PROJECT FRIENDS

V1.0

A python project

Contents

Introduction

Description

FRIENDS is a diary which will hold profiles of ones friends and memories.

The diary will run through a browser on localhost or any address the user specifies. To store and read data it will use an SQL database.

Along with a diary FRIENDS will have a notification app which will run upon the computers start-up. It will check if any of the users' friends have a birthday and display a message if it finds a positive.

Resources

To realise this project using python a set of module will be required.

- **SQLite 3**[1] for the database
- **Bottle**[2] for the server
- **Tkinter**[3] for the notification app.

[1] SQLite 3: https://www.sqlite.org/
[2] Bottle: http://bottlepy.org/
[3] Tkinter: http://www.tkdocs.com/

How to use

Icons

Table 1 Icon description

Icon	Description
✓	Confirm, save
➕	Add new
🗑	Delete
✎	Edit
✗	Cancel

Diary

Creating an article

To create an article navigate to “My diary” page and click the “Add new” button.
Enter a title for the article and write the article itself.
To finish and create the article press the “Save” button.

Editing an article

To edit an article navigate to “My diary” page, select the article you want and click the “Edit” button.
You can edit the title and text.
To save changes press the “Save” button.

Deleting an article

To delete an article navigate to “My diary” page and click the “Edit” button.
Then click the “Delete” button.
Confirm your action by clicking the “Confirm” button.

Links

You can insert links to your friends into the article by using the drop down list to select a friend and clicking the “Insert” button to insert the list at the current cursor position.

This feature is available when creating and editing an article in the diary.

Project structure

The database

The “Friends” table

Description

This table contains each and every person with their details.

Columns

Table 2 Database "friends" table structure

user_id	name	met_how	met_when	bday	gender	zodiac	desc

user_id

- User ID is of an integer datatype.
- It is auto incremented when a new person is created.
- PK[4]

name

- Name is of a text datatype.
- It is unique to each user.

met_how

- Met how of an integer datatype.
- Its value is used to reference items from the “metHow” table.

met_when

- Met when is of a text datatype.
- It stores the year or exact the user met with the person.
- <u>This may be changed in future versions.</u>

bday

- Birthday is of a text datatype.
- It allows the user to input the day and month and a rough year date. Ex.: 12.01.199x.
- <u>This will be changed in future versions.</u>

gender

- Gender is of a text datatype.
- It stores a single character resembling the persons’ gender.
- <u>This may be changed in future versions.</u>

zodiac

- The zodiac is of a text datatype.
- It is calculated automatically by db.py and server.py using the birth date.

desc

- Description is of a text datatype
- It stores non rich text data used to describe the person.

[4] **P**rimary **k**ey - uniquely identifies each record in a database table

The “metHow” table

Description

This table contains a list of places and ways the user met the person.

Columns

Table 3 Database "metHow" table structure

how_id	how

how_id

- How ID is of an integer datatype.
- It is auto incremented when a new row is created.
- PK.
- Its value is used to identify the requested row and pass its data.

how

- How is of a text datatype.
- It stores the place/way the user met the person.

The “diary” table

Description

This table contains articles for the diary.

Columns

Table 4 Database "diary" table structure

article_id	title	article	date

article_id

- Article ID is of an integer datatype.
- It is auto incremented when a new row is created.
- PK.
- Its value is used to identify the requested row and pass its data

title

- Title is of a text datatype.
- It stores the article title.

article

- Article is of a text datatype.
- It stores the article text.

date

- Date is of a text datatype.
- It stores the article creation date.

The server

Static

Description

This directory contains four subfolders with static files

Subfolders

avatar

This contains profile pictures of each person from the database (above).

css

This contains the global stylesheet along with page specific styles.

img

This contains graphics for the website theme.

js

This contains scripts which enhance the websites' functionality.

Views

Description

This directory contains HTML templates which serve as building blocks for the page.

The website is made up from three standard components: header, footer and content.
The **header** and **footer** are constant on each page while the **content** varies depending on the page purpose.

Pages

Header (header.html)

This template displays the header and navigation.

Content

table.html

This template displays a table of people from the database.

profile.html

This template displays the profile of a selected person.

profileEdit.html

This template displays the profile editing form of a selected person.

profileCreate.html

This template displays the profile creation form.

profileDelete.html

This template displays a confirmation message before deleting a user.

diary.html

This template displays a list of article entries from the database.

diaryDisplay.html

This template displays the selected article.

diaryEdit.html

This template displays the article editing form of a selected article.

diaryAdd.html

This template displays the article creation form.

diaryDelete.html

This template displays a confirmation message before deleting an article.

error.html
This template displays an error message depending on received error name.

Footer (footer.html)
This template displays the footer.

Server (server.py)

Description
Handles requests, operates with the database and returns the appropriate content.

Functions

#getZodiac
Outputs zodiac name depending on the received birthdate. The received value is in a D.M.Y format as a text datatype.

1. Returns "None" if received value is equal to "None" and skips the steps below.
2. Splits value into day and month as integers.
3. If the day and month are in range of a zodiac sign, its name is returned.

#checkInt
Outputs Boolean depending on the input value. The received value can be of any datatype.

1. The received value is parsed to an "int" in a handled exception
2. Returns "False" if an error occurs. Otherwise returns "True".

#uploadFile
Saves file to requested folder in the above directory with requested name and extension.

1. Pulls file request from specified input
2. If the file is blank (equal to "None") skips all steps below. <u>This will be enhanced with an error handler.</u>
3. Splits filename into name and extension.
4. If the extension doesn't match the requested file type skips all steps below. <u>This will be enhanced with an error handler.</u>
5. Saves file.

#dataToDict
Converts tuple rows from an SQL database to a dictionary.

1. Extracts row ID from tuple.
2. Save to dictionary with the ID as the key and the rest as the value.
3. Returns dictionary.

#tagToLink
Converts tags in {tag=#} format to a link. <u>More tag types will be added.</u>

1. Splits the first half of the string with "{tag" to a list.
2. Splits values in the list with the "}" to a new list.
   The received value is the tag id.
3. For each value in the new list replace tag with link.

Routes

#index
Handles the GET request for the root/homepage and outputs the appropriate content.

#profile
Handles the GET request for the profile page and outputs the appropriate content.

#editProfile
Handles the GET request for the edit profile page and outputs the appropriate content.

#saveProfile
Handles the POST request from the edit profile page and redirects to the profile page.

#createProfile
Handles the GET request for the create profile page and outputs the appropriate content.

#create
Handles the POST request from the create profile page and redirects to the profile page.

#deleteProfile
Handles the GET request and displays the confirmation page. Checks if confirmed and deletes selected profile.

#diary
Handles the GET request for the diary page and outputs the appropriate content.

#articleDisplay
Handles the GET request for the article page and outputs the appropriate content.

#articleEdit
Handles the GET request for the article edit page and outputs the appropriate content.

#articleSave
Handles the POST request from the article edit page and redirects to the article page.

#articleCreate
Handles the GET request for the article create page and displays the appropriate content.

#add
Handles the POST request from the article create page and redirects to the article page.

#articleDelete
Handles the GET request and displays the confirmation page. Checks if confirmed and deletes selected article.

#static
Handles the GET request for static files from the above directory and returns the requested file.

Notification (notification.pyw)

Description
Checks the database for current events and notifies the user with a pop-up window.

Classes

#App

Displays a notification in a window from received arguments.

1. Draws a label with the notification name.

Bottle (bottle.py)

Description

This is an external python module which is used to run the server.
More at: http://bottlepy.org