

How To Website Really Good

A Guide by Finnbar (Web Admin 2021-22),
who is able to website really good

IMPORTANT NOTE

This is a reference document. You are **not** expected to have knowledge of everything in here, nor have even read it from cover to cover. This is solely here for you to go “ah, I need to do this job” and then have a reference for how to do it.

IMPORTANT NOTE 2

The reason this document is so long is because a fair amount of it is screenshots of the admin panel of the website. You will only need code if you want to add new features to the website, and in some cases you still won't need code.

Contents

Contents	1
Changelog	3
About This Document	4
Accessing the Code	5
Code Structure and Style	6
Updating The Website	8
I've written some new code, and want it to appear on the website. How do?	8
PythonAnywhere emailed me saying that I have to update the system image, help?	9
The Admin Panel	10
Website Apps	11
Assets and Gallery	12
How do you upload an image to the website?	12
How do you upload an image to the gallery?	13
What is a visual asset, and how do I use one?	13
Exec	15
How do I update who holds an exec position?	15
How do I add a new exec position?	15
How do I update the Exec History page?	15
Forum	17
How is the forum structured, and how do you update that structure?	17
How do you delete spam from the forum?	18
How do you ban a spambot?	18
How do you lock a thread?	19
How do you highlight important threads?	19
Inventory	20
How do I make a new inventory?	20
What is the process of loaning items?	21
What is the process of suggesting items?	23
How can I add, edit or delete entries in an inventory?	23
Messaging	25
How does the message reporting system work?	25
Minutes	27
How do you add minutes for a meeting?	27
How do you create new folders for minutes?	28
Navbar	29
How is the navbar arranged, and how can you change this?	29

Newsletters	31
How do you update the newsletter header image?	31
What CSS classes are available in the newsletter?	31
Notifications	33
How do notifications work?	33
How do you write code to send a notification to a user or Discord?	34
Can you send notifications manually?	34
Pages	35
How do you create static pages without having to mess directly with Python code?	35
Redirect	38
How do you create a new redirect and edit/delete existing ones?	38
Rpgs	39
How do I clean up completed events?	39
How do I make subevents?	39
How do I hide events until they're ready to be published?	41
How can I emphasise a particularly important event?	42
How can I set restrictions on who can sign up to an event?	42
Timetable	43
How do you input new schedule information?	43
How do you build a new timetable to show this information?	44
How do you add room links?	46
How do you update the Google Calendars?	46
How do you add special events to render in the Upcoming Events section of the homepage?	46
Users	48
How do you create an achievement?	48
How do you give an achievement to a user?	49
How does membership verification work?	50
Votes	51
Who can run an election? How can tampering be avoided?	51
How do you run an election?	52
How do I rerun an STV election in the case of a candidate withdrawing after the fact?	55
Website Settings	56
What do the website settings currently correspond to?	56
Authentication and Authorisation	57
What's the difference between users and members on the Admin Panel?	57
How do you ban or otherwise lock out a user?	57
How can you help a user who is unable to reset their password but needs to?	58
How do you give permissions to a certain user, for example during handover?	58
Miscellaneous Questions	60
How are the daily summary emails sent out, and how can I create other daily tasks?	60
Why are emoji allowed in some parts of the website but not others? Can I change this?	60

How can I increase the character limits on certain fields?	61
How can I change the website's (CSS) styling?	61
Command Line Tools	63
Appendices	64
FontAwesome	64

Changelog

Version	Author(s)	Notes
v1.0 (2/1/22)	Finnbar Keating (Web Admin 2021-22)	First complete version of this document, containing every app in the site currently.
v1.1 (9/1/22)	Finnbar Keating (Web Admin 2021-22), content from the Exec Handover Document probably written by Anna Bruce (Web Admin 2019-21)	Added <i>Miscellaneous Questions</i> and <i>Accessing the Code</i> , bringing in some content from the Exec Handover Document. Updated other sections according to recent website updates - newsletter editor updates, room links and the tutorial (under Website Settings).
v1.2 (21/1/22)	Finnbar Keating (Web Admin 2021-22)	Added info about editing styling, character limits and rerunning STV elections if a candidate is dropped. Also added dates to version numbers.
v1.3 (18/2/22)	Finnbar Keating (Web Admin 2021-22)	Quick fix in the voting section, detail on how SU membership verification works.
v1.4 (23/4/22)	Finnbar Keating (Web Admin 2021-22)	Fix in STV rerunning section, added page numbers for ease of navigation, mentioned pep8 formatting, merged handover checklist into main Handover Document, renamed Web Admin to Tech Officer when referring to the position in the present.

About This Document

[Everyone should read this bit.]

Hi! Welcome to this little reference document about the website. This document serves a few purposes, which are as follows:

1. For those doing admin stuff on the website (pretty much solely the Tech Officer) to find out how to do that. **Most of this can be done without touching code, and the day-to-day operation of the website requires no code.**
2. For those wanting to update the website's code, as to find out where in the code functionality is implemented, so that future improvements can be put in the right place.

The main body of this document is split into sections corresponding to each folder of code present in the website code, which directly corresponds to a large feature of the website. Each section outside of that is accompanied by a note in square brackets explaining who it is intended for.

Each section has subheadings corresponding to tasks you may want to do with that feature of the website. These subheadings will be as clear as possible - **so glancing at the Contents page should be enough to work out if something is covered.** If something isn't covered and you work out what needs including, please add a subsection including that information!

If the website is updated, the Tech Officer will likely have to update this document. Hopefully, most of these updates will be small improvements, like removing a complex explanation and replacing it with a simple one (e.g. because you've added a form that greatly simplifies doing something). If you add a new feature, it'll be a fairly big section to add, but given that I wrote a section for every feature on the website as of late 2021 you'll be fine.

The final sections of this document are a brief introduction to the command line tools (accessible from a PythonAnywhere console) and some useful appendices. The last appendix is a handover checklist for the website - so the Tech Officer should make sure they've done this alongside any other handover.

With that explanation done, let's go!

Accessing the Code

[This section is useful for anyone who wants to work on website code.]

The code for the website is hosted on our GitHub organisation, available at <https://github.com/WarwickTabletop/tgrsite>. The outgoing web admin should give you edit access to this repository. The first thing you should do is clone the repository, and then read the README .md, which contains all of the information you'll need to set up a local copy of the website for development.

There are three kinds of branches you need to be aware of. If you're not familiar with Git, you will need to become familiar with it¹. The branches are as follows:

- **main**: the main branch of the repository (yeah I know, surprising huh). This is for stuff we'd mostly be satisfied with being on the website, although is mainly a base for feature branches to be merged into. You can make tiny fixes directly to this branch if necessary (such as code formatting or changing constants), but building a new feature should be done on a new branch.
- **stable**: the branch that the website directly pulls off of, so acts as an exact replica of the code on the website. It is currently set up to reject any direct pushes, so each time you want to deploy you should create a pull request. The pull request currently prompts you to get someone else to test and sign off on the pull. It is a good idea not to override this feature. In order to do this, you should find some other friendly programmer (or a couple) to act as a sanity check for any changes, who can be given Triage or higher permissions on the GitHub.
- **Any other branch**: this is for developing a specific feature. When you're working on a new feature, create a branch off of master with a sensible, descriptive name, then do all the development on that, pulling from master when needed to keep the branch up to date.

The repository is also home to the issue tracker, which allows you to note down bugs that need fixing and feature requests. People are unlikely to directly report bugs to the issue tracker, so you'll mostly be talking to yourself in there, but it's still a useful place to keep track of everything. If you also tag your issues appropriately, you can help those who want to contribute!

¹ One introductory resource by CompSoc is <https://uwcs.co.uk/resources/#git-good>. If you prefer to learn by doing, you may also like <https://learngitbranching.js.org/>.

Code Structure and Style

[If you are interested in modifying the code, you should probably read this section. Admittedly, if you're very familiar with Django, you probably know all of this.]

The majority of top-level folders in the website code correspond to what Django calls *apps*, which are essentially groups of features. The website has a lot of these, which can contain the following files/folders, with the most common ones bolded.

- **templates/appname**, which contains all of the Django templates for this application.
- `templatetags`, which contains various bits of code that aid the rendering of templates.
- `management`, which contains backend tasks that have to be run manually or on a cron job.
- **admin.py**, which contains definitions of what should be shown in the Admin panel. To quote Anna (previous Web Admin): [this is] (imo) the single most useful feature of Django. Knowing how to adjust the admin panels has one of the best time input to saved ratio of any change you can make.
- `apps.py`, which defines the apps present in this folder. In general, you don't need to worry about this.
- **forms.py**, which defines the forms of an app. These can be rendered automatically later, allowing you to create pages that are essentially filling in a form and adding the contents of said form to a database really easily. If a page looks like a form, the form will likely be defined in this file.
- **models.py**, which contains the models (essentially database tables, but declared in a useful way) of the app. If you're unfamiliar with the idea of models, you should look up the MVC (Model-View-Controller) paradigm.
- `tests.py`. Maybe one day someone will write some tests.
- **urls.py**, which contains the routes of an app - essentially mapping the URLs that a user can access to the views that are shown.
- **views.py**, which contains code for giving a webpage back to the user. This usually involves getting some data from the database, and then displaying that data using a Django template.

A few of the folders don't correspond to apps. At the time of writing, these are as follows:

- `bootstrap`, which contains our own compiled version of Bootstrap 4.
- `static_resources`, which contains a bunch of important files needed by the website frontend.
- `tgrsite`, which is a sort of meta-app containing all of the other apps. It also contains templates used throughout the site, such as error pages and frequently used template snippets; as well as settings for the entire website.
- `templatetags`, which technically counts as an app but isn't really used like one. `templatetags` contains various bits of code that are used by multiple apps so don't

really have one home - for example, all of the code that helps with Markdown previewing lives in there. If you can't find something in one of the other apps, you may find it in `template tags`.

Note that this section is not a substitute for a Django tutorial - the correct terminology has been used throughout which should be easy enough to look up. If you're not familiar with Django and want to improve the website, you need to first become familiar with Django.

Code should be formatted in pep8 style. There are a variety of autoformatters that do this for you - so you should consider installing one of those. I (Finnbar) personally use VSCode so have autopep8 set up in that², but other editors are available.

² <https://stackoverflow.com/questions/67697045/visual-studio-code-and-autopep8-formatter> describes the process.

Updating The Website

[Another section about code. However, the Tech Officer will need to be familiar with this, as they are likely the only one who can perform these steps due to needing access to PythonAnywhere.]

I've written some new code, and want it to appear on the website. How do?

The website is hosted on PythonAnywhere, which gives you console access. In general, you should only ever have to enter a few fixed commands in the console and then you'll be sorted - so don't worry if you're unfamiliar with the console!

The website's code is always identical to that on the stable branch of the website's code, as mentioned in *Accessing the Code*. As such, the process of updating the website is pulling that branch from GitHub, and then doing some setup. The full process is as follows:

1. Log into PythonAnywhere using the Tabletop account credentials. For fairly obvious reasons, I'm not putting those credentials here³.
2. Open up a console. This can be done by going to the Consoles tab and starting a new Bash console, or reusing an existing Bash console listed under "Your consoles".
3. You need to be in the tgrsite directory to perform the website update. This should look something like the following (where XX:YY is the current time):

```
XX:YY ~/tgrsite (stable)$
```

If it does not look like that, run the following command:

```
cd ~/tgrsite
```
4. Next, you need to make sure that Python is able to access the relevant libraries. To do this, use the workon command as follows:

```
workon tgrvenv
```
5. With that done, it's now time to actually update the website. First, we update the code with:

```
git pull
```
6. Then the entire website update process is simply run with:

```
../update.sh
```

It may complain about MySQL here - feel free to ignore the warnings. It will also warn you about overwriting existing files, which you can happily say "yes" to.
7. If that worked, go to the Web tab of PythonAnywhere, and click "Reload www.warwicktabletop.co.uk". It'll take the website down for a moment, but then it'll be back up and ready to go.

Note that the Web tab mentioned in the final step is where you can find the access, error and server logs, which can help you debug any particularly spicy bugs.

³ Nah just kidding the password is 69420 lmao

PythonAnywhere emailed me saying that I have to update the system image, help?

PythonAnywhere has an underlying “system image” which dictates the operating system and system libraries that your code runs on. Naturally, as time passes the underlying OS will become old and need to be retired into the place where old operating systems go (the insecure shed). Fortunately, PythonAnywhere’s process for upgrading is fairly well-documented. I won’t be listing the entire process here because it may change, but what follows are the high-level steps I did and documentation that explains how to do them.

