VEDUCAT

Virtual Education Platform

TEAM MEMBERS

1.TRAN THIEN - ITITIU21318

2.UNG HUYNH PHUC - ITITIU21284

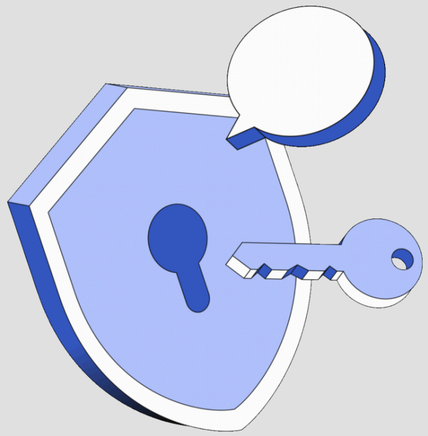
3.LE THIEN NGAN - ITCSIU21091

TASKS

UNG HUYNH PHUC	UI/UX DESIGN & FRONTEND DEVELOPMENT
TRAN THIEN	SYSTEM DEVELOPMENT & BACKEND DEVELOPMENT
LE THIEN NGAN	BACKEND DEVELOPMENT & TESTING

TABLE OF CONTENTS

AUTHN & AUTHZ



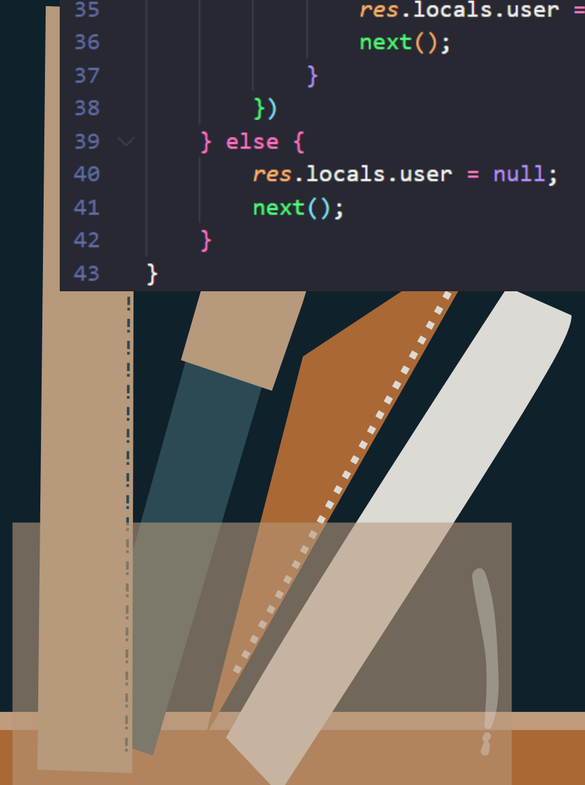
LOGIN

```
module.exports.login_post = async (req, res) => {
    const { userID, password } = req.body;

    try {
        const user = await User.login(userID, password )
        const token = createToken(user._id);
        res.cookie('jwt', token, { httpOnly: true, maxAge: maxAge * 1000 });
        res.status(200).json({ user: user._id})
    } catch (error) {
        const errors = handleErrors(error)
        res.status(400).json({ errors })
    }
}
```

```
31  userSchema.pre('save', async function (next) {
32      const salt = await bcrypt.genSalt();
33      this.password = await bcrypt.hash(this.password, salt);
34      next();
35  })
36
37  // static methods to login user
38  userSchema.statics.login = async function (userID, password) {
39      const user = await this.findOne({ userID });
40      if (user) {
41          const isMatch = await bcrypt.compare(password, user.password);
42          if (isMatch) {
43              return user;
44          }
45          throw new Error('incorrect password');
46      }
47      throw new Error('incorrect user ID');
48  }
```

