

Appendix H: Developer Guide

a. Sync App

i. Installing Visual Studio

1. Install Visual Studio 2017. It can be downloaded from <https://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=fb43725a-5c9b-e011-969d-0030487d8897&vsro=8>. Sign in using your account and get the product key.
2. Install Visual Studio by following the instructions on the screen. Enter your product key when asked to do so.
3. Launch Visual Studio Installer and select modify. Check the checkbox .Net Desktop Development. Click the modify button and wait for the install to complete.
4. Launch Visual Studio 2017. Click on File -> Open -> Project/Solution. Browse to the folder you downloaded the source code and select Sync.sln.
5. Do note that you may be prompted to restart Visual Studio in Administrator Mode as the Sync App requires administrator access.

ii. Creating Gmail Accounts

1. You will need to create 2 Gmail accounts, one for the client, one for the server.
2. On both accounts, click on your username followed by My Account.

Privacy

My Account

Figure 106: Gmail My Account Button

3. Click on sign-in & security and scroll all the way to the bottom of the page.

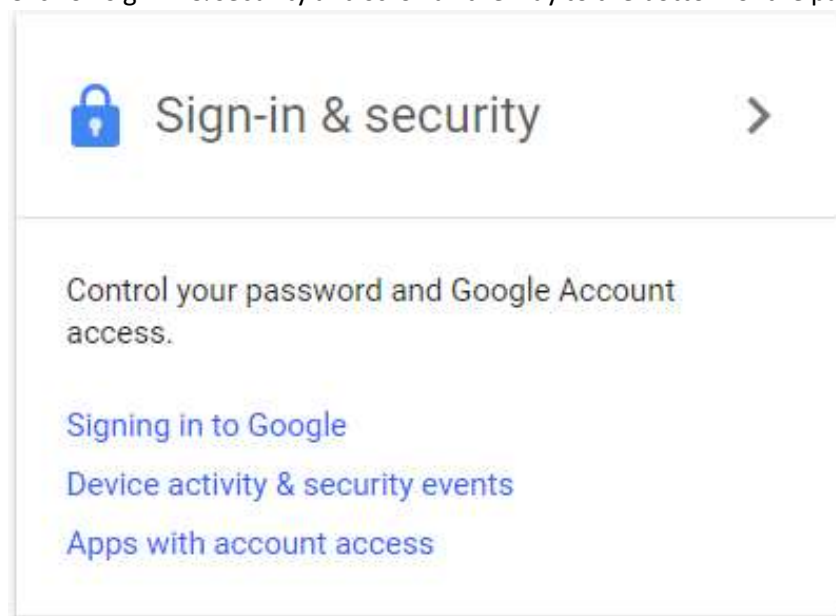


Figure 107: Gmail Sign-in and Security

4. Allow less secure apps to access.

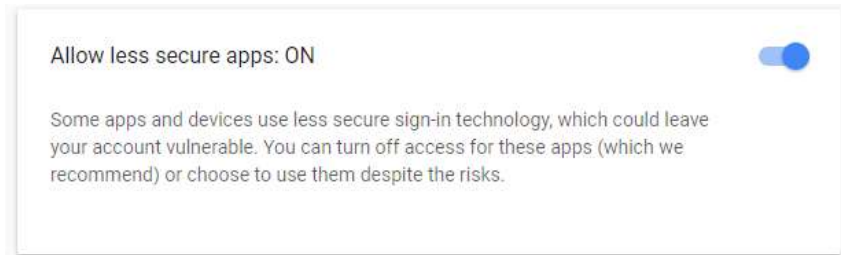


Figure 108: Gmail Allow Less Secure Apps

5. On the server email, click on create new label on the bottom of the left pane.

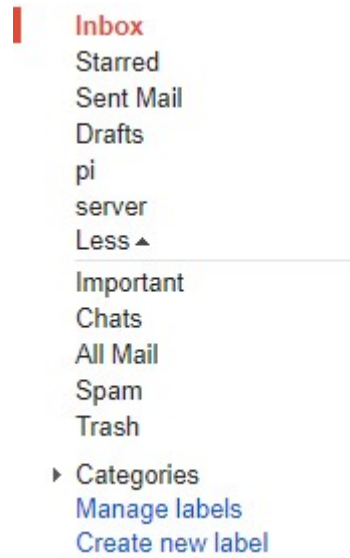


Figure 109: Gmail Create New Label

6. Create a label for Server and Pi so that the emails can be filed accordingly.
7. On the client's email click on the settings icon followed by settings.

