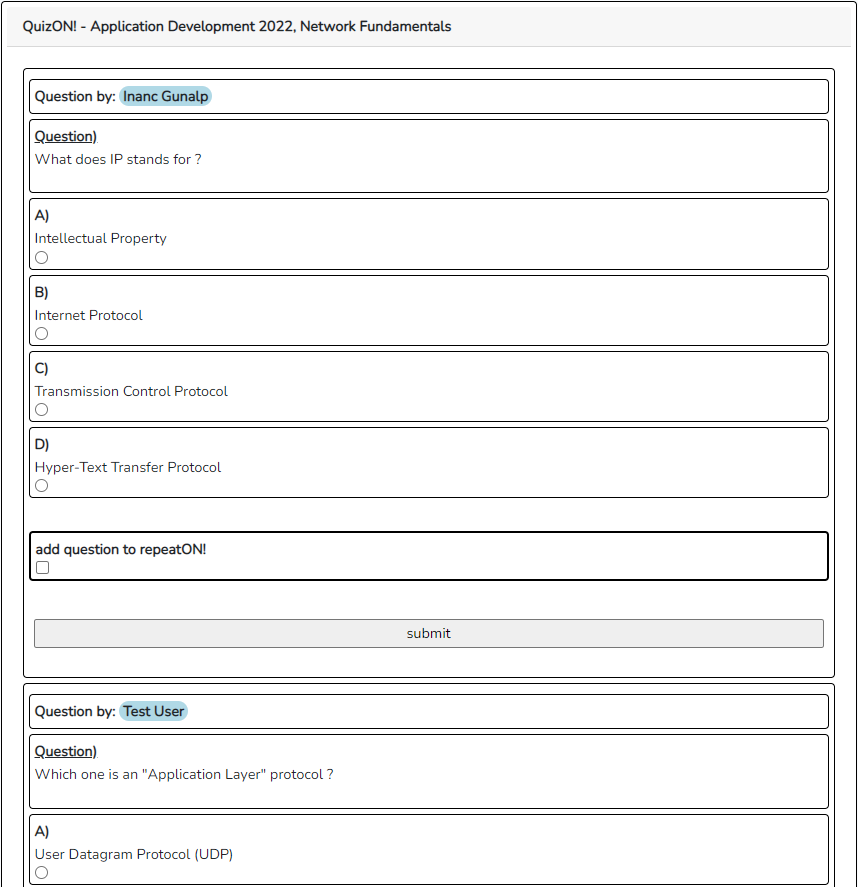
Graphical user interface, application, Word

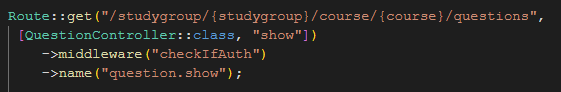
Description automatically generated

**Example GET Request**

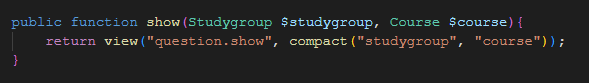
**Note!** Since we have already seen how MVC cycle works, in this example, you can find some other details could be interesting.

* User does a GET Request to reach the page below.

