Final JAVA Project

Image filtering

Master's degreeComputer Engineering and Software Quality

Produced by :

ASTIGHFAR Ismail

EL QARRYCHY abdellatif

Ask by :

Pr.	**EL HIBAOUI Abdelaaziz**	**Département d'Informatique**

College year: 2022-2023

Table of Contents

List Of Figures

Introduction

This report presents an image filtering application developed using Java Swing and RMI with sockets. The application provides a user-friendly graphical user interface (GUI) that allows users to select and apply various filters to images, such as convolution, Noise, and Gray. The main server receives image filtering requests from the client and then distributes the processing work among multiple worker nodes to achieve improved performance.

The report provides a detailed overview of the application's architecture, design, and implementation. The focus of the report is on the class diagram and sequence diagram, which illustrate the application's structure and functionality. Additionally, the report includes snippets of code and screenshots of the GUI to demonstrate how the application works.

The class diagram describes the application's objects and their relationships, including the client, server, and worker. The sequence diagram illustrates the flow of control and data between these objects during the image filtering process, including the distribution of the image processing work to the worker nodes. The report also describes the image filtering process in detail, including how the selected filter is applied to the image, and the implementation of the load balancing strategy to distribute the processing work among the worker nodes.

Overall, this report presents a comprehensive overview of the image filtering application, including its architecture, design, and implementation details. It also discusses the advantages of using a client-server model with RMI and sockets, and how the load balancing strategy improves performance by distributing the processing work among multiple worker nodes. The report concludes by highlighting the strengths and limitations of the application and potential future improvements.

Project planning

Every project requires a plan that visually represents the progress of the different activities that make up a project. The plan is often represented as a Gantt chart. The Gantt chart is a tool used (often in combination with a PERT network) in scheduling and project management that allows for the visualization of the various tasks that make up a project over time.

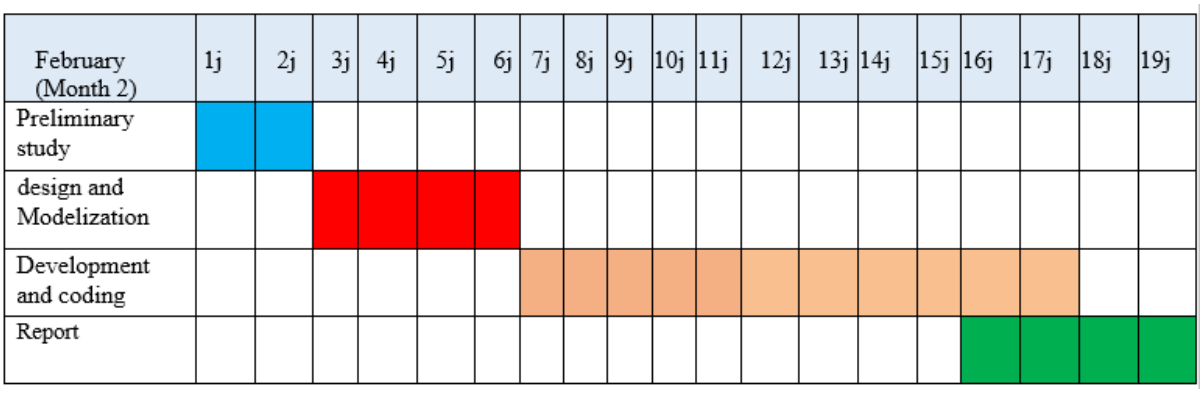

Figure 1 : Gantt diagram

System Architecture

The overall architecture of the image filtering application is based on a client-server model. The client is responsible for selecting and uploading the image to be processed, as well as choosing the filter to be applied to the image. The server receives the image filtering requests from the client, distributes the processing work among multiple worker nodes, and returns the processed image to the client.

The client-server model provides a number of benefits for this application. It allows the processing workload to be distributed across multiple worker nodes, which can improve performance by reducing processing time. It also enables the application to be more scalable, as additional worker nodes can be added to the system as needed to handle an increased workload.

The system is composed of three main components: the client, the server, and the worker nodes. The client component is responsible for providing the user interface for the application, including allowing the user to select the image to be processed, choose the filter to be applied, and display the processed image. The client sends the image filtering request to the server, which then distributes the processing work among the worker nodes.

The server component acts as the central controller for the application. It receives image filtering requests from the client, and then routes the request to one or more worker nodes based on a load balancing strategy. The server also receives the processed image from the worker nodes and returns it to the client.

The worker node component is responsible for actually processing the image filtering request. Each worker node is capable of applying the selected filter to a portion of the image, and can communicate with the server to receive additional processing work as needed.

Overall, the client-server model and the use of multiple worker nodes provide a scalable and efficient architecture for the image filtering application. The next section will provide a more detailed view of the architecture and design of the different components of the system.

Chaptre 1: Conception of the project

In this section we will construct a static view of the solution in the form of UML modelling.

1. Introduction

Before any project is carried out, the design plays a very important role in the preparation and organization of the project. It allows to create a general view on the different principles of our application. In this chapter we will justify the choice of UML and the modelling tool, the presentation of the different UML diagrams.

2. Choice of UML

The language UML (Unified Modeling Language) was designed to be our visual modeling language due to its semantically and syntactically rich. It consists of many diagrams of which three have been chosen for the description of our application: the class diagram and the sequence diagram.

3. Modelling tool

Figure 2: Modelling tool

Visual paradigm is a UML, Systems Modeling Language(SysML) and Business Process Modeling Notation (BPMN) software based on the Object Management Group (OMG).

5.Use case diagram

The diagram below shows all the use cases and describes the functional requirements of the system. Each use case therefore corresponds to a business function of the system, according to the point of view of one of its actors. The functional specifications follow directly from the use case diagram. It is a question of taking each case and describing very precisely the interactions of the actors with the system.

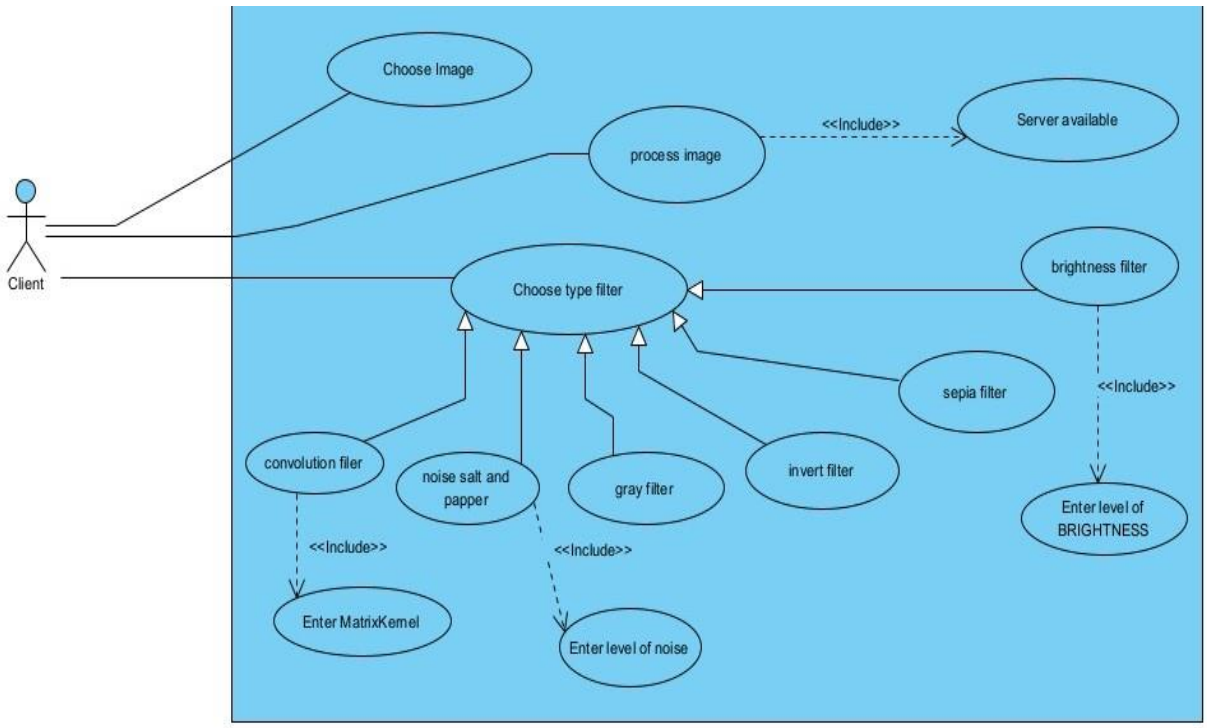

Figure 3: Use case diagram

4. Class Diagram

A class diagram provides a global view of a system by presenting its classes, interfaces and collaborations, and the relationships between them. Class diagrams are static: they show what interacts but not what happens during the interaction. The figure below represents the different classes of our system.

The class diagram includes the following Packages :

Client :

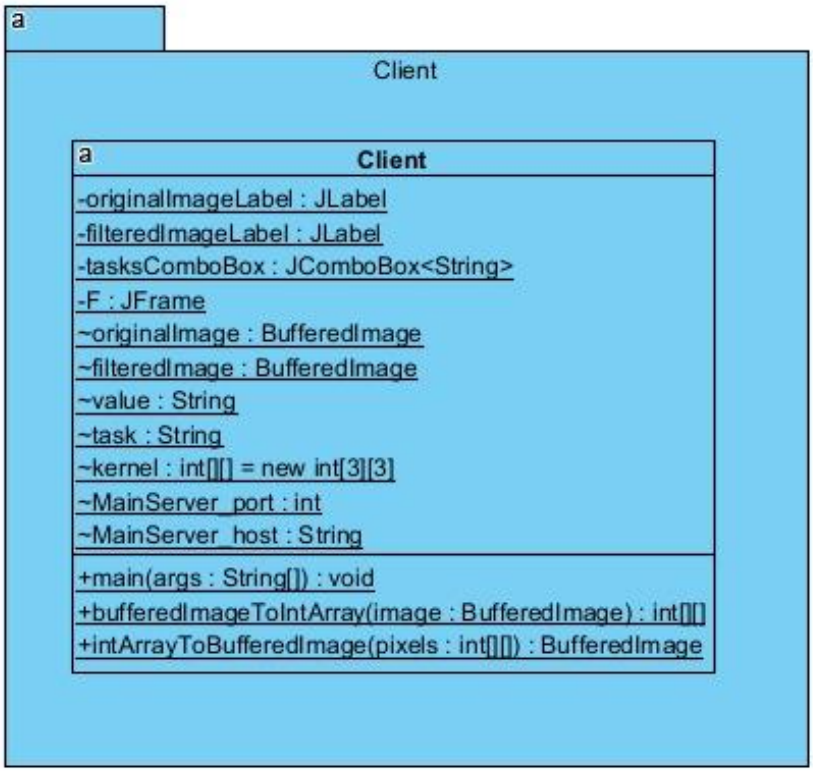

Figure 4 : Class diagram Client

Main Server & Filters Ressources :

