



Hope Matters
Medical Data Base
User's Guide

Introduction

This program is an interface with a data base which will hold and manage all patient records. The program will give Hope Matters employees access to enter, retrieve, and modify patient information in the data base. All prior forms and records storage systems have been integrated into this system as well as patient finance management. The goal is to streamline records management and make information easier to enter and retrieve.

User Types

There will be three different user types with different system permissions assigned to them. They are illustrated in the chart below. Your user type will be determined by your assignment and will allow you the access permissions you need to perform your job tasks. Your user type will be assigned by the system administrator(s).

User type	Input/review ability	Finance access	Data base admin
Basic	Yes	No	No
Finance	Yes	Yes	No
Admin	Yes	No	Yes

Logging into the System

To access the system, click on the icon on your computer desktop. The icon is shown here on the right. This will bring you to the log in page. If you are accessing the system via internet during the beta testing period, you must use this

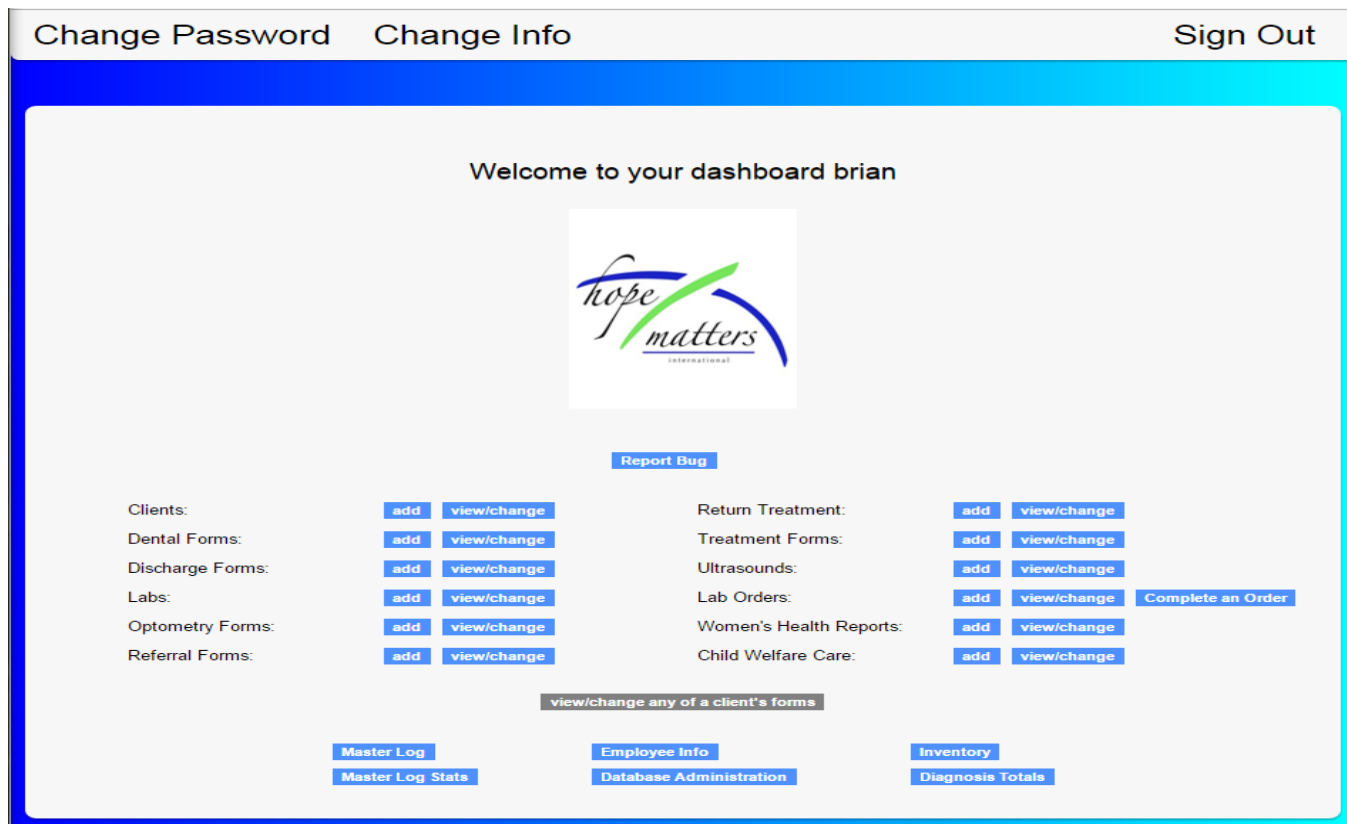


link <http://andrew-klassen.net:100/>. In order for the program to work correctly you must be using a supported browser such as Google Chrome, FireFox, or Edge. A system administrator will assign you a user name and an initial password. The first time you log in, you should change your password so only you know it. **Your password must be at least 8 characters and contain at least one upper and one lower case letter.** Enter your user name and password in the screen as seen here on the left and click "Login." This will take you to the "Dashboard."

A screenshot of the login page for the Hope Matters system. It features the Hope Matters International logo at the top. Below the logo, the text "Please sign in" is displayed. There are two input fields: "Username" and "Password". Below these fields is a blue "Login" button.

The Dashboard

The dashboard is the central hub of the system and is where you can access all parts of the system. It looks like this.



Notice it shows your name at the top center. If you don't see your name, someone else is logged onto the computer. You need to log off and log back in with your user name and password.

At the top left we see the words, "Change Password." Any time you wish to change your password, you can do so by clicking those words. Next to that is "Change Info." By clicking here, you can change your personnel information. At the top right are the words "Sign Out." Click here to sign out.

In the center of the dashboard is a list of all of the forms in the system. To add any of these forms, click on the word, "add" next to the form you wish to add. This will take you to a page where you can add this type of form. If you wish to retrieve a form to review or make changes, click on the words, "view/change" next to the form you wish to see. This will take you to a page where you can see forms already in existence and make changes if needed. If you wish to see all forms pertaining to a specific patient you can do so by clicking on "view/change all of a client's forms" at the bottom/center of the Dashboard. Below that we see "Master

Log,” “Employee Info,” “Inventory,” “Master Log Stats,” “Database Administration,” and “Diagnosis Totals.” These options will be reviewed later in this instruction manual.

Add View/Change Forms

The center of the dashboard lists each of the available form types. From here you can choose a form and either add a new one, or view or change an existing form. When seeing a new client, you must first add the client on the “Client” form. Once added you can create any other form type by clicking on the form and looking up the client. The Add and View/Change functions work similarly for all forms.

Adding a Client:

To add a client, click on the “add” icon next to “Clients.” This will take you to the form shown on the right. Enter the information as requested by each box. When you are finished, click on the blue box at the bottom that says “Add New Client.” The client information will be saved and the boxes will go blank. The computer data base will assign the new client a client ID number. The curser will return to the “First Name” box. This allows you to enter multiple clients. Note that the date of birth must be entered with a two digit month first, then two digit day followed by four digit year. This insures the information will go into the data base in a form it recognizes. If a box is not applicable, it can be left blank.

Viewing or Changing a Client:

To view or change a client’s information, click on the “view/change” button next to the word “Clients.” This takes you to a search page. If you know the client’s assigned ID number you may retrieve their information by putting their number in the box and clicking on “Search.” You may also search by entering either the first or last name of the client in the box and clicking

The form is titled "Please Fill out the Form" and contains the following fields and options:

- First Name
- Last Name
- Date of Birth: mm/dd/yyyy
- If Child, Guardian Name
- National ID
- Phone Number
- Occupation
- Education
- Village/Location
- Emergency Contact
- Allergies
- HIV status: ☐ + ☐ - ☐ unknown
- Alcohol Use: ☐ never ☐ sometimes ☐ often
- Gender: ☐ Male ☐ Female
- Significant Medical History: (text area)
- Regular Medications or Herbs: (text area)
- At the bottom is a blue button labeled "Add New Client".

The page has a blue header with the text "Find Clients". Below the header, it says "Click on a client to edit their information." and "Search for a client by ID or name. Use * to see all clients." There is a search box with a "Search" button and a "Date of Birth (optional) mm/dd/yyyy" field.

on “Search.” Adding a date of birth is optional. If you wish to see a list of all people listed in the data base, simply put an asterisk in the search box and click “Search.” This will give you a list of clients listed by their order of entry into the data base from oldest to newest.

To see the client's information, click on their name and their client form will come up.

Other Forms:

Once a new client has been added, you can add any of the other forms for that client. All of the other forms work similarly to the New Client form. You add and view or change them in the same way. On the top left corner of your dashboard will be the words "Form Selection." Click here to search for the same form on another client. When you complete a form, click on the blue "Submit" box at the bottom of the page to save it. Notice your user name will be shown above the "Submit" box.

To add a form, click on the word "Add" next to the name of the form you wish to add. You will be taken to a search page similar to the one shown above. Search for and choose the client you wish to add the form for. The form will come up on the screen with the client's general information on top. The client's information cannot be changed here. If you wish to change it, you must go back to the "Client" form.

Inputting Information:

Text boxes: Many of the input fields are text boxes. In these boxes you can type the information however it is appropriate. The computer does not request any specific form

Check boxes or circles: If there is a check box or circle next to the options listed, click on the box or circle next to the choice or choices that are appropriate. Boxes allow you to choose as many as are appropriate.

Circles only allow one choice.

1. **Subjective Complaint:** ☐ Far Vision ☐ Near Vision ☐ Hypertension

Number boxes: If a number is required, there will be a number box. Only a number can be entered into this box. You can type the appropriate number in the box. If you click on the box, arrows will appear. You can tap on the arrows to increase or decrease the numbers as shown here.

No Lenses			With Lenses	
Far	Pinhole	Near	Far	Near
1	0	0		

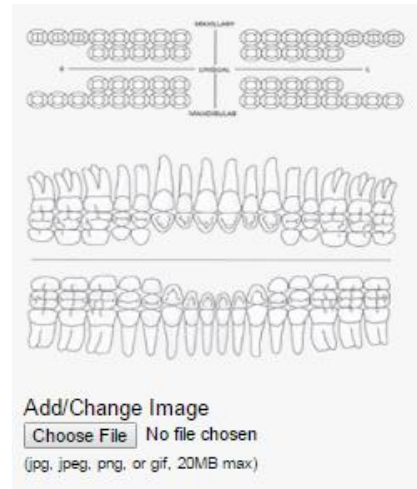
Dates: When entering a date, enter it with a two digit month first, then two digit day, followed by the four digit year. You can also click on the down arrow to the far right of the date box which will bring up a calendar. Click on the date of birth on the calendar.

Drop Down Menus: Some forms contain drop down menus. When you click on the box, you will be given a list of choices. You must pick from the choices given. The program will not accept any other input. This makes the input uniform and easier to retrieve statistics later.

Photos: Some forms such as the dental form or women's health report allow the inclusion of a picture such as a diagram. You can see an example from the dental form here on the right. To write on the picture first go to the copy of the picture/diagram you have in another file on your computer.

Open the picture in "Paint" and make any necessary marks or comments on the diagram. When you are done, save the file. Then come back to the dental form and click on the "Choose File" button at the bottom. This will take you to a search page on your computer. Choose the picture you just completed and

save it to this form. The picture will then appear in place of the one on the form. If you do not already have a copy of this picture saved on your computer in another location, you can right click on the picture on the dental form and select "copy image." Then open "Paint" and hold down the "Ctrl" key and the letter "V." This will transfer the photo to "Paint." The save and transfer the photo as explained above. Depending on the internet browser you use, the photo may appear a bit distorted when saved on the form. This is normal. If you are reviewing the picture and need to see an undistorted copy, click on the words "Download Image" just below the picture. This will bring up a copy of the picture in an undistorted format.



Lab Orders:

Ordering Labs: To order labs, go to the lab orders form and add a form just like you would add any other form. When the lab form appears for the client you have chosen, select the lab tests you want to have performed by clicking on the square box next to the lab test desired. When you are done, click on the blue "Submit Lab Order" box at the bottom of the screen.

Completing Lab Orders: To input the lab test results, go to the "Complete an Order" box next to "Lab Orders." When you choose the client lab form, the labs requested will have a checked box next to them. Enter the lab test results in the boxes given. If the lab test does not have a check in the box next to it, it is not being requested and will not allow you to input any results.

Treatment Form:

The treatment form allows you to document your findings and treatment plan. You create a treatment form for a client the same way you create other forms. The treatment form has a series of text boxes that allow the user to input information regarding their findings and plan. At the bottom of the form is a blue box that says, "Add Diagnoses." Click on this box to add a diagnoses. This will take you to a different screen with a diagnosis box. The diagnosis box is a drop down menu. You must choose a diagnosis from the list given. Choose the diagnosis you wish to use and click "add," The diagnosis will appear below. If you wish to remove a diagnosis, type the diagnosis you wish to remove in the diagnosis box and click "remove." The diagnosis will disappear from the list.

Additional Options

Master Log:

Intro: This is where financial information and records such as billing transactions are kept and managed. When you click on the Master Log, you will go to a client search screen. To search for an existing financial transaction, use the Search box near the top of the screen. To create a new financial transaction, use the blue “Pick Client” box at the bottom of the screen. Search for the client you are looking for.

New Financial Transactions: Use the “Pick Client” box at the bottom of the screen to find an existing client you wish to create a financial transaction for. When you choose the client, you will be taken to this screen.

Client ID: 73 Client Name: John Smith

Enter All Transactions

Payment ID	First Name	Last Name	Client ID	Dept.	Revisit	Billed	Paid	Owes	Time Edited
------------	------------	-----------	-----------	-------	---------	--------	------	------	-------------

New Client

Client's total owes: \$0

Add a Payment

Client Name: John Smith

Department: General

Revisit: ☐ yes ☒ no

Billed:

Paid:

Notes:

Submit

On the right side of the form under “Add a Payment,” you should see your client’s name. Choose the department from the drop down box. Enter the amount to be billed and the amount the client has payed. You can add any notes you like in the text box below. When you submit this, the information will show on the left. The most recent transaction will always be listed on top. The total amount the client still owes will be shown at the bottom. See the picture below to see what this looks like. To review transactions of only a certain type, you can chooses the transaction types you wish to see from the dropdown box at the top of the screen and click the “Enter” box. You will only see the transaction types you requested, however the amount the client still owes will be shown for all transaction types.

Client ID: 73 Client Name: John Smith

Payment ID	First Name	Last Name	Client ID	Dept.	Revisit	Billed	Paid	Owes	Time Edited
67	John	Smith	73	general	yes	\$20	\$5	\$15	Feb 14 2017 03:03 PM
66	John	Smith	73	general	no	\$10	\$0	\$10	Feb 14 2017 02:59 PM

Client's total owes: \$25

Existing Financial Transactions: to view or ammend a financial record, use the search box at the top of the Master Log form to find the financial record you are looking for. You can search using the client's name or an asterisk to get a list of all clients. You can select a specific transaction type from the drop down box to the right or search for all transactions pertaining to that client. This will give you a list of financial records. choose the record you wish to view. The record will look like the one to the right. You can make changes to the record such as add money payed or adjust the billed amount. When you submit this form, the records will be amended. You can then search for the client the same way you did above when creating the new financial transaction to see what if anything the client still owes.

Change the Payment

Payment ID: 67

Client Name: John Smith

Department:

Revisit: ☒ yes ☐ no

Billed:

Paid:

Notes:

Master Log Stats

Master Log Stats shows you an overview of all financial transactions showing how many client encounters have occurred, how much was billed and paid for each type, and how much is owed. This can be viewed in total or from and to specific dates as entered at the top of the form.

Employee Info

Employee Info allows you to search through a list of Hope Matters employees for basic information such as Job title, normal work location and hours. It also shows the employees contact information such as email and phone number. You can not change this information. If you wish to change your personal information, you can do so from the dashboard by selecting "Change Info" at the top of the page.

Database Administration

This link is only available to database administrators. It will take the user to a separate program which will allow you to review the raw database data this program generates. This will allow you to generate statistical information and manage the information contained in the program.

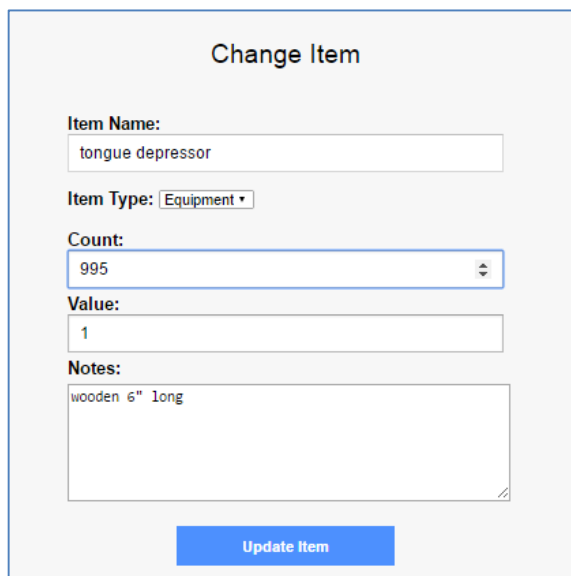
Inventory

Inventory allows you to manage medications and supplies. From here you can add items, then subtract them as they are used, or add when they are resupplied.

Adding Items: To add a new item to inventory click on the "Add Item" box at the top of the page. This lets you add a new item to the inventory that does not already exist. This is not for adding additional units of an item you already have in inventory. When you click the box, you will be asked for an item name such as tongue depressor or stethoscope. From the drop down box below, choose whether the item you are entering is a supply, medication, or piece of equipment. In the "Count" box put in how many of the items you are entering into inventory. Next give their individual value in Kenyan Shillings. You may add any other

necessary information in notes such as a brand name or color. Save the item by clicking on the blue “Add Item” box at the bottom.

Searching items already in inventory: To review or ammend items already in inventory, use the search box on the inventory page. You can mark the circles above to search all items or the item types you wish to see such as equipment or medications. Type the item in the search box or use the asterisk for a complete list of all items and click “Search.” You will be given a list of all items that match your search criteria. Choose the item you wish to view. You will see a page like the one on the right. You can change the information such as subtract from the count to reflect items used and disposed of or add to the count when new inventory is added. Click “Update Item” when done to save the changes.



The screenshot shows a web form titled "Change Item". It contains several input fields: "Item Name:" with the text "tongue depressor", "Item Type:" with a dropdown menu showing "Equipment", "Count:" with a numeric input field showing "995", "Value:" with a numeric input field showing "1", and "Notes:" with a text area containing "wooden 6" long". At the bottom right of the form is a blue button labeled "Update Item".

Diagnosis Totals

Diagnosis totals will give you a total number of times a diagnosis was used in a specific date range. Enter the diagnosis you wish to search for in the “Diagnosis” box. Provide the date range you wish to search and click “Generate Stats.” This will give you the total number of times that diagnosis was used within the specific date range.

Report Bug

If you find a problem (a bug) with this program which needs to be addressed by the programmer, you can report it by filling out a bug report. Click on “Report Bug” located near the center of the dashboard and fill out the report. First note the severity level of the bug from level 1 to level 3. The defenitions for the different severity levels are listed on the bug report. Note the location of the bug. That would be the page you were on when the problem occurred. Then type in a description of what the problem is and how it happened. If you are able to capture a screen shot of the problem or error message you received, that could also be helpful. When you are done, click “Add Report.”

Capturing a Screen Shot: To capture a screen shot, press the “Print Scrn” key on your keyboard while the problem or error message is still visible on your screen. This caputres whatever is on the screen of your computer. Then open a program such as Word. Right click

on the blank page and select “Paste” from the menu. You should see a copy of your computer screen saved on the page. Save this file on your computer. Then go to the bug report form. On the bottom of the form you will see “Upload a Snapshot.” Click “Choose File.” Find the file you just created with the screen shot and attach the file.

Capturing an Error Report: When the program detects a bug in itself, it will generate an error report. It will look something like this.

```
*****
Error

Please create a bug report that contains the information below,
and send it to the following email.

aklassen@andrew-klassen.net
*****

Error Location: update_item.php

SQL Query:

INSERT INTO inventory_history (inventory_id, name, type, count, value, notes, timestamp, created_by) SELECT inventory_id, name, type, count, value, notes, timestamp,
created_by FROM inventory WHERE inventory_id='10';

Database Error:

SQLSTATE[23000]: Integrity constraint violation: 1062 Duplicate entry 'tongue depressor' for key 'name_UNIQUE'

Client's Web Browser: Google Chrome
Version: 56.0.2924.87
Platform: windows

User Account ID: 10
Time: 2017-02-14 16:21:31

*****
Back to dashboard
*****
```

To save this, you can capture it in a screen shot as described above, or you can highlight the text of the report, right click the screen and choose “copy.” Then go to the bug report. Right click in the “Description” box and select “paste.” That should copy the text of the error report into the bug report.

Secrets User's Guide

Feature Summary:

The secrets feature allows the user to store and share sensitive information in an optimally secure, reliable, and confidential environment. This includes passwords, login credentials, and PHI (protected health information). Every secret is encrypted internally and is not viewable by those who do not have permissions to view/administer the secret. Even those to have direct access to the database do not have the ability to see the secret's value without the secret owner's permission. This is described in more detail during the technical portion of the documentation.

Also most enterprise environments force employees to work with more login credentials than a human can comfortably handle. Traditional ways for employees to combat this was to either write passwords down or to make all of them the same. This is insecure and creates management/accountability problems when a credential needs to be shared. This is exactly what this portion of the program aims to solve.

Components of a Secret:

Label: A label is short effective description of what the secret value contains. This word or phrase is used by others to identify the secret they are looking for. Labels are required to be unique. (50 characters max)

Examples
bank info
amazon account
Andrew's ssh key
wifi password

Description: This is a more detailed description of the secret. Text in this field is not included in search results and this field is not strictly required, but often a bad idea to leave empty. If you want to provide other users with the ability to request access to the secret, the description is the optimal way to do it. (see the example below) (255 characters max)

Label: Bank Info

Description:

This secret contains login information to the Village of Hope bank account. Please talk to Andrew Klassen if you need access to this secret.

Email: aklassen80@yahoo.com
Date Created: 12/08/2018

Secrets are also not timestamped internally, so you need to add a creation date if the time of creation is essential for the secret. By default the least amount of information possible regarding the secret, is stored inside the database to make secrets confidential.

Remember that a secret's label and description are the only two components to a secret that should be known to all users. Therefore, no sensitive information should be contained in these fields.

Value: This is the portion the secret that is stored in an encrypted form inside the database and should contain sensitive information. Although it was designed for sensitive information, it can be any 5000 characters of text that you want to kept secret. Value information is only viewable to user's who authenticate the secret. (see viewing a secret) Below is an example of what a typical secret's value might contain.

Value:

Username: my_user_name
Password: doNotUseThisPassword123*
Recovery email: myname@gmail.com

Below is how the secret's value above looks to a database administrator. Notice how it has no resemblance to the original value and is completely unreadable.

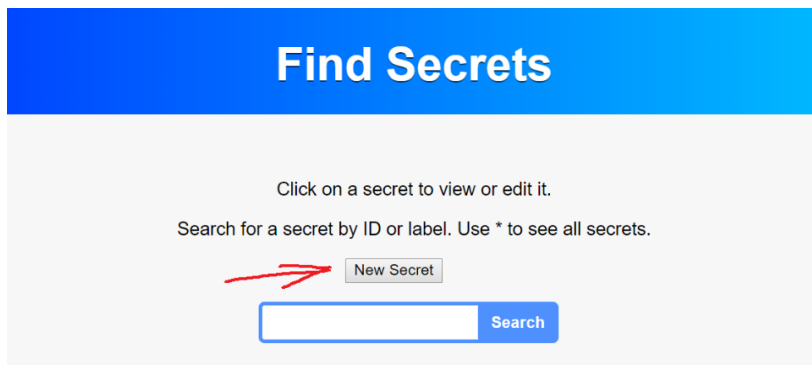
«ÁÝ•ú-Zæí7[zİ£P'[%oÂ/é9'xăéœNÝ:*ÀpEÁÝxÂžojàîrtİ€dVkÔ%üÜV~à-Äocä|đ+çÊÄiÿí•ÉB/
¶'b
•Ěá%!X~P%@cêĐií©Ì

Key: The key is a string value that is using to decrypt and authorize the value of a secret. A key value exists in one of the 3 following forms.

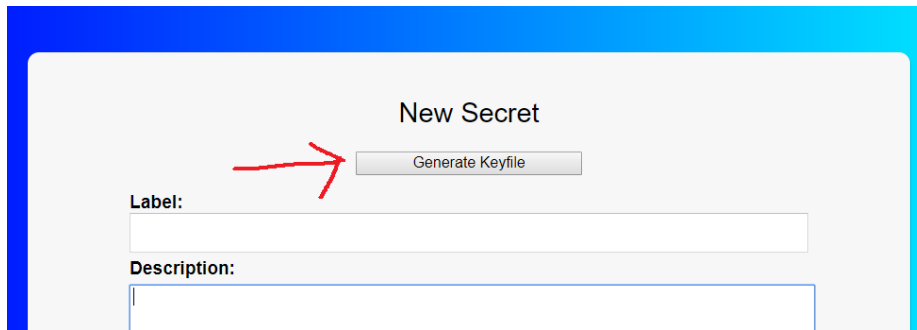
Type	Description	Character Limit	Strength
Password	This is a traditional password. It can be the same as the users initial login, but this is not recommend unless the secret is not sensitive.	50	Ok, depending on what the password is. Encyption with a weak password is effectively useless.
Keyfile	A keyfile is a .txt file that contains a string value for authentication. (see generating a keyfile) Keyfiles generated through the program are 3072 characters long.	3072	Recommended because it allows the user to authenticate using an extremely complex key while not having to remember anything. Keyfiles should not be used on shared computers.
Both	Password and a keyfile. Note: that this is not either or. If both a password and keyfile are provided for a given key, the program will expect both.	3122	Best because user is required to have the key and a password that they only know and have never written.

Generating a Keyfile:

Select “New Secret”



Select “generate keyfile” button on the top of the page

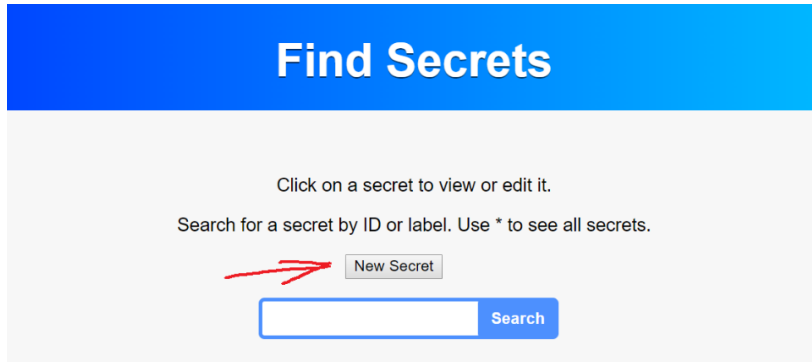


The screenshot shows a web interface titled "New Secret". At the top, there is a button labeled "Generate Keyfile". A red arrow points to this button. Below the button, there are two input fields: "Label:" and "Description:". The "Label:" field is empty, and the "Description:" field is also empty.

This button will generate a keyfile for you and download it to your downloads library folder. The key will be named "key.txt". Rename it, and then move it to a safe location that you will remember. The keyfile will contain a string of 3072 randomly generated characters. These characters are used as the decryption password. The keyfile is also not stored on the server. So do not loose or share this file. The same keyfile can be used to authenticate multiple secrets and doing so is recommended.

Creating a Secret:

Select "New Secret"



The screenshot shows a web interface titled "Find Secrets". Below the title, there is a text prompt: "Click on a secret to view or edit it." followed by "Search for a secret by ID or label. Use * to see all secrets." Below this, there is a button labeled "New Secret" and a search bar with a "Search" button. A red arrow points to the "New Secret" button.

Fill out the form and Click "Add Secret".

New Secret

[Generate Keyfile](#)

Label:

Description:

This is the wifi password.

Value:

Password: P@ssword123

Password:

Key File
[Choose File](#) key.txt
(.txt document 3072 characters max)

[Add Secret](#)

View a Secret: (assuming you already have a key for it)

Find the key you want to view, and then click on it.

Find Secrets

Click on a secret to view or edit it.

Search for a secret by ID or label. Use * to see all secrets.

[New Secret](#)

[Search](#)

Secret ID	Label
45	wifi

Authorize Secret

Label: wifi

Description:

This is the wifi password.

Password:

Key File

Choose File | No file chosen

(.txt document 3072 characters max)

View Secret Auto Authorize Create Key

At the authorize screen, you have two options.

1. If you have a key to the secret that uses the same password as your initial login, you can select “**Auto Authorize**” and view the secret without providing any credentials. **This method of authorization is only recommended for non-sensitive secrets.**
2. Provide your password and/or keyfile and then click “**view secret**”.

(usage of create key is explained later)

The “value” is the portion of the secret you will end up using for authentication on a 3rd party application or service.

Secret

Label: wifi

Description:

This is the wifi password.

Value:

Password: P@ssword123

Warning: Deleting this key will permanently remove it for all users.

Delete Secret

Sharing your Secrets with others/adding additional keys:

There are two types of permissions when dealing with keys. Every key is either one of the following types.

Type	Description
admin	<ul style="list-style-type: none">- can share key with other users- can permanently delete the key if the secret has been compromised- can read the value of the key
Read	<ul style="list-style-type: none">- can read the value of the key

The first key is always an admin key. If you have a secret that is essential to the entire organization. It is recommended that you give an admin key to an Administrator. This will prevent you from being locked out in the event that you either lose your keyfile or forget the key password.

Assuming that you have an admin key, view the secret that you wish to share. Then, navigate towards the bottom of the secret.

Add Key

Password:

Key File

Choose File

No file chosen

(.txt document 3072 characters max)

OR

Username:

Admin

Read-Only

Add Key

At this portion of the screen, you can add additional keys or select the username of the person you wish to share the secret with. It is good practice to only do this when the other person has requested access to the secret from you. You might also notice that there are two special usernames.

andrew

dummy

dummy2

gretchen

Server Admins

(special user)

Everyone

(special user)

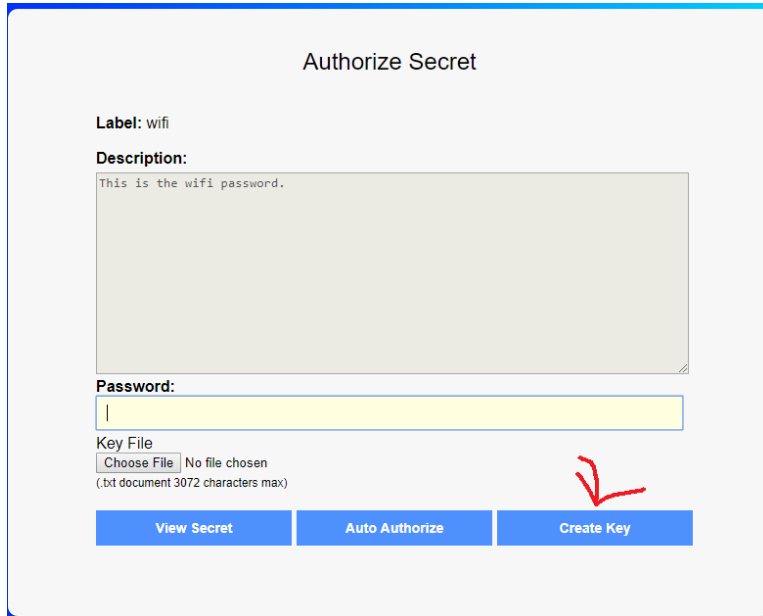
Username	Description
Server Admins	- these are people in the program who have been designated as administrators
Everyone	- this is everyone in the program, use sparingly

Note: Everyone and Server Admins means everyone and admins at that point in time. If a new person joins the organization, you will need to give them access individually.

Once you have either added your password, keyfile, password and keyfile, or selected a username, then select the appropriate permission for the key and click “Add Key”.

Getting Access to a Secret:

Request permission from a user who owns an admin key to the secret. After they have given you permission, navigate to the secret authorization page and select “Create Key”.



Authorize Secret

Label: wifi

Description:

This is the wifi password.

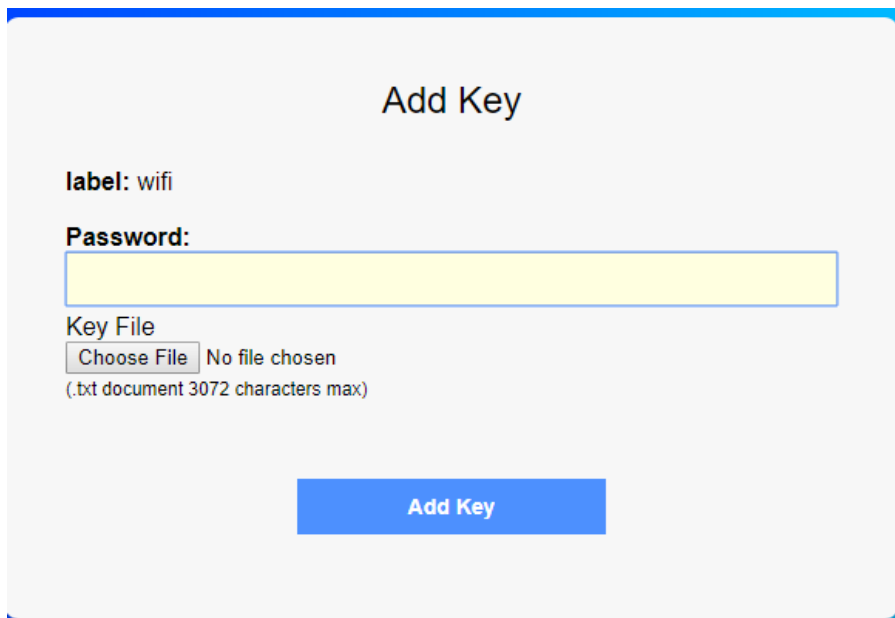
Password:

Key File

Choose File No file chosen
(.txt document 3072 characters max)

View Secret Auto Authorize Create Key

Then add your password, keyfile, or both and click “Add Key”.



Add Key

label: wifi

Password:

Key File

Choose File No file chosen
(.txt document 3072 characters max)

Add Key

Deleting and Changing a Secret:

If a secret gets compromised, an admin key owner can delete the secret. This will permanently delete the secret for everyone.

This feature is also helpful if you accidentally gave permission to someone you were not suppose to give permission to. The permissions and content of a secret can never be changed due to their confidential and anonymous nature. So the way you would handle this case is by performing the following.

1. Create a new secret the same as the old one (although the label must be unique).
2. Delete the old secret.
3. Add the correct permissions and users to the new secret.

To delete the secret, view the secret using an admin key and click the red “Delete Key” button.

Warning: Deleting this key will permanently remove it for all users.

Delete Secret

Secrets Technical Documentation

Common Methods for Sharing Secrets and Attributes that Make Them Less than Optimal:

There are no formal rules or laws that define what is and is not secure for every organization. This is mostly due to every organization being different, and the fact that security is subjective. It is the organization's duty to determine what is and isn't secure and then to write and enforce policies that they believe will allow them to operate in the most secure and efficient manner.

It is, however, possible to define security mechanisms that are less than optimal. A security mechanism that is less than optimal, is one that is less secure compared to a solution that has no additional impact on the user.

An example of this was the way the program initially stored the user's passwords. In the earlier versions of the program, user's passwords were stored into the database in clear text. Security was not a big concern at the time and hashing does add complexity to the administration side of the program. But as I learned more about hashing and became good at working with it, I decided to hash the passwords in order to improve security. It was more work on my part, but in doing so I improved security without changing the user's login experience. In this example, storing the user's passwords in clear text is an authentication method that is sub-optimally secure.

I have no desire to try to argue that the program is better than common methods of sharing secrets. But I will happily point out that the program makes common secret sharing methods less than optimal in regards to security. Below are security weaknesses to common secret sharing methods.

1. Secrets over Email

- a. email is used for many online services, including account creation, recovery, and public forms, by gaining access to a user's email, you can pretend you forgot your password to an account created by the email and gain access to the 3rd party account
- b. they are frequently targeted by attackers due to the amounts of information that they expose
- c. emails are almost always publicly accessible
- d. emails are not always encrypted during transit, due to ssl/tls requiring additional setup and often a monthly cost
- e. emails are never encrypted on the server's live filesystem, they are stored as text files, usually under /var/spool/mail, and are available to an attacker in clear text if the server gets compromised

- f. using email means your are involving a 3rd party in protecting your secrets in addition to your own organization, historically every major email service provider has had some form of security breach, HIPPA reporting is still required if this type of breach happens
- 2. Secrets written down on paper
 - a. paper can easily be stolen
 - b. paper can easily be losed or destroyed
 - c. harder to backup and control from a central point
- 3. Secrets on word document or notepad, stored on client machine
 - a. not centrally backup or controlled
 - b. bad if on shared computer
 - c. If the machine is unencrypted, an attacker can easily snatch the file using a live usb or by removing the hard drive and plugging it into his computer
 - d. hard to safely share
- 4. Secrets over verbal communication
 - a. hard to remember, or end up using the same passwords so that everyone can remember
 - b. administration nightmare
 - c. easier for someone to eavesdrop
- 5. Secrets over internal chat service
 - a. chat servers typically only store conversation over a specified amount of time
 - b. chat servers are often unencrypted
 - c. they often show the pervious conversation every time a user is selected, this makes shoulder surfing easier
 - d. no searchable way to find secrets
- 6. Secrets saved in browser
 - a. easy for attacker to snatch if physically near client machine
 - b. all passwords are stored in clear text
 - c. no sharing
 - d. no centralized management

What makes the Secrets Feature Good:

1. All secret values are encrypted internally using 256 bit AES (CBC). This is implemented at the field level, meaning that the values are encrypted to a user working on the database. Useful if a volunteer or contractor were to do work on the database. Also prevents attacker from seeing secret values if the database was compromised or if a database dump/backup was stolen.
2. Because the application is web based, secret values are only on the server, which is in a physically secure area.
3. It is centrally managed.
4. Multiple user's can access and provide access to the same secret without sharing credentials.
5. Secrets are essentially anonymous. They are not timestamped, there is never any record in the database as to who owns or has access to a secret, and it is not possible to tell how many users have access to a particular secret. (This prevents high access individuals from being targeted.) While at the same time, the server and owner of the secret always know and have complete control over the secret.
6. Session variables involving the secrets are wiped as soon as possible. This prevents secrets from being exposed in the event that the user's php session is hijacked.
7. When keyfiles are generated they are done solely in JavaScript, meaning that the keyfile never exists on the server.
8. When keyfiles are uploaded to the server, they are stored in php's default temp directory and then wiped immediately after the php script finishes executing.
9. Since secret values can be encrypted with keyfiles, this makes the encrypted secret values effectively immune to dictionary and brute force attacks.

Hashing vs Encryption:

Hashing: The purpose of hashing is to give the database the ability to verify something without storing it. Storing user passwords in a hashed form is a common way to store passwords because you are often only wanting to verify the the users password upon login.

Hashes are also often used to verify the integrity of a file. If you take a file and run it through a hash function and then change one character of the file and then run it back though the same hash function and compare the hashes, the hashes will look completely different. The hashes created from these files would also be irreversible, meaning that you would never be able to re-create the original file from its hash. Additionally the hashes sizes are always fixed in length.

So a users password regardless of the complexity will always have the same size hash. In the program, the users passwords are hashed with salted argon2. They are always 96 characters in size. Below is an example of one.