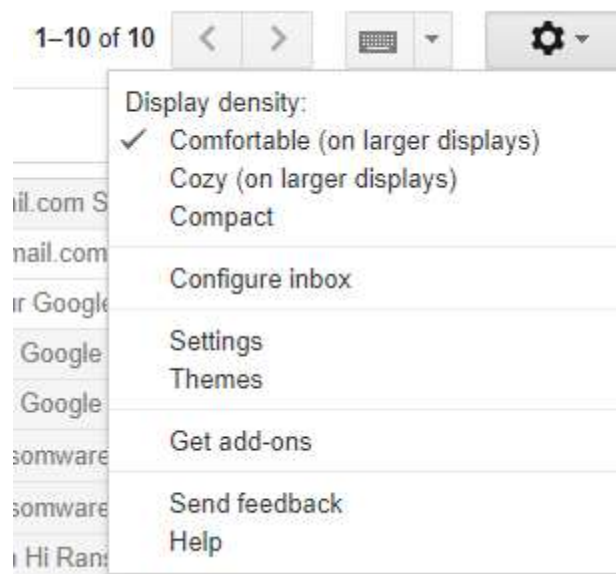
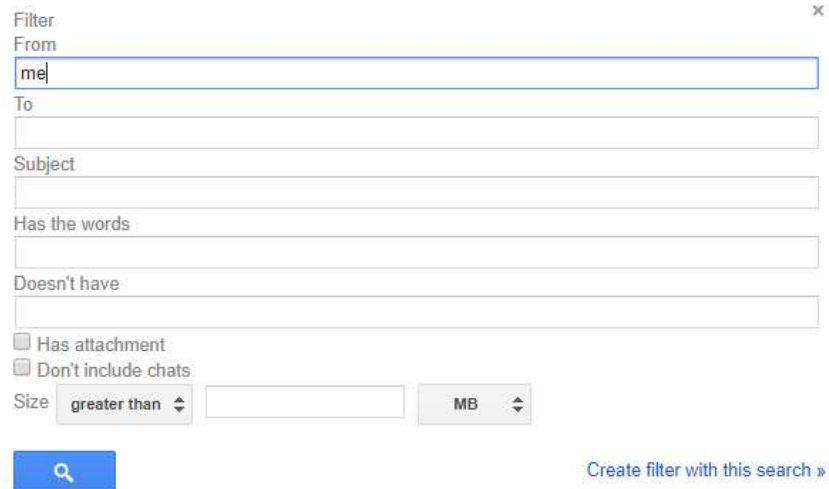


Figure 110: Gmail Settings

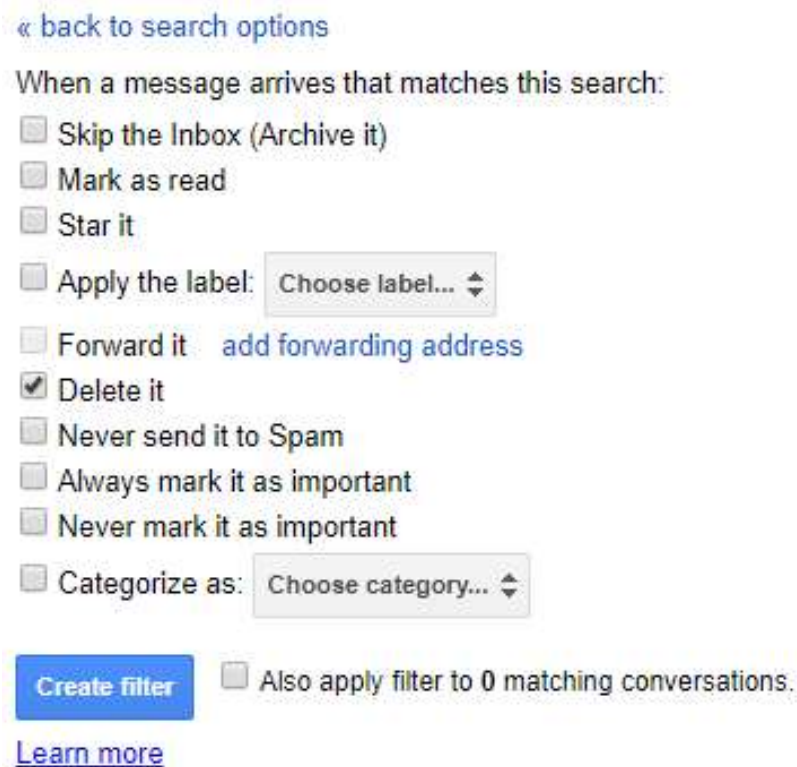
8. Click on Filters and Blocked Addresses. Click on New Filter and enter the from address as me.



The image shows the 'Filter' creation interface in Gmail. It has a title bar 'Filter' with a close button 'x'. Below the title bar are several input fields: 'From' (containing 'me'), 'To', 'Subject', 'Has the words', and 'Doesn't have'. Below these are two checkboxes: 'Has attachment' and 'Don't include chats'. At the bottom, there is a 'Size' section with a dropdown menu set to 'greater than', an empty input field, and a unit dropdown set to 'MB'. A blue search button with a magnifying glass icon is on the left, and a link 'Create filter with this search »' is on the right.

Figure 111: Gmail Create Filter Prompt

9. Click on filter with this search and then check the box delete it. Click on create filter.



The image shows the 'Create filter' dialog in Gmail. At the top is a link '« back to search options'. Below it is the text 'When a message arrives that matches this search:'. There is a list of checkboxes: 'Skip the Inbox (Archive it)', 'Mark as read', 'Star it', 'Apply the label:' (with a dropdown menu 'Choose label...'), 'Forward it' (with a link 'add forwarding address'), 'Delete it' (checked), 'Never send it to Spam', 'Always mark it as important', 'Never mark it as important', and 'Categorize as:' (with a dropdown menu 'Choose category...'). At the bottom left is a blue 'Create filter' button. To its right is a checkbox 'Also apply filter to 0 matching conversations.' and a link 'Learn more'.