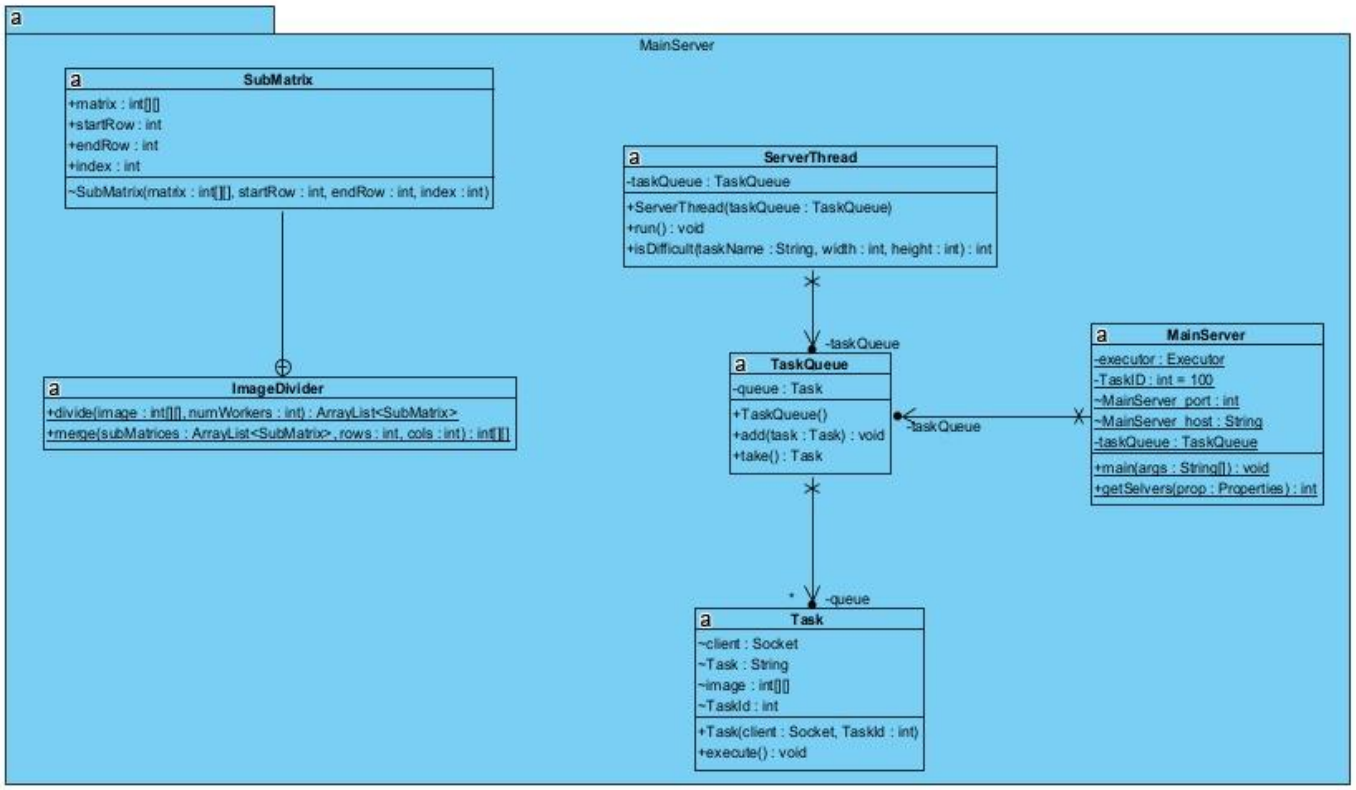

Figure 5 : Class diagram Main Server

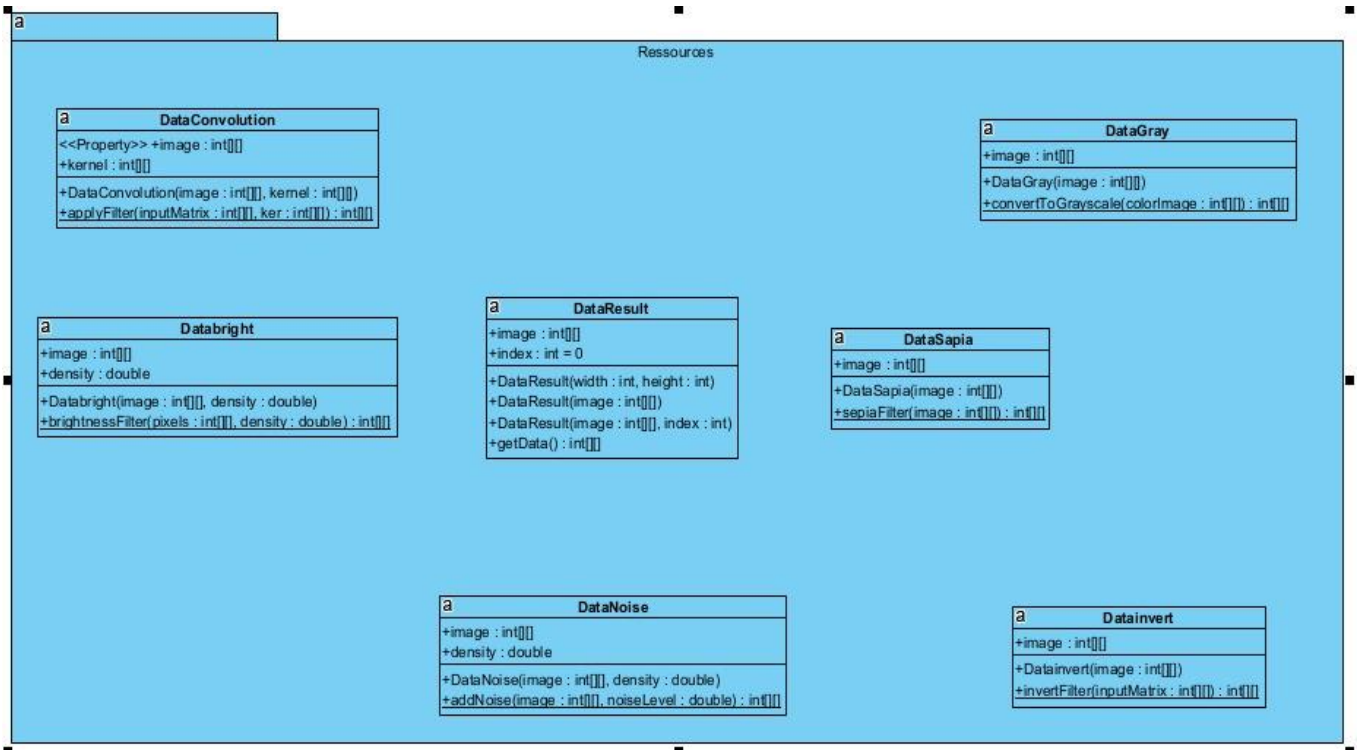

Figure 6 : Class Diagram Ressources

Worker & Rmi Ressources :

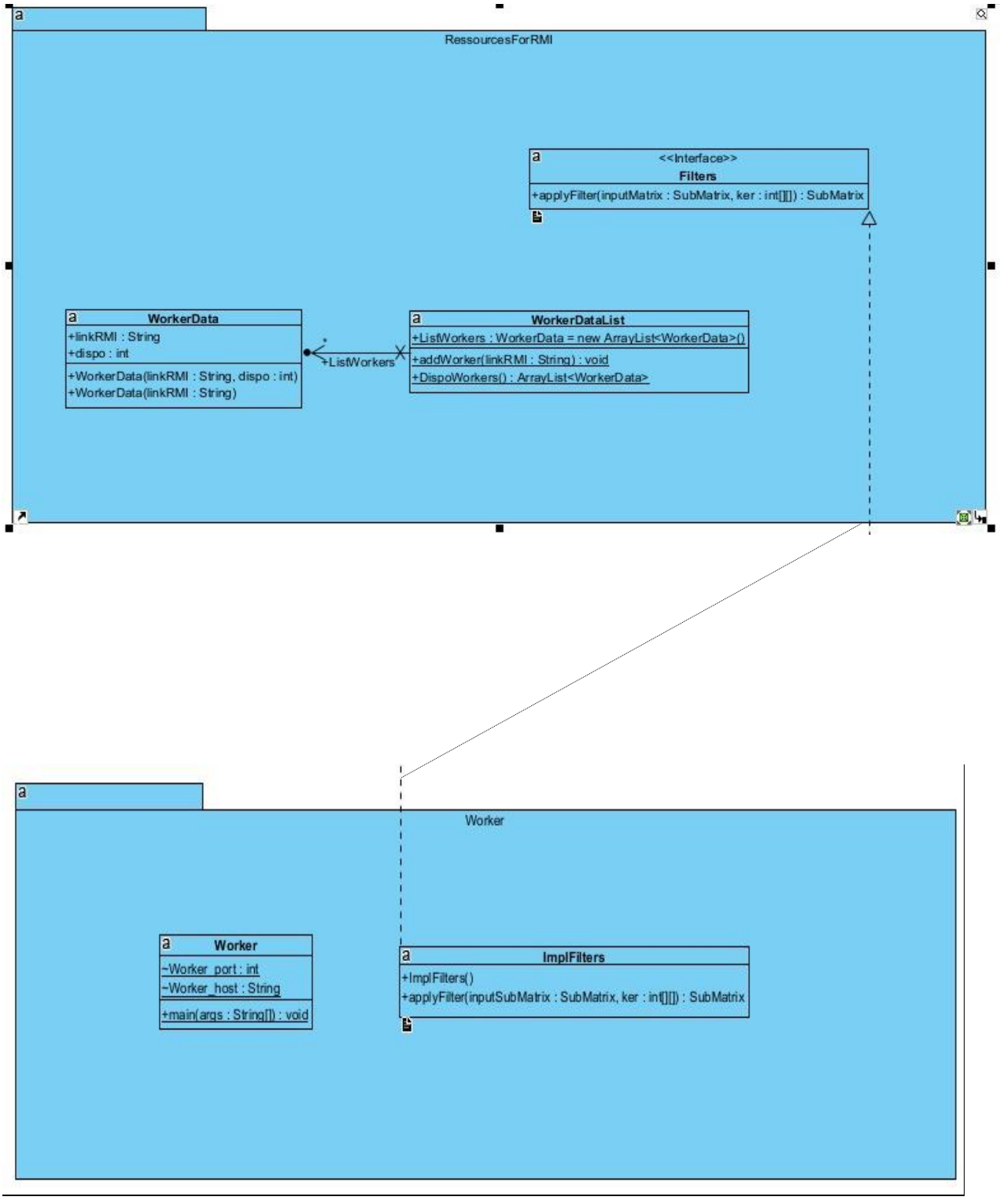

Figure 7: Class diagram Worker And RMI Ressources

5. Sequence Diagram

The sequence diagram is a UML interaction diagram. It represents the chronology of the transmission of messages between system objects and actors. It can be used to illustrate a possible scenario for a use case, the execution of an operation or simply the scenario of an interaction between classes of the system.

