Flask Interface: Design Decisions & User Flow

Why Flask?

Flask is a powerful and flexible micro-framework for building web applications in Python. For machine learning deployment, it offers:

- Full control over routing, backend logic, and rendering
- Seamless integration with deep learning models (Keras, TensorFlow, etc.)
- Template-based UI using Jinja2 for dynamic rendering
- Easy to scale and deploy to platforms like Heroku, Render, or AWS

UI Design Decisions

Element	Decision
Backend Framework	Flask used for modular routing, model prediction, and preprocessing
HTML Template	Jinja2 (`index.html`) for dynamic feedback and clean UI separation
Review Input Field	`<textarea>` element to allow multi-line review input
Button Placement	`<button>` directly below input for natural submission flow
Feedback Display	Display predictions from both models inside a `.results` section
Styling	Custom CSS with dark-themed container, rounded buttons, responsive layout

Preprocessing Pipeline

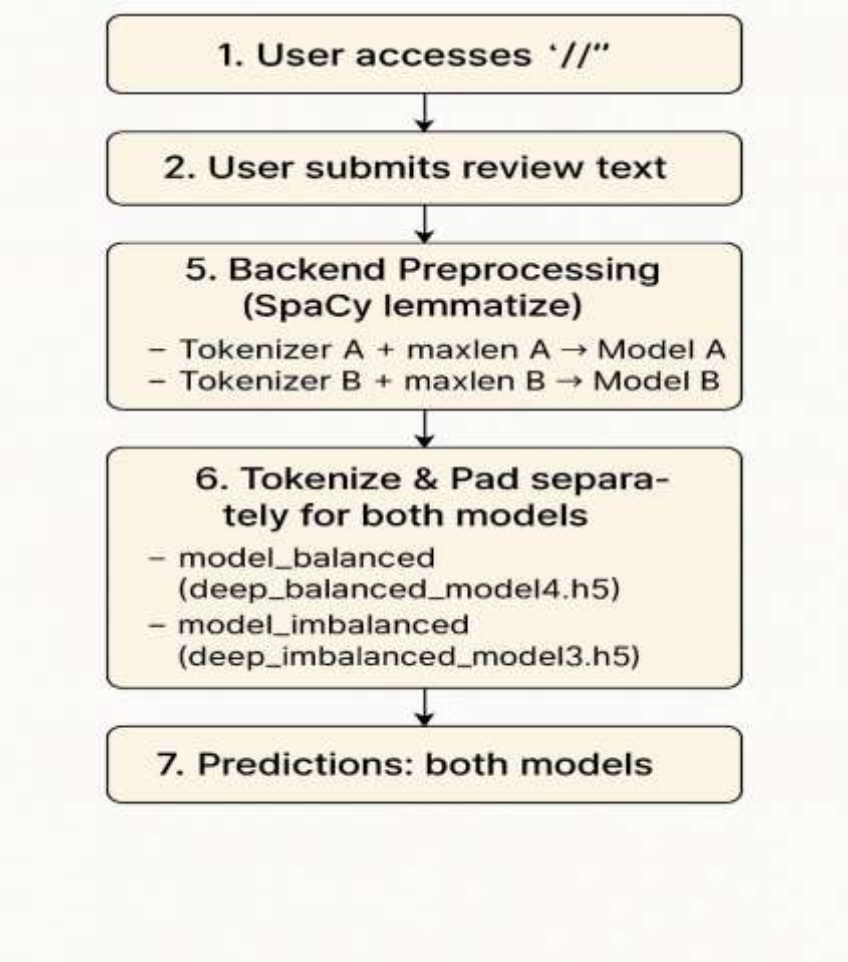
- Preprocessing is handled by a **custom SpaCy pipeline** using lemmatization, lowercasing, and stopword removal (in `utils/preprocess.py`).
- Applies to both models (balanced and imbalanced) for consistency and cleaner predictions.

User Flow

1. **App Launch**
   The user lands on the web page and sees a dark-themed form titled **"Review Rating Prediction"**.
2. **Review Entry**
   A large text box allows the user to enter their product review.
3. **Submit Review**

- Upon clicking **"Predict with Both Models"**:
  - Text is preprocessed via SpaCy (`clean_text`).
  - Tokenized and padded separately using model-specific tokenizers and maxlen values.
  - Predictions made by:
    - `deep_balanced_model4.h5` (trained on balanced data)
    - `deep_imbalanced_model3.h5` (trained on imbalanced data)

4. **Prediction Display**
   - Results from both models are rendered on the same page:
     - **Balanced Model Prediction** shown in a bold paragraph
     - **Imbalanced Model Prediction** shown below that



1. User accesses `/`
The user visits the home page of the Flask web app. The app renders the `index.html` template with an empty form for review input.

2. User submits review text
The user types a product review in the `<textarea>` field and clicks the **"Predict with Both Models"** button to trigger prediction.

3. Backend Preprocessing (SpaCy lemmatize)
The input text is processed using the SpaCy NLP pipeline:

- Converts to lowercase
- Removes stopwords and punctuation
- Keeps only meaningful (alphabetic) tokens
- Applies lemmatization (e.g., "running" → "run")

This ensures consistent, clean input for both models.

4. Tokenizer + maxlen → Model A & Model B

- The preprocessed text is passed through two different tokenizers:
  - `tokenizer_balanced` (trained on balanced dataset)
  - `tokenizer_imbalanced` (trained on imbalanced dataset)
- Each tokenizer converts the text into a sequence of integers.
- The sequences are padded to fixed lengths using model-specific `maxlen` values.

5. Deep Learning Predictions
The padded sequences are fed into two pre-trained deep learning models:

- `deep_balanced_model4.h5` → trained on balanced data
- `deep_imbalanced_model3.h5` → trained on imbalanced data
  Each model outputs a probability vector over 5 classes (for star ratings 1 to 5).

Using `np.argmax(prediction) + 1`, the app selects the most probable class and adjusts it to a 1–5 scale.

6. Output Display
The predicted ratings from both models are displayed on the web page (e.g.,
Balanced Model Prediction: ☆ 4 stars
Imbalanced Model Prediction: ☆ 5 stars).
This gives the user a dual perspective on the review's predicted rating.

Detailed Steps (Backend)

Step	Action
1. User Input	Text entered via `<textarea>`
2. Preprocessing	SpaCy-based lemmatization and cleanup via `spacy_preprocess()`
3. Tokenization + Padding	Model-specific tokenizer + `pad_sequences()` with matching `maxlen` values
4. Deep Model Prediction	Predictions from Keras models loaded using `load_model()`
5. Output Rendering	Renders to `index.html` using Jinja2 with values `prediction_balanced` and `prediction_imbalanced`

Additional Notes

- **Custom Preprocessing**: Built with SpaCy for consistent and intelligent text normalization.
- **Modular Structure**: Tokenizers, models, and max lengths are loaded separately for easy maintenance.
- **Template Simplicity**: Clean layout using inline CSS and native Bootstrap-like styling (without dependency).
- **Model Consistency**: Predictions are adjusted with `np.argmax() + 1` to match the 1–5 star rating system.

Code for the app.py

```
from flask import Flask, render_template, request
import pickle
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
import spacy

app = Flask(__name__)

# Load SpaCy
nlp = spacy.load("en_core_web_sm")

# Load models
model_balanced = load_model("../Models/deep_balanced_model4.h5")
model_imbalanced = load_model("../Models/deep_imbalanced_model3.h5")

# Load tokenizers and max_len
with open("../Models/deep_tokenizer_balanced.pkl", "rb") as f:
    tokenizer_balanced = pickle.load(f)
with open("../Models/maxlen_balanced1.pkl", "rb") as f:
    maxlen_balanced = pickle.load(f)
```

```
with open("../Models/deep_tokenizer_imbalanced.pkl", "rb") as f:
    tokenizer_imbalanced = pickle.load(f)
with open("../Models/maxlen_imbalanced1.pkl", "rb") as f:
    maxlen_imbalanced = pickle.load(f)

def spacy_preprocess(text):
    doc = nlp(text)
    tokens = [token.lemma_ for token in doc if token.is_alpha and not token.is_stop]
    return " ".join(tokens)

@app.route("/", methods=["GET", "POST"])
def index():
    prediction_balanced = None
    prediction_imbalanced = None
    review_text = ""

    if request.method == "POST":
        review_text = request.form["review"]
        cleaned_text = spacy_preprocess(review_text)

        # Tokenize and pad for both models
        seq_balanced = tokenizer_balanced.texts_to_sequences([cleaned_text])
        padded_balanced = pad_sequences(seq_balanced, maxlen=maxlen_balanced, padding="post",
truncating="post")
        pred_balanced = model_balanced.predict(padded_balanced)
        prediction_balanced = np.argmax(pred_balanced) + 1  # add 1 to match 1–5 scale

        seq_imbalanced = tokenizer_imbalanced.texts_to_sequences([cleaned_text])
        padded_imbalanced = pad_sequences(seq_imbalanced, maxlen=maxlen_imbalanced,
padding="post", truncating="post")
        pred_imbalanced = model_imbalanced.predict(padded_imbalanced)
        prediction_imbalanced = np.argmax(pred_imbalanced) + 1

    return render_template("index.html",
                           prediction_balanced=prediction_balanced,
                           prediction_imbalanced=prediction_imbalanced,
                           review_text=review_text)

if __name__ == "__main__":
    app.run(debug=True)
```

code for spacy preprocesing(preprocess.py)

```
import re
import spacy
nlp = spacy.load("en_core_web_sm")

def clean_text(text):
    text = text.strip().lower()
    doc = nlp(text)
    tokens = [
```

```
        token.lemma_
        for token in doc
        if not token.is_stop and not token.is_space and token.is_alpha
    ]
    return " ".join(tokens)
```

code for frontend:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Review Rating Prediction</title>
    <style>
        body {
            font-family: 'Segoe UI', sans-serif;
            background: #f0f2f5;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }

        .container {
            background: #ffffff;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
            width: 500px;
        }

        h2 {
            text-align: center;
            color: #333;
        }

        textarea {
            width: 100%;
            height: 120px;
            padding: 10px;
            margin-top: 10px;
            font-size: 14px;
            border: 1px solid #ccc;
            border-radius: 6px;
            resize: vertical;
        }

        button {
            width: 100%;
            padding: 10px;
            margin-top: 12px;
```

```
            font-size: 16px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 6px;
            cursor: pointer;
        }

        button:hover {
            background-color: #45a049;
        }

        .results {
            margin-top: 20px;
            text-align: center;
        }

        .results span {
            font-weight: bold;
            color: #333;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Review Rating Prediction</h2>
        <form method="POST">
            <textarea name="review" placeholder="Enter your review..." required>{{ review_text if
review_text else '' }}</textarea>
            <button type="submit">Predict with Both Models</button>
        </form>
        {% if prediction_balanced is not none and prediction_imbalanced is not none %}
        <div class="results">
            <p>Balanced Model Prediction: <span>{{ prediction_balanced }} stars</span></p>
            <p>Imbalanced Model Prediction: <span>{{ prediction_imbalanced }} stars</span></p>
        </div>
        {% endif %}
    </div>
</body>
</html>
```