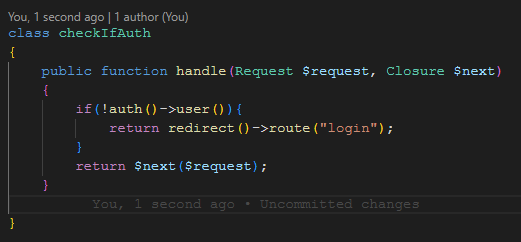




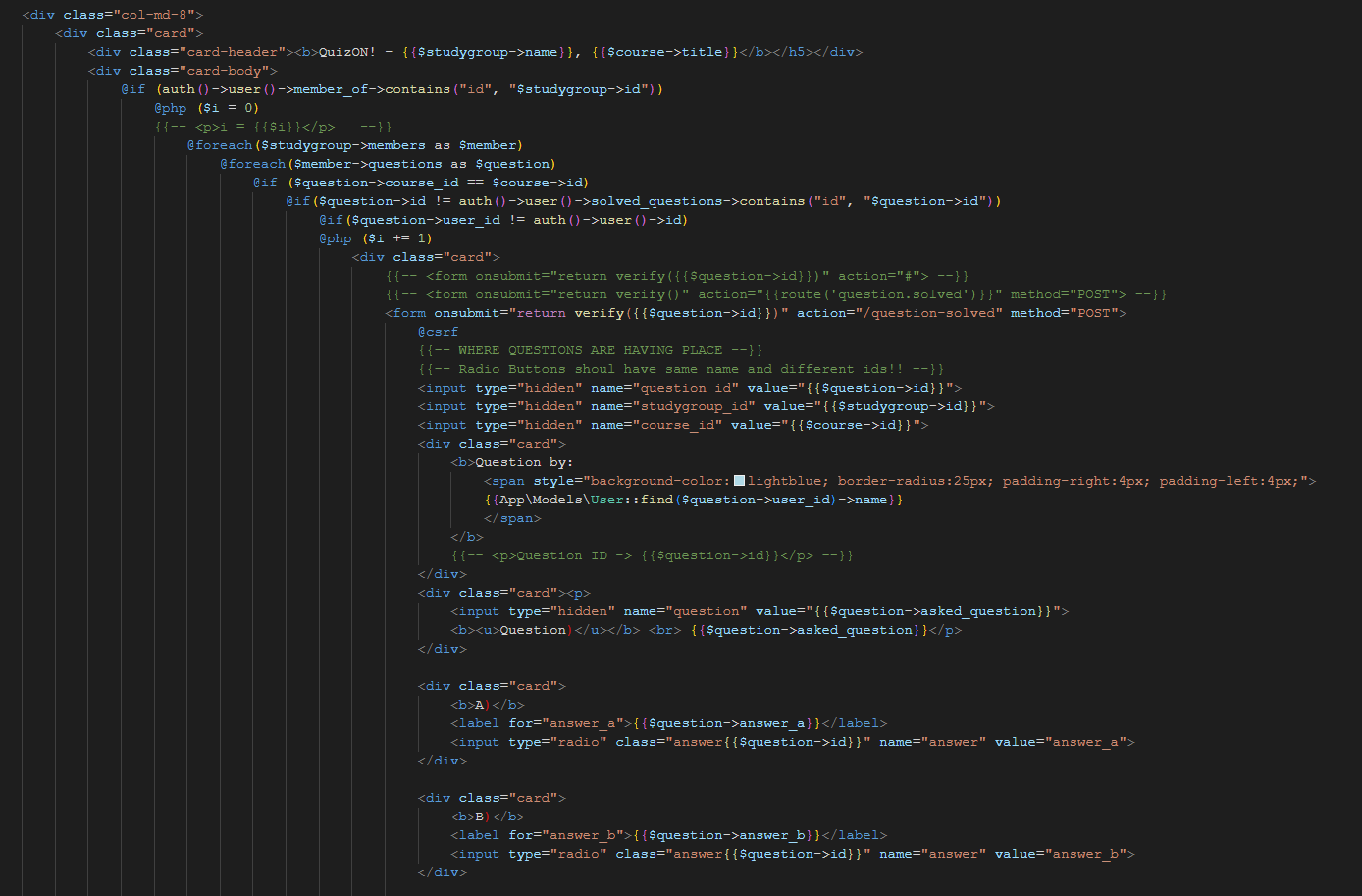
* Notice that, in URI, portions with “{~}” brackets, represent parameters that we can use in our Controller methods.
  + In this case, parameters represent IDs for releated studyGroup & course.
* Middleware method used in our Route, acts as a security object, which I will mention below with more details.



* As can be seen from image above, studygroup & course ID parameters we have passed in our Route are ready to use as variables in our “show” method.
* View function in our show method, is directing us to the view page “show.blade.php” inside of the folder “question”.
* PATH = view/question/show.blade.php



* Image above is the middleware method I have mentioned earlier.
* In this case, $request represents the Route logic we are going to execute to reach our QuizON page while $next is giving us the authorization to move on.
* In this case what handle function does is to check if there is an authenticated user that trying to reach the page. Otherwise, it simply directs the user to login page.
* A big advantage of middlewares are, instead of defining the conditionals everytime we have a new controller file, we can simply use the suiting middleware for our routes!



