Table of Contents

Table of Figure

Introduction

This is a group subject. we needed to adopt agile working methods and record meetings appropriately. Ideally we would assign members to the following positions: a database designer, an information architect, a programmer, a web designer and a tester. As well as a scrum specialist and product owner, but can have more than one person in any technical role. No one has to take on the role of project manager, but we recommend a technical lead.

Role	**Member**
Product Owner	Bui Chi Huong
Database Designer	Son, Minh
Developer	Front-end: Tuan Back-end: Son, Minh
Tester	Bui Chi Huong, Khang
Scrum master	Son

We are required to build a secure web-based role-based system to gather student contributions to an annual college journal within a major university.

The system includes the following criteria:

Tools:

NodeJs:

NodeJS is built on the Javascript V8 Engine platform, NodeJS is used to create websites such as sales sites, forums and social networking sites in narrow scope. Node JS is one of the many open source developers used by many programmers around the world.

Node JS can run on many platforms such as window, IOS, Linux so it is also one of the advantages for programmers. Node JS has a very large library in the form of different Javascript Modules, which simplifies programming and minimizes working time.

(https://vietpro.net.vn/nodejs-la-gi/)

Reasons choose NodeJs:

Node.js is very fast: we are a project to build a student management website so it is very important because we are trying to make a big product and we want to make sure. that it can expand quickly, and fully satisfy a large number of users as the website grows.

Node.js is capable of handling multiple connections at a time, while PHP doesn't. So apart from the performance and scalability benefits we can learn a little more about JavaScript as well, so it's easy to learn and learn a whole new language.

(https://vietpro.net.vn/nodejs-la-gi/)

Mongo:

MongoDB is an open source, NoSql database and is currently used by millions of programmers.

MongoDB is a document oriented database, data information is stored in JSON documents instead of a tabular form like a relational database, so queries and interactions will be very fast.

With relational databases, we have the concept of tables, relational databases (like MySQL or SQL Server ...) use tables to store data, with MongoDB will use the concept of collection instead of board

Compared with RDBMS, the MongoDB collection corresponds to the table, and the document will correspond to row, MongoDB will use the documents instead of row in the RDBMS.

Collections in MongoDB are structured very flexibly, allowing data to be stored without following a specific structure.

Information is stored together for quick query access through the query language contained in MongoDB.(

https://viblo.asia/p/mongodb-la-gi-co-so-du-lieu-phi-quan-he-bJzKmgoPl9N)

Reasons choose Mongo:

MongoDB uses to store data in the form of JSON Document, so each collection will have flexible sizes and documents, this flexibility is very good in storing data, so whatever we want, please insert it freely. .

Data in MongoDB has no mutual binding, no join as in RDBMS, so when inserting, deleting or updating it does not need to take time to check whether data constraints like in RDBMS are satisfied. MongoDB is very extensible.

High performance: MongoDB's query speed (find, update, insert, delete) is much faster than other relational database management systems. With a large enough data, the test shows that MongoDB's insert speed can be faster than other candidates. (https://viblo.asia/p/mongodb-la-gi-co-so-du-lieu-phi-quan-he-bJzKmgoPl9N)

Heroku:

Heroku is a cloud platform that allows developers to build, deploy, manage and scale applications (PaaS - Platform as a service).

heroku is very flexible and easy to use, providing developers with the simplest way to bring the product to the user. It helps developers to focus on website development without paying too much attention to server operation or hardware.

(https://topdev.vn/blog/heroku-la-gi/**)**

Reasons choose Heroku:

- **Provide the best user experience.**
- **A service ecosystem.**
- **Support to connect to salesforce.**

(https://cuongquach.com/heroku-la-gi.html**)**

Html-css:

HTML stands for Hypertext Markup Language, which is the preferred language used by programmers to write Web pages. Hypertext is the way Web pages (HTML documents) are linked together. Links on a Web page are called Hypertext. HTML is a Markup Language, which means we use HTML to mark up a text document with tags to help the Web browser understand the structure and display it on the screen. Today, HTML has been widely used to format Web pages with the help of various tags included in the HTML language.

(https://vietjack.com/html/html_la_gi.jsp)

CSS is the formatting language for websites - short for Cascading Style Sheet language. It is used to style and style elements written in markup language, such as HTML. CSS can be used to control the formatting of multiple web pages at the same time, saving programmers time and effort. It distinguishes the layout of the website and the main content of the page by controlling the layout, color, and font.

(https://www.hostinger.vn/huong-dan/css-la-gi**)**

Requirement:

Functional + non-functional

Id	Criteria	Type	Reasons
1	The University has a Marketing Manager to oversee the process.	Function	It is the function of adding marketing manager to manage, this function depends on the hierarchy of the system
2	All Faculties have a Marketing Coordinator who is responsible for managing the process for their Faculty.	Function	It is the function of adding marketing coordinator to the head of departments, this function belongs to the hierarchy of the system
3	All students have the opportunity to submit one or more articles as Word documents To the magazine. (assumptions)	Function	This is one of the requirements describing WHAT THE SYSTEM MUST DO to enable students to submit their reports.
4	All students can also upload high quality images, e.g. Photographs.	Function	This is one of the requirements that describes WHAT THE SYSTEM MUST DO before a report is submitted
5	All new contributions are disabled after a closure date for new entries, but updates Can continue to be done until a final closure date.	Non-Function	This function requires an external condition that is not a function that the system is capable of handling
6	All students must agree to Terms and Conditions before they can submit.	Function	This is one of the requirements describing WHAT THE SYSTEM MUST DO
7	Once a contribution is submitted the system emails a notification to the Faculty's Marketing Coordinator, who must make a comment within 14 days.	Non-function	The system must be able to do the commenting in the system this is a must-have non-function
8	A Marketing Coordinator can only access contributions by students in their Faculty.	Function	The system can help the exam coordinator reach his students
9	Each Marketing Coordinator needs to be able to interact with the students in their Faculty in order to edit the contributions and to select those for publication.	Function	The system can make it possible for the Marketing Coordinator to be able to interact with students in the Faculty to edit contributions and to select those contributions for publication.
10	The University Marketing Manager can view all the selected contributions but cannot edit any. They need to be able to download all the selected contributions After the final closure date in a ZIP file for transfer out of the system.	Function	The system must be able to do the download
11	An administrator maintains any system data, e.g. Closure dates for each academic Year.	Function	The system must be able to do the set time
12	A guest account for each Faculty can be used to view the selected reports.	Function	View selected contributions as what the marketing decentralization system could use
13	Statistical analysis (e.g. Number of contributions per Faculty) needs to be available.	Function	The system can automatically analyze statistics.
14	The interface must be suitable for all devices (eg mobile phones, tablets, desktops).	Non-function	Are the criteria that a software system needs to meet when being executed. A suitable interface is required

Database:

ERD

We use MongoDB to store one-to-many data and models, the data from the collection can be compared to the data in another collection.

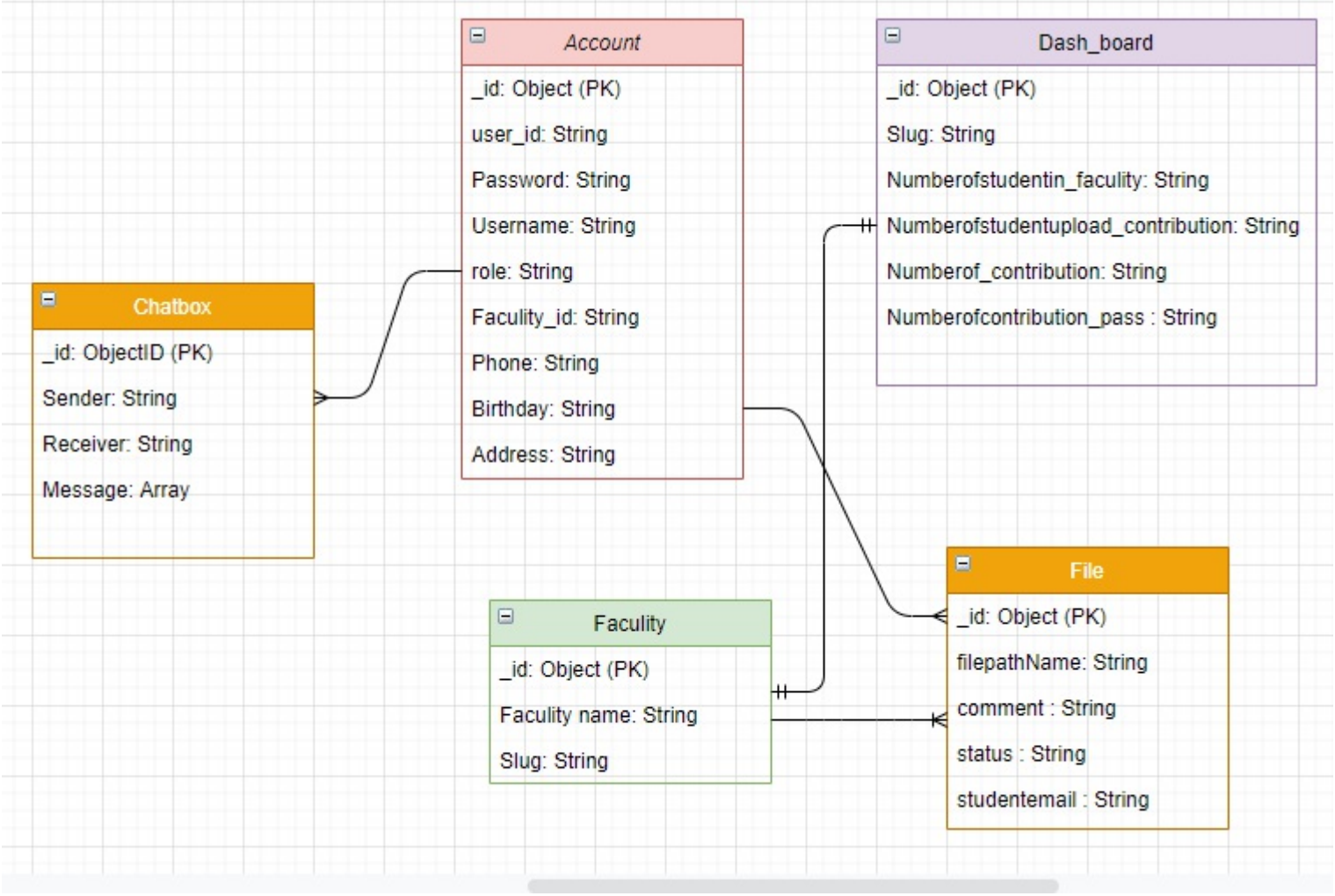

Figure 1:ERD

To meet the project requirements, we have designed 5 collections included:

_The account collection will contain attributes of the user, such as name, age, email, password, phone number, address, date of birth. The slug field we created is for the purpose of referencing other collections in the database. If the account role is student then a student can submit multiple files to the system.

_The faculty collection will contain the name and slug of the faculty. We use the slug to reference the student and the marketing coordinators available in that faculty. One faculty will have many students and only one marketing coordinator

_The chat collection will contain recipient's email name and sender's email and message. The message field is an array, the data will be pushed when users chat with each other. One faculty will have many students and only one marketing coordinator. If the account's role is a student, then 1 student will only be able to chat with one marketing coordinator. If the account's role is the marketing coordinator, the marketing coordinator can chat with all students in the faculty

_ The dashboar collection will include the following properties: slug, the number of students in the department, the total number of students paid in the department, the total contribution paid and the total number of contributions passed. The slug field will refer to the faculty in the collection faculty in the database. Each 1 dashboar will be associated with 1 faculty and vice versa

_ The file collection will include: the path of the file, the file name and the student's email. The path of file field will be the path to upload the file to the server and the email field will reference the account collection to identify who uploaded the file.

Data security

To secure the user's data, we used bcrypt and salt to encrypt information and authenticate. We use the hash () function to convert a value into another. The reason we use the hashing algorithm is because it cannot be decoded or converted back to its value.

The hash () function takes 2 parameter values, which are the data to be encoded and "salt". Salt is added to the hash head to protect the password in case of attack. The value of "Salt" denotes the complexity of the hash function, the higher its value, the

longer the password has to be hashed.

```
let username = req.body.username;
let password = req.body.password;
let email = req.body.email;
let slug = req.body.slug;

        const salt = bcrypt.genSaltSync(saltRounds);
        const hash = bcrypt.hashSync(password, salt);
        let newStudent = AccountModel({
            username,
            password :hash,
            email,
            slug
        })
        newStudent.save(function(err,data){
```

Figure 2: Data security

Upon login, we have turned user information into a token including "_id". Server side will look under _id for authentication, login and authorization. As soon as the token is signed, we use cookies to store the token. Storing information in cookies for too long can present many security risks. That's why we have set "maxAge" of cookies to last for 1 day to avoid hacker attacks

```
let token = jwt.sign({_id : req.user._id},'minh',{expiresIn :'1d'})
res.cookie("token",token,{maxAge: 24*60*60*1000, httpOnly: true });
```

Figure 3: Data security

To ensure cookies are only sent to urls with https addresses, we have used "httpOnly: true" to protect against XSS attacks.

We use a middleware to check the information the user enters upon login. By decompiling the token stored in the cookie to get the user's "_id" for comparison in the database. If the user's account already exists, the server will continue, if not, the server will disallow it and return to the login page.

```
let checkAuth = async (req,res,next)=>{
    try {
        var token = req.cookies.token || req.body.token
        let decodeAccount = jwt.verify(token,'minh')
        let user = await getUserById(decodeAccount._id)
        if(user){
            req.userLocal = user;
            next();
        }else{
            return res.status(400).json({
                message : "tk k ton tai",
                status: 400,
                error : true,
            })
        }
    } catch (error) {
        res.status(500).json({
            status: 500,
            error : true
        },
        res.redirect('/'))
```

Figure 4: Data security

To prevent a user of this role from being able to access another role's site, we have used an additional middleware to check information on the user cookies.

```
let checkAdmin = (req,res,next)=>{
    if (req.userLocal.role === "admin"){
        next()
    }else{
        res.status(400).json({
            message : "no permission",
            status: 400,
            error : true,
        })
    }
}
```

Figure 5: Data security

Data validation

In order to have actual data and avoid errors in the process of running the project, the system has validated some information of user accounts and information of departments.

Regarding email registration: Admin, when registering an account for a user, will have to enter the correct format for the mail. If not, it will report errors and not allow registration.

Notification: email must contain @ gmail.com, etc

```
<p>Email</p>
<input type="email" name = "username" required>
```

Figure 6: Validate email

About username: The system will check the user name by checking the characters in the name, if the Admin enters a username with special characters or numbers, the system will send an error and prompt the user to enter it correctly. form of name.

Notification: Username shouldn't contain number. e.g. Smith

```
<p>Name</p>
<input type="text" name = "name" pattern="[a-zA-
Z]+" title="Username shouldn't contain number. e.g. john" required >
```

Figure 7: Validate name

About the date of birth of the account: The system will display a calendar for the admin to choose when registering an account for the user. Using the calendar is more accurate and avoids minor errors like stamp, date format, etc

```
<p>Date of birth</p>
<input type="date" name = "birthday" required>
```

Figure 8: Validate birthday

About phone number information: Phone numbers are formatted as starting with the area code +84 (vietnam) and having at least 8 numbers. If the user enters text or special characters, the system will issue a format error and force the user to re-enter.

Notification:Phone number must not contain letters or any other characters.Phone number must be between 8-10 numbers.

```
<p>Phone</p>
<input type= "text" name = "Phone"  pattern="(\+84|0)\d{9,10}" minlength="8" m
axlength="10" title="Phone number must not contain letters or any other charac
ters.Phone number must be between 8-10 numbers." required>
```

Figure 9: Validate phone

About the faculty name: When the admin creates the faculty name, the faculty name will also have the same format as the username. If the user enters numbers or special characters in the faculty name, the system will also report an error.

Notification: Faculty shouldn't contain number. e.g. Electronic technology

```
<p>Faculty</p>
<input type="text" name = "faculty" pattern="[a-zA-
Z]+" title="Faculty shouldn't contain number. e.g. Electronic technology" requ
ired >
```

Figure 10: Validate Faculty name

About password: In order for a password to be secure, a password needs to have at least 8 characters and next to it does not contain any special characters. Having special characters can cause difficulty logging in.

Notification: Password needs to be at least 8 characters without special characters (/,?, Etc)

```
<p>password</p>
<input type="password" name="password"title="Password needs to be at least 8 c
haracters without special characters (/,?, Etc)" minlength="8" pattern="[a-zA-
Z0-9]+"  required>
```

Figure 11: Validate password

Design:

Sitemap:

Following are the links from the main page to the smaller pages of each role.

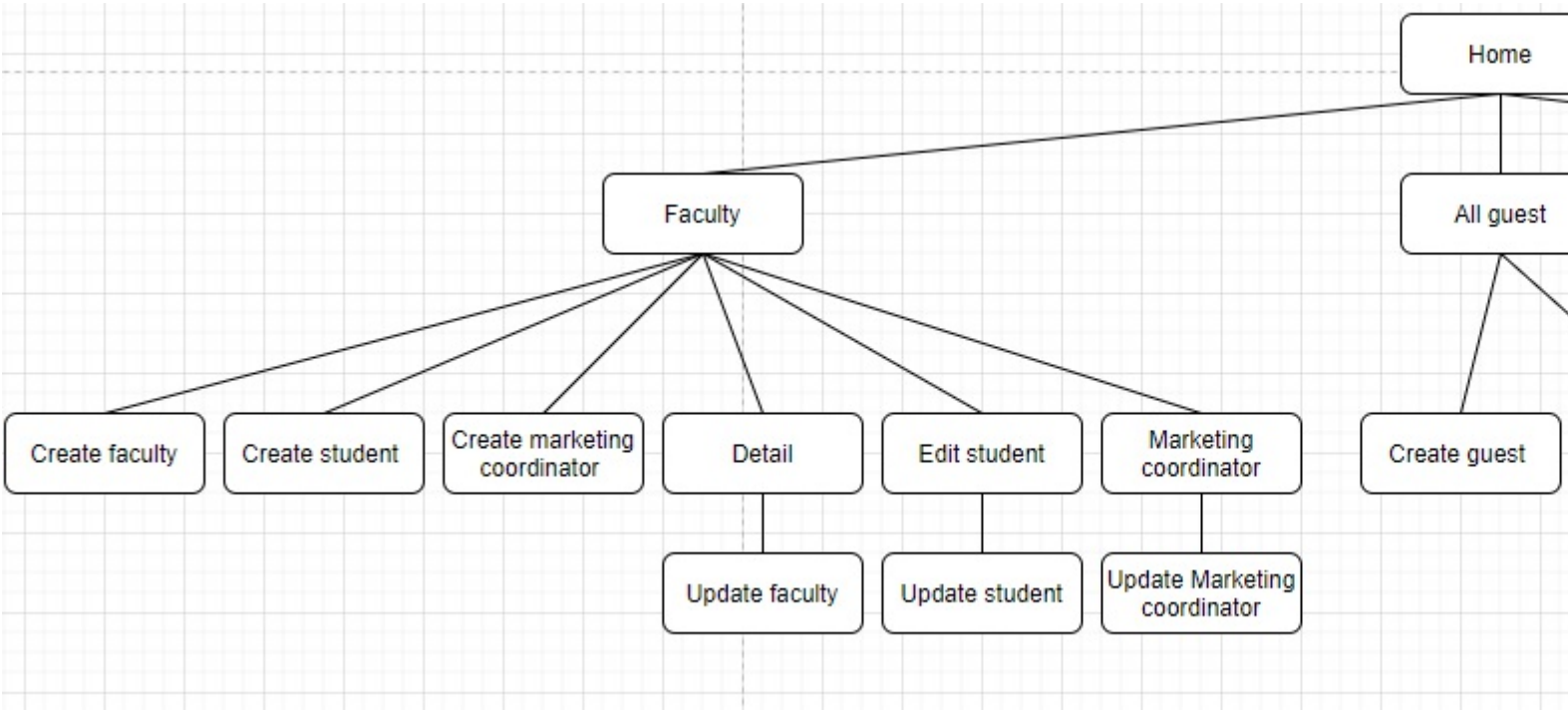

Figure 12:Sitemap admin

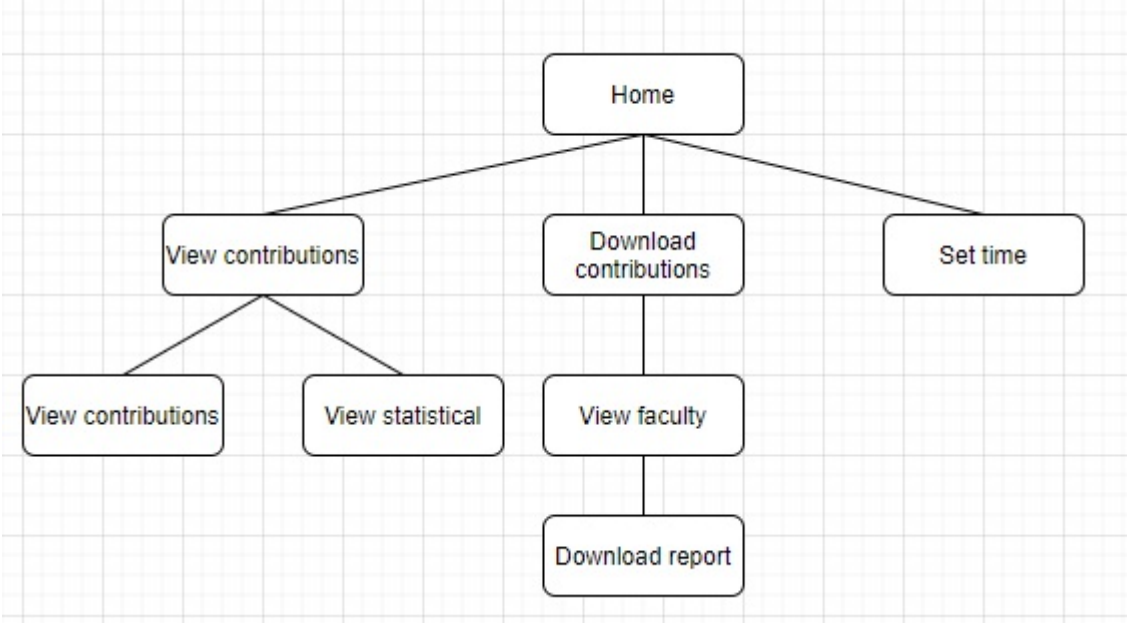

Figure 13:Sitemap marketing manager

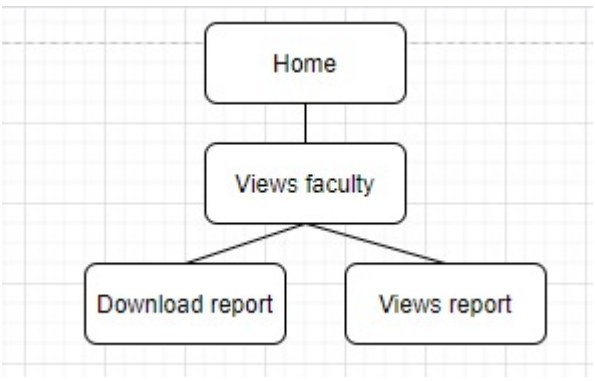

Figure 14:Sitemap guest

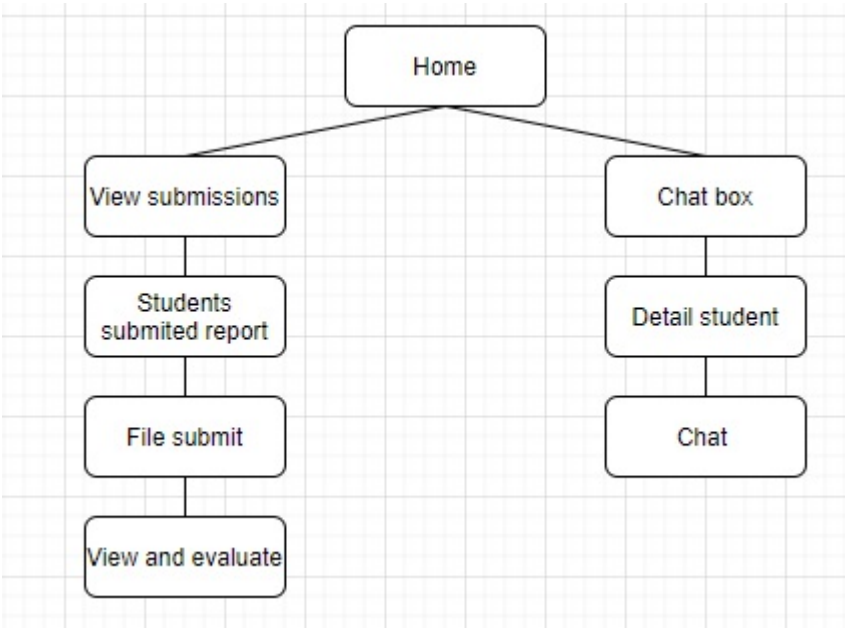

Figure 15:Sitemap marketing coordinator

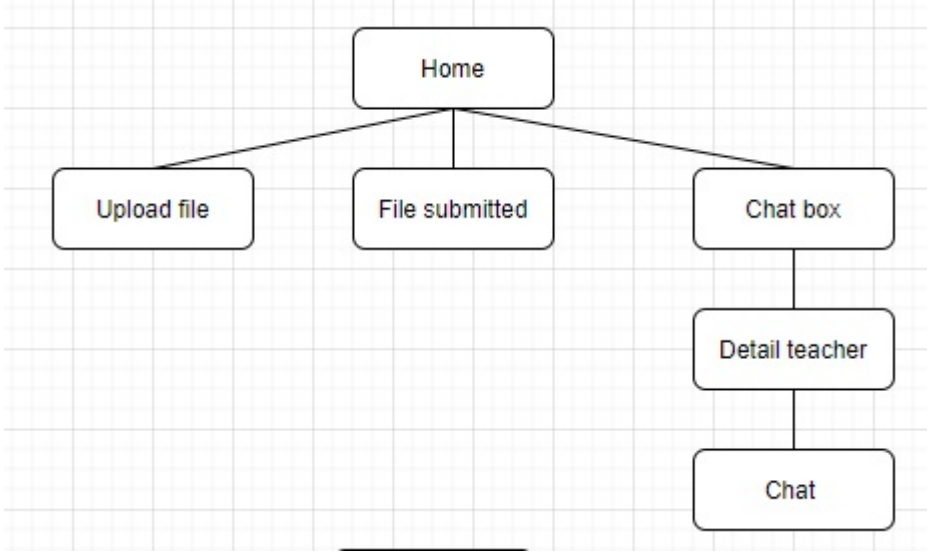

Figure 16:Sitemap student

Wireframe

According to the requirements of customers, we have to design interfaces for 5 different objects: administrator, Student, Marketing coordinator, Marketing manager, and Guest. For each object, we will design a separate interface, interface. The interface will be designed optimally and suitable for devices such as computers, mobile phones, ...

Login

First, when a user visits the website, the login page will be displayed first. Login is designed with a header in h tag with large clear font. Both the username and password fields are displayed in the text in the input area, making it easy for users to understand.

Figure 17:Wireframe login of computer and mobile

Student

The layout of the interface is divided into 4 main parts: the header contains the "X" button to logout, the menu contains the functions that the user can use and it is displayed on the left-hand side, the content contains the content, the Finally, the footer.

When entering the student interface, the menu will display student functions such as home, upload file, submitted file, and chat box. Content will display personal information of students including user name, email, faculty, Phone, Birthday, and address. Information will be included in the table to make it easier for the user to observe. The title section uses h-tags and large fonts so that users can better understand the website they use.

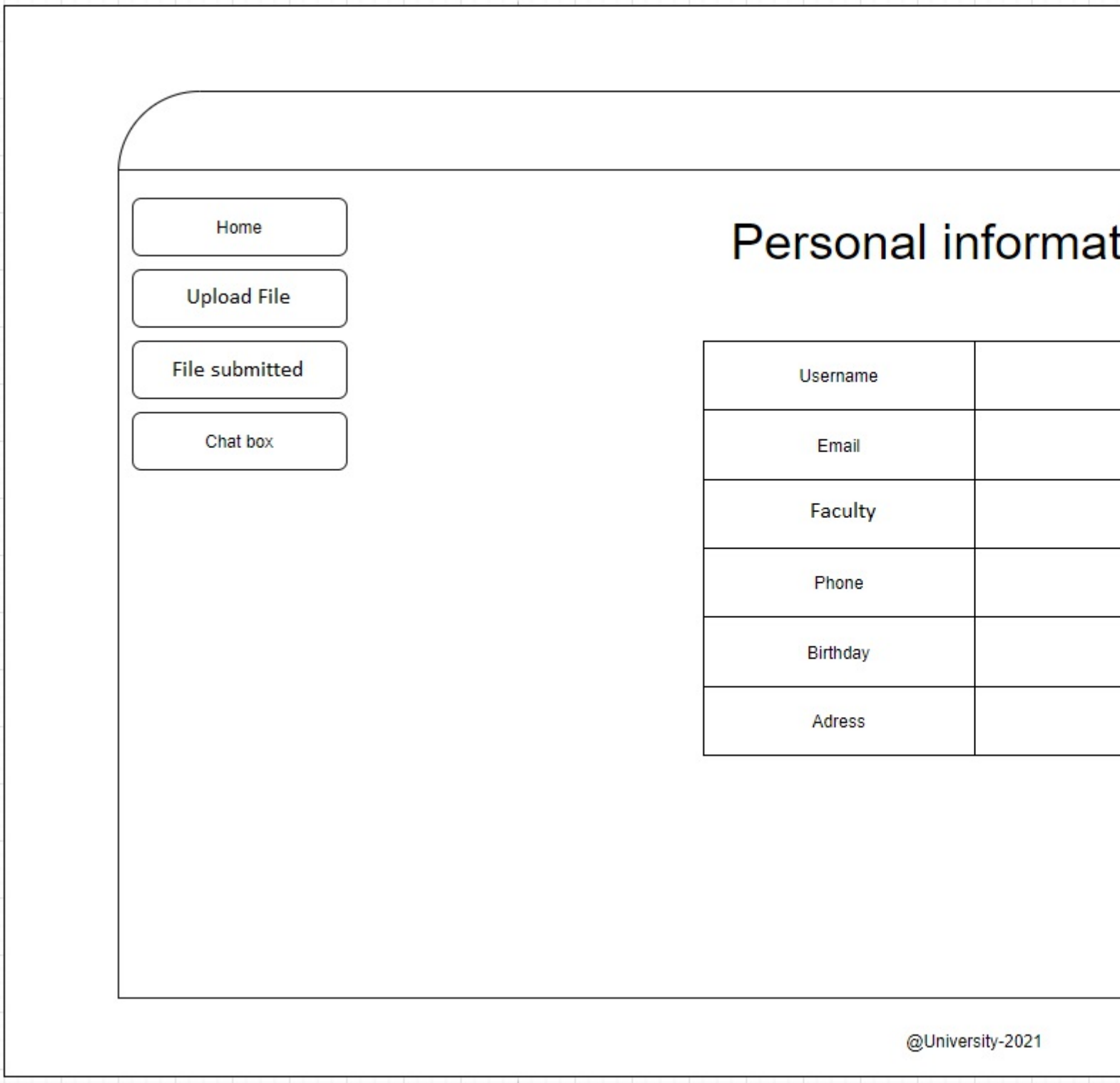

Figure 18:Wireframe home page of student

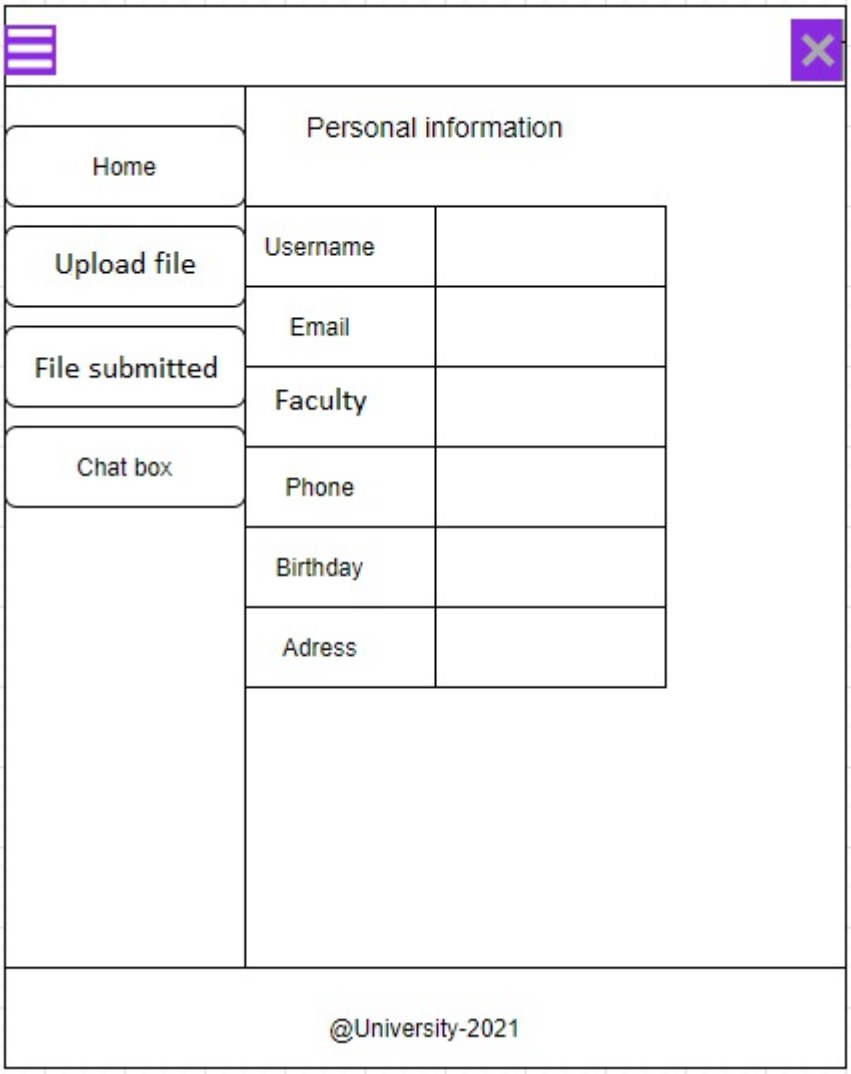

Figure 19:Wireframe home page of student mobile

To be suitable for mobile, the menu section will be hidden and display an icon for the left-hand menu to save space for the mobile device. When you need to use the menu, just press on the icon for the menu, the menu will appear and when the user presses the icon of the menu, the menu will be hidden.

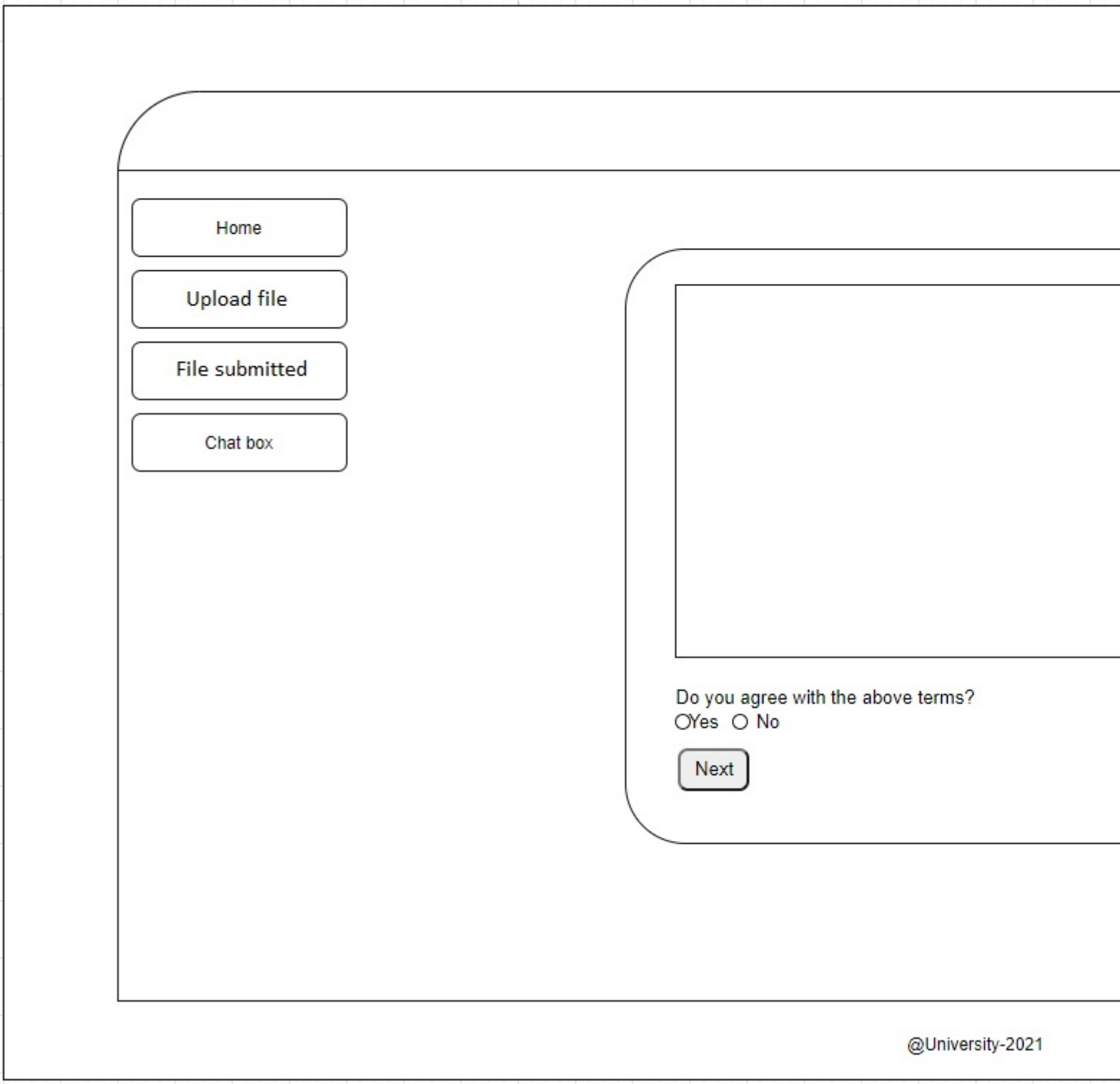

Figure 20:Wireframe popup terms

The terms display will use a popup to show the student's terms when trying to upload files.

Figure 21:Wireframe upload file

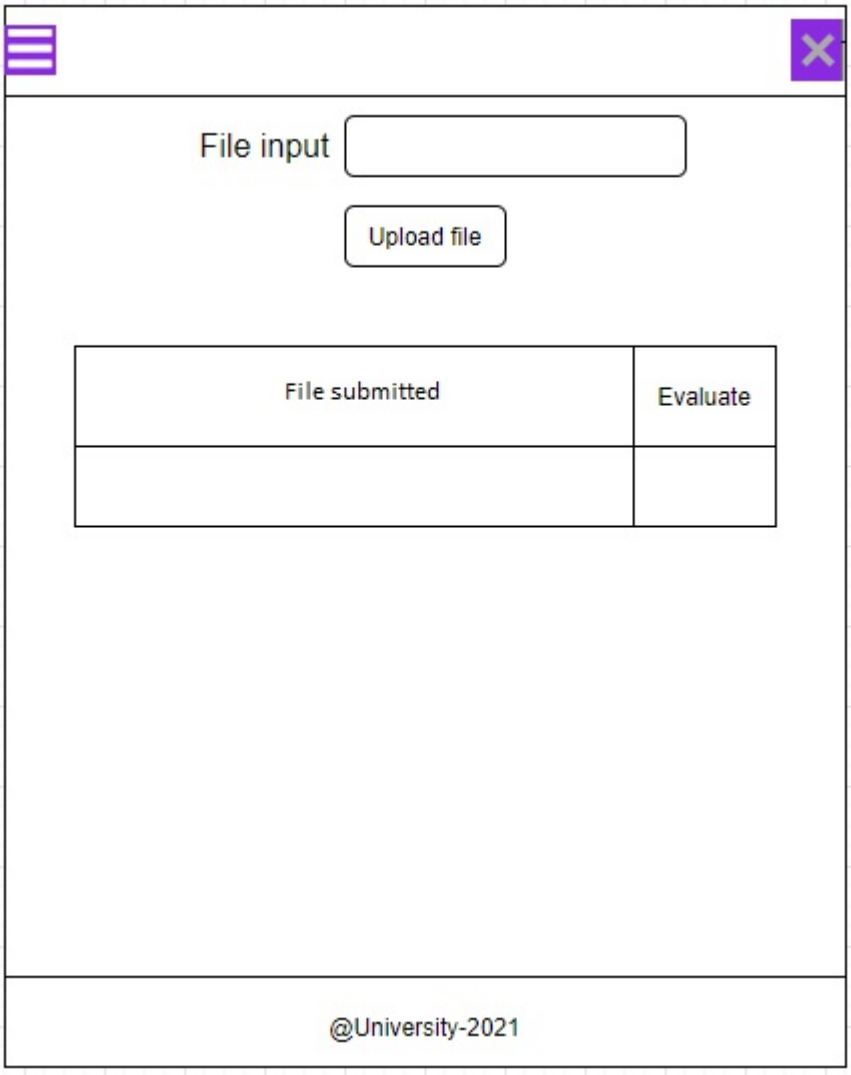

Figure 22:Wireframe upload file mobile

Using the input file header right before the file upload area will help users know the part they can upload files and the file upload button is located next to the user so that users can enter files and easily upload files.

The submitted files are placed in tables for easier viewing and management by users.

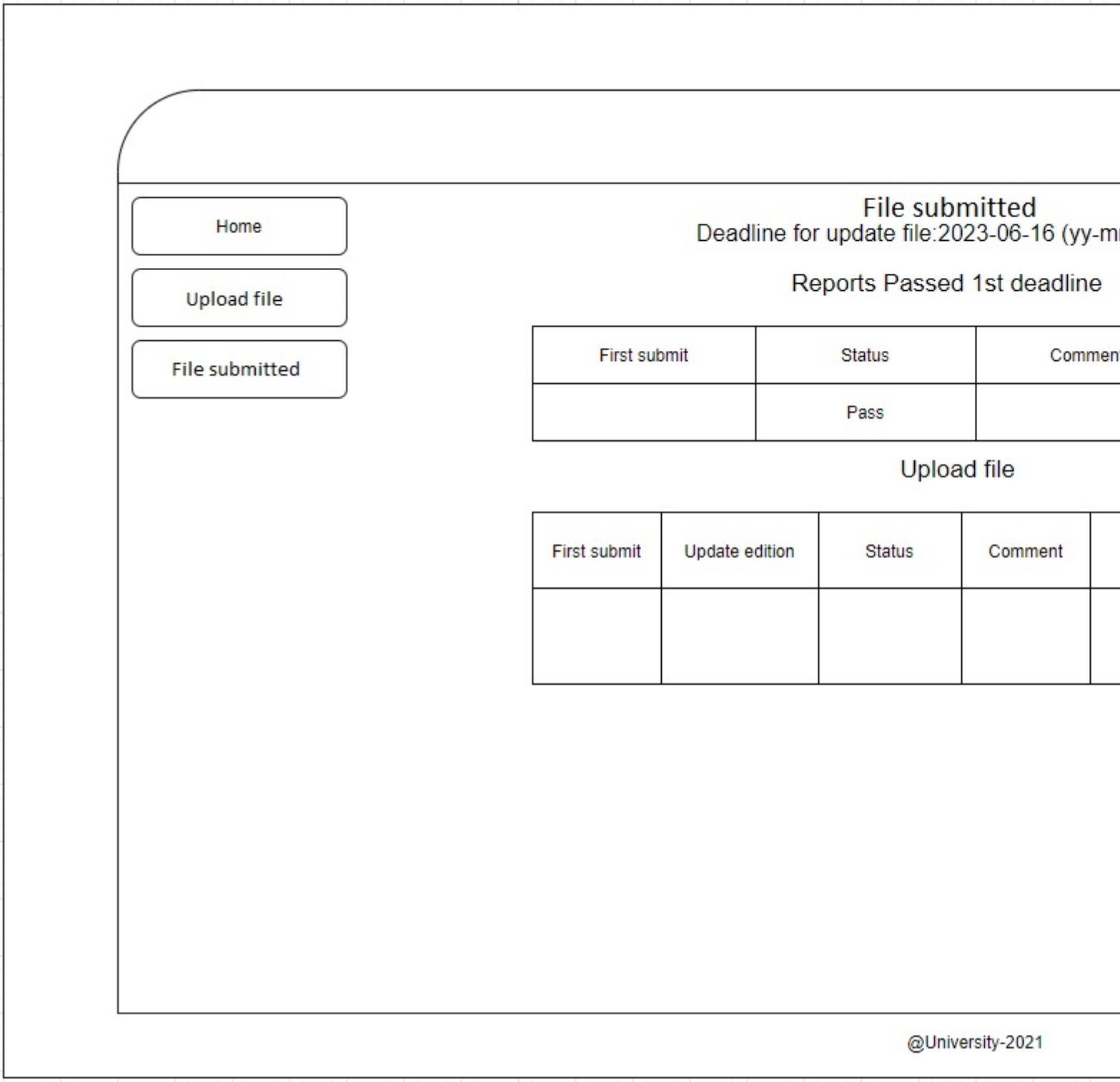

Figure 23:Wireframe file submitted

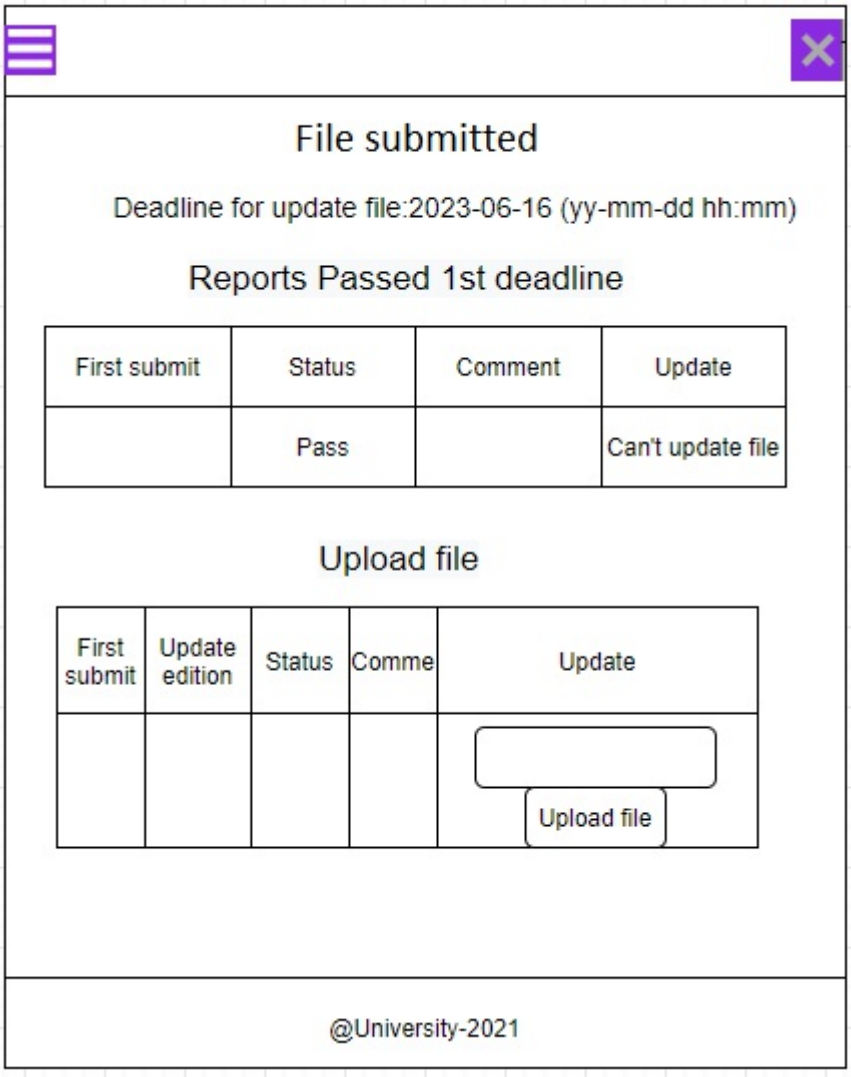

Figure 24:Wireframe file submitted mobile

It shows the deadline for the report, table 1 shows the reports that have passed and the file could not be uploaded. Table 2 shows the reports that have not been graded or failed. If there is a deadline, users can upload another file.

