Detailed Architecture and Code for MVP Development with Amazon Platform

1. Architecture Overview

For the MVP of the pet social network, the architecture will focus on scalability, performance, and ease of deployment. The architecture will be built using a microservices approach, hosted on a cloud platform (AWS or Azure), with a robust backend to handle user profiles, social interactions, and AI-driven functionalities. The front end will be developed using modern web technologies, ensuring responsiveness and a seamless user experience across devices.

Cloud Platform: AWS or Azure (depending on cost-effectiveness and existing infrastructure).

- **Compute:** AWS Lambda / Azure Functions for serverless operations, EC2 instances for microservices.
- **Storage:** AWS S3 / Azure Blob Storage for media, RDS / Azure SQL for relational data.
- **Database:** PostgreSQL for relational data (user profiles, posts), Redis for caching, and Elasticsearch for search functionality.
- **AI/ML Services:** AWS SageMaker / Azure ML for AI/ML models.
- **Authentication:** AWS Cognito / Azure AD B2C for user authentication and management.
- **APIs:** RESTful API for data communication between front-end and back-end services, API Gateway for routing.
- **Front-End Framework:** React.js for building responsive UI, with Redux for state management.
- **Back-End Framework:** Node.js with Express.js for REST API development.
- **DevOps:** CI/CD pipeline using Jenkins, GitHub Actions, or Azure DevOps.

2. Detailed Architecture Components

2.1. Core Table: Pets

Design: The Pets table will be the core table, with all other tables linked via foreign keys to ensure data integrity and facilitate complex queries.

Database Schema:

- **sql**
- **CREATE TABLE Pets (**
- PetId UUID PRIMARY KEY,
- OwnerId UUID FOREIGN KEY REFERENCES Users(UserId),
- Name VARCHAR(100),

- Breed VARCHAR(100),
- Age INT,
- Bio TEXT,
- ProfilePictureUrl VARCHAR(255),
- CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
- UpdatedAt TIMESTAMP
- **);**
-
- **CREATE TABLE Users (**
- UserId UUID PRIMARY KEY,
- Name VARCHAR(100),
- Email VARCHAR(100) UNIQUE,
- PasswordHash VARCHAR(255),
- ProfilePictureUrl VARCHAR(255),
- CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
- UpdatedAt TIMESTAMP
- **);**
-
- **CREATE TABLE Roles (**
- RoleId UUID PRIMARY KEY,
- RoleName VARCHAR(50)
- **);**
-
- **CREATE TABLE UserRoles (**
- UserRoleId UUID PRIMARY KEY,
- UserId UUID FOREIGN KEY REFERENCES Users(UserId),
- RoleId UUID FOREIGN KEY REFERENCES Roles(RoleId)
- **);**
-

- **CREATE TABLE HealthRecords (**
- RecordId UUID PRIMARY KEY,
- PetId UUID FOREIGN KEY REFERENCES Pets(PetId),
- VaccinationDetails JSONB,
- Allergies JSONB,
- MedicalHistory TEXT,
- CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
- UpdatedAt TIMESTAMP
- **);**
-
- **CREATE TABLE PetVaccinations (**
- VaccinationId UUID PRIMARY KEY,
- PetId UUID FOREIGN KEY REFERENCES Pets(PetId),
- VaccineName VARCHAR(100),
- DateAdministered DATE,
- NextDueDate DATE,
- CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
- UpdatedAt TIMESTAMP
- **);**
-
- **CREATE TABLE PetCommunities (**
- CommunityId UUID PRIMARY KEY,
- CommunityName VARCHAR(100),
- Description TEXT,
- CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
- UpdatedAt TIMESTAMP
- **);**
-
- **CREATE TABLE Marketplaces (**

- MarketplaceId UUID PRIMARY KEY,
- Name VARCHAR(100),
- Description TEXT,
- CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
- UpdatedAt TIMESTAMP
- **);**
-
- **CREATE TABLE Routes (**
- RouteId UUID PRIMARY KEY,
- PetId UUID FOREIGN KEY REFERENCES Pets(PetId),
- RouteName VARCHAR(100),
- PathCoordinates JSONB, -- GPS coordinates stored as JSON
- CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
- UpdatedAt TIMESTAMP
- **);**

User Profiles for Pets and Owners

- Backend Microservice: User Service
- *Functions*: Handles user and pet profile creation, updates, and retrievals.
- *Technology*: Node.js with Express.js, using Amazon RDS for data storage.
- ***Endpoints*:**
  - POST /api/users: Create a new user profile.
  - POST /api/pets: Create a new pet profile associated with a user.
  - GET /api/users/{userId}: Retrieve user profile details.
  - GET /api/pets/{petId}: Retrieve pet profile details.
  - PUT /api/users/{userId}: Update user profile.
  - PUT /api/pets/{petId}: Update pet profile.
  - DELETE /api/users/{userId}: Delete user profile.
  - DELETE /api/pets/{petId}: Delete pet profile.

2.2. **ML Model**: Predictive analytics for pet health based on profile data.

Implementation:

Use AWS SageMaker to develop and deploy a model that predicts potential health issues based on breed and age.

Example endpoint in Python using Flask for serving the ML model:

python

```
@app.route('/api/v1/predict_health', methods=['POST'])
def predict_health():
    data = request.get_json()
    breed = data['breed']
    age = data['age']
    prediction = model.predict([[breed, age]])
    return jsonify({'health_risk': prediction})
```

3. **Advanced Pet Profiles**

3.1. **Backend Microservice**: Pet Health Service

3.1.1. *Functions*: Handles additional health-related data storage, vaccination records, and breed certifications.

3.1.2. *Technology*: Node.js with Express.js, PostgreSQL for relational data, Redis for caching frequently accessed health data.

3.1.3. *Endpoints*:

3.1.3.1. POST /api/v1/pets/{petId}/health: Add health records for a pet.

3.1.3.2. GET /api/v1/pets/{petId}/health: Retrieve health records for a pet.

3.1.3.3. PUT /api/v1/pets/{petId}/health: Update health records.

3.1.3.4. DELETE /api/v1/pets/{petId}/health: Delete health records.

3.2. **Database Schema**:

sql

```
CREATE TABLE PetHealthRecords (
    recordId SERIAL PRIMARY KEY,
    petId INT REFERENCES Pets(petId),
    vaccinationDetails JSONB,
    allergies JSONB,
```

```
  medicalHistory TEXT,
  createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updatedAt TIMESTAMP
);
```

3.3. **AI Integration**:

3.3.1. **ML Model**: Predictive health analytics and personalized care recommendations.

3.3.2. **Implementation**:

3.3.2.1. Utilize Azure ML to create a model that offers care recommendations based on historical health data and current trends.

3.3.2.2. Integrate with existing health APIs to fetch real-time data for improved predictions.

4. **Front-End Development**

4.1. **UI/UX Design**:

4.1.1. ***Technology***: React.js with Material-UI for component-based design.

4.1.2. *Components*:

4.1.2.1. **Profile Page**: Display user and pet profiles with editable sections.

4.1.2.2. **Health Dashboard**: Visualize health records with charts and notifications for upcoming vaccinations or health checks.

4.1.2.3. **Community Page**: Allows users to join and interact within pet communities.

4.2. **VR/AR Integration**

4.2.1. **Amazon Sumerian**: Used to create immersive VR/AR experiences within the platform.

4.2.2. *Use Case*: Virtual pet adoption fairs, AR-based pet training guides.

4.2.3. *Integration*: Embed Amazon Sumerian scenes within the React.js front-end using sumerian-host:

```
javascript
<iframe
    src="https://sumerian.amazonaws.com/sumerian-scene-id"
    width="100%"
    height="500"
    allow="fullscreen">
</iframe>
```

4.3. **State Management**:

4.3.1. *Technology*: Redux for managing the state across components.

