Here's a structured breakdown of the required and additional service endpoints, along with descriptions, example requests/responses, and a communication flow diagram for both the MVP and stretch features.

1. Service Endpoints (MVP)

1.1 Authentication Endpoints

- `POST /api/auth/signup`
  - **Purpose**: Registers a new user account.

Request:
```
{
  "username": "john_doe",
  "email": "john@example.com",
  "password": "securePassword123"
}
```

  -

Response (Success):
```
{
  "user_id": "12345",
  "message": "User registered successfully"
}
```

  -

Response (Error):
```
{
  "error": "Email already in use"
}
```

  -
- `POST /api/auth/login`
  - **Purpose**: Authenticates a user and generates a session token.

Request:
```
{
  "email": "john@example.com",
  "password": "securePassword123"
}
```

- ○

Response (Success):

```
{
  "token": "jwt_token_here",
  "user_id": "12345"
}
```

- ○

Response (Error):

```
{
  "error": "Invalid credentials"
}
```

- ○

1.2 Lead Management Endpoints

- **`GET /api/leads`**
  - **Purpose**: Retrieves a list of all leads for the authenticated user.

Response (Success):

```
[
  {
    "lead_id": "lead123",
    "name": "Jane Doe",
    "contact_info": "jane@example.com",
    "score": 10
  },
  ...
]
```

- ○

Response (Error):

```
{
  "error": "Unauthorized access"
}
```

- ○

- `POST /api/leads`
  - **Purpose**: Adds a new lead.

Request:
```
{
  "name": "Jane Doe",
  "contact_info": "jane@example.com"
}
```

  -

Response (Success):
```
{
  "lead_id": "lead123",
  "message": "Lead added successfully"
}
```

  -

Response (Error):
```
{
  "error": "Invalid lead data"
}
```

  -

1.3 Inquiry Endpoints

- `POST /api/leads/{lead_id}/inquiries`
  - **Purpose**: Adds a new inquiry for a specific lead, updating the lead’s score.

Request:
```
{
  "message": "I would like more information about the property"
}
```

  -

Response (Success):
```
{
  "inquiry_id": "inquiry123",
  "new_score": 15,
  "message": "Inquiry added and score updated"
}
```

- ◦

Response (Error):

```
{
  "error": "Lead not found"
}
```

- ◦

2. Service Endpoints (Stretch Feature)

2.1 Analytics and Reminders Endpoints

- **`GET /api/leads/{lead_id}/analytics`**
  - **Purpose**: Retrieves a list of interactions and analytic events for a specific lead.

Response (Success):

```
[
  {
    "event_type": "page_view",
    "event_value": "/property/123",
    "event_date": "2024-11-04T10:00:00Z"
  },
  ...
]
```

- ◦

Response (Error):

```
{
  "error": "Lead not found"
}
```

- ◦
- **`POST /api/leads/{lead_id}/reminders`**
  - **Purpose**: Sets a follow-up reminder for a lead.

Request:

```
{
  "reminder_date": "2024-11-10T09:00:00Z",
  "reminder_message": "Follow up with lead about pricing details"
}
```

◦

Response (Success):

```
{
  "reminder_id": "reminder123",
  "message": "Reminder added successfully"
}
```

◦

Response (Error):

```
{
  "error": "Invalid reminder data"
}
```

◦

3. Example Error Cases

For erroneous requests, typical responses include:

- **Validation Errors**:
  - Status: `400 Bad Request`

Example Response:

```
{
  "error": "Missing required field"
}
```

◦

- **Authentication Errors**:
  - Status: `401 Unauthorized`

Example Response:

```
{
  "error": "Invalid token"
}
```

◦

- **Resource Not Found**:
  - Status: `404 Not Found`

Example Response:

```
{
  "error": "Lead not found"
}
```

○

4. Communication Flow Diagram

The flowchart below represents how the UI pages interact with each of these endpoints:

```
User Interface Pages
┌──────────────────────────────────────────────┐
│            Login Page                        │
│            ────────                          │
│  - POST /api/auth/signup                     │
│  - POST /api/auth/login                      │
└──────────────────────────────────────────────┘
          │
          ▼
┌──────────────────────────────────────────────┐
│            Dashboard Page                    │
│            ──────────────                    │
│  - GET /api/leads                            │
│  - POST /api/leads                           │
└──────────────────────────────────────────────┘
          │
          ▼
┌──────────────────────────────────────────────┐
│            Lead Details Page                 │
│            ────────────────                  │
│  - POST /api/leads/{lead_id}/inquiries       │
│  - GET /api/leads/{lead_id}/analytics        │
│  - POST /api/leads/{lead_id}/reminders       │
└──────────────────────────────────────────────┘
```

This diagram represents the interaction paths for the user:

- Users authenticate through the login page.

- The dashboard page allows users to retrieve and manage leads.
- Within a specific lead’s detail view, users can submit inquiries (MVP), view analytics, and set reminders (stretch features).

This structure, using RESTful principles, abstracts the backend interactions and maintains a clear separation between the client and the database layer. Each endpoint is designed to support specific user goals while ensuring that data management tasks are organized and scalable.