

# EXPENSE TRACKER

*Personal Finance Management System*

By:

Krupa Devar : 24107065

Shravni Bhosale : 24107025

Shravani Karpe : 24107047

Project Guide : Ms. Aavani Nair

Department of Computer Science Engineering

(Data Science)

A.P. Shah Institute of Technology, Thane

Academic Year: 2025-26



# Outline

- Introduction
- Problem Statement
- System Design
- Technologies and Methodologies
- Implementation
- Conclusion
- References



# Introduction

## Real-time Problem Observation:

- Students and individuals struggle with tracking daily expenses
- Lack of simple, accessible expense management tools
- Manual record-keeping is time-consuming and error-prone
- Difficulty in analyzing spending patterns

## Motivation:

- Need for digital solution to manage personal finances
- Enable better financial decision-making
- Provide instant access to expense history
- Simplify budget tracking for students

## Objectives:

- Develop web-based expense tracking application
- Implement database integration for persistent storage
- Create user-friendly interface for expense management
- Enable real-time calculation of total expenditure



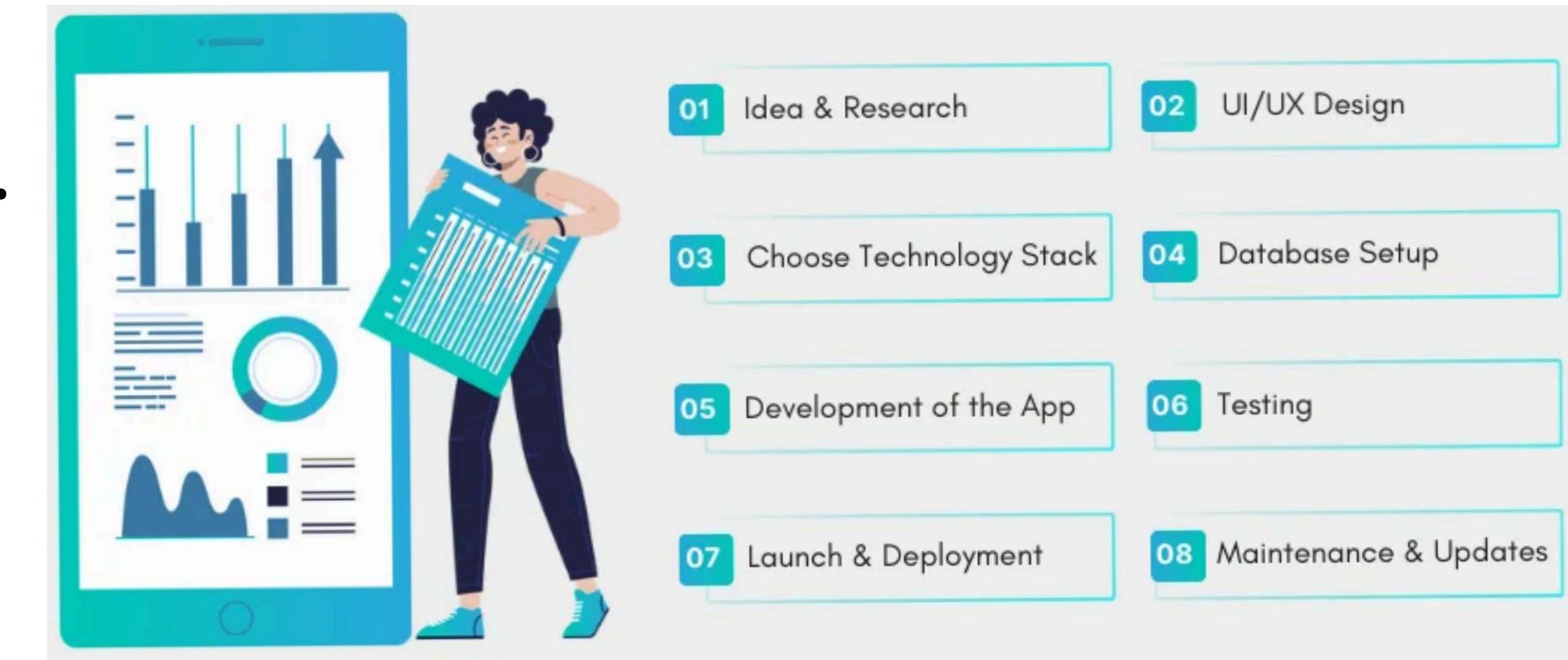
# Problem Statement

## Core Problem:

Individuals lack an efficient system to record, view, and analyze their daily expenses, leading to poor financial awareness and budget management.

## Key Points to Address:

- Easy expense entry mechanism
- Categorized expense tracking
- Real-time data storage and retrieval
- Total expenditure calculation
- Cross-platform accessibility