AUTHN & AUTHZ

```
// check curent user (để duy trì đăng nhập)
const checkUser = (req, res, next) => {
    const token = req.cookies.jwt;

    // check json web token exists & is verified
    if (token) {
        jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, async (err, decodedToken) => {
            if (err) {
                console.log(err.message);
                res.locals.user = null;
                next();
            } else {
                let user = await User.findById(decodedToken.id)
                res.locals.user = user;
                next();
            }
        })
    } else {
        res.locals.user = null;
        next();
    }
}
```



```
const requireAuthZ = (permission) => async (req, res, next) => {
    const token = req.cookies.jwt;

    if (token) {
        jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, async (err, decodedToken) => {
            if (err) {
                console.log('ERROR: 1', err.message);
                return res.redirect('/login');
            } else {
                let user = res.locals.user;
                if (!user || !user.role) {
                    console.log('ERROR: 2');
                    res.cookie('jwt', '', { maxAge: 1 });
                    return res.redirect('/login');
                }
                if (!permission.includes(user.role)) {
                    console.log('ERROR: 3');
                    console.log(user.role);
                    console.log(permission);
                    res.cookie('jwt', '', { maxAge: 1 });
                    return res.redirect('/login');
                }
                next();
            }
        });
    } else {
        console.log('ERROR: 4');
        return res.redirect('/login');
    }
};
```

ROUTER

```
function route(app) {
    // routes
    app.get('*', checkUser)
    app.use('/student', student)
    app.use('/teacher', teacher)
    app.use('/admin', admin)
    app.use('/', siteRouter)
}
```

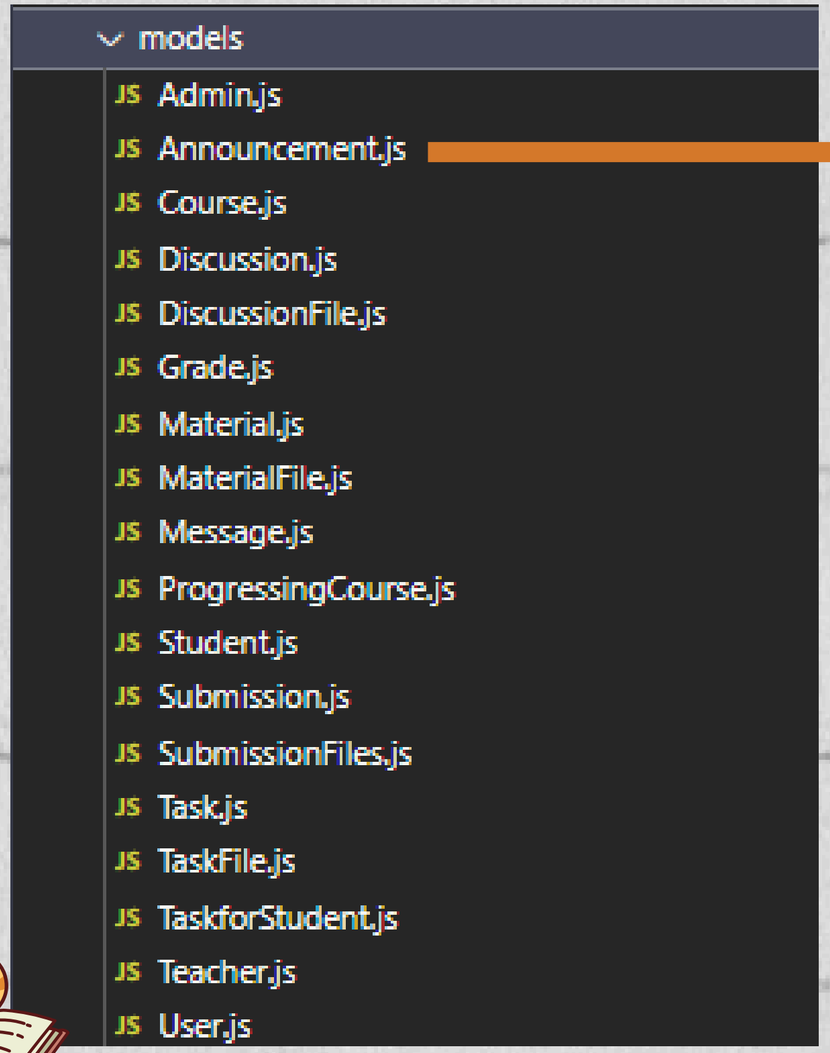
```
// Redirect user to the corresponding page
router.get('/', (req, res) => {
    const user = res.locals.user;
    if (!user) {
        return res.render('login');
    }
    const { role } = user;
    switch (role) {
        case 'student':
            res.redirect('/student');
            break;
        case 'teacher':
            res.redirect('/teacher');
            break;
        case 'admin':
            res.redirect('/admin');
            break;
        default:
            res.json({ message: 'No permission' });
            break;
    }
});
```

