Expense Tracker API Documentation

Table of Contents

Overview

The Expense Tracker API provides a comprehensive backend solution for personal expense management. It includes user authentication, full CRUD operations for expenses, and powerful reporting features.

Key Features

- Secure Firebase-based authentication
- Complete expense lifecycle management
- Advanced filtering and sorting capabilities
- Monthly aggregated reports
- Category-wise expense analysis
- Input validation and error handling

Technologies Used

- **Node.js** with Express.js framework
- **MongoDB** for data persistence
- **Firebase Authentication** for user management
- **express-validator** for input validation

Authentication

All authentication endpoints use Firebase Authentication for secure user management.

1. Register User

Create a new user account with email and password.

Endpoint: /api/register
Method: POST
Authentication Required: No

Request Body

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Validation Rules

- Email: Must be valid email format
- Password: Minimum 6 characters

Success Response

Code: 201 Created

```
{
  "success": true,
  "message": "User registered successfully",
  "uid": "firebase_user_unique_id"
}
```

Error Responses

Code: 400 Bad Request

```
{
  "success": false,
  "message": "Email already registered"
}
```

Code: 400 Bad Request

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    {
      "field": "email",
      "message": "Valid email is required"
    }
  ]
}
```

---

2. Login User

Authenticate user and receive access token.

Endpoint: /api/login
Method: POST
Authentication Required: No

Request Body

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Success Response

Code: 200 OK

```
{
  "success": true,
  "message": "Login successful",
  "token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "uid": "firebase_user_unique_id"
}
```

Note: Save the token for all subsequent authenticated requests.

Error Responses

Code: 401 Unauthorized

```
{
  "success": false,
  "message": "Invalid email or password"
}
```

---

3. Logout User

Logout the authenticated user and revoke refresh tokens.

Endpoint: /api/logout
Method: POST
Authentication Required: Yes

Request Headers

```
Authorization: Bearer <your_token>
```

Success Response

Code: 200 OK

```
{
  "success": true,
  "message": "Logout successful"
}
```

Error Responses

Code: 401 Unauthorized

```
{
  "success": false,
  "message": "Unauthorized: Invalid or expired token"
}
```

Expense Management

All expense endpoints require authentication. Include the Bearer token in the Authorization header.

Authentication Header Format

Authorization: Bearer <your_token>

---

4. Create Expense

Add a new expense entry.

Endpoint: /api/expenses
Method: POST
Authentication Required: Yes

Request Body

```
{
  "title": "Lunch at Restaurant",
  "amount": 250,
  "category": "Food",
  "date": "2025-10-12"
}
```

Field Specifications

Field	Type	Required	Validation
title	String	Yes	Max 100 characters, non-empty
amount	Number	Yes	Positive number (min: 0)
category	String	Yes	Valid category from allowed list
date	String	Yes	ISO 8601 format (YYYY-MM-DD)

Valid Categories

- Food
- Travel
- Shopping
- Entertainment
- Bills
- Healthcare
- Education
- Other

Success Response

Code: 201 Created

```
{
  "success": true,
  "message": "Expense added successfully",
  "id": "670f1f77bcf86cd799439011"
}
```

Error Responses

Code: 400 Bad Request - Missing required fields

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    {
      "field": "amount",
      "message": "Amount must be a positive number"
    }
  ]
}
```

Code: 401 Unauthorized - Invalid/Missing token

```
{
  "success": false,
  "message": "Unauthorized: Invalid or expired token"
}
```

---

5. Get All Expenses

Retrieve all expenses for the authenticated user with optional filtering and sorting.

Endpoint: /api/expenses
Method: GET
Authentication Required: Yes

Query Parameters (All Optional)

Parameter	Type	Description	Example
category	String	Filter by category	?category=Food
startDate	String	Filter from date (inclusive)	?startDate=2025-10-01
endDate	String	Filter until date (inclusive)	?endDate=2025-10-31
sortBy	String	Field to sort by	?sortBy=amount
order	String	Sort order (asc/desc)	?order=desc

Example Requests

```
GET /api/expenses
GET /api/expenses?category=Food
GET /api/expenses?startDate=2025-10-01&endDate=2025-10-31
GET /api/expenses?sortBy=amount&order=desc
GET /api/expenses?category=Food&sortBy=date&order=desc
```

Success Response

Code: 200 OK

```
{
  "success": true,
  "count": 3,
  "data": [
    {
      "id": "670f1f77bcf86cd799439011",
      "title": "Lunch at Restaurant",
      "amount": 250,
      "category": "Food",
      "date": "2025-10-12T00:00:00.000Z"
    },
    {
      "id": "670f1f77bcf86cd799439012",
      "title": "Grocery Shopping",
      "amount": 1500,
      "category": "Shopping",
      "date": "2025-10-11T00:00:00.000Z"
    },
    {
      "id": "670f1f77bcf86cd799439013",
      "title": "Uber Ride",
      "amount": 300,
      "category": "Travel",
      "date": "2025-10-10T00:00:00.000Z"
    }
  ]
}
```

6. Get Single Expense

Retrieve a specific expense by its ID.

Endpoint: /api/expenses/:id
Method: GET
Authentication Required: Yes

URL Parameters

Parameter	Type	Description
id	String	MongoDB ObjectId

Example Request

```
GET /api/expenses/670f1f77bcf86cd799439011
```

Success Response

Code: 200 OK

```
{
  "success": true,
  "data": {
    "id": "670f1f77bcf86cd799439011",
    "title": "Lunch at Restaurant",
    "amount": 250,
    "category": "Food",
    "date": "2025-10-12T00:00:00.000Z"
  }
}
```

Error Responses

Code: 400 Bad Request - Invalid ID format

```
{
  "success": false,
  "message": "Invalid expense ID"
}
```

Code: 404 Not Found - Expense not found

```
{
  "success": false,
  "message": "Expense not found"
}
```

---

7. Update Expense

Update an existing expense. Only provided fields will be updated.

Endpoint: /api/expenses/:id
Method: PUT
Authentication Required: Yes

URL Parameters

Parameter	Type	Description
id	String	MongoDB ObjectId

Request Body (All fields optional)

```
{
  "title": "Dinner at Restaurant",
  "amount": 500,
  "category": "Food",
  "date": "2025-10-12"
}
```

Example Request

```
PUT /api/expenses/670f1f77bcf86cd799439011
```

Success Response

Code: 200 OK

```
{
  "success": true,
  "message": "Expense updated successfully"
}
```

Error Responses

Code: 400 Bad Request - Validation error

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    {
      "field": "category",
      "message": "Invalid category"
    }
  ]
}
```

Code: 404 Not Found - Expense not found

```
{
  "success": false,
  "message": "Expense not found"
}
```

---

8. Delete Expense

Permanently delete an expense.

Endpoint: /api/expenses/:id
Method: DELETE
Authentication Required: Yes

URL Parameters

Parameter	Type	Description
id	String	MongoDB ObjectId

Example Request

```
DELETE /api/expenses/670f1f77bcf86cd799439011
```

Success Response

Code: 200 OK

```
{
  "success": true,
  "message": "Expense deleted successfully"
}
```

Error Responses

Code: 404 Not Found - Expense not found

```
{
  "success": false,
  "message": "Expense not found"
}
```

Reports

Generate analytical reports for expense tracking.

---

9. Monthly Expense Report

Get aggregated expense summary for a specific month with category-wise breakdown.

Endpoint: /api/reports/monthly
Method: GET
Authentication Required: Yes

Query Parameters (All Required)

Parameter	Type	Description	Validation
month	Integer	Month number	1-12
year	Integer	Year	2000-2100

Example Request

```
GET /api/reports/monthly?month=10&year=2025
```

Success Response

Code: 200 OK

```
{
  "success": true,
  "data": {
    "total": 4400,
    "categories": {
      "Bills": 2000,
      "Shopping": 1200,
      "Entertainment": 500,
      "Travel": 300,
      "Food": 400
    }
  }
}
```

Response Details

- **total**: Sum of all expenses for the month
- **categories**: Object with category names as keys and total amounts as values
- Categories are sorted by amount in descending order

Error Responses

Code: 400 Bad Request - Invalid parameters

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    {
      "field": "month",
      "message": "Month must be between 1 and 12"
    }
  ]
}
```

---

10. Category-wise Expense Report

Get all expenses filtered by a specific category.

