\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_

/\ \ /\ \ /\ \ /\ \ /\ \ /\ \ /\ \ /\ \ /\ \ /\ \ /\ \

/::\ \ /::\ \ /::\ \ /::\\_\_\_\_\ /::\ \ /::\ \ /::\ \ /::\ \ /::\ \ /::\ \ /::\ \

\:::\ \ /::::\ \ /::::\ \ /:::/ / /::::\ \ /::::\ \ /::::\ \ \:::\ \ /::::\ \ /::::\ \ \:::\ \

\:::\ \ /::::::\ \ /::::::\ \ /:::/ / /::::::\ \ /::::::\ \ /::::::\ \ \:::\ \ /::::::\ \ /::::::\ \ \:::\ \

\:::\ \ /:::/\:::\ \ /:::/\:::\ \ /:::/ / /:::/\:::\ \ /:::/\:::\ \ /:::/\:::\ \ \:::\ \ /:::/\:::\ \ /:::/\:::\ \ \:::\ \

\:::\ \ /:::/\_\_\:::\ \ /:::/ \:::\ \ /:::/\_\_\_\_/ /:::/\_\_\:::\ \ /:::/\_\_\:::\ \ /:::/\_\_\:::\ \ \:::\ \ /:::/ \:::\ \ /:::/\_\_\:::\ \ \:::\ \

/::::\ \ /::::\ \:::\ \ /:::/ \:::\ \ /::::\ \ /::::\ \:::\ \ /::::\ \:::\ \ \:::\ \:::\ \ /::::\ \ /:::/ \:::\ \ \:::\ \:::\ \ /::::\ \

/::::::\ \ /::::::\ \:::\ \ /:::/ / \:::\ \ /::::::\ \ \_\_\_\_\_ /::::::\ \:::\ \ /::::::\ \:::\ \ \_\_\_\:::\ \:::\ \ \_\_\_\_ /::::::\ \ /:::/ / \:::\ \ \_\_\_\:::\ \:::\ \ \_\_\_\_ /::::::\ \

/:::/\:::\ \ /:::/\:::\ \:::\ \ /:::/ / \:::\ \ /:::/\:::\ \ /\ \ /:::/\:::\ \:::\ \_\_\_\ /:::/\:::\ \:::\ \ /\ \:::\ \:::\ \ /\ \ /:::/\:::\ \ /:::/ / \:::\ \ /\ \:::\ \:::\ \ /\ \ /:::/\:::\ \

/:::/ \:::\\_\_\_\_\/:::/\_\_\:::\ \:::\\_\_\_\_\/:::/\_\_\_\_/ \:::\\_\_\_\_\/:::/ \:::\ /::\\_\_\_\_\ /:::/\_\_\:::\ \:::| |/:::/ \:::\ \:::\\_\_\_\_\/::\ \:::\ \:::\\_\_\_\_\/::\ \/:::/ \:::\\_\_\_\_\/:::/\_\_\_\_/ \:::\\_\_\_\_\/::\ \:::\ \:::\\_\_\_\_\ /::\ \/:::/ \:::\\_\_\_\_\

/:::/ \::/ /\:::\ \:::\ \::/ /\:::\ \ \::/ /\::/ \:::\ /:::/ / \:::\ \:::\ /:::|\_\_\_\_|\::/ \:::\ /:::/ /\:::\ \:::\ \::/ /\:::\ /:::/ \::/ /\:::\ \ \::/ /\:::\ \:::\ \::/ / \:::\ /:::/ \::/ /

/:::/ / \/\_\_\_\_/ \:::\ \:::\ \/\_\_\_\_/ \:::\ \ \/\_\_\_\_/ \/\_\_\_\_/ \:::\/:::/ / \:::\ \:::\/:::/ / \/\_\_\_\_/ \:::\/:::/ / \:::\ \:::\ \/\_\_\_\_/ \:::\/:::/ / \/\_\_\_\_/ \:::\ \ \/\_\_\_\_/ \:::\ \:::\ \/\_\_\_\_/ \:::\/:::/ / \/\_\_\_\_/

