

Assignment Grouping Master

config/db.js

```
const mongoose = require("mongoose");

const connectDB = async () => {
  try {
    await mongoose.connect("mongodb://127.0.0.1:27017/assignmentDB");
    console.log("MongoDB Connected");
  } catch (error) {
    console.error(error.message);
    process.exit(1);
  }
};

module.exports = connectDB;
```

controllers/assignmentController.js

```
const Assignment = require("../models/Assignment");
const fs = require("fs");
const path = require("path");

/* CREATE */

exports.createAssignment = async (req, res) => {
  try {
    const assignment = new Assignment({
      name: req.body.name,
```

```
    todo: req.body.todo,
    date: req.body.date,
    file: req.file?.filename || ""
  });

  const saved = await assignment.save();
  res.status(201).json(saved);

} catch (err) {
  if (err.code === 11000) {
    return res.status(409).json({
      message: "Duplicate assignment not allowed"
    });
  }
  res.status(500).json({ message: err.message });
}

};

/* READ */

exports.getAssignments = async (req, res) => {
  try {
    const data = await Assignment.find().sort({ createdAt: -1 });
    res.json(data);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
};
```

```
/* UPDATE (delete old file if new uploaded) */

exports.updateAssignment = async (req, res) => {

  try {

    const assignment = await Assignment.findById(req.params.id);

    if (!assignment) return res.status(404).json({ message: "Not found" });

    // ✎ delete old file

    if (req.file && assignment.file) {

      const oldPath = path.join(__dirname, "..", "uploads", assignment.file);

      if (fs.existsSync(oldPath)) fs.unlinkSync(oldPath);

    }

    assignment.name = req.body.name;

    assignment.todo = req.body.todo;

    assignment.date = req.body.date;

    if (req.file) assignment.file = req.file.filename;

    const updated = await assignment.save();

    res.json(updated);

  } catch (err) {

    res.status(500).json({ message: err.message });

  }

};

/* DELETE (delete file also) */

exports.deleteAssignment = async (req, res) => {

  try {

    // Find assignment first
```

```
const assignment = await Assignment.findById(req.params.id);

if (!assignment) {
    return res.status(404).json({ message: "Assignment not found" });
}

// Delete file from uploads folder

if (assignment.file) {

    const filePath = path.join(__dirname, "..", "uploads", assignment.file);

    if (fs.existsSync(filePath)) {

        fs.unlinkSync(filePath); // ⚡ FILE DELETED

    }
}

// Delete MongoDB record

await Assignment.findByIdAndUpdateAndDelete(req.params.id);

res.json({ success: true });

} catch (err) {

    console.error(err);

    res.status(500).json({ message: err.message });

}

};
```

models/Assignment.js

```
const mongoose = require("mongoose");

const AssignmentSchema = new mongoose.Schema( {
```

```
    name: { type: String, required: true },  
    todo: { type: String, required: true },  
    date: { type: String, required: true },  
    file: { type: String } // filename stored from Multer  
,  
    { timestamps: true }  
};
```

```
AssignmentSchema.index(
```

```
    { name: 1, todo: 1, date: 1 },
```

```
    { unique: true }  
);
```

```
module.exports = mongoose.model("Assignment", AssignmentSchema);
```

routes/assignmentRoutes.js

```
const express = require("express");  
  
const multer = require("multer");  
  
const {  
    createAssignment,  
    getAssignments,  
    updateAssignment,  
    deleteAssignment  
} = require("../controllers/assignmentController");  
  
const router = express.Router();  
  
/* Multer config */  
  
const storage = multer.diskStorage({
```

```
destination: "uploads/",  
  
filename: (req, file, cb) => {  
  
  cb(null, Date.now() + "-" + file.originalname);  
  
}  
  
});  
  
const upload = multer({ storage });  
  
// ROUTES  
  
router.post("/", upload.single("file"), createAssignment);  
  
router.get("/", getAssignments);  
  
// ● UPDATE MUST USE multer  
  
router.put("/:id", upload.single("file"), updateAssignment);  
  
// ● DELETE  
  
router.delete("/:id", deleteAssignment);  
  
module.exports = router;
```

server.js

```
const express = require("express");  
  
const cors = require("cors");  
  
const path = require("path");  
  
const connectDB = require("./config/db");  
  
const assignmentRoutes = require("./routes/assignmentRoutes");  
  
const app = express();  
  
// Connect to MongoDB  
  
connectDB();
```

```
// Middleware

app.use(cors());

app.use(express.json());

app.use("/uploads", express.static(path.join(__dirname, "uploads"))); // serve uploaded files
```

```
// Routes
```

```
app.use("/api/assignments", assignmentRoutes);
```

```
// Start server
```

```
const PORT = process.env.PORT || 5000;
```

```
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

```
uploads
```

```
src/components/AssignmentDialog.js
```

```
import React, { useState, useEffect } from "react";
```

```
import {
```

```
    Dialog,
```

```
    DialogTitle,
```

```
    DialogContent,
```

```
    DialogActions,
```

```
    Button,
```

```
    TextField,
```

```
    Stack,
```

```
    IconButton,
```

```
    Typography
```

```
} from "@mui/material";
```

```
import VisibilityIcon from "@mui/icons-material/Visibility";  
  
export default function AssignmentDialog({ open, onClose, data, mode, onSave }) {  
  
  const [form, setForm] = useState({  
  
    name: "",  
  
    date: "",  
  
    todo: "",  
  
    file: null, // newly selected file  
  
    fileName: "" // existing uploaded file name  
  
  });  
  
  useEffect(() => {  
  
    if (data) {  
  
      setForm({  
  
        name: data.name || "",  
  
        date: data.date || "",  
  
        todo: data.todo || "",  
  
        file: null,  
  
        fileName: data.file || ""  
  
      });  
  
    }  
  
  }, [data]);  
  
  const handleSave = () => {  
  
    const fd = new FormData();  
  
    fd.append("name", form.name);  
  
    fd.append("date", form.date);  
  
    fd.append("todo", form.todo);  
  };  
}
```

```
if (form.file) fd.append("file", form.file);

onSave(fd, data._id);

onClose();

};

return (

<Dialog open={open} onClose={onClose} fullWidth maxWidth="sm">

<DialogTitle>

{mode === "edit" ? "Edit Assignment" : "View Assignment"}

</DialogTitle>

<DialogContent>

<Stack spacing={2} sx={{ m: 2 }}>

{/* Name */}

<Stack direction="row" spacing={2}>

<Typography sx={{ width: 100 }} color="text.secondary">

Name:

</Typography>

{mode === "edit" ? (
<TextField
fullWidth
value={form.name}
onChange={(e) =>
setForm({ ...form, name: e.target.value })
}
/>
) : (
```

```
<Typography>{form.name}</Typography>
```

```
)}
```

```
</Stack>
```

```
/* Date */
```

```
<Stack direction="row" spacing={2}>
```

```
<Typography sx={{ width: 100 }} color="text.secondary">
```

```
Date:
```

```
</Typography>
```

```
{mode === "edit" ? (
```

```
<TextField
```

```
type="date"
```

```
fullWidth
```

```
value={form.date}
```

```
onChange={(e) =>
```

```
setForm({ ...form, date: e.target.value })
```

```
}
```

```
/>
```

```
):(
```

```
<Typography>{form.date}</Typography>
```

```
)}
```

```
</Stack>
```

```
/* Todo */
```

```
<Stack direction="row" spacing={2}>
```

```
<Typography sx={{ width: 100 }} color="text.secondary">
```

```
Todo:
```

```
</Typography>
```

```
{mode === "edit" ? (  
  <TextField  
    fullWidth  
    value={form.todo}  
    onChange={(e) =>  
      setForm({ ...form, todo: e.target.value })  
    }  
  />  
) : (  
  <Typography>{form.todo}</Typography>  
)  
</Stack>
```