1. Backup the database. Hopefully, you will not need to use this backup, but this is so that if something goes *really* wrong, you won’t lose any user data. This is done via a console command given here: <https://help.pythonanywhere.com/pages/MySQLBackupRestore> Note that the URL they give in the command is for American PythonAnywhere rather than EU PythonAnywhere, so you’ll have to find the correct URL in the Databases tab of PythonAnywhere.
Also note: when I did this in November 2021, it really didn’t like the `set-gtid-purged` flag. If it doesn’t like it, omit the flag.
2. Disable the webapp so people cannot attempt to access it while you’re performing the upgrade. Go to the Web tab of PythonAnywhere, and there should be a fairly obvious button that disables the web app.
3. Actually upgrade the system image. This is documented on the main System Image page: <https://help.pythonanywhere.com/pages/ChangingSystemImage>
4. Delete pip’s cache, so that it doesn’t use any old packages. This can be done from a Bash prompt.

```
cd ~ && rm -rf .cache
```
5. Delete the existing virtual environment (this is the thing that means you have to do `workon tgrvenv` when you want to do stuff). I did this by deleting all of its associated files (`cd ~/.virtualenvs && rm -rf tgrvenv`) but there may be other ways to do this.
6. Create a new virtual environment with the same name as the old one. This should be possible via:

```
mkvirtualenv --python=python3.6 tgrvenv
```
7. Now, reinstall all of the packages required to run the website with `cd tgrsite && pip install -r requirements.txt`. This will take a while.
8. You also need to install a database handling package. Unless you’ve changed the database, it should be a MySQL database - so run `pip install mysqlclient`.
9. Finally, go back to the Web tab, reenale the web app and then restart it.

This should all work out. If not, check the error log (via the Web tab) for any error messages.

The Admin Panel

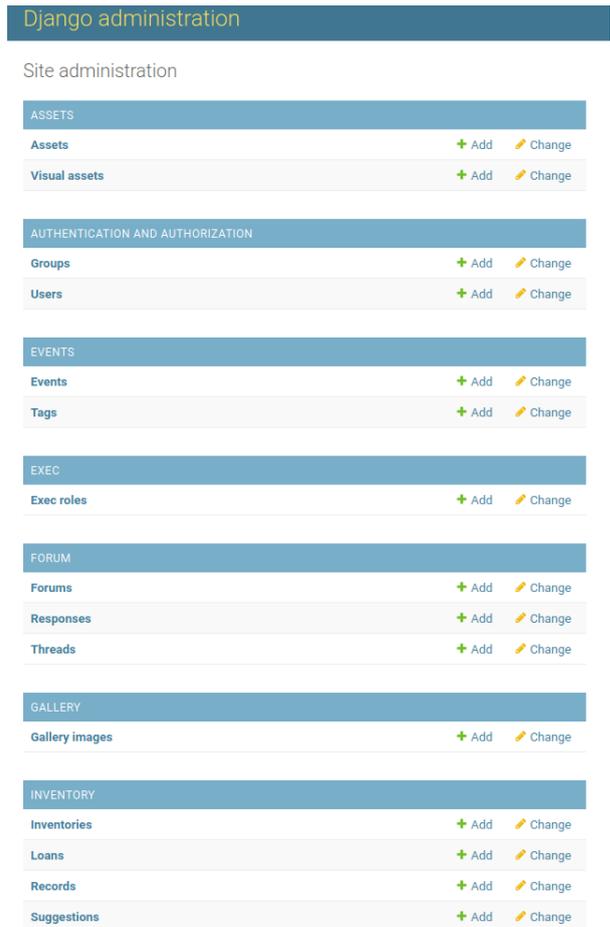
[If you are performing admin actions on the website, and have not done admin for a Django website before, you should read this section.]

The admin panel is where all of the website magic happens! It will look something like the screenshot on the right (note that only part of the panel is screenshotted for privacy and because it is too tall), and is broken up into the individual apps of the website (along with a few bonus ones). Predictably, it is accessible by clicking on Admin Site on the sidebar of most pages on our website, or by visiting <https://www.warwicktabletop.co.uk/admin/>.

The panel itself is fairly intuitive - for the most part, each entry corresponds to a model (essentially a database table) that you can add to or edit by clicking on that entry.

The panel comes with a very important caveat: if you add something with the admin panel rather than the equivalent website feature (e.g. adding a new event using the admin panel rather than clicking “create event” on the website), **some checking will be omitted**. Anything enforced directly by the form will still apply (e.g. character limits), but anything extra (e.g. whether an event is full) will be ignored - so you can make very weird stuff happen if you’re not careful. It also won’t do anything done after submitting a form on the website - for example, it will not notify anyone of a new newsletter created using the admin panel. As such, the panel should generally be used to fix errors and to do stuff not possible in the frontend of the website.

You could theoretically also directly access and edit the underlying database via PythonAnywhere instead of using the Admin Panel. This is omitted from this guide entirely because simply put, **you should not be doing this**. Changing anything via the database will skip most checks put in when editing data using the website - so can lead to all sorts of unpredictable and unhelpful behaviour, and could end up breaking the website completely. If you really, really need to do something with the database, make sure to talk to someone who knows the website well before doing anything, and also consider backing up the database first.



Website Apps

We now go through each of the website's main apps in turn. First, here is a quick helpful summary of each app on the website:

- **Assets** allows for images to be hosted on the website. It also has visual assets, which are used in a few places to provide versions of an asset for the different themes of the website.
- **Exec** shows all of the current exec of the society on the Exec page, allowing them to have a bio on that page.
- **Forum** is a forum. Our forum isn't really used except for hosting important AGM/EGM documentation, but maybe you can change that!
- **Gallery** provides a place to show off the best images taken at the society.
- **Inventory** provides the ability to keep inventories of the society's equipment, and to loan out said equipment.
- **Messaging** allows members to private message each other. Unfortunately, this has mostly been superseded by Discord.
- **Minutes** allows VPs to post the minutes for an exec meeting.
- **Navbar** allows you to update the navbar at the top of every page.
- **Newsletters** deals with the publishing of newsletters, including emailing them out to everyone subscribed to them.
- **Notifications** allows individuals to be notified of happenings on the website, such as the publishing of new newsletters and events.
- **Pages** allows for new static pages to be created without having to update the website as a whole.
- **Redirect** allows for redirects to be created, e.g. for short links to important pages.
- **Rpgs** allows members to publish events on the website - it is called rpgs solely for historic reasons.
- **Timetable** allows a schedule to be rendered based on room bookings. It also automatically adds links to common rooms.
- **Users** provides general things attributed with a user. It also provides achievements and talks to the Warwick SU API to verify if a website member has bought their membership.
- **Votes** allows for elections to be run.
- **Website Settings** lets certain settings be defined in code, which can be set using the admin panel (thus avoiding small code changes).

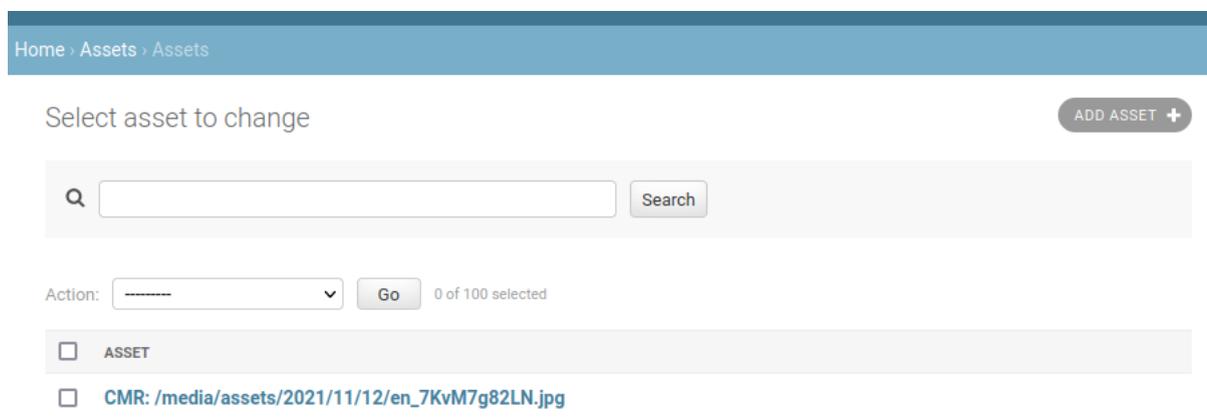
Assets and Gallery

[This section is useful for every exec interacting with the website.]

The assets app allows for images to be hosted on the website. It also has visual assets, which are used in a few places to provide versions of an asset for the different themes of the website. The Gallery app is similar, except it also adds the asset to the Gallery page (<https://www.warwicktabletop.co.uk/gallery/>).

How do you upload an image to the website?

In the Admin Panel, go to the Assets > Assets section. It should look something like this (but with many more assets).



Click "Add asset", which should get you the following form to fill in.

Add asset

Name:
Enter a name to help recognise it.

AssetFile: No file selected.

When you're done, click SAVE. You'll be returned to the list of all assets, except your new one will have been added. The URL is also shown in the notification and the new entry for your asset - in this example, the URL is warwicktabletop.co.uk/media/assets/2021/11/12/annual-german-meeting.png.

✔ The asset "very-secret-agm-thing: /media/assets/2021/11/12/annual-german-meeting.png" was added successfully.

Select asset to change

ADD ASSET +

Q Search

Action: Go 0 of 100 selected

- ASSET
- [very-secret-agm-thing: /media/assets/2021/11/12/annual-german-meeting.png](#)

How do you upload an image to the gallery?

Follow the same process, except with the Gallery > Gallery Images section of the Admin Panel. You will get a slightly different form (pictured below), which also asks for a caption (what to display on the gallery page) and full caption (what to display when the image is clicked on).

Add gallery image

Image: No file selected.

Caption:

Full caption:

As img: -

What is a visual asset, and how do I use one?

A visual asset is an asset with a light version and a dark version. These can currently only be used to give images to achievements, but are covered here for completeness.

Possible improvement for a future Tech Officer! Make them useful elsewhere. You may also want to look into having a URL that returns the right image for the current theme (discussed later in this section).

The idea is generally the same as the other two, except you'll be using the Assets > Visual Assets part of the Admin Panel instead. The form you'll be filling in is pictured below.

Change visual asset

Name:

Ruin Icon

Enter a name to help recognise it.

LightAssetFile:

Currently: [assets/themed/light/2021/03/28/ruin_opt.svg](#)

Change: No file selected.

DarkAssetFile:

Currently: [assets/themed/dark/2021/03/28/ruin_opt_dark.svg](#) Clear

Change: No file selected.

Note how you get two URLs - one for the light version and another for the dark version. There currently isn't a way to ping a single URL to get the correct image. This is currently done for performance reasons (according to Anna, who implemented it), but it might be useful in future to have a URL that can redirect appropriately.

Exec

[This section is useful to the Tech Officer and basically nobody else.]

The Exec app shows all of the current exec of the society on the Exec page, allowing them to have a bio on that page.

How do I update who holds an exec position?

All of the details of who exec is currently are modified in the Exec > Exec Roles section of the Admin Panel. Go there and click on the exec that you want to update, then simply change the incumbent (last entry) to the correct member. An example of this is shown on the right.

Note that the new member will not be granted website exec powers immediately (see [Authentication and Authorisation](#)) and will also have to manually update their bio. If someone is returning to the same role, you can leave the bio in place - otherwise you should back up the bio and then clear it. This makes it easy to tell who has written their bio and who still needs to since you don't have to check whether the bio was written by the right person, just whether it is present⁴.

Home · Exec · Exec roles · Web Admin (finnbar)

Change exec role

Sort index: Index for sorting. Lower value = earlier in list.

Role title:

Bio:

Responsibilities:

Incumbent: Member who is currently in this role

How do I add a new exec position?

Simply click “add new role” on the top right of the Exec > Exec Roles section. You will have to specify the role title and a “Sort Index” - this represents how high they should be on the list. This can be negative. Currently, the following indices are used: President (-19), Vice President (-18), Treasurer (-17), faction reps (1-7), non-faction reps (11-12), assistant reps (21-22), and appointed roles (31-33).

How do I update the Exec History page?

The Exec History page (<https://www.warwicktabletop.co.uk/page/exec-history/>) is a static webpage, which is updated using the Pages app. Which unfortunately means updating HTML manually. The page is just a massive HTML table, so you can pretty much learn by example just by reading the code already there.

Possible improvement for a future Tech Officer! Is there a way we could avoid this? Could implement this similarly to the Timetables part of the site, or at least have some way to say who was in an exec position (available via Exec Roles) in a given year that can then be made

⁴ This is a tip from Anna, and she knows what she's doing so you should listen to her.

into a lovely timetable.

Forum

[This section is useful to the Tech Officer and anyone else managing the forum.]

The Forum app is a forum. It has subforums and all of that exciting stuff as well.

How is the forum structured, and how do you update that structure?

The forum is a bit of a misnomer, because the main object that the forum is built out of is called a forum... so your forum contains multiple forums.

Possible improvement for a future Tech Officer! If you feel like renaming the main object to something other than forum (subforum perhaps?), please do.

Think of a forum as a folder, which can contain other forums (subforums - for example the CCGs forum contains individual forums for MtG, Netrunner and Yu-Gi-Oh at time of writing) and threads. Threads act as you'd expect - they consist of the post that started them, and a collection of responses. Forum, response and thread are the names of the three models contained within the Forum app.

