GalleryGenie
An Intelligent Image Tagging System

Konrad Simlinger

April 28, 2023

Contents

1 Description 3

2 Database 4
2.1 Database Design 4
2.1.1 EER-Model 5
2.2 Database Development Methodology 6

3 User Interface 7
3.1 Prototyping 7
3.1.1 Login Screen 7
3.2 Colour Palletes 8
3.3 Structure 9

4 Code Architecture 9
4.1 Structure 9
4.2 GalleryGenieApp 9
4.3 GalleryGenieServer 9
4.4 Developer Notes 9

1 Description

GalleryGenie is an intuitive and intelligent image organization and management application that streamlines the process of managing large collections of digital images. The application offers an easy-to-use and visually appealing interface for storing, categorizing, and searching for images. ~~With its advanced image analysis algorithms, GalleryGenie automatically generates tags for each uploaded image, which helps users to easily search for and find specific images~~. Additionally, when a new image is uploaded, the application compares it to the images already stored in the gallery and identifies any similarities using a sophisticated likeness scoring algorithm. If a high likeness score is detected, the user is prompted to automatically copy some or all of the tags from the compared image, saving the user time and effort. With features like automatic tagging and similarity detection, GalleryGenie is the perfect tool for anyone looking to efficiently manage their digital image collections.

2 Database

2.1 Database Design

u_users	
u_id	INT PK
u_name	VARCHAR
u_creationdate	DATE

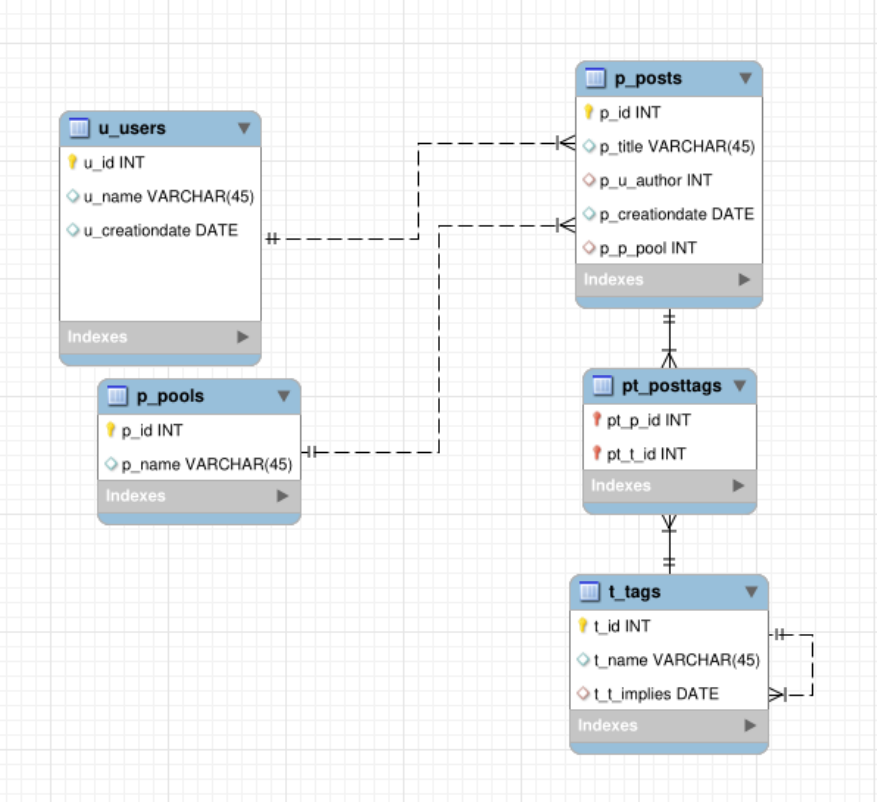
p_pools	
p_id	INT PK
u_name	VARCHAR

p_posts	
p_id	INT PK
p_title	VARCHAR
p_u_author	INT FK
p_p_pool	INT FK
p_creationdate	DATE

pt_posttags	
pt_p_id	INT PK FK
pt_t_id	INT PK FK

t_tags	
t_id	INT PK
t_name	VARCHAR
t_t_implies	INT FK

2.1.1 EER-Model



2.2 Database Development Methodology

I chose to use the code-first database development methodology for my project because it allows me to quickly and easily develop and iterate on my database schema directly within my application code. With code-first development, I can define my data model in a simple, declarative syntax using C# classes and annotations, rather than having to manually create tables and columns using SQL scripts or external database management tools.

This approach has several advantages. First, it makes it much faster and easier to develop my database schema, as I can simply define it alongside the rest of my application code, without having to switch back and forth between different tools or languages. Additionally, by using C# classes to define my data model, I can take advantage of the full range of object-oriented programming techniques, such as inheritance, encapsulation, and abstraction, to create a more modular and maintainable schema.

Another major advantage of code-first development is that it allows me to easily integrate my database development process with my overall software development workflow. By defining my data model directly in code, I can use the same version control tools and development processes that I use for the rest of my application code, making it easier to manage changes and track issues.

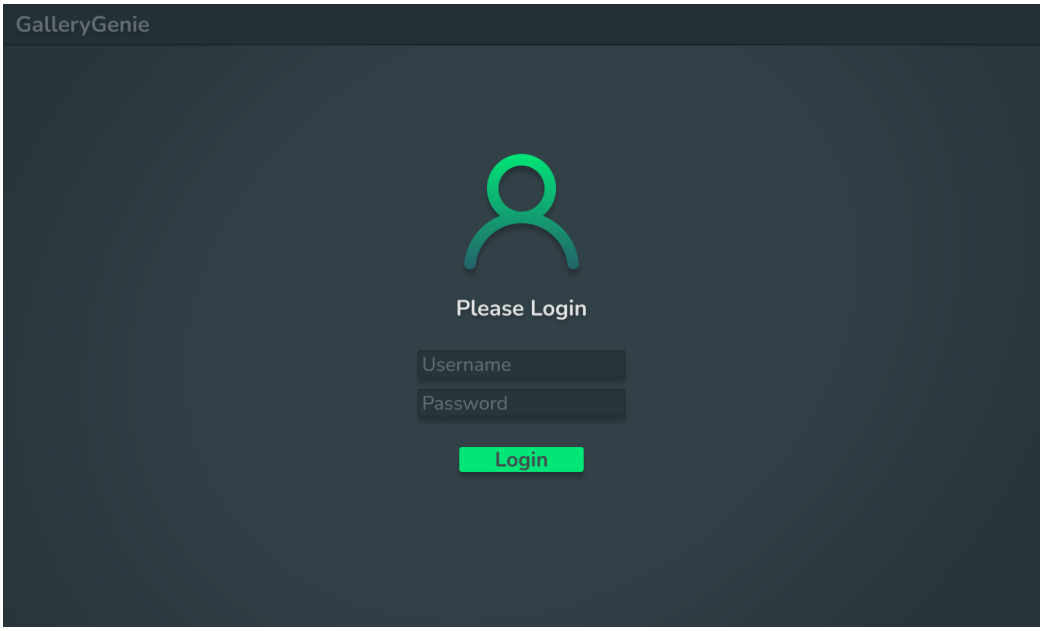
Overall, the code-first approach allows me to rapidly develop and iterate on my database schema, while also integrating it more seamlessly with the rest of my application code. By enabling me to take full advantage of object-oriented programming techniques and to use the same development tools and processes for my data model as for the rest of my code, code-first development is a powerful and flexible methodology that is well-suited to my project's needs.

3 User Interface

The user interface of my application is designed to be intuitive and user-friendly. Each screen is carefully crafted to provide the user with the information and controls they need to easily manage and organize their images. The user interface will feature a clean and modern design, with a focus on simplicity and ease-of-use. It also incorporates a range of interactive features, such as drag-and-drop image uploading and automatic tag suggestions, to make the user experience as smooth and efficient as possible.

3.1 Prototyping

3.1.1 Login Screen



3.2 Colour Palletes

3.3 Structure

4 Code Architecture

4.1 Structure

The Project is strictly divided into two parts. The GalleryGenieApp and GalleryGenieServer. Currently the idea is that the former does not need a live server to function. But instead the Server is an addition to the client. Although the WPF-App is capable of saving and managing the images on its own, it can be used in unison with the GalleryGenieServer which functions in a very simple way.

4.2 GalleryGenieApp

4.3 GalleryGenieServer

The GalleryGenieServer is a plain .NET Application that listens for connection from a GalleryGenieClient. Once a client connects to the Server over the internet, every action that is taken on the client gets rerouted to the server instead of being saved and executed in the local storage.

4.4 Developer Notes

Because this project is designed to be done in and for school, I really don't expect to finish the GalleryGenieApp and the GalleryGenieServer. Mostly because I overshot my own capabilities and free time. So the main priority of the project lingers on the WPF-App.