```
/* File */
```

```
<Stack direction="row" spacing={2} alignItems="flex-start">  
  <Typography sx={{ width: 100 }} color="text.secondary">  
    File:  
  </Typography>
```

```
{mode === "edit" ? (  
  <Stack spacing={1}>  
    /* Existing file */  
    {form.fileName ? (  
      <Stack direction="row" spacing={1} alignItems="center">  
        <Typography>{form.fileName}</Typography>  
        <IconButton  
          size="small"  
          color="primary"
```

```
    onClick={() =>
      window.open(
        `http://localhost:5000/uploads/${form.fileName}`,
        "_blank"
      )
    }
  >
  <VisibilityIcon />
</IconButton>
</Stack>
) : (
<Typography>No file uploaded</Typography>
)
/* Upload new file */
<Button component="label" variant="outlined" size="small">
  Upload New File
<input
  hidden
  type="file"
  onChange={(e) =>
    setForm({ ...form, file: e.target.files[0] })
  }
/>
</Button>
/* New file preview */
{form.file && (
```

```
<Stack direction="row" spacing={1} alignItems="center">

  <Typography variant="body2">

    New File: {form.file.name}

  </Typography>

  <IconButton size="small" color="primary" onClick={() =>
    window.open(
      URL.createObjectURL(form.file),
      "_blank"
    )
  }>

    >

    <VisibilityIcon />

  </IconButton>

</Stack>

) }

</Stack>

) : form.fileName ? (
  <Stack direction="row" spacing={1} alignItems="center">

    <Typography>{form.fileName}</Typography>

    <IconButton size="small" color="primary" onClick={() =>
      window.open(

```

```
    `http://localhost:5000/uploads/${form.fileName},  
    "_blank"  
)  
}  
>  
<VisibilityIcon />  
</IconButton>  
</Stack>  
) : (  
<Typography>No file uploaded</Typography>  
)  
</Stack>  
</Stack>  
</DialogContent>  
  
<DialogActions>  
{mode === "edit" && (  
<Button variant="contained" onClick={handleSave}>  
  Save  
</Button>  
)  
<Button variant="outlined" onClick={onClose}>  
  Close  
</Button>  
</DialogActions>  
</Dialog>  
);  
}
```

AssignmentForm.js

```
import React, { useState } from "react";
import axios from "axios";
import {
  Stack,
  TextField,
  Button,
  IconButton,
  Typography,
  Paper
} from "@mui/material";
import AddCircleIcon from "@mui/icons-material/AddCircle";
import EditIcon from "@mui/icons-material/Edit";
import DeleteIcon from "@mui/icons-material/Delete";
import UploadFileIcon from "@mui/icons-material/UploadFile";
import VisibilityIcon from "@mui/icons-material/Visibility";
export default function AssignmentForm({ onSubmit }) {
  const [name, setName] = useState("");
  const [date, setDate] = useState("");
  const [todoInput, setTodoInput] = useState("");
  const [file, setFile] = useState(null);
  const [tempTodos, setTempTodos] = useState([]);
  const [editingIndex, setEditingIndex] = useState(null);
  const [editingValue, setEditingValue] = useState("");
```

```
// Add todo
```

```
const handleAddTodo = () => {
  if (!todoInput.trim()) return;

  setTempTodos([...tempTodos, todoInput.trim()]);
  setTodoInput("");
};

};
```

```
// Cancel form
```

```
const handleCancel = () => {
  setName("");
  setDate("");
  setTodoInput("");
  setTempTodos([]);
  setFile(null);
  setEditingIndex(null);
  setEditingValue("");
};

};
```

```
// File select
```

```
const handleFileChange = (e) => {
  const selected = e.target.files[0];
  if (!selected) return;

  if (selected.size > 5 * 1024 * 1024) {
    alert("File must be under 5MB");
    return;
}
```

```
    setFile(selected);

};

