This complete implementation includes:

1. **Full Authentication Flow**
   * User registration
   * Login with JWT
   * Protected endpoints using token authentication
2. **User Management**
   * Get current user details
   * Update user information
   * Delete user account
3. **Group Management**
   * Create groups
   * List user's groups
   * Get group details
   * Add/remove group members
   * View group balances
4. **Expense Management**
   * Create expenses with different split types
   * List user's expenses
   * Get expense details
   * Delete expenses (with balance reversal)
   * Automatic balance calculations
5. **Balance Calculation System**
   * Handles equal, percentage, and exact splits
   * Maintains debt relationships between users
   * Automatically updates balances on expense creation/deletion

Key Features:

* Proper MongoDB ObjectId handling
* Full error handling
* Input validation
* CORS support
* Clean JSON responses
* ISO 8601 date formatting
* Password hashing with bcrypt
* JWT authentication

To test the API endpoints:

**Create User**

curl -X POST http://localhost:5000/auth/register \

-H "Content-Type: application/json" \

-d '{"name":"Anish Sood","email":"anish@example.com","password":"secret"}'

**Login**

curl -X POST http://localhost:5000/auth/login \

-H "Content-Type: application/json" \

-d '{"email":"anish@example.com","password":"secret"}'

**Create Group**

curl -X POST http://localhost:5000/groups \

-H "Content-Type: application/json" \

-H "Authorization: <JWT\_TOKEN>" \

-d '{"name":"Vacation Group"}'

**Add Expense (Equal Split)**

curl -X POST http://localhost:5000/expenses \

-H "Content-Type: application/json" \

-H "Authorization: <JWT\_TOKEN>" \

-d '{

"description":"Dinner",

"amount":100,

"split\_type":"equal",

"split\_between":["user1\_id","user2\_id","user3\_id"],

"group\_id":"group\_id"

}'

**Get Group Balances**

curl http://localhost:5000/groups/<group\_id>/balances \

-H "Authorization: <JWT\_TOKEN>"

This implementation provides all the core functionality needed for a Splitwise-like application. You can extend it further by adding:

* Pagination for large datasets
* Expense categories
* Settle-up functionality
* Activity feed
* Proper input validation schemas
* Rate limiting
* Enhanced error messages