Simplii- Component Description

In this file, we describe the organization of our code.

Contents

Simplii

Meta Files

We list out all the non-functional files present in the repository here.

CITATION.md

Contains information about how to cite this work.

CODE_OF_CONDUCT.md

Lays out guidelines for people who wish to contribute to this work.

CONTRIBUTING.md

Lays out instructions for people who wish to contribute to this work.

LISCENCE

Specifies the licence under which this work is released.

requirements.txt

Lists out the dependencies for this software.

Code

We describe all the Code Components present in the repository in this section

app.py

Global variables

TODO_TASKS_PATH

This variable is used to store the path of the directory that stores all the tasks currently present in the TODO-list.

COMPLETED_TASKS_PATH

This variable is used to store the path of the directory that stores all the tasks currently present in the COMPLETED-list. We move the tasks from the TODO-list to the COMPLETED-list once the user says he/she has completed the task.

ALL_QUOTES

This variable reads all the quotes in out quote-database and stores them as a list of strings. We save this variable as a global variable because the quote database remains constant throughout the lifetime of the software. The index of a quote in ALL_QUOTES is the same as its author in ALL_AUTHORS.

ALL_AUTHORS

This variable reads all the authors in out quote-database and stores them as a list of strings. We save this variable as a global variable because the quote database remains constant throughout the lifetime of the software. The index of a quote in ALL_QUOTES is the same as its author in ALL_AUTHORS.

Functions

refresh_data()

Description:

This function loads all the data required to display the home page from file system.

If API, url to hit: Not an API

Inputs: No Inputs

Outputs:

The function returns a JSON (Python dictionary) object which is specified below:

```
data = {
      "name_block":{
            "initialized": “yes” or “no”,
            "name": UserName in a string format,
            "email_id": Email ID in a string format,
            "email_notifications": “yes” or “no”
            },
      "quote_block":{
            "quote": Some Quote in a string format,
            "author": Corresponding Author in a string format
            },
      "task-list-block":{
            "task-list": tasks (schema described below)
            }
    }

tasks = {
      “task_ID1”:{
            “task_name”:Name of the task in a string format,
            “estimate”:User’s work estimate in hours,
            “deadline”:Task’s due-date in a string format
      }
      “task_ID2”:{
            “task_name”:Name of the task in a string format,
            “estimate”:User’s work estimate in hours,
            “deadline”:Task’s due-date in a string format
      }
      .
      .
      .
      “task_IDN”:{
            “task_name”:Name of the task in a string format,
            “estimate”:User’s work estimate in hours,
            “deadline”:Task’s due-date in a string format
      }
}
```

getnewTaskID()

Description:

This function is used to generate a new unique TaskID. All our TaskIDs are alphanumeric strings of length 5. Eg. “AAAA2”, “2342V”. No special characters are used for obtaining TaskIDs.

This function randomly produces a possible TaskID and checks if it is already in use. If the generated TaskID is in use, then we keep on generating random TaskIDs until we find one that is not in use.

If API, url to hit: Not an API

Inputs: No Input

Outputs:

The function returns an alphanumeric string of length 5 which is not being used by any other task (todo or completed) in the system.

```
possible_id: An alphanumeric string of length 5
```

homePage()

Description:

This function renders the home page.

If API, url to hit: “/” [GET]

Inputs: No Inputs

Outputs:

This function renders the template “index.html” present in the “templates” folder on the browser.

update_user_information()

Description:

This API updates the user information like name, email_id and redirects to home page.

If API, url to hit: “/update_user_info” [POST]

Inputs:

This API is triggered by the submit button of the “Change Preferences” form and all the data of the form is sent along with the HTTP request in the following format.

```
request.form = {
      “name”:Updated name of the user in a string format,
      “email_id”:Updated Email ID in a string format,
      “initialized”:This is always set to “yes” after the first time,
      “email_notifications”: “yes” or “no”
}
```

Outputs:

This function renders the template “index.html” present in the “templates” folder on the browser.

add_new_task()

Description:

Adds a new task to the to-do list.

If API, url to hit: "/add_task" [POST]

Inputs:

This API is triggered by the submit button of the "Add Task" form and following values are sent along with the HTTP request in the following format.

```
request.values = {
      "taskName":Name of the task in a string format,
      "deadline":Due date of the task in a string format,
      "estimateInput": User's work estimates in hours,
      "taskType":"physical" or "intellectual",
      "quant/verbal": "NA" if task is physical or a number in [0,5],
      "contentconsump": "NA" if task is physical or a number in [0,5]
}
```

Outputs

This function renders the template "index.html" present in the "templates" folder on the browser.

delete_task_byID ()

Description:

Deletes a task from the to-do list given its ID. This API moves a task from the TODO list to the COMPLETED list.

If API, url to hit: "/delete_task" [POST]

Inputs:

This API is triggered by the submit button of the "Delete Task" buttons in the Tasks table. The following values are sent along with the HTTP request.

```
request.values = {
      "id": TaskID in string format
}
```

Outputs:

This function renders the template "index.html" present in the "templates" folder on the browser.

docs

This folder contains various documentation files including the one you are currently reading.

architectureImage

This folder contains our project's ER diagram and the Flow-chart for our webapp.

lintingEvidence

This folder contains evidence of linting which is required by the Project grading rubric.

proj1rubric.md

This is our self-evaluation sheet.

VideoPresentation.mp4

This is the short, animated video which introduces our project.

scripts

All our JavaScript functions are stored in the scripts folder.

name-block-scripts.js

This file contains all the JavaScript code to display and control the BootStrap Modal and other scripts used by the name-block components. This block uses the following APIs:

- update_user_information()

task-block-scripts.js

This file contains all the JavaScript code to display and control the Add Tasks form and the table that displays the task-list as well as other scripts used by the task-block components. This block uses the following APIs:

- add_new_task()
- delete_task_byID()

static

static/images

All images displayed in our project are stored in this directory.

static/tasks

todo

This directory contains all the information about tasks that are in the TODO list.

completed

This directory contains all the information about task that are in the completed list.

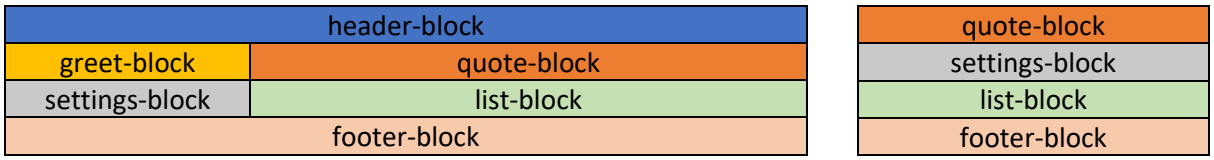
quotes.csv

This file contains our quotes database with two columns: “quote” and “author”

styles.css

This file contains our custom CSS which is used to define the layout of our project. The layout of our project is as follows:

Fig 1: (a) Layout for screen with width >= 1000px; (b) Layout for screen with width < 999 px



This file also contains CSS to individually style different components of the page.

user_infromation.json

This file contains all the information about the current user in the following format:

```
{
      “name”:Name of the user in a string format,
      “email_id”: Email ID in a string format,
      “initialized”:This is always set to “yes” after the first time,
      “email_notifications”: “yes” or “no”
}
```

templates/index.html

This is the jinja2 template that is rendered by the flask server to generate our home page. In the