

cat.js

```
1 export const categories = ['Food', 'Transport', 'Utilities', 'Entertainment', 'Other'];  
2
```

```

1 // This file wraps IndexedDB with Promise-based functions for cost management.
2
3 /**
4  * Opens or creates the IndexedDB.
5  * @returns {Promise<IDBDatabase>}
6  */
7 function openDB() {
8     return new Promise((resolve, reject) => {
9         const request = indexedDB.open('costManagerDB', 1);
10
11         request.onupgradeneeded = (event) => {
12             const db = event.target.result;
13             if (!db.objectStoreNames.contains('costs')) {
14                 db.createObjectStore('costs', {keyPath: 'id', autoIncrement: true});
15             }
16         };
17
18         request.onsuccess = () => resolve(request.result);
19         request.onerror = () => reject(request.error);
20     });
21 }
22
23 /**
24  * Adds a new cost item to the 'costs' store.
25  * @param {Object} costData - The cost data to be added.
26  * @param {number} costData.sum - The sum of the cost.
27  * @param {string} costData.category - The category of the cost.
28  * @param {string} costData.description - The description of the cost.
29  * @param {string} costData.date - The date of the cost.
30  * @returns {Promise<number>} A promise that resolves to the ID of the added cost item.
31  */
32 export function addCost(costData) {
33     return new Promise(async (resolve, reject) => {
34         try {
35             const db = await openDB();
36             const tx = db.transaction('costs', 'readwrite');
37             const store = tx.objectStore('costs');
38             const request = store.add({
39                 sum: costData.sum,
40                 category: costData.category,
41                 description: costData.description,

```

idb.js

```
42         date: costData.date,
43     });
44     request.onsuccess = () => resolve(request.result);
45     request.onerror = () => reject(request.error);
46 } catch (error) {
47     reject(error);
48 }
49 });
50 }
51
52 /**
53  * Retrieves all costs for a specific month and year.
54  * @param {number} month - The month for which to retrieve costs (1-12).
55  * @param {number} year - The year for which to retrieve costs.
56  * @returns {Promise<Array<Object>>}} A promise that resolves to an array of cost items for the specified month and year.
57  */
58 export function getCostsByMonthYear(month, year) {
59     return new Promise(async (resolve, reject) => {
60         try {
61             const db = await openDB();
62             const tx = db.transaction('costs', 'readonly');
63             const store = tx.objectStore('costs');
64             const costs = [];
65             store.openCursor().onsuccess = (event) => {
66                 const cursor = event.target.result;
67                 if (cursor) {
68                     const cost = cursor.value;
69                     const costDate = new Date(cost.date);
70                     if (
71                         costDate.getMonth() + 1 === month &&
72                         costDate.getFullYear() === year
73                     ) {
74                         costs.push(cost);
75                     }
76                     cursor.continue();
77                 } else {
78                     resolve(costs);
79                 }
80             };
81         } catch (error) {
82             reject(error);
83         }
84     });
85 }
```

idb.js

```
83     }  
84     });  
85 }  
86
```

# App.css

```
1 body {
2   margin: 0;
3   padding: 0;
4   /* Instead of center, set align-items to flex-start */
5   display: flex;
6   align-items: flex-start;
7   justify-content: center;
8   min-height: 100vh;
9 }
10
11 body.light-mode {
12   background-color: rgb(237, 235, 235);
13   color: black;
14 }
15
16 body.dark-mode {
17   background-color: #222222;
18   color: rgb(255, 241, 241);
19 }
20
21 :root {
22   font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
23   line-height: 1.5;
24   font-weight: 400;
25
26   color-scheme: light dark;
27   color: rgba(255, 255, 255, 0.87);
28   background-color: #242424;
29
30   font-synthesis: none;
31   text-rendering: optimizeLegibility;
32   -webkit-font-smoothing: antialiased;
33   -moz-osx-font-smoothing: grayscale;
34 }
35
36 a {
37   font-weight: 500;
38   color: #646cff;
39   text-decoration: inherit;
40 }
41
```

