Overview

The purpose of this document is to track the active set of requirements that are used for the CSCI 2340: Software Engineering Art Generation semester-long project. Throughout the semester, this document will be maintained to accurately reflect the requirements that are used within the system. The requirements are broken into separate sections: Art Visualization (AG_ART_REQ), Export Output (AG_EXPORT_REQ), Security (AG_SEC_REQ), System (AG_SYS_REQ), and User Interface (AG_UI_REQ) requirements.

Requirements

Art Visualization

This section contains the requirements relative to the Art Visualization and Export Output for GENERATIVE

1. **Art**
   a. *AG_ART_REQ_1*: ***User*** can choose from Five (5) different art templates to start, which artistically visualize the following socio-environmental data:
      i. Atmospheric Gas Levels
      ii. Social Housing Displacement
      iii. Melting Ice Caps
      iv. Fluctuating Ocean Temperatures
      v. Rising Global Temperatures
   b. *AG_ART_REQ_2*: ***User*** can manipulate art templates' (through UI e.g. RGBA sliders). I.e. Art responds meaningfully to the user input through these aspects.
      i. Color, Texture, & Noise
      ii. Dynamic Movement Shape & Amplitude
      iii. Dynamic Movement's Speed, Frequency, & Entropy
   c. *AG_ART_REQ_3*: System will update certain patches daily by pulling a NASA Image of the Day into the project repository
2. **Export**
   a. *AG_EXPORT_REQ_1*: ***User*** can generate and export Art as `.mov` files
   b. *AG_EXPORT_REQ_2*: All Art files are 5 seconds long

Security

This section contains the requirements relative to Security for the Art Generation project.

AG_SEC_REQ_1
Requirement: The application MUST ensure that data privacy policies are in place and adhered to when handling user data.
Reasoning: We want to let the user feel secure to login into our web applicant with zero risk.

System

This section contains the requirements relative to the system of the Art Generation project.

1. **Requirement**: The system should contain relatively low latency for processing the art generation.
   Reasoning: Users most likely will not want to watch slow, non-realtime graphics.

2. **Requirement**: The user can download the generated art as either an image or video file of a maximum of 10 seconds to their local device.
   Reasoning: Allows the user to save interesting outputs and manually share at their own will.

3. **Requirement**: The user can store their generated art on their profile, and revisit the saved art in a gallery on their user profile page.
   Reasoning: Allows the user to save interesting outputs to their profile without requiring manual download onto their device.

4. **Requirement**: The user can select from multiple environmentally relevant datasets that inform the art generation in meaningful ways.
   Reasoning: Having multiple databases controlling art generation adds variation to the art generation and creates abstract representations of meaningful data.

User Interface

This section contains the requirements relative to the User Interface of the Art Generation project.

AG_UI_REQ_1
Requirement: The user should be able to easily read the information page and understand what the website is about.
Reasoning: To lower the barrier for first-time users, it is crucial that users clearly understand how to use our website to generate art. This enhances user experience and increases user engagement.

AG_UI_REQ_2
Requirement: The user should be able to adjust and tune the input parameters for creating art using intuitive sliders/input boxes on the control panel page
Reasoning: Allowing users to manipulate parameters themselves provides a deeper sense of personal involvement and the creation of their art pieces. This level of customization enhances user engagement and satisfaction.

AG_UI_REQ_3
Requirement: The user should be able to select different datasets to generate different effects of the art generator.
Reasoning: Since we have multiple datasets in our system, letting the user choose which dataset is impacting their art generation is important for user customization.

AG_UI_REQ_4
Requirement: The user interface should be intuitive and easy to use.
Reasoning: The art generation project should not include too many options. Users should really focus on input selection, output display, and output sharing. Keeping a simple and intuitive UI is important to not scare users away.

AG_UI_REQ_5
Requirement: Users should be able to view their newly generated art after it has been processed in the backend.
Reasoning: Viewing the art they have generated is a main point of interest for users interested in the app.

AG_UI_REQ_6
Requirement: The user should be able to easily login/logout using their existing Gmail account.
Reasoning: Providing the ability to sign in with a Gmail account streamlines the registration process, and reduces the complexity for users to create a new account. This simplifies access and enhances user convenience.

AG_UI_REQ_7
Requirement: The user should be able to view their home profile and view the artwork they have generated and saved.
Reasoning: Enabling users to access a personalized home profile where they can view their saved artwork enhances the user experience by allowing them to easily revisit and manage their creations. This feature supports user engagement and satisfaction.

AG_UI_REQ_8
Requirement: The user should be able to view the controls/metadata associated with generating their saved artworks.
Reasoning: Enables users to recall the settings they used to create the given art, which is helpful for recreating art and understanding how the output they're viewing was generated originally.

