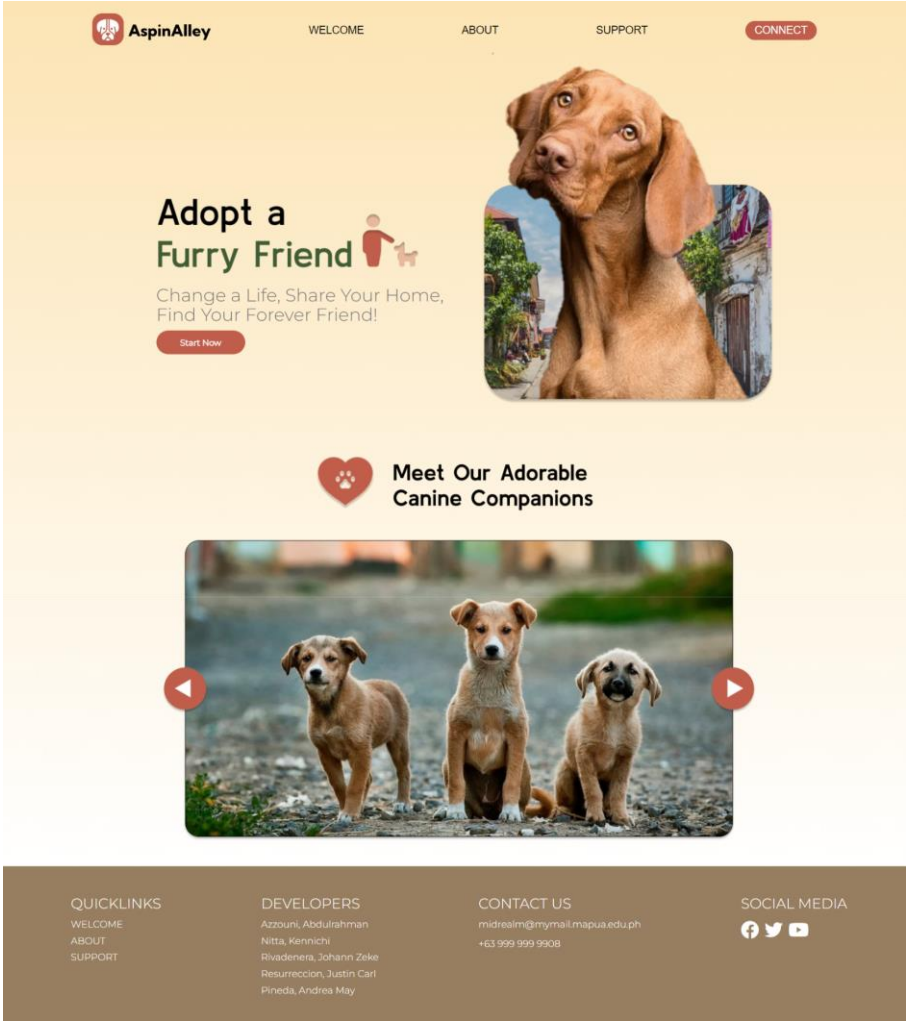
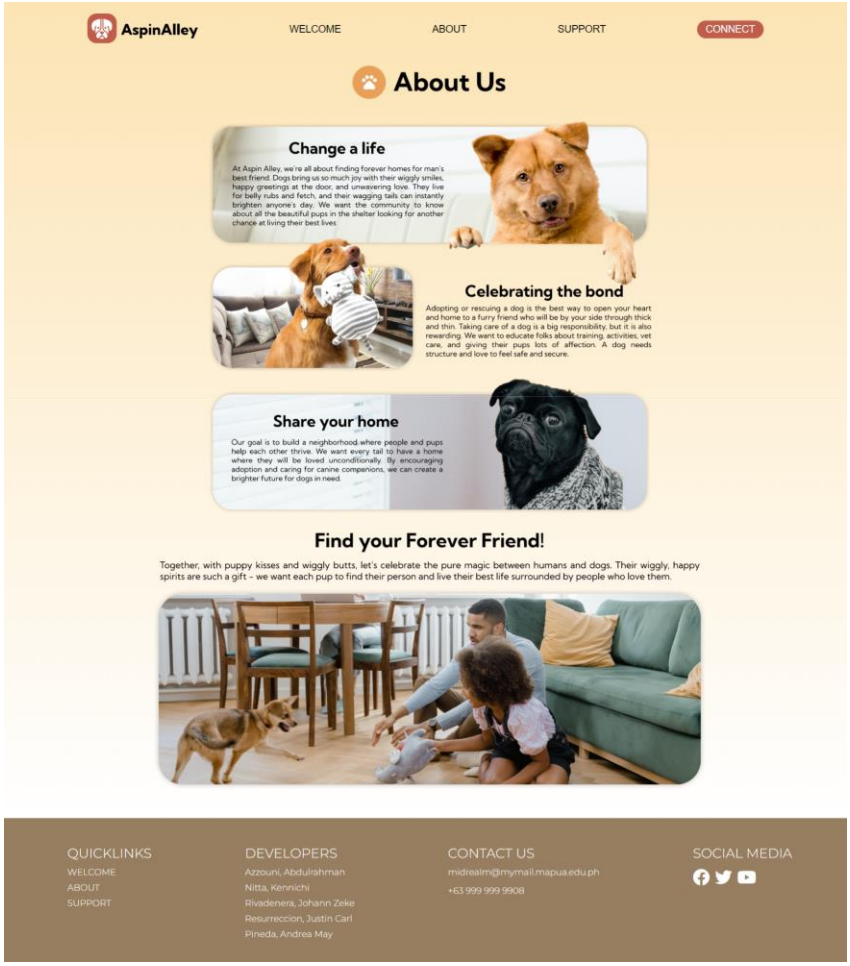
[IT192] APPLICATION DEVELOPMENT 2

