

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

**MANAGER FOR SCIENTIFIC EXPERIMENTS
AND LONG DURATION TESTING**

ZEKİCAN KAYAOĞLU

**SUPERVISOR
DR. GÖKHAN KAYA**

**GEBZE
2024**

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

MANAGER FOR SCIENTIFIC
EXPERIMENTS AND LONG DURATION
TESTING

ZEKİCAN KAYAOĞLU

SUPERVISOR
DR. GÖKHAN KAYA

2024
GEBZE

 <p>GEBZE TECHNICAL UNIVERSITY</p>	<p>GRADUATION PROJECT JURY APPROVAL FORM</p>
--	--

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 25/01/2024 by the following jury.

JURY

Member

(Supervisor) : Dr. Gökhan Kaya

Member :

ABSTRACT

Working on a single computer in long-term experiments and tests is quite inefficient both for that computer and for our time. The aim of this graduation project is to develop a program that can be used for such situations and saves time by working with more than one computer on the same network.

The aim of the project is to ensure that after defining an experiment on a computer with a simple script language, this experiment is converted into jobs, distributed to working computers and run on this computer. The experiment result should be collected on master computer without leaving any files on other computers.

The program that provides these features is written in Java language and has two different interfaces for the main computer and the working computers. The project meets the required specifications and has successfully passed many tests. However, further development of the project is still possible.

ÖZET

Uzun süreli deneylerde ve testlerde tek bir bilgisayarda çalışmak hem o bilgisayar için hem de zamanımız açısından oldukça verimsizdir. Bu bitirme projesinde amaç bu tür durumlar için kullanılabilecek, aynı ağdaki birden fazla bilgisayarla çalışma yaparak zaman tasarrufu sağlayan bir program geliştirmek.

Projenin amacı basit bir script diliyle bir bilgisayardan deney tanımı yapıldıktan sonra bu deneyin işlere dönüştürülüp çalışan bilgisayarlara dağıtılmasını ve bu bilgisayarda çalıştırılmalarını sağlamaktır. Deney sonucunu da diğer bilgisayarlarda herhangi bir dosya bırakmadan ana bilgisayarda toplanmasıdır.

Bu özellikleri sağlayan program Java diliyle yazılmıştır ve ana bilgisayar ve çalışan bilgisayarlar için iki farklı arayüze sahiptir. Proje gerekli özelliklere karşılayabilecek durumdadır ve birçok testi başarıyla geçmiştir. Ancak projenin daha da geliştirilmesi hala mümkündür.

ACKNOWLEDGEMENT

I was in constant communication with my supervisor Dr. Gökhan from the day I started the project. So I would like to thank him for always supporting me in terms of ideas. And I would like to thank Mr. Osman Utar for his support with the computers in the laboratory.

Zekican Kayaoğlu

CONTENTS

Abstract	iv
Özet	v
Acknowledgement	vi
Contents	viii
List of Figures	ix
1 INTRODUCTION	1
1.1 Project Description	1
1.2 Project Purpose	1
2 Project Content	2
2.1 Project Requirements	2
2.2 Project Architecture	3
3 Project Implementation	5
3.1 Introduction and Preparation	5
3.2 Planning	5
3.3 Code Development	6
3.4 Testing and Debugging	6
3.5 Abandoned Ideas	6
4 Use of The Project	8
4.1 Experiment Initialization	8
4.1.1 Start Server	8
4.1.2 Client Connection	8
4.1.3 Experiment Definition	9
4.2 Start of Experiment	11
4.3 Execution Time of the Experiment	11
4.4 End of Experiment	13
4.4.1 Experiment Results	13
4.4.2 Client Performances	13
4.4.3 Client Status	14

5	Tests	15
5.1	Test Scenarios	15
5.2	Test Results	15
6	Success Criteria	16
6.1	Success Criteria of the Project	16
6.2	Completed Criteria	16
7	Conclusion	17
7.1	Successful Conclusions	17
7.2	Failed Conclusions	17
7.3	What can be added to the project?	17
	Bibliography	18

LIST OF FIGURES

2.1	Project Diagram	4
3.1	Timeline	5
4.1	Server Interface	8
4.2	Client Interface	9
4.3	Script Example	10
4.4	Beginning of Experiment	11
4.5	States of Output Files	11
4.6	Output.txt Example	12
4.7	Error Status	12
4.8	ErrorOutput.txt Example	12
4.9	Experiment Output	13
4.10	Experiment Output Folders	13
4.11	Client Performances	14
4.12	Client Status	14
5.1	Scenario Results	15

1. INTRODUCTION

1.1. Project Description

With this project, the user can upload a script file that defines the Python experiment to the desktop application and start the experiment. Then, the script is read, tasks are created and shared to auxiliary computers, waiting for the work to be done, and the results are collected again on the user's computer.

In the experiment definition script file, the user can also use parameters and input files when creating jobs, and the output files of previous jobs can be used as input in the next jobs. In the server interface, the user can see the experiment phase, the total number of jobs and the number of remaining jobs, the current status of actively running clients and the files coming from these clients, and the total experiment time at the end of the experiment. At the end of the experiment, the outputs of all jobs and the performance status of the clients (the number of jobs they have done and their total working time) are uploaded to the server computer.

1.2. Project Purpose

With this project, instead of running jobs of different definitions on a single computer, by using auxiliary computers on the same network to run these jobs, the workload and time costs per single computer will be significantly reduced and more testing opportunities will be provided.

2. PROJECT CONTENT

2.1. Project Requirements

- An interface where the user can start the server and load the script file, then follow the job management, see the number of jobs, see the status of the files coming from connected clients, learn the duration of the experiment, in short, follow the experiment.
- An interface where client computers can be named and connected to the server and the status of sent files can be seen.[1]
- Reading the experiment script received from the user and creating the necessary task list.
- Socket system for communication of the master computer with the slave computers.
- Sending the files required for the jobs from the server to the clients, and the output files from the clients to the server computer.
- Executing the task sent to the client and creating the result files.
- Instant transfer of the file transfers of the completed tasks to the interfaces.
- Downloading the output files coming to the server into folders according to the task number in the specified file directory.
- As a result of the experiment, the clients' creating a file for information on working times and the number of tasks they run.

Used for these requirements:

- Netbeans for Java development
- Socket.io[2] library for socket system
- Swing library for interface
- Virtual machine for project testing
- For big tests computer lab

2.2. Project Architecture

In the project, server and client interface applications were created entirely using Java. The user installs the script via the server interface. Experiment definitions are converted into jobs on the server and distributed to clients via socket. These jobs are executed in clients, and their outputs and some performance reports are sent back to the server via socket.

There is constant communication between the server and the client while the experiment is being performed. When the client is done, it sends and deletes the created files and requests a new job from the server. If there is still work in the task pool, a new job is sent, but if not, the client waits for the entire experiment to be completed, and when the entire experiment is finished, clients transmit their performance reports to the server.

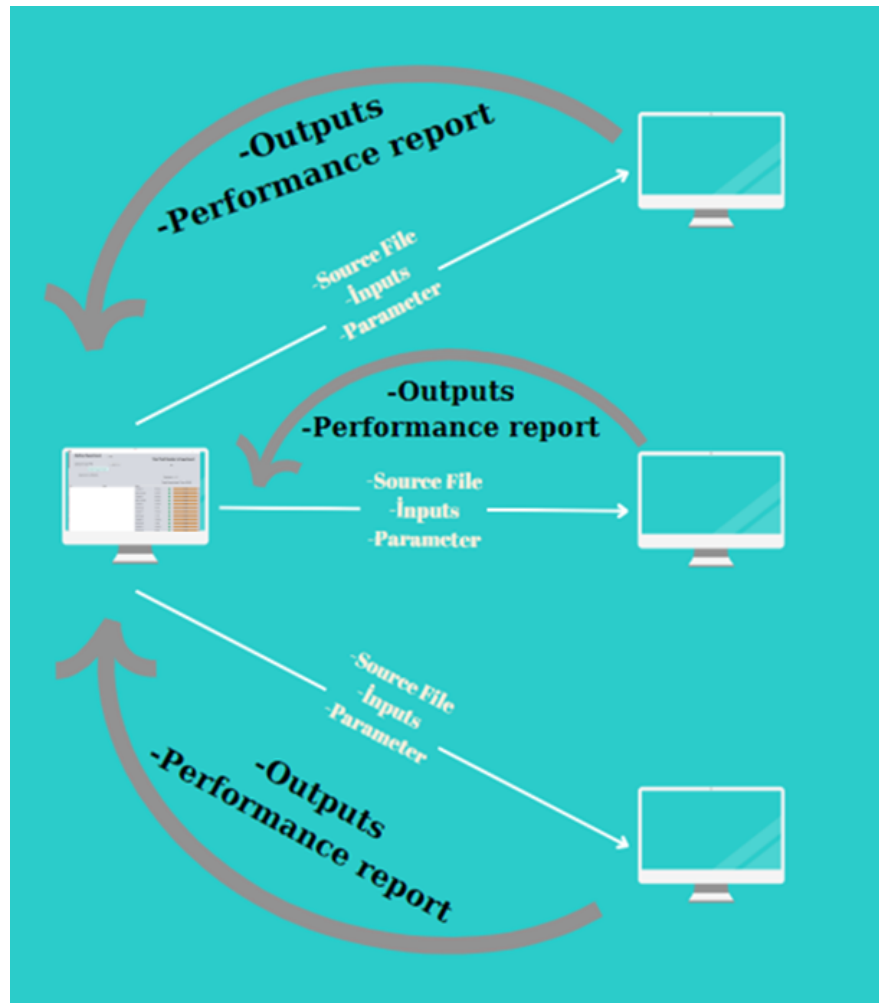


Figure 2.1: Project Diagram

3. PROJECT IMPLEMENTATION

3.1. Introduction and Preparation

Since I will be using interface for the project, I chose to write in Java. When I researched socket programming libraries, I found that there were sufficient resources.[3] The most important part for the project was that this communication should occur correctly, the rest would be left entirely to my programming knowledge and ability to solve problems.