// Submit form

const handleSubmit = async () => {

  if (!name || !date || tempTodos.length === 0) {

    alert("Please fill all required fields");

    return;
  }

  try {

    for (const todo of tempTodos) {

      const formData = new FormData();

      formData.append("name", name);

      formData.append("date", date);

      formData.append("todo", todo);

      if (file) formData.append("file", file);

    }

    const res = await axios.post(
      "http://localhost:5000/api/assignments",
      formData,
      { headers: { "Content-Type": "multipart/form-data" } }
    );

    onSubmit(res.data);
  }
}

handleCancel();

} catch (err) {
```

```
console.error(err);  
  
// ✅ Handle backend duplicate error  
  
if (err.response?.status === 409) {  
  
  alert("This assignment already exists!");  
  
} else {  
  
  alert("Failed to submit assignment");  
  
}  
  
}  
  
};  
  
return (  
  
<>  
  
/* FORM INPUTS */  
  
<Stack direction="row" spacing={2} mb={2}>  
  
<TextField  
  
  label="Name"  
  
  size="small"  
  
  value={name}  
  
  onChange={(e) => setName(e.target.value)}  
  
/>  
  
<TextField  
  
  label="Todo"  
  
  size="small"  
  
  value={todoInput}  
  
  onChange={(e) => setTodoInput(e.target.value)}  
  
/>
```

```
<TextField  
  type="date"  
  size="small"  
  InputLabelProps={{ shrink: true }}  
  value={date}  
  onChange={(e) => setDate(e.target.value)}  
/>
```

```
<IconButton color="success" onClick={handleAddTodo}>  
  <AddCircleIcon />  
</IconButton>  
</Stack>
```

```
{/* TODO PREVIEW */}  
  
{tempTodos.length > 0 && (  
  <Paper sx={{ p: 2, mb: 2, width: 400 }}>  
    <Typography variant="subtitle2">Todo Preview</Typography>  
  
    {tempTodos.map((todo, index) => (  
      <Stack  
        key={index}  
        direction="row"  
        justifyContent="space-between"  
        alignItems="center"  
        sx={{ mt: 1 }}  
      >  
        {editingIndex === index ? (  
          <TextField
```

```
size="small"

value={editingValue}

onChange={(e) => setEditingValue(e.target.value)}

/>

):(

<Typography>{todo}</Typography>

)}

|<Stack direction="row">

{editingIndex === index ? (

<>

<IconButton

color="success"

onClick={() => {

const updated = [...tempTodos];

updated[index] = editingValue.trim();

setTempTodos(updated);

setEditingIndex(null);

setEditingValue("");

}

}>

✓

</IconButton>

|<IconButton

color="warning"

