Protracktor

Release 0.0.1

Quad-core

Nov 20, 2020

CONTENTS

Protracktor is a local proctoring and activity tracking application. It is designed to address the issues faced by the students of the IIT-Bombay in this online setup. The online environment of conducting classes & delivering lecture content, assignments, proctoring and evaluation has created several opportunities along with challenges towards which this application is projected.

Protracktor aims to let users manage their screen time and to ease the management of video proctoring by managing video recordings and activity tracker so that students are able to give uninterrupted exams without dealing with the pain of setting up video cam and screen recording everytime the connection is lost while being video proctored. The application will automatically detect loss of internet connection and start webcam and screen recording. It will also generate time-stamped activity statistics for whatever time the exam was scheduled so that any suspicious activity can easily be detected. This will both be beneficial for students as well as teachers for conducting remote proctoring exams. To acheive this following 4 options are added in the application:

- Webcam Recording
- Screen recording
- Activity Manager
- Data usage per app

CHAPTER

ONE

CONTENTS

Following are the list of modules in the application

1.1 check_internet Reference

Following are the classes and functions defined within this module:

`Project.system.check_internet.`**`check`**`()`

This method issues notification if Internet comes back.

Returns True or False

Return type bool

`Project.system.check_internet.`**`connect`**`(`*host='http://google.com'*`)`

This method checks internet connectivity by pinging google.com

Returns True or False

Return type bool

`class` `Project.system.check_internet.`**`myThread5`**`(`*val1*, *val2*, *course_name*`)`

`run``()`

This method controls the calling of other methods like checking of internet connectivity and calling notification functions.

`Project.system.check_internet.`**`notify`**`(`*val1*, *val2*`)`

This method issue notification to the user if Internet is down and starts the local proctoring thread .

Parameters

- **`val1`** (*`bool`*) – bool argument
- **`val2`** (*`bool`*) – bool argument

Returns True or False

Return type bool

1.2 data_usage_per_app Reference

Following are the classes and functions defined within this module:

class `Project.system.data_usage_per_app.`**`myThread3`**

`run`()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

1.3 screen Reference

Following are the classes and functions defined within this module:

class `Project.system.screen.`**`myThread1`**

`run`()

This method generates the Screen Recording file. It specifies the device screen resolution and takes screenshot using PyAutoGUI. The screenshot is converted to a numpy array. PyAutoGUI captures the screen in RGB(Red, Green, Blue) form and OpenCV converts it to BGR(Blue, Green, Red) and then writes that in an output file. The file is saved in ScreenRecordings folder and named by current date and time in .avi format.

Returns Screen Recording file

1.4 webcam Reference

Following are the classes and functions defined within this module:

class `Project.system.webcam.`**`myThread2`**

`run`()

This method captures video from Webcam using OpenCV and writes each frame of the video in an output file. The file is then saved in WebcamRecordings folder and named by current date and time in .avi format.

Returns Webcam Recording file

1.5 read_config Reference

Following are the classes and functions defined within this module:

`Project.system.read_config.`**`read_config`**()

This method reads the config file to get the course no. :returns: course no stored in the config file :rtype: str

`Project.system.read_config.`**`write_config`**(*str*)

This method updates the config file with new course no.

Parameters **`str`** – String argument

Return type str

1.6 window_activity Reference

Following are the classes and functions defined within this module:

`Project.system.window_activity.`**`currtime`**`(`*tformat=None*`)`

This method checks for the file and returns the time in respective format.

Parameters **`tformat`** (*`str or None`*) – *File* as the string argument

Returns time

Return type str

`Project.system.window_activity.`**`get`**`(`*command*`)`

This method gets the output of the `xdotool` commands and decodes it to utf-8 format

Parameters **`command`** (*`list`*) – `xdotool` commands are as argument to command parameter

Returns output of command in utf-8 format

Return type str

`class` `Project.system.window_activity.`**`myThread4`**

`run``()`

This method initiates the running of this module and continuosly runs the activity tracking with the help of other methods described here

`Project.system.window_activity.`**`plot`**`()`

This method plots the bar an *Application name Vs Percentage of time used* graph from the final csv file generated by `summarize()` method and saves it int the respective results directory.

`Project.system.window_activity.`**`summarize`**`(`*t*, *winlist*, *applist*`)`

This method performs the actual activity of listing the window activity within a text and CSV files in the respective results directory named corresponding to their timestamp to provide a detailed statistics of usage of each application.

It is repeatedly called by `run()` after a fixed time interval to update the values in the files.

Parameters

- **`t`** (*`int`*) – Total time in seconds
- **`winlist`** (*`list`*) – A list which stores the active tab within an application over the period for which the application ran
- **`applist`** (*`list`*) – A list which stores active applications over the period for which the application ran

`Project.system.window_activity.`**`time_format`**`(`*s*`)`

This method coverts cumulative time in seconds into HH:MM:SS

Parameters **`s`** (*`int`*) – total seconds time

Returns time in HH:MM:SS format

Return type str

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX