Bachelorthesis

Development of a Web-Application
for executing Job-Scripts on an IBM-Mainframe
in the context of Education

Bachelorarbeit gemäß § 17 der Allgemeinen Prüfungsordnung vom 01.08.2008
im Bachelorstudiengang Informationsmanagement und Unternehmenskommunikation
an der Hochschule für angewandte Wissenschaften Neu-Ulm

Erstkorrektor	Prof. Dr. Phillipp Brune
Betreuer	Dr. Kevin Henrichs
Verfasser	Marc Morschhauser (Matr.-Nr.: 204041)

Thema erhalten	01.01.2020
Arbeit abgegeben	01.05.2020

Unterschrift des Studierenden

Unterschrift und Firmenstempel der Ausbildungsstelle

In association with:

1 Abstract

Hier kommt der/die/das Abstract.

List of Figures

2 List of abbreviations

IoT Internet of Things

I/O Input / Output

OS Operating System

ULE Ubiquitous Learning Environment

JCL Job Control Language

JES Job Entry Subsystem

OLTP Online Transaction Processing

LPAR Logial Partitions

LAN Local Area Network

FTP File Transfer Protocol

SSH Secure Shell

SFTP Secure File Transfer Protocol

Contents

3 Introduction

Big Data, Cloud, Blockchain, Internet of Things (IoT) - Digitalization is steadily proceeding and with it the amount of data that has to be processed every day. Many web-services at this point are dependent on flexibility in first place. Big server farms facilitate an unprecedented agility to operators of online-businesses referring to their resource-planning. Urgent situations with high data traffic can be handled by adding physical or also virtual servers within minutes. However there are services where minutes make the difference between millions of euros. Those services demand for information technology, which not only has a high data throughput but also ensures the highest availability. The IBM Mainframe in its newest construction provides an Input / Output (I/O) of 288GB per second while being available 99,999% of the time running. This makes the Mainframe a cornerstone in present finance-sector's IT-management. Due to it's security and it's high speed it also finds it's use in other big industries - thus also insurances, the aerospace industry or big retail companies benefit from this supercomputer.
But since the IT improved considerably over the last decades, the Mainframe was presumed dead and many companies discontinued training their staff to maintain this technology. Contrary to their expectations they are dependent on it till today and they will presumably be in the remote future. What is left is a big gap in the division of skilled professionals that is hard to close.
The Goethe-University in Frankfurt, Germany, attends to participate in closing this gap. Therefore the university procured one of the aforementioned IBM supercomputers to train their students dealing with it.

This Bachelor Thesis approaches a Front-End-Application, which will be running on a Linux Virtual-Machine but executing tasks on the mainframe's Operating System (OS) called z/OS. By building this bridge between the Linux VM and the z/OS, students will be able to get in touch with the Mainframe located in Frankfurt University through a web-browser and as a consequence experience how the system is working.

The resources for this activity are allocated by the *Talentschmiede AG* - a Frankfurt located IT consultancy which is in close contact to the finance sector and knows and cares about the gap of skilled professionals. Further it is supported by the *Academic Mainframe Consortium e.V.* - an association founded by Mainframe Experts which are also willing to acquire new educated staff in the division of Mainframe.

4 Related Work

While a malicious tongue once suggested that " the last mainframe will be unplugged in 1996 " , it reconsidered when IBM introduced it's latest version of it in 2008 [12]. Till the present day the mainframe is the backbone of the financial markets worldwide and just as important for other big industries[1] [11]. The view on the mainframe technology as an IT-dinosaur has to change and the need of adding the mainframe technology to the IS Curriculum in a wide range was overdue already 10 years before [18] [19] [4]. Sharma et al. analysed the need of Large System Education regarding to mainframe education and they are investigating " the academic response to the need for large systems specialists " [15]. In [3] A. Corridori addresses the concerns and the opportunites that come with adding mainframe content to universities curricula and also previews ways to do this. How the economy and with it, the labor market can benefit from it is stated in [16].

