# Team Ludus

# Michigan State University

# Digital Playbill Builder

# Project Plan

# Spring 2024


**Ludus Contacts**

Ben St. John

**Michigan State Capstone Team**

Yufan Ai

Joe Davis

Swetha Jagannathan

Alayna Johnson

Courtney Thang

# Table of Contents

## Executive Summary

In 2016, Kevin Schneider, a theater director, and Zachary Collins, who was then a student, embarked on a side project aimed at simplifying online ticket sales for Kevin's shows. After successfully developing the software, Kevin noticed a remarkable turnout for one of his shows using the platform. The success garnered the attention from other directors who expressed interest in using the innovative product for their own productions. The growing demand from fellow directors led Kevin and Zachary to recognize a significant gap in the performing arts industry: a need for a product like theirs. Consequently, they launched what is now known as Ludus. In 2019, after a couple years of growing the business, Ludus opened their first office in Holland, Michigan. They now service over 2,000 organizations across the country, earning their spot in the 2023 Inc. 5000 Fastest Growing Companies in America list.

The mission at Ludus is to "build tools that empower everyone involved within the arts to focus on what they do best: sing, dance, play, act, direct, and perform". Although Ludus started as a ticketing platform, it has expanded to include other services such as marketing, fundraising and registration, helping organizations ranging from K-12 schools, colleges, community theaters and performing arts centers of all sizes. Their impressive growth is proof that they honor their mission by continuing to introduce revolutionary technology to accommodate the performing arts industry. Hence, Ludus' latest initiative aims to introduce the Digital Playbill Builder.

In order to provide an innovative and convenient way for performing arts organizations to develop and distribute information about specific events they are hosting, Ludus brings the Digital Playbill Builder. This web application will consolidate multiple steps in the process of creating programs and playbills by being a one stop shop for development, information collection and storage, and event information distribution via the web. Not only will this web application impact performing arts organizations, but also the audiences attending the events will now have a completely new experience when coming to these shows.

## Functional Specification

Paper playbills and programs have traditionally been the go-to way for displaying information for all sorts of events since the 19th century. Utilizing the advancement of modern technology, Ludus aims to introduce a completely new method of playbills and programs to the performing arts industry. It will redefine how performing arts organizations showcase information about their performances and productions, creating a digitally immersive experience for the audience. These organizations can utilize this web application to design, host, and share digital playbills for all of their events. The application will enable users to streamline the creation of playbills, implement a new way to present cast lists, advertise for other shows or partnered companies, and facilitate donations for the organizations themselves.

The organizations will be able to create and design custom playbills in the drag-and-drop builder that allows for upload of custom media, designs, advertisements, and even videos and gif animation. The user-friendly builder ensures simplicity without compromising versatility. The web application also allows for subdomain customization which allows the organizations to further make their playbills unique to their organization. To alleviate the need for organizations to manually collect and incorporate cast information, the builder allows organizations to send a form for each of their cast members to fill out with desired portraits and biographies to include in the playbill. Though the purpose of the Digital Playbill Builder is to create an easy way for organizations to create and distribute their playbills digitally, there is still the ability to develop a paper alternative that can be printed via the paper playbill builder.

Once the organizations publish their finished playbills to the public, performance attendees are able to scan a QR code or type in a URL where their performance's playbill will reside for them to view. The convenience of having the digital experience allows attendees to immediately access all of the information of the performance on their devices without having to flip through multiple pages of a program. These digital playbills also have the capacity to provide animated content and interactive information that will bring a better experience for the audience while waiting for their show to begin. All-in-all, a once static, paper playbill/program, can now be a fun, convenient, and unique experience for attendees before the performance starts.

# Design Specifications

## Overview

Ludus' Digital Playbill Builder is a flexible web-based application used by arts organizations to design, host, and share a digital playbill with their audiences. The builder will be accessible via a user's web browser and provide a simple drag-and-drop interface for designers to create a dynamic playbill for their production.

## Account Information

When a user opens the web application, they are prompted to create a new account or log into an existing one. Account creation is free, and users who have existing Ludus accounts can link their Ludus account with the builder. Additionally, when creating an account, a user will create a custom subdomain for their projects. A user can have one of three roles: collaborator, admin for an organization, or owner of an organization.

