2 / 2Create a Next.Js application, the application has the following requirements: 1. There are two tabs on the page - First tab has title "Choose File", Second tab has title "File Info" 2. Upon click the tab title, page body will show a dash line rectangular box with the page width, there is text "Drop files here or Click here to choose file" in the center of the box 3. User can drag and drop any single file to the rectangular box, after drop the file, the application will automatically switch the second tab "File Info", and page body will have rectangular pane with header of the file name, and shows the information about the file - name, size, last modified date, last access datetime etc. 4. Active selected tab should be highlighted visually, the default active tab is "Choose File" 5. When users click the dash line rectangular box, a file choose dialog pops up to allow users to select a file from local file system. after choosing a file, the second tab "File Info" will be activated, and show file information in body like step 3 6. Use Material UI to make UI more modern. Provide step by step setup and create the Next.js app to achieve the above listed requirements.

To create the Next.js application with the requirements mentioned, follow the step-by-step instructions below:

Step 1: Install and create a new Next.js app

1. Make sure you have Node.js 12 or later installed. You can check the version by running `node -v` in the terminal.
2. Install the `create-next-app` CLI tool globally with the following command: `npm install -g create-next-app`.
3. Create a new Next.js app by running `create-next-app my-next-app`. Replace "my-next-app" with your preferred project name.
4. Change to the new project directory using `cd my-next-app`.

Step 2: Install required dependencies

Install Material-UI and required dependencies:

```
bashCopy code
npm install @mui/material @emotion/react @emotion/styled
```

Install the `date-fns` library to handle date formatting:

```
bashCopy code
npm install date-fns
```

Step 3: Create components

Create a new folder called "components" in the root directory of the project. Inside the "components" folder, create the following files:

- `TabPanel.js`
- `FileDropzone.js`
- `FileInfo.js`

Step 4: Implement the TabPanel component

Open `TabPanel.js` and paste the following code:

```
jsxCopy code
import React from 'react';
import Box from '@mui/material/Box';

export default function TabPanel(props) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`tabpanel-${index}`}
      {...other}
    >
      {value === index && <Box>{children}</Box>}
    </div>
  );
}
```

Step 5: Implement the FileDropzone component

Open `FileDropzone.js` and paste the following code:

```
jsxCopy code
import React from 'react';
import { Box, Typography } from '@mui/material';

const styles = {
  container: {
    border: '2px dashed #000',
    borderRadius: 4,
    padding: 2,
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
    cursor: 'pointer',
  },
};

export default function FileDropzone({ onFileSelected }) {
  const handleFileInput = (e) => {
    onFileSelected(e.target.files[0]);
  };

  const openFileDialog = () => {
    const fileInput = document.createElement('input');
    fileInput.type = 'file';
    fileInput.addEventListener('change', handleFileInput);
    fileInput.click();
  };

  return (
    <Box
      sx={styles.container}
      onClick={openFileDialog}
    >
      <Typography variant="h6">
        Drop files here or Click here to choose a file
      </Typography>
    </Box>
  );
}
```

Step 6: Implement the FileInfo component

Open `FileInfo.js` and paste the following code:

```
jsxCopy code
import React from 'react';
import { Box, Typography } from '@mui/material';
import { format } from 'date-fns';

const styles = {
  container: {
    padding: 2,
  },
};

export default function FileInfo({ file }) {
  return (
    <Box sx={styles.container}>
      <Typography variant="h6">File Information:</Typography>
      <Typography>Name: {file.name}</Typography>
      <Typography>Size: {file.size} bytes</Typography>
      <Typography>
        Last Modified: {format(file.lastModified, 'PPpp')}
      </Typography>
      {/* Add any other file information you'd like to
display here */}
    </Box>
  );
}
```

1 / 1continue

Step 7: Implement the main application logic

Open the `pages/index.js` file and replace its content with the following code:

```
jsxCopy code
import React, { useState } from 'react';
import {
  Box,
  AppBar,
  Tabs,
  Tab,
  Container,
} from '@mui/material';
import TabPanel from '../components/TabPanel';
import FileDropzone from '../components/FileDropzone';
import FileInfo from '../components/FileInfo';

export default function Home() {
  const [tabIndex, setTabIndex] = useState(0);
  const [selectedFile, setSelectedFile] = useState(null);

  const handleTabChange = (event, newValue) => {
    setTabIndex(newValue);
  };

  const handleFileSelected = (file) => {
    setSelectedFile(file);
    setTabIndex(1);
  };

  return (
    <Container maxWidth="md">
      <Box sx={{ flexGrow: 1 }}>
        <AppBar position="static">
          <Tabs
            value={tabIndex}
            onChange={handleTabChange}
            indicatorColor="secondary"
            textColor="inherit"
            variant="fullWidth"
          >
            <Tab label="Choose File" />
            <Tab label="File Info" />
          </Tabs>
        </AppBar>
        <TabPanel value={tabIndex} index={0}>
          <FileDropzone onFileSelected={handleFileSelected} /
>
        </TabPanel>
        <TabPanel value={tabIndex} index={1}>
          {selectedFile && <FileInfo file={selectedFile} />}
        </TabPanel>
      </Box>
    </Container>
  );
}
```

Step 8: Test the application

1. Save all the files.
2. Run the development server using the command `npm run dev` in the terminal.
3. Open your browser and navigate to `http://localhost:3000`. You should now see the "Choose File" tab, and you can interact with the file dropzone to select a file. After selecting a file, the "File Info" tab will show the selected file's information.

Your Next.js application should now meet all the requirements specified.

1 / 1Make a dash line rectangular box to be 400px height

To set the height of the dash line rectangular box to 400px, update the `styles` object in the `FileDropzone.js` component.

Step 1: Update the FileDropzone component

Open `FileDropzone.js` and modify the `styles` object as follows:

```
javascriptCopy code
const styles = {
  container: {
    border: '2px dashed #000',
    borderRadius: 4,
    padding: 2,
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
    cursor: 'pointer',
    height: '400px', // Add this line
  },
};
```
Step 2: Test the application

1. Save the `FileDropzone.js` file.
2. Make sure the development server is running. If not, run the command `npm run dev` in the terminal.
3. Open your browser and navigate to `http://localhost:3000`. You should now see the dash line rectangular box with a height of 400px.

1 / 1Fix the issue that "a dash line rectangular box" failed to capture the drop file.

To capture the drop file in the "a dash line rectangular box," you need to add event listeners for drag and drop events. Update the `FileDropzone.js` component as follows:

Step 1: Update the FileDropzone component

Open `FileDropzone.js` and modify the component to handle drag and drop events:

```
jsxCopy code
import React, { useCallback } from 'react';
import { Box, Typography } from '@mui/material';

const styles = {
  container: {
    border: '2px dashed #000',
    borderRadius: 4,
    padding: 2,
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
    cursor: 'pointer',
    height: '400px',
  },
};

export default function FileDropzone({ onFileSelected }) {
  const handleFileInput = (e) => {
    onFileSelected(e.target.files[0]);
  };
  const openFileDialog = () => {
    const fileInput = document.createElement('input');
    fileInput.type = 'file';
    fileInput.addEventListener('change', handleFileInput);
    fileInput.click();
  };

  const handleDrop = useCallback(
    (e) => {
      e.preventDefault();
      e.stopPropagation();
      const file = e.dataTransfer.files[0];
      onFileSelected(file);
    },
    [onFileSelected],
  );

  const handleDragOver = useCallback((e) => {
    e.preventDefault();
    e.stopPropagation();
  }, []);

  return (
    <Box
      sx={styles.container}
      onClick={openFileDialog}
      onDrop={handleDrop}
      onDragOver={handleDragOver}
    >
      <Typography variant="h6">
        Drop files here or Click here to choose a file
      </Typography>
    </Box>
  );
}
```

Step 2: Test the application

1. Save the `FileDropzone.js` file.
2. Make sure the development server is running. If not, run the command `npm run dev` in the terminal.
3. Open your browser and navigate to `http://localhost:3000`. You should now be able to drag and drop a file into the dashed rectangle box. After dropping the file, the application should automatically switch to the "File Info" tab and display the file's information.