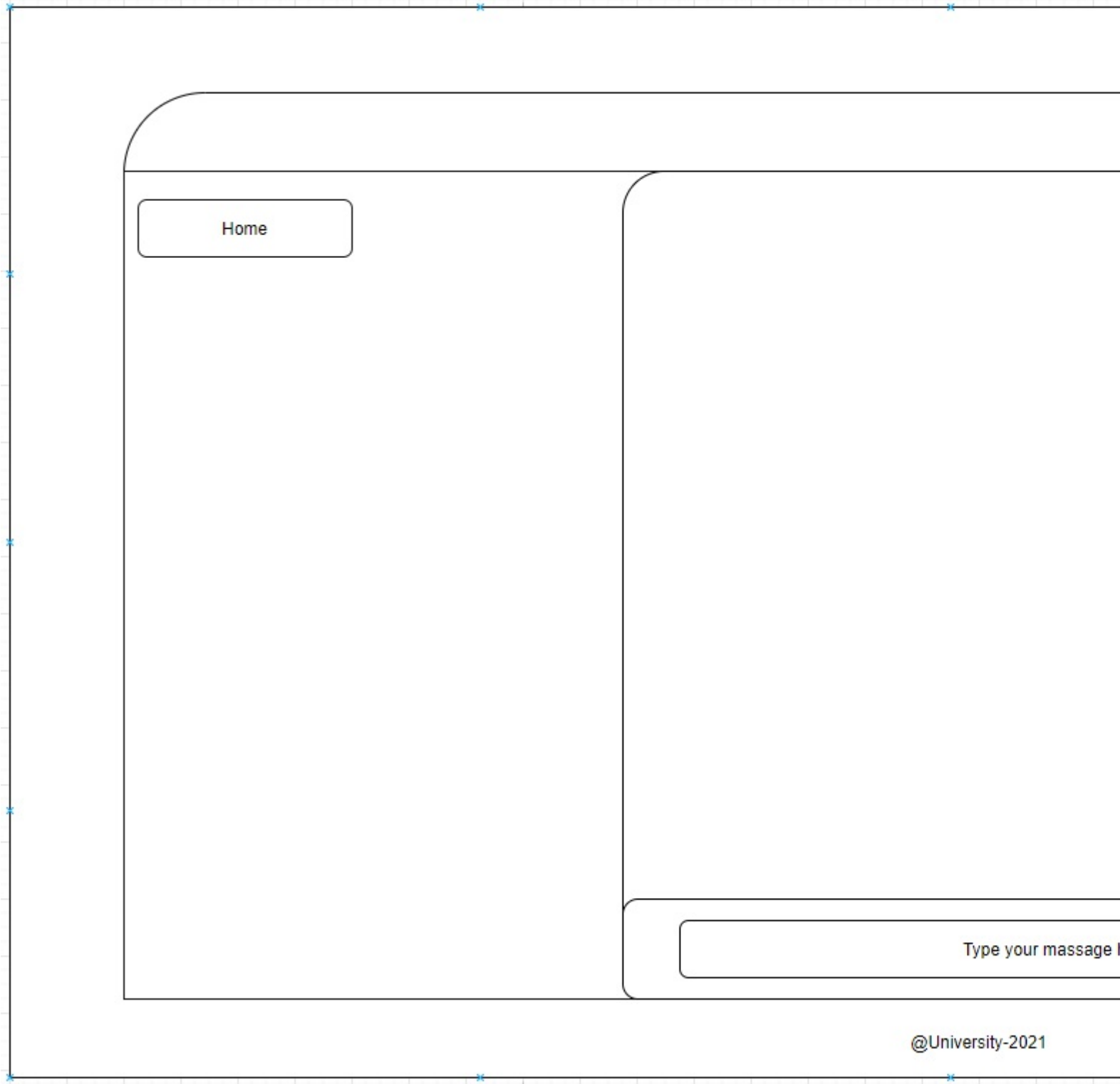

Figure 25: Wireframe chat box of Student and marketing coordinator

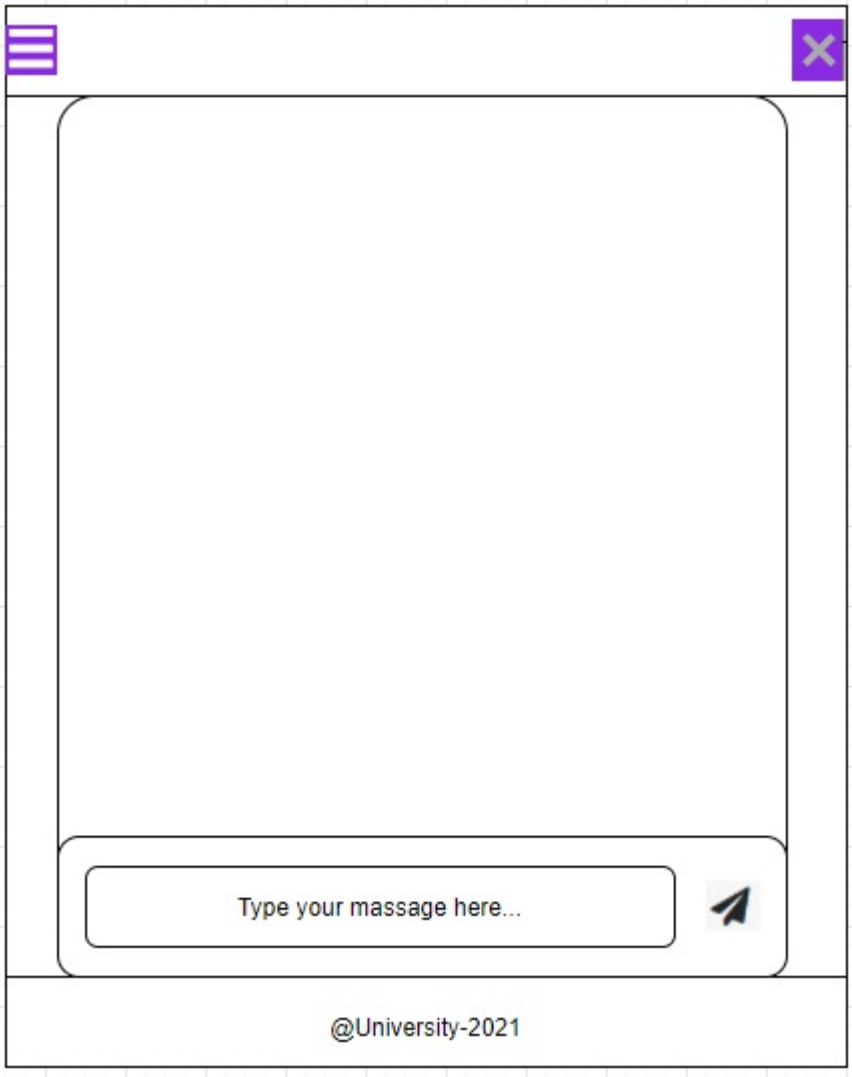

Figure 26: Wireframe chat box of Student and marketing coordinator mobile

Students and coordinators can contact each other through the chat box. The text chat section has: type your massage here to help users know where to write the chat box. Next to it is an icon for the button to send the message.

Admin

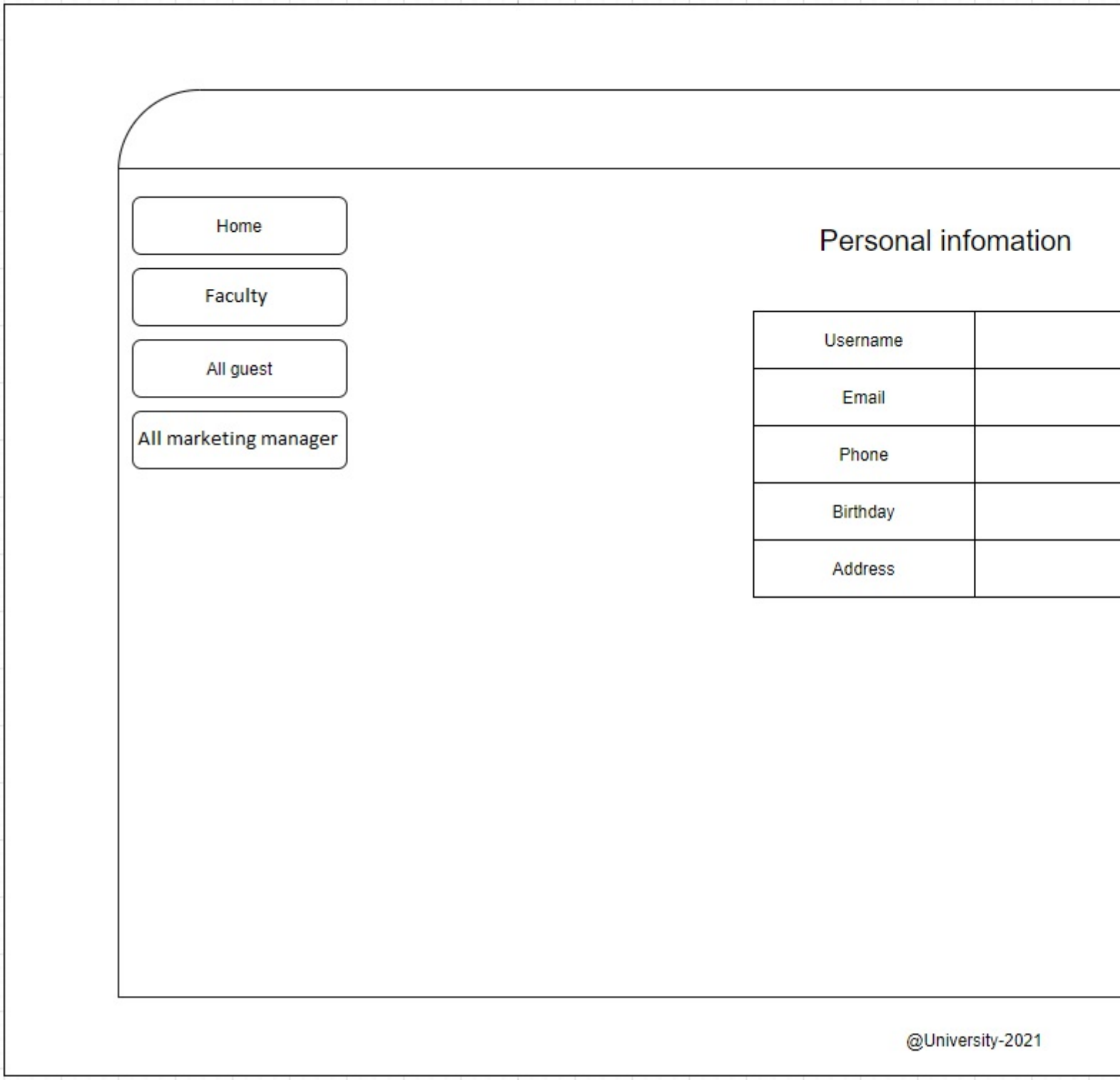

Figure 27:Wireframe home page of admin

Figure 28:Wireframe home page of admin mobile

The admin interface has a menu with the functions: home, faculty, all guest, and all marketing manager. Home content will display the user's personal information. The web framework is designed similarly, the content will contain different content.

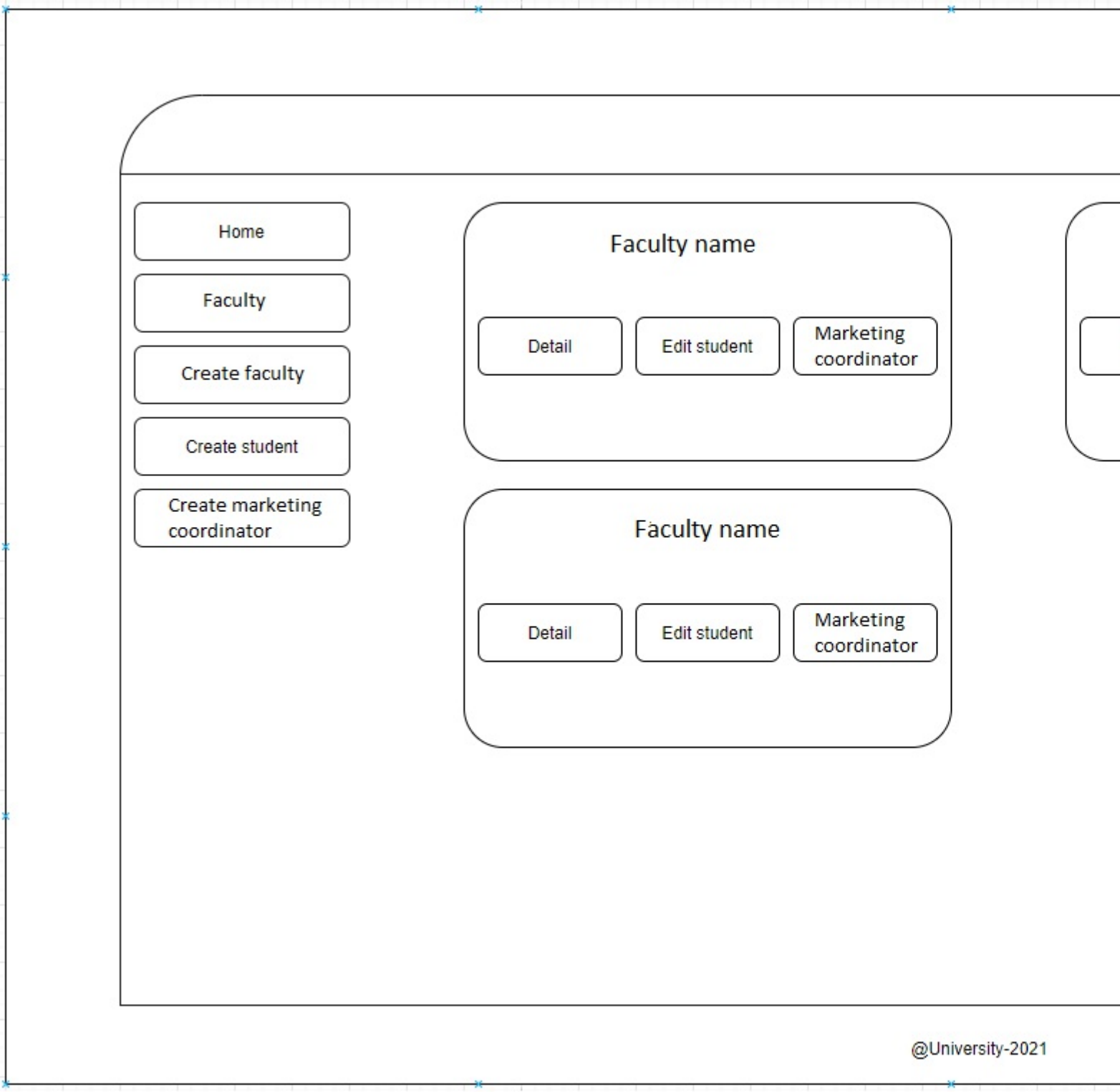

Figure 29:Wireframe all faculty of admin

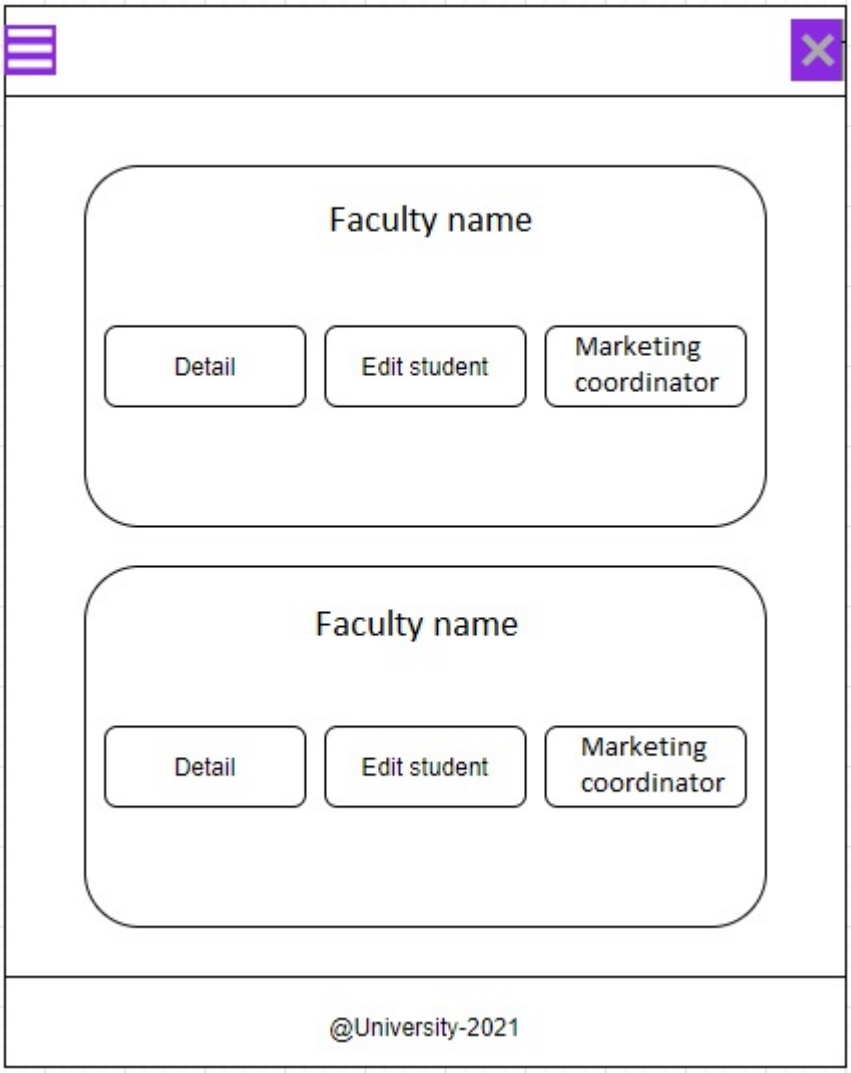

Figure 30:Wireframe all faculty of admin mobile

In faculty, it will display the faculty. The faculty will be wrapped into a frame and functions are placed in the button so that the user can observe and understand its usage and functions.

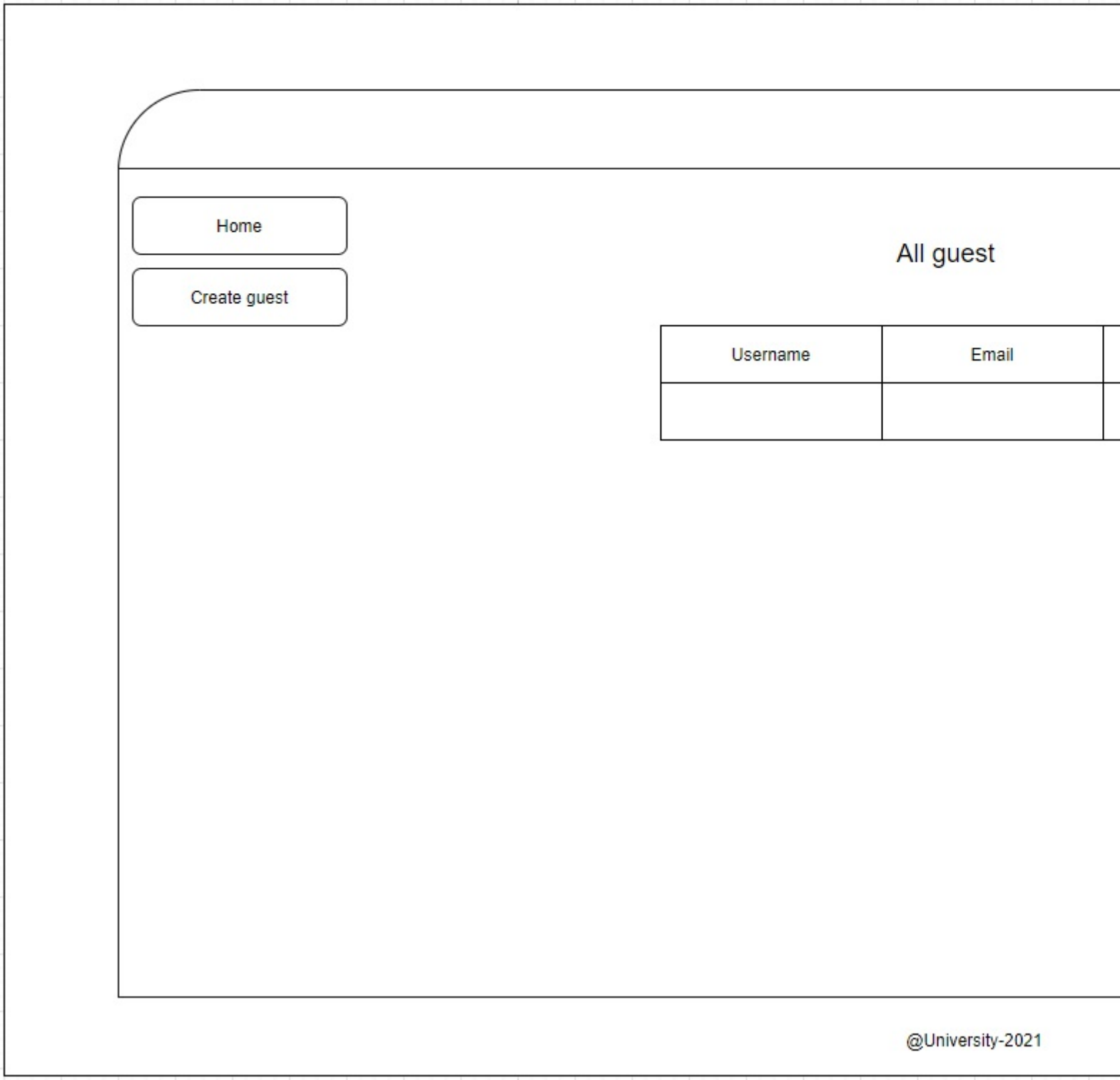

Figure 31:Wireframe all guest of admin

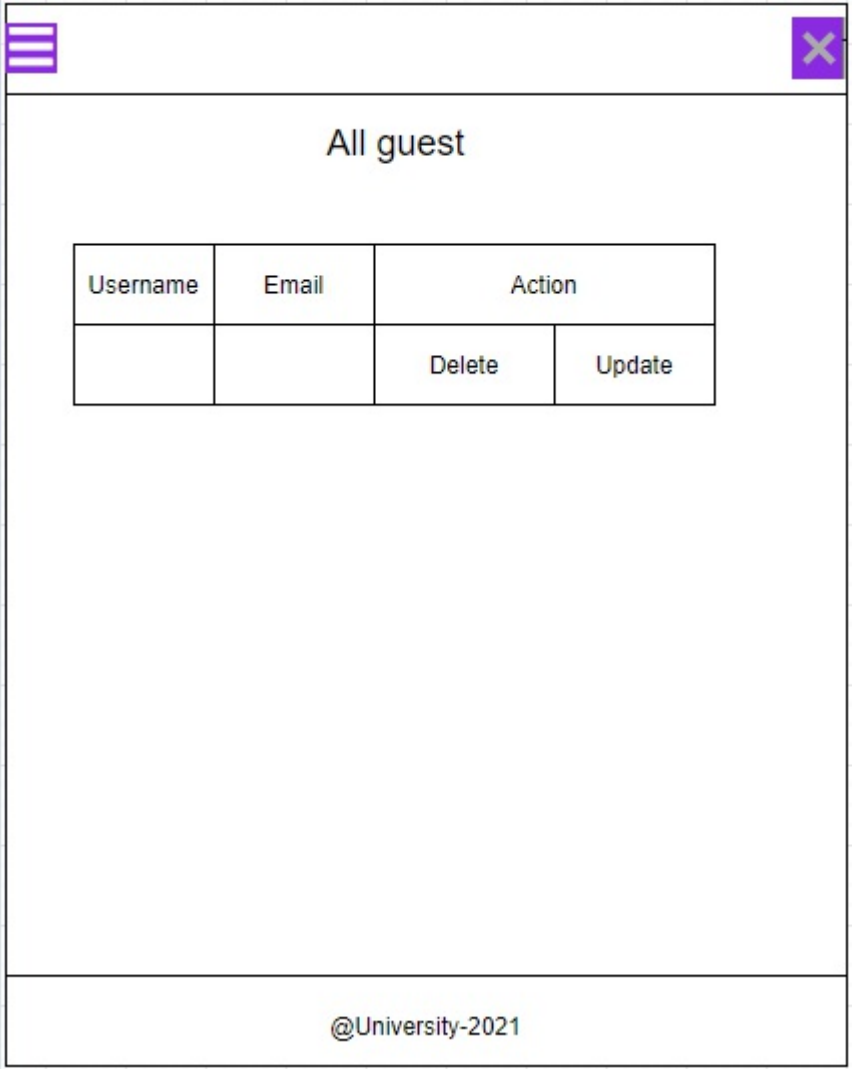

Figure 32:Wireframe all guest of admin mobile

Show all guest and marketing managers are included in the table for easier management.

Marketing manager:

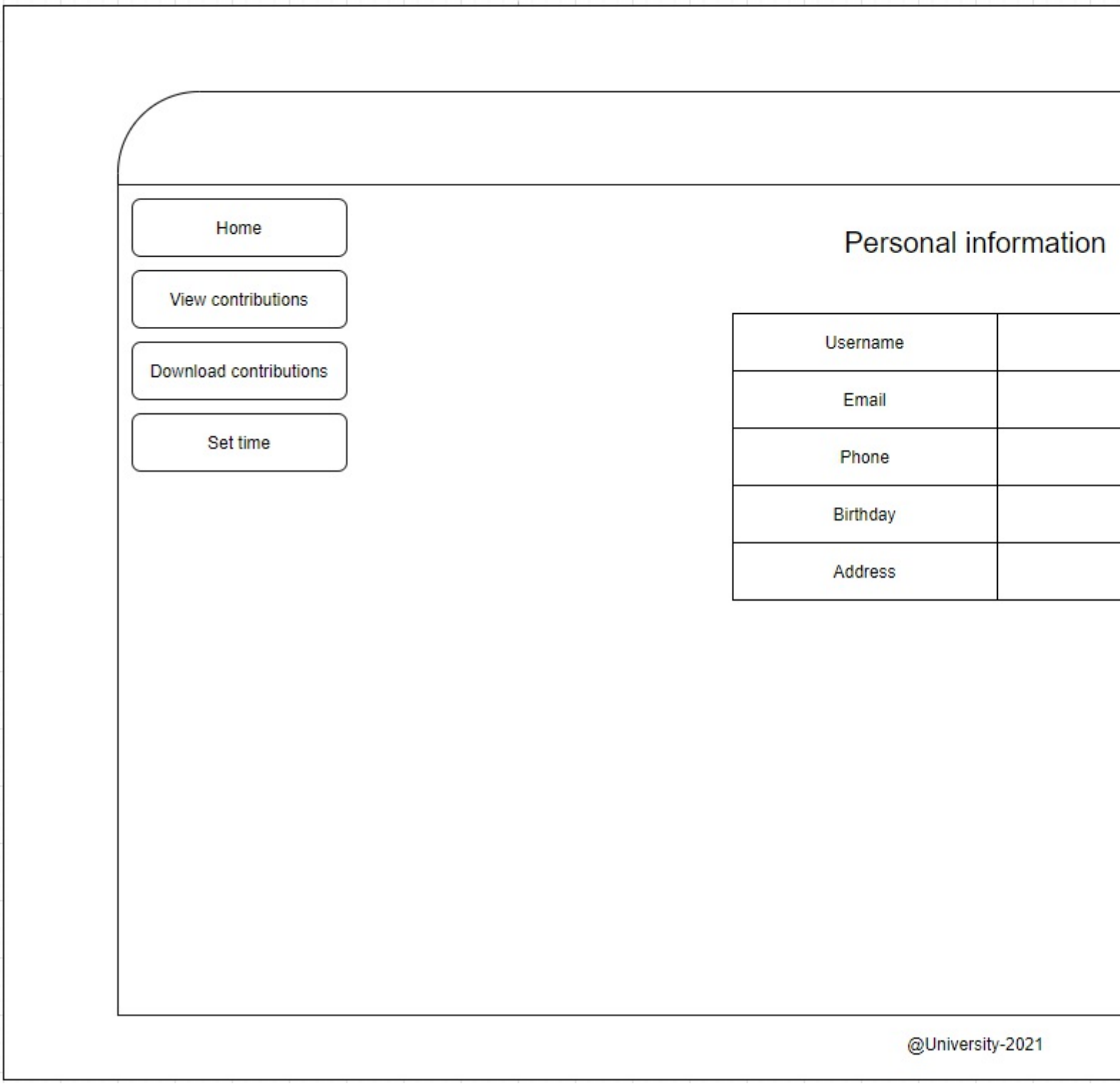

Figure 33:Wireframe home page of marketing manager

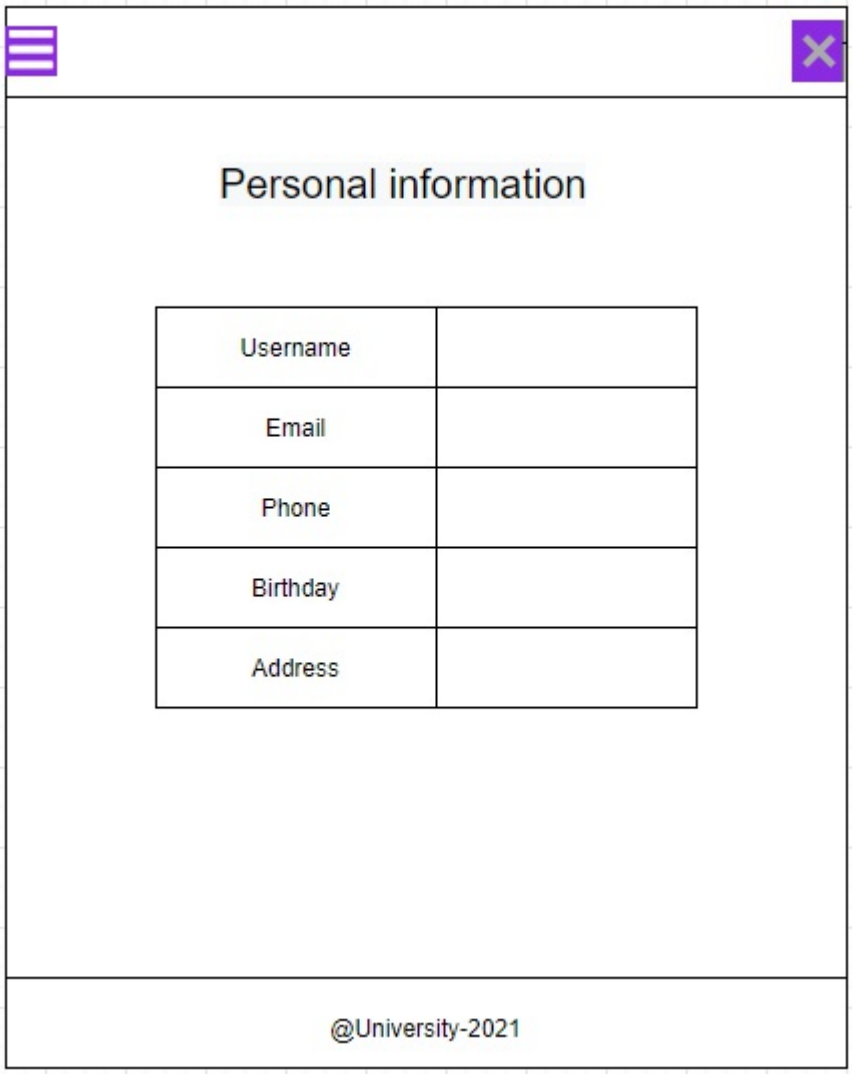

Figure 34:Wireframe home page of marketing manager mobile

The marketing manager has functions such as home, view contributions, download contributions, and set times.

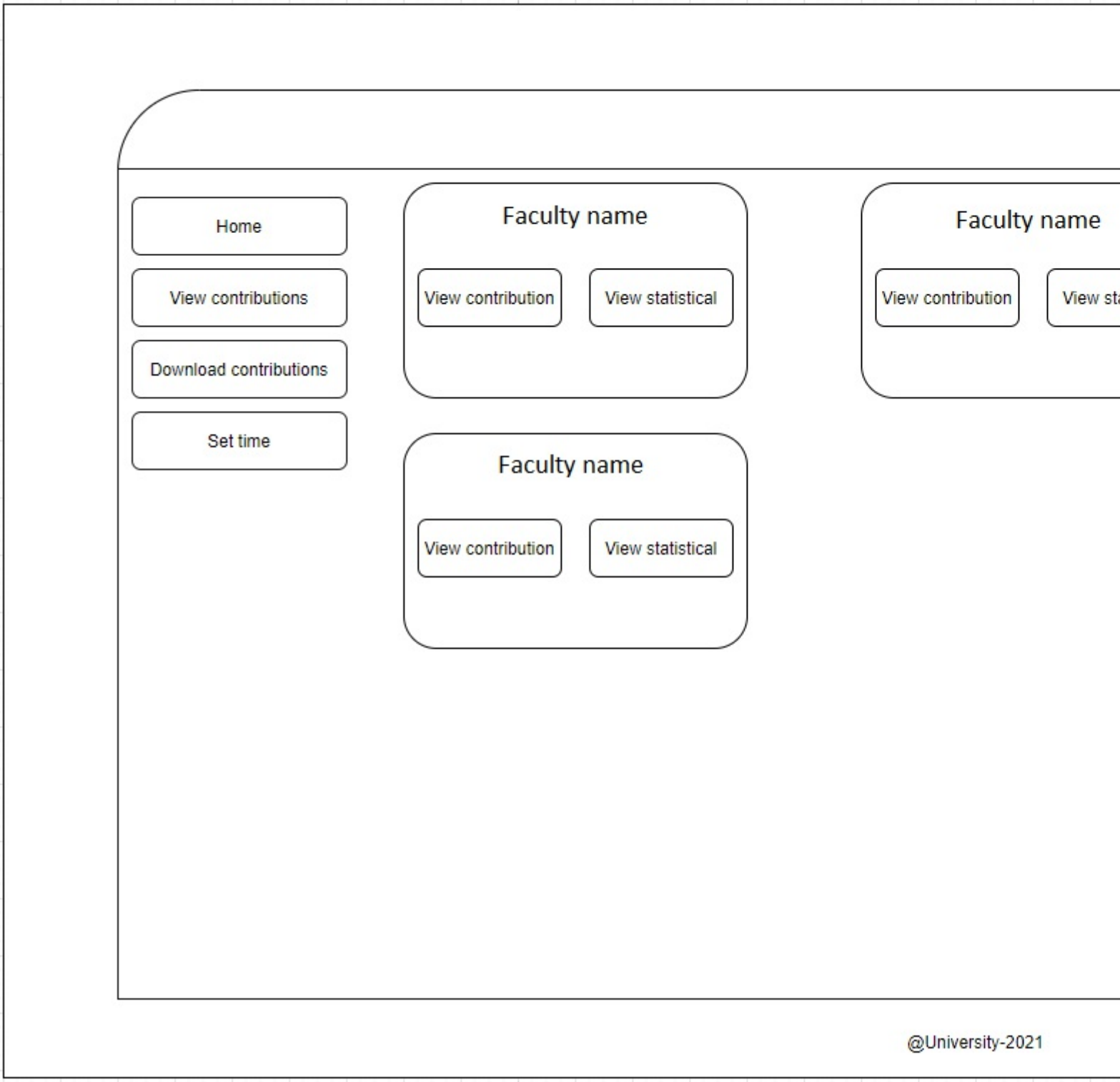

Figure 35:Wireframe all faculty of marketing manager

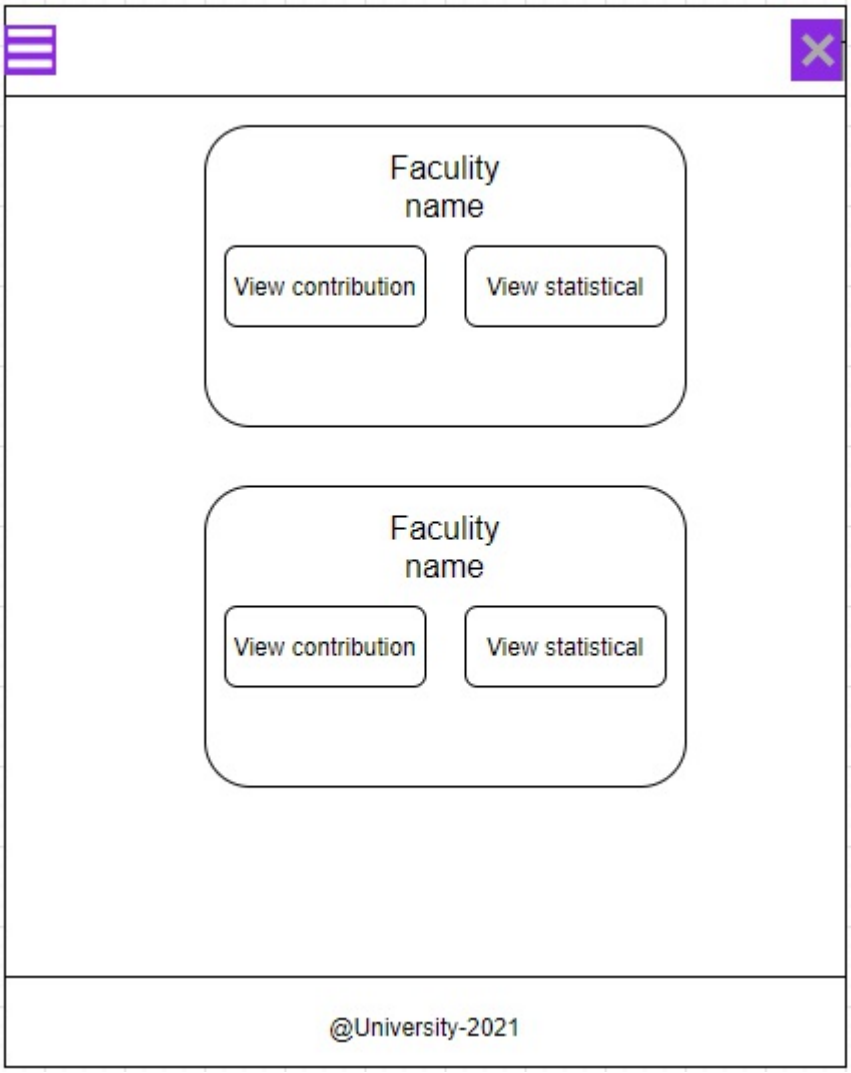

Figure 36:Wireframe all faculty of marketing manager mobile

In faculty members do not have the view contribution and view statistical functions to use by the user who wants to view the faculty. All functions are located in buttons for easy viewing.

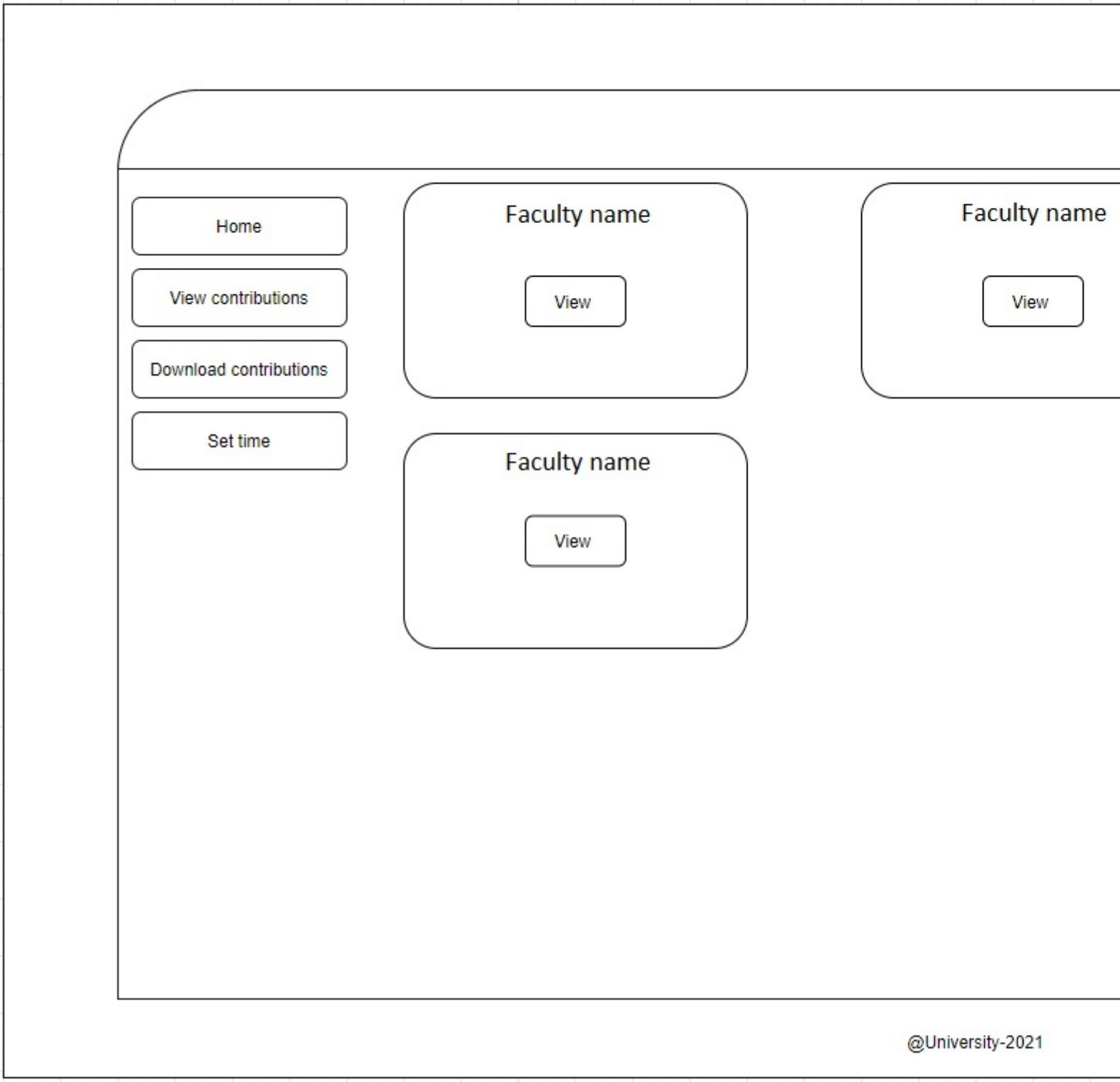

Figure 37:Wireframe download contributions of marketing manager

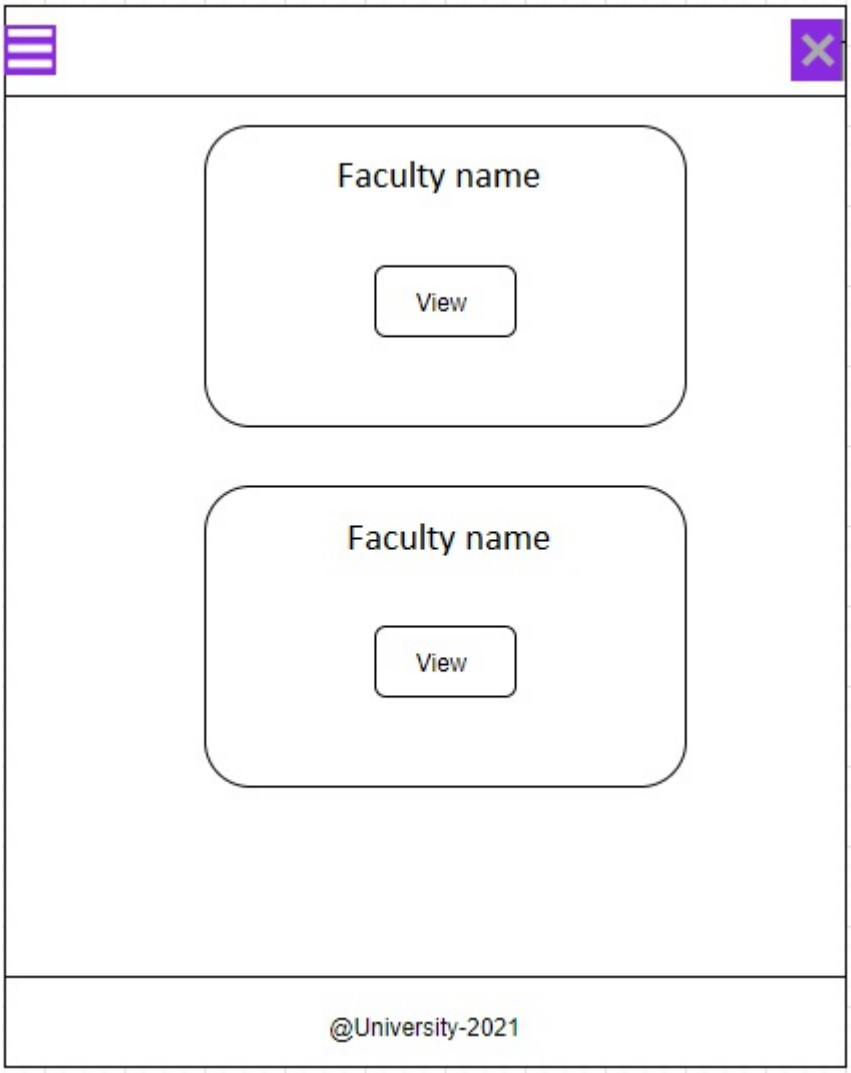

Figure 38:Wireframe download contributions of marketing manager mobile

These are download contributions.

Figure 39:Wireframe set time of marketing manager

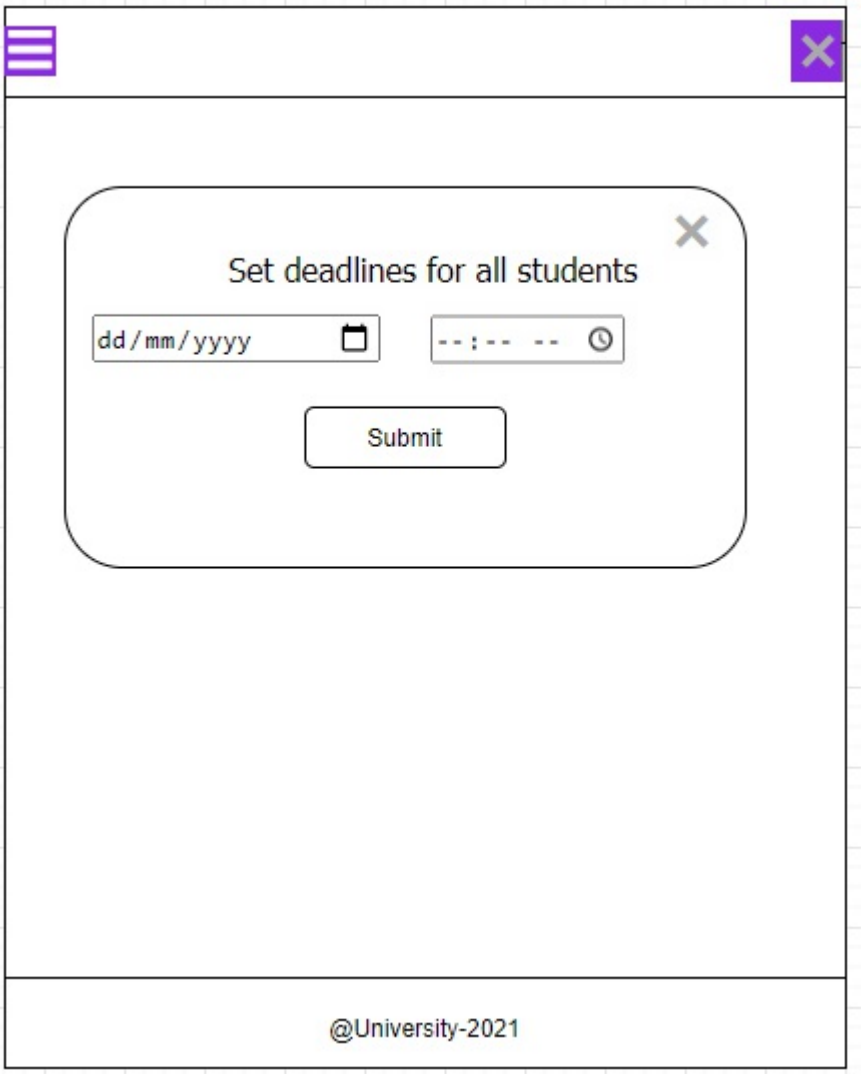

Figure 40:Wireframe set time of marketing manager mobile

The marketing manager can set a time for the report. The date entry uses the date tag input for users to easily enter the correct category. The set time is also designed in a popup format that is both easy to use and saves an area.

