ArtHub - 13

Erin Scheunemann

Brian Garret Keith

River Kelly

https://csci331.cs.montana.edu/~b62v473/ArtHub/

https://github.com/RK-MSU/CSCI-331-Final-Project

Creative Objective

We created ArtHub as a place for artists who want to start accepting commissions. We recognized that current websites made for artists had limited functionality for commissions and that there were no websites designated just for that purpose. For example, Esty, a website made for people to sell handmade items, only has functionality that allows artists to sell already completed works of art. For artists who want to do commission work, create custom art for people by request, they have to modify the normal Etsy listing to suit this purpose and that limits them. The price and expected delivery date may be fixed after a customer places an order so there is no way for the artist and client to negotiate a price and time frame that works for both of them. Our goal with our website was to create a space for artists to share their artwork and to take commission requests. Artists can post their artwork to gain the attention of the potential client base. Our website also adds commission functionality where clients can request artwork from artists, artists can manage their commissions moving them from pending to accepted and finally to finish, and clients can view their requests and statuses. Along with that artists and clients can communicate either through the email system or the direct messaging system.

Tech summary

As seniors in Computer Science, one of the most valuable lessons learned is to never reinvent the wheel. The primary resource we used in our project was an open-source content management system called ExpressionEngine. Using this PHP backend provided us with the bare-bones essentials necessary for any website. Out of the box, ExpressionEngine provides member authentication, role-based permissions, a flexible tool for creating any data model, and HTML templates to render content into customizable web pages.

The key feature in ExpressionEngine is called Channels, this is what allows you to create custom data models. Channels are described by the associated fields. Fields may be either text, select options, or files to name a few. For example, one of the channels on our site is “posts”. The post channel represents the blog posts for the artist members within our website. The post channel has a title, date, description, and image field. Once a channel and its fields are created, Channel Entries are generated. This is what makes up the content. Using the HTML templates, channel entries are then parsed and rendered to make up a web page. Within our site, we have specific templates designated for each of the relevant content. There are templates designated to render the artist’s profile, blog posts, comments, and commissions.

ExpressionEngine is modeled after the principles of container dependency injection. This is intended to maintain the separation of individual modules within the system. Doing so also allows developers to easily customize system functionality, known as Addons. If a feature does not already exist, an addon can be created to implement the desired functionality. This ability was highly desirable in the development of our site. Some of the features that were not already available in ExpressionEngine that were added in our custom add-on were; support for private member messaging and uploading files on the front end of the website.

The private messaging module was created to allow members to send messages to one another. This meant that the addon was responsible for maintaining its own database tables and also had the ability to deliver data to the front end via parsable template tags. To implement file uploading on the front end, the add-on needed to maintain consistency with the existing system. This meant that the add-on needed to follow the database schema and file system behaviors so that newly uploaded files would be recognized by the system.

Another resource used to allow file uploading was AngularJS. This is a JavaScript library that uses a model view controller implementation. AngularJS was used to manage the process of events necessary for a user to select a file from the system. For example, the post channel has a file image field. If a user wishes to add an image to this field they have two options; select an existing image from the file system, or upload a new file. Both of the processes were supported by AngularJS.

Another JavaScript library we used was jQueryUI Datepicker. For field types that needed a formatted date input, this library provided a clean and easy-to-used popup date picker. When a user clicks on a date field input, a popup window appears allowing the user to select a date from a calendar-based prompt.

To style our site, we used Bootstrap as a foundation with some custom CSS. Using Bootstrap allowed us to have a flexible groundwork to create a grid-based layout design. Another resource that assisted in styling our website was Less CSS. This was acquired using Node Package Manager and GulpJs. Less CSS is a dynamic preprocessor to the CSS style sheet language. It was designed to get rid of the redundancy commonly found in CSS style sheets.

River Kelly

As for my work on the website, I was responsible for creating the third-party PHP addon in the backend, developing HTML templates related to Artist posts and comments, and maintaining Github files. HTML templates are accessible via the Control Panel user interface on the back-end, but since the addon is written in PHP and is located in the core of the filesystem, after any small change I would have had to reupload my progress. To avoid this, and to avoid potentially crashing the site, I installed a local version of the website to demo and test. As for developing the HTML templates for the Artist's blog, profile, and comments, I was able to do this via the Control Panel Template Manager.

One of the main items that I took away from this project was the process of developing a website as a team. In the past, I did not have to worry about crashing a website due to bad code. Which I did a few times during this project. After the first couple of mistakes, I became very strict on whether or not I should upload my code. I made sure, and double-checked, before uploading new code to the backend. The last thing I wanted was to delay my teammates because of a careless mistake I made.

If I were to do this project over again, the main thing I would do differently would be to better separate the backend addon. Typically in developing code, it is ideal to loosely couple modules. This is key because when classes or other items are tightly bound, small changes can make devastating bugs. The addon makes for this website to support file management and private messaging between users. The ideal way to accomplish this would have been to separate the two distinctly different functionalities into separate modules. This would make the code cleaner and easier to digest.