1 / 1Make a ""a dash line rectangular box" visually different when a file is dragging over it, back to normal when dragging file out of it or no dragging file

To change the appearance of the "a dash line rectangular box" when a file is dragged over it, you need to update the component's state and styling based on the drag events. Update the `FileDropzone.js` component as follows:

Step 1: Update the FileDropzone component

Open `FileDropzone.js` and modify the component to handle drag events and change the appearance accordingly:

```
jsxCopy code
import React, { useState, useCallback } from 'react';
import { Box, Typography } from '@mui/material';

const styles = {
  container: (isDragActive) => ({
    border: '2px dashed #000',
    borderRadius: 4,
    padding: 2,
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
    cursor: 'pointer',
    height: '400px',
    backgroundColor: isDragActive ? '#f0f0f0' :
'transparent', // Change the background color when a file is
dragged over the box
  }),
};

export default function FileDropzone({ onFileSelected }) {
  const [isDragActive, setIsDragActive] = useState(false);
  const handleFileInput = (e) => {
    onFileSelected(e.target.files[0]);
  };

  const openFileDialog = () => {
    const fileInput = document.createElement('input');
    fileInput.type = 'file';
    fileInput.addEventListener('change', handleFileInput);
    fileInput.click();
  };

  const handleDrop = useCallback(
    (e) => {
      e.preventDefault();
      e.stopPropagation();
      setIsDragActive(false);
      const file = e.dataTransfer.files[0];
      onFileSelected(file);
    },
    [onFileSelected],
  );

  const handleDragOver = useCallback((e) => {
    e.preventDefault();
    e.stopPropagation();
    setIsDragActive(true);
  }, []);

  const handleDragLeave = useCallback((e) => {
    e.preventDefault();
    e.stopPropagation();
    setIsDragActive(false);
  }, []);

  return (
    <Box
      sx={styles.container(isDragActive)}
      onClick={openFileDialog}
      onDrop={handleDrop}
      onDragOver={handleDragOver}
      onDragLeave={handleDragLeave}
    >
      <Typography variant="h6">
        Drop files here or Click here to choose a file
      </Typography>
    </Box>
  );
}
```

Step 2: Test the application

1. Save the `FileDropzone.js` file.
2. Make sure the development server is running. If not, run the command `npm run dev` in the terminal.

3. Open your browser and navigate to `http://localhost:3000`. You should now see the dashed rectangle box change its appearance when a file is dragged over it and revert to the original appearance when the file is dragged out or dropped.

1 / 1Modify the Next.js application to center the UI on screen, make a text size a bigger

To center the UI on the screen and increase the text size, update the `pages/index.js` and `components/FileDropzone.js` files as follows:

Step 1: Update the Home component

Open `pages/index.js` and modify the `Container` component and its children:

```
jsxCopy code
import React, { useState } from 'react';
import {
  Box,
  AppBar,
  Tabs,
  Tab,
  Container,
  Typography,
} from '@mui/material';
import TabPanel from '../components/TabPanel';
import FileDropzone from '../components/FileDropzone';
import FileInfo from '../components/FileInfo';

export default function Home() {
  const [tabIndex, setTabIndex] = useState(0);
  const [selectedFile, setSelectedFile] = useState(null);

  const handleTabChange = (event, newValue) => {
    setTabIndex(newValue);
  };

  const handleFileSelected = (file) => {
    setSelectedFile(file);
    setTabIndex(1);
  };

  return (
    <Container
      maxWidth="md"
      sx={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        justifyContent: 'center',
        minHeight: '100vh',
        padding: '1rem',
      }}
    >
      <Typography variant="h3" component="h1" gutterBottom>
        My File Info App
      </Typography>
      <Box sx={{ flexGrow: 1, width: '100%' }}>
        <AppBar position="static">
          <Tabs
            value={tabIndex}
            onChange={handleTabChange}
            indicatorColor="secondary"
            textColor="inherit"
            variant="fullWidth"
          >
            <Tab label="Choose File" />
            <Tab label="File Info" />
          </Tabs>
        </AppBar>
        <TabPanel value={tabIndex} index={0}>
          <FileDropzone onFileSelected={handleFileSelected} /
>
        </TabPanel>
        <TabPanel value={tabIndex} index={1}>
          {selectedFile && <FileInfo file={selectedFile} />}
        </TabPanel>
      </Box>
    </Container>
  );
}
```
Step 2: Update the FileDropzone component

Open `components/FileDropzone.js` and modify the `Typography` component:

```
jsxCopy code
// ... (other imports and code)

return (
  <Box
    sx={styles.container(isDragActive)}
    onClick={openFileDialog}
    onDrop={handleDrop}
    onDragOver={handleDragOver}
    onDragLeave={handleDragLeave}
  >
    <Typography variant="h5">
      Drop files here or Click here to choose a file
    </Typography>
  </Box>
);

// ... (other code)
```

Step 3: Test the application

1. Save both `pages/index.js` and `components/FileDropzone.js` files.
2. Make sure the development server is running. If not, run the command `npm run dev` in the terminal.
3. Open your browser and navigate to `http://localhost:3000`. You should now see the UI centered on the screen and the text size increased.

1 / 1make "3rem" gap between tab bar and tab panel

To add a `3rem` gap between the tab bar and tab panel, update the `pages/index.js` file as follows:

Step 1: Update the Home component

Open `pages/index.js` and modify the `Box` component containing the `TabPanel` components:

```
jsxCopy code
import React, { useState } from 'react';
import {
  Box,
  AppBar,
  Tabs,
  Tab,
  Container,
  Typography,
} from '@mui/material';
import TabPanel from '../components/TabPanel';
import FileDropzone from '../components/FileDropzone';
import FileInfo from '../components/FileInfo';
```

```
export default function Home() {
  const [tabIndex, setTabIndex] = useState(0);
  const [selectedFile, setSelectedFile] = useState(null);

  const handleTabChange = (event, newValue) => {
    setTabIndex(newValue);
  };

  const handleFileSelected = (file) => {
    setSelectedFile(file);
    setTabIndex(1);
  };

  return (
    <Container
      maxWidth="md"
      sx={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        justifyContent: 'center',
        minHeight: '100vh',
        padding: '1rem',
      }}
    >
      <Typography variant="h3" component="h1" gutterBottom>
        My File Info App
      </Typography>
      <Box sx={{ flexGrow: 1, width: '100%' }}>
        <AppBar position="static">
          <Tabs
            value={tabIndex}
            onChange={handleTabChange}
            indicatorColor="secondary"
            textColor="inherit"
            variant="fullWidth"
          >
            <Tab label="Choose File" />
            <Tab label="File Info" />
          </Tabs>
        </AppBar>
        <Box sx={{ marginTop: '3rem' }}> {/* Add this Box 
component to create the gap */}
          <TabPanel value={tabIndex} index={0}>
```

```
            <FileDropzone onFileSelected={handleFileSelected}
/>
          </TabPanel>
          <TabPanel value={tabIndex} index={1}>
            {selectedFile && <FileInfo file={selectedFile} /
>}
          </TabPanel>
        </Box>
      </Box>
    </Container>
  );
}
```

Step 2: Test the application

1. Save the `pages/index.js` file.
2. Make sure the development server is running. If not, run the command `npm run dev` in the terminal.
3. Open your browser and navigate to `http://localhost:3000`. You should now see a `3rem` gap between the tab bar and tab panel.

0