greathosting.com – IT services hosting

Table of Contents

1 Introduction 1

2 Implemented features 1

2.1 Webhosting service 1
2.2 Accessing services 1
2.3 Provider's website 3
2.4 Registration 4
2.5 WordPress installation 6
2.6 Password reset 6
2.7 Securing bash script 9
2.8 Email notifications 9

3 Cloud implementation 11

4 Design and code 13

4.1 Use case 13
4.1.1 Use case: Create an account 14
4.1.2 Use case: Manage files 15
4.1.3 Use case: Manage database 16
4.1.4 Use case: Access system 17
4.1.5 Use case: Reset password 18
4.2 Sequence diagram: reset password 19
4.3 Pseudo code 23
4.3.1 Scheduled account creator script 23
4.3.2 Registration form submission 25
4.4 Database design 27
4.5 Form validation rules 29

5 Testing 31

5.1 Test ID #1 32
5.2 Test ID #2 36
5.3 Test ID #3 38
5.4 Test ID #4 42
5.5 Test ID #5 44
5.6 Test ID #6 46
5.7 Test ID #7 49
5.8 Test ID #8 51
5.9 Test ID #9 54
5.10 Test ID #10 58

6 Conclusion 60

7 Appendix iv

7.1 Usernames and passwords iv
7.2 Attached media iv
7.3 Cloud deployment availability iv

8 References v

9 Bibliography v

1 Introduction

This project implements a LAMP-based webhosting service according to the specification provided by the (fictional) business owner John Great. The project aims to fulfil all the required and desired features of the specification. This document provides a detailed description of the implemented features, as well as technical details about the implementation. It also discusses the ways that used to deploy and demonstrate the system. Finally, the document also aims to provide evidence of the functionality through various test scenarios.

2 Implemented features

2.1 Webhosting service

The webhosting service is delivered as a LAMP stack based on Ubuntu 16.04 LTS. Visitors can register for accounts using the provider's website, which is also hosted on the system. Each registered account holder able to host their files within the "public_html" folder of their home directory. These hosted files can be accessed by the public using either "*username*.greathosting.com" or "greathosting.com/~*username*" URL syntax, where the *username* is selected by the visitor at the time of registration. A MySQL database is also given to each registered account which can be accessed using the same username and password provided at registration. Account holders cannot access databases other than theirs.

2.2 Accessing services

Visitors with accounts can access the system by using an SSH client, which allows them to run non-privileged commands and manage their files. For manipulating files, it is also possible to connect using an SFTP client. Note that normal FTP is not enabled for security reasons.

Furthermore, the system also provides the account holders with a web-based solution to manage files, so no additional software installation is required for the account holders. This solution uses the Monstaftp[1] software, which is a free PHP based (S)FTP client. This is installed on the system and accessible by clicking on the "Manage files" menu on the provider's website.

The system also provides a web-based method to access and manage the database. This is achieved by PhpMyAdmin[2]. Visitors with accounts can access PhpMyAdmin by either using the "Manage database" menu on the provider's website or by navigating to "greathosting.com/phpmyadmin" directly.

[1] See: www.monstaftp.com

[2] See: www.phpmyadmin.net

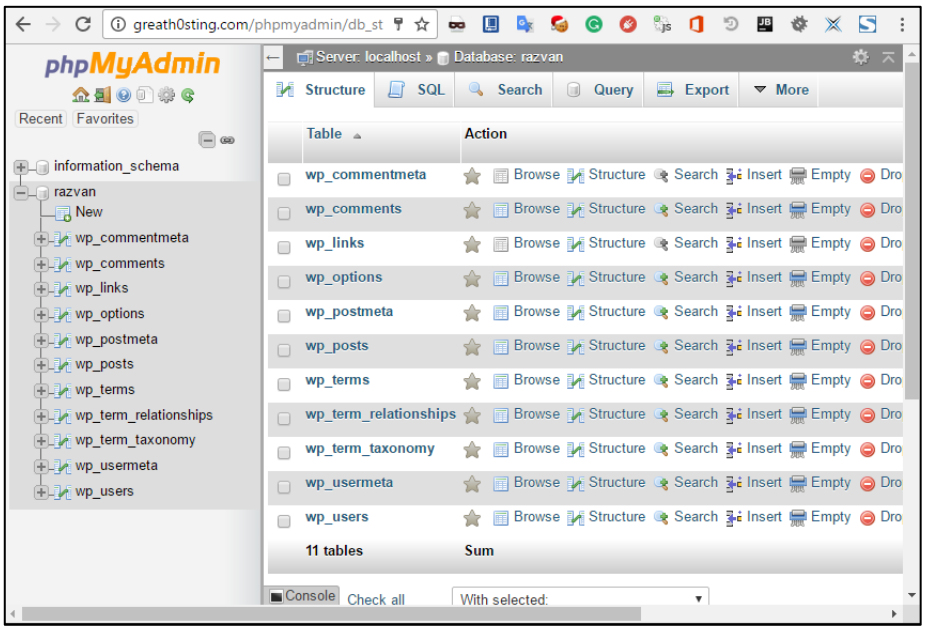


Figure 1 – PhpMyAdmin after the successful login for user "razvan". The figure also shows that the user has access to a database with the same name as his username.

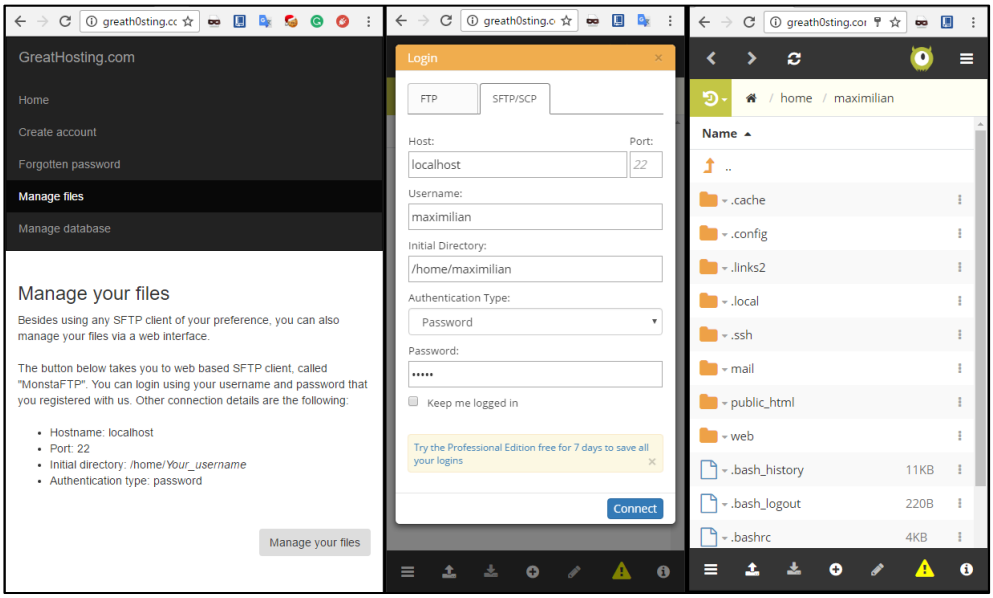


Figure 2 – Three steps of accessing the files using Monstaftp web access. The first step (left) provides a short help on how to login. The second step (middle) is the login form of Monstaftp and it is currently filled with login details. The third screen (on the right) is the content of the home folder after the successful login.

2.3 Provider's website

The service provider requires a website to promote its services and make them accessible to the visitors. This site is built from scratch using the Bootstrap framework[3]. The site's design is responsive and mobile-friendly. The provider's site is available on the "greathosting.com" URL. The related files are stored within the "/var/www/html/" and "/var/www/settings/" folders. The website has five main menus or functions:

- "Home": This page summarises the delivered features of this project. In the case of a real service provider, the purpose of this page would probably be about merchandising.
- "Create account": Takes the visitor to the registration page where she can create an account.
- "Forgotten password": Takes the visitor to the "reset password" page, where she can go through a process of changing her password.
- "Manage files": Takes the visitor to an informational page about how to access her files using the web-based SFTP client. This page also contains a link to the client.
- "Manage database": This is a direct link to the PhpMyAdmin's login page, which allows the registered visitor to manage her database.

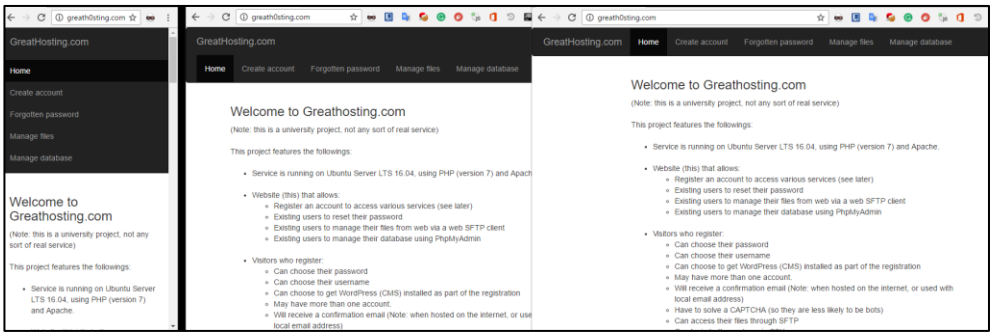


Figure 3 – The responsive design of the provider's site has three breakpoints: mobile (left), tablet (middle), and desktop (right) view.

[3] See: www.getbootstrap.com

2.4 Registration

Visitors can register an account by selecting the "Create account" menu on the "greathosting.com" website. Alternatively, they can visit "greathosting.com/register" link directly, which redirects to the same registration form. This redirection achieved by Apache's URL rewrite function.

Visitors can choose their username and password. The same username-password pair will be used by the system and the database. They also required to provide data to the other fields: first name, last name, and email address. A visitor is allowed to register multiple accounts with the same email address. To combat robots registering accounts, a captcha challenge must be solved. The captcha is generated using the free PHP-based Securimage software[4]. The captcha is re-generated with each page load.

The form uses both client-side and server-side validations. The client-side validation is based on HTML5 regex and does not require JavaScript. The server side validation (PHP) uses the same validation rules as the client-side, but also checks against existing users or reserved words. The rules are detailed in their own dedicated section within the "Design and code" section. Validation errors -just as other error types- are communicated to the visitor in a meaningful way. If the registration page needs to be reloaded due to an error, the previous contents of the registration fields will be populated back for convenience. The two exceptions to this are the password field and the captcha field, which are not populated back for security reasons. In regards to security, it would be a recommended future improvement to use HTTPS instead of HTTP for the registration page to prevent sending the sensitive visitor information unencrypted. Furthermore, the registration form uses the GET form submission method by default as it displays the form data in the browser's navigation bar and simplifies debugging. This also imposes a security risk; thus, it is made easily configurable to use the POST method instead, by changing only the "$usePost = false;" variable to *true* in the source code.

If the visitor provides valid data at registration, then the system saves it to the "greathosting" database's "customers" table. The password for accessing this database is encrypted with AES-128-CBC algorithm and stored in "/var/www/settings/settings.php" file, which is only accessible by the "www-data" user. Due to its location, this file is not served by Apache to the public. The key for the encryption is also stored in the same file. To further improve on the security of registration, the password of the new visitor account also gets encrypted before gets placed into the "customers" table.

Placing the registration form's data into the database is just the first step of the account creation. At this point, the visitor will see a green confirmation message asking her to wait for a confirmation email for the next steps. However,

[4] See: www.phpcaptcha.org

the actual creation of the system account and the database is done by the "Scheduled Account Creator" script, which is run by Cron in every minute. This script is running with root privileges to be able to achieve its goal. The script sends a notification email about the successful or unsuccessful account creation to the registered visitor's email address. To improve security, this script deletes the encrypted password stored in the "customers" database once the related account is created and its password's hash stored by the system. The script is written using PHP as opposed to being a BASH script. This allows sharing the database connection and encryption code with the registration page. Besides reusability, using the same language through the entire source code promotes consistency. The script's code makes only a small number of shell execution calls. A pseudo-code of the account creator script is provided in the "Design and code" chapter.

To enhance the uptime of the system, the account creator script chooses to reload the Apache configuration instead of restarting Apache. By doing so, there is no downtime when users are registering accounts, as the new incoming connections will be served with the new configuration.

The account creator script also creates logs for both successful and unsuccessful operations. The logfile is located at "/var/log/greathosting.log". It is worth to note that this log file can grow quickly. A possible future improvement could be implementing compression for the logfile.

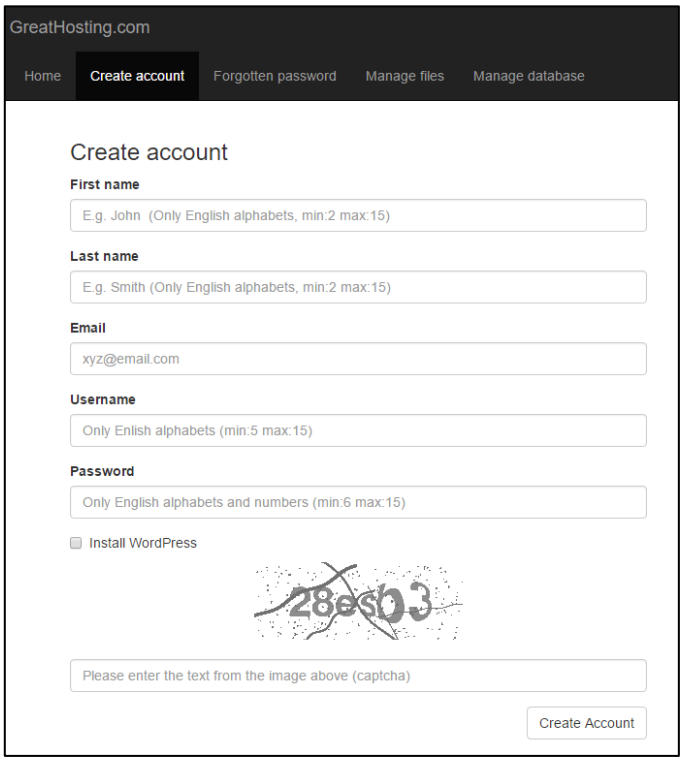


Figure 4 – The registration form on the provider's website.

2.5 WordPress installation

At the time of the registration, visitors' have the option to get the WordPress CMS installed on their account automatically. If this option is selected, then the Scheduled Account Creator script uses the "/etc/skelwordpress/" folder as a skeleton folder to create the home directory of the account. This skeleton directory contains the WordPress files in its "public_html" subdirectory. The script also configures the database connection settings for WordPress. It also creates a WordPress administrator user with the same username as the account. However, it does not setup a password for the admin account. Instead, a random password is generated and sent by email separate from the registration confirmation email. Setting up consistent password between WordPress and the account can be considered as a future improvement.

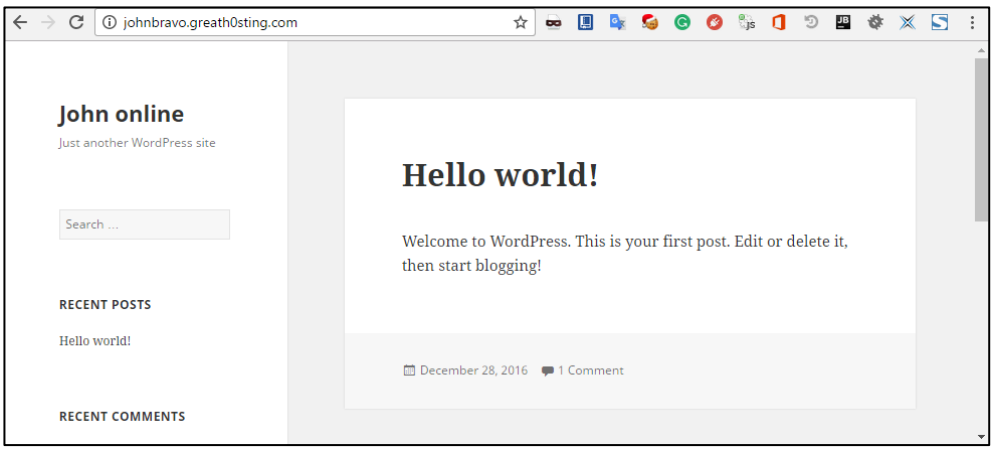


Figure 5 – WordPress is installed automatically as part of the registration for the user with "johnbravo" username.

2.6 Password reset

The system offers a password reset functionality for the typical scenario when the password gets forgotten. This is probably the most complex feature of the system and its understanding is supported with detailed sequence diagrams in the "Design and code" chapter.

The process starts with the visitor navigating to the "Forgotten password" page using the corresponding menu on the page and providing the email address associated with her account. If the input has valid email syntax, the system confirms the request and asks the visitor to check her inbox for the password reset email. For security reasons, the system does not reveal if the provided email address has or has not got an associated account. In the case when multiple accounts associated with the email address, the system sends separate password reset emails for each.

The system then generates the password reset email(s) which contain a password reset link. The link redirects back to the password reset page, but containing a token within the URL parameters. This valid token in the URL serves as proof that the visitor has access to the email address the account belongs to. The token is generated by taking the MD5 hash value of the timestamp when the password reset email was generated. The token is stored within the customer's record within the "customers" table, so it can be matched with the one received in the parameter. Tokens can expire in order to improve security. They can get expired once they are used or if not being used within 24 hours. The expiry date of the token also stored in the "customers" table.

Once the visitor clicks through the valid password reset link, she is able to provide a new password. The new password gets encrypted and stored in the "customers" table. Then the visitor is notified that the password reset should happen in a few minutes.

The reason for the delay is that the password changing procedure requires root privileges and it is done by the same script which executes the account creations. The script decrypts the password and sets it for the related system account and database user. The admin password for WordPress installation is not changed. Once the password change executed, the script removes the encrypted password from the "customers" database.

It's worth to mention that the input fields of the password reset scenario are using the same validation rules as with the account registration. This is achieved by re-using code via classes.

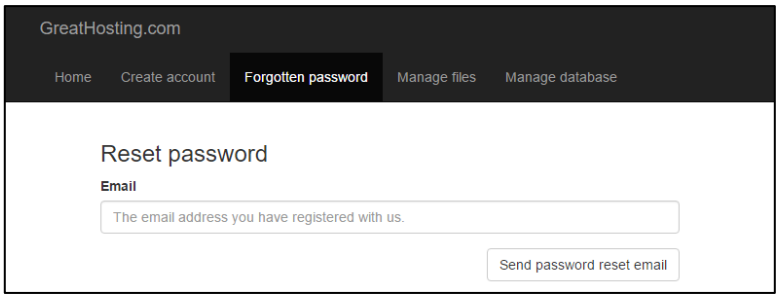


Figure 6 – The visitor needs to provide the email address associated with her account as the first step of the password reset.

