Date: 29/01/2024
Commit ID: d4acee0

PetConnect

Software Architecture Report

Date: 29/01/2024
Commit ID: d4acee0

Contents

Date: 29/01/2024
Commit ID: d4acee0

About the project

The primary objective of PetConnect is to establish a comprehensive and user-friendly platform facilitating pet adoption. The software aims to create a seamless experience for users, from initial registration and profile setup to exploring adoption announcements and engaging with the community. The platform intends to bridge the gap between potential pet adopters and pet owners, providing a secure and engaging environment for users with diverse needs and preferences.

Fulfilled Capabilities

- Implemented a unified and secure user authentication process for login and registration.
- Established a basic profile setup feature for new users.
- Developed an "About Us" section to provide detailed insights into the platform.
- Featured blog articles on the homepage related to pet care and adoption.
- Incorporated testimonials for users to gain insights into others' experiences.
- Implemented a Terms and Conditions pop-up for user acknowledgment.
- Provided a contact form for easy communication with the platform's team.
- Facilitated a seamless adoption announcements lifecycle for pet owners, allowing them to create, update, and remove announcements to ensure accuracy and relevance.
- Provided visitors with the capability to explore and filter adoption announcements based on specific criteria, enhancing the ability to find animals that match preferences and lifestyle.
- Implemented a commenting system, allowing potential adopters to leave comments on adoption announcements for inquiries and community engagement.
- Enabled users to send messages to pet owners, facilitating communication about details regarding the animal or expressing interest in adoption.
- Streamlined the pet adoption application process for potential adopters, allowing them to submit applications easily and efficiently.
- Implemented content moderation tools for administrators to review and manage user-generated content, including adoption announcements, comments, and posts, ensuring a positive and safe community environment.
- Introduced role-based permissions, allowing administrators to assign different roles and responsibilities to staff members involved in platform management, facilitating effective delegation.

What to do next

- Enhance User Profiles: Develop and implement an advanced user profile system, allowing users to provide more detailed information about themselves and their

Date: 29/01/2024
Commit ID: d4acee0

preferences. This may include additional fields such as personal preferences, lifestyle, and past pet ownership experiences.

- Improve UI/UX: Focus on refining the user interface (UI) and user experience (UX) to make the platform more intuitive and user-friendly. This involves optimizing navigation, visual elements, and overall design to ensure a seamless and enjoyable experience for users interacting with the pet adoption platform.

Guides

Run the Project Locally

1. Clone the repository from GitHub:

   ```
   git clone https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-party.git
   ```

2. Navigate to the project directory:

   ```
   cd petconnect
   ```

3. Ensure you have ASP.NET Core 6 SDK installed.
4. Run the application:

   ```
   dotnet run
   ```

5. Open your browser and navigate to `http://localhost:5000` to access the locally running PetConnect application.

Build the Project

1. Navigate to the project directory:

   ```
   cd petconnect
   ```

2. Ensure you have ASP.NET Core 6 SDK installed.
3. Build the application:

   ```
   dotnet build
   ```

Deploy the Project

Local Deployment

1. Ensure you have a suitable hosting environment, such as IIS.
2. Build the project using the instructions above.

Date: 29/01/2024
Commit ID: d4acee0

3. Deploy the built application to your local hosting environment.

Contributions

Design pattens used

MVC (Model-View-Controller)

PetConnect follows the MVC architectural pattern, where the application logic is divided into three components:

- **Model:** Represents the data and business logic of the application.
- **View:** Manages the presentation and user interface.
- **Controller:** Handles user input, processes requests, and interacts with the Model and View.

How to contribuite

1. Fork the repository on GitHub.
2. Clone your fork locally:

```
git clone https://github.com/inginerie-software-2023-2024/proiect-inginerie-software-party.git
```

3. Create a new branch for your changes:

```
git checkout -b feature/your-feature
```

4. Make your changes and commit them.
5. Push the changes to your fork:

```
git push origin feature/your-feature
```

6. Open a pull request to the main repository, describing your changes and improvements.

Application entry points

The application entry point is the `Program.cs` file. Specifically, the `app.Run()` method is the last method called in the `Program` class, and it effectively serves as the entry point for the PetConnect application.

Date: 29/01/2024
Commit ID: d4acee0

Data sources

The application relies on SQL Server as its only data source. The connection string is stored in the `appsettings.json` file.

Entity Framework Core is utilized as the Object-Relational Mapping (ORM) framework to interact with the database.

Data inputs

PetConnect receives data inputs through:

- **User Inputs:** Users provide data through the application's user interface, such as registration forms, profile setup, and adoption applications.

Configuration files

PetConnect has the following configuration files:

- **appsettings.json:** Contains the database connection string

User journey

The user journey can be found here.

Most valuable output

The most valuable output of the PetConnect web app is the seamless and user-centric pet adoption process, embodying the platform's core mission to connect potential adopters with furry companions in need of loving homes. This feature serves as the heart of PetConnect, encapsulating a user experience designed to facilitate and enhance the adoption journey for both pet seekers and pet owners.

The adoption process begins with a user-friendly interface that allows visitors to explore and filter adoption announcements based on specific criteria such as species, age, and location. The platform provides detailed descriptions of each animal, including characteristics, photos, and relevant information, empowering potential adopters to make informed decisions about which pet aligns with their preferences and lifestyle.

PetConnect distinguishes itself by enabling a direct and meaningful interaction between potential adopters and pet owners. The platform allows users to leave comments on adoption announcements, fostering a sense of community engagement and providing a space for inquiries and discussions. Moreover, the messaging functionality allows interested adopters to communicate directly with pet owners, facilitating a transparent and personal dialogue about the pet's history, behavior, and needs.

Date: 29/01/2024
Commit ID: d4acee0

In essence, the adoption process within PetConnect embodies the platform's commitment to creating meaningful connections between animals and caring individuals. By prioritizing user engagement, transparent communication, and a streamlined application process, PetConnect delivers a valuable and emotionally resonant experience that transcends a mere transactional adoption process, fostering a community dedicated to the well-being of pets and their potential adopters.

Development plans

How the CI/CD pipeline works

The Continuous Integration and Continuous Deployment (CI/CD) pipeline for PetConnect is implemented using GitHub Actions. The config file can be found in ".github/workflows" directory in the PetConnect GitHub repository.

At the time of the writing of the Architecture report, only the CI pipeline is implemented.

CI pipeline

- The CI pipeline is triggered automatically whenever changes are pushed to the main branch of the repository.
- GitHub Actions checks out the latest code from the main branch and initiates the build process.
- Automatic build checks are performed to ensure that the code compiles successfully.

QA process

Unit Tests

The unit test are implemented using NUnit. The mocking of test data is implemented using Moq Framework.

Test Suites

UsersController

- **ShowProfile_WithValidId_ReturnsViewResult:**
  - **Purpose:** Verifies that the `ShowProfile` action in the `UserController` returns a `ViewResult` when provided with a valid user ID.
  - **Method:** The test sets up the necessary mocks for the `UserManager` and asserts that the result is a `ViewResult` with the correct view name and model.
- **EditProfile_UserDoesNotExist_ReturnsNotFoundResult:**

Date: 29/01/2024
Commit ID: d4acee0

- **Purpose:** Ensures that the `EditProfile` action returns a `NotFoundResult` when the user does not exist.
- **Method:** The test sets up mocks for `ClaimsPrincipal` and `UserManager` to simulate a scenario where the user does not exist. It then asserts that the result is a `NotFoundResult`.

➢ **EditProfile_UserExists_ReturnsViewResultWithUser:**
- **Purpose:** Validates that the `EditProfile` action returns a `ViewResult` with the correct user model when the user exists.
- **Method:** Similar to the previous test, this one ensures that the action returns a `ViewResult` with the correct user model when the user exists.

➢ **ShowProfile_WithInvalidId_ReturnsNotFoundResult:**
- **Purpose:** Verifies that the `ShowProfile` action returns a `NotFoundResult` when provided with an invalid user ID.
- **Method:** The test sets up the mock for the `UserManager` to simulate a scenario where the user ID is invalid, and it asserts that the result is a `NotFoundResult`.

BlogsController

➢ **Blog1_ReturnsViewResult:**

- **Purpose:** Verifies that the `Blog1` action in the `BlogsController` returns a `ViewResult`.
- **Method:**
  - Arranges an instance of the `BlogsController`.
  - Acts by calling the `Blog1` action.
  - Asserts that the result is not null and is of type `ViewResult`.

➢ **Blog2_ReturnsViewResult:**

- **Purpose:** Ensures that the `Blog2` action in the `BlogsController` returns a `ViewResult`.
- **Method:**
  - Arranges an instance of the `BlogsController`.
  - Acts by calling the `Blog2` action.
  - Asserts that the result is not null and is of type `ViewResult`.

➢ **Blog3_ReturnsViewResult:**

- **Purpose:** Validates that the `Blog3` action in the `BlogsController` returns a `ViewResult`.
- **Method:**
  - Arranges an instance of the `BlogsController`.
  - Acts by calling the `Blog3` action.
  - Asserts that the result is not null and is of type `ViewResult`.

Date: 29/01/2024
Commit ID: d4acee0

Dependencies

APIs Used

Asp.net Identity

Asp.net Identity is utilized for user authentication and authorization within the PetConnect project. It provides a robust framework for managing user identities, including features like user registration, login, and role-based access control.

Libraries

Tailwind CSS

Tailwind CSS is a utility-first CSS framework used for styling the user interface of the PetConnect application. It streamlines the styling process by providing a set of utility classes that can be easily applied to HTML elements, allowing for efficient and consistent UI development.

SignalR

SignalR is a library for adding real-time functionality to web applications. In the context of PetConnect, SignalR may be employed to enable real-time communication between users, facilitating features such as instant messaging or live updates.

Dependency confusion

PetConnect, like any project with external dependencies, is potentially susceptible to Dependency Confusion. It's essential to implement best practices to mitigate this risk:

- Regularly update dependencies to their latest secure versions.
- Use package-lock files or similar mechanisms to lock dependency versions.
- Verify the legitimacy of external packages and repositories.
- Leverage security tools and services that can detect and prevent dependency-related vulnerabilities.

Date: 29/01/2024
Commit ID: d4acee0