BHAVYA ANAND (WEEK-3)

Project Progress Report

Project Title: File Management System
Reporting Period: *25th December,2023 to 29nd December,2023.*

Key Objectives:

Project Setup: Establish the project structure, dependencies, and initial configurations.

Backend Development: Implement basic backend functionality, including user authentication, database connection, and API endpoints.

Security Measures: Integrate security features such as password hashing using bcrypt and JWT token generation for authorization.

Utility Components: Create utility classes and components for handling asynchronous operations, API responses, and errors.

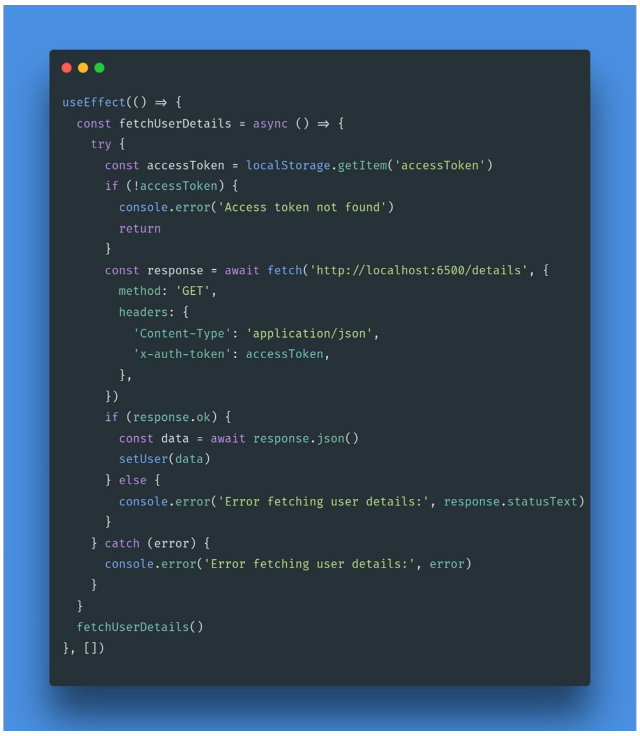
Middleware Integration: Add necessary middleware, such as CORS, for proper API endpoint access.

Accomplishments:

[Bhavya Anand]

Profile Page:

1. Developed a profile page to fetch and display details of the currently logged-in user.

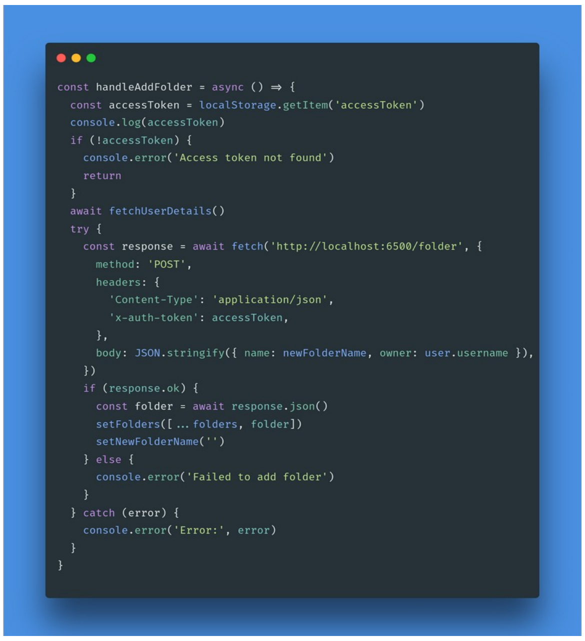


```
useEffect(() => {
  const fetchUserDetails = async () => {
    try {
      const accessToken = localStorage.getItem('accessToken')
      if (!accessToken) {
        console.error('Access token not found')
        return
      }
      const response = await fetch('http://localhost:6500/details', {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json',
          'x-auth-token': accessToken,
        },
      })
      if (response.ok) {
        const data = await response.json()
        setUser(data)
      } else {
        console.error('Error fetching user details:', response.statusText)
      }
    } catch (error) {
      console.error('Error fetching user details:', error)
    }
  }
  fetchUserDetails()
}, [])
```