Here is our viewPage to see the questions for QuizON.

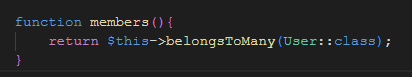
But before we start with this page, we need to give a look to our models, and their relationships with each other.

Models in use here are,

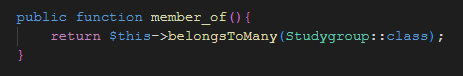
* StudyGroup Model
* User Model
* Question Model
* Course Model

Many To Many Relationship

At StudyGroup Model



At User Model



Necessary Pivot Table

Text

Description automatically generated

When user joins (or creates) a studygroup, studygroup\_user pivot table fills-in automatically.

The magic that happens here is a Laravel convention.

* If the pivot table has been created in alphabetical order with the name of necessary models (as seen: studygroup\_user), and relations have been fetched like as seen above, Laravel is smart enough to understand that this table needs to be filled with user and studygroup informations, which is also another convention.
  + Laravel understands that “\_id” is to grab that spesific model’s ID number.

One to Many Relationship

Question Model

Text

Description automatically generated

Text

Description automatically generated

Course Model

Text

Description automatically generated

Text

Description automatically generated

Questions Table

Graphical user interface, text, application, chat or text message

Description automatically generated

While a question is the property of a course and a studyGroup, same course can have many questions and that course should also be the property of the studyGroup that mentioned.

Too Many Relationship(s)

User Model

Text

Description automatically generated

Since we have the idea of, how Laravel Eloquent Relationships work, we can go back to our view file and check out what is really going on.

Text

Description automatically generated

* First if statement (1) above is to check if the user is a member of that studyGroup, else user will be directed to a page that he/she can join the studyGroup.
* With foreach loops (1&2), we are grabbing every question of the users that which are the member of that studyGroup.
* With following if statement (2), we are checking if that question belongs to that spesific course for the QuizON in session.
* All the following if statement (3) does is to check if the user already has solved the question.
* And at last, if (4) the question does not belong to the user, we can present it in our viewPage!

Finally, we are done with our GET & POST Requests, which have included plenty of information.

We have created models, controllers, migration files for our database system... But how?

* Laravel works with CLI commands!
  + To create a Model

>> php artisan make:model <model-name>

* + To create a Model + a Migration for that model

>> php artisan make:model <model-name> -m

* + To create a Migration

>> php artisan make:migration

* + To create a Controller

>> php artisan make:controller <controller-name>

* And so on. With Laravel, to create a template file is this easy!

Before ending the technical portion of this report, I would like to talk about two other concepts of StudyON,

* Scheduling
* Notifications

**Scheduling**

**Text

Description automatically generated**

With Laravel, task scheduling is easy!

* First step to take is to go to the Kernel.php file inside of Console directory.
* We can find the method “schedule” ready to use right there.
* All we need to do is to put the logic we want to implement and then, from our CLI, run the command that will do the work.
  + >> php artisan schedule:work

myScheduleLogic

* With php built-in Carbon object, we are retrieving current time.
* With using Question model, we are retrieving all the questions, from questions table and manipulating each of them in the for loop.
* Inside of the loop, we are reaching to the date which the questions have been created at.
* With a couple Carbon method, we can get the difference between now and then as integer values!
  + In this case, difference is shown as how many days it has been.
* At last, if the difference is more than or equal to seven days, we put the question to the releated studyGroup’s questionBank right before we delete the question from the QuizON section.

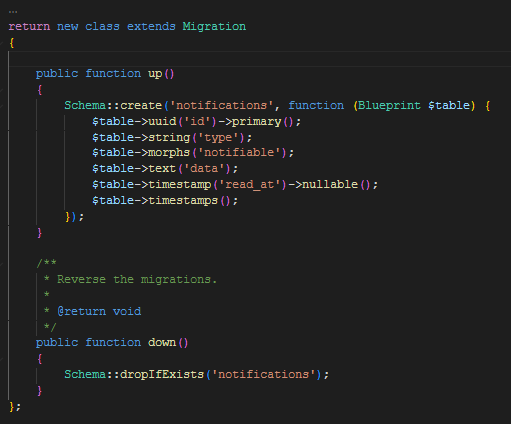
This schedule logic has been implemented to work every minute, but it can be changed to many date and time options too!

