SkinLens Installation Manual

Team Biased

Table of Contents

1. Introduction

Welcome to the **SkinLens Installation Manual**. SkinLens is an advanced **AI-powered skin condition diagnosis platform** that leverages deep learning and image recognition to analyze skin condition images and provide potential diagnoses. It is designed to assist dermatologists and individuals by delivering quick, accurate, and actionable results.

This installation manual is intended for **developers**, **system administrators**, and **technical teams** who want to set up and deploy the SkinLens application in both **development** and **production environments**. Whether you are setting up the project for local testing or deploying it to the cloud, this guide provides step-by-step instructions to ensure a smooth and successful setup process.

Key Features of SkinLens

1. **AI-Powered Diagnosis**: Leverages a pre-trained deep learning model to recognize 20+ skin conditions.
2. **User-Friendly Interface**: React-based frontend for seamless interaction and diagnosis tracking.
3. **Real-Time Predictions**: Backend powered by Flask and TensorFlow for fast image processing.
4. **Cloud Integration**: Supports cloud storage and hosting using Google Cloud Services (GCS) and Firebase.
5. **Secure and Scalable**: Designed for both small-scale deployments and enterprise-level production environments.

Scope of the Manual

This document will guide you through:

- **Setting up the development environment** for testing and modifications.
- **Installing backend dependencies** such as Python, TensorFlow, and the AI model.
- **Configuring the frontend** React-based application for real-time diagnosis.
- **Setting up Firebase** for cloud storage and user authentication.
- **Deploying the application** to production using Docker containers or Firebase Hosting.
- **Troubleshooting** common issues that may arise during setup or deployment.

By the end of this manual, you will have a fully functional **SkinLens application** running locally or in a production environment, ready to process images and deliver AI-powered skin condition diagnoses.

2. System Requirements

To ensure the successful setup and deployment of the **SkinLens** application, your system must meet the following **hardware** and **software requirements**. This section is divided into requirements for both **development** and **production** environments.

2.1 Hardware Requirements

Component	Minimum	Recommended
Processor	Core i5 2.0 GHz	Core i7 2.5 GHz or equivalent
Memory (RAM)	8 GB	16 GB
Storage	10 GB free space	20 GB free space
GPU (optional)	NVIDIA GTX 1050 or equivalent	NVIDIA RTX 3060 or higher
Network	Stable internet connection	High-speed broadband connection

Note: A **GPU** is optional but highly recommended for running TensorFlow models in production environments, as it significantly improves inference and training performance.

2.2 Software Requirements

Component	Required Version	Details
Operating System	Windows 10/11, macOS 12+, Ubuntu 18.04+	64-bit systems only
Python	3.10+	For backend (Flask and TensorFlow)
Node.js	18+	For frontend React application
NPM (Node Package Manager)	8.0+	Bundled with Node.js
Docker	24+	For containerized deployments
Git	2.40+	For cloning and version control
Google Cloud SDK	Latest	Required for accessing GCS models
Firebase CLI	Latest	For frontend deployment to Firebase
TensorFlow	2.17.0	Required for backend model inference

Component	Required Version	Details
Web Browser	Google Chrome, Firefox, Edge	Latest version for frontend testing

2.3 Development Environment

In the development environment, you will require the following tools and dependencies:

1. **IDE or Code Editor**:
   - Visual Studio Code (recommended)
   - PyCharm or any preferred Python IDE for backend development
   - React IDE tools for frontend development
2. **Virtual Environment Setup**:
   Use Python venv to manage backend dependencies and isolate the environment:

   ```
   python3 -m venv venv
   source venv/bin/activate
   ```

3. **Frontend Developer Tools**:
   - Node.js and NPM for managing React dependencies
   - Browser Developer Tools for inspecting UI components

2.4 Production Environment

For deploying the application in production, you will need:

1. **Cloud Infrastructure**:
   - **VM/Server**: Ubuntu 20.04+ (recommended for cloud deployments)
   - **Docker Engine**: To containerize and deploy the backend and frontend
   - **Google Cloud Storage (GCS)**: For storing AI models and user-uploaded images
   - **Firebase**: For frontend hosting, Firestore (database), and Firebase Authentication
2. **System Configuration**:
   - Ensure ports **8080** (backend) and **3000** (frontend) are open.
   - Sufficient storage space for logs, models, and user data.
3. **SSL Certificates**:
   - Required for secure communication in production (HTTPS).
   - Tools like **Let's Encrypt** can be used to generate free SSL certificates.

2.5 Optional Tools

- **Postman**: For testing backend API endpoints.
- **Docker Desktop**: For container management on local machines (Windows/Mac).
- **Kubernetes**: For advanced production deployments requiring container orchestration.

3. Installation Procedures

This section provides step-by-step guidance to set up the **SkinLens** application for both **Development Environment** and **Production Environment**. Follow the instructions carefully to ensure a smooth setup.

3.1 Development Environment

The development environment is used for local testing and feature development. Follow the steps below:

Step 1: Clone the Repository

Start by cloning the **SkinLens** project repository:

```
git clone https://github.com/htmw/2024F-Biased.git
cd 2024F-Biased
```

Step 2: Backend Setup

1. Navigate to the backend directory:

   ```
   cd flask-server
   ```

2. **Set up a virtual environment** to isolate dependencies:

   ```
   python3 -m venv venv
   source venv/bin/activate  # On Windows: venv\Scripts\activate
   ```

3. **Install required Python dependencies**:

   ```
   pip install --upgrade pip
   pip install -r requirements.txt
   ```

4. **Download the AI model**:
   - The model is hosted on Google Cloud Storage (GCS). Download it to the models/ directory:

     ```
     wget -P models/ https://storage.googleapis.com/skinlens-models/final_model.keras
     ```

5. Verify the backend setup:

   ```
   python app.py
   ```

The API should start running at http://localhost:8080.

Step 3: Frontend Setup

1. Navigate to the frontend directory:

```
cd ../frontend
```

2. Install frontend dependencies using **NPM**:

```
npm install
```

3. Start the development server:

```
npm start
```

4. The frontend will be available at:

```
http://localhost:3000
```

Step 4: Firebase Configuration

To connect the application to Firebase:

1. Go to Firebase Console and create a new project.
2. Obtain the Firebase configuration settings.
3. Update the firebase.js file located in the frontend/src/config/ directory:

```
const firebaseConfig = {
    apiKey: "YOUR_API_KEY",
    authDomain: "YOUR_AUTH_DOMAIN",
    projectId: "YOUR_PROJECT_ID",
    storageBucket: "YOUR_STORAGE_BUCKET",
    messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
    appId: "YOUR_APP_ID"
};
export default firebaseConfig;
```

Step 5: Test the Application

1. Ensure both the backend and frontend are running:
    - Backend: http://localhost:8080
    - Frontend: http://localhost:3000
2. Test the workflow:
    - Upload an image from the frontend.
    - Confirm the prediction is returned successfully from the backend.

3.2 Production Environment

The production environment is used for deployment on a live server or cloud infrastructure.

Step 1: Prepare the Server

1. Set up a **Linux VM or cloud server** (e.g., AWS, GCP, or Azure) with **Ubuntu 20.04+**.
2. Update and install essential tools:

```
sudo apt update
sudo apt install -y python3 python3-pip python3-venv git curl docker docker-compose
```

Step 2: Backend Setup

1. Clone the repository to the server:

```
git clone https://github.com/htmw/2024F-Biased.git
cd 2024F-Biased/flask-server
```

2. Create a virtual environment and install dependencies:

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

3. Download the model to the server:

```
wget -P models/ https://storage.googleapis.com/skinlens-models/final_model.keras
```

4. Start the backend server:

```
python app.py
```

Step 3: Frontend Setup

1. Navigate to the frontend directory:

```
cd ../frontend
```

2. Build the frontend for production:

```
npm install
npm run build
```

3. Use **Firebase Hosting** or any static hosting service to deploy the frontend.

Step 4: Deploy with Docker (Optional)

For a containerized deployment, use **Docker** to host both the frontend and backend.

