**1. Imports and Setup**

from flask import Flask, render\_template, request, redirect, url\_for, flash, session, jsonify

from flask\_sqlalchemy import SQLAlchemy

from flask\_bcrypt import Bcrypt

from chatbot\_model import get\_response

import os

* Flask → Core of your web application.
* render\_template → To render HTML files.
* request → To handle form and JSON data sent by users.
* redirect & url\_for → Redirect users to different pages.
* flash → Show temporary messages (like “Login successful”).
* session → Keep track of logged-in users.
* jsonify → Send JSON responses for API routes (like chatbot responses).
* SQLAlchemy → ORM for working with databases in Python.
* Bcrypt → Hashing passwords securely.
* get\_response → Function from chatbot\_model.py that generates the chatbot’s reply.
* os → General-purpose operating system operations (not actively used here yet).

**2. Flask App Configuration**

app = Flask(\_\_name\_\_)

app.secret\_key = "secret123"

* Creates the Flask app.
* secret\_key is used to encrypt session data and flash messages.
* **Important:** For production, you should use a stronger, random secret key.

**3. Database Setup**

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///database.db'

db = SQLAlchemy(app)

bcrypt = Bcrypt(app)

* Sets up SQLite database database.db.
* db = SQLAlchemy(app) → Connects Flask with the database.
* bcrypt = Bcrypt(app) → Sets up password hashing.

**4. User Model**

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(150), nullable=False, unique=True)

password = db.Column(db.String(150), nullable=False)

* Defines a **User table** with three columns:
  + id → Unique primary key.
  + username → Must be unique.
  + password → Stores hashed passwords.

with app.app\_context():

db.create\_all()

* Creates the database and table if it doesn’t exist.

**5. Home Page (Chatbot)**

@app.route("/")

def home():

if "user\_id" not in session:

return redirect(url\_for("login"))

return render\_template("index.html")

* Checks if the user is logged in (user\_id in session).
* If not, redirects to the login page.
* If logged in, shows the chatbot page index.html.

**6. Registration**

@app.route("/register", methods=["GET", "POST"])

def register():

if request.method == "POST":

username = request.form.get("username")

password = request.form.get("password")

...

* Handles GET (show form) and POST (submit form).
* Checks if the username already exists:

existing\_user = User.query.filter\_by(username=username).first()

* Hashes the password using bcrypt:

hashed\_password = bcrypt.generate\_password\_hash(password).decode('utf-8')

* Saves the new user to the database:

db.session.add(new\_user)

db.session.commit()

* Redirects to login page with a success message.

**7. Login**

@app.route("/login", methods=["GET", "POST"])

def login():

if request.method == "POST":

username = request.form.get("username")

password = request.form.get("password")

user = User.query.filter\_by(username=username).first()

if user and bcrypt.check\_password\_hash(user.password, password):

session["user\_id"] = user.id

session["username"] = user.username

flash("Login successful!", "success")

return redirect(url\_for("home"))

else:

flash("Invalid username or password", "error")

return redirect(url\_for("login"))

return render\_template("login.html")

* Checks credentials.
* If valid, stores user info in session (session["user\_id"]) to keep them logged in.
* If invalid, shows error using flash.

**8. Logout**

@app.route("/logout")

def logout():

session.clear()

flash("You have been logged out.", "info")

return redirect(url\_for("login"))

* Clears session (logs out the user).
* Redirects to login page.

**9. Chatbot API**

@app.route("/get", methods=["POST"])

def chatbot\_response():

if "user\_id" not in session:

return jsonify({"response": "Please log in to chat with me!"})

user\_message = request.json["message"]

bot\_reply = get\_response(user\_message)

return jsonify({"response": bot\_reply})

* Receives a JSON POST request with user’s message.
* Calls get\_response() from chatbot\_model.py to generate a reply.
* Returns a JSON response to the frontend.

**10. Running the App**

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

* Starts the Flask development server.
* debug=True → Automatically reloads code on changes and shows error messages.

**Summary of Flow**

1. User registers → credentials saved in DB.
2. User logs in → session created.
3. User accesses / → sees chatbot page.
4. Chatbot messages are sent to /get → response returned via JSON.
5. User logs out → session cleared.