ARTISTIC STYLIZER PLATFORM: PASTYLE

CMPT 475N-200

PROF. PABLO RIVAS
Nick Klacik, Kevin Kleinschmidt, Brandon Litwin, Nicole Padrazo, & Ian Sniffen

Table of Contents:

1. Project Summary:

Our Artistic Stylizer Platform, Pastyle, is an application that allows users to upload and enhance their images through style transfer. Users can upload images, select a style to be applied to their image, and then have the style transferred. The main components of this project are the front-end web platform for user navigation, a machine learning API that performs the style transfer, and a database that holds user and photo information. Once images are styled, users have the options to pay for image watermark removal, to order their image printed on canvas, or share the image to social media. We use a Stripe API integration to handle payment processing for watermark removal. For the purposes of this project, Stripe is set up in a sandbox environment and therefore will not accept any actual payments. When you choose to have your content printed, a pop up store from CanvasPop will appear and allow you to pay for and select a canvas size for your content to be printed on. This CanvasPop integration is live, works, and has had successful orders placed from our website. In regards to system administration, we do have users that have administrative access, which gives the user the ability to see logs of uploads, processing time, and allows them to promote other users to have admin privileges. Our project was developed primarily using PHP, HTML, CSS, and minimal amounts of JavaScript. The code we developed was implemented on a server running Ubuntu 18.04, in which we use Apache Web Server for web hosting and Postgres for our database.

2. Software Project Analysis:

User Requirements:

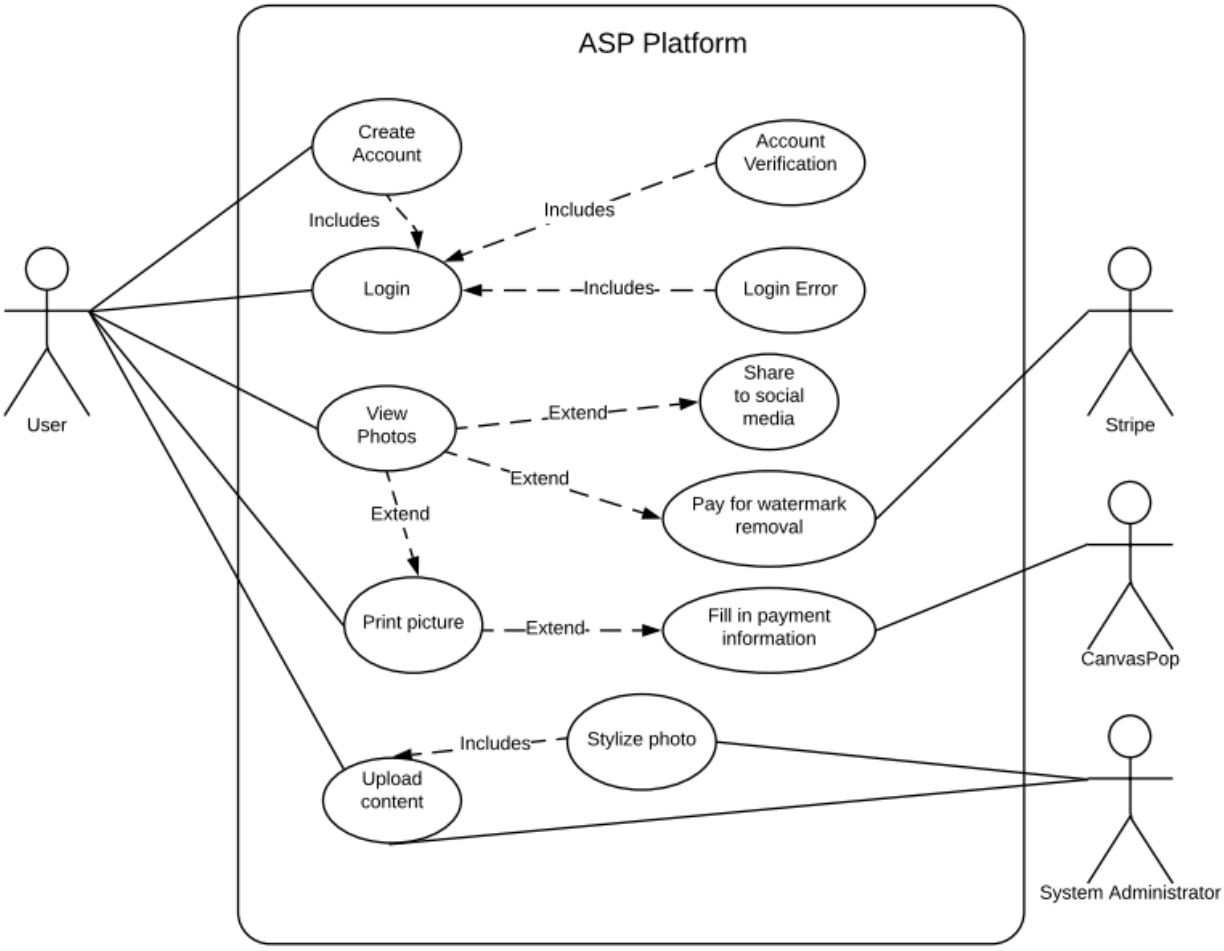
Functional Requirements:

- Stable web platform in which users can:
  - Create an account
  - Upload images for style transfer
  - Purchase a watermark-free version of their images
  - Order images printed on canvas
  - Share images to social media
- Machine learning API for image style transfer
  - Takes user uploads and transfer artistic styles onto the user content
- Data store for user credentials and photos
  - Users can authenticate to the site
  - Users can view images attached to their account
- System Logs
  - System administrators receive logs of site activity

Non-Functional Requirements:

- Capacity
  - The system should be able to handle 100 users
- Security
  - No unauthenticated users should be able to see data through the site
  - Passwords should be hashed upon creation
- Design
  - Should follow good UI design practices

Use Case Diagram:



Our use case diagram (UML) includes the four main actors in our system. The primary actor is the users, who interact with the front-end web platform. Users start by creating an account and logging into their account. Extensions of these use cases are account verification and login errors. When a user creates an account, they must meet the system requirements by entering a first and last name, email address, and a password consisting of at least one uppercase letter, a number, a special character, and a minimum of eight characters. When a user attempts to log in, our application runs a check of the database to make sure the account exists, otherwise the user receives an error and must attempt login again. Once a user is logged into their account, they have the ability to view images that they have uploaded and styled previously or upload a new photo. From the View Page, the user can see the photos they have stylized and have the options to buy the watermark free version of the image, share the image on social media, or have the image printed on canvas. On the Upload page, the user uploads a photo to the system and selects one of the six default styles to be applied to the image. As mentioned in the project summary, Stripe is the payment processing service API that Pastyle uses to handle payments for images to have watermarks removed. CanvasPop is another API used by Pastyle that handles the canvas printing services.

Use Case Name: Login |ID: 1 | Importance Level: High

Primary Actor: User |Use Case Type: Overview, essential

Stakeholders and Interests:
User – selects the option to create a new user account
System Administrator – receives logs of created user accounts

Brief Description: The user will create an account, filling in a name, email, and password. All passwords are hashed upon creation.

Trigger: Selecting the “Create Account” option on the homepage

Relationships:
Association
Include: User login follows creating an account
Extend:
Generalization:

Normal Flow of Events:

1. Select “New User? Register here.”
2. Fill in desired credentials
3. Account created
4. Advance to login

Alternate/Exceptional Flows:

1a. User credentials do not meet the system requirements, user credentials are rejected

Use Case Name: Login |ID: 2 | Importance Level: High

Primary Actor: User |Use Case Type: Overview, essential

Stakeholders and Interests:
User – selects the option to login into their user account
System Administrator –

Brief Description: The user will enter their username and password to login to their user account

Trigger: Selecting the “Login” option on the homepage

Relationships:
Association
Include:
Extend: Account verification and login failure
Generalization:

Normal Flow of Events:

1. Select “Login”
2. Enter the username and password
3. Verify authentication
4. Login successfully

Alternate/Exceptional Flows:

1a. Authentication fails, login fails

Use Case Name: Upload content |ID: 3 | Importance Level: High

Primary Actor: User |Use Case Type: Overview, essential

Stakeholders and Interests:
User – selects the photo to upload to the application
System Administrator – receives logs of photo upload information

Brief Description: The user uploads a content or style photo onto their user account

Trigger: Selecting the "Login" option on the homepage

Relationships:
Association
Include: Stylize content
Extend:
Generalization:

Normal Flow of Events:

1. User selects "Upload photo"
2. Content is uploaded
3. Select photo from device
4. Select style to be applied
5. User selects "Stylize"
6. Photo is stylized

Alternate/Exceptional Flows:

1a. File type is not accepted, upload fails
1b. File size is not accepted, upload fails

Use Case Name: View Photos |ID: 4 | Importance Level: High

Primary Actor: User |Use Case Type: Overview, essential

Stakeholders and Interests:
User – views photos, selects the photo to view enlarged

Brief Description: The user views photos they have stylized, with the option to view images enlarged

Trigger: Selecting the "View" option in the menu bar

Relationships:
Association
Include:
Extend: Share to social media, pay for watermark removal, or printing picture on canvas
Generalization:

Normal Flow of Events:

1. User selects "View" in menu bar
2. View enlarged image
3. Click through bottom pane to change enlarged image

Alternate/Exceptional Flows:

1a. No photos exist on the user's account

Use Case Name: Print picture |ID: 5 | Importance Level: High

Primary Actor: User |Use Case Type: Overview, essential

Stakeholders and Interests:
User – selects photo to print and fills in payment information
CanvasPop – handles user payment information and printing

Brief Description: The user fill in payment and shipping information and receive their photo printed on canvas

Trigger: Selecting the "Print Image" option on the View page

Relationships:
Association
Include:
Extend: Fill in CanvasPop payment and shipping information
Generalization:

Normal Flow of Events:

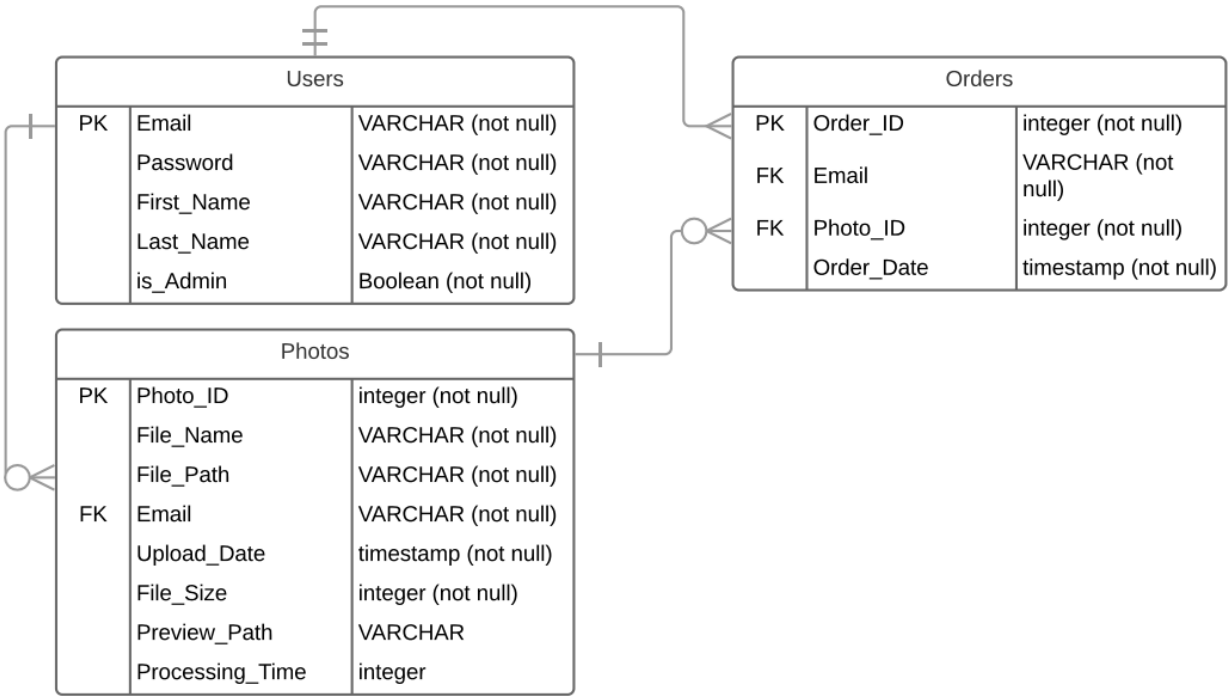
1. Select "Print Image"
2. Choose canvas type, size, frame, and quantity
3. Fill in shipping information and method
4. Fill in payment method
5. Transaction completed

Alternate/Exceptional Flows:

1a. Image is too small for CanvasPop to print

3. Software Project Design:

Entity-Relationship Diagram:



The database design that our team has put together made up of four tables to hold user account information, photo details, logs, and orders. The Users table includes all user information including email, password, name, and administrator status. The user email address serves as the primary key for this table and is referenced as a foreign key in the Photos and Orders tables. Other than the is_Admin attribute, all of the attributes in the Users table are variable character data types. User passwords will be hashed upon creation of the password for security purposes. The is_Admin attribute is a boolean value of either true or false, which determines if a user will be given administrator privileges. All users that have an is_Admin value of true will receive administrator privileges. None of the attributes in the Users table can have a null value.

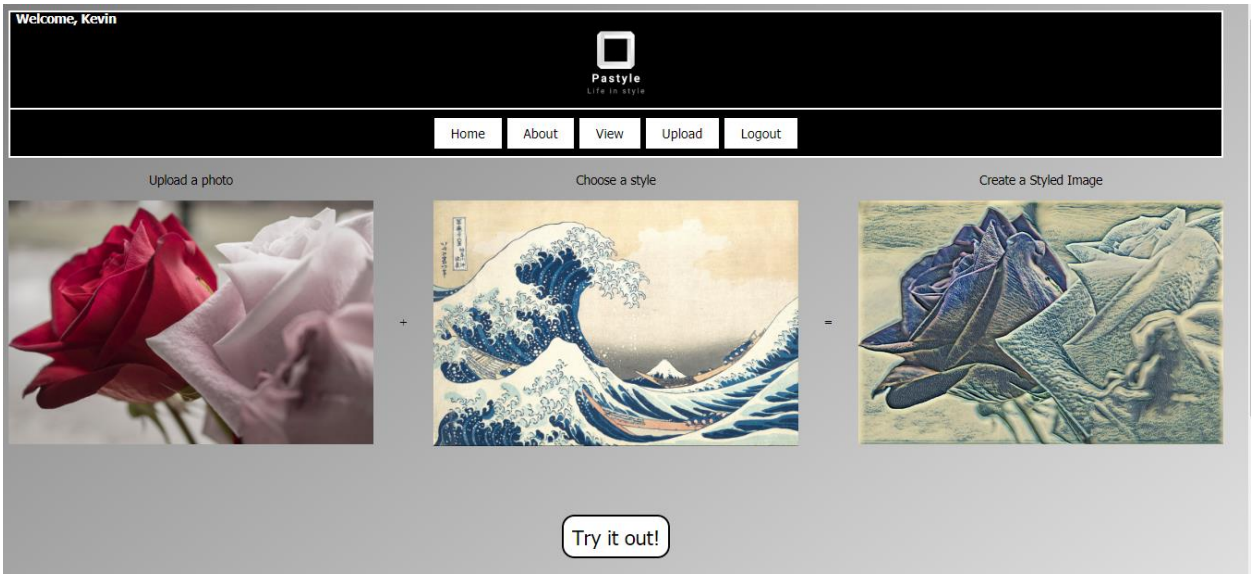
The Photos table is used to store information on each content photo that is uploaded onto the system. Photos will not be stored on the database, but instead will be stored on the production server. After discussion we felt that storing the photos on the database may bog the system down and cause the database to be slow. When a user uploads a photo, each photo will be given a unique Photo_ID. The File_Name attribute will insert the name of the image uploaded by the user and add the Photo_ID to the end of the File_Name. Therefore, users can upload the same file multiple times without receiving an error. The File_Path attribute that stores the file path where the watermark free photo is located on the server. Similar to File_Path, the Preview_Path attribute will hold the file path to the watermarked version of the image. Until a user has paid for their photos, they will see the watermarked versions of their photos in the view page. The photos table will also hold the photo's upload date, file size, and the email address associated with the user who uploaded the photo. None of the attributes in the Photos table can have a null value other than the Preview_Path and Processing_Time. The Processing_Time attribute will be used to hold the time it takes for an image to be stylized. This information will be helpful for system administrators, as it is displayed on the Administrator page. Once a user has paid for their photo, the watermarked version will be deleted, and therefore the Preview_Path will be deleted and have a value of null. Email is a foreign key referencing the Email in the Users table. The relationship between the Users and Photos table is a one to zero or many relationship. A user can upload zero or many photos and a photo can but uploaded by one user.

The Orders table stores information from payments for watermark removal. This table does not store data from the print service API, CanvasPop. Each order that is placed will receive an Order_ID that is unique to that order, making Order_ID the primary key of this table. The order date and price will also be stored in the Orders table. User_ID and Photo_ID are foreign

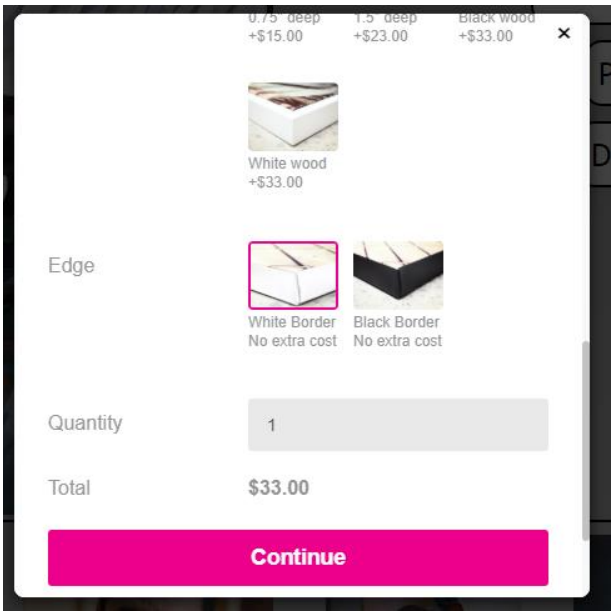
keys in the Orders table that reference the Users and Photos tables. We had considered making a candidate key for the Orders table using the User_ID and Photo_ID, but ruled it out due to the fact that a user could order an image more than once, so an Order_ID is needed to specify each unique order. The relationship between the Users and Orders table is a one and only one to many relationship while Photos and Orders have a one to zero or many relationship. A user can place many orders but an order can be made by one and only one user. A photo can be ordered zero or many times while an order can include one photo. The system will have six default styles that the users will be able to choose from to use on their content images, but we will not be storing any information regarding the styles in the database.

User Interface/Navigation:

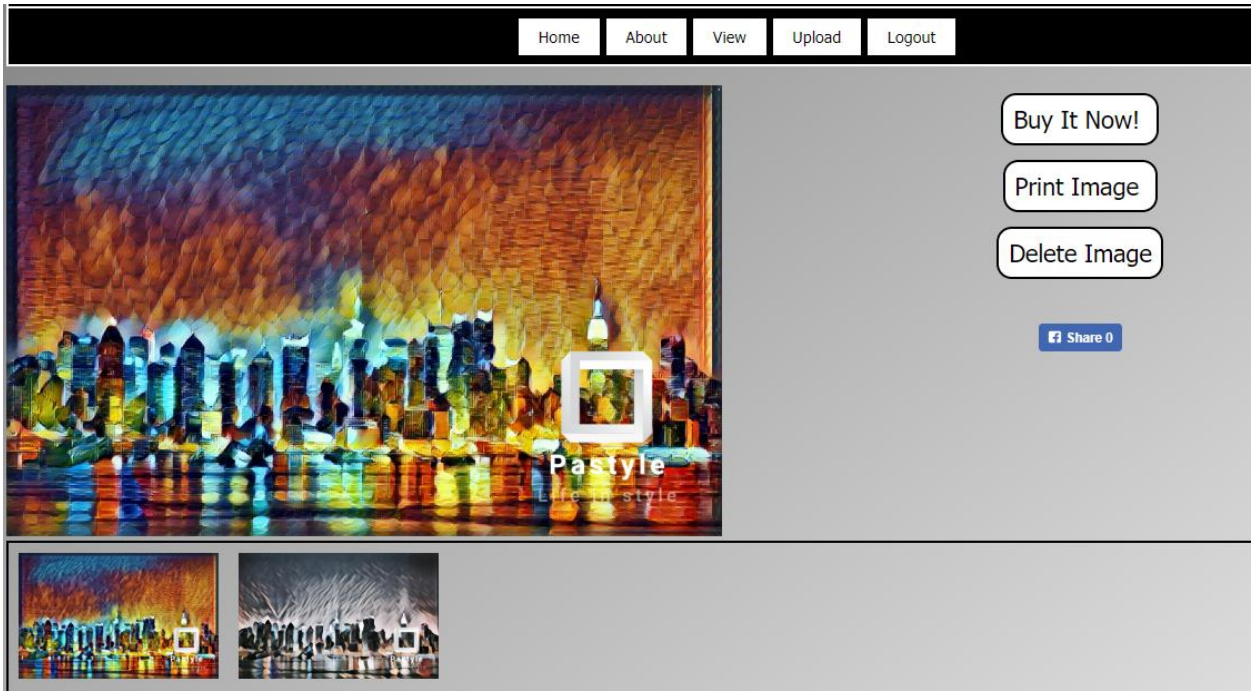
When designing our user interface, we tried to keep the design simple and to the point. We did not want our users to have trouble navigating to their desired page. Our goal was to follow the Laws of Usability developed by Steve Krug. The UI design we have put together uses text very minimally, but is clear enough for the user to understand how to navigate and use our system. Navigating around a website should be unambiguous. We tried to avoid clutter on each page, as it may confuse users and make it difficult to navigate the site. On our home page, we display the basic directions on how to use our site. We thought it would be best to include this on the homepage so that the user knows immediately how to use the main feature of the site. Below, you can see the direction we give users on the home page.



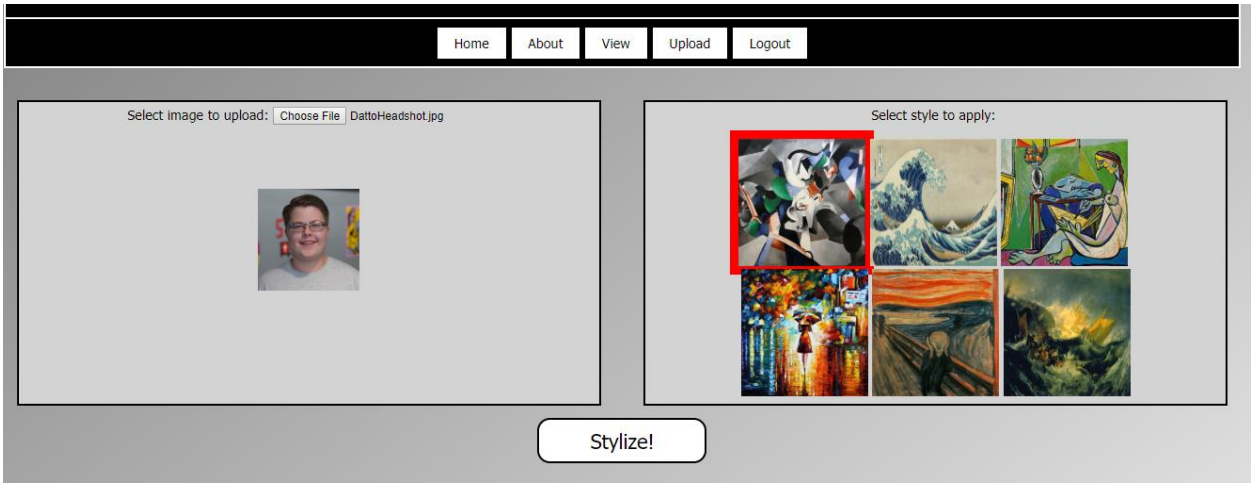
You can navigate around our website and not have to think about what each button does, because our design is clear and to the point. For example, after login, a user can navigate from the homepage, upload a photo, and have the photo stylized in five clicks. After reaching the homepage, the user has the option to view the About, View, and Upload pages. Otherwise the user can log out. If the user has administrative privileges, the navigation bar will also include a button for the Admin page. The About page gives a brief description of the project. On the View Page, users can view their past uploads, pay for watermark removal, or purchase their photos on printed canvas via CanvasPop. To pay for the Pastyle watermark to be removed from a photo, the user can select “Buy it Now” and they will be redirected to our instance of Stripe. Users then fill in payment information for the transaction to be completed. When using the CanvasPop feature, a pop-up store will appear and guide the user through the process of selecting a canvas type and size and filling in payment and shipping information. Below you can see an example of the pop-up store.



Users can click through all of their stylized images on the view page, with images uploaded previously showing up at the bottom of the page. The options a user has on the View page are presented very simply, with four buttons that state the action of each button. Below is a screenshot of the View page.

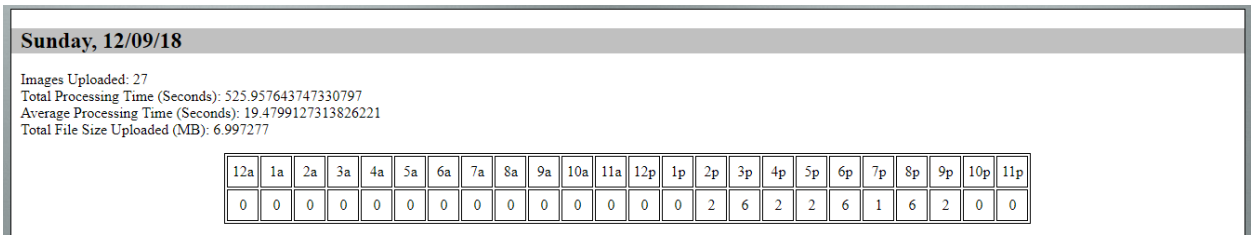


Whichever image is selected will appear enlarged at the top of the page. Users also have the option to share their images to Facebook. If chosen, a pop-up window for Facebook will appear prompting the user for login or direct them to share their image in a post. The Upload page is where the user can actually apply a style transfer to their own photos. With the simple directions for the user to upload a photo and then select a style to be applied, this page is straight forward and can be navigated without having to think about that the actions are. Once the user hits "Stylize", the photo the user uploaded will have the style transferred to it using our machine learning API. Below you can see that when a user uploads a photo, they are presented with a preview of the image that they uploaded. The style selected is highlighted with a red border so the user knows which one they've selected without confusion. In our design, we tried to make the demonstration shown on the homepage represent what the process would look like on the Upload page.



This process should take no longer than thirty seconds. When the style transfer is complete, the user will be redirected to the View page, where their photo will show up enlarged. The last page of our site is the Admin page. Only users with the Is_Admin attribute set to true will be able to access this page. On the Admin page, users can view the system logs from the past seven days.

We provide the number of images uploaded, total processing time, average processing time, and total amount of data uploaded. In addition, we created a table that shows the user the number of uploads by the hour. This will be beneficial to system administrators looking to find when there is the most activity on the site. Below, you can see an example of what an entry for one day of the system logs would look like on the Admin page.



Sunday, 12/09/18

Images Uploaded: 27
Total Processing Time (Seconds): 525.957643747330797
Average Processing Time (Seconds): 19.4799127313826221
Total File Size Uploaded (MB): 6.997277

12a	1a	2a	3a	4a	5a	6a	7a	8a	9a	10a	11a	12p	1p	2p	3p	4p	5p	6p	7p	8p	9p	10p	11p
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	6	2	2	6	1	6	2	0	0

4. Infrastructure Specifications:

1. Server Platform
    - LAPP stack (Linux Apache Postgres PHP)
        - o Production Server (Photos stored locally)
        - o Test/backup server (Photos stored locally)

1.1. Physical system requirements

1.1.1. Storage capacity
- 100mb per user.

1.1.2. Speed requirements / response time parameters
- Stylizer should take a maximum of 20 seconds.

1.1.3. Scalability plans
- More storage per user

1.2. Virtual system requirements

1.2.1. OS to be supported
- Linux (Ubuntu)

1.2.2. Number of images expected
- 1 (Ubuntu)

1.3. Connectivity

1.3.1. Network considerations
- Access through browser

1.3.2. Interconnection to what other systems

- Fast Style Transfer in Tensorflow (picture stylizing API)
- CanvasPop (printing service)
- Stripe (payment processing service)

2. Reliability

2.1. Service Level Agreements

2.1.1. Uptime requirements

- Will maintain uptime during 7am-11pm hour range, as this will be when most users will be using our system

2.1.2. Response time requirements

- Within 24 hour period of outage

3. Recoverability

3.1. Where are things backed up? How often?

- Separate VM server, backed up every two days

3.2. Access to backups?

- System Administrators

3.3. What data is transient and doesn't need to be stored long term?

- Photos that have not been paid for in 24 hours since upload to prevent storage issues

4. Security and Privacy

4.1. Database

4.1.1. Access controls by userid / roles

- System Administrators will receive an Is_Admin value of TRUE. If Is_Admin = TRUE, the user will be able to view logs

4.1.2. Update vs. Access

- System Administrators will have the ability to promote/demote other uses to system administrator status, altering the Is_Admin value of the user

4.2. Account information

4.2.1. User data

- All passwords are hashed upon creation. Access to the database is only provisioned for system administrators.

4.3. Admin access controls

4.3.1. Adding new users, deleting old

- Users can be deactivated after 90 days of inactivity

5. Maintenance

5.1. Planned down time requirements

5.1.1. Database maintenance

- To be completed during inactive site hours (12am-3am)

5.1.3. Times of year when IT does maintenance
- Every three months

5.1.4. Times of year when the systems are not available?
- During maintenance. Users will be notified in advance of maintenance times

5. Product Deployment Instructions:

Prerequisites:

- Ubuntu 18.04 server with the following installed:
  - Apache Webserver
  - Postgres
  - PHP
  - Python and pip

1. Clone our GitHub repository to the server
   Link: https://github.com/nickklacik/ASP-Capping2018

2. Copy the contents of WebPages directory to the Apache Webserver
   a. Copy contents to /var/www/html
   b. Contents of /Images in WebPages must be in /var/www/html (there are no references to a /Images subdirectory in our code)
   c. Change the Apache index to index.php
3. Build the database with the following CREATE TABLE statements
   a. Create a database called “postgres”
   b. USERS

```
CREATE TABLE USERS
(
EMAIL          VARCHAR (50)    PRIMARY KEY,
FIRST_NAME     VARCHAR (20)    NOT NULL,
LAST_NAME      VARCHAR (30)    NOT NULL,
PASSWORD       VARCHAR (255)   NOT NULL,
IS_ADMIN       BOOLEAN    NOT NULL,
);
```

   c. ORDERS

```
CREATE TABLE ORDERS
(
ORDER_ID       SERIAL          PRIMARY KEY,
EMAIL          VARCHAR (50)    NOT NULL,
PHOTO_ID       SERIAL          NOT NULL,
ORDER_DATE     TIMESTAMP       NOT NULL,
```

```
CONSTRAINT ORDERS_EMAIL_FKEY FOREIGN KEY (EMAIL)
    REFERENCES USERS (EMAIL) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT ORDERS_PHOTO_ID_FKEY FOREIGN KEY (PHOTO_ID)
    REFERENCES PHOTOS (PHOTO_ID) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

d. PHOTOS

```
CREATE TABLE PHOTOS
(
PHOTO_ID        SERIAL          PRIMARY KEY,
FILE_NAME       VARCHAR (30)    NOT NULL,
FILE_PATH       VARCHAR (30)    NOT NULL,
EMAIL           VARCHAR (50)    NOT NULL,
UPLOAD_DATE     TIMESTAMP       NOT NULL,
FILE_SIZE       numeric         NOT NULL,
PREVIEW_PATH    VARCHAR (75)    NOT NULL,
PROCESSING_TIME     INTEGER     NOT NULL,
CONSTRAINT PHOTOS_EMAIL_FKEY FOREIGN KEY (EMAIL)
    REFERENCES USERS (EMAIL) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

4. Edit Postgres configuration file
    a. Change authentication method

    ```
    vim /etc/postgresql/9.6/main/pg_hba.conf
    ```

    b. Change database administrative login by Unix domain socket to trust method as shown below

```
# Database administrative login by Unix domain socket
local   all             postgres                                trust

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

    c. Reload Postgres config files
5. Fast Style Transfer API/Tensorflow Integration
    a. To download the API:

```
git clone https://github.com/lengstrom/fast-style-transfer.git
```

b. Use pip to install Tensorflow and other modules needed for the API:
```
pip install tensorflow pillow scipy numpy ffmpeg moviepy imageio
```

c. FFMPEG needs to be downloaded as a plugin for imageio. To do this, run the following command:
```
python
```

d. In python, run the following, then exit:
```
import imageio imageio.plugins.ffmpeg.download()
```

e. To allow the API to be run from the webpage, the www-data user needs to have access to the file. To do this, the sudoers file must be edited. To do this, run the following command:
```
sudo visudo
```

f. Add the following line to the file and save.
```
"www-data ALL=(ALL) NOPASSWD: ALL"
```

6. Image Watermarking
    a. Install the gd library of PHP (supports 'imagecreatefromjpeg' and 'imagecreatefrompng' functions
    ```
    sudo apt-get install php5-gd
    ```
7. Stripe Integration
    a. Install php libraries: `curl and mbstring`
    ```
    sudo apt-get install php-curl
    sudo apt-get install php-mbstring
    ```
    b. Install Stripe with composer in the web root directory (ours is var/www/html/)
    ```
    sudo apt-get install composer
    composer require stripe/stripe-php
    ```
    c. Restart apache
    ```
    sudo service apache2 restart
    ```

Access to Our Production Server:

1. Download the SSH secret key
2. Open command line or Putty and run
   `ssh -i ml-stylzr.key developer@sp.marist.ai`
3. Password: `r3df0x@`

Test Cases:

1. On a browser navigate to sp.marist.ai
2. Log in using the following credentials:
    a. Username: admin@test.com

    b. Password: P@ssword1!
3. Go to the Upload Page
    a. Upload a photo
    b. Select a style
    c. Press “Stylize” (then redirected to View page)
4. Select “Remove Watermark” (redirected to Stripe)
    a. Enter in first and last name, email
    b. Enter 42424242424242… for all credit card information
    c. Return to view page
5. Return to Upload Page and repeat step 3
6. At View Page, select “Print Image”
    a. Go through CanvasPop ordering process
    b. Close CanvasPop
7. Delete current photo by selecting “Delete Image”
    a. First image that was uploaded should appear
8. Select “Share”
    a. Will be redirected to Facebook pop-up
9. On navigation bar, select “Admin”
    a. View logs
10. Logout

6. Cost Analysis

Pastyle was created by a group of five team members, including one IT project manager, IT consultant, and three computer programmers. The project ran for approximately fifteen weeks over the course of the semester, allowing time for gathering requirements, development, and testing. Since the concept is relatively simple, the cost of hosting the website itself is relatively low. However, the cost of the whole development process can be calculated using some of the following assumptions:

- the average IT project manager makes approximately $85,000 annually ($1,635 weekly/$40 an hour)
- the average IT consultant makes approximately $$75,000 annually ($1,445 weekly/$36 an hour)
- the average computer programmer makes approximately $69,000 annually ($1,325 weekly/$33 an hour)
- each member worked approximately three hours per week during the fifteen week process

Using the assumptions above, the total salaried cost of development for the entire project is:

IT Project Manager	$1,800
IT Consultant	$1,620
Computer Programmer (3)	$4,455
Total	**$7,875**

Additionally, operating under the assumption that access to a development environment was already available, the cost of a domain name and hosting the site on a shared server can be relatively low, especially for a small project of this size. For the unique domain, the cost is normally around $15, with the cost of shared hosting ranging anywhere from $5 to $100 per month. Additionally, assuming the site would acquire a small amount of maintenance, if a programmer needs to work on the site for an hour twice a year, it would cost about $35 per hour. For a site of this nature, the cost of hosting the domain as well as the server for about a year after production would be as follows:

Domain	$15
Maintenance	$70
Server	$360
Total	**$445**

Overall, using all of the assumptions made above, the entire cost of the project is as follows:

Salaries	
IT Project Manager	$1,800
IT Consultant	$1,620
Computer Programmer (3)	$4,455
Total	**$7,875**
Hardware	
Domain	$15
Maintenance	$70
Server	$360
Total	**$445**
Total Project Cost	**$8,320**

Due to the current setup of the website, it is currently just a platform to create stylized images that connects users with a printing service. Right now, the only profit that can be made is from

the $0.50 charge for removing the watermark from the image, creating an extremely miniscule ROI. However, if in the future, the site decided to upcharge a user 15% of their $35 order, the profit would be around $5 per order, making the whole order $40 for the smallest available canvas. If just 50 orders are placed per month, that is $3,000 profit per year, creating an ROI of 36%. Additionally, there is always the potential to charge companies to place advertisements on the site, charging per impression.

7. Project Plan

Capping 2018 Project Plan

Group 3: Nick Klacik, Kevin Kleinschmidt, Brandon Litwin, Nicole Padrazo, Ian Sniffen

Week	Date	Milestone	Tasks	Status	Responsible	Notes
1	8/29	Class 1	Determine team roles; select project; determine project goals & requirements; define ways of communication	Completed	All	Ian, Brandon, Nick - CS; Nicole - IT; Kevin - IS
2	9/4	Team Meeting 1	Create GitHub repo; begin UML and project plan; gather requirements	Completed	All	Planned to have ER diagram draft completed by 9/12
2	9/5	Class 2	Continue gathering requirements; ask prof. questions; work on UML; begin IT setup; finalize dev. roles & approach	Completed	All	Print service API determined: Pwinty
	9/7	Team Meeting 2	Determine weekly meeting time; work on HW 1 &2; review discussions from previous class;	Completed	All	HW to be completed, VM environment to be set up, Ian & Kevin do ER diagram, Brandon & Nick figure out APIs
3	9/12	Class 3	DB Design w/Alan, IS w/Algozzine	Completed	All	
		HW 1 Due	Completed UML diagram and project plan, user and client requirements gathered	Completed	All	Pushed to GitHub repo
		Decide which API we use	Determine which API we are going to use for the actual image stylizing (w/approval from prof)	Completed	CS	
		Team Meeting 3	Review ERD before professor review; determine which UI wireframes we are doing for 9/26; go over IT requirements	Completed	All	Need to complete ERD documentation; set up LAPP stack; draft UI wireframes
			Create ERD, meet w/DB professor	Completed	IS/CS	Ian and Kevin meet w/Schwartz on 9/18
4	9/19	Class 4	Project Management and Design Tooling w/Algozzine	Completed	All	
		HW 2 Due	Completed ER diagram with supporting documentation; List of IT requirements	Completed	IS/IT	
		IT Environment	Have IT environment up and running so we can begin coding	Completed	IT/IS	
			Have backup schedule/plan determined	Completed	IT/IS	
		Team Meeting 4	Discussed schedule going forward, assigned tasks to the team	Completed	All	
5	9/26	Class 5	Group work in class	Completed	All	
		HW 3 Due	Have a mockup of our UI layout and design, need five wireframes	Completed	IS/CS	Home, user photos, login, user account page, stylize page

		LAPP Stack	Have LAPP stack running, we need Postgres set up so we can create tables	Completed	IT	
		API	Research API and Pwinty	Completed	Ian	
	9/28	Team Meeting 5	Pwinty discussion; UI build overview; review comments made on ERD	Completed	All	
		DB Tables	Have table creation queries written (to be implemented when Postgres is set up	Completed	Kevin	
6	10/3	Class 6	Group work in class	Completed	All	
		Database Due	Have a functional DB system prototype, including ER diagram and documentation, can query data live	Completed	IT/IS	Need to SSH into server, query data
		VM	Install Torch on the server, set up GitHub on the server, determine backup method (possibly another VM)	Completed	IT	
	10/5	UI	Have a homepage, login, style transfer page, admin page completed in HTML & CSS	Completed	CS	
7	10/10	Class 7	Group work in class	Completed	All	
	10/16	Team meeting	Finalize demo prototype; prepare for demo	Completed	All	
8	10/17	Class 8	Group work in class	Completed	All	
		Have something running	Have something that runs, basic UI, functionality works properly	Completed	CS	
		CanvasPop	CanvasPop integration	Completed	Ian	
		UI	New CSS integration	Completed	Nicole	
9	10/24	Class 9	Group work in class	Completed	All	

			Major application functionality running, bugs and security measures should be only remaining components	Completed	All	
		Admin Page	Begin admin page	Completed	IS	
		Style API	Test out new style API on VM w/4GB RAM	Completed	CS/IT	
10	10/31	Class 10	Group work in class	Completed	All	
