Project Name: Micro Collection Partner (MCP) System

Codes:

App.js file :-

```
import logo from "./logo.svg";
import "./App.css";

function App() {
  return (
    <div className="App">
      <h1 style={{ padding: "20px" }}>Welcome to MCP Dashboard</h1>

    </div>
  );
}

export default App;
```

MCPDashboard.js file :-

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

const MCPDashboard = () => {
  const [walletBalance, setWalletBalance] = useState(10000);
```

```
const [amount, setAmount] = useState("");
const [selectedPartner, setSelectedPartner] = useState(1);
const [topUpAmount, setTopUpAmount] = useState("");
const [transactionHistory, setTransactionHistory] = useState([]);

const [pickupPartners, setPickupPartners] = useState([
 {
  id: 1,
  name: "Rahul",
  status: "Active",
  orders: { total: 10, completed: 7, pending: 3 },
  wallet: 500,
  role: "Collector",
  commission: 10,
 },
 {
  id: 2,
  name: "Anjali",
  status: "Inactive",
  orders: { total: 5, completed: 5, pending: 0 },
  wallet: 300,
  role: "Supervisor",
  commission: 15,
 },
```

```
]);

const [newPartner, setNewPartner] = useState({
  name: "",
  role: "Collector",
  commission: 0,
});

const handleTopUp = () => {
  const amt = parseInt(topUpAmount || 0);
  if (amt > 0) {
    setWalletBalance(walletBalance + amt);
    setTransactionHistory([
      ...transactionHistory,
      {
        type: "Top-up",
        amount: amt,
        target: "MCP Wallet",
        time: new Date().toLocaleString(),
      },
    ]);
    setTopUpAmount("");
  }
};
```

```
const handleFundChange = (id, type) => {
  const amt = parseInt(amount || 0);
  if (amt <= 0) return;

  const updatedPartners = pickupPartners.map((partner) => {
    if (partner.id === id) {
      const newWallet =
        type === "add"
          ? partner.wallet + amt
          : Math.max(partner.wallet - amt, 0);
      return { ...partner, wallet: newWallet };
    }
    return partner;
  });

  setPickupPartners(updatedPartners);

  if (type === "add") {
    setWalletBalance(walletBalance - amt);
  } else {
    setWalletBalance(walletBalance + amt);
  }
```

```
    setTransactionHistory([
      ...transactionHistory,
      {
        type: type === "add" ? "Added to Partner" : "Deducted from
Partner",
        amount: amt,
        target: pickupPartners.find((p) => p.id === id).name,
        time: new Date().toLocaleString(),
      },
    ]);

    setAmount("");
  };

  const toggleStatus = (id) => {
    const updatedPartners = pickupPartners.map((partner) => {
      if (partner.id === id) {
        return {
          ...partner,
          status: partner.status === "Active" ? "Inactive" : "Active",
        };
      }
      return partner;
    });
```

```
  setPickupPartners(updatedPartners);
};

const handleAddPartner = () => {
  if (!newPartner.name || newPartner.commission < 0) return;
  const newId = pickupPartners.length + 1;
  const newEntry = {
    id: newId,
    name: newPartner.name,
    status: "Active",
    orders: { total: 0, completed: 0, pending: 0 },
    wallet: 0,
    role: newPartner.role,
    commission: newPartner.commission,
  };
  setPickupPartners([...pickupPartners, newEntry]);
  setNewPartner({ name: "", role: "Collector", commission: 0 });
};

const deletePartner = (id) => {
  setPickupPartners(pickupPartners.filter((p) => p.id !== id));
};

return (
```

```
    <div style={{ padding: "20px", fontFamily: "Arial" }}>
      <h1>MCP Dashboard</h1>

      <div>
        <p>
          <strong>MCP Wallet Balance:</strong> ₹{walletBalance}
        </p>
        <input
          type="number"
          value={topUpAmount}
          onChange={(e) => setTopUpAmount(e.target.value)}
          placeholder="Add to MCP Wallet"
        />
        <button onClick={handleTopUp}>Top-Up Wallet</button>
      </div>

      {/* Add Partner */}
      <h2>Add Pickup Partner</h2>
      <input
        type="text"
        placeholder="Name"
        value={newPartner.name}
        onChange={(e) => setNewPartner({ ...newPartner, name:
e.target.value })}
```

