Studying up on Study Abroad: Design and Development of MT TravelBlog

by
Rookery Sarah Baruch

A thesis presented to the Honors College of Middle Tennessee State University in partial fulfillment of the requirements for graduation from the University Honors College

Spring 2018

Studying up on Study Abroad: Design and Development of MT TravelBlog

by
Rookery Sarah Baruch

APPROVED:

_____
Dr. Medha Sarkar
Computer Science Department

_____
Dr. Chrisila Pettey
Computer Science Department

_____
Dr. Joshua Phillips
Computer Science Department

_____
Dr. Philip E. Phillips
Associate Dean
University Honors College

Department of Computer Science

Middle Tennessee State University; Murfreesboro, Tennessee, USA.

For my family away from home: thank you Angélina, Régis, Lilian, and Ewen LeGros for opening your home to me. My time in France remains one of my most cherished memories because of the people I met and laughed with in Caen.

Acknowledgements

I would like to thank Dr. Medha Sarkar for her support and guidance in the past year and a half. Thank you to the various MTSU departments and colleges that collaborated with me to make this website a reality: to Dean Bud Fischer and the College of Basic and Applied Sciences for supporting the maintenance of this website financially; to the Computer Science Department and the Office of Education Abroad for your help in administering the finished product; to the Information Technology Department for helping me sort out security issues; and to the Honors College for giving me the opportunity to complete this thesis.

I would like to thank the Institute of International Education for inspiring the idea for this thesis project, which began as a Follow-on Service Project upon my receipt of the Gilman Scholarship.

I extend a big thank you to my Creative Connections Software Engineering team. Thank you Elizabeth Rouze, Travis Hoover, Martavious Dorsey, Katrin Ghobrial, Roxanna Yavari, and John Westbrooks for helping to make MT TravelBlog mobile-friendly.

Finally, thank you to all the friends who have helped me hunt for bugs in my code and errors in my writing. I couldn't have done it without you.

Abstract

This thesis describes the process of the design and development of MT TravelBlog, including its purpose, its use and maintenance, and the technologies involved in its creation. MT TravelBlog is a web application envisioned and developed by Rookery Baruch and administered jointly by the MTSU Office of Education Abroad and the Computer Science Department. It provides MTSU students with a platform for sharing their study abroad experiences through blogging, posting of photos, commenting, and instant messaging. The benefit that this custom web application has over general-purpose blogging sites is that its membership is limited to MTSU, and it is designed with the needs of MTSU students and administration in mind. Users may make their content visible to anyone on the Internet or solely by other TravelBlog members. Content is limited to descriptions of study abroad programs that are available at MTSU, so any student who reads about a program on the TravelBlog site knows that this program is available to him or her. Commenting and instant messaging capabilities put in contact students who can help each other prepare for their trips abroad. Administrative oversight by the Office of Education Abroad allows its representatives to respond to students' needs and remove any inappropriate content. MT TravelBlog employs Microsoft ASP .NET framework, MVC design pattern, Twitter Bootstrap front-end framework, and SQL database. It is hosted via Microsoft Azure and maintained by the MTSU Computer Science Department. It becomes available for student use in summer 2018.

Table of Contents

List of Abbreviations

API – Application Program Interface

ASP – Active Server Pages

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

ITD – Information Technology Department

MTSU – Middle Tennessee State University

MVC – Model View Controller

OEA – Office of Education Abroad

SQL – Structured Query Language

Introduction

The summer after my sophomore year, I had the opportunity to study abroad in France. During my preparation for that trip, I discovered that while information on activities included in the program and museum admission prices was readily available on the program website, information on cellular data reliability and the availability of vegetarian dining options was nearly impossible to find. Fortunately, I knew someone who had completed the same program the previous summer, and I was able to run all of my questions by her. Even before settling on a program, I had consulted her and other friends who had studied abroad, allowing their experiences to shape my decision on what program would best suit me. I wondered how anyone would adequately plan without friends who had already studied abroad in the target country. I then had the idea to create a website that would put MTSU students in contact with each other for the purpose of sharing study abroad experiences.

Out of that idea grew MT TravelBlog, a web application that allows MTSU students who have studied abroad to blog about each of their trips. Those who have yet to study abroad can browse the blogs, looking for ones in their preferred locations or of their preferred length. Such students can benefit from the stories of others without necessarily knowing them outside of the TravelBlog. If students would like to ask specific questions of each other, they can do so by posting comments on blogs or by instant messaging other TravelBlog members directly. In order to ensure that all information shared on the website is relevant and useful, contributing permissions are restricted to MTSU students with accounts. This guarantees that all the programs described in the blogs are available to MTSU students and that any member who asks a

question of another member will necessarily be talking to an MTSU student. Begun as a Follow-on Service Project for the Benjamin A. Gilman International Scholarship, the project has been expanded and completed as an honors thesis.

Background

"Web development" is a very general umbrella term that can imply one developer designing one webpage with no interactive parts and using one markup language, or an entire team of developers designing a complex web application with multiple frameworks and a long list of libraries. Every web developer must consider the various moving parts that interact to animate a webpage or website in order to decide which technologies a project necessitates. Another important consideration is the capabilities that a website needs, dictated by its purpose. For instance, if one wishes only to create an informational page for an upcoming event, a single webpage with no interactive parts and no server-side code may suffice. Alternatively, one may build a website for a hair studio, listing pricing, personal biographies, and photos on separate pages, accessible by a common home page. This is an example of a more complicated project that still requires no server-side code. However, if one wants to do more than present information and adds features allowing users to interact with the website, it would qualify as a web application, requiring server-side code. An example of this would be the addition of a contact page to the hair studio website, allowing users to submit comments viewable by the hair studio employees.

Perhaps the most basic component of a web development project is the "front end," or the user interface. This includes everything on a website that a user sees, and it is composed of up to three components: layout, styling, and scripting. Layout is the only strictly necessary component of a front-end design, being a file or series of files that specifies the elements to appear on a webpage and the order in which they appear. Such files are written in a markup language, most commonly Hypertext Markup Language

(HTML). Markup languages are useful for displaying content, but they offer limited options for specifying how content should be displayed. For that, Cascading Style Sheets (CSS) is needed. CSS allows developers to apply styling to specific elements, like setting a background image or the size of a text area. The final component, scripting, is necessary if any part of the front-end design is dynamic, as with accordion sections, where clicking on a section causes it to expand without reloading the page. The most common front-end scripting language is JavaScript.

The term "front end" implies a contrast with the "backend," which encompasses all processes occurring and data stored behind the scenes. For example, when a website user submits a comment, this involves not only the text area and submit button formatted on the front end, but also the process defining what to do with the text when the submit button is clicked and a place to store the text long-term. Data can simply be stored in files, but database storage is preferable when data need to be easily searched and retrieved. Storing data in relational databases has the additional advantage of allowing like data to be stored together and related data to be found by association. Every time data is written to or requested of a relational database, it is done by means of a Structured Query Language (SQL) query. The technology in a web application that mediates between the front end and the data store could be any number of things, and it is selected based on the needs of the application and the preferences of the developer. Fundamentally, this component, often referred to as "server-side code," is composed of code in one or several programming languages, handling all the business logic that causes a web application to respond to user input in complex ways.

Depending on the complexity and number of tasks a web application is required to perform, a framework may be used to simplify and organize code. Different frameworks, many specific to the programming languages and technologies being used, exist to streamline the process of communicating between the front end and the backend. After choosing the framework, the developer must write code using the framework's tools to manipulate data and pass it back and forth from storage to the user interface.

A final consideration that plays into web application development is security. One of the most common security risks is unprotected input boxes on the user interface. An input box with no validation is an open target for SQL injection, an attack in which a SQL query is entered into a text field, in the hope that sensitive information will be returned by the database. To avoid such attacks, secure websites validate text entered into text fields before it ever reaches the database by putting constraints on the type of text that will be accepted. A phone number field, for instance, may only accept numeric text and have a maximum length of ten, and an email field may prohibit spaces and require that the text entered follow the standard email pattern of "username@emailprovider.xxx."

A second standard practice for securing a website against SQL injection involves account security. Account passwords are perhaps the most sensitive pieces of information stored by any web application, and they are difficult to validate because secure passwords follow no pattern and use a wide variety of characters. For this reason, passwords are never stored as-is. Instead, a cryptographic hash function is used to convert the password into a meaningless series of characters called a "hashed password," which is stored in the database. If a successful SQL injection attack were to return a list

of these hashed passwords, it would be extraordinarily difficult to reverse-engineer the original passwords, given that the cryptographic hash function is unknown and likely utilizes several variables, the values of which are equally unknown. Instead, hackers tend to use brute force programs that test common passwords, converted into hashed password by common hash functions, against the stolen list of hashed passwords.

To slow the brute force attack, many secure websites append unique, randomly generated character sequences, called “salts,” to each password before hashing it. A password’s salt is stored with the hashed password, and when a login attempt is made, the salt is retrieved and appended to the password entry attempt before it too is hashed. The two hashed passwords are compared to each other, and a match results in user login. The reason for this extra step is to ensure that hackers who manage to steal a list of hashed passwords cannot simply compare them to other lists of common passwords hashed with common hashed functions. The unique salts ensure that in order to find matches, hackers must potentially check against each password in the list every common password plus the current salt, hashed with every common hash function. Assuming strong user-created passwords and a strong hash function, the hacker is stalled enough by this trial-and-error process for the attack to be discovered and for notices to be sent to users to change their passwords. This method of password protection is referred to as "password hashing."

Methodology

Web Application Components

Core Components and Features

Stated briefly, the objective of MT TravelBlog is to provide MTSU students with a website to blog and communicate with each other about their study abroad experiences. From the outset, this goal necessitates several components, one of which is a database for storing blogs and user correspondences. Because the design of the website requires frequent retrieval of related information (e.g. a blog, information about its owning user, all of the blog's comments, and the related commenters' usernames to be displayed at once on a single page), a SQL relational database is the natural database choice. The design also requires that users be validated MTSU students to contribute to MT TravelBlog in any way, and it requires that only the owner of a blog or comment may make changes to it. Implementation of this requires an account system, by which a valid MTSU student may apply for membership and log into the system with a unique set of credentials upon approval. These individual user accounts must be secure, using a password hashing method to ensure that only password hashes are stored in the database.

That each user account must be approved to become active implies a website administrator and some system for allowing the administrator to exercise control over user accounts and website functioning. The TravelBlog allows this control by means of an administrative portal, accessible only to those logged in with administrative accounts. From this portal, administrators may approve or reject new members and new bloggers, change account types, and respond to user questions and inappropriate content. Users

who have been approved as members must also be supplied with a way to apply for blogging privileges, and there must be a form accessible by anyone to allow users to submit questions to administration.

The primary functioning of the website depends on the ability of users to write blogs and communicate with each other, which means the web application must include a blog system and a system for comment management. The blog system must support text, image, and video content; and it must store two versions of every blog simultaneously. A published version will be viewable by anyone with an account, and a saved version will be private to the owning user until he or she publishes it. The system must be capable of storing blog content with its formatting information, deciding which is which, and displaying both correctly. It must additionally include a user-friendly interface that allows blog owners to tinker with a blog's content and appearance without constant server calls, and to save when ready, calling the server only once. The accompanying comments system must allow comments to be owned by a user, to be posted in reply to a blog or another comment, and to exist in one of four states: normal, flagged, deleted, or removed. It must also include some way of notifying relevant parties when a comment has been altered or has received a reply. Finally, a multimedia content management system is required to store, retrieve, and enforce permissions on photo and video content uploaded by users.

Additional Features

Other TravelBlog features not integral to its core functioning include a user profile management system, and a series of informational home pages visible to all users,

whether or not they have accounts. The profile management system allows users to post limited, relevant information about themselves in order to aid members in the process of getting to know each other. Each member's profile page may be marked private, so that it is invisible to all non-members. By default, it auto-fills with the member's username, dates of joining and last activity, and membership type. If a member has blogs or photos, links to them will be displayed on his or her profile page. Optionally, members may also share biographical information, contact information, and information about their travel plans and languages spoken here.

Web Application Structure

Data Storage

As previously stated, all text data is stored in a relational SQL database. MT TravelBlog was designed database-first, meaning that the structure of the database was decided first, and all application code was written with that database structure in mind. This allowed me to exercise a great degree of control over how data is stored and related on the backend, and to organize it as intuitively as possible. Designing a relational database involves thinking about collections of related information in terms of entities, and then thinking about the relationships between those entities in terms of cardinalities. An entity, for example, may be a Person (capitalized as an entity name), where each Person has a unique ID assigned to it, a username, an email, and a hashed password. It may equally be a Blog, where each Blog has a unique ID, an owner, a list of content, and a list of comments. The first step toward designing a database is the formulation of a complete list of all entities involved in the web application concept. Then, each entity in the list must be assigned a collection of attributes, or properties that must be stored for each instance of the entity. ID, username, email, and hashed password are the attributes of the Person entity in the former example. A last step in the database design is to determine the nature of the relationships, or the cardinalities, between all of the entities. For example, each Blog has exactly one Person author, but each Person may author more than one Blog. This indicates a one-to-many cardinality, as opposed to a one-to-one (for example, one Person has one Profile) or a many-to-many (for example, visits to a

particular Country may be described by more than one Blog, and one Blog may describe visits to more than one Country).

Once all entities and relationships necessary to the TravelBlog web application were determined, the database was created with SQL code, with one table for each entity and each many-to-many relationship, and one table column per entity attribute. A database table is a storage location in which all instances of a certain entity or relationship type are stored together. For instance, user login information about all registered users may be stored in a table called Persons. Each user would have a row in the Persons table dedicated to him or her, and the columns of that row would store information about the user, such as his or her unique ID (called a "primary key"), username, email address, and hashed password. In order to link a Person instance with the Blog instances of which he or she is author, the Person instance's primary key must be stored as a "foreign key" in an author ID column of each of those Blog instances. The alternative—storing the IDs of each Blog instance authored by a Person as foreign keys in that Person instance's row—would not work, because the number of Blogs authored by each Person varies from zero to many. However, because each Blog necessarily has exactly one author, the relationship between rows of the Blogs table and rows of the Persons table can be stored unambiguously by requiring that each Blog instance have as an attribute the ID of the owning Person instance. The difficulty, demonstrated here, with recording relationships between table rows that may be related to many rows of a different table is the reason that many-to-many relationships require tables to themselves. In the earlier example of a many-to-many relationship, a Country instance was described as being potentially related to multiple Blogs instances, and vice versa. The way that this

sort of relationship is stored in a SQL database is by means of a Blog and Country relation table with two columns per row, one storing a Blog instance ID and the other storing a Country instance ID. This way, a Blog describing visits to four countries may be represented four separate times in the relation table, and a Country ID may appear in the table as many times as the country is mentioned in various Blogs.

The database servicing the web application was initially created locally and was later copied to Microsoft Azure using a free student account. Because most of the technologies on which the TravelBlog relies are developed by Microsoft, Azure Web Hosting provides a streamlined solution for hosting and managing the entire web application, including the database.

Communication between Application and Database

All communication with the database takes place via a custom Web Application Program Interface (API), in which every data retrieval or storage action required by the application has an accompanying API function. The TravelBlog's server-side code calls an API function, which queries the database and returns any relevant information. This way, the server-side code may communicate with the database while remaining ignorant of the database location. Because of this, if SQL code is entered into a textbox with no validation, it still will not cause sensitive values to be returned. If a user enters invalid information into a textbox, the worst-case scenario is that an error will be thrown.

Microsoft ASP .NET Framework and MVC Design Pattern

The .NET MVC Framework utilizes the MVC design pattern and supports C# server-side code. MVC stands for Model View Controller, a method of code organization by separation of concerns. All front-end code is contained within a series of Views, which in the .NET Framework are .cshtml files. These Views contain no business logic and exist only to allow users to interface with the backend. Information is displayed by combination of HTML and Razor syntax, which allows C# to be interspersed in blocks throughout an HTML document. Each independent webpage in the application has a dedicated View, and collections of related Views may share the same layouts. The application home pages, for example, are contained within the home layout, which has a navigation menu on the left, a header bar with branding at the top, and a footer with copyright information on the bottom. The individual pages (the home, Search Blogs, Find People, Find Funding, Help, Code of Conduct, and Become a Blogger pages) all share the home layout and are rendered in its center. In the case of user-created blogs, there is a blog layout that includes only the header bar and footer, within which all blog Views are rendered. There is a View file for viewing of published blogs by non-owners, and there is a View file for editing of blogs by their owners. Both Views are intended to be generic so that they can display any blog returned from the database. Other groups of Views with dedicated layout pages include those for the administrative portal pages and those for the user profile pages.

One of the advantages of the .NET MVC Framework is that Views can be reused. One View suffices to display all blogs because different information can be presented through the View depending on which blog is being requested. This information

package, referred to in MVC as a Model, is simply an instance of a C# class intended to store specific information long enough to pass it back and forth between a View and a piece of server-side code called a Controller. Inside the Controllers is where all business logic is performed and where communication with the API occurs. Each group of related Views has a Controller, which contains various actions responsible to do such things as instantiate Model instances, send Model instances to Views, receive Model instances from Views, manipulate information, call actions of other Controllers, and call API functions.

Twitter Bootstrap Front-end Framework

Much as .NET functions as a framework for the web application's backend, Twitter Bootstrap simplifies and homogenizes styling and scripting on the front end. Bootstrap is a free CSS and JavaScript library that makes available to developers a host of customizable styling templates like navigation bars, buttons, and drop-down menus. Many of these templates are used on MT TravelBlog, including the front-page image carousel, the navigation and branding bars at the tops of the site pages, and the side-panel menu in the administrative portal. Bootstrap also contains a grid-style content alignment system built with CSS Flexible Box Layout Module. The system detects the viewport size, or the width and height of the screen on which a webpage is viewed, and divides the total width into twelve columns. A developer can assign a width in number of columns to each HTML element on a webpage, so that a calendar in a sidebar may take up two columns on a webpage (roughly seventeen percent of the page width), and a text area may take up ten columns to its right (roughly eighty-three percent). Were the developer

to add a third image element and assign it a width of four columns, it would be pushed down below the two previous elements because together they consume one hundred percent of the page width.

The benefit of the Bootstrap grid system is twofold. Firstly, it provides an easily implementable alternative to the traditional element-sizing technique, which assigns each element an exact width and height in pixels. Developers using this technique can size elements perfectly to fit their current screen sizes, but when they attempt to view the same webpage on wider or taller monitors, they will find that its formatting is no longer optimal, meaning potentially that the webpage is not visually appealing or even that site navigation is impeded. Specifying element size in terms of percentages of the page can sometimes help resolve such formatting issues. For instance, imagine a webpage that contains a text area covering eighty percent of the page, with a menu covering the remaining twenty percent on the left. This setup works well for large or wide screens, on which twenty percent is a sizeable display area, but on smaller or narrower screens, twenty percent allows less real display room, and the menu may become too small to read.

The use of the Bootstrap grid system resolves the issue entirely, allowing developers to dictate that each element take up a different number of columns depending on the size of the screen on which the page is viewed. This means that the same webpage can appear drastically differently on a smartphone and on a projector screen because the code is designed to be screen-size responsive. For example, the homepage of MT TravelBlog appears on larger monitors to have the layout described in the previous example: a narrow menu on the left, and a wider text area filling the rest of the page. On

mobile, however, the left panel menu disappears, collapsing into a dropdown menu in the header bar, which is triggered to open by button-click.

This demonstrates the Bootstrap grid system's second advantage—the ease of developing for both desktop and mobile at once. Most mobile web browser users are familiar with the difficulties of using a phone to access a site that was not built to be compatible with small screens. Sometimes the requested page displays exactly as it would on a computer monitor, forcing users to zoom in all the way and scroll horizontally and vertically in order to read the page. In the case of certain websites, users are prohibited from entering the site using a mobile browser and are prompted instead to install a mobile application. Other websites present mobile users with the options to view a usually limited mobile-friendly site or the full desktop version that is often difficult to navigate. The use of Bootstrap simplifies the process of cross-platform development, leaving developers with only one set of code to maintain and giving users a streamlined experience regardless of the screen used to view a site. For this reason, Bootstrap is rapidly becoming an industry standard, and it is integrated with ASP .NET projects by default.

Account Registration and Login

Because the TravelBlog web application must allow contributors both to contribute and to have unique access to their contributions, a system of accounts is a necessary component of the website. Additionally, only MTSU students will be allowed to register new accounts, necessitating a process by which new users can be validated as MTSU students. Authentication will not be handled by the campus login system as

initially expected, so alternate credentials and verification are required. The TravelBlog's implementation of a user authentication system relies on a multistep process. First, a new user registers by supplying his or her full name, MTSU email address, and a new username and password to the online registration form. As long as the email address and the username are unique and the email is formatted properly, the submission is accepted. The user now has an account with a permissions type of "unapproved," meaning that his or her credentials will be accepted by the system, but he or she cannot yet contribute to the site by blogging or commenting. Next, the user validates his or her email address by following a link in an email sent automatically by the system. If the user attempts to log in after registering and before validating the registered email, he or she is redirected to a page reminding him or her to validate the email. Upon registration, a notice will also appear in the administrative portal, asking a site administrator in the Office of Education Abroad (OEA) to verify that a current MTSU student with the name and MTSU email address given does, in fact, exist. Once a user's registration has been approved by the OEA, that user becomes a full member with permissions to comment on blogs and apply for blogs of his or her own.

The basic infrastructure for the creation of individual accounts can be set up easily using ASP .NET by selecting certain settings at the time the new project is created. The code and local database generated automatically can be altered later. To create the TravelBlog project, I began by selecting individual user accounts as the account registration type, which equipped the project with code to accept new user information, hash the password, and add a new entry for the user in the database. I later changed the database location, altered the password hashing settings to add a unique salt before

applying the hash function, and added code to enforce uniqueness of new user email addresses and usernames. I also added code to allow for email validation, sending a confirmation email to the given address and updating a field in the user's database entry when his or her email has been validated. The built-in .NET account management system allows a registered user to log in and view or edit pages that are private to him or her, and it includes an account management page, on which I added an option for users to update their passwords and email addresses.

Permissions

The website allows access with four tiers of permissions. The first, guest permissions, refers to the parts of the website that are visible to anyone on the Internet, without being logged in to the site with an account. These parts include the home pages and any blog and user profile pages that have been marked public by their owners. Member permissions are given to any MTSU student who registers as a new user on the site and is approved by OEA administration. Members have guest permissions, and they can also view blogs and user profiles marked private by their owners, post comments on blogs, and apply for blogging permissions. Blogger permissions are given as a result of a member's application for a blog. A member applies to become a blogger by submitting a form containing information about a past study abroad trip, and once an OEA administrator has verified that he or she has been on such a trip, the member's account type is changed to blogger. A new blogger has member permissions and can also edit, save, and publish the blog for which he or she applied. If a blogger wants to blog about more than one study abroad trip, he or she must apply for a second blog. Therefore,

every member who has at least one blog is a blogger, but a blogger may have more than one blog. No blogger can author more than one blog for the same trip or a blog that is unrelated to a study abroad trip. Administrator permissions include all member permissions and access to the administrative portal, in which administrators can approve new members and blog requests, respond to user help requests, control account types, and respond to flagged comments.

Home Pages

The website's home pages are the collection of informational pages that a user encounters before logging in. The home page, itself, is a webpage with an image carousel displaying photos from recently added blogs, and below it a brief description of the site. Users can navigate between this and the other home pages by clicking the options in the navigation menu to the left side of the page. The other pages include a blog search page, a member search page, a page on which members can apply to become bloggers, a page containing the site's code of conduct, a help page, and a page on which scholarships and financial aid for study abroad are listed. The Search Blogs page allows users to scroll through the full list of existing blogs or filter the list by author name, blog name, program type, program length, and country visited. Entering text into the author name field begins a search of the list of usernames, first names, last names, and full names. Every matched blog record appears in the result list, and even though bloggers' true names are searched, only their usernames appear in the results list. Furthermore, if a user who is not logged in uses this search feature, only blogs marked public will appear in the results list. Only a member who is logged in has access to all blogs on the site.

The Find People page provides a similar search feature, allowing users to search site members by their usernames, first names, last names, or full names. Again, true names are protected, and members with profiles marked private do not appear in results lists for members who are not logged in. The Become a Blogger page contains a form that can be submitted by members looking to blog about a study abroad trip. The form asks for specific information about the trip, like program type and countries visited, that can be confirmed or refuted by OEA records. If someone who is not logged in attempts to navigate to the Become a Blogger page, he or she is redirected to the login page. The Code of Conduct page outlines the rules for usage of the MT TravelBlog website, developed by OEA administrators and read by all new members when they join the site. The Help page contains a frequently asked questions section and a form through which users can submit specific questions to administrators. Lastly, the Find Funding page lists study abroad funding opportunities, which will be kept up to date by MTSU Global Ambassadors. Users can return to the home pages from anywhere on the site by clicking the TravelBlog branding in the header bar.

Blog System

Blogs are stored with their formatting as plaintext in the database, and they are displayed as multipage documents with text, photos, and videos of supported types. Blogs are public by default, meaning they are visible to anyone on the Internet, but they can be marked private by their authors, so that they will not be visible to anyone who is not currently logged in. Photos and videos featured on public blogs are public by extension, and media featured on private blogs is private by extension. A blog's owner

may choose to edit it from their account management page, at which point he or she will be taken to a blog editing view. In this view, bloggers can add, delete, and alter text blocks, photos, and videos. They can also toggle whether the blog is public or private and whether comments are enabled or disabled. When a blogger is done editing a blog, the blogger may choose to save the blog, which records the changes privately, or to publish the blog, which makes the changes public to other TravelBlog members. Because of this, two versions of each blog are stored at all times—the saved version and the public version, though at times the two may be identical.

Comments System

Comments may be posted by TravelBlog members on any blog for which comments have been enabled by the author. They display newest to oldest at the bottom of the associated blog. If a blog is public, its comments are public as well, and the same is true of a private blog. A comment may be posted in reply to the blog or in reply to another comment on the blog. Along with a comment's text, its author, timestamp, flagged status, and removal status is stored. A comment may be flagged as inappropriate by any user, and, once flagged, a comment will be reviewed by an administrator. Administrators may remove any comment and unflag comments that have been flagged. The owner of a comment may also remove that comment, and a blog author may remove any comment that appears on his or her blog. Comments are never fully removed from the database because they may accrue replies in the time that they are visible. Removed comments simply display as "removed" in a blog's comment list, so that all replies remain visible.

