cat.js

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

```

idb.js

```
// This file wraps IndexedDB with Promise-based functions for cost management.

/**
 * Opens or creates the IndexedDB.
 * @returns {Promise<IDBDatabase>}
 */
function openDB() {
    return new Promise((resolve, reject) => {
        const request = indexedDB.open('costManagerDB', 1);

        request.onupgradeneeded = (event) => {
            const db = event.target.result;
            if (!db.objectStoreNames.contains('costs')) {
                db.createObjectStore('costs', {keyPath: 'id', autoIncrement: true});
            }
        };

        request.onsuccess = () => resolve(request.result);
        request.onerror = () => reject(request.error);
    });
}

/**
 * Adds a new cost item to the 'costs' store.
 * @param {Object} costData - The cost data to be added.
 * @param {number} costData.sum - The sum of the cost.
 * @param {string} costData.category - The category of the cost.
 * @param {string} costData.description - The description of the cost.
 * @param {string} costData.date - The date of the cost.
 * @returns {Promise<number>} A promise that resolves to the ID of the added cost item.
 */
export function addCost(costData) {
    return new Promise(async (resolve, reject) => {
        try {
            const db = await openDB();
            const tx = db.transaction('costs', 'readwrite');
            const store = tx.objectStore('costs');
            const request = store.add({
                sum: costData.sum,
                category: costData.category,
                description: costData.description,
                date: costData.date,
            });
            request.onsuccess = () => resolve(request.result);
            request.onerror = () => reject(request.error);
        } catch (error) {
            reject(error);
        }
    });
}

/**
 * Retrieves all costs for a specific month and year.
 * @param {number} month - The month for which to retrieve costs (1-12).
 * @param {number} year - The year for which to retrieve costs.
 * @returns {Promise<Array<Object>>} A promise that resolves to an array of cost items for the specified month and year.
 */
export function getCostsByMonthYear(month, year) {
    return new Promise(async (resolve, reject) => {
        try {
            const db = await openDB();
            const tx = db.transaction('costs', 'readonly');
            const store = tx.objectStore('costs');
            const costs = [];
            store.openCursor().onsuccess = (event) => {
                const cursor = event.target.result;
                if (cursor) {
                    const cost = cursor.value;
                    const costDate = new Date(cost.date);
                    if (
                        costDate.getMonth() + 1 === month &&
                        costDate.getFullYear() === year
                    ) {
                        costs.push(cost);
                    }
                    cursor.continue();
                } else {
                    resolve(costs);
                }
            };
        } catch (error) {
            reject(error);
        }
    });
}

```

App.css

```css
body {
    margin: 0;
    padding: 0;
    /* Instead of center, set align-items to flex-start */
    display: flex;
    align-items: flex-start;
    justify-content: center;
    min-height: 100vh;
}

body.light-mode {
    background-color: rgb(237, 235, 235);
    color: black;
}

body.dark-mode {
    background-color: #222222;
    color: rgb(255, 241, 241);
}

:root {
    font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
    line-height: 1.5;
    font-weight: 400;

    color-scheme: light dark;
    color: rgba(255, 255, 255, 0.87);
    background-color: #242424;

    font-synthesis: none;
    text-rendering: optimizeLegibility;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}

a {
    font-weight: 500;
    color: #646cff;
    text-decoration: inherit;
}

```

```css
a:hover {
    color: #535bf2;
}

h1 {
    font-size: 3.2em;
    line-height: 1.1;
}

button {
    border-radius: 8px;
    border: 1px solid transparent;
    padding: 0.6em 1.2em;
    font-size: 1em;
    font-weight: 500;
    font-family: inherit;
    background-color: #1a1a1a;
    cursor: pointer;
    transition: border-color 0.25s;
}

button:hover {
    border-color: #646cff;
}

button:focus,
button:focus-visible {
    outline: 4px auto -webkit-focus-ring-color;
}

