API Documentation: Lead Management System

Introduction

This document provides a comprehensive overview of the backend API for the **Lead Management System**. The API is built to handle the entire lifecycle of a sales lead, from creation to status updates and retrieval. It follows RESTful principles and is designed to be scalable and maintainable.

By using **TypeScript**, the project ensures type safety and enhances code quality. **Zod** is used for robust request body validation, providing clear error messages and a secure data layer. The application is containerized with **Docker** for seamless and consistent deployment across different environments.

Technologies Used

- **Runtime:** Node.js
- **Web Framework:** Express.js
- **Database:** MongoDB
- **Language:** TypeScript
- **Validation:** Zod
- **Containerization:** Docker

Getting Started with Docker

To run this application, you only need to have Docker and Docker Compose installed.

1. **Clone the repository:**
   git clone
   https://github.com/MarutiBandagar9121/LeadManagementSystem-ExpressJS.git
   cd LeadManagementSystem-ExpressJS

2. Set up environment variables:
   Create a .env file in the root directory with the following variables:
   # express application port
   PORT=3000

   # database
   MONGO_USERNAME=user
   MONGO_PASSWORD=password
   MONGO_DB_NAME=leads_db

MONGO_HOST=mongo
MONGO_PORT=27017

3. Build and run the containers:
   The provided docker-compose.yml file handles setting up both the application and the database.
   docker compose up -d --build

   The API should now be running and accessible at http://localhost:3000.

API Endpoints

This section details all the available endpoints. For each endpoint, we specify the HTTP method, the URL path, required request data, and a sample response.

Create New Lead

- **URL:** /api/leads/create
- **Method:** POST
- **Description:** Creates a new lead in the database. The request body is validated using Zod to ensure all required fields are present and correctly formatted.
- **Request Body (application/json):**

```
{
  "firstName":"Maruti",
  "lastName":"Bandagar",
  "email":"maruti@gmail.com",
  "phone":"+917755919112",
  "message":"This is an test lead",
  "source":"WEBSITE",
  "organizationName":"XYZ"
}
```

- **Success Response (Status 201):**

```
{
  "success": true,
  "data": {
    "id": "68be363c2834b0c73f46fd30"
  },
  "message": "Lead created successfully",
  "timestamp": "2025-09-08T01:49:48.272Z"
}
```

- **Error Response (Status 400):**

```
{
```

```
  "success": false,
  "error": {
    "code": "INVALID_DATA_FORMAT",
    "message": "Invalid Payload",
    "timestamp": "2025-09-08T01:49:07.142Z",
    "path": "/api/lead/create",
    "details": {
      "formErrors": [],
      "fieldErrors": {
        "email": [
          "Invlaid email Id"
        ]
      }
    }
  }
}
```


Get All Leads

- **URL:** /api/leads/get-all
- **Method:** GET
- **Description:** Retrieves a list of all leads from the database.
- **Success Response (Status 200):**


```
{
  "success": true,
  "data": [
    {
      "_id": "68bdb161dc55994e374f76e6",
      "firstName": "Basavaraj",
      "lastName": "Bandagar",
      "email": "basu@gmail.com",
      "phone": "+918855919112",
      "message": "This is an test lead",
      "source": "WEBSITE",
      "organizationName": "Estatehub",
      "leadStatus": "NEW",
      "createdAt": "2025-09-07T16:22:57.362Z",
      "updatedAt": "2025-09-07T16:22:57.362Z",
      "leadAssignedTo": "Temp User",
      "__v": 0
    },
    {
      "_id": "68be363c2834b0c73f46fd30",
```

```
            "firstName": "Maruti",
            "lastName": "Bandagar",
            "email": "maruti@gmail.com",
            "phone": "+917755919112",
            "message": "This is an test lead",
            "source": "WEBSITE",
            "organizationName": "XYZ",
            "leadStatus": "NEW",
            "createdAt": "2025-09-08T01:49:48.250Z",
            "updatedAt": "2025-09-08T01:49:48.250Z",
            "leadAssignedTo": "Temp User",
            "__v": 0
        }
    ],
    "message": "Leads fetched successfully",
    "timestamp": "2025-09-08T01:50:19.984Z"
}
```


Get Lead by ID

- **URL:** /api/leads/:id
- **Method:** GET
- **Description:** Retrieves a single lead by its unique MongoDB ID. The ID is validated to ensure it's a valid ObjectId format before a database query is made.
- **Success Response (Status 200):**


```
{
    "success": true,
    "data": {
        "_id": "68be363c2834b0c73f46fd30",
        "firstName": "Maruti",
        "lastName": "Bandagar",
        "email": "maruti@gmail.com",
        "phone": "+917755919112",
        "message": "This is an test lead",
        "source": "WEBSITE",
        "organizationName": "XYZ",
        "leadStatus": "NEW",
        "createdAt": "2025-09-08T01:49:48.250Z",
        "updatedAt": "2025-09-08T01:49:48.250Z",
        "leadAssignedTo": "Temp User",
        "__v": 0
    },
    "message": "Lead fetched successfully",
```

```
  "timestamp": "2025-09-08T01:51:43.556Z"
}
```