4.3.2. *Example*:

javascript

```
const initialState = {
  user: {},
  pets: [],
  healthRecords: {}
};

function rootReducer(state = initialState, action) {
  switch (action.type) {
    case 'SET_USER':
      return { ...state, user: action.payload };
    case 'ADD_PET':
      return { ...state, pets: [...state.pets, action.payload] };
    case 'SET_HEALTH_RECORDS':
      return { ...state, healthRecords: action.payload };
    default:
      return state;
  }
}
```

4.4. **API Integration**:

4.4.1. Use Axios for making API calls to the backend services.

4.4.2. Example API call:

javascript

```
axios.post('/api/v1/users', userData)
  .then(response => {
```

```
    console.log('User created:', response.data);
  })
  .catch(error => {
    console.error('Error creating user:', error);
  });
```

5. **DevOps and CI/CD Pipeline**
   5.1. **Continuous Integration**: Implemented using GitHub Actions for automatic testing and deployment.
   5.2. **Continuous Deployment**: Jenkins or Azure DevOps to deploy the microservices on AWS or Azure.
   5.3. **Monitoring and Logging**: Use AWS CloudWatch or Azure Monitor for real-time monitoring and logging.
   5.4. **Sample CI/CD Pipeline Configuration** (using GitHub Actions):

```
name: CI/CD Pipeline
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Set up Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'
    - name: Install dependencies
      run: npm install
    - name: Run tests
      run: npm test
    - name: Deploy to AWS
```

```
if: github.ref == 'refs/heads/main'
run: |
  npm run build
  aws s3 sync build/ s3://your-bucket-name --delete
```

PSEUDOCODE FOR MVP FRONTEND AND BACKEND

To create an attractive, friendly, and innovative front end for the website and mobile app for the MVP phase of the pet social network, we can focus on modern design principles, intuitive navigation, and seamless user experience (UX). The design will be built using React.js for the web front end and React Native for the mobile app. Here's a breakdown of the core screens, design concepts, and sample code for each screen.

Design Concepts

1. User-Centric Design: Prioritize ease of use with a clean and minimalistic interface.
2. Responsive and Adaptive: Ensure the design works seamlessly across various devices and screen sizes.
3. Innovative Features: Incorporate modern UI elements like animated transitions, micro-interactions, and augmented reality (AR) components for an engaging experience.
4. Pet-Friendly Themes: Use colors, fonts, and imagery that resonate with pet lovers, with playful icons and animations.

Core Screens and Code for MVP

1. Landing Page (Website)

Objective: Introduce the platform with an engaging hero section, clear call-to-actions (CTAs), and highlights of key features.

Design Elements:

- Hero Section: Full-width background image of pets, a welcoming message, and sign-up/login buttons.
- Feature Highlights: Icons with brief descriptions of core features.
- CTA Buttons: "Join Now" and "Learn More" buttons.

Sample Code (React.js):

javascript

```
import React from 'react';

import './LandingPage.css';  // Import custom CSS for styling


const LandingPage = () => {

  return (

    <div className="landing-page">

      <header className="hero-section">
```

```
      <div className="hero-text">
        <h1>Welcome to PetConnect</h1>
        <p>Your social network for pets</p>
        <button className="cta-button">Join Now</button>
        <button className="cta-button secondary">Learn More</button>
      </div>
    </header>

    <section className="features">
      <div className="feature">
        <img src="/images/profile-icon.png" alt="Profile Icon" />
        <h3>Create Pet Profiles</h3>
        <p>Showcase your pet’s personality with custom profiles.</p>
      </div>
      <div className="feature">
        <img src="/images/community-icon.png" alt="Community Icon" />
        <h3>Join Communities</h3>
        <p>Connect with other pet owners and share experiences.</p>
      </div>
      <div className="feature">
        <img src="/images/marketplace-icon.png" alt="Marketplace Icon" />
        <h3>Explore the Marketplace</h3>
        <p>Find the best products and services for your pet.</p>
      </div>
    </section>
  </div>
 );
};
```

```
export default LandingPage;
```

CSS (LandingPage.css):

```
css
.landing-page {
  font-family: 'Roboto', sans-serif;
  color: #333;
}

.hero-section {
  background-image: url('/images/hero-bg.jpg');
  background-size: cover;
  text-align: center;
  padding: 100px 20px;
}

.hero-text {
  max-width: 600px;
  margin: 0 auto;
  color: #fff;
}

.cta-button {
  background-color: #ff6b6b;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 25px;
  cursor: pointer;
  margin: 10px;
```

```
}

.cta-button.secondary {
  background-color: #ffa07a;
}

.features {
  display: flex;
  justify-content: space-around;
  padding: 50px 20px;
}

.feature {
  text-align: center;
  max-width: 300px;
}

.feature img {
  width: 50px;
  height: 50px;
  margin-bottom: 15px;
}
```

2. User Profile (Web and Mobile)

Objective: Allow users to manage their profile and their pet's profile with easy-to-use forms and editable fields.

