Node.Js

Authentication & Authorization

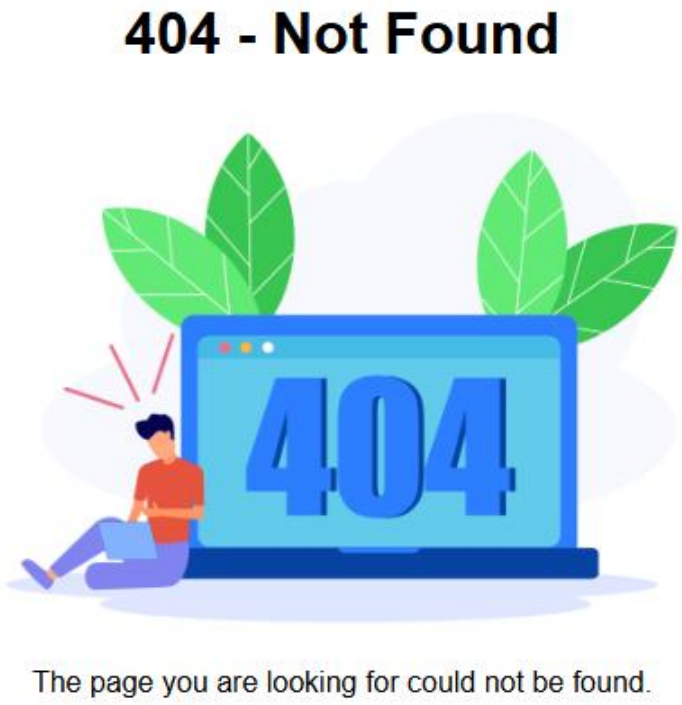
19/01/2025

תיקיית קבצים סטטיים

תיקיית קבצים סטטיים

בחלק זה נלמד מה היא תיקיית קבצים סטטיים וכיצד נוכל להשתמש בה.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה הם "**קבצים סטטיים**"?
- מדוע שנזדקק לתיקיית קבצים סטטיים?
- איזה סוג של קבצים נוכל לשים בתיקייה זו?
- כיצד נוכל לצפות בקובץ שנמצא בתיקייה זו על ידי שימוש בדפדפן?
- כיצד נוכל לצפות בקובץ שנמצא בתיקייה זו על ידי שימוש ב-**postman**?
- כיצד נוכל להחזיר קובץ **html** מעוצב במצבים שבהם משתמש פונה לכתובת שגוייה בשרת?
- מה זה "**process.cwd()**" ולשם מה נשתמש במתודה זו?




```
// Add a static files middleware
app.use(express.static('public'));
```

```
export const badPathHandler = (req, res) => {
    console.error(chalk.red('Path Not found'));
    res.status(404).sendFile('public/404.html', { root: process.cwd() });
}
```

תיקיית קבצים סטטיים - תרגול

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.

My API

Welcome to my API! The available endpoints are:

```
GET /users
GET /users/:id
POST /users
PUT /users/:id
DELETE /users/:id
```

For more information, please refer to the postman api documentation

תרגיל	תיאור המשימה
Ex-1	צרו קובץ **index.html** מעוצב בתיקיית הקבצים הסטטיים אשר מתאר את ה-**API** בקצרה ובדקו שניתן לגשת אליו דרך הדפדפן.
Ex-2	צרו קובץ **html** מעוצב למצב שגיאה (**404**) בתיקיית הקבצים הסטטיים.
Ex-3	טפלו ב-**middleware** שאחראי על טיפול בפניות לנתיבים שגויים כך שיחזיר את קובץ ה-**404** המעוצב שיצרתם.
Ex-4	בדקו שהקובץ שיצרתם מתקבל כצפוי באמצעות שימוש ב-**postman** ובדפדפן בעת פנייה לנתיב שגוי בשרת.

מחבר: גל לביא

File System (fs)

file system (or fs)

