המכללה האקדמית להנדסה
אורט בראודה
ORT BRAUDE COLLEGE

Software Engineering Department

ORT Braude College

Course 61766: Extended Project in Software Engineering

Efficient cloud storage, using Bin Packing algorithms

Aviv Mahulya 311289128

Amit Molek 318298262

Supervisors:
Mrs. Elena Kramer
Dr. Dan Lamberg

TABLE OF CONTENTS

Abstract *With the increasing size of cloud data centers and the increasing amount of data which stored in the "cloud", comes the requirement to decrease the amount of disks and storage devices. In the Bin Packing problem a sequence of items with different sizes must be packed into a finite number of bins with limited size. The objective is to pack those items into the smallest number of bins. In this project we will focus on online Bin Packing problems which mean that at any moment only the current and previous items sizes are known. We are going to calculate the average results of some Bin Packing algorithms and use it in order to store files on disks efficiently.*

Keywords: *Bin Packing, cloud storage****,*** *Any-Fit****,*** *Online algorithm, Next-Fits*

1.INTRODUCTION

Because of its applicability to large number of applications and because of its theoretical interest Bin Packing has been widely researched and investigated. In the classical one-dimensional Bin Packing problem we are given a list of items each with size between 0 and 1, and the objective is to pack them into a minimum number of bins with size of 1. In computational complexity theory, it is a combinatorial NP-hard problem. Even though the Bin Packing problem has an NP-hard computational complexity, optimal solutions to very large instances of the problem can be produced with sophisticated algorithms. From the nature of cloud storage, where we don't have any information on upcoming files, we need to handle every file immediately or fast enough. That's why in this project we will focus on online Bin Packing. In our problem the files of different size are supposed to arrive sequentially, and our algorithm has to select the algorithm that will perform best for the given input. This project main purpose is to store files on cloud, while using the minimum number of disks, using the optimal algorithm for the given input. In order to do it we will analyze Any-Fit (First-Fit, Next-fit, Best-fit, Worst-fit) algorithms expected results and determined which algorithm to choose.

2.BACKGROUND AND RELATED WORK

2.1 The Bin Packing problem

In the classic version of the problem, first introduced by Ullman, we are given a list L of n items, with sizes S_i, i = 1, 2, ..., n, S_i [0,1] and a supply of unit-capacity containers, called bins. The problem consists of packing each item into a bin so that the sum of sizes of items in a bin does not exceed the bin capacity, and the total number of bins used is as small as possible. As we can see bin-packing is a minimization problem.
Bin-packing has many variants as this abstract problem models a large variety of real world problems [5], such as minimizing the number of machines necessary for completing all tasks by a given deadline, called machine scheduling problems, and in our case, storage allocation problems, where we want to minimize the necessary containers for storing all the items. It is widely known that bin-packing is NP-hard and thus can be resource intensive task, that's is why many algorithms do not guarantee to find an optimal solution but find near-optimal solution in polynomial time. Many of the algorithms depend on the way the given input is read, one item at a time or all the items are known from advance, these algorithms are part of algorithms called Online and Offline algorithms, respectively.
Online Algorithms [4], Bin Packing algorithm is called *online* if it is given the items from L one at a time and is must handle each item into a bin immediately on arrival. A new a item is packed according to the packing and sizes of items that have already arrived before. There is no information in advance of the following items to come. Since online algorithms do not know the future items, they work in real-time environments where packing must be immediate and very fast. This is the type of algorithms that will be using in this paper.
Offline Algorithms, as opposed to online algorithms, have complete knowledge about the list of items and can apply strategies that take to account the future items. Offline algorithms have a variant called *Restricted Offline Algorithms*. In this variant the input is seen as a sequence that is known in advance.

In this variant, the items are given as a list, and the packing must be immediate according to the list, but the algorithm has complete information about the list of items.
In bin-packing there can be additional constraints, such as *Cardinality constraints*, a parameter i bounds, from above, the number of items that can be stored to each bin. It gives us a more accurate model for packing, for example, in a data center where it usually stores only a constant number of files per container.

