Safari Across America

Software Design Description

Prepared by: Laurence Finn

Document Version: 1.0, June 13, 2024

Revision Control Page

Version	Date	Author(s)	Updates
1.0	6/13/2024	Laurence Finn	Initial release

Table of Contents

1. Introduction 4

1.1. Document Scope 4

1.2. System Overview 4

1.3. References 4

2. Source Code 5

2.1. Source Code File Types 5

2.2. Source Code Directory Structure 5

3. Database Design 6

3.1. Entity Relationship Diagram – Animals 7

3.2. Entity Relationship Diagram – ASP.NET Identity Tables 7

3.3. Stored Procedures 8

4. Website Design 9

4.1. Programming Languages 9

4.2. ASP.NET Classes 10

5. Program Workflow 12

1. Introduction

1.1. Document Scope

The purpose of this Software Design Description (SDD) is to provide an overview of the Safari Across America website software architecture, code architecture, and database.

1.2. System Overview

Safari Across America is a website that tracks data about exotic wild animals across the United States. The website consists of five main modules – View All Animals, the Interactive Map, Animal Details, and the capabilities to Submit Animals and Submit Images for existing animals. There is also an Administrative Moderation module that allows approving, editing, and deleting animals and images.

The website uses ASP.NET Core with Razor Pages and C# for the front-end UI/UX, and a Microsoft SQL Server database for the back-end data storage. It can be deployed to a Microsoft Internet Information Services (IIS) or Kestrel server.

The main feature of the site is the Interactive Map, where users can click on a state to view all the animals that live there. The site also allows visitors to submit animals of their own (with images), as well as view the entire database in a sortable, filterable table.

Of note, the website also allows users to view an informational "Details" page about each animal. These pages include the typical height, weight, diet, and lifespan of the animal with color photos. The "Details" page design is a report template that populates according to the animal selected by the user. It can then be printed by the user.

The purpose of this document is to provide a detailed breakdown of the system design, including an overview of the database and ASP.NET Core classes/webpages.

The codebase for the project can be found here: https://github.com/laurence-finn/safari

1.3. References

The cover page image is a royalty-free image used in compliance with the Pixabay Content License. The image source can be found here: https://pixabay.com/photos/peacock-bird-animal-plumage-90051/

2. Source Code

2.1. Source Code File Types

The project contains different types of code files, which follow the traditional pattern and best practices for ASP.NET Core web applications. These types include:

- **Data Context:** The data context class is responsible for configuring the relationships between models in the database, as well as connecting/configuring the database itself. For example, the data context establishes that one Diet Type can be connected to many Animals.

- **Data Entities (Models):** These are classes which correspond directly with the tables in the SQL database. They contain properties that are accessed by the Repository and Page Models.

- **Repository:** The repository is a C#/.NET class that contains the code to create, update, and delete information from the SQL database by executing stored procedures. The methods in the repository class are called by the Page Models.

- **Page Models:** Also known as "code-behind classes," these are C#/.NET classes that handle HTTP GET and HTTP POST messages that are sent from the Razor Pages.

- **Razor Pages (Views):** These are cshtml files, which are like HTML files except they can also contain C# code for handling business logic (like iterating through a list of Animals to display them in a table, for example). The views contain all the front-end code that the user sees directly, as well as elements like buttons that send HTTP POST messages to the Page Models.

- **Partial Views:** These are partial razor page files that are intended be rendered inside of the Razor Pages (Views).

2.2. Source Code Directory Structure

The following is a summary of the source code directories and their contents:

Table 2-1: Source Code Directory Structure

Directory	Usage
Safari.Data	Contains the data elements of the project.
Safari.Data/DataContext	Contains WildlifeDataContext, which is the data context class for the project.
Safari.Data/Entities	Contains the Data Entity (Model) classes for the project.
Safari.Data/Migrations	Contains the Entity Framework Core migrations for the project.

Directory	Usage
Safari.Data/Repositories	Contains the WildlifeRepository class (and its interface IWildlifeRepository).
Safari.Data/Scripts	Contains SQL scripts for creating the database and modifying the individual stored procedures.
Safari.Web	Contains the web elements of the project, as well as Program.cs, which is the main class that builds the app container. Appsettings.json is also present in this folder and contains settings like the database connection string.
Safari.Web/Properties	Contains the launchSettings.json file, which specifies how the application should be launched in development or deployment environments.
Safari.Web/wwwroot	Contains the static files, like images, that the client can connect to.
Safari.Web/Areas/Identity/Pages	Contains the files required for running the ASP.NET Core Identity features of the website (like login).
Safari.Web/Pages	Contains the webpages for the project.
Safari.Web/Pages/Admin	Contains the webpages for the Admin areas of the website.
Safari.Web/Pages/Shared	Contains shared/partial views that are rendered inside of other webpages.
Safari.Web/Profiles	Contains AnimalProfile, which is used by AutoMapper to map the data in AnimalViewModel.
Safari.Web/ViewModels	Contains AnimalViewModel, which is used for displaying animal data in tables.

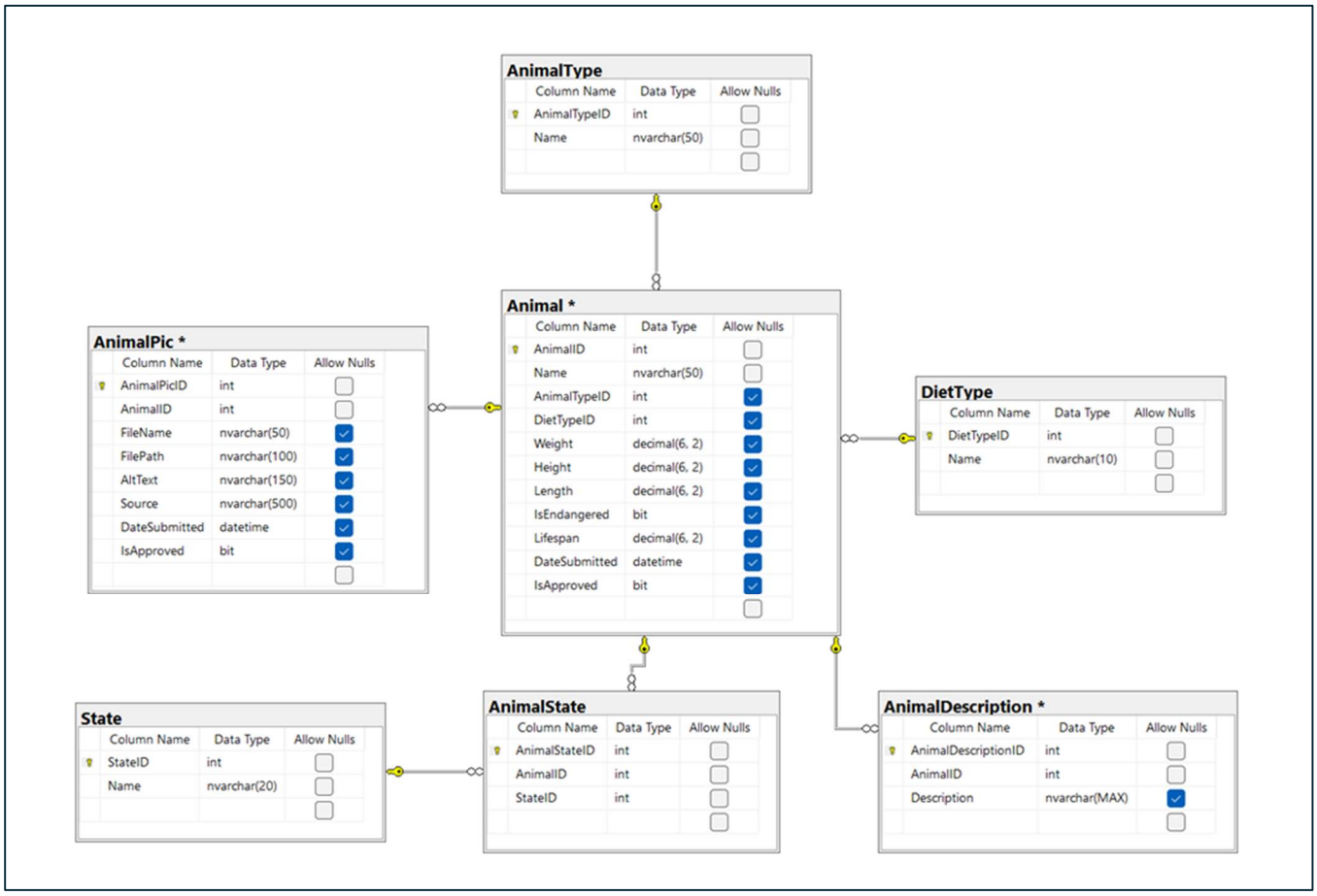
3. Database Design

Microsoft SQL Server provides the data storage for Safari Across America. The database is organized into relational entities, and stored procedures provide the create, read, update, and delete functions. Database maintenance (such as data corrections) shall be achieved by using these stored procedures or other SQL queries directly in SQL Server Management Studio.

3.1. Entity Relationship Diagram – Animals

The following is an Entity Relationship Diagram (ERD) for “WildlifeData”, the database used for Safari Across America. This diagram addresses the tables that store data that is displayed on the site:

Figure 3-1: Entity Relationship Diagram - Animals



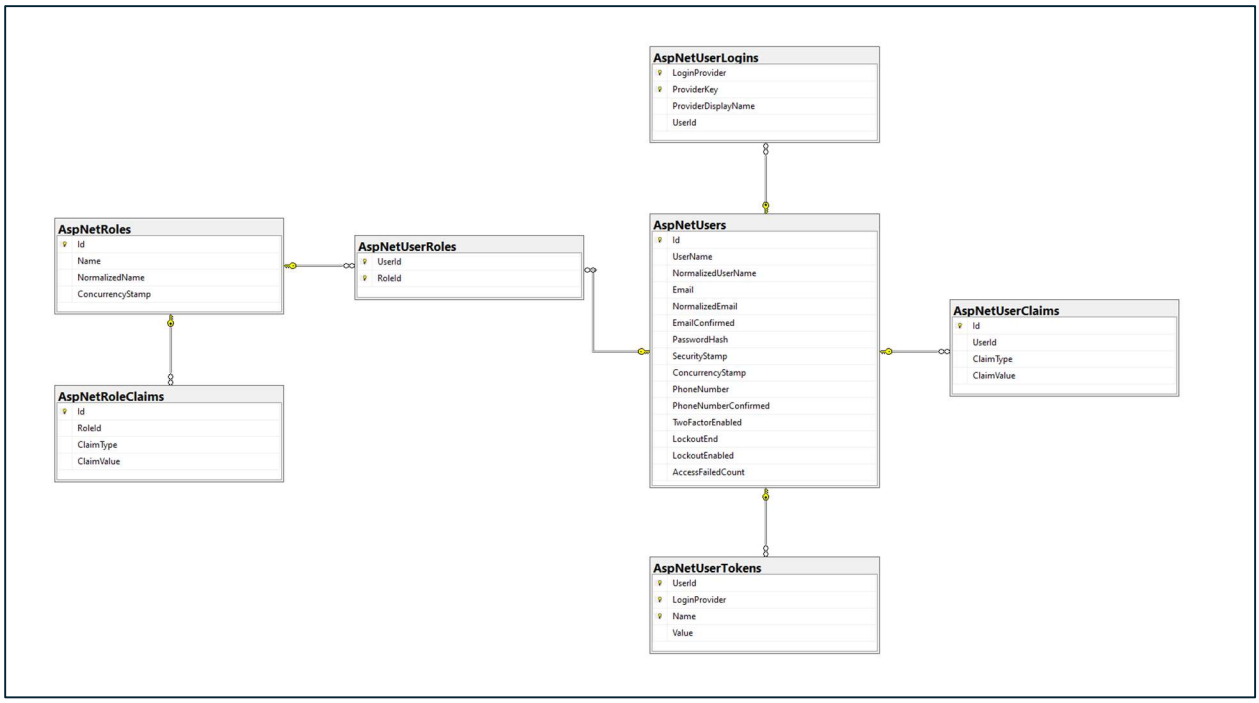
The Animal entity is the primary table within the database. It contains the AnimalTypeID, which relates to AnimalType, a lookup table that contains the main types of animals in the database (like Mammals or Reptiles). Similarly, it has DietTypeID, which relates to the lookup table DietType, containing three types of animal diet (Carnivore, Omnivore, Herbivore). Images are stored in the AnimalPic table, and one animal can have many pictures. Descriptions are stored in the AnimalDescription table as raw HTML. State is a lookup table containing all 50 states, and AnimalState is a table relating animals to states (as they can have many).

3.2. Entity Relationship Diagram – ASP.NET Identity Tables

The SQL database contains tables which are automatically created (“scaffolded”) when implementing the ASP.NET Core Identity feature, which this project uses to manage the

Administrator account and the areas it is authorized to access. The following is an ERD of these tables:

Figure 3-2: Entity Relationship Diagram - ASP.NET Identity Tables



While the database currently only contains one pre-seeded Admin account, future updates could include more users and user types.

3.3. Stored Procedures

The following stored procedures are present in the SQL database. Stored procedures were chosen for their high performance and security through server-side input validation (to prevent attacks such as SQL injection).

- **Insert_Animal:** Creates a record in the Animal table.
- **Insert_AnimalDescription:** Creates a record in the AnimalDescription table.
- **Insert_AnimalPic:** Creates a record in the AnimalPic table, noting the filename and filepath of the image that was uploaded to the website.
- **Insert_AnimalState:** Creates a record in the AnimalState table (which relates AnimalIds and StateIds in a many-to-many relationship).
- **Update_Animal:** Updates an Animal record.
- **Update_AnimalDescription:** Updates an AnimalDescription record.
- **Update_AnimalPic:** Updates an AnimalPic record.

- **Delete_Animal:** Deletes an Animal record. This stored procedure also deletes all animal descriptions, pictures, and states associated with the animal.
- **Delete_AnimalDescription:** Deletes an AnimalDescription record.
- **Delete_AnimalState:** Deletes an AnimalState record.
- **Delete_AnimalPic:** Deletes an AnimalPic record; deleting the image file itself is handled by the website.
- **Delete_AllAnimalState:** Deletes all AnimalState records for a specified AnimalId.

4. Website Design

4.1. Programming Languages

The following programming languages are used within this project:

- **C#/.NET:** C# and .NET are used within this project to provide the business logic between the data store (SQL database) and the front-end components (the web pages).
- **HTML:** The Hypertext Markup Language (HTML) is used to
- **CSS:** Cascading Style Sheet (CSS) language is used to provide the style settings for the web pages in this project.
- **JavaScript:** JavaScript is used for interactive front-end features on the website, such as the input validation, multi-selector boxes for states, and HTML text editing for animal descriptions.

Additionally, the following libraries are used:

- **ASP.NET Core:** This is a C#/.NET framework for building web applications.
- **Automapper:** A C#/.NET framework for mapping models to view models.
- **Bootstrap:** This is a CSS framework that provides useful tools such as image carousels and navigation bars.
- **DataTables:** A JavaScript library that enhances HTML tables with sorting and filtering tools.
- **FontAwesome:** A CSS library that provides icons, such as the question mark info icon used on submission forms in the website.
- **JQuery Validation:** This is the standard JavaScript library that is used with ASP.NET to provide input validation for submission forms. Specifically, it used for creating and updating Animal and AnimalPic records on the website.
- **Select2:** A JavaScript library that provides the multi-selector tool for picking states of residency for animals.
- **TinyMCE:** A JavaScript library that provides HTML text editing tools for the Animal Description text boxes present in the submission forms on this website.

4.2. ASP.NET Classes

The source code for Safari Across America follows the ASP.NET Razor Pages design pattern, which includes entity models, page models, and other classes that are used for executing the business logic of the website. The following is a list of the classes with their location in the file directory structure of the source code.

Animal | Safari.Data/Entities/Animal.cs

This is an Entity/Model class that contains properties corresponding with the Animal table in the SQL database.

AnimalDescription | Safari.Data/Entities/AnimalDescription.cs

This is an Entity/Model class that contains properties corresponding with the AnimalDescription table in the SQL database.

AnimalPic | Safari.Data/Entities/AnimalPic.cs

This is an Entity/Model class that contains properties corresponding with the AnimalPic table in the SQL database.

AnimalState | Safari.Data/Entities/AnimalState.cs

This is an Entity/Model class that contains properties corresponding with the AnimalState table in the SQL database.

AnimalType | Safari.Data/Entities/AnimalType.cs

This is an Entity/Model class that contains properties corresponding with the AnimalType table in the SQL database.

ApplicationUser | Safari.Data/Entities/AnimalDescription.cs

This is an Entity/Model class that contains properties corresponding with the AspNetUsers table in the SQL database.

DietType | Safari.Data/Entities/DietType.cs

This is an Entity/Model class that contains properties corresponding with the DietType table in the SQL database.

State | Safari.Data/Entities/State.cs

This is an Entity/Model class that contains properties corresponding with the State table in the SQL database.

IWildlifeRepository | Safari.Data/Repositories/IWildlifeRepository.cs

This is an Interface class that defines the methods that must be present in the WildlifeRepository class.

WildlifeRepository | Safari.Data/Repositories/WildlifeRepository.cs

This is a Repository class, implementing the IWildlifeRepository interface, which defines the data access methods such as AddAnimalAsync that call related stored procedures in the SQL database.

LoginModel | Safari.Web/Areas/Identity/Pages/Account/Login.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Admin Login page on the website.

RegisterModel | Safari.Web/Areas/Identity/Pages/Account/Register.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Register page on the website. The code for this page is designed to disable registration for any users, as the single user account (an admin) is already pre-seeded.

AdminAnimalsPageModel | Safari.Web/Pages/Admin/AdminAnimals.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Moderate Animals page on the website.

AdminImagesPageModel | Safari.Web/Pages/Admin/AdminAnimals.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Moderate Images page on the website.

AdminIndexPageModel | Safari.Web/Pages/Admin/AdminIndex.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Moderate (Admin Index) page on the website.

ApproveEditAnimalPageModel | Safari.Web/Pages/Admin/ApproveEditAnimal.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Approve/Edit Animal page on the website.

ApproveEditImagePageModel | Safari.Web/Pages/Admin/ApproveEditImage.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Approve/Edit Image page on the website.

DeleteAnimalPageModel | Safari.Web/Pages/Admin/DeleteAnimal.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Delete Animal page on the website.

DeleteImagePageModel | Safari.Web/Pages/Admin/DeleteImage.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Delete Image page on the website.

DetailsPageModel | Safari.Web/Pages/Details.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Details page on the website.

ErrorModel | Safari.Web/Pages/Error.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Error page on the website.

IndexPageModel | Safari.Web/Pages/Index.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Index (Home) page on the website.

MapPageModel | Safari.Web/Pages/Map.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Map page on the website.

ReferencesPageModel | Safari.Web/Pages/References.cshtml.cs

This is a Page Model class that serves as the code-behind class for the References page on the website.

SubmitAnimalPageModel | Safari.Web/Pages/SubmitAnimal.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Submit Animal page on the website.

SubmitImagePageModel | Safari.Web/Pages/SubmitImage.cshtml.cs

This is a Page Model class that serves as the code-behind class for the Submit Image page on the website.

ViewAnimalsPageModel | Safari.Web/Pages/ViewAnimals.cshtml.cs

This is a Page Model class that serves as the code-behind class for the View Animals page on the website.

AnimalProfile | Safari.Web/Profiles/AnimalProfile.cs

This is a Profile class that is required to use AutoMapper to map the Animal model to the AnimalViewModel view model.

AnimalViewModel | Safari.Web/ViewModels/AnimalViewModel.cs

This is a View Model class that takes information from other models (such as Animal and AnimalPic) and collects them into a single view model for use in tables, like those found in the View All or Map pages.

Program | Safari.Web/Program.cs

This is the main class that runs the program. It adds all the required services to an app container and starts the web application.

5. Program Workflow

The program flow for this project follows the standard for typical ASP.NET web services. It begins with the server running Program.cs, which creates a container with the services required for the app

to function. The HTTPS service is also started at this time, and the SQL database is connected. At this point, the server listens for connections from the client (the user's browser). Users then have control over the program through their client. The application exits if the process is terminated on the server side.

Figure 5-1: Program Workflow