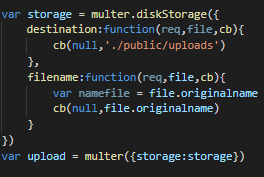
# Function

## Upload file and images



First, the system will set a location to store the files that students upload to the system and their names when saved at that location. After that, the uploading of the student's file will have to go through the following tests with the condition: can upload only 1 word file and 1 image file (optional).

The system will check the number of files uploaded by the student. In case of uploading a file to the system, the system will force it to be a word file. After checking, if it is not a word file, the system will report an error and send a notice. If passing the test, the system will proceed to set the name and address to save the file and save that information to the database.

* Case upload 1 file: Only docx is acceptable

fileRouter.post('/upload',upload.array('filePath',2),(req,res)=>{

    x = req.files[0].originalname

    if(req.files.length == 1){

        if(x.endsWith('png')|| x.endsWith('jpg')|| x.endsWith('gif')|| x.endsWith('docx')){

            if(x.endsWith('png')||x.endsWith('jpg')||x.endsWith('gif')){

                res.send('<script>alert("Only file formats docx, img, png, gif can be uploaded. You must upload at least 1 docx file and 1 image file (optional)");window.location.href = "/file";</script>');

            }else{

                xdoc ='uploads/'+  req.files[0].originalname

                var x1 = './public/' + xdoc

                var xx = x1.split('.');

                filePath1 = '.' + xx[1] + '.pdf'

                var filePath = xdoc.split('.');

                filePath = filePath[0] + '.pdf'

                docxConverter(x1,filePath1,function(err,result){

                    if(err){

                      console.log(err);

                    }

                    console.log('result'+result);

                });

                let email = req.cookies.email

                var temp = new fileModel({

                    filePathdoc: xdoc,

                    filePath:filePath,

                    nameFile : x,

                    studentemail: email,

                    slug: req.cookies.slug,

                })

                temp.save((err,data)=>{

                    if(err){

                        console.log(err)

                    }

* Case upload 2 file: 1 docx and 1 images are acceptable

For the case of uploading 2 files to the system. First, the system will proceed to check the files. If a student uploads files other than word (docx), image (png, jpg, gif) files, the system will send an error message to the student with an uploadable file type. After the list is checked, the system will conduct to check whether the two files are word files or image files. If two files have the same properties, a notification will be sent. If not, you will set a name and address to save the file and save the information in the database.

 y = req.files[1].originalname

        //check type of file 1 and file 2

        if(x.endsWith('png')||x.endsWith('jpg')||x.endsWith('gif')||x.endsWith('docx')||y.endsWith('png')||y.endsWith('jpg')||y.endsWith('gif')||y.endsWith('docx')){

            if(x.substr(x.length - 4,x.length) !== y.substr(y.length-4,y.length)){

                if(x.endsWith('png')||x.endsWith('jpg')||x.endsWith('gif')){

                    res.send('<script>alert("Only file formats docx, img, png, gif can be uploaded. You must upload at least 1 docx file and 1 image file (optional)");window.location.href = "/file";</script>');

                }else{

                    for(var i = 0;i<2;i++){

                        if(req.files[i].originalname.endsWith('png')||req.files[i].originalname.endsWith('jpg')||req.files[i].originalname.endsWith('gif')){

                            imgpath = 'uploads/'+  req.files[i].originalname

                            console.log(imgpath)

                        }

                        else if(req.files[i].originalname.endsWith('docx')){

                            y = req.files[i].originalname

                            x ='uploads/'+  req.files[i].originalname

                        }

                        }

                        var x1 = './public/' + x

                        var xx = x1.split('.');

                        filePath1 = '.' + xx[1] + '.pdf'

                        var filePath = x.split('.');

                        filePath = filePath[0] + '.pdf'

                        docxConverter(x1,filePath1,function(err,result){

                            if(err){

                            console.log(err);

                            }

                            console.log('result'+result);

