### High-Level and Low-Level Design of the System

### 1. High-Level Design (HLD)

HLD provides an overall system architecture, including components, technologies, and the flow of data. It outlines the project at a conceptual level.

#### **Key Components in HLD:**

1. **Frontend:**
   * **Technology Stack:** React, Material UI (MUI)
   * **Main Pages:**
     + Dashboard page that includes:
       - Transaction Table
       - Statistics
       - Bar Chart
       - Pie Chart
     + Month selection dropdown
2. **Backend:**
   * **Technology Stack:** Node.js, Express.js
   * **API Endpoints:**
     + /api/transactions: Fetch transaction data from a third-party API and store it in the database.
     + /api/statistics: Provides statistics for the selected month.
     + /api/charts/bar: Provides data for the bar chart based on the selected month.
     + /api/charts/pie: Provides data for the pie chart based on the selected month.
3. **Database:**
   * **Technology Stack:** MongoDB
   * **Collections:**
     + transactions: Stores transaction details.
     + statistics: Stores precomputed statistics for transactions.
     + charts: Stores data for visualizations (bar and pie charts).
4. **Third-Party API:**
   * The application will fetch transaction data from a third-party API on startup and store it in the MongoDB database.

#### **Component Interactions:**

* **Frontend (React) → Backend (Node.js)**:
  + The frontend dashboard components will send HTTP requests to the backend to retrieve transaction, statistics, and chart data for a selected month.
* **Backend (Node.js) → MongoDB**:
  + The backend will interact with MongoDB to fetch or update data based on the requests from the frontend.
* **Third-Party API → Backend (Node.js)**:
  + The backend fetches the transaction data from the third-party API and initializes the MongoDB database when the app starts.

### 2. Low-Level Design (LLD)

LLD focuses on the implementation details, providing a closer look at how each component will be built and function.

#### **Frontend Components in LLD:**

1. **Dashboard.js**
   * **Functionality:**
     + Fetches data for transactions, statistics, and charts based on the selected month.
     + Displays the data in various components such as the table, bar chart, and pie chart.
     + Provides a dropdown to select the month.
2. **TransactionTable.js**
   * **Props:**
     + selectedMonth: The selected month passed from the Dashboard component.
   * **Functionality:**
     + Sends an API request to the backend to fetch transaction data for the selected month.
     + Displays the transaction data in a paginated table format using MUI’s table component.
3. **Statistics.js**
   * **Props:**
     + selectedMonth: The selected month.
   * **Functionality:**
     + Fetches statistics for the selected month from the backend.
     + Displays key metrics such as the total number of transactions, total revenue, etc.
4. **BarChart.js**
   * **Props:**
     + selectedMonth: The selected month.
   * **Functionality:**
     + Fetches bar chart data from the backend.
     + Displays the bar chart using a charting library like Chart.js or Recharts.
5. **PieChart.js**
   * **Props:**
     + selectedMonth: The selected month.
   * **Functionality:**
     + Fetches pie chart data from the backend.
     + Displays the pie chart using a charting library.
6. **Month Selector Dropdown (TextField)**
   * Provides an interface for users to select the month, and passes the selected month to other components.

#### **Backend API Design in LLD:**

1. **API: /api/transactions**
   * **Method:** GET
   * **Description:** Fetches transaction data for a specific month from MongoDB.
   * **Implementation:**
     + Query the transactions collection based on the selected month.
     + Return the transaction data in a paginated format.
2. **API: /api/statistics**
   * **Method:** GET
   * **Description:** Fetches statistics for the selected month from MongoDB.
   * **Implementation:**
     + Query the statistics collection for the selected month.
     + Return key metrics such as total transactions, average transaction value, etc.
3. **API: /api/charts/bar**
   * **Method:** GET
   * **Description:** Fetches data for the bar chart based on the selected month.
   * **Implementation:**
     + Query the charts collection for bar chart data.
     + Return the data for the bar chart.
4. **API: /api/charts/pie**
   * **Method:** GET
   * **Description:** Fetches data for the pie chart based on the selected month.
   * **Implementation:**
     + Query the charts collection for pie chart data.
     + Return the data for the pie chart.
5. **Database Initialization on Startup**
   * **Description:** Fetch data from the third-party API and insert it into MongoDB every time the application starts.
   * **Implementation:**
     + When the server starts, it will send a request to the third-party API to fetch transaction data.
     + The data will be inserted into the transactions collection in MongoDB.
     + The backend can process this data to generate statistics and chart data and store them in the corresponding collections.

#### **Database Design in LLD:**

1. **transactions Collection:**
   * **Fields:**
     + \_id: Transaction ID (auto-generated)
     + date: Date of the transaction
     + amount: Amount involved in the transaction
     + category: Category of the transaction
     + description: Description of the transaction
     + month: The month when the transaction occurred (to filter by month)
2. **statistics Collection:**
   * **Fields:**
     + \_id: ID
     + month: The month for which statistics are stored
     + totalTransactions: Total number of transactions
     + totalAmount: Total transaction amount
     + averageTransactionValue: Average value of a transaction
3. **charts Collection:**
   * **Fields:**
     + \_id: ID
     + month: The month for which the data is generated
     + barData: Data for the bar chart
     + pieData: Data for the pie chart

### ****Summary of HLD and LLD:****

1. **HLD** outlines the system architecture with a clear separation between frontend (React) and backend (Node.js) communicating through APIs, supported by MongoDB for persistent data storage.
2. **LLD** delves deeper into the specific components of the frontend (like TransactionTable, Statistics, BarChart, etc.), backend API endpoints, and the database schema, defining how the data flows and how each part of the system is implemented.

This combination of HLD and LLD will ensure that your project has a solid architectural foundation and detailed implementation steps for each part of the system.