chronophore Documentation

Release 0.6.0

Amin Mesbah

Mar 28, 2017

Contents

CHAPTER 1

What is Chronophore?

Chronophore is a time-tracking program. It keeps track of users' hours as they sign in and out.

This project was started to help keep track of students and tutors signing in and out at a tutoring center in a community college.

CHAPTER 2

Installation

Chronophore can be installed with pip:

```
$ pip install chronophore
```

CHAPTER 3

Usage

```
usage: chronophore [-h] [--testdb] [-v] [--debug] [-V] [--tk]

Desktop app for tracking sign-ins and sign-outs in a tutoring center.

optional arguments:
  -h, --help     show this help message and exit
  --testdb       create and use a database with test users
  -v, --verbose  print a detailed log
  --debug        print debug log
  -V, --version  print version info and exit
  --tk           use old tk interface
```

CHAPTER 4

Documentation

Chronophore's Documentation is available at https://chronophore.readthedocs.io.

Hosting is graciously provided by the good people at Read The Docs. Many thanks!

Chronophore's Database

Working with the Database

Chronophore's database lives in a single SQLite file (*chronophore.sqlite* by default). It can be opened, browsed, and edited with a number of different programs.

DB Browser for SQLite, aka *SQLiteBrowser*, is one such program. It offers an intuitive graphical interface that is somewhat similar to Microsoft Excel or Access. It can be installed on Windows, Mac, or Linux.

Browse the Database

1. Run *SQLiteBrowser*.
2. Click the "Open Database" button.
3. Find and select Chronophore's database, then click open.
4. Click the "Browse Data" tab.
5. Use the drop-down menu to select a table to browse.
6. Data can be sorted and filtered by clicking on the column headers.

Add New Users

1. Switch to the "Browse Data" tab.
2. In the "Table" drop-down menu, select the "users" table.

3. Click "New Record". A new row will be created.
4. Fill in at least the required fields: *user_id*, *first_name*, *last_name*, *is_student*, and *is_tutor*.
5. Repeat to add a new record for each student.
6. Once all new records have been added, click the "Write Changes" button. This will commit your changes to the database.

Important: Empty cells should say "*NULL*". If an empty cell doesn't have the word "*NULL*" in it:

1. Double click the cell.
2. Click the "Clear" or "Set As NULL" button.
3. Click "Write Changes" button.

Delete Users

This should almost never be necessary. If a user is no longer a part of the program, simply fill in the *date_left* cell in their record. If a user truly needs to be removed from the database, their Timesheet entries must be removed first:

1. Use the drop-down menu in the "Browse Data" tab to switch to the "timesheet" table.
2. Filter the *user_id* column by the user's user id.
3. Select each row to be removed, then click the "Delete Record" button.
4. Use the drop-down menu in the "Browse Data" tab to switch to the "users" table.
5. Select the row for the user you want to delete, then click the "Delete Record" button.
6. If you are sure the correct records have been deleted, click the "Write Changes" button.

Warning: Once changes have been written, they are permanent. Unwritten changes can be reverted with the "Revert Changes" button.

Export Data

SQLiteBrowser can export tables from the database as CSV files. The CSV files can then be imported into, for example, spreadsheet software.

1. Click "File" -> "Export" -> "Table(s) as CSV File".
2. Select which tables to export, then click the "Ok" button.

View the Database Structure

Information about the schema and structure of the database is available in the "Database Structure" tab.

The Schema

Chronophore's database has a relatively simple schema with only two tables.

Timesheet

This is the table that Chronophore writes to. It stores an entry for every time someone signs in, then updates it when they sign out. It should generally not be edited by hand.

It contains the following fields:

Field Name	Significance
uuid	A unique ID for each entry (*Primary Key*).
date	The date the user signed in.
forgot_sign_out	*1* if the user never signed out, *0* otherwise.
time_in	Sign in time.
time_out	Sign out time.
user_id	The user's unique ID (*Foreign Key*).
user_type	Whether the user signed in as a *student* or a *tutor*.