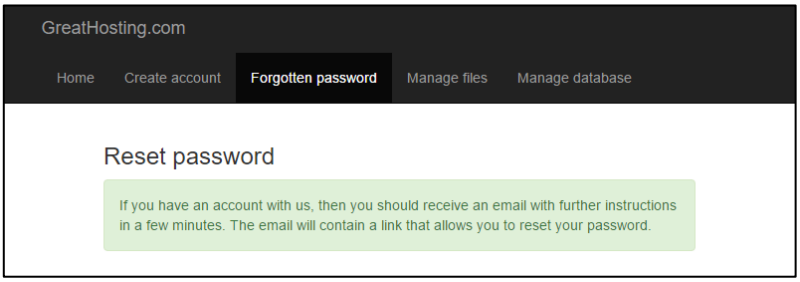


Figure 7 – The system accepts the email address and ask the visitor to check her email account.

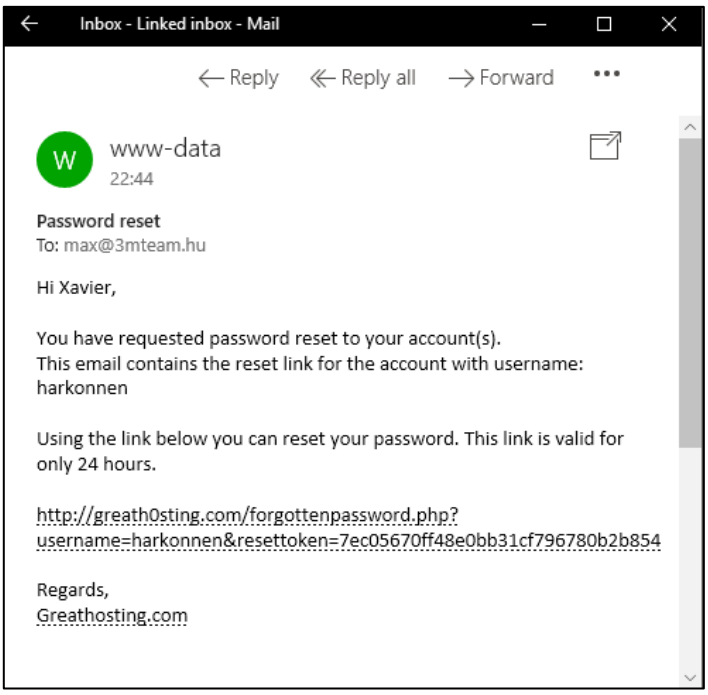


Figure 8 – The password reset email sent to the user with "harkonnen" username. The email contains the password reset link which is valid for 24 hours. Note the token value within the link.

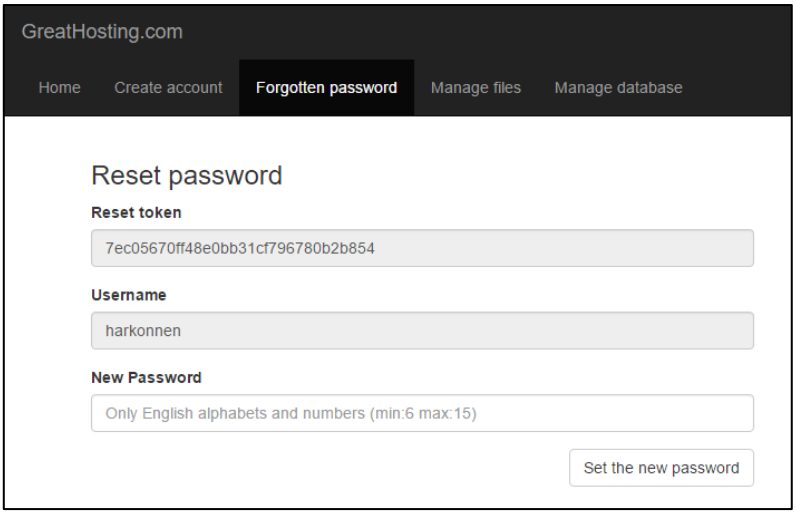


Figure 9 – After clicking through the password reset email the visitor can provide the new password. Note that the two greyed fields are there only for confirmation and the visitor is unable to edit them.

2.7 Securing bash script

A short BASH script was created to simplify the task of ensuring secure file permission on the system. The script restricts the access of the file "/var/www/settings/settings.php" to be only accessible by Apache. This read-only file contains the encryption key and the encrypted password for the "greathosting" database. Furthermore, the script ensures that the Scheduled Account Creator script accessible only for the Root user. The script file is located at "/var/www/secureconfiguration.sh".

2.8 Email notifications

The password reset flow is not the only scenario where email is sent to the account owner. At the end of the registration, the visitor will receive a notification email. This can be one of the two types: notification of an error or notification of success. In the case of the unsuccessful registration, the visitor receives a plaintext email asking her to try the registration again or contact the system administrator.

In the case of a successful registration, the visitor receives an email containing information on how to use the services. To improve its appearance, this email is in a HTML format. The HTML design is based on the Foundation for Emails[5] library. However, an alternative, plaintext based confirmation email was also implemented. By editing the Scheduled Account Creator script's line of "$useResponsiveEmail = true;" to be *false*, it is easy to switch to plaintext confirmation emails.

Emails sending relies on the local configuration of Postfix. As chapter "Cloud implementation" explains, the local VM is limited to send emails only within its own domain of "greathosting.com". This could make difficult viewing the HTML-based confirmation emails, as there is no GUI-based email client installed on the system. To tackle this problem, Dovecot is configured to allow POP3 access to emails. This allows retrieving the emails remotely, from a GUI-based email client. However, the current configuration of Dovecot works with plaintext-based password authentication. This method is less secure, and it is recommended to switch to an encrypted method as a potential future development.

[5] See: http://foundation.zurb.com/emails.html

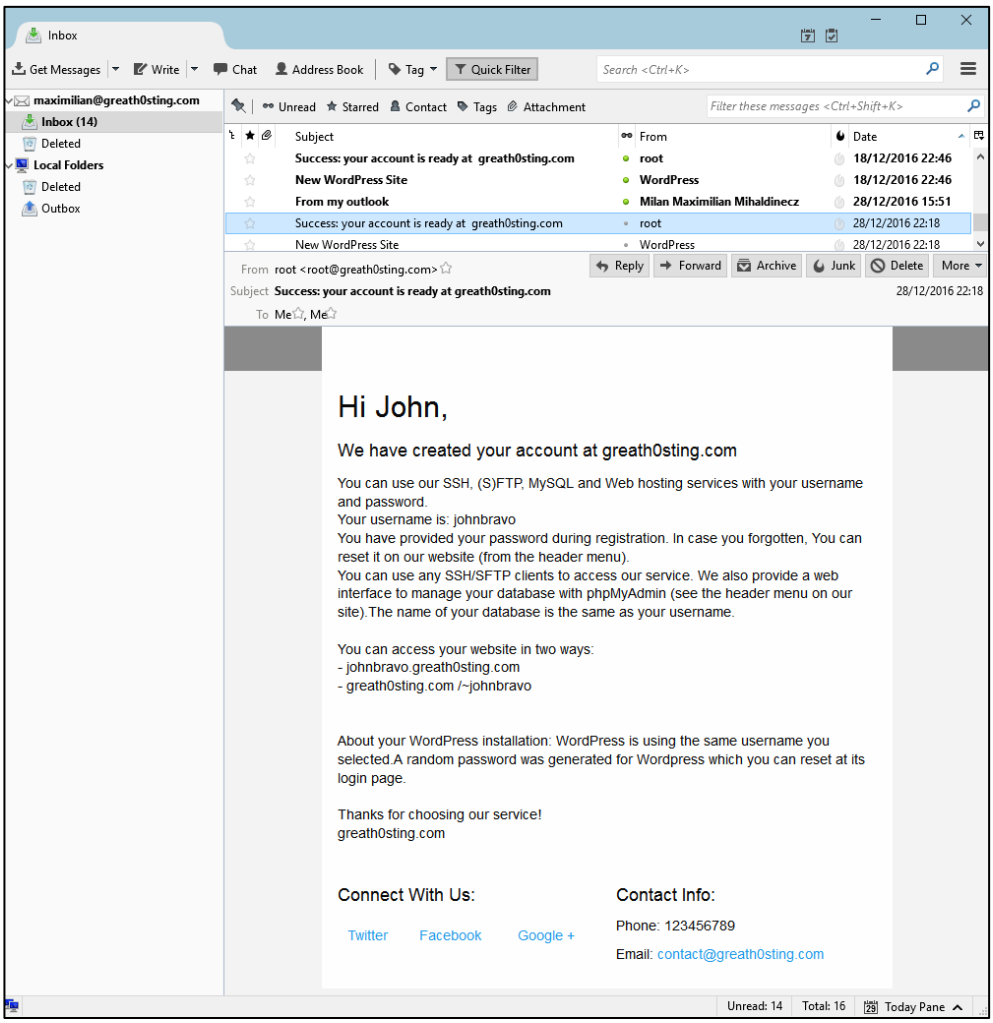


Figure 10 – An HTML-based notification email about successful account creation for the user with username "johnbravo". The email address used at registration was a local email address for user "maximilian". The email is retrieved by using the Thunderbird[6] email client.

[6] See: www.mozilla.org/en-GB/thunderbird/

3 Cloud implementation

The system described in the previous chapter has also been deployed as a virtual machine in the Cloud. While this deployment is full-featured, it is not fully identical to the attached virtual machine version. This is due to the differences in technical limitations of a simulated and the Internet network.

The difference in domain names is one these technical limitations. The provided specification would require the usage of "greathosting.com" as the domain name. However, this domain name is already in use on the Internet. Therefore, the similar "greath0sting.com" domain name used as a replacement (note: the letter "o" has been replaced with the number zero). In order to acquire a domain name, it must be registered with an accredited Domain Name Registrar organisation. The "greath0sting.com" was registered through GoDaddy[7].

The other difference in the two deployment lies with email sending. The local VM version's Postfix service was configured to serve local deliveries only, while the Cloud deployed variant uses SMTP to be able to send emails to other domains as well. This is achieved by selecting different pre-configuration offered by the "mailutils" tool: the local VM uses the "Local only" configuration and the Cloud version uses the "Internet Site" one. Thus, the confirmation or password reset emails will work with any 'common' email address for the Cloud deployment in theory. In practice, testing has found that most email providers will reject these emails as spams. The probability of this happening could be lowered by adding certificates or using a trusted SMTP relay. However, for demonstration purposes, it is sufficient to add "greath0sting.com" to the whitelist of the recipient's spam filter(s).

The deployment is hosted in the Microsoft Azure Cloud as an "A1 Basic" instance. It's worth to point out that many other providers could be selected based on the functional requirements. Considering non-functional requirements such as privacy, security, or portability would possibly allow prioritising between providers, but such requirements were not provided in the specification.

The process of the deployment was based on using an automated, clean installation of the Ubuntu Server OS and then replicate the configuration manually as done with the local VM. An alternative method could be to convert the existing local VMware VM to a Hyper-V VM and upload it to Azure (Tacacho, 2016). This may be an easier and less error prone approach than the manual one in the case of multiple VMs.

By default, the Azure's network policy blocks all communication with the fresh deployment, except using SSH on port 22. This has been modified to allow communication on all ports between 1 and 1000 in order allow access to other

[7] See: www.godaddy.com

services, such as HTTP(S) or POP3. A more secure solution would be to explicitly allow communication only on those ports where the desired services are running, such as 80, 443, and 110.

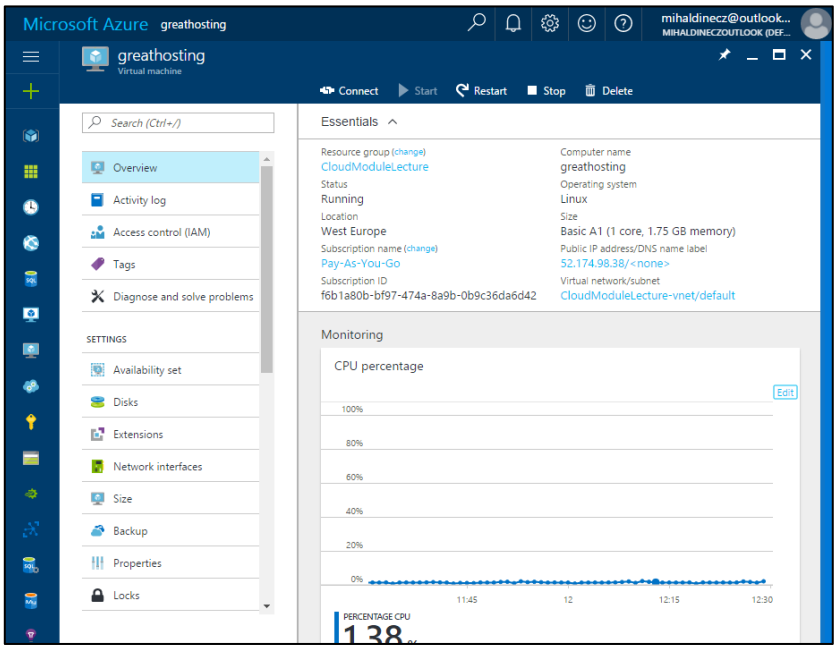


Figure 11 – The Microsoft Azure Cloud deployment's configuration panel. The instance is running.

The deployment is using a static IP address. This is important for the nameservers provided by the Domain Name Registrar to be able to resolve the domain name without needing reconfiguration. Azure provided "52.174.98.38" as the static IP, and it is placed into the DNS resource records as the only address ("A") record.

Type	Name	Value	TTL
A	@	52.174.98.38	600 seconds
CNAME	*	@	1 Hour
CNAME	_domainco...	_domainconnect.gd.doma...	1 Hour
NS	@	ns51.domaincontrol.com	1 Hour
NS	@	ns52.domaincontrol.com	1 Hour

Figure 12 – DNS resource record settings used on the Domain Name Registrar's site for resolving the "greath0sting.com" domain.

4 Design and code

4.1 Use case

Use cases defined to model and describe the users' interaction with the system. Five use cases and an actor has been identified. This section provides detailed use case descriptions for all the five identified use cases.

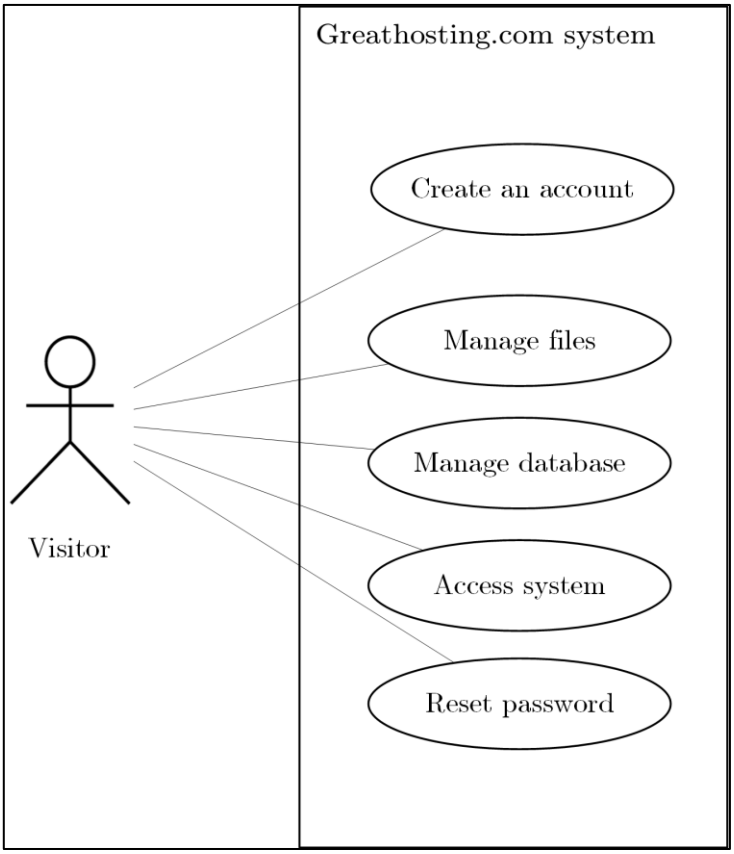


Figure 13 – Use case diagram for the greathosting.com system.

4.1.1 Use case: Create an account

Use Case: Create an account	ID: 1	Importance level: High
Primary actor: Visitor	Use case type: Detail, essential	

Stakeholders and interests:
Visitor – Wants to create a webhosting account on the system.

Brief description:
This use case describes how a new account can be created.

Normal flow of events

Actor action	**System response**
1. Visitor navigates to www.greathosting.com/register address.	2. System displays the registration form.
3. Visitor provides its First Name, Last Name, Email address, Desired username, password, and Captcha text.	4. System accepts the input and saves them to a new database record. Then it shows confirmation message to the user and advises her to wait for the confirmation email.
7. Customer receives the confirmation email which includes instructions on how to access her account.	5. System check the customer database each minute and notices that a new account needs to be created. 6. System creates the new system account and a new database associated with it. Then it sends a confirmation email to the customer.

Alternate/exceptional flows

1b: Visitor may navigate to www.greathosting.com and uses the header menu to get on the registration page.

4b: The provided data fails the validation. The system will re-display the registration form and an appropriate error message. The form is pre-populated with the previously submitted data, except for the password and captcha fields. A new Captcha image is shown on the registration form. The user is required to retry "step 3".

6b: System has a serious error and unable to create an account. It logs the error but unable to notify the customer. The customer will not be to access the system.

6c: System has a moderate error and unable to create an account. It logs the error and sends a notification email to the customer. The customer may re-try the registration or contact the website owner.

Table 1 – Detailed description of the "Create an account" use case.

4.1.2 Use case: Manage files

Use Case: Manage files	ID: 2	Importance level: High
Primary actor: Visitor	Use case type: Detail, essential	

Stakeholders and interests:
Visitor – Wants to access the system to view, modify, or delete files and folders under her account.

Brief description and pre-requirements :
This use case describes how a visitor can access an interface to be able to manage her files or folders. The visitor must have an account created through use case ID #1.

