Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: rendt

High-Level Design Report

Project Group Members:
Cenk Er 21600937
Huseyn Allahyarov 21503572
Ibrahim Mammadov 21603109
Mahammad Shirinov 21603176
Nurlan Farzaliyev 21503756

Supervisor: İbrahim Körpeoğlu
Jury Members: Hamdi Dibeklioğlu and Özcan Öztürk

Analysis Report
December 31, 2019
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

High-Level Design Report

Project Short-Name: rendt

1 Introduction

With the ever-growing advancements in computer algorithms, machine learning tools, and with the availability and accessibility of such tools, distributed computing and cloud computing systems have become extremely widespread, to the point of almost being a necessity.[1][2]

The biggest players in these fields are currently Amazon (AWS), Microsoft (Azure) and Google (Google Cloud). What's common among these providers is that they all have large centralized networks of nodes somewhere in a server farm (or several farms), and they provide users with computing power and other services such as storage using parts of that network.

The fact that all the nodes belong to one party and are hosted in one or a few dedicated locations gives them great reliability and ability to offer good pricing. However, since there are few providers and millions of users, only the few big players make revenue. Also, the amount of computing power that is available is dictated by the sole provider.

With Rendt we propose an alternative solution to this vast need of distributed and cloud computing power, where multiple parties can offer their machines and get paid in return, while users will still have access to the computing power that they need for their projects and experiments. Similar solutions have been used in the academic community to solve problems like the decomposition of natural numbers as a sum of three cubes,[3] but they are voluntary in nature and have no commercial use.[4]

1.1 Purpose of the system

Even with all this need for computing power, there is still a large amount of idle or underused machines, which are mostly computers that belong to personal users or small companies that don't have distributed computing infrastructure. Rendt is a platform where these idle resources can be shared with people who need them and turned into profit. Users can choose a resource on their machine that they want to share, choose a type of task they are willing to share it for, and possibly state their price. Then, other users in need of that particular resource will see different offers for their query, choose the one they like and start using resources. After the tasks are run and results sent back to the task issuer, the payment will be made and transaction will be completed.

Using Rendt, somebody with powerful GPUs in their laptops who may not need them for personal use, at least not all the time, can turn it into profit, and dually, somebody who needs GPUs to run get some work done (train neural networks) but doesn't have access to any can look for people

offering their resources and have access to them without owning any GPUs. Rendt is going to play the role of a match-maker and regulator in such transactions. It will host users that share their resources, along with detailed information about their computing power, types and availability of resources and other information, for example, the time the person is willing to lend their resources or the history of the tasks they have successfully (or unsuccessfully) completed.

Rendt will be used for remotely running different kinds of tasks, such as training neural networks, rendering videos, or plain mathematical (CPU-heavy) calculations. This compartmentalization will help users better find suitable offers, since machines best suited for a particular kind of job will be tagged by that category by the host themselves. Additionally, Rendt will support distributed computations[5] as well: if the user's task is parallelizable[6], they will be able to request more than one node for their task and run it on them and then receive the single result back, as if running their task on some centralized cluster.

When a user issues a task and selects a node(s) to run it on (or leaves the selection to the system), the task goes to our central server, where a match is registered, and the transfer of the needed files (data, binaries, scripts) is initiated (e.g. by passing the files from issuer to the central server and from there to the host, by P2P[7] file transfer etc.). At this stage, a part (or the whole) of the payment the issuer has agreed to pay is deducted from their account and held by Rendt. The host node(s) start running the task(s), and once done, transfer the result files back to the issuer. Then, the host receives the payment and the transaction is over.

1.2 Design goals

1.2.1 Performance

To provide native performance, the system will be implemented with a high level language such as C++ to execute the tasks natively without any performance compromises. Virtualization is expected to be used to provide security, but should not decrease performance. With wide virtualization support in majority of CPUs, performance drop should be the case.

1.2.2 Compatibility

Rendt should be a cross-platform application to support a wide variety of systems and OSs. To tackle this issue, we intend to use CMake[8] with C++ which can compile a C++ program to run on Windows, MacOS and Linux systems. With the compatibility provided, rendt will be available for the use of a wide variety of users with different hardware, OS and configuration.

