```
// DocumentComponents.tsx
import { useState } from "react"
import { format } from 'date-fns'
import { Card, CardContent, CardFooter, CardHeader, CardTitle } from "../spread_sheet_swarm/ui/card"
import { File, FileText, FileSpreadsheet, FileImage, FileIcon, FileJson, Share2, Plus } from "lucide-react"
import { Button } from "../spread_sheet_swarm/ui/button"
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
  DialogTrigger,
} from "../spread_sheet_swarm/ui/dialog"
import { Label } from "../spread_sheet_swarm/ui/label"
import { Input } from "../spread_sheet_swarm/ui/input"
import { ScrollArea } from "../ui/scroll-area"
import { AccessControl } from "@/lib/access-control"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "../ui/select"

// Types
interface User {
  id: string;
```

```
  email: string;
  role: 'admin' | 'user';
  organization_id?: string;
}

interface Document {
  id: string;
  name: string;
  type: string;
  size: string;
  user_id: string;
  organization_id?: string;
  uploaded_by_email: string;
  upload_date: string;
  content: string;
  checksum: string;
  version: number;
  last_modified: string;
  status: 'processing' | 'completed' | 'error';
  metadata?: Record<string, any>;
  permissions: {
    can_read: string[];
    can_write: string[];
    can_delete: string[];
    is_public: boolean;
  };
```

```
}

interface DocumentCardProps {
  document: Document;
  currentUser: User;
  onShare: (shareWith: { email: string; permissions: ('read' | 'write' | 'delete')[] }) => void;
  onClick: () => void;
}

interface ShareDocumentDialogProps {
  isOpen: boolean;
  onClose: () => void;
  onShare: (shareWith: { email: string; permissions: ('read' | 'write' | 'delete')[] }) => void;
}

interface DocumentDetailsModalProps {
  document: Document;
  currentUser: User;
  onClose: () => void;
  onShare: (shareWith: { email: string; permissions: ('read' | 'write' | 'delete')[] }) => void;
}

// Document Card Component
const DocumentCard = ({ document, currentUser, onShare, onClick }: DocumentCardProps) => {
  const [showShareDialog, setShowShareDialog] = useState(false);
```

```
  const getIconForDocumentType = (type: string) => {
    switch (type) {
      case "txt":
        return <FileText className="h-6 w-6" />
      case "csv":
        return <FileSpreadsheet className="h-6 w-6" />
      case "pdf":
        return <FileIcon className="h-6 w-6" />
      case "image":
        return <FileImage className="h-6 w-6" />
      case "json":
        return <FileJson className="h-6 w-6" />
      default:
        return <File className="h-6 w-6" />
    }
  };

  return (
    <>
                <Card  className="flex  flex-col  cursor-pointer  hover:shadow-lg  transition-shadow
duration-200">
        <CardHeader className="flex flex-row items-center justify-between">
          <CardTitle className="flex items-center gap-2">
            {getIconForDocumentType(document.type)}
            <span className="truncate">{document.name}</span>
          </CardTitle>
```

```
    {AccessControl.canWrite(document, currentUser.id) && (
      <Button
        variant="ghost"
        size="sm"
        onClick={(e) => {
          e.stopPropagation();
          setShowShareDialog(true);
        }}
      >
        <Share2 className="h-4 w-4" />
      </Button>
    )}
  </CardHeader>
  <CardContent onClick={onClick}>
    <p className="text-sm text-gray-500">Type: {document.type}</p>
    <p className="text-sm text-gray-500">Size: {document.size}</p>
    <p className="text-sm text-gray-500">Uploaded by: {document.uploaded_by_email}</p>
  </CardContent>
  <CardFooter className="text-sm text-gray-400 mt-auto">
    Uploaded on {format(new Date(document.upload_date), 'MMM dd, yyyy')}
  </CardFooter>
</Card>

{showShareDialog && (
  <ShareDocumentDialog
    isOpen={showShareDialog}
```

```
          onClose={() => setShowShareDialog(false)}
          onShare={onShare}
        />
      )}
    </>
  );
};

// Share Document Dialog Component
const ShareDocumentDialog = ({ isOpen, onClose, onShare }: ShareDocumentDialogProps) => {
  const [email, setEmail] = useState('');
  const [permissions, setPermissions] = useState<('read' | 'write' | 'delete')[]>(['read']);

  const handleShare = () => {
    onShare({ email, permissions });
    onClose();
  };

  return (
    <Dialog open={isOpen} onOpenChange={onClose}>
      <DialogContent>
        <DialogHeader>
          <DialogTitle>Share Document</DialogTitle>
          <DialogDescription>
            Enter the email of the user you want to share this document with.