## Home Page/Dashboard

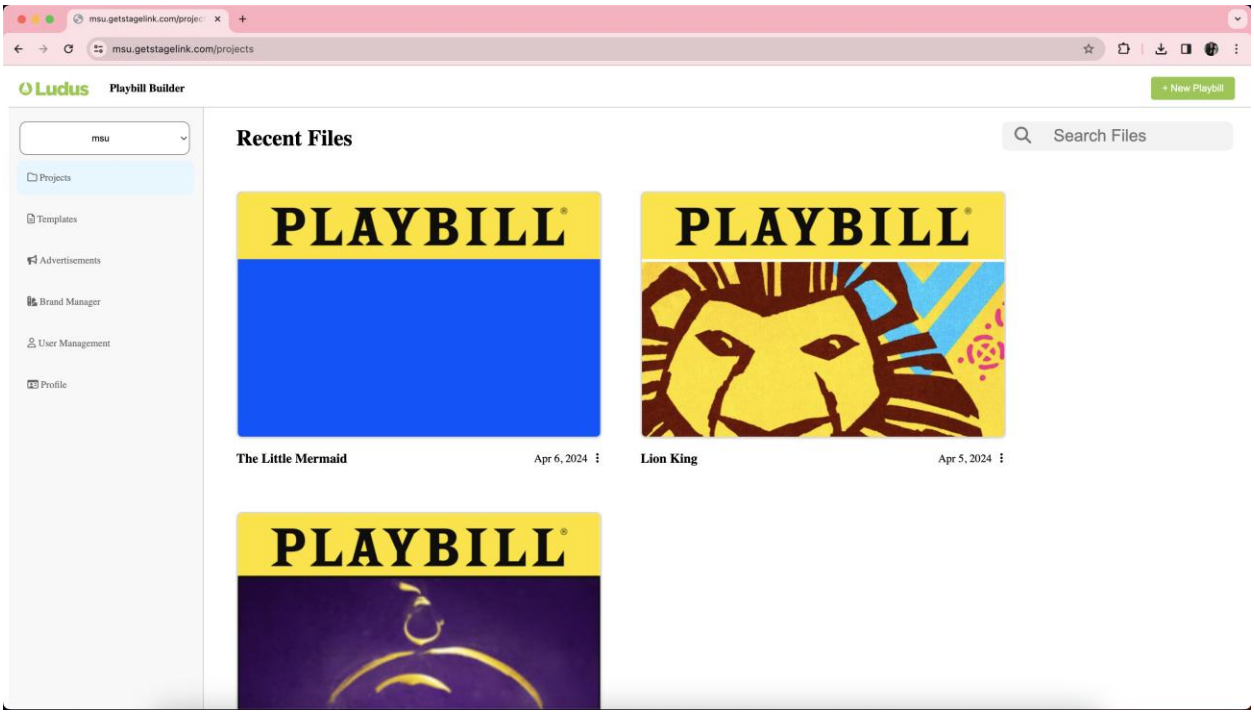


**Figure 1:** The projects tab of the user's dashboard

Once the user has logged in, they are greeted by their dashboard. Within the side navigation bar, the user is able to select projects, advertisements, brand manager, user management and profile, as shown in Figure 1.

Figure 1 also displays a user's homepage when in the projects tab. Projects on the dashboard are sorted by most recently opened and provide the user a thumbnail image of their project as it currently looks. Users are also able to create a new project with the button in the top right corner or clone a pre-existing project.

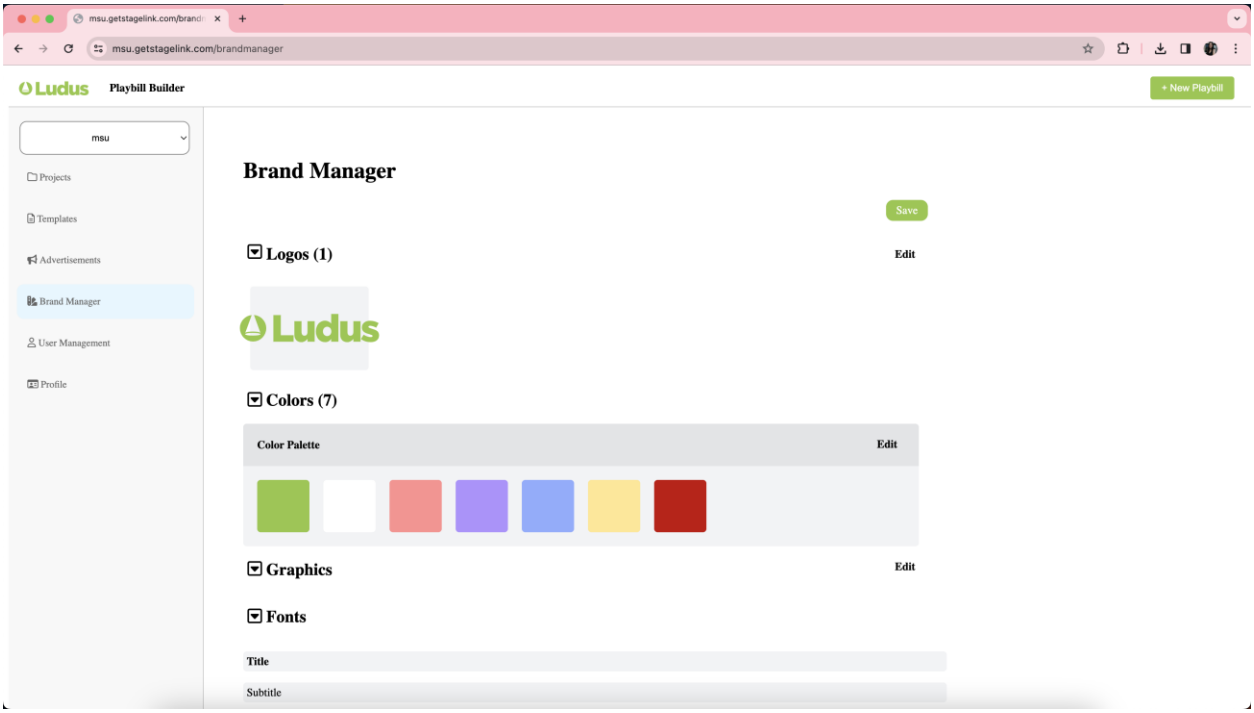


**Figure 2:** The brand manager tab of the user's dashboard

The fourth tab on the dashboard is the brand manager, as shown in Figure 2. This is where a user that is part of an organization can edit the themes and logos for their organization. Within this tab, they are able to add and edit graphics, fonts, and colors. The brand will then show up as an accessible theme within the editor, allowing the user to drag and drop graphics from the brand section (as shown in the sidebar of Figure 3). When adding text or changing an element's color, the user will be able to select from "brand fonts" and "brand colors".

## Builder

The builder is where a user will actually create their playbill. There are two modes a user can create a playbill in – desktop and mobile– each of which has its own view mode. The view mode is switched using a toggle at the top of the screen, as shown below in Figure 3.