Normal flow of events

Actor action	System response
1. Visitor navigates to www.greathosting.com/managefiles.php	2. System displays an informational page about how to access the web-based file manager. Also displays a link to the file manager.
3. Visitor clicks on the link of the web-based file manager.	4. System redirects to greathosting.com/mftp/ and asks for the connection details.
5. Visitor selects SFTP connection from the menu and provides the hostname, account's username, account's home directory, and password.	6. System displays her home folder and offers a graphical interface to manage her files and folders.

Alternate/exceptional flows

1B: Visitor may navigate to www.greathosting.com and uses the header menu's "Manage files" option.

1C: Visitor uses an SFTP software to connect to greathosting.com with her account's username and password. The SFTP software provides the means to display and manage files and folders. In this flow, no other steps needed.

Table 2 – Detailed description of the "Manage files" use case.

4.1.3 Use case: Manage database

Use Case: Manage database	ID: 3	Importance level: Moderate
Primary actor: Visitor	Use case type: Detail, essential	

Stakeholders and interests:
Visitor – Wants to access the database associated with her account to view or modify its content.

Brief description and pre-requirements :
This use case describes how a visitor can access an interface to be able to manage her database. The visitor must have an account created through use case ID #1.

Normal flow of events

Actor action	System response
1. Visitor navigates to www.greathosting.com/phpmyadmin	2. System displays a login page prompting for username and password.
3. Visitor provides the username and password related to her account.	4. The authentication is successful, and the system redirects her to an new interface. This interface allows her to manipulate her database by either using SQL queries or the graphical interface.
5. Visitor may view or modify the database's content.	

Alternate/exceptional flows

1B: Visitor may navigate to www.greathosting.com and uses the header menu's "Manage database" option.

4B: The authentication is not successful. The system displays the login page again with an appropriate error message.

Table 3 – Detailed description of the "Manage database" use case.

4.1.4 Use case: Access system

Use Case: Access system	ID: 4	Importance level: High
Primary actor: Visitor	Use case type: Detail, essential	

Stakeholders and interests:
Visitor – Wants to securely access the webhosting system in order to manage files on her account or execute applications.

Brief description and pre-requirements :
This use case describes how a visitor can securely access the webhosting system using SSH. The visitor must have an account created through use case ID #1.

Normal flow of events

Actor action	System response
1. Visitor opens an SSH connection to the greathosting.com host. The visitor can choose an SSH client she prefers. 3. Visitor provides the username and password related to her account. 5. Visitor may use the available Linux/ Unix commands to manage her files.	2. System prompts for username and password. 4. The authentication is successful, and the system provides a prompt to the user. The visitor start directory will be her home directory. Visitor does not have root or sudo privileges.

Alternate/exceptional flows

4B: The authentication is not successful. The system requires to type in the password again.

Table 4 – Detailed description of the "Access system" use case.

4.1.5 Use case: Reset password

Use Case: Reset password	ID: 5	Importance level: Low
Primary actor: Visitor	Use case type: Detail, essential	

Stakeholders and interests:
Visitor – Wants to change the password of an account which is associated with her email address. She is may or may not be aware of the current password.

Brief description and pre-requirements :
This use case describes how a visitor can reset her system accounts' password, including the database one. The visitor must have an account created through use case ID #1.

Normal flow of events

Actor action	System response
1. Visitor navigates to www.greathosting.com/forgottenpassword.php	2. System prompts for the email address associated with her system account.
3. Visitor provides her email address and submits the request.	4. System informs the visitor that she should receive an email with further directions, given she provided the correct email address.
5. Visitor checks her associated email account and notices the password reset email. She navigates to the password reset link contained in this email.	6. System shows the visitor the account's username she is about to change password for, and asks for the new password.
7. Visitor provides the new password and submits the form.	8. System accepts the new password and shows a confirmation message.
	9. System sets the new password for the account and the related database.

Alternate/exceptional flows

1B: Visitor may navigate to www.greathosting.com and uses the header menu's "Forgotten password" option.

5B: Visitor does not receive an email as she provided an incorrect email address or due to an error in email delivery. She is unaware of this issue. She may or may not try to go back to step #1.

5C: Visitor has more than one account registered with the same email address. She receives multiple password reset emails, one for each account. She will use the password reset link which belongs to the account that needs to be reset.

6B: More than 24 hours passed since the password reset link's creation and the system considers it to be expired. The visitor will be presented with step #2 without any error message.

6C: The password reset link has already been used to successfully reset the password, and the system considers it to be expired. The visitor will be presented with step #2 without any error message.

8B: The new password is rejected as it is using an invalid format. Step #6 will be shown again by the system with an appropriate error message included.

Table 5 – Detailed description of the "Reset password" use case.

4.2 Sequence diagram: reset password

The "reset password" use case is the most complex of the five use cases. Therefore, it is selected to be explained through sequence diagrams. To support the format of the printed media, the describing sequence diagram has been broken down into four figures. Each figure explains specific steps from the reset password use case. The diagrams describe the normal flow of events, without any alternate flows.

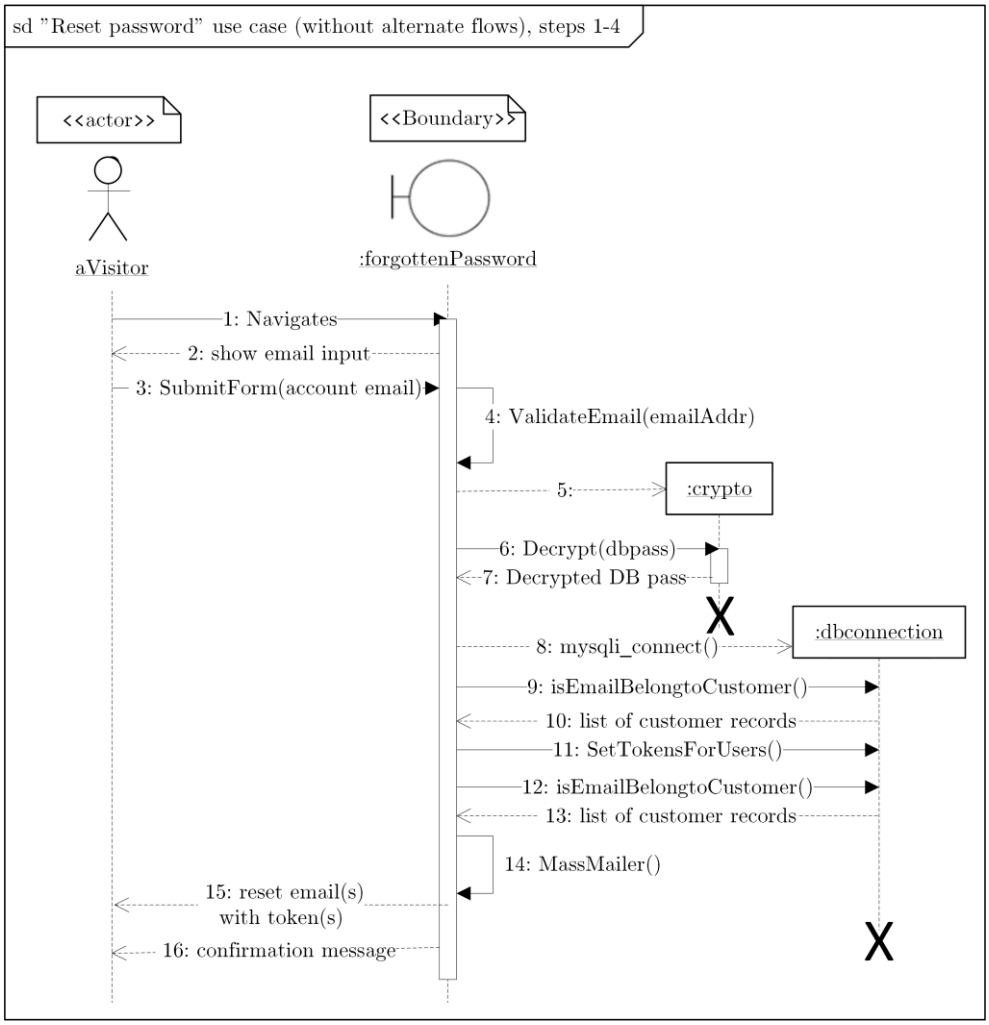


Figure 14 – Password reset use case steps 1-4: visitor provides the email address associated with the account she needs the password reset for. The system sends password reset email(s).

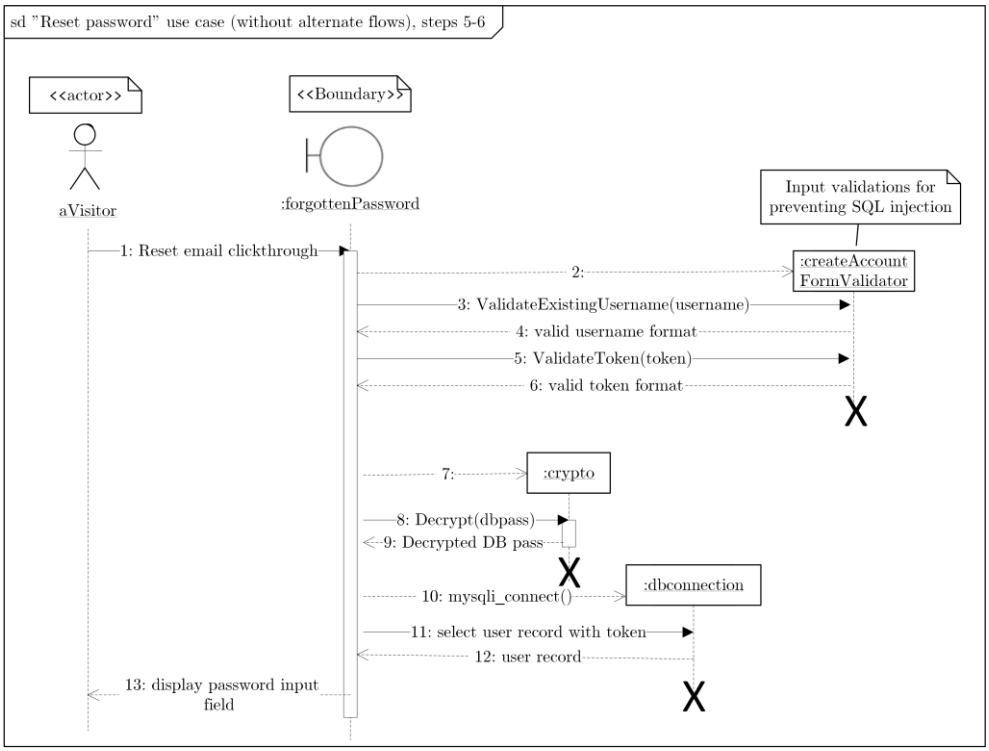


Figure 15 – Password reset use case steps 5-6: Visitor clicks through the password reset link received in the password reset email. The system asks her to provide the new password.

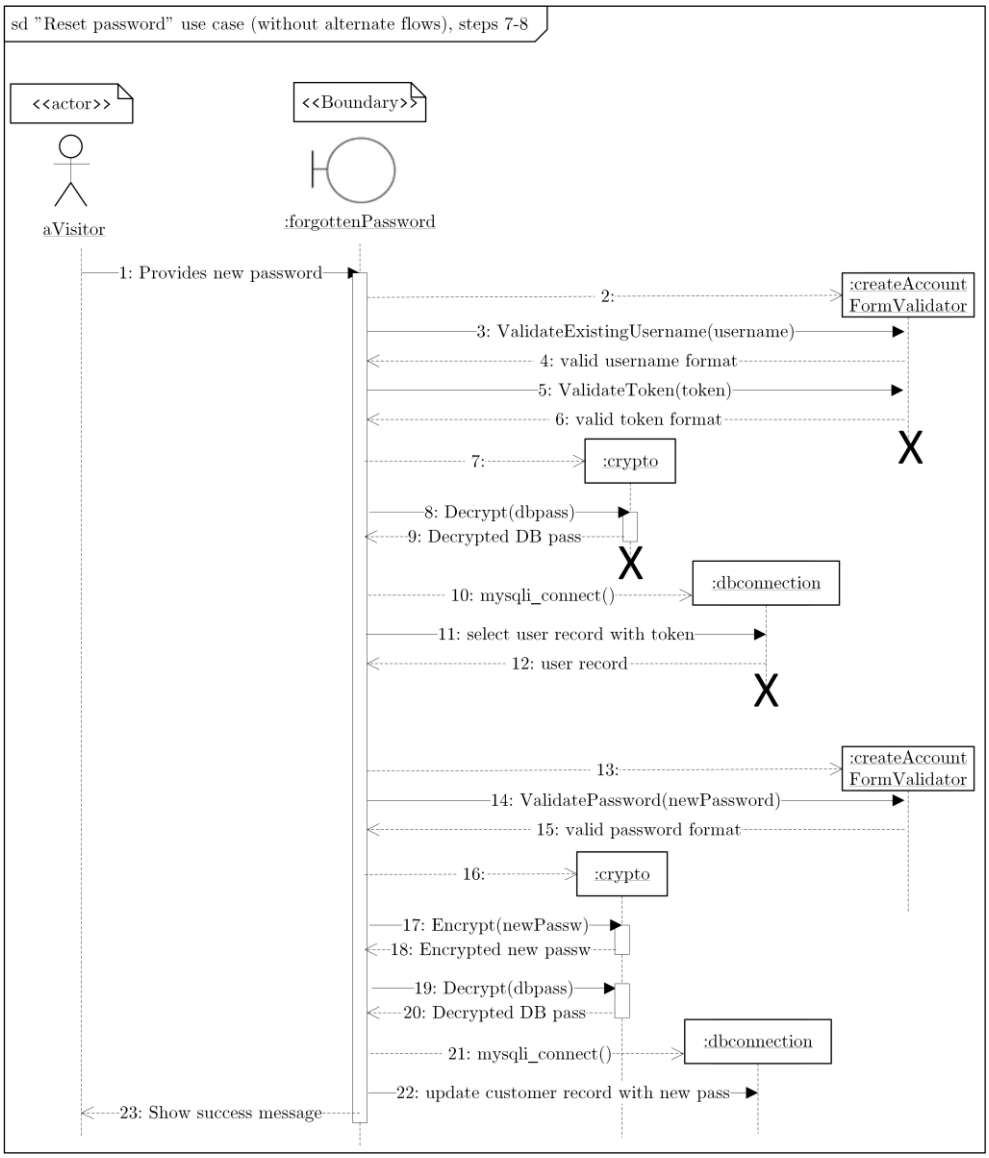


Figure 16 - Password reset use case steps 7-8: The visitor provides the new password and the system shows a confirmation message.

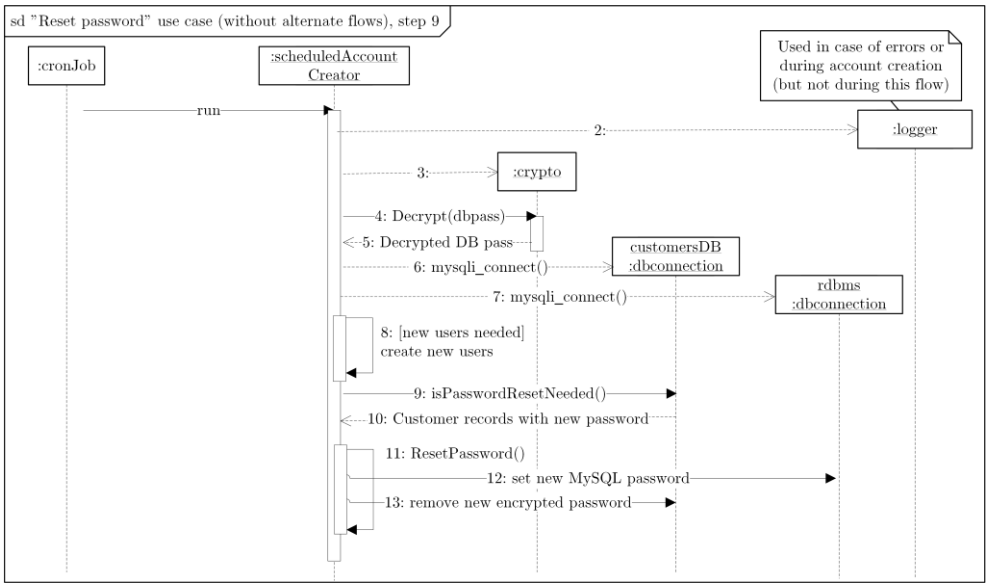


Figure 17 – Password reset use case step 9: The system changes the system password and database password for the requested account.

4.3 Pseudo code

4.3.1 Scheduled account creator script

```
Use Responsive Email Format = true; //instead of plain
Open logfile for writing;

Try
{
      Read the encrypted database password from file;
      Decrypt the encrypted password;
      Connect to the customer's database;
} catch errors
{
      Log if error occurs and then terminate the script immediately;
}

Select registered customers from the customer's database where the
password field is not empty;

Loop (through each customer record where password is not empty)
{
      Decrypt the stored encrypted user password;
      If (record shows Wordpress installation needed)
      {
            Select the skeleton directory with Wordpress files when
            creating the system user;
      }
      Create new system user using the username and (decrypted)
      password from the records;
      If (user created successfully)
      {
            Send confirmation email to the user; Log success;
      } else
      {
            Send email with error message to the user; Log error;
      }
      Update the customer record to remove the stored encrypted
      password;
      Create a new database with the same name as the username;
```

```
        Create a new database user with the same username and
        decrypted password used with system user creation.
        Set all privileges to true on the new database for the newly
        created user;
        Log the database creation success or error if occurred.
        Create the Apache virtual host for the new user using the
        username as the subdomain;
        Log both success or error during the Apache virtual host
        creation;

        If (record shows Wordpress installation needed)
        {
                Reload Apache configuration to enable the virtual host;
                Edit the Wordpress "wp-config.php" configuration file
                within the user's home directory to contain the user's
                database connection details: database name, username,
                and password;
                Edit the "myinstall.php" Wordpress file within the
                user's home directory to:
                        Set the admin username the same as system
                        username;
                        Set the admin user's email address to the user's
                        registration email address;
                        Set the site's title as "UserFirtName online";
                Run the "myinstall.php";
                Delete the "myinstall.php";
        }
}
Select registered customers where password reset needed;
Loop (through customer records where password reset needed)
{
        For the username in the record:
                Decrypt the newly provided password;
                Set the system password to the new password;
                Set the MySQL password to the new password;
                Delete the encrypted new password from the customer
                database;
}
Close logfile and database connections;
Reload apache configuration if new user(s) was created;
```

4.3.2 Registration form submission

```
If (form validated successfully)
{
        Try
        {
                Read the encrypted database password from file;
                Decrypt the encrypted password;
                Connect to the customer's database;
        } catch errors
        {
                Set error message and DisplayRegFormWithError();
                End;
        }

        Try
        {
        Create a new customer record in the customer's database and:
                Set username field from the form input
                Set email address field from the form input
                Set firstname field from the form input
                Set lastname field from the form input
                Set password field by encrypting the form input
                Set the Wordpress Boolean field from the form input
        } catch errors
        {
                Set error message and DisplayRegFormWithError();
                End;
        }

        Close customer database connection;
        DisplayRegistrationSuccess();
}
Else
{
        DisplayRegFormWithError();
}
```

```
Function DisplayRegFormWithError()
{
        Display the registration page with registration form;
        Display the validation error message;
        Populate these registration form fields with the previously
        submitted data: First name, last name, email address,
        username;
}

Function DisplayRegistrationSuccess()
{
        Display the registration page without the registration form;
        Display successful account creation message;
}
```

4.4 Database design

A database is created to store information of the registered visitors and to act as an intermediary between the Scheduled Account Creator script and the web-facing pages. Therefore, the data requirement analysis described two types of information which needed to be stored: permanent user details and temporary variables.