Design Elements:

- Profile Picture Upload: Users can upload images of their pets.
- Editable Fields: Name, Breed, Age, Bio, etc.
- Save/Cancel Buttons: Save changes or cancel edits.

Sample Code (React.js for Web):

javascript

```
import React, { useState } from 'react';

const UserProfile = () => {
  const [profile, setProfile] = useState({
    name: '',
    breed: '',
    age: '',
    bio: '',
    profilePicture: ''
  });

  const handleChange = (e) => {
    setProfile({
      ...profile,
      [e.target.name]: e.target.value
    });
  };

  const handleSubmit = () => {
    // Handle form submission
  };

  return (
    <div className="profile-page">
      <h2>Edit Pet Profile</h2>
      <form onSubmit={handleSubmit}>
        <div className="form-group">
```

```
  <label>Pet Name</label>
  <input
    type="text"
    name="name"
    value={profile.name}
    onChange={handleChange}
  />
</div>
<div className="form-group">
  <label>Breed</label>
  <input
    type="text"
    name="breed"
    value={profile.breed}
    onChange={handleChange}
  />
</div>
<div className="form-group">
  <label>Age</label>
  <input
    type="number"
    name="age"
    value={profile.age}
    onChange={handleChange}
  />
</div>
<div className="form-group">
  <label>Bio</label>
  <textarea
```

```
          name="bio"
          value={profile.bio}
          onChange={handleChange}
        />
      </div>
      <div className="form-group">
        <label>Profile Picture</label>
        <input
          type="file"
          name="profilePicture"
          onChange={handleChange}
        />
      </div>
      <button type="submit" className="save-button">Save</button>
      <button type="button" className="cancel-button">Cancel</button>
    </form>
  </div>
 );
};

export default UserProfile;
```

Sample Code (React Native for Mobile):

```
javascript
import React, { useState } from 'react';
import { View, Text, TextInput, Button, Image, TouchableOpacity } from 'react-native';

const UserProfile = () => {
 const [profile, setProfile] = useState({
   name: '',
```

```
  breed: '',
  age: '',
  bio: '',
  profilePicture: ''
});

const handleChange = (field, value) => {
  setProfile({
    ...profile,
    [field]: value
  });
};

return (
  <View style={{ padding: 20 }}>
    <Text style={{ fontSize: 20, fontWeight: 'bold' }}>Edit Pet Profile</Text>
    <TextInput
      placeholder="Pet Name"
      value={profile.name}
      onChangeText={(text) => handleChange('name', text)}
      style={{ marginVertical: 10, borderBottomWidth: 1 }}
    />
    <TextInput
      placeholder="Breed"
      value={profile.breed}
      onChangeText={(text) => handleChange('breed', text)}
      style={{ marginVertical: 10, borderBottomWidth: 1 }}
    />
    <TextInput
```

```
        placeholder="Age"
        value={profile.age}
        onChangeText={(text) => handleChange('age', text)}
        style={{ marginVertical: 10, borderBottomWidth: 1 }}
      />
      <TextInput
        placeholder="Bio"
        value={profile.bio}
        onChangeText={(text) => handleChange('bio', text)}
        style={{ marginVertical: 10, borderBottomWidth: 1 }}
      />
      <TouchableOpacity>
        <Text>Upload Profile Picture</Text>
      </TouchableOpacity>
      <Button title="Save" onPress={() => {}} />
      <Button title="Cancel" color="red" onPress={() => {}} />
    </View>
  );
};

export default UserProfile;
```

3. Community Page (Web and Mobile)

Objective: Allow users to explore and join pet communities, participate in discussions, and share content.

Design Elements:

- Community List: Displays all available communities with options to join or view more details.
- Discussion Board: Allows users to post messages, images, and comments.
- Search Bar: For finding specific communities.

Sample Code (React.js for Web):

javascript

```
import React from 'react';


const CommunityPage = () => {
  return (
    <div className="community-page">
      <h2>Explore Pet Communities</h2>
      <div className="community-list">
        <div className="community-card">
          <h3>Dog Lovers</h3>
          <p>A community for dog enthusiasts.</p>
          <button className="join-button">Join</button>
        </div>
        <div className="community-card">
          <h3>Cat Admirers</h3>
          <p>Share your love for cats.</p>
```

To create an attractive and innovative front-end for the website and mobile app for the MVP phase, we can use Amazon's AWS platform for cloud services, and leverage technologies like React.js for web development and React Native for mobile development. The goal is to design an intuitive, pet-centric user interface that provides a seamless experience for users, focusing on pet profiles, community engagement, and marketplace features.

Core Screens and Architecture Overview

1. Landing Page

This page will serve as the main entry point for users, featuring a hero section with images of pets, an introduction to the platform, and a clear call-to-action to sign up or learn more. The design will be responsive, ensuring a consistent experience across web and mobile devices.

Code Sample:

```javascript
```

```
import React from 'react';
import './LandingPage.css';

const LandingPage = () => {
  return (
    <div className="landing-page">
      <header className="hero-section">
        <div className="hero-text">
          <h1>Welcome to PetConnect</h1>
          <p>Your Social Network for Pets</p>
          <button className="cta-button">Join Now</button>
        </div>
      </header>
      <section className="features">
        <div className="feature-item">
          <img src="/assets/profile-icon.png" alt="Profile Icon"/>
          <h3>Create Pet Profiles</h3>
          <p>Build a detailed profile for your pet.</p>
        </div>
        <div className="feature-item">
          <img src="/assets/community-icon.png" alt="Community Icon"/>
          <h3>Join Communities</h3>
          <p>Connect with other pet owners.</p>
        </div>
        <div className="feature-item">
          <img src="/assets/marketplace-icon.png" alt="Marketplace Icon"/>
          <h3>Explore Marketplace</h3>
          <p>Find products and services for your pet.</p>
        </div>
```

```
      </section>
    </div>
  );
};

export default LandingPage;
```

CSS (LandingPage.css):

```
css
.landing-page {
  font-family: 'Roboto', sans-serif;
  color: #333;
  text-align: center;
}

.hero-section {
  background-image: url('/assets/hero-bg.jpg');
  background-size: cover;
  padding: 100px 20px;
  color: white;
}

.hero-text h1 {
  font-size: 3rem;
  margin-bottom: 20px;
}

.cta-button {
  background-color: #ff6b6b;
  color: white;
```

```
  padding: 10px 20px;
  border-radius: 5px;
  font-size: 1rem;
  margin-top: 20px;
  cursor: pointer;
}
```

2. User Profile (Pets and Owners)

Users will manage their profiles and their pets' profiles. Key features include profile picture upload, editable fields for pet information, and bio sections. The interface will be clean and easy to navigate, with forms that allow for quick updates.

Code Sample:

javascript

```
import React, { useState } from 'react';

const UserProfile = () => {
  const [profile, setProfile] = useState({
    petName: '',
    breed: '',
    age: '',
    bio: '',
    profilePicture: ''
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setProfile(prevProfile => ({
      ...prevProfile,
      [name]: value
    }));
```

```
  };

  return (
    <div className="user-profile">
      <h2>Edit Pet Profile</h2>
      <form>
        <label>Pet Name</label>
        <input type="text" name="petName" value={profile.petName} onChange={handleChange} />

        <label>Breed</label>
        <input type="text" name="breed" value={profile.breed} onChange={handleChange} />

        <label>Age</label>
        <input type="number" name="age" value={profile.age} onChange={handleChange} />

        <label>Bio</label>
        <textarea name="bio" value={profile.bio} onChange={handleChange}></textarea>

        <label>Profile Picture</label>
        <input type="file" name="profilePicture" onChange={handleChange} />

        <button type="submit">Save Profile</button>
      </form>
    </div>
  );
};

export default UserProfile;
```

3. Community Page

Users can explore and join pet communities, participate in discussions, and share content. This screen will feature a list of available communities, a search bar, and a discussion board. It’s designed for engagement, allowing users to interact with other pet owners.