The potentials of present digital media used to educate in a wide range is discovered by Cope et al. [2]. It is described that "the learner's relationship to knowledge and the processes of pedagogy have not changed in any significant way" but through technology " the educational paradigm has changed ". Vicki Jones et al. also address the integration of modern information technology in everyday's life and the advantages of this change regarding education [8]. Here it is stated that " Adaptive learning can offer great advantages in providing students with specific and personalised knowledge as and when required ". While the term " Adaptive Learning " is responsive to the methods that are used to transfer knowledge [13] , " Ubiquitous Learning Environment (ULE) " describes an environment with the possibility of learning everywhere and anytime[6]. Hwang et al. emphasize the fact, that ULEs are an innovative approach for teaching complex topics [7]. To establish a ULE there are many technologies needed [14], this thesis is supposed to do a first step in this direction to learn on the Mainframe anywhere and anytime.

Kiefer, COBOL as a modern language [10]
Khadka, How do professionals perceive legacy systems and software modernization? [9]
Vinaja, 50th anniversary of the mainframe computer: a reflective analysis [17]

.....

5 An overview on the IBM Mainframe

Prior to entering the main issue of implementing the web-application, this chapter introduces a few mainframe terminologies to assure a full understanding of the steps that take place in the following chapters.
Because, while many people just take the easy way by calling almost every computer a *server*, and the term *mainframe* is often just used to point out that this is largest server in use, in this thesis the term *mainframe* describes the IBM Supercomputer which is capable of "supporting thousands of applications and input/output devices to simultaneously serve thousands of users" [5].
So the most common utilization of the mainframe is divided in two categories:

- Online Transaction Processing (OLTP), incl. web-based applications
- Batch Processing

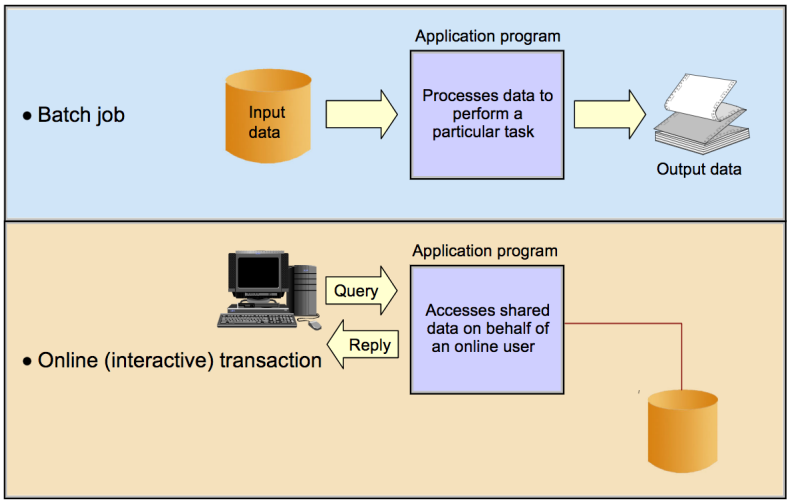

Figure 1-3 Typical mainframe workloads

Figure 1: Typical workloads

5.1 Online Transaction Processing

One of the core functions of many businesses is OLTP. For this case the mainframe serves a large amount of applications to make it possible to execute thousands of transactions in short time while handling not only a huge amount of different transaction types, but also to do this with many different users at a glance.
For end-users those transaction processes are often commonly known, while they appear in everyday's life, such as:

- Credit card payments in supermarkets
- ATM transactions
- Online purchasings

—- **Explain Online part and why not consolidated in this thesis** —-

5.2 Batch Processing

The other main function of the mainframe is processing data in a batch. Eventually terabytes of. These processes are generally done without much user interaction. A batch job simply gets committed and processes the data that is determined in the job-statement (further informations in chapter 7)

An equivalent concept can be found in a UNIX script file or a Windows command file, but a z/OS batch job might process millions of records.

