Brian Baecher

CS-499

9/21/2025

1. *Briefly describe the artifact. What is it? When was it created?*

It's a bit difficult to speak in terms of discrete 'artifacts' when the project we've agreed to is a reinterpretation of an entire project into another language / stack. I think the best way to approach this right now is just to demonstrate the progress I've made thus far, and compare and contrast it to the original project from CS-465.

The chunk of the project I've completed thus far is what we'll treat as an 'artifact' - and is essentially the home/landing page of the 'Travlr Getaways' application, as well as the bit of progress I've made in recreating and expanding the MongoDB storing records, and the back-end API the front-end is communicating with. The original project was completed during May-August 2025 term for the CS-465 Fullstack Development class.

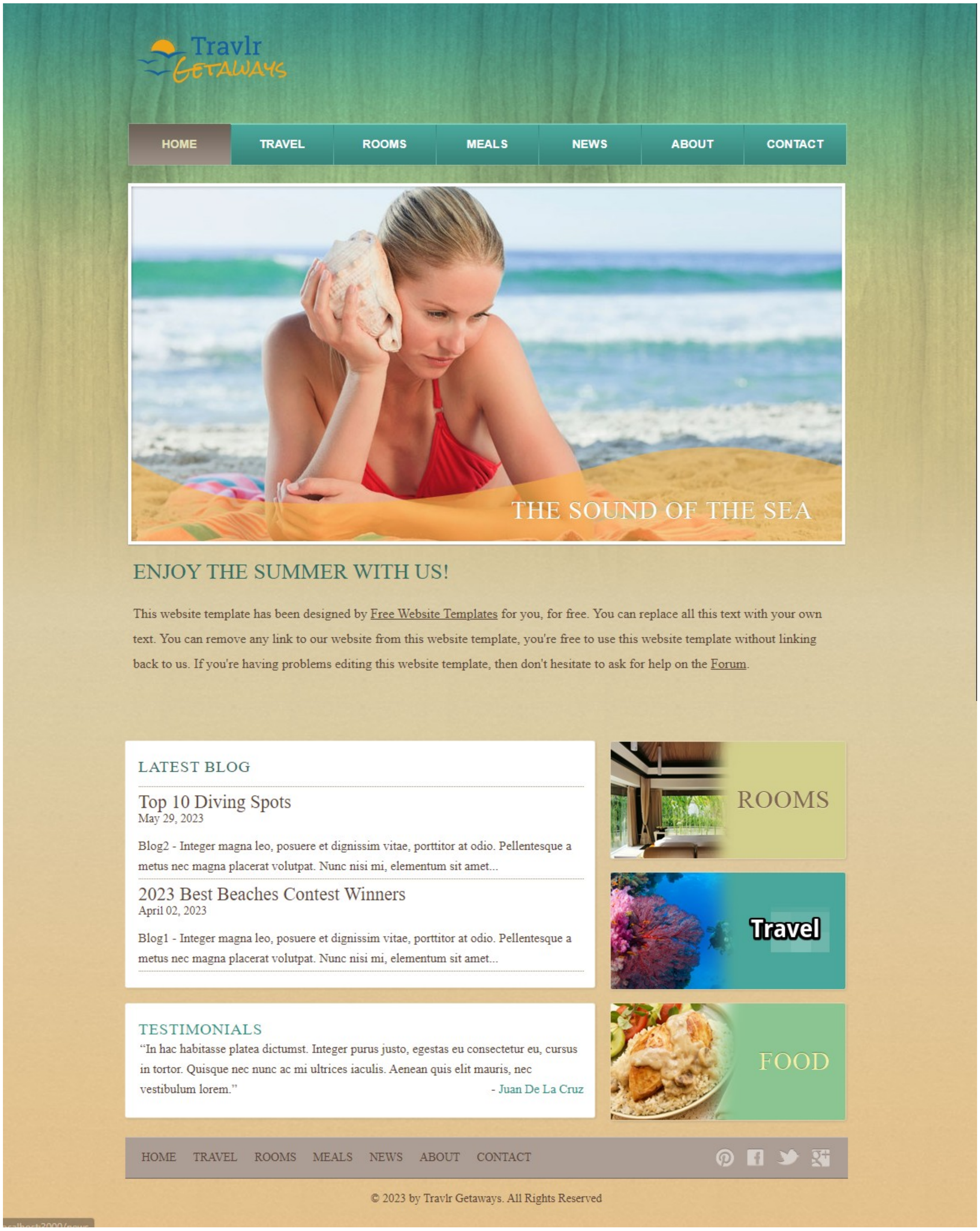
2. *Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? How was the artifact improved?*

I chose this project because I wanted to demonstrate my understanding of fullstack development, and by recreating the project with a different stack hope to show that I understand the fundamentals regardless of the particular language/stack.

The first and most notable improvement is the use of a modern web framework for the customer-facing portion of the site. It struck me as strange that the original project only used Angular for the administrative portion of the application, whereas the customer-facing site used Express to serve static HTML. So that's the first improvement, and to my mind has already brought the project closer to modern standards.

To show the differences at present, I'm including screenshots of the home page from the original project, and from the prototype/draft that is what I currently have. Bear in mind that the draft is just that, and that styling/formatting/content is not finalized. I've more been focused on laying the groundwork, setting up the component structure and the API/DB, so at least in terms of styling the current draft is bare-bones. I plan to implement functionality first and focus on aesthetics towards the end of the project.

Original:



ENJOY THE SUMMER WITH US!

This website template has been designed by Free Website Templates for you, for free. You can replace all this text with your own text. You can remove any link to our website from this website template, you're free to use this website template without linking back to us. If you're having problems editing this website template, then don't hesitate to ask for help on the Forum.

LATEST BLOG

Top 10 Diving Spots
May 29, 2023

Blog2 - Integer magna leo, posuere et dignissim vitae, porttitor at odio. Pellentesque a metus nec magna placerat volutpat. Nunc nisi mi, elementum sit amet...

2023 Best Beaches Contest Winners
April 02, 2023

Blog1 - Integer magna leo, posuere et dignissim vitae, porttitor at odio. Pellentesque a metus nec magna placerat volutpat. Nunc nisi mi, elementum sit amet...

TESTIMONIALS

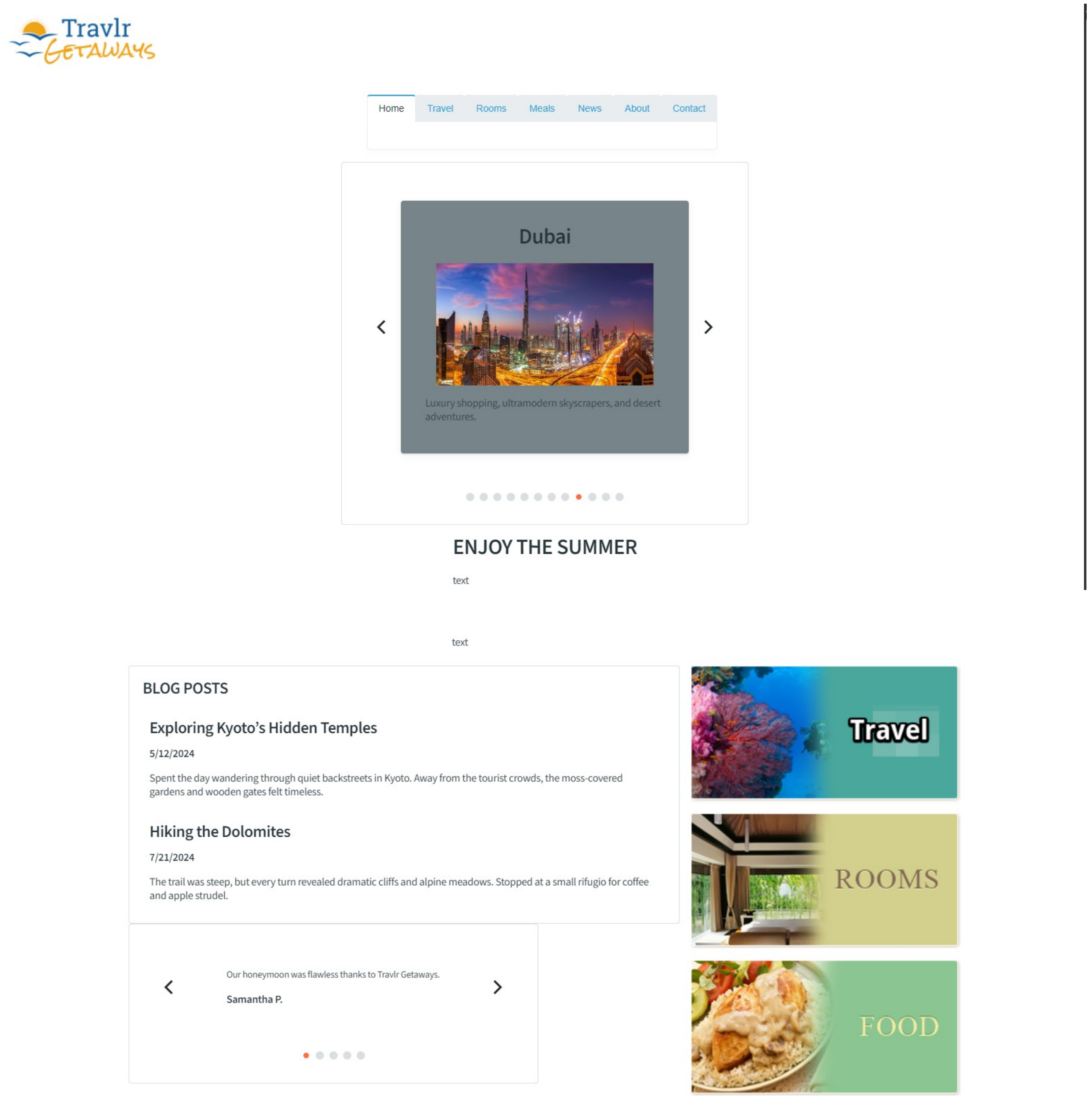
"In hac habitasse platea dictumst. Integer purus justo, egestas eu consectetur eu, cursus in tortor. Quisque nec nunc ac mi ultrices iaculis. Aenean quis elit mauris, nec vestibulum lorem."

- Juan De La Cruz

HOME TRAVEL ROOMS MEALS NEWS ABOUT CONTACT

© 2023 by Travlr Getaways. All Rights Reserved

Current Version:



Dubai

Luxury shopping, ultramodern skyscrapers, and desert adventures.

ENJOY THE SUMMER

text

text

BLOG POSTS

Exploring Kyoto's Hidden Temples

5/12/2024

Spent the day wandering through quiet backstreets in Kyoto. Away from the tourist crowds, the moss-covered gardens and wooden gates felt timeless.

Hiking the Dolomites

7/21/2024

The trail was steep, but every turn revealed dramatic cliffs and alpine meadows. Stopped at a small rifugio for coffee and apple strudel.

Our honeymoon was flawless thanks to Travlr Getaways.

Samantha P.

I'm using the Radzen component library/framework to assist in the development of the site's Blazor components. It is a useful resource that provides many base components which can be fairly easily extended, and it saves time from having to tinker with CSS rules (which is one part of web-development I've always disliked). Much of Radzen's strength lies in its components that are dedicated to data-visualization, which I have yet to implement in the project but hope to do so in coming weeks. One example of a Radzen component I've used to liven up the site is the Caurosel component. You'll see that the Home.razor page includes two of these components. One for the image banner at the top of the screen, and one for the testimonials in the bottom left (the same component is used twice, and is differentiated by the Type parameter on the component, the type supplied determines the carousel's content and behavior). The use of this component adds a bit of motion and interactivity to the page. If I have time, I may revisit the CarouselComponent.razor file to more effectively handle the two types it accepts as a valid Parameter. At the moment, it is given a Type parameter, and conditionally renders the caurosel's content according to it. It works, but there are two arrays in the class that both exist for every instance (Destination[] destinations, and Testimonial[] testimonials), when only one is needed for any particular Carousel. For the sake of time I've left the two arrays in there, but ideally would combine them into a single enumerable so that there are no wasted/unused variables.

Another improvement is the expansion of the database records, and the corresponding increase to the back-end's functionality. In the original project, the only records stored in MongoDB were the Destinations. I've added two additional collections to the database, one for blog posts, and one for customer testimonials. Accordingly I've also added API controllers and front-end services for the three existing collections so that the data can be used in Blazor components. This is an improvement, as the original site had things like 'blog posts' hard-coded into the HTML, whereas now our site is using database records to inform the content - meaning that site admins can add a new blog post, and it will appear in its component without any modifications needed. Same goes for testimonials. This is a way of expanding our use of the utility provided by databases to enrich the webpage.

3.Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

I think I've gotten a decent start. I'd much prefer to continue working on it rather than writing about it, as there's a long way to go just to reach the point where the original project has been fully recreated. I'm hoping that I reach that point fairly soon so I can focus on extending/adding functionality rather than just replicating.

4.Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

So far it's been a fairly smooth process. I am already fairly familiar with .NET/Blazor, and along with my general understanding of fullstack development I haven't run into any huge issues yet, but again this is a long ways off from being complete and I do anticipate challenges appearing, particularly in authentication and handling the administrative side of the application (the original project's admin side ran on an entirely separate port/URL from the customer-facing site, and I'm thinking that I'll combine them into one. This means I need to be more deliberate about user roles/auth in order to prevent unauthorized users from accessing the admin portion of the site).

One good bit of practice/learning in my efforts thus far have been in considering component structure, what portions of the site can be hard-coded into razor's HTML and what needs to be dynamic. Unfortunately I haven't noticed too many opportunities to have components be re-used across routes, as each one is fairly distinct. There are a couple opportunities I've identified, namely the Carousel component I described earlier, and the DestinationCard component, which I intend to use for the "Travel" route, where vacation packages are listed.

One area I have found a bit tricky is handling the models used by both the front and back ends. In order for the back end to work with our database, the class definitions in the API portion of the project need to include the MongoDB driver and make use of the annotations defining their corresponding fields. The front-end models are exactly the same object definitions, but do not need to include any bloat in the form of the Mongo driver, as the front-end never interacts directly with the database. I'm now in the position where each class used in the back-end has an identically named counter-part in the front-end. There are no conflicts, as the classes themselves are in separate namespaces, but just from a development perspective it is likely a bit confusing to have these classes share the exact names, and as I expand the classes I'll have to update its counterpart accordingly. I suppose all that matters is that serialization between objects sent from the front-end's services to the API is working properly, I could easily refactor the back-end's class names to include something like DTO at the end, just for the sake of clarity.

The most significant challenge I've faced is time constraints. Between working, other class assignments, and writing about this project instead of working on it leaves me with less time than I'd like.