/:::/ / \:::\ \:::\ \ \:::\ \ \::::::/ / \:::\ \::::::/ / \::::::/ / \:::\ \:::\ \ \::::::/ / \:::\ \ \:::\ \:::\ \ \::::::/ /

/:::/ / \:::\ \:::\\_\_\_\_\ \:::\ \ \::::/ / \:::\ \::::/ / \::::/ / \:::\ \:::\\_\_\_\_\ \::::/\_\_\_\_/ \:::\ \ \:::\ \:::\\_\_\_\_\ \::::/\_\_\_\_/

\::/ / \:::\ \::/ / \:::\ \ /:::/ / \:::\ /:::/ / /:::/ / \:::\ /:::/ / \:::\ \ \:::\ \ \:::\ /:::/ / \:::\ \

\/\_\_\_\_/ \:::\ \/\_\_\_\_/ \:::\ \ /:::/ / \:::\/:::/ / /:::/ / \:::\/:::/ / \:::\ \ \:::\ \ \:::\/:::/ / \:::\ \

\:::\ \ \:::\ \ /:::/ / \::::::/ / /:::/ / \::::::/ / \:::\ \ \:::\ \ \::::::/ / \:::\ \

\:::\\_\_\_\_\ \:::\\_\_\_\_\ /:::/ / \::::/ / /:::/ / \::::/ / \:::\\_\_\_\_\ \:::\\_\_\_\_\ \::::/ / \:::\\_\_\_\_\

\::/ / \::/ / \::/ / \::/\_\_\_\_/ \::/ / \::/ / \::/ / \::/ / \::/ / \::/ /

\/\_\_\_\_/ \/\_\_\_\_/ \/\_\_\_\_/ ~~ \/\_\_\_\_/ \/\_\_\_\_/ \/\_\_\_\_/ \/\_\_\_\_/ \/\_\_\_\_/ \/\_\_\_\_/

\_\_.....\_\_ \_\_ \_\_ \_\_\_

.-'' '. | |/ `.' `.

.| .-,.--. / .-''"'-. `. | .-. .-. '

.' |\_ | .-. |/ /\_\_\_\_\_\_\_\_\ \ \_\_ | | | | | | \_\_

\_ .' || | | || | .:--.'. | | | | | | .:--.'.

.' |'--. .-'| | | |\ .-------------'/ | \ || | | | | | / | \ |

. | / | | | | '- \ '-.\_\_\_\_...---.`" \_\_ | || | | | | | `" \_\_ | |

.'.'| |// | | | | `. .' .'.''| ||\_\_| |\_\_| |\_\_| .'.''| |

.'.'.-' / | '.'| | `''-...... -' / / | |\_ / / | |\_

.' \\_.' | / |\_| \ \.\_,\ '/ \ \.\_,\ '/

`'-' `--' `" `--' `"

,-.----.

,---,. ,--, \ / \ \_\_\_

,' .' | ,--, ,--.'| | : \ ,--.'|\_

,---.' |,--.'| ,---, | | : | | .\ : \_\_ ,-. ,---. .--. | | :,'

| | .'| |, ,-+-. / | : : ' . : |: |,' ,'/ /| ' ,'\ .--,`| : : ' :

: : : `--'\_ ,--.'|' | ,--.--. | ' | | | \ :' | |' | / / | | |. ,---. ,---. .;\_\_,' /

: | |-,,' ,'| | | ,"' | / \ ' | | | : . /| | ,'. ; ,. : '--`\_ / \ / \| | |

| : ;/|' | | | | / | |.--. .-. || | : ; | |`-' ' : / ' | |: : ,--,'| / / | / / ':\_\_,'| :

| | .'| | : | | | | | \\_\_\/: . .' : |\_\_ | | ; | | ' ' | .; : | | '. ' / |. ' / ' : |\_\_