Administrative Portal

The administrative portal is a collection of pages visible only to users logged in with administrator accounts. It is accessible from administrators' account management pages, and it operates much like the home pages, with a navigation menu on the left side allowing users to switch back and forth between pages. On the first page, administrators can see a list of newly registered members and their registration details. After checking the new users' registration details to ascertain if they are MTSU students, administrators can click to confirm or reject each. New users who have been confirmed by an administrator gain the ability to comment and apply for blogs. New users who have been rejected are sent emails explaining their rejection and are removed from the database.

The second portal page allows administrators to confirm or reject member blog applications. Much like the member confirmation process, confirmation of a blog application requires that administrators check OEA records to make sure that a member has been on the study abroad trip that he or she describes in the application and that the member does not yet have a blog devoted to that trip. If a blog application is rejected, the member is sent a message explaining why. If the application is approved, the member gains access to a blank blog which can be edited, saved, and published. When a member's first blog application is approved, that member's account type changes from "member" to "blogger."

The third and fourth pages list flagged comments and help request messages for administrative review. Administrators can view flagged comments, decide if they are in fact inappropriate, and remove or unflag them. Each help request, sent from the help

page of the website, includes message text, a timestamp, and a reply email address. Administrators can reply to each message and mark it resolved. On the last portal page is a list of TravelBlog site members, which can be filtered in the same way as members on the Find People home page. From this page, administrators can change the account types of non-administrators to make them administrators, or they may "pause" a user's account. When a user's account is paused, he or she is temporarily prevented from posting comments, and if he or she has blogs, those blogs are temporarily removed from the site. This feature is included to give administrators recourse in the case of a member who continually posts inappropriate comments, or in the case that a blogger has included inappropriate content in a blog. An administrator can pause a user's account and send him or her a message stating that the account will be reinstated when the user reviews the TravelBlog code of conduct and agrees to adhere to it, or, when applicable, when the user removes the offending content.

Profile System

Each registered user has a profile page displaying basic information about him or her, including his or her username, account type, date joined, date of last activity, and any blogs he or she owns and has favorited. Optionally, each user may select a profile picture to be displayed on his or her profile, on each comment posted by the user, and on each blog he or she authors. An optional biography section displays information about a user's academic pursuits, class standing, and interests. Sections on travel and language allow users to indicate to each other what languages they speak, which areas of the world they have visited, and which areas of the world they would like to visit. A photo section

allows users to organize travel photos into albums and share them with each other, if they choose. Each user can decide how much information to share on his or her profile page, and each can choose whether to make his or her profile page public or private. As with blogs, private profile pages will only be accessible to members who are logged in. An attempt to access a private profile page when not logged in results in redirection to the login page.

Photo System

Each photo uploaded to MT TravelBlog has exactly one owner and may be uploaded directly to a user's profile or to a blog. Each photo uploaded by a user is added to his or her All Photos album, which is created automatically and is private to each user. From there, it can be added to other user-created albums, which can be marked public or private. When a user's blog application is approved, a photo album is created for it in the user's profile, and this album assumes the same public or private status as the user gives to the blog. Neither the All Photos nor blog-specific albums may be deleted by their owning users, but photos within them may be deleted. Photos deleted from blog-specific albums are inaccessible when the blog is displayed and are replaced by a message stating that the image was not found. Photos deleted from the All Photos album are deleted everywhere that they exist on the site.

Challenges and Results

As with all software projects, challenges arose during the creation of MT TravelBlog, not the least of which was the uncertainty about how the website would be hosted and administered. The thesis proposal from Spring 2017 describes the plan as it originally stood. The MT TravelBlog website would be integrated as a part of the MTSU website, considered an extension of the OEA website, and administered jointly by the Information Technology Department (ITD) and the Office of Education Abroad. There would be a team of computer science students, much like the MT Mobile App team, who maintain the site and keep it up to date. The team and the site itself would be funded jointly by contributions from the College of Basic and Applied Sciences and the Information Technology Department. Unfortunately, the plan needed to be reworked in Fall 2017, when the ITD discovered that the Terra Dotta StudioAbroad web product currently in use by the OEA supports limited options for blogging already. The feature is not enabled, but it could be enabled through correspondence with Terra Dotta. When this alternative was discovered, ITD rescinded its offer of funding and administrative support for reasons of contract and convenience.

At that point, Dr. Sarkar and I conferred with the Office of Education Abroad to discuss the benefits of both blogging options—my separate solution and the integrated solution provided by Terra Dotta. We determined that though the Terra Dotta solution allowed students to log in with their existing MTSU credentials, MT TravelBlog would be more customizable, more user-friendly for MTSU students and administrators, and more likely to attract student users. For that reason, my mentor and I decided to proceed with the project, and we devised a new plan to host and maintain MT TravelBlog with the

OEA. There would no longer be a dedicated student maintenance team, but the Computer Science Department would help to host the website and maintain it technically. The OEA would act as administrators, overseeing site usage. Because the TravelBlog would no longer be administered by the ITD, it would not have access to MTSU's credential verification system. To ensure that the website remains MTSU student-exclusive, the multistep registration plan, in which students register and are then approved or rejected by the OEA, was developed. To simplify that process and all oversight for the OEA, I added the creation of an administrative portal to my agenda. Finally, the updating of site information, like study abroad funding and front page carousel photos, would be delegated to student Global Ambassadors, supervised by the OEA. MT TravelBlog creation has continued under the current plan, and we are preparing for an official launch in Summer 2018.

Appendix

Glossary of Terms and Definitions

Authentication

Verification of user identity, usually by acceptance of a username and password pair, and comparison of this pair to a set kept secret by an application.

Backend

The collection of application processes occurring out of sight of the user, without ever being directly accessed by the user.

Business logic

The server-side code describing the rules and processes that allow a web application to serve specific purposes.

C#

An object-oriented programming language developed by Microsoft to be used in its ASP .NET web development framework.

Cardinality

In a relational database, cardinality describes the nature of the relationship between two entities. In a library database, the relationship between the Author and Book entities would have a cardinality of many-to-many, because an author may publish more than one book, and a book may be written by more than one author.

Controller

In the MVC design pattern, a Controller is a class containing actions that control the functioning of a specific portion of an application. It is here that one finds application business logic, allowing data to be sent to storage, manipulated, or sent to the front end.

Controller action

In the ASP .NET MVC Framework, a Controller action is a function containing code to perform a specific task. User registration may be handled by one action of a user accounts Controller, while user login is handled by another.

Foreign key

In a relational database, a foreign key is the unique ID of an instance of Table 1, referenced by an instance of Table 2, in order to relate the two. In a library database, a Checkouts table may record the unique IDs of Patrons beside the unique IDs of Books as foreign keys, indicating who has checked out which books.

Framework

An intentionally generic software on which developers built complex applications tailored to specific purposes.

Front-end

The user interface of a web application, including all graphics and functionality accessed directly by users.

Markup language

A language that allows formatting and styling of plaintext in a text file.

Model

In the MVC design pattern, a Model is a generic template, of which instances serve as data packages to be passed back and forth from storage to a user interface. In the ASP .NET MVC framework, Models are C# classes.

MVC Design Pattern

A software design pattern that separates code into three interrelated parts: Models, Views, and Controllers. The intention is to enforce the writing of clean and organized code by separating code that dictates what users see from code that manipulates data.

Password hashing

A method of securing user accounts by applying a cryptographic hash function to user-created passwords and storing the hashed passwords. Password attempts entered at login are hashed and compared to the expected hash password value. This method renders unnecessary storage of plain string passwords.

Primary key

The unique ID of each entry in a relational database table.

Razor syntax

An ASP .NET syntax allowing for the creation of dynamic webpages by embedding server code in front-end pages.

Relational database

A type of database in which similar data is stored together in dedicated tables, and in which the data rows in each table may be related to data rows in other tables by means of foreign key references.

Salt

In password hashing, a salt is a randomly generated string appended to a user-created password before the cryptographic hash function is applied. Salts are stored with the passwords to which they have been applied.

Separation of concerns

An idea in software engineering that code should be organized in a manner that allows different sections to be specialized for specific purposes. Popularly, this is interpreted to mean that one portion of the code is responsible to present information to users while another is responsible to manipulate data. These portions communicate with each other but do not handle each other's responsibilities.

Server-side code

In a web application, server-side code is all code requiring a server call, as opposed to client-side code, which is executed in-browser.

SQL injection

An attack in which SQL code is entered into unprotected user information fields on a webpage, in the hope that the code will cause sensitive data to be returned by the database.

Validation

Enforcement of requirements on a webpage's user information fields, ensuring that data entered is valid according to what the application expects.

View

In the MVC design pattern, a View is a page that can receive data from a Controller action and present the data to a user or receive data from the user and send it to a Controller action.

Web application

A program involving both client-side code and server-side code, in which users interact with an in-browser user interface and business logic is handled on a remote server.

Web development

The process of developing web applications and websites.