# System Architecture

## Three-Tier Architecture

### Presentation Layer (Frontend):

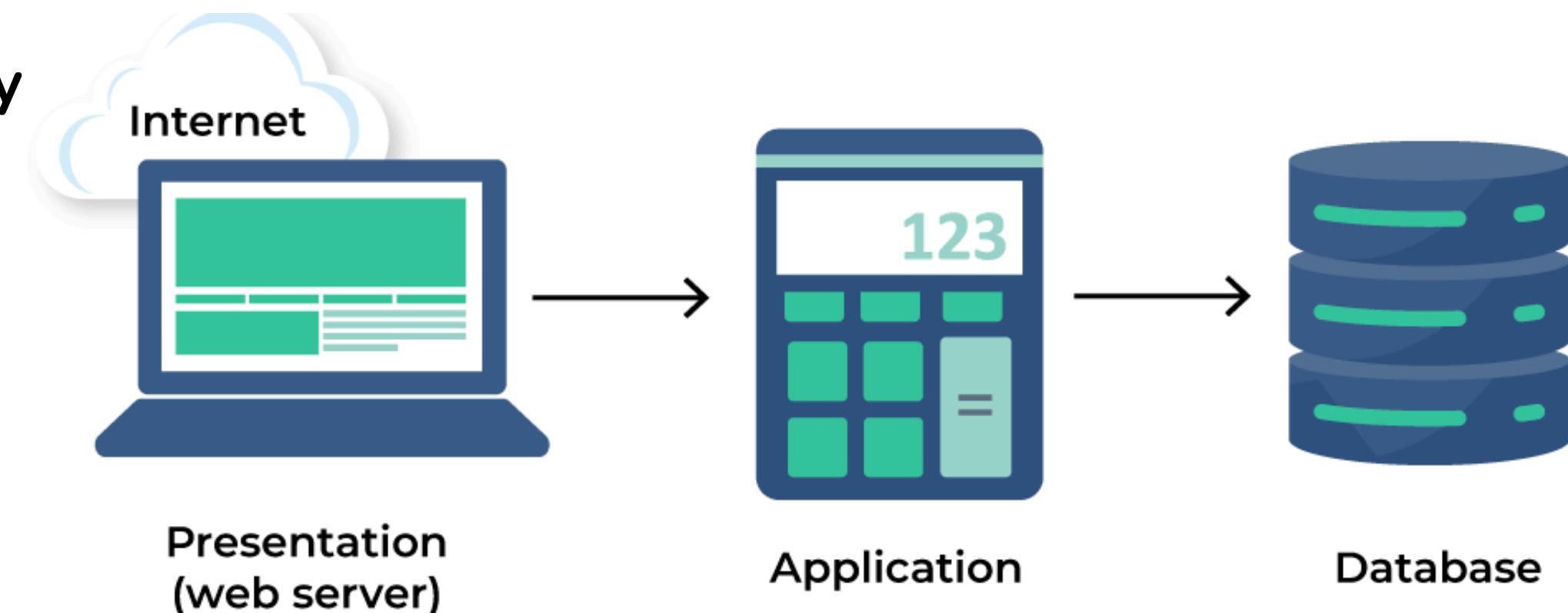
- JSP pages with HTML/CSS/JavaScript
- User interface for data input and display

### Application Layer (Backend):

- Java Servlets handling HTTP requests
- Business logic processing
- Data validation

### Data Layer:

- MySQL database
- JDBC connectivity
- Persistent data storage



# System Design - Component Diagram

## Key Components:

### Model Layer:

- `Expense.java` - Data model with attributes (`id`, `description`, `amount`, `category`, `date`)

### DAO Layer:

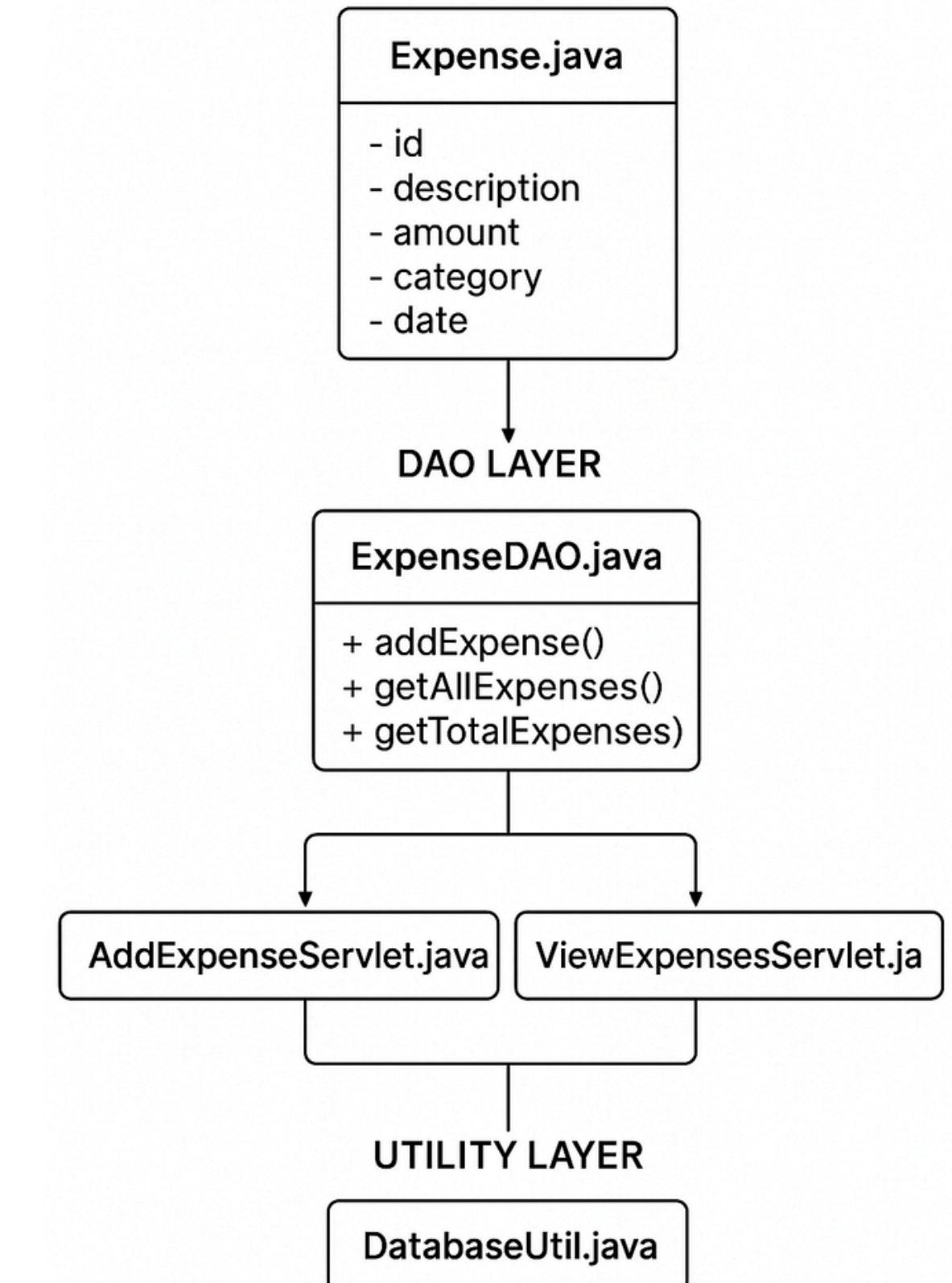
- `ExpenseDAO.java` - Database operations (CRUD)
- Methods: `addExpense()`, `getAllExpenses()`, `getTotalExpenses()`