MVC MODEL

MODEL

mongo DB

MODEL



```js
const mongoose = require('mongoose');

const announcementSchema = new mongoose.Schema({
    pCourseID: {
        type: 'string',
        lowercase: true,
    },
    title: {
        type: 'string',
        required: [true, 'Please enter a title']
    },
    content: {
        type: 'string',
        required: [true, 'Please enter a content']
    },
    receivers: {
        type: [String],
        default: [],
        required: [true, 'Please enter a receivers ID']
    },
    senderID: {
        type: 'string',
        required: [true, 'Please enter a sender ID'],
        lowercase: true
    },
    type: {
        type: 'string',
        lowercase: true
    },
    createdAt: {
        type: Date,
        default: Date.now
    },
});

const Announcement = mongoose.model('announcement', announcementSchema);

module.exports = Announcement;
```

CONTROLLER

```
router.get('/course/:_id/announcement', requireAuthZ('student'), studentController.announcement_get);
```

```
// GET /student/course/:_id/announcement
async announcement_get(req, res, next) {
    const user = res.locals.user;
    const { _id } = req.params;
    try {
        const student = await Student.findOne({ studentID: user.userID });
        if (!student) {
            return res.status(404).json({ message: "Student not found" });
        }

        // Current course
        let pCourse = await ProgressingCourse.findById(_id);
        const course = await Course.findOne({ courseID: pCourse.courseID });
        pCourse = { ...pCourse._doc, courseName: course.name, courseDescription: course.description };

        const announcements = await Announcement.find({ pCourseID: pCourse._id, receivers: student.studentID });

        res.render('student/course_announcement', { user, student, pCourse, announcements});
    } catch (error) {
        console.error(error);
        res.status(500).send({ message: "An error occurred", error });
    }
}
```

VIEW

```
// Announcements
let announcements = '<%- JSON.stringify(announcements).replace(/\\/g, "\\\\") %>';
announcements = JSON.parse(announcements);
// SORT ANNOUNCEMENTS BY DATE: newest to oldest
announcements.sort((a, b) => new Date(b.createdAt).getTime() - new Date(a.createdAt).getTime());

const announcementList = document.querySelector('.announcement-listed');
const announcementUl = announcementList.querySelector('ul');

announcements.forEach(announcement => {
    const li = document.createElement('li');
    const div = document.createElement('div');
    div.classList.add('announcement-container');
    const a = document.createElement('a');
    a.href = '#';
    a.textContent = `${announcement.title}`;
    const description = document.createElement('div');
    description.classList.add('description');
    const span = document.createElement('span');
    span.textContent = `${announcement.content}`;
    description.appendChild(span);
    div.appendChild(a);
    div.appendChild(description);
    li.appendChild(div);
    announcementUl.appendChild(li);
});
if (announcements.length === 0) {
    const li = document.createElement('li');
    li.classList.add('empty-content');
    li.textContent = 'No announcements yet';
    announcementUl.appendChild(li);
}
```

