User Guide

Usage

Manager Website

The website is used to manage the users of the app and edit the content of the app.

Adding a new user

If somebody new seeks help from Changing Lives, they can be assigned to a member of staff. This staff member then simply needs to log in to the website using their details and click on the USERS tab in the blue bar near the top. This opens the user management display. From here the new user button can be clicked to easily add a new user with no details required apart from a nickname. This could be the users actual name or otherwise, it is just so that members of staff can tell who's who as all usernames are anonymous. This nickname is never shown in the app. Then once OK is clicked, the new user details are shown which can then be given to the user who is now a part of the Changing Lives app.

Editing and removing a user

While usernames cannot be changed (this is to keep them anonymous), it is easy to change a user's nickname from here by pressing the edit button next to their name. If someone has forgotten their password. The procedurce is to contanct their staff member at Changing Lives. The staff member can then click 'edit' next to he user's name and reset the password. A temporary password is provided which can then be given to the user. A user can also be removed by the remove button beside it, for instance if they stop requiring Changing Lives' services.

As the list of users grows, the search bar at the top allows you to more easily find the correct user by searching both nickname and username.

Changing the app content

The CONTENT tab allows staff members to change what information is shown to users on the app. A list of the app sections is displayed, and they can be edited and removed similarly to the users. They can also be reordered by the 'move up/down' buttons. This allows staff to prioritise certain sections which are more important.

Creating a new section is a easy as pressing the new section button and filling in the details. All sections have a title, and then there is room for some specific text about the topic which will be useful to the users of the app. We recommend possibly using this to have a daily tip for some sections as it is easy to change later. Below this is where files can be added. The website only accepts PDFs (see the appendix on converting word documents to PDF). Click browse to open up your file browser, allowing you to select one or more files. Then click 'add' to add them. They should appear below and they can be renamed to a useful name to display to users. Then click OK to submit the section.

Reviewing the logs

The LOGS tab shows a list of all actions which are taken on the website and app. This can be used for security in the event of someone unwanted getting access to a staff account. It is easy to see exactly what each account did. A more important feature is the restore feature. If a member of staff accidentally deletes something (e.g. a section or a user), then all they have to do to bring it back is to find the action in the logs and click on it. Then some more information will appear. Click restore to have the item brought back.

Setting up the environment for local testing

This section is a more technical part which details how to install and run the server locally. The server should be deployed on a cloud service if it is being used by real users, so this is intended for testing the server prior to deployment.

Default login details

Website login details `username: clstaff, password: admin20`

Database login details `username: ?, password: ?`

Server

First you will need to install Nodejs, which is what the server is built in. To do this go to https://nodejs.org/en/, download Nodejs and then install the file. To start the server, open up a terminal in the location of the server files (also the location of this user guide). Then run the following commands:

```
npm install
npm start
```

You should see a message indicating that the server is running.

Website

Once the server is running, open up an internet browser (Google Chrome, Microsoft Edge or Firefox - Internet Explorer is NOT supported), and type localhost:3000 in the address bar. This will now display the Changing Lives App Manager website.

Temporary server for you to test the project

We have hosted a temporary server on Digital Ocean so you can test our product before deciding how you would like to host it. This can be reached at the address

`134.122.108.72:3000/`

Appendix

Converting a word document to a PDF

For security and ease of use reasons, the Changing Lives App Manager website only allows file uploads of the type PDF. As most of the Changing Lives documents are Word documents, this will guide you through converting them to PDFs.

1. Open the document in Microsoft Word
2. Click File, then Export
3. Click Create PDF document

This guide was written for the Office 365 version of Word, but should follow similarily for other versions.

System Maintenance

Overview

This section will give some essential information about what to do after the system has been delivered. You (the client) will need to set up your own server and database. The system has been developed using the Google Cloud Platform MySQL database and a local server (only accessible by devices connected to the same network). However, this is only for testing purposes. If the system is put into use, you will need to pay for the costs of the server and database service(s). This part of the user manual will also give some information about predicting maintenance costs.

Database

Version

The testing database uses **MySQL 5.7** and is hosted by Google Cloud Platform. When setting up your own database, please make sure the database version of MySQL is below 8.0, as Node.js cannot authenticate to MySQL 8.0 and above due to the default authentication plugin in those versions. See https://stackoverflow.com/questions/50373427/node-js-cant-authenticate-to-mysql-8-0 for more detailed information.

Connecting to the Database

The `.env` File

The system requires an environment file, called *.env*, on the top level. This file contains the credentials needed to connect to the MySQL database, along with the secret values used for creating authentication tokens. The file contains the following keys, each with a corresponding value: `MYSQL_HOST`, `MYSQL_USER`, `MYSQL_PORT`, `MYSQL_PASSWORD`, `MYSQL_DATABASE`, `USER_KEY` and `STAFF_KEY`. All values are strings except `MYSQL_PORT`.