' : ' ' : |\_\_ | | | |/ ," .--.; || | '.'| : ' | ; : | | : | : | |' ; /|' ; :\_\_ | | '.'|

| | | | | '.'|| | |--' / / ,. |; : ; : : : | , ; \ \ / \_\_| : '' | / |' | '.'| ; : ;

| : \ ; : ;| |/ ; : .' \ , / | | : ---' `----'.'\_\_/\\_: || : || : : | , /

| | ,' | , / '---' | , .-./---`-' `---'.| | : : \ \ / \ \ / ---`-'

`----' ---`-' `--`---' `---` \ \ / `----' `----'

`--`-'



Madeleine Chan

2018 August

Lecturer: Helena Lingor

Leuphana Univeristy, Lüneburg

# Introduction

In this project I will build an application with the codename: Stint. This application will provide you with a very anarchic movie archive, whose variations grace many classic digital art creations.

Besides, I aimed to show off the results of Python study in Technological Basics I while remaining playful and emphasized on artistic aspects throughout the development.

Through some skimming of ASCII art and early digital art creations, I am trying to blend exhibits of fun digital arts from pioneers such as Knowlton and Schwartz into a more amusing user experience.

Stint does comprise of some basic things including addition, modification, search and deletion of media.

Furthermore, it additionally equips functionalities like sharing, peer recommendations and informative trivia which I have planned to implement in future releases.

To start with, I will incorporate Stint the ability to reflect music and any artistic background about its media repository.

By and large, Stint has four prime functions

• Maintain a bunch of media large or small.

• Report the media it kept in addition to their fringe interests.

• Look up your own media and from peers.

• Have fun with classy digital art.

# Presentation of the main idea

My goal is to build an application code named Stint that manages media repository for example movie collections utilising Python Flask framework.

Extended functionalities such as automated collection of the media’s associated music and any background information, also any share recommendations will be provided.

In order to enrich user experiences, the interface should be enriched with Kwonlton’s photographic mosaic and Schwartz’s Pixillation arts.

The Next steps is to extent Stint with image capability and to visualise data.

## Design approaches

The higher-level functions of Stint are developed into more detailed ones in stages so it is progress in a traceable manner.

In order to make the implementation intellectually manageable it is further divided into several modules, this reduces chances of errors as different modules can be developed and tested separately.

The program is split into components, which describe the interaction between its parts and data structures that clarifies the conceptual integrity.

The organisation of the program elaborates mechanism in modules & their sub-modules to display a hierarchical relationship between them.

Stint is divided into modules like data maintenance, login and data collection, etc.

The pre & post conditions and operations to be performed are described in these modules.

## Software Architecture

This is the framework for Stint itself.

This serves to help ensure a good design that can satisfy general requirements in areas such as performance, reliability, scalability, and interoperability.

And it is shown below as a Layered style:

## The Programming environment

The python platform is so flexible and open, choosing a particular set of tools to construct Stint was a little struggle.

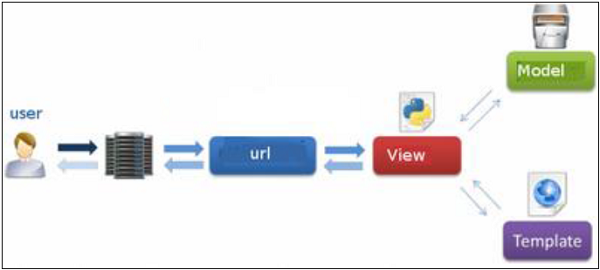
Although I knew I would start with Flask, there are still many options to choose from and making such a choice is hard for some silly aboulomania like me.

Visual Studio community was chosen because it’s free, aids editing for novice users and all-in-one. By all-in-one I mean I was advertised by some web articles that the debugger will be helpful but it’s yet to be realised.