2.2 The algorithms of Bin Packing

We will describe some algorithms in this field that are being studied widely. The algorithm Next-Fit keeps only one open bin at any time, and if the next item cannot be packed into the open bin, then the bin gets closed, and this item is packed into a newly opened bin. In the case of First-Fit and Best-Fit opened bins do not get closed, the next item is always packed into the first bin where it fits, into a bin with the highest level where it fits, respectively. And if there is no such bin, the item is packed into a new bin.
Another algorithm is the Worst Fit, the algorithm selects the largest possible free space that the item can be stored on. The generalization of First-Fit, Best-Fit and Worst-Fit algorithms is called Any-Fit algorithms, these algorithms allow to pack the new item into any open bin where it fits. The item is packed into a new bin if and only if there is no bin where it can be packed. The name of the algorithms are abbreviated as NS, FF, BF, WF respectively [1]. Another algorithm called Harmonic (HARMk, k=3) was first introduced by Lee and Leah [3]. The algorithm is based on the idea of classifying each item by size first and then pack it according to its class. HARM_k splits the interval (0,1] into to 2 subintervals forms. The first form is $(\frac{1}{i+1}, \frac{1}{i}]$, i=1, ..., k-1 and one final subinterval $(0, \frac{1}{k}]$. Where each bin will hold only items from the same subinterval. The packing is done using Next-Fit.

2.3 Evaluating performance

The quality of the online algorithms is usually evaluated using competitive analysis. When we discuss the performance of the online algorithms, we use the term *competitive ratio*. The idea of competitiveness is to compare the output generated by the online algorithm to the output produced by an optimal offline algorithm. Competitive analysis is a strong worst-case performance measure is the sense that a competitive algorithm must perform well on all inputs.
The competitive ratio is a standard measure for evaluating the quality of an online algorithm. For an algorithm ALG and a input L, let ALG(L) denote the action of the algorithm on L and its cost for this input, where cost is the number of bins used. An optimal offline algorithm, which uses a minimum number of bins for packing the items, is denoted by OPT. OPT(L) also denote the number of bins that OPT uses for a given input L. The algorithm is r-competitive if $ALG(L) \leq r \cdot OPT(L)$ for any input.

2.3.1 Our approach to evaluating performance

Previous researches used only the lower and upper bounds for each algorithm in order to compare algorithms results. D´osa and Sgall [2] have shown that FF, BF $\leq 1.7 \cdot$ Opt and that i the best bound for the worst-case behavior of Any-Fit algorithms.
Worst-case performance measure [5] will give you a good understanding of what to expect from the algorithm, but most of the time it represents extreme cases for the algorithm. For average-case we need a different way of measure. In the Bin Packing problem we want to know how many bins will be open for a given average-case input. The expected value [6] of a random variable, marked *as E,* is the average of the values that the variable is expected to be, weighted by the probabilities of obtaining the different values.
Our desired measure is one that will give us an approximation of how many bins an algorithm is expected to open. We define $\bar{E}$ as the expected value of the count of open bins, that will give us

exactly what we desire. If we can calculate $\bar{E}$ for each algorithm, we can get a good approximation of what algorithm will open fewer bins for the given input. We calculate $\bar{E}$ as follows:
For our calculations we assumed that the item's size in bytes distribute uniformly in range [1, 2, 3, …, M]. Where M is the size in bytes each bin can hold.
Let X(n,m,i) be the probability that exactly i items will be in the first bin, where m is the size of the first bin and n is the length of the input. For each algorithm X defined differently:

First Fit

$$X(n,m,i) = \frac{1}{M}\left(\sum_{j=1}^{m} X(n-1,m-j,i-1) + \sum_{j=m+1}^{M} X(n-1,m,i)\right)$$
$$X(n,m,0) = (1-\frac{m}{M})^n$$
$$X(1,m,1) = \frac{m}{M}$$

Next Fit

$$X(n,m,i) = \frac{1}{M}\left(\sum_{j=1}^{m} X(n-1,m-j,i-1)\right)$$
$$X(n,m,0) = \left(1-\frac{m}{M}\right)$$
$$X(1,m,1) = \frac{m}{M}$$