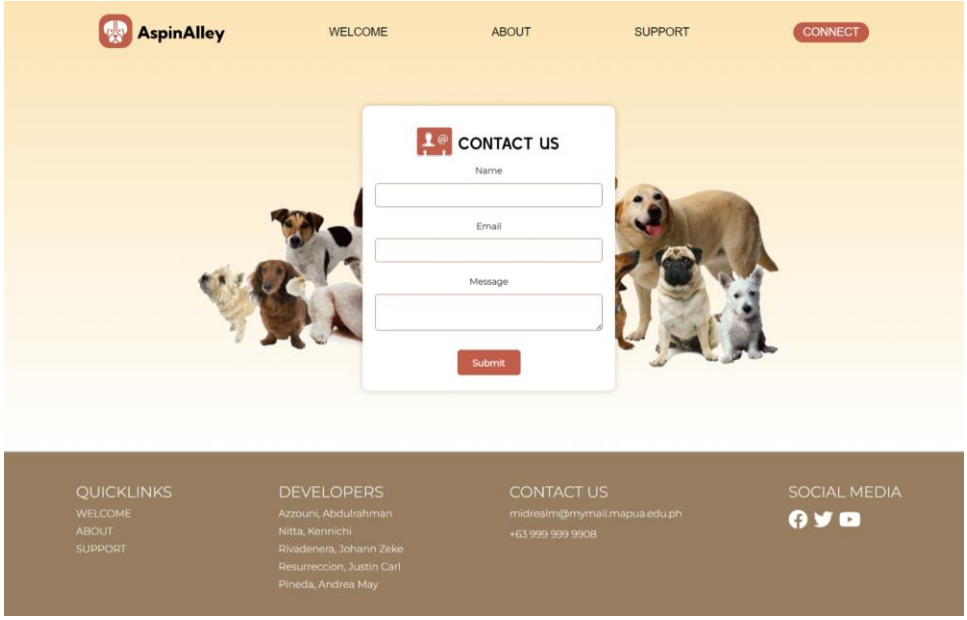
TERM-END PROJECT

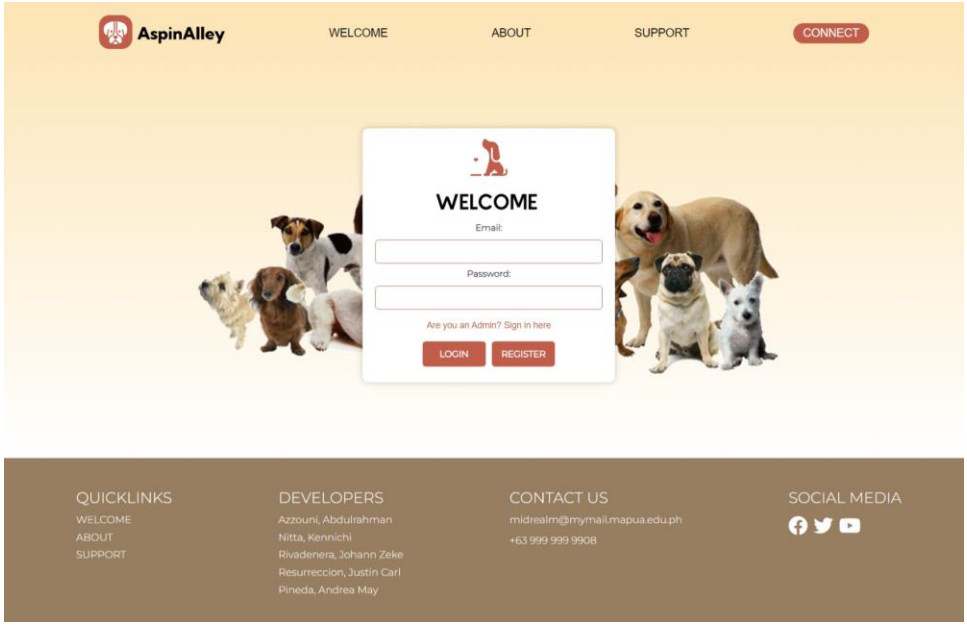
Developed by Andrea May M. Pineda, Abdulrahman Al Azzouni, Kennichi Nitta, Johann Zeke Rivadenera and Justin Carl Resurreccion (FOPI01 – 1Q2324)

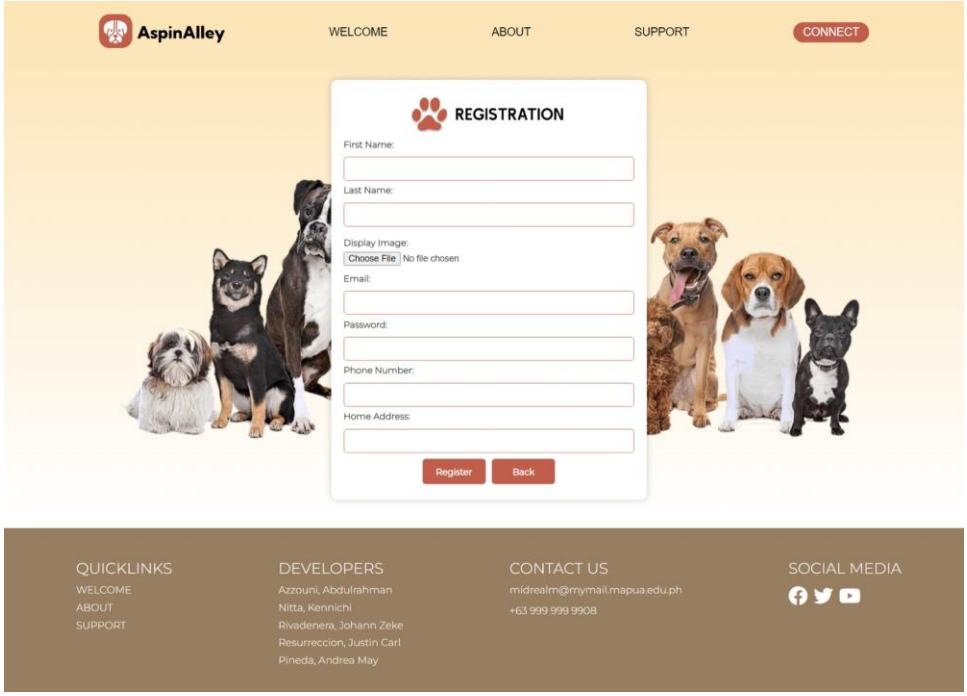
[ammpineda/dog-adoption-management-system (github.com)]

