**Service Layers**

**Authentication:**

* POST /api/auth/login: Authenticate users based on username and password.
* POST /api/auth/logout: Log out the currently authenticated user.

**LOGIN PAGE >< API >< PostgreSQL**

**Login:**

* Purpose: Allowing users to log into the system.
* Example Request:

{

"username": "receptionistKatie",

"password": "password123"

}

**Logout:**

* Purpose: Allowing users to log out of the system.
* Example Response:

{

"message": "Logout successful"

}

**Appointments:**

* GET /api/appointments: Retrieve a list of appointments.
* POST /api/appointments: Create a new appointment.
* PUT /api/appointments/:appointmentId: Update an existing appointment.
* DELETE /api/appointments/:appointmentId: Cancel an appointment.

**A screenshot of a computer

Description automatically generated>< API >< PostgreSQL**

**Retrieve Appointments:**

* Purpose: Providing a list of appointments to the user interface.
* Example Request: Internal
* Example Response:

{

"appointments": [

{

"appointment\_id": 1,

"date": "2023-11-10",

"time": "10:00 AM",

"patient\_id": 1,

"doctor\_id": 2,

"status": "scheduled"

}

]

}

**Create Appointment:**

* Purpose: Allowing users to schedule new appointments.
* Example Request:

{

"date": "2023-11-15",

"time": "03:00 PM",

"patient\_id": 2,

"doctor\_id": 2

}

**Patients:**

* GET /api/patients: Retrieve a list of patients.
* POST /api/patients: Create a new patient record.
* PUT /api/patients/:patientId: Update an existing patient record.
* GET /api/patients/search?query=<search\_query>: Search for patients based on name, species, breed, or other relevant attributes.

**A screenshot of a computer

Description automatically generated>< API >< PostgreSQL**

**Retrieve Patients:**

* Purpose: Providing a list of patients to the user interface.
* Example Request: Internal
* Example Response:

{

"patients": [

{

"patient\_id": 1,

"name": "Ash",

"species": "Dog",

"breed": "Golden Retriever",

"age": 5,

"medical\_history": {

"vaccinations": "Up-to-date",

"allergies": "None",

"conditions": "Healthy"

},

"client\_id": 1

},

**Create Patient:**

* Purpose: Allowing users to add new patient records.
* Example Request:

{

"name": "Rex",

"species": "Dog",

"breed": "Labrador",

"age": 3

}

**Search Patients:**

* Purpose: Allowing users to search for patients based on various criteria.
* Example Request: GET /api/patients/search?query=Ash

**Clients:**

* GET /api/clients: Retrieve a list of clients.
* POST /api/clients: Create a new client record.
* PUT /api/clients/:clientId: Update an existing client record.
* GET /api/clients/search?query=<search\_query>: Search for clients based on name or contact information.

**A screenshot of a computer

Description automatically generated>< API >< PostgreSQL**

**Retrieve Clients:**

* Purpose: Retrieve a list of clients.
* Example Response:

{

"clients": [

{

"client\_id": 1,

"name": "John Doe",

"contact\_info": "123 Main St, Anytown, USA"

},

{

"client\_id": 2,

"name": "Jane Smith",

"contact\_info": "456 Elm St, Anytown, USA"

}

]

}

**Create Client Record:**

* Purpose: Create a new client record.
* Example Request:

{

"name": "New Client",

"contact\_info": "789 Oak St, Anytown, USA"

}

**Update Client Record:**

* Purpose: Update an existing client record.
* Example Request:

{

"name": "Updated Client Name",

"contact\_info": "New Address"

}

**Search Clients:**

* Purpose: Allowing users to search for clients based on various criteria.
* Example Request:
  + GET /api/clients/search?query=John Doe
* Example Response:

{

"clients": [

{

"client\_id": 1,

"name": "John Doe",

"contact\_info": "123 Main St, Springfield, USA"

}

]

}

**Communication Diagram:**

USER INTERFACE >< API >< DATABASE

* The User Interface communicates with specific Service Layers based on user actions and requirements.
* The Service Layers process requests, handle business logic, and interact with the Database to retrieve or modify data.
* Responses from the Database are sent back through the Service Layers to the User Interface for display or further actions.

By adhering to this service layer architecture, VetsPlace ensures a clear separation between the user interface and the database. Changes in either the user interface or the database can be made independently.