Adventure Works Photo Sharing Application (Proposed)

Detailed Planning Document

2/12/2020

Authors: *Justo Garrido*

Introduction

The author has examined the initial investigation document. Based on the use cases, technical requirements, and other content in that document, the author has created the detailed plans below. The board has already agreed that the photo sharing application will be built as a website based on Microsoft's ASP.NET MVC technology. Therefore the details presented here include the names and properties of model classes and controllers developers must create. Views have also been identified and wireframe diagrams included to help envision the user interface for important parts of the site.

The application design is likely to evolve throughout the development process as requirements change. The development team will adopt Agile practices to ensure such changes are reflected in the final product. Therefore this document should not be considered a complete definition of the final application.

MVC Model

Developers will create a model with the following model classes. For each model class, properties have been listed and descriptions given.

Table 1: MVC Model

	Model Class	Description	Properties	Data Types
1.	Photo	The photo model class represents a photo that authenticated users can upload to the website.	PhotoID	Integer
2.			Title	String
3.			PhotoFile	Binary
4.			Description	String
5.			CreatedDate	DateTime
6.			Owner	Integer
7.	Comment	The comment model class represents a comment that authenticated users can add to photos. This enables users to discuss others' photos. Each comment is associated with just one photo.	CommentID	Integer
8.			User	String
9.			Subject	String
10.			Body	String
11.			PhotoID	Integer

MVC Controllers

Developers will create the following controllers. For each controller, actions have been listed and descriptions given.

Table 2: MVC Controllers

	Controller	Action	Description
1.	PhotoController	DisplayGallery (GET)	The action runs when the user requests the Photo Gallery page. The action obtains all the photos from the database and passes them to the DisplayGallery view.
2.		DisplayRecent (GET)	This action is similar to the DisplayGallery action except that only the most recent photos are obtained from the database. This smaller collection of photos is passed to the DisplayGallery view.
3.		DisplayPhoto (GET)	This action runs when the user clicks a photo's “Details” link in a gallery. The action obtains full details of a single photo from the database and passes it to the DisplayPhoto view.
4.		AddPhoto (GET)	This action runs when the user clicks the “Add a Photo” link. The action creates a new instance of the Photo model class and passes it to the AddPhoto view.
5.		AddPhoto (POST)	This action runs when the user clicks “Save” in the AddPhoto view. The action saves the file and details of the new photo to the database and redirects the user to the DisplayGallery view.
6.		DeletePhoto (GET)	This action runs when the user clicks a “Delete this Photo” link in the DisplayPhoto view. The action displays the DeletePhoto view, which requests confirmation for the deletion.
7.		DeletePhoto (POST)	This action runs when the user clicks “Delete” in the DeletePhoto view. The action deletes the current photo, with its associated comments from the database and redirects the user to the DisplayGallery view.
8.	CommentController	DisplayComments (GET)	This action runs when the DisplayPhoto view is displayed. The action requires the current PhotoID as a parameter and uses it to get all the comments for the current photo from the database. The action returns the _DisplayComments partial view.
9.		AddComment	This action runs when the user clicks

		(GET)	the “Add a Comment” link in the DisplayPhoto view. The action creates a new instance of the Comment model class and sets its PhotoID to be the ID of the current photo. It passes this new comment to the AddComment view.
10.		AddComment (POST)	This action runs when the user clicks “Submit” in the AddComment view. The action saves the details of the new comment in the database and redirects the user to the DisplayPhoto view.

MVC Views

Developers will create the following views. Each view has been listed together with the controller it is associated with.

Table 3: MVC Views

	Controller	View	Description
1.	PhotoController	DisplayGallery	This view displays a collection of photos in the thumbnail size. For each photo the Title, Owner, and Created Date values are displayed.
2.		DisplayPhoto	This view displays a single photo in full size. The Title and Owner values appear above the photo. The Photo Name, Description, and other values appear beneath the photo. Under these details, all the comments for the current photo are listed with an “Add a Comment” link.
3.		AddPhoto	This view displays a form that the user can use to upload and describe a new photo.
4.		DeletePhoto	This view displays a form that the user can use to confirm the deletion of a photo. The view displays details of the current photo such as its title and description.
5.	CommentController	DisplayComments	This partial view, which is used on the DisplayPhoto

			form, displays all the comments associated with the current photo.
6.		AddComment	This view displays a form that the user can use to create a new comment for a photo.

Hosting Recommendations

Since the photo sharing application will be developed in ASP.NET Core MVC, it must be hosted on a Microsoft web server. The author recommends the following hosting configuration:

Web Server

The author recommends using Microsoft Azure to host the Photo Sharing application. Microsoft Azure can host any ASP.NET website, including the ASP.NET Core MVC application proposed in this document. Scaling is very simple because Microsoft, not Adventure Works, is responsible for adding server resources at times of high traffic. Costs are minimal: they depend on the amount of data served to visitors but it is not necessary to maintain our own hardware.

Database

The author recommends using SQL Database, within Microsoft Azure, to host the Photo Sharing application underlying database. As for the web server, this recommendation ensures high-availability hosting for the database with good value for money. This makes particular sense if the web site is hosted in Microsoft Azure.