3.2. Planning

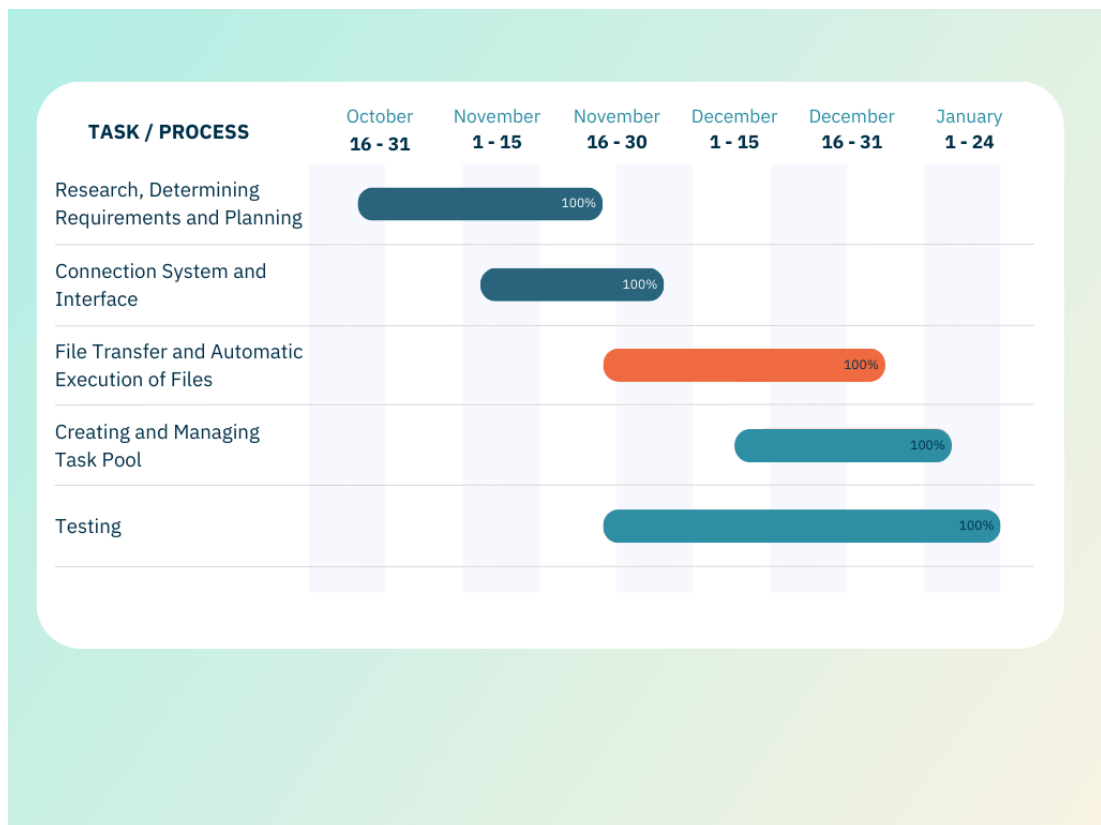


Figure 3.1: Timeline

3.3. Code Development

1. First, I started writing socket communication codes for the server and client sides.
2. Then, I designed two separate interfaces to initiate socket communication.
3. After writing the file transfer codes, I tested the socket connection and file transfers with these interfaces.
4. I prepared the section to display these transfers in the interfaces during file transfers.
5. After these parts were successful, I wrote the part of creating a task pool according to the experiment description received from the user.
6. After the task pool was created, I added the necessary parts for the management and distribution of this pool.
7. Afterwards, I added the necessary code sections to execute these sent files on the clients and return them.
8. Finally, I finished the project by writing the codes for collecting the test results on the server and displaying certain results to the user in the interface.

3.4. Testing and Debugging

I continued by performing tests in almost all phases of the project. When the socket communication and file transfers were completed, I tested the system to make sure it was working correctly and continued. In the first tests of the project, I performed my tests with 2 clients using a virtual machine. Afterwards, when I thought the project was over, I started doing major tests on the computers at school, but I encountered many problems. While solving these problems, I ran tests in the laboratory for days.

3.5. Abandoned Ideas

- At first, experiment definition was done by the user uploading experiment files instead of script files, but later, with my supervisor's idea, I switched to using script files.
- The project was planned to be compatible with all operating systems. Windows compatible codes are still included in the project, but after a while, continued only with Ubuntu because it was more difficult to test.

- A task execution system compatible with all programs, not just Python, was considered, but this was abandoned because the requirements on client computers were increased considerably.

4. USE OF THE PROJECT

4.1. Experiment Initialization

4.1.1. Start Server

The server interface is as shown in the Figure 3.1. The user can start the server with the "Start Server" button. The list of clients connecting to the server can be seen in this interface.

