Spring 2019

Content Management System

Raymond Gines
rkg18@zips.uakron.edu

Honor's Project: Imprint CMS

By

Raymond Gines

Table of Contents

Abstract

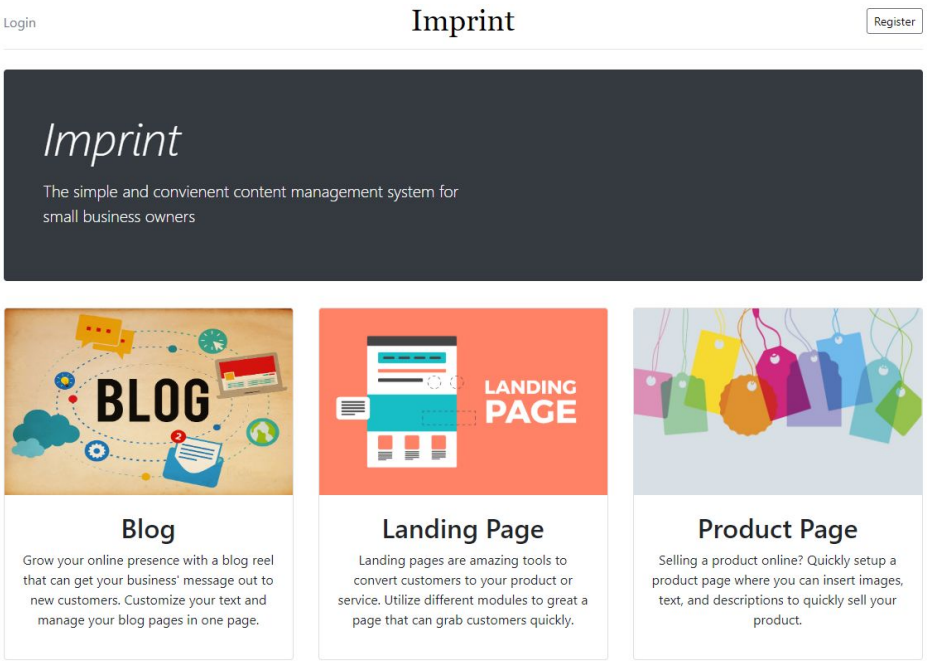
The Imprint CMS (Content Management System) aims to emulate the key features of popular existing CMS applications such as Wordpress, Drupal, Magento, etc. This will be a lightweight alternative that can be used to kick start a digital presence by producing landing pages, product pages, and blog posts with ease. Individuals will be able to register and create their own account that will be secured in a database that they can use to login and manage their pages and posts. In addition, they will have the option of changing multiple settings such as the theme and color scheme of their website.

I will be developing this application from scratch and utilizing the software development lifecycle to gather requirements, design the features, code, test, and deploy the application to the web. The overall goal is to be able to setup a website with the main features that can dynamically create pages on the fly and edit them as one wishes. I want to be able to experience all levels of development from the front-end, back-end, and server-side to create a single web application that is usable and provides value to users.

Github Repository: https://github.com/rkg18/imprint_cms

1 Introduction

The Imprint Content Management System was built with the aim of utilizing different web programming languages and frameworks to build a singular web application that would make it simpler to generate web content for individuals with a non-technical background. The development, creation, and research of this project is to emulate the development of real-world applications from scratch. This covers the standard software development cycle of gathering requirement, planning, development, maintenance, and repeating the process to perfect a usable web application.



1.1 Languages & Frameworks

The web application was built on a foundation of Python as a base language and using a micro framework extension known as Flask. Flask makes it easy to translate

Python code to browser and web languages. This programming language helps process form data and display information on database to the web interface. It helps with mostly backend data receiving, storing, and sending.

For the front-end side of the application, I've utilized common languages such as HTML and CSS for the structure and styling of the application. In addition to that, I am using a web library known as Bootstrap to help streamline the responsive design of the website. This library enables the application to have a usable interface on any device - mobile, tablet, or desktop. To top it off, I will be using JavaScript to change the stylings and settings of the page through actions.

1.2 Database

For this project, I have elected to use a lightweight SQL database known as SQLite that makes it easy to access, store, and collect information on the go. It is included in the library for the Python/Flask framework for convenient use and application.

The database will be used to mainly store user information such as login details, as well as digital content information such as titles, text, url, authors, media, etc. As for the images, we will be downloading them to the server but only storing the name of the respective image or video in the database to save space.

1.3 Libraries

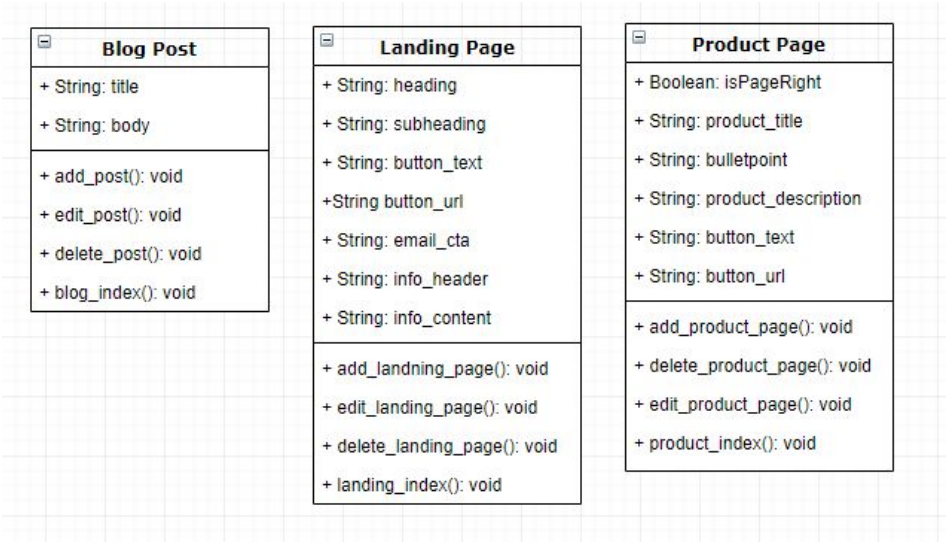
One of the main reasons I chose to use Python was because of the extensive access of 3rd party libraries that can help jumpstart a web application on a solo project

such as this. I utilized several open-source libraries in order to streamline the development and design of the project.

- **Slugify** – Takes a string or text (such as a title) and transforms it into a valid URL slug so it can properly display on a webpage.
- **Google Maps API** – Since this project was design with a small business in mind I wanted them to have the ability to broadcast their location so it would be easier to find their business.
- **GeoPy** – This library helps with parsing addresses to and from latitude and longitude coordinates that would then be plugged into the Google Maps API.

2 Design Process

The following will be a much more detailed log of what I have learned and applied in my project in terms of software, technology stacks, and the frameworks used. I will be explaining the main features of a CMS that I will be emulating and break down my design and thought process. Below is a basic UML design of the 3 main features and their respective properties and methods. In addition to that, I will be explaining the design of the other smaller features.



2.1 Requirements

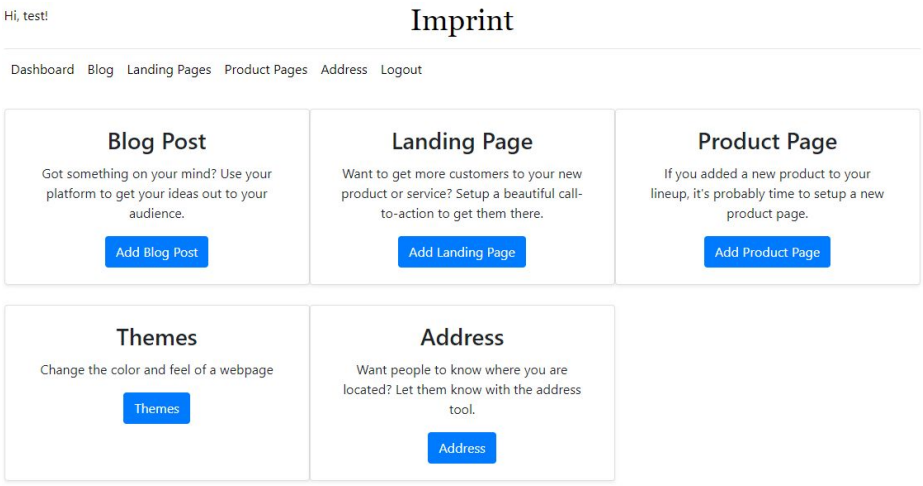
The main features I wanted to replicate with this project boiled down to 3 main items:

- Modern Responsive Design
- Flexibility in Adding, Editing, Deleting Pages
- Login Credentials and Authentication

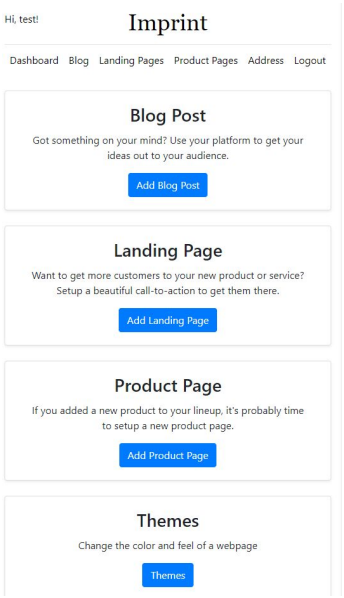
The main goal of the application, from a functional standpoint, is to allow a non-technical user to create a website and digital presence with a web interface. With this comes an individual's personal login credentials to take ownership of their website design, the attractive UI aspect of responsive design which is in accordance with modern day, and the ability to manipulate the content however they like. These were broad requirements going into the project.

