Technical summary

Photo Colorization Application

June 24, 2020

Kristofer Bitney

WGU C964

Table of Contents

[Problem Statement 2](#_Toc43899152)

[Application Users 2](#_Toc43899153)

[Application Description 2](#_Toc43899154)

[Neural Network Architecture 3](#_Toc43899155)

[Neural Network Training 4](#_Toc43899156)

[Data Definition 4](#_Toc43899157)

[Software Engineering Methodology 5](#_Toc43899158)

[Requirements Analysis 5](#_Toc43899159)

[Design 6](#_Toc43899160)

[Implementation 6](#_Toc43899161)

[Verification 6](#_Toc43899162)

[Deployment 7](#_Toc43899163)

[Maintenance 7](#_Toc43899164)

[Project Timeline 7](#_Toc43899165)

[Resource and Funding Breakdown 7](#_Toc43899166)

[Table 1: Project Schedule 8](#_Toc43899167)

[Table 2: Project Budget and Outcomes 8](#_Toc43899168)

[Conclusion 9](#_Toc43899169)

[Appendix A: Requirements 10](#_Toc43899170)

[Appendix B: Photo Colorization User Interface Design 11](#_Toc43899171)

[Appendix C: Client-Server Relationship Design 15](#_Toc43899172)

[Appendix D: Usability Test Questionnaire 16](#_Toc43899173)

# Problem Statement

Photo colorization is the process of adding color to a grayscale photo to replicate what the photo would have looked like had it been taken with a color camera. Automated photo colorization is a relatively new process that has become more widely available during the last decade as machine learning technologies have improved. Before “deep learning” algorithms, photo colorization was a manual process. Today it is possible to colorize photos quickly, cheaply, and with a high level of realism. I propose we develop and deploy a photo colorization application to augment our existing photo restoration services.

# Application Users

Our photo restoration products attract a variety of customers, from families with cherished memories to photography hobbyists. A significant demographic in our customer base are the aging and elderly population who seek to maintain the quality of old photos. Family photo albums can have personal and emotional value to our customers due to the role of albums as records from a time before digital photography and mobile phone cameras were ubiquitous.

Color photography became accessible to the average American household in the early 1960s. Earlier photos were grayscale, also known as “black and white”. The aging segment of our customer base often own grayscale photos. This is a market opportunity. We can bundle photo colorization with our existing service packages to increase package value with virtually no increase to marginal costs.

# Application Description

The application I propose uses a client-facing front-end web page to make photo colorization as easy as selecting a grayscale image file and clicking a button. Once the user selects an image and clicks the button, the web page will send the selected image file to a back-end HTTP web server to handle the request. The back-end web server processes the image by inference through a neural network, which will be trained to colorize the image, and returns the colorized image file to the client.

The front-end web page will be written in HTML5/CSS3. The page will be responsive, allowing users to access the application from desktop and mobile devices with little inconvenience. The client-side script will be written in JavaScript compliant with the ECMAScript 6 standard. The script will not require third-party JavaScript libraries or frameworks.

The back-end web server will be written in Python using Starlette, a lightweight Asynchronous Server Gateway Interface (ASGI) framework. ASGI replaces the Web Gateway Server Interface (WSGI) standard that describes how Python web servers should communicate with front-end applications. The Starlette framework implements the ASGI standard, facilitating the use of coroutines and tasks for asynchronous web services.

I will train a neural network using the FastAI Python library. Built on top of the PyTorch deep learning framework, FastAI reduces boilerplate code and provides useful API’s to increase developer productivity.

Neural Network Architecture

The neural network will use a UNet architecture. In a UNet architecture, the input image is first processed by a series of neural network layers that are often based on a standard convolutional neural network architecture—the encoder. As the encoder processes the input, the channel dimension of the image—the number of sets of local weights, sometimes called Kalman filters—grows while the height and width dimensions shrink. The local weight matrices, which represent latent features, are then processed through a series of convolutional layers that progressively increase the height and width dimensions of the output while decreasing the size of the channel dimension—the decoder. Intuitively, the encoder deconstructs the input image to recognize patterns and the decoder uses the patterns to reconstruct an output image. In order to preserve information, cross connections are used to connect the encoder with the decoder at each major change in feature dimensions.[[1]](#footnote-1)

