LearnoDoro

The Pomodoro Timer App for Students

Product Vision

For the everyday student grinding through lectures, assignments, and group projects, who struggles with time management and seeks tangible proof of their dedication, LearnoDoro is the solution. It offers a customized study regimen enriched by insightful feedback. While numerous other timer apps exist, most offer just basic timer functionality and miss out on the analytics on how your time is spent. LearnoDoro ensures that students (and whoever finds good use of it) gain a comprehensive overview of precisely how their study hours are allocated.

What even is Pomodoro?

Definiton according to ChatGPT: The Pomodoro Technique is a time management method that involves working in focused 25-minute intervals (called "Pomodoros") followed by short breaks to improve productivity and concentration.

The Pomodoro Process

1. Choose task to do
2. Set Pomodoro timer (usually 25 min)
3. Do the task until the timer reaches 0
4. take a short break (ca. 5–10 min)
5. Return to Step 2 and repeat until four pomodoros are completed
6. After four pomodoros, take a long break (ca. 20-30 min) instead of a short break
7. Return to step 2

Core Features

1. Pomodoro Timer
    - Users select a "Course" at the start of each Pomodoro. Optionally, they may choose a specific "Task" within that course
    - Start/Stop the countdown timer
    - Settings to customize study time duration, short break length, long break length and the number of Pomodoros required before a long break
2. Courses
    - Create a "Course"
    - Create a "Task" within a "Course"
    - Time spent working on a specific "Course" is logged
3. Tasks
    - Create a "Task" in a "Course"
    - Time spent working on a specific "Task" is logged too
4. Logs and Analytics
    - Logs each "StudySession" spent on each "Course" and "Task"
    - Comparative Analysis across "Courses" and "Tasks"
    - View study time over selected time-periods
    - Visual Representations like bar graphs, pie charts and line charts for easy understanding
    - Export "StudySessions" to .csv for further analysis

Installation and Setup

LearnoDoro offers two convenient methods for installation and setup:

Using `main.py`:

- Edit config.py in the /Software directory to set your database save location.
- In DEVELOPMENT mode (DEVELOPMENT = True), database and settings are stored in /Software/data.
- In production mode (DEVELOPMENT = False), database and settings are stored in the user's local appdata.

Using `main.exe`:

- No code modifications required.
- All data is automatically saved in the user's local appdata.

Once setup is complete, LearnoDoro is ready for use. Happy studying!

Usage Instructions

Simplicity is key in LearnoDoro, ensuring you spend more time studying and less figuring out the app.

Here's your quick guide to get started:

1. Launch LearnoDoro
2. Add a Course: First-timers, add a new course. Or, select an existing course from the dropdown next to `Add Course`.
3. Add a Task (Optional): You can add a new task to your course or choose an existing one from the dropdown next to `Add Task`.
4. Check Settings (Optional): Go to `Settings` to adjust the timer to your preference. Default settings follow the traditional Pomodoro technique.
5. Start Studying: Set your course, task (optional), and timer settings, then hit `Start` to begin your study session.
6. Follow the Timer: Study and take breaks as indicated by the timer.
7. Stop and Log: Once done, click `Stop`. Your session will be logged and viewable in `Analytics`.

Using Analytics:

1. Access Analytics: Go to `Analytics` for a session overview. A new window will open.
2. Select Courses/Tasks: Choose the courses and tasks you want insights on.
3. Set Date Range: Enter the start and end dates as per the format shown in the window.
4. View Charts (Optional): To see a chart, click `Show Graph` and choose the chart type.
5. Display Analytics: Click `Display Analytics` to view your study data.
6. Export Logs (Optional): To download study logs for the selected period, click `Export Study Logs to CSV`.

Dependencies

Dependency	Reason for inclusion
`ttkthemes`	App GUI
`matplotlib`	Visualize the data (pie,bar,line charts)
`pytest`	Testing implementations in this project using automatic test cases.

Software Documentation

LearnoDoroApp

`LearnoDoroApp` is a Python application built using the Tkinter library for creating a GUI-based study timer application. The application incorporates Pomodoro technique elements, allowing users to manage study sessions with courses and tasks, and track their time efficiently.

Features

- Add, select, and delete courses and tasks.
- Start and stop a study timer with custom study and break durations.
- Display and log study sessions with analytics.
- Export study session data to CSV.

Class Methods

`__init__(self)`

Initializes the application window with specified themes and styles. It also checks and initializes the database if necessary, loads settings, and sets up the main frame and its widgets.