onClick={() => setEditingIndex(null)}>

|>
```

```
    <IconButton>
```

```
        </IconButton>
```

```
    </>
```

```
    ) : (
```

```
        <>
```

```
        <IconButton>
```

```
            color="primary"
```

```
            onClick={() => {
```

```
                setEditingIndex(index);
```

```
                setEditingValue(todo);
```

```
            }}
```

```
        >
```

```
        <EditIcon />
```

```
    </IconButton>
```

```
    <IconButton>
```

```
        color="error"
```

```
        onClick={() =>
```

```
            setTempTodos(tempTodos.filter((_, i) => i !== index))
```

```
        }
```

```
    >
```

```
        <DeleteIcon />
```

```
    </IconButton>
```

```
    </>
```

```
    )}
```

```
    </Stack>
```

```
    </Stack>
```

```
)})
```

```
</Paper>
```

```
)}
```

```
/* FILE + ACTIONS */
```

```
<Stack direction="row" spacing={2} alignItems="center" sx={{ mb: 3 }}>
```

```
<Button
```

```
variant="contained"
```

```
component="label"
```

```
startIcon={<UploadFileIcon />}
```

```
>
```

```
Upload
```

```
<input hidden type="file" onChange={handleFileChange} />
```

```
</Button>
```

```
{file && (
```

```
<Stack direction="row" spacing={1} alignItems="center">
```

```
<Typography variant="body2">{file.name}</Typography>
```

```
<IconButton
```

```
color="primary"
```

```
onClick={() => window.open(URL.createObjectURL(file))}
```

```
>
```

```
<VisibilityIcon />
```

```
</IconButton>
```

```
<IconButton color="error" onClick={() => setFile(null)}>
```

```
<DeleteIcon />
```

```
</IconButton>
```

```
</Stack>
```

```
)}
```

```
<Button variant="contained" onClick={handleSubmit}>
```

```
Submit
```

```
</Button>
```

```
<Button variant="outlined" color="error" onClick={handleCancel}>
```

```
Cancel
```

```
</Button>
```

```
</Stack>
```

```
</>
```

```
);
```

```
}
```

AssignmentTable.js

```
import { Box, IconButton, Stack, Button } from "@mui/material";
```

```
import { DataGrid } from "@mui/x-data-grid";
```

```
import VisibilityIcon from "@mui/icons-material/Visibility";
```

```
import EditIcon from "@mui/icons-material/Edit";
```

```
import DeleteIcon from "@mui/icons-material/Delete";
```

```
import React, { useState } from "react";
```

```
import AssignmentDialog from "./AssignmentDialog";
```

```
import axios from "axios";
```

```
export default function AssignmentTable({ rows, setRows }) {
```

```
  const [dialogOpen, setDialogOpen] = useState(false);
```

```
  const [dialogData, setDialogData] = useState({});
```

```
  const [dialogMode, setDialogMode] = useState("view");
```

```
const handleOpenDialog = (row, mode) => {
    setDialogData(row);
    setDialogMode(mode);
    setDialogOpen(true);
};
```

```
// ✓ UPDATE
```

```
const handleSaveEdit = async (formData, id) => {
    const res = await axios.put(
        `http://localhost:5000/api/assignments/${id}`,
        formData,
        { headers: { "Content-Type": "multipart/form-data" } }
    );
};
```

```
setRows((prev) =>
```

```
    prev.map((r) => (r._id === id ? res.data : r))
);
```

```
};
```

```
// ✓ DELETE (FIXED)
```

```
const handleDelete = async (id) => {
    await axios.delete(`http://localhost:5000/api/assignments/${id}`);
    setRows((prev) => prev.filter((r) => r._id !== id));
};
```

```
const columns = [
```

```
    { field: "name", headerName: "Name", flex: 1 },
    { field: "date", headerName: "Date", flex: 1 },
```

```
{ field: "todo", headerName: "Todo", flex: 2 },  
{  
  field: "file",  
  headerName: "File",  
  flex: 0.7,  
  renderCell: (params) =>  
    params.value ? (  
      <Button  
        size="small"  
        onClick={() =>  
          window.open(`http://localhost:5000/uploads/${params.value}`)}  
      >  
        View  
      </Button>  
    ) : (  
      "—"   
    )  
,  
{  
  field: "actions",  
  headerName: "Actions",  
  flex: 1,  
  renderCell: (params) => (  
    <Stack direction="row" spacing={1} >  
      <IconButton color="primary" onClick={() => handleOpenDialog(params.row, "view")}>  
        <VisibilityIcon />
```

```
</IconButton>
```

```
<IconButton color="success" onClick={() => handleOpenDialog(params.row, "edit")}>
```