Erin Scheunemann

I was incharge of the commission and request functionality for the website. This involved handling things on both the artist's and the client's side. On the artist's side I had to work out how artists could manage their commissions and update the client of any changes or progress they have made. On the client's side I had to figure out how a client could request artwork, check the status of these requests. For both I had to figure out a way for them to communicate. The first thing I did was create the commission request form that clients fill out. To do this I used the channel form functionality that ExpressionEngine provides. These entries are connected to the commission channel. The client fills out an expected delivery date, price range, and a description of what they want. When the client submits the form the status is automatically changed to 'Pending' and the entry is linked to the artist they are commissioning. I initially had trouble figuring out how to link the artist to the commission since the author of the entry was automatically set to be the client. To fix this I used a hidden html field to change the author to be the artist.

My next task was figuring out how to let the artist manage their commissions. Since the commission entry was linked to the artist through the author attribute, I could list all the commission requests they had in three separate sections: pending, accepted, and finished. The artist can then go to the commission's page and manage them based on their status. For pending commissions the artist can either accept or deny the commission and then it takes them to a page where they email the client with why they accepted or denied the commission. Initially I tried doing the emailing through PHP I put directly on the webpage but I found out that ExpressionEngine has an email add-on and so I used that to create the email page. For accepted commissions, artists can update the client using the same email functionality, add finalized prices and dates, and mark commissions as finished. For finished commissions there's a way to put that commission back in the accepted category in case the artist accidentally marks an unfinished commission as finished.

My final task I worked on was the functionality for clients to view and manage their requests. They can view their requests page which lists all past and present commission requests along with their statuses. This part was difficult because when I linked the author to the commission entry instead of the client I removed any connection to the client's account. In order to fix this I had to add a field to the commission request form that required clients to input their screen name. I then used this input on the requests page to loop through the commission entries and only display the ones whose screen name input matched the currently logged in user's screen name.

While the learning curve for ExpressionEngine was steep since I had never used anything like it, it allowed us to add a lot of different functionality to our website. I had never created a website before so the planning process was new to me. Our team decided to just dive in: we created a list of things we needed to do and started right away. If I were to do this again with another team I would have put more thought into the original planning of our website specifically relating to functionality. Our original list was missing a lot of the functionality that we ended up with so it made integrating different parts of the website harder since we were adding functionality that wasn't considered when developing previous parts. Another thing that I would have done differently was research email APIs earlier on in the process. ExpressionEngine's email add-on works but it takes a while to send emails and may send them to the recipients spam folder since the emails aren't sent directly through one of the mainstream email services. Overall I think what we have been able to add to the website with the 6 weeks we’ve had is impressive and the process has been smooth so far.

Garrett Keith

I was primarily in charge of UI and UX. When creating a website these are always a very important aspect, it can have all the functionality in the world but if no one can figure out how to use it it’s a bit useless. The primary focus for the website was the artwork and the artists, we wanted it to be a place where they could present their artwork and themselves, as well as find new clients for commissions. One of the challenges was a balance between displaying the art well, and displaying the information attached to each artwork. If it’s just the artwork it looks great but users cannot learn about and connect to the artists as well. If there’s too much text it gets too busy and the focus shifts off the artwork. We decided to fix this by having a couple different layouts for the website posts. The home page displays an image dominated grid, with only small descriptions for each piece, allowing for casual browsing and the ability to look through a high volume of posts in a short amount of time. Under the posts sections the layout is more balanced, showing the posts in a list format and displaying much more of the description. It also has advanced search features where users can narrow down their search based on medium or keywords. This allows users to get more detailed information on a more specific number of posts if they already have a pretty good idea of what they are looking for.

We used a similar strategy for each artist's personal profile page. On their homepage they can choose to pin posts of work they especially like or they think represents them well, this way anytime someone visits their profile they will immediately see a small showcase of the artist's best work in addition to the artist bio and a bit of info. There is also a button that says show all posts, which allows a user to browse the artist’s entire portfolio, from most recent to least. I believe this is an effective way to combat the problems mentioned above. The rest of the website

UI is pretty straightforward, with a navbar and some dropdowns in the header, and some links in the footer. All together I think it makes for a simple, intuitive, yet powerful website.

Conclusion

This project pushed us to learn and spend time working on skills that we can use in our professional lives. In this project we used technology that was new to most of us like ExpressionEngine.We also had to learn how to properly plan and develop a website. The learning curve for ExpressionEngine set us back in our timeline but also excelled us forward, allowing us to add more functionality than thought possible to our website. Our team went with an informal method of planning; listing our objectives and checking them off when they got done with very loose deadlines. This made the development process a little bit confusing and hard to keep track of. In the future, if our team were to work together again, we would put more thought into our planning process. This project overall was a pleasant experience and we all have come away with something we can be proud to put on our resumes.

References

“Datepicker.” *jQuery UI*, https://jqueryui.com/datepicker/. Accessed 2 December 2021.

“ExpressionEngine Docs.” *Welcome! — ExpressionEngine 6.1.6 Documentation*, https://docs.expressionengine.com/latest/index.html. Accessed 29 November 2021.

“Introduction · Bootstrap v5.1.” *Bootstrap*, https://getbootstrap.com/docs/5.1/getting-started/introduction/. Accessed 29 November 2021.