To change the structure of the forum, you will be editing Forum objects, so need to go to Forum > Forums in the Admin Panel. This lists every forum and some important data about it, as shown by a cropped screenshot below.

Select forum to change ADD FORUM +

Action: Go 0 of 24 selected

<input type="checkbox"/>	NAME	DESCRIPTION	LOCATION	THREADS	SUBFORUMS
<input type="checkbox"/>	Character sheets	Post your 2021-22 character sheets here.	LARP	4	
<input type="checkbox"/>	Adventure Summaries (old)	If you missed a week, or you simply want to catch up, this is where you can find what's been happening.	LARP	2	
<input type="checkbox"/>	Character Sheets	Post your 2019-20 character sheets here.	LARP	9	
<input type="checkbox"/>	Announcements	(2019-20)	LARP	0	
<input type="checkbox"/>	Previous Years	The Characters and Announcements of years gone by	LARP	5	Character Sheets, IC Chat Area, OC Chat Area, Announcements, Adventure Archives
<input type="checkbox"/>	Official Society Business	A place for our Official Business™, such as AGMs, EGMs, and our Constitution	-	16	
<input type="checkbox"/>	Adventure Archives	Summaries of adventures from years gone by.	LARP / Previous Years	3	

If a new (sub)forum is required, simply click Add Forum in the top right. The basic bits that need filling in are screenshotted below.

Add forum

Parent:  

Sort index:  
Index for sorting. Lower value = earlier in list.

Name:

Description:

The parent is key here - this defines the single parent that a forum has. If it has no parent, then it appears at the top level (aka at <https://www.warwicktabletop.co.uk/forum/>) - otherwise it appears in the folder that you set it to. For example, if a new CCG became popular in the society, you could add a forum for it whose parent is the CCGs forum. **Please note that you can theoretically make an infinite loop here, by setting A to be the parent of B, and B to be the parent of A. Don't do this.** I genuinely don't know what happens if you do this, but it is at best confusing and at worst will cause the website to crash. EDIT: Anna has told me that it works completely fine. Still, from a moral standpoint you probably shouldn't.

The rest of the form should be self-explanatory. This form also reappears if you ever want to edit a forum (by clicking on the relevant forum in the list in Forum > Forums) - for example, to change its parent (move where it is) or update its name/description.

Finally, a forum can be deleted from the same page that it is edited - the delete button is at the bottom left of the page. If you do this, every associated thread and response will also be deleted. As such, only do this if you are absolutely certain that you want to, and try to prefer renaming to archive the forum instead. (For example, you might create an Archives forum that contains all old forums, and move a forum that you would delete into there instead.)

How do you delete spam from the forum?

The simple answer is to delete the associated object - if there is a single spam response to a thread, you can delete it by going to Forum > Responses, clicking on the offending object and deleting it. The same applies to a spam thread, except in Forum > Threads.

How do you ban a spambot?

This is covered in [Authentication and Authorisation](#).

How do you lock a thread?

Sometimes you only want exec to be able to respond to a thread, for example because it acts as a log rather than a place for conversation (for example: <https://www.warwicktabletop.co.uk/forum/thread/299/>). To make this happen, you can lock a thread, which prevents anyone except those with permissions (exec) from replying. Simply go to Forum > Threads, click on the thread you want to lock, and check the “is locked” checkbox, as pictured. Don’t forget to hit Save when you’re done!

Change thread

Forum: Official Society Business  

Title: Constitution and Mascot History

Body:
In the 2019 Term 3 EGM, it was mandated that the history of the constitution, along with old copies, would be kept on the society website. It was also mandated that the history of mascots would be kept on this site. It's now 2020.

This thread will serve as a record of these changes, which will be updated with each General Meeting. Or at least that's the aim, let's be honest we're all busy. Please note that current constitution can be found [on our documents page](https://www.warwicktabletop.co.uk/page/documents/).

Date posted:
Date: 10/02/2020 Today |
Time: 10:42:44 Now |

Is pinned

Is locked

Author: finnbar  

How do you highlight important threads?

Important threads can be highlighted by pinning them, which puts them at the top of the list with a cute little pin icon. It'll look a little like this:

Official Society Business

Forum / Official Society Business

Threads

 Constitution and Mascot History finnbar 	1 year, 9 months ago 3 responses
 EGM: 2021 Edition (meeting details and motion suggestions) morrowkei 	4 months, 2 weeks ago 2 responses

To do this, follow the same process for locking a thread, except tick “is pinned”.

Inventory

[This section is useful to anyone dealing with loaning out equipment, so likely the Board Games Rep among others.]

The Inventory app provides the ability to keep inventories of the society's equipment, and to loan out said equipment.

How do I make a new inventory?

This is done via the Admin Panel, in Inventory > Inventories. Click on "Add Inventory" in the top right corner to get the below form.

Add inventory

Name:	<input type="text"/>
Display:	<input type="text"/>
<input checked="" type="checkbox"/> Suggestions	
<input type="checkbox"/> Loans	
Loan conditions:	<div style="border: 1px solid #ccc; height: 100px;"></div>

The fields are as follows:

- **Name:** what you want the inventory to be called. For example, we have inventories for each of the parts of the society, like RPGs. But we also have had inventories for more specific purposes, such as one containing a subset of our board games for loaning out during COVID (called BoardgameLoans).
- **Display:** what name you want displayed when referring to an inventory. If this is left blank, it will default to the name.
- **Suggestions:** whether to accept suggestions for things to add to the inventory. If this is checked, the inventory will have a suggestions page available for people to submit their suggestions to. For example, the board games inventory currently has one: <https://www.warwicktabletop.co.uk/inventory/boardgames/suggestions/>.
- **Loans:** whether to allow items in an inventory to be loaned out. If this is checked, the inventory will have a loans page available for people to ask for loans. The board games inventory also has one of these: <https://www.warwicktabletop.co.uk/inventory/boardgames/borrow/>.

- **Loan conditions:** what conditions the loan has on it. Each loans page has a loan conditions section - the text in this field is what will show up in that section. Markdown is accepted.

The inventory will then be available at the url <https://www.warwicktabletop.co.uk/inventory/x>, where x is the name of your new inventory. You'll have to edit the navbar to include this entry if you'd like it to be up there - this is discussed in the Navbar section.

What is the process of loaning items?

To loan out some items using the website, the first step is to go to the loans section and request a loan. The loans section looks like the below screenshot to an admin, with buttons to request a loan, request a loan for someone other than you, and a list of loan requests (mostly censored from the screenshot as this is real data). Non-admins will only see the Request Loan button and a list of their loan requests.

Board Games Loans

The screenshot shows the 'Board Games / Loans' page. At the top, there is a breadcrumb trail 'Board Games / Loans'. Below this, there are two main buttons: 'Loan Conditions' (with a document icon) and 'Request Loan' (with a green border). Below the 'Request Loan' button is another button, 'Request Loan for Member' (with an orange border). The main content area is a list of loan requests, each represented by a horizontal bar that has been redacted with black. The bottom-most request is visible, showing a back arrow, the name 'Finnbar (finnbar 👑)', the dates '1 Dec 2021 - 4 Dec 2021', and a purple circle with the number '4'.

Clicking on Request Loan leads to a form for the borrower to fill in. They simply have to select what items they want and when they want to borrow them for. (Note that only items the society owns are allowed to be borrowed, not items being held by the society from its

members.) Once this is filled in, an admin can access the Loan Request page by simply clicking on the associated request. An example Loan Request page for a completed loan is given below.

Loan Request

Completed

Edit

Requested by Finnbar ( **finnbar** )

When?
1 Dec 2021 - 4 Dec 2021

What?

Codenames
Concept
Identik
Wavelength

Request authorised by:
Finnbar ( **finnbar** )

Objects taken:
1 Dec 2021
Witnessed by: Finnbar ( **finnbar** )

Objects returned:
4 Dec 2021
Witnessed by: Finnbar ( **finnbar** )

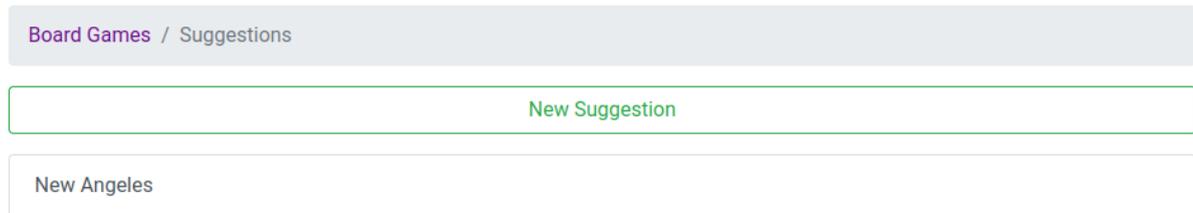
(In this case, the borrower dealt with the loan themselves as they were exec at the time. Normally this would not happen and you'd have a variety of people listed.)

The page gives the entire process of a loan - the loan being created, approved, items being physically taken and finally being returned. For each part of the process, a button is provided (not present in the screenshot due to this being a completed loan) which the exec member witnessing the loan can click to say that part of the process is done. So in short, when part of a loan is done, the relevant exec doing this should go to the relevant Loan Request page, and click the button to say that it's done. (Thank Anna for that, it's a very nice loans system.)

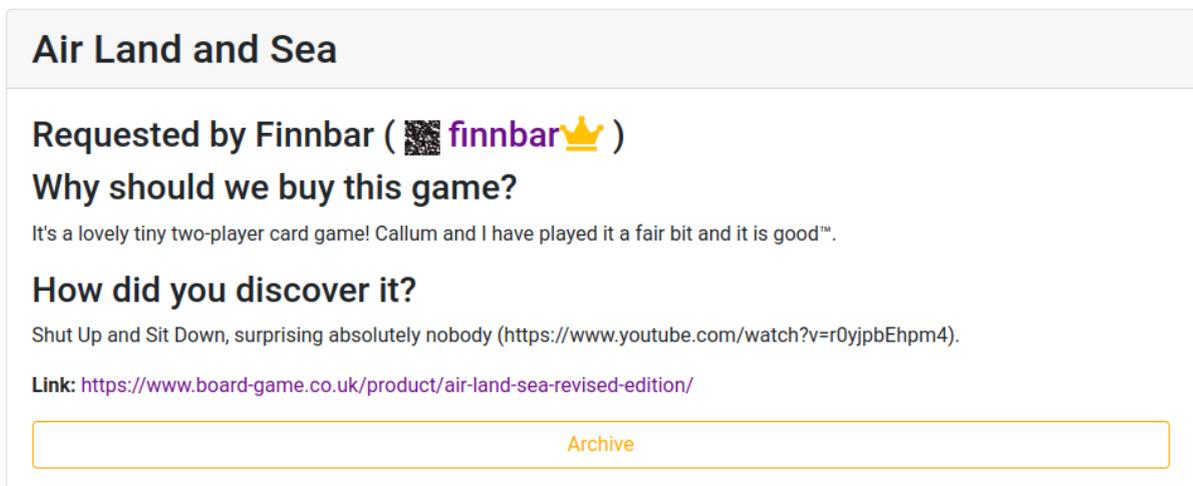
What is the process of suggesting items?

This one's much easier! Here's the suggestions page as seen by an admin (although only the first suggestion is shown to avoid showing lots of real data) - the page is the same for a non-admin except without the list of the games.

Board Games Suggestions



The New Suggestion button leads to a simple form for people to fill in with their suggestions. Admins can then click on a suggestion in the list to see the full form response, and then archive it once it has been fulfilled or decided against. Here is an example suggestion from the website (hey it's one of mine) - you can see the full form response, a helpful link and a button to archive the suggestion.



How can I add, edit or delete entries in an inventory?

Adding items is simply done from the main page of that inventory - admins will see an Add New button that they can use to add a new item, as screenshotted below. Editing an item is similarly simple - click on its entry and there will be a large Edit button.

Board Games

Board Games	
Suggestions	Loans
Add new	
1856	
7 Wonders	
A Feast for Odin	
Agricola	
All Creatures Big and Small	
Amun-Re	
Articulate	

Annoyingly, to delete an entry you'll have to go via the Admin Panel. Inventory > Records holds all entries in all inventories, and you can click on an entry to both edit (by changing the fields and clicking Save) and delete it (by clicking Delete in the bottom left). You can also bulk-delete items by clicking on the checkboxes next to the items you want to delete on the Inventory > Records page, and then select "delete items" from the Action dropdown at the top. You can also filter the view that you see down to just one inventory via the sidebar on the right. A screenshot annotated with all of those can be seen below.

