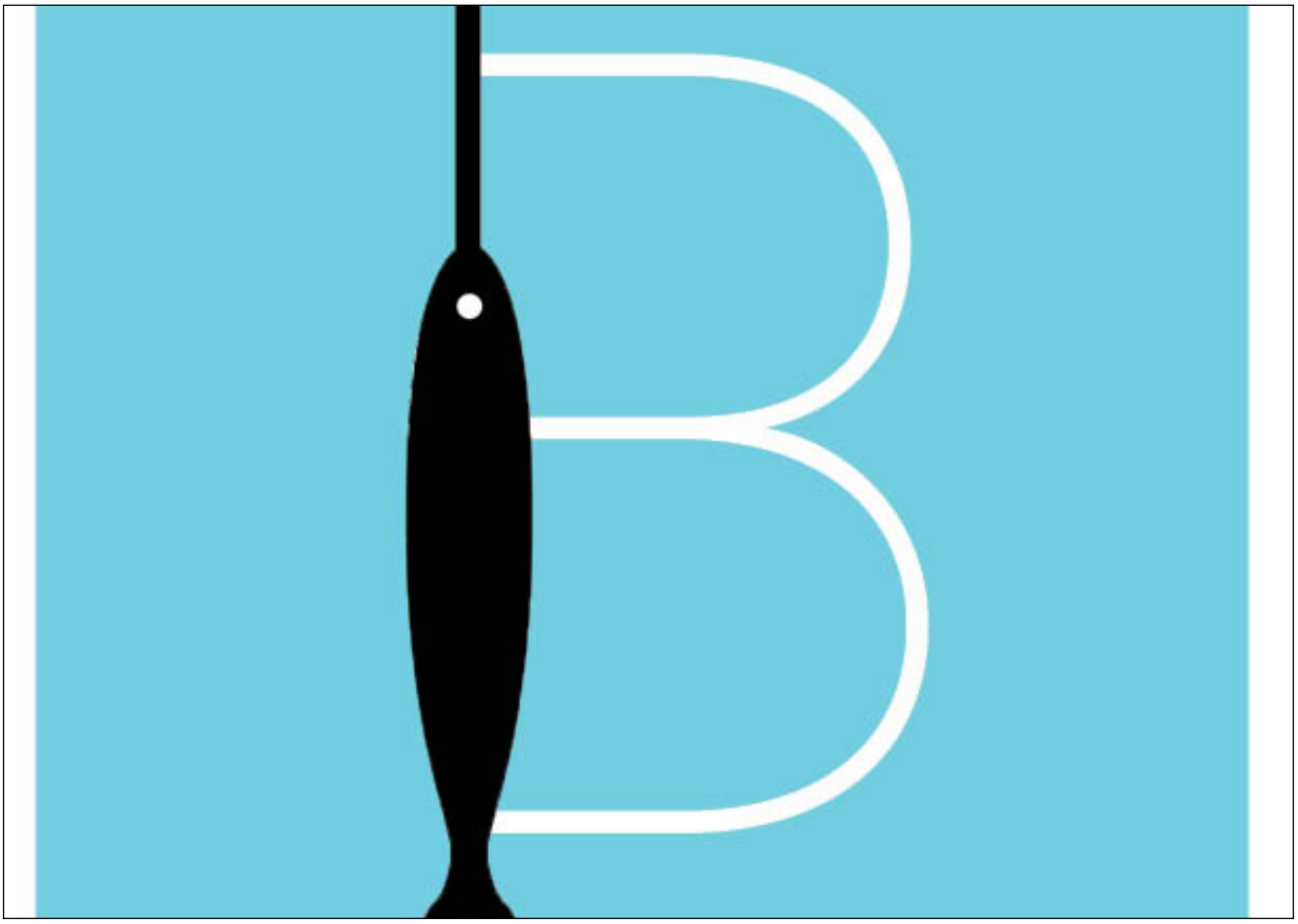
Ballyhoo

Lets find your hook.

Garrett Filippini - June 19, 2018



Glossary

1. Introduction: Discuss the main ideas to be accomplished during this business venture.
2. Technical
3. Installation Process
4. Packages:
5. Site Layout:
6. Scripts:
7. Licensing/Legal:
8. About:

Introduction

In short, Ballyhoo presents an easy, non-intrusive, clutter-less way to access and sort through your news articles. We want our users to feel in control and at ease when accessing news sources. It is easy now in days to search for reputable articles and somehow end up on a site that you have never heard before and will more than likely never end up on again. It is our mission to bring you the most reliable and diverse articles possible without impeding your trust, or your privacy.

Ballyhoo implements a feature called Hook's which essentially define what the article is about. A user can select as many Hook's as they would like, and your feed will consist of only articles containing those Hook's.

In addition to Hook's, we implement a the Veracity Scoring feature. We as an organization recognize that in todays press, there is a diluted market causing the quality of work deteriorating. We will present you with articles that are vetted by an unbiased algorithm capable of presenting you with the highest qualities of articles. The algorithm takes into account spelling errors, plagiarism, number of sources, etc. in order to bring you, without being biased, the articles of the utmost quality.



Technical

This project is composed of many technical parts, each piece tuned specifically to acquire the highest quality content to bring to our users. We have scripts to pull articles from a diverse pool of news sources. As these articles are pulled from their corresponding sources, we parse through the articles carefully filtering out any noun, and thus creating the Hook's of the article. We then store these articles into a database, content, images, Hook's and all which can then be accessed through one of our many (future) front end interfaces. The script for webscraping, and data-wrangling these articles is written in python powered by an MIT licensed node.js script. These articles are then saved into a local file that is then read into a python script that parses for the Hook's, assigns a Veracity Score, and ultimately stores the article into a database to be queried by Hook's by the user. Now that we have an overview on how the technical practices work, we can take a closer look at the Model View Controller (MVC) to see how the tech works together coherently.

The Model of this project refers to the schema (database) or the storage. This will be handled by a hosted SQL server. It is simple enough to create a basic schema where we can create plenty of read-only users (our user base) that can query the database efficiently by Hook's. We will store the user accounts in a separate schema with passwords encrypted using RSA encryption techniques. Any queries to the SQL database will be passed through prepared statements to avoid any SQL Injection attacks. We will also keep minimal third party software installed on the server to limit vulnerabilities to our users while implementing the safest practices while returning the highest quality product.

The View is our interface. Ballyhoo is going to be developed primarily as a web interface produced in HTML, Javascript, and CSS. Using open source works such as Bootstrap, Angular, and React we are capable of making a top tier interface. Clean clear and concise. We want to present our users with an easy to maneuver clutter free space to enjoy reading the news without throwing ads into their face, and without disruptive suggestion to articles that do not pertain to the Hook's they have implemented.

Lastly we look at the Controller of the project, think of this as the brain of Ballyhoo. The Controller will be written in python with a Flask server routing the site. This allows us to pass variables from our schema to our frontend with extraordinary speed through the implementation of Jinja2. This is system tied into the project through Flask that essentially complies our HTML files and can read in special Jinja2 variables into the frontend.