Figure 112: Gmail Create Filter Button

10. This is done to delete all the emails sent by the Sync App.

iii. Sync App Overview

For the Sync App there are mainly 4 forms. The first form (WhichLab.cs) is the WhichLab form where users can select the labs. The second form (Form2.cs) is the preferences form. The third form (SelectDate.cs) is the SelectDate form where users can select a time period to restore their files. The last form (Form1.cs) is the main form for the Sync App. There are also a few classes to format the Json objects received from the cloud server servlets.

There is also a Preference class where you will need to set your server email address, client email address as well as the client's email password. You will also need to set the server IP/hostname as well as the username and password to sftp to the Raspberry Pi. A SharedFunctions (SharedFunctions.cs) class is also created to place functions that are used in multiple forms.

The Sync App would first check for any ransom processes running, followed by known ransomware files as well as ransomware extensions. If there are any found, the backup will stop. Else, it would proceed to get all the files that the user selected to backup. It would then zip it up, encrypt it with the user secret and secret key and then email it to the server account on Gmail using the client account.

In order to get and update the secret key, rooms, notifications, ransomware processes, trusted processes, ransomware files, ransomware extensions and file retention period the Sync App would first get a session key from the cloud server and then encrypt it using the secret key. It would then send a post request to the cloud server with the session key and encrypted session key as well as the timestamp to retrieve all the variables.

In order to delete file versioning and the account, it would send an email to the Gmail server. The user's new processes will also be send by the Sync App hourly to the cloud server.

b. Cloud Server

i. Installing Ubuntu Server 16.04

Ubuntu Sever 16.04 can be downloaded from <https://www.ubuntu.com/download/server>.

ii. Installing Tomcat 8 to Ubuntu Server

1. Login as the root user
2. `sudo apt-get update`
3. `sudo apt-get install tomcat8`
4. Browse to `http://server_IP_address:8080`. You should be able to see the default splash screen.
5. Deploy the Ransom.war file to the Ubuntu server by accessing the Tomcat Web Application Manager at `<serverip>:8443/manager`.
6. Edit `/var/lib/tomcat8/conf/web.xml`.
7. Add the following lines before the closing web app tag to redirect the users to a custom exception page.

```
<error-page>
```

```

        <exception-
type>java.lang.Exception</exception-type>
        <location>/error.jsp</location>
</error-page>
<error-page>
        <error-code>404</error-code>
        <location>/error.jsp</location>
</error-page>

```

8. Save the file and restart tomcat `8. sudo systemctl restart tomcat8`

iii. *Generating a Self-Signed Certificate*

1. `keytool -genkey -alias tomcat -keyalg RSA`
2. Enter the following. `dmit2.bulletplus.com`, `DISM`, `DMIT`, `SG`, `SG`. Enter `yes`.
3. Edit the `/var/lib/tomcat7/conf/server.xml`.
4. Uncomment the SSL part of the file and change it to look like this.

```

<Connector SSLEnabled="true" acceptCount="100"
clientAuth="false" disableUploadTimeout="true"
enableLookups="false" maxThreads="25"port="8443"
keystoreFile="/home/server/.keystore"
keystorePass="<yourkeystorepass>"
protocol="org.apache.coyote.http11.Http11NioProto
col" scheme="https" secure="true"
sslProtocol="TLS" />

```
5. Be sure to change your `keystoreFile` path as well as the `keystorePass`.
6. Browse to <https://<yourserverip>:8443>.
7. You might receive an error message saying that the certificate is not trusted. You can ignore the error and proceed on. You should be able to see the default splash screen.

iv. *Installing the CloudUI*

1. Browse to <https://<yourserverip>:8443/manager>
2. Enter your credentials and then select WAR file to deploy.
3. Upload the `ROOT.war` file and click `Deploy`.
4. Browse to <https://<yourserverip>:8443/>. You should be able to see the login page.
5. Ensure that you have created a directory named `installer` under `/home/server` and place the Sync App installer files in the directory.

v. *Installing MySQL to Ubuntu Server*

1. `sudo apt-get install mysql-server`
2. Follow the instructions on the screen to complete the installation.
3. To access MySQL type `mysql -u root -p`. Enter your root password.
4. If you are able to access it, the installation of MySQL should be completed.

vi. *Populating the MySQL Tables*

1. There is a total of 10 tables under the webapp database.
2. You can use the `.sql` file provided to restore the database.
3. The details of each table are shown below.

t_notification

Column Name	c_id	c_message	c_messtimestamp	c_timestamp
Type	integer	varchar(500)	datetime	datetime
Default Value	auto_increment		current_timestamp	current_timestamp
Nullable	No, Primary Key	No	No	No

*Table 4: t_notification Table Structure***t_ransomprocess**

Column Name	c_id	c_processname	c_info	c_timestamp
Type	integer	varchar(100)	varchar(1000)	datetime
Default Value	auto_increment		(empty string '')	current_timestamp
Nullable	No, Primary Key	No	Yes	No

*Table 5: t_ransomprocess Table Structure***t_trustedprocess**

Column Name	c_id	c_processname	c_info	c_timestamp
Type	integer	varchar(100)	varchar(1000)	datetime
Default Value	auto_increment		(empty string '')	current_timestamp
Nullable	No, Primary Key	No	Yes	No