Guest:

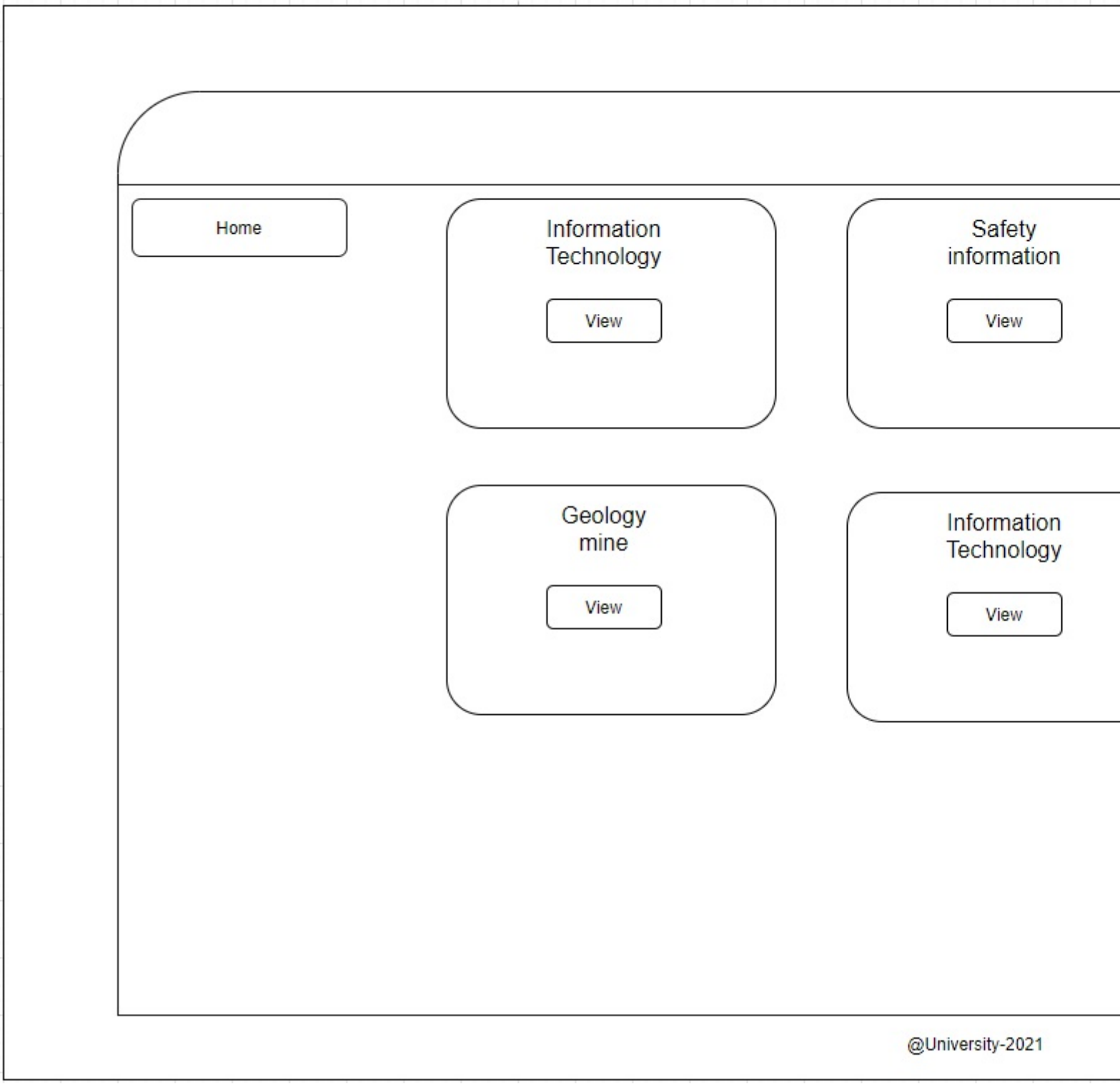

Figure 41: Wireframe view faculty of guest

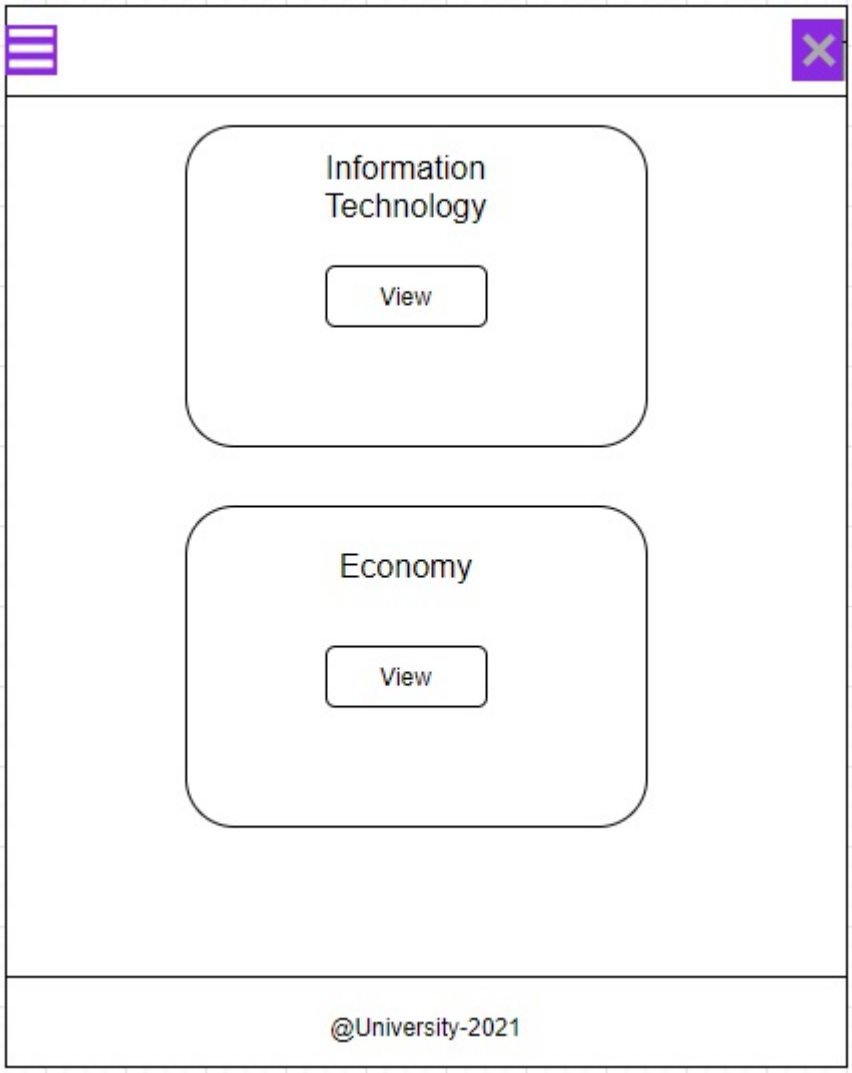

Figure 42:Wireframe view faculty of guest mobile

It will display the moderated faculty and guests can choose to view the faculty's posts.

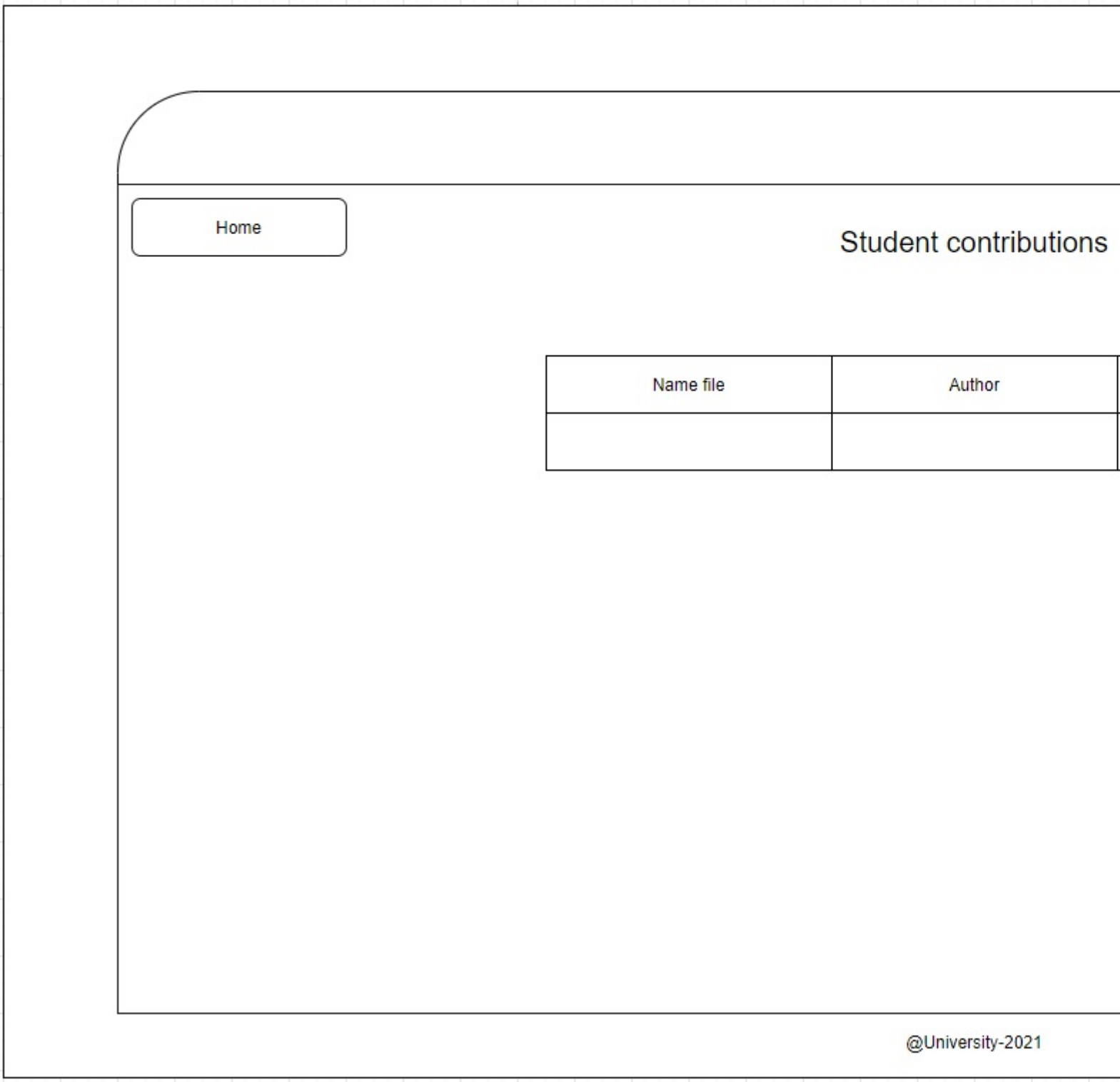

Figure 43:Wireframe student contributions of guest

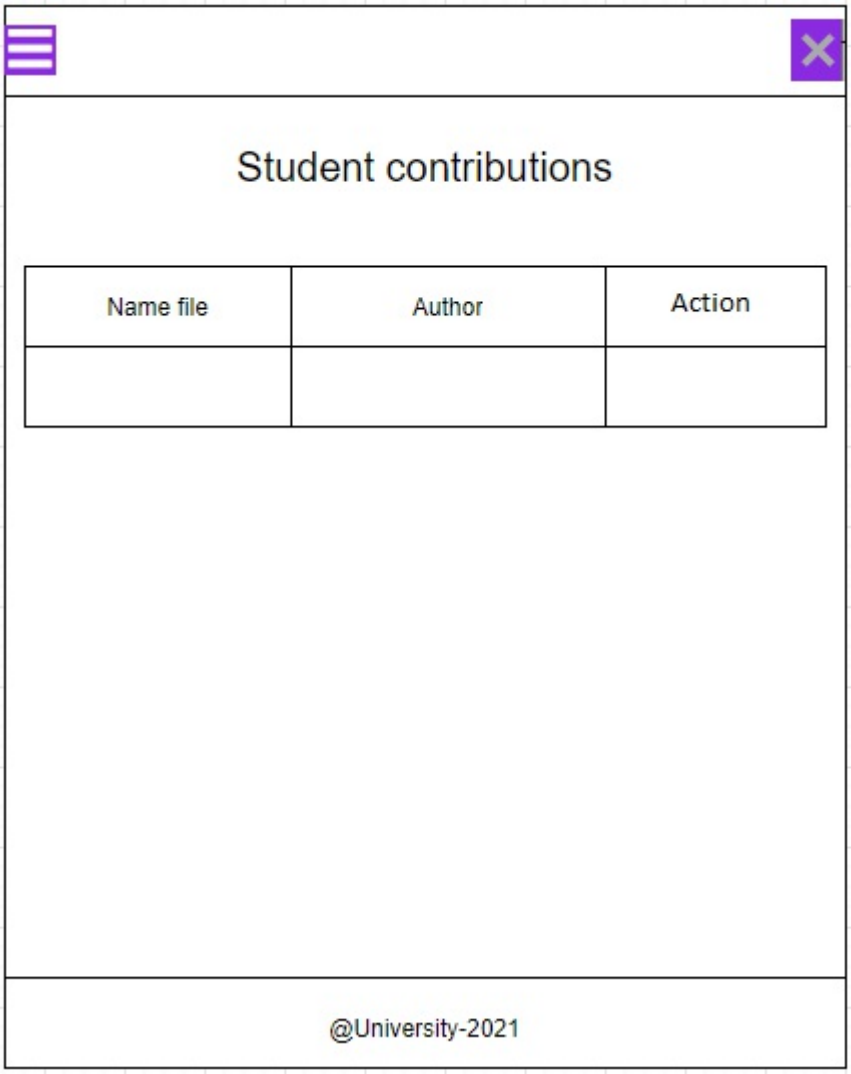

Figure 44:Wireframe student contributions of guest mobile

After selecting the faculty, all faculty posts will be displayed and it is stored in the form table for easy viewing by the user.

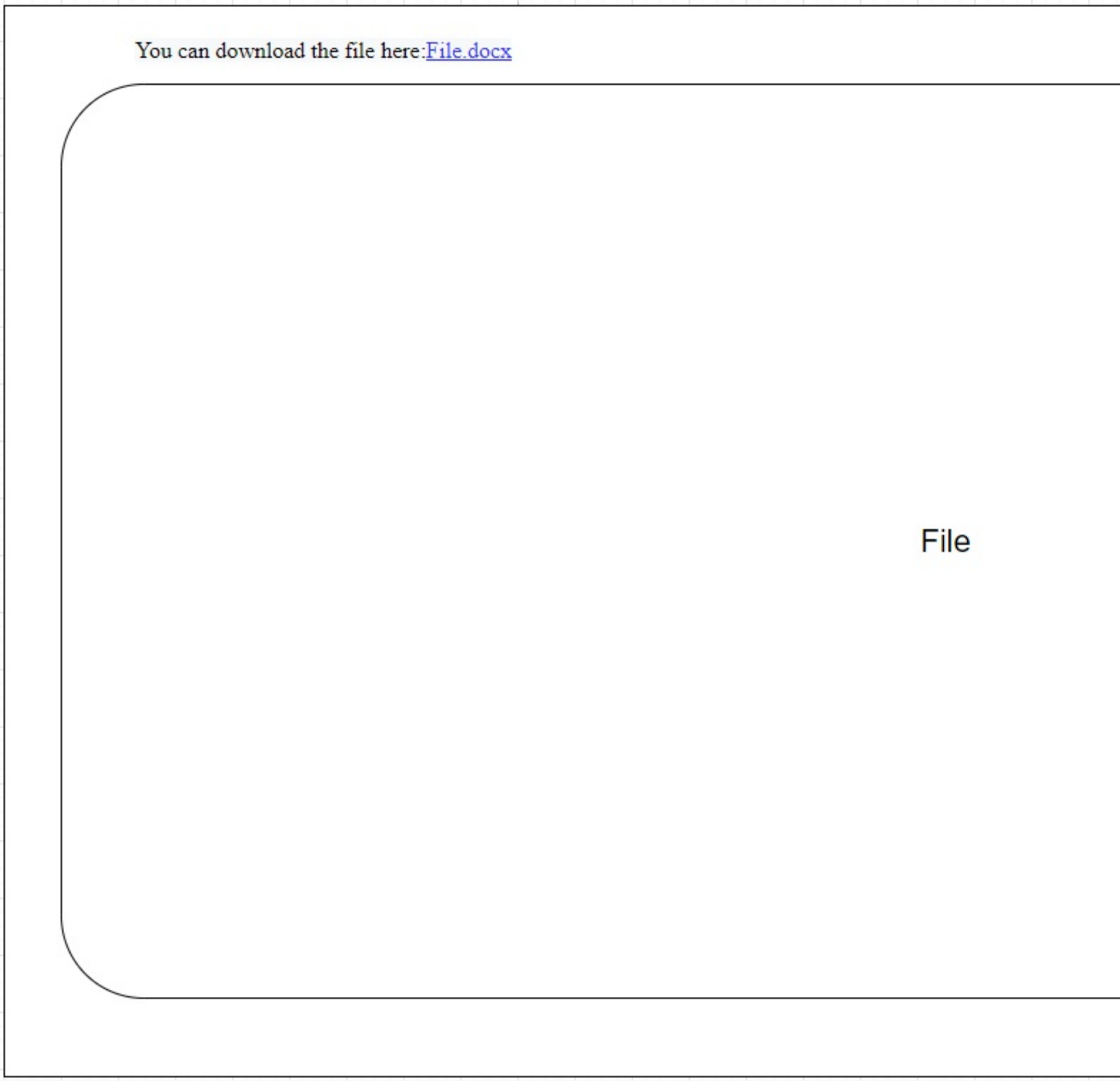

Figure 45:Wireframe view report

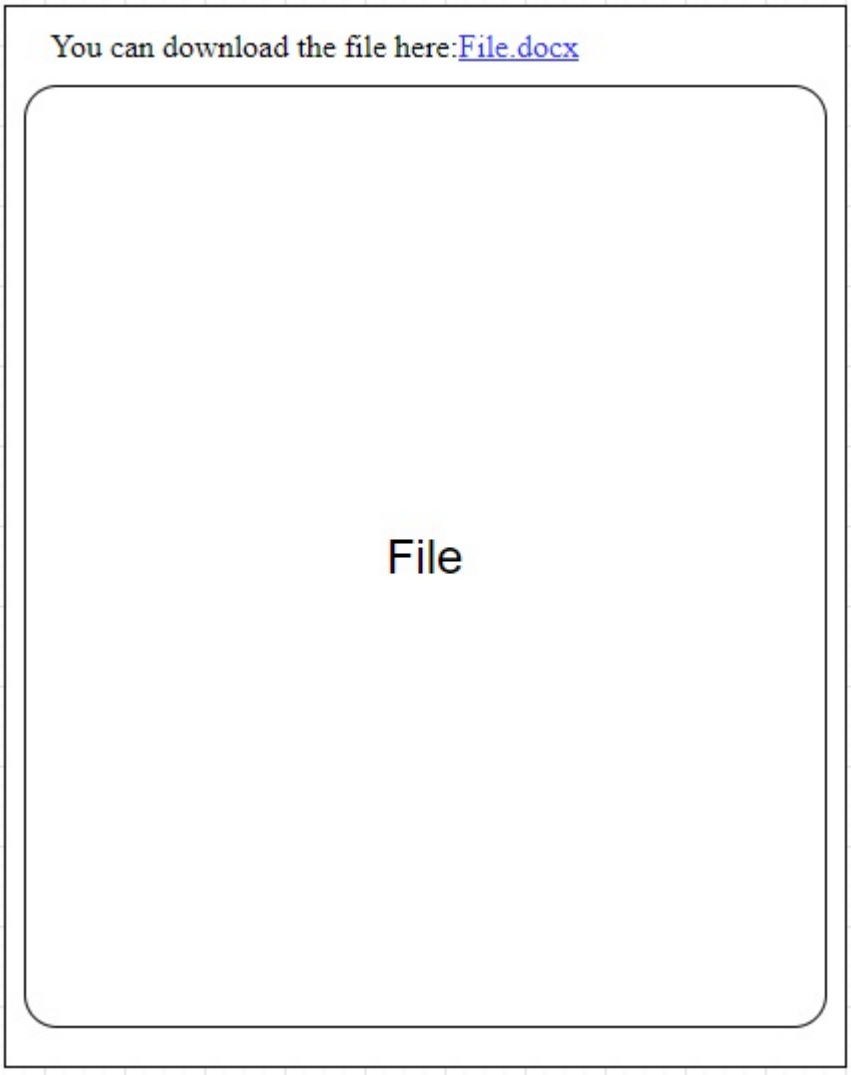

Figure 46:Wireframe view report mobile

The top section is a downloadable file with a title right next to it. Next are the files and images for the guests to view.

Marketing coordinator

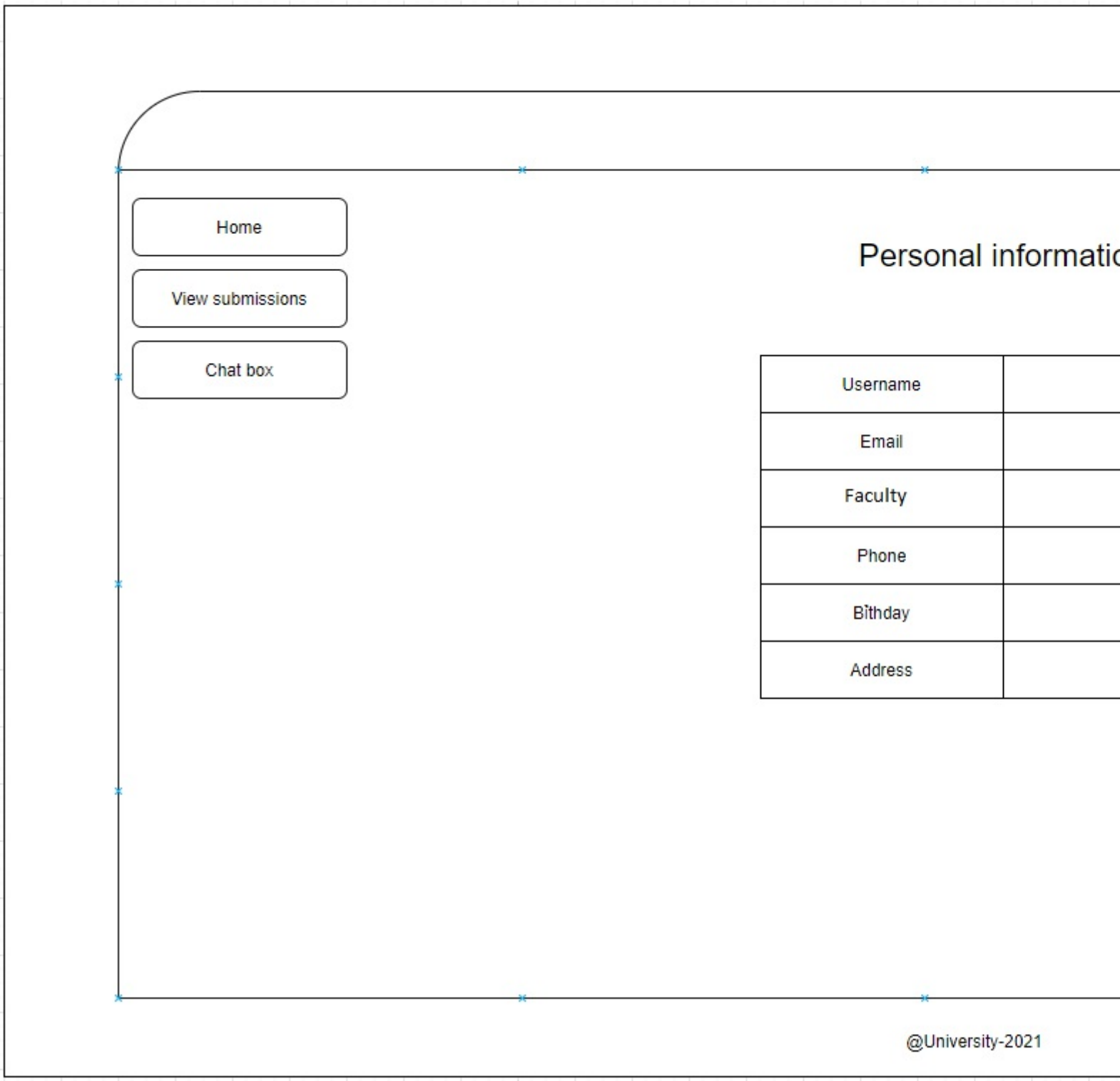

Figure 47:Wireframe home page of marketing coordinator

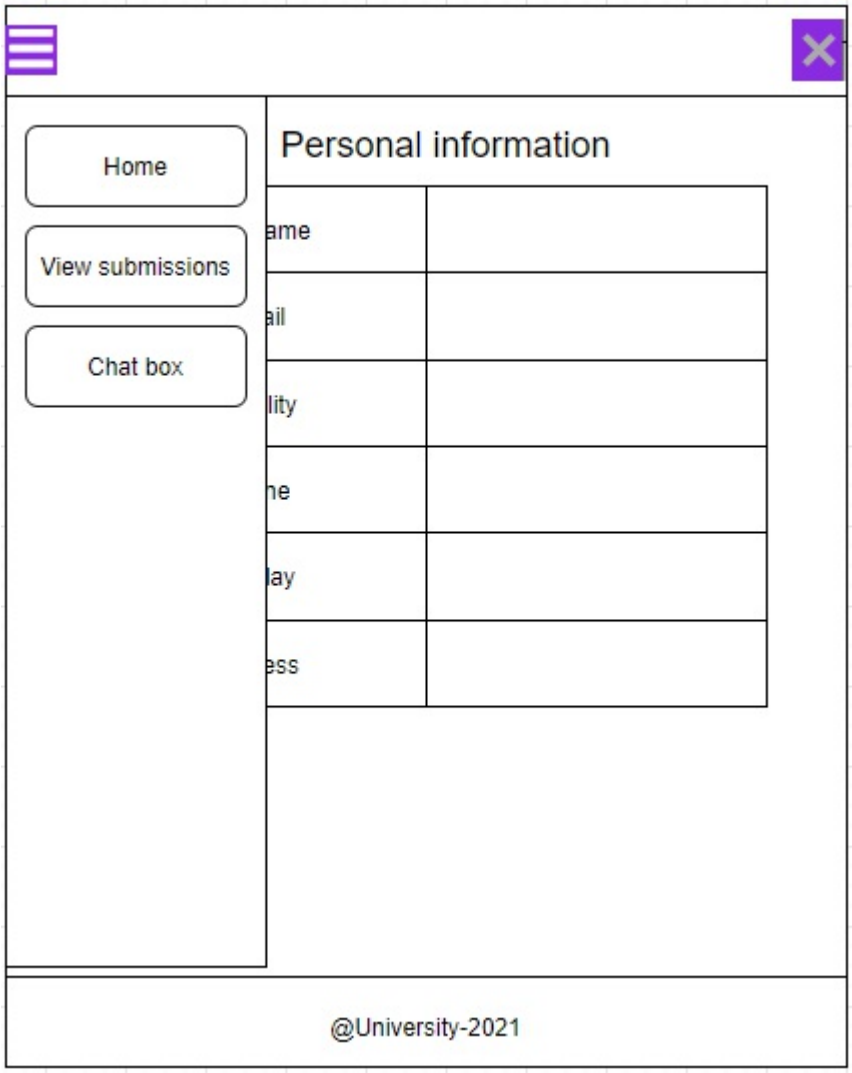

Figure 48:Wireframe home page of marketing coordinator mobile

The marketing coordinator has the following functions: Home, view submissions, chat box.

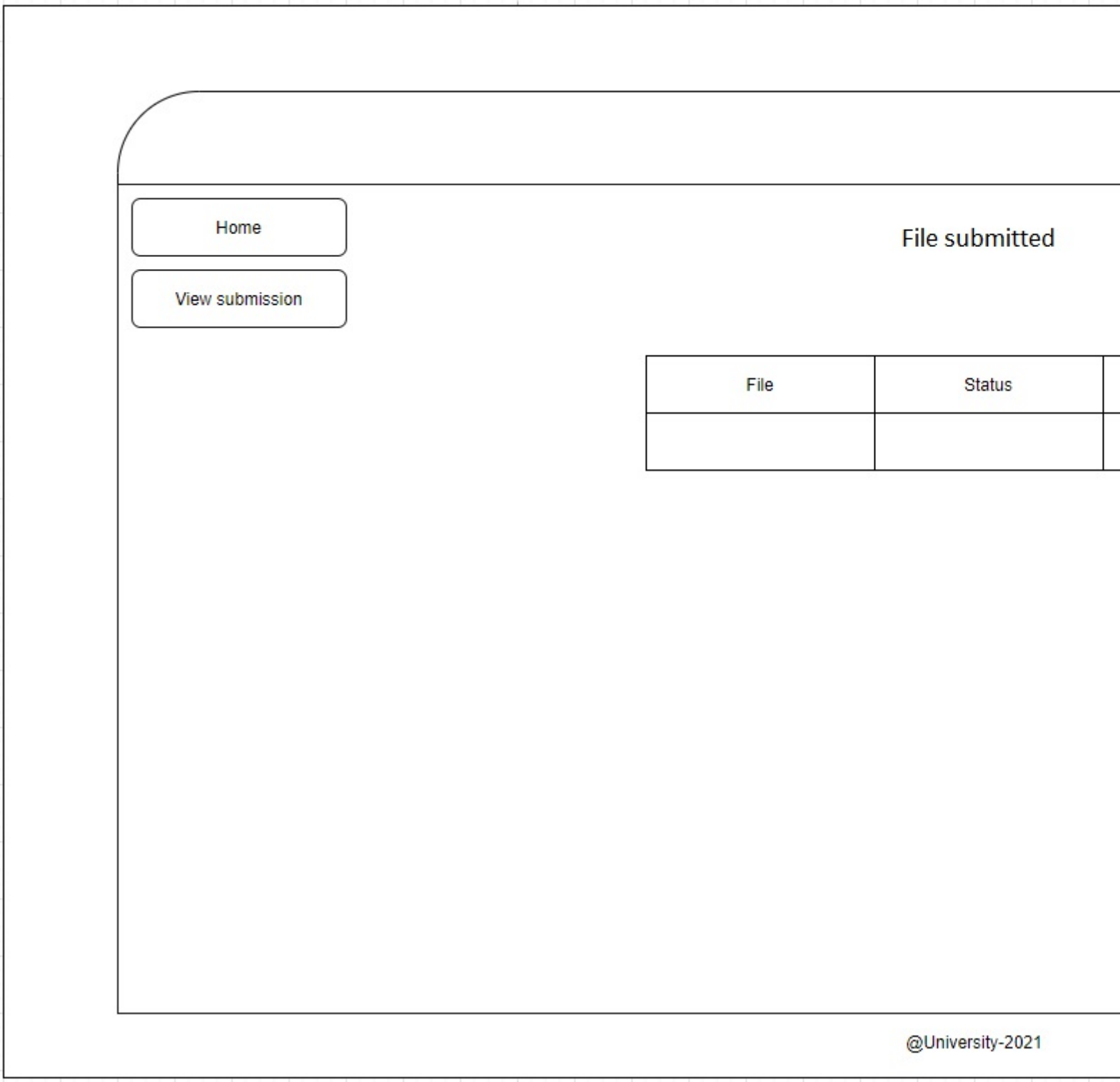

Figure 49:Wireframe file submitted of marketing coordinator

Figure 50:Wireframe file submitted of marketing coordinator mobile

The submitted file section is placed in the table for easier viewing by the user.

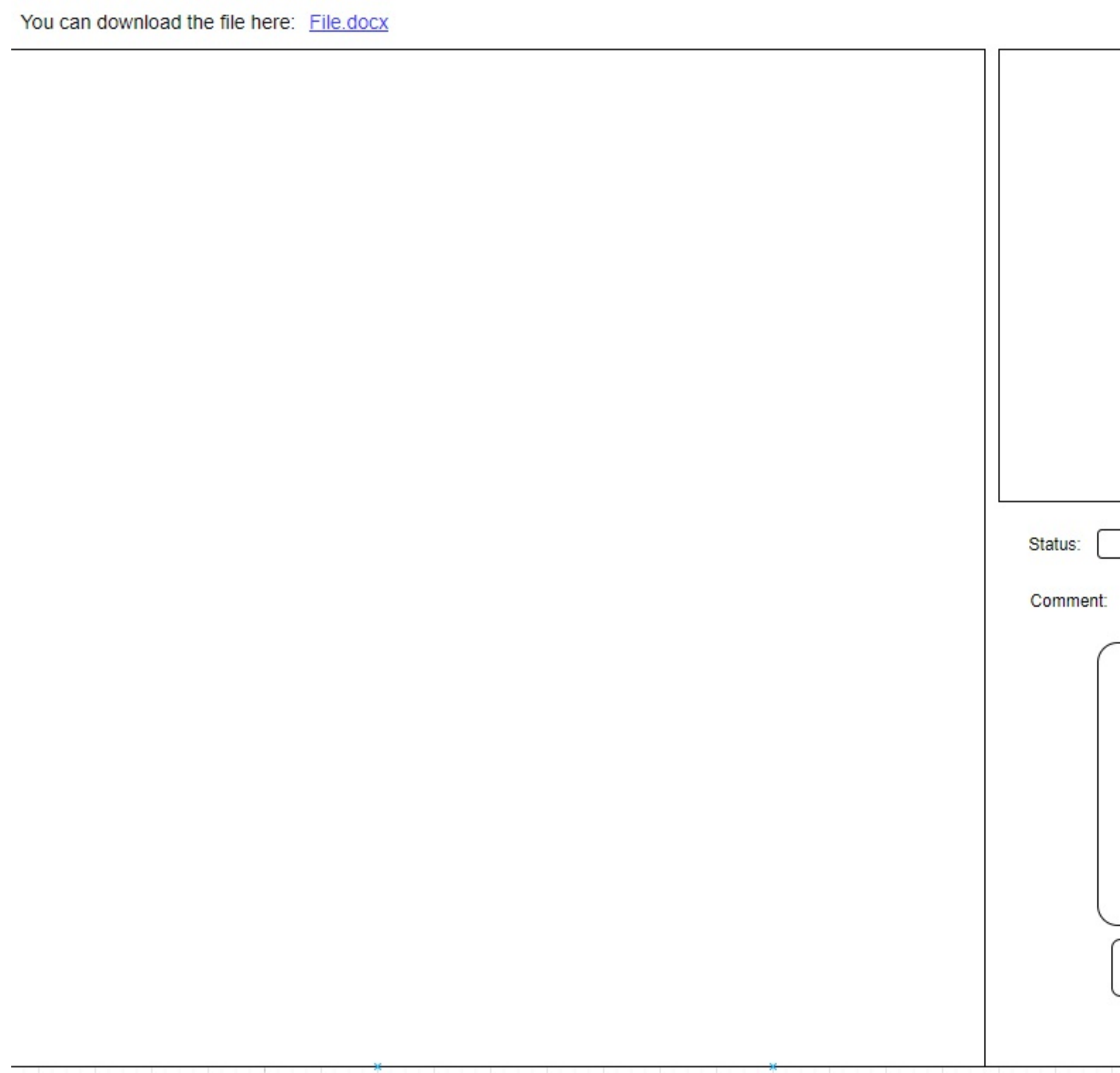

Figure 51:Wireframe view and evaluate of marketing coordinator

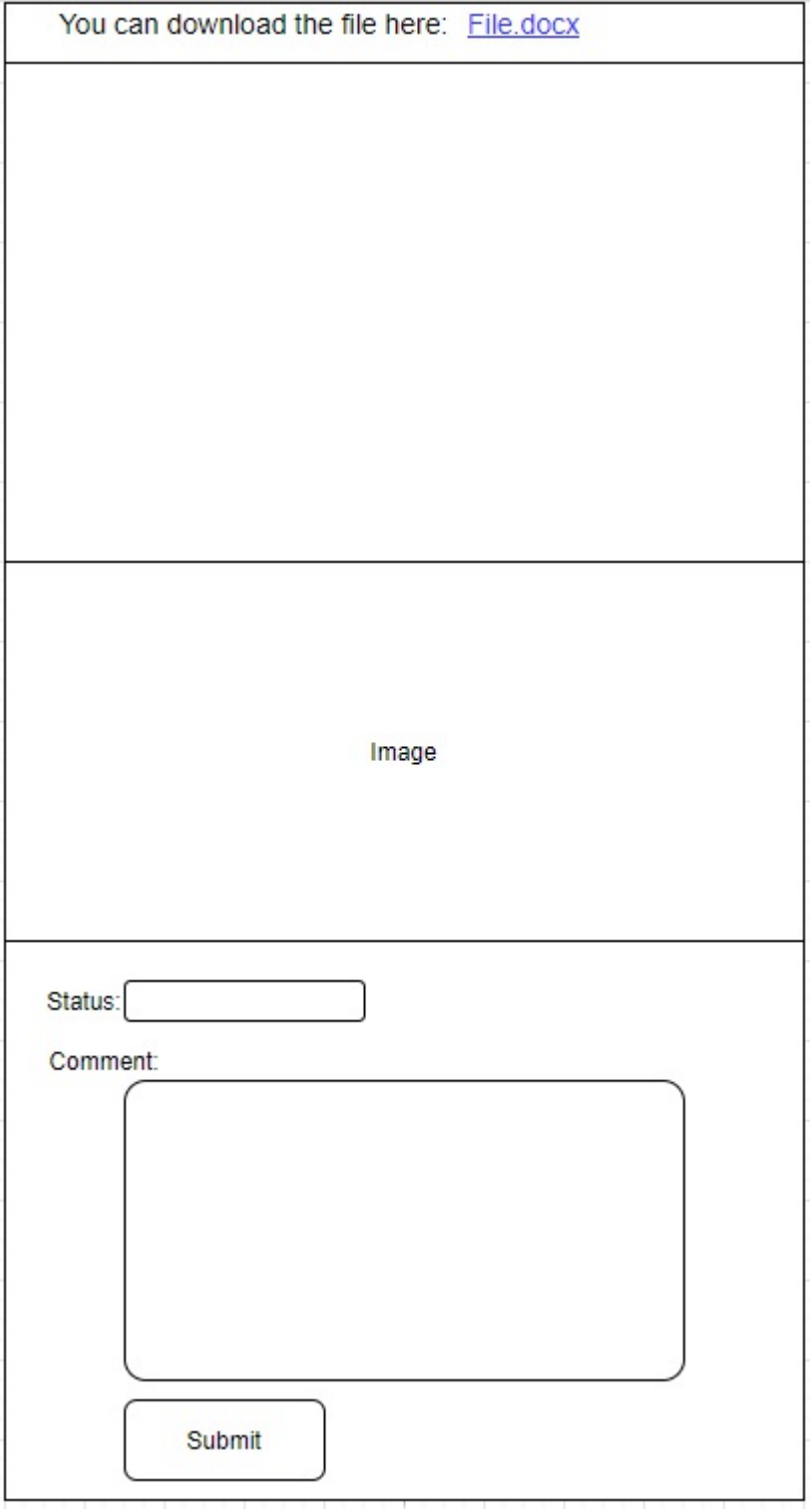

Figure 52:Wireframe view and evaluate of marketing coordinator mobile

The Marketing coordinator can view the student's uploaded files and rate them. The rating: status shows the current score and can choose the point to change without having to manually enter.

Interface

The colors used on the web interface are all light and gentle colors that do not cause inhibition.

When switching from desktop to mobile the menu is hidden to save screen space. The fonts have all been adjusted to suit the phone.

Buttons will add effects.

Login

First, when a user visits the website, the login page will be displayed first. Then the user will enter the username and password,

the system will check the username and password to provide an interface for each object above. It is suitable for both computers and mobile.

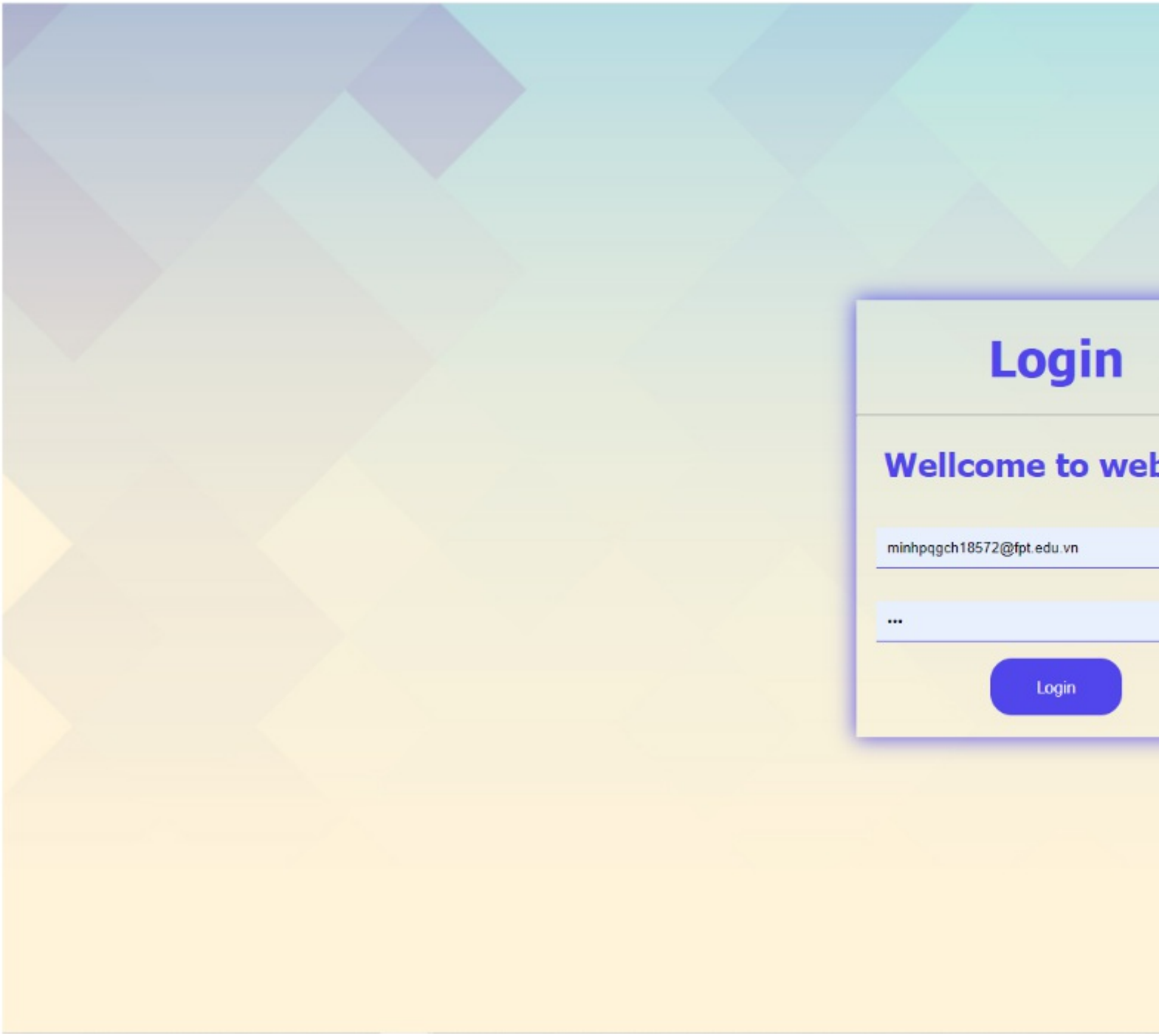

Figure 53:Login

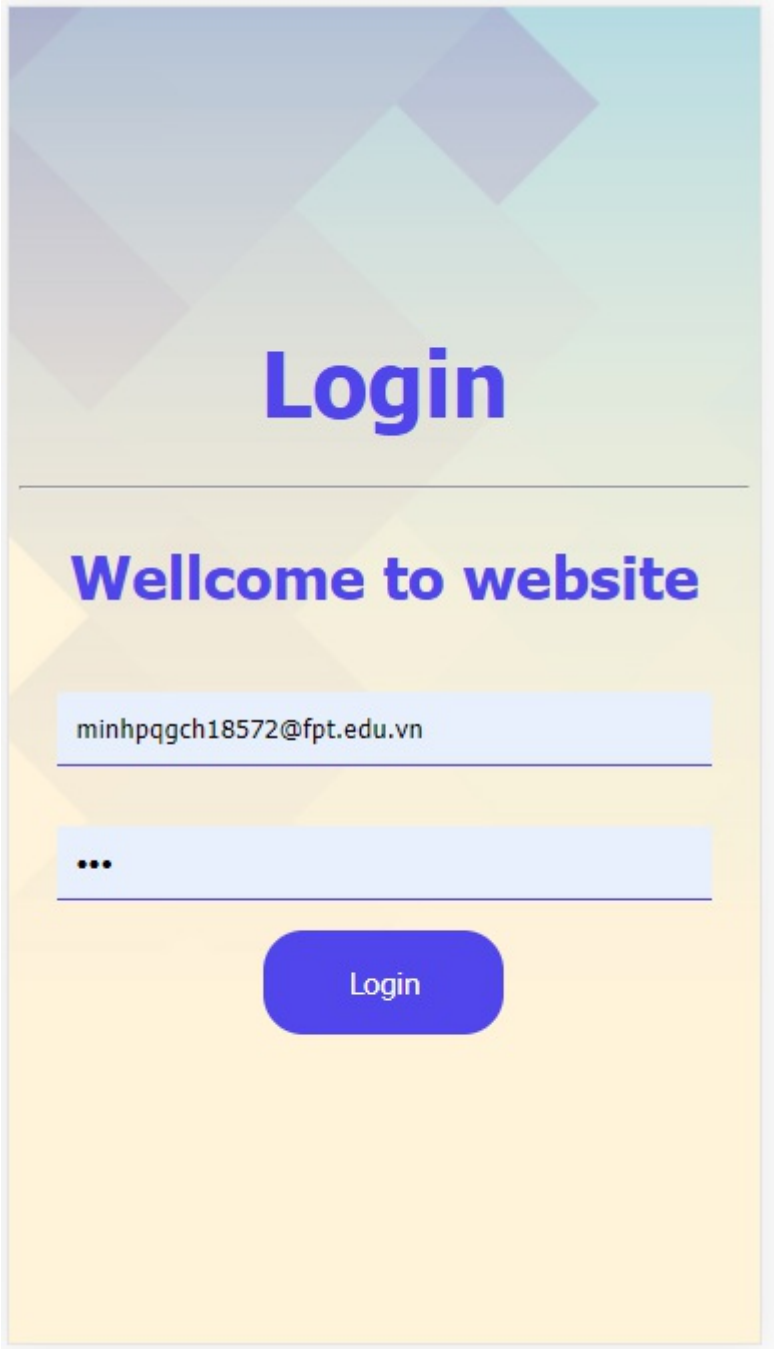

Figure 54:Login mobile

Student

The layout of the interface is divided into 4 main parts: the header contains the "X" button to logout, the menu contains the functions that students can use and it is displayed on the left-hand side, the content contains the content, the last part. the same is the footer.

When entering the student interface, the menu will display student functions such as home, upload file, submitted file, and chat box. Content will display personal information of students including user name, email, faculty, Phone, Birthday, and address.

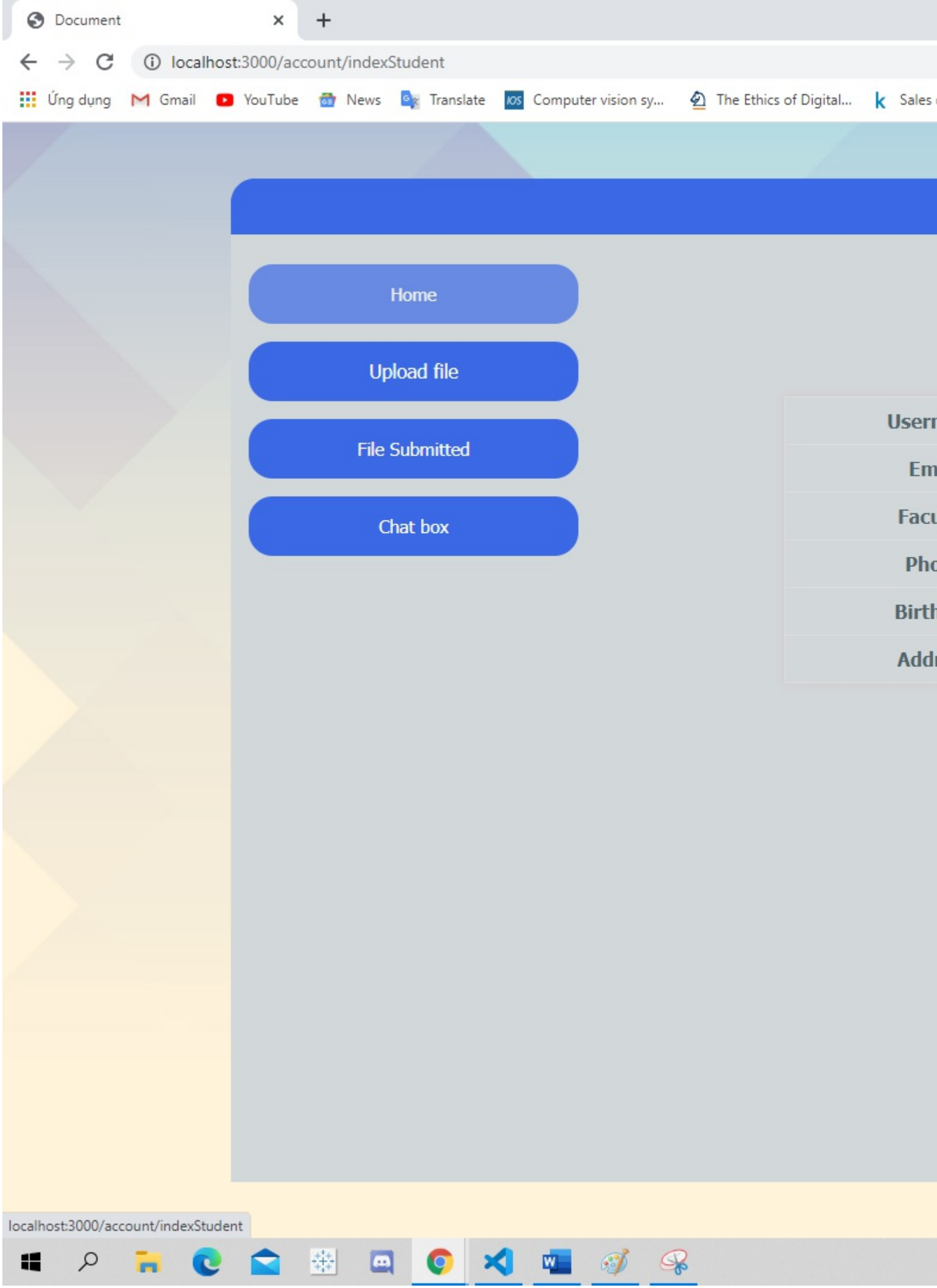

Figure 55:home page student

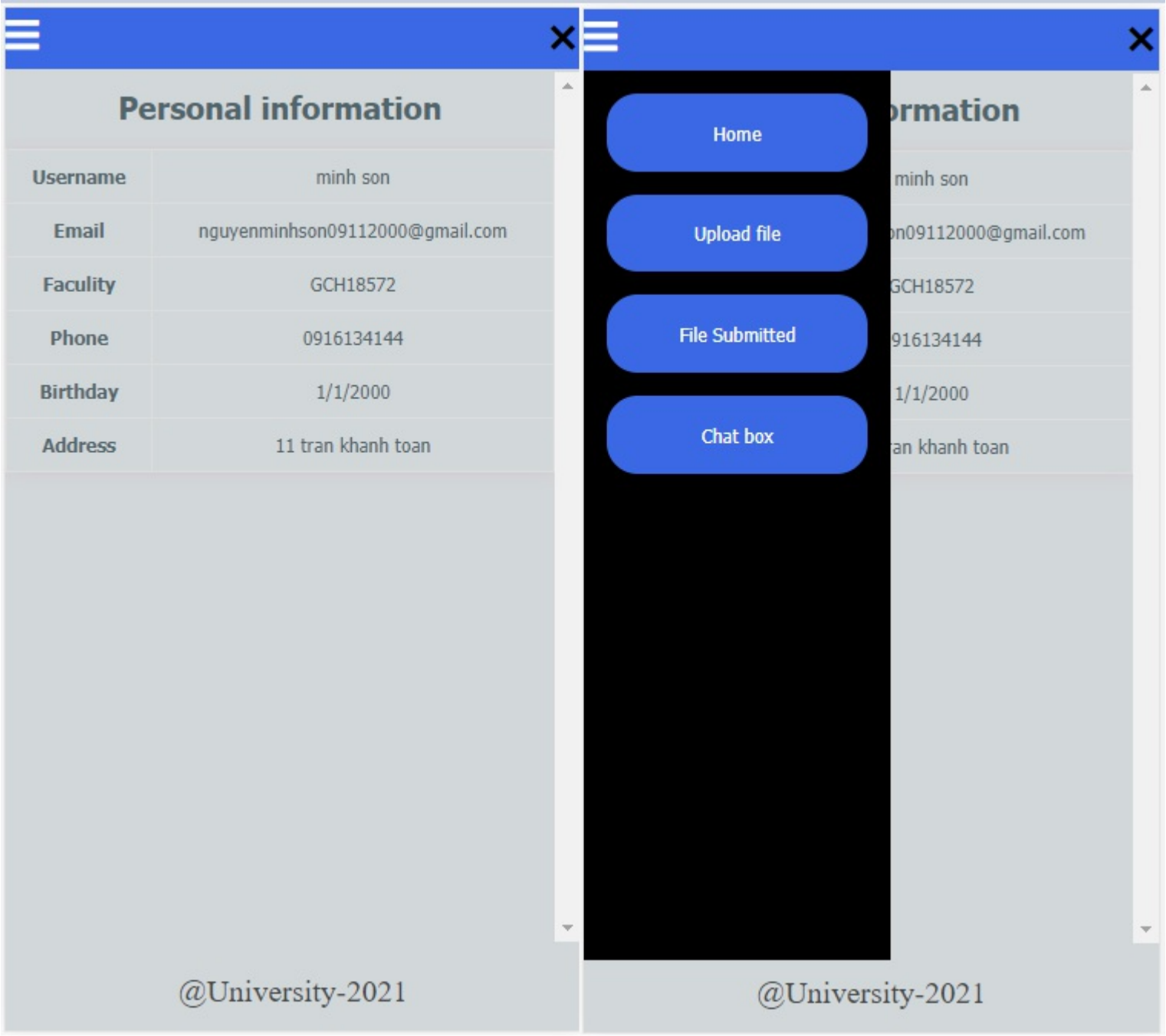

Figure 56:homepage student mobile

To be suitable for mobile, the menu section will be hidden and display an icon for the left-hand menu to save space for the mobile device. When you need to use the menu, just press on the icon for the menu, the menu will appear and when the user presses the icon of the menu, the menu will be hidden.