VIEW

Announcement

Hello students
good morning

Hi Everyone
VEDUCAT - Virtual Education Platform

WEB SOCKET

WEB SOCKET

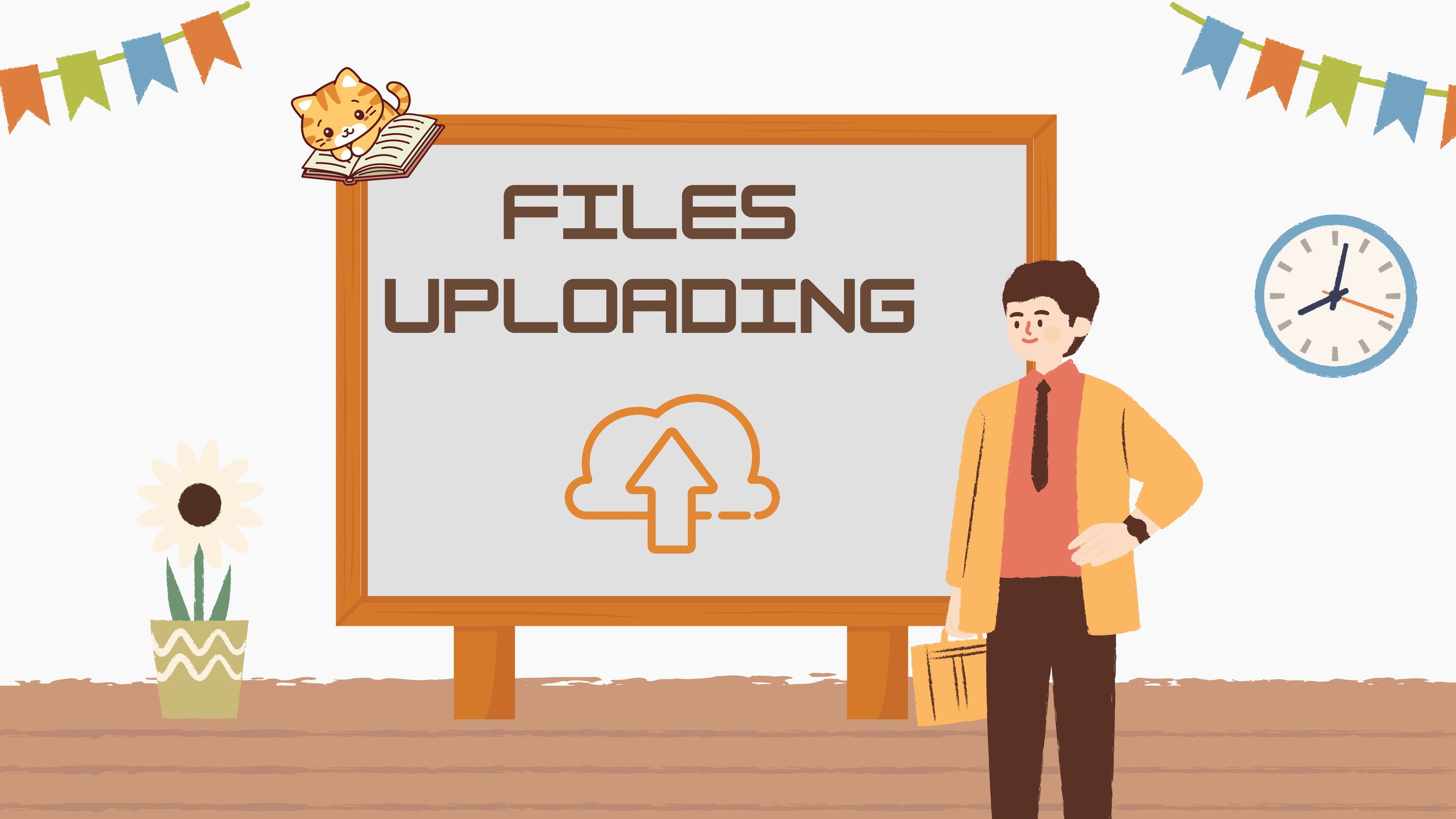
```
// Initialize database connection
mongoose.connect(process.env.MONGO_DB)
  .then((result) => {
    console.log('Connected to MongoDB: http://localhost:3000');
    const server = app.listen(process.env.PORT)

    const io = require('socket.io')(server)

        var usp = io.of('/user')
        usp.on('connection', onConnected)
    function onConnected(socket) {
      console.log('Socket connected')

      socket.on('disconnect', () => {
        console.log('Socket disconnected')
      })

      socket.on('message', (data) => {
        socket.broadcast.emit('chat-message', data)
      })
    }

  })
  .catch((err) => console.log(err));
```

WEB SOCKET

```
            if(res.status >= 200 && res.status < 300) {
                socket.emit('message', data);
                const div = document.createElement('div');
                formChat.querySelector('textarea').value = '';
                div.classList.add('owner-msg');
                div.innerHTML = `<div class="user-name">
                        <i class="fas fa-user-circle"></i>
                        <span>${data.senderID}</span>
                        </div>
                        <div class="msg">
                            <p>${data.message}</p>
                        </div>
                        <div class="time">
                            <span>${moment(data.createAt).fromNow()}</span>
                        </div>`;
                document.querySelector('.discussion-board').appendChild(div);
                document.querySelector('#discussion-board').scrollTop = document.querySelector('#discussion-board').scrollHeight;
            }

        }
```