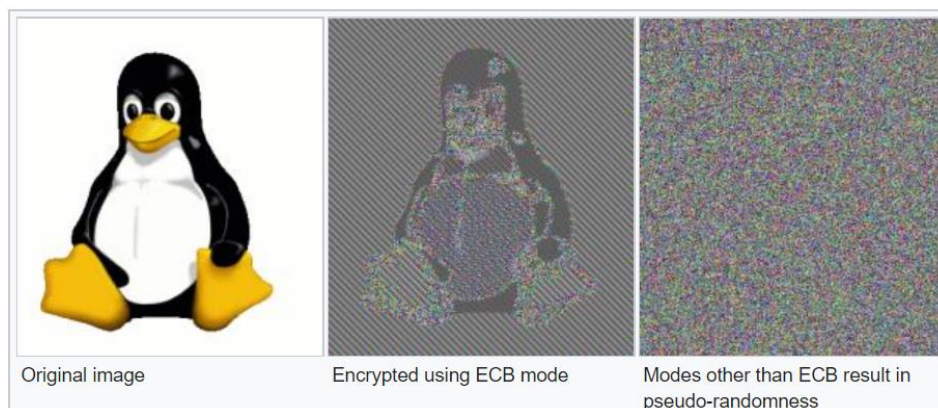
```
$argon2id$v=19$m=1024,t=2,p=2$SVoyaEs5UDB6cDhEWDdRMA$gnbtpKRyoarlj3Mlc1kdDwcn  
cUX5mb5xjmm/PIWBr30
```

Encryption: Stores the file in a cryptographic way that can only be read by a user who has the key. The program uses 256 bit AES when working with secret values. Keep in mind that this is not MySQL's standard level of encryption. MySQL uses 128 bit AES by default when using the AES_ENCRYPT() and AES_DECRYPT() functions. Following parameter was added the program's MySQL config in order to change the default encryption mode.

```
block_encryption_mode=aes-256-cbc
```

AES with a 128 bit key size is deemed to be uncrackable by todays standards, assuming its used correctly. However, 256 bit key sizes are often required for compliance reasons. The larger the key size, the bigger the performance hit.

Its also worth noting that that MySQL's default AES mode, ECB (Electronic Codebook) is potentially unsafe in cases where a lot of repetitive data is encrypted. For an example, take a look at the images below.



Since MySQL's default AES mode encrypts every block of ciphertext the same, the background as well as the penguin's skin end up having the same color pattern after the image has been encrypted. This creates a contrast that allows a human to easily tell what content the encrypted image contains. This is why the choice to use CBC was made. The advantage to ECB is that it is simpler and does not need an initialization vector in order to operate.

The initialization vector is essentially a value used to randomize the encryption, and because of this it is generated uniquely per every encrypted_value and stored as a field in the same record. The encrypted value's initialization vector is needed every time the value is decrypted. So it is important that the initialization vectors are not altered. However, it is not necessary for the initialization vector to be kept secret.

key_hash in the secret tables is the argon2 hashed form of the decrypted value. It's used to verify if the user has decrypted the value correctly.

Secrets Table Structure:

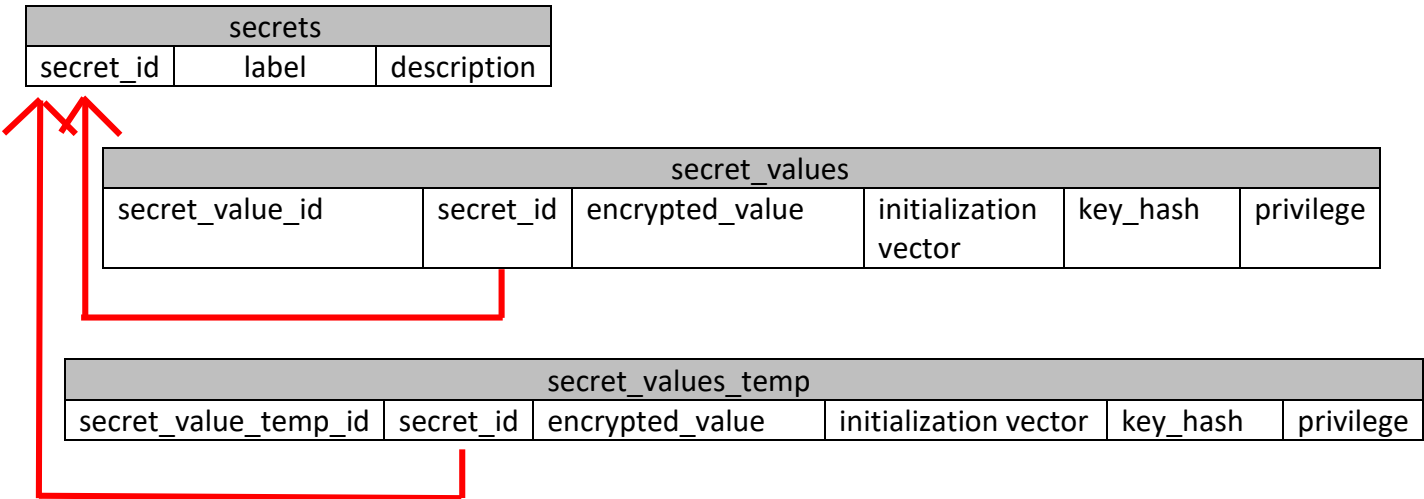


Table name	Description
secrets	Used to store public knowledge of the secret.
secrets_values	Used to store encrypted values corresponding to the secrets stored in the secrets table. With encryption you almost always decrypt using only one key. The program gets around this limitation by storing another copy of the value that is encrypted with a different key. So if 4 users each have one key to secret, the database will contain 4 encrypted copies of the secret's value.
secrets_values_temp	This table hold the encrypted values used during "one time values". One time values exist when a user is given the ability to create a key, but has not done it. This is discussed more in the secret sharing and creation

	portion of the documentation. This table is held in memory only. Meaning, that it is wiped every time the server is shutdown. This prevents this table from being exploited in the event that the server is stolen.
--	---

Secret Sharing and Creation Process:

1. **New secret is created.** A record containing the secret's public information is inserted into the secrets table. The initial key's data is inserted into the secret_values table. This key is given the admin privilege.
2. **The owner of the secret decides to view his secret.** The program will take the key provided by the user and attempt to decrypt every admin encrypted value pertaining to the secret. If one is decrypted, that key with the admin privilege is used. If no admin keys are found, it will attempt to decrypt the read keys. If there is a read key, than that key is used and the read permission is applied.
3. **Secret owner decides to share his key with other user.** The other user's password hash + client id, are used to encrypt the secrets value. Then, the encrypted value is inserted to the secret_values_temp table. If a one time value already exist for the other user, then it is deleted and replaced with a new one.
4. **Other user creates a key.** Program will attempt to decrypt all values in the secret_values_temp table. As soon as it finds one, the value is then re-encrypted using the key that the other user has just provided, and then this is inserted into the secret keys as a permanent key. After this, the one time key is deleted.

Deleting a Secret as an Admin:

A database admin can always delete a key regardless of whether or not he access to it by using the following queries.

```
USE hope_matters;
```

```
DELETE FROM secrets WHERE secret_id = '<secret_id>';
```

```
DELETE FROM secret_values WHERE secret_id = '<secret_id>';
```

```
DELETE FROM secret_values_temp WHERE secret_id = '<secret_id>';
```

Addition Keys in Keyfile:

The secrets portion of the program does not scale well due to its confidentiality. In computer science terms, it scales with a worst case complexity of $O(n)$. Meaning that if there are 4 keys existing for a secret, it would take the program up to 4 times longer to authenticate the user's key. This would be a problem in an organization that has 300 employees and the secret is shared with everyone. It also would mean adding a individual to secrets every time there is a new hire.

The solution to this is to have a unique key that is shared amongst the group of users. However, if this is the case, the user would have to keep track of multiple keys. So I created the ability to have multiple keys in a single keyfile. Below is an example of a keyfile with multiple keys. (key lengths were shortened to save space)

```
imR4_hPcZ^mzl*W)_&oHlc}@DG@H}kYE6CN}TCnuuDhiZ@^:$GF{sL_KZPbiG8tKHUwvE7O4X6q
YeE^oTkU}4kA32rb*W*5ya2MA5PZb^bedO@77bZO$

---additional_key---

v8j8gFJTg&j{ica0SrTGMryw{phq^!3N5rI8vN76QOtYIF79sS!7sXv7B!NocJa$So*gp{{y5tX2vYGe%%
1@Pf8djcpzwWddJFIF@YdM:deEeyM7isKRoZVkiV#

---additional_key---

_@AEUb2T}}3sam0ANN}9DKTPXw2@BWkHCKOL%$K6IJ7yueU9fOQjT2daerjnROLED}R$33DSaQ
v3Khn2RXCoBR4KHJz^(1*(XGlcYl{v#
```

The key at the top is the master key. This key should always be unique to the user. The master key is the only key that would be used if the user was adding a key to a secret using this keyfile. However, if the user was authenticating using this keyfile, the program would attempt to authenticate starting from the first key in the file, going down to the bottom. For this reason, its recommended to place least privilege/less specific keys towards the end of the file. There is no limit to how many additional keys a keyfile can have. However, a keyfile must only have one master key.