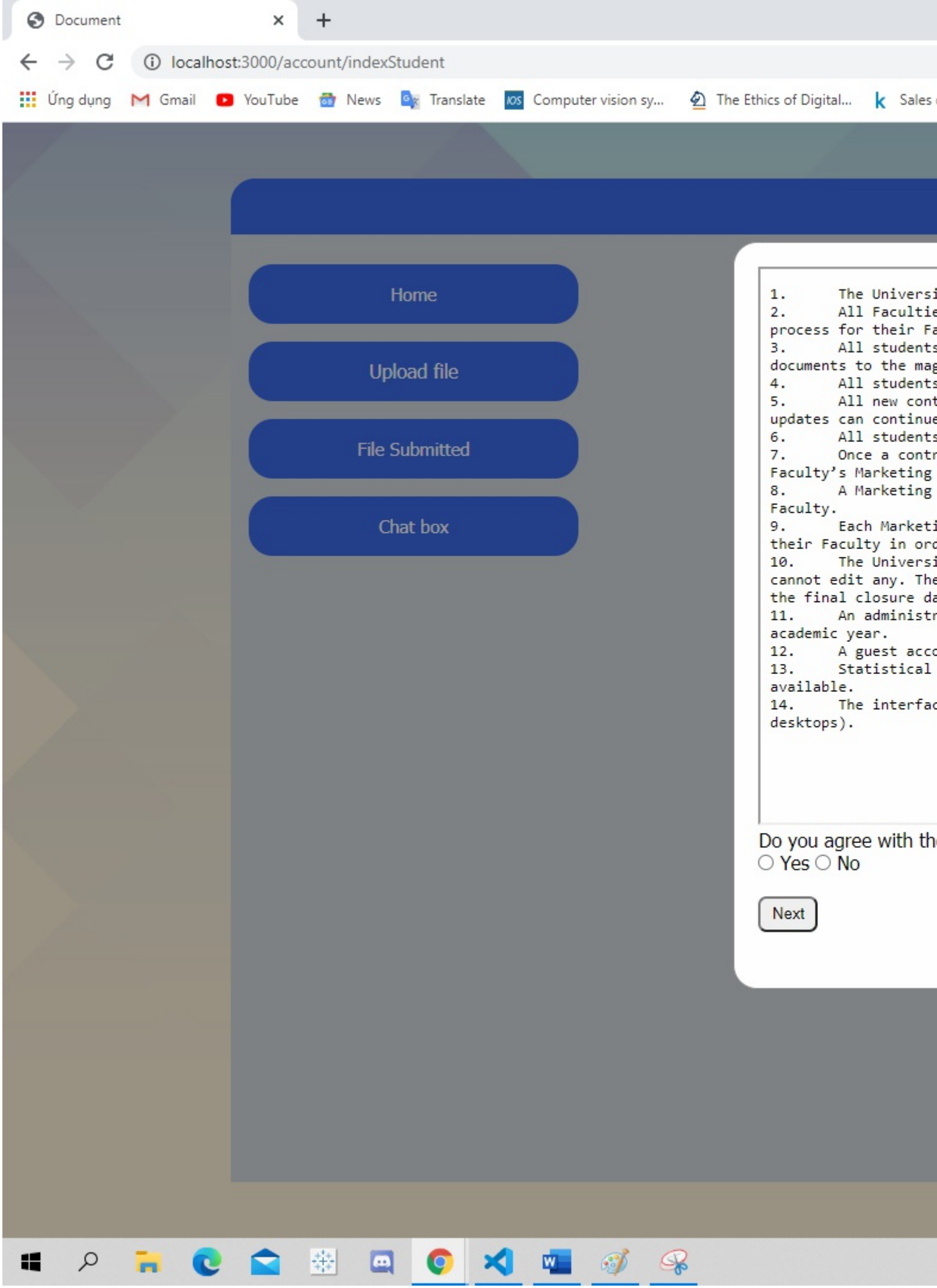

Figure 57:popup terms desktop

After clicking to Upload file, it will display the terms and students need to agree with the terms before they can access the updated files.

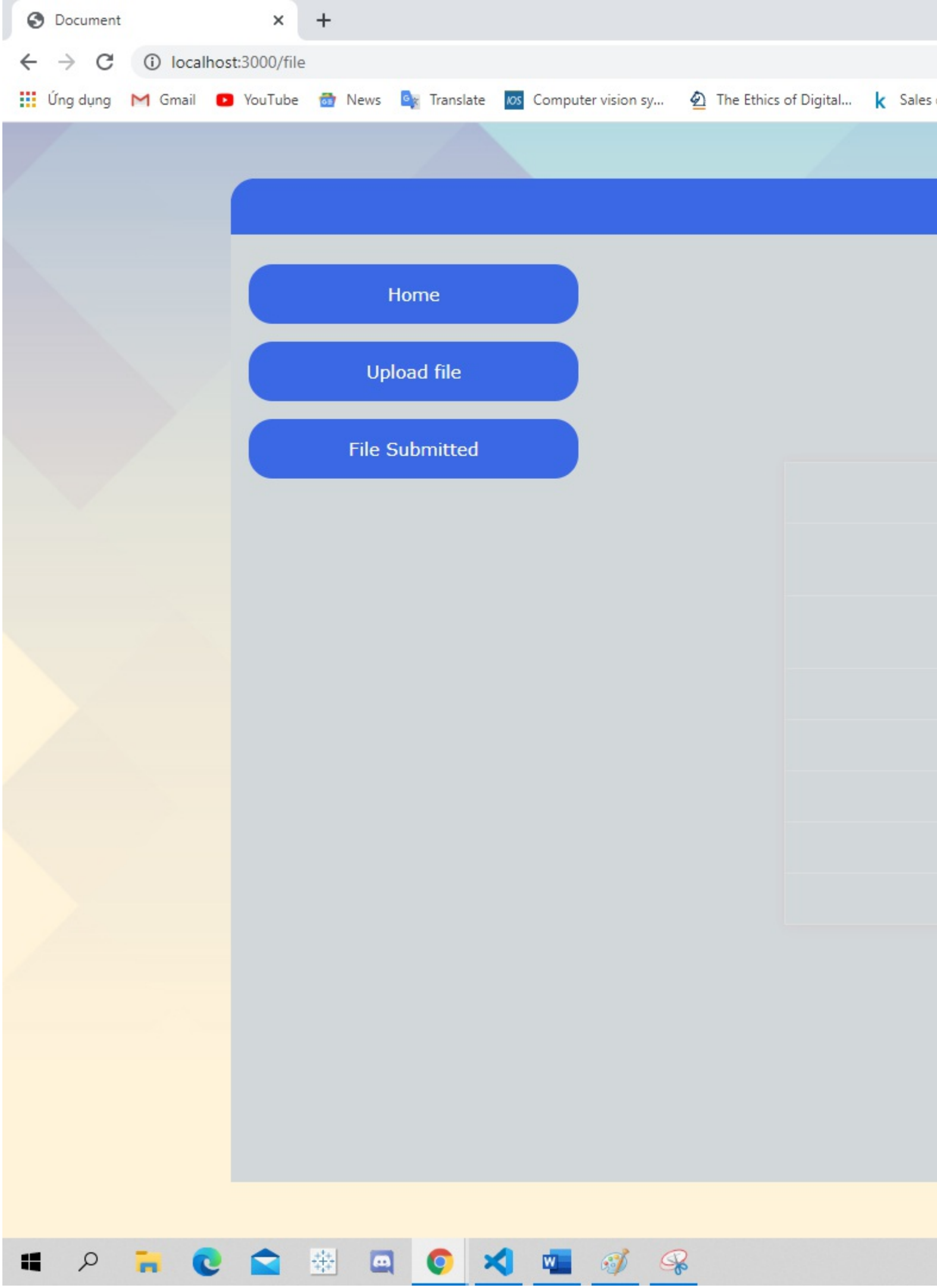

Figure 58:upload file

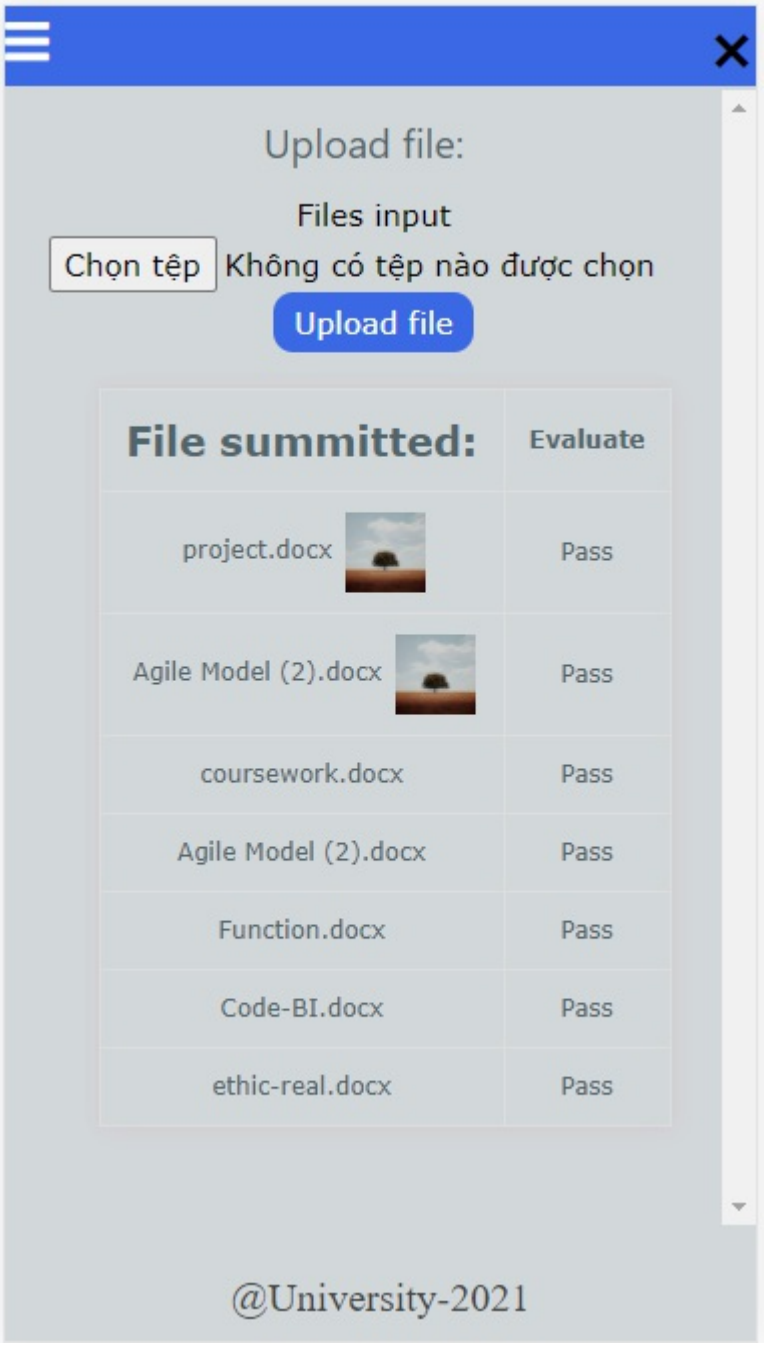

Figure 59:Upload file mobile

The input file section is for the user to import the files and images to upload the file. The table below shows the files that the user has uploaded. The content uses scroll, so the user can scroll down to see the content if the content is long.

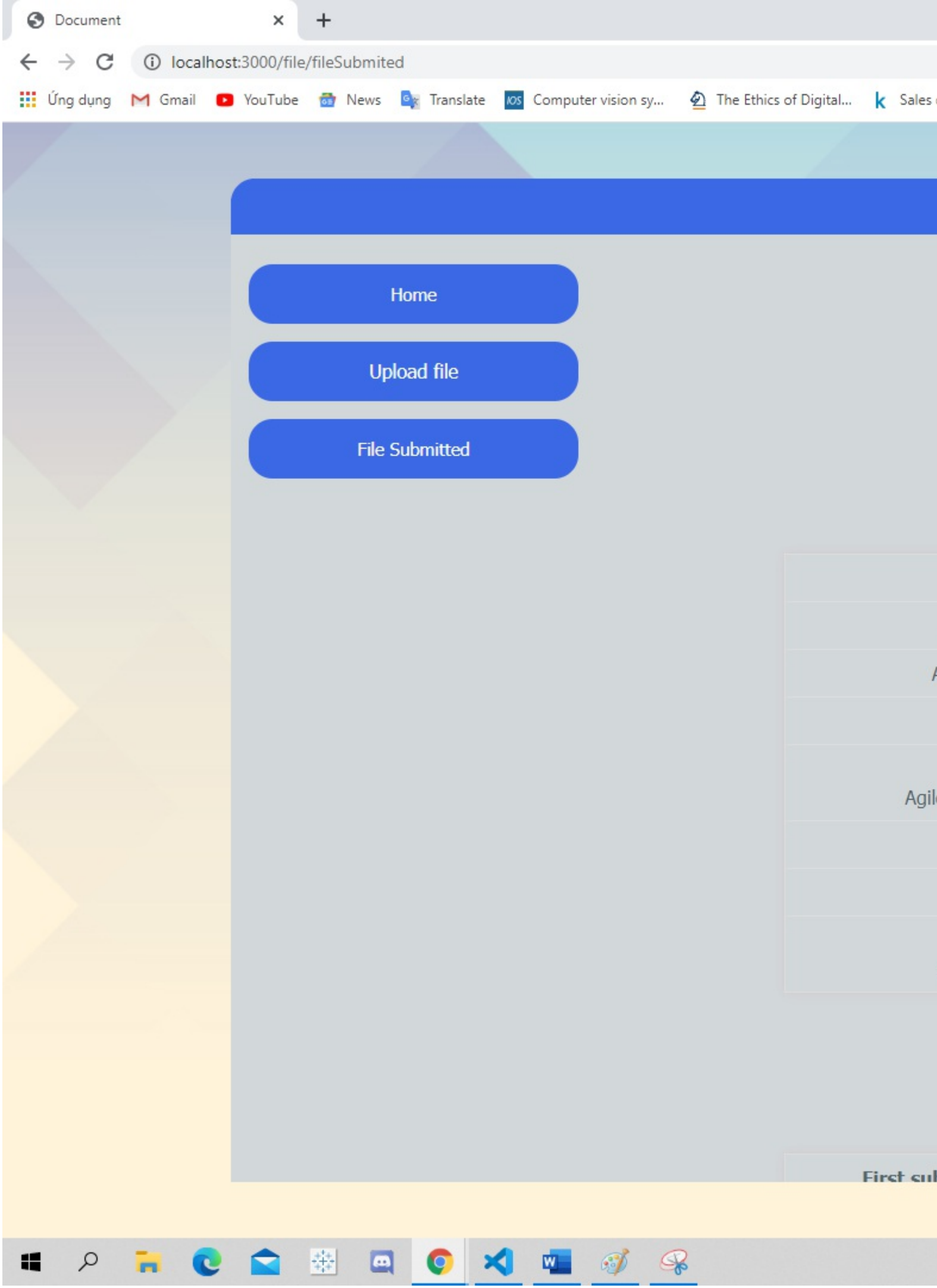

Figure 60:File submitted

Figure 61:File submitted mobile

It shows the deadline for the report, table 1 shows the reports that have passed and the file could not be uploaded. Table 2 shows the reports that have not been graded or failed. If there is a deadline, users can upload another file.

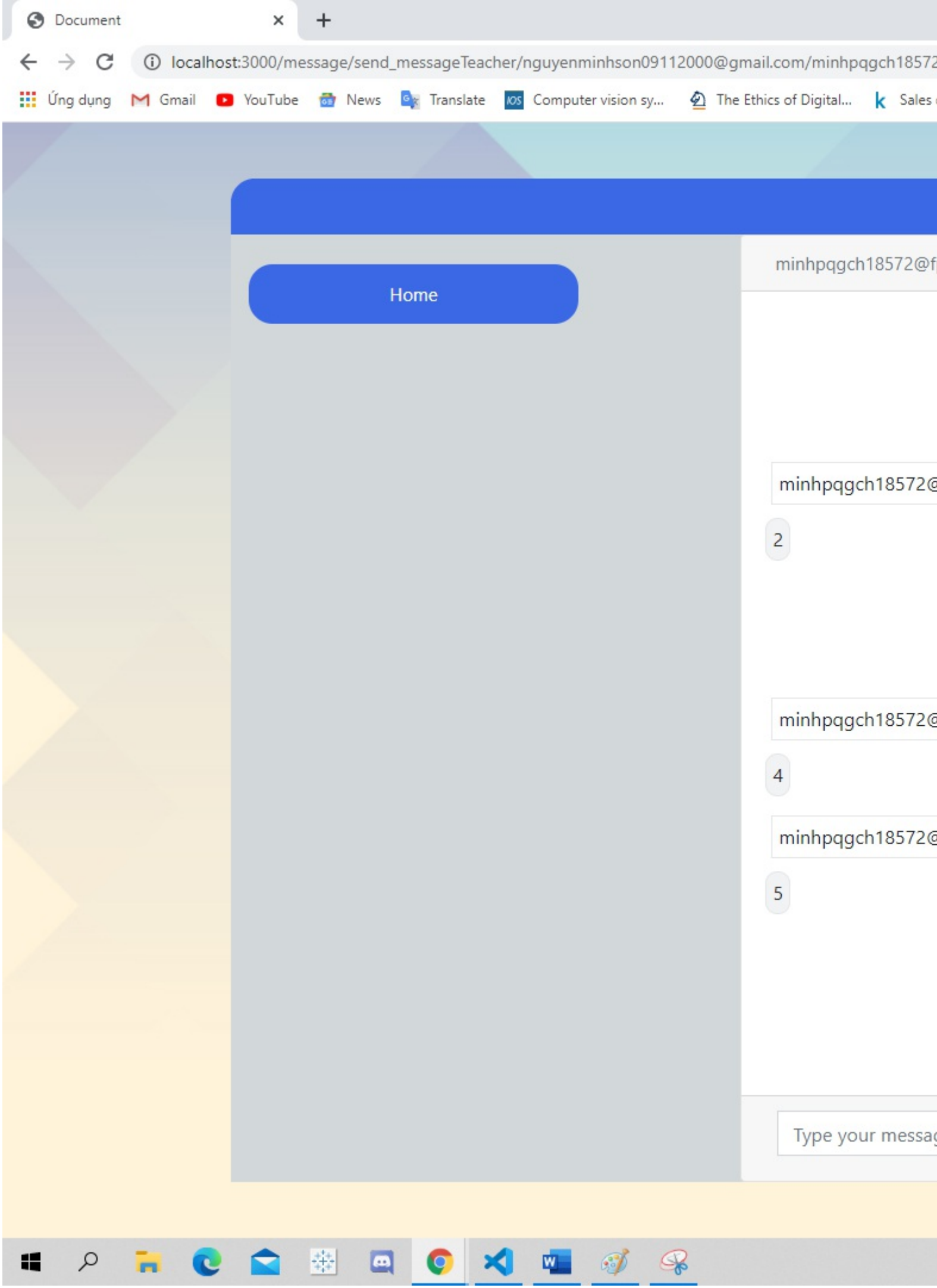

Figure 62:chat box of Student and marketing coordinator

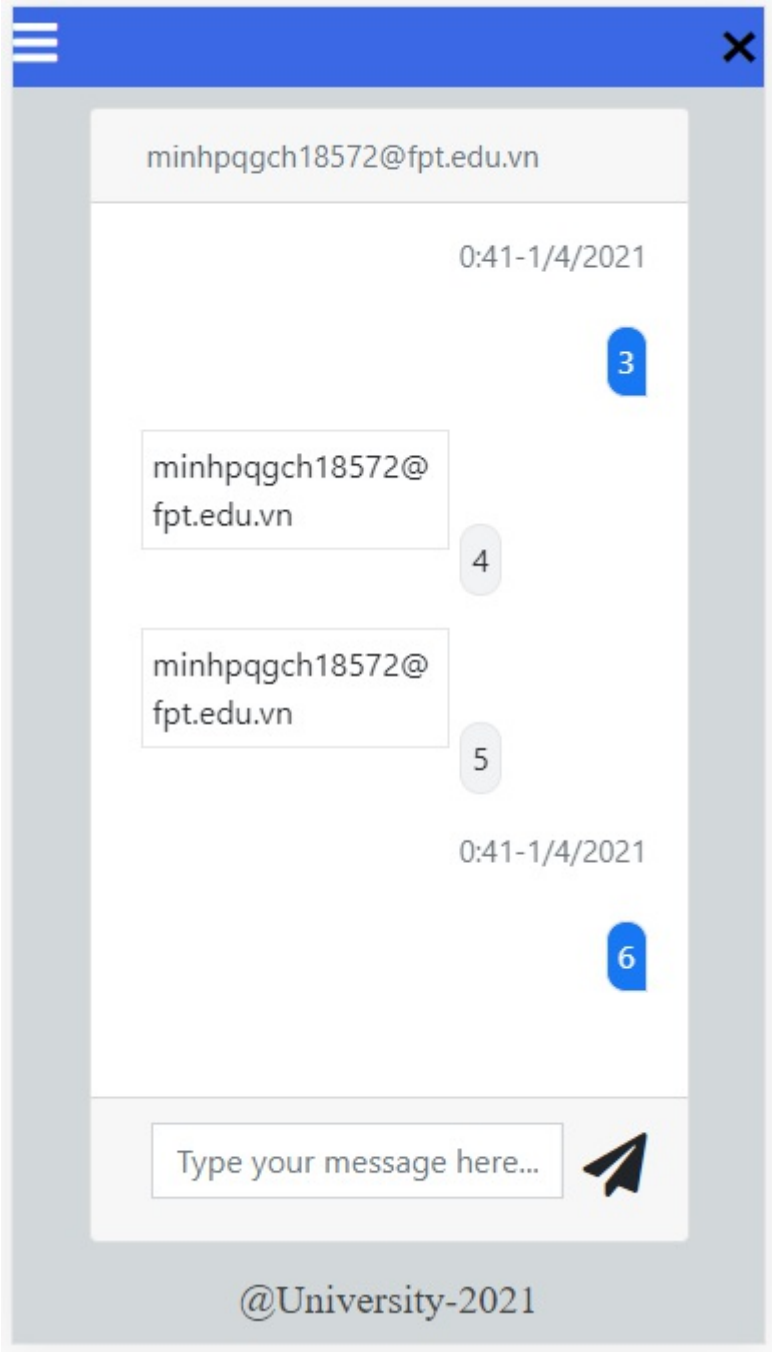

Figure 63:chat box of Student and marketing coordinator mobile

Students and coordinators can contact each other through the chat box.

Admin

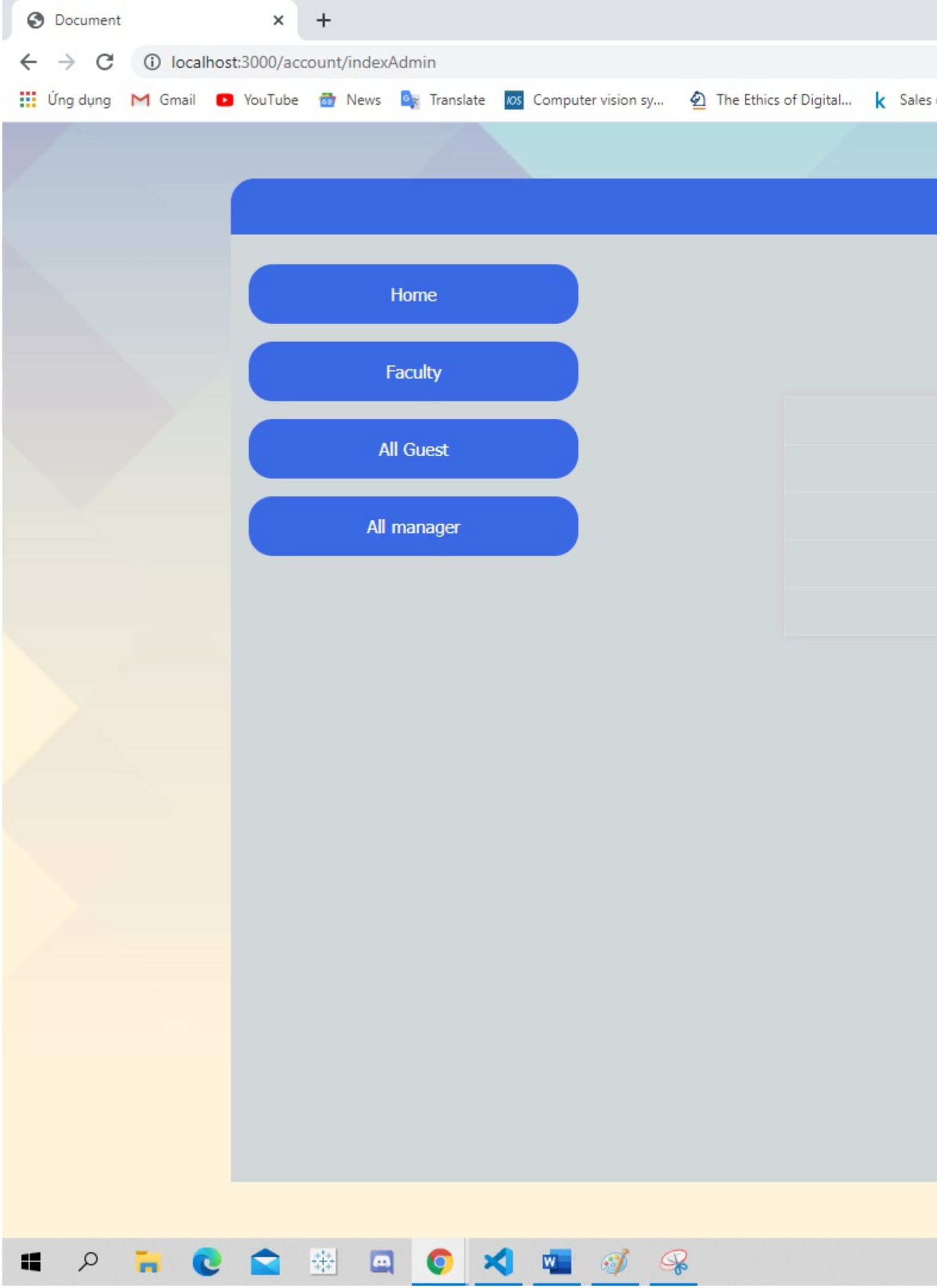

Figure 64:home page of admin

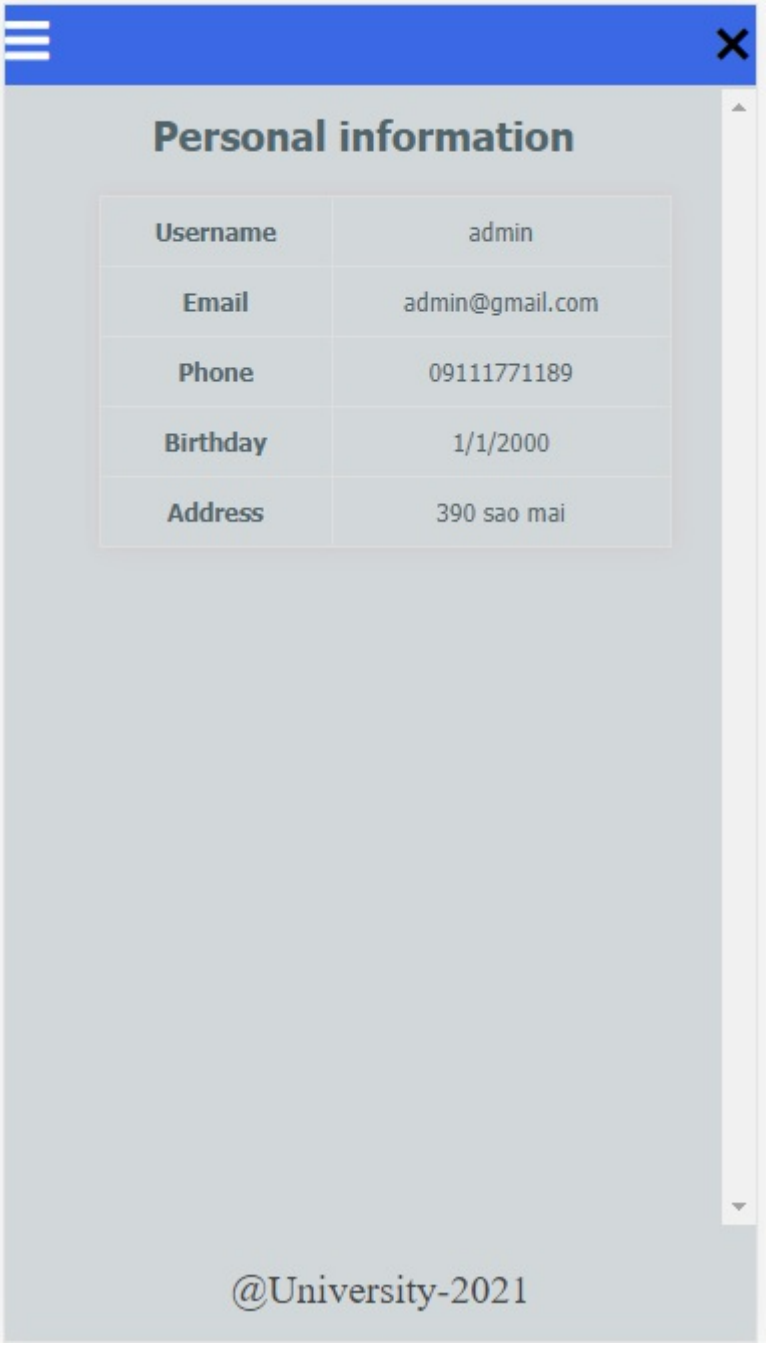

Figure 65:Home page of admin mobile

The admin interface has a menu with the following functions: home, faculty, all guest, and all marketing manager. Home content will display the user's personal information.

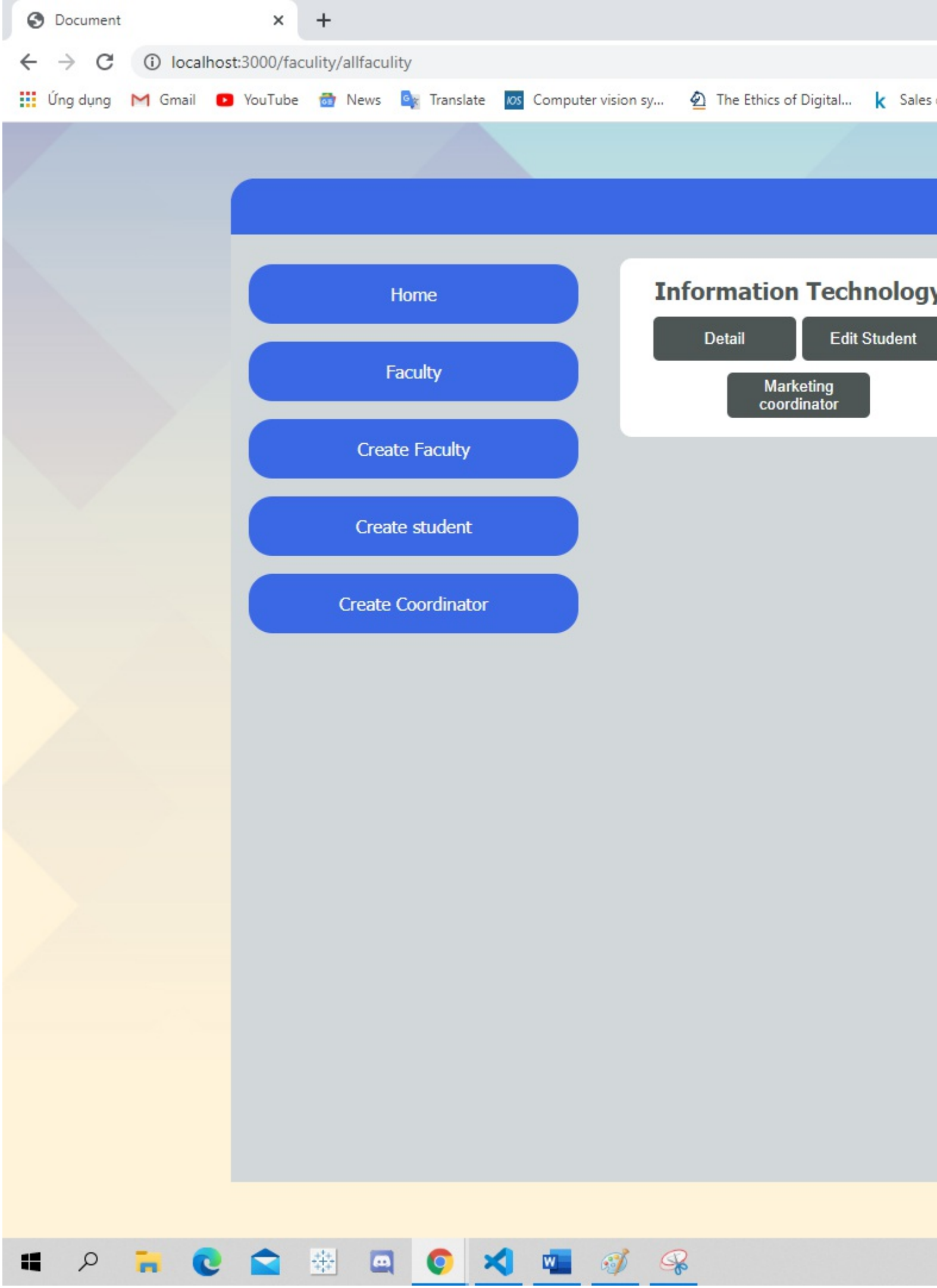

Figure 66:All faculty of admin

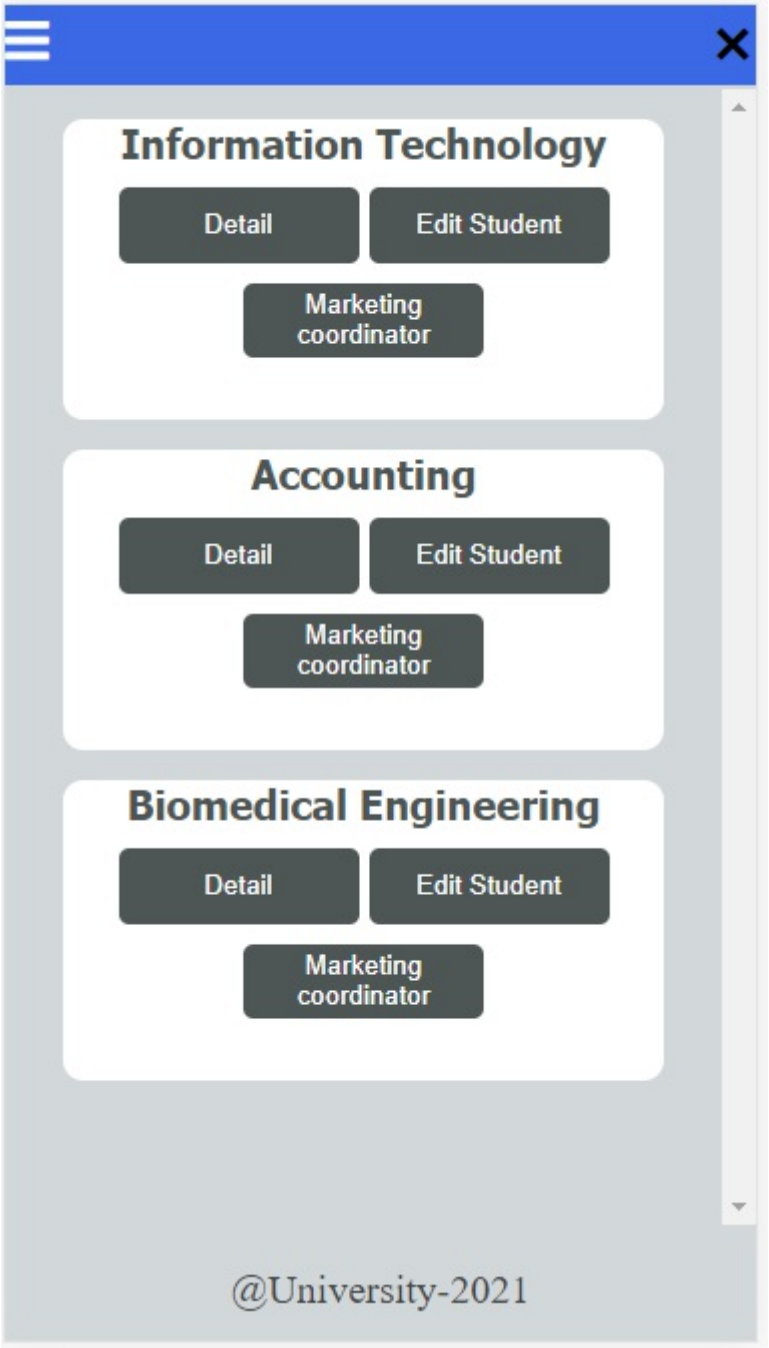

Figure 67:all faculty of admin mobile

In faculty, it will have the functions: create faculty, create student and create marketing coordinator. Content will display the Faculty name and can be edited by "Detail", can view the students in the department, can view the marketing coordinator of that department.

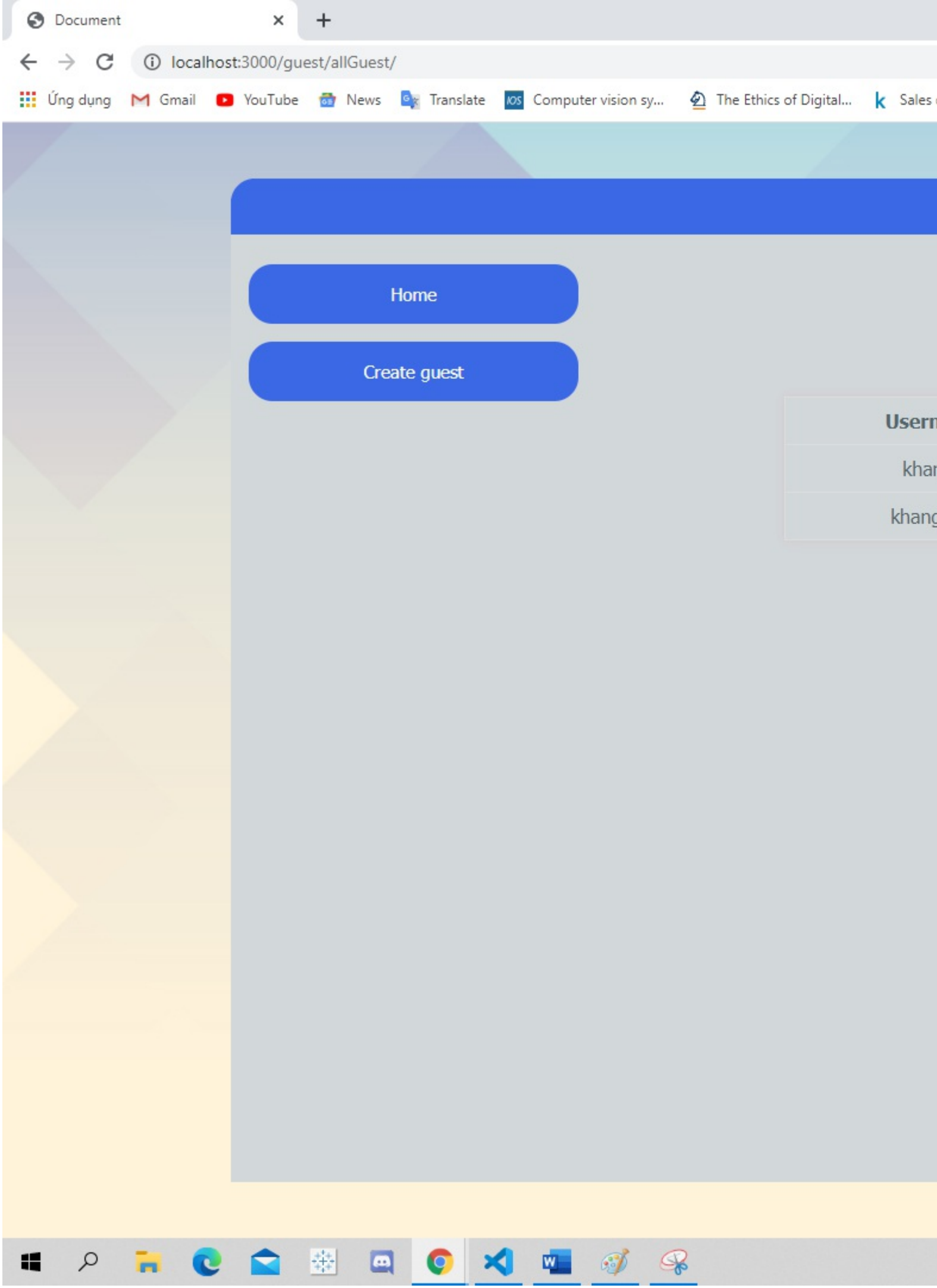

Figure 68:all guest of admin

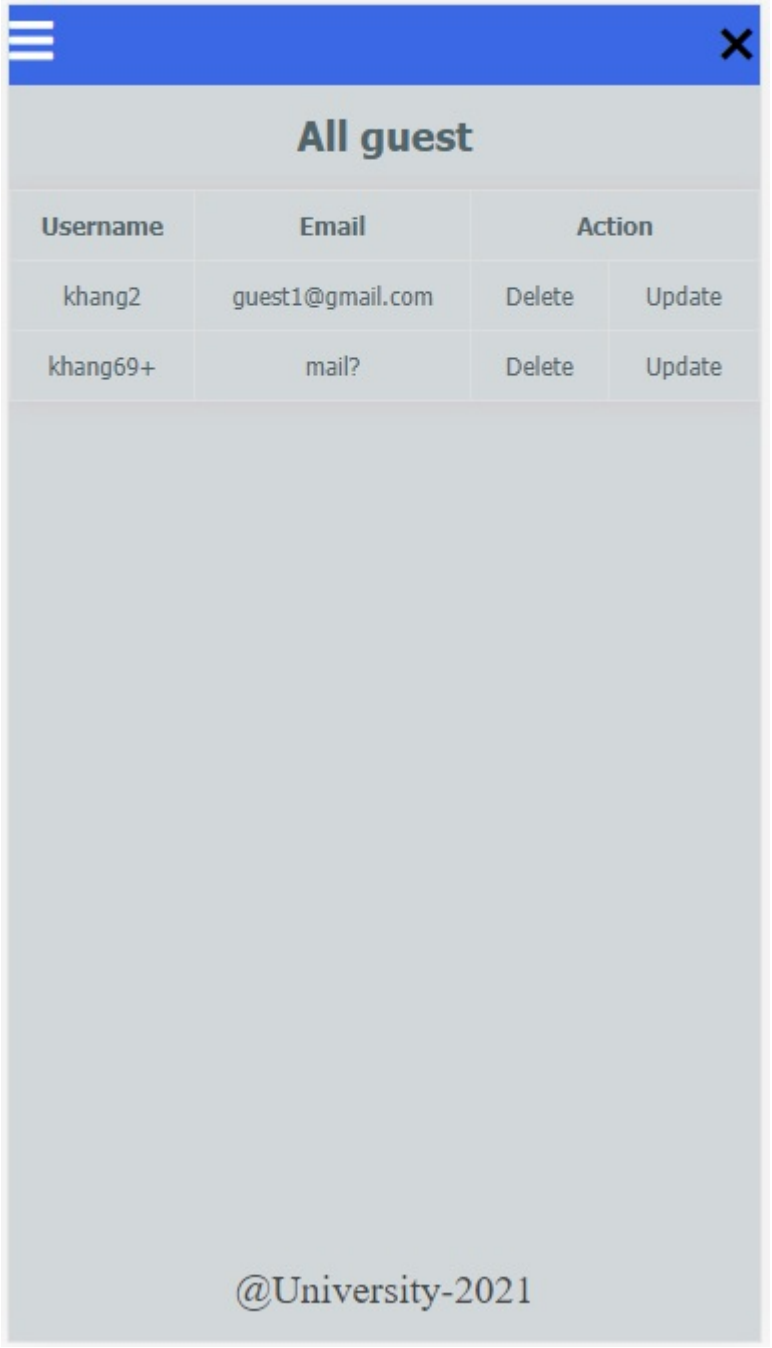

Figure 69:all guest of admin mobile

Show all guest and marketing managers. Admin can create guest and marketing coordinator, manage them.

Marketing manager:

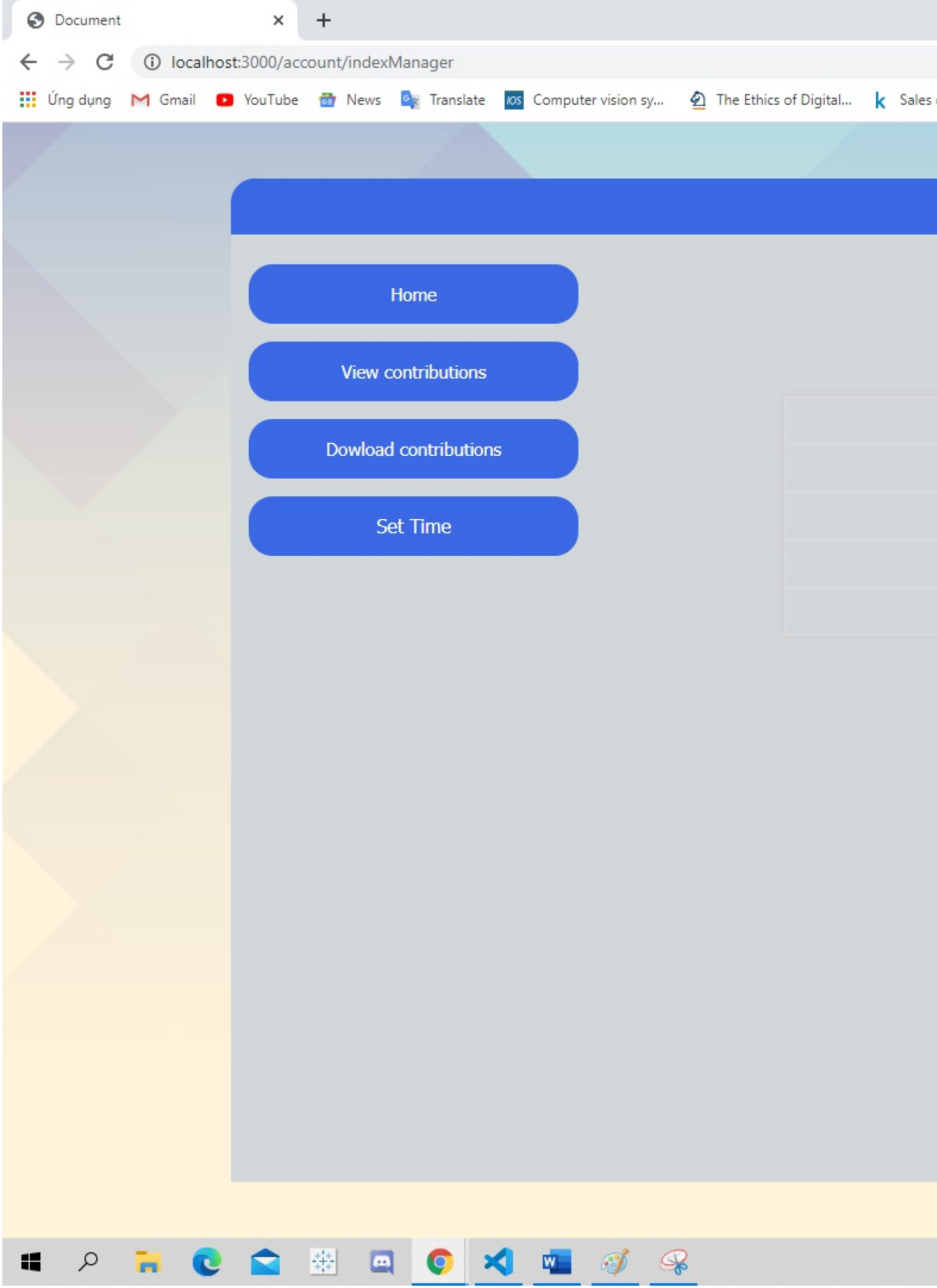

Figure 70:home page of marketing manager

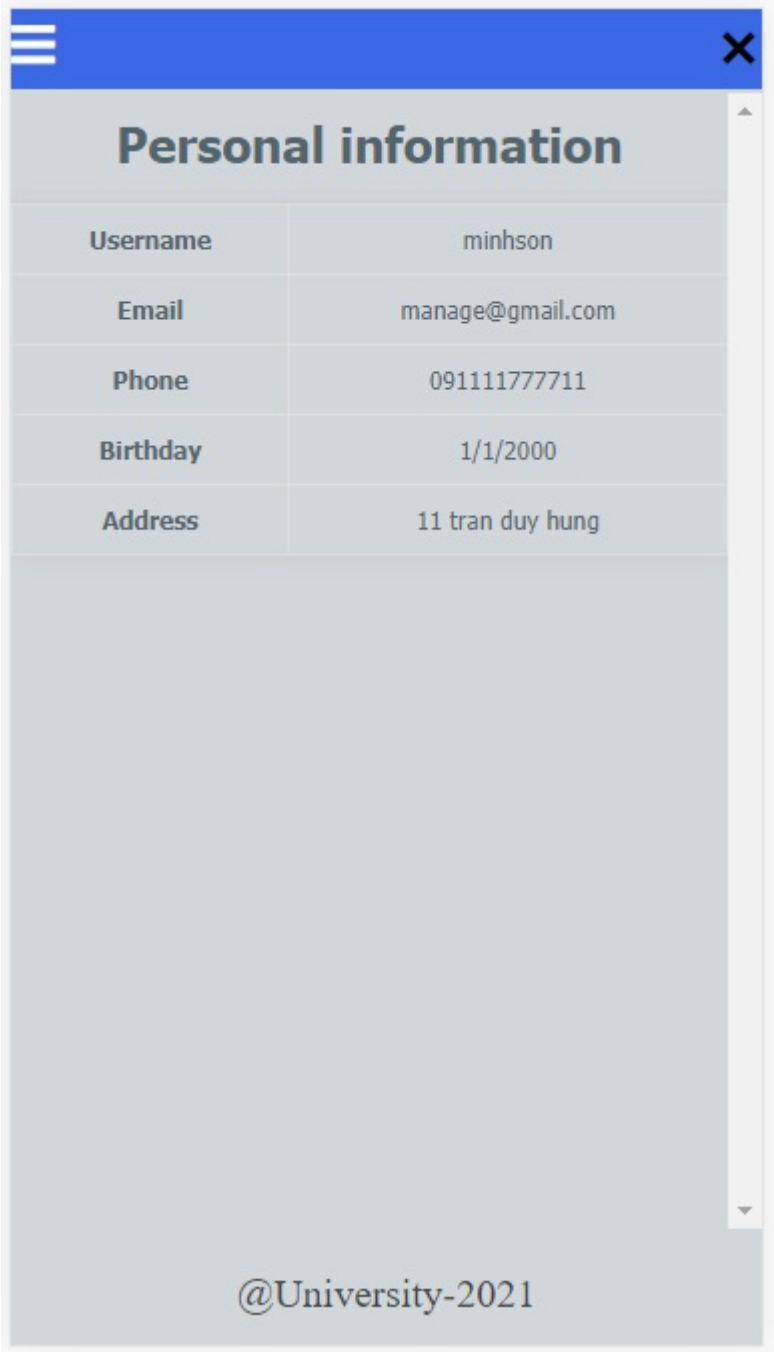

Figure 71:home page of marketing manager mobile

The marketing manager has functions such as home, view contributions, download contributions, and set times.

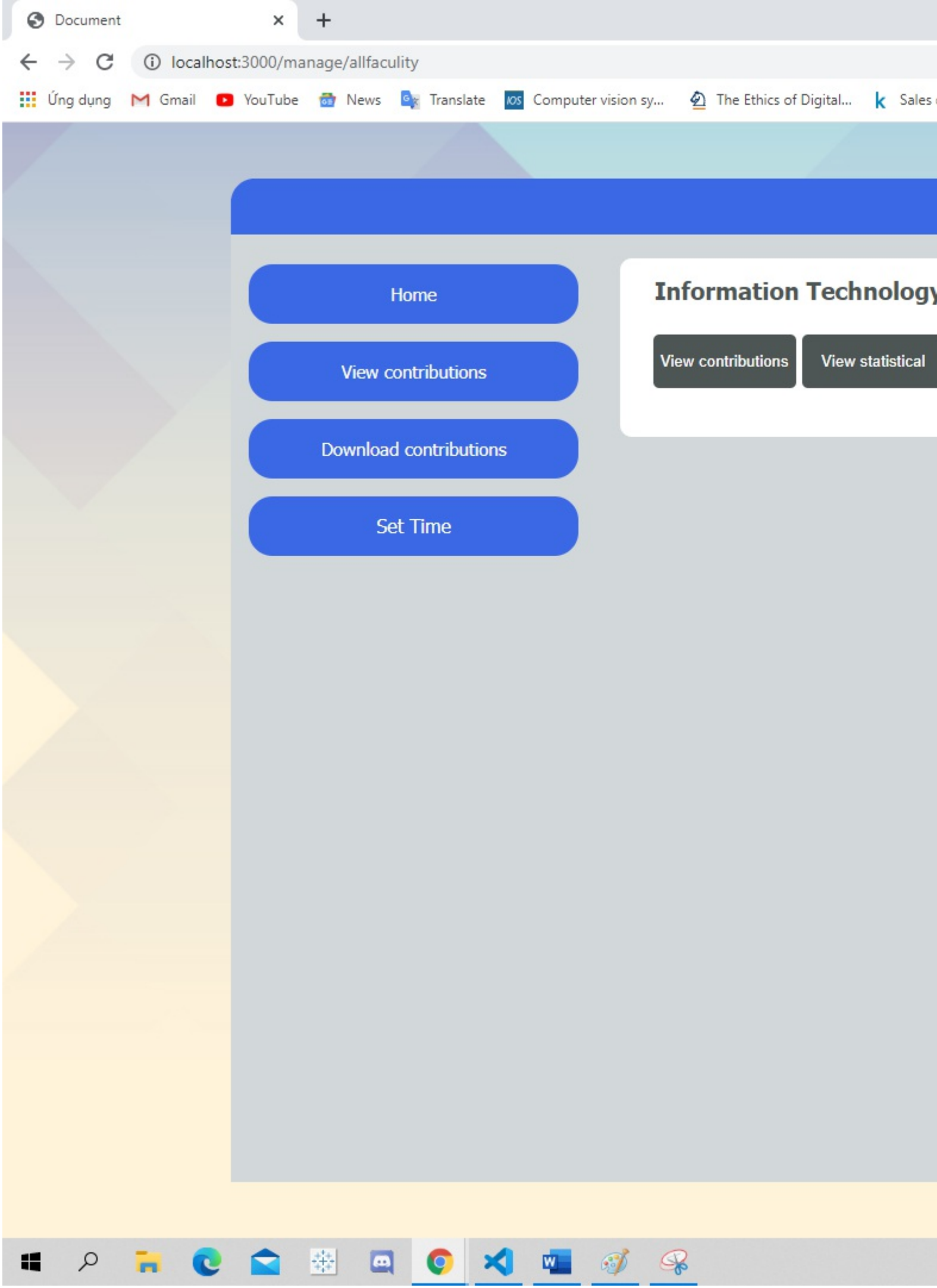

Figure 72:all faculty of marketing manager

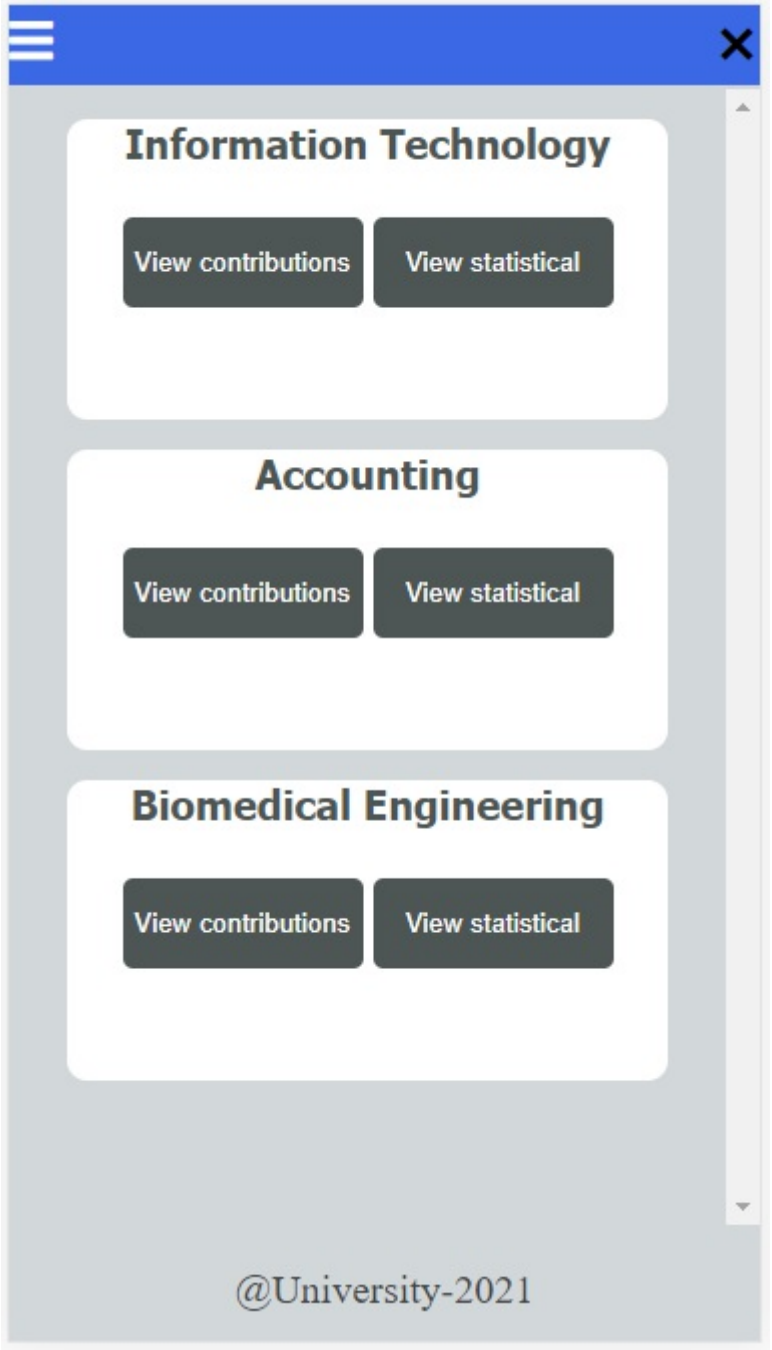

Figure 73:all faculty of marketing manager mobile

When click to views contribution marketing manager can view faculty reports and view statistical faculty. From there we can know the submission rate, the student's pass rate.

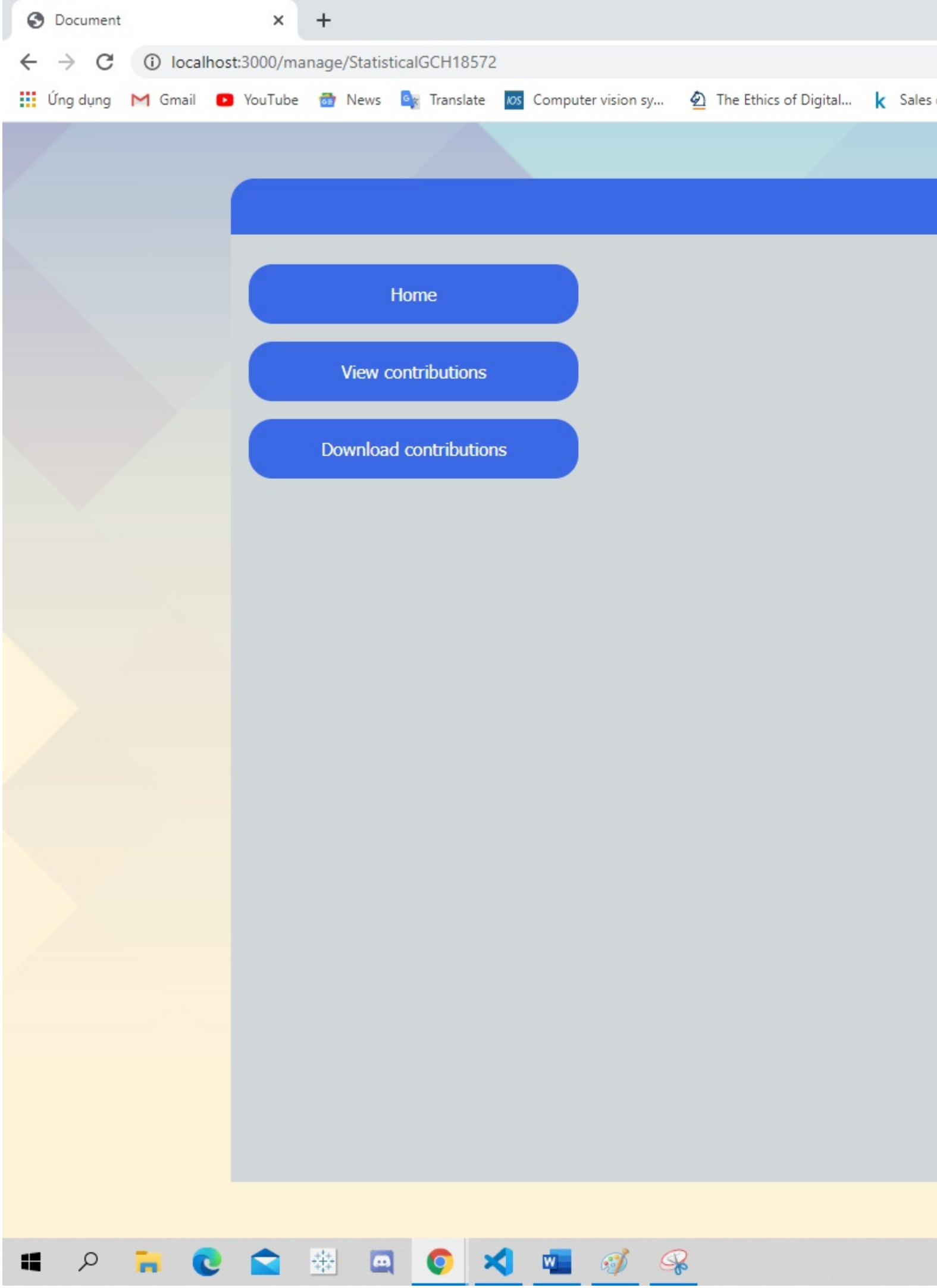

Figure 74:bar chart

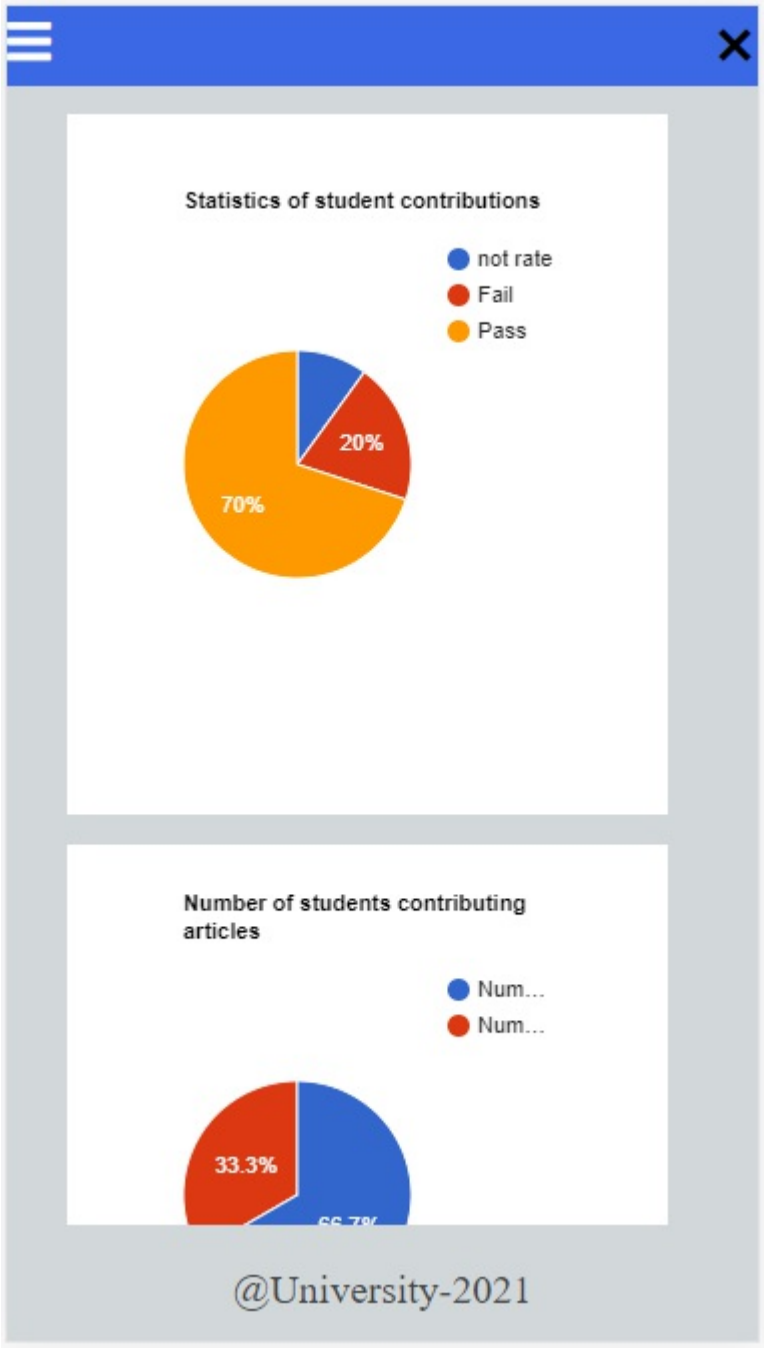

Figure 75:bar chart mobile

When you click to download the contribution marketing manager, you can download faculty reports.

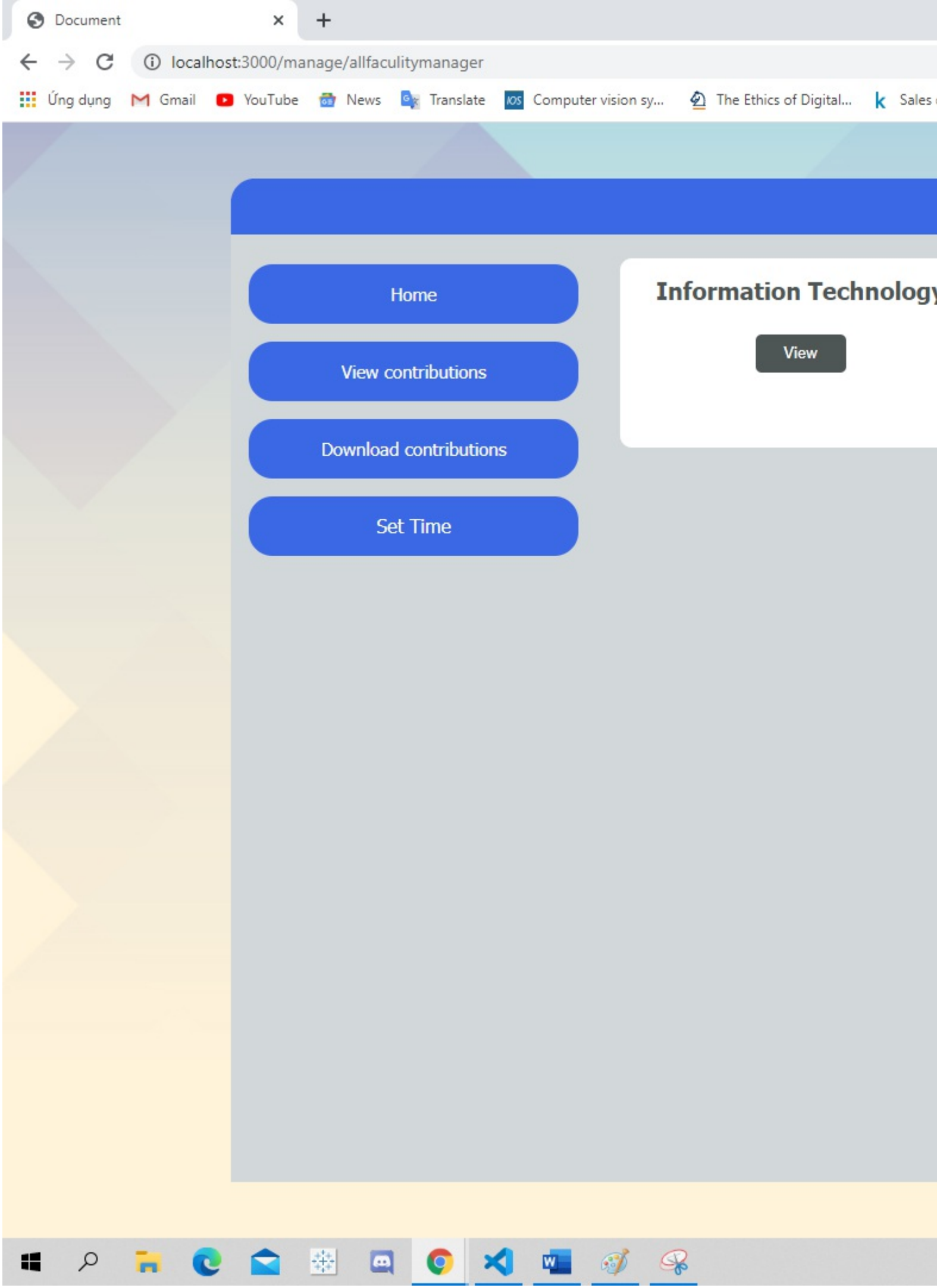

Figure 76:download contributions of marketing manager

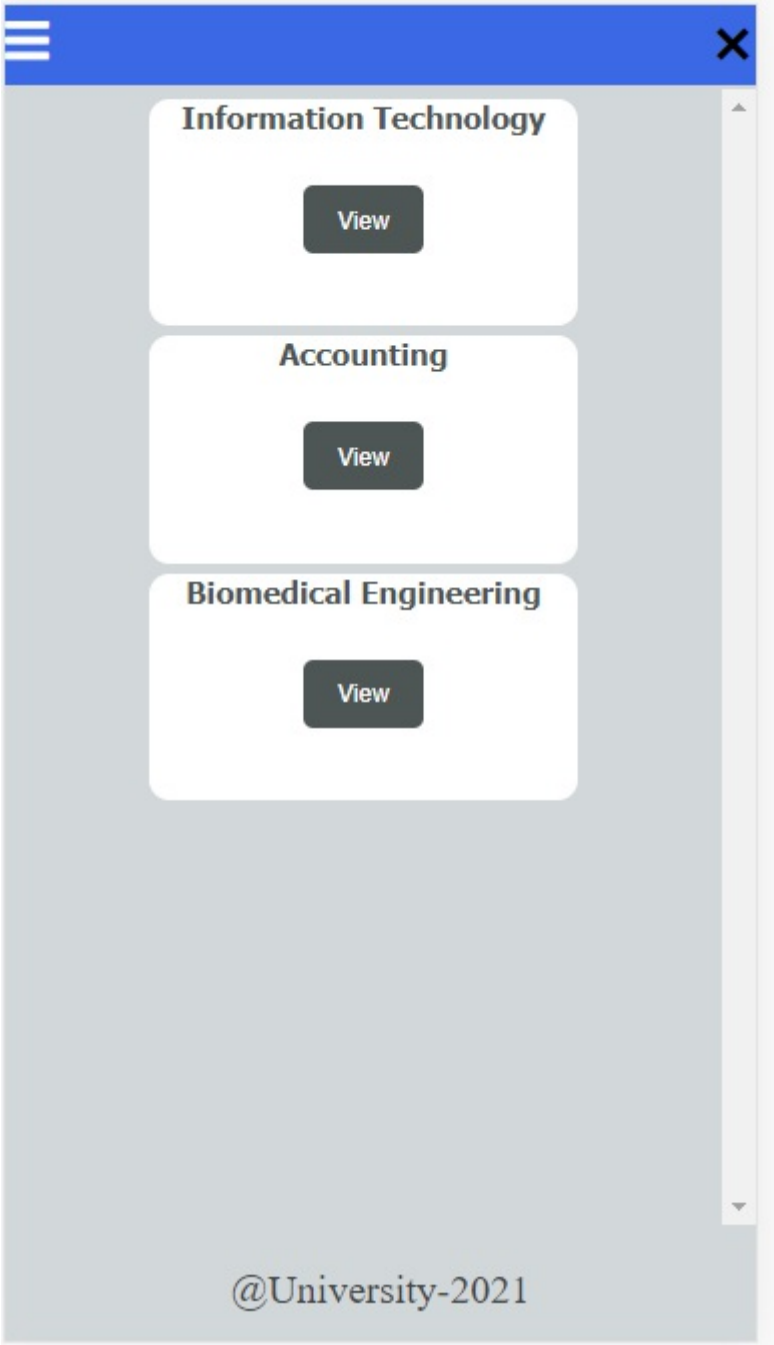

Figure 77:download contributions of marketing manager mobile

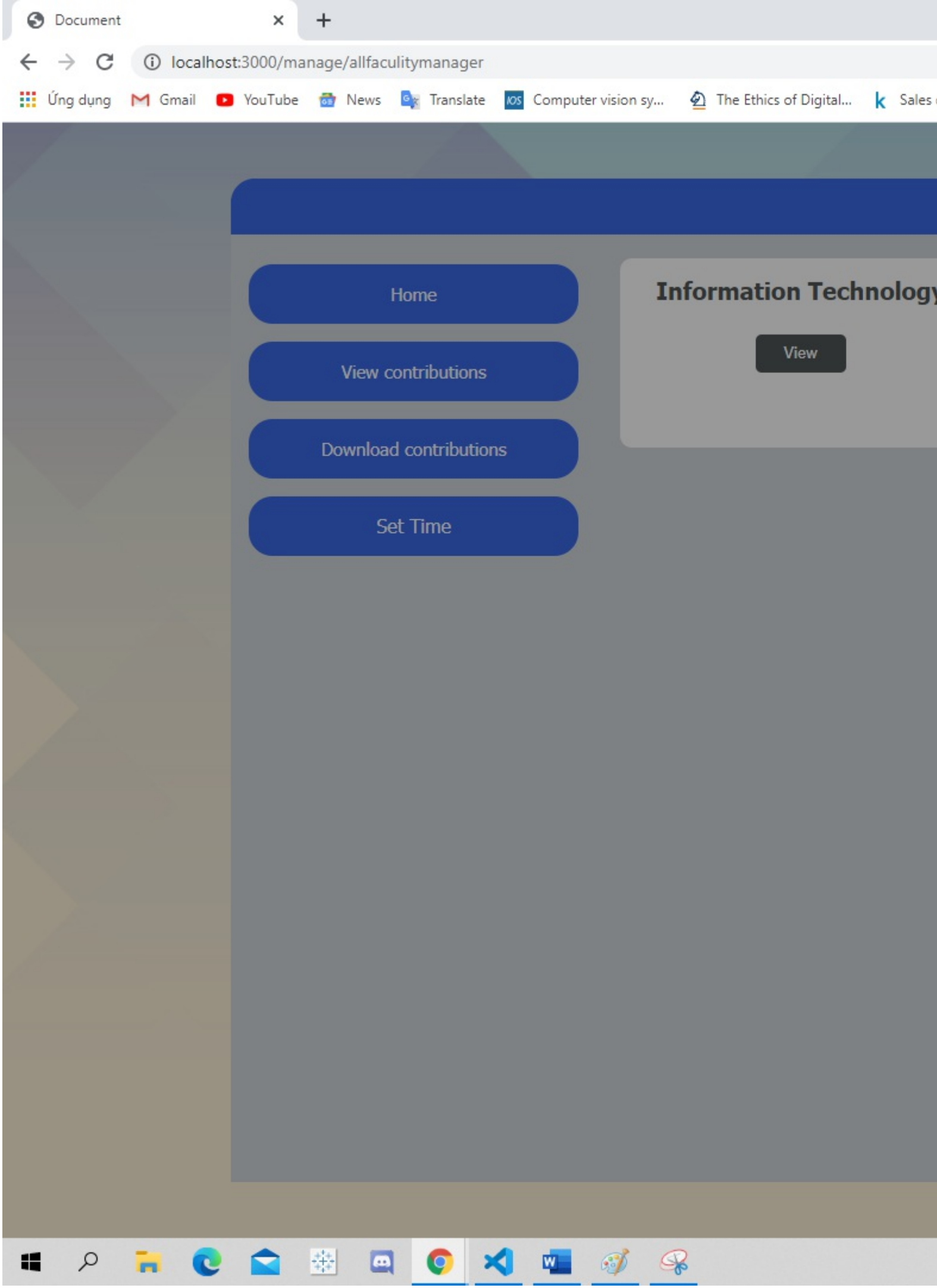

Figure 78:set time of marketing manager

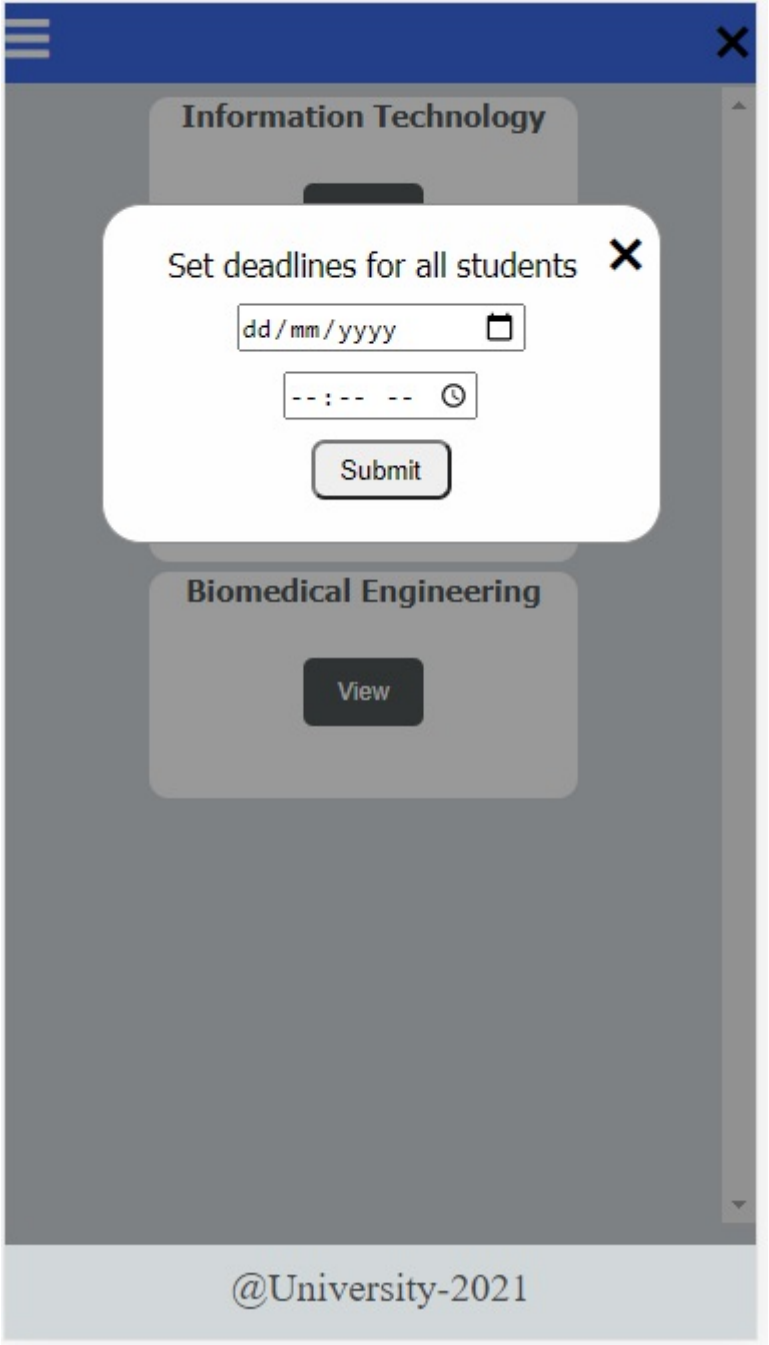

Figure 79: set time of marketing manager mobile

A marketing manager can set a time for the report of students.

Guest:

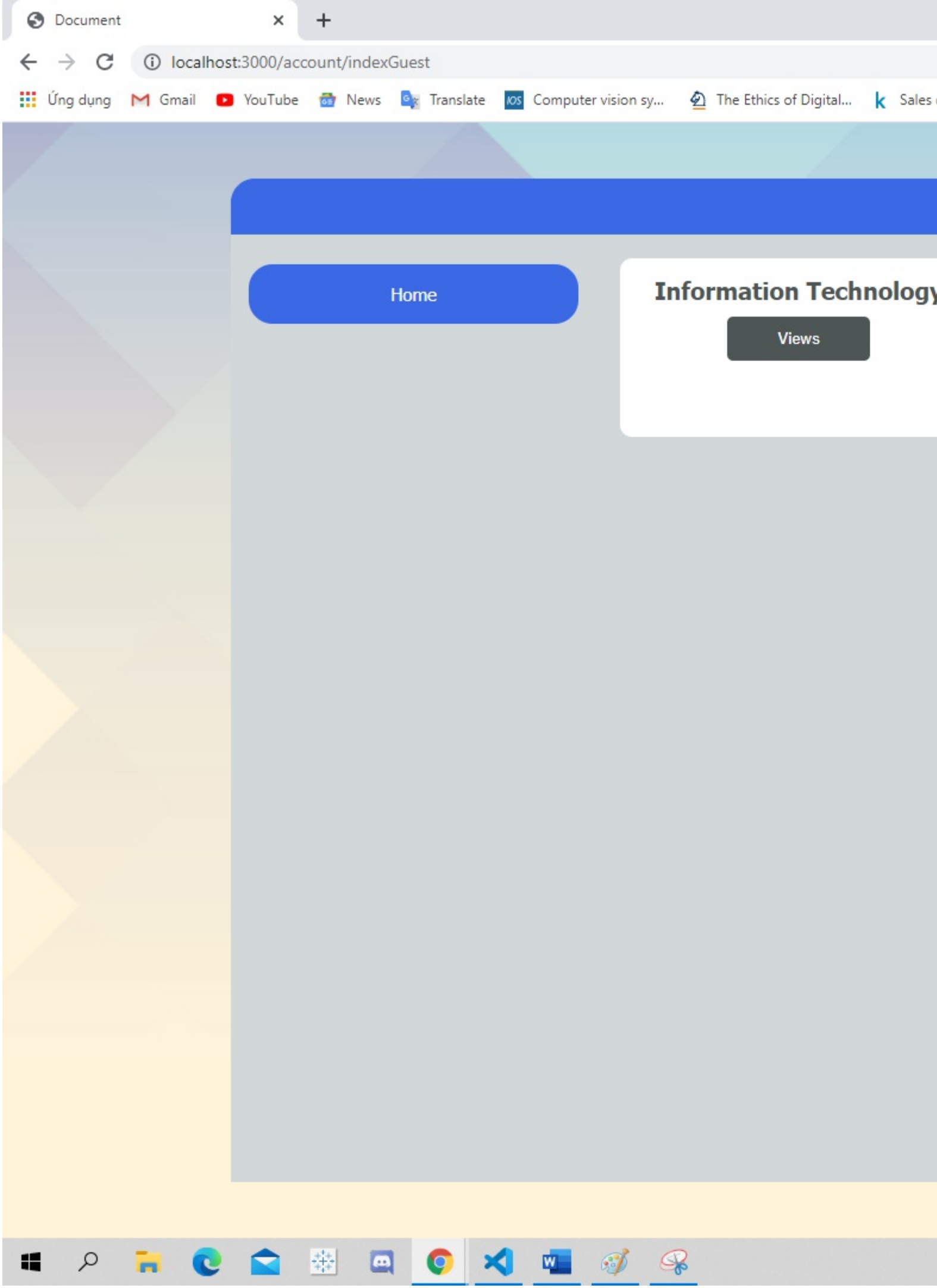

Figure 80:view faculty of guest

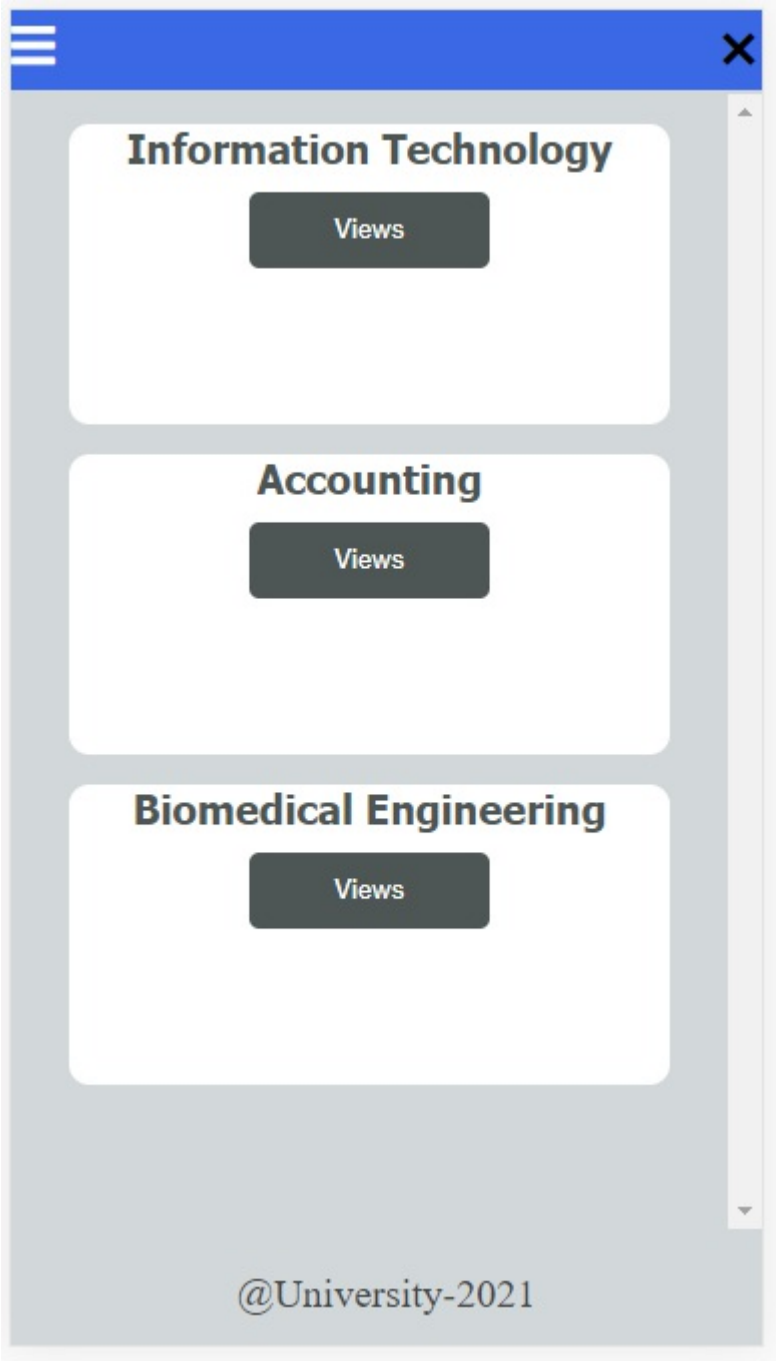

Figure 81:View faculty of guest mobile

It will show faculty that has been moderated and the guest can choose to be able to view faculty's posts.

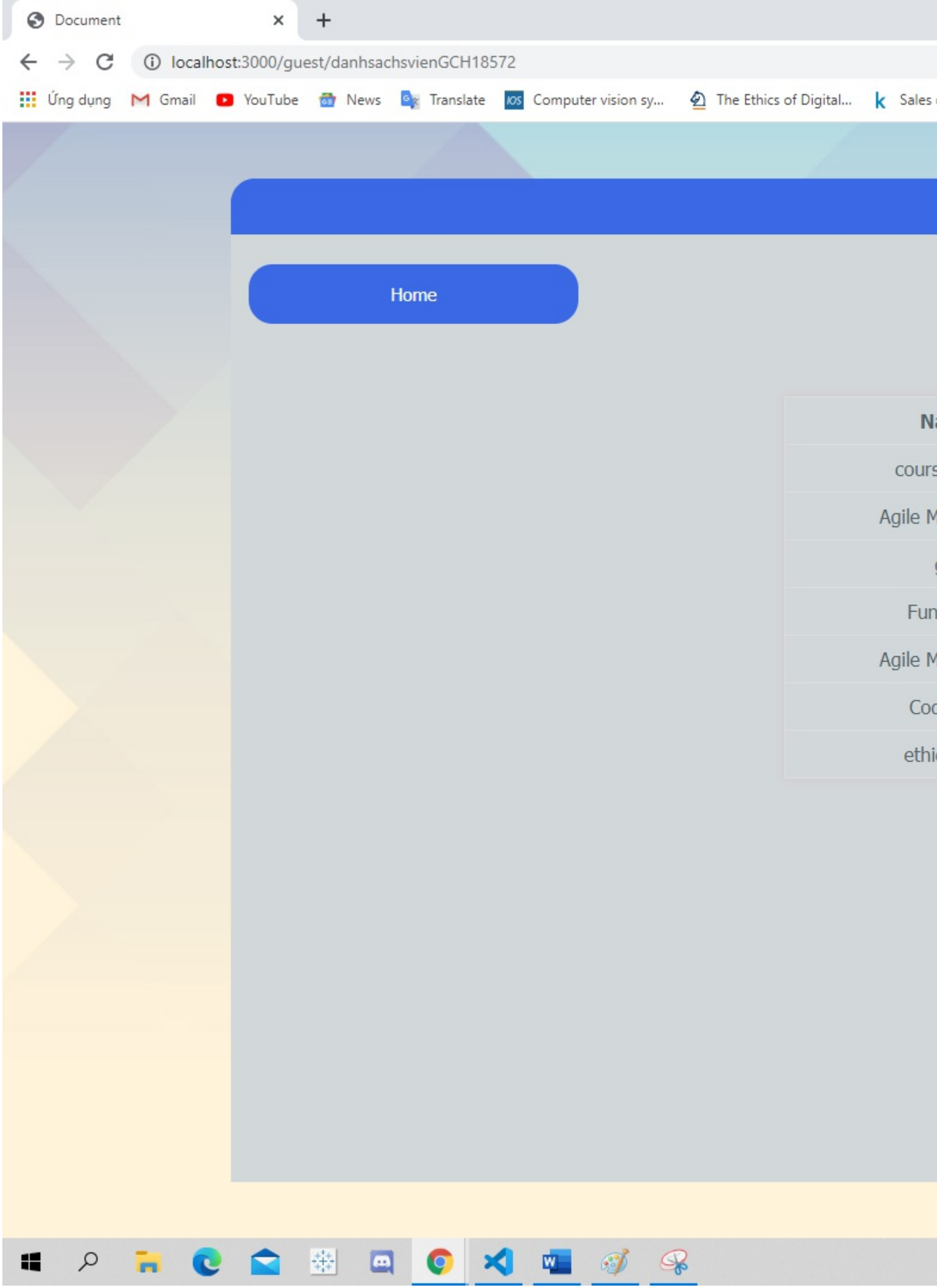

Figure 82:Student contributions of guest

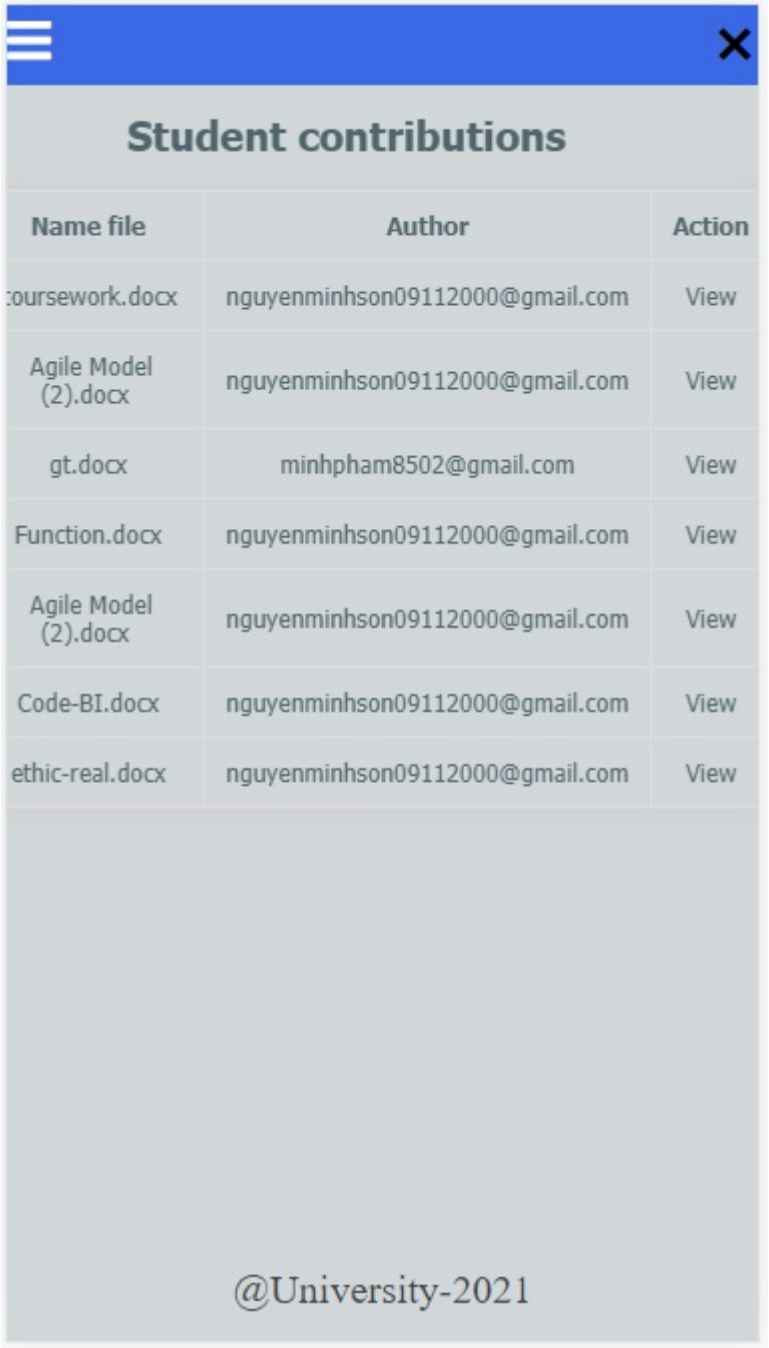

Figure 83:Student contributions of guest mobile

After selecting faculty, all posts of faculty will be displayed. When you select the view, it will display a report and image.

Document

localhost:3000/guest/danhgiabaibao60632e80873a9422804f3b35

Ứng dụng | Gmail | YouTube | News | Translate | Computer vision sy... | The Ethics of Digital... | Sales

You can download the file hereAgile Model (2).docx

Agile Model (2).pdf 1 / 3 100%

Activities: Compare Waterfall vs Agile, their pros and cons (20mins)

Agile SDLC model is a combination of iterative (loop) and incremental (bigger overt
adaptability and customer satisfaction by rapid delivery of working software product.
incremental builds. These builds are provided in iterations. Each iteration typically last
weeks). Every iteration involves cross functional teams working simultaneously on va

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product (prototype) is displayed to the custome

What is Agile?