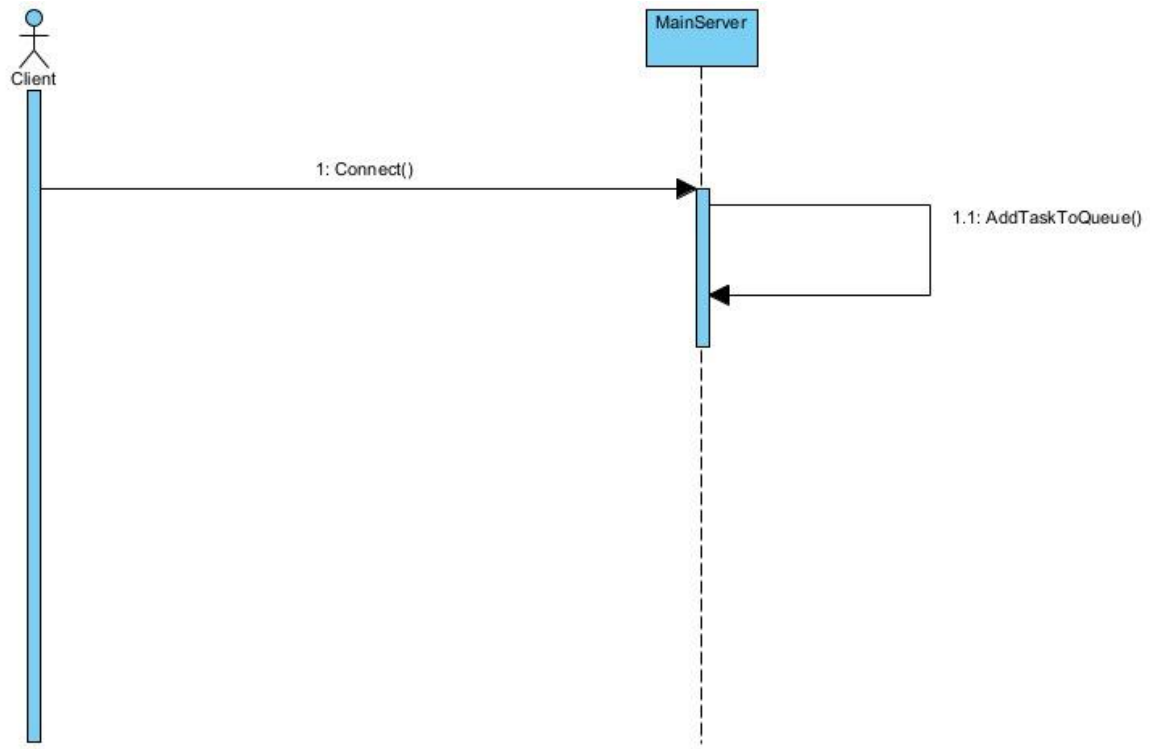

Figure 8: Sequence Diagram

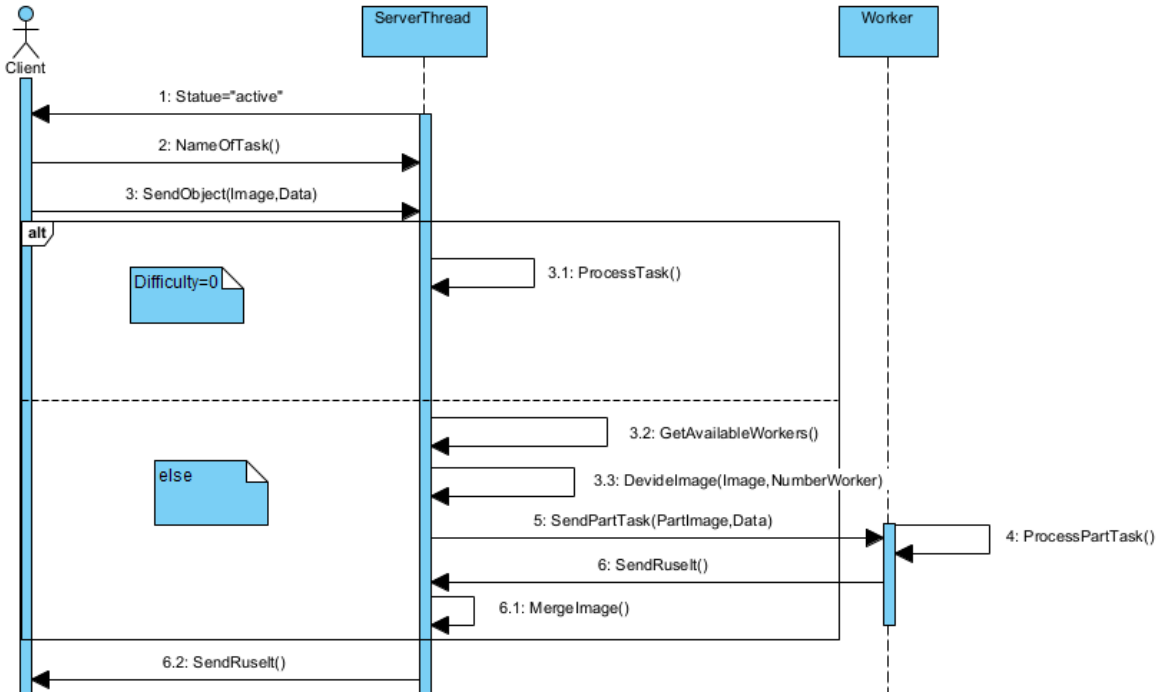

Figure 9: Sequence Diagram

6. Conclusion

In this chapter we have presented the design of our system using the different UML diagrams (class diagram and sequence diagrams) that helped us visualize the needs of the system to be developed as well as the different interactions between the objects involved in its operation, which leads us to our next chapter; project development.

Chaptre 2: Project development

In this chapter we will talk about the development tools and languages, the stages of implementation and give an idea of the application and its functioning.

1. Introduction

In this chaptre we will discuss folder structure and the main concepts of the application, the we move to the GUI and how it's works and how the user use the application.

2. Folder Structure

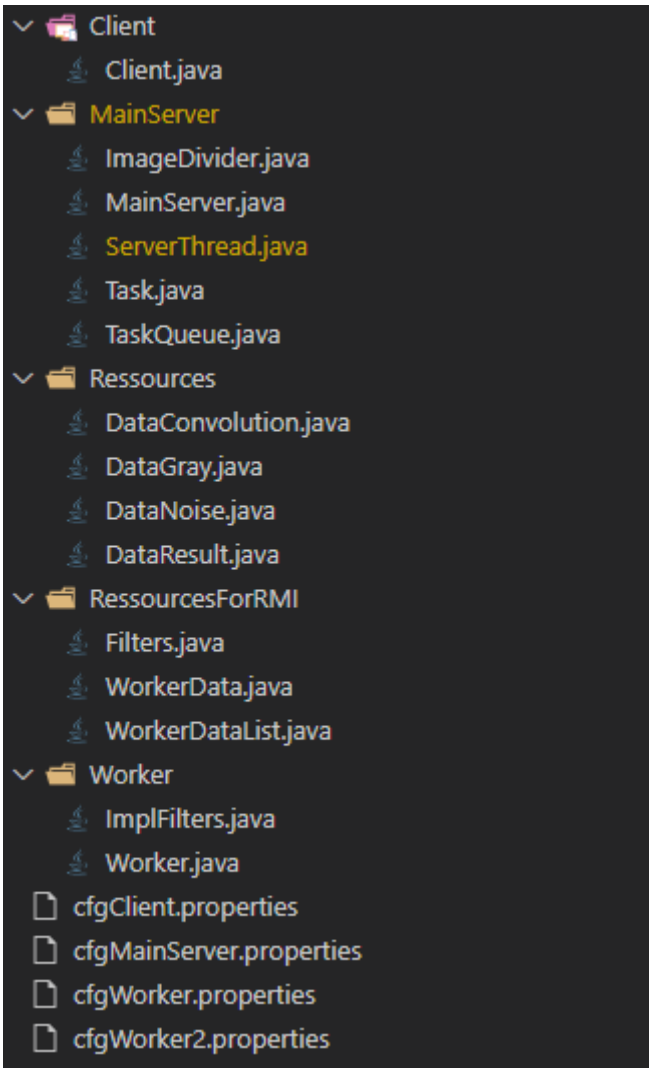

Figure 10: Folder Structure

3. GUI

First the user will open the application and then choose an image to apply a filter :