```
    />
    <select
      value={newPartner.role}
      onChange={(e) => setNewPartner({ ...newPartner, role:
e.target.value })}
    >
      <option value="Collector">Collector</option>
      <option value="Supervisor">Supervisor</option>
    </select>
    <input
      type="number"
      placeholder="Commission / Order"
      value={newPartner.commission}
      onChange={(e) =>
        setNewPartner({
          ...newPartner,
          commission: parseInt(e.target.value || 0),
        })
      }
    />
    <button onClick={handleAddPartner}>Add Partner</button>

    <h2>Pickup Partners</h2>
    {pickupPartners.map((partner) => {
```

```
const completedPercent =
  (partner.orders.completed / partner.orders.total) * 100 || 0;

return (
  <div
    key={partner.id}
    style={{
      border: "1px solid #ccc",
      padding: "10px",
      marginBottom: "15px",
    }}
  >
    <p>
      <strong>Name:</strong> {partner.name}
    </p>
    <p>
      <strong>Role:</strong> {partner.role}
    </p>
    <p>
      <strong>Status:</strong> {partner.status}
      <button
        onClick={() => toggleStatus(partner.id)}
        style={{ marginLeft: "10px" }}
      >
```

```
            Toggle
          </button>
        </p>
        <p>
          <strong>Commission/Order:</strong> ₹{partner.commission}
        </p>
        <p>
          <strong>Wallet:</strong> ₹{partner.wallet}
        </p>
        <div>
          <button>✅ Completed:
{partner.orders.completed}</button>
          <button>🕒 Pending: {partner.orders.pending}</button>
          <button>📦 Total: {partner.orders.total}</button>
        </div>

        <div
          style={{
            background: "#eee",
            height: "10px",
            borderRadius: "6px",
            margin: "10px 0",
          }}
        >
```

```
          <div
            style={{
              width: `${completedPercent}%`,
              background: "#4caf50",
              height: "100%",
              borderRadius: "6px",
            }}
          />
        </div>
        <small>{completedPercent.toFixed(0)}% Completed</small>

        <div style={{ marginTop: "10px" }}>
          <input
            type="number"
            value={partner.id === parseInt(selectedPartner) ? amount :
""}
            onChange={(e) => {
              setSelectedPartner(partner.id);
              setAmount(e.target.value);
            }}
            placeholder="Amount"
          />
          <button onClick={() => handleFundChange(partner.id,
"add")}>
```

```
            Add Money
          </button>
          <button onClick={() => handleFundChange(partner.id,
"subtract")}>
            Deduct Money
          </button>
          <button
            onClick={() => deletePartner(partner.id)}
            style={{ marginLeft: "10px", color: "red" }}
          >
            Delete Partner
          </button>
        </div>
      </div>
    );
  })}

  <h2>Transaction History</h2>
  <ul>
    {transactionHistory.map((tx, index) => (
      <li key={index}>
        [{tx.time}] {tx.type} ₹{tx.amount}{" "}
        {tx.target ? `to/from ${tx.target}` : ""}
      </li>
```

```
        ))}
      </ul>
    </div>
  );
};

export default MCPDashboard;
```

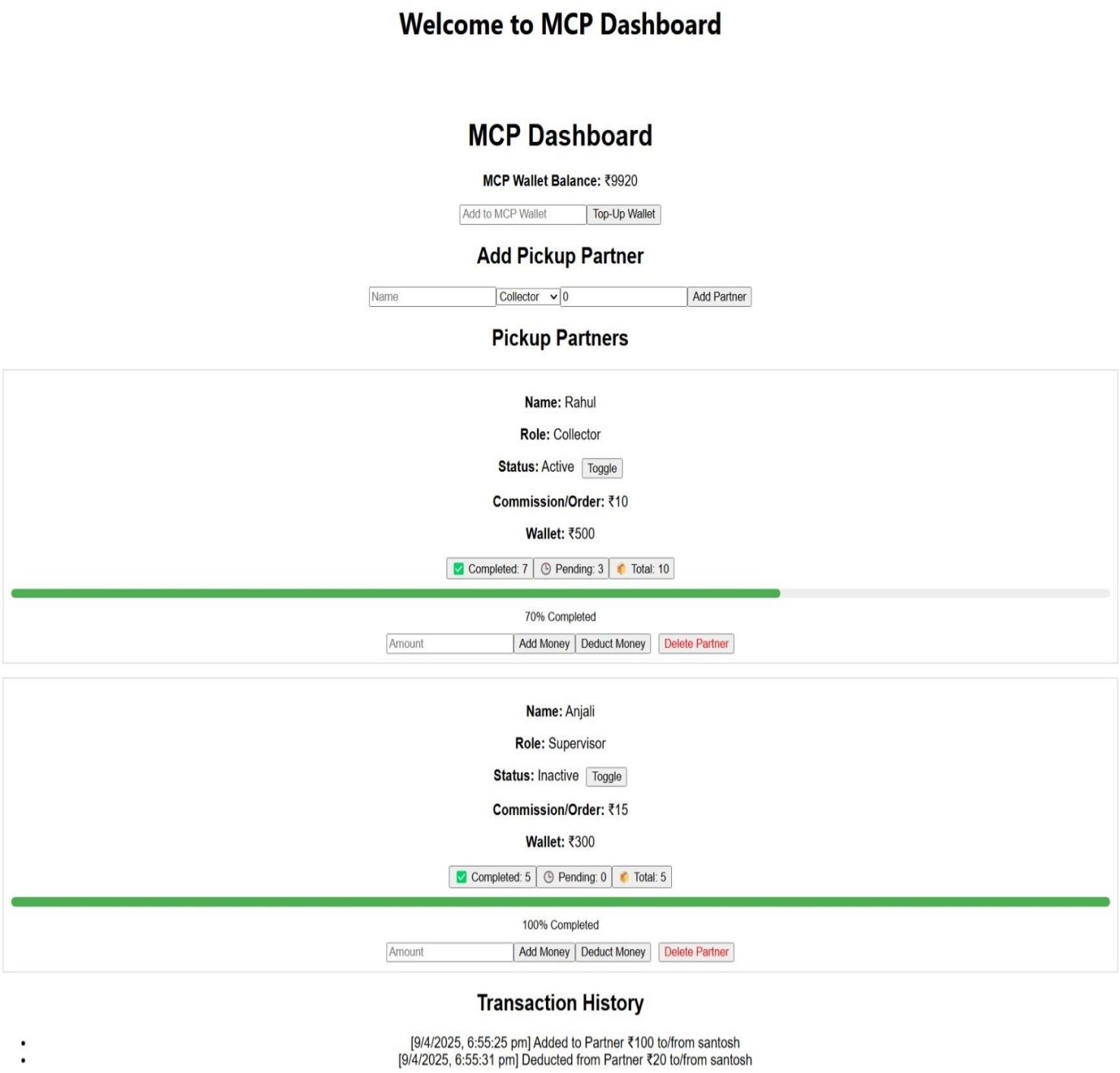
Then we to import the MCPDashboard.js in App.js

Then the App.js file will be as :-

```
import logo from "./logo.svg";
import "./App.css";
import MCPDashboard from "./MCPDashboard";
function App() {
  return (
    <div className="App">
      <h1 style={{ padding: "20px" }}>Welcome to MCP Dashboard</h1>

      <MCPDashboard />
    </div>
  );
}
export default App;
```

Here the OUTPUT Oof the above code :-



The detailed explanation of the codes:-

1.Importing React and useStat

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

React is the library that lets you build interactive UIs.

useState is a React Hook that allows us to track state (data that changes over time) inside functional components.

2.Component Definition

```
const MCPDashboard = () => {
```

This defines a **functional component** called MCPDashboard. It returns JSX (HTML-like syntax) that React will render to the screen.

3.State Declarations

```
const [walletBalance, setWalletBalance] = useState(10000);
```

- This keeps track of the **MCP wallet balance** (initially ₹10,000).
- setWalletBalance is a function to update the balance.

```
const [amount, setAmount] = useState("");
```

- Used to temporarily hold the input amount for **adding/deducting money** from a pickup partner's wallet.

```
const [selectedPartner, setSelectedPartner] = useState(1);
```

- Stores the **currently selected partner ID** when performing wallet operations.

```
const [selectedPartner, setSelectedPartner] = useState(1);
```

- Holds the **input for MCP wallet top-up**.

```
const [transactionHistory, setTransactionHistory] = useState([]);
```

- An array storing all **wallet-related transactions** (top-ups, fund changes) with details like type, amount, and time.

4.Pickup Partners List

```
const [pickupPartners, setPickupPartners] = useState([...]);
```

- This is the core list of all pickup partners with details:
    - id, name, status, orders, wallet, role, and commission.

5.New Partner Form State

```
const [newPartner, setNewPartner] = useState({
  name: "",
  role: "Collector",
  commission: 0,
});
```

- Tracks form data for **adding a new partner**.

Functional Logic

6.handleTopUp

```
const handleTopUp = () => {
```

- Called when user clicks "Top-Up Wallet".
- It:
    1. Converts topUpAmount to integer.
    2. Increases walletBalance.
    3. Logs the transaction.
    4. Clears the input field.

7.handleFundChange

```
const handleFundChange = (id, type) => {
```

- Triggered when adding or deducting money from a partner's wallet.
- Steps:
    1. Validates amount.
    2. Finds the partner by ID.
    3. Updates partner’s wallet.
    4. Adjusts MCP’s balance accordingly.

5. Adds a transaction to history.

8.toggleStatus

```
const toggleStatus = (id) => {
```

- Toggles a partner's status between **Active** and **Inactive**.

9.handleAddPartner

```
const handleAddPartner = () => {
```

- When adding a new partner:
    1. Checks if name exists and commission is valid.
    2. Creates a new partner object.
    3. Adds to the list.
    4. Resets form input.

10.deletePartner

```
const deletePartner = (id) => {
```

- Removes a partner from the list based on their id.

JSX: The UI Rendering

11.Main Container

```
<div style={{ padding: "20px", fontFamily: "Arial" }}>
  <h1>MCP Dashboard</h1>
```

- Styles the page and sets the title.

12.MCP Wallet Top-Up Section

```
<p><strong>MCP Wallet Balance:</strong> ₹{walletBalance}</p>
<input ... />
<button onClick={handleTopUp}>Top-Up Wallet</button>
```

- Shows wallet balance.
- Input for adding money.
- Button to top-up the wallet.

13.Add New Pickup Partner

```
<h2>Add Pickup Partner</h2>
<input ... /> // Name
<select ... /> // Role
<input ... /> // Commission
<button onClick={handleAddPartner}>Add Partner</button>
```

- Inputs to add a new partner with name, role, and commission.

14.Render List of Pickup Partners

```
{pickupPartners.map((partner) => {
```

- Loops over each partner and renders:
    - Name, role, status, wallet, commission, order stats.
    - Buttons to toggle status, add/deduct funds, and delete the partner.

15. Completion Bar

```
const completedPercent =
  (partner.orders.completed / partner.orders.total) * 100 || 0;
```

- Calculates what percentage of orders are completed for a progress bar.

16. Order Stats

```
<button> Completed: {partner.orders.completed}</button>
```

- Displays completed, pending, and total order stats.

16. Progress Bar

```
<div style={{ width: `${completedPercent}%` }} />
```

- A visual indicator showing order completion % using a filled bar.

17. Fund Management

```
<input value={partner.id === selectedPartner ? amount : ""} ... />
<button onClick={() => handleFundChange(partner.id, "add")}>Add</button>
<button onClick={() => handleFundChange(partner.id, "subtract")}>Deduct</button>
```

- Inputs and buttons to add/deduct money for each partner.

18.Transaction History Section

```
<h2>Transaction History</h2>
<ul>
  {transactionHistory.map((tx, index) => (
    <li key={index}>...</li>
  ))}
</ul>
```

- Lists all transactions (top-ups and partner wallet updates).

19.Export the Component

```
export default MCPDashboard;
```

- Makes this component usable in other parts of the app.

---

20. Summary of Features

Feature	Purpose
MCP Wallet Top-Up	Add money to main wallet
Add Pickup Partner	Dynamically add new collectors/supervisors
Partner Wallet Management	Transfer money to/from individual partners
Status Toggle	Activate/Deactivate partners
Order Stats	Track total, completed, and pending orders
Progress Bar	Visual representation of order completion
Transaction History Logging	Shows full audit trail of wallet-related actions