1.2.3 Maintainability

Rendt will utilize a server to serve the users and renters. Server will be responsible from payment, reliability rating decision, rental requests and responses, file transfer and displaying available PCs. Server will be

maintained by us. Maintenance covers server rental payment, network issues with the server and providing the system online.

1.2.4 Security

Rendt will be providing secure payment system, execution process and execution environment. To provide a secure payment system, money transfer will be sent and hold until the execution ends. Renter will get paid only after the execution finished successfully. To provide a secure execution process, rendt will operate in the background without giving away any information about the task to provide confidentiality for the user. Rendt will be operating via virtual machine (VM) or a technology alike to provide a secure execution environment. VM will provide the necessary security to avoid getting attacked in the renter's end. Any damage will be happening in the virtual environment.

1.2.5 User interface

UI should be straightforward and a user without an advanced level of technical knowledge of computers should be able to operate the rendt app and be able to use it for renting or using rental PCs. To provide such a straightforward system, VM initialization and installation should be handled by either the rendt installation or with some automated or manual process. This is an ongoing process.

1.2.6 Scalability

System should be able to serve as many requests as possible. With the increasing number of requests and users, system should be able to serve without any compromises. To tackle this issue system should be able to serve with minimal resource allocation per request.

1.2.7 Reliability

Rendt should be a reliable source for the users that need better performing hardware. For the rendt's side, server should be up and running, serve the requests as necessary, provide the security, scalability and compatibility. For the users/renters' side, user reliability rating should be considered before rentals alongside the reviews from the past experiences of the user.

1.3 Definitions, acronyms and abbreviations

RENDT: Remote Execution and Distribution of Tasks

Renter: User looking for a leased computer to be rented

Leaser: User leasing his/her computer for rental

Remote: Distant as in computer(s) not nearby

Execution: Compiling and running a task to get a result

Distribution: Running different tasks of the program on multiple leased computers for performance

Task: Unit(s) of program to be run and executed

Virtualization: A virtual environment for tasks to be executed safely

VM: Virtual Machine (or Environment)

OS: Operating System

Cross-platform: Compatible with multiple platforms and devices

1.4 Overview

Rendt is aimed to be a cross-platform desktop application that runs in the background without the interference of rented PC's user. The project should run natively to avoid compromising the performance and security should be provided to protect the rented PC from any malicious code or users. Security is an ongoing research mostly aimed at virtualization technologies. Another point to be considered is the secure transfer, compilation and execution of the tasks without giving out any details about the task to the renter to provide confidentiality. Rendt's main scope is general code execution at the moment. Code execution may be very power demanding depending on the type of the task to be executed. For tasks such as image analysis and computer vision, code execution depends solely on the hardware and performance of the CPU and, in some specific cases, GPU and execution time is inversely proportional with the hardware performance. Users, especially renters, should be informed of the drawbacks of performance rental and the decrease in CPU and GPU lifetime as a consequence as well as the responsibility of renting. Secure and safe payment and refund systems should be provided for both renters and users. Price of rentals should be calculated in terms of the hardware and performance. Renters will be rated in terms of reliability, in other words, if a renter does not take the responsibility of renting and fails to finish execution of the task, a bad reputation is to be expected. Rendt's users scope is expected to be large and unlimited and to accomplish this goal, the system should be straightforward to use and technical knowledge of the user and requirements should be minimal.

2 Current Software Architecture

To start with, we researched about similar projects to see if such a system is feasible. We came across BOINC[9] which is a system dedicated to volunteering CPU cycles for scientific research for institutions such as SETI - Search for extraterrestrial intelligence[10]. It is used by research institutions to use volunteers' computers for research purposes. However, it is limited in the sense that only a few privileged users can run jobs, and the volunteers don't profit from this interaction.

