Image Optimiser

Requirement: Images that are uploaded by vendors are generally in landscape view but user mobiles on which our app shows these images, in portrait view, hence cropping of images is required but point of interest in image can only be identified by designer team, hence GUI is required. They were using photoshop for completion of this purpose but re-opening images, defining parameters, again and again in photoshop required skills and were very time consuming. So there was a need to develop GUI tool kit easy and fast to handle images and transfer Images between server and local, were the necessity of this project, because they could only download and upload file from or to server one by one which was also very time consuming.

Work: I created a basic layout for this purpose (shown on next page) and decided that following windows would be needed for completion of this project:

1. **Input-Output Window:** Takes input and output directory as first input from user for Cropping and resizing images one by one. This window also connects user to other window with menubar. Open tab contains option of choosing single file as well as directory for input as shown in Figure 2, or you can also directly click on Input Directory and Output Directory button to open a FileDialouge box to choose start and destination folder. S3_Bucket tab connects to Download and Upload window as shown in Figure 3. Other tab contains Size Reducer button option which open a form for resizing images in bulk instead doing at individual level as show in Figure 4. Exit button closes window and would take you terminal from where you opened this window. From Last is designed for auto filling data of Input and output folder that was submitted last time. When some user is working on large project like folder of 500 or more photos and suddenly in middle he needs to close this software and when he comes next time he wants to start from where he left last, then he can do this by just clicking on this button.

Figure 1

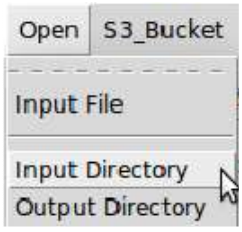


Figure 2

Figure 3

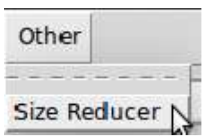
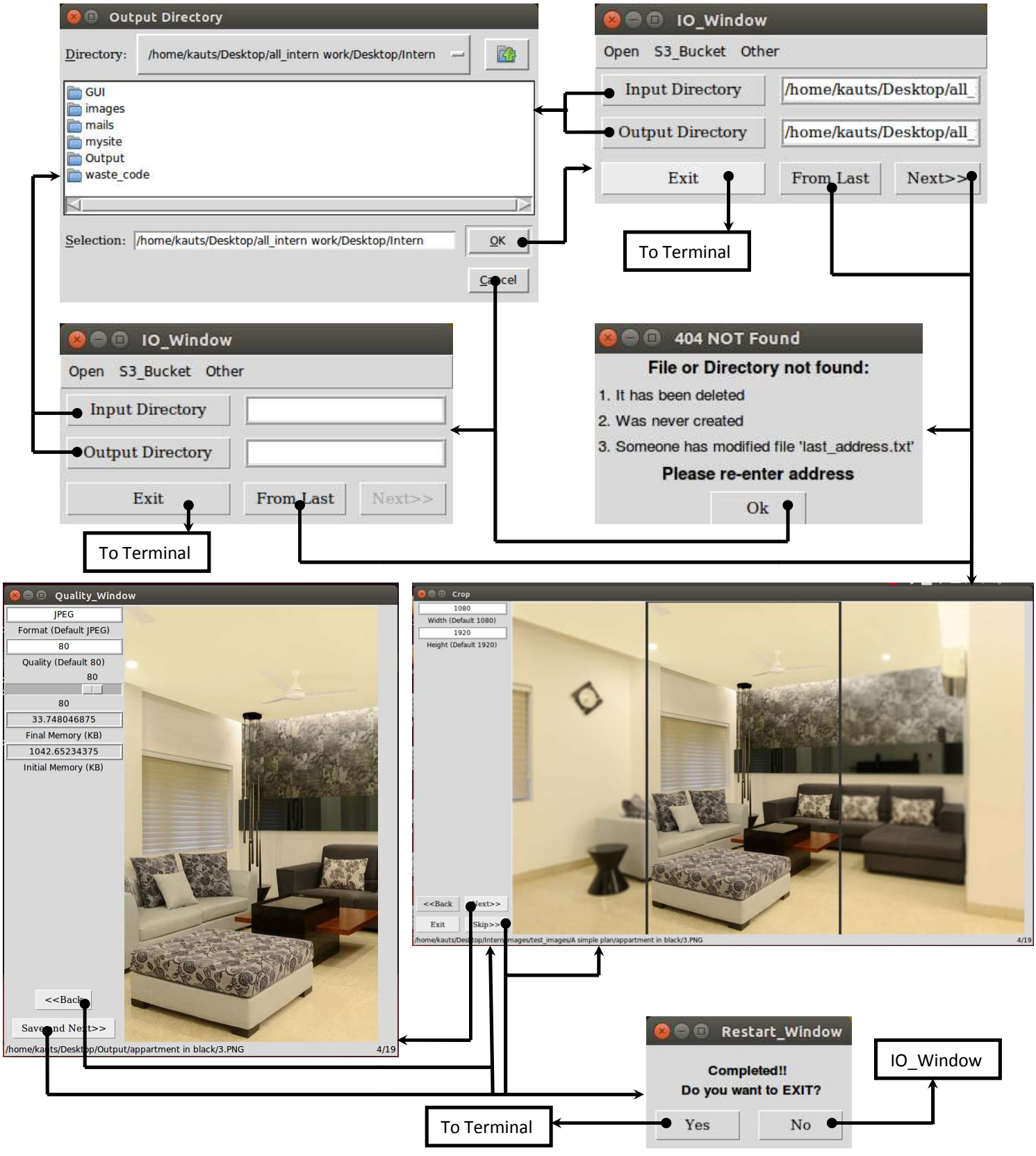


Figure 4

Process Layout of Cropping and Resizing GUI (Graphic User Interface) Developed



2. **Cropping Window:** After clicking on Next>> button, this window opens. This window provides option of selecting height and width ratio in which user want to crop and resize image. Cropping rectangle is shown by black boundary with focused region; other region is blurred to give user a better idea of what he is cropping. Next>> button takes to next step of resizing as setting quality of image, skip button skips images and move to next

Figure 5

one, it also get disabled when there is no next image remain. Using <<Back button user may go to last cropped image and re-edit that, Exit button takes to io_window. In the taskbar user can also see current image location and done images slash total images as giving current status of work. There was one more check button added later in this window named “keep aspect ratio” as user may only resize current picture without cropping it. (Figure 5)

3. **Quality Window:** This window provides option to set quality of image. Quality is percentage of color shades that a user wants to show in image. In JPEG, RGB is used hence color shades may vary from 0-255. PNG does not provide this option because of presence of 4^{th} byte in pixel for transparency. This is a kind of lousy compression done by JPEG format after re-sizing and entropy encoding so this window comes after cropping window. This window has option of format as user may select an option between JPEG and PNG according to their choice if JPEG distorting image too much. This window provides a knob which can be altered by user. This knob gives dynamic feature to window as lowering quality would distort image too much after a limit. In Final

Memory and Initial Memory Text Widget user can see how much compression on image has done dynamically. User can also manually enter value of quality in provided text box. User can also go back to cropping window of same image if he/she has cropped wrong part in image using <<Back button. Save and Next>> button would save this cropped and resized image in destination folder and would take to next image. If all Images are done then a completed window would pop up and would ask if you want restart any other task. Taskbar shows current status of task and picture, as address of image and how much task has completed. This window is shown in Figure 6.

