smartsacco

Microfinance Management

V1.2.0

Administrator Manual

1 System Requirements

smartsacco Mircofinance Management is a software solution for small scale micro-finance institutions. It was first developed for Taifa sacco (Savings and Credit Cooperative) in Kenya.

1.1 Server-side Requirements

As smartsacco is based on web technology, you will need a web server for the application and the database to run on. This can either be a remote server which you access over the Internet, or a local web server running on any local device. Since smartsacco was designed to run with minimal system requirements, even a simple laptop can provide sufficient power to host both the application and the database.

You have to consider carefully the advantages and disadvantages of hosting smartsacco either locally or remote. While both options clearly have their merits, both of them also come with some trade-offs. The following overview shall help you to make an informed decision.

1.1.1 Remote Web Server

Advantages of a remote web server:

- You can access your system from any Internet-capable device around the world.
- You don't have to worry about backups. Virtually all hosters will perform regular backups of your data for you.
- There is no need for you to be concerned with technical maintenance of server systems. Your web hoster would be responsible for continued service.

Disadvantages of a remote web server:

- You need to make sure your institution is constantly connected to the Internet. If Internet connection fails you will be unable to access your data. Therefore, a disruption in Internet service will result in a disruption of your business operations.
- In addition to network connection, you would also have to pay for the services of the web hoster. You would have to make sure you can cover these bills on a regular basis. Failure to pay the web hoster will result in your application being taken offline together with your web space.
- When using a remote web server, you have to ensure your connection is always secured using HTTPS protocol. Otherwise you would end up transmitting sensitive data over unprotected network connections. Using HTTPS requires an SSL certificate which might incur extra cost.
- Hosting sensitive data remotely raises concerns about data security and integrity. Although smartsacco includes prudent security features, there is no guarantee your system would not become a target of malicious attacks.

Summary:

Using a remote web server to host smartsacco Microfinance Management is the more convenient option as backups and technical maintenance will be taken care of. It is, on the other hand, more costly and depends on the constant availability of services. Data security has to be considered as well.

1.1.2 Local Web Server

Advantages of a local web server:

- No need for constant Internet access to operate your business.

- Virtually no delay in accessing the smartsacco application.
- Locally kept data will always be under your control.

Disadvantages of a local web server:

- You have to make sure your system is professionally maintained. In case of any problem, you or a local service provider of your choice will be responsible to fix it.
- You have to find a solution to backup your database on a daily basis. Without a proper backup solution, you will be extremely vulnerable to data loss. (See chapter 5 for a discussion of possible solutions.)

Summary:

Using a local web server to host smartsacco Microfinance Management is the cheaper and possibly safer option. It is, on the other hand, more labour-intensive as server maintenance and data backups fall under your responsibility. Setting up and maintaining your own web server requires a certain level of IT expertise from either inside or outside your organisation.

1.2 User-side Requirements

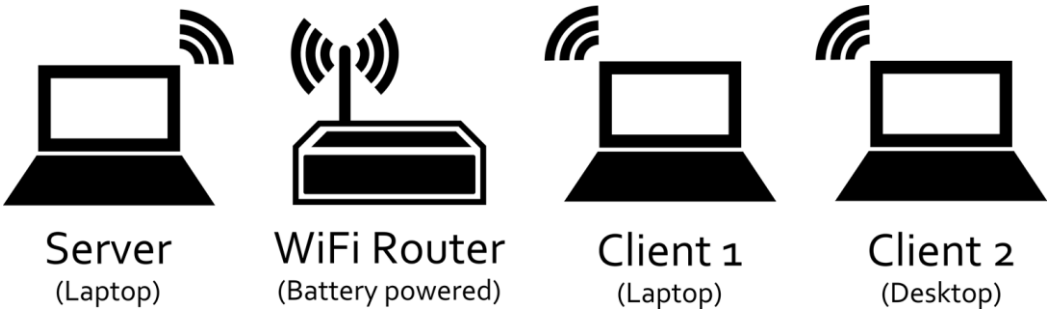
Users will require a standard web browser to use the program. It is recommended to use Mozilla Firefox, Chrome, or Chromium on the client side.