With this research and the suggestions from our innovation expert, we started to research the different technologies to implement such a system. First, we started to gather information about WebAssembly which is a web-based technology that our innovation expert suggested us to look into. WebAssembly[11] is a fairly new technology that allows codes written in high level languages like C/C++, Rust to be compiled into web. With WebAssembly we could implement a system that doesn't require any installation and supports all the platforms. However, with extensive research into the matter, it turned out WebAssembly would not be a good choice for such a system. The main reason for this is that WebAssembly uses Enscripten[12] under the hood which converts the code written in high level languages to vanilla javascript. However, Enscripten is not able to convert all the code natively and performance decrease is to be expected because of its usage of web workers rather than cpu threads. Finally, WebAssembly and Enscripten cannot natively support multithreaded code and low level libraries such as OpenMPI[13] is not supported by this technology.

Finally, we came to a conclusion that a program written in a high level language is mandatory and it should run cross-platform with native performance and unfortunately, download and installation is necessary. With all this research, we finally settled for implementation with C++ that runs cross-platform with the help of CMake.

3 Proposed Software Architecture

3.1 Overview

In this section a detailed system design is presented. One of the important points of this section is what components the application comprises, how they interact with each other and how all the different modules and components fit together. This is shown using diagrams in the immediately following subsection, and also elaborated on in section 4. The next subsection outlines how these modules map on hardware instances. After that, design decisions made so far are discussed, regarding data management, access control, global software control and boundary conditions.

