# **Technical Documentation for FrammerAI Contact Management System**

## **Overview**

The FrammerAI Contact Management system allows users to perform CRUD (Create, Read, Update, Delete) operations on contact data directly from a web interface. The system is built using Flask (Python) for the backend, MySQL for the database, and JavaScript with AJAX for the frontend. The application is designed to handle large datasets efficiently using pagination and supports sorting and searching functionalities.

## **Features**

1. User-Friendly Interface:
   * Simple and intuitive web interface with Bootstrap and DataTables integration.
   * SweetAlert2 for enhanced user notifications and alerts.
2. CRUD Operations:
   * Add, update, delete, and view records without refreshing the page.
3. Efficient Data Management:
   * Pagination for large datasets.
   * Sorting and searching capabilities powered by DataTables.
4. Robust Backend:
   * RESTful API endpoints implemented using Flask.
   * Secure data storage in a MySQL database.
5. Form Validation:
   * Client-side validation for Name, Email, and Phone fields.
   * AJAX-based form submission for seamless user experience.

## **System Architecture**

### **Frontend**

* Technologies: HTML, CSS, JavaScript, jQuery, Bootstrap, DataTables, SweetAlert2.
* Key File: index.html
* Functionalities:
  + A responsive form to capture Name, Email, and Phone inputs.
  + Table for displaying contact records with real-time updates using AJAX.
  + DataTables integration for sorting, searching, and pagination.

### **Backend**

* Technologies: Python (Flask), PyMySQL.
* Key Files:
  + \_\_init\_\_.py: Application factory and CORS configuration.
  + routes.py: API endpoints for CRUD operations.
  + db.py: Database connection handler.
* Functionalities:
  + RESTful API endpoints to handle client requests.
  + Database operations for storing, retrieving, updating, and deleting records.

### **Database**

* Technologies: MySQL.
* Schema:
  + Table: records
    - id (Primary Key): Auto-increment integer.
    - name: String, stores the contact's name.
    - email: String, stores the contact's email.
    - phone: String, stores the contact's phone number.

## **API Documentation**

### **Base URL**

http://<your-server-address>

### **Endpoints**

#### 1. POST /api/data

* Description: Adds a new record.
* Request Body:

json

Copy code

{

"name": "John Doe",

"email": "john.doe@example.com",

"phone": "1234567890"

}

* Response:
  + 201 Created: Record added successfully.
  + 400 Bad Request: Missing or invalid fields.

#### 2. GET /api/data

* Description: Retrieves all records.
* Response:

json

Copy code

[

{

"id": 1,

"name": "John Doe",

"email": "john.doe@example.com",

"phone": "1234567890"

}

]

#### 3. PUT /api/data/<id>

* Description: Updates a record.
* Request Body:

json

Copy code

{

"name": "Jane Doe",

"email": "jane.doe@example.com",

"phone": "0987654321"

}

* Response:
  + 200 OK: Record updated successfully.
  + 400 Bad Request: Missing or invalid fields.

#### 4. DELETE /api/data/<id>

* Description: Deletes a record.
* Response:
  + 200 OK: Record deleted successfully.

#### 5. GET /api/table\_data

* Description: Retrieves paginated data for DataTables.
* Query Parameters:
  + draw: Counter for DataTables.
  + start: Starting record index.
  + length: Number of records per page.
* Response:

json

Copy code

{

"draw": 1,

"recordsTotal": 100,

"recordsFiltered": 100,

"records": [

{

"id": 1,

"name": "John Doe",

"email": "john.doe@example.com",

"phone": "1234567890"

}

]

}

## **Setup and Deployment**

### **Requirements**

* Python 3.10+
* MySQL 8.0+

### **Installation Steps**

1. Clone the repository:

bash

Copy code

git clone <repository-url>

cd frammerai

1. Create a virtual environment:

bash

Copy code

python -m venv .venv

source .venv/bin/activate

1. Install dependencies:

bash

Copy code

pip install -r requirements.txt

1. Configure MySQL:
   * Create a database named frammerai.
   * Update the config.py file with your database credentials.
2. Run database migrations:

bash

Copy code

python manage.py db upgrade

1. Start the Flask server:

bash

Copy code

python app.py

1. Access the application at http://127.0.0.1:5000.

## **Technical Details**

### **Frontend JavaScript (scripts.js)**

* Validates form inputs before submission.
* Makes AJAX requests for CRUD operations.
* Initializes and configures DataTables.

### **Backend (Flask API)**

* Uses blueprints for modular API structure.
* Handles SQL operations with parameterized queries to prevent SQL injection.

### **Data Security**

* Inputs are validated on both the client and server sides.
* Cross-Origin Resource Sharing (CORS) is enabled for secure API access.