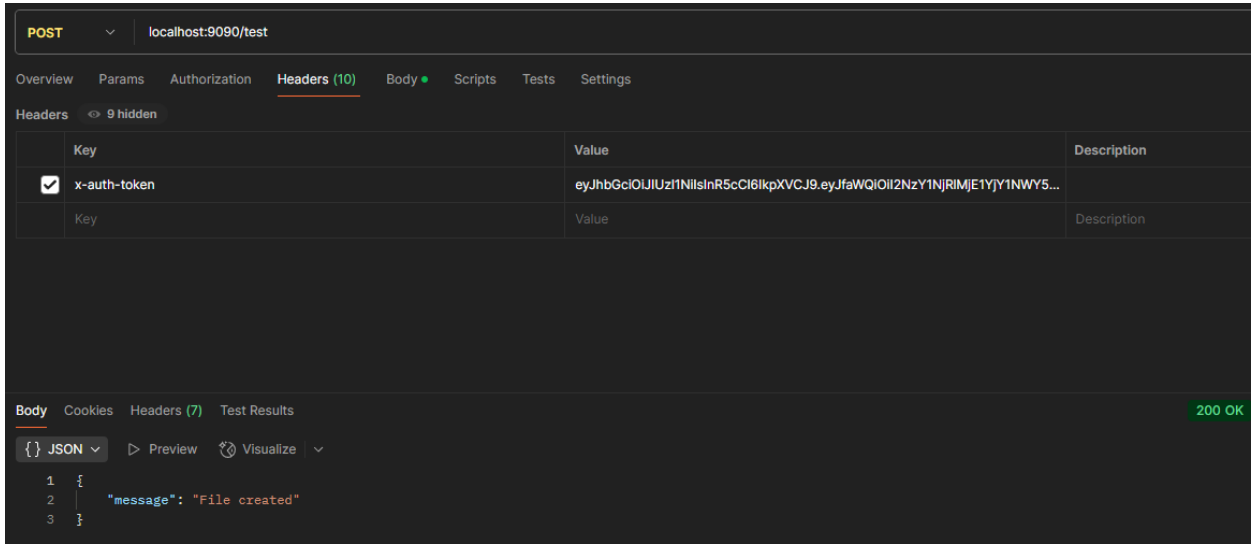
בחלק זה נכיר את **fs** ונלמד כיצד להשתמש בה על מנת לבצע מניפולציות בקבצים ותיקיות.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- לשם מה נשתמש במודול "**fs**"?
- לשם מה נשתמש במתודה "**join**" של המודול "**path**" ?
- כיצד נשתמש במתודה "**mkdir**" על מנת ליצור תיקייה במידה ואינה קיימת?
- כיצד נשתמש במתודה "**writeFile**" על מנת ליצור קובץ במידה ואינו קיים?
- מה משמעות ה-"**flags**" ואיזה קיימים?
- איזה עוד מתודות קיימות ב-"**fs**"?

```
router.post("/test", async (req, res) => {
    fs.mkdir(path.join(process.cwd(), 'test'), { recursive: true }, (err) => {
        if (err) {
            return res.status(500).json({ message: err.message });
        } else {
            fs.writeFile(path.join(process.cwd()) + '/test/test.txt', 'Hello World!', { flag: 'w' }, (err) => {
                if (err) {
                    return res.status(500).json({ message: err.message });
                } else {
                    return res.json({ message: 'File created' });
                }
            });
        }
    });
});
```

fs - תרגול

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.



תרגיל	תיאור המשימה
Ex-1	צרו נתיב **POST** בשם "**test**", הנתיב ידרוש **token** ב-**header**.
Ex-2	בעת קריאה לנתיב ייווצר קובץ בשם "**test.txt**" בתוך תיקייה בעלת שם לבחירתכם, הקובץ יכיל טקסט אשר מברך לשלום את המשתמש תוך שימוש בשמו.
Ex-3	בדקו את תפקוד הנתיב באמצעות שימוש ב-**postman**.

File Logger Using morgan & fs

מחבר: גל לביא

File Logger Using morgan & fs

בחלק זה נשפר את ה- **logger** שלנו ונוסיף לו את היכולת לרשום **log** לתוך קובץ טקסט.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מדוע שנרצה לשמור את הלוגים של השרת בקובץ?
- מה הוא "**stream**" וכיצד זה מסייע לנו?
- איזה שם כדאי שניתן לקובץ (התאריך של היום) ומדוע?
- מדוע שנזדקק לבצע לוקליזציה לתאריך ושעה של לוגים וכיצד נבצע זאת?
- האם ניתן להמשיך להדפיס את הלוגים לקונסול בנוסף לקובץ? כיצד המתודה "**next**" תסייע לנו לבצע זאת?

```
// Console format with colors
const consoleFormat = (tokens, req, res) => {
    const color = res.statusCode >= 400 ? chalk.red : chalk.green;

    return [
        chalk.cyan(getTodayDate()),
        chalk.cyan(getTodayTime()),
        color(tokens.method(req, res)),
        color(tokens.url(req, res)),
        color(tokens.status(req, res)),
        tokens['response-time'](req, res) + 'ms',
    ].join(' | ');
};

// File format without colors
const fileFormat = (tokens, req, res) => {
    return [
        getTodayDate(),
        getTodayTime(),
        tokens.method(req, res),
        tokens.url(req, res),
        tokens.status(req, res),
        tokens['response-time'](req, res) + 'ms',
    ].join(' | ');
};

// Create console logger
const consoleLogger = morgan(consoleFormat);

// Create file logger
const fileLogger = morgan(fileFormat, {
    stream: accessLogStream
});
```

```
//create logs directory if it doesn't exist
const logsDir = path.join(process.cwd(), 'logs');

if (!fs.existsSync(logsDir)) {
    fs.mkdirSync(logsDir);
}

//create a write stream (in append mode)
const accessLogStream = fs.createWriteStream(path.join(logsDir, `${getTodayDate()}.txt`), { flags: 'a' });
```