The screenshot shows the 'Inventory > Records' page. At the top, there is a search bar and an 'Action' dropdown menu. Below the search bar, there is a table with columns: NAME, QUANTITY, OWNER, and INVENTORY. The table lists several board games, including '1856', '7 Wonders', 'A Feast for Odin', 'Agricola', 'All Creatures Big and Small', 'Amun-Re', 'Cada Collection', and 'Articulate'. To the right of the table is a 'FILTER' sidebar with options like 'By inventory', 'All', 'Board Games', 'LARP', 'archive', 'RPGs', 'CCGs', 'Wargames', 'Misc', and 'BoardgameLoans'. Annotations with arrows point to the 'Action' dropdown (labeled 'Action dropdown (bulk deletion)'), the checkboxes in the table (labeled 'Checkboxes (bulk deletion)'), and the 'FILTER' sidebar (labeled 'Filter sidebar').

Possible improvement for a future Tech Officer! C'mon add a delete button to inventory entries. When this feature was made, we didn't expect deletion to be very commonly used, but (at time of writing) we've had two auctions - so making deletion easier may be sensible.

Messaging

[This section is useful to the Tech Officer, who is the only individual that ~~uses the messaging system~~ sees reported messages.]

Messaging allows members to private message each other. Harmful messages can also be reported.

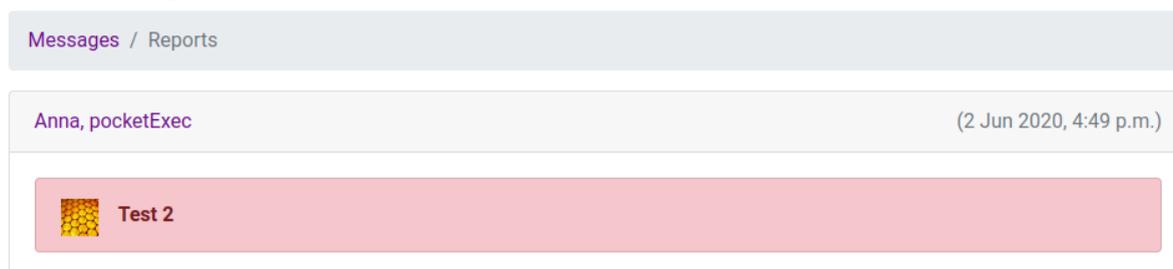
How does the message reporting system work?

To report a message, a user should click the “report message” button available by hovering over a given message. When this is done, the Tech Officer will be notified about this report and it will be made available in the Pending Reports section of the website

(<https://www.warwicktabletop.co.uk/messages/reports/>, also linked on the Messages page).

Here is a screenshot of the main Pending Reports page (with fake data).

Messages



The screenshot shows a breadcrumb trail: Messages / Reports. Below this, a message from Anna, pocketExec is shown, dated (2 Jun 2020, 4:49 p.m.). The message content is a red box with a yellow honeycomb icon and the text "Test 2".

You can then click on a report to see the context, for example like the below.

Anna, pocketExec



The screenshot shows a breadcrumb trail: Messages / Reports / Anna, pocketExec. Below this, a report for Anna's "Test 2" message is shown, dated 8 months, 4 weeks ago. The report reason is "Testing", dated 3 Apr 2021, 10:38 a.m., from pocketExec. Below the report, a message from pocketExec is shown, dated 1 year, 6 months ago, with the text "Test".

In this example, Anna’s “Test 2” message has been reported with the reason “Testing”. You can also see the context - pocketExec sent the message “Test” first! (What a scoundrel.)

To resolve this, you can click Resolve Report (hover over the offending message), which allows you to respond to the report. You’ll see a modal that you can fill in, which will be reported back to those involved. If you need to take more serious action (such as banning the user) you’ll have

to take this manually using the Admin Panel - more on this in Authentication and Authorisation.

Minutes

[This section is useful to the Tech Officer, and possibly the VP as well.]

The Minutes app allows VPs to post the minutes for an exec meeting.

How do you add minutes for a meeting?

Creating entries in the minutes is pretty self-explanatory. The Minutes page has a Create New button, which brings you to a form as screenshotted below.

Name

This is the name that appears in the URL - Week_n is a sensible example. Letters, numbers or underscores only

Folder

Title

Body

B **I**   **H**      

Date

[Save](#)

The fields are as follows:

- **Name:** what the URL should show. Usually this should be week_n, for a given week number. Note that this will be shown after the folder name.
- **Folder:** what folder this entry should be in. The Tech Officer should create a folder per term of each year (e.g. 2021-22/term_1) which is detailed in the next section - select the relevant folder from the dropdown here.
- **Title:** the title that should be shown at the top of the minutes entry, e.g. "Term 1 Week 4/5 - Socs Fair, Merch, Auction, Quiz".
- **Body:** the contents of the minutes. Accepts Markdown.

- **Date:** the date at which the meeting was conducted (not when the minutes were published). Minutes are ordered based on when they were conducted rather than when the minutes were published.

How do you create new folders for minutes?

To create a new folder for minutes (e.g. because you've entered a new term or have some meetings about a specific project that you'd like to group together), you'll need to use the Admin Panel. Go to Meetings > Folders and click Add Folder in the top right. This simply asks you to specify a name for the folder and its parent.

Note that if you want to create a new subfolder in a new folder (e.g. because you've just entered a new year and need a new Term 1 folder within that), you have to create the parent folder (2022-23, for example) and then create the term_1 folder with 2022-23 as its parent.

Navbar

[This section is useful to the Tech Officer, and to any amateur web designers who want to know what the limitations are.]

The Navbar app allows you to update the navbar at the top of every page.

How is the navbar arranged, and how can you change this?

The navbar is controlled by the Navbar part of the Admin Panel, which means you can change it without messing with any website code. There are two parts to the Navbar app - Bar Items, which are the single items on the navbar like Events and Schedule; and Bar Dropdowns, which are the lists of items accessible from a dropdown like Society and Equipment. Note that the logo on the left and all of the various account bits on the right (notifications, account, logout) are not controllable via the Admin Panel and would have to be moved with code.



Each item on the navbar, regardless of whether it is a Bar Item or a Bar Dropdown, has a sort index, which dictates the order of items on the navbar. Items are sorted from smallest index to largest, so you just need to number the entries accordingly to get them to display in the right order. If you give two entries the same sort index, they will be displayed in an arbitrary order, possibly random with each refresh. (I don't know the exact details.) So don't do this.

To create a single item on the navbar, go to Bar Item and click Add New in the top right. You'll get a form like the following.

Add bar item

Sort index:	<input type="text" value="0"/>
Text:	<input type="text"/>
Icon:	<input type="text"/> <small>The name of the icon to use (including the fa- prefix)</small>
Icon set:	<input type="text" value="-----"/> <small>The fontawesome icon set this logo is from. (Please don't use Regular Style (far) icons: keep the style consistent)</small>
Target:	<input type="text"/> <small>The url this menu item should lead to.</small>

We've already discussed most of these or they are obvious, but there are two awkward ones that we should discuss in a little more detail. Icon and Icon Set refer to FontAwesome icons - see the FontAwesome appendix for details on this, and what the fields should be.

To create a dropdown on the navbar, go to Bar Dropdowns and add a new one using the button in the top right. The form for this is similar but longer.

Add bar dropdown

Sort index:

Text:

Icon:
The name of the icon to use (including the fa- prefix)

Icon set:
The fontawesome icon set this logo is from. (Please don't use Regular Style (far) icons: keep the style consistent)

DROP DOWN ITEMS

Drop down item: #1 ✕

Sort index:

Text:

Icon:
The name of the icon to use (including the fa- prefix)

Icon set:
The fontawesome icon set this logo is from. (Please don't use Regular Style (far) icons: keep the style consistent)

Target:
The url this menu item should lead to.

[+ Add another Drop down Item](#)

This essentially creates a folder (the top half of the form) containing each dropdown (created using the bottom half of the form, created identically to bar items).

When you want to edit any of these, go to the relevant part of the Admin Panel (Bar Items or Bar Dropdowns) and edit them by clicking on the entry you'd like to edit.

Tech Officer note: if you'd ever like to upgrade FontAwesome, you'll need to both change the dropdown in use here for Icon Set and the underlying file loaded by the website.

Newsletters

[This section is useful to the Tech Officer. It can also partially serve as a reference to the Co-Op Officer, and notes how emails are sent which may be useful to future implementers.]

The Newsletters app deals with the publishing of newsletters, including emailing them out to everyone subscribed to them.

How do you update the newsletter header image?

Annoyingly, this is something you have to do in code. But you should only need to do this once per year. If you prefer to learn via just looking at previous examples, this commit should suffice: <https://github.com/WarwickTabletop/tgrsite/commit/2ec34de9159f5318a92d09164a78006348f297b2>

Two things need to be done in this - the image has to be uploaded to `static_resources`, and then an entry needs to be made for it in the Newsletters model. The latter is just a tuple of the path to the image (after `static_resources` - so in the linked example the full path is `static_resources/images/newsletter_banner21-22.png`, but you only need everything after `static_resources`) and the year that it is relevant for. After this, it should just work - but since this is a database field, you can always update this with the correct entry if it doesn't work.

Possible Tech Officer improvement! Somehow allow the creation of header images without changing code. This might be possible via a new database table mapping image URLs to years with some default for those not included.

What CSS classes are available in the newsletter?

All of the classes corresponding to the newsletter can be found in this `.scss` file: https://github.com/WarwickTabletop/tgrsite/blob/main/bootstrap/tgrsite_newsletter.scss. To go through these in turn:

Note: to add to these, you'll need to edit the linked `scss` file and then compile using the `.sh` script in the same folder. If you don't do this, the actual CSS files that the website uses won't be updated (SCSS isn't read by the website) so your changes will have no effect. See [How can I change the website's \(CSS\) styling?](#) in the Miscellaneous section for details on this.

- `.news-bg` adds a background to the associated header. The Co-Op Officer should use this with every heading. `.news-bg-1` and `.news-bg-2` fix what colour the background is.
- `.news-image-left` and `.news-image-right` make an image float left or right. This means that when you shrink it (using `.news-width-x`, explained next), the image will stay on that side and the text will take up all surrounding space.

- `.news-width-x` (defined for `.news-width-1` to `.news-width-5`) sets the width of an image. `width-1` is the smallest and `width-5` is the largest. You also have `.news-width-full` to force an image to be the full width (otherwise it defaults to a certain height to avoid taking up too much space).

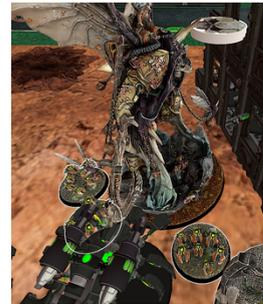
Finally, Bootstrap provides `.clearfix`, which makes sure that anything following it starts on a new empty line. This is useful for text which is much smaller than the image following it - for example, in this screenshotted newsletter, each paragraph is followed by a `clearfix`. This means that the second paragraph isn't started next to the first image - it has its own section with its own image next to it.

What happened last week?

An update on the standings of the Blood Bowl League! The scores are evidently wrong because all the teams are equal in the only field that matters - team names.

Team Name	Race	Coach	Played	Points	TD #	Win	Draw	Loss	TD +	TD -
The Carcassonne Meeples	Imperial Nobility	Alex	1	3	2	1	0	0	2	0
Da Black Rock Nuttaz	Orc	Liam O	1	0	1	0	0	1	1	2
The Rodents	Skaven	Joshua	1	0	2	0	0	1	0	2
Filly shades of green	Wood Elf	Matthew	1	3	1	1	0	0	2	1
Old Bones	zombing Undead	Lewis	1	3	1	1	0	0	1	0
La Vampire Voila	Vampires	Reynin	1	0	1	0	0	1	0	1
Gittertown Gnam-Skambler	zombing Undead	Adam	1	3	1	1	0	0	1	0
The Wood the Bad and the Ugly	Wood Elf	Josh	1	0	1	0	0	1	0	1

Scarabs - tiny horrible self-replicating robot bugs which are basically evil flying roombas, "heroically" sacrificed themselves to blow up Mortarion, a millenia old all powerful demon primarch and general all around durable individual.



Brave Sir Knight discovered the weakness of heavy armour on the sports field - when you trip on the touchdown line, your bones break off into smaller bones that declare independence, and everyone in the crowd laughs at you, you can't easily turn around to yell back at them.



These are documented within the editor as well - click on the "?" button in the newsletter editor to get the full list. If you add any more classes, don't forget to update that reference as well.

Notifications

[This section is useful to the Tech Officer and to developers looking to add notifications to their code.]

The Notifications app allows individuals to be notified of happenings on the website, such as the publishing of new newsletters and events.

How do notifications work?

Notifications are sent out by various parts of the website to let users know of stuff that has happened. Their implementation is provided in `notifications/utils.py`⁵, which shows how to generate a notification - just call the relevant method in that file.

Notifications are currently just strings paired with an icon and a URL. This means that you cannot currently change the contents of a notification after it is sent, and a notification will have a dead URL if its associated content is deleted (for example, if an event being notified about is deleted).

Possible Tech Officer improvement: I feel like I don't need to spell this out since the previous paragraph is about limitations, but you may want to look into this. However, this is likely quite difficult - issue #274 (<https://github.com/WarwickTabletop/tgrsite/issues/274>) has some commentary that may be worth looking into.