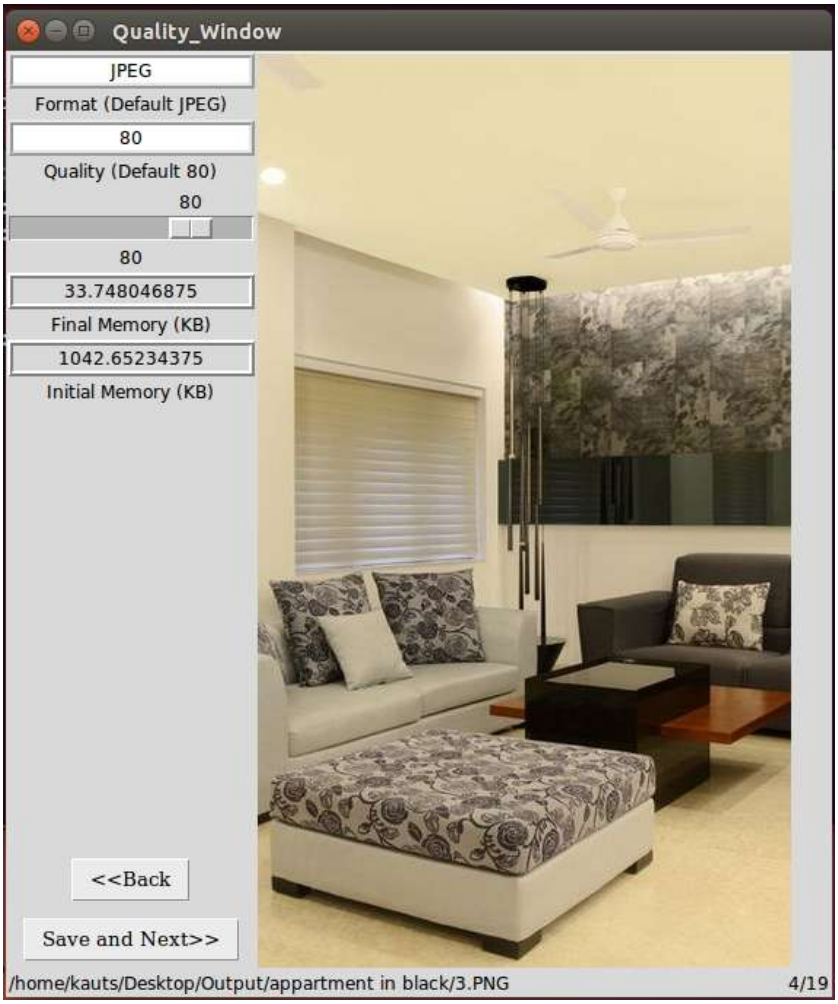


Figure 6

4. **404 Window:** This window pops up when manually provided address is not found in local and also pop up when someone try to use From Last button and address written in system generated file is not valid. This window provides some reasons of failure and user get another chance to confirm and correct addresses by clicking on Ok button. (Figure 7)
5. **Restarting Window:** This window pops up when some tasks is completed by software, then user either can restart new task by clicking on No button or user may exit to terminal by clicking on Yes button. This window is created using two frames packed in parent frame. One frame is on top where text is written and other frame is in bottom where grid

Figure 7

Figure 8

structure is used for putting keys. It's 200x100 initially but can be maximize with weight 1 to above frame and weight 0 in y-direction to below frame. All dimensions including padding are mentioned in appendix where all codes are attached. (Figure 8)

6. **Downloading and Uploading Window:** This Window is created using 3x3 framework with wt. 1,1,0 among rows and 0,1,0 among columns for handling expansion of window. Padding given between columns and rows is 5 and initial window size is 350x120 pixels at the center of screen. In download From button and in upload To button opens S3 FileDialouge to choose file from s3 bucket and similarly To button in download and From button in upload opens Local FileDialouge to choose folder in local machine as shown in layout on next page. By clicking on next button user can jump to io_window and simply clicking on Download and Upload button, user can easily transfer files between local and bucket in bulk.

Figure 9

Figure 10

7. **S3-FileDialouge Window:** This is 600x400 pixels window, created using grid framework with 6x7 row v/s column configuration. For handling expansion of this window 0,1,1,1,0,0,0 wt. is assigned to rows and 0, 1, 1,1,0,0 to columns. Last column is assigned to vertical scroll bar and similarly 5th row is assigned to horizontal scroll bar. These wt are assigned so that on expansion on list box, selection button, text area would expand and not buttons. In this window selection button gives flexibility to directly jump among parent folders, up button to move up by one among parent directory, Cancel button to close s3-filedialouge window and leave entry boxes as it is, Ok button to overwrite text boxes in download and upload window to what path has been selected. User can also see all files and folders inside a parent folder by just double clicking on it. It was tried to make s3-filedialouge box completely like local-filedialouge box.

