Split Bill Between Friends

Task Summary:

Create a Laravel web app where users can:

- Create a bill (e.g., dinner bill, trip expense).
- Add friends involved in the bill.
- Add items with who paid and who shared.
- See how much each person owes or is owed.

Core Requirements (Must Do):

1. **Create Bill**
    - Inputs:
        - Bill name (e.g., "Goa Trip", "Dinner at XYZ")
        - Total people involved (names or emails)
2. **Add Expenses**
    - Multiple entries like:
        - **Item name**
        - **Amount**
        - **Paid by** (friend)
        - **Shared by** (one or more friends)
3. **Calculate Split**
    - System calculates:
        - How much each friend spent
        - How much each friend owes

        - Final balance sheet (who pays whom and how much)

Database Schema Suggestion:

```
// bills
id | name | created_at | updated_at

// friends
id | name | bill_id | created_at | updated_at

// expenses
id | bill_id | title | amount | paid_by (friend_id) | created_at

// expense_shares (pivot)
id | expense_id | friend_id | created_at
```

Example Logic:

- Alice, Bob, and Charlie are in a bill.
- Alice pays ₹600 for dinner shared by all.
- Bob pays ₹300 for snacks shared only with Charlie.
- Final output:
    - Alice paid ₹600 but owes only ₹200 → gets ₹400 back.
    - Bob paid ₹300→ owes ₹200 to Alice
    - Charlie paid ₹0 → owes ₹150 to bob and ₹200 to Alice

Evaluation Criteria

Technical Skills

- Correct use of Laravel MVC structure (models, views, controllers)
- Proper use of migrations and database relationships (hasMany, belongsToMany)
- Clean routing and controller methods
- Form validation implemented (server-side)

Logic & Problem Solving

- Accurate calculation of each person's share
- Correctly tracks who paid and who owes
- Logic handles multiple friends and expenses
- Balance sheet displays net amounts correctly

UI/UX & Presentation

- User-friendly forms to add bills, friends, and expenses
- Final result (split summary) is clearly readable
- Optional: Responsive or styled UI using Bootstrap/Tailwind

Code Quality

- Code is clean, readable, and properly commented
- Separation of logic (e.g., business logic not all in controller)
- Use of Laravel features like collections, blade templating, etc.

AI Usage

- Candidate clearly indicates what parts were done with AI help
- AI-generated code is understood and integrated well
- Candidate shows ability to adapt/improve AI suggestions