@media (prefers-color-scheme: light) {
    :root {
        color: #213547;
        background-color: #ffffff;
    }

    a:hover {
        color: #747bff;
    }

    button {
        background-color: #f9f9f9;
    }
}

```

app.jsx

```jsx
import React, {useState, useEffect} from 'react';
import CostForm from './components/cost-form';
import Report from './components/report';
import './App.css';
import {ThemeProvider} from '@mui/material/styles';
import {
    CssBaseline,
    Container,
    Typography,
    Button,
    Box,
    IconButton,
} from '@mui/material';
import {Brightness4, Brightness7} from '@mui/icons-material';
import {lightTheme, darkTheme} from './theme';

/**
 * Main application component for the Cost Manager.
 * Handles theme switching and view switching between adding costs and viewing reports.
 *
 * @component
 */
function App() {
    // State to manage the current view ('add' or 'report')
    const [view, setView] = useState('add');
    // State to manage the current theme mode ('light' or 'dark')
    const [themeMode, setThemeMode] = useState('dark');

    useEffect(() => {
        document.body.className =
            themeMode === 'light' ? 'light-mode' : 'dark-mode';
    }, [themeMode]);

    return (
        <ThemeProvider theme={themeMode === 'light' ? lightTheme : darkTheme}>
            <CssBaseline/>
            <Container maxWidth="md">
                <Box
                    sx={{
                        display: 'flex',
                        flexDirection: 'column',
```

app.jsx

```
                    alignItems: 'center',
                    mt: 4,
                }}
            >
                <Typography variant="h3" component="h1" gutterBottom>
                    Cost Manager
                </Typography>

                <Box sx={{mb: 4, display: 'flex', gap: 2}}>
                    <Button
                        variant="contained"
                        color="primary"
                        onClick={() => setView('add')}
                    >
                        Add Cost
                    </Button>
                    <Button
                        variant="contained"
                        color="secondary"
                        onClick={() => setView('report')}
                    >
                        Monthly Report
                    </Button>
                    <IconButton
                        onClick={() =>
                            setThemeMode((prev) => (prev === 'light' ? 'dark' : 'light'))
                        }
                        color="inherit"
                    >
                        {themeMode === 'light' ? <Brightness4/> : <Brightness7/>}
                    </IconButton>
                </Box>

                {view === 'add' && <CostForm/>}
                {view === 'report' && <Report/>}
            </Box>
        </Container>
    </ThemeProvider>
  );
}

```

app.jsx

```
export default App;

```

main.jsx

```jsx
import {StrictMode} from 'react';
import {createRoot} from 'react-dom/client';
import './App.css';
import App from './app.jsx';

createRoot(document.getElementById('root')).render(
    <StrictMode>
        <App/>
    </StrictMode>
);

```

theme.js

```js
import {createTheme} from '@mui/material/styles';

/**
 * Light theme configuration for the application.
 *
 */
const lightTheme = createTheme({
    palette: {
        mode: 'light',
        primary: {
            main: '#1976d2',
        },
        secondary: {
            main: '#dc004e',
        },
    },
});

/**
 * Dark theme configuration for the application.
 *
 * @type @type {Theme}
 */
const darkTheme = createTheme({
    palette: {
        mode: 'dark',
        primary: {
            main: '#1976d2',
        },
        secondary: {
            main: '#dc004e',
        },
    },
});

export {lightTheme, darkTheme};

```

report.jsx

```jsx
import React, {useState} from 'react';
import {getCostsByMonthYear} from '../db/idb';
import {Pie} from 'react-chartjs-2';
import {categories} from '../db/cat';
import {Chart as ChartJS} from 'chart.js/auto';
import {
    TextField,
    Button,
    Box,
    Paper,
    Typography,
    Table,
    TableBody,
    TableCell,
    TableContainer,
    TableHead,
    TableRow,
    TableFooter
} from '@mui/material';

