Gbiv System Design Specification

Release 0.3

Dux D-zine

Nov 15, 2022

TABLE OF CONTENTS

CHAPTER

ONE

SYSTEM OVERVIEW

CHAPTER

TWO

SOFTWARE ARCHITECTURE

CHAPTER

THREE

SOFTWARE MODULES

3.1 Frontend Manager

3.2 Backend Manager

3.3 Web Interface

3.3.1 Main Page (Upload an Image)

3.3.2 Popular Palettes Page

3.3.3 Color Theory Page

3.3.4 About Us Page

3.4 Color Analysis

The primary function of this module is to the color analysis and generation that happens after a user has uploaded a photo. This module is made up of several sub-modules (divided by functionality) which are further divided into sub-sub modules. The static model below gives a visual picture of how the color analysis module is structured.

As the above model shows, essentially all of the work with color manipulation and analysis is done within the module. This makes for a high level of cohesion that allows for a weak coupling with other modules in the system. In fact, the color analysis module only has to interact with a single module which is the "Backend Manager." The backend manager passes an image in the form of a .png, .jpg, or .jpeg file and this module returns several sets of color codes as lists of hex code strings. The inter-module interactions of this part of the system are further specified in the dynamic model below.

This module was designed with a high degree modularity in mind. By separating the color analysis process into two parts, we are able to define two classes of sub-functions that share common features: palette generator functions and related color finder functions. This allows for code re-use and also source code that is easier to read and interpret. We also designed this module to have simple data types as both inputs and outputs. This allows easier integration with the rest of the system and fits well into our chosen framework (Flask).