          </DialogDescription>
```

```
</DialogHeader>
<div className="grid gap-4 py-4">
  <div className="grid grid-cols-4 items-center gap-4">
    <Label htmlFor="email" className="text-right">
      Email
    </Label>
    <Input
      id="email"
      type="email"
      value={email}
      onChange={(e) => setEmail(e.target.value)}
      className="col-span-3"
    />
  </div>
  <div className="grid grid-cols-4 items-center gap-4">
    <Label className="text-right">Permissions</Label>
    <div className="col-span-3 flex gap-2">
      <label className="flex items-center gap-2">
        <input
          type="checkbox"
          checked={permissions.includes('read')}
          onChange={(e) => {
            if (e.target.checked) {
              setPermissions([...permissions, 'read']);
            } else {
              setPermissions(permissions.filter(p => p !== 'read'));
```

```
      }
    }}
  />
  Read
</label>
<label className="flex items-center gap-2">
  <input
    type="checkbox"
    checked={permissions.includes('write')}
    onChange={(e) => {
      if (e.target.checked) {
        setPermissions([...permissions, 'write']);
      } else {
        setPermissions(permissions.filter(p => p !== 'write'));
      }
    }}
  />
  Write
</label>
<label className="flex items-center gap-2">
  <input
    type="checkbox"
    checked={permissions.includes('delete')}
    onChange={(e) => {
      if (e.target.checked) {
        setPermissions([...permissions, 'delete']);
```

```
                } else {
                  setPermissions(permissions.filter(p => p !== 'delete'));
                }
              }}
            />
            Delete
          </label>
        </div>
      </div>
      <DialogFooter>
        <Button onClick={onClose} variant="outline">Cancel</Button>
        <Button onClick={handleShare}>Share</Button>
      </DialogFooter>
    </DialogContent>
  </Dialog>
 );
};

// Document Preview Component
const DocumentPreview = ({ document }: { document: Document }) => {
 switch (document.type) {
  case "csv":
   return (
    <table className="w-full border-collapse">
     <tbody>
```

```
          {document.content.split("\n").map((row, i) => (
            <tr key={i}>
              {row.split(",").map((cell, j) => (
                <td key={j} className="border px-2 py-1">{cell}</td>
              ))}
            </tr>
          ))}
        </tbody>
      </table>
    );
  case "txt":
    return <pre className="whitespace-pre-wrap">{document.content}</pre>;
  case "pdf":
    return (
      <div className="text-center">
        <p>PDF preview not available.</p>
        <Button className="mt-2">Download PDF</Button>
      </div>
    );
  case "image":
    return <img src={document.content} alt={document.name} className="max-w-full h-auto" />;
  default:
    return <p>Preview not available for this file type.</p>;
 }
};
```

```
// Document Details Modal Component
const DocumentDetailsModal = ({ document, currentUser, onClose, onShare }: DocumentDetailsModalProps) => {
  const [showShareDialog, setShowShareDialog] = useState(false);

  return (
    <>
      <Dialog open={true} onOpenChange={onClose}>
        <DialogContent className="sm:max-w-[700px]">
          <DialogHeader>
            <div className="flex items-center justify-between">
              <DialogTitle>{document.name}</DialogTitle>
              {AccessControl.canWrite(document, currentUser.id) && (
                <Button
                  variant="outline"
                  size="sm"
                  onClick={() => setShowShareDialog(true)}
                >
                  <Share2 className="h-4 w-4 mr-2" />
                  Share
                </Button>
              )}
            </div>
            <DialogDescription>
              Type: {document.type.toUpperCase()} | Size: {document.size} |
                                    Uploaded by: {document.uploaded_by_email} on {format(new
```

```
Date(document.upload_date), 'MMM dd, yyyy')}