The permanent user details are collected during the registration process: username, email address, first name, last name, and WordPress installation request. These are required fields and needed to allow features such as notification emails and password resets.

The temporary fields are stored only until the Scheduled Account Creator picks them up and removes after its task completed. Such data is the password, reset token, reset token expiration date, and new password (for password reset).

Based on this analysis, it can be assumed that the size of this table will be small due to the few types of permeant data. It is also expected that the temporary variables will rarely be populated during the lifetime, allowing to further conserve space. As there is no indication for this in the specification, it is assumed that no other types of data needed to be captured in the future.

Considering all this above, it has been decided that all the data will be kept in one single table. The advantage of this approach is in the simplicity of queries which would read or manipulate the table. On the negative side, this approach decreases flexibility as adding significant amount new type of customer data would impact performance negatively. An alternative, more flexible solution could have been splitting the stored data into separate tables.

Username has been selected as the primary key of the table as it is not possible to have two accounts in the system with the same username. Alternatively, the email could have been the primary key, but that would prevent a visitor to have multiple accounts.

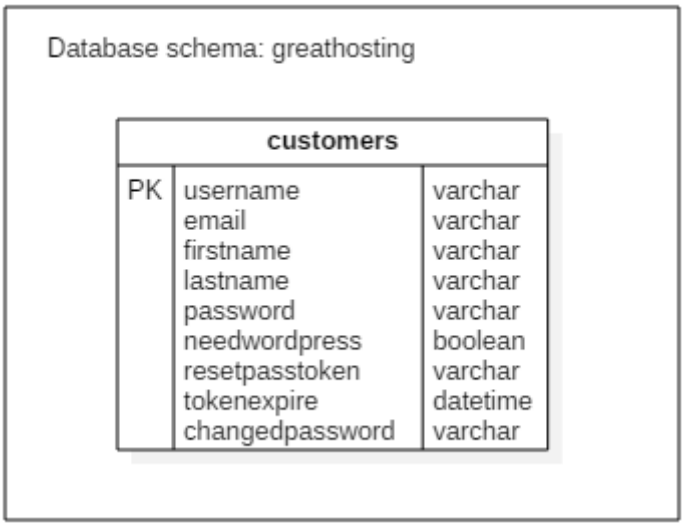


Figure 18 – Schema for the "greathosting" database. The only table it contains is the "customers" table.

username	email	firstname	lastname	password	needwordpress	resetpasstoken	tokenexpire	changedpassword
amazon	maximilian@greath0sting.com	Xena	Amazon	NULL	1	NULL	NULL	NULL
atreides	max@3mteam.hu	Leto	Atreides	NULL	1	01472ca36cd85bfa91af9566c991aa26	2016-12-29 22:44:07	NULL
beethoven	maximilian@greath0sting.com	Ludwig	Beethoven	NULL	1	NULL	NULL	NULL
disturbed	maximilian@greath0sting.com	Disturbed	Decadence	NULL	1	NULL	NULL	NULL
harkonnen	max@3mteam.hu	Xavier	Harkonnen	NULL	1	7ec05670ff48e0bb31cf796780b2b854	2016-12-29 22:44:07	NULL
hercules	maximilian@greath0sting.com	Hercules	Hercules	NULL	1	NULL	NULL	NULL
jackson	maximilian@greath0sting.com	Michael	Jackson	NULL	1	NULL	NULL	NULL
johnbravo	maximilian@greath0sting.com	John	Bravo	NULL	1	NULL	NULL	NULL
kriegor	maximilian@greath0sting.com	Xan	Kriegor	NULL	1	NULL	NULL	NULL
maniac	maximilian@greath0sting.com	Mercy	Maniac	NULL	1	NULL	NULL	NULL
ordos	maximilian@greath0sting.com	Freya	Ordos	NULL	1	NULL	NULL	NULL
razvan	razvan.dinita@anglia.ac.uk	Razvan	Dinita	NULL	1	1f5b5411eb6b995849f7d873db4c28d2	2016-12-17 13:59:17	NULL
testuser	maximilian@greath0sting.com	Test	User	NULL	1	NULL	NULL	NULL

Figure 19 – An example content of the "customers" table.

4.5 Form validation rules

This section describes the client-side and server-side validation rules used on the registration form and password reset form.

First name, Last name	
Validation rule	**Rationale or comments**
Required	User must provide it.
Minimum characters: 2	To prevent user errors.
Maximum characters: 15	To prevent overflow attack.
Only: a-z and A-Z characters	Only English alphabets. To prevent mistyping or malicious characters. This may be negative experience for users with international names.

Table 6 – Validation of first name and last name

Email	
Validation rule	**Rationale or comments**
Required	User must provide it.
Maximum characters: 60	To prevent overflow attack.
Only email format.	Uses browsers' built-in HTML5 email format validation on the client-side. Uses PHP's FILTER_VALIDATE_EMAIL filter on the server-side.

Table 7 – Validation of the email address

Password	
Validation rule	**Rationale or comments**
Required	User must provide it.
Minimum length: 6	To enrich password strength.
Maximum length: 15	To prevent overflow attack.
Only: a-z and A-Z and 0-9 characters	Prevent injection attacks which would use special characters.

Table 8 – Validation of the password

Username	
Validation rule	**Rationale or comments**
Required	User must provide it.
Minimum characters: 5	Promotes choosing non-system usernames
Maximum characters: 15	To prevent overflow attack.
Only: a-z characters	Only lowercase English alphabets to be valid Linux system username and subdomain name.
Not existing user	The username should not be in use already on the Linux system. This includes system usernames, such as "root". This is validated on server side only.
Cannot be a reserved word: greathosting, phpmyadmin, mysql, greathostingdbuser, rootonly, fonts, securimage, register, mftp	Checked on the server side only. A list of reserved names for existing MySQL users, or installed web software, or folders in the main web directory.

Table 9 – Validation of the username

5 Testing

This section provides evidence of the functionality of the system by conducting manual test scenarios. To replicate the tests executed on the VM, it is necessary to edit the host OS's "hosts" file to point to the IP address used by the VM. This must be set for the main "greathosting.com" domain and its subdomains as well. This step is assumed and not discussed within the test scenarios.

Test ID	Objective	Result
1	Register a new account on the VM	Pass
2	Access an email account on the VM using a POP3 email client.	Pass
3	Reset the password of an account on the VM	Pass
4	Access the system via SSH with a registered account on the VM	Pass
5	Access the files on the VM using a desktop-based SFTP client.	Pass
6	Access the files on the VM using the web-based SFTP client	Pass
7	Access the database with PhpMyAdmin on the VM	Pass
8	Confirm the client-side and server-side registration form validation on the VM	Pass
9	Register a new account with WordPress installation on the Cloud deployment	Pass, with note
10	Access the WordPress admin page	Pass

Table 10 – Summary of the completed tests.

5.1 Test ID #1

Objective	Register a new account on the VM
Expected result	1. greathosting.com/register opens the registration page 2. The account is successfully registered according to the logfile. 3. *alberte.greathosting.com* is accessible 4. *greathosting.com/~alberte* is accessible 5. Confirmation email is visible in the command line *mail*.
Reproduction steps	1. Navigate to greathosting.com/register 2. Register with the following details: First name: Albert Last name: Einstein Email: maximilian@greathosting.com Username: alberte Password: alberte1 3. Wait a minute and then display the */var/log/greathosting.log* file with the *cat* command 4. Visit both *alberte.greathosting.com* and *greathosting.com/~alberte* after editing the hosts file 5. Sign-in to the VM with the "maximilian" username and check the email account with the *mail* command for the confirmation email.
Date	29/12/2016
Result	Pass

Table 11 – Details of test scenario #1.

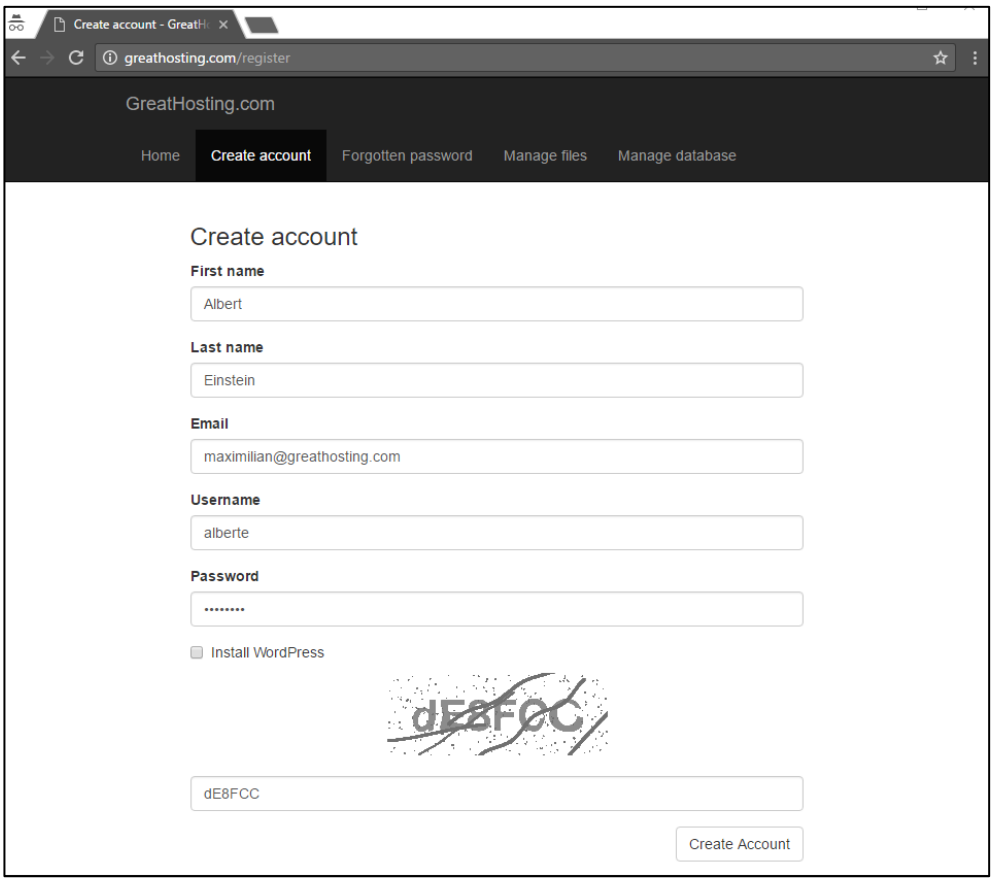


Figure 20 – Step 1 is completed and the form is filled with the registration data.

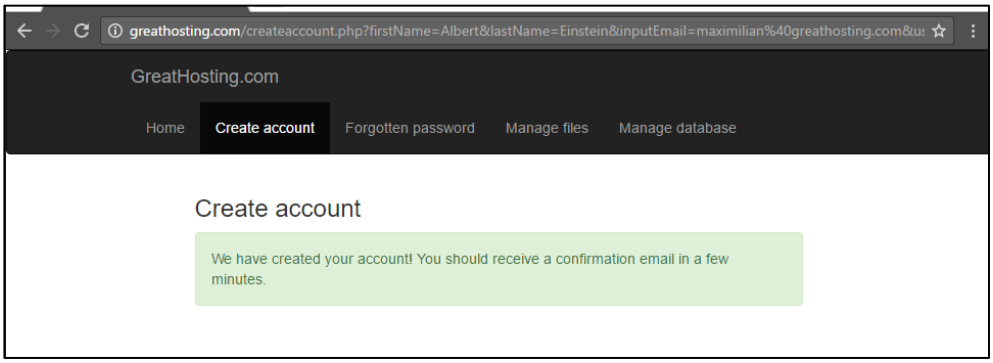


Figure 21 - Registration is confirmed by the user-facing message.

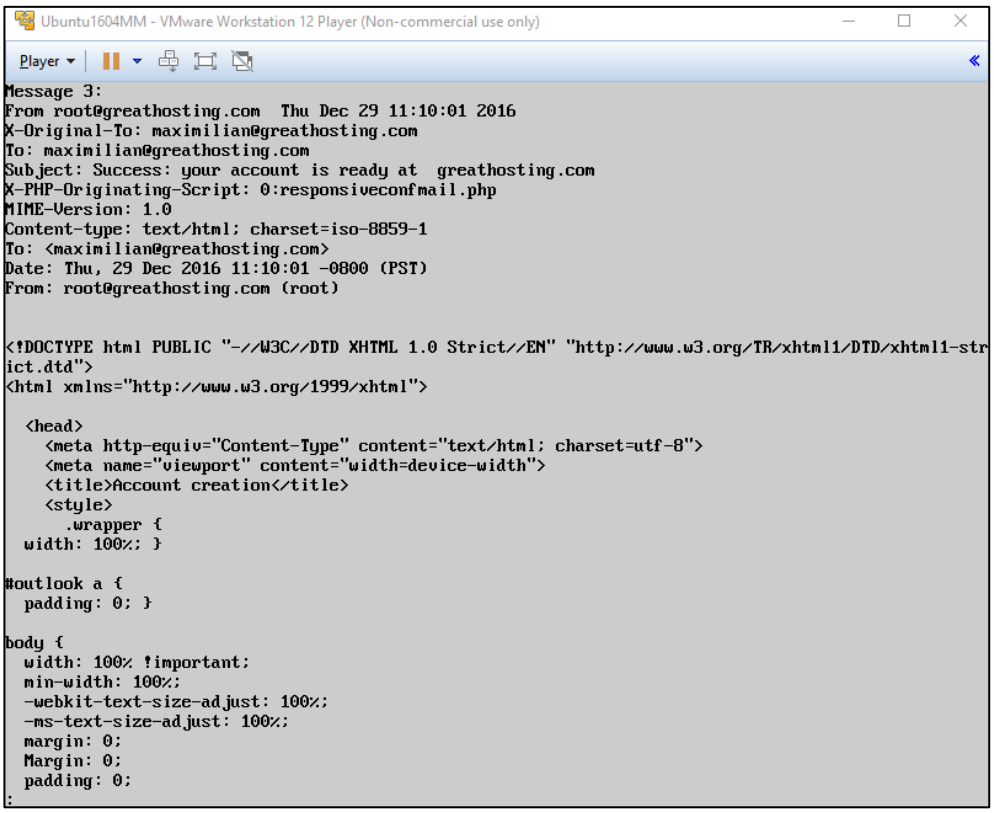
```
Thu, 29 Dec 2016 11:04:01 -0800: No password changes needed.
Thu, 29 Dec 2016 11:05:01 -0800: Scheduled account creator and password resetter script launched.
Thu, 29 Dec 2016 11:05:01 -0800: Required files found.
Thu, 29 Dec 2016 11:05:01 -0800: No account needs to be created.
Thu, 29 Dec 2016 11:05:01 -0800: No password changes needed.
Thu, 29 Dec 2016 11:06:01 -0800: Scheduled account creator and password resetter script launched.
Thu, 29 Dec 2016 11:06:01 -0800: Required files found.
Thu, 29 Dec 2016 11:06:01 -0800: No account needs to be created.
Thu, 29 Dec 2016 11:06:01 -0800: No password changes needed.
Thu, 29 Dec 2016 11:07:01 -0800: Scheduled account creator and password resetter script launched.
Thu, 29 Dec 2016 11:07:01 -0800: Required files found.
Thu, 29 Dec 2016 11:07:01 -0800: No account needs to be created.
Thu, 29 Dec 2016 11:07:01 -0800: No password changes needed.
Thu, 29 Dec 2016 11:08:01 -0800: Scheduled account creator and password resetter script launched.
Thu, 29 Dec 2016 11:08:01 -0800: Required files found.
Thu, 29 Dec 2016 11:08:01 -0800: No account needs to be created.
Thu, 29 Dec 2016 11:08:01 -0800: No password changes needed.
Thu, 29 Dec 2016 11:09:01 -0800: Scheduled account creator and password resetter script launched.
Thu, 29 Dec 2016 11:09:01 -0800: Required files found.
Thu, 29 Dec 2016 11:09:01 -0800: No account needs to be created.
Thu, 29 Dec 2016 11:09:01 -0800: No password changes needed.
Thu, 29 Dec 2016 11:10:01 -0800: Scheduled account creator and password resetter script launched.
Thu, 29 Dec 2016 11:10:01 -0800: Required files found.
Thu, 29 Dec 2016 11:10:01 -0800: Sending responsive html email to: maximilian@greathosting.com
Thu, 29 Dec 2016 11:10:01 -0800: User created: alberte
Thu, 29 Dec 2016 11:10:01 -0800: Encrypted password removed from the DB for user: alberte

Thu, 29 Dec 2016 11:10:01 -0800: Database and access rights created for user: alberte
Thu, 29 Dec 2016 11:10:01 -0800: Virtual host (subdomain) created for user: alberte

Thu, 29 Dec 2016 11:10:01 -0800: No password changes needed.
Thu, 29 Dec 2016 11:10:01 -0800: Apache config reloaded
Thu, 29 Dec 2016 11:11:01 -0800: Scheduled account creator and password resetter script launched.
Thu, 29 Dec 2016 11:11:01 -0800: Required files found.
Thu, 29 Dec 2016 11:11:01 -0800: No account needs to be created.
Thu, 29 Dec 2016 11:11:01 -0800: No password changes needed.
maximilian@greathosting:~$ _
```

Figure 22 – The output of the “cat /var/log/greathosting.log”. The logfile confirms the creation of the new user with username “alberte”.

Figure 23 – Step 4 is completed, both URLs are working.



```
Ubuntu1604MM - VMware Workstation 12 Player (Non-commercial use only)
Player
Message 3:
From root@greathosting.com  Thu Dec 29 11:10:01 2016
X-Original-To: maximilian@greathosting.com
To: maximilian@greathosting.com
Subject: Success: your account is ready at  greathosting.com
X-PHP-Originating-Script: 0:responsiveconfmail.php
MIME-Version: 1.0
Content-type: text/html; charset=iso-8859-1
To: <maximilian@greathosting.com>
Date: Thu, 29 Dec 2016 11:10:01 -0800 (PST)
From: root@greathosting.com (root)


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-str
ict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Account creation</title>
    <style>
      .wrapper {
  width: 100%; }

#outlook a {
  padding: 0; }

body {
  width: 100% !important;
  min-width: 100%;
  -webkit-text-size-adjust: 100%;
  -ms-text-size-adjust: 100%;
  margin: 0;
  Margin: 0;
  padding: 0;
:
```

Figure 24 – The HTML based confirmation email arrived to "maximilian@greathosting.com". However, it is not convenient to read it in a text-based email client such as *mail*.

5.2 Test ID #2

Objective	Access an email account on the VM using a POP3 email client.
Expected result	1. Thunderbird can retrieve the emails of "maximilian@greathosting.com" . 2. The recent confirmation email for the user "alberte" created in "Test ID #1" should be present and correctly displayed.
Reproduction steps	1. Launch Thunderbird and launch the process of adding a new account in manual mode. 2. Use the following connection information: Email address: maximilian@greathosting.com Password: xaq88 Incoming protocol: POP3 Incoming hostname: greathosting.com Incoming port: 110 Incoming SSL: None Incoming Authentication: Normal Password Username: maximilian Note that the outgoing connection information is not necessary to be set up correctly in this scenario. 3. Retrieve the emails and find the registration confirmation for "alberte" user from "Test ID #1".
Date	29/12/2016
Result	Pass

Table 12 – Details of test scenario #2.

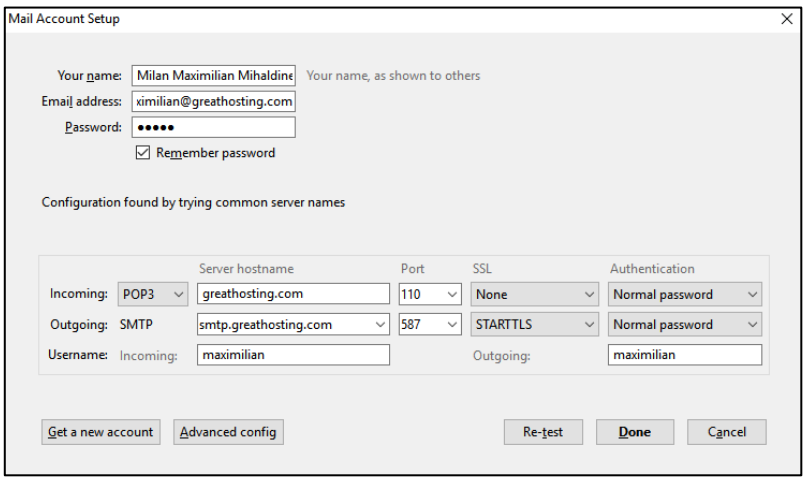


Figure 25 – Step 1 and 2: Adding a new account to Thunderbird to access the emails of "maximilian@greathosting.com".

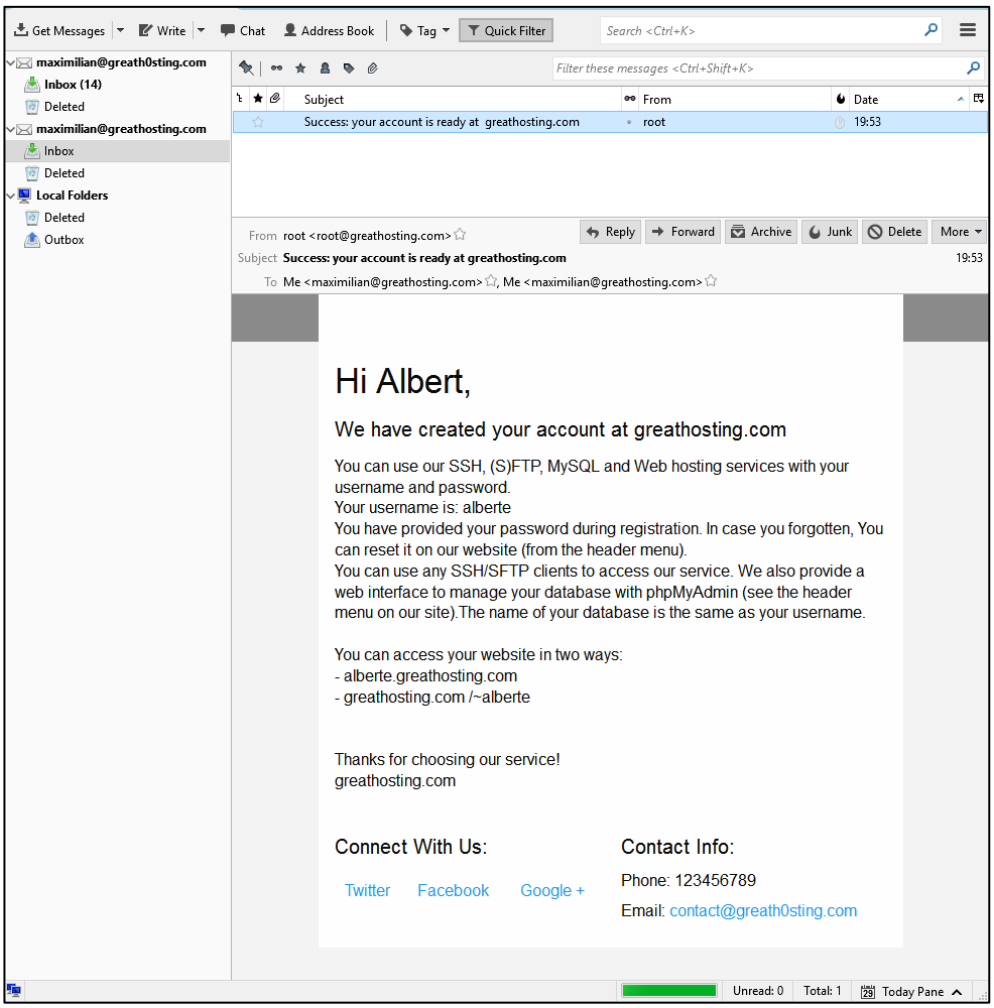


Figure 26 – The confirmation email is retrieved and correctly displayed for the user created in "Test ID #1" scenario.

5.3 Test ID #3

Objective	Reset the password of an account on the VM
Expected result	1. The account owner receives the password reset email. 2. The new password gets accepted.
Reproduction steps	1. Navigate to greathosting.com/forgottenpassword.php 2. Request the password reset email for the "maximilian@greathosting.com" address, which is used for registering user "alberte" in Test ID #1. 3. Retrieve the password reset email and use the reset link within it. 4. Set the new password to be "alberte88" 5. Login into the VM as "alberte" using the new password. Then sign in to MySQL using the new password and the "mysql -p" command.
Date	29/12/2016
Result	Pass

Table 13 – Details of test scenario #3

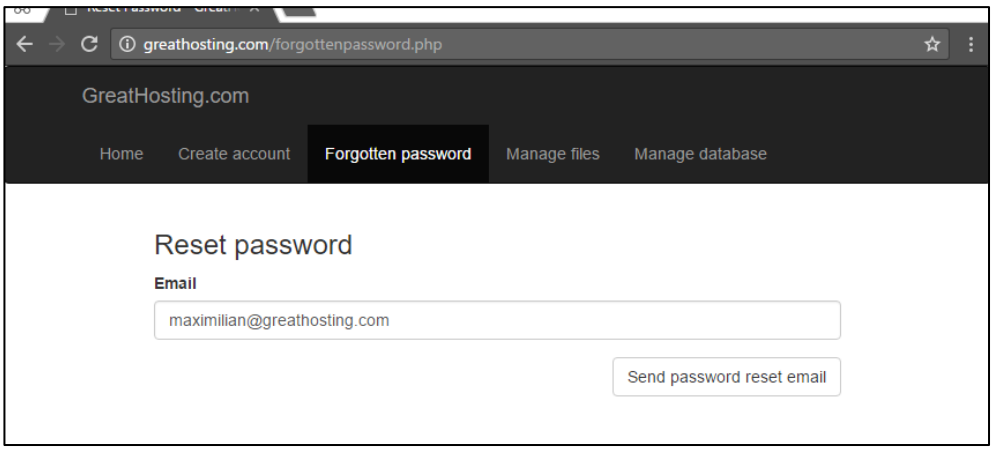


Figure 27 – Step 1: About to request the password reset.

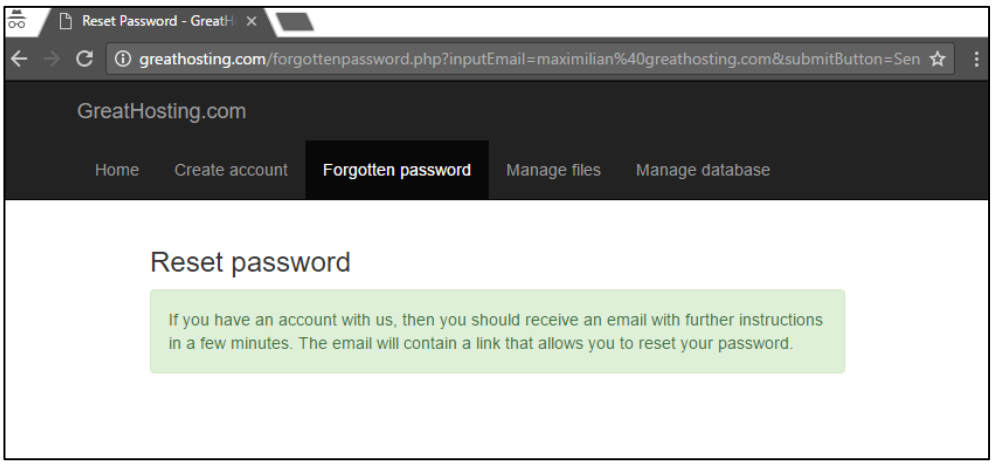


Figure 28 – Password reset request accepted.

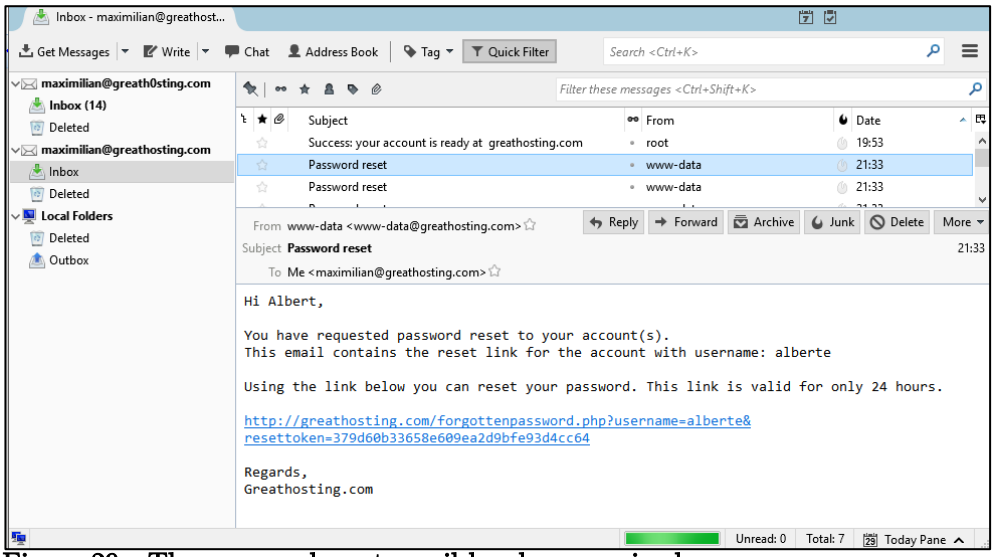


Figure 29 - The password reset email has been received.

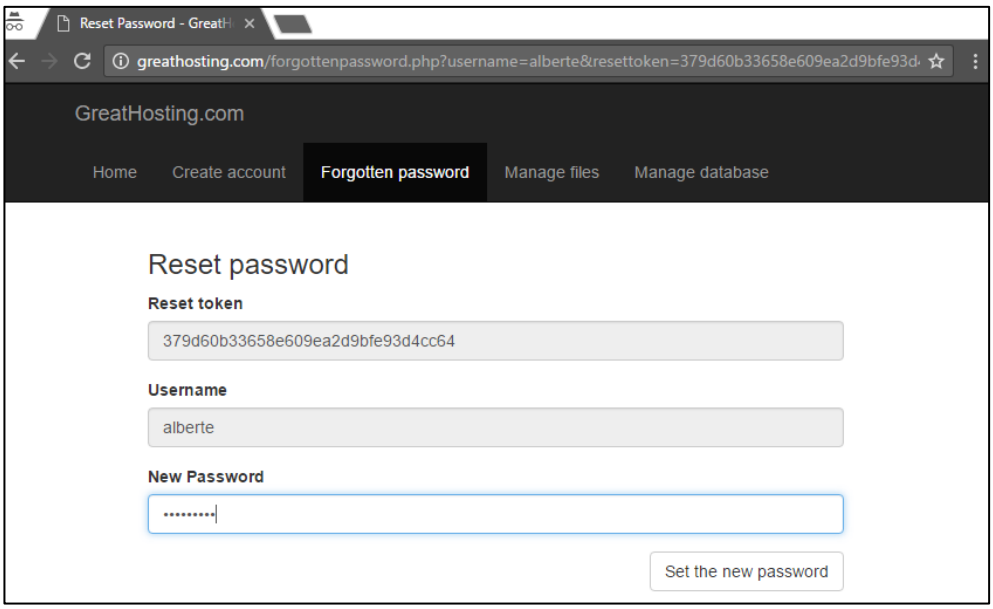


Figure 30 – The new password "alberte88" is about to be submitted.

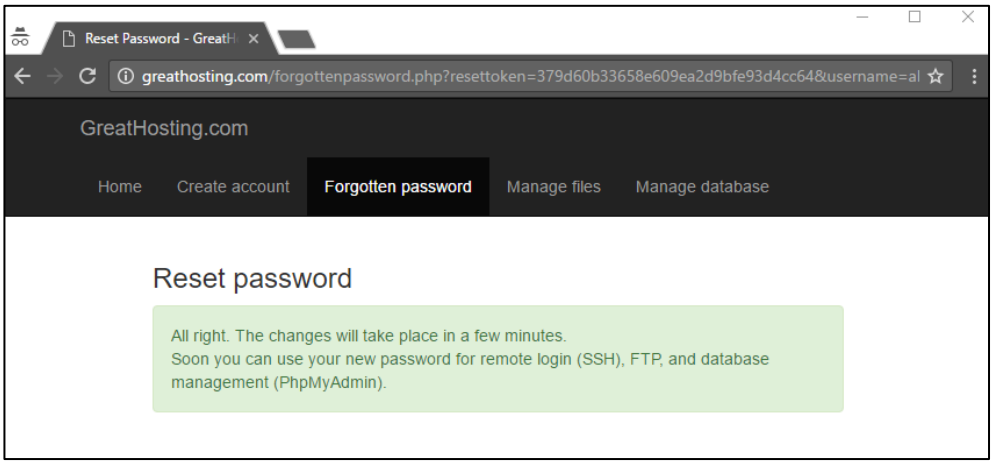


Figure 31 – The system accepts the new password.

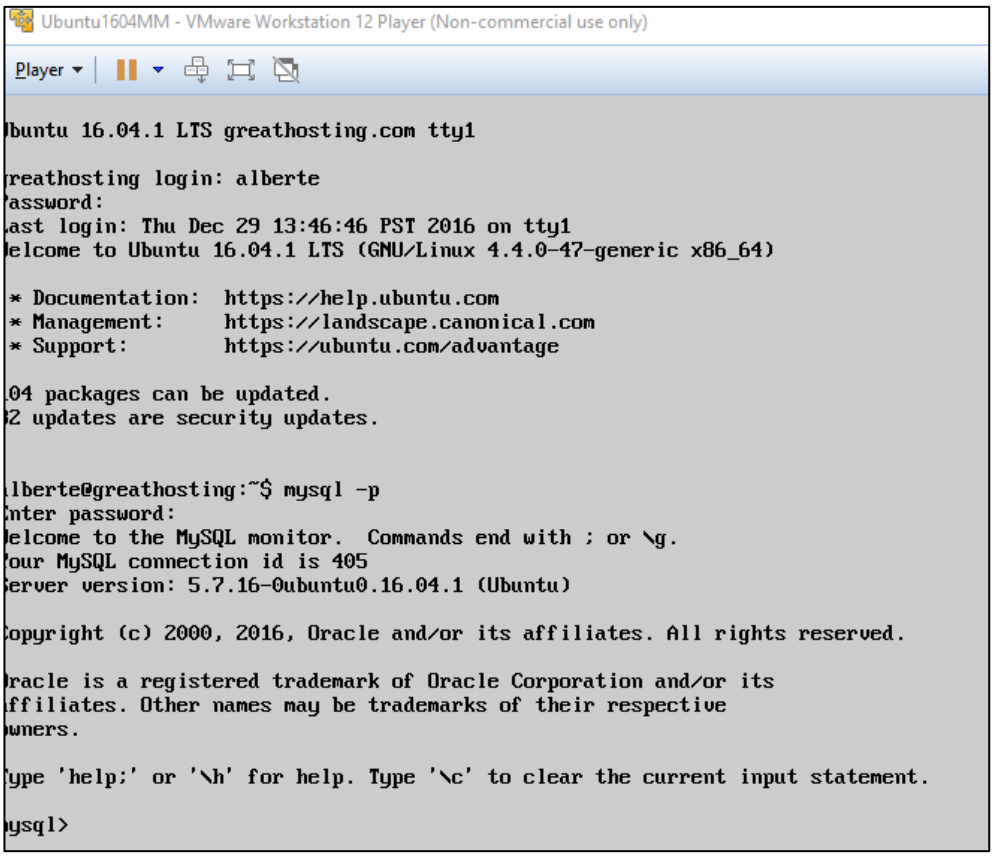


Figure 32 – Successful login with the new password into the system and MySQL.