תרגול - File Logger Using morgan & fs

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.

תרגיל	תיאור המשימה
Ex-1	עדכנו את ה-logger שלכם כך שיכתוב את הלוגים לקובץ יומי.
Ex-2	שימו לב שאתם משתמשים בתאריך והשעה הלוקליים.
Ex-3	בצעו קריאות ל-API (מוצלחות ושגויות).
Ex-4	בדקו שהלוגים על הקריאות שביצעתם אכן מופיעים בקובץ הלוגים היומי.

```node_modules/morgan/index.js
// Get today's local date in the format of 'DD-MM-YYYY'
const getTodayDate = () => {
    const date = new Date();
    return date.toLocaleDateString('he-US', {
        year: 'numeric',
        month: '2-digit',
        day: '2-digit'
    }).replace(/\./g, '-');
};

// Get the current local time in the format of 'HH:MM:SS'
const getTodayTime = () => {
    const date = new Date();
    return date.toLocaleTimeString('he-US', {
        hour: '2-digit',
        minute: '2-digit',
        second: '2-digit'
    });
};
```

```node_modules/morgan/index.js
// Combined middleware that uses both loggers
export const morganLogger = (req, res, next) => {
    consoleLogger(req, res, (err) => {
        if (err) return next(err);
        fileLogger(req, res, next);
    });
};
```

```
18-01-2025 | 21:44:57 | GET | /users/676564e215b655f93493ec0d | 401 | 2.559ms
18-01-2025 | 22:12:46 | GET | /users/login | 401 | 0.988ms
18-01-2025 | 22:12:49 | GET | /users/login | 401 | 0.861ms
18-01-2025 | 22:13:16 | POST | /users/login | 200 | 105.222ms
18-01-2025 | 22:13:49 | POST | /users/676564e215b655f93493ec0d | 404 | 0.989ms
18-01-2025 | 22:13:53 | GET | /users/676564e215b655f93493ec0d | 403 | 1.056ms
18-01-2025 | 22:14:40 | GET | /users/676564e215b655f93493ec0d | 403 | 0.661ms
18-01-2025 | 22:15:37 | GET | /users/676564e215b655f93493ec0d | 403 | 2.624ms
18-01-2025 | 22:16:11 | GET | /users/676564e215b655f93493ec0e | 200 | 5.520ms
18-01-2025 | 22:16:41 | GET | /users/676564e215b655f93493ec0e | 200 | 2.363ms
```


מחבר: גל לביא

Get Log File by date

בחלק זה נלמד כיצד באפשרותנו לייבא **log** של תאריך מסוים על ידי שימוש בבקשת **API**.
בסיום הנושא תוכלו לענות על השאלות הבאות:

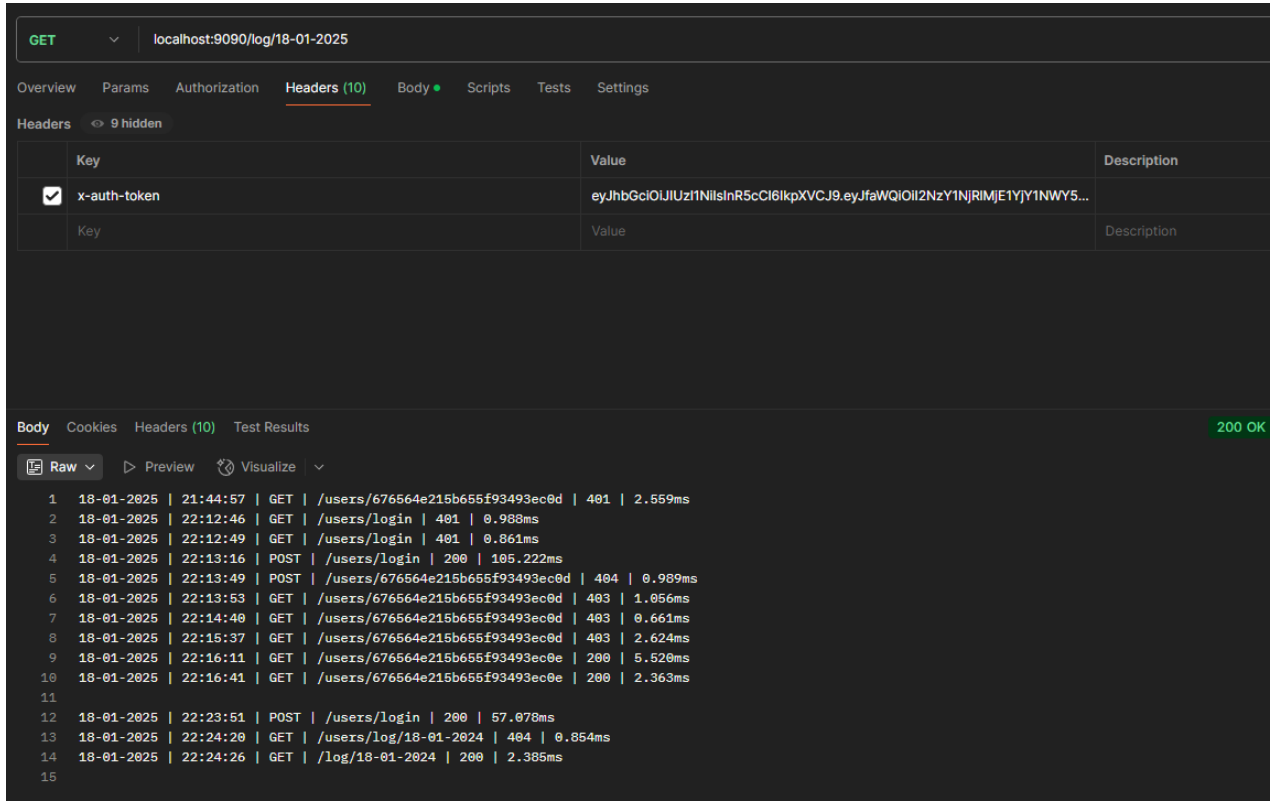
- באמצעות איזו מתודה נוכל להחזיר קובץ מה-**API**?
- מדוע שנרצה לקבל תאריך ב-**params**?
- מדוע נרצה שנתיב זה יהיה נגיש למשתמש מסוג "**Admin**" בלבד?

```
router.get("/log/:date", auth, checkAuthLevel(3), (req, res) => {
    try {
        const { date } = req.params;
        return res.sendFile(path.join(process.cwd(), 'logs', `${date}.txt`));
    } catch (error) {
        return res.status(500).json({ message: error.message });
    }
});
```

תרגול - Get Log File by date

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.

תרגיל	תיאור המשימה
Ex-1	צרו נתיב **GET** אשר נגיש למשתמש מסוג "**Admin**" בלבד עבור ייבוא קובץ לוג לפי תאריך, בקשו את התאריך ב-**params**.
Ex-2	ייבאו קובץ לוג יומי לפי תאריך ע"י שימוש ב-**postman**.



העלאת קבצים לשרת באמצעות שימוש ב-multer

העלאת קבצים לשרת באמצעות שימוש ב-multer

בחלק זה נלמד כיצד נוכל להעלות קבצים לשרת באמצעות שימוש ב-multer.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה היא "**multer**"?
- מתי נרצה לאפשר למשתמש להעלות קבצים לשרת?
- האם ניתן להגביל את גודל וסוגי הקבצים שניתן להעלות לשרת?
- לאיזה תיקייה נרצה לאפשר למשתמש להעלות קבצים?
- כיצד נשתמש ב-**id** של המשתמש לצורך נתינת שם לתיקייה?
- כיצד נוסיף מזהה ייחודי לשם הקובץ ומדוע שנרצה לעשות זאת?
- מה זה "**uuidv4**" וכיצד נוכל לייצר מזהה כזה בתוכנית **express**?
- כיצד נוכל לטפל במצב ששם הקובץ בעברית?
- כיצד נוכל להחזיר למשתמש את הכתובת שדרכה ניתן לגשת לקובץ שהועלה לשרת?

