BENGALURU NORTH UNIVERSITY

Tamakka post, Kolar - 563103

A MINI PROJECT REPORT ON

“Coloration”

Submitted in partial fulfilment of the requirement for the award of the degree

BACHELOR OF COMPUTER APPLICATION

Submitted By

Ms. Hinduja M (REG NO. R2011206)

Ms. Angel Anto (REG NO. R2011204)

Ms. K Lavanya Sai (REG NO. R2011208)

Under The Guidance Of

Mr. Anuraj A

Assistant Professor

Department of Computer Science

CHRIST COLLEGE OF SCIENCE AND MANAGEMENT

Alambady, Malur, Karnataka- 563160

For the Academic Year 2022-2023

CHRIST COLLEGE OF SCIENCE AND MANAGEMENT

(Affiliated to Bengaluru North University)

Department of Computer Science

Alambady, Malur, Karnataka- 563160

CERTIFICATE

This is to certify that the Project Work entitled "**Coloration**" is a bonafide work carried out by **Ms. Hinduja M** , **Ms. Angel Anto** , **Ms. K Lavanya Sai** bearing REGISTER NO: R2011206, R2011204, R2011208 in partial fulfilment for the award of **Bachelor of Computer Application** under **Bangalore North University**, Bangalore, during the year **2022-2023**. It is certified that all corrections/suggestions given for Internal Assessment have been incorporated in the report. This Project report has been approved as it satisfy the academic requirements in respect of project work prescribed by the said degree

……………………	………………………	..………………………….
Project guide	**Head of Department**	**Principal**
Mr. Anuraj A	**Ms. S Athilakshmi**	**Rev. Fr. Jince George CMI**

External Viva

Name of Examiners: **Signature with Date:**

1)

2)

CHRIST COLLEGE OF SCIENCE AND MANAGEMENT

(Affiliated to Bengaluru North University)

Department of Computer Science

Alambady, Malur,Karnataka- 563160

DECLARATION

We hereby declare that the project work entitled "Coloration" to the **Bangalore North University**, during the academic year **2022-2023** is a record of an original work done by under the guidance of **Mr. Anuraj A, Assistant Professor, Department of Computer Science, Christ College of Science and Management, Malur** and this project work is submitted in the partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Application. The results embodied in this project work have not been submitted to another University or Institute for the award of any degree or diploma.

Hinduja M (R2011206)

Angel Anto (R2011204)

K Lavanya Sai (R2011208)

ACKNOWLEDGEMENT

An Endeavour can be successful only when it is packed with the proper guidance and blessing. We hereby take pleasure in acknowledgement all those who have enabled me to complete this project.

We have great pleasure in expressing our deep sense of gratitude to, and I consider myself proud to be a part of Christ college of Science and Management, the institution that stood by my way in the entire Endeavour.

We take opportunity to express our thanks to our Principal **Rev. Fr. Jince George CMI**, Christ College of science of Management, Malur for permitting to carry work in college as per curriculum.

We take this opportunity to express our heartfelt thanks to **S. Athilakshmi Assistant Professor & Head,** Department of Computer Science for providing necessary facilities and guidance for the project.

We also extend our sincere thanks to the project **internal guide Anuraj A, Assistant Professor,** Department of Computer Science, Christ College of Science and Management, for his constant encouragement and support throughout our course, especially for the useful suggestions given during the course of the project period.

Hinduja M (R2011206)

Angel Anto (R2011204)

K Lavanya Sai (R2011208)

ABSTRACT

Coloration is an android application to create their own color combination. Today we came to know the circumstances that most of the people are dependent on physical designers or manuals to get color combinations. Through this application the user can get to know more about unknown colors and creative combinations. This application will be utilized for a variety of purposes, including animation, web design, art, architecture, costume design and etc. This project also includes a special feature where the user can create their own combinations of colors through this application. This android application is built in a way that we can create a login for all users. After logging in, the user is directed towards the homepage where he/she could see the major families of colors. Besides these, one could see options for adding favorite colors and also to create new combination of colors. Through this application, the user has many choices that can be selected by themselves instead of the designers influence or the limited options given by their designers. The details of each color have been provided under each family of colors such as color name, hexacode and description of the color. One could know all about the unknown colors in the color family so that they could decide their favorite colors and it perfect complementary color. This application also helps the designers to get ideas about color combinations and improve their knowledge. Thus, this application makes a perfect choice for the ones who loves various vibrant colors and its complementary color.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Color theory is both the science and art of using colors. This is an android application where the user can explore more unknown colors. Through this project the user can pick any color and find the complimentary color for that particular chosen color even without having to know about the concept called color theory. This application will be used for many purpose such as costume designing, web designing, art, architecture, interior designing, animation, etc… This application also helps the user to discover new color combinations.

The existing system is that people will have to find a guide for designing their inventiveness and asking help with them for colour combinations. In that case the designer will show only particular colors and combinations which is very common and used by everyone and also which is profitable to them without users complete satisfaction. In this case the user doesn’t have the chance to explore and choose their own colors.

In this application user can create their own color combinations easily by just using this application. To begin with, the user can create an account and login. In the home page, the user will find 9 families of colors which is yellow, orange, red, pink, violet, blue, green, brown and grey. From this the user can select any one family of color which will display them all the colors which belongs to that particular family. From the list of colors of that family, the user can pick any one color which will show them the proper complimentary colors for that chosen color.

1.2 Problem and Motivation

Today we came to know about the circumstances that most of the people depends on designers in order to custom anything such as wall painting, costume designing, art, animation, etc.. and also people have their own desires and ideas to be fulfilled which they depend on the designers every time by choosing any one of the option given by the designers even if they are not completely satisfied with that.

Thus, this application provides a platform where the users can know more about colors and color combinations even if they don’t have the proper knowledge on the concept of color theory. The users can customize their own combinations and get to fulfill their desires without the help of any designers.

1.3 Purpose of the Project

The purpose of this Android application on color theory is to educate users on the fundamentals of color theory and how to apply it in design. This app could provide information on color models, color schemes, and the psychological effects of color, as well as offer tips and best practices for using color effectively in design.

This app could be useful for graphic designers, web developers, and anyone else involved in creating visual content. It could also be helpful for hobbyists or students who are interested in learning about color theory and how it can be applied in various contexts.

In addition to providing educational content, an Android app on color theory could include tools and resources for users to experiment with different color schemes and palettes. This include features such as color pickers, color combination generators, and sample color schemes to inspire users and help them create effective designs.

Overall, the purpose of this Android application on color theory would be to empower users with knowledge and tools to make informed decisions about color in their designs and create visually appealing and effective content.

1.4 Objective of the Project

The main purpose of this project is it allows the user to explore more colours which are unknown to them. So, this project objectives includes following things:

- This project helps to consumes the time and effort of the user
- This project helps to create new colors using the complementary color
- It also helps the user to make their vision into reality
- We can also add our favorite combinations to Wishlist
- This project reduces the difficulty of browsing every time when there is a need for information on color theory and color combinations.

1.5 Scope of the Project

Not everyone is familiar with all colors. There are a few colors whose precise names we don't know. People today are very conscious of color, yet they might not have the right information or training to know what a color’s specific name is or what its complements are. The user can utilize this application to identify ways to help them get past this obstacle. The major goal of this project is to make it easier for users to locate and explore new undiscovered colors by simply utilizing this application, which will teach them more about colors and enable them to bring their own visions to life.

The user can choose any of their favorite colors and this application will propose complementary hues for that particular color based on their selection.

CHAPTER 2

LITERATURE SURVEY

A literature survey is a piece of discursive prose, not a list describing or summarizing one piece of literature after another. It is an iterative process, assessing and distilling information. One of the key purposes of the literature survey is to investigate a problem that no one else has addressed. A literature review establishes familiarity with and understanding of current research in a particular field before carrying out a new investigation. Conducting a literature review should enable you to find out what research has already been done and identify what is unknown within your topic.

2.1 INTRODUCTION

The body of the literature review is intended to give your audience an overview of the already existing research on your topic. This can serve several purposes, including:

1. Color theory basics: This section could include a brief overview of the color wheel, color models such as RGB, CMYK, and HSL, and color terminology such as hue, saturation, and brightness.
2. The psychological impact of color: This section could focus on the psychological effects of color on the human brain, including how different colors can evoke different emotions and moods. This information can be useful in designing user interfaces that convey a particular tone or emotion.
3. Color schemes: This section could discuss the various color schemes that are commonly used in design, such as monochromatic, complementary, and analogous schemes. It could also cover how to use these schemes effectively in Android app design.
4. Color accessibility: This section could focus on the importance of designing for color accessibility, particularly for users with color vision deficiencies. It could cover techniques such as using high contrast color combinations, providing alternative color modes, and using textures or patterns to convey information.
5. Best practices for color design in Android: This section could provide guidelines and best practices for designing with color in Android, including tips for selecting a color palette, using color in typography, and designing with color consistency across different devices.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATIONS

3.1 Introduction

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS). This report lays a foundation for software engineering activities SRS is a formal report, which acts as a representation of software that enables the customers to review their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system.

The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. The SRS could be written by the client of a system and by a developer of the system. The first case, SRS is used to define the needs and expectations of the users. The second case, SRS is written for various purposes and serves as a contract document between customer and developer.

3.2 System Requirements

3.2.1 Hardware Requirements

RAM	**:**	2 GB RAM or above
ROM	**:**	16 GB or above

3.2.2 Software Requirements

Operating system	**:**	Android 8 or above
Front end	**:**	FLUTTER
IDE	**:**	Android Studio
Back end	**:**	Google Firebase

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 Functional Requirements

The functional requirements for a system describe the functionality or the services that the system is expected to provide.

- **User registration and login:** The app allow users to create an account, login with their credentials, and maintain their profile information.
- **Search and filter:** The app allow users to search for content or filter content based on specific criteria such as color names, families, etc.
- **Navigation:** The app have a clear and easy-to-use navigation system that allows users to move between different screens and sections of the app.
- **Multi-theme support**: The app support multiple theme to accommodate users from different settings i.e. dark or light.
- **Accessibility:** The app is accessible to users with disabilities, with features such as voice control, text-to-speech, and font size adjustment.

4.2 Non-Functional requirements

These are the requirements which are not directly concerned with the specific functions delivered by the system.

- **Performance:** The app is fast and responsive, with minimal load times and smooth transitions between screens.
- **Scalability:** The app can handle a large number of users and data without significant performance degradation.
- **Security:** The app is secure, with robust authentication, encryption, and protection against hacking, malware, and other cyber threats.
- **Compatibility:** The app is compatible with a wide range of Android devices, screen sizes, and resolutions.
- **Usability:** The app is easy to use and navigate, with clear and intuitive interfaces and consistent design elements.
- **Reliability:** The app is stable and reliable, with minimal crashes, bugs, or errors.
- **Maintainability:** The app is easy to maintain, update, and extend, with well-documented code and modular design.

CHAPTER 5

FEASIBILITY STUDY

A feasibility study is an important process in project management that assesses the viability of a proposed project. The purpose of a feasibility study is to determine whether the project is feasible, both technically and financially, and whether it is worth investing in.

5.1 Technical Feasibility

This assesses that the project is technically possible and the necessary technology exists to support the project. This may include an analysis of the project's design, development, testing, and implementation requirements. The application is designed to run smoothly and efficiently on a wide range of Android devices. The user interface of the application is intuitive and easy to use, with clear instructions and visual cues for selecting and manipulating colors. This application is compatible with a wide range of Android versions and devices, to ensure that it can reach the largest possible audience.

5.2 Economic Feasibility

The economic feasibility of a coloration application would depend on several factors, including the cost of development, the potential revenue from the application, and the market demand for such an application. This assesses the financial viability of the project, including an analysis of the costs and benefits of the project, its potential return on investment, and its potential impact on the organization's finances. Developers would need to conduct market research and evaluate the potential costs and revenue streams to determine the viability of developing a color theory application.

5.3 Operational Feasibility

The operational feasibility of coloration application depend on its ability to meet the needs and expectations of its users and stakeholders, and to function effectively in its intended operational environment. Developers need to conduct user research and testing to ensure that the application meets these requirements and is well-suited for its intended audience. This assesses that the project is operationally feasible, including an analysis of the project's impact on the organization's operations and the organization has the resources and expertise to manage the project.

5.4 Schedule Feasibility

The schedule feasibility of coloration application depends on the complexity of the project, the expertise and resources of the development team, and the development process used. It is essential to carefully plan and allocate sufficient time for development, testing, and debugging to ensure that the application can be launched within the desired timeline. This assesses that the project can be completed within the specified timeline and whether the project team has the necessary resources and expertise to complete the project on time.

CHAPTER 6

SYSTEM ANALYSIS

System analysis is a process of analysing, designing, and implementing complex systems. It involves studying a system's components, their interactions, and the system's environment to understand how the system works and identify areas where it can be improved. The process of system analysis includes

6.1 Existing System

In the existing system, users seek for a physical designer or a manual to know about the color combinations. And also users browse through internet to know more about colors. In which all of this method they get a minimal color combinations which is very widely popular or commonly used colors, which is profitable for the designers even if the users are not completely satisfied. So in this situation the users are not having any further references about the colors and get convinced themselves.