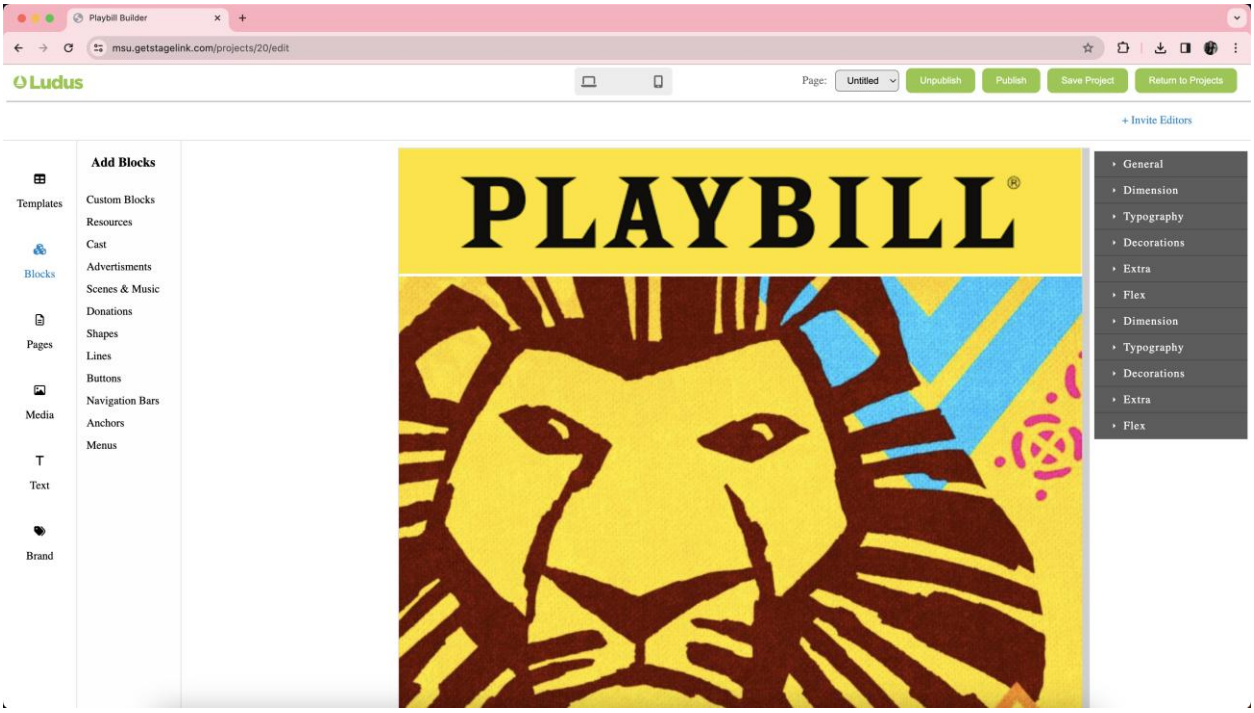


**Figure 3:** Builder in desktop view

Each view mode of the designer will show how the canvas would look at its respective ratio. From within the builder, the user can drag and drop components onto the canvas, including blocks, text, and media. If the user would prefer a templated canvas for them to personalize, they can also select from a range of site-wide templates in the sidebar. A user can also upload and use images and vectors, stored in the “media” tab of the sidebar. These can be either accessible at the user level or shared between all collaborators of the project. The builder also has the functionality of creating a paper playbill. The paper playbill builder comes with a grid view as shown in Figure 4 that displays all the pages in the current project for faster visualization and access.

