Here is one example of the Integration of a Machine Learning Model with reference to our project and how it can be applied.

Frontend

- The services folder can contain the API calls that interact with the machine learning service in the backend. These API requests will include user data (skin type, concerns) and fetch personalized product recommendations from the backend.
- The recommendations based on the machine learning model's results will be displayed on relevant pages using the components defined in the components folder.

Backend

- The routes will define endpoints that send user data to the recommender service.
- The controllers will handle requests, ensuring that the user's input is pre-processed if necessary before sending it to the ML service via HTTP or other communication methods.
- The services folder will house the logic for communicating with the ML microservice, sending relevant data and retrieving the recommended products.
- The models and MongoDB database will store users' skin profiles and interaction history, which the backend can send to the ML model for recommendations.

Recommender (ML Microservice):

- The machine learning model can be deployed as a separate microservice. The model will reside in the recommender/app/ directory.
- service.py will expose API endpoints to accept user data (skinType, concerns) from the backend and return recommended products.
- model.py will load the trained machine learning model and make predictions based on the input data.
- utils.py can be used to preprocess incoming user data and post-process the output.

Docker Compose:

- The docker-compose.yml file can be set up to run the backend and ML microservice in separate containers, making sure that both services communicate efficiently in a local or production environment.

Tools and Frameworks Needed:

Frontend:

- Fetch API: For making HTTP requests from the React frontend to the backend API.

Backend:

- Express: Framework for Node.js for handling API requests and routing the user data to the ML microservice.
- Mongoose: For interacting with MongoDB to store and retrieve user profiles.

Recommender Service:

- Python (FastAPI or Flask): To create the API endpoints for the ML service.
- Scikit-learn or TensorFlow/PyTorch: These will be used for model training and prediction for the ML algorithm.
- Docker: To make containers for the ML service for consistency across environments.