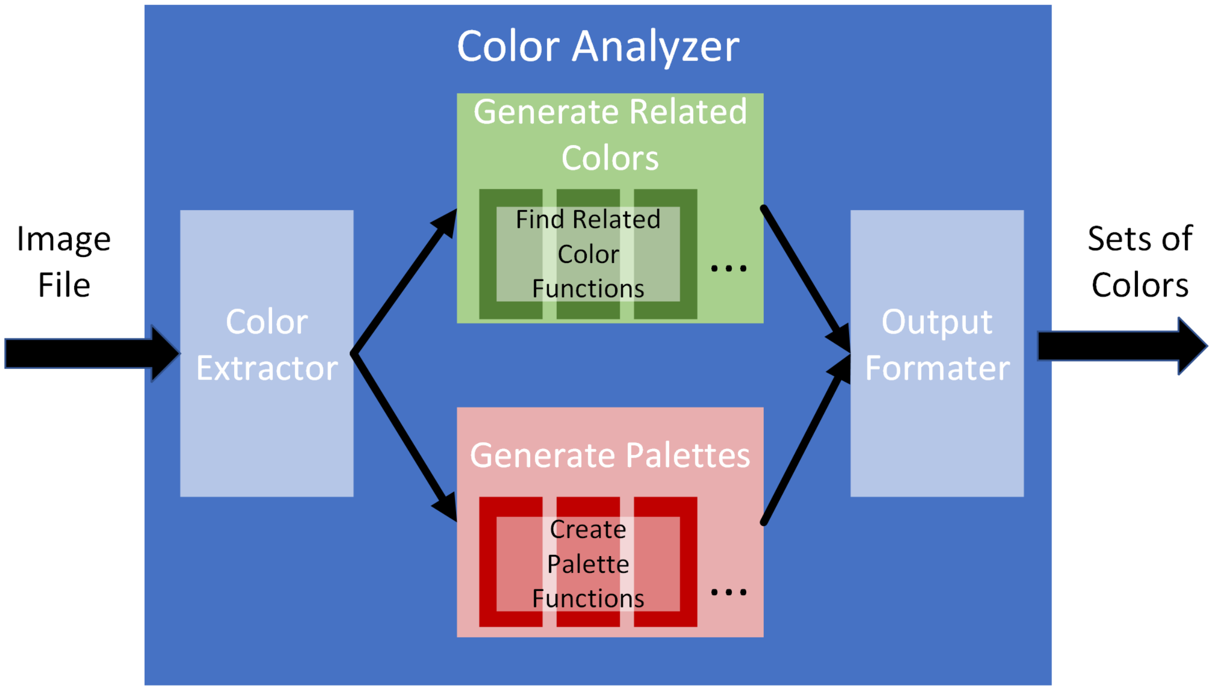

Fig. 3.1: Color Analysis Module Static Model

3.5 Palette Database

The purpose of the Palette Database module will be to store popular palettes that many visitors to Gbiv have looked at. This will allow users to view a range of different color combinations and get inspiration for their design projects. Because there is only one collection of elements in the database for this project, the design of the database itself is somewhat straightforward. The static model below shows the layout visually:

The technology we will be using to implement our database (MongoDB) comes with a library that allows for efficient interfacing through python. Because of this built in advantage, we have designed the system so that the database has one module with which it communicates, the “Database Interpretor.” This interface consists of a single type of input and a single type of output. When a user visits the “Popular Palettes” page, the frontend will query that backend which will reach the palette DB as a request to view the collection of palettes. When this query happens, the database will pass the collection on to the database interpretor module in a format that allows for easy movement to the end-users. Below we have included a dynamic model to demonstrate this interface.

The design choices for the palette database module were made with the goal of simplicity. By keeping the number of collections to a minimum and formatting all data entries identically, the organization and movement of Gbiv’s data can be straightforward and efficient. This prevents database accessing from being a bottleneck for performance, as well as reduces the need for more modules for data formatting.

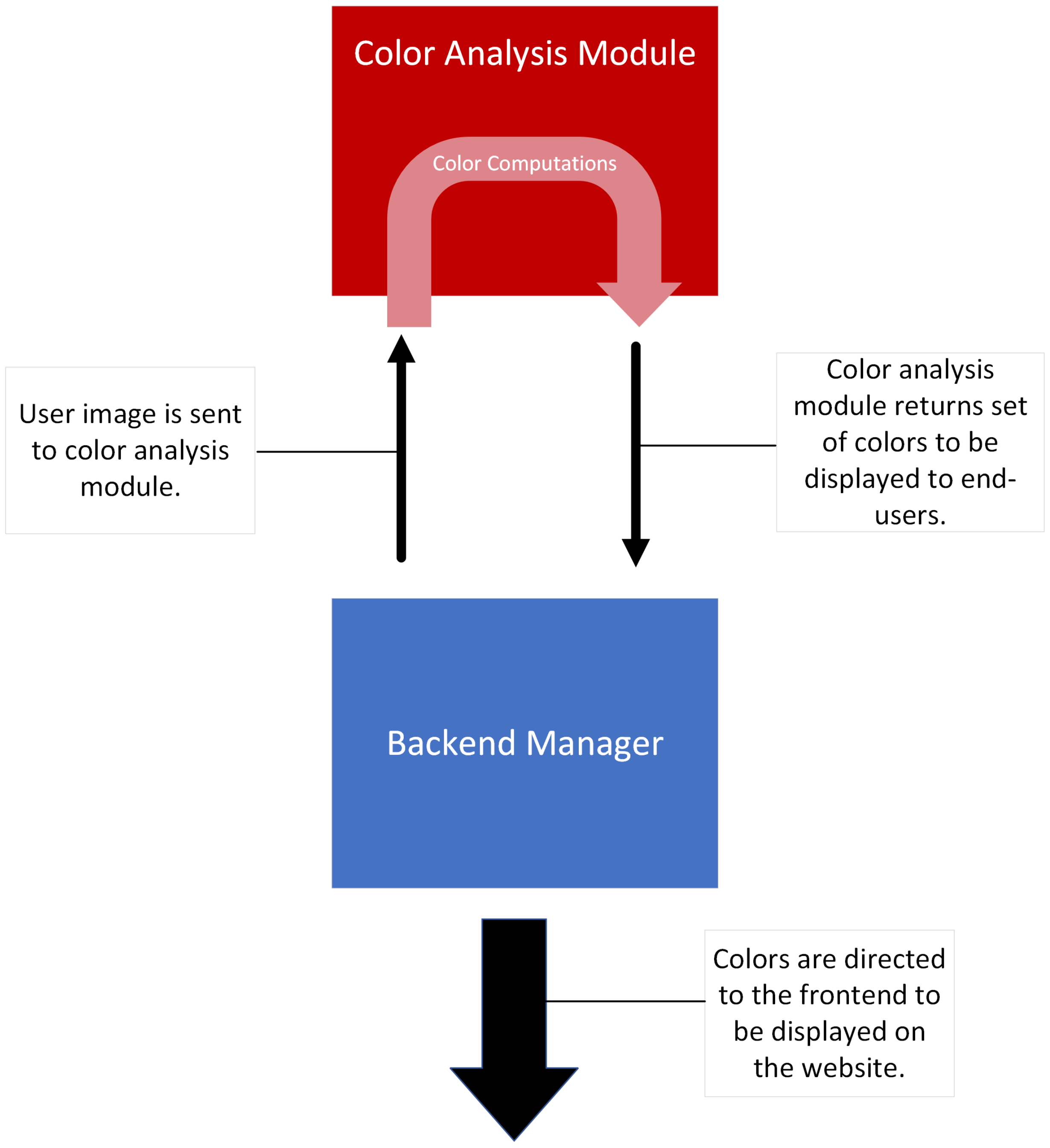
Color Analysis Module
Color Computations
User image is sent to color analysis module.
Color analysis module returns set of colors to be displayed to end-users.
Backend Manager
Colors are directed to the frontend to be displayed on the website.