6.2 Proposed System

- Users activities can be tracked through the Admin.
- The concept of color theory can also be referred through this app.
- Application has a login system which makes the app portable.
- Search option will benefit the users to access all the colors.
- Users can create their own colors with their own complementary colors.
- Create colors has two modes i.e., custom mode and coloration mode.
- Favorites option will allow the users to save their desired colors.
- Users can also edit and save their profile settings.

6.3 System Description:

6.3.1 Modules

This software has 2 modules:

- Admin
- User

Admin module:

- Admin will be provided with the login ID and password in this application.
- The admin keeps the information as sensitive as possible.

- Admin can add appropriate details of colors such as color name, hexacode, RGB code, Color description, etc..

User module:

- The user has to login or create account.
- On login, the user will be redirected to the home page in which different color families will be shown. By selecting any of the color family, the user could see the list of colors under that particular color family selected.
- The user can add any colors to their favorites list.
- The user can create their own colors and save it in my colors option.

CHAPTER 7

DETAILED DESIGN

Detailed design is an essential phase of a project where a high-level project plan is transformed into a detailed technical plan. It involves the creation of detailed specifications, plans, and drawings that serve as the blueprint for the project. Detailed design is critical because it ensures that the project will be constructed or implemented accurately and efficiently, meeting all the specified requirements and objectives.

7.1 Data Flow Diagram

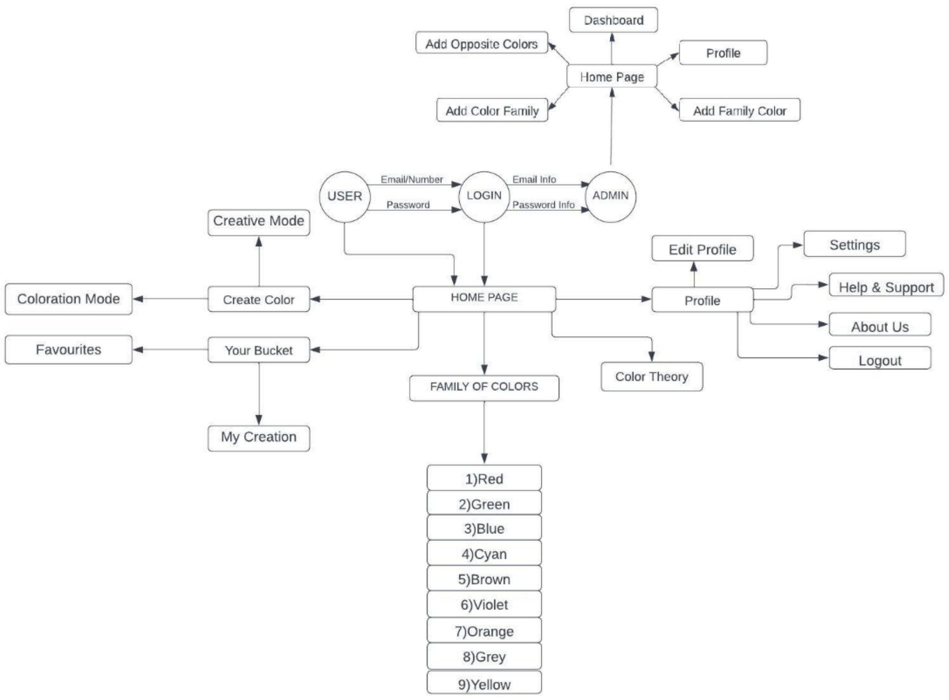


Fig.7.1.1 Data Flow Diagram

CHAPTER 8

SYSTEM DESIGN

8.1 The Architecture:

8.1.1 Output Design

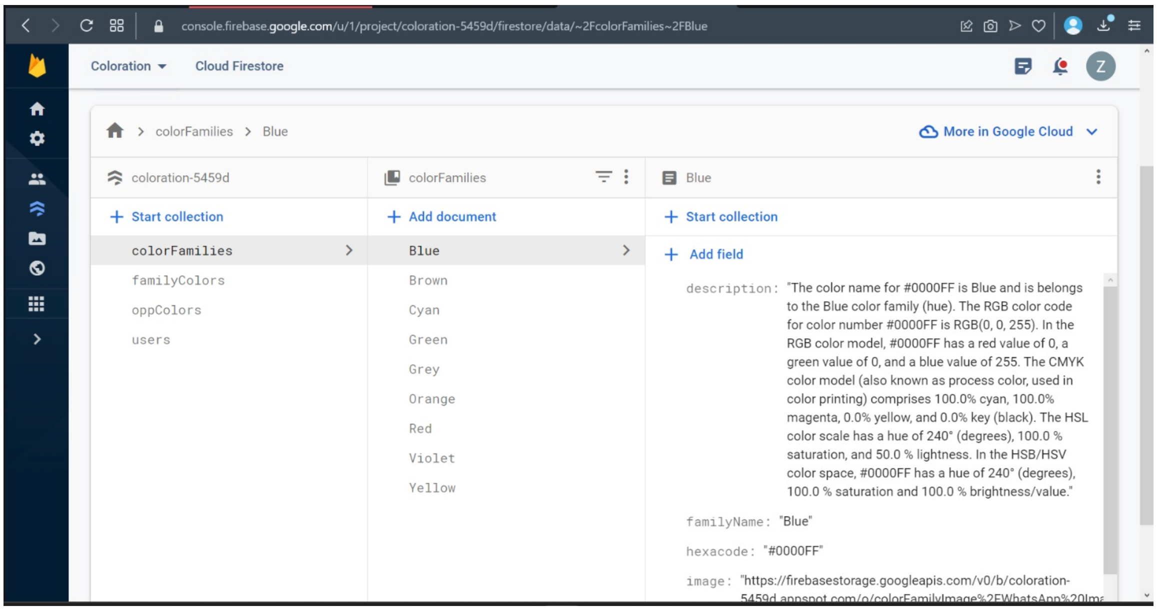
Output design generally refers to the results and information that are generated by the system for many end users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application. The output design should be clear, user-friendly, and visually appealing to ensure that the user can easily understand the information provided by the application.

8.1.2 Code Design

The code design for this application should be structured, modular, and well-documented to ensure that the application is scalable, maintainable, and easy to understand for future developers. The design should be consistent, with a clear separation of concerns and a consistent coding style, to make it easier to read and understand.

8.2 Database Table Design:

The general theme behind a database is to handle information as an integrated whole. After designing input and output, the analyst must concentrate on database design or how data should be organized around user requirements.



coloration-5459d	colorFamilies
colorFamilies	Blue
familyColors	Brown
oppColors	Cyan
users	Green
	Grey
	Orange
	Red
	Violet
	Yellow

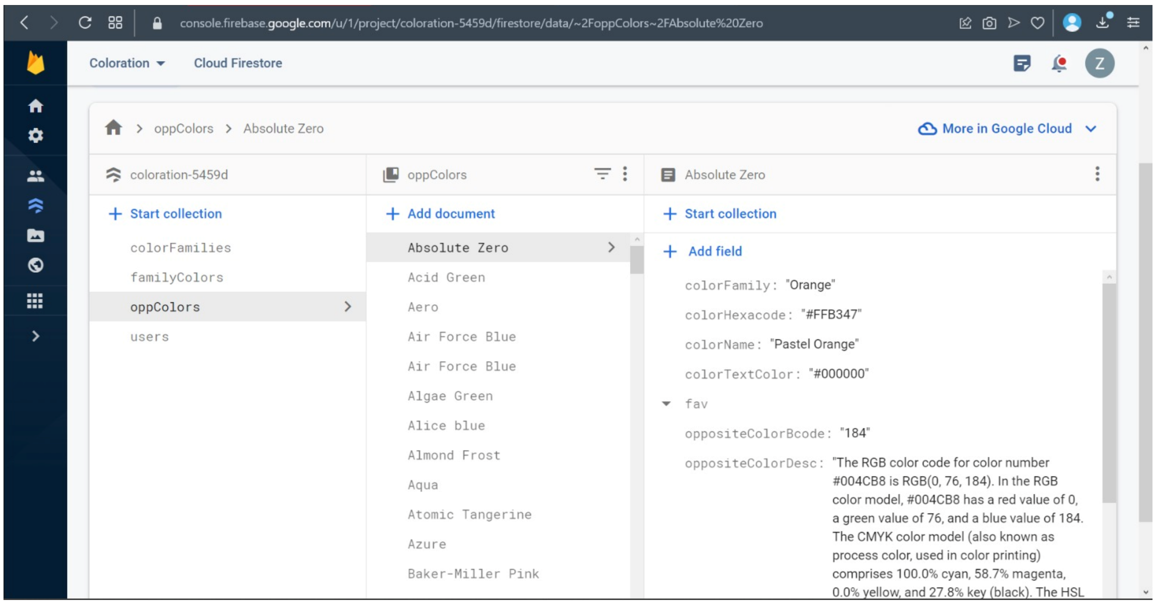
description: "The color name for #0000FF is Blue and is belongs to the Blue color family (hue). The RGB color code for color number #0000FF is RGB(0, 0, 255). In the RGB color model, #0000FF has a red value of 0, a green value of 0, and a blue value of 255. The CMYK color model (also known as process color, used in color printing) comprises 100.0% cyan, 100.0% magenta, 0.0% yellow, and 0.0% key (black). The HSL color scale has a hue of 240° (degrees), 100.0 % saturation, and 50.0 % lightness. In the HSB/HSV color space, #0000FF has a hue of 240° (degrees), 100.0 % saturation and 100.0 % brightness/value."

familyName: "Blue"

hexacode: "#0000FF"

image: "https://firebasestorage.googleapis.com/v0/b/coloration-5459d.appspot.com/o/colorFamilyImage%2FWhatsApp%20Ima

Fig 8.2.1 Color Families Table



console.firebase.google.com/u/1/project/coloration-5459d/firestore/data/~2FoppColors~2FAbsolute%20Zero

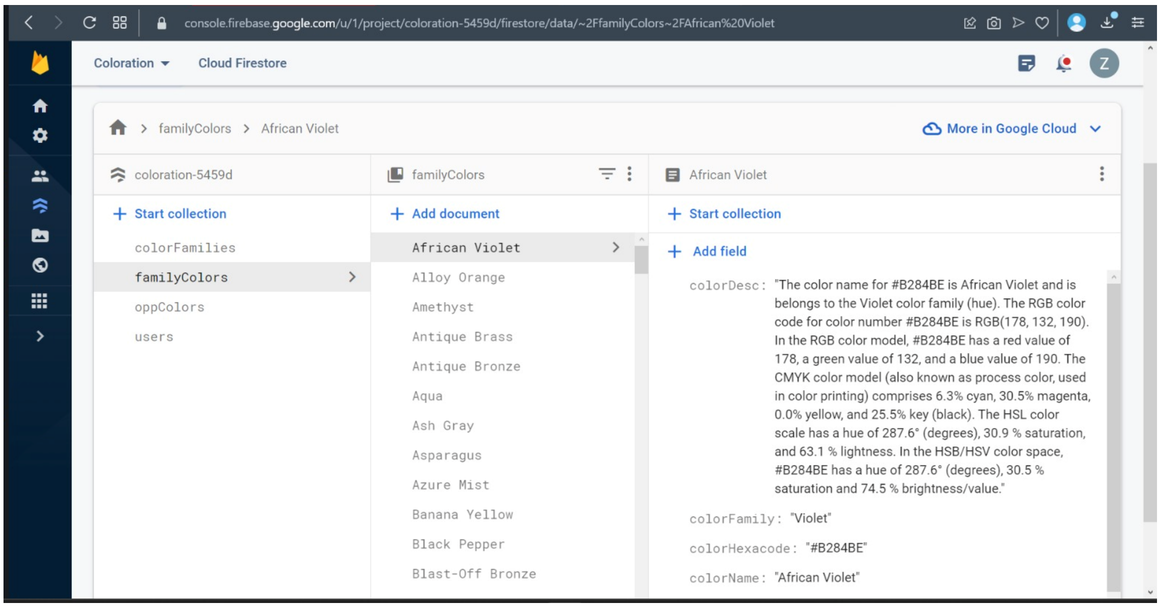
Coloration ▾ Cloud Firestore

> oppColors > Absolute Zero

More in Google Cloud

coloration-5459d	oppColors	Absolute Zero
+ Start collection	+ Add document	+ Start collection
colorFamilies	Absolute Zero	+ Add field
familyColors	Acid Green	colorFamily: "Orange"
oppColors	Aero	colorHexacode: "#FFB347"
users	Air Force Blue	colorName: "Pastel Orange"
	Air Force Blue	colorTextColor: "#000000"
	Algae Green	fav
	Alice blue	oppositeColorBcode: "184"
	Almond Frost	oppositeColorDesc: "The RGB color code for color number #004CB8 is RGB(0, 76, 184). In the RGB color model, #004CB8 has a red value of 0, a green value of 76, and a blue value of 184. The CMYK color model (also known as process color, used in color printing) comprises 100.0% cyan, 58.7% magenta, 0.0% yellow, and 27.8% key (black). The HSL
	Aqua	
	Atomic Tangerine	
	Azure	
	Baker-Miller Pink	