## Framework

Stint is developed using Flask framework which provides advantages as:

* It was taught in Tech Basics, so I can just leverage the knowledge.
* it is simpler than Django.



**Flask**

Testing

For the formality, it is a process of technical investigation, performed on behalf of stakeholders f.i. you, the user, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate.

Testing process deals with the verification and validation of a system to ensure that it meets requirements.

Due to time constraints, thorough testing is not feasible at this stage and I intended to leave it to later parts of Stint’s development.

# Description of functionalities

Through some skimming of ASCII art and early digital art creations, I am trying to blend exhibits of fun digital arts from pioneers such as Knowlton and Schwartz into a more amusing user experience.

Stint does comprise of some basic things including addition, modification, search and deletion of media.

Furthermore, it additionally capable of functionalities like sharing, peer recommendations and informative trivia which I have planned to implement in future releases.

To start with, I will incorporate Stint the aptitude to reflect music and any artistic background about its media repository.

By and large, Stint has four prime functions

• Maintain a bunch of media large or small.

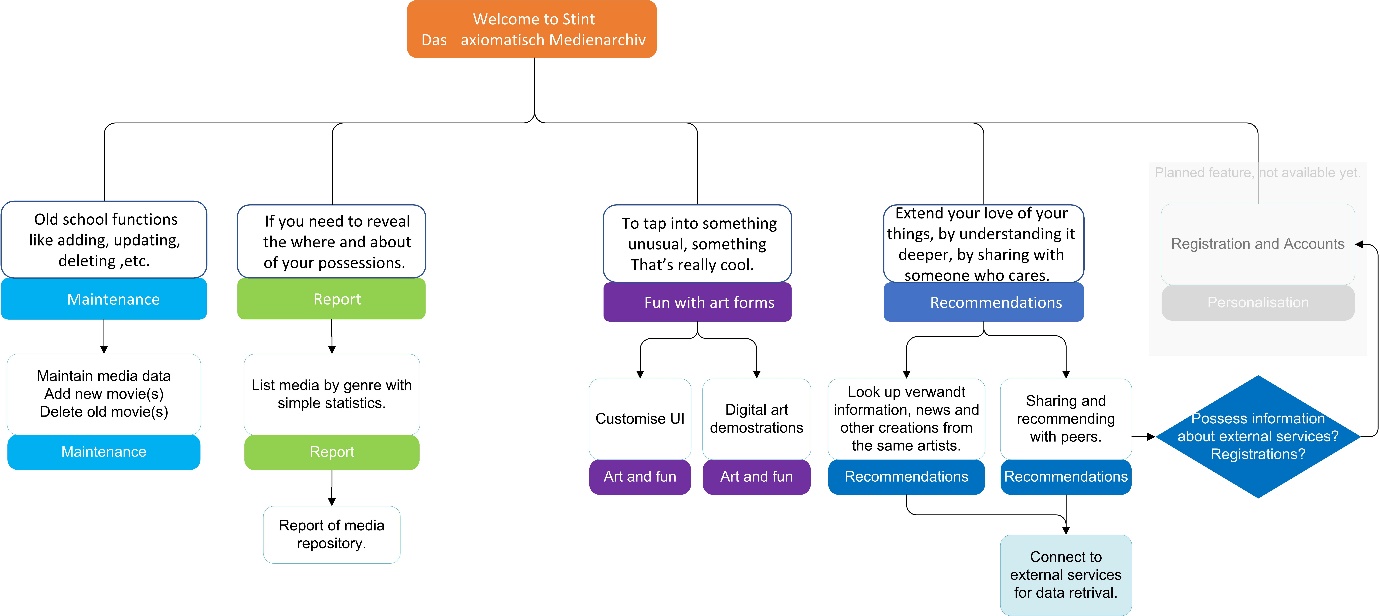
• Report the media it kept in addition to their fringe interests.

• Look up your own media and from peers.