BHAVYA ANAND (WEEK-3)

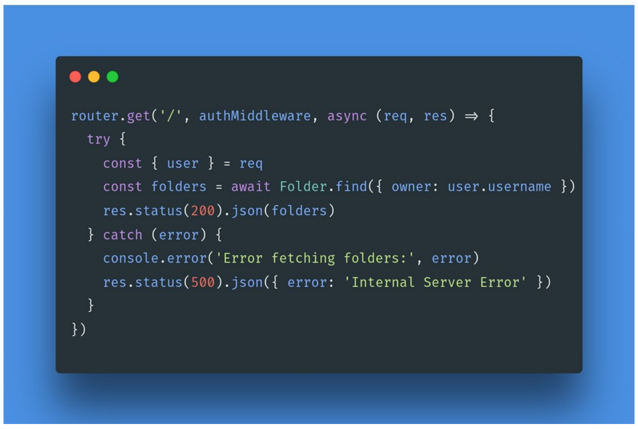
Dashboard and Folder Management:

1. Created a dashboard with the ability to add folders, implementing an API endpoint to save folder details to the database.



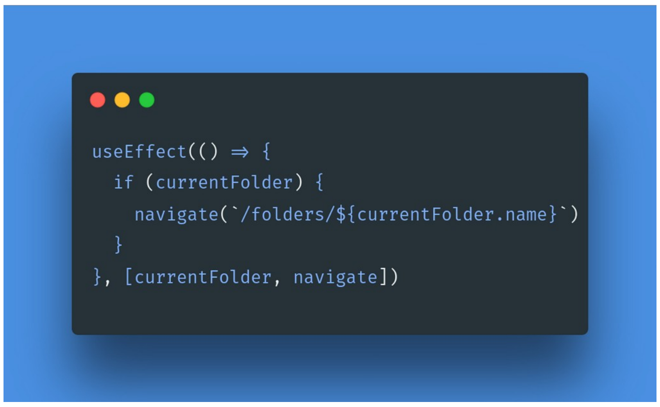
```
const handleAddFolder = async () ⇒ {
  const accessToken = localStorage.getItem('accessToken')
  console.log(accessToken)
  if (!accessToken) {
    console.error('Access token not found')
    return
  }
  await fetchUserDetails()
  try {
    const response = await fetch('http://localhost:6500/folder', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'x-auth-token': accessToken,
      },
      body: JSON.stringify({ name: newFolderName, owner: user.username }),
    })
    if (response.ok) {
      const folder = await response.json()
      setFolders([ ... folders, folder])
      setNewFolderName('')
    } else {
      console.error('Failed to add folder')
    }
  } catch (error) {
    console.error('Error:', error)
  }
}
```

2. Implemented an API to fetch folders created by the currently logged-in user and displayed them on the dashboard.



```
router.get('/', authMiddleware, async (req, res) ⇒ {
  try {
    const { user } = req
    const folders = await Folder.find({ owner: user.username })
    res.status(200).json(folders)
  } catch (error) {
    console.error('Error fetching folders:', error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```

3. Added functionality to navigate to a clicked folder and add files to the selected folder.

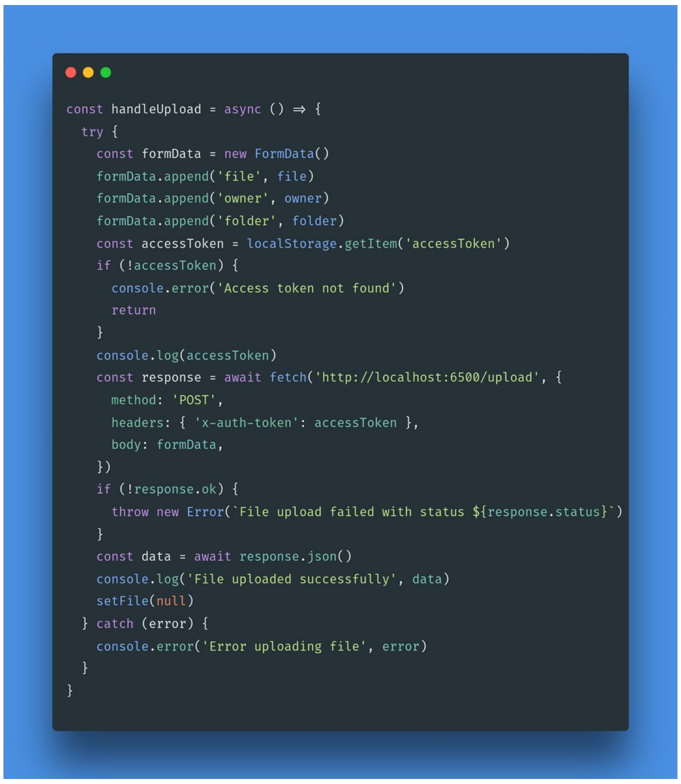