*Table 6: t_trustedprocess Table Structure***t_userprocess**

Column Name	c_id	c_processname	c_approvedcount	c_denycount
Type	integer	varchar(100)	integer	integer
Default Value	auto_increment		0	0
Nullable	No, Primary Key	No	No	No

*Table 7: t_userprocess Table Structure***t_ransomwarefile**

Column Name	c_id	c_file	c_info	c_timestamp
Type	integer	varchar(100)	varchar(1000)	datetime
Default Value	auto_increment		(empty string '')	current_timestamp
Nullable	No, Primary Key	No	Yes	No

*Table 8: t_ransomwarefile Table Structure***t_ransomwareextension**

Column Name	c_id	c_extension	c_info	c_timestamp
Type	integer	varchar(100)	varchar(1000)	datetime
Default Value	auto_increment		(empty string '')	current_timestamp
Nullable	No, Primary Key	No	Yes	No

*Table 9: t_ransomwareextension Table Structure***t_users**

Column Name	c_id	c_username	c_password	c_role
Type	integer	varchar(100)	varchar(128)	varchar(10)
Default Value	auto_increment			user
Nullable	No, Primary Key	No	No	No

c_noofmonths	c_directorysize	c_salt	c_hash	c_attempts	c_sessionkey
integer	bigint(20)	varchar(50)	varchar(255)	integer	varchar(500)
6	0	NULL	NULL	0	NULL
No	No	Yes	Yes	No	Yes

Table 10: t_users Table Structure

t_secretkey

Column Name	c_id	c_secretkey	c_timestamp
Type	integer	varchar(500)	datetime
Default Value	auto_increment		current_timestamp
Nullable	No, Primary Key	No	No

Table 11: t_secretkey Table Structure

t_session

Column Name	c_sessionkey	c_timestamp
Type	varchar(500)	datetime
Default Value		current_timestamp
Nullable	No	No

Table 12: t_session Table Structure

t_rooms

Column Name	c_id	c_roomname	c_ip	c_sshfingerprint	c_timestamp
Type	integer	varchar(100)	varchar(100)	varchar(100)	datetime
Default Value	auto_increment				current_timestamp
Nullable	No, Primary Key	No	No	No	No

Table 13: t_rooms Table Structure

Note that the first row of data in the t_secretkey table must be “defaultkey123456”. Also, the first row of data in the t_users table should be (1, 'admin', 'C7AD44CBAD762A5DA0A452F9E854FDC1E0E7A52A38015F23F3EAB1D80B931DD472634DFAC71CD34EBC35D16AB7FB8A90C81F975113D6C7538DC69DD8DE9077EC', 'admin', 6, 0, NULL, NULL, 0, NULL). Also note that there should be a user named webapp with the password of “1qwer\$#@!” in the webapp database.

vii. Installing MySQL Community on Windows

1. On your Windows PC, install MySQL community. It can be downloaded from <https://dev.mysql.com/downloads/workbench/>.
2. After the installation has completed, launch MySQL community and click on the plus icon next to MySQL connections.
3. Enter a connection name.
4. Select the connection method to be Standard TCP/IP over SSH.
5. Enter your server’s IP address followed by ‘:’ and the SSH port number. The default port number is 22.
6. Enter the username for your server as well as password.
7. MySQL hostname should be 127.0.0.1.
8. Port number should be 3306.
9. Enter the username as root, followed by the password.

viii. *Installing python scripts*

NOTE: pip3 install as both user and root

1. This script will download attachments from Gmail. It will then decrypt the attachments using the secret key and store them in the appropriate folders. It will also help to delete the file versions that are more than 10 days old as well as when the user requested for to delete all file versioning. Also, it will help the user to delete his account and to count the amount of space that the user has occupied. It will also help to populate the number of users who approved or deny a process.
2. Ensure you have created 2 Gmail accounts. Refer to the Sync App Developer Guide for more details. Change the email addresses and password to the 2 Gmail accounts you created.
3. Also, you will need to change the database username and password if you did not create a user named webapp.

```
db =
```

```
MySQLdb.connect("localhost", "webapp", "lqwer$#@!",  
"webapp")
```

4. Install pip. `sudo apt install python3-pip`
5. Install the library pyAesCrypt. `pip3 install pyAesCrypt`
6. Install the library requests. `pip3 install requests`
7. Enable cron jobs. `crontab -e`
8. Enter this line as one line at the bottom of the file (remember to replace the path).