2.2 Modern Responsive Design

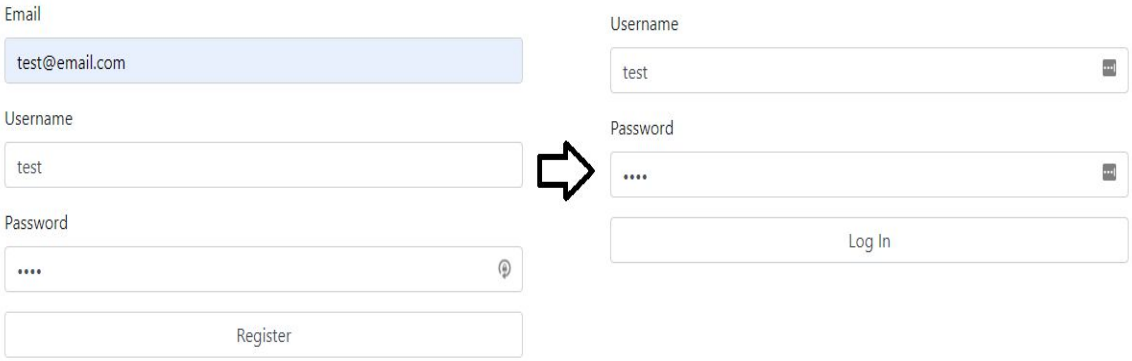
The modern responsive design was accomplished by using a library known as Bootstrap to help streamline the frontend process of creating a web interface that can function on mobile, desktop, and tablet. This way the application can be viewed on any size device based on screen resolution. The reason for this is because of Bootstrap's column/table mechanism it uses to position and properly size the pages contents according to the user's screen resolution.



The mobile view can be seen in the figure below. The dashboard applications will stack on top of one another so the user can see the entirety of the application's contents as well as access them easily



2.3 User Credentials and Authentication



The creation of users was done through HTML form handling and Flask to fetch and store information in the SQLite database. There are 2 form: register and login. They will both use Flask to fetch via the POST form method and will then error check the inputted information against the existing database to ensure there are no duplicates. Once all the information is validated it will be stored in the ‘user’ table using a SQL insert statement.

```python
@bp.route('/register', methods=('GET', 'POST'))
def register():
    if request.method == 'POST':
        email = request.form['email']
        username = request.form['username']
        password = request.form['password']

        db = get_db()

        error = None

        if not email:
            error = "Email is required"
        elif not username:
            error = 'Username is required.'
        elif not password:
            error = 'Password is required.'
        elif db.execute(
            'SELECT id FROM user WHERE username = ?', (username,)
        ).fetchone() is not None:
            error = 'User {} is already registered.'.format(username)
```

In addition, to add an extra layer of security, I leverage Flask's hashing function in order to further secure and hide password details instead of storing the plaintext information in the SQLite database. This way no individual with access to the database can see the plaintext password.

2.4 Content Management

One of the key aspects of any CMS is adding and managing different types of content. I will have 3 categories of pages that I developed with this prototype: landing page, product page, and blog posts. These pages are designed to cater to the common needs of business owners and a digital presence. Individuals who require a web pages to convert customers using an offer via a landing page to. The product page is designed for selling and informing online customer about a particular product or item on their own website. They can both be accessed through the dashboard and have their own respective forms to fill in the needed information that will be converted to the respective page the

user chose. Lastly, is the blog post which is just a content medium to help spread more information about a company's industry, needs, events, etc.

The goal of this content management system is to dynamically create web pages that is easy to add, edit, and delete. In the below image will serve as an example of the form handling, data insertion into the SQLite database, and passing that information to HTML templates to dynamically create the page.

```
""" Add Post function lets user add a blog post to their website """
@bp.route('/add-post', methods=('GET','POST'))
@login_required
def add_post():
    if request.method == 'POST':
        title = request.form['title']
        url = slugify(title)
        body = request.form['body']

        plain_body = body.replace('\n','<br>')

        error = None

        if not title:
            error = "A Title is required"
        elif not url:
            error = "A URL is required"

        if error is not None:
            flash(error)
        else:
            db = get_db()
            db.execute("INSERT INTO posts (title, body, plain_body, url, author_id) VALUES (?,?,?,?, ?)"
            db.commit()

            return redirect(url_for('blog.blog_index'))
```