3.2 Subsystem Decomposition

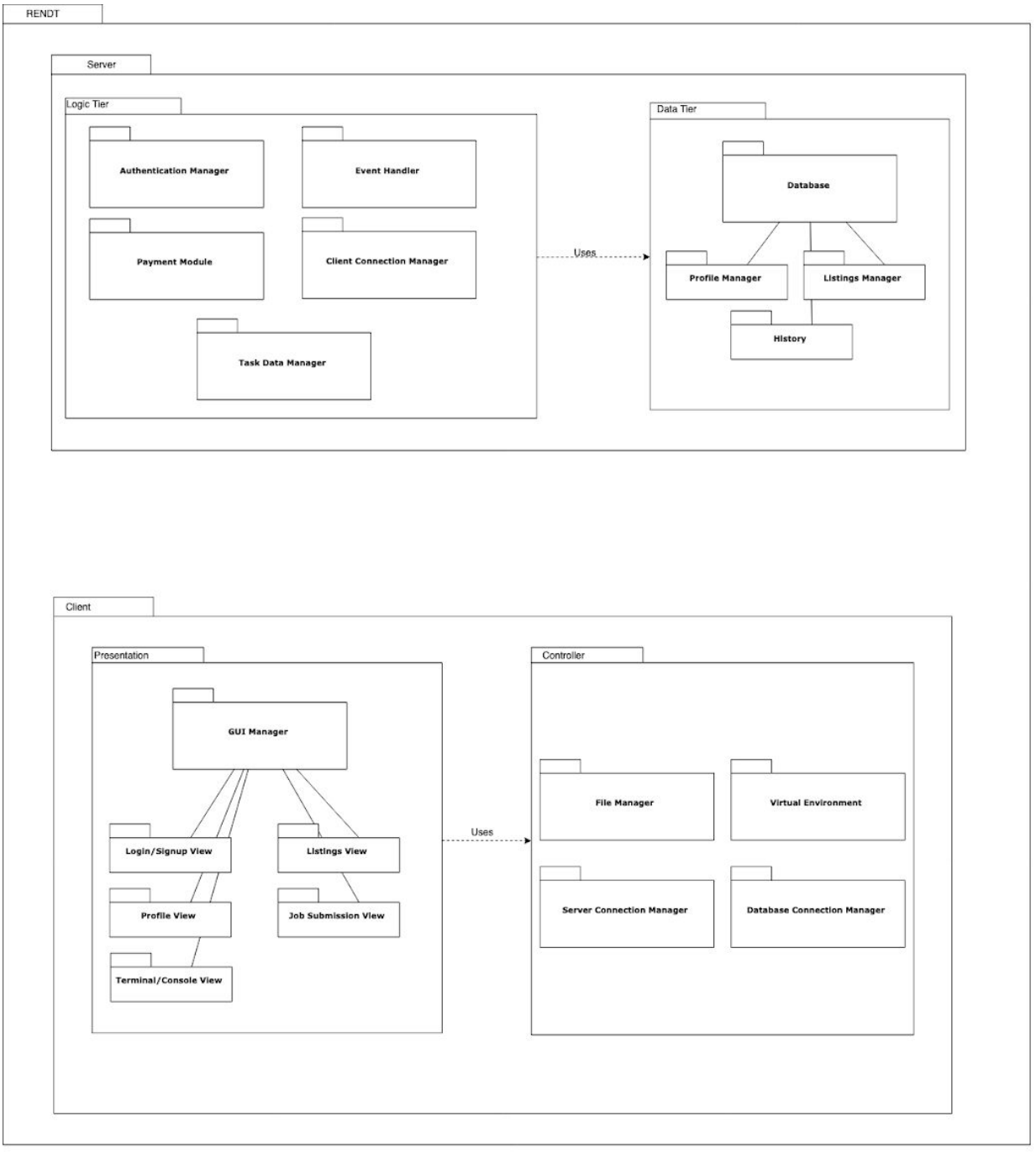

Figure 1: Subsystem Decomposition

rendt, at the highest level, follows a server-client architecture. The client side is going to be a desktop application allowing users to interact with the server (e.g. see available leasers, send/receive data) or submit jobs. It will also oversee the execution of the task in the receiver's machine, and send relevant data back to the server. The server side, in turn, will assume the role of delegating jobs. It will authenticate users, provide renting users information about available leasers, establish the connection between renter and leaser users and carry out the payment in the end. It will store some internal data on users, listings and past transactions, and

present these to users when required. More detailed information about the services of different modules within the components is given in section 4.

3.3 Hardware-Software mapping

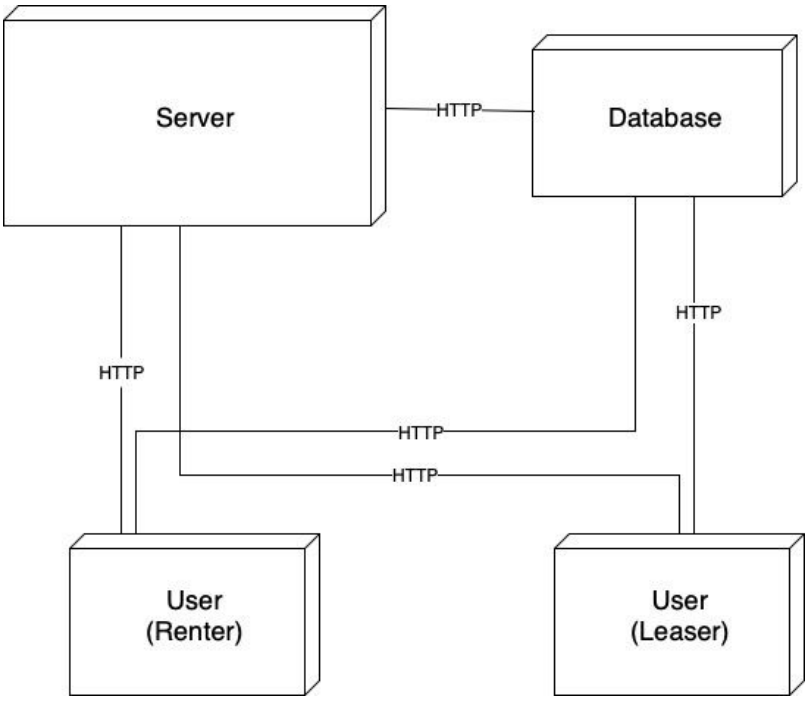

Figure 2: Component Diagram

Server side of the system consists of a Database system and a Server system. Inside the server there is another database to hold information about clients. The other database system is for file logistics that happens between clients when a job is sent for execution. When users should send for execution and/or send executed tasks back, our server allocates space on this database and tells the clients about where they should download the data and where to upload it.

Our clients will run the application on any OS of their choosing on their private computers. Being connected to the network is important, especially during the time when the file is being sent. A client (renter) will send a file to be processed to another client (leasers) which will process the file and send it back. Every send operation will be by a HTTP.