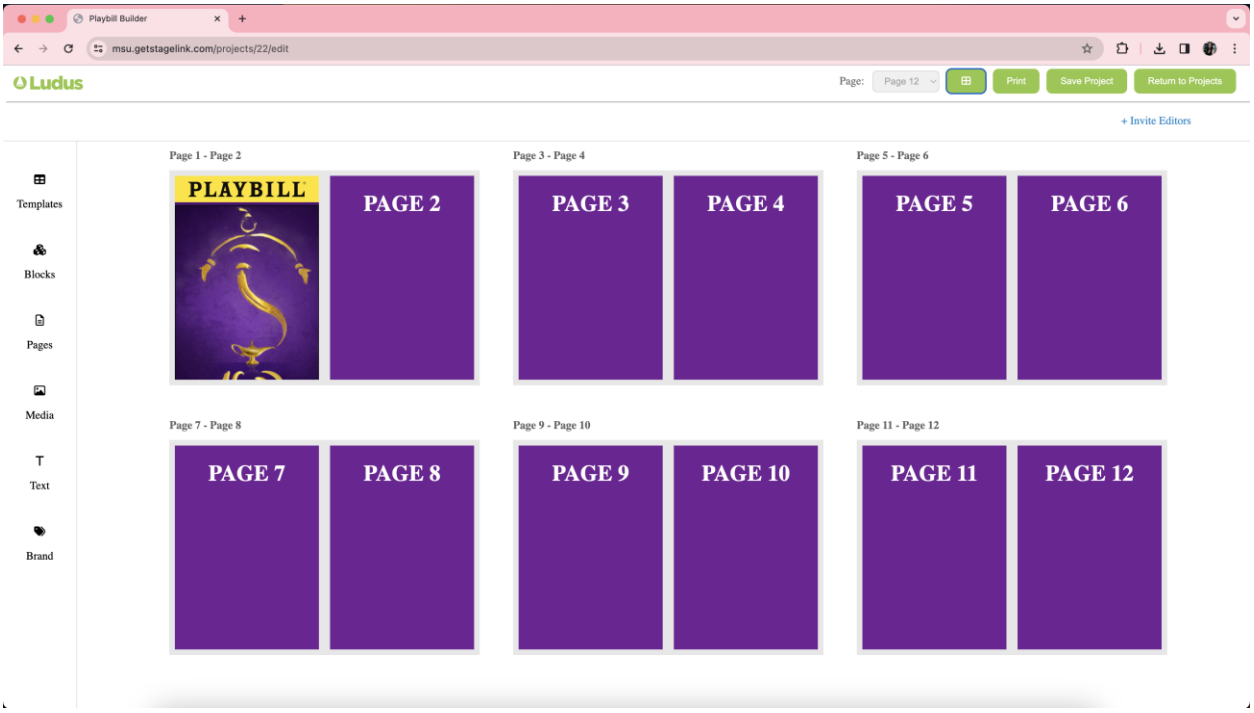


**Figure 4:** Paper builder grid view

## Templates

If a user does not want to start with a blank canvas, they can choose from a variety of pre-made templates. These are specific to each view mode (desktop, mobile, print). Users can also clone an individual existing project or an organization's project. Cloning an organization's project only brings over the designs from the project, allowing the user to customize the information.

## Blocks

Blocks are pre-made custom components that can be used across multiple designs where the users can drag a block onto their canvas, then customize the block to fit their production. Any edits made to a block are contextual to the playbill they are on, but admin users can edit global blocks through a dedicated block editor. Additionally, when a user selects multiple items on their canvas, they are given the option to create a new block. Any blocks that a user creates will live in the "custom blocks" section of the designer toolbar, visible in Figure 5.

## Back End Template and Blocks Builder

Only accessible to the Ludus team is a back-end builder page similar to the normal playbill builder page. Instead of creating playbills, the back-end builder provides the Ludus team a way to create and distribute new templates and blocks site wide for all users on the

platform to use. Upon publishing the new blocks and templates, the toolbar section associated with these new elements is updated across the platform for immediate use.

## Cast List Component & Form

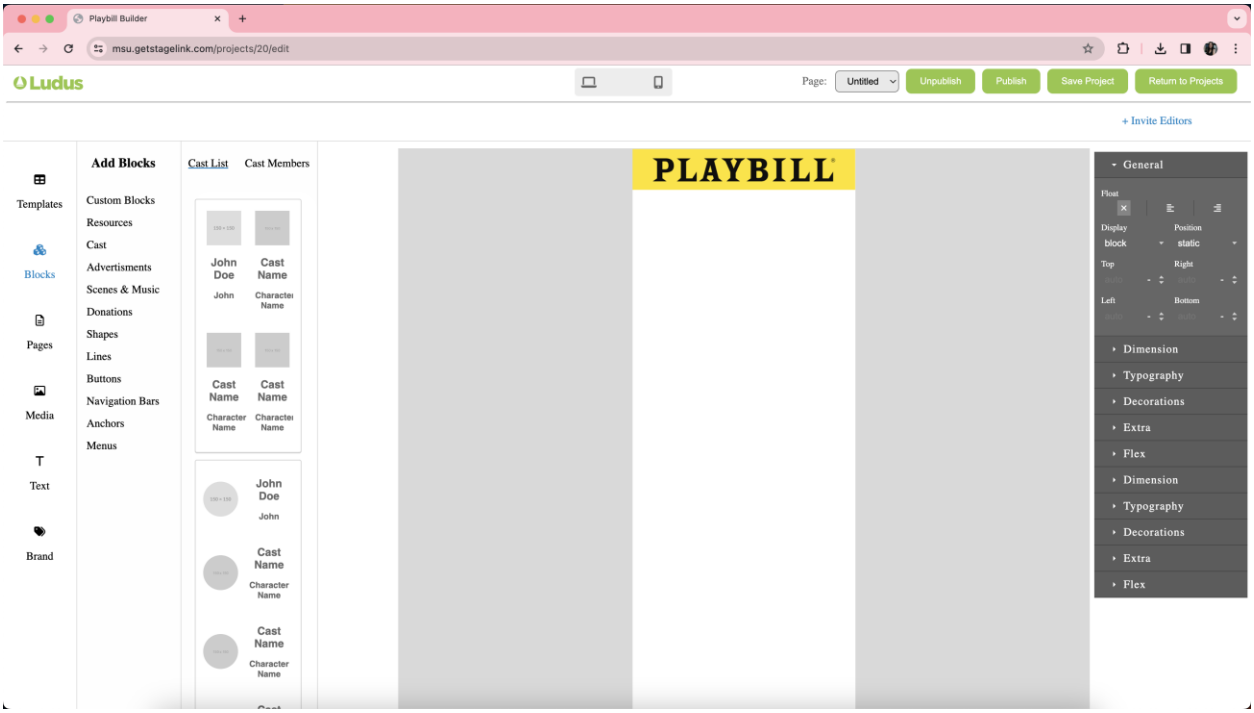


**Figure 5:** Cast list component within the builder

One of the blocks a user can drag and drop into their canvas is a cast list, as shown in Figure 5. The cast list components get updated with cast member information when a cast member fills out and submits their information from the cast form as shown in Figure 6. When cast information is needed, the user sends out cast list forms to each cast member's email. The form enables the cast members to fill out their own information, including name, role, biography, and upload a headshot of themselves. The cast list component is not editable by the user creating the playbill, the information can only be changed by the cast members themselves via the cast form.

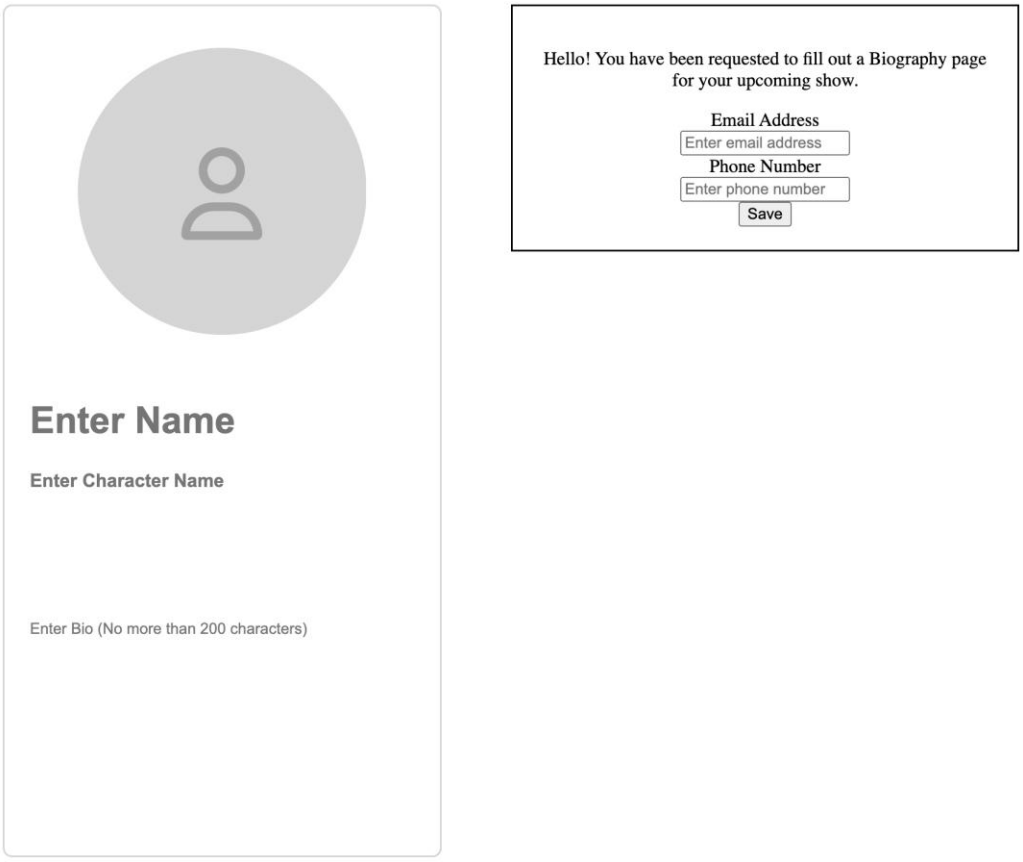


**Figure 6:** Cast list form

## Donation Component

Another block that a user can add to their playbill is a dedicated donation block, shown below in Figure 7. This will show up in the playbill as a separate “donate” button that links to a third-party donation service. The user just has to add the component and provide a link for the donation service. On the published version of the playbill, when the donation button is clicked, it will redirect the patron to the specified donation link.

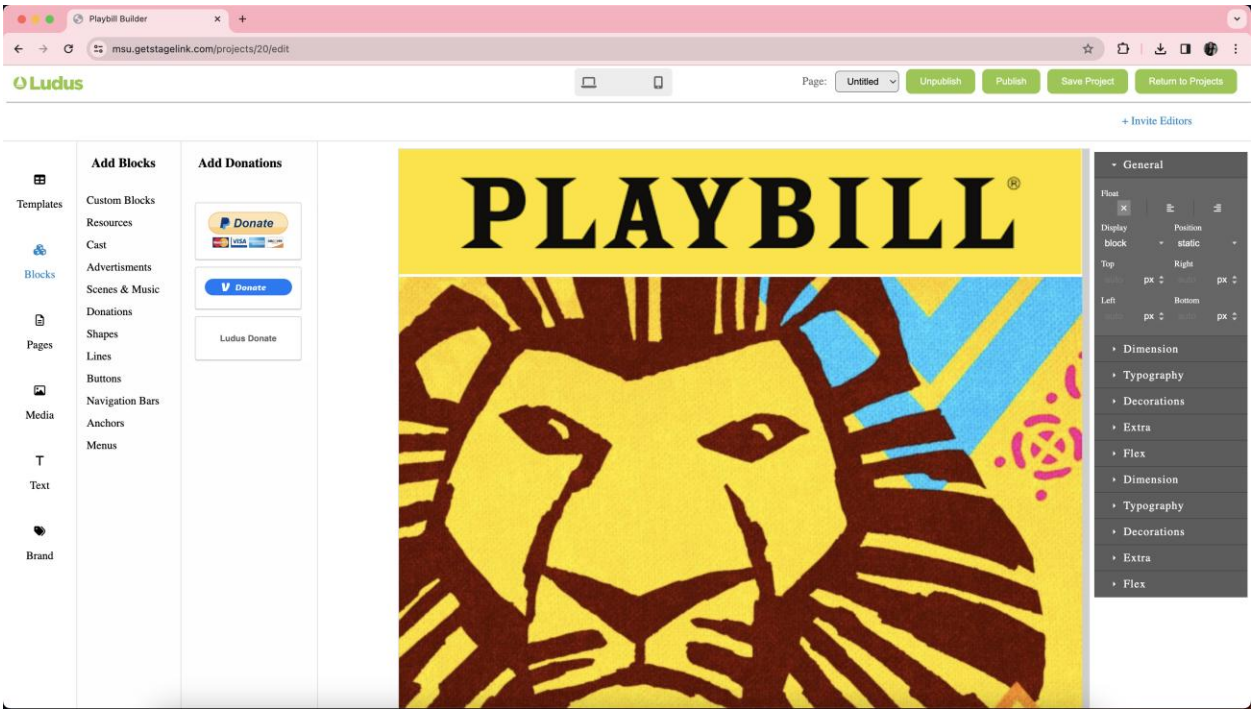


**Figure 7:** Donation component within the builder

## Sharing and Viewing

When a user is finished creating their playbill, they can publish it. Each published playbill will have a unique URL that hosts the digital version to be viewed by patrons. Publishing will also generate a QR code for a patron to scan in order to be taken to the hosted URL of the playbill. Playbills can be published multiple times to allow for updates and can also be unpublished.

