

ART SPACE

CRUD Web App

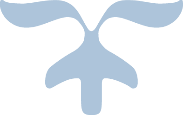


Table of Contents

[INTRODUCTION 1](#_gjdgxs)

[Related Work 1](#_30j0zll)

[Methods 1](#_1fob9te)

[RELATIONAL SCHEMA 2](#_2et92p0)

[CHALLENGES 3](#_tyjcwt)

[FUTURE WORK 4](#_3dy6vkm)

[CONCLUSION 4](#_1t3h5sf)

[REFERENCES 4](#_4d34og8)

ART SPACE

## INTRODUCTION

PROBLEM STATEMENT: Art galleries handle precious art and their information needs to be handled just as delicately as the work they sell.

Art space is a web-based database management system.

## Related Work

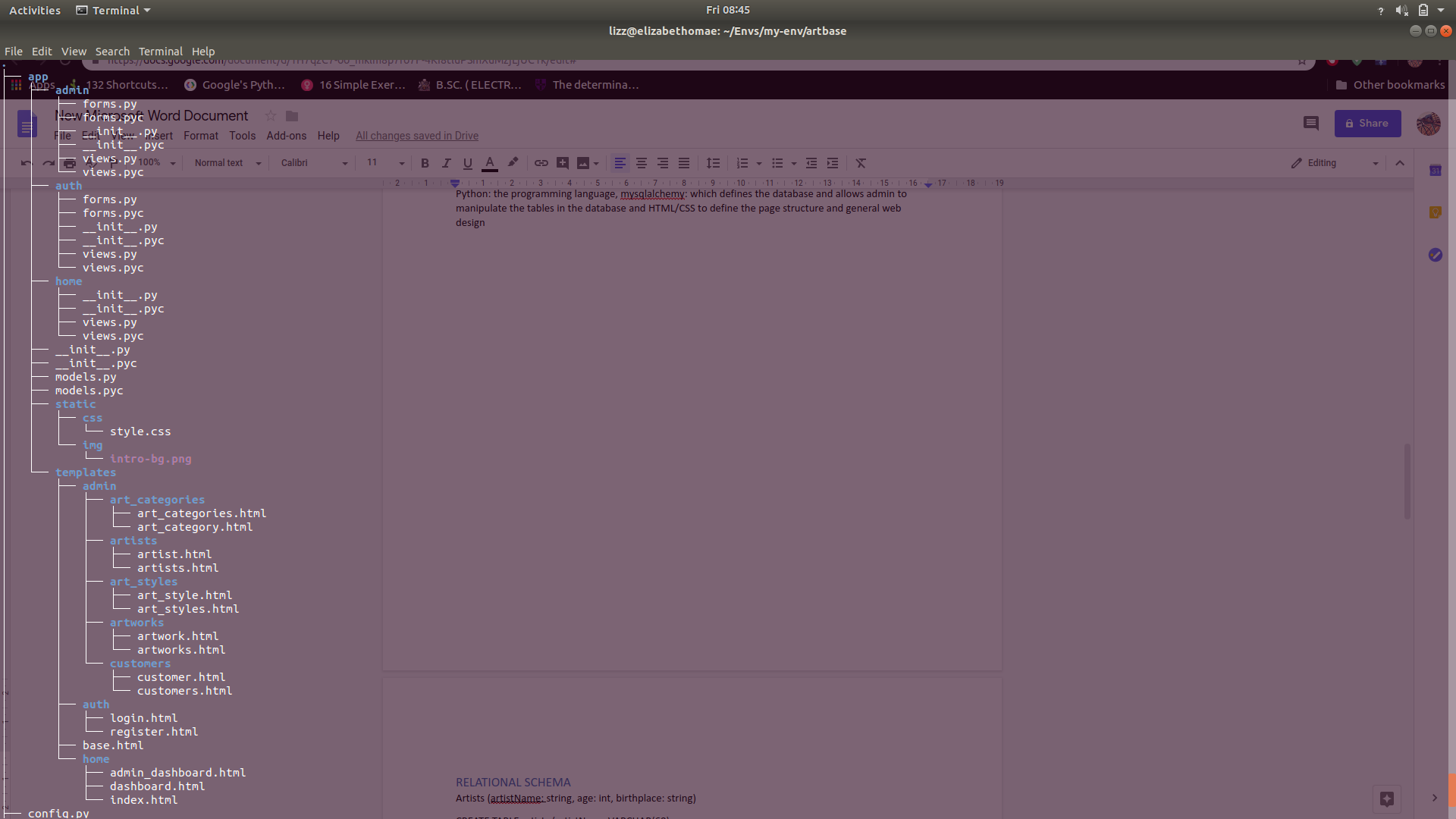
Artlogic (<https://artlogic.net>)

