ClockIN

Developer’s documentation

https://github.com/LukeHartell/ClockIN.git
07.05.2024

ClockIN

File Name: ClockIN.py

Overview

This script serves as the entry point for the "ClockIN" application, which is currently in its Alpha version. It initializes necessary files and starts the main application loop.

Dependencies

- **MainApplication**: A class from MainApplication.py that handles the main GUI and functionalities of the application.
- **utilities**: A module providing utility functions like initialize_files.

Constants

- **projectname**: Name of the project, set to "ClockIN".
- **version**: Current version of the application, set to "(Alpha)".

Functions

- **on_close()** handles the closure of the application by safely terminating the main application instance. This function is connected to the window manager's delete window protocol to ensure graceful shutdown.

Main Execution Block

If the script is run as the main program, it performs the following actions:
1. File Initialization:
    - Calls **initialize_files** from the utilities module to prepare necessary files and directories for the application.

2. Main Application Initialization:
    - Instantiates **MainApplication** from MainApplication.py.
    - Configures the application to call **on_close** when the application window's close button is clicked.
    - Enters the main event loop of the application to keep it running until manually closed.

Comments and Disabled Code

- There are commented sections that handle specific scenarios when the application is compiled into a standalone executable using PyInstaller. These include:
- Importing pyi_splash to manage splash screens for the frozen application.
- Closing the splash screen before launching the main application.

Utilities

File Name: utilities.py

Overview:

This module handles file and directory initialization and management for the "ClockIN" application. It also sets up default settings and prepares data files for the application's operation.

Dependencies

- **csv**: For handling CSV file operations.
- **os**: For interacting with the operating system to manage file paths and directories.
- **datetime**: For obtaining current date and time.
- **json**: For handling JSON data storage and retrieval.
- **config**: For fetching default configuration settings from config.py.
- **tkinter.messagebox**: For displaying error messages in GUI.
- **decimal.Decimal**, **decimal.getcontext**: For setting precision in Decimal operations.

Constants

- **sheet_headers**: Headers for the timesheet CSV files.
- **appdata_path**: Path to the AppData folder where the application data is stored.
- **projectname**: Name of the project, set to "ClockIN".
- **version**: Current version of the application, set to "Alpha".
- **current_year**: Current year as a string, used in file paths.
- **usersettings_path**: Path to the user settings JSON file.
- **timesheet_path**: Path to the current year's timesheet CSV file.
- **saldi_path**: Path to the current year's saldo JSON file.

Functions

- **initialize_files()**

Description

Initializes necessary files and directories for the application.
- Calls create_project_directories to set up project folder structure.
- Calls initialize_default_files to create and set up default files with initial data.
- Handles exceptions by displaying an error message through a GUI popup.

create_project_directories()
Description: Creates the base directory for the project and a subdirectory for the current year if they do not exist.

initialize_default_files()
Description: Creates default user settings, timesheet, and saldo files if they do not exist.
- Uses JSON to create the user settings file with default settings obtained from config.get_default_settings().
- Creates a timesheet CSV file with predefined headers.
- Initializes saldo JSON files with starting values for leave and flex time balances.

Config

File Name: config.py

Overview

This module defines configuration settings and default user preferences for the "ClockIN" application. It provides utility functions to fetch predefined settings which include Danish names for days and months, user details, and default work hours.

Constants

- **danish_months**: Tuple containing the names of months in Danish.
- **danish_days**: Tuple containing abbreviations for days of the week in Danish.

Functions

- **get_default_settings()** returns a dictionary containing default settings for a user.

Main application

File Name: MainApplication.py

Overview

This module serves as the main application class for the "ClockIN" software, integrating various components and functionalities. It manages the user interface, settings, and state transitions, and handles various calculations related to work hours and user balances.

Dependencies

- **tkinter** for GUI elements.
- **os**, sys for system operations and path management.
- **json** for data storage and retrieval.
- **csv** for timesheet management.
- **datetime** for handling date and time operations.
- **CalendarPage**, **SaldiPage**, **SettingsPage**, **ReportPage**, **NewDayOffPage** for different GUI pages.

Classes Description

MainApplication(tk.Tk) responsible for initializing the GUI, loading settings, and managing application state.

Methods

- **__init__(self, *args, **kwargs)**
  Initializes the application, sets window properties, prepares the environment (development or packaged), and initializes file structures and settings. Calls various setup methods to prepare the application for use.

- **setup_frames_and_menu(self)**
  Configures the main container and individual frames for each page of the application. Sets up the main navigation menu allowing users to switch between different parts of the application.

- **show_frame(self, context)**
  Raises the specified frame to the top of the window stack for display. Handles special conditions like refreshing certain pages upon being displayed.

- **ensure_default_settings(self)**
  Checks for the existence of user settings files and writes default settings if they are missing.