- **Error Response (Status 404):**


```
{
  "success": false,
  "error": {
    "code": "NOT_FOUND",
    "message": "Lead not found",
    "timestamp": "2025-09-08T01:52:21.760Z",
    "path": "/api/lead/68be363c2834b0c73f46fd35",
    "details": ""
  }
}
```


Update Lead Status

- **URL:** /api/leads/update-Status
- **Method:** POST
- **Description:** Updates the status of an existing lead. The request payload is validated using Zod to ensure a valid lead ID and one of the predefined status values.
- **Request Body (application/json):**


```
{
  "id":"68be363c2834b0c73f46fd30",
  "status":"PROPOSAL_SENT"
}
```


- **Success Response (Status 200):**


```
{
  "success": true,
  "data": {
    "id": "68be363c2834b0c73f46fd30",
    "message": "Lead status updated successfully",
    "status": "PROPOSAL_SENT"
  },
  "message": "Lead status updated successfully",
  "timestamp": "2025-09-08T01:53:01.856Z"
}
```


- **Error Response (Status 400):**


```
{
  "success": false,
  "error": {
```

```
    "code": "INVALID_DATA_FORMAT",
    "message": "Invalid Payload",
    "timestamp": "2025-09-08T01:53:38.015Z",
    "path": "/api/lead/update-Status",
    "details": {
      "formErrors": [],
      "fieldErrors": {
        "status": [
          "Status is invalid"
        ]
      }
    }
  }
}
```


Get All Leads by Status

- **URL:** /api/leads/get-all-by-status/:status
- **Method:** GET
- **Description:** Retrieves all leads that match the specified status. The status value provided in the URL parameter is validated against a predefined enum.
- **Success Response (Status 200):**


```
{
  "success": true,
  "data": [
    {
      "_id": "68be363c2834b0c73f46fd30",
      "firstName": "Maruti",
      "lastName": "Bandagar",
      "email": "maruti@gmail.com",
      "phone": "+917755919112",
      "message": "This is an test lead",
      "source": "WEBSITE",
      "organizationName": "XYZ",
      "leadStatus": "PROPOSAL_SENT",
      "createdAt": "2025-09-08T01:49:48.250Z",
      "updatedAt": "2025-09-08T01:49:48.250Z",
      "leadAssignedTo": "Temp User",
      "__v": 0
    }
  ],
  "message": "Leads fetched successfully",
  "timestamp": "2025-09-08T01:55:04.916Z"
```

}

- **Error Response (Status 400):**


```
{
  "success": false,
  "error": {
    "code": "INVALID_DATA_FORMAT",
    "message": "Invalid Payload",
    "timestamp": "2025-09-08T01:55:31.909Z",
    "path": "/api/lead/get-all-by-status/PROPOSAL_SEN",
    "details": {
      "formErrors": [
        "Invalid option: expected one of
\"NEW\"|\"CONTACTED\"|\"QUALIFIED\"|\"PROPOSAL_SENT\"|\"NEGOTIATION\"|\"WON\"|
\"LOST\"|\"ON_HOLD\"|\"UNQUALIFIED\""
      ],
      "fieldErrors": {}
    }
  }
}
```


Data Validation with Zod

Data integrity is crucial. The API uses Zod to define schemas for all incoming request bodies, ensuring that only valid and well-structured data is processed. This prevents common vulnerabilities and maintains a clean database.

Lead Status Enum

```
enum LeadStatusEnum {
  NEW = "NEW",
  CONTACTED = "CONTACTED",
  QUALIFIED = "QUALIFIED",
  PROPOSAL_SENT = "PROPOSAL_SENT",
  NEGOTIATION = "NEGOTIATION",
  WON = "WON",
  LOST = "LOST",
  ON_HOLD = "ON_HOLD",
  UNQUALIFIED = "UNQUALIFIED",
}
```

Zod Schemas

```
export const createLeadSchema = z.object({
  firstName: z.string().min(1, "First name is required"),
  lastName: z.string().optional(),
  email: z.email("Invlaid email Id"),
  phone: z.string(),
  message: z.string().optional(),
  source: z.string().optional(),
  organizationName: z.string().optional(),
});
```

Project Structure

The project follows a modular and organized structure to enhance maintainability:

- **/src**: Contains all source code.
  - **/controller**: Logic for handling requests and responses.
  - **/routes**: Defines API endpoints.
  - **/models**: MongoDB schema definitions.
  - **/validator**: Zod schemas for validation.
- **/dist**: Transpiled JavaScript code.
- **docker-compose.yml**: Docker configuration.
- **.env.example**: Environment variables example file
- **package.json**: Project dependencies and scripts.

Conclusion

This project demonstrates a strong command of modern backend development principles, including robust data validation, containerization for easy deployment, and maintainable code architecture with TypeScript. This documentation will serve as a clear and professional guide for anyone looking to understand and interact with your API.