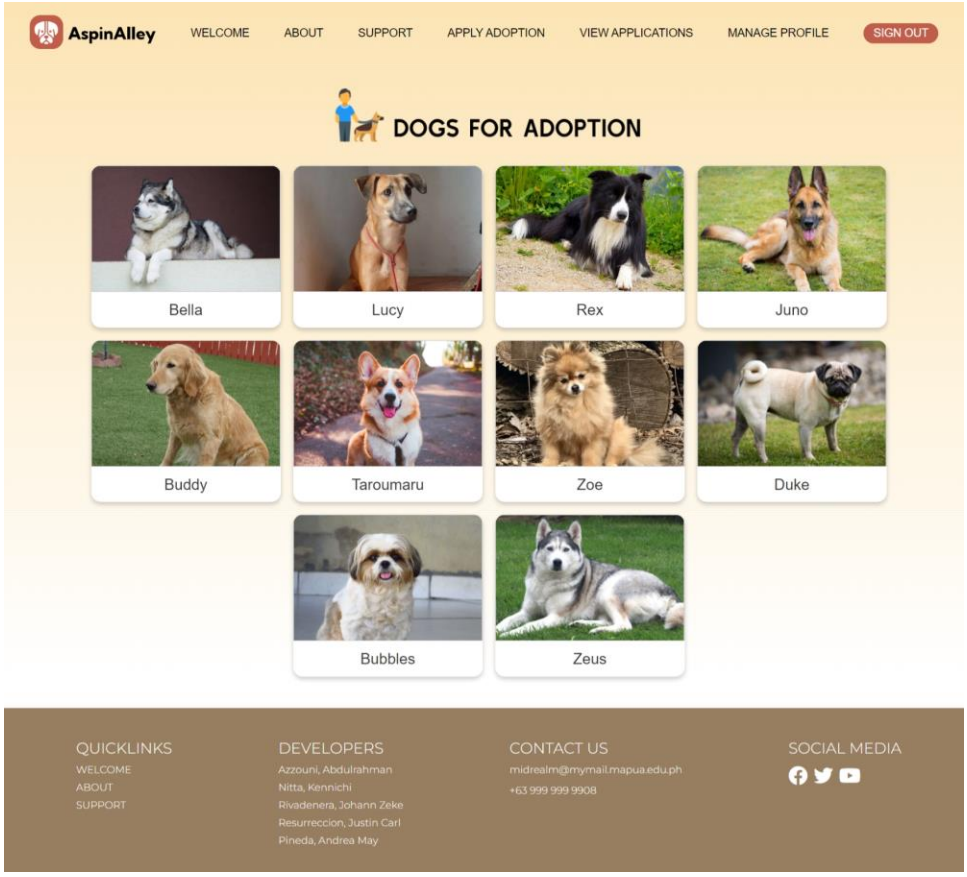


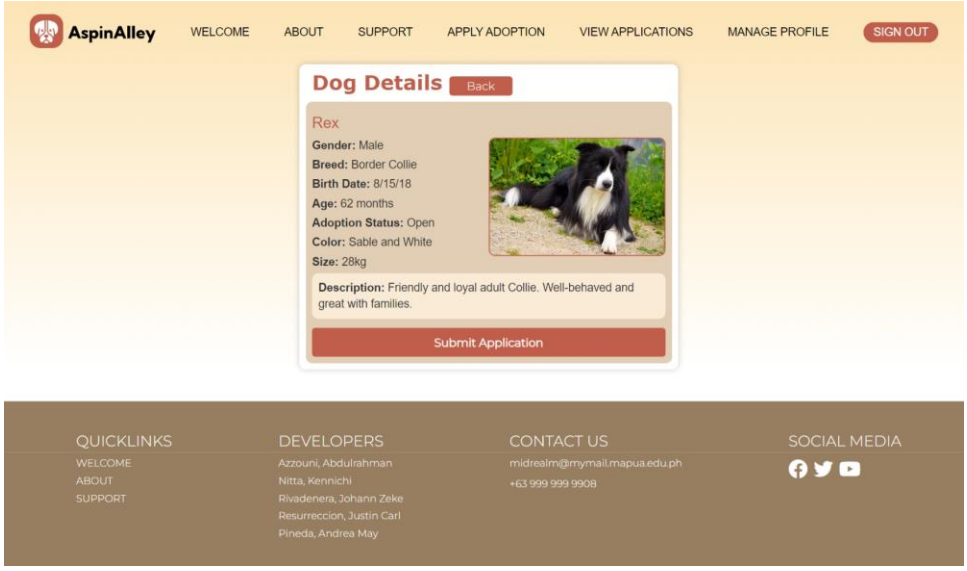

AspinAlley
WELCOME
ABOUT
SUPPORT
CONNECT
About Us
Change a life
At Aspin Alley, we're all about finding forever homes for man's best friend. Dogs bring us so much joy with their wiggly smiles, happy greetings at the door, and unwavering love. They live for belly rubs and fetch, and their wagging tails can instantly brighten anyone's day. We want the community to know about all the beautiful pups in the shelter looking for another chance at living their best lives.
Celebrating the bond
Adopting or rescuing a dog is the best way to open your heart and home to a furry friend who will be by your side through thick and thin. Taking care of a dog is a big responsibility, but it is also rewarding. We want to educate folks about training, activities, vet care, and giving their pups lots of affection. A dog needs structure and love to feel safe and secure.
Share your home
Our goal is to build a neighborhood where people and pups help each other thrive. We want every tail to have a home where they will be loved unconditionally. By encouraging adoption and caring for canine companions, we can create a brighter future for dogs in need.
Find your Forever Friend!
Together, with puppy kisses and wiggly butts, let's celebrate the pure magic between humans and dogs. Their wiggly, happy spirits are such a gift - we want each pup to find their person and live their best life surrounded by people who love them.
QUICKLINKS
WELCOME
ABOUT
SUPPORT
DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May
CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908
SOCIAL MEDIA


AspinAlley
WELCOME
ABOUT
SUPPORT
CONNECT
CONTACT US
Name
Email
Message
Submit
QUICKLINKS
WELCOME
ABOUT
SUPPORT
DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May
CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908
SOCIAL MEDIA


AspinAlley
WELCOME
ABOUT
SUPPORT
CONNECT
WELCOME
Email:
Password:
Are you an Admin? Sign in here
LOGIN
REGISTER
QUICKLINKS
WELCOME
ABOUT
SUPPORT
DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May
CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908
SOCIAL MEDIA


AspinAlley
WELCOME
ABOUT
SUPPORT
CONNECT
REGISTRATION
First Name:
Last Name:
Display Image:
Choose File No file chosen
Email:
Password:
Phone Number:
Home Address:
Register
Back
QUICKLINKS
WELCOME
ABOUT
SUPPORT
DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May
CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908
SOCIAL MEDIA


AspinAlley
WELCOME
ABOUT
SUPPORT
APPLY ADOPTION
VIEW APPLICATIONS
MANAGE PROFILE
SIGN OUT
DOGS FOR ADOPTION
Bella
Lucy
Rex
Juno
Buddy
Taroumaru
Zoe
Duke
Bubbles
Zeus
QUICKLINKS
WELCOME
ABOUT
SUPPORT
DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May
CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908
SOCIAL MEDIA


AspinAlley
WELCOME
ABOUT
SUPPORT
APPLY ADOPTION
VIEW APPLICATIONS
MANAGE PROFILE
SIGN OUT
Dog Details
Back
Rex
Gender: Male
Breed: Border Collie
Birth Date: 8/15/18
Age: 62 months
Adoption Status: Open
Color: Sable and White
Size: 28kg
Description: Friendly and loyal adult Collie. Well-behaved and great with families.
Submit Application
QUICKLINKS
WELCOME
ABOUT
SUPPORT
DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May
CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908
SOCIAL MEDIA

AspinAlley WELCOME ABOUT SUPPORT APPLY ADOPTION VIEW APPLICATIONS MANAGE PROFILE SIGN OUT

MY APPLICATIONS

ID	Status	Submitted Date	Review Date	Approval Date	Dog Name	Dog Breed	Age	Gender
3	Submitted	12/23/23, 12:22 AM			Rex	Border Collie	62 months	Male

QUICKLINKS
WELCOME
ABOUT
SUPPORT

DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May

CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908

SOCIAL MEDIA

AspinAlley WELCOME ABOUT SUPPORT APPLY ADOPTION VIEW APPLICATIONS MANAGE PROFILE SIGN OUT

MANAGE MY PROFILE

First Name:
Childe

Last Name:
Landau

Display Image:
Choose File No file chosen

Email Address:
childelandau@gmail.com

Password:
servalcomplex0426

Phone Number:
09219247810

Home Address:
Tayabas City, Quezon Province

Edit Back

QUICKLINKS
WELCOME
ABOUT
SUPPORT

DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May

CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908

SOCIAL MEDIA

AspinAlley WELCOME ABOUT SUPPORT MANAGE DOGS MANAGE USERS MANAGE APPLICATIONS SIGN OUT

MANAGE DOGS

Search by ◉ Name ○ Gender ○ Status Search Add Dog

Name	Gender	Breed	Age	Adoption Status
Bella	Female	Alaskan Malamute	53 months	Adopted
Lucy	Female	Aspin	66 months	Adopted
Rex	Male	Border Collie	62 months	Reserved
Juno	Male	German Shepherd	78 months	Open
Buddy	Male	Golden Retriever	68 months	Open
Taroumaru	Male	Pembroke Welsh Corgi	23 months	Open
Zoe	Female	Pomeranian	13 months	Open
Duke	Male	Pug	14 months	Open
Bubbles	Male	Shih Tzu	15 months	Open
Zeus	Male	Siberian Husky	41 months	Open

QUICKLINKS
WELCOME
ABOUT
SUPPORT

DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May

CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908

SOCIAL MEDIA

AspinAlley WELCOME ABOUT SUPPORT MANAGE DOGS MANAGE USERS MANAGE APPLICATIONS SIGN OUT

MANAGE ADOPTERS

Search by ◉ Name ○ Email ○ Phone Number Search Add Adopter

Name	Email	Phone Number
Childe Landau	childelandau@gmail.com	09219247810
Aether Trailblazer	aetherblazer@gmail.com	09190980201
Neuvillette Yang	hydrosovereign@gmail.com	09217805168
Ghislaine Silvamillion	chiefdedoldia@gmail.com	09805248721
Mei Kaslana	kaslanamei@gmail.com	09187245680

QUICKLINKS
WELCOME
ABOUT
SUPPORT

DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May

CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908

SOCIAL MEDIA

AspinAlley WELCOME ABOUT SUPPORT MANAGE DOGS MANAGE USERS MANAGE APPLICATIONS SIGN OUT

MANAGE APPLICATIONS

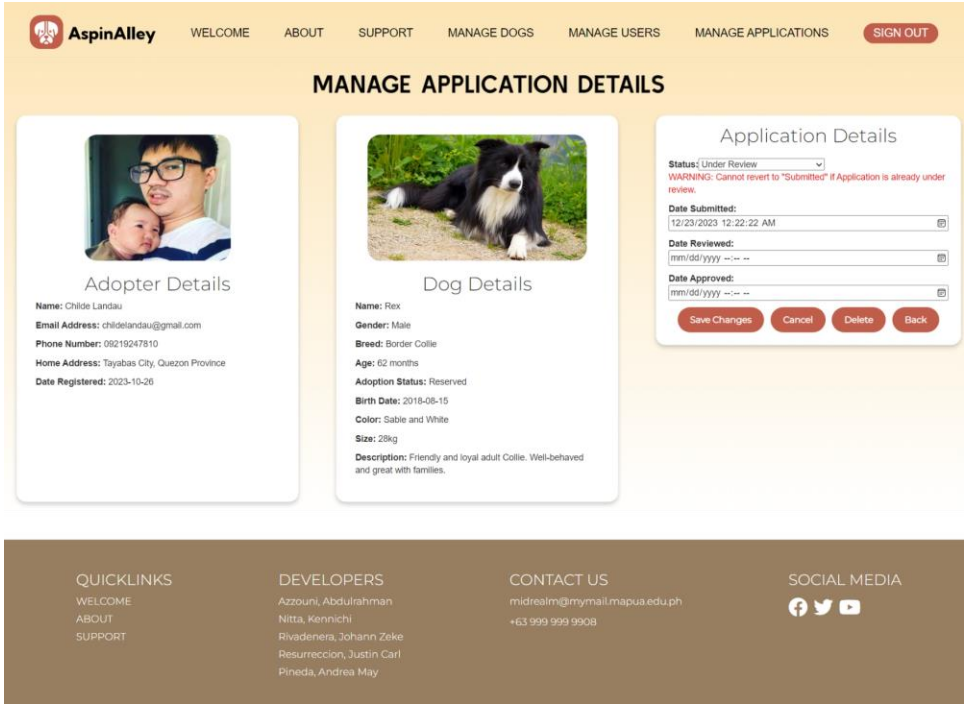
ID	Status	Submitted Date	Review Date	Approval Date	Applicant Name	Dog Name
1	Approved	10/26/23, 6:53 PM	10/26/23, 7:50 PM	10/26/23, 7:51 PM	Mei Kaslana	Bella
2	Approved	10/26/23, 6:53 PM	10/26/23, 7:15 PM	10/26/23, 7:36 PM	Mei Kaslana	Lucy
3	Submitted	12/23/23, 12:22 AM			Childe Landau	Rex

QUICKLINKS
WELCOME
ABOUT
SUPPORT

DEVELOPERS
Azzouni, Abdulrahman
Nitta, Kennichi
Rivadenera, Johann Zeke
Resurreccion, Justin Carl
Pineda, Andrea May

CONTACT US
midrealm@mymail.mapua.edu.ph
+63 999 999 9908

SOCIAL MEDIA



MODELS

```
package com.project.backend.model;
[redacted]
import javax.persistence.*;
import java.util.Date;
[redacted]
@Entity
@Table(name = "admin")
public class Admin {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String email;
    private String password;
[redacted]
    public Admin(int id, String name, String email, String password) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
    }
[redacted]
    public Admin() {
```

```
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

```
package com.project.backend.model;
```

```java
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonManagedReference;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.util.Date;
import java.util.List;

@Entity
@Table(name="adopter")
public class Adopter {
    @Id
    @GeneratedValue(generator = "custom-id", strategy = GenerationType.IDENTITY)
    @GenericGenerator(name = "custom-id", strategy =
"com.project.backend.service.AdopterIdGenerator")
    private int id;
    private String firstName;
    private String lastName;
    private String fullName;
    private String displayImage; // image address path
    private String email;
    private String password;
    private String phoneNumber;
    private String homeAddress;
    @Temporal(TemporalType.DATE)
    private Date registeredDate;

    @OneToMany(mappedBy = "applicant")
    @JsonIgnore
    private List<Application> applications;

    public Adopter(int id, String firstName, String lastName, String fullName,
String displayImage, String email, String password, String phoneNumber, String
homeAddress, Date registeredDate, List<Application> applications) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.fullName = fullName;
        this.displayImage = displayImage;
        this.email = email;
        this.password = password;
        this.phoneNumber = phoneNumber;
        this.homeAddress = homeAddress;
        this.registeredDate = registeredDate;
```

```java
        this.applications = applications;
    }

    public Adopter() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName() {
        this.fullName = this.getFirstName() + " " + this.getLastName();
    }

    public String getDisplayImage() {
        return displayImage;
    }

    public void setDisplayImage(String displayImage) {
        this.displayImage = displayImage;
    }
```

```java
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getHomeAddress() {
        return homeAddress;
    }

    public void setHomeAddress(String homeAddress) {
        this.homeAddress = homeAddress;
    }

    public Date getRegisteredDate() {
        return registeredDate;
    }

    public void setRegisteredDate(Date registeredDate) {
        this.registeredDate = registeredDate;
    }

    public List<Application> getApplications() {
        return applications;
    }
```

```
    public void setApplications(List<Application> applications) {
        this.applications = applications;
    }
}
```

```java
package com.project.backend.model;
[redacted]
import com.fasterxml.jackson.annotation.JsonIgnore;
import org.hibernate.annotations.GenericGenerator;
[redacted]
import javax.persistence.*;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.Date;
@Entity
@Table(name="application")
public class Application {
    @Id
    @GeneratedValue(generator = "custom-id", strategy = GenerationType.IDENTITY)
    @GenericGenerator(name = "custom-id", strategy =
"com.project.backend.service.ApplicationIdGenerator")
    private int id;
    private String status;
    private LocalDateTime submittedDate;
    private LocalDateTime reviewDate;
    private LocalDateTime approvalDate;
[redacted]
    @ManyToOne
    @JoinColumn(name = "applicant_id")
    private Adopter applicant;
[redacted]
    @OneToOne
    @JoinColumn(name = "dog_id")
    private Dog dog;
[redacted]
    public Application() {
    }
[redacted]
    public Application(int id, String status, LocalDateTime submittedDate,
LocalDateTime reviewDate, LocalDateTime approvalDate, Adopter applicant, Dog dog)
{
        this.id = id;
        this.status = status;
        this.submittedDate = submittedDate;
        this.reviewDate = reviewDate;
        this.approvalDate = approvalDate;
        this.applicant = applicant;
        this.dog = dog;
    }
[redacted]
```