`set_course_task_widgets_state(self, state)`

Enables or disables the widgets related to course and task management based on the given state.

`refresh_app_state(self)`

Refreshes the application's state, particularly the timer display based on the current settings.

`start_timer(self)`

Starts the study timer and updates the application state to reflect this. It also disables certain widgets during the timer's operation.

`stop_timer(self)`

Stops the study timer, resets it, and updates the application state accordingly. It logs the study session and re-enables disabled widgets.

`reset_timer(self)`

Resets the timer display to the initial study time.

`update_timer(self)`

Updates the timer display every second and manages the transitions between study, short break, and long break phases.

`update_study_time(self)`

Updates the total study time in the database for the current course and task.

`add_course(self)`

Adds a new course based on user input and updates the course-related widgets.

`delete_course(self)`

Deletes the selected course and updates the application state and widgets accordingly.

`update_course_combobox(self)`

Updates the list of courses in the course combobox.

`on_course_selected(self, event)`

Handles the event when a course is selected from the combobox. Updates tasks and labels accordingly.

`add_task(self)`

Adds a new task to the selected course based on user input.

`delete_task(self)`

Deletes the selected task from the current course.

`update_task_combobox(self)`

Updates the list of tasks in the task combobox.

`on_task_selected(self, event)`

Handles the event when a task is selected from the combobox.

`open_settings(self)`

Opens a new window for adjusting application settings.

`save_changes(self)`

Saves changes made in the settings window to the application settings.

`log_study_session(self)`

Logs the completed study session into the analytics system.

`open_analytics_window(self)`

Opens a new window for displaying study analytics.

`on_start_entry_click(self, event)`

Handles focus events on the start date entry field in the analytics window.

`on_start_focusout(self, event)`

Handles focus out events on the start date entry field in the analytics window.

`on_end_entry_click(self, event)`

Handles focus events on the end date entry field in the analytics window.

`on_end_focusout(self, event)`

Handles focus out events on the end date entry field in the analytics window.

`update_analytics_task_listbox(self, event)`

Updates the task list in the analytics window based on the selected courses.

`validate_courses_and_dates(self)`

Validates the selected courses and dates for analytics purposes.

`display_analytics(self)`

Displays the analytics based on the selected parameters.

`export_to_csv(self)`

Exports the study session data to a CSV file based on the selected courses and dates.

init_db

`init_db_` is a function that sets up and initializes the database for the LearnoDoro application using SQLite3. It ensures that the necessary tables are created and properly structured to store courses, tasks, and study sessions.

Function Description

The function connects to a SQLite database using the path defined in `db_path` from the `config` module. It then proceeds to execute a series of SQL commands to set up the database schema.

Database Schema

Courses Table

- `courseID`: TEXT, Primary Key.
- `study_time`: INTEGER, a column added to store the cumulative study time for each course.

Tasks Table

- `taskID`: TEXT, part of the Composite Primary Key.
- `courseID`: TEXT, part of the Composite Primary Key and a Foreign Key referencing `courseID` in the Courses table.
- `study_time`: INTEGER, a column added to store the cumulative study time for each task.

StudySessions Table

- `sessionID`: INTEGER, Primary Key.
- `courseID`: TEXT, Foreign Key referencing `courseID` in the Courses table. Not NULL.
- `taskID`: TEXT, Foreign Key referencing a combination of `taskID` and `courseID` in the Tasks table.
- `start_time`: TIMESTAMP, marks the start of the study session. Not NULL.
- `end_time`: TIMESTAMP, marks the end of the study session. Not NULL.
- `study_duration`: INTEGER, the duration of the study session. Renamed for clarity.

Operations

- The function first checks if the `Courses` and `Tasks` tables exist and creates them if they don't.
- Next, it alters the `Courses` and `Tasks` tables to add a `study_time` column if it doesn't already exist.
- Finally, it creates the `StudySessions` table to log individual study sessions.

Notes

- The function uses exception handling to manage database connections and queries.
- It's important that this function is called at the start of the application to ensure the database is correctly set up.

Course

The `Course` class in the LearnoDoro application is designed to represent a course entity and manage its associated tasks. It provides functionalities to add, save, load, and delete courses and their tasks from a SQLite database.

Class Attributes

`courseID`

- A string representing the unique identifier for the course.

`tasks`

- A list of `Task` objects associated with the course, loaded from the database.

Class Methods

`__init__(self, courseID)`

- Initializes a new `Course` instance with the given `courseID` and loads its associated tasks.

