ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Project Documentation Format

1. INTRODUCTION

- **Project Title**: **GrainPalette**
  (This is a fictional color palette generator for designers and developers)
- **Team Members**:
  - **Hema– Frontend Developer**
  - **Mounika– Backend Developer**
  - **Alekya– UI/UX Designer**
  - **Upendra– Project Manager**

---

2. Project Overview

- **Purpose**:
  GrainPalette is a web-based application designed to help designers and developers generate harmonious color palettes for use in their projects. The tool allows users to explore, customize, and save color combinations for websites, apps, and other digital designs. The goal is to simplify the color selection process and make it easier for users to find aesthetically pleasing palettes that fit their design needs.
- **Features**:
  - **Color Palette Generator**: Automatically generates color palettes based on a user-defined base color or by random generation.

    - **Adjustable Settings**: Users can modify the palette by adjusting hue, saturation, and brightness levels.
    - **Save Palettes**: Users can save their favorite palettes and export them for use in design tools or code.
    - **Accessibility Enhancements**: Includes a color-blind friendly mode to assist users with color vision deficiencies.
    - **Palette Sharing**: Allows users to share their color palettes with others via a unique URL or social media.
    - **Palette Preview**: Visualize the generated palette applied to mockup designs, such as web pages or mobile app screens, to see how the colors work together in context.
    - **Dark and Light Mode**: The app supports both dark and light themes to accommodate different user preferences and lighting conditions.

---

3. Architecture

GrainPalette uses a **hybrid architecture** combining a **deep learning model** (built in Python with TensorFlow/Keras) and a **web application** for user interaction.

AI/ML Model (Python, TensorFlow/Keras)

- **Input**: Single base color or uploaded image
- **Model Type**: CNN-based encoder-decoder trained on curated color palette datasets
- **Output**: Harmonious color palette (5–7 HEX or RGB values)
- **Libraries Used**: TensorFlow, Keras, NumPy, OpenCV, Matplotlib

Web Interface

- **Frontend**: React.js
    - Fetches palette from AI backend via REST API
    - Includes UI for uploading images or choosing base color
- **Backend**: Python (Flask/FastAPI)
    - Hosts trained ML model
    - Exposes REST API endpoints for generating palettes
    - Handles image preprocessing
- **Deployment**: Dockerized architecture deployable on AWS/GCP/Heroku

---

4. Setup Instructions

1. Prerequisites

- Python 3.8+
- Node.js (for frontend)
- MongoDB (if storing users/palettes)
- pip / virtualenv
- TensorFlow 2.x
- Docker (optional but recommended)

2. Installation Steps

Backend (Python + AI Model)

bash

CopyEdit

```
git clone https://github.com/username/grainpalette

cd grainpalette/backend
```

```
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Run the Flask/FastAPI server:

bash

CopyEdit

```
python app.py
# or for FastAPI
uvicorn app:app --reload
```

Frontend (React)

bash

CopyEdit

```
cd grainpalette/client
npm install
npm start
```

Docker (Optional Unified Setup)

bash

CopyEdit

```
docker-compose up --build
```

---

5. Folder Structure

bash

CopyEdit

```
grainpalette/
```

```
├── client/              # React frontend
│   └── src/
│       ├── components/
│       ├── pages/
│      └── App.js
├── backend/              # Python backend
│   ├── app.py            # API server (Flask/FastAPI)
│   ├── model/
│   │   └── palette_model.h5 # Trained Keras model
│   ├── utils/
│   │   └── image_processing.py
│   └── requirements.txt
├── data/                 # Dataset (images + palettes)
├── docker-compose.yml
└── README.md
```

---

6. Running the Application

1. Start Backend (Python)

bash

CopyEdit

```
cd backend
python app.py
```

2. Start Frontend (React)

bash

CopyEdit

```
cd client
npm start
```

3. Test the API

Send a POST request to:

bash

CopyEdit

```
POST http://localhost:5000/generate
Body: {
  "base_color": "#ff5733"
}
```

---

7. API Documentation

POST /generate

- **Description**: Generate a color palette from a base color
- **Request Body**:

json

CopyEdit

```
{
  "base_color": "#ff5733"
}
```

- **Response**:

json

CopyEdit

```
{
  "palette": ["#ff5733", "#ffe1e0", "#c70039", "#900c3f", "#581845"]
}
```

POST /generate-from-image

- **Description**: Upload an image and receive a palette
- **Multipart Form**:
  - **file**: Image file
- **Response**:

json

CopyEdit

```
{
  "palette": ["#123456", "#654321", "#abcdef"]
}
```

---

8. Authentication

- **JWT-based authentication** for secure access
- **Login/Register Endpoints**
  - /auth/login
  - /auth/register
- Tokens stored in localStorage or HTTPOnly cookies
- Protected routes (e.g., save palette, view history)

---

9. User Interface

Key UI Components:

- Color Picker Input
- Image Upload Module
- Real-time Palette Preview
- Save to Favorites Button
- Palette History Page

(You can insert screenshots here in final documentation)

---

10. Testing

- **Model Testing**:
  - Unit tests for inference speed, output palette validation
  - Visual inspection using matplotlib
- **API Testing**:
  - **Postman** for manual testing
  - **PyTest** or **unittest** for automated backend testing
- **Frontend Testing**:
  - **Jest** + **React Testing Library**

---

11. Screenshots or Demo

- Live Demo: *[Insert hosted demo link]*
- Sample UI:

bash

CopyEdit

[Insert screenshots of color palette generator, palette preview, history page, etc.]

---

12. Known Issues

- Model struggles with very dark/low contrast images
- Occasional palette outputs are too similar in hue
- Need better mobile responsiveness on frontend

---

13. Future Enhancements

✅ **AI/ML Model**

- Train on a **larger dataset** with labeled palette moods (e.g., "warm", "vintage")
- Add **style transfer** for palette generation based on famous artworks
- Integrate **GANs** for creative color outputs

✅ **Web App**

- Add **drag-and-drop** for uploading files
- Enable **palette sharing** via custom links
- Add **dark mode** and downloadable **CSS/SCSS exports**