

# AIM:Personal Expense Tracker

## Create Expense

### POST /api/expenses

This API is used to add a new expense to the system. It accepts expense details like title, amount, category, and user ID, then saves the record in the database and returns the created expense.

The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: <http://localhost:8080/api/expenses>
- Body tab selected, showing JSON input:

```
1
2   "title": "Lunch",
3   "amount": 250,
4   "category": "Food",
5   "user_id": 5
6
7
```

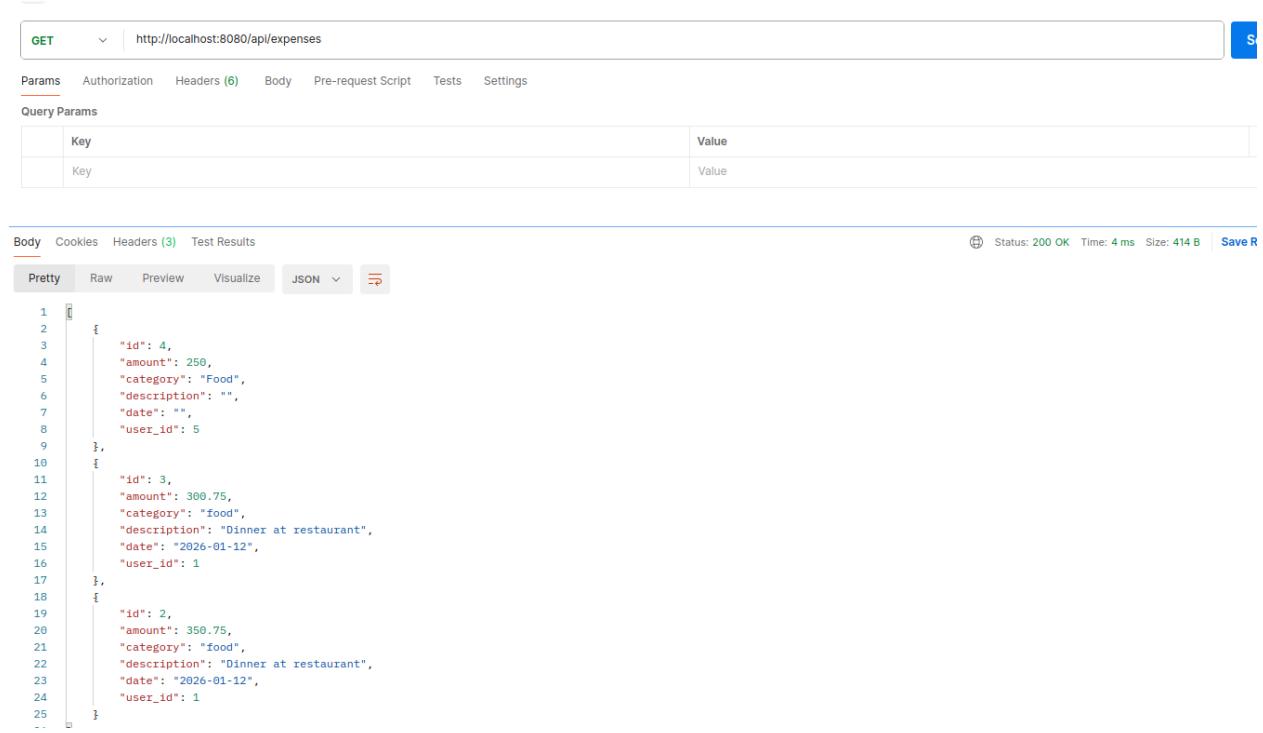
- Response tab showing JSON output:

```
1
2   "id": 4,
3   "amount": 250,
4   "category": "Food",
5   "description": "",
6   "date": "",
7   "user_id": 5
8
```

# Get All Expenses

## GET /api/expenses

This API returns a list of all expenses available in the system. It is mainly used to display expenses in dashboards or reports.