`get_courseID(self)`

- Returns the `courseID` of the course.

`add_task(self, task)`

- Adds a `Task` object to the course's task list.

`save_to_db(self)`

- Saves the course to the database. It inserts a new record into the `Courses` table with the course's `courseID`.

`load_tasks_from_db(self)`

- Loads tasks associated with the course from the database and updates the `tasks` list. It queries the `Tasks` table for tasks that have the same `courseID`.

`@classmethod`

`load_all_courses(cls)`

- A class method that loads all courses from the database. It returns a list of `Course` objects, each initialized with a `courseID` from the database.

`@staticmethod`

`delete_from_db(courseID)`

- A static method that deletes a course and its associated tasks and study sessions from the database. It removes records from `Courses`, `Tasks`, and `StudySessions` tables where the `courseID` matches.

Database Interaction

- The class interacts with a SQLite database defined by `db_path` in the `config` module.
- Database operations include inserting new courses, deleting courses, and querying tasks and courses.

Usage

- This class is primarily used for managing course data in the LearnoDoro application, including creating new courses, managing tasks within courses, and deleting courses when necessary.

Task

The `Task` class in the LearnoDoro application is used to represent and manage individual tasks associated with a course. It provides methods to handle task-related operations in a SQLite database.

Class Attributes

`taskID`

- A string representing the unique identifier for the task.

`courseID`

- A string representing the identifier of the course to which the task belongs.

Class Methods

`__init__(self, taskID, courseID)`

- Initializes a new `Task` instance with the given `taskID` and `courseID`.

`get_taskID(self)`

- Returns the `taskID` of the task.

`get_courseID(self)`

- Returns the `courseID` of the course associated with the task.

`save_to_db(self)`

- Saves the task to the database. It attempts to insert a new record into the `Tasks` table with the task's `taskID` and `courseID`.
- If a task with the same name already exists for the selected course, an `sqlite3.IntegrityError` is caught and a message is printed.

`@staticmethod`

`load_tasks_for_course(courseID)`

- A static method that loads tasks for a specific course from the database. It queries the `Tasks` table for tasks that have the specified `courseID`.
- Returns a list of `Task` objects, each initialized with `taskID` and `courseID` from the database.

`delete_from_db(taskID, courseID)`

- A static method that deletes a task and its associated study sessions from the database. It removes records from the `Tasks` and `StudySessions` tables where both `taskID` and `courseID` match.
- Before deleting the task, it retrieves and adjusts the `study_time` in the `Courses` table, ensuring that the cumulative study time for the course remains accurate.

Database Interaction

- The class interacts with a SQLite database defined by `db_path` in the `config` module.
- Database operations include inserting new tasks, querying tasks, and deleting tasks.

Usage

- This class is primarily used for managing task data within the LearnoDoro application, including creating new tasks, loading tasks for a course, and deleting tasks when necessary.

LogsAndAnalytics

The `LogsAndAnalytics` class in the LearnoDoro application is responsible for logging study sessions and generating analytics based on the logged data. It interacts with a SQLite database to store and retrieve study session information.

Class Methods

`log_study_time(self, courseID, taskID, start_time, end_time, study_duration)`

- Logs a study session into the database.
- Parameters:
  - `courseID`: Identifier of the course.
  - `taskID`: Identifier of the task.
  - `start_time`: Timestamp marking the start of the study session.
  - `end_time`: Timestamp marking the end of the study session.
  - `study_duration`: Duration of the study session in minutes.

`study_time_distribution(self, courses, tasks=None, start_date=None, end_date=None)`

- Calculates the distribution of study time for selected courses and/or tasks within a specified date range.
- Returns a dictionary where keys are course or task IDs and values are the corresponding study durations.
- Parameters:
  - `courses`: List of course IDs.
  - `tasks`: Optional list of task IDs.
  - `start_date`: Optional start date for the period.
  - `end_date`: Optional end date for the period.

`get_daily_study_times(self, selected_courses, selected_tasks, start_date, end_date)`

- Retrieves daily study times for the selected courses and tasks within the given date range.
- Returns a dictionary with course/task IDs as keys and lists of tuples (date, study time) as values.
- Parameters:
  - `selected_courses`: List of selected course IDs.
  - `selected_tasks`: List of selected task IDs.
  - `start_date`: Start date of the period.
  - `end_date`: End date of the period.

