Product Requirements & Specification Document

Project Name

FinLedger - Simple Expense Tracker API

Description

FinLedger is a backend RESTful API for tracking personal expenses. It supports user registration, authentication, and CRUD operations on expenses. The API is built using Node.js, Express, and MySQL.

---

1. Goals & Objectives

Goal	Description
User Management	Allow users to register, log in, and manage their account.
Expense Tracking	Enable users to create, read, update, and delete expense records.
Data Security	Ensure user data privacy and secure authentication.
Simplicity & Usability	Provide a clean, easy-to-use API for integration with frontend or mobile apps.

---

2. Functional Requirements

2.1 User Management

Feature	Description
Registration	Users can register with email and password.
Authentication	JWT-based login and session management.
Profile	Users can view and update their profile.

2.2 Expense Management

Feature	Description
Create Expense	Add a new expense record.
Read Expenses	List all expenses for the authenticated user.
Update Expense	Edit an existing expense.
Delete Expense	Remove an expense record.

---

3. Non-Functional Requirements

Requirement	Specification
Performance	API response time < 500ms for all endpoints.
Security	Passwords hashed (bcrypt), JWT for auth.
Scalability	Support up to 10,000 users.
Documentation	OpenAPI (Swagger) specification provided.
Error Handling	Consistent error responses (JSON).

---

4. API Endpoints

4.1 Authentication

Method	Endpoint	Description	Auth Required
POST	/api/register	Register new user	No
POST	/api/login	User login	No

4.2 User

Method	Endpoint	Description	Auth Required
GET	/api/user	Get user profile	Yes
PUT	/api/user	Update user profile	Yes

4.3 Expenses

Method	Endpoint	Description	Auth Required
GET	/api/expenses	List all expenses	Yes
POST	/api/expenses	Create new expense	Yes
GET	/api/expenses/:id	Get expense by ID	Yes
PUT	/api/expenses/:id	Update expense by ID	Yes
DELETE	/api/expenses/:id	Delete expense by ID	Yes

---

5. Data Model

5.1 User

Field	Type	Constraints
id	INT	PK, auto-increment
email	VARCHAR	Unique, required
password	VARCHAR	Hashed, required
name	VARCHAR	Optional

created_at	DATETIME	Default now

5.2 Expense

Field	Type	Constraints
id	INT	PK, auto-increment
user_id	INT	FK -> User(id)
amount	DECIMAL	Required
category	VARCHAR	Required
date	DATE	Required
note	VARCHAR	Optional
created_at	DATETIME	Default now

---

6. Security

- Passwords stored using bcrypt hashing.
- JWT tokens for authentication; tokens required for all expense/user endpoints.
- Input validation and sanitization on all endpoints.

---

7. Error Handling

- All errors returned as JSON:

```
{
  "error": "Error message",
  "details": "Optional details"
}
```

- Standard HTTP status codes used.

---

8. Technology Stack

Layer	Technology
Language	Node.js
Framework	Express
Database	MySQL
Auth	JWT, bcrypt
Documentation	Swagger (OpenAPI)

---

9. Out of Scope

- No frontend/UI.
- No third-party payment integrations.
- No advanced analytics or reporting.

---

10. Milestones

Milestone	Description
API Skeleton	Project setup, basic routing
User Management	Registration, login, profile
Expense CRUD	All expense endpoints
Security & Validation	Auth, input validation
Documentation	Swagger/OpenAPI docs
Testing	Unit & integration tests

---

11. Acceptance Criteria

- All endpoints function as specified.
- Only authenticated users can access their data.
- API passes all test cases.
- Documentation is complete and accurate.

End of Document