Abstract

The daily diary app is a Python-based application designed to provide users with a convenient platform for journaling their daily experiences, thoughts, and emotions. Utilizing Python's versatile programming capabilities and user-friendly interface design, the app enables users to create, update, and organize diary entries seamlessly.

The daily diary app aims to provide users with a personalized and private space to reflect on their daily lives, fostering self-awareness and emotional well-being.

Introduction

This Python-based project aims to offer you a convenient and personalized platform to record your thoughts, experiences, and reflections on a daily basis. Whether you're looking to jot down your daily achievements, express your feelings, or simply keep track of your day-to-day activities, our app provides a user-friendly interface for seamless diary management.

With features like easy entry creation, categorization options, and secure storage, our Daily Diary App strives to be your reliable companion in capturing life's moments.

Literature Survey

A daily diary app allows users to record their thoughts, experiences, and events on a daily basis. Developing such an app requires a thorough understanding of various components, including user interface design, data storage, and user authentication. In this literature survey, we explore existing literature and resources relevant to building a daily diary app using Python.

1. User Interface Design:

- HTML: It provides a simple way to create windows, dialogs, and other graphical elements required for the diary app's user interface.

2. Data Storage:

- SQLite: SQLite is a lightweight, embedded database engine that allows storing and querying data within the application. It's ideal for small-scale applications like a diary app, offering simplicity and efficiency.

3. User Authentication:

- Flask: Flask is a micro web framework for Python, which can be used to implement user authentication features such as login, registration, and session management. It provides flexibility and scalability for handling user accounts in the diary app.

Problem Statement

In today's fast-paced world, it's easy to overlook the importance of reflection and introspection. A daily diary app aims to provide users with a convenient platform to record their thoughts, experiences, and emotions on a daily basis. This Python project will focus on creating a user-friendly diary application where users can log their daily activities, thoughts, and feelings, and later revisit and reflect upon them. The app will allow users to create, edit, and delete diary entries, as well as browse through past entries for self-reflection and personal growth.

Method

1. **Backend with Python**: Use Python's web frameworks like Flask to handle server-side operations. Create endpoints to receive and process poll data.

2. **Database Integration**: Store poll questions, options, and responses in a database like SQLite using Python libraries like SQLite.

3. **Frontend with HTML**: Design the user interface using html to display login screens, windows, dialogs.

4. **Security Measures**: Implement security measures like hashing technique for the password safety and validation to prevent unauthorized access or misuse.

Source code:

```
from flask import Flask, render_template, request, redirect, url_for
import os
import json

app = Flask(__name__)

def load_users():
    if os.path.exists("users.json"):
        with open("users.json", "r") as file:
            return json.load(file)
    else:
        return {}

def save_users(users):
    with open("users.json", "w") as file:
        json.dump(users, file)
def load_diaries(username):
    if os.path.exists(f"{username}_diaries.json"):
        with open(f"{username}_diaries.json", "r") as file:
            return json.load(file)
    else:
        return []
def save_diaries(username, diaries):
    with open(f"{username}_diaries.json", "w") as file:
```

```
        json.dump(diaries, file)

@app.route("/", methods=["GET", "POST"])
def register():
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        users = load_users()
        if username not in users:
            users[username] = {"password": password}
            save_users(users)
            return redirect(url_for("login"))
        else:
            return "Username already exists!"
    return render_template("register.html")

@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        users = load_users()
        if username in users and users[username]["password"] == password:
            return redirect(url_for("main_page", username=username))
        else:
            return "Invalid username or password!"
    return render_template("login.html")

@app.route("/<username>/main_page")
def main_page(username):
    return render_template("main_page.html", username=username)

@app.route("/<username>/add_diary", methods=["GET", "POST"])
def add_diary(username):
    if request.method == "POST":
        date = request.form["date"]
        title = request.form["title"]
        content = request.form["content"]
        diaries = load_diaries(username)
        diaries.append({"date": date, "title": title, "content": content})
        save_diaries(username, diaries)
        return redirect(url_for("main_page", username=username))
```

```
    return render_template("add_diary.html")

@app.route("/<username>/previous_diaries")
def previous_diaries(username):
    diaries = load_diaries(username)
    return render_template("previous_diaries.html", diaries=diaries)

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

Result and Discussion

The Daily Diary App was successfully developed and tested, meeting the project's objectives. The key features implemented include:

1. **User Authentication:** Users can create accounts and securely log in to access their diary entries.
2. **Entry Creation:** The app allows users to create new diary entries, capturing details such as date, time, title, and content.
3. **Entry Management:** Users can view, edit, and delete existing diary entries, providing flexibility in managing their journal entries.
4. **User Interface:** The graphical user interface (GUI) was designed to be intuitive and visually appealing, ensuring a seamless user experience.
5. **Database Integration:** Utilized SQLite database to store and manage diary entries, ensuring data persistence and reliability.

Conclusion

In conclusion, the development of our Python-based daily diary app marks a significant step towards efficient organization and reflection. Through this project, we've successfully combined functionality with user-friendliness, offering individuals a convenient platform to record their daily experiences and thoughts. With further enhancements and user feedback, this app holds the potential to become an indispensable tool for personal reflection and organization.

Reference

- Authors: R. J. K. Jacob, J. O. Wobbrock, M. P. Matthew Kay, J. A. Zimmerman, A. R. Leggett, H. Hsu, A. E. Bonsignore
- Basic Knowledge on Flask : https://github.com/topics/diary-application?l=python

- Flask module Guides: https://youtu.be/77u1ReKg-Wo?si=Ron2ihpp6U0UjoYn
- Diary Journal: https://realpython.com/django-diary-project-python/