Endpoint: /api/reports/category
Method: GET
Authentication Required: Yes

Query Parameters (Required)

Parameter	Type	Description	Validation
category	String	Category name	Valid category from list

Example Request

GET /api/reports/category?category=Food

Success Response

Code: 200 OK

```
{
  "success": true,
  "count": 3,
  "data": [
    {
      "id": "670f1f77bcf86cd799439011",
      "title": "Lunch",
      "amount": 250,
      "date": "2025-10-12T00:00:00.000Z"
    },
    {
      "id": "670f1f77bcf86cd799439012",
      "title": "Breakfast",
      "amount": 100,
      "date": "2025-10-11T00:00:00.000Z"
    },
    {
      "id": "670f1f77bcf86cd799439013",
      "title": "Dinner",
      "amount": 400,
      "date": "2025-10-10T00:00:00.000Z"
    }
  ]
}
```

Error Responses

Code: 400 Bad Request - Invalid category

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    {
      "field": "category",
      "message": "Invalid category"
    }
  ]
}
```

Error Codes

The API uses standard HTTP status codes to indicate success or failure.

Status Codes

Code	Description
200	OK - Request successful
201	Created - Resource created successfully
400	Bad Request - Invalid input or validation error
401	Unauthorized - Authentication required or token invalid
404	Not Found - Resource not found
500	Internal Server Error - Server encountered an error

Error Response Format

All error responses follow this structure:

```
{
  "success": false,
  "message": "Error description",
  "errors": [
    {
      "field": "field_name",
      "message": "Specific error message"
    }
  ]
}
```

Common Error Scenarios

1. Missing Authorization Token

Request:

```
GET /api/expenses
```

Response: 401 Unauthorized

```
{
  "success": false,
  "message": "Unauthorized: No token provided"
}
```

2. Invalid Token Format

Request:

```
Authorization: Bearer invalid_token
```

Response: 401 Unauthorized

```
{
  "success": false,
  "message": "Unauthorized: Invalid or expired token"
}
```

3. Validation Errors

Request:

```
{
  "title": "",
  "amount": -50
}
```

Response: 400 Bad Request

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    {
      "field": "title",
      "message": "Title cannot be empty"
    },
    {
      "field": "amount",
      "message": "Amount must be a positive number"
    },
    {
      "field": "category",
      "message": "Category is required"
    },
    {
      "field": "date",
      "message": "Valid date is required (YYYY-MM-DD format)"
    }
  ]
}
```

Request Examples

Complete Workflow Example

Step 1: Register User

```
curl -X POST http://localhost:5000/api/register \
  -H "Content-Type: application/json" \
  -d '{
    "email": "john@example.com",
    "password": "john123456"
  }'
```

Step 2: Login

```
curl -X POST http://localhost:5000/api/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "john@example.com",
    "password": "john123456"
  }'
```

Save the returned token.

Step 3: Create Expense

```
curl -X POST http://localhost:5000/api/expenses \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_TOKEN_HERE" \
  -d '{
    "title": "Lunch at Cafe",
    "amount": 350,
    "category": "Food",
    "date": "2025-10-15"
  }'
```

Step 4: Get All Expenses

```
curl -X GET http://localhost:5000/api/expenses \
  -H "Authorization: Bearer YOUR_TOKEN_HERE"
```

Step 5: Get Monthly Report

```
curl -X GET "http://localhost:5000/api/reports/monthly?month=10&year=2025" \
  -H "Authorization: Bearer YOUR_TOKEN_HERE"
```

Step 6: Update Expense

```
curl -X PUT http://localhost:5000/api/expenses/EXPENSE_ID_HERE \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_TOKEN_HERE" \
  -d '{
    "amount": 400
  }'
```

Step 7: Delete Expense

```
curl -X DELETE http://localhost:5000/api/expenses/EXPENSE_ID_HERE \
  -H "Authorization: Bearer YOUR_TOKEN_HERE"
```

Step 8: Logout

```
curl -X POST http://localhost:5000/api/logout \
  -H "Authorization: Bearer YOUR_TOKEN_HERE"
```