National College of Ireland
MSc in Web Technologies
2013/2014

Hugh Kelly
13117386
hugh.kelly@student.ncirl.ie

Deployment Project Report

I hereby certify that this material, which I now submit for assessment of the programme of study leading to the award of Master of Science in Web Technologies is entirely my own work and has not been taken from the work of others save and to the extent that such work has been citied and acknowledged within the text of my work.

Signed: ……

Date:

Student Number: 13117386

Table of Contents

1. Introduction....4
2. Delivery Process Specification (Deliverable 1)....5
2.1. User requirements....5
2.2. Function requirements....6
Use case for Process Trigger....6
Use case for Monitor....6
Use case for Process debug....6
Use case of Report request....6
2.3. Risks and Controls....6
3. Delivery Process Design (Deliverable 2)....7
3.1. Detailed design....7
Process Flow....7
Ideal Architecture....7
Actual Architecture....7
3.2. Build process design....8
Build Sub Functions....8
3.3. Integration process....8
Integration Sub Functions....8
3.4. Test process....9
Test Sub functions....9
3.5. Deployment process....9
Deployment sub functions....9
3.6. Monitoring process....10
Monitoring sub functions....10
4. Code Base (Deliverable 3)....11
5. Test Plan (Deliverable 4)....26
5.1. Unit tests....26
5.2. Integration testing....26
5.3. System Testing....26
5.4. End user testing....26
6. Execution Plan (Deliverable 5)....27
7. Execution Record (Deliverable 6)....28

1. Introduction

The purpose of this document is to describe the implementation of a delivery process to manage the build, integration, test, deployment and monitoring of a web site. The content of the website will be a mix of static content, and content dynamically generated from a database.

The delivery process will be developed using the V-model SDLC and best practices as identified by ITIL. The project requirements as given will provide the 'User Requirements' and the requested Execution Plan will be equivalent to the User Acceptance Test. The requirements will be categorised according to the MoSoCoWo (Moscow) mnemonic i.e. Must have, Should have, Could have, Won't have. Given the relatively small size of this project the “Small V” model has been adopted. Test plans and a functional specification / design plan will be derived from the given requirements.

[Please note that the deployment as described in this document does not take place to a remote server or AWS]

2. Delivery Process Specification (Deliverable 1)

2.1. User requirements

No.	MoSoCoW	Statement
1	Mo	It must be possible to deploy content and components with a single command.
2	So	Error logging should be in one place and should make sure that 6 key parameters (e.g. memory, I/O etc) are within thresholds.
3	Mo	There must be four distinct phases to the implementation: Build, Integration, Test, Deploy.
4	Mo	Build process must download content from a repository such as Github.
5	So	Build process should check that all components and resources are in place for testing.
6	So	Build process should make sure that the environment is clean and revisions of necessary components are at the right level.
7	Co	Build process will integrate the static HTML content from 2 or more files, into one.
8	Mo	Integration process will integrate the static content with the components that provide the dynamic content, so as to create the overall content
9	Mo	Test process will make sure that the static content is properly constructed (HTML tags etc.).
10	Mo	Test process must make sure that the dynamic content functions as required.
11	Co	Deployment process ensures all components (content, packages etc) are in place for production.
12	Co	Deployment process ensures all resources (memory, disk, I/O etc) are in place for production.
13	Mo	Deployment process will unpack the content and move it to its proper location on the production server.
14	So	Deployment process should backup the content prior to deployment of new content. If the deployment fails, the old site should be kept in place.

2.2. Function requirements

Use case for Process Trigger

A user implements a website from github repository to the production server by entering a command on the deployment server console command line.

Use case for Monitor

A user can monitor the state of the production server by observing the console log.

Use case for Process debug

A user (i.e. developer) can debug the delivery process using meaningful feedback from the delivery process itself.

Use case of Report request

A user can request a report after the delivery process has completed.

2.3. Risks and Controls

No.	Impact	Risk Statement	Control
1	High	Corrupted content could replace valid working content on the live platform	Content is backed up prior to deployment of new content. If the deployment fails, the old site is kept in place.
2	High	Network unavailable	Check network availability, if connectivity issue abort deployment.
3	High	Disk Space low	Check disk space, if too low abort deployment.
4	High	Memory low	Check available memory, if too low abort deployment.

3. Delivery Process Design (Deliverable 2)

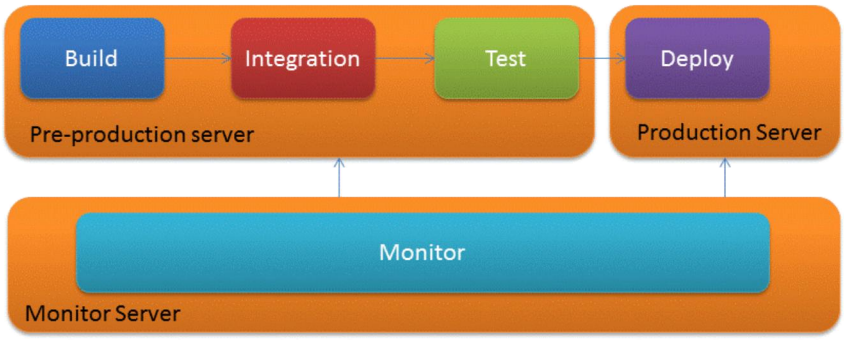
3.1. Detailed design

The deployment path will follow the process flow as shown below, i.e. build the application, integrate it, test it and then deploy to the server. These four phases will all be carried out in a 'sandbox', so that if there are issues the deployment code can easily be examined and dealt with.

Process Flow

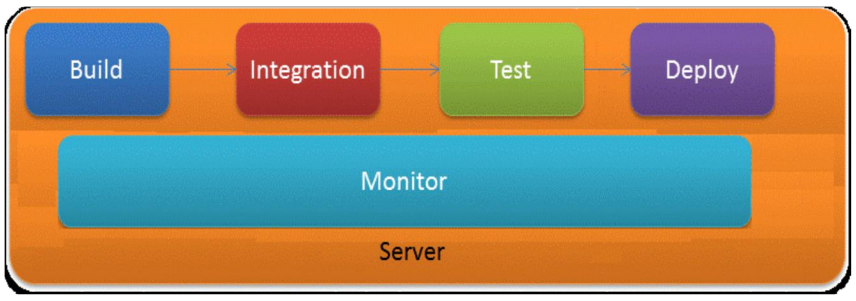
Ideally the architecture of the delivery process would be as shown below but due to time pressure it has not been possible to design for three servers as shown. Rather the entire Build, Integration, Test, Deploy and Monitor process will take place on one server although the overall sequence of the processes will be followed as if three servers were being used.

Ideal Architecture



The actual design is as shown below with all activity taking place on one server.

Actual Architecture



3.2. Build process design

The build process will download content from a repository (e.g. github). It checks that all components and resources are in place for testing. It integrates the static HTML content and makes sure that the environment is clean and revisions of necessary components are at the right level. This corresponds to ITIL's Service Transition 'Release and Deployment Management' as it concerns obtaining the components from the code repository that make up the deployment process.

Build Sub Functions

Create sandbox

Download from Github

Create stage directories in sandbox (build, integration, test, deploy)

Make webpackage and move webpackage to 'build'

Get md5sum to check if webpackage has changed since last deploy

Move webpackage to 'build' directory

Check build

If errors abort deployment and send email to admin else ...

Clean-up

Report/Log

3.3. Integration process

Integration integrates the static content with the components that provide the dynamic content, so as to create the overall content. Corresponds to ITIL's Service Transition 'Configuration Management' practice as it I concerns bringing together the various components that make up the deployment process.

Integration Sub Functions

Put webpackage in integrate area

Extract webpackage

Modify static html

Package up for Test phase

If errors abort and send email to admin

Report/Log

3.4. Test process

The test process makes sure that the static content is properly constructed (HTML tags etc.), and that the dynamic content functions as required. It corresponds to ITIL's Service Transition 'Service Validation' practice.

Test Sub functions

Move webpackage to test area

Extract webpackage to test area

Html checks

Tar webpackage for deploy

If errors abort deploy and send email to admin

Report/Log

3.5. Deployment process

Deployment ensures that all components (content, packages etc) and resources (memory, disk, I/O etc) are in place for production. It unpacks the content and move it to its proper location on the production server. It backs up the content prior to deployment of new content. If the deployment fails, the old site is kept in place. Ideally deployment should take place to a remote production server but difficulties were encountered in implementing this, namely getting ssh and scp to work consistently when invoked from scripts. Ssh public and private keys were created successfully using ssh-keygen and it was possible to issue the public keys to other servers (using ssh-agent, ssh-add and ssh-copy-id) but, although it was possible from the command line to issue ssh commands to a remote system and to copy files using scp, both ssh and scp did not work consistently when invoked from bash script requiring user intervention to enter a password. Therefore the capability of deploying to a remote server has been omitted from this implementation.

This process corresponds to both ITILS's 'Release and Deployment' practice and to initiating Service Operation e.g. 'Event Management' / 'Incident Management' are initiated during this process (i.e. logging and monitoring of system events and system status).

Deployment sub functions

If no errors move webpackage to deploy area and extract

Stop services

Remove and re-install apache

Remove and re-install mysql

Backup up existing website

Copy over new website

Start db service

Backup Existing database

Check mysql database

Restore original dbtest database

Start web service

Check deployed web page

If errors rollback and send email

Clean Up/Report/Log

3.6. Monitoring process

Monitoring ensures that the site is functioning from a HTML, HTTP and other socket layer perspective. It should make sure that 6 key parameters (e.g. memory, I/O etc) are within thresholds. It should report errors. This concern's ITIL's Service Operation 'Event Management'.

Monitoring sub functions

Check mysql port

Check http port

Check disk space

Check memory

Check Network availability

Check Network utilisation

4. Code Base (Deliverable 3)

The code has been implementated on a modular basis with bash and ruby scripts:

clean_env.sh – bash script to clean environment prior to B/I/T/D execution.

```
#!/bin/bash
#
# # NCI MscWebTech 2013-14
# # Deployment Project
# # Hugh Kelly 13117386
#
# Script that cleans and prepares env
#

# ---------------------------------------------------------------
#
# This clean and preparation process will clean up the B/I/T/D server
environment.
#

# - Enable sources
apt-get update

# - Stop web and db services
service apache2 stop
service mysql stop

# - Remove apache and install latest version
apt-get -q -y remove apache2
apt-get -q -y install apache2
/usr/sbin/apache2 -v

# - Remove mysql and install latest version
apt-get -q -y remove mysql-server mysql-client
echo mysql-server mysql-server/root_password password password | debconf-set-
selections
echo mysql-server mysql-server/root_password_again password password | debconf-
set-selections
apt-get -q -y install mysql-server mysql-client
mysql --version

# - Start web and db services
service apache2 start
service mysql start

# - install ruby
sudo apt-get install ruby1.9.1 ruby1.9.1-dev \
  rubygems1.9.1 irb1.9.1 ri1.9.1 rdoc1.9.1 \
  build-essential libopenssl-ruby1.9.1 libssl-dev zlib1g-dev
sudo update-alternatives --install /usr/bin/ruby ruby /usr/bin/ruby1.9.1 400 \
         --slave   /usr/share/man/man1/ruby.1.gz ruby.1.gz \
                        /usr/share/man/man1/ruby1.9.1.1.gz \
        --slave   /usr/bin/ri ri /usr/bin/ri1.9.1 \
        --slave   /usr/bin/irb irb /usr/bin/irb1.9.1 \
        --slave   /usr/bin/rdoc rdoc /usr/bin/rdoc1.9.1
ruby --version

# - remove any previous stuff
cd
rm /tmp/logfile.txt
rm deploy.rep
#
echo ENVIRONMENT CLEANED AND PREPARED
```

deploy.sh – bash script to set up crontab for monitoring and to kick off the B/I/T/D execution.

```
#!/bin/bash
#
# # NCI MscWebTech 2013-14
# # Deployment Project
# # Hugh Kelly 13117386
#
# Script that sets up cron and kicks off deployment
#
# Set up logging and monitoring cron job
crontab -r
(crontab -l 2>/dev/null; echo "* * * * * /home/testuser/deploy/logmon.sh >>
/tmp/logfile.txt") | crontab -
#
# call deployment script
sudo /home/testuser/deploy/deployment.sh
```