The UNet architecture will use an encoder based on the ResNet architecture. The major innovation of ResNet is the use of an identity transformation in convolutional neural networks, sometimes called a skip connection.[[2]](#footnote-2) A skip connection prevents overfitting by parameterizing the weighted sum of a network layer’s output with its input features; the output of such a layer influences the model only to the extent that it offers improvement over its input. In other words, skip connections are shrinkage operators that provide a form of regularization.[[3]](#footnote-3)

In ResNet, a skip connection is applied to each block of two convolutional layers, a ReLu activation layer, and a batch normalization layer. There is some consensus that the ReLu activation—which simply truncates negative values at zero—tends to outperform activations based on sigmoidal functions in most settings. ReLu activations also require less computation. It has become a best practice to use ReLu activation functions in neural networks.

A batch normalization layer standardizes the output of an activation layer to have zero mean and unit variance. Although the empirical benefits of batch normalization have been widely confirmed, the mechanism explaining its benefits are poorly understood. Batch normalization was originally introduced as a method to improve training speed and generalization by reducing “internal covariate shift”.[[4]](#footnote-4) However, there are competing explanations claiming better empirical support.[[5]](#footnote-5)

## Neural Network Training

I will train the neural network using a Generative Adversarial Network (GAN) framework. In a GAN framework, the primary neural network—referred to as the “generator”—is trained with a revised loss function based on an auxiliary neural network called the “critic”. The critic is a binary classification model trained to distinguish between generated (colorized) images and real color images. The revised loss function is a weighted sum of a traditional loss function—in this case mean squared error—and the loss of the critic.

At each training step, the generator’s parameters are updated based on the gradient of the revised loss function with respect to the parameters. The generator therefore learns to fool the critic. As the generator’s output improves, the critic becomes less capable of distinguishing between real color photos and colorized photos. Once the critic’s loss reaches a pre-specified threshold, the critic is re-trained so it can once again accurately distinguish between generated and real color photos. As this process repeats, the generator and critic compete and improve together.

I will use a variety of best-practice methods to improve model training, including:

* Pre-training the generator and critic
* Use of variable learning rates[[6]](#footnote-6)
* Use of weight decay, also known as L2 regularization
* Starting with small images, and increasing image size as training progresses
* Randomized data augmentation

## Data Definition

I will use of the Common Objects in Context (COCO) dataset, which contains about 330,000 photos of people, animals, objects, and environments.[[7]](#footnote-7) I will create a set of input images by converting COCO photos to grayscale and then use the original color versions as labels. To reduce training time, I will use only the 118,000 images in the COCO 2017 Train Images dataset. Future iterations of the model may benefit from use of the full COCO dataset. Images will be resized as needed to facilitate neural network training. The diversity of the COCO data—and its permissive licensing—make the data amply sufficient and appropriate for use in a photo colorization product.

The COCO dataset is distributed under a Creative Commons Attribution 4.0 License. Use of images must abide by the Flickr Terms of Use. We must cite the COCO dataset in our publications to indicate it was used. Copyrights of the images in the dataset are owned by a large number of individuals who have permitted inclusion of their images in the dataset, so the images themselves should not be sold in association with our photo colorization product. We will retain copyright of our neural network model and application regardless of the status of individual image intellectual property rights.

I will augment the COCO image data with randomized feature transformations during training. For example, input and target images will be subject to random changes in lighting, positional translations, and rotations. Such changes can improve training by effectively increasing the size and diversity of the available dataset. Image transformations will be applied lazily (i.e., as needed) and will therefore not be known a priori except as probabilistic expectations.

# Software Engineering Methodology

I will use the Waterfall software engineering methodology to develop the photo colorization application. The Waterfall model approaches project management in a traditional manner, linearly flowing through the steps of the Software Development Life Cycle:

1. Requirements Analysis
2. Design
3. Implementation (Development)
4. Verification (Testing)
5. Deployment
6. Maintenance

The Waterfall methodology is effective because it emphasizes completion of planning and design before development begins. The photo colorization application project team will include only one person, so strong planning and design are necessary to minimize errors that would otherwise be prevented by code reviews, pair programming, testing specialists, or a quick read by a second pair of eyes.