Custom Forms

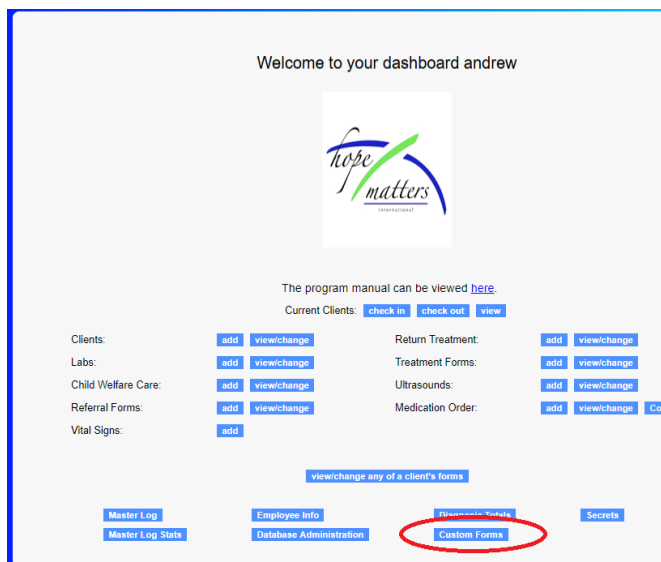
Feature Summary:

The custom forms feature allows the user to create forms in a quick and efficient way without the need to edit the program's source code or database structure. This is implemented through json, which is a universal file format for storing structured data in a minimalistic and readable way.

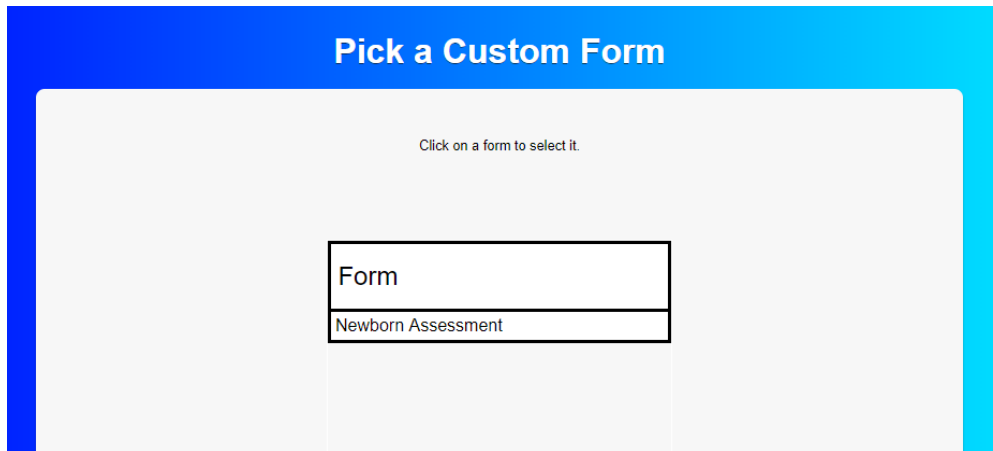
All of the features except the ability to use custom forms, are only available to privileged accounts. If you need access, contact your administrator. If you are the administrator, access can be given by creating a database account for the user that is the same as their program account (this includes their password). This database account needs full access to the custom forms database.

Using Custom forms:

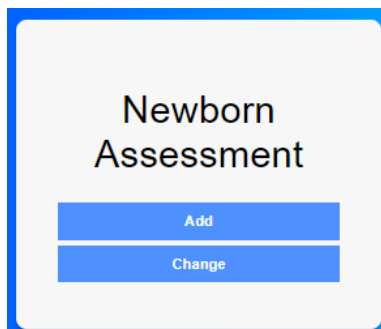
Select the custom forms button on the dashboard.



Select the form you want to use from the table.



You will then be redirected to the form specific dashboard. At this dashboard the form will work the same as all the others.



Writing a custom form:

Custom forms exist in the form of JSON files. You can create a JSON file through any common text editor. When, you are done writing your JSON file, you save it with the .json extension. It's often beneficial for a person to see an example of a finished json form before the components of a JSON form are explained. Take a look at the following examples.

Json Example:

```
{
  "form_name": "Cool New Form",
  "meta": {
    "client_linked": true
  },
  "body": {
    "This is my static text." : "text",
    "thing1" : "textbox",
    "thing2" : "textbox",
    "Notes" : "textarea"
  }
}
```

```

{
  "form_name": "Add Client",
  "meta": {
    "client_linked": false
  },
  "body": {
    "Please fill out the Form" : "text",
    "First name" : "textbox",
    "Last name" : "textbox",
    "Date of birth" : "textbox",
    "Guardian name" : "textbox",
    "Phone number" : "textbox",
    "Occupation" : "textbox",
    "Education" : "textbox",
    "Location" : "textbox",
    "Emergency contact" : "textbox",
    "Allergies" : "textbox",
    "Hiv status" : {
      "type" : "radio_button_group",
      "buttons" : [
        "+",
        "-",
        "unknown"
      ]
    },
    "Alcohol use" : {
      "type" : "radio_button_group",
      "buttons" : [
        "never",
        "sometimes",
        "often",
        "unknown"
      ]
    },
    "Gender" : {
      "type" : "radio_button_group",
      "buttons" : [
        "male",
        "female"
      ]
    },
    "Significant medical history" : "textarea",
    "Regular medications or herbs" : "textarea"
  }
}

```

Json files are object oriented. In fact, JSON is short for JavaScript Object Notation. Curly braces are used to define the composition of an object. The object its self is an unordered set of name/value pairs.

```

{
  "form_name" : "myform"
}

```

The json above would be described as an object with a value named "form name", whose value is "myform". The double quotes are important and necessary in this example because both the

name and its value are strings (letter values). Names and their values are separated with a colon. The name is always on the left, and is value to the right. A json file will always store a single object, and because of this it will start and end with a set of curly braces.

Take a look at the slightly more complex example below.

```
{
  "form_name" : "myform",
  "meta" : {
    "client_linked" : true
  }
}
```

In this case, the object has 2 values. The first one has a name "form_name" and its value is "myform". The second value is named "meta" and its value is an object. It's object has a value named "client_linked" whos value is true. Objects who are values to other object are refered to as nested objects. There is no limit to how many levels of object nesting you can have. There are also 2 important differences from this example compared to the last.

1. Did you notice the comma after "myform"? Any time you have a list of name value key pairs you need to have a comma after every key pair except the last one. This lets the JSON interpreter know that another value is to be expected. This is no different than the way that the comma is used throughout the english language.
2. "client_linked"'s value does not have quotes around it. This is because its value is a boolean. It's not the literal phrase "true". This will be the only boolean value you will have to use through custom forms.

Also note that there is no requirements as to how the JSON is indented or spaced. However, this documentation always presents them the same way in order to make the JSON more readable and consistent. But below are different examples of the same JSON, all of which are completely valid.

```
{
  "form_name" : "myform",
  "meta" : {
    "client_linked" : true
  }
}

{
  "form_name ": "myform ",
  "meta": {"client_linked": true}
}

{"form_name ":"myform ","meta":{"client_linked":true}}
```

With custom forms, your JSON will always consist of the following 3 items.

1. A form name. This is what the form will be displayed as to the user.
2. A meta object. The word meta means “in reference to itself”. The meta object consists of name value pairs that pertain to the form, but will not be a visible field on the form its self. As of now, the only meta value required is “client_linked”, which determines as to whether or not a client is selected before filling out the form. In most cases you will want clients linked to the form. However, there are cases were you have forms that are not related to the clients. An example of this would be a form for checking out equipment.
3. A body object. The body of the form is were you list the fields that the form should contain. It is a requirement that every value name is unique. The value name should always be what you want the field to be labeled as. The value is what the type of field should be.

Below is the example shown earlier, comparing the JSON to what the actually form looks like.

```
{
  "form_name": "Cool New Form",
  "meta": {
    "client_linked": true
  },
  "body": {
    "This is my static text." : "text",
    "thing1" : "textbox",
    "thing2" : "textbox",
    "Notes" : "textarea"
  }
}
```

The screenshot displays a web form with a light gray background and a blue border. At the top, there is a section titled "Client's General Info" containing a grid of client details: Client ID: 1, First Name: Andrew, Today's Date: 01/13/2019, Sex: male, Last Name: Klassen, Age: 23, Residence: location, and Occupation: occupaton. Below this section is the main form area titled "Cool New Form". It contains a static text field with the value "This is my static text.", followed by two text input fields labeled "Thing1" and "Thing2". Below these is a text area labeled "Notes". At the bottom center of the form is a blue "Submit" button.

Notice how the order of the fields in the JSON correspond to the order in the form.

There are some elements of a custom form that are slightly more complex. These are array fields. I added one to the previous example.

```
{
  "form_name": "Cool New Form",
  "meta": {
    "client_linked": true
  },
  "body": {
    "This is my static text.": "text",
    "thing1": "textbox",
    "thing2": "textbox",
    "Notes": "textarea",
    "client gender": {
      "type": "radio_button_group",
      "buttons": [
        "Male",
        "Female"
      ]
    }
  }
}
```

“client gender” is a set of radio buttons and looks like the following.

The screenshot shows a web form with two main sections. The top section, titled "Client's General Info", contains a grid of labels and values: Client ID: 1, First Name: Andrew, Today's Date: 01/13/2019, Sex: male, Last Name: Klassen, Age: 23, Residence: location, and Occupation: occupation. The bottom section, titled "Cool New Form", contains a static text label "This is my static text.", followed by two text input fields labeled "Thing1:" and "Thing2:". Below these is a large text area labeled "Notes:". At the bottom of this section are two radio buttons labeled "Male" and "Female", with "Female" selected. A blue "Submit" button is located at the very bottom of the form.

The value “buttons” is an array, and because of this the name/value concept does not need to apply to the names listed inside of its brackets.

All potential form field types are listed below.

Form Field Types:

Note: text in green is not literal

Field Name	Json	Main Table Column	Description
text	"<static_text>" : "text"	none	This form field is used to display text on the form that will never change. Often used to describe an aspect of the form.
bold_text	"<static_text>" : "bold_text"	none	Same as text except it is bold.
date	"<date>" : "date"	date	This field is for storing dates. In the form it generates a textbox with a drop down calendar for the user to use. The default value for this field is always the current date.
integer	"<integer>" : "<integer>"	int(11)	Creates a textbox that is restricted to integer values. The default is always 0.
textbox	"<textbox_label>" : "textbox"	varchar(50)	Should be used to store quick, searchable values in the database.
textarea	"<textarea_label>" : "textarea"	tinytext	Allows the user to provide up to 255 characters of text.
textbox_array	<pre>"<array_name>" : { "type" : "textbox_array", "labels" : ["<textbox_label>", "<textbox_label>", ...] }</pre>	varchar(25) (per textbox)	Creates an array of textboxes at the same level of height. In the database, textboxes that are an element of the array are stored using the following naming convention. <textbox_label_array_name>
textarea_large	"<testarea_label>" : "textarea_large"	text	Allows the user to provide up to 5000 characters of text. Both the textarea and textarea_large fields should be avoided if possible because they are the most costly fields on a form.
checkbox	"<checkbox_label>" : "checkbox"	enum('yes', 'no')	The checkbox will submit a yes if it is checked, no if it is not checked.
radio_button_group	<pre>"<button_group_label>" : { "type" : "radio_button_group", "buttons" : ["<first_button_label>", "<second_button_label>", ...] }</pre>	enum('<button_label1>', '<button_label2>', ...)	In a radio button group the last button is always checked by default. If you are wanting a yes and no pair of radio buttons, you should use a checkbox instead.
image_small	"<image>" : "image_small"	none	Creates a static image on the form that is 100px wide and 100px in height.
image_medium	"<image>" : "image_medium"	none	Creates a static image on the form that is 300px wide and 300px in height.
image_large	"<image>" : "image_large"	none	Creates a static image on the

			form that is 800px wide and 800px in height.
image_large-short	"<image>" : "image_large-short"	none	Creates a static image on the form that is 800px wide and 400px in height.
image-upload_small	"<image>" : "image-upload_small"	varchar(1000)	Allows the user to upload an image. A template image 100px wide and 100px in height is shown to the user on the "add form" page, like the static image.
image-upload_medium	"<image>" : "image-upload_medium"	varchar(1000)	Allows the user to upload an image. A template image 300px wide and 300px in height is shown to the user on the "add form" page, like the static image.
image-upload_large	"<image>" : "image-upload_large"	varchar(1000)	Allows the user to upload an image. A template image 800px wide and 800px in height is shown to the user on the "add form" page, like the static image.
image-upload_large-short	"<image>" : "image-upload_large-short"	varchar(1000)	Allows the user to upload an image. A template image 800px wide and 400px in height is shown to the user on the "add form" page, like the static image.

At the current time of writing, the custom form feature does not have anything in the way of error detection. This can be frustrating with long forms because missing a quote or comma on a JSON file that is 300 lines is hard to catch. But because JSON is a common and universal format, there are several other free tools that can provide you with error detection and syntax highlighting. **It is recommended that you use one to save your self from unnessassary pain and frustration.**

On Linux, Gedit has native support for JSON. In a windows enviroment, Notepad++ supports it. Below are some good web based tools.

<https://jsonformatter.org/json-editor>

<https://jsonlint.com/>

Getting the JSON from an existing form:

The user has the ability to get the JSON of any existing custom form. This is useful if you want to be able to back up your form or share it with people a running different copy of the same program. It also comes in handy when need to duplicate parts of an existing form.

To download a form's JSON, select the "Download json" button on the custom forms main page. Then, select the form you wish to download.

Form
Cool New Form
Newborn Assessment

Add form
Choose File No file chosen
Upload json

Download json

Delete Form

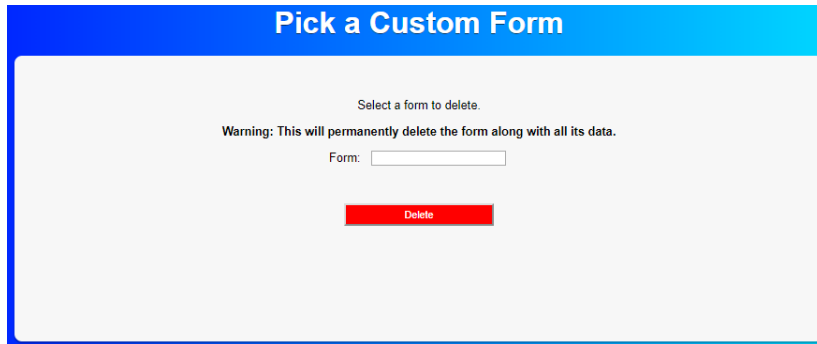
Uploading a JSON/form:

You upload forms by using the "Add form" button in the feature above. However, there are some technical details worth noting

1. The JSON file you upload needs the .json extension.
2. You can have multiple JSON files concatenated into a single JSON file if you want to upload multiple forms at once.
3. If your JSON contains any image fields, you need to take the JSON file and images that you refer to and place them into a zip folder. Then, upload the zipped folder. Make sure this zip folder has the .zip extension.
4. If you have multiple JSON files you can upload all of them at once by using a zip folder.

Deleting a form:

Deleting a form works by using the “delete form” button on the same page above. Make sure you do not ignore the warning on the page. If accidental deletion does occur, you should be able to restore from your offline backups.



The screenshot shows a dialog box titled "Pick a Custom Form" with a blue header. The main content area is light gray and contains the following text and elements:

- Text: "Select a form to delete."
- Text: "Warning: This will permanently delete the form along with all its data."
- Text: "Form:" followed by a text input field.
- A red button labeled "Delete".