app.py

Functions and Features

1. **InvestmentFund Class**
   - **Attributes**:
     - `id`: Unique identifier for the fund.
     - `name`: Name of the fund.
     - `manager_name`: Name of the fund manager.
     - `description`: Description of the fund.
     - `nav`: Net Asset Value of the fund.
     - `creation_date`: Date when the fund was created.
     - `performance`: Performance percentage of the fund.
   - **Methods**:
     - `to_dict()`: Converts the fund object to a dictionary for easy JSON serialization.
2. **API Endpoints**
   - **`GET /funds`**:
     - Retrieves a list of all funds.
     - **Function**: `get_funds()`
     - **Response**: JSON list of fund objects.
   - **`POST /funds`**:
     - Creates a new fund.
     - **Function**: `create_fund()`
     - **Request Body**: JSON object with fund details (name, manager_name, description, nav, performance).
     - **Response**: JSON object of the created fund.
   - **`GET /funds/<int:fund_id>`**:
     - Retrieves details of a specific fund by its ID.
     - **Function**: `get_fund(fund_id)`
     - **Response**: JSON object of the fund.
   - **`PUT /funds/<int:fund_id>`**:
     - Updates the performance of a specific fund by its ID.
     - **Function**: `update_fund(fund_id)`
     - **Request Body**: JSON object with updated performance.
     - **Response**: JSON object of the updated fund.
   - **`DELETE /funds/<int:fund_id>`**:
     - Deletes a specific fund by its ID.
     - **Function**: `delete_fund(fund_id)`
     - **Response**: JSON message indicating successful deletion.

migrate_data.py

Functions and Features

1. **Database Migration**
   - **Function**: Migrate data from a JSON file to the SQLite database.
   - **Steps**:
     - Connect to the SQLite database.
     - Read data from a JSON file.
     - Insert data into the database.
     - Commit the transaction and close the connection.

testing.py

Functions and Features

1. **Unit Tests**
   - **Class**: `InvestmentFundTestCase`
   - **Setup**: Initializes the test environment and creates the test database.
   - **TearDown**: Cleans up the test environment and drops the test database.
   - **Test Cases**:
     - `test_get_funds()`: Tests retrieving all funds.
     - `test_create_fund()`: Tests creating a new fund.
     - `test_get_fund()`: Tests retrieving a fund by ID.
     - `test_update_fund()`: Tests updating a fund's performance.
     - `test_delete_fund()`: Tests deleting a fund.
     - `test_migrate_data()`: Tests data migration from JSON to database.
     - `test_invalid_get_fund()`: Tests retrieving a non-existent fund.
     - `test_invalid_update_fund()`: Tests updating a non-existent fund.
     - `test_invalid_delete_fund()`: Tests deleting a non-existent fund.
     - `test_empty_create_fund()`: Tests creating a fund with missing fields.
     - `test_partial_create_fund()`: Tests creating a fund with incomplete data.

Error Handling

- **Invalid Input**: Returns `400 Bad Request` with error message.
- **Resource Not Found**: Returns `404 Not Found` with error message.
- **Internal Server Error**: Returns `500 Internal Server Error` with error message.