**Notifications**

It’s obvious that Laravel is a perfect framework to work on web applications, thanks to all built-in objects that saves us so much time/effort and Notifications are one of those objects.

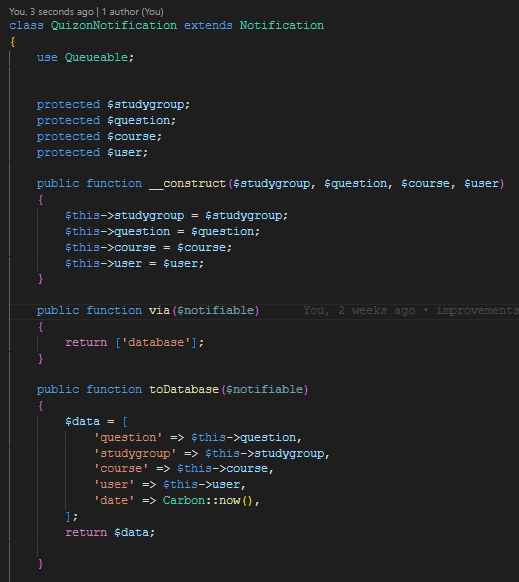
* To start, we can make a BluePrint for our notifications.

>> php artisan notifications:table

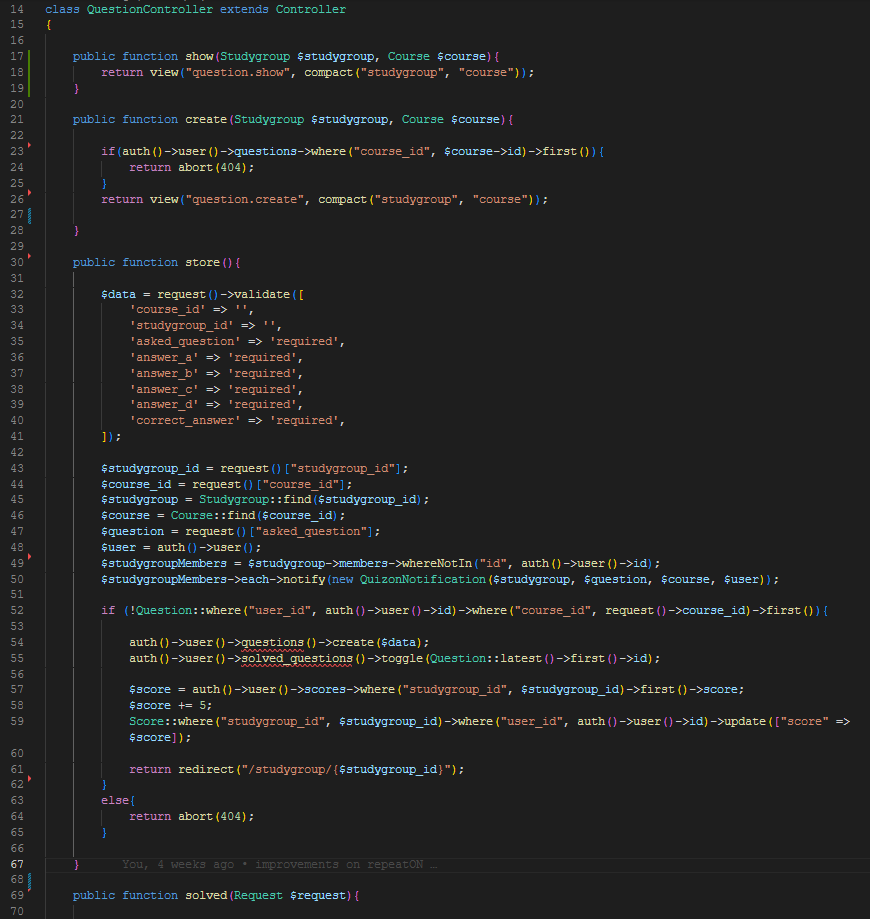


* “data” in our bluePrint represents a JSON object that will keep any information we need for our upcoming notification object.
* In this example we will be seeing how the QuizonNotification Class has been made.
* To start, we will need another CLI command,