—- **Explain deeper and bridge to System Operators** —-

5.3 Separation of duties

distinguish between:

- System programmers
- System administrators (for example, DBA, storage, network, security, and performance)
- Application designers and programmers
- **System operators (explain)**
- Production control analysts

5.4 Mainframe Operation Systems

z/OS
z/VM
z/VSE

Linux for zSeries
z/TPF

5.5 Virtualisation with z/VM

As an aid to consolidation, the mainframe offers software virtualization, through z/VM. z/VM?s extreme virtualization capabilities, which have been perfected since its introduction in 1967, make it possible to virtualize thousands of distributed servers on a single server, resulting in the significant reduction in the use of space and energy.
z/Virtual Machine (z/VM) has two basic components: a control program (CP) and a single-user operating system (CMS). As a control program, z/VM is a hypervisor because it runs other operating systems in the virtual machines it creates. Any of the IBM mainframe operating systems such as z/OS, Linux on System z, z/VSE, and z/TPF can be run as guest systemsin their own virtual machines, and z/VM can run any combination of guest systems.
—- Explain and show z/VM Constellation, LPARs etc —-
more in : `http://www.redbooks.ibm.com/redbooks/pdfs/sg247603.pdf`

6 Research gap

While the web-access to z/OS for OLTP is very established because of the usage, that is often conducted by end-users or computers, that are not located in immediate proximity to the server, the access to the JES-spool through web-applications is less common, since the system-operators are usually working in ultimate contact to the mainframe.
Certainly there are operators handling batch-processes through a web front-end due to flexibility and simplicity reasons. But while these applications are to simplify the process of managing the workload in daily-business, this thesis tries to establish an environment where the jobs have to be done in full amplitude, but on a test-system, implemented just for educational reasons.
This setup will have the ambitions of getting people into JCL quite quickly on the one hand and give them the opportunity to train their abilities and, in this way reduce potential uncertainties towards working on large-scale systems, on the other hand. With this method a pertinent and long-lasting learning effect is to be achieved in short time.

7 Dive into JCL

—- Important to understand JCL for accessing JES Spool?! —-

To get familiar with JCL, this chapter will give an introduction in how these Statements are working. There is talk about "Statements", because JCL is used to tell the OS what to do. Each Statement is an independent work unit, known as "Job" - therefore this language is called "Job Control Language". Each Job consists of instructions that are either typed in by an operator or they are stored and get transmitted to the computer.

7.1 Job Control Environment

So to understand how a job is executed, it is important to know which components are needed for this process.