```
<EditIcon />
```

```
</IconButton>
```

```
<IconButton color="error" onClick={() => handleDelete(params.row._id)}>
```

```
<DeleteIcon />
```

```
</IconButton>
```

```
</Stack>
```

```
)
```

```
}
```

```
];
```

```
return (
```

```
<Box sx={{ height: 400 }}>
```

```
<DataGrid
```

```
rows={rows}
```

```
columns={columns}
```

```
getRowId={(row) => row._id}
```

```
pageSizeOptions={[5]}
```

```
/>
```

```
<AssignmentDialog
```

```
open={dialogOpen}
```

```
onClose={() => setDialogOpen(false)}
```

```
data={dialogData}
```

```
mode={dialogMode}
```

```
onSave={handleSaveEdit}
```

```
/>
```

```
</Box>
```

```
);
```

```
}
```

SearchBar.js

```
import { TextField } from "@mui/material";
```

```
export default function SearchBar({ search, setSearch }) {
```

```
    return (
```

```
        <TextField
```

```
            size="small"
```

```
            placeholder="Search name or todo..."
```

```
            value={search}
```

```
            onChange={(e) => setSearch(e.target.value)}
```

```
            sx={{ mb: 2, width: 300, maxWidth: "90%" }} // set desired width
```

```
    />
```

```
);
```

```
}
```

pages/AssignmentGroupingMaster.js

```
import React, { useState } from "react";
```

```
import { Container, Typography } from "@mui/material";
```

```
import AssignmentForm from "../components/AssignmentForm";
```

```
import SearchBar from "../components/SearchBar";
```

```
import AssignmentTable from "../components/AssignmentTable";
```

```
import axios from "axios";
```

```
export default function AssignmentGroupingMaster() {  
  
  const [rows, setRows] = useState([]);  
  
  const [search, setSearch] = useState("");  
  
  /* ADD row */  
  
  const handleSubmit = (row) => {  
  
    setRows((prev) => [...prev, row]);  
  };  
  
  /* DELETE row (backend + frontend) */  
  
  const deleteRow = async (id) => {  
  
    try {  
  
      await axios.delete(`http://localhost:5000/api/assignments/${id}`);  
  
      setRows((prev) => prev.filter((r) => r._id !== id)); // ✓ FIXED  
    } catch (err) {  
  
      console.error(err);  
  
      alert("Delete failed");  
    }  
  };  
  
  const filteredRows = rows.filter(  
    (r) =>  
      r.name.toLowerCase().includes(search.toLowerCase()) ||  
      r.todo.toLowerCase().includes(search.toLowerCase())  
  );  
  
  return (  
    <Container maxWidth="lg">
```

```
<Typography variant="h5" align="center" sx={{ my: 2 }}>
```

```
Assignment Grouping Master
```

```
</Typography>
```

```
<AssignmentForm onSubmit={handleSubmit} />
```

```
<SearchBar search={search} setSearch={setSearch} />
```

```
{/* ✅ PASS setRows */}
```

```
<AssignmentTable
```

```
rows={filteredRows}
```

```
setRows={setRows}
```

```
onDelete={deleteRow}
```

```
/>
```

```
</Container>
```

```
);
```

```
}
```

App.js

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
```

```
import AssignmentGroupingMaster from "./pages/AssignmentGroupingMaster";
```

```
function App() {
```

```
return (
```

```
<BrowserRouter>
```

```
<Routes>
```

```
<Route path="/" element={<AssignmentGroupingMaster />} />
```

```
</Routes>
```

```
</BrowserRouter>
```

```
);
```

```
}
```

```
export default App;
```

Assignment Grouping Master

+

Name	Date	Todo	File	Actions
Atchaya	2025-12-30	doing react	<input type="button" value="VIEW"/>	○ ✎ ✖

1 row selected1–1 of 1< >

Assignment Grouping Master

+

Name	Date	Todo	File	Actions
Atchaya	2025-12-30	doing react	<input type="button" value="VIEW"/>	○ ✎ ✖

1 row selected1–1 of 1< >

Assignment Grouping Master

Name Todo Date +

Todo Preview

doing testing

Name	Date	Todo	File	Actions
Atchaya	2025-12-30	doing react	VIEW	

1 row selected

1–1 of 1 < >

Assignment Grouping Master

Name Todo Date +

Todo Preview

doing testing

Name	Date	Todo	File	Actions
Atchaya	2025-12-30	doing react	VIEW	

1–1 of 1 < >