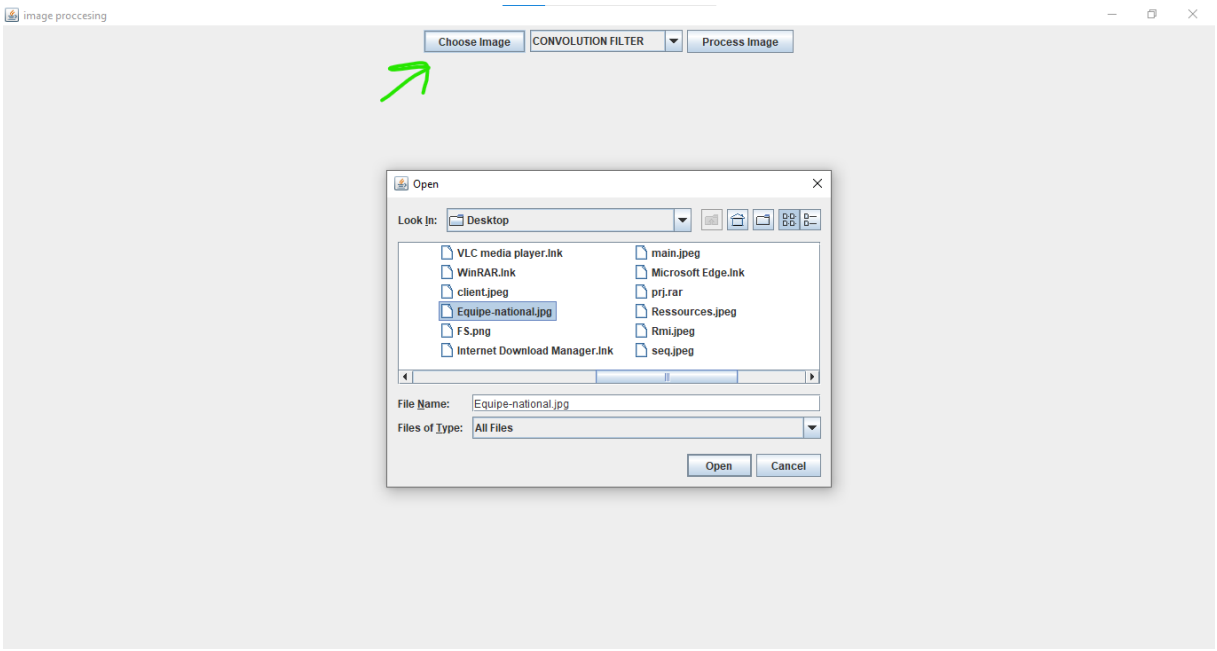

Figure 11: choose an image

then the image original image will show :

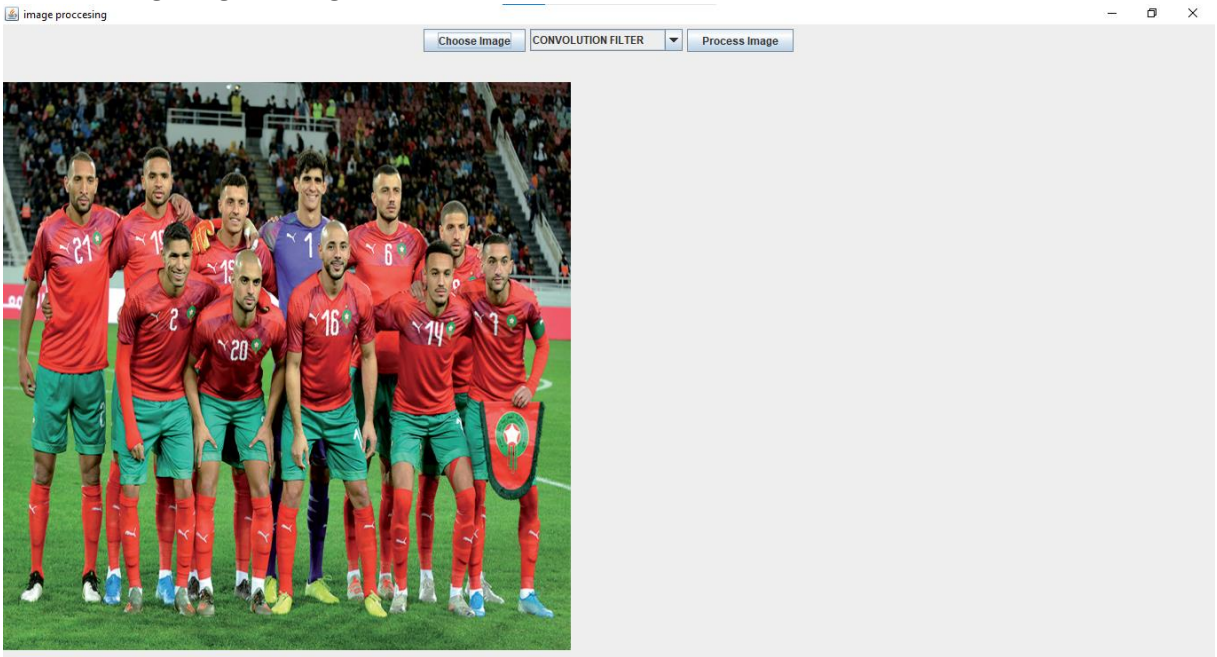

Figure 12 : show an image

The user choose a filter from the dropdown in this example the filter is convolution after he click he will need to choose a kernel file