Agile model believes that every project needs to be handled differently and the existi
the project requirements. In Agile, the tasks are divided to time boxes (small time fra

Iterative approach is taken and working software build is delivered after each iteratio
features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model –

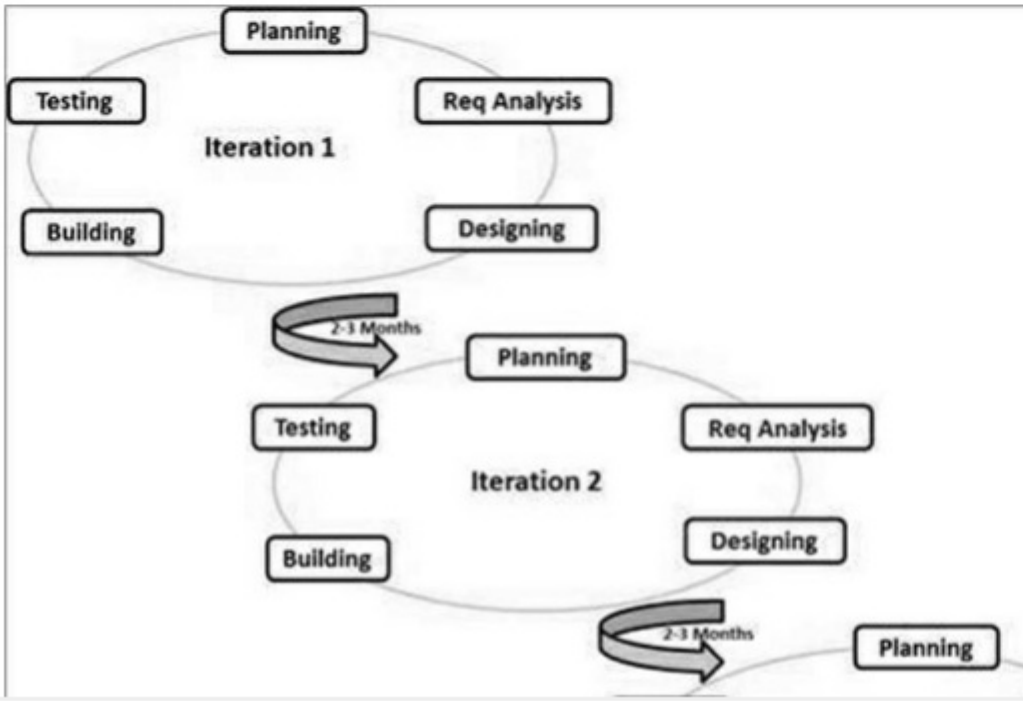

Figure 84:View report

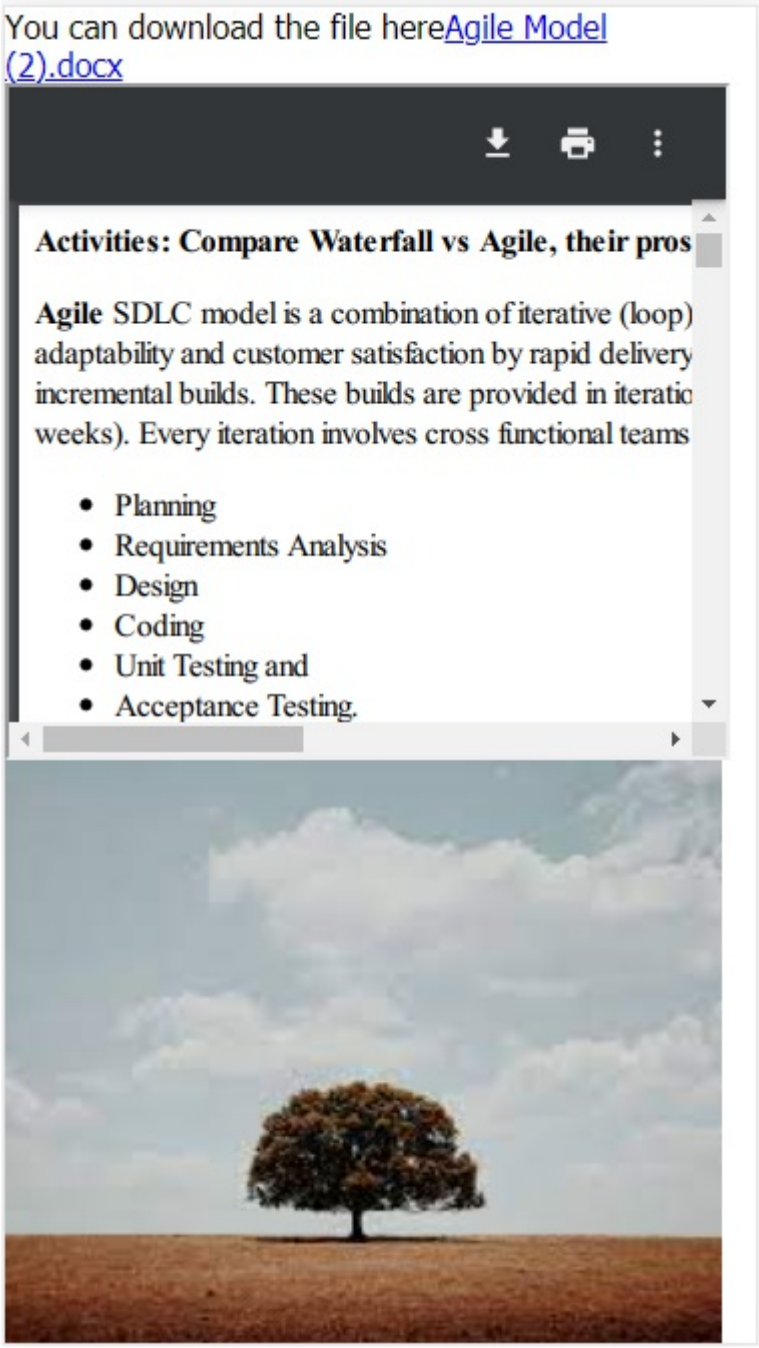

Figure 85:View report mobile

Marketing coordinator

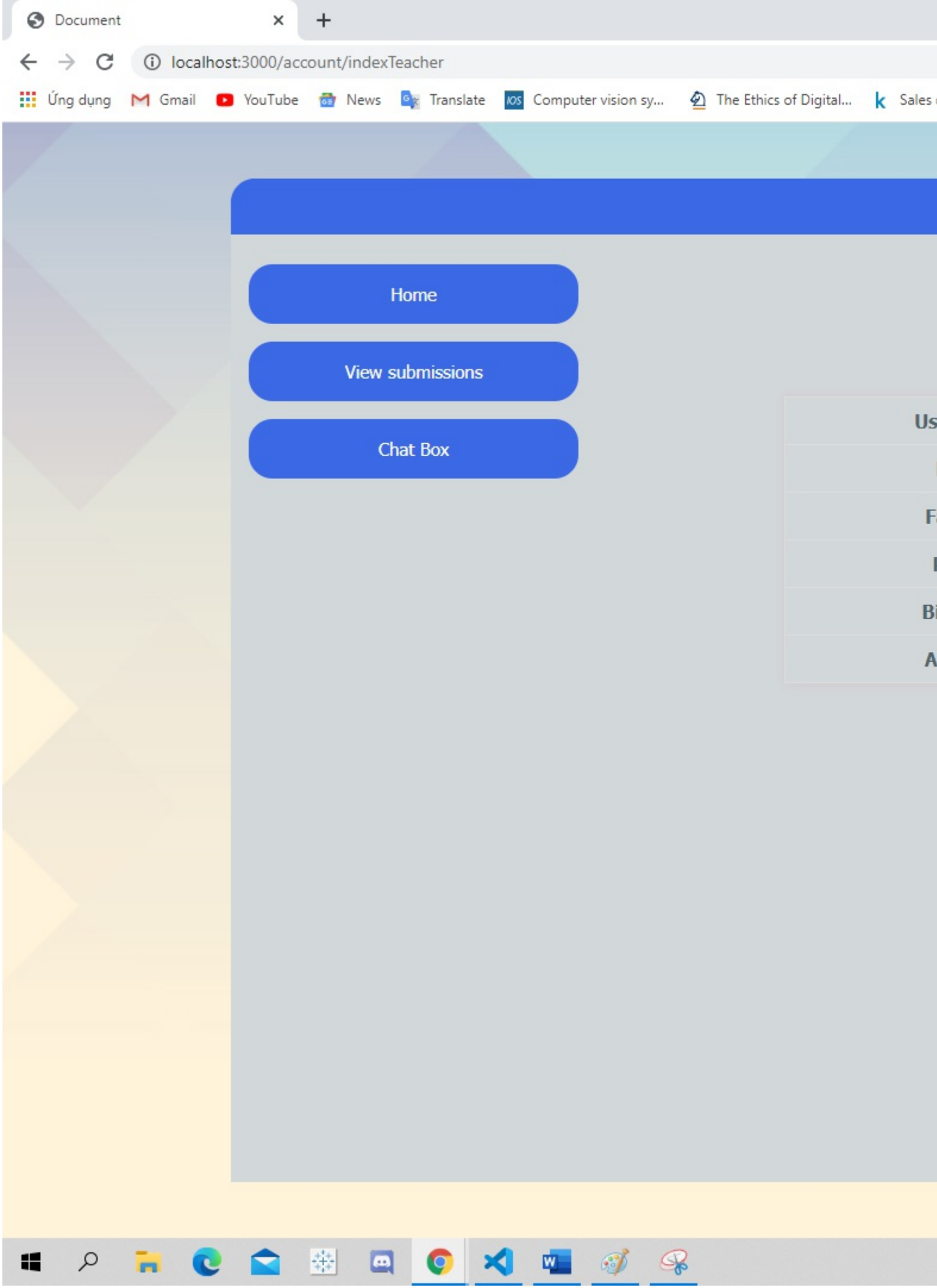

Figure 86:Home page of marketing coordinator

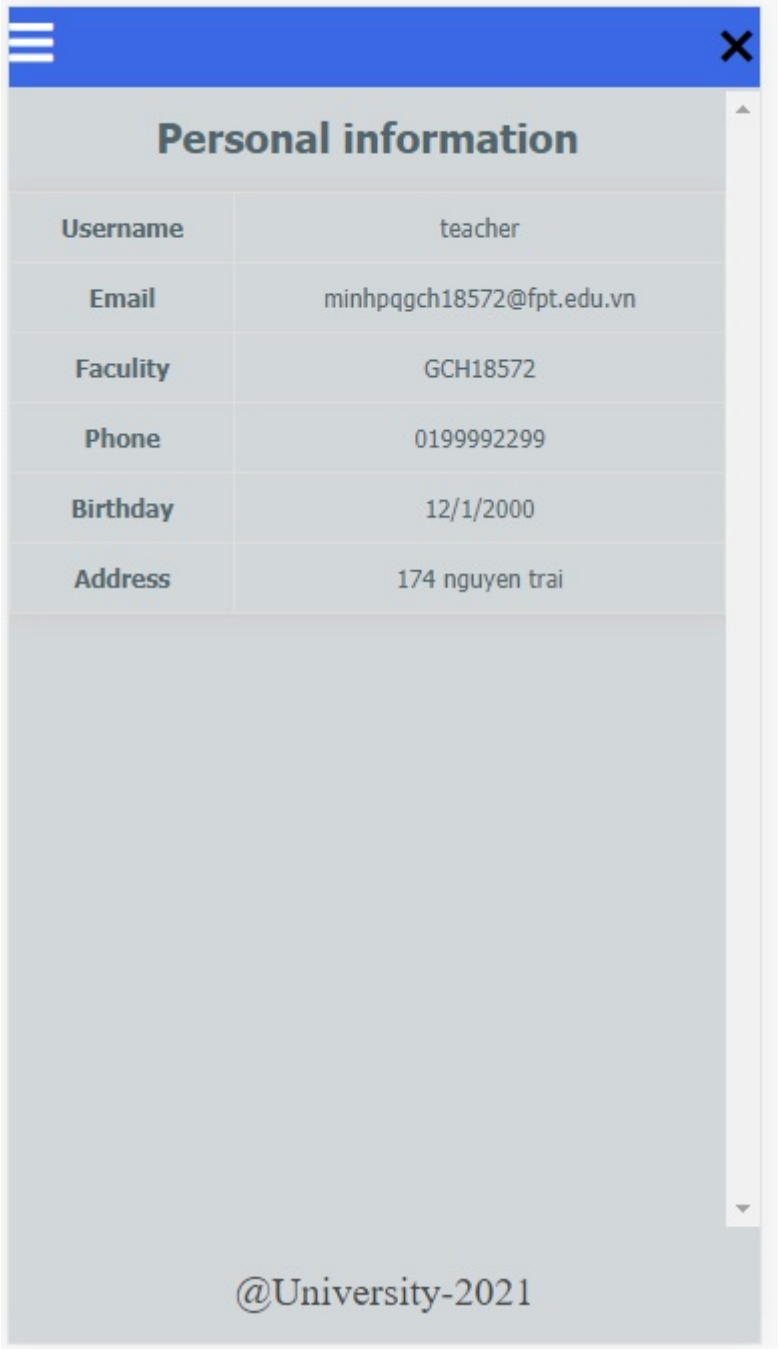

Figure 87:home page of marketing coordinator mobile

The marketing coordinator has the following functions: Home, view submissions, chat box.

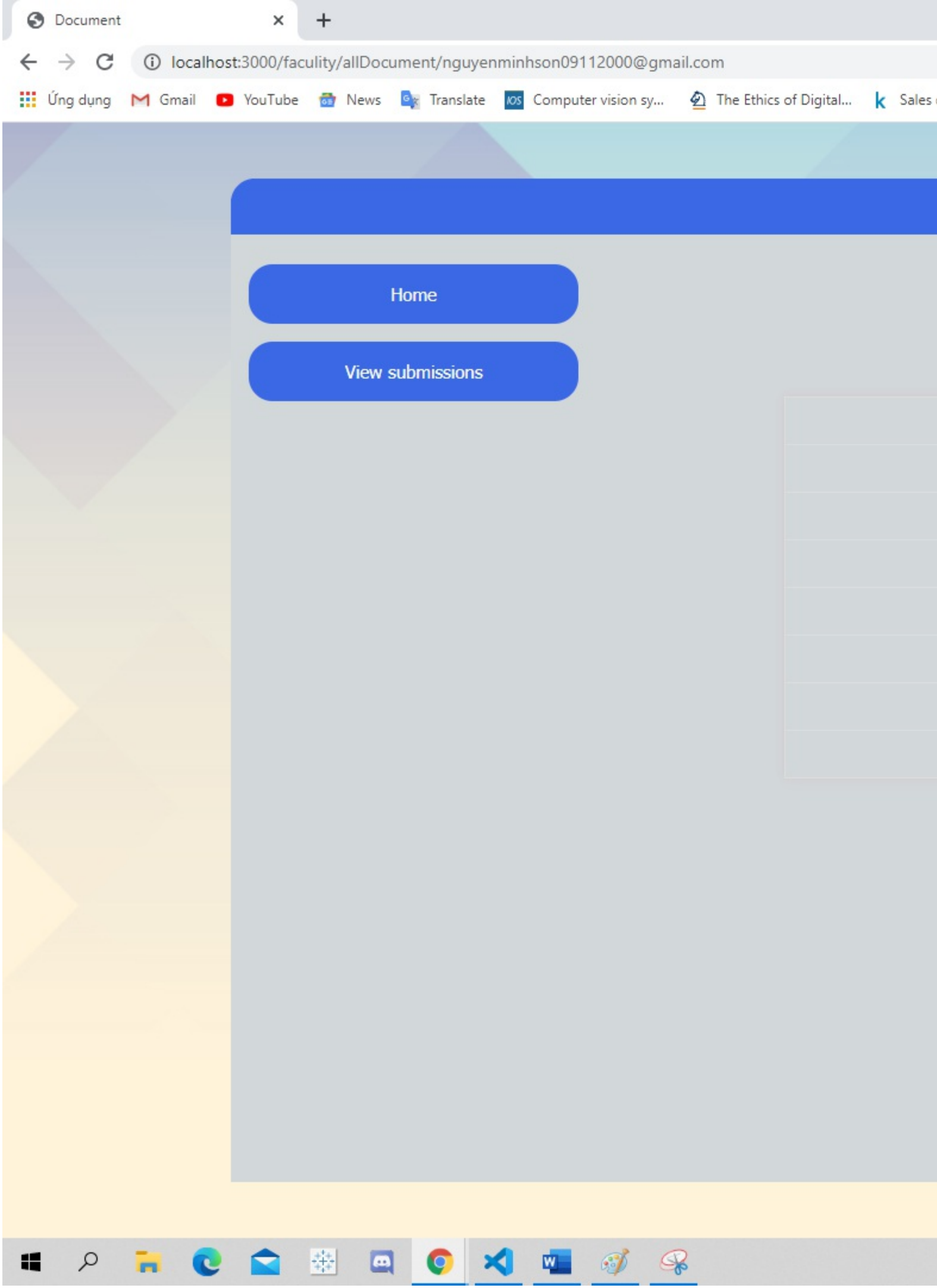

Figure 88:File submitted of marketing coordinator

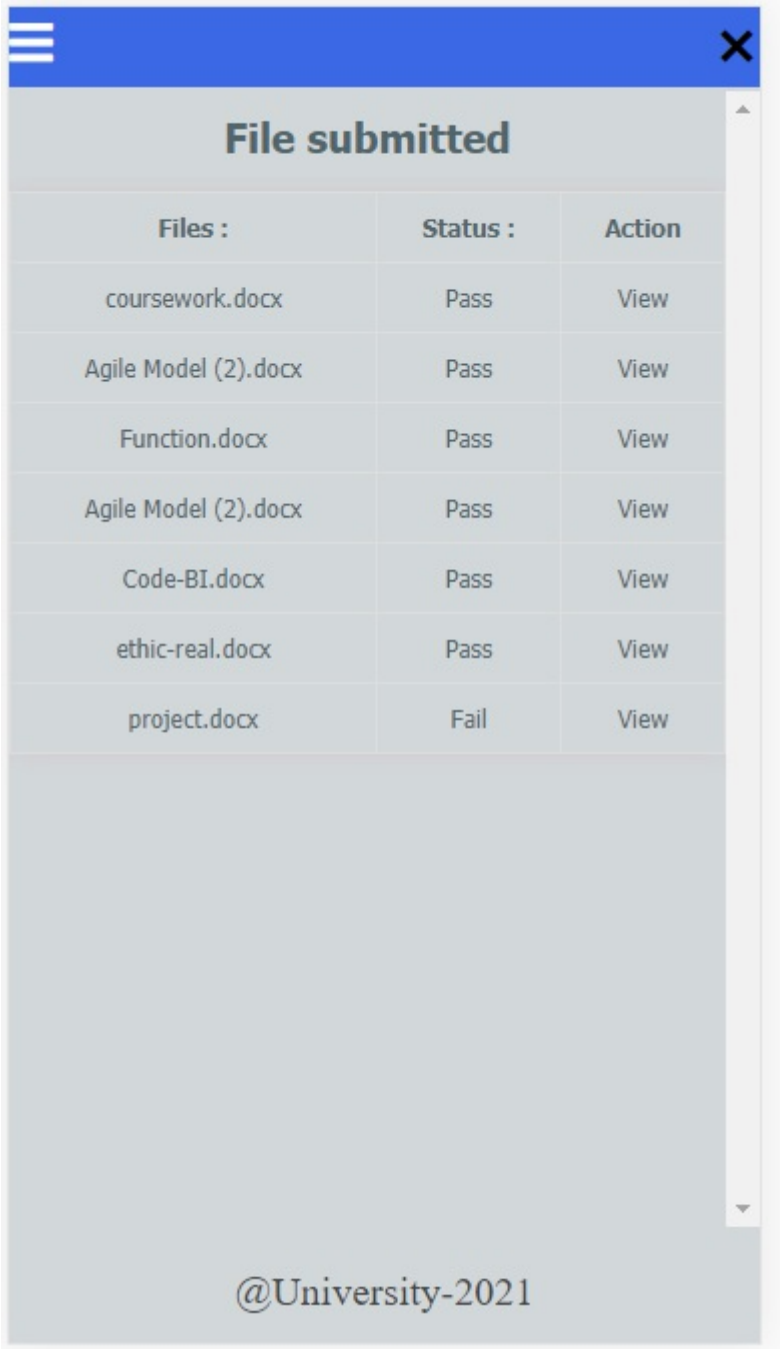

Figure 89:File submitted of marketing coordinator mobile

The Marketing coordinator can view the student's uploaded files and rate them.

Document

localhost:3000/faculty/evaluate/60632e80873a9422804f3b35

Ứng dụng | Gmail | YouTube | News | Translate | Computer vision sy... | The Ethics of Digital... | Sales

You can download the file hereAgile Model (2).docx

Agile Model (2).pdf 1 / 3 100%

Activities: Compare Waterfall vs Agile, their pros and cons (20mins)

Agile SDLC model is a combination of iterative (loop) and incremental (bigger overt
adaptability and customer satisfaction by rapid delivery of working software product.
incremental builds. These builds are provided in iterations. Each iteration typically last
weeks). Every iteration involves cross functional teams working simultaneously on va

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product (prototype) is displayed to the custome

What is Agile?

Agile model believes that every project needs to be handled differently and the existi
the project requirements. In Agile, the tasks are divided to time boxes (small time fra

Iterative approach is taken and working software build is delivered after each iteratio
features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model –

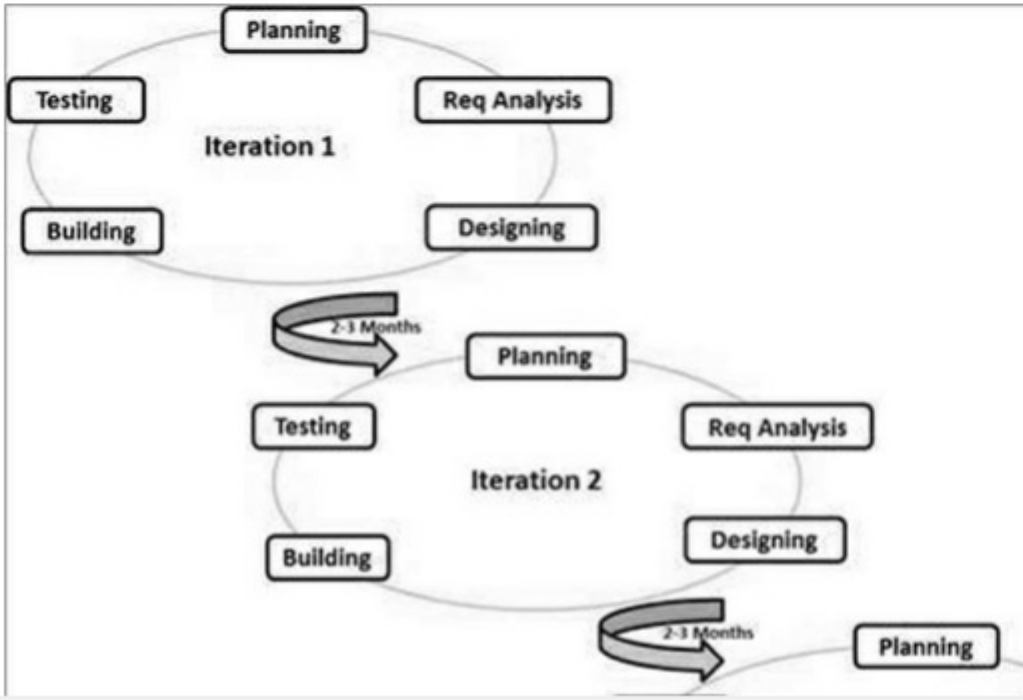

Figure 90:View and evaluate of marketing coordinator

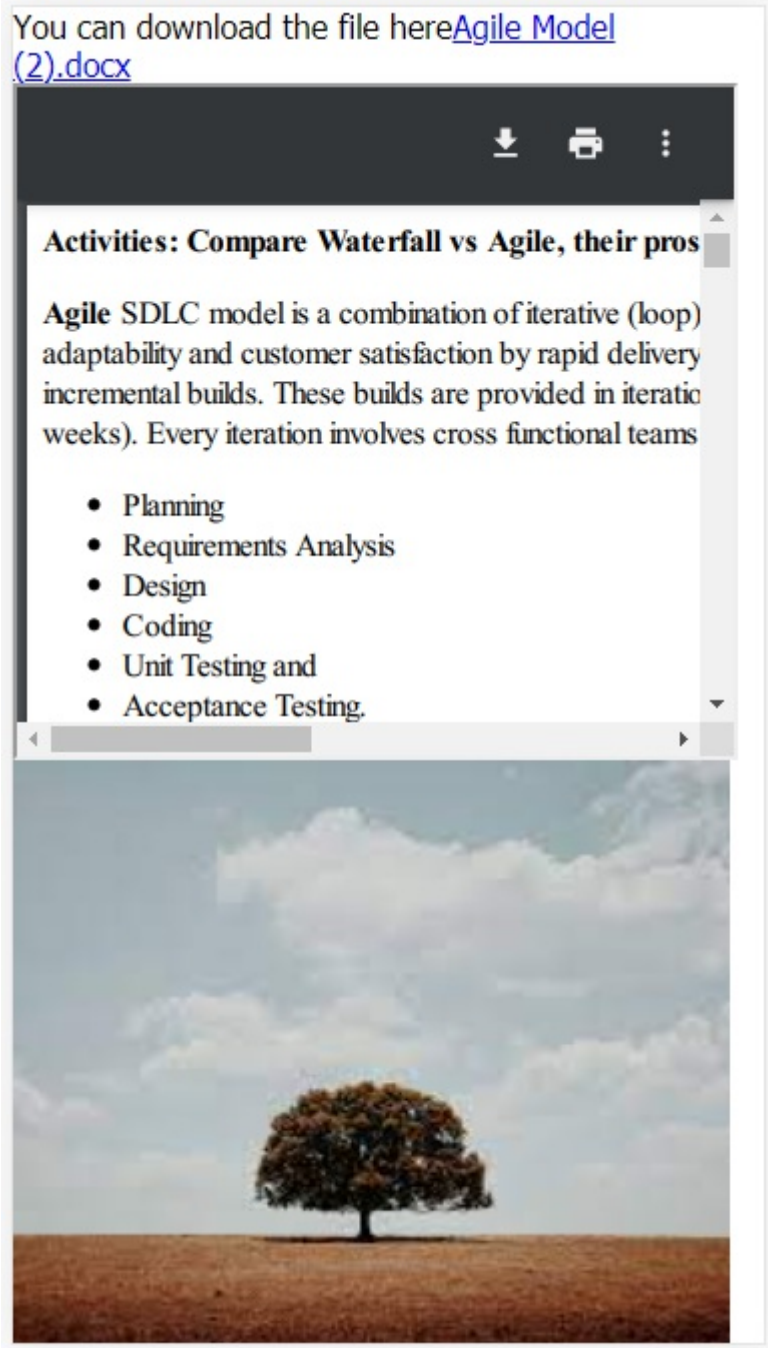

Figure 91:View and evaluate of marketing coordinator mobile

The marketing Coordinator can text with students.

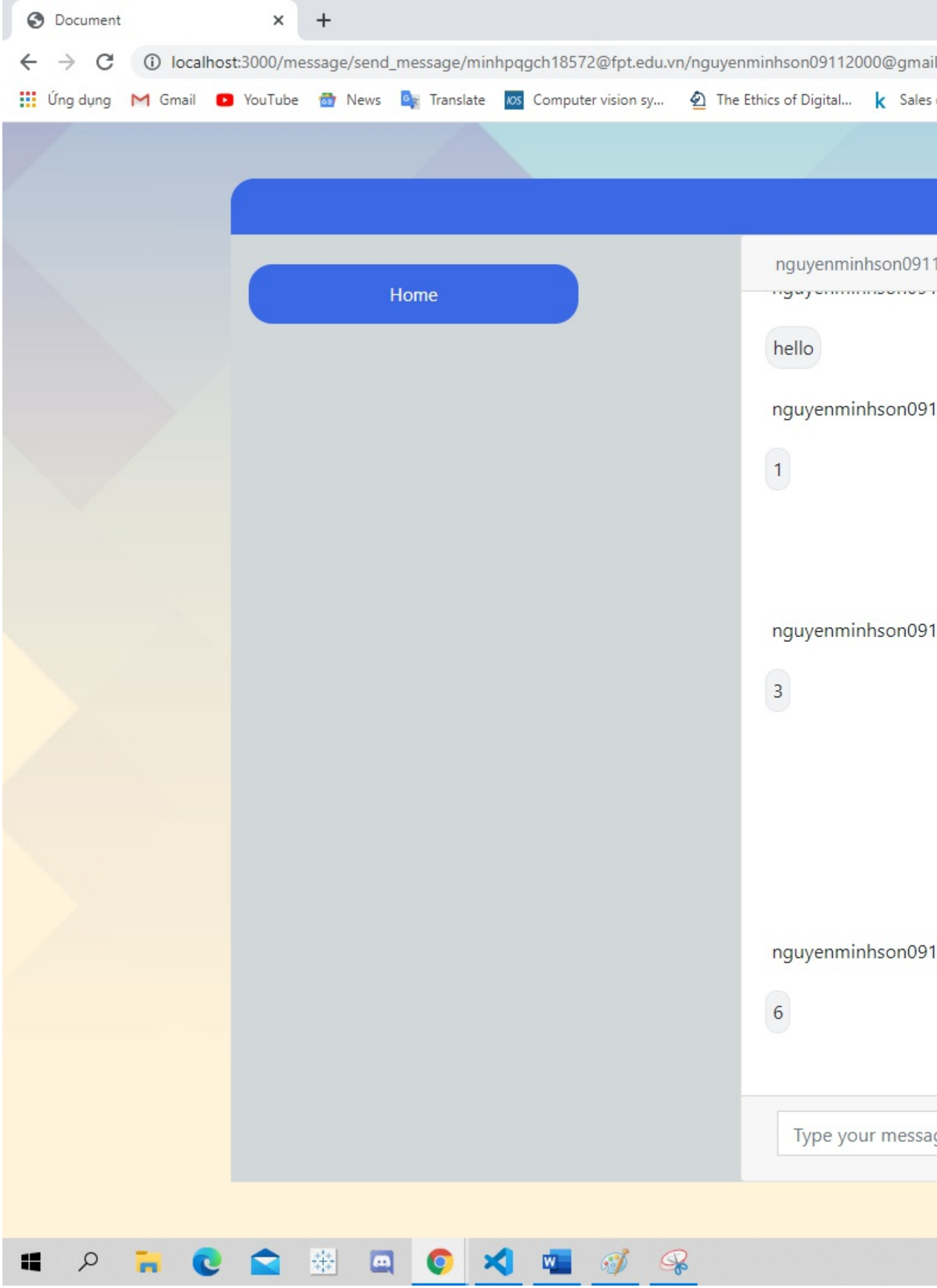

Figure 92:Chat box of marketing coordinator

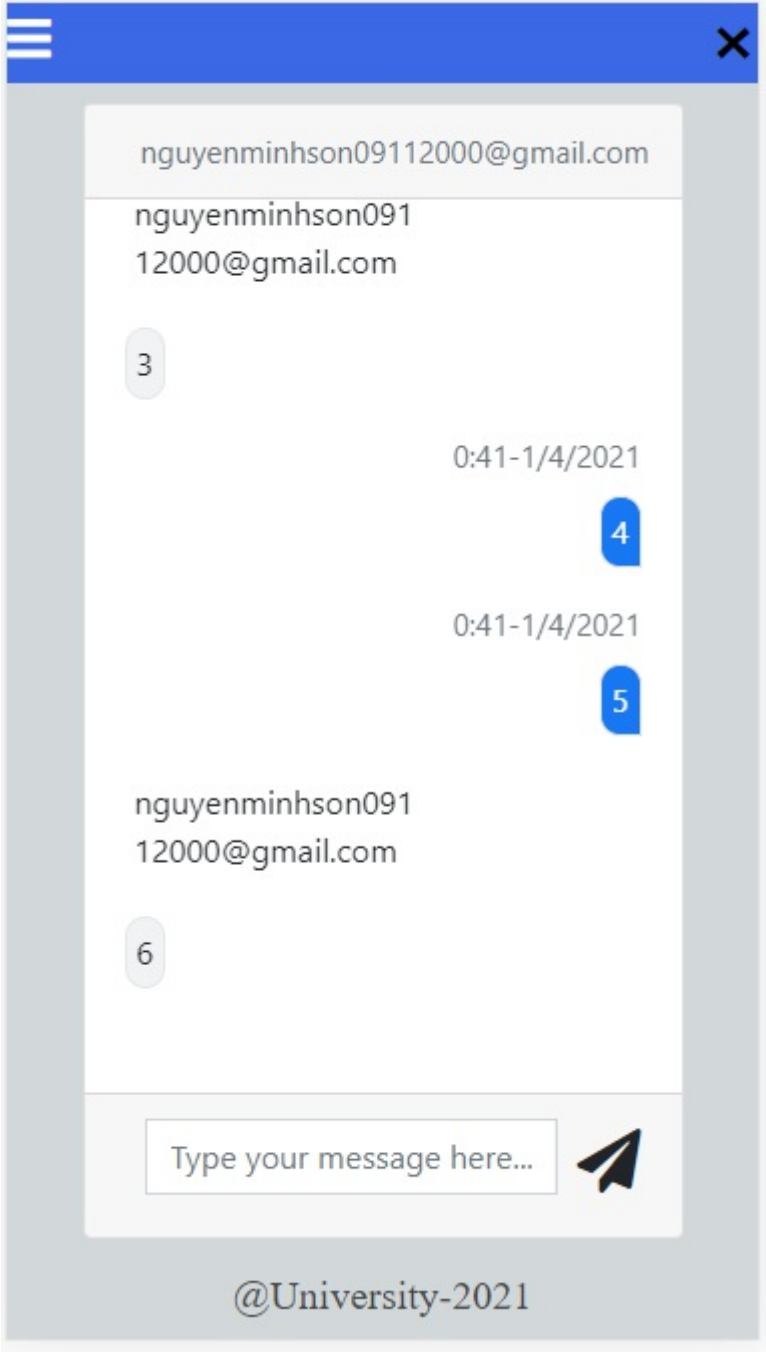

Figure 93:Chat box of marketing coordinator mobile

Usability

The website's interface is easy to recognize and easy to use functions:

- The menu is fixed on the right-hand side and all functions are displayed in the menu bar so the user can use it easily.
- When using on the phone, the menu is hidden and displays the menu icon so the user can click on it to display the menu. The fonts will also be adjusted to suit the phone screen size.
- The website has headers for the user to see what function they are using.
- Naming functions are quite simple so users can remember the functions in the website easily.
- When entering incorrect data, the types of data to be entered will be printed out so that the user can easily recognize and modify.
- The interface is used in creative colors that is comfortable and does not cause inhibition to the viewer.

Accessibility

- The headings use h-tags and are bold, large font easy to see for everyone to see.
- Display the correct data format when input is incorrect.
- Not only using the mouse to navigate and use the website, but the keyboard can also use those functions.
- Img tags all have an alt attribute.

Implementation (Functionality):

Use case Diagram:

Below is an overview Diagram of the whole system. Diagram includes 5 actors that are guest, students, admin, marketing coordinator and marketing manager. They are the people who interact directly with the system. With a clear decentralization, each task and their way of interacting is different on the system. For example, guest can view the selected report but Students do not have permission to do this ...

Figure 94 System use case diagram

- 1. Guest use case diagram:

With guest authorization, there are 3 functions allowed to use and interact with the system, namely login / logout, view profile, view the selected reprot.

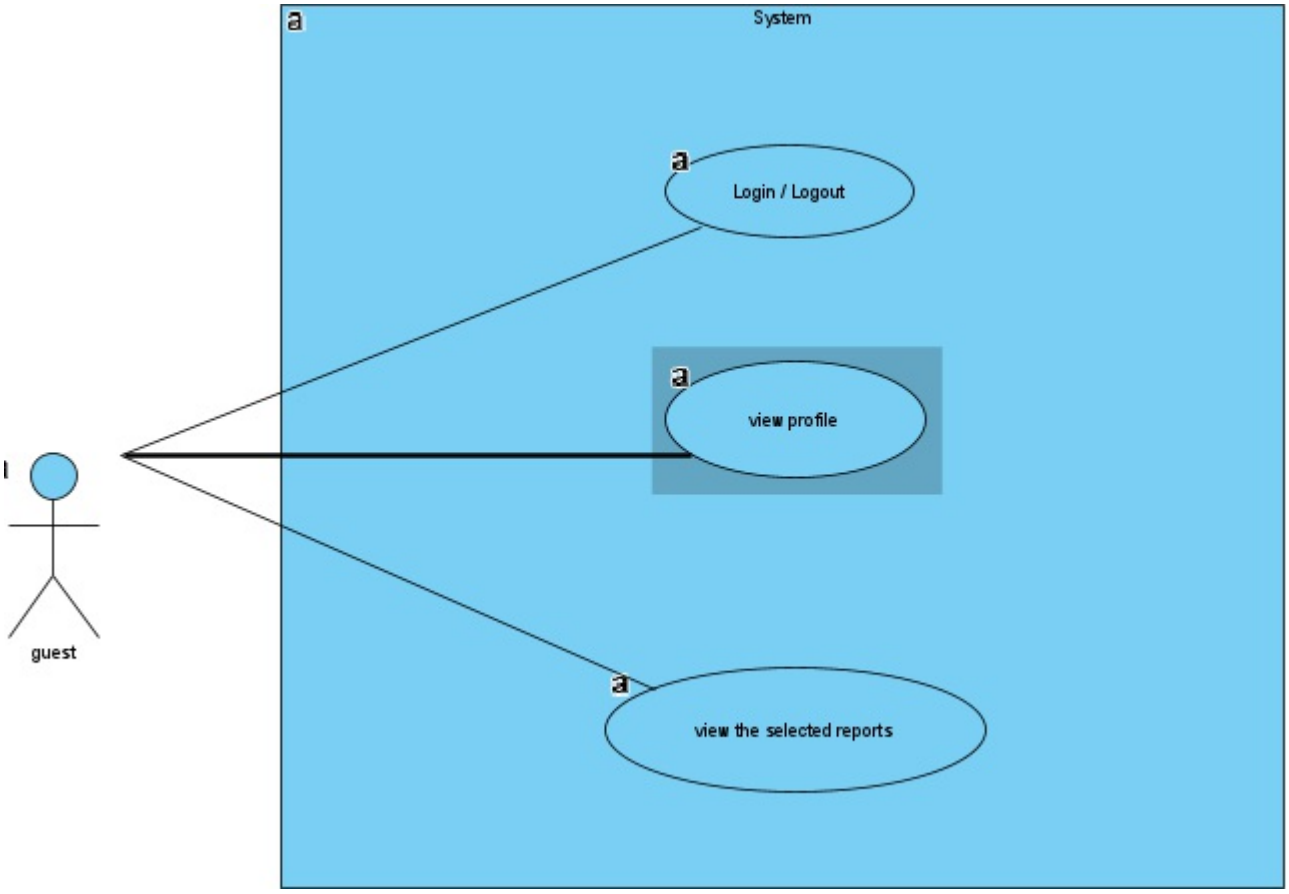

Figure 95 Guest use case diagram

- 1. Administrator use case diagram

With the Administrator decentralization, there are 3 functions allowed to use and interact with the system, they are any system data, Statistical analysis and account management.

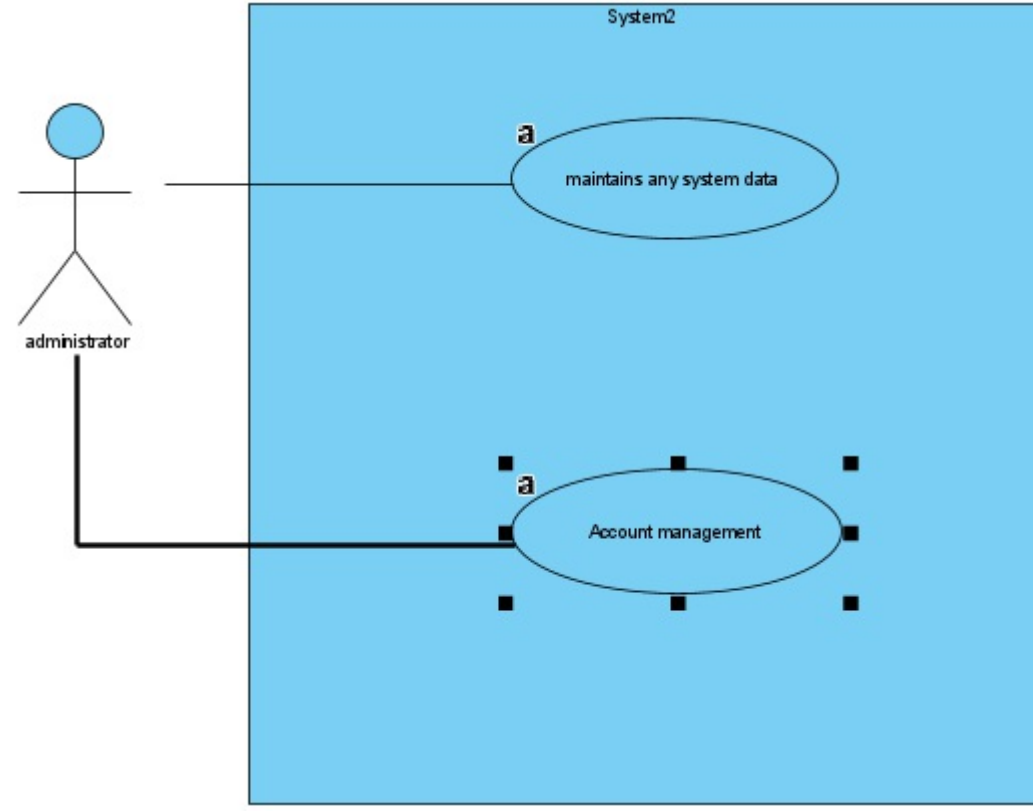

Figure 96 Administrator use case diagram

- 1. Marketing coordinator use case diagram

With Administrator privileges, there are 5 functions allowed to use and interact with the system, namely login / logout, chat box, view the selected reports, give comment, dashboard.

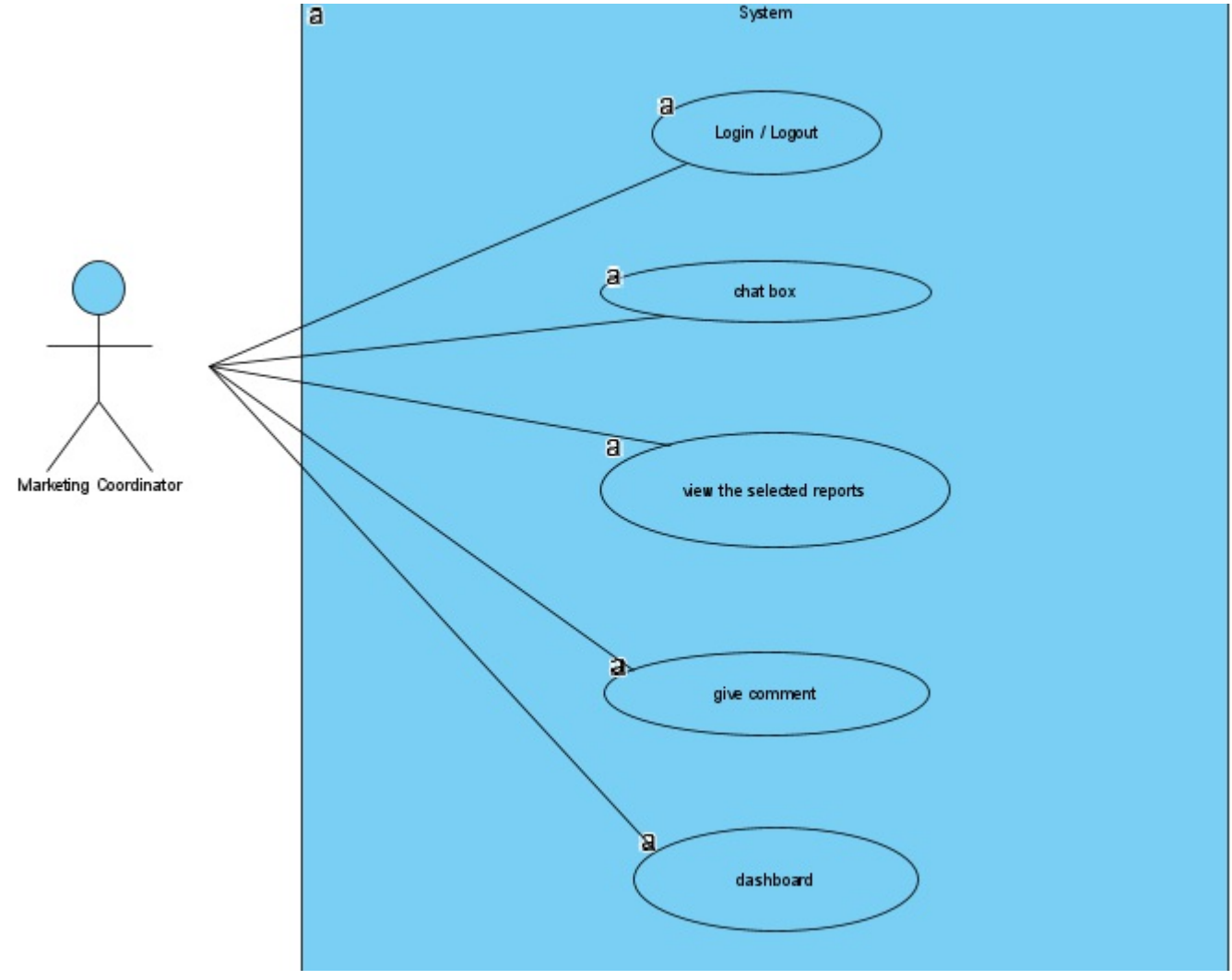

Figure 97 Marketing coordinator use case diagram

- 1. Students use case diagram

With Administrator rights, there are 5 functions that are allowed to use and interact with the system, namely login / logout, chat box, view profile, submit one or more articles as word documents and picture, update articles.

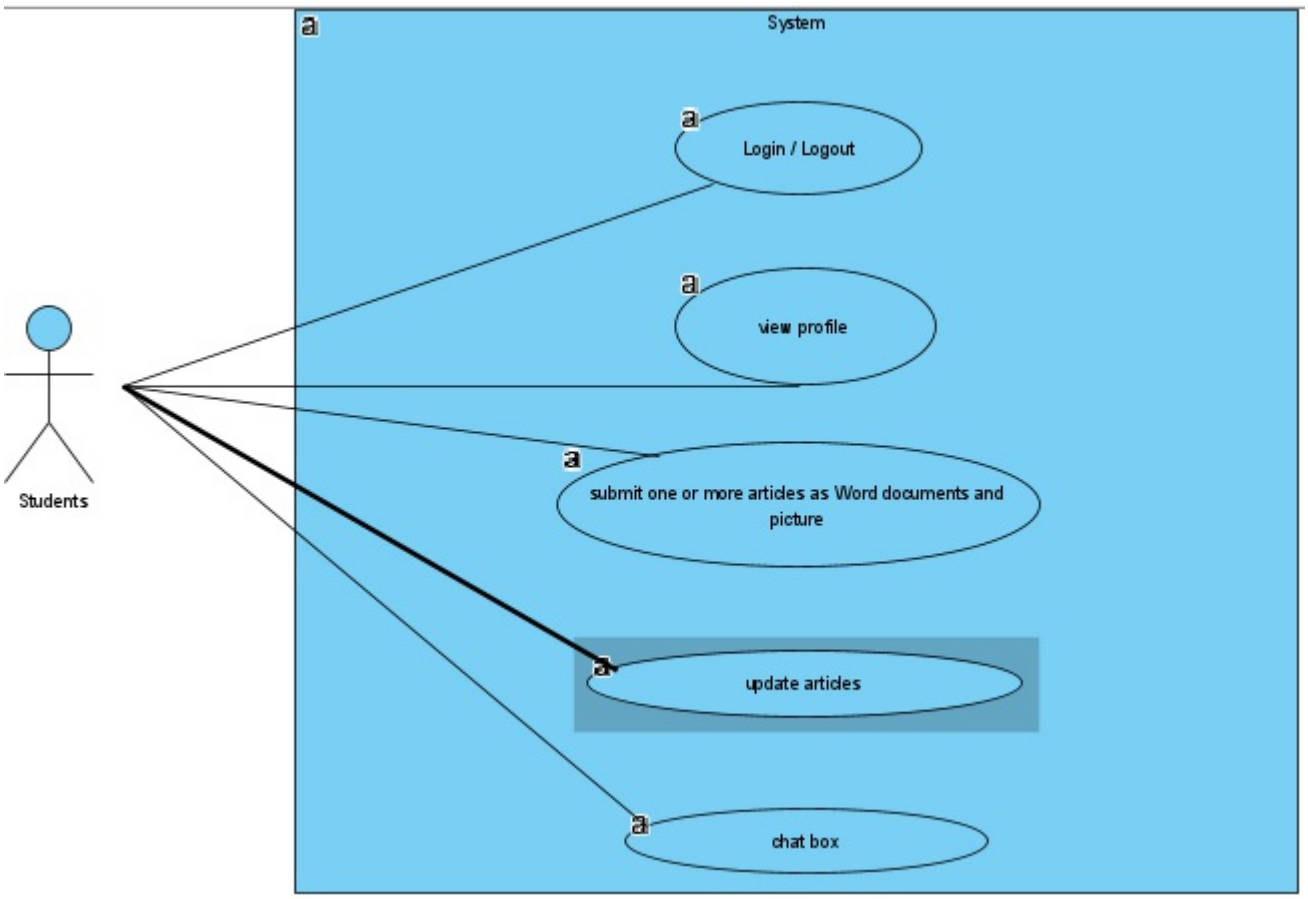

Figure 98 Student use case diagram

- 1. Marketing manager use case diagram

With Administrator privileges there are six functions allowed to use and interact with the system: login / logout, download the selected reports, view profile, view the selected reports, set time, Statistical Analysis.