Fig. 3.2: Color Analysis Module Dynamic Model

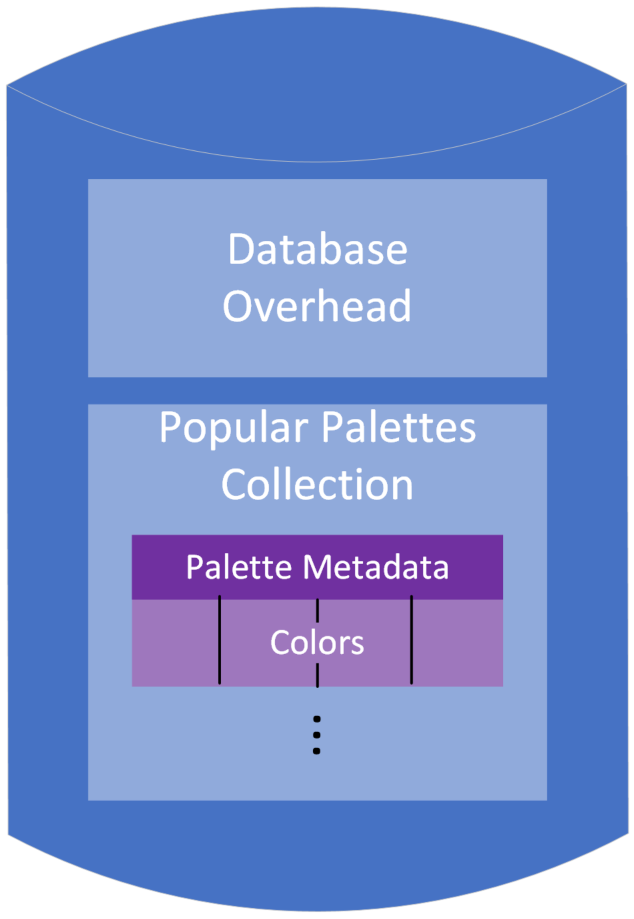

Fig. 3.3: Palette Database Static Model

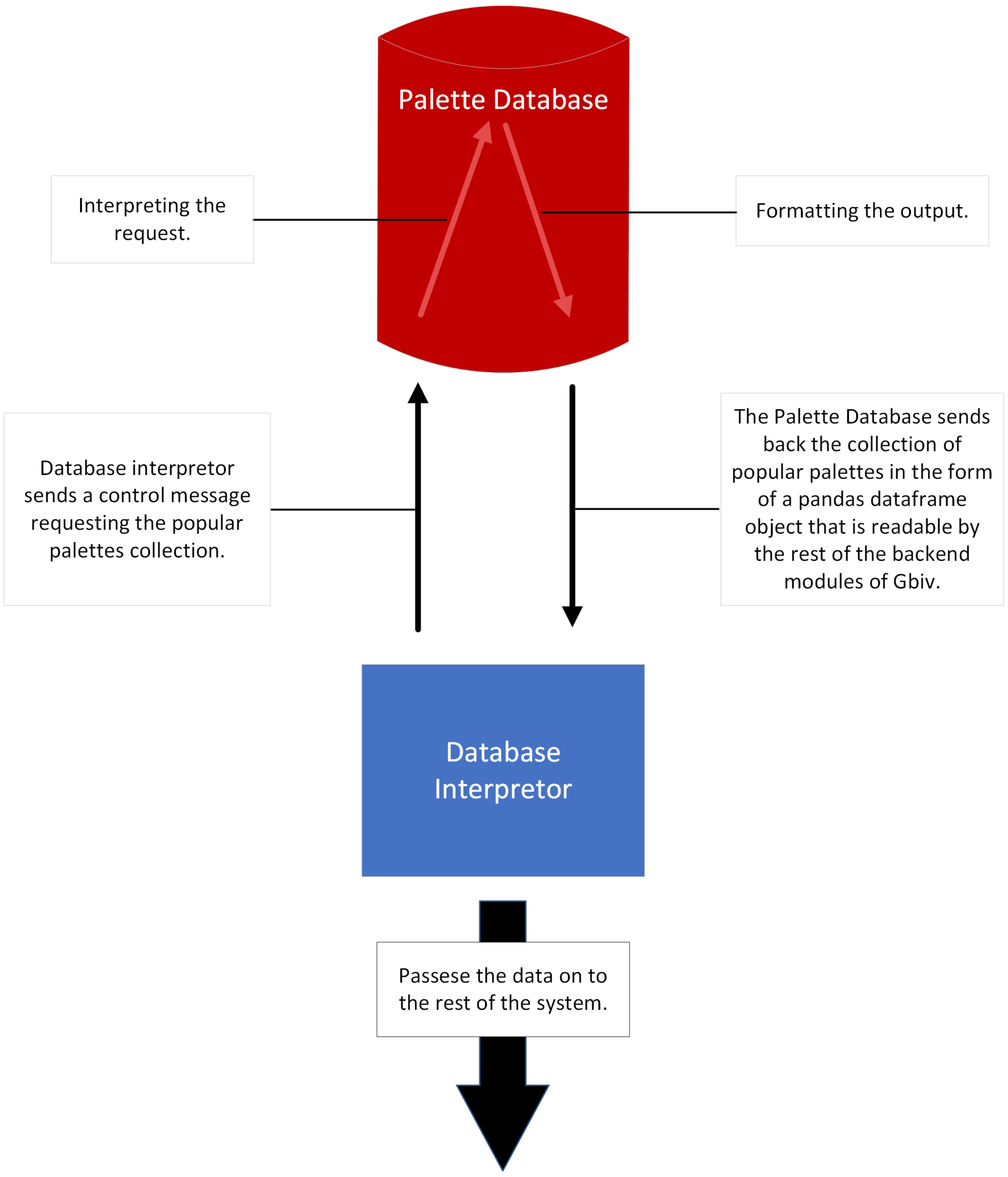
Palette Database
Interpreting the request.
Formatting the output.
Database interpretor sends a control message requesting the popular palettes collection.
The Palette Database sends back the collection of popular palettes in the form of a pandas dataframe object that is readable by the rest of the backend modules of Gbiv.
Database Interpretor
Passese the data on to the rest of the system.

Fig. 3.4: Palette Database Dynamic Model

3.6 Database Interpretor

CHAPTER

FOUR

DYNAMIC MODELS OF OPERATIONAL SCENARIOS

CHAPTER

FIVE

ACKNOWLEDGMENTS

The format of this SDS document was originally created by Professor Juan Flores. Reference to a completed SDS document was provided by Ronny Fuentes, Kyra Novitzky, Jack Sanders, Stephanie Schofield, Callista West with their Fetch project SDS.