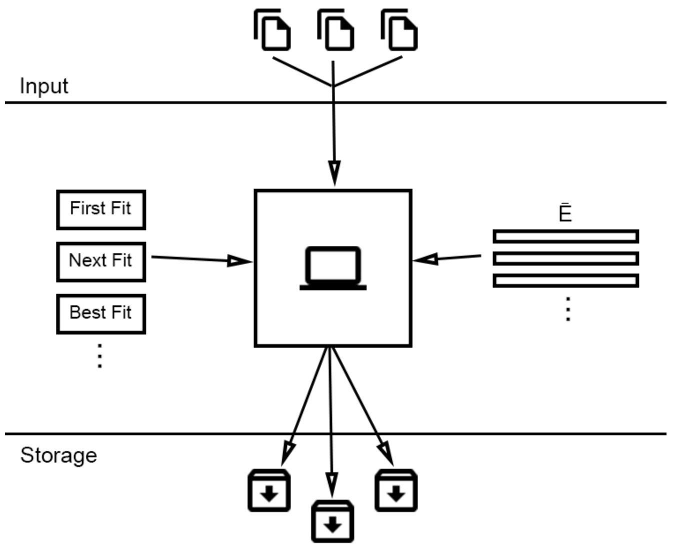
Let Y(n,k) be the probability that exactly k bins will be open. Using X we can calculate Y:

$$Y(n,k) = \sum_{j=1}^{M} X(n,M,j) * Y(n-j,k-1)$$
$$Y(0,0) = Y(1,1) = 1$$
$$Y(1,k) = 0\,, k \neq 1$$
$$Y(n,0) = 0\,, n > 0$$

Finally, we can define $\bar{E}$ as:

$$\bar{E} = \sum_{j=1}^{n} Y(n,j) * j$$

3.SYSTEM DESIGN



The system is constructed by a client and a cloud server, where the client can upload multiple files to the server and the server will decide in what disk (HDD, SDD, Tape, …) to store the files with fewer disks, or as we say, bins open.

3.1 Input

The system gets files or a single file as input with maximum size of M bytes. The user can upload any type of file format.

3.2 Storage

After the file or files were uploaded to the cloud (server), the server compares the different algorithms to check for the optimal algorithm, where the optimal algorithm is the one that will store the files in less bins for the given input and determine what algorithm to use to store the file or files.

3.3 Pre-Calculations

Using expected value of the count of open bins, or $\bar{E}$, we can compare how many bins each algorithm is expected to open. For example, if a user upload N files we can check that First Fit is expected to open p bins and Next Fit is expected to open q bins, if p<q, than we can determine that First Fit is the optimal algorithm for this input.
With that in mind we pre calculate $\bar{E}$ for each algorithm and hold that data in fast access where we can access and compare the values of each algorithm quickly to decide what is the optimal algorithm.

3.4 Parameters

$\bar{E}$ is depended on several parameters that we can change to tweak the system. Meaning
the system can be expanded and farther optimize for different values of parameters like: maximum bin size, item’s size distribution and more.

3.5 The algorithm

As we explained before, using the pre-calculated values of $\bar{E}$, we can determine the optimal algorithm for the given input.

$$MIN\{\bar{E}_{First-Fit}(n), \bar{E}_{Next-Fit}(n), \bar{E}_{Best-Fit}(n), \dots\}$$

Furthermore, the system can hold multiple calculations of E for different values of the parameters, to try and represent the input as close as we can, to make E more accurate. Meaning we can access the pre-calculations that was done for the input parameters, or at least close to the input parameters.

$$MIN\{\bar{E}_{First-Fit}(n, File\ size, Bin\ size, \dots), \bar{E}_{Next-Fit}(n, File\ size, Bin\ size, \dots), \dots\}$$

4.EXPECTED RESULTS

In this project, we intend to create a personal cloud server storage, that will obtain a noticeable improvement compared to a system that uses one algorithm to store the files.
We will analyze the real-world results against our approach of evaluating performance (E) and expect reasonably close results, also the comparison will help us to validate the approach. Furthermore, hopefully we will be able to further explore the subject and enhance the system.