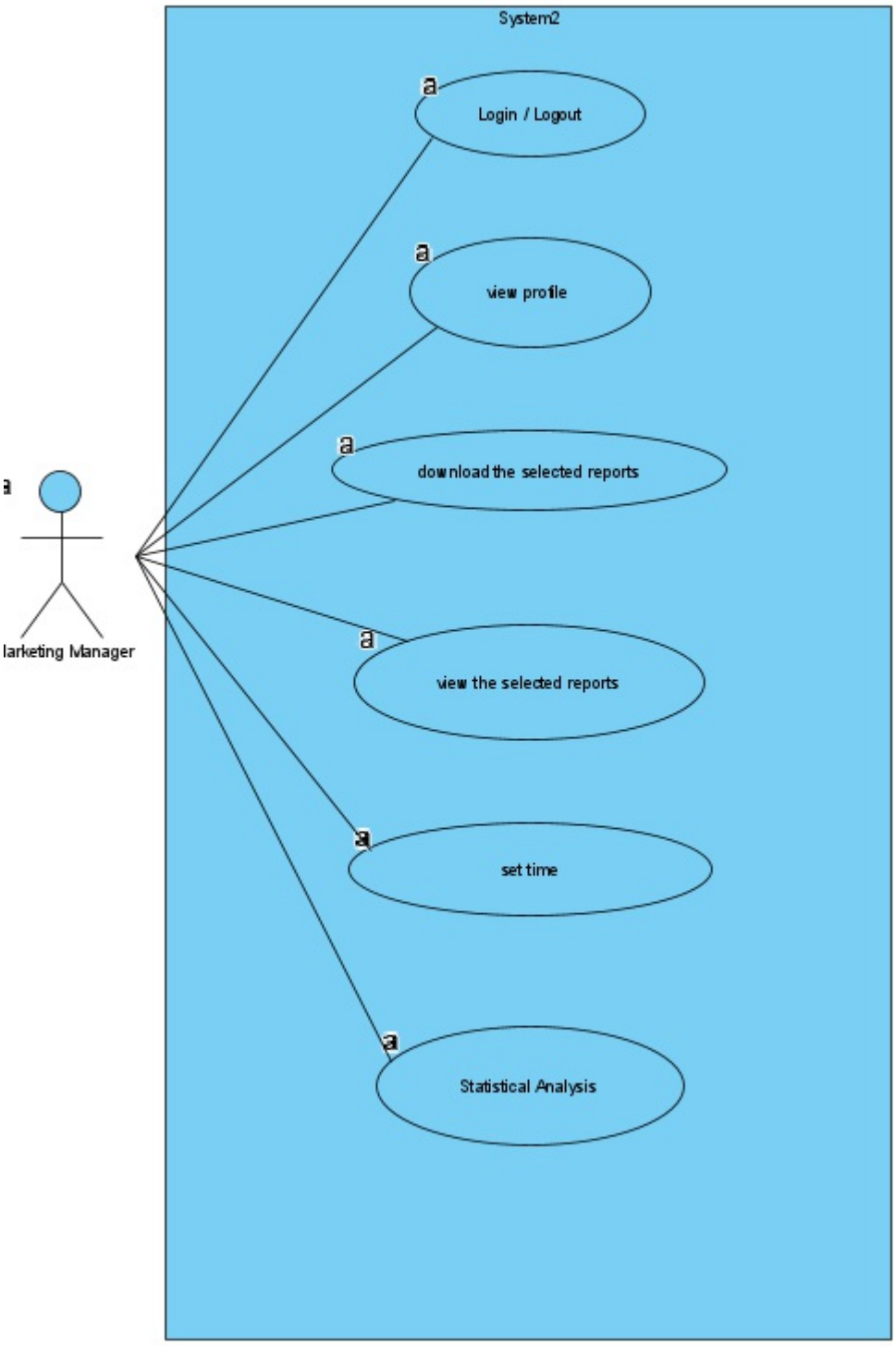

Figure 99 Marketing manager use case diagram

Activity Diagram:

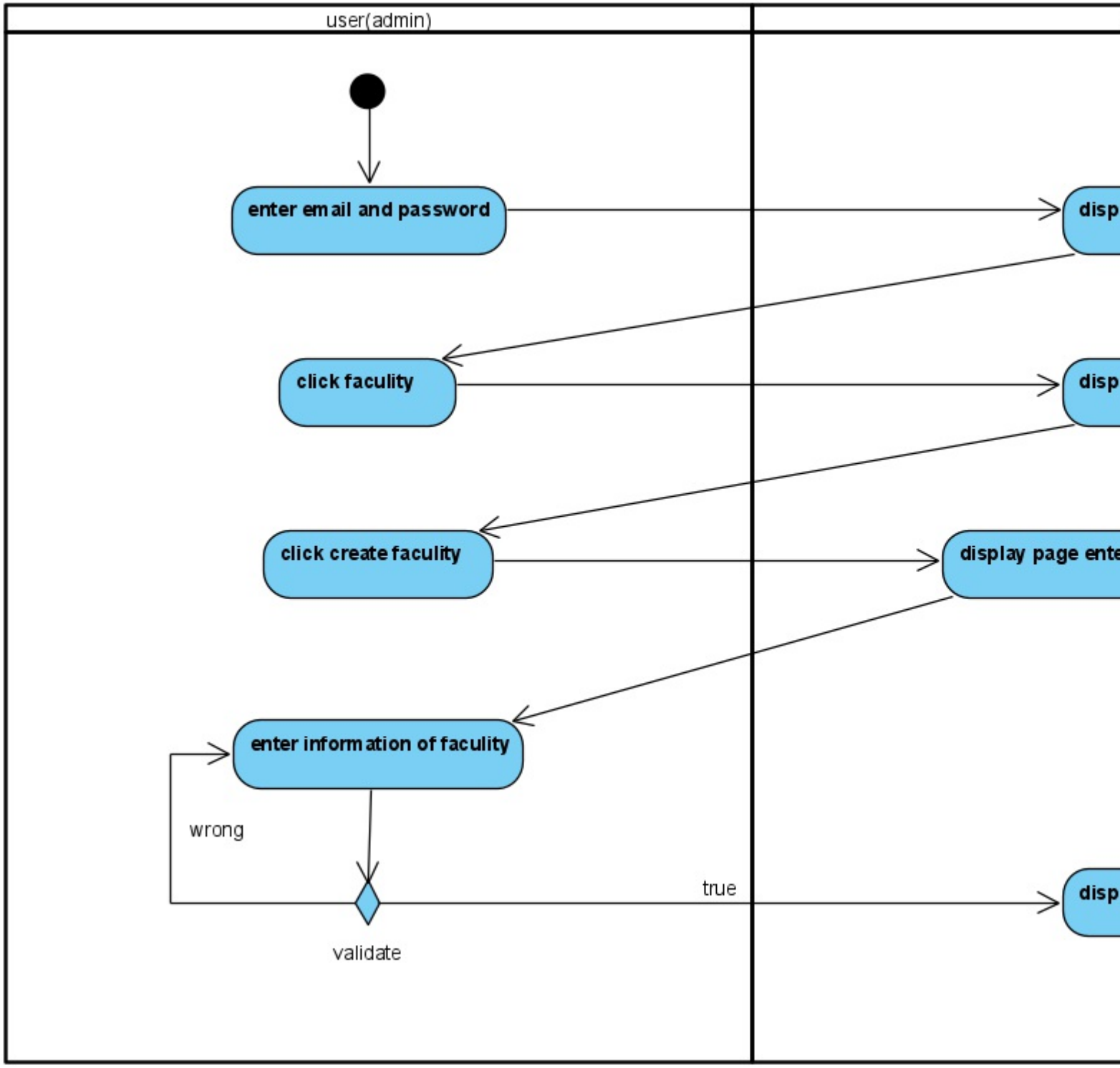

Figure 100 Create faculty activity diagram

Activity create faculty

Creating a new course is admin's right, so the only person interacting with the system here is admin. First, admin log in to his existing account which has been created on the database then the admin information screen appears, click on faculty, the screen displays the faculty page, click on create faculty, the screen displays the page to Enter information for faculty. Then it will divide two cases where an admin filled correctly and enough information, the system will now successfully create the class and send you to the faculty of all, and if wrong, the system will report an error and ask you to correct it. it's correct.

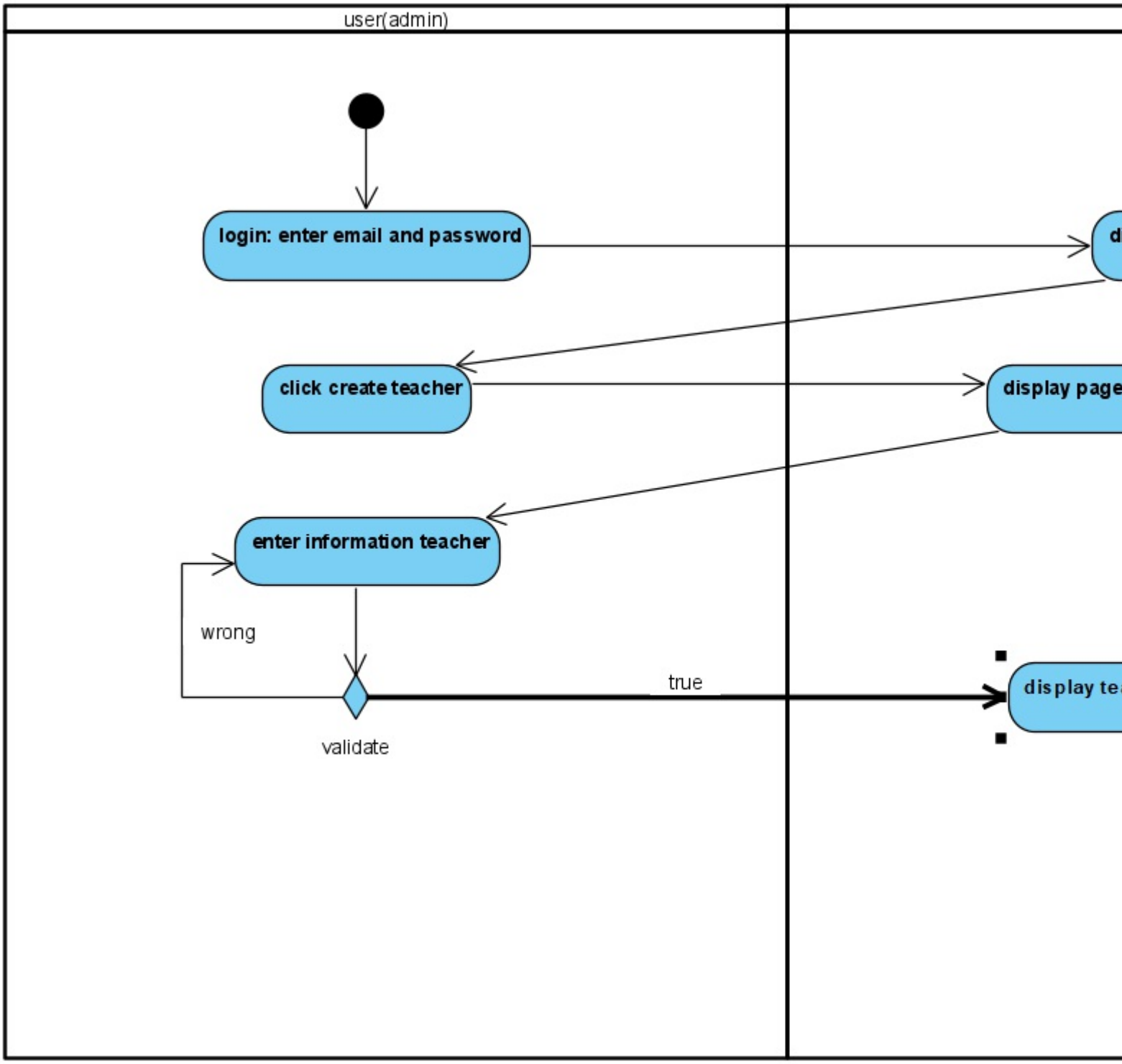

Figure 101 Create teacher activity diagram

Activity create teacher

Creating a teacher account is an admin function, first the admin logs in to an available account exclusively for the admin on the system. The admin information screen appears. Click on create teacher, the screen will display the teacher information page. Here appear two problems, the first problem if entering the correct screen will take us to the account information page of all teachers. Also, if it is wrong, the system will notify you of a mistake and ask to correct the wrong input.

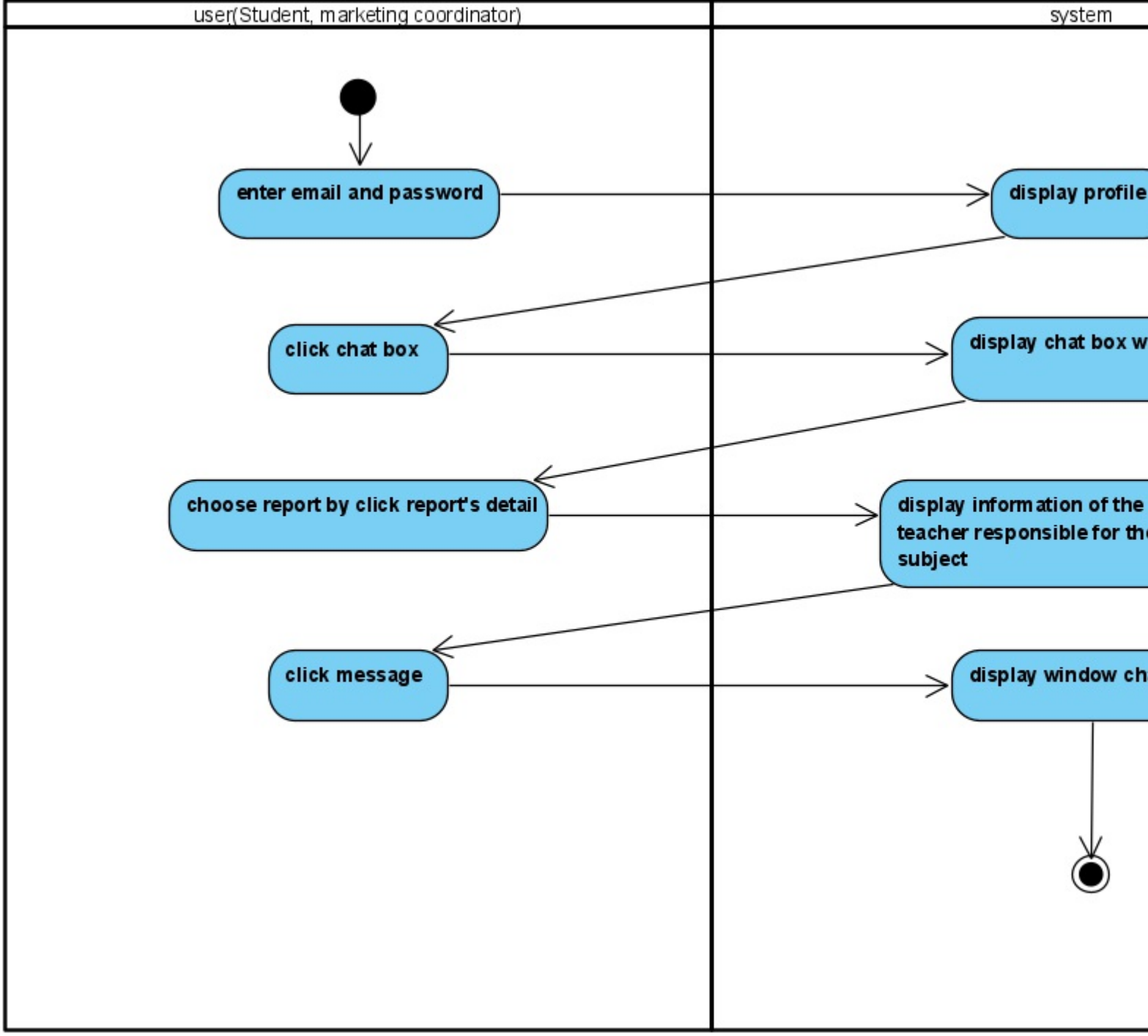

Figure 102 Chat box activity diagram

Activity chat box

For chat box function, we created for two roles: Marketing Coordinator and Student. The way its work is very simple, at the beginning, users logged in the system and go straight to their profile page. After that, user can click on “chat box” button and will be lead to the “chat box display” website. In that page, user can choose to see which chat box they want to discuss between submitter and marker by click on “report’s detail”. By clicking on that button, the system will display information of the individual responsible for that submission. Last thing they need to do is click on “message” and the system will show the chat box for them to interact.

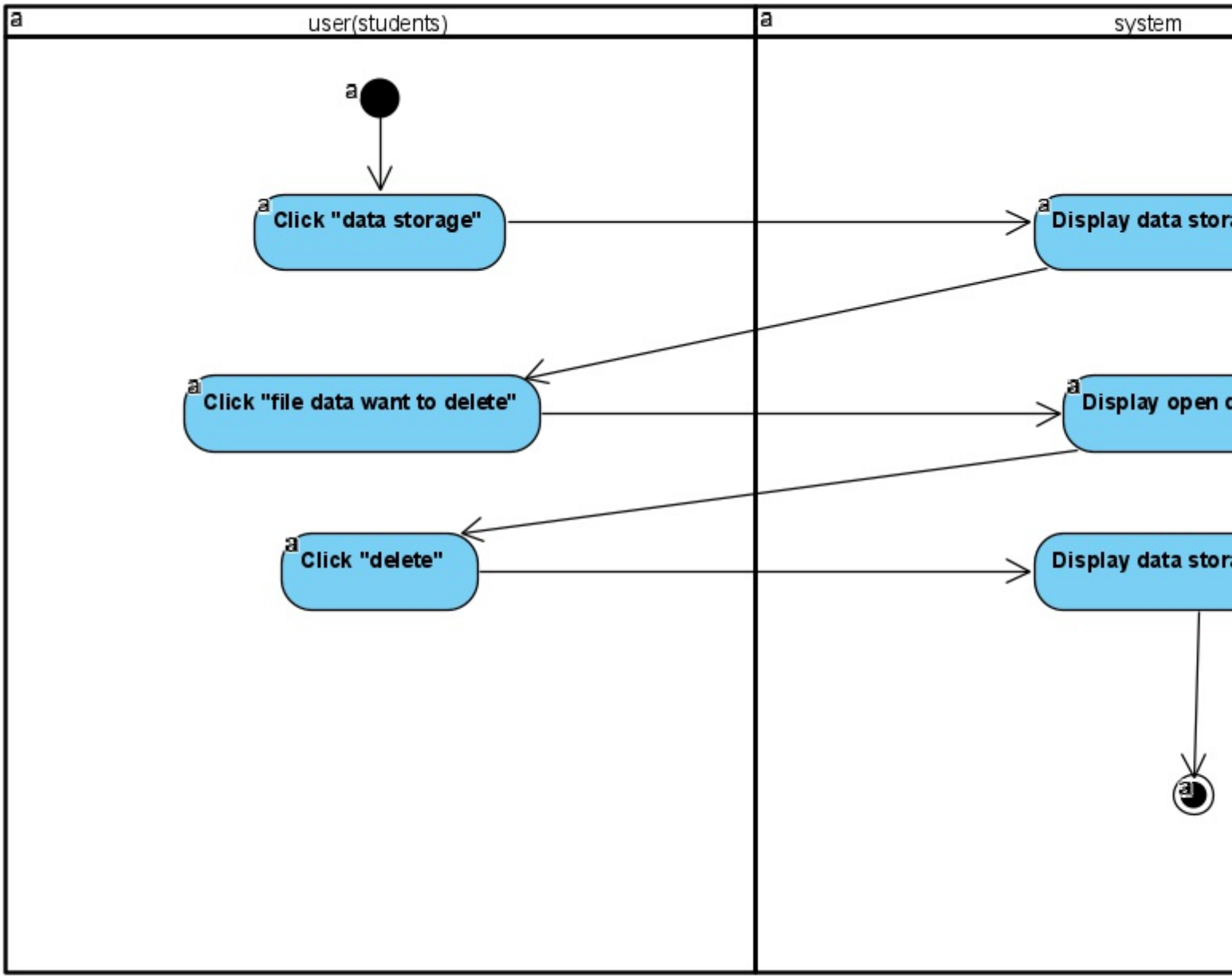

Figure 103 Delete/Update article activity diagram

Activity delete/update articles

This function is for Student only, it is use for student to edit their submission after they submitted it on the web. First, the student need to click on “data storage” to see their submission, the system will display a list of their submission, all they have to do now is click on the article they want to delete/update. after their page finish loading, they can click on update or delete button of the submission they want to update/delete. When they done all of that, the system will return them to the data storage page.

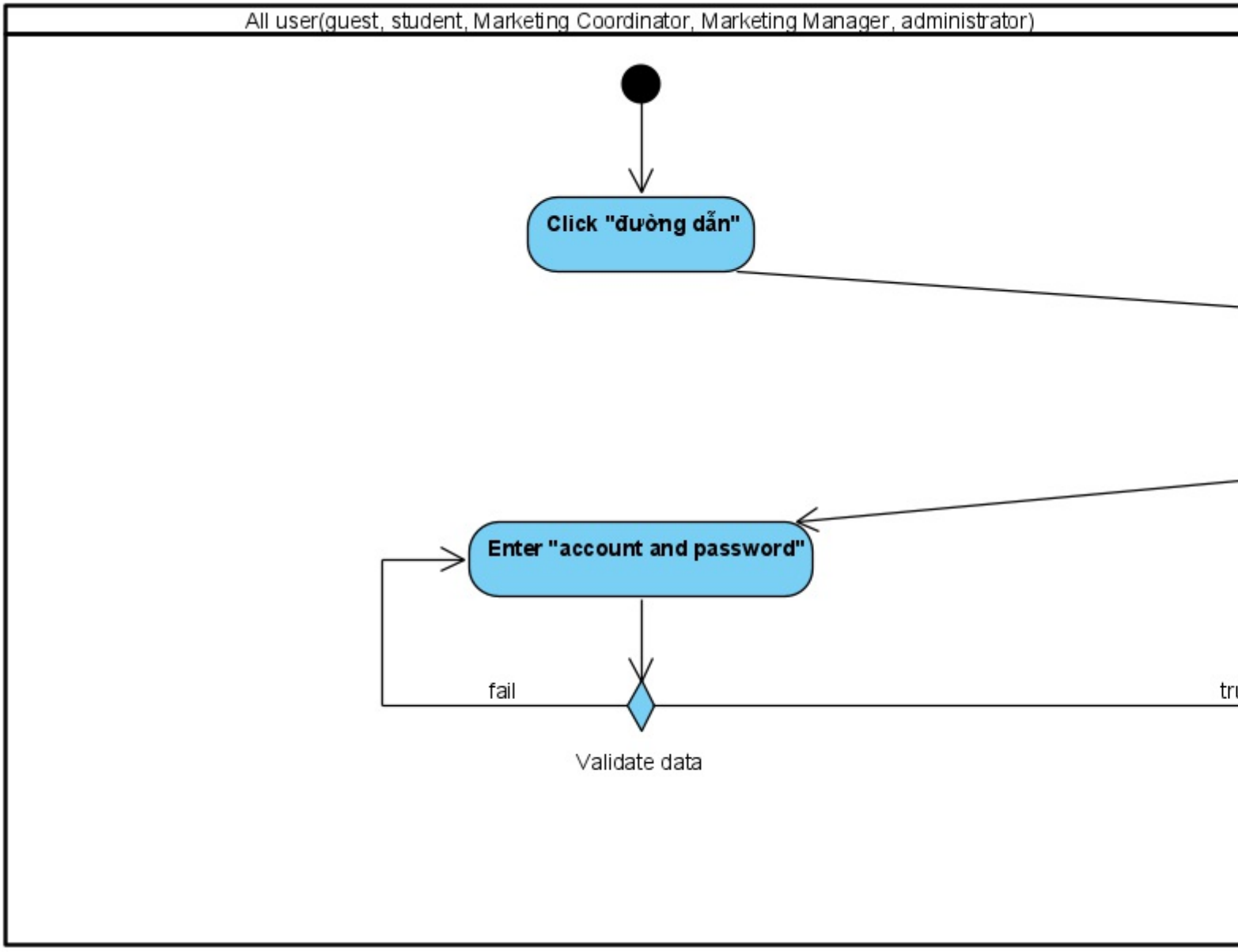

Figure 104 Login activity diagram

Activity login

The login function is created for all users. The purpose is very simple, to distinguish their role. First they need to click on the link which lead to the website, the system will immediately show them the login page. After they put in the correct email and password of their account, the system will let them in the system and show up their profile.

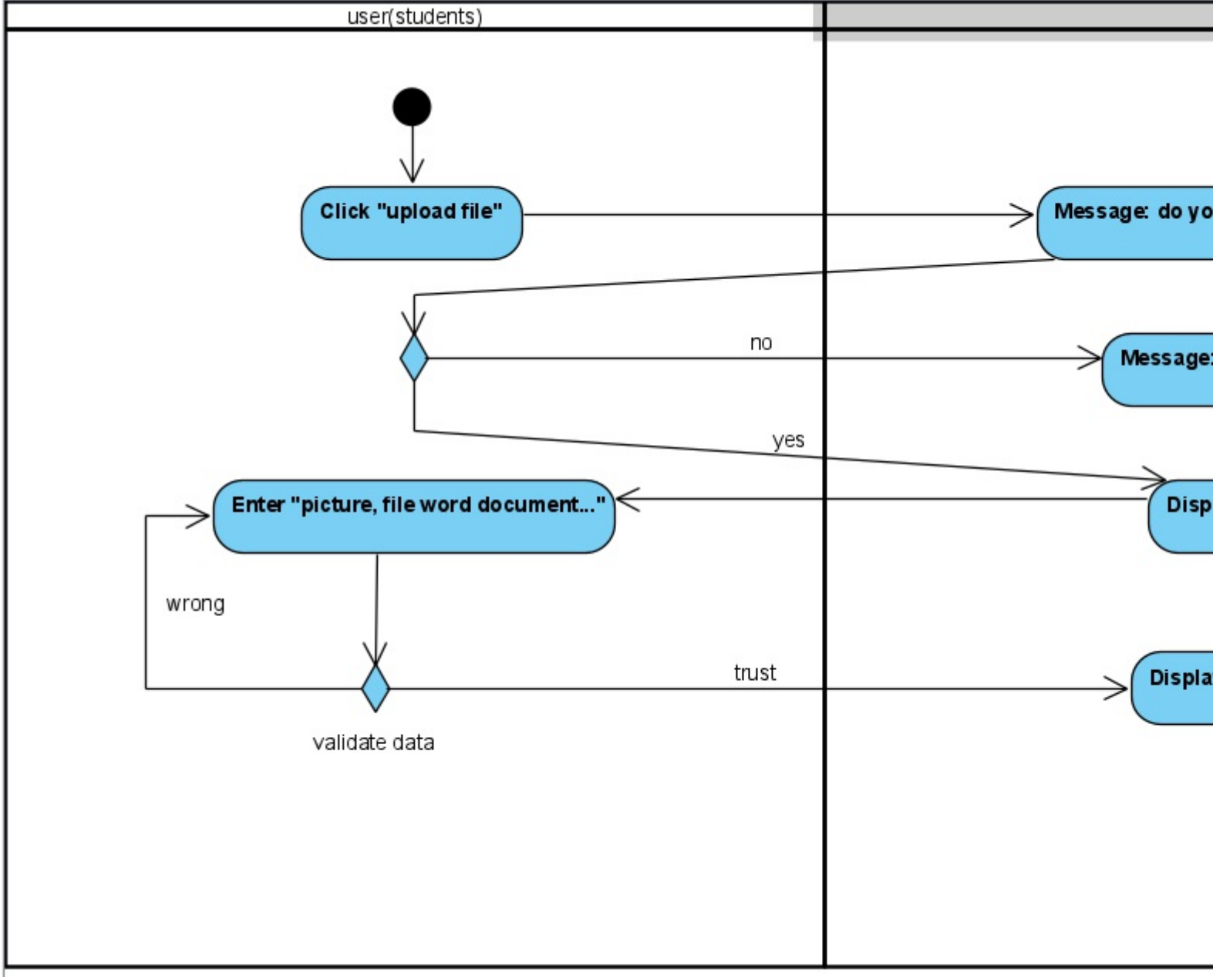

Figure 105 Upload activity diagram

Activity upload file

In this project, only student can upload file to submit their article. Before they can upload their submission, they must agree to the term of the web, if they choose yes, the system will bring them to the submit page. To upload, their file must follow some rule of the web and after the file bypass them all, the file will be up on the web and the system will show their personal archive.

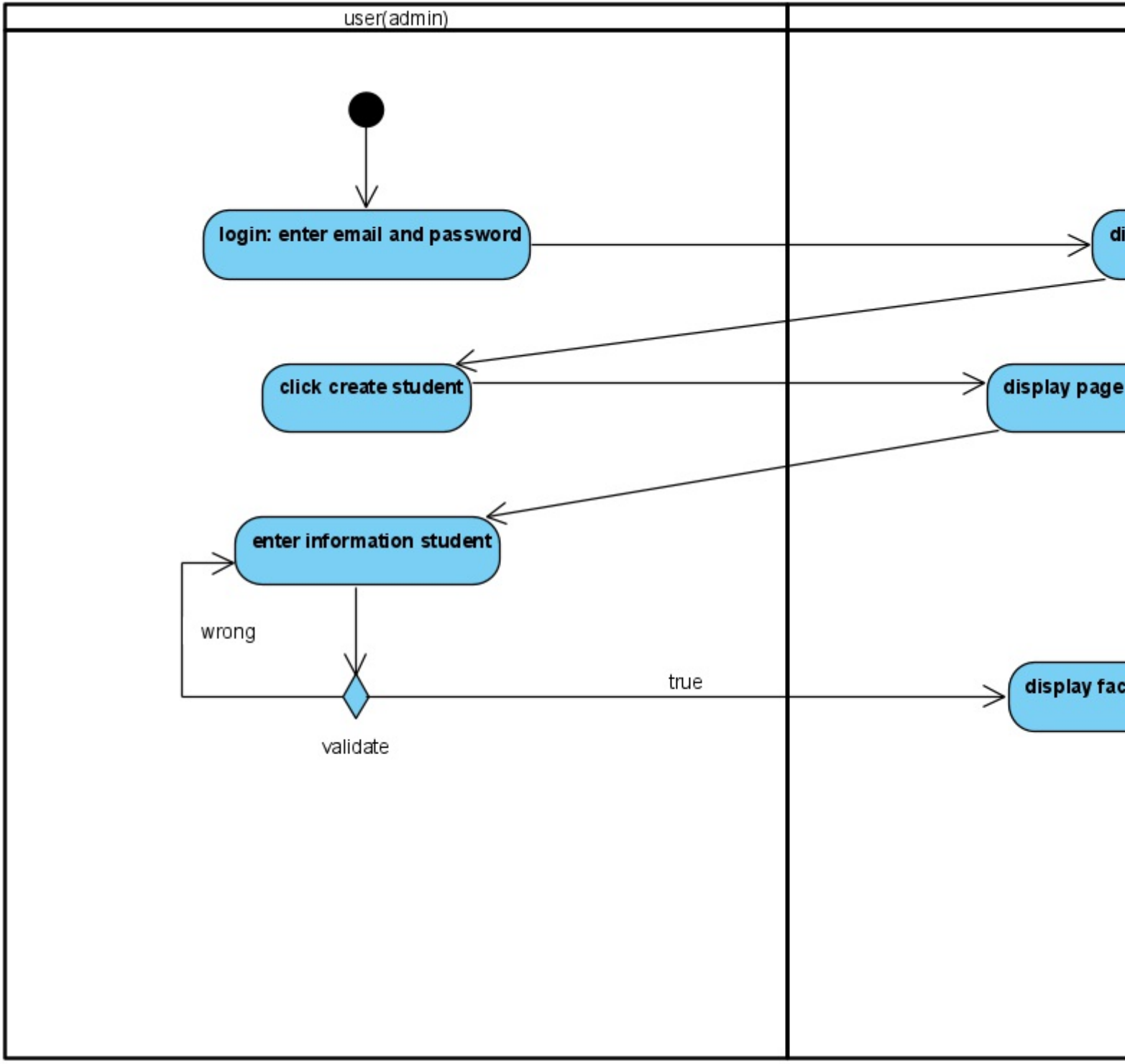

Figure 106 Add student activity diagram

Activity add student to faculty

Except administrator, all other role's account can be created by admin including student. To do that, user must logged in by an admin account then click on "create student" After being brought to the create student page, they can enter student's information and click on create, when the system finish validating all the information standard, the account will be create and jump straight to the faculty's all students page.

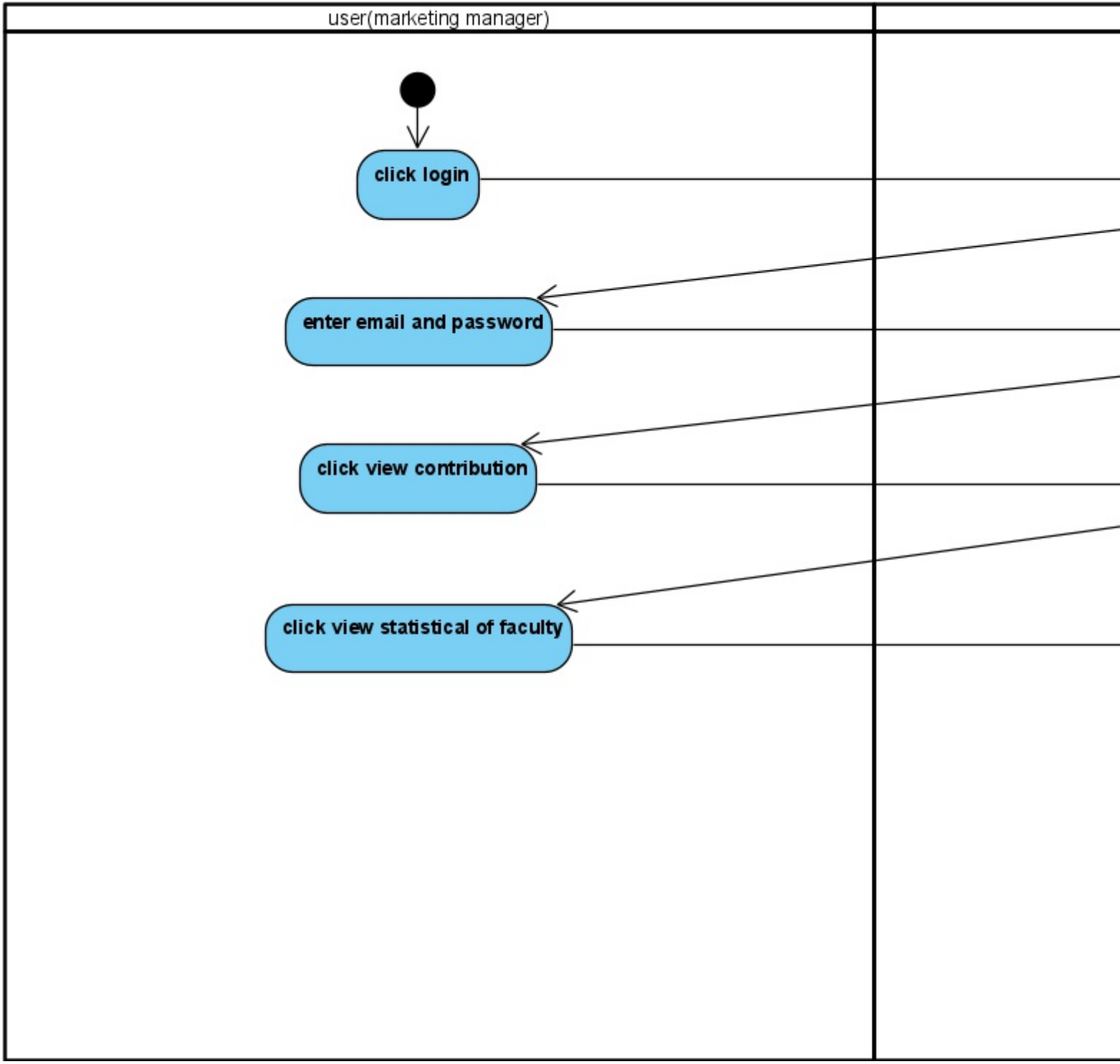

Figure 107 Dashboard activity diagram

Activity Dashboard

The dashboard is one of the function requirement of marketing manager role. After the logged in, they can click on view contribution and the system will bring up all faculty for them to choose, when they click on the button " view statistical of faculty" the system will display that faculty dashboard for the user.

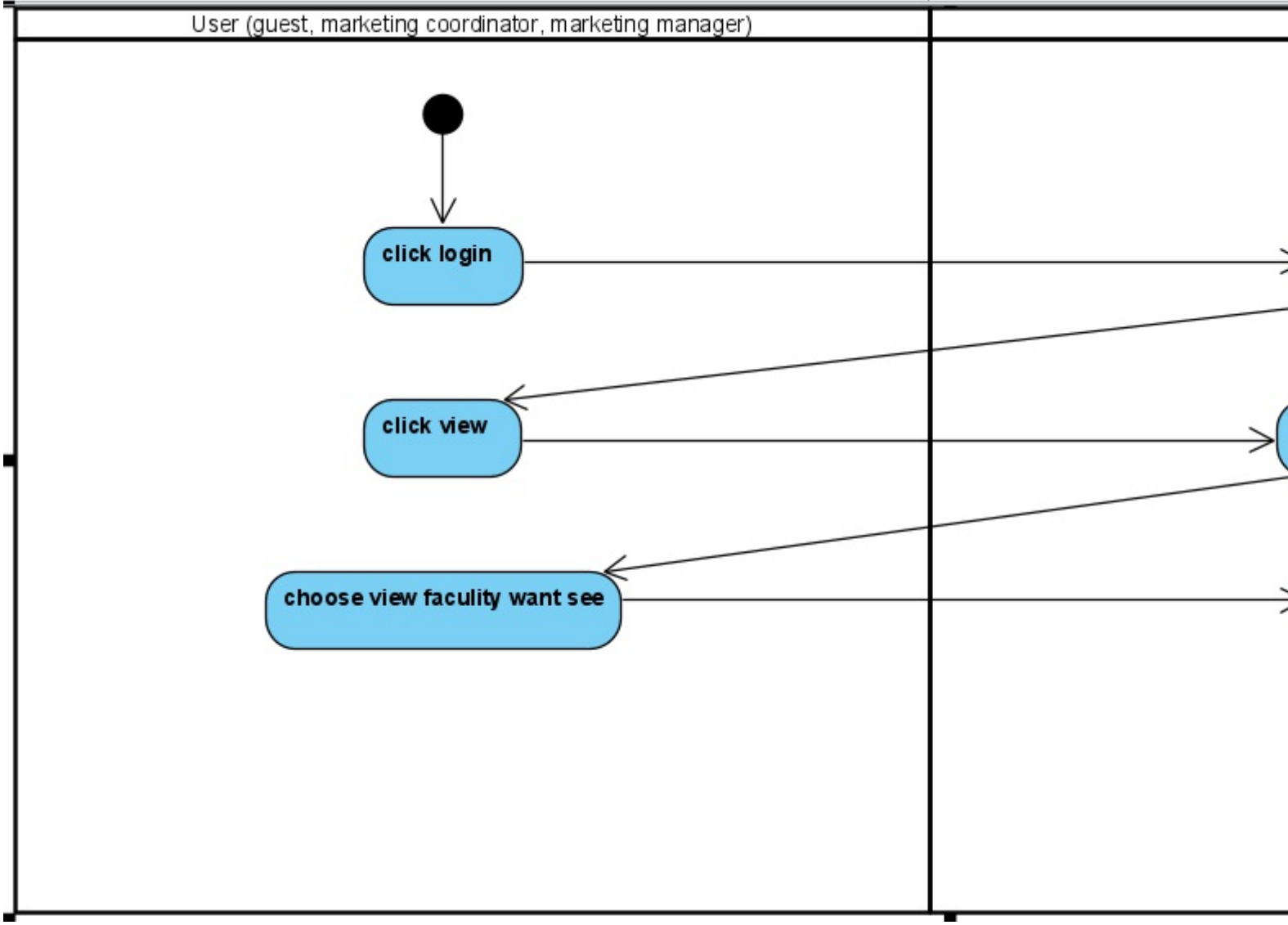

Figure 108 Download activity diagram

Activity download the selected reports

In order to download the selected report, the user must be Marketing manager, marketing coordinator or guest. For Guest and marketing coordinator, they must click on view article so the download button can show up. But with marketing manager, you can click on “download contribution” button, the system will bring you all the faculty. From that, the manager can click on view and selected article to download without have to read it.

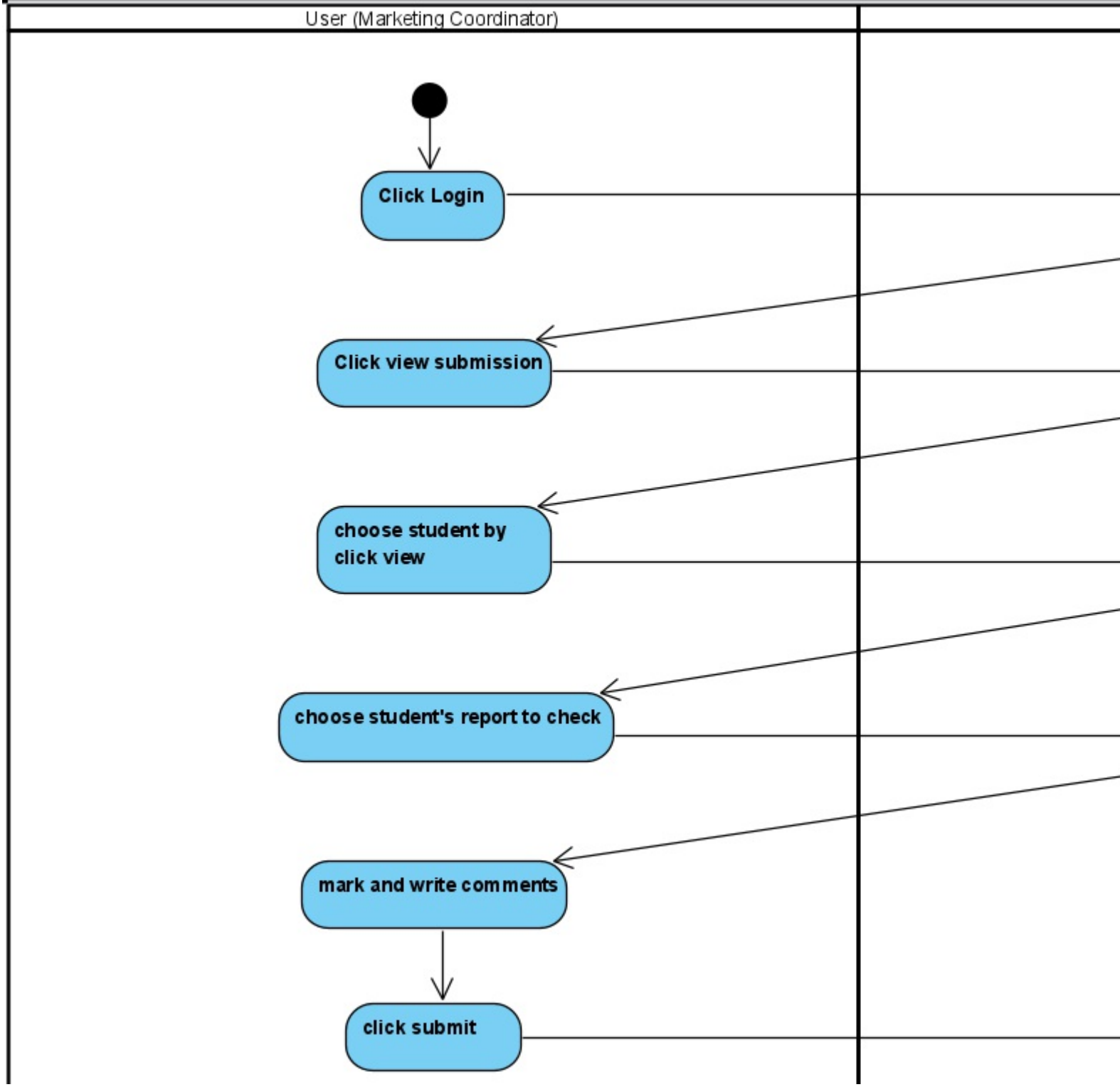

Figure 109 Comment activity diagram

Activity give comments

The comments function is only for marketing coordinator to work on after they logged in and when they are marking a submission. After they click on student, every submission of that student in that faculty will show up, the coordinator can choose one of them to mark. The last activity is when they finish grading, write their comment on the comment section of the article and press submit.

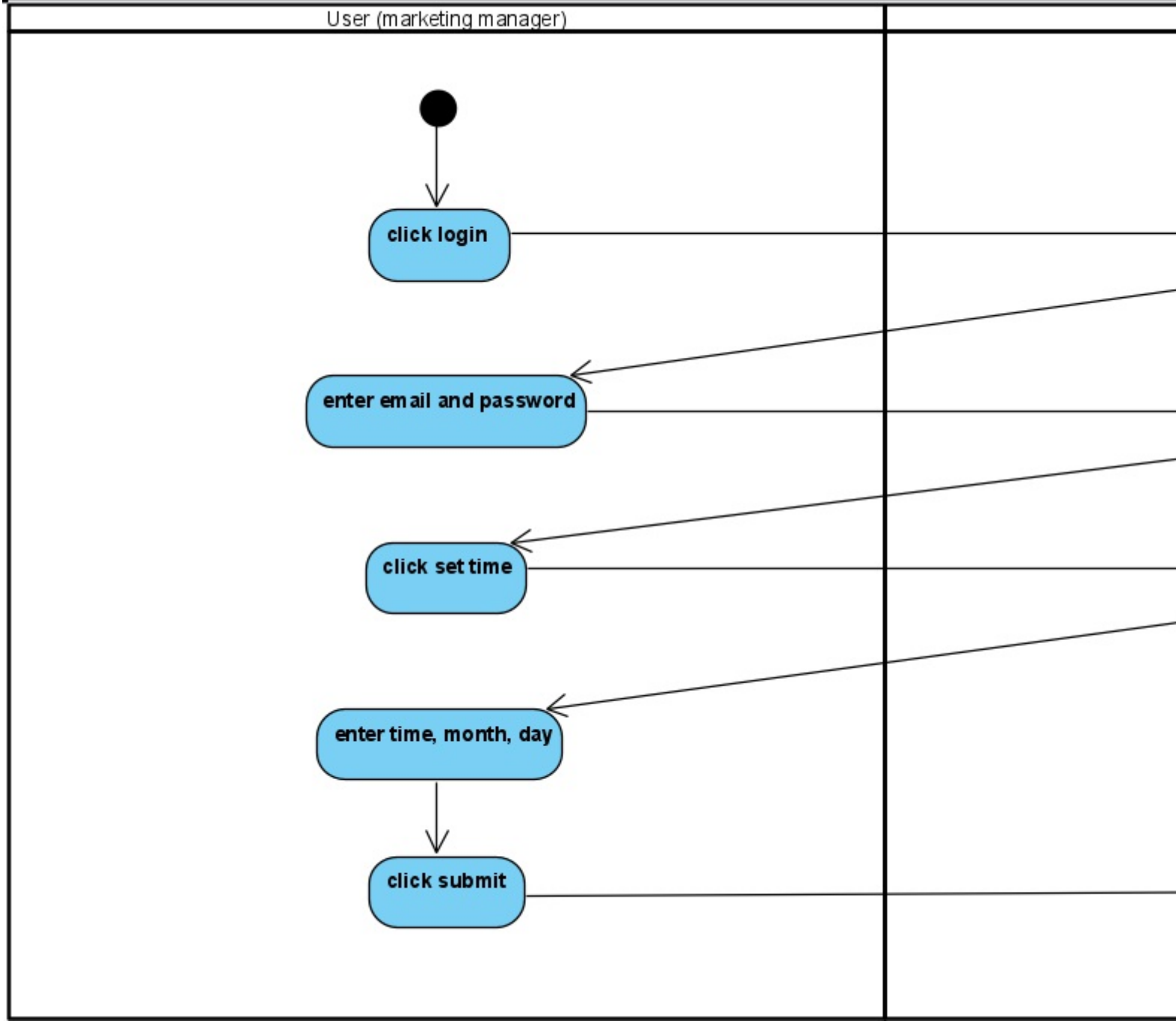

Figure 110 Set time activity diagram

Activity set time

Set time is the activity where only Marketing manager can do. When they finished logging in, they can click on set time and the system will display a Date and Timer for them to set. Entering time, month, day and year, they can click on submit and set that time for all student.

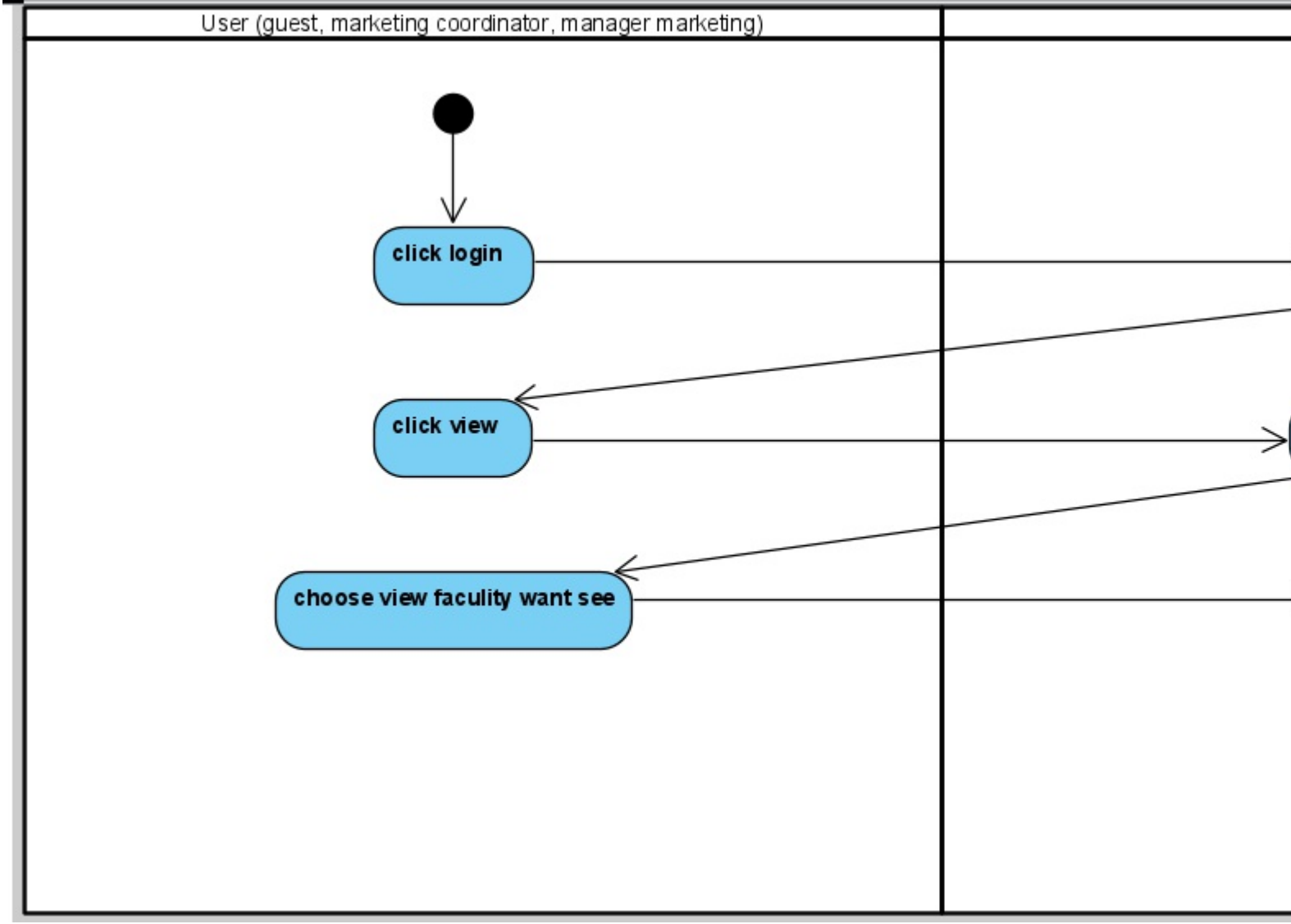

Figure 111 View article activity diagram

Activity view the selected reports

Viewing article can be performed by Marketing manager, marketing coordinator and guest. Just like every other activity, they must login first. After click on view, the system will lead them to the page where it display all faculty. The only thing they have to do left is choose the faculty and click on the article they want to view.