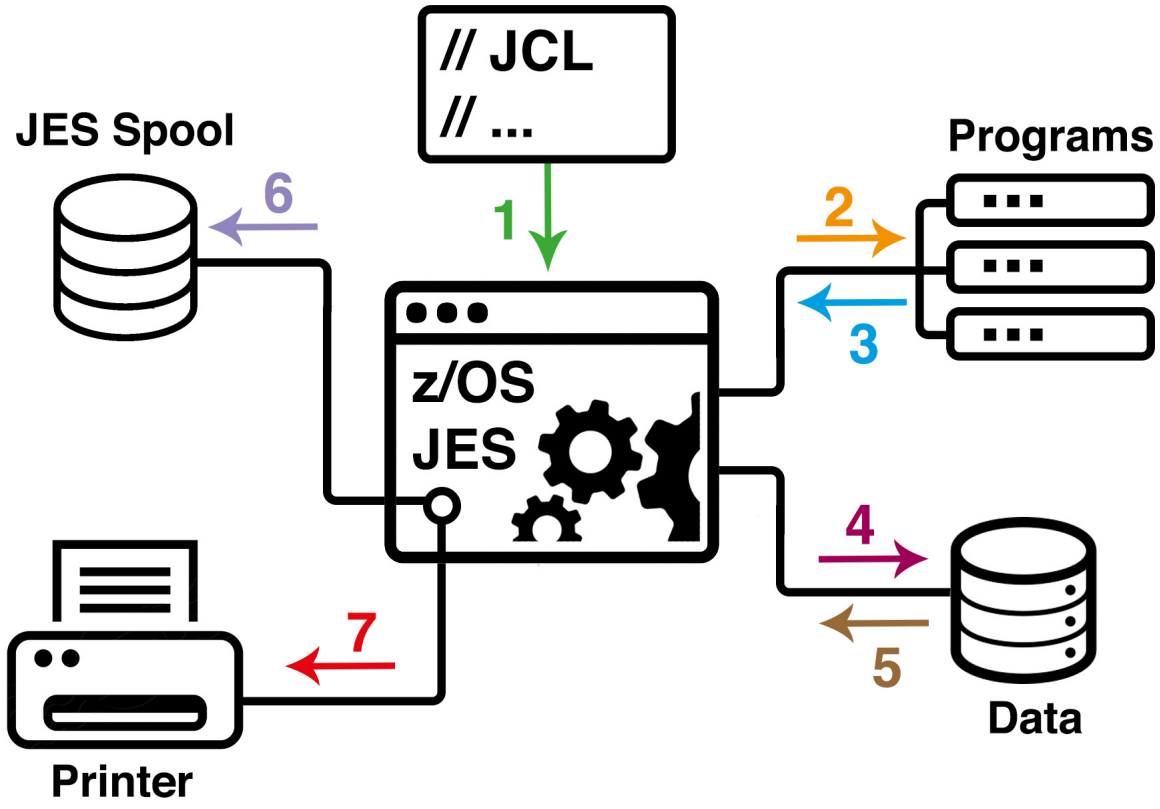

Figure 2: JCL Functionality, Source: IBM

On figure 2 you can see a job process described in 7 steps.

1. JCL submit

2. JCL requests program
3. Program gets loaded
4. Resources for program get allocated through JCL
5. Resources get provided to program
6. Program writes output to JES Spool
7. Output to gets transferred to printer as requested

The z/OS has written (hard coded) application programs which are not associated with any physical resources. Those programs are just names, which include internal file-names. Through JCL those programs can be opened for reading and writing during execution. The Job Entry Subsystem (JES) is there to evaluate and accept or not accept a job. If the syntax is right and the job is accepted the JES runs it on the OS and controls this process. The results get transferred to an output unit.

7.2 Job structure

-each statement 80characters
-JCL is introduced with //

7.2.1 EXEC-Statements

Within a job, there are working executions introduced through an "EXEC-statement". Every job needs at least one execution, but there can obviously be many more in addition. If a job has no execution it stops.

```
1 //STEP0001 EXEC PGM=IEBGENER
2 ...
3 //STEP0002 EXEC PROC=PRDPROC1
4 //STEP0003 EXEC PRDPROC2
```

7.2.2 DD-Statements

- SYSUT1
- SYSUT2
- SYSIN

- SYSPRINT

-JCL links the program file names with physical resources (e.g. data set names or unix file names).
–JCL is used to process programs in the background ("batch")
- And to process programs in the foreground("started task")
-JCL instruct z/OS -> Start / submit - JCL SUBMIT Statement will result in batch process of one or more programs (BACKGROUND)
- JCL START will result in FOREGROUND processing of a processing program

7.2.3 Background / Foreground jobs

Every Batch Job must contain JOB-statement & EXEC statement
->JOB statement highlight the beginning of a batch job & assigns a name to the job
JCL started tasks do not require a JOB Statement ->both have at least one EXEC statement -> marks the beginning of a job step , assigns name to the step & identifies the program or procedure to be executed in the step.

8 Access possibilities

In order to execute JCL-Jobs on z/OS from the Linux host, an access point is required to submit the data. This chapter is to reveal how the z/VM connects the Linux-host and the z-OS Target-system in this case and how this connection can be used to transfer data among them.
To enable the communication between different applications, a protocol is needed. The protocol that is used in the System Z is the **TCP!** (**TCP!**)/**IP!** (**IP!**), which is provided through the HiperSockets function. HiperSockets is a technology developed by IBM to enable high-speed communication between Logial Partitions (LPAR) with a hypervisor or between applications inside a z/VM as it is the case here.
You could imagine a HiperSockets-network like an internal Local Area Network (LAN), linking all partitions for internal communication.