5.PRELIMINARY SOFTWARE ENGINEERING DOCUMENTS

5.1 Requirements (Use Case)

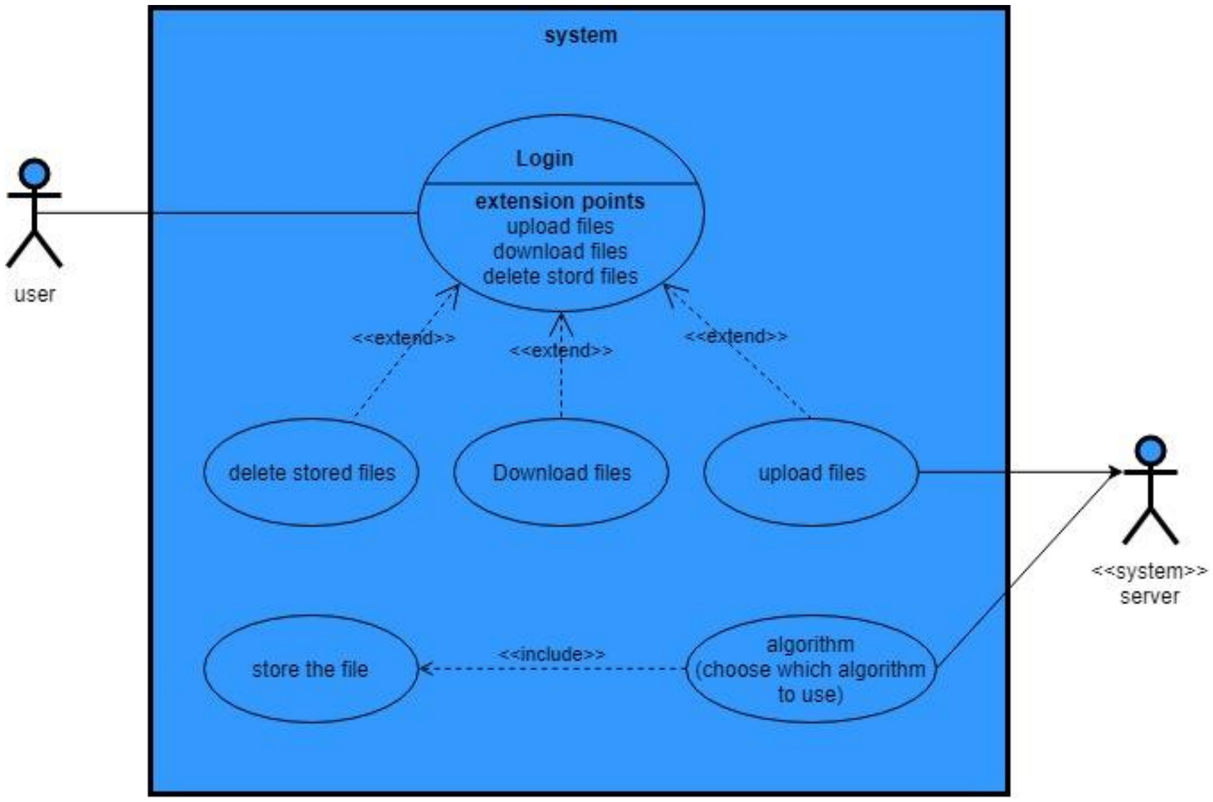