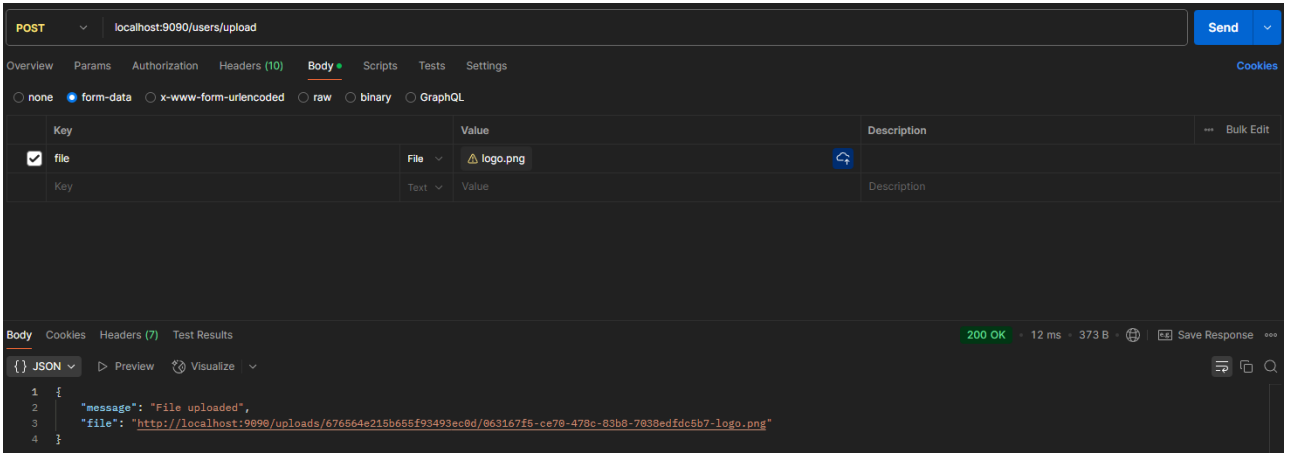
[expressjs/multer: Node.js middleware for handling `multipart/form-data`.]

```
const normalizeFileName = (fileName) => {
    return Buffer.from(fileName, 'latin1').toString('utf8');
};

const dest = multer.diskStorage({
    destination: (req, file, callback) => {
        const dir = `public/uploads/${req.user._id}`;
        fs.mkdirSync(dir, { recursive: true });
        callback(null, dir);
    },
    filename: (req, file, callback) => {
        const normalizedFilename = `${randomUUID()}-${normalizeFileName(file.originalname)}`;
        req.fileName = `http://localhost:9090/uploads/${req.user._id}/${normalizedFilename}`;
        callback(null, normalizedFilename);
    }
});

const upload = multer({
    storage: dest,
    limits: { fileSize: 1000000 }, // 1MB
    fileFilter: (req, file, callback) => {
        const normalizedFilename = normalizeFileName(file.originalname);

        if (!normalizedFilename.match(/\.(pdf|docx|doc|png|jpg|jpeg)$/)) {
            return callback(
                new Error("Please upload a PDF, Word document or an image")
            );
        }
        callback(null, true);
    }
}).single('file');
```

```
router.post("/upload", auth, upload, async (req, res) => {
    try {
        return res.json({ message: "File uploaded", file: req.fileName });
    } catch (error) {
        return res.status(500).json({ message: error.message });
    }
});
```

העלאת קבצים לשרת באמצעות שימוש ב-multer - תרגול

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.



תרגיל	תיאור המשימה
Ex-1	צרו נתיב **POST** אשר נגיש למשתמשים רשומים בלבד - עבור העלאת קבצים לשרת.
Ex-2	טפלו ב-**middleware** כך שיאפשר העלאה של קבצים קטנים מ-**5** מגה, מסוג תמונה, מסמך או **PDF** בלבד. שם הקובץ יהיה מורכב ממזהה ייחודי + שם הקובץ המקורי. הוסיפו את נתיב הקובץ בשרת ל-**request** במפתח בשם "**fileName**".
Ex-3	נתיב ה-**POST** שייצרתם יחזיר את כתובת הנתיב בשרת שאליו הועלה הקובץ. בדקו העלאת קבצים לשרת באמצעות שימוש ב-**postman**.
Ex-4	בדקו שהקובץ שיצרתם מתקבל כצפוי באמצעות שימוש בדפדפן.

תודה על ההקשבה

אני וצוות המכללה כאן עבורכם