                        });

                        let email = req.cookies.email

                        var temp = new fileModel({

                            filePathdoc: x,

                            filePath:filePath,

                            nameFile : y,

                            studentemail: email,

                            slug: req.cookies.slug,

                            filePathAnh:imgpath,

                        })

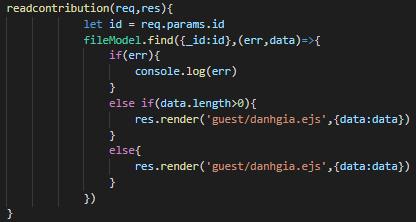
                        temp.save((err,data)=>{

                            if(err){

                                console.log(err)

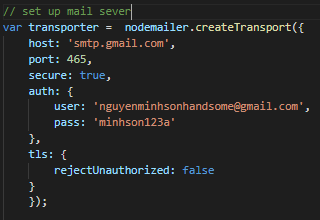
                            }

## View the selected reports.

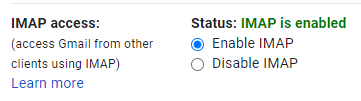
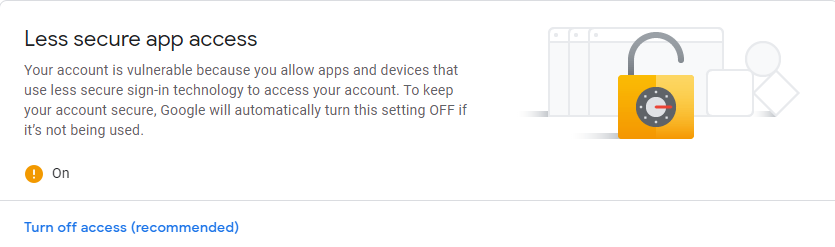


From the front-end, the user will click and read the articles they are interested in, the back-end will get the information's id of that article and get the address of that article in the database through the id and display the post through the <iframe> tag on the front-end.

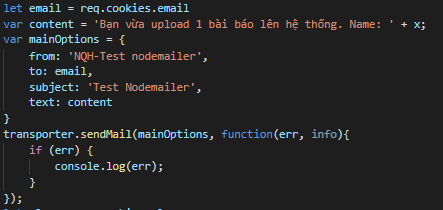
## Receive email notification after student posting articles to the system.



First, we need to install mail for the server through the nodemailer library. Besides, the server's mail account also needs to set up some settings on google such as: allowing low security applications to access. . This will allow the server to access this mail without being intercepted by Google's security. In addition, you need to enable IMAP to retrieve email messages from the mail server over a TCP / IP connection.

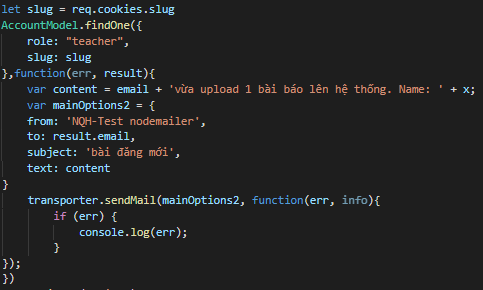


• Send email to student



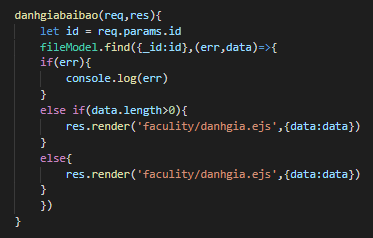
To send mail to students every time they post to the system, the system will need to write the content of the mail. Then proceed to set up your mailing address and subject. After composing the mail, it will proceed to send the mail through the trasporter.sendMail function. This operation will happen after saving the file information to the datase.

* Send email to Marketing coordinator

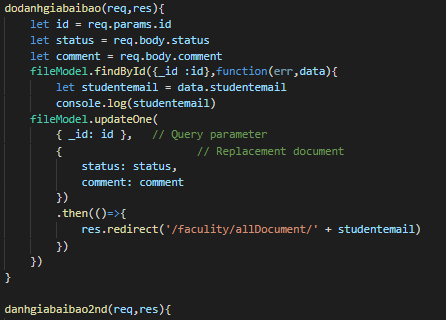


After students upload a post to the system, Marketing coordinator will also receive a notification about that post. Based on the student's scientific code (slug) and role as Marketing coordinator, the system will look in the database and retrieve the mail of that Marketing coordinator. After obtaining the Marketing coordinator's mail, compose the letter and send it to Marketing coordinator. This operation will happen after saving the file information to the datase.

## Make a comment



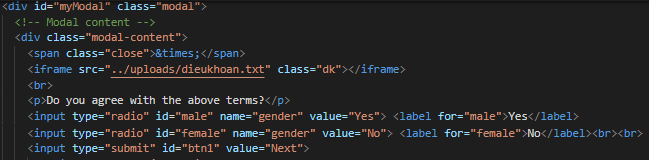
First, the system will get the id of the selected article and from the id will get the information of that article and send it to the interface page for evaluation. All information (status, comment, etc) of that file will also be printed out in the interface page.



After reading and giving comments and post status (Pass or Fail), the system will get the id and comment information, post status and save it to the database via the post id.

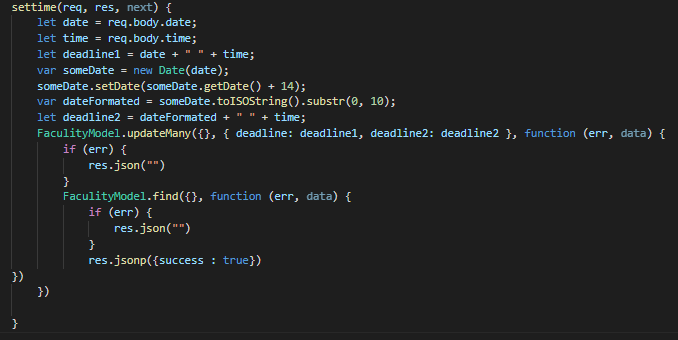
## Agree to Terms and Conditions before student submit.

To check if students agree with the terms before uploading files to the system, the system will create a pop-up form to test. If the student does not agree, an alert will be sent to the student that agrees to the terms. If the student has agreed to the terms, a check of the time that students are allowed to upload the work will be conducted. First, the system will get the current date and time and compare it with the date and time that Markerting Manager has given (var dealine). If it is past time, a notice will be sent out that it expired. The system will automatically redirect to the uploadfile page for the student if there is still a deadline for submission.



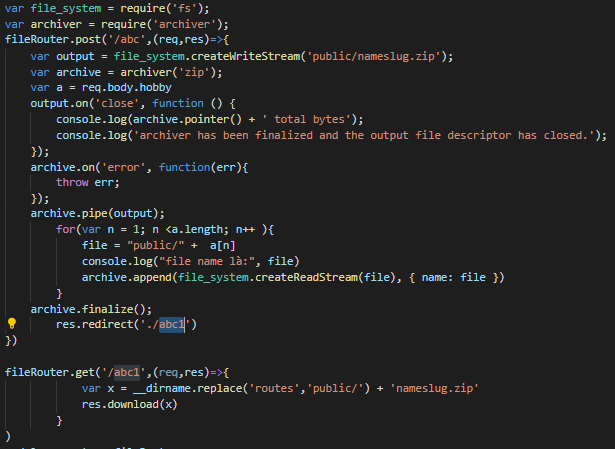


## Set time for upload file managine



To set deadlines for students, the system will get information about the date and time that the manager wants to install and save to the database. After getting the deadline (deadline 1), the system will automatically add 14 days to the deadline (deadline 2) and proceed to save the deadline information and edit it in the database. If the save is successful, the system will return a message.