Connection details for the Test Database

```
***REMOVED FROM GIT-HOSTED USER MANUAL***
```

Please note that this database is for testing purposes only, and *should not* be used for a deployed/live version of the system.

MySQL Workbench

MySQL workbench is a useful tool for setting up the MySQL database. Download MySQL workbench from https://dev.mysql.com/downloads/workbench/ by selecting your operating system and downloading the corresponding file.

Creating a Connection

After opening MySQL Workbench, you will see a plus button after **MySQL Connections** in the home page. Click it and a window will show up. First, set up a **connection name** as any relevant name (e.g. "Group 22 Test Database"). Then, fill the **Hostname**, **Port**, **Username** and **Password** from the connection details above. Then click **OK**. The program will jump back to the home page and you will find a new connection instance has been created. Click this and you will be connected to the database editor.

Database Structure

The database consists of six separate tables, which are: **users**, **sections**, **files**, **parent_comments**, **child_comments** and **logs**.

Table 1: `users`

Column	Data Type	Primary Key	Automatically Increment
user_id	int(11)	YES	YES
real_name	char(32)	NO	NO
username	char(32)	NO	NO
password	char(128)	NO	NO
password_salt	char(32)	NO	NO
is_admin	bit(1)	NO	NO
force_reset	bit(1)	NO	NO

Note: `real_name` is the internal name for the user's *nickname*, which is not required to be the same as their real name to preserve anonymity.

Table 2: `sections`

Column	Data Type	Primary Key	Automatically Increment
section_id	int(11)	YES	YES
user_id	int(11)	NO	NO
article_text	text	NO	NO
section_name	char(32)	NO	NO
position	int(11)	NO	NO

Table 3: `files`

Column	Data Type	Primary Key	Automatically Increment
file_id	int(11)	YES	YES
file_name	char(32)	NO	NO
file_link	text	NO	NO
section_id	int(11)	NO	NO
user_id	int(11)	NO	NO

Table 4: `parent_comments`

Column	Data Type	Primary Key	Automatically Increment
parent_id	int(11)	YES	YES
user_id	int(11)	NO	NO
parent_comment	text	NO	NO
parent_title	char(32)	NO	NO

Table 5: `child_comments`

Column	Data Type	Primary Key	Automatically Increment
child_id	int(11)	YES	YES
user_id	int(11)	NO	NO
parent_id	int(11)	NO	NO
child_comment	text	NO	NO

Table 6: `logs`

Column	Data Type	Primary Key	Automatically Increment
logId	int(11)	YES	YES
userId	int(11)	NO	NO
dateTime	datetime	NO	NO
action	char(32)	NO	NO
entity	char(32)	NO	NO

Column	Data Type	Primary Key	Automatically Increment
newData	json	NO	NO
oldData	json	NO	NO

Creating Your Own Database

You will need to select a platform (such as Google Cloud Platform or Microsoft Azure) to create a MySQL instance, and get the connection information. Then, use MySQL workbench to connect to the database. Use the provided resources, including the tables above to create a new database (see the tutorial for using the MySQL model to create a MySQL database at https://www.youtube.com/watch?v=K6w0bZjl_Lw.)After creating the new database, update the connection details in the .env file.

Server

Our system has been developed and tested using a local server. To use this system, the client needs to create their own server (if they do not have a server) support it. The server running the Changing Lives website may be fit for this purpose.

Predicting Maintenance Costs

The majority of the predicted maintenance costs will be the costs of the MySQL database and the server.

MySQL Database Costs

The cost of a suitable MySQL database in Google Cloud Platform is $30-$100 (£24-£80) per month. However, the cost of MySQL database in Microsoft Azure is at least $130 (£105) per month.

Server Costs

DigitalOcean offers many different price plans for servers, with the cheapest option at £5 per month and the recommended option at £40 per month.

Future Development

During development, our code has been refactored into a more readable and learnable way to help with future additions and maintenance by the client. The API and application can be extended with new routes or features by using a copy of an existing route as a base to add the new functionality to, and updating the routes file to include it. This may require a corresponding extension to the database, which can be done by creating new columns or tables using MySQL workbench.

Changing Lives API Documentation

Endpoints

User Endpoints

Create

Route	`/api/users/create`
Type	POST
Body	`{'realName': string, 'isAdmin': bit}`

Route	`/api/users/create`
Response	• No errors: Status 200 and `{'password': string, 'username': string}` • No permission: Status 403 • Errors: Status 500

Remove

Route	`/api/users/remove`
Type	POST
Body	`{'userId': integer, 'password': string}`
Response	• No errors: Status 200 • No permission: Status 403 • Could not delete: Status 400 • Errors: Status 500

Change

Route	`/api/users/change`
Type	POST
Body	`{'realName': string, 'isAdmin': bit}`
Response	• No errors: Status 200 • No permission: Status 403 • Could not change: Status 400 • Errors: Status 500