Function

Login

To be able to log into the system, users will have to enter email and password in the login page. The server side will receive the information that the user submits and will check if there is email in the database, otherwise the user will be taken back to the login page.

```
let checkLogin = async (req,res,next)=>{
    try {
        let user = req.body.email;
        await AccountModel.findOne({
            email:user
        })
        .then(user=>{
            if(!user){
                var message= "Username or password is invalid"
                res.render("login",{message:message})
            }else{
                req.user = user
                next();
            }
        })
    } catch (error) {
        console.log(error)
        return res.status(500).json({
            message : "error sever",
            status: 500,
            error : true
        })
    }
}
```

Figure 112: Login code

In the event that there is data in the database, the server will return the user's token in the cookie and examine that person's role and return it to the homepage with each different role. Since the token can be decoded, to avoid the disclosure of important information, we only store "_id" in the token. When signed token will have a period of 1 day, should not be too long to avoid being hacked.

```
let loginController = function(req,res){
    bcrypt.compare(req.body.password, req.user.password, function(err,result){
        if(err){
            return res.status(500).json({
                message : "error sever",
                status: 500,
                error : true
            })
        }
        if(result){
            let token = jwt.sign({_id : req.user._id},'minh',{expiresIn :'1d'})
            res.cookie("token",token,{maxAge: 24*60*60*1000});
            let user = req.user
            res.cookie('email',user.email, { maxAge: 90000000, httpOnly: true });
            res.cookie('id',user._id, { maxAge: 90000000, httpOnly: true });
            res.cookie('slug',user.slug, { maxAge: 90000000, httpOnly: true });
            if(user.role === "admin"){
                res.redirect("./indexAdmin")
            }
            if(user.role === "student"){
                res.redirect("./indexStudent")
            }
            if(user.role === "teacher"){
                res.redirect("./indexTeacher")
            }
            if(user.role === "guest"){
                res.redirect("./indexGuest")
            }
            if(user.role === "manager"){
                res.redirect("./indexManager")
            }
            }else{
                var message= "Username or password is invalid"
```

Figure 113: Authorize code

With the following logins, the server side will decode the token again to verify that it is them. To be able to decode the token, it is necessary to have the correct secret code generated in the server. After decoding the token, the server will check the role and direct the users to the home page depending on their role

```
let checkAuth = async (req,res,next)=>{
    try {
        var token = req.cookies.token || req.body.token
        let decodeAccount = jwt.verify(token,'minh')
        let user = await getUserById(decodeAccount._id)
        if(user){
            req.userLocal = user;
            next();
        }
    } catch (error) {
        res.status(500).json({
            message : "error sever",
            status: 500,
            error : true
        },
        res.redirect('/'))
    }
}
```

Figure 114: check Auth code

```
router.get('/indexAdmin',checkAuth ,checkAdmin, indexAdmin)
router.get('/indexTeacher',checkAuth ,checkTeacher, indexTeacher)
router.get('/indexStudent',checkAuth ,checkStudent, indexStudent)
router.get('/indexGuest',checkAuth,checkGuest,indexGuest)
router.get('/indexManager',checkAuth,checkManager,indexManager)
```

Figure 115: Home account

CRUD for Admin

Since the students and Marketing Coordinators are in faculties, I will create the faculties first. Here we add a slug, which can be understood as the faculty id that we define with each other

```
docreate(req,res){
    var facultyname = req.body.facultyname

    var newFaculity = FacultyModel({
        facultyname : facultyname,
        slug: req.body.slug,
    })
    newFaculity.save(function(err){
        if(err){
            console.log(err)
        }else{
            var newDashboard = DashboardtModel({
                slug:req.body.slug
            })
            newDashboard.save(function(err){
                res.redirect('/faculity/allfaculity')
            })
        }
    })
}
```

Figure 116: Create faculty code

The function to add students is password encrypted when creating and has the role of "student". Admin can select students to go to faculties through the select box

```
doAddStudent(req,res){
        let username = req.body.username;
        let password = req.body.password;
        let email = req.body.email;
        let slug = req.body.slug;

        const salt = bcrypt.genSaltSync(saltRounds);
        const hash = bcrypt.hashSync(password, salt);
        let newStudent = AccountModel({
            username,
            password :hash,
            email,
            slug
        })
        newStudent.save(function(err,data){
        if(err){
            console.log(err)
        }else{
            res.render('./student/faculity_student')
        }
        })
}
```

Figure 117: Add student code

```
  <select name="slug">
    {{#each faculity}}
    <option value="{{slug}}" name="slug">{{slug}}</option>
    {{/each}}
  </select>
```

Figure 118:Add student code

For the update and delete functions, we will get the _id of the object we want to manipulate and execute as needed. Like students, other users like guest or manager have the same thing. The difference here is that the roles of each person, guest and manager are not in faculty

Upload file and images

```
var storage = multer.diskStorage({
    destination:function(req,file,cb){
        cb(null,'./public/uploads')
    },
    filename:function(req,file,cb){
        var namefile = file.originalname
        cb(null,file.originalname)
    }
})
var upload = multer({storage:storage})
```

Figure 119: 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 the test passes, the system will convert the file to PPF and start setting the file name and location (both docx and PDF) and save that information to

the database.

- Case upload 1 file: Only docx is acceptable

```
x = req.files[0].originalname
if(req.files.length == 1){
  if(x.endsWith('docx')){
    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);
      }
    });
    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)
      }
```

Figure 120:Case upload 1 file code

- 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 2 files have the same properties, the system will send a notification. Otherwise, the system will convert the file to PPF and start setting the name, file location (both docx and PDF), image address and save that information in the database.

```
y = req.files[1].originalname

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

Figure 121:Case upload 2 file code

```
        docxConverter(x1,filePath1,function(err,result){
          if(err){
            console.log(err);
          }
        });
        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)
          }
```

Figure 122:Case upload 2 file code

View the selected reports.

```
readcontribution(req,res){
            let id = req.params.id
            fileModel.find({_id:id},(err,data)=>{
                if(err){
                    console.log(err)
                }
                else if(data.length>0){
                    res.render('guest/danhgia.ejs',{data:data})
                }
                else{
                    res.render('guest/danhgia.ejs',{data:data})
                }
            })
}
```

Figure 123: View the selected reports code.

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.

```
// set up mail sever
var transporter =  nodemailer.createTransport({
    host: 'smtp.gmail.com',
    port: 465,
    secure: true,
    auth: {
        user: 'nguyenminhsonhandsome@gmail.com',
        pass: 'minhson123a'
    },
    tls: {
        rejectUnauthorized: false
    }
    });
```

Figure 124: Set mail sever

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.

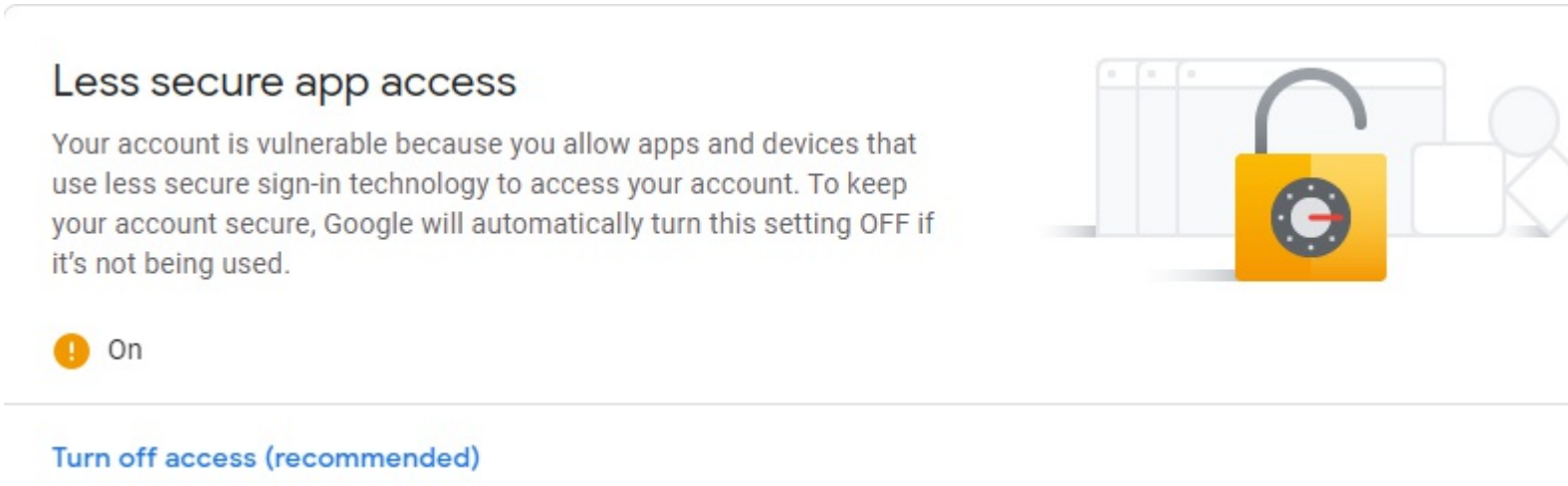

Figure 125: Turn on less secure app access

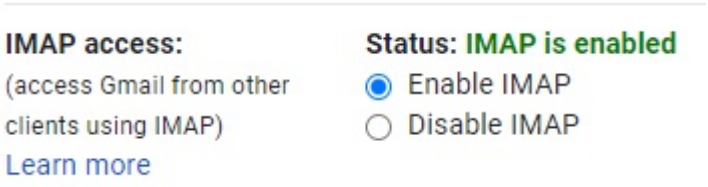

Figure 126:Enable IMAP

• Send email to student

```
let email = req.cookies.email
var content = 'Bạn vừa upload 1 bài báo lên hệ thống. Name: ' + x;
var mainOptions = {
    from: 'NQH-Test nodemailer',
    to: email,
    subject: 'Test Nodemailer',
    text: content
}
transporter.sendMail(mainOptions, function(err, info){
    if (err) {
        console.log(err);
    }
});
```

Figure 127: Send email to student code

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

```
let slug = req.cookies.slug
AccountModel.findOne({
    role: "teacher",
    slug: slug
},function(err, result){
    var content = email + 'vừa upload 1 bài báo lên hệ thống. Name: ' + x;
    var mainOptions2 = {
    from: 'NQH-Test nodemailer',
    to: result.email,
    subject: 'bài đăng mới',
    text: content
}
    transporter.sendMail(mainOptions2, function(err, info){
        if (err) {
            console.log(err);
        }
});
})
```

Figure 128: Send email to Marketing coordinator code

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

```
danhgiabaibao(req,res){
    let id = req.params.id
    fileModel.find({_id:id},(err,data)=>{
    if(err){
        console.log(err)
    }
    else if(data.length>0){
        res.render('faculity/danhgia.ejs',{data:data})
    }
    else{
        res.render('faculity/danhgia.ejs',{data:data})
    }
    })
}
```

Figure 129: Make a comment code

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.

```
dodanhgiabaibao(req,res){
    let id = req.params.id
    let status = req.body.status
    let comment = req.body.comment
    fileModel.findById({_id :id},function(err,data){
        let studentemail = data.studentemail
        console.log(studentemail)
    fileModel.updateOne(
        { _id: id },   // Query parameter
        {                    // Replacement document
            status: status,
            comment: comment
        })
        .then(()=>{
            res.redirect('/faculity/allDocument/' + studentemail)
        })
    })
}

danhgiabaibao2nd(req,res){
```

Figure 130: Make a comment code

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.

```
<div id="myModal" class="modal">
  <!-- Modal content -->
  <div class="modal-content">
    <span class="close">&times;</span>
    <iframe src="../uploads/dieukhoan.txt" class="dk"></iframe>
    <br>
    <p>Do you agree with the above terms?</p>
    <input type="radio" id="male" name="gender" value="Yes"> <label for="male">Yes</label>
    <input type="radio" id="female" name="gender" value="No"> <label for="female">No</label><br><br>
    <input type="submit" id="btn1" value="Next">
```

Figure 131: Agree to Terms and Conditions before student submit code

```
document.getElementById("btn1").onclick = function ()
{
  let ts = Date.now();
  let date_ob = new Date(ts);
  let date = date_ob.getDate().toString().padStart(2, "0");;
  let month = (date_ob.getMonth() + 1).toString().padStart(2, "0");
  let hour = date_ob.getHours().toString().padStart(2, "0");;
  let minutes = date_ob.getMinutes().toString().padStart(2, "0");;
  let year = date_ob.getFullYear();
  dl = year + "-" + month + "-" + date + " " + hour + ":" + minutes;
  var checkbox = document.getElementsByName("gender");
    for (var i = 0; i < checkbox.length; i++){
      if (checkbox[i].checked === true){
        if(checkbox[i].value ==='No'){
          alert("You must agree to continue")
        }else{
          var dealine = document.getElementById("del1").innerHTML ;
          if(dl < dealine){
            location.href = "/file"
            }else{
              alert("Out of date to upload file")
            }
        }
      }
    }
};
```

Figure 132: Agree to Terms and Conditions before student submit code

Set time for upload file managine

```
settime(req, res, next) {
    let date = req.body.date;
    let time = req.body.time;
    let deadline1 = date + " " + time;
    var someDate = new Date(date);
    someDate.setDate(someDate.getDate() + 14);
    var dateFormated = someDate.toISOString().substr(0, 10);
    let deadline2 = dateFormated + " " + time;
    FacultyModel.updateMany({}, { deadline: deadline1, deadline2: deadline2 }, function (err, data) {
        if (err) {
            res.json("")
        }
        FacultyModel.find({}, function (err, data) {
            if (err) {
                res.json("")
            }
            res.jsonp({success : true})
})
    })

}
```

Figure 133: Set time for upload file managine code

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:

```
var file_system = require('fs');
var archiver = require('archiver');
fileRouter.post('/abc',(req,res)=>{
    var output = file_system.createWriteStream('public/nameslug.zip');
    var archive = archiver('zip');
    var a = req.body.hobby
    output.on('close', function () {
        console.log(archive.pointer() + ' total bytes');
        console.log('archiver has been finalized and the output file descriptor has closed.');
    });
    archive.on('error', function(err){
        throw err;
    });
    archive.pipe(output);
        for(var n = 1; n <a.length; n++ ){
            file = "public/" +  a[n]
            console.log("file name là:", file)
            archive.append(file_system.createReadStream(file), { name: file })
        }
    archive.finalize();
        res.redirect('./abc1')
})

fileRouter.get('/abc1',(req,res)=>{
            var x = __dirname.replace('routes','public/') + 'nameslug.zip'
            res.download(x)
        }
)
```

Figure 134: Download article as zip code

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.

Chat box

The chat function will be in the details of the marketing coordinator or student depending on the user's role. If you are a student, there is only 1 coordinator, but if it is the coordinator, you can choose from a list of students.

If the coordinator and the student do not have any chat box before, the interface will have a button to add chat. When you click this button, the server side will create a collection of "chat" including the receiver and the sender. All conversation content will be saved in this collection. After adding chat, the next time, the user does not need to add a chat box anymore but can click the message button.

Server side will query in database according to sender and receiver email. New messages recorded by the user will be pushed into the collection

```
socket.on("new_user_message", (data) => {
    socket.join(`${data.cookiesemail}and${data.user}`);
    socket.join(`${data.user}and${data.cookiesemail}`);
    var query1 = {
        userSend: data.cookiesemail,
        userReceive: data.user,
    }
    var query2 = {
        userSend: data.user,
        userReceive: data.cookiesemail,
    }
```

Figure 135: Chat box code

```
let dbo = db.db("test");
dbo.collection("chats").updateOne(query1, {
    "$push": {
        message: {
            check: 1,
            Mes: `${data.mes}`,
            index_time:new Date().valueOf(),
            date: `${new Date().getHours()}:${new Date().getMinutes()}-${new Date().getDate()}/

        }
    }
});
dbo.collection("chats").updateOne(query2, {
    "$push": {
        message: {
            check: 0,
            Mes: `${data.mes}`,
            index_time:new Date().valueOf(),
            date: `${new Date().getHours()}:${new Date().getMinutes()}-${new Date().getDate()}/

        }
    }
});
```

Figure 136: Chat box code

Testing:

In order to make sure the project working properly, we need to make sure every function of it work as it should be. For that purpose, just like almost every other project, we decided to create a session which focus on testing the project. In this session, we will create a plan to test the service and after that we will show the test result in this report.

Test plan

Function requirement

Front-end: Login, Download submission by ZIP format, Chat box, Statistic, report and submission, give and read comment on submission, accept or reject submission, see submit deadline and edit deadline.

Back-end: Statistic management, notification email auto sending, exceptional report managing, comment management, chat box generator, account creator, setting deadline and maintaining system, password encode generator.

Targeted users

Our system is created for a university magazine so our user is 99 percent university students and school's staffs.

Environment requirement chuyển lên đàu test

- Server test: Heroku.
- Device: laptop
- Browser: Chrome, CocCoc, Microsoft Edge, Mozilla Firefox.
- Test plan: Microsoft word
- Test case: Microsoft word
- Image capture tool: Snipping tool
- Set up test report: Microsoft Word

- Tool Support Test Interface: Chrome

Testing implementation plan (Test strategy)

- Strategy for building test case
- The test case must follow the interface of manuscript functions and all function of Front-end and Back-end must be included in the test case.
- Defining risk by risks then solve the problem that may occur
- Testing tool
- Integration test: Testing using Google Chrome and Microsoft Edge on PC and laptop, Safari for phone
- System test: Test on PC and mobile phone.
- Project testing recourses
- Nguyen Trung Doan Khang: Responsible for Implementation of project test
- Test order
- System test: 25/03/2021
- Administrator features test: 29/03/2021
- Marketing manager features test: 30/03/2021
- Marketing coordinator features test: 30/03/2021
- Student features test: 30-31/03/2021
- Guest features test: 31/03/2021

Potential risks

Risk	Solution
Inexperienced tester	This is one of the most potential risk, the solution is easy, we can hire a well recommended tester or let them train our staff for a greater course.
The infrastructure may not fulfill requirement to test	Continuously checking on the machines, do guarantee work on time.
Lacking interactions between tester, developer and users	This may lead the project to a dead-end because of misunderstood, we should create a daily or weekly meeting to get more information between each other
Everyone may get difficult in job because of time strict	We can get more people to directly or indirectly help us with the job. Another solution is managing everyone work every day so we can make sure that they finish the job on time or sooner than we need
Test case may not include every error and may occur when we did not expect it	The only thing we can do is test every case that we can think of if their an error we may fix that in the future

Test case

Administrator

Test case	Input	Expected result	Actual result	Test condition	Test log
1 Create account for Marketing Manager	Create a marketing manager account with email: manager1@gmail.com And password: 123	Create successfully	Create successful and go to “All manager” page	Pass	17:16-02/03/2021 Test on URL: https://localhost:3000/
1.2 Create account for Marketing Coordinator	Create a marketing Coordinator account with email: teacher1@gmail.com And password: 123	Create successfully	Create successful and go to “All Teacher” page	Pass	17:20-02/03/2021 Test on URL: https://localhost:3000/
	Create a Student account				

1.3 Create account for Student	with email: student1@gmail.com And password: 123	Create successfully	Create successful and go to “All Student” page	Pass	17:33-02/03/2021 Test on URL: https://localhost:3000/
1.4 Create account for Guest	Create a Guest account with email: guest1@gmail.com And password: 123	Create successfully	Create successful and go to “All Guest” page	Pass	17:36-02/03/2021 Test on URL: https://localhost:3000/
1.5 Create an account which contain a number or special characters	Create a Guest account with user name: guest1+	Create fail	Create successful and go to “All Guest” page	Fail	17:37-02/03/2021 User name can still contain number and special character Test on URL: https://localhost:3000/
1.6 Create an account	Create a Marketing Manager account with email: Manager2	Create fail	Create successful and go to “All manager” page	Fail	17:37-02/03/2021 Email can be create without @gmail.com Test on URL: https://localhost:3000/
1.7 Create an account	Create a Guest account with password: +++	Create fail	Create successful and go to “All guest” page	Fail	17:37-02/03/2021 Password can contain special characters. Test on URL: https://localhost:3000/
2 Update account for Marketing Manager	Click on Update and update information	Update successfully	Update successful and go to “All manager” page	Pass	17:37-02/03/2021 Test on URL: https://localhost:3000/
2.1 Update account for Marketing Coordinator	Click on Update and update information	Update successfully	Update successful and go to “All Teacher” page	Pass	17:40-02/03/2021 Test on URL: https://localhost:3000/
2.2 Update account for Student	Click on Update and update information	Update successfully	Update successful and go to “All Student” page	Pass	17:42-02/03/2021 Test on URL: https://localhost:3000/
2.3 Update account for Guest	Click on Update and update information	Update successfully	Update successful and go to “All Guest” page	Pass	17:45-02/03/2021 Test on URL: https://localhost:3000/
3.1 Delete account for Marketing Manager	Click on Delete	Delete successfully	Delete successful and go to “All manager” page	Pass	17:47-02/03/2021 Test on URL: https://localhost:3000/
3.2 Delete account for Marketing Coordinator	Click on Delete	Delete successfully	Delete successful and go to “All Teacher” page	Pass	17:52-02/03/2021 Test on URL: https://localhost:3000/

3.3 Delete account for Student	Click on Delete	Delete successfully	Delete successful and go to “All Student” page	Pass	17:58-02/03/2021 Test on URL: https://localhost:3000/
3.4 Delete account for Guest	Click on Delete	Delete successfully	Delete successful and go to “All Guest” page	Pass	18:00-02/03/2021 Test on URL: https://localhost:3000/
4.1 Create a new Faculty	Create a new faculty with Faculty Name: Python	Create successfully	Create successful and go to “Faculty” page	Pass	18:10-02/03/2021 Test on URL: https://localhost:3000/
4.2 Update a faculty	Click on detail then update on “Faculty page”	Update successfully	Update successful and go to “Faculty” page	Pass	18:12-02/03/2021 Test on URL: https://localhost:3000/
4.3 Delete a faculty	Click on detail then Delete on “Faculty page”	Delete successfully	Delete successful and go to “Faculty” page	Pass	18:13-02/03/2021 Test on URL: https://localhost:3000/
5 Search	Typing Java on search bar	Search success and only Java faculty appear	Only JAVA faculty appear on the screen	Pass	18:13-02/03/2021 Test on URL: https://localhost:3000/
6.1 Login	Login as administrator with email: admin@gmail.com Password: 123	Login successfully.	Login successful and Home page is admin’s Personal information	Pass	17:05-25/03/2021 Test on URL: https://localhost:3000/
6.2 Login	Login as administrator with email: admin@gmail.com Password: 1	Cannot login	Cannot login and inform Username or password is invalid	Pass	17:08-25/02/2021 Wrong password Test on URL: https://localhost:3000/

1. Marketing Manager

Test case	Input	Expected result	Actual result	Test condition	Test log
1.1 Login	Login as Marketing manager with email: manager1@gmail.com Password: 123	Login successfully.	Login successful and Home page is manager’s Personal information	Pass	13:05-25/02/2021 Test on URL: https://localhost:3000/
1.2 Login	Login as Marketing Manager with email: admin@gmail.com Password: 1	Cannot login	Cannot login and inform Username or password is invalid	Pass	17:08-25/02/2021 Wrong password Test on URL: https://localhost:3000/
2 View Statistic	Click on View contribution then view statistical	View successful	When click on, pie chart of that faculty showed up	Pass	13:08-20/03/2021 Test on URL:

					https://localhost:3000/ 13:10-
3 View all submission by faculty	Click on View contribution then view contributions again on a faculty	View successful	All contribution of that faculty showed up	Pass	14/03/2021 Test on URL: https://localhost:3000/ 13:12-
4 Download submission into ZIP	Click on download contribution	Download successful	Contribution was downloaded in ZIP format	Pass	17/03/2021 Test on URL: https://localhost:3000/ 13:13-
5 Set deadline for student	Click on “set time”	Time settled	Deadline was all set for student	Pass	20/03/2021

1. Marketing Coordinator

Test case	Input	Expected result	Actual result	Test condition	Test log
1.1 Login	Login as Marketing manager with email: manager1@gmail.com Password: 123	Login successfully.	Login successful and Home page is manager’s Personal information	Pass	13:05- 25/02/2021 Test on URL: https://localhost:3000/
1.2 Login	Login as Marketing Manager with email: admin@gmail.com Password: 1	Cannot login	Cannot login and inform Username or password is invalid	Pass	17:08-25/02/2021 Wrong password Test on URL: https://localhost:3000/
2 View Statistic	Click on View contribution then view statistical	View successful	When click on, pie chart of that faculty showed up	Pass	13:08- 20/03/2021 Test on URL: https://localhost:3000/
3 View all submission by faculty	Click on View contribution then view contributions again on a faculty	View successful	All contribution of that faculty showed up	Pass	13:10- 14/03/2021 Test on URL: https://localhost:3000/
4 Download submission into ZIP	Click on download contribution	Download successful	Contribution was downloaded in ZIP format	Pass	13:12- 17/03/2021 Test on URL: https://localhost:3000/
5 Set deadline for student	Click on “set time”	Time settled	Deadline was all set for student	Pass	13:13- 20/03/2021

Test case	Input	Expected result	Actual result	Test condition	Test log

1.1 Login	Login as Marketing manager with email: manager1@gmail.com Password: 123	Login successfully.	Login successful and Home page is manager's Personal information	Pass	13:05- 25/02/2021 Test on URL: https://localhost:3000/

1. Student

Test case	Input	Expected result	Actual result	Test condition	Test log
1.1 Login	Login as Student with email: student1@gmail.com Password: 123	Login successfully.	Login successful and Home page is student's Personal information	Pass	14:20- 25/02/2021 Test on URL: https://localhost:3000/
1.2 Login	Login as Student with email: admin@gmail.com Password: 1	Cannot login	Cannot login and inform Username or password is invalid	Pass	14:22-25/02/2021 Wrong password Test on URL: https://localhost:3000/
2.1 submit article as word or pdf	Click on Upload file, accept term then choose file to upload	File uploaded	File uploaded and returned to Summited file	Pass	14:30-07/03/2021 Test on URL: https://localhost:3000/
2.2 submit article as images/photos	Click on Upload file, accept term then choose file to upload	File uploaded	File uploaded and returned to Summited file	Pass	14:30-08/03/2021 Test on URL: https://localhost:3000/
2.3 Edit submission	Click on Update file	File updated	File updated then returned to Summited file	Pass	14:36-29/03/2021 Test on URL: https://localhost:3000/
3 Read comment for own submission	Click on File submitted	View comment of submitted file	Comment showed on Submitted file page	Pass	15:18-29/03/2021 Test on URL: https://localhost:3000/

1. Guest

Test case	Input	Expected result	Actual result	Test condition	Test log
1.1 Login	Login as Guest with email: guest1@gmail.com Password: 123	Login successfully.	Login successful and Home page is guest's Personal information	Pass	15:50- 25/03/2021 Test on URL: https://localhost:3000/
1.2 Login	Login as guest with email: admin@gmail.com Password: 1	Cannot login	Cannot login and inform Username or password is invalid	Pass	15:56-25/02/2021 Wrong password Test on URL: https://localhost:3000/

2 View accepted submission by faculty	Click on view button of each faculty	All accepted submission of that faculty showed	All accepted submission of that faculty showed	Pass	16:00-13/03/2021 Test on URL: https://localhost:3000/

Thiếu ảnh chứng cứ test

Agile:

Product backlog

Story ID	As a/an	Story	prio
1	guest	I want to login my Account so that I can do my job	must
2		I want to view the selected reports.	must
3		I want to the interface must be suitable for all devices	could
4	student	I want to login my Account so that I can do my job	must
5		I want to submit one or more articles as Word documents to the magazine.	must
6		I want to upload high quality images,e.g. photographs.	must
7		I have to agree to Terms and Conditions before they can submit.	should
8		I want to the interface must be suitable for all devices	could
9	Marketing Coordinator	I want to login my Account so that I can do my job	must
10		I want to receive email notification after student posting articles to the system	could
11		I have to make a comment within 14 days.	should
12		I want to interact with students in their Faculty to edit and to select those for publication.	Want t
13		I want to the interface must be suitable for all devices	could
14	Marketing Manager	I want to login my Account so that I can do my job	must
15		I want to view all the selected contributions but cannot edit any	should
16		I want to download all the selected contributions after the final closure date in a ZIP file.	must
17		I want to set time for upload file managine	should
18		I want to statistical analysis number of contributions per Faculty	could
19		I want to the interface must be suitable for all devices	could
20	administrator	I want to login my Account so that I can do my job	must
21		I want to update Terms and Conditions before student posting.	could
22		I want to the interface must be suitable for all devices	could

Figure 137:Product backlog

Sprints and burndown chart

Sprint 1:

Task name	volunteer	estimate time (hours)	Day 1 (22/02)	Day 2	Day 3	Day 4	Day 5
Create database for account	Sơn, minh	5	3	2	0		
Code front-end for login page	Tuấn	5	3	1	0		
Code backend for login and authorization functionality	Minh	8	6	4	2	0	
Test the login function and authorize the accounts (Guest,student, Marketing Coordinator,Marketing manager,admin).	hướng khang	4	4	3	2	0	
Code interface for functions: add, edit, delete accounts(Guest,student, Marketing Coordinator,Marketing manager,admin).	Tuấn	10	10	10	10	8	
Code backend for functions: add, edit, delete faculty and accounts(Guest,student, Marketing Coordinator,Marketing manager,admin).	Sơn, minh	13	13	13	13	13	11
Test functions: add, edit, delete faculty and accounts(Guest,student, Marketing Coordinator,Marketing manager,admin).	hướng khang	6	6	6	6	6	6
	Daily progress	51	45	39	33	27	2
	Ideal progress	51	45,3	39,7	34,0	28,3	22,

Figure 138: Sprint 1

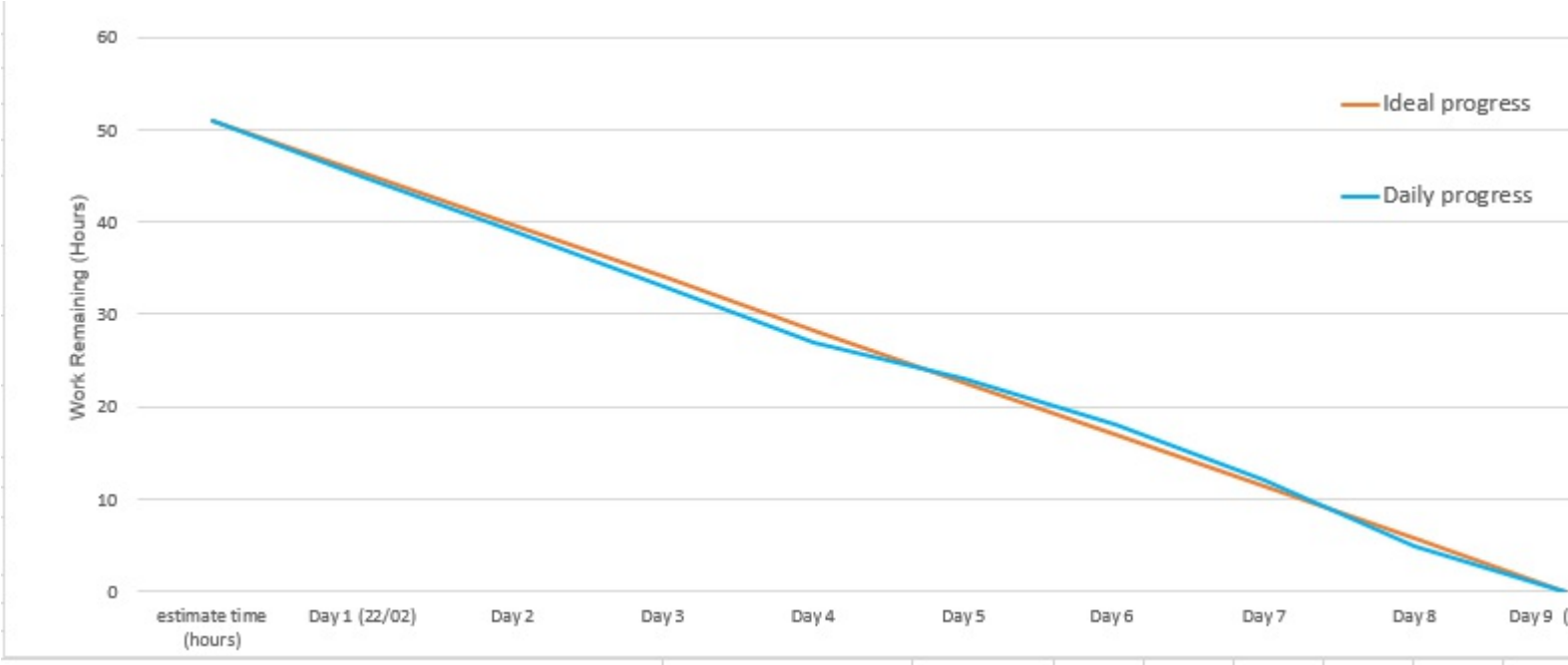

Figure 139: Burndown chart Sprint 1

The work schedule of spirnt 1 is completed according to the set plan. Since Son and Minh are familiar with the technology, the implementation and construction of the functions took place as planned.

Sprint 2:

Task name	volunteer	estimate time (hours)	Day 1 (03/03)	Day 2	Day 3	Day 4
code interface sends notice to students before posting.	Tuấn	3	1	0		
Code to check if the student agrees to the term or not.	Sơn	3	2	1	0	
Create database for upload file and image	Sơn,minh	6	4	2	0	
Code front-end for upload file and images	Tuấn	3	3	2	1	0
Code back-end for upload file	Sơn,minh	5	5	4	3	1
Test upload file (docx)	hướng khang	3	3	3	3	2
Code back-end for upload images	Sơn,minh	5	5	5	3	1
Test upload image	hướng khang	3	3	3	3	2
Code back-end for the Marketing Coordinator receives email notifications every time a student posts on the system.	Sơn	4	4	4	4	4
Test Marketing Coordinator receives email notifications every time a student posts on the system.	hướng khang	2	2	2	2	2
Code frontend for the Marketing Coordinator reviews the student's articles	Tuấn	4	4	4	4	4
Code backend for the Marketing Coordinator reviews the student's articles	Sơn	6	6	6	6	6
Test function: Marketing Coordinator reviews the student's articles	hướng khang	4	4	4	4	4
	Daily progress	51	46	40	33	26
	Ideal progress	51	45,3	39,7	34,0	28,3

Figure 140: Sprint 2

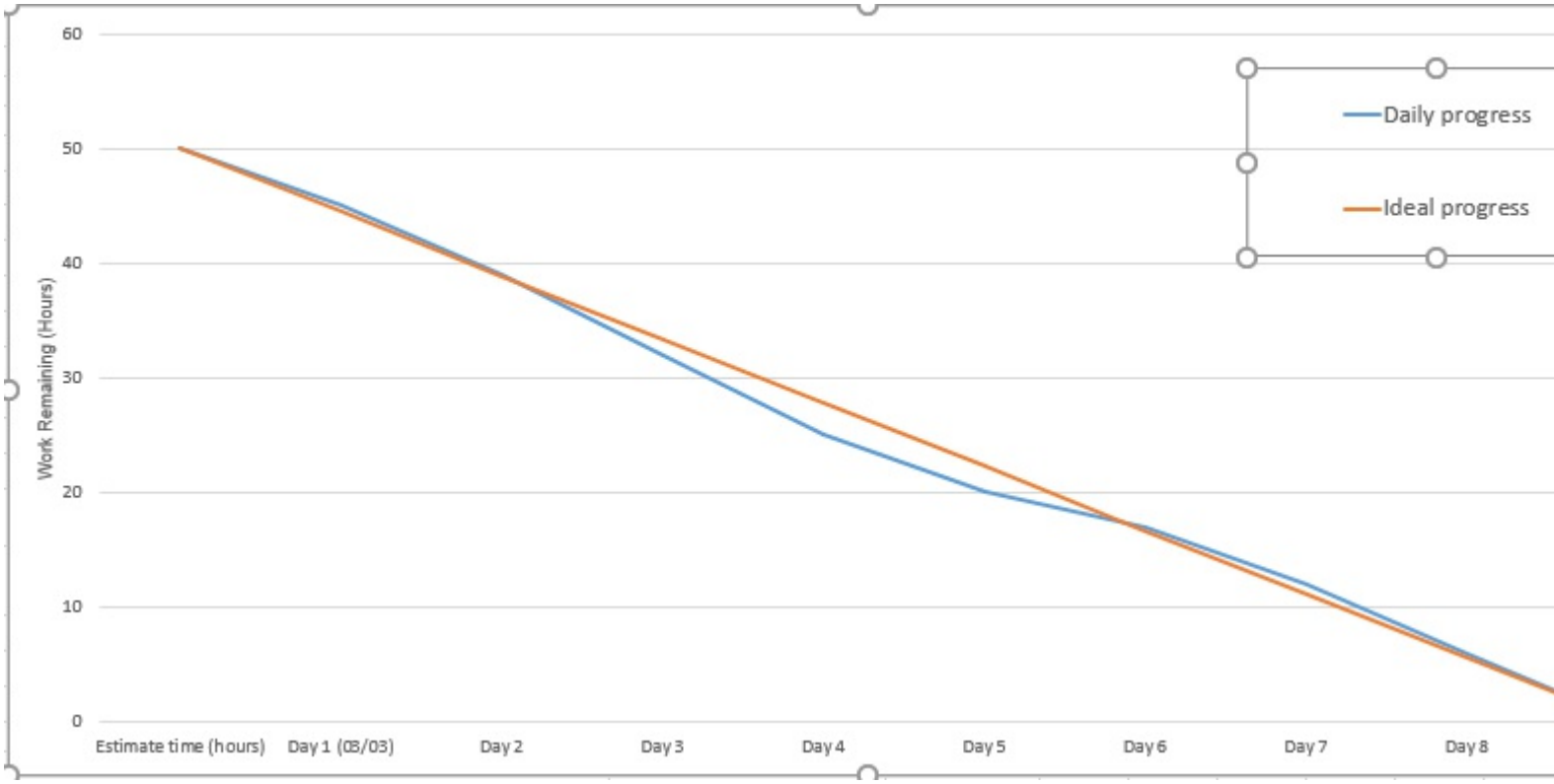

Figure 141: Burndown chart Sprint 2

In sprint 2, we also got the job done on time and was a bit over-schedule in the midst of spirnt. In the middle of sprint 2, due to the work being similar in nature and functionality, the implementation took place faster than intended. (upload file, upload image, student and marketing coordinator receive mail).

Sprint 3:

Task name	volunteer	estimate time (hours)	Day 1 (12/03)	Day 2	D
code interface for customers to read the selected articles	Tuấn	3	1	0	
code back-end for customers to read the selected articles	Sơn	4	1	0	
Test customers read the selected articles	hướng khang	2	2	0	
code interface for manager to read the selected articles	Tuấn	3	3	2	
code back-end for manager to read the selected articles	Sơn	4	4	3	
Test manager read the selected articles	hướng khang	2	2	2	
Code interface for the manager to choose which posts to download	Tuấn	4	4	4	
Code back-end for the manager to choose which posts to download	Sơn	6	6	6	
Test download selected posts	hướng khang	4	4	4	
Create database for dashboard	Sơn,minh	3	3	3	
Code back-end to statistic data and save to database (dashboard)	Sơn,minh	6	6	6	
Code interface for displaying statistical data (pie chart).	Sơn	4	4	4	
Test statistic data and displaying	hướng khang	3	3	3	
Code interface for the manager to set deadline for submission	Tuấn	2	2	2	
Code back-end for the manager to set deadline for submission	Sơn	3	3	3	
Test set time	hướng khang	1	1	1	
	Daily progress	54	49	43	
	Ideal progress	54	48	42	

Figure 142: Sprint 3

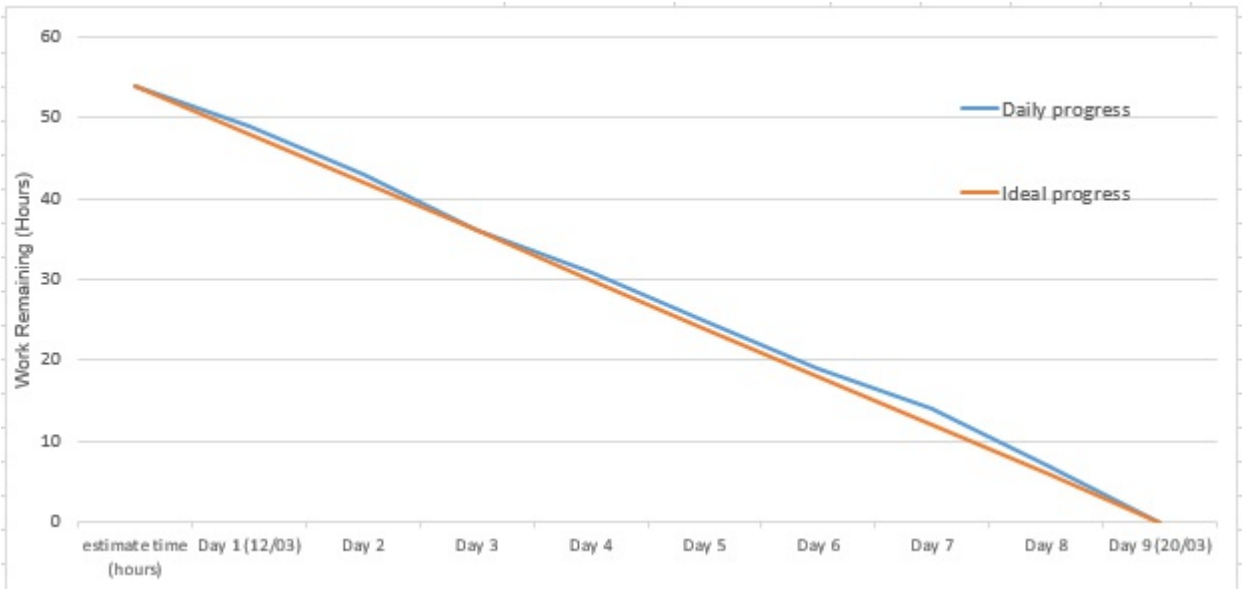

Figure 143: Burndown chart Sprint 3

In sprint 3, functionality is assessed to a moderate level (3-6). Therefore, the construction implementation is quite accurate time estimate. In addition, the statistical viewing function is quite new, but also quickly and completely explored. Therefore, the work was also completed as planned.

Sprint 4:

Task name	volunteer	estimate time (hours)	Day 1 (21/03)	Day 2	Day 3	Day 4	Day 5	Day 6	D
Code validate data	Sơn	2	0						
Create database for chat box	Minh	8	5	2	0				
Code interface for chat box	Minh, Tuấn	8	8	6	4	2	0		
Code back-end for chat box	Minh	8	8	8	10	6	3	0	
Deplay code to sever	Sơn	3	3	3	3	3	3	0	
Test Role Admin in sever	Khang, Hướng	3	3	3	3	3	3	3	
Test Role Marketing Manager in sever	Khang, Hướng	3	3	3	3	3	3	3	
Test Role Marketing Coordinator in sever	Khang, Hướng	3	3	3	3	3	3	3	
Test Role student in sever	Khang, Hướng	3	3	3	3	3	3	3	
Test Role Guest in sever	Khang, Hướng	3	3	3	3	3	3	3	
Test chat box in sever	Khang, Hướng	3	3	3	3	3	3	3	
	Daily progress	47	42	37	35	29	24	18	
	Ideal progress	47	41,8	36,6	31,3	26,1	20,9	15,7	

Figure 144: Sprint 4

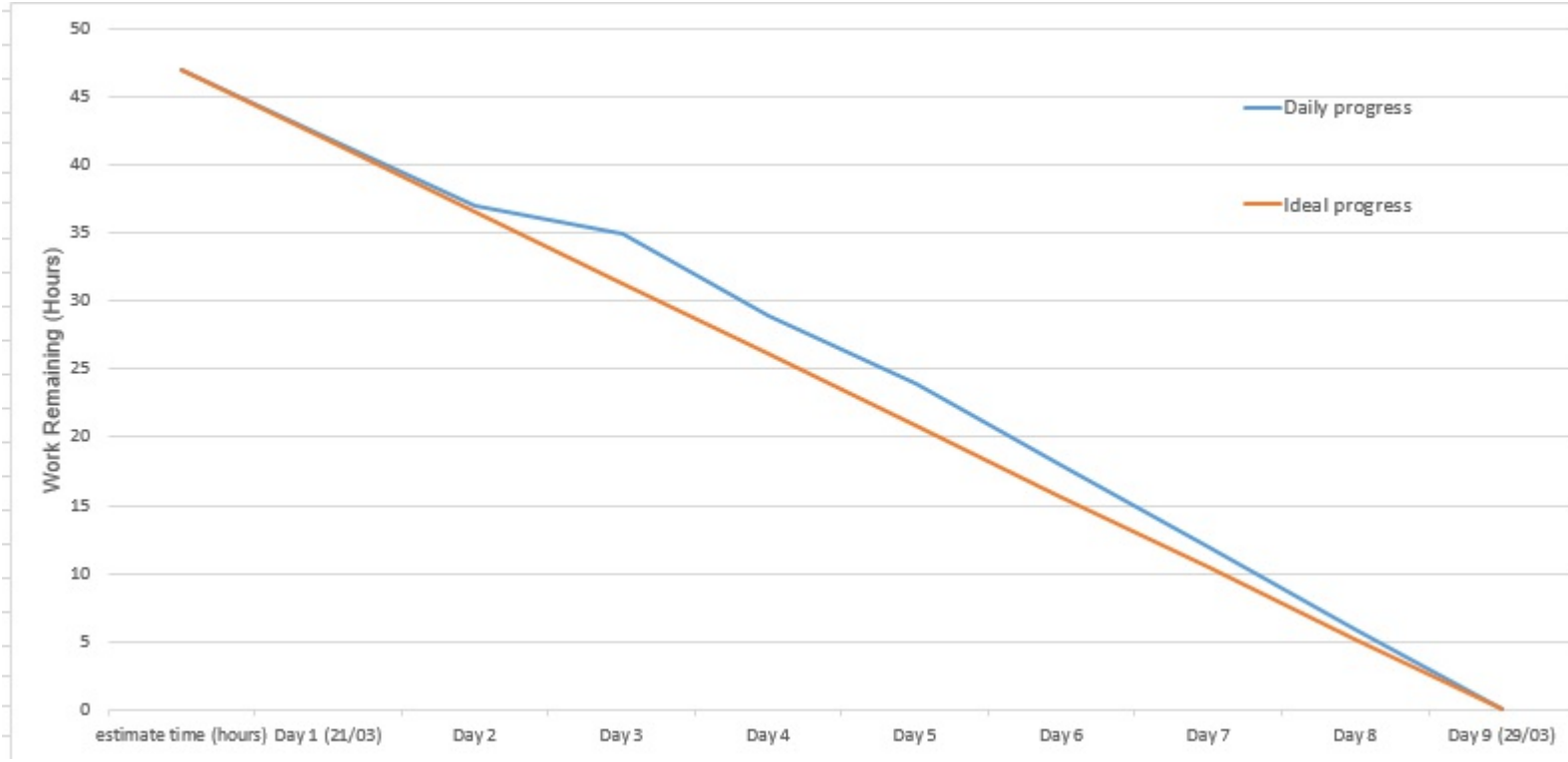

Figure 145: Burndown chart Sprint 4

At spirnt 4, the implementation of building chat functions for students and the marketing coordinator took longer than expected. We have evaluated this functionality as difficult compared to the power of the coder (8/10 points) and the implementation of this function has slowed the project forward. In addition, we have to validate the account information left by spirnt 1.When we finished the chat function, we quickly started pushing the project and testing it on the server and there was no risk incurred. The project has also ended as scheduled.

Meeting minutes

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:

Back-end	Nodejs
Front-end	Mongo
Database	Html-Css
Sever	Heroku

Figure 146:Technical

Daily Meeting:

During each sprint, the team will meet on the 4th and last day of spirnt to discuss the progress of the work and to plan for the next Sprint.

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

The first timeline of sprint 1 is 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.
- Hướng,Khang: The decentralization test has been completed and there is no risk.

Next work to be completed when Sprint 1 ends (March 02, 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 of sprint 1 is on March 02, 2021 (the end of Sprint 1):

The work has been completed

- 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.

Remaining work: The account information validation will be move to 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 of sprint 2 is 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 of sprint 2 is on March 11, 2021 (the end of Sprint 2)

The work has been completed

- 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.

Remaining work: None. The sprint 2 work has ended as planned.

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

In Sprint 2, the student functions and Marketing coordinator have been completed. Following sprint 3, the job will be to build the Client and Marketing Manager functions.

The first timeline on March 15, 2021

In the first 4 days of sprint 3, the team has completed 2 functions of the Customer and Marketing Manager, which is the function of reading articles that has been evaluated and approved by Marketing coordinator.

- Tuan: Create an interface for reading articles for Guest and Marketing Manager.
- Sơn : Write back-end code to perform the process of getting post information and displaying the post on the front-end. And learn how to perform the function of downloading articles as zip files of Marketing Manager.
- Hướng, Khang: Do the test and there is no problem during the test.
- Minh: Researching charts to perform statistical functions for Marketing Manager.

Remaining work:

- Function of marketing manager: Set deadline for upload file, download selected article as zip, view statistic chart.

The second timeline on March 20, 2021 (the end of Sprint 3)

The work has been completed

- Tuan: Design the interface for Marketing Manager to choose the articles to download and the interface for Marketing Manager to set a deadline for the student's article loading.
- Sơn, Minh: Created database for statistical graphs and wrote back-end code to statistic data from database.
- Sơn: The back-end code for the selection process, uploading students' articles to Marketing Manager and setting deadlines for students to submit. Create an interface to display statistics graphs.
- Hướng, Khang: Test the above functions and no errors occur.

Remaining work: None. The sprint 3 work has ended as planned.

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

After completing the functions of for Guest and Marketing Manager in Sprint 3. At Sprint 4, we will proceed to handle the problems in the previous sprints and push the project to the server to progress to test the project. run on server.

The first timeline on March 24, 2021

During the first 4 days of spirnt 4, we completed the following tasks:

- Sơn: Execute code completion for account information validation (work needs to be done from sprint 1)
- Minh: Designing database for student chat and Marketing Coordinator and researching back-end code for chat. Since the chat is considered to be the most difficult part, the database design takes a lot of time

Remaining work: Complete the student chat and Marketing Coordinator and test the project running on heroku.

The second timeline on March 28, 2021

- Sơn: Pushes the project onto heroku. During the push of the code, there were some errors such as: Node version not specified in package.json, Node_modules checked into source control and promptly processed.
- Minh, Tuan: Research and design interfaces for student chat functions and Marketing Coordinator.
- Minh: Doing the back-end coding for the chat.
- Khang, Hướng: Test the project running on heroku and the result runs as if running localhost and did not detect any new errors.

Conclusion