**ANALYZING AND FORECASTING CONSUMER TRENDS FOR TECH PRODUCTS IN 2025**

**1. Project Overview and Scope**

* **Title**: Ứng dụng phân tích và dự báo xu hướng tiêu dùng sản phẩm công nghệ 2025
* **Objective**: Analyze consumer trends for tech products and forecast how they will impact various industries in 2025. The project will focus on 28 specific topics provided, utilizing data from sources like IDC, Statista, and others.
* **App Features**:
  + User and Admin login functionality
  + Data visualization dashboards with interactive elements
  + Chatbot integration to assist users in querying data and trends by industry
  + Prediction of consumer trends for each industry
* **Technologies**:
  + **Frontend**: ReactJS or Angular
  + **Backend**: Node.js or Python (Flask/Django)
  + **Database**: MongoDB/PostgreSQL/MySQL
  + **Visualization**: Plotly/D3.js for interactive graphs
  + **Chatbot**: Dialogflow, Rasa, or Microsoft Bot Framework

**2. Data Collection and Scraping**

* **Sources**: IDC, Statista, and web scraping other relevant sites.
* **Libraries for Scraping**:
  + **BeautifulSoup** and **Selenium** for web scraping
  + **pandas** for data handling
* **Code Example**: Use BeautifulSoup to scrape Statista data.

import requests

from bs4 import BeautifulSoup

import pandas as pd

url = 'https://www.statista.com/statistics/some-link/'

headers = {'User-Agent': 'Mozilla/5.0'}

response = requests.get(url, headers=headers)

soup = BeautifulSoup(response.content, 'html.parser')

# Extracting specific data

table = soup.find('table', {'class': 'stat-table'})

data = []

for row in table.find\_all('tr'):

columns = row.find\_all('td')

data.append([col.text for col in columns])

df = pd.DataFrame(data, columns=['Year', 'Tech Consumption'])

df.to\_csv('tech\_trends.csv', index=False)

**3. Data Preprocessing and EDA**

* **Cleaning Data**: Handle missing data, remove duplicates, convert data types.
* **Exploratory Data Analysis (EDA)**: Understand distributions, outliers, and correlations using matplotlib and seaborn.

import seaborn as sns

import matplotlib.pyplot as plt

# Example: Plot distribution of consumption over years

sns.lineplot(x='Year', y='Tech Consumption', data=df)

plt.title('Tech Consumption Trends Over the Years')

plt.show()

**4. Data Visualization with Interactive Dashboards**

* Use **Plotly Dash** or **Streamlit** to create an interactive web app with data visualization.
* Example: Interactive dashboard for visualizing tech consumption by industry.

import plotly.express as px

fig = px.line(df, x='Year', y='Tech Consumption', title='Tech Consumption Trends')

fig.show()

* **Streamlit Example**:

import streamlit as st

import pandas as pd

import plotly.express as px

df = pd.read\_csv('tech\_trends.csv')

st.title('Tech Consumption Dashboard')

year = st.slider('Select Year', min\_value=int(df['Year'].min()), max\_value=int(df['Year'].max()), step=1)

filtered\_data = df[df['Year'] == year]

fig = px.bar(filtered\_data, x='Industry', y='Consumption')

st.plotly\_chart(fig)

**5. Backend for Login, Admin, and User Management**

* Use **Flask** or **Django** to handle the user and admin authentication, session management, and secure APIs for data interaction.

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

from flask\_sqlalchemy import SQLAlchemy

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

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

db = SQLAlchemy(app)

class User(db.Model):

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

username = db.Column(db.String(50), unique=True)

password = db.Column(db.String(50))

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

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

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

if user:

session['username'] = user.username

return redirect(url\_for('dashboard'))

return render\_template('login.html')

@app.route('/dashboard')

def dashboard():

return 'Welcome to the Tech Trend Dashboard!'

**6. Chatbot Integration for Consumer Trend Assistance**

* Use **Dialogflow** or **Rasa** to build a chatbot that allows users to ask about trends for each industry. You will need to train the chatbot on key phrases related to consumer trends and integrate it with your app.
* **Dialogflow Integration** (sample request/response setup):

import dialogflow

from google.api\_core.exceptions import InvalidArgument

project\_id = 'your-project-id'

session\_id = 'current-user-session-id'

language\_code = 'en'

session\_client = dialogflow.SessionsClient()

session = session\_client.session\_path(project\_id, session\_id)

text\_input = dialogflow.types.TextInput(text='Tell me about tech trends in 2025', language\_code=language\_code)

query\_input = dialogflow.types.QueryInput(text=text\_input)

response = session\_client.detect\_intent(session=session, query\_input=query\_input)

print(response.query\_result.fulfillment\_text)

**7. Prediction of Consumer Trends Using Machine Learning**

* **Predictive Models**: Use models like **Linear Regression**, **Decision Trees**, or **ARIMA** to forecast trends.
* **Example**: Linear Regression on tech consumption data.

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

X = df[['Year']]

y = df['Tech Consumption']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

model = LinearRegression()

model.fit(X\_train, y\_train)

predictions = model.predict(X\_test)

plt.scatter(X\_test, y\_test)

plt.plot(X\_test, predictions, color='red')

plt.show()

**8. Project Structure**

* **Frontend**: Handles the UI, graphs, and interactivity.
* **Backend**: Handles API requests, authentication, and database management.
* **Chatbot**: Provides consumer trend assistance based on user queries.
* **Database**: Stores user information, consumption data, and prediction results.
* **Machine Learning Models**: Predict trends for each industry and visualize predictions in the dashboard.

**Next Steps**

1. Set up your environment with Flask/Django for the backend and ReactJS for the frontend.
2. Implement the data scraping and preprocessing steps.
3. Build the UI for users and admins, ensuring login functionality.
4. Implement the chatbot for user interaction with industry trends.
5. Visualize data trends with interactive charts and deploy the app for user access.

We'll use **Flask** for the backend and **ReactJS** for the frontend as an example.

**1. Project Folder Structure**

tech-trend-analysis/

│

├── backend/ # Backend application with Flask

│ ├── app.py # Flask entry point

│ ├── models.py # Database models

│ ├── routes.py # API routes for login, admin, data, etc.

│ ├── templates/ # HTML templates (for login page, if not using React)

│ └── static/ # Static files (CSS, JS, images)

│

├── frontend/ # Frontend application with ReactJS

│ ├── public/ # Public files (index.html)

│ ├── src/ # Source files for ReactJS

│ │ ├── components/ # React components

│ │ ├── pages/ # Main page components (Dashboard, Login, etc.)

│ │ ├── App.js # Main React file

│ │ ├── index.js # React entry point

│ │ └── services/ # API call handlers

│ └── package.json # React dependencies

│

├── chatbot/ # Chatbot integration files

│ ├── intents.json # Intents and responses for chatbot

│ └── chatbot\_integration.py # Flask chatbot integration

│

├── database/ # Database management (SQL, migrations)

│ └── create\_tables.sql # SQL scripts to create tables

│

├── requirements.txt # Python dependencies

├── README.md # Project documentation

└── .env # Environment variables

**2. Backend (Flask) Setup**

**app.py** (Flask entry point):

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

from flask\_cors import CORS

from routes import setup\_routes

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

CORS(app)

# Configuration

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

app.config['SECRET\_KEY'] = 'your-secret-key'

# Database

db = SQLAlchemy(app)

# Setup routes

setup\_routes(app)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**models.py** (Database models):

from app import db

class User(db.Model):

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

username = db.Column(db.String(50), unique=True, nullable=False)

password = db.Column(db.String(50), nullable=False)

role = db.Column(db.String(20), nullable=False) # 'user' or 'admin'

class TrendData(db.Model):

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

industry = db.Column(db.String(100), nullable=False)

year = db.Column(db.Integer, nullable=False)

consumption = db.Column(db.Float, nullable=False)

**routes.py** (API routes for login and data):

from flask import request, jsonify

from models import User, TrendData

from app import db

def setup\_routes(app):

# Login route

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

def login():

data = request.get\_json()

user = User.query.filter\_by(username=data['username'], password=data['password']).first()

if user:

return jsonify({"message": "Login successful", "role": user.role})

return jsonify({"message": "Invalid credentials"}), 401

# Data fetching route

@app.route('/get\_trends', methods=['GET'])

def get\_trends():

industry = request.args.get('industry')

trends = TrendData.query.filter\_by(industry=industry).all()

return jsonify([{"year": t.year, "consumption": t.consumption} for t in trends])

# Admin route to add data

@app.route('/add\_trend', methods=['POST'])

def add\_trend():

data = request.get\_json()

new\_trend = TrendData(industry=data['industry'], year=data['year'], consumption=data['consumption'])

db.session.add(new\_trend)

db.session.commit()

return jsonify({"message": "Trend added successfully"})

**3. Frontend (ReactJS) Setup**

**src/App.js**:

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import LoginPage from './pages/LoginPage';

import Dashboard from './pages/Dashboard';

import AdminPage from './pages/AdminPage';

function App() {

return (

<Router>

<Switch>

<Route path="/login" component={LoginPage} />

<Route path="/dashboard" component={Dashboard} />

<Route path="/admin" component={AdminPage} />

</Switch>

</Router>

);

}

export default App;

**src/pages/LoginPage.js**:

import React, { useState } from 'react';

import axios from 'axios';

import { useHistory } from 'react-router-dom';

const LoginPage = () => {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const history = useHistory();

const handleLogin = async () => {

try {

const response = await axios.post('http://localhost:5000/login', { username, password });

if (response.data.role === 'admin') {

history.push('/admin');

} else {

history.push('/dashboard');

}

} catch (error) {

alert('Login failed');

}

};

return (

<div>

<h1>Login</h1>

<input type="text" value={username} onChange={e => setUsername(e.target.value)} placeholder="Username" />

<input type="password" value={password} onChange={e => setPassword(e.target.value)} placeholder="Password" />

<button onClick={handleLogin}>Login</button>

</div>

);

};

export default LoginPage;

**src/pages/Dashboard.js**:

import React, { useState, useEffect } from 'react';

import axios from 'axios';

const Dashboard = () => {

const [trends, setTrends] = useState([]);

useEffect(() => {

const fetchTrends = async () => {

const response = await axios.get('http://localhost:5000/get\_trends?industry=Tech');

setTrends(response.data);

};

fetchTrends();

}, []);

return (

<div>

<h1>Tech Industry Trends</h1>

<ul>

{trends.map((trend, index) => (

<li key={index}>{trend.year}: {trend.consumption}</li>

))}

</ul>

</div>

);

};

export default Dashboard;

**src/pages/AdminPage.js**:

import React, { useState } from 'react';

import axios from 'axios';

const AdminPage = () => {

const [industry, setIndustry] = useState('');

const [year, setYear] = useState('');

const [consumption, setConsumption] = useState('');

const handleAddTrend = async () => {

try {

await axios.post('http://localhost:5000/add\_trend', { industry, year, consumption });

alert('Trend added successfully');

} catch (error) {

alert('Failed to add trend');

}

};

return (

<div>

<h1>Add New Tech Trend</h1>

<input type="text" value={industry} onChange={e => setIndustry(e.target.value)} placeholder="Industry" />

<input type="number" value={year} onChange={e => setYear(e.target.value)} placeholder="Year" />

<input type="number" value={consumption} onChange={e => setConsumption(e.target.value)} placeholder="Consumption" />

<button onClick={handleAddTrend}>Add Trend</button>

</div>

);

};

export default AdminPage;

**4. Run the Application**

1. **Backend (Flask)**:
   * Install dependencies from requirements.txt:

pip install -r requirements.txt

* + Run the backend:

python app.py

1. **Frontend (ReactJS)**:
   * Install dependencies:

npm install

* + Run the frontend:

npm start

1. **Access the app**:
   * Open the browser and navigate to http://localhost:3000/login for the frontend interface.

========================================================

**Backend: Chatbot Integration (Flask)**

**========================================================**

**chatbot/chatbot\_integration.py**:

from flask import Flask, request, jsonify

import dialogflow\_v2 as dialogflow

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

@app.route('/chatbot', methods=['POST'])

def chatbot\_response():

data = request.get\_json()

user\_message = data['message']

# Dialogflow session setup

session\_client = dialogflow.SessionsClient()

session = session\_client.session\_path('your-project-id', 'session-id')

text\_input = dialogflow.types.TextInput(text=user\_message, language\_code='en')

query\_input = dialogflow.types.QueryInput(text=text\_input)

response = session\_client.detect\_intent(session=session, query\_input=query\_input)

chatbot\_reply = response.query\_result.fulfillment\_text

return jsonify({"reply": chatbot\_reply})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Make sure to replace 'your-project-id' with your actual Dialogflow project ID.

**2. Database Management (SQL Script)**

**database/create\_tables.sql**:

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT NOT NULL UNIQUE,

password TEXT NOT NULL,

role TEXT NOT NULL

);

CREATE TABLE trends (

id INTEGER PRIMARY KEY AUTOINCREMENT,

industry TEXT NOT NULL,

year INTEGER NOT NULL,

consumption FLOAT NOT NULL

);

INSERT INTO users (username, password, role) VALUES ('admin', 'admin123', 'admin');

Run this script to create the necessary tables for the project.

**3. Frontend Components**

Here’s the code for the remaining React components.

**src/components/Chatbot.js** (Chatbot UI Component):

import React, { useState } from 'react';

import axios from 'axios';

const Chatbot = () => {

const [message, setMessage] = useState('');

const [reply, setReply] = useState('');

const handleSubmit = async () => {

const response = await axios.post('http://localhost:5000/chatbot', { message });

setReply(response.data.reply);

};

return (

<div>

<h2>Chat with TrendBot</h2>

<input

type="text"

value={message}

onChange={(e) => setMessage(e.target.value)}

placeholder="Ask about trends"

/>

<button onClick={handleSubmit}>Send</button>

<p>Bot Reply: {reply}</p>

</div>

);

};

export default Chatbot;

**src/pages/HomePage.js** (Home Page):

import React from 'react';

import { Link } from 'react-router-dom';

import Chatbot from '../components/Chatbot';

const HomePage = () => {

return (

<div>

<h1>Welcome to the Tech Trend Analysis Platform</h1>

<p>Explore the latest trends in the tech industry.</p>

<Link to="/dashboard">Go to Dashboard</Link>

<Chatbot />

</div>

);

};

export default HomePage;

**src/services/api.js** (API Call Handlers):

import axios from 'axios';

const API\_URL = 'http://localhost:5000';

export const login = async (username, password) => {

const response = await axios.post(`${API\_URL}/login`, { username, password });

return response.data;

};

export const getTrends = async (industry) => {

const response = await axios.get(`${API\_URL}/get\_trends?industry=${industry}`);

return response.data;

};

export const addTrend = async (data) => {

const response = await axios.post(`${API\_URL}/add\_trend`, data);

return response.data;

};

**4. .env File**

Create an .env file to store your environment variables.

**.env**:

SECRET\_KEY=your-secret-key

SQLALCHEMY\_DATABASE\_URI=sqlite:///tech\_trends.db

**5. Frontend Router Update**

In your App.js, update the routes to include the **HomePage** and **Chatbot** components:

**src/App.js**:

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import LoginPage from './pages/LoginPage';

import Dashboard from './pages/Dashboard';

import AdminPage from './pages/AdminPage';

import HomePage from './pages/HomePage';

function App() {

return (

<Router>

<Switch>

<Route path="/login" component={LoginPage} />

<Route path="/dashboard" component={Dashboard} />

<Route path="/admin" component={AdminPage} />

<Route path="/" component={HomePage} />

</Switch>

</Router>

);

}

export default App;

**6. Final Setup**

1. **Backend**:
   * Install Dialogflow and other required Python packages:

pip install flask dialogflow google-api-python-client flask-sqlalchemy

1. **Frontend**:
   * Install React and Axios for the API calls:

npm install axios react-router-dom

1. **Run Backend**:

python app.py

1. **Run Frontend**:

npm start

Access the app at http://localhost:3000 and try interacting with the chatbot on the **Home Page** or exploring the dashboard and admin pages.