BRICKMMO Radio Requirements (Reporter CMS & Radio Page)

Andrew Barker n01553010

(In conjunction with Shavin Singh)

Tech Stacks

Reporter cms - Laravel-Blade-PHP
Producer cms - Laravel-PHP
Online Radio Station React
Open AI API
Text to Voice API
Shoutcast Radio mp3 player

Introduction

This document outlines the development plan for the BrickMMO Radio Project regarding the Reporter CMS and the front-facing Radio Station. The producer CMS and broadcasting db will be covered in the sister requirements document produced by Shavin Singh. This plan outlines the project's purpose, the development stack, application features and associated breakdowns, and a 7-week overview of the project's timeline.

Project Purpose

This is 24hr radio station based around the happenings of the BRICKMMO Lego City. The purpose is to bring the city to life by networking its elements together, simulating an actual city. For instance, all the other BRICKMMO projects will have a connection to the API. They will update the radio producer on updates to the city and other related scheduled segments. The Station will have a cms for reporters to submit ideas for radio segments (reports, games, ads, jokes). These segments' prompts will then be approved within separate producer cms (both by a human and via programmed logic). At this point, the prompt is sent to Open AI API to generate a script for broadcasting on the radio site within a daily schedule using mp3s generated via a Text-To-Voice API.

Features List

Must Have

- Reporter CMS
- SQL DB, all tables in place (see ERD) with functional relational communication and accompanying scripts
- Reporter & Producer CRUD
- Producer CMS
- Radio site and audio player

- Reporter CMS outputting report data in the proper format for Producer CMS to parse, review, and send to open ai script generation.
- Render returned scripts from Open AI onto a forward-facing radio site.
- Text-to-voice API configured and dj simulation
- Music playlist random generation

Should Have

- Same functionality as Reports for Ads, Jokes, Games segments and sub-segments.
- Unique login for different Reporters and Producers to only see and edit their content separate from admin login, who see all global data and have total CRUD capacity.
- Functionality for API calls from other BRICKMMO capstone projects (Pixilate, GPS, Colours)
- Styling and UI
- CMS displays relational lists related to Segments of the same type or topic.
- Daily rendering of unique updated 24hr content schedule
- Scheduled daily content from other BRICKMMO capstone projects
- API Documentation

Nice To Have

- Widget for uses to install radio player on own sites, RSS.
- Others can build their radio station from our central Producer API (SDK).
- Broadcasting in the actual city via speakers
- Featured segments appear as text blurbs on the radio homepage
- Radio home page form for listeners to get in touch

Narrative Description

Andrew will be building the Reporter cms using Laravel-Blade and a SQL database. Everything starts with a prompt created by a ‘Reporter’ (real person) or coming from another BRICKMMO component by automation. Prompts are a data component of a radio segment. The segment will contain a prompt inputted by the reporter with all related data. For example, the primary data from a report would include the reporter's name, segment title, and date created. There are also sub-categories for some Segment types. For instance, Report segments can be organized into sub-categories:

1. Local news
2. Arts & culture
3. Politics & current affairs

4. Sports
5. Science & technology
6. Music
7. Community stories
8. Human interest stories
9. Investigative reporting
10. International news

Any sub-category data will be included in the data saved for use later by the other features.

The database table structure and cms logic will be organized in such a manner as to feed the Producer CMS templated data depending upon where it is coming from (Reporter or BRICK MMO API), what type of Segment it is, and what accompanying features, if any, the data has (for example, the segment is a song request, and has a box ticked denoting it as a dedication). The Reporter CMS will output JSON formatted to include the segment's origin (Reporter name or BRICKMMO API), prompt text, the type of segment (by id), the segment name, and any possible sub-data.

When a reporter is logged in to the reporter cms, they will have access to all their previously created “Segments” and templates for creating any other type of Segment.

Once the producer receives data, it will either pass it immediately to the Open AI API (as is the case with data coming from another BRICKMMO component) or be submitted for review by a producer before Open AI script generation. If desired, the Reporter submitted data can also bypass producer approval and be passed to Open AI script generation immediately.

After generating Scripts, the Producer cms will store the scripts in a pool reserved for the broadcast schedule. The broadcast schedule will be organized with randomness and shuffling to maintain content variance. Still, other elements like the news and BRICKMMO content can be scheduled regularly and repeatedly.

The database tables will be structured to generate unique form inputs for each segment type so that all relational data will be accessible via foreign keys and joins using SQL. This functionality will be achieved by incorporating a table containing different template structures depending on the segment type created, read, updated, or deleted, thus adding more variations of segment types as easily as adding new template definitions. See ERD for more information.

Native Laravel error messages will display unless custom features need. For example, reporters can only log in using a Humber email address, so a custom regex will be used alongside a custom error msg.
Different login credentials will separate roles. Reporters can only perform CRUD on their own data, whereas admins will have global CRUD capabilities.

