**SKILLSTACK- TRACKER**

Al-Ameen S

Full Stack Engineer Role

**INTRODUCTION**

SkillStack – Tracker is a straightforward and intuitive web application designed to manage and monitor your personal skill development journey. It helps users stay focused, motivated, and consistent by tracking progress across various learning activities such as courses, tutorials, and certifications.

The platform allows users to add learning goals with details like skill name, resource type, platform, and progress status. It provides an organized dashboard view where users can visualize their growth and analyze their learning patterns effectively.

Built using **ReactJS** for the frontend and **Flask (Python)** for the backend, SkillStack ensures a smooth and responsive experience. **SQLite** is used as the database for efficient data storage and management. The application aims to encourage continuous learning by simplifying how users plan, track, and achieve their skill goals.

**OBJECTIVES**

* To help users **track and manage their personal learning goals** efficiently.
* To provide a **centralized platform** for recording courses, tutorials, and certifications.
* To offer **progress monitoring**, so users can see how much they’ve learned over time.
* To display **visual insights** via a simple dashboard for better learning decisions.

**FEATURES**

* **Add Learning Goal**: Easily add courses, videos, or articles you’re currently working on.
* **Track Your Progress**: Stay on top of your learning by marking goals as started, in-progress, or completed.
* **Keep Notes & Details**: Save notes, log hours spent, and rate the difficulty of each skill
* **Genarate Summary**: Use Google Gemini AI to generate quick, one-sentence summaries of your notes.
* **Delete**: Remove skills you’re no longer pursuing with a simple delete button.
* **Dashboard**: Visualize skill growth with interactive insight,including progress tracking,hours spent and category-wise breakdown.

**TECH REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Component** | **Tool / Technology** | **Purpose** |
| Frontend | ReactJS | Build the user interface, including the dashboard for skill tracking and insights |
| Frontend | Node.js 16+ | Run the React development server |
| Backend | Python 3.10+ | Backend programming |
| Backend | Flask | Build APIs to supply dashboard data and manage user requests |
| Database | SQLite | Store user data, including skills, progress, and notes |
| Development Environment / Tools | VS Code / IDE of choice | Coding and project management |
| Operating System | Windows 10 / Linux / macOS | Platform support |

**PROJECT SETUP AND EXECUTION**

**Backend Setup:**

1. Open the terminal and navigate to the backend folder:

* cd backend

1. Install all the required Python dependencies:

* pip install -r requirements.txt

1. Create a .env file in the backend folder and add your Google API key:

* GOOGLE\_API\_KEY='your\_key\_here'

1. Run the Flask server:

* flask run

**Frontend Setup:**

1. Open a new terminal and navigate to the frontend folder:

* cd frontend

1. Install all dependencies:

* npm install

1. Start the React server:

* npm start

**CODE STRUCTURE**

import os

import requests

from flask import Flask, request, jsonify

from flask\_sqlalchemy import SQLAlchemy

from flask\_cors import CORS

from dotenv import load\_dotenv

import google.generativeai as genai

load\_dotenv()

api\_key = os.getenv("GOOGLE\_API\_KEY")

print(f"Loaded API key partial: {api\_key[:10] if api\_key else 'None'}")

if api\_key:

genai.configure(api\_key=api\_key)

else:

print("Google API key not found")

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

CORS(app)

basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///' + os.path.join(basedir, 'instance', 'skills.db')

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

os.makedirs(os.path.join(basedir, 'instance'), exist\_ok=True)

class Skill(db.Model):

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

skill\_name = db.Column(db.String(100), nullable=False)

resource\_type = db.Column(db.String(50))

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

progress = db.Column(db.String(20), default='started')

hours\_spent = db.Column(db.Float, default=0)

difficulty = db.Column(db.Integer, default=1)

notes = db.Column(db.Text)

def to\_dict(self):

return {key: getattr(self, key) for key in self.\_\_table\_\_.c.keys()}

@app.before\_request

def create\_all\_tables():

db.create\_all()

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

def handle\_skills():

"""Handles getting all skills and adding a new one."""

if request.method == 'POST':

data = request.json

new\_skill = Skill(

skill\_name=data['skill\_name'],

resource\_type=data['resource\_type'],

platform=data['platform'],

notes=data.get('notes')

)

db.session.add(new\_skill)

db.session.commit()

return jsonify(new\_skill.to\_dict()), 201

skills = Skill.query.all()

return jsonify([skill.to\_dict() for skill in skills])

@app.route('/skills/<int:id>', methods=['PUT', 'DELETE'])

def handle\_single\_skill(id):

"""Handles updating or deleting a single skill."""

skill = db.session.get(Skill, id)

if not skill:

return jsonify({"error": "Skill not found"}), 404

if request.method == 'PUT':

data = request.json

for key, value in data.items():

setattr(skill, key, value)

db.session.commit()

return jsonify(skill.to\_dict())

if request.method == 'DELETE':

db.session.delete(skill)

db.session.commit()

return "", 204

@app.route('/summarize-notes', methods=['POST'])

def summarize\_skill\_notes():

"""Uses the Google Gemini API for free summarization."""

api\_key = os.getenv("GOOGLE\_API\_KEY")

if not api\_key:

return jsonify({"error": "AI feature is not configured. Please set GOOGLE\_API\_KEY in .env"}), 500

notes = request.json.get('notes', '')

if not notes.strip():

return jsonify({"summary": "No notes to summarize."})

try:

model = genai.GenerativeModel('models/gemini-flash-latest')

prompt = f"Summarize the following learning notes in one concise sentence: {notes}"

response = model.generate\_content(prompt)

summary = response.text if hasattr(response, 'text') else str(response)

return jsonify({"summary": summary})

except Exception as e:

print(f"Google AI API error: {e}")

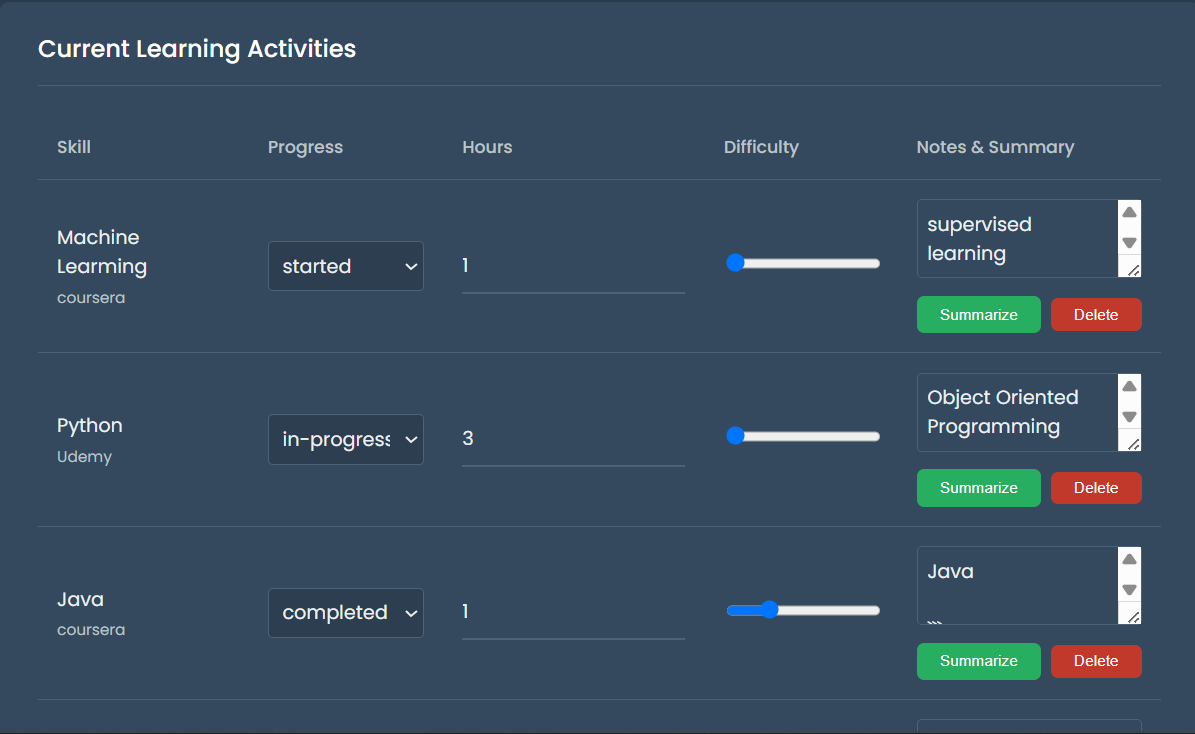
print(f"API Key partial: {api\_key[:10] if api\_key else 'None'}")

return jsonify({"error": f"Failed to generate summary from AI: {str(e)}"}), 500

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

app.run(debug=True)

**RESULT AND SCREENSHOTS**





**CONCLUSION**

The SkillStack – Tracker application provides a simple and effective way to manage and monitor personal skill development. It allows users to track learning goals, progress, and hours spent across multiple platforms. The dashboard offers a visual summary of skill growth, helping users stay motivated and consistent. This project demonstrates the integration of ReactJS frontend, Flask backend, and SQLite database in a real-world application.

**Future enhancements** could include user authentication, advanced analytics, and AI-driven recommendations to further improve the learning experience.