```java
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public LocalDateTime getSubmittedDate() {
        return submittedDate;
    }

    public void setSubmittedDate(LocalDateTime submittedDate) {
        this.submittedDate = submittedDate;
    }

    public LocalDateTime getReviewDate() {
        return reviewDate;
    }

    public void setReviewDate(LocalDateTime reviewDate) {
        this.reviewDate = reviewDate;
    }

    public LocalDateTime getApprovalDate() {
        return approvalDate;
    }

    public void setApprovalDate(LocalDateTime approvalDate) {
        this.approvalDate = approvalDate;
    }

    public Adopter getApplicant() {
        return applicant;
    }

    public void setApplicant(Adopter applicant) {
```

```
        this.applicant = applicant;
    }
```

```
    public Dog getDog() {
        return dog;
    }
```

```
    public void setDog(Dog dog) {
        this.dog = dog;
    }
}
```

```java
package com.project.backend.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonManagedReference;
import org.hibernate.annotations.GenericGenerator;
import org.hibernate.annotations.Type;

import javax.persistence.*;
import java.util.Calendar;
import java.util.Date;

@Entity
@Table(name="dog")
public class Dog {
    @Id
    @GeneratedValue(generator = "custom-id", strategy = GenerationType.IDENTITY)
    @GenericGenerator(name = "custom-id", strategy =
"com.project.backend.service.DogIdGenerator")
    private int id;
    private String name;
    private String displayImage;
    private String breed;
    @Temporal(TemporalType.DATE)
    private Date birthDate;
    private String age;
    private String gender;
    private String color;
    private String size;
    private String adoptionStatus; // Default value
    @Type(type = "text")
    private String description;
    @Temporal(TemporalType.DATE)
    private Date registeredDate;

    @OneToOne(mappedBy = "dog")
    @JsonIgnore
    private Application application;

    public Dog() {
    }

    public Dog(int id, String name, String displayImage, String breed, Date
birthDate, String age, String gender, String color, String size, String
```

```
adoptionStatus, String description, Date registeredDate, Application application)
{
        this.id = id;
        this.name = name;
        this.displayImage = displayImage;
        this.breed = breed;
        this.birthDate = birthDate;
        this.age = age;
        this.gender = gender;
        this.color = color;
        this.size = size;
        this.adoptionStatus = adoptionStatus;
        this.description = description;
        this.registeredDate = registeredDate;
        this.application = application;
    }
[redacted]
    public int getId() {
        return id;
    }
[redacted]
    public void setId(int id) {
        this.id = id;
    }
[redacted]
    public String getName() {
        return name;
    }
[redacted]
    public void setName(String name) {
        this.name = name;
    }
[redacted]
    public String getDisplayImage() {
        return displayImage;
    }
[redacted]
    public void setDisplayImage(String displayImage) {
        this.displayImage = displayImage;
    }
[redacted]
    public String getBreed() {
        return breed;
    }
[redacted]
```

```java
    public void setBreed(String breed) {
        this.breed = breed;
    }

    public Date getBirthDate() {
        return birthDate;
    }

    public void setBirthDate(Date birthDate) {
        this.birthDate = birthDate;
    }

    public String getAge() {
        return age;
    }

    public void setAge(Date birthDate) {
        if (birthDate != null) {
            this.birthDate = birthDate;

            int computedAge = calculateAge(this.birthDate);
            this.age = String.valueOf(computedAge);
        }
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public String getSize() {
        return size;
    }
```

```java
    public void setSize(String size) {
        this.size = size;
    }

    public String getAdoptionStatus() {
        return adoptionStatus;
    }

    public void setAdoptionStatus(String adoptionStatus) {
        this.adoptionStatus = adoptionStatus;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Application getApplication() {
        return application;
    }

    public void setApplication(Application application) {
        this.application = application;
    }

    public Date getRegisteredDate() {
        return registeredDate;
    }

    public void setRegisteredDate(Date registeredDate) {
        this.registeredDate = registeredDate;
    }

    private int calculateAge(Date birthDate) {
        Calendar birthCalendar = Calendar.getInstance();
        birthCalendar.setTime(birthDate);

        Calendar currentCalendar = Calendar.getInstance();

        int years = currentCalendar.get(Calendar.YEAR) -
birthCalendar.get(Calendar.YEAR);
```

```
        int months = currentCalendar.get(Calendar.MONTH) -
birthCalendar.get(Calendar.MONTH);
[redacted]
        if (currentCalendar.get(Calendar.DAY_OF_MONTH) <
birthCalendar.get(Calendar.DAY_OF_MONTH)) {
            months--;
        }
[redacted]
        if (months < 0) {
            months += 12;
            years--;
        }
[redacted]
        return years * 12 + months;
    }
}
[redacted]
```

```java
package com.project.backend.model;

public class LoginRequest {
    private String email;
    private String password;

    public LoginRequest(String email, String password) {
        this.email = email;
        this.password = password;
    }

    public LoginRequest() {
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

}
```

```java
package com.project.backend.model;

public class LoginResponse {
    private int userId;
    private String message;

    public LoginResponse(int userId, String message) {
        this.userId = userId;
        this.message = message;
    }