The Waterfall model requires that projects are well understood in advance of their design and implementation. Because the model formally calls for a linear project flow, “bugs” that enter the Requirements Analysis and Design phases can compound in complexity by the time Implementation is completed and Verification begins. In other words, returning from one step in the life cycle to fix bugs in prior steps is generally more costly in a Waterfall project than for projects using other software engineering models. When a project can be well planned and such bugs are minimized, the Waterfall model offers traditional project management benefits—including predictability and cost savings.

Iterative and incremental (Agile) methodologies generally exist to facilitate efficient teamwork and communication for projects with small teams of developers where uncertainty is expected. Without the need for teamwork or communication, Agile processes are generally not possible or not useful. On the other hand, the costs of allowing “feedback” in a Waterfall methodology—i.e. allowing time to revisit prior steps—are minimal for small projects where interpersonal coordination is not necessary. The greatest weakness of the Waterfall model—inflexibility—is obviated for a small project with only one developer.

## Requirements Analysis

Requirements are features the application must provide. The goal of the Requirements Analysis phase is to produce a list of application requirements that comprehensively describes the completed application. Requirements should be clear, unambiguous, consistent, prioritized, and verifiable.[[8]](#footnote-8) The requirements list is used as the basis for application design.

See *Appendix A: Requirements* for the full list of photo colorization application requirements, created using the FURPS method of requirements categorization.

## Design

The design process has two stages. High-level design provides an abstract overview of the application, while low-level design provides more specific implementation details. Because the scope of the photo colorization application is relatively limited, I combine high-level design and low-level design into one stage of design.

A wireframe design of the application’s graphical user interface can be found in *Appendix B: Photo Colorization User Interface Design*. The wireframe design details the layout of the web page to be created using HTML5, CSS3, and JavaScript.

The wireframe design has four pages. The application’s primary function—photo colorization—takes place on the first page. Users can select a photo to submit, submit it, and receive an enhanced version. The second page contains a “carousel” of examples with color, grayscale, and colorized versions of images. Users can select the 🡨 or 🡪 buttons to browse the examples. The third page displays charts and graphs related to the neural network and data. Users can click on navigation items to change the contents displayed and learn more about the machine learning methodology used in the project. The fourth page displays contact information for the company, including internal technical support information to help employees submit a support ticket if they need assistance.[[9]](#footnote-9)

The graphic in *Appendix C: Client-Server Relationship Design* describes the relationship between the user interface, the client-side script, and the server-side script. It also specifies the roles encapsulated by the client and the server. The client-side script manages the user interface components and sends HTTP requests to the server. The server hosts the web site, accepts HTTP requests containing image data, processes images using a trained neural network, and sends HTTP responses containing colorized versions of the images back to clients.

## Implementation

Implementation involves development of the neural network and photo colorization application, as well as other tasks that must be done to prepare the application for deployment. Implementation details will follow from the project requirements and design phases.

I will use Jupyter Notebooks and the Gradient platform to develop the neural network. I will use PyCharm Community Edition, a free development environment, to develop the client and server web application scripts.

## Verification

The verification phase of the Software Development Life Cycle is the phase where testing occurs. First, I will write use unit tests and integration tests to ensure the application code produces expected results. I will then conduct systems tests to ensure the application functions as expected, including an audit of the application to ensure that it meets the project requirements and design specifications.

Finally, to ensure the application appeals to end users, I will conduct usability tests with potential application users.[[10]](#footnote-10) I will administer a standardized questionnaire to ensure I get the feedback necessary to make the project successful. For a copy of the questionnaire, see *Appendix D: Usability Test Questionnaire*.

When the project has successfully passed all of the aforementioned tests, the application will be made available to company decision makers for validation. This is also called acceptance testing. The project is successfully validated when its primary stakeholders agree the application meets the goals determined before the project began and the requirements determined during the Requirements Analysis project phase.

## Deployment

I will deploy the application as a Docker container using the Render web server hosting service. The Render service integrates with GitHub, which will host the application’s code. Render provides tools for monitoring application availability and performance, including terminal logs and a memory profiler. Render includes free SSL certificates for secure HTTP communication.

## Maintenance

Application maintenance requirements will be minimal in the absence of new feature requests. As features are added and the scope of the project increases, maintenance costs may increase. The current proposal makes no assumptions about maintenance duration or costs.