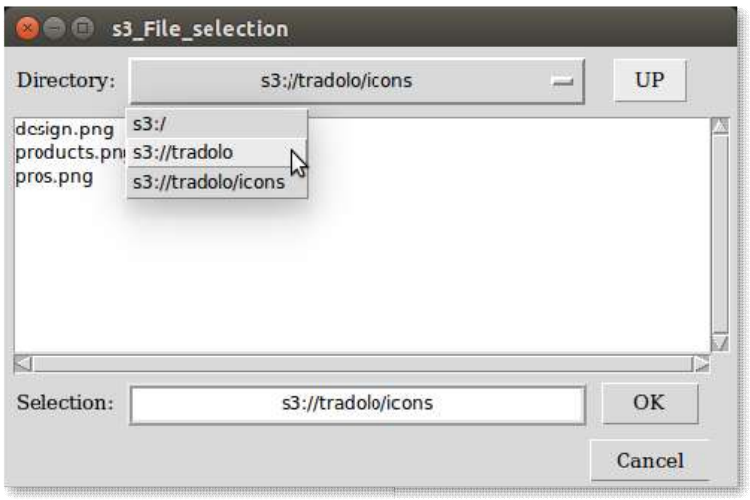
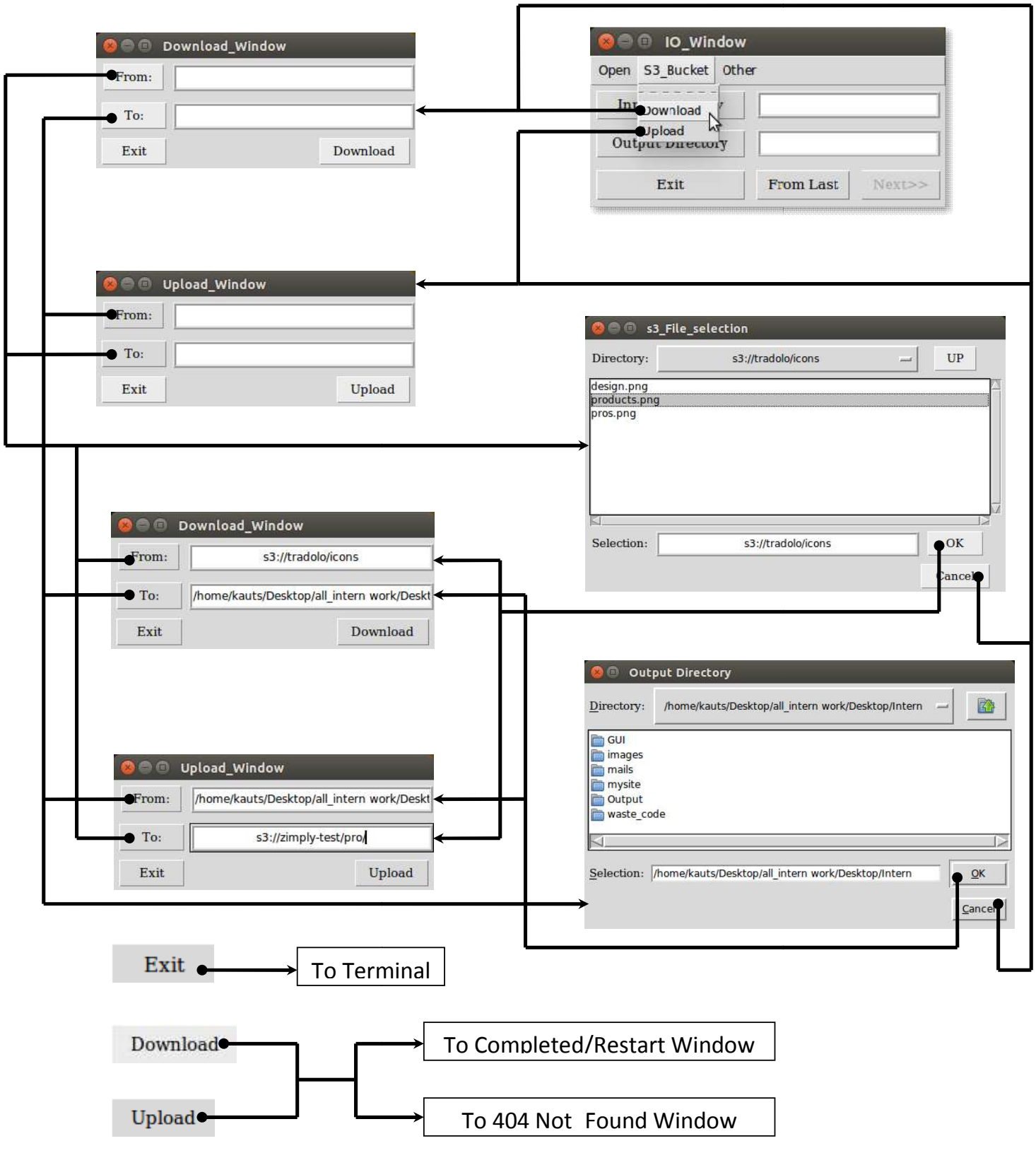