## App.css

```
42 a:hover {
43     color: #535bf2;
44 }
45
46 h1 {
47     font-size: 3.2em;
48     line-height: 1.1;
49 }
50
51 button {
52     border-radius: 8px;
53     border: 1px solid transparent;
54     padding: 0.6em 1.2em;
55     font-size: 1em;
56     font-weight: 500;
57     font-family: inherit;
58     background-color: #1a1a1a;
59     cursor: pointer;
60     transition: border-color 0.25s;
61 }
62
63 button:hover {
64     border-color: #646cff;
65 }
66
67 button:focus,
68 button:focus-visible {
69     outline: 4px auto -webkit-focus-ring-color;
70 }
71
72 @media (prefers-color-scheme: light) {
73     :root {
74         color: #213547;
75         background-color: #ffffff;
76     }
77
78     a:hover {
79         color: #747bff;
80     }
81
82     button {
```

## App.css

```
83     background-color: #f9f9f9;
84 }
85 }
86
```

# app.jsx

```
1 import React, {useState, useEffect} from 'react';
2 import CostForm from './components/cost-form';
3 import Report from './components/report';
4 import './App.css';
5 import {ThemeProvider} from '@mui/material/styles';
6 import {
7   CssBaseline,
8   Container,
9   Typography,
10  Button,
11  Box,
12  IconButton,
13 } from '@mui/material';
14 import {Brightness4, Brightness7} from '@mui/icons-material';
15 import {lightTheme, darkTheme} from './theme';
16
17 /**
18  * Main application component for the Cost Manager.
19  * Handles theme switching and view switching between adding costs and viewing reports.
20  *
21  * @component
22  */
23 function App() {
24   // State to manage the current view ('add' or 'report')
25   const [view, setView] = useState('add');
26   // State to manage the current theme mode ('light' or 'dark')
27   const [themeMode, setThemeMode] = useState('dark');
28
29   useEffect(() => {
30     document.body.className =
31       themeMode === 'light' ? 'light-mode' : 'dark-mode';
32   }, [themeMode]);
33
34   return (
35     <ThemeProvider theme={themeMode === 'light' ? lightTheme : darkTheme}>
36       <CssBaseline/>
37       <Container maxWidth="md">
38         <Box
39           sx={{
40             display: 'flex',
41             flexDirection: 'column',
```



app.jsx

```
42         alignItems: 'center',
43         mt: 4,
44     }}
45     >
46     <Typography variant="h3" component="h1" gutterBottom>
47         Cost Manager
48     </Typography>
49
50     <Box sx={{mb: 4, display: 'flex', gap: 2}}>
51         <Button
52             variant="contained"
53             color="primary"
54             onClick={() => setView('add')}
55         >
56             Add Cost
57         </Button>
58         <Button
59             variant="contained"
60             color="secondary"
61             onClick={() => setView('report')}
62         >
63             Monthly Report
64         </Button>
65         <IconButton
66             onClick={() =>
67                 setThemeMode((prev) => (prev === 'light' ? 'dark' : 'light'))
68             }
69             color="inherit"
70         >
71             {themeMode === 'light' ? <Brightness4/> : <Brightness7/>}
72         </IconButton>
73     </Box>
74
75     {view === 'add' && <CostForm/>}
76     {view === 'report' && <Report/>}
77 </Box>
78 </Container>
79 </ThemeProvider>
80 );
81 }
82
```

app.jsx

```
83 export default App;  
84
```

# main.jsx

```
1 import {StrictMode} from 'react';
2 import {createRoot} from 'react-dom/client';
3 import './App.css';
4 import App from './app.jsx';
5
6 createRoot(document.getElementById('root')).render(
7   <StrictMode>
8     <App/>
9   </StrictMode>
10 );
11
```

theme.js

```
1 import {createTheme} from '@mui/material/styles';
2
3 /**
4  * Light theme configuration for the application.
5  *
6  */
7 const lightTheme = createTheme({
8   palette: {
9     mode: 'light',
10    primary: {
11      main: '#1976d2',
12    },
13    secondary: {
14      main: '#dc004e',
15    },
16  },
17 });
18
19 /**
20  * Dark theme configuration for the application.
21  *
22  * @type @type {Theme}
23  */
24 const darkTheme = createTheme({
25   palette: {
26     mode: 'dark',
27     primary: {
28       main: '#1976d2',
29     },
30     secondary: {
31       main: '#dc004e',
32     },
33   },
34 });
35
36 export {lightTheme, darkTheme};
37
```