Fig. 1: Use Case Diagram

5.2 Design (GUI, UML Diagrams)

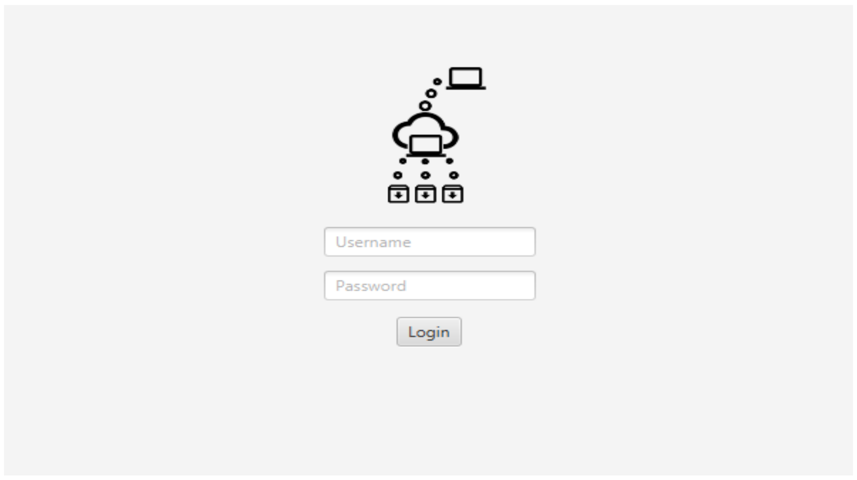
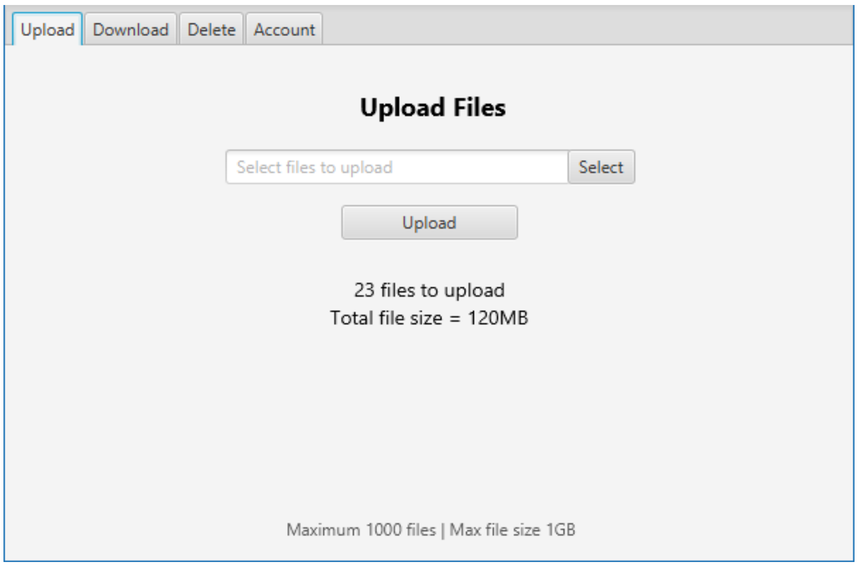