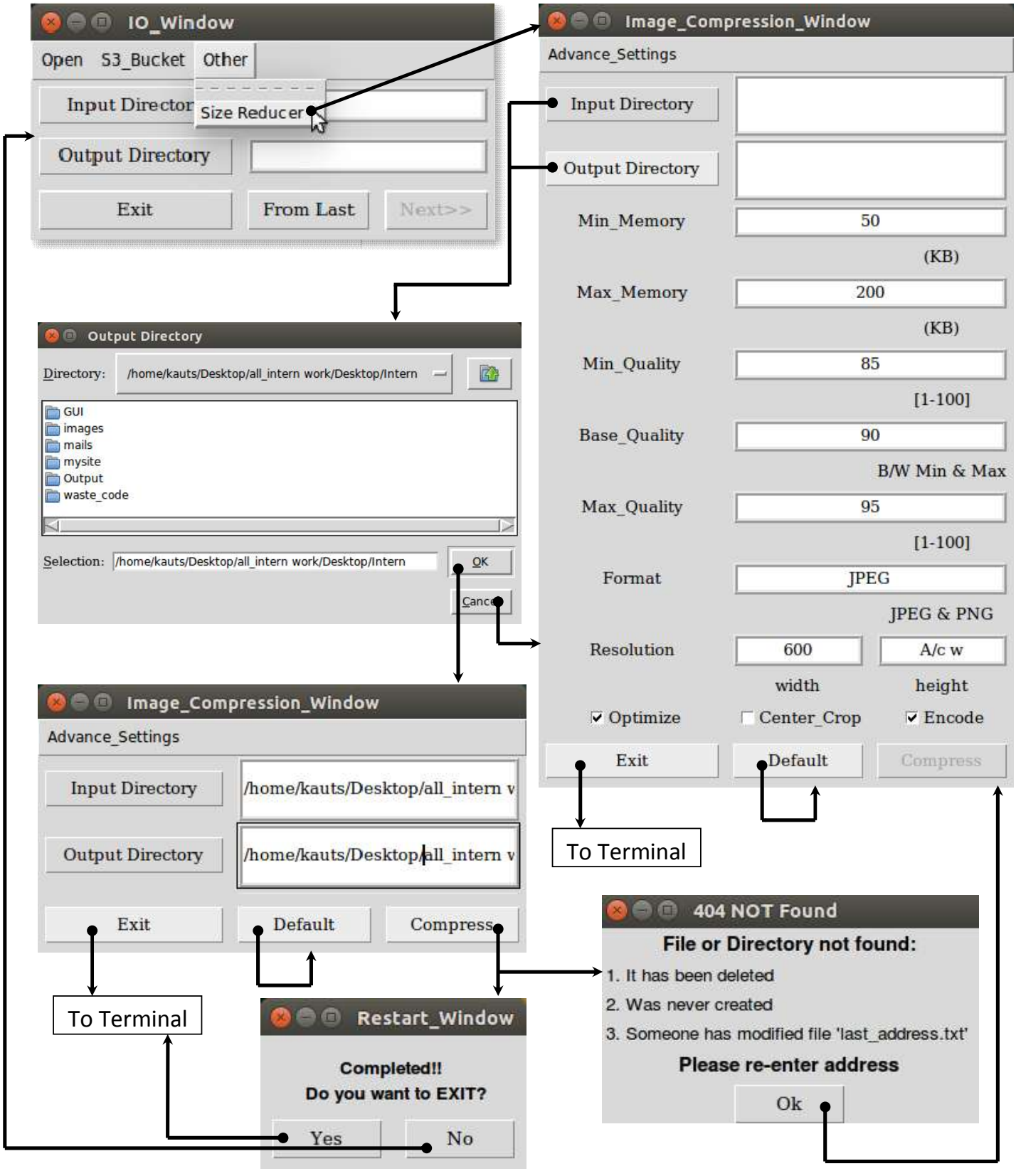
Figure 11

8. **Bulk Resizing form Window:** This window is 300x500 with 18x3 row v/s column configurations. Wt provided to columns are 0,1,1 and 1,1,0,…,0 to rows for handling

Process Layout of Downloading and Uploading GUI (Graphic User Interface) Developed



Process Layout of Size Reducing Tab GUI (Graphic User Interface) Developed



maximization of window. This window opens through on-click on size_reducer button in io-window as shown in process layout of size reducing tab. By clicking on compress button user can resize and compress images in bulk under provided parameters. It was designed by considering user flexibility of parameters and hence it was tried to make it completely soft coded and not hard coded. This window calls a script which iterate over input directory, find better parameter for a particular image by recursively saving image at different parameters so that given condition would be satisfied and memory would be as low as possible without degrading quality of image.