```
useEffect(() ⇒ {
  if (currentFolder) {
    navigate(`/folders/${currentFolder.name}`)
  }
}, [currentFolder, navigate])
```

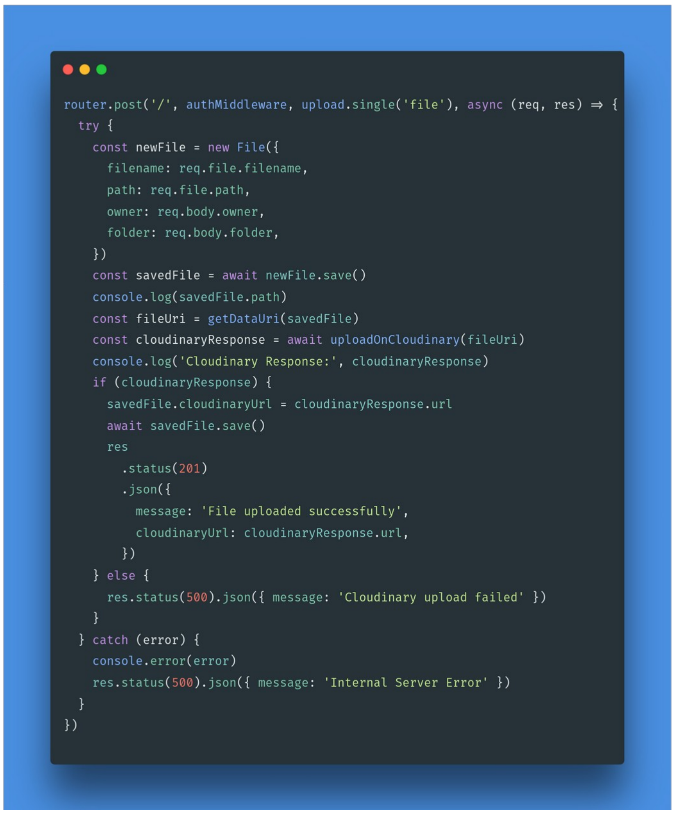
BHAVYA ANAND (WEEK-3)

File and Folder Creation:

1. Implemented API endpoint for adding file details (name, owner, creation date, and time) to the database using Multer.

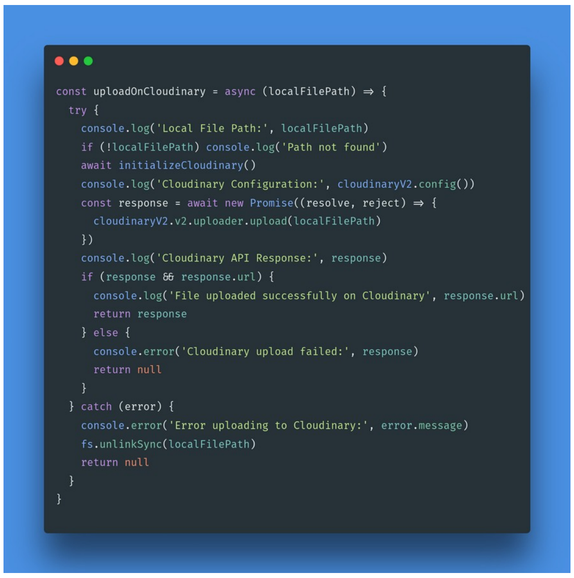


```
const handleUpload = async () => {
  try {
    const formData = new FormData()
    formData.append('file', file)
    formData.append('owner', owner)
    formData.append('folder', folder)
    const accessToken = localStorage.getItem('accessToken')
    if (!accessToken) {
      console.error('Access token not found')
      return
    }
    console.log(accessToken)
    const response = await fetch('http://localhost:6500/upload', {
      method: 'POST',
      headers: { 'x-auth-token': accessToken },
      body: formData,
    })
    if (!response.ok) {
      throw new Error(`File upload failed with status ${response.status}`)
    }
    const data = await response.json()
    console.log('File uploaded successfully', data)
    setFile(null)
  } catch (error) {
    console.error('Error uploading file', error)
  }
}
```



```
router.post('/', authMiddleware, upload.single('file'), async (req, res) => {
  try {
    const newFile = new File({
      filename: req.file.filename,
      path: req.file.path,
      owner: req.body.owner,
      folder: req.body.folder,
    })
    const savedFile = await newFile.save()
    console.log(savedFile.path)
    const fileUri = getDataUri(savedFile)
    const cloudinaryResponse = await uploadOnCloudinary(fileUri)
    console.log('Cloudinary Response:', cloudinaryResponse)
    if (cloudinaryResponse) {
      savedFile.cloudinaryUrl = cloudinaryResponse.url
      await savedFile.save()
      res
        .status(201)
        .json({
          message: 'File uploaded successfully',
          cloudinaryUrl: cloudinaryResponse.url,
        })
    } else {
      res.status(500).json({ message: 'Cloudinary upload failed' })
    }
  } catch (error) {
    console.error(error)
    res.status(500).json({ message: 'Internal Server Error' })
  }
})
```

2. Integrated Cloudinary services to save files, creating links for efficient file management.



```
const uploadOnCloudinary = async (localFilePath) => {
  try {
    console.log('Local File Path:', localFilePath)
    if (!localFilePath) console.log('Path not found')
    await initializeCloudinary()
    console.log('Cloudinary Configuration:', cloudinaryV2.config())
    const response = await new Promise((resolve, reject) => {
      cloudinaryV2.v2.uploader.upload(localFilePath)
    })
    console.log('Cloudinary API Response:', response)
    if (response && response.url) {
      console.log('File uploaded successfully on Cloudinary', response.url)
      return response
    } else {
      console.error('Cloudinary upload failed:', response)
      return null
    }
  } catch (error) {
    console.error('Error uploading to Cloudinary:', error.message)
    fs.unlinkSync(localFilePath)
    return null
  }
}
```

Upcoming Goals:

File Retrieval:

Develop API for fetching all files of the current folder for the currently logged-in user and display them on the dashboard.

Enhanced File Management:

Add functionality for searching and sorting files by name.

UI Enhancement:

Adding UI enhancements to make the experience more user-friendly.

Challenges:

Authentication Error During Folder Retrieval:

1. Encountered authentication errors while fetching folders for the currently logged-in user.
2. Successfully addressed the issue by incorporating authentication into a separate function and ensuring its execution before fetching folders.

Changes to Project Plan:

No significant changes to the original project plan; however, continuous adjustments will be made based on evolving requirements and feedback.

BHAVYA ANAND (WEEK-3)

Action Items:

File/Folder Management:

1. Implement CRUD operations for files and folders.
2. Establish secure file upload and download mechanisms.

Performance Monitoring:

1. Implement metrics tracking for key performance indicators.
2. Set up monitoring tools for continuous performance assessment.

Attachments:

Click here to access the GitHub repository for the project.

(All the codes have been already pushed to GitHub)