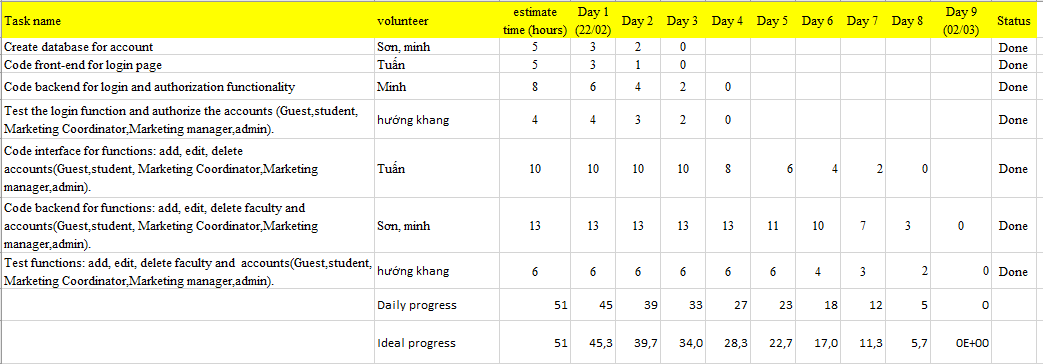
## Download zip:



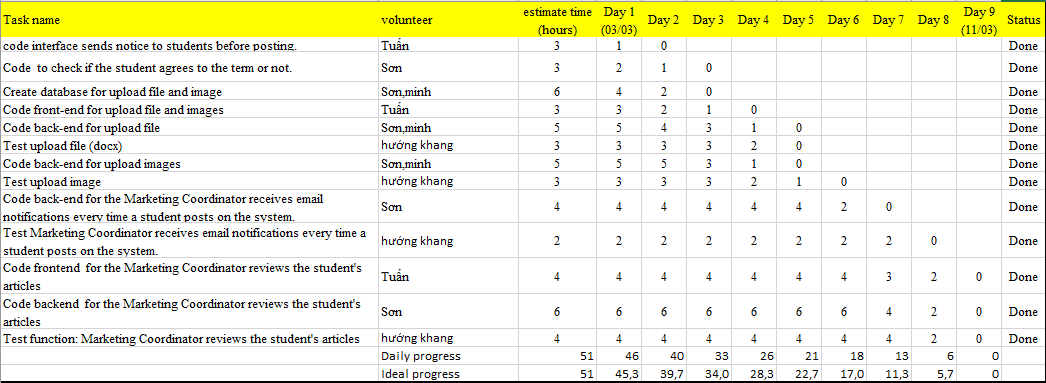
The system will use archiver library to compress docx files to zip. First, the system will create zip file: nameslug.zip and save it in public folder. After creating the zip file, the system will get the address of the article through the check box on the front-end and proceed to insert them into the zip file through the archive.append () function and then proceed to finish the compression process. Then the system will switch to the new router and download the file.

# Sprint Backlog & Burndown Chart

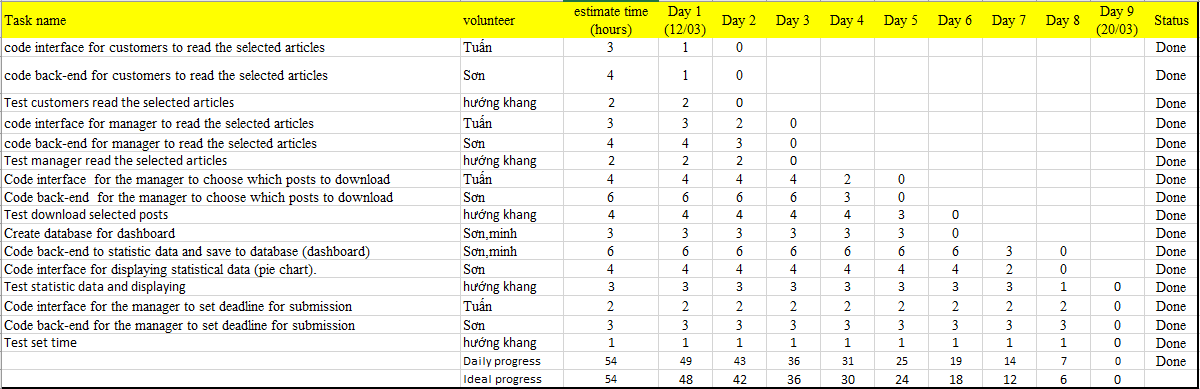
Sprint 1:



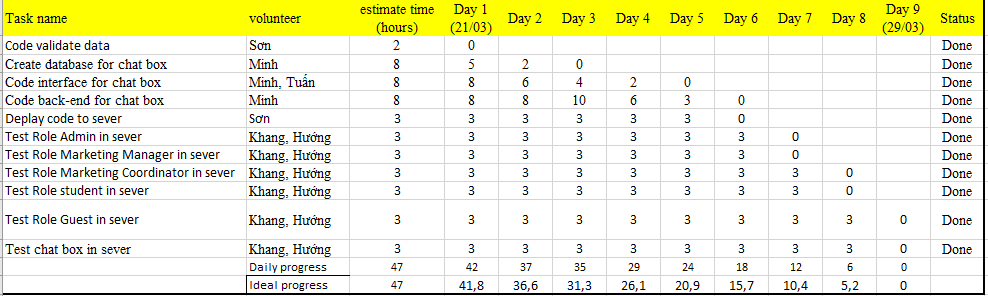
Sprint 2:



Sprint 3:



Sprint 4:



# Meetings

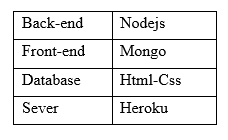
## Sprint meeting:

The first meeting at the start of the project was held on 21/2/2021 at 2:00 pm to 5:00 p.m.

The meeting was held to discuss the following issues:

* Discuss each person's competencies according to strengths and weaknesses and then assign work to roles (Scrum master, product owner, coder, design, etc) in project development.
* Discuss and make a decision about the technology to be used for the project development.
* Analyze project requirements and evaluate how important they are to the project and how difficult it is to carry out those requirements (creating a product backlog).
* Divide the work by weeks based on how important the project is and how difficult it is to do it.

Technical:



## Daily Meeting:

### Sprint 1 (22/2/2021- 2/3/2021)

We set 2 timelines for sprint 1 to monitor the situation.

**The first timeline on February 25, 2021:**

Because Son and Minh have researched and practiced with technologies from previous courses, database manipulation and decentralization are implemented quickly. In the first half of sprint 1, we accomplished the following:

* Sơn, Minh: Dabase has been created to store account information.
* Tuan: The user interface has been designed for the login.
* Minh: Decentralized the accounts based on roles.
* Orientation: The decentralization test has been completed and there is no risk.

Next work to be completed when Sprint 1 ends (March 01, 2021): Complete functions for Admin:

• Add, delete and update guest account

• Add,edit and delete student account

• Add, delete and update marketing coordinator account

• Add, edit and delete marketing manager account

• Add, editing and deleting faculties

**The second timeline on March 01, 2021 (the end of Sprint 1):**

* Son, Minh: 2 coders agreed to divide the work of the back-end code for the work of implementing the functions of the admin and completed the job on time and in a timely manner, assigned to the direction and testing.
* Tuan: The functions have the same interface structure so the design doesn't take too much time, Tuan has finished designing the interface for the functions to add and delete the accounts (4 roles) and for faculty.
* Khang, Huong: During the test, the direction and direction did not detect errors in adding and deleting accounts, but there are some validate problems that need to be solved when creating accounts such as: Name can enter numbers, electricity phone may enter text, enter wrong email, etc.

Note: The account information validation will be done in sprint 4.