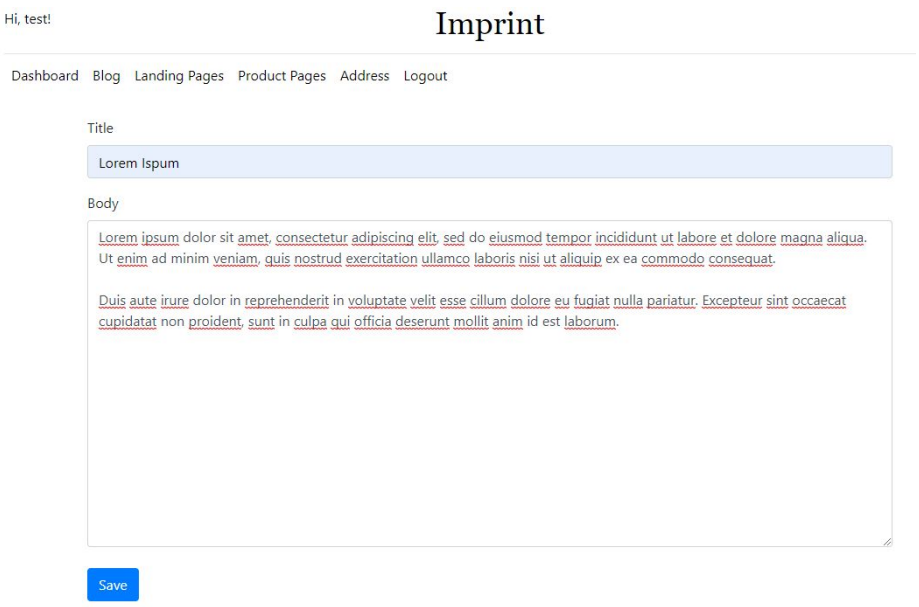
3 Blog

To add, edit, and delete posts I mainly used Python and Flask to manage receiving data from forms via the POST protocol. I then had to check for valid input and insert/update/delete the respective information. When displaying the web page and dynamically creating them, I had to ensure I structured the page that made sense from a user experience standpoint and display the correct information from the database. I

encapsulated all the blog information in a flask “Blueprint” object to hold the methods and properties and handle the web page creation.

3.1 Add Blog Post

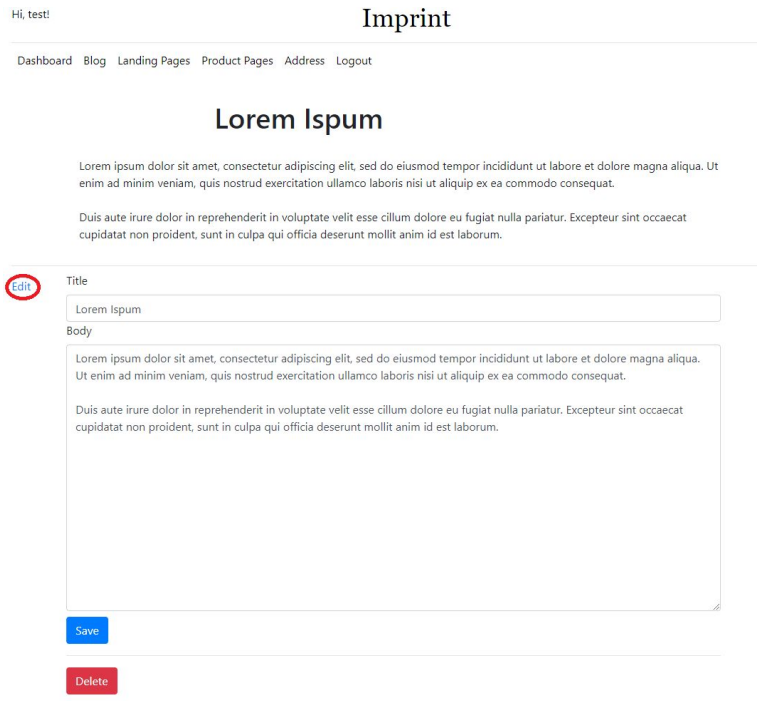
An important option of traditional content management systems are a blog reel that users can read to better inform themselves about a product or service before buying. This is also useful in growing an online presence via SEO (Search Engine Optimization). I have designed this blog post to allow users to input title and their respective body of text and format them in a natural way.



The output will be a dedicated blog reel under ‘/blog’ URL. But the pages can clicked to lead to the dedicated page that can be read in its entirety instead of a blog reel.