Currently, when any of the following happens, the website sends notifications.

- An achievement is awarded.
- A message is received.
- Various occurrences on an event: when someone joins your event, leaves it, when you are kicked or kick someone from an event.
- Everyone is notified when an event is created.
- When your forum post is replied to.
- When your loan request is accepted/rejected.
- Admins are notified when anything happens on a loan so they can respond to it.
- The Tech Officer is notified whenever a message report is filed.

These notifications rely on a user having subscribed to these notifications, which is controlled by a user's associated Notification Subscriptions object. An example of one such object in the Admin Panel is given on the right. While you can change these notification subscriptions in the Admin Panel, in general

Change Notifications Subscription

Newsletters:	Full Email
Receive Direct Messages:	Summary Email
A Loan Request Gets Updated:	Online Only
Someone Joins Your Event:	Summary Email
Someone Leaves Your Event:	Summary Email
Removal from an Event:	Online Only
Addition to an Event:	Online Only
Reply to a Forum Post You Participated in:	Online Only
A New Event is Created:	Summary Email
You Get an Achievement:	Online Only
Miscellaneous:	Summary Email
Member:	finnbar

⁵ <https://github.com/WarwickTabletop/tgrsite/blob/main/notifications/utils.py>

you should prefer asking a user to do it themselves via <https://www.warwicktabletop.co.uk/notifications/preferences/>. (However, if someone were to email you saying that they want to unsubscribe, then you may want to use the Admin Panel rather than going back to them like “do it yourself”.)

How do you write code to send a notification to a user or Discord?

In short, you need the helper functions in `notifications.utils`. There are four that you need to worry about:

- `notify(member, type, content, url)`⁶ notifies the given member if they are subscribed to notifications of that type, with the content in `content` and a link pointing towards the given `url`.
- `notify_bulk(members, type, content, url)` does the same but for a collection of members - e.g. if you wanted to notify everyone in a message thread about a new message, you'd probably use this.
- `notify_everyone(type, content, url)` notifies every user on the website. Useful for things like a new event or newsletter.
- `notify_discord(content, member)` sends a notification to the Discord webhook, associating it with the provided member.

With these, you can send a notification from anywhere within code. This means adding a new notification to something happening is easy! Just don't go drunk with power.

Can you send notifications manually?

Simply put, you can't. The relevant database model isn't accessible from the Admin Panel, so without manually creating it in the database directly (which you generally should not do since that bypasses most sensible checks done by the code) you can't create a notification.

Possible Tech Officer improvement: have a way to generate notifications (only for admins...) that can be sent to individuals or the entire membership.

⁶ Each of these also have an optional merge-key argument, which does specific database stuff that I'm not going to pretend to be able to explain.

Pages

[This section is useful to any exec who wants to add a static page to the website.]

The Pages app allows for new static pages to be created without having to update the website as a whole.

How do you create static pages without having to mess directly with Python code?

Various pages on the website, including the homepage itself, are just static pages rendered from HTML saved in the database via the Pages app. This means that you can create new static pages (no database interaction) without having to update the underlying code being run by the website. There is a very long form for creating this, but it is mostly ignorable - we'll go through the elements of it now.

Name:	<input type="text"/>
	<small>Internal name of page</small>
Title:	<input type="text"/>
	<small>Page title to display in titlebar of browser/tab</small>
Page title:	<input type="text"/>
	<small>Title to appear at the top of the page</small>
Breadcrumb child:	<input type="text"/>
	<small>Child element of the breadcrumb list</small>
Body:	<div style="border: 1px solid #ccc; height: 100px; width: 100%;"></div>
	<small>Page contents</small>

The first part of the form is the important bit. We have the following fields:

- **Name:** the internal name of a page. This is what appears in the URL - a page called x will be present at <https://www.warwicktabletop.co.uk/page/x/>.
- **Title:** this is what is displayed on the title of your browser or webpage. This is likely a shortened version of the page title.
- **Page Title:** this is what is displayed at the top of the page - we'll look at an example of a complete webpage in a moment to see it in action.
- **Breadcrumb Child:** the last element of the breadcrumb list. Again, we'll look at an example in a moment.

- Body:** this is where all of your HTML goes (or Markdown, if you tick the box near the end of the form allowing Markdown). Note that the entire page will be contained in a Bootstrap container, so you can use all of Bootstrap's useful stuff (and actively should) such as its Grid system⁷. If you're unfamiliar with Bootstrap, you can look at existing pages by clicking on them in the Pages part of the Admin Panel.

To see these in action, here is an existing webpage with the important parts labelled.

page title → Merchandise 2021

breadcrumb child

Merchandise 2021

Quick Links

- Admin site
- My profile
- Notification Settings
- Log out
- Report a Bug

Edit

Please note that any products with customisation on them have an associated Google Form, which you must fill in after buying said products!

Anything you've ordered will be available for collection from society events in term 2.

Hoodies

These are warm and comfy, have our logo on them, and come in a variety of colours with three (short) lines of personalisation! So yes, your hoodie can say fuck. Get them on the [SU website](#).

After buying your hoodie on the SU website, you **must** fill in this [Google Form](#) to inform us how you would like it customised.

Pride/Pronoun Badges

Display your pronouns and flags in style, with these pins designed by resident frog (and I guess artist too) Ares. They're cheap, fun, and very pretty - so get them now on the [SU website](#).

After buying your pronoun badge on the SU website, you **must** fill in this [Google Form](#) to inform us how you would like it customised.

Exec Role Badges

Are you an exec or ex-exec who wants to make their role clear? Hey we have a badge for that. Exec and ex-exec, go to the [SU website](#) to get your badge!

After buying your role badge on the SU website, you **must** fill in this [Google Form](#) to inform us how you would like it customised.

leftbar

body →

I hope you enjoyed my annotations. Onto the other part of the form!

⁷ <https://getbootstrap.com/docs/4.0/layout/grid/#grid-options>

Head:
Custom HTML to go in the of the page

Css:
Custom CSS styles for the page

Leftbar:
Left sidebar contents (use cards)

Markdown
Enable markdown rendering in this page

Permission:

Achievement:  

WIDGETS

[+ Add another Widget](#)

BREADCRUMB PARENTS

URL 	NAME 	ORDER 	DELETE?
+ Add another Breadcrumb Parent			

Note that the Head, CSS and Leftbar fields have been shrunk to make it all fit into the screenshot. With that disclaimer out of the way, let's go through the remaining parts of the form.

- **Head:** HTML to include in the head of the page, such as including new Javascript scripts and fancy stuff.
- **CSS:** any custom CSS to include in the page. Generally you shouldn't need much here, since Bootstrap will do most of the heavy lifting.
- **Leftbar:** any HTML you'd like to render in the leftbar (pictured earlier).
- **Permission:** who should be able to access the page - this can be public, only logged in users, members, ex-exec (and exec), or exec only.
- **Achievement:** what achievement should be awarded on visiting this page, if any.
- **Widget:** what widgets a page should have. Currently this can be one of two widgets - Event Poster, which shows off a pretty poster for the next event; and Upcoming Events, which lists all upcoming events. These use the special events created in the Timetable - check that section for details.
- **Breadcrumb parent:** to make longer breadcrumbs that contain more than a child (for example if you want to have a whole folder hierarchy) you can add a breadcrumb parent. This is a link that displays before the breadcrumb child - for example, if your breadcrumb child is Tabletop Radio 2021-22, you might want a breadcrumb parent called Tabletop Radio that links to some kind of navigation page.

Most of these are unneeded and their defaults will do, but they're listed for if you need them.

Redirect

[This section is useful to the Tech Officer, and any exec interested in rickrolling.]

The Redirect app allows for redirects to be created, e.g. for short links to important pages. We use it to link to forms (preferring clear links to Google shortlinks) and to other pages of note.

How do you create a new redirect and edit/delete existing ones?

Unsurprisingly, through the Admin Panel, through Redirect > Redirects. Here's the form you'll be presented with:

Add redirect

Source:	<input type="text"/>
Sink:	<input type="text"/>
<input type="checkbox"/> Permanent	
Achievement:	<input type="text"/>  
Usages:	0 <small>The number of times that link has been used</small>
Url:	-

This is a simple form, but I'll go through the fields anyway:

- Source: how you want the redirect to look. If your source is x, then a user going to <https://www.warwicktabletop.co.uk/x> will fire the redirect. Don't make a redirect sink have the same name as an existing app, as it will not work.
- Sink: the URL you want to redirect to. Put your rickroll link here.
- Permanent: whether the redirect is permanent. If this is checked, then when a user uses this redirect for the first time the browser will note this and never query your URL ever again - so it will never check if the redirect has changed. Only tick this if you're sure that this redirect will never need changing.
- Achievement: what achievement to give a user that uses this URL. Defaults to no achievement.
- Usages: a readonly field saying how many times a redirect has been used. Useful for knowing how many people you got with that rickroll.
- URL: a readonly field with the URL that redirects.

Editing and deleting redirects are done in the same way as everything else in the Admin Panel - click on an entry to get its details, and save it to update those details or hit the delete button to delete it.

Rpgs

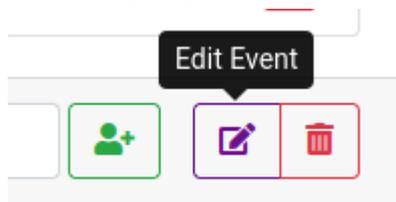
[This section is useful to the Tech Officer, and anyone who may have to clean up events.]

The Rpgs app allows members to publish events (which don't have to be tabletop RPGs) to the website, which can be signed up to by other members.

How do I clean up completed events?

Users should set dates on their events after which they're done (the Finishes field) so that the events are automatically marked as completed after that date. Many users don't, so you will sometimes have to go through and clear up events.

The Tech Officer (along with a few other exec) has the ability to edit published events regardless of who owns them. To mark an event as completed (as to hide it from the main Events page), go to an event, and click on Edit Event in the bottom right of the event.



Then, go down to the checkbox "Is in the past", tick it and save.

Is in the past
Has the event already happened?

You can also do this via the Admin Panel if you really want.

How do I make subevents?

You can now make events which have children events. This requires users to sign up to the main event as well as a child event - for example, for Intro to RPGs you usually want a main Intros event, and then an event for each oneshot being run. This looks something like this:

RPG Intros: The Introening

Events / RPG Intros: The Introening

RPG Intros: The Introening Now

Many rpg
Description here.

Run by: **finnbar** 
Players: **0/69**

[Message Current Members](#)

Sub-Events

The One-Shot: Blah
Not D&D 0/6

Username   

Each event then links to its parent, as shown:

The One-Shot: Blah

Events / The One-Shot: Blah

The One-Shot: Blah

Not D&D aa

↑ Sub-event of RPG Intros: The Introening

Blah.

Run by: **finnbar** 
Players: **0/6**

 This event requires you to sign up to the parent event, RPG Intros: The Introening.

[Message Current Members](#)

Username   

An event can be designated as a subevent through the Admin Panel by setting the event to have a parent. Go to Events > Events and select the event you want to be a subevent. Near the bottom of the form you'll find these three fields:

Parent: ✎ + ✖
The parent event. Signing up to this event will automatically sign you up to the parent event.

Child signup only
Prevent signing up to the event directly. This forces people to signup to at least one of the children events if they want to signup to the main event.

Success message:

Replace the default success message with a custom one. Only replaces on a direct signup, transitive signups use the default message. Accepts markdown formatting.

Parent lets you set the parent event of a subevent. **Child signup only** forces a user to sign up to the child event, which will automatically sign up a user to the parent event - if unchecked, then a user can sign up to the parent without signing up to a child event. **Success message** sets the message said when someone signs up to an event successfully - so you can say something like “please sign up to the signup spreadsheet to pick your team now” rather than the default.

How do I hide events until they’re ready to be published?

The easiest way is to save the event as a draft - make sure that the “Publish this event immediately” is unchecked, and then the event will remain a draft and unavailable to all except admins on the website.

An alternative way is to unlist an event. This means that you can only access the event directly with the URL it is situated at, rather than from the front page. This doesn’t make an event unavailable to anyone however - so anyone can access the event if they have the URL. You can activate this from the Admin Panel: go to Events > Events, click on the relevant event, check the Unlisted checkbox and save.

Unlisted
Prevent this from appearing on the events listing

This is also useful for subevents due to a subtlety in how the unlisting works (thanks to Anna for this tip) - any unlisted event will still appear as a subevent within the parent. This means that you can stop the main event list from being spammed with loads of subevents simply by unlisting every subevent since users can still access those subevents from the main event.

You can also combine these two methods if you’d like by doing both of the above processes - thus not listing it on the Events page nor allowing access to anyone except admins.

How can I emphasise a particularly important event?

Events can be pinned (displayed prominently at the top of the events list) through the Admin Panel. Go to Events > Events, select the relevant event and go down to the Pinned checkbox. Check this and save changes to pin that event to the top of the list.

How can I set restrictions on who can sign up to an event?