# Project Timeline

The full software development life cycle will take approximately three months from Requirements Analysis through Deployment. The project schedule presented in *Table 1: Project Schedule* describes the project milestones with start dates, end dates, duration, dependencies, and human resource requirements. Note that duration estimates are in calendar days. The schedule implies a project work estimate of 52 FTE workdays, or 416 FTE hours. The schedule assumes the developer must learn at least some of the programming languages, libraries, and frameworks required to complete the development milestones. Application maintenance is assumed to be ongoing once the application is deployed, and is not counted in this duration estimate.

# Resource and Funding Breakdown

I recommend a small project budget that includes the following items:

* Salary & benefits of a single developer for 52 FTE work days ($264 per FTE)
* Use of GPU compute services for three months ($8 per month)
* Ongoing server hosting ($7 per month)

Salary costs assume a $30 hourly wage and benefits costs (e.g., health insurance subsidy) equal to 10% of salary. Other developer-related costs, such as use of a computer, office space, and office supplies are not included in this budget. Server hosting costs will be as low as $7 per month following deployment. However, as our photo colorization service grows in popularity, server hosting costs may increase to accommodate more internet traffic.

The total project budget is about $13,773. A budget breakdown by milestone item and deliverable is presented in *Table 2: Project Budget and Outcomes*. Where applicable, costs are distributed by FTE estimates. Costs are rounded to the nearest dollar.

## Table 1: Project Schedule

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Milestone | Start Date | End Date | Duration | Dependencies | Required FTE |
| A | Requirements | 4/1/2020 | 4/7/2020 | 7 |  | 0.5 |
| B | Design | 4/8/2020 | 4/14/2020 | 7 | A | 0.5 |
| C | Develop neural network | 4/15/2020 | 5/14/2020 | 30 | A, B | 1 |
| D | Develop server-side script | 5/15/2020 | 5/21/2020 | 7 | A, B | 1 |
| E | Develop client-side script | 5/22/2020 | 5/31/2020 | 11 | A, B | 1 |
| F | Develop web page | 6/1/2020 | 6/10/2020 | 10 | A, B | 1 |
| G | Testing | 6/11/2020 | 6/19/2020 | 9 | C-F | 0.5 |
| H | Deployment | 6/22/2020 | 6/26/2020 | 5 | G | 0.5 |
| I | Maintenance | 6/27/2020 | - | - | G | 0.1 |

## Table 2: Project Budget and Outcomes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Milestone | FTE Hours | Budget | Deliverables |
| A | Requirements | 20 | $662 | Requirements List |
| B | Design | 20 | $662 | Photo Colorization User Interface Design; Client-Server Relationship Design |
| C | Develop neural network | 176 | $5,827 | Jupyter notebook script; Trained neural network |
| D | Develop server-side script | 40 | $1,324 | Back-end server script |
| E | Develop client-side script | 48 | $1,589 | Front-end script |
| F | Develop web page | 64 | $2,119 | HTML/CSS files |
| G | Testing | 28 | $927 | Unit tests, Usability Test Questionnaire |
| H | Deployment | 20 | $662 | Photo Colorization Application |
| I | Maintenance | - |  | Continuing availability and performance |
|  | Total |  |  |  |

# Conclusion

Photo colorization services present a marketing opportunity for our organization. Development time and costs may be as high as three months and $14,000 for a single developer. However, ongoing maintenance costs will be minimal and the value the application adds to our photo restoration packages may be substantial. I recommend a demand analysis to estimate the potential revenues associated with the addition of this application to our product line.

# Appendix A: Requirements

**Functionality**

* The user can select a grayscale photo, submit the photo, and receive a colorized copy of it.
* If the user selects and submits a color photo, the application will convert it to grayscale and return a colorized copy to the user.
* The user can visually compare their submitted photo to the colorized version.
* The application presents users with self-contained instructions for use.
* Users can interact with the service from popular browsers.
* The service can be accessed and used on desktop and mobile devices.
* Photo colorization is performed by a neural network hosted on a server.

**Usability**

* Users can colorize their photos through a graphical user interface.
* Photo colorization functions are fully abstracted: users can use the service without understanding how it works.
* The full process of photo submission and receipt requires less than five clicks to complete (where photo selection is counted as one click).
* The application is so simple it can be used without instructions by at least 90% of users.