Code Sample:

javascript

```
import React from 'react';


const CommunityPage = () => {
  return (
    <div className="community-page">
      <h2>Pet Communities</h2>
      <div className="community-list">
        <div className="community-item">
          <h3>Dog Lovers</h3>
          <p>Join the discussion with other dog owners.</p>
          <button className="join-button">Join</button>
        </div>
        <div className="community-item">
          <h3>Cat Enthusiasts</h3>
          <p>Share your love for cats.</p>
          <button className="join-button">Join</button>
        </div>
      </div>
    </div>
  );
};


export default CommunityPage;
```

Backend Architecture Overview

Using AWS technologies:

- Database: Amazon RDS (PostgreSQL) for relational data management.
- Authentication: AWS Cognito for user management and authentication.
- File Storage: AWS S3 for storing profile images and other media.
- API Management: AWS API Gateway and AWS Lambda for scalable and secure API endpoints.
- AI and ML Integration: AWS Rekognition for image recognition (e.g., pet breed identification), and AWS SageMaker for machine learning model deployment.
- AR/VR: AWS Sumerian for developing AR/VR experiences.

Database Schema

- Core Table: Pets
    - Fields: pet_id, owner_id, name, breed, age, bio, profile_picture_url, etc.
- Related Tables:
    - Users: user_id, name, email, password_hash, profile_picture_url, etc.
    - HealthRecords: record_id, pet_id, vaccination_date, vaccine_name, etc.
    - Communities: community_id, name, description, etc.
    - MarketplaceItems: item_id, seller_id, title, description, price, etc.
    - Routes: route_id, name, location, length, etc.

Sample Schema for the Pets Table (PostgreSQL)

sql

```
CREATE TABLE Pets (
  pet_id SERIAL PRIMARY KEY,
  owner_id INT REFERENCES Users(user_id),
  name VARCHAR(255) NOT NULL,
  breed VARCHAR(255),
  age INT,
  bio TEXT,
  profile_picture_url TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
);

CREATE TABLE Users (
  user_id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  profile_picture_url TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE HealthRecords (
  record_id SERIAL PRIMARY KEY,
  pet_id INT REFERENCES Pets(pet_id),
  vaccination_date DATE,
  vaccine_name VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Communities (
  community_id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE MarketplaceItems (
  item_id SERIAL PRIMARY KEY,
  seller_id INT REFERENCES Users(user_id),
```

```
  title VARCHAR(255) NOT NULL,
  description TEXT,
  price DECIMAL(10, 2),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Routes (
  route_id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  location GEOGRAPHY,
  length FLOAT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

AWS Lambda Function for User Registration (Node.js)

javascript

```
const AWS = require('aws-sdk');
const bcrypt = require('bcryptjs');
const cognito = new AWS.CognitoIdentityServiceProvider();

exports.handler = async (event) => {
  const { email, password, name } = JSON.parse(event.body);

  const hashedPassword = bcrypt.hashSync(password, 8);

  const params = {
    ClientId: process.env.COGNITO_CLIENT_ID,
    Username: email,
    Password: hashedPassword,
```

```
    UserAttributes: [
      { Name: 'name', Value: name },
      { Name: 'email', Value: email }
    ]
  };

  try {
    const result = await cognito.signUp(params).promise();
    return {
      statusCode: 200,
      body: JSON.stringify({ message: 'User registered successfully', userId: result.UserSub })
    };
  } catch (err) {
    return {
      statusCode: 500,
      body: JSON.stringify({ message: 'Error registering user', error: err.message })
    };
  }
};
```

Conclusion

This detailed technical approach leveragesHere's an approach to create an attractive and innovative front-end for your pet-focused social network MVP, with sample code for key screens:

Design Principles

1. User Experience (UX) Focus: Ensure the UI is intuitive, friendly, and engaging for pet owners.
2. Responsive Design: Use flexible layouts to ensure the site looks great on all devices.

3. Pet-Centric Theme: Use playful colors, pet-related icons, and images to create a warm atmosphere.

Core Screens

1. Home/Landing Page

The landing page will welcome users with an engaging hero section, a brief introduction to the platform, and clear calls-to-action.

Code Sample (React.js)

javascript

