Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core backend architecture for **FinLedger - Simple Expense Tracker API**. The service enables user registration, authentication, and CRUD operations on expenses using Node.js, Express, and MySQL.

---

1. System Components

Component	Description	Key Responsibilities
API Server	Express.js application	Handle HTTP requests, route mapping
Auth Module	JWT-based authentication	User login, token validation
User Controller	User management endpoints	Register, login
Expense Controller	Expense management endpoints	CRUD for expenses
MySQL Database	Persistent storage	Store users and expenses
ORM/DB Layer	Database abstraction (e.g., Sequelize/Raw)	Query execution, data mapping

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`UserController`	Handles user endpoints	`register()`, `login()`
`ExpenseController`	Handles expense endpoints	`create()`, `read()`, `update()`, `delete()`
`AuthMiddleware`	JWT validation	`verifyToken()`
`UserModel`	User DB model	`id`, `email`, `passwordHash`
`ExpenseModel`	Expense DB model	`id`, `userId`, `amount`, `category`, `date`, `description`

---

3. Data Structure Overview

Model	Fields
User	`id` (PK, int), `email` (unique, string), `passwordHash` (string)
Expense	`id` (PK, int), `userId` (FK, int), `amount` (decimal), `category` (string), `date` (date), `description` (string)

---

4. Algorithms/Logic

User Registration/Login:

```
register(email, password):
    if email exists in DB:
        return error
    hash password
    store user in DB

login(email, password):
    fetch user by email
    if password matches hash:
        generate JWT token
        return token
    else:
        return error
```

Expense CRUD (Authenticated):

```
createExpense(userId, data):
    insert expense with userId

getExpenses(userId):
    select expenses where userId

updateExpense(userId, expenseId, data):
    update expense where id=expenseId and userId

deleteExpense(userId, expenseId):
    delete expense where id=expenseId and userId
```

---

5. Error Handling

Scenario	Handling Approach
Invalid credentials	Return 401 Unauthorized
JWT token missing/invalid	Return 401 Unauthorized
Resource not found (expense)	Return 404 Not Found
DB constraint violation	Return 400 Bad Request
Server/database error	Return 500 Internal Server Error
Unauthorized expense access	Return 403 Forbidden

End of Document