3.2 Edit and Delete Post

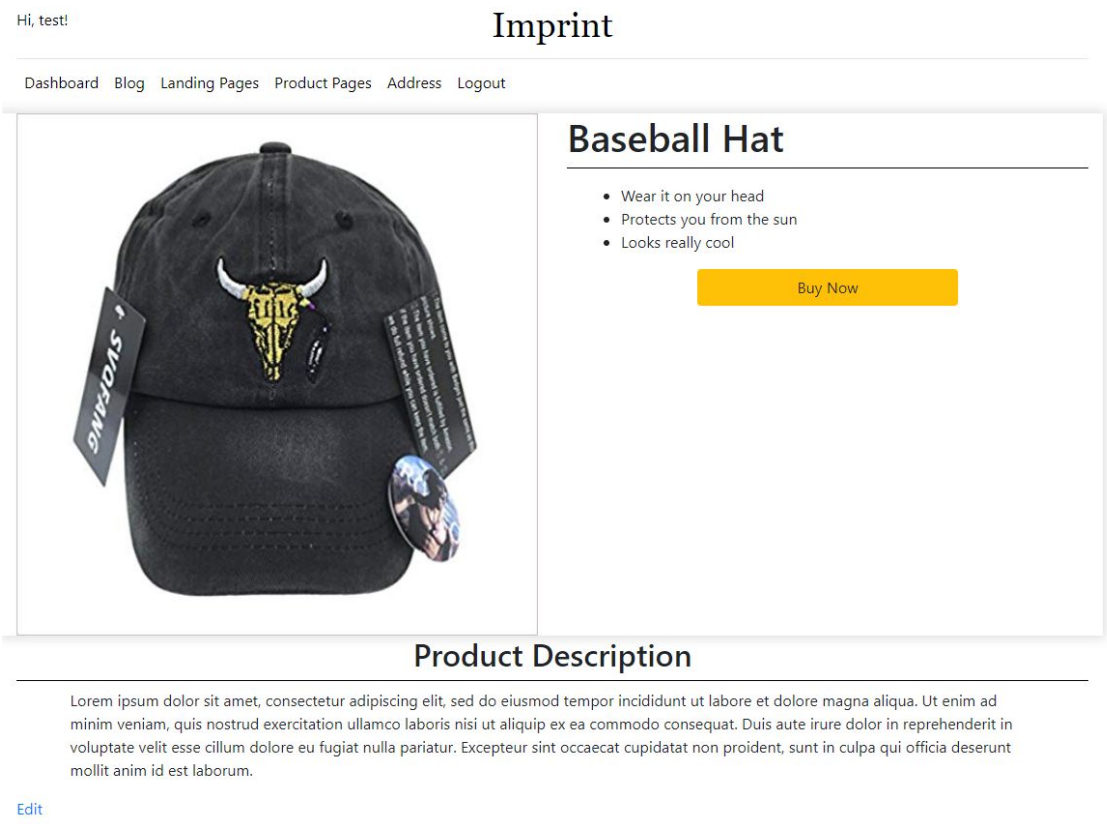
If an admin ever wants to update or remove a post I have added a simple button feature that can remove a post on command with a secondary error message to ensure it is what the user intended.



In addition, if a user selects the edit option. The application will receive the existing information and allow the user to edit and change anything they want or receive.

4 Product Page

Since the purpose of this application is to cater to small business owners, one of the main pieces of content that they can use is to add a product page. They can easily input their button text, images, links, and descriptions. The ability to manage and dynamically add, edit, and delete a content page is crucial for business owners in a digital age and online ecommerce presence. Like the blog posts before, I used Python and Flask to handle the data that came in from form input. I then used the respective SQL command to add or change the information for the product page.



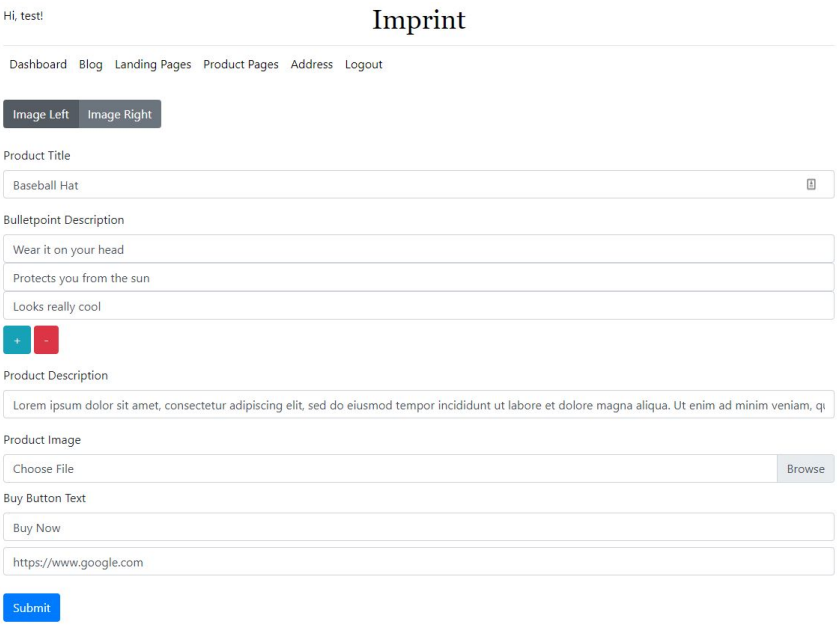
4.1 Product Index

The index page allows the owner to view and manage all their products in a concentrated area. In addition, they have easy access to essential information like product title and urls. And they also have the ability to view pages directly or go to an edit mode for them. Lastly, there is an option to add another product page so the user owner doesn't have to go the dashboard to add a product page.