The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, the URL 'http://localhost:8080/api/expenses', and a blue 'Send' button. Below the header are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Under the 'Params' tab, there is a table titled 'Query Params' with one row: 'Key' (Value). In the main content area, there are tabs for 'Body', 'Cookies', 'Headers (3)', and 'Test Results'. The 'JSON' tab is selected, showing a pretty-printed JSON response. The response body contains three expense objects:

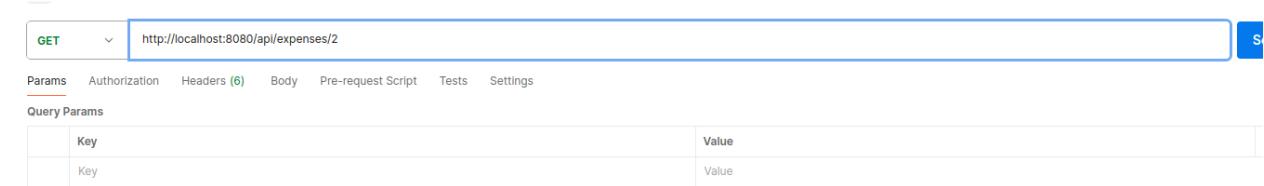
```
1 [
2   {
3     "id": 4,
4     "amount": 250,
5     "category": "Food",
6     "description": "",
7     "date": "",
8     "user_id": 5
9   },
10  {
11    "id": 3,
12    "amount": 300.75,
13    "category": "food",
14    "description": "Dinner at restaurant",
15    "date": "2026-01-12",
16    "user_id": 1
17  },
18  {
19    "id": 2,
20    "amount": 350.75,
21    "category": "food",
22    "description": "Dinner at restaurant",
23    "date": "2026-01-12",
24    "user_id": 1
25  }
```

At the top right of the main content area, there is a status bar showing 'Status: 200 OK', 'Time: 4 ms', 'Size: 414 B', and a 'Save R' button.

# Get Expense by ID

**GET /api/expenses/{id}**

This API fetches a single expense using its unique ID. It is useful when viewing the details of a specific expense.



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/expenses/2`. A query parameter `Key` is added with a value of `Key`.

---



The response body is displayed in JSON format:

```
1 "id": 2,
2   "amount": 350.75,
3   "category": "food",
4   "description": "Dinner at restaurant",
5   "date": "2026-01-12",
6   "user_id": 1
```

# Update Expense

## PUT /api/expenses/{id}

This API updates an existing expense. The expense ID is passed in the URL, and the updated values are sent in the request body.

The screenshot shows the Postman application interface. At the top, there is a header bar with the method **PUT**, the URL **http://localhost:8080/api/expenses/2**, and a blue search icon. Below the header are tabs for **Params**, **Authorization**, **Headers (8)**, **Body** (which is currently selected), **Pre-request Script**, **Tests**, and **Settings**. Under the **Body** tab, the **JSON** radio button is selected. The request body contains the following JSON:

```
1 "title": "Dinner",
2 "amount": 400,
3 "category": "Food",
4 "description": "",
5 "date": "",
6 "user_id": 1
```

At the bottom of the interface, there are tabs for **Body**, **Cookies**, **Headers (3)**, and **Test Results**. The **Test Results** tab is active, showing a status message: **Status: 200 OK Time: 12 ms Size: 187 B** and a **Save R** button. Below the status message, there are buttons for **Pretty**, **Raw**, **Preview**, **Visualize**, and **JSON**.