The published playbill incorporates vertical "page stitching" so that the scrolling experience is seamless, rather than flipping through pages like a book. Figure 8 shows the QR code generated when a user publishes their playbill. After publishing the playbill, the QR code is accessible on the builder page when the user hovers over the publish button for later access as well.

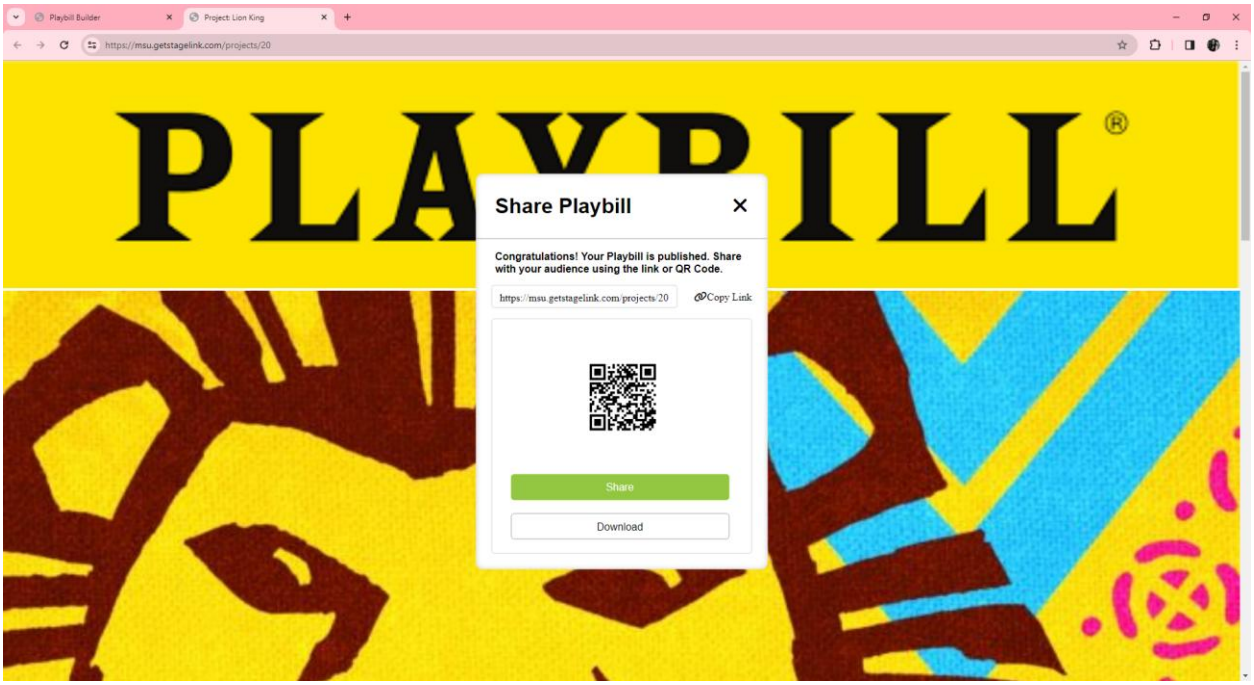


**Figure 8:** QR code displayed after publishing a playbill

# Technical Specifications

## Overview

The Digital-Playbill-Builder's backend is constructed using PHP with the Laravel framework. Simultaneously, the front end, developed using JavaScript and HTML & CSS, with an extensive use of the GrapesJS library, for rendering graphical interfaces and interactives elements.

Regarding system architecture, the application utilizes a MySQL 8 Managed Database to store data such as user profiles, login information, design templates, and playbill content. Stripe API is integrated for secure payment processing to further enhance the application's functionality. The application adopts a Docker-based containerization approach, facilitating uniform deployment across different environments.

The Digital Playbill Builder has a list of features for user engagement, including cast list management, advertising space management, interactive components, and a brand manager. For deployment, the application leverages the capabilities of Nginx on a Linux server.

## System Architecture

Figure 9 presents a completed overview of the application's architectural framework.