Live Page	Edit Page	Product Title	URL Slug
View	Edit	dsagasdg	/dsagasdg
View	Edit	Phone Ring Holder	/phone-ring-holder
View	Edit	Yooooo	/yooooo
View	Edit	Test	/test
Add Product Page	.	.	.

4.2 Add Product

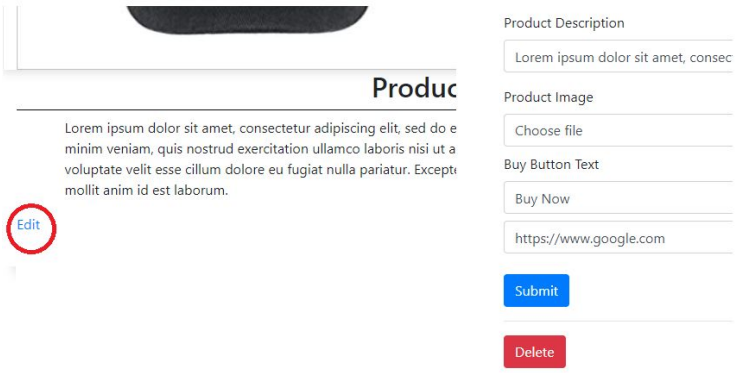
The add product feature provides the functionality to dynamically create and customize a product page. While you can easily edit the text of the page, you can also change things like the layout, URL, bullet descriptions of the page. An admin of a website can add and edit as many bullet point description as they want. In addition, there is the ability to select the layout of the page whether to have the product image on the left or right as well as the main product title/bullets. This allows individuals to pick a page layout that is suitable to their user experience. Lastly, a user can add and update the product photo as many times as they want as changes come about.



4.3 Edit and Delete Product

The option to edit a product page allows owners to keep up and enhance their product listing and description to get a better online conversion. This way if there are they need keywords or images one would like to upload, it can be easily changed.

And if there is ever a need to delete a product if an item is being discontinued or being sold out, then the option to delete the page is there to free up some space for the product listing.



5 Landing Page

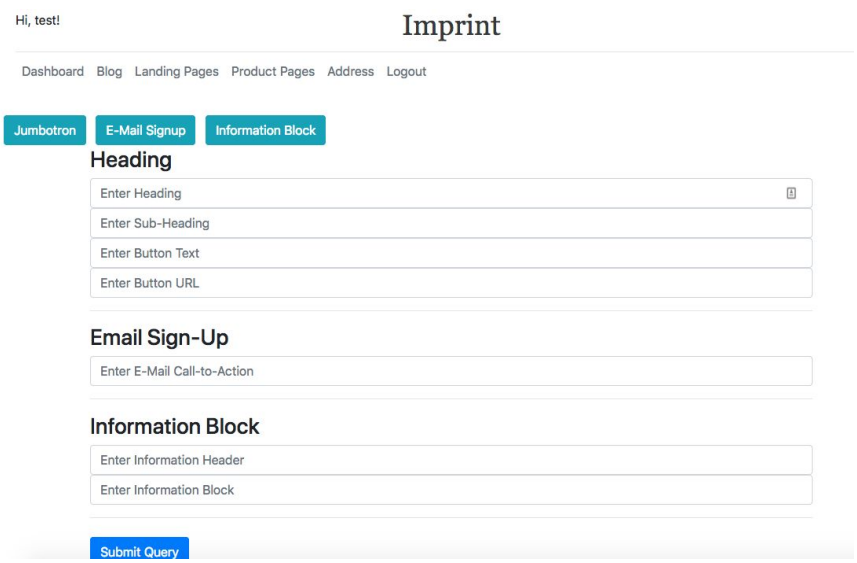
The landing page feature allows users to easily add and edit landing pages to attract and convert users to do a certain action. This landing page function allows individuals to grabs email or lead them to a product or offer page. I just Javascript and JQuery to grant users the ability to add and remove certain Landing Page elements they would or would not need. I used Python/Flask to handle the form data and execute SQL statements to update the given information.