• Have fun with classy digital art.

# Description of the main structure

## Functional Flow chart of Stint



Targeting operational and clean code, my principal is to keep the logic and dependencies in my scripts, files and folder structures clear.

This leads to plan for how does the data flow and what features and functions to group together.

It is relatively easy to build a python project by adhering to the limited constraints and the model for importing modules. However, constructing various components of my project and their interactions is still challenging.

# Description of used modules / libraries / frameworks

This section is rather incomplete since many of the components of Stint are still in code fragments status and new functions are to be added to as I learn more.

* Flask framework, provides the basis of this web application.
* Pyfiglet, provides function of creation ASCII arts from freely available and home-made Figlets, which are the fonts of this particular type of ASCII art.
* SQLAlchemy, provides data manipulate ability.

# Possible extensions of the currently feasible

Fuller development of current scripts so as to provide a more completed application flow and functions.

## Unfeasible parts

Most of the part of Stint are still at code fragments stage, so the greatest stopper would be time constraints.

## Inference of tools, skills

I would fancy visualisation of data and combine some hardware integration as forecasted in Technological Basics course outlines.

Testing

Due to time constraints, thorough testing is not feasible at this stage and I intended to leave it to later parts of Stint’s development.

Addition to the ASCII arts’ pleasure to include more of this kind of compilations, maybe mainly written in python.

## knowledge necessary for accomplishment

One of the first next steps is to fathom the Flask WTF extension, that can make most of the functions of Stint that require user input much simpler to write.

A possible further step is to acquire OAuth framework, so fridge information of the media repository and peer recommendations would be largely probable.

## Estimation of time required to acquire those skills

|  |  |  |
| --- | --- | --- |
| Skill | Time Estimation | Enables |
|  | Main function |  |
| Flask framework | 3 weeks | Full function flow |
| SQLAlchemy | 2 weeks | Data manipulate |
| Digital arts extensions | 3 weeks | Art and fun provision. |
|  | Extensions |  |
| OAuth framework | 4 weeks | Registration and peer functions. |
|  |  |  |
|  | Further functions |  |
| Data visualization | 3 weeks | Data visualization |
| Hardware capability | 3 weeks | Hardware capability |
|  |  |  |
| Total | 8 weeks |  |

\* Some time durations can be overlapping, so the total time is shorter.

# Conclusion

I would foresee that as more coding and testing progress on Stint, the application will become more refined and in-line with the evolving skills of its creator - that’s me, and less shortcomings will be seen. If you care enough to stay tuned of the progress of Stint, you may be increasing surprised by how it delights you as it will become more mature and bugs free over time.

It is worthwhile to look at the vast possibilities that Python can offer including testing facility such as nose framework.

# Credits:

Text to ASCII Art Generator (TAAG) http://patorjk.com/software/taag/

https://www.artinamericamagazine.com/reviews/lillian-schwartz/

https://hyperallergic.com/329466/paying-tribute-lillian-schwartz-computer-art-pioneer/

Java Ascii Versatile Editor http://www.jave.de/

https://docs.python-guide.org/dev/virtualenvs/

https://code.visualstudio.com/docs/python/tutorial-flask

https://docs.microsoft.com/en-us/visualstudio/python/learn-flask-visual-studio-step-01-project-solution

Python Figlet https://github.com/pwaller/pyfiglet

Pankaj Kumar Principles and Practices of Software Production (CE00003-2)

Josefine Raab Bus Routes London App for Tech Basics II

Helena Lingor Technological Basics Python 2018

Kenneth Reitz The Hitchhiker’s Guide To Python

Miguel Grinberg Flask Web Development

Joël Perras Flask Blueprints

Gareth Dwyer Flask By Example

Serge Demeyer Object-Oriented Reengineering Patterns

How to code in Python 3 https://www.digitalocean.com/community/tutorials/

https://ymichael.com/2014/12/17/python-testing-with-nose-for-beginners.html