**Reliability**

* The application is available for use during business hours.
* The application is secure and does not expose confidential data.
* There are tools for monitoring application availability.

**Performance**

* The user should not need to download the neural network or any other large files.
* The user should receive the colorized copy of their photo within five seconds.
* Multiple users can use the application concurrently.
* There are tools for monitoring application performance.

**Supportability**

* The photo colorization neural network does not require additional tuning following deployment.
* The saved neural network underpinning the application’s photo colorization functionality can be swapped with an updated neural network file and the application can continue to function without further modification.
* The server hosting the photo colorization neural network can respond to photo colorization requests from arbitrary websites or applications—e.g., such as if the company chooses to integrate photo colorization functionality into other systems.

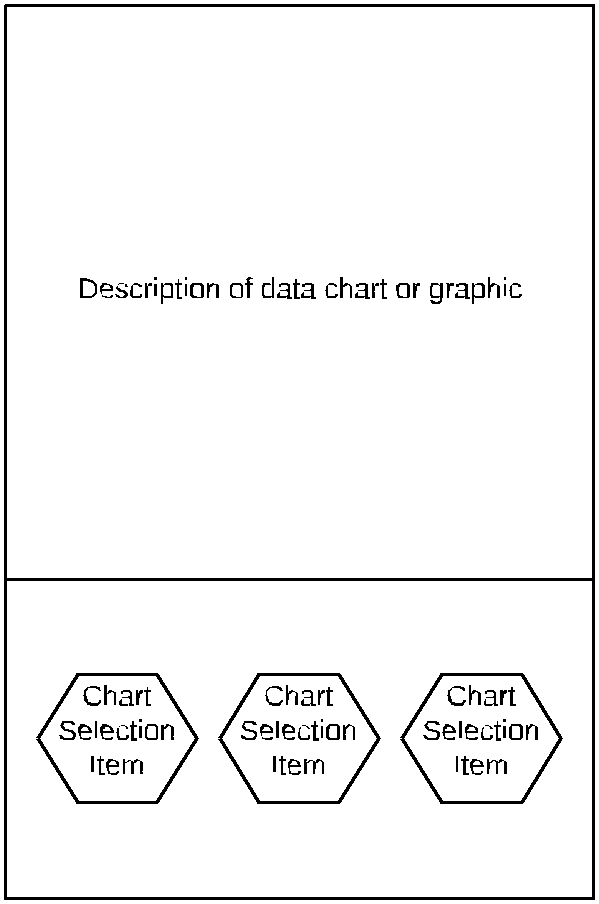
# Appendix B: Photo Colorization User Interface Design