>> php artisan make:notification QuizonNotification

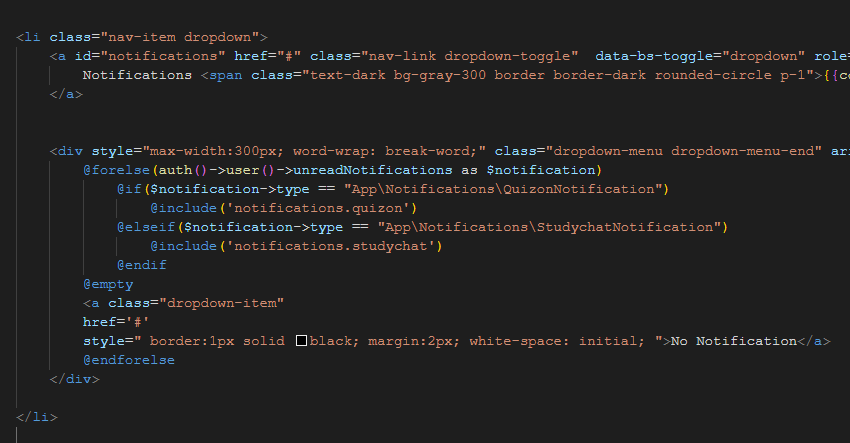


* As seen, in “\_\_construct” method, there are some known parameters by our QuizON feature.
  + “\_\_construct” methods can be seen as the main method, which lets us use the parameters in various methods belong in the same class.
* Since the concept of our notification system is to display information in the application window, we are using the built-in “toDatabase” method.
  + Information in this method will be written into the data column of our Notifications table, which is in JSON format as I mentioned above.

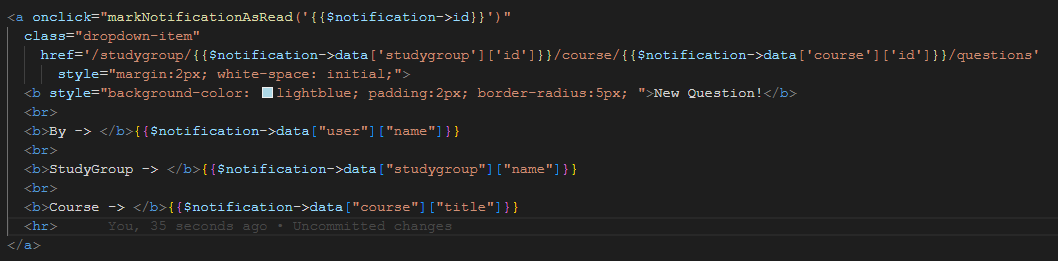


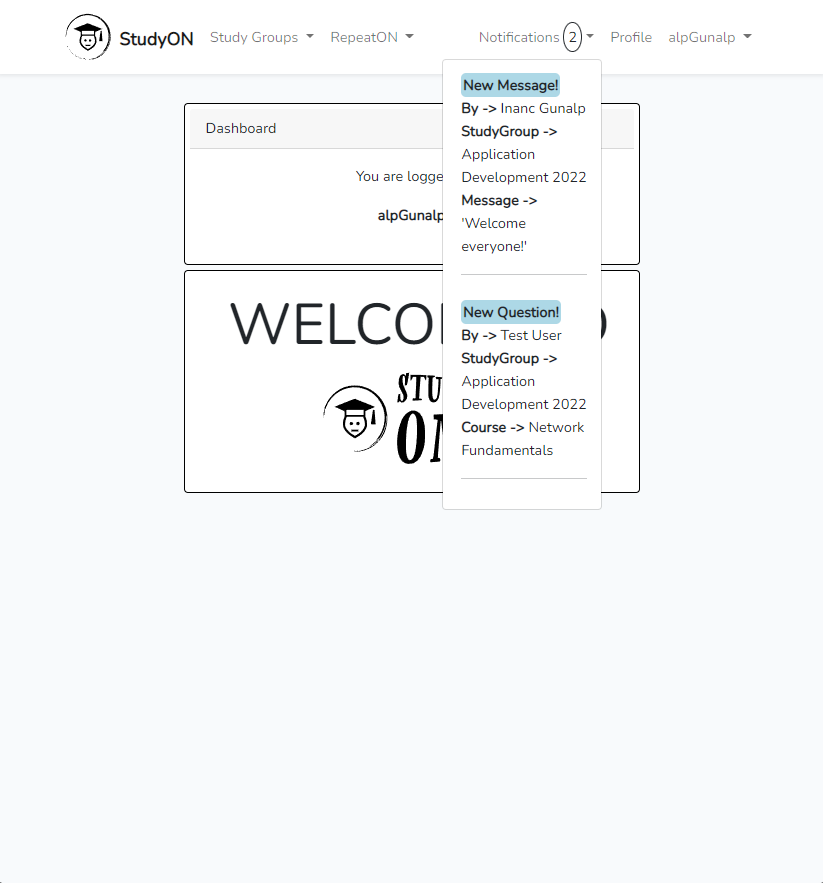
* Notice that, this is a portion of the store method, we have seen from the QuestionContoller.
* As seen here, we are using a built-in method called “notify” to reach and create a new QuizonNotification object, and the parameters for this object can also be seen as variables which are self-explanatory enough.
* With using this technique, we have successfully inserted the data into our database table, “Notifications”.

**Note!** Laravel handles the relationship in between User Model and Notification Objects, since they are both built-in objects. As seen, we have not fetched any relationship in between users and notifications for the method “notify” ++ upcoming method “unreadNotifications”.



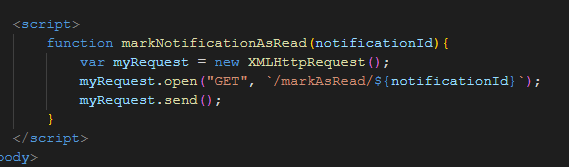
* Here above, as seen that with a for loop, we can reach the “unreadNotifications” for the user.
  + “unreadNotifications” returns the notification data row from our database table.
* With using type method, we can reach different kinds of Notifications from the same table.
  + In our case it is the QuizonNotification Class.
* “@include” lets us separate our code into portions that we can store in different view files.



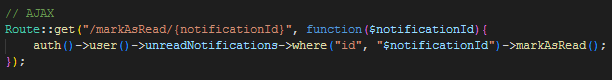
* When we come over to our viewPage for QuizonNotifications, we can manipulate the JSON data to grab any values we want to present to the User.
* **Note!** Notice that, we have used an onclick event in our anchor tag and how we pass the notification ID within it. We will come back to it very soon.
* As can be seen, notification presented at the bottom, follows the instructions given in the code right above.

Notifications, Last Step -> AJAX

* When user clicks on the notification, original route takes user to the page releated with the notification.
* But also, with using AJAX, when user clicks on the notification, it will be marked as read via onclick event I have mentioned earlier.



* XMLHttpRequest, is a built-in object for every modern browser!
  + With using it, we can create another GET Request and reach out to a Route method, which will include the necessary functionality to manipulate our notification mark as read!



* Notice that, this Route looks a bit different than other ones that we have seen.
  + Since all it does is to serve for a single purpose, we don’t need to reach out for a ControllerClass and ask for a method.
  + We can simply write the necessary method.
  + And we are DONE!

**Conclusion**

**myReflection:**

* It has been an exciting Learn & Implement journey.
* Project itself has some flaws and bugs need to be cured, but with the scale of the project.
* I’m hoping that there will be always a new bug to fix, with always refreshed new features, so I can continue learning more for to keep my path future-proof.

**totalCost:**

* Only cost was the time I have spend on the project, which can be seen as a good investment!

**thingsToAdd&Change:**

* Better User Interface
* User Profile features.
* Report Question.
* Overall advancement in main features.
* DeleteUser, modifyMessage, updateQuestion and so on.
* Better Notification system.
* Better implementation on task scheduling for question removal.
* OnDeleteCascade method for relational tables.
* And so on, since there will be a need for a change and update, always!

**Appendices**

**code:**

* There is no non-custom code, copy-paste etc. included in this project.
* Every line is written by me and included via Laravel framework itself.
* Code can be found as another attachment which is sent with this report.
* Github repository link:
  + https://github.com/inancalp/studyON-alpha