    public LoginResponse(){

    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

REPOSITORIES

```
package com.project.backend.repository;
[redacted]
import com.project.backend.model.Admin;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
[redacted]
@Repository
public interface AdminRepository extends CrudRepository<Admin, Integer> {
[redacted]
    @Query("SELECT a FROM Admin a WHERE a.email = :email")
    Admin findAdminByEmail(@Param("email") String email);
[redacted]
    @Query("SELECT a FROM Admin a WHERE a.password = :password")
    Admin findAdminByPassword(@Param("password") String pass);
[redacted]
}
[redacted]
```

```
package com.project.backend.repository;
[redacted]
import com.project.backend.model.Admin;
import com.project.backend.model.Adopter;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
[redacted]
import java.util.List;
import java.util.Optional;
[redacted]
@Repository
public interface AdopterRepository extends CrudRepository<Adopter, Integer> {
```

```
    @Query("SELECT a FROM Adopter a WHERE CONCAT(a.firstName, ' ', a.lastName)
LIKE %:name%")
    List<Adopter> findByName(@Param("name") String name);
[redacted]
    @Query("SELECT a FROM Adopter a WHERE CONCAT(a.email) LIKE %:email%")
    List<Adopter> findByEmail(@Param("email")String email);
[redacted]
    @Query("SELECT a FROM Adopter a WHERE CONCAT(a.phoneNumber) LIKE %:phone%")
    List<Adopter> findByPhoneNumber(@Param("phone")String phoneNumber);
[redacted]
    @Query("SELECT a FROM Adopter a WHERE a.email = :email")
    Adopter findUserByEmail(@Param("email") String email);
[redacted]
    @Query("SELECT a FROM Adopter a WHERE a.password = :password")
    Adopter findUserByPassword(@Param("password") String pass);
}
[redacted]
```

```
package com.project.backend.repository;
[redacted]
import com.project.backend.model.Application;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
[redacted]
import java.util.List;
[redacted]
@Repository
public interface ApplicationRepository extends CrudRepository<Application,
Integer> {
    @Query("SELECT a FROM Application a WHERE CONCAT(a.applicant.fullName) LIKE
%:applicant%")
    List<Application> findByApplicantName(@Param("applicant") String
applicantName);
    @Query("SELECT a FROM Application a WHERE CONCAT(a.dog.name) LIKE %:dog%")
    List<Application> findByDogName(@Param("dog") String dogName);
    @Query("SELECT a FROM Application a WHERE a.applicant.id LIKE %:id%")
    List<Application> findByApplicantId(@Param("id") int id);
}
[redacted]
```

```java
package com.project.backend.repository;
[redacted]
import com.project.backend.model.Dog;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
[redacted]
import java.util.List;
[redacted]
@Repository
public interface DogRepository extends CrudRepository<Dog, Integer> {
    @Query("SELECT a FROM Dog a WHERE CONCAT(a.name) LIKE %:name%")
    List<Dog> findByName(@Param("name") String name);
[redacted]
    List<Dog> findByGender(@Param("gender") String gender);
[redacted]
    @Query("SELECT a FROM Dog a WHERE CONCAT(a.adoptionStatus) LIKE
%:adoptionStatus%")
    List<Dog> findByAdoptionStatus(@Param("adoptionStatus")String
adoptionStatus);
}
[redacted]
```