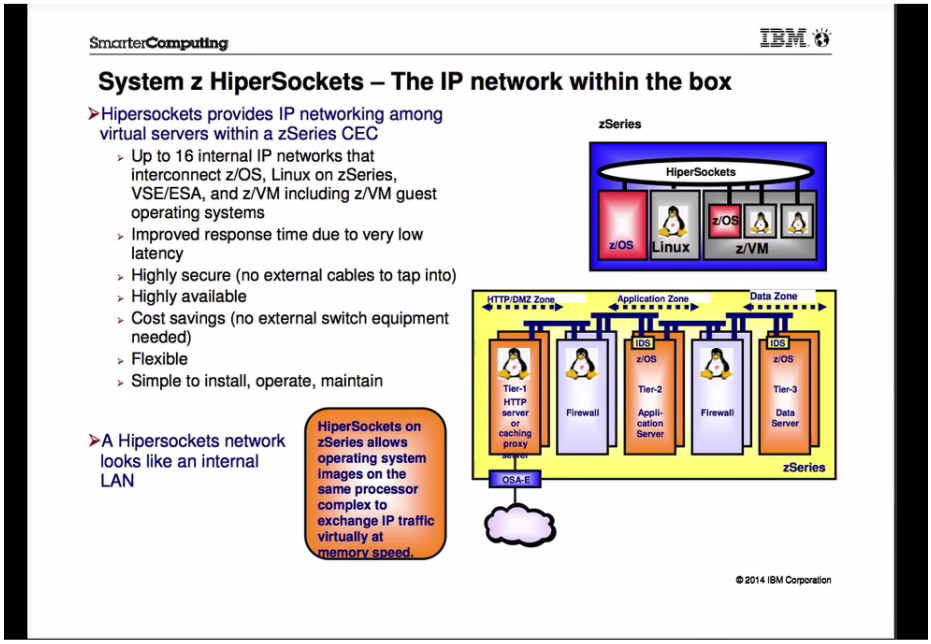

Figure 3: TCP/IP Connection / Hipersockets

—- Explain Hipersockets / TCP/IP Connection —-

A way we can use this connection to transfer jcl-jobs is:

Transmission Control Protocol

The Transmission Control Protocol (TCP) provides a reliable vehicle for delivering packets between hosts on an internet. TCP takes a stream of data, breaks it into datagrams, sends each one individually using IP, and reassembles the datagrams at

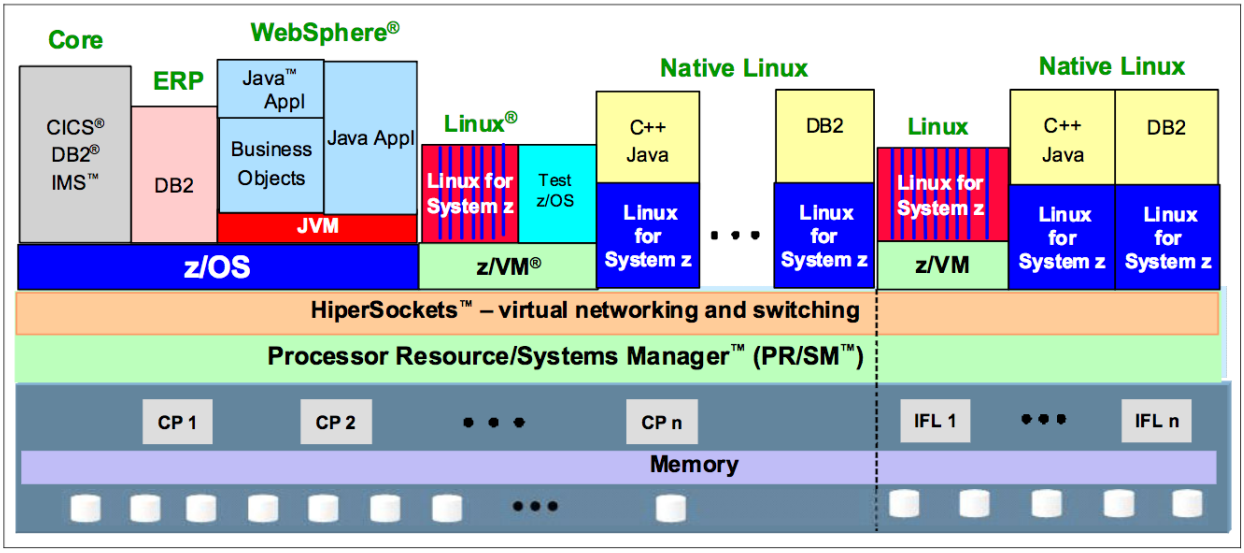

Figure 1-1 Multiple workloads on the mainframe

Figure 4: workloads

the destination node. If any datagrams are lost or damaged during transmission, TCP detects this and resends the missing datagrams. The received data stream is a reliable copy of the transmitted data stream.