3.4 Persistent Data Management

Persistent data in our application includes all user information (both renter and leaser) like password and username, all technical information related to computer version and characteristics, reliability points which all users are going to have, information about online users and credit card information (upon user's request). The server will also store a record of past transacitons between users.We are planning to use SQL engine for

our database creation and manipulation, for both databases, one residing in the server and the other separate one connected to the server. The data transferred between the users will pass through this separate larger database, and will be kept there only temporarily, until the receiving party successfully downloads it.

3.5 Access Control and Security

In order to use rendt users must create an account in the first place. There will be 2 different kinds of user types in this system, the renter and the leaser. The users are not required to sign up with multiple accounts in order to submit and receive jobs. Having a single account will be enough to be able to control transactions, since the users will be provided with a functionality which will allow them to switch in between user modes.

Since each user will have their own account they will be able to access the data that belongs only to them. And since the security is one of the priorities in this system even the leaser will not have any kind of access to the renter’s data.

Our server will be web based, where all the transactions will take place. However, for security purposes, leaser will have to install and download an application which will run a virtual machine where all the processes will be executed. By providing this sort of system we are disabling the leaser from accessing or do any kinds of operation on the renter’s data.

3.6 Global Software Control

The server of our application will have an event-driven control system to control the transactions made between the users. Renter will request the server to send the data to the leaser. The server will generate request to leaser to find whether the leaser is ready to start the operation or not, if the result will be positive the delivery process will start. Server will get the renter's data and send it to the leaser with scripts for downloading all needed APIs and libraries for the execution. After the execution is complete the results will be saved in an appropriate format and sent back to the server. The server will request payment from the renter and after the execution of payment process the data will be sent to the renter.

3.7 Boundary Conditions

3.7.1 Initialization

As mentioned earlier in order to benefit from rendt users have to sign up first. Users need to have some kind of browser on their computers so that they can go to the respective web-site and sign up. Users are required to enter an email address, a username and password. Users will be signed up if all the requirements are met, such as if the entered email address is valid or not, if the username is available or not and if the password is strong enough. The validity of username and password will be checked when the users enter the respective information. But in order to check

validity of email address users will receive a verification code that will be sent to their email addresses that they have provided.

3.7.2 Termination

The users (both renter and leaser) can log out any time they wish, except the cases when they are in the middle of the process. The leaser can log out immediately after the execution on users computer was done and the renter can do the same after the payment procedures was done. If the renter leaves before receiving his data and the leaser before receiving his money, the application will save both in appropriate places (for money it is users balance, for code it is specific area in user's account which will be created for such cases)

3.7.3 Failure

The failure could happen if in the middle of transaction one of the users lose internet connection. First of all if this will happen constantly it will affect users reliability points so the other users know who they are dealing with. If renter leaves before money transaction was done the users will not receive executed files, they will be kept on the server for some period of time (so the renter can handle internet connection problems and proceed with payment afterwards) after which it will be erased from the server. If the leaser leaves until the code was executed and sent to the server, the renter would be notified immediately and the operation would be canceled.

4 Subsystem Services

4.1 Server

Server is the component in rendt's architecture that is responsible for overseeing job allocation and delivery of results, as well as payment. It is organized into a Logic Tier and Data Tier.