5.4 Test ID #4

Objective	Access the system via SSH using with registered account on the VM
Expected result	1. The login over SSH should be successful. 2. After login, the visitor should be in his home folder of /home/alberte and the folder should contain a "public_html" folder.
Reproduction steps	1. Use an SSH client to connect to greathosting.com 2. Use the following connection details Port: 22 Username: alberte Password: alberte88 3. After login, list the current directory with "pwd" and the available content of the current folder using "ls".
Date	29/12/2016
Result	Pass

Table 14 - Details of test scenario #4

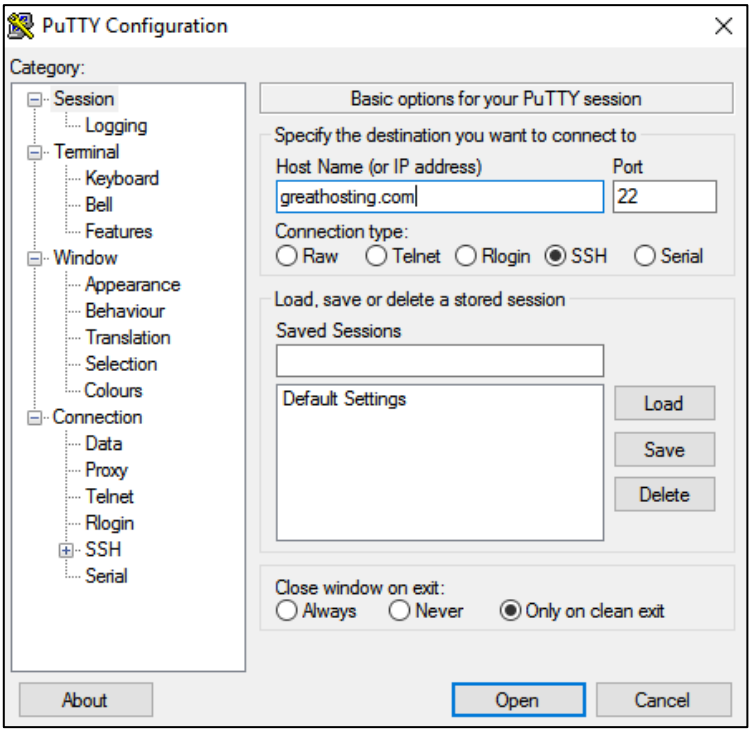


Figure 33 – About to connect to greathosting.com with an SSH client.

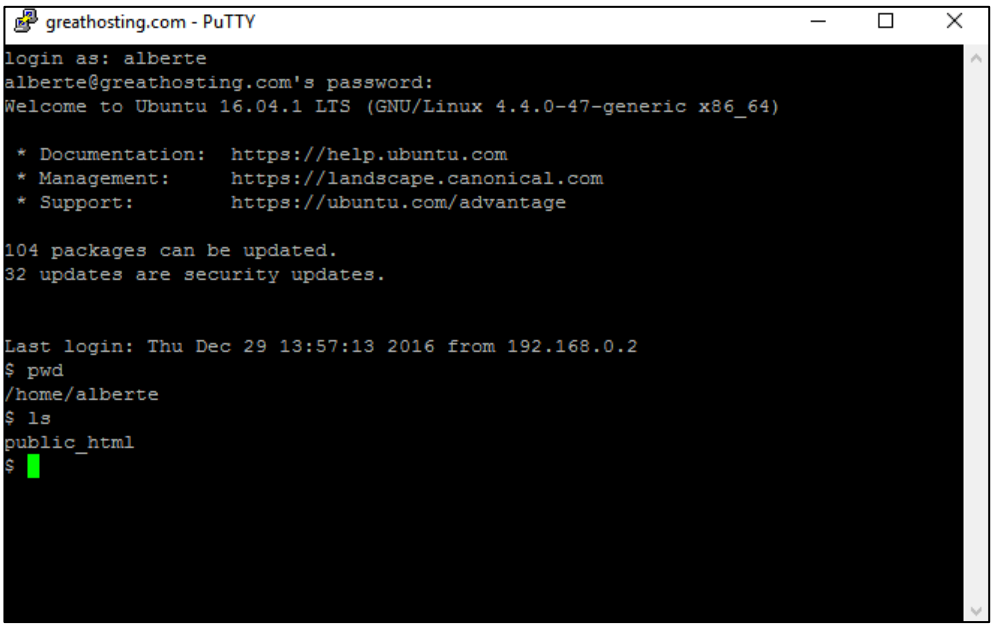


Figure 34 – Successful login. The user is in his home folder, which contains the "public_html" folder as expected.

5.5 Test ID #5

Objective	Access the files on the VM using a desktop-based SFTP client.
Expected result	1. The login over SFTP should be successful. 2. Downloading the "index.html" within the user's "public_html" directory should be successful.
Reproduction steps	1. Use an SFTP client to connect to greathosting.com 2. Use the following connection details Port: 22 Username: alberte Password: alberte88 3. Navigate to the "public_html" folder and download the "index.html" file.
Date	29/12/2016
Result	Pass

Table 15 - Details of test scenario #5

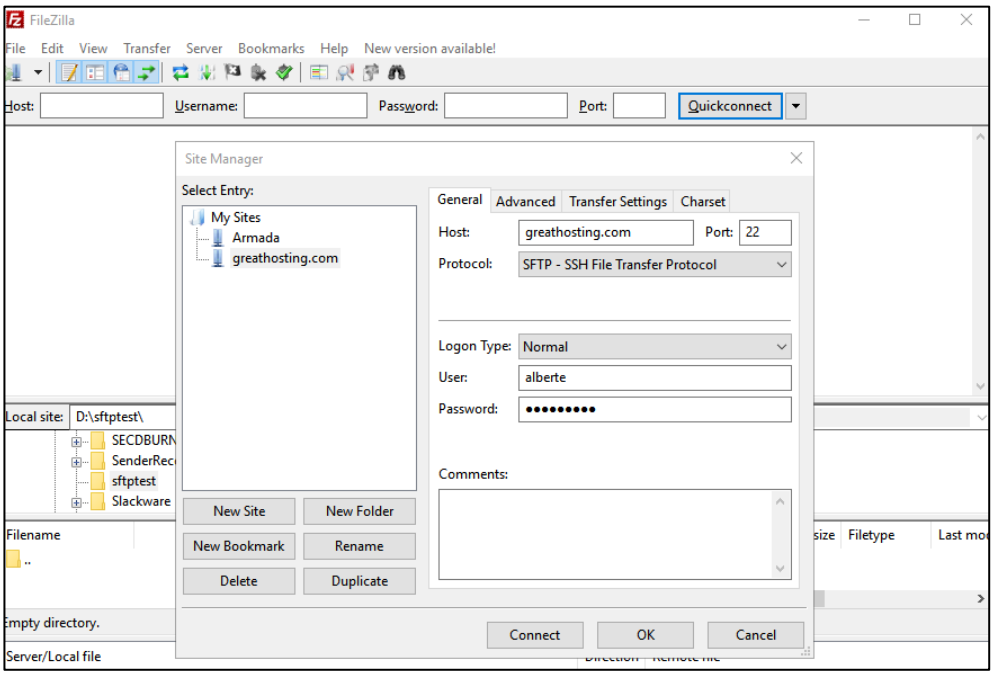


Figure 35 – Setting up the SFTP connection within the FileZille SFTP client.

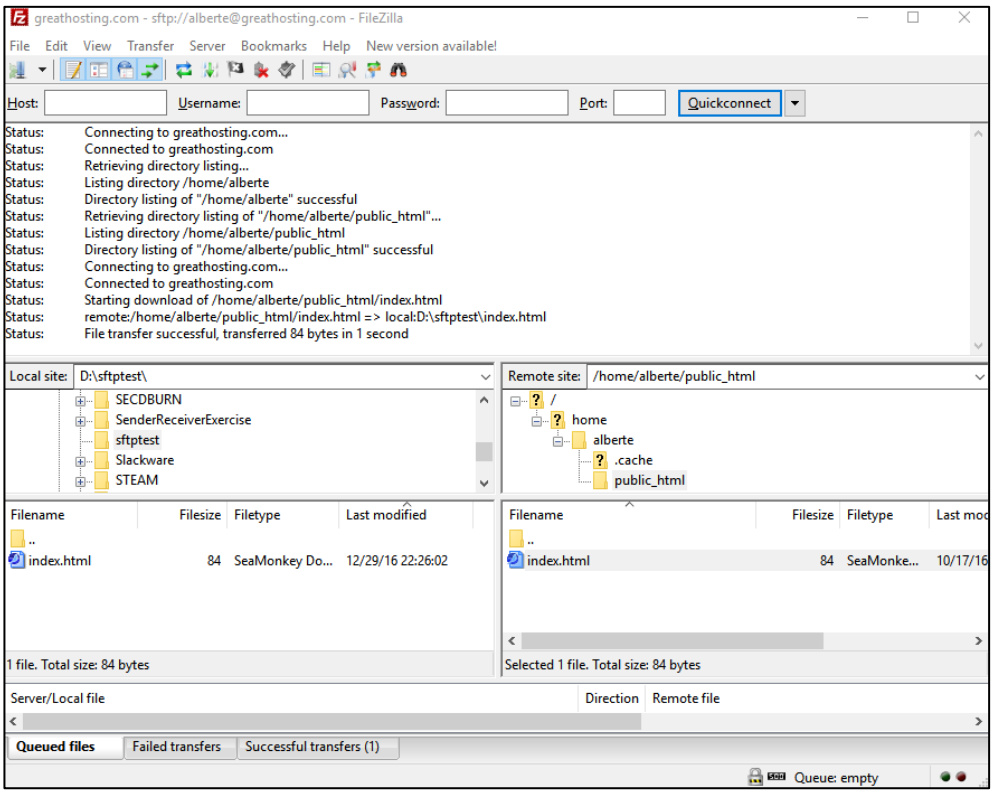


Figure 36 – The SFTP login is successful and the "index.html" file has been downloaded.

5.6 Test ID #6

Objective	Access the files on the VM using a web-based SFTP client.
Expected result	1. The login over SFTP should be successful. 2. Creating a new "test.txt" file within the user's home directory should be successful.
Reproduction steps	1. Navigate to http://greathosting.com/mftp/ 2. Select the SFTP connection and use the following connection data: Host: localhost Port: 22 Username: alberte Password: alberte88 Initial directory: /home/alberte Authentication type: password 3. Navigate to the "public_html" folder and create a new "test.txt" file. Place the string "This is a test file." into the file.
Date	29/12/2016
Result	Pass

Table 16 - Details of test scenario #6

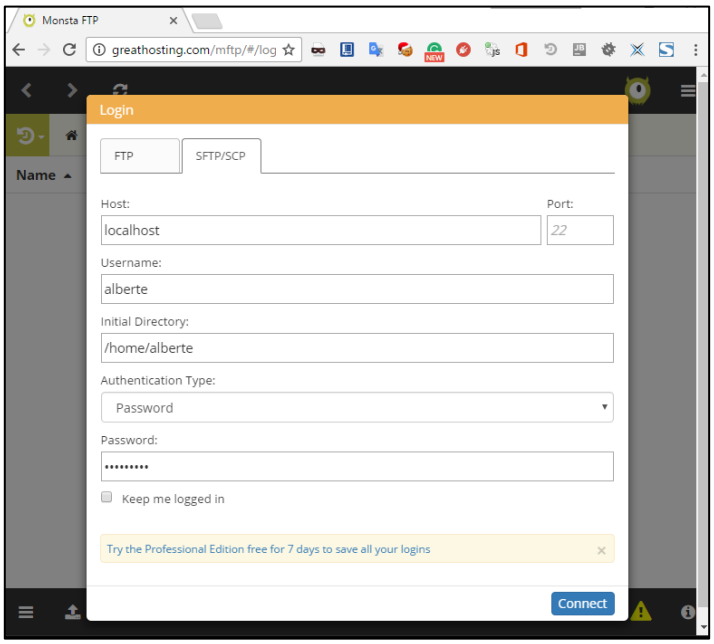


Figure 37 – Navigated to the web-based SFTP and filled out the connection details form.

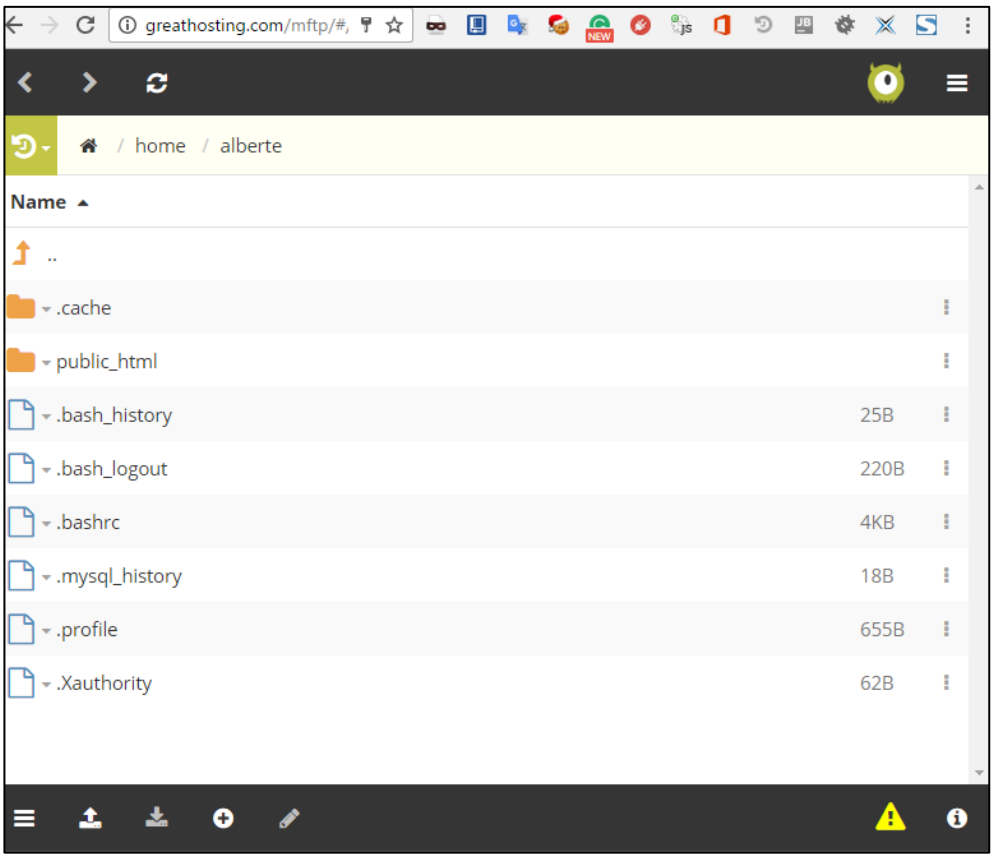


Figure 38 – The contents of the home directory are listed after the successful login.

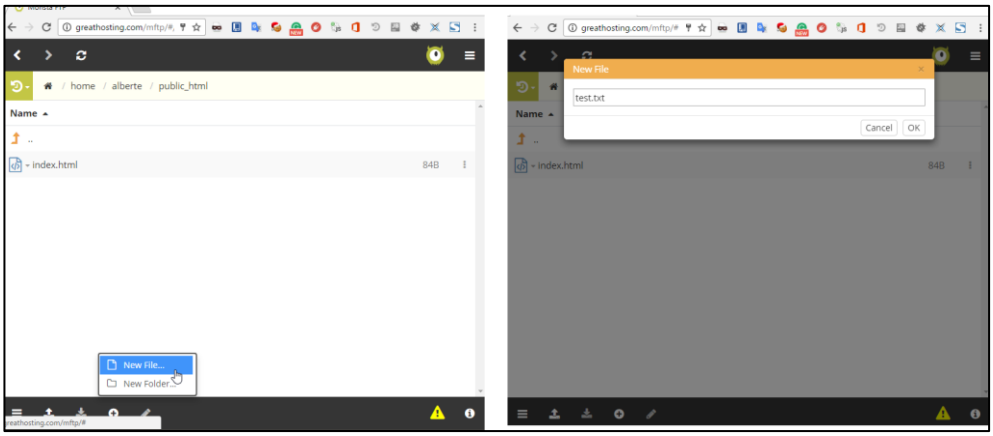


Figure 39 – Creating a new file called “test.txt” within the “public_html” folder.

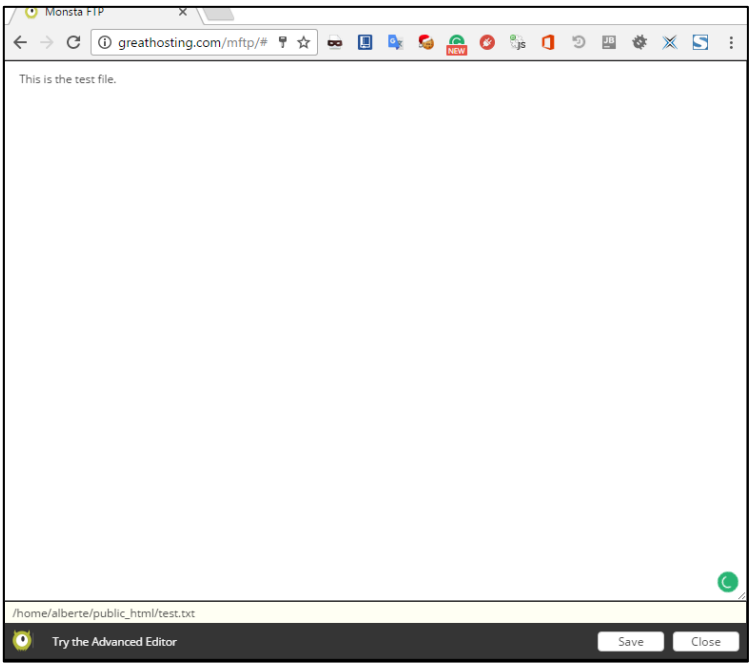


Figure 40 – Editing the new text file.