1 of 4



2 of 4

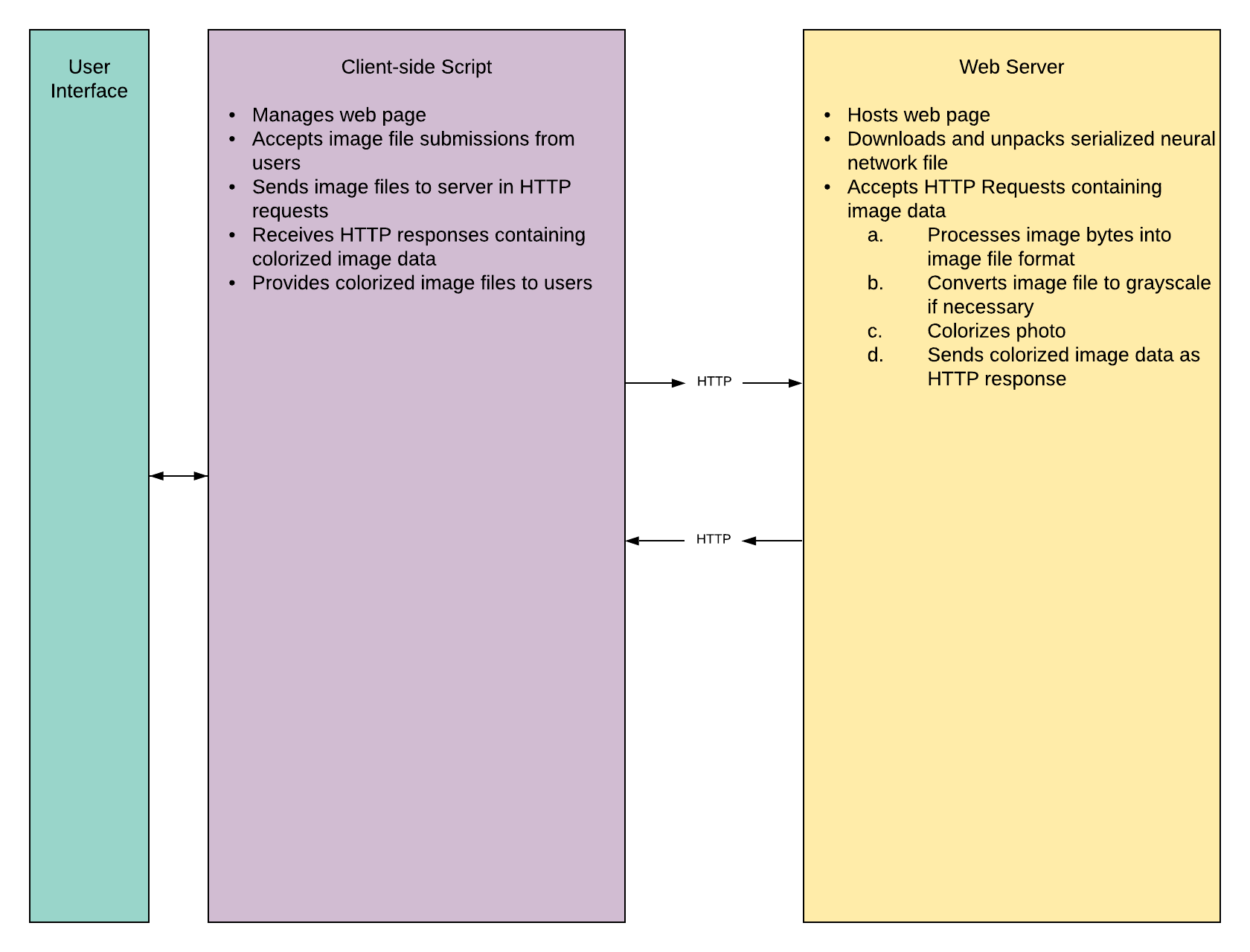


3 of 4



4 of 4

# Appendix C: Client-Server Relationship Design



# Appendix D: Usability Test Questionnaire

Try to colorize an image. If you do not have a grayscale image, use a color image instead.

1. Were you successful?

2. If so, were you satisfied with the result?

3. Describe any challenges you experienced.

4. How can the process be simpler?

5. How long did the process take to complete?

6. Feel free to provide any additional feedback you’d like to give:

1. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv:1505.04597 [Cs]*. <http://arxiv.org/abs/1505.04597> [↑](#footnote-ref-1)
2. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *ArXiv:1512.03385 [Cs]*. <http://arxiv.org/abs/1512.03385> [↑](#footnote-ref-2)
3. Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the Loss Landscape of Neural Nets. *ArXiv:1712.09913 [Cs, Stat]*. <http://arxiv.org/abs/1712.09913> [↑](#footnote-ref-3)
4. offe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv:1502.03167 [Cs]*. <http://arxiv.org/abs/1502.03167> [↑](#footnote-ref-4)
5. Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2019). How Does Batch Normalization Help Optimization? *ArXiv:1805.11604 [Cs, Stat]*. <http://arxiv.org/abs/1805.11604> [↑](#footnote-ref-5)
6. Smith, L. N. (2017). Cyclical Learning Rates for Training Neural Networks. *ArXiv:1506.01186 [Cs]*. <http://arxiv.org/abs/1506.01186> [↑](#footnote-ref-6)
7. http://cocodataset.org [↑](#footnote-ref-7)
8. Stephens, R. (2015). *Beginning software engineering*. John Wiley & Sons. [↑](#footnote-ref-8)
9. In reality, the page will contain my personal contact information—including my LinkedIn and GitHub—to show off this project to prospective employers. [↑](#footnote-ref-9)
10. In reality, these people will be friends and family. [↑](#footnote-ref-10)