### Controller Layer:

- `AddExpenseServlet.java` - Handles expense creation
- `ViewExpensesServlet.java` - Retrieves and displays expenses

### Utility Layer:

- `DatabaseUtil.java` - Database connection management



# Database Design

## Expense Table Schema:

- id (INT, PRIMARY KEY, AUTO\_INCREMENT)
- description (VARCHAR, 255)
- amount (DECIMAL, 10,2)
- category (VARCHAR, 100)
- date (DATE)
- created\_at (TIMESTAMP)

## Categories Supported:

Food, Transport, Shopping, Bills,  
Entertainment, Other



### EXPENSE TABLE

Field Name	Data Type	Constraints
id	INT	PRIMARY KEY, AUTO_INCREMENT
description	VARCHAR(255)	NOT NULL
amount	DECIMAL(10.2)	NOT NULL
category	VARCHAR(100)	NOT NULL
date	DATE	NOT NULL
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

### Sample Data

id	description	amount	category	date
1	Office Supplies	250.00	Bills	2025-10-01
2	Team Lunch	450.00	Food	2025-10-02
3	Metro Pass	500.00	Transport	2025-10-03

### Categories Supported

Food Transport Shopping Other

# Technologies and Methodologies

## Frontend Technologies:

- HTML5 for structure
- CSS3 for styling and animations
- JavaScript for client-side validation

## Backend Technologies:

- Java JDK 11
- Java Servlets API 4.0
- JSP (JavaServer Pages) 2.2

## Database:

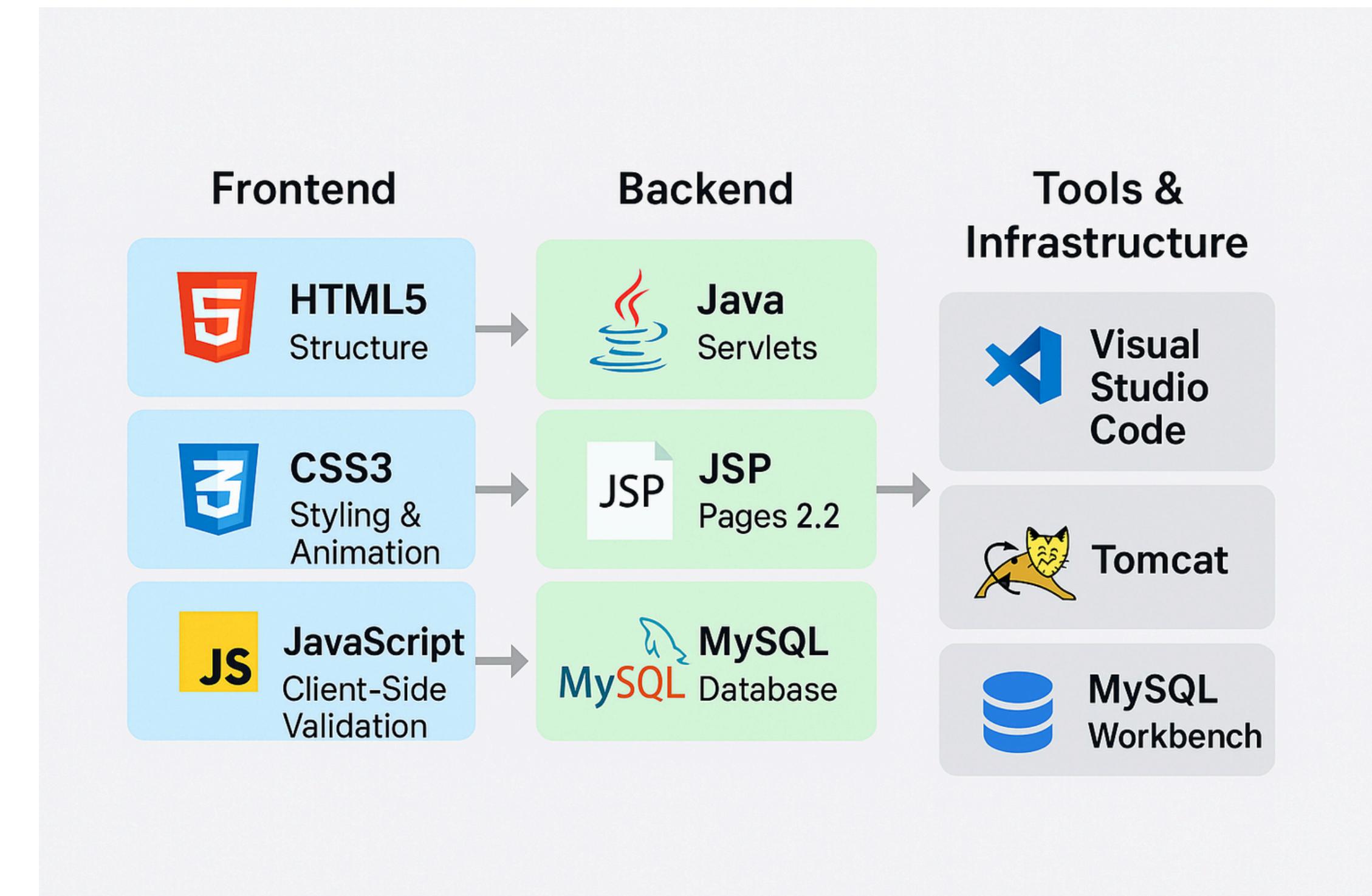
- MySQL 8.0
- JDBC Driver for connectivity