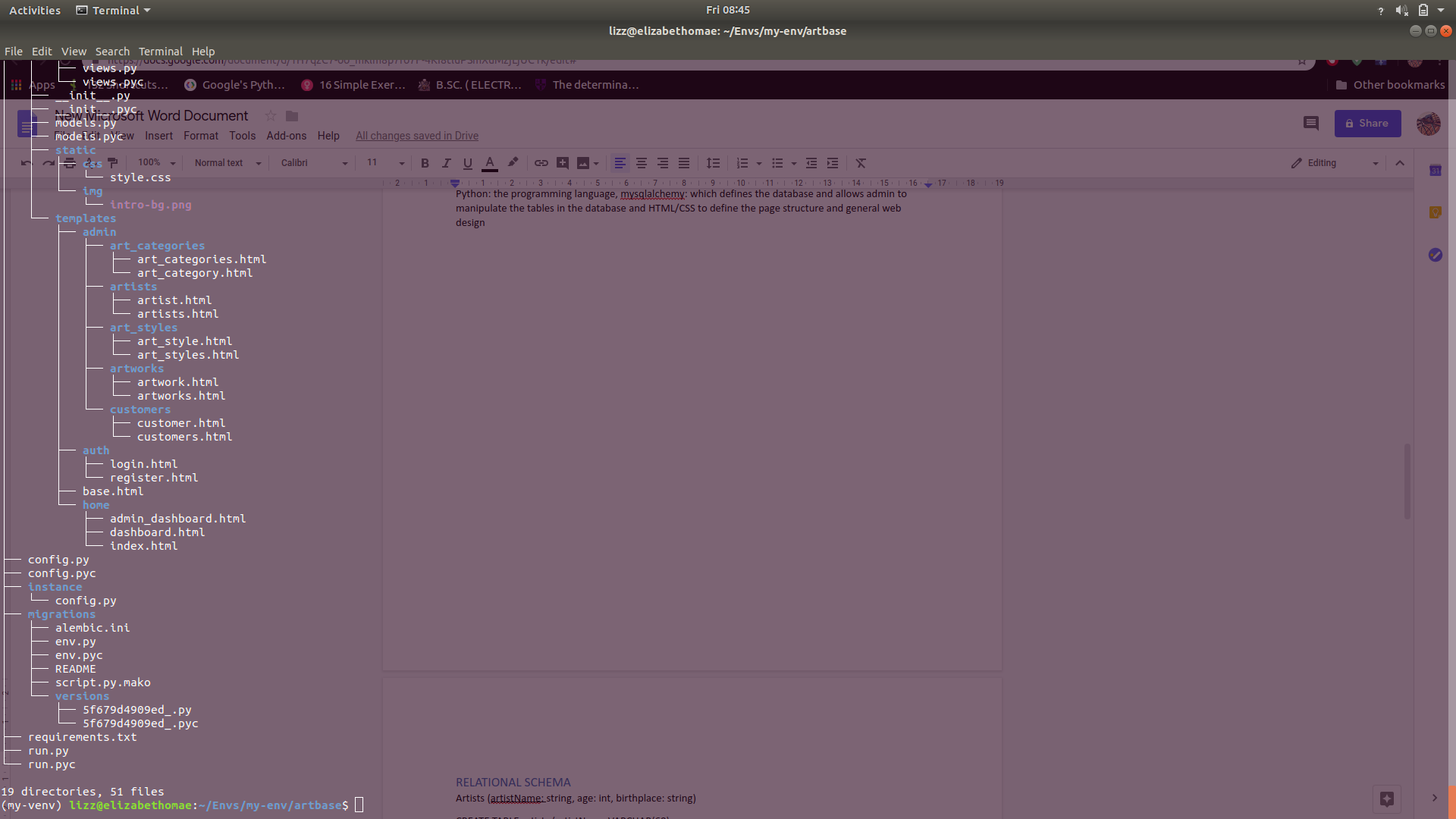
Artlogic is, simply put, Art space on cutting edge steroids. It is commercial web-based gallery data management system