Users

This table stores information about the registered users. A user will not be able to sign in unless they have a record in this table. Chronophore itself doesn't ever write to this database. It must be edited with another application such as DB Browser for SQLite.

It contains the following fields:

Field Name	Significance
user_id	The user's unique ID (*Primary Key*).
date_joined	The date on which the user was registered.
date_left	The date on which the user was no longer registered.
education_plan	*1* if the user has submitted an education plan, *0* otherwise.
school_email	The user's school email.
personal_email	The user's personal email.
first_name	The user's first name.
last_name	The user's last name.
major	The user's declared major.
is_student	*1* if the user is a student, *0* otherwise.
is_tutor	*1* if the user is a tutor, *0* otherwise.

API Reference

Modules

chronophore

chronophore.chronophore.**get_args**()
Parse command line arguments.

chronophore.chronophore.**set_up_logging**(*log_file*, *console_log_level*)
Configure logging settings and return a logger object.

chronophore.chronophore.**main**()
Run Chronophore based on the command line arguments.

config

chronophore.config.**_load_config**(*config_file*)
Load settings from config file and return them as a dict. If the config file is not found, or if it is invalid, create and use a default config file.

Parameters **config_file** – *pathlib.Path* object. Path to config file.

Returns Dictionary of config options.

chronophore.config.**_load_options**(*parser*)
Load config options from parser and return them as a dict.

Parameters **parser** – *ConfigParser* object with the values loaded.

Returns Dictionary of config options.

chronophore.config.**_use_default**(*config_file*)
Write default values to a config file. If another config file already exists, back it up before replacing it with the new file.

Parameters **config_file** – *pathlib.Path* object. Path to config file.

Returns *ConfigParser* object with the values loaded.

controller

exception chronophore.controller.**AmbiguousUserType**(*message*)
This exception is raised when a user with multiple user types tries to sign in.

exception chronophore.controller.**UnregisteredUser**(*message*)
This exception is raised when a user id doesn't match any user in the database.

class chronophore.controller.**Status**(*valid*, *in_or_out*, *user_name*, *user_type*, *entry*)
Status is a namedtuple used by the *sign()* function to return relevant information to the gui about a sign-in or sign-out attempt.

valid
Whether or not the user was valid.

in_or_out
Whether the user was signing in or out.

user_name
The name of the user.

user_type
Whether the user was a student or a tutor.

entry
The *chronophore.models.Entry* object signed into or out of.

chronophore.controller.**flag_forgotten_entries**(*session*, *today=None*)
Flag any entries from previous days where users forgot to sign out.

Parameters

- **session** – SQLAlchemy session through which to access the database.
- **today** – (optional) The current date as a *datetime.date* object. Used for testing.

chronophore.controller.**signed_in_users**(*session=None*, *today=None*, *full_name=True*)
Return list of names of currently signed in users.

Parameters

- **session** – SQLAlchemy session through which to access the database.
- **today** – (optional) The current date as a *datetime.date* object. Used for testing.
- **full_name** – (optional) Whether to display full user names, or just first names.

Returns List of currently signed in users.

chronophore.controller.**get_user_name**(*user*, *full_name=True*)
Return the user's name as a string.

Parameters

- **user** – *models.User* object. The user to get the name of.
- **full_name** – (optional) Whether to return full user name, or just first name.

Returns The user's name.

chronophore.controller.**sign_in**(*user*, *user_type=None*, *date=None*, *time_in=None*)
Add a new entry to the timesheet.

Parameters

- **user** – *models.User* object. The user to sign in.
- **user_type** – (optional) Specify whether user is signing in as a *'student'* or *'tutor'*.
- **date** – (optional) *datetime.date* object. Specify the entry date.
- **time_in** – (optional) *datetime.time* object. Specify the sign in time.

Returns The new entry.

chronophore.controller.**sign_out**(*entry*, *time_out=None*, *forgot=False*)
Sign out of an existing entry in the timesheet. If the user forgot to sign out, flag the entry.

Parameters

- **entry** – *models.Entry* object. The entry to sign out.
- **time_out** – (optional) *datetime.time* object. Specify the sign out time.
- **forgot** – (optional) If true, user forgot to sign out. Entry will be flagged as forgotten.