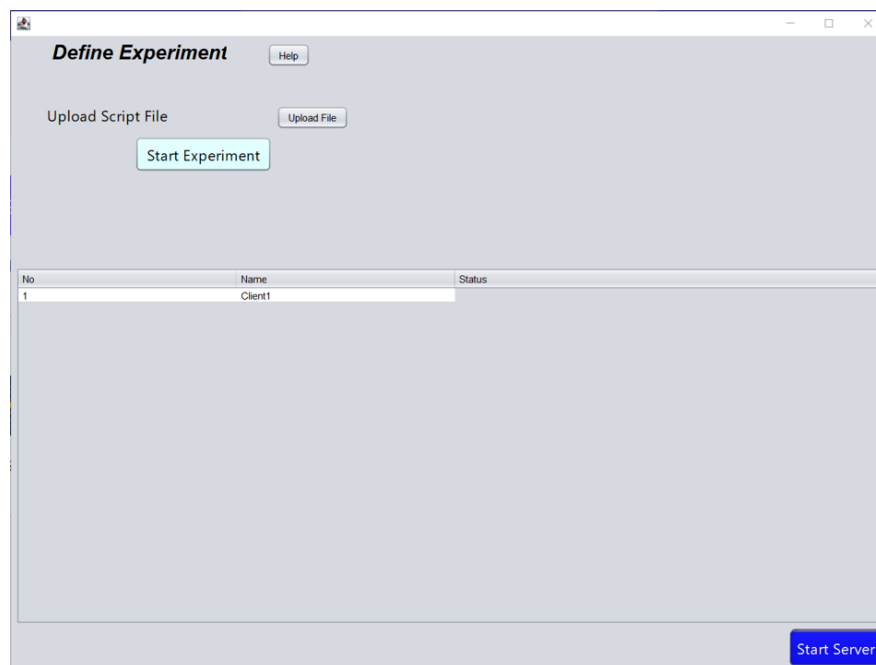


Figure 4.1: Server Interface

4.1.2. Client Connection

After the server is started, you can connect to the server by entering the name in the interface in Figure 3.2 on the client computers and pressing the "Connect" button.