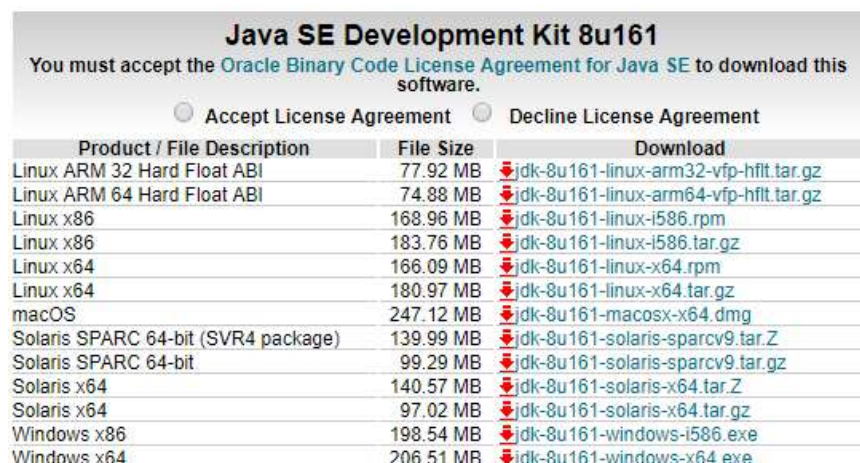
```
*/15 * * * * /usr/bin/python3  
/home/server/downloadattachmentserver3.py
```

9. Enter another line to update the secret key (remember to replace the path).

```
0 0 1 * * /usr/bin/python  
/home/server/randomkeygenerator.py
```

ix. *Installing eclipse on Windows*

1. Go to <http://www.eclipse.org/downloads/> to download and install Java SE.



Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.92 MB	jdk-8u161-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u161-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.96 MB	jdk-8u161-linux-i586.rpm
Linux x86	183.76 MB	jdk-8u161-linux-i586.tar.gz
Linux x64	166.09 MB	jdk-8u161-linux-x64.rpm
Linux x64	180.97 MB	jdk-8u161-linux-x64.tar.gz
macOS	247.12 MB	jdk-8u161-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.99 MB	jdk-8u161-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u161-solaris-sparcv9.tar.gz
Solaris x64	140.57 MB	jdk-8u161-solaris-x64.tar.Z
Solaris x64	97.02 MB	jdk-8u161-solaris-x64.tar.gz
Windows x86	198.54 MB	jdk-8u161-windows-i586.exe
Windows x64	206.51 MB	jdk-8u161-windows-x64.exe

Figure 113: Java SE Download

2. Go to <http://www.eclipse.org/downloads/> and download eclipse oxygen.



Figure 114: Eclipse Download

3. After installing eclipse, launch it. Click on Help → Install new software
4. Select work with all available sites.
5. Search for web.
6. Check the checkbox Web, XML, Java EE and OSGi Enterprise Development and click finish.

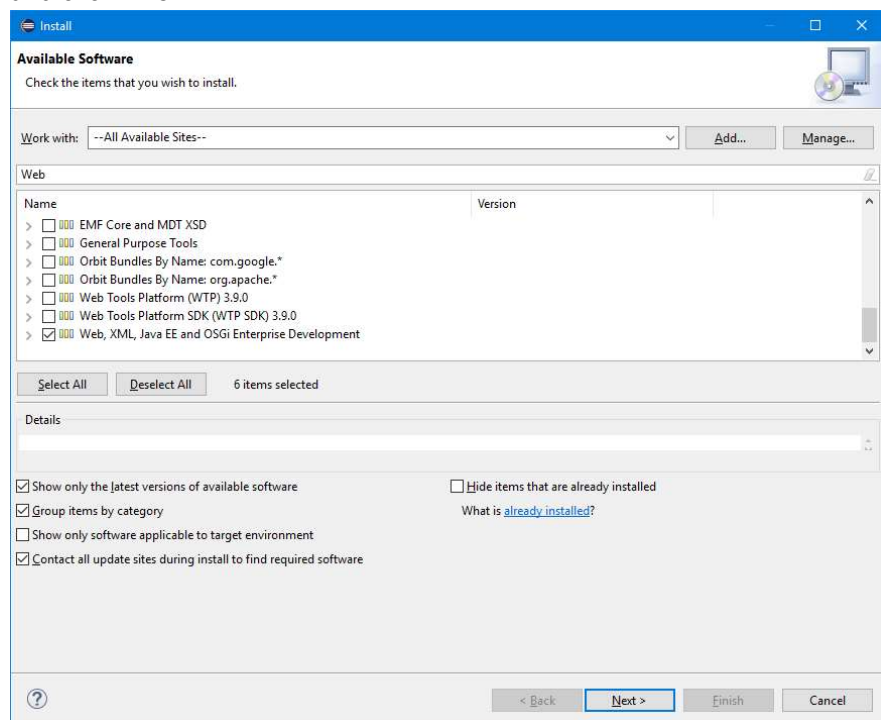


Figure 115: Eclipse Install Software

7. After the installation is done, open the AntiRansom folder. Note that you would need to change the DBConn file to match the username and password of the MySQL database.

x. *Summary of JSP WebUI*

The JSP WebUI will have two main types of users. The user that uses the Sync App to backup and restore files and the domain expert(admin). The JSP files that starts with the word “admin” are used for the domain expert while the others will be used to handle the user requests and shared functions.

There will also be servlets that will serve as a communication method for the Sync App and Raspberry Pi to retrieve data from the database on the server. These servlets will have the word “get” in the name of the servlet. For example, the servlet “getNotification.java” which is used to pull notifications from the server to the Sync App. The servlet “updateNomonths.java” will also be used for the communication.

The servlet “getDataPi.java” will be used to serve only the Raspberry Pi. Do note that you will need to change the email address and password to your client’s Gmail address and password in the settings.java file.

c. Raspberry Pi

i. *Installing Raspbian*

1. Go to <https://www.raspberrypi.org/downloads/raspbian/> and download the RASPBIAN STRETCH LITE image zip file.
2. Refer to this link on how to install the Raspbian on the micro SD card. <https://www.raspberrypi.org/documentation/installation/installing-images/>
3. After burning the image, insert the micro SD card in the raspberry pi and hook it to a monitor. You would also need an external keyboard. Login as user pi with the password raspberry.
4. Type `sudo raspi-config`.
5. Select `interfacing options` and enable `SSH`.
6. Select `localisation options` and set the `timezone` to `Singapore`.
7. To install NTFS support type `sudo apt-get install ntfs-3g`
8. Type `sudo mkdir -p /media/USBHDD1` to create a mount point for your SSD.
9. Type `sudo chmod 770 /media/USBHDD1` to change the directory permissions.
10. Ensure that your SSD is connected. Type `sudo mount /dev/sda1 /media/USBHDD1` to mount the drive.
11. Type `sudo nano /etc/fstab`. Add the following line in the file.
`/dev/sda1 /media/USBHDD1 ntfs defaults 0 0`

ii. *Installing python scripts*

NOTE: pip3 install as both user and root

1. This script will download attachments from Gmail. It will then decrypt the attachments using the secret key and store them in the appropriate folders. It will also help to delete the file versions that are more than 10 days old as well as when the user requested for to delete all file versioning. Also, it will help the user to delete his account. The secret keys will be stored in a text file named ‘`Keys.txt`’.

2. Ensure you have created 2 Gmail accounts. Refer to the Sync App Developer Guide for more details. Change the email addresses and password to the 2 Gmail accounts you created.
3. Install pip. `sudo apt install python3-pip`
4. Install the library pyAesCrypt. `sudo pip3 install pyAesCrypt`
5. Install the library requests. `sudo pip3 install requests`
6. Edit the file. `nano /home/pi/downloadattachmentpi3.py`
7. Change the roomname to the roomname that exist in the MySQL database. `roomname="T2031"#Declare room name here`
8. Also, ensure to change the `detach_dir` variable to where the downloaded files are stored.
9. Also, you might need to change the cloud server address. `cloudserveraddress = "dmit2.bulletplus.com"`
#address of the cloud server
10. Also, you might need to change the directory of the python script. `pythonscriptdirectory = "/home/pi/"#directory of the python script`
11. Enable cron jobs. `crontab -e`
12. Enter this line as one line at the bottom of the file (remember to replace the path)
`*/15 * * * * /usr/bin/python3
/home/pi/downloadattachmentpi3.py`

iii. Overview of Django website

1. The website on Raspberry Pi is using the Django framework which uses the programming language python 2.7.
2. To run the Django website, in the base folder of the Django project there is a file named 'manage.py'. This is the python script to run the website, perform migrations, and other commands.
 1. 'python manage.py runserver' will bring the Django website online and to view it, the website address is '127.0.0.1:8000/sync'. Should there be a need to use the server's own IP address instead of the localhost, add in one more argument to the command above which is `<ip_address>:<port_number>` and just browse to `<ip_address>:<port_number>/sync`
 2. 'python manage.py migrate' will help to deal with the changes made to the database. For example, when mysql is used instead of the default sqlite3.
3. These are the following python libraries installed on the Raspberry PI for Django:
 - i. django 1.11
 - ii. django-jinja
 - iii. jinja2
 - iv. requests 2.10
 - v. pyAesCrypt (works with python 3 only)

iv. *Deploying Django website on Raspberry PI with Apache2*

1. Ensure that python 2 and 3 are installed on the Raspberry PI
2. `sudo pip install Django-jinja`
3. `sudo pip install Django=1.11`
4. `sudo pip install jinja2`
5. `sudo pip install requests==2.10` and `pyAesCrypt` (refer to the python script installation guide for more details)
6. `sudo apt-get install apache2`
7. `sudo apt-get install libapache2-mod-wsgi`
8. `mkdir /etc/apache2/ssl`
9. `openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt`
10. `a2enmod ssl`
11. Run 'a2ensite' and check if `default-ssl.conf` and `000-default.conf` are enabled, both of them should be enabled.
12. Edit the files at `/etc/apache2/sites-enabled/{000-default.conf, default-ssl.conf}`

i. `000-default.ssl`:

Add in the following lines in the virtualhost container

```
WSGIScriptAlias /
/home/user/fyp/fyp/apache/wsgi.py

<Directory "/home/pi/fyp/fyp/apache/">

    Require all granted

</Directory>

Redirect /
https://192.168.196.132/ (Replace with the
IP of the Raspberry Pi)
```

ii. `default-ssl.conf`:

Add in the following lines in the virtualhost container

```
WSGIScriptAlias / /home/user/fyp/fyp/apache/wsgi.py
```

```
<Directory "/home/pi/fyp/fyp/apache/">

    Require all granted

</Directory>
```

```
<Directory "/home/pi/fyp">

    Require all granted

    AllowOverride FileInfo

    Options ExecCGI MultiViews
FollowSymLinks

    MultiviewsMatch Handlers
```

```

        Order deny,allow

        Allow from all

    </Directory>

    Alias /static
    /home/pi/fyp/syncui/static

    <Directory
    "/home/pi/fyp/syncui/static">

        Require all granted

        Options Indexes MultiViews FollowSymLinks

        AllowOverride None

        Order allow,deny

        allow from all

    </Directory>

