**4. Documentation**

**📌 1. API Documentation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Endpoint** | **Request Body** | **Description** |
| GET | /people | – | Returns list of all people |
| POST | /people | { "name": "Alice" } | Adds a new person |
| POST | /expenses | JSON (see below) | Creates a new expense and splits it |
| GET | /balances | – | Shows current balance for each person |
| GET | /settlements | – | Optimizes who should pay whom |

**✅ Sample /expenses Request Body:**

json

CopyEdit

{

"amount": 1000,

"description": "Dinner",

"paidBy": "Alice",

"shares": [

{

"name": "Alice",

"shareType": "EQUAL",

"value": 0

},

{

"name": "Bob",

"shareType": "EQUAL",

"value": 0

}

]

}

**⚙️ 2. Settlement Calculation Logic**

**Step-by-step:**

1. For every expense, calculate each person's share:
   * EQUAL: split total evenly.
   * EXACT: use given value.
   * PERCENT: (value%) × total.
2. Calculate how much each person owes or is owed.
3. Determine net balances:
   * Positive → user is owed.
   * Negative → user owes money.
4. Match debtors to creditors greedily to minimize transactions:
   * Person A owes Person B, etc.

**Example**:

* Alice paid ₹1000
* Bob and Charlie each owe ₹500
* Settlements:
  + Bob → Alice ₹500
  + Charlie → Alice ₹500

**🚧 3. Known Limitations / Assumptions**

* No user authentication or sessions (stateless)
* Assumes names are unique identifiers (can be improved)
* No currency support or multi-group logic
* Transactions are not optimized beyond a basic greedy algorithm