`display_pie_chart(self, distribution)`

- Displays a pie chart visualizing the study time distribution.
- Parameters:
  - `distribution`: Dictionary with labels as keys and sizes as values.

`display_bar_chart(self, distribution)`

- Displays a bar chart showing the study time distribution.
- Parameters:
  - `distribution`: Dictionary with labels as keys and sizes as values.

`display_line_chart(self, daily_distribution)`

- Displays a line chart comparing daily study times for selected courses/tasks.
- Parameters:
  - `daily_distribution`: Dictionary with course/task IDs as keys and lists of (date, study time) tuples as values.

Database Interaction

- The class interacts with a SQLite database defined by `db_path` in the `config` module.
- Database operations include inserting study session records and querying study time distributions.

Visualization

- This class uses `matplotlib` to generate various types of charts for visualizing study data, including pie charts, bar charts, and line charts.

Usage

- This class is primarily used to log study sessions and generate analytics for visual representation of study patterns and durations in the LearnoDoro application.

Settings

The `Settings` class in the LearnoDoro application is designed to manage the settings for study sessions, including study time, break lengths, and intervals for long breaks. It provides functionalities to save and load these settings to and from a file.

Class Attributes

- Private attributes for storing settings for study time, short break length, long break length, and intervals for long breaks.

Initialization

`__init__(self)`

- Initializes the settings with default values:
  - `__study_time`: 25 minutes.
  - `__short_break_length`: 5 minutes.
  - `__long_break_length`: 30 minutes.
  - `__long_break_intervals`: 4 (interval count after which a long break is taken).

Properties and Setters

- `study_time`: Gets or sets the study time duration. Must be a `timedelta` object.
- `short_break_length`: Gets or sets the short break duration. Must be a `timedelta` object.
- `long_break_length`: Gets or sets the long break duration. Must be a `timedelta` object.
- `long_break_intervals`: Gets or sets the interval count for long breaks. Must be an integer.

Methods

`save_to_file(self, filename)`

- Saves the current settings to a file in JSON format.
- Parameters:
  - `filename`: The path of the file where settings will be saved.

`load_from_file(self, filename)`

- Loads settings from a specified file. If the file is not found, or its contents are not in the expected format, the method fails silently.
- Parameters:
  - `filename`: The path of the file from which settings will be loaded.

File Interaction

- The class uses the JSON format to save and load settings to and from a file.
- Exception handling is implemented to manage scenarios where the file might not exist or contain invalid data.

Usage

- This class is used to manage user-specific settings related to study times and break intervals in the LearnoDoro application, providing persistence of these settings across sessions.

config

This document explains the configuration settings for the LearnoDoro application, which are crucial for its operation in different environments: development, testing, and production.

Configuration Options

`DEVELOPMENT`

- A boolean flag used to indicate the development mode.
- Set to `True` during development for debugging and testing purposes.
- Set to `False` when deploying the application for production.

`TESTING`

- A boolean flag used to indicate the testing mode.
- Set to `True` when running automated tests.
- When set to `True`, along with `DEVELOPMENT=True`, data files (database and settings) are stored in a `Tests` directory for isolated testing.

File Path Configuration

Depending on the mode of operation (development, testing, or production), the application configures file paths for database, settings, sound, and icons differently:

`app_base_path`

- Determines the base path for application-related files.
- In a bundled PyInstaller environment, it points to the `data` directory within the `sys._MEIPASS` path.
- In a normal Python environment, it points to the `data` directory relative to the current file.

`data_base_path`

- Specifies the base path for data files like the database and settings.
- In development mode (`DEVELOPMENT=True`), it's set to the `data` directory within the application directory.
- In production mode (`DEVELOPMENT=False`), it uses the `LOCALAPPDATA` directory specific to the user's environment.
- In testing mode (`TESTING=True`), it points to a `Tests` directory, typically used for isolated test environments.

File Paths

- `db_path`: Path for the SQLite database file, named `LearnoDoroApp.db`.
- `settings_path`: Path for the settings file in JSON format, named `settings.json`.
- `sound_path`: Path for the sound file used in the application, relative to `app_base_path`.
- `icon_path`: Path for the application's icon file, also relative to `app_base_path`.

Directory Creation

- Ensures the creation of the `data_base_path` directory, making it if it doesn't exist.

Usage

- These settings are crucial for developers who want to modify LearnoDoro and then compile the application into an executable file.
- By adjusting `DEVELOPMENT` and `TESTING` flags, developers can control where the application stores its data, making it flexible for different stages of the development lifecycle.