Database schema

[Users]
- user_id (PK)
- username (unique)
- password
- email
- created_at

[Transactions]
- transaction_id (PK)
- user_id (FK)
- amount
- category (Food, Rent, etc.)
- type (Income/Expense)
- date
- description

[Budgets]
- budget_id (PK)
- user_id (FK)
- category
- limit_amount
- current_spent
- start_date
- end_date

[Notifications]
- notification_id (PK)
- user_id (FK)
- message
- status (Sent/Pending)
- created_at

Class diagram

`UserService`: Handles core user data.
`AuthService`: Manages authentication.
`TransactionService`: Adds/delete/updates transactions.
`BudgetService`: Adds/delete/updates budget
`NotificationService`: Triggers alerts when budgets are exceeded.
`AnalyticsService`: For dashboard graphs

API design

Endpoint: `/api/v1/transactions`

Method	Endpoint	Purpose	Request Body
`POST`	`/api/v1/transactions`	Add new transaction	`{userId, amount, category}`
`GET`	`/api/v1/transactions`	Fetch all transactions	Query params: `userId`, `date`
`DELETE`	`/api/v1/transactions/{id}`	Delete a transaction	Path param: `id`

Example Request:

```
POST /api/transactions
{
  "userId": 101,
  "amount": 2000,
  "category": "Groceries",
  "type": "Expense",
  "description": "Monthly groceries"
}
```

Endpoint: `/api/v1/budgets`

Method	Endpoint	Purpose	Request Body
`POST`	`/api/v1/budgets`	Add new budget	`{userId, category}`
`GET`	`/api/v1/budgets`	Fetch all budgets	Query params: `userId`, `date`
`DELETE`	`/api/v1/budgets/{id}`	Delete a budget	Path param: `id`

Kafka Event design

Topic: `budget_alerts`
Producer: `BudgetService`
Consumer: `NotificationService`

Topic: `transaction_data`
Producer: `TransactionService`
Consumer: `BudgetService`

Security Plan

1. **JWT Authentication** for secure API access.
2. **Encrypt sensitive data** like passwords and transaction details.
3. **Rate Limiting** to prevent brute-force attacks.

LLD Deliverables

- **Database schema** with table relationships.
- **Class diagrams** showing object interactions.
- **API contracts** detailing endpoints and data structures.
- **Kafka event models** for inter-service communication.
- **Security plan** addressing common vulnerabilities.
- **Redis caching** for caching
- **Web sockets** for notification