—- Explain TCP/IP Connection in own words —-

8.1 Access through Java with FTP

What becomes possible through the TCP/IP in System Z is the File Transfer Protocol (FTP) Server, an IP-application used to transfer files between any kind of platform. The z/OS FTP-Server is a bit different from normal FTP-Servers as it provides not only the possibility to transfer files and get access to z/OS System Services, but it also provides access to the Job Entry Subsystem which is, as mentioned in chapter 7, needed to submit JCL-Jobs.

With the help of Java you can use the FTP server to get access to a number of JES functions, including the following:

- Submitting a job
- Displaying the status of jobs
- Receiving the spool output of a job (JCL messages and SYSOUT)

- Deleting a job
- Submitting a job and automatically receiving output

—- Importance of output for Learning environment —-

8.2 Access via SSH/ SFTP

While FTP gives us the opportunity to get access to the JES-Spool to run JCL-Jobs, this is not really a safe way to work this out.
To get a safe communication, it needs encryption what can be established through the SSH-protocol - making the FTP a Secure File Transfer Protocol.

The Secure Shell Protocol

The SSH protocol (also referred to as Secure Shell) is a method for secure remote login from one computer to another. It provides several alternative options for strong authentication, and it protects the communications security and integrity with strong encryption. It is a secure alternative to the non-protected login protocols (such as telnet, rlogin) and insecure file transfer methods (such as FTP).
—- Explain SSH in own words —-

9 Requirements

A tool named Co:Z SFTP, an open-source product developed by Dovetail Technologies, will help to establish a safe connection with before mentioned techniques. It works as a port of OpenSSH SFTP for z/OS and therefore enables the access to z/OS datasets and spool files.

`http://www.redbooks.ibm.com/redbooks/pdfs/sg247603.pdf` from p.137

10 Development Environment

In this chapter, a development environment will be created. An Oracle Virtual-Machine called VirtualBox will be installed on MacOSX. In this virtual machine the open and free Linux-System Ubuntu 16.04 LTS will be running, to make it possible to implement an Apache HTTP-Server. The Apache Server makes it possible to develop a simple dynamic website offline on a localhost that can be accessed through any web-browser as long as the virtual machine is running.

Schematische Darstellung!!

Electron Framework short explanation

11 Design

We use a one-page Framework (hyperspace ansprechen?!)
Co:Z Documentation can be visited on http://dovetail.com/docs/sftp/using.html
Implementation of 4 sections:

11.1 Welcome

Not much explanation needed - Get started leads to first section: Your Jobs. Perhaps gamification part with points/stars or something. For use of the next sections an SFTP-Client is implemented to get a connection to Co:Z-SFTP
ssh2.js library explanation?!

```
var Client = require('ssh2').Client;
var connection = new Client();

var connSettings = {
     host: '141.2.192.32',
     port: 22, // Guest-SFTP Port
     username: 'u0xxx',
     password: 'xxxxxxxx'
};
```

11.2 Your Jobs

In this section the user can see the jobs he or she submitted so far and in addition the job the user is currently working on.

```
      <!-- Here the JES-Spool is listed -->
      <div id="dirout"></div>
```

With the following JS-function, the jobs are passed to the application.

```
var remotePathToList = '//-JES';

var conn = new Client();
conn.on('ready', function() {
    conn.sftp(function(err, sftp) {
         if (err) throw err;

         sftp.readdir(remotePathToList, function(err, list) {
               if (err) throw err;
```

```
10
11                  var rows = list;
12
13                      var jobs = "<table border='1|1'>";
14                      for (var i = 0; i < rows.length; i++) {
15                          jobs+="<tr>";
16                          jobs+="<td>"+rows[i].filename+"</td>";
17                          jobs+="<td>"+rows[i].longname+"</td>";
18
19                          jobs+="</tr>";
20                      }
21                      jobs+="</table>";
22                  document.getElementById("dirout").innerHTML = jobs;
23
24                  // Do not forget to close the connection, otherwise you'll get troubles
25                  conn.end();
26          });
27      });
28  }).connect(connSettings);
```

- Line 1: Navigating to the //-JES directory, which is a directory, defined through Co:Z and contains the job-files.
- Lines 3-5: Here the connection is established with the above described parameters.
- Line 6: Throws an error if something went wrong (in console! to be changed?!).
- from Line 8: the directory is readed. Here two variables, err and list, are defined. The err variable is just to get output in case of errors. List defines the Objects located in the //-JES directory.
- from Line 11: The List content gets defined as rows, those rows are given to a table that is defined as a variable "jobs". The function in jobs, adds a row for each object.
- Line 22: Calls a div in index.html with the id="dirout" which is empty by default. Here it puts the table with the listed jobs.
- After this the connection is ended, to get always just a status of the directory in the moment it is charged.

11.3 Submit a new Job

First a file is created as a text-file. explain buttons.
Screenshots!?

```
        <button onclick="createFile()">Create Job</button>
```

```
  <div style="margin-top:30px;">
    <p id="filesuccess">Write your Job in the textarea above and
        click on Create Job.</p>
    <div class="field">
      <button id="subBut" style="display:none;" onclick="transFile()">
          Submit the Job now</button>
      <div id="messages"></div>
    </div>
  </div>
```

Use node filesystem. Change content - innerHTML and color. explain variables.

```
// Create Job file
function createFile() {
  var fs = require('fs');
  var filepath = "temp/JCL.txt";
  var fileContent = document.getElementById("JCLcontent").value;
  var filesuccess = document.getElementById("filesuccess");
  var subBut = document.getElementById("subBut");

  //Check if textarea contains content
  if (fileContent == '') {

    filesuccess.style.color = "#ff9800";
    filesuccess.innerHTML = "Please write a Job first.";

  } else if (fileContent != '') {

    // if it does, create textfile
    fs.writeFile(filepath, fileContent, (err) => {
                  if (err) throw err;

      //output as success or "create a job first"
      filesuccess.style.color = "#00e676";
      filesuccess.innerHTML = "Your Job has been created.
          Submit now to the JES-Spool.";
      subBut.style.display = "block";
```

```
26
27          });
28    }
29 };
```

Change the mode ls = readdir in node. Explain streams.

```
1  //Transfer Job File to Mainframe
2  function transFile() {
3  var changeMod = '/+mode=text'
4
5  var conn = new Client();
6    conn.on('ready', function() {
7      console.log('Client -> ready');
8        conn.sftp(function(err, sftp) {
9             if (err) throw err;
10
11              // change Transfer Mode with ls /+mode=text
12              sftp.readdir(changeMod, function(err, list) {
13                     if (err) throw err;
14                     // List the Changing in the console
15                     console.dir('mode -> text');
16                     // This time no connection closing, to get the output-action
17              });
18
19
20              var fs = require("fs"); // Use node filesystem
21
22              //Read the textfile and send it to subdirectory, called "myjob"
23              var readStream = fs.createReadStream( 'temp/JCL.txt' );
24              var writeStream = sftp.createWriteStream( '//-JES.INTRDR/MYJOB' );
25
26              writeStream.on('close',function () {
27                  console.log( "file transferred succesfully" );
28                  document.getElementById('three').style.display = 'block';
29              });
30
31              writeStream.on('end', function () {
32                  console.log( "sftp connection closed" );
33                  conn.close();
34              });
35
36              // initiate transfer of file
37              readStream.pipe( writeStream );
38          });
39
40        }).connect(connSettings);
```

```
41   };
```

"three" = Result area -> gets displayed after submitting

11.4 Result

Here describe Error handling and output divs.

```
<section id="three" class="wrapper style3 fade-up">

   <div class="inner">
      <h2>Your Result</h2>
      <p>Here you can see if your job was submitted to the mainframe.
      To get more details on the job status go to the overview.
      You can also check the return code there.</p>

  <div class="features">
  <section id="resultinfo">
      <span class="icon major fa-info"></span>
      <h3>No submit</h3>
      <p>You first have to submit something to get a result.</p>
  </section>

   <section id="goodresult" style="display:none;">
      <span class="icon major fa-check"></span>
      <h3>You succeeded</h3>
      <p>Your file was submitted.
      Go to the overview to check its status.</p>
   </section>

   <section id="badresult" style="display:none;">
     <span class="icon major fa-times"></span>
     <h3>Something went wrong</h3>
     <p>Bad Connection. Check your VPN.</p>
   </section>
   </div>

   // The page has to get reloaded to view the new jobs in the list
     <ul class="actions">
       <li><a href="#one" class="button"
        onClick="window.location.reload()">Go to overview</a></li>
     </ul>

   </div>
 </section>
```