## Server:

- Apache Tomcat 9.0

## Development Tools:

- Visual Studio Code
- MySQL Workbench



# Implementation - MVC Pattern

## Model-View-Controller Architecture:

### Model:

- Expense entity class with encapsulation
- Getter/setter methods for data access

### View:

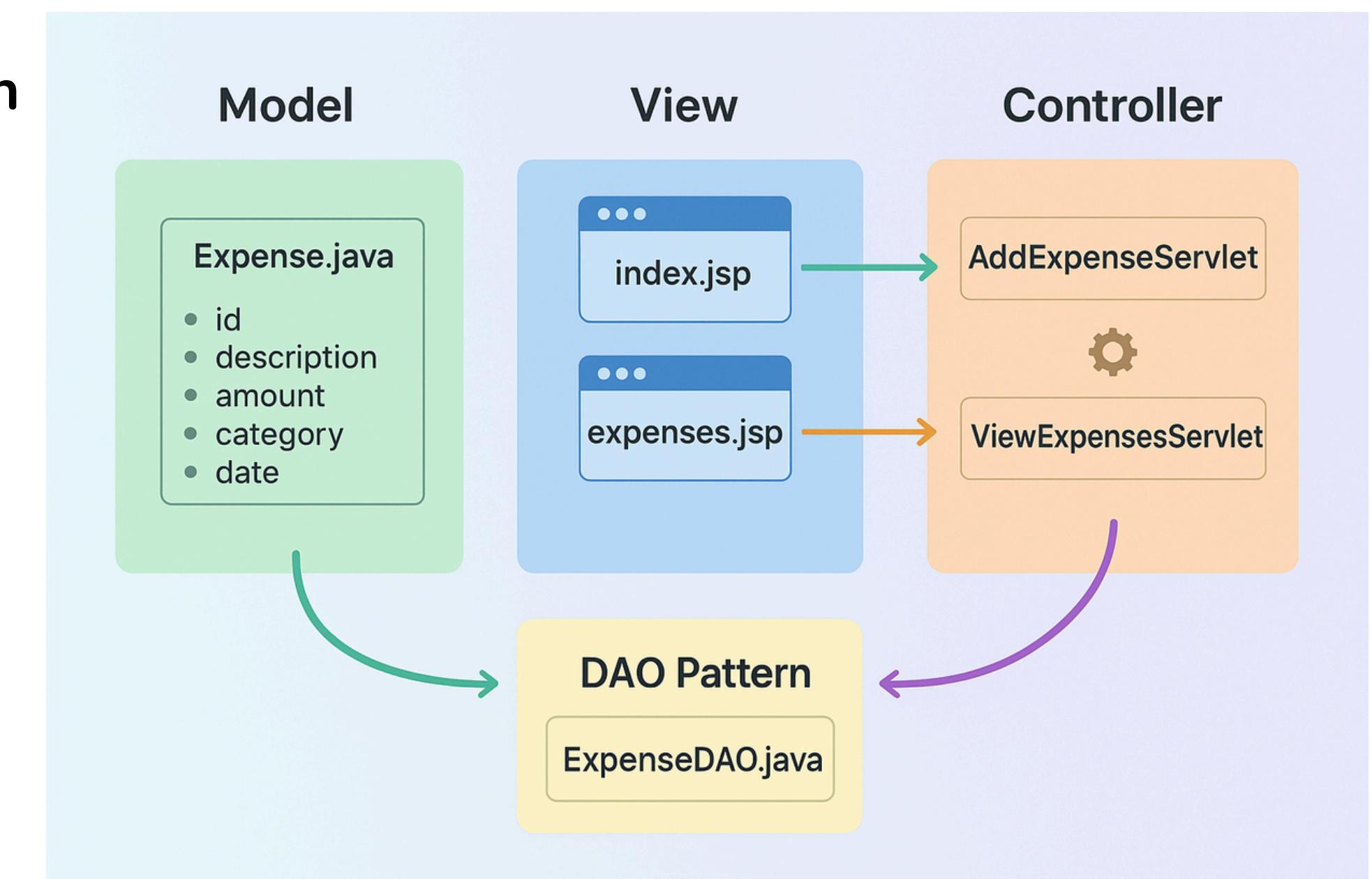
- index.jsp - Expense entry form
- expenses.jsp - Expense list display

### Controller:

- Servlets process HTTP requests
- Forward data to appropriate views
- Handle business logic