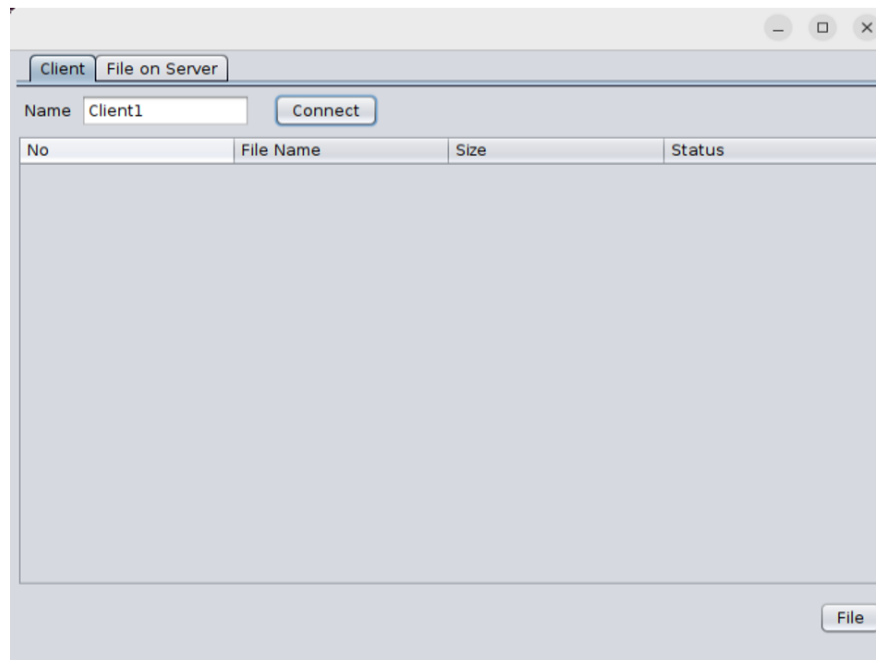


Figure 4.2: Client Interface

4.1.3. Experiment Definition

Afterwards, the user creates a script as shown in the 4.3 and uploads it to the server with "Upload File" button.



```
script - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
var_if inputFiles = "C:\Users\zekican\Desktop\test\source\giris_foto.jpg"
var_if inputFiles2 = "C:\Users\zekican\Desktop\test\source\panda.jpg"

var_id inputFiles3 = "C:\Users\zekican\Desktop\test\input"

var_if inputFiles5 = "C:\Users\zekican\Desktop\test\video\video.mp4"
var_if inputFiles4 = "C:\Users\zekican\Desktop\test\input\video.mp4"
var_if inputFiles6 = "C:\Users\zekican\Desktop\test\source\b"

var_p param1 = "12"
var_p param3 = "122"
var_pl param2 = [12,13,43,324,20]

var_exe exe4 = C:\Users\zekican\Desktop\test\source\foto.py inputFiles

Wait

var_out out1 = exe4

var_exe exe5 = C:\Users\zekican\Desktop\test\source\foto3.py out1

Wait

var_out out2 = exe5

var_exe exe6 = C:\Users\zekican\Desktop\test\source\foto.py out2
```

Figure 4.3: Script Example

Parameters in Script:

var_if: The input file can be defined by giving a single file path.

var_id: By giving the folder path containing many files, all files in it can be defined as input files.

var_p: A single parameter can be defined.

var_pl: Multiple parameters can be defined by providing a parameter list.

var_exe: The jobs to be executed can be defined using Python codes, input variables and parameter variables in different ways.

var_out: An out can be defined to use the outputs of previous jobs as input again.

Wait: The "Wait" keyword should be placed between the tasks to be done sequentially.

When the "Start Experiment" button is pressed, the program reads this script, creates tasks, and then starts transferring the jobs to the connected clients one by one.

4.2. Start of Experiment

When the experiment starts, as seen in the 4.4, both the total number of tasks in the experiment, the number of remaining tasks, and the experiment status are transferred to the user in the server interface.



Figure 4.4: Beginning of Experiment

4.3. Execution Time of the Experiment

As the tasks in the experiment are completed by the clients, the transfer status of the file is shown with a progress bar, with the task number written next to the status section next to each client. And with the icon next to them, the user can see what the status of the task is as shown in 4.5. The output files resulting from the task execution and, in addition to these files, an output.txt file showing which client is running the terminal output and the task are also sent from the client to the server.


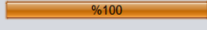

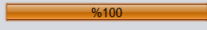

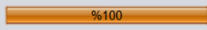

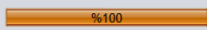

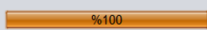

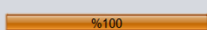
No	Name	Status
1	Client1	1_output.txt 82 bytes   %100
		cikti_foto.jpg 6,2 KB   %100
		2_output.txt 68 bytes   %100
		renkli_foto.jpg 7,3 KB   %100
		3_output.txt 83 bytes   %100
		cikti_foto.jpg 6,1 KB   %100

Figure 4.5: States of Output Files



Figure 4.6: Output.txt Example

If an error occurs during the task execution as shown in the 4.7, the user can understand it from the cross icon and the error output for these tasks is transferred to the server in the "ErrorOutput.txt" file.

No	Name	Status
1	1_output.txt	82 bytes ✓ %100
	cikti_foto.jpg	6,2 KB ✓ %100
	2_output.txt	68 bytes ✓ %100
	2_ErrorOutput.t..	382 bytes ✗ %100
	3_output.txt	70 bytes ✓ %100
	3_ErrorOutput.t..	483 bytes ✗ %100