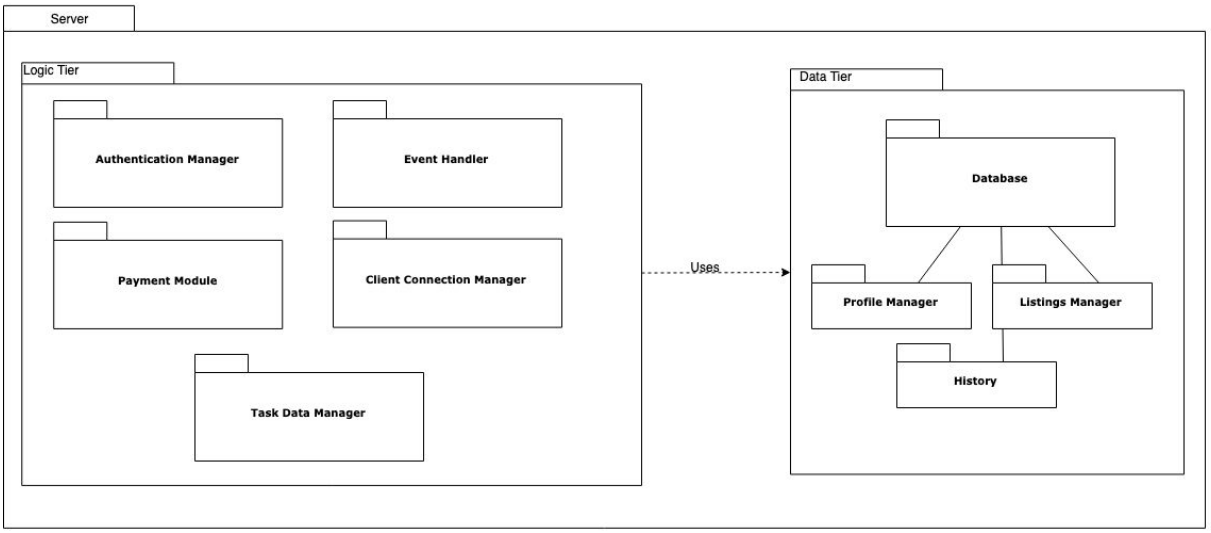

Figure 3: Server Subsystem

4.1.1 Logic Tier

The logic tier incorporates all the vital responsibilities of the server.

Authentication Manager: is responsible for authentication of users: creating of accounts and accessing existing ones.

Client Connection Manager: is responsible for establishing and managing network connections with client ends. Will receive requests from clients and deliver them to relevant components, and also send clients notifications/updates.

Event Handler: the main processing unit in the server. Will process client requests, interact with the Data Tier as needed and provide the results. Also responsible for establishing the connection specified by a renter with the specified leaser, and managing it to the end.

Payment Module: is responsible for the payment after the completion of a job.

Task Data Manager: is responsible for overseeing the data transfer between renters and leasers through the separate database component.

4.1.2 Data Tier

This layer is responsible for storing data about the user profiles, active listings and also past actions that have taken place on the server.

Database: The database internal to the server, used to store private information.

Profile Manager: Keeps information about user profiles.

Profile Manager: Keeps information about current and past listings (offers by the leasers).

History: Keeps past transactions between users and some relevant information about them, like duration, cost, outcome, reliability etc.

4.2 Client

Client subsystem consists of presentation (view) and controller. These are respectively the GUI and functionality managers of client side subsystem.