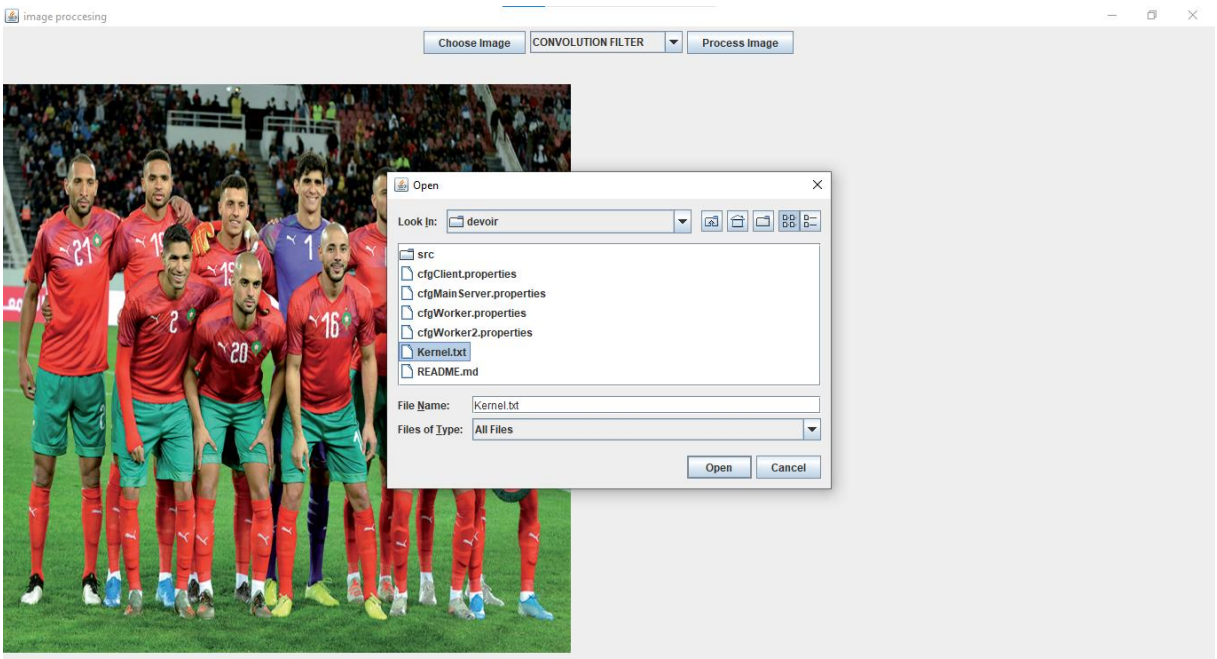

Figure 13 : Choose kernel Cnnvolution Filter

Then he click the process image button to send the image and the kernel to main server wich will devide the image and send each part to a worker to apply the filter and then merge it back and send the filtered image to the client wich diplay in the gui like this

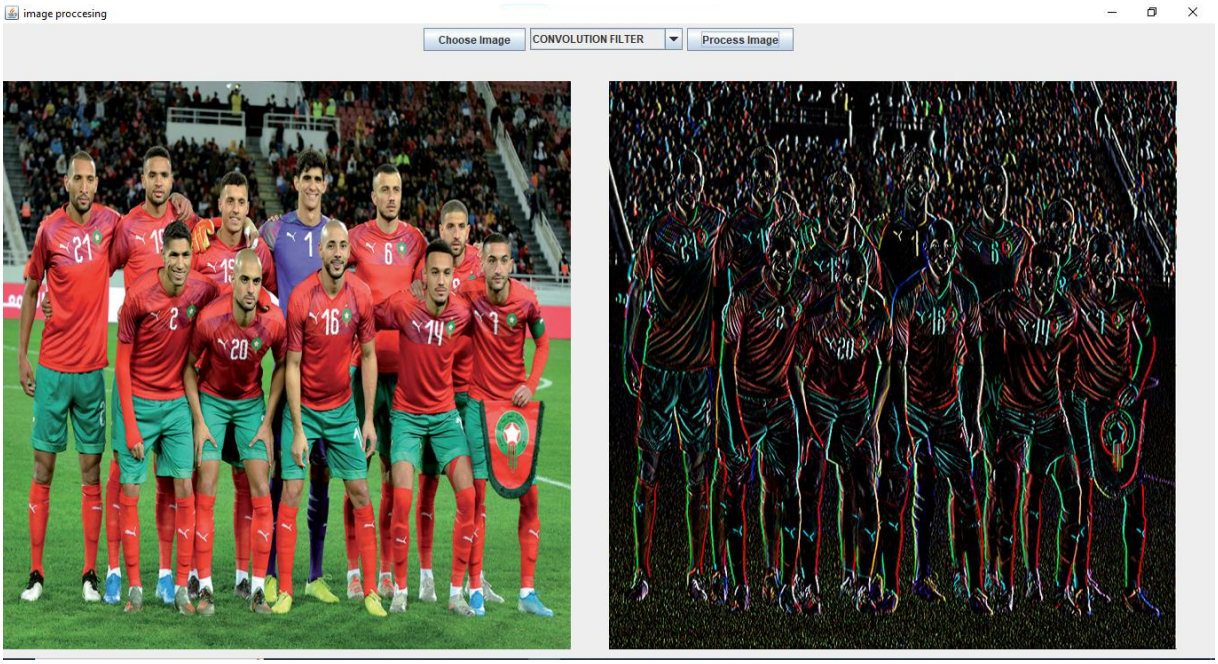

Figure 14 : Convolution Fliter

The user can also choose another filter like Salt & Pepper Noise and type the density wich the level on noise that he want