### DAO Pattern:

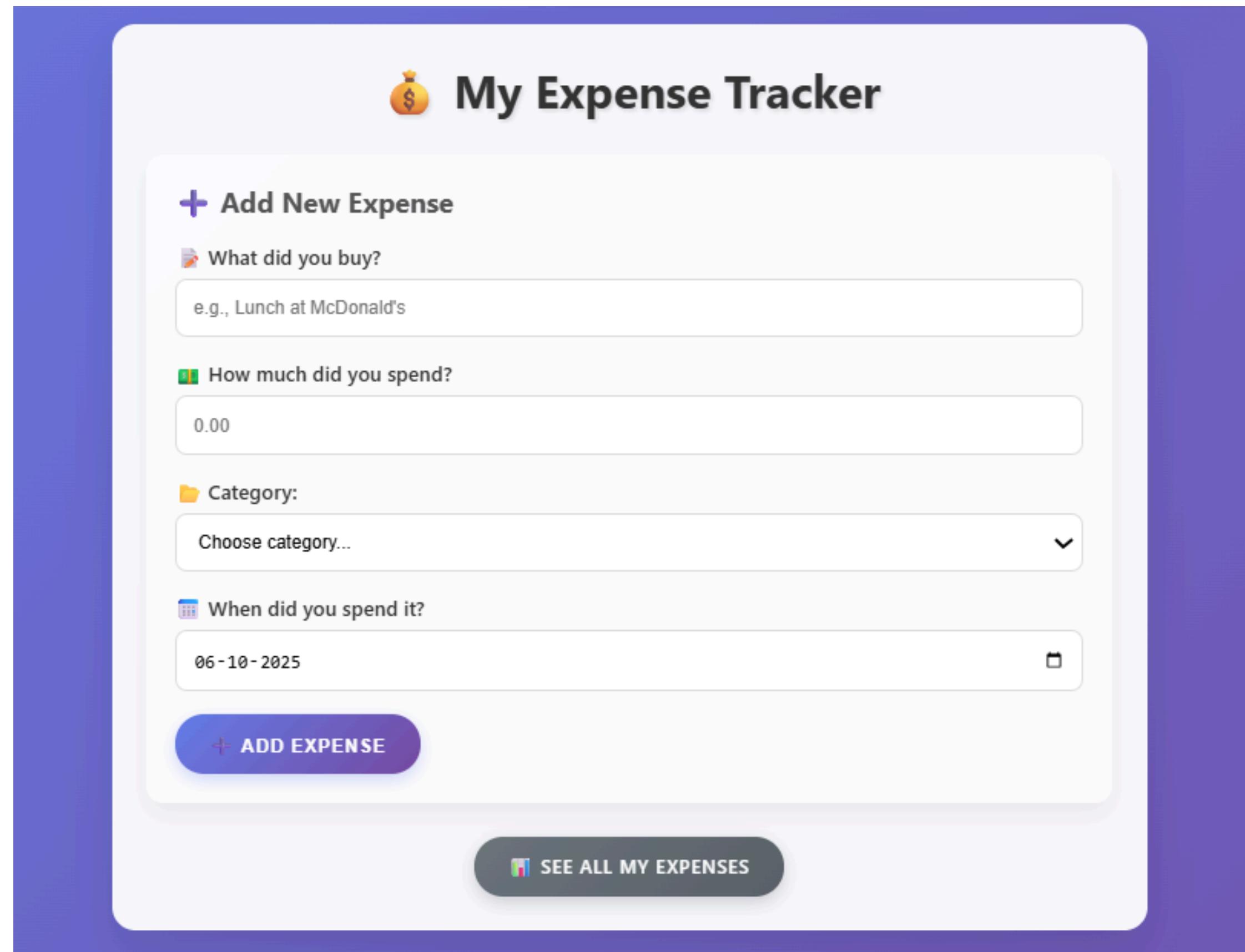
- Separates database operations
- Reusable data access methods



# Key Features Implemented

## Add Expense Functionality:

- Form with description, amount, category, date fields
- Client-side validation using JavaScript
- Server-side processing via servlet
- Success/error message display



# Key Features Implemented

## View Expenses

### Functionality:

- Tabular display of all expenses
- Automatic total calculation
- Sorted by date (newest first)
- Professional UI with gradient design

The screenshot shows a mobile application interface for managing expenses. At the top, there is a header with the text "All My Expenses" next to a bar chart icon. Below this is a prominent orange banner displaying the text "Total Spent: ₹3299.00" with a gold coin icon. The main content area is a table with five columns: DATE, DESCRIPTION, CATEGORY, and AMOUNT. The table lists five expense entries:

DATE	DESCRIPTION	CATEGORY	AMOUNT
2025-10-05	Project Books	Shopping	₹1200.00
2025-10-04	Internet Bill	Bills	₹899.00
2025-10-03	Metro Pass	Transport	₹500.00
2025-10-02	Team Lunch	Food	₹450.00
2025-10-01	Office Supplies	Bills	₹250.00

At the bottom of the screen is a purple button with the text "+ ADD NEW EXPENSE" and a plus sign icon.