5.1 Add Landing Page

The add landing page functions provides a dynamic user interface to add different elements that would be needed in general landing page. The 3 types of landing page blocks include:

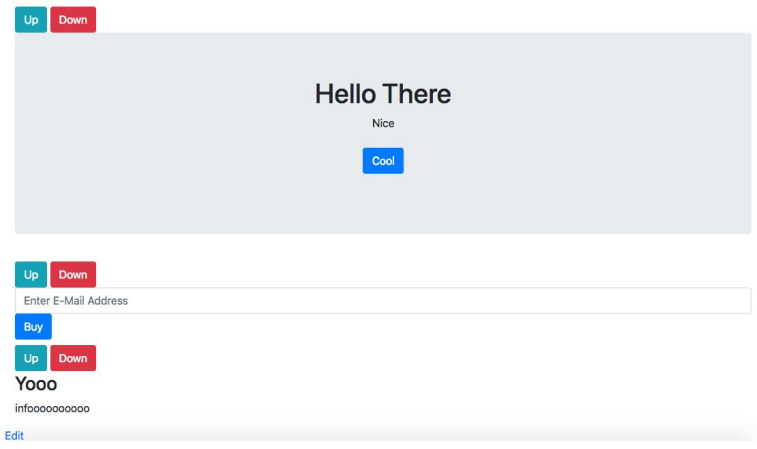
1. Jumbotron with Button Call-to-Action
2. Email Block to capture email leads
3. Information Block to convey additional features and such

These 3 blocks can be added or used depending on the users choices or options.



5.2 Edit Landing Page

The edit feature allows individuals to change and update the content of a given landing page. In addition, users are able to shift the elements of a page up and down to get their desired layout for the product. I used JQuery to be able to shift the different rows of elements up and down to allow users to customize the structure of the respective page.

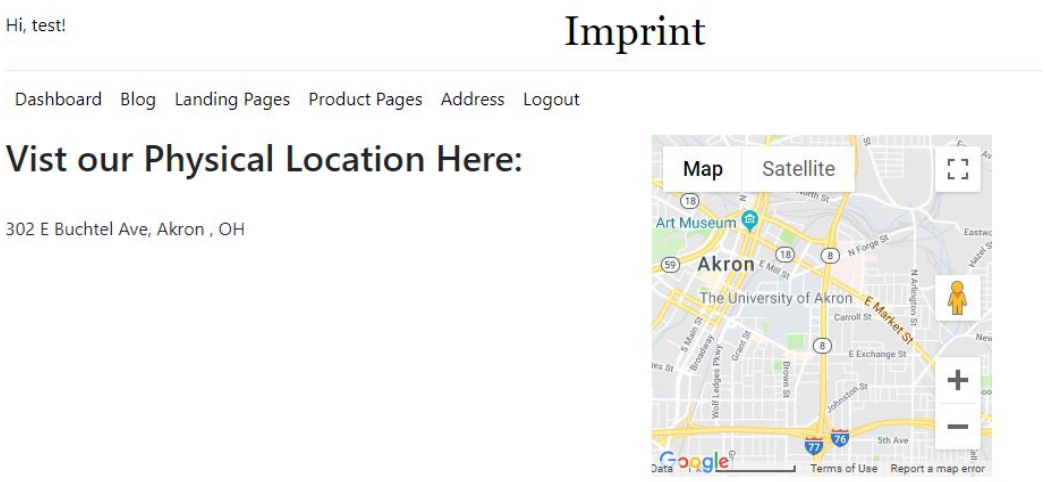


6 Other Content Management Essentials

To add to that usability of a content management system, I added additional features that I felt would compliment the use of this application such as an address to display a physical location and different colored themes to change the look and aesthetic of a website.