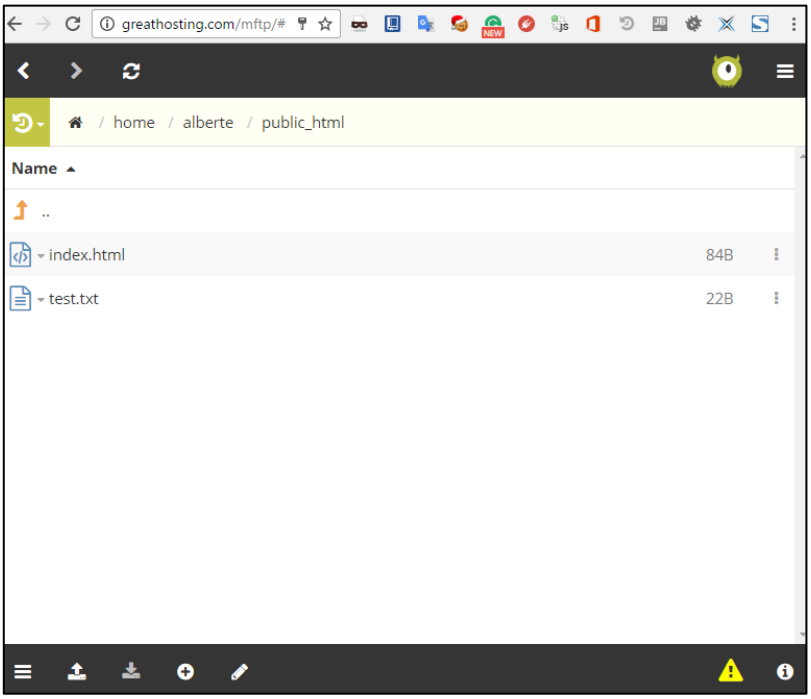


Figure 41 – The new file has been edited and saved.

5.7 Test ID #7

Objective	Access the database with PhpMyAdmin on the VM
Expected result	1. The login over PhPMyAdmin should be successful. 2. The "alberte" user should have a database with the same name.
Reproduction steps	1. Navigate to http://greathosting.com/phpmyadmin/ 2. Login with the following Username: alberte Password: alberte88 3. Check if the database "alberte" exists.
Date	29/12/2016
Result	Pass

Table 17 - Details of test scenario #7

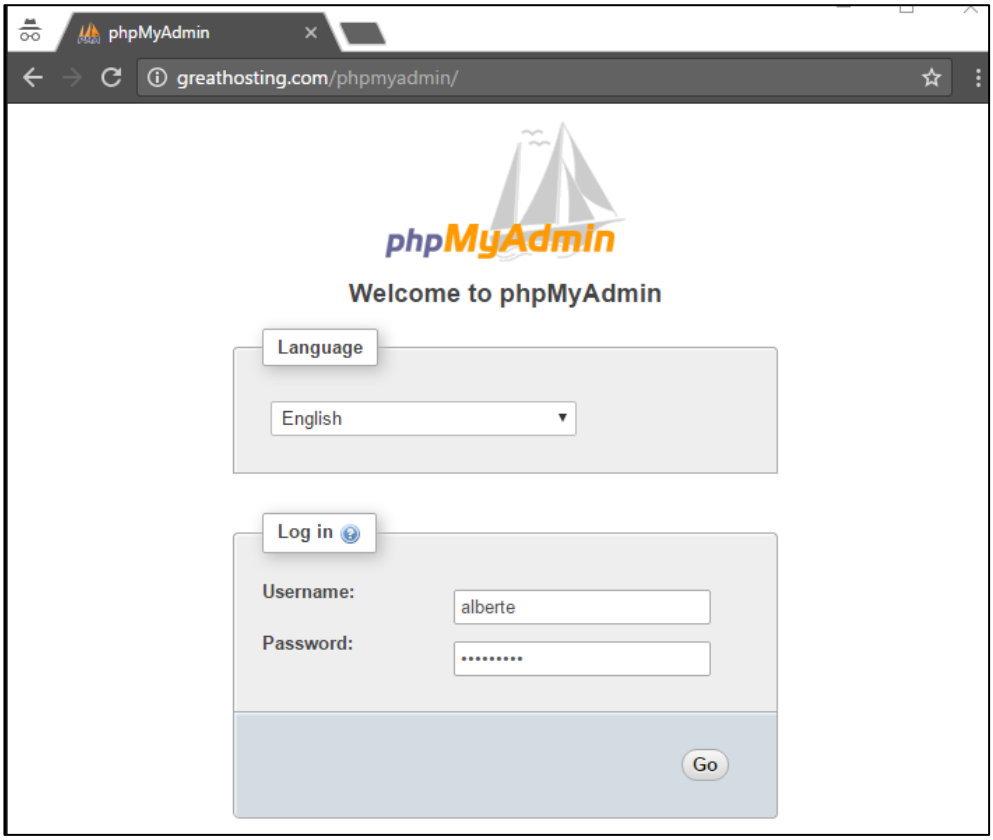


Figure 42 – Step 1: signing into PhpMyAdmin.

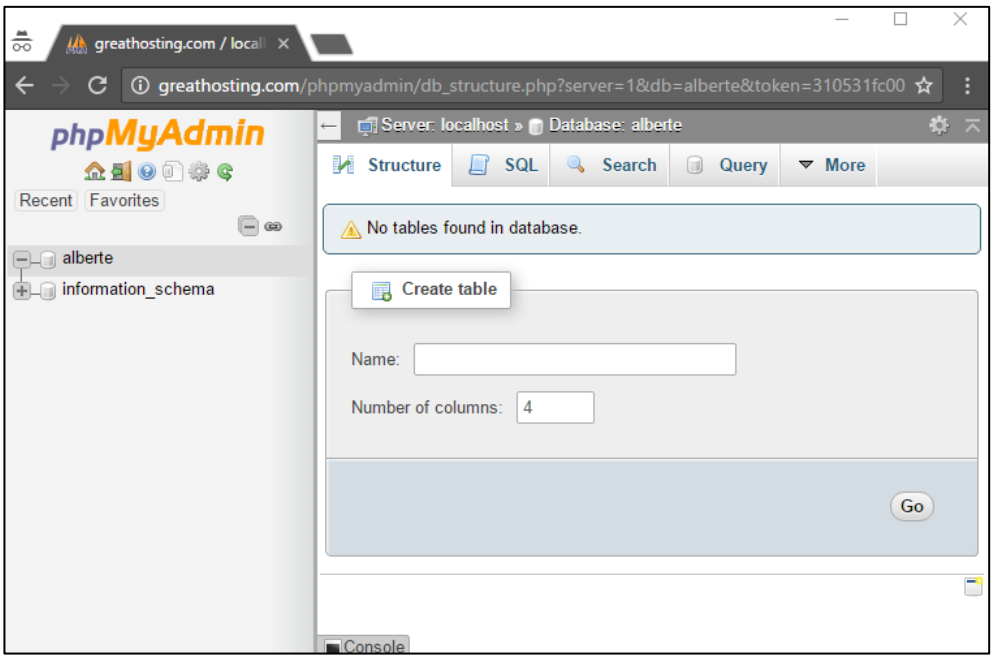


Figure 43 – The database "alberte" exists and it is empty.

5.8 Test ID #8

Objective	Confirm the client-side and server side registration form validation on the VM
Expected result	1. The server-side validation should prevent creating a new user with "maximilian" username, as it exists already. 2. The client-side validation should prevent using special characters. 3. The server-side validation should refuse the incorrect captcha. 4. The correct field values should be populated back in case of a server-side validation error. The password and captcha field should not be populated back. 5. All fields must be populated. 6. The error messages should be relevant and helpful.
Reproduction steps	1. Navigate to http://greathosting.com/register/ 2. Try to submit the unfilled field. 3. Fill out and submit the form as the following: First name: Charles Last name: Darwin Email: maximilian@greathosting.com Username: maximilian Password: darwin Captcha: *According to the image* 4. Use the data from the previous point, but change the username to "darwin" and make a mistake in the captcha before submission. 5. Use the data from the third point, but change the username to "darwin!!!".
Date	29/12/2016
Result	Pass

Table 18 - - Details of test scenario #8

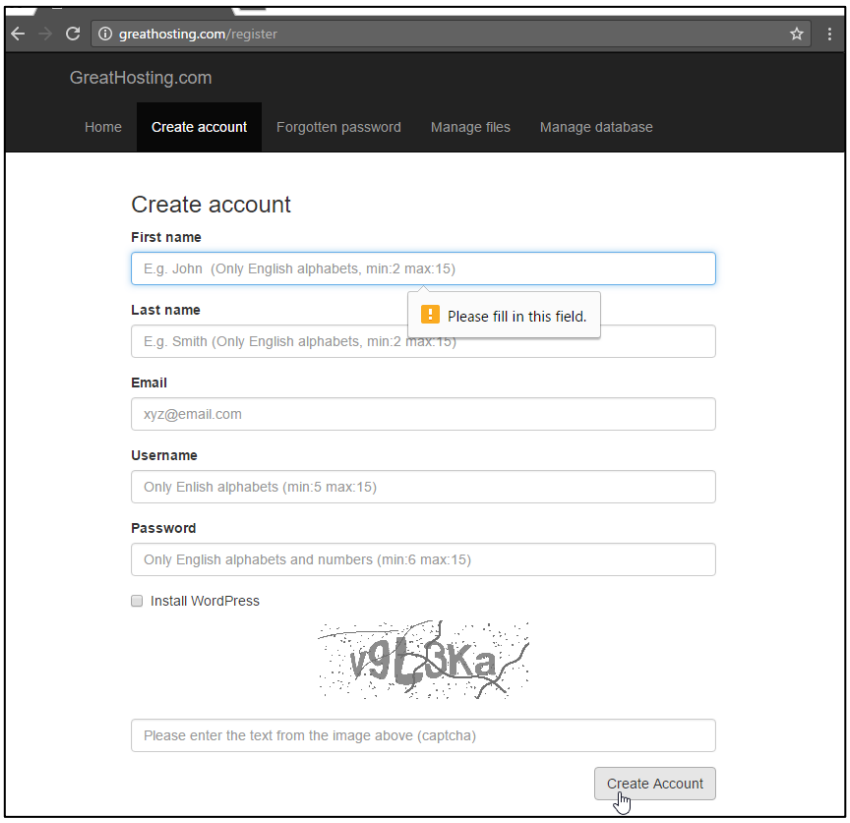


Figure 44 – Hitting the submit button without filling out the form. The browser asks to fill out the first unfilled field in the list.

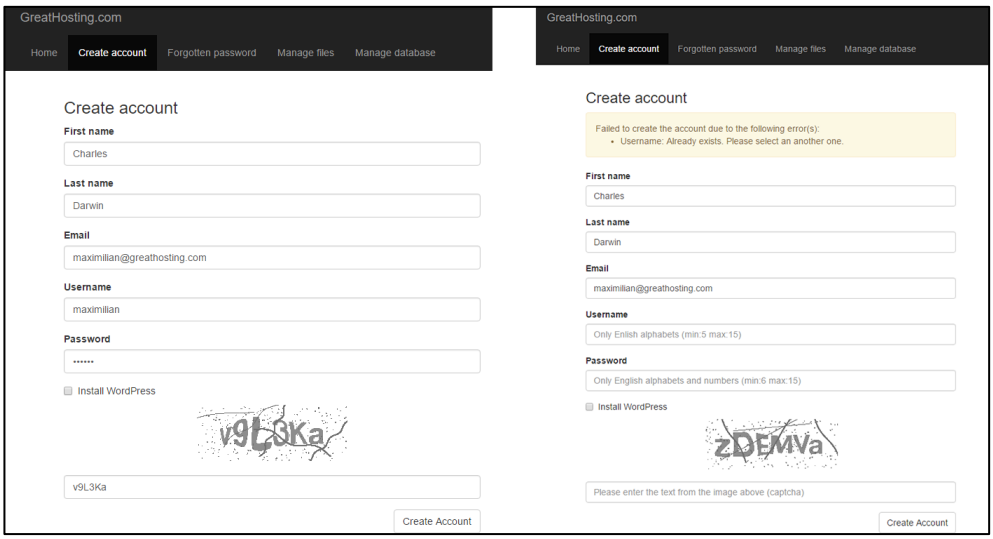


Figure 45 – Server-side validation refuses to create a new user with "maximilian" username as it exists already. Note that this user is not in the "customers" table. The validation check for existing system users as well as usernames in the customer table. Also, note that the username field was not repopulated as it was incorrect.

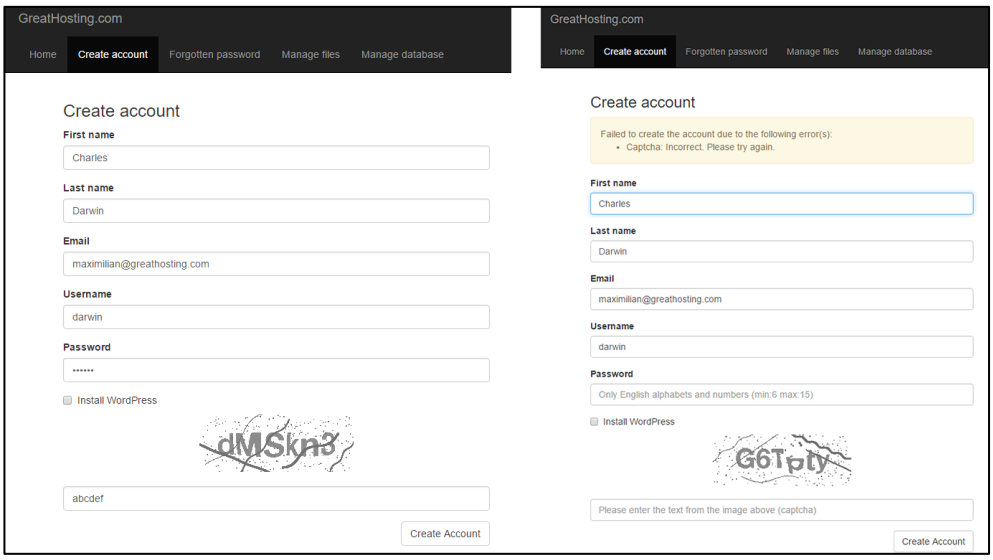


Figure 46 – The server-side validation displays an error regarding the incorrect captcha. Note that all the fields were repopulated, except the password and the captcha.

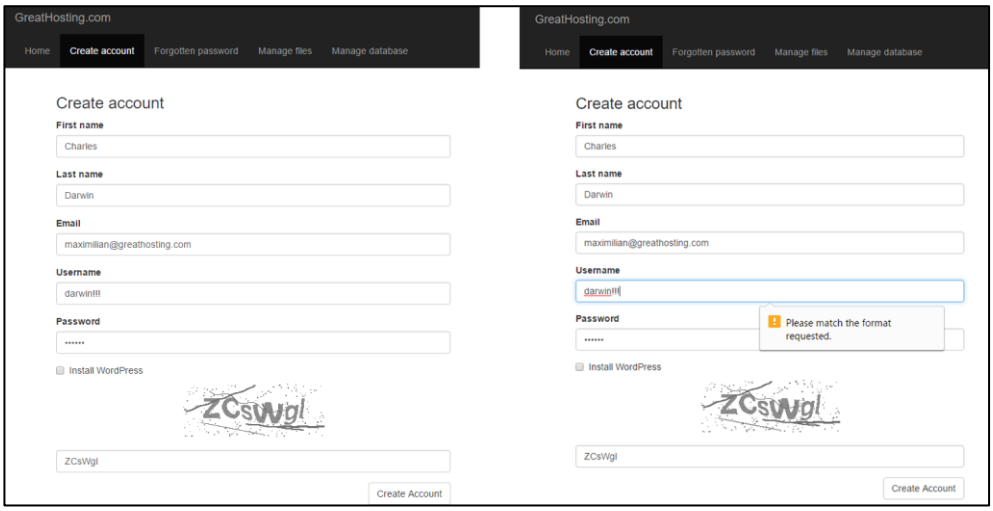


Figure 47 – The client-side validation prevented the use of special characters in the username ("darwin!!!").

5.9 Test ID #9

Objective	Register a new account with WordPress installation on the Cloud deployment
Expected result	1. greath0sting.com/register opens the registration page 2. *alberte.greath0sting.com* is accessible 3. *greath0sting.com/~alberte* is accessible 4. The WordPress installation should happen automatically. 5. Confirmation email should be received on an email address with a different domain, as long as "greath0sting.com" is whitelisted. 6. There should be two confirmation emails: one for the system and one for the WordPress installation. 7. The WordPress email should contain login information for the WordPress admin pages. 8. There is no need to edit the hosts file on the local computer, as the greath0sting.com is an actual registered domain.
Reproduction steps	1. Navigate to greath0sting.com/register 2. Register with the following details: First name: Albert Last name: Einstein Email: max@3mteam.hu Username: alberte Password: alberte1 3. Check for the two confirmation emails. 4. Visit both *alberte.greathosting.com* and *greath0sting.com/~alberte* . The installed WordPress page should be visible.
Date	29/12/2016
Result	Pass, with note: WordPress works with the subdomain address format and shows "page not found" error using the "/~alberte" format.

Table 19 - Details of test scenario #9.