Edit

Route	`/api/users/edit`
Type	POST
Body	`{'nickname': string, 'userId': bit}`
Response	• No errors: Status 200 • No permission: Status 403 • Errors: Status 500

Reset

Route	`/api/users/reset`
Type	POST
Body	`{'userId': bit}`

Route	`/api/users/reset`

Response

- No errors: Status 200
- No permission: Status 403
- Errors: Status 500

Login

Route	`/api/users/login`
Type	POST
Body	`{'userName': string, 'userPassword': string}`

Response

- No errors: Status 200
- No permission: Status 403
- Input validation failed: Status 400
- Incorrect username/password: Status 401
- Errors: Status 500

List

Route	`/api/users/list/?search=string&uname=string&rname=string`
Type	GET

Response

- No errors: Status 200 and array of JSON user objects
- No permission: Status 403
- Errors: Status 500

Section Endpoints

Create

Route	`/api/sections/create`
Type	POST
Body	`{'sectionName': string, 'sectionText': string, 'files': array}`

Response

- No errors: Status 200
- No permission: Status 403
- Errors: Status 500

Edit

Route	`/api/sections/edit`
Type	POST
Body	`{'sectionName': string, 'sectionText': string, 'sectionId': integer, 'files': array, 'fileRemove': jsonBody`

Route	`/api/sections/edit`
Response	<ul><li>No errors: Status 200</li><li>No permission: Status 403</li><li>Errors: Status 500</li></ul>

Move

Route	`/api/sections/move`
Type	POST
Body	`{'sectionId': integer, 'moveUp': string}`
Response	<ul><li>No errors: Status 200</li><li>No permission: Status 403</li><li>No section provided: Status 400</li><li>Errors: Status 500</li></ul>

Remove

Route	`/api/sections/remove`
Type	POST
Body	`{'sectionId': integer}`
Response	<ul><li>No errors: Status 200</li><li>No permission: Status 403</li><li>No section provided: Status 400</li><li>Errors: Status 500</li></ul>

Restore

Route	`/api/sections/restore`
Type	POST
Body	`{'sectionName': string, 'sectionText': string, 'sectionFiles': jsonBody}`
Response	<ul><li>No errors: Status 200</li><li>No permission: Status 403</li><li>Errors: Status 500</li></ul>

List

Route	`/api/sections/list/?sectionId=integer`
Type	GET

Route	`/api/sections/list/?sectionId=integer`
Response	• No errors: Status 200 and array of JSON section objects • No permission: Status 403 • Errors: Status 500

Log Endpoints

List

Route	`/api/logs/list/?seach=string&uname=string&ename=string&action=string&entity=string&sdate=string&edate=string`
Type	GET
Response	• No errors: Status 200 and list of JSON log objects • No permission: Status 403 • Errors: Status 500

Files Endpoints

List

Route	`/api/files/list`
Type	POST
Body	`{'sectionId': integer}`
Response	• No errors: Status 200 and list of JSON file objects • Error querying data: Status 400 • Errors: Status 500

Forums (Parent) Endpoints

Create

Route	`/api/forums/parent/create`
Type	POST
Body	`{'parentTitle': string, 'parentComment': string}`
Response	• No errors & Resource created: Status 201 • No permission: Status 403 • Errors: Status 500

Remove

Route	`/api/forums/parent/remove`
Type	POST

Route	`/api/forums/parent/remove`
Body	`{'parentId': integer}`
Response	• No errors: Status 200 • Error querying data: Status 400 • No permission: Status 403 • Errors: Status 500

Restore

Route	`/api/forums/parent/restore`
Type	POST
Body	`{'parentId': integer, 'creatorId': integer, 'parentTitle': string, 'parentComment': string}`
Response	• No errors & Resource created: Status 201 • Forum no longer available: Status 410 • No permission: Status 403 • Errors: Status 500

List

Route	`/api/forums/parent/list/?search=string`
Type	GET
Response	• No errors: Status 200 and list of JSON parent comment objects • Error querying data: Status 400 • Errors: Status 500

Forums (Child) Endpoints

Create

Route	`/api/forums/child/create`
Type	POST
Body	`{'parentId': integer, 'childComment': string}`
Response	• No errors & Resource created: Status 201 • No permission: Status 403 • Errors: Status 500

Remove

Route	`/api/forums/child/remove`
Type	POST

Route	`/api/forums/child/remove`
Body	`{'childId': integer}`
Response	• No errors: Status 200 • Error querying data: Status 400 • No permission: Status 403 • Errors: Status 500

Restore

Route	`/api/forums/child/restore`
Type	POST
Body	`{'creatorId': integer, 'parentId': integer, 'childComment': string}`
Response	• No errors & Resource created: Status 201 • Forum no longer available: Status 410 • No permission: Status 403 • Errors: Status 500

List

Route	`/api/forums/child/list/?parentId=integer`
Type	GET
Response	• No errors: Status 200 and list of JSON child comment objects • Error querying data: Status 400 • Errors: Status 500