Figure 4.7: Error Status

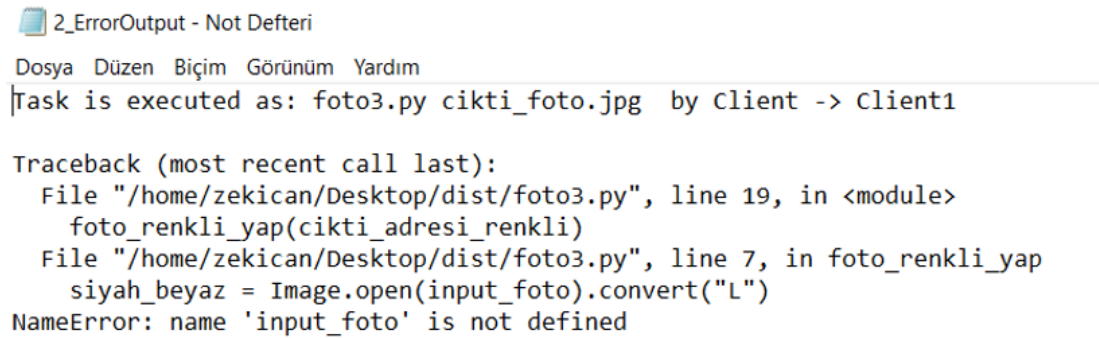


Figure 4.8: ErrorOutput.txt Example

4.4. End of Experiment

4.4.1. Experiment Results

At the end of the experiment, the user can view the experiment information as shown in the 4.9.

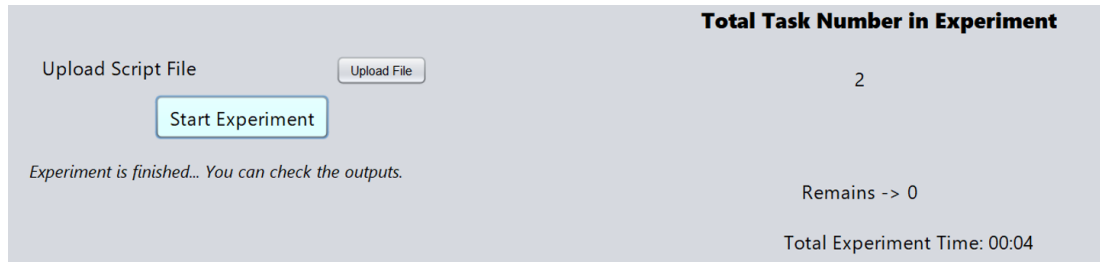


Figure 4.9: Experiment Output

When the experiment ends, the output files of that task can be viewed in a separate folder for each task in the experiment. There is also a performance folder for clients as shown in the 4.10.

Ad	Tarih	İçerik	Boyut
ClientPerformances	20.01.2024 18:07	Dosya klasörü	
Task1	20.01.2024 21:00	Dosya klasörü	
Task2	20.01.2024 21:00	Dosya klasörü	
Task3	20.01.2024 18:07	Dosya klasörü	

Figure 4.10: Experiment Output Folders

4.4.2. Client Performances

At the end of the experiment, client performances are automatically generated on the server computer and are as shown in the 4.11.

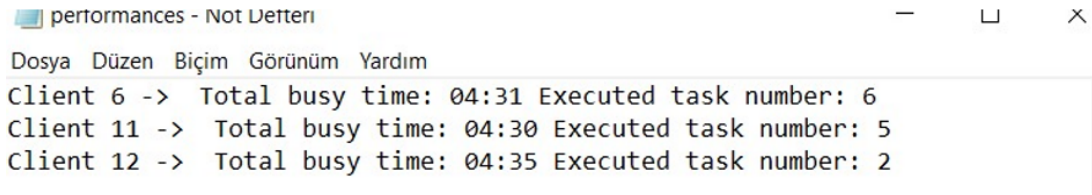


Figure 4.11: Client Performances

4.4.3. Client Status

File sending on the client side can also be monitored instantly via the interface, and the sent files can be seen at the end of the experiment. Of course, these files are deleted from the Client computer after they are sent.