# Think about this granularity of every aspect of design that we have? What design choices have you had to make in your own system? What design choices have you had to make when integrating with other teams? Completing a list of these can lead to a huge list of trackable specifications.

# We need specs for the different data sets we are going to use
# We need specs for the different modules we are going to use
# We need specs for how we are going to control the modules
# We need specs for codecs, for data transmission, file types, etc
# Any data being sent needs a requirement for why it is being sent and specifications for how it is being sent
# Specifications for number of videos that can be saved, number of data sets that can be uploaded

Requirements

User

- Social Media Connectivity
- Users can download a video
- Users can save videos to their profile
- Users can create an account with google authorization
- Users should be able to upload their own videos or images as sources
- Users should be able to upload presets for their own data sets to be used by themself or other account holding users.
- Users should be able to use public presets uploaded by other users
- Users should be able to view public creations uploaded by other users.
- Non-users can download a video
- Non-users can generate art
- Users can generate art
- Users should have control over parameters of art generation

System

- Art generation module should stop after art is generated
- Compatible on all major web browsers (Safari, Firefox, Edge, Chrome)
- Video generation requests should be handled in order
- Each art generation module should exist in its own file
- Logging should occur on every run to track defects and debug system problems
- Output videos should be saved to disk
- The system should not consume too much storage for memory of video files and logging output

Specifications

- Backend on Windows
- Videos should be 5 seconds long
- Videos should be created in an uncompressed format (mov)
- Videos should be received on front-end as a compressed format (mp4)

- Videos should be a certain pixel dimension (?)
- Users should be able to supply their own source images
- A queue should be used to track incoming art generation requests
- The maximum number of generation requests that should be tracked at once should be 64
- API should be used to supply source images
- API should be used to supply data for user selected ranges
- API should be used to affect art generation
- There should be a one second buffer added to the generation delay to preent corruption of generated files.
- The Art Generation Driver should store the instance id for art generatio in a JSON
- The Art Generation Driver should store the output path for art generation in a JSON
- The Art Generation Driver should store the art driver id for art generation in a JSON
- The Art Generation Driver should store all required parameters for art generation in a JSON
- Touch Designer should read from a JSON file containing all relevant parameters upon start up
- The Art Generation Driver Subsystem should launch Touch Designer from Python
- The Art Generation Driver Subsystem should not launch another Touch Designer instance until the previous has finished
- All objects related to start up and tear down in TD must be initialized at start
- Based off of module ID, all objects related to art generation must be initialized at start
- User sliders should be in ranges based off API data
- User sliders selections should be sent to the back end in range 0 to 99
- User sliders selections should be transmitted as an integer
- User sliders data should be received in the backend as range 0 to 99, and converted to an appropriate range based off of the TD Module being used
- Color choices by user should be sent as a hexademical representation (0xRRGGBB)
- Color choices should be converted from hexademical to RGB (0->255) representation.
- The front end should always send 6 slider values to the backend
- Slider values that are not populated on the front end should be sent as None to the backend
- The front end should send a flag bit set to one associated with color data to ensure data is properly handled on backend
- The front end should send a flag bit set to zero associated with normal slider data to ensure data is properly handled on backend
- The backend should check incoming slider data's flag bit to determine how to handle the input.
- Users should be able to select between different models of art generation
- Selected art generation models should be sent to the backend as a 1-indexed integer
- Unit Testing should cover generation input and output for all modules
- The backend should automatically clean up logging files when running
- The backend should automatically clean up generated art files when running
- The ArtGenerationId should be a unique identifier
- The ArtGenerationId should track the current request of art generation
- The ArtGenerationId should be used to label relevant movie files
- The ArtGenerationId should be used to send data to the front end

- If TouchDesigner is unable to initialize properly, it should quit immediately and notify the subsystem.

DB / API reqs + specs
Requirements
Art is generated using relevant social and environmental data.
User is able to switch between database for different generation outputs.
The user can save their desired art as an image or video.
The user can view and revisit their saved art
The user can manipulate sliders that change the art generation output

Specifications
The Backend Command Interface provides a HTTP GET endpoint to the front-end which retrieves environmental data from 6 different APIs.
The Backend Command Interface provides an HTTP POST endpoint to the front-end which, given a users identification key, stores the user specified image/video to a MongoDB collection.
The Backend Command Interface provides an HTTP GET endpoint to the front-end which, given a users identification key, retrieves and outputs a list of public urls representing users saved art.
User selected slider values are mapped into appropriate ranges for various qualitative model fields including speed, texture, and size.
Saved videos are trimmed to 5 seconds and stored as mp4 files in a google cloud platform bucket saved images are saved in a GCP bucket as jpg objects.
API data is parsed for the max and minimum range of each respective datasets average measurement field.
a thumbnail is generated for each saved user video and stored alongside the generated art, to be displayed on users profile page.