Styling will be handled so that there is a consistent theme across the cms and radio station and will incorporate the preexisting BRICKMMO Lego colour templates.

It should have functionality for all the types of segments and their sub-categories, but the first stage of mvp will focus on Reports and Reports sub-categories.

Shavin will handle the Producer cms, Ai text generation, text-to-voice, and broadcast scheduling and interaction. Details on these subjects can be read in Shavin Singh's requirements document, and an overview can be seen in the following:

The segment package comes in from the reporter and gets sorted into a "scripts" table flagged with a "needs approval" column. A producer can log in, view all the records marked "needsApproval", and select the one they want. Opening it up, they can verify the prompt, and if it's ok, click "send to AI writer." after they click that, the message is packaged and sent to chatGPT, which responds with a script for the producer to read. The producer reads the response script and selects "approve the script," The script is sent to the text-to-speech API, which will return an mp3 file that the producer can listen to.

After preparing everything, the producer selects "Approve Segment," which removes the "needsApproval" flag from the script. The script ID is then saved in a "bank" database, and an "approved_at" date is marked to indicate when it was approved, making the script ready for use. The database stores the original segment prompt, the raw script, and a path to the mp3 audio recording. The radio queries the most recently approved segments once daily and saves the audio file paths in an array of about 25 tracks. The array is shuffled, and the tracks are played sequentially, starting with the first track. When the mp3 finishes playing, the next track is loaded. To prevent track repetition, each track is removed from the array after it is played using "array_shift."

When the queue has run out (or is near to running out), the station will query the bank again, getting the 25 most recent segments. The player will endlessly loop if no segments are provided but will grab and play the most recent segments between trips to the bank.

Approving a segment has three steps a producer must do:

1. approve the user-generated prompt
2. approve chatGPT script
3. approve the Text-to-speech audio file

Andrew will be handling the Radio Station front end using React. It will utilize a SHOUTcast mp3 player and be fed the mp3 content described above from the producer cms. The audio content will be a continuous stream comprised of all the content types described above (i.e. music, news, jokes, tria etc..), some of which will be randomly selected, while other elements will play at regular times daily. For instance, most musical content will be selected via a shuffle function. Still, content from the colours BRICKMMO application will occur regularly, i.e. a segment for "the colour of the day." The site will have related content regarding the BRICKMMO city

background and additional content regarding the radio station. An overview of how the site works and utilizes content generated from user-inputted prompts will be explained, along with a link for someone to become a reporter (if eligible). If the RSS functionality is implemented, there will also be documentation on how to utilize the service.

User Stories

Listener User Story

A listener can visit the broadcast website (ex. BrickRadio.com) and click “play” on a web player (likely Shoutcast). The audio stream will play from its current live position.
The listener will also be able to peruse the site, which will have information on the BRICKMMO projects, their origin, and some featured stories and images from the BR Radio station content. The listener can also link to the reporter sign-in page and register as a reporter if they are eligible.

Reporter User Story

Adding a Segment: A Reporter can select “login” from the broadcast website and register for an account. Once registered, they can go to “Segments,” An interface will appear, allowing them to choose a type of Segment (Report, Joke, Trivia, Advertisement), write content and submit a cover image. The user selects the segment type, and the form will populate with fields that are appropriate for that type. Once they are done, they can click “Submit for Approval.” They will receive a confirmation message, then be redirected to their dashboard. Once submitted, they can review all their segments that have not been approved yet, and Modify or Delete them as necessary.

Modifying a Segment: In the Reporter dashboard, users can click “Segments” in the navigation bar, providing a drop-down menu of the different types of Segments. Once they click a Segment type, they will see a list of the selected segment type they have submitted but have not been approved as yet. If the chosen Segment type is a “Report,” they can click “Edit Report” and make changes to that report.

Deleting a Segment: In the case of a “Report” type Segment, In the “View My Reports” screen, a Reporter can delete a report that has not been approved yet. They will receive confirmation before removing it.

Reading Segments: In the case of a “Report” type Segment, In the “View My Reports” screen, a Reporter can see all the reports they have created, pending approval, and already approved. From this view, they can click Update, or Delete, at which point they will be brought to the desired page. It will also be possible to click the title of the Report and see any more related

details, if applicable. Additional details may be relational data, such as other reports based on the same topic.

ERD/Dataflow/Wireframes

A detailed ERD diagram, Dataflow Diagram, BPMN, and UI/CRUD Wireframes, alongside notes for the Reporter side of the DB, can be found here (please note the links to different pages on the top left):

https://www.figma.com/file/f00JHhkbHqh2G2gYljc8di/Radio-MMO?type=design&node-id=0%3A1&t=GRYmYPJVdy7SG6Ui-1