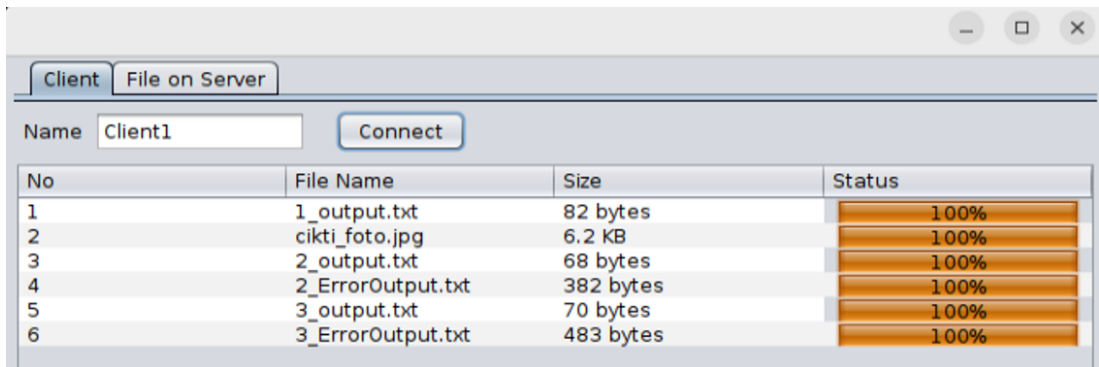


Figure 4.12: Client Status

5. TESTS

5.1. Test Scenarios

I created many different scenarios, taking into account the size of the files used in the tasks, task running time and the number of tasks parameters. I tried some of these scenarios in the project laboratory and showed the data I obtained in the 5.1.

EXPERIMENT	1 COMPUTER	1 CLIENT	3 CLIENT	7 CLIENT
Medium Duration 6 Tasks	10m 42s		5m 37s	4m 14s
Mixed 62 Tasks	14m 5s	15m 24s	7m 34s	2m 18s
Short Duration 147 Tasks	5s		32s	
Short Duration 42 Tasks		21s	13s	
Long Duration 14 Tasks - Small Files	105m			16m
Long Duration 21 Tasks - Large Files	152m			33m

Figure 5.1: Scenario Results

5.2. Test Results

- The factor that affects the experiment time the most is the size of the files used in the tasks. If the files used as input and output files are large files, this increases the experiment time considerably.
- Using the project may not be effective in tasks with very short working time. Because file transfer and server-client communication may take longer than running the task.
- In long-term tasks, choosing the number of clients proportional to the number of tasks is a good option to make full use of all clients. Otherwise, some clients may remain idle.

6. SUCCESS CRITERIA

6.1. Success Criteria of the Project

- At least 90% of the client computers in the project must perform the desired functions.
- Monitoring the stage of the clients in the experiment via the server with a delay of at most three seconds.

6.2. Completed Criteria

- In the tests performed, it was seen that all clients used worked as desired.
- The display of file transfers between the clients used in the experiment is transferred with a time difference of less than 1 second.

7. CONCLUSION

7.1. Successful Conclusions

- A task list can be created by reading the script and successfully defining the experiment.
- Successfully executing the loop tasks according to "Wait" keyword in the experiment.
- Successful execution of tasks on client computers.
- The output can be transmitted to the user as an error status or as a success status, depending on the possibility of positive or negative output of the running task.
- Successful instant transfer of the test status to the user.
- At the end of the experiment, data such as test duration and working performance of the clients are successfully displayed to the user.

7.2. Failed Conclusions

- There are bugs in the user interface that will not affect the operation of the program during client listing.
- If any client is crashed, the program does not terminate correctly because the task it is interested in does not finish.

7.3. What can be added to the project?

- Efforts can be made to reduce the time spent on server-client communication in the experiment.
- An extra file storage server or computer can be used to avoid keeping the server computer too busy.
- The user can be instantly shown which client is busy with which task.
- Necessary sections can be added to enable experiment definition that is compatible not only with Python but also with all programs.

BIBLIOGRAPHY

- [1] (), [Online]. Available: <https://docs.oracle.com/javase/tutorial/uiswing/>.
- [2] (), [Online]. Available: <https://github.com/DJ-Raven/socket-io-library>.
- [3] (), [Online]. Available: <https://socket.io/docs/v4/>.