deployment.sh – B/I/T/D execution bash script

```
#!/bin/bash
#
# # NCI MscWebTech 2013-14
# # Deployment Project
# # Hugh Kelly 13117386
#
# Script that deploys a website from github
#
#    1  Build Process
#    2  Integration Process
#    3  Test Process
#    4  Deployment Process
#

ADMINISTRATOR="mscwebtech.kelly@gmail.com"
PASSWORD="Y0urPetsName"
DEBUG=false

# FUNCTION LIBRARY
source /home/testuser/deploy/func.lib


# ---------------------------------------------------------------------
# ---------------------------------------------------------------------
# 1 BUILD process  <--------------------------------------------------
# ---------------------------------------------------------------------
#
# The build process will download content from a repository (e.g. github). It
# checks that all components and resources are in place for testing. The build
# process makes sure that the environment is clean and revisions of necessary
# components are at the right level.
#

# - Set up sandbox
cd /tmp
SANDBOX=sandbox_$RANDOM
mkdir $SANDBOX
cd    $SANDBOX/
ERRORCHECK=0

# - Download from Github
if [[ "DEBUG" == "true" ]]
then
  mkdir webpackage               # make simple static web site
  touch webpackage/index.htm
```

```
  touch webpackage/form.htm
  touch webpackage/script1.plx
  touch webpackage/script2.plx
else
  git clone https://github.com/jhugh/NCIRL.git
  mv NCIRL webpackage
fi

# - Create Stage Directories in sandbox
mkdir build
mkdir integrate
mkdir test
mkdir deploy

# - Make webpackage and move webpackage to 'build'
tar -zcvf webpackage_preBuild.tgz webpackage
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=1
fi

# - Get md5sum to check if webpackage has changed since last deploy
MD5SUM=$(md5sum webpackage_preBuild.tgz | cut -f 1 -d' ')
PREVMD5SUM=$(cat /tmp/md5sum)
FILECHANGE=0
if [[ "$MD5SUM" != "$PREVMD5SUM" ]]
then
        FILECHANGE=1
        echo $MD5SUM not equal to $PREVMD5SUM
else
        FILECHANGE=0
        echo $MD5SUM equal to $PREVMD5SUM
fi
echo $MD5SUM > /tmp/md5sum
if [ $FILECHANGE -eq 0 ]
then
        echo no change in files, doing nothing and exiting
        cd
        exit
fi

# - Move webpackage to 'build'
mv webpackage_preBuild.tgz build
rm -rf webpackage
cd build
tar -zxf webpackage_preBuild.tgz
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=2
fi

# - Check build
INDEX_EXIST=$(ls webpackage/Apache/www | grep 'index.html')
if [[ "$INDEX_EXIST" != "index.html" ]]
then
  ERRORCHECK+=4
fi
tar -zcf webpackage_preIntegrate.tgz webpackage
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=8
fi

# - If errors send email to admin
if [ $ERRORCHECK -ne 0 ]
then
        echo $ERRORCHECK build errors, exiting
           /home/testuser/deploy/sendmail.rb $ADMINISTRATOR $PASSWORD "CRITICAL
Deploy Error: $ERRORCHECK" "There is a problem with Build."
        cd
```

```
        exit
else
        echo $ERRORCHECK build errors
fi
B_ERRORS=$ERRORCHECK


# ------------------------------------------------------------------
# ------------------------------------------------------------------
# 2 INTEGRATION process <-------------------------------------------
# ------------------------------------------------------------------
#
# Integration integrates the static content with the components that provide the
# dynamic content, so as to create the overall content.
#
ERRORCHECK=0

# - Put webpackage in integrate area
mv webpackage_preIntegrate.tgz ../integrate
rm -rf webpackage
cd ../integrate

# - Extract webpackage
tar -zxf webpackage_preIntegrate.tgz
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=1
fi

# - Modify static html
#DATESTAMP=$(date +"%D")
TIMESTAMP=$(date +"%m-%d-%Y %T")
#TIMEDATE="$DATESTAMP $TIMESTAMP"
echo $TIMESTAMP
sed    -i    s/"It    works"/"MScWebTech    Deployment    project    $TIMESTAMP"/
webpackage/Apache/www/index.html
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=2
fi

# - Tar up for Test phase
tar -zcf webpackage_preTest.tgz webpackage
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=4
fi

# - If errors send email to admin
if [ $ERRORCHECK -ne 0 ]
then
        echo $ERRORCHECK integration errors, exiting
           /home/testuser/deploy/sendmail.rb $ADMINISTRATOR $PASSWORD "CRITICAL
Deploy Error: $ERRORCHECK" "There is a problem with Integration."
        cd
        exit
else
        echo $ERRORCHECK integration errors
fi
I_ERRORS=$ERRORCHECK



# ------------------------------------------------------------------
# ------------------------------------------------------------------
# 3 TEST process  <-------------------------------------------------
# ------------------------------------------------------------------
#
# The test process makes sure that the static content is properly constructed
# (HTML tags etc.), and that the dynamic content functions as required.
```

```
#
ERRORCHECK=0

# - Move webpackage to test area
mv webpackage_preTest.tgz ../test
rm -rf webpackage
cd ../test

# - Extract webpackage to test are
tar -zxf webpackage_preTest.tgz
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=1
fi

# - Html checks
HTMLTAG=$(cat webpackage/Apache/www/index.html | grep '<html>' | cut -d '>' -f
1)
if [[ "$HTMLTAG" != "<html" ]]
then
  ERRORCHECK+=2
fi
HTMLTAG=$(cat webpackage/Apache/www/index.html | grep 'MScWebTech' | cut -d '>'
-f 1)
if [[ "$HTMLTAG" != "<html" ]]
then
  ERRORCHECK+=4
fi

# - Tar webpackage for deploy
tar -zcf webpackage_preDeploy.tgz webpackage
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=8
fi

# - If errors send email to admin
if [ $ERRORCHECK -ne 0 ]
then
        echo $ERRORCHECK test errors, exiting
           /home/testuser/deploy/sendmail.rb $ADMINISTRATOR $PASSWORD "CRITICAL
Deploy Error: $ERRORCHECK" "There is a problem with Test."
        cd
        exit
else
        echo $ERRORCHECK test errors
fi
T_ERRORS=$ERRORCHECK



# ------------------------------------------------------------------
# ------------------------------------------------------------------
# 4 DEPLOYMENT process  <-------------------------------------------
# ------------------------------------------------------------------
#
# Deployment ensures that all components (content, packages etc) and resources
# (memory, disk, I/O etc) are in place for production. It unpacks the content
# and moves it to its proper location on the production server. It backs up the
# content prior to deployment of new content. If the deployment fails, the old
# site is kept in place.
#

# - If no errors move webpackage to deploy area and extract
if [ $ERRORCHECK -eq 0 ]
then
        mv webpackage_preDeploy.tgz ../deploy
        rm -rf webpackage
        cd ../deploy
```

```
        tar -zxf webpackage_preDeploy.tgz
fi

# - Stop services
service apache2 stop
service mysql stop

# - Remove and re-install apache
apt-get -q -y remove apache2
apt-get -q -y install apache2

# - Remove and re-install mysql
apt-get -q -y remove mysql-server mysql-client
echo mysql-server mysql-server/root_password password password | debconf-set-
selections
echo mysql-server mysql-server/root_password_again password password | debconf-
set-selections
apt-get -q -y install mysql-server mysql-client

# - Backup up existing website
mkdir backup
mkdir backup/www
cp /var/www/* backup/www
mkdir backup/cgi-bin
cp /usr/lib/cgi-bin/* backup/cgi-bin

# - Copy website
cp webpackage/Apache/www/* /var/www/
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=1
fi
cp webpackage/Apache/cgi-bin/* /usr/lib/cgi-bin/
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=2
fi
chmod a+x /usr/lib/cgi-bin/*
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=4
fi

# - Start db service
service mysql start

# - Backup Existing database
mysqldump -uroot -ppassword --databases dbtest > backup.sql
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=8
fi

# - Check mysql database
cat <<FINISH | mysql -uroot -ppassword
drop database if exists dbtest;
CREATE DATABASE dbtest;
GRANT ALL PRIVILEGES ON dbtest.* TO dbtestuser@localhost IDENTIFIED BY
'dbpassword';
use dbtest;
drop table if exists custdetails;
create table if not exists custdetails (
name         VARCHAR(30)   NOT NULL DEFAULT '',
address         VARCHAR(30)   NOT NULL DEFAULT ''
);
insert into custdetails (name,address) values ('John Smith','Street Address');
select * from custdetails into outfile 'db_check.txt';
FINISH
DBCHECK=$(cat /var/lib/mysql/dbtest/db_check.txt | grep 'John Smith' | cut -d '
' -f 1)
if [[ "$DBCHECK" != "John" ]]
```

```
then
  ERRORCHECK+=16
fi
rm /var/lib/mysql/dbtest/db_check.txt

# - Restore original dbtest database
mysql -uroot -ppassword < backup.sql
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=32
fi

# - Start web service
service apache2 start

# - Check deployed web page
wget -O received_page.txt "http://localhost/index.html"
if [ "$?" -ne "0" ]; then
  ERRORCHECK+=64
fi
HTMLTAG=$(cat received_page.txt | grep 'MScWebTech' | cut -d '>' -f 1)
if [[ "$HTMLTAG" != "<html" ]]
then
  ERRORCHECK+=128
fi

# - Errorcheck, rollback and send email if issues
if [ $ERRORCHECK -ne 0 ]
then
        echo $ERRORCHECK deploy errors, rolling back and exiting
        echo Rolling Back Website
        service apache2 stop
        cp backup/www/* /var/www/
        cp backup/cgi-bin/* /usr/lib/cgi-bin/
        chmod a+x /usr/lib/cgi-bin/*
        service apache2 start
           /home/testuser/deploy/sendmail.rb $ADMINISTRATOR $PASSWORD "CRITICAL
Deploy Error: $ERRORCHECK" "There is a problem with Deploy."
        cd
        exit
else
        echo $ERRORCHECK deploy errors
fi
D_ERRORS=$ERRORCHECK

# - Tidy up
cd /tmp
rm -rf $SANDBOX
cd

# - report all errors
echo
echo Error Summary
echo Build errors: $B_ERRORS
echo Intgration errors: $I_ERRORS
echo Test errors: $T_ERRORS
echo Deploy errors: $D_ERRORS
echo


# - check all is running ok
ERRORCOUNT=0
# - CHECK MYSQL
isMysqlRunning
if [ "$?" -eq 1 ]; then
        echo Mysql process is Running
else
        echo Mysql process is not Running
```

```
        ERRORCOUNT=$((ERRORCOUNT+1))
fi

isMysqlListening
if [ "$?" -eq 1 ]; then
        echo Mysql is Listening
else
        echo Mysql is not Listening
        ERRORCOUNT=$((ERRORCOUNT+1))
fi

isMysqlRemoteUp
if [ "$?" -eq 1 ]; then
        echo Remote Mysql TCP port is up
else
        echo Remote Mysql TCP port is down
        ERRORCOUNT=$((ERRORCOUNT+1))
fi

# - CHECK HTTP
isApacheRunning
if [ "$?" -eq 1 ]; then
        echo Apache process is Running
else
        echo Apache process is not Running
        ERRORCOUNT=$((ERRORCOUNT+1))
fi

isApacheListening
if [ "$?" -eq 1 ]; then
        echo Apache is Listening
else
        echo Apache is not Listening
        ERRORCOUNT=$((ERRORCOUNT+1))
fi

isApacheRemoteUp
if [ "$?" -eq 1 ]; then
        echo Remote Apache TCP port is up
else
        echo Remote Apache TCP port is down
        ERRORCOUNT=$((ERRORCOUNT+1))
fi


# - SEND EMAIL WHEN DEPLOYMENT IS COMPLETE
if  [ $ERRORCOUNT -gt 0 ]
then
          ./deploy/sendmail.rb $ADMINISTRATOR $PASSWORD "CRITICAL Deploy Issue"
"There are $ERRORCOUNT errors with Apache or Mysql"
else
              ./deploy/sendmail.rb  $ADMINISTRATOR  $PASSWORD  "No  Deploy  Issue"
"Deployment completed successfully"
fi
```

func.lib – bash script library of system functions used by other scripts

```
#!/bin/bash
#
# Function Library
#
# # NCI MscWebTech 2013-14
# # Deployment Project
# # Hugh Kelly 13117386
#
```

```
#
# 1. Functions
#    1.1 Level 0 functions
#    1.2 Level 1 functions
#    1.3 Level 2 functions
#
#
# ********************************************************************
# 1. FUNCTIONS
# ********************************************************************
#
#
# 1.1 Level 0 functions <-----------------------------------------------
#
#

function isRunning {
PROCESS_NUM=$(ps -ef | grep "$1" | grep -v "grep" | wc -l)
if [ $PROCESS_NUM -gt 0 ] ; then
        #echo $PROCESS_NUM
        return 1
else
        return 0
fi
}


function isTCPlisten {
TCPCOUNT=$(netstat -tupln | grep tcp | grep "$1" | wc -l)
if [ $TCPCOUNT -gt 0 ] ; then
        return 1
else
        return 0
fi
}


function isUDPlisten {
UDPCOUNT=$(netstat -tupln | grep udp | grep "$1" | wc -l)
if [ $UDPCOUNT -gt 0 ] ; then
        return 1
else
        return 0
fi
}


function isTCPremoteOpen {
timeout 1 bash -c "echo >/dev/tcp/$1/$2" && return 1 ||  return 0
}


function isIPalive {
PINGCOUNT=$(ping -c 1 "$1" | grep "1 received" | wc -l)
if [ $PINGCOUNT -gt 0 ] ; then
        return 1
else
        return 0
fi
}


function getCPU {
app_name=$1
cpu_limit="5000"
app_pid=`ps aux | grep $app_name | grep -v grep | awk {'print $2'}`
app_cpu=`ps aux | grep $app_name | grep -v grep | awk {'print $3*100'}`
```

```
if [[ $app_cpu -gt $cpu_limit ]]; then
     return 0
else
     return 1
fi
}


function freeMEM {
MFREE=`cat /proc/meminfo | grep MemFree: | awk '{print $2}'`
if [ "$MFREE" -lt 160000 ]; then
        return 1
else
        return 0

fi
}

function totalMEM {
TOTAL=`cat /proc/meminfo | grep MemTotal: | awk '{print $2}'`
echo "Total Memory here $TOTAL"
if [ "$TOTAL" -lt 1000000 ]; then
        return 1
else
        return 0

fi
}

function cpuINFO {
echo "Getting cpu Details"
CPUTYPE=`cat /proc/cpuinfo | grep "vendor_id"  | awk '{print $3}'`
#CPUTYPE=$(grep "model name" /proc/cpuinfo)
echo $CPUTYPE
c=`iostat | awk '{if(NR==4) {print $1}}'`
echo $c
CPU=`ps aux | grep "mySql" | grep -v grep | awk '{print $3}'`
        echo "CPU status is:" $CPU
}


#
# 1.2 Level 1 functions <-----------------------------------------------
#
function isApacheRunning {
        isRunning apache2
        return $?
}

function isApacheListening {
        isTCPlisten 80
        return $?
}

function isApacheRemoteUp {
        isTCPremoteOpen 127.0.0.1 80
        return $?
}

function isMysqlRunning {
        isRunning mysqld
        return $?
}

function isMysqlListening {
        isTCPlisten 3306
        return $?
```

```
}

function isMysqlRemoteUp {
        isTCPremoteOpen 127.0.0.1 3306
        return $?
}


#
# 1.3 Level 2 functions <---------------------------------------------
#
function isDepMysqlRunning {
       ssh testuser@dep-server . func.lib isMysqlRunning
       return $?
}

function isDepMysqlListening {
       ssh testuser@dep-server isMysqlListening
       return $?
}

function isDepMysqlRemoteUp {
       ssh testuser@dep-server isMysqlRemoteUp
       return $?
}

APACHE_ERROR=1
function unitTestApache {
    APACHE_ERROR=1
    isApacheRunning
    if [ "$?" -eq 1 ]; then
        isApacheListening
        if [ "$?" -eq 1 ]; then
            isApacheRemoteUp
            if [ "$?" -eq 1 ]; then
                echo "Apache is Ok"
            else
                APACHE_ERROR=0
            fi
        else
            APACHE_ERROR=0
        fi
    else
        APACHE_ERROR=0
    fi
    if [ $APACHE_ERROR -eq 0 ]; then
        echo "Apache is NOT Ok"
    fi
}

MYSQL_ERROR=1
function unitTestMysql {
    MYSQL_ERROR=1
    isMysqlRunning
    if [ "$?" -eq 1 ]; then
        isMysqlListening
        if [ "$?" -eq 1 ]; then
            isMysqlRemoteUp
            if [ "$?" -eq 1 ]; then
                echo "Mysql is Ok"
            else
                MYSQL_ERROR=0
            fi
        else
            MYSQL_ERROR=0
        fi
    else
```

```
        MYSQL_ERROR=0
    fi
    if [ $MYSQL_ERROR -eq 0 ]; then
        echo "Mysql is NOT Ok"
    fi
}

function unitTestMem {
   freeMEM
   if [ "$?" -eq 1 ]
   then
      totalMEM
      if [ "$?" -eq 1 ]
      then
          echo "Memory OK"
      else
          echo "Memory NOT ok"
      fi
   else
      echo "Memory NOT ok"
   fi
}
```

inttest.sh – bash script to carry out integration test

```
#!/bin/bash
#
# Integration Test
#
# # NCI MscWebTech 2013-14
# # Deployment Project
# # Hugh Kelly 13117386
#
#
#

source /home/testuser/deploy/func.lib

unitTestApache
unitTestMysql
unitTestMem
```

logmon.sh – bash script called by cron to provide monitoring log

```
#!/bin/bash
#
# # NCI MscWebTech 2013-14
# # Deployment Project
# # Hugh Kelly 13117386
#
# Script that monitors apache and mysql
#

source /home/testuser/deploy/func.lib

# ------------------------------------------------------------------
# ------------------------------------------------------------------
# Monitoring Process  <--------------------------------------------
# ------------------------------------------------------------------
# Monitoring ensures that the site is functioning from a HTML, HTTP and other
socket
# layer perspective. It should make sure that 6 key parameters (e.g. memory, I/O
etc)
# are within thresholds. It should report errors. This is kicked off by cron job
# set up by deploy.sh which calls this script.
#
# Monitoring sub functions:
# - Check mysql port
```

```
# - Check http port
# - Check disk space
# - Check memory
# - Check Network availability
# - Check Network utilisation


isApacheRunning
if [ "$?" -eq 1 ]; then
        echo $(date +"%F %T") Apache process is Running
else
        echo $(date +"%F %T") CRITICAL Apache process is not Running
fi

isApacheListening
if [ "$?" -eq 1 ]; then
        echo $(date +"%F %T") Apache is Listening
else
        echo $(date +"%F %T") CRITICAL Apache is not Listening
fi

isApacheRemoteUp
if [ "$?" -eq 1 ]; then
        echo $(date +"%F %T") Remote Apache TCP port is up
else
        echo $(date +"%F %T") CRITICAL Remote Apache TCP port is down
fi

isMysqlRunning
if [ "$?" -eq 1 ]; then
        echo $(date +"%F %T") Mysql process is Running
else
        echo $(date +"%F %T") CRITICAL Mysql process is not Running
fi

isMysqlListening
if [ "$?" -eq 1 ]; then
        echo $(date +"%F %T") Mysql is Listening
else
        echo $(date +"%F %T") CRITICAL Mysql is not Listening
fi

isMysqlRemoteUp
if [ "$?" -eq 1 ]; then
        echo $(date +"%F %T") Remote Mysql TCP port is up
else
        echo $(date +"%F %T") CRITICAL Remote Mysql TCP port is down
fi


getCPU
if [ "$?" -eq 0 ]; then
        echo $(date +"%F %T") CRITICAL Processor usage is above Limits
else
        echo $(date +"%F %T") Processor usage is within normal range
fi

freeMEM
if [ "$?" -eq 0 ]; then
        echo $(date +"%F %T") CRITICAL Memory usage is above Limits
else
        echo $(date +"%F %T") Memory usage is within normal range
fi

# - total disk space
TOTALDISKSPACE=`df  /dev/sda1 | sed '1d' | awk '{print $2}' | cut -d'%' -f1`
echo $(date +"%F %T") "Total Disk Space          :" $TOTALDISKSPACE
```

```
# - available disk space
AVAILABLEDISKSPACE=`df  /dev/sda1 | sed '1d' | awk '{print $4}' | cut -d'%' -f1`
echo $(date +"%F %T") "Available Disk Space      :" $AVAILABLEDISKSPACE

# - percentage used disk space
PERCENTUSEDDISKSPACE=`df -H /dev/sda1 | sed '1d' | awk '{print $5}' | cut -d'%'
-f1`
echo $(date +"%F %T") "Used Disk Space           : ${PERCENTUSEDDISKSPACE}%"

# - disk capacity threshold
DISKSPACE=`df  /dev/sda1 | sed '1d' | awk '{print $5}' | cut -d'%' -f1`
ALERT=30
if [ ${DISKSPACE} -ge ${ALERT} ]; then
    echo $(date +"%F %T") "WARNING disk ${DISKSPACE}% full"
fi

# - check network
ping_return=`ping -c1 google.com 2>&1 | grep unknown`
if [ ! "$ping_return" = "" ]; then
        echo $(date +"%F %T") "CRITICAL - Network status : DOWN - attempting to
restart !!!"
       service network restart
else
       echo $(date +"%F %T") "Network status            : up"
fi

# - display system status
rload="$($_CMD uptime |awk -F'average:' '{ print $2}')"
rfreeram="$($_CMD free -mto | grep Mem: | awk '{ print $4 " MB" }')"
rtotalram="$($_CMD free -mto | grep Mem: | awk '{ print $2 " MB" }')"
rusedram="$($_CMD free -mto | grep Mem: | awk '{ print $3 " MB" }')"

rhostname="$($_CMD hostname)"
echo $(date +"%F %T") "Current System info:"
echo $(date +"%F %T") "    host       " $rhostname
echo $(date +"%F %T") "    load       " $rload
echo $(date +"%F %T") "    total ram  " $rtotalram
echo $(date +"%F %T") "    free ram   " $rfreeram
echo $(date +"%F %T") "    ram used   " $rusedram
echo $(date +"%F %T")
```

sendmail.rb – ruby script to send email to administrator (mscwebtech.kelly@gmail.com).

```
#!/usr/bin/ruby

## SEND EMAIL VIA GMAIL

require 'net/smtp'
require 'date'
require 'time'

@time = Time.now.asctime

## SET UP CONNECTION PARAMS
username = "#{ARGV[0]}"
pass = "#{ARGV[1]}"
domain = "gmail.com"

## CREATE MESSAGE
from = username
to = from
message = <<END_OF_MESSAGE
From: #{from}
To: #{to}
```

```
Subject: #{ARGV[2]}
#{ARGV[3]}
[sent at #{@time} ]
END_OF_MESSAGE

## SEND MESSAGE
smtp = Net::SMTP.new('smtp.gmail.com', 587 )
smtp.enable_starttls
smtp.start(domain, username, pass, :login) do |smtp|
        smtp.send_message message, from, to
end
```

All code and this document can be obtained from Github:

git clone https://github.com/jhugh/deploy.git

5. Test Plan (Deliverable 4)

5.1. Unit tests

Before each function was developed a corresponding unit test was created first, with a unit test for each element of the design, coded in func.lib.

5.2. Integration testing

The integration test was carried out by running a script that called all the unit test functions, inttest.sh.

5.3. System Testing

The full deployment was run and the outcome was checked against the use cases. The four use cases were satisfied:

Use case for Process Trigger: enter `./deploy/deploy.sh | tee deploy.rep`

Use case for Monitor: `/tmp/logfile.txt` generated by cron job, monitored by '`tail -f`' command.

Use case for Process debug: errors can be generated by the script at various points, an error code is produced where the value of the error code indicates where the error occurred within a particular B/I/T/D phase.

Use case of Report request: the execution of the deployment script generates an email indicating success or failure of the deployment as well as a monitor log (/tmp/logfile.txt) and a console session log of the deployment (/home/testuser/deploy.rep).

5.4. End user testing

This was carried out by following the Execution plan and checking the outcome against the user requirements. All were satisified except 5, 7, 11 and 12.

6. Execution Plan (Deliverable 5)

The following is an Execution plan, to be executed by a third party.

Step	Topic	Verify Y/N
1	Set Up And Clear Environment check home directory is /home/testuser git clone https://github.com/jhugh/deploy.git chmod 775 deploy chmod 775 deploy/deploy.sh sudo ./deploy/clean_env.sh | tee clean_env.rep	
2	Run Deployment scripting ./deploy/deploy.sh | tee deploy.rep	
3	Demonstrate static content is delivered through browser <ubuntu vm ip>/index.html	
4	Demonstrate dynamic content is delivered through browser <ubuntu vm ip>/form.html	
5	Demonstrate Logging is functioning correctly from another window run “tail -f /tmp/logfile.txt”	
6	Run Deployment scripting again	
7	Demonstrate Content is delivered correctly and logging is functioning correctly: repeat steps 2, 3, 4 and 5 and check that dynamic content (db data) is preserved	
8	Stop web server, verify logging of event in log sudo service apache2 stop	
9	Stop Database, verify logging of event in log sudo service mysql stop	
10	Start web server, verify logging of event in log sudo service mysql start	
11	Start Database, verify logging of event in log sudo service apache2 start	
12	Change the content in code repository. git clone https://github.com/jhugh/NCIRL.git cd NCIRL vi Apache/www/index.html (make some change) git add Apache/www/index.html git commit -m “modified” git push origin master cd Run deployment scripting again (steps 2, 3, 4 and 5) and verify that content is delivered correctly and logging is functioning correctly.	

Executed by : _____ Witnessed by : _____ Date : _____

7. Execution Record (Deliverable 6)

The completed Execution Plan record follows this section. Also listed below are various parts of the logs, web page snap shots and reports generated while the Execution Plan was being run:

clean_env.rep . . . (log of clean environment script)

```
Ign http://security.ubuntu.com precise-security InRelease
Ign http://ie.archive.ubuntu.com precise InRelease
...
```

... (sudo apt-get update running)

```
...
Hit http://security.ubuntu.com precise-security/universe Translation-en
Fetched 660 kB in 4s (133 kB/s)
Reading package lists...
 * Stopping web server apache2
 ... waiting    ...done.
mysql stop/waiting
Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no longer
required:
  apache2-utils libaprutil1-dbd-sqlite3 apache2.2-bin libapr1
libaprutil1-ldap
  apache2.2-common ssl-cert apache2-mpm-worker libaprutil1
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  apache2
0 upgraded, 0 newly installed, 1 to remove and 137 not upgraded.
After this operation, 29.7 kB disk space will be freed.
(Reading database ... 88139 files and directories currently installed.)
Removing apache2 ...
Reading package lists...
Building dependency tree...
Reading state information...
The following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 137 not upgraded.
Need to get 0 B/1,496 B of archives.
After this operation, 29.7 kB of additional disk space will be used.
Selecting previously unselected package apache2.
(Reading database ... 88135 files and directories currently installed.)
Unpacking apache2 (from .../apache2_2.2.22-1ubuntu1.4_i386.deb) ...
Setting up apache2 (2.2.22-1ubuntu1.4) ...
Server version: Apache/2.2.22 (Ubuntu)
Server built:   Jul 12 2013 13:38:27
Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no longer
required:
  libnet-daemon-perl libdbi-perl mysql-client-core-5.5 libdbd-mysql-perl
```

```
  mysql-server-5.5 libterm-readkey-perl mysql-client-5.5 libplrpc-perl
  mysql-server-core-5.5
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  mysql-client mysql-server
0 upgraded, 0 newly installed, 2 to remove and 137 not upgraded.
After this operation, 231 kB disk space will be freed.
(Reading database ... 88139 files and directories currently installed.)
Removing mysql-client ...
Removing mysql-server ...
Reading package lists...
Building dependency tree...
Reading state information...
The following NEW packages will be installed:
  mysql-client mysql-server
0 upgraded, 2 newly installed, 0 to remove and 137 not upgraded.
Need to get 0 B/22.2 kB of archives.
After this operation, 231 kB of additional disk space will be used.
Selecting previously unselected package mysql-client.
(Reading database ... 88133 files and directories currently installed.)
Unpacking mysql-client (from .../mysql-client_5.5.35-
0ubuntu0.12.04.1_all.deb) ...
Selecting previously unselected package mysql-server.
Unpacking mysql-server (from .../mysql-server_5.5.35-
0ubuntu0.12.04.1_all.deb) ...
Setting up mysql-client (5.5.35-0ubuntu0.12.04.1) ...
Setting up mysql-server (5.5.35-0ubuntu0.12.04.1) ...
mysql  Ver 14.14 Distrib 5.5.34, for debian-linux-gnu (i686) using
readline 6.2
 * Starting web server apache2
   ...done.
mysql start/running, process 7424
Reading package lists...
Building dependency tree...
Reading state information...
zlib1g-dev is already the newest version.
build-essential is already the newest version.
libruby1.9.1 is already the newest version.
libssl-dev is already the newest version.
ruby1.9.1 is already the newest version.
ruby1.9.1-dev is already the newest version.
ri1.9.1 is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 137 not upgraded.
ruby 1.9.3p0 (2011-10-30 revision 33570) [i686-linux]
ENVIRONMENT CLEANED AND PREPARED
```

deploy.rep . . . (log of deployment script)

```
Cloning into 'NCIRL'...

webpackage/

webpackage/README.md

...

... (webpackage unzipping)

...

webpackage/Utilities/logicaloperators.pl
```

```
webpackage/Utilities/NetmaskCalc.pl
d92edc59831a9cc593ea661393fe93e8 not equal to
01884844301631e86352ae033279db74
0 build errors
01-23-2014 18:21:41
0 integration errors
0 test errors
 * Stopping web server apache2
 ... waiting    ...done.
mysql stop/waiting
...
... (ensuring Apache and Mysql are uptodate)
...
mysql start/running, process 11161
 * Starting web server apache2
   ...done.
0 deploy errors

Error Summary
Build errors: 0
Intgration errors: 0
Test errors: 0
Deploy errors: 0

Mysql process is Running
Mysql is Listening
Remote Mysql TCP port is up
Apache process is Running
Apache is Listening
Remote Apache TCP port is up
```

logfile(sample).txt . . . (part of monitoring log)

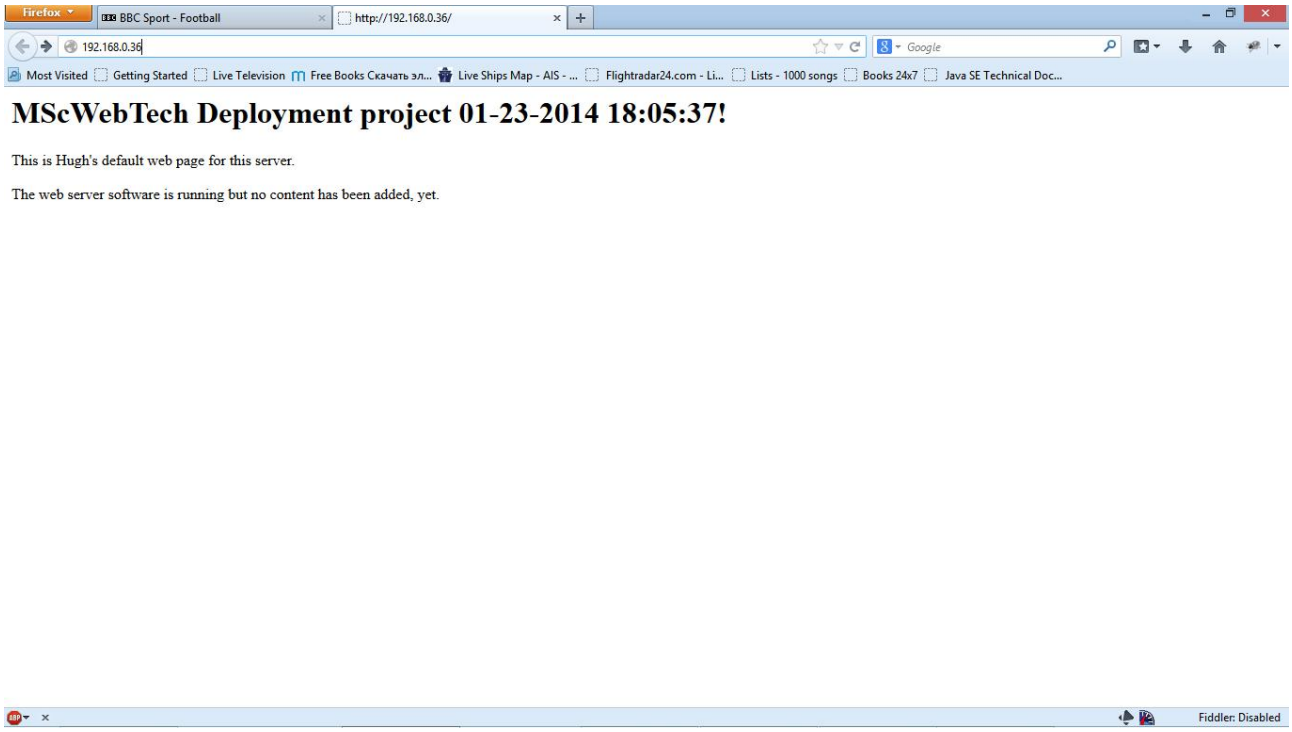
```
2014-01-23 18:05:01 Apache process is Running
2014-01-23 18:05:01 Apache is Listening
2014-01-23 18:05:01 Remote Apache TCP port is up
2014-01-23 18:05:01 Mysql process is Running
2014-01-23 18:05:01 Mysql is Listening
2014-01-23 18:05:01 Remote Mysql TCP port is up
2014-01-23 18:05:01 Processor usage is within normal range
```

```
2014-01-23 18:05:01 Memory usage is within normal range
2014-01-23 18:05:01 Total Disk Space          : 7739864
2014-01-23 18:05:01 Available Disk Space      : 5568320
2014-01-23 18:05:01 Used Disk Space           : 25%
2014-01-23 18:05:01 Network status            : up
2014-01-23 18:05:01 Current System info:
2014-01-23 18:05:01     host        ubuntu
2014-01-23 18:05:01     load        0.25, 0.08, 0.06
2014-01-23 18:05:01     total ram   496 MB
2014-01-23 18:05:01     free ram    32 MB
2014-01-23 18:05:01     ram used    463 MB
2014-01-23 18:05:01
2014-01-23 18:06:01 Apache process is Running
2014-01-23 18:06:01 Apache is Listening
2014-01-23 18:06:01 Remote Apache TCP port is up
2014-01-23 18:06:01 Mysql process is Running
2014-01-23 18:06:01 Mysql is Listening
2014-01-23 18:06:01 Remote Mysql TCP port is up
2014-01-23 18:06:01 Processor usage is within normal range
2014-01-23 18:06:01 Memory usage is within normal range
2014-01-23 18:06:01 Total Disk Space          : 7739864
2014-01-23 18:06:01 Available Disk Space      : 5568276
2014-01-23 18:06:01 Used Disk Space           : 25%
2014-01-23 18:06:01 Network status            : up
2014-01-23 18:06:01 Current System info:
2014-01-23 18:06:01     host        ubuntu
2014-01-23 18:06:01     load        0.27, 0.11, 0.07
2014-01-23 18:06:01     total ram   496 MB
2014-01-23 18:06:01     free ram    28 MB
2014-01-23 18:06:01     ram used    468 MB
2014-01-23 18:06:01
.
.
2014-01-23 18:16:01 Apache process is Running
2014-01-23 18:16:01 Apache is Listening
2014-01-23 18:16:01 Remote Apache TCP port is up
2014-01-23 18:16:01 Mysql process is Running
2014-01-23 18:16:01 Mysql is Listening
2014-01-23 18:16:01 Remote Mysql TCP port is up
2014-01-23 18:16:01 Processor usage is within normal range
```

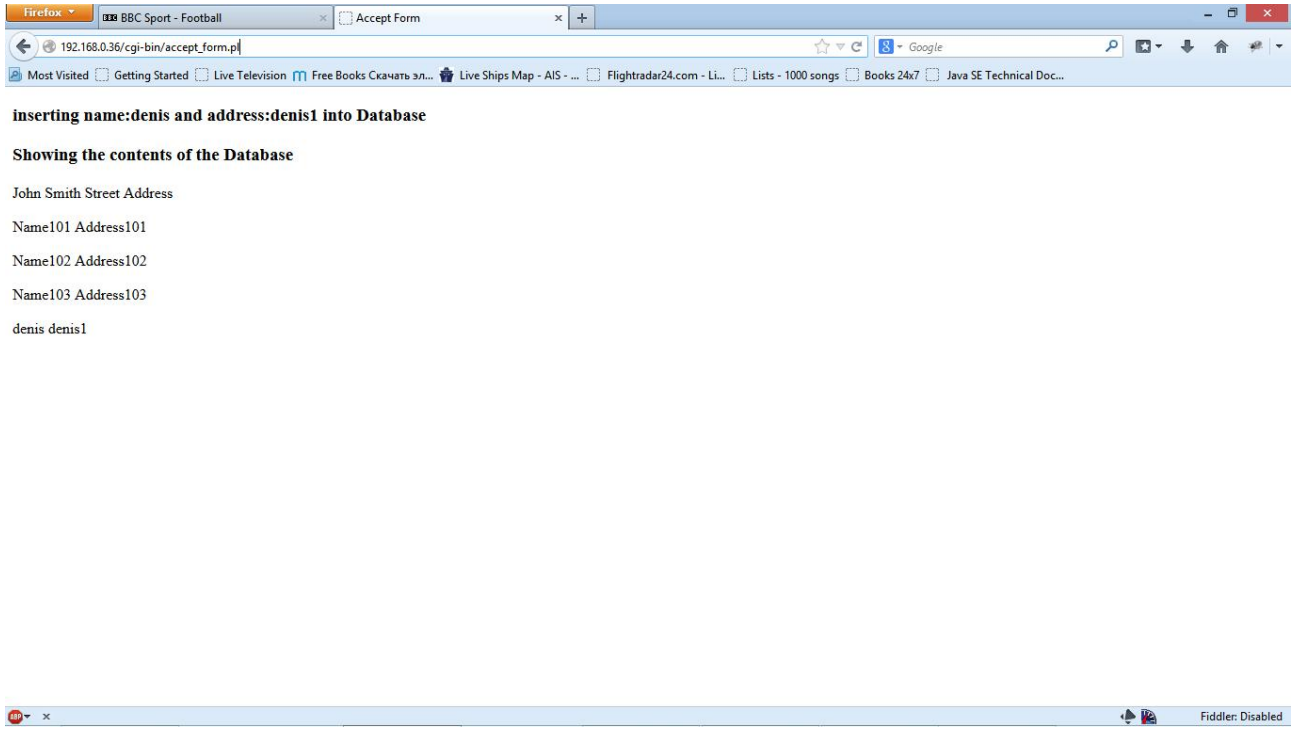
```
2014-01-23 18:16:01 Memory usage is within normal range
2014-01-23 18:16:01 Total Disk Space          : 7739864
2014-01-23 18:16:01 Available Disk Space      : 5568284
2014-01-23 18:16:01 Used Disk Space           : 25%
2014-01-23 18:16:01 Network status            : up
2014-01-23 18:16:01 Current System info:
2014-01-23 18:16:01     host        ubuntu
2014-01-23 18:16:01     load        0.00, 0.01, 0.05
2014-01-23 18:16:01     total ram   496 MB
2014-01-23 18:16:01     free ram    19 MB
2014-01-23 18:16:01     ram used    476 MB
2014-01-23 18:16:01
2014-01-23 18:17:01 CRITICAL Apache process is not Running
2014-01-23 18:17:01 CRITICAL Apache is not Listening
2014-01-23 18:17:01 CRITICAL Remote Apache TCP port is down
2014-01-23 18:17:01 Mysql process is Running
2014-01-23 18:17:01 Mysql is Listening
2014-01-23 18:17:01 Remote Mysql TCP port is up
2014-01-23 18:17:01 Processor usage is within normal range
2014-01-23 18:17:01 Memory usage is within normal range
2014-01-23 18:17:01 Total Disk Space          : 7739864
2014-01-23 18:17:01 Available Disk Space      : 5568272
2014-01-23 18:17:01 Used Disk Space           : 25%
2014-01-23 18:17:01 Network status            : up
2014-01-23 18:17:01 Current System info:
2014-01-23 18:17:01     host        ubuntu
2014-01-23 18:17:01     load        0.00, 0.01, 0.05
2014-01-23 18:17:01     total ram   496 MB
2014-01-23 18:17:01     free ram    23 MB
2014-01-23 18:17:01     ram used    472 MB
2014-01-23 18:17:01
2014-01-23 18:18:01 CRITICAL Apache process is not Running
2014-01-23 18:18:01 CRITICAL Apache is not Listening
2014-01-23 18:18:01 CRITICAL Remote Apache TCP port is down
2014-01-23 18:18:01 CRITICAL Mysql process is not Running
2014-01-23 18:18:01 CRITICAL Mysql is not Listening
2014-01-23 18:18:01 CRITICAL Remote Mysql TCP port is down
2014-01-23 18:18:01 Processor usage is within normal range
2014-01-23 18:18:01 Memory usage is within normal range
2014-01-23 18:18:01 Total Disk Space          : 7739864
```

```
2014-01-23 18:18:01 Available Disk Space      : 5568280
2014-01-23 18:18:01 Used Disk Space           : 25%
2014-01-23 18:18:01 Network status            : up
2014-01-23 18:18:01 Current System info:
2014-01-23 18:18:01     host        ubuntu
2014-01-23 18:18:01     load        0.00, 0.01, 0.05
2014-01-23 18:18:01     total ram   496 MB
2014-01-23 18:18:01     free ram    52 MB
2014-01-23 18:18:01     ram used    443 MB
2014-01-23 18:18:01
2014-01-23 18:19:01 Apache process is Running
2014-01-23 18:19:01 Apache is Listening
2014-01-23 18:19:01 Remote Apache TCP port is up
2014-01-23 18:19:01 Mysql process is Running
2014-01-23 18:19:02 Mysql is Listening
2014-01-23 18:19:02 Remote Mysql TCP port is up
2014-01-23 18:19:02 Processor usage is within normal range
2014-01-23 18:19:02 Memory usage is within normal range
2014-01-23 18:19:02 Total Disk Space          : 7739864
2014-01-23 18:19:02 Available Disk Space      : 5568276
2014-01-23 18:19:02 Used Disk Space           : 25%
2014-01-23 18:19:02 Network status            : up
2014-01-23 18:19:02 Current System info:
2014-01-23 18:19:02     host        ubuntu
2014-01-23 18:19:02     load        0.00, 0.01, 0.05
2014-01-23 18:19:02     total ram   496 MB
2014-01-23 18:19:02     free ram    20 MB
2014-01-23 18:19:02     ram used    475 MB
2014-01-23 18:19:02
```

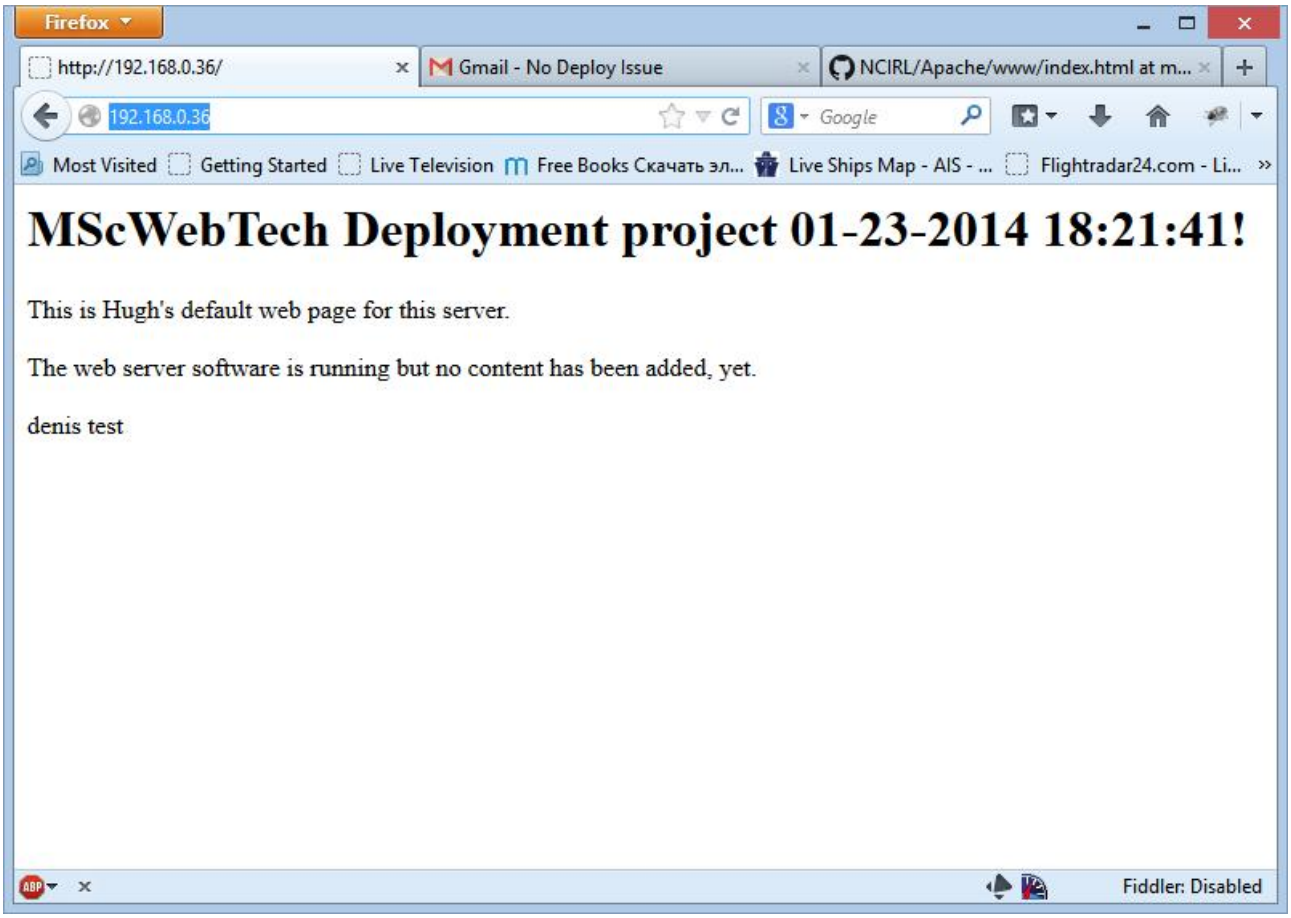
Static web page as first deployed



Dynamic web page when first deployed



Static web page after Step 12 showing modification to static html.



Dynamic webpage after Step 12 showing data persistence