1.3 Network Requirements

If you are going to use more than one computer for accessing smartsacco Microfinance Management, you will require some kind of local area network infrastructure. You may choose between a wired LAN or a wireless Wi-Fi solution. Wireless networks are generally less secure than wired ones because an intruder does not need physical access to the network. On the other hand, you can find many mobile Wi-Fi routers in the market which are battery powered. To make a standard LAN router equally independent from constant power supply would require some tinkering.

1.4 Exemplary Setup

The following describes the setup that is used. This is meant to serve merely as an example for one possible IT-infrastructure layout to operate smartsacco Microfinance Management. Please feel very free to pursue other ways of implementation.



Three of four devices can be powered using a battery. In case of a power failure, only the desktop PC will go out of service.

Regarding operating systems, only one client laptop runs on a Microsoft Windows system. The server as well as the desktop computer are operated under Linux Mint. In an environment where Linux systems are extremely uncommon and malware is primarily spread via USB flash drives, using Linux helps to prevent your systems becoming infected.

2 How to Setup a Web Server on Ubuntu-based Linux

The steps in this tutorial require you to have installed Ubuntu or any other Ubuntu-based Linux distribution on your machine. Please refer to online tutorials of your respective distribution for assistance with the setup process.

2.1 Installing Apache

Apache is a free open source software which runs over 50% of the world's web servers.

To install Apache, open the terminal and type in these commands:

```
sudo apt-get update
sudo apt-get install apache2
```

To check if Apache is installed, direct your browser to your server's IP address (for example http://127.0.0.1 for your localhost). The page should display the words "It works!".

2.2 Installing MySQL

MySQL is a powerful database management system used for organizing and retrieving data. To install MySQL, open a terminal and type in these commands:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

During the installation, MySQL will ask you to set a root password. If you miss the chance to set the password while the program is installing, it is very easy to set the password later from within the MySQL shell. It is recommended, however, to choose a password during the installation process.

Once you have installed MySQL, we should activate it with this command:

```
sudo mysql_install_db
```

Finish up by running the MySQL set up script:

```
sudo /usr/bin/mysql_secure_installation
```

The prompt will ask you for your current root password. Type it in.

```
Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

Then the prompt will ask you if you want to change the root password. Go ahead and choose N and move on to the next steps.

It's easiest just to say Yes to all the options. At the end, MySQL will reload and implement the new changes.

```
By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? [Y/n] y
... Success!
```

```
Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...
```

Once you're done with that you can finish up by installing PHP.

2.3 Installing PHP

PHP is an open source web scripting language that is widely used to build dynamic web pages. To install PHP, open a terminal and type in this command.

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

After you answer 'Yes' to the prompt twice, PHP will install itself. It may also be useful to add php to the directory index, to serve the relevant php index files:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Add index.php to the beginning of index files. The page should now look like this:

```
<IfModule mod_dir.c>
        DirectoryIndex index.php index.html index.cgi index.pl  index.xhtml index.htm
```

```
</IfModule>
```

smartsacco requires the PHP GD module to handle picture files. To install this module, open a terminal and type this command:

```
sudo apt-get install php5-gd
```

Congratulations! You now have completed installing all components of the LAMP stack (**L**inux, **A**pache, **M**ySQL, **P**HP) on your server!

2.4 Double-checking your components

Although LAMP is installed, we can still take a look and see the components online by creating a quick PHP info page. To set this up, first create a new file:

```
sudo nano /var/www/html/info.php
```

And add in the following line:

```
<?php
    phpinfo();
?>
```

Then save your file and exit.

Restart Apache so that all of the changes take effect:

```
sudo service apache2 restart
```

Finish up by visiting your PHP info page, for example under (make sure to replace the example IP address with your correct one):

```
http://127.0.0.1/info.php
```

You should see a page displaying all relevant system information. Make sure to delete this file after successfully testing your server installation.

2.5 Installing phpMyAdmin

phpMyAdmin is a free web software to work with MySQL on the web. It provides a convenient visual front end to the MySQL capabilities. The easiest way to install phpMyAdmin is through the console using apt-get:

```
sudo apt-get install phpmyadmin apache2-utils
```

During the installation, phpMyAdmin will walk you through a basic configuration. Once the process starts up, follow these steps:

- Select Apache2 for the server.
- Choose 'Yes' when asked about whether to configure the database for phpMyAdmin with dbconfig-common.
- Enter your MySQL password when prompted.
- Enter the password that you want to use to log into phpMyAdmin.

After the installation has completed, add phpMyAdmin to the Apache configuration by typing:

```
sudo nano /etc/apache2/apache2.conf
```

Add the phpMyAdmin config to the file. It should look like this:

```
Include /etc/phpmyadmin/apache.conf
```

Restart Apache by typing:

```
sudo service apache2 restart
```

You can then access phpMyAdmin by directing your web browser to (make sure to replace the example IP address with your correct one):

```
http://127.0.0.1/phpmyadmin
```

2.6 Securing phpMyAdmin with .htaccess

Unfortunately, older versions of phpMyAdmin have had serious security vulnerabilities including allowing remote users to eventually exploit root on the underlying server. One can prevent a majority of these attacks through a simple process: locking down the entire directory with Apache's native user/password restrictions which will prevent these remote users from even attempting to exploit older versions of phpMyAdmin.

To set this up start off by allowing the .htaccess file to work within the phpMyAdmin directory. You can accomplish this in the phpMyAdmin configuration file:

```
sudo nano /etc/phpmyadmin/apache.conf
```

Under the directory section, add the line “AllowOverride All” under “Directory Index”, making the section look like this:

```
<Directory /usr/share/phpmyadmin>
    Options FollowSymLinks
    DirectoryIndex index.php
    AllowOverride All
    [...]
```

With the .htaccess file allowed, we can proceed to set up a native user whose login would be required to even access the phpMyAdmin login page. Start by creating the .htaccess page in the phpMyAdmin directory:

```
sudo nano /usr/share/phpmyadmin/.htaccess
```

Follow up by setting up the user authorization within .htaccess file. Copy and paste the following text in:

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /etc/apache2/.phpmyadmin.htpasswd
Require valid-user
```

Below you’ll see a quick explanation of each line:

- **AuthType**: This refers to the type of authentication that will be used to the check the passwords. The passwords are checked via HTTP and the keyword Basic should not be changed.
- **AuthName**: This is text that will be displayed at the password prompt. You can put anything here.

- **AuthUserFile**: This line designates the server path to the password file (which we will create in the next step.)
- **Require valid-user**: This line tells the .htaccess file that only users defined in the password file can access the phpMyAdmin login screen.

Now we will go ahead and create the valid user information. Start by creating a .htpasswd file. Use the htpasswd command, and place the file in a directory of your choice as long as it is not accessible from a browser. Although you can name the password file whatever you prefer, the convention is to name it .htpasswd.

```
sudo htpasswd -c /etc/apache2/.phpmyadmin.htpasswd username
```

A prompt will ask you to provide and confirm your password. Once the username and passwords pair are saved you can see that the password is encrypted in the file.

Finish up by restarting Apache:

```
sudo service apache2 restart
```

phpMyAdmin will now be much more secure since only authorized users will be able to reach the login page. Accessing *http://127.0.0.1/phpmyadmin* (make sure to replace the example IP address with your correct one) should display a login form. Fill it in with the username and password that you generated. After you login you can access phpMyAdmin with the MySQL username and password.

3 How to Setup a Web Server on Windows

This chapter is under development. Meanwhile, consult Mr. Google...

4 Installing smartsacco on your web server

Once you have the web server running, you can install the smartsacco application and database on it.

4.1 Get smartsacco

Download the latest release version of smartsacco Microfinance Management from *https://www.smartech.com/smartsacco*. Choose either the ZIP- or the TAR.GZ-file.

Unpack the archive into the root document directory of your webserver. On Ubuntu-based Linux systems, this will usually be under */var/www* or */var/www/html*.

4.2 Setting up smartsacco

4.2.1 The automatic smartsacco Setup Assistant

Start your web browser and direct it to your server's address. If you are working on the same machine which acts as the web server, you can use *localhost* or *127.0.0.1*.

Because there is no database available yet, you will be directed to the smartsacco Setup Assistant.