## Methods

Art space utilizes Flask: which essentially links the database to the raw python code and the internet, Python: the programming language, mysqlalchemy: which defines the database and allows admin to manipulate the tables in the database and HTML/CSS to define the page structure and general web design

Directory Structure





## RELATIONAL SCHEMA

Artists (artistName: string, age: int, birthplace: string)

CREATE TABLE artists (artistName VARCHAR(60),

age INTEGER,

birthplace VARCHAR(60)

PRIMARY KEY(artistName),

)

Art styles (styleName: string)

CREATE TABLE Art\_Styles (styleName VARCHAR(60),

PRIMARY KEY (styleName),

)

Artworks (title: string, type: string, price, year, creator: string)

CREATE TABLE Artworks (title VARCHAR(60),

type VARCHAR(60)

price INTEGER,

year INTEGER,

creator VARCHAR(60),

PRIMARY KEY(title),

FOREIGN KEY(artistName)REFERENCES artists(artistName),

)

Art categories (categoryName: string)

CREATE TABLE Art\_Categories (categoryName VARCHAR(60),

PRIMARY KEY (categoryName))

Customers (id: string, customerName: string, address: string, expenditure: int)

CREATE TABLE Customers (id INTEGER,

CustomerName VARCHAR(60),

address VARCHAR(60),

expenditure INTEGER

PRIMARY KEY(customerName),

)

Categorizing

CREATE TABLE categories (title VARCHAR(60)

categoryName VARCHAR(60)

` FOREIGN KEY (title) REFERENCES artwork(title)

FOREIGN KEY (categoryName) REFERENCES Art\_Categories(categoryName),

)

Styles

CREATE TABLE styles (artistName VARCHAR(60)

styleName VARCHAR(60)

` FOREIGN KEY (artistName) REFERENCES artists(artistName),

FOREIGN KEY (styleName) REFERENCES Art\_Styles(styleName)

)

artistPreference

CREATE TABLE artistPreference ( artistName VARCHAR(60),

customerName VARCHAR(60),

` FOREIGN KEY (artistName) REFERENCES artwork(artistName),

FOREIGN KEY (customerName) REFERENCES customers(customerName)

)

categoryPreference

CREATE TABLE categoryPreference (customerName VARCHAR(60),

categoryName VARCHAR(60),

` FOREIGN KEY (customerName) REFERENCES customers(customerName),

FOREIGN KEY (categoryName) REFERENCES Art\_Categories(categoryName)

)

## CHALLENGES

During the working I would get indiscernible logic errors that more often than not led to restart the work keeping only essential files such as the models.py.

In editing entities that are assigned values from another in many to many relationship, editing and deleting seemed mutually exclusive and it was difficult to delete certain entities after editing.

To beat this I made the foreign keys taken into association tables to be primary in the association tables so that there are no duplicate entrys therefore it’s possible to delete the entities. However, this caused an error in editing since the app does not allow duplicate entry, therefore every time one needs to edit, fields influenced by many to many relationship need to be changed

## FUTURE WORK

If I had more time I would work on the user interface to make it more appealing as well as make the app more interactive, giving it more functionality since right now only admin has functionality.

## CONCLUSION

Objectively, I was successful at creating a three-tier CRUD web app with a few hiccups on certain editing.

## REFERENCES

Template files from Mbithe Nzomo’s Build A CRUD Web App with Python and Flask.

APPENDIX