# report.jsx

```
1 import React, {useState} from 'react';
2 import {getCostsByMonthYear} from '../db/idb';
3 import {Pie} from 'react-chartjs-2';
4 import {categories} from '../db/cat';
5 import {Chart as ChartJS} from 'chart.js/auto';
6 import {
7   TextField,
8   Button,
9   Box,
10  Paper,
11  Typography,
12  Table,
13  TableBody,
14  TableCell,
15  TableContainer,
16  TableHead,
17  TableRow,
18  TableFooter
19 } from '@mui/material';
20
21 /**
22  * Report component that displays a monthly report of costs by category.
23  * It includes a form to select the month and year, a table to display the report,
24  * and a pie chart to visualize the data.
25  *
26  * @component
27  */
28 function Report() {
29
30   const initReport = {};
31   categories.forEach((category) => {
32     initReport[category] = {
33       count: 0,
34       total: 0,
35       instances: [],
36     };
37   });
38
39   const [monthYear, setMonthYear] = useState('');
40   const [categoryReport, setCategoryReport] = useState(initReport);
41   const [chartData, setChartData] = useState(null);
```

## report.jsx

```
42     const [totalCount, setTotalCount] = useState(0);
43     const [totalSum, setTotalSum] = useState(0.0);
44
45     const handleFetchCosts = async () => {
46         if (monthYear) {
47             const [year, month] = monthYear.split('-');
48             const fetchedCosts = await getCostsByMonthYear(
49                 parseInt(month),
50                 parseInt(year)
51             );
52
53             const report = {...initReport}; // Copy initial empty report
54             let count = 0;
55             let sum = 0.0;
56
57             fetchedCosts.forEach((cost) => {
58                 if (categories.includes(cost.category)) {
59                     const rep = report[cost.category];
60                     rep['count']++;
61                     rep['total'] += parseFloat(cost.sum);
62                     rep['instances'].push(cost);
63
64                     count++;
65                     sum += parseFloat(cost.sum);
66                 }
67             });
68
69             setCategoryReport(report);
70             setTotalCount(count);
71             setTotalSum(sum);
72
73
74             setChartData({
75                 labels: categories,
76                 datasets: [
77                     {
78                         label: 'Total costs for month',
79                         data: categories.map((category) => report[category]['total']),
80                         backgroundColor: [
81                             '#FF6384',
```

# report.jsx

```

83         '#36A2EB',
84         '#FFCE56',
85         '#8B008B',
86         '#00FF00',
87     ],
88 },
89 ],
90 });
91 }
92 };
93
94
95 return (
96     <Paper elevation={3} sx={{p: 4, borderRadius: 2}}>
97         <Typography variant="h5" gutterBottom align="center" sx={{mb: 4}}>
98             Monthly Report
99         </Typography>
100
101         <Box sx={{mb: 4, display: 'flex', gap: 2}}>
102             <TextField
103                 label="Month"
104                 type="month"
105                 value={monthYear}
106                 onChange={(e) => setMonthYear(e.target.value)}
107                 variant="outlined"
108                 placeholder={'YYYY-MM'}
109                 fullWidth
110                 slotProps={{inputLabel: {shrink: true}}}
111             />
112             <Button
113                 variant="contained"
114                 color="primary"
115                 onClick={handleFetchCosts}
116             >
117                 Get Report
118             </Button>
119         </Box>
120
121         <Box sx={{display: 'flex', gap: 4}}>
122             <TableContainer component={Paper} sx={{flex: 1}}>
123                 <Table>

```

```

124     <TableHead>
125       <TableRow>
126         <TableCell>Category</TableCell>
127         <TableCell>Count</TableCell>
128         <TableCell>Total</TableCell>
129       </TableRow>
130     </TableHead>
131
132     <TableBody>
133       {categories.map((category) => (
134         <React.Fragment key={category}>
135           <TableRow key={category}>
136             <TableCell>{category}</TableCell>
137             <TableCell>{categoryReport[category]['count']}</TableCell>
138             <TableCell>{categoryReport[category]['total']}</TableCell>
139           </TableRow>
140
141           {categoryReport[category]['instances'].length > 0 && <TableRow>
142             <TableCell colspan={3}>
143               <Table size="small">
144                 <TableHead>
145                   <TableRow>
146                     <TableCell>Description</TableCell>
147                     <TableCell>Sum</TableCell>
148                     <TableCell>Day</TableCell>
149                   </TableRow>
150                 </TableHead>
151                 <TableBody>
152                   {categoryReport[category]['instances'].map((cost) => (
153                     <TableRow key={cost.id}>
154                       <TableCell>{cost.description}</TableCell>
155                       <TableCell>{cost.sum}</TableCell>
156                       <TableCell>
157                         {cost.date.substring(cost.date.length - 2)}
158                       </TableCell>
159                     </TableRow>
160                   )))}
161                 </TableBody>
162               </Table>
163             </TableCell>
164           </TableRow>}]