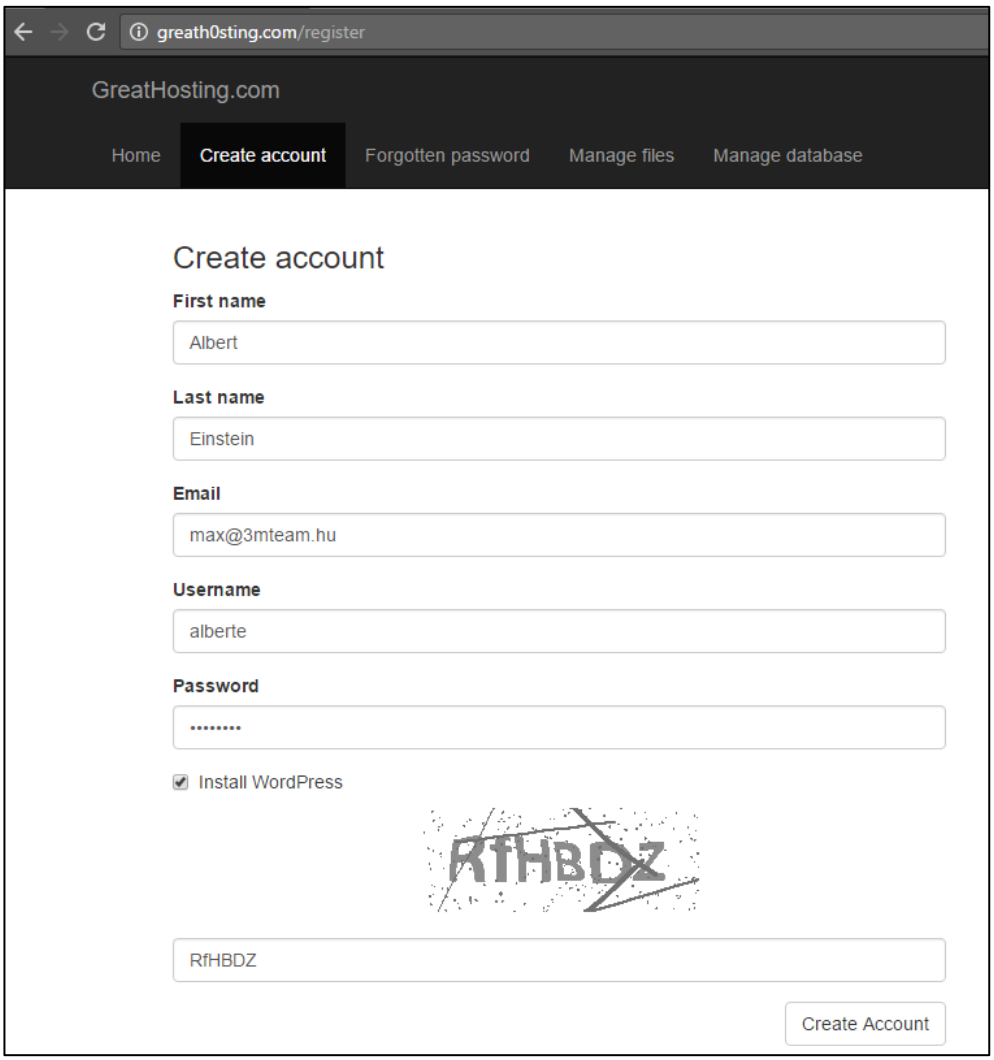


Figure 48 – The registration form loaded and then filled on the Cloud-deployed system.

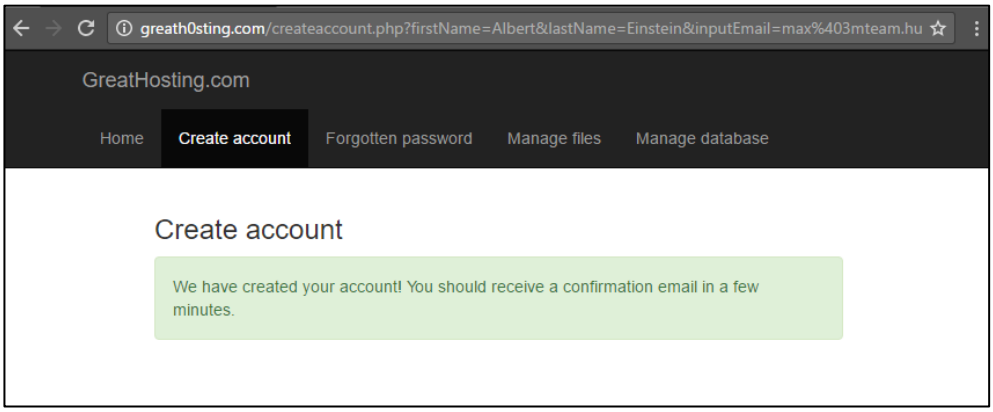


Figure 49 – System confirmed the account registration.

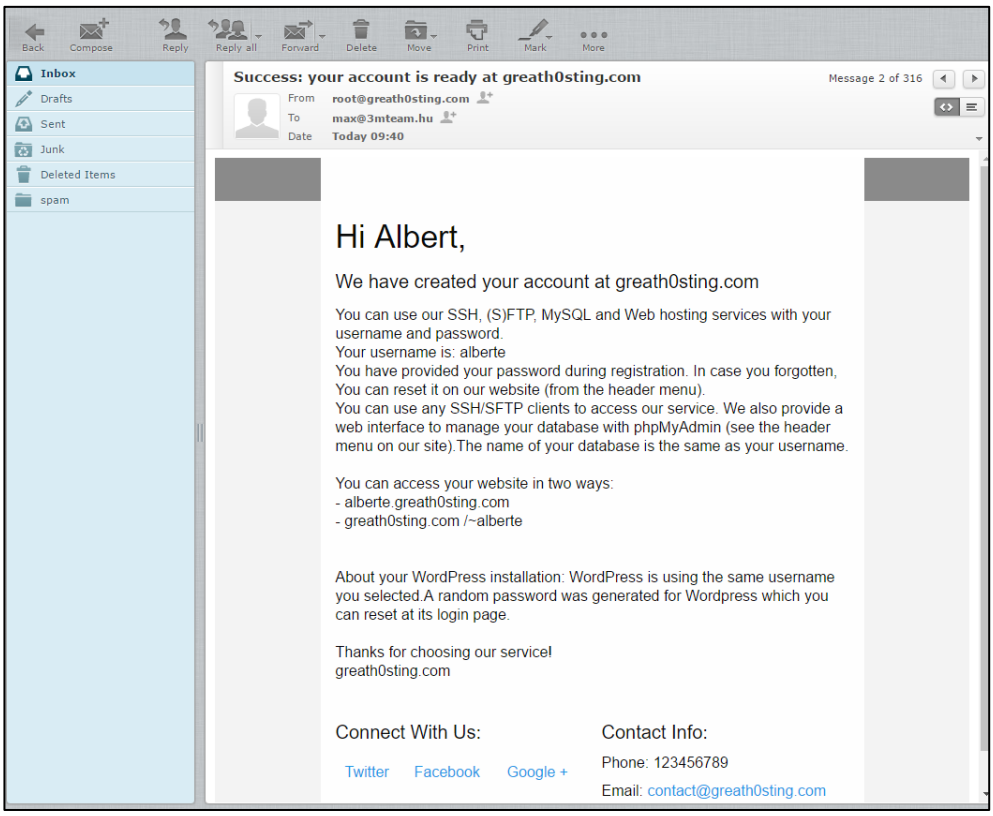


Figure 50 – The confirmation email has arrived to "max@3mteam.hu".

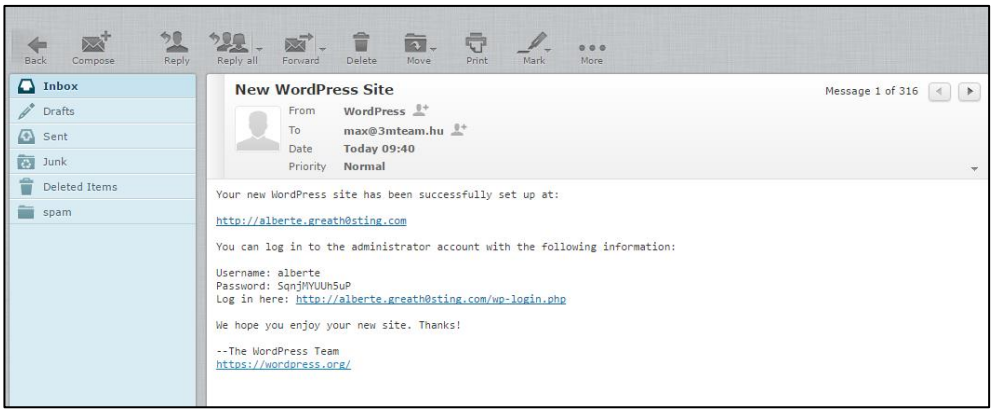


Figure 51 – The confirmation email from WordPress has arrived. It contains the username, password, and link to access the admin page of WordPress.

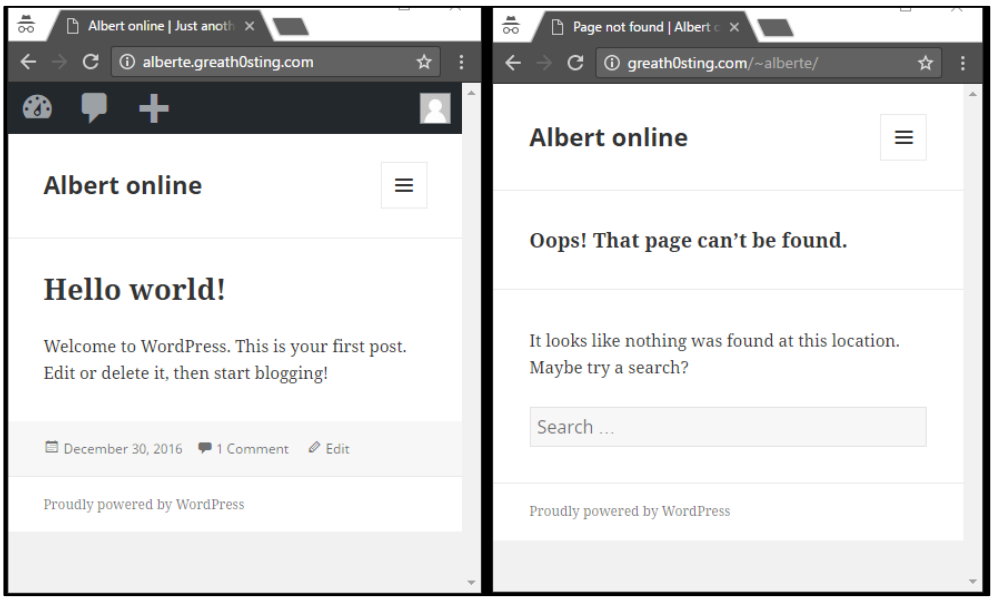


Figure 52 – Both "alberte.greath0sting.com" and "greath0sting.com/~alberte" displays the WordPress site of the "alberte" user. There was no need to edit the local computer's hosts file. However, WordPress shows a "page cannot be found" error when using the "greath0sting.com/~alberte" link.

5.10 Test ID #10

Objective	Access the WordPress admin page
Expected result	1. The WordPress admin page is accessible using the login details received in email.
Reproduction steps	1. Open the WordPress confirmation email received through "Test ID #9". 2. Use the login details in the email to sign-in to the WordPress admin.
Date	29/12/2016
Result	Pass

Table 20 - Details of test scenario #10.

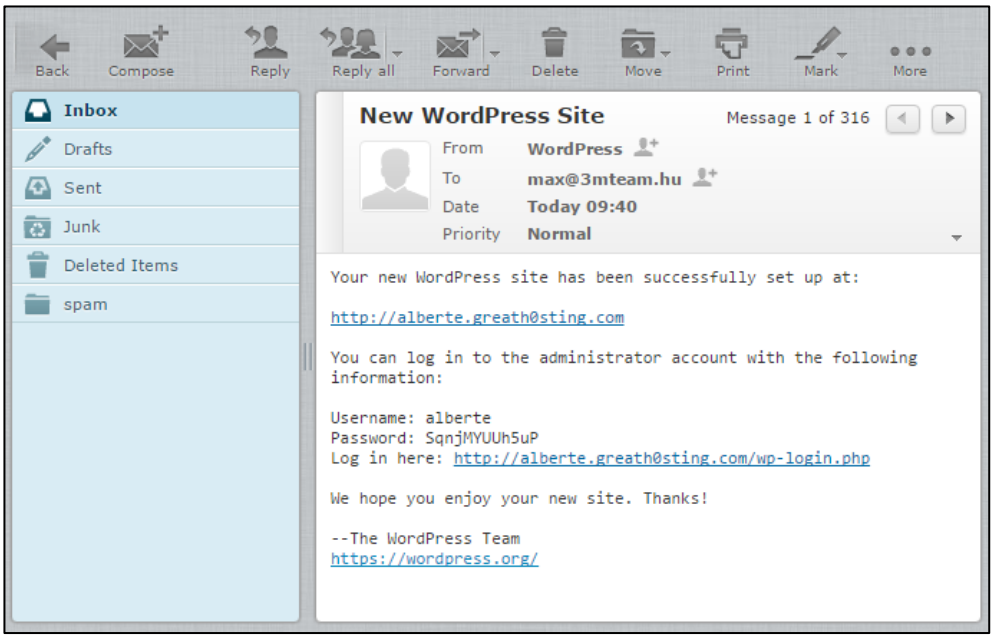


Figure 53 – The WordPress confirmation email from Test ID #9.

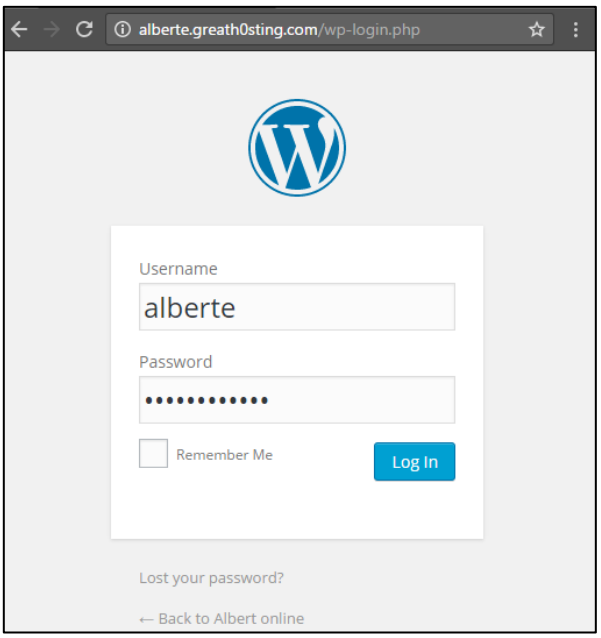


Figure 54 – The URL within the email takes to the login page. The username and password typed in from the email.

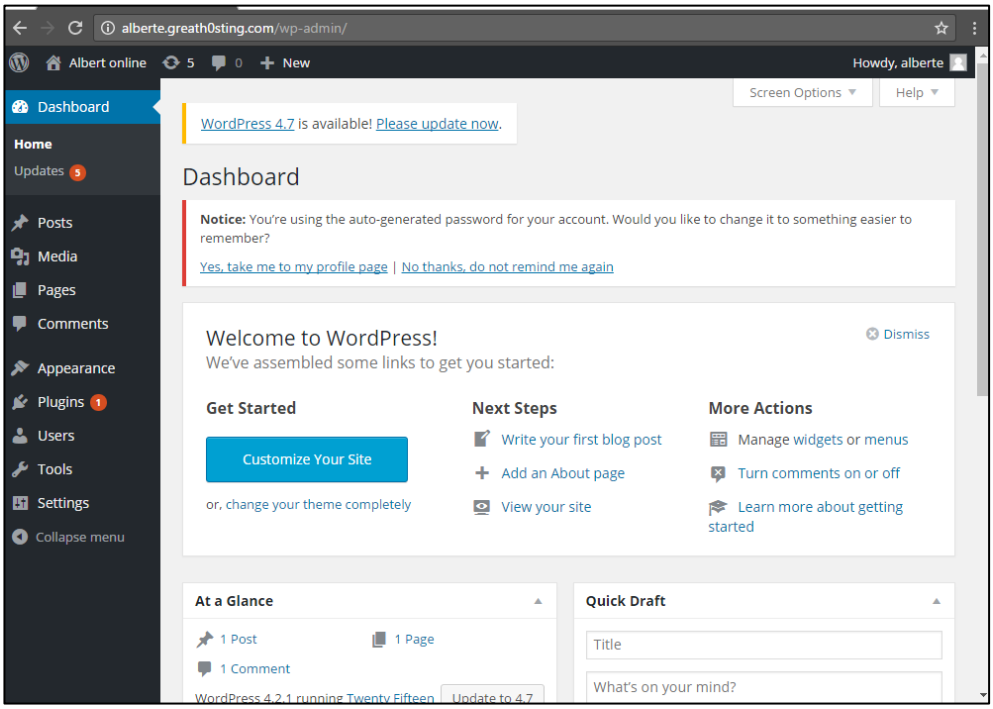


Figure 55 – The WordPress Admin page displayed after the successful login.

6 Conclusion

The project offered a design that addresses the required and desired features of the specification. Visitors are able to register to access and use the system, which offers webhosting, database, and terminal services. Visitors can access the system via common SSH and SFTP clients or through the provided web-based interfaces such as PhPMyAdmin and MonstaFTP. For their convenience, visitors can get WordPress installed automatically to their web space and reset their password through self-service if needed. Through adequate validation and robot filtering, the system ensures a reasonable quality of the stored users' data. The project has been deployed as both as a virtual machine and as a live system in the Cloud. Both deployments have gone through sufficient testing, and the results were documents. Suggestion for future improvements has been made regards to security and functionality. It can be concluded that the implementation completely fulfils the expectations of the specification, while also providing additional features to satisfy 'unthought' needs of the business owner.

7 Appendix

7.1 Usernames and passwords

This section provides the usernames and passwords required to access the virtual machine and cloud deployment of the system.

// Section removed for the GitHub publication

7.2 Attached media

The printed work has two flash drives with identical content attached to it. The drives contain the VMWare virtual machines and the electronic version of this document. The used version of VMware was "VMware Workstation 12 Player 12.5.0 build-4352439". The images have been tested on two separate machines.

The Virtual Machines contain the source code of the developed system. This can be found within the "/var/www/" folder.

7.3 Cloud deployment availability

The cloud deployment of the system is planned to stay available until 31st January 2017. After that point, the virtual machine will be turned off and decommissioned.

Please note that the availability of this system is not guaranteed as it makes possible to anyone to access and use the system's resources. In such case, it is possible that the service will be turned off earlier than planned.

8 References

Tacacho, F., 2016. *Migrate your VMware machine to Azure.* [Online]
Available at:
<http://microsoft.opennessatcee.com/azureboxes/2016/02/28/migrate-vmware-to-azure/>
[Accessed 28 December 2016].

9 Bibliography

Blum, R. and Bresnahan, C., 2011. *Linux command line and Shell scripting bible*. Indianapolis, Ind.: Wiley.

Nemeth, E., 2011. *UNIX and Linux system administration handbook*. Upper Saddle River, NJ: Prentice Hall.