/**
 * Report component that displays a monthly report of costs by category.
 * It includes a form to select the month and year, a table to display the report,
 * and a pie chart to visualize the data.
 *
 * @component
 */
function Report() {

    const initReport = {};
    categories.forEach((category) => {
        initReport[category] = {
            count: 0,
            total: 0,
            instances: [],
        };
    });

    const [monthYear, setMonthYear] = useState('');
    const [categoryReport, setCategoryReport] = useState(initReport);
    const [chartData, setChartData] = useState(null);
```

report.jsx

```
    const [totalCount, setTotalCount] = useState(0);
    const [totalSum, setTotalSum] = useState(0.0);

    const handleFetchCosts = async () => {
        if (monthYear) {
            const [year, month] = monthYear.split('-');
            const fetchedCosts = await getCostsByMonthYear(
                parseInt(month),
                parseInt(year)
            );

            const report = {...initReport}; // Copy initial empty report
            let count = 0;
            let sum = 0.0;


            fetchedCosts.forEach((cost) => {
                if (categories.includes(cost.category)) {
                    const rep = report[cost.category];
                    rep['count']++;
                    rep['total'] += parseFloat(cost.sum);
                    rep['instances'].push(cost);

                    count++;
                    sum += parseFloat(cost.sum);
                }
            });

            setCategoryReport(report);
            setTotalCount(count);
            setTotalSum(sum);


            setChartData({
                labels: categories,
                datasets: [
                    {
                        label: 'Total costs for month',
                        data: categories.map((category) => report[category]['total']),
                        backgroundColor: [
                            '#FF6384',
                            '#36A2EB',
                            '#FFCE56',
                            '#8B008B',
                            '#00FF00',
                        ],
                    },
                ],
            });
        }
    };


    return (
        <Paper elevation={3} sx={{p: 4, borderRadius: 2}}>
            <Typography variant="h5" gutterBottom align="center" sx={{mb: 4}}>
                Monthly Report
            </Typography>

            <Box sx={{mb: 4, display: 'flex', gap: 2}}>
                <TextField
                    label="Month"
                    type="month"
                    value={monthYear}
                    onChange={(e) => setMonthYear(e.target.value)}
                    variant="outlined"
                    placeholder={'YYYY-MM'}
                    fullWidth
                    slotProps={{inputLabel: {shrink: true}}}
                />
                <Button
                    variant="contained"
                    color="primary"
                    onClick={handleFetchCosts}
                >
                    Get Report
                </Button>
            </Box>

            <Box sx={{display: 'flex', gap: 4}}>
                <TableContainer component={Paper} sx={{flex: 1}}>
                    <Table>
```

report.jsx

```
                        <TableHead>
                            <TableRow>
                                <TableCell>Category</TableCell>
                                <TableCell>Count</TableCell>
                                <TableCell>Total</TableCell>
                            </TableRow>
                        </TableHead>

                        <TableBody>
                            {categories.map((category) => (
                                <React.Fragment key={category}>
                                    <TableRow key={category}>
                                        <TableCell>{category}</TableCell>
                                        <TableCell>{categoryReport[category]['count']}</TableCell>
                                        <TableCell>{categoryReport[category]['total']}</TableCell>
                                    </TableRow>

                                    {categoryReport[category]['instances'].length > 0 && <TableRow>
                                        <TableCell colSpan={3}>
                                            <Table size="small">
                                                <TableHead>
                                                    <TableRow>
                                                        <TableCell>Description</TableCell>
                                                        <TableCell>Sum</TableCell>
                                                        <TableCell>Day</TableCell>
                                                    </TableRow>
                                                </TableHead>
                                                <TableBody>
                                                    {categoryReport[category]['instances'].map((cost) => (
                                                        <TableRow key={cost.id}>
                                                            <TableCell>{cost.description}</TableCell>
                                                            <TableCell>{cost.sum}</TableCell>
                                                            <TableCell>
                                                                {cost.date.substring(cost.date.length - 2)}
                                                            </TableCell>
                                                        </TableRow>
                                                    ))}
                                                </TableBody>
                                            </Table>
                                        </TableCell>
                                    </TableRow>}
```