WEB SOCKET

```
socket.on('chat-message', (message) => {
    othersMessage(message);
});

function othersMessage(message) {
    const div = document.createElement('div');
    div.classList.add('other-msg');
    div.innerHTML = `<div class="user-name">
                        <i class="fas fa-user-circle"></i>
                        <span>${message.senderID}</span>
                    </div>
                    <div class="msg">
                        <p>${message.message}</p>
                    </div>
                    <div class="time">
                        <span>${moment(message.createAt).fromNow()}</span>
                    </div>`;
    document.querySelector('.discussion-board').appendChild(div);
    document.querySelector('#discussion-board').scrollTop = document.querySelector('#discussion-board').scrollHeight;
}
```


FILES
UPLOADING

FILES UPLOADING

MULTER

```
//Multer configuration
const storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, 'storage/teacher');
    },
    filename: function (req, file, cb) {
        const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
        cb(null, `${uniqueSuffix}-${file.originalname}`);
    }
});

const upload = multer({
    storage: storage,
    limits: { fileSize: 1024* 1024 * 1024 * 10 },
    fileFilter: (req, file, cb) => {
        const allowedTypes = ['application/pdf', 'image/png', 'image/jpeg', 'application/zip', 'application/x-rar-compressed'];
        if (allowedTypes.includes(file.mimetype)) {
            cb(null, true);
        } else {
            cb(new Error('Invalid file type'), false);
        }
    }
})
const cpUpload = upload.fields([{ name: 'file', maxCount: 10 }])
```

FILES UPLOADING

MULTER

```
    // Get the form container and form
    const formContainer = document.querySelector('.form-container');
    const form = formContainer.querySelector('form');

    form.addEventListener('submit', async (e) => {
        e.preventDefault();

        const title = form.querySelector('input[name="title"]').value;
        const description = form.querySelector('textarea').value;
        const files = form.querySelector('input[type="file"]').files;
        const courseID = pCourse._id;
        const deadline = form.querySelector('input[type="datetime-local"]').value;
        const formData = new FormData(e.target);

        formData.append('courseID', courseID);
        formData.append('description', description);
        try {
            const res = await fetch(`/teacher/course/${pCourse._id}/homework`, {
                method: 'POST',
                body: formData,
            });
            const data = await res.json();
            if(res.status >= 200 && res.status < 300) {
                location.reload();
            }
        } catch (error) {
            console.log(error);
        }
    });
</script>
```

FILES UPLOADING

MULTER

```
router.post('/course/:_id/homework', cpUpload, requireAuthZ('teacher'), teacherController.homework_post);
```

```
async homework_post (req, res, next) {
    const user = res.locals.user;
    const { _id } = req.params;
    const { title, description,deadline} = req.body;
    const files = req.files;

    upload(req, res, async (err) => {
```

```
            const uploadFiles = files.file.map(file => ({
                taskID: homework._id,
                taskName: title,
                fileName: file.originalname,
                filePath: file.path,
                fileType: file.mimetype,
                fileSize: file.size,
            }));
            await TaskFile.create(uploadFiles);
        }
```

REFERENCES

1. [F8 OFFICIAL]. (2020, JULY 20). NODEJS & EXPRESSJS [VIDEO]. YOUTUBE.
2. [NET NINJA]. (2020, AUGUST 12). NODE.JS AUTH TUTORIAL (JWT) [VIDEO].
3. [PROGRAMMING EXPERIENCE]. (2023, MARCH 8). HOW TO CHAT WITH USERS USING SOCKET.IO IN DYNAMIC CHAT APP - DYNAMIC CHAT APP IN NODE JS MONGO #10 [VIDEO]. YOUTUBE.
4. TRULYMITTAL (N.D.). CHAT-SOCKET.IO. GITHUB.
5. [RANDOM CODER]. (2020, AUGUST 30). SEND AUTOMATED EMAILS WITH NODE.JS! (WITH NODEMAILER) [VIDEO]. YOUTUBE.

DISPLAY DEMO


THANK YOU!
Give us your confusion
well done!