- **write_default_settings_if_needed(self, settings_path, default_settings)**
  Writes default settings to a specified file path if they do not exist.

- **load_user_settings(self)**
  Loads user settings from a JSON file, handling file not found errors by reverting to default settings.

- **load_user_saldi(self)**
  Retrieves saved saldi from a file, handling errors and missing files by providing default values.

- **calculate_saldi(self)**
  Orchestrates the calculation of all types of saldi (balances), updates the application state, and refreshes relevant UI components.

- **beregn_ugetimer(self)**
  Calculates the total work hours for a week based on user-configured work hours.

- **calculate_flex_saldo(self)**
  Computes the balance for flex time by accounting for initial balances, consumption, and any configured biases.

- **calculate_ferie_saldo(self)**
  Calculates the vacation balance based on employment duration, predefined accrual rates, and consumption.

- **calculate_omsorgsdage_saldo(self)**
  Computes the balance for care days based on the number of eligible children and consumption.

- **calculate_seniordage_saldo(self)**
  Determines the balance of senior days based on the user's age and other relevant factors.

- **save_saldi(self, saldi)**
  Saves the current state of all saldi to a file.

- **decimal_to_hours_minutes(self, decimal_hours)**
  Converts hours in decimal format to a string format (HH:MM).

- **hours_minutes_to_decimal(self, time_str)**
  Converts time in HH:MM format to decimal hours.

Kalender

File Name: CalendarPage.py

Overview

This module defines the CalendarPage class which is part of the GUI of the "ClockIN" application. It manages calendar interactions, time entries for clock-in and clock-out, and flex balance calculations.

Dependencies

- **tkinter** for GUI components.
- **csv** for handling CSV file operations.
- **datetime** for managing dates and times.
- **tkcalendar** for calendar widget functionalities.
- **tktimepicker** for providing analog time pickers.
- **json** for JSON file operations.
- **config** module for accessing configuration settings.
- **NewDayOffPage** for navigating to the day off registration page.
- **utilities** for accessing shared paths and headers.

Class Description

CalendarPage(tk.Frame) extends tkinter's Frame class to handle the calendar page, allowing users to select dates, enter times, and view flex balances.

Methods

- **__init__(self, parent, controller)**
  Initializes the page, sets up widgets, and loads initial data.

- **setup_widgets(self)**
  Sets up all GUI components including calendar, time pickers, and buttons.

- **on_date_select(self, event)**
  Called when a date is selected on the calendar. It updates the time entries based on the selected date.

- **get_date_and_day(self)**
  Returns the selected date and the day of the week.

- **update_time_entries(self, date, settings, selected_day)**
  Updates the time picker values based on the selected date and user settings.

- **set_time_pickers(self, clock_in, clock_out)**
  Sets the time picker widgets to specific times.

- **validate_and_parse_time(self, time_str)**
  Validates and parses a time string into hours and minutes.

- **read_all_data(self)**
  Reads all time entry data from the timesheet CSV file.

- **load_balances(self)**
  Loads balance data from a JSON file.

- **save_balances(self, balances)**
  Saves updated balance data to a JSON file.

- **write_all_data(self, data)**
  Writes all time entry data back to the CSV file, recalculating flex balances as necessary.

- **calculate_weekly_flex(self)**
  Calculates the total flex balance for the current week.

- **get_normtid_decimal(self, user_settings)**
  Parses and returns the standard daily working time from user settings.

- **get_daily_norm(self, selected_day, use_avg_hours, user_settings, current_normtid_decimal)**
  Calculates the norm time for a specific day based on user settings.

- **update_balances_and_refresh(self, flex_saldo, user_settings, balances)**
  Updates flex balances and refreshes the UI to display the new balances.

- **refresh_calendar(self)**
  Refreshes the calendar display, updating event tags and applying color coding based on time entry statuses.

- **format_time(self, time_tuple)**
  Formats a time tuple into a HH:MM string.

- **save_times(self)**
  Saves the entered times to the timesheet and recalculates balances.

- **clear_times(self)**
  Clears the time entries for the selected date and updates the display.

- **set_default_times(self, selected_day, settings)**
  Sets the time pickers to default or specified times based on settings.

- **set_current_time(self, picker)**
  Sets the given time picker to the current time and saves the times.

- **reset_flex_saldo(self)**
  Resets the flex balance to zero or a predetermined base value.

- **load_flexhours_to_widget(self)**
  Loads and displays the total and weekly flex hours in the UI.

Button Flow

"Registrér fri"

Calls **self.controller.show_frame(NewDayOffPage):**
Triggers the display of the **NewDayOffPage** within the application's frame structure.