```

iii. Edit the following values, the bold characters are the values to replace:

- i. DocumentRoot **/home/pi/fyp/**
- ii. SSLEngine **on** #if it is commented, uncomment it also
- iii. SSLCertificateFile **/etc/apache2/ssl/apache.crt** #if it is commented, uncomment it also
- iv. SSLCertificateKeyFile **/etc/apache2/ssl/apache.key** #if it is commented, uncomment it also

13. Copy the Django project files and place it under the home directory of the Raspberry PI user.

- iv. If the user on the Raspberry Pi is not the user specified in the `views.py` file, the `view_file.html` file, and the `aseCrypt.py` file, replace the name 'Pi' with the current user on the Raspberry Pi case sensitive.

14. `systemctl enable apache2`

15. `systemctl start apache2`

16. Browse to https://<IP_ADDRESS>/sync to view the webpage once the website is completely deployed

v. *Guide on working on Django Projects*

1. For Django, the webpages are contained in an app package. This means that now the current Django website has only app which is syncui and the address to access the index of this app is `<website_address>/sync`.
2. Should there be a need to use more than one app for the Django projects, follow these steps (It will also cover on how to add a new webpage for an app):
 - i. In the Django project folder run `'python manage.py startapp <app_name>'` to create the new app, there will

be a new folder which is the new app name and it will contain some files to get the app working.

- ii. Go to `'/<project>/<project>/urls.py'`. This particular file determines which app or page the Django framework will bring the user to when entering a url link.
- iii. Add a new entry in the `urlpatterns` list in `urls.py`.
`"url(r'<name>', include('<app_name>.urls'))"`. For the first parameter, any name is accepted but for the second parameter the `app_name` must be the correct app name for the new app created. Also, since this section uses regular expression, it is possible to use it to make the app more dynamic for its url.
- iv. Go to `'/<project>/<app_name>/urls.py'`, there will be another `urls.py` file which is used for the webpages on that app itself.
- v. For this part, it is okay to copy the `urls.py` file in the `syncui` folder to the new app `urls.py` file. The only changes to be made is to delete all of the `urlpatterns` entries except for the `index` entry which is `"url(r'^$', views.index, name='index')"`
- vi. `url(r'^$', views.index, name='index')`
 1. The first parameter is the url link
 2. The second parameter is which function in the `views.py` in the app folder to use when that url link is browsed
 3. The third parameter is just the name, to keep things organized, just use the name of the function in `views.py` that is called.
 4. To browse to this page the url `'<website_address>/<app>/'` is the correct url to access to the index page. If the first parameter `"r'^view_file/'"` is used, then the correct url link is `'<website_address>/<app>/view_file/'`
 5. For the last step which is to load the webpage, go to `'/<project>/<app_name>/views.py'`.
 6. Ensure that the following is imported
 - a. `from django.shortcuts import render`
 - b. `from django.http import HttpResponse`
 - c. `from django.template import RequestContext`
 7. Follow this step

```
def setting(request, **kwargs):#soon to be deleted
    return render(request, 'syncui/setting.html')
```

Figure 116: Django Function Header

Setting is the function name specified in the app `urls.py` file which should be changed to `index` for this

example. 'syncui/setting.html' is the notation for '<app_name>/<html_file>.html' The return render would load the webpage for the user.

vi. *Description of how the current Django setup works*

Flow of the website when the user logs in successfully

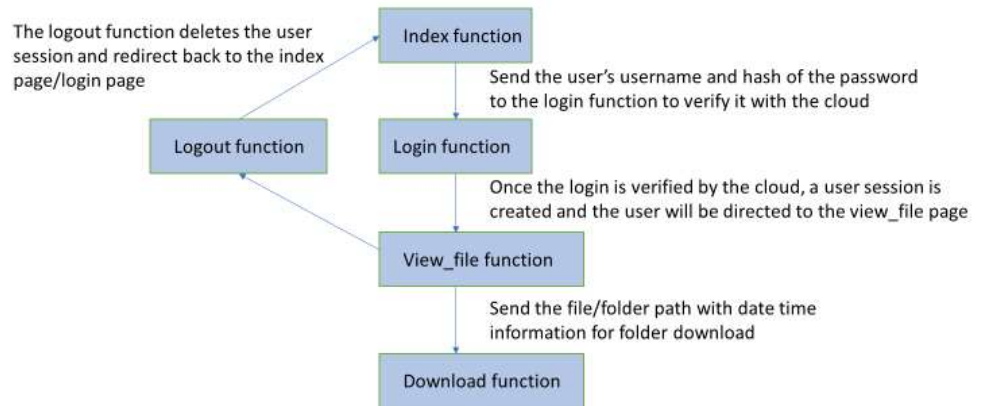


Figure 117: Django User Login Flow

When the user browses to the home page/index page/login page at https://<IP_ADDRESS>/sync, the index function on **views.py** is called and will check if whether the user has a session with the website. If it is, it will call on the function **view_file_new** in **views.py** which will load the **view_file** page for the user. If not, it will just render the home page for the user.