		Security	Password validation, password requirements	Completed	CS	
		Demo prep	Prepare demo for week 11	Completed	All	
		Style API	Implement new style API	Completed	CS	
		Stripe	Stripe integration	Completed	CS	
		Watermark	Set up photo watermarking	Completed	CS	
11	11/7	Class 11	Group work in class	Completed	All	
		Draft Presentation Due	Complete project prototype first-pass; demo ready	Completed	All	Completed demo, will work on having site ready for an order to be made by Pablo on 11/16
		UI	Final UI/UX Design due	Completed	IS/CS	
		Public IP	Get public server running, so we can actually make orders	Completed	All	
		Upload Page	Fix Upload Image page	Completed	CS	
		View Page	Delete photo functionality	Completed	CS	
		Social Media	Add social media sharing feature	Completed	CS	
12	11/14	Class 12	No Class	Completed	All	

		First Order!	Have Pablo make first order by 11/16	Completed	All	Order made by Pablo and Ian successfully
	11/21	Have final deliverable complete	99% complete, running product, with only remaining work being bug fixes	Completed	All	
13	11/28	Class 13	Group work in class	Completed	All	
		Contingency period	Deliverable should be complete, use this time to fix bugs and prepare presentation	Completed	All	Clean up code, review commenting
		Draft Documentation	Draft of final paper/documentation	Completed	IS	Make adjustments to UML, ERD, and design documentation
		Admin Page	Complete admin page & integrate	Completed	CS	
14	12/5	Class 14	No Class	Completed	All	
		Paper Due	Final documentation due	Completed	IS	
		Code Review	Clean up code, review commenting	Completed	All	
		Web Responsiveness	Figure out web responsiveness so the site looks good on mobile	Completed	CS	
		Team Meeting	Review entire site, make final adjustments, finalize paper and practice presentation/demo	Completed	All	
15	12/12	Class 15	Final presentation	Completed	All	

8. Ethics

During the design and development of this project, there were ethical considerations that came up. Our platform has no restrictions on the content of the images uploaded, which led us to think about copyright laws and inappropriate content. Due to the fact that our platform allows users to upload images and have them altered through the style transfer algorithm we implemented, the issue of image copyrights was brought up. If users want to, they can upload copyrighted images to Pastyle and have them stylized. Uploading inappropriate content such as sexually explicit or graphically violent images. After looking into these issues, we determined that laws such as Fair Use Laws and the First Amendment can protect our users. Similar to other media platforms such as YouTube, and Instagram, we could implement policies banning copyrighted or violent images. But that also brings up the question of whether it is fair of us to prohibit users from uploading certain images onto our platform.

The ethical implication that we thought about at the start of the project was copyright. Is it legal to upload a copyrighted image, alter it, and claim it as your own? Fair Use Laws can protect our users from these copyright implications. Fair Use Laws "promotes freedom of expression by permitting the unlicensed use of copyright-protected works in certain circumstances" [1]. When considering whether an action falls under Fair Use, the purpose and use, nature of the work, the amount, and effect of the particular case are reviewed. In the case of Pastyle, the purpose of the website was educational purposes, as the development of the site was for a class project. Currently, the team that developed Pastyle are not intending on using the site for commercial nature. The amount of images uploaded that are protected under copyright laws will most likely be low, as many users will be uploading and stylizing their own original photos. The value of these copyrighted works should not change as a result of users taking these copyrighted images and altering them with the style transfer. After considering all of these factors, it can be concluded that if a user were to upload and stylize a photo on our platform, it would fall under Fair Use due to the nature of the website and the effect on the value of the copyrighted image. If we were to change the nature of the use of Pastyle to be used for profit, we would most likely state that we are not liable for the images uploaded by users.

The other ethical consideration regarding images uploaded to Pastyle was the uploading of sexually explicit or graphically violent images. Without any restrictions, users can upload any image of their choice onto their account. If a user were to upload an explicit or violent image and we were to not remove it, could there be any implications? If the image was incriminating or deemed explicit there could be legal action taken against the user and the system administrators for not removing the photo and/or user from the site. One way to mitigate this issue would be for the blocking of all inappropriate content from the system. The issue to consider in prohibiting

users from uploading graphic content can be looked at as a First Amendment issue. Under current law, sexually explicit images such as pornography are protected under First Amendment rights. The factor that is reviewed in Freedom of Speech cases regarding sexually explicit content is obscenity. Obscenity and child pornography are the two types of pornography that are not protected under the First Amendment. Therefore, if a user were to upload images to our site that were considered obscene, the user would not be protected by Freedom of Speech.

To prevent any user from uploading any inappropriate images, a filtering system could be implemented on the site. The ethical consideration that must be made regarding a filtering system would be whether it is fair for the site to prohibit users from uploading any image of their choice. A filtering system would be beneficial from Pastyle because it would end any issues regarding inappropriate content completely. While this can be done is it fair to the users? If you look at this from a utilitarian perspective. The idea of utilitarianism is that actions are determined to be right or wrong depending on the effects of the actions on society. The action that creates the most pleasure or happiness would be considered morally right compared to the action that creates pain and unhappiness. Therefore, implementing a filtering system for inappropriate content on the Pastyle platform would be considered morally right. Prohibiting sexual and graphic content on the site would cause more pleasure and happiness for the users and system administrators as a whole. Users would be more satisfied with the Pastyle service and system administrators and developers would not face any legal issues regarding inappropriate content being uploaded to the platform. While it may be unlikely that users of the platform would be uploading inappropriate content, it was an ethical consideration that was looked at during the development process of the project.

When looking at the ethical implications of Pastyle, it is very unlikely that we come across these issues, but it is important that these issues have been brought up and a plan on how to handle them has been determined. As a team, Pastyle would not be liable for users who upload copyrighted images and if needed, we could implement a filtering system to prevent inappropriate content. Fair Use and First amendment rights can protect users the ethical issues mentioned, but stating that we are not liable for user actions would be beneficial. While it may not have been mentioned previously, the ethical issue of privacy should not be a problem with Pastyle. User passwords are hashed upon creation for security purposes. We do not share user data with any other parties. Copyright, inappropriate content, and privacy are the main ethical considerations that would need to be addressed if Pastyle were to be for profit and scaled up to handle more users. We believe we have found possible solutions for these ethical issues via liability and image filtering.

Works Cited:

[1] Copyright.gov, “More Information on Fair Use”, https://www.copyright.gov/fair-use/more-info.html, (posted Jul. 2018, accessed Dec. 10, 2018)

[2] D. Hudson, “Obscenity and Pornography”, https://www.mtsu.edu/first-amendment/article/1004/obscenity-and-pornography, (accessed Dec.10, 2018)

9. Appendices

L. Gatys, A. Ecker, M. Bethge, “A Neural Algorithm of Artistic Style”, https://arxiv.org/abs/1508.06576, (posted Aug. 25, 2015)

D. Ulyanov, A. Vedaldi, V. Lempitsky, “Instance Normalization: The Missing Ingredient for Fast Stylization”, https://arxiv.org/abs/1607.08022, (posted Jul. 27, 2016)

L. Engstrom, “Fast Style Transfer in TensorFlow”, https://github.com/lengstrom/fast-style-transfer, (posted Oct. 31, 2016)