# Key Features Implemented

## Database

### Operations:

- INSERT for

- adding

- expenses

- SELECT for
- retrieving data

- Prepared
- statements for
  - SQL injection
  - prevention

#### INSERT

✓ Add expenses

#### SELECT

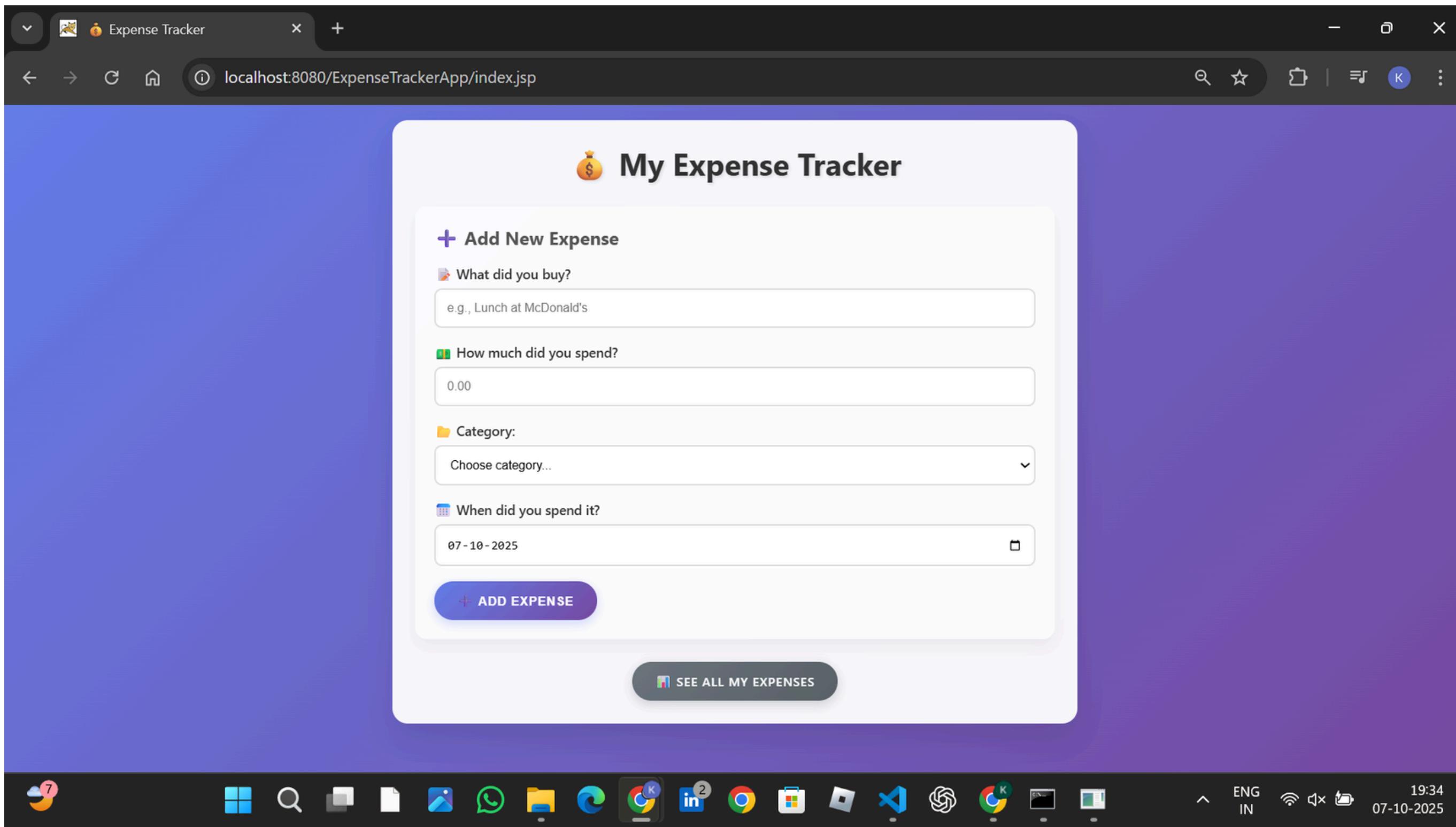
✓ Get all data

#### PREPARED STATEMENTS

✓ SQL injection prevention

# Implementation Screenshots

## 1. Add Expense Form Interface



# Implementation Screenshots

## 2. Expenses List with Total

The screenshot shows a web browser window titled "All My Expenses" with the URL "localhost:8080/ExpenseTrackerApp/viewExpenses". The main content area displays a summary card with the heading "All My Expenses" and a red banner showing "Total Spent: ₹3299.00". Below this is a table of expense items:

DATE	DESCRIPTION	CATEGORY	AMOUNT
2025-10-05	Project Books	Shopping	₹1200.00
2025-10-04	Internet Bill	Bills	₹899.00
2025-10-03	Metro Pass	Transport	₹500.00
2025-10-02	Team Lunch	Food	₹450.00
2025-10-01	Office Supplies	Bills	₹250.00

At the bottom of the card is a purple button labeled "+ ADD NEW EXPENSE". The browser's address bar also shows "All My Expenses".

# Implementation Screenshots

## 3. Database Table in MySQL Workbench

The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Includes icons for Home, SQL, Script, Database, Tools, and Help.
- Query Editor:** Shows the following SQL code:

```
1 • USE expense_tracker;
2 • SELECT * FROM expenses;
```
- Result Grid:** Displays the data from the 'expenses' table:

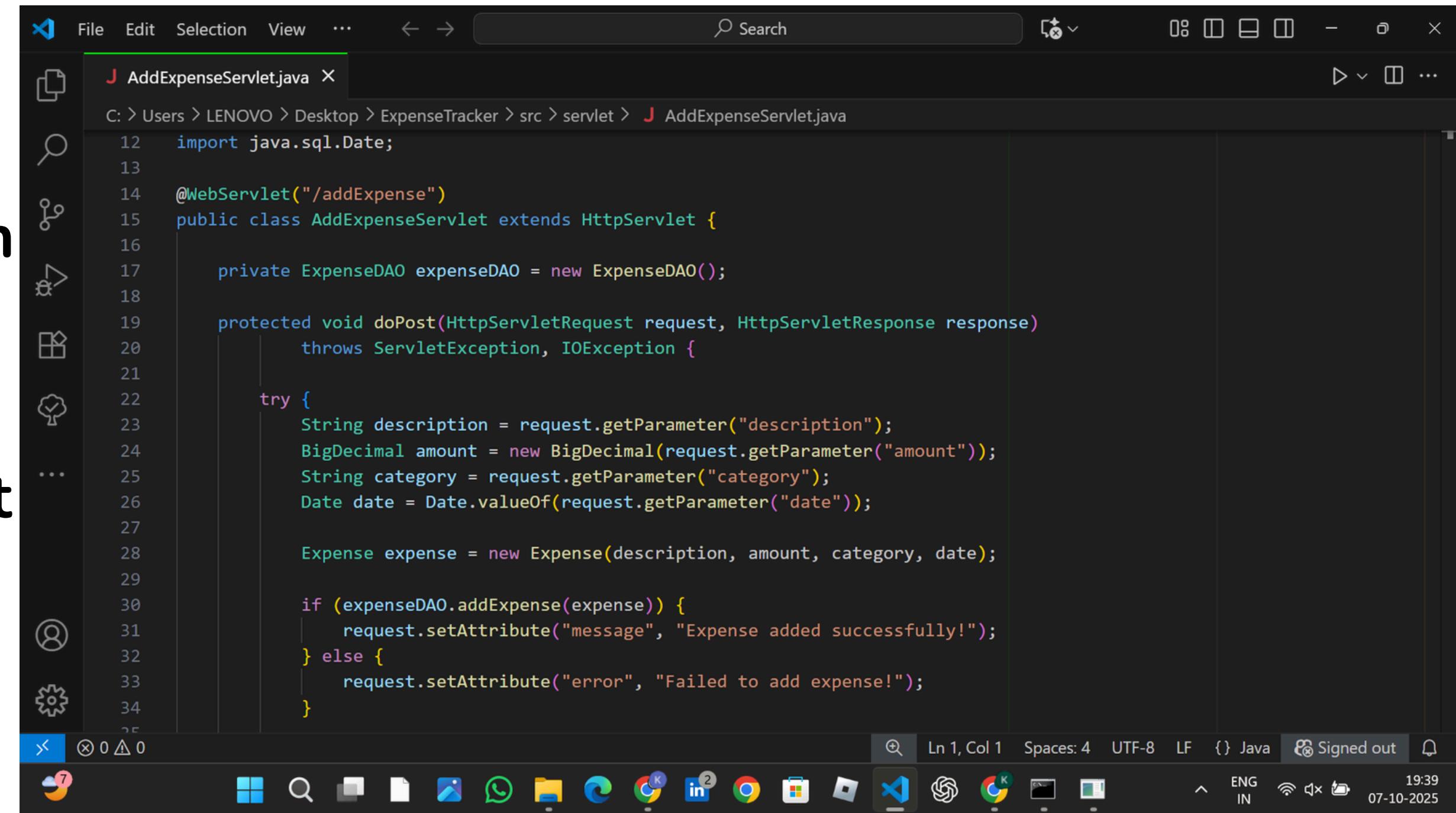
	id	description	amount	category	date	created_at
1	1	Office Supplies	250.00	Bills	2025-10-01	2025-10-04 20:40:13
2	2	Team Lunch	450.00	Food	2025-10-02	2025-10-04 20:40:13
3	3	Metro Pass	500.00	Transport	2025-10-03	2025-10-04 20:40:13
4	4	Internet Bill	899.00	Bills	2025-10-04	2025-10-04 20:40:13
5	5	Project Books	1200.00	Shopping	2025-10-05	2025-10-04 20:40:13
- Output Panel:** Shows the execution history:

#	Time	Action	Message	Duration / Fetch
1	19:46:49	USE expense_tracker	0 row(s) affected	0.000 sec
2	19:46:49	SELECT * FROM expenses LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

# Code Implementation Highlights

**Servlet Example -**  
**AddExpenseServlet:**

- **doPost()** method handles form submission
- Retrieves parameters from request
- Creates Expense object
- Calls DAO to persist data
- Forwards to JSP with status message



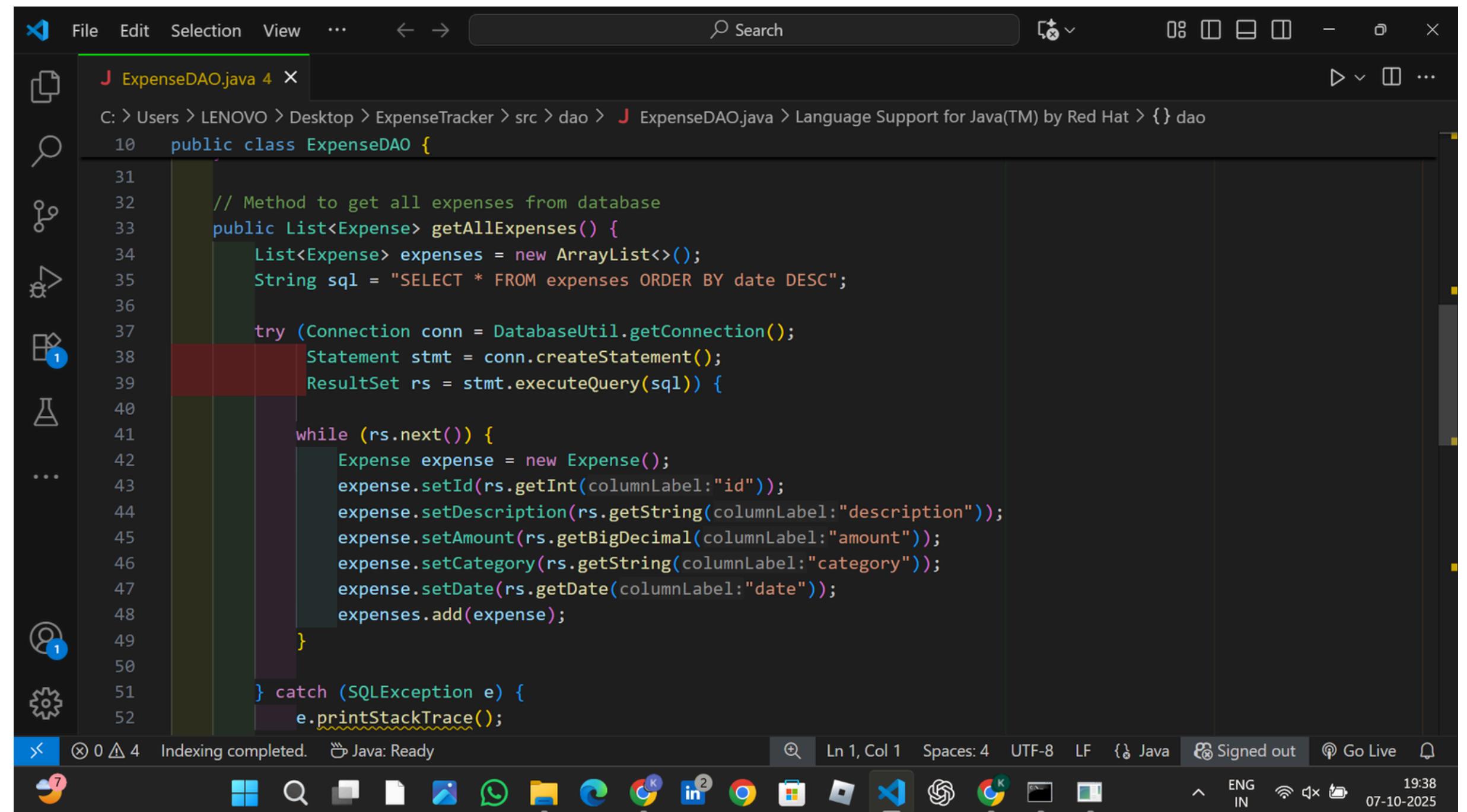
The screenshot shows a Java code editor with the file `AddExpenseServlet.java` open. The code implements a `HttpServlet` for handling POST requests to add an expense. It uses `HttpServletRequest` and `HttpServletResponse` to get parameters and set attributes. It also uses `BigDecimal` for amounts and `Date` for dates. The code is part of a project named `ExpenseTracker`.

```
File Edit Selection View ... < > Search
J AddExpenseServlet.java X
C: > Users > LENOVO > Desktop > ExpenseTracker > src > servlet > J AddExpenseServlet.java
12 import java.sql.Date;
13
14 @WebServlet("/addExpense")
15 public class AddExpenseServlet extends HttpServlet {
16
17     private ExpenseDAO expenseDAO = new ExpenseDAO();
18
19     protected void doPost(HttpServletRequest request, HttpServletResponse response)
20         throws ServletException, IOException {
21
22         try {
23             String description = request.getParameter("description");
24             BigDecimal amount = new BigDecimal(request.getParameter("amount"));
25             String category = request.getParameter("category");
26             Date date = Date.valueOf(request.getParameter("date"));
27
28             Expense expense = new Expense(description, amount, category, date);
29
30             if (expenseDAO.addExpense(expense)) {
31                 request.setAttribute("message", "Expense added successfully!");
32             } else {
33                 request.setAttribute("error", "Failed to add expense!");
34             }
35         }
36     }
37 }
```

# Code Implementation Highlights

**DAO Example - Database Operations:**

- `getConnection()` establishes DB link
- `PreparedStatement` prevents SQL injection
- `ResultSet` processes query results
- Exception handling for errors



```
File Edit Selection View ... < > Search
J ExpenseDAO.java 4 X
C: > Users > LENOVO > Desktop > ExpenseTracker > src > dao > J ExpenseDAO.java > Language Support for Java(TM) by Red Hat > {} dao
10 public class ExpenseDAO {
31
32     // Method to get all expenses from database
33     public List<Expense> getAllExpenses() {
34         List<Expense> expenses = new ArrayList<>();
35         String sql = "SELECT * FROM expenses ORDER BY date DESC";
36
37         try (Connection conn = DatabaseUtil.getConnection();
38              Statement stmt = conn.createStatement();
39              ResultSet rs = stmt.executeQuery(sql)) {
40
41             while (rs.next()) {
42                 Expense expense = new Expense();
43                 expense.setId(rs.getInt(columnLabel:"id"));
44                 expense.setDescription(rs.getString(columnLabel:"description"));
45                 expense.setAmount(rs.getBigDecimal(columnLabel:"amount"));
46                 expense.setCategory(rs.getString(columnLabel:"category"));
47                 expense.setDate(rs.getDate(columnLabel:"date"));
48                 expenses.add(expense);
49             }
50
51         } catch (SQLException e) {
52             e.printStackTrace();
}
Indexing completed. Java: Ready
Ln 1, Col 1 Spaces: 4 UTF-8 LF Java Signed out Go Live 19:38 07-10-2025
```

# Testing and Results

## Functional Testing:

- Successfully adds expenses to database
- Correctly displays all stored expenses
- Accurate total calculation
- Proper error handling

## Performance:

- Fast response time (<1 second)
- Efficient database queries
- Smooth user experience

## Browser Compatibility:

- Tested on Chrome, Firefox, Edge
- Responsive design works on different screen sizes

## Test Results



Functional Tests



Performance



Compatibility

## Test Cases

Add Expense

✓ Pass

View Expenses

✓ Pass

Total Calculation

✓ Pass

Error Handling

✓ Pass



# Conclusion



## Achievements

- ✓ Developed fully functional expense tracking system
- ✓ Implemented three-tier architecture successfully
- ✓ Integrated frontend with backend seamlessly
- ✓ Achieved persistent data storage using MySQL



## Learning Outcomes

- 💡 Practical understanding of Java web development
- 💡 Hands-on experience with servlets and JSP
- 💡 Database connectivity using JDBC
- 💡 Full-stack application deployment



## Future Enhancements

- 🔒 User authentication system
- ✏️ Delete and edit expense functionality
- 🔍 Expense filtering by date range
- 📊 Data visualization with charts
- ⬇️ Export to PDF/Excel

# References

- [1] Oracle, "Java Servlet Technology Overview", Oracle Documentation, 2024.  
<https://docs.oracle.com/javaee/7/tutorial/servlets.htm>
- [2] Oracle, "JavaServer Pages Technology", Oracle Documentation, 2024.  
<https://docs.oracle.com/javaee/7/tutorial/jspst.htm>
- [3] MySQL, "MySQL 8.0 Reference Manual", MySQL Documentation, 2024.  
<https://dev.mysql.com/doc/refman/8.0/en/>
- [4] Apache Software Foundation, "Apache Tomcat 9 Documentation", 2024.  
<https://tomcat.apache.org/tomcat-9.0-doc/>
- [5] Oracle, "JDBC Database Access", Oracle Java Tutorials, 2024.  
<https://docs.oracle.com/javase/tutorial/jdbc/>

# THANK YOU

