Software Architecture and Design.

The implementation of the Photo Editor application involves multiple connected layers. The application's main method is found in the PhotoEditorApplication class. This class extends the Application class. The main method launches the start method. Next, a new FXMLLoader object is created using the PhotoEditorGUI.fxml file. The .fxml file is stored under: src -> main -> resources -> photoeditor.photoeditorFX -> PhotoEditorGUI.fxml. The GUI contains several tabs including File, Apply Filter, Adjust Color, and Resize. Each of the tabs contain different actions for the application. A stage is created and is ready to be displayed to the user.

The PhotoEditorController has several types of class variables. There is a Label object that changes the drag and drop label visibility in the application window. The ComboBox<String> object named comboBox1 contains the dropdown options in the Increase/Decrease Size menu. The ImageView variable is the imageView of the main window of the application. There are several Slider objects for the color sliders and the size sliders. The UserPhoto object named photo contains information about the photo that is uploaded by the user. The WritableImage wim object is necessary for the edited photo in the ImageView to be saved into a new file. And lastly the Stage object named mainStage represents the main stage of the application. This was necessary for the applyQuit method to work correctly.

The PhotoEditorController class contains methods that control the PhotoEditorGUI. The initialize method sets up the initial values and selections for the application. First, it initializes the size options for the Increase/Decrease Size dropdown menu. Next, it adds listeners to the Adjust Color slider controls. The sliders control brightness, contrast, hue, and saturation. Next, it adds a listener to the size selection of the Increase/Decrease Size dropdown menu including the values selected from the height/width sliders. This section of the code calls the private method parseScale which returns a double type value that aligns with the size selection. Two listeners are also added to the height and width sliders. Both the adjust color sliders and the height/width sliders affect the photo appearance in the imageView of the application window.

The next three methods in the controller class are onDragOver, onDragDropped, and onDragExited. These three methods allow the user to drag and drop a photo file directly into the application to edit. When a photo is dropped into the window, the slider/dropdown selections are defaulted back to starting values, the imageView is set to display the image file, and a UserPhoto object named photo is created and initialized. There is a private helper method called typeChecker that will only return true if the file uploaded is an accepted file type. The accepted file types are .png, .jpg, .peg, and .gif. If a user attempts to upload a file that is not an accepted file type, they will receive a pop up warning and the upload will be rejected. The onDragDropped method will only run these instructions if there is currently no photo in the application. If there is a photo already uploaded, the user will need to clear the workspace before adding a new photo into the window.

There are several options under the File tab. These options include: Open New File, Clear Workspace, Revert to Original, Save, and Quit. The Open New File selection is connected to the applyOpen method. If a photo has been added, this method will ask the user if they want to save their current photo with a yes/no selection window. For both yes/no selections the sliders and dropdowns are reset. If the user selects yes, the program will run the applySave method to save the photo to the user's device. It will then run the deleteImage helper method which deletes the photo from the imageView as well as deleting it from the UserPhoto object photo. If the user selects no, the deleteImage method is called. After this yes/no selection, the imageView is reset to its original size.

The next option under File is the Clear Workspace option. This option calls the applyReset method. If a photo has been uploaded, a yes/no selection window will appear asking the user if they want to revert their photo back to its original look. If the user selects yes, the sliders and dropdowns, imageView size, and the photo colors are all reset to the photo's original values. If the user selects no, the window simply disappears and the photo is not reset. If there is no photo uploaded before this selection is clicked, an error pop up box will appear warning the user.

Next in the File tab, as briefly mentioned earlier in the Open New File selection, is a save option. This option calls the applySave method when clicked. If there is an image uploaded, a file chooser window will appear. It will ask the user which location on their device they want to save the photo. There are also options to select the file extension of the newly created file. Depending on if the save was successful, a pop up box will appear confirming the successful save or warning that there was an error.

The last possible selection under the File tab is a quit operation. This selection calls the applyQuite method. The method starts by displaying a yes/no pop up window asking the user if they want to quit. If the user selects yes, the window is disposed and the program is completely exited. If the user selects no, the quitFrame will disappear and the user can continue editing in the application.

The next tab is named Apply Filters. There are several possible selections including original, black and white, grayscale, red, blue, and green. These selections are connected to various methods also within the controller class. The applyOriginal method is called when the original option is clicked. This method copies the original look and information from the image's source file. The imageView is updated to the photo's original colors and size. The applyBlackAndWhite, applyGrayscale, applyRed, applyBlue, and apply Green methods are similar in implementation. They create a new BufferedImage type object by creating/instianting their correlated Filter class object and using the applyFilter method. For example, within the applyBlackAndWhite method a BlackAndWhiteFilter class object named filter is created. The applyFilter method is used on the filter object and a new BufferedImage is returned. The imageView is then updated to match the revised image.

The filter classes extend the abstract Filter class. The Filter class has two class variables of type BufferedImage (to represent the photo), and an int (to represent the intensity of the filter). The parent class filter has getters and setters for these variables as well as an abstract method called applyFilter that returns a BufferedImage. The classes that extend the Filter class BlackAndWhiteFilter, GrayscaleFilter, RedFilter, BlueFilter, and GreenFilter override this method to create different color effects on the image's individual pixels. By changing the levels of red, blue, and green light in each pixel, the filters are able to edit the overall image.

There are several other helpful private methods in the controller class. The createCopy method creates a copy of the photo object and returns a BufferedImage type object. The deleteImage method sets the ImageView and the UserPhoto photo object to null. The selectFile method opens a file chooser window to select an accepted image file. The displayFile method displays the file into the ImageView or sends an error message that the photo could not be uploaded. There are two types of private pop up message methods. The yesNoBox displays a prompt with yes and no buttons that the user can select. The popUpBox method displays a prompt in a new popup window. And finally the resetSilders and resetSize methods reset the color sliders, size dropdown selection, and the size sliders.

The Photo interface sets a contract that all Photo objects must follow. This interface class is implemented through the UserPhoto class. The UserPhoto class contains several useful information about the photo file that is uploaded to the application. This includes the file itself, the BufferedImage of the photo, the file name/path/size, photo width, photo height, photo number and if the photo is edited or not. The class contains several getters and setters to read/write to the class variables.

Currently in the program’s source folder, there is also a disconnected class named Album. This class would theoretically contain an array of UserPhoto objects creating an album of uploaded and edited photos. It would also number each UserPhoto that was added. Its methods include getters and setters, grabPhoto (grabs a specific photo), addPhoto (adds a photo to the album), and deletePhoto (deletes a photo from the album). This class could be implemented for future versions of this application.

The Adjust Color tab has four sliders to manipulate the colors of the uploaded image. The brightness slider adjusts the photo by either adding white or black colors to all pixels. The hue slider adds different shades of colors to the photo. A user can add several different shades of colors to the image. The contrast slider affects the blend intensity between the different colors in the image. For example, if a photo has the black and white filter applied to it, it will make the blacks more black and the whites more white. The saturation slider affects the intensity of the overall colors. This makes the photo’s colors either pop or blend together.

The Size tab has three components. The first component is the Increase/Decrease Size dropdown menu. When a new size is selected, the photo’s imageView size will be edited to correlate with the selected option. This allows the user to make an image appear smaller or larger. The second and third components are the height and width sliders. By adjusting the sliders, the photo’s height and width can be stretched or shrunk.

Through all these Photo Editor application classes, a user can upload and download an image. The image can have a filter applied over it. The individual photo color components can be manipulated. And finally, the size of the photo can be dramatically changed.