Returns The signed out entry.

chronophore.controller.**undo_sign_in**(*entry*, *session=None*)
Delete a signed in entry.

Parameters

- **entry** – *models.Entry* object. The entry to delete.
- **session** – (optional) SQLAlchemy session through which to access the database.

chronophore.controller.**undo_sign_out**(*entry*, *session=None*)
Sign in a signed out entry.

Parameters

- **entry** – *models.Entry* object. The entry to sign back in.
- **session** – (optional) SQLAlchemy session through which to access the database.

chronophore.controller.**sign**(*user_id*, *user_type=None*, *today=None*, *session=None*)
Check user id for validity, then sign user in if they are signed out, or out if they are signed in.

Parameters

- **user_id** – The ID of the user to sign in or out.
- **user_type** – (optional) Specify whether user is signing in as a *'student'* or *'tutor'*.
- **today** – (optional) The current date as a *datetime.date* object. Used for testing.
- **session** – (optional) SQLAlchemy session through which to access the database.

Returns *Status* named tuple object. Information about the sign attempt.

models

class chronophore.models.**User**(***kwargs*)

Schema for the 'users' table.

user_id
The user's unique ID (*Primary Key*).

date_joined
The date on which the user was registered.

date_left
The date on which the user was no longer registered.

education_plan
1 if the user has submitted an education plan, *0* otherwise.

school_email
The user's school email.

personal_email
The user's personal email.

first_name
The user's first name.

last_name
The user's last name.

major
The user's declared major.

is_student
1 if the user is a student, *0* otherwise.

is_tutor
1 if the user is a tutor, *0* otherwise.

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class chronophore.models.**Entry**(***kwargs*)

Schema for the 'timesheet' table.

uuid
A unique ID for each entry (*Primary Key*).

date
The date the user signed in.

forgot_sign_out
1 if the user never signed out, *0* otherwise.

time_in
Sign in time.

time_out
Sign out time.

user_id
The user's unique ID (*Foreign Key*).

user_type
Whether the user signed in as a *student* or a *tutor*.

__init__(***kwargs*)
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

chronophore.models.**set_sqlite_pragma**(*dbapi_connection*, *connection_record*)
Upon every db connection, issue a command to ensure foreign key constraints are enforced.

This is a sqlite-specific issue: http://stackoverflow.com/questions/2614984/sqlite-sqlalchemy-how-to-enforce-foreign-keys http://docs.sqlalchemy.org/en/latest/dialects/sqlite.html#foreign-key-support

chronophore.models.**add_test_users**(*session*)
Add two hobbits and a wizard to the users table for testing purposes. These are not necessarily the same test users as in the unit tests.

This function is idempotent.

qtview

tkview

class chronophore.tkview.**TkChronophoreUI**
Simple Tkinter GUI for chronophore :

- Entry for user id input
- Button to sign in or out
- List of currently signed in users

_set_signed_in()
Populate the signed_in list with the names of currently signed in users.

_show_feedback_label(*message*, *seconds=None*)
Display a message in lbl_feedback, which then times out after some number of seconds. Use after() to schedule a callback to hide the feedback message. This works better than using threads, which can cause problems in Tk.

_sign_button_press(**args*)
Validate input from ent_id, then sign in to the Timesheet.

__weakref__
list of weak references to the object (if defined)

class chronophore.tkview.**TkUserTypeSelectionDialog**(*parent*, *title=None*, *entry_to_clear=None*)
A modal dialog presenting the user with options for what kind of user to sign in as.

body(*master*)
Create dialog body. Return widget that should have initial focus.

Inherited from tkinter.simpledialog.Dialog

buttonbox()
Inherited from tkinter.simpledialog.Dialog

ok(*event=None*)
This method is identical to tkinter.simpledialog.Dialog.ok(), but with 'self.withdraw()' commented out.

apply()
Inherited from tkinter.simpledialog.Dialog

Index

Symbols

A

B

D

E

F

G

I

L

M

O

P

S

T

U

V