Fig 8.2.2 Opposite colors Table



console.firebase.google.com/u/1/project/coloration-5459d/firestore/data/~2FfamilyColors~2FAfrican%20Violet

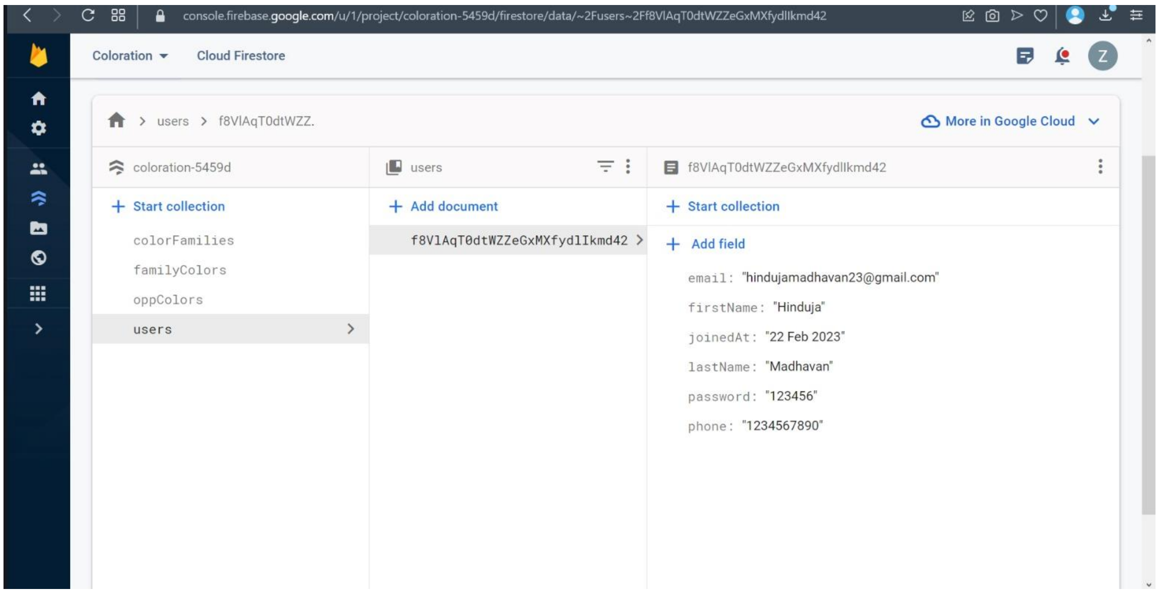
Coloration ▾ Cloud Firestore

> familyColors > African Violet

More in Google Cloud

coloration-5459d	familyColors	African Violet
+ Start collection	+ Add document	+ Start collection
colorFamilies	African Violet	+ Add field
familyColors	Alloy Orange	colorDesc: "The color name for #B284BE is African Violet and is belongs to the Violet color family (hue). The RGB color code for color number #B284BE is RGB(178, 132, 190). In the RGB color model, #B284BE has a red value of 178, a green value of 132, and a blue value of 190. The CMYK color model (also known as process color, used in color printing) comprises 6.3% cyan, 30.5% magenta, 0.0% yellow, and 25.5% key (black). The HSL color scale has a hue of 287.6° (degrees), 30.9 % saturation, and 63.1 % lightness. In the HSB/HSV color space, #B284BE has a hue of 287.6° (degrees), 30.5 % saturation and 74.5 % brightness/value."
oppColors	Amethyst	colorFamily: "Violet"
users	Antique Brass	colorHexacode: "#B284BE"
	Antique Bronze	colorName: "African Violet"
	Aqua	
	Ash Gray	
	Asparagus	
	Azure Mist	
	Banana Yellow	
	Black Pepper	
	Blast-Off Bronze	

Fig 8.2.3 Family Colors Table



```
email: "hindujamadhavan23@gmail.com"
firstName: "Hinduja"
joinedAt: "22 Feb 2023"
lastName: "Madhavan"
password: "123456"
phone: "1234567890"
```

Fig 8.2.4 Users Table

CHAPTER 9

DETAILED DESIGN

9.1 Use Case Diagram

A use case diagram is a type of diagram used in software development to visually represent the interactions between actors (users or external systems) and the system being developed. It shows the functionality provided by the system, as well as the interactions between the system and its users.

A use case diagram typically consists of use cases, actors, and the relationships between them. A use case is a specific task or goal that a user wants to accomplish using the system. An actor is any entity outside the system that interacts with the system. The relationships between actors and use cases indicate which actors interact with which use cases

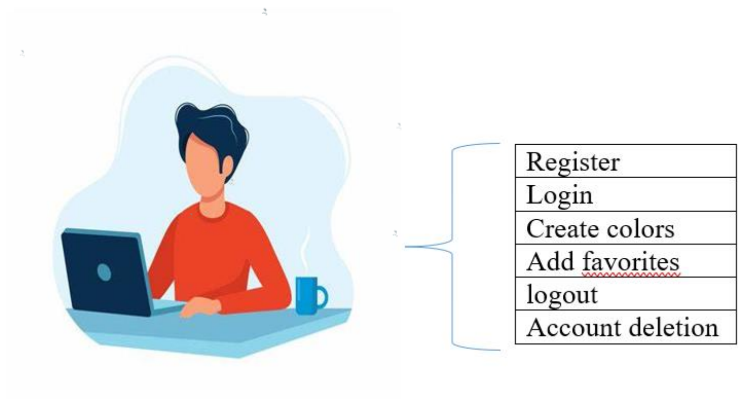


Fig.9.1.1 User Use Case

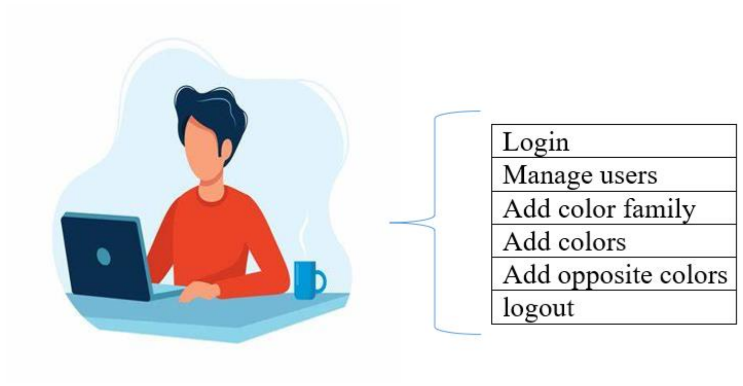


Fig 8.1.2 Admin Use Case

Fig.9.1.2 Admin Use Case

CHAPTER 10

IMPLEMENTATION

10.1 Introduction

Implementation is the process of turning the design and plans of a software project into a working and functional system. It involves the actual writing of code and the creation of all the necessary components, such as user interfaces, databases, algorithms, and other software artifacts. The implementation phase is a critical stage in the software development life cycle, as it involves actually creating a working system that can be used by end-users. It requires careful planning, attention to detail, and collaboration between developers and other stakeholders to ensure that the software meets the requirements and functions as intended.

10.2 Coding

MAIN.DART

```
import 'package:colorations_admin/Source/Screens/splash_screen.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_easyloading/flutter_easyloading.dart';
import 'package:get/get.dart';
import 'Source/Helpers/routes.dart';

void main() async {
  const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyAmsyD_tipbjHsTvFqM64pGtwUqZBanfl8',
    appId: '1:855954393160:web:f4b6a25524660f1e11b07d',
    messagingSenderId: '855954393160',
    projectId: 'coloration-5459d',
    authDomain: 'coloration-5459d.firebaseapp.com',
    storageBucket: 'coloration-5459d.appspot.com',
  );
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options: web);
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      builder: EasyLoading.init(),
      debugShowCheckedModeBanner: false,
      title: 'Coloration Admin',
      routes: routes,
      theme: GAppTheme.lightTheme,
      //home: const SplashScreen(),
      home: const SplashScreen(),
    );
  }
}

class GAppTheme {
  GAppTheme._();
  static ThemeData lightTheme = ThemeData(
    brightness: Brightness.dark,
    primarySwatch: Colors.indigo,
    inputDecorationTheme: const InputDecorationTheme(
      contentPadding: EdgeInsets.all(10),
      border: OutlineInputBorder(),
      floatingLabelStyle: TextStyle(color: Colors.white70),
      focusedBorder: OutlineInputBorder(
        borderSide: BorderSide(width: 2.0, color: Colors.white70),
      ),
    ),
    elevatedButtonTheme: ElevatedButtonThemeData(
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.indigo,
        foregroundColor: Colors.white70,
        side: const BorderSide(color: Colors.indigo),
        padding: const EdgeInsets.symmetric(vertical: 15.0),
      ),
    ),
  );
}
```

LOGIN SCREEN

```
import 'package:colorations_admin/Source/Entry%20Point/entry_point.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_easyloading/flutter_easyloading.dart';
import 'package:get/get.dart';
import '../../Helpers/responsive_widget.dart';
import '../../Repository/shared_preference.dart';
import '../../Themes/colors.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key key}) : super(key: key);

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}
```

```
class _LoginScreenState extends State<LoginScreen> {
  // Variables
  final TextEditingController _email = TextEditingController();
  final TextEditingController _pass = TextEditingController();
  final PrefService _prefService = PrefService();
  final _formKey = GlobalKey<FormState>();

  // function
  void signInAdmin() async {
    EasyLoading.show();

    try {
      await FirebaseAuth.instance
          .signInWithEmailAndPassword(
        email: _email.text.toString().trim(),
        password: _pass.text.toString().trim(),
      )
          .then((value) {
        EasyLoading.dismiss();
        Get.offAll(() => const EntryPoint());
      });
    } on FirebaseAuthException catch (e) {
      if (e.code == "user-not-found") {
        EasyLoading.showToast("No User found on this Email",
            toastPosition: EasyLoadingToastPosition.center);
      }
      EasyLoading.showToast(e.toString(),
          toastPosition: EasyLoadingToastPosition.center);
    }
```

```
}

Widget createDialog(BuildContext context) {
  return CupertinoAlertDialog(
    title: Text(
      "Contact Lavanya Sai, for Resetting Password",
      style: TextStyle(fontSize: 18),
    ),
    content: Text(
      "Is this helpfull?",
      style: TextStyle(fontSize: 16),
    ),
    actions: [
      CupertinoDialogAction(
        child: Text("Yes"),
        onPressed: () => Navigator.pop(context),
      ),
      CupertinoDialogAction(
        child: Text("No"),
        onPressed: () => {
          EasyLoading.showToast("Thank You!"),
          Navigator.pop(context),
        },
      ),
    ],
  );
}

@override
Widget build(BuildContext context) {
  double height = MediaQuery.of(context).size.height;
  double width = MediaQuery.of(context).size.width;
  return Scaffold(
   backgroundColor: AppColors.backColor,
   body: SizedBox(
    height: height,
    width: width,
    child: Row(
     crossAxisAlignment: CrossAxisAlignment.start,
     mainAxisAlignment: MainAxisAlignment.start,
     children: [
      ResponsiveWidget.isSmallScreen(context)
        ? const SizedBox()
        : Expanded(
          child: Container(
           height: height,
           color: AppColors.mainBlueColor,
           child: const Center(
            child: Text('Coloration - Admin',
              style: TextStyle(
               fontSize: 48.0,
               color: AppColors.whiteColor,
               fontWeight: FontWeight.w800,
              )),
           ),
          ),
         ),
      Expanded(
       child: Container(
        height: height,
margin: EdgeInsets.symmetric(
   horizontal: ResponsiveWidget.isSmallScreen(context)
       ? height * 0.032
       : height * 0.12),
color: AppColors.backColor,
child: SingleChildScrollView(
  padding: const EdgeInsets.only(bottom: 40.0),
  child: Form(
   key: _formKey,
   child: Column(
     crossAxisAlignment: CrossAxisAlignment.start,
     mainAxisAlignment: MainAxisAlignment.start,
     children: [
      SizedBox(height: height * 0.2),
      RichText(
        text: const TextSpan(
         children: [
          TextSpan(
              text: "Let's",
              style: TextStyle(
                fontSize: 25.0,
                color: AppColors.blueDarkColor,
                fontWeight: FontWeight.normal,
              )),
          TextSpan(
              text: ' Login',
              style: TextStyle(
                fontWeight: FontWeight.w800,
                color: AppColors.blueDarkColor,
                fontSize: 25.0,
          )),
      ],
    ),
  ),
  SizedBox(height: height * 0.02),
  const Text(
    'Hey, Enter your details to get Login.',
    style: TextStyle(
      fontSize: 12.0,
      fontWeight: FontWeight.w400,
      color: AppColors.textColor,
    ),
  ),
  SizedBox(height: height * 0.064),
  const Padding(
    padding: EdgeInsets.only(left: 16.0),
    child: Text('Email',
        style: TextStyle(
          fontSize: 12.0,
          color: AppColors.blueDarkColor,
          fontWeight: FontWeight.w700,
        )),
  ),
  const SizedBox(height: 6.0),
  Container(
    height: 50.0,
    width: width,
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(16.0),
      color: AppColors.whiteColor,
    ),
    child: TextFormField(
      controller: _email,
      textInputAction: TextInputAction.next,
      validator: (value) {
        if (value.isEmpty) {
          return "Email is required";
        }
        return null;
      },
      style: const TextStyle(
        fontWeight: FontWeight.w400,
        color: AppColors.blueDarkColor,
        fontSize: 12.0,
      ),
      decoration: InputDecoration(
        border: InputBorder.none,
        hoverColor: Colors.black,
        prefixIcon:  Icon(Icons.email,color:
  AppColors.blueDarkColor.withOpacity(0.9)),
        contentPadding: const EdgeInsets.only(top: 16.0),
        hintText: 'Enter Email',
        hintStyle: TextStyle(
          fontWeight: FontWeight.w400,
          color: AppColors.blueDarkColor.withOpacity(0.5),
          fontSize: 12.0,
        ),
      ),
    ),
  ),
  SizedBox(height: height * 0.014),
  const Padding(
    padding: EdgeInsets.only(left: 16.0),
    child: Text('Password',
        style: TextStyle(
          fontSize: 12.0,
          color: AppColors.blueDarkColor,
          fontWeight: FontWeight.w700,
        )),
  ),
  const SizedBox(height: 6.0),
  Container(
    height: 50.0,
    width: width,
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(16.0),
      color: AppColors.whiteColor,
    ),
    child: TextFormField(
      controller: _pass,
      textInputAction: TextInputAction.done,
      validator: (value) {
        if (value.isEmpty) {
          return "Password is required";
        }
        return null;
      },
      style: const TextStyle(
        fontWeight: FontWeight.w400,
        color: AppColors.blueDarkColor,
        fontSize: 12.0,
              ),
              obscureText: true,
              decoration: InputDecoration(
                border: InputBorder.none,
                hoverColor: Colors.black,
                prefixIcon:  Icon(Icons.lock,color:
AppColors.blueDarkColor.withOpacity(0.9),),
                contentPadding:
                    const EdgeInsets.only(top: 16.0),
                hintText: 'Enter Password',
                hintStyle: TextStyle(
                  fontWeight: FontWeight.w400,
                  color:
                      AppColors.blueDarkColor.withOpacity(0.5),
                  fontSize: 12.0,
                )),
          ),
        ),
        SizedBox(height: height * 0.03),
        Align(
          alignment: Alignment.centerRight,
          child: TextButton(
            onPressed: () {
              showCupertinoDialog(
                context: context,
                builder: createDialog,
              );
            },
            child: const Text(
              'Forgot Password?',
              style: TextStyle(
                fontSize: 12.0,
                color: AppColors.mainBlueColor,
                fontWeight: FontWeight.w600,
              ),
            ),
          ),
        ),
        SizedBox(height: height * 0.05),
        Material(
          color: Colors.transparent,
          child: InkWell(
            onTap: () async {
              if (_formKey.currentState.validate()) {
                _prefService
                    .createCache(_pass.text)
                    .whenComplete(() => signInAdmin());
              }
            },
            borderRadius: BorderRadius.circular(16.0),
            child: Ink(
              padding: const EdgeInsets.symmetric(
                  horizontal: 70.0, vertical: 18.0),
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(16.0),
                color: AppColors.mainBlueColor,
              ),
              child: const Text(
                'Sign In',
                style: TextStyle(
                  fontWeight: FontWeight.w700,
                                  color: AppColors.whiteColor,
                                  fontSize: 16.0,
                                ),
                              ),
                            ),
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

class LogOut extends StatefulWidget {
  const LogOut({Key key}) : super(key: key);

  @override
  State<LogOut> createState() => _LogOutState();
}

class _LogOutState extends State<LogOut> {
  final PrefService _prefService = PrefService();
  @override
  void initState() async {
    _prefService.removeCache('password').whenComplete(() => FirebaseAuth
        .instance
        .signOut()
        .whenComplete(() => Get.offAll(() => const LoginScreen())));
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

COLOR FAMILY

```
import 'package:colorations_admin/Source/Constants/constants.dart';
import 'package:colorations_admin/Source/Helpers/user_model.dart';
import 'package:flutter/material.dart';
import 'package:flutter_easyloading/flutter_easyloading.dart';
import 'package:google_fonts/google_fonts.dart';

import '../../Repository/Colors Repository/firebase_services.dart';

class ColorFamilyScreen extends StatefulWidget {
  const ColorFamilyScreen({Key key}) : super(key: key);

  @override
  State<ColorFamilyScreen> createState() => _ColorFamilyScreenState();
}
```

```
class _ColorFamilyScreenState extends State<ColorFamilyScreen> {
  final TextEditingController _familyName = TextEditingController();
  final TextEditingController _desc = TextEditingController();
  final TextEditingController _hexaCode = TextEditingController();
  final TextEditingController _rgbCode = TextEditingController();
  final _formKey = GlobalKey<FormState>();

  // Calling Classes
  final FirebaseService _service = FirebaseService();

  clear() {
    setState(() {
      _familyName.clear();
      _desc.clear();
      _hexaCode.clear();
      _rgbCode.clear();
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Add Color Family"),
        backgroundColor: Colors.transparent,
        centerTitle: true,
        elevation: 0,
        actions: [
          IconButton(onPressed: clear, icon: Icon(Icons.clear,size: 25))
        ],
      ),
      backgroundColor: bgColor,

      //
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Container(
                padding: EdgeInsets.only(left: 25, right: 25, top: 20),
                child: Form(
                  key: _formKey,
                  child: Column(
                    children: [
                      // form Starts
                      TextFormField(
                        keyboardType: TextInputType.text,
                        controller: _familyName,
                        validator: (value) {
                          if (value.isEmpty) {
                            return "Family Name is required";
                          }
                        },
                        textInputAction: TextInputAction.next,
                        decoration: InputDecoration(label: Text("Family Name")),
                      ),
                      const SizedBox(height: 20),

                      TextFormField(
                        keyboardType: TextInputType.text,
                        controller: _hexaCode,
                        validator: (value) {
                          if (value.isEmpty) {
                            return "Hexacode is required";
                          }
                        },
                        textInputAction: TextInputAction.next,
                        decoration: InputDecoration(label: Text("Hexa Code")),
                      ),

                      const SizedBox(height: 20),

                      // email textfield
                      TextFormField(
                        keyboardType: TextInputType.number,
                        controller: _rgbCode,
                        validator: (value) {
                          if (value.isEmpty) {
                            return "RGB Value is required";
                          }
                        },
                        textInputAction: TextInputAction.next,
                        decoration: InputDecoration(label: Text("RGB Values")),
                      ),

                      const SizedBox(height: 20),

                      // phone number textfield
                      TextFormField(
                        keyboardType: TextInputType.multiline,
                        minLines: 1,
                        maxLines: 8,
                        controller: _desc,
                        validator: (value) {
                          if (value.isEmpty) {
                            return "Description is required";
                          }
                        },
                        textInputAction: TextInputAction.done,
                        decoration: InputDecoration(label: Text("Description")),
                      ),

                      const SizedBox(height: 25),

                      // sign in button
                      SizedBox(
                        width: double.infinity,
                        child: ElevatedButton(
                          onPressed: () {
                            if (_formKey.currentState.validate()) {
                              //
                              EasyLoading.show();
                              _service.saveColorFamily(
                                data: {
                                  'familyName': _familyName.text.firstCaps(),
                                  'description': _desc.text.firstCaps(),
                                  'hexacode': _hexaCode.text.toUpperCase(),
                                  'rgbCode': _rgbCode.text,
                                },
                                reference: _service.colorFamilies,
                                docName: _familyName.text.firstCaps(),
                              ).then((value) {
                                clear();
                                EasyLoading.showSuccess("Added Successfully");
                                EasyLoading.dismiss();
                              });
                            }
                          },
                          child: Text(
                            "Add Family".toUpperCase(),
                            style: GoogleFonts.poppins(
                                fontSize: 16.0,
                                fontWeight: FontWeight.w600,
                                color: Colors.white),
                          ),
                        ),
                      ),
                    ],
                  ),
                ),
              ),
             ],
            ),
            ),
            ),
            );
            }
            }
```

FAMILY COLORS

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:colorations_admin/Source/Helpers/user_model.dart';
import 'package:flutter/material.dart';
import 'package:flutter_easyloading/flutter_easyloading.dart';
import 'package:google_fonts/google_fonts.dart';

import '../../Constants/constants.dart';
import '../../Repository/Colors Repository/firebase_services.dart';

class FamilyColorsScreen extends StatefulWidget {
  const FamilyColorsScreen({Key key}) : super(key: key);

  @override
  State<FamilyColorsScreen> createState() => _FamilyColorsScreenState();
}

class _FamilyColorsScreenState extends State<FamilyColorsScreen> {
  // calling service
  final FirebaseService service = FirebaseService();
  Object _selectedValue;
  bool noFamilyNameSelected = false;
  final _formKey = GlobalKey<FormState>();
  QuerySnapshot snapshot;

  final TextEditingController _colorName = TextEditingController();
  final TextEditingController _fore = TextEditingController();
  final TextEditingController _desc = TextEditingController();
  final TextEditingController _hexaCode = TextEditingController();
  final TextEditingController _rgbCode = TextEditingController();
```

```
//Widget
Widget dropDownButton() {
  return DropdownButton(
    value: _selectedValue,
    hint: Text("Select Color Family"),
    items: snapshot.docs.map((e) {
      return DropdownMenuItem<String>(
        value: e['familyName'],
        child: Text(e['familyName']),
      );
    }).toList(),
    onChanged: (selectedCat) {
      setState(() {
        _selectedValue = selectedCat;
        noFamilyNameSelected = false;
      });
    },
  );
}

clear() {
  setState(() {
    _selectedValue = null;
    _colorName.clear();
    _desc.clear();
    _hexaCode.clear();
    _rgbCode.clear();
    _fore.clear();
  });
}
```

```
@override
void initState() {
  getColorFamiliesList();
  super.initState();
}

getColorFamiliesList() {
  return service.colorFamilies.get().then((QuerySnapshot querySnapshot) {
    setState(() {
      snapshot = querySnapshot;
    });
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Add Family Color"),
      backgroundColor: Colors.transparent,
      centerTitle: true,
      elevation: 0,
      actions: [
        IconButton(onPressed: clear, icon: Icon(Icons.clear, size: 25))
      ],
    ),
    backgroundColor: bgColor,

    //
```

```
body: SafeArea(
  child: SingleChildScrollView(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Container(
          padding: EdgeInsets.only(left: 25, right: 25, top: 20),
          child: Form(
            key: _formKey,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                // form Starts
                snapshot == null
                    ? Text("Loading...")
                    : SizedBox(
                        width: double.infinity,
                        child: dropDownButton(),
                      ),

                SizedBox(height: 8),
                if (noFamilyNameSelected == true)
                  Text(
                    "No Family Name Selected",
                    style: TextStyle(color: Colors.red),
                  ),
                SizedBox(height: 20),
                //
                TextFormField(
                  keyboardType: TextInputType.text,
                  controller: _colorName,
                  validator: (value) {
                    if (value.isEmpty) {
                      return "Color Name is required";
                    }
                  },
                  textInputAction: TextInputAction.next,
                  decoration: InputDecoration(label: Text("Color Name")),
                ),
                const SizedBox(height: 20),

                TextFormField(
                  keyboardType: TextInputType.text,
                  controller: _hexaCode,
                  validator: (value) {
                    if (value.isEmpty) {
                      return "Hexacode is required";
                    }
                  },
                  textInputAction: TextInputAction.next,
                  decoration: InputDecoration(label: Text("Hexa Code")),
                ),

                const SizedBox(height: 20),

                // email textfield
                TextFormField(
                  keyboardType: TextInputType.number,
                  controller: _rgbCode,
                  validator: (value) {
                    if (value.isEmpty) {
                      return "RGB Value is required";
                    }
                  },
                  textInputAction: TextInputAction.next,
                  decoration: InputDecoration(label: Text("RGB Values")),
                ),

                const SizedBox(height: 20),

                // email textfield
                TextFormField(
                  keyboardType: TextInputType.text,
                  controller: _fore,
                  validator: (value) {
                    if (value.isEmpty) {
                      return "Font Color is required";
                    }
                  },
                  textInputAction: TextInputAction.next,
                  decoration: InputDecoration(label: Text("Font Color")),
                ),

                const SizedBox(height: 20),

                // phone number textfield
                TextFormField(
                  keyboardType: TextInputType.multiline,
                  minLines: 1,
                  maxLines: 8,
                  controller: _desc,
                  validator: (value) {
                    if (value.isEmpty) {
                      return "Description is required";
                    }
                  },
                  textInputAction: TextInputAction.done,
                  decoration: InputDecoration(label: Text("Description")),
                ),

                const SizedBox(height: 25),

                // sign in button
                SizedBox(
                  width: double.infinity,
                  child: ElevatedButton(
                    onPressed: () {
                      if (_selectedValue == null) {
                        setState(() {
                          noFamilyNameSelected = true;
                        });
                        return;
                      }
                      if (_formKey.currentState.validate()) {
                        //
                        EasyLoading.show();
                        service.saveColorFamily(
                          data: {
                            'colorFamily': _selectedValue,
                            'colorName': _colorName.text.firstCaps(),
                                 'colorDesc': _desc.text.firstCaps(),
                                 'colorHexacode': _hexaCode.text.toUpperCase(),
                                 'colorRGBcode': _rgbCode.text,
                                 'textColor': _fore.text.toUpperCase(),
                                 'fav': [],
                                },
                                reference: service.familyColors,
                                docName: _colorName.text.firstCaps(),
                               ).then((value) {
                                clear();
                                EasyLoading.showSuccess("Added Successfully");
                                EasyLoading.dismiss();
                               });
                              }
                             },
                             child: Text(
                              "Add Color".toUpperCase(),
                              style: GoogleFonts.poppins(
                                fontSize: 16.0,
                                fontWeight: FontWeight.w600,
                                color: Colors.white),
                             ),
                            ),
                           ),
                          ],
                         ),
                        ),
                       ),
                      ],
                     ),
                        ),
                      ),
                    );
                  }
                }
```

OPPOSITE COLORS

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:colorations_admin/Source/Helpers/user_model.dart';
import 'package:flutter/material.dart';
import 'package:flutter_easyloading/flutter_easyloading.dart';
import 'package:google_fonts/google_fonts.dart';

import '../../Constants/constants.dart';
import '../../Repository/Colors Repository/firebase_services.dart';

class OppositeColorsScreen extends StatefulWidget {
  const OppositeColorsScreen({Key key}) : super(key: key);

  @override
  State<OppositeColorsScreen> createState() => _OppositeColorsScreenState();
}

class _OppositeColorsScreenState extends State<OppositeColorsScreen> {
  //
  final FirebaseService service = FirebaseService();
  Object _selectedValue;
  Object _colorFamilySelectedValue;
  bool noColorNameSelected = false;
  bool noFamilyNameSelected = false;
  final _formKey = GlobalKey<FormState>();
```

```
QuerySnapshot snapshot, colorFamilySnapshot;
String hexacode, textcolor;

final TextEditingController _colorName = TextEditingController();
final TextEditingController _fore = TextEditingController();
final TextEditingController _desc = TextEditingController();
final TextEditingController _hexaCode = TextEditingController();
final TextEditingController _rCode = TextEditingController();
final TextEditingController _gCode = TextEditingController();
final TextEditingController _bCode = TextEditingController();

//Widget
Widget dropDownButton() {
  return DropdownButton(
    value: _selectedValue,
    hint: Text("Select Color"),
    items: snapshot.docs.map((e) {
      return DropdownMenuItem<String>(
        value: e['colorName'],
        child: Text(e['colorName']),
      );
    }).toList(),
    onChanged: (selectedCat) {
      setState(() {
        _selectedValue = selectedCat;
        noColorNameSelected = false;
      });
      getFamilyColorHexaCode();
    },
  );
```

```
}

Widget colorFamilyDropDownButton() {
  return DropdownButton(
    value: _colorFamilySelectedValue,
    hint: Text("Select Color Family"),
    items: colorFamilySnapshot.docs.map((e) {
      return DropdownMenuItem<String>(
        value: e['familyName'],
        child: Text(e['familyName']),
      );
    }).toList(),
    onChanged: (selectedCat) {
      setState(() {
        _colorFamilySelectedValue = selectedCat;
        noFamilyNameSelected = false;
      });
      getFamilyColorsList();
    },
  );
}

clear() {
  setState(() {
    _selectedValue = null;
    _colorFamilySelectedValue = null;
    _colorName.clear();
    _desc.clear();
    _hexaCode.clear();
    _rCode.clear();
```

```
      _gCode.clear();
      _bCode.clear();
      _fore.clear();
    });
  }

  @override
  void initState() {
    getColorFamilyList();
    super.initState();
  }

  getFamilyColorsList() {
    return service.familyColors
        .where("colorFamily", isEqualTo: _colorFamilySelectedValue)
        .get()
        .then((QuerySnapshot querySnapshot) {
      setState(() {
        snapshot = querySnapshot;
      });
    });
  }

  getFamilyColorHexaCode() {
    return service.familyColors
        .doc(_selectedValue.toString())
        .get()
        .then((value) {
      setState(() {
        hexacode = value['colorHexacode'];
        textcolor = value['textColor'];
      });
    });
  }

  getColorFamilyList() {
    return service.colorFamilies.get().then((QuerySnapshot querySnapshot) {
      setState(() {
        colorFamilySnapshot = querySnapshot;
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Add Opposite Color"),
        backgroundColor: Colors.transparent,
        centerTitle: true,
        elevation: 0,
        actions: [
          IconButton(onPressed: clear, icon: Icon(Icons.clear, size: 25))
        ],
      ),
      backgroundColor: bgColor,

      //
      body: SafeArea(
        child: SingleChildScrollView(
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Container(
        padding: EdgeInsets.only(left: 25, right: 25, top: 0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // form Starts
              colorFamilySnapshot == null
                  ? Text("Loading...")
                  : SizedBox(
                      width: double.infinity,
                      child: colorFamilyDropDownButton()),
              SizedBox(height: 8),
              if (noFamilyNameSelected == true)
                Text(
                  "No Color Family Selected",
                  style: TextStyle(color: Colors.red),
                ),
              Padding(
                padding: const EdgeInsets.only(top: 8.0),
                child: Column(
                  children: [
                    snapshot == null
                        ? Text("")
                        : SizedBox(
                            width: double.infinity,
                            child: dropDownButton(),
                          ),
                    SizedBox(height: 8),
                    if (noColorNameSelected == true)
                      Text(
                        "No Color Selected",
                        textAlign: TextAlign.left,
                        style: TextStyle(color: Colors.red),
                      ),
                    SizedBox(height: 12),
                  ],
                ),
              ),
              //
              TextFormField(
                keyboardType: TextInputType.text,
                controller: _colorName,
                validator: (value) {
                  if (value.isEmpty) {
                    return "Color Name is required";
                  }
                },
                textInputAction: TextInputAction.next,
                decoration: InputDecoration(label: Text("Color Name")),
              ),
              const SizedBox(height: 15),

              TextFormField(
                keyboardType: TextInputType.text,
                controller: _hexaCode,
                validator: (value) {
                  if (value.isEmpty) {
                    return "Hexacode is required";
                  }
                },
                textInputAction: TextInputAction.next,
                decoration: InputDecoration(label: Text("Hexa Code")),
              ),

              const SizedBox(height: 15),

              // email textfield
              TextFormField(
                keyboardType: TextInputType.number,
                controller: _rCode,
                validator: (value) {
                  if (value.isEmpty) {
                    return "RGB Value is required";
                  }
                },
                textInputAction: TextInputAction.next,
                decoration:
                    InputDecoration(label: Text("RGB (R Value)")),
              ),

              const SizedBox(height: 15),

              // email textfield
              TextFormField(
                keyboardType: TextInputType.number,
                controller: _gCode,
                validator: (value) {
                  if (value.isEmpty) {
                    return "RGB Value is required";
                  }
                },
                textInputAction: TextInputAction.next,
                decoration:
                    InputDecoration(label: Text("RGB (G Value)")),
              ),

              const SizedBox(height: 15),

              // email textfield
              TextFormField(
                keyboardType: TextInputType.number,
                controller: _bCode,
                validator: (value) {
                  if (value.isEmpty) {
                    return "RGB Value is required";
                  }
                },
                textInputAction: TextInputAction.next,
                decoration:
                    InputDecoration(label: Text("RGB (B Value)")),
              ),

              const SizedBox(height: 15),

              // email textfield
```

```
TextFormField(
  keyboardType: TextInputType.text,
  controller: _fore,
  validator: (value) {
    if (value.isEmpty) {
      return "Font Color is required";
    }
  },
  textInputAction: TextInputAction.next,
  decoration: InputDecoration(label: Text("Font Color")),
),

const SizedBox(height: 15),

// phone number textfield
TextFormField(
  keyboardType: TextInputType.multiline,
  minLines: 1,
  maxLines: 8,
  controller: _desc,
  validator: (value) {
    if (value.isEmpty) {
      return "Description is required";
    }
  },
  textInputAction: TextInputAction.done,
  decoration: InputDecoration(label: Text("Description")),
),

const SizedBox(height: 15),
// sign in button
SizedBox(
  width: double.infinity,
  child: ElevatedButton(
    onPressed: () {
      if (_selectedValue == null) {
        setState(() {
          noColorNameSelected = true;
          noFamilyNameSelected = true;
        });
        return;
      }
      if (_formKey.currentState.validate()) {
        EasyLoading.show();
        service.saveColorFamily(
          data: {
            'colorName': _selectedValue,
            'colorHexacode': hexacode.toUpperCase(),
            'colorTextColor': textcolor.toUpperCase(),
            'colorFamily': _colorFamilySelectedValue,
            'oppositeColorName': _colorName.text.firstCaps(),
            'oppositeColorDesc': _desc.text.firstCaps(),
            'oppositeColorHexacode':
                _hexaCode.text.toUpperCase(),
            'oppositeColorRcode': _rCode.text,
            'oppositeColorGcode': _gCode.text,
            'oppositeColorBcode': _bCode.text,
            'textColor': _fore.text.toUpperCase(),
            'fav': [],
                  },
                  reference: service.oppColors,
                  docName: _colorName.text.firstCaps(),
                ).then((value) {
                  clear();
                  EasyLoading.showSuccess("Added Successfully");
                  EasyLoading.dismiss();
                });
              }
            },
            child: Text(
              "Add Color".toUpperCase(),
              style: GoogleFonts.poppins(
                  fontSize: 16.0,
                  fontWeight: FontWeight.w600,
                  color: Colors.white),
            ),
          ),
        ),
      ],
    ),
   ),
  ),
 ],
),
),
),
);
}
}
```

COLORS LIST

```
import 'package:cloud_firestore/cloud_firestore.dart';
import
'package:colorations_admin/Source/Repository/Colors%20Repository/firebase_services.dar
t';
import 'package:flutter/material.dart';
import 'package:flutter/src/foundation/key.dart';
import 'package:flutter/src/widgets/container.dart';
import 'package:flutter/src/widgets/framework.dart';
import 'package:get/get.dart';

import '../../../Constants/constants.dart';
import '../../../Helpers/custom_tile.dart';
import '../../../Helpers/user_model.dart';

class AllColorsList extends StatefulWidget {
  const AllColorsList({Key key}) : super(key: key);

  @override
  State<AllColorsList> createState() => _AllColorsListState();
}

class _AllColorsListState extends State<AllColorsList> {
  final FirebaseService service = FirebaseService();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: bgColor,
      appBar: AppBar(
        title: Text("All Family Colors"),
        centerTitle: true,
        leading: IconButton(
          onPressed: () => Get.back(),
          icon: Icon(Icons.arrow_back_ios, color: Colors.white70),
        ),
        elevation: 0,
        backgroundColor: Colors.transparent,
      ),

      // body
      body: SingleChildScrollView(
        child: Container(
          padding: const EdgeInsets.only(left: 18, top: 6, right: 18),
          child: StreamBuilder<QuerySnapshot>(
            stream: service.familyColors.snapshots(),
            builder: (context, AsyncSnapshot<QuerySnapshot> snapshots) {
              if (snapshots.hasError) {
                Center(child: Text("Something went wrong"));
              }
              if (snapshots.connectionState == ConnectionState.waiting) {
                Center(child: CircularProgressIndicator());
              }
              if (snapshots.data.size == 0) {
                return Text("No Colors Added");
              }
              return ListView.builder(
                shrinkWrap: true,
                primary: false,
                itemCount: snapshots.data.size,
                itemBuilder: (c, index) {
                  var data = snapshots.data.docs[index];
```

```
                final bgColor = Color(hexColor(data['colorHexacode']));
                final color = Color(hexColor(data['textColor']));
                return Padding(
                  padding: const EdgeInsets.only(bottom: 14),
                  child: CustomTile(
                    snap: snapshots,
                    index: index,
                    icon: Icons.color_lens,
                    txtColor: color,
                    title: data['colorName'],
                    color: bgColor,
                    subTitle1: data['colorHexacode'],
                    subTitle2: data['colorRGBcode'],
                  ),
                );
              },
            );
          },
        ),
      ),
    ),
  );
 }
}
```

FAMILY LIST

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:colorations_admin/Source/Constants/constants.dart';
import
'package:colorations_admin/Source/Repository/Colors%20Repository/firebase_services.dar
t';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import '../../../Helpers/custom_tile.dart';
import '../../../Helpers/user_model.dart';

class AllFamilyList extends StatelessWidget {
  const AllFamilyList({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final FirebaseService service = FirebaseService();
    return Scaffold(
      backgroundColor: bgColor,
      appBar: AppBar(
        title: Text("All Families"),
        centerTitle: true,
        leading: IconButton(
          onPressed: () => Get.back(),
          icon: Icon(Icons.arrow_back_ios, color: Colors.white70),
        ),
        elevation: 0,
        backgroundColor: Colors.transparent,
      ),
      body: SingleChildScrollView(
        child: Container(
          padding: const EdgeInsets.only(left: 18, top: 6, right: 18),
          child: StreamBuilder<QuerySnapshot>(
            stream: service.colorFamilies.snapshots(),
            builder: (context, AsyncSnapshot<QuerySnapshot> snapshots) {
              if (snapshots.hasError) {
                Center(child: Text("Something went wrong"));
              }
              if (snapshots.connectionState == ConnectionState.waiting) {
                Center(child: CircularProgressIndicator());
              }
              if (snapshots.data.size == 0) {
                return Text("No Color Family Added");
              }
              return ListView.builder(
                shrinkWrap: true,
                primary: false,
                itemCount: snapshots.data.size,
                itemBuilder: (c, index) {
                  var data = snapshots.data.docs[index];
                  final bgColor = Color(hexColor(data['hexacode']));
                  return Padding(
                    padding: const EdgeInsets.only(bottom: 14),
                    child: CustomTile(
                      snap: snapshots,
                      index: index,
                      icon: Icons.location_pin,
                      txtColor: Colors.white,
                      title: data['familyName'],
                      color: bgColor,
                      subTitle1: data['hexacode'],
                      subTitle2: data['rgbCode'],
                    ),
                  );
                },
              );
            },
                ),
              ),
            ),
          );
        }
}
```

OPPOSITE COLORS

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

import '../../../Constants/constants.dart';
import '../../../Helpers/custom_tile.dart';
import '../../../Helpers/user_model.dart';
import '../../../Repository/Colors Repository/firebase_services.dart';

class AllOppositeColorsList extends StatelessWidget {
  const AllOppositeColorsList({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final FirebaseService service = FirebaseService();

    return Scaffold(
      backgroundColor: bgColor,
      appBar: AppBar(
        title: const Text("All Family Colors"),
        centerTitle: true,
        leading: IconButton(
          onPressed: () => Get.back(),
          icon: Icon(Icons.arrow_back_ios, color: Colors.white70),
        ),
        elevation: 0,
        backgroundColor: Colors.transparent,
      ),

      // body
      body: SingleChildScrollView(
        child: Container(
          padding: const EdgeInsets.only(left: 18, top: 6, right: 18),
          child: StreamBuilder(
            stream: service.oppColors.snapshots(),
            builder: (context, AsyncSnapshot<QuerySnapshot> snapshots) {
              if (snapshots.hasError) {
                Center(child: Text("Something went wrong"));
              }
              if (snapshots.connectionState == ConnectionState.waiting) {
                Center(child: CircularProgressIndicator());
              }

              if (snapshots.data.size == 0) {
                return Text("No Color Added");
              }
              return ListView.builder(
                shrinkWrap: true,
                primary: false,
                itemCount: snapshots.data.size,
                itemBuilder: (c, index) {
                  var data = snapshots.data.docs[index];
                  final bgColor =
                    Color(hexColor(data['oppositeColorHexacode']));
                  final color = Color(hexColor(data['textColor']));
                  final rgb =

"${data['oppositeColorRcode']},${data['oppositeColorGcode']},${data['oppositeColorBcod
e']}";
                  return Padding(
                    padding: const EdgeInsets.only(bottom: 14),
                    child: CustomTile(
                      snap: snapshots,
                      index: index,
                      icon: Icons.color_lens,
                      txtColor: color,
                      title: data['oppositeColorName'],
                      color: bgColor,
                      subTitle1: data['oppositeColorHexacode'],
                      subTitle2: rgb,
                    ),
                  );
                },
              );
            },
          ),
        ),
      ),
    );
  }
}
```

USERS LIST

```
import 'package:colorations_admin/Source/Constants/constants.dart';
import 'package:colorations_admin/Source/Helpers/user_model.dart';
```

```
import
'package:colorations_admin/Source/Repository/Colors%20Repository/firebase_services.dar
t';

import 'package:flutter/material.dart';

import 'package:get/get.dart';


import '../../../Helpers/custom_tile.dart';


class AllUsersList extends StatelessWidget {
  const AllUsersList({Key key}) : super(key: key);


  @override
  Widget build(BuildContext context) {
    final FirebaseService service = FirebaseService();


    return Scaffold(
      backgroundColor: bgColor,
      appBar: AppBar(
        title: Text("All Users"),
        centerTitle: true,
        leading: IconButton(
          onPressed: () => Get.back(),
          icon: Icon(Icons.arrow_back_ios, color: Colors.white70),
        ),
        elevation: 0,
        backgroundColor: Colors.transparent,
      ),
      body: SingleChildScrollView(
        child: Container(
          padding: const EdgeInsets.only(left: 18, top: 6, right: 18),
          child: FutureBuilder<List<UserModel>>(
future: service.getAllUsers(),
builder: (context, snapshots) {
  if (snapshots.connectionState == ConnectionState.done) {
    if (snapshots.hasData) {
      //
      return ListView.builder(
        shrinkWrap: true,
        itemCount: snapshots.data.length,
        itemBuilder: (c, index) {
          var fullName =
              "${snapshots.data[index].firstName} ${snapshots.data[index].lastName}";
          return Padding(
            padding: EdgeInsets.only(bottom: 14),
            child: CustomTile(
              snapshots: snapshots,
              index: index,
              title: fullName,
              txtColor: Colors.white,
              icon: Icons.account_circle_rounded,
              color: secondaryColor,
              subTitle1: snapshots.data[index].phoneNo,
              subTitle2: snapshots.data[index].email,
            ),
          );
        },
      );
    }
  }
  if (snapshots.hasError) {
    Center(child: Text("Something went wrong"));
```

```
              }
              if (snapshots.connectionState == ConnectionState.waiting) {
                Center(child: CircularProgressIndicator());
              }
              return Center(child: Text("No Users Found"));
            },
          ),
        ),
      ),
    );
  }
}
```

HOMEPAGE SCREEN

```
import
'package:colorations_admin/Source/Screens/Components/Dashboard%20Comps/content.dar
t';
import 'package:flutter/material.dart';


class HomepageScreen extends StatelessWidget {
  const HomepageScreen({Key key}) : super(key: key);


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color(0xFF212332),
      body: SafeArea(
        child: Row(
          crossAxisAlignment: CrossAxisAlignment.start,
          // ignore: prefer_const_literals_to_create_immutables
          children: [
            Expanded(
              flex: 5,
              child: DashboardContent(),
            ),
          ],
        ),
      ),
    );
  }
}
```

SPLASH SCREEN

```
import 'dart:async';
import 'dart:math';
import 'package:colorations_admin/Source/Constants/constants.dart';
import 'package:colorations_admin/Source/Entry%20Point/entry_point.dart';
import 'package:colorations_admin/Source/Screens/homepage_screen.dart';
import 'package:flutter/material.dart';
import 'package:flutter_easyloading/flutter_easyloading.dart';
import 'package:get/get.dart';

import '../Repository/shared_preference.dart';
import 'Authentication/login_screen.dart';

class SplashScreen extends StatefulWidget {
  const SplashScreen({Key key}) : super(key: key);

  @override
  State<SplashScreen> createState() => _SplashScreenState();
}
```

```
class _SplashScreenState extends State<SplashScreen> {
  final PrefService _prefService = PrefService();
  int textCount;

  @override
  void initState() {
    _prefService.readCache('password').then((value) {
      if (value != null) {
        return Timer(Duration(seconds: 3), () {
          Get.offAll(() => EntryPoint());
          EasyLoading.dismiss();
        });
      } else {
        return Timer(Duration(seconds: 3), () {
          Get.offAll(() => LoginScreen());
          EasyLoading.dismiss();
        });
      }
    });

    int txtCount = Random().nextInt(5);
    setState(() {
      textCount = txtCount;
    });
    super.initState();
  }

  // Pages for Navigation Bar
  List<String> textList = <String>[
    "Setting you up!",
    "Please Wait...",
    "Loading...",
    "Working on Data!",
    "Fetching the Data..."
  ];

  Widget centerContent() {
    EasyLoading.show();
    return Center(
      child: Padding(
        padding: const EdgeInsets.only(top: 450),
        child: Column(
          children: [
            Text(
              textList[textCount],
              style: TextStyle(
                color: Colors.white,
                fontSize: 18,
                fontWeight: FontWeight.bold,
              ),
            ),
          ],
        ),
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: bgColor,
      body: Center(
          child: Column(
        children: [centerContent()],
      )),
    );
  }
}
```

FIREBASE SERVICES

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:colorations_admin/Source/Helpers/user_model.dart';
import 'package:firebase_auth/firebase_auth.dart';

import '../../Helpers/colors_model.dart';

class FirebaseService {
// Variables
  final db = FirebaseFirestore.instance;

  CollectionReference colorFamilies =
      FirebaseFirestore.instance.collection("colorFamilies");
  CollectionReference familyColors =
      FirebaseFirestore.instance.collection("familyColors");
  CollectionReference oppColors =
      FirebaseFirestore.instance.collection("oppColors");

  Future<void> saveColorFamily(
      {CollectionReference reference,
      Map<String, dynamic> data,
      String docName}) {
```

```
    return reference.doc(docName).set(data);
  }

  // Fetch all the users from database
  Future<List<UserModel>> getAllUsers() async {
    final snapshot = await db.collection("users").get();
    final userData =
        snapshot.docs.map((e) => UserModel.fromSnapshot(e)).toList();
    return userData;
  }

  Future<List<ColorsModel>> getAllFamilyColors() async {
    final snapshot = await db.collection("familyColors").get();
    final colorData =
        snapshot.docs.map((e) => ColorsModel.fromSnapshot(e)).toList();
    return colorData;
  }
}
```

CHAPTER 11

SOFTWARE TESTING

11.1 Introduction

Testing is an essential part of the software development process, which involves verifying and validating that the software meets its requirements, works as intended, and is free of defects or errors. In this color theory application, Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done. The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. And then shown that the application works properly in all the condition.

11.2 Test Units

In this coloration application, we have opted for two types of testing which is Unit Testing and the System Testing.

The basic units in Unit Testing are:

- Color scheme generation functions: Unit tests can be written to verify that color scheme generation functions are working correctly. For example, a unit test could be written to ensure that the analogous color scheme generation function returns a valid set of analogous colors for a given base color.
- User interface components: Unit tests can be written to verify that user interface components are working correctly. For example, a unit test could be written to ensure that the color component correctly updates the selected color when a user clicks on it.
- Database functions: Unit tests can be written to verify that database functions, such as adding and retrieving color palettes, are working correctly. For example, a unit test could be written to ensure that a newly added color palette can be retrieved from the database and contains the expected colors.

The basic units in System testing are:

Integration of all programs is correct or not

- Checking whether the entire system after integrating is working as expected.

- The system is tested as whole after the unit testing.

Test Case Number	Testing Scenario	Expected Result	Result
TC-01	Display of onboarding screen	“Onboarding Screen should appear after Splash screen”	pass
TC-02	Display of Complementary colors	“Complementary screen should be appeared after user click on particular color”	pass
TC-03	Display of Family colors	“Family colors screen should be displayed after user click on particular color Family”	pass
TC-04	When heart icon is clicked on the adds then it should be favourited	“Adds should be added to the user favourites list”	pass
TC-05	Display of prompt when add details are missed	“Prompt message should be displayed”	Pass
TC-06	When user creates a color, then it navigates to HOME page	“Screen should navigate to the HOME page”	pass
TC-07	When user creates a color it should be displayed in MY COLORS page	“Add should be displayed in MY COLORS page”	pass

Table 11.2.1 Testing

CHAPTER 12

SNAPSHOTS

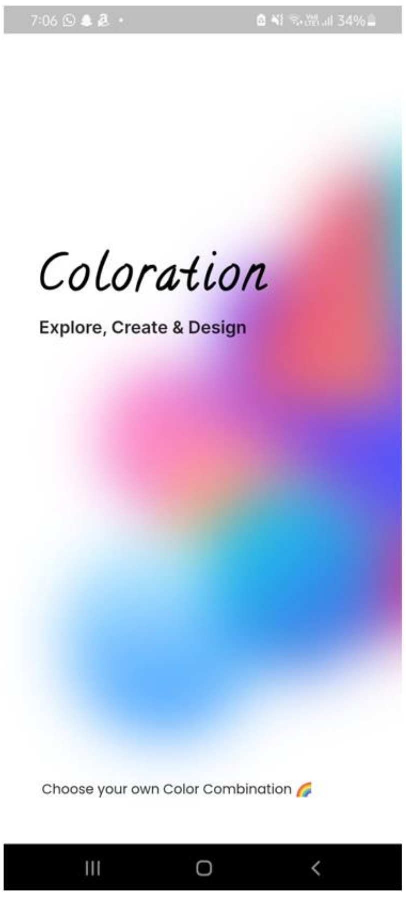


Fig.12.1 Splash Screen

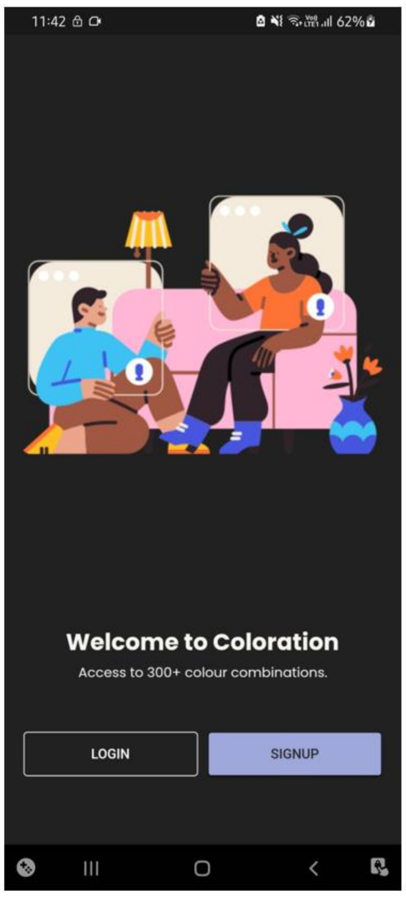
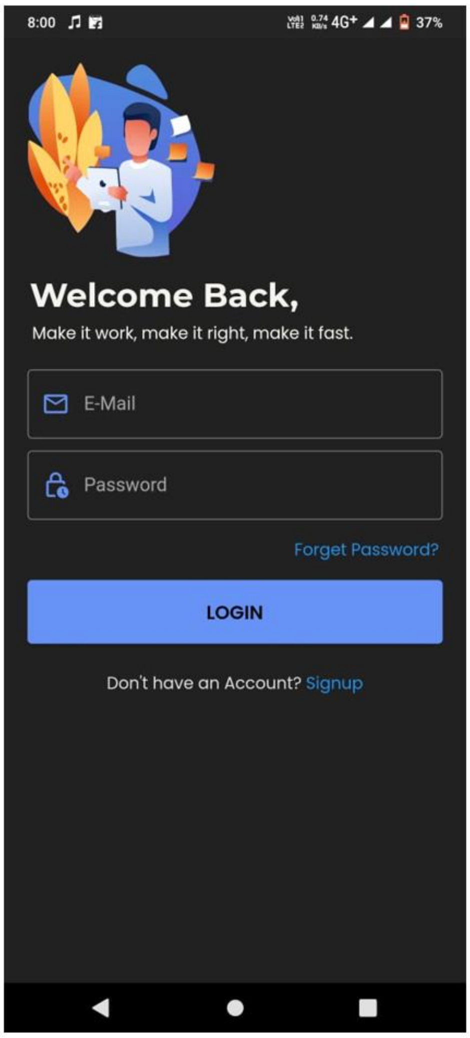


Fig.12.2 Onboarding Screen


8:00 4G+ 37%

Welcome Back,

Make it work, make it right, make it fast.

E-Mail

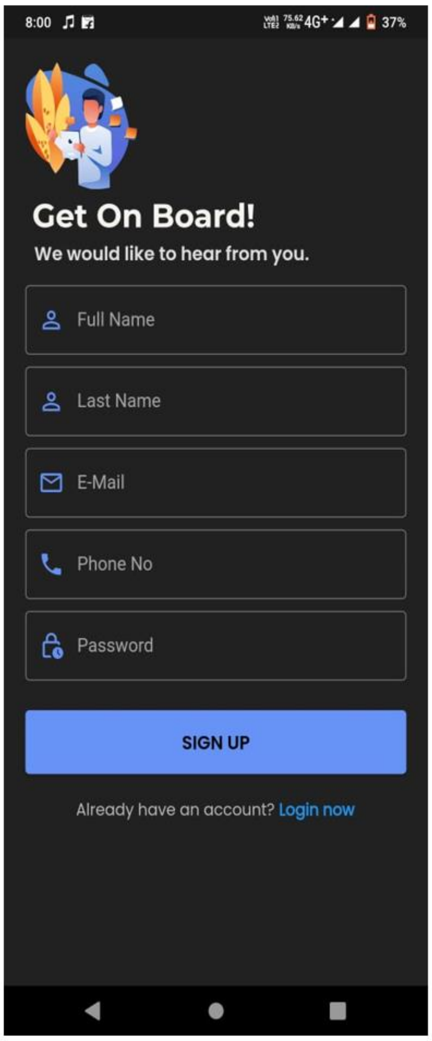
Password

Forget Password?

LOGIN

Don't have an Account? Signup

Fig.12.3 Login Screen


8:00 4G+ 37%

Get On Board!

We would like to hear from you.

Full Name

Last Name

E-Mail

Phone No

Password

SIGN UP

Already have an account? Login now

Fig.12.4 Sign Up Screen

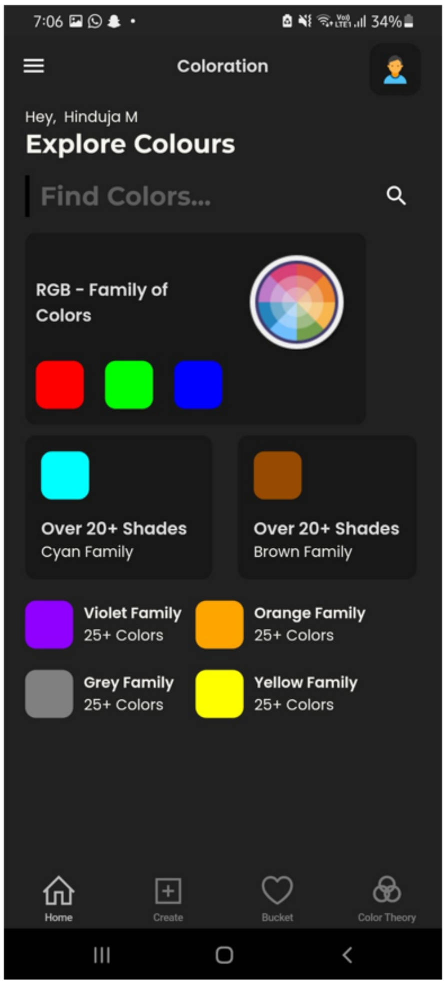


Fig.12.5 Home Page

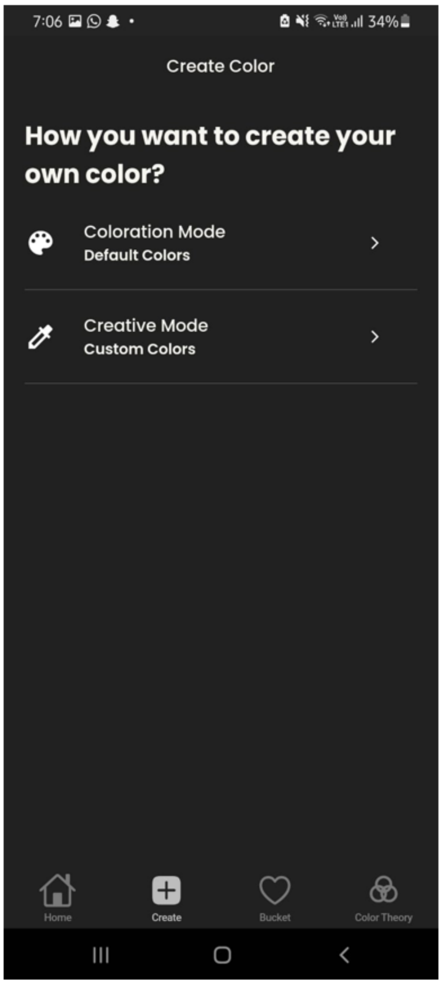
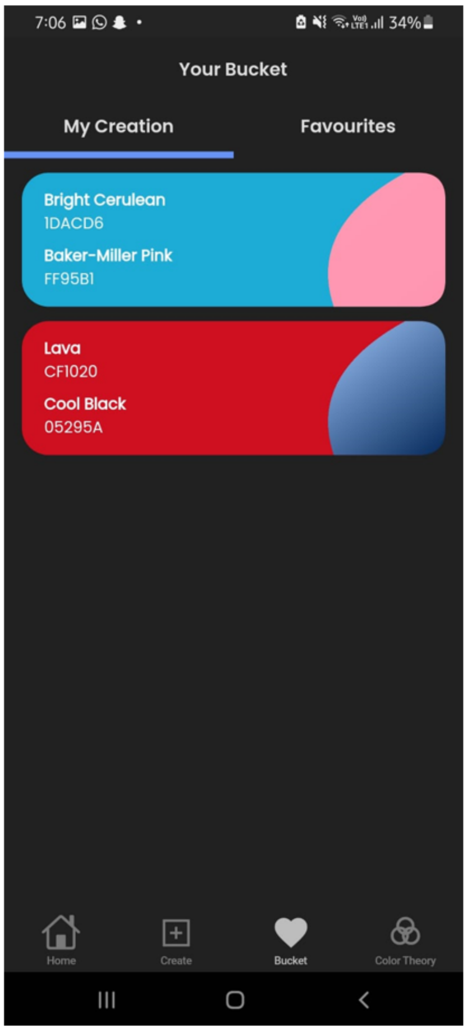


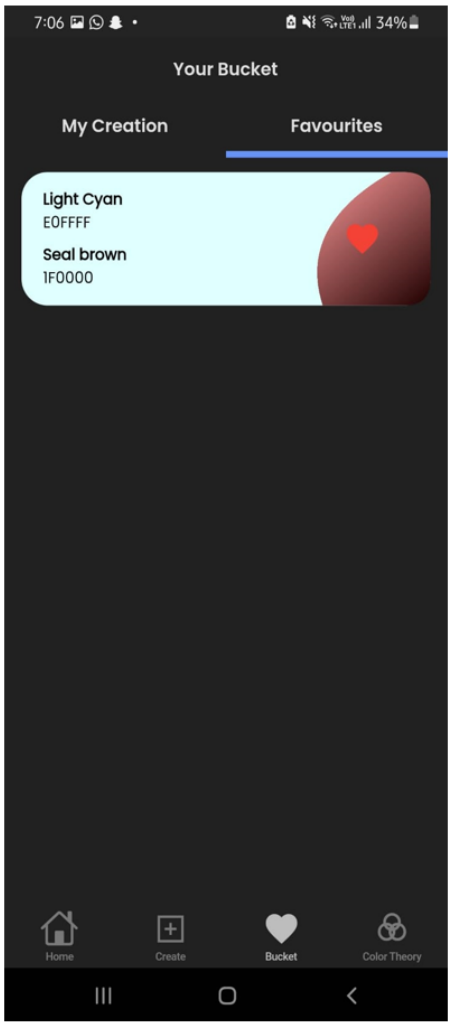
Fig.12.6 Create Color Page



Bright Cerulean
1DACD6
Baker-Miller Pink
FF95B1

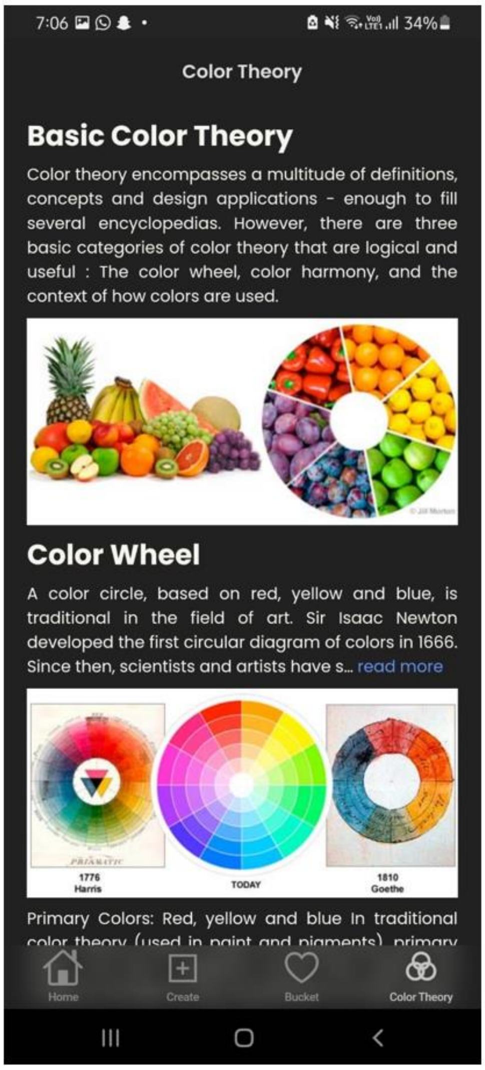
Lava
CF1020
Cool Black
05295A

Fig.12.7 My Creation Page



Light Cyan
E0FFFF
Seal brown
1F0000

Fig.12.8 Favorites Page



Color Theory

Basic Color Theory

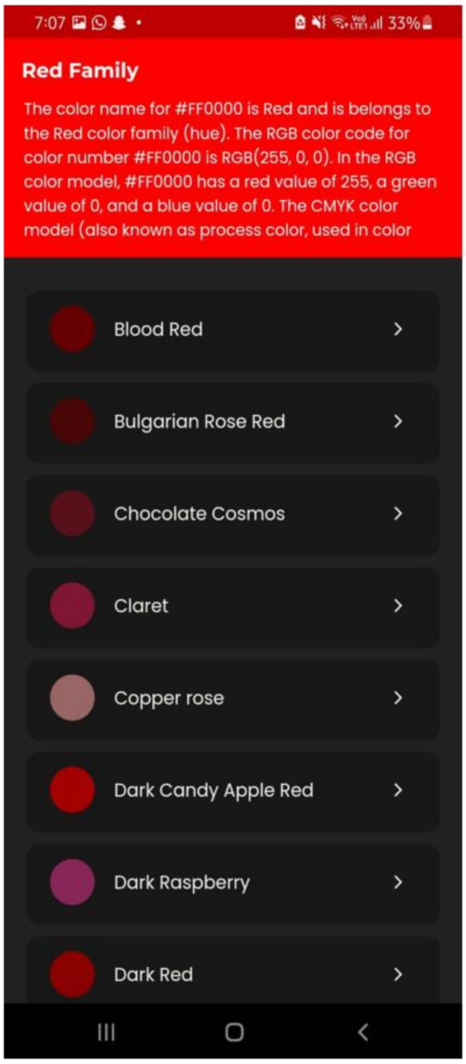
Color theory encompasses a multitude of definitions, concepts and design applications - enough to fill several encyclopedias. However, there are three basic categories of color theory that are logical and useful : The color wheel, color harmony, and the context of how colors are used.

Color Wheel

A color circle, based on red, yellow and blue, is traditional in the field of art. Sir Isaac Newton developed the first circular diagram of colors in 1666. Since then, scientists and artists have s... read more

Primary Colors: Red, yellow and blue In traditional color theory (used in paint and pigments), primary

Fig.12.9 Color Theory Page

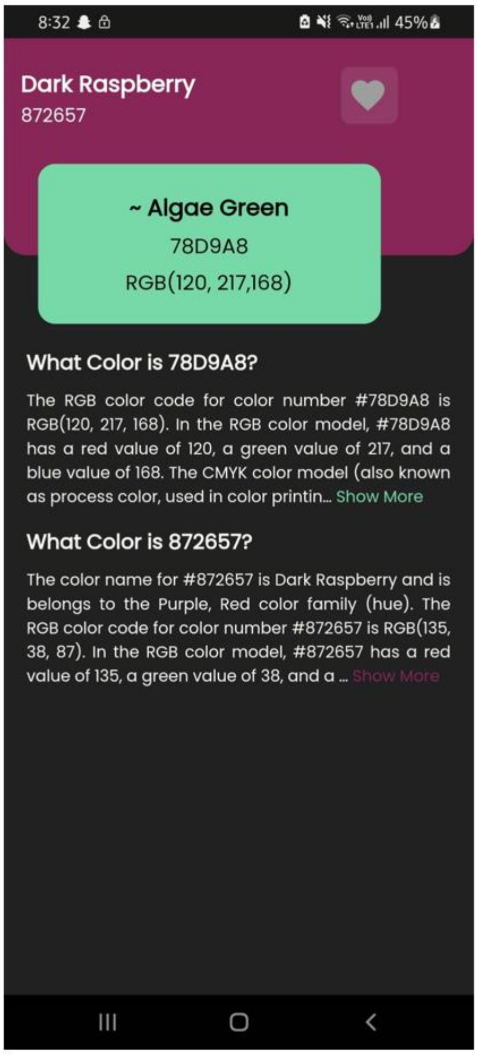


Red Family

The color name for #FF0000 is Red and is belongs to the Red color family (hue). The RGB color code for color number #FF0000 is RGB(255, 0, 0). In the RGB color model, #FF0000 has a red value of 255, a green value of 0, and a blue value of 0. The CMYK color model (also known as process color, used in color

- Blood Red
- Bulgarian Rose Red
- Chocolate Cosmos
- Claret
- Copper rose
- Dark Candy Apple Red
- Dark Raspberry
- Dark Red

Fig.12.10 Red Family colors Page


8:32 45%

Dark Raspberry
872657

~ Algae Green
78D9A8
RGB(120, 217,168)

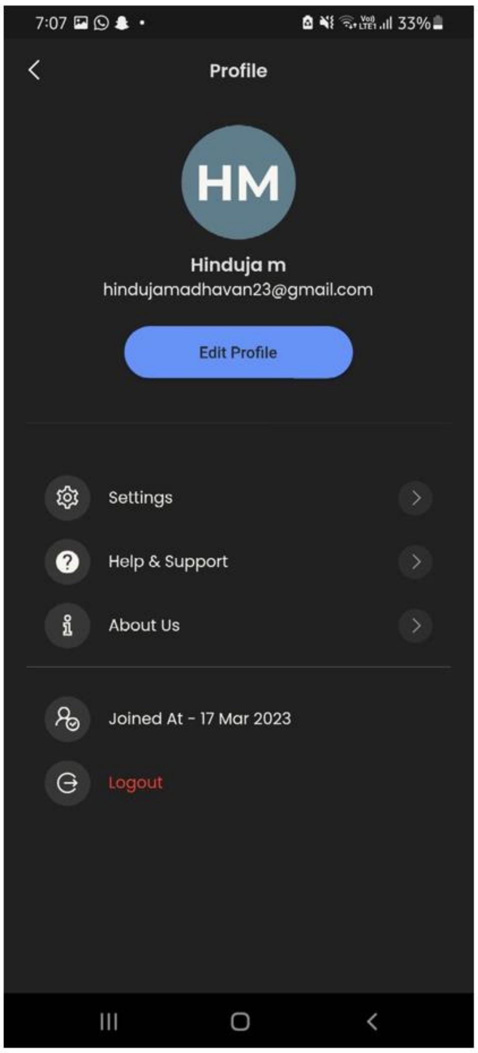
What Color is 78D9A8?

The RGB color code for color number #78D9A8 is RGB(120, 217, 168). In the RGB color model, #78D9A8 has a red value of 120, a green value of 217, and a blue value of 168. The CMYK color model (also known as process color, used in color printin... Show More

What Color is 872657?

The color name for #872657 is Dark Raspberry and is belongs to the Purple, Red color family (hue). The RGB color code for color number #872657 is RGB(135, 38, 87). In the RGB color model, #872657 has a red value of 135, a green value of 38, and a ... Show More

Fig.12.11 Dark Raspberry color Screen


7:07 33%

Profile

HM

Hinduja m
hindujamadhavan23@gmail.com

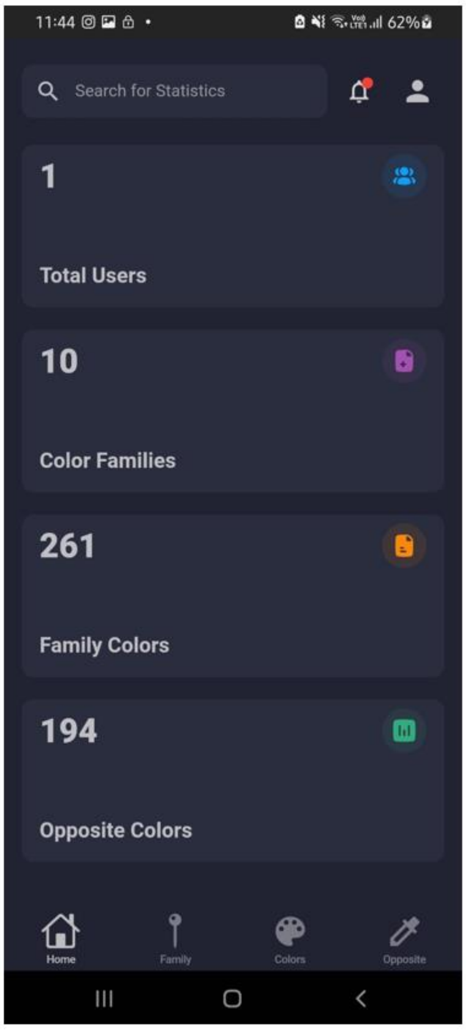
Edit Profile

Settings
Help & Support
About Us

Joined At - 17 Mar 2023

Logout

Fig.12.12 Profile Settings Page



Search for Statistics

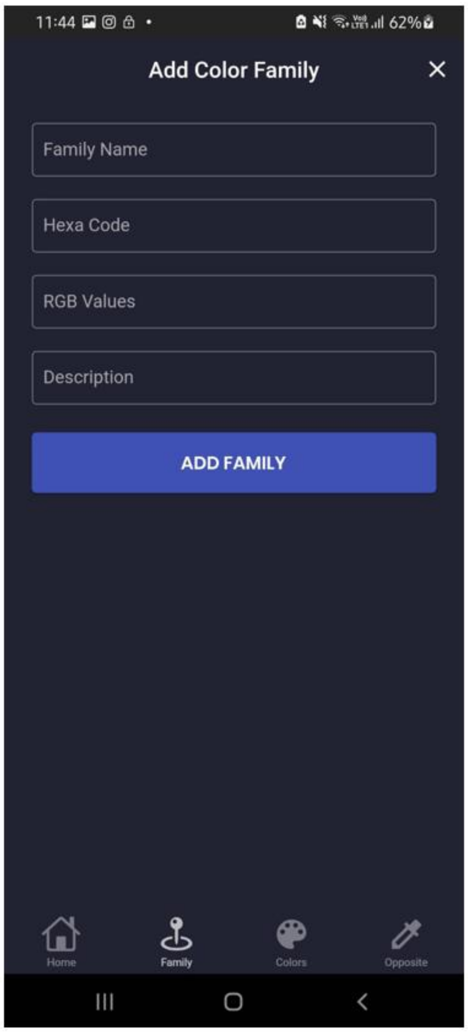
1
Total Users

10
Color Families

261
Family Colors

194
Opposite Colors

Fig.12.13 Admin-Dashboard



Add Color Family

Family Name

Hexa Code

RGB Values

Description

ADD FAMILY

Fig.12.14 Admin-Add Color Family Screen

CHAPTER 13

SYSTEM MAINTENANCE

13.1 Introduction

System maintenance refers to the ongoing process of keeping computer systems, networks, and software applications up-to-date, secure, and functioning properly. It involves activities such as software updates, hardware repairs and replacements, data backups, security patches, and system performance optimization.In our coloration application we have opted for the perfective maintenance.

13.1.1 Perfective Maintenance

Perfective maintenance is a type of system maintenance that involves making improvements to a system to enhance its functionality, performance, or usability. The goal of perfective maintenance is to make the system more efficient, user-friendly, and effective for its intended purpose.

CHAPTER 14

CONCLUSION

This android application allows the user to know about various colour combinations without having any basic knowledge of colour theory. The most selected colour will be suggested by the application to new users. This application will be used for many purposes such as costume designing, web designing, art, architecture, interior designing, animation, etc.… This application also helps the user to add their favorite color combinations to the Wishlist from where they can use it for future purpose. The main aim of this project is to avoid the middle man in dealings.

CHPATER 15

FUTURE ENHANCEMENTS

The application could use advanced algorithms to more accurately match colors, including the ability to match colors based on different lighting conditions and color spaces. The application could include a feature that simulates how colors appear to people with different types of color blindness. This would be helpful for designers who want to ensure their designs are accessible to everyone. The application could incorporate data analysis and trend forecasting to help users stay ahead of the curve when it comes to color trends. The application could be integrated with popular design software such as Adobe Photoshop and Illustrator to make it easier for users to incorporate color theory into their designs. The application could include a feature that allows users to create custom color palettes based on their preferences and needs. This could be especially useful for branding and marketing professionals who need to create unique color schemes for their clients. The application could be made as an offline application instead of Online.

REFERENCES

TEXT REFERENCES

[1] Kenneth E. Kendall, *System Analysis and Design*, Julie E. Kendall.

[2] Dawn Griffiths, *Head First Android Development*, O'Reilly, 1 September 2017.

[3] Grayce M. Booth, *Software Engineering*, 1973.

[4] Ryan Cohen, *GUI Design for Android Apps*, Apress, 28 August 2014.

[5] Alessandro Biessek, *Flutter for Beginners*, Packt Publishing, September 12, 2019.

WEB REFERENCES

[6] Flutter Mapp (1 Jan 2021), Flutter Explained
Available: URL<https://youtu.be/CD1Y2DmL5JM>.

[7] Google and community (September 2015) “Flutter Dev” Packages.
Available: URL<https://docs.flutter.dev/get-started/install>.

[8] Firebase Inc. (November 2011) “Google Firebase” Fire store.
Available: URL<https://firebase.google.com>.

[9] Refsnes Data AS (May 1998) “Flutter” Flutter Framework.
Available: URL<https://www.w3schools.blog/which-one-is-better-kotlin-vs-flutter>.