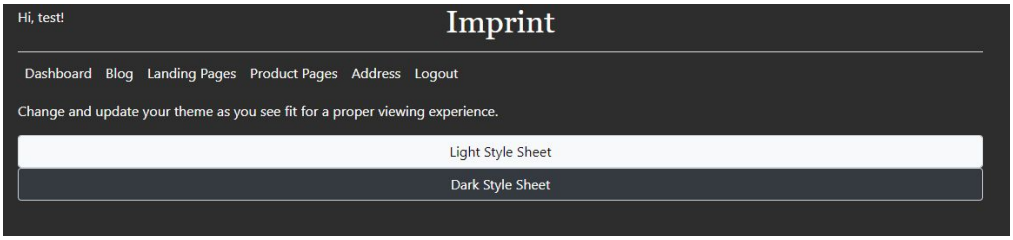
6.1 Address

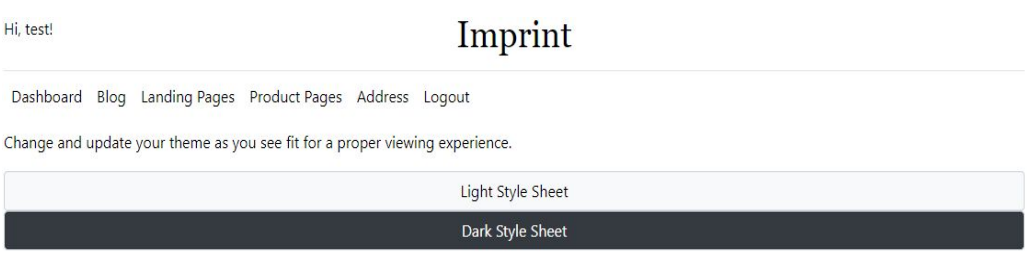
One of the key contact points for a small business is their location. I have implemented an input form where admins can input their business address and it will output a Google Map showing their business location. I have implemented this using the Google Maps Cloud API and a respective python flask API. I grabbed form data using Python and Flask and stored it into its own table. I then converted the address into latitude and longitude coordinates using a third party library, GeoPy. I then inputted those coordinates into the Google Maps API to display the surround area.



6.2 Themes

Themes are an easy way to change the look and feel of a website with different color palettes. For simplicity sakes, I added a dark and light theme to the project. To do this I used 2 CSS stylesheets and Javascript to change and shift the color of the different website elements.



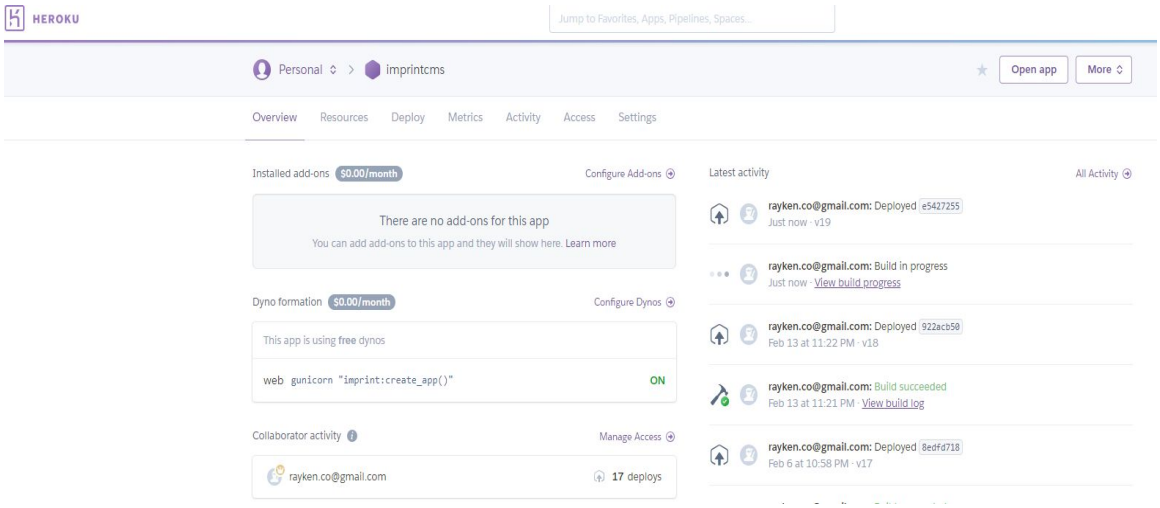


7 Deployment

One of the other aspects of shipping a product is deployment to a production where it can be seen by the public. While this is a personal project, I still wanted to be able to deploy this web application so it could be seen by companies.

A deployment service that is free to use and is relatively easy for Python/Flask web apps is Heroku. Heroku allowed me to compile and build the web app to deploy the web under their services. With building I had to ensure that I several different elements to be able to build the site with all the functionality on my local server and ensure it has the

same functionality on their server. To do that, I had setup the version of Python is the same on my local and production server. In addition, I had the setup a package manager to ensure all the 3rd party libraries would be applied as well.



8 Learning Outcomes

With this project, I was able to accomplish what I set out to do, which was to be able to built a small-to-medium usable web application using the Python/Flask stack. I was exposed the different elements of a software application: front-end, server-side, backend, database, and design.

For front-end, I applied my HTML/CSS/Javascript knowledge to make a modern and attractive design. I also leveraged open source libraries such as Bootstrap to help with the responsive user interface.

On the back-end I was able to use Python and Flask to help manage and manipulate the forms and its respective data to store and receive the information I needed to properly output the page. In addition to that, I was able to use Flask to dynamically

generate content that I was getting from the database to structure my pages. This complimented my front-end design when trying to structure where all the data and content should go.

I got exposure to using a database with SQLite and SQL when trying to execute SQL commands to get the records I needed that would be able to properly display the page.

9 Conclusion

This project will be deployed to Heroku for live site testing and usability while being worked on in my local machine. This project and application was meant to be a small scale demonstration of emulating usable and popular applications that exist out there today, but I wanted to build my own from scratch using different languages and tools to handle the development from beginning to the end. I was able to accomplish those goals and be exposed to the development cycle of building a usable and "real-world" application through this Honor's Project.