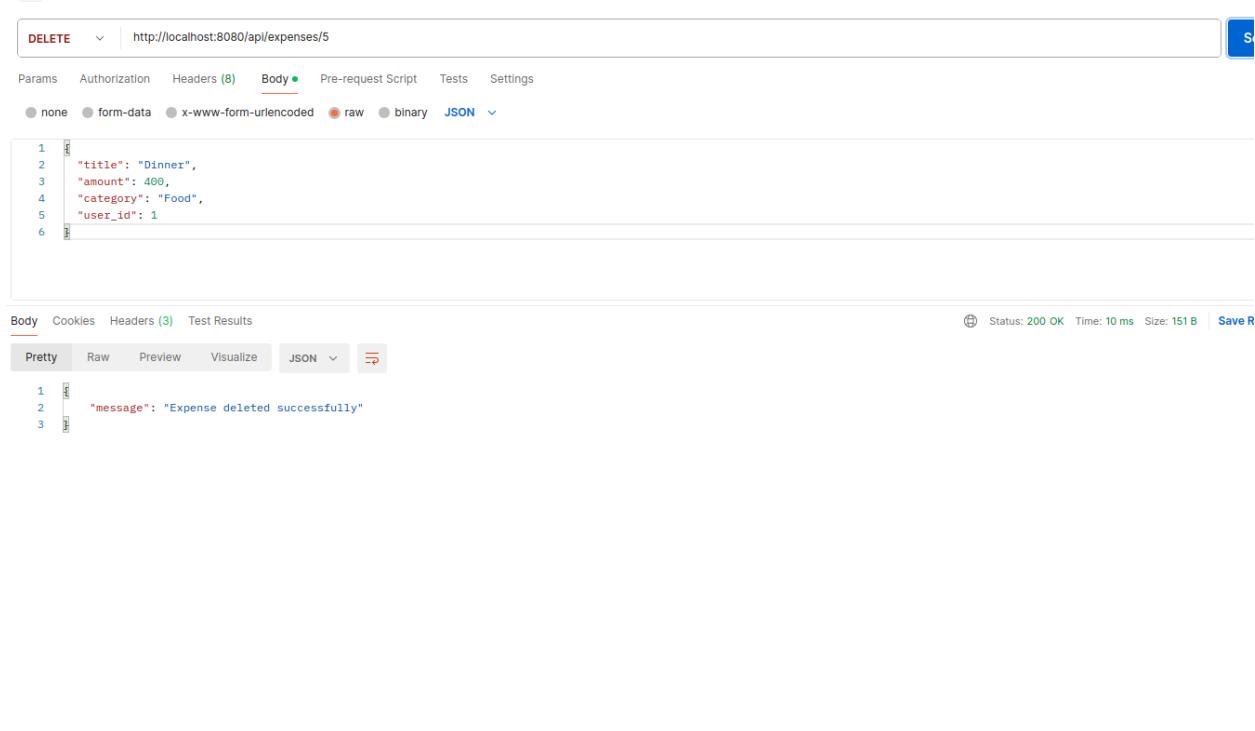
The response body is displayed below the test results, showing the updated expense object:

```
1 {
2   "id": 2,
3   "amount": 400,
4   "category": "Food",
5   "description": "",
6   "date": "",
7   "user_id": 1
8 }
```

# Delete Expense

## DELETE /api/expenses/{id}

This API deletes an expense based on its ID. Once deleted, the expense is removed from the system and a confirmation message is returned.



The screenshot shows the Postman application interface for making a DELETE request to the expense API. The URL is set to `http://localhost:8080/api/expenses/5`. The request method is `DELETE`. The `Body` tab is selected, showing a JSON payload:

```
1 "title": "Dinner",
2 "amount": 400,
3 "category": "Food",
4 "user_id": 1
```

The `Response` tab displays the following JSON response:

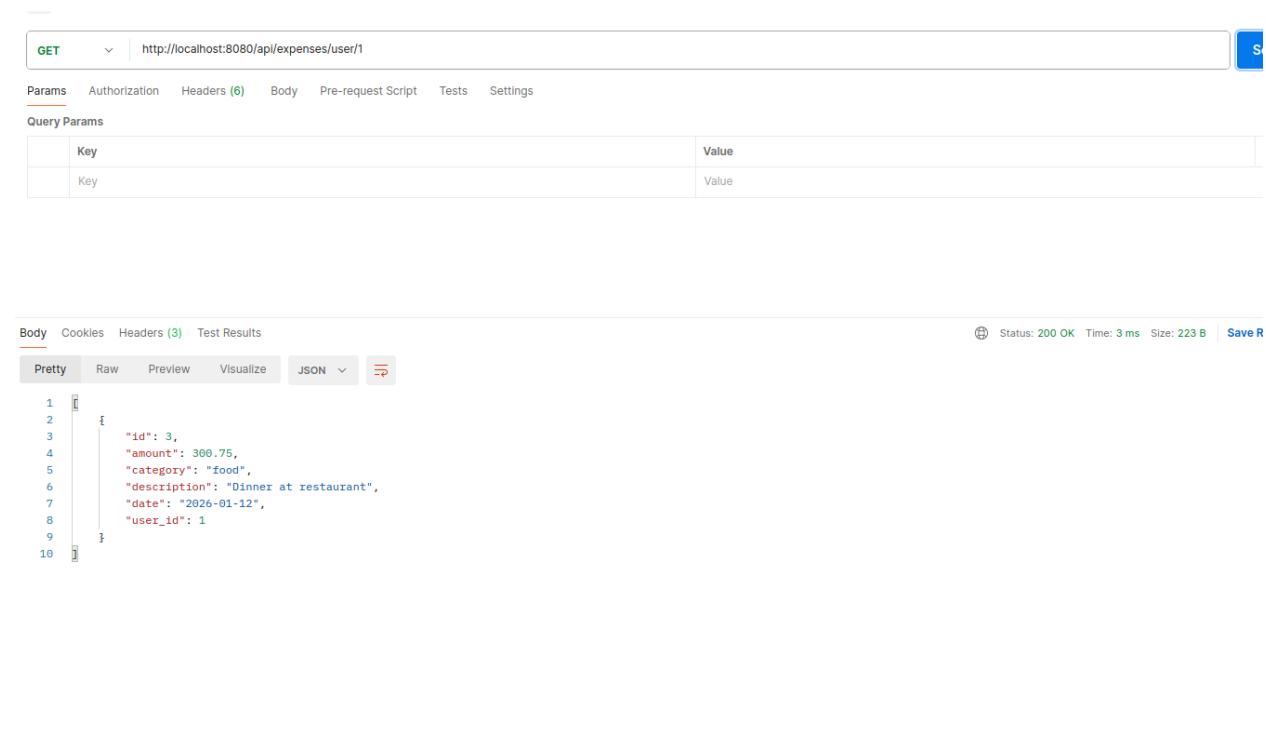
```
1 {"message": "Expense deleted successfully"}
```

Details at the top right of the response area indicate: Status: 200 OK, Time: 10 ms, Size: 151 B, and a `Save R` button.

# Get Expenses by User

**GET /api/expenses/user/{user\_id}**

This API retrieves all expenses created by a specific user. It helps in showing user-wise expense history.



The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, a URL field containing 'http://localhost:8080/api/expenses/user/1', and a blue 'Send' button. Below the header are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Under the 'Params' tab, there is a table titled 'Query Params' with one row: 'Key' (Value) and 'Value' (Value). In the main content area, there are tabs for 'Body', 'Cookies', 'Headers (3)', and 'Test Results'. The 'Body' tab is active and displays a JSON response. The response is shown in a 'Pretty' format with line numbers from 1 to 10. The JSON object has one key-value pair: 'id': 3. The detailed JSON output is as follows:

```
1  [
2   {
3     "id": 3,
4     "amount": 300.75,
5     "category": "food",
6     "description": "Dinner at restaurant",
7     "date": "2026-01-12",
8     "user_id": 1
9   }
10 ]
```

At the top right of the body panel, there is status information: 'Status: 200 OK', 'Time: 3 ms', 'Size: 223 B', and a 'Save Response' button.

# Get Expenses by Category

**GET /api/expenses/category/{category}**

This API returns all expenses that belong to a particular category, such as Food or Travel. It is commonly used for filtering and analysis.

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/expenses/category/food`. The 'category' parameter is set to 'food'. The response status is 200 OK.

The screenshot shows the Postman interface displaying the JSON response for the expense with id 3. The response is a single object with the following properties:

```
1 | {
2 |   "id": 3,
3 |   "amount": 300.75,
4 |   "category": "food",
5 |   "description": "Dinner at restaurant",
6 |   "date": "2026-01-12",
7 |   "user_id": 1
8 |
9 |
10 | }
```

The status bar indicates: Status: 200 OK Time: 11 ms Size: 223 B Save R