```



# report.jsx

```
165         </React.Fragment>
166     )})
167 </TableBody>
168 <TableFooter>
169     <TableRow>
170         <TableCell>Total</TableCell>
171         <TableCell>{totalCount}</TableCell>
172         <TableCell>{totalSum}</TableCell>
173     </TableRow>
174 </TableFooter>
175 </Table>
176 </TableContainer>
177 {totalCount > 0 && ( // Only show chart if there are costs
178     <Box sx={{flex: 1}}>
179         <Pie data={chartData} options={{plugins: {legend: {position: 'bottom'}}}}/>
180     </Box>
181 )}
182 </Box>
183 </Paper>
184 );
185 }
186
187 export default Report;
```

## cost-form.jsx

```
1 import React, {useState} from 'react';
2 import {addCost} from '../db/idb';
3 import {categories} from '../db/cat';
4 import {
5   TextField,
6   Button,
7   Box,
8   Paper,
9   Typography,
10  Snackbar,
11  Alert,
12  Select,
13  MenuItem,
14  FormControl,
15  InputLabel
16 } from '@mui/material';
17
18 /**
19  * CostForm component allows users to add a new cost entry.
20  * It includes form fields for sum, category, description, and date.
21  * On form submission, the data is saved and a success alert is shown.
22  *
23  * @component
24  */
25 function CostForm() {
26   // State variables for form inputs and alert
27   const [sum, setSum] = useState('');
28   const [category, setCategory] = useState('');
29   const [description, setDescription] = useState('');
30   const [date, setDate] = useState('');
31   const [openAlert, setOpenAlert] = useState(false);
32
33   // Handle form submission
34   const handleSubmit = async (event) => {
35     event.preventDefault();
36     await addCost({sum, category, description, date});
37     // Reset form fields
38     setSum('');
39     setCategory('');
40     setDescription('');
41     setDate('');
```

## cost-form.jsx

```
42     setOpenAlert(true); // Show success alert
43   };
44
45   return (
46     <Paper elevation={3} sx={{p: 4, borderRadius: 2}}>
47       <Typography variant="h5" gutterBottom align="center" sx={{mb: 4}}>
48         Add New Cost
49       </Typography>
50
51       <Box
52         component="form"
53         onSubmit={handleSubmit}
54         sx={{
55           display: 'flex',
56           flexDirection: 'column',
57           gap: 3,
58         }}
59       >
60         {/* Input field for sum */}
61         <TextField
62           label="Sum"
63           type="number"
64           value={sum}
65           onChange={(e) => setSum(e.target.value)}
66           required
67           variant="outlined"
68         />
69
70         {/* Dropdown menu for category selection */}
71         <FormControl fullWidth>
72           <InputLabel>Category</InputLabel>
73           <Select
74             label="category"
75             value={category}
76             onChange={(e) => setCategory(e.target.value)}
77             required
78             variant="outlined">
79             {categories.map((category) => (
80               <MenuItem key={category} value={category}>
81                 {category}
82               </MenuItem>
```

```
83         )})
84       </Select>
85     </FormControl>
86
87     {/* Input field for description */}
88     <TextField
89       label="Description"
90       value={description}
91       onChange={(e) => setDescription(e.target.value)}
92       required
93       variant="outlined"
94       multiline
95       rows={2}
96     />
97
98     {/* Input field for date */}
99     <TextField
100       type="date"
101       value={date}
102       onChange={(e) => setDate(e.target.value)}
103       required
104       variant="outlined"
105     />
106
107     {/* Submit button */}
108     <Button
109       type="submit"
110       variant="contained"
111       color="primary"
112       size="large"
113       sx={{mt: 2}}
114     >
115       Add Cost
116     </Button>
117   </Box>
118
119   {/* Success alert */}
120   <Snackbar
121     open={openAlert}
122     autoHideDuration={6000}
123     onClose={() => setOpenAlert(false)}
```

## cost-form.jsx

```
124 >
125 <Alert severity="success" sx={{width: '100%'}}>
126   Cost added successfully!
127 </Alert>
128 </Snackbar>
129 </Paper>
130 );
131 }
132
133 export default CostForm;
```