SERVICES

```java
package com.project.backend.service;
[redacted]
import com.project.backend.model.Admin;
import com.project.backend.repository.AdminRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
[redacted]
import java.util.List;
[redacted]
@Service
public class AdminService {
[redacted]
```

```java
    @Autowired
    AdminRepository adminRepository;

    public Admin registerAdmin(Admin admin){
        return adminRepository.save(admin);
    }

    public void deleteAdmin(int adminId){
        adminRepository.deleteById(adminId);
    }

    public Admin getAdminById(int adminId){
        return adminRepository.findById(adminId)
                .orElse(null);
    }

    public List<Admin> getAllAdmins(){
        return (List<Admin>) adminRepository.findAll();
    }

    public Admin getAdminByEmail(String email) {
        return adminRepository.findAdminByEmail(email);
    }

    public Admin getAdminByPassword(String password){
        return adminRepository.findAdminByPassword(password);
    }

}
```

```java
package com.project.backend.service;
[redacted]
import com.project.backend.model.Admin;
import com.project.backend.model.Adopter;
import com.project.backend.model.Application;
import com.project.backend.repository.AdopterRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
[redacted]
import java.util.Date;
import java.util.List;
import java.util.Optional;
[redacted]
@Service
public class AdopterService {
    @Autowired
    AdopterRepository adopterRepository;
[redacted]
    @Autowired
    ApplicationService applicationService;
[redacted]
    public Adopter registerAdopter(Adopter adopter){
        adopter.setFullName();
        adopter.setRegisteredDate(new Date());
        return adopterRepository.save(adopter);
    }
    public Adopter updateAdopter(int adopterId, Adopter update){
        Optional<Adopter> existingOptional =
adopterRepository.findById(adopterId);
[redacted]
        if(existingOptional.isPresent()){
            Adopter existing = existingOptional.get();
[redacted]
            if(update.getFirstName()!=null){
                existing.setFirstName(update.getFirstName());
            }
            if(update.getLastName()!=null){
```

```
                existing.setLastName(update.getLastName());
            }
            if(update.getDisplayImage()!=null){
                existing.setDisplayImage(update.getDisplayImage());
            }
            if(update.getEmail()!=null){
                existing.setEmail(update.getEmail());
            }
            if(update.getPassword()!=null){
                existing.setPassword(update.getPassword());
            }
            if(update.getPhoneNumber()!=null){
                existing.setPhoneNumber(update.getPhoneNumber());
            }
            if(update.getHomeAddress()!=null){
                existing.setHomeAddress(update.getHomeAddress());
            }
            existing.setFullName();
            return adopterRepository.save(existing);
        } else{
            return null;
        }
    }
    public void deleteAdopter(int adopterId) {
        Adopter adopter = adopterRepository.findById(adopterId).orElse(null);
[redacted]
        List<Application> applications = adopter.getApplications();
[redacted]
        // Delete each associated application
        if(applications != null && !applications.isEmpty()){
            for (Application application : applications) {
                applicationService.deleteApplication(application.getId());
            }
        }
[redacted]
        adopterRepository.deleteById(adopterId);
[redacted]
    }
    public List<Adopter> getAllAdopters(){ return (List<Adopter>)
adopterRepository.findAll(); }
    public Adopter getAdopterById(int adopterId) {
        return adopterRepository.findById(adopterId).orElse(null);
    }
    public List<Adopter> getAdoptersByName(String adopterName){
        return adopterRepository.findByName(adopterName);
```

```
    }
    public List<Adopter> getAdoptersByEmail(String email) { return
adopterRepository.findByEmail(email); }
    public List<Adopter> getAdoptersByPhoneNumber(String phoneNumber) { return
adopterRepository.findByPhoneNumber(phoneNumber); }
[redacted]
    public Adopter getUserByEmail(String email) {
        return adopterRepository.findUserByEmail(email);
    }
[redacted]
    public Adopter getUserByPassword(String password){
        return adopterRepository.findUserByPassword(password);
    }
}
[redacted]
```

```java
package com.project.backend.service;

import com.project.backend.model.Adopter;
import com.project.backend.model.Application;
import com.project.backend.model.Dog;
import com.project.backend.repository.AdopterRepository;
import com.project.backend.repository.ApplicationRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.LocalDateTime;
import java.util.Date;
import java.util.List;
import java.util.Optional;

@Service
public class ApplicationService {
    @Autowired
    private ApplicationRepository applicationRepository;
    @Autowired
    private AdopterService adopterService;
    @Autowired
    private DogService dogService;
    public Application createApplication(Application application) {
        Adopter applicant =
adopterService.getAdopterById(application.getApplicant().getId());
        Dog dog = dogService.getDogById(application.getDog().getId());

        application.setStatus("Submitted");
        application.setApplicant(applicant);

        dog.setAdoptionStatus("Reserved");
        application.setDog(dog);

        LocalDateTime currentDateTime = LocalDateTime.now();
```

```
        application.setSubmittedDate(currentDateTime);
        return applicationRepository.save(application);
    }
```

```
    public Application updateApplication(int applicationId, Application update){
        Optional<Application> existingOptional =
applicationRepository.findById(applicationId);
        Application application =
applicationRepository.findById(applicationId).orElse(null);
        Dog dog = dogService.getDogById(application.getDog().getId());
```

```
        if(existingOptional.isPresent()){
            Application existing = existingOptional.get();
```

```
            if (update.getStatus()!=null){
                if(existing.getStatus().equalsIgnoreCase("Under Review" )&&
update.getStatus().trim().equalsIgnoreCase("Submitted")){
                    existing.setStatus("Under Review");
                } else if (existing.getStatus().equalsIgnoreCase("Approved" )){
                    existing.setStatus("Approved");
                } else {
                    existing.setStatus(update.getStatus());
                }
            }
            if (update.getReviewDate()!=null){
                existing.setReviewDate(update.getReviewDate());
            } else if (update.getStatus().trim().equalsIgnoreCase("Under
Review")){
                dog.setAdoptionStatus("Reserved");
                LocalDateTime currentDateTime = LocalDateTime.now();
                existing.setReviewDate(currentDateTime);
            }
            if (update.getApprovalDate()!=null){
                existing.setApprovalDate(update.getApprovalDate());
            } else if (update.getStatus().trim().equalsIgnoreCase("Approved")){
                dog.setAdoptionStatus("Adopted");
                LocalDateTime currentDateTime = LocalDateTime.now();
                existing.setApprovalDate(currentDateTime);
            }
            if (update.getApplicant()!=null){
                existing.setApplicant(update.getApplicant());
            }
            if (update.getDog()!=null){
                existing.setDog(update.getDog());
            }
```

```
            existing.setDog(dog);
            return applicationRepository.save(existing);
        }else{
            return null;
        }
    }

    public List<Application> getAllApplications() {
        return (List<Application>) applicationRepository.findAll();
    }

    public Application getApplicationById(int id) {
        return applicationRepository.findById(id).orElse(null);
    }

    public List<Application> searchByApplicantName(String applicantName) {
        return applicationRepository.findByApplicantName(applicantName);
    }

    public List<Application> searchByDogName(String dogName) {
        return applicationRepository.findByDogName(dogName);
    }

    public List<Application> searchByApplicantId(int id) {
        return applicationRepository.findByApplicantId(id);
    }

    public void deleteApplication(int applicationId){
        Application application =
applicationRepository.findById(applicationId).orElse(null);
        Dog dog = dogService.getDogById(application.getDog().getId());

        dog.setAdoptionStatus("Open");

        applicationRepository.deleteById(applicationId);
    }

}
```

```java
package com.project.backend.service;

import com.project.backend.model.Adopter;
import com.project.backend.model.Application;
import com.project.backend.model.Dog;
import com.project.backend.repository.DogRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.swing.text.html.Option;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Optional;

@Service
public class DogService {
    @Autowired
    DogRepository dogRepository;

    @Autowired
    ApplicationService applicationService;

    public Dog registerDog(Dog dog){
        dog.setRegisteredDate(new Date());
        dog.setAge(dog.getBirthDate());
        dog.setAdoptionStatus("Open");
        return dogRepository.save(dog);
    }

    public Dog updateDog(int dogId, Dog update){
        Optional<Dog> existingOptional = dogRepository.findById(dogId);

        if(existingOptional.isPresent()){
            Dog existing = existingOptional.get();
```

```
            if(update.getName()!=null){
                existing.setName(update.getName());
            }
            if(update.getDisplayImage()!=null){
                existing.setDisplayImage(update.getDisplayImage());
            }
            if(update.getBreed()!=null){
                existing.setBreed(update.getBreed());
            }
            if(update.getBirthDate()!=null){
                existing.setBirthDate(update.getBirthDate());
                existing.setAge(update.getBirthDate());
            }
            if(update.getGender()!=null){
                existing.setGender(update.getGender());
            }
            if(update.getColor()!=null){
                existing.setColor(update.getColor());
            }
            if(update.getSize()!=null){
                existing.setSize(update.getSize());
            }
            if(update.getAdoptionStatus()!=null){
                existing.setAdoptionStatus(update.getAdoptionStatus());
            }
            if(update.getDescription()!=null){
                existing.setDescription(update.getDescription());
            }
            if(update.getApplication()!=null){
                existing.setApplication(update.getApplication());
            }
[redacted]
            return dogRepository.save(existing);
        } else{
            return null;
        }
    }
[redacted]
    public void deleteDog(int dogId) {
        Dog dog = dogRepository.findById(dogId).orElse(null);
[redacted]
        Application application = dog.getApplication();
[redacted]
        // Delete each associated application
        if(application != null){
```

```
                applicationService.deleteApplication(application.getId());
            }
            dogRepository.deleteById(dogId);
        }

        public List<Dog> getAllDogs(){ return (List<Dog>) dogRepository.findAll(); }

        public Dog getDogById(int id) {
            return dogRepository.findById(id).orElse(null);
        }

        public List<Dog> getDogsByName(String name) {
            return dogRepository.findByName(name);
        }

        public List<Dog> getDogsByGender(String gender) {
            return dogRepository.findByGender(gender);
        }

        public List<Dog> getDogsByAdoptionStatus(String adoptionStatus) {
            return dogRepository.findByAdoptionStatus(adoptionStatus);
        }

    }
```

```java
package com.project.backend.service;

import org.hibernate.HibernateException;
import org.hibernate.engine.spi.SharedSessionContractImplementor;
import org.hibernate.id.IdentifierGenerator;
import java.io.Serializable;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

public class IdGenerator implements IdentifierGenerator {

    // Generates ID number for new adopter using the next number after the latest
ID registered
    @Override
    public Serializable generate(SharedSessionContractImplementor session, Object
object) throws HibernateException {
        Connection connection = session.connection();
        try {
            Statement statement = connection.createStatement();
            ResultSet rs = statement.executeQuery("SELECT MAX(id) as max_id FROM
model");
            if (rs.next()) {
                int maxId = rs.getInt("max_id");
                int nextId = maxId + 1;
                return nextId;
            }
        } catch (Exception e) {
            throw new HibernateException("Error.", e);
        }
        return null;
    }
}
```

CONTROLLERS

```java
package com.project.backend.controller;
[redacted]
import com.project.backend.model.Admin;
import com.project.backend.service.AdminService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
[redacted]
import java.util.List;
[redacted]
@CrossOrigin(origins="http://localhost:4200")
@RestController
@RequestMapping("/api/admin")
public class AdminController {
    @Autowired
    private AdminService adminService;
[redacted]
    @PostMapping("/register-admin")
    public Admin registerAdmin(@RequestBody Admin admin){
        return adminService.registerAdmin(admin);
    }
    @GetMapping("/get-admin/{adminId}")
    public Admin getAdminById(@PathVariable int adminId){
        return adminService.getAdminById(adminId);
    }
    @GetMapping("/all-admin")
    public List<Admin> getAllAdmins(){
        return adminService.getAllAdmins();
    }
    @DeleteMapping("/delete-admin/{adminId}")
    public void deleteAdmin(@PathVariable int adminId){
        adminService.deleteAdmin(adminId);
    }
```

```java
    @GetMapping("/get-admin/email/{email}")
    public ResponseEntity<Admin> getAdminByEmail(@PathVariable String email) {
        Admin admin = adminService.getAdminByEmail(email);

        if (admin != null) {
            return new ResponseEntity<>(admin, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
}
```

```java
package com.project.backend.controller;

import ch.qos.logback.classic.Logger;
import com.project.backend.model.Adopter;
import com.project.backend.service.AdopterService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@CrossOrigin(origins="http://localhost:4200")
@RestController
@RequestMapping("/api/adopter")
public class AdopterController {
    @Autowired
    private AdopterService adopterService;

    @PostMapping("/register-adopter")
    public ResponseEntity<String> registerUser(@RequestBody Adopter adopter) {
        try {
            // Validate and save the adopter registration data
            adopterService.registerAdopter(adopter);
```

```
            // Return a successful response
            return new ResponseEntity<>("Registration successful",
HttpStatus.OK);
        } catch (Exception e) {
            // Handle registration error and return an error response
            return new ResponseEntity<>("Registration failed",
HttpStatus.BAD_REQUEST);
        }
    }
```