"Tjek ind nu"/"Tjek ud nu"

Calls **self.set_current_time(self.clock_in_picker)** or **self.set_current_time(self.clock_out_picker):**
set_current_time(picker): Sets the given time picker to the current time.
Retrieves current system time.
Sets the hour and minute on the specified picker.
Calls **self.save_times()**:
save_times(): Processes and stores the current times into the timesheet.
Calls **self.get_date_and_day():** Retrieves the selected date and day name.
Calls **self.format_time(self.clock_in_picker.time())** and **self.format_time(self.clock_out_picker.time()):** Formats the times from the pickers.
Calls **self.read_all_data():** Reads the existing timesheet data.
Modifies the data with new times.
Calls **self.write_all_data(all_data):** Writes the updated data back to the CSV file and updates flex balances.
Iterates through data, recalculating flex based on the norm and actual hours worked.
Updates the CSV file.
Calls **self.refresh_calendar():** Updates the calendar display to reflect the new data.
Calls **self.refresh_calendar()** after saving the times.

"Gem"

Calls **self.save_times():**
Similar to the flow described in "Tjek ind nu"/"Tjek ud nu", it processes and saves the times entered, updates the timesheet, recalculates balances, and refreshes the calendar.

"Ryd"

Calls **self.clear_times():**
clear_times(): Clears the time entries for the selected date.
Calls **self.get_date_and_day():** Retrieves the selected date and day name.
Checks if the date exists in the times data; if yes, deletes it.
Calls **self.write_all_data(self.times_data):** Recalculates the data after clearing the specific date's entries.
Calls **self.refresh_calendar():** Updates the calendar display.
Calls **self.set_default_times(selected_day, settings):** Resets the time pickers.
Retrieves settings from the controller.
Sets the time pickers to either default times or specified working hours from settings.
Calls **self.load_flexhours_to_widget():** Updates the flex hours display.

Saldi

File Name: SaldiPage.py

Overview

This module defines the SaldiPage class, a component of the GUI for the "ClockIN" application that displays various types of saldi (balances) such as flex hours, care days, and senior days. The page automatically refreshes upon loading to show the latest balance data.

Dependencies

- **tkinter** for GUI components.
- **json** for handling JSON file operations.
- **utilities** for accessing the saldi_path.

Class Description

SaldiPage(tk.Frame) extends tkinter's Frame class to handle the display of balance data on the Saldi page.

Methods

- **__init__(self, parent, controller):** Initializes the SaldiPage, sets up the layout, and initiates saldo calculations and display.

- **refresh(self):** Updates the saldo display. Clears existing widgets in the table frame to prevent duplication and re-creates table headers for saldo type and available amount.

Indstillinger

File Name: SettingsPage.py

Overview

This module defines the SettingsPage class within the "ClockIN" application. It handles user-specific settings adjustments, including personal details, work hours configuration, and handling of specific biases like flex time and vacation days. The page provides interactive forms for entering and saving these settings, and it recalculates saldi based on changes to critical settings.

Dependencies

- **tkinter** for GUI components.
- **json** for handling JSON file operations.
- **utilities** for accessing the usersettings_path.
- **ReportPage**, **SaldiPage** for interacting with other components of the application.

Class Description

SettingsPage(tk.Frame) extends tkinter's Frame class to provide a settings interface for users to update their preferences and work configurations.

Methods

- **__init__(self, parent, controller):**
  Initializes the settings page, loads user settings, and sets up widgets.

- **load_widgets(self):**
  Sets up the layout for user input fields including personal details, work hours, and bias adjustments.

- **save_and_recalculate_saldi(self):**
  Saves the settings and recalculates saldi if critical settings that affect calculations have changed.

- **add_child(self):**
  Adds a child's birth year to the settings for calculating care days.

- **remove_selected_child(self):**
  Removes a selected child from the settings.

- **save_config(self):**
  Updates and saves the user configuration to a JSON file. Checks for changes in settings that might affect calculations and triggers a recalculation if necessary.

- **load_settings_to_widgets(self):**
  Populates the input fields with existing user settings.

- **load_workhours_to_widget(self):**
  Updates the display of weekly work hours based on settings.

Button Flow

"Gem indstillinger"

Calls **save_and_recalculate_saldi** to save the settings and potentially recalculate saldi based on changes. This method interacts with:

save_config: Saves the updated settings to the usersettings_path. This method checks for changes in work hours and biases, and returns a boolean indicating if a recalculation of saldi is necessary.

calculate_saldi and refresh: If save_config returns True, indicating that critical settings affecting calculations have changed, it calls the controller's calculate_saldi method followed by a refresh on the SaldiPage.

"Tilføj barn"

Opens a dialog to enter a child's birth year and adds it to a list, affecting the calculation of care days.

"Fjern valgte"

Removes the selected child from the list, affecting the settings related to care days.