Fig. 2: Login screen



Upload Files

23 files to upload
Total file size = 120MB

Maximum 1000 files | Max file size 1GB

Fig. 3: Upload file screen

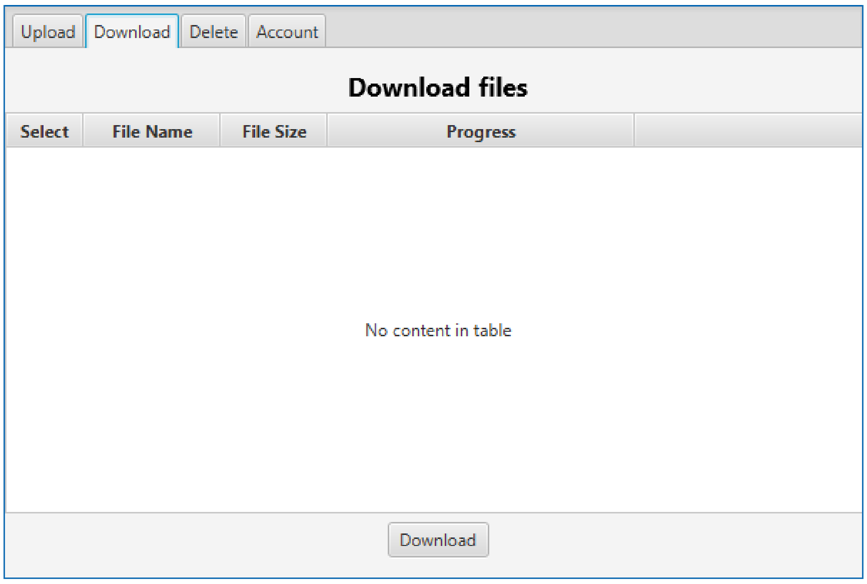


Fig. 4: Download file screen

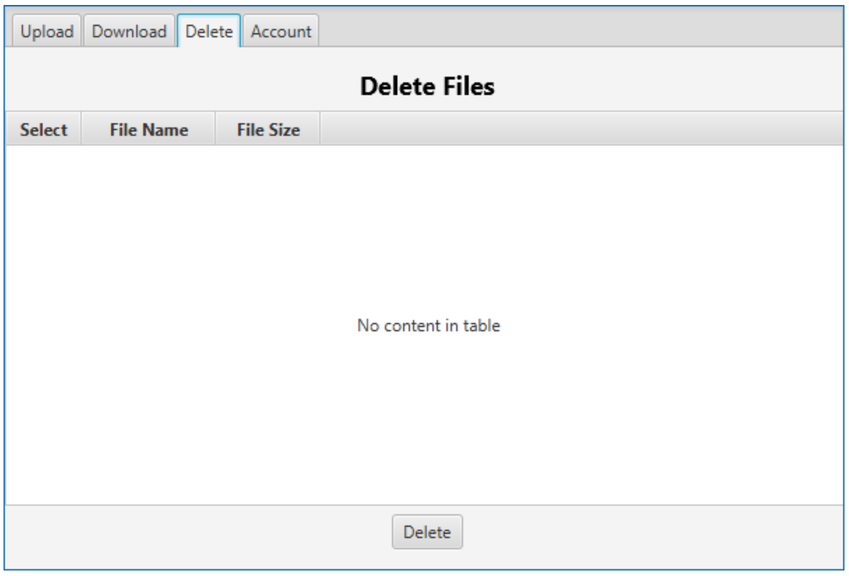
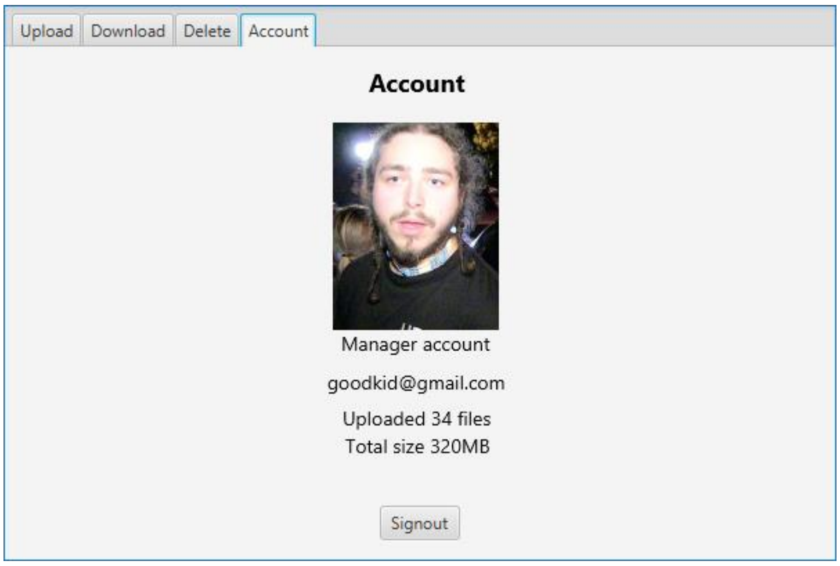