The website provides two ways to restrict signups, which both have their associated checkboxes in the Admin Panel. Go to Events > Events and select the relevant event. You can find the following two checkboxes in the form:

Discord

Require users to have a discord username listed before signing up

Member only

Require users to be verified members to sign up

The Discord checkbox forces users to have associated a Discord username with their account - which is particularly useful for events being run on Discord. The Member Only checkbox only allows users who have verified their society membership to join the event. (Discussion of Membership Verification can be found in the Users section.)

Timetable

[This section is useful to the Tech Officer.]

The Timetable app allows a schedule to be rendered based on room bookings. It also automatically adds links to common rooms. A schedule is represented by a Timetable object, which groups weeks with booking information together.

How do you input new schedule information?

Alright, this is a bit of a long process but it gets you the timetable image and links for free so it's worth it. The Timetable part of the Admin Panel has a bunch of different objects which we'll go through.

Timetable > Events contains the weekly events that we run. These consist of a name, time and sort index, as shown with the Wargames example below.

Change event

Description:	<input type="text" value="Wargames"/>
Date time line:	<input type="text" value="Mon 17:00-22:00"/>
Sort key:	<input type="text" value="1"/>

Note that a name alone does *not* have to be unique (but the name and date together should be unique), and in some cases you'll get events with the same name but different times - so make sure that you're working with the correct event when adding timetable information.

Timetable information for a given week is added using Timetable > Weeks. These weeks contain time information for when a given week occurs (i.e. when the start of the week is) and the room bookings for events in that week. Below is an example, which has two parts - the time information about the week (it is week 39 2022⁸, which starts on the 27th June), and each event that happens with its associated room. Rooms consist of the full code of the room, and optionally any special characters to give extra information about that room in a given timetable. (This will make more sense when we talk about timetables as a whole.)

⁸ Please note that you should be using actual week numbers - i.e. week 39 is term 3 week 10. If you use week numbers within the term, you'll get duplicates.

Delete

StartDate:

Number:

Year:
Academic year (use greater year, i.e. 18/19 is 2019)

BOOKINGS

Booking: 2022 week 39: Wargames : Mon 17:00-22:00

Room:

Event: Wargames : Mon 17:00-22:00 v ✎ +

Booking: 2022 week 39: MtG Draft : Tue 18:00-22:00

Room:

Event: MtG Draft : Tue 18:00-22:00 v ✎ +

Booking: 2022 week 39: Board Games : Wed 14:00-22:00

Room:

Event: Board Games : Wed 14:00-22:00 v ✎ +

Booking: 2022 week 39: CCGs : Thurs 17:00-22:00

Room:

Event: CCGs : Thurs 17:00-22:00 v ✎ +

When you get to adding information for a new term, you'll be creating many of these week objects (using the usual Add Week button in the top right). However, there is a helpful shortcut here - the Save as New button. When you click on an existing week, you can make the relevant changes to turn it into the next week (which may be only updating the week number and start date if the bookings are identical to the previous week), and then click Save as New to create a new week instead of overwriting the previous one. This means you don't have to reenter every room booking for every week.

How do you build a new timetable to show this information?

Once you have weeks with their event information included, you can build a timetable to show this information. Go to Timetables > Timetables, and click Add Timetable in the top right. You'll get a form like this:

Add timetable

Title:

Events:

Wargames : Mon 17:00-22:00
Online Wargames : Mon 17:00-22:00
Online Tabletop : Mon 17:00-22:00
MtG Draft : Tue 18:00-22:00
(Online) Wargames : Tue 17:00-22:00
Online Draft : Tue 18:00-22:00
Draft Type : Subject to Change

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Weeks:

2022 week 0
2022 week 1
2022 week 2
2022 week 3
2022 week 4
2022 week 5
2022 week 6

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Notes:

Active

Colour:

----- v

✏ + ✖

To quickly go through each of the fields:

- Title: what to display at the top of a given timetable.
- Events: what events to include in that timetable. So for a normal term, you'd select Wargames, MtG Draft, Board Games (the 2-10pm version), CCGs, RPGs and LARP.
- Weeks: what weeks the timetable should cover. So for a term 2 timetable, you'll need weeks 15-24.
- Notes: what to display below the timetable. If your room had a special character at the end, you might use this section to denote what the special character means.
- Active: whether to display this timetable on the Schedule page.
- Colour: what colour scheme to use. These can be created in Timetables > Colour Schemes, but you should probably just use the existing ones provided.

Once you've done this, your timetable will be accessible from <https://www.warwicktabletop.co.uk/timetable/x/>, where x is the ID of that timetable. It will also be accessible on the Schedule page if you've made it active.

How do you add room links?

Whenever you enter a room for which a room link has been prepared, it will render as a link to the Interactive Campus Map - which is useful for those that don't know the campus so well. However, you have to provide these links yourself. Go to Timetables > Room Links, which gives you a simple form mapping a room number to a link.

<https://warwick.ac.uk/about/visiting/maps/interactive/> lets you search for a room and gives a link to that room - find the room you need on the map, get the link and provide that link when creating the new Room Link. With this done, all timetable entries consisting solely of that room (and any special characters) will become a link to that room.

The room link renderer ignores some special characters (*^+-!&%\$#@=?) - use these to mark rooms with extra conditions on them⁹. The renderer also splits on commas, so if you enter multiple comma-separated rooms, it'll display a link for each.

How do you update the Google Calendars?

In short, log into Google Calendar as the society (I would recommend a Firefox Container for society logins so that you don't mix it with any personal Google logins) and update it by hand.

These Google Calendars are linked on the Schedule page, behind the Subscribe button. If you need to change the links, or add/remove new calendars, you can do this via Timetable > Google Calendars - the entries are each a calendar that is displayed, which can be updated or deleted.

How do you add special events to render in the Upcoming Events section of the homepage?

The website supports the rendering of special events via Timetable > Special Events. Currently, this only shows them wherever an Upcoming Events or Event Poster widget are used (see the Pages section for details on widgets).

To add these events, simply create a new Special Events entry via the Add Special Event button at the top right. You'll get a form like this:

⁹ <https://www.warwicktabletop.co.uk/timetable/8/> is a good example of these special characters in use.

Add special event

Title:	<input type="text"/>
Url:	<input type="text"/>
Room:	<input type="text"/>
Week:	<input type="text" value="1"/> 
Display date:	<input type="text"/> <small>The description of date and time to display</small>
Sort date:	<input type="text"/> Today  <small>The date to sort by, usually start date</small>
Hide date:	<input type="text"/> Today  <small>The date to hide this event after</small>
Poster:	<input type="button" value="Browse..."/> No file selected.

There are quite a few fields here, so let's go through them:

- Title: the name of the event, like Tabletop Weekend.
- URL: if there is a page associated with this event, provide its URL so that users can easily go to it.
- Room: where is it?
- Week: the week number of that event.
- Display date: a human-readable form of the date (like "Sunday Week 9").
- Sort date: the actual start date of the event. Used to determine how to sort the special events being shown.
- Hide date: the actual end date of the event. The special event will stop being shown *after* that day, so you do not need to add an extra day.
- Poster: a link to a beautiful poster to show off. This is done via the Event Poster widget, which is automatically enabled on the homepage (for non-mobile devices).

Once this is done, the event will automatically be shown in the aforementioned places until it is finished - at which point it will be hidden (but never deleted).

Users

[This section is useful to the Tech Officer, and to programmers considering the Warwick SU API.]

The Users app provides general things attributed with a user. It also provides achievements and talks to the Warwick SU API to verify if a website member has bought their membership.

How do you create an achievement?

To create a new achievement, you have to create an Achievement object in the Admin Panel. Go to Users > Achievements and click Add Achievement in the top right. You will get a form like this:

Add achievement

Name:	<input type="text"/>
Description:	<input type="text"/>
Trigger name:	<input type="text"/>
Image:	<input type="text"/>    
Fa icon:	<input type="text" value="fa-medal"/> <small>The name of the icon to use (including the fa- prefix)</small>
Icon set:	<input type="text" value="Solid Style (fas)"/>  <small>The fontawesome icon set this logo is from. (Please don't use Regular Style (far) icons: keep the style consistent)</small>
<input type="checkbox"/> Is hidden	
<input checked="" type="checkbox"/> Is fair	

This has a variety of fields, which we'll go through now.

- **Name:** what the achievement is called. This is shown to users directly.
- **Description:** a short description of the achievement, rendered below its name.
- **Trigger name:** how to give this achievement in code. Achievements are given in code via `give_achievement_once(member, trigger, achieved_date)`, where the trigger is the string you choose here.
- **Image:** this is a Visual Asset, discussed in the Assets and Gallery section. If this is not specified, you instead get a FontAwesome icon specified by the next two fields.
- **Fa Icon and Icon Set:** together these define the FontAwesome icon to use for this achievement. See the FontAwesome appendix for details on this.

- Is hidden: whether an achievement is hidden in the list of possible achievements. Achievements that a user hasn't got yet, but are not hidden, will be displayed as greyed out in the full achievements list. If it is hidden, it will not be displayed until a user has got it.
- Is fair: whether an achievement counts towards the total number of achievements. Previously, all achievements counted towards the total, so getting 100% was impossible. A few people complained loudly about this. As such, achievements which are very hard to get are now marked as "unfair". I don't know why this was such a point for people, but there you go.
Note that there are a few achievements about the age of your account. Current policy on whether such an achievement is fair is simply whether such an achievement is possible given that the website was made live in 2017. (So as of 1st Jan 2022, the 1-4 years of membership achievements are live.) You get to decide whether this policy continues! There's an argument for not making it continue given that most people will be at the uni for at most four years¹⁰, but I'll let you have this debate.

Fill these in, save and you're sorted.

How do you give an achievement to a user?

To manually give an achievement, you have to create an Achievement Award in the Admin Panel. Go to Users > Achievement Awards and click Add Achievement Award¹¹ in the top right. You'll get a form like this:

Add achievement award

Achievement:  

Member:  

Achieved at:

Date: Today 

Time: Now 

Simply select the achievement and the member to receive it, and it will be awarded. Note that no notification will be sent.

Possible Tech Officer improvement! You could make a form for doing this so that you don't have to go via the Admin Panel. It could also allow for bulk achievement awarding, and could actively notify users when they get an achievement.

¹⁰ says the PhD student who did their undergrad here

¹¹ or AAA for short, which is a noise I commonly make

How does membership verification work?

To the surprise of pretty much everyone, Warwick SU's website has an API. This is entirely undocumented, so I have very little knowledge of the full functionality of said API, but can comment on the API endpoint we use - the Members API endpoint. The code that works with this is in `users/ut i l s . py`¹², in which the Members API query gets us the University ID and email addresses of all members. This is why our verification process is to ask for your University ID and then email the associated email address to confirm ownership.

The API key needed should be made available to you by the previous Tech Officer - if you cannot get it, you can also find it within `l o c a l _ c o n f i g . py` on the website's server (so access it via PythonAnywhere). If the key suddenly expires, you'll have to talk to the SU. With this, it's just a case of querying the relevant URL (provided in `ut i l s . py`) with the API key appended. (Yes, this is probably not great for security, but this is controlled solely by the SU.)

¹² <https://github.com/WarwickTabletop/tgrsite/blob/main/users/ut i l s . py>

Votes

[This section is useful to anyone running an election.]

The Votes app allows for elections to be run. It has been tested vigorously both through multiple elections and explicit testing done after development was completed¹³.

Who can run an election? How can tampering be avoided?

Elections can be run and managed by anyone with the relevant permissions. These can be granted to individual users - see the Authentication and Authorisation section for details. (If you give someone permissions for a given election, do not forget to revoke said permissions afterwards.) The permissions are as follows. Note that these are provided as the code needed to check for the given permission, although the human-readable name can be inferred from these fairly easily.

- `PERMS.votes.view_election` allows a user to view the progress of an election. This only allows them to see the admin page available to those running an election, which solely contains the list of elections being run and the results of any election that is closed. **Nobody can see the progress of an election that is still open to voting - i.e. they cannot check how many votes have been cast.**
- `PERMS.votes.add_ticket` allows a user to distribute tickets for an election - if a user is given a ticket, they have a vote in this election. Note that the database strictly enforces uniqueness of tickets, so a user can only have one ticket per election and thus only gets one vote.
- `PERMS.votes.add_election` allows a user to create an election.
- `PERMS.votes.change_election` allows a user to update an election, including marking it as complete and changing its description.
- `PERMS.votes.add_candidate` allows a user to add candidates to an election.
- `PERMS.votes.change_candidate` allows a user to edit a candidate.

There are a few permissions that allow for the viewing of votes, but only because Django requires that every object either has a permission associated with it or is accessible by anyone, and unsurprisingly we didn't want the latter to happen. You should not give these permissions to anyone.

One awkward side-effect of Django's permissions system is that the Tech Officer is always a superuser on the website, so has every permission - meaning that they are technically able to edit elections regardless of whether they really should be. There is really no way to avoid this - any online voting system will be flawed by the fact that someone has to have access to the underlying database that the website uses, and in this case it is the Tech Officer that has this access.