The **database host** is the web server that hosts the MySQL engine. Type in the respective address. If the database is hosted on your current local device, type *127.0.0.1*.

The **database user** is the MySQL user name which you created during the MySQL setup process. If you are using a remote server, your hoster should provide you with a database user.

The **database password** is the MySQL password which you created during the MySQL setup process. If you are using a remote server, your hoster should provide you with a database password.

The **database name** can be any name you choose for your database. If the specified database does not yet exist on the MySQL server, it will automatically be created for you.

Finally, choose which **type of database** you would like the assistant to import for you. In case you are setting up a productive system, choose *Fresh Database*. This will give you a clean system to start with. In case you are installing smartsacco for testing purposes, choose *Test Database*. This contains datasets of 100 fictional customers and 12 employees.

Clicking Setup will start the import of the selected database type.

The database import will take a few minutes to complete. Make sure to not interrupt the process. Do not leave the page! You will be forwarded automatically after the import is complete.

If you imported a fresh database, smartsacco Setup Assistant will ask you to create a new administrator account after the import. If using the test database, the default login is:

Username: *admin*
Password: *password*

4.2.2 Manual Setup

If, for any reason, the automatic setup assistant doesn't work for you, you can also manually configure smartsacco Microfinance Management.

4.2.2.1 Database

First, set up the database on your MySQL server with any name you may choose. Then import either *smartsacco_fresh.sql* or *smartsacco_test.sql*. You will find either file in the directory called *database* in the downloaded smartsacco package.

4.2.2.2 config.php

Secondly, use any text editor to create a new file named *config.php* in the sub-directory called *config*. The file must contain the following:

```
define ('DB_HOST', 'HOST ADDRESS');
define ('DB_USER', 'DATABASE USERNAME');
define ('DB_PASS', 'DATABASE PASSWORD');
define ('DB_NAME', 'DATABASE NAME');
```

Replace the yellow highlighted items with the correct information about your installation. You may use the *config.php.example* file as a template.

4.2.2.3 Create a first user account

In case you imported the fresh version of the smartsacco database (see 1.2.0), you will have to manually create a first user account.

You will need to generate a peppered and salted password hash. In order to generate this hash, direct your browser to the following page:

http://[YOUR-SERVER]/[YOUR-SMARTSACCO-DIRECTORY]/setup_pwhash.php

Choose a password of at least six characters and enter it two times into the respective fields. When you click Generate the assistant will first create a new system-wide password pepper (stored under */config/pepper.php*) and secondly salt and hash the entered password. The resulting hash will show in an information box below the form. Select and copy the full hash string.

In your database, access the *user* table and insert a new entry with the following values:

- **user_id***:* Leave the ID empty as this value has to be automatically assigned by MySQL.
- **user_name**: Type a username of your choice.
- **user_pw**: Paste the salted password hash you generated and copied earlier.
- **usergroup_id**: Type *1* since the first user must have administrator rights!
- **user_created**: Leave this field empty.

You may now direct your browser to the root directory of your smartsacco installation and log into the system using the newly created username and password.

5 Backups

Particularly when using a locally hosted database, it is most important to implement a good backup strategy. For backups to be useful, they need to be **comprehensive** and **regular**. Without this, you run a high risk of data loss.

5.1 Backup Storage Strategies

There are at least two different strategies of the storage of backup. They both come with their own assets and drawbacks. The first option is to use an external, but local storage device, such as a flash drive or a USB hard-disk. The alternative would be to upload data backups to an online storage service (cloud-based solution).

5.1.1 External Local Storage Device

Advantages:

- Apart from a one-off investment in the device, this solution comes at no cost.
- No need for Internet access to store backup files.
- Availability of backup files is equally independent from Internet access.

Disadvantages:

- Backups must be stored manually. This requires a high degree of organisational discipline.
- The external storage device must be kept in a safe location separate from the database host.

Summary:

Using a local backup solution is the cheaper option. However, the storage device needs to be operated manually by a human maintainer. This is what makes this option extremely prone to error. Moreover, safe storage of the backup device must be guaranteed at all times. Storing the host computer and the backups within the same room (or even building) will inevitably nullify the efforts of keeping backups in the first place. Using this option you must ensure your backups will protect you against all possible types of data loss, such as technical failure, theft, fire, and others.