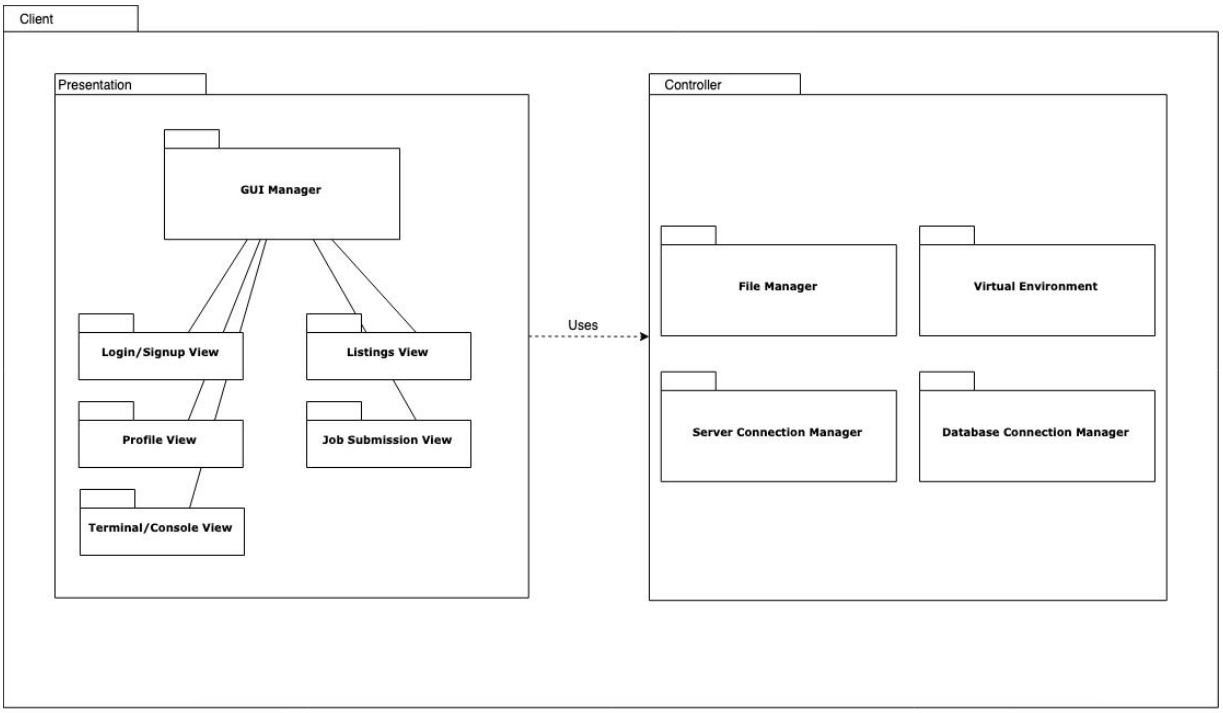

Figure 4: Client Subsystem

4.2.1 Presentation

Client side GUI elements are in a package called Presentation.

GUI Manager: This class creates main user interface which works with other view classes.

Login/Sign up View: This is the first page the user will see when they open the application.

Profile View: This view displays information about the user, e.g. username of the user, email, their computer specifications and their rating if the user is a leaser.

Terminal/ Console View: This view will create a console view to run/execute commands in bash.

Listings View: Displays the list of computers with some of the specifications are explicitly shown which are available at that moment.

Job Submission View: This view will show the computers detailed specifications that is chosen by the renter and a spot to drag/drop the file to be processed.

4.2.2 Controller

This controller package is for the main functionalities of the application.

File Manager: This class responsible for managing files that are being sent from renter to the leaser.

Virtual Environment: This class is responsible for running jobs on leaser clients that are sent by senders.

Server Connection Manager: This class will manage connection between the server and the clients.

Database Connection Manager: This class will manage connection with the database that will be used during sending and receiving of execution files.

5 References

Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[1] "What is distributed computing and what's driving its adoption?". Packt. https://hub.packtpub.com/what-is-distributed-computing-and-whats-driving-its-adoption/ (accessed October 14, 2019).

[2] "The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle". Dr. Brijender Kahanwal, Dr. T. P. Singh. International Journal of Latest Research in Science and Technology, Vol. 1, No. 2, pp. 183-187, 2012

[3] University of Bristol. "Sum of three cubes for 42 finally solved -- using real life planetary computer." ScienceDaily. www.sciencedaily.com/releases/2019/09/190906134011.htm (accessed October 14, 2019).

[4] "Seven Ways to Donate Your Computer's Unused Processing Power". Vice. https://www.vice.com/en_us/article/bmj9jv/7-ways-to-donate-your-computers-unused-processing-power (accessed October 14, 2019).

[5] Wikipedia. "Distributed Computing" https://en.wikipedia.org/wiki/Distributed_computing

[6] Wikipedia. "Parallel Computing" https://en.wikipedia.org/wiki/Parallel_computing

[7] Wikipedia. "Peer-to-peer" https://en.wikipedia.org/wiki/Peer-to-peer

[8] Wikipedia. "CMake" https://en.wikipedia.org/wiki/CMake

[9] Wikipedia. "Berkeley Open Infrastructure for Network Computing" https://en.wikipedia.org/wiki/Berkeley_Open_Infrastructure_for_Network_Computing

[10] Wikipedia. "Search for extraterrestrial intelligence" https://en.wikipedia.org/wiki/Search_for_extraterrestrial_intelligence

[11] Wikipedia. "WebAssembly" https://en.wikipedia.org/wiki/WebAssembly

[12] Wikipedia. "Emscripten" https://en.wikipedia.org/wiki/Emscripten

[13] Open MPI: Open Source High Performance Computing https://www.open-mpi.org/