### Sprint 2 (3/3/2021- 11/3/2021)

In sprint 1, the team completed the admin functions. In sprint 2, the whole team will practice building functions for students and some functions for Marketing Coordinator. In sprint 2, we set 2 timelines for sprint 2 to monitor the situation.

**The first timeline on March 6, 2021**

During the first half of sprint 2, we completed the following work in the plan:

Tuan: The interface has been designed for the display of the school's terms when students decide to post their articles on the system.

Son: Code work completed to check if the student agrees with the term. To do this, Tuan has helped paint better understand Pop-up in the interface. From there, Sơn learned to use the <script> tag in Html to test it right at the front-end.

Sơn, Minh: Both of them carried out to find out about the necessary attributes of the file to the database and have since completed the database creation.

Tuan: After designing and displaying the terms, Tuan continues to design the interface for students who upload their articles to the system and have finished.

The next work to be completed when Sprint 2 ends (March 11, 2021): Complete functions for students and some functions of marketing coordinator.

* Student: Upload file (docx), img, receive mail when uploading an article to the system.
* Marketing coordinator: receives an email when a student of the department posts an article on the system, evaluates the student's articles.

**The second timeline on March 11, 2021 (the end of Sprint 2)**

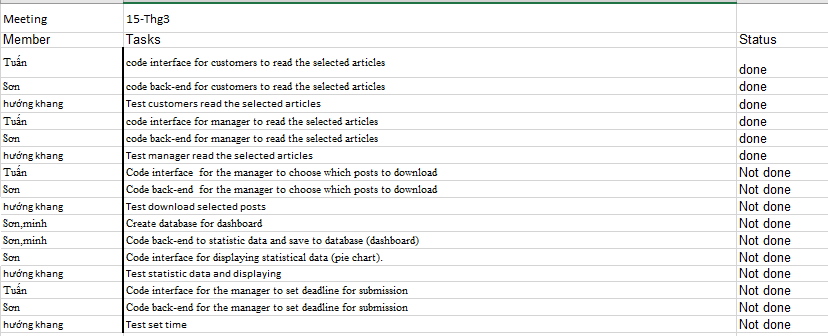
* Sơn, Minh: After completing the design of the interface for uploading files and images, I painted and I learned together about Nodejs libraries to save files to the system. Through many options I decided to use the multer library and perform the file and images work successfully.
* Hướng, Khang: During the file upload test, the system crashes when there are cases such as uploading files in incorrect format docx, png, jpg.
* Sơn: After performing the file saving function, Sơn learned about the library to help the system send mail and found the nodemailer library. Through the manuals, Son has completed 2 functions of receiving mail for teachers and students.
* Hướng, Khang: During the test of receiving mail, no errors are detected
* Tuan: After designing the page to upload files for students, Tuan has finished designing the interface to display the articles and the interface that helps Marketing coordinator evaluate the students' articles.
* Sơn: After discussing with Tuấn, the two of you decided to display the post with the <iframe> tag in PDF format. After closing, Son has completed the back-end coding for the article evaluation.
* Hướng, Khang: Orientation and Khang have checked the article reviews and no errors have occurred.

Note: The sprint 2 work has ended as planned.

### Sprint 3 (12/3/2021- 20/3/2021)

We set 2 timelines for sprint 1 to monitor the situation.

The first timeline on March 15, 2021, we met together and divided the work as follows:



The second timeline on March 20, 2021, we have released the results of our work during the 1st milestone of operation and report the work progress will be made in the 2nd timeline:

### Sprint 4 (21/3/2021- 29/3/2021)

We set 2 timelines for sprint 1 to monitor the situation.

The first timeline on March 24, 2021, we met together and divided the work as follows:

Sơn:

Tuấn:

Minh:

Khang:

Hướng:

The second timeline on March 28, 2021, we have released the results of our work during the 1st milestone of operation and report the work progress will be made in the 2nd timeline:

Sơn:

Tuấn:

Minh:

Khang:

Hướng: