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 **c**entralized 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
2. Install all the required Python dependencies:
    - pip install -r requirements.txt
3. Create a .env file in the backend folder and add your Google API key:
    - GOOGLE_API_KEY='your_key_here'
4. Run the Flask server:
    - flask run

Frontend Setup:

1. Open a new terminal and navigate to the frontend folder:
    - cd frontend
2. Install all dependencies:
    - npm install
3. 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')
```

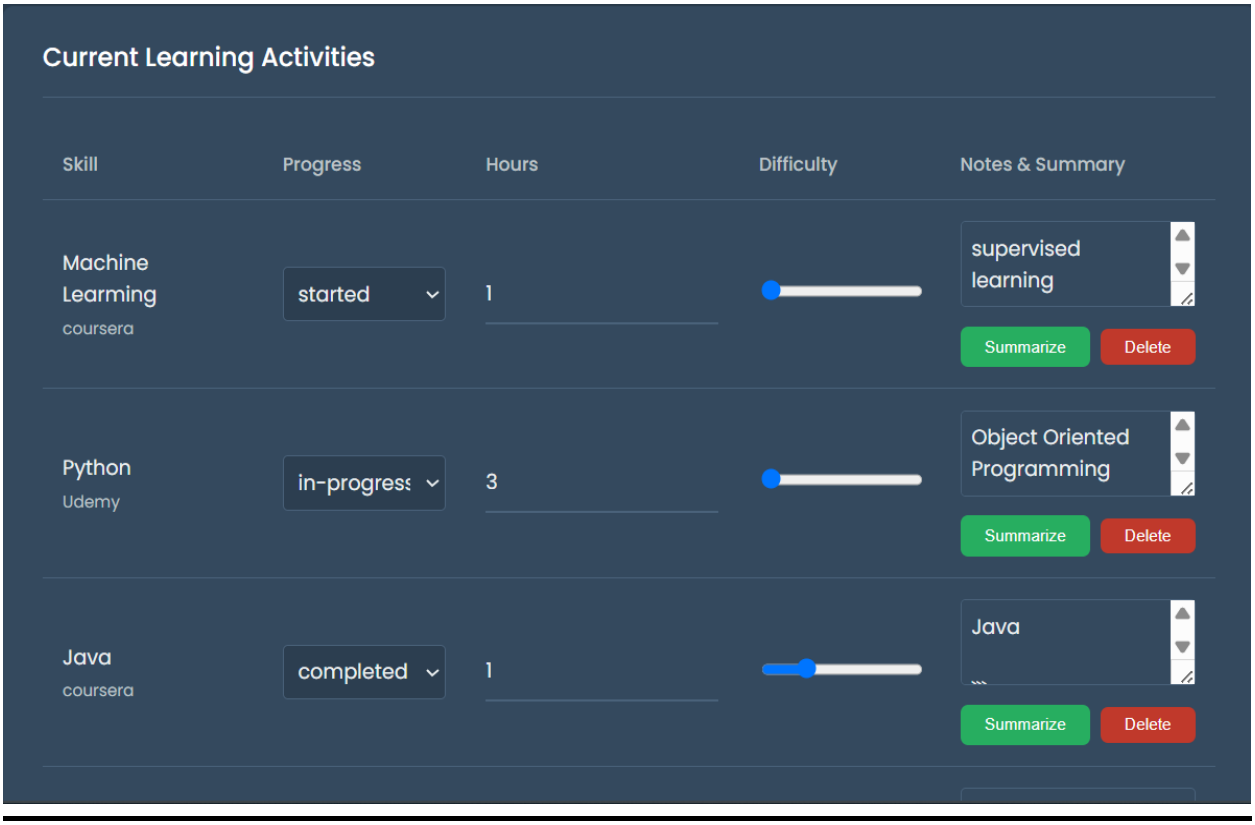
```
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
os.makedirs(os.path.join(basedir, 'instance'), exist_ok=True)
class Skill(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    skill_name = db.Column(db.String(100), nullable=False)
    resource_type = db.Column(db.String(50))
    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)
```

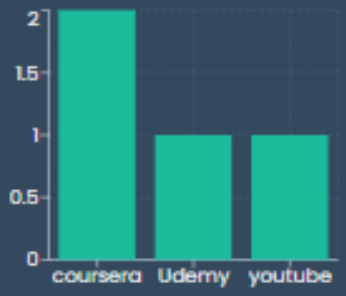
RESULT AND SCREENSHOTS



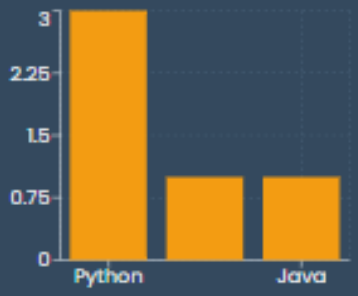
Skill Growth Insights

Progress Distribution

Skills by Platform (Category-wise)

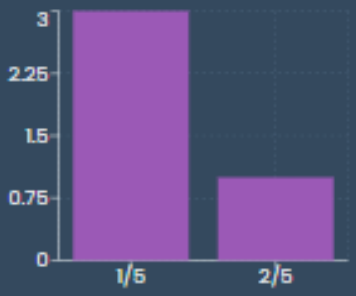


Top 5 Skills by Hours Invested



Skills by Resource Type

Skills by Difficulty Level



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.