            </DialogDescription>
          </DialogHeader>
          <div className="mt-4">
            <h3 className="text-lg font-semibold mb-2">Preview:</h3>
            <ScrollArea className="h-[300px] w-full rounded-md border p-4">
              <DocumentPreview document={document} />
            </ScrollArea>
          </div>
          <DialogFooter>
            <Button onClick={onClose}>Close</Button>
          </DialogFooter>
        </DialogContent>
      </Dialog>

      {showShareDialog && (
        <ShareDocumentDialog
          isOpen={showShareDialog}
          onClose={() => setShowShareDialog(false)}
          onShare={onShare}
        />
      )}
    </>
  );
};
```

```
// DocumentComponents.tsx (Add this to the previous file)
interface AddDocumentDialogProps {
  onAddDocument: (doc: Omit<Document, 'user_id' | 'organization_id' | 'uploaded_by_email'>) =>
void;
}

const AddDocumentDialog = ({ onAddDocument }: AddDocumentDialogProps) => {
  const [name, setName] = useState("");
  const [type, setType] = useState("");
  const [size, setSize] = useState("");
  const [content, setContent] = useState("");

  const handleSubmit = () => {
    const newDocument: any = {
      id: uuidv4(),
      name,
      type,
      size,
      content,
      checksum: '',
      version: 1,
      last_modified: new Date().toISOString(),
      upload_date: new Date().toISOString(),
      status: 'processing' as const,
      metadata: {},
```

```
    permissions: {
      can_read: [],
      can_write: [],
      can_delete: [],
      is_public: false,
    },
  };
  onAddDocument(newDocument);
  setName("");
  setType("");
  setSize("");
  setContent("");
};

return (
  <Dialog>
    <DialogTrigger asChild>
      <Button>
        <Plus className="mr-2 h-4 w-4" /> Add Document
      </Button>
    </DialogTrigger>
    <DialogContent className="sm:max-w-[425px]">
      <DialogHeader>
        <DialogTitle>Add New Document</DialogTitle>
        <DialogDescription>
          Enter the details of the new document you want to add to the data hub.
```

```
  </DialogDescription>
</DialogHeader>
<div className="grid gap-4 py-4">
  <div className="grid grid-cols-4 items-center gap-4">
    <Label htmlFor="name" className="text-right">
      Name
    </Label>
    <Input
      id="name"
      value={name}
      onChange={(e) => setName(e.target.value)}
      className="col-span-3"
    />
  </div>
  <div className="grid grid-cols-4 items-center gap-4">
    <Label htmlFor="type" className="text-right">
      Type
    </Label>
    <Select value={type} onValueChange={setType}>
      <SelectTrigger className="col-span-3">
        <SelectValue placeholder="Select document type" />
      </SelectTrigger>
      <SelectContent>
        <SelectItem value="txt">Text</SelectItem>
        <SelectItem value="csv">CSV</SelectItem>
        <SelectItem value="pdf">PDF</SelectItem>
```

```
      <SelectItem value="image">Image</SelectItem>
      <SelectItem value="json">JSON</SelectItem>
    </SelectContent>
  </Select>
</div>
<div className="grid grid-cols-4 items-center gap-4">
  <Label htmlFor="size" className="text-right">
    Size
  </Label>
  <Input
    id="size"
    value={size}
    onChange={(e) => setSize(e.target.value)}
    className="col-span-3"
    placeholder="e.g., 2.3 MB"
  />
</div>
<div className="grid grid-cols-4 items-center gap-4">
  <Label htmlFor="content" className="text-right">
    Content
  </Label>
  <Input
    id="content"
    value={content}
    onChange={(e) => setContent(e.target.value)}
    className="col-span-3"
```

```
            placeholder="Enter content or file path"
          />
        </div>
      </div>
      <DialogFooter>
        <Button type="submit" onClick={handleSubmit}>
          Add Document
        </Button>
      </DialogFooter>
    </DialogContent>
  </Dialog>
 );
};

export { DocumentCard, ShareDocumentDialog, DocumentDetailsModal, DocumentPreview, AddDocumentDialog };

function uuidv4() {
  throw new Error("Function not implemented.")
}
```