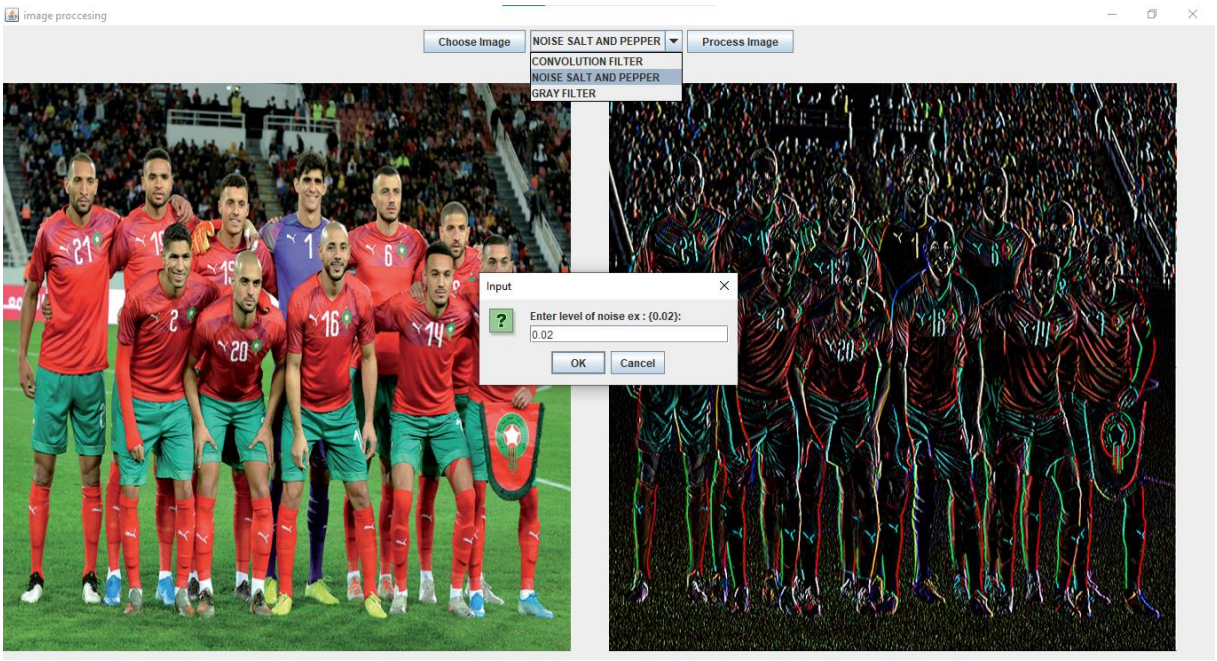

Figure 15 : Choose level the Noise Filter

The user click process button the main server will recieve the value and perform the filter and send the filtered image back

Figure 16: Noise Salt And Pupper

Last filter is gray filter the user just choose it and the main server will apply it

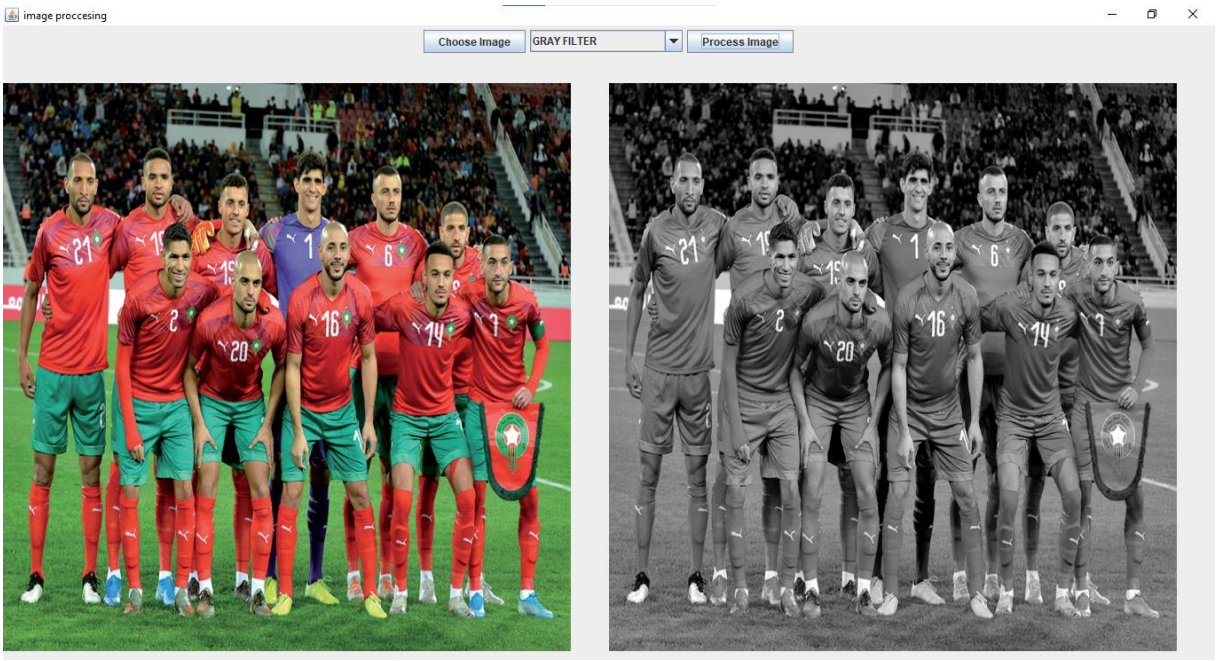

Figure 17 : Gray Filter

Conclusion

In summary, this report has presented an image filtering application that uses a client-server model to process image filtering requests. The application allows a user to select an image and apply a filter to it, with the server distributing the processing work to worker nodes as necessary.

The strengths of the application include its modular architecture, which allows for easy scalability and parallel processing of image filtering requests. The user interface is intuitive and easy to use, and the image filtering process is fast and efficient. Additionally, the use of Java Swing and RMI with sockets ensures that the application is platform-independent and can be run on any system that supports Java.

One potential weakness of the application is that it currently does not take advantage of hardware acceleration or parallel processing on the client side. This could be improved in future versions of the application to speed up the image filtering process even further.

Future improvements could also include expanding the range of image filters that the application supports, as well as incorporating more advanced image processing techniques. For example, the application could be extended to support machine learning algorithms for object detection or recognition. Additionally, further research could be conducted to explore the potential for the application to be used in other domains, such as medical imaging or remote sensing.

In conclusion, the image filtering application presented in this report demonstrates the power of client-server models for efficient image processing. While there is room for improvement, the application provides a solid foundation for further research and development in this area.