Fig. 5: Delete files screen



Account

Manager account

goodkid@gmail.com

Uploaded 34 files
Total size 320MB

Fig. 6: Account screen

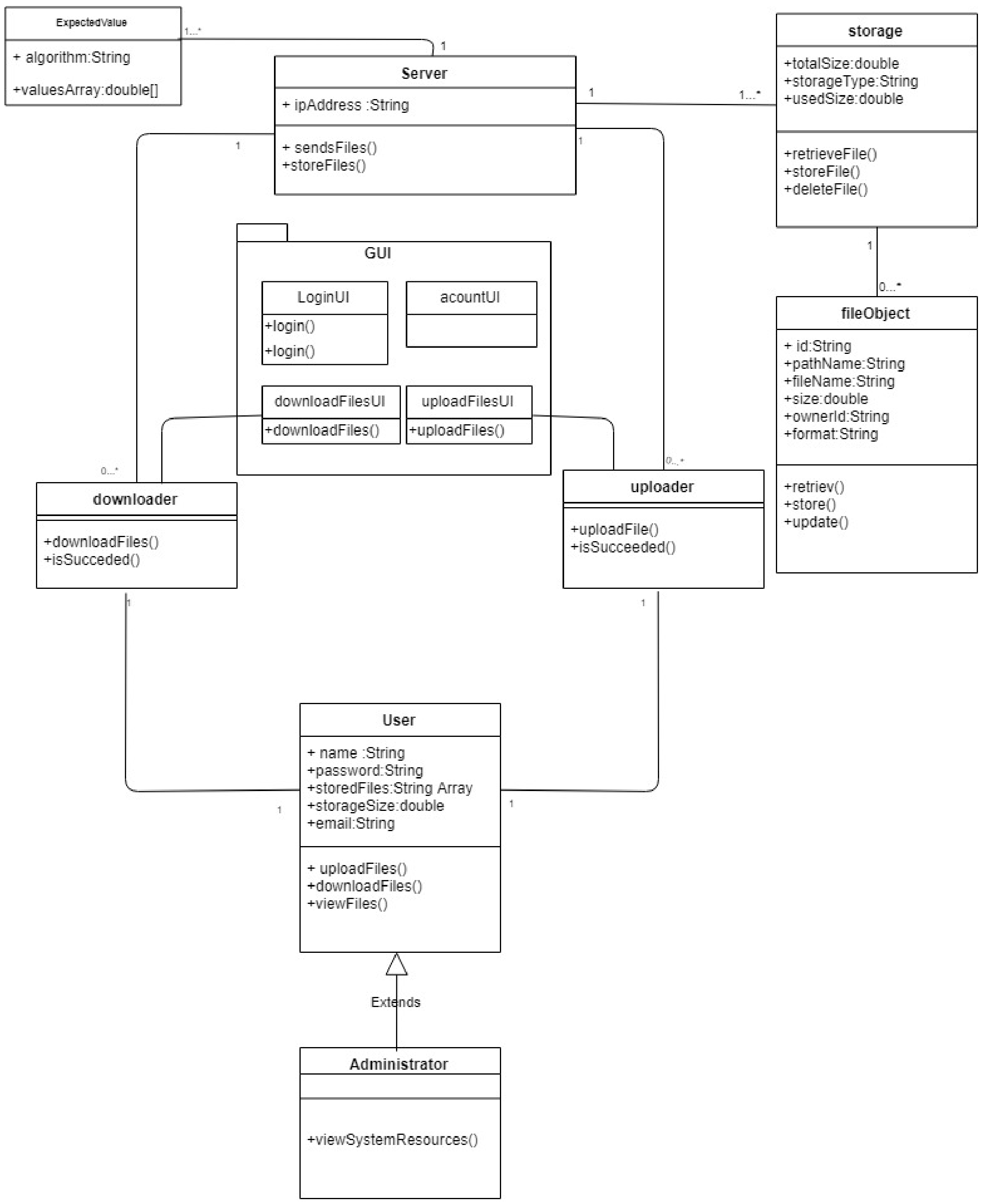


Fig. 7: Class diagram

5.3 Testing plan

In order to check out the system performance we will run the program on some significant input:

Test No.	**Test subject**	**Expected result**
1.	login with correct details	files manager screen will be displayed
2.	login with incorrect details	error message appear
3.	choose 1 file and press “upload”	“file uploaded successfully” prompt
4.	press “upload” without choosing file	error message appear
5.	Try to upload too big file	error message appear
6.	see the stored files (press “see my files”)	list with the stored files will be displayed
7.	download stored file	file will be downloaded to computer
8.	go back to upload files window(on my files press “back”)	the default page (Upload files)screen will be displayed.
9.	choose many files and press “upload”	“N file uploaded successfully” prompt(N is the number of files)
10.	account sign out	client signs out the account and return to login page
11.	select N files to upload with total size of M	number of file indicator will show “N files to upload”, total size indicator will show “Total files size = M”
12.	select 0 files to delete, and press “delete” button	error message appear

Table 1. Testing plan

6.REFERENCES

[1] L. Epstein, M. Favrholdt, and J.Kohrt, "Comparing Online Algorithms for Bin Packing Problems", Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark
[2] Jiri Sgall, " Online Bin Packing: Old algorithms and new results", Computer Science Institute of Charles University, Faculty of Mathematics and Physics.
[3] Leah Epstein, "Harmonic Algorithm for Bin Packing",Department of Mathematics,University of Haifa,Haifa,Israel
[4] Janos Balogh, Jozsef Bekesi, Gyorgy Dosa, Leah Epstein, Hans Kellerer, Zsolt Tuza, “Online Results for Black and White Bin Packing”, Springer Sceience, Business Media New York
[5] E. G. Coffman Jr., M. R. Garey, D. S. Johnson, “Approximation Algorithms For Bin Packing: A Survey”
[6] MIT OpenCourseWare, “Mathematics for Computer Science Fall 2010”