5.1.2 Online Storage Service (Cloud-based)

Advantages:

- Backups can be fully automated. No need for interaction from a human user.
- This guarantees regularity.
- Separate locations protect against theft and other physical risks.

Disadvantages:

- Constant Internet access is vital for this option to work.
- Storing backups online might compromise the privacy and confidentiality of your data.

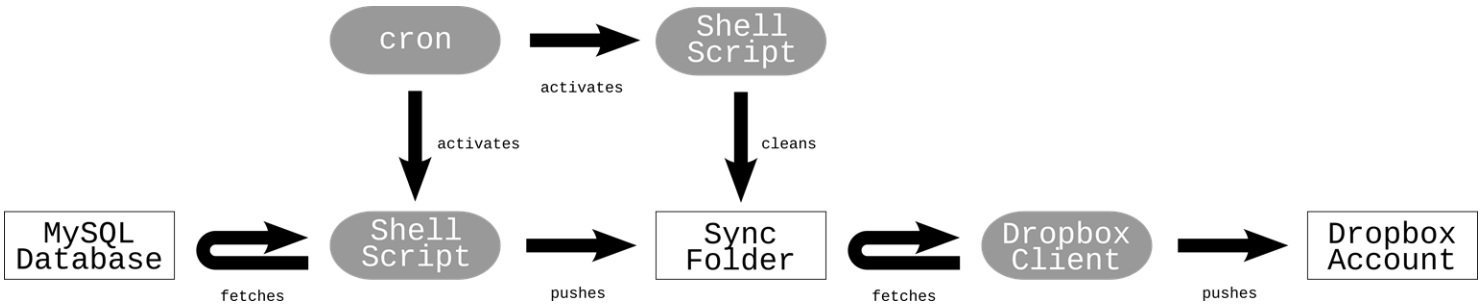
Summary:

Storing your backups online is by far the more convenient option, since uploads can be fully automated. This option guarantees regular backups without the need for any human interaction. However, you will need to ensure to have permanent Internet connection. (The required data volume, however, is very small.)

5.2 Exemplary Backup Solution for Linux Systems

The following describes their backup solution and the technologies it uses.

5.2.1 Online Storage Service (Dropbox)



The client application creates a sync folder on installation. Any file copied into this folder will automatically be uploaded to the online storage account. Since the client does all this by itself, we do not have to take care of any of the uploading ourselves.

5.2.2 Backup Shell Script

The backup shell script is the core component of the SACCO's backup solution. Each time it is called, it creates a new dump from the MySQL-database and saves it to the sync folder of the Dropbox client. In addition, the script also copies all uploaded files (like pictures and documents) to the sync folder.

```
#!/bin/sh

_now=$(date+"%Y-%m-%d")

_file="./Dropbox/db_backup/mangoo_bckp_$_now.sql"

mysqldump --user DBUSER --password=DBPW –opt DBNAME > "$_file"

cp -r SMARTSACCOFOLDER/uploads/* ./Dropbox
```

Replace the yellow highlighted items with the correct information about your installation and save it to your home folder as *.smartsacco_bckp* (note the leading dot!).

5.2.3 Clean-up Shell Script

As the sync folder will inevitably fill up over time, it is reasonable to implement a clean-up script. This script identifies and deletes backup files older than a defined number of days. When files get deleted from the sync folder, they will also be deleted on the online storage account.

```
#!/bin/sh

find ./Dropbox/db_backup -mtime +90 -type f -delete
```

5.2.4 Cron

Cron is a time-based job scheduler on UNIX-like systems. In our backup scenario, we use cron to regularly fire up our shell scripts. Since cron is driven by a table file called crontab, we need to make some additions to that file.

```
30 12 * * * ~/.smartsacco_bckp

30 16 * * * ~/.smartsacco_bckp

35 16 * * * ~/.smartsacco_bckp_del_old
```

As you can see, the backup bash script is called twice a day; at 12:30pm and at 4:30pm. You may adjust the times according to your need and business hours. At 4:35pm the clean up script is called (see 5.2.3).