Catch TimeOut Error in ssh2 library, Client function.

```
function Client() {

 [ ... ]

   function startTimeout() {
    if (self.config.readyTimeout > 0) {
      self._readyTimeout = setTimeout(function() {
        var err = new Error('Timed out while waiting for handshake');

        document.getElementById("badlist").style.display = "block";
        document.getElementById("resultinfo").style.display = "none";
        document.getElementById("badresult").style.display = "block";

        err.level = 'client-timeout';
        self.emit('error', err);
        sock.destroy();
      }, self.config.readyTimeout);
    }
  }
};
```

12 Proof of Concept

13 Evaluation

References

[1] Elias G Carayannis and Yiannis Nikolaidis. Enterprise networks and information and communication technologies (ict) standardization.

[2] Bill Cope and Mary Kalantzis. Ubiquitous learning: An agenda for educational transformation. *Ubiquitous learning*, pages 3–14, 2009.

[3] A Corridori. Ways to include enterprise computing content in current curriculum paper presented at the enterprise computing conference. 2009.

[4] David Douglas and Christine Davis. Enterprise computing: Bridging the gap to generation-y with rdz and zlinux web. Enterprise Computing Conference, 2009.

[5] Mike Ebbers, Wolfgang Bosch, Hans Joachim Ebert, Helmut Hellner, Jerry Johnston, Marco Kroll, Wilhelm Mild, Wayne O'Brien, Bill Ogden, Ingolf Salm, et al. *Introduction to the New Mainframe: IBM Z/VSE Basics*. IBM Redbooks, 2016.

[6] Gwo-Jen Hwang, Tsai Chin-Chung, and Stephen JH Yang. Criteria, strategies and research issues of context-aware ubiquitous learning. *Journal of Educational Technology & Society*, 11(2), 2008.

[7] Gwo-Jen Hwang, Tzu-Chi Yang, Chin-Chung Tsai, and Stephen JH Yang. A context-aware ubiquitous learning environment for conducting complex science experiments. *Computers & Education*, 53(2):402–413, 2009.

[8] Vicki Jones and Jun H Jo. Ubiquitous learning environment: An adaptive teaching system using ubiquitous technology. In *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, volume 468, page 474. Perth, Western Australia, 2004.

[9] Ravi Khadka, Belfrit V Batlajery, Amir M Saeidi, Slinger Jansen, and Jurriaan Hage. How do professionals perceive legacy systems and software modernization? In *Proceedings of the 36th International Conference on Software Engineering*, pages 36–47. ACM, 2014.

[10] Charles Kiefer. Cobol as a modern language. 2017.

[11] S Lohr. Ibm seeks to make the mainframe modern technology. *New York Times*, 3, 2006.

[12] Steve Lohr. Why old technologies are still kicking. *New York Times*, 4, 2008.

[13] Carol Midgley. *Goals, goal structures, and patterns of adaptive learning*. Routledge, 2014.

[14] Ken Sakamura and Noboru Koshizuka. Ubiquitous computing technologies for ubiquitous learning. In *Wireless and Mobile Technologies in Education, 2005. WMTE 2005. IEEE International Workshop on*, pages 11–20. IEEE, 2005.

[15] Aditya Sharma and Marianne C Murphy. Teach or no teach: Is large system education resurging? *Information Systems Education Journal*, 9(4):11, 2011.

[16] Aditya Sharma, Cameron Seay, and Marilyn K McClelland. Alive and kicking: Making the case for mainframe education marianne c. murphy mmurphy@ nccu. edu.

[17] Robert Vinaja. 50 th aniversary of the mainframe computer: a reflective analysis. *Journal of Computing Sciences in Colleges*, 30(2):116–124, 2014.

[18] I Wallis and B Rashed. Who's watching the mainframe. *IBM Systems Journal*, 2007.

[19] W Wong. Old tech skills again in demand: Mainframe computing jobs vacant as the baby boomers who set up systems begin retiring with few educated to fill the spots. *Chicago Tribune*, 2009.

14 Appendices