Gbiv Developer's Manual

Release 1.0

Dux D-Zine

Dec 02, 2022

CONTENTS:

CHAPTER

ONE

OVERVIEW

1.1 Technologies Used

- **Flask:** Used as a framework throughout the front and backend. Helped to pass data between modules and render dynamic web pages.
- **ColorThief:** Python library that allows the extraction of the dominant color in an image. Finds the statistical mode pixel hex value and returns that.
- **Sphinx:** Used in documentation generation (i.e., turning plaintext reStructuredText files into pdf and html outputs).
- **Microsoft Visio:** Used to create diagrams in documentation.
- **Jira:** Used for team management and organization of tasks.

1.2 Languages Used

- **Python:** Used throughout the project, notably used with the Flask library to create the structure of the application. The primary functionality of Gbiv was also developed using Python.
- **HTML/CSS** HTML and CSS both generated by Flask and hand written. These languages were essential in styling the website and creating a sophisticated user interface.
- **Jinja:** Used extensively on the frontend to create dynamic web pages that can change in response to user interaction as well as outputs from the backend side of the system.
- **reStructuredText:** Used as plaintext markup language for documentation.

CHAPTER

TWO

FRONTEND DEVELOPMENT GUIDE

2.1 Overview

This section describes how a developer would go about contributing to the frontend side of Gbiv. This includes the web interface (both static and dynamic pages) as well as the UI/UX as a whole.

2.2 Getting Started

2.2.1 Necessary Software

When developing frontend software for Gbiv, it is often helpful to be able to host the website locally to see how changes in the source code affect the application visually. In order to do this, one must download the libraries and packages listed in the requirements.txt file in the root directory. These can be downloaded in your personal system or by setting up a python virtual environment as described in this link. In either your machine or the virtual environment, run the command `python3 -m pip install -r requirements.txt` in the command line of your system. Note that depending on the OS you are using and certain configuration settings, this command may vary slightly.

2.2.2 Proficiencies

If a developer wishes to contribute to the styling, specifically of the static pages, basic knowledge of HTML and CSS should suffice for programming experience. However, to add on to the dynamic functionality of Gbiv's frontend, one should familiarize themselves with HTML/CSS as well as Python and Jinja. Python is how inputs and outputs are routed to and from the frontend side, while Jinja is used to generate dynamic page builds based on these interfaces.

2.3 Possible Areas of Improvement

- Smoother styling for suggested palettes and related colors
- Implementing a user interface that adapts to user uploaded photos (e.g. incorporating the extracted color in the design)
- More visual explanation on the color theory page

2.4 Best Practices

To begin with, all code should be written according to the accepted style guide for the language being used. Comments should be frequent enough to explain what is going on, but not so common as to complicate the reading of source code.

When it comes to Gbiv's frontend specifically, Jinja should be used for dynamic web page building, but used sparingly (i.e., only when needed for interaction with the backend). As much as possible, CSS code should be kept to the styles.css file and HTML code should be separated into files based on the page that the code is building on. For Python source files, docstrings should be included for each function and user-defined object. In addition to docstrings inline comments should be used as needed.

CHAPTER

THREE

BACKEND DEVELOPMENT GUIDE

3.1 Overview

This section is to aid developers who wish to contribute to Gbiv on the backend side of things. This applies to backend modules that define Gbiv's framework and routing, as well as the color analyzer module which provides the primary logic behind Gbiv's functionality.

3.2 Getting Started

3.2.1 Necessary Software

The software necessary to develop on the backend is very similar to what is needed for frontend developers. Downloading all the requirements listed in the requirements.txt file in the root directory will cover your bases for machine compatibility. If you would like to set up a python virtual environment to do testing, visit this link. Then, either on your local computer or in the virtual environment, run `python3 -m pip install -r requirements.txt` (exact syntax may vary based on OS in use).

3.2.2 Proficiencies

To contribute to or modify the backend codebase a developer should be well versed in python and have at least a basic understanding of the Flask framework.

3.3 Possible Areas of Improvement

- More palettes provided
- Add related colors section in recommender system
- Add filter tags for sample palettes and generated recommendations.

3.4 Best Practices

Developers should adhere to Python style guidelines as well as maintain adequate version control of altered elements to ensure that the codebase remains functional. It is also important to thoroughly comment any added code and to include docstrings/some other type of description for new objects and/or functions.

CHAPTER

FOUR

DOCUMENTATION DEVELOPMENT GUIDE

4.1 Overview

This section is intended to aid those who wish to contribute to Gbiv's documentation. The documentation for this project can be found in the "docs" folder in the root directory and includes user documentation, an SRS, an SDS, a initial project proposal, and a developer's manual (this document).

4.2 What You Need to Start

4.2.1 Software Necessary

To generate builds in the documentation folders, you must have Sphinx downloaded and the Read-The-Docs theme if you wish to create html versions. For more information check out these links to documentation for Sphinx and Read The Docs . In order to create graphics that fit with the style of diagrams used in the documentation, one should use Microsoft Visio (available for purchase here).

4.2.2 Proficiencies

To add to the documentation a developer should be proficient in reStructuredText for modifying content, Sphinx for generating builds, and Python for configuring Sphinx. Furthermore, contributors to documentation should be familiar with the basics of technical writing–key to this is knowing your audience and adapting the writing style to fit who will be reading the content.

4.3 Possible Areas of Improvement

- Further discussion on the specific principles of color theory utilized in the color recommendation script
- More explicit documentation for the source code itself (for development purposes)
- A more in depth explanation of how libraries were used in the system

4.4 Best Practices

Developers should follow standards of reStructuredText as a style guide. They should keep section formatting consistent with existing docs. It is also important to test builds with Sphinx often and to not push anything that has Sphinx errors or warnings. Diagrams included in the documentation should be high resolution and with clear labels and explanations.