[redacted]

```
    @PutMapping("/update-adopter/{adopterId}")
    public Adopter updateAdopter(@PathVariable int adopterId, @RequestBody
Adopter update) {
        return adopterService.updateAdopter(adopterId, update);
    }
    @GetMapping("/get-adopter/id/{adopterId}")
    public Adopter getAdopterById(@PathVariable int adopterId) {
        return adopterService.getAdopterById(adopterId);
    }
    @GetMapping("/get-adopter/name/{adopterName}")
    public List<Adopter> getAdoptersByName(@PathVariable String adopterName) {
        List<Adopter> adoptersByName = new ArrayList<>();
        adoptersByName = adopterService.getAdoptersByName(adopterName);
        return adoptersByName;
    }
    @GetMapping("/get-adopter/email/{adopterEmail}")
    public List<Adopter> getAdoptersByEmail(@PathVariable String adopterEmail){
        List<Adopter> adoptersByEmail = new ArrayList<>();
        adoptersByEmail = adopterService.getAdoptersByEmail(adopterEmail);
        return adoptersByEmail;
    }
    @GetMapping("/get-adopter/phone/{adopterPhoneNumber}")
    public List<Adopter> getAdoptersByPhoneNumber(@PathVariable String
adopterPhoneNumber){
        List<Adopter> adoptersByPhoneNumber = new ArrayList<>();
        adoptersByPhoneNumber =
adopterService.getAdoptersByPhoneNumber(adopterPhoneNumber);
        return adoptersByPhoneNumber;
    }
    @GetMapping("/all-adopter")
    public List<Adopter> getAllAdopters() { return
adopterService.getAllAdopters(); }
    @DeleteMapping("/delete-adopter/{adopterId}")
    public void deleteAdopter(@PathVariable int adopterId) {
adopterService.deleteAdopter(adopterId); }
```

```
}
```

```
package com.project.backend.controller;

import com.project.backend.model.Application;
import com.project.backend.service.ApplicationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@CrossOrigin(origins="http://localhost:4200")
@RestController
@RequestMapping("/api/application")
public class ApplicationController {
    @Autowired
    private ApplicationService applicationService;

    @PostMapping("/submit")
    public Application submitApplication(@RequestBody Application application){
        return applicationService.createApplication(application);

    }
    @PutMapping("/update/{id}")
    public Application updateApplication(@PathVariable int id,@RequestBody
Application application){
        return applicationService.updateApplication(id, application);
    }
    @GetMapping("/get-application/id/{id}")
    public Application getApplicationById(@PathVariable int id){
        return applicationService.getApplicationById(id);
    }
    @GetMapping("/get-application/applicant/{name}")
    public List<Application> getApplicationByApplicant(@PathVariable String
name){
        List<Application> applicationsByApplicant = new ArrayList<>();
```

```
        applicationsByApplicant = applicationService.searchByApplicantName(name);
        return applicationsByApplicant;
    }
    @GetMapping("/get-application/dog/{name}")
    public List<Application> getApplicationsByDog(@PathVariable String name){
        List<Application> applicationsByDog = new ArrayList<>();
        applicationsByDog = applicationService.searchByDogName(name);
        return applicationsByDog;
    }
    @GetMapping("/get-application/applicant-id/{id}")
    public List<Application> getApplicationsByApplicantId(@PathVariable int id){
        List<Application> applicationsByApplicantId = new ArrayList<>();
        applicationsByApplicantId = applicationService.searchByApplicantId(id);
        return applicationsByApplicantId;
    }
    @GetMapping("/all-application")
    public List<Application> getAllApplications(){
        return applicationService.getAllApplications();
    }
    @DeleteMapping("/delete/{id}")
    public void deleteApplication(@PathVariable int id){
        applicationService.deleteApplication(id);
    }
[redacted]
}
[redacted]
```

```
package com.project.backend.controller;
[redacted]
import com.project.backend.model.Dog;
import com.project.backend.service.DogService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
[redacted]
import java.util.List;
[redacted]
@RestController
@CrossOrigin(origins="http://localhost:4200")
@RequestMapping("/api/dog")
public class DogController {
    @Autowired
```

```java
    private DogService dogService;

    @PostMapping("/register-dog")
    public Dog registerDog(@RequestBody Dog dog) {
        return dogService.registerDog(dog);
    }
    @PutMapping("/update-dog/{dogId}")
    public Dog updateDog(@PathVariable int dogId, @RequestBody Dog update) {
        return dogService.updateDog(dogId, update);
    }
    @DeleteMapping("/delete-dog/{dogId}")
    public void deleteDog(@PathVariable int dogId) {
        dogService.deleteDog(dogId);
    }
    @GetMapping("/get-dog/id/{id}")
    public Dog getDogById(@PathVariable int id) {
        return dogService.getDogById(id);
    }
    @GetMapping("/get-dog/name/{name}")
    public List<Dog> getDogsByName(@PathVariable String name) {
        return dogService.getDogsByName(name);
    }
    @GetMapping("/get-dog/gender/{gender}")
    public List<Dog> getDogsByGender(@PathVariable String gender) {
        return dogService.getDogsByGender(gender);
    }
    @GetMapping("/get-dog/status/{status}")
    public List<Dog> getDogsByAdoptionStatus(@PathVariable String status) {
        return dogService.getDogsByAdoptionStatus(status);
    }
    @GetMapping("/all-dog")
    public List<Dog> getAllDogs() {
        return dogService.getAllDogs();
    }

}
```

```java
package com.project.backend.controller;
```

```java
import com.project.backend.model.Admin;
import com.project.backend.model.Adopter;
import com.project.backend.model.LoginRequest;
import com.project.backend.model.LoginResponse;
import com.project.backend.service.AdminService;
import com.project.backend.service.AdopterService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin(origins="http://localhost:4200")
@RestController
public class AuthenticateController {
    @Autowired
    private AdopterService adopterService;

    @Autowired
    private AdminService adminService;

    @PostMapping("/api/admin/login")
    public ResponseEntity<String> loginAsAdmin(@RequestBody LoginRequest user) {

        Admin confirmUser = adminService.getAdminByEmail(user.getEmail());

        if (confirmUser != null &&
confirmUser.getPassword().equals(user.getPassword())) {
            int userId = confirmUser.getId();
            return ResponseEntity.ok("Login successful");
        } else {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Login
failed");
        }
    }

    @PostMapping("/api/adopter/login")
    public ResponseEntity<LoginResponse> loginAsAdopter(@RequestBody LoginRequest
user) {
        Adopter confirmUser = adopterService.getUserByEmail(user.getEmail());

        if (confirmUser != null &&
confirmUser.getPassword().equals(user.getPassword())) {
            int userId = confirmUser.getId();
```

```
            return ResponseEntity.ok(new LoginResponse(userId, "Login
successful"));
        } else {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(new
LoginResponse(0, "Login failed"));
        }
    }
[redacted]
}
```