```
import React from 'react';
import './LandingPage.css';

const LandingPage = () => {
  return (
    <div className="landing-page">
      <header className="hero">
        <h1>Welcome to PetConnect</h1>
        <p>The social network for pet lovers</p>
        <button className="cta-button">Join Now</button>
      </header>
      <section className="features">
        <div className="feature">
          <img src="/images/profile.png" alt="Profile" />
          <h3>Create Pet Profiles</h3>
          <p>Share your pet’s unique personality.</p>
        </div>
        <div className="feature">
          <img src="/images/community.png" alt="Community" />
          <h3>Join Communities</h3>
```

```
        <p>Connect with fellow pet lovers.</p>
      </div>
      <div className="feature">
        <img src="/images/marketplace.png" alt="Marketplace" />
        <h3>Explore the Marketplace</h3>
        <p>Find the best products for your pet.</p>
      </div>
    </section>
  </div>
 );
};

export default LandingPage;
```

CSS (LandingPage.css)

```
.landing-page {
 font-family: Arial, sans-serif;
 color: #444;
 text-align: center;
}

.hero {
 background-color: #f9f9f9;
 padding: 100px 20px;
 background-image: url('/images/hero-bg.jpg');
 background-size: cover;
}
```

```
.cta-button {
  background-color: #ff7f50;
  color: white;
  padding: 15px 30px;
  border-radius: 30px;
  font-size: 1.2rem;
  cursor: pointer;
  margin-top: 20px;
}

.features {
  display: flex;
  justify-content: space-around;
  padding: 50px 20px;
}

.feature img {
  width: 60px;
  height: 60px;
}

.feature {
  max-width: 300px;
}
```

2. User Profile (Web)

This screen allows users to manage their profiles and their pets' profiles. It includes editable fields for pet details, profile picture upload, and a bio section.

Code Sample (React.js)

```javascript
import React, { useState } from 'react';

const UserProfile = () => {
  const [profile, setProfile] = useState({
    petName: '',
    breed: '',
    age: '',
    bio: '',
    profilePicture: ''
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setProfile(prevProfile => ({
      ...prevProfile,
      [name]: value
    }));
  };

  return (
    <div className="user-profile">
      <h2>Edit Pet Profile</h2>
      <form>
        <label>Pet Name</label>
        <input type="text" name="petName" value={profile.petName} onChange={handleChange} />

        <label>Breed</label>
        <input type="text" name="breed" value={profile.breed} onChange={handleChange} />
```

```
      <label>Age</label>
      <input type="number" name="age" value={profile.age} onChange={handleChange} />

      <label>Bio</label>
      <textarea name="bio" value={profile.bio} onChange={handleChange}></textarea>

      <label>Profile Picture</label>
      <input type="file" name="profilePicture" onChange={handleChange} />

      <button type="submit">Save Profile</button>
    </form>
  </div>
 );
};
export default UserProfile;
```

3. Community Page

This screen will allow users to explore and join various pet communities. Users can participate in discussions, share photos, and connect with other pet owners.

Code Sample (React.js)

javascript

```
import React from 'react';

const CommunityPage = () => {
 return (
  <div className="community-page">
   <h2>Explore Pet Communities</h2>
```

```
    <div className="community-list">
     <div className="community-card">
      <h3>Dog Lovers</h3>
      <p>Join the discussion with fellow dog owners.</p>
      <button className="join-button">Join</button>
     </div>
     <div className="community-card">
      <h3>Cat Enthusiasts</h3>
      <p>Share your love for cats.</p>
      <button className="join-button">Join</button>
     </div>
    </div>
   </div>
  );
};

export default CommunityPage;
```

Architecture Overview

The application will be built using AWS services to ensure scalability and reliability. Key services include:

- Database: Amazon RDS for PostgreSQL
- Authentication: AWS Cognito
- Storage: AWS S3 for images and media
- Backend: AWS Lambda with Node.js for serverless functions
- AI Integration: AWS Rekognition for image analysis (e.g., breed recognition)

Conclusion

This detailed architecture and code structure for the MVP phase lays a strong foundation for a scalable, feature-rich pet social network. By integrating AI and ML technologies from the outset and adopting a

microservices architecture on a cloud platform, the platform is positioned to evolve rapidly while ensuring a seamless and engaging user experience. This approach not only accelerates development but also ensures that the platform is capable of handling future growth and feature expansions effectively.