1. Build the Docker images:

```
docker build -t skinlens-backend ./flask-server
docker build -t skinlens-frontend ./react-frontend
```

2. Run the containers:

```
docker run -d -p 8080:8080 skinlens-backend
docker run -d -p 3000:3000 skinlens-frontend
```

3. Verify that both services are running:
   - Backend: http://<server-ip>:8080
   - Frontend: http://<server-ip>:3000

Step 5: SSL and Security

1. Use **Let's Encrypt** to set up SSL certificates for HTTPS:

```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d yourdomain.com
```

2. Secure the server with a firewall:

```
sudo ufw allow 8080
sudo ufw allow 3000
sudo ufw enable
```

Step 6: Testing and Monitoring

1. Test the production environment using tools like **Postman** or curl.
2. Monitor logs and performance:

```
docker logs <container-id>
```

3. Set up health checks to ensure the application is running smoothly.

4. Setting Up the Backend

The backend of SkinLens is built using **Flask** and **TensorFlow** for serving predictions from the pre-trained AI model. Follow these steps to set up and configure the backend properly.

4.1 Installing Python and Dependencies

1. **Verify Python Installation**
   Ensure that Python 3.10 or higher is installed on your machine.
   Check the version using the following command:

   ```
   python3 --version
   ```

   If Python is not installed, download and install it from the official website: Python Downloads.

2. **Create a Virtual Environment**
   Creating a virtual environment ensures that the dependencies do not conflict with the system Python packages. Run the following commands:

   ```
   cd flask-server
   python3 -m venv venv
   source venv/bin/activate  # For Windows: venv\Scripts\activate
   ```

   - This creates a virtual environment named venv and activates it.

3. **Install Required Dependencies**
   With the virtual environment activated, install all required dependencies using the requirements.txt file:

   ```
   pip install --upgrade pip
   pip install --no-cache-dir -r requirements.txt
   ```

   - The requirements.txt includes all backend dependencies such as Flask, TensorFlow, NumPy, OpenCV, and other necessary libraries.

4. **Verify Dependency Installation**
   To confirm that all dependencies are installed successfully, run:

   ```
   pip freeze
   ```

   - This will display all installed packages and their versions.

4.2 Model Setup

The backend requires a pre-trained AI model for skin condition prediction. The model file must be downloaded and placed in the appropriate directory. Follow these steps:

1. **Download the Pre-trained Model**
   The pre-trained model is hosted on Google Cloud Storage. Use the wget command to download it to the models directory:

```
wget -P models/ https://storage.googleapis.com/skinlens-
models/final_model.keras
```

   - Ensure you have an active internet connection for this step.
2. **Verify the Model File**
   After downloading, verify that the model file is present in the correct location:

```
ls models/
```

   - You should see the file final_model.keras in the models directory.
3. **Testing the Backend Model Loading**
   Run a simple Python script to ensure the model loads correctly:

```
from tensorflow import keras

model = keras.models.load_model("models/final_model.keras")
print("Model loaded successfully!")
```

   - If the model loads without errors, the setup is successful.
4. **Backend Configuration**
   Ensure the backend configuration (app.py) has the correct path to the model file:

```
LOCAL_MODEL_PATH = "models/final_model.keras"
```

   - Verify this line in your app.py file to confirm that the backend uses the correct model path.

By completing these steps, the backend environment is ready, and the model is configured for serving predictions. The backend API will load the model and handle prediction requests seamlessly.

Next, proceed to **Section 5: Setting Up the Frontend**.

5. Setting Up the Frontend

The frontend of SkinLens is built using **React.js** and serves as the user interface for uploading images, interacting with the backend API, and viewing predictions. Follow these steps to set up and configure the frontend properly.

5.1 Installing Node.js and Dependencies

1. **Verify Node.js Installation**
   Ensure that Node.js (version 18 or higher) and npm are installed on your machine. Check the versions using the following commands:

   ```
   node -v
   npm -v
   ```

   If Node.js is not installed, download and install it from the official website: Node.js Downloads.

2. **Navigate to the Frontend Directory**
   Go to the frontend directory:

   ```
   cd frontend
   ```

3. **Install Frontend Dependencies**
   Install all required npm dependencies using the package.json file:

   ```
   npm install
   ```

   - This command will install all React-related packages, including Firebase SDK, Axios, and other libraries used by the frontend.

4. **Verify Dependency Installation**
   After installation, verify that all dependencies are installed successfully:

   ```
   npm list
   ```

   - You should see a list of all installed npm packages and their versions.

5.2 Configuring Firebase for the Frontend

The frontend interacts with Firebase for user authentication, storage, and Firestore. Follow these steps to configure Firebase:

1. **Create a Firebase Project**
   - Go to the Firebase Console.
   - Create a new project and register a web app.
   - Copy the Firebase configuration provided during setup.

2. **Update Firebase Configuration in the Frontend**
   Replace the placeholder configuration in src/firebase.js with your project's details:

```
import { getFirestore } from "firebase/firestore";

const firebaseConfig = {
    apiKey: "YOUR_API_KEY",
    authDomain: "YOUR_AUTH_DOMAIN",
    projectId: "YOUR_PROJECT_ID",
    storageBucket: "YOUR_STORAGE_BUCKET",
    messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
    appId: "YOUR_APP_ID"
};

const app = initializeApp(firebaseConfig);
export const storage = getStorage(app);
export const db = getFirestore(app);
```

3. **Enable Required Firebase Services**
   - Go to the Firebase Console and enable the following services:
     - **Firestore Database:** Used for storing case reports and images.
     - **Firebase Storage:** Used for storing uploaded skin condition images.
     - **Authentication:** Used for user login and registration.

5.3 Running the Frontend in Development Mode

1. **Start the React Development Server**
   Run the following command to start the frontend development server:

```
npm start
```

   - By default, the server will run on http://localhost:3000.
2. **Verify Frontend Connectivity**
   - Ensure that the backend API is running (from Section 7.1: Backend API).
   - Open the frontend application in a browser and check the following:
     - The **Login Page** loads correctly.
     - The **Image Upload Page** works as expected.
     - API responses are displayed properly after submitting an image for prediction.

5.4 Building the Frontend for Production

Once development is complete, create a production-ready build of the React application:

1. **Generate a Production Build**
   Run the following command to create an optimized build:

```
npm run build
```

   - The build will be created in the build folder inside the react-frontend directory.

2. **Verify the Build**
   To verify that the production build works as expected, use a simple local server:

```
npx serve -s build
```

   - Access the production build at http://localhost:5000.

By completing these steps, the frontend of SkinLens is ready for both development and production environments. You can now connect the frontend to the backend API and Firebase services seamlessly.

Next, proceed to **Section 6: Database Configuration**.

6. Database Configuration

The **SkinLens** project uses **Firebase Firestore** and **Firebase Storage** for database management and image storage. Firestore provides a NoSQL database for storing case reports, user information, and metadata, while Firebase Storage is used for uploading and retrieving images. Follow the steps below to configure the database for the SkinLens project.

6.1 Setting Up Firebase Firestore

1. **Create a Firebase Project**
   - Log in to the Firebase Console.
   - Click **Add Project** and follow the steps to create a new project.
   - Once the project is created, navigate to the **Firestore Database** section.
2. **Enable Firestore**
   - In the Firebase Console, go to **Firestore Database** > **Create Database**.
   - Choose **Production Mode** for secure access or **Test Mode** during initial development.
     - **Production Mode:** Only authenticated users can access the database.
     - **Test Mode:** Allows open access for reading and writing data (use only for development).
3. **Set Firestore Security Rules**
   Update Firestore rules to restrict access and ensure security:
   - For **Development Environment** (Test Mode), use:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

   - For **Production Environment** (Secure Mode), use:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
    match /cases/{caseId} {
      allow read, write: if request.auth != null;
    }
  }
}
```

   - Save the rules and **Publish** them.

6.2 Configuring Firebase Storage

Firebase Storage is used to store images uploaded by users for prediction.

1. **Enable Firebase Storage**
    - In the Firebase Console, navigate to **Storage**.
    - Click **Get Started** and choose a bucket location.
    - Firebase will set up a default storage bucket (e.g., your-project-id.appspot.com).
2. **Set Storage Security Rules**
   Update the rules to allow file uploads during development:
    - For **Development Environment**, use:

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if true;
    }
  }
}
```

    - For **Production Environment**, restrict access to authenticated users:

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

    - Save and **Publish** the rules.
3. **Verify Storage Access**
    - Test uploading a sample image to Firebase Storage.
    - Ensure it is accessible and can be retrieved using the URL.

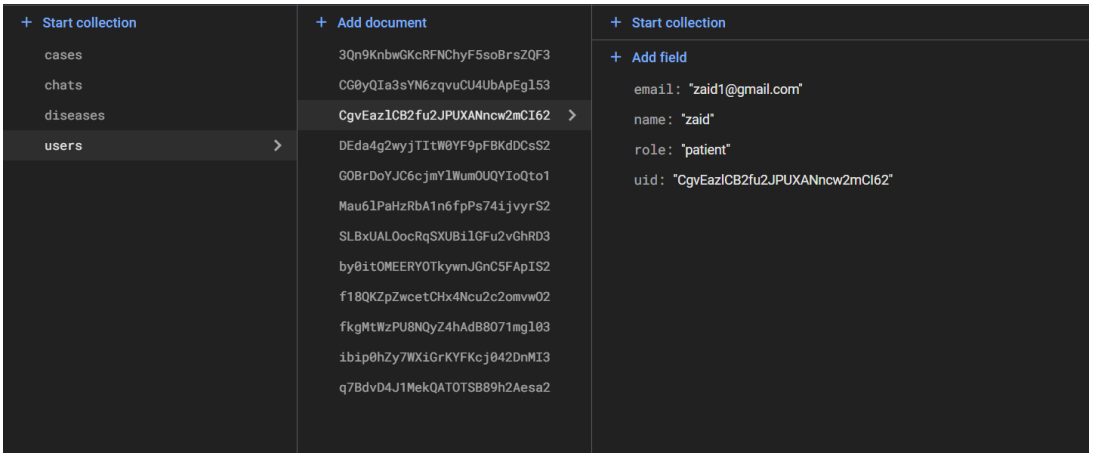
6.3 Integrating Firebase Configuration in Code

1. **Obtain Firebase Configuration**
    - Go to **Project Settings** in the Firebase Console.
    - Under **Your Apps**, select your web app and find the Firebase configuration details.
2. **Update Firebase Configuration in the Project**
    - Replace placeholder configuration in the firebase.js file located in the frontend/src directory.

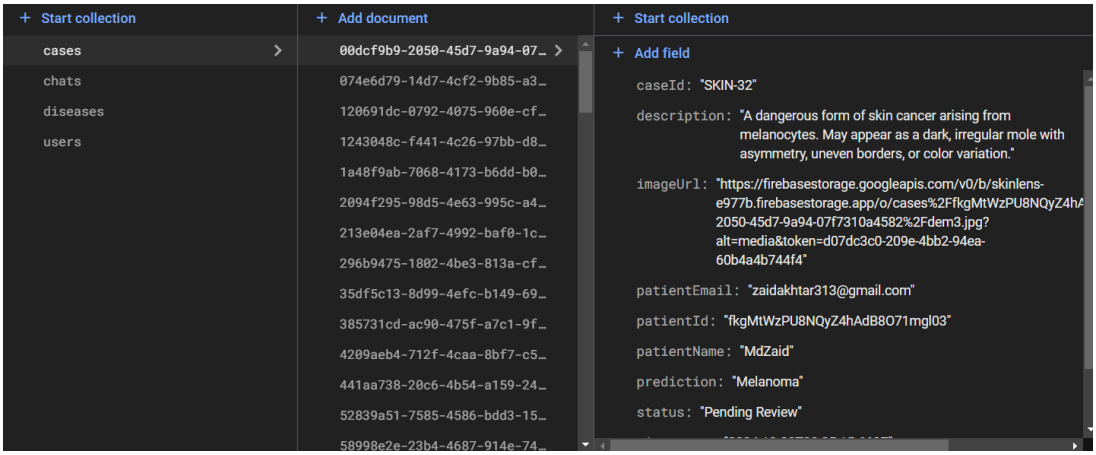
6.4 Firestore Collections and Document Structure

Create the following Firestore collections to store application data:

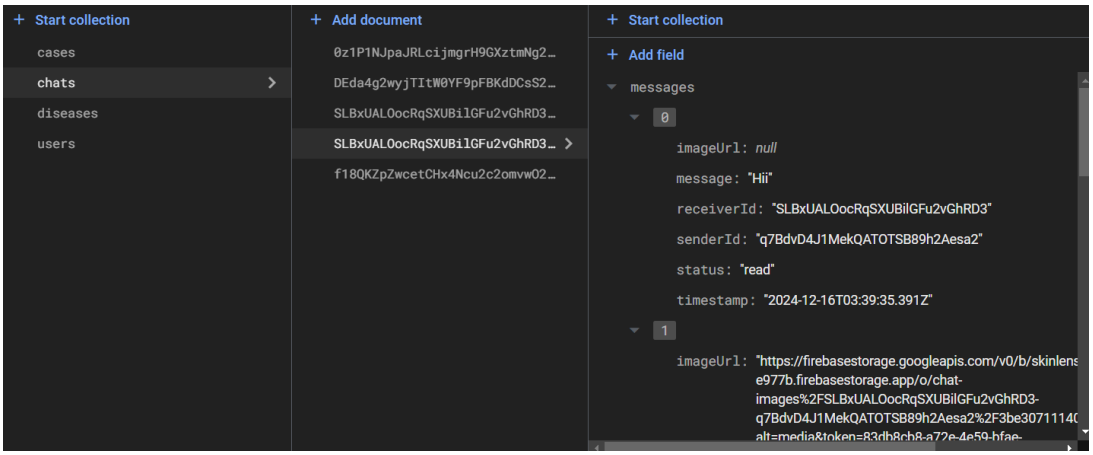
1. **Users Collection**
    - **Purpose:** Stores user authentication details and profile information.
    - **Document Structure:**



2. **Cases Collection**
   - **Purpose:** Stores details of uploaded images and prediction results.
   - **Document Structure:**



3. **Chats Collection** (for chat functionality)
   - **Purpose:** Stores chat messages between patients and dermatologists.
   - **Document Structure:**



6.5 Verifying Database Configuration

1. **Run the Backend**

- Start the Flask server (refer to Section 7.1).
- Verify that API requests can successfully read/write data to Firestore.

2. **Test Database Operations**
   - Test user registration and login.
   - Upload an image and verify it is stored in Firebase Storage.
   - Check that predictions and metadata are correctly stored in Firestore.

With Firebase Firestore and Storage configured, the backend and frontend components of SkinLens can now communicate seamlessly. Proceed to **Section 7: Starting the Application** for instructions on running the API and frontend.

7. Starting the Application

This section explains how to start the **SkinLens** application by running the backend API and frontend UI. After completing the previous setup steps, follow these instructions to ensure the application is fully functional in a local environment.

7.1 Starting the Backend API

The backend API is responsible for handling requests, running predictions, and interacting with the database.

Steps to Start the Backend:

1. **Navigate to the Backend Directory:** Open a terminal and change to the backend folder:

```
cd flask-server
```

2. **Activate the Virtual Environment:**
   - For Linux/macOS:

```
source venv/bin/activate
```

   - For Windows:

```
venv\Scripts\activate
```

3. **Run the Flask Application:** Start the Flask server:

```
python app.py
```

4. **Access the API:**
   - The backend will run on http://localhost:8080 by default.
   - Test the prediction endpoint using a tool like Postman or curl:

```
curl -X POST -H "Content-Type: application/json" -d '{"imageUrl":
"YOUR_IMAGE_URL"}' http://localhost:8080/predict
```

Verify the Backend:

- Check the terminal output for any errors.
- If the server starts successfully, you will see output like:

```
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
```

7.2 Starting the Frontend UI

The frontend is the user interface that allows patients and dermatologists to interact with the SkinLens platform.

Steps to Start the Frontend:

1. **Navigate to the Frontend Directory:** Open a new terminal and switch to the frontend folder:

```
cd frontend
```

2. **Install Dependencies (if not done already):** Ensure all required packages are installed:

```
npm install
```

3. **Start the Frontend Development Server:** Run the development server:

```
npm start
```

4. **Access the Frontend:**
    - Open your browser and navigate to http://localhost:3000.
    - You should see the SkinLens homepage.

Verify the Frontend:

- Ensure the frontend loads without errors.
- Test navigation to features like login, image upload, and viewing reports.

7.3 Testing the Integration

To confirm that the backend and frontend are working together:

1. **Log in as a Patient:**
    - Use the frontend to register or log in with test credentials.
    - Upload an image for diagnosis.
2. **Verify Backend Processing:**
    - Check the Flask terminal for incoming requests and ensure the model processes the image.
3. **Confirm Database Updates:**
    - Verify that the uploaded image and its prediction are stored in Firebase.
4. **Test Dermatologist Features:**
    - Log in as a dermatologist to review reports, add comments, and communicate with patients.

7.4 Debugging Common Issues

- **Backend Not Running:**
  - Ensure the virtual environment is activated.
  - Verify Python dependencies are installed.
- **Frontend Not Loading:**
  - Ensure Node.js is installed and the `npm start` command is executed.
- **Backend and Frontend Communication Errors:**
  - Check if the backend (`http://localhost:8080`) and frontend (`http://localhost:3000`) are running on the correct ports.
- **Database Not Updating:**
  - Verify Firebase configuration in `firebase.js` and backend settings.

7.5 Next Steps

Once the application is running successfully in the local environment:

- Test all user flows thoroughly.
- Proceed to **Section 8: Deployment to Cloud** for instructions on deploying SkinLens to a production environment.

8. Deployment to Cloud

8.1 Firebase Setup

1. **Create Firebase Project**:
   - Go to Firebase Console, create a project, and register a web app.
2. **Update Configuration**:
   - Replace firebaseConfig in frontend/src/firebase.js with your project details.
3. **Enable Services**:
   - Authentication, Firestore, and Storage.
4. **Deploy Frontend**:

```
cd react-frontend
npm run build
firebase deploy
```

8.2 Hosting the Backend

1. **Google App Engine**:
   - Add app.yaml to flask-server:

```
runtime: python310
entrypoint: gunicorn -b :8080 app:app
```

   - Deploy:

```
gcloud app deploy
```

2. **Docker Deployment**:
   - Build Docker images:

```
docker build -t skinlens-backend ./flask-server
docker build -t skinlens-frontend ./react-frontend
```

   - Push images to Docker Hub:

```
docker tag <image> <username>/<repo>:latest
docker push <username>/<repo>:latest
```

   - Use docker-compose.yml for local hosting:

```yaml
version: '3.8'
services:
  backend:
    image: <username>/skinlens-backend:latest
    ports:
      - "8080:8080"
  frontend:
    image: <username>/skinlens-frontend:latest
    ports:
      - "3000:3000"
```

- Start containers:

```
docker-compose up -d
```

8.3 Kubernetes Deployment

1. **Create Kubernetes Deployments**:
   - Backend example:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: skinlens-backend
spec:
  replicas: 2
  template:
    spec:
      containers:
      - name: backend
        image: <username>/skinlens-backend:latest
```

   - Apply:

Bash

```
kubectl apply -f backend-deployment.yaml
kubectl apply -f frontend-deployment.yaml
```

2. **Expose Services**:
   - Use a LoadBalancer or Ingress.

9. Troubleshooting

9.1 Backend Issues

1. **Backend Not Starting**:
   - **Error:** ModuleNotFoundError
     Solution: Install dependencies:

     ```
     cd flask-server
     pip install -r requirements.txt
     ```

   - **Error:** Model file not found.
     Solution: Ensure the model is downloaded to flask-server/models/:

     ```
     wget -P flask-server/models/ https://storage.googleapis.com/skinlens-models/final_model.keras
     ```

2. **Port Conflicts**:
   - **Solution:** Check if the port 8080 is in use:

     ```
     lsof -i :8080
     kill -9 <PID>
     ```

3. **Docker Backend Not Running**:
   - **Solution:** Check container logs:

     ```
     docker logs <container-id>
     ```

9.2 Frontend Issues

1. **Frontend Not Starting**:
   - **Error:** npm command not found
     Solution: Install Node.js:

     ```
     https://nodejs.org/en/download/
     ```

2. **Blank Screen on Frontend**:
   - **Solution:** Verify firebaseConfig in react-frontend/src/firebase.js is correctly configured.
3. **Port Conflicts**:
   - **Solution:** Change the port:

     ```
     npm start -- --port 4000
     ```

9.3 Deployment Issues

1. **Firebase Deployment Fails**:
   - **Solution:** Ensure Firebase CLI is installed:

```
npm install -g firebase-tools
firebase login
```

2. **Docker Image Not Found**:
   - **Solution:** Check image tags and repository:

```
docker images
```

3. **Kubernetes Service Not Exposed**:
   - **Solution:** Verify LoadBalancer status:

```
kubectl get services
```

10. Contact Information

For further assistance, please reach out:

- **Support Email**: skinlensbiased@gmail.com
- **GitHub Repository**: https://github.com/htmw/2024F-Biased.git

We’re here to help! For immediate support, raise an issue on our GitHub page or contact us via email.