Finance Management System — Requirements Document

# 1. Project Overview

The Finance Management System is a Python-based application using MySQL for persistent storage. It allows users to manage personal expenses, categorized by type, with full CRUD operations, authentication, and reporting. The application follows object-oriented principles and includes user-defined exceptions, modular package organization, and unit testing.

# 2. Objectives

- Securely allow user registration and login.  
- Enable users to add, view, update, and delete expenses.  
- Categorize expenses under custom or predefined types.  
- Generate expense reports filtered by time and category.  
- Store all data in a structured MySQL database.  
- Write unit tests to ensure code reliability.

# 3. Technologies Used

|  |  |
| --- | --- |
| Technology | Purpose |
| Python 3.x | Core programming language |
| MySQL | Relational database for storage |
| MySQL Connector | Python package for DB interaction |
| unittest / pytest | Testing framework |
| GitHub | Version control and submission |

# 4. Functional Requirements

4.1 User Management  
- Register: New user creation.  
- Login: Validate user credentials.  
- Delete: Remove user by ID.

4.2 Expense Management  
- Add Expense: Add expense with category, amount, date, and description.  
- Update Expense: Modify expense details.  
- Delete Expense: Remove expense by ID.  
- List Expenses: View all expenses for a user.

4.3 Expense Categorization  
- Expenses are tagged with categories like Food, Travel, Utilities.  
- Categories stored in a separate table.

4.4 Report Generation  
- Show expenses filtered by date range.  
- Display total expenses by category.

# 5. Database Schema

## 5.1 Users

|  |  |  |
| --- | --- | --- |
| Column | Type | Notes |
| user\_id | INT (PK) | Auto-incremented |
| username | VARCHAR(100) | Unique, not null |
| password | VARCHAR(100) | Hashed/plain, not null |
| email | VARCHAR(100) | Unique, not null |

## 5.2 ExpenseCategories

|  |  |  |
| --- | --- | --- |
| Column | Type | Notes |
| category\_id | INT (PK) | Auto-incremented |
| category\_name | VARCHAR(100) | Not null |

## 5.3 Expenses

|  |  |  |
| --- | --- | --- |
| Column | Type | Notes |
| expense\_id | INT (PK) | Auto-incremented |
| user\_id | INT (FK) | References Users(user\_id) |
| amount | DECIMAL(10, 2) | Not null |
| category\_id | INT (FK) | References ExpenseCategories(category\_id) |
| expense\_date | DATE | Not null |
| description | TEXT | Optional |

# 6. Python Package Structure

entity/: Contains model classes: User, Expense, Category  
dao/: Interface-like service structure and DB logic  
exception/: Custom exceptions like UserNotFoundException  
util/: DB connection and property reading utility  
main/: Main driver program (menu-based interaction)  
test/: Unit test scripts using unittest or pytest  
config/: Property file with DB credentials

# 7. Custom Exceptions

- UserNotFoundException  
- ExpenseNotFoundException  
These are to be raised from business logic and caught in the main program to handle errors gracefully.

# 8. Unit Testing

- Test if a user is successfully created.  
- Test if an expense is correctly added.  
- Test retrieval of expense records.  
- Test whether exceptions are raised on invalid inputs.

# 9. Deliverables

- Full Python source code  
- MySQL schema dump (.sql file)  
- Unit test cases  
- db.properties file for DB connection  
- GitHub repo link (as submission)

# 10. Submission Guidelines

- Upload the completed project to GitHub.  
- Ensure the repo contains a README and instructions.  
- Share the link with trainers and Hexavarsity.