When the user enters in their correct credential information, the user is directed to the page https://<IP_ADDRESS>/sync/login where the **login** function on **views.py** is called and will get the session key from the cloud server, it will encrypt the session key with the current secret key and will send the **encrypted session key** with the **session key**, the **username** and the **hash of the password**.

When the cloud server response which should be a success as the correct credentials are sent, the user session is created and will call on the **view_file_new** function on **views.py** which will load the **view_file** page for the user to download their backup. If the credentials are incorrect or the website cannot communicate with the cloud server, the user is redirected to the index page with the respective error message.

For the **view_file_new** function in **views.py** which loads the **view_file** page, it gets the request parameters which are the **cwd** (base directory of the user's backup when the user first enters the **view_file** page) and a list called **base** which is used to keep track of the subdirectories when browsing in the **view_file** page.

The **view_file_new** function then loads the **view_file** webpage for the user and also sends the following variables as well which are:

- 1) Dict variable whose key is the original filename and the value is a list with the type (file or folder), and the size of the file or folder
- 2) The base list mentioned earlier
- 3) The size of the base list in integer
- 4) The cwd variable which is the current working directory of the directory that the user is browsing to
- 5) A dict variable called version where the key is the original file name and the value is a list with the file name of the different versions of that file
- 6) The user session
- 7) Status variable where its value is a string 'in'

Under the **view_file** page, the user can either browse deeper into the subdirectories of their backup, download a file, folder, or do a full restore. This is what happens when a user downloads a file or folder:

1. File

The user will go to the download url which is https://<IP_ADDRESS>/sync/download_file, the full file path and the file name are sent over as well.

A temporary zip file will be created to zip up the file that is being downloaded, the name of the zip file will be semi randomized to prevent and collision issues over the webserver.

Then the zip file is encrypted using the secret key from the AESCrypt library from a python script called by a call function and the response object is loaded with the AES encrypted file and is renamed to SPSync.sync which will begin the download for the user.

Once the download is done, the zip file and the AES encrypted file are deleted to manage the storage space of the webserver

2. Folder

Similar to the file download, the user will go to the download url which is https://<IP_ADDRESS>/sync/download_file, the full file path, the file name, and the date time value are sent over instead.

A temporary zip file will be created to zip up the file that is being downloaded, the name of the zip file will be semi randomized to prevent and collision issues over the webserver.

Then the zip file is encrypted using the secret key from the AESCrypt library from a python script called by a call function and the response object is loaded with the AES encrypted file and is

renamed to SPSync.sync which will begin the download for the user.

Once the download is done, the zip file and the AES encrypted file are deleted to manage the storage space of the webserver

When a user on the **view_file** page browses to other directories, the current webpage will send the **full path of the directory** to browse to and an **array or list for the base list** in the **view_file_new** function in the **views.py** function. Then like the **view_file_new** function will use the variables that are sent to display the correct contents of the directory in the new **view_file** page.

When the user logs out, the url is https://<IP_ADDRESS>/sync/logout. The **logout** function in **views.py** will be called and will delete the user session and redirect the user to the **index** page of the website.

One thing to note is that under the **view_file** page, all links except for the logout link are **forms** that will send **POST request** to the respective webpages.

All of the urls are **session protected**.

vii. *Guide to current Django setup*

1. For the current Django project, the static files are stored in `syncui/static/<css/fonts/img/js>/`
2. The `html` files are stored in `'syncui/jinja2/syncui/<filename>.html'`. The reason why it is stored in this way is due to how jinja2 works.
3. In `fyp/fyp/jinja2.py`, this is the python script that is required by jinja2 in order to use jinja2 for the webpages.
 - i. Should there be a need to use a function in the webpages, create a function in `jinja2.py`
 - ii. Then add a new line in the environment function `'env.globals['<function_name>'] = <function_name>'` and the webpages will be able to use this particular function using jinja2.
4. Jinja2 Guide:
 - i. Jinja2 is a templating language where it's syntax is similar to python's own syntax
 - ii. To create a webpage using the current base template created in this project just do this in the new html file:

```
{% extends "syncui/header.html" %}

{% block content %}

    <your_html_content_with_jinja2_stuff>

{% endblock %}
```

- iii. There are many things available in jinja2 such as:
 1. `{% set no = 1 %}` which creates a variable `no` with the value 1.
 2. `{% for var in list %}{% endfor %}` is the syntax to use for for loops.
 3. `{% if <condition> %}{% endif %}` for if statements.

4. When the webpage using jinja2 needs to perform an operation like appending a new item in a list, do this:

```
{% if current_1.append(i) is none %}  
{% endif %}
```

This will prevent 'none' from printing out when the webpage is displayed.

5. `{{var}}` will print out the variable var in the webpage.
5. Function guide on the views.py
 - i. Functions responsible for the flow of the website
 1. index (home/index/login page)
 2. download_file
 3. view_file_new (file explorer page)
 4. login (do the login validations)
 5. logout
 - ii. Other functions responsible to complete other tasks
 1. zipdir (zip folders and exclude files whose versions are earlier than the date time specified)
 2. format_chromeDT (Format the date time value for chrome browsers before parsing the value in the zipdir function)
 3. get_fakedir
 4. getfoldersize
 5. format_dir_name