DAY 6 – State Management, Calculations, Filters, Categorization & Expense Tracker (Full Detailed Notes)

---

1. What Is State Management?

Simple Definition

State = The data that changes the UI when updated.

React updates UI automatically when state changes.

Examples of state:

- Input values
- Lists
- Toggle states
- Counters
- Selected items
- Active menu/category

---

2. Why State Is Important?

✓ Controls values ✓ Handles updates ✓ Manages UI changes ✓ Stores dynamic data ✓ Makes the app interactive ✓ Helps create real projects

---

3. Expense Tracker – Why This Project?

This app teaches:

✓ Form handling ✓ Array operations ✓ Adding items ✓ Removing items ✓ Calculating totals ✓ Filtering by category ✓ Conditional rendering ✓ Component interaction

It's a real-world app — students love it.

---

4. Expense Data Structure (Important)

Each expense contains:

```
{
  id: 1,
  title: "Groceries",
  amount: 500,
  category: "Food"
}
```

This structure helps with:

- Listing
- Filtering
- Displaying
- Calculating totals

---

5. Step 1 — Define States

```
const [title, setTitle] = useState("");
const [amount, setAmount] = useState("");
const [category, setCategory] = useState("Food");

const [expenses, setExpenses] = useState([]);
```

---

6. Step 2 — Controlled Inputs for Form

Title Input

```
<input
  value={title}
  onChange={(e) => setTitle(e.target.value)}
  placeholder="Expense Title"
/>
```

Amount Input

```
<input
  type="number"
  value={amount}
  onChange={(e) => setAmount(e.target.value)}
  placeholder="Amount"
/>
```

Category Dropdown

```
<select value={category} onChange={(e) => setCategory(e.target.value)}>
  <option value="Food">Food</option>
  <option value="Travel">Travel</option>
  <option value="Shopping">Shopping</option>
  <option value="Bills">Bills</option>
  <option value="Other">Other</option>
</select>
```

---

7. Step 3 — Add Expense Function

```
function addExpense() {
  if (!title.trim() || !amount.trim()) return;

  const newExpense = {
    id: Date.now(),
    title,
    amount: Number(amount),
    category
  };

  setExpenses([...expenses, newExpense]);

  // clear input fields
  setTitle("");
  setAmount("");
  setCategory("Food");
}
```

Key Concepts:

- `Date.now()` used for ID
- Spread operator to create new array
- Resetting form after adding

---

8. Step 4 — Display All Expenses

```
{expenses.map(exp => (
  <div key={exp.id} className="expense-item">
    <span>{exp.title}</span>
```

```
      <span>₹{exp.amount}</span>
      <span>{exp.category}</span>
    </div>
  ))}
```

---

9. Step 5 — Delete an Expense

```
function deleteExpense(id) {
  setExpenses(expenses.filter(e => e.id !== id));
}
```

Button:

```
<button onClick={() => deleteExpense(exp.id)}>Delete</button>
```

---

10. Step 6 — Calculate Total Amount

Using `reduce()`

```
const total = expenses.reduce((sum, exp) => sum + exp.amount, 0);
```

Display:

```
<h2>Total: ₹{total}</h2>
```

---

11. Step 7 — Filter Expenses by Category

Setup filter state:

```
const [filter, setFilter] = useState("All");
```

UI:

```
<select value={filter} onChange={(e) => setFilter(e.target.value)}>
  <option value="All">All</option>
  <option value="Food">Food</option>
  <option value="Travel">Travel</option>
  <option value="Shopping">Shopping</option>
  <option value="Bills">Bills</option>
</select>
```

Apply filter:

```
const filteredExpenses =
  filter === "All"
    ? expenses
    : expenses.filter(exp => exp.category === filter);
```

Display filtered:

```
{filteredExpenses.map(...)}
```

---

12. Step 8 — Breakdown Per Category (Advanced Feature)

Using reduce:

```
const categoryTotals = expenses.reduce((acc, exp) => {
  acc[exp.category] = (acc[exp.category] || 0) + exp.amount;
  return acc;
}, {});
```

Display:

```
{Object.entries(categoryTotals).map(([cat, amt]) => (
  <p key={cat}>{cat}: ₹{amt}</p>
))}
```

---

13. Step 9 — Beautiful UI Layout (CSS)

```
.expense-container {
  width: 400px;
  margin: auto;
  padding: 20px;
  background: #f3f3f3;
  border-radius: 10px;
}

.expense-item {
  display: flex;
  justify-content: space-between;
  padding: 10px;
  background: white;
  margin: 5px 0;
  border-radius: 5px;
}
```

---

14. Full Expense Tracker Code (Everything Combined)

```
function ExpenseTracker() {
  const [title, setTitle] = useState("");
  const [amount, setAmount] = useState("");
  const [category, setCategory] = useState("Food");
  const [expenses, setExpenses] = useState([]);
  const [filter, setFilter] = useState("All");

  function addExpense() {
    if (!title.trim() || !amount.trim()) return;

    const newExpense = {
      id: Date.now(),
      title,
      amount: Number(amount),
      category
    };

    setExpenses([...expenses, newExpense]);
    setTitle("");
    setAmount("");
    setCategory("Food");
  }

  function deleteExpense(id) {
    setExpenses(expenses.filter(e => e.id !== id));
  }
  const filteredExpenses =
    filter === "All"
      ? expenses
      : expenses.filter(e => e.category === filter);

  const total = filteredExpenses.reduce(
    (sum, exp) => sum + exp.amount,
    0
  );

  return (
    <div className="expense-container">
      <h1>Expense Tracker</h1>

      <input
        value={title}
        onChange={(e) => setTitle(e.target.value)}
        placeholder="Title"
      />

      <input
        type="number"
        value={amount}
        onChange={(e) => setAmount(e.target.value)}
        placeholder="Amount"
      />

      <select value={category} onChange={(e) => setCategory(e.target.value)}>
        <option>Food</option>
        <option>Travel</option>
        <option>Shopping</option>
        <option>Bills</option>
        <option>Other</option>
      </select>

      <button onClick={addExpense}>Add Expense</button>

      <h2>Filter by Category</h2>
      <select value={filter} onChange={(e) => setFilter(e.target.value)}>
        <option>All</option>
        <option>Food</option>
        <option>Travel</option>
        <option>Shopping</option>
        <option>Bills</option>
        <option>Other</option>
      </select>

      <h2>Total: ₹{total}</h2>

      {filteredExpenses.map(exp => (
        <div key={exp.id} className="expense-item">
          <span>{exp.title}</span>
          <span>₹{exp.amount}</span>
          <span>{exp.category}</span>
          <button onClick={() => deleteExpense(exp.id)}>Delete</button>
        </div>
      ))}
    </div>
  );
}
```