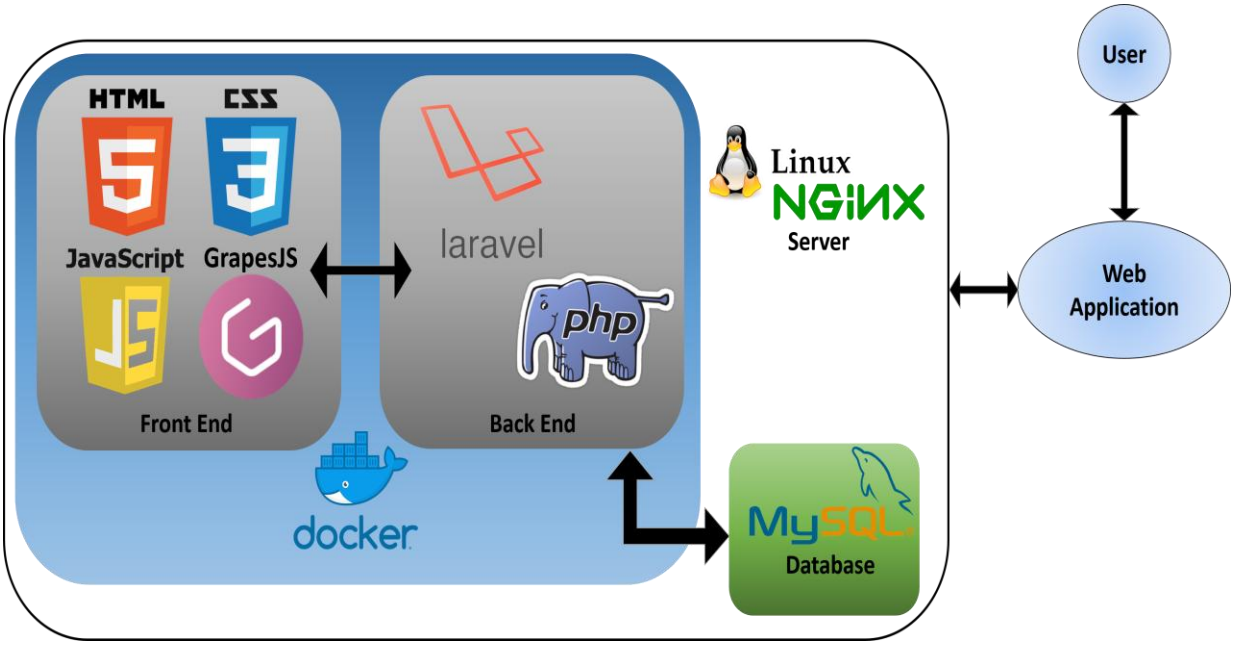


**Figure 9:** System Architecture

### Backend: PHP Laravel Framework

PHP, in combination with the Laravel framework, is chosen for the backend development. Laravel provides an efficient structure for managing the application's server-side functionalities and database interactions, essential for dynamic content management in digital playbills.

### Database: MySQL 8 Managed Database.

MySQL is the database of the Digital Playbill's management system, and its proven reliability and performance are critical for handling the application's data requirements, including user information, playbill templates, and design elements.

### Frontend: JavaScript, GrapesJS, HTML and CSS

JavaScript, GrapesJS, HTML and CSS are utilized for frontend development. This combination is essential for creating interactive and responsive user interfaces, crucial for the design and editing features of the Digital Playbill Builder. The GrapesJS library adds the HTML5 Canvas capability of drag and dropping interactive elements.

### Server: Nginx + Linux.

The application deployment will utilize Nginx on a Linux server, providing a stable and efficient platform for hosting the web-based Digital Playbill Builder, ensuring reliable access for users.

### Containerization/Composer: Docker

Docker is used for containerization, ensuring consistency across development, testing, and production environments. This supports a streamlined workflow, particularly important for the collaborative nature of the project.

## System Components

### Platform

- Web-based application.
- Responsive design for desktop, tablet, and mobile devices.

### Design and User Interface

- Blank canvas design.
- Pre-made templates.
- Tools for adding shapes, text, images, graphics, videos, GIFs, and links.

- Customization tools for color, font, size, etc.
- Grouping and ungrouping of elements.
- Multi-page design capability.
- Hidden pages and relative page links.
- Donation component integration.

### Functionality

- Cast List Management: A dedicated component for managing cast information.
- Ad Space Management: For displaying variable advertisement sections in the playbill.
- Block System: Pre-made custom components reusable across designs.
- Sharing and Viewing: Seamless experience for viewers to scan QR codes and view playbills digitally.
- Printable Editor: For creating physical versions of the digital playbill.

### Additional Features

- Dashboard for managing designs and templates.
- User account management with multi-tenant subdomain organization.
- Media Center for organizing images, videos, and gifs.
- Brand manager for specifying brand colors and logos.
- Project invitation and collaboration for multiple users on the same project.
- Overwrite protection for collaborative editing.
- Zoomable interface for enhanced readability.

### Integration and Testing

- Unit Tests: Incorporate a suite of automated unit tests using Laravel's built-in testing framework.
- Behavior-Driven Development (BDD) Tests: Implement BDD tests to assess the editor functionality, recommending the use of tools like Behat (refer to the Laracasts Behat Laravel Extension: Behat-Laravel-Extension).

### Stretch Goals

- Dark Mode - an alternative color scheme for the user interface that uses light-colored text, icons, and user interface elements on a dark background.
- Expansion of the assortment of blocks and templates.

- Patron Purchasable Shoutouts – Allowing patrons to purchase and send personalized shoutouts to cast members.
- AI designer - AI integration for design assistance using natural language processing and Open AI Api configuration.
- Billing & Payments – Integration with Stripe API to incorporate payment platform for purchasing playbill creation.
- Order a Keepsake for the Patron - Integrate a feature in the Digital Playbill Builder allowing patrons to order high-quality, printed keepsake versions of digital playbills.

# Risk Analysis

### Uploading and Storing Media

Difficulty: Easy

Description: Users can upload and share photos, videos, and GIFs to use in their projects. Uploaded media will be stored for future use and displayed in a media tab to drag onto playbills using HTML Canvas.

Mitigation: Research and practice uploading, storing, and retrieving different media formats with a database to ensure preservation of users' saved media before practicing integration with HTML Canvas.

### Publishing Playbills

Difficulty: Medium

Description: Playbills will be published to a unique URL and have a generated downloadable QR code. Published playbills will still be available to unpublish, edit, and update to the same URL.

Mitigation: Researching how other websites implement publishing pages and allow creator edits. Each team member has made an account for the website builder Wix to examine its layout and execution of site publishing.

### Dashboard Display

Difficulty: Medium

Description: Users can save and store project files in their account. The main dashboard will display users' saved projects and allow users to organize files into folders and share them with others to edit. The dashboard will have a search bar to enable users to search a filename and quickly find projects.

Mitigation: Plan and develop a mock file management system. This mock will allow us to practice efficiently retrieving shared files to catch potential issues with file preservation and display for all users with access permissions.

### Cast List Editor and Creation

Difficulty: Medium

Description: Playbills will have a drag-and-drop cast list that can be managed within a dedicated editor. Cast list bios and headshots are updated by individual cast members using a form with a secure link sent by the director. As forms are completed, projects with the corresponding cast list will be updated.

Mitigation: Creating a mock cast list to practice generating matching forms with a secure link. The forms will be used to practice finding and updating projects that contain the matching cast list, and deciding how to handle updating projects that are being edited when a form is submitted.

### Project Builder

Difficulty: Hard

Description: Building a drag and drop editor that allows users to drag different content onto a canvas layout. The editor will display numerous components and allow for the saving of new content. Data will be efficiently stored and managed to minimize overhead and maintain editor responsiveness.

Mitigation: Delegating toolbar sections to different team members to utilize strengths and quickly identify any components that may be difficult to integrate; meanwhile, creating a test project using each component type to test performance and observe the efficiency of our data management.

# Schedule

**01/08 – 01/14 – Week 1**

- Initial meeting with team
- Analyzed project proposal
- Initial meeting with Ludus
- Researched relevant technologies (PHP, Laravel, HTML Canvas)

**01/15 – 01/21 – Week 2**

- Initial triage meeting with TM Sam (Weekly on Tuesdays)
- Work on Team Status Report
- Set up GitLab / GitHub Repositories
- Set up PHP Laravel framework with Docker
- Set up barebones website
- Researched similar websites (Wix and Canva)

**01/22 – 01/28 – Week 3**

- Team Status Report Presentation
- Work on Project Plan Document and Presentation
- Build UI for Home Page (Projects Dashboard, Ad Center, Brand Manager)
- Build UI for Builder page
- Establish Schema for User and Project database tables
- Functioning Login/Create Account

**01/29 – 02/04 – Week 4**

- Project Plan Presentation
- Toolbar tabs UI (media, text, brand)
- Projects, advertisements, and organization back end
- Advertisement page UI
- Brand manager page UI
- User authentication
- Media section back end

**02/05 – 02/11 – Week 5**

- Template toolbar section
- Projects dashboard
- Secondary blocks toolbar section
- Brand manager back end

- GrapesJS research

**02/12 – 02/18 – Week 6**

- Work on Alpha presentation
- Subdomain creation
- Adding collaborators
- Design Day Booklet
- Brand manager page linking to builder page
- Text toolbar
- Builder page refactor to utilize GrapesJS

**02/19 – 02/25 – Week 7**

- Alpha Presentations
- Project saving and loading onto dashboard
- Cast form back end linking to front end
- URL routing for projects
- Donations blocks
- Advertisement blocks
- Project autosave and autoload
- Subdomain implementation
- User management page
- Cast form page
- Cast form back end
- Media blocks

**02/26 – 03/03 – Week 8**

- Work on team status report
- Github pipeline set up for server deployment
- Cast form linking to specific project
- Page toolbar section implementation
- Hidden pages implementation
- Add links to donation and button blocks
- Shapes, lines, buttons blocks
- Project sharing with users
- Project overwrite protection
- Brand section blocks
- Subdomain implementation
- Laravel emailer for cast form

- Page anchors toolbar
- Project searching on dashboard
- Block customization toolbar
- Scene and music blocks

**03/04 – 03/10 – Week 9**

- Team Status Report Presentation 2
- Printable editor
- Page anchor routing bugfix
- Menu blocks
- Back end for advertisements and ad space
- QR code generation
- Project renaming, duplication, copying, and deleting
- Resource blocks
- Button block modal
- Media file size upload and storage bugfix
- Paper playbill builder implementation

**03/11 – 03/17 – Week 10**

- Work on team status report
- Project cloning
- Backend Template + Block builder
- Menu blocks
- Linking buttons to playbill pages
- Brand blocks
- Dashboard for ‘Super Admin’ (Ludus team)
- Backend Template Builder
- Blocks grouping
- Custom blocks
- Right click menu for blocks on canvas
- Printing paper playbill
- Paper playbill pages addition instead of scrolling

**03/18 – 03/24 – Week 11**

- Team Status Report Presentation 3
- UI changes
- Project binding to blocks (specific blocks accessible to specific projects only)
- Custom blocks

- Interactive elements
- Donations and Buttons blocks published interactivity
- Backend Template + Block Builder display in toolbar sections
- Saving projects as templates
- Playbill unpublish
- Paper playbill page break implementation
- Paper builder grid view
- Placeholder blocks for back end builder

**03/25 – 03/31 – Week 12**

- Ad center implementation
- Media Center implementation
- Subdomain switch when user organization switch in dashboard for project access
- Bugfixing
- UI changes
- Feature Complete Application
- Beta Presentations

**04/01 – 04/07 – Week 13**

- Bug fixing
- Work on status report
- Project video script and recording

**04/08 – 04/14 – Week 14**

- Team Status Report for project video
- Project Video editing

**04/15 – 04/21 – Week 15**

- All Deliverables Due
- Design Day

**04/08 – 04/14 – Week 16**

- Capstone Wrap-Up