¹³ See [How To Test a Voting System in Just Many Steps](#)

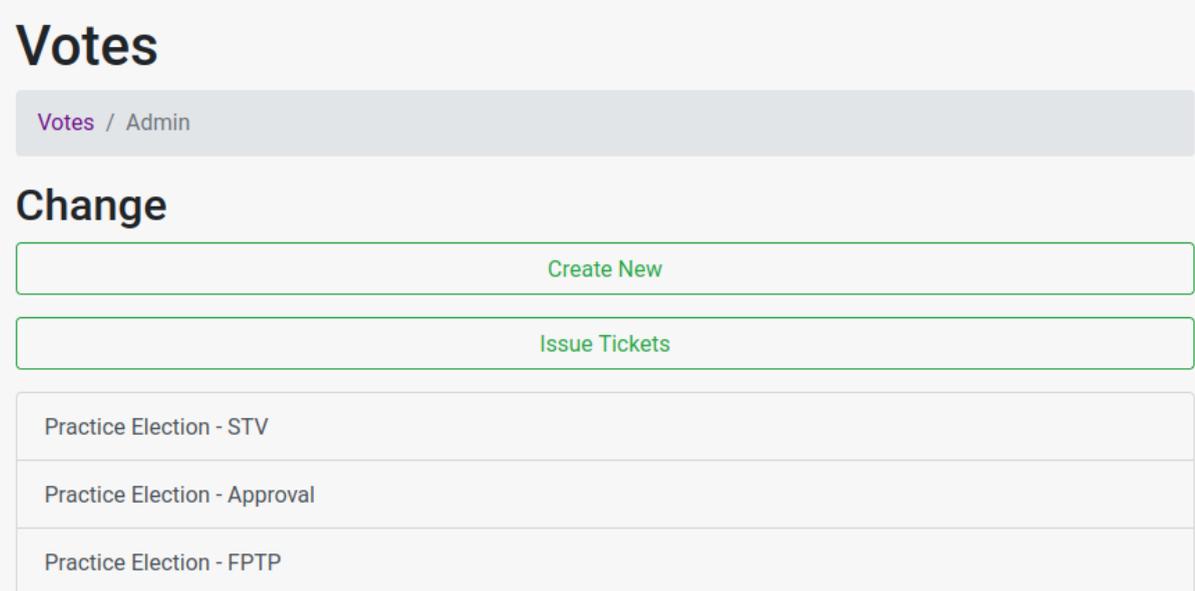
The voting system is designed to make it very hard to mess with the votes directly. It is impossible to link a vote to the person that made it - a user gets a ticket which is immediately deleted upon voting, and votes are stored as a UUID so the names are entirely random. The votes themselves are also readonly in the Admin Panel and cannot be edited in there nor on the website itself. The only way to modify the votes would be to edit them directly in the database (which as has been said, is unavoidable) - but the system is designed to make this as difficult as possible.

Fundamentally though, as Tech Officer you hold the same responsibilities as a Returning Officer during an election - in an in-person election a Returning Officer could tamper with the votes received but are trusted not to do that, and being Tech Officer during an online election requires the same trust and responsibility.

How do you run an election?

Running an election consists of four steps - creating the election, distributing tickets, opening the election and finally closing the election. Fortunately, this is all done via the website, so actively discourages using the Admin Panel for this.

First, you should go to the Admin section of Votes - this is done by going to the votes page and clicking Admin. This will give you a list of current elections, a list of results and a few buttons for managing elections. The below screenshot shows part of the Admin section on the website currently, with the usual practice votes set up.



The screenshot shows the 'Votes' Admin interface. At the top, there is a breadcrumb trail 'Votes / Admin'. Below this is a section titled 'Change' containing two buttons: 'Create New' and 'Issue Tickets'. Underneath these buttons is a table listing three practice elections:

Practice Election - STV
Practice Election - Approval
Practice Election - FPTP

Unsurprisingly, the Create New button creates a new election. This leads to a short form letting you set the type of election, its name and its description. After completing this, you'll need to add candidates to the election - go back to the list screenshotted above and click on the entry for that election. This is the page where you can update any descriptions of the election (in case of error) and add new candidates. Here is the example of part of the Practice STV Election:

Name

Practice Election - STV

Description

B I   **H**      

This is an election to practice voting in a STV election.

Vote type

Single Transferable Vote

Max votes

2

Ignored except in Plurality. Number of candidates selectable per vote

Seats

1

Ignored except in STV. Number of people who can win

Open

[Save](#)

Candidates

[Create New](#)

Test Candidate 1

The first candidate.

Click on Create New to add each candidate. Each candidate has a name, short description and status (whether they are standing in the election or are withdrawn, candidates should always start as Standing) - fill these in to create the candidate. **NOTE:** remember to create a candidate representing RON in an STV election.

Once the election is set up with all of the candidates, you need to distribute tickets for the election. Go back to the main Admin page and click Issue Tickets. You'll be given a list like this:

Votes

[Votes](#) / [Admin](#) / [Tickets](#)

Create Ticket by...

Verified Members

[ID list](#)

[Verification Date](#)

All Members

[All](#)

[Username List](#)

[Specific Member](#)

For AGMs and similar, you'll want the ID List option¹⁴ - download the SU membership list two weeks before your AGM and use that to get the required IDs. For other votes, you'll likely want to just give everyone a ticket with the All option, but you can also give a specific list of usernames a ticket with the Username List option.

When you select the relevant option, you'll be given a form to fill in. This allows you to select which elections to give tickets to members to, alongside details relevant to that method such as the target date or list of usernames. Once you've done this, all the relevant members will have their tickets and you are ready to open the election.

Only once you have done all of this, you can open the election. This is very simple - go to the election's edit page from earlier, set the Open checkbox to checked and save. The election will be available to all that can vote on it on their individual Vote pages.

Once it is time to close the election, uncheck the Open checkbox and save. This will automatically calculate the results, which will be available at the bottom of the main Admin page in a list like the one below.

Results

Practice Election - STV
Practice Election - Approval
Practice Election - FPTP

Click on the election and it will give you a full breakdown of results for you to publish!

WARNING: for STV elections, when you click on an election's results for the first time, it calculates the results and then shows them to you. On future clicks, it will *not* recalculate them - so if any more votes have come through, they will not be counted. As such, only look at STV results once the election has been completed. The results will only be available while the election is closed so you should be fine, but if you accidentally closed and reopened an election then you may have this problem. (See the next section for more details on this.)

Phew, thank goodness that's done. Your last job is to clean up once the election is final (7 days after the results are announced, assuming no complaints) - you must delete all tickets left in the database. (As long as you're not running elections starting on different days within fewer than seven days of each other, you can just delete every ticket in the Admin Panel.) Do **not** delete any votes.

¹⁴ Don't use the Verification Date option, as that relies on people actually verifying their membership on time...

How do I rerun an STV election in the case of a candidate withdrawing after the fact?

In very rare cases, you'll need to rerun the election due to someone stepping down (e.g. if they won multiple roles). To do this, you need to delete the associated STV results object so that the website knows to recalculate the results.

WARNING: The reason for the results only being calculated once is that the STV process naturally includes a little bit of randomness in some cases (breaking ties) so rerunning the process each time would possibly change the result. As such, do not perform this process unless you are sure you want to rerun the election.

First, update the election as in the previous section to mark candidates who are no longer running as withdrawn. Do not delete the candidate who has withdrawn - click on the edit button associated with the candidate, which is the left button pictured. Then change their state using the dropdown and save.

Once the candidates list is correctly updated, go to Votes > STV Results, and click on the relevant election. Scroll down to the bottom and click the delete button. Then go back to the results page for the election (as detailed in the previous section) and it will recalculate the results for you.

Test Candidate 1

The first candidate.



Website Settings

[This section is useful to the Tech Officer.]

The Website Settings app lets certain settings be defined in code, which can be set using the admin panel (thus avoiding small code changes).

What do the website settings currently correspond to?

The website has two flavours of settings: String Properties, which are used for shorter bits of text such as single URLs; and Text Properties, which are used for longer-form paragraphs of text. At time of writing, we have the following settings.

String Properties, which can be accessed at Website Settings > String Properties:

- `notifications_webhook_url`: the webhook URL that the website pings to send notifications to #website-notifications on the Discord server.
- `facebook_page`: the URL of our Facebook page, which is rendered in the footer of the website.
- `su_page`: the URL of our SU page, which is also rendered in the footer.
- `bug_tracker`: the URL that the Report a Bug link on the sidebar should point to.
- There are a few others, but I do not think they're in use. If you confirm they're not in use, delete them!

Text Properties, which can be accessed at Website Settings > Text Properties:

- `exec_history`: the text that appears at the top of the Exec page (in Markdown), which nominally points to the Exec History page.
- `newsletter_template`: the text that fills the newsletter editor when a new newsletter is created.
- `newsletter_pitch`: text that appears when you sign up for a new account telling you that you can subscribe to the newsletter.
- `tutorial_text`: the HTML rendered within the tutorial modal when someone is shown the tutorial (only happens once).
- `tutorial_css`: the CSS used on that tutorial modal.

If you add more, please document them here.

Authentication and Authorisation

We heavily rely on Django's inbuilt authentication and authorisation, which is pretty sensible given that professionals will have written and verified that. This does however mean that a few things are in the Authentication and Authorisation part of the Admin Panel, which isn't tied to an app we've created. In this section, we look at the tasks you might want to do relating to Authentication and Authorisation.

What's the difference between users and members on the Admin Panel?

Due to relying on Django's stock Users, we need a way of associating these users with extra information that isn't needed in the general case. As such, there are two different objects for you to worry about when referring to individuals on the website. Users are Django's authorisation objects, so have a few details such as email, but also deal with permissions and passwords. You can therefore reset passwords, update associated email addresses, and set permissions of a user from Authentication and Authorisation > Users. Members are our objects which associate users with extra information - such as their bios, pronouns and discord usernames. These can be updated in Users > Members - each Member has an equivalent user defined by the `equiv_user` field.

If you would ever like to associate more information with someone using the website, you should update the Members model, rather than attempting to mess with the Users one.

How do you ban or otherwise lock out a user?

Sometimes, we get spambots on the website. This is fortunately very rare due to our amazing bespoke Captcha system called CAT¹⁵, but still, you may be in the place where you want to prevent a user from further accessing the website.

The simple solution is to deactivate their account via the Admin Panel. Django allows for deactivation by simply unchecking the Active checkbox of a given User - go to Authentication and Authorisation > Users, click on the offending user's entry and uncheck that box..

You can attempt to delete a user by using the delete button in Authentication and Authorisation > Users, **but this is very unlikely to work** - a fair number of models in the application have SQL ON DELETE PROTECT constraints, so will require you to manually delete all associated objects in order to delete the user. As such, prefer deactivating a user or locking them out over deletion.

¹⁵ heh heh I made this one - Finnbar

How can you help a user who is unable to reset their password but needs to?

If a user loses access to their email account (for example, because they were using a University email address and have since graduated) and needs to reset their password, you may do this for them. It goes without saying that you should be extremely careful here - make sure you are **certain** that the person contacting you is the owner of the account.

Go to the user's entry in Authentication and Authorisation > Users, and go to the password entry. It will look something like this, but with less censoring. Note that the raw password is **never** available to you.

Password: algorithm: [REDACTED] iterations: [REDACTED] salt: [REDACTED] hash: [REDACTED]
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).

Click on the "this form" link, and choose a new password for them. This should be some temporary password like "changemenow", so that they can change it themselves to something more secure. You should also scroll down to the Email Address entry and change that accordingly.

This technique can also be used to lock a spambot out of their own account, but in general you should prefer deactivating the account. It also goes without saying that you should not abuse this power to lock out people randomly.

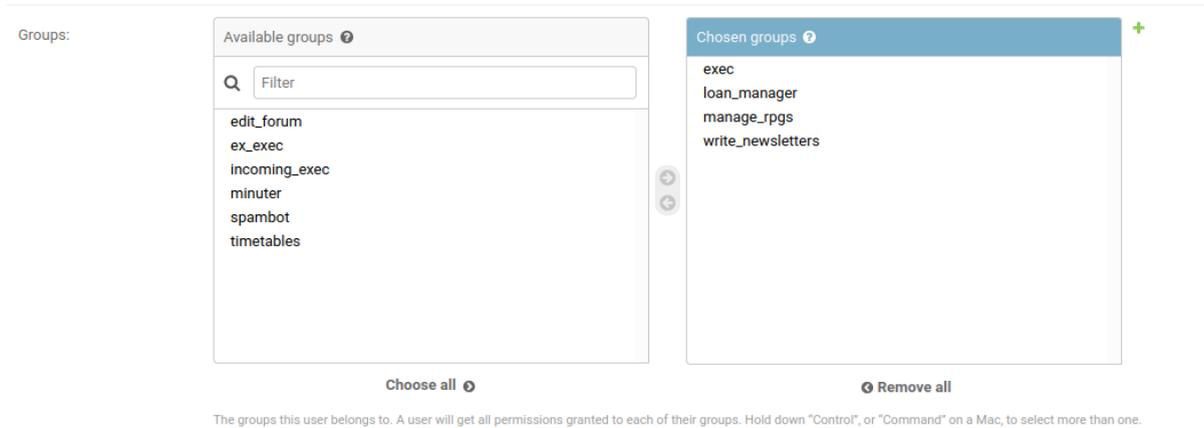
How do you give permissions to a certain user, for example during handover?

In general, you should only have to deal with adding groups to a user. Groups (available at Authentication and Authorisation > Groups) are collections of permissions that are grouped together for easier management. This means that rather than having to give a user the ten permissions that they need to do something, you can just give them the single group that does this. Currently, the following groups are present on the website:

- `edit_forum`, which allows the editing of forum posts not owned by a user. This is usually given to the Vice President to help with cleanup on rare occasions.
- `ex_exec`, which only gives the ability to view in-progress newsletters (useful during term 3 handover) and gives them a nice badge on the website.
- `exec`, which gives a variety of handy permissions for general exec use (such as adding entries to inventories and reading in-progress newsletters) and gives them a nicer badge on the website.
- `loan_manager`, which allows loans to be approved and marked as taken. This is usually given to the Board Games Rep.

- `manage_rpgs`, which allows games not owned by this user to be edited, e.g. to mark them as complete. This is usually given to the RPGs Rep.
- `minuter`, which allows for the creation of minutes. This is usually given to the Vice President.
- `spambot`, which is just given to identify who is a spambot.
- `timetables`, which allows a user to update the timetables.
- `write_newsletters`, which allows a user to write and publish newsletters, as well as update the newsletter template. This is usually given to the Co-Op Officer and President.

To give a user access to a group and the permissions within it, go to Authentication and Authorisation > Users, scroll down to the Groups part and double click on the relevant group on the left hand side to add it. You can also double click on a group on the right hand side to remove it. See below for an example.



If a group does not have the permissions needed, and you don't want to create a new group for some reason (possibly because you're assigning a single permission to a single user temporarily), you can do the same with individual permissions. This is directly below the Groups selector.

As well as groups and permissions, there are two special permissions created by Django that some incoming exec need to be given (and outgoing exec should have removed). These are two of the three checkboxes present in the top of the permissions section: **staff status**, which allows a user to access the Admin Panel at all; and **superuser status**, which gives a user every permission. Staff status should be given to all exec so that they can access the parts of the Admin Panel that they need access to. The Tech Officer should be the only superuser.

Miscellaneous Questions

[This section is mainly for the Tech Officer, although briefly touches on daily tasks which may be of interest to developers.]

This section is for questions that don't really belong under one website app.

How are the daily summary emails sent out, and how can I create other daily tasks?

The daily summary emails are dealt with via a Django admin command called `daily`, whose source is present in the `templatetags` app¹⁶. Currently, this does a few jobs: sending out the daily summary emails, deleting empty messaging threads and updating membership verifications¹⁷. If you need anything else to be run once per day, simply add to this command.

On the PythonAnywhere side, this task is managed within the Tasks tab. This is currently set to call the `daily` command at 6 minutes past midnight each day. The only restriction is that the whole task shouldn't take more than 33 minutes of compute time, as you'll end up in the tarpit (documented here: <https://help.pythonanywhere.com/pages/WhatAreCPUSeconds>).

Why are emoji allowed in some parts of the website but not others? Can I change this?

The simple answer is that this is a very awkward database issue. By default, the database tables use an encoding that does not include emoji so actively drops any emoji you include. You need tables to have the encoding `utf8mb4_unicode_ci`. Most tables should have this already due to Anna going through the relevant tables and updating the entries herself, but if you find a table without the correct encoding and want it to have said encoding, you need to follow the below steps.

In PythonAnywhere, the only way to do this is via directly entering SQL commands - go to PythonAnywhere, click on Consoles and start up a new MySQL prompt.

DISCLAIMER: YOU ARE NOW DIRECTLY EDITING THE DATABASE. Please make sure you are typing these commands correctly. The only reason we're doing this at all is that there's no way to change this within Django.

¹⁶ <https://github.com/WarwickTabletop/tgrsite/blob/main/templatetags/management/commands/daily.py>

¹⁷ To summarise this process briefly, just in case it is useful to you: this pulls from the SU API, which produces an XML linking uni IDs to email addresses, then updates existing Membership objects to be active iff they are present in that XML.

This prompt allows you to directly run commands on the database. First, you will likely want to run `show tables`; in order to discern what tables you have access to. Note that Django's representation of tables is very high-level, so you won't just see a table per model in the code - you'll also see all of the join tables that Django uses in the background to implement foreign key constraints and similar. You shouldn't need to update the encoding on those tables, since they just contain keys from other tables.

Make sure you've found the table you want, and then run the following command, substituting `table_name` for the relevant table.

```
ALTER TABLE table_name CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

Once you've done this, the table should be able to accept emoji. Close the MySQL prompt. Don't open it again unless you know what you're doing (or forgot to update a table).

How can I increase the character limits on certain fields?

Annoyingly, this can only be done via code. Fortunately it is pretty easy. First, go into `models.py` of the relevant app - e.g. if you wanted to increase the maximum length of titles of forum posts, you'd go to `forum/models.py`.

Look for the field which sounds like the correct one (or ask someone who knows the code better if you're stuck) - it will be set to something like `models.CharField(max_length=128)`. Simply increase the `max_length` number, and you're good to go! (After making migrations, but the `website update.sh` script will do that for you.) If there is no `max_length` number, there is no cap on the length of values of that field.

That being said, maximum lengths are enforced for a reason - for example, overly long titles look bad on most screens. Furthermore, you really don't want random users to be able to fill the website with Bee Movie script-length spam. In general, anything that's user-facing should have a reasonably short character limit, while anything that's only exec-facing can have much more lenient limits or no limits at all.

How can I change the website's (CSS) styling?

The website uses a modified version of Bootstrap for its styling, which is present in the `bootstrap` folder of the repo. These are written in Sass, which is a more advanced version of CSS. This does however require you to compile SCSS files to their CSS counterparts when you edit them, since browsers cannot naturally read CSS.

To do this, you should use the provided `buildbootstrap.sh` script provided in the `bootstrap` folder. This requires `sassc` (the Sass compiler, available as an installable program); and `postcss-cli`, `cssnano` and `autoprefixer` (installed via NPM). Whenever you edit a file in

bootstrap, run `buildbootstrap.sh`, setting the source to the file you just edited and the target to its equivalent CSS file in `static_resources`. For example, if you updated `tgrsite_local.scss`, you would run the following from the bootstrap directory (indentation and newline added for legibility):

```
./buildbootstrap.sh tgrsite_local.scss  
  ../static_resources/tgrsite_local.css
```

With that, you're ready to go!

Command Line Tools

The website comes with three command line tools that may be of use to you. These are as follows:

- `age_achievements`, which was used to give older users the achievements for having accounts of a certain age. This is no longer needed, but serves as a gentle introduction to writing your own command line tools.
- `deadname_remove`, which is properly exciting - when given a deadname to replace with a real name (provide these as arguments to the script), it performs an interactive find-and-replace within all of the newsletters so you can substitute the relevant instances of the deadname with a real one. Here's a screenshot of it in action:

```
Matches found in Body of newsletter #10 (https://www.warwicktabletop.co.uk/newsletters/10)!  
  
**Magic: The Gathering Chaos Draft (Tuesday 18:00 to 22:00, B2.02 Science Concourse)**  
*"Mythic rares bent after riffle shuffle..."*  
For the Magic Draft this week, we're holding a chaos draft! This me  
Type 'yes' to substitute ares for goose.  
Type 'no' to not perform this substitution.  
Choice: █
```

It's very cool, and frankly a shame that you won't get to use it much. But it's far better than the manual process I was doing for deadname removal before.

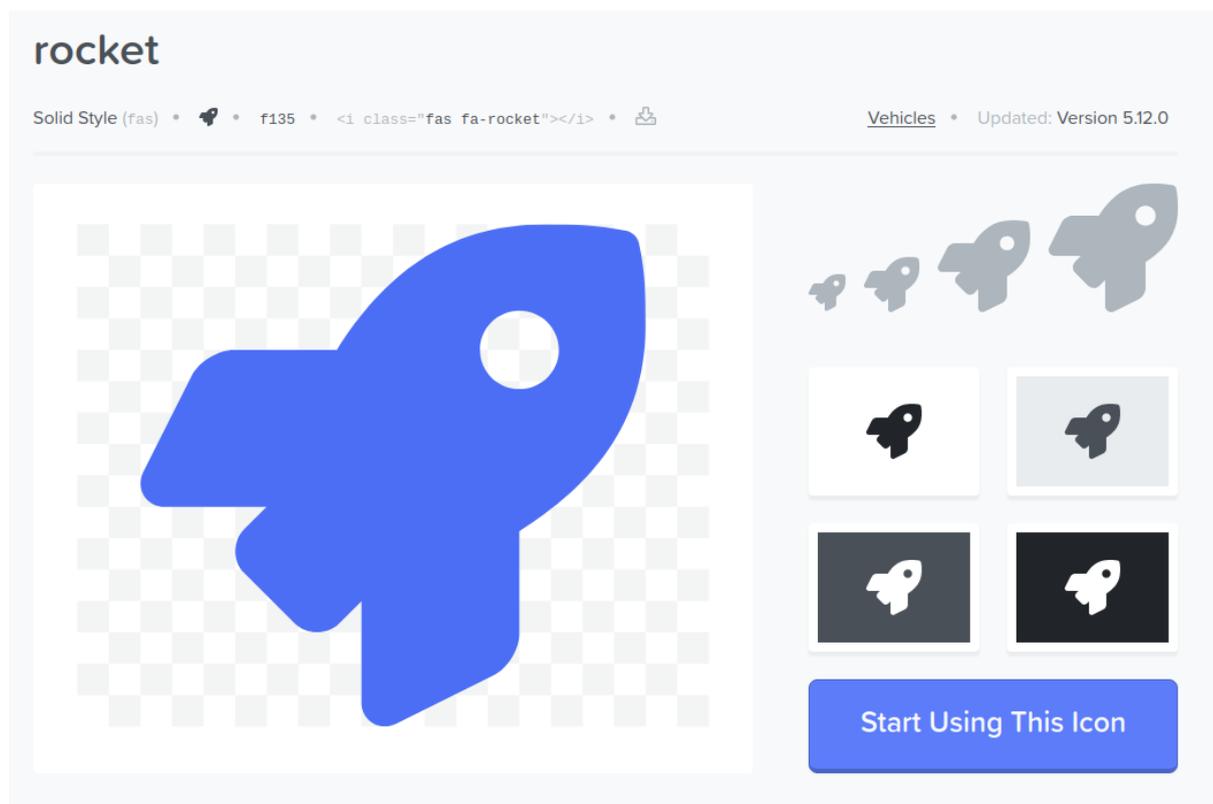
- `message_censor`, which allows you to censor all messages that you've sent containing sensitive information such as a password.

Developers may want to consult Django's documentation on custom admin commands (<https://docs.djangoproject.com/en/4.0/howto/custom-management-commands/>) if they'd like to write their own!

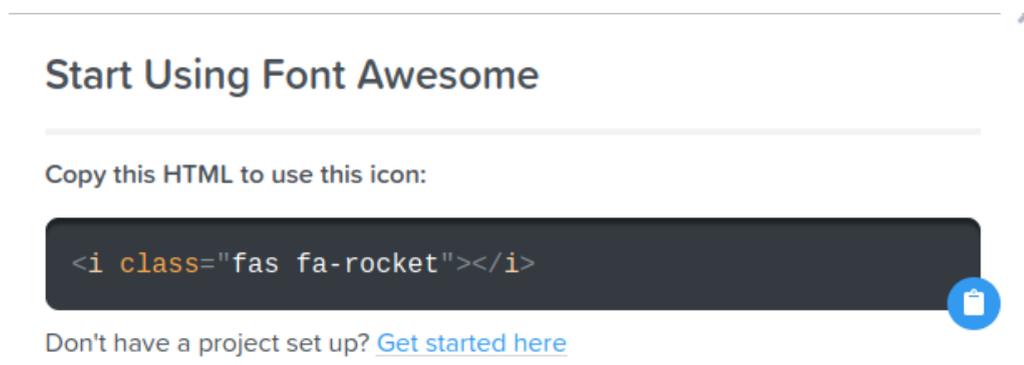
Appendices

FontAwesome

FontAwesome 5 is used throughout the website to render helpful icons, such as on the navbar and in achievements. <https://fontawesome.com/> allows you to search through the icons to find one that you want - for example, if you wanted a rocket icon you might find one like in the screenshot below.



If you click Start Using This Icon, you'll get the two bits of information you'll need to use this icon in the navbar.



Here the icon is fa-rocket and the icon set is fas.