```jsx
                                </React.Fragment>
                            ))}
                        </TableBody>
                        <TableFooter>
                            <TableRow>
                                <TableCell>Total</TableCell>
                                <TableCell>{totalCount}</TableCell>
                                <TableCell>{totalSum}</TableCell>
                            </TableRow>
                        </TableFooter>
                    </Table>
                </TableContainer>
                {totalCount > 0 && ( // Only show chart if there are costs
                    <Box sx={{flex: 1}}>
                        <Pie data={chartData} options={{plugins: {legend: {position: 'bottom'}}}}/>
                    </Box>
                )}
            </Box>
        </Paper>
    );
}

export default Report;
```

cost-form.jsx

```jsx
import React, {useState} from 'react';
import {addCost} from '../db/idb';
import {categories} from '../db/cat';
import {
    TextField,
    Button,
    Box,
    Paper,
    Typography,
    Snackbar,
    Alert,
    Select,
    MenuItem,
    FormControl,
    InputLabel
} from '@mui/material';

/**
 * CostForm component allows users to add a new cost entry.
 * It includes form fields for sum, category, description, and date.
 * On form submission, the data is saved and a success alert is shown.
 *
 * @component
 */
function CostForm() {
    // State variables for form inputs and alert
    const [sum, setSum] = useState('');
    const [category, setCategory] = useState('');
    const [description, setDescription] = useState('');
    const [date, setDate] = useState('');
    const [openAlert, setOpenAlert] = useState(false);

    // Handle form submission
    const handleSubmit = async (event) => {
        event.preventDefault();
        await addCost({sum, category, description, date});
        // Reset form fields
        setSum('');
        setCategory('');
        setDescription('');
        setDate('');
```

cost-form.jsx

```jsx
        setOpenAlert(true); // Show success alert
    };

    return (
        <Paper elevation={3} sx={{p: 4, borderRadius: 2}}>
            <Typography variant="h5" gutterBottom align="center" sx={{mb: 4}}>
                Add New Cost
            </Typography>

            <Box
                component="form"
                onSubmit={handleSubmit}
                sx={{
                    display: 'flex',
                    flexDirection: 'column',
                    gap: 3,
                }}
            >
                {/* Input field for sum */}
                <TextField
                    label="Sum"
                    type="number"
                    value={sum}
                    onChange={(e) => setSum(e.target.value)}
                    required
                    variant="outlined"
                />

                {/* Dropdown menu for category selection */}
                <FormControl fullWidth>
                    <InputLabel>Category</InputLabel>
                    <Select
                        label="category"
                        value={category}
                        onChange={(e) => setCategory(e.target.value)}
                        required
                        variant="outlined">
                        {categories.map((category) => (
                            <MenuItem key={category} value={category}>
                                {category}
                            </MenuItem>
                        ))}
                    </Select>
                </FormControl>

                {/* Input field for description */}
                <TextField
                    label="Description"
                    value={description}
                    onChange={(e) => setDescription(e.target.value)}
                    required
                    variant="outlined"
                    multiline
                    rows={2}
                />

                {/* Input field for date */}
                <TextField
                    type="date"
                    value={date}
                    onChange={(e) => setDate(e.target.value)}
                    required
                    variant="outlined"
                />

                {/* Submit button */}
                <Button
                    type="submit"
                    variant="contained"
                    color="primary"
                    size="large"
                    sx={{mt: 2}}
                >
                    Add Cost
                </Button>
            </Box>

            {/* Success alert */}
            <Snackbar
                open={openAlert}
                autoHideDuration={6000}
                onClose={() => setOpenAlert(false)}
```

cost-form.jsx

```jsx
                >
                    <Alert severity="success" sx={{width: '100%'}}>
                        Cost added successfully!
                    </Alert>
                </Snackbar>
            </Paper>
        );
}

export default CostForm;
```